ABSTRACT

Title of Dissertation: IMPACT OF SEMANTICS, PHYSICS, AND ADVERSARIAL MECHANISMS IN DEEP LEARNING

Ilya Kavalerov
Doctor of Philosophy, 2020

Dissertation Directed by: Professor Rama Chellappa
Department of Electrical Engineering

Deep learning has greatly advanced the performance of algorithms on tasks such as image classification, speech enhancement, sound separation, and generative image models. However many current popular systems are driven by empirical rules that do not fully exploit the underlying physics of the data. Many speech and audio systems fix STFT preprocessing before their networks. Hyperspectral Image (HSI) methods often don't deliberately consider the spectral spatial trade off that is not present in normal images. Generative Adversarial Networks (GANs) that learn a generative distribution of images don't prioritize semantic labels of the training data. To meet these opportunities we propose to alter known deep learning methods to be more dependent on the semantic and physical underpinnings of the data to create better performing and more robust algorithms for sound separation and classification, image generation, and HSI segmentation. Our approaches take inspiration from from Harmonic Analysis, SVMs, and classical statistical detection theory, and further the state-of-the art in source separation, defense against audio

adversarial attacks, HSI classification, and GANs.

Recent deep learning approaches have achieved impressive performance on speech enhancement and separation tasks. However, these approaches have not been investigated for separating mixtures of arbitrary sounds of different types, a task we refer to as universal sound separation. To study this question, we develop a dataset of mixtures containing arbitrary sounds, and use it to investigate the space of mask-based separation architectures, varying both the overall network architecture and the framewise analysis-synthesis basis for signal transformations. We compare using a short-time Fourier transform (STFT) with a learnable basis at variable window sizes for the feature extraction stage of our sound separation network. We also compare the robustness to adversarial examples of speech classification networks that similarly hybridize established Time-frequency (TF) methods with learnable filter weights.

We analyze HSI images for material classification. For hyperspectral image cubes TF methods decompose spectra into multi-spectral bands, while Neural Networks (NNs) incorporate spatial information across scales and model multiple levels of dependencies between spectral features. The Fourier scattering transform is an amalgamation of time-frequency representations with neural network architectures. We propose and test a three dimensional Fourier scattering method on hyperspectral datasets, and present results that indicate that the Fourier scattering transform is highly effective at representing spectral data when compared with other state-of-the-art methods. We study the spectral-spatial trade-off that our Scattering approach allows. We also use a similar multi-scale approach to develop a defense against audio

adversarial attacks. We propose a unification of a computational model of speech processing in the brain with commercial wake-word networks to create a cortical network, and show that it can increase resistance to adversarial noise without a degradation in performance.

Generative Adversarial Networks are an attractive approach to constructing generative models that mimic a target distribution, and typically use conditional information (cGANs) such as class labels to guide the training of the discriminator and the generator. We propose a loss that ensures generator updates are always class specific, rather than training a function that measures the information theoretic distance between the generative distribution and one target distribution, we generalize the successful hinge-loss that has become an essential ingredient of many GANs to the multi-class setting and use it to train a single generator classifier pair. While the canonical hinge loss made generator updates according to a class agnostic margin a real/fake discriminator learned, our multi-class hinge-loss GAN updates the generator according to many classification margins. With this modification, we are able to accelerate training and achieve state of the art Inception and FID scores on Imagenet128. We study the trade-off between class fidelity and overall diversity of generated images, and show modifications of our method can prioritize either each during training. We show that there is a limit to how closely classification and discrimination can be combined while maintaining sample diversity with some theoretical results on K+1 GANs.

IMPACT OF SEMANTICS, PHYSICS, AND ADVERSARIAL MECHANISMS IN DEEP LEARNING

by

Ilya Kavalerov

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
2020

Advisory Committee:
Professor Rama Chellappa, Chair/Advisor
Professor Wojciech Czaja, Co-Chair/Co-Advisor
Professor Behtash Babadi
Professor Ramani Duraiswami
Professor David Jacobs

Dedication

To Catherine for her patience and encouragement.

Acknowledgments

First and foremost I'd like to thank my advisor, Professor Rama Chellappa for giving me an invaluable opportunity to work on challenging and extremely interesting projects. He has always made himself available for help and advice and there has never been an occasion when I've knocked on his door and he hasn't given me time. It has been a pleasure to work with and learn from such an extraordinary individual.

I would also like to thank my co-advisor, Professor Wojciech Czaja. Without his extraordinary ideas and expertise, this thesis would have been a distant dream. Thanks are due to Professor Behtash Babadi, Professor Ramani Duraiswami, and Professor David Jacobs for agreeing to serve on my thesis committee and for sparing their invaluable time reviewing the manuscript.

My colleagues at UMIACS and lab-mates in Iribe have enriched my graduate life in many ways and deserve a special mention. Weilin Li for his great questions about Harmonic Analysis and Deep Learning theory. Anirudh Nanduri, Steve Schwarcz, Yogesh Balaji, Josh Gleason, Manasij Venkatesh, Zeyad Emam, Liam Fowl, Frank Zheng, Ryan Synk, Ankan Bansal, Arthita Ghosh for their helpful conversations and collaborations. My colleagues at Google Boston: Scott Wisdom, John Hershey, Kevin Wilson, and Brian Patton for all their enthusiasm and patience. My colleagues at Google Mountain View: Shicheng Xu, Nima Asgharbeygi,

Sudarshan Srinivasan, Vaibhav Aggarwal for re-framing my worldview by showing me just how big machine learning can scale in impact and otherwise.

I would also like to acknowledge faculty with whom I have had the privilege of taking classes with. Professor John Benedetto for his great clarity and balanced proofs, and fun anecdotes of world renown mathematicians. Professor Behtash Babadi for his great detail, connectedness, and care in teaching. Professor Prakash Narayan for his distillation of essential simplicity. Professor Piya Pal for her liveliness and imagination. Professor Armand Makowski for his perspective on symmetry and perfection. And Professor Uzi Vishkin for his creativeness.

I owe my deepest thanks to my family - my mother and father who have always stood by me and encouraged me at all times. Words cannot express the gratitude I owe them.

I would like to acknowledge the support of the MURI from the Army Research Office under the Grant No. W911NF-17-1-0304. This is part of the collaboration between US DOD, UK MOD and UK Engineering and Physical Research Council (EPSRC) under the Multidisciplinary University Research Initiative. I would like to acknowledge the support of the Applied Research Laboratory for Intelligence and Security (ARLIS). I would also like to acknowledge support of Tensorflow Research Cloud (TFRC) for their generosity in making available compute time on TPUs for me, and the Google Cloud Platform (GCP) Research Credits for their grant.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

A1	Auditory Cortex
ACGAN	Auxiliary Classifier Generative Adversarial Network
ADC	Analogue to digital converter
AMGAN	Activation Maximixation Generative Adversarial Network
ART	Adversarial Robustness Toolbox
ASR	Automatic speech recognition
BLSTM	Bidirectional LSTM
BM	Binary Mask
CLDNN	Convolutional-LSTM-dense neural network
CNN	Convolutional Neural Network
CS	Crammer-Singer
CSC	Crammer-Singer Complement
DFFN	Deep Feed Forward Network
DLRGF	Discriminative low-rank Gabor filter
DWT	Discrete Wavelet Transform
EAP	Extended Attribute Profile
EMA	Exponential Moving Average
FA	False Alarm
FB	Filter Bank
FC	Fully Connected
FCN	fully convolutional network
FGSM	Fast Gradient Sign Method
FID	Frechet Inception Distance
FM	Feature Matching
FST	Fourier Scattering Transform
GAN	Generative Adversarial Network
GB	Gigabyte
GPU	Graphics Processing Unit
HSI	Hyperspectral Imagery
IFID	Intra-class Frechet Inception Distance
IP	Indian Pines
IS	Inception Distance
KL	Kullback-Leibler
KNN	K Nearest Neighbors

KSC	Kennedy Space Center
LDA	Linear Discriminant Analysis
LFBE	Log Filter Bank Energy
LL	Log Likelihood
LS	Least Squares
LSTM	Long Short Term Memory
MAP	Maximum a posteriori
MFCC	Mel-frequency cepstral coefficients
MHGAN	Multi-Hinge Generative Adversarial Network
MR	Miss Rate
MRF	Markov Random Field
MTCNN	Multi-task Cascaded Convolutional Network
NN	Neural Network
OA	Overall Accuracy
PCA	Principal Component Analysis
PGD	Projected Gradient Descent
PJ	Projection Discrimination
PMF	Probability Mass Function
PU	Pavia University
RBF	Radial Basis Function
RGB	Red Green Blue
RMS	Root Mean Square
SAGAN	Self Attention Generative Adversarial Network
SDR	Signal to Distortion ratio
SI	Scale Invariant
SNGAN	Spectral Normalization Generative Adversarial Network
SNR	Signal to Noise ratio
SoTA	State of the art
SSL	Semi-supervised Learning
SSS	Strictly Site Specific
STFT	Short Time Fourier Transform
STRF	Spectro-temporal Receptive field
SVM	Support Vector Machine
TDB-HW	time-delayed bottleneck highway network
TDCN	time-dilated convolutional network
TFDS	Tensorflow Datasets
TIMIT	Texas Instruments Massachusets Institute of Technology speech dataset
TPU	Tensor Processing Unit
WGAN	Wasserstein Generative Adversarial Network
WSJ	Wall Street Journal speech dataset

WST Wavelet Scattering Transform
WT Wavelet Transform

Chapter 1: Introduction

The question of how to best represent a signal (such as audio or an image) for a task (such as denoising or classification) is a fundamental problem in auditory signal processing and computer vision. Encoding schemes like Huffman coding that were invented for transmitting generic bits ignore whether data is semantically meaningful when encoding it. For example, when representing an audio recording of speech data for the ultimate goal of transcription, background noises can be considered as nuisances and left out. On the other hand when representing the same audio for the task of geographic localization, these background noises may contain relevant information and should not be discarded. This motivates the idea that optimal representations of data depend on the task of interest, and should extract information that is semantically meaningful for the task and discard information that is irrelevant. It is an exciting challenge to consider this problem across data modalities such as audio, color images, and hyperspectral images, and more. Each of these domains have unique characteristics that should be exploited in the aim of optimally representing data generated in them, and each of them can bring their own challenges.

Over the last decade the rapid advancement of Deep Learning techniques,

which come with the promise of learning hierarchical representations of data with minimal user design required, have been at the forefront of tasks such as classification, denoising, and generative modeling on a variety of modalities. Many researchers have advanced the performance of deep learning techniques by building in semantically or physically meaningful priors to deep learning methods. In this dissertation, we aim to contribute along this direction of improving deep learning methods by modifying them to take more advantage of the physics and semantics behind the data they process. Our focus is in improving representations in challenging scenarios when the data is extremely diverse, or when data is extremely limited. In addition to finding better representations for data for the tasks of denoising and classification, we also study a popular generative method from Deep Learning: Generative Adversarial Networks (GANs). We find an opportunity for exploiting semantic information in a task that is in many ways the opposite of representation learning since GANs transform an (often random) embedding vector to a human friendly data modality like images.

Here is an overview of the contributions of this dissertation:

In Chapters 2 to 4 we study the tradeoffs between learnable and classical bases. In Chapter 2 we address source separation of widely varying sound events in audio recordings. Though much work has been done on speech enhancement, we present work on separating mixtures of arbitrary sounds, a new task we refer to as *universal sound separation*. We study two paradigms of source separation models, class-based masking and multiple-instance segmentation, on large datasets

we created explicitly for exploring the capacity of our models, and find that multiple-instance segmentation is better suited to real world tasks. We investigate the space of mask-based separation architectures, varying both the overall network architecture and the framewise analysis-synthesis basis for signal transformations, and propose novel modifications to improve separation performance. In terms of the framewise analysis-synthesis basis, we find that for STFTs, we find that longer windows (25-50 ms) work best for speech/non-speech separation, while shorter windows (2.5 ms) work best for arbitrary sounds. For learnable bases, shorter windows (2.5 ms) work best on all tasks. Surprisingly, for universal sound separation, STFTs outperform learnable bases.

In Chapter 3 we investigate new techniques that are able to capture the special physical properties of hyperspectral data for hyperspectral image (HSI) classification, the one of the most active tasks in geoscience and remote sensing. We build a bridge in this chapter between time-frequency methods that decompose spectra into multi-spectral bands, with hierarchical neural networks that incorporate spatial information across scales and model multiple levels of dependencies between spectral features. To accommodate for the low training sample scenario we investigate the use of the Fourier scattering transform, which is an amalgamation of time-frequency representations with neural network architectures, and advance the state of the art in spectral-spatial classification. We test the proposed three dimensional Fourier scattering method on standard hyperspectral datasets, and show that the Fourier scattering transform is highly effective at representing spectral data when compared with other state-of-the-art spectral-spatial classification methods.

In Chapter 4 we examine similar enhancements for neural networks for robust speech classification. We integrate knowledge from Neuroscience of how the brain represents speech specifically to develop a defense against adversarial attacks on audio. We enhance neural networks used for wake-word detection by integrating a computational approximation to biological sound processing known as the cortical representation. We optimize this architecture to perform at the same level as commercial voice assistant while being several times more resistant to adversarial noise.

Having addressed the discriminatory tasks of source separation and classification in the challenging scenarios of extremely diverse data and adversarial data in audio and scarce data in hyperspectral imagery, we also examine how semantic information can be further exploited in generative tasks in Chapters 5 and 6.

In Chapter 5 we propose a new algorithm to incorporate class conditional information into the discriminator of GANs via a multi-class generalization of the commonly used Hinge loss. Our innovation takes inspiration from SVMs and controls the margins that a spectrally constrained Wasserstein GAN learns. Our approach is in contrast to most GAN frameworks in that we train a classifier and conditional discriminator with the same hinge loss style loss function, instead of an auxiliary cross entropy classifier. We discover a way to connect the tasks of discrimination and classification even more closely by adjusting the embeddings learned by the projection discrimination layer, resulting in a controllable trade-off between class-fidelity (and therefore quality) and diversity. With our multi-hinge loss modification we were able to improve the state of the art Inception Scores and Frechet Inception

Distances on the popular vision Imagenet-128 dataset.

In Chapter 6 we study the interplay between discrimination and classification tasks in GANs more closely in K+1 GANs, which unify these two tasks further. We provide theoretical insight into why K+1 GANs have not developed the same level of performance as conditional Wasserstein GANs. We further confirm the principle learned in Chapter 5 that discrimination and classification cannot be combined too closely, as it results in trading diversity for quality for GANs.

Finally in Chapter 7 we conclude and discuss possible directions for future work to further the understanding of how deep learning methods can exploit the semantic content of data.

Chapter 2: Universal Sound Separation

A fundamental challenge in machine hearing is that of selectively listening to different sounds in an acoustic mixture. Extracting estimates of each source is especially difficult in monaural recordings where there are no directional cues. Recent advances have been made in solving monaural speech enhancement and speech separation problems in increasingly difficult scenarios, thanks to deep learning methods [11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21]. However, separation of arbitrary sounds from each other may still be considered a "holy grail" of the field.

In particular, it is an open question whether current methods are best suited to learning the specifics of a single class of sounds, such as speech, or can learn more general cues for separation that can apply to mixtures of arbitrary sounds. Previous experiments have focused mainly on scenarios where at least one of the target signals to be separated is speech. In speech enhancement, the task is to separate the relatively structured sound of a single speaker from a much less constrained set of non-speech sounds. For separation of multiple speakers, the state of the art has progressed from speaker-dependent separation [22], where models are trained on individual speakers or speaker combinations, to speaker-independent speech separation [16, 17, 18], where the system has to be flexible enough to separate

unknown speakers. In particular, ConvTasNet is a recently proposed model [2] that uses a combination of learned time-domain analysis and synthesis transforms with a time-dilated convolutional network (TDCN), showing significant improvements on the task of speech separation relative to previously state-of-the-art models based on short-time Fourier transform (STFT) analysis/synthesis transforms and long short-term memory (LSTM) recurrent networks. Despite this progress, it is still unknown how current methods perform on separation of arbitrary types of sounds. The fact that human hearing is so adept at selective listening suggests that more general principles of separation exist and can be learned from large databases of arbitrary sounds.

Separation scenarios broadly fall into two categories: *class-based* separation and *multiple-instance* separation [16]. In class-based separation, the task is to separate whole classes of signals from one another, such as speech versus non-speech. They can be viewed as classifying each time-frequency (T-F) bin in an input mixture signal. A caveat for class-based approaches is scaling them to finer grained separation problems with a multitude of sound event classes, where the model complexity may grow prohibitively with the number of classes present in the dataset. Such models cannot be applied to test data that contains sound classes not trained upon. Moreover, given that training datasets require explicit class labels, it may be challenging to prepare enough data belonging to distinct and non-overlapping classes. Finally, the class-based separation side-steps the commonplace real-world condition where there is more than one source of the same type, such as in a conversation.

In multiple-instance separation, multiple sounds of the same type may be

present, as in the case of speaker-independent speech separation. Although class information may be useful, each occurrence of a type of sound may have its own distinctive traits. These traits are cues that enable multiple sounds to be distinguished from each other using a *partition-based* segmentation approach. Partition-based approaches do not depend on the number of classes in the data, so they may flexibly adapt to richer datasets. Since they do not require class labels, they may be applied with more abundant and larger training datasets. As classes may contain significant within-class variation, and different classes may overlap with each other, the class-based approach may fail to learn these distinguishing traits, whereas a system trained only to separate each sound from one another, regardless of its class, may have the advantage of training on more varied cues.

In the first part of this chapter, we investigate the source separation problem for non-speech audio and evaluate our models' ability to segment diverse non-speech sounds. We compare a straightforward class-based segmentation approach with a more scalable partition-based segmentation approach [16] on two large and diverse datasets of real world non-speech sounds, and show that these models can even separate some sounds that humans struggle to consistently segment. Our focus is on the single channel setting, but our models can easily be extended to multiple channels as in [23].

In the second part of this chapter, we improve our partition based approach. We evaluate ConvTasNet on both speech/non-speech separation and universal sound separation tasks for the first time. We provide a systematic comparison of different combinations of masking network architectures and analysis-synthesis transforms,

optimizing each over the effect of window size. We also propose novel variations in architecture, including alternative feature normalization, improved initialization, longer range skip-residual connections, and iterative processing that further improve separation performance on all tasks.

