ABSTRACT

Title of dissertation: TOWARDS ROBUST AND DOMAIN INVARIANT FEATURE REPRESENTATIONS IN DEEP LEARNING

Swaminathan Sankaranarayanan
Doctor of Philosophy, 2018

Dissertation directed by: Professor Rama Chellappa
Department of Electrical and Computer Engineering

A fundamental problem in perception-based systems is to define and learn representations of the scene that are more robust and adaptive to several nuisance factors. Over the recent past, for a variety of tasks involving images, learned representations have been empirically shown to outperform handcrafted ones. However, their inability to generalize across varying data distributions poses the following question: Do representations learned using deep networks just fit a given data distribution or do they sufficiently model the underlying structure of the problem ? This question could be understood using a simple example: If a learning algorithm is shown a number of images of a simple handwritten digit, then the representation learned should be generic enough to identify the same digit in a different form. With regards to deep networks, although the learned representation has been shown to be robust to various forms of synthetic distortions such as random noise, they fail in the presence of more implicit forms of naturally occurring distortions. In this dissertation, we propose approaches to mitigate the effect of such distortions and in the

process, study some vulnerabilities of deep networks to small imperceptible changes that occur in the given input. The research problems that comprise this dissertation lie in the cross section of two open topics: (1) Studying and developing methods that enable neural networks learn robust representations (2) Improving generalization of neural nets across domains. The first part of the dissertation approaches the problem of robustness from two broad viewpoints: Robustness to external nuisance factors that occur in the data and robustness (or a lack thereof) to perturbations of the learned feature space. In the second part, we focus on learning representations that are invariant to external covariate shift, which is more commonly termed as domain shift.

Towards learning representations robust to external nuisance factors, we propose an approach that couples a deep convolutional neural network with a low-dimensional discriminative embedding learned using triplet probability constraints to solve the unconstrained face analysis problem. While previous approaches in this area have proposed scalable yet ad-hoc solutions to this problem, we propose a principled and parameter free formulation which is based on maximum likelihood estimation. In addition, we employ the principle of transfer learning to realize a deep network architecture that can train faster and on lesser data yet significantly outperforms existing approaches on the unconstrained face verification task. We demonstrate the robustness of the approach to challenges including age, pose, blur and clutter by performing clustering experiments on challenging benchmarks.

Recent seminal works have shown that deep neural networks are susceptible to visually imperceptible perturbations of the input. In this dissertation, we build

on their ideas in two unique ways: (a) We show that neural networks that perform pixel-wise semantic segmentation tasks also suffer from this vulnerability, despite being trained with more extra information compares to simple classification tasks. In addition, we present a novel self correcting mechanism in segmentation networks and provide an efficient way to generate such perturbations (b) We present a novel approach to regularize deep neural networks by perturbing intermediate layer activations in an efficient manner, thereby exploring the trade-off between conventional regularization and adversarial robustness within the context of very deep networks. Both of these works provide interesting directions towards understanding the secure nature of deep learning algorithms.

While humans find it extremely simple to generalize their knowledge across domains, machine learning algorithms including deep neural networks suffer from the problem of domain shift across what are commonly termed as 'source' (S) and 'target' (T) distributions. Let the data that a learning algorithm is trained on be sampled from S. If the real data used to evaluate the model is then sampled from T, then the learnt model under-performs on the target data. This inability to generalize is characterized as domain shift. Our attempt to address this problem involves learning a common feature subspace, where distance between source and target distributions are minimized. Estimating the distance between different domains is highly non-trivial and is an open research problem in itself. In our approach we parameterize the distance measure by using a Generative Adversarial Network (GAN). A GAN involves a two player game between two mappings com- monly termed as generator and discriminator. These mappings are learned simultaneously

by employing an adversarial game, i.e. by letting the generator fool the discriminator and enabling the discriminator to outperform the generator. This adversarial game can be formulated as a minimax problem. In our approach, we learn three mappings simultaneously: the generator, discriminator and a feature mapping that contains information about both the content and the domain of the input. We deploy a two-level minimax game, where the first level is a competition between the generator and a discriminator similar to a GAN; the second level game is where the feature mapping attempts to fool the discriminator thereby introducing domain invariance in the learned feature representation. We have extensively evaluated this approach for different tasks such as object classification and semantic segmentation, where we achieve state of the art results across several real datasets. In addition to the conceptual novelty, our approach presents a more efficient and scalable solution compared to other approaches that attempt to solve the same problem.

In the final part of this dissertation, we describe some ongoing efforts and future directions of research. Inspired from the study of perturbations described above, we propose a novel metric on how to effectively choose pixels to label given an image, for a pixel-wise segmentation task. This has the potential to significantly reduce the labeling effort and our preliminary results for the task of semantic segmentation are encouraging. While the domain adaptation approach proposed above considered static images, we propose an extension to video data aided by the use of recurrent neural networks. Use of full temporal information, when available, provides the perceptual system additional context to disambiguate among smaller object classes that commonly occur in real scenes.

Towards robust and domain invariant feature representations in Deep Learning

by

Swaminathan Sankaranarayanan

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
2018

Advisory Committee:
Professor Rama Chellappa, Chair/Advisor
Professor Larry Davis
Professor Behtash Babadi
Professor Joseph Jaja
Professor David Jacobs (Dean's Representative)

Dedication

To my parents, who were extremely supportive in every move I made towards my path to science and who led the way in showing how to pick oneself up when the chips are down.

Acknowledgements

I would like to thank my advisor Prof. Chellappa, who has provided me with much needed direction and sage academic advice over the past years of my study. He has always given me freedom to pursue ideas in my own way and for that I am extremely grateful. The discussions I have had with him over the years has broadened my perspective on the field of computer vision and indeed, have served as a lesson in diplomacy! I feel honored to have worked under such a great researcher and a wonderful person.

It is an honor to have Professor Larry Davis, Professor David Jacobs, Professor Joseph Jaja and Professor Behtash Babadii in my dissertation committee. I am thankful to them for serving in my committee and providing insightful and diverse suggestions to improve this dissertation.

I am very much thankful for my mentors during my time at GE Global Research. SerNam and Arpit have both been incredible mentors to me both at a technical and personal level. The work ethic and technical acumen displayed by SerNam amazes me and continually inspires me to perform my best work. I am extremely thankful to him for providing unlimited creative freedom to explore a diverse range of ideas. Arpit is an excellent researcher who was my academic mentor during my time with GE and afterward. He provided me with great technical counsel and also some much needed life lessons, which surely helped me survive the better part of Albany winter! Their support was extremely important and timely in that it shaped the later part my PhD research.

I would like to thank all my colleagues at UMIACS for making my graduate life memorable. I would also like to thank UMIACS staff for providing valuable tech support over the years.

Graduate study is possibly one of the most testing times during one's life. I would like to thank my network of friends who helped ease the pressure. A special thanks to Vidya, who was pillar of support during my entire PhD; who put up with me during the tough times and was there to celebrate the bright moments. To you, I owe a lot!

I am more of a skeptic about most things in life, but I do believe in how family is important in shaping someone's thought process and attitude. I have had a wonderful family who supported me throughout my PhD and gave me a lot of freedom in pursuing my career.

Last but not the least in any sense, I am much thankful to a person, rather a character, who inspired me to pursue science when I started reading his stories as a kid. Sherlock Holmes, a salute to you Sir!

Contents

List of Tables

List of Figures

NOTATIONS

- Vectors are represented using boldface lowercase letters such as {x,y}
- Matrices are represented using boldface uppercase letters such as {X,Y}
- Parametric mappings are denoted as italicized uppercase letters such as $\{\boldsymbol{D}, D\}$
- $\exp(a)$ - Exponential of scalar a.
- $\ln(a)$ - Natural logarithm of scalar a.
- $\mathbb{R}^n$ - n-dimensional real vector space.
- Sym_d^+ - space of d-dimensional real symmetric positive definite matrices
- $\boldsymbol{I}$ - Identity matrix of appropriate size.
- $\mathbf{1}$ - Matrix of appropriate size with all ones.
- $\mathbf{0}$ - Matrix of appropriate size with all zeros.
- A^T - Transpose of matrix A.
- trace(A) - Trace of matrix A.
- span(A) or [A] - Column span of matrix A.
- $|||\mathrm{A}|||_F$ - Frobenius norm of matrix A.
- $\mathrm{A} \succeq 0$ - $mrmA$ is symmetric and positive semidefinite.
- $\mathrm{A} \succ 0$ - A is symmetric and positive definite.
- $\|\mathrm{a}\|_2$ - ℓ_2 norm of vector $\boldsymbol{a}$.

Chapter 1. Introduction

1.1 Motivation

Feature representations have been the cornerstone of perception-based systems which combine algorithms from machine learning and computer vision to enable a machine understand the natural scene. Starting from the early work of David Marr [Mar82], that provides a computational framework for processing the visual information, several advances have been made in how to efficiently compress the information yet retain as much information content as possible. These started as naturally occurring features in the images such as corners, edges [S+94] and associated gradients such as optical flow [PCF06], followed by sophisticated hand-crafted features such as Scale Invariant Feature Transform (SIFT) [Low04], Histogram of Gradients (HOG) [DT05] and Speeded up Robust Features (SURF) [BTVG06]. These feature representations were shown to remarkably improve the ability of a learning algorithm to perform visual tasks such as identifying the class of a given image. Handcrafted representations lack the ability to learn and adapt to variations due to illumination, pose, resolution and distrbutional shifts. Artificial Neural Networks (ANNs) have been studied for more than three decades with the advent of algorithms such as Neocognitron [FM82] and Hopfield networks [Hop82]. These were followed

by early instances of the modern day convolutional neural networks by Lecun *et al* [LBBH98]. While these algorithms provided a way to simulate a neural process of learning and adapting to different forms of distortions presented in the input, they were notoriously hard to train or converge if provided with a large collection of input images. Concurrent development in statistical machine learning, optimization theory and computational breakthroughs such as use of Graphical Processing units (GPUs) led to performing backpropagation in a computationally efficient manner. The notion of "deep" networks, which is the subject of this dissertation, came about since the work of Krizhevsky *et al.* [KSH12a], who demonstrated a significant improvement in performance on the Imagenet data challenge by training a deep neural network consisting of 60M parameters to learn, using backpropagation. Since then, deep networks have been adopted by the machine learning and computer vision communities and achieved state of the art results on several benchmarks.

Despite impressive performance gains, deep networks are not known to obtain great generalization. On the one hand, this statement does not treat DNNS fairly, since studies have shown that they generalize better than previously existing learning algorithms and feature representations. On the other hand, if DNNs do indeed simulate the neural process of modeling visual stimuli, then one would expect them to model the underlying structure of an object if enough examples of the object are given and thereby generalize well to different appearances of the object due to different viewpoints, blur and other forms of noise, especially given that these are easier tasks for humans. Thus, in this dissertation, we study the behavior and performance of DNNs on data which is drawn from a different distribution compared

to the data using which the networks were trained. This difference can be a result of natural distortion due to viewpoint changes in face analysis, adversarial perturbations of the input or explicit shift in the data distribution such as synthetic/real datasets.

1.2 Proposed Approaches and Contributions

In this section, we briefly describe the approaches introduced in this dissertation and their key contributions.

Triplet Probabilistic Embedding: In this approach, we propose a greedy optimization method to construct a low-dimensional embedding of the deep features that are more robust to external nuisance factors yet augment the discriminative information present in the original representation. To obtain the original representation, we propose a deep network architecture for the face recognition task and a faster training approach inspired from prior work in transfer learning. The key contributions include a faster training mechanism that can yield better performance on relatively smaller datasets and a greedy online optimization based on a maximum likelihood formulation that replaces the ad-hoc solutions to the same problem proposed in previous works. We show the efficacy of the above approaches on challenging benchmarks such as the IJB-A dataset and on controlled settings such as Celebrity in Frontal-Profile datasets, both of which contain celebrity images in the wild. We show through clustering experiments how the proposed embedding approach results in better qualitative description of the nuisance factors present in the

data.

Guided Perturbations: While TPE dealt with naturally occurring distortions to the input images, in this approach, we address a specific type of distortion called "adversarial perturbations" which are minor perturbations of the feature space that when carried over to the input destabilizes the behavior of well trained deep networks. The effect of these perturbations have been well studied for classification problems in the past, but in this thesis we shed light on their effect on the task of semantic segmentation. We present an intriguing behavior: pretrained deep networks can be made to improve their predictions by structurally perturbing the input. We observe that these perturbations - referred as Guided Perturbations - enable a trained network to improve its prediction performance without any learning or change in network weights. We perform various ablative experiments to understand how these perturbations affect the local context and feature representations. Our studies show that guided perturbations are a by product of adversarial perturbations for the case of the semantic segmentation task, which includes an explicit ensemble decision step at the output. Furthermore, we demonstrate that this idea can improve the performance of several existing approaches on the semantic segmentation and scene labeling tasks.

Layerwise Adversarial Regularization: A traditional approach to mitigate the effect of adversarial perturbations mentioned in the previous work is to infuse adversarial noise during the training of the deep network. In earlier works, adver-

sarial training has been shown to regularize small scale neural networks in addition to increasing their robustness to adversarial examples. However, its impact on very deep state of the art networks has not been fully investigated. We present an efficient approach to perform adversarial training by perturbing intermediate layer activations and study the use of such perturbations as a regularizer during training. We use these perturbations to train very deep models such as ResNets and show improvement in performance both on adversarial and original test data. Our experiments highlight the benefits of perturbing intermediate layer activations compared to perturbing only the inputs. The results on CIFAR-10 and CIFAR-100 datasets show the merits of the proposed adversarial training approach. Additional results on WideResNets show that our approach provides significant improvement in classification accuracy for a given base model, outperforming dropout and other base models of larger size. Furthermore, our ablative experiment on training with corrupted labels shows the significant advantage provided by the proposed layerwise regularization approach.

Generative approach to address Domain Shift: Visual domain adaptation is a problem of immense importance in computer vision. Previous approaches showcase the inability of even deep neural networks to learn informative representations across domain shifts. This problem is more severe for tasks where acquiring hand labeled data is extremely hard and tedious. In this approach, we focus on adapting the representations learned by segmentation networks across synthetic and real domains. Contrary to previous approaches that use a simple adversarial objective or

superpixel information to aid the process, we propose an approach based on Generative Adversarial Networks (GANs) that brings the embeddings closer in the learned feature space. To showcase the generality and scalability of our approach, we show that we can achieve state of the art results on two challenging scenarios of synthetic to real domain adaptation. Additional exploratory experiments show that our approach: (1) generalizes to novel domains and (2) results in improved alignment of source and target distributions.

1.3 Organization

Chapter 2 introduces the ideas of metric learning, deep learning and convolutional neural networks, generative modeling including generative adversarial networks and variational autoencoders, which will be used in subsequent chapters of this dissertation. Chapter 3 presents the triplet probabilistic embedding approach for face analysis. Chapter 4 discusses the effect and presents a thorough experimental analysis of input perturbations on neural networks primarily designed for semantic segmentation. Chapter 5 describes the layerwise adversarial regularization approach for very deep state of the art deep networks. Chapter 6 addresses the problem of domain shift at scale for the task of semantic segmentation, describing our solution based on a generative adversarial network. Chapter 7 concludes the dissertation and discusses future research directions.

Chapter 2. Background and Literature Review

2.1 Deep Learning and Convolutional Neural Networks

In this section, we provide a brief review of the properties of Convolutional Neural Networks (CNNs) and their widespread use in computer vision. For a more detailed review the reader is referred to [GBC16].

A standard Neural Network (NN) consists of many functional units called neurons each producing a sequence of real-valued activations. Input neurons get activated through sensors per- ceiving the environment, other neurons get activated through weighted connections from previously active neurons.Some neurons may influence the environment by triggering actions. Designing a learning algorithm is about finding weights that make the NN exhibit desired behavior, such as driving a car. Depending on the problem and how the neurons are connected, such behavior may require long causal chains of computational stages (Sec. 3), where each stage transforms (often in a non-linear way) the aggregate activation of the network. Deep Learning is about accurately assigning the parameters across many such stages in order to obtain the desired behavior. For more details about evolution of the field, the readers may refer to [Sch15].

Traditionally, neural network approaches have been used to learn complex

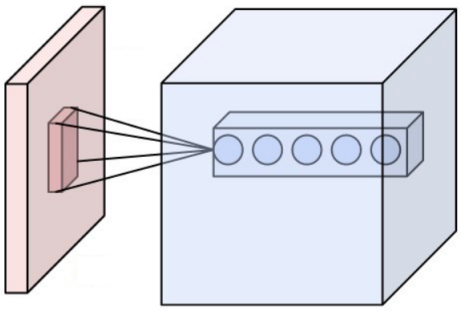


Figure 2.1: Convolutional layer with each neuron connecting to its receptive field at the input. Figure from [Kar16]

functional dependencies by considering input features and neurons as vector inputs. To specialize this idea to structures inputs like images, Convolutional Neural Networks (CNN) were developed. In CNN's, the input to the neural nets are images and the neurons are convolutional filters which connect to a local region of the input. A convolutional layer, which is the core building block of a CNN, is shown in Figure 2.1. The layer's parameters consist of a set of learnable filters (or kernels), which have a small receptive field, but extend through the full depth of the input volume. The network operates in two modes: forward pass and backward pass. During the forward pass, each filter is convolved across the width and height of the input volume, computing the dot product between the entries of the filter and the input and producing a two-dimensional activation map of that filter. As a result, the network learns filters that activate when they see some specific type of feature at some spatial position in the input.

Stacking the activation maps for all filters along the depth dimension forms the full output volume of the convolution layer. Every entry in the output volume

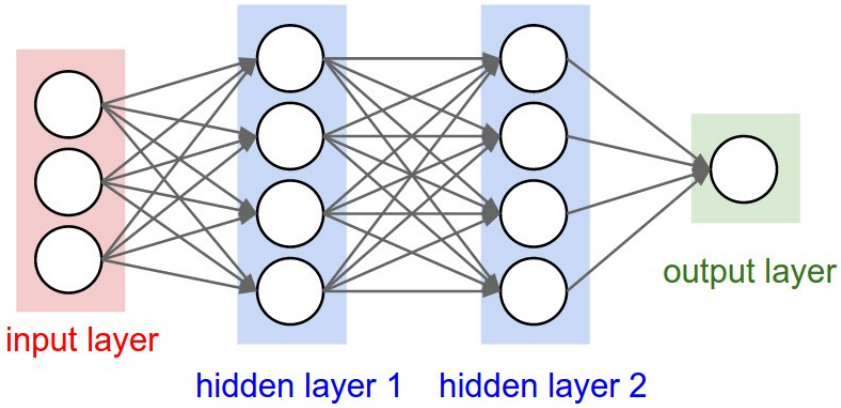


Figure 2.2: A simple (shallow) 3-layer neural network with two hidden layers and one output layer. Figure from [Kar16]

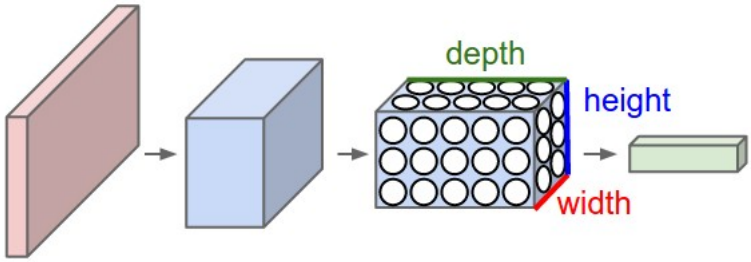


Figure 2.3: Arrangement of layers in a CNN. Figure from [Kar16]

can thus also be interpreted as an output of a neuron that looks at a small region in the input and shares parameters with neurons in the same activation map.

During the backward pass, the parameters of the convolutional filters are updated based on the gradient of the loss function that is propagated backwards along the network. This procedure is aptly termed as *backpropagation*. More details regarding the computation of gradients for traditional CNN architectures can be found in [IGC16]. As an example, let us consider a simple neural network, shown in Figure 2.2 and perform gradient computation using backpropagation.

A CNN arranges its neurons in three dimensions (width, height, depth), as

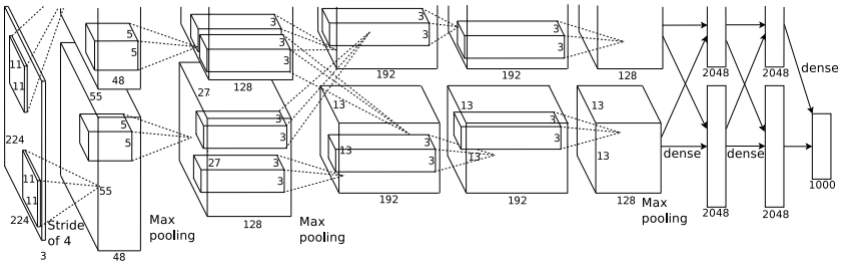


Figure 2.4: Architecture of AlexNet - winner of 2012 ImageNet Object Classification Challenge. Figure from [KSH12a]

visualized in Figure 2.3 in one of the layers. Every layer of a CNN transforms the 3D input volume to a 3D output volume of neuron activations. In this example, the red input layer holds the image, so its width and height would be the dimensions of the image, and the depth would be 3 (Red, Green, Blue channels).

More complex architectures for image classification tasks were proposed over the years. One very popular architecture is *AlexNet* [KSH12a] which was the winner of the 2012 ImageNet Classification Challenge. The network is shown in Figure 2.4.

The main functional units of AlexNet architecture are explained below:

- Convolutional Layer: computes the output of neurons that are connected to local regions in the input, each computing a dot product between their weights and a small region they are connected to in the input volume.

- Max. Pooling: performs a downsampling operation along the spatial dimensions (width, height) by taking a maximum over pre specified window size.

- Rectified Liner Unit (ReLU): applies an elementwise activation function, such as the max(0,x) thresholding at zero. This leaves the size of the volume un-

changed.

- Stride: applies the convolutional filter at specified stride - higher strides results in less operations and sparser feature maps which are beneficial for classification.

- Fully Connected Layer (FC): learns a linear classifier over the convolutional features and the output neurons equal the number of input classes over the training dataset.

- Dropout: strategy employed to learn FC layers that has been shown to improve convergence. During training, a *random* fraction of connections is deactivated and only the rest are updated. The fraction is specified as a parameter for the algorithm.

Several strategies have been employed to learn the parameters of a deep architecture. The most common learning algorithm, that is also employed throughout this dissertation, is the mini-batch Stochastic Gradient Descent (SGD), which is a standard gradient descent technique with gradients computed over a mini batch of training examples at each iteration. Since the function learnt by a deep network is typically a non-convex function the possibility of getting stuck in local minima is very high. To avoid such pitfalls, the gradient descent update is augmented using *momentum* [SMDH13].

Let the magnitude of weights at time t be denoted by W_t. Let the update value be V_{t+1} and the updated weights be W_{t+1} at iteration $t+1$. Then given the

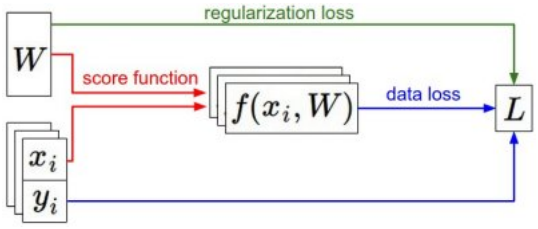


Figure 2.5: Simplified Dataflow diagram for training a traditional CNN such as AlexNet

update V_t and current weights W_t, the update with momentum can be written as:

$$V_{t+1} = \mu V_t - \alpha \nabla L(W_t); W_{t+1} = W_t + V_{t+1} \tag{2.1}$$

where α is the learning rate and μ is the momentum, which is a scalar.

The dataset of image-label pairs (X, Y) is given and fixed. The weights start out as random numbers and can change. During the forward pass the score function computes class scores, stored in vector f. The loss function contains two components: The data loss computes the compatibility between the scores f and the labels y. The regularization loss is only a function of the weights. During gradient descent, we compute the gradient on the weights (and optionally on data if we wish) and use them to perform a parameter update.

The deep features that are learnt over different layers of the AlexNet network are shown in Figure 2.6. The important observation from the visualization is that the shallow layers learn more basic and general features like corners/edges which are combined in the higher layers to learn abstract concepts like face/eyes etc. In Chapter 3, we utilize this property of deep architectures to design a deep network that has accelerated convergence using the knowledge already learned from a different task.

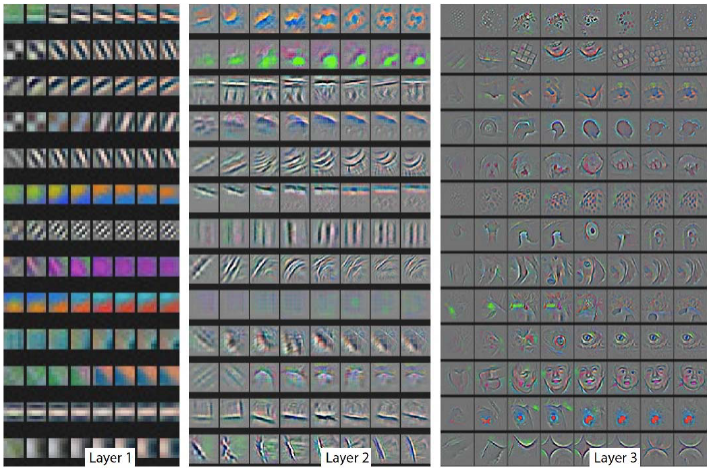


Figure 2.6: Deeper layers learn higher levels of abstraction. Figure from [Kar16].

2.2 Metric Learning

Consider a situation where we must compute similarity or distances over pairs of images (for example, for clustering or nearest neighbor classification). A basic question that arises is precisely how to assess the similarity or distance between the pairs of images. For instance, if our goal is to find matching faces based on identity, then we should choose a distance function that emphasizes appropriate features (hair color, ratios of distances between facial keypoints, etc). But we may also have an application where we want to determine the pose of an individual, and therefore require a distance function that captures pose similarity. To handle multiple similarity or distance metrics, we could attempt to determine by hand an appropriate distance function for each task, using an appropriate choice of features and the combination of those features. However, this approach may require significant effort and may not be robust to changes in the data. A desirable alternative is to apply metric learning,which aims to automate this process and learn task-specific

distance functions in a supervised manner.

As discussed above, specific features are important for specific tasks and the success of the metric learning method depends on the type of features used. Assuming that we are already given a suitable representation (say, SIFT/HOG/CNN features) we would like to learn a parameterized mapping $\boldsymbol{D} : \mathbb{R}^N \times \mathbb{R}^N \rightarrow \mathbb{R}_0^+$, that takes in two feature vectors that belong to a N-dimensional space and outputs a scalar value. Note that, the feature vectors could belong to more complex spaces such as Riemannian manifold spaces or need not be vectors at all. We have considered only $\mathbb{R}^N$ here for the purpose of illustration.

Before going into more specifics, lets define what we mean by a metric. A mapping $\boldsymbol{D} : \mathbb{X} \times \mathbb{X} \rightarrow \mathbb{R}_0^+$ over a vector space $\mathbb{X}$ is called a metric if for all vectors $\{\mathrm{x}_i, \mathrm{x}_j, \mathrm{x}_k\} \in \mathbb{X}$, it satisfies the following properties:

- Triangular inequality: $\boldsymbol{D}(\mathrm{x}_i, \mathrm{x}_j) + \boldsymbol{D}(\mathrm{x}_j, \mathrm{x}_k) \geq \boldsymbol{D}(\mathrm{x}_i, \mathrm{x}_k)$
- Non negativity: $\boldsymbol{D}(\mathrm{x}_i, \mathrm{x}_j) \geq 0$
- Symmetry: $\boldsymbol{D}(\mathrm{x}_i, \mathrm{x}_j) = \boldsymbol{D}(\mathrm{x}_j, \mathrm{x}_i)$
- Distinguishability: $\boldsymbol{D}(\mathrm{x}_i, \mathrm{x}_j) = 0 \iff \mathrm{x}_i = \mathrm{x}_j$

In strict terms, a mapping that satisfies only the first three conditions is called a *pseudometric* but in this dissertation we use the terms metric and pseudometric interchangeably.

In this dissertation, we consider only linear metric learning methods, since the input features used are from CNN's which are highly optimized non-linear functions

of the input. More details on different types of metric learning approaches could be found in [Kul12]. Specifically, we focus on the class of methods which learn a Mahalanobis metric based on a given set of constraints. The constraints could be of the following types:

- First-order: constraints of the form that x belongs to class c
- Pairwise: (x, y) belong to the same or different class
- Triplet : $(\mathrm{x}, \mathrm{y}, \mathrm{z})$ such that x is more similar to y than z.

In this approach, we work with metrics that deal with either pairwise or triplet constraints. One of the main proponents of metric learning approaches over pairwise and triplet constraints is the Large Margin Nearest Neighbour (LMNN) metric learning approach [WBS05]. The loss function considered in LMNN is the following:

$$L(\boldsymbol{M}) = (1-\mu) \sum_{i,j \in P} \boldsymbol{D_M}(\mathrm{x}_i, \mathrm{x}_j) + \mu \sum_{i,j} \sum_{l \in N} max(1 + \boldsymbol{D_M}(\mathrm{x}_i, \mathrm{x}_j) - \boldsymbol{D_M}(\mathrm{x}_i, \mathrm{x}_l), 0) \tag{2.2}$$

where $\boldsymbol{D_M}(\mathrm{x}_i, \mathrm{x}_j) = (\mathrm{x}_i - \mathrm{x}_j)^T \boldsymbol{M} (\mathrm{x}_i - \mathrm{x}_j)$; P corresponds to the set of positives which contain items that belong to the same class as x_i and N is the set of negatives which contain items that belong to a different class than x_i. To understand the loss function lets consider the two terms separately. The first term $\boldsymbol{D_M}(\mathrm{x}_i, \mathrm{x}_j)$ considers the distance between items from the same class: intra-class distance. The second term is active only if: $\boldsymbol{D_M}(\mathrm{x}_i, \mathrm{x}_j) > 1 + \boldsymbol{D_M}(\mathrm{x}_i, \mathrm{x}_l) > 0$, that is, if the distance between the positive pairs $(\mathrm{x}_i, \mathrm{x}_j)$ is greater than the distance between negative

pairs $(\mathrm{x}_i, \mathrm{x}_l)$. Thus, optimizing this loss function brings together samples from the same class and pushes apart samples from the different class. This would then result in better discrimination between classes in a given data. Since the above formulation is convex in $\boldsymbol{M}$, the optimization problem is solved by framing it in the form of a Semi-definite Programming (SDP) problem. In its original form, this does not scale to large amounts of data since solving an SDP becomes very expensive when presented with millions of data points. We propose a simple modification of the above problem that could be solved in a very scalable way.

2.3 Adversarial Machine Learning

In machine learning, the notion of adversarial learning relates to modeling secure learning systems. A detailed description can be obtained from the work of Huang *et al.* [HJN+11]. Here we provide a short overview. In general, these systems are modeled as a game between an attacker and a defender where the attacker's job is to manipulate data to evade the learning algorithm picked by the defender in order to thwart the defender's objective. This game can be formalized in terms of a learning algorithm H, the model M and the attackers data corruption strategies A_{train} and A_{eval}. The resulting game can be described as follows:

- Defender: Choose learning algorithm H and model M for selecting hypotheses based on observed data
- Attacker: Choose attack procedures A_{train} and A_{eval}. This can be done in a white-box setting (with the knowledge of H and M or black-box setting

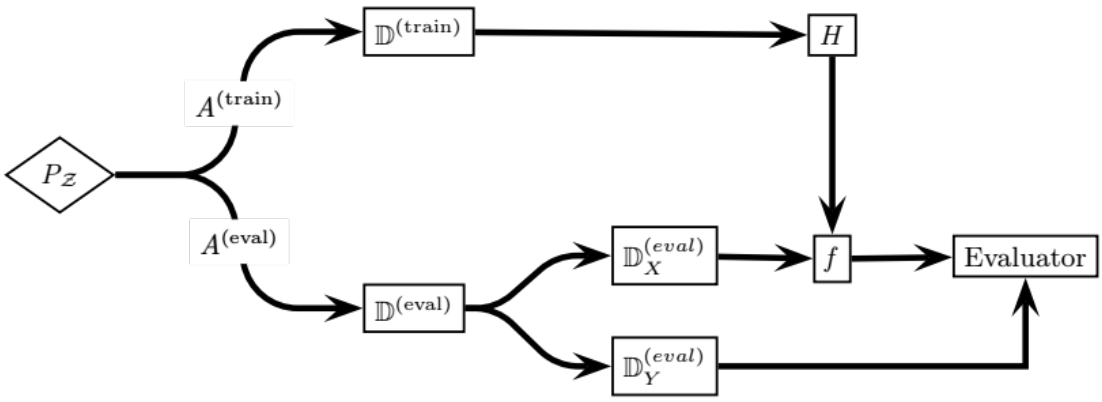


Figure 2.7: Adversarial game between attacker and defender. Figure taken from [HJN+11]. Refer to the text for more details.

(input-output knowledge obtained by querying the model in finite time)

- Learning: Obtain dataset D_{train} with contamination from A_{train}. Learn hypothesis f ∈ H.
- Evaluation: Obtain dataset D_{eval} with contamination from A_{eval}. Compare predictions f(x) to y for each data point $(x, y) \in D_{eval}$.

