ABSTRACT

Title of Dissertation: SCALABLE METHODS FOR ROBUST MACHINE LEARNING

Alexander Jacob Levine
Doctor of Philosophy, 2023

Dissertation Directed by: Professor Soheil Feizi
Department of Computer Science

In recent years, machine learning systems have been developed that demonstrate remarkable performance on many tasks. However, naive metrics of performance, such as the accuracy of a classifier on test samples drawn from the same distribution as the training set, can provide an overly optimistic view of the suitability of a model for real-world deployment. In this dissertation, we develop models that are *robust*, in addition to performing well on large-scale tasks.

One notion of robustness is *adversarial* robustness, which characterizes the performance of models under adversarial attacks. Adversarial attacks are small, often imperceptible, distortions to the inputs of machine learning systems which are crafted to substantially change the output of the system. These attacks represent a real security threat, and are especially concerning when machine learning systems are used in safety-critical applications.

To mitigate this threat, certifiably robust classification techniques have been developed. In a certifiably robust classifier, for each input sample, in addition to a classification, the classifier also produces a certificate, which is a guaranteed lower bound on the magnitude of any perturbation

required to change the classification. Existing methods for certifiable robustness have significant limitations, which we address in Parts I and II of this dissertation:

1. Currently, randomized smoothing techniques are the only certification techniques that are viable for large-scale image classification (i.e. ImageNet). However, randomized smoothing techniques generally provide only high-probability, rather than exact, certificate results. To address this, we develop **deterministic** randomized smoothing-based algorithms, which produce exact certificates with finite computational costs. In particular, in Part I of this dissertation, we present to our knowledge the first deterministic, ImageNet-scale certification methods under the ℓ_1, ℓ_p (for $p < 1$), and "ℓ_0" metrics.

2. Certification results only apply to particular metrics of perturbation size. There is therefore a need to develop new techniques to provide provable robustness against different types of attacks. In Part II of this dissertation, we develop randomized smoothing-based algorithms for several new types of adversarial perturbation, including Wasserstein adversarial attacks, Patch adversarial attacks, and Data Poisoning attacks. The methods developed for Patch and Poisoning attacks are also deterministic, allowing for efficient exact certification.

In Part III of this dissertation, we consider a different notion of robustness: test-time adaptability to new objectives in reinforcement learning. This is formalized as goal-conditioned reinforcement learning (GCRL), in which each episode is conditioned by a new "goal," which determines the episode's reward function. In this work, we explore a connection between off-policy GCRL and knowledge distillation, which leads us to apply Gradient-Based Attention Transfer, a knowledge distillation technique, to the Q-function update. We show, empirically and theoretically, that this can improve the performance of off-policy GCRL when the space of goals is high-dimensional.

SCALABLE METHODS FOR ROBUST MACHINE LEARNING

by

Alexander Jacob Levine

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2023

Advisory Committee:
Professor Soheil Feizi, Chair/Advisor
Professor Behtash Babadi
Professor Furong Huang
Professor Jia-Bin Huang
Professor Abhinav Shrivastava

Dedication

To my parents.

Acknowledgments

Firstly, I would like to thank my advisor, Soheil Feizi, for guiding me through my research for the past five years. Professor Feizi provided me with extensive advice and support, but also gave me wide latitude to explore my own interests and set my own course throughout my PhD. He particularly helped me improve my skill in academic writing, through his precise attention to detail and deft intuition for how to frame ideas in a way that they will be understood.

I would also like to thank the co-authors with whom I have collaborated over the course of my PhD: Aounon Kumar, Wenxiao Wang, Sahil Singla, Priyatham Kattakinda, Jiang Liu, Wei-An Lin, Chun Pong Lau, Professor Tom Goldstein, and Professor Rama Chellappa, and additionally my dissertation commitee members, Behtash Babadi, Furong Huang, Jia-Bin Huang, and Abhinav Shrivastava.

I also owe a debt of gratitude to the many labmates and office friends with whom I had many interesting, insightful, and occasionally productive conversations over the five years, including Phil Pope, Michael Curry, Mazda Moayeri, Saptarashmi Bandyopadhyay, Chenghao Deng, Aya Abdelsalam Ismail, among many others.

I particularly would like to thank my housemates, Niall Williams, Connor Baumler, Manasi Shingane – and especially Marina Knittel and Alex Rowden, with whom I have lived for all of the past four years. Without all of you, getting through graduate school may have been *possible*, but it would certainly have been less pleasant.

I would like to thank Kristen Michaelson for her tremendous support over the last two years. Finally, I would like to thank my family, who made me the person who I am today.

This project was supported in part by NSF CAREER AWARD 1942230, HR 00111990077, HR 001119S0026-GARD-FP-052, HR 00112090132, NIST 60NANB20D134, ONR grant 1337-0299, ONR YIP award N00014-22-1-2271, the Simons Fellowship on "Foundations of Deep Learning," and the University of Maryland Ann G. Wylie Dissertation Fellowship.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

1.1 Motivation

Over the past decade, remarkable improvements have been made in the performance of machine learning models on a variety of tasks. However, naive metrics of performance, such as the raw accuracy of a classifier on test samples drawn from the same distribution as the training set, fail to capture a complete picture of the real-world usability of a model. This dissertation is centered on designing models which are *robust*, in addition to being performant on large-scale tasks.

Several notions of "robustness" have been proposed in recent years. *Adversarial* robustness is a broad and widely-studied field which characterizes the *worst-case* behavior of machine learning systems under small input perturbations [7, 8, 9]. Researchers study the threat of *adversarial attacks*, in which a malicious actor corrupts the input to a system in an imperceptible or minimally-perceptible way, such that the behavior of the system is greatly impacted. (Concretely, for example, an attacker may make a human-imperceptible change to an image, which causes an image classifier to misclassify the image, when it would otherwise be classified correctly.) These adversarial attacks take advantage of the fact that standard deep neural network architectures are highly sensitive to small changes of their inputs [10]. These adversarial distortions can represent a real security threat: they are especially concerning when machine learning systems are used in

safety-critical applications, such as in medical applications or in self-driving vehicles (e.g., self-driving systems could be disrupted by putting small stickers on traffic signs, to cause the sign to be recognised incorrectly [11]).

While practical defenses against adversarial attacks have been proposed [3, 9, 12, 13, 14, 15, 16, 17], these can be rendered ineffective by improved adversarial attacks [18, 19, 20, 21]. In order to overcome this, one area of active research is the design of *certifiably-robust* classifiers where, for each input $\boldsymbol{x}$, one can compute a magnitude ρ, such that *all* perturbed inputs $\boldsymbol{x}'$ within a radius ρ of $\boldsymbol{x}$ are guaranteed to be classified in the same way as $\boldsymbol{x}$. Typically, ρ represents a distance in an ℓ_p norm: $\|\boldsymbol{x} - \boldsymbol{x}'\|_p \leq \rho$, for some p which depends on the technique used. A variety of techniques have been proposed for certifiably robust classification [22, 23, 24, 25, 26, 27]. Certifiably-robust classifiers for non-ℓ_p threat models, such as geometric transformations [28] and patch attacks [4, 29, 30, 31], have also been introduced.

Certification techniques based on *randomized smoothing* [1, 5, 32, 33, 34, 35, 36], are, at the time of writing, the only robustness certification techniques that scale to tasks as complex as ImageNet classification (See Li et al. [37] for a recent and comprehensive review and comparison of robustness certification methods.) In these smoothing methods, a "base" classifier is used to classify a large set of randomly-perturbed versions $(\boldsymbol{x} + \epsilon)$ of the input image $\boldsymbol{x}$ where ϵ is drawn from a fixed distribution. The final classification is then taken as the plurality-vote of these classifications on noisy versions of the input. If samples $\boldsymbol{x}$ and $\boldsymbol{x}'$ are close, the distributions of $(\boldsymbol{x} + \epsilon)$ and $(\boldsymbol{x}' + \epsilon)$ will substantially overlap, leading to provable robustness. Salman et al. [33] and Levine et al. [38] show that these certificates can in some cases be understood in terms of Lipschitz continuity, where the *expectation* of the output of the base classifier (or a function thereof) over the smoothing distribution is shown to be Lipschitz. However, randomized

smoothing techniques have some limitations:

- Smoothing techniques generally provide only high-probability, rather than exact, certificate results. This is a consequence of the fact that the final classification is made using Monte-Carlo samples of ϵ from the noise distribution. Furthermore, larger amounts of computational power are needed to produce higher-probability (i.e., less likely to be incorrect) certificates, potentially leading to environmental and sustainability concerns.

- Certification results only apply to particular metrics of perturbation size. (For example, ℓ_1 or ℓ_2 metrics.) There is therefore a need to develop new techniques to provide provable robustness against different types of attacks.

Parts I and II of this dissertation are focused on partially addressing these limitations.

Apart from *adversarial* robustness, other notions of robust machine learning have also been considered. This includes robustness to natural distributional shifts, where the "test" data (i.e., the data that the model is applied to at deployment) is drawn from a significantly different distribution than the training data — due to natural circumstances, rather than a malicious adversary [39, 40, 41]. A related concept, in reinforcement learning, is "zero-shot generalization", where the environment that a reinforcement learning agent is tested/deployed in is significantly different from the environment(s) in which the agent was trained [42, 43, 44]. However, the defining assumption of "zero-shot" generalization, which is that there are *unknown* changes between training and test time environments, can be made looser in many practical applications. In particular, the field of goal-conditioned reinforcement learning (GCRL) models situations where only the *objectives* that an agent needs to accomplish may vary at test time and furthermore where the new desired objective is *provided explicitly* to the agent at test time [45, 46, 47, 48, 49]. This can be

thought of as *robustness to deployment-time changes in an RL system's requirements*. Concretely, in a goal-conditioned RL environment, in each episode, there is an provided "goal" that specifies the reward function during that episode but does not affect the environmental dynamics.

Various techniques have been proposed for GCRL, such as automatic curriculum generation and goal relabeling [49, 50]. In Part III of this dissertation, we propose a novel method for GCRL that is designed to improve performance in environments where the space of possible goals is high-dimensional, in order to allow for greater scalability.

1.2 Contributions

1.2.1 Part I: Large-Scale, Deterministic Robustness Certificates for ℓ_p $(p \leq 1)$

In Part 1, we develop a framework for **ImageNet-scale, deterministic** certification for ℓ_p threat models with $p \leq 1$, based on a modified form of "randomized smoothing." Recall that for $p > 0$, the ℓ_p distance is defined as:

$$\|\boldsymbol{x} - \boldsymbol{y}\|_p := \left(\sum_{i=1}^{d} |x_i - y_i|^p \right)^{1/p}, \tag{1.1}$$

while the "ℓ_0" metric is typically defined as

$$\|\boldsymbol{x} - \boldsymbol{y}\|_0 := \sum_{i=1}^{d} \mathbf{1}_{(x_i \neq y_i)}, \tag{1.2}$$

where $\mathbf{1}_{(\cdot)}$ denotes an indicator function; in other words, $\|\boldsymbol{x} - \boldsymbol{y}\|_0$ denotes the number of indices i at which x_i and y_i differ.[1]

First, in Chapter 2, we develop a randomized smoothing certification technique for the ℓ_0 threat model. Our technique, *randomized ablation*, works by *ablating* (i.e., removing) randomly-selected features from the input sample $\boldsymbol{x}$, rather than introducing random noise. This results in large improvements over the previous state-of-the-art ℓ_0 randomized smoothing technique [1] in terms of the magnitude of the produced certificates. For example, on ImageNet images, we are able to certify the median sample against corruptions of up to $\rho = 16$ pixels; while [1] can certify, in median, against corruptions to only *one* pixel.

Next, in Chapter 3 we develop a *deterministic* randomized smoothing-based algorithm for the ℓ_1 metric, which produces exact, rather than high-probability, certificates with finite computational costs. Our method is the first deterministic certification method for this metric that scales to ImageNet, while significantly outperforming prior randomized methods. The proposed method, Deterministic Smoothing with Splitting Noise (DSSN) uses a novel *non-additive* distribution of smoothing noise. To develop DSSN, we first develop SSN, a randomized method which involves generating each noisy smoothing sample by first randomly dividing the input space and then returning a representation of the center of the subdivision occupied by the input sample $\boldsymbol{x}$. In contrast to the prior state-of-the-art for the ℓ_1 metric, uniform additive smoothing [1, 51], the SSN certification does not require the random noise components used to be independent. Thus, smoothing can be done effectively in just one dimension and can therefore be efficiently derandomized for quantized data (e.g., images).

[1] Note that, unlike in the $p < 0$ case, the definition of the "ℓ_0" distance does not require that the individual elements x_i be scalars. In particular, on image data, the "ℓ_0" threat model can refer to the number of pixel positions at which two images differ, where each pixel position contains multiple color channels [8]. In our experiments in Chapter 2, we focus on image data, using the "pixel position" definition of the ℓ_0 threat model.

To the best of our knowledge, this is the first work to provide deterministic "randomized smoothing" for a norm-based adversarial threat model while allowing for an arbitrary classifier (i.e., a deep model) to be used as a base classifier and without requiring an exponential number of smoothing samples. On CIFAR-10 and ImageNet datasets, we provide substantially larger ℓ_1 robustness certificates compared to prior works, establishing a new state-of-the-art. The determinism of our method also leads to significantly faster certificate computation.

Finally, in Chapter 4, we extend the method proposed in Chapter 3 to ℓ_p distances for $p \in (0, 1)$. This represents the first certification method of any kind for these threat models, and also provides deterministic certification at ImageNet scale. Our technique works by producing a deterministically-smoothed classifier that is globally Lipschitz with respect to the ℓ_p^p metric, defined as:

$$\|\boldsymbol{x} - \boldsymbol{y}\|_p^p := \sum_{i=1}^{d} |x_i - y_i|^p, \tag{1.3}$$

for any $0 < p < 1$. However, our method is in fact even more general: we can construct classifiers which are globally Lipschitz with respect to any metric defined as the sum of concave functions of each feature. Empirically, we demonstrate that our proposed guarantees are highly non-vacuous when certifying under ℓ_p ($p < 1$) threat models, compared to the trivial solution of using the technique from Chapter 3 directly and applying norm inequalities. Additionally, in Section 4.5, we show that the proposed method can in fact be extended to the ℓ_0 metric, providing deterministic certificates comparable to those proposed in Chapter 2 – in fact, this extension can be considered as a straightforward derandomization of the algorithm presented in Chapter 2.

Note that the experiments of these chapters focus specifically on image classification tasks; however, the techniques developed can be applied more broadly. In particular, the results for the

ℓ_0 metric apply straightforwardly to any vectorized data, while the deterministic smoothing techniques for ℓ_p $p \in (0, 1]$ distances apply to vectorized data that exist in *bounded, quantized* spaces. This includes bitmapped images (where pixels have values in the set $\{0, \frac{1}{255}, \frac{2}{255}, ..., \frac{254}{255}, 1\}$) but also other domains, such as video data.

1.2.2 Part II: Scalable Robustness Certificates beyond ℓ_p Threat Models

In Part 2, we propose extensions to randomized smoothing which allow for certification under non-ℓ_p threat models.

In Chapter 5, we develop a certifiably robust image classifier that is robust under distortions in the Wasserstein metric (or "earth-mover distance"), where local shifts in pixel intensity between nearby pixels are considered smaller distortions than longer-range shifts. Note that Wasserstein-bounded adversarial *attacks* had been previously proposed by [2]: our work represents the first provable defense of any kind under this threat model. We develop this certificate by considering the space of possible "probability flows" between images, and representing this space such that Wasserstein distance between images is upper-bounded by ℓ_1 distance in this flow-space. We can then apply existing randomized smoothing certificates for the ℓ_1 metric.

In Chapter 6, we propose a certifiably robust defense against "patch" adversarial attacks [52, 53, 54, 55]. In this threat model, which is specific to the image domain, the attacker is free to make arbitrary changes within a single, bounded region of an image of limited size (i.e., within a square patch). Note that although the attacker may only affect a single patch per image, the patch location can be chosen by the attacker for each image. Patch attacks can be regarded as an abstracted model of *physical* adversarial attacks [11, 52], where an attacker makes an adversarial

pattern visible in a real-world environment (e.g., by attaching a small sticker to a road sign) in order to disrupt computer vision systems.

Compared to the previous state of the art patch certification method proposed by Chiang et al. (2020) [4], our method can be trained significantly faster, achieves high clean and certified robust accuracy on CIFAR-10, and provides certificates at ImageNet scale. For example, for a 5×5 patch attack on CIFAR-10, our method achieves up to around 57.6% certified accuracy (with a classifier with around 83.8% clean accuracy), compared to at most 30.3% certified accuracy for the prior method (with a classifier with around 47.8% clean accuracy). Our results therefore effectively established a new state-of-the-art of certifiable defense against patch attacks on CIFAR-10 and ImageNet.

In Chapter 7, we propose certified defenses against *adversarial poisoning attacks*, which distort training data of a classifier in order to corrupt the test-time behavior of a classifier. Specifically, we propose two novel provable defenses: (i) Deep Partition Aggregation (DPA), a certified defense against a *general* poisoning threat model, defined as the insertion or deletion of a bounded number of samples to the training set — by implication, this threat model also includes arbitrary distortions to a bounded number of training samples and/or labels; and (ii) Semi-Supervised DPA (SS-DPA), a certified defense against label-flipping poisoning attacks. Note that our certificates apply to an individual *test* sample: in DPA for example, for each given test sample $\boldsymbol{x}$, we report a magnitude $\rho(\boldsymbol{x})$ representing the minimum number of training samples which must be inserted or deleted in order to change the outputof the trained classifier given input $\boldsymbol{x}$.

DPA is an ensemble method where base models are trained on partitions of the training set determined by a hash function. DPA is related to both *subset aggregation* [56], a well-studied ensemble method in classical machine learning, as well as to randomized smoothing. Our defense

against label-flipping poison attacks, SS-DPA, uses a semi-supervised learning algorithm as its base classifier model: each base classifier is trained using the entire unlabeled training set in addition to the labels for a partition. SS-DPA significantly outperforms the prior state of the art certified defense for label-flipping attacks [6] on both MNIST and CIFAR-10, while DPA represented the first certified defence to be proposed against general poisoning attacks.

The certification techniques presented in Chapters 6 and 7 are both *deterministic*, and in fact can be though of as specialised forms of the deterministic ℓ_0 certificate introduced in Chapter 4, in Section 4.5.[2] The Wasserstein smoothing technique proposed in Chapter 5 is a randomized technique, providing only probabilistic certificates: although our approach is to relate the problem of Wasserstein certification to certification in the ℓ_1 metric, the details of our reduction rely on the use of *additive* smoothing noise, which means that the deterministic approach proposed in Chapter 3 cannot be applied. See Section D.4 for details.

As in Part I, the experiments in Part II are conducted on image data – the Wasserstein and patch threat models considered in Chapters 5 and 6 respectively are in fact fairly specific to image data. However, the poisoning threat models considered in Chapter 7 are broadly applicable to any supervised classification task, and could be directly applied to other domains, such as text, video, audio, or sensor measurement classification.

[2]Note that Chapters 6 and 7 were originally published before the work on deterministic ℓ_p ($p < 1$) and ℓ_0 certificates in Chapter 4 was conducted, but after Chapter 2, discussing *randomized* smoothing for ℓ_0, was published. The connection between the deterministic ℓ_0 certificate and SS-DPA is discussed explicitly in Section 4.5, while the patch defense in Chapter 6 is discussed in terms of the randomized ℓ_0 (i.e., Chapter 2) defense in Sections 6.1-6.2.

1.2.3 Part III: Test-time Adaptability as Robustness in Reinforcement Learning

Many applications of reinforcement learning can be formalized as goal-conditioned environments, where, in each episode, there is a "goal" that affects the rewards obtained during that episode but does not affect the dynamics. This ability to adapt to new objectives, specified only at test time, represents a form of robustness against real-time changes in a system's requirements for operation.

In Chapter 8, we explore a connection between off-policy reinforcement learning in goal-conditioned settings and *knowledge distillation*, the task of efficiently training a "student" neural network to match the behavior of a "teacher" function. In particular: the current Q-value function and the target Q-value estimate are both functions of the goal, and one would like to train the Q-value function to match its target for *all* goals. This therefore can be framed as a knowledge distillation problem: one can view the target Q-value estimate as a (stochastic, difficult to compute) teacher function, and we are trying to fit the Q-network to represent the same function of the goal. We therefore apply Gradient-Based Attention Transfer [57], a knowledge distillation technique, to the Q-function update. We empirically show that this can improve the performance of goal-conditioned off-policy reinforcement learning when the space of goals is high-dimensional. We also show that this technique can be adapted to allow for efficient learning in the case of multiple simultaneous sparse goals, where the agent can attain a reward by achieving any one of a large set of objectives, all specified at test time. Finally, to give theoretical support, we give examples of classes of environments where (under some assumptions) standard off-policy algorithms require at least $O(d^2)$ replay buffer transitions to learn an optimal policy, while our technique requires only $O(d)$ transitions, where d is the dimensionality of the goal and state space.

Part I

Improved, Large-Scale, Deterministic Robustness Certificates for ℓ_p $(p \leq 1)$ Distances

Chapter 2: Robustness Certificates for Sparse Adversarial Attacks by Randomized Ablation[1]

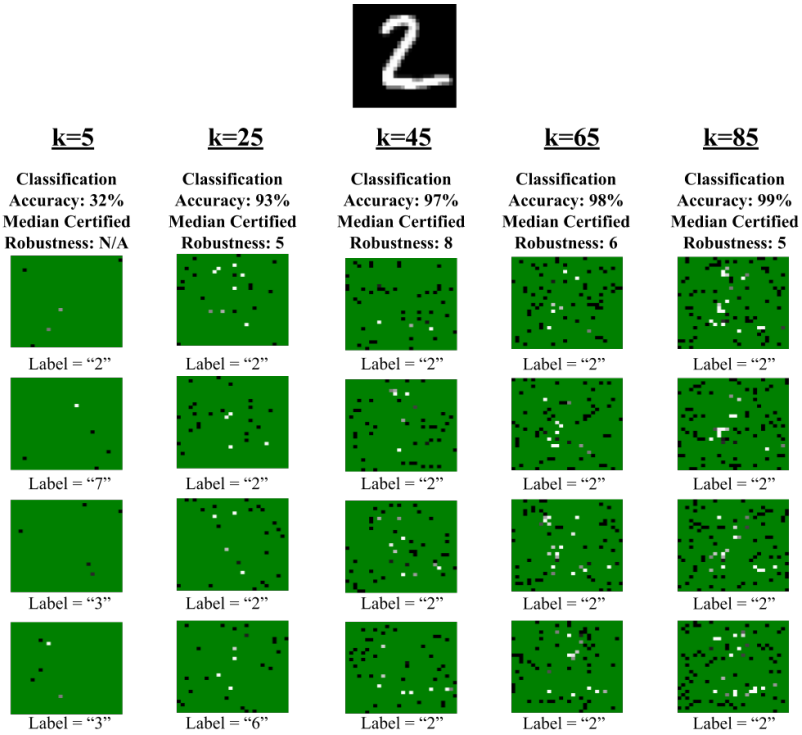


Figure 2.1: An illustration of our proposed certifiably robust classification scheme on MNIST. At the top, the image to be classified is shown. For randomly ablated images, we retain only k out of 784 total pixels (green pixels in these images are not used in classification). For each value of k, we show four randomly ablated images along with their base classifier labels. For small values of k, the *smoothed* classifier's accuracy in the test set is low ($\sim 32\%$ for $k = 5$) while the accuracy increases for moderate values of k ($\sim 97\%$ for $k = 45$). In each case, we compute the *median certified robustness* for the smoothed classifier of the ℓ_0 attack magnitude that classifications are provably protected against. The median is over the MNIST test set. For example, for $k = 45$, we guarantee the robustness of our proposed method against all ℓ_0 adversarial attacks that perturb 8 or fewer pixels.

[1] A form of chapter has been published in AAAI 2020 [58].

Most existing work in adversarial examples has used ℓ_p norms as distance metrics, focusing in particular on ℓ_∞ and ℓ_2 norms [7, 9, 10, 59, 60]. The ℓ_0 metric, which is simply the number of pixels at which $\boldsymbol{x}'$ differs from $\boldsymbol{x}$, has also been the target of adversarial attacks. This metric presents a distinct challenge, because $d(\boldsymbol{x}, \boldsymbol{x}')$ is non-differentiable. However, both gradient-based (white-box) attacks [9, 61] and zeroth-order (black-box) attacks [3] have been proposed under the ℓ_0 attack model. The ℓ_0 attack model is the focus of this chapter.

Several practical defenses against adversarial attacks under the ℓ_0 attack model have been proposed in the last couple of years. These methods include defensive distillation [13], as well as attempts to recover $\boldsymbol{x}$ from $\boldsymbol{x}'$ using compressed sensing [14] or generative models [3, 15]. However, as new defenses are proposed, new attacks are also developed for which these defenses are vulnerable (e.g. [19]). Experimental demonstrations of a defense's efficacy based on currently existing attacks do not provide a general proof of security.

In this chapter, we develop a certifiably robust classification scheme for the ℓ_0 metric (i.e. sparse adversarial perturbations). To guarantee the robustness of the classification against sparse adversarial attacks, we propose a novel smoothing method based on performing random *ablations* on the input image, rather than adding random noise. In our proposed ℓ_0 smoothing method, for each sample generated from $\boldsymbol{x}$, a majority of pixels are randomly dropped from the image before the image is given to the base classifier. If a relatively small number ρ of pixels have been adversarially corrupted (which is the case in sparse adversarial attacks), then it is highly likely that none of these pixels are present in a given ablated sample. Then, for the majority of possible random ablations, $\boldsymbol{x}$ and $\boldsymbol{x}'$ will give the same ablated image. Therefore, the expected number of "votes" for each class can only differ between $\boldsymbol{x}$ and $\boldsymbol{x}'$ by a bounded amount. Using this, we can prove that with high probability, the smoothed classifier will classify $\boldsymbol{x}$ robustly against any

sparse adversarial attack which is allowed to perturbed certain number of input pixels, provided that the 'gap' between the number of votes for the top class and the number of 'votes' for any other class at $\boldsymbol{x}$ is sufficiently large. (See Figure 2.1)

Our ablation method produces significantly larger robustness guarantees compared to a more direct extension of randomized smoothing to the ℓ_0 metric provided in a concurrent work by [1]: see the Discussion section for a comparison of the techniques.

We note that our proposed approach bears some similarities to [62], in that both works aim to defend against ℓ_0 adversarial attacks by randomly ablating pixels. However, several differences exist: most notably, [62] presents a *practical* defense with no robustness certificate given. By contrast, the main contribution of this work is a provable guarantee of robustness to adversarial attack.

In summary, our contributions are as follows:

- We develop a novel defense technique against sparse adversarial attacks (threat models that use the ℓ_0 metric) based on randomized ablation.
- We characterize robustness guarantees for our proposed defense against arbitrary sparse adversarial attacks.
- We show the effectiveness of the proposed technique on standard datasets: MNIST, CIFAR-10, and ImageNet.

2.1 Preliminaries and Notation

We will use $\mathcal{S}$ to represent the set of possible pixel values in an image. For example, in an 24-bit RGB color image, $\mathcal{S} = \{0, 1, ..., 255\}^3$, while in a binarized black-and-white image,

$\mathcal{S} = \{0, 1\}$. We will use $\mathcal{X} = \mathcal{S}^d$ to represent the set of possible images, where d is the number of pixels in each image. Additionally, we will use $\mathcal{S}_{\text{NULL}}$ to represent the set $\mathcal{S} \cup \{\text{NULL}\}$, where NULL is a null symbol representing the absence of information about a pixel, and $\mathcal{X}_{\text{NULL}} = {\mathcal{S}_{\text{NULL}}}^d$ to represent the set of images where some elements in the images may be replaced by the null symbol. Note that NULL is *not* the same as a zero-valued pixel, or black. For example, if $\mathcal{S} = \{0, 1\}$ and $d = 5$, then $[0, 1, 1, 0, 1]^T \in \mathcal{X}$, while $[\text{NULL}, 1, \text{NULL}, 0, 1]^T \in \mathcal{X}_{\text{NULL}}$.

Also, let $[d]$ represent the set of indices $\{1, ..., d\}$, let $\mathcal{H}(d, k) \subseteq \mathcal{P}([d])$ represent all sets of k unique indices in $[d]$, and let $\mathcal{U}(d, k)$ represent the uniform distribution over $\mathcal{H}(d, k)$. (To sample from $\mathcal{U}(d, k)$ is to sample k out of d indices uniformly *without replacement*. For example, an element sampled from $\mathcal{U}(5, 3)$ might be $\{2, 4, 5\}$.)

We define the operation $\textsc{Ablate} \in \mathcal{X} \times \mathcal{H}(d, k) \to \mathcal{X}_{\text{NULL}}$, which takes an image and a set of indices, and outputs the image, with all pixels *except* those in the set replaced with the null symbol NULL. For example, $\textsc{Ablate}([0, 1, 1, 0, 1]^T, \{2, 4, 5\}) = [\text{NULL}, 1, \text{NULL}, 0, 1]^T$

For images $\boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}$, let $\|\boldsymbol{x} - \boldsymbol{x}'\|_0$ denote the ℓ_0 distance between $\boldsymbol{x}$ and $\boldsymbol{x}'$, defined as the number of pixels at which $\boldsymbol{x}$ and $\boldsymbol{x}'$ differ. Note that we are following the convention used by [8], where, for a color image, the number of channels in which the images differ at a given pixel location does not matter: any difference at a pixel location (corresponding to an index in $[d]$) counts the same. This differs from [61], in which channels are counted separately. Also (in a slight abuse of notation) let $\boldsymbol{x} \ominus \boldsymbol{x}'$ denote the set of pixel indices at which $\boldsymbol{x}$ and $\boldsymbol{x}'$ differ, so that $\|\boldsymbol{x} - \boldsymbol{x}'\|_0 = |\boldsymbol{x} \ominus \boldsymbol{x}'|$. Finally, for multiclass classification problems, let c be the number of classes.

2.2 Certifiably Robust Classification Scheme

First, we note that in this section, we closely follow the notation of [63], using appropriate analogs between the ℓ_2 smoothing scheme of that work, and the proposed ℓ_0 ablation scheme of this work. In particular, let $f \in \mathcal{X}_{\text{NULL}} \to [c]$ denote a *base classifier*, which is trained to classify images with some pixels ablated. Let $g \in \mathcal{X} \to [c]$ represent a *smoothed classifier*, defined as:

$$g(\boldsymbol{x}) = \arg\max_i \left[\Pr_{\mathcal{T} \sim \mathcal{U}(d,k)} (f(\textsc{Ablate}(\boldsymbol{x}, \mathcal{T})) = i) \right] \tag{2.1}$$

where k is the *retention constant*; i.e., the number of pixels retained (not ablated) from $\boldsymbol{x}$. In other words, $g(\boldsymbol{x})$ denotes the class *most likely to be returned* if we first randomly ablate all but k pixels from $\boldsymbol{x}$ and then classify the resulting image with the base classifier f. To simplify notation, we will let $p_i(\boldsymbol{x})$ denote the probability that, after ablation, f returns the class i:

$$p_i(\boldsymbol{x}) = \Pr_{\mathcal{T} \sim \mathcal{U}(d,k)} (f(\textsc{Ablate}(\boldsymbol{x}, \mathcal{T})) = i) \,. \tag{2.2}$$

Thus, $g(\boldsymbol{x})$ can be defined simply as $\arg\max_i [p_i(\boldsymbol{x})]$.

We first prove the following general theorem, which can be used to develop a variety of related robustness certificates.

Theorem 2.1. *For images $\boldsymbol{x}, \boldsymbol{x}'$, with $\|\boldsymbol{x} - \boldsymbol{x}'\|_0 \leq \rho$, for all classes $i \in [c]$:*

$$|p_i(\boldsymbol{x}') - p_i(\boldsymbol{x})| \leq \Delta \tag{2.3}$$

where

$$\Delta = 1 - \frac{\binom{d-\rho}{k}}{\binom{d}{k}}. \tag{2.4}$$

See Figure 2.2 for a plot of how the constant Δ scales with k and ρ. We present a short proof of Theorem 2.1 here:

Proof. Recall that (with $\mathcal{T} \sim \mathcal{U}(d, k)$):

$$\begin{aligned} p_i(\boldsymbol{x}) &= \Pr(f(\textsc{Ablate}(\boldsymbol{x}, \mathcal{T})) = i) \\ p_i(\boldsymbol{x}') &= \Pr(f(\textsc{Ablate}(\boldsymbol{x}', \mathcal{T})) = i) \end{aligned} \tag{2.5}$$

By the law of total probability:

$$\begin{aligned} &p_i(\boldsymbol{x}) = \\ &\Pr([f(\textsc{Ablate}(\boldsymbol{x}, \mathcal{T})) = i] \wedge [\mathcal{T} \cap (\boldsymbol{x} \ominus \boldsymbol{x}') = \varnothing]) + \\ &\Pr([f(\textsc{Ablate}(\boldsymbol{x}, \mathcal{T})) = i] \wedge [\mathcal{T} \cap (\boldsymbol{x} \ominus \boldsymbol{x}') \neq \varnothing]) \\ &p_i(\boldsymbol{x}') = \\ &\Pr([f(\textsc{Ablate}(\boldsymbol{x}', \mathcal{T})) = i] \wedge [\mathcal{T} \cap (\boldsymbol{x} \ominus \boldsymbol{x}') = \varnothing]) + \\ &\Pr([f(\textsc{Ablate}(\boldsymbol{x}', \mathcal{T})) = i] \wedge [\mathcal{T} \cap (\boldsymbol{x} \ominus \boldsymbol{x}') \neq \varnothing]) \end{aligned} \tag{2.6}$$

Note that if $\mathcal{T} \cap (\boldsymbol{x} \ominus \boldsymbol{x}') = \varnothing$, then $\boldsymbol{x}$ and $\boldsymbol{x}'$ are identical at all indices in $\mathcal{T}$. Then in this case, $\textsc{Ablate}(\boldsymbol{x}, \mathcal{T})) = \textsc{Ablate}(\boldsymbol{x}', \mathcal{T}))$, which implies:

$$\begin{aligned} &\Pr(f(\textsc{Ablate}(\boldsymbol{x}, \mathcal{T})) = i \mid \mathcal{T} \cap (\boldsymbol{x} \ominus \boldsymbol{x}') = \varnothing) = \\ &\Pr(f(\textsc{Ablate}(\boldsymbol{x}', \mathcal{T})) = i \mid \mathcal{T} \cap (\boldsymbol{x} \ominus \boldsymbol{x}') = \varnothing) \end{aligned} \tag{2.7}$$

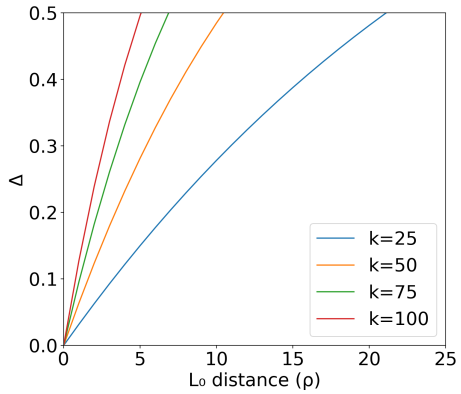


Figure 2.2: The bounding constant Δ from Theorem 2.1, shown for MNIST-sized images (d=784). The constant k is the number of pixels retained in each randomly ablated sample.

Multiplying both sides of (2.7) by $\Pr(\mathcal{T} \cap (\boldsymbol{x} \ominus \boldsymbol{x}') = \varnothing)$ gives:

$$\begin{aligned} &\Pr([f(\textsc{Ablate}(\boldsymbol{x}, \mathcal{T})) = i] \wedge [\mathcal{T} \cap (\boldsymbol{x} \ominus \boldsymbol{x}') = \varnothing]) = \\ &\Pr([f(\textsc{Ablate}(\boldsymbol{x}', \mathcal{T})) = i] \wedge [\mathcal{T} \cap (\boldsymbol{x} \ominus \boldsymbol{x}') = \varnothing]) \end{aligned} \tag{2.8}$$

Substituting (2.8) into (2.6) and rearranging yields:

$$\begin{aligned} &p_i(\boldsymbol{x}') = p_i(\boldsymbol{x}) - \\ &\Pr([f(\textsc{Ablate}(\boldsymbol{x}, \mathcal{T})) = i] \wedge [\mathcal{T} \cap (\boldsymbol{x} \ominus \boldsymbol{x}') \neq \varnothing]) + \\ &\Pr([f(\textsc{Ablate}(\boldsymbol{x}', \mathcal{T})) = i] \wedge [\mathcal{T} \cap (\boldsymbol{x} \ominus \boldsymbol{x}') \neq \varnothing]) \end{aligned} \tag{2.9}$$

Because probabilities are non-negative, this gives:

$$\begin{aligned} &p_i(\boldsymbol{x}) - \Pr([f(\textsc{Ablate}(\boldsymbol{x}, \mathcal{T})) = i] \wedge [\mathcal{T} \cap (\boldsymbol{x} \ominus \boldsymbol{x}') \neq \varnothing]) \\ &\leq p_i(\boldsymbol{x}') \leq \\ &p_i(\boldsymbol{x}) + \Pr([f(\textsc{Ablate}(\boldsymbol{x}', \mathcal{T})) = i] \wedge [\mathcal{T} \cap (\boldsymbol{x} \ominus \boldsymbol{x}') \neq \varnothing]) \end{aligned} \tag{2.10}$$

By the conjunction rule, this implies:

$$p_i(\boldsymbol{x}) - \Pr(\mathcal{T} \cap (\boldsymbol{x} \ominus \boldsymbol{x}') \neq \varnothing) \leq p_i(\boldsymbol{x}') \leq p_i(\boldsymbol{x}) + \Pr(\mathcal{T} \cap (\boldsymbol{x} \ominus \boldsymbol{x}') \neq \varnothing) \tag{2.11}$$

Note that:

$$\Pr(\mathcal{T} \cap (\boldsymbol{x} \ominus \boldsymbol{x}') \neq \varnothing) = 1 - \Pr(\mathcal{T} \cap (\boldsymbol{x} \ominus \boldsymbol{x}') = \varnothing) = 1 - \frac{\binom{d-|\boldsymbol{x} \ominus \boldsymbol{x}'|}{k}}{\binom{d}{k}} \tag{2.12}$$

Where the last equality follows because $\mathcal{T}$ is an uniform choice of k elements from d: there are $\binom{d}{k}$ total ways to make this selection, $\binom{d-|\boldsymbol{x} \ominus \boldsymbol{x}'|}{k}$ of which contain no elements from $(\boldsymbol{x} \ominus \boldsymbol{x}')$. Then:

$$\Pr(\mathcal{T} \cap (\boldsymbol{x} \ominus \boldsymbol{x}') \neq \varnothing) = 1 - \frac{\binom{d-|\boldsymbol{x} \ominus \boldsymbol{x}'|}{k}}{\binom{d}{k}} = 1 - \frac{\binom{d-\|\boldsymbol{x} - \boldsymbol{x}'\|_0}{k}}{\binom{d}{k}} \leq 1 - \frac{\binom{d-\rho}{k}}{\binom{d}{k}} = \Delta \tag{2.13}$$

Combining inequalities (2.13) and (2.11) gives the statement of Theorem 2.1. □

2.2.1 Practical Robustness Certificates

Depending on the architecture of the base classifier, it may be infeasible to directly compute $p_i(\boldsymbol{x})$, and therefore to compute $g(\boldsymbol{x})$. However, we can instead generate a representative sample from $\mathcal{U}(d, k)$, in order to bound $p_i(\boldsymbol{x})$ with high confidence. In particular, let $\underline{p_i(\boldsymbol{x})}$ represent a lower bound on $p_i(\boldsymbol{x})$, with $(1 - \alpha)$ confidence, and let $\overline{p_i(\boldsymbol{x})}$ represent a similar upper bound. We first develop a certificate analogous for the ℓ_0 attack to the certificate presented in [63]:

Corollary 2.1. *For images $\boldsymbol{x}, \boldsymbol{x}'$, with $\|\boldsymbol{x} - \boldsymbol{x}'\|_0 \leq \rho$, if:*

$$\underline{p_i(\boldsymbol{x})} - \Delta > 0.5 \tag{2.14}$$

then, with probability at least $1-\alpha$*:*

$$g(\boldsymbol{x}') = i \tag{2.15}$$

Proof. With probability at least $1-\alpha$:

$$.5 < \underline{p_i(\boldsymbol{x})} - \Delta \leq p_i(\boldsymbol{x}) - \Delta \leq p_i(\boldsymbol{x}') \tag{2.16}$$

where the final inequality is from Theorem 2.1. Then $g(\boldsymbol{x}') = i$ from the definition of g. □

This bound applies directly to the true population value of $g(\boldsymbol{x}')$, not necessarily to an empirical estimate of $g(\boldsymbol{x}')$. Following [63], we therefore use a separate sampling procedure to estimate the value of the classifier $g(.)$, which itself has a bounded failure rate independent from the failure rate of the certificate, and which may abstain from classification if the top class probabilities are too similar to distinguish based on the samples. Note that by using a large number of samples, this estimation error can be made arbitrarily small. In fact, because Corollary 2.1 is directly analogous to the condition for ℓ_2 robustness presented in [63], we borrow both the empirical classification and the empirical certification procedures from that paper wholesale. We refer the reader to that work for details: it is sufficient to say that with these procedures, we can bound $\underline{p_i(\boldsymbol{x})}$ with $(1-\alpha)$ confidence and also estimate $g(\boldsymbol{x}')$ with $(1-\alpha)$ confidence. This is the procedure we use in our experiments.

Alternatively, one can instead use a certificate analogous to the certificate presented in [64].

Corollary 2.2. *For images* $\boldsymbol{x}, \boldsymbol{x}'$*, with* $\|\boldsymbol{x} - \boldsymbol{x}'\|_0 \leq \rho$*, if:*

$$\underline{p_i(\boldsymbol{x})} - \Delta > \arg\max_{k \neq i} \overline{p_k(\boldsymbol{x})} + \Delta \tag{2.17}$$

then, with probability at least $1 - \alpha$*:*

$$g(\boldsymbol{x}') = i. \tag{2.18}$$

Proof. For each $k \neq i$:

$$\begin{aligned} p_k(\boldsymbol{x}') &\leq p_k(\boldsymbol{x}) + \Delta \leq \overline{p_k(\boldsymbol{x})} + \Delta \leq \arg\max_{k \neq i} \overline{p_k(\boldsymbol{x})} + \Delta \\ &< \underline{p_i(\boldsymbol{x})} - \Delta \leq p_i(\boldsymbol{x}) - \Delta \leq p_i(\boldsymbol{x}') \end{aligned} \tag{2.19}$$

where the first and last inequalities are from Theorem 2.1. □

In a multi-class setting, Corollary 2.2 might appear to give a tighter certificate bound. However, the upper and lower bounds on $p_j(\boldsymbol{x})$ must hold simultaneously for all j with a total failure rate of $(1 - \alpha)$. This can lead to greater estimation error if the number of classes c is large.[2]

2.2.2 Architectural and training considerations

Similar to existing works on smoothing-based certified adversarial robustness, we train our base classifier f on noisy images (i.e. ablated images), rather than training g directly. For performance reasons, during training, we ablate the same pixels from all images in a minibatch. We use the same retention constant k during training as at test time.

2.2.2.1 Encoding $\mathcal{S}_{\text{NULL}}$

We use standard CNN-based architectures for the classifier $f(.)$. However, this presents an architectural challenge: we need to be able to represent the absence of information at a pixel

[2] After the publication of this work, [65] derived a tighter empirical estimation specifically for the certificate derived here.

(the symbol NULL), as distinct from any color that can normally be encoded. Additionally, we would like the encoding of NULL to be equally far from every possible encodable color, so that the network is not biased towards treating it as one color moreso than another. To achieve these goals, we encode images as follows: for greyscale images where pixels in $\mathcal{S}$ are floating point values between zero and one (i.e. $\mathcal{S} = [0, 1]$), we encode $s \in \mathcal{S}$ as the tuple $(s, 1 - s)$, and then encode NULL as $(0, 0)$. Practically, this means that we double the number of color channels from one to two, with one channel representing the original image and the other channel representing its inverse. Then, NULL is represented as zero on both channels: this is distinct from grey $(0.5, 0.5)$, white $(1, 0)$, or black $(0, 1)$. Notably, the values over the channels add up to one for a pixel representing any color, while it adds up to zero for a null pixel. For color images, we use the same encoding technique increasing the number of channels from 3 to 6. The resulting channels are then $(\text{red}, \text{green}, \text{blue}, 1 - \text{red}, 1 - \text{green}, 1 - \text{blue})$, while NULL is encoded as $(0, 0, 0, 0, 0, 0)$.[3]

2.3 Results

In this section, we provide experimental results of the proposed method on MNIST, CIFAR-10, and ImageNet. When reporting results, we refer to the following quantities:

- The *certified robustness* of a particular image $\boldsymbol{x}$ is the maximum ρ for which we can certify (with probability at least $1 - \alpha$) that the smoothed classifier $g(\boldsymbol{x}')$ will return the *correct* label where $\boldsymbol{x}'$ is any adversarial perturbation of $\boldsymbol{x}$ such that $\|\boldsymbol{x} - \boldsymbol{x}'\|_0 \leq \rho$. If the unperturbed classification $g(\boldsymbol{x})$ is itself incorrect, we define the certified robustness as N/A (Not Applicable).

[3] On CIFAR-10, we scaled colors between 0 and 1 when using this encoding. On ImageNet, we normalized each channel to have mean 0 and standard deviation 1 before applying this encoding: in this case, the NULL symbol is still distinct, although it is not equidistant from all other colors.

- The *certified accuracy at* ρ on a dataset is the fraction of images in the dataset with *certified robustness* of at least ρ. In other words, it is the guaranteed accuracy of the classifier $g(.)$, if all images are corrupted with any ℓ_0 adversarial attack of measure up to ρ.

- The *median certified robustness* on a dataset is the median value of the *certified robustness* across the dataset. Equivalently, it is the maximum ρ for which the *certified accuracy at* ρ is at least 0.5. When computing this median, images which $g(.)$ misclassifies when unperturbed (i.e., *certified robustness* is N/A) are counted as having $-\infty$ certified robustness. For example, if the robustness certificates of images in a dataset are $\{$N/A,N/A,1,2,3$\}$, the *median certified robustness* is 1, not 2.

- The *classification accuracy* on a dataset is the fraction of images on which our empirical estimation of $g(.)$ returns the correct class label, and does not abstain.

- The *empirical adversarial attack magnitude* of a particular image $\boldsymbol{x}$ is the minimum ρ for which an adversarial attack can find an adversarial example $\boldsymbol{x}'$ such that $\|\boldsymbol{x} - \boldsymbol{x}'\|_0 \leq \rho$, and such that our empirical classification procedure misclassifies or abstains on $\boldsymbol{x}'$.

- The *median adversarial attack magnitude* on a dataset is the median value of the *empirical adversarial attack magnitude* across the dataset.

Unless otherwise stated, the uncertainty α is 0.05, and 10,000 randomly-ablated samples are used to make each prediction. The empirical estimation procedure we use to generate certificates, from [63], requires two sampling steps: the first to identify the majority class i, and the second to bound $\underline{p_i(\boldsymbol{x})}$. We use 1,000 and 10,000 samples, respectively, for these two steps.

2.3.1 Results on MNIST

We first tested our robust classification scheme on MNIST, using a simple CNN model as the base classifier (see appendix for architectural details.) Results are presented in Table 2.1. We varied the number of retained pixels k in each sample: note that for small k, certified robustness and accuracy both increase as k increases. However, after a certain threshold, here achieved at $k = 45$, certified robustness starts to decrease with k, while classification accuracy continues to increase. This can be understood by considering Figure 2.2: For larger k, the bounding constant Δ grows considerably faster with the ℓ_0 distance ρ. In other words, a larger fraction of ablated samples must be classified correctly to achieve the same certified robustness. For small k, the fraction of ablated samples classified correctly increases sufficiently quickly with k to counteract this effect; however, after a certain point, it is no longer beneficial to increase k because a large majority of samples are already classified correctly by the base classifier (For example, see Figure 2.1).

We also tested the empirical robustness of our classifier to an ℓ_0 adversarial attack. Specifically, we chose to use the black-box *Pointwise attack* proposed by [3]. We choose a black-box attack because comparisons to other robust classifiers using gradient-based attacks (such as the ℓ_0 attack proposed by [8]) may be somewhat asymmetric since our smoothed classifier is non-differentiable (because the base classifier's output is discretized.) While [33] does propose a gradient-based scheme for attacking ℓ_2-smoothed classifiers which are similarly non-differentiable, adapting such a scheme would be a non-trivial departure from the existing ℓ_0 Carlini-Wagner attack, precluding a direct comparison to other robust classifiers. By contrast, a practical reason we choose the Pointwise Attack is that the reference implementation of the attack is available as part of the

Retained pixels k	Classification accuracy (Percent abstained)	Median certified robustness
5	32.32% (5.65%)	N/A
10	74.90% (5.08%)	0
15	86.09% (2.82%)	0
20	90.29% (1.81%)	3
25	93.05% (1.02%)	5
30	94.68% (0.77%)	7
35	95.40% (0.66%)	7
40	96.27% (0.52%)	8
45	**96.72% (0.45%)**	**8**
50	97.16% (0.32%)	7
55	97.41% (0.34%)	7
60	97.78% (0.18%)	7
65	98.05% (0.15%)	6
70	98.18% (0.20%)	6
75	98.28% (0.20%)	6
80	98.37% (0.12%)	5
85	98.57% (0.12%)	5
90	98.58% (0.16%)	5
95	98.73% (0.11%)	5
100	98.75% (0.16%)	4

Table 2.1: Robustness certificates on MNIST, using different numbers of retained pixels (k). The maximum median certified robustness on the MNIST test set is achieved when using $k = 40$ or $k = 45$ retained pixels: because $k = 45$ gives better classification accuracy, we use this model (highlighted in bold) when evaluating against adversarial attacks.

Foolbox package [66], meaning that we can directly compare our results to that of [3], without any concerns about implementation details. We note that [3] reports a median adversarial attack magnitude of 9 pixels for an unprotected CNN model on MNIST, which is comparable to the *mean* adversarial attack magnitude of 8.5 reported for the ℓ_0 Carlini-Wagner attack. This suggests that the attack is comparably effective. Results are presented in Table 2.2. Note that our model appears to be significantly more robust to ℓ_0 attack than any of the models tested by [3], at a slight cost of classification accuracy (We would anticipate this trade-off, see [67].) Also note that while there is a gap between the median certified lower bound for the magnitude of any attack, 8

pixels, and the empirical upper bound given by an extant attack, 31 pixels, these quantities are at least in the same order of magnitude, indicating that our certificate is a non-trivial guarantee. See Figure 3 for examples of adversarial attacks on our classifier.

Model	Classification accuracy	Median adversarial attack magnitude
CNN	99.1%	9.0
Binarized CNN	98.5%	11.0
Nearest Neighbor	96.9%	10.0
ℓ_∞-Robust [9]	98.8%	4.0
[3]	99.0%	16.5
Binarized [3]	99.0%	22.0
Our model ($k = 45$)	**96.7%**	**31.0**

Table 2.2: Median adversarial attack magnitude on MNIST using the Pointwise attack from [3], taking the best attack on each image from 10 random restarts. Note that all values except for our model are taken directly from [3]. For every evaluation performed by the black-box attack, 10,000 ablated samples were used to calculate class scores of our model: this was to ensure stability of the evaluated scores. Additionally, causing our model to abstain from classifying was counted as a successful attack, even if the correct class score was still marginally highest. Because the black-box attack performs a large number of classifications, and each of these classifications required 10,000 evaluations of the base classifier, we used only a subset of the MNIST test set, consisting of 275 images.

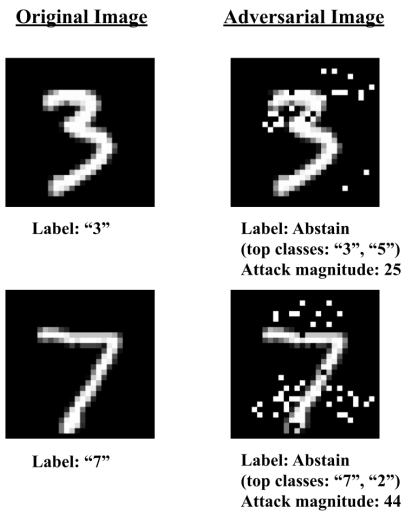


Figure 2.3: Adversarial examples to our classifier on MNIST. Note that because we consider the classifier abstaining to be a successful attack, these adversarial examples are in fact on the boundary between classes, rather than being entirely misclassified.

2.3.2 Results on CIFAR-10

Retained pixels k	Classification accuracy (Percent abstained)	Median certified robustness
25	68.41% (1.76%)	6
50	74.21% (1.19%)	7
75	**78.25% (0.93%)**	**7**
100	80.91% (0.86%)	6
125	83.25% (0.60%)	5
150	85.22% (0.53%)	4

Table 2.3: Robustness certificates on CIFAR-10, using different numbers of retained pixels (k), and using ResNet18 [68] as the base classifier. Note that without smoothing, the base implementation of an unprotected ResNet18 classifier which we used [69] has a classification accuracy of 93.02% on CIFAR-10.

Retained pixels k	Base classifier training accuracy	Base classifier test accuracy
25	83.16%	57.72%
50	96.63%	68.29%
75	99.33%	74.08%
100	99.76%	77.88%
125	99.91%	80.48%
150	99.95%	83.16%

Table 2.4: Accuracy of the base classifier f in CIFAR-10 experiments, on training versus test data, using ResNet18. Note that the base classifier significantly overfits to the training data. (Training accuracies are averaged over the final epoch of training.)

We implemented our technique on CIFAR-10 using ResNet18 (with the number of input channels increased to 6) as a base classifier; see Table 2.3 for our robustness certificates as a function of k. The median certified robustness is somewhat smaller than for MNIST: however, this is in line with the performance of empirical attacks. For example, the ℓ_0 attack proposed by [8] achieves a mean adversarial attack magnitude of 8.5 pixels on MNIST and 5.9 pixels on CIFAR-10. This suggests that CIFAR-10 samples are more vulnerable to ℓ_0 adversarial attacks

Retained pixels k	Base classifier training accuracy	Base classifier test accuracy
25	83.89%	57.58%
50	96.91%	69.45%
75	99.09%	75.22%
100	99.66%	79.54%
125	99.78%	81.83%
150	99.92%	84.43%

Table 2.5: Accuracy of the base classifier f in CIFAR-10 experiments, on training versus test data, using ResNet50. Note that the base classifier significantly overfits to the training data: however, for $k > 25$, this higher-capacity model overfits less than ResNet18.

compared to the MNIST ones. Intuitively, this is because CIFAR-10 images are both visually complex and low-resolution, so that each pixel carries a large amount of information regarding the classification label. Also note that the classification accuracy on unperturbed images is somewhat reduced. For example, in a model using $k = 150$, the median certified robustness is 4 pixels, and the classifier accuracy is 85.22%. The trade-off between accuracy and robustness is also more pronounced. However, it is not unusual for practical ℓ_0 defenses to achieve accuracy below 90% on CIFAR-10 [15, 70]: our defense may therefore still prove to be usable.

One phenomenon which we encountered when applying our technique to CIFAR-10 was over-fitting of the base classifier (see Table 2.4), which was unexpected because during the training, the classifier is always exposed to new random ablations of the training data. However, the network was still able to memorize the training data, despite never being exposed to the complete images. While interpolation of even randomly labeled training data is a known phenomenon in deep learning [71], we were surprised to see that over-fitting may happen on ablated images, where a particular ablation is likely never repeated in training. In order to better understand this, we use a model trained on a higher-capacity network architecture, ResNet50. The results for the base classifier are given in Table 2.5. Surprisingly, increasing network capacity decreased the

generalization gap slightly for $k \geq 50$ (Note that because the improvement to the base classifier is only marginal, and because ResNet50 is substantially more computationally intensive to use as a base classifier to classify 10,000 ablated samples per image, we opted to compute certificates using the ResNet18 model).

2.3.3 Results on ImageNet

We implemented our technique on ImageNet using ResNet50 (again with the number of input channels increased to 6) as a base classifier; see Table 2.6 for our robustness certificates as a function of k. For testing, we used a random subset of 400 images from the ILSVRC2012 validation set. Note that ImageNet classification is a 1,000-class problem: here we consider only top-1 accuracy. Because these top-1 accuracies are only moderately above 50 percent, the calculation of the median certified robustness is skewed by relatively large fraction of misclassified points: on the points which are correctly classified, the certificates can be considerably larger. For example, at $k = 1000$, if we consider only the 61% of images which are certified for the correct class, the median certificate is *33 pixels*. Similarly, considering only images with certificates other than 'N/A', the median certificates for $k = 500$ and $k = 2000$ are 63 pixels and 16 pixels, respectively.

Retained pixels k	Classification accuracy (Percent abstained)	Median certified robustness
500	52.75% (1.75%)	0
1000	**61.00% (0.00%)**	**16**
2000	62.50% (1.75%)	11

Table 2.6: Robustness certificates on ImageNet, using different numbers of retained pixels k, and using ResNet50 [68] as the base classifier. For ImageNet, $d = 224 \times 224$. Note that without smoothing, the base implementation of an unprotected ResNet50 classifier can be trained on ImageNet to a top-1 accuracy of 76.15% [72].

2.4 Discussion

2.4.1 Comparison to Lee et al. 2019 [1]

In a concurrent work, [1] also present a randomized-smoothing based robustness certification scheme for the ℓ_0 metric. In this scheme, each pixel is retained with a fixed probability κ and is otherwise assigned to a *random* value from the remaining possible pixel values in $\mathcal{S}$. Note that there is no NULL in this scheme. As a consequence, the base classifier lacks explicit information about *which* pixels are retained from the original image, and which have been randomized. The resulting scheme has considerably lower median certified robustness on the datasets tested in both works[4] (Table 2.7):

Dataset	Median certified robustness (pixels) (Lee et al. 2019) [1]	Median certified robustness (pixels) (our model)
MNIST	4	**8**
ImageNet	1	**16**

Table 2.7: Comparison of robustness certificates in [1] and in this work, using the optimal choices of hyperparameters tested in each work. Numbers for [1] are derived from those reported in that work. Note that for ImageNet, [1] considers each color channel as a separate pixel: therefore the median image is robust to distortion in only *one channel* of one pixel. By contrast, our model is robust to distortions in *all channels* in 16 pixels (or, in the limiting case, one channel in 16 pixels).

To illustrate quantitatively how our robust classifier obtains more information from each ablated sample than is available in the *randomly noised* samples in [1], let us consider images of ImageNet scale. Because [1] considers each color channel as a separate pixel when computing

[4] [1] uses a similar scheme to ours to derive an empirical bound on $\underline{p_i}(\boldsymbol{x})$; however, that work uses 100 samples to select i and 100,000 samples to bound it, and reports bounds with $\overline{99.9\%}$ confidence (α = .001). In order to provide a fair comparison, we repeated our certifications on MNIST and ImageNet (for optimized values of k) using these empirical certification parameters. This did not change the median robustness certificates.

certificates, we will use $\mathcal{S} = \{0, ..., 255\}$, and $d = 3 * 224 * 224$. Using [1]'s certificate scheme, in order to certify for one pixel of robustness with $\kappa = 0.1$ probability of pixel retention, we would need to accurately classify noised images with probability $p_i(\boldsymbol{x}) = .596$. Meanwhile, using our ablation scheme, in order to certify one pixel of robustness by correctly classifying same fraction ($p_i(\boldsymbol{x}) = .596$) of ablated images, we can retain at most $k = 14521$ pixels. This is 9.6% of pixels, slightly fewer than the expected number retained in [1]'s scheme.

However, we will now calculate the *mutual information* between each ablated/noised image and the original image for each scheme: this is the expected number of bits of information about the original image which are obtained from observing the ablated/noised image. For illustrative purposes, we will make the simplifying assumption that the dataset overall is uniformly distributed (while this is obviously not true for image classification, it is a reasonable assumption in other classification tasks.) In our scheme, we have simply

$$I_{\text{ablate}} = \log_2 |\mathcal{S}| * k = 8 * k = 116168 \text{ bits.} \tag{2.20}$$

Each of the k retained pixels provides 8 bits of information. However, in the noising scheme from [1], we instead have:

$$I_{\text{Lee et al.}} = d\left(\log_2 |\mathcal{S}| + \kappa \log_2 \kappa + (1-\kappa) \log_2 \frac{1-\kappa}{|\mathcal{S}| - 1}\right) \approx 50590.4 \text{ bits.} \tag{2.21}$$

Therefore, despite using slightly fewer pixels from the original image, over twice the amount of information about the original image is available in our scheme when making each ablated classification. (A derivation of Equation 2.21 is provided in the appendix.)

2.4.2 Alternative encodings of $\mathcal{S}_{\text{NULL}}$

The multichannel encoding of $\mathcal{S}_{\text{NULL}}$ described above, while theoretically well-motivated, is not the only possible encoding scheme. In fact, for MNIST and CIFAR-10, we tested a somewhat simpler encoding for the NULL symbol: we simply used the mean pixel value on the training set, similarly to the practical defense proposed by [62]. We tested using the optimal values of k from the Results section above ($k = 45$ for MNIST and $k = 75$ for CIFAR-10). This resulted in only marginally decreased accuracy and certificate sizes (Table 2.8):

$\mathcal{S}_{\text{NULL}}$ encoding	Classification acc. (Pct. abstained)	Median certified robustness
MNIST		
Multichannel	**96.72% (0.45%)**	**8**
Mean	96.27% (0.43%)	7
CIFAR-10		
Multichannel	**78.25% (0.93%)**	**7**
Mean	77.71% (1.05%)	**7**

Table 2.8: Accuracy and robustness using different encoding schemes for $\mathcal{S}_{\text{NULL}}$.

To understand this, note that the *mean* pixel value (grey in both datasets) is not necessarily a *common* value: it is still possible to distinguish which pixels are ablated (Figure 2.4).

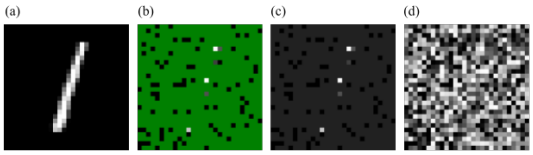


Figure 2.4: (a) An image from MNIST. (b) The image with $k = 85$ pixels ablated, with a unique NULL encoding. (c) The same image with NULL encoded as the mean pixel value (dark grey). Note that both black and white pixels are still distinguishable. (d) If we replace ablated pixels with random noise, the image is no longer easily distinguishable.

Chapter 3: Improved, Deterministic Smoothing for ℓ_1 Certified Robustness[1]

In this chapter, we propose a *non-additive* smoothing method for ℓ_1-certifiable robustness on quantized data that is *deterministic*.[2] By "quantized" data, we mean data where each feature value occurs on a discrete level. For example, standard image files (including standard computer vision datasets, such as ImageNet and CIFAR-10) are quantized, with all pixel values belonging to the set $\{0, 1/255, 2/255, ..., 1\}$. We call our method **D**eterministic **S**moothing with **S**plitting **N**oise (**DSSN**). DSSN produces *exact* certificates, rather than high-probability ones. It also produces certificates in substantially less time than randomized smoothing because a large number of noise samples are no longer required. In addition to these benefits, the certified radii generated by DSSN are significantly larger than those of the prior state-of-the-art.

To develop DSSN, we first propose a randomized method, Smoothing with Splitting Noise (**SSN**). Rather than simple additive noise, SSN uses "splitting" noise to generate a noisy input $\tilde{\boldsymbol{x}}$: first, we generate a noise vector $\boldsymbol{s}$ to split the input domain $[0, 1]^d$ into subdivisions. Then, the noisy input $\tilde{\boldsymbol{x}}$ is just the center of whichever sub-division $\boldsymbol{x}$ belongs to. In contrast to prior smoothing works, this noise model is *non-additive*.

In contrast to additive uniform noise where the noise components (ϵ_i's in ϵ) are indepen-

[1] A form of chapter has been published in ICML 2021 [73].

[2] We note that previous works have proposed deterministic forms of randomized-smoothing certificates [6, 29, 74, 75, 76]. However, this work is, to our knowledge, the first to certify robustness for arbitrary base-classifiers in a norm-based threat model, with a runtime that does not grow exponentially with dimension. See the Section 3.5 for more details.

dently distributed, in SSN, the splitting vector components (s_i's in $\boldsymbol{s}$) *do not* need to be independently distributed. Thus, unlike the additive uniform smoothing where noise vectors must be drawn from a d-dimensional probability distribution, in SSN, the splitting vectors can be drawn from a *one-dimensional* distribution. In the quantized case, the splitting vector can be further reduced to a choice between a small number of elements, leading to a derandomized version of SSN (i.e. DSSN).

Below, we summarize our contributions:

- We propose a novel randomized smoothing method, **SSN**, for the ℓ_1 adversarial threat model (Theorem 3.2).
- We show that **SSN** effectively requires smoothing in *one-dimension* (instead of d), thus it can be efficiently derandomized, yielding a deterministic certifiably robust classification method called **DSSN**.
- On ImageNet and CIFAR-10, we empirically show that DSSN significantly outperforms previous smoothing-based robustness certificates, effectively establishing a new state-of-the-art.

3.1 Preliminaries and Notation

Let $\boldsymbol{x}$, $\boldsymbol{x}'$ represent two points in $[0,1]^d$. We assume that our input space is bounded: this assumption holds for many applications (e.g., pixel values for image classification). If the range of values is not $[0,1]$, all dimensions can simply be scaled. A "base" classifier function will be denoted as $f : \mathbb{R}^d \to [0,1]$. In the case of a multi-class problem, this may represent a single logit.

Let $\delta := \boldsymbol{x}' - \boldsymbol{x}$, with components δ_1, ..., δ_d. A function $p : [0,1]^d \to [0,1]$ is said to be c-Lipschitz with respect to the ℓ_1 norm iff:

$$|p(\boldsymbol{x}) - p(\boldsymbol{x}')| \leq c\|\delta\|_1, \quad \forall \boldsymbol{x}, \boldsymbol{x}'. \tag{3.1}$$

Among the various techniques that have been proposed for certifiably robust classification [22, 23, 24, 25, 26, 27] are those that rely on Lipschitz analysis: if a classifier's logit functions can be shown to be Lipschitz-continuous, this immediately implies a robustness certificate [77, 78]. In particular, consider a classifier with logits $\{p_A, p_B, p_C, ...\}$, all of which are c-Lipschitz. Suppose for an input $\boldsymbol{x}$, we have $p_A(\boldsymbol{x}) > p_B(\boldsymbol{x}) \geq p_C(\boldsymbol{x}) \geq ....$ Also suppose the gap between the largest and the second largest logits is d (i.e. $p_A(\boldsymbol{x}) - p_B(\boldsymbol{x}) = d$). The Lipschitzness implies that for all $\boldsymbol{x}'$ such that $\|\boldsymbol{x} - \boldsymbol{x}'\| < d/(2c)$, $p_A(\boldsymbol{x}')$ will still be the largest logit: in this ball,

$$p_A(\boldsymbol{x}') > p_A(\boldsymbol{x}) - \frac{d}{2} \geq p_{\text{others}}(\boldsymbol{x}) + \frac{d}{2} > p_{\text{others}}(\boldsymbol{x}'), \tag{3.2}$$

where the first and third inequalities are due to Lipschitzness.

Let $\mathcal{U}(a,b)$ represent the uniform distribution on the range $[a,b]$, and $\mathcal{U}^d(a,b)$ represent a random d-vector, where each component is *independently* uniform on $[a,b]$.

Let $\mathbf{1}_{\text{(condition)}}$ represent the indicator function, and $\mathbb{1}$ be the vector $[1,1,...]^T$. In a slight abuse of notation, for $z \in \mathbb{R}, n \in \mathbb{R}^+$, let $z \bmod n := z - n\lfloor \frac{z}{n} \rfloor$ where $\lfloor \cdot \rfloor$ is the floor function; we will also use $\lceil \cdot \rceil$ as the ceiling function. For example, $9.5 \bmod 2 = 1.5$.

We will also discuss quantized data. We will use q for the number of quantizations. Let

$$[a,b]_{(q)} := \left\{ i/q \;\middle|\; \lceil aq \rceil \le i \le \lfloor bq \rfloor \right\}. \tag{3.3}$$

In particular, $[0,1]_{(q)}$ denotes the set $\{0, 1/q, 2/q, ..., (1-q)/q, 1\}$. Let $\mathbf{x}, \mathbf{x}'$ represent two points in $[0,1]^d_{(q)}$. A domain-quantized function $p : [0,1]^d_{(q)} \to [0,1]$ is said to be c-Lipschitz with respect to the ℓ_1 norm iff:

$$|p(\mathbf{x}) - p(\mathbf{x}')| \le c\|\delta\|_1, \quad \forall \mathbf{x}, \mathbf{x}' \in [0,1]^d_{(q)}, \tag{3.4}$$

where $\delta := \mathbf{x}' - \mathbf{x}$. The uniform distribution on the set $[a,b]_{(q)}$ is denoted $\mathcal{U}_{(q)}(a,b)$.

3.2 Prior Work on Uniform Smoothing for ℓ_1 Robustness

Lee et al. [1] proposed an ℓ_1 robustness certificate using uniform random noise:

Theorem 3.1 (Lee et al. [1])**.** *For any $f : \mathbb{R}^d \to [0,1]$ and parameter $\lambda \in \mathbb{R}^+$, define:*

$$p(\boldsymbol{x}) := \mathop{\mathbb{E}}_{\epsilon \sim \mathcal{U}^d(-\lambda,\lambda)} \left[f(\boldsymbol{x} + \epsilon) \right]. \tag{3.5}$$

Then, $p(.)$ is $1/(2\lambda)$-Lipschitz with respect to the ℓ_1 norm.

Yang et al. [5] later provided a theoretical justification for the uniform distribution being optimal among *additive* noise distributions for certifying ℓ_1 robustness[3]. Yang et al. [5] also provided experimental results on CIFAR-10 and ImageNet which before our work were the state-

[3]More precisely, Yang et al. [5] suggested that distributions with d-cubic level sets are optimal for ℓ_1 robustness.

of-the-art ℓ_1 robustness certificates.

Following Cohen et al. [32], Yang et al. [5] applied the smoothing method to a "hard" (in Salman et al. [33]'s terminology) base classifier. That is, if the base classifier returns the class c on input $\boldsymbol{x} + \epsilon$, then $f_c(\boldsymbol{x} + \epsilon) = 1$, otherwise $f_c(\boldsymbol{x} + \epsilon) = 0$. Also following Cohen et al. [32], in order to apply the certificate in practice, Yang et al. [5] first takes $N_0 = 64$ samples to estimate the plurality class A, and then uses $N = 100,000$ samples to lower-bound $p_A(\boldsymbol{x})$ (the fraction of noisy samples $\tilde{\boldsymbol{x}}$ classified as A) with high probability. The other smoothed logit values ($p_B(\boldsymbol{x})$, etc.) can then all be assumed to be $\leq 1 - p_A(\boldsymbol{x})$. This approach has the benefit that each logit does not require an independent statistical bound, and thus reduces the estimation error, but it has the drawback that certificates are impossible if $p_A(\boldsymbol{x}) \leq 0.5$, creating a gap between the clean accuracy of the smoothed classifier and the certified accuracy near $\rho = 0$.

We note that the stated Theorem 3.1 is slightly more general than the originally stated version by Lee et al. [1]: the original version assumed that only $p_A(\boldsymbol{x})$ is available, as in the above estimation scheme, and therefore just gave the ℓ_1 radius in which $p_A(\boldsymbol{x}')$ is guaranteed to remain ≥ 0.5. For completeness, we provide a proof of the more general form (Theorem 3.1) in the appendix.

In this work, we show that by using *deterministic smoothing* with *non-additive noise*, improved certificates can be achieved, because we (i) avoid the statistical issues presented above (by estimating all smoothed logits *exactly*), and (ii) improve the performance of the base classifier itself.

3.3 Our Proposed Method

In this paper, we describe a new method, Smoothing with Splitting Noise (**SSN**), for certifiable robustness against ℓ_1 adversarial attacks. In this method, for each component x_i of $\boldsymbol{x}$, we randomly split the interval $[0, 1]$ into sub-intervals. The noised value $\tilde{x}_i$ is the middle of the sub-interval that contains x_i. We will show that this method corresponds closely to the uniform noise method, and so we continue to use the parameter λ. The precise correspondence will become clear in Section 3.3.2.1: however, for now, λ can be interpreted as controlling (the inverse of) the frequency with which the interval $[0, 1]$ is split into sub-intervals. We will show that this method, unlike the additive uniform noise method, can be efficiently derandomized. For simplicity, we will first consider the case corresponding to $\lambda \geq 0.5$, in which at most two sub-intervals are created, and present the general case later.

Theorem 3.2 ($\lambda \geq 0.5$ Case)**.** *For any $f : \mathbb{R}^d \to [0, 1]$, and $\lambda \geq 0.5$ let $\boldsymbol{s} \in [0, 2\lambda]^d$ be a random variable with a fixed distribution such that:*

$$s_i \sim \mathcal{U}(0, 2\lambda), \quad \forall i. \tag{3.6}$$

*Note that the components $s_1, ..., s_d$ are **not** required to be distributed independently from each other. Then, define:*

$$\tilde{x}_i := \frac{\min(s_i, 1) + \mathbf{1}_{x_i > s_i}}{2}, \quad \forall i \tag{3.7}$$

$$p(\boldsymbol{x}) := \mathop{\mathbb{E}}_{\boldsymbol{s}}[f(\tilde{\boldsymbol{x}})]. \tag{3.8}$$

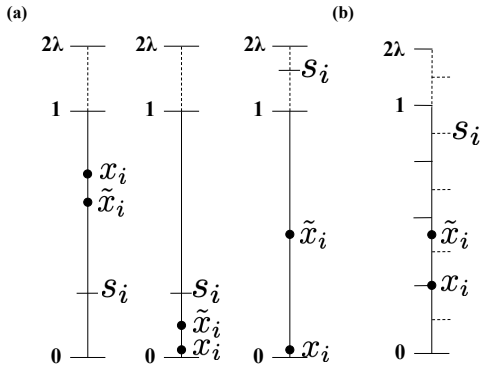


Figure 3.1: (a) Definition of $\tilde{\boldsymbol{x}}$ in the $\lambda \geq 0.5$ case. If $s_i \in [0, 1)$, then it "splits" the interval $[0, 1]$: $\tilde{x}_i$ is the center of whichever sub-interval x_i occurs in. If $s_i > 1$, $\tilde{x}_i = 0.5$, and no information about the original pixel is kept. (b) An example of $\tilde{\mathbf{x}}$ in the *quantized* $\lambda \geq 0.5$ case. Here, $q = 4$ and $2\lambda = 5/4$. We see that $\mathbf{x}_i = 1/4$ lies directly on a quantization level, while $s_i = 7/8$ lies on a half-step between quantization levels. We choose s_i to lie on "half-steps" for the sake of symmetry: the range of $\tilde{\mathbf{x}}_i$ is symmetrical around $1/2$.

Then p is $1/(2\lambda)$-Lipschitz with respect to the ℓ_1 norm.

To understand the distribution of $\tilde{x}_i$, we can view s_i as "splitting" the interval $[0, 1]$ into two sub-intervals, $[0, s_i]$ and $(s_i, 1]$. $\tilde{x}_i$ is then the middle of whichever sub-interval contains x_i. If $s_i \geq 1$, then the interval $[0, 1]$ is not split, and $\tilde{x}_i$ assumes the value of the middle of the entire interval ($= 1/2$): see Figure 3.1-a.

Proof. Consider two arbitrary points $\boldsymbol{x}, \boldsymbol{x}'$ where $\delta := \boldsymbol{x}' - \boldsymbol{x}$. Note that $\max(x_i, x_i') - \min(x_i, x_i') = |x_i' - x_i| = |\delta_i|$. For a fixed vector $\boldsymbol{s}$, additionally note that $\tilde{x}_i = \tilde{x}_i'$ unless s_i falls between x_i and x_i' (i.e., unless $\min(x_i, x_i') \leq s_i < \max(x_i, x_i')$). Therefore:

$$\Pr_{\boldsymbol{s}}[\tilde{x}_i \neq \tilde{x}_i'] = \frac{|\delta_i|}{2\lambda}. \tag{3.9}$$

By union bound:

$$\Pr_{\boldsymbol{s}}[\tilde{\boldsymbol{x}} \neq \tilde{\boldsymbol{x}'}] = \Pr_{\boldsymbol{s}}\left[\bigcup_{i=1}^{d} \tilde{x}_i \neq \tilde{x}_i'\right] \leq \sum_{i=1}^{d} \frac{|\delta_i|}{2\lambda} = \frac{\|\delta\|_1}{2\lambda} \tag{3.10}$$

Then:

$$
\begin{aligned}
&|p(\boldsymbol{x}) - p(\boldsymbol{x}')| \\
&= \left| \mathop{\mathbb{E}}_{\boldsymbol{s}}[f(\tilde{\boldsymbol{x}})] - \mathop{\mathbb{E}}_{\boldsymbol{s}}[f(\tilde{\boldsymbol{x}}')] \right| \\
&= \left| \mathop{\mathbb{E}}_{\boldsymbol{s}}[f(\tilde{\boldsymbol{x}}) - f(\tilde{\boldsymbol{x}}')] \right| \\
&= \left| \Pr_{\boldsymbol{s}}[\tilde{\boldsymbol{x}} \neq \tilde{\boldsymbol{x}}'] \mathop{\mathbb{E}}_{\boldsymbol{s}}[f(\tilde{\boldsymbol{x}}) - f(\tilde{\boldsymbol{x}}') | \tilde{\boldsymbol{x}} \neq \tilde{\boldsymbol{x}}'] + \Pr_{\boldsymbol{s}}[\tilde{\boldsymbol{x}} = \tilde{\boldsymbol{x}}'] \mathop{\mathbb{E}}_{\boldsymbol{s}}[f(\tilde{\boldsymbol{x}}) - f(\tilde{\boldsymbol{x}}') | \tilde{\boldsymbol{x}} = \tilde{\boldsymbol{x}}'] \right|
\end{aligned}
\tag{3.11}
$$

Because $\mathbb{E}_{\boldsymbol{s}}[f(\tilde{\boldsymbol{x}}) - f(\tilde{\boldsymbol{x}}') | \tilde{\boldsymbol{x}} = \tilde{\boldsymbol{x}}']$ is zero, we have:

$$
\begin{aligned}
&|p(\boldsymbol{x}) - p(\boldsymbol{x}')| \\
&= \Pr_{\boldsymbol{s}}[\tilde{\boldsymbol{x}} \neq \tilde{\boldsymbol{x}}'] \left| \mathop{\mathbb{E}}_{\boldsymbol{s}}[f(\tilde{\boldsymbol{x}}) - f(\tilde{\boldsymbol{x}}') | \tilde{\boldsymbol{x}} \neq \tilde{\boldsymbol{x}}'] \right| \\
&\leq \frac{\|\delta\|_1}{2\lambda} \cdot 1
\end{aligned}
\tag{3.12}
$$

where in the final step, we used Equation 3.10, as well as the assumption that $f(\cdot) \in [0, 1]$. Thus, by the definition of Lipschitz-continuity, p is $1/(2\lambda)$-Lipschitz with respect to the ℓ_1 norm. □

It is important that we do **not** require that s_i's be independent. (Note the union bound in Equation 3.10: the inequality holds regardless of the joint distribution of the components of $\boldsymbol{s}$, as long as each s_i is uniform.) This allows us to develop a deterministic smoothing method below.

3.3.1 Deterministic SSN (DSSN)

If SSN is applied to quantized data[4] (e.g. images), we can use the fact that the noise vector $\boldsymbol{s}$ in Theorem 3.2 is *not* required to have independently-distributed components to derive an efficient derandomization of the algorithm. In order to accomplish this, we first develop a quantized version of the SSN method, using input $\mathbf{x} \in [0,1]_q^d$ (i.e. $\mathbf{x}$ is a vector whose components belong to $\{0, 1/q, ..., 1\}$). To do this, we simply choose each of our splitting values s_i to be on one of the half-steps between possible quantized input values: $\boldsymbol{s} \in [0, 2\lambda - 1/q]_{(q)}^d + \mathbb{1}/(2q)$. We also require that 2λ is a multiple of $1/q$ (in experiments, when comparing to randomized methods with continuous λ, we use $\lambda' = \lfloor 2\lambda q \rfloor / 2q$.) See Figure 3.1-b.

Corollary 3.1 ($\lambda \geq 0.5$ Case)**.** *For any $f : \mathbb{R}^d \to [0,1]$, and $\lambda \geq 0.5$ (with 2λ a multiple of $1/q$), let $\boldsymbol{s} \in [0, 2\lambda - 1/q]_{(q)}^d + \mathbb{1}/(2q)$ be a random variable with a fixed distribution such that:*

$$s_i \sim \mathcal{U}_{(q)}\left(0, 2\lambda - 1/q\right) + 1/(2q), \quad \forall i. \tag{3.13}$$

*Note that the components $s_1, ..., s_d$ are **not** required to be distributed independently from each other. Then, define:*

$$\tilde{\mathbf{x}}_i := \frac{\min(s_i, 1) + \mathbf{1}_{\mathbf{x}_i > s_i}}{2}, \quad \forall i \tag{3.14}$$

$$p(\mathbf{x}) := \mathop{\mathbb{E}}_{\boldsymbol{s}}\left[f(\tilde{\mathbf{x}})\right]. \tag{3.15}$$

[4]Note that standard image files such as ImageNet and CIFAR-10 are quantized, with all pixel values belonging to the set $\{0, 1/255, 2/255, ...255/255\}$. As Carlini and Wagner [8] notes, if a natural dataset is quantized, adversarial examples to this dataset must also be quantized (in order to be recognized/saved as valid data at all). Therefore, our assumption of quantized data is a rather loose constraint which applies to many domains considered in adversarial machine learning.

Then, $p(.)$ is $1/(2\lambda)$-Lipschitz with respect to the ℓ_1 norm on the quantized domain $\mathbf{x} \in [0,1]^d_{(q)}$.

Proof. Consider two arbitrary quantized points $\mathbf{x}, \mathbf{x}'$ where $\delta = \mathbf{x}' - \mathbf{x}$. Again, note that

$$\max(\mathbf{x}_i, \mathbf{x}'_i) - \min(\mathbf{x}_i, \mathbf{x}'_i) = |\mathbf{x}'_i - \mathbf{x}_i| = |\delta_i|. \tag{3.16}$$

For a fixed vector $\boldsymbol{s}$, additionally note that $\tilde{\mathbf{x}}_i = \tilde{\mathbf{x}}'_i$ unless s_i falls between $\mathbf{x}_i$ and $\mathbf{x}'_i$ (i.e., unless $\min(\mathbf{x}_i, \mathbf{x}'_i) \le s_i < \max(\mathbf{x}_i, \mathbf{x}'_i)$). Note that δ_i must be a multiple of $1/q$, and that there are exactly $q \cdot |\delta_i|$ discrete values that s_i can take such that the condition $\min(\mathbf{x}_i, \mathbf{x}'_i) \le s_i < \max(\mathbf{x}_i, \mathbf{x}'_i)$ holds. This is out of $2\lambda q$ possible values over which s_i is uniformly distributed. Thus, we have:

$$\Pr_{\boldsymbol{s}}[\tilde{\mathbf{x}}_i \neq \tilde{\mathbf{x}'}_i] = \frac{|\delta_i|}{2\lambda} \tag{3.17}$$

The rest of the proof proceeds as in the continuous case (Theorem 3.2). □

If we required that s_i's be independent, an exact computation of $p(\mathbf{x})$ would have required evaluating $(2\lambda q)^d$ possible values of $\boldsymbol{s}$. This is not practical for large d. However, because we do not have this independence requirement, we can avoid this exponential factor. To do this, we first choose a single scalar splitting value s_{base}: each s_i is then simply a constant offset of s_{base}. We proceed as follows:

First, before the classifier is ever used, we choose a single, fixed, arbitrary vector $\mathbf{v} \in [0, 2\lambda - 1/q]^d_{(q)}$. In practice, $\mathbf{v}$ is generated pseudorandomly when the classifier is trained, and the seed is stored with the classifier so that the same $\mathbf{v}$ is used whenever the classifier is used. Then,

at test time, we sample a scalar variable as:

$$s_{\text{base}} \sim \mathcal{U}_{(q)}(0, 2\lambda - 1/q) + 1/(2q). \tag{3.18}$$

Then, we generate each s_i by simply adding the base variable s_{base} to v_i:

$$\forall i, \quad s_i := (s_{\text{base}} + v_i) \mod 2\lambda \tag{3.19}$$

Note that the marginal distribution of each s_i is $s_i \sim \mathcal{U}_{(q)}(0, 2\lambda - 1/q) + 1/(2q)$, which is sufficient for our provable robustness guarantee. In this scheme, the only source of randomness at test time is the single random scalar s_{base}, which takes on one of $2\lambda q$ values. We can therefore evaluate the exact value of $p(\mathbf{x})$ by simply evaluating $f(\tilde{\mathbf{x}})$ a total of $2\lambda q$ times, for each possible value of s_{base}. Essentially, by removing the independence requirement, the splitting method allows us to replace a d-dimensional noise distribution with a *one*-dimensional noise distribution. In quantized domains, this allows us to efficiently derandomize the SSN method without requiring exponential time. We call this resulting deterministic method **DSSN**.

One may wonder why we do not simply use $s_1 = s_2 = s_3 ... = s_d$. While this can work, it leads to some undesirable properties when $\lambda > 0.5$. In particular, note that with probability $(2\lambda - 1)$, we would have all splitting values $s_i > 1$. This means that every element $\tilde{x}_i$ would be 0.5. In other words, with probability $(2\lambda - 1)/(2\lambda)$, $\tilde{\boldsymbol{x}} = 0.5 \cdot \mathbb{1}$. This restricts the expressivity of the smoothed classifier:

$$p(\mathbf{x}) = \frac{2\lambda - 1}{2\lambda} f(0.5 \cdot \mathbb{1}) + \frac{1}{2\lambda} \mathop{\mathbb{E}}_{\boldsymbol{s}<1} \left[f(\tilde{\mathbf{x}}) \right]. \tag{3.20}$$

This is the sum of a constant, and a function bounded in $[0, 1/(2\lambda)]$. Clearly, this is undesirable. By contrast, if we use an offset vector $\mathbf{v}$ as described above, not every component will have $s_i > 1$ simultaneously. This means that $\tilde{\boldsymbol{x}}$ will continue to be sufficiently expressive over the entire distribution of s_{base}.