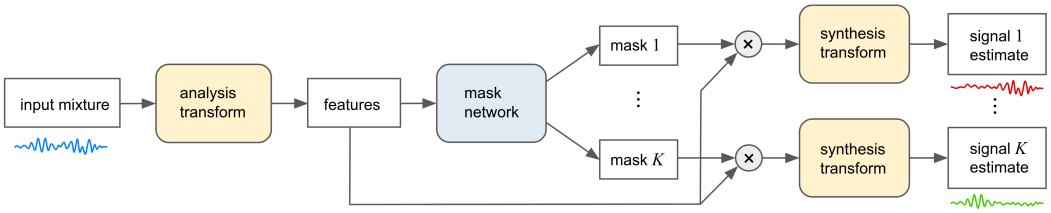


Figure 2.1: Architecture for mask-based separation experiments. We vary the mask network and analysis/synthesis transforms.

2.1 Background

A variety of networks have been successfully applied to two-source separation problems, including LSTMs and bidirectional LSTMs (BLSTMs) [12, 14], U-Nets [24], Wasserstein GANs [25], and fully convolutional network (FCN) encoder-decoders followed by a BLSTM [26]. For multi-source separation, a variety of architectures have been used that directly generate a mask for each source, including BLSTMs [16, 19], CNNs [27], DenseNets followed by an LSTM [28], separate encoder-decoder networks for each source [29], joint one-to-many encoder-decoder networks with o decoder per source [30], and TDCNs with learnable analysis-synthesis basis [2]. Our models are most similar to [19] and [2].

Networks that perform source separation in an embedding space rather than directly in the T-F space, such as deep clustering, have been effective. Deep clustering learns a high dimensional embedding for each T-F unit that can be optimized with

objectives like deep LDA or whitened k-means [16, 21]. Deep attractor networks learn both the embedding function and embedding space class centroids that are used to segment the embedding space in a nearest neighbor fashion [31]. These networks are convenient in that they solve the more general problem of partition-based rather than class-based segmentation [16]. Direct masking and embedding-based models are also complementary in that they can be combined in a chimera network, with one head performing mask inference (MI), and the other acting as a regularizer by minimizing a loss on the learned embeddings. Such models have achieved state-of-the-art performance on music separation [32] and two-speaker source separation [21]. In this chapter, we focus on direct masking approaches with permutation-invariant losses, and leave the exploration of deep clustering related approaches for multisource non-speech source separation to future work.

Many datasets for source separation have focused on mixing speech with non-speech or other speech, with the speech sourced from datasets such as TIMIT [25] and WSJ0 [16, 31, 33]. Small datasets used for the non-speech multi-source separation setting have included distressing sounds from DCASE 2017 [27], music instrument tracks in Demixing Secrets Dataset 100 [32], and speech and music in SiSEC-2015 [26, 29]. A larger music dataset for the separation of vocals from all accompaniments in commercial music has been investigated by Spotify [24]. The datasets we introduce are the first of their kind investigated in multisource separation in that up to 5 overlapping real-world sounds are mixed together at once, from a selection of up to 77 distinct sound classes.

2.2 Class based vs. Partition based separation

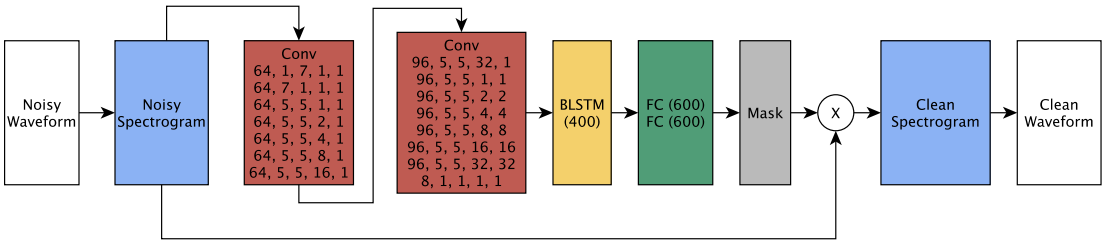


Figure 2.2: Structure of the CLDNN mask network. In the convolutional layers, the 5 integers per layer correspond to the number of filters, the kernel size in frames, the kernel size in frequency, the dilation in frames, the dilation in frequency.

We use a mask-based enhancement system driven by a deep neural network. The masking network we use consists of 14 dilated 2D convolutional layers, a bidirectional LSTM, and two dense layers, which we will refer to as a convolutional-LSTM-dense neural network (CLDNN). The CLDNN is based on a network which achieves state-of-the-art performance on CHiME2 WSJ0 speech enhancement [3] and strong performance on a large internal dataset at Google [4]. The architecture of CLDNN is shown in Fig. 2.2.

We consider two different settings, a class-based separation network, which estimates a mask for each class of sound in the dataset, and a partition-based network, which estimates a fixed number of masks corresponding to the maximum number of expected sources in the mixture, regardless of class information.

In the speech enhancement setting, this network, shown in Fig. 2.1, produces a speech mask and a noise mask, performing two-class separation. In the case of C different classes present in the dataset, we alter the final layers of the network to always output C masks, independent of the number of classes present per mixture.

The loss function used to train this network is:

$$\mathcal{L}_{\text{Class}} = \sum_{c=1}^{C} \|R_c^\alpha - |M_c \odot X|^\alpha\|_2^2 \tag{2.1}$$

where R_c denotes the short-time Fourier transform (STFT) magnitude for the clean source of the c-th class, α a power-law compression factor (here set to $\alpha = 0.3$), M_c the mask predicted for the c-th source, X the complex STFT of the mixture, and $\odot$ the element-wise multiplication. When class c is not present in the mixture X, $R_c = 0$. Thus, when the number of classes in a mixture is less than the maximum number of classes C in the dataset, the network is implicitly performing classification of the sounds present in the mixture by attributing energy to the masks corresponding to these sources.

To generalize to real-world datasets which have an unknown and large number C of possible sound classes that could be in a mixture, we consider a different system in which we alter the final layers of the network to produce $K \leq C$ masks instead, where K is the maximum number of clean audio signals we want to extract. In this case, the order of the masks generated by the network is no longer tied to the type of sound, and there is thus no longer a direct matching with the references at training time, so we need to use a permutation free (PF) loss:

$$\mathcal{L}_{\text{PF}} = \min_{\sigma \in S_K} \sum_{k=1}^{K} \left\| R_k^\alpha - |M_{\sigma(k)} \odot X|^\alpha \right\|_2^2 \tag{2.2}$$

where S_K denotes the symmetric group of degree K (the set of permutations on

$\{1, \ldots, K\}$) [16, 17, 18]. Only the error for the best permutation is used in training. Although the cardinality of S_K is $K!$, in our experiments $K \leq 5$ and this minimization did not lengthen training time significantly. Even for larger K, the time-consuming loss function computation can be first done in parallel for all pairs $(i, j), 1 \leq i, j \leq K$, and the exhaustive search over permutations for the best combination is performed on the scores, which is relatively fast to compute. At very large scales ($K \gg 5$), a more efficient algorithm could be used to speed up training time. However, note that at run time there is no matching problem so the cost of searching permutations goes away.

Both models are trained end-to-end. To retrieve the denoised audio, an inverse STFT is applied to the masked T-F representations $M_c \odot X$. An advantage of the PF model is that the same network can be tested on other datasets potentially containing more classes than the model was trained on.

2.2.1 Dataset construction

To investigate the general source separation problem, we extract loud events from the ProSound database [34] in 3 s clips, and mix them together. For each dataset, 70% of the source material was used for training audio, while 20% and 10% were reserved exclusively for validation and test audio.

Urban street corner dataset. The urban street corner dataset is constructed to simulate the noises heard on a city street corner, an auditory scene prevalent in online multimedia. The five classes of sounds chosen are a car accelerating (revving,

wheels squealing), the fricative sound of a car passing, an engine (of various volumes) idling, a horn honking, and footsteps over various hard and soft surfaces.

Despite the source recordings of these sound classes in the ProSound database being minutes-long, they often contained only a few salient sound events just a few seconds long, separated by quiet or ambience. We automatically identified the start of these sound events in the recordings by detecting when the local (15-60ms) root-mean-squared (RMS) power changed from below the average to above average. This condition is triggered several times per recording for footsteps and honking, but often only once for cars passing. When sources are mixed, we select a 3-second segment centered on the detected event time with a random offset of up to half a second before or after.

To generate each mixture clip, K sound classes are chosen randomly, for each of which one sound event is chosen at random, a 3 s segment is extracted with jitter, and the segments for the K classes are added together. There is no adaptive volume normalization, and to avoid clipping the intensity of each premixed segment is halved.

We report results for $K \in \{2, 3, 5\}$. The sounds often significantly overlapped in the STFT domain, and were sometimes hard for human listeners to distinguish. The mixed dataset comprises 10 hours of audio and is constructed using a few hours of unmixed source audio.

Diverse dataset. The diverse dataset is created to have tremendous variability and to leverage a larger training set size by removing any class restrictions on the source audio. Sounds used included crawling insects, animal calls, creaking doors,

construction noises, musical instruments, speech, composed music, and artificial sounds (e.g., science fiction sounds). Ambience/environment sounds are excluded.

Because the source material is so unconstrained, the 20 hours of mixtures we created contain many more unique types of sources compared to the urban street corner dataset. However, the large number of classes in the dataset (77) makes it challenging to consider a class-specific approach for separating the mixtures. In this larger set of classes, there are many sound files that are shorter than 3 s; in this case, the short clips are looped with a random delay up to a second to create a single source segment. Overall, 11,797 audio files are used as source material for the training set, 3,370 for the validation set, and 1,686 for the test set. The same class distribution was maintained in these three data partitions.

2.2.2 Training and evaluation setup

All experiments are performed using TensorFlow [35], trained with the Adam [36] optimizer with batch size 2 on a single NVIDIA Tesla V100 GPU. Separation performance is measured using scale-invariant signal-to-distortion ratio improvement (SI-SDRi) [17, 37], which evaluates the fidelity of a signal estimate $\hat{s}$, represented as a vector, relative to the ground truth signal s while accommodating a possible scale mismatch. SI-SDR is computed as

$$\text{SI-SDR}(s, \hat{s}) \stackrel{\text{def}}{=} 10 \log_{10} \frac{\|\alpha s\|^2}{\|\alpha s - \hat{s}\|^2}, \tag{2.3}$$

where $\alpha = \text{argmin}_a \|as - \hat{s}\|^2 = \langle s, \hat{s} \rangle / \|s\|^2$, and $\langle a, b \rangle$ denotes the inner product. SI-SDRi is the difference between the SI-SDR of the estimated signal and that of the input mixture signal. The sample rate for the mixtures was 16 kHz, and all STFTs use a square-root Hann window, where windowed frames are zero-padded to the next power of 2 above the window size.

We use a permutation-invariant loss to align network outputs with the reference sources during training, where the loss used for a gradient step on a batch is the minimum error across the set S_K of all permutations of the K estimated sources, compared to the fixed K reference sources [16, 17, 18]. Although the cardinality of S_K is $K!$, in our experiments $K \leq 3$ and this minimization did not lengthen training time significantly. Even for larger K, the time-consuming loss function computation can be first done in parallel for all pairs $(i, j), 1 \leq i, j \leq K$, and the exhaustive search over permutations for the best combination is performed on the scores.

All networks use negative signal-to-noise ratio (SNR) as their training loss f between time-domain reference source y and separated source $\hat{y}$, defined as

$$f(y, \hat{y}) = -10 \log_{10} \left(\frac{\sum_t y_t^2}{\sum_t (y_t - \hat{y}_t)^2} \right). \tag{2.4}$$

Compared to negative SI-SDR used to train ConvTasNet [2], this negative SNR objective has the advantage that the scale of separated sources is preserved and consistent with the mixture, which is further enforced by our use of mixture consistency layers [4]. Since we measure loss in the time domain, gradients are backpropagated through the synthesis transform and its overlap-add layer, so STFT consistency

[4, 38] is implicitly enforced when using the STFT.

2.2.3 Experiments

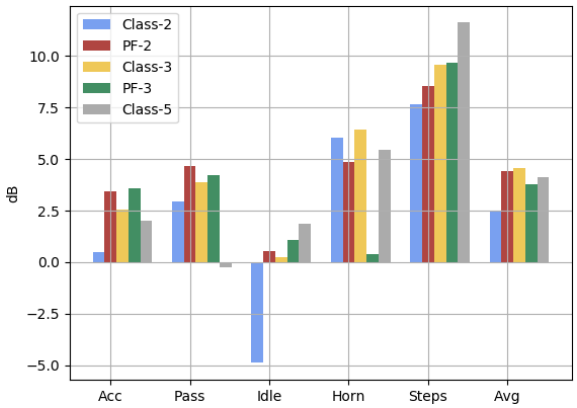


Figure 2.3: Median improvement in SI-SDR in dB on the urban street corner dataset, for each class and for the average.

For the urban street corner dataset, Fig. 2.3 compares the scale-invariant SDR improvements obtained by all models with $K \in \{2, 3, 5\}$, for the C denoised signals reported separately for each class of signal, using the mixed noisy signal as baseline. Class-K and PF-K denote class-specific and permutation-free models trained and tested with K sources in each mixed noisy clip. The PF model outperforms the corresponding class-specific model for every sound class other than horn, the most harmonic sound class. These results also indicate, not surprisingly, that some types of sounds may be much easier to segment than others: transient events such as footsteps and horn are relatively better separated than smoother sound textures such as engine idling. Overall the best scores on the urban street corner data were 4.4 dB, 4.6 dB, and 4.1 dB SI-SDR improvements on mixtures of 2, 3, and 5 sounds, respectively. Some examples of learned segmentations are shown in Figs. 2.4 and

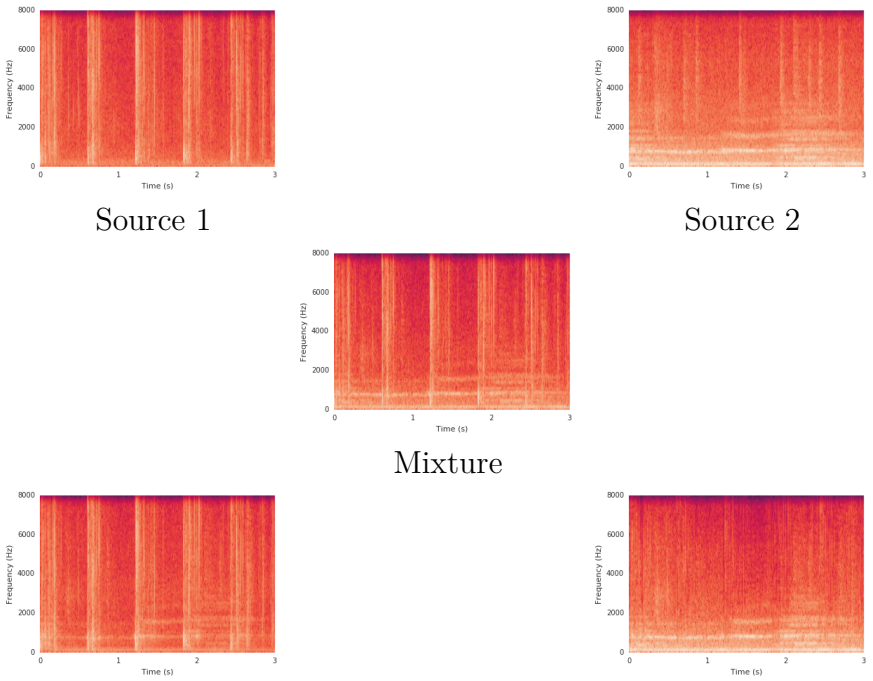


Figure 2.4: Footsteps mixed with car accelerating.

Table 2.1: Median scale invariant SDR (dB) for 2-source separation for the diverse dataset.

	Source 1	Source 2
Noisy	0.5	-0.6
PF	7.5	7.8

2.5.

The results on the diverse data are shown in Tables 2.1 and 2.2. The network seems to do surprisingly well, averaging around 7.7 dB of improvement in SI-SDR in the two-source case, and 6 dB improvement in the three-source case.

2.2.4 Discussion

The results on the urban street corner data show that permutation-free loss seems to work better than class-based separation in most cases. This is a surprise,

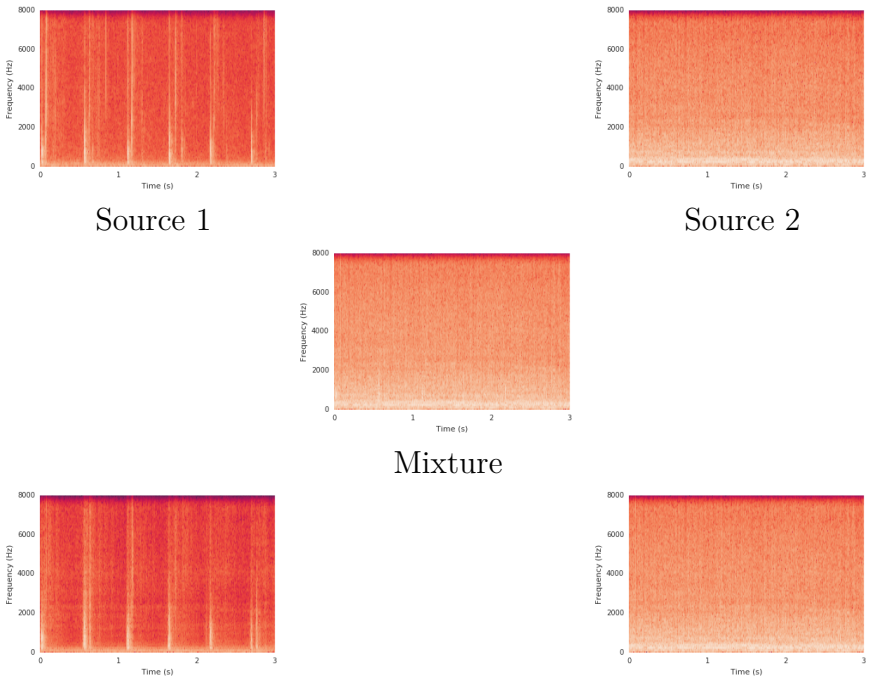


Figure 2.5: Footsteps mixed with engine idling

Table 2.2: Median scale invariant SDR (dB) for 3-source separation for the diverse dataset.

	Source 1	Source 2	Source 3
Noisy	-5.2	-4.9	-6.0
PF	1.1	0.8	0.2

since the class-based loss provides more information than the permutation-free loss. One hypothesis might be that the class-based scenario is a harder problem: in addition to separating the signals, the network has to identify which signal belongs to which class. It is also interesting to note that we observed no obvious correspondence between the classes and the order of the PF network outputs, so perhaps the boundaries between different types of sounds follow some pattern that made them easier to learn than the class boundaries.