The game described above includes the interactions between the defender and attacker in choosing the model and attack strategies. These steps are depicted in Figure 2.7. The defender chooses H to select hypotheses that predict well regardless of A_{train} and A_{eval}, while the attacker chooses A_{train} and A_{eval} to produce poor predictions. Furthermore, different types of game strategies define different valid moves that players can make. In exploratory attacks, the procedure A_{train} is not used in the game, and thereby the attacker only influences D_{eval}. Meanwhile, in the causative game the attacker also has indirect influence on f through its choice of A_{train}. In

an integrity attack, the attacker desires false negatives and therefore will use A_{train} and/or A_{eval} to create or discover false negatives, whereas in an availability, the attacker will also try to create or exploit false positives. Finally, in a targeted attack the attacker only cares about the predictions for a small number of instances, while an indiscriminate attacker cares about prediction for a broad range of instances. In this section, we have described an adversarial machine learning system in general. In the next section, we describe a concrete application of this system for generative modeling, namely generative adversarial networks, which employ on a two player adversarial game to train a generative model.

2.4 Generative Modeling and Variational Inference

Generative modeling involves the process of learning models that can allow us to sample from the underlying true data distribution. This has been a long standing open research topic in machine learning and over the past decade has spawned a plethora of research. Earlier approaches to generative modeling included Markov Random Fields (MRFs) and its variants, sampling techniques such as Monte-Carlo approaches and their variants. Even though these techniques are in use today, with the growing amounts of data and computational power, recent approaches have focused on developing more efficient sampling schemes by coming up with approximate solutions to the density estimation problem. In this section, we will review two recent popular approaches to generative modeling: Variational Auto Encoders (VAEs) which are based on Variational Inference and Generative Adversarial Net-

works (GANs) which rely on an adversarial game.

2.4.1 Generative Adversarial Networks: Overview

In this section, we provide a brief overview of GANs accompanied by a standard derivation of the equilibrium point of the adversarial game. For a more detailed analysis, we suggest the reader to refer to the work of Goodfellow *et al.* [GPAM+14]. Most work on deep generative models focused on models that provided a parametric specification of a probability distribution function. The model can then be trained by maximizing the log likelihood. In this family of model, perhaps the most successful is the deep Boltzmann machines [SL10]. Such models generally have intractable likelihood functions and therefore require numerous approximations to the likelihood gradient. GANs belong to the class of algorithms commonly termed as "generative machines", which can be trained without exact parameterization of the likelihood function and by using exact backpropagation. In the GAN framework, the generative model is pitted against an adversary: a discriminative model that learns to determine whether a sample is from the model distribution or the data distribution. As observed in [GPAM+14], the generative model can be thought of as analogous to a team of counterfeiters, trying to produce fake currency and use it without detection, while the discriminative model is analogous to the police, trying to detect the counterfeit currency. Competition in this game drives both teams to improve their methods until the counterfeits are indistinguishable from the genuine currency.

2.4.2 Problem Setup and Theoretical Results:

Let X be the data drawn as samples from the true distribution p_{data}. A GAN consists of two mappings learned simultaneously: generator $\mathrm{G}(\mathrm{z};\theta_g)$ and discriminator $\mathrm{D}(\mathrm{x};\theta_d)$. For ease of understanding, both G and D are differential functions modeled as multi layer neural networks. To learn the generators distribution p_g over data X, we define a prior on input noise variables $p_z(\mathrm{z})$, then represent a mapping to data space as $\mathrm{G}(\mathrm{z};\theta_g)$. $\mathrm{D}(\mathrm{x};\theta_d)$ outputs a single scalar. D(x) represents the probability that x is from the data rather than the generator distribution p_g. D is trained to maximize the probability of assigning the correct label to both training examples and samples from G, which is simultaneously trained to minimize the log loss, $log(1\mathrm{D}(\mathrm{G}(\mathrm{z})))$. In other words, D and G play the following two-player minimax game with the value function $V(\mathrm{G},\mathrm{D})$:

$$\min_{G} \quad \max_{D} \quad V(\mathrm{G},\mathrm{D}) = \mathbb{E}_{\mathrm{x}\sim p_{data}}[f(\mathrm{D}(\mathrm{x}))] + \mathbb{E}_{\mathrm{z}\sim p_{noise}}[f(1-\mathrm{D}(\mathrm{G}(\mathrm{z})))] \tag{2.3}$$

where f is any non-increasing function. A commonly used choice is the log function. The training involves alternating stochastic optimization steps of ascending the gradient of D and descending the gradient of the generator G. In what follows, we will prove two results that compute the global optimum of the game presented above and show that it can be achieved given infinite capacity and training time.

Proposition For G fixed, the optimal discriminator D is given by $\mathrm{D}^*_{\mathrm{G}} = \frac{p_{data}}{p_{data}+p_g}$

Proof The objective is to maximize the quantity $V(\mathrm{G},\mathrm{D})$.

$$\min_{G} \max_{D} V(\mathrm{G},\mathrm{D}) = \int_{\mathrm{x}} p_{data}(\mathrm{x}) \log(\mathrm{D}(\mathrm{x}))dx + \int_{\mathrm{z}} p_{noise}(\mathrm{z}) \log(1-\mathrm{D}(\mathrm{G}(\mathrm{z})))dz$$
$$= \int_{\mathrm{x}} (p_{data}(\mathrm{x}) \log(\mathrm{D}(\mathrm{x}))dx + p_g(\mathrm{z}) \log(1-\mathrm{D}(\mathrm{x})))dx \qquad (2.4)$$

It remains then to find the value of $\mathrm{D}(\mathrm{x}) \in (0,1)$ which maximizes the value function formulated above. By simple algebra, we obtain $\mathrm{D}^*_{\mathrm{G}}(\mathrm{x}) = \frac{p_{data}}{p_{data}+p_g}$, concluding the proof.

Alternatively, the training objective for D can also be interpreted as maximizing the log-likelihood for estimating the conditional probability $P(Y = y|\mathrm{x})$, where Y indicates whether x comes from p_{data} (with $y = 1$) or from p_g (with $y = 0$). The minimax game in Eq. 2.3 can now be reformulated as:

$$C(\mathrm{G}) = \max_{\mathrm{D}} V(\mathrm{G},\mathrm{D})$$
$$= \mathbb{E}_{\mathrm{x}\sim p_{data}}[\log(\mathrm{D}^*_{\mathrm{G}}(\mathrm{x}))] + \mathbb{E}_{\mathrm{x}\sim p_{data}}[\log(1-\mathrm{D}^*_{\mathrm{G}}(\mathrm{x}))]$$
$$= \mathbb{E}_{\mathrm{x}\sim p_{data}}[\log(\frac{p_{data}}{p_{data}+p_g(\mathrm{x})})] + \mathbb{E}_{\mathrm{x}\sim p_{data}}[\log(1-\frac{p_{data}}{p_{data}+p_g(\mathrm{x})})] \qquad (2.5)$$

Theorem The global minimum of the virtual training criterion $C(\mathrm{G})$ is achieved if and only if $p_g = p_{data}$. At that point, $C(\mathrm{G})$ achieves the value -log 4.

Proof At $p_g = p_{data}$, it follows from Eq. 2.4 that $\mathrm{D}^*_{\mathrm{G}}(x) = 1/2$. Substituting this in Eq. 2.5, we see that $C(\mathrm{G}) = -\log 4$. Conversely, to see that this is the best possible value reached for $C(\mathrm{G})$, observe that the following hold:

$$
\begin{aligned}
C(\mathrm{G}) &= -\log(4) + \log(4) + \max_{\mathrm{D}} \quad V(\mathrm{G},\mathrm{D}) \\
&= -\log(4) - 2\log(1/2) + \int_{\mathrm{x}} (p_{data}(\mathrm{x}) \log(\frac{p_{data}}{p_{data} + p_g(\mathrm{x})}) \\
&+ p_g(\mathrm{x})(1 - \log(\frac{p_{data}}{p_{data} + p_g(\mathrm{x})})))dx \\
&= -\log(4) + KL(p_{data}||\frac{p_{data} + p_g}{2}) + KL(p_g||\frac{p_{data} + p_g}{2}) \\
&= -\log(4) + 2\ JSD(p_g||p_{data}) \qquad (2.6)
\end{aligned}
$$

Since the Jensen-Shannon divergence between two distributions is non-negative, the global minimum of $C(\mathrm{G})$ is $-\log(4)$, which is achieved when $p_g = p_{data}$, concluding the proof.

2.4.3 Variational Autoencoders

In the previous section, we described GANs, a class of generative models that learn a generator mapping by framing it as an adversarial game, which at the equilibrium point yields the optimal solution, $p_g = p_{data}$. The issue with GANs is when this equilibrium is achieved and more broadly, does this equilibrium exist ? These two questions have spanned a plethora of research in the recent years and still remain an actively researched open problem. Several techniques have been proposed to stabilize the GAN training but without giving any theoretical guarantees. In this section, we describe a generative model that gives a nice theoretical guarantee of convergence by maximizing a loss function that can be formulated as a lower bound of the true data likelihood. This brings us to the topic of Variational Autoencoders (VAEs). We provide a brief overview of the mechanism of the VAEs and derive

the evidence lower bound being maximized by a VAE. For detailed exposition, the reader is referred to the works of Kingma *et al.* [KW13] and Doersch *et al.* [Doe16].

An autoencoder is a multi-layer neural network that is optimized to reconstruct the input signal at the output. Training an autoencoder without explicit regularization techniques results in overfitting, leading the model to assign zero variance to points in the training set, i.e. memorizing the training data. Several variants were proposed to alleviate this issue including contractive or denoising versions. Variational autoencoders tackle this problem by introducing stochasticity when sampling from the latent space.

2.4.4 VAE: Problem Setup

Consider some dataset $\mathrm{X} = \{\mathrm{x}^{(i)}\}_{i=1}^{N}$ consisting of N i.i.d. samples of some continuous or discrete variable $\mathbf{x}$. We operate under the assumption that the data are generated by some random process, involving an unobserved continuous random variable $\mathbf{z}$ following a two-step procedure: (1) a value $\mathrm{z}^{(i)}$ is generated from some prior distribution $p_{\theta^*}(\mathrm{z})$ (2) a value $\mathrm{x}^{(i)}$ is generated from some conditional distribution $p_{\theta^*}(\mathrm{x}|\mathrm{z})$. The true parameters θ^* as well as the values of the latent variables $\mathrm{z}^{(i)}$ are unknown. The VAE is based on the Autoencoding Variational Bayes (AEVB) algorithm that does not make common simplifying assumptions. More specifically, the estimator should work well in cases where: (a) the marginal likelihood $p_\theta(\mathrm{z})$ and the posterior density $p_\theta(\mathrm{z}|\mathrm{x})$ are intractable (hence algorithms like EM cannot be applied (b) The datasets used to learn the parameters are large hence sampling techniques such as MCMC approaches become intractable. The AEVB algorithm

tackles these issues by optimizing a variational lower bound and designing a Stochastic Variational estimator to approximate the true posterior density.

2.4.5 Variational Lower Bound and AEVB algorithm

The marginal log likelihood over data is composed of a sum over the marginal log likelihoods of individual datapoints, $\log\ p_\theta(\mathrm{x}^{(1)}, \mathrm{x}^{(2)}, ..., \mathrm{x}^{(N)}) = \sum_{i=1}^{N} \log\ p_\theta(\mathrm{x}^{(i)})$. The technique of variational inference involves estimating an intractable density by assuming a proposal distribution and minimizing the distance between it and the sampled true density, thereby converting the density estimation problem to an optimization problem. For our case, lets assume the proposal distribution to be $q_\phi(\mathrm{z}|\mathrm{x})$, which is called the encoder, since it encodes the inputs x to the latent code z. Each data likelihood term can then be expressed using the proposal distribution as follows:

$$\begin{aligned}
\log p_\theta(\mathrm{x}^{(i)}) &= \int q_\phi(\mathrm{z}|\mathrm{x}^{(i)}) \log p_\theta(\mathrm{x}^{(i)}) dz \\
&= \int q_\phi(\mathrm{z}|\mathrm{x}^{(i)}) (\log p_\theta(\mathrm{x}^{(i)}) \frac{q_\phi(\mathrm{z}|\mathrm{x}^{(i)})}{q_\phi(\mathrm{z}|\mathrm{x}^{(i)})}) dz \\
&= \int q_\phi(\mathrm{z}|\mathrm{x}^{(i)}) \log \frac{p_\theta(\mathrm{x}^{(i)}, \mathrm{z})}{q(\mathrm{z}|\mathrm{x}^{(i)})} + \int q_\phi(\mathrm{z}|\mathrm{x}^{(i)}) \log \frac{q_\phi(\mathrm{z}|\mathrm{x}^{(i)})}{p_\theta(\mathrm{z}|\mathrm{x}^{(i)})} \\
&= \mathbb{E}_{q_\phi(\mathrm{z}|\mathrm{x})}(\frac{\log p_\theta(\mathrm{x}^{(i)}, \mathrm{z})}{q_\phi(\mathrm{z}|\mathrm{x}^{(i)})}) + KL(q_\phi(\mathrm{z}|\mathrm{x}^{(i)})||p_\theta(\mathrm{z}|\mathrm{x}^{(i)})) \qquad (2.7)
\end{aligned}$$

$$(2.8)$$

Since the KL divergence term is non-negative, the log likelihood for each term is lower bounded by the first term in Eq. 2.8. For ease of terminology let the first term be represented as $\mathcal{L}(\theta, \phi; \mathrm{x}^{(i)})$. This term is popularly referred to the Evidence Lower Bound or ELBO. From above, the ELBO can be seen as a lower bound on

the true data likelihood. By maximizing the ELBO, we can then get as close to the true density as possible. Thus we have:

$$\begin{aligned}
\log p_\theta(\mathrm{x}^{(i)}) &>= \mathcal{L}(\theta, \phi; \mathrm{x}^{(i)}) \\
&= \mathbb{E}_{q_\phi(\mathrm{z}|\mathrm{x})}(\log p_\theta(\mathrm{x}^{(i)}, \mathrm{z}) - \log q_\phi(\mathrm{z}|\mathrm{x}^{(i)})) \\
&= -KL(q_\phi(\mathrm{z}|\mathbf{x}^{(i)})||p_\theta(\mathrm{z})) + \mathbb{E}_{q_\phi(\mathrm{z}|\mathrm{x})}[\log(p_\theta(\mathrm{x}^{(i)}|\mathrm{z}))] \qquad (2.9)
\end{aligned}$$

The second equality above is obtained by expanding the joint probability term into the prior and the marginal likelihood and rearranging the terms. The goal of the AEVB algorithm is to estimate the lower bound by using a stochastic gradient procedure. Under some mild conditions outlined in [KW13] for a chosen approximate posterior $q_\phi(\mathrm{z}|\mathrm{x})$ we can reparameterize the random variable $\tilde{z} \sim q_\phi(\mathrm{z}|\mathrm{x})$ using a differentiable transformation $g_\phi(\epsilon, \mathrm{x})$ of an (auxiliary) noise variable, i.e. $\tilde{z} = g_\phi(\epsilon, \mathrm{x})$ with $\epsilon \sim p(\epsilon)$. For the case of VAEs, the prior distribution commonly used is the gaussian distribution; q_ϕ and p_θ are modeled as deep neural networks. In the case of gaussian distribution, the transformation g_ϕ can be easily obtained directly: $\mathrm{z} \sim p(\mathrm{z}|\mathrm{x}) = \mathcal{N}(\mu, \sigma^2)$. Then, a valid reparameterization is $\mathrm{z} = \mu + \sigma\epsilon$, where $\epsilon \sim \mathcal{N}(0, 1)$. This reparameterization enables one to backpropagate through the stochastic sampling process, which is not possible otherwise.

Furthermore, using the form of gaussian distribution, the ELBO term $\mathcal{L}(\theta, \phi; \mathrm{x}^{(i)})$ can be estimated analytically as a function of the parameters (θ, ϕ), which can be optimized using standard backpropagation. The similarity to autoencoder can be more clearly observed by noting that in Eq. 2.9, when the decoder is implemented

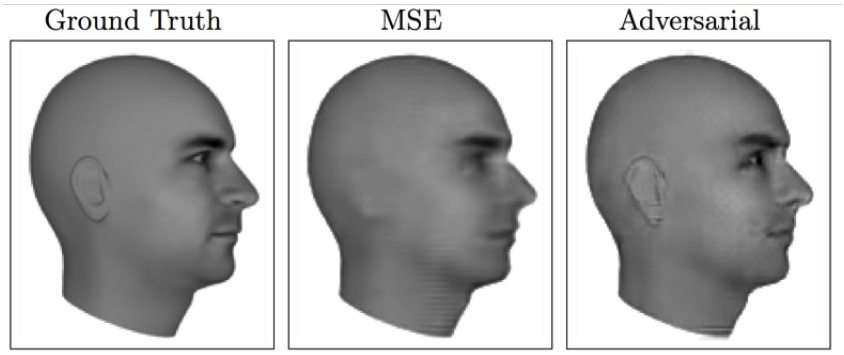


Figure 2.8: Video frame prediction task using different loss functions compared to the ground truth frame. Traditional VAEs typically result in slightly blurry outputs compared to adversarial generative models such as GANs. Figure from [LKC15].

as a Gaussian MLP, the second term reduces to a pixelwise mean square error. The KL term then provides an implicit regularization that prevents the posterior from collapsing by ensuring that it does not deviate too far from the prior distribution.

While both VAEs and GANs provide an ability to learn a underlying data distribution from large datasets, GANs have been more popular especially for image based tasks such as the ones considered in this dissertation. The reason is the quality of the images produced by GANs are superior to the ones produced by VAEs. This can be alluded to the respective loss functions. Since VAEs use a MSE loss, the result is an averaged version of the output signal whereas the adversarial loss employed by GANs picks the most plausible explanation for the input resulting in a sharp output. This is illustrated in Figure 2.8 by an example from Lotter *et al.* [LKC15], where a model is trained to predict the next frame of a given video sequence using different loss functions. It is clear that the adversarial loss yields a much sharper

output compared to the MSE loss which is used in VAE-based approaches. In this dissertation, we employ GANs as generative model due to their ease of training within the context deep neural networks. However, the methods described are not limited to the type of generative models and usage of VAEs will be a subject of future work.

Chapter 3. Triplet Probabilistic Embedding for Face Analysis

3.1 Introduction

[1] Recently, with the advent of curated face datasets like Labeled faces in the Wild (LFW) [HRBLM07] and advances in learning algorithms like deep neural nets, there is more hope that the unconstrained face verification problem can be solved. A face verification algorithm compares two given templates that are typically not seen during training. Research in face verification has progressed well over the past few years, resulting in the saturation of performance on the LFW dataset, yet the problem of unconstrained face verification remains a challenge. This is evident by the performance of traditional algorithms in the publicly available IJB datasets ([KTB+15], [CSPC15]) that were released recently. Moreover, despite the superb performance of CNN-based approaches compared to traditional methods, a drawback of such methods is the long training time needed. In this chapter, we present a Deep CNN (DCNN) architecture that ensures faster training, and investigate how much the performance can be improved if we are provided domain specific data. Specifically, our contributions are as follows:

- We propose a deep network architecture and a training scheme that ensures

[1] The material presented in this chapter is part of an externally published work [SACC16]

faster training time.

- We formulate a triplet probability embedding learning method to improve the performance of deep features for face verification and subject clustering.

During training, we use a publicly available face dataset to train our deep architecture. Each image is pre-processed and aligned to a canonical view before passing it to our deep network whose features are used as the representation of the image. In the case of IJB-A dataset, the data is divided into 10 splits, each split containing a training set and test set. Hence, to further improve performance, we learn the proposed triplet probability embedding using the training set provided with each split over the features extracted from our DCNN model. During the deployment phase, given a face template, we extract the deep features using the raw CNN model after some automatic pre-processing steps such as face detection and fiducial extraction. The deep features are projected onto a low-dimensional space using the embedding matrix learned during training (note that the projection involves only matrix multiplication). We use the 128-dimensional feature as the final representation of the given face template.

This chapter is organized as follows: Section 3.2 places our work among the recently proposed approaches for face verification. Section 3.3 details the network architecture and the training scheme. The triplet probabilistic embedding learning method is described in Section 3.4 followed by results on the IJB-A and CFP datasets and a brief discussion in Section 6.4. In Section 3.6, we demonstrate the ability of the proposed method to cluster a media collection from the LFW and IJB-A datasets.

3.2 Related Work

The proposed approach broadly consists of two components: the deep network used as a feature extractor and the learning procedure that projects the input features onto a discriminative low-dimensional space. In the past few years, there have been numerous works in using deep features for tasks related to face verification. The DeepFace [TYRW14] approach uses a carefully crafted 3D alignment procedure to preprocess face images and feeds them to a deep network (with 120M parameters) that is trained with a large training set. A kernel classifier is then trained on the resulting features to make the final verification decision. More recently, Facenet [SKP15a] uses the inception architecure and a large private dataset to train a deep network using a triplet distance loss function. The training time for this network is of the order of few weeks. Since the release of the IJB-A dataset [KTB+15], there have been several works that have published verification results for this dataset. Previous approaches presented in [WOJ15] and [PVZ15] train deep networks using the CASIA-WebFace dataset [YLLL14] and the VGG-Face dataset respectively, requiring substantial training time. This chapter proposes a network architecture and a training scheme that needs shorter training time and a small query time.

The idea of learning a compact and discriminative representation has been around for decades. Weinberger *et al.* [WBS05] used a Semi Definite Programming (SDP)-based formulation to learn a metric satisfying pairwise and triplet distance constraints in a large margin framework. More recently, this idea has been successfully applied to face verification by integrating the loss function within the deep

network architecture ([SKP15a], [PVZ15]). Joint Bayesian metric learning has been another popular metric used for face verification ([SPVZ13], [CPC15]). These methods either require a large dataset for convergence or learn a metric directly therefore not amenable to subsequent operations like discriminative clustering or hashing. Classic methods like t-SNE [VdMH08], t-STE [VDMW12] and Crowd Kernel Learning (CKL) [TLB+11] perform extremely well when used to visualize or cluster a given data collection. They either operate on the data matrix directly or the distance matrix generated from the data by generating a large set of pairwise or triplet constraints. The objective of the optimization algorithm is to minimize the violations in the constraint set. While these methods perform very well on a given set of data points, they do not generalize to out-of-sample data. In the current work, we aim to generalize such formulations, to a more traditional classification setting, where domain specific training and testing data is provided. We formulate an optimization problem based on triplet probabilities that performs dimensionality reduction aside from improving the discriminative ability of the test data. The embedding scheme described in this chapter is a more general framework that can be applied to any setting where labeled training data is available.

3.3 Network Architecture

This section details the architecture and training algorithm for the deep network used in our work. Our architecture consists of 7 convolutional layers with varying kernel sizes. The initial layers have a larger size rapidly subsampling the

image and reducing the parameters while the later layers consist of small filter sizes, which has proved to be very useful in face recognition tasks ([PVZ15], [YLLL14]). Furthermore, we use the Parametric Rectifier Linear units (PReLUs) instead of ReLU, since they allow a negative value for the output based on a learned threshold and have been shown to improve the convergence rate [HZRS15].

Layer	Kernel Size/Stride	#params
conv1	11x11/4	35K
pool1	3x3/2	
conv2	5x5/2	614K
pool2	3x3/2	
conv3	3x3/2	885K
conv4	3x3/2	1.3M
conv5	3x3/1	2.3M
conv6	3x3/1	2.3M
conv7	3x3/1	2.3M
pool7	6x6/2	
fc6	1024	18.8M
fc7	512	524K
fc8	10548	10.8M
Softmax Loss		Total: 39.8M

Table 3.1: Deep Network architecture details

The top three convolutional layers (conv1-conv3) are initialized with the weights from the AlexNet model [KSH12a] trained on the ImageNet challenge dataset. Several recent works ([YCBL14], [LW15]) have empirically shown that this transfer of knowledge across different networks, albeit for a different objective, improves performance and more significantly reduces the need to train over a large number of

iterations.

The compared methods either learn their deep models from scratch ([PVZ15], [YRC+16]) or finetune only the last layer of fully pre-trained models. The former approach results in large training time and the latter approach does not generalize well to the task at hand (face verification) and hence results in a sub-optimal performance. In the current work, even though we use a pre-trained model (AlexNet) to initialize the proposed deep network, we do so only for the first three convolutional layers, since they retain more generic information ([YCBL14]). Subsequent layers learn representations which are more specific to the task at hand. Thus, to learn more domain specific information, we add 4 convolutional layers each consisting of 512 kernels of size 3×3. The layers conv4-conv7 do not downsample the input thereby learning more complex higher dimensional representations. This hybrid architecture proves to be extremely effective as our raw CNN representation outperforms some very deep CNN models on the IJB-A dataset (Table 2 in Results). In addition, we achieve that performance by training our deep network on the relatively smaller CASIA-WebFace dataset.

The architecture of our network is shown in Figure 3.1. Layers conv4-conv7 and the fully connected layers *fc6-fc8* are initialized from scratch using random Gaussian distributions. PReLU activation functions are added between each layer. Since the network is used as a feature extractor, the last layer *fc8* is removed during deployment, thus reducing the number of parameters to 29M. The inputs to the network are 227x227x3 RGB images. When the network is deployed, the features are extracted from the *fc7* layers resulting in a dimensionality of 512. The network

is trained using the Softmax loss function for multiclass classification using the Caffe deep learning platform [JSD+14].

3.4 Learning a Discriminative Embedding

In this section, we describe our algorithm for learning a low-dimensional embedding such that the resulting projections are more discriminative. Aside from an improved performance, this embedding provides significant advantages in terms of memory and enables post-processing operations like visualization and clustering.

Consider a triplet $t := (\mathrm{v}_i, \mathrm{v}_j, \mathrm{v}_k)$, where v_i (anchor) and v_j (positive) are from the same class, but v_k (negative) belongs to a different class. Consider a function $S_W : \mathbb{R}^N \times \mathbb{R}^N \mapsto \mathbb{R}$ that is parameterized by the matrix $\mathrm{W} \in \mathbb{R}^{n \times N}$, that measures the similarity between two vectors $\mathrm{v}_i, \mathrm{v}_j \in \mathbb{R}^N$. Ideally, for all triplets t that exist in the training set, we would like the following constraint to be satisfied:

$$S_{\mathrm{W}}(\mathrm{v}_i, \mathrm{v}_j) > S_{\mathrm{W}}(\mathrm{v}_i, \mathrm{v}_k) \tag{3.1}$$

Thus, the probability of a given triplet t satisfying (3.1) can be written as:

$$p_{ijk} = \frac{e^{S_{\mathrm{W}}(\mathrm{v}_i, \mathrm{v}_j)}}{e^{S_{\mathrm{W}}(\mathrm{v}_i, \mathrm{v}_j)} + e^{S_{\mathrm{W}}(\mathrm{v}_i, \mathrm{v}_k)}} \tag{3.2}$$

The specific form of the similarity function is given as: $S_{\mathrm{W}}(\mathrm{v}_i, \mathrm{v}_j) = (\mathrm{W}\mathrm{v}_i)^T \cdot (\mathrm{W}\mathrm{v}_j)$. In our case, v_i and v_j are deep features normalized to unit length. To learn the embedding W from a given set of triplets $\mathbb{T}$, we solve the following optimization:

$$\underset{\mathrm{W}}{\operatorname{argmin}} \sum_{(\mathrm{v}_i,\mathrm{v}_j,\mathrm{v}_k)\in\mathbb{T}} -\log(p_{ijk}) \tag{3.3}$$

(3.3) can be interpreted as maximizing the likelihood (3.1) or minimizing the negative log-likelihood (NLL) over the triplet set $\mathbb{T}$. In practice, the above problem is solved in a Large-Margin framework using Stochastic Gradient Descent (SGD) and the triplets are sampled online. The gradient update for the W is given as:

$$\begin{aligned}\mathrm{W}_{\tau+1} = \mathrm{W}_\tau - \eta * \mathrm{W}_\tau * (1 - p_{ijk}) * (\mathrm{v}_i(\mathrm{v}_j - \mathrm{v}_k)^T \\ +(\mathrm{v}_j - \boldsymbol{v}_k)\mathrm{v}_i^T)\end{aligned} \tag{3.4}$$

where W_τ is the estimate at iteration τ, $\mathrm{W}_{\tau+1}$ is the updated estimate, $(\mathrm{v}_i, \mathrm{v}_j, \mathrm{v}_k)$ is the triplet sampled at the current iteration and η is the learning rate.

By choosing the dimension of W as $n \times N$ with $n < N$, we achieve dimensionality reduction in addition to better performance. For our work, we fix $n = 128$ based on cross validation and $N = 512$ is the dimensionality of our deep features. W is initialized with the first n principal components of the training data. At each iteration, a random anchor and a random positive data point are chosen. To choose the negative, we perform hard negative mining, ie. we choose the data point that has the least likelihood (3.2) among the randomly chosen 2000 negative instances at each iteration.

Since we compute the embedding matrix W by optimizing over triplet probabilities, we call this method Triplet Probability Embedding (TPE). The technique closest to the one presented in this section, which is used in recent works

([SKP15a], [PVZ15]) computes the embedding W based on satisfying a hinge loss constraint:

$$\underset{\mathrm{W}}{\operatorname{argmin}} \sum_{(\mathrm{v}_i,\mathrm{v}_j,\mathrm{v}_k)\in\mathbb{T}} \max\{0, \alpha + (\mathrm{v}_i - \mathrm{v}_j)^T \mathrm{W}^{\mathrm{T}}\mathrm{W}(\mathrm{v}_i - \mathrm{v}_j) - (\mathrm{v}_i - \mathrm{v}_k)^T \mathrm{W}^{\mathrm{T}}\mathrm{W}(\mathrm{v}_i - \mathrm{v}_k)\} \tag{3.5}$$

α acts a margin parameter for the loss function. To be consistent with the terminology used in this dissertation, we call it Triplet Distance Embedding (TDE). To appreciate the difference between the two approaches, let us look at the gradient update step for (3.5):

$$\mathrm{W}_{\tau+1} = \mathrm{W}_\tau - \eta * \mathrm{W}_\tau * ((\mathrm{v}_i - \mathrm{v}_j)(\mathrm{v}_i - \mathrm{v}_j)^T - (\mathrm{v}_i - \mathrm{v}_k)(\mathrm{v}_i - \mathrm{v}_k)^T) \tag{3.6}$$

Figure 3.1 shows the case where the gradient update for the TDE method (3.6) occurs. If the value of α is not appropriately chosen, a triplet is considered good even if the positive and negative are very close to one another. But under the proposed formulation, both cases referred to in Figure 3.1 will update the gradient but their contribution to the gradient will be modulated by the probability with which they violate the constraint in (3.1). This modulation factor is specified by the $(1 - p_{ijk})$ term in the gradient update for TPE in (5.4) implying that if the likelihood of a sampled triplet satisfying (3.1) is high, then the gradient update is given a lower weight and vice-versa. Thus, in our method , the margin parameter (α) is automatically set based on the likelihood.

To compare the relative performances of the raw features before projection,

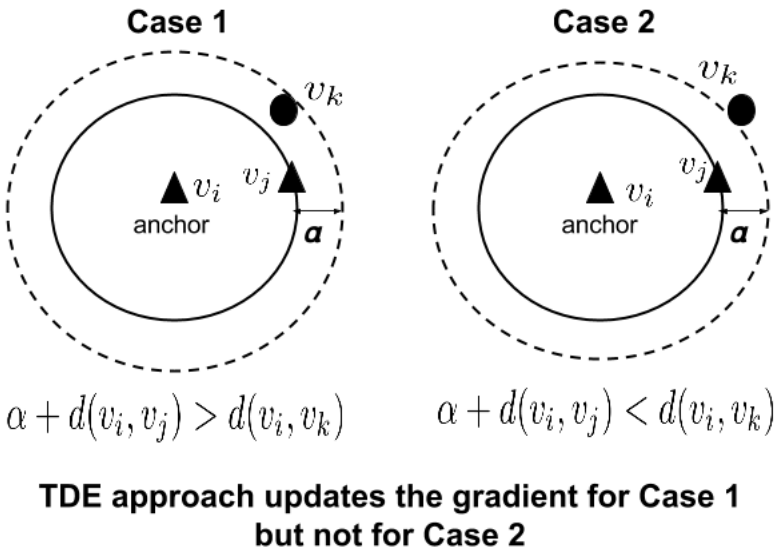


Figure 3.1: Gradient update scenarios for the TDE method (3.5). The notation is explained in the text

with TDE and with TPE (proposed method), we plot the traditional ROC curve (TAR (vs) FAR) for split 1 of the IJB-A verify protocol for the three methods in Figure 3.2. The Equal Error Rate (EER) metric, which is a popular measure to compare classification systems is specified for each method. The performance improvement due to TPE is significant, especially at regions of FAR= $\{10^{-4}, 10^{-3}\}$. We observed a similar behaviour for all the ten splits of the IJB-A dataset.