3.3.2 Relationship to Uniform Additive Smoothing

In this section, we explain the relationship between SSN and uniform additive smoothing [5] with two main objectives:

1. We show that, for each element x_i, the *marginal* distributions of the noisy element $\tilde{x}_i$ of SSN and the noisy element $(x_i + \epsilon_i)$ of uniform additive smoothing are directly related to one another. However we show that, for large λ, the distribution of uniform additive smoothing $(x_i + \epsilon_i)$ has an undesirable property which SSN avoids. This creates large empirical improvements in certified robustness using SSN, demonstrating an additional advantage to our method separate from derandomization.

2. We show that additive uniform noise does *not* produce correct certificates when using arbitrary joint distributions of ϵ. This means that it cannot be easily derandomized in the way that SSN can.

3.3.2.1 Relationship between Marginal Distributions of $\tilde{x}_i$ and $(x_i + \epsilon_i)$

To see the relationship between uniform additive smoothing and SSN, we break the marginal distributions of each component of noised samples into cases (assuming $\lambda \geq 0.5$):

$$x_i + \epsilon_i \sim \begin{cases} \mathcal{U}(x_i - \lambda, 1 - \lambda) & \text{w. prob. } \frac{1-x_i}{2\lambda} \\ \mathcal{U}(1 - \lambda, \lambda) & \text{w. prob. } \frac{2\lambda-1}{2\lambda} \\ \mathcal{U}(\lambda, x_i + \lambda) & \text{w. prob. } \frac{x_i}{2\lambda} \end{cases} \tag{3.21}$$

$$\tilde{x}_i \sim \begin{cases} \frac{\mathcal{U}(x_i,1)}{2} & \text{w. prob. } \frac{1-x_i}{2\lambda} \\ \frac{1}{2} & \text{w. prob. } \frac{2\lambda-1}{2\lambda} \\ \frac{\mathcal{U}(1,x_i+1)}{2} & \text{w. prob. } \frac{x_i}{2\lambda} \end{cases} \tag{3.22}$$

We can see that there is a clear correspondence (which also justifies our re-use of the parameter λ.) In particular, we can convert the marginal distribution of uniform additive noise to the marginal distribution of SSN by applying a simple mapping: $\tilde{x}_i \sim g(x_i + \epsilon_i)$ where:

$$g(z) := \begin{cases} \frac{z+\lambda}{2} & \text{if } z < 1 - \lambda \\ \frac{1}{2} & \text{if } 1 - \lambda < z < \lambda \\ \frac{z-\lambda+1}{2} & \text{if } z > \lambda \end{cases} \tag{3.23}$$

For $\lambda = 0.5$, this is a simple affine transformation:

$$\tilde{x}_i \sim 1/2(x_i + \epsilon_i) + 1/4 \tag{3.24}$$

In other words, in the case of $\lambda = 0.5$, $\tilde{x}_i$ is also uniformly distributed. However, for $\lambda > 0.5$, Equation 3.21 reveals an unusual and undesirable property of using uniform additive noise: *regardless of the value of* x_i, there is always a fixed probability $\frac{2\lambda-1}{2\lambda}$ that the smoothed value $x_i + \epsilon_i$ is uniform on the interval $[1 - \lambda, \lambda]$. Furthermore, this constant probability represents the only case in which $(x_i + \epsilon_i)$ can assume values in this interval. These values therefore carry no information about x_i and are all equivalent to each other. However, if λ is large, this range dominates the total range of values of $x_i + \epsilon_i$ which are observed (See Figure 3.2-b.)

By contrast, in SSN, while there is still a fixed $\frac{2\lambda-1}{2\lambda}$ probability that the smoothed component $\tilde{x}_i$ assumes a "no information" value, this value is always *fixed* ($\tilde{x}_i = 1/2$). Empirically, this dramatically improves performance when λ is large. Intuitively, this is because when using uniform additive smoothing, the base classifier must *learn to ignore* a very wide range of values (all values in the interval $[1 - \lambda, \lambda]$) while in SSN, the base classifier only needs to learn to ignore a specific constant "no information" value $1/2$. See Figure 3.2 for a visual comparison of the two noise representations.[5]

3.3.2.2 Can Additive Uniform Noise Be Derandomized?

As shown above, in the $\lambda = 0.5$ case, SSN leads to marginal distributions which are simple affine transformations of the marginal distributions of the uniform additive smoothing. One might

[5]Note that this use of a "no information" value bears some similarity to the "ablation" value in Chapter 2 [58].

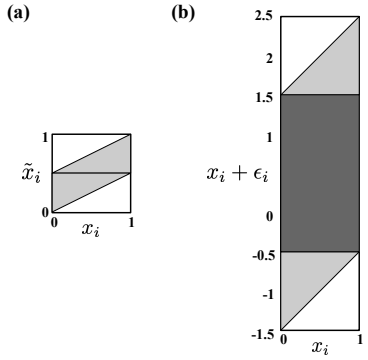


Figure 3.2: Range of noise values possible for each sample feature x_i, under (a) SSN, for any $\lambda \geq 0.5$ and (b) uniform additive smoothing, $\lambda = 1.5$. Possible pairs of clean and noise values are shown in grey (both light and dark). In uniform additive smoothing, note that all values of $x_i + \epsilon_i$ in the range [-0.5,1.5], shown in dark grey, can correspond to *any* value of x_i. This means that these values of $x_i + \epsilon_i$ carry no information about x_i whatsoever. By contrast, using SSN, only the value $\tilde{x}_i = 1/2$ has this property.

then wonder whether we can derandomize additive uniform noise in a way similar to DSSN. In particular, one might wonder whether arbitrary joint distributions of ϵ can be used to generate valid robustness certificates with uniform additive smoothing, in the same way that arbitrary joint distributions of $\boldsymbol{s}$ can be used with SSN. It turns out that this is not the case. We provide a counterexample (for $\lambda = 0.5$) below:

Proposition 3.1. *There exists a base classifier $f : \mathbb{R}^2 \to [0, 1]$ and a joint probability distribution $\mathcal{D}$, such that $\epsilon_1, \epsilon_2 \sim \mathcal{D}$ has marginals $\epsilon_1 \sim \mathcal{U}(-0.5, 0.5)$ and $\epsilon_2 \sim \mathcal{U}(-0.5, 0.5)$ where for*

$$p(\boldsymbol{x}) := \mathop{\mathbb{E}}_{\epsilon \sim \mathcal{D}} \left[f(\boldsymbol{x} + \epsilon) \right], \tag{3.25}$$

*$p(.)$ is **not** 1-Lipschitz with respect to the ℓ_1 norm.*

Proof. Consider the base classifier $f(\boldsymbol{z}) := \mathbf{1}_{z_1 > 0.4 + z_2}$, and let ϵ be distributed as $\epsilon_1 \sim \mathcal{U}(-0.5, 0.5)$ and $\epsilon_2 = \epsilon_1$. Consider the points $\boldsymbol{x} = [0.8, 0.2]^T$ and $\boldsymbol{x'} = [0.6, 0, 4]^T$. Note that $\|\delta\|_1 = 0.4$.

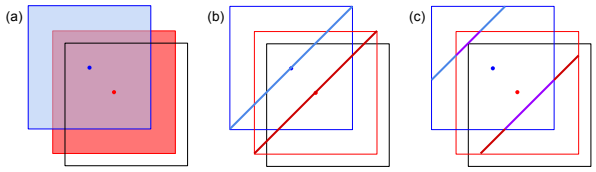


Figure 3.3: Comparison of independent uniform additive noise, correlated uniform additive noise, and correlated SSN, in $\mathbb{R}^2$ for $\lambda = 0.5$. In all figures, the blue and red points represent points $\boldsymbol{x}$ and $\boldsymbol{x}'$ and the black border represents the range $[0,1]^2$. (a) Distributions of $\boldsymbol{x} + \epsilon$ and $\boldsymbol{x}' + \epsilon$ for independent uniform additive noise. The robustness guarantee relies on the significant overlap of the shaded regions, representing the sampled distributions. Note that by Equation 3.24, these are also the distributions of for $2\tilde{\boldsymbol{x}} - \mathbb{1}/2$ and $2\tilde{\boldsymbol{x}}' - \mathbb{1}/2$ using SSN with s_1 and s_2 distributed independently. (b) Using correlated additive noise ($\epsilon_1 = \epsilon_2$) does *not* produce an effective robustness certificate: the sampled distributions $\boldsymbol{x}+\epsilon$ and $\boldsymbol{x}'+\epsilon$ (blue and red lines) do not overlap. (c) Using correlated splitting noise ($s_1 = s_2$) produces an effective robustness certificate, because distributions of $\tilde{\boldsymbol{x}}$ and $\tilde{\boldsymbol{x}}'$ overlap significantly. Here, for consistency in scaling, we show the distributions of $2\tilde{\boldsymbol{x}} - \mathbb{1}/2$ and $2\tilde{\boldsymbol{x}}' - \mathbb{1}/2$ (blue line and red line), with the overlap shown as purple. Note that this is a *one-dimensional* smoothing distribution, and therefore can be efficiently derandomized.

However,

$$\begin{aligned} p(\boldsymbol{x}) &= \mathop{\mathbb{E}}_{\epsilon}[f(\boldsymbol{x}+\epsilon)] = \mathop{\mathbb{E}}_{\epsilon_1}[f(.8+\epsilon_1, .2+\epsilon_1)] = 1 \\ p(\boldsymbol{x}') &= \mathop{\mathbb{E}}_{\epsilon}[f(\boldsymbol{x}'+\epsilon)] = \mathop{\mathbb{E}}_{\epsilon_1}[f(.6+\epsilon_1, .4+\epsilon_1)] = 0 \end{aligned} \tag{3.26}$$

Thus, $|p(\boldsymbol{x}) - p(\boldsymbol{x}')| > \|\delta\|_1$. □

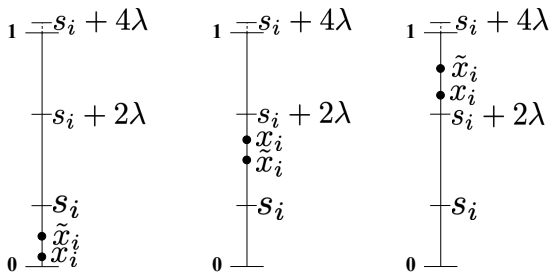


Figure 3.4: Example of $\tilde{x}_i$ in the $\lambda < 0.5$ case. In this case, the interval $[0,1]$ is split into sub-intervals $[0, s_i]$, $(s_i, s_i + 2\lambda]$, and $(s_i + 2\lambda, 1]$. $\tilde{x}_i$ is assigned to the middle of whichever of these intervals x_i falls into.

In the appendix, we provide intuition for this, by demonstrating that despite having similar

marginal distributions, the *joint* distributions of $\tilde{\boldsymbol{x}}$ and $(\boldsymbol{x}+\epsilon)$ which can be generated by SSN and additive uniform noise, respectively, are in fact quite different. An example is shown in Figure 3.3.

3.3.3 General Case, including $\lambda < 0.5$

In the case $\lambda < 0.5$, we split the $[0,1]$ interval not only at $s_i \in [0, 2\lambda]$, but also at every value $s_i + 2\lambda n$, for $n \in \mathbb{N}$. An example is shown in Figure 3.4. Note that this formulation covers the $\lambda \geq 0.5$ case as well (the splits for $n \geq 1$ are simply not relevant).

Theorem 3.2 (General Case)**.** *For any $f : \mathbb{R}^d \to [0,1]$, and $\lambda > 0$ let $\boldsymbol{s} \in [0, 2\lambda]^d$ be a random variable, with a fixed distribution such that:*

$$s_i \sim \mathcal{U}(0, 2\lambda), \quad \forall i. \tag{3.27}$$

*Note that the components $s_1, ..., s_d$ are **not** required to be distributed independently from each other. Then, define:*

$$\tilde{x}_i := \frac{\min(2\lambda\lceil\frac{x_i - s_i}{2\lambda}\rceil + s_i, 1)}{2} + \frac{\max(2\lambda\lceil\frac{x_i - s_i}{2\lambda} - 1\rceil + s_i, 0)}{2}, \quad \forall i \tag{3.28}$$

$$p(\boldsymbol{x}) := \mathop{\mathbb{E}}_{\boldsymbol{s}}[f(\tilde{\boldsymbol{x}})]. \tag{3.29}$$

Then, $p(.)$ is $1/(2\lambda)$-Lipschitz with respect to the ℓ_1 norm.

The proof for this case, as well as its derandomization, are provided in the appendix. As with the $\lambda \geq 0.5$ case, the derandomization allows for $p(\boldsymbol{x})$ to be computed exactly using $2\lambda q$ evaluations of f.

3.4 Experiments

We evaluated the performance of our method on CIFAR-10 and ImageNet datasets, matching all experimental conditions from [5] as closely as possible (further details are given in the appendix.) Certification performance data is given in Table 3.1 for CIFAR-10 and Figure 3.5 for Imagenet. Note that instead of using the hyperparameter λ, we report experimental results in terms of $\sigma = \lambda/\sqrt{3}$: this is to match [5], where this gives the standard deviation of the uniform noise.

We find that DSSN significantly outperforms Yang et al. [5] on both datasets, particularly when certifying for large perturbation radii. For example, at $\rho = 4.0$, DSSN provides a 36% certified accuracy on CIFAR-10, while uniform additive noise provides only 27% certified accuracy. In addition to these numerical improvements, DSSN certificates are *exact* while randomized certificates hold only with *high-probability*. Following Yang et al. [5], all certificates reported here for randomized methods hold with 99.9% probability: there is no such failure rate for DSSN.

Additionally, the certification runtime of DSSN is reduced compared to randomized methods. Although in contrast to [5], our certification time depends on the noise level, the fact that Yang et al. [5] uses 100,000 smoothing samples makes our method much faster even at the largest tested noise levels. For example, on CIFAR-10 at $\sigma = 3.5$, using a single NVIDIA 2080 Ti GPU, we achieve an average runtime of 0.41 seconds per image, while Yang et al. [5]'s method requires 13.44 seconds per image.

Yang et al. [5] tests using both standard training on noisy samples as well as stability training [35]: while our method dominates in both settings, we find that the stability training leads to less of an improvement in our methods, and is in some cases detrimental. For example,

in Table 3.1, the best certified accuracy is always higher under stability training for uniform additive noise, while this is not the case for DSSN at $\rho < 3.0$. Exploring the cause of this may be an interesting direction for future work.[6]

In Figure 3.6, we compare the uniform additive smoothing method to DSSN, as well the *randomized* form of SSN with independent splitting noise. At mid-range noise levels, the primary benefit of our method is due to derandomization; while at large noise levels, the differences in noise representation discussed in Section 3.3.2.1 become more relevant. In the appendix, we provide complete certification data at all tested noise levels, using both DSSN and SSN with independent noise, as well as more runtime data. Additionally we further explore the effect of the noise representation: given that Equation 3.23 shows a simple mapping between (the marginal distributions of) SSN and uniform additive noise, we tested whether the gap in performance due to noise representations can be eliminated by a "denoising layer", as trained in [79]. We did not find evidence of this: the gap persists even when using denoising.

3.5 Prior Works on Derandomized Smoothing

While this work is, to the best of our knowledge, the first to propose a derandomized version of a randomized smoothing algorithm to certify for a norm-based threat model without resricting the base classifier or requiring time exponential in the dimension d of the input, prior deterministic "randomized smoothing" certificates have been proposed. These include:

- **Certificates for non-norm (ℓ_0-like) threat models**. This includes certificates against patch adversarial attacks such as [29] (Chapter 6 of this dissertation) and subsequent improve-

[6] On CIFAR-10, Yang et al. [5] also tests using semi-supervised and transfer learning approaches which incorporate data from other datasets. We consider this beyond the scope of this work where we consider only the supervised learning setting.

	ρ =0.5	ρ =1.0	ρ =1.5	ρ =2.0	ρ =2.5	ρ =3.0	ρ =3.5	ρ =4.0
Uniform Additive Noise	70.54% (83.97% @ σ=0.5)	58.43% (78.70% @ σ=1.0)	50.73% (73.05% @ σ=1.75)	43.16% (73.05% @ σ=1.75)	33.24% (69.56% @ σ=2.0)	25.98% (62.48% @ σ=2.5)	20.66% (53.38% @ σ=3.5)	17.12% (53.38% @ σ=3.5)
Uniform Additive Noise (+Stability Training)	71.09% (78.79% @ σ=0.5)	60.36% (74.27% @ σ=0.75)	52.86% (65.88% @ σ=1.5)	47.08% (63.32% @ σ=1.75)	42.26% (57.49% @ σ=2.5)	38.55% (57.49% @ σ=2.5)	33.76% (57.49% @ σ=2.5)	27.12% (57.49% @ σ=2.5)
DSSN - Our Method	**72.25%** (81.50% @ σ=0.75)	**63.07%** (77.85% @ σ=1.25)	**56.21%** (71.17% @ σ=2.25)	**51.33%** (67.98% @ σ=3.0)	**46.76%** (65.40% @ σ=3.5)	42.66% (65.40% @ σ=3.5)	38.26% (65.40% @ σ=3.5)	33.64% (65.40% @ σ=3.5)
DSSN - Our Method (+Stability Training)	71.23% (79.00% @ σ=0.5)	61.04% (71.29% @ σ=1.0)	54.21% (66.04% @ σ=1.5)	49.39% (64.26% @ σ=1.75)	45.45% (59.88% @ σ=2.5)	**42.67%** (57.16% @ σ=3.0)	**39.46%** (56.29% @ σ=3.25)	**36.46%** (54.96% @ σ=3.5)

Table 3.1: Summary of results for CIFAR-10. Matching Yang et al. [5], we test on 15 noise levels ($\sigma \in \{0.15, 0.25n \text{ for } 1 \leq n \leq 14\}$). We report the best certified accuracy at a selection of radii ρ, as well as the clean accuracy and noise level of the associated classifier. Our method dominates at all radii, although stability training seems to be less useful for our method. Note that these statistics are based on reproducing Yang et al. [5]'s results; they are all within ± 1.5 percentage points of Yang et al. [5]'s reported statistics.

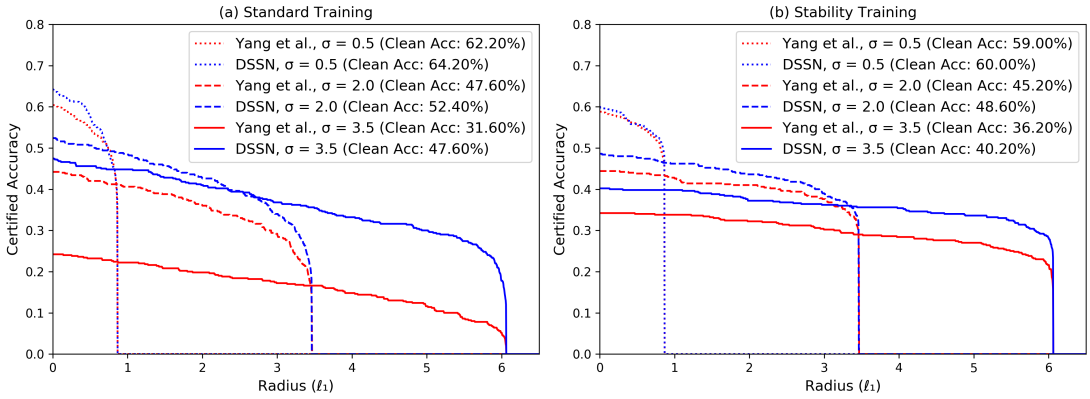


Figure 3.5: Results on ImageNet. We report results at three noise levels, with and without stability training. Our method dominates in all settings: however, especially at large noise, stability training seems to *hurt* our clean accuracy, rather than help it.

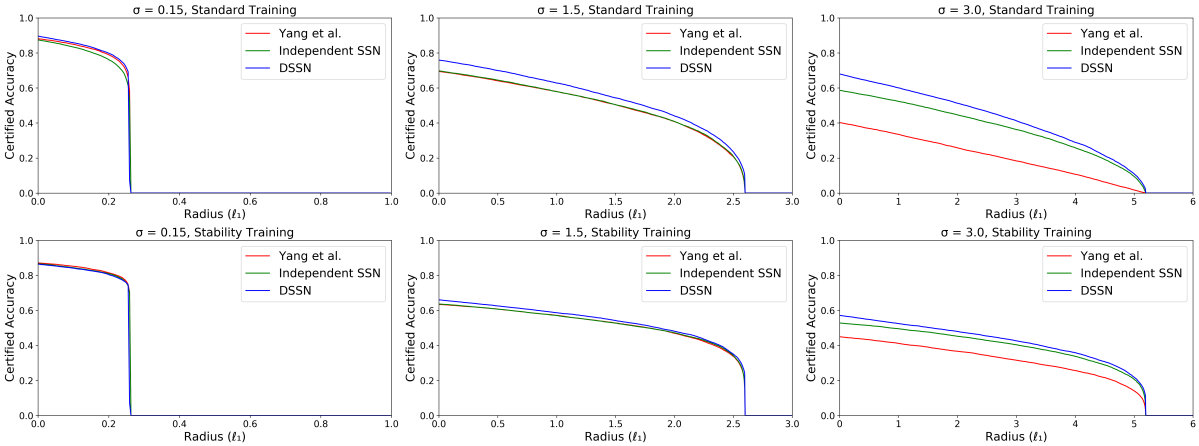


Figure 3.6: Comparison on CIFAR-10 of additive smoothing [5] to DSSN, as well as SSN with *random, independent* splitting noise, using the estimation scheme from [5]. At very small levels of noise ($\sigma = 0.15$), there is little difference between the methods: in fact, with stability training, additive smoothing slightly outperforms DSSN. At intermediate noise levels, additive noise and independent SSN perform very similarly, but DSSN outperforms both. This suggests that, at this level, the primary benefit of DSSN is to eliminate estimation error (Section 3.2). At high noise levels, the largest gap is between additive noise and independent SSN, suggesting that in this regime, most of the performance benefits of DSSN are due to improved base classifier performance (Section 3.3.2.1).

ments on it such as [30]); as well as poisoning attacks under a label-flipping [6] or whole-sample insertion/deletion [74] (Chapter 7 of this dissertation) threat-model. These threat models are "ℓ_0-like" because the attacker entirely corrupts some portion of the data, rather than just distorting it. Our work in Chapters 6 and 7, both published before this chapter was originally written, deal with this by ensuring that only a bounded fraction of base classifications can possibly be simultaneously exposed to any of the corrupted data. In the respective cases of patch adversarial attacks and poisoning attacks, it is shown that this can be done with a finite number of base classifications. Rosenfeld et al. [6]'s method, by contrast, is based on the randomized ℓ_0 certificate proposed by Lee et al. [1], and is discussed below.

- **Certificates for restricted classes of base classifiers**. This includes k-nearest neighbors [75] (for ℓ_2 poisoning attacks) and linear models [6] (for label-flipping poisoning attacks).

In these cases, existing randomized certificates are evaluated exactly for a restricted set of base classifier models. (Cohen et al. [32] and Lee et al. [1]'s methods, respectively.) It is notable that these are both poisoning certificates: in the poisoning setting, where the corrupted data is the training data, true randomized smoothing is less feasible, because it requires training very large ensemble of classifiers to achieve desired statistical properties. Weber et al. [75] also attempts this directly, however.

- **Certificates requiring time exponential in dimension** d. This includes, in particular, a concurrent work, [76], which provides deterministic ℓ_2 certificates. In order to be practical, this method requires that the first several layers of the network be Lipschitz-bounded by means other than smoothing. The "smoothing" is then applied only in a low-dimensional space. The authors note that this method is unlikely to scale to ImageNet.

Chapter 4: Provable Adversarial Robustness for Fractional ℓ_p Threat Models[1]

In this work, we extend the results of the previous chapter to cover ℓ_p "norms" for $p < 1$. More precisely, we develop a deterministic smoothing method that guarantees Lipschitzness with respect to the ℓ_p^p metric for $p \in (0, 1)$, defined as:

$$\ell_p^p(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{d} |x_i - y_i|^p \tag{4.1}$$

This immediately provides ℓ_p^p-metric certificates, which can be converted to ℓ_p certificates by simply raising the radius to the power of $1/p$. Our technique is in fact more general than this, and can be applied to ensure the Lipschitz continuity of a function to a larger family of "elementwise-concave metrics" defined as the sum of concave functions of coordinate differences.

While not as frequently encountered as other ℓ_p norms, ℓ_p "norms" for $p \in (0, 1)$ (which are in fact quasi-norms, because they violate the triangle inequality) are used in several machine-learning applications. Some known applications of ℓ_p, $(p < 1)$ "norms" in machine learning include clustering [81, 82], dimensionality reduction [83], and image retrieval [84]. While ℓ_p, $p \in (0, 1)$ adversarial attacks have yet to emerge in practice, [85] have recently proposed an algorithm for ℓ_p-constrained optimization with $p < 1$: the authors mention that this could be used to generate adversarial examples. This suggests that developing defenses to such attacks

[1] A form of chapter has been published in AISTATS 2022 [80].

is a valuable exercise. Fractional ℓ_p threat models can also be thought of as "soft" versions of the widely-considered ℓ_0 threat model, allowing the attacker, in addition to entirely changing some pixels, to slightly impact additional pixels at a "discount", without paying the full price in perturbation budget for modifying them. This may be relevant, for example, in physical ℓ_0 attacks. Furthermore, readers may find other uses for ensuring that a trained function is ℓ_p^p-Lipschitz for $p < 1$.

Our technique inherits some of the limitations of DSSN: notably that the deterministic variant applies exclusively to bounded, quantized input domains: that is, inputs where the value in each dimension only assumes values in $[0, 1]$ which are multiples of $1/q$, for some quantization parameter q. However, this applies to many domains of practical interest in machine learning, such as image classification, which typically uses $q = 255$. Because the image domain is perhaps the most widely-studied domain of adversarial robustness, this restriction does not pose a significant limitation in practice. (Even in their randomized variants, both DSSN and this algorithm assume bounded input domains: that is, inputs $\boldsymbol{x} \in [0, 1]^d$.)

In Section 4.5, we also consider the $p = 0$ limit of our algorithm. In that case, we show that our method simplifies to essentially a deterministic variant of the "randomized ablation" ℓ_0 smoothing defense proposed in Chapter 2 [58]. In fact, this deterministic variant was implicitly discussed in Chapter 7 [74] (published before this chapter was written), where a specialized form of it is used to provably defend against poisoning attacks. Here, we apply it to evasion attacks directly. While this simplified ℓ_0 defense perhaps somewhat under-performs the randomized variant, it provides deterministic certificate results at greatly reduced runtime.

In summary, in this work, we propose a novel, deterministic method for ensuring that a trained function on bounded, quantized inputs is Lipschitz with respect to any ℓ_p^p metric for $p \in$

$(0, 1)$. This has immediate applications to provable adversarial robustness: we use our method to generate robustness certificates for fractional ℓ_p quasi-norms on CIFAR-10 and ImageNet.

4.1 Notation and Preliminaries

We first specify some notation. Let $\mathcal{U}(a, b)$ represent the uniform distribution on the range $[a, b]$, and let Beta(α, β) represent the beta distribution with parameters α, β. Let $\lfloor \cdot \rfloor$ and $\lceil \cdot \rceil$ represent the floor and ceiling functions. Let $[d]$ be the set $1, ..., d$. Let $\mathbf{1}_{\text{(condition)}}$ be the indicator function. As in Chapter 3, we use '$a \pmod b$' with real-valued a, b to indicate $a - b\lfloor a/b \rfloor$.

Next, we define the general set of metrics our technique applies to, of which ℓ_p^p metrics are an example.

Definition 4.1 (Elementwise-concave metric (ECM))**.** *For any $\boldsymbol{x}, \boldsymbol{y}$, let $\delta_i := |x_i - y_i|$. An elementwise-concave metric (ECM) is a metric on $[0, 1]^d$ in the form:*

$$d(\boldsymbol{x}, \boldsymbol{y}) := \sum_{i=1}^{d} g_i(\delta_i), \tag{4.2}$$

where $g_1, ..., g_d \in [0, 1] \to [0, 1]$ are increasing, concave functions with $g_i(0) = 0$.

Note that the ℓ_p^p metrics with $p \leq 1$ are ECM's, with $\forall i,\ g_i(z) = z^p$. Note also that any distance function meeting the definition of an ECM is in fact a metric, unless some g_i is the zero function.

We also restate the main theorem from Chapter 3 [73], which this chapter extends upon:

Theorem 4.1 ([73])**.** *For any $f : [0, 1]^d \times [0, 1]^d \to [0, 1]$, and $\Lambda > 0$, let $\boldsymbol{s} \in [0, \Lambda]^d$ be a random*

variable, with a fixed distribution such that:

$$s_i \sim \mathcal{U}(0, \Lambda), \quad \forall i. \tag{4.3}$$

*Note that the components $s_1, ..., s_d$ are **not** required to be distributed independently from each other. Then, define:*

$$x_i^{upper} := \min(\Lambda\lceil \frac{x_i - s_i}{\Lambda} \rceil + s_i, 1) \ , \quad \forall i \tag{4.4}$$

$$x_i^{lower} := \max(\Lambda\lceil \frac{x_i - s_i}{\Lambda} \rceil + s_i - \Lambda, 0) \ , \quad \forall i \tag{4.5}$$

$$p(\boldsymbol{x}) := \mathop{\mathbb{E}}_{\boldsymbol{s}} \left[f(\boldsymbol{x}^{lower}, \boldsymbol{x}^{upper}) \right]. \tag{4.6}$$

Then, $p(.)$ is $1/\Lambda$-Lipschitz with respect to the ℓ_1 norm.

We provide a visual explanation of this theorem in Figure 4.1. The basic intuition is that the $[0, 1]$ domain of each dimension is divided into "bins", with dividers at each value $s_i + n\Lambda, \forall n \in \mathbb{N}$. Then, x_i^{lower} and x_i^{upper} are the lower- and upper-limits of the bin which x_i is assigned to. For two points $\boldsymbol{x}$ and $\boldsymbol{y}$, let $\delta_i := |x_i - y_i|$: the probability of a divider separating x_i and y_i is $\min(\delta_i/\Lambda, 1)$. this means that the probability that $(\boldsymbol{y}^{\text{lower}}, \boldsymbol{y}^{\text{upper}})$ differs from $(\boldsymbol{x}^{\text{lower}}, \boldsymbol{x}^{\text{upper}})$ is at most $\|x - y\|_1/\Lambda$. The Lipschitz property follows from this.

Note that we have modified the notation from the original statement of the theorem: in particular, we use Λ instead of 2λ.[2] Additionally, we pass both $\boldsymbol{x}^{\text{lower}}$ and $\boldsymbol{x}^{\text{upper}}$ to the base classifier f, even though these are redundant when Λ is fixed: this is because we are about to break this assumption. (We include a proof sketch in the modified notation in Appendix C.1.1.)

[2]Recall that in Chapter 3, we defined λ such that it corresponded directly to the noise level λ used in [51]: here, it will be more notationally convenient to used the scaled quantity $\Lambda := 2\lambda$

Note also that this is a *randomized* algorithm; we will discuss the derandomization in Section 4.3, where we present the derandomization of our proposed method.

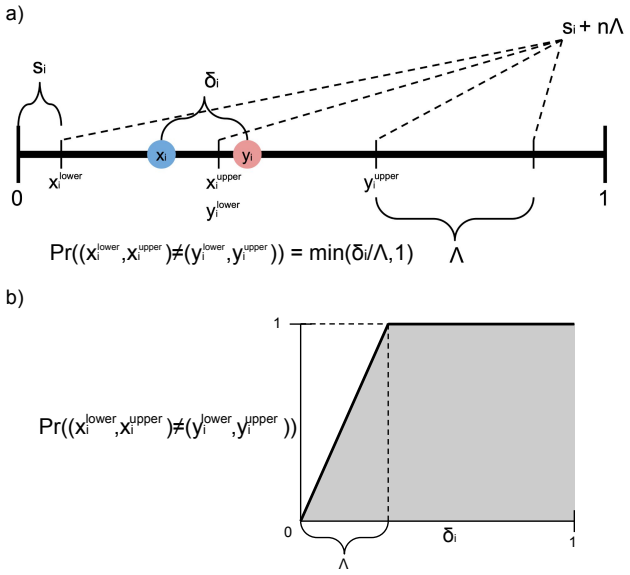


Figure 4.1: A visual explanation of Theorem 4.1 from Chapter 3, with new notation. (a) Whether x_i and y_i belong to the same bin depends on the value of the bin-divider offset s_i. However, because this is uniformly distributed, the probability that they are mapped to different bins is simply δ_i/Λ, if $\delta_i < \Lambda$, and 1 otherwise. (b) Graph of the probability that x_i and y_i are assigned to different bins, as a function of their difference δ_i.

4.2 Proposed Method

In this paper, we modify the algorithm described in Theorem 4.1 by allowing Λ itself to vary randomly in each dimension, according to a fixed distribution $\mathcal{D}_i$:

$$\begin{aligned} \Lambda_i &\sim \mathcal{D}_i \\ s_i &\sim \mathcal{U}(0, \Lambda_i) \end{aligned} \tag{4.7}$$

The reason for doing this is that, by mixing the smoothing distributions for various Λ in each dimension, the probability that x_i and y_i are assigned to different "bins" assumes a concave

relationship to their difference δ_i, as illustrated in Figure 4.2. In fact, by doing this, we are able to make the probability of splitting x_i and y_i to be any arbitrary smooth concave increasing function of δ_i, allowing us to enforce Lipschitzness with respect to arbitrary ECMs, as shown in the upcoming Theorem 4.2.

Note that, if we allow the support of $\mathcal{D}_i$ to be $(0, \infty)$, there is some redundancy in the noise model specified by Equation 4.7: in particular, whenever $\Lambda_i > 1$, there are at most two bins, with the single divider s_i uniformly on the range $[0, 1]$ with probability $1/\Lambda_i$ and otherwise with the entire range $[0, 1]$ falling into one bin. For simplicity, therefore, we can allow the support of $\mathcal{D}_i$ to be $(0, 1] \cup \{\infty\}$, where $\Lambda_i = \infty$ signifies to consider the entire domain as one bin. Formally, our noise process is defined as follows:

Definition 4.2 (Variable-Λ smoothing)**.** *For any $f : [0,1]^d \times [0,1]^d \to [0,1]$, and distribution $\mathcal{D} = \{\mathcal{D}_1, ...\mathcal{D}_d\}$, such that each $\mathcal{D}_i$ has support $(0, 1] \cup \{\infty\}$, let:*

$$\Lambda_i \sim \mathcal{D}_i \tag{4.8}$$

If $\Lambda_i = \infty$, then $x_i^{upper} := 1$, $x_i^{lower} := 0$, otherwise:

$$s_i \sim \mathcal{U}(0, \Lambda_i) \tag{4.9}$$

$$x_i^{upper} := \min(\Lambda_i \lceil \frac{x_i - s_i}{\Lambda_i} \rceil + s_i, 1) \tag{4.10}$$

$$x_i^{lower} := \max(\Lambda_i \lceil \frac{x_i - s_i}{\Lambda_i} \rceil + s_i - \Lambda_i, 0) \tag{4.11}$$

$$\tag{4.12}$$

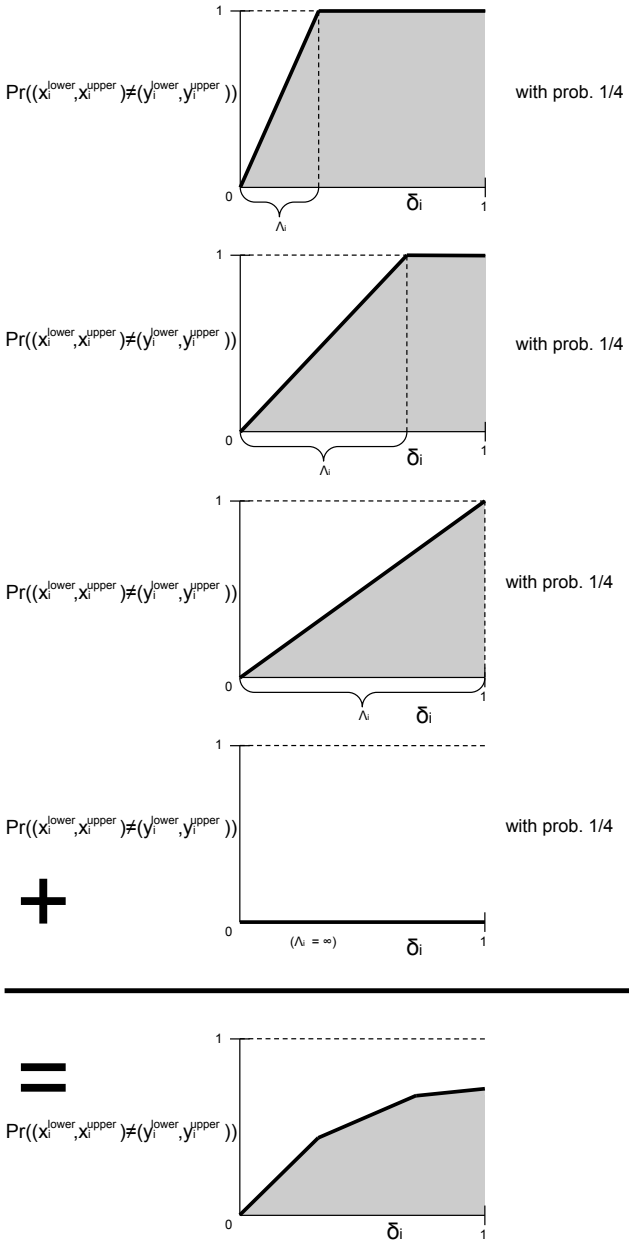


Figure 4.2: A mixture of various values of Λ creates a concave relationship between the probability that x_i and y_i are distinguishable to the base classifier and the difference δ_i between their values. This is because the slope of each of the curves in the mixture goes to zero at $\delta_i = \Lambda_i$.

The smoothed function is defined as:

$$p_{\mathcal{D},f}(\boldsymbol{x}) \coloneqq \mathop{\mathbb{E}}_{\boldsymbol{s}}\left[f(\boldsymbol{x}^{lower}, \boldsymbol{x}^{upper})\right]. \tag{4.13}$$

Note that we make no assumptions about the joint distributions of Λ or of $\boldsymbol{s}$.

We can now present our main theorem, describing how to ensure Lipschitzness with respect to an ECM:

Theorem 4.2. *Let $d(\cdot,\cdot)$ be an ECM defined by concave functions $g_1, ..., g_d$. Let $\mathcal{D}$ and $f(\cdot)$ be the Λ-distribution and base function used for Variable-Λ smoothing, respectively. Let $\boldsymbol{x}, \boldsymbol{y} \in [0,1]^d$ be two points. For each dimension i, let $\delta_i \coloneqq |x_i - y_i|$. Then:*

(a) The probability that $(x_i^{lower}, x_i^{upper}) \neq (y_i^{lower}, y_i^{upper})$ is given by $\Pr_i^{split}(\delta_i)$, where:

$$\Pr_i^{split}(z) \coloneqq \Pr_{\mathcal{D}_i}(\Lambda_i \leq z) + z\mathbb{E}_{\mathcal{D}_i}\left[\frac{\mathbf{1}_{(\Lambda_i \in (z,1])}}{\Lambda_i}\right] \tag{4.14}$$

(b) If $\forall i \in [d]$ and $\forall z \in [0,1]$,

$$\Pr_i^{split}(z) \leq g_i(z), \tag{4.15}$$

then, the smoothed function $p_{\mathcal{D},f}(\cdot)$ is 1-Lipschitz with respect to the metric $d(\cdot,\cdot)$.

(c) Suppose g_i is continuous and twice-differentiable on the interval $(0,1]$. Let $\mathcal{D}_i$ be constructed as follows:

- *On the interval $(0,1)$, Λ_i is distributed continuously, with pdf function:*

$$pdf_{\Lambda_i}(z) = -zg_i''(z) \tag{4.16}$$

- $\Pr(\Lambda_i = 1) = g_i'(1)$
- $\Pr(\Lambda_i = \infty) = 1 - g_i(1)$

then,

$$\Pr_i^{split}(z) = g_i(z) \quad \forall z \in [0,1]. \tag{4.17}$$

If all $\mathcal{D}_i$ are constructed this way, then the conclusion of part (b) above applies.

Here, $\Pr_i^{\text{split}}(\delta_i)$ represents the probability that the base classifier is given the information necessary to distinguish x_i from y_i; in order to ensure that the base classifier receives as much information as possible, we would like to design $\mathcal{D}_i$ to make $\Pr_i^{\text{split}}(z)$ as large as possible, for all $z \in [0,1]$. However, if we want our smoothed classifier to have the desired Lipschitz property, $\Pr_i^{\text{split}}(z)$ can be no larger than $g_i(z)$, as stated in part (b) of the theorem. Part (c) of the theorem shows how to design $\mathcal{D}_i$ such that $\Pr_i^{\text{split}}(z)$ takes exactly its maximum allowed value, $g_i(z)$, everywhere.

We can apply part (c) of Theorem 4.2 to derive smoothing distributions for Lipschitzness on fractional-p ℓ_p^p metrics, simply by taking $g_.(z) = \frac{z^p}{\alpha}$:

Corollary 4.1. *For all $p \in (0,1]$, $\alpha \in [1,\infty)$, if we perform Variable-Λ smoothing with all Λ_i's distributed identically (but not necessarily independently) as follows:*

$$\begin{aligned} \Lambda_i &\sim \textit{Beta}(p,1), \ \textit{with prob.} \ \frac{1-p}{\alpha} \\ \Lambda_i &= 1, \ \textit{with prob.} \ \frac{p}{\alpha} \\ \Lambda_i &= \infty, \ \textit{with prob.} \ 1 - \frac{1}{\alpha} \end{aligned} \tag{4.18}$$

then, the resulting smoothed function will be $1/\alpha$-Lipschitz with respect to the ℓ_p^p metric[3].

We use the fact that 1-Lipschitzness with respect to $d(\cdot,\cdot)/\alpha$ is equivalent to $1/\alpha$-Lipschitzness with respect to $d(\cdot,\cdot)$. We can verify that taking $p = 1$, $\alpha = \Lambda$ recovers Theorem 4.1 for $\Lambda \geq 1$.

4.3 Quantization and Derandomization

While the previous section describes a *randomized smoothing* scheme for guaranteeing ℓ_p^p-Lipschitz behavior of a function, in this section, we would like to derandomize this algorithm to ensure an exact, rather than high-probability, guarantee. For the fixed-Λ case, Chapter 3 derives such a derandomization in a two-step argument. First, a *quantized* form of Theorem 4.1 is proposed. To explain this, we reintroduce some notation from Chapter 3. Let q be the number of quantizations (e.g., 255 for images). Let

$$[a,b]_{(q)} := \left\{ i/q \;\middle|\; \lceil aq \rceil \leq i \leq \lfloor bq \rfloor \right\}. \tag{4.19}$$

For example, $[0,1]_{(q)}$ represents the set $\{0, \frac{1}{q}, \frac{2}{q}, ..., \frac{q-1}{q}, 1\}$. Departing slightly from Chapter 3, we define $\mathcal{U}_{(q)}(a,b)$ as the uniform distribution on the set $[a, b - \frac{1}{q}]_{(q)} + \frac{1}{2q}$. (e.g., $\mathcal{U}_{(q)}(0,1)$ is uniform on $\{\frac{1}{2q}, \frac{3}{2q}, ...\frac{2q-1}{2q}\}$: these are the *midpoints between* the quanizations in $[0,1]_{(q)}$).

Chapter 3 shows that Theorem 4.1 applies essentially unchanged in the quantized case: in particular, if the domain of $p(\cdot)$ is restricted to $[0,1]_{(q)}$, (and assuming that Λ is a multiple of 1/q)

[3] If we desire *weaker* Lipschitz guarantees, i.e., with $\alpha < 1$, the assumptions of Theorem 4.2-c no longer hold. We deal with this case in Appendix C.1.3.1, but it is not particularly relevant for our application: note that, as long as the classifier's accuracy remains high, then certificates scale with $1/\alpha$, so larger α is generally desirable. In our experiments, we find that the classifier's accuracy remains high even for α much greater than 1.

then the theorem still applies when Equation 4.3 is replaced with:

$$s_i \sim \mathcal{U}_{(q)}(0, \Lambda), \quad \forall i. \tag{4.20}$$

When this quantized form is used, there are only a discrete number of outcomes (= Λq) for each s_i.

Building on this, the second step in the argument is to leverage the fact that Theorem 4.1 makes no assumption on the joint distribution of s_i's to *couple* all of the elements of $\boldsymbol{s}$. In particular, s_i's are set to have fixed offsets from one another (mod Λ). In other words, the outcomes for each s_i are *cyclic permutations* of each other. This preserves the property that each s_i is uniformly distributed, while also ensuring that there are now only Λq outcomes of the smoothing process *in total*. Then expectation in Equation 4.13 can be evaluated exactly and efficiently (See Figure 4.3-a.)

For our Variable-Λ method, we use a similar strategy for derandomization: We quantize the smoothing process in a similar way, modifying Definition 4.2 by redefining the support of $\mathcal{D}_i$ as $[\frac{1}{q}, 1]_{(q)} \cup \{\infty\}$ and replacing Equation 4.10 with:

$$s_i \sim \mathcal{U}_{(q)}(0, \Lambda_i). \tag{4.21}$$

We also define a quanitized version of ECM's as a metric on $[0, 1]^d_{(q)}$ where the domain of each g_i is restricted to $[0, 1]_{(q)}$. This yields a quantized version of Theorem 4.2 which we spell out fully in Appendix C.1.4. The most significant difference occurs in part (c) where we use quantized forms of derivatives:

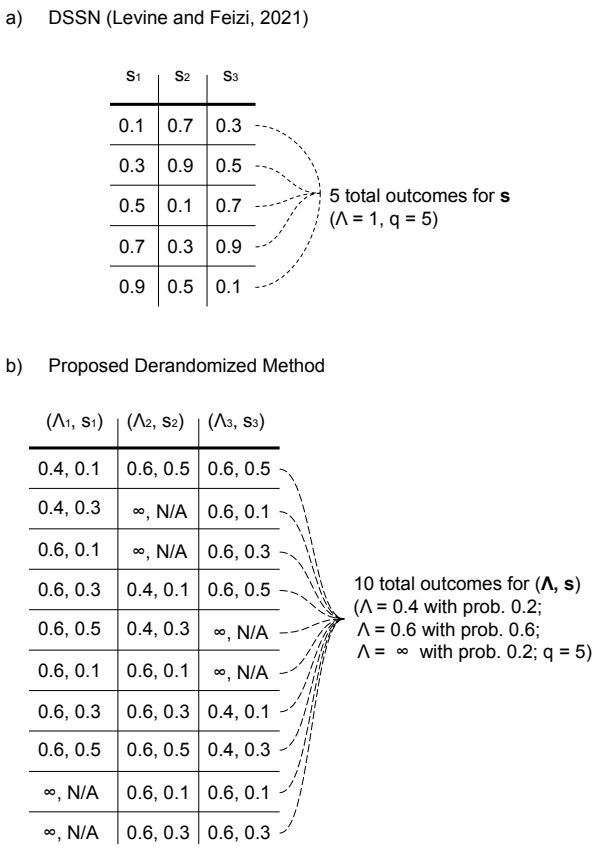


Figure 4.3: (a) "Fixed offset" method of sampling outcomes in DSSN (Chapter 3). In this case, the fixed offset is that $s_1 = s_2 - 0.6 = s_3 - 0.2 \pmod{\Lambda}$ Note that in the sample of 5 outcomes, each s_i is uniform on $\{0.1, 0.3, 0.5, 0.7, 0.9\}$, which is to say, $s_i \sim \mathcal{U}_{(5)}(0, 1)$, as desired. (b) Fixed-offset sampling applied to variable-Λ smoothing, for a given distribution of Λ. For each (Λ_i, s_i), we list out each possible outcome, in some cases repeated in order to achieve the desired distribution over Λ. Outcomes are then cyclically permuted for each dimension i to define the coupling. As in DSSN, the offsets for the cyclic permutations are arbitrary, but fixed throughout training and testing. Specifically, the offsets are chosen pseudorandomly using a fixed seed. We use a seed of 0 for experiments in the main text; other values are explored on CIFAR-10 in Appendix C.6. Note that Theorem 4.3 does not require cyclic permutations: choosing arbitrary permutations for each s_i would work. However, storing such arbitrary permutations for each dimension would be highly memory-intensive. In Appendix C.7, we show (at small-scale: CIFAR-10) that using arbitrary (pseudorandom) permutations as opposed to cyclic permutations confers no practical benefit.

Theorem 4.3 (c). *If $\mathcal{D}_i$ is constructed as follows:*

- *On the interval $[\frac{1}{q}, \frac{q-1}{q}]_{(q)}$, Λ_i is distributed as:*

$$\Pr(\Lambda_i = z) = -qz\Big[g_i(z - \frac{1}{q}) + g_i(z + \frac{1}{q}) - 2g_i(z)\Big] \quad \forall z \in [\frac{1}{q}, \frac{q-1}{q}]_{(q)} \tag{4.22}$$

- $\Pr(\Lambda_i = 1) = q\Big[g_i(1) - g_i(\frac{q-1}{q})\Big]$
- $\Pr(\Lambda_i = \infty) = 1 - g_i(1)$

then

$$\Pr_i^{split}(z) = g_i(z), \quad \forall z \in [0, 1]. \tag{4.23}$$

Now that we have defined a quantized version of our smoothing method, we attempt the coupling step (using the fact that Theorem 4.2 also makes no assumptions about joint distributions of $\boldsymbol{s}$ or Λ). However, this presents greater challenges than the ℓ_1 case. In the ℓ_1 case, all outcomes for each s_i occur with equal probability $1/(\Lambda q)$ so we can arbitrarily associate each outcome for s_1 with a unique outcome for s_2, and so on (for example using the fixed offset method described above). However, Equation 4.22 assigns real-number probabilities to each value of Λ_i. This means that the outcomes (Λ_i, s_i) for each dimension occur with non-uniform probabilities, making the coupling process more difficult.

One naive solution (at least in the case where g_i's are all the same function, for example for ℓ_p^p metrics) is to couple the Λ_i's such that they are all equal to one another; in other words, the

sampling process becomes:

$$\begin{aligned} \Lambda &\sim \mathcal{D}. \\ s_i &\sim \mathcal{U}(0, \Lambda) \quad \forall i \end{aligned} \tag{4.24}$$

We can then apply the fixed-offset coupling of $\boldsymbol{s}$ for each possible value of Λ, evaluating $q\Lambda$ outcomes for each value. We then exactly compute the final expectation $p(\cdot)$ as the *weighted* average of $f(\cdot)$ over these outcomes, with the weights for each Λ being determined by Theorem 4.3-c. However, this naive "Global Λ" method underperforms in practice (see Figure 4.4) and has significant theoretical drawbacks (e.g., notice that this method simply produces the average of several ℓ_1-Lipschitz functions). We explain this further in Appendix C.2.

What we do instead is to design $\mathcal{D}$ such that, for some constant integer B, all outcomes for (Λ_i, s_i) each have probability in the form n/B, where $n \in \mathbb{N}$. By repeating each outcome n times, this allows us to generate a list of B total outcomes which occur with uniform probability. We then couple these using cyclic permutations as in Chapter 3, so that we require a total of B smoothing samples (See Figure 4.3-b.)

Note that the distribution $\mathcal{D}$ given by Equation 4.22 is not necessarily of this form. However, even though the distribution given by Equation 4.22 is in some sense "optimal" in that it causes $\Pr^{\text{split}}(z)$ to perfectly match $g(z)$, thereby providing the most information to the base classifier, it is only necessary for the Lipschitz guarantee that $\Pr^{\text{split}}(z)$ is nowhere greater than $g(z)$. It turns out (as explained in full detail in Appendix C.3) that for a fixed budget B, finding a distribution over Λ with all outcomes in the form n/B such that $\Pr^{\text{split}}(z)$ approximates but never exceeds a given $g(z)$ can be formulated as a mixed integer linear program. Solving these MILP's

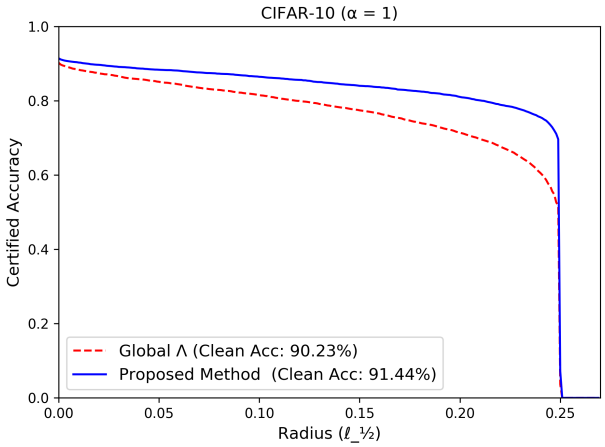


Figure 4.4: Using a global value for Λ as suggested in Equation 4.24 leads to suboptimal certified robustness.

yields distributions for Λ that cause $\Pr^{\text{split}}(z)$ to satisfyingly approximate $g(z)$ (see Figure 4.5.) We also show in the appendix that an arbitrarily close approximation can always be obtained with B sufficiently large.

4.4 Results

Our results are presented in Table 4.1 and Figure 4.3.

In Table 4.1, we present certificates that our algorithm generates on CIFAR-10 for $\ell_{1/2}$ and $\ell_{1/3}$ quasi-norms. As a baseline, we compare to DSSN (Chapter 3) certificates for ℓ_1, using norm inequalities to derive certificates for ℓ_p ($p < 1$). In particular we use the standard norm inequality:

$$\ell_p(\boldsymbol{x}, \boldsymbol{y}) \geq \ell_1(\boldsymbol{x}, \boldsymbol{y}), \ \ \forall p \in (0, 1) \tag{4.25}$$

$\ell_{1/2}$

ρ	10	20	30	40	50	60	70	80
DSSN (From ℓ_1)	42.69% (60.42% @ α=18)	35.04% (60.42% @ α=18)	28.89% (60.42% @ α=18)	23.46% (60.42% @ α=18)	18.81% (60.42% @ α=18)	13.76% (60.42% @ α=18)	8.38% (60.42% @ α=18)	1.27% (60.42% @ α=18)
DSSN (From ℓ_1) (Stab. Training)	41.32% (55.38% @ α=12)	35.56% (50.11% @ α=18)	32.07% (50.11% @ α=18)	28.70% (50.11% @ α=18)	24.95% (50.11% @ α=18)	20.79% (50.11% @ α=18)	16.20% (50.11% @ α=18)	6.98% (50.11% @ α=18)
Variable-Λ	**56.74%** (73.22% @ α=15)	**49.80%** (70.57% @ α=18)	43.60% (70.57% @ α=18)	37.97% (70.57% @ α=18)	32.37% (70.57% @ α=18)	25.83% (70.57% @ α=18)	18.19% (70.57% @ α=18)	5.02% (70.57% @ α=18)
Variable-Λ (Stab. Training)	55.21% (69.87% @ α=9)	48.72% (62.74% @ α=15)	**45.05%** (60.44% @ α=18)	**42.26%** (60.44% @ α=18)	**38.62%** (60.44% @ α=18)	**34.42%** (60.44% @ α=18)	**29.01%** (60.44% @ α=18)	**16.28%** (60.44% @ α=18)

$\ell_{1/3}$

ρ	90	180	270	360	450	540	630	720
DSSN (From ℓ_1)	34.98% (60.42% @ α=18)	27.86% (60.42% @ α=18)	22.69% (60.42% @ α=18)	18.49% (60.42% @ α=18)	14.32% (60.42% @ α=18)	10.37% (60.42% @ α=18)	5.99% (60.42% @ α=18)	0.89% (60.42% @ α=18)
DSSN (From ℓ_1) (Stab. Training)	35.54% (50.11% @ α=18)	31.30% (50.11% @ α=18)	28.06% (50.11% @ α=18)	24.75% (50.11% @ α=18)	21.33% (50.11% @ α=18)	18.27% (50.11% @ α=18)	13.97% (50.11% @ α=18)	6.07% (50.11% @ α=18)
Variable-Λ	**55.66%** (74.57% @ α=18)	49.04% (74.57% @ α=18)	43.27% (74.57% @ α=18)	38.21% (74.57% @ α=18)	33.37% (74.57% @ α=18)	27.17% (74.57% @ α=18)	20.27% (74.57% @ α=18)	6.87% (74.57% @ α=18)
Variable-Λ (Stab. Training)	54.63% (70.21% @ α=12)	**49.88%** (64.30% @ α=18)	**46.92%** (64.30% @ α=18)	**44.11%** (64.30% @ α=18)	**41.03%** (64.30% @ α=18)	**37.56%** (64.30% @ α=18)	**32.46%** (64.30% @ α=18)	**20.84%** (64.30% @ α=18)

Table 4.1: Certified accuracy as a function of fractional ℓ_p distance ρ, for $p = 1/2$ and $1/3$, on CIFAR-10. We train using standard smoothed training [32] as well as with stability training [35]. As a baseline, we compare to certificates computed from the ℓ_1 certificates given by DSSN. We test with $\alpha = \{1, 3, 6, 9, 12, 15, 18\}$ where $1/\alpha$ is the Lipschitz constant of the model (as mentioned in Section 4.2, for DSSN, $\Lambda = \alpha$), and report the highest certificate for each technique over all of the models. In parentheses, we report the the clean accuracy and the α parameter for the associated model. Complete results for all models are reported in Appendix C.8, as are base classifier accuracies for each model. For $p = 1/2$, we also provide results for larger values of α (up to $\alpha = 30$) in Appendix C.9.

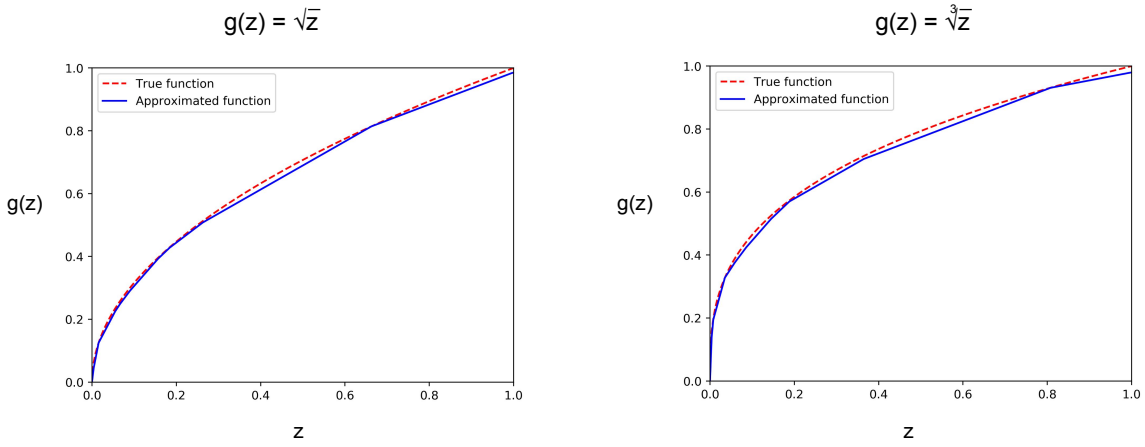


Figure 4.5: Approximations of g for $p = \frac{1}{2}$ and $p = \frac{1}{3}$ using a budget of $B = 1000$ smoothing samples. In both cases, the true and approximated functions differ by at most 0.02.

Moreover, since our domain is $[0, 1]$, we have:

$$\ell_p(\boldsymbol{x}, \boldsymbol{y}) = \left(\sum_{i=1} \delta_i^p\right)^{1/p} \geq \left(\sum_{i=1}^{d} \delta_i\right)^{1/p} = (\ell_1(\boldsymbol{x}, \boldsymbol{y}))^{1/p}, \qquad \forall p \in (0, 1) \tag{4.26}$$

This means that we can compute the baseline, ℓ_1-based certificate as:

$$\text{Cert.}(\ell_p) = \max(\text{Cert.}(\ell_1), \text{Cert.}(\ell_1)^{1/p}) \tag{4.27}$$

Table 4.1 shows that our method outperforms this baseline significantly on CIFAR-10 at a wide range of scales. For example, at an $\ell_{1/3}$ radius of 720, the proposed method has over 14 percentage points higher certified-robust accuracy, using a model that also has over 14 percentage points higher clean accuracy.

In Figure 4.6, we present certificate results of our method on ImageNet-1000, showing that our method scales to high-dimensional datasets, with one model able to certify many samples as robust to an $\ell_{1/2}$ radius close to 80 while maintaining over 50% clean accuracy.

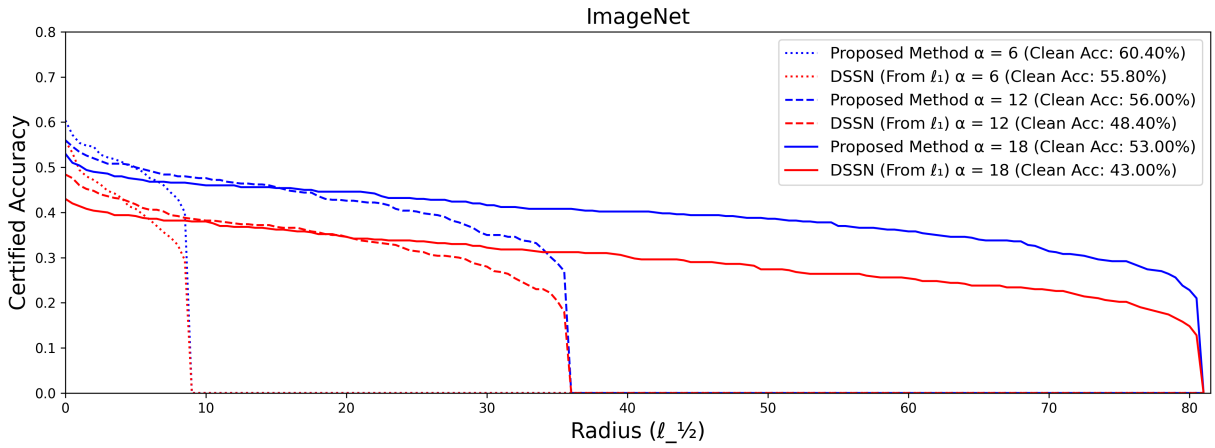


Figure 4.6: Certified Accuracy for variable-Λ smoothing as a function of $\ell_{1/2}$ norm on ImageNet-1000. We use a subset consisting of 500 samples from the validation set for evaluation. $1/\alpha$ is the $\ell_{1/2}^{1/2}$ Lipschitz constant of the classifier logits. As a baseline, we compare to certificates computed from the ℓ_1 certificates given by DSSN. Base classifier accuracies are reported in Appendix C.10.

Our architecture and training settings were largely the same as used in Chapter 3, using WideResNet-40 for CIFAR-10 and ResNet-50 for ImageNet. One additional challenge was the presence of both $\boldsymbol{x}^{\text{lower}}$ and $\boldsymbol{x}^{\text{upper}}$ as inputs to the base classifier $f(\cdot)$. DSSN does not need this, because when Λ is fixed, $\boldsymbol{x}^{\text{lower}}$ and $\boldsymbol{x}^{\text{upper}}$ can both be computed from their mean. In order to use the extra information, we doubled the input channels to the first convolutions layer, and represented the two images in different channels. We explore other representation techniques in Appendix C.5. An explicit description of our certification procedure is given in Appendix C.4.

We use 1000α smoothing samples to approximate the metric functions $g = \frac{z^p}{\alpha}$, where $1/\alpha$ is the Lipschitz constant. Note that, unlike in randomized smoothing, this "sampling" does not mean that our final certificates are non-deterministic or approximate: they are exact certificates for the fractional ℓ_p-quasi-norm.

4.5 Deterministic ℓ_0 Certificates

Consider the following ECM, parameterized by α:

$$g_i(z) := \begin{cases} 0 \text{ if } z = 0 \\ \frac{1}{\alpha} \text{ otherwise} \end{cases} \tag{4.28}$$

Note that the resulting metric $d(\boldsymbol{x}, \boldsymbol{y})$ is in fact $\|\boldsymbol{x}-\boldsymbol{y}\|_0/\alpha$. However, because this is not a continuous function, we cannot apply Theorem 4.2-c directly. However, if we still want $\Pr_i^{\text{split}}(z) = g_i(z)$, we have two options:

- Option 1: expand the support of $\mathcal{D}_i$ to include $\Lambda_i = 0$, where, if $\Lambda_i = 0$, then $x_i^{\text{lower}} = x_i^{\text{upper}} = x_i$. We can then distribute Λ_i as:

$$\Lambda_i = \begin{cases} 0 \text{ with prob. } \frac{1}{\alpha} \\ \infty \text{ otherwise} \end{cases} \tag{4.29}$$

  It is easy to verify that in this case, $\Pr_i^{\text{split}}(z) = g_i(z)$. (In particular, if $x_i = y_i$, then $\Pr((x_i^{\text{lower}}, x_i^{\text{upper}}) \neq (y_i^{\text{lower}}, y_i^{\text{upper}})) = 0$; otherwise $\Pr((x_i^{\text{lower}}, x_i^{\text{upper}}) \neq (y_i^{\text{lower}}, y_i^{\text{upper}})) = \Pr(\Lambda = 0) = 1/\alpha$. See Figure 4.7.)

- Option 2: consider the quantized case. Then we can just apply Theorem 4.3-c directly, yielding

$$\Lambda_i = \begin{cases} 1/q \text{ with prob. } \frac{1}{\alpha} \\ \infty \text{ otherwise} \end{cases} \tag{4.30}$$

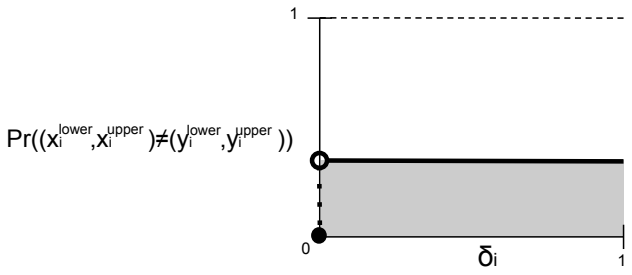


Figure 4.7: Diagram of the ℓ_0 $g_i(z)$ function.

Note that with $\Lambda = 1/q$, the original value of the pixel is always preserved, with $x_i^{\text{upper}} = x_i + 0.5/q$, $x_i^{\text{lower}} = x_i - 0.5/q$.

In practice, we use Option 2 in our experiments, because we are using quantized image datasets (and for code consistency). However, either option will yield classifiers that $1/\alpha$-Lipschitz with respect to the ℓ_0 metric, and in either case we can achieve efficient derandomization. If α is an integer (as in our experiments), then we only need $B = \alpha$ smoothing samples: each pixel is preserved ($\Lambda = 0$ or $\Lambda = 1/q$) in exactly one sample, and is ablated ($\Lambda = \infty$) in the other $\alpha - 1$ samples. The choice of which pixels to retain in which samples should be arbitrary, but should remain fixed throughout training and testing. (This is a direct application of the "fixed offset" method mentioned above).

In practice, this produces an algorithm which is very similar to the "randomized ablation" randomized ℓ_0 certificate proposed in Chapter 2: in both techniques, we are retaining some pixels unchanged while completely removing information about other pixels. In fact, this deterministic variant of "randomized ablation" was previously proposed to provide provable robustness against poisoning attacks in [74] (Chapter 7 of this dissertation, originally published before this chapter): in particular, the technique proposed for label-flipping poisoning attacks is basically identical, with the features being training-data labels rather than pixels: the idea is to train α separate

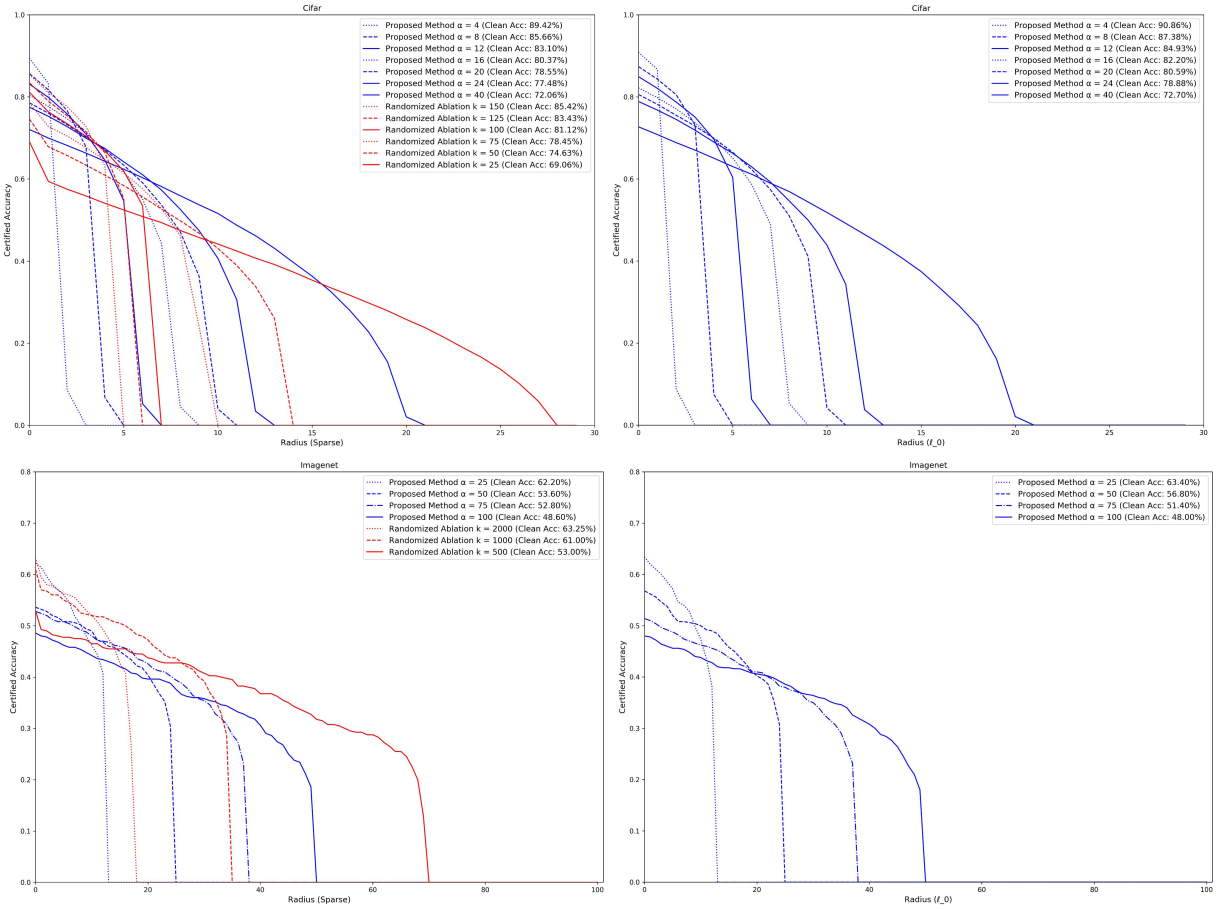


Figure 4.8: Sparse and ℓ_0 certification results, on CIFAR-10 (top) and ImageNet (bottom). In the left column, we compare to randomized ablation (Chapter 2), where certificates were reported with 95% confidence. Results are directly from that chapter: note that training times, model architectures, and parameters somewhat differ, in addition to the smoothing method.[4] On ImageNet, we use a subset of 500 images from the validation set; the results from Chapter 2 are using a different random subset of 400 validation images, so this may cause some variance. The parameter k is the number of pixels retained in each image in Randomized Ablation: because ImageNet has 50176 pixels, the fraction of retained pixels is roughly 50000/k, which functionally corresponds to α in our model: it is the appropriate to compare $k = 2000$ with $\alpha = 25$, etc. For CIFAR-10, there are 1024 pixels, so a similar heuristic of $\alpha \approx 1000/k$ can be used. In the right column, we show certificates for ℓ_0 attacks, where the attack budget represents the number of individual pixel channels, rather than whole pixels, attacked. We did not test for this threat model in Chapter 2.

models, each using a disjoint arbitrary subset of labels, and then take the consensus output at test time. Chapter 7 notes that the certificate is looser than that of Chapter 2, due to the use of a

[4]This is because the experimental setup in this chapter closely follows that in Chapter 3, which itself closely follows the setup of Yang et al. (2020) [51] to allow for direct comparisons: note that Yang et al. (2020) was published

union bound, however there are added benefits of determinism and using only a small number of smoothing samples (Chapter 2 uses 11,000 smoothing samples (1000 for prediction and 10,000 for bounding); in the case of Chapter 7, each "smoothing sample" requires training a classifier).

Note that, on image data, there are two somewhat different definitions of "ℓ_0 adversarial attack" which are often used: ℓ_0 attacks in the space of features, where each feature is a single color channel of a pixel value, and "sparse" attacks, where the attack magnitude signifies the number of pixel positions modified, but potentially all channels may be affected. Our method can be applied in both situations: to certify for "sparse" attacks, simply insure that $\Lambda_i = \Lambda_j$ if features i, j are channels of the same pixel: then $\Pr((x_i^{\text{lower}}, x_i^{\text{upper}}) \neq (y_i^{\text{lower}}, y_i^{\text{upper}}) \cup (x_j^{\text{lower}}, x_j^{\text{upper}}) \neq (y_j^{\text{lower}}, y_j^{\text{upper}})) \leq 1/\alpha$.

In Figure 4.8, we compare the certificates generated by this deterministic "sparse" certificate to the results of Chapter 2. While the reported certificates are somewhat worse, particularly on ImageNet, note that these are exact, rather than probabilistic certificates, and furthermore that the number of forward-passes required to certify is significantly reduced, leading to reduced certification times. For example, on ImageNet, the most computationally-intensive certification for the deterministic method used 100 forward-passes, and averaged 0.13 seconds / image for certification using a single GPU. By contrast, each randomized certification from Chapter 2 averaged 16 seconds, using four GPUs (note that this is around four times less efficient than expected, compared to the proposed derandomized method, based on the number of smoothing samples alone: the other implementation differences seem to be also be at play, as well as differences in the time required for noise sampling). We also provide certificates for feature-level ℓ_0 attacks, which we did not test in Chapter 2.

after Chapter 2 [58] was originally published.

Part II

Scalable Robustness Certificates beyond ℓ_p Threat Models

Chapter 5: Wasserstein Smoothing: Certified Robustness against Wasserstein Adversarial Attacks[1]

Recently, non-additive threat models [2, 87, 88, 89] have been introduced which aim to minimize the distance between $\boldsymbol{x}$ and $\tilde{\boldsymbol{x}}$ according to metrics other than ℓ_p threat models. Among these attacks is the attack introduced by [2] which considers the Wasserstein distance between $\boldsymbol{x}$ and $\tilde{\boldsymbol{x}}$, normalized such that the pixel intensities of the image can be treated as probability distributions. Informally, the Wasserstein distance between probability distributions $\boldsymbol{x}$ and $\tilde{\boldsymbol{x}}$ measures the minimum cost to 'transport' probability mass in order to transform $\boldsymbol{x}$ into $\tilde{\boldsymbol{x}}$, where the cost scales with both the amount of mass transported and the distance over which it is transported with respect to some underlying metric. The intuition behind this threat model is that shifting pixel intensity a short distance across an image is less perceptible than moving the same amount of pixel intensity a larger distance (See Figure 5.1 for an example of a Wasserstein adversarial attack.)

A variety of practical approaches have been proposed to make classifiers robust against adversarial attack, including adversarial training [9], defensive distillation [13], and obfuscated gradients [90]. However, as new defenses are proposed, new attack methods are often developed which defeat them [12, 18, 19]. While updated defenses are often then proposed [12], in general,

[1] A form of chapter has been published in AISTATS 2020 [86].

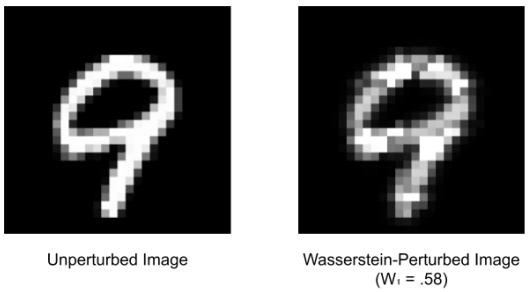


Figure 5.1: An illustration of Wasserstein adversarial attack [2].

we cannot be confident that newer attacks will not in turn defeat these defenses.

To escape this cycle, approaches have been proposed to develop certifiably robust classifiers [22, 23, 33, 34, 63, 91]: in these classifiers, for each image $\boldsymbol{x}$, one can calculate a radius ρ such that it is provably guaranteed that any other image $\tilde{\boldsymbol{x}}$ with distance less than ρ from $\boldsymbol{x}$ will be classified similarly to $\boldsymbol{x}$. This means that no adversarial attack can ever be developed which produces adversarial examples to the classifier within the certified radius.

One effective approach to develop certifiably robust classification is to use randomized smoothing with a probabilistic robustness certificate [33, 34, 63, 91]. In this approach, starting with a *base classifier* $f(\boldsymbol{x})$ one uses a smoothed classifier $p(\boldsymbol{x})$, which represents the expectation of $f(\boldsymbol{x})$ over random perturbations of $\boldsymbol{x}$. Based on this smoothing, one can derive an upper bound on how steeply the scores assigned to each class by p can change, which can then be used to derive a radius ρ in which the highest class score must remain highest[2].

In this work, we present the first certified defense against Wasserstein adversarial attacks using an adapted randomized smoothing approach, which we call *Wasserstein smoothing*. To develop the robustness certificate, we define a (non-unique) representation of the difference between two images, based on the flow of pixel intensity necessary to construct one image from

[2]In practice, samples are used to estimate the expectation $p(\boldsymbol{x})$, producing an empirical smoothed classifier $\tilde{p}(\boldsymbol{x})$: the certification is therefore probabilistic, with a degree of certainty dependent on the number of samples.

another. In this representation, we show that the ℓ_1 norm of the minimal flow between two images is equal to the Wasserstein distance between the images. This allows us to apply existing ℓ_1 smoothing-based defenses [3], by adding noise in the space of these representations of flows. We show empirically that this gives improved robustness certificates, compared to using a weak upper bound on Wasserstein distance given by randomized smoothing in the feature space of images directly. We also show that our Wasserstein smoothing defense protects against Wasserstein adversarial attacks in practice, with significantly improved empirical robustness compared to baseline models. For small adversarial perturbations on the MNIST dataset, our method achieves higher accuracy under adversarial attack than all existing practical defenses for the Wasserstein threat model. In summary, we make the following contributions:

- We develop a novel certified defense for the Wasserstein adversarial attack threat model. This is the first certified defense, to our knowledge, that has been proposed for this threat model.
- We demonstrate that our certificate is nonvacuous, in that it can certify Wasserstein radii larger than those which can be certified by exploiting a trivial ℓ_1 upper bound on Wasserstein distance.
- We demonstrate that our defense effectively protects against existing Wasserstein adversarial attacks, compared to an unprotected baseline.

After the publication of this work, [92] used the mathematical results developed here to design a bound-propagation based Wasserstein certification method which achieves superior results on

[3] This chapter was originally published before [51] demonstrated that uniform random noise was superior for ℓ_1 certification, and therefore uses the Laplace noise method from [34]. However, the techniques demonstrated can easily be adapted to use uniform noise. In Appendix D.4, we show why the deterministic smoothing method discussed in Chapter 3 cannot be straightforwardly adapted to the Wasserstein smoothing setting.

MNIST.

5.1 Background

Let $\boldsymbol{x} \in [0,1]^{n \times m}$ denote a two dimensional image, of height n and width m. We will normalize the image such that $\sum_i \sum_j x_{i,j} = 1$, so that $\boldsymbol{x}$ can be interpreted as a probability distribution on the discrete support of pixel coordinates of the 2D image.[4] Also, let $[n]$ denote the set of integers 1 through n, and let $< A, B >$ denote the elementwise inner product between A and B. Following the notation of [2], we define the p-Wasserstein distance between $\boldsymbol{x}$ and $\boldsymbol{x}'$ as:

Definition 5.1. *Given two distributions $\boldsymbol{x}, \boldsymbol{x}' \in [0,1]^{n \times m}$, and a distance metric $d \in ([n] \times [m]) \times ([n] \times [m]) \to \mathbb{R}$, the p-Wasserstein distance is:*

$$W_p(\boldsymbol{x}, \boldsymbol{x}') = \min_{\Pi \in \mathbb{R}_+^{(n \cdot m) \times (n \cdot m)}} < \Pi, C >, \tag{5.1}$$

$$\Pi \mathbf{1} = \boldsymbol{x}, \ \Pi^T \mathbf{1} = \boldsymbol{x}',$$

$$C_{(i,j),(i',j')} := \left[d\left((i,j),(i',j')\right) \right]^p.$$

Note that $C_{(i,j),(i',j')}$ is the cost of transporting a mass unit from the position (i,j) to (i',j') in the image. For the purpose of matrix multiplication, we are treating $\boldsymbol{x}, \boldsymbol{x}'$ as vectors of length nm. Similarly, the transport plan matrix Π and the cost matrix C are in $\mathbb{R}^{nm \times nm}$.

Intuitively, $\Pi_{(i,j),(i',j')}$ represents the amount of probability mass to be transported from pixel (i,j) to (i',j'), while $C_{(i,j),(i',j')}$ represents the cost per unit probability mass to transport

[4] In the case of multi-channel color images, the attack proposed by [2] does not transport pixel intensity between channels. This allows us to defend against these attacks using our 2D Wasserstein smoothing with little modification. See Section 5.5.3, and Corollary D.2 in the appendix.

this probability. We can choose $d(.,.)$ to be any measure of distance between pixel positions in an image. For example, in order to represent the ℓ_1 distance metric between pixel positions, we can choose:

$$d\left((i,j),(i',j')\right) = |i-i'| + |j-j'|. \tag{5.2}$$

Moreover, to represent the ℓ_2 distance metric between pixel positions, we can choose:

$$d\left((i,j),(i',j')\right) = \sqrt{(i-i')^2 + (j-j')^2}. \tag{5.3}$$

Our defense directly applies to the 1-Wasserstein metric using the ℓ_1 distance as the metric $d(.,.)$, while the attack developed by [2] uses the ℓ_2 distance. However, because images are two dimensional, these differ by at most a constant factor of $\sqrt{2}$, so we adapt our certificates to the setting of [2] by simply scaling our certificates by $1/\sqrt{2}$. All experimental results will be presented with this scaling. We emphasize that this it *not* the distinction between 1-Wasserstein and 2-Wasserstein distances: this paper uses the 1-Wasserstein metric, to match the majority of the experimental results of [2].

To develop our certificate, we rely an alternative linear program formulation for the 1-Wasserstein distance on a two-dimensional image with the ℓ_1 distance metric, provided by [93]:

$$W_1(\boldsymbol{x}, \boldsymbol{x}') = \min_{\mathbf{g}} \sum_{(i,j)} \sum_{(i',j') \in \mathcal{N}(i,j)} \mathbf{g}_{(i,j),(i',j')} \tag{5.4}$$

where $\mathbf{g} \geq 0$ and $\forall(i,j)$,

$$\sum_{(i',j') \in \mathcal{N}(i,j)} \mathbf{g}_{(i,j),(i',j')} - \mathbf{g}_{(i',j'),(i,j)} = \boldsymbol{x}'_{i,j} - \boldsymbol{x}_{i,j}$$

Here, $\mathcal{N}(i,j)$ denotes the (up to) four immediate (non-diagonal) neighbors of the position (i,j); in other words, $\mathcal{N}(i,j) = \{(i',j') \mid |i-i'| + |j-j'| = 1\}$. For the ℓ_1 distance in two dimensions, [93] prove that this formulation is in fact equivalent to the linear program given in Equation 5.1. Note that only elements of $\mathbf{g}$ with $|i-i'|+|j-j'| = 1$ need to be defined: this means that the number of variables in the linear program is approximately $4nm$, compared to the n^2m^2 elements of Π in Equation 5.1. While this was originally used to make the linear program more tractable to be solved directly, we exploit the form of this linear program to devise a randomized smoothing scheme in the next section.

5.2 Robustness Certificate

In order to present our robustness certificate, we first introduce some notation. Let $\boldsymbol{\delta} = \{\boldsymbol{\delta}^{\text{vert.}} \in \mathbb{R}^{(n-1)\times m}, \boldsymbol{\delta}^{\text{horiz.}} \in \mathbb{R}^{n\times(m-1)}\}$ denote a *local flow plan*. It specifies a net flow between adjacent pixels in an image $\boldsymbol{x}$, which, when applied, transforms $\boldsymbol{x}$ to a new image $\boldsymbol{x}'$. See Figure 2 for an explanation of the indexing. For compactness, we write $\boldsymbol{\delta} \in \mathbb{R}^r$ where $r = (n-1)m + n(m-1) \approx 2nm$, and in general refer to the space of possible local flow plans as the *flow domain*. We define the function Δ, which applies a local flow to a distribution.