In this section, two models have been presented for separating arbitrary audio,

a class-specific and a non-class-specific permutation free (PF) model. Surprisingly, using the class information in the outdoor scene dataset does not always improve the SI-SDR of the separated audio. This points to the strength of the PF model which discards the class information in this setting. This model also proves to generalize to the non-class-constrained setting with the diverse dataset. The SI-SDR performance on separating the diverse sounds was in fact competitive with separating the constrained sounds, speaking to the adaptability of the network used when trained with the permutation free loss. In the next section we follow up on these promising results and train even higher performing sound separation networks.

2.3 Masking Networks with Learnable analysis-synthesis bases

In this section we experiment with combinations of different network architectures and different analysis-synthesis bases for neural-network mask-based separation systems. All masking networks use a sigmoid activation to predict a real number in $[0, 1]$ to modulate each basis coefficient.

2.3.1 Masking network architectures

The first masking network we use is the CLDNN from the previous section. Our second masking network is a TDCN inspired by ConvTasNet [2]. We employ the same parameters as the best noncausal model reported by [2]. We also consider an improved version of ConvTasNet's TDCN masking network, which we refer to as "improved TDCN" (TDCN++). This new architecture includes three improvements

to the original ConvTasNet network. First, global layer normalization within the TDCN, which normalizes over all features and frames, is replaced with a feature-wise layer normalization over frames. This is inspired by cepstral mean and variance normalization used in automatic speech recognition systems. Second, we add longer-range skip-residual connections from earlier repeat inputs to later repeat inputs after passing them through dense layers. This presumably helps with gradient flow from layer to layer during training. Third, we add a learnable scaling parameter after each dense layer. The scaling parameter for the second dense layer in each convolutional block – which is applied right before the residual connection – is initialized to an exponentially decaying scalar equal to 0.9^L, where L is the layer or block index. This initial scaling contributes to better training convergence by first learning the contributions of the bottom layers, similar to layer-wise training, and then easily adjusting the scale of each block's contribution through the learnable scaling parameter. This initialization is partly inspired by "Fixup" initialization in residual networks [39].

A third network variant we consider is an iterative improved TDCN network (iTDCN++), in which the signal estimates from an initial mask-based separation network serve as input, along with the original mixture, to a second separation network. This architecture is inspired by [17], in which a similar iterative scheme with LSTM-based networks led to significant performance improvements. In our version, both the first and second stage networks are identical copies of the TDCN++ network architecture, except for the inputs and parameters. In the second stage, the noisy mixture and initial signal estimates are transformed by the same basis (STFT

or learned) prior to concatenation of their coefficients. Because, with two iterations, the network is twice as deep as a single-stage TDCN++, we also include a twice deeper TDCN++ model (2xTDCN++) for comparison.

2.3.2 Analysis-synthesis bases

Whereas earlier mask-based separation work had used STFTs as the analysis-synthesis basis due to the sparsity of many signals in this domain, ConvTasNet [2] uses a learnable analysis-synthesis basis. The analysis transform is a framewise basis analogous to the STFT, and can also be described as a 1D convolution layer where the kernel size is the window size, the stride is the hop length, and the number of filters is the number of basis vectors. A ReLU activation is applied to the analysis coefficients before processing by the mask network. The learnable synthesis transform can be expressed as a transposed 1D convolution and operates similarly to an inverse STFT, where a linear synthesis basis operates on coefficients to produce frames which are overlap-added to form a time-domain signal. Unlike an STFT, this learnable basis and its resulting coefficients are real-valued.

The original work [2] found that ConvTasNet performed best with very short (2.5 ms) learnable basis functions. However, this window size is an important parameter that needs to be optimized for each architecture, input transform, and data type. We therefore compare a learnable basis with STFT as a function of window size, in combination with CLDNN and TDCN masking networks. All models apply mixture consistency projections to their outputs [4], which ensure the estimated sources add up to the input mixture. Note that the TDCN with STFT basis is a

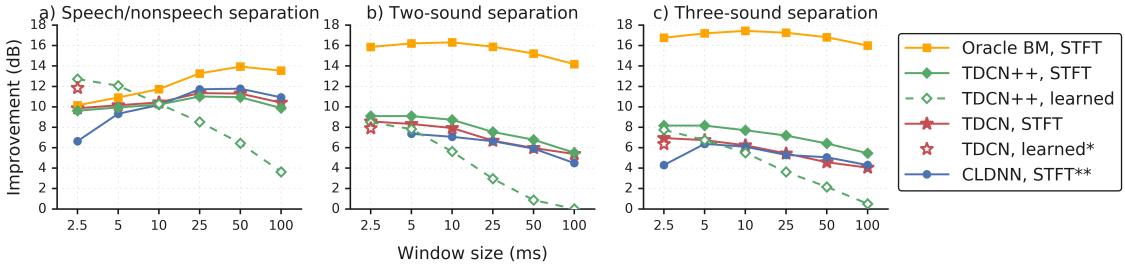


Figure 2.6: Mean SI-SDR improvement in dB on the test set as a function of basis window size in ms, using different combinations of network architectures and bases, on a) speech/non-speech separation, b) two-sound universal separation, and c) three-sound universal separation. Systems * and ** come from [2] and [3, 4], respectively. "Oracle BM" corresponds to an oracle binary STFT mask, a theoretical upper bound on our systems' performance. Note the CLDNN STFT failed to converge for 2.5 ms windows on two-sound separation and is omitted.

Table 2.3: Mean scale-invariant SDR improvement (dB) for speech/non-speech separation and two-source or three-source sound separation. Note that the bottom four TDCN networks below the thick line are twice as deep as the top four TDCN networks above the thick line.

	Speech/non-speech separation			Two-source separation			Three-source separation		
Masking network, basis	**Best win. size**	**Val. SI-SDRi**	**Test SI-SDRi**	**Best win. size**	**Val. SI-SDRi**	**Test SI-SDRi**	**Best win. size**	**Val. SI-SDRi**	**Test SI-SDRi**
CLDNN, STFT [3, 4]	50 ms	11.9	11.8	5.0 ms	7.8	7.4	5.0 ms	6.7	6.4
TDCN, learned [2]	2.5 ms	12.6	12.5	2.5 ms	8.5	7.9	2.5 ms	6.8	6.4
TDCN, STFT	25 ms	11.5	11.3	2.5 ms	9.4	8.6	2.5 ms	7.6	7.0
TDCN++, learned	2.5 ms	**12.7**	**12.7**	2.5 ms	9.1	8.5	2.5 ms	8.4	7.7
TDCN++, STFT	25 ms	11.1	11.0	2.5 ms	**9.9**	**9.1**	5.0 ms	**8.8**	**8.2**
2xTDCN++, learned	2.5 ms	13.3	13.2	2.5 ms	8.1	7.6	2.5 ms	8.0	7.3
2xTDCN++, STFT	25 ms	11.2	11.1	5.0 ms	9.3	8.3	5.0 ms	9.0	8.0
iTDCN++, learned	2.5 ms	**13.5**	**13.4**	2.5 ms	9.3	8.7	2.5 ms	8.1	7.4
iTDCN++, STFT	25 ms	11.6	11.5	2.5 ms	**10.6**	**9.8**	2.5 ms	**9.6**	**8.7**

novel combination that, as we show below, performs best on the universal separation task.

2.3.3 Experiments

Results on the universal data are shown in Figure 2.6 and Table 2.3, and audio demos may be found online: `https://universal-sound-separation.github.io`. Figure 2.6 shows results for different window sizes, where for each size, the hop is half the window size. For comparison, speech/non-speech separation performance

on data described in [4] is shown[1] alongside results for two-source and three-source universal sound separation. We also tried training CLDNN networks with learned bases, but these networks failed to converge and are not shown. For all tasks, we show the performance of an oracle binary mask using an STFT for varying window sizes. These oracle scores provide a theoretical upper bound on the possible performance of our methods.

The differences between tasks in terms of basis type are striking. Notice that for speech/non-speech separation, longer STFT windows are preferred for all masking networks, while shorter windows are best when using a learnable basis. For universal sound separation, the optimal window sizes are shorter in general compared to speech/non-speech separation, regardless of the basis.

Window size is an important variable since it controls the frame rate and temporal resolution of the network, as well as the basis size in the case of STFT analysis and synthesis transforms. The frame rate also determines the temporal context seen by the network. On the speech/non-speech separation task, for all masking networks, 25-50 ms is the best window size. Speech may work better with such relatively long windows for a variety of reasons: speech is largely voiced and has sustained harmonic tones, with both the pitch and vocal tract parameters varying relatively slowly. Thus, speech is well described by sparse patterns in an STFT with longer windows as preferred by the models, and may thus be easier to separate in this domain. Speech is also highly structured and may carry more predictable

[1]Note that we consider here the more general "speech/non-speech separation" task, in contrast to the "speech enhancement" task, which typically refers to separating only the speech signal.

longer-term contextual information than arbitrary sounds; with longer windows, the LSTM in a CLDNN has to remember information across fewer frames for a given temporal context.

For universal sound separation, the TDCNs prefer short (2.5 ms or 5 ms) frames, and the optimal window size for the CLDNN is 5 ms or less, which in both cases is much shorter than the optimal window size for speech/non-speech separation. This holds both with learned bases and with the STFT basis. Surprisingly the STFT outperforms learned bases for sound separation overall, whereas the opposite is true for speech/non-speech separation. In contrast to speech/non-speech separation, where a learned basis can exploit the structure of speech signals, it is perhaps more difficult to learn general-purpose basis functions for the wide variety of acoustic patterns present in arbitrary sounds. In contrast to speech, arbitrary sounds may contain more percussive components, and hence be better represented using an STFT with finer time resolution. To fairly compare different models, we report results using the optimal window size for each architecture, determined via cross-validation.

Table 2.3 shows summary comparisons using the best window size for each masking network and basis. The optimal performance for speech/non-speech separation is achieved by models using learnable bases, while for universal sound separation, STFTs provide a better representation. For both two-source and three-source separation, the iTDCN++ with 2.5 ms STFT basis provides the best average SI-SDR improvement of 9.8 dB and 8.7 dB, respectively, on the test set, whereas the 2xTDCNN++, is not competitive on the universal separation task. For speech/non-

speech separation, the iTDCN++ with a 2.5 ms learnable basis achieves the best performance of 13.4 dB SI-SDRi. The iterative networks were trained with the loss function applied to the output of each iteration. These results point to iterative separation as a promising direction for future exploration.

Figure 2.7 shows scatter plots of input SI-SDR versus improvement in SI-SDR for each example in the test set. Panel a) displays results for the best model from Table 2.3, and panel b) displays results for oracle binary masking computed using an STFT with 10 ms windows and 5 ms hop. Oracle binary masking achieves 16.3 dB mean SI-SDRi, and indicates the potential separation that can be achieved on this dataset.

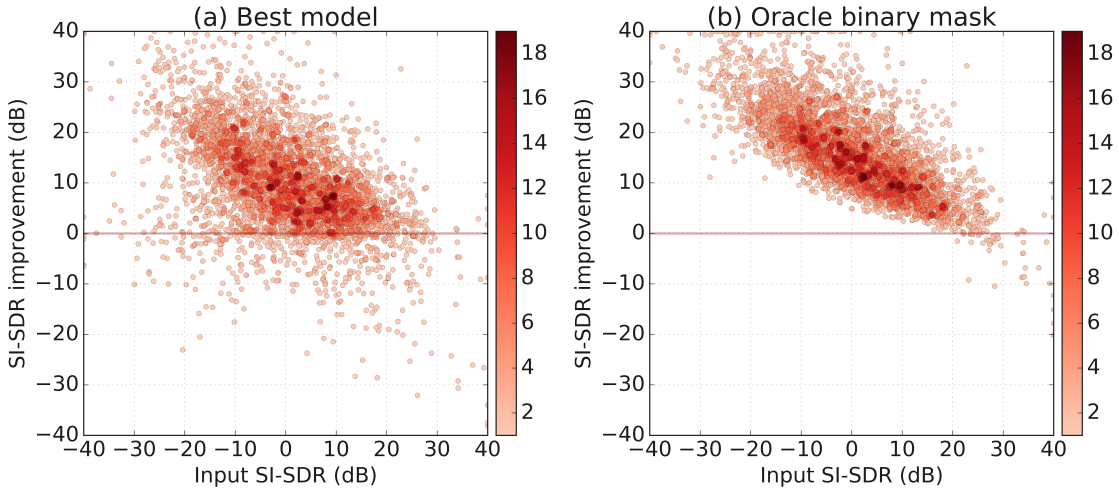


Figure 2.7: Scatter plots of input SI-SDR versus SI-SDR improvement on two-source universal sound mixture test set for a) our best model (iTDCN++, STFT) and b) oracle binary masking using an STFT with 10 ms window and 5 ms hop. The darkness of points is proportional to the number of overlapping points.

2.3.4 Conclusion

We compared different combinations of network architectures and analysis-synthesis transforms, optimizing each over the effect of window size. We also pro-

posed novel variations in architecture, including longer-range skip-residual connections and iterative processing, that improve separation performance on all tasks. Interestingly, the optimal basis and window size are different when separating speech versus separating arbitrary sounds, with learned bases working better for speech/non-speech separation, and STFTs working better for sound separation. The best models, using iterative TDCN++, produce an average SI-SDR improvement of almost 10 dB on sound separation, and over 13 dB on speech/non-speech separation. Overall, these are extremely promising results which show that perhaps the holy grail of universal sound separation may soon be within reach.

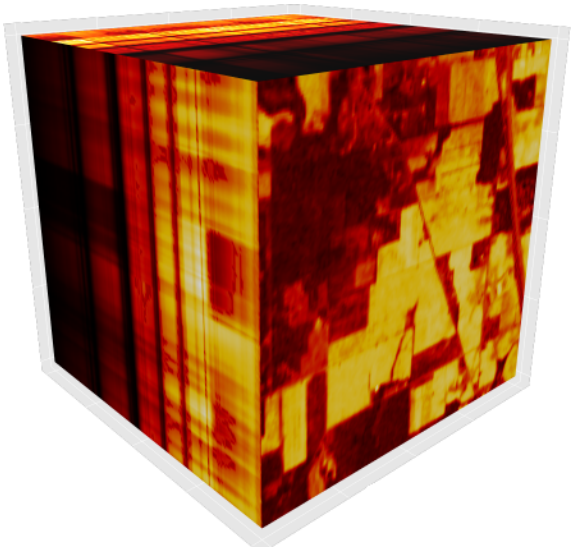


Figure 3.1: Hyperspectral data cube

Chapter 3: Three-Dimensional Fourier Scattering Transform and Classification of Hyperspectral Images

Hyperspectral image sensors routinely collect hundreds of bands of different wavelength channels of the surface of the Earth [40]. Due to the rapidly growing amount of available hyperspectral imagery (HSI) [41], there is much interest in the development of algorithms that can take advantage of these resources for a wide range of applications, from anomaly detection to automatic classification. However, several characteristics of HSI data make these tasks challenging: the high dimensionality of the data, the low spatial resolution, the resulting unfavorable signal-to-noise

ratio, and the fact that labeled data is scarce and typically not transferable across different domains. This emphasizes the continued need for development of more efficient processing algorithms.

One potential source of needed advancements is the field of machine learning. As recently noted in the review by He *et al.* [42], neural networks (NNs) and deep learning, having already achieved breakthroughs in the traditional image classification or segmentation tasks, are now gaining popularity in HSI applications [43, 44, 45, 46, 47]. These hierarchical networks built for feature extraction at multiple levels have a potential to produce highly informative data features, which cannot be achieved by manually-designed feature extractors. However, the cost of these improvements is the increase in computational complexity of learning algorithms, which stems from the fact that there is now a great number of parameters to train. This predicament is often resolved by combining the learning scheme with an appropriately chosen representation transform which maximizes the information content in the first layer, consecutively leading to a reduction of the time needed for training the algorithm. It is thus in this context that we note that time-frequency representations for HSI have recently proven to provide both meaningful and high quality results [42, 48, 49], especially when the filters are specifically designed for the HSI data [50]. At the same time, these time-frequency representations are an underutilized tool when compared with the more popular wavelet based methods [51, 52, 53, 54, 55, 56].

The *Fourier scattering transformation* (FST), introduced in [57, 58], can be viewed as a modern machine learning-inspired approach to time-frequency analysis.

It unifies deep learning architectures with time-frequency generated filters in order to capture higher order correlations between different time-frequency coefficients. One notable difference from deep learning is that the FST uses fixed filters instead of adaptable or learned ones such as those in NNs. However, the advantage is that the FST does not require computationally expensive training and enjoys theoretical guarantees that NNs lack [57]. In particular, the FST is invariant under small diffeomorphic nonlinearities or perturbations.

The *three-dimensional Fourier scattering transform* (3D FST) algorithm, which is introduced in this chapter, is inspired by the mathematical Fourier scattering transformation for square integrable functions on a continuum. The novelty of this algorithm is the deviation from the original continuous time one-dimensional setting designated for traditional function approximation. Instead, here we design and employ three dimensional time-frequency generated filters to deal simultaneously with the discrete spatial and spectral aspects of the data. Furthermore, we implement these filters in a purpose-built architecture which takes advantage of recent developments in convolutional NNs. When used on HSI data, it provides a multi-layer spectral-spatial decomposition. We argue that the spectra generated by standard material classes are more discriminable in the time-frequency domain when compared to other representations. We note that this argument was also made in [50], which showed that decomposing the signal using time-frequency filters provides informative features for HSI data. However, the 3D FST further refines this idea by integrating together the spectral and spatial information in a multi-layer setting, where deeper layers are able to capture more complex features.

We demonstrate that the 3D FST provides state-of-the-art performance on HSI data. We compare to results with neural network and wavelet scattering based methods [56, 59, 60]. One notable method that we compare with is the *three-dimensional wavelet scattering transform* (3D WST). The *wavelet scattering transform* (WST) was originally developed by Mallat [61] and the 3D WST was applied to HSI classification in [56]. Our results show that time-frequency Fourier features are more suitable than time-scale wavelet features for HSI discrimination and classification purposes. Among others, we obtain state of the art results on Indian Pines at 10% and 5% of training data, and on Pavia University at 1% and 0.5%. We also provide an open source implementation of all code used for our algorithms and experiments.[1]

The rest of this chapter is organized as follows. Section 3.1.1 reviews related work on using neural networks, wavelets, and time-frequency bases for feature extraction and the classification of HSI data. Section 3.1.2 provides background information on scattering transforms. Section 3.2 defines the 3D FST and how it provides a joint spectral-spatial representation suited for HSI data. Section 3.3 introduces the datasets that 3D FST is evaluated on, explains the parameter choices in the 3D FST, and discusses the results on these datasets while comparing them to other competing methods.