3.5 Experimental setup and Results

In this section we evaluate the proposed method on two challenging datasets:

1. **IARPA Janus Benchmark-A (IJB-A)** [KTB+15]: This dataset contains 500 subjects with a total of 25,813 images (5,399 still images and 20,414 video

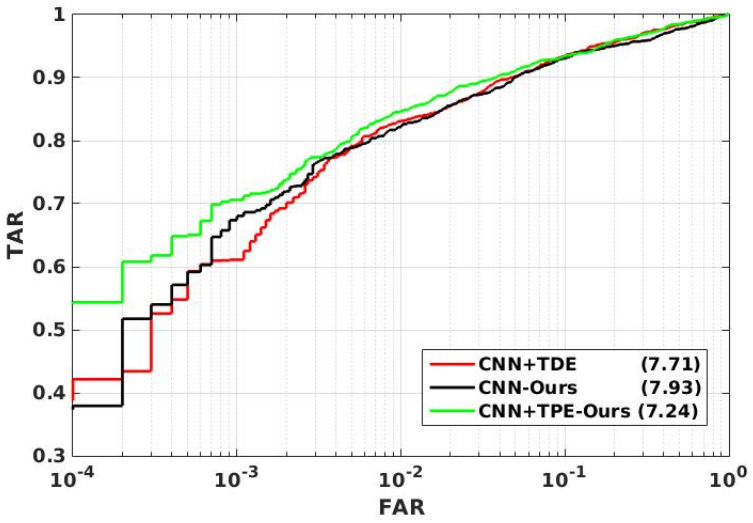


Figure 3.2: Performance improvement on IJB-A split 1: FAR (vs) TAR plot. EER values are specified in brackets.

frames sampled with a rate of 1 in 60). The faces in the IJB-A dataset contain extreme poses and illuminations, much harder than LFW [HRBLM07]. An additional challenge of the IJB-A verification protocol is that the template comparisons include image to image, image to set and set to set comparisons. In the experiments, if a given test template of the IJB-A data we perform two kinds of pooling to produce its final representation:

- Average pooling (CNN_{ave}): The deep features of the images and/or frames present in the template are combined by taking a componentwise average to produce one feature vector. Thus each feature equally contributes to the final representation.
- Media pooling (CNN_{media}): The deep features are combined keeping in mind the media source they come from. The metadata provided with IJB-A gives us

(a) Frontal-Frontal

(b) Frontal-Profile

Figure 3.3: Sample comparison pairs from the CFP dataset

Figure 3.4: Images from the IJB-A dataset

the *media id* for each item of the template. Thus to get the final feature vector, we first take an intra-media average and then combine these by taking the inter-media average. Thus each feature's contribution to the final representation is weighted based on its source. Some sample images from the IJB-A dataset are shown in Figure 3.4.

2. **Celebrities in Frontal-Profile (CFP)** [SCC+16]: This dataset contains 7000 images of 500 people. The dataset is used for evaluating how face verification approaches handle pose variation. Hence, it consists of 5000 images in frontal view and 2000 images in extreme profile. The data is organized into 10 splits, each containing equal number of frontal-frontal and frontal-profile comparisons. Sample comparison pairs of the CFP dataset are shown in Figure 3.3.

3.5.1 Pre-processing

In the training phase, given an input image, we use the HyperFace method [RPC16] for face detection and fiducial point extraction. The HyperFace detector automatically extracts all the faces from a given image. For the IJB-A dataset, since most images contain more than one face, we use the bounding boxes provided along with the dataset to select the person of interest from the list of automatic detections. We select the detection that has the maximum area overlap with the manually provided bounding box. In the IJB-A dataset, there are few images for which the HyperFace detector cannot find the relevant face. For the missed cases, we crop the face using the bounding box information provided with the dataset and pass it to HyperFace to extract fiducials. We use six fiducial points (eyes and mouth corners) to align the detected image to a canonical view using the similarity transform. For the CFP dataset, since the six keypoints cannot be computed for profile faces we only use three keypoints on one side of the face for aligning them.

3.5.2 Parameters and training times

The training of the proposed deep architecture is done using SGD with momentum, which is set to 0.9 and the learning rate is set to 1e-3 and decreased uniformly by a factor of 10 every 50K iterations. The weight decay is set to 5e-4 for all layers. The training batch size is set to 256. The training time for our deep network is 24 hours on a single NVIDIA TitanX GPU. For the IJB-A dataset, we use the training data provided with each split to obtain the triplet embedding which

takes 3 minutes per split. This is the only additional splitwise processing that is done by the proposed approach. During deployment, the average enrollment time per image after pre-processing, including alignment and feature extraction is 8ms.

3.5.3 Evaluation Pipeline

Given an image, we pre-process it as described in Section 3.5.1. The deep features are computed as an average of the image and its flip. Given two face images to compare, we compute their cosine similarity score. More specifically, for the IJB-A dataset, given a template containing multiple faces, we *flatten* the template features by average pooling or media pooling to obtain a vector representation. For each split, we learn the TPE projection using the provided training data. The final representation is obtained as: $y = \mathbf{W}x$, where x is the deep feature and $\mathbf{W}$ is the TPE projection matrix. Given two templates for comparison, we compute the cosine similarity score using the projected 128-d representations.

3.5.4 Evaluation Metrics

We report two types of results for the IJB-A dataset: Verification and Identification. For the verification protocol, we report the False Non-Match Rate (FNMR) values at several False Match Rates (FMR). For the identification results, we report open set and closed set metrics. The True Positive Identification Rate (TPIR) quantifies the fraction of subjects that are classified correctly among the ones that exist in probe but not in gallery. For the closed set metrics, we report the CMC numbers at different values of False Positive Identification Rates (FPIRs) and Ranks. More

	IJB-A Verification (FNMR@FMR)			IJB-A Identification			
Method	0.001	0.01	0.1	FPIR=0.01	FPIR=0.1	Rank=1	Rank=10
GOTS [KTB+15]	0.8 (0.008)	0.59 (0.014)	0.37 (0.023)	0.047 (0.02)	0.235 (0.03)	0.443 (0.02)	-
VGG-Face [PVZ15]	0.396 (0.06)	0.195 (0.03)	0.063(0.01)	0.46 (0.07)	0.67 (0.03)	0.913 (0.01)	**0.981 (0.005)**
Masi *et al.* [MTL+16]	0.275	0.118	-	-	-	0.902	0.968
NAN [YRC+16]	0.215 (0.03)	0.103 (0.01)	0.041 (0.005)	-	-	-	-
CNN_{ave} (Ours)	0.287 (0.05)	0.146 (0.01)	0.051 (0.006)	0.626 (0.06)	0.795 (0.02)	0.90 (0.01)	0.974 (0.004)
CNN_{media} (Ours)	0.234 (0.02)	0.129 (0.01)	0.048 (0.005)	0.67 (0.05)	0.82 (0.013)	0.925 (0.01)	0.978 (0.005)
CNN_{media}+TPE (Ours)	**0.187 (0.02)**	**0.10 (0.01)**	**0.036 (0.005)**	**0.753 (0.03)**	**0.863 (0.014)**	**0.932 (0.01)**	0.977 (0.005)

Table 3.2: Identification and Verification results on the IJB-A dataset. For identification, the scores reported are TPIR values at the indicated points. The results are averages over 10 splits and the standard deviation is given in the brackets for methods which have reported them. '−' implies that the result is not reported for that method. The best results are given in bold.

details on the evaluation metrics for the IJB-A protocol can be found in [KTB+15].

For the CFP dataset, following the protocol set in [SCC+16], we report the Area under the curve (AUC) and Equal Error Rate (EER) values as averages across splits, in addition to the classification accuracy. To obtain the accuracy for each split, we threshold our CNN similarity scores where the threshold is set to the value that provides highest classification accuracy over the training data for each split.

3.5.5 Discussion

Performance on IJB-A

Table 3.2 presents the results for the proposed method compared to the other approaches applied to the IJB-A verification and identification protocols. The compared methods are described below:

- Government-of-the-Shelf (GOTS) [KTB+15] is the baseline performance provided along with the IJB-A dataset.

- Parkhi *et al.* [PVZ15] train a very deep network (22 layers) over the VGG-Face dataset which contains 2.6M images from 2622 subjects.

- The Neural Aggregation network (NAN) [YRC+16] is trained over large amount of videos from the CELEB-1000 dataset [LZLY14] starting from the GoogleNet [SLJ+15] architecture. Furthermore, a separate siamese network is trained for verification experiments starting from the NAN network.

- Masi *et al.* [MTL+16] use a deep CNN based approach that includes a combination of in-plane aligned images, 3D rendered images to augment their performance. The 3D rendered images are also generated during test time per template comparison, which results in a large query time per template comparison.

Compared to these methods, the proposed method trains a single CNN model on the CASIA-WebFace dataset which consists of about 500K images and requires much shorter training time and has a very fast query time (8ms after face detection per image pair). Our raw CNN features after media pooling have the best performance among the compared methods in Table 3.2 in the IJB-A verification and identification protocols with the exception of Rank-10 accuracy where we are very close to the best result. The TPE method provides significant improvement for both the identification and verification tasks as shown in Table 3.2.

Table 3.3 shows results using the method by Crosswhite *et al.* [CBP+16] that

FMR	Ours	Crosswhite *et al.* [CBP+16]
1e-2	0.10 (0.012)	0.061 (0.013)
1e-1	0.036 (0.005)	0.017 (0.007)

Table 3.3: Recent results on the IJB-A verification protocol. Reported as FNMR at FMR

uses the VGG-Face network [PVZ15] descriptors (4096-d) as the raw features. They use the concept of template adaptation [WHT09] to improve the face recognition performance as follows: when pooling multiple faces of a given template, they train a linear SVM with the features of this template as positive and a fixed set of negatives extracted from the training data of the IJB-A splits. Let's denote the pooled template feature and classifier pair as (t, w). Then, at query time when comparing two templates (t_1, w_1) and (t_2, w_2), the similarity score is computed as: $\frac{1}{2}(t_1 \cdot w_2 + t_2 \cdot w_1)$. Even when using a carefully engineered fast linear classifier training algorithm, this procedure increases the run time of the pooling procedure and hence the query time per template comparison significantly. In contrast, our approach requires a matrix multiplication and a vector dot product per comparison. By using a simple neural network architecture, a relatively smaller training dataset and a fast embedding method we have realized a faster and more efficient end-to-end system. By incorporating video data into our approach, the performance could be improved further. We leave this for future work.

	Frontal-Frontal			Frontal-Profile		
Algorithm	Accuracy	EER	AUC	Accuracy	EER	AUC
Sengupta *et al.* [SCC+16]	96.40 (0.69)	3.48 (0.67)	99.43 (0.31)	84.91 (1.82)	14.97 (1.98)	93.00 (1.55)
Human Accuracy	96.24 (0.67)	5.34 (1.79)	98.19 (1.13)	**94.57 (1.10)**	**5.02 (1.07)**	**98.92 (0.46)**
CNN (Ours)	**96.93 (0.61)**	**2.51 (0.81)**	**99.68 (0.16)**	89.17 (2.35)	8.85 (0.99)	97.00 (0.53)

Table 3.4: Results on the CFP dataset [SCC+16]. The numbers are averaged over ten test splits and the numbers in brackets indicate standard deviations of those runs. The best results are given in bold.

Performance on CFP

On the CFP dataset, we set a new state-of-art on both Frontal-Frontal and Frontal-Profile comparisons, the latter by a large margin. More specifically, for the Frontal-Profile case, we manage to reduce the error rate by **40.8%**. It should be noted that for a fair comparison we have used our raw CNN features without performing TPE. This shows that the raw CNN features we learn are effective even at extreme pose variations.

3.6 Clustering Faces

This section illustrates how the proposed TPE method can be used to cluster a given data collection. We perform two clustering experiments:

1. We perform clustering on the entire LFW [HRBLM07] dataset that consists of 13233 images of 5749 subjects. It should be noted that about 4069 subjects have only one image.

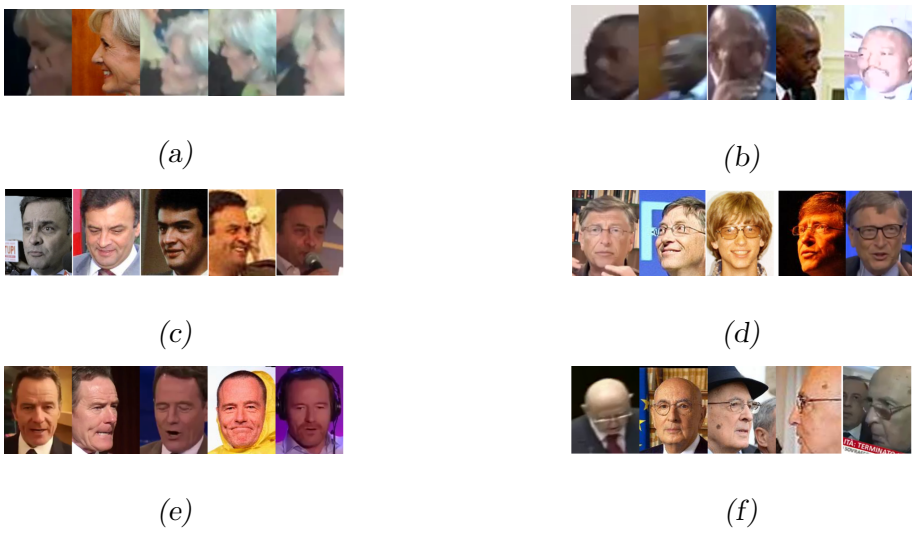


Figure 3.5: Sample clusters output from the Clustering approach discussed in Section 6 for the data from the split 1 of the IJB-A dataset. Top row (a,b) shows robustness to pose and blur; Middle row (c,d) contains clusters that are robust to age; Bottom row (e,f) shows instances that are robust to disguise.

2. We use the IJB-A dataset and cluster the templates corresponding to the query set for each split the IJB-A verify protocol.

For evaluating the clustering results, we use the metrics defined in [OWJ16]. These are summarized below:

- *Pairwise Precision (P_{pair})*: The fraction of pairs of samples within a cluster among all possible pairs which are of the same class , over the total number of same cluster pairs.

- *Pairwise Recall (R_{pair})*: The fraction of pairs of samples within a class among all possible pairs which are placed in the same cluster, over the total number of same-class pairs.

Using these metrics, the F_1-score is computed as:

$$F_1 = \frac{2 * P_{pair} * R_{pair}}{R_{pair} + P_{pair}} \tag{3.7}$$

The simplest way we found to demonstrate the effectiveness of our deep features and the proposed TPE method, is to use the standard MATLAB implementation of the agglomerative clustering algorithm with the average linkage metric. We use the cosine similarity as as our basic clustering metric. The simple clustering algorithm that we have used here has computational complexity of $O(N^2)$. In its current form, this does not scale to large datasets with millions of images. We believe this is not an insurmountable limitation and we are currently working on a more efficient and scalable (yet approximate) version of this algorithm.

3.6.1 Clustering LFW

The images in the LFW dataset are pre-processed as described in Section 3.5.1. For each image and its flip, the deep features are extracted using the proposed architecture and their component-wise average normalized to unit L_2 norm is used as the final representation. We run the clustering algorithm over the entire data in a single shot. The clustering algorithm takes as input a cut-off parameter which acts as a distance threshold (below which any two clusters will not be merged). In our experiments, we vary this cut-off parameter over a small range and evaluate the resulting clustering using the F_1-score. We pick the result that yields the best F_1-score. Table 3.5 shows the result of our approach and compares it to a recently released clustering approach based on approximate Rank-order clustering [OWJ16].

Method	F_1-score	Clusters
[OWJ16]	0.87	6508
CNN (Ours)	**0.955**	5351

Table 3.5: F_1-score for comparison of the two clustering schemes on the LFW dataset. The ground truth cluster number is 5749.

Method	F_1-score	Clusters	After Pruning
CNN_{media}	0.79	293 (22)	173
CNN_{media}+TPE	**0.843**	258(17)	167

Table 3.6: Clustering metrics over the IJB-A 1:1 protocol. The standard deviation is indicated in brackets. The ground truth subjects per each split is 167.

It should be noted that, in the case of [OWJ16], the clustering result is chosen by varying the number of clusters and picking the one with the best F_1-score. In our approach, we vary the cut-off threshold which is the property of the deep features and hence is a more intuitive parameter to tune. We see from Table 3.5 that aside from better performance, the total cluster estimate is closer to the ground truth value of 5749 than [OWJ16].

3.6.2 Clustering IJB-A

The IJB-A dataset is processed as described in Section 3.5.1. In this section, we aim to cluster the query templates provided with each split for the verify protocol. We report the results of two experiments: with the raw CNN features (CNN_{media})

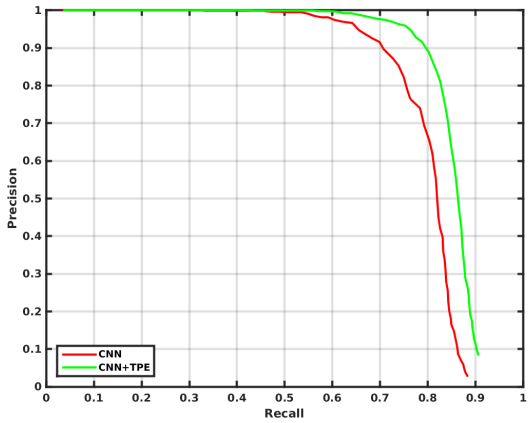


Figure 3.6: Precision-Recall curve plotted over cut-off threshold varied from 0 to 1.

and with the projected CNN features, where the projection matrix is learned through the proposed TPE method (CNN$_{media}$+TPE). The cut-off threshold required for our clustering algorithm is learned automatically based on the training data, i.e. we choose the threshold that gives the maximum F_1-score over the training data. The scores reported in Table 3.6 are average values over ten splits. As expected, the TPE method improves the clustering performance of our raw features. The subject estimate is the number of clusters produced as a direct result of our clustering algorithm. The pruned estimate is obtained by ignoring the clusters which have less than 3 images.

For a more complete evaluation of our performance over varying threshold values, we plot the Precision-Recall (PR) curve for the IJB-A clustering experiment in Figure 3.6. As can be observed, the PR curve for clustering the IJB-A data using embedded features exhibits a better performance at all operating points. This is a more transparent evaluation than reporting only the F_1-score since the latter effectively fixes the operating point but the PR curve reveals the performance at all

operating points.

3.7 Conclusion

In this chapter, we proposed a deep CNN-based approach coupled with a low-dimensional discriminative embedding learned using triplet probability constraints in a large margin fashion. The proposed pipeline enables a faster training time and improves face verification performance especially at low FMRs. We demonstrated the effectiveness of the proposed method on two challenging datasets: IJB-A and CFP and achieved performance competitive with the state of the art while using a deep model which is more compact and trained using a moderately sized dataset. We demonstrated the robustness of our features using a simple clustering algorithm on the LFW and IJB-A datasets. Avenues for future work include extension of the proposed approach to video data. One promising direction is as follows: Given frames of a video, the sample covariance matrix can be reliably computed , which can be considered to be a point on the Grassmann manifold. The TPE approach can then be extended to operate on arbitrary data spaces such as Riemannian manifolds. The proposed clustering scheme can be scaled in a distributed manner to handle large scale scenarios such as large impostor sets of the order of millions.

Chapter 4. Perturbation analysis of Segmentation Networks

4.1 Introduction

[1] Convolutional Neural Networks (CNNs) have achieved state of the art results on several computer vision benchmarks such as ILSVRC [RDS+15] and PASCAL VOC [EEVG+15] over the past few years. Despite their overwhelming success, recent results have highlighted that they can be sensitive to small adversarial noise in the input [GSS14a] or can be easily fooled using structured noise patterns [NYC15]. To understand how a CNN can learn complex and meaningful representations but at the same time be easily fooled by simple and imperceptible perturbations still remains an open research problem. The work of Goodfellow *et. al.* [GSS14a] and Szegedy *et al.* [SZS+13a] among others, bring out the intriguing properties of neural networks by introducing perturbations in either the hidden layer weights or the input image. While these approaches have focused on understanding the effect of adversarial noise in deep networks, in this chapter we present an intriguing observation: input perturbations can enable a CNN to correct its mistakes. We find that these perturbations exploit the local neighborhood information from network's prediction which in turn results in contextually smooth predictions.

[1] The material presented in this chapter is part of an externally published work [SJNL17]

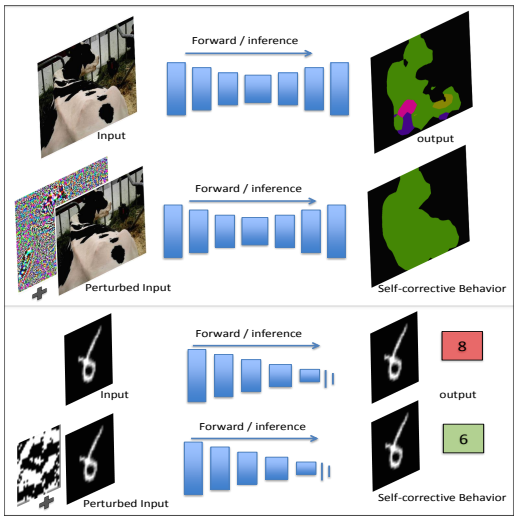


Figure 4.1: Self-corrective behavior due to Guided Perturbations (GP) for segmentation and classification tasks.

In almost all the CNN-based approaches, the output is obtained using a single forward pass during the prediction time. In the proposed approach, we use the prediction made by the network during the forward pass to generate perturbations at the input. Specifically, we backpropagate the gradient of the prediction error all the way to the input. We would like to emphasize that the error gradients are generated purely based on the network's prediction without any knowledge of ground truth. We *perturb* the input image by adding to it a scaled version of the gradient signal. This is fed back to the network again for prediction. Figure 4.1 shows an example of the self-corrective behavior of the generated perturbations for segmentation and classification tasks. This example shows that these perturbations of the input image could be viewed as a form of structured distortion that is added to the input such that the context gets amplified in each pixel's neighborhood which enables the network to correct its own mistakes. The proposed approach is simple and easy to implement and does not require retraining or modification in network's

architecture.

Existing approaches to improve performance on segmentation and classification tasks have been geared towards novelties in network architecture or using large amount of training data or both. While these are valid ways to improve the networks performance, the proposed approach highlights an inherent behavior of CNNs that can be used to improve their prediction without requiring additional learning or training data. We would like to note here that while the behavior of Guided Perturbations (GP) is similar to Conditional Random Fields (CRF) based approaches, the difference in our case is that there is no explicit modeling of context or neighborhood interactions. Since our approach is network independent, this doesn't preclude networks which model context explicitly and we show improvements in such networks too.

To the best of our knowledge, this is the first approach to show the existence of a self-corrective behavior in CNNs and use of such behavior for improvement in performance on segmentation and classification tasks. To summarize, the major contributions of the proposed approach are:

- We present a novel and intriguing observation: there exist structured perturbations which when used to perturb the input leads to a corrective behavior in CNNs.
- We propose a generalized framework to improve the performance of any pretrained CNN model that is architecture independent and requires no learning assuming the network is trained end-to-end.

4.2 Background

In recent years, there have been several approaches that attempt to analyze the behavior of CNNs for classification problems. Mahendran *et al* [MV15] proposed an approach to invert the function learned by the CNN in order to generate as faithful a reconstruction of the input as possible. This is performed by minimizing a regularized energy function that approximates the representation function that is learned by the deep network. Another interesting work in this direction is the fooling images work of Nguyen *et al* [NYC15] that is further extended by Yosinski *et al* [YCN+15]. The main objective in both the approaches is to synthesize images to confuse CNN by maximizing the activation of individual neurons from different layers of a deep network. This leads to interesting results such as images that look like random noise but which the CNN classifies into different classes with high confidence. The approaches that are closer in spirit to our proposed approach are: Szegedy *et al* [SZS+13a] and Goodfellow *et al* [GSS14a]. Their study shows that there exist a lot of adversarial examples which are the result of minor pertubations of the input that causes the CNN to misclassify input images on classification tasks; these examples can be generated by adding a fraction of the gradient that is generated by wiggling the classifier output in the direction of the target class.

One of the applications that this chapter focuses on is semantic segmentation. Significant research has gone into understanding the expressive ability of CNNs for such problems. Recent methods for image segmentation such as Fully Convolutional Networks (FCN) by Long *et al* [SLD16] have provided an easy framework that casts

the image segmentation problem as a pixelwise label classification problem. The major difference in their work was the image level output generation and backpropogation which was made possible by the work of Zeiler *et al* [ZKTF10]. This image level back propogation provides a simple way to learn a discriminative representation of classes at the pixel level. Several recent approaches such as *CRFasRNN* by Zheng *et al* [ZJRP+15], DeepLab by Chen *et al* [CPK+16] and GCRF by Vemulapalli *et al* [VTLC16] have improved the FCN framework by explicitly modeling context. CRFasRNN casts the CRF iterations, which has been traditionally used as a post processing function in image segmentation problem to ensure label compatibility, as a Recurrent Neural Network. They formulate the steps required to perform a mean field iteration in a CRF including message passing and learning a label compatibility transform as a layer in a CNN, which is unrolled in time over T iterations. The unary potentials are computed using the FCN-8s network which is then refined using the RNN strucure. By casting this as a CNN layer they perform end-to-end training. More recently, Yu *et al* [YK15] propose to train a multiscale context aggregation module on top of a modified FCN-8s network. This context module improves the performance of the base network on its own or in combination with CRF based approaches.

In this chapter, we describe an interesting property of deep networks about how it can change its predictions for the better using minor perturbations of the input. Furthermore, we provide useful applications of our approach by showing how it can be used to improve prediction performance on challenging computer vision tasks.

4.3 Our Approach

In this section, we describe our approach to generate guided perturbations by using the gradient information obtained from the network's output. We perform experiments to study the different aspects of these perturbations and how they affect the network representations. Since our approach to generate guided perturbation is different for segmentation and classification tasks, we discuss them separately.

4.3.1 Semantic Segmentation

Figure 4.2 illustrates our approach for semantic segmentation task. Given an input image we perform a forward pass to compute the output - which is usually the

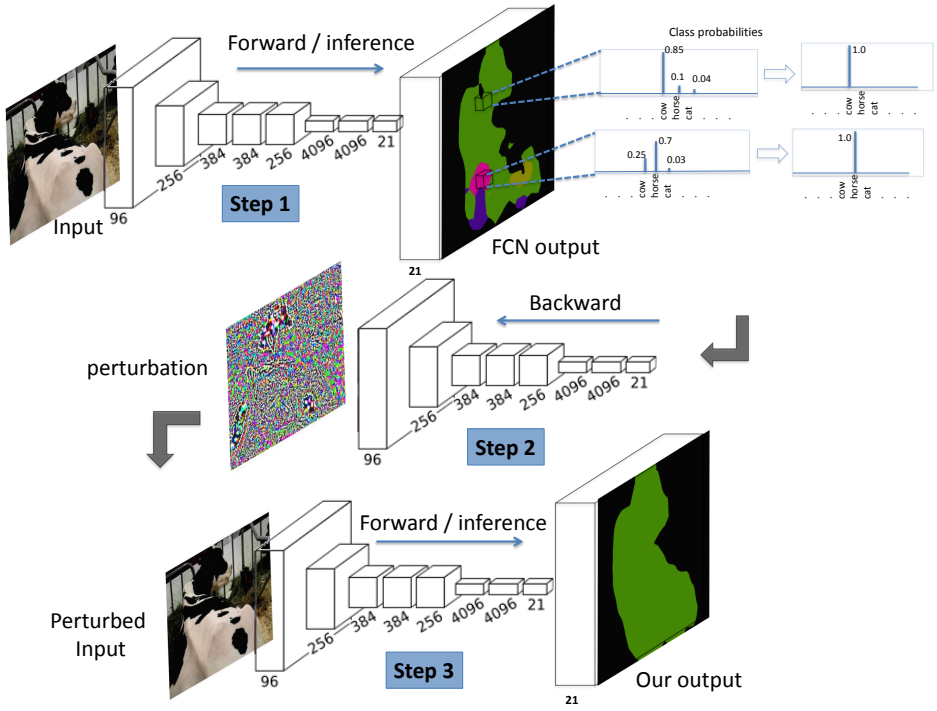


Figure 4.2: Processing pipeline for the proposed approach for semantic segmentation

output of a softmax function that gives a class probability vector for each pixel. The prediction output is then binarized by setting the probability of the most confident class to one and the others to zero. This is done for each pixel and the error gradient is computed at the softmax layer by setting this modified output as ground truth. Let $\mathrm{X} \in \mathbb{R}^{M \times N \times C_{in}}$ represent the input image to the deep network, $\mathrm{Y} \in \mathbb{R}^{M \times N \times C_{out}}$ represent ground truth labeling, where C_{in} is the number of input channels, C_{out} is the number of classes and $M \times N$ is the dimensionality of the input image. Let θ represent the parameters of the network and $\mathcal{J}(\theta, \mathrm{X}, \mathrm{Y})$ represent the loss function that is optimized during training. During prediction time, let Y_{pred} be the predicted labeling. In order to generate an error gradient for backpropagation, we create a pseudo ground truth labeling Y_{pseudo} by modifying Y_{pred} as follows: We initialize Y_{pseudo} with Y_{pred}. Let the k^{th} component of Y_{pseudo} be represented as $\mathrm{y_k} = [y_{k_1}, .., y_{k_{C_{out}}}]$, which is a C_{out}-dimensional score vector. We modify $\mathrm{y_k}$ to be a 1-hot encoded vector with the maximally confident class set to 1 and others to zero. Then, the error gradient signal is computed based on the loss function $\mathcal{J}(\theta, \mathrm{X}, \mathrm{Y}_{pseudo})$ and backpropagated through the network up to the input. Let the backpropagated error gradient signal at the input be represented as: $\nabla_X \mathcal{J}(\theta, \mathrm{X}, \mathrm{Y}_{pseudo})$. The perturbed input in then generated as follows:

$$\mathrm{X}_{per} = \mathrm{X} + \epsilon sign(\nabla_X \mathcal{J}(\theta, \mathrm{X}, \mathrm{Y}_{pseudo})),\ \epsilon > 0 \tag{4.1}$$

where ϵ is a non negative scaling factor that is model dependent and $sign(.)$ represents the signum function taken elementwise. X_{per} is then fed into the network for a forward pass to generate the final output.

It can be argued that the above method of generating gradients using the network's prediction can lead to inaccurate gradient information propagated through the network especially in cases where the network's output contains many misclassified pixels. The key insight we provide in this chapter is that despite misclassifications by the deep network, the gradients at the input obtained from the network's prediction, in general, improve the final output of the deep network.

4.3.2 Understanding Guided Perturbations

In this section, we perform several experiments to provide insight into different aspects of guided perturbations. Please refer to Figure 4.2 for the steps (Step 1, Step 2, Step 3) mentioned in this section.

Impact of perturbations on filter responses: To get a clear understanding of what happens during the forward pass in Step 3 that vastly changes the network's prediction, we visualize the filter responses for the FCN-32s network in Figure 4.3. This model was chosen due to its simpler architecture but we observed similar behavior in other deep architectures too. In Figure 4.3, we plot the average filter responses at different layers through the deep network after upsampling them bilinearly to image size. As can be observed, the influence of the added perturbations are not visually explicit until the *pool5* layer but the difference of the filter responses in Column (c) indicate that the information propagates from layers as early as *pool2*.

Next, we analyze the pixels for which the network predictions changed from Step 1 to Step 3. Figure 4.4(c) shows the pixels that were classified wrongly during

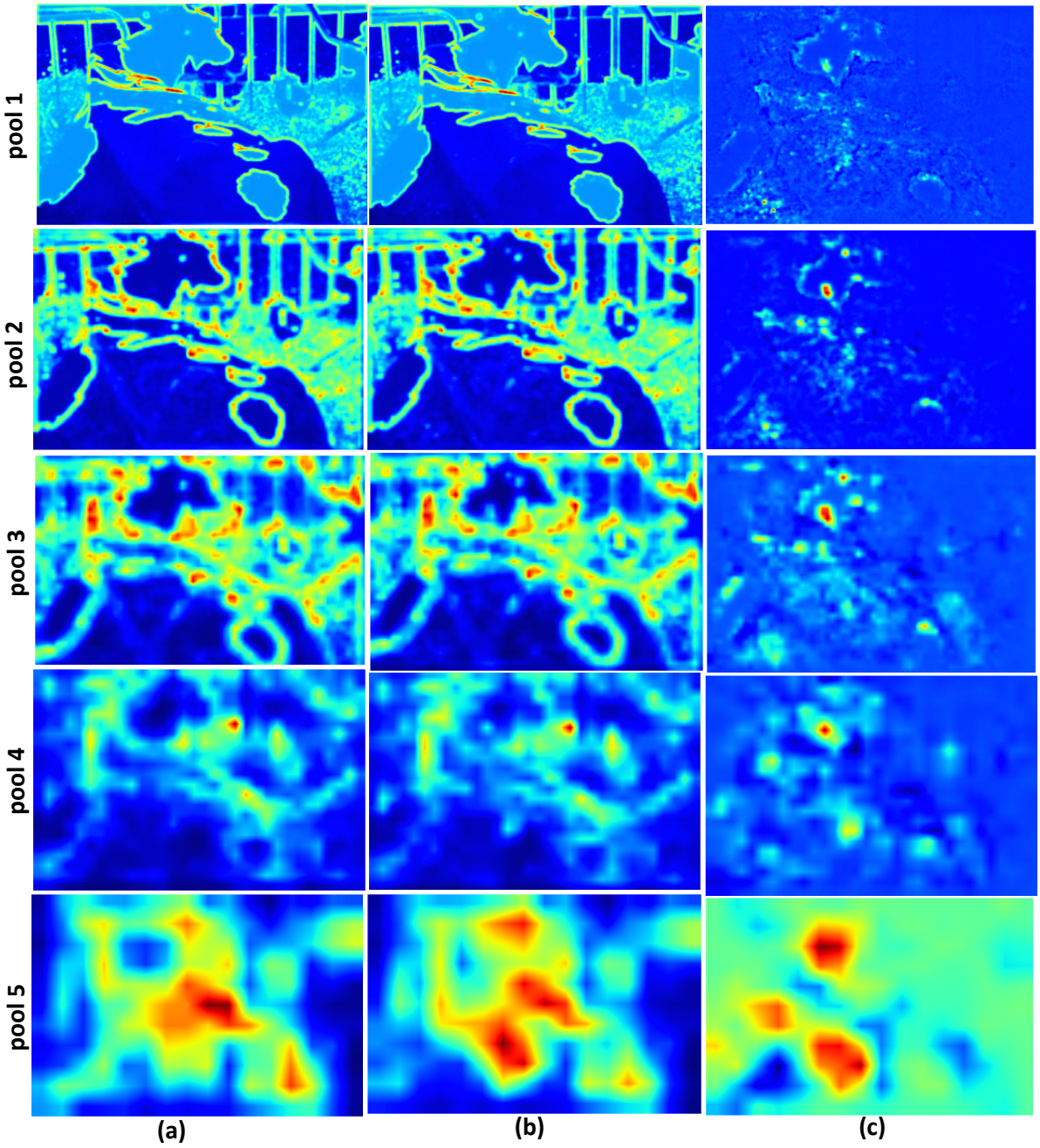


Figure 4.3: Visualization of filter responses showing how the correct context is propagated along the FCN-32s network. Column (a): filter responses during the forward pass using the original input. Column (b): filter responses during the forward pass using the perturbed input. Column (c): difference between (a) and (b)

the forward pass in Step 1 but were correctly classified at the final output. On the other hand, 4.4(d) shows the pixels that were correctly classified in Step 1 but were incorrect at the final output. Observe that, the correctly classified pixels between Step 1 and Step 3 are mostly internal to the image where additional contextual information is available for the network to switch its prediction whereas the small number of misclassified pixels are largely concentrated along the boundary regions of the image where the context is ambiguous. We present more of such visualization examples in the supplementary material.