Definition 5.2. *The local flow plan application function* $\Delta \in \mathbb{R}^{n\times m} \times \mathbb{R}^r \to \mathbb{R}^{n\times m}$ *is defined as:*

$$\Delta(\boldsymbol{x}, \boldsymbol{\delta})_{i,j} = \boldsymbol{x}_{i,j} + \boldsymbol{\delta}^{vert.}_{i-1,j} - \boldsymbol{\delta}^{vert.}_{i,j} + \boldsymbol{\delta}^{horiz.}_{i,j-1} - \boldsymbol{\delta}^{horiz.}_{i,j} \tag{5.5}$$

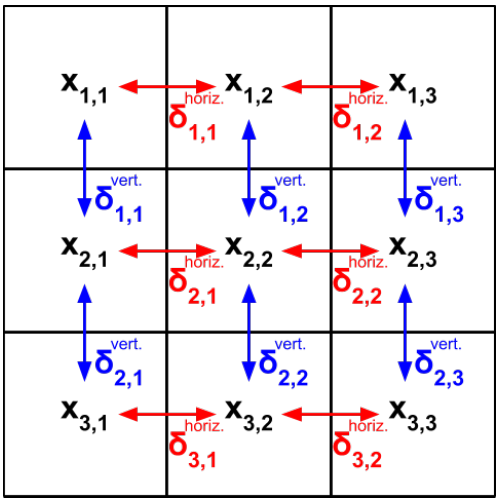


Figure 5.2: Indexing of the elements of the local flow map $\boldsymbol{\delta}$, in relation to the pixels of the image $\boldsymbol{x}$, with $n = m = 3$.

where we let $\boldsymbol{\delta}^{vert.}_{0,j} = \boldsymbol{\delta}^{vert.}_{n,j} = \boldsymbol{\delta}^{horiz.}_{i,0} = \boldsymbol{\delta}^{horiz.}_{i,m} = 0$.[5]

Note that local flow plans are additive:

$$\Delta(\Delta(\boldsymbol{x}, \boldsymbol{\delta}), \boldsymbol{\delta}') = \Delta(\boldsymbol{x}, \boldsymbol{\delta} + \boldsymbol{\delta}') \tag{5.6}$$

Using this notation, we make a simple transformation of the linear program given in Equation 5.4, removing the positivity constraint from the variables and reducing the number of variables to $\sim 2nm$:

Lemma 5.1. *For any normalized probability distributions* $\boldsymbol{x}, \boldsymbol{x}' \in [0,1]^{n \times m}$*:*

$$W_1(\boldsymbol{x}, \boldsymbol{x}') = \min_{\boldsymbol{\delta}:\, \boldsymbol{x}' = \Delta(\boldsymbol{x}, \boldsymbol{\delta})} \|\boldsymbol{\delta}\|_1 \tag{5.7}$$

where W_1 *denotes the 1-Wasserstein metric, using the* ℓ_1 *distance as the underlying distance*

[5]Note that the new image $\boldsymbol{x}' = \Delta(\boldsymbol{x}, \boldsymbol{\delta})$ is not necessarily a probability distribution because it may have negative components. However, note that normalization is preserved: $\sum_i \sum_j x'_{i,j} = 1$. This is because every component of $\boldsymbol{\delta}$ is added once and subtracted once to elements in $\boldsymbol{x}$.

metric d.

Therefore, we can upper-bound the Wasserstein distance between two images using the ℓ_1 norm of *any feasible* local flow plan between them. This enables us to extend existing results for ℓ_1 smoothing-based certificates [34] to the Wasserstein metric, by adding noise in the flow domain.

Definition 5.3. *We denote by* $\mathcal{L}(\sigma) = \text{Laplace}(0, \sigma)^r$ *as the Laplace noise with parameter* σ *in the flow domain of dimension* r.

Given a classification score function $f : \mathbb{R}^{n \times m} \to [0, 1]^k$, we define p as the *Wasserstein-smoothed* classification function as follows:

$$p(\boldsymbol{x}) = \mathop{\mathbb{E}}_{\boldsymbol{\delta} \sim \mathcal{L}(\sigma)} \left[f(\Delta(\boldsymbol{x}, \boldsymbol{\delta})) \right]. \tag{5.8}$$

Let i be the class assignment of $\boldsymbol{x}$ using the Wasserstein-smoothed classifier p (in other words, $i = \arg\max_{i'} p_{i'}(\boldsymbol{x})$).

Theorem 5.1. *For any normalized probability distribution* $\boldsymbol{x} \in [0, 1]^{n \times m}$*, if*

$$p_i(\boldsymbol{x}) \geq e^{2\sqrt{2}\rho/\sigma} \max_{i' \neq i} p_{i'}(\boldsymbol{x}) \tag{5.9}$$

then for any perturbed probability distribution $\tilde{\boldsymbol{x}}$ *such that* $W_1(\boldsymbol{x}, \tilde{\boldsymbol{x}}) \leq \rho$*, we have:*

$$p_i(\tilde{\boldsymbol{x}}) \geq \max_{i' \neq i} p_{i'}(\tilde{\boldsymbol{x}}). \tag{5.10}$$

All proofs are presented in the appendix.

5.3 Intuition: One-Dimensional Case

To provide an intuition about the proposed Wasserstein smoothing certified robustness scheme, we consider a simplified model, in which the support of $\boldsymbol{x}$ is a one-dimensional array of length n, rather than a two-dimensional grid (i.e. $\boldsymbol{x} \in \mathbb{R}^n$). In this case, we can denote a local flow plan $\boldsymbol{\delta} \in \mathbb{R}^{n-1}$, so that for $\boldsymbol{x}' = \Delta(\boldsymbol{x}, \boldsymbol{\delta})$:

$$\boldsymbol{x}'_i = \boldsymbol{x}_i + \boldsymbol{\delta}_{i-1} - \boldsymbol{\delta}_i \tag{5.11}$$

where $\boldsymbol{\delta}_0 = \boldsymbol{\delta}_n = 0$. In this one-dimensional case, for any fixed $\boldsymbol{x}, \boldsymbol{x}'$ (with the normalization constraint that $\sum_i x_i = \sum_i x'_i = 1$), there is a unique solution $\boldsymbol{\delta}$ to $\boldsymbol{x}' = \Delta(\boldsymbol{x}, \boldsymbol{\delta})$:

$$\boldsymbol{\delta}_i = \sum_{j=1}^{i} x_j - \sum_{j=1}^{i} x'_j \tag{5.12}$$

Note at this reminds us a well-known identity describing optimal transport between two distributions X, Y which share a continuous, one-dimensional support (see Section 2.6 of [94], for example):

$$W_1(X, Y) = \int_{-\infty}^{\infty} |F_X(z) - F_Y(z)| dz \tag{5.13}$$

where F_X, F_Y denote cumulative density functions. If we apply this result to our discretized case, with the index i taking the place of z, and apply the identity to $\boldsymbol{x}$ and $\boldsymbol{x}'$, this becomes:

$$W_1(\boldsymbol{x}, \boldsymbol{x}') = \sum_{i=1}^{n} \left| \sum_{j=1}^{i} x_j - \sum_{j=1}^{i} x'_j \right| = \sum_{i=1}^{n} |\boldsymbol{\delta}_i| = \|\boldsymbol{\delta}\|_1 \tag{5.14}$$

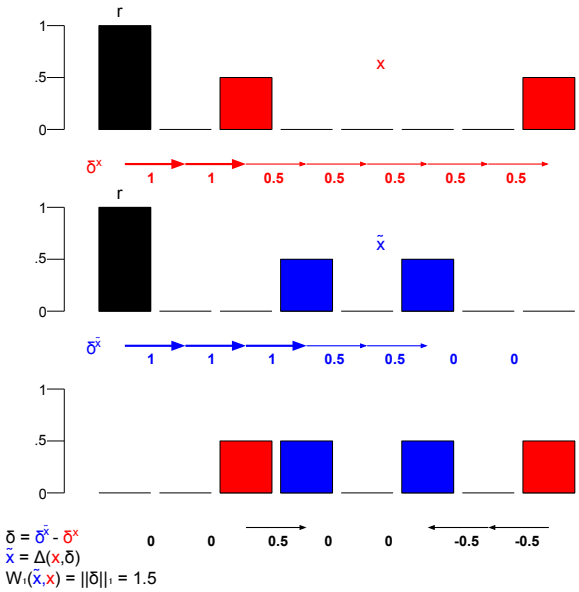


Figure 5.3: An illustrative example in one dimension. $\boldsymbol{r}$ (black) denotes a fixed reference distribution. With this starting distribution fixed, $\boldsymbol{x}$ (red) and $\tilde{\boldsymbol{x}}$ (blue) can both be uniquely represented in the flow domain as $\boldsymbol{\delta}^{\boldsymbol{x}}$ and $\boldsymbol{\delta}^{\tilde{\boldsymbol{x}}}$. Note that the Wasserstein distance between $\boldsymbol{x}$ and $\tilde{\boldsymbol{x}}$ is then equivalent to the ℓ_1 distance between $\boldsymbol{\delta}^{\boldsymbol{x}}$ and $\boldsymbol{\delta}^{\tilde{\boldsymbol{x}}}$. In the one-dimensional case, this shows that we can transform the samples into a space where the Wasserstein threat model is equivalent to the ℓ_1 metric. We can then use a pre-existing ℓ_1 certified defense in the flow space to defend our classifier.

By the uniqueness of the solution given in Equation 5.12, for any $\boldsymbol{x}$, we can define $\boldsymbol{\delta}^{\boldsymbol{x}}$ as the solution to $\boldsymbol{x} = \Delta(\boldsymbol{r}, \boldsymbol{\delta})$, where $\boldsymbol{r}$ is an arbitrary fixed reference distribution (e.g. suppose $r_1 = 1, r_i = 0$ for $i \neq 1$). Therefore, instead of operating on the images $\boldsymbol{x}, \tilde{\boldsymbol{x}} \in \mathbb{R}^n$ directly, we can equivalently operate on $\boldsymbol{\delta}^{\boldsymbol{x}}$ and $\boldsymbol{\delta}^{\tilde{\boldsymbol{x}}}$ in the flow domain instead. We will therefore define a flow-domain version of our classifier f:

$$f^{\text{flow}}(\boldsymbol{\delta}) := f(\Delta(\boldsymbol{r}, \boldsymbol{\delta})). \tag{5.15}$$

We will now perform classification entirely in the flow-domain, by first calculating $\boldsymbol{\delta}^{\boldsymbol{x}}$ and then using $f^{\text{flow}}(\boldsymbol{\delta}^{\boldsymbol{x}})$ as our classifier. Now, consider $\boldsymbol{x}$ and an adversarial perturbation $\tilde{\boldsymbol{x}}$, and let $\boldsymbol{\delta}$ be

the unique solution to $\tilde{\boldsymbol{x}} = \Delta(\boldsymbol{x}, \boldsymbol{\delta})$. By Equation 5.14, $\|\boldsymbol{\delta}\|_1 = W_1(\boldsymbol{x}, \tilde{\boldsymbol{x}})$. Then:

$$\tilde{\boldsymbol{x}} = \Delta(\boldsymbol{x}, \boldsymbol{\delta}) = \Delta(\Delta(\boldsymbol{r}, \boldsymbol{\delta}^{\boldsymbol{x}}), \boldsymbol{\delta}) = \Delta(\boldsymbol{r}, \boldsymbol{\delta}^{\boldsymbol{x}} + \boldsymbol{\delta}) \tag{5.16}$$

where the second equality is by Equation 5.6. Moreover, by the uniqueness of Equation 5.12, $\boldsymbol{\delta}^{\tilde{\boldsymbol{x}}} = \boldsymbol{\delta}^{\boldsymbol{x}} + \boldsymbol{\delta}$, or $\boldsymbol{\delta}^{\tilde{\boldsymbol{x}}} - \boldsymbol{\delta}^{\boldsymbol{x}} = \boldsymbol{\delta}$. Therefore

$$\|\boldsymbol{\delta}^{\tilde{\boldsymbol{x}}} - \boldsymbol{\delta}^{\boldsymbol{x}}\|_1 = W_1(\boldsymbol{x}, \tilde{\boldsymbol{x}}). \tag{5.17}$$

In other words, if we classify in the flow-domain, using f^{flow}, the ℓ_1 distance between point $\boldsymbol{\delta}^{\boldsymbol{x}}, \boldsymbol{\delta}^{\tilde{\boldsymbol{x}}}$ is the Wasserstein distance between the distributions $\boldsymbol{x}$ and $\tilde{\boldsymbol{x}}$. Then, we can perform smoothing in the flow-domain, and use the existing ℓ_1 robustness certificate provided by [34], to certify robustness. Extending this argument to two-dimensional images adds some complication: images can no longer be represented uniquely in the flow domain, and the relationship between ℓ_1 distance and the Wasserstein distance is now an upper bound. Nevertheless, the same conclusion still holds for 2D images as we state in Theorem 5.1. Proofs for the two-dimensional case are given in the appendix.

5.4 Practical Certification Scheme

To generate probabilistic robustness certificates from randomly sampled evaluations of the base classifier f, we adapt the procedure outlined by [63] for ℓ_2 certificates. We consider a *hard smoothed classifier* approach: we set $f_j(\boldsymbol{x}) = 1$ if the base classifier selects class j at point $\boldsymbol{x}$,

Noise standard deviation σ	**Wasserstein Smoothing** Classification accuracy (Percent abstained)	**Wasserstein Smoothing** Median certified robustness	**Wasserstein Smoothing** Base Classifier Accuracy
0.005	98.71(00.04)	0.0101	97.94
0.01	97.98(00.19)	0.0132	94.95
0.02	93.99(00.58)	0.0095	79.72
0.05	74.22(03.95)	0	43.67
0.1	49.41(01.29)	0	30.26
0.2	31.80(08.40)	N/A	25.13
0.5	22.58(00.84)	N/A	22.67
Noise standard deviation σ	**Laplace Smoothing** Classification accuracy (Percent abstained)	**Laplace Smoothing** Median certified robustness	**Laplace Smoothing** Base Classifier Accuracy
0.005	98.87(00.06)	0.0062	97.47
0.01	97.44(00.19)	0.0053	89.32
0.02	91.11(01.29)	0.0030	67.08
0.05	61.44(07.45)	0	33.80
0.1	34.92(09.36)	N/A	25.56
0.2	24.02(05.67)	N/A	22.85
0.5	22.57(01.05)	N/A	22.70

Table 5.1: Certified Wasserstein Accuracy of Wasserstein and Laplace smoothing on MNIST

and $f_j(\boldsymbol{x}) = 0$ otherwise. We also use a stricter form of the condition given as Equation 5.9:

$$p_i(\boldsymbol{x}) \geq e^{2\sqrt{2}\rho/\sigma}(1 - p_i(\boldsymbol{x})) \tag{5.18}$$

This means that we only need to provide a probabilistic lower bound of the expectation of the largest class score, rather than bounding every class score. This reduces the number of samples necessary to estimate a high-confidence lower bound on $p_i(\boldsymbol{x})$, and therefore to estimate the certificate with high confidence. [63] provides a statistically sound procedure for this, which we use: refer to that paper for details. Note that, when simply evaluating the classification given by $p(\boldsymbol{x})$, we will also need to approximate p using random samples. [63] also provides a method to do this which yields the expected classification with high confidence, but may abstain from

classifying. We will also use this method when evaluating accuracies.

Since the Wasserstein adversarial attack introduced by [2] uses the ℓ_2 distance metric, to have a fair performance evaluation against this attack, we are interested in certifying a radius in the 1-Wasserstein distance with underlying ℓ_2 distance metric, rather than ℓ_1. Let us denote this radius as ρ_2. In two-dimensional images, the elements of the cost matrix C in this metric may be smaller by up to a factor of $\sqrt{2}$, so we have:

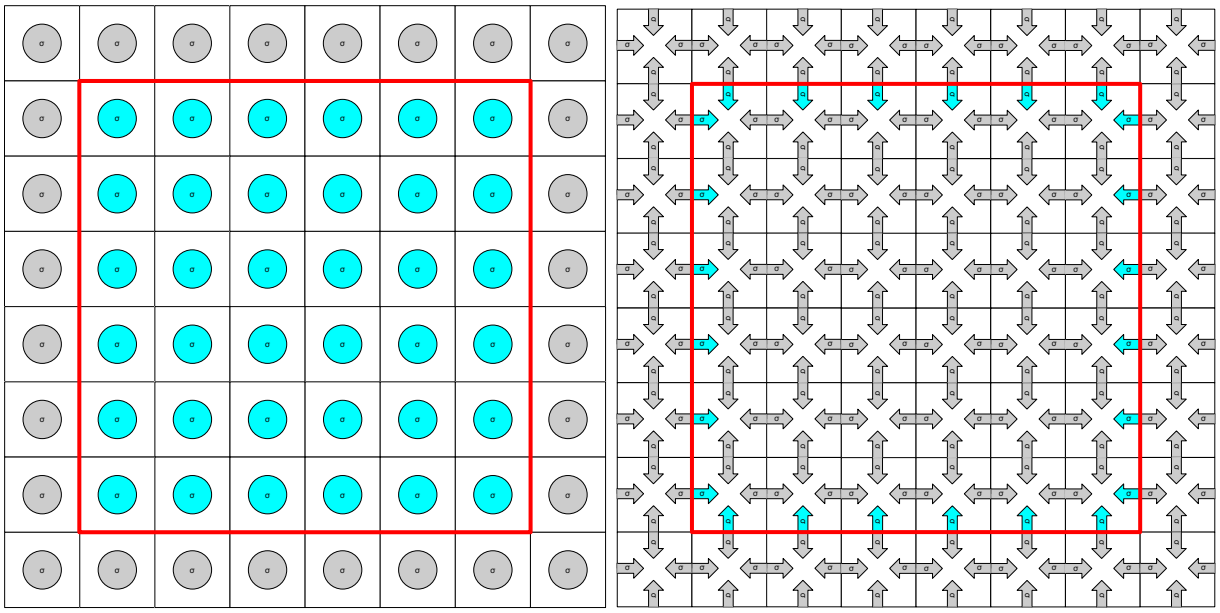
$$\rho_2 \geq \frac{1}{\sqrt{2}}\rho \tag{5.19}$$

Therefore, by certifying to a radius of $\rho = \sqrt{2}\rho_2$, we can effectively certify against the ℓ_2 metric 1-Wasserstein attacks of radius ρ_2. (We provide a more formal proof of this claim as Corollary D.3 in the appendix.) Our condition then becomes:

$$p_i(\boldsymbol{x}) \geq e^{4\rho_2/\sigma}(1 - p_i(\boldsymbol{x})). \tag{5.20}$$

5.5 Experimental Results

In all experiments, we use 10,000 random noised samples to predict the smoothed classification of each image; to generate certificates, we first use 1000 samples to infer which class has highest smoothed score, and then 10,000 samples to lower-bound this score. All probabilistic certificates and classifications are reported to 95% confidence. The model architectures used for the base classifiers for each data set are the same as used in [2]. When reporting results, *median*



(a) Laplace Smoothing (b) Wasserstein Smoothing

Figure 5.4: Schematic diagram showing the difference between Laplace and Wasserstein smoothing on the variance of the aggregate pixel intensity in a square region, outlined in red. See the text of Section 5.5.1. In both figures, pixels are represented as square tiles. In (a), noise on individual pixels is represented with circles, which are gray if they do *not* contribute to the overall pixel intensity in the outlined region, but are cyan if they *do* contribute. We see that the noise is proportional (in variance) to the area of the region. In (b), under Wasserstein smoothing, noise is represented by arrows between pixels which exchange intensity. Again, these are gray if they do not contribute to the overall pixel intensity in the outlined region, and cyan if they do contribute. Note that arrows in the interior do not contribute to the aggregate intensity, because equal values are added and subtracted from adjacent pixels. The noise is proportional (in variance) to the perimeter of the region. This provides a plausible intuition as to why base classifiers, when given noisy images, classify with higher accuracy on Wasserstein smoothed images compared to Laplace smoothed images, as seen empirically in Table 5.1.

certified accuracy refers to the maximum radius ρ_2 such that at least 50% of classifications for images in the data set are certified to be robust to at least this radius, and these certificates are for the correct ground truth class. If over 50% of images are not certified for the correct class, this statistic is reported as N/A.

5.5.1 Comparison to naive Laplace Smoothing

Note that one can derive a trivial but sometimes tight bound, that, under any ℓ_p distance metric, if $W_1(\boldsymbol{x}, \tilde{\boldsymbol{x}}) \leq \rho/2$, then $\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|_1 \leq \rho$. (See Corollary D.1 in the appendix.) This enables us to write a condition for ρ_2-radius Wasserstein certified robustness by applying Laplace smoothing directly, and simply converting the certificate. In our notation, this condition is:

$$p_i^{\text{Laplace}}(\boldsymbol{x}) \geq e^{4\sqrt{2}\rho_2/\sigma}(1 - p_i^{\text{Laplace}}(\boldsymbol{x})) \tag{5.21}$$

where $p^{\text{Laplace}}(\boldsymbol{x})$ is a smoothed classifier with Laplace noise added to every pixel independently. It may appear as if our Wasserstein-smoothed bound should only be an improvement over this bound by a factor of $\sqrt{2}$ in the certified radius ρ_2. However, as shown in Table 5.1, we in fact improve our certificates by a larger factor. This is because, for a fixed noise standard deviation, the base classifier is able to achieve a higher accuracy after adding noise in the flow-domain, compared to adding noise directly to the pixels. When adding noise in the flow-domain, we add and subtract noise in equal amounts between adjacent pixels, preserving more information for the base classifier.

To give a concrete example, consider some $k \times k$ square patch of an image. Suppose that the overall aggregate pixel intensity in this patch (i.e. the sum of the pixel values) is a salient feature for classification (This is a highly plausible situation: for example, in MNIST, this may indicate whether or not some region of an image is occupied by part of a digit.) Let us call this feature μ, and calculate the variance of μ in smoothing samples under Laplace and Wasserstein smoothing, both with variance σ^2. Under Laplace smoothing (Figure 5.4-a), k^2 independent instances of

Laplace noise are added to μ, so the resulting variance will be $k^2\sigma^2$: this is proportional to the area of the region. In the case of Wasserstein smoothing, by contrast, probability mass exchanged between between pixels in the interior of the patch has no effect on the aggregate quantity μ. Instead, only noise on the perimeter will affect the total feature value μ: the variance is therefore $4k\sigma^2$ (Figure 5.4-b). Wasserstein smoothing then reduces the effective noise variance on the feature μ by a factor of $O(k)$.

5.5.2 Empirical adversarial accuracy

We measure the performance of our smoothed classifier against the Wasserstein-metric adversarial attack proposed in [2], and compare to models tested in that work. Results are presented in Figure 5.5. For testing, we use the same attack parameters as in [2]: the "Standard" and "Adversarial Training" results are therefore replications of the experiments from that paper, using the publicly available code and pretrained models.

In order to attack our hard smoothed classifier, we adapt the method proposed by [33]: in particular, note that we cannot directly calculate the gradient of the classification loss with respect to the image for a *hard* smoothed classifier, because the derivatives of the logits of the base classifier are not propagated. Therefore, we must instead attack a *soft* smooth classifier: we take the expectation over samples of the *softmaxed* logits of the base classifier, instead of the final classification output. In each step of the attack, we use 128 noised samples to estimate this gradient, as used in [33].

In the attack proposed by [2], the images are attacked over 200 iterations of projected gradient descent, projected onto a Wasserstein ball, with the radius of the ball every 10 iterations.

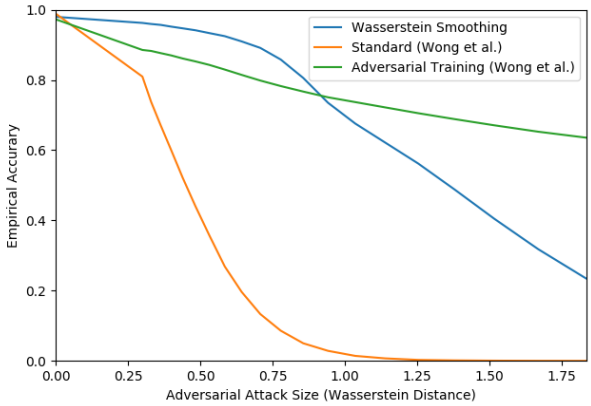


Figure 5.5: Comparison of empirical robustness on MNIST to models from [2]. Wasserstein smoothing is with $\sigma = 0.01$. (This is the amount of noise which maximizes certified robustness, as seen in Table 5.1.)

The attack succeeds, and the final radius is recorded, once the classifier misclassifies the image. In order to preserve as much of the structure (and code) of the attack as possible to provide a fair comparison, it is thus necessary for us to evaluate each image using our hard classifier, with the full 10,000 smoothing samples, at each iteration of the attack. We count the classifier abstaining as a misclassification for these experiments. However, note that this may somewhat underestimate the true robustness of our classifier: recall that our classifier is nondeterministic; therefore, because we are repeatedly evaluating the classifier and reporting a perturbed image as adversarial the first time it is missclassified, we may tend to over-count misclassifications. However, because we are using a large number of noise samples to generate our classifications, this is only likely to happen with examples which are close to being adversarial. Still, the presented data should be regarded as a lower bound on the true accuracy under attack of our Wasserstein smoothed classifier.

In Figure 5.5, we note two things: first, our Wasserstein smoothing technique appears to be an effective empirical defense against Wasserstein adversarial attacks, compared to an unprotected ('Standard') network. (It is also more robust than the binarized and ℓ_∞-robust models

Noise standard deviation σ	Classification accuracy (Percent abstained)	Median certified robustness	Base Classifier Accuracy
0.00005	87.01(00.24)	0.000101	86.02
0.0001	83.39(00.42)	0.000179	82.08
0.0002	77.57(00.66)	0.000223	75.46
0.0005	68.75(01.01)	0.000209	65.12
0.001	61.65(01.77)	0.000127	57.03

Table 5.2: Certified Wasserstein Accuracy of Wasserstein smoothing on CIFAR10

tested by [2]: see appendix.) However, for large perturbations, our defense is less effective than the adversarial training defense proposed by [2]. This suggests a promising direction for future work: [33] proposed an adversarial training method for smoothed classifiers, which could be applied in this case. Note however that both Wasserstein adversarial attacks and smoothed adversarial training are computationally expensive, so this may require significant computational resources.

Second, the median radius of attack to which our smoothed classifier is empirically robust is larger than the median certified robustness of our smoothed classifier by two orders of magnitude. This calls for future work both to develop improved robustness certificates as well as to develop more effective attacks in the Wasserstein metric.

5.5.3 Experiments on color images (CIFAR-10)

[2] also apply their attack to color images in CIFAR-10. In this case, the attack does not transport probability mass between color channels: therefore, in our defense, it is sufficient to add noise in the flow domain to each channel independently to certify robustness (See Corollary D.2 in the appendix for a proof of the validity of this method). Certificates are presented in Table 5.2, while empirical robustness is presented in Figure 5.6. Again, we compare directly to models from

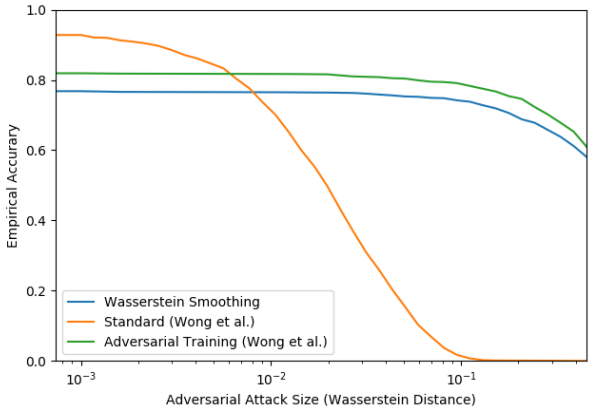


Figure 5.6: Comparison of empirical robustness on CIFAR-10 to models from [2]. Wasserstein smoothing is with $\sigma = 0.0002$. (This is the amount of noise which maximizes certified robustness, as seen in Table 5.2.) Note that we test on a random sample of 1000 images from CIFAR-10, rather than the entire data set.

[2]. We note that again, empirically, our model significantly outperforms an unprotected model, but is not as robust as a model trained adversarially. We also note that the certified robustness is orders of magnitude smaller than computed for MNIST: however, the unprotected model is also significantly less robust empirically than the equivalent MNIST model.

Chapter 6: (De)Randomized Smoothing for Certifiable Defense against Patch Attacks[1]

6.1 Introduction

In many instances the threat models considered for adversarial attacks (e.g. small ℓ_∞ distortions to every pixel of an image) implicitly require the attacker to be able to directly interfere with the input to a neural network. This limits practicality of such attacks as well as defenses against them. On the other hand, the development of *physical* adversarial attacks [11], in which small visible changes are made to real world objects in order to disrupt classification of images of these objects, represents a more concerning security threat. Unlike ℓ_p attacks, physical adversarial attacks can be perceptible (e.g. adding an adversarial sticker on a stop sign is a perceptible change). Nevertheless, humans would still correctly classify the attacked image while the classification model would fail to predict the correct label. Therefore, the attacked image is an adversarial example.

Physical adversarial attacks can often be modeled as "patch" adversarial attacks, in which the attacker can make arbitrary changes to pixels within a region of bounded size. Indeed, there is often a direct relationship between the two: for example, the universal patch attack proposed by [52] is an effective physical sticker attack. The attack method proposed in [52] is universal

[1] A form of chapter has been published in NeurIPS 2020 [29].

in a sense that pixels of the adversarial patch do not depend on the attacked image. Image-specific patch attacks have also been proposed, such as LaVAN [53], which reduces ImageNet classification accuracy to 0% using only a 42×42 pixel square patch (on images of size 299×299). In this paper, we consider *all* attacks (image-specific or universal) on square patches of size $m \times m$.

Practical defenses against patch attacks have been proposed.[54, 55] For the aforementioned 42×42 pixel attacks on ImageNet, [55] claims the current state-of-the-art practical defense. However, [4] has recently broken this defense, reducing the classification accuracy on ImageNet to 14%. In the same work, [4] also proposes the first *certified* defense against patch adversarial attacks, which uses interval bound propagation [23]. In a certifiably robust classification scheme, in addition to providing a classification for each image, the classifier may also return an assurance that the classification will provably not change under any distortion of a certain magnitude and threat model. One then reports both the *clean accuracy* (normal accuracy) of the model, as well as the *certified accuracy* (percent of images which are both correctly classified, and for which it is guaranteed that the classification will not change under a certain attack type). Unlike practical defenses, certified defenses guarantee that *no* future adversary (under a certain threat model) will break the defense.

The certified defense proposed by [4], however, does not scale well to practical classification tasks on complex inputs such as CIFAR-10 or ImageNet samples. Specifically, while this certified defense performs well on MNIST, it achieves poor certified accuracy on CIFAR-10 and, to quote from the paper itself, "is unlikely to scale to ImageNet." In this work, we propose a certified defense against patch attacks which overcomes these issues. In particular, our certifiable defense method leads to the following results:

Dataset and Attack Size	Chiang et al. [4] Certified Acc (Clean Acc)	Our method Certified Acc (Clean Acc)
MNIST 5×5	**60.4% (92.0%)**	52.44% (96.54%)
CIFAR 5×5	30.3% (47.8%)	**57.58% (83.82%)**
ImageNet 42×42	N/A	**13.9% (44.6%)**

Table 6.1: Comparison of the certified accuracy of our defense vs. [4]. For each technique, we report the certified and clean accuracies of the model with parameters giving the highest certified accuracy.

Notably, our method achieves a more than 27 percentage point increase in certified robustness on CIFAR-10 compared to [4]. Moreover, our method has top-1 *certified* accuracy on ImageNet classification which is approximately equal to the 14% *empirical* accuracy of the state-of-the art practical defense [55] under the attack proposed by [4] (although our clean accuracy is lower, 44% vs. 71%). On MNIST, which is often regarded as a toy dataset in deep learning applications, our method also achieves a relatively high certified robustness (but not as high as the method of [4]) and clean accuracy (slightly higher than that of [4]). Further, the certified defense proposed by [4] also has a computationally expensive training algorithm: the training time for the reported best model was 8.4 GPU hours for MNIST, and 15.4 GPU hours for CIFAR-10, using NVIDIA 2080 Ti GPUs. Our models, by contrast, took approximately 1.0 GPU hour to train on MNIST, and 2.5 GPU hours to train on CIFAR-10, on the same model of GPU.

Our certifiably robust classification scheme is based on *randomized smoothing*, a class of certifiably robust classifiers which have been proposed for various threat models, including ℓ_2 [33, 63, 91], ℓ_1 [64] and ℓ_0 [1] and Wasserstein (Chapter 5) metrics. All of these methods rely on a similar mechanism where noisy versions of an input image $\boldsymbol{x}$ are used in the classification. Such noisy inputs are created either by adding random noise to all pixels [64] or by removing (*ablating*) some of the pixels (Chapter 2 above). A large number of noisy images are then classified by a

base classifier and then the consensus of these classifications is reported as the final classification result. For an adversarial image $\boldsymbol{x}'$ at a bounded distance from $\boldsymbol{x}$, the probability distributions of possible noisy images which can be produced from $\boldsymbol{x}$ and $\boldsymbol{x}'$ will substantially overlap. This implies that, if a sufficiently large fraction of noisy images derived from $\boldsymbol{x}$ are classified to some class c, then with high confidence, a plurality of noisy images derived from $\boldsymbol{x}'$ will also be assigned to this class.

Patch adversarial attacks can be considered a special case of ℓ_0 (sparse) adversarial attacks: in an ℓ_0 attack, the adversary can choose a limited number of pixels and apply unbounded distortions to them. A patch adversarial attack is therefore a sparse adversarial attack where the attacker is additionally constrained to selecting only a block of adjacent pixels to attack, rather than any arbitrary pixels. One state-of-the-art certified defense against sparse adversarial attacks is the randomized smoothing method proposed in Chapter 2. In this method, a base classifier, $f(\boldsymbol{x})$, is trained to make classifications based on only a small number of independently randomly-selected pixels: the rest of the image is *ablated*, meaning that it is encoded as a null value. At test time, the final classification $g(\boldsymbol{x})$ is taken as the class most likely to be returned by f on a randomly ablated version of the image. In practice, we find that applying the defense method developed in Chapter 2 for sparse attacks directly to patch attacks yields poor results (see Figure 6.1). This is because the defense proposed in Chapter 2 does not incorporate the additional structure of the attack. For patch attacks, we can use the fact that the attacked pixels form a contiguous square to develop a more effective defense. In this paper, we propose a *structured ablation* scheme, where instead of independently selecting pixels to use for classification, we select pixels in a correlated way in order to reduce the probability that the adversarial patch is sampled. Empirically, structured ablation certificates yields much improved certified accuracy to patch attacks, compared to

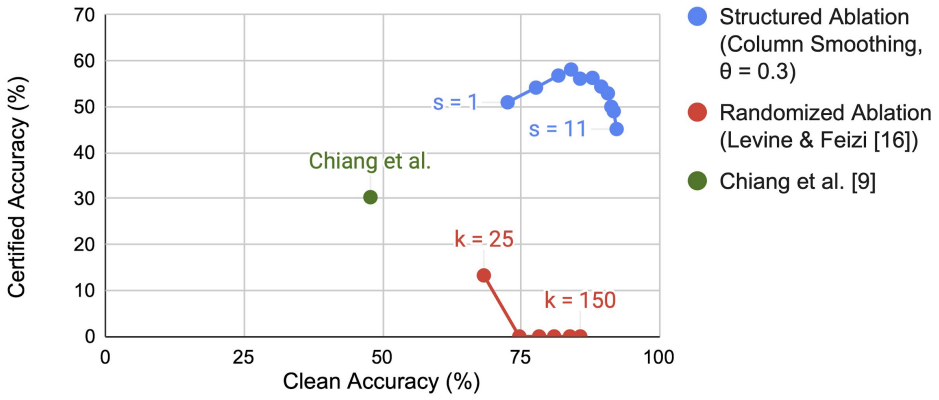


Figure 6.1: Clean and Certified accuracies for 5×5 adversarial patches on CIFAR-10. We compare our proposed method, Structured Ablation (for a range of its hyperparameter s) with the certified defense for patch attacks proposed by [4], and with a naive application of the ℓ_0 defense proposed in Chapter 2 (for a range of that technique's hyperparameter k). Our defense achieves significantly higher certified and clean accuracies compared to the other methods.

the naive ℓ_0 certificate.

By reducing the total number of possible ablations of an image, structured ablation allows us to de-randomize our algorithm, yielding improved, deterministic certificates. For ℓ_0 robustness, Randomized Ablation (Chapter 2) achieves the largest median certificates on MNIST by using a base classifier f which classifies using only 45 out of 784 pixels. There are $\binom{784}{45} \approx 4 \times 10^{73}$ ways to make this selection. It is therefore not feasible to evaluate precisely the probability that $f(\boldsymbol{x})$ returns any particular class c: one must estimate this based on random samples.[2] Using our proposed methods, the number of possible ablations is small enough so that it is tractable to classify using all possible ablations: we can *exactly* evaluate the probability that $f(\boldsymbol{x})$ returns each class. Our certificate is therefore exact, rather than probabilistic, so our classifications are provably robust in an absolute sense.

Determinism provides another benefit: the absence of estimation error increases the certified accuracies that can be reported. Additionally, because estimation error is no longer a concern,

[2]Note that this chapter was originally published before the deterministic ℓ_0 smoothing method discussed in Section 4.5 was written.

derandomization allows us to use more rich information from the base classifier without incurring an additional cost in increased estimation error. We take advantage of this to allow the base classifier to abstain in cases where it cannot make a high-confidence prediction towards any class. This leads to substantially increased certificates on MNIST, although the effects on CIFAR-10 are not significant.

After the initial distribution of this work, [30] improved upon it by proposing a tighter certificate. In a concurrent work to ours, [31] also proposes a method similar to "block smoothing" proposed below. After the publication of this work, several subsequent works have proposed improved certification techniques using a similar framework [95, 96, 97, 98].

6.2 Certifiable Defenses against Patch Attacks

6.2.1 Baseline: Sparse Randomized Ablation (Chapter 2)

As mentioned in the introduction, patch attacks can be regarded as a restricted case of ℓ_0 attacks. In particular, let ρ be the magnitude of an ℓ_0 adversarial attack: the attacker modifies ρ pixels and leaves the rest unchanged. A patch attack, with an $m \times m$ adversarial patch, is also an ℓ_0 attack, with $\rho = m^2$. We can then attempt to apply existing certifiably robust classification schemes for the ℓ_0 threat model to the patch attack threat model: we simply need to certify to an ℓ_0 radius of $\rho = m^2$. Consider specifically the ℓ_0 smoothing-based certifiably robust classifier introduced in Chapter 2. In this classification scheme, given an input image $\boldsymbol{x}$, the base classifier f classifies a large number of distinct randomly-ablated versions of $\boldsymbol{x}$, in each of which only k pixels of the original image are randomly and independently selected to be retained and used by the base classifier f. Therefore, for *any choice* of ρ pixels that the attacker could choose to attack,

the probability that any of these ρ pixels is also one of the k pixels used in f's classification is:

$$\begin{aligned}\Delta &:= \Pr(f \text{ uses attacked pixels}) \\ &= 1 - \frac{\binom{hw-\rho}{k}}{\binom{hw}{k}} \approx k\frac{\rho}{hw} = \frac{km^2}{hw} \quad (k, \rho << hw),\end{aligned}$$

where ρ is the number of attacked pixels, k is the number of retained pixels used by the base classifier, and the overall dimensions of the input image $\boldsymbol{x}$ are $h \times w$. To understand this, note that the classifier has k opportunities to choose an attacked pixel, and ρ out of hw pixels are attacked. Clearly, if f does not use any of the attacked pixels, then its output will not be corrupted by the attacker. Therefore, the attacker can change the output of $f(\boldsymbol{x})$ with probability at most Δ. Let c be the majority classification at $\boldsymbol{x}$ (i.,e., $g(\boldsymbol{x}) = c$). If $f(\boldsymbol{x}) = c$ with probability greater than $0.5 + \Delta$, then for any distorted image $\boldsymbol{x}'$, one can conclude that $f(\boldsymbol{x}') = c$ with probability greater than 0.5, and therefore that $g(\boldsymbol{x}') = c$. As discussed in the introduction, while this technique produces state-of-the-art guarantees against general ℓ_0 attacks, it yields rather poor certified accuracies when applied to patch attacks, because it does not take advantage of the structure of the attack (See Figure 6.1; data for MNIST are provided in supplementary material.).

6.2.2 Proposed Method: Structured Ablation

To exploit the restricted nature of patch attacks, we propose two *structured ablation* methods, which select correlated groups of pixels to reduce the probability Δ that the adversarial patch is sampled:

- **Block Smoothing**: In this method, we select a single $s \times s$ square block of pixels, and ablate the rest of the image. The number of retained pixels is then $k = s^2$. Note that for an $m \times m$

adversarial patch, out of the $h \times w$ possible selections for blocks to use for classification, $(m + s - 1)^2$ of them will intersect the patch. Thus, we have:

$$\Delta_{\text{block}} = \frac{(m+s-1)^2}{hw} = \frac{(m+\sqrt{k}-1)^2}{hw} < \frac{4\max(m^2,k)}{hw}. \tag{6.1}$$

As illustrated in Figure 6.2, this implies a substantially decreased probability of intersecting the adversarial patch, compared to sampling k pixels independently.

- **Band Smoothing**: In this method, we select a single band (a **column** or a **row**) of pixels of width s, and ablate the rest of the image. In the case of a column, the number of retained pixels is then $k = sh$. For an $m \times m$ adversarial patch, out of the w possible selections for bands to use for classification, $m + s - 1$ of them will intersect the patch. Then we have:

$$\Delta_{\text{col.}} = \frac{m+s-1}{w} = \frac{m+k/h-1}{w} < \frac{2\max(hm,k)}{hw}. \tag{6.2}$$

For both of these methods, it is tractable to use the base classifier to classify all possible ablated versions of an image (i.e. hw and w possible ablations for block and column smoothing, respectively). This allows us to exactly compute the smoothed classifier, $g(\boldsymbol{x})$, yielding deterministic certificates.

Our experiments show that structured ablation produces higher certified accuracy than ℓ_0 randomized ablation. This is because, for similar values of Δ, structured ablation methods yield much higher base classifier accuracies (Figure 6.3). Empirically, we find that the band method (and specifically, column smoothing) produces the most certifiably robust classifiers (Figure 6.5). In supplementary materials, we explore structured ablation using multiple blocks or bands of

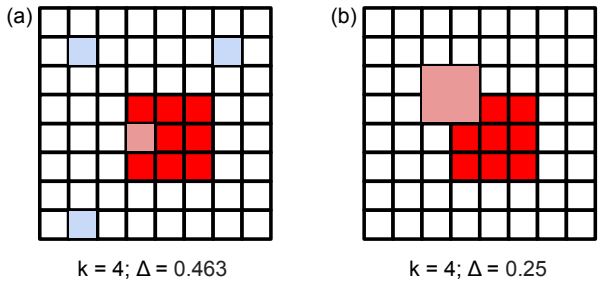


Figure 6.2: Likelihood of selecting a pixel which is part of the attacked patch (red) for (a) sparse randomized ablation, as proposed in Chapter 2 (b) Structured ablation, using a block of size $s = 2$. In both cases, $k = 4$ pixels are retained. However, in the sparse case, if *any* of the four independently-selected pixels sample the patch, then the classification may be impacted: this occurs with probability $\Delta = 1 - \binom{64-9}{4} / \binom{64}{4} \approx 0.463$. In contrast, the probability that the block overlaps with the adversarial patch is only $\frac{16}{64} = 0.25$.

pixels.

We now explicitly describe our algorithms, starting with block smoothing. For an input image $\boldsymbol{x}$, let the base classifier be specified as $f_c(\boldsymbol{x}, s, x, y)$, where $\boldsymbol{x}$ is the input image, s is the block size, (x, y) is the position of the retained block, and $c \in \mathbb{N}$ is a class label. In other words, $f(\boldsymbol{x}, s, x, y)$ is the base classification, where the classifier uses only the pixels in an $s \times s$ block with upper-left corner (x, y) (If the retained block would exceed the borders of the image, it wraps around: see Figure 6.4). For each class c, $f_c(\boldsymbol{x}, s, x, y)$ will either be 0 or 1; however, note that we *do not* require that $f_c(\boldsymbol{x}, s, x, y) = 1$ for any class c (it may abstain, returning zero for all classes), and we also allow for $f_c(\boldsymbol{x}, s, x, y)$ to equal 1 for multiple classes (see Section 6.2.2.1 for details). To make our final classification and compute our robustness certificate, we count the number of blocks on which the base classifier returns each class:

$$\forall c, \;\; n_c(\boldsymbol{x}) := \sum_{x=1}^{w} \sum_{y=1}^{h} f_c(\boldsymbol{x}, s, x, y) \tag{6.3}$$

The final smoothed classification is simply the plurality class returned: $g(\boldsymbol{x}) := \arg\max_c n_c(\boldsymbol{x})$.

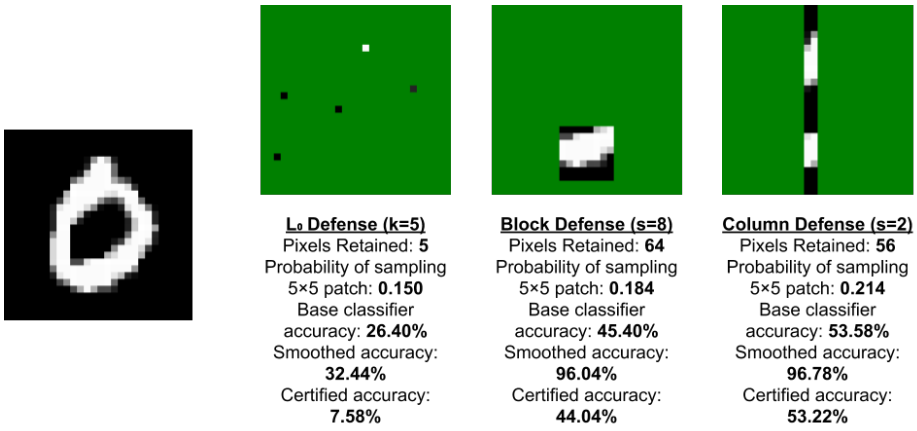


Figure 6.3: Comparison of the ℓ_0 defense proposed in Chapter 2 to the newly-proposed defenses on MNIST, for a 5×5 patch attack. While sampling a single block or band slightly increases the probability Δ that an adversarially distorted pixel is used, the large increase in the total number of retained pixels, and therefore the base classifier accuracy, more than makes up for this increase in Δ. However, the number of retained pixels alone does not perfectly correspond to higher base classifier accuracy: while the band method uses slightly fewer pixels than the block method, the base classifier has substantially higher accuracy, leading to higher certified accuracy.

In the case of ties, we deterministically return the smaller-indexed class. Because the adversarial patch only intersects $(m + s - 1)^2$ blocks, the adversary can only alter the output of $(m + s - 1)^2$ of the evaluations of the base classifier. This yields the following guarantee:

Theorem 6.1. *For any image $\boldsymbol{x}$, base classifier f, smoothing block size s, and patch size m, if:*

$$n_c(\boldsymbol{x}) \geq \max_{c' \neq c} \left[n_{c'}(\boldsymbol{x}) + \mathbf{1}_{c>c'} \right] + 2(m + s - 1)^2 \tag{6.4}$$

then for any image $\boldsymbol{x}'$ which differs from $\boldsymbol{x}$ only in an $(m \times m)$ patch, $g(\boldsymbol{x}') = c$.

In Theorem 6.1, the indicator function term $(\mathbf{1}_{c>c'})$ is present because we break ties deterministically by label index during the final classification. Proofs are provided in supplementary materials.

Note that the classifier counts $n_c(\boldsymbol{x})$ can be though of as exact estimates for the probability

that the base classifier returns the class c, simply scaled up by a factor of hw. For the column smoothing case (or row smoothing, by simple transpose), we can compute a similar certificate. In this case, the base classifier is $f_c(\boldsymbol{x}, s, x)$, where s is now the width of the retained column of pixels, and x is the position of the leftmost edge of this column. We then only need to sum over one dimension:

$$\forall c, \;\; n_c(\boldsymbol{x}) \coloneqq \sum_{x=1}^{w} f_c(\boldsymbol{x}, s, x). \tag{6.5}$$

Again, we classify using $g(\boldsymbol{x}) \coloneqq \arg\max_c n_c(\boldsymbol{x})$. To derive the final guarantee, we now use that the adversarial patch will overlap with only $(m + s - 1)$ columns:

Theorem 6.2. *For any image $\boldsymbol{x}$, base classifier f, smoothing column size s, and patch size m, if:*

$$n_c(\boldsymbol{x}) \geq \max_{c' \neq c} \left[n_{c'}(\boldsymbol{x}) + \mathbf{1}_{c>c'} \right] + 2(m + s - 1) \tag{6.6}$$

then for any image $\boldsymbol{x}'$ which differs from $\boldsymbol{x}$ only in an $(m \times m)$ patch, $g(\boldsymbol{x}') = c$.

6.2.2.1 Implementation Details

In practice, we use a deep network as our base classifier, and set $f_c(\boldsymbol{x}, s, x, y) = 1$ if the logit corresponding to class c is greater than a threshold hyperparameter θ. This allows the base classifier to abstain from classifying in the case that there is no usable information in the retained block, as well as to "vote" for multiple classes, which may be beneficial if the base classifier top-1 accuracy is low.

The input of to the neural network used as the base classifier is a copy of the image $\boldsymbol{x}$, with all pixels except for those in the retained block or band replaced with a specially-encoded

'NULL' value. We encode the additional 'NULL' value in the input in the same manner described for randomized ablation in Chapter 2 for each dataset tested: this involves adding additional color channels, so that the NULL value is distinct from all real pixel colors. During training, as in prior smoothing works, we train f on ablated samples, using a single randomly-determined ablation pattern (selection of block or column to retain) on all samples in each batch.

6.2.3 Comparison to Conventional Randomized Smoothing

In conventional randomized smoothing, rather than computing the probability that f returns each class directly, one must instead lower-bound, with high confidence, the probability p_c that f returns the plurality class c and upper-bound the probabilities $p_{c'}$ that f returns all other classes, based on samples. This leads to decreased certified accuracy due to estimation error. Additionally, all of these bounds must hold simultaneously: in order to ensure that the gap between p_c and $p_{c'}$ is sufficiently large for each c' to prove robustness, one must bound the population probabilities for *every* class. Some works [64, 99] do this directly using a union bound, leading to increased error as the number of classes increases. Others, following [63], instead only use samples to lower-bound the probability p_c that the base classifier returns the top class. One can then upper bound all other class probabilities by observing that $\forall c',\ p_{c'} \leq 1 - p_c$. In other words, rather than determining whether c will stay the plurality class at an adversarial point, one instead

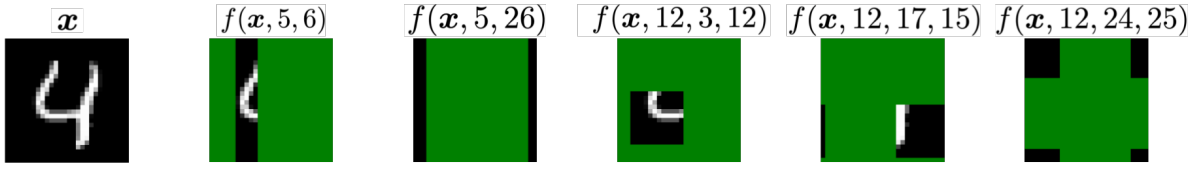


Figure 6.4: Representation of which pixels are used by the base classifier f, as a function of indexing. Ablated pixels are represented in green.

determines whether c will stay the *majority* class. This is also the estimation method used in Chapter 2 for ℓ_0 certificates: this is why, when describing that method in Section 6.2.1, we gave the condition for certification as $p_c > 0.5 + \Delta$. In our deterministic method, we can use a less strict condition, that $\forall c',\ p_c - p_{c'} > 2\Delta$, where $p_c = n_c/hw$ for block smoothing, and $p_c = n_c/w$ for column smoothing. (As described above, we can sometimes even certify in the *equality* case, when it is assured that c will be selected if there is a tie between the class probabilities at the distorted point.)

In this work, we sidestep the estimation problem entirely by computing the population probabilities exactly. This substantially reduces evaluation time: for example, column smoothing on CIFAR-10 requires 32 forward passes, compared to 10^4 – 10^5 for randomized ablation (Chapter 2). (We provide measured evaluation times in supplementary material.) However, by avoiding the assumption of [63], that all probability not assigned to c is instead assigned to a single adversarial class, we can make an additional optimization: we can add an 'abstain' option. If there is no compelling evidence for any particular class in an ablated image (i.e., if all logits are below a threshold value θ), our classifier abstains. This prevents blocks which contain no information from being assigned to an arbitrary, likely incorrect class. Figure 6.5-a shows that this significantly increases the certified accuracy on MNIST, although it has little effect on CIFAR-10. Our threshold system also allows the base classifier to select *multiple* classes, if there is strong evidence for each of them. This is intended to increase certified accuracy in the case of a large number of classes, where the top-1 accuracy of the base classifier might be low: if the correct class consistently occurs within the top several classes, it may still be possible to certify robustness.

In a concurrent work, [6] also proposes a derandomization of a randomized smoothing tech-

nique. However, the threat model considered is quite different: [6] develops a defense against label-flipping poisoning attacks, where the adversary changes the labels of training samples. Notably, [6]'s result only applies directly to linear base classifiers. By making this restriction, [6] is able to analytically determine the probabilities of $f(\boldsymbol{x})$ returning each class. By contrast, our de-randomized technique for patch attacks does not restrict the architecture of the base classifier f, in practice a deep network.

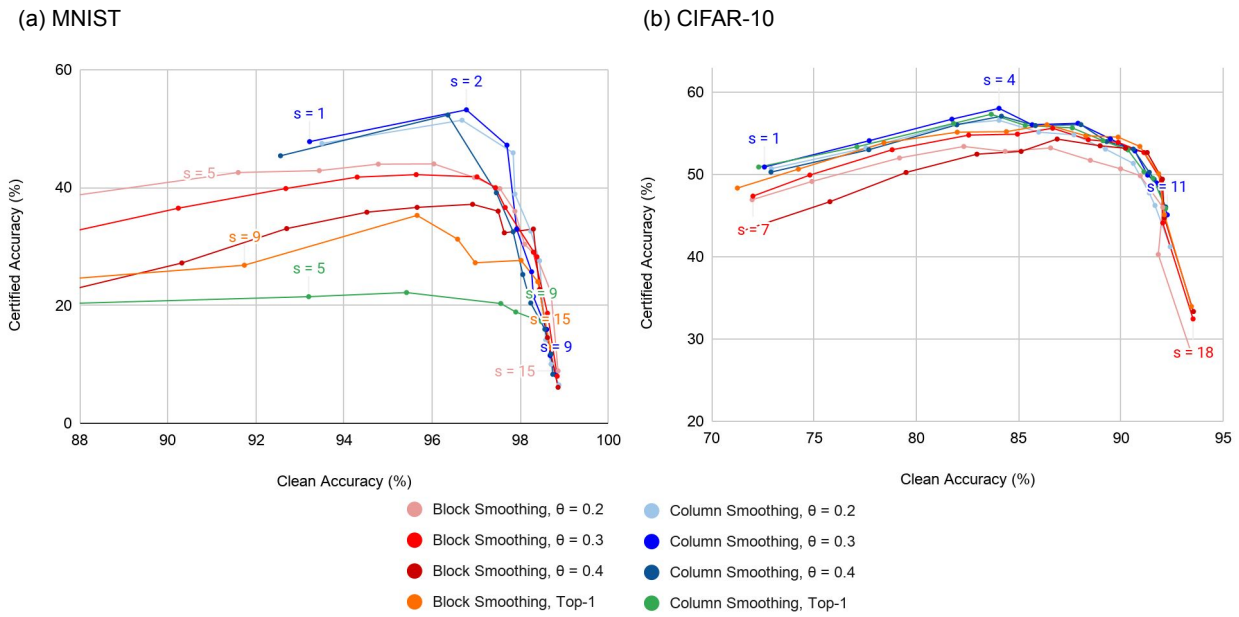


Figure 6.5: Validation set certificates for 5×5 patches on (a) MNIST, (b) CIFAR-10. Best certified accuracy is achieved using Column Smoothing for both datasets, with $s = 2, \theta = 0.3$ for MNIST and $s = 4, \theta = 0.3$ for CIFAR-10. Column smoothing (blue lines) gives better certified accuracies than block smoothing (red lines), but the effect is small on CIFAR-10.

6.3 Results

Certified robustness against patch attacks is presented for 5×5 patches on MNIST and CIFAR-10 in Figure 6.5, using both block and column smoothing (On MNIST, we also tested smoothing with rows rather than columns, with slightly worse results: see supplementary mate-

rials.) Results in the figures are using a validation set of 5,000 images; the final results reported in Table 6.1 are on a separate test set of 5,000 images. On both datasets, we have found that column smoothing produces better certified accuracies than block smoothing. However, the performance gap is larger on MNIST than on CIFAR-10. We have also tested with the base classifier returning only the top-one class, rather than thresholding the logits to abstain on low confidence predictions. We find that thresholding produces a large improvement on MNIST, but has had little effect on CIFAR-10. This is possibly because MNIST images, when ablated, will often have zero information (i.e., be entirely black), while in natural images, the retained region will always have some information. In both datasets, we found that the column smoothing certificates are not highly sensitive to the threshold hyperparameter θ.

Experiments using multiple blocks and columns, rather than just a single block or column for each base classification, are presented in supplementary materials.

In Figure 6.6, we show how our certificates scale to different patch sizes, beyond the standard 5×5. On CIFAR-10, we maintain high certified accuracy even at a patch size of 9×9. Notably, the optimal column width s seems not to depend on the patch size, suggesting that a single trained model can defend against a broad class of patch attacks.

On ImageNet-1000 (ILSVRC2012), we have tested certified robustness to 42×42 patch attacks with column smoothing alone, using column width $s = 25$, and over the θ hyperparameter range $\theta = \{0.1, 0.2, 0.3, 0.4\}$. We have used 1,000 images for validation, and 1,000 for test, using the optimal $\theta = 0.2$; test set results are presented in Table 6.1. Full validation results for all datasets are presented as tables in supplementary materials.

We also compare column smoothing certificates for MNIST and CIFAR-10 to *randomized* column smoothing smoothing certificates on both datasets: see Table 6.2. We find that the "deran-

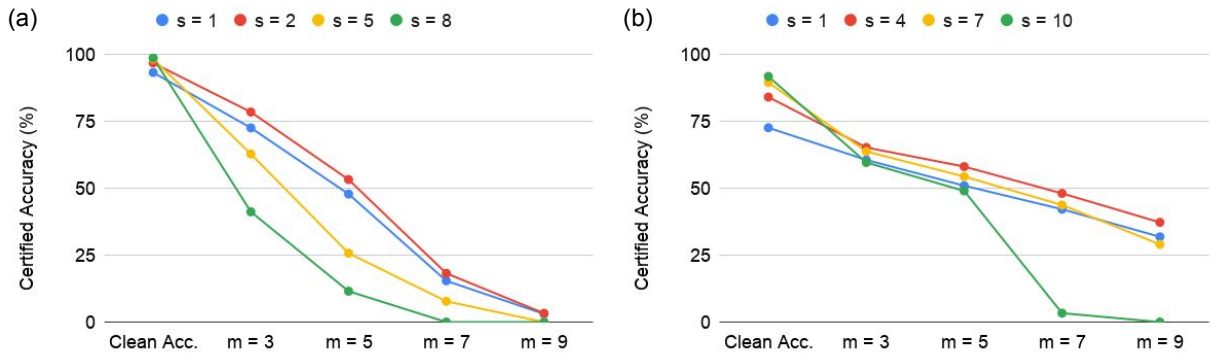


Figure 6.6: Validation set certificates for $m \times m$ patches on (a) MNIST, (b) CIFAR-10. For all experiments, we use Column Smoothing, $\theta = 0.3$. On CIFAR-10, we maintain high certified accuracy even at $m = 9$. The optimal column width s seems not to depend on the patch size, suggesting that a single trained model will defend against a broad class of patch attacks.

domization" improves the certificates independently of the effect of thresholding (for example, it increases the certified accuracy on CIFAR-10 by nearly 7 percentage points.)

Dataset	Derandomized $\theta = .3$	Derandomized Top-1 class	Randomized Column Smoothing
MNIST	53.22% (s = 2)	22.20% (s = 6)	16.32% (s = 6)
CIFAR-10	58.08% (s = 4)	57.36% (s = 4)	50.38% (s = 6)

Table 6.2: Comparison of Certified Accuracies for derandomized versus randomized structured ablation (column smoothing) for 5×5 adversarial patches for MNIST and CIFAR-10. We compare randomized structured ablation to both the "Top-1 class" method (without abstaining or thesholding) as well as to the thresholding method, with the optimal $\theta = 0.3$. Here, we show the certified accuracy for the optimal value of the hyperparameter s for each method: results for all s are presented in supplementary materials.

6.3.1 Empirical Robustness

We evaluated the empirical robustness of our method, specifically column smoothing, on CIFAR-10, using a modified version of the IFGSM patch attack from [4]. In particular, because the zero-one base-classifications f_c are non-differentiable, we cannot attack $n_c(\boldsymbol{x})$ directly. Instead, in order to generate the attacks, we use a surrogate model in which f_c returns SoftMax

scores. Note that this is similar to [33]'s attack on Gaussian-smoothed classifiers, but there is no need to consider random sampling in this case. Further details on the attack are provided in supplementary materials. Results are presented in Figure 6.7-a. We note that, as expected, our certified lower bounds hold, and furthermore that our model is significantly more robust to patch adversarial attacks compared with an undefended baseline model. We also evaluated the robustness of our attack to a non-patch adversarial attack, specifically an ℓ_∞-bounded IFGSM attack. Because all base classifiers are attacked simultaneously in this model, our method provides no robustness guarantee, and one might worry that the model could be particularly vulnerable to the attack. However, while the accuracy under this attack was reduced compared to an undefended baseline model, this was not a dramatic effect: see Figure 6.7-b.

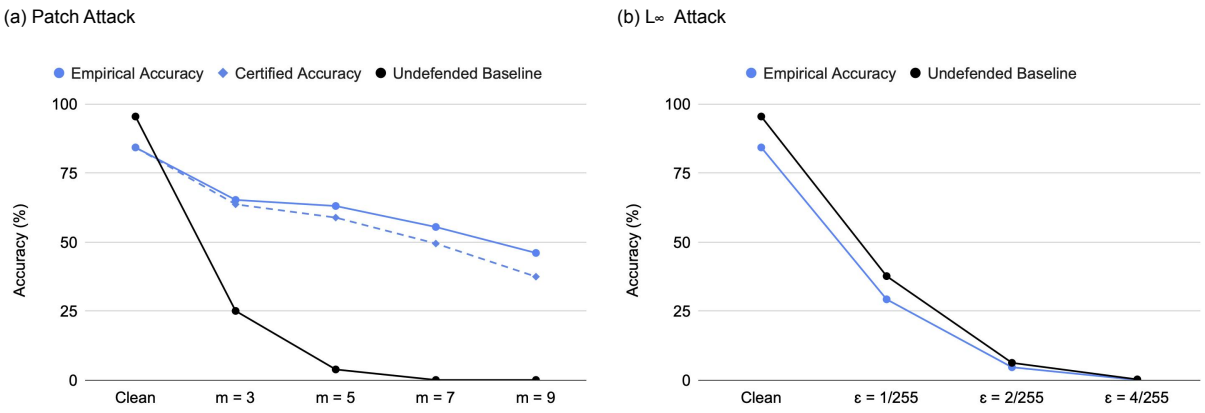


Figure 6.7: Empirical attacks against column smoothing on 500 images from the CIFAR-10 test set, versus an unprotected baseline model. We use optimal hyperparameters ($s = 4, \theta = 0.3$) for column smoothing. For the ℓ_∞ attack, we used IFGSM for 50 iterations and a step size of $0.5/255$. For the patch attack, we use the patch-IFGSM attack from [4], with 80 random starts, 150 iterations per random start, and step size of 0.05.

Chapter 7: Deep Partition Aggregation: Provable Defenses against General Poisoning Attacks[1]

Adversarial poisoning attacks are an important vulnerability in machine learning systems. In these attacks, an adversary can manipulate the training data of a classifier, in order to change the classifications of specific inputs at test time. Several poisoning threat models have been studied in the literature, including threat models where the adversary may insert new poison samples [100], manipulate the training labels [6, 101], or manipulate the training sample values [102, 103]. A certified defense against a poisoning attack provides a *certificate* for each test sample, which is a guaranteed lower bound on the magnitude of any adversarial distortion of the training set that can corrupt the test sample's classification. In this work, we propose certified defenses against two types of poisoning attacks:

General poisoning attacks: In this threat model, the attacker can insert or remove a bounded number of samples from the training set. In particular, the attack magnitude ρ is defined as the cardinality of the *symmetric difference* between the clean and poisoned training sets. This threat model also includes any distortion to an sample and/or label in the training set — a distortion of a training sample is simply the removal of the original sample followed by the insertion of the distorted sample. (Note that a sample distortion or label flip therefore increases

[1] A form of chapter has been published in ICLR 2021 [74].

the symmetric difference attack magnitude by two.)

Label-flipping poisoning attacks: In this threat model, the adversary changes only the label for ρ out of m training samples. [6] has recently provided a certified defense for this threat model, which we improve upon.

In the last couple of years, certified defenses have been extensively studied for evasion attacks, where the adversary manipulates the test samples, rather than the training data (e.g. [22, 23, 33, 58, 63, 64, 86, 91], etc.) In the evasion case, a certificate is a lower bound on the distance from the sample to the classifier's decision boundary: this guarantees that the sample's classification remains unchanged under adversarial distortions up to the certified magnitude.

[6] provides an analogous certificate for label-flipping poisoning attacks: for an input sample $\boldsymbol{x}$, the certificate of $\boldsymbol{x}$ is a lower bound on the number of labels in the training set that would have to change in order to change the classification of $\boldsymbol{x}$.[2] [6]'s method is an adaptation of a certified defense for sparse (ℓ_0) evasion attacks proposed by [1]. The adapted method for label-flipping attacks proposed by [6] is equivalent to randomly flipping each training label with fixed probability and taking a consensus result. If implemented directly, this would require one to train a large ensemble of classifiers on different noisy versions of the training data. However, instead of actually doing this, [6] focuses only on linear classifiers and is therefore able to analytically calculate the expected result. This gives deterministic, rather than probabilistic, certificates. Further, because [6] considers a threat model where only labels are modified, they are able to train an *unsupervised* nonlinear feature extractor on the (unlabeled) training data before applying their technique, in order to learn more complex features.

[2] [104] also refers to a "certified defense" for poisoning attacks. However, the definition of the certificate is substantially different in that work, which instead provides overall accuracy guarantees under the assumption that the training and test data are drawn from similar distributions, rather than providing guarantees for individual realized inputs.

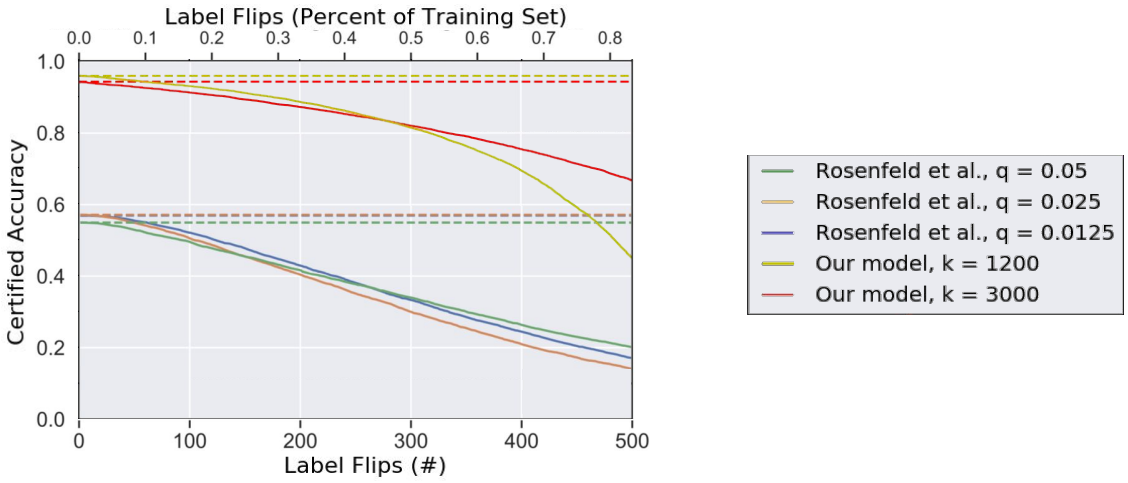


Figure 7.1: Comparison of certified accuracy to label-flipping poison attacks for our defense (SS-DPA algorithm) vs. [6] on MNIST. Solid lines represent certified accuracy as a function of attack size; dashed lines show the clean accuracies of each model. Our algorithm produces substantially higher certified accuracies. Curves for [6] are adapted from Figure 1 in that work. The parameter q is a hyperparameter of [6]'s algorithm, and k is a hyperparameter of our algorithm: the number of base classifiers in an ensemble.

Based on the improved provable defense against ℓ_0 evasion attacks proposed in Chapter 2, in this paper, we develop certifiable defenses against general and label-flipping poisoning attacks that significantly outperform the current state-of-the-art certifiable defenses. In particular, we develop a certifiable defense against *general* poisoning attacks called **Deep Partition Aggregation (DPA)** which is based on partitioning the training set into k partitions, with the partition assignment for a training sample determined by a hash function of the sample. The hash function can be any deterministic function that maps a training sample $\mathbf{t}$ to a partition assignment: the only requirement is that the hash value depends only on the value of the training sample $\mathbf{t}$ itself, so that neither poisoning other samples, nor changing the total number of samples, nor reordering the samples can change the partition that $\mathbf{t}$ is assigned to. We then train k base classifiers separately, one on each partition. At the test time, we evaluate each of the base classifiers on the test sample $\boldsymbol{x}$ and return the plurality classification c as the final result. The key insight is that removing

a training sample, or adding a new sample, will only change the contents of one partition, and therefore will only affect the classification of one of the k base classifiers. This immediately leads to robustness certifications against general poisoning attacks which, to the best of our knowledge, is the first one of this kind.

If the adversary is restricted to flipping labels only (as in [6]), we can achieve even larger certificates through a modified technique. In this setting, the unlabeled data is trustworthy: each base classifier in the ensemble can then make use of the entire training set without labels, but only has access to the labels in its own partition. Thus, each base classifier can be trained as if the entire dataset is available as unlabeled data, but only a very small number of labels are available. This is precisely the problem statement of semi-supervised learning [105, 106, 107, 108, 109]. We can then leverage these existing semi-supervised learning techniques directly to improve the accuracies of the base classifiers in DPA. Furthermore, we can ensure that a particular (unlabeled) sample is assigned to the same partition regardless of label, so that only one partition is affected by a label flip (rather than possibly two). The resulting algorithm, **Semi-Supervised Deep Partition Aggregation (SS-DPA)** yields substantially increased certified accuracy against label-flipping attacks, compared to DPA alone and compared to the current state-of-the-art. Furthermore, while our method is de-randomized (as [6] is) and therefore yields deterministic certificates, our technique does not require that the classification model be linear, allowing deep networks to be used.

On MNIST, SS-DPA substantially outperforms the existing state of the art [6] in defending against label-flip attacks: we certify at least half of images in the test set against attacks to over 600 (1.0%) of the labels in the training set, while still maintaining over 93% accuracy (See Figure 7.1, and Table 7.1). In comparison, [6]'s method achieves less than 60% clean accuracy on

MNIST, and most test images cannot be certified with the correct class against attacks of even 200 label flips. We are also the first work to our knowledge to certify against general poisoning attacks, including insertions and deletions of new training images: in this domain, we can certify at least half of test images against attacks consisting of over 500 arbitrary training image insertions or deletions. On CIFAR-10, a substantially more difficult classification task, we can certify at least half of test images against label-flipping attacks on over 300 labels using SS-DPA (versus 175 label-flips for [6]), and can certify at least half of test images against general poisoning attacks of up to nine insertions or deletions using DPA. To see how our method performs on datasets with larger numbers of classes, we also tested our methods on the German Traffic Sign Recognition Benchmark [110], a task with 43 classes and on average $\approx$ 1000 samples per class. Here, we are able to certify at least half of test images as robust to 176 label flips, or 20 general poisoning attacks. **These results establish new state-of-the-art in provable defenses against label-flipping and general poisoning attacks.**

7.1 Related Works

[58] (Chapter 2 of this dissertation) propose a *randomized ablation* technique to certifiably defend against sparse attacks. That method *ablates* some pixels, replacing them with a null value. Since it is possible for the base classifier to distinguish exactly which pixels originate from $\boldsymbol{x}$, this results in more accurate base classifications and therefore substantially greater certified robustness than [1]. For example, on ImageNet, [1] certifies the median test image against distortions of one pixel, while our ℓ_0 method certifies against distortions of 16 pixels.

Our proposed method is related to classical ensemble approaches in machine learning,

namely bootstrap aggregation and subset aggregation [56, 111, 112, 113]. However, in these methods each base classifier in the ensemble is trained on an independently sampled collection of points from the training set: multiple classifiers in the ensemble may be trained on (and therefore poisoned by) the same sample point. The purpose of these methods has typically been to improve generalization. Bootstrap aggregation has been proposed as an *empirical* defense against poisoning attacks [114] as well as for evasion attacks [115]. However, at the time of the initial distribution of this work, these techniques had not yet been used to provide *certified* robustness.[3] Our unique *partition aggregation* variant provides deterministic robustness certificates against poisoning attacks. See Appendix F.4 for further discussion.

[75] have recently proposed a different randomized-smoothing based defense against poisoning attacks by directly applying [63]'s smoothing ℓ_2 evasion defense to the poisoning domain. The proposed technique can only certify for clean-label attacks (where only the existing samples in the dataset are modified, and not their labels), and the certificate guarantees robustness only to bounded ℓ_2 distortions of the training data, where the ℓ_2 norm of the distortion is calculated across all pixels in the *entire* training set. Due to well-known limitations of dimensional scaling for smoothing-based robustness certificates [51, 123, 124], this yields certificates to only very small distortions of the training data. (For binary MNIST [13,007 images], the maximum reported ℓ_2 certificate is 2 pixels.) Additionally, when using deep classifiers, [75] proposes a randomized certificate, rather than a deterministic one, with a failure probability that decreases to zero only as the number of trained classifiers in an ensemble approaches infinity. Moreover, in [75], unlike in our method, each classifier in the ensemble must be trained on a noisy version

[3] In a concurrent work, [116] consider using bootstrap aggregation directly for certified robustness. Their certificates are therefore probabilistic, and are not as large, in median, as the certificates reported here for MNIST and CIFAR-10. After the publication of this work, several methods have been proposed which extend or improve upon the framework presented here: [117, 118, 119, 120, 121, 122]

of the entire dataset. These issues hinder [75]'s method to be an effective scheme for certified robustness against poisoning attacks. After the initial distribution of this work, a recent revision of [6] has suggested using randomised smoothing techniques on training samples, rather than just training labels, as a general approach to poisoning defense. Both this work and [75] could be considered as implementations of this idea, although this generalized proposal in [6] does not include a derandomization scheme (unlike [6]'s proposed derandomized defense against label-flipping attacks) .

Other prior works have provided *distributional*, rather than *pointwise* guarantees against poisoning attacks. In these works, there is a (high-probability) guarantee that the classifier will achieve a certain level of average overall accuracy on test data, under the assumption that the test data and clean (pre-poisoning) training data are drawn from the same distribution. These works do not provide any guarantees that apply to specific test samples, however. As mentioned above, [104] provides such a distributional guarantee, specifically for a threat model of addition of poison samples. Other such works include [125], which considers label-flipping attacks and determines conditions under which PAC-learning is possible in the presence of such attacks, and [126] provides similar guarantees for replacement of samples. Other works [127, 128] provide distributional guarantees for unsupervised learning under poisoning attacks. [129] proposes provably effective poisoning *attacks* with high-probability pointwise guarantees of effectiveness on test samples. However, that work relies on properties of the distribution that the training set is drawn from. [130] provide a robust training algorithm which provably approximates the clean trained model despite poisoning (rather than the behavior at a certain test point): this result also makes assumptions about the distribution of the clean training data.

7.2 Proposed Methods

7.2.1 Notation

Let $\mathcal{S}$ be the space of all possible unlabeled samples (i.e., the set of all possible images). We assume that it is possible to sort elements of $\mathcal{S}$ in a deterministic, unambiguous way. In particular, in the case of image data, we can sort images lexicographically by pixel values: in general, any data that can be represented digitally will be sortable. We represent labels as integers, so that the set of all possible labeled samples is $\mathcal{S}_L := \{(\mathbf{x}, c) | \mathbf{x} \in \mathcal{S}, c \in \mathbb{N}\}$. A training set for a classifier is then represented as $T \in \mathcal{P}(\mathcal{S}_L)$, where $\mathcal{P}(\mathcal{S}_L)$ is the power set of $\mathcal{S}_L$. For $\boldsymbol{t} \in \mathcal{S}_L$, we let $\textbf{sample}(\boldsymbol{t}) \in \mathcal{S}$ refer to the (unlabeled) sample, and $\textbf{label}(\boldsymbol{t}) \in \mathbb{N}$ refer to the label. For a set of samples $T \in \mathcal{P}(\mathcal{S}_L)$, we let $\textbf{samples}(T) \in \mathcal{P}(\mathcal{S})$ refer to the set of unique unlabeled samples which occur in T. A classifier model is defined as a deterministic function from *both* the training set *and* the sample to be classified to a label, i.e. $f : \mathcal{P}(\mathcal{S}_L) \times \mathcal{S} \to \mathbb{N}$. We will use $f(\cdot)$ to represent a base classifier model (i.e., a neural network), and $g(\cdot)$ to refer to a robust classifier (using DPA or SS-DPA).

$A \ominus B$ represents the set symmetric difference between A and B: $A \ominus B = (A \setminus B) \cup (B \setminus A)$. The number of elements in A is $|A|$, $[n]$ is the set of integers 1 through n, and $\lfloor z \rfloor$ is the largest integer less than or equal to z. $\mathbf{1}$ represents the indicator function: $\mathbf{1}_{\text{Prop}} = 1$ if Prop is true; $\mathbf{1}_{\text{Prop}} = 0$ otherwise. For a set A of sortable elements, we define $\textbf{Sort}(A)$ as the sorted list of elements. For a list L of unique elements, for $\boldsymbol{l} \in L$, we will define $\textbf{index}(L, l)$ as the index of l in the list L.

7.2.2 DPA

The Deep Partition Aggregation (DPA) algorithm requires a base classifier model $f : \mathcal{P}(\mathcal{S}_L) \times \mathcal{S} \to \mathbb{N}$, a training set $T \in \mathcal{P}(\mathcal{S}_L)$, a deterministic hash function $h : \mathcal{S}_L \to \mathbb{N}$, and a hyperparameter $k \in \mathbb{N}$ indicating the number of base classifiers which will be used in the ensemble.

At the training time, the algorithm first uses the hash function h to define partitions $P_1, ...,$ $P_k \subseteq T$ of the training set, as follows:

$$P_i := \{\boldsymbol{t} \in T | \, h(\boldsymbol{t}) \equiv i \pmod{k}\}. \tag{7.1}$$

The hash function h can be any deterministic function from $\mathcal{S}_L$ to $\mathbb{N}$: however, it is preferable that the partitions are roughly equal in size. Therefore we should choose an h which maps samples to a domain of integers significantly larger than k, in a way such that $h(.) \pmod{k}$ will be roughly uniform over $[k]$. In practice, for image data, we let $h(\boldsymbol{t})$ be the sum of the pixel values in the image $\boldsymbol{t}$.

Base classifiers are then trained on each partition: we define trained base classifiers $f_i : \mathcal{S} \to \mathbb{N}$ as:

$$f_i(\boldsymbol{x}) := f(P_i, \boldsymbol{x}). \tag{7.2}$$

Finally, at the inference time, we evaluate the input on each base classification, and then count the number of classifiers which return each class:

$$n_c(\boldsymbol{x}) := |\{i \in [k] | f_i(\boldsymbol{x}) = c\}|. \tag{7.3}$$

This lets us define the classifier which returns the consensus output of the ensemble:

$$g_{\textbf{dpa}}(T, \boldsymbol{x}) := \arg\max_c n_c(\boldsymbol{x}). \tag{7.4}$$

When taking the argmax, we break ties deterministically by returning the smaller class index. The resulting robust classifier has the following guarantee:

Theorem 7.1. *For a fixed deterministic base classifier f, hash function h, ensemble size k, training set T, and input $\boldsymbol{x}$, let:*

$$\begin{aligned} c &:= g_{\textbf{dpa}}(T, \boldsymbol{x}) \\ \bar{\rho}(\boldsymbol{x}) &:= \left\lfloor \frac{n_c - \max_{c' \neq c}(n_{c'}(\boldsymbol{x}) + \mathbf{1}_{c'<c})}{2} \right\rfloor. \end{aligned} \tag{7.5}$$

Then, for any poisoned training set U, if $|T \ominus U| \leq \bar{\rho}(\boldsymbol{x})$, then $g_{\textbf{dpa}}(U, \boldsymbol{x}) = c$.

All proofs are presented in Appendix F.1. Note that T and U are *unordered* sets: therefore, in addition to providing certified robustness against insertions or deletions of training data, the robust classifier $g_{\textbf{dpa}}$ is also invariant under re-ordering of the training data, provided that f has this invariance (which is implied, because f maps deterministically from a set; see Section 7.2.2.1 for practical considerations). As mentioned in the chapter introduction, DPA is a deterministic variant of randomized ablation (Chapter 2) adapted to the poisoning domain. Each base classifier ablates most of the training set, retaining only the samples in one partition. However, unlike in randomized ablation, the partitions are deterministic and use disjoint samples, rather than selecting them randomly and independently. In Appendix F.3, we argue that our derandomization has little effect on the certified accuracies, while allowing for exact certificates using finite samples.

We also discuss how this chapter relates to Chapter 6, which proposes a de-randomized ablation technique for a restricted class of sparse evasion attacks (patch adversarial attacks).

7.2.2.1 DPA Practical Implementation Details

One of the advantages of DPA is that we can use deep neural networks for the base classifier f. However, enforcing that the output of a deep neural network is a deterministic function of its training data, and specifically, its training data as an unordered set, requires some care. First, we must remove dependence on the order in which the training samples are read in. To do this, in each partition P_i, we sort the training samples prior to training, taking advantage of the assumption that $\mathcal{S}$ is well-ordered (and therefore $\mathcal{S}_L = \mathcal{S} \times \mathbb{N}$ is also well ordered). In the case of the image data, this is implemented as a lexical sort by pixel values, with the labels concatenated to the samples as an additional value. The training procedure for the network, which is based on standard stochastic gradient descent, must also be made deterministic: in our PyTorch [131] implementation, this can be accomplished by deterministically setting a random seed at the start of training. As discussed in Appendix F.6, we find that it is best for the final classifier accuracy to use different random seeds during training for each partition. This reduces the correlation in output between base classifiers in the ensemble. Thus, in practice, we use the partition index as the random seed (i.e., we train base classifier f_i using random seed i.)

7.2.3 SS-DPA

Semi-Supervised DPA (SS-DPA) is a defense against label-flip attacks. In SS-DPA, the base classifier may be a *semi-supervised* learning algorithm: it can use the entire unlabeled train-

ing dataset, in addition to the labels for a partition. We will therefore define the base classifier to also accept an unlabelled dataset as input: $f : \mathcal{P}(\mathcal{S}) \times \mathcal{P}(\mathcal{S}_L) \times \mathcal{S} \to \mathbb{N}$. Additionally, our method of partitioning the data is modified both to ensure that changing the label of a sample affects only one partition rather than possibly two, and to create a more equal distribution of samples between partitions.

First, we will sort the unlabeled data $\textbf{samples}(T)$:

$$T_{\text{sorted}} \coloneqq \textbf{Sort}(\textbf{samples}(T)). \tag{7.6}$$

For a sample $\boldsymbol{t} \in T$, note that $\textbf{index}(T_{\text{sorted}}, \textbf{sample}(\boldsymbol{t}))$ is invariant under any label-flipping attack to T, and also under permutation of the training data as they are read. We now partition the data based on sorted index:

$$P_i \coloneqq \{\boldsymbol{t} \in T | \, \textbf{index}(T_{\text{sorted}}, \textbf{sample}(\boldsymbol{t})) \equiv i \pmod{k}\}. \tag{7.7}$$

Note that in this partitioning scheme, we no longer need to use a hash function h. Moreover, this scheme creates a more uniform distribution of samples between partitions, compared with the hashing scheme used in DPA. This can lead to improved certificates: see Appendix F.5. This sorting-based partitioning is possible because the unlabeled samples are "clean", so we can rely on their ordering, when sorted, to remain fixed. As in DPA, we train base classifiers on each partition, this time additionally using the entire unlabeled training set:

$$f_i(\boldsymbol{x}) \coloneqq f(\textbf{samples}(T), P_i, \boldsymbol{x}). \tag{7.8}$$

The inference procedure is the same as in the standard DPA:

$$
\begin{aligned}
n_c(\boldsymbol{x}) &:= |\{i \in [k] | f_i(\boldsymbol{x}) = c\}| \\
g_{\textbf{ssdpa}}(T, \boldsymbol{x}) &:= \arg\max_c n_c(\boldsymbol{x})
\end{aligned}
\tag{7.9}
$$

The SS-DPA algorithm provides the following robustness guarantee against label-flipping attacks.[4]

Theorem 7.2. *For a fixed deterministic semi-supervised base classifier f, ensemble size k, training set T (with no repeated samples), and input $\boldsymbol{x}$, let:*

$$
\begin{aligned}
c &:= g_{\textbf{ssdpa}}(T, \boldsymbol{x}), \\
\bar{\rho}(\boldsymbol{x}) &:= \left\lfloor \frac{n_c - \max_{c' \neq c}(n_{c'}(\boldsymbol{x}) + \mathbf{1}_{c' < c})}{2} \right\rfloor .
\end{aligned}
\tag{7.10}
$$

For a poisoned training set U obtained by changing the labels of at most $\bar{\rho}$ samples in T, $g_{\textbf{ssdpa}}(U, \boldsymbol{x}) = c$.