[1] `https://github.com/ilyakava/pyfst`

3.1 Background

3.1.1 Previous Work

Not surprisingly, many methods available in the literature have concentrated solely on analyzing the spectrum content for the classification of HSI data. More recently, to improve classification performance, spectral-spatial techniques which better exploit the properties of HSI data have become popular. Our review of deep learning, neural network, wavelet, and time-frequency methods can be roughly split into four categories: 1) purely spectral techniques 2) spectral methods that incorporate spatial pre/post processing, 3) purely spatial methods that may include spectral pre/post processing, and 4) those that integrate spectral-spatial information at once.

1. Pixelwise methods that extract wavelet, time-frequency, and neural network features solely in the spectral domain have been developed to address the challenges in HSI classification [62, 63, 64, 65, 66, 67]. The neural network (NN) family of methods iteratively composes layers of matrix multiplications or linear convolutions with a pointwise non-linear function. The weights in these matrices, or convolution filters, are adapted using backpropagation during an initial training phase. NNs consisting of 1D convolutions with spectra [63], and 2D convolutions of reshaped 1D spectral vectors [66], as well as other Deep Belief Networks (DBN) [64] have been evaluated. Another family of methods are the wavelet and time-frequency methods which use a pre-determined basis

to extract edge-like features. 1D Morlet wavelet features with trainable scale and translation parameters have been extracted and input to a 2 layer NN [65, 68].

A compromise between the learned and potentially deep and complex neural networks, and classical wavelet features are scattering transforms, which are particular types of operators introduced by S. Mallat [61]. The coefficients are computed with a hierarchical network structure that captures several types of invariances in data. 1D Fourier scattering features coupled with an SVM have also proved to be effective, outperforming 1D wavelet scattering features [67]. These purely spectral methods improve upon the performance of simpler machine learning algorithms, like the application of SVMs on the 1D spectra of each pixel, but ignore the significant spatial structure present in HSI data.

2. A variety of methods have successfully used spatial information in the pre-processing steps (more rarely in post-processing, as well) to improve classification performance while still focusing on the spectral aspects of the data [43, 46, 69, 70, 71]. Ashitha *et al.* [71] classify 1D wavelet scattering features with an SVM after smoothing each channel of the HSI with 2D Gaussian filters. Sandwiching a spectral NN between two 2D Gaussian blur layers with trainable variance greatly improves performance and remains one of the most competitive HSI methods [46]. Lee *et al.* followed up on this work with a deep spectral NN with residual connections following a single 3D filter layer [69]. Acquarelli *et al.* made changes instead to the training process and included

a spatial term in the regularizer of a purely spectral 1D convolutional neural network (CNN) [43].

3. Spatial methods that include some spectral pre-processing have also been applied to HSI data [44, 45, 47, 52, 67, 72]. Classical 2D CNNs and Recurrent CNNs (RCNN) have been used on each channel of HSI input independently [72]. Also popular has been using PCA or other dimensionality reduction methods to reduce the number of channels in the the HSI before using a 2D CNN [44, 47, 60], or 2D wavelet scattering [52]. After using PCA to reduce the dimension, *Attribute Profiles* can be used to extract spatial features on each principle component and expand the dimension by a small amount [42]. These features capture morphological properties like area and shape per principle component, and the concatenation of many such 2D feature images can then be fed into a 2D CNN [59]. Our previous work performed competitively using 1D Fourier scattering preprocessing followed by 2D wavelet scattering [67]. Recently Deng *et al.* used a new CapsNet NN architecture [73] with 2D filters on each channel independently to achieve competitive results.

4. Integrated spectral-spatial methods which combine information from spectral signatures and spatial neighborhoods simultaneously are also common. Some papers consider sequences of 1D spectra [62], for example in a variety of NNs known as Long Short Term Memory (LSTM), or convert the HSI cube to a matrix and use standard 2D methods [74]. However, by far the most popular methods involve building 3D filters [44, 48, 49, 50, 51, 54, 55, 56, 72]. 3D

convolutional layers in NNs, CNNs, and RCNNs, both shallow and deep have been evaluated [44, 72]. But the lack of training data challenges models with many learnable parameters, and these networks struggled in comparison to methods with predetermined filters such as the 3D Gabor wavelets that Shen and Jia *et al.* used to extract features [49, 53, 54], and classify with a variety of algorithms, for instance a sparse representation based classification (3D WT+SRC) [54]. Bau *et al.* [48] used the real part of 3D Gabor filters sampled densely in the time-frequency domain to get features used with a Mahalanobis distance classifier. He *et al.* [50] decomposed the same filter into 8 subfilters, using only 3 to construct a discriminative low-rank Gabor mother filter (DLRGF) used to extract features, a hand designed feature which proved to be very competitive with a least squares based classifier (3D DLRGF+LS). Qian and Cao *et al.* [51, 55] used a Haar 3D wavelet filter bank (3D DWT-FB) and discrete wavelet transform (3D DWT) with various classifiers. Tang *et al.* [56] proposed a 3D Gabor wavelet scattering approach to extract features (3D WST), which decomposes the HSI across multiple wavelet scales and orientations and uses local averaging to keep class labels consistent in neighborhoods, and classified with a radial basis function SVM (3D WST+RBF-SVM).

The paradigm that we have ordered our review by has been the degree to which each method integrates spectral-spatial information, which greatly affects classification performance. Another characteristic that can be used to distinguish HSI methods, which we would like to mention here, is the amount of interaction

between the spectral-spatial features that each technique employs, as [42] points out.

In the terminology of [42], the simplest *dependency system* is the case where features are extracted directly from the HSI data. This encompasses the majority of the methods we presented. However, NNs with multiple layers naturally model a hierarchical interaction of features, with as many degrees of interaction as number of layers in the network: [43, 44, 45, 47, 62, 63, 64, 66, 69, 72]. The same can be said for the layers of scattering networks: [52, 56, 67, 71, 74]. This distinguishes these two classes of approaches from wavelet techniques [51, 54, 55, 75] or from time-frequency methods [42, 48, 49, 50], and yields more sophisticated features that provide improved classification results, as we demonstrate in Section 3.3.

3.1.2 Fourier Scattering Transform

We begin this section by formally defining the mathematical concept of the scattering transformation. Fix a sequence $\Phi = \{\phi, \phi_\lambda\}_{\lambda \in \Lambda}$ of square integrable functions on $\mathbb{R}^d$, where Λ is the index set of the sequence. Given an input function f defined on $\mathbb{R}^d$, we iteratively convolve it with this sequence and take the modulus in the following way. For each index $\lambda \in \Lambda$, let

$$U[\lambda](f) = |f * \phi_\lambda|,$$

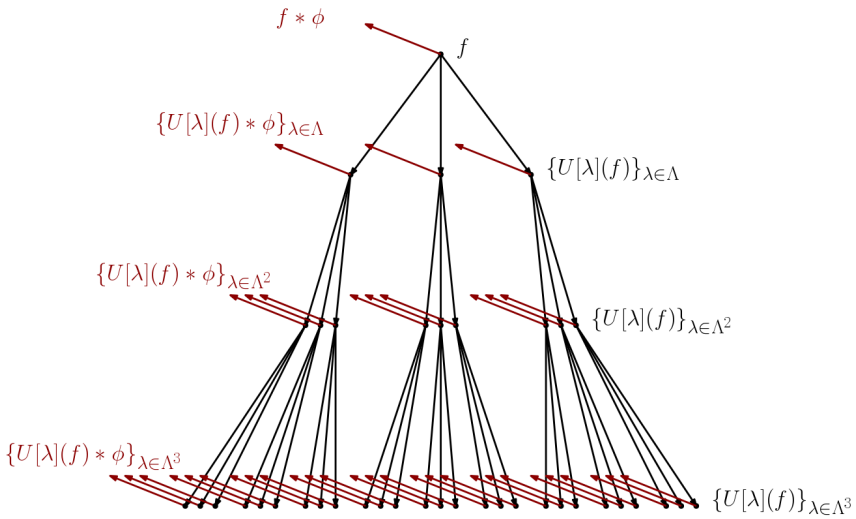


Figure 3.2: The network structure of the scattering transform for square integrable signals in a Hilbert space. The functions $U[\lambda](f)$ are generated iteratively and they are represented by the black dots. The scattering coefficients of f, represented by the red dots, are found by convolving each $U[\lambda](f)$ with ϕ.

where $*$ is the convolution of functions on $\mathbb{R}^d$. We can extend this rule to multi-indices. For each $\lambda = (\lambda_1, \ldots, \lambda_k) \in \Lambda^k$, let

$$U[\lambda](f) = U[\lambda_k] \cdots U[\lambda_1](f).$$

The *scattering transform* $\mathcal{S}_\Phi$ associated with Φ is formally defined as the sequence of functions

$$\mathcal{S}_\Phi(f) = \{f * \phi,\ U[\lambda](f) * \phi\}_{\lambda \in \Lambda^k, k \geq 1}.$$

See Figure 3.2 for a visualization of the scattering transform as a convolutional network.

The mathematical properties of the scattering transform and the features that it generates greatly depend on the underlying sequence of functions Φ. Mallat [61] and his collaborators [76] primarily considered the wavelet (time-scale) case, where ϕ

is the father wavelet and $\{\phi_\lambda\}_{\lambda\in\Lambda}$ are dilations of the mother wavelet function. The resulting transform is called the *wavelet scattering transform* (WST) and it provides a powerful multi-scale representation [76]. In contrast, we study the time-frequency analogue [57, 58], where ϕ is a band-limited function and $\{\phi_\lambda\}$ are frequency modulations of ϕ. The resulting transformation is called the *Fourier scattering transform* (FST) and it provides a novel hierarchical time-frequency representation of the data.

Although wavelet-based techniques have recently dominated the spectrum of signal processing applications in the field of HSI analysis, due to their overall impact on image processing, see e.g., [49, 51, 54, 65], the time-frequency methods form a natural foundation for spectral data exploration. They were the basis for the early Fourier transform imaging spectroscopy methods [77, 78], as well as for recent attempts to analyze hyperspectral imagery [50].

We note here that wavelet and Fourier scattering transforms provide entirely different representations: the WST computes localization and scale characteristics, whereas the FST provides frequency distribution information. Both transformations satisfy several similar representation-theoretic properties: they are energy preserving, non-expansive, and contract sufficiently small translations and diffeomorphisms, see the results in [57, 61] for precise statements. However, it is the FST that has a provable exponentially fast convergence of finite approximations, which is crucial in implementations. These properties explain why FST is an effective feature extractor. Indeed, as a mathematical construct, the scattering transform has an infinite number of layers and each node has infinitely many children. Thus, for practical applications we can only compute a finite subset of its coefficients. Thanks to the

aforementioned property, the FST can be truncated to a finite approximation without loosing its properties. Indeed, theoretical results guarantee that the total energy contained in the k-th order FST coefficients is at most ε^{k-1} of the original energy of f for some small $\varepsilon \in (0, 1)$, see [57].

We close this section by observing that, in a recent work, He *et al.* [50] introduced the concept of *discriminative low-rank Gabor filtering* (DLRGF) for spectral-spatial classification. The DLRGF is built upon a foundation formed by a 3D harmonic modulated with a 3D Gaussian - a concept which in mathematics is known sometimes as a *Gabor frame*. The mathematical representation system $U[\lambda](f) * \phi$ generated by the FST, using the iterative convolution process described above, is in fact a generalization of a Gabor frame and is known as a *uniform covering frame*. As such, our method provides the same theoretical guarantees for the analysis of HSI data as that of [50]. But the main difference between these two methods is the manner in which they are computed. DLRGF builds the representation using a priori filters, while our method constructs the efficient representation through an appropriately designed iterative procedure. The advantage of the latter is that it can be made significantly faster, even by orders of magnitude. The implementation process which enables this is described in the next section.

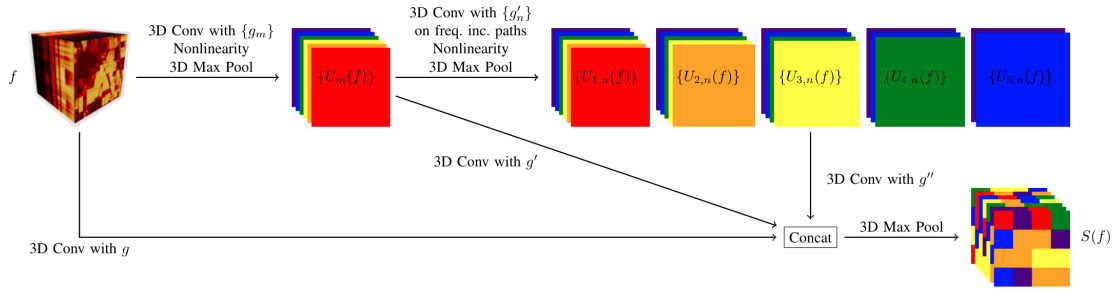


Figure 3.3: Our proposed 3D FST network for HSI data. The padded input f is the HSI cube is convolved with the collection of filters $\{g_m\}$ and the modulus nonlinearity is applied followed by a downsampling to create the first order intermediate 3D FST coefficients. These are in turn convolved along frequency increasing paths with $\{g'_n\}$, followed by a modulus nonlinearity and downsampling to create the second order intermediate 3D FST coefficients. Then the input and the intermediate coefficients are locally averaged with g, g' and g'', concatenated, and downsampled a final time to create our 3D FST representation $S(f)$. Because of the downsampling throughout the feature size is never significantly increased compared to the size of the input. This is in contrast to the network in Figure 3.2 where the number of scattering coefficient functions $\mathcal{S}_\Phi(f)$ grows exponentially with depth.

3.2 Proposed Discrete Fourier Scattering Network for HSI Classification

The FST is a generic transformation and a mathematical model that is suitable for many applications and purposes. To differentiate between the generic FST and the particular algorithm which we introduce in this chapter for HSI classification, we shall call the latter the *three-dimensional Fourier scattering transform* (3D FST) algorithm. We now give its detailed description.

In the context of HSI data, we consider its data dimension to be equal to $d = 3$ and we represent an HSI datacube as a function f defined on a 3-orthotope (i.e., rectangular box) subset of $\mathbb{Z}^3$. That is, $f(x, y, b)$ is the value of the image at spatial location (x, y) and spectral band b. Fix a function g on $\mathbb{Z}^d$, which is

typically called the *window function*. Following standard convention, we select the window g to be compactly supported in a 3-dimensional rectangle with side lengths $M = (M_1, M_2, M_3)$. Let Λ_M be the collection of $m \in \mathbb{Z}^3$ such that $0 \leq m_j \leq M_j - 1$ for each j. We define the functions $\{g_m\}_{m \in \Lambda_M}$ by the formula,

$$g_m(x, y, b) = \exp\left(2\pi i\left(\frac{xm_1}{M_1} + \frac{ym_2}{M_2} + \frac{bm_3}{M_3}\right)\right) g(x, y, b). \tag{3.1}$$

Here, x and y are the spatial coordinates and b is the spectral coordinate.

There is the usual trade-off with time-frequency representations: Larger values of M provide worse spatial localization but better frequency resolution, whereas a smaller M yields the opposite effect. For this reason, it is reasonable to use different functions in each layer of the network to maximize the performance of the transform. Let $M' = (M'_1, M'_2, M'_3)$ and $M'' = (M''_1, M''_2, M''_3)$ be multi-integers and let g' and g'' denote functions supported in rectangles of size M' and M'' respectively. We define $\{g'_m\}_{m \in \Lambda_{M'}}$ and $\{g''_m\}_{m \in \Lambda_{M''}}$ analogous to the definition of g_m given in equation (3.1), except with M' and M'' replacing M and g' and g'' replacing g, respectively.

Time-frequency representations are inherently redundant. We can downsample the features in such a way that we do not lose important information, e.g., see [79], and this type of result is closely related to the classical Shannon sampling theorem. In our case, we do not downsample in the spatial dimensions. We fix positive integers P, P', P'' which shall be the downsampling factors in the each of the three layers.

The zero order 3D FST coefficient, $S_0(f)$, is defined as

$$S_0(f)(x, y, b) = (f * g)(x, y, Pb).$$

Here, $*$ denotes the convolution operator on $\mathbb{Z}^3$. This is simply a local averaging of the input by the window function g and downsampled by P. The first order intermediate 3D FST coefficients are

$$U_m(f)(x, y, b) = |(f * g_m)(x, y, Pb)|.$$

The collection $\{U_m(f)\}_{m \in \Lambda_M}$ can be interpreted as the modulus of the *windowed Fourier transform* of f (also called the *short-time Fourier transform* in signal processing or the *spectrogram* in audio processing).

The windowed Fourier transform is not stable to small perturbations of the input function. The basic reason is that if f consists of a single high frequency component, then there exists a $\widetilde{f}$ such that $\widetilde{f}$ is a small diffeomorphism of f and its frequency support is disjoint from that of f; consequently, f and $\widetilde{f}$ are very different in both the L^2 metric, see [61, 80] for a rigorous analysis. To avoid this behavior in 3D FST we proposes to locally average $U_m(f)$ with g'. The first order 3D FST coefficients are then:

$$S_m(f)(x, y, b) = (U_m(f) * g')(x, y, P'b).$$

Hence, the first order 3D FST coefficients carry information about a spatially-

averaged short-time Fourier transform of f.

While naive local averaging improves stability to small deformations, it also removes a significant amount of high-frequency information because g is a low-pass filter. The lost components are aggregated in the functions $U_m(f) * g_n$. However, these functions suffer from the same instability properties as $U_m(f)$. The second order intermediate 3D FST coefficients are thus,

$$U_{m,n}(f)(x, y, b) = |(U_m(f) * g'_n)(x, y, P'b)|.$$

These intermediate coefficients are also unstable to small diffeomorphisms, so we perform a local averaging. The second order 3D FST coefficients are now defined to be:

$$S_{m,n}(f)(x, y, b) = (U_{m,n}(f) * g'')(x, y, P''b).$$

We also note that theoretical results in [57] guarantee that $S_{m,n}(f)$ is small when $n \geq m$, so we can improve the computational efficiency of the algorithm by only computing the coefficients for which $n \neq m$.