Approximating ideal gradient direction: In this experiment, we would like to answer the question: what are the ideal perturbations that can be generated at the input? The best one can do is to use the ground truth to generate error gradients at the softmax layer which is then backpropagated to generate the perturbed input. When this perturbed input is fed back to the network, the result is a vastly improved prediction as shown in Figure 4.5 (c). While perturbations from ground truth significantly improve the performance, this information is not available during prediction time. The novelty of the current work is that the ground truth gradient

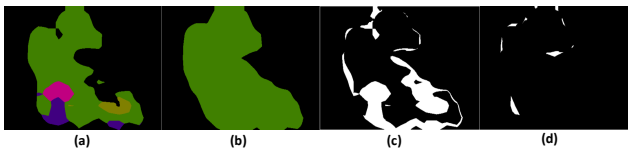


Figure 4.4: (a) Output of FCN-32s network (b) Output from the proposed approach (c) Pixels that were incorrectly classified by FCN-32s corrected by our approach (d) Pixels that were incorrectly classified by our approach that FCN-32s classified correctly.

direction is being approximated well enough by the predicted gradient directions that are computed using only the network's prediction.

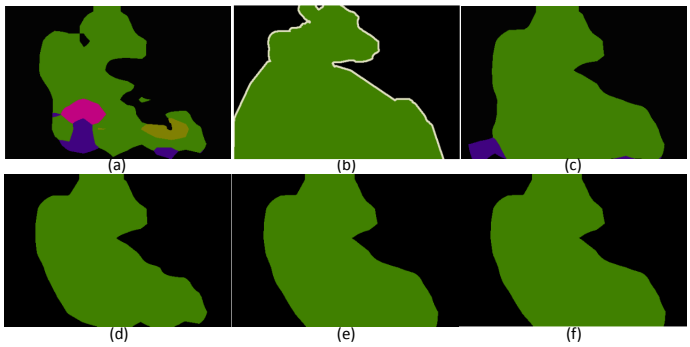


Figure 4.5: (a) Output of FCN-32s (b)Ground truth labeling (c) Output of perturbed input using ground truth gradients (d)-(f) Output of perturbed input using guided perturbations for iteration 1, 2 and 3 respectively.

To understand the extent of usefulness of the predicted gradients, we performed an experiment where the three steps outlined in our approach (Figure 4.2) is applied over successive iterations Figure 4.5 (d) shows the output of our approach obtained in the first iteration and Figure 4.5 (e)-(f) show the output over successive iterations. This shows that the most significant improvement happens at the first iteration and the subsequent iterations yield little improvement. We observe similar behavior on average over the PASCAL VOC2012 validation set.

Intuition based on overlapping receptive fields: In a CNN, the receptive fields of neighboring pixels define a context for their interactions. The advantage of having overlapping receptive fields is that the neighborhood connectivity is established automatically without explicitly specifying it. As long as the errors made by the CNN are sparse with respect to each pixel's receptive fields, the error gra-

dients when accumulated over the entire network and used to perturb the input image exhibit a corrective behavior. The effect of GP can be seen as a type of residual information that is propagated through the network which results in contextual smoothing. This is evident by looking at the filter responses in Figure 4.3, more specifically in Column 3, which shows the difference in responses with and without GP. It can be observed from the *pool5* responses that the peak activations occur around neighborhoods where there are competing classes. GP perturb these neighborhoods the most thus resulting in contextually smooth predictions in those regions.

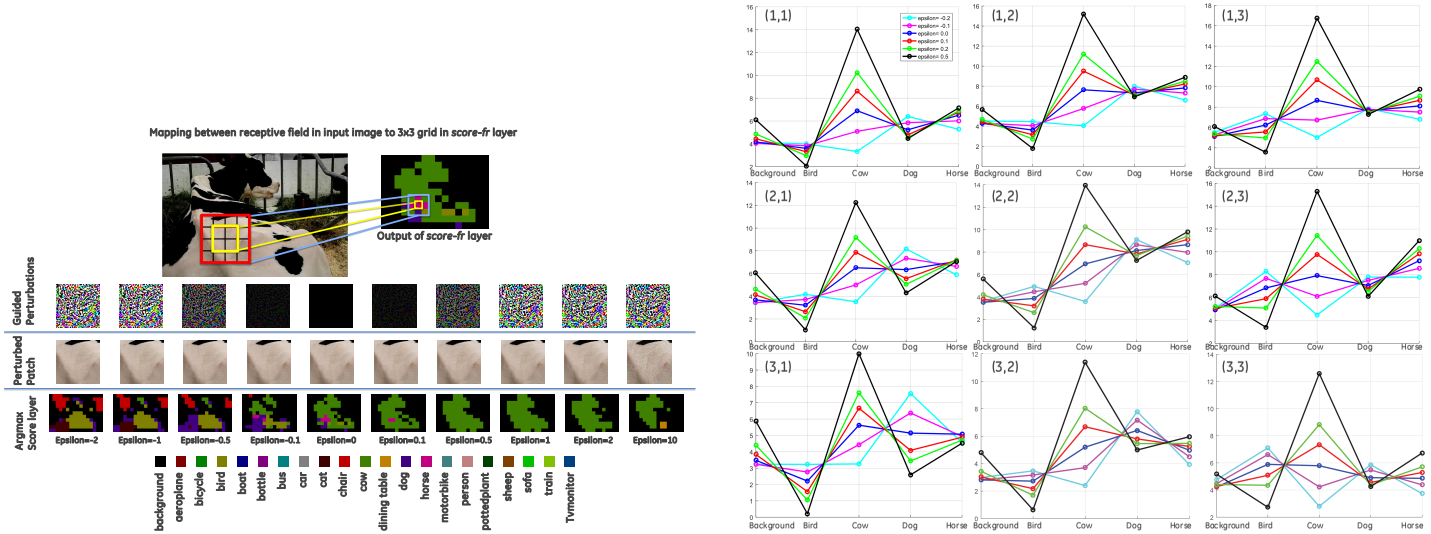
Analyzing GP in depth: In Figure 4.6, we show how guided perturbations impact the decisions made by the deep network by considering a local region in the input image and tracking its classification scores at *score-fr* layer (before upsampling layer) across different values of ϵ. In the top half of Figure 6.4a, the patch of interest in the RGB image is marked by a red box and its corresponding region in the *score-fr* output is marked by a blue box. Immediately below, the following are shown for different values of ϵ: (1) Guided perturbations generated at the input (2) perturbed RGB patches (3) output of the *score-fr* layer. Important observations that can be made from Figure 6.4a are:

- Input perturbations corresponding to positive ϵ improve the score output over a vast range of values. This visualization shows how guided perturbations are able to operate at a local level by leveraging neighborhood contextual information as can be directly observed from the images of score layer shown

in the bottom row.

- Even a small negative value of ϵ results in a large adverse effect on the score output, without any perceptible change in the perturbed RGB patch. This shows that a negative ϵ corresponds to an adversarial setting hence showing that adversarial perturbations exist for semantic segmentation networks. It should be noted that the effect is more severe in this case as compared to the case of image classification since a very small ϵ and a single step adversarial attack results in a huge decrease in performance. This can also be seen in Figure 4.8.

This discussion motivates our choice of using $\epsilon > 0$ to generate GP. To further analyze how these input perturbations affect the actual classifier score, we show in Figure 4.6b, the predictions of the deep network for the 3x3 grid in the *score-fr* output from Figure 6.4a, for different values of ϵ. For clarity, we only show the predicted scores of the top-5 classes. From the score values of the grid position (2,2), we can observe that as ϵ increases, the score of true class (*cow*) keeps increasing while the scores of the confusing classes do not vary much. The other plots show that this trend is observed across the entire neighborhood of the 3x3 grid. Thus, it can be inferred that perturbations at the input affect the decision of the deep network in a contextually consistent manner. We again observe that the score of the true class drops significantly even for a small negative ϵ which is consistent with our earlier observations.



(a) *(b)*

Figure 4.6: (a) The top half shows an RGB patch in the input image and its corresponding patch in the *score-fr* layer output of the FCN-32s network, before upsampling to image size. In the bottom half, we show, for different values of ϵ the guided perturbations, the perturbed RGB patches and the *score-fr* output. Notice how the the scores become contextually smoother for $\epsilon > 0$. (b) The actual score values of the top-5 predicted classes for the 3x3 grid marked in blue in figure (a) are plotted. Observe that for a range of positive values of ϵ, the correct class score (*cow*) dominates the others across the entire neighborhood. The legend in (1,1) applies to all the plots. Best viewed in screen. Please zoom for clarity.

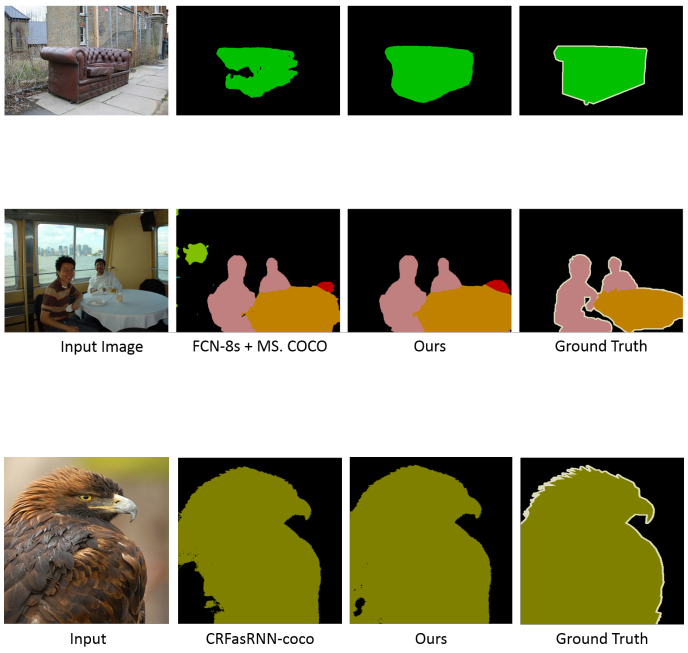


Figure 4.7: Qualitative results on the PASCAL VOC2012 reduced validation set. In the top two rows, we compare our result with the FCN-8s part of CRFasRNN that has been trained on MS.COCO dataset [LMB+14a] and publicly released by [ZJRP+15]. In the bottom row, we compare with the complete CRFasRNN framework [ZJRP+15]. More results can be found in supplementary material.

4.4 Experiments

In this section, we perform several experiments showing how our approach could be seamlessly applied on top of several pretrained deep networks. We test our method on the semantic segmentation task on PASCAL VOC2012 dataset [EEVG+15], scene labeling task on the PASCAL Context 59-class dataset [MCL+14a] and classification tasks on the MNIST and CIFAR10 datasets [Kri09]. These results

support how our approach is able to generalize across different types of problems in computer vision and highlights the advantage that it can be used with any pretrained model.

4.4.1 Evaluation Metrics

We evaluate our approach using the mean Intersection over Union (mIoU) metric commonly used for semantic segmentation as reported in [SLD16]. Let n_{ij} be the number of pixels of class i predicted to belong to class j, N_{cl} be number of classes, and $t_i = \sum_j n_{ij}$ be the total number of pixels of class i. It is then formulated as mean IoU $= \frac{1}{N_{cl}} \sum_i \frac{n_{ii}}{t_i + \sum_j n_{ji} - n_{ii}}$. For MNIST and CIFAR-10, we use classification accuracy as a metric to compare against the baseline.

4.4.2 Semantic Segmentation

We use PASCAL VOC2012 dataset for evaluating our approach for semantic segmentation task. It consists of 21 classes including background. We use the following pre-trained models as baselines and show the improvement that can be obtained using our approach for each of them:(1) FCN-32s and FCN-8s [SLD16]: these models are trained using the SBD dataset [HAB+11] that consists of 9,600 images. (2) FCN-8s-coco and CRFasRNN [ZJRP+15]: these are trained using the images from MS COCO [LMB+14b] and the SBD dataset using a total of 77,784 images. (3) Deeplab [CPK+14]: We evaluate on the Deeplab-VGG16 and ResNet101 models which use *atrous* convolutions and multi scale evaluation. They are also trained on MS COCO and the SBD datasets.

Table 4.1: Results on the reduced VOC2012 validation set with 346 images. '-coco' denotes that the model was trained on MS COCO data in addition to the SBD dataset. Numbers in brackets show the magnitude of change compared to the corresponding base models.

Method	Base	Base+GP
FCN-32s	62.10	64.71 (+**2.6**)
FCN-8s	63.97	66.97 (+**3.0**)
	MS-COCO data	
FCN-8s-coco	69.85	71.99 (+**2.1**)
CRFasRNN-coco	72.95	73.75 (+**0.8**)
Deeplab-VGG16	66.9	69.1 (+**2.2**)
Deeplab-ResNet101	74.1	75.3 (+**1.2**)

Table 4.2: Results on the PASCAL VOC2012 test set consisting of 1456 images using FCN-8s as the base network. Use of Guided Perturbations improves the performance of the base network on 19 out of 21 classes.

Method	bkg	aero	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbk	person	plant	sheep	sofa	train	tv	mean
FCN-8s [SLD16]	92.0	82.4	**36.1**	75.6	61.4	65.4	83.3	77.2	80.1	27.9	66.8	51.5	73.6	71.9	78.9	77.1	55.3	73.4	44.3	74.0	**63.2**	67.2
FCN-8s + GP	**92.4**	**84.4**	35.9	**79.3**	**62.6**	**70.5**	**86.2**	**80.0**	**82.8**	**28.0**	**71.9**	**55.2**	**74.6**	**75.6**	**80.2**	**77.4**	**56.9**	**75.6**	**45.8**	**77.4**	63.18	**69.3**

For all these methods, we use the publicly available models and apply the proposed approach on them. We use a single NVIDIA TitanX GPU for our experiments and CAFFE library [JSD+14] for implementation. The pretrained models used in this section are obtained from the CAFFE Model Zoo [mod] at the time of submis-

sion. All the reported results are computed with 1 iteration of our approach unless mentioned otherwise. Table 4.1 shows the results of applying the proposed approach to the different pretrained models during prediction time over a reduced validation set of 346 images as done in [ZJRP+15]. As can be observed, the proposed approach results in increased performance over all the listed pretrained models. This reiterates the fact that our approach is indeed architecture independent and can be easily integrated even with complex feedforward architectures like CRFasRNN. Table 4.2 shows the evaluation of our approach on PASCAL VOC2012 *test set* using FCN-8s pretrained network as the base model to demonstrate the improvement shown by our method in an unbiased setting. The ϵ value used for the test set was tuned on the validation set.

4.4.3 Scene Labeling

The scene labeling task is a dense pixel labeling task that is evaluated on the PASCAL Context dataset. While there are more than 400 classes defined, the challenge entails evaluating on the 59 classes that are specified as most frequent [MCL+14a]. The labeled classes contain scene elements in addition to objects that appear in the PASCAL VOC segmentation challenge, making this a much harder benchmark. To evaluate our approach on this task, we use the FCN-8s model from [SLD16] as our baseline that was trained on the standard training split of 10,000 images provided with the dataset. The results, which were generated on the validation set consisting of 5105 images are shown in Table 4.3. We improve the performance of the FCN-8s network by 1.3% which is significant given the large size

of the validation set. Please note that the ϵ value was not tuned to fit this dataset rather the best performing ϵ from Table 4.1 was used.

Table 4.3: Results on the PASCAL-Context 59-classes validation set.

Method	mean IU
FCN-8s	39.12
FCN-8s + GP	**40.44**

4.5 Ablative Experiments

For all the experiments in this section, we use FCN-32s network and the validation set used in section 4.4.2.

Speed-Performance trade-off The GPs generated at the input layer of a deep network improves the performance of the base model. However, there is a computational overhead due to performing an additional backward and forward pass. As an alternative, the backward pass could be performed up to an intermediate layer in the deep network instead of the input layer. In this section, we provide results addressing the trade off between computational time and resulting performance due to perturbing layers other than the input. It can be observed from Table 4.4 that even using the perturbed input from as late as *pool4* layer the improvement in performance remains almost constant while computation time drops significantly. This experiment shows that effect of GP is not only observed at the input but also in the intermediate layers of the deep network and hence can be leverages for reducing the

Table 4.4: Trade-off between performance and computation times obtained by truncating guided perturbations over different layers across the deep network. Original time taken is 0.12s per image. The baseline performance is 62.1%

layer	input	pool2	pool3	pool4
Time	0.33s	0.27s	0.24s	0.22s
mIOU	64.71	64.61	64.55	64.3

Table 4.5: Results with and without CRF

Method	Base	+CRF	+GP	+CRF+GP
FCN-8s-coco	69.8	71.1	72.0	72.7
Deeplab-ResNet-101	74.1	74.9	75.3	75.8

computational cost.

Comparison with CRF approaches The behavior of GP resembles the contextual smoothing provided by VRF approaches that have been popularly used in Semantic Segmentation. In this section, we provide empirical evidence that GP captures additional dependencies in the data compared to pairwise interactions that are modeled by CRFs. Table 4.5 shows the mean IoU values for cases with/without CRF applied on top of the network outputs. These results demonstrate that GP indeed captures extra dependencies compared to CRF and that GP can even improve upon CRF outputs.

Guided Perturbations (vs) other strategies In this section, we perform an ablative experiment where we perturb the input image in different ways in order to distinguish them from Guided Perturbations and show that the GP yields the most improvement in performance. As explained in Section 4.3.1, to generate a guided perturbation, we replace the softmax output with a one-hot encoded vector for the class of maximum confidence. We consider different methods to modify the label distribution that is obtained from the softmax function as follows:

- *random-onehot*: The class label is chosen in an uniformly random manner and used as ground truth instead of the maximum probability class.
- *Uniform-label*: An uniform label distribution is produced by assigning equal probability to all the classes and used as encoding to generate the error gradient.
- *top2-label*: Modified label distribution contains equal probability to top two predicted classes and used as encoding to generate the error gradient.

Figure 4.8 shows the effect of different types of label distribution on the segmentation performance. At the outset, it can be observed that GP gives the best quantitative performance of 64.7% compared to the second best case, which is the uniform setting with negative ϵ which scores 63.8%. We can also observe that when we perform GP, $\epsilon < 0$ corresponds to the adversarial setting. Intuitively, this setting is equivalent to maximizing the loss of the softmax classification function during training. Hence, the backpropagated gradient always moves away from the correct class. In our approach, GP is always generated by setting $\epsilon > 0$ as mentioned in

sections 4.3.1 and 4.5. The setting involving choosing a random label to generate the one-hot vector at the softmax output results in poor performance across all values of ϵ since gradient directions become random and the resulting perturbations adversely affect the performance of the deep network on the perturbed input image.

The interesting case to analyze from Figure 4.8 is the performance of the *Uniform-label* setting for $\epsilon < 0$. To understand this effect, Figure 4.9 illustrates a toy example showing the difference between the error gradients generated using GP and *Uniform-label* setting for a different possible output score distributions from the CNN. In this toy example, the CNN is trained to classify among 5 classes. Observe that, for the unimodal case, the gradient signal generated for a uniform output label distribution has the same relative magnitude as the gradient signal generated for GP but the dominant gradient direction is exactly the opposite. However, GP still gives a better performance compared to the uniform label distribution. In this case, the score vector is bimodal and hence there are two dominant directions in the gradient signal. Notice that the top gradient direction in the case of GP still points towards the correct class and all other directions move away from the correct class, as expected. But in the case of uniform label distribution, there are two competing directions and hence there is higher probability for the gradient to move in the wrong direction.

Effect of Scaling parameter We evaluate the performance of our approach using FCN-32s and FCN-8s networks over a range of scaling parameter ϵ on the validation set. Figure 4.10 shows how the performance varies based on the scaling factor. It

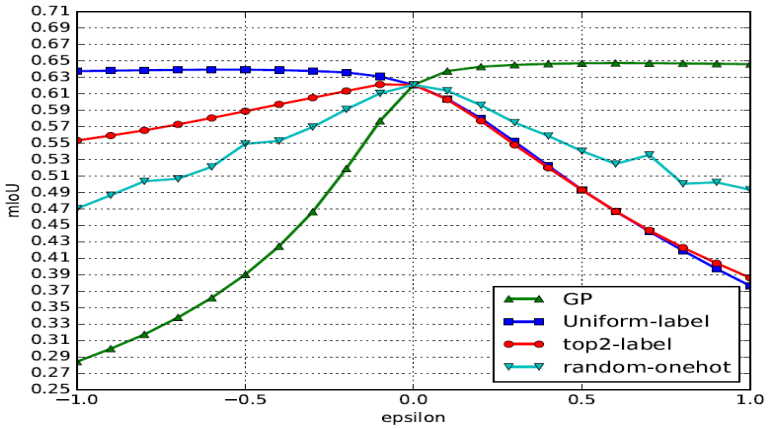


Figure 4.8: Mean IOU values for several perturbations generated by using different types of label distributions on the validation set over the range $\epsilon = [-1, 1]$ with FCN-32s as the base network. Please refer to section 4.5 for details.

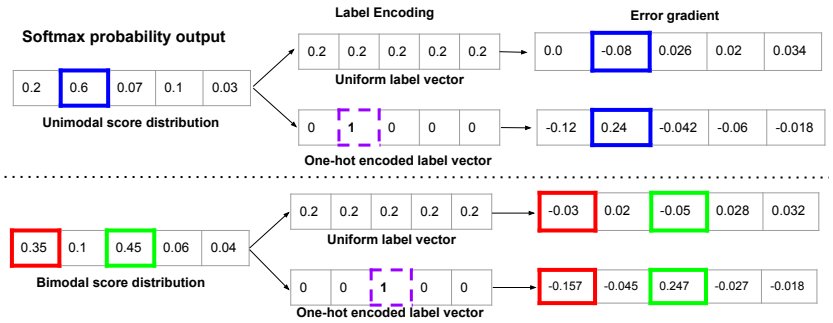


Figure 4.9: Difference in the gradient signal generated between *Uniform-label* setting and GP for the case of unimodal output score distribution (top) and bimodal output score distribution (bottom). The dominant gradient direction in both cases is shown in the colored boxes. The exact derivation for computing these gradient values is given in the supplementary material.

can be observed that improvement in performance is generally obtained over a wide range of values of ϵ. This indicates that network's behavior is not very sensitive to the value of ϵ though there seems to be an optimal value for best performance

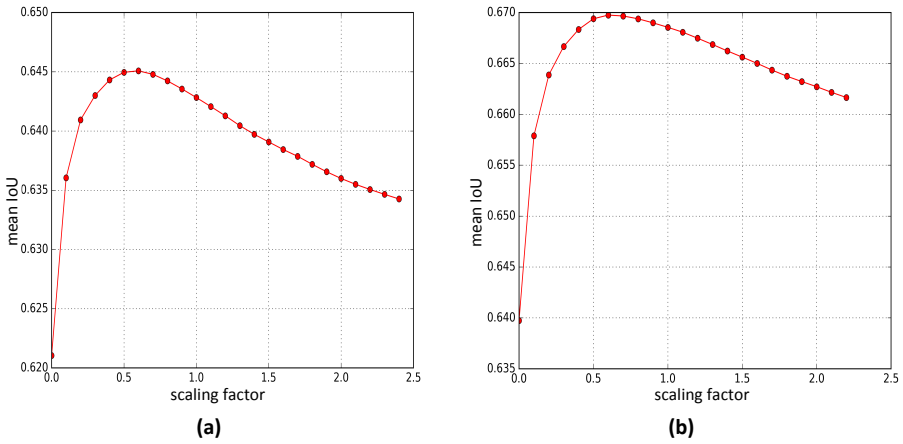


Figure 4.10: Effect of scaling factor ϵ on performance of FCN-32s (left) and FCN-8s (right) networks evaluated on the reduced PASCAL VOC2012 validation set. Best viewed in screen. Please zoom for clarity.

that depends on the deep model. We use $\epsilon = 0.55$ for FCN-32s, $\epsilon = 0.7$ for FCN-8s network and $\epsilon = 0.22$ for CRFasRNN network for our experiments.

Image Classification The method described in Section 4.3.1 for semantic segmentation cannot be applied directly for classification tasks. Since *context* for a classification task is not defined naturally, we extract contextual information from the learned feature space. Given an input image, we first extract the feature from the deep network and use it to select top k nearest neighbors from the training set using euclidean distance metric. We then perturb the test image with the weighted average of gradients generated using the class of the selected nearest neighbors and perform a forward pass to predict the final output. Let nn_i be the class of the i^{th} nearest neighbor. Following the notation established in Section 4.3.1, the equation for perturbed image is given as follows: $\mathrm{X}_{per} = \mathrm{X} + \epsilon \sum_{i=1}^{k} (w_i sign(\nabla_X \mathcal{J}(\theta, \mathrm{X}_i, \mathrm{Y}_{nn_i})))$,

where X_i is the i^{th} nearest neighbor; k is the number of nearest neighbors and w_i is weight associated with each nearest neighbor i and $\mathcal{J}(.)$ corresponds to the loss function. Figure 4.11 shows an example where the network correctly classifies the perturbed input generated using this procedure.

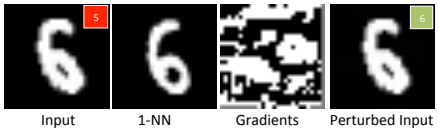


Figure 4.11: The input image is classified as '5'. By perturbing the input from the gradients generated using the top nearest neighbor class, the network changes its prediction to '6'

To evaluate the performance of GP on classification, we tested the method on two standard datasets: MNIST and CIFAR10. MNIST consists of grayscale images of digits while CIFAR10 consists of more realistic images of object classes. We follow the standard training/testing split for both the cases. We use 3 nearest neighbor with equal weights for all our experiments. For MNIST, we use a CNN with 2 conv. layers and 2 fully-connected layers with a 20-50-500-10 architecture and for CIFAR10, we use a CNN with 5 conv. and 2 fully-connected layers with a 64-64-128-128-128-128-10 architecture.

Table 4.6: Results on the classification task on MNIST and CIFAR10 datasets.

Dataset	Baseline	Proposed
MNIST	98.92	**99.15**
CIFAR10	76.31	**76.95**

Table 4.6 shows the results of our classification experiments. GP improves

performance over the baseline on both the datasets. However, the improvement in performance is not as high as in the segmentation case which could be attributed to two reasons: (1) the base networks themselves have learned a very strong representation and (2) the context information in the classification task is relatively weak compared to the segmentation task.

4.6 Conclusion

In this chapter, we have shown novel self-corrective behavior of CNNs for segmentation and classification tasks. We showed that Guided Perturbations can improve the network's performance without additional training or network modification. We have demonstrated this effect on several publicly available datasets and using different deep network architectures. We have presented several experiments that try to understand and explain different aspects of guided perturbations. Incorporating the insight from the perturbation analysis, we showed that the label requirement can be significantly reduced in an active learning setting for the semantic segmentation task. We believe that this behavior can lead to novel network designs and better end-to-end training procedures.

4.7 Error gradient computation for Guided Perturbations

Let the score output of the deep network be: $\mathrm{z} \in \mathbb{R}^{N_c}$. To get a probability distribution over classes, this is passed through a softmax operator whose output is given as: $\mathrm{y} = \left\{ \frac{e^{z_i}}{\sum_i e^{z_i}} \right\}_{i=1}^{N_c}$, where N_c is the number of classes. If $k \in [1, N_c]$ is the

correct class, then the error gradient computed at the softmax output with respect to its input z is given as follows: Let $\sum_C$ denote $\{\sum_{i=1}^{N_c} e^{z_i}\}$, then

$$\text{if } i = k: \quad \frac{\partial y_i}{\partial z_i} = \frac{\sum_C .e^{z_i} - e^{z_i}.e^{z_i}}{\sum_C^2} = \frac{e^{z_i}}{\sum_C}\left(1 - \frac{e^{z_i}}{\sum_C}\right) = y_i(1 - y_i) = y_k(1 - y_i) \tag{4.2}$$

$$\text{if } i \neq k: \quad \frac{\partial y_i}{\partial z_k} = -\frac{0 - e^{z_i}.e^{z_k}}{\sum_C^2} = -\frac{e^{z_i}}{\sum_C}\frac{e^{z_k}}{\sum_C} = -y_i y_k = y_k(0 - y_i) \tag{4.3}$$

(4.3) could be summarized in the following single equation:

$$\frac{\partial \mathrm{y}}{\partial \mathrm{z}} = y_k(\boldsymbol{\ell} - \mathrm{y}) \tag{4.4}$$

where $\boldsymbol{\ell} \in \mathbb{R}^{N_c}$ is the label distribution, which in this case is a one hot vector with $l_k = 1$ and others zero. For a more general case, where $\boldsymbol{\ell}$ defines a distribution among classes, this formula generalizes in a straight forward manner as follows:

$$\frac{\partial \mathrm{y}}{\partial \mathrm{z}} = (\boldsymbol{\ell} \cdot \mathrm{y})(\boldsymbol{\ell} - \mathrm{y}) \tag{4.5}$$

It can be observed that (4.5) is a general version of (4.4) since the maximum probability value y_k is replaced by the dot product between the label distribution and the output of softmax operation.

4.8 Additional Results

This section contains additional material and examples to illustrate the claims made in this chapter. Figures 4.12, 4.13 and 4.14 show additional examples of improved performance when the proposed approach is used with the FCN-8s [SLD16],

FCN8s-coco [ZJRP+15] and CRFasRNN [ZJRP+15] pretrained models respectively. Figure 4.15 shows examples that illustrate that the pixels that are predicted correctly by our approach are more internal to the image whereas the small number of pixels that are predicted wrongly tend to occur towards the boundaries. These examples are generated using the FCN-32s deep network [14]. Finally, Figure 4.16 shows additional results of using our approach for the MNIST classification task.

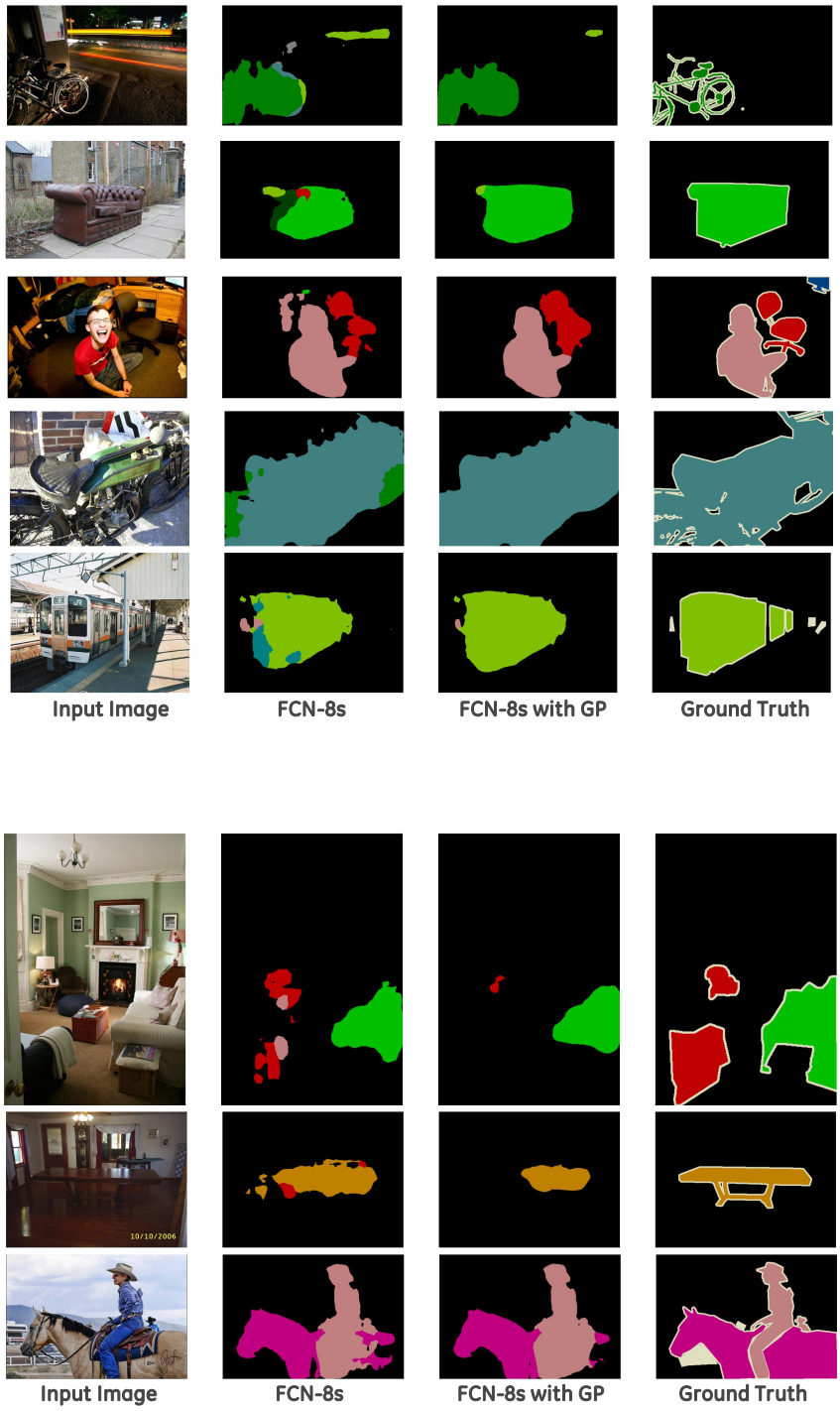
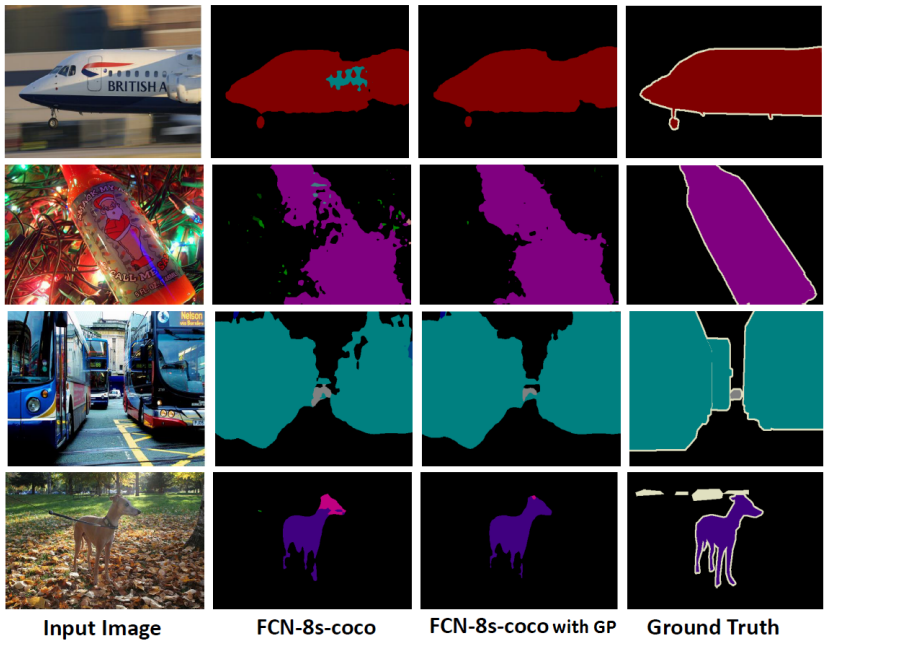


Figure 4.12: Qualitative results on the PASCAL VOC2012 reduced validation set - Comparison with FCN-8s pretrained model. Top half shows the successful outputs, Bottom half shows the failure cases.