7.2.3.1 Semi-Supervised Learning Methods for SS-DPA

In the standard DPA algorithm, we are able to train each classifier in the ensemble using only a small fraction of the training data; this means that each classifier can be trained relatively quickly: as the number of classifiers increases, the time to train each classifier can decrease (see Table 7.1). However, in a naive implementation of SS-DPA, Equation 7.8 might suggest that training time will scale with k, because each semi-supervised base classifier requires to be

[4]The theorem as stated assumes that there are no repeated unlabeled samples (with different labels) in the training set T. This is a reasonable assumption, and in the label-flipping attack model, the attacker cannot cause this assumption to be broken. Without this assumption, the analysis is more complicated; see Appendix F.7.

trained on the entire training set. Indeed, with many popular and highly effective choices of semi-supervised classification algorithms, such as temporal ensembling [107], ICT [105], Teacher Graphs [106] and generative approaches [108], the main training loop trains on both labeled and unlabeled samples, so we would see the total training time scale linearly with k. In order to avoid this, we instead choose a semi-supervised training method where the unlabeled samples are used *only* to learn semantic features of the data, *before* the labeled samples are introduced: this allows us to use the unlabeled samples only once, and to then share the learned feature representations when training each base classifier. In our experiments, we choose RotNet [109] for experiments on MNIST, and SimCLR [132] for experiments on CIFAR-10 and GTSRB. Both methods learn an unsupervised embedding of the training set, on top of which all classifiers in the ensemble can be learned. Note that [6] also uses SimCLR for CIFAR-10 experiments. As discussed in Section 7.2.2.1, we also sort the data prior to learning (including when learning unsupervised features), and set random seeds, in order to ensure determinism.

7.3 Results

In this section, we present empirical results evaluating the performance of proposed methods, DPA and SS-DPA, against poison attacks on MNIST, CIFAR-10, and GTSRB datasets. As discussed in Section 7.2.3.1, we use the RotNet architecture [109] for SS-DPA's semi-supervised learning on MNIST. Conveniently, the RotNet architecture is structured such that the feature extracting layers, combined with the final classification layers, together make up the Network-In-Network (NiN) architecture for the supervised classification [133]. Therefore, on MNIST, we use NiN for DPA's supervised training, and RotNet for SS-DPA's semi-supervised training.

	Training set size	Number of Partitions k	Median Certified Robustness	Clean Accuracy	Base Classifier Accuracy	Training time per Partition
MNIST, DPA	60000	1200 3000	448 509	95.85% 93.36%	76.97% 49.54%	0.33 min 0.27 min
MNIST, SS-DPA	60000	1200 3000	485 645	95.62% 93.90%	80.77% 57.65%	0.15 min 0.16 min
CIFAR, DPA	50000	50 250 1000	9 5 N/A	70.16% 55.65% 44.52%	56.39% 35.17% 23.20%	1.49 min 0.58 min 0.30 min
CIFAR, SS-DPA	50000	50 250 1000	25 124 392	90.89% 90.33% 89.02%	89.06% 86.25% 75.83%	0.94 min 0.43 min 0.33 min
GTSRB, DPA	39209	50 100	20 4	89.20% 55.90%	73.94% 35.64%	2.64 min 1.60 min
GTSRB, SS-DPA	39209	50 100 200 400	25 50 99 176	97.09% 96.76% 96.34% 95.80%	96.35% 94.96% 91.54% 83.60%	2.73 min 1.56 min 1.23 min 0.78 min

Table 7.1: Summary statistics for DPA and SS-DPA algorithms on MNIST, CIFAR, and GTSRB. Median Certified Robustness is the attack magnitude (symmetric difference for DPA, label flips for SS-DPA) at which certified accuracy is 50%. Training times are on a single GPU; note that many partitions can be trained in parallel. Note we observe some constant overhead time for training each classifier, so on MNIST, where the training time per image is small, k has little effect on the training time. For SS-DPA, training times do not include the time to train the unsupervised feature embedding (which must only be done once).

We use training parameters, for both the DPA (NiN) and SS-DPA (RotNet), directly from [109], with a slight modification: we eliminate horizontal flips in data augmentation, because horizontal alignment is semantically meaningful for digits.[5] On CIFAR-10 and GTSRB, we also use NiN (with full data augmentation for CIFAR-10, and without horizontal flips for GTSRB) for DPA experiments. For semi-supervised learning in SS-DPA, we use SimCLR [132] for both datasets,

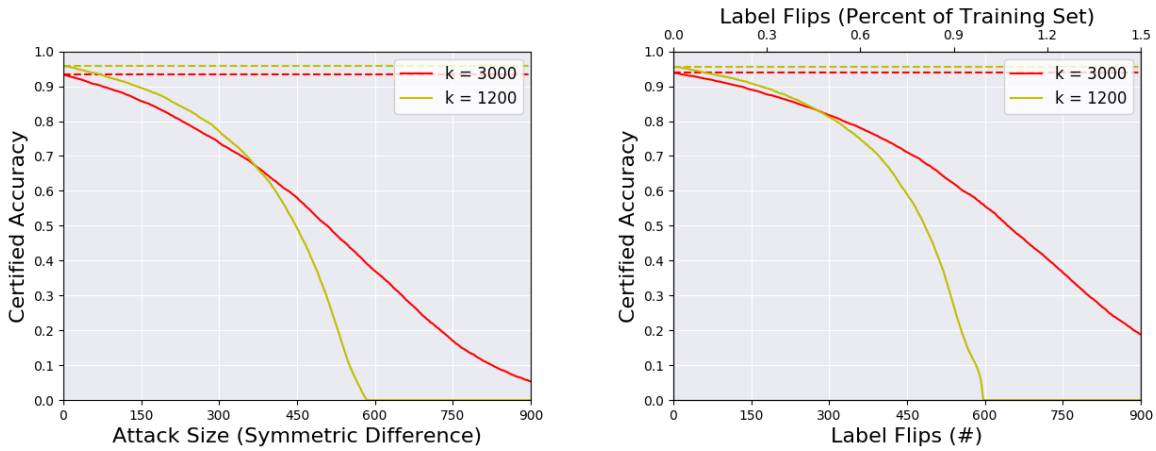
[5]In addition to the de-randomization changes mentioned in Section 7.2.2.1, we made one modification to the NiN 'baseline' for supervised learning: the baseline implementation in [109], even when trained on a small subset of the training data, uses normalization constants derived from the entire training set. This is a (minor) error in [109] that we correct by calculating normalization constants on each subset.

as [6] does on CIFAR-10. SimCLR hyperparameters are provided in Appendix F.9, and additional details about processing the GTSRB dataset are provided in Appendix F.10. Note that for SimCLR we use linear classifiers as the final, supervised classifiers for each partition.

Results are presented in Figures 7.2, 7.3, 7.4, and are summarized in Table 7.1. Our metric, Certified Accuracy as a function of attack magnitude (symmetric-difference or label-flips), refers to the fraction of samples which are both correctly classified and are certified as robust to attacks of that magnitude. Note that *different* poisoning perturbations, which poison different sets of training samples, may be required to poison *each* test sample; i.e. we assume the attacker can use the attack budget separately for each test sample. Table 7.1 also reports Median Certified Robustness, the attack magnitude to which at least 50% of the test set is provably robust.

Our SS-DPA method substantially outperforms the existing certificate [6] on label-flipping attacks: in median, 392 label flips on CIFAR-10, versus 175; 645 label flips on MNIST, versus < 200. With DPA, we are also able to certify at least half of MNIST images to attacks of over 500 poisoning insertions or deletions, and can certify at least half of CIFAR-10 images to 9 poisoning insertions or deletions. On GTSRB, we can certify over half of images to 20 poisoning insertions or deletions, or 176 label flips. Note that this represents a substantially larger fraction of each class (each class has < 1000 training images on average) compared to certificates on CIFAR-10 (5000 training images per class). See Appendix F.8 for additional experiments on GTSRB.

The hyperparameter k controls the number of classifiers in the ensemble: because each sample is used in training exactly one classifier, the average number of samples used to train each classifier is inversely proportional to k. Therefore, we observe that the base classifier accuracy (and therefore also the final ensemble classifier accuracy) decreases as k is increased; see Table 7.1. However, because the certificates described in Theorems 7.1 and 7.2 depend directly on the

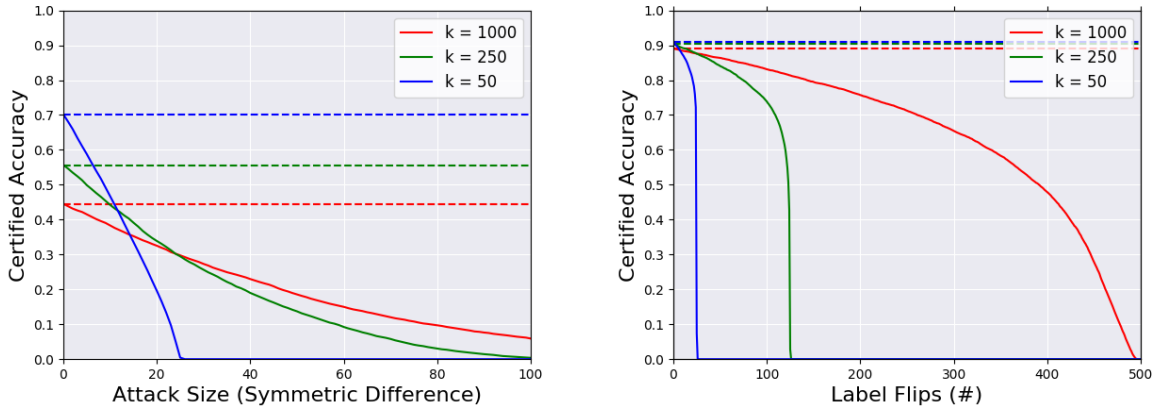


(a) DPA (General poisoning attacks) (b) SS-DPA (Label-flipping poisoning attacks)

Figure 7.2: Certified Accuracy to poisoning attacks on MNIST, using (a) DPA to certify against general poisoning attacks, and (b) SS-DPA to certify against label-flipping attacks. Dashed lines show the clean accuracies of each model.

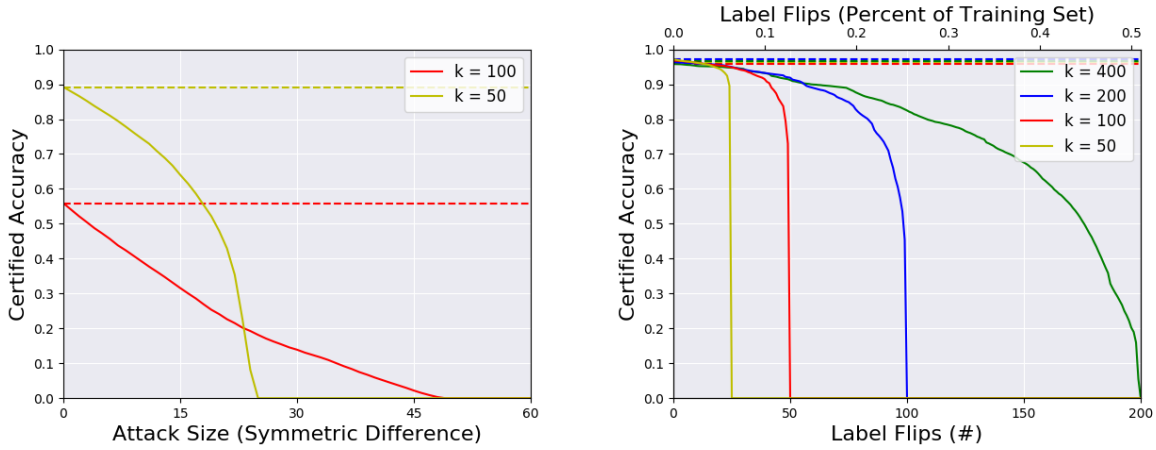
gap in the *number* of classifiers in the ensemble which output the top and runner-up classes, larger numbers of classifiers are necessary to achieve large certificates. In fact, using k classifiers, the largest certified robustness possible is $k/2$. Thus, we see in Figures 7.2, 7.3 and 7.4 that larger values of k tend to produce larger robustness certificates. Therefore k controls a trade-off between robustness and accuracy.

[6] also reports robustness certificates against label-flipping attacks on binary MNIST classification, with classes 1 and 7. [6] reports clean-accuracy of 94.5% and certified accuracies for attack magnitudes up to 2000 label flips (out of 13007), with best certified accuracy less than 70%. By contrast, using a specialized form of SS-DPA, we are able to achieve clean accuracy of 95.5%, with every correctly-classified image certifiably robust up to 5952 label flips (i.e. certified accuracy is also 95.5% at 5952 label flips.) See Appendix F.2 for discussion.



(a) DPA (General poisoning attacks) (b) SS-DPA (Label-flipping poisoning attacks)

Figure 7.3: Certified Accuracy to poisoning attacks on CIFAR, using (a) DPA to certify against general poisoning attacks, and (b) SS-DPA to certify against label-flipping attacks.



(a) DPA (General poisoning attacks) (b) SS-DPA (Label-flipping poisoning attacks)

Figure 7.4: Certified Accuracy to poisoning attacks on GTSRB, using (a) DPA to certify against general poisoning attacks, and (b) SS-DPA to certify against label-flipping attacks.

Part III

Test-time Adaptability as Robustness in Reinforcement Learning

Chapter 8: Goal-Conditioned Q-Learning as Knowledge Distillation[1]

In recent years, many works have focused on applying deep reinforcement learning to goal-conditioned tasks, through approaches such as goal relabeling [50, 135, 136, 137] and automatic curriculum generation [47, 48, 49]. In this work, we focus on model-free off-policy goal-conditioned RL, and present a novel technique for improving performance in this setting. Our approach relies on a connection between the standard Bellman update used in off-policy reinforcement learning in a goal-conditioned setting, and *knowledge distillation*, the task of training a student network to model the same function as a (generally more complex) teacher network.[2] In brief, the Bellman update can be viewed as an instance of (conditional, stochastic) knowledge distillation, where the current Q-value estimate is the student, the target Q-value network (averaged over transitions) is the teacher, and the independent variable is the *goal* that the agent is attempting to reach. We use this insight to develop an novel off-policy algorithm that in some instances has improved performance over baselines for goal-conditioned tasks. Our main contributions are as follows:

1. We propose **ReenGAGE**, a novel technique for goal-conditioned off-policy reinforcement learning, and evaluate its performance.

[1] A form of chapter has been published in AAAI 2023 [134].

[2] Some works use the term "knowledge distillation" to refer to the particular method for this task proposed by [138], while others, such as [139] use it to refer to the task in general; we use the latter definition.

2. We propose **Multi-ReenGAGE**, a variant of ReenGAGE well-suited for goal-conditioned environments with *many simultaneous sparse goals.*

3. We provide theoretical justification for ReenGAGE by showing that it is in some cases asymptotically more efficient, in terms of the total number of replay buffer transitions required to learn an optimal policy, than standard off-policy algorithms such as DDPG.

In most of this work, we focus on continuous action control problems; we extend our method to discrete action spaces in the appendix. Note that while we mostly focus on using ReenGAGE on top of HER [50] and DDPG [140] in this work, it can be easily applied alongside any goal-relabeling scheme or automated curriculum, and can be adapted for other off-policy algorithms such as SAC [141] or TD3 [142]. In particular, we include an application to SAC in the appendix.

8.1 Preliminaries and Notation

We consider control problems defined by goal-conditioned MDPs $(S, A, G, \mathcal{T}, R)$, where S, A, and G denote sets of states, actions, and goals, respectively, G and A are assumed to be continuous spaces, $\mathcal{T} \in S \times A \to \mathcal{P}(S)$ is a stochastic transition function, and $R \in S \times G \to \mathbb{R}$ is a reward function. At every step, starting at state $s \in S$, an agent chooses an action $a \in A$. The system then transitions to $s' \sim \mathcal{T}(s, a)$, and the agent receives the reward $R(s', g)$.

For now, we assume that the reward function $R(s', g)$ is *known a priori* to the learning algorithm (while the transition function is not): this just means that we know how to interpret the objective which we are trying to achieve. Note that existing goal relabeling techniques, such as HER [50], implicitly make this assumption, as it is necessary to compute the rewards of relabeled transitions. We will discuss cases where this assumption can be relaxed in later sections.

We consider both *shaped* rewards, in which $R(s', g)$ is continuous and differentiable everywhere, as well as *sparse* rewards, where $R(s', g)$ is not assumed to be differentiable, but maintains some constant value c_{low} (e.g., c_{low} = -1 or 0) with $\nabla_g R(s', g) = \mathbf{0}$ for a substantial fraction of inputs. (There may be some boundary points where $R(s', g) = c_{\text{low}}$ but $\nabla_g R(s', g)$ is not defined or $\nabla_g R(s', g) \neq \mathbf{0}$, but in theoretical discussion we will assume these are of measure zero).

The objective of goal-conditioned RL is to find a policy $\pi \in S \times G \to A$ such that the discounted future reward:

$$r = \sum_{t=0}^{\infty} \gamma^t R(s_{t+1}, g) \tag{8.1}$$

is maximized in expectation. One common approach is to find the policy π and Q-function $Q \in S \times A \times G \to \mathbb{R}$ that solve the Bellman equation for Q-learning [143], conditioned on a goal g:

$$\forall s, a, g, \;\; Q(s, a, g) = \mathop{\mathbb{E}}_{s' \sim \mathcal{T}(s,a)} \left[R(s', g) + \gamma Q(s', \pi(s', g), g) \right] \tag{8.2}$$

$$\forall s, g, \;\; \pi(s, g) = \arg\max_a Q(s, a, g). \tag{8.3}$$

If functions π and Q satisfy these, then π is guaranteed to be an optimal policy. In practice, off-policy RL techniques, notably DDPG [140] can be used to solve for these functions iteratively by drawing tuples (s, a, s', g) from a replay buffer:

$$\mathcal{L}_{\text{critic}} = \mathop{\mathbb{E}}_{(s,a,s',g) \sim \text{Buffer}} \left[\mathcal{L}_{\text{mse}} \Big[Q_\theta(s, a, g), R(s', g) + \gamma Q_{\theta'}(s', \pi_{\phi'}(s', g), g) \Big] \right] \tag{8.4}$$

$$\mathcal{L}_{\text{actor}} = \mathop{\mathbb{E}}_{(s,g)\sim\text{Buffer}} -Q_\theta(s, \pi_\phi(s,g), g), \tag{8.5}$$

where θ and ϕ are the *current* critic and actor parameters, and θ' and ϕ' are *target* parameters, which are periodically updated to more closely match the current estimates. Note that Equation 8.2 should ideally hold for *all* (s,a,g): therefore the distribution of (s,a,g) in the replay buffer does not need to precisely follow any particular distribution, assuming sufficient visitation of possible tuples.[3] The only necessary constraint on the buffer distribution is that the marginal distribution of s' matches the transition function:

$$\forall s, a, g, \quad \Pr_{\text{Buffer}}[s'|s,a,g] \approx \Pr_{\mathcal{T}}[s'|s,a], \tag{8.6}$$

so that the relation in Equation 2 is respected. This means that the goal which is included in the buffer need not necessarily reflect a "true" historical experience of the agent during training, but can instead be relabeled to enhance training. [50, 135, 136, 137]. Interestingly, [145] shows that HER [50], a popular relabeling technique, actually *does not* respect Equation 8.6 when the transition function is nondeterministic, and therefore may exhibit "hindsight bias."

8.2 Proposed Method

From Equation 8.2, we can take the gradient with respect to g:

[3] In practice, replay buffers which better match the behavioral distribution result in better training, due to sources of "extrapolation error", including incomplete visitation and model inductive bias; see [144].

$$\nabla_g Q(s,a,g) =$$

$$\nabla_g \mathop{\mathbb{E}}_{s' \sim \mathcal{T}(s,a)} \left[R(s',g) + \gamma Q(s', \pi(s',g), g) \right] = \tag{8.7}$$

$$\mathop{\mathbb{E}}_{s' \sim \mathcal{T}(s,a)} \left[\nabla_g R(s',g) + \gamma \nabla_g Q(s', \pi(s',g), g) \right].$$

Because the gradient of the Q-value function is equal to the *expectation of the gradient* of the sum of the reward and the next-step Q-value, this suggests that we can augment the standard DDPG critic loss with a gradient term, which estimates this expected gradient using the replay buffer samples:

$$\begin{aligned} \mathcal{L}_{\textbf{ReenGAGE}} = \mathop{\mathbb{E}}_{(s,a,s',g) \sim \text{Buffer}} \Bigg[& \mathcal{L}_{\text{mse}} \Big[Q_\theta(s,a,g), R(s',g) + \gamma Q_{\theta'}(s', \pi_{\phi'}(s',g), g) \Big] \\ & + \alpha \mathcal{L}_{\text{mse}} \Big[\nabla_g Q_\theta(s,a,g), \nabla_g R(s',g) + \gamma \nabla_g Q_{\theta'}(s', \pi_{\phi'}(s',g), g) \Big] \Bigg] \end{aligned} \tag{8.8}$$

where α is a constant hyperparameter. Note that the second MSE term is applied to a vector: thus we are fitting $\nabla_g Q_\theta(s_0, a_0, g)$ in all $\dim(g)$ dimensions. This allows more information to flow from the target function to the current Q-function (a $\dim(g)-$vector instead of a scalar), and may therefore improve training. We call our method **Re**inforcem**en**t learning with **G**radient **A**ttention for **G**oal-seeking **E**fficiently, or **ReenGAGE**.

In the case of shaped rewards, we can use this loss function directly. In the case of sparse rewards, $\nabla_g R(s',g)$ is not necessarily defined or available. However, it is also zero for a substantial fraction of inputs, and, if $R(s',g) = c_{\text{low}}$, then $\nabla_g R(s',g) = \mathbf{0}$ with high probability. Therefore, we use the gradient loss term only when training on tuples where $R(s',g) = c_{\text{low}}$, and

assume $\nabla_g R(s', g) = \mathbf{0}$:

$$\mathcal{L}^{(\text{sparse})}_{\textbf{ReenGAGE}} = \mathop{\mathbb{E}}_{(s,a,s')} \Bigg[\mathcal{L}_{\text{mse}} \Big[Q_\theta(s, a, g), R(s', g) + \gamma Q_{\theta'}(s', \pi_{\phi'}(s', g), g) \Big] \\ + \mathbf{1}_{R(s',g)=c_{\text{low}}} \alpha \mathcal{L}_{\text{mse}} \Big[\nabla_g Q_\theta(s, a, g), \gamma \nabla_g Q_{\theta'}(s', \pi_{\phi'}(s', g), g) \Big] \Bigg]. \tag{8.9}$$

In this sparse case, if ReenGAGE is used alone (i.e., without goal relabeling), then the reward function R does not need to be known explicitly *a priori*. Instead, the observed values of the rewards $R(s', g)$ from the training rollouts may be used.

Note that ReenGAGE can only be used in goal-conditioned reinforcement learning problems: in particular, we cannot use gradients with respect to states or actions in a way similar to Equation 8.7, because, unlike in Equation 8.7:

$$\nabla_{s,a} \mathop{\mathbb{E}}_{s' \sim \mathcal{T}(s,a)} [(\cdot)] \neq \mathop{\mathbb{E}}_{s' \sim \mathcal{T}(s,a)} [\nabla_{s,a} (\cdot)] \tag{8.10}$$

because the sampling distribution depends on s and a.

8.2.1 Connection to Knowledge Distillation

We can view Equation 8.4 as the loss function of a regression problem, fitting Q_θ to the target. We treat g as the independent variable, and s and a as parameters:

$$\forall g, \quad Q_\theta(g; s, a) \coloneqq \text{Targ.}(g; s, a) \text{ where} \\ \text{Targ.}(g; s, a) = \mathop{\mathbb{E}}_{s' \sim \mathcal{T}(s,a)} R(s', g) + \gamma Q_{\theta'}(s', \pi_{\phi'}(s', g), g) \tag{8.11}$$

This can be framed as a knowledge distillation problem: we can view Targ.$(g; s, a)$ as a (difficult to compute) *teacher* function, and we are trying to fit the network $Q_\theta(g; s, a)$ to represent the same function of g. Note however that conventional "knowledge distillation" [138], which matches the logits of one network to another for classification problems in order to provide richer supervision than simply matching the class label output, cannot be applied here because the output is scalar. However, [57] proposes *Gradient-based Attention Transfer* which instead matches the *gradients* of the student to the teacher using a regularization term. Applied to our Q function and target, this is:

$$\begin{aligned}\mathcal{L}_{\text{GAT}} =& \mathcal{L}_{\text{MSE}}(Q_\theta(g; s, a), \text{Targ.}(g; s, a)) \\ &+\alpha \| \nabla_g Q_\theta(g; s, a) - \nabla_g \text{Targ.}(g; s, a) \|_2^2,\end{aligned} \tag{8.12}$$

which is in fact the ReenGAGE loss function. Therefore we can think of ReenGAGE as applying knowledge distillation (specifically Gradient-based Attention Transfer) to the Q-value update.[4] See Figure 8.1 for an illustration. While the computation of the loss function gradient is somewhat more complex here than in standard training, involving mixed partial derivatives, [57] notes that it can still be performed efficiently using modern automatic differentiation packages; in fact, this "double backpropagation" should only scale the computation time by a constant factor [146]. Further discussion of this and empirical runtime comparisons are provided in the appendix. [57] also propose Activation-based Attention Transfer, which transfers intermediate layer activations from teacher to student network rather than gradients; in fact, they report better performance

[4] [57] actually uses the ℓ_2 distance between the gradients as the regularization term, rather than its square. However, because our gradient estimate is stochastic (in particular, we are using samples of $s' \sim \mathcal{T}(s, a)$ rather than the expectation), we instead use the mean squared error, so that the current Q gradients will converge to the population mean. [57] notes that their particular choice of the ℓ_2 norm is arbitrary: other metrics should work.

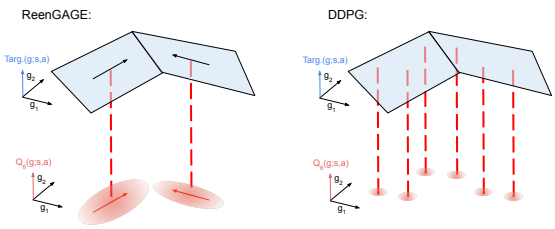


Figure 8.1: Illustration comparing the information flow from the Q-value target function to the current Q-function in ReenGAGE, compared to standard DDPG. In ReenGAGE, for each (s, a, s', g) tuple, the gradient with respect to the goal is used as supervision, while in standard DDPG, only point values are used. Note that in stochastic environments, each tuple only provides a stochastic estimate of the target gradient (in ReenGAGE) or target point value (in DDPG).

using this method than the gradient method. However, this is not applicable in our case. Firstly, in the dense reward case, we cannot model the reward function in this way. Secondly, unlike the gradient operator, activations are nonlinear: so, even in the sparse case, we cannot assume that the activations of a "converged" Q-network perfectly modeling the expected target will be the equal to the expected activations of the target network (i.e., there is no activation equivalent to Equation 8.7.) See [147] for a review of sources of auxiliary network information that can be used for knowledge distillation.

8.3 Toy Example Experiments

We first apply ReenGAGE to a simple sparse-reward environment, which we call **ContinuousSeek**. This task is a continuous variant of the discrete "Bit-Flipping" environment proposed in [50]. In our proposed task, the objective is to navigate from an initial state in d-dimensional space to a desired goal state, by, at each step, adding an ℓ_∞-bounded vector to the current state. Formally:

- $s, g \in [-D, D]^d$

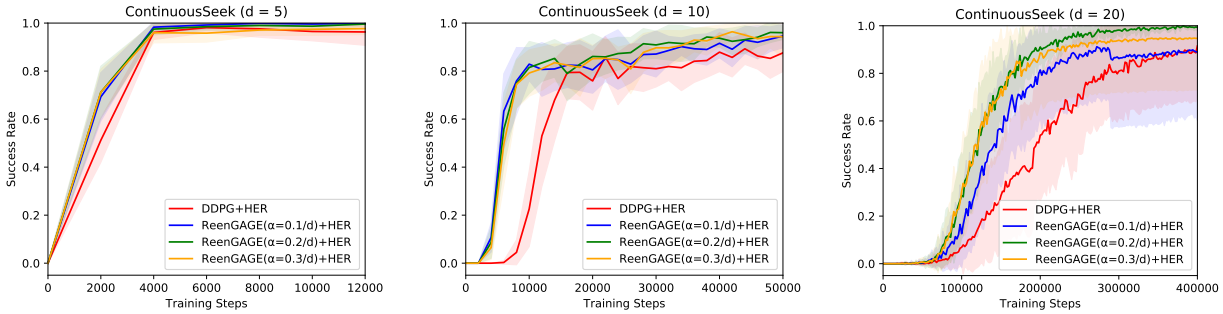


Figure 8.2: ContinuousSeek results. Lines show the mean and standard deviation over 20 random seeds (kept the same for all experiments.) The Y-axis represents the success rate, defined as the fraction of test episodes for which the goal is ever reached.

- $a \in [-1, 1]^d$
- $\mathcal{T}(s, a) = s + a$ (clipped into $[-D, D]^d$)
- $R(s, g) = -1 + \mathbf{1}_{\|s-g\|_\infty \leq \epsilon}$
- initial state $s_0 = \mathbf{0}$.

In our experiments, we use $D = 5$, $\epsilon = 0.1$, and we run for 10 steps per episode. We test with $d = 5, 10$ and 20. The chance that a random state achieves the goal is approximately $\frac{1}{50^d}$, so this is an extremely sparse reward problem (as sparse as "Bit-Flipping" with $5.6 \times d$ bits). As a baseline, we use DDPG with HER.

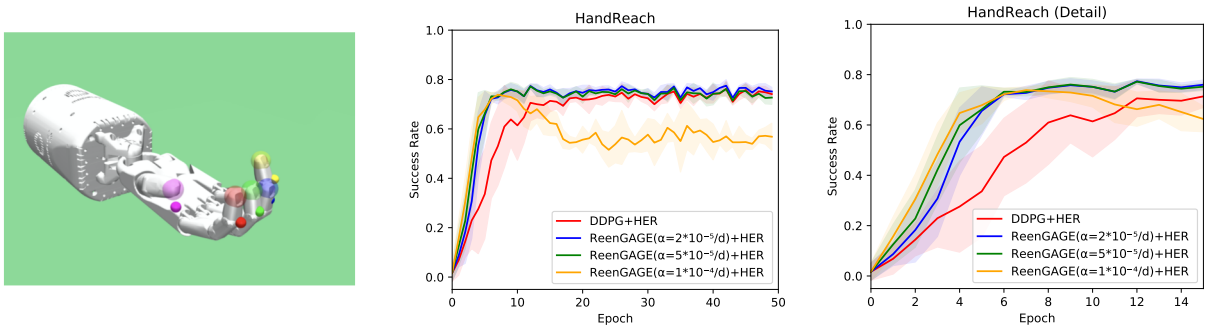
See Figure 8.2 for results. For the baseline and each value of α, we performed a grid search over learning rates $\{0.00025, 0.0005, 0.001, 0.0015\}$ and batch sizes $\{128, 256, 512\}$; the curves shown represent the "best" hyperparameter settings for each α, defined as maximizing the area under the curves. See appendix for results for all hyperparameter settings. We studied the learning rate specifically to ensure that the ReenGAGE regularization term is not simply "scaling up" the loss function with similar gradient updates. Other hyperparameters were kept fixed and are listed in the appendix. We see that ReenGAGE clearly improves over the baseline

for larger-dimensionality goals (d = 10 and d = 20): this shows that ReenGAGE can improve the performance of DDPG in such high-dimensional goal settings. See the appendix for a similar experiment with SAC as the base off-policy learning algorithm instead of DDPG.

8.4 Robotics Experiments

We tested our method on **HandReach**, the environment from the OpenAI Gym Robotics suite [148] with the highest-dimensional goal space (d = 15). In this sparse-reward environment, the agent controls a simulated robotic hand with 20-dimensional actions controlling the hand's joints; the goal is to move all of the fingertips to the specified 3-dimensional positions. As a baseline, we use the released DDPG+HER code from [148], with all hyperparameters as originally presented, and only modify the critic loss term. Results are presented in Figure 8.3. In this environment, we see that ReenGAGE greatly speeds up convergence compared to the baseline. However, at a high value of α, the success rate declines after first converging. This shows that ReenGAGE may cause some instability if the gradient loss term is too large, and that tuning the coefficient α is necessary (see also the Limitations section below).

We also tried our method on the **HandManipulateBlock** environment from the same paper; however, in this lower-dimensional goal environment (d = 7) ReenGAGE was not shown to improve performance. This is compatible with our observation from the ContinuousSeek environment that ReenGAGE leads to greater improvements for higher-dimensional tasks, as the dimensionality of the additional goal-gradient information that ReenGAGE propagates increases. Results are provided in the appendix.



(a) (b) (c)

Figure 8.3: HandReach results. (a) Rendering of the HandReach simulation environment. Figure taken from [148]. (b) Performance of ReenGAGE on HandReach, compared to the baseline from [148]. Lines represent mean and standard deviation for the same set of 5 seeds. The X-axis is the number of training epochs, as defined in [148], while the Y-axis is the success rate, defined by [148] as the fraction of test episodes where the *final* state satisfies the goal. (c) Detailed view of (b), showing the epochs before convergence, where the advantage of ReenGAGE is most clear.

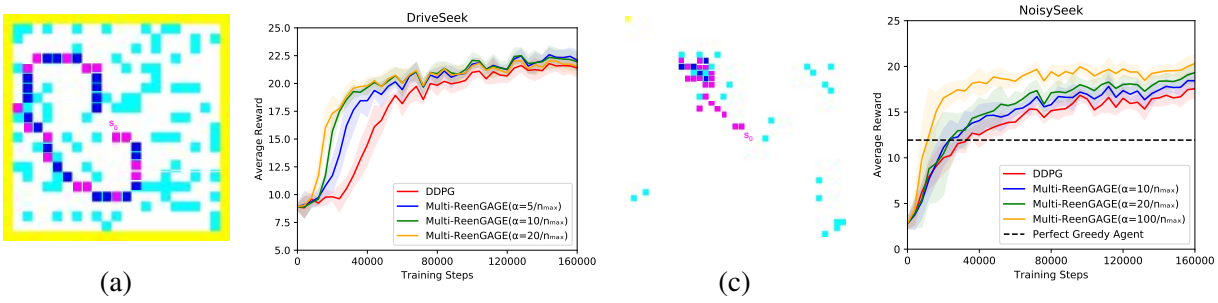
8.5 Multi-ReenGAGE: ReenGAGE for Multiple Simultaneous Goals

In this section, we propose a variant of ReenGAGE for a specific class of RL environments: environments where the agent is rewarded for achieving any goal in a large set of arbitrary sparse goals, all of which are specified at test time. Formally, we consider goals in the form $g = \{g_1, ...g_n\}$, where n may vary but $n \leq n_{\max}$. We consider $\{0, 1\}$ rewards, where the reward function takes the form:

$$R(s', g) = \begin{cases} 1 & \text{if } \bigvee_{g_i \in g} \left(R_{\text{item}}(s', g_i) = 1\right) \\ 0 & \text{otherwise} \end{cases}. \tag{8.13}$$

In our experiments, we only consider cases where the goals are mutually exclusive, so this is equivalent to:

$$R(s', g) = \sum_{g_i \in g} R_{\text{item}}(s', g_i). \tag{8.14}$$



(a) (b) (c) (d)

Figure 8.4: Multi-ReenGAGE results. (a) and (c): Illustrations of DriveSeek and NoisySeek environments, respectively: cyan points show goals that were never reached, blue points show goals that were reached, and magenta points show (rounded) non-goal states that were reached. In (c), we see a NoisyReach agent (correctly) avoiding the trap of going to the nearby isolated points in favor of seeking the larger cluster. (b) and (d): Results for DriveSeek and NoisySeek, respectively. We see that Multi-ReenGAGE substantially improves over standard DDPG for both tasks. Lines are an average of (the same) 5 random seeds. For NoisySeek, we also show the performance of a perfect "greedy" agent, which simply goes towards the nearest individual goal. For NoisySeek, evaluations with more values of α are included in the appendix. Note that for both experiments, the agent takes as input a list of goal coordinates, rather than an image: the agents do not use convolutional layers to interpret the goals. (On DriveSeek, where the coordinates are bounded, we attempted learning from images as well; ReenGAGE still outperformed DDPG, but overall performance was worse for both – this experiment is presented in the appendix.)

We assume that either: (i) the function R_{item} is known *a priori* to the agent, or (ii) the item rewards $R_{\text{item}}(s', g_i)$ are observed separately for each goal g_i at each time step during training. This scenario presents several challenges. Firstly, many goal relabeling strategies cannot be directly applied here: strategies such as HER [50] assume that achieved states can be *projected down* into the space of goals. In this case, the space of goals is *much larger* than the space of possible states, so this assumption is broken. Secondly, we suggest that standard Q-learning is somewhat unsuited to this kind of problem, because it loses information about *which* goal led to a reward. For instance, if there are 100 goals g_i, and a reward is received for a certain state s', there is no direct indication of which goal was satisfied. This means that a very large number of episodes may need to be run in order to learn the effect of *each individual* goal on the reward.

We now describe our approach. For concreteness, we will assume that the agent uses an

architecture based on DeepSets [149] to process the goal set input (although we believe our technique can likely be adapted to using more complex neural set architectures, such as Set Transformer [150]). Concretely, this means that our Q-function takes the form:

$$Q_\theta(s,a,g) := Q^{\text{head}}_{\theta^{\text{h.}}}(s,a,\sum_{g_i \in g}[Q^{\text{encoder}}_{\theta^{\text{e.}}}(s,g_i)]) \tag{8.15}$$

and the policy has a similar architecture. Note that $Q^{\text{encoder}}_{\theta^{\text{e.}}}$ outputs a vector-valued embedding for a given goal g_i. From this baseline, introduce a set of scalar *gate variables* b_i:

$$Q_\theta(s,a,g) := Q^{\text{head}}_{\theta^{\text{h.}}}(s,a,\sum_{i=1}^{n}[b_i Q^{\text{encoder}}_{\theta^{\text{e.}}}(s,g_i)]). \tag{8.16}$$

Each gate b_i is set to 1. However, if b_i were zero, this would be equivalent to the goal g_i being absent from the set g. We then treat the gate variables as *differentiable*. If a certain goal g_i contributes to the Q function (i.e., if it is likely to be satisfied), then we expect $Q_\theta(s,a,g)$ to be *highly sensitive* to b_i; in other words, we expect $\frac{\partial Q_\theta}{\partial b_i}$ to be large. Then $\nabla_b Q$ represents the importance of each goal to the Q-value function. Our key insight is that we can use a ReenGAGE-style loss to transfer $\nabla_b Q$ from target to current Q-value estimate, therefore preserving attention on the relevant goals.

However, this requires us to have a value for $\nabla_b R(s',g)$. Note that the reward can be written as:

$$R(s',g) = \sum_{g_i \in g} b_i R_{\text{item}}(s',g_i). \tag{8.17}$$

Setting $b_i = 0$ is again like g_i being absent. Interpolating:

$$\frac{\partial R}{\partial b_i} := R_{\text{item}}(s', g_i) \tag{8.18}$$

which gives us a "ground-truth" reward gradient we can compute. This yields the following loss function:

$$\begin{aligned}\mathcal{L}_{\textbf{Multi-ReenGAGE}} = \mathcal{L}_{\textbf{DDPG-Critic}} + \alpha\mathcal{L}_{\text{mse}}\Big[&\nabla_b Q_\theta(s, a, g), \\ &R_{\text{item}}(s', g) + \gamma\nabla_b Q_{\theta'}(s', \pi_{\phi'}(s', g), g)\Big]\end{aligned} \tag{8.19}$$

where $[R_{\text{item}}(s', g)]_i := R_{\text{item}}(s', g_i)$. In practice, we make two modifications to this algorithm. First, we use b_i^2 as the gate rather than b_i.[5] While algebraically this should do nothing but multiply the gradient loss term by 4, it is important for vectorized implementation; see the appendix for details.

Second, we share the encoder Q^{encoder} between the Q-function and the policy π. This is so the policy does not have to learn to interpret the goal set "from scratch" and is empirically important (see ablation study in the appendix). We train the encoder only during critic training.

8.5.1 Experiments

We test Multi-ReenGAGE on two environments: **DriveSeek** and **NoisySeek**. Both environments are constructed such that a "greedy" strategy of simply going to the closest individual goal is not optimal, so the entire goal set must be considered. We describe the environments informally here and provide additional detail in the appendix.

[5] And use $2R_{\text{item}}(s', g)$ instead of $R_{\text{item}}(s', g)$.

DriveSeek is a deterministic environment, where the continuous position $s_{\text{pos.}} \in [-10, 10]^2$ always moves with constant ℓ_2 speed 1, in a direction determined by a velocity vector $s_{\text{vel.}}$ on the unit circle. At each step, the agent takes a 1-dimensional action $a \in [-0.5, 0.5]$, which represents *angular acceleration*: it specifies an angle in radians which is added to the angle of the velocity vector. At the edges of the space, the state position wraps around to the opposite edge.

The objective is to reach any of up to $n_{\text{max}} = 200$ goals. The goals all lie on integer coordinates in $[-10, 10]^2$, and the agent receives a reward if its coordinates round to a goal. In addition to $s_{\text{pos.}}$ and $s_{\text{vel.}}$, the agent receives an observation of its current rounded position. The agent also receives as input the current list of goal coordinates. Note that the agent cannot simply stay at a single goal, or take an arbitrary path between goals: it is constrained to making wide turns. Therefore all goals must be considered in planning an optimal trajectory.

NoisySeek is a randomized environment. In it, $s \in \mathbb{R}^2$, $a \in \mathbb{R}^2$, with $\|a\|_2 \leq 1$, and the transition function is defined as $\mathcal{T}(s, a) \sim \mathcal{N}(s + a, I)$. In other words, the agent moves through space at a capped speed, and noise is constantly added to the position. The goals are defined as integer coordinates in a similar manner to in DriveSeek, but without a box constraint. Additionally, the goal distribution is such that goals tend to be clustered together. Note that a "greedy" agent that simply goes to the nearest goal is suboptimal, because the probability of consistently reaching that one goal is low: it is better to seek clusters.

Results are presented in Figure 8.4. We see that Multi-ReenGAGE substantially outperforms the baseline of DDPG on both environments.

8.6 Theoretical Properties

8.6.1 Bias

In the Preliminaries section, we discuss that goal relabeling strategies can exhibit bias if Equation 8.6 is not respected. In the dense reward case, our method does not cause bias of this sort (although such bias may be present if our method is combined with a relabeling strategy.) However, in the sparse reward case, if the transitions are nondeterministic, our method may cause a similar bias. In particular, note that, in the sparse case, Equation 8.9 effectively trains the gradient of the Q-value function to match the following target:

$$\begin{aligned}
&\nabla_g Q_\theta(s,a,g) := \\
&\mathop{\mathbb{E}}_{s'\sim\mathcal{T}(s,a)}\left[\gamma \nabla_g Q_{\theta'}(s', \pi_{\phi'}(s',g),g) | R(s',g) = c_{\text{low}}\right] \approx \\
&\nabla_g \mathop{\mathbb{E}}_{s'\sim\mathcal{T}(s,a)}\left[R(s',g) + \gamma Q_{\theta'}(s', \pi_{\phi'}(s',g),g) | R(s',g) = c_{\text{low}}\right]
\end{aligned} \tag{8.20}$$

where the last line holds exactly if the "boundary" points where $R(s',g) = c_{\text{low}}$ but $\nabla_g R(s',g) \neq \mathbf{0}$ are of measure zero (and the derivative is defined at all such points).

Equation 8.20 shows us that, in the sparse case, our method trains the gradient of the Q-function to match an expected target gradient where the expectation is taken over a biased distribution: if

$$\Pr_{\mathcal{T}}[s'|s,a; R(s',g) = c_{\text{low}}] \neq \Pr_{\mathcal{T}}[s'|s,a], \tag{8.21}$$

then this will cause bias in our target gradient estimate, in a similar manner to the hindsight bias

of HER described by [145].

Note that this is only an issue in nondeterministic environments: in deterministic environments, for a given (s, a, g), either $R(s', g)$ is always $\neq c_{\text{low}}$, in which case the gradient term is never involved in training, or $R(s', g)$ is always c_{low}, in which case

$$\Pr_{\mathcal{T}}[s'|s, a; R(s', g) = c_{\text{low}}] = \Pr_{\mathcal{T}}[s'|s, a] = 1. \tag{8.22}$$

8.6.2 Learning Efficiency

In this section, we provide examples of classes of environments for which our method will result in provably more efficient learning than standard DDPG-style updates. We treat both the dense reward case (in which we have access to the gradient of the reward function) and the sparse reward case (in which we do not).

8.6.2.1 Dense Reward Example

Consider the class of simple, deterministic environments described as follows:

- $g, s, a \in \mathbb{R}^d$; $\|a\|_2 \leq 1$
- $R(s, g) = g^T s$
- $\mathcal{T}(s, a) = s + Ua$, where $U \in \mathbb{R}^{d \times d}$ is an unknown orthogonal (rotation) matrix.

This environment class is illustrated in Figure 8.5. Environments of this class are parameterized by U, so the learning task is to estimate U. We make the following assumptions:

- The "hypothesis class" consists of all environments with dynamics of the type described

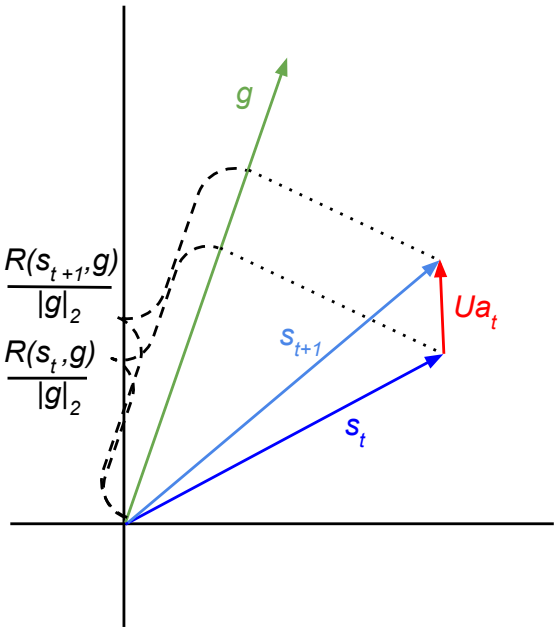


Figure 8.5: Example dense reward environment class. The agent receives a reward proportional to the projection of the state vector onto the goal vector at each step, and can change the state vector by adding an action vector with ℓ_2 distance up to 1 at each step. However, at each step, this action is distorted by an unknown rotation U: the agent must learn to compensate for this distortion.

above. We therefore take as an inductive bias that each model in the considered model class consists of a Q-function $Q_{\tilde{U}}$ and policy $\pi_{\tilde{U}}$ which are in the form of the optimal Q-function and policy for an estimate of U, notated as $\tilde{U} \in \mathbb{R}^{d \times d}$ (constrained to be orthogonal).

- We assume that $Q_{\tilde{U}}$ and $\pi_{\tilde{U}}$ share the same estimated parameter $\tilde{U}$. (This is analogous to – although admittedly stronger than – the parameter sharing we used for Multi-ReenGAGE.) Taken with the above assumption, this implies that $a = \pi_{\tilde{U}}(s, g)$ maximizes $Q_{\tilde{U}}(s, a, g)$, so we do not need to train π separately. Similar parameter sharing occurs between the target policy and Q-function.

- We are comparing our method, minimizing the loss in Equation 8.8, with minimizing the "vanilla" DDPG loss (Equation 8.4).

- States and actions in the replay buffer are in general position.

In this case, the following proposition holds:

Proposition 8.1. *Under the above assumptions, minimizing the ReenGAGE loss can learn U (and therefore learn the optimal policy) using $O(d)$ unique replay buffer transitions. However, minimizing the standard DDPG loss requires at least $O(d^2)$ unique transitions to successfully learn U.*

Proofs are provided in the appendix. This result shows that, in some cases, ReenGAGE requires asymptotically less replay data to successfully learn to perform a task than standard DDPG.

8.6.2.2 Sparse Reward Case

The result shown above might be unsurprising to many readers. Specifically, because the gradient $\nabla_g R(s', g)$ is used by our method and not by standard DDPG, in the dense-reward case, our method is utilizing more information from the environment (to the extent that R, which we assume that agents know *a priori*, is part of the "environment") than the standard algorithm. However, here we show a class of *sparse reward* environments for which the same result holds, despite $\nabla_g R(s', g)$ being unavailable. The environments are constructed as follows:

- $g, a \in \mathbb{R}^d$; $\|a\|_2 \leq 1$; $s \in \mathbb{R}^{2d}$; the state vector consists of two halves, denoted s^1, s^2; we write s as $(s^1; s^2)$.
- $R(s, g) = g^T s^1$

- $\mathcal{T}(s,a) = \begin{cases} (\mathbf{0}; s^1 + Ua) & \text{if } s^1 \neq \mathbf{0} \\ (s^2; \mathbf{0}) & \text{if } s^1 = \mathbf{0} \end{cases}$
- $U \in \mathbb{R}^{d \times d}$ is an unknown orthogonal (rotation) matrix.
- We define $c_{\text{low}} = 0$.

See Figure 8.6 for an illustration. Note that this satisfies sparseness properties: namely, $R(s', g) = 0 = c_{\text{low}}$ at least every other step; and, when $R(s', g) = 0$, then $\nabla_g R(s', g) = \mathbf{0}$ (assuming general position). It is also deterministic, so we do not need to worry about the bias discussed in the previous section. We can therefore apply the sparse version of our method (Equation 8.9), which does *not* use gradient feedback from the reward:

Proposition 8.2. *Under the same assumptions as Proposition 8.1 (replacing Equation 8.8 with Equation 8.9), minimizing the ReenGAGE loss can learn U in the sparse environment class using $O(d)$ unique replay buffer transitions. However, minimizing the standard DDPG loss requires at least $O(d^2)$ unique transitions to successfully learn U.*

This example is admittedly a bit contrived: the single-step reward can always be computed without knowledge of the parameter U. However, it may still give insight about real-world scenarios in which predicting immediate reward is much easier than understanding long-term dynamics.

Note that these two scaling results apply to the number of *replay buffer transitions*. In particular, if a goal relabeling algorithm is used on top of DDPG, then $O(d^2)$ replay buffer transitions may be able to be constructed from $O(d)$ observed training rollout transitions, so standard DDPG *combined with goal relabeling* might only require $O(d)$ training rollout transitions. However, this

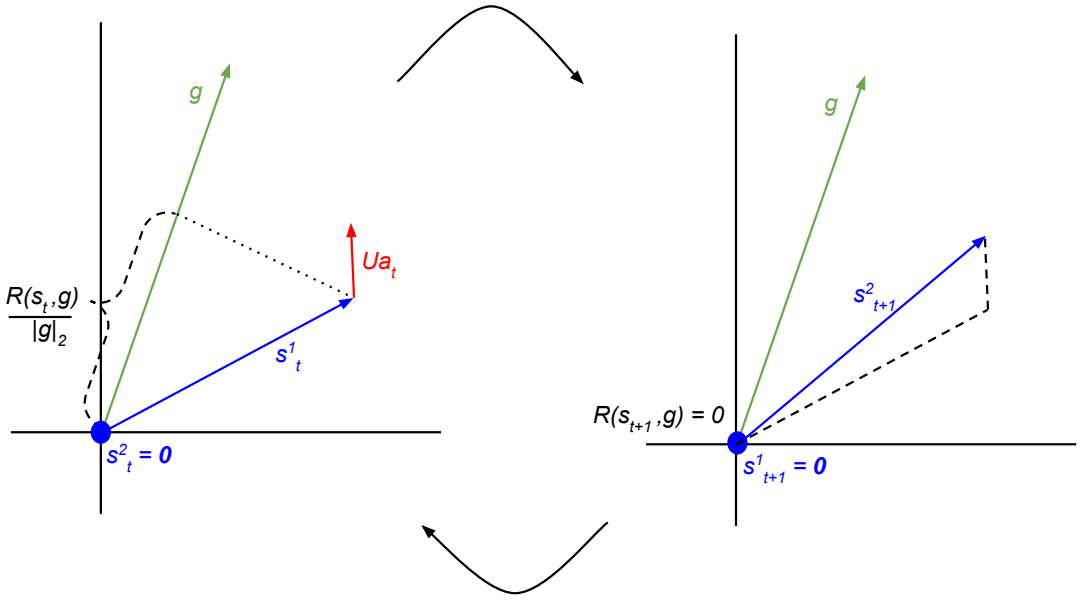


Figure 8.6: Example sparse reward environment class. If s^1 is initially nonzero, then the (rotated) action Ua is added to it, as in the dense case. However, the resulting vector is immediately "stored" in s^2, and s^1 is zeroed: this means that no immediate reward is obtained. In the next step, with s^1 zero, the action is ignored and s^2 is "reloaded" into s^1, resulting in a reward that depends on the *previous* action.

would be computationally expensive, and may not work in practice for particular goal relabeling algorithms. (HER, for instance, only relabels using achieved states from the same episode: if the episode length is $O(1)$ in d, then $O(d^2)$ observed training rollout transitions would still be required for DDPG+HER.) Also, goal relabeling techniques require *a priori* knowledge of the function R, while in the sparse example, ReenGAGE does not (although in the case of this example, we assume that we are using the correct "hypothesis class", i.e., the functional form of $Q_{\tilde{U}}$: constructing this in practice would likely require knowing R).

8.7 Related Works

Many prior approaches have been taken to the goal-conditioned reinforcement learning problem [45]. See [46] for a recent survey of this area. One line of work for this problem involves

automated curriculum generation: here, the idea is to select goals during training that that are dynamically chosen to be the most informative [47, 48, 49]. In the off-policy reinforcement learning setting, a related technique becomes a possibility: one can re-label past experiences with counter-factual goals. This allows a single experienced transition to be used to train for multiple goals, and the re-labeled goals can be chosen using various heuristics to improve training [50, 135, 136, 137]. Note that our proposed method can be combined with any of these off-policy techniques. [145] discusses bias that can result from some goal relabeling techniques. [151] proposes a method based on recursive classification which is in practice similar to hindsight relabeling, but requires less parameter tuning.

In alternative approaches to goal-conditioned RL, [152] has proposed using an on-policy goal-conditioned reinforcement learning technique, using contrastive learning, while [153] and [154] propose model-based techniques.

Note that our proposed method is distinct from *policy distillation* [155]: the goal of policy distillation is to consolidate one or more *already trained* policy networks into a smaller network; whereas our method is intended to improve initial training. Some prior [156, 157, 158] and concurrent [159] works have focused on using attention-based mechanisms to improve either the performance or interpretability of reinforcement learning algorithms. However, to our knowledge, ours is the first to apply gradient-based attention transfer to the critic update to enhance goal-conditioned off-policy reinforcement learning.

Some prior works have been proposed for goal-seeking with structured, complex goals made up of sub-goals, similar to (and in some cases more general than) the multi-goal setting that Multi-ReenGAGE is designed for. Some of these works [160, 161] use a hierarchical policy; however, such a structure may be unable to represent the true optimal policy [162]. [162] pro-

poses a method without this limitation; although the setting considered (Linear Temporal Logic) is different from the multi-goal setting considered here, in that a reward is achieved at most once per episode. [163] proposes a method for arbitrary reward functions specified at test time, under discrete action spaces; in concurrent work [164], this is generalized to continuous actions. [153], a model-based technique mentioned above, can use reward function gradient information to adapt to an arbitrary *shaped* (i.e., non-sparse) reward function at test time.

8.8 Limitations and Conclusion

ReenGAGE has some important limitations. For example, we have seen that the hyper-parameter α requires tuning and can vary greatly (likely due to diverse scales in goal coordinates and rewards); and the benefits of ReenGAGE seem limited to tasks with high goal dimension.

Still, ReenGAGE represents a novel approach to goal-conditioned RL, with benefits demonstrated both empirically and theoretically. In future work, we are particularly interested in exploring the use of Multi-ReenGAGE in safety and robustness applications. In particular, the ability to encode many simultaneous goals at test time could allow the agent to consider many "backup" goals, all of which are acceptable, rather than forcing the agent to focus only a single goal (resulting in total failure if that goal is unreachable.)

Part IV

Conclusion, Future Directions, Appendices, and Bibliography

Chapter 9: Conclusion and Future Directions

In this work, we have presented new state-of-the-art methods for robust machine learning. These include: the first methods for deterministic ℓ_1 and adversarial patch certification to scale to ImageNet, the first methods of any kind for certification against Wasserstein adversarial attacks, "fractional" ℓ_p distortions and "general" adversarial poisoning attacks, as well as greatly improved methods for ℓ_0 and label-flipping poisoning certification. We have also introduced a new method for high-dimensional goal-conditioned reinforcement learning, to allow for improved adaptability to test-time changes in objectives of an RL system. Some possible extensions to this work and future directions in the study of robustness include:

- **Deterministic smoothing for ℓ_p, $p \in (1, 2]$:** It has been shown [51, 123, 124] that randomized smoothing methods are unlikely to be useful for ℓ_p metrics with $p > 2$. However, this leaves open the question as to whether or not randomized smoothing for $p \in (1, 2]$ can be efficiently derandomized, as we showed for $p \in (0, 1]$ in Part I. This would be particularly important, as the ℓ_2 metric is widely considered for adversarial attacks, but there exist no published deterministic certification methods for this norm which scale to ImageNet.

- **Removing restrictions to deterministic methods for ℓ_p, $p \in (0, 1]$:** The deterministic methods presented in Chapters 3 and 4 are specific to bounded, quantized data. We are currently in the early stages of developing a version of these methods which is not restricted

in these ways.

- **Notions of adversarial robustness suitable for reinforcement learning.** While several works have proposed methods for certifiably-robust agents in reinforcement learning under ℓ_p distortions to state observations [165, 166, 167, 168], this notion of robustness is less well-suited to the reinforcement-learning case than to the image-classification case. In particular, while small ℓ_p changes to an image are assumed to be imperceptible and therefore not representative of a meaningful signal, it is entirely possible that two state observations for an RL agent which are very close in ℓ_p space might demand very different actions, depending on the dynamics of the system. (For example, for a robot, avoiding direct collisions may be much more important than avoiding *near* collisions.) It may therefore necessary to develop new notions of provable robustness which are more suitable to systems with discontinuous dynamics.

- **Machine learning with verifiable reasoning.** The certifiable robustness results in this dissertation really only guarantee the *stability* of a model's output, and not necessarily the *correctness* of that output. However, in some domains, it is possible to use machine learning systems to generate output that is verifiably correct. For example, there is a recent line of work using large language models (LLMs) to generate guaranteed-correct formal proofs of given conjectures, by mechanically verifying the correctness of each LLM-generated step of the proof step-by-step as they are generated.[169, 170, 171] However, such techniques do not clearly translate to more open-ended domains, such as computer vision. A difficult but potentially rewarding challenge would be to try to adapt similar methods to more open-ended, real-world domains.

Appendix A: Appendix to Chapter 2

A.1 Architecture and Training Parameters for MNIST

See Tables A.1 and A.2.

Layer	Output Shape
(Input)	$2 \times 28 \times 28$
2D Convolution + ReLU	$64 \times 14 \times 14$
2D Convolution + ReLU	$128 \times 7 \times 7$
Flatten	6272
Fully Connected + ReLU	500
Fully Connected + ReLU	100
Fully Connected + SoftMax	10

Table A.1: Model Architecture of the Base Classifier for MNIST Experiments. 2D Convolution layers both have a kernel size of 4-by-4 pixels, stride of 2 pixels, and padding of 1 pixel.

Training Epochs	400
Batch Size	128
Optimizer	Stochastic Gradient Descent with Momentum
Learning Rate	.01 (Epochs 1-200) .001 (Epochs 201-400)
Momentum	0.9
ℓ_2 Weight Penalty	0

Table A.2: Training Parameters for MNIST Experiments

A.2 Training Parameters for CIFAR-10

As discussed in the main text, we used a standard ResNet18 architecture for our base classifier: the only modification made was to increase the number of input channels from 3 to 6. See Table A.3 for training parameters.

Training Epochs	400
Batch Size	128
Training Set Preprocessing	Random Cropping (Padding:4) and Random Horizontal Flip
Optimizer	Stochastic Gradient Descent with Momentum
Learning Rate	.01 (Epochs 1-200) .001 (Epochs 201-400)
Momentum	0.9
ℓ_2 Weight Penalty	0.0005

Table A.3: Training Parameters for CIFAR-10 Experiments

A.3 Training Parameters for ImageNet

As with CIFAR-10, we used a standard ResNet50 architecture for our base classifier: the only modification made was to increase the number of input channels from 3 to 6. See Table A.4 for training parameters.

A.4 Mutual information derivation for Lee et al. 2019

Here we present a derivation of the expression given in Equation 2.21 in the main text. Let $\mathbf{X}$ be a random variable representing the original image: in this derivation, we assume that $\mathbf{X}$ is

Training Epochs	36
Batch Size	256
Training Set Preprocessing	Random Resizing and Cropping, Random Horizontal Flip
Optimizer	Stochastic Gradient Descent with Momentum
Learning Rate	.1 (21 Epochs) .01 (10 Epochs) .001 (5 Epochs)
Momentum	0.9
ℓ_2 Weight Penalty	0.0001

Table A.4: Training Parameters for ImageNet Experiments

distributed uniformly in $\mathcal{S}^d$. Let $\mathbf{Y}$ be a random variable representing the image, after replacing each pixel with a random, different value with probability $(1 - \kappa)$. By the definition of mutual information, we have:

$$I(\mathbf{X}, \mathbf{Y}) = H(\mathbf{X}) - H(\mathbf{X}|\mathbf{Y}) \tag{A.1}$$

Note that, with $\mathbf{X}$ distributed uniformly, it consists of d i.i.d. instances of a random variable $X_\circ$, itself uniformly distributed in $\mathcal{S}$. Similarly, each component of $\mathbf{Y}$ is an instance of a random variable defined by:

$$Y_\circ = \begin{cases} X_\circ & \text{with probability } \kappa \\ \text{Uniform on } \mathcal{S} - \{X_\circ\} & \text{with probability } 1 - \kappa \end{cases} \tag{A.2}$$

We can then factorize the expression for mutual information, using the fact that each instance of $(X_\circ, Y_\circ)$ is independent:

$$I_{\text{Lee et al.}} = I(\mathbf{X}, \mathbf{Y}) = d(H(X_\circ) - H(X_\circ|Y_\circ)) \tag{A.3}$$

By the definitions of entropy and mutual entropy, we have:

$$I_{\text{Lee et al.}} = -d\Bigg(\sum_{s\in\mathcal{S}} \Pr(X_\circ = s)\log_2 \Pr(X_\circ = s) - \sum_{(s,s')} \Pr(X_\circ = s, Y_\circ = s')\log_2 \frac{\Pr(X_\circ = s, Y_\circ = s')}{\Pr(Y_\circ = s')}\Bigg) \tag{A.4}$$

Note that, by symmetry, $Y_\circ$ is itself uniformly distributed on $\mathcal{S}$. Then we have:

$$I_{\text{Lee et al.}} = -d\Bigg(\sum_{s\in\mathcal{S}} |\mathcal{S}|^{-1}\log_2 |\mathcal{S}|^{-1} - \sum_{(s,s')} \Pr(X_\circ = s, Y_\circ = s')\log_2 \frac{\Pr(X_\circ = s, Y_\circ = s')}{|\mathcal{S}|^{-1}}\Bigg) \tag{A.5}$$

Splitting (s, s') into cases for $(s = s')$ and $(s \neq s')$:

$$I_{\text{Lee et al.}} = -d\Bigg(\sum_{s} |\mathcal{S}|^{-1}\log_2 |\mathcal{S}|^{-1} - \sum_{s} \Pr(X_\circ = Y_\circ = s)\log_2 \frac{\Pr(X_\circ = Y_\circ = s)}{|\mathcal{S}|^{-1}} - \sum_{s\neq s'} \Pr(X_\circ = s, Y_\circ = s')\log_2 \frac{\Pr(X_\circ = s, Y_\circ = s')}{|\mathcal{S}|^{-1}}\Bigg) \tag{A.6}$$

Note that $\Pr(X_\circ = Y_\circ = s) = |\mathcal{S}|^{-1}\kappa$, because $X_\circ = s$ with probability $|\mathcal{S}|^{-1}$, and then $Y_\circ$ is assigned to $X_\circ$ with probability κ. Also, for $s \neq s'$, we have

$$\Pr(X_\circ = s, Y_\circ = s') = |\mathcal{S}|^{-1}(1-\kappa)(|\mathcal{S}|-1)^{-1}, \tag{A.7}$$

because $X_\circ = s$ with probability $|\mathcal{S}|^{-1}$, $Y_\circ$ is not equal to $X_\circ$ with probability $(1-\kappa)$, and then $Y_\circ$ assumes each value in $\mathcal{S} - \{X_\circ\}$ with uniform probability. Plugging these expressions into

Equation A.6 gives:

$$I_{\text{Lee et al.}} = -d\Bigg(\sum_{s} \frac{\log_2 |\mathcal{S}|^{-1}}{|\mathcal{S}|} - \sum_{s} \frac{\kappa}{|\mathcal{S}|} \log_2 \kappa \\ - \sum_{s \neq s'} \frac{(1-\kappa)}{(|\mathcal{S}|-1)|\mathcal{S}|} \log_2 \left[(1-\kappa)(|\mathcal{S}|-1)^{-1}\right]\Bigg) \tag{A.8}$$

Now all summands are constants: we note that summing over all $s \in \mathcal{S}$ is now equivalent to multiplying by $|\mathcal{S}|$ and summing over $(s, s') \in \mathcal{S}^2$ with $s \neq s'$ is equivalent to multiplying by $|\mathcal{S}|(|\mathcal{S}|-1)$:

$$I_{\text{Lee et al.}} = -d\Big(\log_2 |\mathcal{S}|^{-1} - \kappa \log_2 \kappa \\ - (1-\kappa) \log_2 \left[(1-\kappa)(|\mathcal{S}|-1)^{-1}\right]\Big) \tag{A.9}$$

This simplifies to the expression given in the text.

A.5 Additional Adversarial Examples

See Figure A.1.

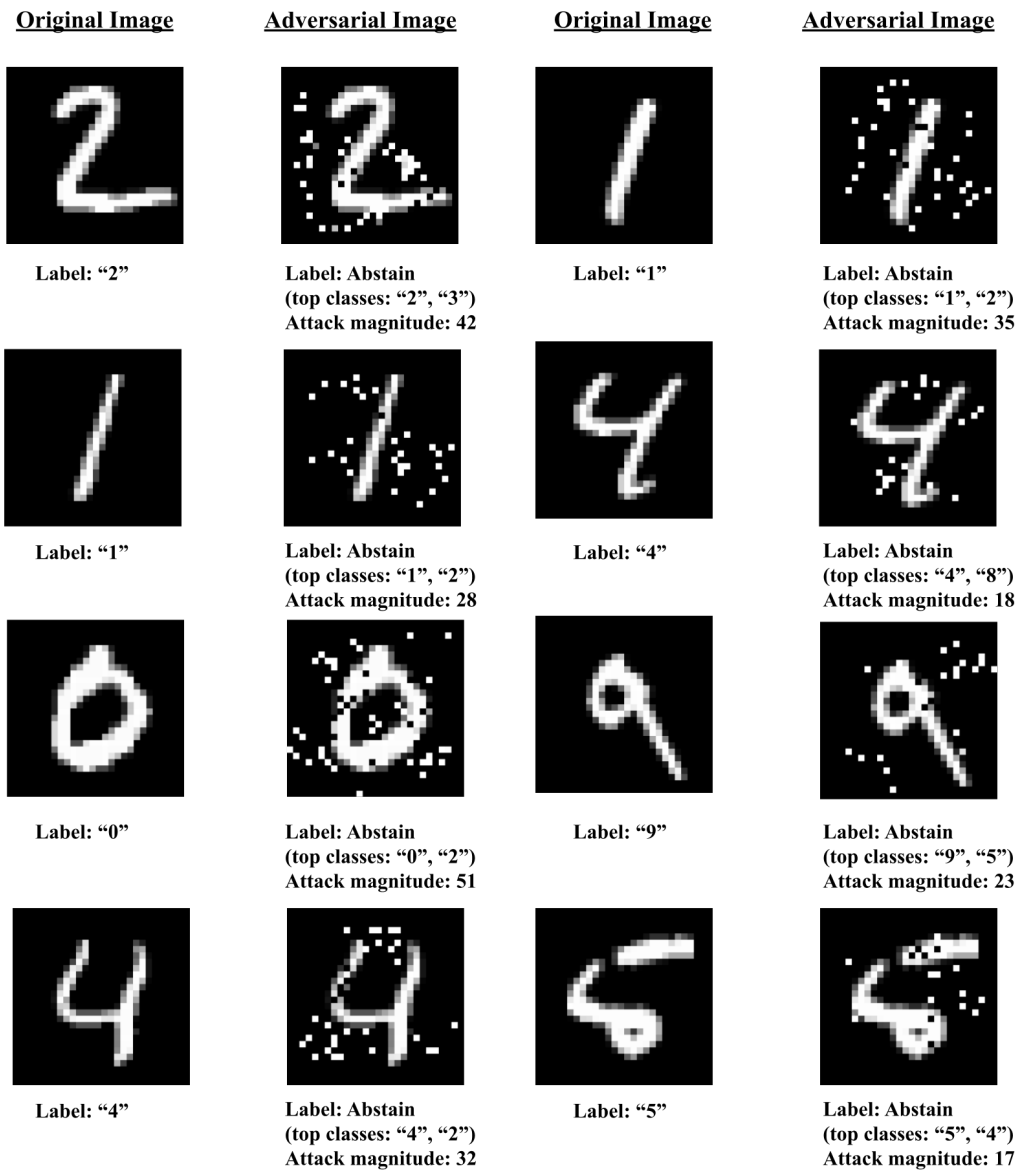


Figure A.1: Additional adversarial examples generated on MNIST by the Pointwise attack on our robust classifier, with $k = 45$.

Appendix B: Appendix to Chapter 3

B.1 Proofs

Theorem 3.1 (Lee et al. [1])**.** *For any $f : \mathbb{R}^d \to [0,1]$ and parameter $\lambda \in \mathbb{R}^+$, define:*

$$p(\boldsymbol{x}) := \mathop{\mathbb{E}}_{\epsilon \sim \mathcal{U}^d(-\lambda,\lambda)} \left[f(\boldsymbol{x} + \epsilon) \right]. \tag{B.1}$$

Then, $p(.)$ is $1/(2\lambda)$-Lipschitz with respect to the ℓ_1 norm.

Proof. Consider two arbitrary points $\boldsymbol{x}, \boldsymbol{x}'$ where $\delta := \boldsymbol{x}' - \boldsymbol{x}$. We consider two cases.

- Case 1: $\|\delta\|_1 \geq 2\lambda$: Then, because $f(\cdot) \in [0,1]$, and therefore $p(\cdot) \in [0,1]$, we have:

$$|p(\boldsymbol{x}) - p(\boldsymbol{x}')| \leq 1 \leq \frac{\|\delta\|_1}{2\lambda} \tag{B.2}$$

- Case 2: $\|\delta\|_1 < 2\lambda$: In this case, for each i, $|\delta_i| < 2\lambda$. Define $\mathcal{B}(\boldsymbol{x})$ as the ℓ_∞ ball of radius λ around $\boldsymbol{x}$, and $\mathcal{U}(\mathcal{B}(\boldsymbol{x}))$ as the uniform distribution on this ball (and, similarly $\mathcal{U}(\cdot)$, on any other set). In other words:

$$p(\boldsymbol{x}) = \mathop{\mathbb{E}}_{\boldsymbol{z} \sim \mathcal{U}(\mathcal{B}(\boldsymbol{x}))} f(\boldsymbol{z}) \tag{B.3}$$

Then,

$$
\begin{aligned}
&|p(\boldsymbol{x}) - p(\boldsymbol{x}')| \\
&= \Big| \mathop{\mathbb{E}}_{\boldsymbol{z}\sim\mathcal{U}(\mathcal{B}(\boldsymbol{x}))} f(\boldsymbol{z}) - \mathop{\mathbb{E}}_{\boldsymbol{z}\sim\mathcal{U}(\mathcal{B}(\boldsymbol{x}'))} f(\boldsymbol{z}) \Big| \\
&= \Big| \Big(\Pr_{\boldsymbol{z}\sim\mathcal{U}(\mathcal{B}(\boldsymbol{x}))} \boldsymbol{z} \in \mathcal{B}(\boldsymbol{x}) \setminus \mathcal{B}(\boldsymbol{x}') \mathop{\mathbb{E}}_{\substack{\boldsymbol{z}\sim\mathcal{U}(\mathcal{B}(\boldsymbol{x}) \\ \setminus\mathcal{B}(\boldsymbol{x}'))}} f(\boldsymbol{z}) \\
&\quad + \Pr_{\boldsymbol{z}\sim\mathcal{U}(\mathcal{B}(\boldsymbol{x}))} \boldsymbol{z} \in \mathcal{B}(\boldsymbol{x}) \cap \mathcal{B}(\boldsymbol{x}') \mathop{\mathbb{E}}_{\substack{\boldsymbol{z}\sim\mathcal{U}(\mathcal{B}(\boldsymbol{x}) \\ \cap\mathcal{B}(\boldsymbol{x}'))}} f(\boldsymbol{z}) \Big) \\
&\quad - \Big(\Pr_{\boldsymbol{z}\sim\mathcal{U}(\mathcal{B}(\boldsymbol{x}'))} \boldsymbol{z} \in \mathcal{B}(\boldsymbol{x}') \setminus \mathcal{B}(\boldsymbol{x}) \mathop{\mathbb{E}}_{\substack{\boldsymbol{z}\sim\mathcal{U}(\mathcal{B}(\boldsymbol{x}') \\ \setminus\mathcal{B}(\boldsymbol{x}))}} f(\boldsymbol{z}) \\
&\quad + \Pr_{\boldsymbol{z}\sim\mathcal{U}(\mathcal{B}(\boldsymbol{x}'))} \boldsymbol{z} \in \mathcal{B}(\boldsymbol{x}) \cap \mathcal{B}(\boldsymbol{x}') \mathop{\mathbb{E}}_{\substack{\boldsymbol{z}\sim\mathcal{U}(\mathcal{B}(\boldsymbol{x}) \\ \cap\mathcal{B}(\boldsymbol{x}'))}} f(\boldsymbol{z}) \Big) \Big| \\
&= \Big| \Pr_{\boldsymbol{z}\sim\mathcal{U}(\mathcal{B}(\boldsymbol{x}))} \boldsymbol{z} \in \mathcal{B}(\boldsymbol{x}) \setminus \mathcal{B}(\boldsymbol{x}') \mathop{\mathbb{E}}_{\substack{\boldsymbol{z}\sim\mathcal{U}(\mathcal{B}(\boldsymbol{x}) \\ \setminus\mathcal{B}(\boldsymbol{x}'))}} f(\boldsymbol{z}) \\
&\quad - \Pr_{\boldsymbol{z}\sim\mathcal{U}(\mathcal{B}(\boldsymbol{x}'))} \boldsymbol{z} \in \mathcal{B}(\boldsymbol{x}') \setminus \mathcal{B}(\boldsymbol{x}) \mathop{\mathbb{E}}_{\substack{\boldsymbol{z}\sim\mathcal{U}(\mathcal{B}(\boldsymbol{x}') \\ \setminus\mathcal{B}(\boldsymbol{x}))}} f(\boldsymbol{z}) \Big|
\end{aligned}
\tag{B.4}
$$

Note that:

$$
\begin{aligned}
&\Pr_{\boldsymbol{z}\sim\mathcal{U}(\mathcal{B}(\boldsymbol{x}'))} \boldsymbol{z} \in \mathcal{B}(\boldsymbol{x}') \setminus \mathcal{B}(\boldsymbol{x}) \\
&= \Pr_{\boldsymbol{z}\sim\mathcal{U}(\mathcal{B}(\boldsymbol{x}))} \boldsymbol{z} \in \mathcal{B}(\boldsymbol{x}) \setminus \mathcal{B}(\boldsymbol{x}')
\end{aligned}
\tag{B.5}
$$

Because both represent the probability of a uniform random variable on an ℓ_∞ ball of radius λ taking a value outside of the region $\mathcal{B}(\boldsymbol{x}) \cap \mathcal{B}(\boldsymbol{x}')$ (which is entirely contained within both

balls.) Then:

$$
\begin{aligned}
&|p(\boldsymbol{x}) - p(\boldsymbol{x}')| \\
&= \Pr_{\boldsymbol{z}\sim\mathcal{U}(\mathcal{B}(\boldsymbol{x}))} \boldsymbol{z} \in \mathcal{B}(\boldsymbol{x}) \setminus \mathcal{B}(\boldsymbol{x}') \\
&\times \Big| \mathop{\mathbb{E}}_{\substack{\boldsymbol{z}\sim\mathcal{U}(\mathcal{B}(\boldsymbol{x})\setminus \\ \mathcal{B}(\boldsymbol{x}'))}} f(\boldsymbol{z}) - \mathop{\mathbb{E}}_{\substack{\boldsymbol{z}\sim\mathcal{U}(\mathcal{B}(\boldsymbol{x}') \\ \setminus\mathcal{B}(\boldsymbol{x}))}} f(\boldsymbol{z}) \Big| \\
&\leq \Pr_{\boldsymbol{z}\sim\mathcal{U}(\mathcal{B}(\boldsymbol{x}))} \boldsymbol{z} \in \mathcal{B}(\boldsymbol{x}) \setminus \mathcal{B}(\boldsymbol{x}').
\end{aligned}
\tag{B.6}
$$

Where, in the last line, we used the fact that $f(\cdot) \in [0, 1]$. Let $\mathcal{V}(\mathcal{S})$ represent the volume of a set $\mathcal{S}$. Note that $\mathcal{B}(\boldsymbol{x}) \cap \mathcal{B}(\boldsymbol{x}')$ is a d-hyperrectangle, with each edge of length

$$
\min(x_i, x_i') + \lambda - (\max(x_i, x_i') - \lambda) = 2\lambda - |\delta_i| \tag{B.7}
$$

Then following Equation B.6,

$$
\begin{aligned}
&|p(\boldsymbol{x}) - p(\boldsymbol{x}')| \\
&\leq \frac{\mathcal{V}(\mathcal{B}(\boldsymbol{x})) - \mathcal{V}(\mathcal{B}(\boldsymbol{x}) \cap \mathcal{B}(\boldsymbol{x}'))}{\mathcal{V}(\mathcal{B}(\boldsymbol{x}))} \\
&= 1 - \frac{\Pi_{i=1}^{d}(2\lambda - |\delta_i|)}{(2\lambda)^d} \\
&= 1 - \prod_{i=1}^{d}\left(1 - \frac{|\delta_i|}{2\lambda}\right)
\end{aligned}
\tag{B.8}
$$

Note that, for $1 \leq d' \leq d$:

$$\begin{aligned}
&\prod_{i=1}^{d'} \left(1 - \frac{|\delta_i|}{2\lambda}\right) \\
&= \prod_{i=1}^{d'-1} \left(1 - \frac{|\delta_i|}{2\lambda}\right) - \frac{|\delta_{d'}|}{2\lambda} \prod_{i=1}^{d'-1} \left(1 - \frac{|\delta_i|}{2\lambda}\right) \\
&\geq \prod_{i=1}^{d'-1} \left(1 - \frac{|\delta_i|}{2\lambda}\right) - \frac{|\delta_{d'}|}{2\lambda}
\end{aligned} \tag{B.9}$$

By induction:

$$\prod_{i=1}^{d} \left(1 - \frac{|\delta_i|}{2\lambda}\right) \geq 1 - \sum_{i=1}^{d} \frac{|\delta_i|}{2\lambda} \tag{B.10}$$

Therefore,

$$\begin{aligned}
&|p(\boldsymbol{x}) - p(\boldsymbol{x}')| \\
&\leq 1 - \prod_{i=1}^{d} \left(1 - \frac{|\delta_i|}{2\lambda}\right) \\
&\leq 1 - \left(1 - \sum_{i=1}^{d} \frac{|\delta_i|}{2\lambda}\right) \\
&= \frac{\|\delta\|_1}{2\lambda}
\end{aligned} \tag{B.11}$$

Thus, by the definition of Lipschitz-continuity, p is $1/(2\lambda)$-Lipschitz with respect to the ℓ_1 norm.

□

Theorem 3.2 (General Case)**.** *For any $f : \mathbb{R}^d \to [0, 1]$, and $\lambda > 0$ let $\boldsymbol{s} \in [0, 2\lambda]^d$ be a random variable, with a fixed distribution such that:*

$$s_i \sim \mathcal{U}(0, 2\lambda), \quad \forall i. \tag{B.12}$$

*Note that the components $s_1, ..., s_d$ are **not** required to be distributed independently from each*

other. Then, define:

$$\tilde{x}_i := \frac{\min(2\lambda\lceil \frac{x_i - s_i}{2\lambda} \rceil + s_i, 1)}{2} \tag{B.13}$$

$$+ \frac{\max(2\lambda\lceil \frac{x_i - s_i}{2\lambda} - 1 \rceil + s_i, 0)}{2}\,, \quad \forall i \tag{B.14}$$

$$p(\boldsymbol{x}) := \mathop{\mathbb{E}}_{\boldsymbol{s}}[f(\tilde{\boldsymbol{x}})]\,. \tag{B.15}$$

Then, $p(.)$ is $1/(2\lambda)$-Lipschitz with respect to the ℓ_1 norm.

Proof. Consider two arbitrary points $\boldsymbol{x}, \boldsymbol{x}'$ where $\delta := \boldsymbol{x}' - \boldsymbol{x}$. We consider two cases.

- Case 1: $\|\delta\|_1 \geq 2\lambda$: Then, because $f(\cdot) \in [0, 1]$, and therefore $p(\cdot) \in [0, 1]$, we have:

$$|p(\boldsymbol{x}) - p(\boldsymbol{x}')| \leq 1 \leq \frac{\|\delta\|_1}{2\lambda} \tag{B.16}$$

- Case 2: $\|\delta\|_1 < 2\lambda$:

  In this case, for each i, $|\delta_i| < 2\lambda$, and therefore $\lceil \frac{x_i - s_i}{2\lambda} \rceil$ and $\lceil \frac{x_i' - s_i}{2\lambda} \rceil$ differ by at most one. Furthermore, $\lceil \frac{x_i - s_i}{2\lambda} \rceil$ differs from $\lceil \frac{x_i}{2\lambda} \rceil$ by at most one, and similarly for x_i'. Without loss of generality, assume $x_i < x_i'$ (i.e., $\delta_i = |\delta_i| = x_i' - x_i$).

  There are two cases:

  - Case A: $\lceil \frac{x_i}{2\lambda} \rceil = \lceil \frac{x_i'}{2\lambda} \rceil$. Let this integer be n. Then:

    * $\lceil \frac{x_i - s_i}{2\lambda} \rceil = \lceil \frac{x_i' - s_i}{2\lambda} \rceil = n$ iff $\frac{s_i}{2\lambda} < \frac{x_i}{2\lambda} - (n-1)$ (which also implies $\frac{s_i}{2\lambda} < \frac{x_i'}{2\lambda} - (n-1)$).
    * $\lceil \frac{x_i - s_i}{2\lambda} \rceil = \lceil \frac{x_i' - s_i}{2\lambda} \rceil = n-1$ iff $\frac{s_i}{2\lambda} \geq \frac{x_i'}{2\lambda} - (n-1)$ (which also implies $\frac{s_i}{2\lambda} \geq \frac{x_i}{2\lambda} - (n-1)$).

    Then $\lceil \frac{x_i - s_i}{2\lambda} \rceil$ and $\lceil \frac{x_i' - s_i}{2\lambda} \rceil$ differ only if $\frac{x_i}{2\lambda} - (n-1) \leq \frac{s_i}{2\lambda} < \frac{x_i'}{2\lambda} - (n-1)$, which occurs with probability $\frac{\delta_i}{2\lambda}$.

– Case B: $\lceil \frac{x_i}{2\lambda} \rceil + 1 = \lceil \frac{x_i'}{2\lambda} \rceil$. Let $n := \lceil \frac{x_i}{2\lambda} \rceil$. Then $\lceil \frac{x_i - s_i}{2\lambda} \rceil$ and $\lceil \frac{x_i' - s_i}{2\lambda} \rceil$ can differ if either:

  * $\lceil \frac{x_i - s_i}{2\lambda} \rceil = n$ and $\lceil \frac{x_i' - s_i}{2\lambda} \rceil = n + 1$. This occurs iff $\frac{s_i}{2\lambda} < \frac{x_i'}{2\lambda} - n$ (which also implies $\frac{s_i}{2\lambda} < \frac{x_i}{2\lambda} - (n-1)$).

  * $\lceil \frac{x_i - s_i}{2\lambda} \rceil = n - 1$ and $\lceil \frac{x_i' - s_i}{2\lambda} \rceil = n$. This occurs iff $\frac{s_i}{2\lambda} \geq \frac{x_i}{2\lambda} - (n-1)$ (which also implies $\frac{s_i}{2\lambda} \geq \frac{x_i'}{2\lambda} - n$).

  In other words, $\lceil \frac{x_i - s_i}{2\lambda} \rceil = \lceil \frac{x_i' - s_i}{2\lambda} \rceil$ iff:

$$\frac{x_i}{2\lambda} - (n-1) > \frac{s_i}{2\lambda} \geq \frac{x_i'}{2\lambda} - n$$

  Or equivalently:

$$\frac{x_i}{2\lambda} - n + 1 > \frac{s_i}{2\lambda} \geq \frac{x_i}{2\lambda} - n + \frac{\delta_i}{2\lambda}$$

  This happens with probability $1 - \frac{\delta_i}{2\lambda}$. Therefore, $\lceil \frac{x_i - s_i}{2\lambda} \rceil$ and $\lceil \frac{x_i' - s_i}{2\lambda} \rceil$ differ with probability $\frac{\delta_i}{2\lambda}$.

Note that $\lceil \frac{x_i - s_i}{2\lambda} - 1 \rceil$ and $\lceil \frac{x_i' - s_i}{2\lambda} - 1 \rceil$ differ only when $\lceil \frac{x_i - s_i}{2\lambda} \rceil$ and $\lceil \frac{x_i' - s_i}{2\lambda} \rceil$ differ. Therefore in both cases, $\tilde{x}_i$ and $\tilde{x}_i'$ differ with probability at most $\frac{|\delta_i|}{2\lambda}$. The rest of the proof proceeds as in the $\lambda \geq 0.5$ case in the main text.