The zero and first order 3D FST coefficients can be interpreted as spatially-smoothed versions of classical spectral-spatial representations. It is not as obvious what the second order coefficients represent. At first glance, the second order 3D FST coefficients appear similar to the Mel-frequency cepstral coefficients (MFCCs), but there is an important distinction. The MFCCs are calculated by fixing the spatial coordinate and then further decomposing the spectrogram along the frequency

axis in log scale. MFCCs play an important role in audio analysis because more global characteristics, which are not captured by the spectrogram, contain important information.

In contrast to the MFCCs, the second order 3D FST coefficients are calculated by fixing the spectral variable and then further decomposing along the spatial coordinate. That is, $U_{m,n}(f)$ describes whether the m-th frequency of f over intervals of length M (a local property captured by the first order coefficients) varies at frequency n over intervals of length MM' (a more global property).

In summary, given a hyperspectral image f, the features generated by the 3D FST at location (x, y) are the collection of vectors

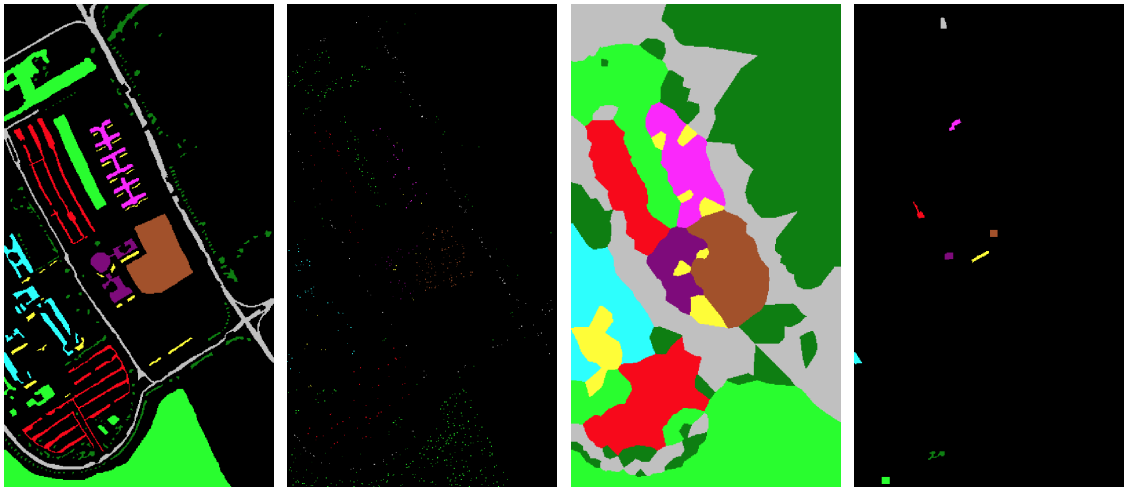
$$\begin{aligned} \text{Zero order:} \quad & S_0(f)(x, y, \cdot) \\ \text{First order:} \quad & \{S_m(f)(x, y, \cdot)\}_{m \in \Lambda_M} \\ \text{Second order:} \quad & \{S_{m,n}(f)(x, y, \cdot)\}_{m \in \Lambda_M, n \in \Lambda_{M'}}. \end{aligned}$$

These vectors are concatenated to form a feature vector for each pixel (x, y) of the hyperspectral image f.

In Figure 3.3 is a schematic of our proposed 3D FST method as we implemented it with standard tensorflow [35] primitives on the GPU: 3D convolutions with nonlinearities and 3D max pooling operations. This spatial domain implementation, as opposed to a frequency domain implementation, has the advantage of benefiting from the highly optimized tensorflow programming interface, which can distribute the necessary computations over ubiquitous and very powerful mod-

Table 3.1: Attributes of the HSI datasets we evaluate on.

Name	Satellite	No. Bands	Bandwidth	Meters per Pixel	Dimensions H×W	No. Labeled Pixels	No. Classes
PaviaU	ROSIS	103	430-860 nm	1.3 m	610x340	42776	9
IP	AVIRIS	224	400-2500 nm	3.7 m	145x145	10249	16
KSC	AVIRIS	224	400-2500 nm	18 m	512x614	5211	13
Botswana	NASA EO-1	242	400-2500 nm	30 m	1476x256	3248	14



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

Figure 3.4: An example using Pavia University to illustrate strictly site specific sampling for creating training masks. (a) Ground Truth Labels. (b) 2% of labels randomly selected. (c) 1-KNN interpolation of the labels in (b) which is correct for 93% of all labels. (d) 2% of the labels selected in a strictly site specific manner. Note how there is only 1 site per class label with all its pixels in a connected set. The 1-KNN interpolation of the labels in (d) would be correct for 22% of all the labels.

ern GPUs. The scattering layer that we programmed can also thus be seamlessly blended into any stage of a deep network since it allows backpropagation, we leave such integrations to future work. Figure 3.3 also illustrates how our method can be performed on an input of any size since it maps any HSI cube to an HSI-feature cube with equal spatial dimension. We release our implementation publicly [81] and discuss its runtime performance in Section 3.3.5.

3.3 Experimental Results and Discussion

3.3.1 Data Sets

We test the performance of these feature extractors on the following hyperspectral databases:

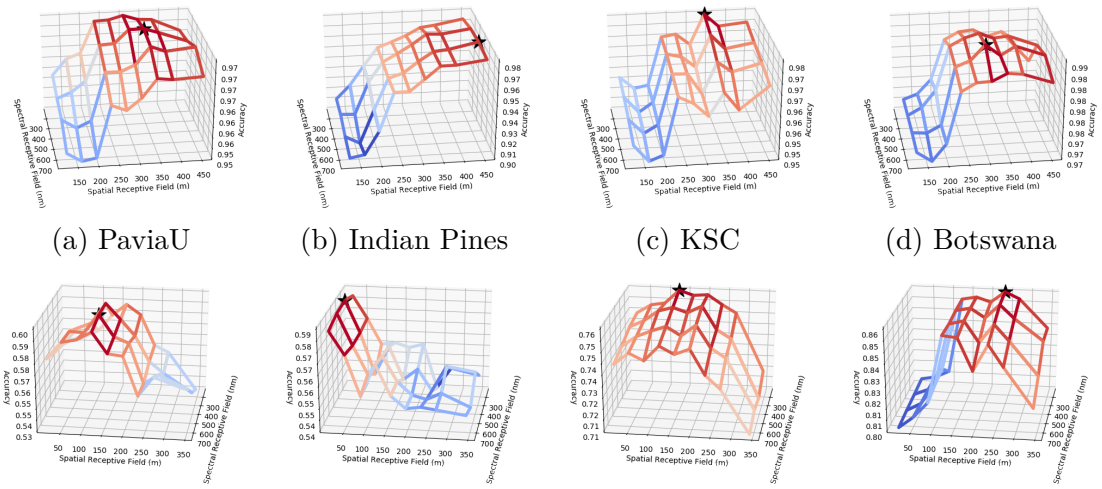
- *Indian Pines (IP)* acquired over the Indian Pines test site in Northwestern Indiana in 1992 by the Airborne Visible / Infrared Imaging Spectrometer (AVIRIS) sensor [82]. 49% of all pixels are labelled.

- *Pavia University (PaviaU)* acquired during a 2001 flight campaign over Pavia, northern Italy, using the reflective optics system imaging spectrometer (ROSIS) sensor [83]. 21% of all pixels are labelled.

- *Kennedy Space Center (KSC)* acquired over the wetlands on the west shore of the Kennedy Space Center (KSC) and the Indian River using the AVIRIS sensor [84]. 1.7% of all pixels are labelled.

- *Botswana* acquired over the Okavango Delta, Botswana in 2001, by the Hyperion sensor on the NASA EO-1 satellite [85]. 0.86% of all pixels are labelled.

Table 3.1 shows additional information on all datasets. All datasets can be downloaded from the webpage [86], and we also make them available in our released code [81].



(a) PaviaU (b) Indian Pines (c) KSC (d) Botswana

(e) PaviaU (SSS) (f) Indian Pines (SSS) (g) KSC (SSS) (h) Botswana (SSS)

Figure 3.5: Gridsearches over the hyperparameters M, M', M'' of the 3D FST for each dataset we analyze. The vertical axis is the average classification accuracy over 10 trials. Each dataset generates a unique shape because of the various degree of spatial or spectral noise that should be smoothed. Larger receptive fields perform more smoothing. Each of the datasets in the second row is constructed in a single site specific manner as elaborated in the main text. Each of the datasets in the first row is constructed with the more typical random distributed sampling. A black star marks the best result per gridsearch which determine the hyperparameters we use with our 3D FST method.

3.3.2 Methods Compared

To better evaluate the performance of our proposed method we implement three state of the art feature HSI classification methods, and score all methods in the same exact training and testing conditions. We release our tensorflow implementation of these methods with the rest of our code [81].

- *PCA + Deep Learning (DFFN)* We re-implement a Deep Feature Fusion Network exactly to the specification of [60], which is publicly available in a caffe implementation [87]. This method projects the HSI data to a few PCA com-

ponents and passes windows to a deep network consisting of three towers with 4 or 5 residual convolution blocks. This is a very deep network that in the case of PaviaU contains 34 2D convolutional layers. It has a large receptive field of 23 or 25. With our implementation we are able to replicate the same performance as in the original paper for the Indian Pines and PaviaU datasets [87].

- *Extended Attribute Profiles + Deep Learning (EAP)* We implement a Deep Learning With Attribute Profiles method to the specification of the EAP-Area method in [59]. This method projects the HSI data to 4 PCA components and then creates APs of length 4 for each component. With a receptive field of 9, the $9 \times 9 \times 36$ cube is passed to a neural network containing 3 2D convolutional layers and 2 Fully Connected layers before a softmax classifier. In our implementation we exceed the performance in the original paper for the PaviaU dataset [59].

- *3D Wavelet Scattering Transform (WST)* We implement a 3D Wavelet scattering approach to the specification of [56]. It is a 2 layer network with $7 \times 7 \times 7$ wavelet filters (receptive field of 19) with 9 orientations and 3 scales per layer. We use the same strategy of downsampling that we employ in our 3D FST in our implementation to enhance performance. We classify the features with a linear SVM and are able to match the performance in [56] for our datasets.

The DFFN and EAP methods are Deep Learning methods for which we perform from-scratch training for every single training dataset. In this chapter the

results we report are from hundreds of training runs for DFFN and EAP. To make sure we train the best neural network possible for each trial, we extract a validation set from each trial's test set, and we train until validation loss is non-decreasing for 20 consecutive epochs. We evaluate every 2 epochs and save the model with the highest validation accuracy. For overall accuracy we report the accuracy on the original test set. We use the same batch sizes as in the original papers [59, 60], and adjust the learning rate (in the range 0.1 to 1e-5) to ensure convergence on our training sets of a variety of sizes. We use the Adam optimizer for training all of our networks [36]. Further details are available in our released source code [81]. We do not use data augmentation for any method. For additional comparison we also include additional popular methods:

- *3D Gabor Filters* Three dimensional Gabor filters are a popular feature extractor for HSI classification [48, 53, 54]. In fact, a truncation at the first layer of our proposed 3D Fourier Scattering feature extractor ($U_m(f)$) is equivalent to Gabor filtering. Thus we compare to such a truncated version of the FST on each dataset, using the same parameters for the Gabor filters as used in the the first layer of our scattering filters. This allows us to directly see what additional information the following layers of the FST extract to aid classification. After extracting the Gabor filters, we use a linear SVM for classification.

- *Raw features* We denote this in the text as the *Raw* method. It is simply the raw spectrum of a pixel used as a feature vector for linear SVM classification.

Every time that we use an SVM for classification, it is linear with the regular-

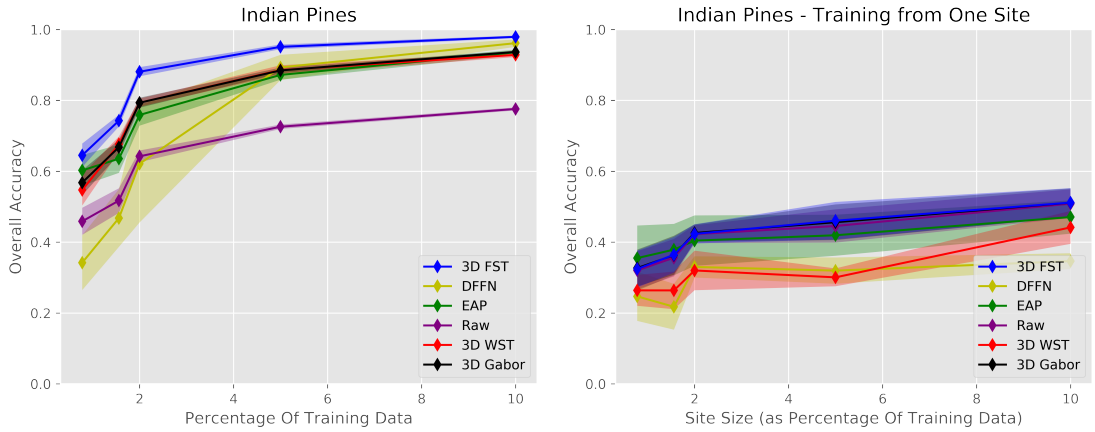


Figure 3.6: Performance of the 6 implemented methods on randomly sampled training sets from Indian Pines. Each point is the average of 10 trials, the shading is the standard deviation over the 10 trials. Each method is tested on the same 10 training sets per training set size. 3D FST performs optimally in all training scenarios on randomly distributed data. SSS training is noisy and low performance for this dataset, FST provides a slight benefit over Gabor features.

ization parameter fixed to $C = 1000$. We chose this parameter by cross-validating for $C = 10^{-10}, 10^{-9}, ..., 10^{10}$ on Raw SVM, where we found $C = 1000$ leads to the best Raw+SVM overall classification accuracy.

3.3.3 Hyperparameters of 3D FST

In this section we perform a robustness analysis on the hyperparameter values of 3D FST and discuss the performance impact of various choices of spatial and spectral window sizes. The promise of deep learning methods like DFFN or EAP is that they learn their parameters to adjust to the dataset. With 3D FST a choice of several parameters serve the same crucial function before the features extracted with it are fed to a learning classifier (a linear SVM in this chapter). When using the Fourier scattering transform on HSI data, there are several parameter choices that impact the effectiveness of our method. Recall that g, g', g'' are the window functions

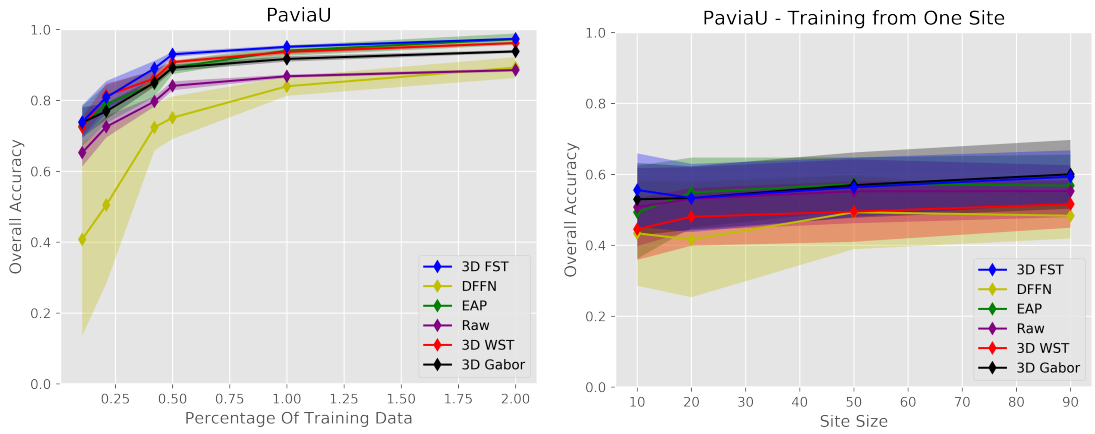
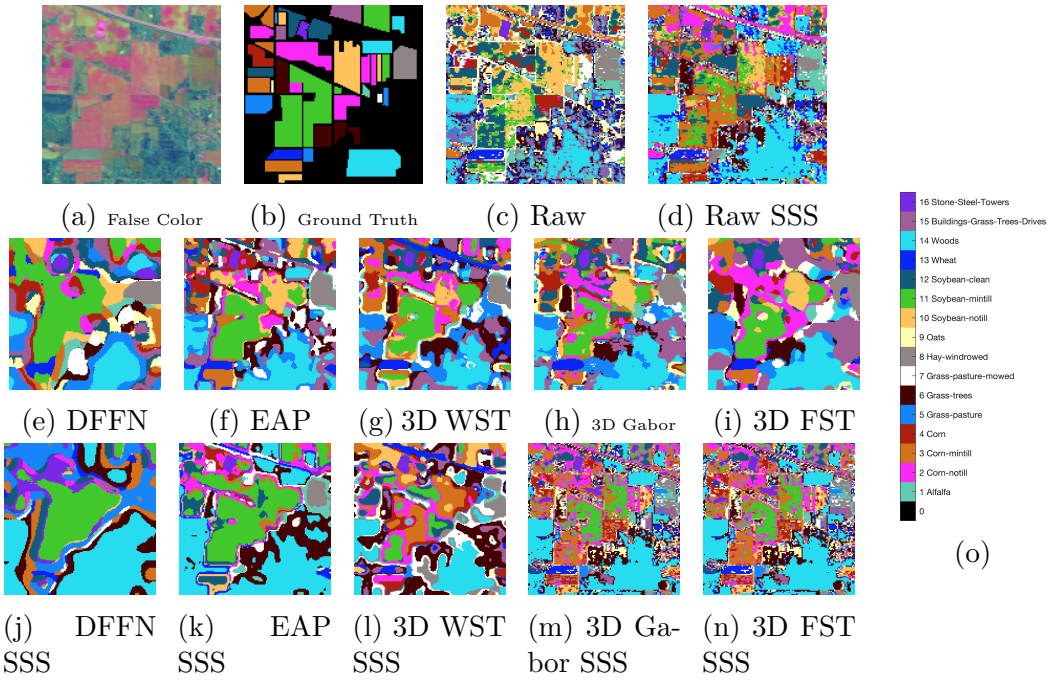


Figure 3.7: Performance of the 6 implemented methods on randomly sampled training sets from PaviaU. Only DFFN, which is the deepest method by far, does not clearly surpass raw features in this setting of very limited training data. SSS training very noisy, and the less deep networks tie, and Wavelet features underperform Fourier features.