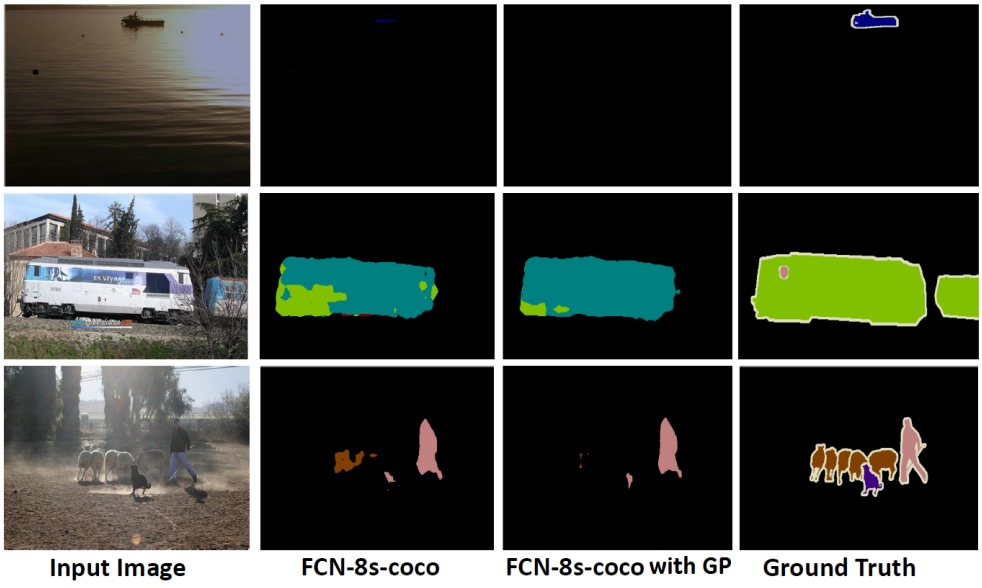


Figure 4.13: Qualitative results on the PASCAL VOC2012 reduced validation set - Comparison with FCN-8s-coco pretrained model. Top half shows the successful outputs, Bottom half shows the failure cases.

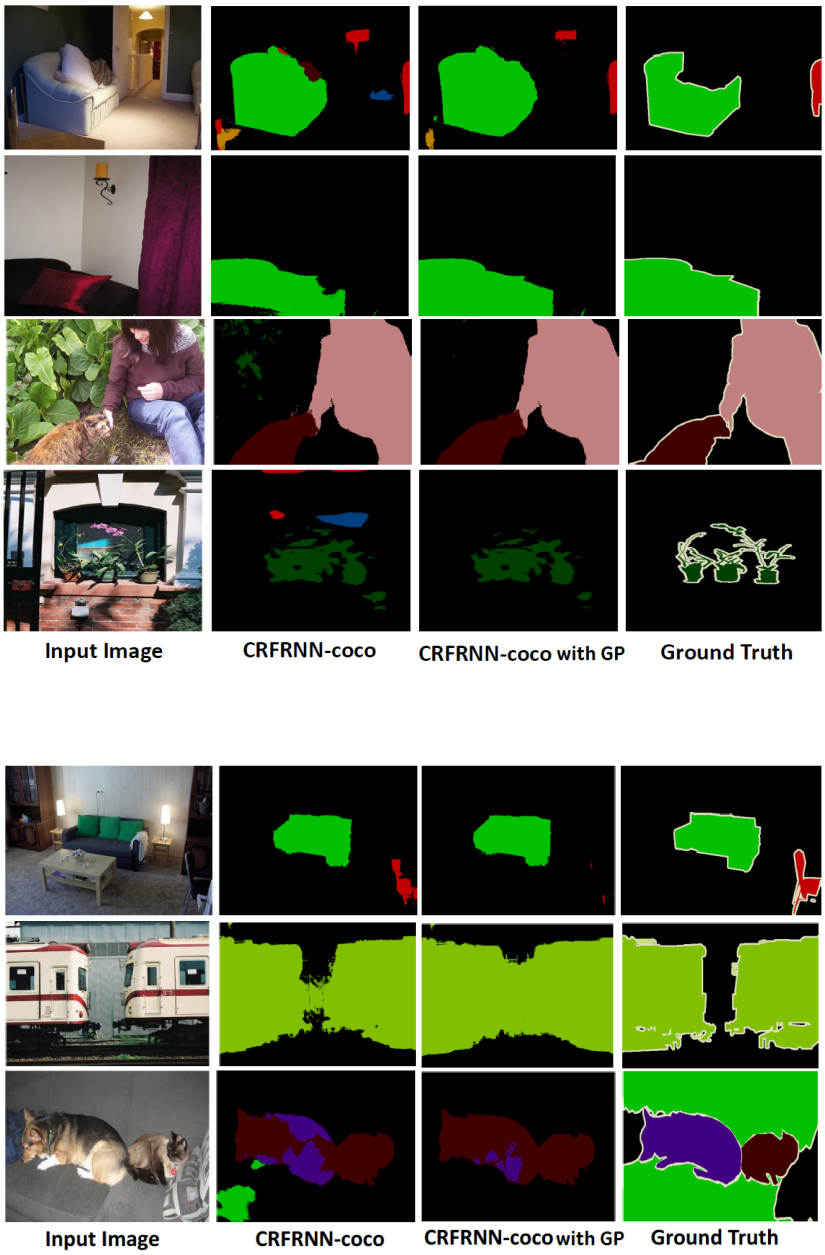


Figure 4.14: Qualitative results on the PASCAL VOC2012 reduced validation set - Comparison with CRFRNN-coco pretrained model. Top half shows the successful outputs, Bottom half shows the failure cases.

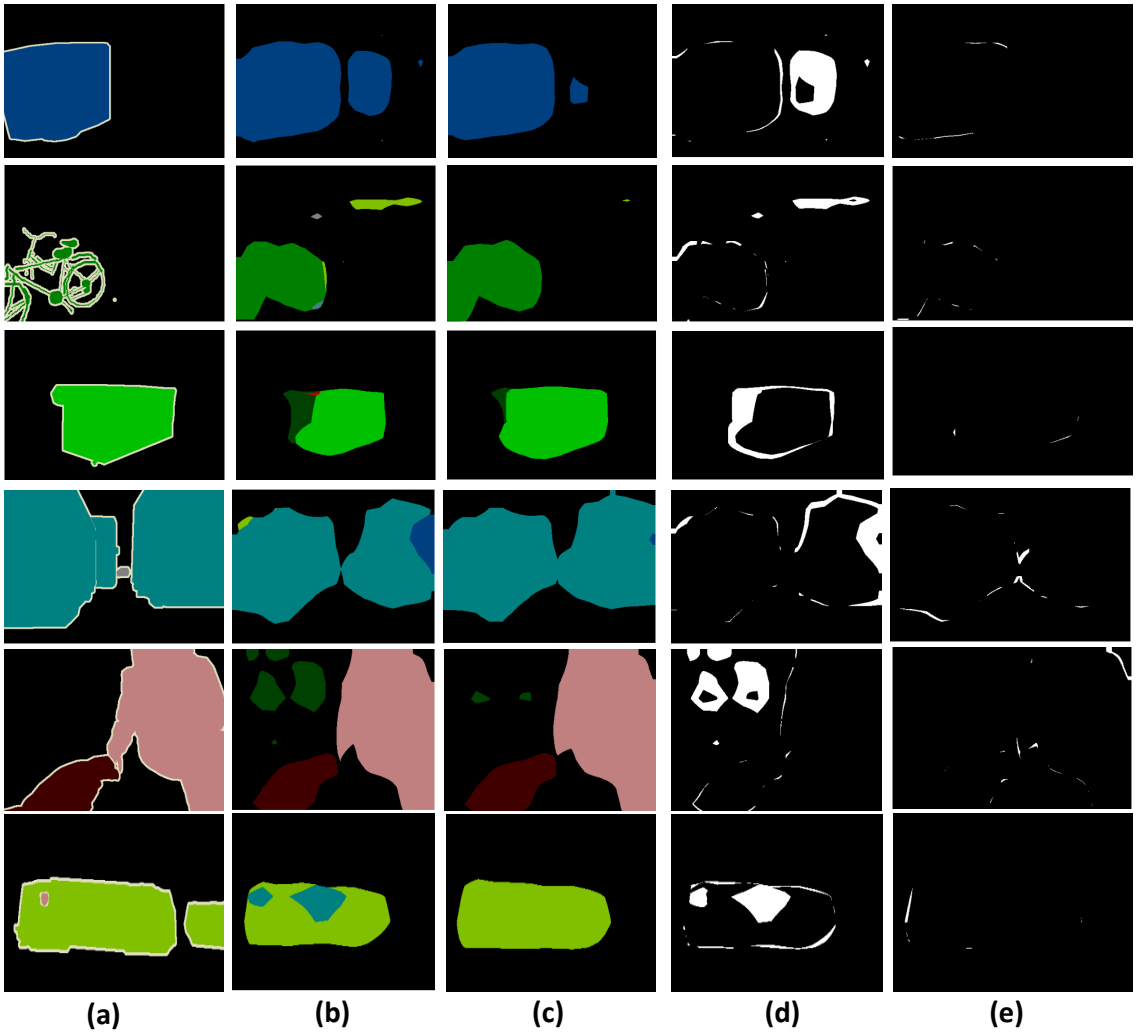


Figure 4.15: (a) Ground truth (b) Output of FCN-32s network (c) Output from the proposed approach (d) Pixels that were incorrectly classified by FCN-32s corrected by our approach (e) Pixels that were incorrectly classified by our approach that FCN-32s classified correctly.

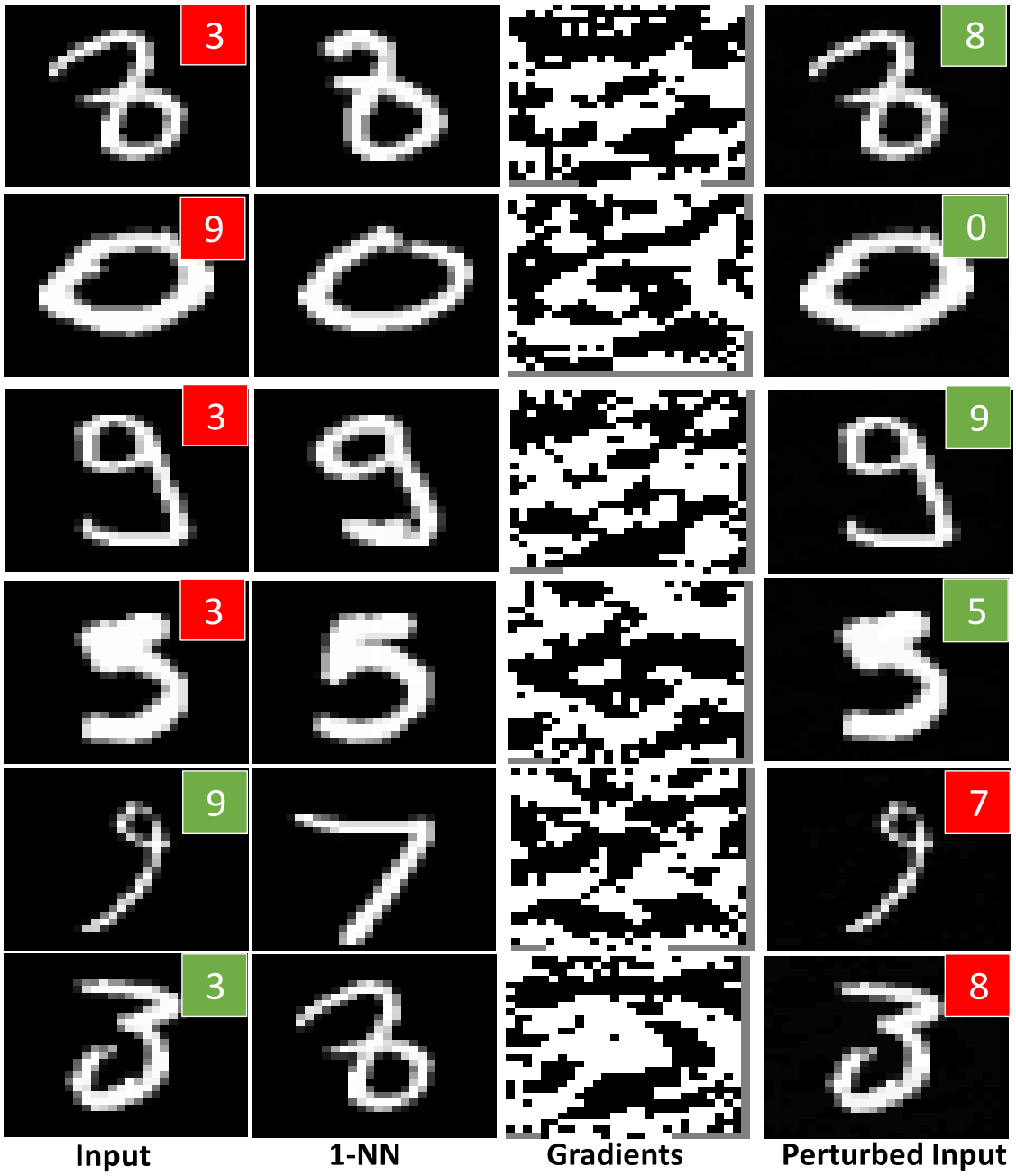


Figure 4.16: Example results of using the proposed approach for MNIST digits classification task. Top four rows shows situations where our approach was successful in correcting the classifier errors while bottom two rows showcase the failures. The red and green labels show the final deep network output: red indicates a mistake and green indicates correct prediction.

Chapter 5. Regularizing Deep Neural Networks by Layerwise Adversarial Training

5.1 Introduction

[1] Deep neural networks (DNNs) have shown tremendous success in several computer vision tasks in recent years [[HZRS16a], [SKP15b], [KSH12b]]. However, seminal works on adversarial examples [[GSS14b], [SZS+13b]] have shown that DNNs are susceptible to imperceptible perturbations at input and intermediate layer activations. From an optimization perspective, they also showed that adversarial training can be used as a regularization approach while training deep networks. The focus of adversarial training techniques has been to improve network robustness to gradient based adversarial perturbations. In this chapter, we propose a novel variant of adversarial training with a focus to improve regularization performance on test data.

The proposed approach is efficient and simple to implement. It uses adversarial perturbations of intermediate layer activations to provide a stronger regularization compared to traditional techniques like Dropout ([SHK+14] [SHK+14]). By gen-

[1] The material presented in this chapter is part of an externally published work [SJCL18]

erating the perturbations from a different class compared to the current input, we ensure that the resulting perturbations in the intermediate layers are directed towards an adversarial class. This forces the network to learn robust representations at each layer resulting in improved discriminability. Even though these perturbations are not adversarial at the input layer, we show that they are strongly adversarial when applied at the intermediate layers.

The proposed regularization approach does not add any significant overhead during training and thus can be easily extended to very deep neural networks, as shown in our experiments. Our approach complements dropout and achieves regularization beyond dropout. It avoids over-fitting and generalizes well by achieving significant improvement in performance on the test set. We show that the trained network develops robustness against adversarial examples even when it is not explicitly trained with adversarial inputs. While previous works have focused on generating adversarial perturbations for standalone images, our work focuses on using these to efficiently regularize training. We perform several ablative experiments to highlight the properties of the proposed approach and present results for very deep networks such as VGG [SZ14b], ResNets [HZRS16a] and state of the art models such as WideResNets [ZK16] on CIFAR-10, CIFAR-100 and ImageNet datasets.

5.2 Related Work

Many approaches have been proposed to regularize the training procedure of very deep networks. Early stopping and statistical techniques like weight decay

are commonly used to prevent overfitting. Specialized techniques such as DropConnect [WZZ+13], Dropout [SHK+14] have been successfully applied with very deep networks such as ResNets. Faster convergence of such deep architectures was made possible by Batch Normalization (BN) [IS15]. One of the added benefit of BN was that the additional regularization provided during training even made dropout regularization unnecessary in some cases.

The work of Szegedy *et al.* [SZS+13b] showed the existence of adversarial perturbations for computer vision tasks by solving a box-constrained optimization approach to generate these perturbations. They also showed that training the network by feeding back these adversarial examples regularizes the training and makes network resistant to adversarial examples. Due to a relatively expensive layerwise training procedure, their analysis was limited to small datasets and shallow networks. [GSS14b] proposed the fast gradient sign method to generate such adversarial examples. To perform adversarial training, they proposed a modified loss function to also account for loss from adversarial examples. They showed significant improvements in the network's response to adversarial examples and obtained a regularization performance beyond dropout. [MMKI17] proposed a virtual adversarial training framework and show its regularization benefits for relatively deep models, while taking three times the normal training time. [MDFF16] proposed an iterative approach to generate much stronger adversarial perturbations and also presented a score function to measure robustness of classifiers against these examples. Furthermore, recent approaches such as deep contrastive smoothing [GR14], distillation [PMW+16] and stability training [ZSLG16] have focused solely on im-

proving robustness of the deep models to adversarial inputs. In this chapter, we present an efficient layerwise approach to adversarial training and demonstrate its ability as a strong regularizer for very deep models beyond the specialized methods mentioned above, in addition to improving model robustness to adversarial inputs.

Recent theoretical works such as [[FFF15], [FMF16]] analyze the effect of random, semi-random and adversarial perturbations on classifier robustness. They presented fundamental upper bounds on the robustness of classifier which depends on factors such as curvature of decision boundary and distinguishability between class cluster centers. Wang *et al.* [WGQ16] introduced the notion of strong robustness for classifiers and point out that the differences between generalization and robustness by characterizing the topology of the learned classification function. We perform empirical studies that show that the proposed approach improves performance by suppressing those dimensions that are unnecessary for generalization. In addition, we observe that pure adversarial training techniques suppress most dimensions resulting in strong robustness against adversarial examples but less improvement in generalization. To the best of our knowledge, ours is the first work to explore this comparison between regularization and adversarial robustness by providing empirical results on very deep networks.

5.3 Our Approach

In this section, we present the proposed regularization approach and highlight the differences between related methods that use adversarial training for regulariza-

tion. In addition, we perform a small scale experiment to study the properties of the proposed approach by analyzing the singular value spectrum of the Jacobian. We also visualize the impact of these perturbations on the intermediate layer activations and conclude by illustrating the connection to robust optimization-based approaches.

We begin by defining some notation. Let $\{\mathrm{X}_i\}_{i=1}^N$ denote the set of images and $\{y_i\}_{i=1}^N$ denote the set of labels. Let $f : \mathrm{X} \in \mathbb{R}^m \mapsto \mathrm{y} \in \mathbb{L}$ denote the classifier mapping that maps the image to a discrete label set, $\mathbb{L}$. In this chapter, f is modeled by a deep CNN unless specified otherwise. We denote the loss function of the deep network by $\mathcal{J}(\theta, \mathrm{X}, \mathrm{y})$ where θ represents the network parameters and $\{\mathrm{X}, \mathrm{y}\}$ are the input and output respectively. The deep network consists of L layers and $\nabla_l \mathcal{J}(\theta^t, \mathrm{X}^t, \mathrm{y}^t)$ denotes the backpropagated gradient of the loss function at the output of the l^{th} layer at iteration t. In the above expression, $l = 0$ corresponds to the input layer and $l = L - 1$, the loss layer. Let X_l^t be the input activation to the l^{th} layer and r_l^t represents the perturbation that is added to X_l^t. For clarity, we drop the subscript l when talking about the input layer.

5.3.1 Overview of Adversarial training methods

Previous works on adversarial training have observed that training the model with adversarial examples acts as a regularizer and improves the performance of the base network on the test data. [SZS+13b] [SZS+13b] define adversarial perturbations r as a solution of a box-constrained optimization as follows: Given an input X and target label y, they minimize $||\mathrm{r}||_2$ subject to (1) $f(\mathrm{X} + \mathrm{r}) = y$ and (2) $\mathrm{X} + \mathrm{r} \in$

$[0,1]^m$. Note that, if $f(\mathrm{X}) = y$, then the optimization is trivial (i.e. $\mathrm{r} = 0$), hence $f(\mathrm{X}) \neq y$. While the exact minimizer is not unique, they approximate it using a box-constrained L-BFGS. More concretely, the value of c is found using line-search for which the minimizer of the following problem satisfies $f(\mathrm{X}+\mathrm{r}) = \hat{y}$, where $\hat{y} \neq y$:

$$\underset{r}{\operatorname{argmin}}\ c||\mathrm{r}||_2 + \mathcal{J}(\theta, \mathrm{X}+\mathrm{r}, y), \quad \text{subject to} \quad \mathrm{X}+\mathrm{r} \in [0,1]^m \tag{5.1}$$

This can be interpreted as finding a perturbed image $\mathrm{X}+\mathrm{r}$ that is closest to X and is misclassified by f. The training procedure for the above framework involves optimizing each layer by using a pool of adversarial examples generated from previous layers. As a training procedure, this is rather cumbersome even when applied to shallow networks having 5-10 layers. To overcome the computational overhead due to the L-BFGS optimization performed at each intermediate layer, [GSS14b] proposed the Fast Gradient Sign (FGS) method to generate adversarial examples. By linearizing the cost function around the value of the model parameters at a given iteration, they obtained a norm constrained perturbation as follows: $\mathrm{r}_{fgs} = \epsilon.sign(\nabla\mathcal{J}(\theta, \mathrm{X}, y))$. They show that the perturbed images $\mathrm{X} + \mathrm{r}_{fgs}$ reliably cause deep models to misclassify their inputs. As noted in [SYN15], the above formulation for adversarial perturbation can be understood by looking at a first order approximation of the loss function $\mathcal{J}(\cdot)$ in the neighborhood of the training sample X:

$$\tilde{\mathcal{J}}(\theta, \mathrm{X}+\mathrm{r}, y) = \mathcal{J}(\theta, \mathrm{X}, y) + \langle \nabla\mathcal{J}(\theta, \mathrm{X}, y), r \rangle \tag{5.2}$$

The FGS solution (r_{fgs}) is the result of maximizing the second term with respect to r, with a l_∞ norm constraint. They trained the network with the following

objective function with an added adversarial objective:

$$\tilde{\mathcal{J}}(\theta, \mathrm{X}, y) = \alpha \mathcal{J}(\theta, \mathrm{X}, y) + (1 - \alpha)\mathcal{J}(\theta, \mathrm{X} + \mathrm{r}_{fgs}, y) \tag{5.3}$$

By training the model with both original inputs and adversarially perturbed inputs, the objective function in (5.3) makes the model more robust to adversaries and provides marginal improvement in performance on the original test data. Intuitively, the FGS procedure can be understood as perturbing each training sample within a L_∞ ball of radius ϵ, in the direction that maximally increases the classification loss.

The focus of the adversarial training techniques described above is to improve a model's robustness to adversarial examples. As a by-product, they observed that an adversarial loss term can also marginally regularize the deep network training. In this chapter, we propose a novel approach which is an extension of the traditional adversarial training. The focus of our approach is to improve regularization and as an interesting by-product, we observe improvements in adversarial robustness of the trained model as well.

5.3.2 Proposed Formulation

The proposed approach differs in the following aspects compared to the formulations discussed above: (1) Generating adversarial perturbations from intermediate layers rather than just using the input layer (2) Using the gradients from the previous batch to generate adversarial perturbations for the activations of the current batch. In order to facilitate the representation of layerwise activations in the loss

Algorithm 1 Efficient layerwise adversarial training procedure for improved regularization

1: Inputs: Deep network f with loss function $\mathcal{J}$ and parameters θ containing C convolutional blocks. B^t is the batch sampled at iteration t of size k, with input-output pairs $\{\mathrm{X}^t, \mathrm{y}^t\}$. Gradient accumulation layers $\{G_c\}_{c=1}^{C}$, with stored perturbations $R^t = \{\mathrm{r}_c^t\}_{c=1}^{C}$, initialized with zero. Perturbation parameter, ϵ.
2: t=0:
3: Sample a batch $\{\mathrm{X}^t, \mathrm{y}^t\}$ of size k images from the training data
4: Perform regular forward pass - G_c's are not active for $t = 0$.
5: Perform backward pass using the classification loss function. Each gradient accumulation layer G_c stores the gradient signal backpropagated to that layer:

$$\mathrm{r}_c^{t+1} = sign(\nabla_c \mathcal{J}(\theta^t, \mathrm{X}^t + R^t, \mathrm{y}^t)), \forall c = [1, C] \tag{5.4}$$

6: **for** t in 1:$|B| - 1$ **do**
7: Sample a batch $\{\mathrm{X}^t, \mathrm{y}^t\}$ of size k from the training data
8: Perform forward pass with perturbation: Each gradient accumulation layer acts as follows. Let X_c^t be the input to block c, then:

$$G_c(X_c^t) = X_c^t + \epsilon \cdot \mathrm{r}_c^t \tag{5.5}$$

9: Perform backward pass updating r_c^t to r_c^{t+1} for all blocks c as in Eq. 5.4 above.
10: **end for**
11: **Test time usage:** During test time, the gradient accumulation layers (G_c) are removed and f behaves as a standard feed forward deep network.

function, we denote the collection of layerwise responses as $X = \{x_l\}_{l=0}^{L-1}$ and the set of layerwise perturbations as $R = \{r_l\}_{l=0}^{L-1}$. Then, $\mathcal{J}(\theta, X + R, y)$ denotes the loss function where intermediate layer activations are perturbed according to the set R. The notation used in the previous section is a special case where $R = \mathrm{r}_{fgs}$. Now, consider the following objective to obtain the perturbation set R:

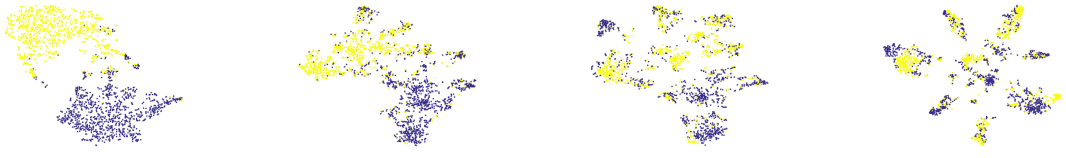
$$\begin{aligned} &\underset{R}{\text{argmax}}\ \mathcal{J}(\theta, X + R, y) \\ &\text{subject to} \quad ||r_l||_\infty \leq \epsilon,\ \forall l,\ f(X + R) \neq y \end{aligned} \tag{5.6}$$

Ideally, for each training example x, the solution to the above problem, consists of generating the perturbation corresponding to the maximally confusing class; in

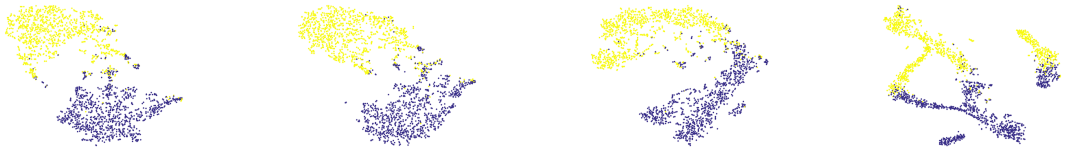
other words, by choosing the class $\hat{y}$ which maximizes the divergence, $KL(p(\mathrm{y}|\mathrm{x}_{L-1}), p(\hat{\mathrm{y}}|\mathrm{x}_{L-1}))$. In the absence of any prior knowledge about true class co-occurences, solving this explicitly for each training sample for every iteration is time consuming. Hence we propose an approximate solution to this problem: the gradients computed from the previous sample at each intermediate layer are cached and used to perturb the activations of the current sample. In a mini-batch setting, this amounts to caching the gradients of the previous mini-batch. To ensure the class constraint in Eq. (5.6) is satisfied, the only requirement is that successive batches have little lateral overlap in terms of class labels. From our experiments, we observed that any random shuffle of the data satisfies this requirement. For more discussion on this, please refer to the experiments section. Given this procedure of accumulating gradients, we are no longer required to perform an extra gradient descent-ascent step as in the FGS method to generate perturbations for the current batch. Since the gradient accumulation procedure does not add to the computational cost during training, this can be seamlessly integrated into any existing supervised training procedure including even very deep networks as shown in the experiments.

The training procedure is summarized in Algorithm 2, where $sign(\cdot)$ denotes the signum function. We add the gradient accumulation layers after the Batch Normalization layer in each convolutional block (conv-BN-relu). In case BN layers are not present, we add gradient accumulation layers after each convolution layer. A subtle detail that is overlooked in the algorithm is that the value of ϵ is not constant over all the layers, rather it is normalized by multiplying with the range of the gradients generated at the respective layers. During test time, the gradient

accumulation layers (G_c) are removed from the trained model.



(a) $\epsilon = 0, 91.9\%$ (b) $\epsilon = 0.02, 69.4\%$ (c) $\epsilon = 0.04, 48.2\%$ (d) $\epsilon = 0.06, 34.6\%$



(e) $\epsilon = 0, 91.9\%$ (f) $\epsilon = 0.02, 91.85\%$ (g) $\epsilon = 0.04, 91.9\%$ (h) $\epsilon = 0.06, 85.8\%$

Figure 5.1: t-SNE visualization of the final fc-layer features of dimension 512 of the VGG network for two randomly chosen classes of the CIFAR-10 data for different values of the intensity, ϵ. The top row shows the effect of the perturbations generated using the proposed approach while the bottom row shows random perturbations of the same intensity. It is clear that the random perturbations do not affect the linear separability of the data, while the proposed perturbations are extremely effective in leading the network to misclassify the perturbed data.

To understand the effect of the the proposed layerwise perturbations described in the previous section, we compare them with random layerwise perturbations. Figure 5.1 shows a two dimensional t-SNE [dMH08] visualization of the embeddings belonging to the penultimate FC-layer for a range of values of ϵ, the intensity of the adversarial perturbation. We used a pretrained VGG network that was trained on the CIFAR-10 dataset to compute the embeddings for two randomly chosen

classes from the test data. In the bottom row, the effect of random perturbations with zero mean and unit standard deviation, applied layerwise on the original data is also shown. From the visualization and the accuracy values, it can clearly be observed that the proposed perturbations when added to the original data makes the network misclassify the original data. Hence training using these perturbations results in good regularization and improved robustness. Notice that even for higher values of ϵ, the data perturbed by layerwise random gradient directions remains clearly linearly separable while the data perturbed by the accumulated gradients is unable to be distinguished by the base model.

5.3.3 Toy example

In order to acquire a better understanding of the regularizing properties of the mapping function learned using the proposed approach, we perform a toy experiment using a small neural network consisting of two fully connected layers of sizes 1024 and 512. Each fully connected layer is followed by a hyperbolic tangent activations. We use the grayscale version of the CIFAR-10 dataset as our testbed and L_2 norm weight decay was applied during training. No data augmentation or other regularization methods such as dropout were used during training. We train three networks: a baseline network, a network with the gradient accumulation layers (as in Algorithm 1) added after each fully connected layer and a network using the FGS training approach. Cross entropy loss was used to train all the networks. In terms of classification accuracy, the proposed method improves the baseline performance from 39.5% to 43.3% on the original data while the accuracy of the FGS network is

40.5%.