□

Corollary 3.1 (General Case)**.** *For any $f : \mathbb{R}^d \to [0,1]$, and $\lambda \geq 0$ (with 2λ a multiple of $1/q$), let $\boldsymbol{s} \in [0, 2\lambda - 1/q]^d_{(q)} + \mathbb{1}/(2q)$ be a random variable with a fixed distribution such that:*

$$s_i \sim \mathcal{U}_{(q)}(0, 2\lambda - 1/q) + 1/(2q), \quad \forall i. \tag{B.17}$$

*Note that the components $s_1, ..., s_d$ are **not** required to be distributed independently from each other. Then, define:*

$$\tilde{\mathbf{x}}_i := \frac{\min\left(2\lambda\left\lceil\frac{\mathbf{x}_i - s_i}{2\lambda}\right\rceil + s_i, 1\right)}{2} \tag{B.18}$$

$$+ \frac{\max\left(2\lambda\left\lceil\frac{\mathbf{x}_i - s_i}{2\lambda} - 1\right\rceil + s_i, 0\right)}{2}\,, \quad \forall i \tag{B.19}$$

$$p(\mathbf{x}) := \mathop{\mathbb{E}}_{\boldsymbol{s}}\left[f(\tilde{\mathbf{x}})\right]. \tag{B.20}$$

Then, $p(.)$ is $1/(2\lambda)$-Lipschitz with respect to the ℓ_1 norm on the quantized domain $\mathbf{x} \in [0,1]^d_{(q)}$.

Proof. The proof is substantially similar to the proof of the continuous case above. Minor differences occur in Cases 2.A and 2.B (mostly due to inequalities becoming strict, because possible values of s_i are offset from values of $\mathbf{x}_i$) which we show here:

- Case A: $\left\lceil\frac{\mathbf{x}_i}{2\lambda}\right\rceil = \left\lceil\frac{\mathbf{x}'_i}{2\lambda}\right\rceil$. Let this integer be n. Then:
  - $\left\lceil\frac{\mathbf{x}_i - s_i}{2\lambda}\right\rceil = \left\lceil\frac{\mathbf{x}'_i - s_i}{2\lambda}\right\rceil = n$ iff $\frac{s_i}{2\lambda} < \frac{\mathbf{x}_i}{2\lambda} - (n-1)$ (which also implies $\frac{s_i}{2\lambda} < \frac{\mathbf{x}'_i}{2\lambda} - (n-1)$).
  - $\left\lceil\frac{\mathbf{x}_i - s_i}{2\lambda}\right\rceil = \left\lceil\frac{\mathbf{x}'_i - s_i}{2\lambda}\right\rceil = n-1$ iff $\frac{s_i}{2\lambda} > \frac{\mathbf{x}'_i}{2\lambda} - (n-1)$ (which also implies $\frac{s_i}{2\lambda} > \frac{\mathbf{x}_i}{2\lambda} - (n-1)$).

  Then $\left\lceil\frac{\mathbf{x}_i - s_i}{2\lambda}\right\rceil$ and $\left\lceil\frac{\mathbf{x}'_i - s_i}{2\lambda}\right\rceil$ differ only if $\frac{\mathbf{x}_i}{2\lambda} - (n-1) < \frac{s_i}{2\lambda} < \frac{\mathbf{x}'_i}{2\lambda} - (n-1)$. There are exactly $q \cdot \delta_i$ discrete values that s_i can take such that this condition holds. This is out of $2\lambda q$ possible values over which s_i is uniformly distributed. Therefore, the condition holds with probability $\frac{\delta_i}{2\lambda}$.

- Case B: $\left\lceil\frac{\mathbf{x}_i}{2\lambda}\right\rceil + 1 = \left\lceil\frac{\mathbf{x}'_i}{2\lambda}\right\rceil$. Let $n := \left\lceil\frac{\mathbf{x}_i}{2\lambda}\right\rceil$. Then $\left\lceil\frac{\mathbf{x}_i - s_i}{2\lambda}\right\rceil$ and $\left\lceil\frac{\mathbf{x}'_i - s_i}{2\lambda}\right\rceil$ can differ if either:
  - $\left\lceil\frac{\mathbf{x}_i - s_i}{2\lambda}\right\rceil = n$ and $\left\lceil\frac{\mathbf{x}'_i - s_i}{2\lambda}\right\rceil = n+1$. This occurs iff $\frac{s_i}{2\lambda} < \frac{\mathbf{x}'_i}{2\lambda} - n$ (which also implies $\frac{s_i}{2\lambda} < \frac{\mathbf{x}_i}{2\lambda} - (n-1)$).

– $\lceil \frac{\mathbf{x}_i - s_i}{2\lambda} \rceil = n - 1$ and $\lceil \frac{\mathbf{x}'_i - s_i}{2\lambda} \rceil = n$. This occurs iff $\frac{s_i}{2\lambda} > \frac{\mathbf{x}_i}{2\lambda} - (n-1)$ (which also implies $\frac{s_i}{2\lambda} > \frac{\mathbf{x}'_i}{2\lambda} - n$).

In other words, $\lceil \frac{\mathbf{x}_i - s_i}{2\lambda} \rceil = \lceil \frac{\mathbf{x}'_i - s_i}{2\lambda} \rceil$ iff:

$$\frac{\mathbf{x}_i}{2\lambda} - (n-1) > \frac{s_i}{2\lambda} > \frac{\mathbf{x}'_i}{2\lambda} - n$$

Or equivalently:

$$\frac{\mathbf{x}_i}{2\lambda} - n + 1 > \frac{s_i}{2\lambda} > \frac{\mathbf{x}_i}{2\lambda} - n + \frac{\delta_i}{2\lambda}$$

There are exactly $q \cdot (1 - \delta_i)$ discrete values that s_i can take such that this condition holds. This is out of $2\lambda q$ possible values over which s_i is uniformly distributed. Therefore, the condition holds with probability $\frac{1-\delta_i}{2\lambda}$. Thus, $\lceil \frac{\mathbf{x}_i - s_i}{2\lambda} \rceil$ and $\lceil \frac{\mathbf{x}'_i - s_i}{2\lambda} \rceil$ differ with probability $\frac{\delta_i}{2\lambda}$.

□

B.2 Experimental Details

For uniform additive noise, we reproduced Yang et al. [5]'s results directly, using their released code. Note that we also reproduced the training of all models, rather than using released models. For Independent SSN and DSSN, we followed the same training procedure as in Yang et al. [5], but instead used the noise distribution of our methods during training. For DSSN, we used the same vector $\mathbf{v}$ to generate noise during training and test time: note that our certificate requires $\mathbf{v}$ to be the same fixed vector whenever the classifier is used. In particular, we used a pseudorandom array generated using the Mersenne Twister algorithm with seed 0, as implemented in NumPy as numpy.random.RandomState. This is guaranteed to produce identical

results on all platforms and for all future versions of NumPy, given the same seed, so in practice we only store the seed (0). In Section B.3, we explore the sensitivity of our method to different choices of pseudorandom seeds.

In a slight deviation from Cohen et al. [32], Yang et al. [5] uses different noise vectors for each sample in a batch when training (Cohen et al. [32] uses the same ϵ for all samples in a training batch to improve speed). We follow Yang et al. [5]'s method: this means that when training DSSN, we train the classifier on each sample only once per epoch, with a single, randomly-chosen value of s_{base}, which varies between samples in a batch.

Training parameters (taken from Yang et al. [5]) were as follows (Table B.1):

	CIFAR-10	ImageNet
Architecture	WideResNet-40	ResNet-50
Number of Epochs	120	30
Batch Size	64 [1]	64
Initial Learning Rate	0.1	0.1
LR Scheduler	Cosine Annealing	Cosine Annealing

Table B.1: Training parameters for experiments.

For all certification results in the main text, and most training results, we used a single NVIDIA 2080 Ti GPU. (Some experiments with denoisers in Section B.4, as well as ImageNet stability training, used two GPUs.)

For testing, we used the entire CIFAR-10 test set (10,000 images) and a subset of 500

[1] There is a discrepancy between the code and the text of Yang et al. [5] about the batch size used for training on CIFAR-10: the paper says to use a batch size of 128, while the instructions for reproducing the paper's results released with the code use a batch size of 64. Additionally, inspection of one of Yang et al. [5]'s released models indicates that a batch size of 64 was in fact used. (In particular, the "num_batches_tracked" field in the saved model, which counts the total number of batches used in training, corresponded with a batch size of 64.) We therefore used a batch size of 64 in our reproduction, assuming that the discrepancy was a result of a typo in that paper.

images of ImageNet (the same subset used by Cohen et al. [32]).

When reporting clean accuracies for randomized techniques (uniform additive noise and Independent SSN), we followed [5] by simply reporting the percent of samples for which the $N_0 = 64$ initial noise perturbations, used to pick the top class during certification, actually selected the correct class. (Notably, [5] does not use an "abstain" option for prediction, as some other randomized smoothing works [32] do.) On the one hand, this is an inexact estimate of the accuracy of the *true* classifier $p(\boldsymbol{x})$, which uses the true expectation. On the other hand, it is the actual, empirical accuracy of a classifier that is being used in practice. This is not an issue when reporting the clean accuracy for DSSN, which is exact.

In DSSN, as originally proposed in [29] (Chapter 6 of this dissertation, published before this chapter was written), if two classes tie in the number of "votes", we predict the first class lexicographically: this means that we can certify robustness up to *and including* the radius ρ, because we are guaranteed consistent behavior in the case of ties. Reported certified radii for DSSN should therefore be interpreted to guarantee robustness even in the $\|\mathbf{x}-\mathbf{x}'\|_1 = \rho$ case. (This is not a meaningful distinction in randomized methods where the space is taken as continuous).

B.3 Effect of pseudorandom choice of $\mathbf{v}$

In Section B.2, we mention that the vector $\mathbf{v}$ used in the derandomization of DSSN, which must be re-used every time the classifier is used, is generated pseudorandomly, using a seed of 0 in all experiments. In this section, we explore the sensitivity of our results to the choice of vector $\mathbf{v}$, and in particular to the choice of random seed. To do this, we repeated all standard-training DSSN experiments on CIFAR-10, using two additional choices of random seeds. We

	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0
Seed = 0	72.25% (81.50% @ σ=0.75)	63.07% (77.85% @ σ=1.25)	56.21% (71.17% @ σ=2.25)	51.33% (67.98% @ σ=3.0)	46.76% (65.40% @ σ=3.5)	42.66% (65.40% @ σ=3.5)	38.26% (65.40% @ σ=3.5)	33.64% (65.40% @ σ=3.5)
Seed = 1	72.01% (81.85% @ σ=0.75)	62.73% (75.64% @ σ=1.5)	56.03% (72.19% @ σ=2.0)	51.20% (67.65% @ σ=3.0)	46.71% (66.93% @ σ=3.25)	42.45% (66.19% @ σ=3.5)	37.87% (66.19% @ σ=3.5)	33.08% (66.19% @ σ=3.5)
Seed = 2	72.62% (81.19% @ σ=0.75)	62.79% (74.26% @ σ=1.75)	56.06% (70.13% @ σ=2.5)	51.02% (70.13% @ σ=2.5)	46.85% (65.33% @ σ=3.5)	42.52% (65.33% @ σ=3.5)	38.22% (65.33% @ σ=3.5)	33.53% (65.33% @ σ=3.5)

Table B.2: Comparison of DSSN using different random seeds to generate $\mathbf{v}$ on CIFAR-10. Matching Yang et al. [5], we test on 15 noise levels ($\sigma \in \{0.15, 0.25n \text{ for } 1 \leq n \leq 14\}$). We report the best certified accuracy at a selection of radii ρ, as well as the clean accuracy and noise level of the associated classifier. We find very little difference between the different seed values, with all certified accuracies within ± 0.65 percentage points of each other.

performed both training and certification using the assigned $\mathbf{v}$ vector for each experiment. Result are summarized in Table B.2. We report a tabular summary, rather than certification curves, because the curves are too similar to distinguish. In general, the choice of random seed to select $\mathbf{v}$ does not seem to impact the certified accuracies: all best certified accuracies were within 0.65 percentage points of each other. This suggests that our method is robust to the choice of this hyperparameter.

B.4 Effect of a Denoiser

As shown in Figure 3.6 in the main text, at large λ, there is a substantial benefit to SSN which is unrelated to derandomization, due to the differences in noise distributions discussed in Section 3.3.2.1. However, Equation 3.23 shows that the difference between uniform additive noise and Independent SSN is a simple, deterministic transformation on each pixel. We therefore wondered whether training a denoiser network, to learn the relationship between $\boldsymbol{x}$ and the noisy sample ($\boldsymbol{x} + \epsilon$ or $\tilde{\boldsymbol{x}}$), would eliminate the differences between the methods. Salman et al.

[79] proposes methods of training denoisers for randomized smoothing, in the context of using smoothing on pre-trained classifiers. In this context, the noisy image first passes through a denoiser network, before being passed into a classification network trained on clean images. We used their code (and all default parameters), in three variations:

1. **Stability Denoising**: In this method, the pre-trained classifier network is required for training the denoiser. The loss when training the denoiser is based on the consistency between the logit outputs of the classifier on the clean input $\boldsymbol{x}$ and on the denoised version of the noisy input. This is the best-performing method in [79]. However, note that it does not directly use the pixel values of $\boldsymbol{x}$ when training the denoiser, and therefore might not "learn" the correspondence between clean and noisy samples (Figure 3.2 in the main text) as easily.

2. **MSE Denoising**: This trains the denoiser via direct supervised training, with the objective of reducing the mean squared error difference between the pixel values of the clean and denoised samples. Then, classification is done using a classifier that is pre-trained only on clean samples. This performs relatively poorly in [79], but should directly learn the correspondence between clean and noisy samples.

3. **MSE Denoising with Retraining**: For this experiment, we trained an MSE denoiser as above, but *then* trained the entire classification pipeline (the denoiser + the classifier) on noisy samples. Note that the classifier is trained from scratch in this case, with the pre-trained denoiser already in place (but being fine-tuned as the classifier is trained).

We tested on CIFAR-10, at three different noise levels, without stability training. See Figure B.1 for results. Overall, we find that at high noise, there is still a significant gap in performance between Independent SSN and [5]'s method, using all of the denoising techniques. One possible

explanation is that it is also more difficult *for the denoiser* to learn the noise distribution of [5], compared to our distributions.

B.5 Additive and splitting noise allow for different types of joint noise distributions

In Section 3.3.2 in the main text, we showed that, in the $\lambda = 0.5$ case, SSN leads to marginal distributions which are simple affine transformations of the marginal distributions of the uniform additive smoothing noise (Equation 3.24). However, we also showed (Proposition 1) that, even in this case, certification is not possible using arbitrary joint distributions of ϵ with uniform additive noise, as it is with SSN. This difference is explained by the fact that, even for $\lambda = 0.5$, the joint distributions of $(\boldsymbol{x} + \epsilon)$ which can be generated by uniform additive noise and the joint distributions of $\tilde{\boldsymbol{x}}$ which can be generated by SSN respectively are in fact quite different.

To quantify this, consider a pair of two joint distributions: $\mathcal{D}$, with marginals uniform on $[-0.5, 0.5]$, and $\mathcal{S}$, with marginals uniform on $[0, 1]$. Let $\mathcal{D}$ and $\mathcal{S}$ be considered *equivalent* if, for $\epsilon \sim \mathcal{D}$ and $\boldsymbol{s} \sim \mathcal{S}$:

$$\tilde{\boldsymbol{x}} \sim (1/2)(\boldsymbol{x} + \epsilon) + \mathbb{1}/4 \quad \forall \boldsymbol{x} \tag{B.21}$$

where $\tilde{\boldsymbol{x}}$ is generated using the SSN noise $\boldsymbol{s}$ (compare to Equation 3.24 in the main text).

Proposition B.1. *The only pair of equivalent joint distributions* $(\mathcal{D}, \mathcal{S})$ *is* $\mathcal{D} \sim \mathcal{U}^d(-0.5, 0.5)$, $\mathcal{S} \sim \mathcal{U}^d(0, 1)$.

Proof. We first describe a special property of SSN (with $\lambda = 0.5$):

Fix a smoothed value $\tilde{\boldsymbol{x}}'$, and let $\mathcal{X}'$ be the set of all inputs $\boldsymbol{x}$ such that $\tilde{\boldsymbol{x}}'$ can be generated

from $\boldsymbol{x}$ under *any* joint splitting distribution $\mathcal{S}$. From Figure 3.2-a in the main text, we can see that this is simply

$$\mathcal{X}' = \{\boldsymbol{x} | \tilde{x}'_i \leq x_i/2 + (1/2) \leq \tilde{x}'_i + (1/2) \quad \forall i\}. \tag{B.22}$$

Notice that to generate $\tilde{\boldsymbol{x}}'$, *regardless of the value of* $\boldsymbol{x} \in \mathcal{X}'$, the splitting vector $\boldsymbol{s}$ must be exactly the following:

$$s_i = \begin{cases} 2\tilde{x}'_i & \text{if } \tilde{x}'_i < 1/2 \\ 2\tilde{x}'_i - 1 & \text{if } \tilde{x}'_i \geq 1/2 \end{cases} \tag{B.23}$$

(This is made clear by Figure 3.1 in the main text.)

If $\boldsymbol{x} \in \mathcal{X}'$, then $\tilde{\boldsymbol{x}}'$ will be generated iff this value of $\boldsymbol{s}$ is chosen. Therefore, given a fixed splitting distribution $\mathcal{S}$, the probability of generating $\tilde{\boldsymbol{x}}'$ must be *constant* for all points in $\mathcal{X}'$.

Now, we compare to uniform additive noise. In order for $\mathcal{D}$ and $\mathcal{S}$ to be equivalent, for the fixed noised point $(\boldsymbol{x} + \epsilon)' = 2\tilde{\boldsymbol{x}}' - \mathbb{1}/2$, it must be the case that all points in $\mathcal{X}'$ are equally likely to generate $(\boldsymbol{x} + \epsilon)'$. But note from Equation B.22 that $\mathcal{X}'$ is simply the uniform ℓ_∞ ball of radius 0.5 around $(\boldsymbol{x} + \epsilon)'$. This implies that $\mathcal{D}$ must be the uniform distribution $\mathcal{D} \sim \mathcal{U}^d(-0.5, 0.5)$, which is equivalent to the splitting distribution $\mathcal{S} \sim \mathcal{U}^d(0, 1)$. □

The *only* case when SSN and uniform additive noise can produce similar distributions of noisy samples is when all noise components are independent. This helps us understand how SSN can work with *any* joint distribution of splitting noise, while uniform additive noise has only been shown to produce accurate certificates when all components of ϵ are independent.

B.6 Complete Certification Data on CIFAR-10 and ImageNet

We provide complete certification results for uniform additive noise, randomized SSN with independent noise, and DSSN, at all tested noise levels on both CIFAR-10 and ImageNet, using both standard and stability training. For CIFAR-10, see Figures B.2, B.3, B.4, and B.5. For ImageNet, see Figure B.6. In Figure B.7 we compare the time required to certify each image for DSSN and Yang et al. [5]'s uniform random noise method, on both datasets.

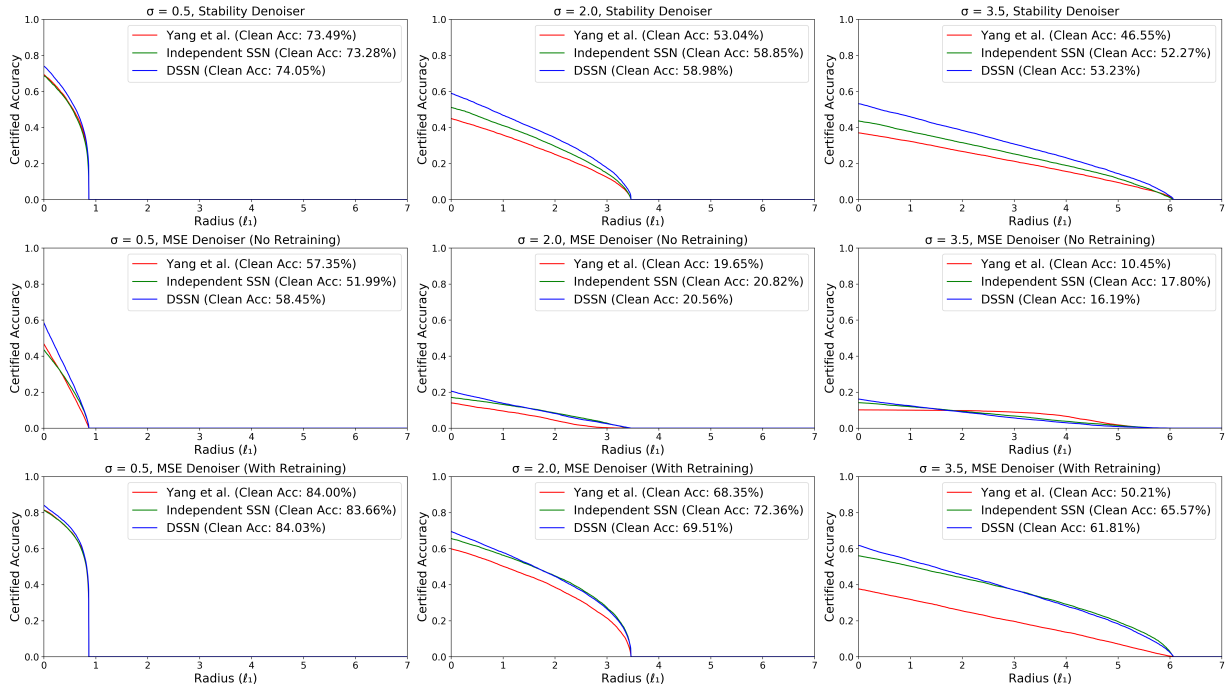


Figure B.1: Certified accuracies of models trained with denoisers, for additive uniform noise, SSN with independent noise, and DSSN. See text of Section B.4 for further details on the denoisers used. For $\sigma \geq 2.0$, Independent SSN outperfroms [5]'s method, suggesting that the difference in noise representations can not be resolved by using a denoiser. (It may appear as if [5]'s method is more robust at large radii for $\sigma = 3.5$ with an MSE denoiser without retraining: however, this is for a classifier with *clean accuracy* $\approx 10\%$, so this is vacuous: similar results can be achieved by simply always returning the same class.)

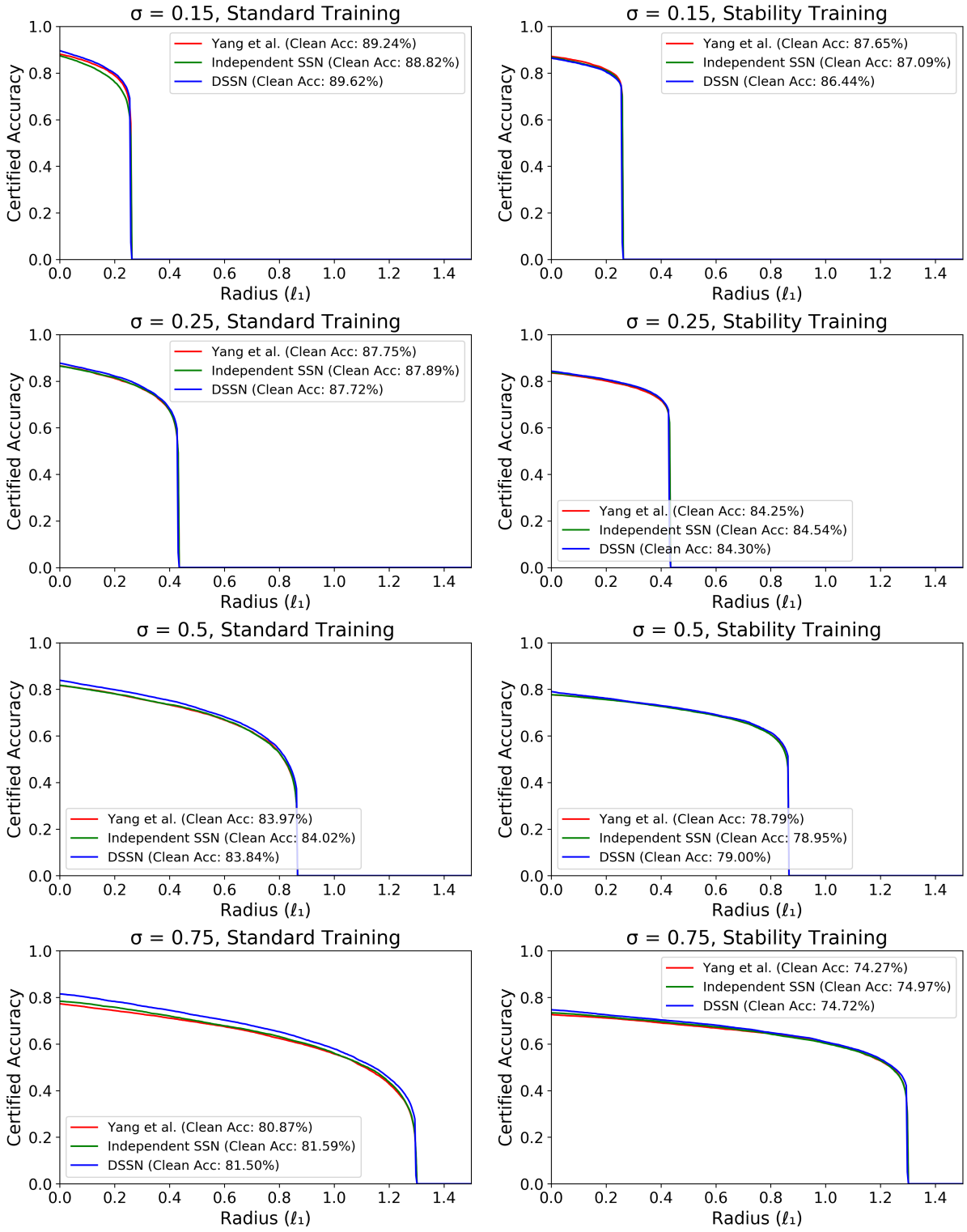


Figure B.2: Certification results for CIFAR-10, comparing uniform additive noise, randomized SSN with independent noise, and DSSN, for $\sigma \in \{0.15, 0.25, 0.5, 0.75\}$

.

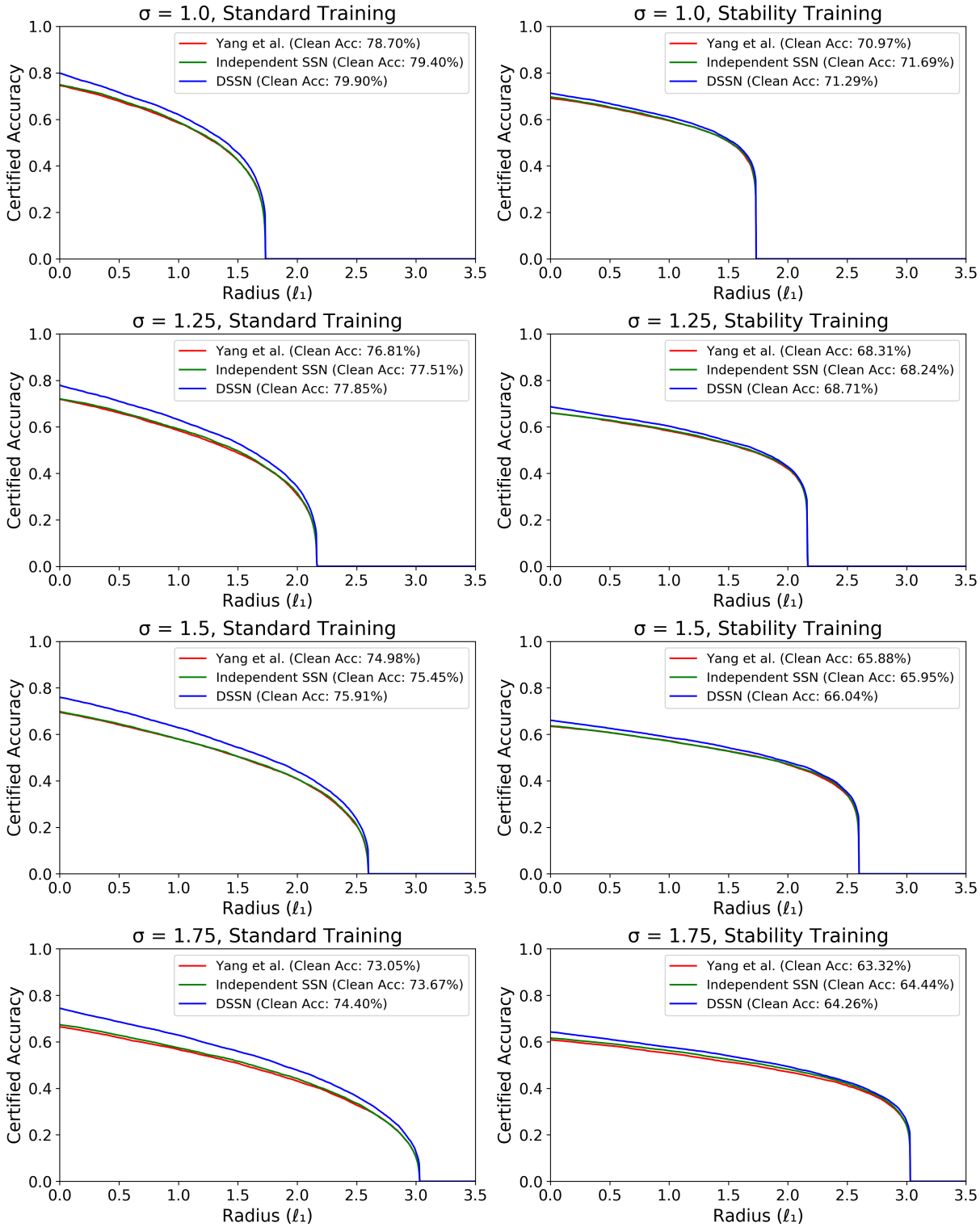


Figure B.3: Certification results for CIFAR-10, comparing uniform additive noise, randomized SSN with independent noise, and DSSN, for $\sigma \in \{1.0, 1.25, 1.5, 1.75\}$

.

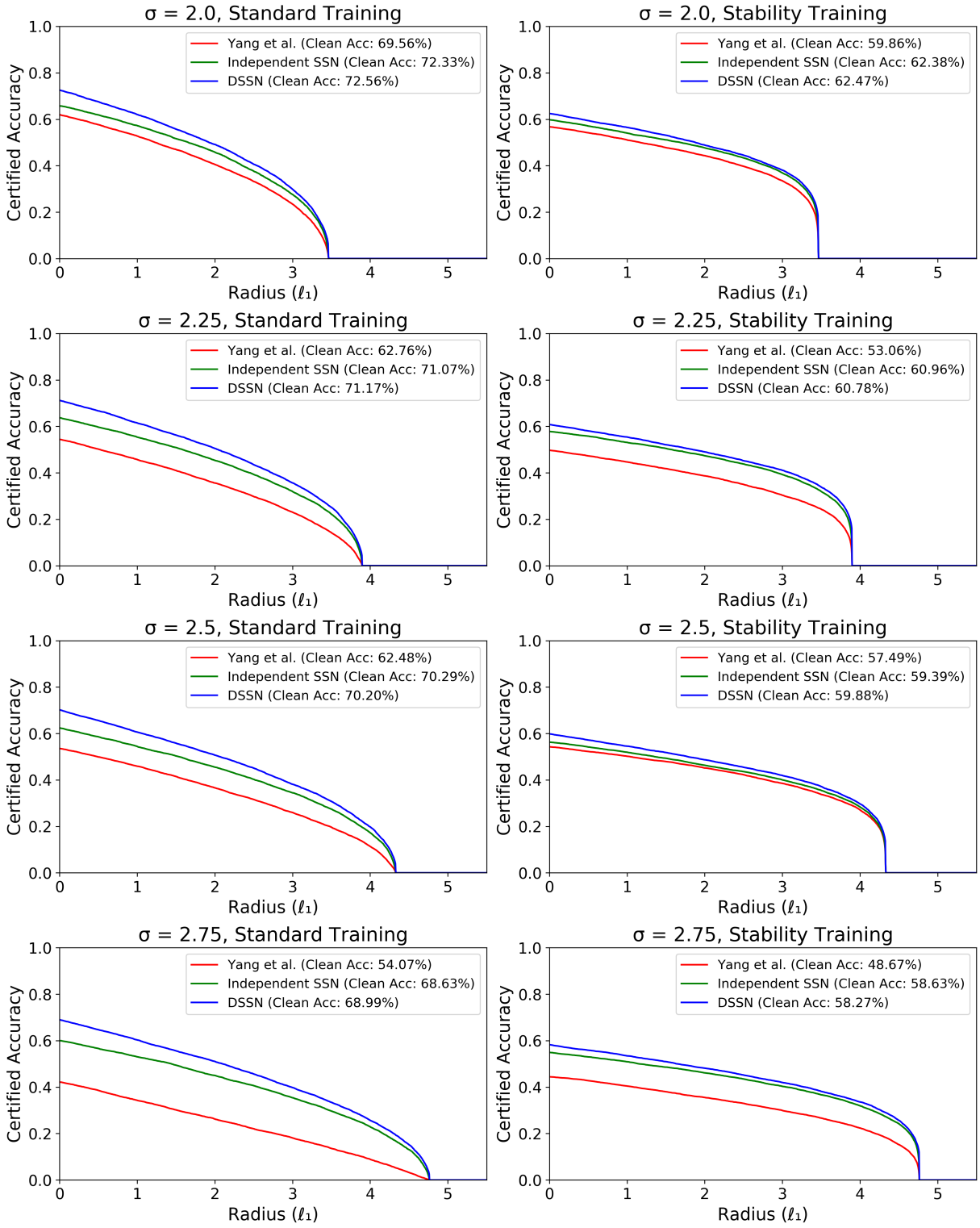


Figure B.4: Certification results for CIFAR-10, comparing uniform additive noise, randomized SSN with independent noise, and DSSN, for $\sigma \in \{2.0, 2.25, 2.5, 2.75\}$

.

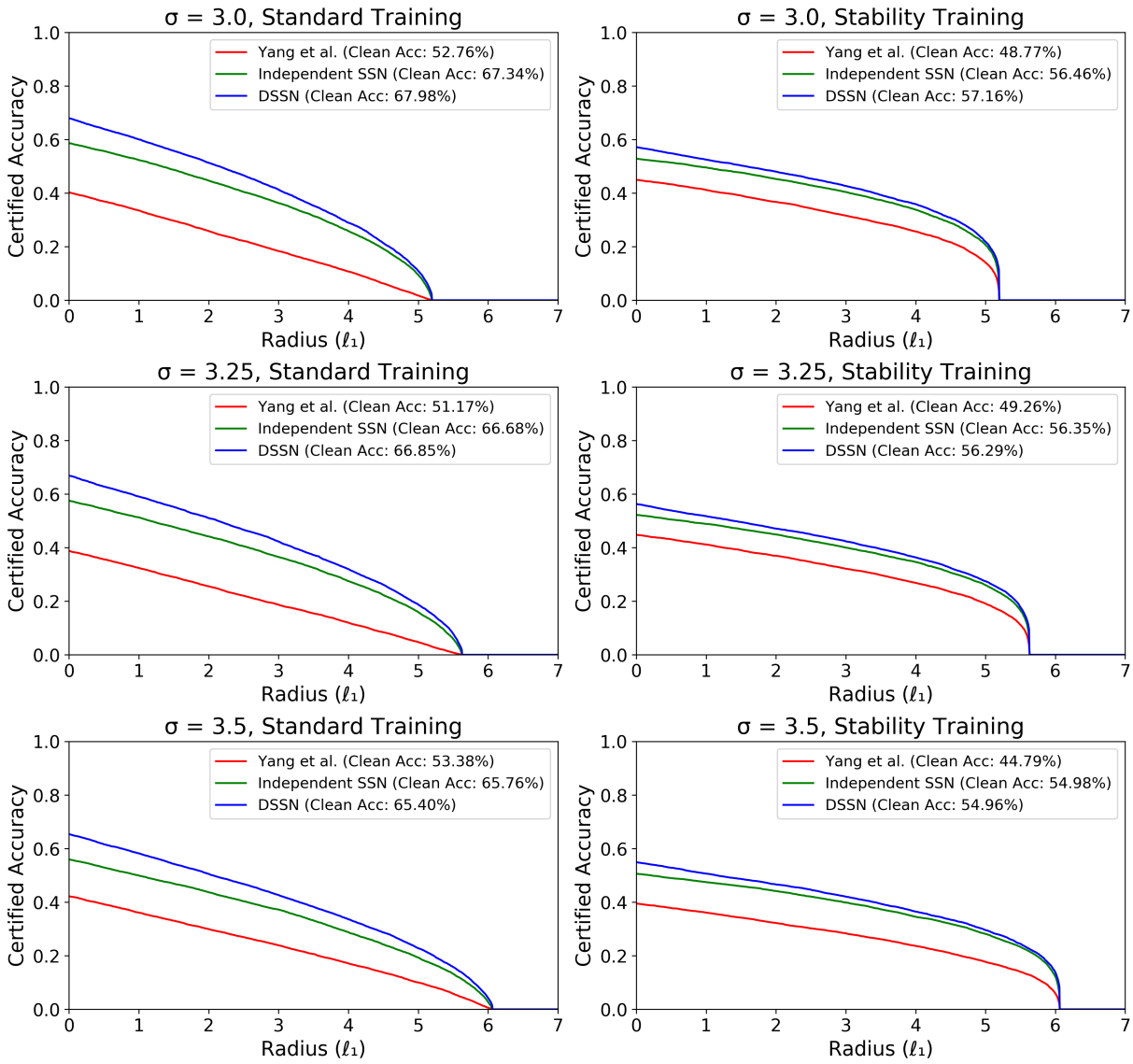


Figure B.5: Certification results for CIFAR-10, comparing uniform additive noise, randomized SSN with independent noise, and DSSN, for $\sigma \in \{3.0, 3.25, 3.5\}$

.

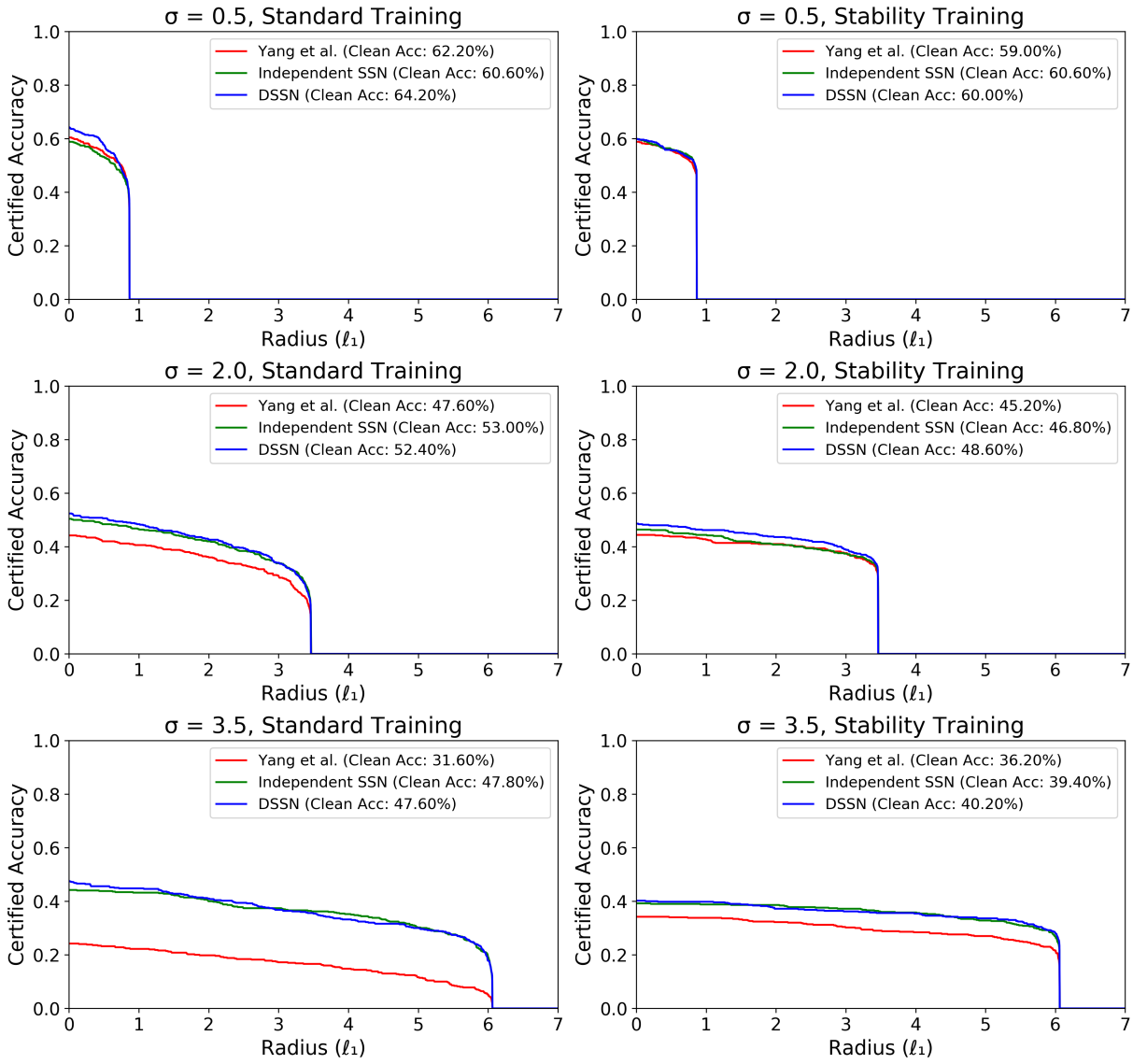


Figure B.6: Certification results for ImageNet, comparing uniform additive noise, randomized SSN with independent noise, and DSSN, for $\sigma \in \{0.5, 2.0, 3.5\}$. Note that we see less improvement in reported certified accuracies due to derandomization (i.e., less difference between Independent SSN and DSSN) in ImageNet compared to in CIFAR-10, particularly at large noise levels.

.

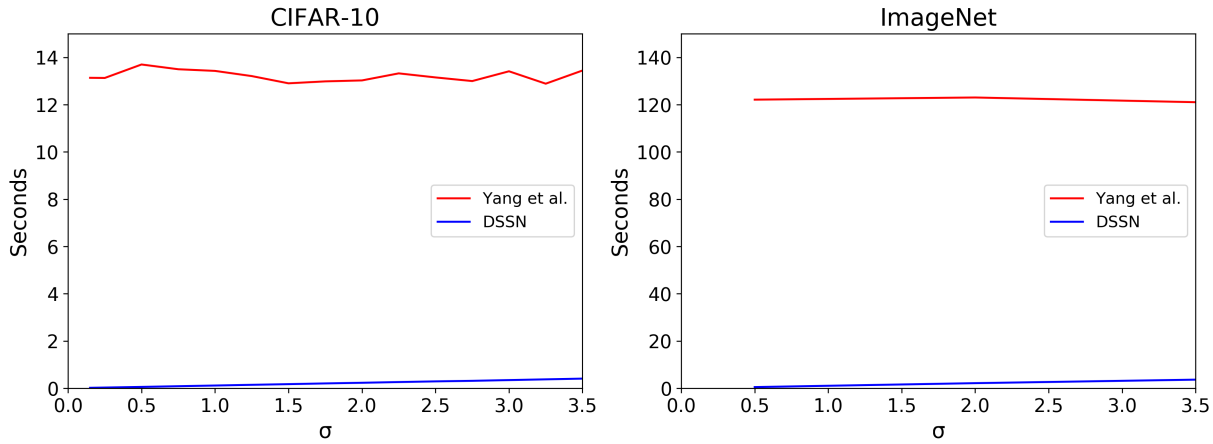


Figure B.7: Comparison of the certification time per image of DSSN and Yang et al. [5]'s uniform additive noise method. We used a single NVIDIA 2080 Ti GPU. Although in contrast to [5], our certification time scales linearly with the noise level, the fact that [5] uses 100,000 smoothing samples makes our method much faster even at the largest tested noise levels.

Appendix C: Appendix to Chapter 4

C.1 Proofs

C.1.1 Proof Sketch of Theorem 4.1 (from Chapter 3) using modified notation

Suppose the $[0, 1]$ domain of dimension i is divided into "bins", with dividers at each value $s_i + n\Lambda, \forall n \in \mathbb{N}$ (See Figure 4.1 in the main text) Then x_i^{lower} and x_i^{upper} are the lower- and upper-limits of the bin which x_i is assigned to. Note that the bins are of size Λ, and that the offset s_i of the dividers is uniformly random. Consider two points $\boldsymbol{x}$ and $\boldsymbol{y}$, and let $\delta_i := |x_i - y_i|$. Then the probability of a divider separating x_i and y_i is $\min(\delta_i/\Lambda, 1)$. By union bound, the probability that $(\boldsymbol{y}^{\text{lower}}, \boldsymbol{y}^{\text{upper}})$ and $(\boldsymbol{x}^{\text{lower}}, \boldsymbol{x}^{\text{upper}})$ differ at all is at most $\Sigma\delta_i/\Lambda = \|x - y\|_1/\Lambda$. If $\boldsymbol{x}$ and $\boldsymbol{y}$ are mapped to the same bin in every dimension, $f(\boldsymbol{x}^{\text{lower}}, \boldsymbol{x}^{\text{upper}}) = f(\boldsymbol{y}^{\text{lower}}, \boldsymbol{y}^{\text{upper}})$. Because the range of f is restricted to the interval $[0, 1]$, this implies that $|p(\boldsymbol{x}) - p(\boldsymbol{y})| = |\mathbb{E}_{\boldsymbol{s}}[f(\boldsymbol{x}^{\text{lower}}, \boldsymbol{x}^{\text{upper}})] - \mathbb{E}_{\boldsymbol{s}}[f(\boldsymbol{y}^{\text{lower}}, \boldsymbol{y}^{\text{upper}})]| \leq \|x - y\|_1/\Lambda$.

C.1.2 Proof of Theorem 4.2

We first need the following lemma, which is implicit in the proofs in Chapter 3, but which we prove explicitly here for completeness. Note that we closely follow the proof of Theorem 3.2 in Chapter 3

Lemma C.1. *For any* $\Lambda_i \in (0,1] \cup \{\infty\}$*, let* $s_i \sim \mathcal{U}(0,\Lambda_i)$*. For any* $x_i, y_i \in [0,1]$*, let* $\delta_i := |x_i - y_i|$ *and define* x_i^{upper}*,* x_i^{lower} *as follows: If* $\Lambda_i = \infty$*, then* $x_i^{upper} := 1$*,* $x_i^{lower} := 0$*, otherwise:*

$$x_i^{upper} := \min(\Lambda_i \lceil \frac{x_i - s_i}{\Lambda_i} \rceil + s_i, 1) \tag{C.1}$$

$$x_i^{lower} := \max(\Lambda_i \lceil \frac{x_i - s_i}{\Lambda_i} \rceil + s_i - \Lambda_i, 0) \tag{C.2}$$

and define y_i^{upper}*,* y_i^{lower} *similarly. Then:*

$$\Pr_{s_i}((x_i^{lower}, x_i^{upper}) \neq (y_i^{lower}, y_i^{upper})) = \min\left(\frac{\delta_i}{\Lambda_i}, 1\right) \tag{C.3}$$

Proof. We first assume $\Lambda_i \in (0,1]$. Without loss of generality, assume $x_i \geq y_i$, so that $\delta_i = x_i - y_i$.

Note that, with probability 1, $(x_i^{\text{lower}}, x_i^{\text{upper}}) \neq (y_i^{\text{lower}}, y_i^{\text{upper}})$ iff $\lceil \frac{x_i - s_i}{\Lambda_i} \rceil \neq \lceil \frac{y_i - s_i}{\Lambda_i} \rceil$.

To see this, note that $\lceil \frac{x_i - s_i}{\Lambda_i} \rceil = \lceil \frac{y_i - s_i}{\Lambda_i} \rceil \implies (x_i^{\text{lower}}, x_i^{\text{upper}}) = (y_i^{\text{lower}}, y_i^{\text{upper}})$ directly from the definitions.

For the converse, $\lceil \frac{x_i - s_i}{\Lambda_i} \rceil \neq \lceil \frac{y_i - s_i}{\Lambda_i} \rceil \implies (x_i^{\text{lower}}, x_i^{\text{upper}}) \neq (y_i^{\text{lower}}, y_i^{\text{upper}})$, first consider the case where $\Lambda_i < 1$. Because the first terms in the "min" or "max" of the definitions of x_i^{lower} and x_i^{upper} differ by a most $\Lambda_i < 1$, both of the $[0,1]$ box constraints cannot be active simultaneously: either $x_i^{\text{lower}} = \Lambda_i \lceil \frac{x_i - s_i}{\Lambda_i} \rceil + s_i - \Lambda_i$ (and not 0) and/or $x_i^{\text{upper}} = \Lambda_i \lceil \frac{x_i - s_i}{\Lambda_i} \rceil + s_i$ (and not 1). Therefore if $\lceil \frac{x_i - s_i}{\Lambda_i} \rceil \neq \lceil \frac{y_i - s_i}{\Lambda_i} \rceil$, whichever of x_i^{upper} or x_i^{lower} is not affected by the box constraint will necessarily differ from y_i^{upper} or y_i^{lower}, so $(x_i^{\text{lower}}, x_i^{\text{upper}}) \neq (y_i^{\text{lower}}, y_i^{\text{upper}})$. For the case where $\Lambda_i = 1$, both constraints can only be simultaneously active if $s_i = 0$ or $s_i = 1$, which both occur with probability zero, and otherwise the same argument from the $\Lambda_i < 1$ case applies.

Therefore, it is sufficient to show that

$$\Pr_{s_i}(\lceil \frac{x_i - s_i}{\Lambda_i} \rceil \neq \lceil \frac{y_i - s_i}{\Lambda_i} \rceil) = \min\left(\frac{\delta_i}{\Lambda_i}, 1\right) \tag{C.4}$$

First, if $\delta_i/\Lambda_i \geq 1$, then $\frac{x_i - s_i}{\Lambda_i}$ and $\frac{y_i - s_i}{\Lambda_i}$ differ by at least one, so their ceilings must differ. Then $\Pr_{s_i}(\lceil \frac{x_i - s_i}{\Lambda_i} \rceil \neq \lceil \frac{y_i - s_i}{\Lambda_i} \rceil) = 1 = \min\left(\frac{\delta_i}{\Lambda_i}, 1\right)$.

Otherwise, if $\delta_i/\Lambda_i < 1$, then $\frac{x_i}{\Lambda_i}$ and $\frac{y_i}{\Lambda_i}$ differ by less than one, so

$$\lceil \frac{x_i}{\Lambda_i} \rceil - \lceil \frac{y_i}{\Lambda_i} \rceil \in \{0, 1\} \tag{C.5}$$

And similarly:

$$\lceil \frac{x_i - s_i}{\Lambda_i} \rceil - \lceil \frac{y_i - s_i}{\Lambda_i} \rceil \in \{0, 1\} \tag{C.6}$$

Also, because s_i/Λ_i is at most one,

$$\begin{aligned} \lceil \frac{x_i}{\Lambda_i} \rceil - \lceil \frac{x_i - s_i}{\Lambda_i} \rceil \in \{0, 1\} \\ \lceil \frac{y_i}{\Lambda_i} \rceil - \lceil \frac{y_i - s_i}{\Lambda_i} \rceil \in \{0, 1\} \end{aligned} \tag{C.7}$$

We consider cases on $\lceil \frac{x_i}{\Lambda_i} \rceil - \lceil \frac{y_i}{\Lambda_i} \rceil$:

- Case $\lceil \frac{x_i}{\Lambda_i} \rceil - \lceil \frac{y_i}{\Lambda_i} \rceil = 0$. Then $\lceil \frac{x_i - s_i}{\Lambda_i} \rceil = \lceil \frac{y_i - s_i}{\Lambda_i} \rceil$ only in two cases:
  - $\lceil \frac{x_i - s_i}{\Lambda_i} \rceil = \lceil \frac{y_i - s_i}{\Lambda_i} \rceil = \lceil \frac{x_i}{\Lambda_i} \rceil$ iff $\frac{s_i}{\Lambda_i} < \frac{y_i}{\Lambda_i} - (\lceil \frac{y_i}{\Lambda_i} \rceil - 1)$ $(\leq \frac{x_i}{\Lambda_i} - (\lceil \frac{x_i}{\Lambda_i} \rceil - 1))$.
  - $\lceil \frac{y_i - s_i}{\Lambda_i} \rceil = \lceil \frac{x_i - s_i}{\Lambda_i} \rceil = \lceil \frac{x_i}{\Lambda_i} \rceil - 1$ iff $\frac{s_i}{\Lambda_i} > \frac{x_i}{\Lambda_i} - (\lceil \frac{x_i}{\Lambda_i} \rceil - 1)$ $(\geq \frac{y_i}{\Lambda_i} - (\lceil \frac{y_i}{\Lambda_i} \rceil - 1))$.

  Then $\lceil \frac{y_i - s_i}{\Lambda_i} \rceil \neq \lceil \frac{x_i - s_i}{\Lambda_i} \rceil$ iff $\frac{y_i}{\Lambda_i} - (\lceil \frac{x_i}{\Lambda_i} \rceil - 1) < \frac{s_i}{\Lambda_i} < \frac{x_i}{\Lambda_i} - (\lceil \frac{x_i}{\Lambda_i} \rceil - 1)$. There are exactly $q(x_i - y_i) =$

$q\delta_i$ values of s_i for which this occurs, out of a total $\Lambda_i q$ values of s_i, so this occurs with probability $\frac{\delta_i}{\Lambda_i}$.

- Case $\lceil \frac{x_i}{\Lambda_i} \rceil - \lceil \frac{y_i}{\Lambda_i} \rceil = 1$. Then $\lceil \frac{x_i - s_i}{\Lambda_i} \rceil \neq \lceil \frac{y_i - s_i}{\Lambda_i} \rceil$ only in two cases:
  - $\lceil \frac{y_i - s_i}{\Lambda_i} \rceil = \lceil \frac{y_i}{\Lambda_i} \rceil$ and $\lceil \frac{x_i - s_i}{\Lambda_i} \rceil = \lceil \frac{x_i}{\Lambda_i} \rceil = \lceil \frac{y_i}{\Lambda_i} \rceil + 1$. This happens iff $\frac{s_i}{\Lambda_i} < \frac{x_i}{\Lambda_i} - \lceil \frac{y_i}{\Lambda_i} \rceil$ ($\leq \frac{y_i}{\Lambda_i} - (\lceil \frac{y_i}{\Lambda_i} \rceil - 1)$).
  - $\lceil \frac{y_i - s_i}{\Lambda_i} \rceil = \lceil \frac{y_i}{\Lambda_i} \rceil - 1$ and $\lceil \frac{x_i - s_i}{\Lambda_i} \rceil = \lceil \frac{y_i}{\Lambda_i} \rceil$. This happens iff $\frac{s_i}{\Lambda_i} > \frac{y_i}{\Lambda_i} - (\lceil \frac{y_i}{\Lambda_i} \rceil - 1)$ ($\geq \frac{x_i}{\Lambda_i} - \lceil \frac{y_i}{\Lambda_i} \rceil$).

  Therefore, $\lceil \frac{y_i - s_i}{\Lambda_i} \rceil = \lceil \frac{x_i - s_i}{\Lambda_i} \rceil$ iff:

$$\frac{x_i}{\Lambda_i} - \lceil \frac{y_i}{\Lambda_i} \rceil < \frac{s_i}{\Lambda_i} < \frac{y_i}{\Lambda_i} - (\lceil \frac{y_i}{\Lambda_i} \rceil - 1) \tag{C.8}$$

  Which is:

$$\frac{y_i}{\Lambda_i} - \lceil \frac{y_i}{\Lambda_i} \rceil + \frac{\delta_i}{\Lambda_i} < \frac{s_i}{\Lambda_i} < \frac{y_i}{\Lambda_i} - \lceil \frac{y_i}{\Lambda_i} \rceil + 1. \tag{C.9}$$

  There are exactly $q(1 - (x_i - y_i)) = q(1 - \delta_i)$ values of s_i for which this occurs, out of a total $\Lambda_i q$ values of s_i, so this occurs with probability $1 - \frac{\delta_i}{\Lambda_i}$. Then $\lceil \frac{y_i - s_i}{\Lambda_i} \rceil \neq \lceil \frac{x_i - s_i}{\Lambda_i} \rceil$ with probability $\frac{\delta_i}{\Lambda_i}$.

So in all cases, for $\delta_i / \Lambda_i < 1$, $\Pr_{s_i}(\lceil \frac{x_i - s_i}{\Lambda_i} \rceil \neq \lceil \frac{y_i - s_i}{\Lambda_i} \rceil) = \frac{\delta_i}{\Lambda_i} = \min\left(\frac{\delta_i}{\Lambda_i}, 1\right)$.

Finally, we consider $\Lambda_i = \infty$. In this case, $(x_i^{\text{lower}}, x_i^{\text{upper}}) = (y_i^{\text{lower}}, y_i^{\text{upper}})) = (0, 1)$ with probability 1, so

$$\Pr_{s_i}((x_i^{\text{lower}}, x_i^{\text{upper}}) \neq (y_i^{\text{lower}}, y_i^{\text{upper}})) = 0 = \frac{\delta_i}{\infty} = \min\left(\frac{\delta_i}{\Lambda_i}, 1\right) \tag{C.10}$$

□

We can now prove each part of the theorem:

Part 1. *Let $\mathcal{D}$ and $f(\cdot)$ be the Λ-distribution and base function used for Variable-Λ smoothing, respectively. Let $\boldsymbol{x}, \boldsymbol{y} \in [0,1]^d$ be two points. For each dimension i, let $\delta_i := |x_i - y_i|$. The probability that $(x_i^{lower}, x_i^{upper}) \neq (y_i^{lower}, y_i^{upper})$ is given by $\Pr_i^{split}(\delta_i)$, where:*

$$\Pr_i^{split}(z) := \Pr_{\mathcal{D}_i}(\Lambda_i \leq z) + z\mathbb{E}_{\mathcal{D}_i}\left[\frac{\mathbf{1}_{(\Lambda_i \in (z,1])}}{\Lambda_i}\right] \tag{C.11}$$

Proof.

$$\begin{aligned}
\Pr((x_i^{\text{lower}}, x_i^{\text{upper}}) \neq (y_i^{\text{lower}}, y_i^{\text{upper}})) &= \\
\mathbb{E}[\mathbf{1}_{(x_i^{\text{lower}}, x_i^{\text{upper}}) \neq (y_i^{\text{lower}}, y_i^{\text{upper}})}] &= \\
\mathbb{E}_{\Lambda_i \sim \mathcal{D}_i}[\mathbb{E}[\mathbf{1}_{(x_i^{\text{lower}}, x_i^{\text{upper}}) \neq (y_i^{\text{lower}}, y_i^{\text{upper}})} | \Lambda_i]] &= \\
\mathbb{E}_{\Lambda_i \sim \mathcal{D}_i}[\Pr[(x_i^{\text{lower}}, x_i^{\text{upper}}) \neq (y_i^{\text{lower}}, y_i^{\text{upper}}) | \Lambda_i]] &= \\
\mathbb{E}_{\Lambda_i \sim \mathcal{D}_i}[\min\left(\frac{\delta_i}{\Lambda_i}, 1\right)] &= \\
\mathbb{E}_{\Lambda_i \sim \mathcal{D}_i}[\frac{\delta_i}{\Lambda_i} \cdot \mathbf{1}_{\Lambda_i > \delta_i}] + \mathbb{E}_{\Lambda_i \sim \mathcal{D}_i}[1 \cdot \mathbf{1}_{\Lambda_i \leq \delta_i}] &= \\
\mathbb{E}_{\Lambda_i \sim \mathcal{D}_i}[\frac{\delta_i}{\Lambda_i} \cdot \mathbf{1}_{\Lambda_i > \delta_i}] + \mathbb{E}_{\Lambda_i \sim \mathcal{D}_i}[1 \cdot \mathbf{1}_{\Lambda_i \leq \delta_i}] &= \\
\delta_i \mathbb{E}_{\mathcal{D}_i}\left[\frac{\mathbf{1}_{(\Lambda_i \in (\delta_i, 1])}}{\Lambda_i}\right] + \Pr_{\mathcal{D}_i}(\Lambda_i \leq \delta_i) = \Pr_i^{\text{split}}(\delta_i)
\end{aligned} \tag{C.12}$$

Where we use the law of total expectation in the third line, Lemma C.1 in the fifth line, and in the last line, we use that δ_i is finite, so $\delta_i/\infty = 0$ □

Part 2. *Let $d(\cdot,\cdot)$ be an ECM defined by concave functions $g_1, ..., g_d$. Let $\mathcal{D}$ and $f(\cdot)$ be the Λ-distribution and base function used for Variable-Λ smoothing, respectively. If $\forall i \in [d]$ and*

$\forall z \in [0,1]$,

$$\Pr_i^{split}(z) \leq g_i(z), \tag{C.13}$$

then, the smoothed function $p_{\mathcal{D},f}(\cdot)$ is 1-Lipschitz with respect to the metric $d(\cdot,\cdot)$.

Proof. Let $\boldsymbol{x}, \boldsymbol{y} \in [0,1]^d$ be two points. For each dimension i, let $\delta_i := |x_i - y_i|$. By union bound:

$$\begin{aligned}
\Pr_{\boldsymbol{s}}((\boldsymbol{x}^{\text{lower}}, \boldsymbol{x}^{\text{upper}}) \neq (\boldsymbol{y}^{\text{lower}}, \boldsymbol{y}^{\text{upper}})) = \\
\Pr_{\boldsymbol{s}}\left[\bigcup_{i=1}^{d}(x_i^{\text{lower}}, x_i^{\text{upper}}) \neq (y_i^{\text{lower}}, y_i^{\text{upper}})\right] \leq \\
\sum_{i=1}^{d} \Pr_i^{\text{split}}(\delta_i) \leq \\
\sum_{i=1}^{d} g_i(\delta_i) = d(\boldsymbol{x}, \boldsymbol{y})
\end{aligned} \tag{C.14}$$

Then:

$$\begin{aligned}
&|p_{\mathcal{D},f}(\boldsymbol{x}) - p_{\mathcal{D},f}(\boldsymbol{y})| \\
&= \left|\mathop{\mathbb{E}}_{\boldsymbol{s}}\left[f(\boldsymbol{x}^{\text{lower}}, \boldsymbol{x}^{\text{upper}})\right] - \mathop{\mathbb{E}}_{\boldsymbol{s}}\left[f(\boldsymbol{y}^{\text{lower}}, \boldsymbol{y}^{\text{upper}})\right]\right| \\
&= \left|\mathop{\mathbb{E}}_{\boldsymbol{s}}\left[f(\boldsymbol{x}^{\text{lower}}, \boldsymbol{x}^{\text{upper}}) - f(\boldsymbol{y}^{\text{lower}}, \boldsymbol{y}^{\text{upper}})\right]\right| \\
&= \Bigg|\Pr_{\boldsymbol{s}}((\boldsymbol{x}^{\text{lower}}, \boldsymbol{x}^{\text{upper}}) \neq (\boldsymbol{y}^{\text{lower}}, \boldsymbol{y}^{\text{upper}}))\mathop{\mathbb{E}}_{\boldsymbol{s}}\left[f(\boldsymbol{x}^{\text{lower}}, \boldsymbol{x}^{\text{upper}}) - f(\boldsymbol{y}^{\text{lower}}, \boldsymbol{y}^{\text{upper}})|(\boldsymbol{x}^{\text{lower}}, \boldsymbol{x}^{\text{upper}}) \neq (\boldsymbol{y}^{\text{lower}}, \boldsymbol{y}^{\text{upper}})\right] \\
&+ \Pr_{\boldsymbol{s}}((\boldsymbol{x}^{\text{lower}}, \boldsymbol{x}^{\text{upper}}) = (\boldsymbol{y}^{\text{lower}}, \boldsymbol{y}^{\text{upper}}))\left[f(\boldsymbol{x}^{\text{lower}}, \boldsymbol{x}^{\text{upper}}) - f(\boldsymbol{y}^{\text{lower}}, \boldsymbol{y}^{\text{upper}})|(\boldsymbol{x}^{\text{lower}}, \boldsymbol{x}^{\text{upper}}) = (\boldsymbol{y}^{\text{lower}}, \boldsymbol{y}^{\text{upper}})\right]\Bigg|
\end{aligned} \tag{C.15}$$

Because $\mathbb{E}_{\boldsymbol{s}}\left[f(\boldsymbol{x}^{\text{lower}}, \boldsymbol{x}^{\text{upper}}) - f(\boldsymbol{y}^{\text{lower}}, \boldsymbol{y}^{\text{upper}})|(\boldsymbol{x}^{\text{lower}}, \boldsymbol{x}^{\text{upper}}) = (\boldsymbol{y}^{\text{lower}}, \boldsymbol{y}^{\text{upper}})\right]$ is zero, we have:

$$
\begin{aligned}
&|p_{\mathcal{D},f}(\boldsymbol{x}) - p_{\mathcal{D},f}(\boldsymbol{y})| \\
&= \Pr_{\boldsymbol{s}}((\boldsymbol{x}^{\text{lower}}, \boldsymbol{x}^{\text{upper}}) \neq (\boldsymbol{y}^{\text{lower}}, \boldsymbol{y}^{\text{upper}})) \left| \mathop{\mathbb{E}}_{\boldsymbol{s}} \left[f(\boldsymbol{x}^{\text{lower}}, \boldsymbol{x}^{\text{upper}}) - f(\boldsymbol{y}^{\text{lower}}, \boldsymbol{y}^{\text{upper}}) | (\boldsymbol{x}^{\text{lower}}, \boldsymbol{x}^{\text{upper}}) \neq (\boldsymbol{y}^{\text{lower}}, \boldsymbol{y}^{\text{upper}}) \right] \right| \\
&\leq d(\boldsymbol{x}, \boldsymbol{y}) \cdot 1
\end{aligned}
\tag{C.16}
$$

In the last step, we use Equation C.14 and the assumption that $f(\cdot,\cdot) \in [0,1]$. Therefore, by the definition of Lipschitz-continuity, $p_{\mathcal{D},f}$ is 1-Lipschitz with respect to $d(\cdot,\cdot)$.

□

Part 3. *Suppose g_i is continuous and twice-differentiable on the interval $(0,1]$. Let $\mathcal{D}_i$ be constructed as follows:*

- *On the interval $(0,1)$, Λ_i is distributed continuously, with pdf function:*

$$
pdf_{\Lambda_i}(z) = -z g_i''(z) \tag{C.17}
$$

- $\Pr(\Lambda_i = 1) = g_i'(1)$
- $\Pr(\Lambda_i = \infty) = 1 - g_i(1)$

then,

$$
\Pr_i^{split}(z) = g_i(z) \quad \forall z \in [0,1]. \tag{C.18}
$$

If all $\mathcal{D}_i$ are constructed this way, then the conclusion of part (b) above applies.

Proof. We first show that this is in fact a normalized probability distribution:

$$\begin{aligned}
\int_0^1 \text{pdf}_{\Lambda_i}(z)dz + \Pr(\Lambda_i = 1) + \Pr(\Lambda_i = \infty) = \\
\int_0^1 -zg_i''(z)dz + g_i'(1) + 1 - g_i(1) = \\
-\left(1 \cdot g_i'(1) - 0 \cdot g_i'(0) - \int_0^1 1 \cdot g_i'(z)dz\right) + g_i'(1) + 1 - g_i(1) = \\
-g_i'(1) + \int_0^1 g_i'(z)dz + g_i'(1) + 1 - g_i(1) = \\
g_i(1) - g_i(0) + 1 - g_i(1) = 1
\end{aligned} \tag{C.19}$$

Where we use integration by parts in the third line, and the fact that $g_i(0) = 0$ in the last line.

We now show that $\Pr_i^{\text{split}}(z) = g_i(z)$ in the special case of $z = 1$:

$$\begin{aligned}
\Pr_i^{\text{split}}(1) &= \Pr_{\mathcal{D}_i}(\Lambda_i \leq 1) + 1\mathbb{E}_{\mathcal{D}_i}\left[\frac{\mathbf{1}_{(\Lambda_i \in (1,1])}}{\Lambda_i}\right] \\
&= \Pr_{\mathcal{D}_i}(\Lambda_i \leq 1) \\
&= 1 - \Pr_{\mathcal{D}_i}(\Lambda_i = \infty) \\
&= 1 - (1 - g_i(1)) = g_i(1)
\end{aligned} \tag{C.20}$$

Where in the second line, we use that $(1, 1]$ represents the empty set, so the term in the expectation is always zero.

Now, we handle the remaining case of $z \in [0, 1)$:

$$\begin{aligned}
\Pr_i^{\text{split}}(z) &= \Pr_{\mathcal{D}_i}(\Lambda_i \le z) + z\mathbb{E}_{\mathcal{D}_i}\left[\frac{\mathbf{1}_{(\Lambda_i \in (z,1])}}{\Lambda_i}\right] \\
&= \int_0^z \text{pdf}_{\Lambda_i}(w)dw + z\left[\int_z^1 \text{pdf}_{\Lambda_i}(w) \cdot \frac{1}{w}dw + \Pr(\Lambda = 1)\frac{1}{1}\right] \\
&= \int_0^z -wg_i''(w)dw + z\left[\int_z^1 -wg_i''(w) \cdot \frac{1}{w}dw + g_i'(1)\right] \\
&= -\left[zg_i'(z) - 0 \cdot g_i'(0) - \int_0^z 1 \cdot g_i'(w)dw\right] + z\left[-\int_z^1 g_i''(w)dw + g_i'(1)\right] \\
&= -\left[zg_i'(z) - (g_i(z) - g_i(0))\right] + z\left[-[g_i'(1) - g_i'(z)] + g_i'(1)\right] \\
&= -zg_i'(z) + g_i(z) - zg_i'(1) + zg_i'(z) + zg_i'(1) = g_i(z)
\end{aligned} \tag{C.21}$$

Where we use integration by parts in the fourth line, and the fact that $g_i(0) = 0$ in the last line.

Now we have that $\Pr_i^{\text{split}}(z) = g_i(z)\ \forall z \in [0, 1]$, as desired. The final statement follows directly from Part b. □

C.1.3 Proof of Corollary 4.1

Corollary 4.1. *For all $p \in (0, 1]$, $\alpha \in [1, \infty)$, if we perform Variable-Λ smoothing with all Λ_i's distributed identically (but not necessarily independently) as follows:*

$$\begin{aligned}
\Lambda_i &\sim \textit{Beta}(p, 1), \ \textit{with prob.}\ \frac{1-p}{\alpha} \\
\Lambda_i &= 1, \ \textit{with prob.}\ \frac{p}{\alpha} \\
\Lambda_i &= \infty, \ \textit{with prob.}\ 1 - \frac{1}{\alpha}
\end{aligned} \tag{C.22}$$

then, the resulting smoothed function will be $1/\alpha$-Lipschitz with respect to the ℓ_p^p metric

Proof. We consider the ECM defined as $\forall i,\ g_i(z) = \frac{z^p}{\alpha}$. One can easily verify that this is a valid

ECM, and that it is twice-differentiable on $(0, 1]$.

We then apply Theorem 4.2-c:

- On the interval $(0, 1)$, we distribute Λ_i continuously, with pdf function:

$$\text{pdf}_{\Lambda_i}(z) = -zg_i''(z) = \frac{-p(p-1)z^{p-1}}{\alpha} = \frac{1-p}{\alpha} \cdot pz^{p-1} = \frac{1-p}{\alpha} \cdot \text{pdf}_{\text{Beta}(p,1)}(z) \tag{C.23}$$

- $\Pr(\Lambda_i = 1) = g_i'(1) = \frac{p \cdot 1^{p-1}}{\alpha} = \frac{p}{\alpha}$

- $\Pr(\Lambda_i = \infty) = 1 - g_i(1) = 1 - \frac{1}{\alpha}$

So distributing Λ as stated in the Corollary will result in $\Pr_i^{\text{split}}(z) = g_i(z)\ \forall z \in [0, 1]$, and therefore the resulting smoothed function will be 1-Lipschitz w.r.t. the ECM. Then, from the definition of Lipschitzness and of the ECM, we have, for all $\boldsymbol{x}$, $\boldsymbol{y}$:

$$|p_{\mathcal{D},f}(\boldsymbol{x}) - p_{\mathcal{D},f}(\boldsymbol{y})| \leq \sum_{i=1}^{d} \frac{|x_i - y_i|^p}{\alpha} = \frac{1}{\alpha} \ell_p^p(\boldsymbol{x}, \boldsymbol{y}) \tag{C.24}$$

So $p_{\mathcal{D},f}$ is also $1/\alpha$-Lipschitz w.r.t. the ℓ_p^p metric. □

C.1.3.1 $\alpha < 1$ Case for Corollary 4.1

In a footnote in the main text, we mentioned that this technique cannot be applied directly to the $\alpha < 1$ case. To explain, note that taking

$$g_i(z) := \frac{z^p}{\alpha}, \quad \forall i \tag{C.25}$$

with $\alpha < 1$ is not a properly-defined ECM, because $g_i \notin [0,1] \to [0,1]$: for example, $g_i(1) = 1/\alpha > 1$. However, for the purpose of building a Lipschitz classifier with range $[0,1]$, we can instead define:

$$g_i(z) \coloneqq \min(\frac{z^p}{\alpha}, 1), \quad \forall i \tag{C.26}$$

This is a proper ECM. Furthermore, for functions $p(\boldsymbol{x}) \in [0,1]^d \to [0,1]$, it is equivalent to be 1-Lipschitz with respect to the ECM defined above in Equation C.26 and to be 1-Lipschitz with respect to the "improper" ECM defined in Equation C.25. To show that 1-Lipschitzness with respect to Equation C.26 implies 1-Lipschitzness with respect to Equation C.25, simply note that, $\forall \boldsymbol{x}, \boldsymbol{y}$:

$$|p(\boldsymbol{x}) - p(\boldsymbol{y})| \leq \sum_{i=1}^{d} \min(\frac{|x_i - y_i|^p}{\alpha}, 1) \leq \sum_{i=1}^{d} \frac{|x_i - y_i|^p}{\alpha} \tag{C.27}$$

To show the opposite direction, consider a function p which is 1-Lipschitz w.r.t. Equation C.25, and note that $\forall \boldsymbol{x}, \boldsymbol{y}$, either:

- $\exists i : \frac{|x_i - y_i|^p}{\alpha} > 1$. Then $d(\boldsymbol{x}, \boldsymbol{y}) \geq 1$ for both metrics, so the 1-Lipschitz constraint is vacuously true regardless of the values of $p(\boldsymbol{x}), p(\boldsymbol{y})$.
- $\nexists i : \frac{|x_i - y_i|^p}{\alpha} > 1$. Then

$$|p(\boldsymbol{x}) - p(\boldsymbol{y})| \leq \sum_{i=1}^{d} \frac{|x_i - y_i|^p}{\alpha} = \sum_{i=1}^{d} \min(\frac{|x_i - y_i|^p}{\alpha}, 1) \tag{C.28}$$

  Therefore, we can consider the ECM in Equation C.26 to derive an appropriate Lipschitz constraint for the ℓ_p^p metric. However, note that this is not twice-differentiable, so Theorem

4.2-c does not directly apply. We can however derive an ad-hoc distribution $\mathcal{D}_i$ such that, according to Theorem 4.2-a, $\Pr_i^{\text{split}}(z) = g_i(z), \ \ \forall z, i.$

In particular, we use:

– On the interval $(0, \alpha^{1/p})$, we distribute Λ_i continuously, with pdf function:

$$\text{pdf}_{\Lambda_i}(z) = \frac{1-p}{\alpha} \cdot pz^{p-1} \tag{C.29}$$

– $\Pr(\Lambda_i = \alpha^{1/p}) = p$

We first show that $\Pr_i^{\text{split}}(z) = g_i(z)$ in the case of $z \geq \alpha^{1/p}$:

$$\begin{aligned}
\Pr_i^{\text{split}}(z) &= \Pr_{\mathcal{D}_i}(\Lambda_i \leq z) + 1\mathbb{E}_{\mathcal{D}_i}\left[\frac{\mathbf{1}_{(\Lambda_i \in (z,1])}}{\Lambda_i}\right] \\
&= \Pr_{\mathcal{D}_i}(\Lambda_i \leq 1) + 0 \\
&= 1 = \min(\frac{z^p}{\alpha}, 1) = g_i(z)
\end{aligned} \tag{C.30}$$

Now, we handle the remaining case of $z \in [0, \alpha^{1/p})$:

$$\begin{aligned}
\Pr_i^{\text{split}}(z) &= \Pr_{\mathcal{D}_i}(\Lambda_i \leq z) + z\mathbb{E}_{\mathcal{D}_i}\left[\frac{\mathbf{1}_{(\Lambda_i \in (z,1])}}{\Lambda_i}\right] \\
&= \int_0^z \text{pdf}_{\Lambda_i}(w)dw + z\left[\int_z^{\alpha^{1/p}} \text{pdf}_{\Lambda_i}(w) \cdot \frac{1}{w}dw + \Pr(\Lambda = \alpha^{1/p})\frac{1}{\alpha^{1/p}}\right] \\
&= \int_0^z \frac{1-p}{\alpha} \cdot pw^{p-1}dw + z\left[\int_z^{\alpha^{1/p}} \frac{1-p}{\alpha} \cdot pw^{p-1}\frac{1}{w}dw + \frac{p}{\alpha^{1/p}}\right] \\
&= \frac{1-p}{\alpha}z^p + z\left[\frac{1}{\alpha}(pz^{p-1} - p\alpha^{(p-1)/p}) + \frac{p}{\alpha^{1/p}}\right] \\
&= \frac{1-p}{\alpha}z^p + \frac{z}{\alpha}(pz^{p-1}) \\
&= \frac{z^p}{\alpha} = g_i(z)
\end{aligned} \tag{C.31}$$

So we have that $\Pr_i^{\text{split}}(z) = g_i(z)\ \forall z \in [0,1]$, as desired.

C.1.4 Theorem 4.3

This is the "quantized" form of Theorem 4.2. In order to introduce it, we need to define a quantized from of ECMs, as well as a quantized form of our smoothing method:

Definition C.3 (Quantized Elementwise-concave metric (QECM))**.** *For any $\boldsymbol{x}, \boldsymbol{y}$, let $\delta_i := |x_i - y_i|$. A quantized elementwise-concave metric (QECM) is a metric on $[0,1]^d_{(q)}$ in the form:*

$$d(\boldsymbol{x}, \boldsymbol{y}) := \sum_{i=1}^{d} g_i(\delta_i), \tag{C.32}$$

where $g_1, ..., g_d \subset [0,1]_{(q)} \to [0,1]$ are increasing, concave functions with $g_i(0) = 0$.

Definition C.4 (Quantized Variable-Λ smoothing)**.** *For any $f : [0,1]^d \times [0,1]^d \to [0,1]$, and distribution $\mathcal{D} = \{\mathcal{D}_1, ...\mathcal{D}_d\}$, such that each $\mathcal{D}_i$ has support $[1/q, 1]_{(q)} \cup \{\infty\}$, let:*

$$\Lambda_i \sim \mathcal{D}_i \tag{C.33}$$

If $\Lambda_i = \infty$, then $x_i^{upper} := 1$, $x_i^{lower} := 0$, otherwise:

$$s_i \sim \mathcal{U}(0, \Lambda_i)_{(q)} \tag{C.34}$$

$$x_i^{upper} := \min(\Lambda_i \lceil \frac{x_i - s_i}{\Lambda_i} \rceil + s_i, 1) \tag{C.35}$$

$$x_i^{lower} := \max(\Lambda_i \lceil \frac{x_i - s_i}{\Lambda_i} \rceil + s_i - \Lambda_i, 0) \tag{C.36}$$

$$\tag{C.37}$$

The quantized smoothed function $p_{D,f} \in [0,1]^d_{(q)} \to [0,1]$ *is defined as:*

$$p_{\mathcal{D},f}(\boldsymbol{x}) := \mathop{\mathbb{E}}_{\boldsymbol{s}}\left[f(\boldsymbol{x}^{lower}, \boldsymbol{x}^{upper})\right]. \tag{C.38}$$

Note that we make no assumptions about the joint distributions of Λ *or of* $\boldsymbol{s}$*.*

Before we state and prove each part of the theorem, we will need a "quantized" form of Lemma C.1: Note again that we closely follow the proof of Corollary 3.1 in Chapter 3, which implicitly contains the same result.

Lemma C.2. *For any* $\Lambda_i \in [1/q, 1]_q \cup \{\infty\}$*, let* $s_i \sim \mathcal{U}(0, \Lambda_i)_{(q)}$*. For any* $x_i, y_i \in [0,1]_{(q)}$*, let* $\delta_i := |x_i - y_i|$ *and define* x_i^{upper}*,* x_i^{lower} *as follows: If* $\Lambda_i = \infty$*, then* $x_i^{upper} := 1$*,* $x_i^{lower} := 0$*, otherwise:*

$$x_i^{upper} := \min(\Lambda_i \lceil \frac{x_i - s_i}{\Lambda_i} \rceil + s_i, 1) \tag{C.39}$$

$$x_i^{lower} := \max(\Lambda_i \lceil \frac{x_i - s_i}{\Lambda_i} \rceil + s_i - \Lambda_i, 0) \tag{C.40}$$

and define y_i^{upper}*,* y_i^{lower} *similarly. Then:*

$$\Pr_{s_i}((x_i^{lower}, x_i^{upper}) \neq (y_i^{lower}, y_i^{upper})) = \min\left(\frac{\delta_i}{\Lambda_i}, 1\right) \tag{C.41}$$

Proof. The proof is mostly identical to the proof of Lemma C.1, with minor differences occurring in the cases on $\lceil \frac{x_i}{\Lambda_i} \rceil - \lceil \frac{y_i}{\Lambda_i} \rceil$, which we show here for completeness:

- Case $\lceil \frac{x_i}{\Lambda_i} \rceil - \lceil \frac{y_i}{\Lambda_i} \rceil = 0$. Then $\lceil \frac{x_i - s_i}{\Lambda_i} \rceil = \lceil \frac{y_i - s_i}{\Lambda_i} \rceil$ only in two cases:
  - $\lceil \frac{x_i - s_i}{\Lambda_i} \rceil = \lceil \frac{y_i - s_i}{\Lambda_i} \rceil = \lceil \frac{x_i}{\Lambda_i} \rceil$ iff $\frac{s_i}{\Lambda_i} < \frac{y_i}{\Lambda_i} - (\lceil \frac{y_i}{\Lambda_i} \rceil - 1)$ $(\leq \frac{x_i}{\Lambda_i} - (\lceil \frac{x_i}{\Lambda_i} \rceil - 1))$.

– $\lceil \frac{y_i - s_i}{\Lambda_i} \rceil = \lceil \frac{x_i - s_i}{\Lambda_i} \rceil = \lceil \frac{x_i}{\Lambda_i} \rceil - 1$ iff $\frac{s_i}{\Lambda_i} \geq \frac{x_i}{\Lambda_i} - (\lceil \frac{x_i}{\Lambda_i} \rceil - 1)$ ($\geq \frac{y_i}{\Lambda_i} - (\lceil \frac{y_i}{\Lambda_i} \rceil - 1)$).

Then $\lceil \frac{y_i - s_i}{\Lambda_i} \rceil \neq \lceil \frac{x_i - s_i}{\Lambda_i} \rceil$ iff $\frac{y_i}{\Lambda_i} - (\lceil \frac{x_i}{\Lambda_i} \rceil - 1) \leq \frac{s_i}{\Lambda_i} < \frac{x_i}{\Lambda_i} - (\lceil \frac{x_i}{\Lambda_i} \rceil - 1)$, which occurs with probability $\frac{x_i - y_i}{\Lambda_i} = \frac{\delta_i}{\Lambda_i}$.

- Case $\lceil \frac{x_i}{\Lambda_i} \rceil - \lceil \frac{y_i}{\Lambda_i} \rceil = 1$. Then $\lceil \frac{x_i - s_i}{\Lambda_i} \rceil \neq \lceil \frac{y_i - s_i}{\Lambda_i} \rceil$ only in two cases:

  – $\lceil \frac{y_i - s_i}{\Lambda_i} \rceil = \lceil \frac{y_i}{\Lambda_i} \rceil$ and $\lceil \frac{x_i - s_i}{\Lambda_i} \rceil = \lceil \frac{x_i}{\Lambda_i} \rceil = \lceil \frac{y_i}{\Lambda_i} \rceil + 1$. This happens iff $\frac{s_i}{\Lambda_i} < \frac{x_i}{\Lambda_i} - \lceil \frac{y_i}{\Lambda_i} \rceil$ ($\leq \frac{y_i}{\Lambda_i} - (\lceil \frac{y_i}{\Lambda_i} \rceil - 1)$).

  – $\lceil \frac{y_i - s_i}{\Lambda_i} \rceil = \lceil \frac{y_i}{\Lambda_i} \rceil - 1$ and $\lceil \frac{x_i - s_i}{\Lambda_i} \rceil = \lceil \frac{y_i}{\Lambda_i} \rceil$. This happens iff $\frac{s_i}{\Lambda_i} \geq \frac{y_i}{\Lambda_i} - (\lceil \frac{y_i}{\Lambda_i} \rceil - 1)$ ($\geq \frac{x_i}{\Lambda_i} - \lceil \frac{y_i}{\Lambda_i} \rceil$).

Therefore, $\lceil \frac{y_i - s_i}{\Lambda_i} \rceil = \lceil \frac{x_i - s_i}{\Lambda_i} \rceil$ iff:

$$\frac{x_i}{\Lambda_i} - \lceil \frac{y_i}{\Lambda_i} \rceil \leq \frac{s_i}{\Lambda_i} < \frac{y_i}{\Lambda_i} - (\lceil \frac{y_i}{\Lambda_i} \rceil - 1) \tag{C.42}$$

Which is:

$$\frac{y_i}{\Lambda_i} - \lceil \frac{y_i}{\Lambda_i} \rceil + \frac{\delta_i}{\Lambda_i} \leq \frac{s_i}{\Lambda_i} < \frac{y_i}{\Lambda_i} - \lceil \frac{y_i}{\Lambda_i} \rceil + 1 \tag{C.43}$$

which occurs with probability $1 - \frac{\delta_i}{\Lambda_i}$. Then $\lceil \frac{y_i - s_i}{\Lambda_i} \rceil \neq \lceil \frac{x_i - s_i}{\Lambda_i} \rceil$ with probability $\frac{\delta_i}{\Lambda_i}$.