(a) False Color (b) Ground Truth (c) Raw (d) Raw SSS

(e) DFFN (f) EAP (g) 3D WST (h) 3D Gabor (i) 3D FST

(j) DFFN SSS (k) EAP SSS (l) 3D WST SSS (m) 3D Gabor SSS (n) 3D FST SSS

(o)

Figure 3.8: Full classification for Indian Pines with 5 samples per class (0.8% of training data). The best performing trial by classification accuracy per model is plotted. The classification performance for these methods is in Figure 3.6.

used in each layer of the 3D FST, where M, M', M'' are the size of their supports, and P, P', P'' are the downsampling parameters. The most impactful parameters to choose are M, M', M''. A large spatial support of the FST windows will lead to larger spatial features being extracted, as well as more spatial blurring. The size of the FST window supports in the spectral domain will also lead to a greater range of frequency features extracted in the spectrum and also more averaging in the spectrum. These hyperparameters can be selected to best exploit the different physical conditions that a dataset was collected according to, like meters/pixel and sampling rate in the spectral domain. We would like to choose M, M', M'' once per dataset, and a natural approach is to choose the M, M', M'' that yield the best classification score on a validation set. However, we also have to be careful when we create our validation set to ensure that M, M', M'' will not be influenced by artificial biases created by picking this validation set.

Uncontrolled random sampling to create a training set of HSI pixels has been previously observed to create a positive bias to methods that use larger spatial neighborhoods [43, 88]. For methods that extract features from neighboring pixels to perform a classification on a single test pixel, if the neighborhood includes a pixel that was present in the training set, then the training and testing features will overlap in the spatial domain. A larger neighborhood used to extract features will lead to a greater overlap between training and testing neighborhoods and thus a greater similarity between training and testing features. Thus sampling a HSI image randomly and choosing the spatial window size to use for feature extraction based on classification performance can lead to an artificial preference for large windows.

This exact effect is displayed in Figure 3.5. In the first row of Figure 3.5 we see a trend that larger spatial windows lead to higher overall accuracy on the test set when the training set is randomly distributed across the image.

To limit this preference for larger spatial neighborhoods we can compare to a sampling strategy similar to [88] for picking the training set. In contrast to sampling randomly from the labels to create a training set like in Figure 3.4b, we may sample in a local manner according to connected components as discussed in [88]. To create a training set, we can instead pick a single pixel per class, and add pixels to our training set only when they are directly adjacent to already selected pixels of the same class. As long as we do not request too many pixels to be in our training set, this method leads to a single site of training pixels per class, and we refer to such a training set as *strictly site specific* (SSS). The largest such training set we create for this chapter (90 samples per class) is shown in Figure 3.4d (90 samples per class is about 1.9% of training data for PaviaU). Note how when sampling 2% of pixels randomly in Figure 3.4b a 1-KNN classifier on the pixel coordinates will perform at 93% accuracy as shown in Figure 3.4c while for a SSS dataset of equal size the same type of classifier will perform at 22% accuracy. This shows that neighborhood overlap in feature generation has the potential to inflate classification accuracy by a large degree for the randomly sampled training set, but not for the strictly site specific training set.

The grid searches over hyperparameters M, M', M'' that we perform are shown in Figure 3.5, the first row contains grid search results on randomly distributed training data, and the second row contains grid search results on SSS datasets which

limit unnecessary spatial bias. For the SSS datasets, for PaviaU we build training sets with 9 sites of size 90 (1.9% of training data), for Indian Pines we use 16 sites sized such that 10% of the data is used for training, for KSC 13 sites of size 20 (5% of training data), and for Botswana 14 sites of size 20 (9% of training data). The gridsearches for randomly distributed data in the first row of Figure 3.5 use the same amount of data as the SSS gridsearches. For each point in the grid we perform 10 trials each on a different training set, and plot the mean accuracy. 3,200 experiments are summarized in Figure 3.5. Each selection of M, M', M'' has 6 degrees of freedom (we use square spatial windows), and each individual window we test has a support of 3,5,7, or 9 for randomly distributed data, and 1,3,5,7 for SSS data. We plot the spatial receptive field (proportional to $M_1 + M_1' + M_1''$) and the spectral receptive field (proportional to $M_3 + M_3' + M_3''$) against the mean accuracy. We find that the general shape of the loss surface does not change as we vary the size of the training sets but is smoother for larger training sets. The best hyperparameters M, M', M'' are:

- *PaviaU.* $7\times7\times5, 7\times7\times5, 7\times7\times5$ which has a receptive field of 24 m by 179 nm (19 spatial samples by 43 spectral samples). For a strictly site specific setting for this dataset we choose the hyperparameters $3 \times 3 \times 7, 3 \times 3 \times 7, 3 \times 3 \times 7$ which has a receptive field of 9 m by 254 nm (7 spatial samples by 61 spectral samples).

- *Indian Pines.* $9\times9\times7, 9\times9\times7, 9\times9\times7$ which has a receptive field of 92 m by 571 nm (25 spatial samples by 61 spectral samples). For a strictly site specific

setting for this dataset we choose the hyperparameters $1\times1\times5, 1\times1\times5, 1\times1\times5$ which has a receptive field of 3 m by 403 nm (1 spatial samples by 43 spectral samples).

- *KSC*. $7\times7\times3, 7\times7\times3, 7\times7\times3$ which has a receptive field of 342 m by 234 nm (19 spatial samples by 25 spectral samples). For a strictly site specific setting for this dataset we choose the hyperparameters $7 \times 7 \times 3, 1 \times 1 \times 3, 1 \times 1 \times 3$ which has a receptive field of 126 m by 234 nm (7 spatial samples by 25 spectral samples).

- *Botswana*. $9 \times 9 \times 7, 5 \times 5 \times 7, 5 \times 5 \times 7$ which has a receptive field of 510 m by 529 nm (17 spatial samples by 61 spectral samples). For a strictly site specific setting for this dataset we choose the hyperparameters $5\times5\times3, 5\times5\times3, 5\times5\times3$ which has a receptive field of 390 m by 216 nm (13 spatial samples by 25 spectral samples).

Some general patterns we observe are that a larger spectral receptive field is preferred for randomly distributed training, and that the spatial window should only decrease in size in the network. In the results of the grid search, we observe the intuitive trade off in making the spatial receptive larger: that noise is reduced, but as a result of blurring the ability to resolve details is lost, hence the optimal value is a spatial window that is neither too large nor too small. In the SSS gridsearches we see that the largest spatial receptive field does not perform the best, hence we conclude that picking parameters with such a search yields a feature extractor that avoids unnecessary spatial bias. This is in line with our expectations since in a

randomly sampled training dataset like in Figure 3.4b, for every test pixel, as we increase the size of the spatial window in which to extract features, the number of training point features that we mix with the test point features grows. But for the SSS training dataset like in Figure 3.4d this is not true for most test points which are not near the single class sites.

The most dramatic difference in hyperparameters for the two sampling strategies is for Indian Pines, where the largest possible spatial window performs best with randomly distributed data, and the smallest possible spatial window performs best with single site data, yet number of spectra samples used does not dramatically differ. This is likely to do with the density of Indian Pines: 49% of pixels have labels (compared with 21%, 2%, and $> 1\%$ for PaviaU, KSC, and Botswana) and because of the human-planned agricultural setting all the pixels belonging to a class are often adjacent. Thus with a randomly distributed training set, growing the window of features around a test pixel often incorporates features from training pixels nearby. At the same time for strictly site specific training sets, growing a window around a test pixel that is not near training pixels can incorporate features from the wrong class since the fields of different crops are nearby.

The receptive field sizes that our grid search picks are generally middle of the range compared to the literature. The EAP method we compare to has a smaller spatial receptive field of 9 while the DFFN has a larger receptive field of up to 25, and the 3D WST has a similar receptive field of 19.

Looking at Figure 3.5 performance varies much less along the spectral dimension than the spatial, and a middle-ground value for the spectral window is often

picked over an extreme value. For PaviaU, which has the fewest spectral bands of all the datasets and most diverse material classes, switching from larger to smaller windows for randomly distributed data versus SSS data causes a preference for a larger spectral window, perhaps to compensate for the decreased spatial features. For Indian Pines where 14 of 16 classes are types of vegetation, we see that the best performing spatial windows have similar performance in the spectral dimension. For KSC and Botswana a smaller spectral window is preferred in general, perhaps also because of the material similarity of most classes. We also note that for all the best hyperparameters that $M_3' = M_3'' = M_3'''$, that the spectral receptive field should be equally distributed throughout the layers of scattering.

Regarding the other hyperparameters, we keep them constant throughout our experiments. Higher downsampling improves the speed of our method but decreases the performance, so we fix P, P', P'' to be as low as possible without making our method prohibitively slow (also note that the higher the downsampling the higher the receptive field of the network). Setting $P = M-2$, $P' = M'-2$, and $P'' = M''-2$ achieves this balance. For simplicity, for $g = g' = g''$ we use the rectangular window. The parameters listed in this section are used for 3D FST experiments throughout the rest of the chapter, though we do note that if we only wanted to improve classification accuracy for randomly sampled distributed training sets, the larger the spatial window the better the accuracy, but only for the reasons of spatial bias noted in this section. The best representation of HSI data is achieved with the M, M', M'' triplets listed above.

3.3.4 Analysis

In this section we compare our methods with both the standard random sampling method and strictly site specific (SSS) training datasets for Indian Pines, PaviaU, KSC, and Botswana. PaviaU and Indian Pines were selected for their popularity, and have a very geometrically structured ground truth. In contrast KSC and Botswana have a more abstract ground truth that has a similar appearance to the SSS sampling strategy.

On Indian Pines our proposed method has the highest classification accuracies for all the training sets used between 5 samples per class to using of 10% of the data, as seen in Figure 3.6. In the full classification maps shown in Figure 3.8 for the highly limited training data scenario of 5 samples per class we can observe many interesting details. For the randomly distributed datasets, DFFN and 3D FST have the same receptive field size, but DFFN blurs classes together much more, it only begins to perform well and above Raw features at 5% of training data. When it comes to using the SSS dataset, the spatial blur seems to increase, showing that the large spatial receptive field cannot adapt to SSS data in Indian Pines. The extra layers of convolutions and nonlinearities that 3D FST adds to 3D Gabor pay off in overall accuracy, and leads to slight spatial blurring in the distributed datasets. For SSS data 3D Gabor and 3D FST are not much different than Raw features, since the hyper-parameter search suggested a spatial window size of 1. EAP has a good balance between blurring and maintaining detail for this dataset and ties with many other methods. 3D WST has the roundest features of all the methods.

We note that for the SSS datasets all methods struggle, and the results are noisy. In some instances the performance can decrease as the site size grows (this happens with the lowest performing methods on SSS data, 3D WST and DFFN, several times). The standard deviation of all the methods for SSS data is also very high and prevents seeing a clear dominating method. Our explanation for this is that the SSS setting is very challenging because of the great variability between different SSS datasets. Since each SSS dataset contains only a single site per class, there is little information on the overall shape of a class, or number of locations in the image a single class may be, and it is rare for training pixels to be different classes but near. Though we believe an artificial bias for spatially smoothing feature extractors is diminished with SSS sampling, this dataset construction may leave little in the training data to be exploited other than the spectral information of each pixel, which would explain the consistent high performance of Raw single-pixel spectral features.

The classification accuracy for our proposed method on PaviaU for randomly sampled data is about the same as EAP and 3D WST at 2% of training data, but performs the best at lower training data percentages and we see more details are preserved in the classification maps in Figure 3.9. We see a similar pattern in overall accuracy in Figure 3.6 to Indian Pines, in that the deepest neural network method has the biggest gains as the amount of training data increases. For PaviaU the filters for 3D FST are a slightly different shape than the $7 \times 7 \times 7$ cube filters of 3D WST, and the two methods perform within a margin of error of each other. However in the full classification maps in Figure 3.9 we see that 3D WST has more spatial smoothing than 3D FST: details like the separation between the painted

metal sheets in the red rectangle and the tree shadows in the blue rectangle are clearer in 3D FST. Likely because this dataset has more thinly articulated features such as the self-blocking bricks parking lots our gridsearch chose a smaller receptive field for this dataset compared to Indian Pines. We also notice in Figure 3.9 that 3D FST best preserves details that we see in Figure 3.9a, which are also clear in the SVM full classification maps, albeit with salt and pepper noise (though Raw misses the Bitumen and Gravel classes). We also see an improvement over 3D Gabor with the additional layers of 3D FST. The shapes in the distributed datasets become less dithered in the distributed training sets.

The SSS scenario is much more challenging as evident when comparing the performance of various methods on PaviaU at 90 samples per class (roughly 2% of training data) in Figure 3.7. The performance in the SSS scenario varies much more wildly between training sets, even though the number of trials performed stays the same as with the randomly sampled datasets. We see that the deep learning methods DFFN and EAP are most affected by the change from a distributed randomly sampled training set to a single site training set. EAP stops correctly classifying the lower parking lot of PaviaU, and the shapes in DFFN become more distorted. 3D FST is the least affected by the shift to a SSS training set. The features in the red, yellow, and blue boxes of Figure 3.9a change only slightly, and the thin asphalt roads are maintained. No single method edges out ahead by much consistently for the PaviaU SSS datasets. We again see that performance does not increase by much as the size of the single site grows. The greatest misclassification error comes from missing large components like the bare soil block in the middle

and the continuous meadows.

For KSC and Botswana the SSS sampling method mirrors the structure of the ground truth much more closely (compare the ground truths in Figures 3.12b and 3.13j with the training set in Figure 3.4d). In effect when we create our training sets for KSC and Botswana we slice away a piece of just one of the connected components per class in the ground truth. Perhaps for this reason for KSC and Botswana we see a slightly clearer margin of better performance for 3D FST. The simple SVM on the Raw spectrum remains competitive, however the smoothing of 3D FST and 3D Gabor are advantageous over the raw SVM for Botswana. For KSC we see among the methods a variety of different levels of smoothing around the blue rectangle of Figure 3.12a, where the separation between mud flats and water is ambiguous. DFFN with its large receptive field blurs the bridge shapes, and EAP mostly erases the islands in the middle body of water.

It is interesting to see how the extra layers of 3D FST add more detail into the 3D Gabor images for KSC and Botswana for both distributed and SSS training. For KSC we can see more articulation in the SSS dataset especially. For Botswana we see a reed border added to the main body of water. But in general the structure of the ground truth besides the water is especially hard to tell from the false color images of KSC and Botswana, and all but the deepest DFFN give abstract results that look appealing but have a different amount of smoothness.

Spatial smoothing in our proposed method is largely influenced by the choices of M, M', M''. The amount of smoothing our grid search found for FST is on the whole less than other state of the art methods for distributed data, even though the

Table 3.2: Feature extraction and SVM computation time on the whole image.

	IP	PaviaU	KSC	Botswana
3D FST Feat. (s total)	9	23	75	71
3D FST Feat. (pixels/s)	2,300	9,000	4,100	5,300
SVM for 3D FST (s total)	5	41	40	90
SVM for 3D FST (pixels/s)	4,200	5,000	7,800	4,200

spatial receptive field is similar to the literature. For SSS data, the spatial smoothing suggested by our gridsearch is at a near minimum, and occasionally edges above the performance of Raw features for this challenging setting.

Finally we look at metrics beyond overall classification scores. Section 3.3.4 shows confusion matrices on our proposed method, and a runner up method on all four datasets at a median amount of data. We see no deviation in any conclusions drawn from overall accuracies since the performance is so high in this setting. No small classes suffer a great amount. Looking across the diagonal of per class accuracies between the two most competitive methods we see a great deal of similarity of performance, though on the aggregate 3D FST inches ahead.

3.3.5 Computational Cost

The runtime performance of our 3D FST is in Table 3.2. In our implementation we classified one patch of pixels (51×51) at a time, which greatly accelerated processing compared to processing one pixel at a time. The average rate that we extracted features from the 4 datasets tested was 5,200 pixels per second. In the classification setting, after all the pixels were processed, a linear SVM was trained, and then the test samples were classified. The average rate that we classified pixels

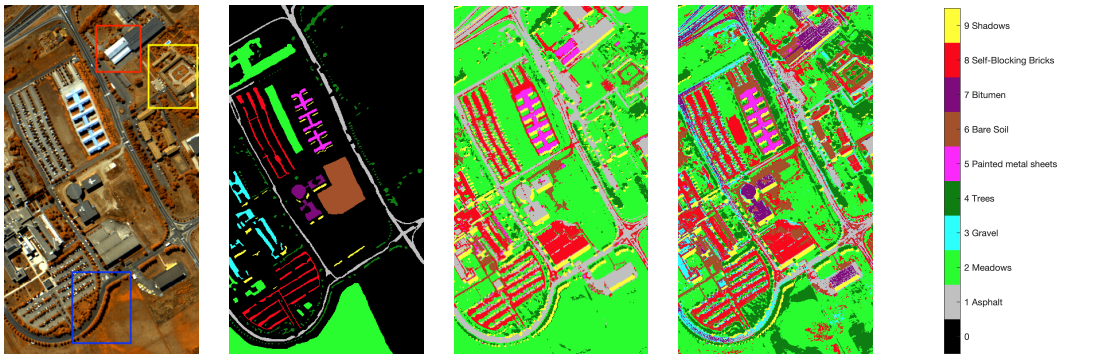
across the 4 datasets was 5,300 pixels per second. The SVM performance numbers in Table 3.2 are computed from the test SVM, and the training was always faster (by at least one order of magnitude) than the testing of the SVM. The major factor in performance was not the filter size directly, but the number of filters used in total, which is influenced by our windows size and downsampling choices. In our experiments the number of filters per layer was proportional to the product of the size of the supports of the fiters in each dimension, for example in the first layer the number of filters was $M_1 \times M_2 \times M_3$. Since the SVM did not take much time total, we saw no need to run PCA or any other form of dimension reduction reduction between feature extraction and classification, though it could have reduced memory usage by throwing away some coefficients. Needless to say the supervised training of the neural networks of EAP and DFFN took much longer to train than the feature extraction of 3D FST and 3D WST. Supervised neural networks also need to be retrained for different training/validation/testing set partitions, while the 3D FST of an HSI cube can be saved once for each dataset, and sampled later. Our downsampling strategy for 3D FST reduced the feature size by about 20x in comparison to the original 3D WST in [56]. All our methods were performed on a NVidia Titan X GPU with 12 GB of memory.