Singular Value Analysis: To gain a deeper understanding of the encoder mapping learned by each network, we perform an analysis similar to [RVM+11] by computing the singular values of the Jacobian of the encoder. Since this is a small architecture, we are able to explicitly compute the Jacobian for each sample in the test set. The average singular value spectrum of the Jacobian for the test data is shown in Figure 5.2. We can make the following observations: (a) The singular value spectrum computed for ours and FGS approach has fewer dominant singular values and decays at a much faster rate compared to base network (b) The FGS training suppresses the response of the network strongly in all the dimensions while our approach achieves a strong suppression only for trailing dimensions. This implies that our network is able to better capture data variations that are relevant for classifying original test data hence providing improved performance. On the other hand, FGS achieves slightly improved robustness against adversarial examples compared to our approach by suppressing the network's response strongly even in leading dimensions, as demonstrated in our ablative experiments in the next section.

5.3.4 Connection to Robust Optimization

Several regularization problems in machine learning such as ridge regression, lasso or robust SVMs have been shown to be instances of a more general robust optimization framework [SNW12]. To point out the connection between the proposed approach and robust optimization, we borrow the idea of uncertainty sets from [SYN15]. To explain briefly, an uncertainty set denoted by $\mathcal{U} = \mathcal{B}_\rho(x, \epsilon)$ rep-

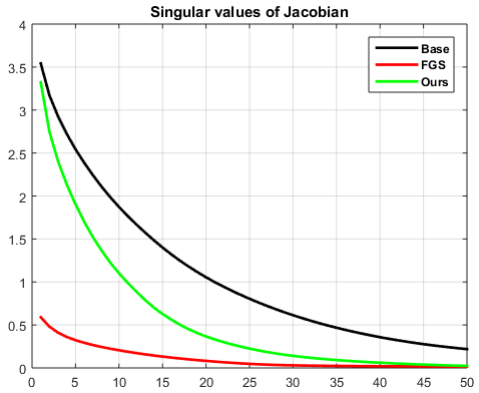


Figure 5.2: Average singular value (SV) spectrum showing top 50 SVs for the toy example presented in the text. A model regularized with the proposed approach is compared with a FGS regularized model and baseline model with no regularization.

resents an epsilon ball around x under norm ρ. [GSS14b] point out that adversarial training can be thought of as training with hard examples that strongly resist classification. Under the setting of uncertainty sets, adversarial training with the FGS method could be seen as generating the worst case perturbations from the input space $\mathcal{U}$ under the l_∞ norm. In this chapter, we extend the idea of uncertainty sets from input activations to intermediate layer activations. This can be thought of as sampling perturbations from the feature space learned by the deep network. Let $\mathcal{U}_l$ represent the uncertainty set of the activation x_l at layer l. Then, the proposed adversarial training approach is equivalent to sampling perturbations from the intermediate layer uncertainty sets which makes the feature representation learned at those layers to become more robust during training. Moreover, by generating perturbations from inputs that do not belong to the same class as the current input, the directions sampled from the uncertainty set tend to move the perturbed

feature representation towards the direction of an adversarial class. This effect can be observed from the t-SNE visualization shown in Figure 5.1.

5.4 Experiments

In this section, we provide an experimental analysis of the proposed approach to show that layerwise adversarial training improves the performance of the model on the original test data and increases robustness to adversarial inputs.

5.4.1 Classification accuracy on CIFAR-10/100

To demonstrate the generality of our training procedure, we present results on CIFAR-10 and CIFAR-100 [KH09] using VGG, ResNet-20 and ResNet-56 networks. For the ResNet networks, we use the publicly available torch implementation [Res17]. For the VGG architecture, we use a publicly available implementation which consists of Batch Normalization [VGG17]. For all the experiments, we use the SGD solver with Nesterov momentum of 0.9. The base learning rate is 0.1 and it is dropped by 5 every 60 epochs in case of CIFAR-100 and every 50 epochs in case of CIFAR-10. The total training duration is 300 epochs. We employ random flipping as a data augmentation procedure and standard mean/std preprocessing was applied conforming to the original implementations. For the ResNet baseline models, without regularization, we find that they start overfitting if trained longer and hence we perform early stopping and report their best results. For the perturbed models, we find that no early stopping is necessary; the learning continues for a longer duration and

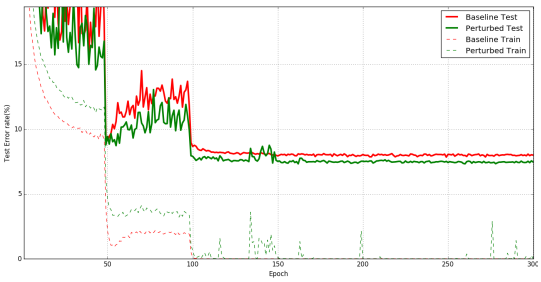
shows good convergence behavior. We refer to the model trained using the proposed approach described in Algorithm 2 as *Perturbed* throughout this section. Dropout was not used in any of the training procedure in this experiment. We explicitly compare our approach against dropout in the ablative experiments. Figure 5.3 plots the training and test error rates for the baseline model and the proposed approach. It can be observed that our method converges faster and achieves better generalization error. Note that, we used only random shuffling in all our experiments since we found that performing a controlled shuffling to improve mini-batch overlap across i. and hence we use

Table 5.1: Classification accuracy (%) on CIFAR-10 and CIFAR-100 for VGG and Resnet architectures. Results reported are average of 5 runs.

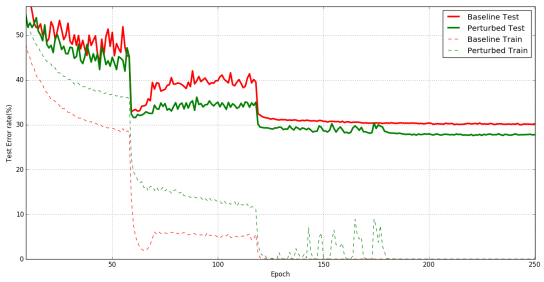
Type (CIFAR-10)	Baseline	Perturbed
VGG	92.1 ± 0.3	92.65 ± 0.2
Resnet-20	90.27 ± 0.4	91.1 ± 0.3
Resnet-56	91.53 ± 0.3	94.1 ± 0.2

Type (CIFAR-100)	Baseline	Perturbed
VGG	69.8 ± 0.5	72.3 ± 0.3
Resnet-20	64.0 ± 0.2	66.9 ± 0.3
Resnet-56	68.2 ± 0.4	71.4 ± 0.5

Imagenet Experiment: To test the applicability of our regularization approach over a large scale dataset, we conducted an experiment using the ImageNet dataset (train: 1.2M images, val: 50K images). We used AlexNet as the base architecture. We used the publicly available implementation from the torch platform [Ale17] and both the baseline and the regularized models were trained from scratch to 60 epochs. The classification accuracies obtained were: Baseline - 56.1%, Proposed - 59.2%, an increase of 3.1%. This shows that our approach can signifi-



(a) CIFAR-10



(b) CIFAR-100

Figure 5.3: Training and test error rates for VGG network trained on CIFAR-10 and CIFAR-100 datasets. The training errors are computed using perturbed activations in each epoch. The red color indicates the baseline model and green indicates the model regularized with the proposed approach, referred as *Perturbed* in the text. Best viewed in color. Please zoom in for clarity.

cantly improve the performance of deep neural networks even on a large and diverse corpus like Imagenet.

5.4.2 Performance with noisy labels

In this experiment, we show the effect of the proposed regularization approach when the training labels provided with the data are corrupted. We assume that the corruption process is bernoulli with probability $p \in \{0, 0.1, 0.25, 0.5, 0.75\}$, with higher values of p denoting larger amount of corruption. While conventional machine learning systems fail to report results for this setting, we would like to point out that this is a practical and realistic situation. Table 5.2 shows the results on CIFAR-10 comparing the proposed approach with Dropout for varying levels of random label corruption. It can be seen that for reasonable amount of label noise, the propose

Table 5.2: Classification accuracy on CIFAR-10 with varying levels of label corruption. Results reported as average over 5 runs.

p	0.0	0.1	0.25	0.5	0.7
Base	92.1	88.5	83.3	73.4	54.1
Dropout	92.4	89.1	84.6	74.8	55.6
Proposed	**92.8**	**91.7**	**88.5**	**81.8**	**59.3**

regularization approach provides a huge improvement in classification performance as compared to other strategies such as Dropout. This can be attributed to the fact that during training, the perturbations of the intermediate layers that correspond to a form of label noise as exhibited in Figure 5.1.

5.5 Ablative Experiments and Discussion

Comparison with Dropout: We perform an experiment where we compare the regularization performance of the proposed adversarial training approach to Dropout. We use the VGG architecture used in the previous sections and perform experiments with and without dropout on CIFAR-10 and CIFAR-100 datasets.

Table 5.3: Comparison of classification accuracy (%) with/without dropout on CIFAR-10 and CIFAR-100 for the VGG model

Type (CIFAR-10)	Baseline	Perturbed
w/o Dropout	92.1	92.7
with Dropout	92.5	93.2

Type (CIFAR-100)	Baseline	Perturbed
w/o Dropout	69.8	72.3
with Dropout	70.5	73.1

The following observations could be made from Table 5.3: (1) The perturbed

model performs better than the baseline model with or without dropout. Thus, the proposed training improves the performance of even dropout based networks. (2) On a complex task like CIFAR-100, the proposed adversarial training based regularization gives better performance compared to that provided by dropout (70.5% (vs) 73.1%). Since the proposed adversarial perturbations are intended to move the inputs towards directions that strongly resist correct classification, they are able to create a more discriminative representation for tasks with a larger number of classes.

Table 5.4: Effect of adding gradient accumulation layers incrementally (from shallow to deeper layers) throughout the deep network. The numbers reported are the classification accuracy using the VGG network on the CIFAR-10 dataset. Baseline performance is 89.4%. *conv1 to* indicates we start adding the gradient accumulation layers from *conv1* upto the mentioned layers such as *pool1*, *pool2* etc.

Layer (conv1 to)	pool1	pool2	pool3	pool4	pool5
Accuracy	89.5	89.62	90.24	91.1	91.3

Perturbing deeper layers: In this section, we analyze the effect of adversarial perturbations starting from the lowest convolutional layers which model edges/shape information to the more deeper layers which model abstract concepts. For this experiment, we use the VGG network with batch normalization that was used in the previous section. The experiments were performed on the CIFAR-10 dataset. No data augmentation or dropout is applied. It is clear from the results in Table 5.4 that the improvement in performance due to the proposed layerwise perturbations become significant when applied to the deeper layers of the network,

Table 5.5: Comparison of the strength of adversarial examples between the FGS approach applied at the input and using the proposed layerwise perturbations. Reported numbers are classification accuracies for different values of ϵ.

Type	$\epsilon = 0$	$\epsilon = 0.05$	$\epsilon = 0.1$	$\epsilon = 0.15$	$\epsilon = 0.2$
FGS [GSS14b]	92.4	53.28	41.58	36.44	33.85
Ours - all layers	92.4	48.56	21.72	14.76	14.19

which is in line with the observation made by [SZS+13b]. While performing layerwise alternate training as proposed by [SZS+13b] becomes infeasible for even moderately deep architectures, our training scheme provides an efficient framework to infuse adversarial perturbations throughout the structure of very deep models.

Adversarial strength of the proposed perturbations: Traditional adversarial training techniques improve the performance on adversarial examples by explicitly making the network robust to adversarial gradient directions. Thus, an important question that needs to be addressed in light of the proposed optimization strategy is: Are the gradient directions generated from the previous samples(as described in Algorithm 1) adversarial ? To answer this question, we perform an empirical experiment to measure the performance of a deep model (3 convolutional layers + 1 fc-layer) trained using CIFAR-10 training data, on the CIFAR-10 test data. No adversarial training was used to train this model. As described earlier, for each test sample, the intermediate layer activations are perturbed using gradients accumulated from the previous sample. For comparison, we also show the performance of the same model on the adversarial data generated using the FGS method.

From the metrics in Table 5.5, it can be observed that using accumulated gradients from the previous batch as adversarial perturbations results in a bigger drop in performance. This signifies that the aggregated effect of layerwise perturbations is more adversarial compared to perturbing only the input layer as done in the FGS approach. We performed an additional experiment where only the input layer was perturbed using the gradients of the previous sample instead of perturbing all the intermediate layers. We found that this resulted in negligible drop in the baseline performance, indicating that these gradients are not adversarial enough when used to perturb only the input.

Variants of the proposed approach: In the proposed training method summarized in Algorithm 2 (referred as Ours-orig), each batch of inputs is perturbed at intermediate layers by the gradients accumulated from the previous batch. In this section, we present an empirical comparison between the following variants:

- FGS-orig: The original FGS joint loss based adversarial training as proposed

Table 5.6: Comparison of classification accuracy (%) between the different variants of the proposed approach and FGS method (FGS-orig) for different values of ϵ

Type	0	0.02	0.04	0.06	0.08	0.1
Baseline	89.4	67.5	49.6	41.2	37.3	34.7
FGS-orig [GSS14b]	88.7	86.4	84.1	81.4	80.5	77.1
FGS-inter	90.9	87.79	83.85	79.65	74.69	69.92
Ours-orig	91.2	87.95	83.84	79.11	73.66	68.37
Ours-joint	91.5	86.07	81.38	75.72	70.25	64.76

by [GSS14b] and shown in Eq. (5.3). We used a value of $\alpha = 0.5$; we did not find other values yield any significant improvements.

- FGS-inter: In this setting, different from [GSS14b], we use the FGS gradients to perturb the intermediate layer activations and use the joint loss with $\alpha = 0.5$.

- Ours-joint: This setting is same as Ours-orig with the exception that we use the joint loss formulation with $\alpha = 0.5$. Note that, Ours-orig corresponds to setting where $\alpha = 0$

All the models are trained on the CIFAR-10 dataset. No data augmentation or dropout regularization is applied. The training parameters are similar to the ones used in the previous section. We generate adversarial test data for the CIFAR-10 test dataset using the FGS method, since it has been shown to generate adversarial examples reliably. We then test the models on the original and adversarial test data for different values of the adversarial strength ϵ. Table 5.6 shows the results of the different training strategies. $\epsilon = 0$ corresponds to the original test data and other values of ϵ indicate the strength of adversarial FGS perturbation added to the input image. From these results, we make the following observations: (1) Approaches based on perturbing intermediate layers (FGS-inter,Ours-orig,Ours-joint) improve the performance on the original data significantly as compared to perturbing only the input but they marginally decrease the adversarial test performance. (2) On the other hand, perturbing only the input layer (FGS-orig) yields the best adversarial test performance among the compared approaches while performing marginally

worse than the baseline on the original test data. These observations indicate the possibility of a trade-off that exists between adversarial robustness and regularization effect over clean data. Referring to the toy example described earlier, the singular value analysis performed there also supports our claim that methods which impart adversarial robustness tend to suppress sensitivity of the model in all the dominant directions; while the proposed approach provides a nice trade-off by retaining those directions which are essential for modeling the variations in the training data. This ensures that our approach results in better regularization performance on clean data while providing comparable robustness on adversarial data.

Results on WideResNet models: Wide Residual Networks are recently proposed deep architectures that generated state of the art results on CIFAR-10 and CIFAR-100 datasets. In this experiment, we use their publicly available implementation and train them from randomly initialized weights using the proposed approach using the parameter settings described in the experiments section. Specifically, the Ours-joint approach described in the previous section is used for training. As data augmentation, we applied flipping and random cropping as done in their native implementation. The results are shown in Table 5.8. For the compared methods, we quote their published accuracy values. It can be observed that the proposed approach results in improved performance compared to both the baseline model and the model regularized with dropout. This demonstrates the generalization ability of our approach to state of art deep models.

Table 5.7: Classification error rates (%) on CIFAR-10 and CIFAR-100 for WideResNet (WRN) architectures. Our results are reported as average of 5 runs. For comparison we provide the published WRN baseline results. $^{(*)}$ denotes the results obtained by a single run.

Model	#params	CIFAR-10	CIFAR-100
WRN-28-10 [ZK16]	36.5M	4.00	19.25
WRN-28-10 with dropout [ZK16]	36.5M	3.89	18.85
WRN-40-10 with dropout* [ZK16]	51.0M	3.8	18.3
WRN-28-10 - **Ours**	36.5M	**3.62** $\pm$ 0.05	**17.1** $\pm$ 0.1

5.5.1 Results on Wide Residual Networks (WRN)

Wide Residual Networks are recently proposed deep architectures that generated state of the art results on CIFAR-10 and CIFAR-100 datasets. In this experiment, we use their publicly available implementation and train them from scratch using the proposed adversarial training approach using the parameter settings described in the original paper. Specifically, the Ours-joint approach is used for training. As data augmentation, we applied flipping and random cropping as done in their native implementation. The results are shown in Table 5.8.

Table 5.8: Classification error rates (%) on CIFAR-10 and CIFAR-100 for WideResNet (WRN) architectures. Our results are reported as average of 5 runs. For comparison we provide the published WRN baseline results. $^{(*)}$ denotes the results obtained by a single run.

Model	#params	CIFAR-10	CIFAR-100
WRN-28-10	36.5M	4.00	19.25
WRN-28-10 with dropout	36.5M	3.89	18.85
WRN-40-10 with dropout*	51.0M	3.8	18.3
WRN-28-10 with Ours-joint	36.5M	**3.62** $\pm$ 0.05	**17.1** $\pm$ 0.1

5.5.2 Response to local perturbation depends on the Jacobian

Let $f : \mathbb{R}^m \mapsto \mathbb{R}^n$ be a mapping between two metric spaces (euclidean, for simplicity) . Then, for $x \in \mathbb{R}^m$, let $J_f(x) \in \mathbb{R}^n \times \mathbb{R}^m$ denote the jacobian of f evaluated at x. Let $\delta x \in \mathbb{R}^m$ be a bounded local perturbation in the neighborhood of x. A first order truncated expansion of $f(x + \delta x)$ is given as:

$$f(x + \delta x) = f(x) + J_f(x)^T \delta x$$

We can bound the frobenius norm of the second term as follows:

$$||J_f(x)^T r||_F \overset{(a)}{\leq} ||J_f(x)||_F \, ||\delta x||_2 = \sqrt{\sum_{i=1}^{min(m,n)} \sigma_i^2} \cdot ||\delta x||_2$$

$$\implies ||J_f(x)^T \delta x||_F \leq \sqrt{\sum_{i=1}^{min(m,n)} \sigma_i^2} \cdot ||\delta x||_2$$

where $|| \cdot ||_F$ denotes the frobenius norm; for vectors, it is the same as the L_2 norm and σ_i denotes the i^{th} singular value of the Jacobian; (a) is a direct application of Cauchy-Schwarz inequality. Applying this result to the singular value spectra plotted earlier in this chapter, we see that the base network without adversarial training is extremely sensitive to local perturbations compared to adversarially trained networks using FGS and the proposed approach.

5.5.3 Setting ϵ parameter

Let $\nabla_l \mathcal{J}(\theta, x, y)$ denote the gradient of the loss function backpropagated to the l^{th} layer. Let $M = max(\nabla_l \mathcal{J}(\theta, x, y))$, $m = min(\nabla_l \mathcal{J}(\theta, x, y))$. Then, the value of ϵ for each layer is calculated as: $\epsilon_l = \epsilon \cdot (M - m)\ \forall l$, where $\epsilon \in \{10, 20, 30\}$. The exact value is cross-validated using a held out set. In practice, we found our training approach to not be overly sensitive to ϵ. We tuned ϵ only for the VGG network on CIFAR-10 and used the same value for all the other networks such as ResNets and WideResNets on both CIFAR-10 and CIFAR-100 datasets. Note that, for cases where a fixed value of ϵ is specified, the same value is used for all layers ignoring the normalizing factor, $(M - m)$.

Generalization to non-image signals: In this chapter, we have considered adversarial perturbations in the space of natural images. The existence of adversarial perturbations has been shown to exist in other types of signals that occur in speech recognition ([SAB+16], [CMV+16]) and language processing tasks [MDG16]. While the focus of the this chapter has been image signals, there exists a natural extension of our framework to the above modalities. Such an extension is trivially possible since end-to-end learning systems such as deep networks are used in speech and language tasks as well. As future work, we propose to extend the current approach to explore robustness aspects of deep networks trained on modalities other than images.

5.6 Conclusion

While the behavior of CNNs to adversarial data has generated some intrigue in computer vision since the work of [SZS+13b], its effects on deeper networks have not been explored well. We observe that adversarial perturbations for hidden layer activations generalize across different samples and we leverage this observation to devise an efficient regularization approach that could be used to train very deep architectures. Through our experiments and analysis we make the following observations: (1) Contrary to recent methods which are inconclusive about the role of perturbing intermediate layers of a DNN in adversarial training, we have shown that for very deep networks, they play a significant role in providing a strong regularization (2) The aggregated adversarial effect of perturbing intermediate layer activations is much stronger than perturbing only the input (3) Significant improvement in classification accuracy entails capturing more variations in the data distribution while adversarial robustness can be improved by suppressing the unnecessary variations learned by the network 5.3.3. By providing an efficient adversarial training approach that could be used to regularize very deep models, we hope that this can inspire more robust network designs in the future.

Chapter 6. Learning from Synthetic Data: Addressing Domain Shift for Semantic Segmentation

6.1 Introduction

[1]

Deep Convolutional Neural Networks (DCNNs) have revolutionized the field of computer vision, achieving the best performance in a multitude of tasks such as Image Classification [HZRS16b], Semantic Segmentation [LSD15], Visual Question Answering [LYBP16], etc. This strong performance can be attributed to the availability of abundant labeled training data. While annotating data is relatively easier for certain tasks like image classification, they can be extremely laborious and time-consuming for others. Semantic segmentation is one such task that requires great human effort as it involves obtaining dense pixel-level labels. The annotation time for obtaining pixel-wise labels for a single image from the CITYSCAPES dataset is about 1 hr., highlighting the level of difficulty ([COR+16], [RVRK16]). The other challenge lies in collecting the data: While natural images are easier to obtain, there are certain domains like medical imaging where collecting data and finding experts

[1] The material presented in this chapter is part of an externally published work [SBJ+18]

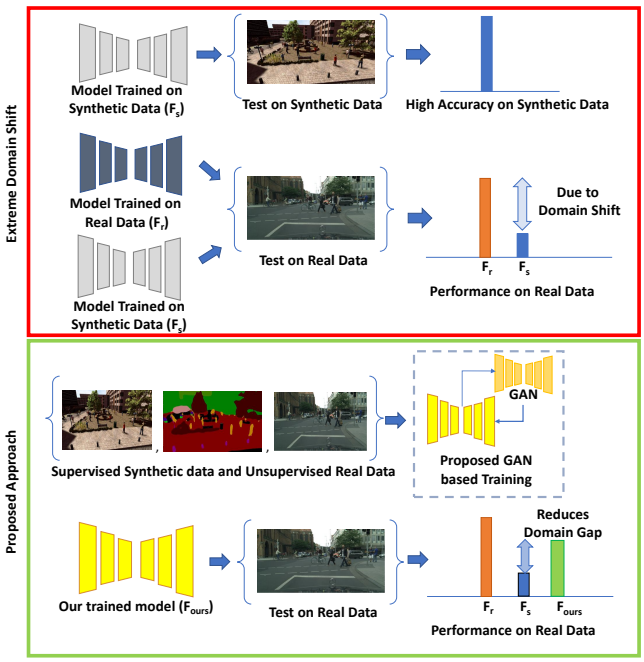


Figure 6.1: Characterization of Domain Shift and effect of the proposed approach in reducing the same

to precisely label them can also be very expensive.

One promising approach that addresses the above issues is the utility of synthetically generated data for training. However, models trained on the synthetic data fail to perform well on real datasets owing to the presence of domain gap between the datasets. Domain adaptation encompasses the class of techniques that address this domain shift problem. Hence, the focus of the approach presented in this chapter is in developing domain adaptation algorithms for semantic segmentation. Specifically, we focus on the hard case of the problem where no labels from the target domain are available. This class of techniques is commonly referred to as unsupervised domain adaptation.

Traditional approaches for domain adaptation involve minimizing some mea-

sure of distance between the source and the target distributions. Two commonly used measures are Maximum Mean Discrepancy (MMD) ([GTX11], [LCWJ15] [LWJ16]), and learning the distance metric using DCNNs as done in Adversarial approaches ([GL14], [THSD17]). Both approaches have had good success in the classification problems; however, as pointed out in [ZDG17], their performance improvement does not translate well to the semantic segmentation problem. This motivates the need for developing new domain adaptation techniques tailored to semantic segmentation.

The method we present in this chapter falls in the category of aligning domains using an adversarial framework. Among the recent techniques that address this problem, FCN in the wild [HWYD16] is the only approach that uses an adversarial framework. However, unlike [HWYD16] where a discriminator operates directly on the feature space, we project the features to the image space using a generator and the discriminator operates on this projected image space. Adversarial losses are then derived from the discriminator. We observed that applying adversarial losses in this projected image space achieved a significant performance improvement as compared to applying such losses directly in the feature space (ref. Table 6.5).

The main contribution of this chapter is that we propose a technique that employs generative models to align the source and target distributions in the feature space. We first project the intermediate feature representations obtained using a DCNN to the image space by training a reconstruction module using a combination of L_1 and adversarial losses. We then impose the domain alignment constraint by forcing the network to learn features such that source features produce target-like images when passed to the reconstruction module and vice versa. This is accom-

plished by employing a series of adversarial losses. As training progresses, the generation quality gradually improves, while at the same time, the features become more domain invariant.

6.2 Related Work

Semantic segmentation is a well studied problem in computer vision. The Fully Convolutional Networks (FCN) by Shelhamer *et al* [LSD15] signified a paradigm shift in how to fully exploit the representational power of CNNs for the pixel labeling task. While performance has been steadily improving for popular benchmarks such as PASCAL VOC [EGW+10] and MS-COCO [L+14], they do not address the challenges of domain shift within the context of semantic segmentation.

Domain adaptation has been widely explored in computer vision primarily for the classification task. Some of the earlier approaches involved using feature reweighting techniques [DI07], or constructing intermediate representations using manifolds [GLC11] [GSSG12] or dictionaries [NQC13]. Since the advent of deep neural networks, emphasis has been shifted to learning domain invariant features in an end-to-end fashion. A standard framework for deep domain adaptation involves minimizing a term that measures domain discrepancy along with the task being solved. Some approaches use Maximum Mean Discrepancy and its kernel variants for this task [LCWJ15] [LWJ16], while others use adversarial approaches.

We focus on adversarial approaches since they are more related to our work. Revgrad [GL14] performs domain adaptation by applying adversarial losses in the

feature space, while PixelDA [BSD+16] and CoGAN [LT16] operate in the pixel space. While these techniques perform adaptation for the classification task, there are very few approaches aimed at semantic segmentation. To the best of our knowledge, [HWYD16] and [ZDG17] are the only two approaches that address this problem. FCN in the wild [HWYD16] proposes two alignment strategies - (1) global alignment which is an extension to the domain adversarial training proposed by [GL14] to the segmentation problem and (2) local alignment which aligns class specific statistics by formulating it as a multiple instance learning problem. Curriculum domain adaptation [ZDG17] on the other hand proposes curriculum-style learning approach where the easy task of estimating global label distributions over images and local distributions over landmark super-pixels is learnt first. The segmentation network is then trained so that the target label distribution follow these inferred label properties.

One possible direction to address the domain adaptation problem is to employ style transfer or cross domain mapping networks to stylize the source domain images as target and train the segmentation models in this stylized space. Hence, we discuss some recent work related to the style transfer and unpaired image translation tasks. The popular work of Gatys *et al.* [GEB15] introduced an optimization scheme involving backpropagation for performing content preserving style transfer, while Johnson *et al.* [JAF16] proposed a feed-forward method for the same. CycleGAN [ZPIE17] performs unpaired image-to-image translation by employing adversarial losses and cycle consistency losses. In our experiments, we compare our approach to some of these style-transfer based data augmentation schemes.

6.3 Method

In this section, we provide a formal treatment of the proposed approach and explain in detail our iterative optimization procedure. Let $X \in \mathbb{R}^{M \times N \times C}$ be an arbitrary input image (with C channels) and $Y \in \mathbb{R}^{M \times N}$ be the corresponding label map. Given an input X, we denote the output of a CNN as $\hat{Y} \in \mathbb{R}^{M \times N \times N_c}$, where N_c is the number of classes. $\hat{Y}_{ij} \in \mathbb{R}^{N_c}$ is a vector representing the class probability distribution at pixel location (i, j) output by the CNN. The source(s) or target (t) domains are denoted by a superscript such as X^s or X^t. The loss function that is primarily used is the pixelwise cross-entropy loss denoted by $L_{ce}(\hat{Y}, Y)$ is given as follows:

$$L_{ce}(\hat{Y}, Y) = \sum_{i=1}^{M} \sum_{j=1}^{N} -\log \left(\frac{\exp(\hat{Y}_{ij}[Y_{ij}])}{\sum_{k=1}^{N_c} \exp(\hat{Y}_{ij}[k])} \right) \tag{6.1}$$

First, we provide an input-output description of the different network blocks in our pipeline. Next, we describe separately the treatment of source and target data, followed by a description of the different loss functions and the corresponding update steps. Finally, we motivate the design choices involved in the discriminator (D) architecture.

6.3.1 Description of network blocks

Our training procedure involves alternatively optimizing the following network blocks:

- The base network, whose architecture is similar to a pre-trained model such as VGG-16, is split into two parts: the embedding denoted by F and the pixel-

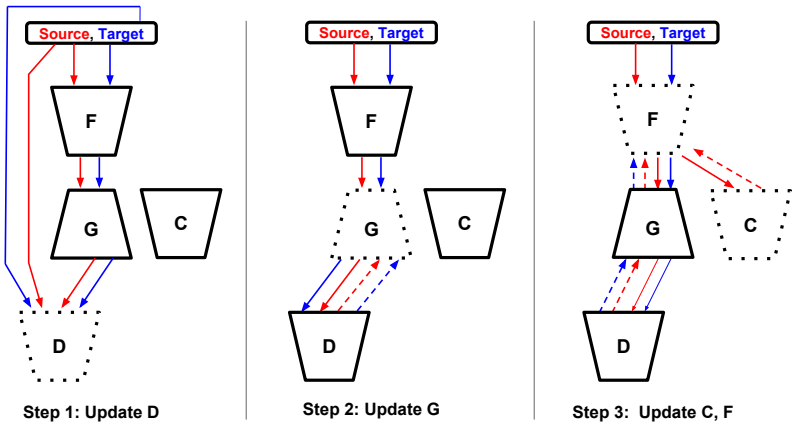


Figure 6.2: The directions of data flow *solid arrows* during the forward pass and gradient flow *dotted arrows* during the backward pass of our iterative update procedure. *Solid* blocks indicate that the block is frozen during that update step while *dotted* block indicate that it is being updated. Red denoted source information and Blue denotes target information.

wise classifier denoted by C. The output of C is a label map up-sampled to the same size as the input of F.

- The generator network (G) takes as input the learned embedding and reconstructs the RGB image.

- The discriminator network (D) performs two different tasks given an input: (a) It classifies the input as real or fake in a domain consistent manner (b) It performs a pixel-wise labeling task similar to the C network. Note that (b) is active only for source data since target data does not have any labels during training.

6.3.2 Treatment of source and target data

Given a source image and label pair $\{X^s, Y^s\}$ as input, we begin by extracting a feature representation using the F network. The classifier C takes the embedding $F(X^s)$ as input and produces an image-sized label map $\hat{Y}^s$. The generator G reconstructs the source input X^s conditioned on the embedding. Following recent successful works on image generation, we do not explicitly concatenate the generator input with a random noise vector but instead use dropout layers throughout the G network. As shown in Figure 6.3, D performs two tasks: (1) Distinguishing the real source input and generated source image as source-real/source-fake (2) producing a pixel-wise label map of the generated source image.

Given a target input X^t, the generator network G takes the target embedding from F as input and reconstructs the target image. Similar to the previous case, D is trained to distinguish between real target data (target-real) and the generated target images from G (target-fake). However, different from the previous case, D performs only a single task i.e. it classifies the target input as target-real/target-fake. Since the target data does not have any labels during training, the classifier network C is not active when the system is presented with target inputs.