□

We now state and prove Theorem 3:

Part 1. *Let $\mathcal{D}$ and $f(\cdot)$ be the Λ-distribution and base function used for Quantized Variable-Λ smoothing, respectively. Let $\boldsymbol{x}, \boldsymbol{y} \in [0,1]^d_{(q)}$ be two points. For each dimension i, let $\delta_i := |x_i - y_i|$. The probability that $(x_i^{lower}, x_i^{upper}) \neq (y_i^{lower}, y_i^{upper})$ is given by $\Pr_i^{split}(\delta_i)$, where:*

$$\Pr_i^{split}(z) := \Pr_{\mathcal{D}_i}(\Lambda_i \leq z) + z \mathbb{E}_{\mathcal{D}_i}\left[\frac{\mathbf{1}_{(\Lambda_i \in (z,1])}}{\Lambda_i}\right] \tag{C.44}$$

Proof. Identical to Theorem 4.2-a, except using Lemma C.2 in place of Lemma C.1.

□

Part 2. *Let $d(\cdot,\cdot)$ be a QECM defined by concave functions $g_1,...,g_d$. Let $\mathcal{D}$ and $f(\cdot)$ be the Λ-distribution and base function used for Quantized Variable-Λ smoothing, respectively. If $\forall i \in [d]$ and $\forall z \in [0,1]_{(q)}$,*

$$\Pr_i^{split}(z) \leq g_i(z), \tag{C.45}$$

then, the smoothed function $p_{\mathcal{D},f}(\cdot)$ is 1-Lipschitz with respect to the metric $d(\cdot,\cdot)$.

Proof. Identical to Theorem 4.2-b, except assuming $\boldsymbol{x}, \boldsymbol{y} \in [0,1]^d_{(q)}$ □

Part 3. *If $\mathcal{D}_i$ is constructed as follows:*

- *On the interval $[\frac{1}{q}, \frac{q-1}{q}]_{(q)}$, Λ_i is distributed as:*

$$\Pr(\Lambda_i = z) = -qz\left[g_i\left(z - \frac{1}{q}\right) + g_i\left(z + \frac{1}{q}\right) - 2g_i(z)\right] \quad \forall z \in \left[\frac{1}{q}, \frac{q-1}{q}\right]_{(q)} \tag{C.46}$$

- $\Pr(\Lambda_i = 1) = q\left[g_i(1) - g_i(\frac{q-1}{q})\right]$
- $\Pr(\Lambda_i = \infty) = 1 - g_i(1)$

then

$$\Pr_i^{split}(z) = g_i(z), \quad \forall z \in [0,1]_{(q)}. \tag{C.47}$$

Proof. We first show that this is in fact a normalized probability distribution:

$$\begin{aligned}
&\sum_{j=1}^{q-1} \Pr\left(\Lambda_i = \frac{j}{q}\right) + \Pr(\Lambda_i = 1) + \Pr(\Lambda_i = \infty) = \\
&\sum_{j=1}^{q-1} -j\left[g_i\left(\frac{j-1}{q}\right) + g_i\left(\frac{j+1}{q}\right) - 2g_i\left(\frac{j}{q}\right)\right] + q\left[g_i(1) - g_i\left(\frac{q-1}{q}\right)\right] + 1 - g_i(1) = \\
&2\sum_{j=1}^{q-1} j g_i\left(\frac{j}{q}\right) - \sum_{j=0}^{q-2}(j+1)g_i\left(\frac{j}{q}\right) - \sum_{j=2}^{q}(j-1)g_i\left(\frac{j}{q}\right) + q\left[g_i(1) - g_i\left(\frac{q-1}{q}\right)\right] + 1 - g_i(1) = \\
&\sum_{j=2}^{q-2}(2j - (j+1) - (j-1))g_i\left(\frac{j}{q}\right) - g_i(0) + (2-2)g_i\left(\frac{1}{q}\right) + (2(q-1) \\
&-(q-2))g_i\left(\frac{q-1}{q}\right) - (q-1)g_i(1) + q\left[g_i(1) - g_i\left(\frac{q-1}{q}\right)\right] + 1 - g_i(1) = 1
\end{aligned} \tag{C.48}$$

Where we use the fact that $g_i(0) = 0$ in the last line.

We now show that $\Pr_i^{\text{split}}(z) = g_i(z)$ in the special case of $z = 1$:

$$\begin{aligned}
\Pr_i^{\text{split}}(1) &= \Pr_{\mathcal{D}_i}(\Lambda_i \le 1) + 1\mathbb{E}_{\mathcal{D}_i}\left[\frac{\mathbf{1}_{(\Lambda_i \in (1,1])}}{\Lambda_i}\right] \\
&= \Pr_{\mathcal{D}_i}(\Lambda_i \le 1) \\
&= 1 - \Pr_{\mathcal{D}_i}(\Lambda_i = \infty) \\
&= 1 - (1 - g_i(1)) = g_i(1)
\end{aligned} \tag{C.49}$$

Where in the second line, we use that $(1, 1]$ represents the empty set, so the term in the expectation is always zero.

Now, we handle the remaining case of $z \in [0, (q-1)/q]_{(q)}$:

$$
\begin{aligned}
&\Pr_i^{\text{split}}(z) \\
&= \Pr_{\mathcal{D}_i}(\Lambda_i \le z) + z\mathbb{E}_{\mathcal{D}_i}\left[\frac{\mathbf{1}_{(\Lambda_i \in (z,1])}}{\Lambda_i}\right] \\
&= \sum_{j=1}^{qz} \Pr\left(\Lambda_i = \frac{j}{q}\right) + z\left[\sum_{j=qz+1}^{q-1} \Pr\left(\Lambda_i = \frac{j}{q}\right)\cdot\frac{q}{j} + \Pr(\Lambda = 1)\frac{1}{1}\right] \\
&= \sum_{j=1}^{qz} -j\left[g_i\left(\frac{j-1}{q}\right) + g_i\left(\frac{j+1}{q}\right) - 2g_i\left(\frac{j}{q}\right)\right] \\
&+ z\left[\sum_{j=qz+1}^{q-1} -j\left[g_i\left(\frac{j-1}{q}\right) + g_i\left(\frac{j+1}{q}\right) - 2g_i\left(\frac{j}{q}\right)\right]\cdot\frac{q}{j} + q\left[g_i(1) - g_i\left(\frac{q-1}{q}\right)\right]\right] \\
&= \sum_{j=1}^{qz} -j\left[g_i\left(\frac{j-1}{q}\right) + g_i\left(\frac{j+1}{q}\right) - 2g_i\left(\frac{j}{q}\right)\right] \\
&+ qz\left[\sum_{j=qz+1}^{q-1} -\left[g_i\left(\frac{j-1}{q}\right) + g_i\left(\frac{j+1}{q}\right) - 2g_i\left(\frac{j}{q}\right)\right] + \left[g_i(1) - g_i\left(\frac{q-1}{q}\right)\right]\right] \\
&= -\sum_{j=0}^{qz-1}(j+1)g_i\left(\frac{j}{q}\right) - \sum_{j=2}^{qz+1}(j-1)g_i\left(\frac{j}{q}\right) + 2\sum_{j=1}^{qz} jg_i\left(\frac{j}{q}\right) \\
&+ qz\left[-\sum_{j=qz}^{q-2} g_i\left(\frac{j}{q}\right) - \sum_{j=qz+2}^{q} g_i\left(\frac{j}{q}\right) + 2\sum_{j=qz+1}^{q-1} g_i\left(\frac{j}{q}\right) + g_i(1) - g_i\left(\frac{q-1}{q}\right)\right] \\
&= \sum_{j=2}^{qz-2}(2j-(j+1)-(j-1))g_i\left(\frac{j}{q}\right) - g_i(0) + (2-2)g_i\left(\frac{1}{q}\right) + (2qz - qz + 1)g_i(z) - qzg_i\left(\frac{qz+1}{q}\right) \\
&+ qz\left[(2-1-1)\sum_{j=qz+2}^{q-2} g_i\left(\frac{j}{q}\right) - g_i(z) + (2-1)g_i\left(\frac{qz+1}{q}\right) + (2-1)g_i\left(\frac{q-1}{q}\right) - g_i(1) + g_i(1) - g_i\left(\frac{q-1}{q}\right)\right] \\
&= (qz+1)g_i(z) - qzg_i\left(\frac{qz+1}{q}\right) \\
&+ qz\left[-g_i(z) + g_i\left(\frac{qz+1}{q}\right)\right] \\
&= g_i(z)
\end{aligned}
\tag{C.50}
$$

Where we use the fact that $g_i(0) = 0$ in the second to last line.

Now we have that $\Pr_i^{\text{split}}(z) = g_i(z)\ \forall z \in [0,1]_{(q)}$, as desired.

□

C.2 Drawbacks of the "Global Λ" Method

In the main text, we briefly discuss using a global value for Λ in order to help with derandomization, as follows:

$$\begin{aligned} &\Lambda \sim \mathcal{D}_{\cdot} \\ &s_i \sim \mathcal{U}(0, \Lambda) \quad \forall i \end{aligned} \tag{C.51}$$

There are several issues with this approach. We will focus our discussion on the ℓ_p^p metric, with $\mathcal{D}_{\cdot}$ given as in Corollary 4.1.

Firstly, notice that if $\alpha > 1$, we have that $\Lambda = \infty$ with a nonzero probability $1 - 1/\alpha$: when $\Lambda = \infty$, then the entire vector $\boldsymbol{x}^{\text{lower}}$ will be the zero vector, and the entire vector $\boldsymbol{x}^{\text{upper}}$ will consist of entirely ones. Then the particular value of $f([0, ..., 0]^T, [1, ..., 1]^T)$ will be weighted with weight $1 - 1/\alpha$, and all other, meaningful values in the ensemble with have a combined weight of $1/\alpha$: the final value of the smoothed function $p_{\mathcal{D},f}$ will the differ from the fixed $f([0, ..., 0]^T, [1, ..., 1]^T)$ only by at most $1/\alpha$ at any point. In other words, we essentially have a 1-Lipschitz function scaled by $1/\alpha$, rather than a $1/\alpha$- Lipschitz function.[1]

However, even in the $\alpha = 1$ case, the "global Λ" technique still underperforms the method we ultimately propose, as shown in Figure 4.4 in the main text. One way to understand this is to note that the guarantee provided by this method is unnecessarily tight. In particular, as mentioned in the main text, the global Λ method produces a smoothed function $p_{\mathcal{D},f}$ that is a weighed average of functions which are each $1/\Lambda$-Lipschitz with respect to the ℓ_1 norm, for various values of Λ,

[1] Note that a similar observation was made in Chapter 3 about using a global value of s_i for $\Lambda > 1$

by Theorem 4.1. Let each of these functions be $p_{\Lambda,f}$, so that

$$p_{\mathcal{D},f} = \mathbb{E}_{\mathcal{D}}[p_{\Lambda,f}] \tag{C.52}$$

Note that for each Λ, by the Lipschitz guarantee and $[0,1]$ bounds on the range:

$$p_{\Lambda,f}(\boldsymbol{x}) - p_{\Lambda,f}(\boldsymbol{y}) \leq \min\left(\frac{\|\boldsymbol{x}-\boldsymbol{y}\|_1}{\Lambda}, 1\right) \tag{C.53}$$

However, note that:

$$\begin{aligned}
p_{\mathcal{D},f}(\boldsymbol{x}) - p_{\mathcal{D},f}(\boldsymbol{y}) = \\
\mathbb{E}_{\Lambda\sim\mathcal{D}}[p_{\Lambda,f}(\boldsymbol{x}) - p_{\Lambda,f}(\boldsymbol{y})] \leq \\
\mathbb{E}_{\Lambda\sim\mathcal{D}}\left[\min\left(\frac{\|\boldsymbol{x}-\boldsymbol{y}\|_1}{\Lambda}, 1\right)\right] = \\
\mathbb{E}_{\Lambda\sim\mathcal{D}}\left[\frac{\|\boldsymbol{x}-\boldsymbol{y}\|_1}{\Lambda}\cdot\mathbf{1}_{\Lambda>\|\boldsymbol{x}-\boldsymbol{y}\|_1}\right] + \mathbb{E}_{\Lambda\sim\mathcal{D}}[1\cdot\mathbf{1}_{\Lambda\leq\|\boldsymbol{x}-\boldsymbol{y}\|_1}] = \\
\mathbb{E}_{\Lambda\sim\mathcal{D}}\left[\frac{\|\boldsymbol{x}-\boldsymbol{y}\|_1}{\Lambda}\cdot\mathbf{1}_{\Lambda>\|\boldsymbol{x}-\boldsymbol{y}\|_1}\right] + \mathbb{E}_{\Lambda\sim\mathcal{D}}[1\cdot\mathbf{1}_{\Lambda\leq\|\boldsymbol{x}-\boldsymbol{y}\|_1}] = \\
\|\boldsymbol{x}-\boldsymbol{y}\|_1\mathbb{E}_{\mathcal{D}}\left[\frac{\mathbf{1}_{(\Lambda\in(\|\boldsymbol{x}-\boldsymbol{y}\|_1,1])}}{\Lambda}\right] + \Pr_{\mathcal{D}}(\Lambda\leq\|\boldsymbol{x}-\boldsymbol{y}\|_1) = \Pr^{\text{split}}_{\mathcal{D}}(\|\boldsymbol{x}-\boldsymbol{y}\|_1)
\end{aligned} \tag{C.54}$$

Where $\Pr^{\text{split}}_{\mathcal{D}}$ is defined in terms of $\mathcal{D}$. exactly as in Theorem 4.3-a. Then, by the mechanics of Theorem 4.3-c and from the construction of $\mathcal{D}$., we have:

$$p_{\mathcal{D},f}(\boldsymbol{x}) - p_{\mathcal{D},f}(\boldsymbol{y}) \leq \Pr^{\text{split}}_{\mathcal{D}}(\|\boldsymbol{x}-\boldsymbol{y}\|_1) = g.(\|\boldsymbol{x}-\boldsymbol{y}\|_1) \tag{C.55}$$

In the case of ℓ_p^p metrics with $p < 1$, this means:

$$p_{\mathcal{D},f}(\boldsymbol{x}) - p_{\mathcal{D},f}(\boldsymbol{y}) \leq \frac{\|\boldsymbol{x} - \boldsymbol{y}\|_1^p}{\alpha} \tag{C.56}$$

But note that:

$$p_{\mathcal{D},f}(\boldsymbol{x}) - p_{\mathcal{D},f}(\boldsymbol{y}) \leq \frac{\|\boldsymbol{x} - \boldsymbol{y}\|_1^p}{\alpha} \leq \frac{\|\boldsymbol{x} - \boldsymbol{y}\|_p^p}{\alpha} \tag{C.57}$$

In other words, we are imposing a tighter guarantee than necessary, which depends only on the ℓ_1 distance between $\boldsymbol{x}$ and $\boldsymbol{y}$: the desired ℓ_p^p guarantee is everywhere at least as loose. So, while this technique technically works, it does not really respect the "spirit" of the fractional ℓ_p^p threat model.

C.3 Designing $\mathcal{D}_i$ for Derandomization using Mixed-Integer Linear Programming

As mentioned in Section 4.3 in the main text, one challenge in the derandomization of our technique is to design a distribution $\mathcal{D}_i$ such that all outcomes (Λ_i, s_i) occur with a probability in the form n/B, where $n \in \mathbb{N}$ is an integer, B is a constant integer, and additionally where:

$$\Pr_i^{\text{split}}(z) \approx g_i(z), \quad \forall z \in [0,1]_{(q)}. \tag{C.58}$$

However, strictly:

$$\Pr_i^{\text{split}}(z) \leq g_i(z), \quad \forall z \in [0,1]_{(q)}. \tag{C.59}$$

We first show that we can formulate Equation C.58 as a linear program in the case where we allow arbitrary probabilities for each value of Λ, and then show that we can convert it into a MILP to obtain probabilities in the desired form.

Note that we are working with the quantized form of Variable-Λ smoothing: for convenience, we will therefore introduce the variables:

$$g^j := g_i\left(\frac{j}{q}\right) \forall j \in [q] \tag{C.60}$$

$$v_j := \Pr\left(\Lambda_i = \left(\frac{j}{q}\right)\right) \forall j \in [q] \tag{C.61}$$

Our distribution $\mathcal{D}_i$ is then defined by the vector $\boldsymbol{v}$: the probability that $\Lambda_i = \infty$ is determined by normalization ($\Pr(\Lambda_i = \infty) = 1 - \Sigma_j v_j$) .

We make Equation C.58 rigorous by using the following objective:

$$\begin{aligned} &\text{minimize } \epsilon \text{ such that} \\ &g_i(z) - \epsilon \le \Pr_i^{\text{split}}(z) \le g_i(z), \quad \forall z \in [0,1]_{(q)}. \end{aligned} \tag{C.62}$$

Note that ϵ is a single scalar: we are attempting to achieve uniform convergence. We can write

$\Pr_i^{\text{split}}(z)$ in the following form:

$$\begin{aligned}\Pr_i^{\text{split}}(z) = \\ \Pr_{\mathcal{D}_i}(\Lambda_i \le z) + z\mathbb{E}_{\mathcal{D}_i}\left[\frac{\mathbf{1}_{(\Lambda_i \in (z,1])}}{\Lambda_i}\right] = \\ \sum_{j=1}^{qz} v_j + z\sum_{j=qz+1}^{q}\frac{v_j}{\left(\frac{j}{q}\right)} = \\ \sum_{j=1}^{qz} v_j + qz\sum_{j=qz+1}^{q}\frac{v_j}{j}\end{aligned} \tag{C.63}$$

Then our optimization becomes (letting $k := qz$):

$$\begin{aligned}&\text{minimize } \epsilon \text{ such that} \\ &g^k - \epsilon \le \sum_{j=1}^{k} v_j + k\sum_{j=k+1}^{q}\frac{v_j}{j} \le g^k, \quad \forall k \in [q].\end{aligned} \tag{C.64}$$

With additional constraints:

- $v_j \ge 0, \ \forall j \in [q]$ (Probabilities are non-negative)
- $\sum_{j=1}^{q} v_j \le 1$ (Normalization: recall that additional probability is assigned to $\Lambda = \infty$)
- $\epsilon \ge 0$

This linear program, with variables $\epsilon, \boldsymbol{v}$, completely describes the problem of designing $\mathcal{D}_i$. If $g_i(z)$ is concave (as it should be, by assumption), then this LP always has an optimal $\epsilon = 0$ solution, given in Theorem 4.3-c. (See the proof of that theorem in Appendix C.1.4).

However, we now want all outcomes to have probabilities in the form n/B. Note that for $\Lambda_i = j/q$, there are j outcomes for s_i, each of which must have equal probabilities. We therefore

need $\Lambda_i = j/q$ to occur with a probability in the form $\frac{nj}{B}$, for some integer n. We will then re-scale our parameters:

$$w_j := \frac{Bv_j}{j} \quad \forall j \in [q] \tag{C.65}$$

Our optimization now becomes:

$$\begin{aligned}
&\text{minimize } \epsilon \text{ such that} \\
&g^k - \epsilon \leq \sum_{j=1}^{k} \frac{j \cdot w_j}{B} + k \sum_{j=k+1}^{q} \frac{w_j}{B} \leq g^k, \quad \forall k \in [q]. \\
&w_j \in \mathbb{N} \\
&\sum_{j=1}^{q} j \cdot w_j \leq B \\
&\epsilon \geq 0
\end{aligned} \tag{C.66}$$

This is a mixed-integer linear program, with variables $\epsilon, \boldsymbol{w}$. Once solved, the desired distribution over Λ can be read off from $\boldsymbol{w}$. In practice, when using this method with $g_i(z) = z^p/\alpha$, we only solved the MILP directly for $\alpha = 1$, using budget $B = 1000$: for larger α, we used the fact that Equation C.63 is linear in $\boldsymbol{v}$ to simply scale down $\Pr_{\mathcal{D}}^{\text{split}}(z)$ by scaling up B as $B = 1000\alpha$, without changing the integer allocations of $\boldsymbol{w}$: in practice, this just means adding additional $\Lambda_i = \infty$ outcomes to the list of possible outcomes that are uniformly selected from. Also, rather than optimizing over ϵ, we held ϵ constant at 0.02, so that the problem became a feasibility problem, rather than an optimization problem. The results are shown in Figure 4.5 in the main text. Each of the two MILPs took $\approx$ 10 minutes or less to solve.

We can show that, with sufficiently large budget, arbitrarily close approximations can always be made. In particular, consider using the optimal real-valued solution from Theorem 4.3-c,

and the simply rounding each w_j down to integers. Because the coefficients on w_j's in Equation C.66 are all non-negative, the upper-bounds on these terms will all still be met. The only lower-bound, the $g^k - \epsilon$ term, will remain feasible because ϵ can be made arbitrarily large. Therefore, this rounding technique will not break feasibility. Now, let's look at optimality. Let $\tilde{w}_j$'s be the real-valued, optimal solutions, and w_j's be the rounded solution. Then we have:

$$g^k - \epsilon \leq \sum_{j=1}^{k} \frac{j \cdot w_j}{B} + k \sum_{j=k+1}^{q} \frac{w_j}{B} \leq \sum_{j=1}^{k} \frac{j \cdot \tilde{w}_j}{B} + k \sum_{j=k+1}^{q} \frac{\tilde{w}_j}{B} = g^k, \quad \forall k \in [q]. \tag{C.67}$$

The tightest lower-bound on epsilon will be the constraint where:

$$\epsilon = \left(\sum_{j=1}^{k} \frac{j \cdot \tilde{w}_j}{B} + k \sum_{j=k+1}^{q} \frac{\tilde{w}_j}{B} \right) - \left(\sum_{j=1}^{k} \frac{j \cdot w_j}{B} + k \sum_{j=k+1}^{q} \frac{w_j}{B} \right) \tag{C.68}$$

However, for each j, $\tilde{w}_j - w_j < 1$, so:

$$\epsilon < \left(\sum_{j=1}^{k} \frac{j}{B} + k \sum_{j=k+1}^{q} \frac{1}{B} \right) \leq \sum_{j=1}^{q} \frac{j}{B} = \frac{q^2 + q}{2B} \tag{C.69}$$

Therefore, with sufficiently large budget B, the error ϵ can be made arbitrarily small.

C.4 Explicit Certification Procedure

In order to use our ℓ_p^p Lipschitz guarantee to generate ℓ_p- norm certificates, we follow a procedure similar to the ℓ_1 certificate from Chapter 3. Concretely, for each class c, let $p_c(\boldsymbol{x})$ be the smoothed, $1/\alpha$-ℓ_p^p-Lipschitz logit function that our algorithm produces. In our implementation, we have the *base* classifier f output "hard" classifications: $f_c(\boldsymbol{x}) = 1$ if the base classifier

classifies $\boldsymbol{x}$ into class c, and zero otherwise. Therefore $p_c(\boldsymbol{x})$ can also be though of as the fraction of base classifier outputs with value c.

If two points $\boldsymbol{x}, \boldsymbol{y}$ differ by at most δ in the ℓ_p "norm", then they must differ by at most δ^p in the ℓ_p^p metric. Then by Lipschitz property, we have:

$$|p_c(\boldsymbol{x}) - p_c(\boldsymbol{y})| \leq \frac{\delta^p}{\alpha} \tag{C.70}$$

Now, assume that $\boldsymbol{x}$ is classified as class c by the smoothed classifier ($c = \arg\max_{c'} p_{c'}(\boldsymbol{x})$). Let c' be any other class. By algebra, we have:

$$p_c(\boldsymbol{x}) - p_{c'}(\boldsymbol{x}) - |p_c(\boldsymbol{x}) - p_c(\boldsymbol{y})| - |p_{c'}(\boldsymbol{x}) - p_{c'}(\boldsymbol{y})| \leq p_c(\boldsymbol{y}) - p_{c'}(\boldsymbol{y}) \tag{C.71}$$

Therefore, using Equation C.70, we have:

$$p_c(\boldsymbol{x}) - p_{c'}(\boldsymbol{x}) - \frac{2\delta^p}{\alpha} \leq p_c(\boldsymbol{y}) - p_{c'}(\boldsymbol{y}) \tag{C.72}$$

Then:

$$\left(\frac{\alpha}{2}(p_c(\boldsymbol{x}) - p_{c'}(\boldsymbol{x}))\right)^{1/p} \geq \delta \implies p_c(\boldsymbol{y}) \geq p_{c'}(\boldsymbol{y}) \tag{C.73}$$

This means that the class is guaranteed not to change to c' within an ℓ_p radius of

$$\left(\frac{\alpha}{2}(p_c(\boldsymbol{x}) - p_{c'}(\boldsymbol{x}))\right)^{1/p} \tag{C.74}$$

of $\boldsymbol{x}$. Computing the minimum of this quantity over all classes $c' \neq c$ gives the certified radius.

The above argument ignores the equality case: at radius δ, the two class probabilities may

still be equal, leading to an unclear classification result. To deal with this, we use a trick also used in Chapter 3. Specifically, at classification time, we break ties deterministically using the class index: if $p_c(\boldsymbol{x}) = p_{c'}(\boldsymbol{x})$ and $c < c'$ then the class c will be the final classification. In the case that $c < c'$, then $p_c(\boldsymbol{y}) \geq p_{c'}(\boldsymbol{y})$ is a sufficient condition to ensure that class c is chosen, so we can certify that the class c will be chosen at all points up to and including radius $\delta = \left(\frac{\alpha}{2}(p_c(\boldsymbol{x}) - p_{c'}(\boldsymbol{x}))\right)^{1/p}$.

To deal with the other case, $c' < c$, we subtract any positive ϵ from both sides of Equation C.72:

$$p_c(\boldsymbol{x}) - p_{c'}(\boldsymbol{x}) - \epsilon - \frac{2\delta^p}{\alpha} \leq p_c(\boldsymbol{y}) - p_{c'}(\boldsymbol{y}) - \epsilon \tag{C.75}$$

$$\left(\frac{\alpha}{2}(p_c(\boldsymbol{x}) - p_{c'}(\boldsymbol{x}) - \epsilon)\right)^{1/p} \geq \delta \implies p_c(\boldsymbol{y}) \geq p_{c'}(\boldsymbol{y}) + \epsilon \implies p_c(\boldsymbol{y}) > p_{c'}(\boldsymbol{y}) \tag{C.76}$$

In our deterministic certification implementation, we use $\epsilon := 1/B$, where B is the number of (nonrandom) smoothing samples: this is the smallest difference possible between two values of $p_{cdot}(\cdot)$. Combining the two cases, we get the final form of our certificate:

$$\min_{c':c' \neq c} \left[\left(\frac{\alpha}{2}\left(p_c(\boldsymbol{x}) - p_{c'}(\boldsymbol{x}) - \frac{\mathbf{1}_{c'<c}}{B}\right)\right)^{1/p}\right] \geq \delta \implies \boldsymbol{y} \text{ is assigned class } c \quad \forall \boldsymbol{x}, \boldsymbol{y},\ \|\boldsymbol{x} - \boldsymbol{y}\|_p \leq \delta \tag{C.77}$$

C.5 Representations of Inputs

As we stated in the main text, we modified the architectures used for f in order to accept both inputs $(\boldsymbol{x}^{\text{lower}}, \boldsymbol{x}^{\text{upper}})$, by doubling the number of input channels in the first layer. We tried a variety of alternative methods as well on CIFAR-10 for p=1/2:

- 'Center': using only a single input $\frac{\boldsymbol{x}^{\text{lower}}+\boldsymbol{x}^{\text{upper}}}{2}$, as in DSSN, but with variable-Λ smoothing.
- 'Center Center': Same as 'Center', but with channels duplicated. This acted as an ablation study, to isolate the effect of the additional information of having both channels from the mere increase in network parameters from doubling the number of channels.
- 'Center Error': Channels are $\frac{\boldsymbol{x}^{\text{lower}}+\boldsymbol{x}^{\text{upper}}}{2}$ and $\frac{\boldsymbol{x}^{\text{upper}}-\boldsymbol{x}^{\text{lower}}}{2}$.
- 'Upper Lower': Channels are $\boldsymbol{x}^{\text{upper}}$ and $\boldsymbol{x}^{\text{lower}}$. This is the method presented the main text, and used in other experiments.

See Table C.1 for results. As would be anticipated, the general trend was:

$$\text{DSSN} < \text{'Center'} \approx \text{'Center Center'} < \text{'Center Error'} \approx \text{'Upper Lower'} \tag{C.78}$$

This tells us that Variable-Λ smoothing has an advantage over DSSN for p=1/2 certification, even if only the center of the interval is given to the base classifier. However, having full knowledge of the range of the interval clearly provides an added benefit.

C.6 Effect of Pseudorandom Seed Value

As mentioned in the main text, we use cyclic permutations with pseudorandom offsets to generate the coupled distribution of $\mathcal{D}$, using a seed value of 0. In Table C.2, we compare alternate choices of seed values for CIFAR-10 with $p = 1/2$. Note that the seed value has very little effect on the certified accuracy: certified accuracies are within 1 percentage point of each other. Similar conclusions about the effect of the seed hyperparameter were found in Chapter 3 for the ℓ_1 case.

ρ	10	20	30	40	50	60	70	80
DSSN (From ℓ_1)	42.69% (60.42% @ α=18)	35.04% (60.42% @ α=18)	28.89% (60.42% @ α=18)	23.46% (60.42% @ α=18)	18.81% (60.42% @ α=18)	13.76% (60.42% @ α=18)	8.38% (60.42% @ α=18)	1.27% (60.42% @ α=18)
DSSN (From ℓ_1) (Stab. Training)	41.32% (55.38% @ α=12)	35.56% (50.11% @ α=18)	32.07% (50.11% @ α=18)	28.70% (50.11% @ α=18)	24.95% (50.11% @ α=18)	20.79% (50.11% @ α=18)	16.20% (50.11% @ α=18)	6.98% (50.11% @ α=18)
Center	49.83% (68.35% @ α=15)	42.26% (65.59% @ α=18)	36.54% (65.59% @ α=18)	31.10% (65.59% @ α=18)	25.65% (65.59% @ α=18)	19.93% (65.59% @ α=18)	13.53% (65.59% @ α=18)	2.68% (65.59% @ α=18)
Center (Stab. Training)	47.98% (64.31% @ α=9)	42.27% (56.96% @ α=15)	38.47% (54.79% @ α=18)	35.31% (54.79% @ α=18)	31.91% (54.79% @ α=18)	28.17% (54.79% @ α=18)	23.10% (54.79% @ α=18)	11.82% (54.79% @ α=18)
Center Center	49.78% (66.06% @ α=18)	42.15% (66.06% @ α=18)	36.15% (66.06% @ α=18)	31.17% (66.06% @ α=18)	25.49% (66.06% @ α=18)	19.87% (66.06% @ α=18)	13.21% (66.06% @ α=18)	2.55% (66.06% @ α=18)
Center Center (Stab. Training)	48.33% (60.17% @ α=12)	42.24% (54.83% @ α=18)	38.84% (54.83% @ α=18)	35.59% (54.83% @ α=18)	32.28% (54.83% @ α=18)	28.11% (54.83% @ α=18)	23.16% (54.83% @ α=18)	11.62% (54.83% @ α=18)
Center Error	56.66% (75.80% @ α=12)	49.61% (70.56% @ α=18)	43.50% (70.56% @ α=18)	37.76% (70.56% @ α=18)	32.26% (70.56% @ α=18)	25.80% (70.56% @ α=18)	18.51% (70.56% @ α=18)	4.99% (70.56% @ α=18)
Center Error (Stab. Training)	55.58% (69.99% @ α=9)	48.73% (63.02% @ α=15)	**45.08%** (60.49% @ α=18)	41.86% (60.49% @ α=18)	38.31% (60.49% @ α=18)	34.39% (60.49% @ α=18)	28.98% (60.49% @ α=18)	**16.45%** (60.49% @ α=18)
Upper Lower	**56.74%** (73.22% @ α=15)	**49.80%** (70.57% @ α=18)	43.60% (70.57% @ α=18)	37.97% (70.57% @ α=18)	32.37% (70.57% @ α=18)	25.83% (70.57% @ α=18)	18.19% (70.57% @ α=18)	5.02% (70.57% @ α=18)
Upper Lower (Stab. Training)	55.21% (69.87% @ α=9)	48.72% (62.74% @ α=15)	45.05% (60.44% @ α=18)	**42.26%** (60.44% @ α=18)	**38.62%** (60.44% @ α=18)	**34.42%** (60.44% @ α=18)	**29.01%** (60.44% @ α=18)	16.28% (60.44% @ α=18)

Table C.1: Comparison of $\ell_{1/2}$ CIFAR-10 certificates for a variety of noise representations. See text of Appendix C.5.

ρ	10	20	30	40	50	60	70	80
Seed: 0	56.74% (73.22% @ α=15)	49.80% (70.57% @ α=18)	43.60% (70.57% @ α=18)	37.97% (70.57% @ α=18)	32.37% (70.57% @ α=18)	25.83% (70.57% @ α=18)	18.19% (70.57% @ α=18)	5.02% (70.57% @ α=18)
Seed: 1	56.64% (73.17% @ α=15)	49.28% (70.14% @ α=18)	43.94% (70.14% @ α=18)	38.53% (70.14% @ α=18)	32.61% (70.14% @ α=18)	26.12% (70.14% @ α=18)	18.43% (70.14% @ α=18)	5.25% (70.14% @ α=18)
Seed: 2	56.60% (73.10% @ α=15)	49.35% (70.44% @ α=18)	43.60% (70.44% @ α=18)	38.01% (70.44% @ α=18)	32.10% (70.44% @ α=18)	25.80% (70.44% @ α=18)	18.52% (70.44% @ α=18)	4.70% (70.44% @ α=18)
Seed: 3	56.80% (72.77% @ α=15)	49.71% (70.74% @ α=18)	43.77% (70.74% @ α=18)	38.30% (70.74% @ α=18)	32.18% (70.74% @ α=18)	25.95% (70.74% @ α=18)	18.04% (70.74% @ α=18)	5.01% (70.74% @ α=18)
Seed: 4	56.82% (73.09% @ α=15)	49.70% (70.56% @ α=18)	43.64% (70.56% @ α=18)	37.79% (70.56% @ α=18)	32.09% (70.56% @ α=18)	25.98% (70.56% @ α=18)	18.54% (70.56% @ α=18)	5.04% (70.56% @ α=18)
Seed: 0 (Stab Training)	55.21% (69.87% @ α=9)	48.72% (62.74% @ α=15)	45.05% (60.44% @ α=18)	42.26% (60.44% @ α=18)	38.62% (60.44% @ α=18)	34.42% (60.44% @ α=18)	29.01% (60.44% @ α=18)	16.28% (60.44% @ α=18)
Seed: 1 (Stab Training)	55.81% (69.84% @ α=9)	48.67% (62.51% @ α=15)	44.43% (60.07% @ α=18)	41.45% (60.07% @ α=18)	38.16% (60.07% @ α=18)	34.17% (60.07% @ α=18)	28.82% (60.07% @ α=18)	16.10% (60.07% @ α=18)
Seed: 2 (Stab Training)	55.18% (69.83% @ α=9)	48.52% (62.80% @ α=15)	44.77% (60.13% @ α=18)	41.38% (60.13% @ α=18)	38.03% (60.13% @ α=18)	34.02% (60.13% @ α=18)	28.81% (60.13% @ α=18)	16.08% (60.13% @ α=18)
Seed: 3 (Stab Training)	55.99% (70.27% @ α=9)	48.64% (62.77% @ α=15)	45.16% (60.02% @ α=18)	41.98% (60.02% @ α=18)	38.60% (60.02% @ α=18)	34.61% (60.02% @ α=18)	29.13% (60.02% @ α=18)	16.60% (60.02% @ α=18)
Seed: 4 (Stab Training)	56.10% (69.87% @ α=9)	48.59% (62.90% @ α=15)	44.76% (60.30% @ α=18)	41.53% (60.30% @ α=18)	38.19% (60.30% @ α=18)	34.17% (60.30% @ α=18)	28.81% (60.30% @ α=18)	16.00% (60.30% @ α=18)

Table C.2: Certified accuracy as a function of fractional ℓ_p distance ρ, for $p = 1/2$ on CIFAR-10, using various values of the seed for pseudo-random generation of cyclic permutations for $\mathcal{D}$. We test with $\alpha = \{1, 3, 6, 9, 12, 15, 18\}$ where $1/\alpha$ is the Lipschitz constant of the model, and report the highest certificate for each technique over all of the models. In parentheses, we report the the clean accuracy and the α parameter for the associated model.

C.7 Effect of Cyclic Permutations vs. Arbitrary Permutations

In the main text, we mention that Theorem 4.3 allows for the use of arbitrary permutations in defining the coupling for the distribution $\mathcal{D}$. However, in practice, we choose to use only cyclic permutations of a single list of outcomes. This is because using arbitrary permutations involves storing in memory the complete permutation (each consisting of B outcomes, with up to $B = 18,000$ in our experiments) for *each* dimension. This does not scale efficiently to higher-dimensional problems. On CIFAR-10 with $p = 1/2$, we did attempt this arbitrary permutation method, using pseudo-randomly generated arbitrary permutations for each dimension. Results are found in Table C.3: in general, we find no major benefit to using arbitrary permutations.

C.8 Complete Certification Results on CIFAR-10

In Figures C.1 and C.2, we show the complete certification results for all models used in Table 4.1 in the main text. Note that our method dominates at every noise level, except when $\alpha = 1$: this is because when $\alpha = 1$, the maximum possible certificate using our method is $(1/2)^{1/p}$, while it is $1/2$ using equivalence of norms from an ℓ_1 certificate. However, this is largely irrelevant, because we show that by selecting larger values of the hyperparameter α, we are able to achieve consistently larger certificates.

In Table C.4, we provide the *base* classifier accuracies for the models. Note that at large α, the form of the certificates using our method, and using ℓ_1 certificates through norm conversion,

	10	20	30	40	50	60	70	80
Cyclic Perm.	56.74% (73.22% @ α=15)	49.80% (70.57% @ α=18)	43.60% (70.57% @ α=18)	37.97% (70.57% @ α=18)	32.37% (70.57% @ α=18)	25.83% (70.57% @ α=18)	18.19% (70.57% @ α=18)	5.02% (70.57% @ α=18)
Cyclic Perm. (Stab. Training)	55.21% (69.87% @ α=9)	48.72% (62.74% @ α=15)	45.05% (60.44% @ α=18)	42.26% (60.44% @ α=18)	38.62% (60.44% @ α=18)	34.42% (60.44% @ α=18)	29.01% (60.44% @ α=18)	16.28% (60.44% @ α=18)
Arbitrary Perm.	56.85% (72.90% @ α=15)	49.62% (70.56% @ α=18)	43.74% (70.56% @ α=18)	38.12% (70.56% @ α=18)	32.08% (70.56% @ α=18)	25.94% (70.56% @ α=18)	18.29% (70.56% @ α=18)	4.70% (70.56% @ α=18)
Arbitrary Perm. (Stab. Training)	55.56% (70.28% @ α=9)	48.56% (62.73% @ α=15)	44.60% (60.08% @ α=18)	41.46% (60.08% @ α=18)	38.13% (60.08% @ α=18)	34.39% (60.08% @ α=18)	28.93% (60.08% @ α=18)	16.26% (60.08% @ α=18)

Table C.3: Certified accuracy as a function of fractional ℓ_p distance ρ, for $p = 1/2$ on CIFAR-10, using either pseudorandom cyclic permutations (as in the main text) or psuedorandom arbitrary permutations. We test with $\alpha = \{1, 3, 6, 9, 12, 15, 18\}$ where $1/\alpha$ is the Lipschitz constant of the model, and report the highest certificate for each technique over all of the models. In parentheses, we report the the clean accuracy and the α parameter for the associated model.

are essentially the same: both are (roughly):

$$\min_{c':c'\neq c}\left[\left(\frac{\alpha}{2}\left(p_c(\boldsymbol{x}) - p_{c'}(\boldsymbol{x})\right)\right)^{1/p}\right] \tag{C.79}$$

where $p_c(\boldsymbol{x})$ is the fraction of the smoothing samples on which the base classifier returns the class c (see Appendix C.4 and Section 4.4 in the main text for details.) Therefore the success of our technique at producing larger certificates is entirely because the base classifier is more accurate under our fractional-ℓ_p noise than under splitting noise with a fixed $\Lambda = \alpha$.

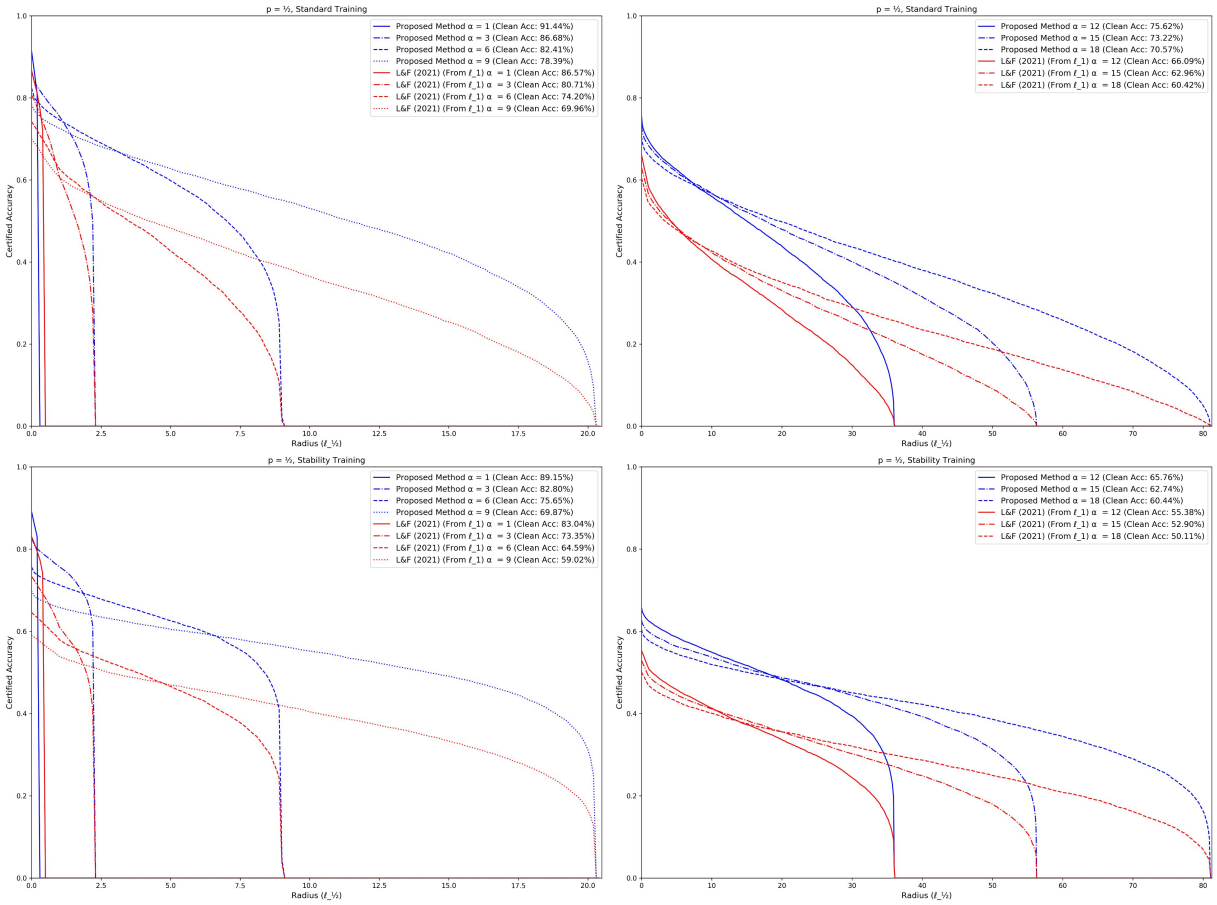


Figure C.1: Full certification results for $p = 1/2$ on CIFAR-10. Left column shows $\alpha \in \{1, 3, 6, 9\}$, right column shows $\alpha \in \{12, 15, 18\}$, top row shows standard training, and bottom row shows stability training. In figure legends, "L&F (2021)" refers to the DSSN algorithm (that is, the algorithm from Levine and Feizi (2021) [73], Chapter 3 of this dissertation).

α	ℓ_1 (DSSN)	ℓ_1 (DSSN) (Stability)	$\ell_{1/2}$	$\ell_{1/2}$ (Stability)	$\ell_{1/3}$	$\ell_{1/3}$ (Stability)
1	83.98%	82.12%	90.16%	88.74%	92.14%	90.84%
3	74.38%	70.89%	83.51%	81.47%	86.35%	84.72%
6	65.68%	61.51%	76.38%	73.51%	79.91%	77.73%
9	59.82%	55.68%	70.97%	67.15%	75.23%	71.82%
12	55.48%	51.68%	66.70%	62.86%	71.42%	67.59%
15	52.21%	48.84%	63.66%	59.51%	67.86%	64.03%
18	49.54%	46.13%	60.75%	56.87%	65.39%	61.25%

Table C.4: Base classifier accuracies on CIFAR-10. Note that as p decreases, the base classifier accuracy increases for a fixed value of α: this leads to larger certificates.

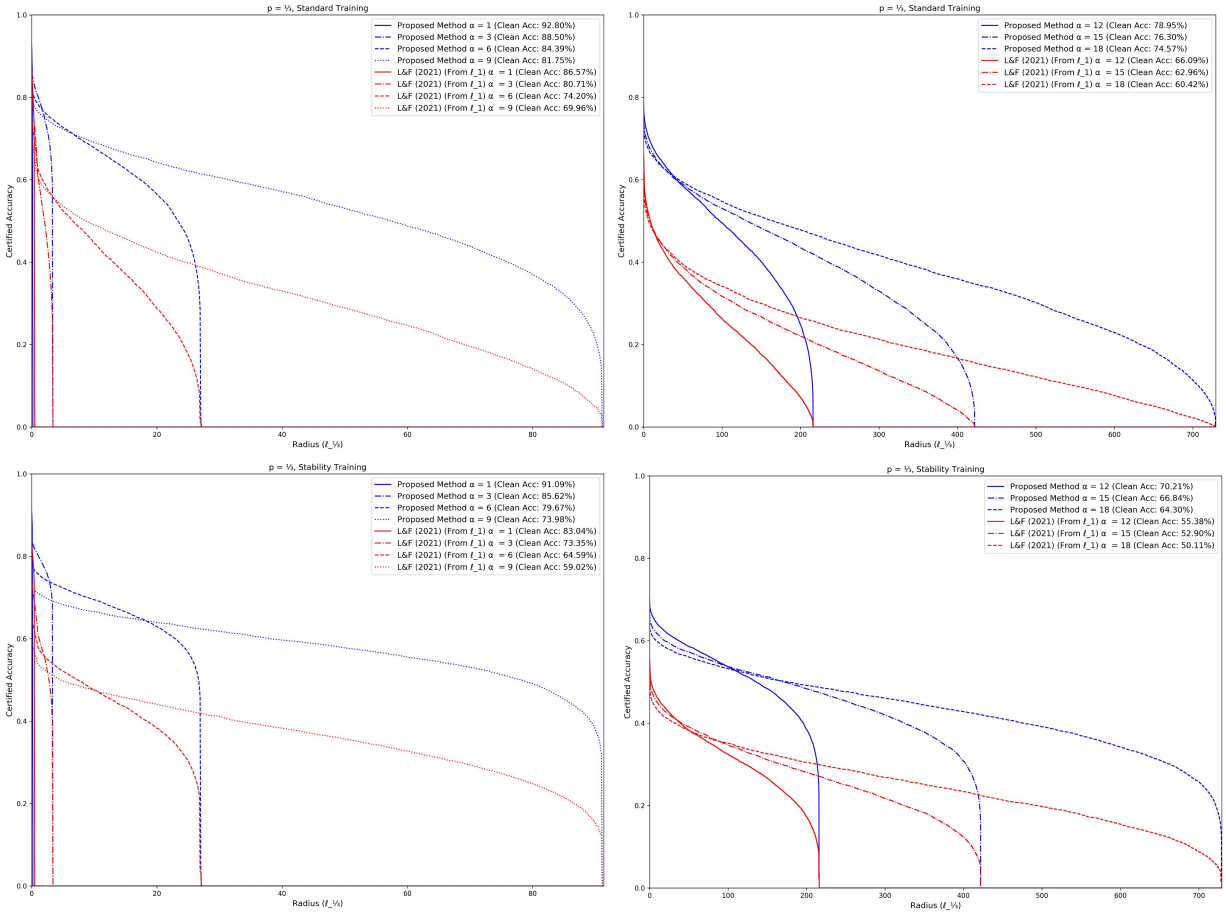


Figure C.2: Full certification results for $p = 1/3$ on CIFAR-10. Left column shows $\alpha \in \{1, 3, 6, 9\}$, right column shows $\alpha \in \{12, 15, 18\}$, top row shows standard training, and bottom row shows stability training. In figure legends, "L&F (2021)" refers to the DSSN algorithm (that is, the algorithm from Levine and Feizi (2021) [73], Chapter 3 of this dissertation).

C.9 CIFAR-10 $p = 1/2$ results with larger values of α

We repeated $p = 1/2$ CIFAR-10 experiments in Table 4.1 in the main text for the additional values of $\alpha \in \{21, 24, 27, 30\}$. Summary results are presented in Table C.5. While this increases certified accuracy under large perturbations, it does so at the cost of decreased clean accuracy. The conclusion that our method significantly outperforms DSSN in the $p < 1$ case still holds. Full results for all classifiers are presented in Figure C.3, and base classifier accuracies are in Table C.6.

	30	60	90	120	150	180	210
DSSN (From ℓ_1)	32.28% (53.35% @ α=30)	24.72% (53.35% @ α=30)	18.95% (53.35% @ α=30)	14.20% (53.35% @ α=30)	9.50% (53.35% @ α=30)	5.42% (53.35% @ α=30)	1.55% (53.35% @ α=30)
DSSN (From ℓ_1) (Stab. Training)	32.39% (47.03% @ α=24)	26.41% (44.38% @ α=30)	22.34% (44.38% @ α=30)	18.68% (44.38% @ α=30)	15.07% (44.38% @ α=30)	11.21% (44.38% @ α=30)	6.21% (44.38% @ α=30)
Variable-Λ	**45.45%** (66.56% @ α=24)	37.45% (63.40% @ α=30)	30.71% (63.40% @ α=30)	24.90% (63.40% @ α=30)	19.40% (63.40% @ α=30)	13.11% (63.40% @ α=30)	5.67% (63.40% @ α=30)
Variable-Λ (Stab Training)	45.05% (60.44% @ α=18)	**38.16%** (56.23% @ α=24)	**33.74%** (52.58% @ α=30)	**29.79%** (52.58% @ α=30)	**25.83%** (52.58% @ α=30)	**20.84%** (52.58% @ α=30)	**14.19%** (52.58% @ α=30)

Table C.5: Certified accuracy as a function of fractional ℓ_p distance ρ, for $p = 1/2$ on CIFAR-10 under large perturbations, with large values of α ($\alpha \in \{21, 24, 27, 30\}$) in addition to the α values used in the main text. As in Table 4.1, we report the highest certificate for each technique over all of the models.

α	ℓ_1 (DSSN)	ℓ_1 (DSSN) (Stability)	$\ell_{1/2}$	$\ell_{1/2}$ (Stability)
21	47.04%	44.14%	58.17%	54.14%
24	45.13%	42.36%	55.91%	52.31%
27	43.49%	40.82%	53.97%	50.46%
30	41.99%	39.36%	52.34%	48.70%

Table C.6: Base classifier accuracies for CIFAR-10, for large values of α.

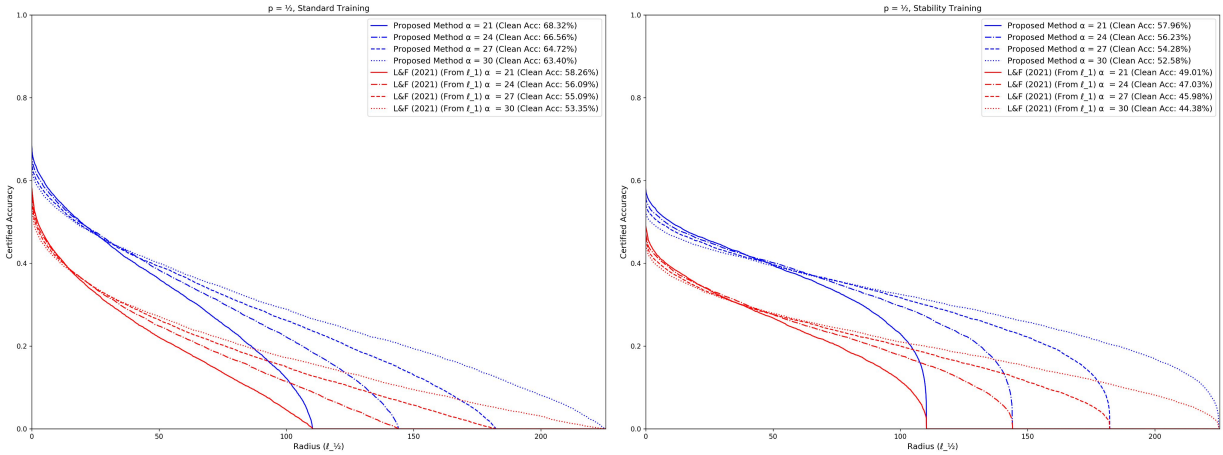


Figure C.3: Full certification results for $p = 1/2$ on CIFAR-10, with $\alpha \in \{21, 24, 27, 30\}$. Left panel shows standard training, right panel shows stability training. In figure legends, "L&F (2021)" refers to the DSSN algorithm (that is, the algorithm from Levine and Feizi (2021) [73], Chapter 3 of this dissertation).

C.10 Base Classifier Accuracies for ImageNet

Base classifier accuracies for the ImageNet results in the main text are provided in Table C.7.

α	ℓ_1 (DSSN)	$\ell_{1/2}$
6	52.50%	58.67%
12	45.49%	53.51%
18	40.39%	49.82%

Table C.7: Base classifier accuracies on ImageNet. Note that for $p = 1/2$, the base classifier accuracy increases compared to $p = 1$ for each fixed value of α: this leads to larger certificates.

Appendix D: Appendix to Chapter 5

D.1 Proofs

Lemma 5.1. *For any normalized probability distributions $\boldsymbol{x}, \boldsymbol{x}' \in [0,1]^{n \times m}$, there exists at least one $\boldsymbol{\delta}$ such that $\boldsymbol{x}' = \Delta(\boldsymbol{x}, \boldsymbol{\delta})$. Furthermore:*

$$\min_{\boldsymbol{\delta}:\ \boldsymbol{x}' = \Delta(\boldsymbol{x}, \boldsymbol{\delta})} \|\boldsymbol{\delta}\|_1 = W_1(\boldsymbol{x}, \boldsymbol{x}') \tag{D.1}$$

Where W_1 denotes the 1-Wasserstein metric, using ℓ_1 distance as the underlying distance metric.

Proof. We first show the equivalence of the above minimization problem with the linear program proposed by [93], restated here:

$$W_1(\boldsymbol{x}, \boldsymbol{x}') = \min_{\mathbf{g}} \sum_{(i,j)} \sum_{(i',j') \in \mathcal{N}((i,j))} \mathbf{g}_{(i,j),(i',j')} \tag{D.2}$$

where $\mathbf{g} \geq 0$ and $\forall (i,j)$,

$$\sum_{(i',j') \in \mathcal{N}((i,j))} \mathbf{g}_{(i,j),(i',j')} - \mathbf{g}_{(i',j'),(i,j)} = \boldsymbol{x}'_{i,j} - \boldsymbol{x}_{i,j}$$

It suffices to show that (1) there is a transformation from the variables $\mathbf{g}$ in Equation D.2 to the

variables $\boldsymbol{\delta}$ in Equation D.1, such that all points which are feasible in Equation D.2 are feasible in D.1 and the minimization objective in Equation D.1 is less than or equal to the minimization objective in Equation D.2, and (2) there is a transformation from the variables $\boldsymbol{\delta}$ in Equation D.1 to the variables g in Equation D.2, such that all points which are feasible in Equation D.1 are feasible in Equation D.2 and the minimization objective in Equation D.2 is less than or equal to the minimization objective in Equation D.1.

We start with (1). We give the transformation as:

$$\begin{aligned} \boldsymbol{\delta}^{\text{vert.}}_{i,j} &:= \mathbf{g}_{(i,j),(i+1,j)} - \mathbf{g}_{(i+1,j),(i,j)} \\ \boldsymbol{\delta}^{\text{horiz.}}_{i,j} &:= \mathbf{g}_{(i,j),(i,j+1)} - \mathbf{g}_{(i,j+1),(i,j)} \end{aligned} \tag{D.3}$$

Where we let $\mathbf{g}_{(n,j),(n+1,j)} = \mathbf{g}_{(n+1,j),(n,j)} = \mathbf{g}_{(i,m+1),(i,m)} = \mathbf{g}_{(i,m),(i,m+1)} = 0$. To show feasibility, we write out fully the flow constraint of Equation D.2:

$$\begin{aligned} &\mathbf{g}_{(i,j),(i+1,j)} - \mathbf{g}_{(i+1,j),(i,j)} + \\ &\mathbf{g}_{(i,j),(i-1,j)} - \mathbf{g}_{(i-1,j),(i,j)} + \\ &\mathbf{g}_{(i,j),(i,j+1)} - \mathbf{g}_{(i,j+1),(i,j)} + \\ &\mathbf{g}_{(i,j),(i,j-1)} - \mathbf{g}_{(i,j-1),(i,j)} = \boldsymbol{x}'_{i,j} - \boldsymbol{x}_{i,j} \end{aligned} \tag{D.4}$$

Substituting in Equation D.3:

$$\boldsymbol{\delta}^{\text{vert.}}_{i,j} + -\boldsymbol{\delta}^{\text{vert.}}_{i-1,j} + \boldsymbol{\delta}^{\text{horiz.}}_{i,j} + -\boldsymbol{\delta}^{\text{horiz.}}_{i,j-1} = \boldsymbol{x}'_{i,j} - \boldsymbol{x}_{i,j} \tag{D.5}$$

But by Definition 5.2, this is exactly:

$$\Delta(\boldsymbol{x}, \boldsymbol{\delta})_{i,j} = \boldsymbol{x}'_{i,j} \tag{D.6}$$

Which is the sole constraint in Equation D.1: then any solution which is feasible in Equation D.2 is feasible in Equation D.1. Also note that:

$$\begin{aligned}
\|\boldsymbol{\delta}\|_1 &= \sum_{i,j} |\boldsymbol{\delta}^{\text{vert.}}_{i,j}| + |\boldsymbol{\delta}^{\text{horiz.}}_{i,j}| \\
&\leq \sum_{i,j} |\mathbf{g}_{(i,j),(i+1,j)}| + |\mathbf{g}_{(i+1,j),(i,j)}| \\
&\quad + |\mathbf{g}_{(i,j),(i,j+1)}| + |\mathbf{g}_{(i,j+1),(i,j)}| \\
&= \sum_{i,j} \mathbf{g}_{(i,j),(i+1,j)} + \mathbf{g}_{(i+1,j),(i,j)} \\
&\quad + \mathbf{g}_{(i,j),(i,j+1)} + \mathbf{g}_{(i,j+1),(i,j)} \\
&= \sum_{i,j} \mathbf{g}_{(i,j),(i+1,j)} + \mathbf{g}_{(i,j),(i,j+1)} \\
&\quad + \sum_{i,j} \mathbf{g}_{(i+1,j),(i,j)} + \mathbf{g}_{(i,j+1),(i,j)} \\
&= \sum_{i,j} \mathbf{g}_{(i,j),(i+1,j)} + \mathbf{g}_{(i,j),(i,j+1)} \\
&\quad + \sum_{i,j} \mathbf{g}_{(i,j),(i-1,j)} + \mathbf{g}_{(i,j),(i,j-1)} \\
&= \sum_{(i,j)} \sum_{(i',j') \in \mathcal{N}((i,j))} \mathbf{g}_{(i,j),(i',j')}
\end{aligned} \tag{D.7}$$

Where the inequality follows from triangle inequality applied to Equation D.3, and in the second sum in the fourth line, we exploit the fact that $\mathbf{g}_{(n,j),(n+1,j)} = \mathbf{g}_{(n+1,j),(n,j)} = \mathbf{g}_{(i,m+1),(i,m)} = \mathbf{g}_{(i,m),(i,m+1)} = 0$ to shift indices. This shows that the minimization objective in Equation D.1 is

less than or equal to the minimization objective in Equation D.2.

Moving on to (2), we give the transformation as:

$$\begin{aligned}
\mathbf{g}_{(i,j),(i+1,j)} &:= \max(\boldsymbol{\delta}^{\text{vert.}}_{i,j}, 0) \\
\mathbf{g}_{(i,j),(i-1,j)} &:= \max(-\boldsymbol{\delta}^{\text{vert.}}_{i-1,j}, 0) \\
\mathbf{g}_{(i,j),(i,j+1)} &:= \max(\boldsymbol{\delta}^{\text{horiz.}}_{i,j}, 0) \\
\mathbf{g}_{(i,j),(i,j-1)} &:= \max(-\boldsymbol{\delta}^{\text{horiz.}}_{i,j-1}, 0)
\end{aligned} \tag{D.8}$$

Note that the non-negativity constraint of Equation D.2 is automatically satisfied by the form of these definitions. Shifting indices, we also have:

$$\begin{aligned}
\mathbf{g}_{(i-1,j),(i,j)} &= \max(\boldsymbol{\delta}^{\text{vert.}}_{i-1,j}, 0) \\
\mathbf{g}_{(i+1,j),(i,j)} &= \max(-\boldsymbol{\delta}^{\text{vert.}}_{i,j}, 0) \\
\mathbf{g}_{(i,j-1),(i,j)} &= \max(\boldsymbol{\delta}^{\text{horiz.}}_{i,j-1}, 0) \\
\mathbf{g}_{(i,j+1),(i,j)} &= \max(-\boldsymbol{\delta}^{\text{horiz.}}_{i,j}, 0)
\end{aligned} \tag{D.9}$$

From the constraint on Equation D.1, we have:

$$
\begin{aligned}
\boldsymbol{x}'_{i,j} - \boldsymbol{x}_{i,j} =& \boldsymbol{\delta}^{\text{vert.}}_{i,j} + \\
& - \boldsymbol{\delta}^{\text{vert.}}_{i-1,j} + \\
& \boldsymbol{\delta}^{\text{horiz.}}_{i,j} + \\
& - \boldsymbol{\delta}^{\text{horiz.}}_{i,j-1} \\
=& \max\left(\boldsymbol{\delta}^{\text{vert.}}_{i,j}, 0\right) - \max\left(-\boldsymbol{\delta}^{\text{vert.}}_{i,j}, 0\right) + \\
& \max\left(-\boldsymbol{\delta}^{\text{vert.}}_{i-1,j}, 0\right) - \max\left(\boldsymbol{\delta}^{\text{vert.}}_{i-1,j}, 0\right) + \\
& \max\left(\boldsymbol{\delta}^{\text{horiz.}}_{i,j}, 0\right) - \max\left(-\boldsymbol{\delta}^{\text{horiz.}}_{i,j}, 0\right) + \\
& \max\left(-\boldsymbol{\delta}^{\text{horiz.}}_{i,j-1}, 0\right) - \max\left(\boldsymbol{\delta}^{\text{horiz.}}_{i,j-1}, 0\right) \\
=& \mathbf{g}_{(i,j),(i+1,j)} - \mathbf{g}_{(i+1,j),(i,j)} + \\
& \mathbf{g}_{(i,j),(i-1,j)} - \mathbf{g}_{(i-1,j),(i,j)} + \\
& \mathbf{g}_{(i,j),(i,j+1)} - \mathbf{g}_{(i,j+1),(i,j)} + \\
& \mathbf{g}_{(i,j),(i,j-1)} - \mathbf{g}_{(i,j-1),(i,j)}
\end{aligned}
\tag{D.10}
$$

Which is exactly the second constraint of Equation D.2: then any solution which is feasible in

Equation D.2 is feasible in Equation D.1. Also note that:

$$
\begin{aligned}
&\sum_{(i,j)} \sum_{(i',j') \in \mathcal{N}((i,j))} \mathbf{g}_{(i,j),(i',j')} \\
&= \sum_{i,j} \mathbf{g}_{(i,j),(i+1,j)} + \mathbf{g}_{(i,j),(i,j+1)} \\
&\quad + \sum_{i,j} \mathbf{g}_{(i,j),(i-1,j)} + \mathbf{g}_{(i,j),(i,j-1)} \\
&= \sum_{i,j} \max(\boldsymbol{\delta}^{\text{vert.}}_{i,j}, 0) + \max(\boldsymbol{\delta}^{\text{horiz.}}_{i,j}, 0) \\
&\quad + \sum_{i,j} \max(-\boldsymbol{\delta}^{\text{vert.}}_{i-1,j}, 0) + \max(-\boldsymbol{\delta}^{\text{horiz.}}_{i,j-1}, 0) \\
&= \sum_{i,j} \max(\boldsymbol{\delta}^{\text{vert.}}_{i,j}, 0) + \max(-\boldsymbol{\delta}^{\text{vert.}}_{i,j}, 0) \\
&\quad + \sum_{i,j} \max(\boldsymbol{\delta}^{\text{horiz.}}_{i,j}, 0) + \max(-\boldsymbol{\delta}^{\text{horiz.}}_{i,j}, 0) \\
&= \sum_{i,j} |\boldsymbol{\delta}^{\text{vert.}}_{i,j}| + |\boldsymbol{\delta}^{\text{horiz.}}_{i,j}| \\
&= \|\boldsymbol{\delta}\|_1
\end{aligned}
\tag{D.11}
$$

Where we again exploit the fact that $\mathbf{g}_{(n,j),(n+1,j)} = \mathbf{g}_{(n+1,j),(n,j)} = \mathbf{g}_{(i,m+1),(i,m)} = \mathbf{g}_{(i,m),(i,m+1)} = 0$ to shift indices, in the fourth line. This shows that the minimization objective in Equation D.2 is less than or equal to the minimization objective in Equation D.1, completing (2).

Finally, now that we have shown that Equations D.1 and D.2 are in fact equivalent minimizations (i.e., we have proven Equation D.1 correct), we would like to show that there is always a feasible solution to D.1, as claimed. By the above transformations, it suffices to show that there is always a feasible solution to Equation D.2. [93] show that any feasible solution the the general Wasserstein minimization LP (Definition 5.1) can be transformed into a solution to Equation D.2, so it suffices to show that the LP in Definition 5.1 always has a feasible solution. This is trivially satisfied by

taking $\Pi = \boldsymbol{x}(\boldsymbol{x}')^T$, where we note that $\boldsymbol{x}$, a probability distribution, is non-negative. □

Theorem 5.1. *Consider a normalized probability distribution $\boldsymbol{x} \in [0,1]^{n \times m}$, and a classification score function $f : \mathbb{R}^{n \times m} \to [0,1]^k$. Let p refer to the Wasserstein-smoothed classification function:*

$$p(\boldsymbol{x}) = \mathop{\mathbb{E}}_{\boldsymbol{\delta} \sim \mathcal{L}(\sigma)} [f(\Delta(\boldsymbol{x}, \boldsymbol{\delta}))] \tag{D.12}$$

Let i be the class assignment of $\boldsymbol{x}$ using the smoothed classifier p (i.e. $i = \arg\max_{i'} p_{i'}(\boldsymbol{x})$). If

$$p_i(\boldsymbol{x}) \geq e^{2\sqrt{2}\rho/\sigma} \max_{i' \neq i} p_{i'}(\boldsymbol{x}) \tag{D.13}$$

Then for any perturbed probability distribution $\tilde{\boldsymbol{x}}$ such that $W_1(\boldsymbol{x}, \tilde{\boldsymbol{x}}) \leq \rho$:

$$p_i(\tilde{\boldsymbol{x}}) \geq \max_{i' \neq i} p_{i'}(\tilde{\boldsymbol{x}}) \tag{D.14}$$

Proof. Let $\boldsymbol{u}$ be the uniform probability vector. As a consequence of Lemma 5.1, for any distribution $\boldsymbol{x}$, there exists a nonempty set of local flow plans $S_{\boldsymbol{x}}$:

$$S_{\boldsymbol{x}} = \{\boldsymbol{\delta} | \boldsymbol{x} = \Delta(\boldsymbol{u}, \boldsymbol{\delta})\} \tag{D.15}$$

Also, we may define a version of the classifier f on the local flow plan domain:

$$f^{\text{flow}}(\boldsymbol{\delta}) = f(\Delta(\boldsymbol{u}, \boldsymbol{\delta})) \tag{D.16}$$

Let $\boldsymbol{\delta}_x$ be an arbitrary element in S_x, and consider any perturbed $\tilde{\boldsymbol{x}}$ such that $W_1(\boldsymbol{x}, \tilde{\boldsymbol{x}}) \leq \rho$. By

Lemma 5.1:

$$\min_{\boldsymbol{\delta}:\, \tilde{\boldsymbol{x}}=\Delta(\boldsymbol{x},\boldsymbol{\delta})} \|\boldsymbol{\delta}\|_1 = W_1(\boldsymbol{x}, \tilde{\boldsymbol{x}}) \tag{D.17}$$

Then, using Equation 5.6:

$$\min_{\boldsymbol{\delta}:\, \tilde{\boldsymbol{x}}=\Delta(\boldsymbol{u},\boldsymbol{\delta}_{\boldsymbol{x}}+\boldsymbol{\delta})} \|\boldsymbol{\delta}\|_1 = W_1(\boldsymbol{x}, \tilde{\boldsymbol{x}}) \tag{D.18}$$

Let the minimum be achieved at $\boldsymbol{\delta}^*$. Making a change of variables ($\boldsymbol{\delta}_{\tilde{\boldsymbol{x}}} = \boldsymbol{\delta}^* + \boldsymbol{\delta}_x$), we have:

$$\|\boldsymbol{\delta}_{\tilde{\boldsymbol{x}}} - \boldsymbol{\delta}_{\boldsymbol{x}}\|_1 = W_1(\boldsymbol{x}, \tilde{\boldsymbol{x}}) \quad \text{where } \tilde{\boldsymbol{x}} = \Delta(\boldsymbol{u}, \boldsymbol{\delta}_{\tilde{\boldsymbol{x}}}) \tag{D.19}$$

Note that for any $\boldsymbol{x}'$ and any $\boldsymbol{\delta}_{\boldsymbol{x}'} \in S_{\boldsymbol{x}'}$ (for $\boldsymbol{\delta}' \sim \mathcal{L}(\sigma)$) :

$$\begin{aligned} p(\boldsymbol{x}') =& \mathbb{E}\left[f(\Delta(\boldsymbol{x}', \boldsymbol{\delta}'))\right] \\ =& \mathbb{E}\left[f(\Delta(\boldsymbol{u}, \boldsymbol{\delta}_{\boldsymbol{x}'} + \boldsymbol{\delta}'))\right] \\ =& \mathbb{E}\left[f^{\text{flow}}(\boldsymbol{\delta}_{\boldsymbol{x}'} + \boldsymbol{\delta}'))\right] \end{aligned} \tag{D.20}$$

We can now apply Proposition 1 from [34], restated here:

Proposition. *Consider a vector $\boldsymbol{v} \in \mathbb{R}^d$, and a classification score function $\boldsymbol{h} : \mathbb{R}^d \to [0,1]^k$. Let $\epsilon \sim \text{Laplace}(0, \sigma)^d$, and let i be the class assignment of $\boldsymbol{v}$ using a Laplace-smoothed version of the classifier $\boldsymbol{h}$:*

$$i = \arg\max_{i'} \mathop{\mathbb{E}}_{\epsilon}\left[\boldsymbol{h}_{i'}(\boldsymbol{v} + \epsilon)\right] \tag{D.21}$$

If:

$$\mathop{\mathbb{E}}_{\epsilon}\left[\boldsymbol{h}_i(\boldsymbol{v} + \epsilon)\right] \geq e^{2\sqrt{2}\rho/\sigma} \max_{i' \neq i} \mathop{\mathbb{E}}_{\epsilon}\left[\boldsymbol{h}_{i'}(\boldsymbol{v} + \epsilon)\right] \tag{D.22}$$

Then for any perturbed probability distribution $\tilde{\boldsymbol{v}}$ such that $\|\boldsymbol{v} - \tilde{\boldsymbol{v}}\|_1 \leq \rho$:

$$\mathop{\mathbb{E}}_{\epsilon}[\boldsymbol{h}_i(\tilde{\boldsymbol{v}} + \epsilon)] \geq \max_{i' \neq i} \mathop{\mathbb{E}}_{\epsilon}[\boldsymbol{h}_{i'}(\tilde{\boldsymbol{v}} + \epsilon)] \tag{D.23}$$

We apply this proposition to f^{flow}, noting that $\|\boldsymbol{\delta}_{\tilde{\boldsymbol{x}}} - \boldsymbol{\delta}_{\boldsymbol{x}}\|_1 = W_1(\boldsymbol{x}, \tilde{\boldsymbol{x}}) \leq \rho$:

$$\begin{aligned} \mathop{\mathbb{E}}_{\boldsymbol{\delta}'}\left[f_i^{\text{flow}}(\boldsymbol{\delta}_{\boldsymbol{x}} + \boldsymbol{\delta}'))\right] &\geq e^{2\sqrt{2}\rho/\sigma} \max_{i' \neq i} \mathop{\mathbb{E}}_{\boldsymbol{\delta}'}\left[f_{i'}^{\text{flow}}(\boldsymbol{\delta}_{\boldsymbol{x}} + \boldsymbol{\delta}'))\right] \\ \implies \mathop{\mathbb{E}}_{\boldsymbol{\delta}'}\left[f_i^{\text{flow}}(\boldsymbol{\delta}_{\tilde{\boldsymbol{x}}} + \boldsymbol{\delta}'))\right] &\geq \max_{i' \neq i} \mathop{\mathbb{E}}_{\boldsymbol{\delta}'}\left[f_{i'}^{\text{flow}}(\boldsymbol{\delta}_{\tilde{\boldsymbol{x}}} + \boldsymbol{\delta}'))\right] \end{aligned} \tag{D.24}$$

Then, using Equation D.20:

$$\begin{aligned} p_i(\boldsymbol{x}) \geq e^{2\sqrt{2}\rho/\sigma} \max_{i' \neq i} p_{i'}(\boldsymbol{x}) \implies \\ p_i(\tilde{\boldsymbol{x}}) \geq \max_{i' \neq i} p_{i'}(\tilde{\boldsymbol{x}}) \end{aligned} \tag{D.25}$$

Which was to be proven. □

Corollary D.1. *For any normalized probability distributions $\boldsymbol{x}, \boldsymbol{x}' \in [0,1]^{n \times m}$, if $W_1(\boldsymbol{x}, \boldsymbol{x}') \leq \rho/2$, then $\|\boldsymbol{x} - \boldsymbol{x}'\|_1 \leq \rho$, where W_1 is the 1-Wasserstein metric using any ℓ_p norm as the underlying distance metric. Furthermore, there exist distributions where these inequalities are tight.*

Proof. Let Π indicate the optimal transport plan between $\boldsymbol{x}$ and $\boldsymbol{x}'$. From Definition 5.1, we have $\Pi\mathbf{1} = \boldsymbol{x}$ and $\Pi^T\mathbf{1} = \boldsymbol{x}'$. Then:

$$(\Pi^T - \Pi)\mathbf{1} = \boldsymbol{x}' - \boldsymbol{x} \tag{D.26}$$

Let Π' represent a modified version of Π, with the diagonal elements set to zero. Note that

$< \Pi', C >=< \Pi, C >$ and $\Pi^T - \Pi = (\Pi')^T - \Pi'$. Then, using triangle inequality:

$$\begin{aligned} &\|(\Pi')^T \mathbf{1}\|_1 + \|(\Pi')\mathbf{1}\|_1 \\ \geq &\|((\Pi')^T - \Pi')\mathbf{1}\|_1 \\ = &\|\boldsymbol{x}' - \boldsymbol{x}\|_1 \end{aligned} \tag{D.27}$$

Because the elements of Π' are non-negative, this is simply:

$$2\sum_{i,j} \Pi'_{i,j} \geq \|((\Pi')^T - \Pi')\mathbf{1}\|_1 = \|\boldsymbol{x}' - \boldsymbol{x}\|_1 \tag{D.28}$$

Then, because the (non-diagonal) elements of C are at least 1 for any ℓ_p norm, we have,

$$2 < \Pi', C > \geq 2\sum_{i,j} \Pi'_{i,j} \geq \|\boldsymbol{x}' - \boldsymbol{x}\|_1 \tag{D.29}$$

Because $< \Pi', C >=< \Pi, C >= W_1(\boldsymbol{x}, \boldsymbol{x}')$, this means that $\|\boldsymbol{x}' - \boldsymbol{x}\|_1 \leq 2W_1(\boldsymbol{x}, \boldsymbol{x}') \leq \rho$, which was to be proven. Note that this inequality can be tight. For example, let $\boldsymbol{x}$ be the distribution where the entire probability mass is at position (i, j), and $\boldsymbol{x}'$ be the distribution where the probability mass is equally split between at positions (i, j) and $(i + 1, j)$. (In other words, $\boldsymbol{x}_{(i,j)} = 1, \boldsymbol{x}'_{(i,j)} = .5, \boldsymbol{x}'_{(i+1,j)} = .5$). In this case, $\|\boldsymbol{x}' - \boldsymbol{x}\|_1 = 1$, $W_1(\boldsymbol{x}, \boldsymbol{x}') = .5$. □

Corollary D.2. *Consider a color image with three channels, denoted $\boldsymbol{x} = [\boldsymbol{x}^R, \boldsymbol{x}^G, \boldsymbol{x}^B]$, normalized such that $\sum_{(i,j)} \boldsymbol{x}^R_{(i,j)} + \boldsymbol{x}^G_{(i,j)} + \boldsymbol{x}^B_{(i,j)} = 1$. Consider a perturbed image $\tilde{\boldsymbol{x}}$ such that $\forall K \in \{R, G, B\}$, $\sum_{(i,j)} \boldsymbol{x}^K_{(i,j)} = \sum_{(i,j)} \tilde{\boldsymbol{x}}^K_{(i,j)}$. Let $W_1(\boldsymbol{x}, \tilde{\boldsymbol{x}})$ denote the 1-Wasserstein distance (with ℓ_1 distance metric) between $\boldsymbol{x}$ and $\tilde{\boldsymbol{x}}$, where, when determining the minimum transport plan, transport between channels is not permitted. Using this definition, let $W_1(\boldsymbol{x}, \tilde{\boldsymbol{x}}) \leq \rho$. De-*

fine:

$$\begin{aligned} \boldsymbol{\delta} &= \{\boldsymbol{\delta}^R, \boldsymbol{\delta}^G, \boldsymbol{\delta}^B\} \\ \Delta(\boldsymbol{x}, \boldsymbol{\delta}) &= \{\Delta(\boldsymbol{x}^R, \boldsymbol{\delta}^{\boldsymbol{R}}), \Delta(\boldsymbol{x}^G, \boldsymbol{\delta}^{\boldsymbol{G}}), \Delta(\boldsymbol{x}^B, \boldsymbol{\delta}^{\boldsymbol{B}})\} \end{aligned} \tag{D.30}$$

and let $\mathcal{L}^{color}(\sigma)$ *represent independent draws of Laplace noise each with standard deviation* σ *in the shape of* $\boldsymbol{\delta}$*. Then if*

$$p_i(\boldsymbol{x}) \geq e^{2\sqrt{2}\rho/\sigma} \max_{i' \neq i} p_{i'}(\boldsymbol{x}) \tag{D.31}$$

then

$$p_i(\tilde{\boldsymbol{x}}) \geq \max_{i' \neq i} p_{i'}(\tilde{\boldsymbol{x}}). \tag{D.32}$$

Proof. Let the mass in each channel be denoted s_K:

$$s_K := \sum_{(i,j)} \boldsymbol{x}^K_{(i,j)} = \sum_{(i,j)} \tilde{\boldsymbol{x}}^K_{(i,j)} \tag{D.33}$$

Consider the formulation of Wasserstein distance given in Definition 5.1. If we represent the elements of $\boldsymbol{x}$ as a vector by concatenating the elements of $\boldsymbol{\delta}^R, \boldsymbol{\delta}^G$, and $\boldsymbol{\delta}^B$, then the restriction that there is no flow between channels amounts to the requirement that Π is block-diagonal:

$$\Pi = \begin{bmatrix} \Pi^R & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \Pi^G & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \Pi^B \end{bmatrix} \tag{D.34}$$

Let $C^{1,1}$ represent the standard cost matrix for 1-Wasserstein transport (with ℓ_1 distance metric). Because the cost of transport within each channel is the same for standard 1-Wasserstein transport

(with ℓ_1 distance metric), we have:

$$C = \begin{bmatrix} C^{1,1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & C^{1,1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & C^{1,1} \end{bmatrix} \tag{D.35}$$

Then we have:

$$< \Pi, C >=< \Pi^R, C^{1,1} > + < \Pi^G, C^{1,1} > + < \Pi^B, C^{1,1} > \tag{D.36}$$

And by Equation D.34, the constraints also factorize out:

$$\begin{aligned} \Pi^R \mathbf{1} = \boldsymbol{x}^R,\ (\Pi^R)^T \mathbf{1} = \tilde{\boldsymbol{x}}^R, \\ \Pi^B \mathbf{1} = \boldsymbol{x}^B,\ (\Pi^B)^T \mathbf{1} = \tilde{\boldsymbol{x}}^B, \\ \Pi^G \mathbf{1} = \boldsymbol{x}^G,\ (\Pi^G)^T \mathbf{1} = \tilde{\boldsymbol{x}}^G \end{aligned} \tag{D.37}$$

Then the variables of each Π^K are separable (in that they appear together in the objective only in the sum and share no constraints). We can then factorize the minimization:

$$W_1(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = \sum_K \min_{\Pi^K \in \mathbb{R}_+^{(n \cdot m) \times (n \cdot m)}} < \Pi^K, C^{(1,1)} >, \tag{D.38}$$

$$\forall K,\ \Pi^K \mathbf{1} = \boldsymbol{x}^K,\ (\Pi^K)^T \mathbf{1} = \tilde{\boldsymbol{x}}^K \tag{D.39}$$

$$\tag{D.40}$$

We can transform each x^K into a normalized probability distribution by scaling it by a factor of

$1/s_K$. We similarly scale each Π^K :

$$\boldsymbol{x}^K_{\text{sc.}} := \frac{\boldsymbol{x}^K}{s_K} \quad \Pi^K_{\text{sc.}} := \frac{\Pi^K}{s_K} \tag{D.41}$$

Then we have:

$$W_1(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = \sum_K s_K \cdot \min_{\Pi^K_{\text{sc.}} \in \mathbb{R}_+^{(n\cdot m)\times(n\cdot m)}} < \Pi^K_{\text{sc.}}, C^{(1,1)} >, \tag{D.42}$$

$$\forall K,\ \Pi^K_{\text{sc.}} \mathbf{1} = \boldsymbol{x}^K_{\text{sc.}},\ (\Pi^K_{\text{sc.}})^T \mathbf{1} = \tilde{\boldsymbol{x}}^K_{\text{sc.}} \tag{D.43}$$

$$\tag{D.44}$$

But note that this is simply:

$$W_1(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = \sum_K s_K \cdot W_1(\boldsymbol{x}^K_{\text{sc.}}, \tilde{\boldsymbol{x}}^K_{\text{sc.}}) \tag{D.45}$$

By Lemma 5.1, this is:

$$W_1(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = \sum_K s_K \cdot \min_{\boldsymbol{\delta}^K_{\text{sc.}} :\, \tilde{\boldsymbol{x}}^K_{\text{sc.}} = \Delta(\boldsymbol{x}^K_{\text{sc.}}, \boldsymbol{\delta}^K_{\text{sc.}})} \|\boldsymbol{\delta}^K_{\text{sc.}}\|_1 \tag{D.46}$$

By the linearity to scaling of Δ and the ℓ_1 norm, this is simply:

$$W_1(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = \sum_K \min_{\boldsymbol{\delta}^K :\, \tilde{\boldsymbol{x}}^K = \Delta(\boldsymbol{x}^K, \boldsymbol{\delta}^K)} \|\boldsymbol{\delta}^K\|_1 \tag{D.47}$$

Which, by Equation D.30, is simply,

$$W_1(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = \min_{\boldsymbol{\delta}:\, \tilde{\boldsymbol{x}} = \Delta(\boldsymbol{x}, \boldsymbol{\delta})} \|\boldsymbol{\delta}\|_1 \tag{D.48}$$

Then all of the mechanics of the proof of Theorem 5.1 apply, and (avoiding unnecessary repetition), we conclude the result. □

Corollary D.3. *Let W^1 denote the ℓ_1 1-Wasserstein distance, and W^2 denote the ℓ_2 1-Wasserstein distance. For a radius ρ_2, define $\rho_1 := \sqrt{2}\rho_2$. Then, for any classifier f and input $\boldsymbol{x}$, if there does not exist any adversarial example $\tilde{\boldsymbol{x}}$ with $W_1(\boldsymbol{x}, \tilde{\boldsymbol{x}}) \leq \rho_1$, then there are also no adversarial examples $\tilde{\boldsymbol{x}}'$ with $W_2(\boldsymbol{x}, \tilde{\boldsymbol{x}}') \leq \rho_2$.*

Proof. We show the contrapositive: If there is an adversarial example $\tilde{\boldsymbol{x}}'$ with $W^2(\boldsymbol{x}, \tilde{\boldsymbol{x}}') \leq \rho_2$, then there is an adversarial example $\tilde{\boldsymbol{x}}$ with $W^1(\boldsymbol{x}, \tilde{\boldsymbol{x}}) \leq \rho_1$. It is sufficient to show that for any arbitrary $\boldsymbol{x}'$, if $W^2(\boldsymbol{x}, \boldsymbol{x}') \leq \rho_2$, then $W^1(\boldsymbol{x}, \boldsymbol{x}') \leq \rho_1$. (The predicate is then satisfied with $\tilde{\boldsymbol{x}}' = \tilde{\boldsymbol{x}} = \boldsymbol{x}'$). In other words, we need to show that

$$\sqrt{2} W^2(\boldsymbol{x}, \boldsymbol{x}') \geq W^1(\boldsymbol{x}, \boldsymbol{x}'), \quad \forall \boldsymbol{x}, \boldsymbol{x}'. \tag{D.49}$$

By the definition of 1-Wasserstein distance, can rewrite this goal as

$$\sqrt{2} \min_{\Pi} < \Pi, C^2 > \; \geq \; \min_{\Pi} < \Pi, C^1 > \tag{D.50}$$

where in both minimizations Π is non-negative and subject to $\Pi \mathbf{1} = \boldsymbol{x}$, $\Pi^T \mathbf{1} = \boldsymbol{x}'$. Here, C^2 and C^1 are the weight matrices for ℓ_2 and ℓ_1 Wasserstein distances.