3.4 Conclusion

In this chapter we have proposed a three-dimensional Fourier scattering transform for HSI classification. This method has the neural network like benefits of

hierarchical feature extraction while bypassing the training process which is computationally expensive in both the amount of required training data and training time. Our three dimensional time-frequency features are well suited for HSI data since they decompose the HSI into multi-frequency bands and remove small perturbations such as noise. The 3D FST is particularly effective when there is limited training data. As supported by the experimental results, 3D FST achieved SoA performance on benchmark datasets, all while executing within a few minutes on a conventional GPU, and using a simple linear SVM for classification.

An advantage of our method is its compatibility with conventional deep learning implementations. This readily allows for a shift from the pre-processing based classification with a linear SVM we presented to an end-to-end feature extraction with a classification deep network. This has the potential to improve classification performance further as both the classification and feature filters will be learned for each dataset, and opens the door for integrating our method with other deep learning techniques like transfer learning or meta learning. Our future work investigates this hybridization of scattering transforms with deep learning where the classification is performed with a neural network following a tune-able scattering transform that serves as a feature extractor, and both are trained jointly. We also leave to future work a task even more challenging than classification from limited data: classification when training data is only available from other datasets. In this case the 3D FST and the classifier will be adapted to a new dataset where training data may not be available at all.



(a) False Color (b) GT (c) Raw (d) Raw SSS (e) Labels

(f) DFFN (g) EAP (h) 3D WST (i) 3D Gabor (j) 3D FST

(k) DFFN SSS (l) EAP SSS (m) 3D WST SSS (n) 3D Gabor SSS (o) 3D FST SSS

Figure 3.9: Full classification on PaviaU. The middle row is trained with 2% of training data with samples randomly selected. The classification performance for these methods is in Figure 3.7. The bottom row is trained with 90 samples per class (1.8% of training data) Strictly Site Specific. The best performing trial by classification accuracy per model is plotted. Though the training set sizes are roughly the same in size, distributed versus site specific training shows a big difference in full classification maps, and an even bigger performance in classification accuracy: with distributed sampling the methods perform with over 90% OA versus with SSS the methods perform with less than 60% OA (see Figure 3.7). The courtyard in the yellow rectangle, the asphalt gap between the painted metal sheets of the red rectangle, and the shadows in the blue rectangle are well preserved in the 3D FST images.

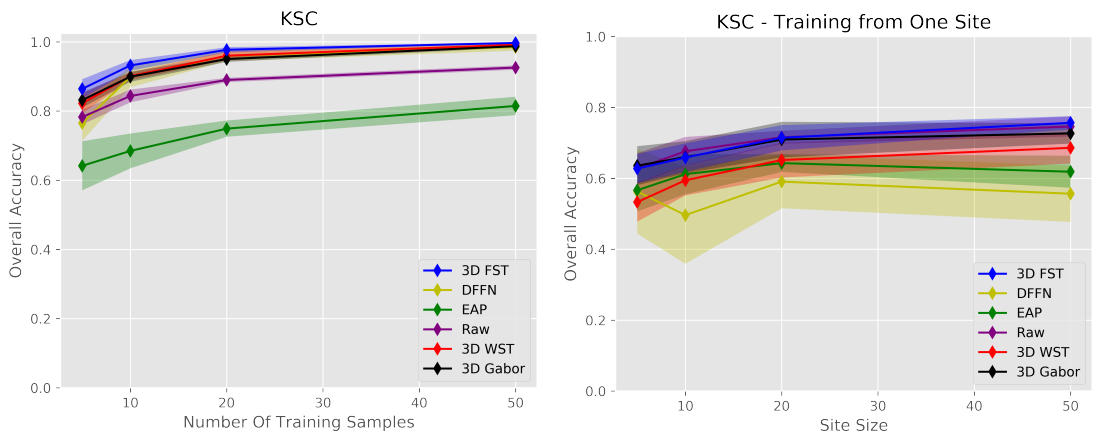


Figure 3.10: Performance of the 6 implemented methods on randomly sampled training sets from KSC. Performance converges quickly at just 50 training samples per class, but FST keeps an edge. For SSS datasets the Fourier features of 3D FST and 3D Gabor methods lead to the best performance.

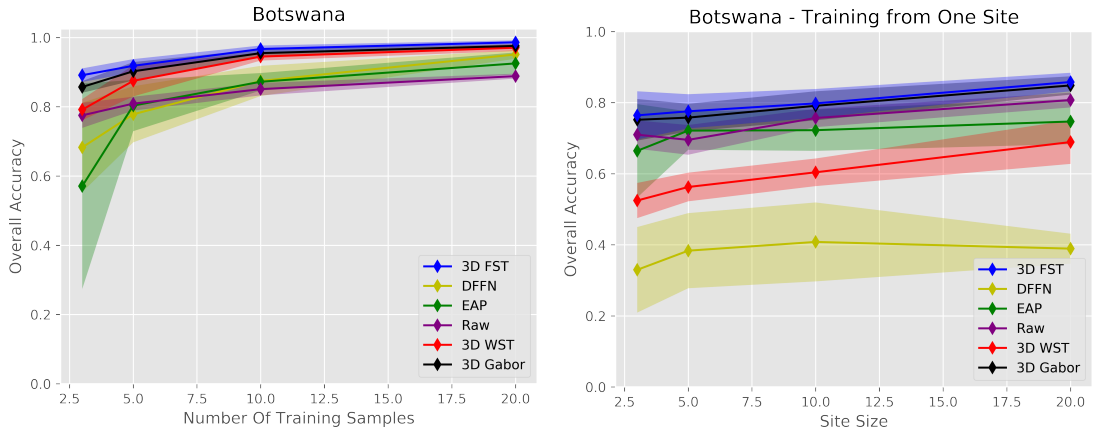
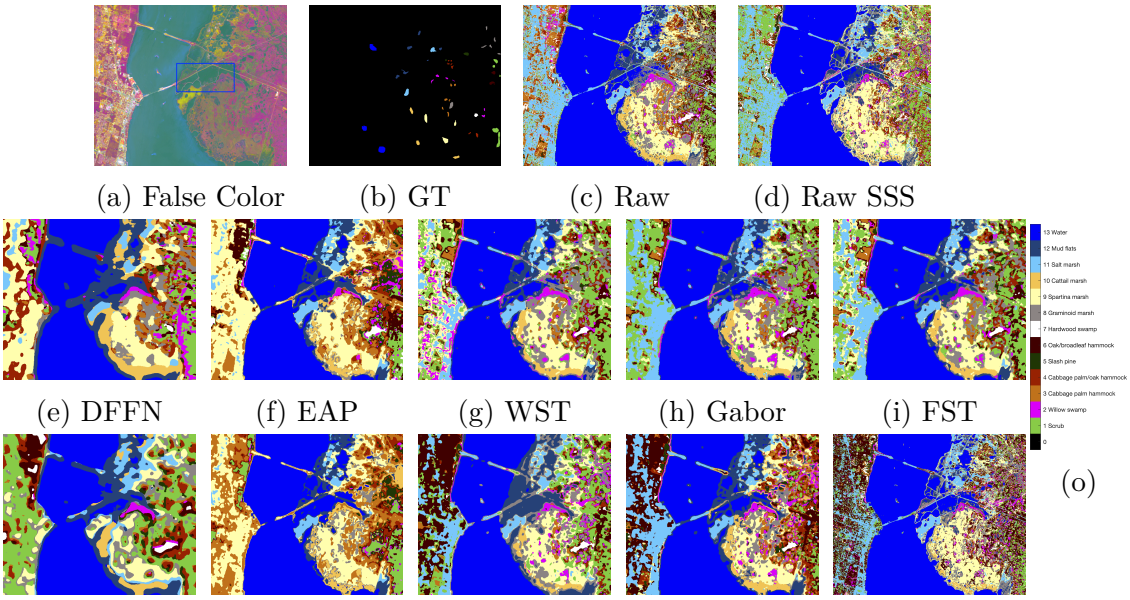


Figure 3.11: Performance of the 6 implemented methods on randomly sampled training sets from Botswana. Performance is very high on the very small datasets tested. The SSS sampling strategy has the biggest spread of any dataset, but the results are consistent in that 3D FST, 3D Gabor, and Raw features are all competitive with each other.



(a) False Color (b) GT (c) Raw (d) Raw SSS

(e) DFFN (f) EAP (g) WST (h) Gabor (i) FST

(o)

(j) DFFN SSS (k) EAP SSS (l) WST SSS (m) Gabor SSS (n) FST SSS

Figure 3.12: Full classification on KSC with 50 samples per class sampled randomly and in a Strictly Site Specific way. The best performing trial by classification accuracy per model is plotted.

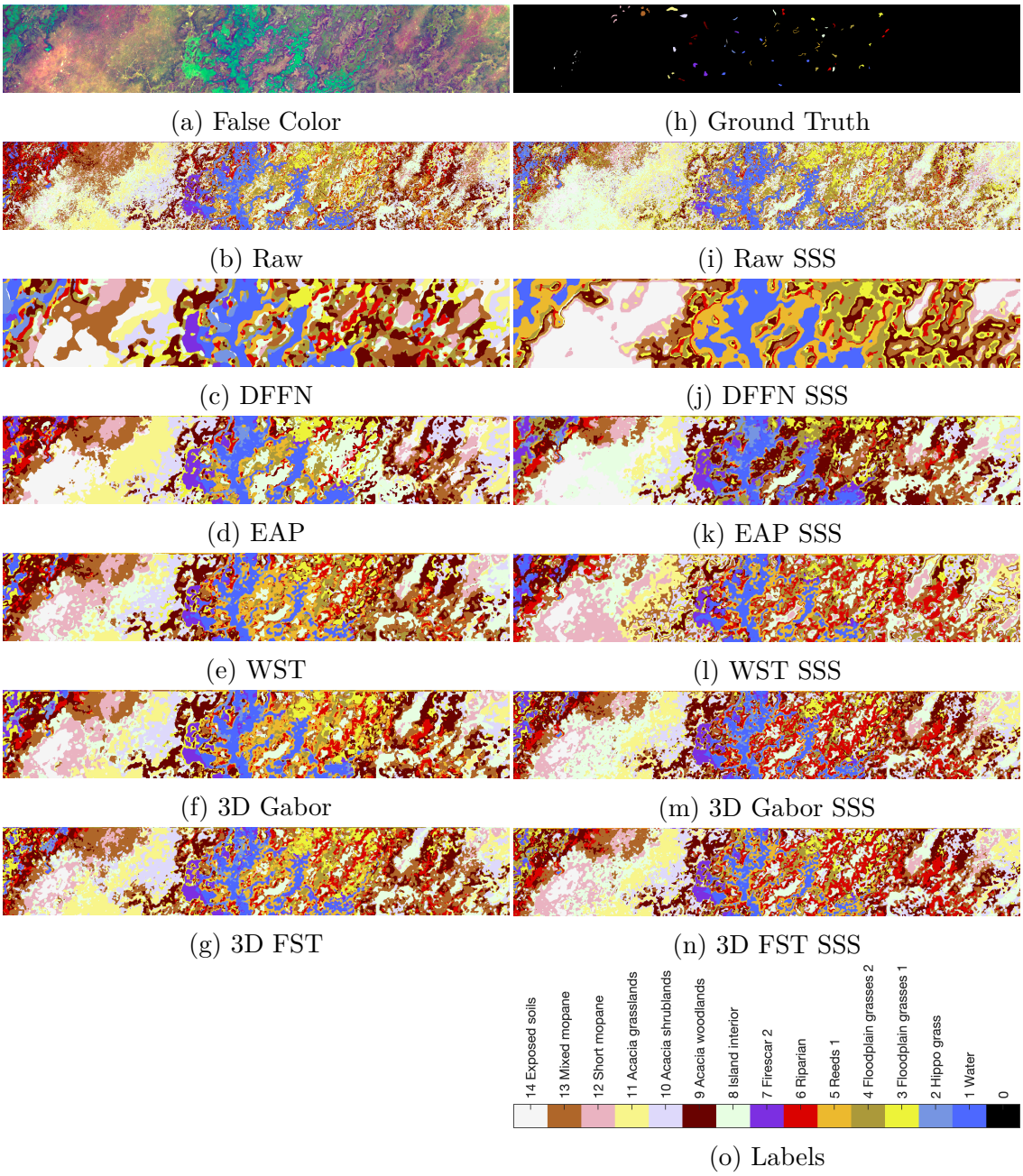
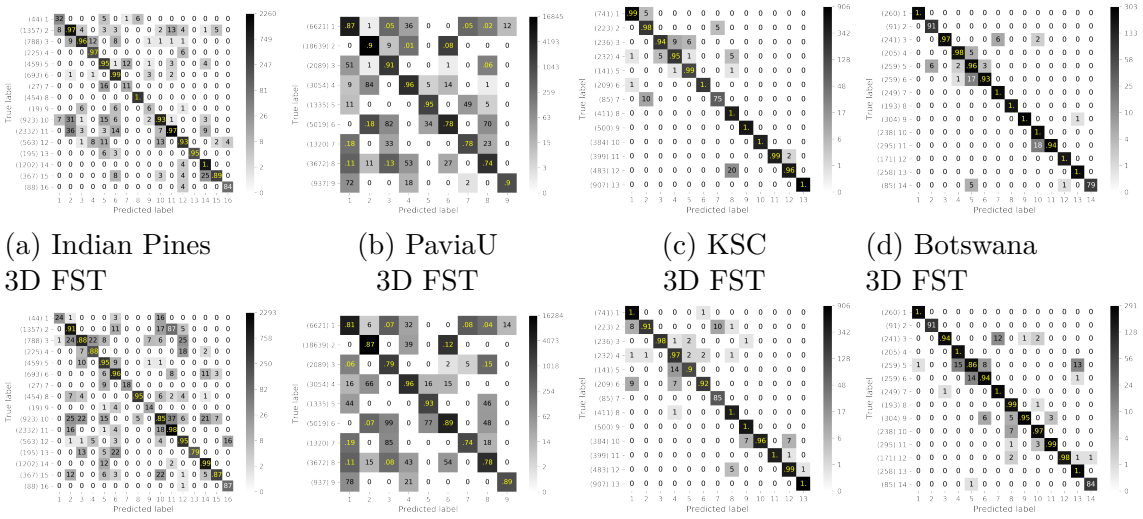


Figure 3.13: Full classification on Botswana with 20 samples per class. This dataset has labels for only 0.86% of all pixels and is the hardest to interpret of all our datasets, but both these attributes have the potential to reveal spatial bias in classification methods. We see this is the case for DFFN and EAP, where compared to the false color image, delicate features near the central water like islands is blurred away.



(a) Indian Pines 3D FST

(b) PaviaU 3D FST

(c) KSC 3D FST

(d) Botswana 3D FST

(e) Indian Pines DFFN

(f) PaviaU WST

(g) KSC WST

(h) Botswana Gabor

Figure 3.14: Confusion matrices. In parentheses on the y-axis is the total number of samples per class in the test set. The number in the (i, j)-th cell is the number of pixels predicted to be class j when the true label is class i, when this number is greater than 99 it is instead expressed as a percentage of all the pixels in that class in the test set and is colored yellow. The colorbar is on a log scale. For Indian Pines 5% of training data was used, for PaviaU 10 samples, for KSC 20 samples, and for Botswana 10 samples. All confusion matrices are made from the best performing trial of the two best methods at the stated amount of randomly distributed training data.

Chapter 4: Cortical Features for Defense Against Adversarial Audio Attacks

As voice assistant systems like Amazon Alexa, Google Assistant, Apple Siri and Microsoft Cortana become more ubiquitous and integrated into modern life, so does the risk of antagonists taking control of devices that we depend on. Adversarial Attacks on Audio are one way that a voice assistant could be subverted to send an unwanted message, or bank transfer, or unlock a home. All the mentioned voice assistants rely on wake-word detection to initiate their automatic speech recognition (ASR) systems that give control over their functions. The wake word detection system is both a voice assistant's first line of defense against adversarial attacks and its least complex feature. An attacker's first step to manipulate a voice assistant would be to issue an human-imperceptible wake-word, or conversely to prevent a user's wake-words from being registered. The latter is called a denial-of-service, and can have safety critical applications, for example when a user needs to call for help but cannot physically reach a device.