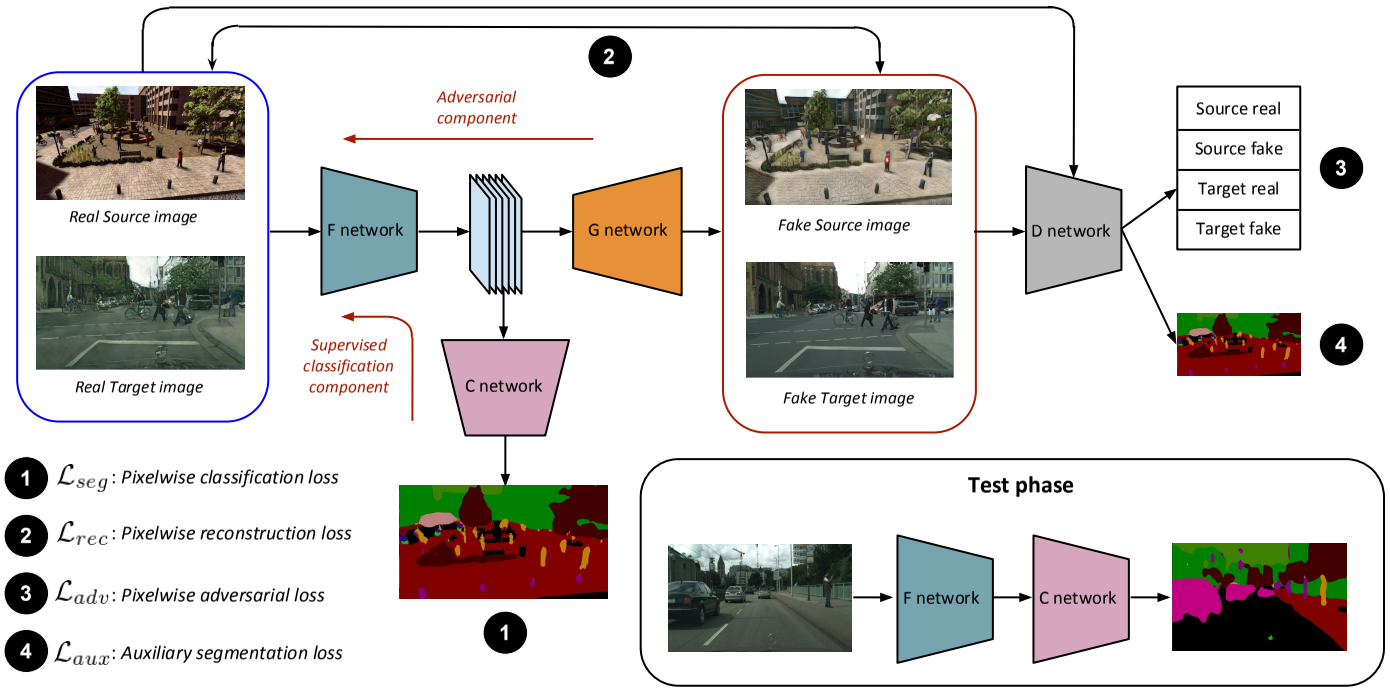


Figure 6.3: During training, the F and C networks are trained jointly with the adversarial framework(G-D pair). F is updated using a combination of supervised loss and an adversarial component. In the bottom right, we show the test time usage. Only the F and C network blocks are used. There is no additional overhead during evaluation compared to the base model.

6.3.3 Iterative optimization

Fig. 6.3 shows various losses used in our method. We begin by describing these losses, and then describe our iterative optimization approach.

The different adversarial losses used to train our models are shown in Table. 6.1. In addition to these adversarial losses, we use the following losses: (1) $\mathcal{L}_{seg}$ and $\mathcal{L}_{aux}$ - pixel-wise cross entropy loss used in standard segmentation networks such as in FCN and (2) $\mathcal{L}_{rec}$ - L_1 loss between input and reconstructed images.

Type	Variants	Description
Within-domain	$\mathcal{L}^s_{adv,D}$	Classify real source input as *src-real*; fake source input as *src-fake*
	$\mathcal{L}^s_{adv,G}$	Classify fake source input as *src-real*
	$\mathcal{L}^t_{adv,D}$	Classify real target input as *tgt-real*; fake target input as *tgt-fake*
	$\mathcal{L}^t_{adv,G}$	Classify fake target input as *tgt-real*
Cross-domain	$\mathcal{L}^s_{adv,F}$	Classify fake source input as real target (*tgt-real*)
	$\mathcal{L}^t_{adv,F}$	Classify fake target input as real source (*src-real*)

Table 6.1: Within-domain and Cross-domain adversarial losses that are used to update our networks during training. G and D networks are updated using only the within-domain losses while F is updated only using the cross domain loss. All these adversarial losses originate from the D network. $\mathcal{L}_{adv,X}$ implies that the gradients from the loss function L are used to update X only, while the other networks are held fixed.

The directions of flow of information across different network blocks are listed in Figure 6.2. In each iteration, a randomly sampled triplet (X^s, Y^s, X^t) is provided to the system. Then, the network blocks are updated iteratively in the following

order:

(1) D-update: For source inputs, D is updated using a combination of within-domain adversarial loss $\mathcal{L}^s_{adv,D}$ and auxiliary classification loss $\mathcal{L}^s_{aux}$. For target inputs, it is updated using only the adversarial loss $\mathcal{L}^t_{adv,D}$. The overall loss $\mathcal{L}_D$ is given by $\mathcal{L}_D = \mathcal{L}^s_{adv,D} + \mathcal{L}^t_{adv,D} + \mathcal{L}^s_{aux}$. The adversarial loss terms for D takes the following form:

$$\begin{aligned}\mathcal{L}^s_{adv,D} &= \min_D \; L_{ce}(D(X^s), Y_{src-real}) + L_{ce}(D(G(F(X^s)), Y_{src-fake}) \\ \mathcal{L}^t_{adv,D} &= \min_D \; L_{ce}(D(X^t), Y_{tgt-real}) + L_{ce}(D(G(F(X^t)), Y_{tgt-fake})\end{aligned} \tag{6.2}$$

where $Y_{src-real}$ and $Y_{src-fake}$ are spatial maps where each pixel is an indicator variable for the classes *src-real* and *src-fake* respectively. Similar explanations hold for the target domain terms. The auxiliary loss is a traditional cross entropy loss applied pixelwise as explained earlier: $\mathcal{L}^s_{aux} = L_{ce}(D(G(F(X^s)), Y)$. Note that the auxiliary loss is applied only for the generated source data and the label Y is same as the ground truth class label map. Contrary to traditional GAN approaches, the discriminator output is a spatial map where each pixel is classified as belonging to several classes. Specific details on the design choice of D is described later.

(2) G-update: In this step, the generator is updated using a combination of an adversarial loss $\mathcal{L}^s_{adv,G} + \mathcal{L}^t_{adv,G}$ intended to fool D and a reconstruction loss $\mathcal{L}_{rec}$. The adversarial loss encourages realistic output from the generator. The pixelwise L_1 loss is crucial to ensure image fidelity between the generator outputs and the corresponding input images. The overall generator loss is given as: $\mathcal{L}_G = \mathcal{L}^s_{adv,G} +$

$\mathcal{L}^t_{adv,G} + \mathcal{L}^s_{rec} + \mathcal{L}^t_{rec}$. The adversarial loss encourages the generator to produce realistic images by obtaining feedback from the discriminator:

$$\mathcal{L}^s_{adv,G} = \min_G L_{ce}(D(G(F(X^s)), Y_{src-real})$$
$$\mathcal{L}^t_{adv,G} = \min_G L_{ce}(D(G(F(X^t)), Y_{tgt-real}) \tag{6.3}$$

Note that the real/fake labels are flipped compared to the discriminator update in Eq. 6.2 and that D and F networks are held fixed during this step.

(3) F-update: The update to the F network is the critical aspect of our framework where the notion of domain shift is captured. The parameters of F are updated using a combination of several loss terms: $\mathcal{L}_F = \mathcal{L}_{seg} + \alpha\,\mathcal{L}^s_{aux} + \beta\,(\mathcal{L}^s_{adv,F} + \mathcal{L}^t_{adv,F})$. As illustrated in Table 6.1, the adversarial loss terms used to update F account for the domain adaptation. More specifically, the iterative updates described here can be considered as a min-max game between the F and the G-D networks. During the D update step discussed earlier, the adversarial loss branch of D learns to classify the input images as real or fake in a domain consistent manner. To update F, we use the gradients from D that lead to a reversal in domain classification, i.e. for source embeddings, we use gradients from D corresponding to classifying those embeddings as from target domain ($\mathcal{L}^s_{adv,F}$) and for target embeddings, we use gradients from D corresponding to classifying those embeddings as from source domain ($\mathcal{L}^t_{adv,F}$). Note that, this is similar to the min-max game between the G-D pair, except in this case, the competition is between classifying the generated image as from source/target domains instead of them being real/fake. The cross domain nature of adversarial

losses can be clearly observed from their formulations:

$$\mathcal{L}^{s}_{adv,F} = \min_{F} L_{ce}(D(G(F(X^{s})), Y_{tgt-real})$$
$$\mathcal{L}^{t}_{adv,F} = \min_{F} L_{ce}(D(G(F(X^{t})), Y_{src-real}) \tag{6.4}$$

Compared to Eq. 6.3, the domain labels are flipped during the F-update. This forces the embedding to move closer for source and target inputs. But this alone does not ensure class consistency, i.e. the source and target embeddings can be projected into a joint space where source classes do not necessarily correspond to the same target classes. To avoid this, the auxiliary loss term acts as a regularizer ensuring that the common subspace is learnt in a class consistent manner. Quantitative evidence of this claim can be found in the section on ablation experiments.

6.3.4 Motivating design choice of D

- In traditional GANs that are derived from the DCGAN [RMC15] implementations, the output of the discriminator is a single scalar indicating the probability of the input being fake or drawn from an underlying data distribution. Recent works on image generation have utilized the idea of *Patch* discriminator in which the output is a two dimensional feature map where each pixel carries a real/fake probability. This results in significant improvement in the visual quality of their generator reconstructions. We extend this idea to our setting by using a variant of the *Patch* discriminator, where each pixel in the output map indicates real/fake probabilities across source and target domains hence resulting in four classes per pixel: *src-real*, *src-fake*, *tgt-real*, *tgt-fake*.

- In general, GANs are hard to train on tasks which involve realistic images of a larger scale. One promising approach to training stable generative models with the GAN framework is the Auxiliary Classifier GAN (AC-GAN) approach by Odena *et al.* where they show that by conditioning G during training and adding an auxiliary classification loss to D, they can realize a more stable GAN training and even generate large scale images. Inspired by their results on image classification, we extend their idea to the segmentation problem by employing an auxiliary pixel-wise labeling loss to the D network.

Both these components prove crucial to our performance. The ablation study performed in Section 6.5.3 shows the effect of the above design choices on the final performance. Specific details about the architectures of these network blocks can be found in the supplementary material.

6.4 Experiments and Results

In this section, we provide a quantitative evaluation of our method by performing experiments on benchmark datasets. We consider two challenging synthetic datasets available for semantic segmentation: SYNTHIA and GTA-5. SYNTHIA [RSM+16] is a large dataset of photo-realistic frames rendered from a virtual city with precise pixel-level semantic annotations. Following previous works ([HWYD16], [ZDG17]), we use the SYNTHIA-RAND-CITYSCAPES subset that contains 9400 images with annotations that are compatible with cityscapes. GTA-5 is another large-scale dataset containing 24966 labeled images. The dataset was

Algorithm 2 Iterative training procedure of our approach

training iterations = N
for t in 1:N **do**
    Sample k images with labels from source domain **S**: $\{\mathrm{S}_i, \mathrm{y}_i\}_{i=1}^{k}$
    Let $\mathrm{s}_i = F(\mathrm{S}_i)$ be the embeddings computed for the source images.
    Sample k images from target domain **T** : $\{\mathrm{T}_i\}_{i=1}^{k}$
    Let $\mathrm{t}_i = F(\mathrm{T}_i)$ be the embeddings computed for the target images.
    Let s_{g_i} and t_{g_i} be the inputs to the generator.
    Update discriminator using the following objectives:

$$\mathcal{L}_D = \mathcal{L}^s_{adv,D} + \mathcal{L}^t_{adv,D} + \mathcal{L}^s_{aux} \tag{6.5}$$

    The generator is updated using a combination of adversarial loss and reconstruction loss for both source and target inputs.

$$\mathcal{L}_G = \mathcal{L}^s_{adv,G} + \mathcal{L}^t_{adv,G} + \mathcal{L}^s_{rec} + \mathcal{L}^t_{rec} \tag{6.6}$$

    Update the embedding F using a linear combination of the adversarial loss and classification loss. Update the classifier C for the source data using the cross entropy loss function.

$$\mathcal{L}_F = \mathcal{L}_{seg} + \alpha\, \mathcal{L}^s_{aux} + \beta\, (\mathcal{L}^s_{adv,F} + \mathcal{L}^t_{adv,F}) \tag{6.7}$$

end for

curated by Richter *et al.* [RVRK16] and is generated by extracting frames from the computer game Grand Theft Auto V.

We used CITYSCAPES [COR+16] as our real dataset. This dataset contains urban street images collected from a moving vehicle captured in 50 cities around Germany and neighboring countries. The dataset comes with 5000 annotated images split into three sets - 2975 images in the *train* set, 500 images in the *val* set and 1595 images in the *test* set. In all our experiments, for training our models we used labeled SYNTHIA or GTA-5 dataset as our source domain and unlabeled CITYSCAPES *train* set as our target domain. We compared the proposed approach with the only two contemporary methods that address this problem: FCN in the

wild [HWYD16] and Curriculum Domain adaptation [ZDG17]. Following these approaches, we designate the 500 images from CITYSCAPES *val* as our test set.

Architecture In all our experiments, we used FCN-8s as our base network. The weights of this network were initialized with the weights of the VGG-16 [SZ14a] model trained on Imagenet [KSH12c]. The architectures we used for D and G networks along with the hyper-parameter settings are described in the supplementary material.

Implementation details In all our experiments, images were resized and cropped to 1024×512. We trained our model for $100,000$ iterations using Adam solver [KB14] with a batch size of 1. Learning rate of 10^{-5} was used for F and C networks, and 2×10^{-4} for G and D networks. While evaluating on CITYSCAPES dataset whose images and ground truth annotations are of size 2048×1024, we first produce our predictions on the 1024×512 sized image and then upsample our predictions by a factor of 2 to get the final label map, which is used for evaluation. We will make our models and code publicly available.

6.4.1 SYNTHIA → CITYSCAPES

In this experiment, we use the SYNTHIA dataset as our source domain, and CITYSCAPES as our target domain. We randomly pick 100 images from the 9400 labeled images of SYNTHIA dataset and use it for validation purposes, the rest of the images are used for training. We use the unlabeled images corresponding to the CITYSCAPES *train* set for training our model. In order to ensure fairness of

experimental results, we followed the exact evaluation protocol as specified by the previous works ([HWYD16], [ZDG17]): The 16 common classes between SYNTHIA and CITYSCAPES are chosen used as our labels. The predictions corresponding to the other classes are treated as belonging to void class, and not backpropagated during training. The 16 classes are: sky, building, road, sidewalk, fence, vegetation, pole, car, traffic sign, person, bicycle, motorcycle, traffic light, bus, wall, and rider.

Table 6.7a reports the performance of our method in comparison with [HWYD16] and [ZDG17]. The source-only model which corresponds to the no adaptation case i.e. training only using the source domain data achieves a mean IOU of 25.7. The target-only values denote the performance obtained by a model trained using CITYSCAPES *train* set (supervised training), and they serve as a crude upper bound to the domain adaptation performance. These values were included to put in perspective the performance gains obtained by the proposed approach. We observe that our method achieves a mean IOU of 34.8, thereby improving the baseline by **9.1 points**, thus resulting in a higher performance improvement compared to other reported methods.

6.4.2 GTA5 → CITYSCAPES

In this experiment, we adapt from the GTA-5 dataset to the CITYSAPES dataset. We randomly pick 1000 images from the 24966 labeled images of GTA-5 dataset and use it for validation purpose and use the rest of the images for training. We use the unlabeled images corresponding to the CITYSCAPES *train* set for training our model. In order to ensure fairness of experimental results, we followed the

Method	Base n/w	road	sidewalk	bldg	wall	fence	pole	t light	t sign	veg	sky	person	rider	car	bus	mbike	bike	mIOU	mIOU gain
Source only [HWYD16]	Dilation-Frontend	6.4	17.7	29.7	1.2	0.0	15.1	0.0	7.2	30.3	66.8	51.1	1.5	47.3	3.9	0.1	0.0	17.4	
FCN wild [HWYD16]	[YK16]	11.5	19.6	30.8	4.4	0.0	20.3	0.1	11.7	42.3	68.7	51.2	3.8	54.0	3.2	0.2	0.6	20.2	2.8
Source only [ZDG17]	FCN8s-VGG16	5.6	11.2	59.6	8.0	0.5	21.5	8.0	5.3	72.4	75.6	35.1	9.0	23.6	4.5	0.5	18.0	22.0	
Curr. DA [ZDG17]	[LSD15]	65.2	26.1	74.9	0.1	0.5	10.7	3.5	3.0	76.1	70.6	47.1	8.2	43.2	20.7	0.7	13.1	29.0	7.0
Ours - Source only	FCN8s-VGG16	24.1	19.1	68.5	0.9	0.3	16.4	5.7	10.8	75.2	76.3	43.2	15.2	26.7	15.0	5.9	8.5	25.7	
Ours - Adapted	[LSD15]	79.1	31.1	77.1	3.0	0.2	22.8	6.6	15.2	77.4	78.9	47.0	14.8	67.5	16.3	6.9	13.0	**34.8**	**9.1**
Target-only	FCN8s-VGG16	96.5	74.6	86.1	37.1	33.2	30.2	39.7	51.6	87.3	90.4	60.1	31.7	88.4	52.3	33.6	59.1	59.5	-

(a) SYNTHIA → CITYSCAPES

Method	Base n/w	road	sidewalk	bldg	wall	fence	pole	t light	t sign	veg	terrain	sky	person	rider	car	truck	bus	train	mbike	bike	mIOU	mIOU gain
Source only [HWYD16]	Dilation-Frontend	31.9	18.9	47.7	7.4	3.1	16.0	10.4	1.0	76.5	13.0	58.9	36.0	1.0	67.1	9.5	3.7	0.0	0.0	0.0	21.2	
FCN wild [HWYD16]	[YK16]	70.4	32.4	62.1	14.9	5.4	10.9	14.2	2.7	79.2	21.3	64.6	44.1	4.2	70.4	8.0	7.3	0.0	3.5	0.0	27.1	5.9
Source only [ZDG17]	FCN8s-VGG16	18.1	6.8	64.1	7.3	8.7	21.0	14.9	16.8	45.9	2.4	64.4	41.6	17.5	55.3	8.4	5.0	6.9	4.3	13.8	22.3	
Curr. DA [ZDG17]	[LSD15]	74.9	22.0	71.7	6.0	11.9	8.4	16.3	11.1	75.7	13.3	66.5	38.0	9.3	55.2	18.8	18.9	0.0	16.8	16.6	28.9	6.6
Ours - Source only	FCN8s-VGG16	73.5	21.3	72.3	18.9	14.3	12.5	15.1	5.3	77.2	17.4	64.3	43.7	12.8	75.4	24.8	7.8	0.0	4.9	1.8	29.6	
Ours - Adapted	[LSD15]	88.0	30.5	78.6	25.2	23.5	16.7	23.5	11.6	78.7	27.2	71.9	51.3	19.5	80.4	19.8	18.3	0.9	20.8	18.4	**37.1**	**7.5**
Target-only	FCN8s-VGG16	96.5	74.6	86.1	37.1	33.2	30.2	39.7	51.6	87.3	52.6	90.4	60.1	31.7	88.4	54.9	52.3	34.7	33.6	59.1	57.6	-

(b) GTA5 → CITYSCAPES

Table 6.2: Results of Semantic Segmentation by adapting from (a) SYTNHIA to CITYSCAPES and (b) GTA-5 to CITYSCAPES. We compare with two approaches that use two different base networks. To obtain a fair idea about our performance gain, we compare with the Curriculum DA approach that uses the same base network as ours. The Target-only training procedure is the same for both the settings since in both cases the target domain is CITYSCAPES. However, the results in (a) are reported over the 16 common classes while the results in (b) are reported over all the 19 classes.

exact evaluation protocol as specified by the previous works ([HWYD16], [ZDG17]): we use 19 common classes between GTA-5 and CITYSCAPES as our labels. The results of this experiment are reported in Table. 6.7b. Similar to the previous experiment, our baseline performance (29.6) is higher than the performance reported in [HWYD16], due to difference in network architecture and experimental settings. On top of this, the proposed approach yields an improvement of 7.5 points to obtain a mIOU of **37.1**. This performance gain is higher than that achieved by the other compared approaches.

Note regarding different baselines: The baseline numbers reported by us do not match with the ones reported in [ZDG17] and [HWYD16] due to different experimental settings (this mismatch was also reported in [ZDG17]). However, we would like to point out that we improve over a stronger baseline compared to the other two methods in both our adaptation experiments. In addition, [ZDG17] uses additional data from PASCAL-CONTEXT [MCL+14b] dataset to obtain the superpixel segmentation. In contrast, our approach is a single stage end-to-end learning framework that does not use any additional data and yet obtains better performance improvement.

6.5 Discussion

In this section, we perform several exploratory studies to give more insight into the functionality and effectiveness of the proposed approach. similar to the previous section, all the evaluation results are reported on the CITYSCAPES *val*

set, unless specified otherwise. We denote this set as the *test set*. We would like to note that owing to space constraints, we have added example results such as label predictions and images sampled from generator network etc in the supplementary material.

6.5.1 Effect of Image Size

The datasets considered in this chapter consists of images of large resolution which is atleast twice larger than the most commonly used Segmentation benchmarks for CNNs i.e. PASCAL VOC (500×300) and MSCOCO (640×480). In this setting, it is instructive to understand the effect of image size on the performance of our algorithm both from a quantitative and computational perspective. Table 6.3 presents the results of our approach applied over three different image sizes along with the training and evaluation times. It should be noted that the Curriculum DA approach [ZDG17] used a resolution of 640×320. By comparing with our main results in Table 6.7a, we see that our approach provides a higher relative performance improvement over a similar baseline.

For computational efficiency, the remaining experiments in this section are run with the image size of 640×320.

6.5.2 Comparison with direct style transfer

Generative methods for style transfer have achieved a great amount of success in the recent past. A simple approach to performing domain adaptation is to use such approaches as a data augmentation method: transfer the images from

Table 6.3: Mean IoU values and computation times across different image size on the SYNTHIA → CITYSCAPES setting. The numbers in **bold** indicate the absolute improvement in performance over the *Source-only* baseline. The reported training and evaluation times are for the proposed approach and are averaged over training and evaluation runs.

Image size	512 × 256	640 × 320	1024 × 512
mIOU-*Source-only*	20.5	22.2	25.7
mIOU-*Ours*	29.3 (+**8.8**)	32.1 (+**9.9**)	34.8 (+**9.1**)
Train time (per image)	1.5s	2.1s	2.9s
Eval time (per image)	0.16s	0.19s	0.3s

the source domain to target domain and use the provided source ground truth to train a classifier on the combined source and target data. In order to compare the proposed approach with this direct data augmentation procedure, we used a state of the art generative approach (CycleGAN [ZPIE17]) to transfer images from source domain to target domain. As can be observed from the results, using generative approaches solely as a data augmentation method provides a relatively small improvement over the source-only baseline and clearly suboptimal compared to the proposed approach. By augmenting the feature learning process with gradients from the G-D pair, our method achieves superior performance and is more reliable in cases where approaches based on pure generation may fail to improve. This experiment highlights the difficulty in achieving domain adaptation by performing a direct style transfer.

Table 6.4: Comparison of semantic segmentation performance on SYNTHIA → CITYSCAPES setting when using a GAN based approach as data augmentation. We use CycleGAN [ZPIE17] as the cross domain generation procedure.

Method	mean IoU
Source-only	22.2
Source + CycleGAN-augmented	25.6
Ours	**32.1**

6.5.3 Component-wise ablation

In this experiment, we show how each component in our loss function affects the final performance. We consider the following cases: (a) Ours(full): the full implementation of our approach (b) Ours w/o auxiliary pixel-wise loss: Here, the output of the D network is a single branch classifying the input as real/fake. This corresponds to $\alpha = 0$ in the F-update step. Note that, setting both α and β as zero corresponds to the source-only setting in our experiments. Setting only $\beta = 0$ does not improve over the source-only baseline as there is no cross domain adversarial loss. (c) Ours w/o *Patch* discriminator: Instead of using the D network as a *Patch* discriminator, we used a regular GAN-like discriminator where the output is a 4-D probability vector that the input image belongs to one of the four classes - *src-real*, *src-fake*, *tgt-real* and *tgt-fake*. (d) Feature space based D: In this setting, we remove the *G-D* networks and apply an adversarial loss directly on the embedding. This is similar to the global alignment setting in the FCN-in-the-wild approach [HWYD16].

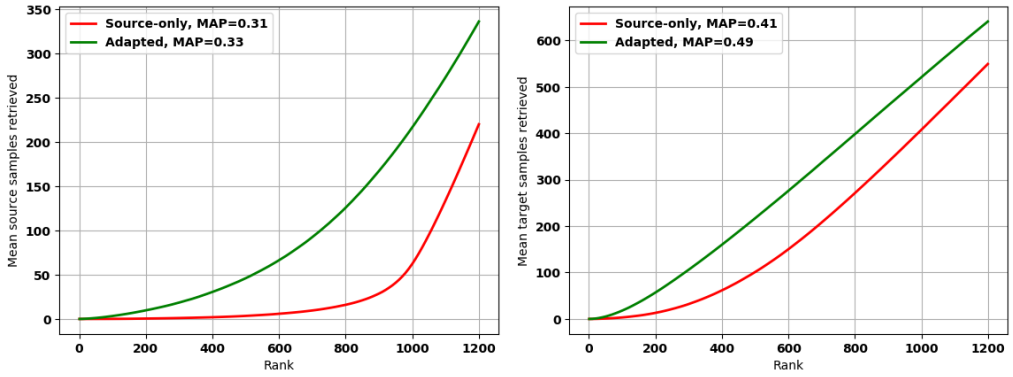
The mean IoU results on the *test set* are shown in Table. 6.5. It can be observed that each component is very important to obtain the full improvement in performance.

Table 6.5: Ablation study showing the effect of each component on the final performance of our approach on the SYNTHIA → CITYSCAPES setting

Method	mean IoU
Source-only	22.2
Feature space based D	25.3
Ours w/o *Patch* Discriminator	28.3
Ours w/o auxiliary loss ($\alpha = 0$)	29.2
Ours	**32.1**

6.5.4 Cross Domain Retrieval

A crucial aspect of domain adaptation is in finding good measures of domain discrepancy that provide a good illustration of the domain shift. While there exist several classical measures such as $\mathcal{A}$-distance [BDBCP07] and MMD [GTX11] for the case of image classification, the extension of such measures for a pixel-wise problem such as semantic segmentation is non-trivial. In this section, we devise a simple experiment in order to illustrate how the proposed approach brings source and target distributions closer in the learnt embedding space. We start with the last layer of the F network, which we label as the embedding layer, whose output is a spatial feature map. We perform an average pooling to reduce this spatial map



(a) Target → Source, $|B_k|$ (vs) k *(b)* Source → Target, $|A_k|$ (vs) k

Figure 6.4: Illustration of Domain Adaptation achieved by the proposed approach. The plot compares the average number of retrieved sampled for the cross domain retrieval task described in Section 6.5.4 between the source-only model and the model adapted using the proposed approach. Target → Source implies that the query set used belongs to target domain (Q_T) and items queried for from the set X belong to the source domain and vice-versa for Source → Target. In general, the values plotted on the y-axis corresponds to the number of samples retrieved from the set X that belong to the opposite domain as to that of the query set.

to a 4096 dimensional feature descriptor for each input image.

We begin the cross domain retrieval task by choosing a pool of $N = N_{src} + N_{tgt}$ images from the combined source and target training set. Let X denote these set of images and F_X denote the set of the feature descriptors computed for X. Then, we choose two query sets, one consisting of source images (S) and the other consisting of target images (T), each disjoint with X. Let the corresponding feature sets be denoted as Q_S and Q_T. We retrieve k-NN lists for each item in the query set from the

combined feature set F_X. For each query point in Q_S, we count the number of target samples retrieved in the corresponding k-NN list. $|A_k|$ indicates the average number of target samples retrieved over the entire source query set Q_S. For each query point in Q_T, we count the number of source samples retrieved in the corresponding k-NN list. $|B_k|$ indicates the average number of source samples retrieved over the entire target query set Q_T. We used cosine similarity as a metric to compute the k-NN lists. If more target samples are retrieved for a source query point (and vice-versa), it suggests that source and target distributions are aligned well in the feature space.

For this experiment, the sizes of query sets and the feature set F_X are as follows: $N_{src} = N_{tgt} = 1000$, $|Q_S| = 1000$, $|Q_T| = 1000$. The mean average precision (mAP) was computed across the entire query sets for the respective cross domain tasks. Figure 6.4 shows the plot of the quantities $|A_k|$ (Fig.6.4b) and $|B_k|$ (Fig.6.4a) for a range of values of k. It can be observed from the plots in both the tasks that for any given rank k, the number of cross domain samples retrieved by the adapted model is higher than the source-only model. This effect becomes more clear as k increases. This observation is supported by better mAP values for the adapted model as shown in Figure 6.4. While this by itself is not a sufficient condition for better segmentation performance, however this along with the results from Table 6.7 imply that the proposed approach performs domain adaptation in a meaningful manner. Owing to the difficulty in visualizing the mapping learned for segmentation tasks, a cross domain retrieval experiment can be seen as a reasonable measure of how domain gap is reduced in the feature space.

6.5.5 Generalization to unseen domains

Table 6.6: Mean IoU segmentation performance measured on a third unseen domain (CamVid dataset) for the models corresponding to the SYNTHIA → CITYSCAPES setting

Method	mean IoU
Source-only	36.1
Ours	**44.4**

A desirable characteristic of any domain adaptation algorithm is domain generalization i.e. improving performance over domains that are not seen during training. To test the generalization capability of the proposed approach, we test the model trained for the SYNTHIA → CITYSCAPES setting on the CamVid dataset [BFC09]. We choose to evaluate our models on the 10 common classes among the CamVid, SYNTHIA and CITYSCAPES datasets. To begin with, we would like to note that for the results presented in Table 6.7 for the SYNTHIA → CITYSCAPES setting, 16 common classes among the two datasets were chosen following previous works. Now, for the CamVid experiment, we choose 10 among these 16 classes which are common with the CamVid dataset. They are the following: *building*, *vegetation*, *t sign*, *sky*, *car*, *road*, *person*, *fence*, *pole*, *sidewalk*.

Table 6.6 shows the mean IoU values computed for the source-only baseline and the adapted model. The proposed approach yields a raw improvement of **8.3** points in performance which is a significant improvement considering the fact that

CamVid images are not seen by the adapted model during training. This experiment showcases the ability of the proposed approach to learn domain invariant representations in a generalized manner.

6.6 Results using DeepLab-ResNet-101 as base model

Method	Base n/w	road	sidewalk	bldg	wall	fence	pole	t light	t sign	veg	sky	person	rider	car	bus	mbike	bike	mIOU	mIOU gain
Ours - Source only	Deeplab-Resnet-101	23.3	15.8	62.6	0.1	0.01	19.7	1.1	5.8	67.6	83.2	50.6	7.1	23.1	16.7	1.2	8.0	24.1	
Ours - Adapted	Deeplab-Resnet-101	81.3	29.3	79.2	5.1	0.3	25.1	8.7	13.4	78.4	84.4	45.3	9.5	68.9	17.4	7.6	21.0	**35.9**	**11.8**

(a) SYNTHIA → CITYSCAPES

Method	Base n/w	road	sidewalk	bldg	wall	fence	pole	t light	t sign	veg	terrain	sky	person	rider	car	truck	bus	train	mbike	bike	mIOU	mIOU gain
Ours - Source only	Deeplab-Resnet-101	72.1	4.5	65.1	2.1	17.4	21.2	27.7	10.6	69.5	11.7	66.8	48.2	12.6	51.0	24.9	9.1	0.0	10.8	0.9	27.7	
Ours - Adapted	Deeplab-Resnet-101	85.4	26.6	79.0	22.0	25.5	29.3	35.1	17.9	76.5	19.9	73.8	55.7	17.9	66.7	24.7	25.3	2.2	20.5	1.1	**37.1**	**9.4**

(b) GTA5 → CITYSCAPES

Table 6.7: Results of Semantic Segmentation performance for the Deeplab-Resnet-101 base model by adapting from (a) SYTNHIA to CITYSCAPES and (b) GTA-5 to CITYSCAPES. Similar to the FCN-8s results presented in the main paper, the results in (a) are reported over the 16 common classes while the results in (b) are reported over all the 19 classes.

In the previous section, for fair comparison with previous works, we used FCN-8s as our base model. In this section, we present results of our approach using Deeplab Resnet-101 network as the base model. We used the single-scale version of the model that was pre-trained on the MS-COCO dataset.

6.7 Generator Visualizations

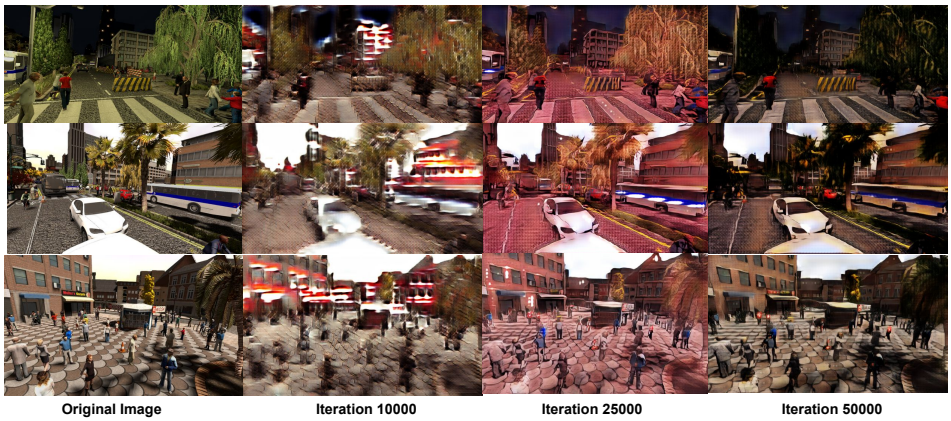


Figure 6.5: Querying the generator space for source images - Progress across iterations

Figure 6.6: Querying the generator space for target images - Progress across iterations

In figures 6.5 and 6.6, we present examples of the generator reconstructions of source and target images during the course of the training procedure. This provides a visual representation of the convergence of the algorithm. As the iterations progress, the generator quality increases thereby resulting in better knowledge transfer between source and target distributions.

6.8 Qualitative Comparison of Label predictions

The label map visualization of the segmentation results obtained by the baseline model and our adapted model are shown in the Figure. 6.7. We observed that our adapted model improves the quality of the label map predictions significantly compared to the source-only model. This improvement is predominant for classes that occupy the major portion of the image like *road* and *car*.