Note that Π is subject to the same constraints in both minimizations: therefore any Π that is feasible in one is feasible in the other. Let Π_2^* be the minimum of the first (ℓ_2) minimization. Recall that

$$C^2_{(i,j),(i',j')} = \sqrt{(i-i')^2 + (j-j')^2} \tag{D.51}$$

while

$$C^1_{(i,j),(i',j')} = |i-i'| + |j-j'|. \tag{D.52}$$

By equivalence of norms, we have:

$$\sqrt{2} C^2_{(i,j),(i',j')} \geq C^1_{(i,j),(i',j')}. \tag{D.53}$$

Then by linearity (and using Π non-negative),

$$\sqrt{2} < \Pi, C^2 > \; \geq \; < \Pi, C^1 > \quad \forall \Pi \geq 0 \tag{D.54}$$

So

$$\begin{aligned} \sqrt{2} \min_{\Pi} < \Pi, C^2 > \; &= \; \sqrt{2} < \Pi_2^*, C^2 > \\ &\geq \; < \Pi_2^*, C^1 > \\ &\geq \; \min_{\Pi} < \Pi, C^1 >, \end{aligned} \tag{D.55}$$

as desired. □

D.2 Training Parameters

In this paper, network architectures models used were identical to those used in [2]. Unless stated otherwise, all parameters of attacks are the same as used in that paper for each data set. For training smoothed models, we train the base classifier using standard cross-entropy loss on individual noised sample images, using the same noise distribution as used when performing smoothed classification. However, during training, rather than using the same image repeatedly while adding different noise (as at test time), we instead train with each image only once per epoch, with one noise draw. In fact, for computational efficiency and as suggested by [34], we re-use the same noise for each image in a batch. Training parameters are as follows (Tables D.1, D.2):

Training Epochs	200
Batch Size	128
Optimizer	Stochastic Gradient Descent with Momentum
Learning Rate	.001
Momentum	0.9
ℓ_2 Weight Penalty	0.0005

Table D.1: Training Parameters for MNIST Experiments

D.3 Comparison to other Defenses in [2]

In addition to proposing adversarial training as a defense against Wasserstein Adversarial attacks, [2] also tests other defenses. On MNIST, binarization of the input and using a provably ℓ_∞-robust classifier were also tested as defenses: our randomized smoothing method is more ef-

Training Epochs	200
Batch Size	128
Training Set Preprocessing	Normalization, Random Cropping (Padding:4) and Random Horizontal Flip
Optimizer	Stochastic Gradient Descent with Momentum
Learning Rate	.01 (Epochs 1-200) .001 (Epochs 201-400)
Momentum	0.9
ℓ_2 Weight Penalty	0.0005

Table D.2: Training Parameters for CIFAR-10 Experiments

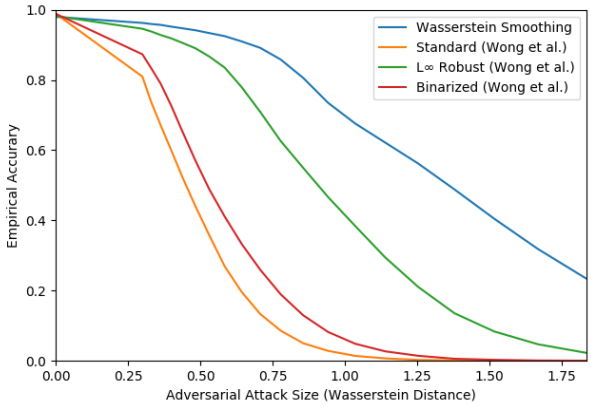


Figure D.1: Comparison of empirical robustness on MNIST to additional defenses from [2], other than adversarial training. Randomized Smoothing shown here is Wasserstein smoothing with $\sigma = 0.01$. (This is the amount of noise which maximizes certified robustness, as seen in Table 5.1.)

fective than these methods at all attack magnitudes (see Figure D.1). On CIFAR-10, [2] only tested a provably ℓ_∞-robust classifier as an additional defense: unfortunately, code was not provided for this model, so we did not attempt to replicate the results.

D.4 Challenges to Deterministic Certification

This chapter was originally published *before* the deterministic ℓ_1 certification method presented in Chapter 3 was developed. However, one may ask whether we can perform Wasserstein smoothing using the DSSN noise distribution in place of the Laplace distribution, in order to derive a deterministic form of Wasserstein smoothing. Here, we show that the proof of the correctness of the Wasserstein smoothing certificate can **not** be easily adapted to using DSSN noise, suggesting that we can not necessarily replace Laplace noise in this method with DSSN noise in order to achieve derandomization.

The key issue is that, in the proof of Theorem 5.1 we rely on the fact (as shown in Equation D.20) that for a given $\tilde{\boldsymbol{x}}$, *for any* $\boldsymbol{\delta}_{\tilde{\boldsymbol{x}}} \in S_{\tilde{\boldsymbol{x}}}$:

$$p(\tilde{\boldsymbol{x}}) = \mathop{\mathbb{E}}_{\boldsymbol{\delta}' \sim \mathcal{L}(\sigma)} \left[f^{\text{flow}}(\boldsymbol{\delta}_{\tilde{\boldsymbol{x}}} + \boldsymbol{\delta}')) \right]. \tag{D.56}$$

We then rely on the fact *there exists* a flow plan $\boldsymbol{\delta}_{\tilde{\boldsymbol{x}}} \in S_{\tilde{\boldsymbol{x}}}$ such that $\|\boldsymbol{\delta}_{\tilde{\boldsymbol{x}}} - \boldsymbol{\delta}_{\boldsymbol{x}}\|_1 = W_1(\boldsymbol{x}, \tilde{\boldsymbol{x}})$.

However, because DSSN noise is non-additive, in the equivalent expression using DSSN noise, the value of the right-hand side (i.e., the expectation of the base classifier under DSSN noise) may depend on the specific choice of flow plan $\boldsymbol{\delta}_{\tilde{\boldsymbol{x}}} \in S_{\tilde{\boldsymbol{x}}}$. If one chooses a flow plan in $S_{\tilde{\boldsymbol{x}}}$ arbitrarily when evaluating $p(\tilde{\boldsymbol{x}})$, it may not be the "closest" flow plan to $\boldsymbol{\delta}_{\boldsymbol{x}}$: the one for which $\|\boldsymbol{\delta}_{\tilde{\boldsymbol{x}}} - \boldsymbol{\delta}_{\boldsymbol{x}}\|_1 = W_1(\boldsymbol{x}, \tilde{\boldsymbol{x}})$.

To demonstrate that under DSSN noise, $p(\tilde{\boldsymbol{x}})$ can depend on the choice of $\boldsymbol{\delta}_{\tilde{\boldsymbol{x}}} \in S_{\tilde{\boldsymbol{x}}}$, consider a three-by-three pixel image, like the one shown in Figure 5.2, and consider the case $\tilde{\boldsymbol{x}} = \boldsymbol{u}$. Two equivalent possible flow plans are:

- $\boldsymbol{\delta}^{a}_{\tilde{\boldsymbol{x}}} = 0$

- $\boldsymbol{\delta}^{b,\,\text{vert.}}_{2,2} = \boldsymbol{\delta}^{b,\,\text{horiz.}}_{3,2} = 1$, $\boldsymbol{\delta}^{b,\,\text{vert.}}_{2,3} = \boldsymbol{\delta}^{b,\,\text{horiz.}}_{2,2} = -1$, all other components zero.

Consider the coupling for DSSN where all components of $\boldsymbol{s}$ are equal, use $\lambda = 0.5$, and also scale the DSSN algorithm such that the input range is $[-1, 1]$ rather than $[0, 1]$ (in other words, let s_i be uniform on [the quantized version of] the range $[-1, 1]$, and "split" the interval into $[-1, s_i)$ and $(s_i, 1]$).

In this setting, note that, for $\boldsymbol{\delta}^{a}_{\tilde{\boldsymbol{x}}}$, all flows will be equal for all noise samples. As a consequence, the total noise added to $\boldsymbol{x}_{2,2}$ will always be zero for all values of $\boldsymbol{s}$.

However, for $\boldsymbol{\delta}^{b}_{\tilde{\boldsymbol{x}}}$, when s_i is near -1, the smoothed value of the flows $\boldsymbol{\delta}^{b,\,\text{vert.}}_{2,3} = \boldsymbol{\delta}^{b,\,\text{horiz.}}_{2,2}$ will be near -1, but the smoothed value of the flows $\boldsymbol{\delta}^{b,\,\text{vert.}}_{2,2} = \boldsymbol{\delta}^{b,\,\text{horiz.}}_{3,2}$ will be near zero, as will all other flows. As a consequence, there will be a net inflow into $\boldsymbol{x}_{2,2}$, and its total noise will not be zero. Therefore the distributions are not equivalent.

Appendix E: Appendix to Chapter 6

E.1 Proofs

We first prove the block smoothing algorithm. Recall the definitions and statement of Theorem 6.1. In particular, recall the base classification counts $n_c(\boldsymbol{x})$:

$$\forall c, \;\; n_c(\boldsymbol{x}) := \sum_{x=1}^{w} \sum_{y=1}^{h} f_c(\boldsymbol{x}, s, x, y) \tag{E.1}$$

And recall the definition of the smoothed classifier:

$$g(\boldsymbol{x}) := \arg\max_c n_c(\boldsymbol{x}), \tag{E.2}$$

where in the case of ties, we choose the smaller-indexed class as the argmax solution.

Theorem 6.1. *For any image $\boldsymbol{x}$, base classifier f, smoothing block size s, and patch size m, if:*

$$n_c(\boldsymbol{x}) \geq \max_{c' \neq c} \left[n_{c'}(\boldsymbol{x}) + \mathbf{1}_{c>c'} \right] + 2(m + s - 1)^2 \tag{E.3}$$

then for any image $\boldsymbol{x}'$ which differs from $\boldsymbol{x}$ only in an $(m \times m)$ patch, $g(\boldsymbol{x}') = c$.

Proof. Let (i, j) represent the upper-right corner of the $m \times m$ patch in which $\boldsymbol{x}$ and $\boldsymbol{x}'$ differ.

Note that, for all c, the output of $f_c(\boldsymbol{x}, s, x, y)$ will be equal to the output of $f_c(\boldsymbol{x}', s, x, y)$, unless the $s \times s$ block retained (starting at (x, y)) intersects with the $m \times m$ adversarial patch (starting at (i, j)). This condition occurs only when both x is in the range between $i - s + 1$ and $i + m - 1$, inclusive, and y is in the range between $j - s + 1$ and $j + m - 1$, inclusive. Note that there are $(m + s - 1)$ values each for x and y which meet this condition, and therefore $(m + s - 1)^2$ such pairs (x, y). Therefore $f_c(\boldsymbol{x}, s, x, y) = f_c(\boldsymbol{x}', s, x, y)$ in all but $(m + s - 1)^2$ cases.

Note that if $i - s + 1 < 0$, then the intersecting values for x, taking into account the wrapping behavior of f, will be $h - (i - s + 1)$ through h, and 0 through $i + m - 1$ (see Figure 6.4 in the main text): there are still $(m + s - 1)$ such values, and a similar argument applies to j.

Therefore, because $f_c(\cdot) \in \{0, 1\}$,

$$\forall c, |n_c(\boldsymbol{x}) - n_c(\boldsymbol{x}')| \leq (m + s - 1)^2. \tag{E.4}$$

Now, consider any $c' \neq c$, such that $n_c(\boldsymbol{x}) \geq [n_{c'}(\boldsymbol{x}) + \mathbf{1}_{c>c'}] + 2(m + s - 1)^2$. There are two cases:

- $c > c'$: In this case, in the event that $n_c(\boldsymbol{x}') = n'_c(\boldsymbol{x}')$, we have that $g(\boldsymbol{x}') = c'$. Therefore, a sufficient condition for $g(\boldsymbol{x}') \neq c'$ is that $n_c(\boldsymbol{x}') > n'_c(\boldsymbol{x}')$. By Equation E.4 and triangle inequality, this must be true if $n_c(\boldsymbol{x}) > [n_{c'}(\boldsymbol{x})] + 2(m + s - 1)^2$, or equivalently, if $n_c(\boldsymbol{x}) \geq [n_{c'}(\boldsymbol{x}) + \mathbf{1}_{c>c'}] + 2(m + s - 1)^2$.

- $c' > c$: In this case, in the event that $n_c(\boldsymbol{x}') = n'_c(\boldsymbol{x}')$, we have that $g(\boldsymbol{x}') = c'$. Therefore, a sufficient condition for $g(\boldsymbol{x}') \neq c'$ is that $n_c(\boldsymbol{x}') \geq n'_c(\boldsymbol{x}')$. By Equation E.4 and triangle inequality, this must be true if $n_c(\boldsymbol{x}) \geq [n_{c'}(\boldsymbol{x}) + \mathbf{1}_{c>c'}] + 2(m + s - 1)^2$.

Therefore, if $n_c(\boldsymbol{x}) \geq \max_{c' \neq c} [n_{c'}(\boldsymbol{x}) + \mathbf{1}_{c>c'}] + 2(m+s-1)^2$, then no class other than c can be output by $g(\boldsymbol{x}')$. □

The column smoothing method can be proved similarly. For completeness, we state and prove Theorem 6.2 here as well. Recall

$$\forall c, \;\; n_c(\boldsymbol{x}) := \sum_{x=1}^{w} f_c(\boldsymbol{x}, s, x) \tag{E.5}$$

Theorem 6.2. *For any image $\boldsymbol{x}$, base classifier f, smoothing band size s, and patch size m, if:*

$$n_c(\boldsymbol{x}) \geq \max_{c' \neq c} [n_{c'}(\boldsymbol{x}) + \mathbf{1}_{c>c'}] + 2(m+s-1) \tag{E.6}$$

then for any image $\boldsymbol{x}'$ which differs from $\boldsymbol{x}$ only in an $(m \times m)$ patch, $g(\boldsymbol{x}') = c$.

Proof. Let (i, j) represent the upper-right corner of the $m \times m$ patch in which $\boldsymbol{x}$ and $\boldsymbol{x}'$ differ. Note that, for all c, the output of $f_c(\boldsymbol{x}, s, x)$ will be equal to the output of $f_c(\boldsymbol{x}', s, x)$, unless the band (of width s) retained, starting at column x, intersects with the $m \times m$ adversarial patch (starting at (i, j)). This condition occurs only when x is in the range between $i - s + 1$ and $i + m - 1$, inclusive. Note that there are $(m + s - 1)$ values for x which meet this condition. Therefore $f_c(\boldsymbol{x}, s, x, y) = f_c(\boldsymbol{x}', s, x, y)$ in all but $(m + s - 1)$ cases.

Again, if $i - s + 1 < 0$, then the intersecting values for x, taking into account the wrapping behavior of f will be $h - (i - s + 1)$ through h, and 0 through $i + m - 1$: there are still $(m + s - 1)$

such values. Therefore, because $f_c(\cdot) \in \{0, 1\}$,

$$\forall c, |n_c(\boldsymbol{x}) - n_c(\boldsymbol{x}')| \leq (m + s - 1). \tag{E.7}$$

The rest of the proof proceeds exactly as in the block smoothing case, with $(m + s - 1)$ substituted for $(m + s - 1)^2$. □

E.2 Full Validation Result Tables for Column and Block Smoothing

Tables E.1 and E.3 present the full validation set clean and certified accuracies for 5×5 patches on MNIST and CIFAR-10, respectively, for all tested values of parameters s and θ, and for both block and column smoothing. Note that this is presented in Figure 6.5 in the main text. Table E.2 presents the validation set clean and certified accuracies for 42×42 patches on ImageNet using column smoothing, for all four tested values of the hyperparameter θ.

E.3 Results for Row Smoothing

We also tested smoothing with rows, rather than columns, on MNIST. This resulted in slightly lower certified accuracy under 5×5 patch attacks (45.32% validation set certified accuracy, versus 53.22% using column smoothing). Full results are presented in Table E.4.

E.4 Multi-column and Multi-block Derandomized Smoothing

In the main text, we argued for having the base classifier use a single contiguous group of pixels on the grounds that, compared to selecting individual pixels, it provides for a smaller risk

of intersecting the adversarial patch. However, there may be some benefit to getting information from multiple distinct areas of an image, even if there is some associated increase in Δ. Rather than just looking at the extremes of entirely independent pixels (Table E.8) versus a single band or block (Figure 6.5 in the main text) we also explored, on MNIST, the intermediate case of using a small number of bands or blocks. In Table E.5, we show all mathematically possible multiple-column certificates on MNIST, as well as several certificates for multiple-blocks with $s = 4$. Interestingly, while the certificates using multiple columns are far below optimal, the certified accuracy for two blocks is only marginally below the best single-block certified accuracy.

For smoothing with multiple blocks or multiple columns, we consider only blocks or columns aligned to a grid starting at the upper-left corner of the image. For example, if using block size $s = 4$, we consider only retaining blocks with upper-left corner (i, j), where i and j are both multiples of 4. This prevents retained blocks from overlapping, and also reduces the (large) number of possible selections of multiple blocks, allowing for derandomized smoothing.

Let the number of retained blocks or bands be κ, and, as in the paper, let the block or band size be s, the image size be $h \times w$, and the adversarial patch size be $m \times m$. For the block case, note that there are $\lceil h/s \rceil \times \lceil w/s \rceil$ such axis-aligned blocks. Of these, the adversarial patch will overlap at most $(\lceil (m-1)/s \rceil + 1)^2$ blocks. For example, for a 5×5 adversarial patch, using block size $s = 4$, the adversarial patch will overlap exactly 4 blocks, regardless of position: see Figure E.1.

When performing derandomized smoothing, we classify all $\binom{\lceil h/s \rceil \times \lceil w/s \rceil}{\kappa}$ possible choices of κ blocks. Of these classifications, at least

$$\binom{\lceil \frac{h}{s} \rceil \times \lceil \frac{w}{s} \rceil - (\lceil \frac{m-1}{s} \rceil + 1)^2}{\kappa}$$

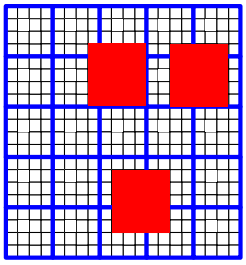


Figure E.1: Multi-block smoothing: for a 5×5 adversarial patch, using block size $s = 4$, the adversarial patch overlaps exactly 4 blocks, regardless of position. Individual pixels are represented by black gridlines. Blocks that may be retained are outlined in blue, and three possible 5×5 adversarial patches are shown in red. Note that this is exact because, in this case, $m - 1$ is divisible by s: in other cases, some choices of adversarial patches may affect fewer than $(\lceil (m-1)/s \rceil + 1)^2$ blocks.

will use none of the at most $(\lceil (m-1)/s \rceil + 1)^2$ blocks which may be affected by the adversary. Therefore, the number of classifications which might be affected by the adversary is at most:

$$\binom{\lceil \frac{h}{s} \rceil \times \lceil \frac{w}{s} \rceil}{\kappa} - \binom{\lceil \frac{h}{s} \rceil \times \lceil \frac{w}{s} \rceil - (\lceil \frac{m-1}{s} \rceil + 1)^2}{\kappa}.$$

We can then use the above quantity in place of the number of classifications $(m+s-1)^2$ that might be affected by the adversarial patch in standard block smoothing (Equation 4). This modification, in addition to classifying all $\binom{\lceil h/s \rceil \times \lceil w/s \rceil}{\kappa}$ selections of κ axis-aligned blocks, is sufficient to adapt the certification algorithm to a multi-block setting.

The column case is similar: there are $\lceil w/s \rceil$ axis-aligned bands (defined as bands which start at a column index which is a multiple of s). Of these, the adversarial patch will overlap at most $(\lceil (m-1)/s \rceil + 1)$ bands. When performing smoothing, we classify all $\binom{\lceil w/s \rceil}{\kappa}$ possible choices of κ bands. Of these classifications, at least

$$\binom{\lceil \frac{w}{s} \rceil - (\lceil \frac{m-1}{s} \rceil + 1)}{\kappa}$$

will use none of the at most $(\lceil (m-1)/s \rceil + 1)$ bands which may be affected by the adversary. Therefore, the number of classifications which might be affected by the adversary is at most:

$$\binom{\lceil \frac{w}{s} \rceil}{\kappa} - \binom{\lceil \frac{w}{s} \rceil - (\lceil \frac{m-1}{s} \rceil + 1)}{\kappa}.$$

Full validation set results for multi-block and multi-band smoothing are shown in Table E.5.

E.5 Comparison with *Randomized* Structured Ablation

As discussed in the main text, there are two benefits to derandomization: first, we can eliminate estimation error, and second, it allows the classifier to abstain or select multiple classes without complicating estimation. In order to distinguish these effects, we present in Tables E.6 and E.7 the certificates on MNIST and CIFAR-10 using *randomized* column smoothing (with the estimation scheme from [63]), versus deterministic column smoothing. We compare to both the "Top-1 class" method (without abstaining or thesholding) as well as to the thresholding method, with $\theta = 0.3$. We find that derandomization alone, without the thresholding method, provides a considerable improvement (around 6 percentage points increase on MNIST, around 7 percentage points on CIFAR-10). On MNIST (although not on CIFAR-10), the thresholding scheme provides a large additional improvement.

E.6 Sparse Randomized Ablation for Patch adversarial Attacks

In Table E.8, we provide the certified accuracies computed from applying sparse Randomized Ablation (Chapter 2) to patch adversarial attacks, as discussed in Section 6.2.1 of the main

text.

E.7 Adversarial Attack Details

In order to test adversarial attacks against our structured ablation model (in particular the column smoothing model) we must work around the non-differentiability of the base classifier f with respect to the image. We accomplish this using a method similar to the attack on smooth classifiers proposed by [33].

In particular, as described in Section 6.2.2.1 in the main text, the base classifier f in our model is implemented using a neural network: let F represent the (SoftMax-ed) logits of this neural network:

$$f_c(\mathbf{x}, s, x) = \begin{cases} 1, \text{ if } F_c(\mathbf{x}, s, x) \geq \theta \\ \\ 0, \text{ if } F_c(\mathbf{x}, s, x) < \theta \end{cases} \tag{E.8}$$

Rather than attacking $n(\boldsymbol{x}) = \sum_{x=1}^{w} f(\boldsymbol{x}, s, x)$, we instead attack a soft smooth classifier, $N(\boldsymbol{x})$:

$$N_c(\boldsymbol{x}) := \frac{1}{w} \sum_{x=1}^{w} F_c(\boldsymbol{x}, s, x) \tag{E.9}$$

The objective of the adversary (as in [4]) is now applied to this soft classifier:

$$\max_{\mathbf{x} \in \text{ (Patch Constraints)}} -\log(\frac{1}{w} \sum_{x=1}^{w} F_y(\boldsymbol{x}, s, x)) \tag{E.10}$$

where y is the true label. The IFGSM patch attack proposed by [4] proceeds by first randomly selecting a patch to attack, and then attacking it with standard IFGSM, without imposing any ℓ_∞

magnitude constraint on the attack (other than as required to produce a feasable image). This is repeated many times on many random patches. However, the most successful attack so far is recorded at each step of optimization, and finally returned at the end of the attack. (Note that this is the most successful attack over all steps of all random initializations.) In [4], this is taken as whichever perturbed version of the image maximizes the objective (Equation E.10). Because we ultimately care about the "hard" smoothed classifier $n(\boldsymbol{x})$, we instead just evaluate the final "hard" classification $n(\boldsymbol{x})$ at each step. We record an attack to return only if it is actually successful at making the final classification incorrect. Note that this does not impose significant computational costs, because we already have the value of each 'soft' base classifier $F_c(\boldsymbol{x})$ at each step.

As mentioned in the main text, for the patch attack, we perform 80 random starts, 150 iterations per random start, and use a step size of 0.05. When attacking patches, we uniformly randomly initialize the pixels in the attacked region. For ℓ_∞ IFGSM, we used IFGSM for 50 iterations and a step size of $0.5/255$: for this, we did not randomize the pixel values before optimizing, but rather started at the initial $\boldsymbol{x}$. Training parameters for baseline models were identical to those for column-smoothed models, except that a regular, full ResNet-18 model was used.

E.8 Evaluation Times

Data on evaluation times (using the optimal hyperparameters to maximize certified accureacy for each method) are shown in Table E.9. We used NVIDIA 2080 Ti GPUs for our experiments.

E.9 Architecture and Training Details

As discussed in the paper, we used the method introduced in Chapter 2 to represent images with pixels ablated: this requires increasing the number of input channels from one to two for greyscale images (MNIST) and from three to six for color images. For MNIST, we used the simple CNN architecture fused in Chapter 2, consisting of two convolutional layers and three fully-connected layers. For CIFAR-10 and ImageNet, we used modified versions ResNet-18 and ResNet-50, respectively, with the number of input channels increased to six. Training details are presented in Table E.10.

For randomized smoothing experiments, we follow the empirical estimation methods proposed by [63]. We certify to 95% confidence, using 1000 random samples to select the putative top class, and 10000 random samples to lower-bound the probability of this class. For sparse randomized ablation on MNIST, we use the same trained models from Chapter 2.

Column Size s	θ = .2		θ = .3		θ = .4		Top-1 class (no threshold)	
	Clean Acc	Cert Acc	Clean Acc	Cert Acc	Clean Acc	Cert Acc	Clean Acc	Cert Acc
1	93.50%	47.52%	93.22%	47.82%	92.56%	45.42%	50.04%	14.78%
2	96.68%	51.46%	**96.78%**	**53.22%**	96.36%	52.34%	72.80%	19.22%
3	97.84%	45.92%	97.70%	47.22%	97.46%	39.14%	82.36%	19.48%
4	97.88%	38.92%	97.92%	32.98%	97.84%	32.52%	85.46%	19.86%
5	98.24%	32.62%	98.26%	25.72%	98.06%	25.26%	93.20%	21.50%
6	98.44%	27.60%	98.30%	21.52%	98.24%	20.42%	95.42%	22.20%
7	98.58%	14.14%	98.60%	15.94%	98.56%	15.98%	97.56%	20.34%
8	98.70%	10.04%	98.68%	11.52%	98.70%	11.76%	97.90%	18.90%
9	98.88%	06.52%	98.82%	08.16%	98.74%	08.32%	98.48%	17.28%
Block Size s	θ = .2		θ = .3		θ = .4		Top-1 class (no threshold)	
1	09.76%	0%	09.76%	0%	09.76%	0%	10.80%	10.80%
2	09.76%	0%	09.76%	0%	09.76%	0%	10.80%	10.80%
3	09.76%	0%	09.76%	0%	09.76%	0%	10.80%	10.80%
4	87.30%	38.06%	86.04%	29.62%	85.94%	19.32%	12.78%	10.80%
5	91.60%	42.58%	90.24%	36.52%	90.32%	27.22%	21.72%	11.08%
6	93.44%	42.90%	92.68%	39.86%	92.70%	33.06%	31.60%	11.74%
7	94.78%	44.00%	94.30%	41.80%	94.52%	35.84%	51.18%	13.30%
8	96.04%	44.04%	95.64%	42.22%	95.66%	36.66%	75.94%	17.64%
9	96.96%	41.74%	97.02%	41.84%	96.92%	37.18%	91.74%	26.84%
10	97.54%	39.84%	97.44%	40.00%	97.50%	36.02%	95.66%	35.30%
11	97.88%	36.00%	97.66%	36.64%	97.64%	32.34%	96.58%	31.26%
12	98.10%	30.40%	98.38%	28.26%	98.30%	32.98%	96.98%	27.26%
13	98.38%	28.26%	98.30%	29.06%	98.44%	22.72%	98.02%	27.66%
14	98.70%	22.22%	98.62%	18.68%	98.62%	14.54%	98.40%	24.04%
15	98.86%	08.90%	98.84%	08.00%	98.86%	06.12%	98.68%	12.90%

Table E.1: Validation set clean and certified accuracies for 5×5 patch adversarial attacks using Block and Column smoothing on MNIST, with results shown for all tested values of parameters s and θ. The value with the highest certified accuracy is shown in bold.

	Clean Accuracy	Certified Accuracy
$s = 25,\ \theta = 0.1$	44.0%	12.0%
$s = 25,\ \theta = 0.2$	**43.1%**	**14.5%**
$s = 25,\ \theta = 0.3$	42.3%	13.8%
$s = 25,\ \theta = 0.4$	40.9%	12.3%

Table E.2: Validation set clean and certified accuracies for 42×42 patch adversarial attacks using Column smoothing on ImageNet, with results shown for all tested values of parameter θ. The value with the highest certified accuracy is shown in bold.

Column Size s	θ = .2		θ = .3		θ = .4		Top-1 class (no threshold)	
	Clean Acc	Cert Acc	Clean Acc	Cert Acc	Clean Acc	Cert Acc	Clean Acc	Cert Acc
1	72.92%	50.74%	72.58%	50.94%	72.90%	50.32%	72.30%	50.94%
2	77.54%	53.26%	77.70%	54.14%	77.68%	53.04%	77.10%	53.40%
3	81.84%	56.14%	81.74%	56.76%	81.98%	56.08%	81.82%	56.24%
4	84.04%	56.62%	**84.04%**	**58.08%**	84.16%	57.12%	83.66%	57.36%
5	85.98%	55.18%	85.66%	56.08%	85.82%	55.98%	85.32%	56.00%
6	87.70%	54.84%	87.90%	56.26%	88.04%	56.10%	87.62%	55.70%
7	89.24%	53.12%	89.48%	54.36%	89.30%	54.04%	89.12%	54.14%
8	90.60%	51.38%	90.68%	52.90%	90.60%	53.12%	90.34%	53.02%
9	91.38%	47.78%	91.30%	49.96%	91.38%	50.30%	91.12%	50.36%
10	91.66%	46.26%	91.74%	49.00%	91.62%	49.44%	91.56%	49.58%
11	92.40%	41.24%	92.26%	45.12%	92.18%	46.08%	92.18%	45.90%
Block Size s	θ = .2		θ = .3		θ = .4		Top-1 class (no threshold)	
1	14.94%	12.44%	14.70%	12.42%	13.62%	10.64%	14.82%	12.80%
2	29.94%	20.96%	25.30%	17.06%	22.66%	14.00%	29.48%	22.58%
3	41.88%	27.70%	36.34%	24.24%	32.88%	18.14%	39.52%	27.80%
4	27.88%	18.80%	31.10%	18.68%	31.98%	16.90%	28.12%	18.34%
5	58.64%	37.72%	57.58%	35.74%	56.00%	29.20%	56.98%	39.64%
6	68.86%	45.88%	67.70%	44.46%	66.26%	39.00%	67.52%	46.20%
7	71.98%	46.96%	72.02%	47.40%	71.32%	43.02%	71.26%	48.38%
8	74.90%	49.18%	74.80%	49.96%	75.78%	46.72%	74.24%	50.68%
9	79.18%	52.04%	78.82%	53.04%	79.50%	50.28%	78.42%	53.88%
10	82.32%	53.44%	82.56%	54.82%	82.96%	52.50%	82.00%	55.18%
11	84.34%	52.84%	84.94%	54.94%	85.12%	52.84%	84.40%	55.24%
12	86.56%	53.26%	86.66%	55.66%	86.88%	54.34%	86.38%	56.08%
13	88.50%	51.76%	88.40%	54.26%	88.98%	53.52%	88.28%	54.74%
14	89.98%	50.72%	89.86%	53.94%	90.22%	53.22%	89.86%	54.58%
15	90.94%	49.88%	91.12%	52.70%	91.28%	52.70%	90.92%	53.44%
16	92.04%	46.04%	91.98%	49.26%	92.02%	49.46%	91.84%	50.12%
17	91.82%	40.30%	92.04%	44.14%	92.12%	44.74%	92.12%	45.14%
18	93.42%	28.42%	93.52%	32.46%	93.54%	33.36%	93.44%	33.98%

Table E.3: Validation set clean and certified accuracies for 5×5 patch adversarial attacks using Block and Column smoothing on CIFAR-10, with results shown for all tested values of parameters s and θ. The value with the highest certified accuracy is shown in bold.

Row Size s	θ = .2		θ = .3		θ = .4	
	Clean Accuracy	Certified Accuracy	Clean Accuracy	Certified Accuracy	Clean Accuracy	Certified Accuracy
1	88.46%	36.54%	85.26%	33.78%	82.86%	25.52%
2	95.58%	43.52%	**93.92%**	**45.32%**	92.04%	43.16%
3	96.28%	41.80%	95.26%	44.96%	94.08%	43.74%
4	97.26%	38.58%	96.40%	42.02%	95.70%	41.82%
5	97.74%	35.74%	97.00%	39.04%	96.52%	39.54%
6	97.60%	32.18%	97.18%	36.98%	96.92%	37.10%
7	98.04%	27.32%	97.62%	32.82%	97.48%	33.50%
8	98.30%	23.16%	98.18%	28.26%	98.06%	29.54%
9	98.24%	17.60%	97.96%	23.80%	97.92%	25.12%

Table E.4: Validation set clean and certified accuracies for 5×5 patch adversarial attacks using Row smoothing on MNIST. Values with highest certified accuracies are shown in bold.

	θ = .2		θ = .3		θ = .4	
	Clean Accuracy	Certified Accuracy	Clean Accuracy	Certified Accuracy	Clean Accuracy	Certified Accuracy
2 columns, s = 1	96.80%	38.12%	**96.50%**	**39.18%**	96.24%	37.38%
2 columns, s = 2	**98.38%**	**31.36%**	98.24%	25.56%	98.10%	25.80%
3 columns, s = 1	97.74%	07.58%	**97.68%**	**09.36%**	97.64%	09.00%
2 blocks, s = 4	**92.32%**	**43.40%**	91.22%	38.78%	91.32%	30.08%
3 blocks, s = 4	**94.98%**	**41.40%**	94.42%	39.38%	94.46%	32.62%
4 blocks, s = 4	**96.26%**	**38.26%**	95.72%	37.50%	95.72%	32.02%

Table E.5: Multi-column and multi-block certificates, with results shown for all tested values of parameter θ. Results are on the MNIST validation set, for 5×5 patches. For each number of blocks/columns and block/column size s, we bold the highest certified accuracy over tested values of the hyperparameter θ.

Column Size s	Derandomized $\theta = .3$	Derandomized Top-1 class	Randomized Column Smoothing
1	47.82%	14.78%	11.80%
2	**53.22%**	19.22%	14.26%
3	47.22%	19.48%	15.14%
4	32.98%	19.86%	15.24%
5	25.72%	21.50%	15.48%
6	21.52%	**22.20%**	**16.32%**
7	15.94%	20.34%	14.50%
8	11.52%	18.90%	14.52%
9	08.16%	17.28%	14.10%

Table E.6: Comparison of Derandomized vs. Randomized Structured Ablation certified accuracies for 5×5 adversarial patches on MNIST.

Column Size s	Derandomized $\theta = .3$	Derandomized Top-1 class	Randomized Column Smoothing
1	50.94%	50.94%	38.16%
2	54.14%	53.40%	41.98%
3	56.76%	56.24%	47.02%
4	**58.08%**	**57.36%**	49.56%
5	56.08%	56.00%	49.58%
6	56.26%	55.70%	**50.38%**
7	54.36%	54.14%	50.04%
8	52.90%	53.02%	48.94%
9	49.96%	50.36%	47.28%
10	49.00%	49.58%	46.46%
11	45.12%	45.90%	43.28%

Table E.7: Comparison of Derandomized vs. Randomized Structured Ablation certified accuracies for 5×5 adversarial patches on CIFAR-10.

MNIST			CIFAR-10		
Retained pixels k	Classification accuracy	Certified accuracy	Retained pixels k	Classification accuracy	Certified accuracy
5	32.44%	7.58%	25	68.28%	13.28%
10	75.02%	5.40%	50	74.68%	0
15	86.32%	4.34%	75	78.26%	0
20	90.36%	0.10%	100	80.98%	0
25	93.20%	0	125	83.82%	0
30	94.72%	0	150	85.70%	0

Table E.8: Certified accuracy to 5×5 adversarial patches from directly applying ℓ_0 smoothing as proposed in Chapter 2. Note that with ℓ_0 smoothing, the geometry of the attack is not taken into consideration: these are therefore actually certified accuracies for any ℓ_0 attack on up to $\rho = 25$ pixels. The certificates are probabilistic, with 95% confidence.

Method and Dataset	Images	Seconds	GPUs	GPU-seconds/image
Column, MNIST	5000	6.61	1	0.00132
Block, MNIST	5000	48.1	1	0.00962
Column, CIFAR-10	5000	30.8	1	0.00616
Block, CIFAR-10	5000	851	1	0.170
Column, ImageNet	1000	622	4	2.49

Table E.9: Evaluation times. Note that evaluation and certification both require evaluating each base classifier, so these are also the certification times (our evaluation script reports both clean and certified accuracy).

	MNIST	**CIFAR-10**	**ImageNet**
Training Epochs	400	350	60
Batch Size	128	128	196
Training Set Preprocessing	None	Random Cropping (Padding:4) and Random Horizontal Flip	Random Horizontal Flip
Optimizer	Stochastic Gradient Descent with Momentum	Stochastic Gradient Descent with Momentum	Stochastic Gradient Descent with Momentum
Learning Rate	.01 (Epochs 1-200) .001 (Epochs 201-400)	.1 (Epochs 1-150) .01 (Epochs 151-250) .001 (Epochs 251-350)	.1 (Epochs 1-20) .01 (Epochs 21-40) .001 (Epochs 41-60)
Momentum	0.9	0.9	0.9
ℓ_2 Weight Penalty	0.0005	0.0005	0.0005

Table E.10: Training Parameters

Appendix F: Appendix to Chapter 7

F.1 Proofs

Theorem 7.1. *For a fixed deterministic base classifier f, hash function h, ensemble size k, training set T, and input $\boldsymbol{x}$, let:*

$$\begin{aligned} c &:= g_{\textbf{\textit{dpa}}}(T, \boldsymbol{x}) \\ \bar{\rho}(\boldsymbol{x}) &:= \left\lfloor \frac{n_c - \max_{c' \neq c}(n_{c'}(\boldsymbol{x}) + \mathbf{1}_{c'<c})}{2} . \right\rfloor \end{aligned} \tag{F.1}$$

Then, for any poisoned training set U, if $|T \ominus U| \leq \bar{\rho}(\boldsymbol{x})$, we have: $g_{\textbf{\textit{dpa}}}(U, \boldsymbol{x}) = c$.

Proof. We define the partitions, trained classifiers, and counts for each training set (T and U) as described in the main text:

$$\begin{aligned} P_i^T &:= \{\boldsymbol{t} \in T \,|\, h(\boldsymbol{t}) \equiv i \pmod{k}\} \\ P_i^U &:= \{\boldsymbol{t} \in U \,|\, h(\boldsymbol{t}) \equiv i \pmod{k}\} \end{aligned} \tag{F.2}$$

$$\begin{aligned} f_i^T(\boldsymbol{x}) &:= f(P_i^T, \boldsymbol{x}) \\ f_i^U(\boldsymbol{x}) &:= f(P_i^U, \boldsymbol{x}) \end{aligned} \tag{F.3}$$

$$n_c^T(\boldsymbol{x}) := |\{i \in [k] | f_i^T(\boldsymbol{x}) = c\}|$$
$$n_c^U(\boldsymbol{x}) := |\{i \in [k] | f_i^U(\boldsymbol{x}) = c\}| \tag{F.4}$$

$$g_{\textbf{dpa}}(T, \boldsymbol{x}) := \arg\max_c n_c^T(\boldsymbol{x})$$
$$g_{\textbf{dpa}}(U, \boldsymbol{x}) := \arg\max_c n_c^U(\boldsymbol{x}) \tag{F.5}$$

Note that here, we are using superscripts to explicitly distinguish between partitions (as well as base classifiers and counts) of the clean training set T and the poisoned dataset U (i.e, P_i^T is equivalent to P_i in the main text). In Equation F.5, as discussed in the main text, when taking the argmax, we break ties deterministically by returning the smaller class index.

Note that $P_i^T = P_i^U$ unless there is some t, with $h(t) \equiv i \pmod k$, in $T \ominus U$. Because the mapping from t to $h(t) \pmod k$ is a deterministic function, the number of partitions i for which $P_i^T \neq P_i^U$ is at most $|T \ominus U|$, which is at most $\bar{\rho}(\boldsymbol{x})$. $P_i^T = P_i^U$ implies $f_i^T(\boldsymbol{x}) = f_i^U(\boldsymbol{x})$, so the number of classifiers i for which $f_i^T(\boldsymbol{x}) \neq f_i^U(\boldsymbol{x})$ is also at most $\bar{\rho}(\boldsymbol{x})$. Then:

$$\forall c' : |n_{c'}^T - n_{c'}^U| \leq \bar{\rho}(\boldsymbol{x}). \tag{F.6}$$

Let $c := g_{\textbf{dpa}}(T, \boldsymbol{x})$. Note that $g_{\textbf{dpa}}(U, \boldsymbol{x}) = c$ iff:

$$\forall c' < c : n_c^U(\boldsymbol{x}) > n_{c'}^U(\boldsymbol{x})$$
$$\forall c' > c : n_c^U(\boldsymbol{x}) \geq n_{c'}^U(\boldsymbol{x})$$

where the separate cases come from the deterministic selection of the smaller index in cases of ties in Equation F.5: this can be condensed to $\forall c' \neq c : n_c^U(\boldsymbol{x}) \geq n_{c'}^U(\boldsymbol{x}) + \mathbf{1}_{c'<c}$. Then by triangle inequality with Equation F.6, we have that $g_{\textbf{dpa}}(U, \boldsymbol{x}) = c$ if $\forall c' \neq c : n_c^T(\boldsymbol{x}) \geq n_{c'}^T(\boldsymbol{x}) + 2\bar{\rho} + \mathbf{1}_{c'<c}$.

This condition is true by the definition of $\bar{\rho}(\boldsymbol{x})$, so $g_{\textbf{dpa}}(U, \boldsymbol{x}) = c$. □

Theorem 7.2. *For a fixed deterministic semi-supervised base classifier f, ensemble size k, training set T (with no repeated samples), and input $\boldsymbol{x}$, let:*

$$\begin{aligned} c &:= g_{\textbf{ssdpa}}(T, \boldsymbol{x}) \\ \bar{\rho}(\boldsymbol{x}) &:= \left\lfloor \frac{n_c - \max_{c' \neq c}(n_{c'}(\boldsymbol{x}) + \mathbf{1}_{c'<c})}{2} \right\rfloor \end{aligned} \tag{F.7}$$

For a poisoned training set U obtained by changing the labels of at most $\bar{\rho}$ samples in T, $g_{\textbf{ssdpa}}(U, \boldsymbol{x}) = c$.

Proof. Recall the definition:

$$T_{\text{sorted}} := \textbf{Sort}(\textbf{samples}(T)). \tag{F.8}$$

Because $\textbf{samples}(T) = \textbf{samples}(U)$, we have $T_{\text{sorted}} = U_{\text{sorted}}$. We can then define partitions and base classifiers for each training set (T and U) as described in the main text:

$$\begin{aligned} P_i^T &:= \{\boldsymbol{t} \in T | \textbf{index}(T_{\text{sorted}}, \textbf{sample}(\boldsymbol{t})) \equiv i \pmod k\} \\ P_i^U &:= \{\boldsymbol{t} \in U | \textbf{index}(T_{\text{sorted}}, \textbf{sample}(\boldsymbol{t})) \equiv i \pmod k\} \end{aligned} \tag{F.9}$$

$$\begin{aligned} f_i^T(\boldsymbol{x}) &:= f(\textbf{samples}(T), P_i^T, \boldsymbol{x}) \\ f_i^U(\boldsymbol{x}) &:= f(\textbf{samples}(T), P_i^U, \boldsymbol{x}) \end{aligned} \tag{F.10}$$

Recall that for any $\boldsymbol{t} \in T$, $\textbf{index}(T_{\text{sorted}}, \textbf{sample}(\boldsymbol{t}))$ is invariant under label-flipping attack to T. Then, for each i, the samples in P_i^T will be the same as the samples in P_i^U, possibly with some labels flipped. In particular, the functions $f_i^T(\cdot)$ and $f_i^U(\cdot)$ will be identical, unless the label of some sample with $\textbf{index}(T_{\text{sorted}}, \textbf{sample}(\boldsymbol{t})) \equiv i \pmod k$ has been changed. If at most

$\bar{\rho}(\boldsymbol{x})$ labels change, at most $\bar{\rho}(\boldsymbol{x})$ ensemble classifiers are affected: the rest of the proof proceeds similarly as that of Theorem 7.1. □

F.2 Binary MNIST Experiments

We perform a specialized instance of SS-DPA on the binary '1' versus '7' MNIST classification task. Specifically, we set $k = m$, so that every partition receives only one label. We first use 2-means clustering on the unlabeled data, to compute two means. This allows for each base classifier to use a very simple "semi-supervised learning algorithm": if the test image and the one labeled training image provided to the base classifier belong to the same cluster, then the base classifier assigns the label of the training image to the test image. Otherwise, it assigns the opposite label to the test image. Formally:

$$\mu_1, \mu_2 := \text{Cluster centroids of } \textbf{samples}(T)$$

$$\textbf{assignment}(\boldsymbol{s}) := \begin{cases} 1, & \text{if } \|\mu_1 - \boldsymbol{s}\|_2 \leq \|\mu_2 - \boldsymbol{s}\|_2 \\ 2, & \text{if } \|\mu_1 - \boldsymbol{s}\|_2 > \|\mu_2 - \boldsymbol{s}\|_2 \end{cases}$$

$$f_i(\boldsymbol{x}) = f(\textbf{samples}(T), \{\boldsymbol{t}_i\}, \boldsymbol{x}) = \begin{cases} \textbf{label}(\boldsymbol{t}_i), & \text{if } \textbf{assignment}(\boldsymbol{x}) = \textbf{assignment}(\boldsymbol{t}_i) \\ 1 - \textbf{label}(\boldsymbol{t}_i), & \text{if } \textbf{assignment}(\boldsymbol{x}) \neq \textbf{assignment}(\boldsymbol{t}_i) \end{cases}$$

Note that each base classifier behaves exactly identically, up to a transpose of the labels: so in practice, we simply count the training samples which associate each of the two cluster centroids

with each of the two labels, and determine the number of label flips which would be required to change the consensus label assignments *of the clusters*. At the test time, each test image therefore needs to be processed only once. The amount of time required for inference is then simply the time needed to calculate the distance from the test sample to each of the two clusters. This also means that every image has the same robustness certificate. As stated in the main text, using this method, we are able to achieve clean accuracy of 95.5%, with every correctly-classified image certifiably robust up to 5952 label flips (i.e. certified accuracy is also 95.5% at 5952 label flips.) This means that the classifier is robust to adversarial label flips on 45.8% of the training data.

[6] also reports robustness certificates against label-flipping attacks on binary MNIST classification with classes 1 and 7. [6] reports clean-accuracy of 94.5% and certified accuracies for attack magnitudes up to 2000 label flips (out of 13007: 15.4%), with the best certified accuracy less than 70%.

F.3 Relationship to Randomized Ablation

As mentioned in the chapter introduction, (SS-)DPA is in some sense related to Randomized Ablation (Chapter 2; used in defense against sparse inference-time attacks) for training-time poisoning attacks. Randomized Ablation is a certified defense against ℓ_0 (sparse) inference attacks, in which the final classification is a consensus among classifications of copies of the image. In each copy, a fixed number of pixels are randomly ablated (replaced with a null value). A direct application of Randomized Ablation to poisoning attacks would require each base classifier to be trained on a *random* subset of the training data, with each base classifier's training set chosen randomly and independently. Due to the randomized nature of this algorithm, estimation

error would have to be considered in practice when applying Randomized Ablation using a finite number of base classifiers: this decreases the certificates that can be reported, while also introducing a failure probability to the certificates. By contrast, in our algorithms, the partitions are *deterministic* and use disjoint, rather than independent, samples. In this section, we argue that our derandomization has little effect on the certified accuracies compared to randomized ablation, even considering randomized ablation with no estimation error (i.e., with infinite base classifiers). In the poisoning case specifically, using additional base classifiers is expensive – because they must each be trained – so one would observe a large estimation error when using a realistic number of base classifiers. Therefore our derandomization can potentially improve the certificates which can be reported, while also allowing for exact certificates using a finite number of base classifiers.

For simplicity, consider the label-flipping case. In this case, the training set has a fixed size, m. Thus, Randomized Ablation bounds can be considered directly. A direct adaptation of Randomized Ablation would, for each base classifier, choose s out of m samples to retain labels for, and would ablate the labels for the rest of the training data. Suppose an adversary has flipped r labels. For each base classifier, the probability that a flipped label is used in classification (and therefore that the base classifier is 'poisoned') is:

$$\Pr(\text{poisoned})_{\text{RA}} = 1 - \frac{\binom{m-r}{s}}{\binom{m}{s}} \tag{F.11}$$

where "RA" stands for Randomized Ablation.

In this direct adaptation, one must then use a very large ensemble of randomized classifiers. The ensemble must be large enough that we can estimate with high confidence the probabilities

(on the distribution of possible choices of training labels to retain) that the base classifier selects each class. If the gap between the highest and the next-highest class probabilities can be determined to be greater than $2\Pr(\text{poisoned})_{\text{RA}}$, then the consensus classification cannot be changed by flipping r labels. This is because, at worst, every poisoned classifier could switch the highest-class classification to the runner-up class, reducing the gap by at most $2\Pr(\text{poisoned})$.

Note that this estimation relies on each base classifier using a subset of labels selected *randomly and independently* from the other base classifier. In contrast, our SS-DPA method selects each subset *disjointly*. If r labels are flipped, assuming that in the worst case each flipped label is in a different partition, using the union bound, the proportion of base classifiers which can be poisoned is

$$\Pr(\text{poisoned})_{\text{SS-DPA}} \leq \frac{r}{k} = \frac{rs}{m} \tag{F.12}$$

where for simplicity we assume that k evenly divides the number of samples m, so $s = m/k$ labels are kept by each partition. Again we need the gap in class probabilities to be at least $2\Pr(\text{poisoned})_{\text{SS-DPA}}$ to ensure robustness. While the use of the union bound might suggest that our deterministic scheme (Equation F.12) might lead to a significantly looser bound than that of the probabilistic certificate (Equation F.11), this is not the case in practice where $rs << m$. For example, in an MNIST-sized dataset ($m = 60000$), using $s = 50$ labels per base classifier, to certify for $r = 200$ label flips, we have $\Pr(\text{poisoned})_{\text{RA}} = 0.154$, and $\Pr(\text{poisoned})_{\text{SS-DPA}} \leq 0.167$. The derandomization only sightly increases the required gap between the top two class probabilities.

To understand this, note that if the number of poisonings r is small compared to the number of partitions k, then even if the partitions are random and independent, the chance that any two poisonings occur in the same partition is quite small. In that case, the union bound in Equation

F.12 is actually quite close to an independence assumption. By accepting this small increase in the upper bound of the probability that each base classification is poisoned, our method provides all of the benefits of de-randomization, including allowing for exact robustness certificates using only a finite number of classifiers. Additionally, note that in the Randomized Ablation case, the *empirical* gap in estimated class probabilities must be somewhat larger than $\Pr(\text{poisoned})_{\text{RA}}$ in order to certify robustness with high confidence, due to estimation error: the gap required increases more as the number of base classifiers decreases. This is particularly important in the poisoning case, because training a large number of classifiers is substantially more expensive than performing a large number of evaluations, as in randomized smoothing for evasion attacks.

We also note that Chapter 6 [29] also uses a de-randomized scheme based on Randomized Ablation to certifiably defend against evasion patch attacks. However, in that chapter, the de-randomization does not involve a union bound over *arbitrary* partitions of the vulnerable inputs. Instead, in the case of patch attacks, the attack is geometrically constrained: the image is therefore divided into geometric regions (bands or blocks) such that the attacker will only overlap with a fixed number of these regions. Each base classifier then uses only a *single* region to make its classification. Also, we note that we use from Chapter 6 the deterministic "tie-breaking" technique when evaluating the consensus class in Equation 7.4, which can increase our robustness certificate by up to one.

F.4 Relationship to Existing Ensemble Methods

As mentioned in the chapter introduction, our proposed method is related to classical ensemble approaches in machine learning, namely bootstrap aggregation ("bagging") and subset

aggregation ("subagging") [56, 111, 112, 113]. In these methods, each base classifier in the ensemble is trained on an independently sampled collection of points from the training set: this means that multiple classifiers in the ensemble may be trained on the same sample point. The purpose of these methods has typically been to improve generalization, and therefore to improve test set accuracy: bagging and subagging decrease the variance component of the classifier's error.

In subagging, each training set for a base classifier is an independently sampled subset of the training data: this is in fact an identical formulation to the "direct Randomized Ablation" approach discussed in Appendix F.3. However, in practice, the size of each training subset has typically been quite large: the bias error term increases with decreasing subsample sizes [112]. Thus, the optimal subsample size for maximum accuracy is large: [113] recommends using $s = m/2$ samples per classifier ("half-subagging"), with theoretical justification for optimal generalization. This would *not* be useful in Randomized Ablation-like certification, because any one poisoned element would affect half of the ensemble. Indeed, in our certifiably robust classifiers, we observe a trade-off between accuracy and certified robustness: our use of many very small partitions is clearly not optimal for the test-set accuracy (Table 7.1).

In bagging, the samples in each base classifier training set are chosen with replacement, so elements may be repeated in the training "set" for a single base classifier. Bagging has been proposed as an *empirical* defense against poisoning attacks [114] as well as for evasion attacks [115]. However, to our knowledge, these techniques have not yet been used to provide *certified* robustness.

Our approach also bears some similarity to Federated Averaging [172] in that base models are trained on disjoint partitions of the dataset. However, in Federated Averaging, the *model*

weights of many distributed base classifiers are periodically averaged and re-distributed during learning, in order to allow for efficient massively-parallel learning. No theoretical robustness guarantees are provided (as this is not the goal of the algorithm) and it would seem difficult to derive them, given that the relationship between model weights and final classification is highly non-linear in deep networks. By contrast, DPA uses the consensus of *final outputs* after all base classifiers are trained independently (or, in the SS-DPA case, independently for labeled data).

F.5 SS-DPA with Hashing

It is possible to use hashing, as in DPA, in order to partition data for SS-DPA: as long as the hash function $h(\boldsymbol{t})$ does not use the sample label in assigning a class (as ours indeed does not), it will always assign an image to the same partition regardless of label-flipping, so only one partition will be affected by a label-flip. Therefore, the SS-DPA label-flipping certificate should still be correct. However, as explained in the main text, treating the unlabeled data as trustworthy allows us to partition the samples evenly among partitions using sorting. This is motivated by the classical understanding in machine learning (e.g. [173]) that learning curves (the test error versus the number of samples that a classifier is trained on) tend to be convex-like. The test error of a base classifier is then approximately a convex function of that base classifier's partition size. Therefore, if the partition size is a random variable, by Jensen's inequality, the expected test error of the (random) partition size is greater than the test error of the mean partition size. Setting all base classifiers to use the mean number of samples should then maximize the average base classifier accuracy. To validate this reasoning, we tested SS-DPA with partitions determined by hashing (using the same partitions as we used in DPA), rather than the sorting method described

	Number of Partitions k	Median Certified Robustness		Clean Accuracy		Base Classifier Accuracy	
		Hash	**Sort**	**Hash**	**Sort**	**Hash**	**Sort**
MNIST, SS-DPA	1200	460	**485**	**95.74%**	95.62%	78.64%	**80.77%**
	3000	606	**645**	93.87%	**93.90%**	55.04%	**57.65%**
CIFAR, SS-DPA	50	**25**	**25**	**90.90%**	90.89%	89.00%	**89.06%**
	250	**124**	**124**	**90.36%**	90.33%	**86.34%**	86.25%
	1000	387	**392**	**89.11%**	89.02%	75.35%	**75.83%**
GTSRB, SS-DPA	50	**25**	**25**	96.98%	**97.09%**	96.33%	**96.35%**
	100	**50**	**50**	**96.79%**	96.76%	94.86%	**94.96%**
	200	**99**	**99**	**96.38%**	96.34%	91.39%	**91.54%**
	400	174	**176**	**95.89%**	95.80%	83.07%	**83.60%**

Table F.1: Comparison of SS-DPA with hashing ('**Hash**' columns, described in Appendix F.5) to the SS-DPA algorithm with partitions determined by sorting ('**Sort**' columns, described in the main text). Note that partitioning via sorting consistently results in higher base classifier accuracies, and can increase (and never decreases) median certified robustness. These effects seem to be larger on MNIST than on CIFAR-10.

in the main text. See Table F.1 for results. As expected, the average base classifier accuracy decreased in most (8/9) experiments when using the DPA hashing, compared to using the sorting method of SS-DPA. However, the effect was minimal in CIFAR-10 and GTSRB experiments: the main advantage of the sorting method was seen on MNIST. This is partly because we used more partitions, and hence fewer average samples per partition, in the MNIST experiments: fewer average samples per partition creates a greater variation in the number of samples per partition in the hashing method. However, CIFAR-10 with $k = 1000$ and MNIST with $k = 1200$ both average 50 samples per partition, but the base classifier accuracy difference still was much more significant on MNIST (2.13%) compared to CIFAR-10 (0.48%).

On the MNIST experiments, where the base classifier accuracy gap was observed, we also saw that the effect of hashing on the smoothed classifier was mainly to decrease the certified

robustness, and that there was not a significant effect on the clean smoothed classifier accuracy. As discussed in Appendix F.6, this may imply that the outputs of the base classifiers using the sorting method are more correlated, in addition to being more accurate.

F.6 Effect of Random Seed Selection

In Section 7.2.2.1, we mention that we "deterministically" choose different random seeds for training each partition, rather than training every partition with the same random seed. To see a comparison between using distinct and the same random seed for each partition, see Table F.2. Note that there is not a large, consistent effect across experiments on either the base classifier accuracy nor the the median certified robustness: however, in most experiments, the distinct random seeds resulted in higher smoothed classifier accuracies. This effect was particularly pronounced using SS-DPA on MNIST: using distinct seeds increased smoothed accuracy by at least 1% on each value of k on MNIST. This implies that shared random seeds make the base classifiers *more correlated* with each other: at the same level of average base classifier accuracy, it is more likely that a plurality of base classifiers will all misclassify the same sample (If the base classifiers were perfectly uncorrelated, we would see nearly 100% smoothed clean accuracy wherever the base classifier accuracy was over 50%. Also if they were perfectly correlated, the smoothed clean accuracy would equal the base classifier accuracy). Interestingly, the base classifier accuracy was significantly lower for both SS-DPA/MNIST experiments when using diverse random seeds; this defies any obvious explanation. However, this does make the correlation effect even more significant: for example, for $k = 3000$, the SS-DPA smoothed classifier accuracy is over 2% larger with distinct seeds, despite the fact that the base classifier is over 1% less accurate.

	Number of Partitions k	Median Certified Robustness		Clean Accuracy		Base Classifier Accuracy	
		Same	**Distinct**	**Same**	**Distinct**	**Same**	**Distinct**
MNIST, DPA	1200	443	**448**	95.33%	**95.85%**	76.21%	**76.97%**
	3000	**513**	509	92.90%	**93.36%**	49.52%	**49.54%**
MNIST, SS-DPA	1200	**501**	485	94.45%	**95.62%**	**82.31%**	80.77%
	3000	**663**	645	91.60%	**93.90%**	**59.33%**	57.65%
CIFAR, DPA	50	**9**	**9**	70.14%	**70.16%**	56.13%	**56.39%**
	250	**5**	**5**	55.24%	**55.65%**	34.96%	**35.17%**
	1000	**N/A**	**N/A**	43.96%	**44.52%**	23.14%	**23.20%**
CIFAR, SS-DPA	50	**25**	**25**	**90.92%**	90.89%	**89.08%**	89.06%
	250	**124**	**124**	90.25%	**90.33%**	**86.25%**	**86.25%**
	1000	391	**392**	88.97%	**89.02%**	75.82%	**75.83%**
GTSRB, DPA	50	**20**	**20**	88.59%	**89.20%**	73.89%	**73.94%**
	100	3	**4**	54.98%	**55.90%**	34.74%	**35.64%**
GTSRB, SS-DPA	50	**25**	**25**	97.06%	**97.09%**	**96.36%**	96.35%
	100	**50**	**50**	**96.79%**	96.76%	94.95%	**94.96%**
	200	**99**	**99**	**96.34%**	**96.34%**	**91.54%**	**91.54%**
	400	**176**	**176**	**95.85%**	95.80%	**83.64%**	83.60%

Table F.2: Comparison of DPA and SS-DPA algorithms using the same random seed for each partition ('**Same**' columns, described in Appendix F.6) to the DPA and SS-DPA algorithms using the distinct random seeds for each partition ('**Distinct**' columns, described in the main text). Note that using distinct random seeds usually results in higher smoothed classifier clean accuracies, and always does so when using DPA.

It is somewhat surprising that the base classifiers become correlated when using the same random seed, given that they are trained on entirely distinct data. However, two factors may be at play here. First, note that the random seed used in training controls the random cropping of training images: it is possible that, because the training sets of the base classifiers are so small, using the same cropping patterns in every classifier would create a systematic bias.

Note that the effect was least significant for SS-DPA on CIFAR-10 and GTSRB, with SimCLR. This may be due to the final supervised classifiers being linear classifiers, rather than deep networks, for these experiments.

F.7 SS-DPA with Repeated Unlabeled Data

The definition of a training set that we use, $T \in \mathcal{P}(\mathcal{S}_L)$, technically allows for repeated samples with differing labels: there could be a pair of distinct samples, $\boldsymbol{t}, \boldsymbol{t}' \in T$, such that $\textbf{sample}(\boldsymbol{t}) = \textbf{sample}(\boldsymbol{t}')$, but $\textbf{label}(\boldsymbol{t}) \neq \textbf{label}(\boldsymbol{t}')$. This creates difficulties with the definition of the label-flipping attack: for example, the attacker could flip the label of t to become the label of t': this would break the definition of T as a set. In most applications, this is not a circumstance that warrants practical consideration: indeed, none of the datasets used in our experiments have such instances (nor can label-flipping attacks create them), and therefore, for performance reasons, our implementation of SS-DPA does not handle these cases. Specifically, to optimize for performance, we verify that there are no repeated sample values, and then sort T itself (rather than $\textbf{samples}(T)$) lexicographically by image pixel values: this is equivalent to sorting $\textbf{samples}(T)$ if no repeated images occur — which we have already verified — and avoids an unnecessary lookup procedure to find the sorted index of the unlabeled sample for each labeled sample. How-

ever, the SS-DPA algorithm as described in Section 7.2.3 can be implemented to handle such datasets, under a formalism of label-flipping tailored to represent this edge case.

Specifically, we define the space of possible labeled data points as $\mathcal{S}'_L = \mathcal{S} \times \mathcal{P}(\mathbb{N})$: each labeled data point consists of a sample along with a *set* of associated labels. We then restrict our dataset T to be any subset of $\mathcal{S}'_L$ such that for all $\boldsymbol{t}, \boldsymbol{t}' \in T$, $\textbf{sample}(\boldsymbol{t}) \neq \textbf{sample}(\boldsymbol{t}')$. In other words, we do not allow repeated sample values in T as formally defined: if repeated samples exist, one can simply merge their sets of associated labels (in the formalism).

Note that using this definition, the size of T will equal the size of $\textbf{samples}(T)$, and the samples will always remain the same: the adversary can only modify the label sets of samples "in place". SS-DPA will always assign an unlabeled sample, *along with all of its associated labels*, to the same partition, regardless of any label flipping. In practice, this is because, as described in Section 7.2.3, the partition assignment of a labeled sample depends only on its sample value, not its label: all labeled samples with the same sample value will be put in the same partition. Note that this is true even if the implementation represents two identical sample values with different labels as two separate samples: one does not actually have to implement labels as sets. Therefore, any changes to any labels associated with a sample will only change the output of one base classifier, so all such changes can together be considered a single label-flip in the context of the certificate. In the above formalism, the certificate represents the number of samples in T whose label *sets* have been "flipped": i.e., modified in any way.

	Number of Partitions k	Median Certified Robustness		Clean Accuracy		Base Classifier Accuracy	
		Equal.	**-**	**Equal.**	**-**	**Equal.**	**-**
GTSRB, DPA	50	**23**	20	**91.84%**	89.20%	**80.98%**	73.94%
	100	**21**	4	**77.81%**	55.90%	**54.83%**	35.64%

Table F.3: Comparison of DPA algorithm with and without using histogram equalization as preprocessing on the GTSRB dataset. We find that histogram equalization substantially improves performance when $k = 100$, although the effect is more modest at $k = 50$.

F.8 GTSRB with Histogram Equalization

Because GTSRB contains images with widely varying lighting conditions, histogram equalization is sometimes applied as a preprocessing step when training classifiers on this dataset (for example, [174]). We tested this preprocessing with DPA, and found that it substantially improved the performance for larger k ($k = 100$). However, for $k = 50$, which had larger accuracy and median certified robustness with or without this preprocessing, the effect was modest (a 2.64% increase in clean accuracy, and an increase of 3 in certified robustness). The greater effect when k is large is likely because each partition contains fewer images of each class, so it is more likely for each base classifier that some lighting conditions are not represented in the training data for every class. Applying histogram equalization to the training and test data reduces this effect.

F.9 SimCLR Experimental Details

We used the PyTorch implementation of SimCLR provided by [175], with modifications to ensure determinism as described in the main text. For training the embeddings, we used a ResNet18 model with batch size of 512 for CIFAR-10 and 256 for GTSRB, initial learning rate

of 0.5, cosine annealing, and temperature parameter of 0.5, and trained for 1000 epochs. For learning the linear ensemble classifiers, we used a batch size of 512, initial learning rate of 1.0, and trained for 100 epochs.

F.10 GTSRB dataset details

The GTSRB dataset consists of images of variable sizes, from 15×15 to 250×250. We resized all images to 48×48 during training and testing (using bilinear interpolation). However, we wanted to ensure that our certificates still applied correctly to the original images. In particular, for SS-DPA, we sort the dataset using pixel values of the original image, with black padding for smaller images. This ensures that repeated images do not occur in the original dataset, (as described in Appendix F.7), even if the resized images could possibly contain repeats. For DPA, we hash using the sum of all pixels in the original image. Finally, for sorting to ensure determinism in training, we use the images after resizing (ensuring no repeated images is not important for this). For both NiN and SimCLR training, we excluded horizontal flips from data augmentation and contrastive learning, because some classes in GTSRB are in fact mirror-images of other classes.

Appendix G: Appendix to Chapter 8

G.1 Proofs

G.1.1 Proposition 8.1 (Gradient reward feedback example)

G.1.1.1 Problem setting

- $g, s, a \in \mathbb{R}^n$; $\|a\|_2 \leq 1$
- Single step reward $R(s, g) = g^T s$
- Transition function: $s_{t+1} := s_t + U a_t$, where $U \in \mathbb{R}^{n \times n}$ is an unknown orthogonal (rotation) matrix
- The "hypothesis class" consists of all environments with dynamics of this form: the learning task is to learn the unknown rotation matrix U. We therefore take as an "inductive bias" that the model class consists of a Q-function $Q_{\tilde{U}}$ and policy $\pi_{\tilde{U}}$ which are in the form of the optimal Q-function and policy for an estimate $\tilde{U} \in \mathbb{R}^{n \times n}$ of U (constrained to be orthogonal). The task is to learn this parameter. We assume that $Q_{\tilde{U}}$ and $\pi_{\tilde{U}}$ share the parameter estimate.
- Let $\tilde{U}$ be the "current" parameter estimate and $\bar{U}$ be the "target" parameter estimate for TD

updates.

- We assume actions in the replay buffer are in general position.

G.1.1.2 Analysis

The optimal Q-value function is of the form:

$$\begin{aligned} Q^*(s,a,g) &= \sum_{n=0}^{\infty} \gamma^n \left(g^T(s+Ua) + n\|g\|_2 \right) \\ &= \frac{g^T(s+Ua)}{1-\gamma} + \frac{\gamma\|g\|_2}{(1-\gamma)^2} \end{aligned} \tag{G.1}$$

The optimal policy π then takes the form:

$$a^* = \frac{U^T g}{\|g\|_2} \tag{G.2}$$

(Note that, because we are sharing the parameter $\tilde{U}$ between $Q_{\tilde{U}}$ and $\pi_{\tilde{U}}$, and because the optimal action for $Q_{\tilde{U}}$ can be written in closed form, we do not require a training step for $\pi_{\tilde{U}}$.)

The "standard" MSE Bellman error for a tuple (s, a, g, s') is

$$
\begin{aligned}
[Q_{\tilde{U}}(s,a,g) &- R(s',g) + \gamma Q_{\bar{U}}(s', \pi_{\bar{U}}(s,g), g)]^2 = \\
\left[\frac{g^T(s+\tilde{U}a)}{1-\gamma} + \frac{\gamma\|g\|_2}{(1-\gamma)^2} \right. &- \left. g^T s' + \gamma\left(\frac{g^T(s' + \bar{U}\frac{\bar{U}^T g}{\|g\|_2})}{1-\gamma} + \frac{\gamma\|g\|_2}{(1-\gamma)^2}\right)\right]^2 = \\
\left[\frac{g^T(s+\tilde{U}a)}{1-\gamma} + \frac{\gamma\|g\|_2}{(1-\gamma)^2} \right. &- \left. g^T s' + \frac{\gamma}{1-\gamma} g^T s' + \frac{\gamma}{1-\gamma}\|g\|_2 + \frac{\gamma^2\|g\|_2}{(1-\gamma)^2}\right]^2 = \text{(using } \bar{U}\bar{U}^T = I) \\
\left[\frac{g^T(s+\tilde{U}a)}{1-\gamma} + \frac{\gamma\|g\|_2}{(1-\gamma)^2} \right. &- \left. \frac{g^T s'}{1-\gamma} + \frac{\gamma\|g\|_2}{(1-\gamma)^2}\right]^2 = \\
\left[\frac{g^T(s+\tilde{U}a)}{1-\gamma} \right. &- \left. \frac{g^T s'}{1-\gamma}\right]^2 = \\
\left[g^T(s+\tilde{U}a) \right. &- \left. g^T s'\right]^2 = \text{(dropping constant factor)} \\
\left[g^T\tilde{U}a \right. &- \left. g^T(s'-s)\right]^2 = \\
\left[g^T\tilde{U}a \right. &- \left. g^T U a\right]^2
\end{aligned}
$$

This is equivalent to fitting the bilinear form $g^T\tilde{U}a$ using the observed scalar $g^T U a$ $(= g^T(s'-s))$.

Noting that orthogonal matrices have $\sim n^2/2$ degrees of freedom, **this will require at least** $O(n^2)$ **samples to learn**.

Now consider our gradient-based Bellman error:

$$\|\nabla_g Q_{\tilde{U}}(s,a,g) - \nabla_g[R(s',g) + \gamma Q_{\bar{U}}(s', \pi_{\bar{U}}(s,g),g)]\|_2^2 =$$

$$\left\|\nabla_g\left[\frac{g^T(s+\tilde{U}a)}{1-\gamma} + \frac{\gamma\|g\|_2}{(1-\gamma)^2}\right] - \nabla_g\left[g^T s' + \gamma\left(\frac{g^T(s' + \bar{U}\frac{\bar{U}^T g}{\|g\|_2})}{1-\gamma} + \frac{\gamma\|g\|_2}{(1-\gamma)^2}\right)\right]\right\|_2^2 =$$

$$\left\|\nabla_g\left[\frac{g^T(s+\tilde{U}a)}{1-\gamma}\right] - \nabla_g\left[\frac{g^T s'}{1-\gamma}\right]\right\|_2^2 =$$

$$\left\|\frac{(s+\tilde{U}a)}{1-\gamma} - \frac{s'}{1-\gamma}\right\|_2^2 =$$

$$\left\|(s+\tilde{U}a) - s'\right\|_2^2 = \text{(dropping constant factor)}$$

$$\left\|\tilde{U}a - (s'-s)\right\|_2^2 =$$

$$\left\|\tilde{U}a - Ua\right\|_2^2$$

Here, we are fitting the linear form $\tilde{U}a$ to the observed vector $(s'-s)$. Assuming general position, **this can be solved with $O(n)$ samples!**

G.1.2 Proposition 8.2 (No reward gradient case example)

G.1.2.1 Problem setting

- $g, a \in \mathbb{R}^n; \|a\|_2 \leq 1$; $s \in \mathbb{R}^{2n}$; the state vector consists of two halves, which we denote s^1 and s^2.
- Single step reward $R(s,g) = g^T s^1$

- Transition function:
  - If $s_t^1 \neq \mathbf{0}$, then $s_{t+1}^1 := \mathbf{0}$, $s_{t+1}^2 := s_t^1 + Ua_t$. (Note that the reward is always zero here.)
  - If $s_t^1 = \mathbf{0}$, then $s_{t+1}^1 := s_t^2$, $s_{t+1}^2 := \mathbf{0}$,

  where $U \in \mathbb{R}^{n \times n}$ is an unknown orthogonal (rotation) matrix
- We make the same assumptions about the hypothesis class and inductive bias as in the last example, and also assume general position.

G.1.3 Analysis

Optimal Q-function

$$
\begin{aligned}
Q^*(s,a,g) &= \begin{cases} \sum_{n \text{ odd}}^{\infty} \gamma^n \left(g^T(s^1 + Ua) + \frac{n-1}{2} \|g\|_2 \right) & \text{if } s^1 \neq \mathbf{0} \\ \sum_{n \text{ even}}^{\infty} \gamma^n \left(g^T s^2 + \frac{n}{2} \|g\|_2 \right) & \text{if } s^1 = \mathbf{0} \end{cases} \\
&= \begin{cases} \sum_{m=0}^{\infty} \gamma^{2m+1} \left(g^T(s^1 + Ua) + m\|g\|_2 \right) & \text{if } s^1 \neq \mathbf{0} \\ \sum_{m=0}^{\infty} \gamma^{2m} \left(g^T s^2 + m\|g\|_2 \right) & \text{if } s^1 = \mathbf{0} \end{cases} \\
&= \begin{cases} \frac{\gamma g^T (s^1 + Ua)}{1-\gamma^2} + \frac{\gamma^3 \|g\|_2}{(1-\gamma^2)^2} & \text{if } s^1 \neq \mathbf{0} \\ \frac{g^T s^2}{1-\gamma^2} + \frac{\gamma^2 \|g\|_2}{(1-\gamma^2)^2} & \text{if } s^1 = \mathbf{0} \end{cases}
\end{aligned}
\tag{G.3}
$$

The optimal policy π then takes the form:

$$
a^* = \frac{U^T g}{\|g\|_2} \tag{G.4}
$$

(Note that if $s^1 = \mathbf{0}$, then the action does not appear in the Q function, so any action is optimal.)

We now consider the standard TD error:

$$[Q_{\tilde{U}}(s,a,g) \;-\; R(s',g) + \gamma Q_{\bar{U}}(s', \pi_{\bar{U}}(s,g), g)]^2 \tag{G.5}$$

Note that if $s^1 = \mathbf{0}$, then the trainable parameter $\tilde{U}$ does not appear anywhere in the expression of $Q_{\tilde{U}}(s,a,g)$. Then no learning can occur from these tuples; we can instead only consider the case where $s^1 \neq 0$. In this case, the immediate reward is always zero, and we also know that $s^{1\prime} = \mathbf{0}$ so the "standard" TD update is:

$$[Q_{\tilde{U}}(s,a,g) \;-\; \gamma Q_{\bar{U}}(s', \pi_{\bar{U}}(s,g), g)]^2 =$$

$$\left[\frac{\gamma g^T(s^1 + \tilde{U}a)}{1-\gamma^2} + \frac{\gamma^3 \|g\|_2}{(1-\gamma^2)^2} \;-\; \gamma\left(\frac{g^T s^{2\prime}}{1-\gamma^2} + \frac{\gamma^2\|g\|_2}{(1-\gamma^2)^2}\right)\right]^2 =$$

$$\left[\frac{\gamma g^T(s^1 + \tilde{U}a)}{1-\gamma^2} \;-\; \frac{\gamma g^T s^{2\prime}}{1-\gamma^2}\right]^2 =$$

$$\left[g^T(s^1 + \tilde{U}a) \;-\; g^T s^{2\prime}\right]^2 = \text{(dropping constant factor)}$$

$$\left[g^T\tilde{U}a \;-\; g^T(s^{2\prime} - s^1)\right]^2 =$$

$$\left[g^T\tilde{U}a \;-\; g^T U a\right]^2$$

This is the same update as in the previous example, and again **we need $O(n^2)$ samples**. (Note that there is a constant factor of 2 increase in the number of needed samples, due to the wasted

samples in which $s^1 = \mathbf{0}$. However, assuming general position, this will only account for half of the replay buffer.)

We now consider the case in which we use our gradient TD update. Again we only can use the tuples where $s^1 \neq 0$. For all of these, the immediate reward is zero, so we can use the gradient-based update for all of them.

$$
\begin{aligned}
\|\nabla_g Q_{\tilde{U}}(s,a,g) &- \gamma \nabla_g Q_{\bar{U}}(s', \pi_{\bar{U}}(s,g), g)\|_2^2 = \\
\left\| \nabla_g \left(\frac{\gamma g^T (s^1 + \tilde{U}a)}{1-\gamma^2} + \frac{\gamma^3 \|g\|_2}{(1-\gamma^2)^2} \right) \right. &- \left. \gamma \nabla_g \left(\frac{g^T s^{2\prime}}{1-\gamma^2} + \frac{\gamma^2 \|g\|_2}{(1-\gamma^2)^2} \right) \right\|_2^2 = \\
\left\| \frac{\gamma(s^1 + \tilde{U}a)}{1-\gamma^2} \right. &- \left. \frac{\gamma s^{2\prime}}{1-\gamma^2} \right\|_2^2 = \\
\left\| (s^1 + \tilde{U}a) \right. &- \left. s^{2\prime} \right\|_2^2 = \text{ (dropping constant factor)} \\
\left\| \tilde{U}a \right. &- \left. (s^{2\prime} - s^1) \right\|_2^2 = \\
\left\| \tilde{U}a \right. &- \left. Ua \right\|_2^2
\end{aligned}
$$

As in the previous gradient feedback case, assuming general position, **this can be solved with $O(n)$ samples!**

G.2 ReenGAGE for Discrete Actions

Note that Equation 8.8 requires that the gradient:

$$\nabla_g Q_{\theta'}(s', \pi_{\phi'}(s', g), g) \tag{G.6}$$

be computable. This means that $\pi_{\phi'}(s', g)$ must be continuous and differentiable, which implies a continuous action space. To extend our method to discrete action spaces, we instead consider the target Q-value of DQN [176], the standard baseline method for discrete Q-learning, in a goal-conditioned setting:

$$R(s', g) + \gamma \max_a Q_{\theta', a}(s', g), \tag{G.7}$$

where $Q_\theta \in S \times G \to \mathbb{R}^{|A|}$. This is not differentiable everywhere with respect to g: in particular, at points where, for some pair of actions a', a'', we have:

$$\max_a Q_{\theta', a}(s', g) = Q_{\theta', a'}(s', g) = Q_{\theta', a''}(s', g), \tag{G.8}$$

the gradient with respect to g is not necessarily defined. In practice, this means that naively using auto-differentiation to take the gradient of Equation G.7 produces the gradient of the target with respect to the goal *assuming the optimal action remains constant*. To overcome this, we consider instead using a *soft target*:

$$\begin{aligned} \text{SoftQTarget}(s', g) :=& R(s', g) + \gamma \, \text{SoftMax}[Q_{\theta'}(s', g)/\tau] \cdot Q_{\theta'}(s', g) \\ =& R(s', g) + \gamma \frac{\sum_{a \in A} Q_{\theta', a}(s', g) e^{Q_{\theta', a}(s', g)/\tau}}{\sum_{a' \in A} e^{Q_{\theta', a'}(s', g)/\tau}}, \end{aligned} \tag{G.9}$$

Where τ is the temperature hyperparameter. (Note that this approaches the standard DQN target as $\tau \to 0$.) We tested three uses of this soft target:

- Soft target for *gradient loss only*:

$$\mathcal{L} = \mathop{\mathbb{E}}_{(s,a,s',g)\sim\text{Buffer}} \Bigg[\mathcal{L}_{\text{Huber}}\Big[Q_{\theta,a}(s,g), R(s',g) + \gamma \max_a Q_{\theta',a}(s',g)\Big] + \alpha \mathcal{L}_{\text{mse}}\Big[\nabla_g Q_{\theta,a}(s,g), \gamma \nabla_g \text{SoftQTarget}(s',g)\Big]\Bigg] \tag{G.10}$$

  Following [176], we use the Huber loss for the main loss term, although we use MSE for the gradient loss.

- Soft target for *both loss terms*:

$$\mathcal{L} = \mathop{\mathbb{E}}_{(s,a,s',g)\sim\text{Buffer}} \Bigg[\mathcal{L}_{\text{Huber}}\Big[Q_{\theta,a}(s,g), \text{SoftQTarget}(s',g)\Big] + \alpha \mathcal{L}_{\text{mse}}\Big[\nabla_g Q_{\theta,a}(s,g), \gamma \nabla_g \text{SoftQTarget}(s',g)\Big]\Bigg] \tag{G.11}$$

- Soft target for both loss terms, *and* take actions nondeterministically, with a probability distribution given by $\text{SoftMax}[Q_{\theta'}(s',g)/\tau]$.

We test on an implementation of the "Bit-Flipping" sparse-reward environment from [50], with dimensionality $d = 40$ bits using DQN with HER as the baseline. Because code for this experiment is not provided by [50], we re-implemented the experiment using the Stable-Baselines3 package [177]. We used the hyperparameters specified by [50], with the following minor modifications (note that for some of these specifications, [50] is unclear about whether the specification was applied for the Bit-Flipping environment, or only in the continuous control, DDPG, environ-

ments tested in that work):

- We train on a single GPU, rather than averaging gradient updates from 8 workers.
- While we use Adam optimizer with learning rate of 0.001 as specified, we use PyTorch defaults for other Adam hyperparameters, rather than TensorFlow defaults.
- We do not clip the target Q-values (this feature is not part of the DQN implementation of [177]).
- We evaluate using the current, rather than target, Q-value function.
- We do not normalize observations (which are already $\{0, 1\}$).

The baselines we present are from our re-implementation, to provide a fair comparison. For our method, we performed a grid search on $\alpha \in \{0.5, 1.0\}$ and $\tau \in \{0.0, 0.5, 1.0\}$. Results are presented in Figure G.1. We observed that the "gradient loss only" method was effective at improving performance, both over standard DQN and over ReenGAGE with standard DQN targets ($\tau = 0$). By contrast, using soft targets for both loss terms made the performance worse, and using nondeterministic actions with soft targets for both loss terms brought the success rate to zero (and hence is not shown). We tested all three methods with $d = 40$ bits, and then applied the successful method ("gradient loss only") to $d = 20$ as well; improvements over baseline for $d = 20$ were minor.

G.3 Hyperparameters and Full Results for ContinuousSeek

As mentioned in the main text, we performed a full hyperparameter search over the batch size (in $\{128, 256, 512\}$) and learning rate (in $\{0.00025, 0.0005, 0.001, 0.0015\}$). The results in

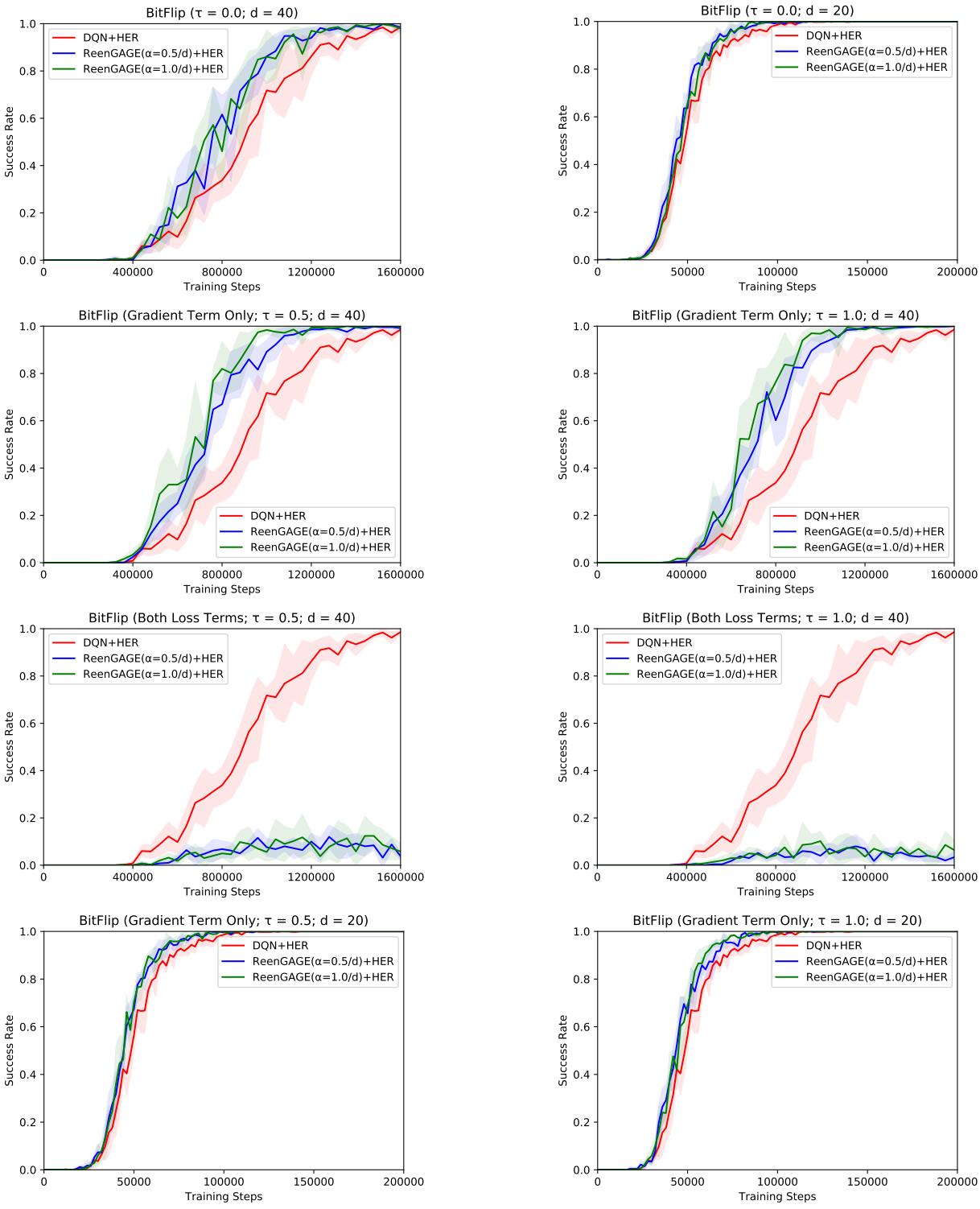


Figure G.1: Results for the Bit-Flipping environment.