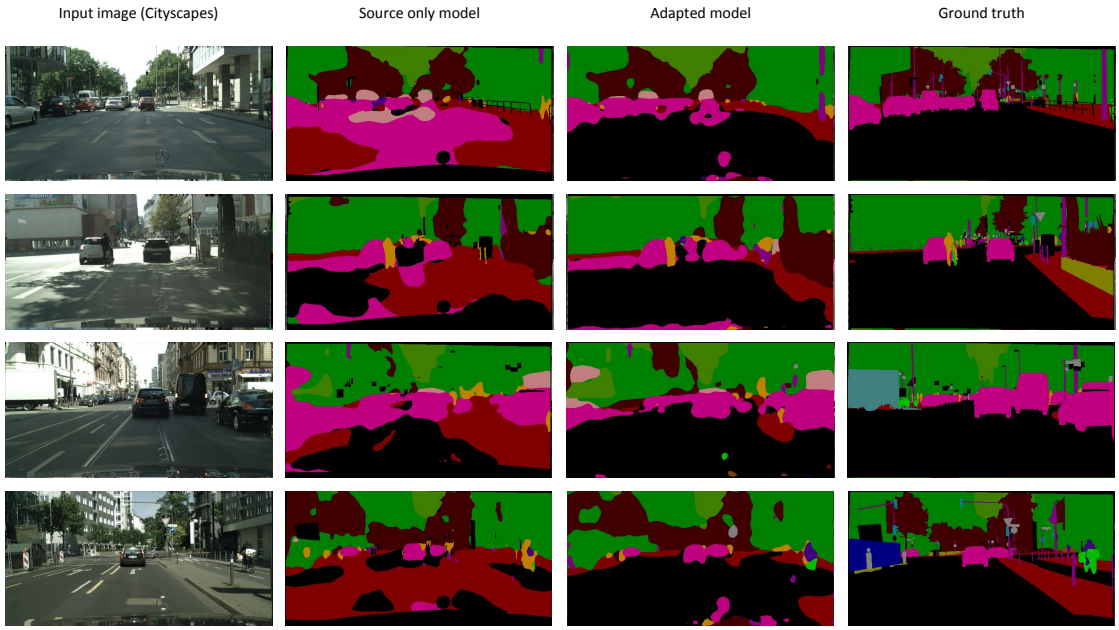


Figure 6.7: Visualization of label map predictions for SYNTHIA → CITYSCAPES experiment. In each row, the first column corresponds to the input image sampled from the target domain (Cityscapes). The second and the third column corresponds to the segmentation results of the baseline model (source-only model) and our adapted model respectively. The last column corresponds to the ground truth

6.9 Architecture and Hyperparameters

The details of the architectures used in our experiments are shown in the Figure. 6.8. As described in Section 3 of the main paper, the entire pipeline of our approach consists of 4 networks - F, C, G and D networks. We perform experiments with two architectures for $F - C$ pair: FCN-8s and Deeplab-Resnet-101. For FCN-8s, we denote the network till $fc7$ layer as F network, and the final classification layers are treated as C network. For Deeplab-Resnet-101, the network till $res5c$ layer is denoted as F network, the rest of the layers are treated as C network.

The architectures of G and D networks are described in Figure. 6.8. The G network is a multi-stage network - it accepts inputs from intermediate layers of the F network to generate the reconstructed image. We observed that fusing information from the earlier layers of F network produced good quality generations compared to using only the response of the final layer. For both G and D networks, we use residual blocks inspired by the recent success of several generative models [32]

There are two hyper-parameters in our approach: α - the weight of auxiliary classification loss and β - the weight of adversarial component (Refer to Section 3 in the main paper). We observed that the network is not very sensitive to these parameters. We found that the parameter setting $\alpha = 0.1$ and $\beta = 0.1$ worked well across all settings, and used it for all our experiments.

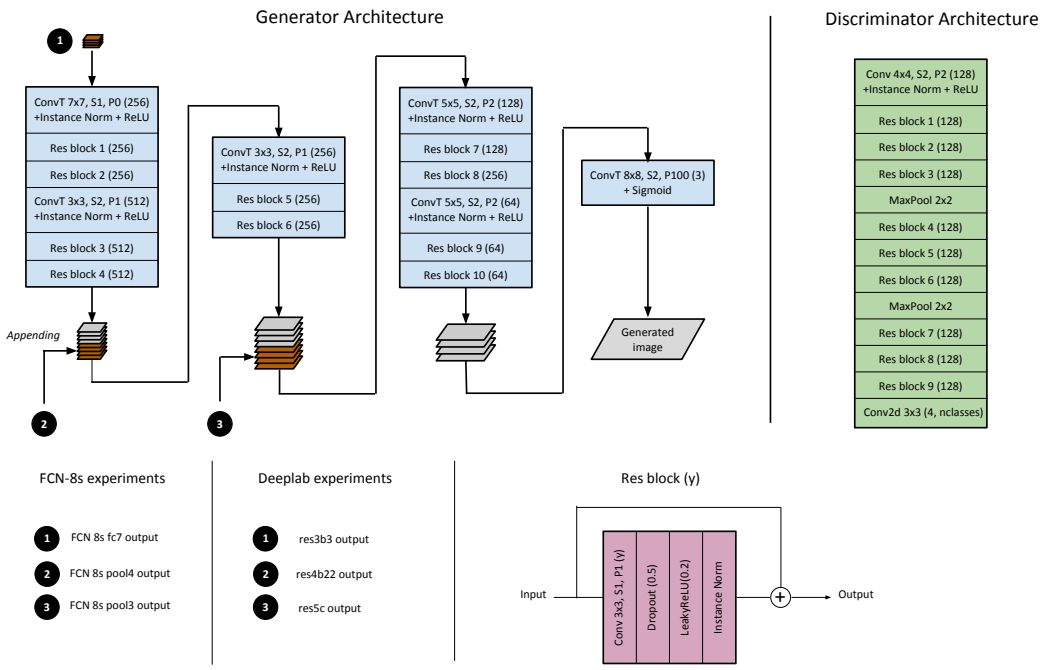


Figure 6.8: Details of the network architectures used in our experiments. Conv - Convolution layer, ConvT - Transposed convolution layer, S - stride, P - padding. For each Conv/ConvT layer, the numbers in the parenthesis denote the number of filters.

6.10 Conclusions

In this chapter, we have addressed the problem of performing semantic segmentation across different domains. In particular, we have considered a very hard case where abundant supervisory information is available for synthetic data (source) but no such information is available for real data (target). We proposed a joint adversarial approach that transfers the information of the target distribution to the learned embedding using a generator-discriminator pair. We have shown the superiority of our approach over existing methods that address this problem using experiments

on two large scale datasets thus demonstrating the generality and scalability of our training procedure. Furthermore, our approach has no extra computational overhead during evaluation, which is a critical aspect when deploying such methods in practice. As future work, we can extend this approach to explicitly incorporate geometric constraints accounting for perspective variations and to adapt over temporal inputs such as videos across different domains.

Chapter 7. Conclusions and Directions for Future Work

7.1 Summary

In this dissertation, we addressed the properties of deep neural networks, which have recently become the workhorses for several machine learning and computer vision tasks. We highlighted their limitations from two broad perspectives: (1) Robustness to perturbations of the input (2) Inability to generalize across domain shift. In the first part, we studied robustness as both an external (as presented by naturally occurring data distortions) and internal(adversarial perturbations of the feature space and thereby the input) artifacts. We extended the concept of adversarial perturbations to a larger class of computer vision problems and showed examples of how they could be used to aid existing systems for semantic segmentation. In the second part, we addressed the problem of domain adaptation in a large scale setting.

In chapter 3, we presented a novel approach using convolutional neural networks to address the problem of face verification under unconstrained environments. o approach consists to two components: a simple deep network architecture and a training scheme which has faster convergence over a relatively diverse dataset, a fast embedding approach that projects the deep features to a more discriminative

low dimensional space in order to improve face verification performance. We presented experimental results on challenging face datasets such as IJB-A and CFP and demonstrated the robustness of the deep features to challenges including age, pose, blur and clutter by performing simple subject specific clustering experiments on LFW and IJB-A datasets. This highlighted the fact that even a deep neural network trained on a large enough dataset learns a lot of redundant information that are detrimental for good generalization performance. By reducing the sensitivity of the deep features on the dimensions as dictated by our triplet embedding approach, we showed that we could achieve better results on challenging unseen data.

In chapter 4, we addressed the perturbation analysis of segmentation networks. A class of perturbations were shown to exist for deep networks that were used for classification tasks. Previous works showed how to obtain these perturbations and demonstrated that they are largely adversarial. In our approach, we showed that for networks that perform explicit contextual modeling at the output, the effect of perturbations need not be always adversarial. Furthermore, we showed how to generated these guided perturbations in a reliable manner and experimentally showed that these are transferable across neural network architectures.

In chapter 5, we studied the effect of adversarial training as regularizer with a focus on very deep state of the art models. We proposed a novel regularization approach derived from a novel variant of traditional adversarial training. Sample gradients corresponding to images belonging to different classes compared to a given input can act as adversarial gradients when perturbed layerwise. We showed a faster way to realize this for the case of batch training. We compared the proposed ap-

proach with popular regularizers such as dropout and showed superior performance for both clean and adversarial data. We demonstrated that the proposed regularizer improves the clean data performance of even state of the art deep models.

Finally in chapter 6, we addressed the problem of domain shift in computer vision applications. We considered the case of the semantic segmentation tasks and a harder case of domain shift involving synthetic to real images. To adapt the learned representation to generalize across the labeled source and unlabeled target distributions, we formulated a minimax game between a generative adversarial network and a base feature network. We demonstrated significant improvement on the large scale task of semantic segmentation over existing approaches on this very hard problem. Our ablative experiments showed that as training progresses, the proposed training procedure results in a good mixing of the source and target distributions in the local neighborhood of the feature network, hence resulting in good domain adaptation performance.

7.2 Directions for Future Work

In this section, we describe some promising future directions that can stem from the different problems considered in this thesis. We provide a very brief outline of directions and preliminary results where applicable.

7.2.1 Embedding Videos using Triplet Constraints

The triplet probabilistic embedding approach discussed in chapter 3 was applied to facial datasets which consisted of static images. In the case of videos, the feature representation was pooled over successive frames such that the entire video was represented as a single feature. A more interesting direction would be to learn an attention mechanism in order to weight the different frames of the video in addition to learning the projection matrix. This can be done with the use of recurrent neural network models such as Lone Short Term Memory (LSTM) blocks in order to unroll videos of variable lengths.

7.2.2 Predictive Influence Estimation

Chapter 4 provided some key observations regarding the effect of simple gradient based perturbations on deep networks used for semantic segmentation tasks. From the perspective of robust statistics, a small perturbation resulting in a non-negligible change in a black box system indicates the unstable nature (topologically speaking, this relates to multiple sharp peaks) of the learned decision surface. Within the context of a pixelwise segmentation task, the effect of such perturbations on any given pixel can be considered as a notion of inertia for the feature representation of that pixel. This is illustrated in Figure 7.1. Pixels which are closer to the decision boundary are the ones whose class distribution will be greatly affected by the perturbations.

The effect of these perturbations on a given set of pixels can be quantified by

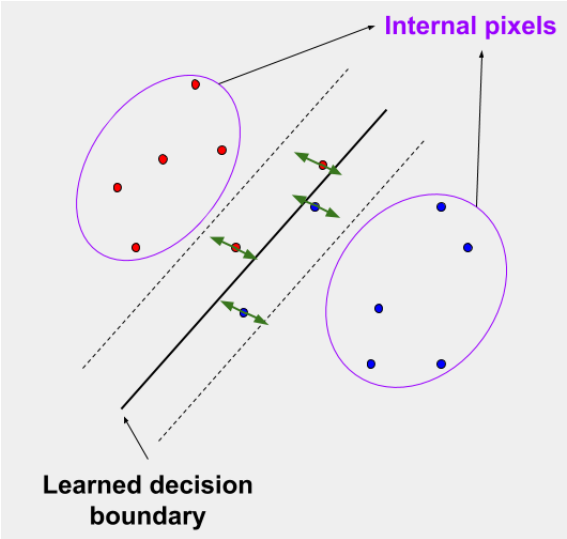


Figure 7.1: Illustration of the effect of the ϵ perturbations for a simple linear case. The feature representations of the pixels close to the decision boundary would change more than the pixels in the interior regions.

the divergence between the predicted probability distributions as a results of the perturbations. This can be concretely described as follows: Let $P(X, \epsilon)$ denote the perturbation process applied to input image X in the direction ϵ. The perturbation process described in Chapter 4 is the linear additive perturbation: $P(X, \epsilon) = X + \epsilon \nabla_X J(X, Y, \theta)$. Let $F_\theta(P(X, \epsilon))$ denote the predictive distribution of the network F with parameters θ. Then, a measure of the influence for the pixels of the image X is the divergence between the two sided perturbations: $D\left[F_\theta(P(X, -\epsilon)), F_\theta(P(X, \epsilon))\right]$, where D is an appropriate measure such as KL of the JSD divergence.

This can provide a metric to sample fewer pixels during training time as opposed to choosing the entire image for the purpose of label annotation, hence potentially resulting in a significant reduction in annotation time and effort. This represents uncertainty modeling and confidence intervals in the Bayesian sense. Relating

the effect of these perturbations to Bayesian inference is an interesting direction in its own right.

7.2.3 Adversarial regularization for non-image data

The regularization approach proposed in chapter 5 is a very general method, in that there is no assumption made about the type of data it is applied to. Deep networks that perform differentiable end to end training for non-image modalities such as speech and language are widely used in the machine learning community. A potential future avenue would be to apply the proposed layerwise adversarial regularization approach to networks that are used to train such non-image datasets.

7.2.4 Domain Adaptation with Temporal Sequences

In chapter 6, we described the problem of domain adaptation and proposed an approach based on generative adversarial networks to learn a common feature space where the distance between source and target distributions are minimized. The datasets that were used for the experiments contained static frames as images and video data was not available. A potential future direction would be to generate synthetic data in the form of videos and utilize it to perform domain adaptation. Figure 7.2 shows one way to utilize the temporal information in source data. The main difference here compared to the original approach is the use of a recurrent neural network which can aggregate information across different frames of the video. The task loss and the adversarial loss can then be applied on the aggregated information resulting in improved disambiguation of smaller objects such as pedestrians,

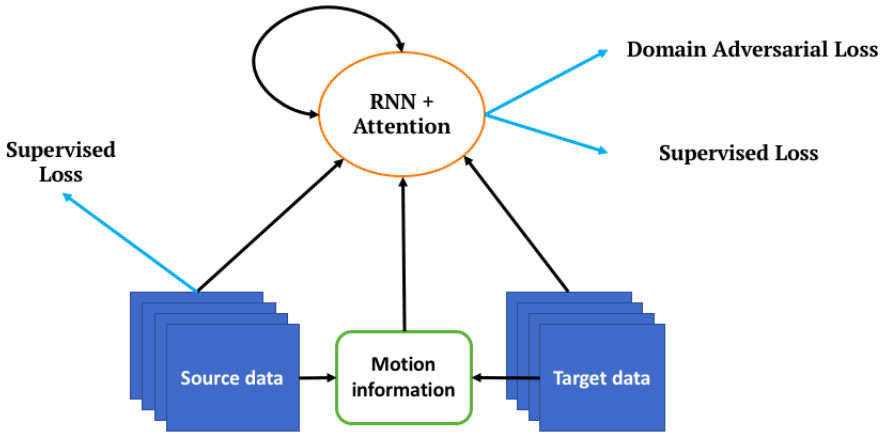


Figure 7.2: Training pipeline for Domain Adaptation over Video data

bicyclists etc. Motion features such as the estimated optical flow between successive frames can be provided as auxiliary input to the feature network.

7.2.5 Domain Generalization

The domain adaptation problem considered in chapter 6 assumes that the target data is available for the learning algorithm during training. In a real world setting, this is rarely the case. A more practical setting is shown in Figure 7.3 where labeled training data from multiple domains is available for the learning algorithm and the task is to learn a generalizable feature representation that should work on unseen data. The unseen data could be drawn from a similar representation compared to the source domains or from a completely novel distribution. This is a natural generalization of the domain adaptation problem to a more realistic yet significantly harder setting. A potential approach developed for this problem should teach the learner to generalize rather than to fit a given distribution. In other words, the network should be designed to fit novel and unseen data. One promising

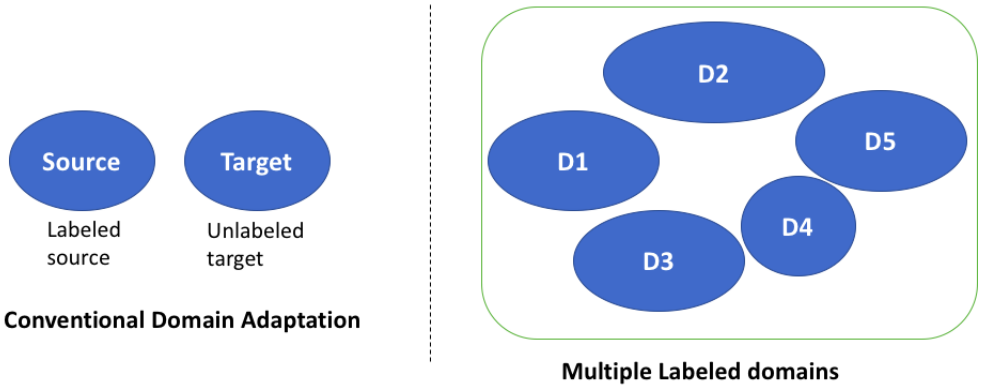


Figure 7.3: Illustration of the Domain Generalization problem where multiple labeled source domains are given during training and the objective is to find the most generalizable representation that works well on novel data.

direction to address this problem is to use recently proposed meta learning based approaches that involves hierarchical levels of learning, reminiscent of the concept of hyperpriors in Bayesian statistics.

Bibliography

[Ale17] Torch implementation of alexnet training on imagenet data. *https://github.com/soumith/imagenet-multiGPU.torch*, 2017.

[BDBCP07] Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. Analysis of representations for domain adaptation. In *Advances in neural information processing systems*, pages 137–144, 2007.

[BFC09] Gabriel J Brostow, Julien Fauqueur, and Roberto Cipolla. Semantic object classes in video: A high-definition ground truth database. *Pattern Recognition Letters*, 30(2):88–97, 2009.

[BSD+16] Konstantinos Bousmalis, Nathan Silberman, David Dohan, Dumitru Erhan, and Dilip Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. *arXiv preprint arXiv:1612.05424*, 2016.

[BTVG06] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006.

[CBP+16] Nate Crosswhite, Jeffrey Byrne, Omkar M. Parkhi, Chris Stauffer, Qiong Cao, and Andrew Zisserman. Template adaptation for face verification and identification. *arXiv preprint arXiv:1603.03958*, 2016.

[CMV+16] Nicholas Carlini, Pratyush Mishra, Tavish Vaidya, Yuankai Zhang, Micah Sherr, Clay Shields, David Wagner, and Wenchao Zhou. Hidden voice commands. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 513–530, Austin, TX, 2016. USENIX Association.

[COR+16] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[CPC15] Jun-Cheng Chen, Vishal M Patel, and Rama Chellappa. Unconstrained face verification using deep cnn features. *arXiv preprint arXiv:1508.01722*, 2015.

[CPK+14] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. *arXiv preprint arXiv:1412.7062*, 2014.

[CPK+16] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *arXiv preprint arXiv:1606.00915*, 2016.

[CSPC15] Jun-Cheng Chen, Swami Sankaranarayanan, Vishal M Patel, and Rama Chellappa. Unconstrained face verification using fisher vectors computed from frontalized faces. *BTAS*, 2015.

[DI07] Hal Daume III. Frustratingly easy domain adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, June 2007.

[dMH08] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

[Doe16] Carl Doersch. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, 2016.

[DT05] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[EEVG+15] Mark Everingham, SM Ali Eslami, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136, 2015.

[EGW+10] Mark Everingham, Luc Gool, Christopher K. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *Int. J. Comput. Vision*, 2010.

[FFF15] Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Analysis of classifiers' robustness to adversarial perturbations. *arXiv preprint arXiv:1502.02590*, 2015.

[FM82] Kunihiko Fukushima and Sei Miyake. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and cooperation in neural nets*, pages 267–285. Springer, 1982.

[FMF16] Alhussein Fawzi, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. Robustness of classifiers: from adversarial to random noise. *CoRR*, abs/1608.08967, 2016.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[GEB15] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. A neural algorithm of artistic style. *CoRR*, abs/1508.06576, 2015.

[GL14] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. *arXiv preprint arXiv:1409.7495*, 2014.

[GLC11] Raghuraman Gopalan, Ruonan Li, and Rama Chellappa. Domain adaptation for object recognition: An unsupervised approach. In *Proceedings of the 2011 International Conference on Computer Vision*, ICCV '11, 2011.

[GPAM+14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[GR14] Shixiang Gu and Luca Rigazio. Towards deep neural network architectures robust to adversarial examples. *arXiv preprint arXiv:1412.5068*, 2014.

[GSS14a] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[GSS14b] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[GSSG12] Boqing Gong, Yuan Shi, Fei Sha, and Kristen Grauman. Geodesic flow kernel for unsupervised domain adaptation. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2066–2073, 2012.

[GTX11] Bo Geng, Dacheng Tao, and Chao Xu. Daml: Domain adaptation metric learning. *IEEE Transactions on Image Processing*, 20(10):2980–2989, 2011.

[HAB+11] Bharath Hariharan, Pablo Arbeláez, Lubomir Bourdev, Subhransu Maji, and Jitendra Malik. Semantic contours from inverse detectors. In *2011 International Conference on Computer Vision*, pages 991–998. IEEE, 2011.

[HJN+11] Ling Huang, Anthony D Joseph, Blaine Nelson, Benjamin IP Rubinstein, and JD Tygar. Adversarial machine learning. In *Proceedings of the 4th ACM workshop on Security and artificial intelligence*, pages 43–58. ACM, 2011.

[Hop82] John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.

[HRBLM07] Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. (07-49), October 2007.

[HWYD16] Judy Hoffman, Dequan Wang, Fisher Yu, and Trevor Darrell. Fcns in the wild: Pixel-level adversarial and constraint-based adaptation. *CoRR*, abs/1612.02649, 2016.

[HZRS15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *arXiv preprint arXiv:1502.01852*, 2015.

[HZRS16a] 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.

[HZRS16b] 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*, 2016.

[IGC16] Yoshua Bengio Ian Goodfellow and Aaron Courville. *Deep Learning*. 2016. Book in preparation for MIT Press.

[IS15] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[JAF16] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II*, pages 694–711, 2016.

[JSD+14] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[Kar16] CS231n: Andrej Karpathy. CS231n Course Notes: Convolutional Neural Networks for Visual Recognition. `http://cs231n.stanford.edu/ /`, 2016. [Online; accessed 19-April-2016].

[KB14] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[KH09] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images, https://www.cs.toronto.edu/ kriz/cifar.html. 2009.

[Kri09] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[KSH12a] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[KSH12b] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[KSH12c] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 2012.

[KTB+15] Brendan F Klare, Emma Taborsky, Austin Blanton, Jordan Cheney, Kristen Allen, Patrick Grother, Alan Mah, Mark Burge, and Anil K Jain. Pushing the frontiers of unconstrained face detection and recognition: Iarpa janus benchmark a. *algorithms*, 13:4, 2015.

[Kul12] Brian Kulis. Metric learning: A survey. *Foundations and Trends in Machine Learning*, 5(4):287–364, 2012.

[KW13] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[L+14] Tsung-Yi Lin et al. Microsoft coco: Common objects in context. In *ECCV*, 2014.

[LBBH98] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[LCWJ15] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael I. Jordan. Learning transferable features with deep adaptation networks. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 97–105, 2015.

[LKC15] William Lotter, Gabriel Kreiman, and David Cox. Unsupervised learning of visual structure using predictive generative networks. *arXiv preprint arXiv:1511.06380*, 2015.

[LMB+14a] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*, pages 740–755. Springer, 2014.

[LMB+14b] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.

[Low04] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[LSD15] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[LT16] Ming-Yu Liu and Oncel Tuzel. Coupled generative adversarial networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 469–477. 2016.

[LW15] Mingsheng Long and Jianmin Wang. Learning transferable features with deep adaptation networks. *arXiv preprint arXiv:1502.02791*, 2015.

[LWJ16] Mingsheng Long, Jianmin Wang, and Michael I. Jordan. Unsupervised domain adaptation with residual transfer networks. *CoRR*, abs/1602.04433, 2016.

[LYBP16] Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. Hierarchical question-image co-attention for visual question answering. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, 2016.

[LZLY14] L. Liu, L. Zhang, H. Liu, and S. Yan. Toward large-population face identification in unconstrained videos. *IEEE Transactions on Circuits and Systems for Video Technology*, 24(11):1874–1884, Nov 2014.

[Mar82] David Marr. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. Henry Holt and Co., Inc., New York, NY, USA, 1982.

[MCL+14a] Roozbeh Mottaghi, Xianjie Chen, Xiaobai Liu, Nam-Gyu Cho, Seong-Whan Lee, Sanja Fidler, Raquel Urtasun, and Alan Yuille. The role of context for object detection and semantic segmentation in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 891–898, 2014.

[MCL+14b] Roozbeh Mottaghi, Xianjie Chen, Xiaobai Liu, Nam-Gyu Cho, Seong-Whan Lee, Sanja Fidler, Raquel Urtasun, and Alan Yuille. The role of context for object detection and semantic segmentation in the wild. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[MDFF16] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2574–2582, 2016.

[MDG16] Takeru Miyato, Andrew M Dai, and Ian Goodfellow. Adversarial training methods for semi-supervised text classification. *arXiv preprint arXiv:1605.07725*, 2016.

[MMKI17] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *arXiv preprint arXiv:1704.03976*, 2017.

[mod] Caffe model zoo. `https://github.com/BVLC/caffe/wiki/Model-Zoo`. Accessed: 2010-09-30.

[MTL+16] Iacopo Masi, Anh Tuan Tran, Jatuporn Toy Leksut, Tal Hassner, and Gerard Medioni. Do we really need to collect millions of faces for effective face recognition? *arXiv preprint arXiv:1603.07057*, 2016.

[MV15] Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. In *2015 IEEE conference on computer vision and pattern recognition (CVPR)*, pages 5188–5196. IEEE, 2015.

[NQC13] Jie Ni, Qiang Qiu, and Rama Chellappa. Subspace interpolation via dictionary learning for unsupervised domain adaptation. In *2013 IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, June 23-28, 2013*, pages 692–699, 2013.

[NYC15] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 427–436. IEEE, 2015.

[OWJ16] Charles Otto, Dayong Wang, and Anil K. Jain. Clustering millions of faces by identity. *arXiv preprint arXiv:1604.00989*, 2016.

[PCF06] N. Paragios, Y. Chen, and O. Faugeras, editors. *Handbook of Mathematical Models in Computer Vision*. Springer, 2006.

[PMW+16] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *Security and Privacy (SP), 2016 IEEE Symposium on*, pages 582–597. IEEE, 2016.

[PVZ15] Omkar M Parkhi, Andrea Vedaldi, and Andrew Zisserman. Deep face recognition. *BMVC*, 2015.

[RDS+15] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[Res17] Torch implementation of residual networks. *https://github.com/facebook/fb.resnet.torch*, 2017.

[RMC15] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[RPC16] Rajeev Ranjan, Vishal M. Patel, and Rama Chellappa. Hyperface: A deep multi-task learning framework for face detection, landmark localization, pose estimation, and gender recognition. *arXiv ppreprint arXiv:1603.01249*, 2016.

[RSM+16] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio M Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3234–3243, 2016.

[RVM+11] Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 833–840, 2011.

[RVRK16] Stephan R. Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *European Conference on Computer Vision (ECCV)*, volume 9906 of *LNCS*, pages 102–118. Springer International Publishing, 2016.

[S+94] Jianbo Shi et al. Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*, pages 593–600. IEEE, 1994.

[SAB+16] Dmitriy Serdyuk, Kartik Audhkhasi, Philémon Brakel, Bhuvana Ramabhadran, Samuel Thomas, and Yoshua Bengio. Invariant representations for noisy speech recognition. *arXiv preprint arXiv:1612.01928*, 2016.

[SACC16] Swami Sankaranarayanan, Azadeh Alavi, Carlos D Castillo, and Rama Chellappa. Triplet probabilistic embedding for face verification and clustering. In *Biometrics Theory, Applications and Systems (BTAS), 2016 IEEE 8th International Conference on*, pages 1–8. IEEE, 2016.

[SBJ+18] Swami Sankaranarayanan, Yogesh Balaji, Arpit Jain, Ser Nam Lim, and Rama Chellappa. Learning from synthetic data: Addressing domain shift for semantic segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[SCC+16] Soumyadip Sengupta, Jun-Cheng Chen, Carlos D. Castillo, Vishal M Patel, Rama Chellappa, and David W Jacobs. Frontal to profile face verification in the wild. In *WACV*, 2016.

[Sch15] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015.

[SHK+14] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[SJCL18] Swami Sankaranarayanan, Arpit Jain, Rama Chellappa, and Ser Nam Lim. Regularizing deep networks using efficient layerwise adversarial training, 2018.

[SJNL17] Swami Sankaranarayanan, Arpit Jain, and Ser Nam Lim. Guided perturbations: Self-corrective behavior in convolutional neural networks. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[SKP15a] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, 2015.

[SKP15b] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 815–823, 2015.

[SL10] Ruslan Salakhutdinov and Hugo Larochelle. Efficient learning of deep boltzmann machines. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 693–700, 2010.

[SLD16] E. Shelhamer, J. Long, and T. Darrell. Fully convolutional networks for semantic segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PP(99):1–1, 2016.

[SLJ+15] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

[SMDH13] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th international conference on machine learning (ICML-13)*, pages 1139–1147, 2013.

[SNW12] Suvrit Sra, Sebastian Nowozin, and Stephen J Wright. *Optimization for machine learning*. MIT Press, 2012.

[SPVZ13] K. Simonyan, O. M. Parkhi, A. Vedaldi, and A. Zisserman. Fisher Vector Faces in the Wild. In *BMVC*, 2013.

[SYN15] Uri Shaham, Yutaro Yamada, and Sahand Negahban. Understanding adversarial training: Increasing local stability of neural nets through robust optimization. *arXiv preprint arXiv:1511.05432*, 2015.

[SZ14a] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[SZ14b] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[SZS+13a] 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.

[SZS+13b] 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.

[THSD17] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. *CoRR*, abs/1702.05464, 2017.

[TLB+11] Omer Tamuz, Ce Liu, Serge Belongie, Ohad Shamir, and Adam Tauman Kalai. Adaptively learning the crowd kernel. *arXiv preprint arXiv:1105.1033*, 2011.

[TYRW14] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lars Wolf. Deepface: Closing the gap to human-level performance in face verification. In *CVPR 2014*, pages 1701–1708. IEEE, 2014.

[VdMH08] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *JMLR*, 9(2579-2605):85, 2008.

[VDMW12] Laurens Van Der Maaten and Kilian Weinberger. Stochastic triplet embedding. In *Machine Learning for Signal Processing (MLSP), 2012 IEEE International Workshop on*, pages 1–6. IEEE, 2012.

[VGG17] Vgg torch implementation with batch normalization. *https://github.com/szagoruyko/wide-residual-networks*, 2017.

[VTLC16] Raviteja Vemulapalli, Oncel Tuzel, Ming-Yu Liu, and Rama Chellappa. Gaussian conditional random field network for semantic segmentation. CVPR, 2016.

[WBS05] Kilian Q Weinberger, John Blitzer, and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. In *NIPS*, 2005.

[WGQ16] Beilun Wang, Ji Gao, and Yanjun Qi. A theoretical framework for robustness of (deep) classifiers under adversarial noise. *CoRR*, abs/1612.00334, 2016.

[WHT09] L. Wolf, T. Hassner, and Y. Taigman. The one-shot similarity kernel. In *ICCV*, 2009.

[WOJ15] Dayong Wang, Charles Otto, and Anil K Jain. Face search at scale: 80 million gallery. *arXiv preprint arXiv:1507.07242*, 2015.

[WZZ+13] Li Wan, Matthew Zeiler, Sixin Zhang, Yann L. Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In Sanjoy Dasgupta and David Mcallester, editors, *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, volume 28, pages 1058–1066. JMLR Workshop and Conference Proceedings, May 2013.

[YCBL14] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *NIPS*, 2014.

[YCN+15] Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson. Understanding neural networks through deep visualization. *arXiv preprint arXiv:1506.06579*, 2015.

[YK15] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.

[YK16] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. In *ICLR*, 2016.

[YLLL14] Dong Yi, Zhen Lei, Shengcai Liao, and Stan Z Li. Learning face representation from scratch. *arXiv preprint arXiv:1411.7923*, 2014.

[YRC+16] Jiaolong Yang, Peiran Ren, Dong Chen, Fang Wen, Hongdong Li, and Gang Hua. Neural aggregation network for video face recognition. *arXiv preprint arXiv:1603.05474*, 2016.

[ZDG17] Yang Zhang, Philip David, and Boqing Gong. Curriculum domain adaptation for semantic segmentation of urban scenes. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[ZJRP+15] Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip HS Torr. Conditional random fields as recurrent neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1529–1537, 2015.

[ZK16] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *CoRR*, abs/1605.07146, 2016.

[ZKTF10] Matthew D Zeiler, Dilip Krishnan, Graham W Taylor, and Rob Fergus. Deconvolutional networks. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2528–2535. IEEE, 2010.

[ZPIE17] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[ZSLG16] Stephan Zheng, Yang Song, Thomas Leung, and Ian Goodfellow. Improving the robustness of deep neural networks via stability training. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4480–4488, 2016.