Replay Buffer Size	1000000
Frequency of Training	Every 1 environment step
Gradient Descent Steps per Training	1
Initial Steps before Training	1000
Discount γ	0.95
Polyak Update τ	0.005
Normal action noise for training σ	0.03
Architecture (both actor and critic)	Fully Connected; 2 hidden layers of width 256; ReLU activations
HER number relabeled goals k	4
HER relabeling strategy	'Future'
Evaluation episodes	50
Evaluation Frequency	Every 2000 environment steps

Table G.1: Hyperparameters for ContinuousSeek

the main paper represent the best single curve (as defined by area under the curve, or in other words best average score over time) for the baseline and each value of the ReenGAGE α term. Complete results are presented in Figures G.2, G.3, and G.4. The values of these parameters which yielded the best average performance, and were therefore reported in the main text, were as follows (Table G.2):

	DDPG+HER		ReenGAGE(α = 0.1)+HER		ReenGAGE(α = 0.2)+HER		ReenGAGE(α = 0.3)+HER	
	Batch	LR	Batch	LR	Batch	LR	Batch	LR
d = 5	512	0.001	512	0.0015	512	0.0015	512	0.0015
d = 10	256	0.0005	512	0.001	512	0.001	512	0.001
d = 20	256	0.0005	128	0.0005	128	0.0005	128	0.0005

Table G.2: "Best" batch sizes and learning rates for ContinuousSeek for DDPG and ReenGAGE.

Note that for the larger-scale experiments (d = 10 and d = 20) the "best" hyperparameters for the baseline DDPG+HER model lie in the interior of the search space of both the batch size and learning rate: this would seem to imply (assuming concavity) that increasing the range of

the search space of these parameters would likely not improve the baseline. However, this is not the case for ReenGAGE: we could perhaps achieve even better performance for ReenGAGE by conducting a larger search, specifically in the batch size dimension.

Other hyperparameters were fixed, and are listed in Table G.1. We use the implementation of HER with DDPG from Stable-Baselines3 [177] as our baseline; any unlisted hyperparameters are the default from this package. Note that we use the "online" variant of HER provided in the Stable-Baselines3 package.

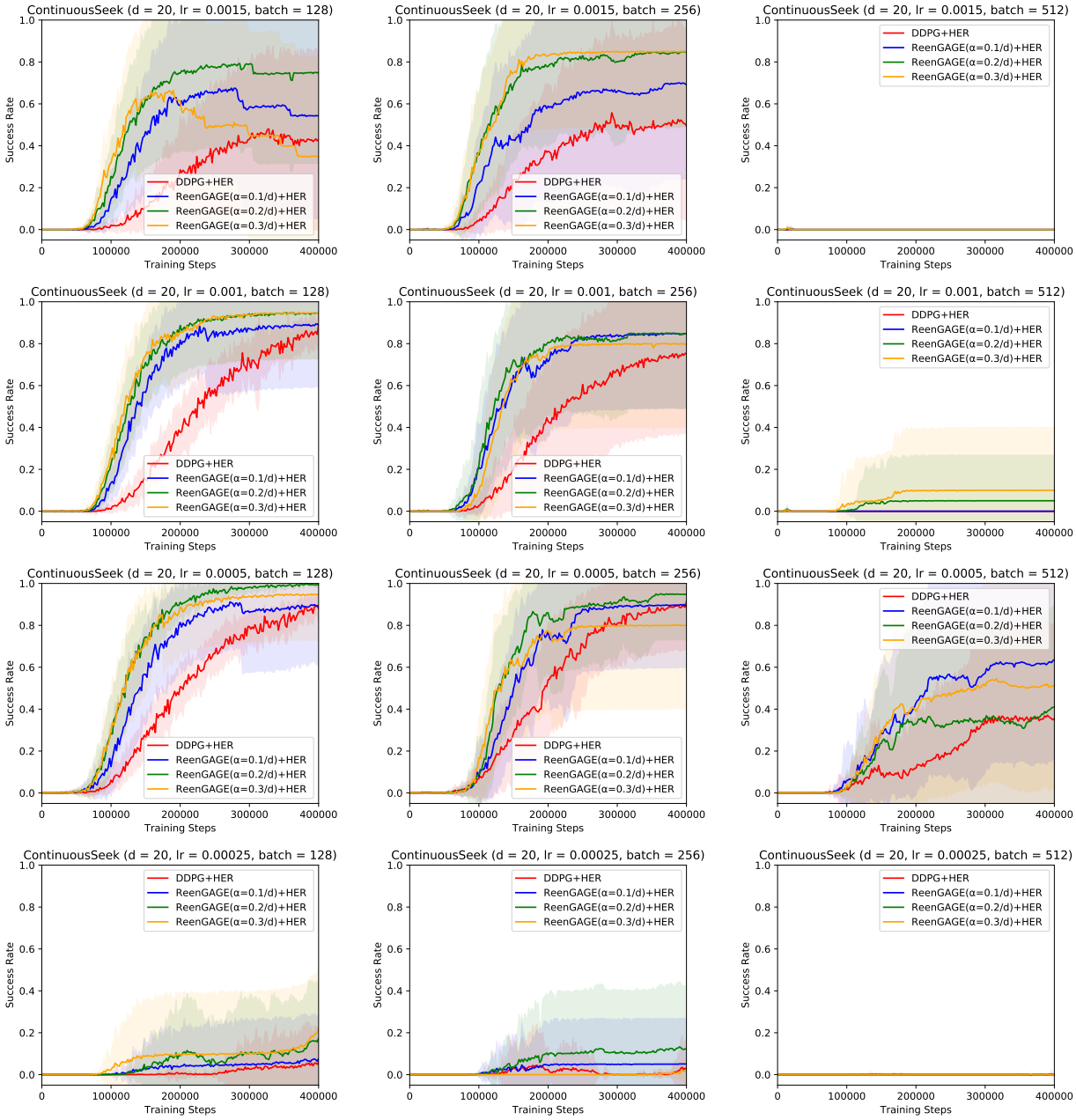


Figure G.2: Complete ContinuousSeek Results for $d = 20$.

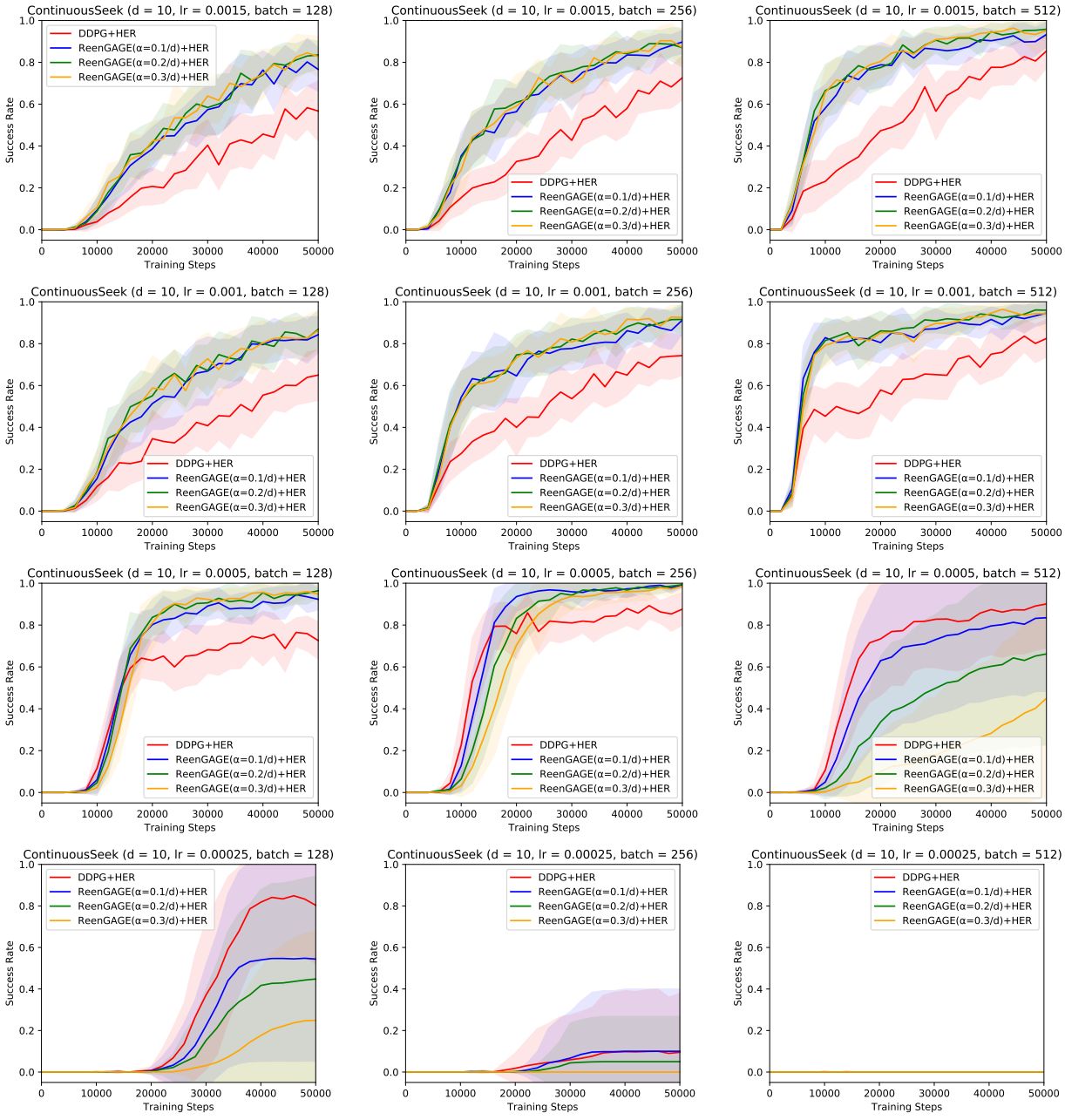


Figure G.3: Complete ContinuousSeek Results for $d = 10$.

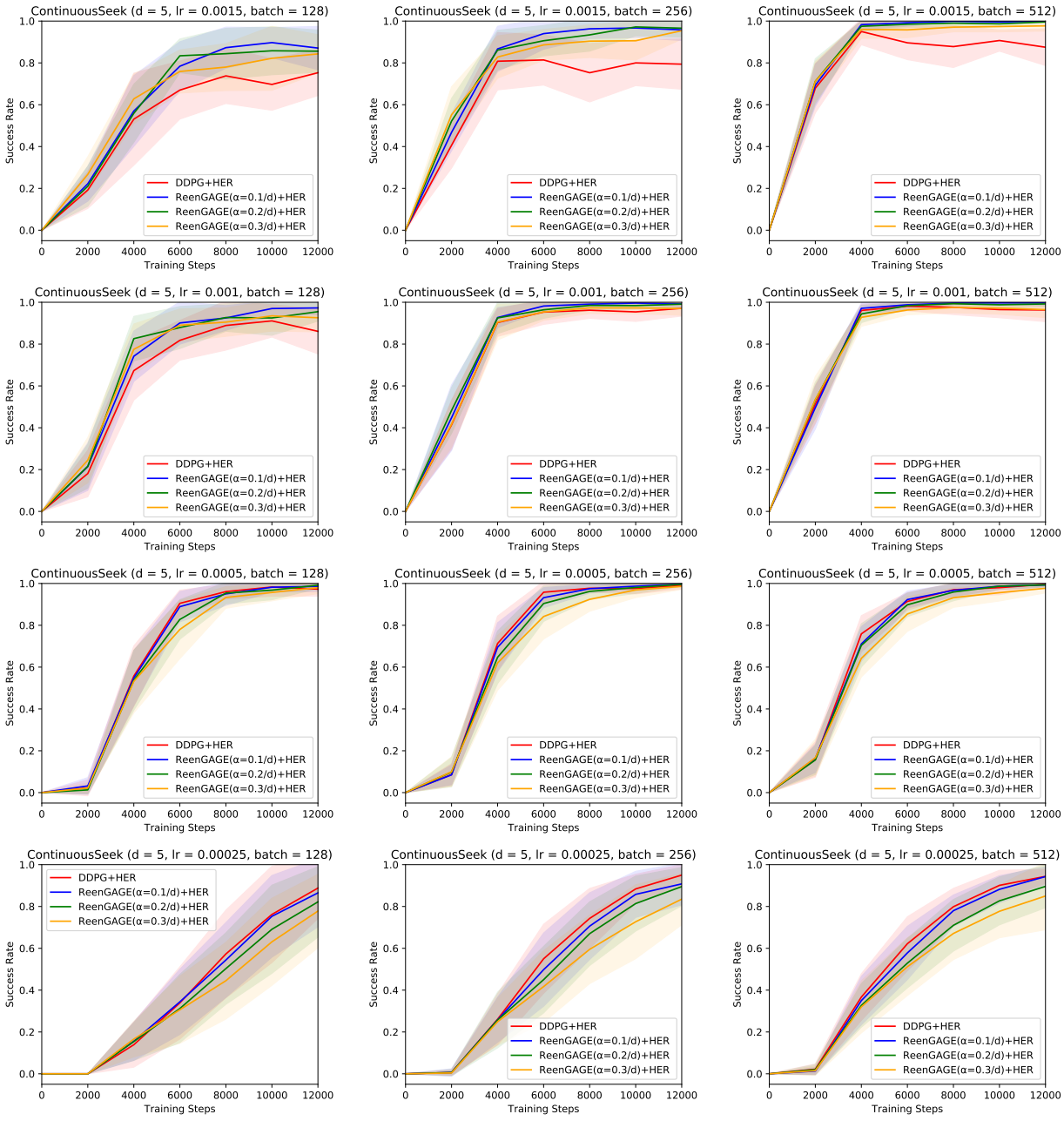


Figure G.4: Complete ContinuousSeek Results for $d = 5$.

G.4 Additional Results for Robotics Experiments

In Figure G.5, we provide additional results for the robotics experiments. In Figure G.5-(a) and (b), we give results for the **HandManipulateBlock** environment: as mentioned in the text, we did not see a significant advantage to using ReenGAGE for this environment.

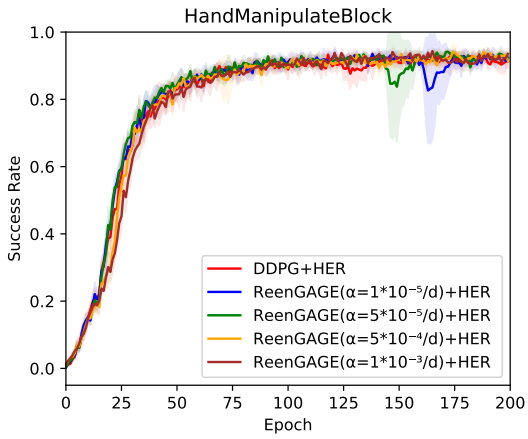
In addition to the lower-dimensional goal space of this problem ($d = 7$, versus $d = 15$ for HandReach), one additional possible reason why ReenGAGE may underperform in this setting is that, unlike in the HandReach case, the dimensions of the goal vector represent diverse quantities of significantly varying scales: some represent angular measurements, while others represent position measurements.

In order to handle this, we attempted normalizing each dimension of the ReenGAGE loss term. We accomplish this by multiplying the MSE loss for each coordinate i by σ_i^2, where σ_i is the running average standard deviation of dimension i of the goals. ([148]'s implementation already computes these averages in order to normalize inputs to neural networks.) Intuitively, we do this because the derivative in a given coordinate is in general inversely proportional to the scale of that coordinate (i.e., if $y = 2x$, then $\frac{df}{dy} = 0.5\frac{df}{dx}$), and because the ReenGAGE MSE loss in each dimension is proportional to the square of the derivative.

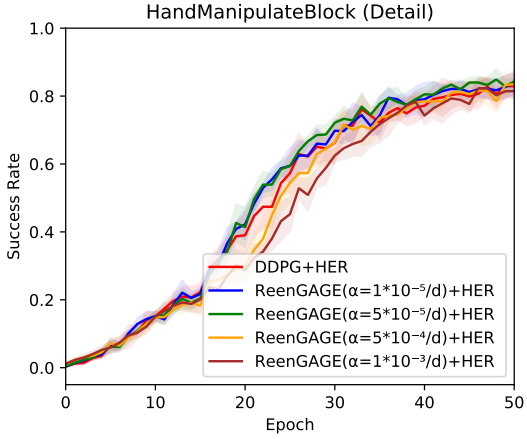
However, unfortunately, this did not result in superior performance for HandManipulate-Block: see Figure G.5-(c) and (d). As a sanity check, we confirmed that this method performed similarly to the non-normalized ReenGAGE method on HandReach (G.5-(e)).

Finally, in Figure G.5-(f), we show some additional experiments on HandReach. Specifically, we show results using a single random seed for a wider range of α than shown in the main text: this was performed as a first pass to find the appropriate range of the hyperparameter α

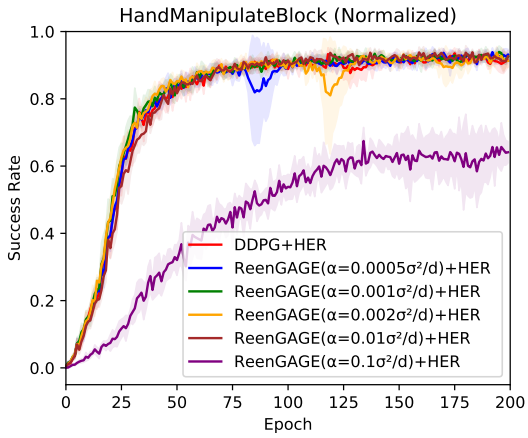
to use for the complete experiments. As shown in the main text results, ReenGAGE becomes unstable if too-high values of α are used.



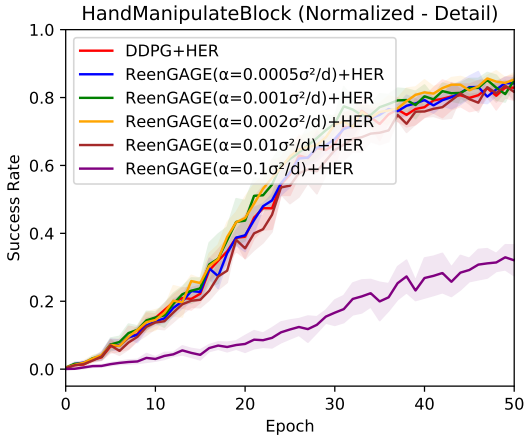
(a)



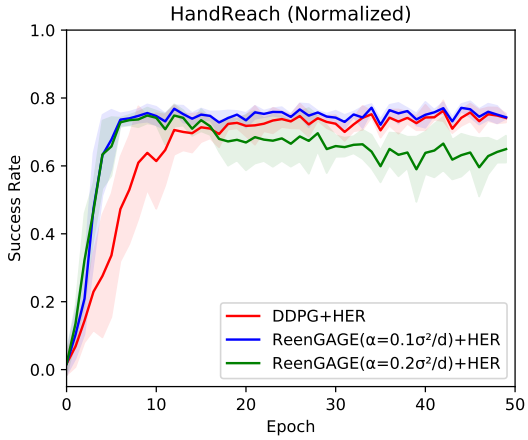
(b)



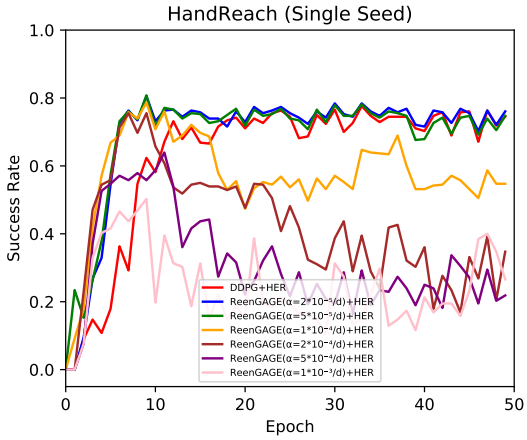
(c)



(d)



(e)



(f)

Figure G.5: Additional Results from Robotics experiments. See text for details.

G.5 ReenGAGE with SAC

In this section, we explore applying ReenGAGE on top of SAC [141], a variant of DDPG which uses a stochastic policy and rewards for high-entropy polices, uses the current policy to compute targets, and also uses an ensemble critic architecture. The application of ReenGAGE to SAC is straightforward: when computing estimates for $\nabla_g \text{Targ.}(g; s, a)$, we use the well-known "reparameterization trick" to differentiate through the stochastic policy (which depends on the goal) and we also differentiate through the entropy reward (which depends on policy's action distribution, and therefore on the goal).

In our experiments, we combine ReenGAGE with SAC and HER, test on our ContinuousSeek environment, and compare to an SAC+HER baseline. Hyperparameter-optimized "best" results for each value of α are shown in Figure G.6. As in our DDPG experiments, we performed a grid search over batch size and learning rate hyperparameters. However, using the range of these parameters we used for DDPG led to all best baselines and ReenGAGE models having values of the hyperparameters lying at the "edges" and "corners" of the hyperparameter grid search space. We therefore increased the search space size, testing all learning rates in $\{0.00025, 0.0005, 0.001, 0.0015, 0.0025\}$ and batch sizes in $\{128, 256, 512, 1024, 2048, 4096, 8192\}$. This led to optimal hyperparameters for the baseline in the interior of the search space for the larger scale experiments (d = 10 and d = 20); see Table G.3. As with the DDPG ContinuousSeek experiments, the ReenGAGE models still lie on edges/corners of the hyperparameter search space, so it may be possible to get even better ReenGAGE performance by increasing the search space further. Full results are presented in Figures G.7, G.8, and G.9. Note that due to computational limits, we only use 10 random seeds in these experiments, as opposed to 20 for

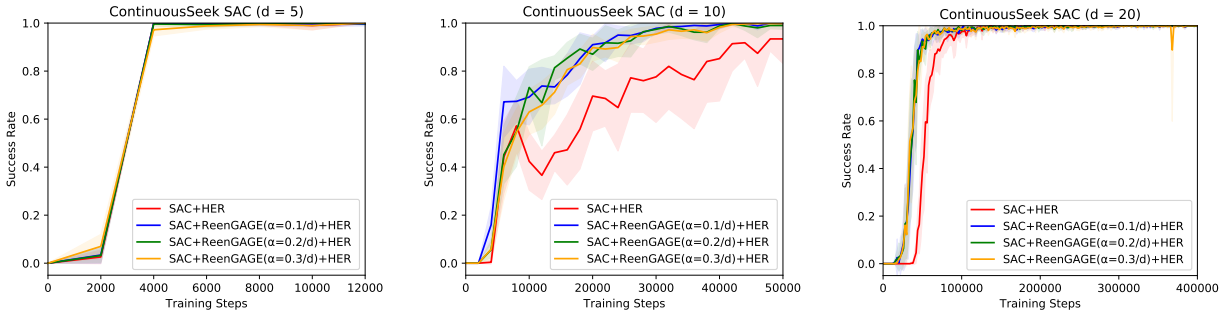


Figure G.6: ContinuousSeek results with SAC. Lines show the mean and standard deviation over 10 random seeds (kept the same for all experiments.) The Y-axis represents the success rate, defined as the fraction of test episodes for which the goal is ever reached.

the DDPG experiments.

Other, non-optimized hyperparameters were fixed, and are generally the same as those for DDPG listed in Table G.1. As with DDPG, we use the implementation of SAC from Stable-Baselines3 [177] as our baseline; any unlisted hyperparameters are the default from this package.

	SAC+HER		SAC+ReenGAGE(α=.1)+HER		SAC+ReenGAGE(α=.2)+HER		SAC+ReenGAGE(α=.3)+HER	
	Batch	LR	Batch	LR	Batch	LR	Batch	LR
d=5	8192	0.0025	4096	0.0025	2048	0.0025	2048	0.0025
d=10	1024	0.0015	1024	0.0025	512	0.0025	512	0.0025
d=20	4096	0.0005	8192	0.0025	8192	0.0025	8192	0.0025

Table G.3: "Best" batch sizes and learning rates for ContinuousSeek for SAC and SAC+ReenGAGE.

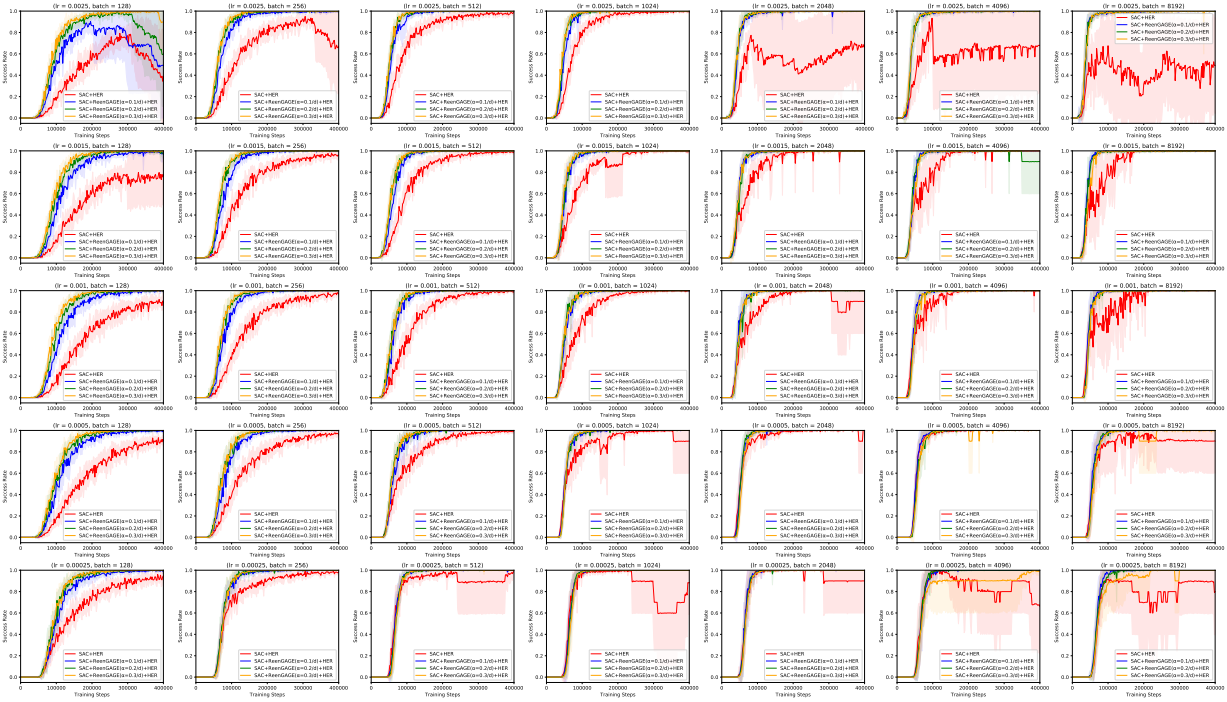


Figure G.7: Complete SAC ContinuousSeek results for $d = 20$.

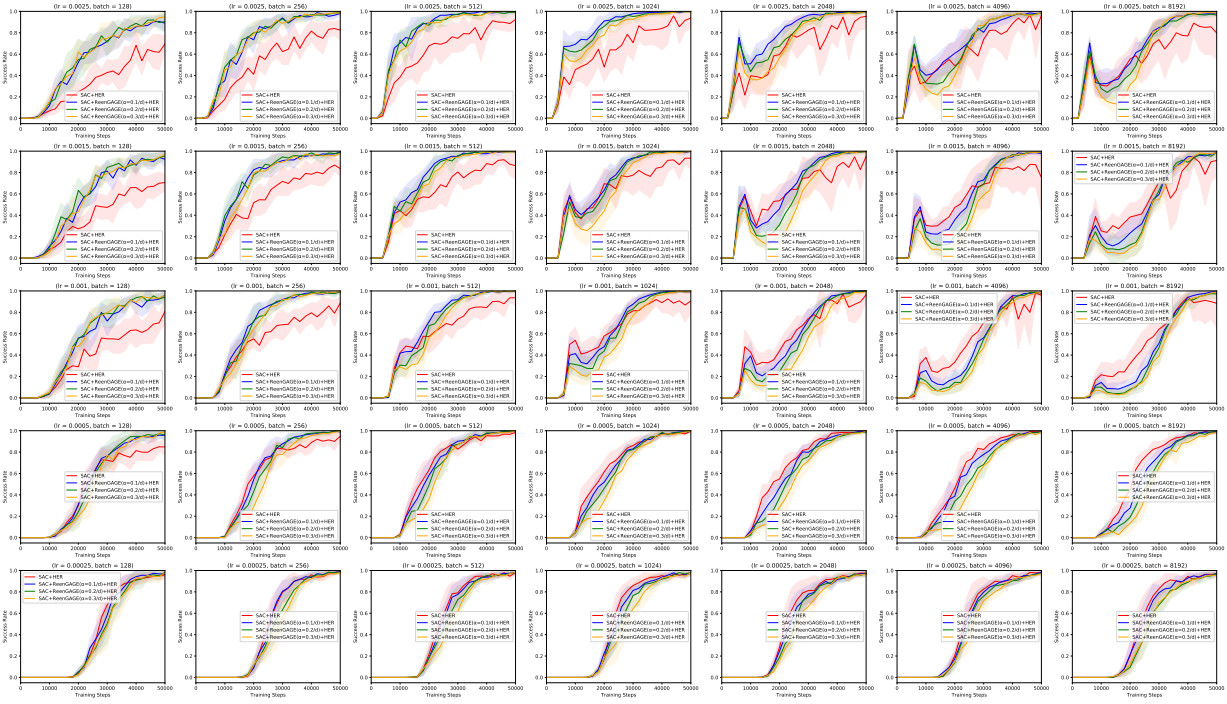


Figure G.8: Complete SAC ContinuousSeek results for $d = 10$.

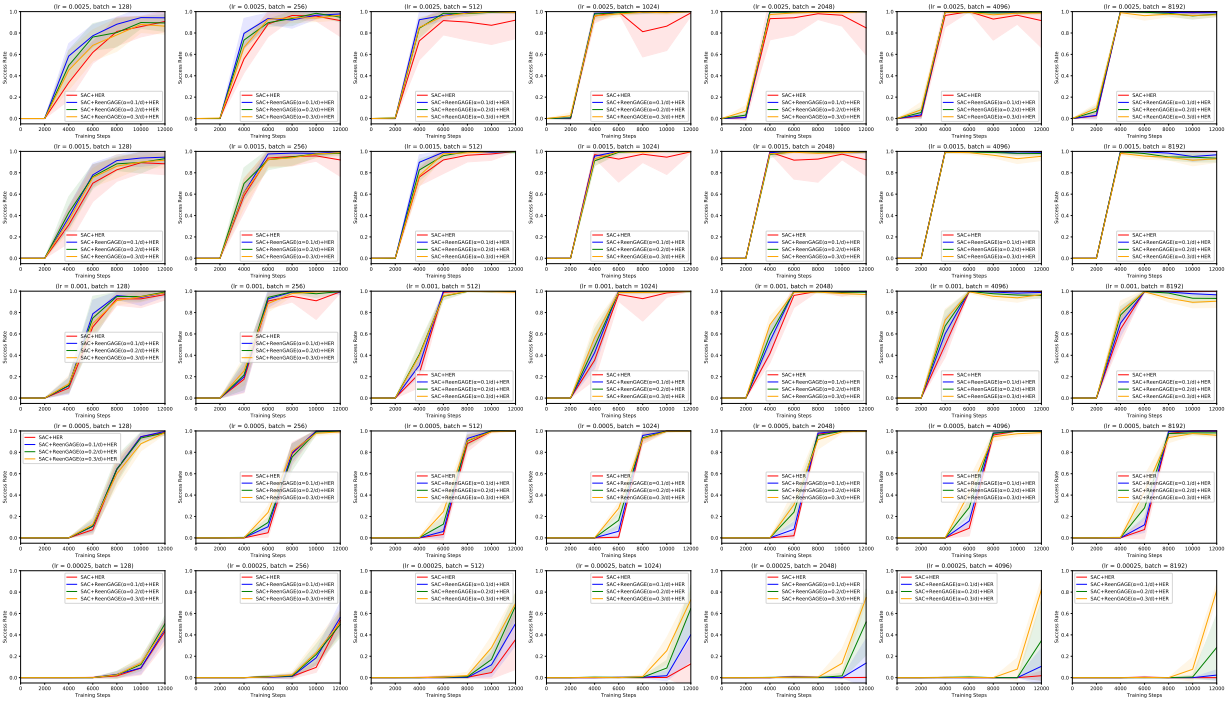


Figure G.9: Complete SAC ContinuousSeek results for $d = 5$.

G.6 Multi-ReenGAGE Implementation Details

G.6.1 Batch Implementation

To efficiently implement the multi-goal Q-function in Equation 8.15 as a batch equation, we use a constant-size representation of the goal set $g = \{g_1, ..., g_n\}$. Specifically, we let $g = \{g_1, ..., g_{n_{\max}}\}$, where $\{g_{n+1}, ..., g_{n_{\max}}\}$ are set to a dummy value (practically, $\mathbf{0}$), and the gate variables $\{b_{n+1}, ..., b_{n_{\max}}\}$ are all set to zero. Then:

$$Q_\theta(s, a, g) := Q^{\text{head}}_{\theta^{\text{h.}}}(s, a, \sum_{i=1}^{n}[b_i Q^{\text{encoder}}_{\theta^{\text{e.}}}(s, g_i)]) = Q^{\text{head}}_{\theta^{\text{h.}}}(s, a, \sum_{i=1}^{n_{\max}}[b_i Q^{\text{encoder}}_{\theta^{\text{e.}}}(s, g_i)]). \tag{G.12}$$

However, while this is equal to the intended form of the Q-function without the dummy

inputs, the gradient with respect to the full vector b differs from the intended form. Specifically, note that:

$$\frac{\partial Q_\theta(s,a,g)}{\partial b_i} = Q^{\text{encoder}}_{\theta^{\text{e.}}}(s,g_i) \cdot (\nabla Q^{\text{head}}_{\theta^{\text{h.}}})(s,a,\sum_{i=1}^{n_{\max}}[b_i Q^{\text{encoder}}_{\theta^{\text{e.}}}(s,g_i)]) \tag{G.13}$$

In particular, this is nonzero even when b_i is zero, so the gradient will depend on the **dummy** value $Q^{\text{encoder}}_{\theta^{\text{e.}}}(s,\mathbf{0})$. This is clearly not intended. To prevent this, we instead use the form:

$$Q^{\text{head}}_{\theta^{\text{h.}}}(s,a,\sum_{i=1}^{n_{\max}}[b_i^2 Q^{\text{encoder}}_{\theta^{\text{e.}}}(s,g_i)]). \tag{G.14}$$

(For the target, we construct the policy network similarly using b_i^2's). Note that the above form of the Q-function is equal to the intended form of the Q-function, because $0^2 = 0$ and $1^2 = 1$. However, in this case:

$$\begin{aligned}
&\frac{\partial Q_\theta(s,a,g)}{\partial b_i} = \\
&2b_i Q^{\text{encoder}}_{\theta^{\text{e.}}}(s,g_i) \cdot (\nabla Q^{\text{head}}_{\theta^{\text{h.}}})(s,a,\sum_{i=1}^{n_{\max}}[b_i Q^{\text{encoder}}_{\theta^{\text{e.}}}(s,g_i)]) = \\
&\begin{cases}
2Q^{\text{encoder}}_{\theta^{\text{e.}}}(s,g_i) \cdot (\nabla Q^{\text{head}}_{\theta^{\text{h.}}})(s,a,\sum_{i=1}^{n_{\max}}[b_i Q^{\text{encoder}}_{\theta^{\text{e.}}}(s,g_i)]) & \text{if } b_i = 1 \\
0 & \text{if } b_i = 0
\end{cases}
\end{aligned} \tag{G.15}$$

Which is the intended gradient of the Q-function without the dummy inputs, times a constant factor of two.

For the reward, we also use:

$$R(s', g) = \sum_{g_i \in g} b_i^2 R_{\text{item}}(s', g_i) \tag{G.16}$$

So that the gradient is:

$$\frac{\partial R}{\partial b_i} = b_i R_{\text{item}}(s', g_i) = \begin{cases} 2R_{\text{item}}(s', g_i) & \text{if } b_i = 1 \\ 0 & \text{if } b_i = 0 \end{cases} \tag{G.17}$$

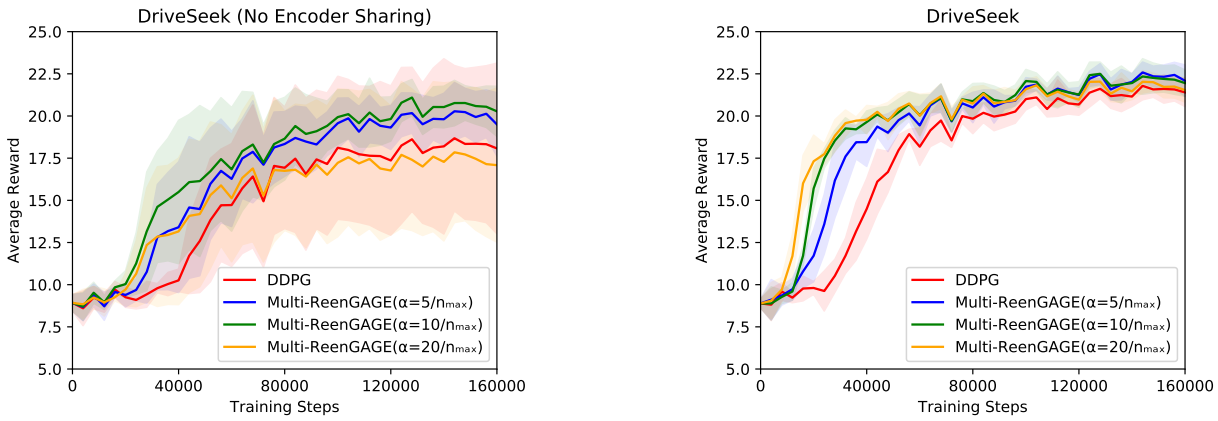
Which again the intended gradient of the Q-function without the dummy inputs, times a constant factor of two. Putting everything together, the final gradient loss term is in the form:

$$\mathcal{L}_{\textbf{Multi-ReenGAGE}} = \mathcal{L}_{\textbf{DDPG-Critic}} + \begin{cases} \alpha \mathcal{L}_{\text{mse}}\Big[2\nabla_b Q_\theta(s, a, g), 2R_{\text{item}}(s', g) + 2\gamma \nabla_b Q_{\theta'}(s', \pi_{\phi'}(s', g), g)\Big] & \text{if } b_i = 1 \\ \alpha \mathcal{L}_{\text{mse}}\Big[0, 0\Big] \ (= 0) & \text{if } b_i = 0 \end{cases} \tag{G.18}$$

Which is the desired loss, up to a constant factor.

G.6.2 Shared Encoder Ablation Study

We performed an ablation study on sharing the encoder between the Q-value function and the policy on DriveSeek when using Multi-ReenGAGE. Results are presented in Figure G.10. We see that sharing the encoder improves the performance of both Multi-ReenGAGE and the DDPG baseline.



(a) (b)

Figure G.10: Ablation study of encoder sharing for Multi-ReenGAGE on DriveSeek. (a) Without Encoder Sharing; (b) With Encoder Sharing.

G.6.3 Training Hyperparameters for Experiments

The hyperparameters used for the Multi-ReenGAGE experiments are presented in Table G.4.

G.6.4 Additional Details about Environments

In this section, we provide additional details about the DriveSeek and NoisySeek environments not included in the main text, in order to more completely describe them.

G.6.4.1 DriveSeek

In the DriveSeek environment, the initial position is always fixed at $(0, 0)$, and the initial velocity vector is always at 0 radians. Episodes last 40 time steps. Goals are sampled by the following procedure: first, a number of goals n is chosen uniformly at random from $\{1, ..., 200\}$; then n goals are chosen uniformly without replacement from the integer coordinates in $[-10, 10]^2$.

Replay Buffer Size	1000000
Frequency of Training	Every 1 environment step
Gradient Descent Steps per Training	1
Initial Steps before Training	1000
Discount γ	0.95
Batch Size	256
Learning Rate	0.001
Polyak Update τ	0.005
Normal action noise for training σ	0.05 for DriveSeek; 0.1 for NoisySeek
Embedding Dimension	20
Extractor Architecture	Fully Connected; 2 hidden layers of width 400; ReLU activations
Head Architecture (both actor and critic)	Fully Connected; 1 hidden layer of width 400; ReLU activation
Evaluation episodes	100
Evaluation Frequency	Every 4000 environment steps

Table G.4: Hyperparameters for Multi-ReenGAGE Experiments

In addition to the goals, the observation s that the agent and policy receives is 6 dimensional: it consists of the current position $s_{\text{pos.}}$, the *rounded* version of $s_{\text{pos.}}$ (this is the "achieved goal": a reward is obtained if this matches one of the input goals), and the sine and cosine of the velocity vector.

G.6.4.2 NoisySeek

In the NoisySeek Environment, the initial position is fixed a $(0, 0)$. Episodes last 40 time steps. Goals are sampled by the following procedure: first, a number of clusters n_c is chosen from a geometric distribution with parameter $p = 0.15$, and a maximum number of goals n' is chosen uniformly at random from $\{1, ..., 200\}$. Then, cluster centers are chosen from a Gaussian distribution with mean 0 and standard deviation 10 in both dimensions. Next, a Dirichlet distribution of order $K = n_c$, with $\alpha_1, ..., \alpha_K = 1$, is used to assign a probability p_j to each cluster. Next, each of the n' goals are assigned to a cluster, with probability p_j of being assigned to cluster j. Then, each goal is determined by adding Gaussian noise with mean 0 and standard deviation 2 in both dimensions to the cluster center assigned to that goal, and then rounding to the nearest integer

coordinates. Finally, goals are de-duplicated.

In addition to the goals, the observation s that the agent and policy receives is 4 dimensional: it consists of the current position s, and the *rounded* version of s (this is the "achieved goal": a reward is obtained if this matches one of the input goals).

G.6.5 NoisySeek results for additional α values

We tested NoisySeek with additional values of α, which we did not include in the main text to avoid cluttered presentation; the trend is generally the same as shown in the main text. Results are shown in Figure G.11.

G.6.6 DriveSeek with CNN Architecture

Because the positions in the DriveSeek environment are bounded, we attempted using a CNN architecture to interpret the state and goals. In particular, because all possible goals appear at unique locations in two-dimensional space, rather than using a DeepSets [149]–style architecture, we can directly surface the goal "gates" b_i as part of the input image: the location in the image corresponding to g_i is blank if b_i is zero (the goal is absent) and colored in if b_i is one (the goal is present). See Figure G.12. Note that, as in the DeepSet implementation, we use b_i^2 in the representation, so that goals which are absent have zero associated attention. We used the standard "Nature CNN" architecture from [176] with otherwise the same training hyperparameters as used in the main-text experiment. The CNN was shared between the actor and the critic networks, and trained only using the critic loss; its output was then fed into separate actor and critic heads, consisting of a single hidden layer of width 400, as in the DeepSets-based architecture. Results

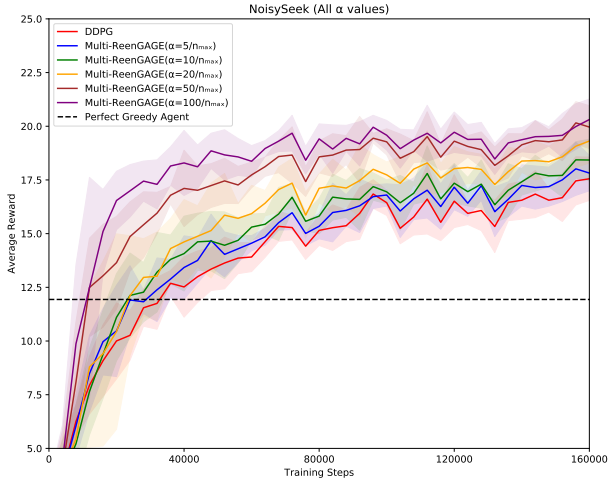


Figure G.11: Results for NoisySeek Environment including additional values of α.

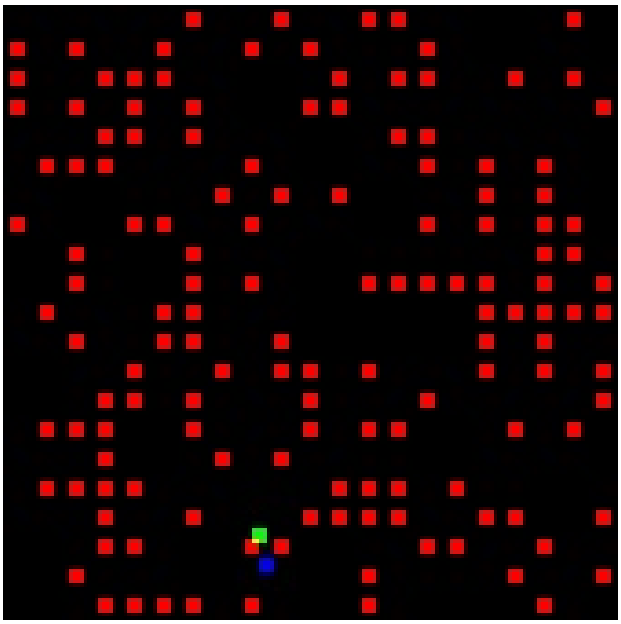


Figure G.12: Example input to CNN architecture for DriveSeek. We use one channel (red) to represent the goals (note that these pixels directly correspond to the differentiable indicator variables b_i^2 for each possible goal), another channel (green) to represent the current position, and a third channel (blue) to represent the next position if the action a is zero: in other words, it indicates $\mathbf{s}_{\text{vel.}}$. Goals are spaced out so that (approximately, up to single-pixel rounding) a goal is achieved if the current position indicator (green) overlaps with the (red) goal. The exact position and velocity vectors are also provided as a separate input, apart from the CNN.

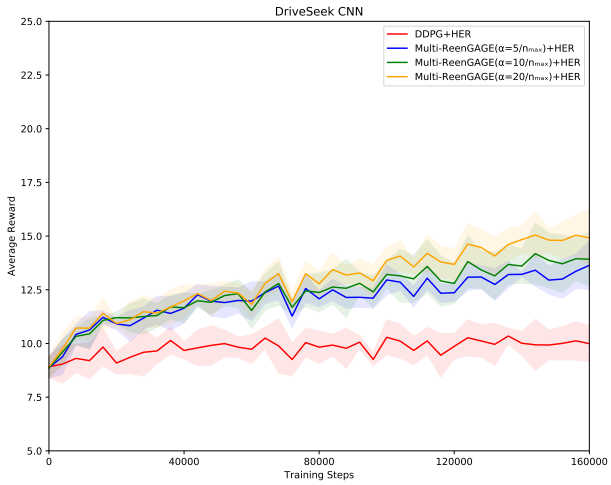


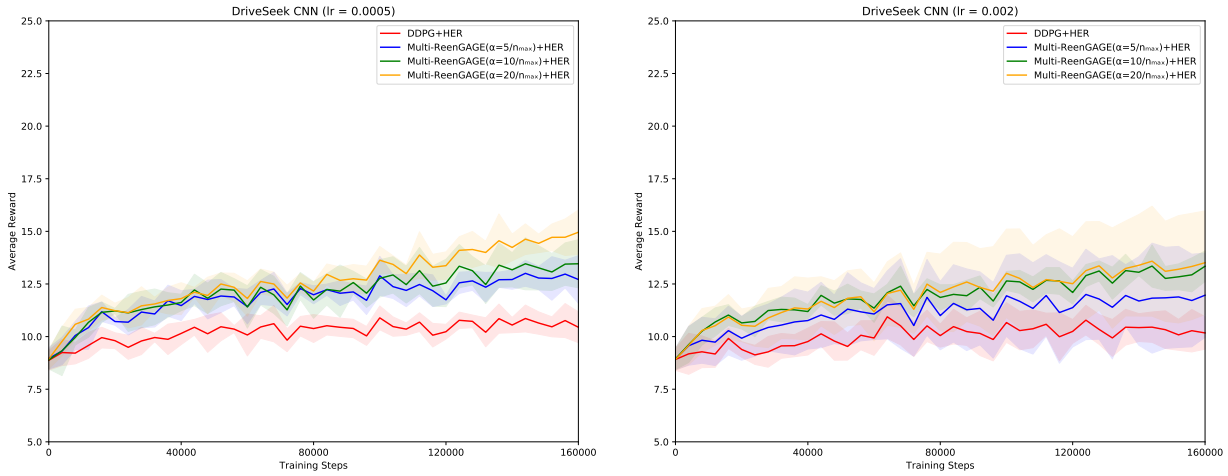
Figure G.13: Results for DriveSeek with CNN architecture.

are shown in Figure G.13.

In general, the performance was worse than using the DeepSets-style architecture; however, the models using Multi-ReenGAGE still outperform the standard DDPG models. One possible explanation for this is that the hyperparameters are poorly-tuned for the CNN architecture. Doubling and halving the learning rate (to 0.002 and 0.0005, respectively) did not seem to affect the results much (Figure G.14), but it is possible that other hyperparameter adjustment may lead to better performance. Another possible explanation for the poor performance is that the pixel-resolution of the images (each pixel has width of 0.125 units) was not sufficient to capture the real-valued dynamics of the environment (although we did also include the real-valued state position and velocity vectors [and rounded state position] as inputs concatenated to the CNN output).

G.7 Runtime Discussion and Empirical Comparisons

As noted in [146], loss terms involving gradients with respect to inputs, such as the ReenGAGE loss, should only scale the computational cost of computing the parameter gradient update



(a) (b)

Figure G.14: DriveSeek with CNN architecture with varied values of the learning rate.

by a constant factor.

For an intuition as to why this is the case, recall the well-known fact that for a differentiable scalar-output function $f(\mathbf{x}, \theta)$ represented by some computational graph which requires n operations to compute, standard backpropagation can compute the gradient $\nabla_{\mathbf{x}} f(\mathbf{x}, \theta)$ (or the gradient $\nabla_{\theta} f(\mathbf{x}, \theta)$) using only cn operations, for a constant c. However, now note that $\nabla_x f(\mathbf{x}, \theta)$ can *itself* be though of as a cn-operation component of a larger computational graph. Thus, if we consider the scalar-valued function $h(\nabla_{\mathbf{x}} f(\mathbf{x}, \theta))$, where h itself takes k additional operations to compute, then we can upper-bound the total number of operations required to compute the gradient $\nabla_{\theta} h(\nabla_{\mathbf{x}} f(\mathbf{x}, \theta))$ by $c(cn + k) = c^2 n + ck$; in other words, a constant factor c^2 of n, plus some overhead. In the particular case of the ReenGAGE loss term (in the sparse case, for simplicity), x corresponds to the goal g and f corresponds to $Q_{\theta}(s, a, g)$, while h corresponds to $\|\nabla_g f(g, \theta) - \nabla_g \gamma Q_{\theta'}(s', \pi_{\phi'}(s', g), g)\|_2^2$. If the Q function requires n operations to evaluate and the policy π requires m operations, then the overall computational cost can therefore be upper-bounded by $c(cn + c(n + m) + k) = 2c^2 n + c^2 m + ck$, where k is the (trivial) amount of

computation required to compute the norm. This is therefore a constant factor of the time needed to compute the value of the Q function and its target (with some trivial overhead due to the norm computation).

[146] provides explicit algorithms for computations of gradients of forms similar to the form $\nabla_\theta h(\nabla_{\mathbf{x}} f(\mathbf{x}, \theta))$ discussed above and confirms the constant-factor increase in computational complexity; in particular, $h(\cdot)$ corresponds to $p(\cdot)$ in Equation 10 of [146]. (Note that [146]'s analysis is somewhat more general, allowing for a vector-valued f, with $p(\cdot)$ a function of a Jacobian-vector product of f rather than simply the gradient).

To provide empirical support for this, we provide runtime comparisons for training with and without the ReenGAGE loss term, for the experiments in the main text. Note that we are comparing total runtimes, so these times include the environment simulation; however this should be relatively minor for all experiments (because the environments themselves are relatively simple) except possibly the robotics experiments.

For ContinuousSeek and Multi-ReenGAGE experiments, all tests were run on a single GPU. We used a pool of shared computational resource, so the GPU models may have varied between runs; GPUs possibly used were NVIDIA RTX A4000, RTX A5000, and RTX A6000 models. (This adds some uncertainty to our runtime comparisons.) Robotics experiments were run on 20 CPUs each, as described by [148].

Runtime comparison results are given in Table G.5. For ContinuousSeek, we consider only experiments with $d = 20$, batch size = 256; for others, we include all experiments shown in the main text. For runtimes with ReenGAGE, we average over all values of α included in the main text. In general, runtime increases ranged from 34%-60%.

Environment	Runtime without ReenGAGE (s)	Runtime with ReenGAGE (s)	Mean Percent Increase
ContinuousSeek	2549 ± 203	3591 ± 248	40.9%
HandReach	2533 ± 2	3395 ± 208	34.0%
DriveSeek	8931 ±2037	12467 ± 1354	39.6%
NoisySeek	7385 ± 503	11814 ± 1519	60.0%

Table G.5: Effect of ReenGAGE on runtimes. Error values shown are standard deviations over all runs.

Bibliography

[1] Guang-He Lee, Yang Yuan, Shiyu Chang, and Tommi Jaakkola. Tight certificates of adversarial robustness for randomly smoothed classifiers. In *Advances in Neural Information Processing Systems*, pages 4910–4921, 2019.

[2] Eric Wong, Frank Schmidt, and Zico Kolter. Wasserstein adversarial examples via projected Sinkhorn iterations. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6808–6817, Long Beach, California, USA, 09–15 Jun 2019. PMLR. URL `http://proceedings.mlr.press/v97/wong19a.html`.

[3] L Schott, J Rauber, M Bethge, and W Brendel. Towards the first adversarially robust neural network model on mnist. In *Seventh International Conference on Learning Representations (ICLR 2019)*, pages 1–16, 2019.

[4] PingYeh Chiang, Renkun Ni, Ahmed Abdelkader, Chen Zhu, Christoph Studor, and Tom Goldstein. Certified defenses for adversarial patches. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=HyeaSkrYPH`.

[5] Greg Yang, Tony Duan, J. Edward Hu, Hadi Salman, Ilya Razenshteyn, and Jerry Li. Randomized smoothing of all shapes and sizes. In *Proceedings of the 37th International Conference on Machine Learning*, pages 10693–10705, 2020.

[6] Elan Rosenfeld, Ezra Winston, Pradeep Ravikumar, and Zico Kolter. Certified robustness to label-flipping attacks via randomized smoothing. In *International Conference on Machine Learning*, pages 8230–8241. PMLR, 2020.

[7] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015*, 2015. URL `http://arxiv.org/abs/1412.6572`.

[8] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. IEEE, 2017.

[9] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[10] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[11] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. Robust physical-world attacks on deep learning visual classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1625–1634, 2018.

[12] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*, 2017.

[13] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 582–597. IEEE, 2016.

[14] Mitali Bafna, Jack Murtagh, and Nikhil Vyas. Thwarting adversarial examples: An l_0-robust sparse fourier transform. In *Advances in Neural Information Processing Systems*, pages 10075–10085, 2018.

[15] Dongyu Meng and Hao Chen. Magnet: a two-pronged defense against adversarial examples. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 135–147. ACM, 2017.

[16] Nicolas Papernot and Patrick McDaniel. Extending defensive distillation. *arXiv preprint arXiv:1705.05264*, 2017.

[17] Pouya Samangouei, Maya Kabkab, and Rama Chellappa. Defense-GAN: Protecting classifiers against adversarial attacks using generative models. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=BkJ3ibb0-`.

[18] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International Conference on Machine Learning*, pages 274–283, 2018.

[19] Nicholas Carlini and David Wagner. Defensive distillation is not robust to adversarial examples. *arXiv preprint arXiv:1607.04311*, 2016.

[20] Florian Tramer, Nicholas Carlini, Wieland Brendel, and Aleksander Madry. On adaptive attacks to adversarial example defenses. *Advances in neural information processing systems*, 33:1633–1645, 2020.

[21] Anish Athalye and Nicholas Carlini. On the robustness of the cvpr 2018 white-box adversarial example defenses, 2018.

[22] Eric Wong and Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning*, pages 5283–5292, 2018.

[23] Sven Gowal, Krishnamurthy Dvijotham, Robert Stanforth, Rudy Bunel, Chongli Qin, Jonathan Uesato, Timothy Mann, and Pushmeet Kohli. On the effectiveness of interval bound propagation for training verifiably robust models. *arXiv preprint arXiv:1810.12715*, 2018.

[24] Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Semidefinite relaxations for certifying robustness to adversarial examples. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 10900–10910, Red Hook, NY, USA, 2018. Curran Associates Inc.

[25] Vincent Tjeng, Kai Y. Xiao, and Russ Tedrake. Evaluating robustness of neural networks with mixed integer programming. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HyGIdiRqtm.

[26] Huan Zhang, Tsui-Wei Weng, Pin-Yu Chen, Cho-Jui Hsieh, and Luca Daniel. Efficient neural network robustness certification with general activation functions. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31, pages 4939–4948. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/d04863f100d59b3eb688a11f95b0ae60-Paper.pdf.

[27] Sahil Singla and Soheil Feizi. Second-order provable defenses against adversarial attacks. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 8981–8991. PMLR, 13–18 Jul 2020. URL http://proceedings.mlr.press/v119/singla20a.html.

[28] Marc Fischer, Maximilian Baader, and Martin Vechev. Certified defense to image transformations via randomized smoothing. *Advances in Neural Information Processing Systems Foundation (NeurIPS)*, 2020.

[29] Alexander Levine and Soheil Feizi. (de)randomized smoothing for certifiable defense against patch attacks. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/47ce0875420b2dbacfc5535f94e68433-Abstract.html.

[30] Chong Xiang, A. Bhagoji, V. Sehwag, and P. Mittal. Patchguard: Provable defense against adversarial patches using masks on small receptive fields. *ArXiv*, abs/2005.10884, 2020.

[31] Z. Zhang, B. Yuan, Michael McCoyd, and D. Wagner. Clipped bagnet: Defending against sticker attacks with clipped bag-of-features. *2020 IEEE Security and Privacy Workshops (SPW)*, pages 55–61, 2020.

[32] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified adversarial robustness via randomized smoothing. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1310–1320, Long Beach, California, USA, 09–15 Jun 2019. PMLR. URL http://proceedings.mlr.press/v97/cohen19c.html.

[33] Hadi Salman, Jerry Li, Ilya Razenshteyn, Pengchuan Zhang, Huan Zhang, Sebastien Bubeck, and Greg Yang. Provably robust deep learning via adversarially trained smoothed classifiers. In *Advances in Neural Information Processing Systems*, pages 11292–11303, 2019.

[34] Mathias Lecuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. Certified robustness to adversarial examples with differential privacy. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 656–672. IEEE, 2019.

[35] Bai Li, Changyou Chen, Wenlin Wang, and Lawrence Carin. Certified adversarial robustness with additive noise. In *Advances in Neural Information Processing Systems*, pages 9464–9474, 2019.

[36] Jiaye Teng, Guang-He Lee, and Yang Yuan. ℓ_1 adversarial robustness certificates: a randomized smoothing approach, 2020. URL https://openreview.net/forum?id=H1lQIgrFDS.

[37] Linyi Li, Xiangyu Qi, Tao Xie, and Bo Li. Sok: Certified robustness for deep neural networks. *arXiv preprint arXiv:2009.04131*, 2020.

[38] Alexander Levine, Sahil Singla, and Soheil Feizi. Certifiably robust interpretation in deep learning. *arXiv preprint arXiv:1905.12105*, 2019.

[39] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations*, 2018.

[40] Aounon Kumar, Alexander Levine, Tom Goldstein, and Soheil Feizi. Provable robustness against wasserstein distribution shifts via input randomization. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=HJFVrpCaGE.

[41] Rohan Taori, Achal Dave, Vaishaal Shankar, Nicholas Carlini, Benjamin Recht, and Ludwig Schmidt. Measuring robustness to natural distribution shifts in image classification. *Advances in Neural Information Processing Systems*, 33:18583–18599, 2020.

[42] Robert Kirk, Amy Zhang, Edward Grefenstette, and Tim Rocktäschel. A survey of zero-shot generalisation in deep reinforcement learning. *Journal of Artificial Intelligence Research*, 76:201–264, 2023.

[43] Karl Cobbe, Christopher Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning, 2019.

[44] Shivam Goel, Gyan Tatiya, Matthias Scheutz, and Jivko Sinapov. Novelgridworlds: A benchmark environment for detecting and adapting to novelties in open worlds. In *AAMAS Adaptive Learning Agents (ALA) Workshop*, 2021.

[45] Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1312–1320, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/schaul15.html.

[46] Minghuan Liu, Menghui Zhu, and Weinan Zhang. Goal-conditioned reinforcement learning: Problems and solutions. In Lud De Raedt, editor, *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 5502–5511. International Joint Conferences on Artificial Intelligence Organization, 7 2022. doi: 10.24963/ijcai.2022/770. URL https://doi.org/10.24963/ijcai.2022/770. Survey Track.

[47] Carlos Florensa, David Held, Xinyang Geng, and Pieter Abbeel. Automatic goal generation for reinforcement learning agents. In *International conference on machine learning*, pages 1515–1528. PMLR, 2018.

[48] Sainbayar Sukhbaatar, Zeming Lin, Ilya Kostrikov, Gabriel Synnaeve, Arthur Szlam, and Rob Fergus. Intrinsic motivation and automatic curricula via asymmetric self-play. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=SkT5Yg-RZ.

[49] Yunzhi Zhang, Pieter Abbeel, and Lerrel Pinto. Automatic curriculum learning through value disagreement. *Advances in Neural Information Processing Systems*, 33:7648–7659, 2020.

[50] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

[51] Greg Yang, Tony Duan, Edward Hu, Hadi Salman, Ilya Razenshteyn, and Jerry Li. Randomized smoothing of all shapes and sizes. *arXiv preprint arXiv:2002.08118*, 2020.

[52] Tom B. Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. Adversarial patch, 2017.

[53] Danny Karmon, Daniel Zoran, and Yoav Goldberg. Lavan: Localized and visible adversarial noise. *arXiv preprint arXiv:1801.02608*, 2018.

[54] Jamie Hayes. On visible adversarial perturbations & digital watermarking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1597–1604, 2018.

[55] Muzammal Naseer, Salman Khan, and Fatih Murat Porikli. Local gradients smoothing: Defense against localized adversarial attacks. *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1300–1307, 2018.

[56] Faisal Zaman and Hideo Hirose. Effect of subsampling rate on subbagging and related ensembles of stable classifiers. In Santanu Chaudhury, Sushmita Mitra, C. A. Murthy, P. S. Sastry, and Sankar K. Pal, editors, *Pattern Recognition and Machine Intelligence*, pages 44–49, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. ISBN 978-3-642-11164-8.

[57] Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Sks9_ajex.

[58] Alexander Levine and Soheil Feizi. Robustness certificates for sparse adversarial attacks by randomized ablation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 4585–4593, 2020.

[59] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9185–9193. IEEE, 2018.

[60] Alexey Kurakin, Ian J Goodfellow, and Samy Bengio. Adversarial examples in the physical world. In *Artificial Intelligence Safety and Security*, pages 99–112. Chapman and Hall/CRC, 2018.

[61] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 372–387. IEEE, 2016.

[62] Hossein Hosseini, Sreeram Kannan, and Radha Poovendran. Dropping pixels for adversarial robustness. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019.

[63] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified adversarial robustness via randomized smoothing. In *International Conference on Machine Learning*, pages 1310–1320, 2019.

[64] M. Lecuyer, V. Atlidakis, R. Geambasu, D. Hsu, and S. Jana. Certified robustness to adversarial examples with differential privacy. In *2019 2019 IEEE Symposium on Security and Privacy (SP)*, pages 726–742, Los Alamitos, CA, USA, may 2019. IEEE Computer Society. doi: 10.1109/SP.2019.00044. URL https://doi.ieeecomputersociety.org/10.1109/SP.2019.00044.

[65] Jinyuan Jia, Binghui Wang, Xiaoyu Cao, Hongbin Liu, and Neil Zhenqiang Gong. Almost tight l0-norm certified robustness of top-k predictions against adversarial perturbations. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=gJLEXy3ySpu.

[66] Jonas Rauber, Wieland Brendel, and Matthias Bethge. Foolbox: A python toolbox to benchmark the robustness of machine learning models. *arXiv preprint arXiv:1707.04131*, 2017.

[67] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019. URL https://openreview.net/forum?id=SyxAb30cY7.

[68] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[69] Kuang Liu. 95.16% on cifar10 with pytorch. https://github.com/kuangliu/pytorch-cifar, 2019.

[70] Weilin Xu, David Evans, and Yanjun Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. *arXiv preprint arXiv:1704.01155*, 2017.

[71] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017. URL https://openreview.net/forum?id=Sy8gdB9xx.

[72] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*, 2017.

[73] Alexander J Levine and Soheil Feizi. Improved, deterministic smoothing for l_1 certified robustness. In *International Conference on Machine Learning*, pages 6254–6264. PMLR, 2021.

[74] Alexander Levine and Soheil Feizi. Deep partition aggregation: Provable defenses against general poisoning attacks. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=YUGG2tFuPM.

[75] Maurice Weber, Xiaojun Xu, Bojan Karlas, Ce Zhang, and Bo Li. Rab: Provable robustness against backdoor attacks. *arXiv preprint arXiv:2003.08904*, 2020.

[76] Ching-Chia Kao, Jhe-Bang Ko, and Chun-Shien Lu. Deterministic certification to adversarial attacks via bernstein polynomial approximation, 2020.

[77] Qiyang Li, S. Haque, C. Anil, J. Lucas, R. Grosse, and J. Jacobsen. Preventing gradient attenuation in lipschitz constrained convolutional networks. In *NeurIPS*, 2019.

[78] Cem Anil, James Lucas, and Roger Grosse. Sorting out Lipschitz function approximation. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 291–301. PMLR, 09–15 Jun 2019. URL http://proceedings.mlr.press/v97/anil19a.html.

[79] Hadi Salman, Mingjie Sun, Greg Yang, Ashish Kapoor, and J Zico Kolter. Denoised smoothing: A provable defense for pretrained classifiers. *Advances in Neural Information Processing Systems*, 33, 2020.

[80] Alexander J. Levine and Soheil Feizi. Provable adversarial robustness for fractional lp threat models. In Gustau Camps-Valls, Francisco J. R. Ruiz, and Isabel Valera, editors, *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pages 9908–9942. PMLR, 28–30 Mar 2022. URL https://proceedings.mlr.press/v151/levine22a.html.

[81] Charu C. Aggarwal, Alexander Hinneburg, and Daniel A. Keim. On the surprising behavior of distance metrics in high dimensional space. In Jan Van den Bussche and Victor Vianu, editors, *Database Theory — ICDT 2001*, pages 420–434, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg. ISBN 978-3-540-44503-6.

[82] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the Twentieth Annual Symposium on Computational Geometry*, SCG '04, page 253–262, New York, NY, USA, 2004. Association for Computing Machinery. ISBN 1581138857. doi: 10.1145/997817.997857. URL https://doi.org/10.1145/997817.997857.

[83] Dimitris G. Chachlakis and Panos P. Markopoulos. Novel algorithms for lp-quasi-norm principal-component analysis. In *2020 28th European Signal Processing Conference (EUSIPCO)*, pages 1045–1049, 2021. doi: 10.23919/Eusipco47968.2020.9287335.

[84] Peter Howarth and Stefan Rüger. Fractional distance measures for content-based image retrieval. In David E. Losada and Juan M. Fernández-Luna, editors, *Advances in Information Retrieval*, pages 447–456, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. ISBN 978-3-540-31865-1.

[85] Hao Wang, Xiangyu Yang, and Xin Deng. A hybrid first-order method for nonconvex ℓ_p-ball constrained optimization, 2021.

[86] Alexander Levine and Soheil Feizi. Wasserstein smoothing: Certified robustness against wasserstein adversarial attacks. In Silvia Chiappa and Roberto Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 3938–3947. PMLR,

26–28 Aug 2020. URL http://proceedings.mlr.press/v108/levine20a.html.

[87] Cassidy Laidlaw and Soheil Feizi. Functional adversarial attacks. *arXiv preprint arXiv:1906.00001*, 2019.

[88] Logan Engstrom, Brandon Tran, Dimitris Tsipras, Ludwig Schmidt, and Aleksander Madry. Exploring the landscape of spatial robustness. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1802–1811, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

[89] Felix Assion, Peter Schlicht, Florens Greßner, Wiebke Gunther, Fabian Huger, Nico Schmidt, and Umair Rasheed. The attack generator: A systematic approach towards constructing adversarial attacks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019.

[90] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pages 506–519. ACM, 2017.

[91] Bai Li, Changyou Chen, Wenlin Wang, and Lawrence Carin. Second-order adversarial attack and certifiable robustness. *arXiv preprint arXiv:1809.03113*, 2018.

[92] Tobias Wegel, Felix Assion, David Mickisch, and Florens Greßner. A framework for verification of wasserstein adversarial robustness. *Second Graduate Student Conference: Geometry and Topology meet Data Analysis and Machine Learning (GTDAML2021)*, 2021.

[93] Haibin Ling and Kazunori Okada. An efficient earth mover's distance algorithm for robust histogram comparison. *IEEE transactions on pattern analysis and machine intelligence*, 29(5):840–853, 2007.

[94] Gabriel Peyré, Marco Cuturi, et al. Computational optimal transport. *Foundations and Trends® in Machine Learning*, 11(5-6):355–607, 2019.

[95] Hadi Salman, Saachi Jain, Eric Wong, and Aleksander Madry. Certified patch robustness via smoothed vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15137–15147, 2022.

[96] Chong Xiang and Prateek Mittal. Patchguard++: Efficient provable attack detection against adversarial patches. *arXiv preprint arXiv:2104.12609*, 2021.

[97] Yuheng Huang, Lei Ma, and Yuanchun Li. Patchcensor: Patch robustness certification for transformers via exhaustive testing. *ACM Trans. Softw. Eng. Methodol.*, apr 2023. ISSN 1049-331X. doi: 10.1145/3591870. URL https://doi.org/10.1145/3591870. Just Accepted.

[98] Zhaoyu Chen, Bo Li, Jianghe Xu, Shuang Wu, Shouhong Ding, and Wenqiang Zhang. Towards practical certifiable patch defense with vision transformer. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15127–15137, 2022.

[99] Huijie Feng, Chunpeng Wu, Guoyang Chen, Weifeng Zhang, and Yang Ning. Regularized training and tight certification for randomized smoothed classifier with provable robustness, 2020.

[100] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*, 2017.

[101] Han Xiao, Huang Xiao, and Claudia Eckert. Adversarial label flips attack on support vector machines. In *Proceedings of the 20th European Conference on Artificial Intelligence*, pages 870–875, 2012.

[102] Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, ICML'12, page 1467–1474, Madison, WI, USA, 2012. Omnipress. ISBN 9781450312851.

[103] Ali Shafahi, W. Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein. Poison frogs! targeted clean-label poisoning attacks on neural networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 6103–6113. Curran Associates, Inc., 2018. URL `http://papers.nips.cc/paper/7849-poison-frogs-targeted-clean-label-poisoning-attacks-on-neural-networks.pdf`.

[104] Jacob Steinhardt, Pang Wei W Koh, and Percy S Liang. Certified defenses for data poisoning attacks. In *Advances in neural information processing systems*, pages 3517–3529, 2017.

[105] Vikas Verma, Alex Lamb, Juho Kannala, Yoshua Bengio, and David Lopez-Paz. Interpolation consistency training for semi-supervised learning. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 3635–3641. AAAI Press, 2019.

[106] Yucen Luo, Jun Zhu, Mengxi Li, Yong Ren, and Bo Zhang. Smooth neighbors on teacher graphs for semi-supervised learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[107] Samuli Laine and Timo Aila. Temporal ensembling for semi-supervised learning. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017. URL `https://openreview.net/forum?id=BJ6oOfqge`.

[108] Durk P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In Z. Ghahramani, M. Welling,

C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3581–3589. Curran Associates, Inc., 2014. URL http://papers.nips.cc/paper/5352-semi-supervised-learning-with-deep-generative-models.pdf.

[109] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=S1v4N2l0-.

[110] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks*, 32:323 – 332, 2012. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2012.02.016. URL http://www.sciencedirect.com/science/article/pii/S0893608012000457. Selected Papers from IJCNN 2011.

[111] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[112] Andreas Buja and Werner Stuetzle. Observations on bagging. *Statistica Sinica*, pages 323–351, 2006.

[113] Peter Lukas Bühlmann. Bagging, subagging and bragging for improving some prediction algorithms. In *Research report/Seminar für Statistik, Eidgenössische Technische Hochschule (ETH)*, volume 113. Seminar für Statistik, Eidgenössische Technische Hochschule (ETH), Zürich, 2003.

[114] Battista Biggio, Igino Corona, Giorgio Fumera, Giorgio Giacinto, and Fabio Roli. Bagging classifiers for fighting poisoning attacks in adversarial classification tasks. In *International workshop on multiple classifier systems*, pages 350–359. Springer, 2011.

[115] Charles Smutz and Angelos Stavrou. When a tree falls: Using diversity in ensemble classifiers to identify evasion in malware detectors. In *23rd Annual Network and Distributed System Security Symposium, NDSS 2016, San Diego, California, USA, February 21-24, 2016*. The Internet Society, 2016. URL http://wp.internetsociety.org/ndss/wp-content/uploads/sites/25/2017/09/when-tree-falls-using-diversity-ensemble-classifiers-identify-evasion-malware-detectors.pdf.

[116] Jinyuan Jia, Xiaoyu Cao, and Neil Zhenqiang Gong. Intrinsic certified robustness of bagging against data poisoning attacks, 2020.

[117] Ruoxin Chen, Zenan Li, Jie Li, Junchi Yan, and Chentao Wu. On collective robustness of bagging against data poisoning. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 3299–3319. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/chen22k.html.

[118] Jinyuan Jia, Yupei Liu, Xiaoyu Cao, and Neil Zhenqiang Gong. Certified robustness of nearest neighbors against data poisoning and backdoor attacks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 9575–9583, 2022.

[119] Zayd Hammoudeh and Daniel Lowd. Reducing certified regression to certified classification for general poisoning attacks. In *2023 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, pages 484–523. IEEE, 2023.

[120] Keivan Rezaei, Kiarash Banihashem, Atoosa Chegini, and Soheil Feizi. Run-off election: Improved provable defense against data poisoning attacks. *arXiv preprint arXiv:2302.02300*, 2023.

[121] Wenxiao Wang and Soheil Feizi. On practical aspects of aggregation defenses against data poisoning attacks. *arXiv preprint arXiv:2306.16415*, 2023.

[122] Wenxiao Wang, Alexander J Levine, and Soheil Feizi. Improved certified defenses against data poisoning with (Deterministic) finite aggregation. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 22769–22783. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/wang22m.html.

[123] Aounon Kumar, Alexander Levine, Tom Goldstein, and Soheil Feizi. Curse of dimensionality on randomized smoothing for certifiable robustness. In *International Conference on Machine Learning*, pages 5458–5467. PMLR, 2020.

[124] Avrim Blum, Travis Dick, Naren Manoj, and Hongyang Zhang. Random smoothing might be unable to certify ℓ_∞ robustness for high-dimensional images. *arXiv preprint arXiv:2002.03517*, 2020.

[125] Robert H. Sloan. Four types of noise in data for pac learning. *Information Processing Letters*, 54(3):157 – 162, 1995. ISSN 0020-0190. doi: https://doi.org/10.1016/0020-0190(95)00016-6. URL http://www.sciencedirect.com/science/article/pii/0020019095000166.

[126] Nader H. Bshouty, Nadav Eiron, and Eyal Kushilevitz. Pac learning with nasty noise. *Theoretical Computer Science*, 288(2):255–275, 2002. ISSN 0304-3975. doi: https://doi.org/10.1016/S0304-3975(01)00403-0. URL https://www.sciencedirect.com/science/article/pii/S0304397501004030. Algorithmic Learning Theory.

[127] I. Diakonikolas, G. Kamath, D. M. Kane, J. Li, A. Moitra, and A. Stewart. Robust estimators in high dimensions without the computational intractability. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 655–664, Los Alamitos, CA, USA, oct 2016. IEEE Computer Society. doi: 10.1109/FOCS.2016.85. URL https://doi.ieeecomputersociety.org/10.1109/FOCS.2016.85.

[128] K. A. Lai, A. B. Rao, and S. Vempala. Agnostic estimation of mean and covariance. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 665–674, 2016. doi: 10.1109/FOCS.2016.76.

[129] Saeed Mahloujifar, Dimitrios I. Diochnos, and Mohammad Mahmoody. The curse of concentration in robust learning: Evasion and poisoning attacks from concentration of measure. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):4536–4543, Jul. 2019. doi: 10.1609/aaai.v33i01.33014536. URL https://ojs.aaai.org/index.php/AAAI/article/view/4373.

[130] Ilias Diakonikolas, Gautam Kamath, D. Kane, Jerry Li, J. Steinhardt, and Alistair Stewart. Sever: A robust meta-algorithm for stochastic optimization. In *ICML*, 2019.

[131] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[132] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. A simple framework for contrastive learning of visual representations. *ArXiv*, abs/2002.05709, 2020.

[133] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

[134] Alexander Levine and Soheil Feizi. Goal-conditioned q-learning as knowledge distillation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 8500–8509, 2023.

[135] Ashvin V Nair, Vitchyr Pong, Murtaza Dalal, Shikhar Bahl, Steven Lin, and Sergey Levine. Visual reinforcement learning with imagined goals. *Advances in neural information processing systems*, 31, 2018.

[136] Rui Yang, Meng Fang, Lei Han, Yali Du, Feng Luo, and Xiu Li. Mher: Model-based hindsight experience replay. In *Deep RL Workshop NeurIPS 2021*, 2021.

[137] Meng Fang, Tianyi Zhou, Yali Du, Lei Han, and Zhengyou Zhang. Curriculum-guided hindsight experience replay. *Advances in neural information processing systems*, 32, 2019.

[138] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *ArXiv*, abs/1503.02531, 2015.

[139] Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6):1789–1819, 2021.

[140] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. URL http://arxiv.org/abs/1509.02971.

[141] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

[142] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR, 2018.

[143] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3):279–292, 1992.

[144] Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International conference on machine learning*, pages 2052–2062. PMLR, 2019.

[145] Yannick Schroecker and Charles Lee Isbell. Universal value density estimation for imitation learning and goal-conditioned reinforcement learning, 2021. URL https://openreview.net/forum?id=S2UB9PkrEjF.

[146] Christian Etmann. A closer look at double backpropagation. *ArXiv*, abs/1906.06637, 2019.

[147] Yen-Chang Hsu, James Smith, Yilin Shen, Zsolt Kira, and Hongxia Jin. A closer look at knowledge distillation with features, logits, and gradients. *arXiv preprint arXiv:2203.10163*, 2022.

[148] Matthias Plappert, Marcin Andrychowicz, Alex Ray, Bob McGrew, Bowen Baker, Glenn Powell, Jonas Schneider, Josh Tobin, Maciek Chociej, Peter Welinder, Vikash Kumar, and Wojciech Zaremba. Multi-goal reinforcement learning: Challenging robotics environments and request for research, 2018. URL https://arxiv.org/abs/1802.09464.

[149] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/f22e4747da1aa27e363d86d40ff442fe-Paper.pdf.

[150] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the*

36th International Conference on Machine Learning, volume 97 of *Proceedings of Machine Learning Research*, pages 3744–3753. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/lee19d.html.

[151] Benjamin Eysenbach, Ruslan Salakhutdinov, and Sergey Levine. C-learning: Learning to achieve goals via recursive classification. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=tc5qisoB-C.

[152] Benjamin Eysenbach, Tianjun Zhang, Ruslan Salakhutdinov, and Sergey Levine. Contrastive learning as goal-conditioned reinforcement learning. *arXiv preprint arXiv:2206.07568*, 2022.

[153] Michael Janner, Yilun Du, Joshua B Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. *arXiv preprint arXiv:2205.09991*, 2022.

[154] Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=wgeK563QgSw.

[155] Andrei A Rusu, Sergio Gomez Colmenarejo, Çaglar Gülçehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy distillation. In *ICLR (Poster)*, 2016.

[156] Anthony Manchin, Ehsan Abbasnejad, and Anton van den Hengel. Reinforcement learning with attention that works: A self-supervised approach. In *International conference on neural information processing*, pages 223–230. Springer, 2019.

[157] Jinyoung Choi, Beom-Jin Lee, and Byoung-Tak Zhang. Multi-focus attention network for efficient deep reinforcement learning. In *Workshops at the Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[158] Haiping Wu, Khimya Khetarpal, and Doina Precup. Self-supervised attention-aware reinforcement learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(12): 10311–10319, May 2021. URL https://ojs.aaai.org/index.php/AAAI/article/view/17235.

[159] David Bertoin, Adil Zouitine, Mehdi Zouitine, and Emmanuel Rachelson. Look where you look! saliency-guided q-networks for generalization in visual reinforcement learning. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=-_I3i2orAV.

[160] Junhyuk Oh, Satinder Singh, Honglak Lee, and Pushmeet Kohli. Zero-shot task generalization with multi-task deep reinforcement learning. In *International Conference on Machine Learning*, pages 2661–2670. PMLR, 2017.

[161] Sungryull Sohn, Junhyuk Oh, and Honglak Lee. Hierarchical reinforcement learning for zero-shot generalization with subtask dependencies. *Advances in neural information processing systems*, 31, 2018.

[162] Pashootan Vaezipoor, Andrew C Li, Rodrigo A Toro Icarte, and Sheila A Mcilraith. Ltl2action: Generalizing ltl instructions for multi-task rl. In *International Conference on Machine Learning*, pages 10497–10508. PMLR, 2021.

[163] Ahmed Touati and Yann Ollivier. Learning one representation to optimize all rewards. *Advances in Neural Information Processing Systems*, 34:13–23, 2021.

[164] Ahmed Touati, Jérémy Rapin, and Yann Ollivier. Does zero-shot reinforcement learning exist? In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=MYEap_OcQI.

[165] Huan Zhang, Hongge Chen, Chaowei Xiao, Bo Li, Mingyan Liu, Duane Boning, and Cho-Jui Hsieh. Robust deep reinforcement learning against adversarial perturbations on state observations. *Advances in Neural Information Processing Systems*, 33:21024–21037, 2020.

[166] Björn Lütjens, Michael Everett, and Jonathan P How. Certified adversarial robustness for deep reinforcement learning. In *conference on Robot Learning*, pages 1328–1337. PMLR, 2020.

[167] Michael Everett, Björn Lütjens, and Jonathan P How. Certifiable robustness to adversarial state uncertainty in deep reinforcement learning. *IEEE Transactions on Neural Networks and Learning Systems*, 33(9):4184–4198, 2021.

[168] Aounon Kumar, Alexander Levine, and Soheil Feizi. Policy smoothing for provably robust reinforcement learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=mwdfai8NBrJ.

[169] Stanislas Polu and Ilya Sutskever. Generative language modeling for automated theorem proving. *arXiv preprint arXiv:2009.03393*, 2020.

[170] Stanislas Polu, Jesse Michael Han, Kunhao Zheng, Mantas Baksys, Igor Babuschkin, and Ilya Sutskever. Formal mathematics statement curriculum learning. In *The Eleventh International Conference on Learning Representations*, 2022.

[171] Albert Qiaochu Jiang, Sean Welleck, Jin Peng Zhou, Timothee Lacroix, Jiacheng Liu, Wenda Li, Mateja Jamnik, Guillaume Lample, and Yuhuai Wu. Draft, sketch, and prove: Guiding formal theorem provers with informal proofs. In *The Eleventh International Conference on Learning Representations*, 2022.

[172] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. In Aarti Singh and Jerry Zhu, editors, *Proceedings of Machine Learning Research*, volume 54, pages 1273–1282, Fort Lauderdale, FL, USA, 20–22 Apr 2017. PMLR. URL http://proceedings.mlr.press/v54/mcmahan17a.html.

[173] Shun-ichi Amari, Naotake Fujita, and Shigeru Shinomoto. Four types of learning curves. *Neural Computation*, 4(4):605–618, 1992.

[174] Two Six Labs. Armory. https://github.com/twosixlabs/armory, 2019.

[175] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. *arXiv preprint arXiv:2004.11362*, 2020.

[176] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[177] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021. URL `http://jmlr.org/papers/v22/20-1364.html`.