A number of recent works have achieved adversarial examples for automatic speech recognition. The Carlini-Wagner attack [89] can produce an adversarial waveform that is 99.9% similar to the original, but produces any other intended

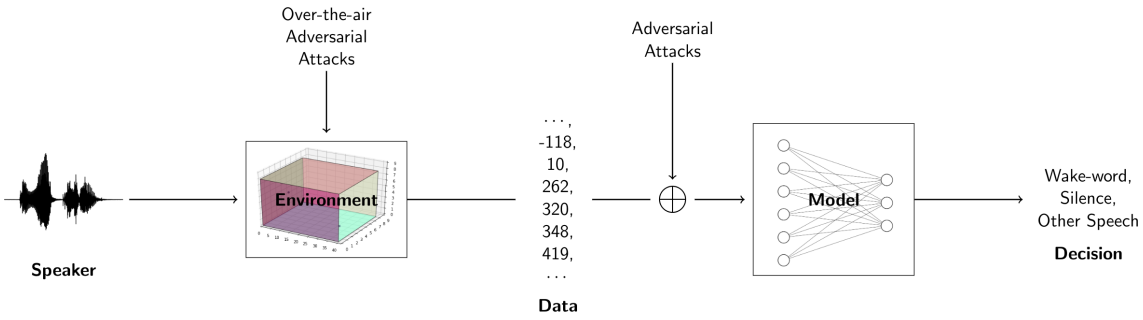


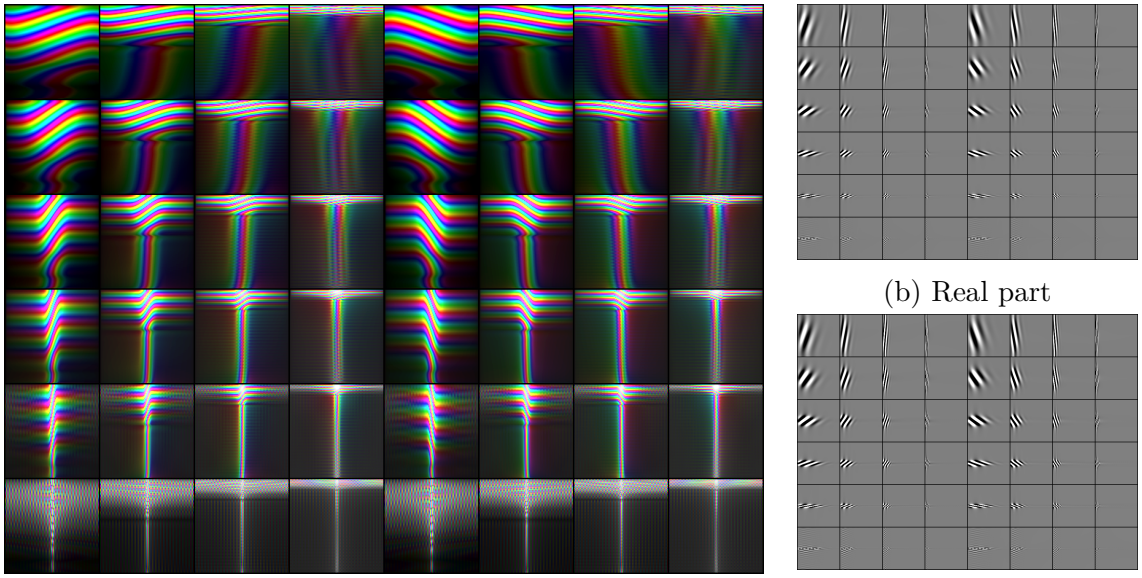
Figure 4.1: A schematic showing the flow of information for a voice assistant wake-word detector and two attack vectors. The speaker generates audio for their intended command. The environment is an acoustic environment containing the speaker, other sources of audio like background and other speakers, distortions like reverberations, and the voice assistant's microphone and audio hardware including the ADC. The data (assumed to be 16 bit integer and 16kHz) is then available to a model which ultimately makes the decision whether or not to "wake up" its general ASR capability. Image Credits: the FC nodes [5].

sentence by the ASR. Adversarial examples have since developed and improved in several ways. An important attribute for practical adversarial audio is that it be causal. Often an adversarial example is a function of the complete original audio that it is meant to modify, so thus original audio from hundreds of milliseconds in the future must be accessed to play an adversarial sound in the present. One solution is *universal adversarial examples* [90, 91], examples that can behave adversarially regardless of what original audio they are superimposed on. Universal adversarial audio examples have been demonstrated with as little as one training sample for ASR [92]. Another important attribute for practical adversarial audio is that it be robust, that is the adversarial example remains effective despite some level of distortions. This goal is not completely orthogonal to universal adversarial examples, since examples cannot be universal without also being invariant to translation in time, and when superimposed with some other audio. The ultimate test of robustness is over-the-air attacks, where as shown in Fig. 4.1 the adversarial

example is introduced to the environment generating the data for the model rather than to the data directly. Robust adversarial examples have been demonstrated to exist for simulated environment ASR systems [93], and more recently robust and universal examples have been generated for ASR systems [94], and even commercial wake word systems [95].

Across many languages speech has a pattern of spectrotemporal modulations that differ from most environmental noise and distortions [96], and the brain has developed a hierarchical neural system that uniquely responds to these signals [6]. These specialized neuronal structures are in the auditory cortex, and for example are largely unresponsive to pure tones [6]. It has been demonstrated in animals that the auditory cortex encodes vocalization features while suppressing or being invariant to the noise features [97, 98]. Invasive experiments in humans also have shown that auditory cortex neurons selectively suppress of acoustic features of noise in their neural responses [99].

This characteristic of speech-noise separability found the auditory cortex is exactly the same characteristic we would like in artificial intelligence voice assistants. For example, if a wake-word system had an intermediate representation that was tuned to speech the same way our brains are, then it would be necessary to perceptively modify speech to change that representation and ultimately the output of the wake-word model, making imperceptible adversarial examples impossible. Indeed the aforementioned biological findings have inspired a computational model of the mammalian auditory cortex to create such a representation [100, 101]. Our aim in this chapter is to incorporate the advantages of this cortical representation into a



(a) Complex Colorized

(b) Real part

(c) Imaginary part

Figure 4.2: The 48 cortical filters used in the cortical network. Along the rows are the six scale parameters $\psi \in \{.25, .5, 1, 2, 4, 8\}$, the left four columns are the rate parameters $\omega \in \{4, 8, 16, 32\}$ with phase $\phi = +1$ and the four right columns have phase $\phi = -1$ and the same scale $\times$ rate parameters. Colorization with hue $= (\angle z + \pi)/(2\pi) + 1/2$ and lightness $= 1 - 1/(1 + |z|^{0.3})$.

voice assistant and show the advantages of the cortical representation for defending against adversarial audio attacks.

4.1 Cortical Transform

The details of human hearing have been studied for centuries. In this section we highlight some of the consensus of the last few decades of modeling for the first stages of how the brain makes sense of speech. In the early stage, acoustic signals are transformed into 2 dimensional spectrograms in the cochlea, which acts as a filter bank, and filtered by the hair cell and lateral inhibitory network [6, 102]. These lower auditory areas are tonotopically organized: each cochlear location is

associated with a best frequency it responds to, with time as the other axis this representation can be interpreted as an auditory image [6]. As signals move out to the auditory cortex (A1), what was essentially 1 dimensional processing in the cochlea becomes 2 dimensional in A1 [101]. Invasive experiments have measured the responses of ferrets' A1 neurons and characterized them with corresponding spectro-temporal receptive fields (STRFs) [103]. An STRF of a neuron indicates the frequencies and time lags that correlate with an increased response of that neuron; these two dimensions are often referred to as scale (spectral) and rate (temporal).

A computational model for these receptive fields can be estimated from the measurements of these neurons responses [101, 104]. On the scale axis the transfer function of these neurons are well approximated by the second derivative of a Gaussian function $h(y) = \Omega(1 - 2(\Omega y)^2)\exp\left(-(\Omega y)^2\right)$ [101]. Neurons are tuned to be responsive to a variety of rates and scales [103], for scale features we parameterize the approximation of the neural response as:

$$R_{2,\psi}(y) = (\Omega_2 y/\psi)^2)\exp\left(1 - (\Omega_2 y/\psi)^2\right),\ y = 1, \ldots, L, \psi \in \Psi \tag{4.1}$$

where $R_{2,\psi}(y) = \mathcal{F}\{r_{2,\psi}(f)\}$ [101]. On the rate axis, the response is modeled by a function with a central excitatory band surrounded by inhibitory side bands, this is approximated with:

$$r_{1,\omega}(t) = (t/\Omega_1\omega)^2\omega\exp\left(-\alpha t/\Omega\omega\right)\sin(2\pi t\ \Omega_1\omega),\ t = 1, \ldots, L, \omega \in \mathcal{W} \tag{4.2}$$

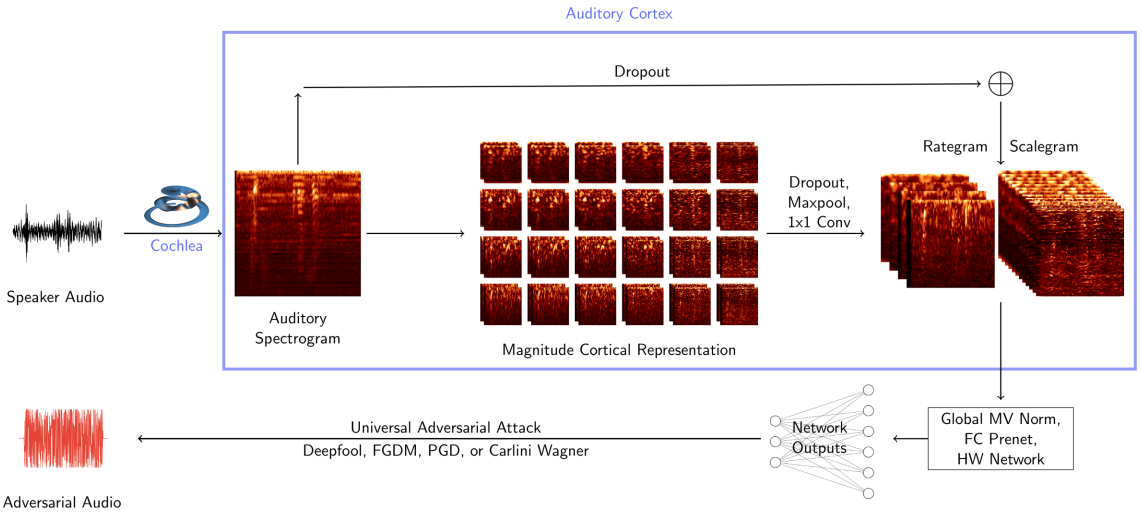


Figure 4.3: A schematic showing the proposed cortical network, meant to emulate the first two stages of audio processing in the brain (labeled in light blue). Integrating this features extraction directly into the network gives us two advantages over prepossessing: 1) The feature representation can be enhanced with learning (dropout with 1x1 convolutions) 2) We can backpropagate through these layers are find adversarial audio directly without any lossy transformations (such as Griffin-Lim). Image Credits: the cochlea [6], the FC nodes [5].

where $\alpha = 3.5$ was chosen empirically, as detailed in [101]. Finally the 2D STRF filters are

$$f(\omega, \psi, \phi) = \begin{cases} r_{2,\psi} \otimes r_{1,\omega} \ , & \phi = 1 \\ r_{2,\psi} \otimes \overline{r_{1,\omega}} \ , & \phi = -1 \end{cases} , \ \omega \in \mathcal{W}, \psi \in \Psi \tag{4.3}$$

When phase is -1 the rate filter is conjugated. These filters with $\Psi = \{.25, .5, 1, 2, 4, 8\}$ (cyc/oct) and $\mathcal{W} = \{4, 8, 16, 32\}$ (Hz) are displayed in Fig. 4.2.

Taking these biophysical observations, we can build an audio processing pipeline that transforms sound into a two dimensional auditory spectrogram, and then applies the STRFs in Eq. (4.3) to form a higher dimensional *cortical representation* from the magnitude response. This abstraction captures important physiological observations: selectivity to combined spectral and temporal features, and with proper

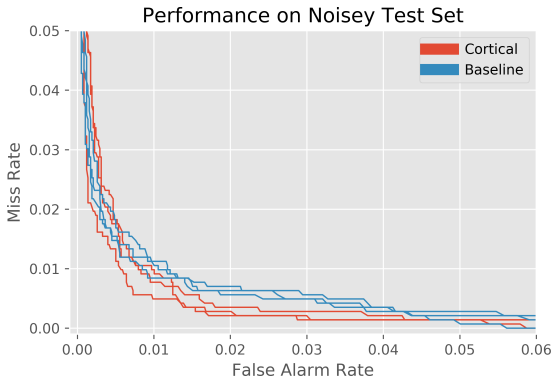


Figure 4.4: False alarm and Miss rates of the networks that we compare. One line is plotted for each of three training trials for each of the two architectures.

choice of $\mathcal{W}$ the temporal dynamics of phase locking which decreases to less than 30 Hz in the cortex [100] (something only possible with large filters). This pipeline can be seen in Fig. 4.3. The cortical transform of audio is a high dimensional tensor with dimensions (time $\times$ frequency $\times$ rate $\times$ scale). This can be reduced to two 3D tensors by reducing/max pooling across the rate (*scalegram*) or scale (*rategram*) [102].

Such a pre-processing feature extraction pipeline has previously been applied successfully to speaker detection [105] and phoneme classification [102, 106]. The cortical representation captures the distinctive dynamics of phonemes, which are easily visually discernible in rategrams and scalegrams [102]. The magnitude cortical response carries enough information to reconstruct the spectrogram that generated it [100]. Because of its ability to capture speech specific features, inverting the magnitude cortical response has been applied to speech enhancement since it has high fidelity in the reconstruction of speech and low fidelity in the reconstruction of background noise [107].

4.2 Proposed Architecture

Since we study defending against adversarial attacks on the wake-word systems of voice assistants, we choose a baseline network from the Amazon Alexa team. A variety of networks have been published by team-members at Amazon [108, 109, 110] and cybersecurity practitioners [111, 112], and the time-delayed bottleneck highway network (TDB-HW) is the latest in the literature [108]. It consists of a feature extractor stage and a classifier stage. Before the feature extraction stage, an auditory spectrogram (a log-mel filter bank spectrogram, abbreviated LFBE [110]) features are fed into a two layer FC prenet with dropout. The feature extraction stage of TDB-HW consists of four highway blocks. Then a bottleneck with 20 left (200ms) and 10 right (100ms) contexts is applied. Finally six highway blocks and one FC layer is applied to provide a classification. This is the network we refer to as "baseline".

For our Cortical network, the input LFBE features remain the same, and are convolved with 48 size 32x32 complex STRF filters from Eq. (4.3). These 48 filters are shown in Fig. 4.2, and have parameters which mimic the response measured in auditory cortex neurons [101]. After applying an absolute value and dropout, max-pooling is applied along the rate and scale axes of the features to yield a scalegram and a rategram. These are concatenated, a 1x1 convolution is applied, and the original spectrogram with substantial dropout (0.9) is added to this 3D tensor. A small prenet is applied before the TDB-HW network is applied. The residual addition is motivated by the loss of 2D high frequency features by the STRFs, and

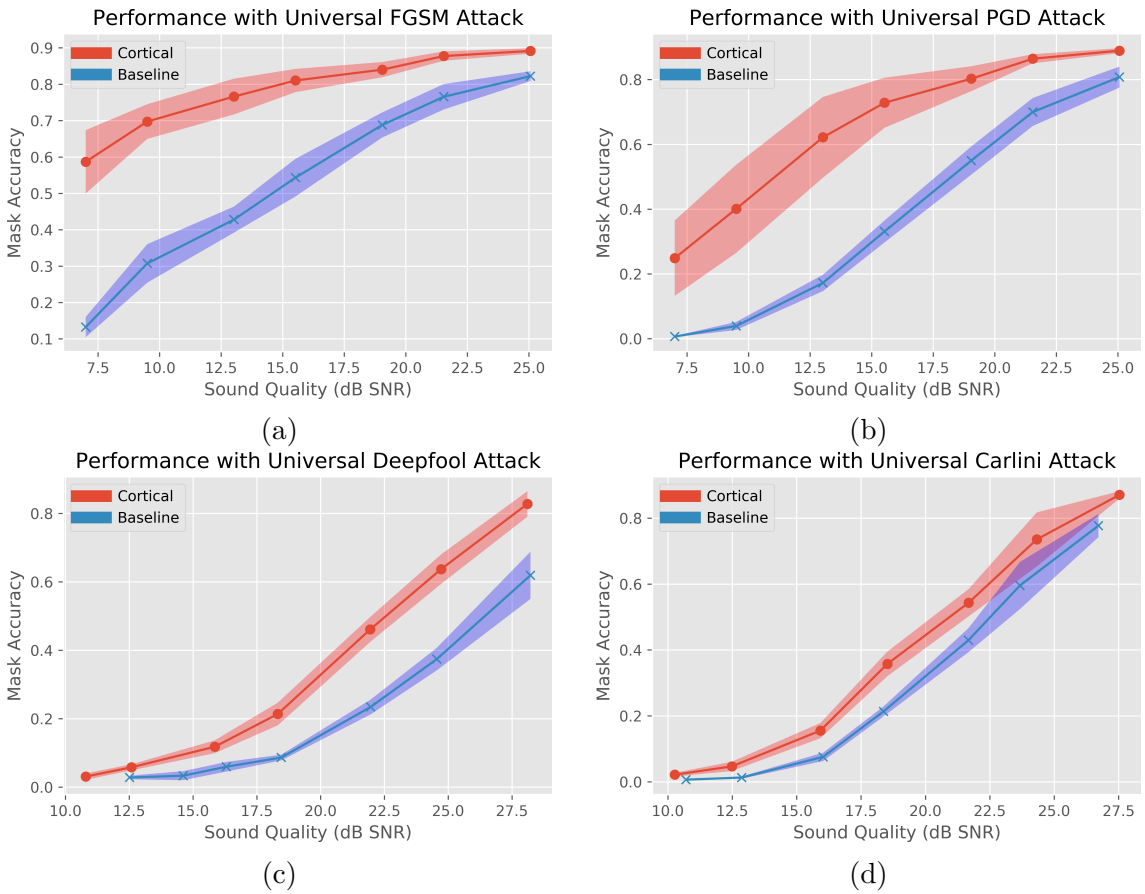


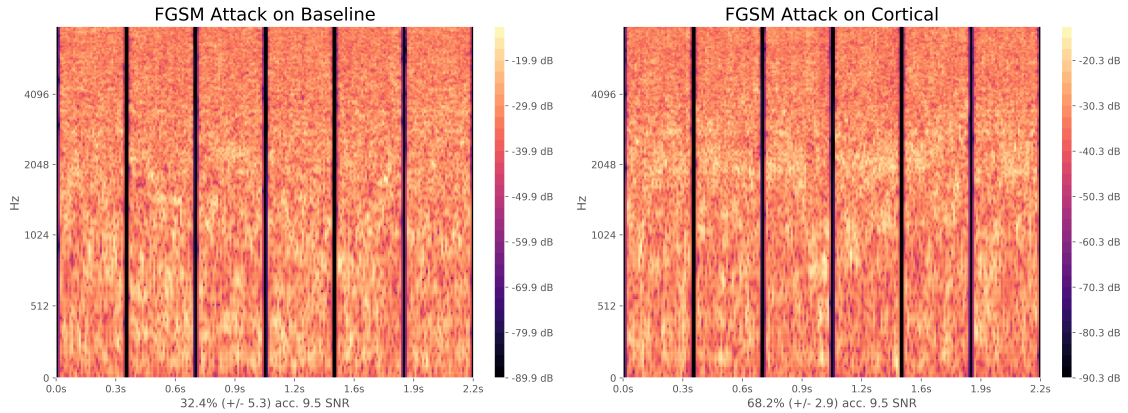
Figure 4.5: We trained three baseline and three cortical networks, and attacked each network four times with various SOTA attacks from IBM's ART toolbox [7]. Each point in each figure corresponds to the hyperparameters for a single type of attack constrained to an ℓ_∞ ball with radius $\varepsilon \in \{0.0025, 0.00375, 0.005, 0.0075, 0.01, 0.015, 0.02\}$ (the x-axis is converted to the SNR of the achieved attack, larger ε leads to smaller SNR). Each point is the mean of 12 trials, that is the mean validation mask accuracy for the best (minimum accuracy) adversarial noise found during the attack trial. The shaded region shows 1 standard deviation across the 12 trials.

a way to recover some of that information. The STRFs are not learned during training, so the total learnable parameters are similar between the two models.

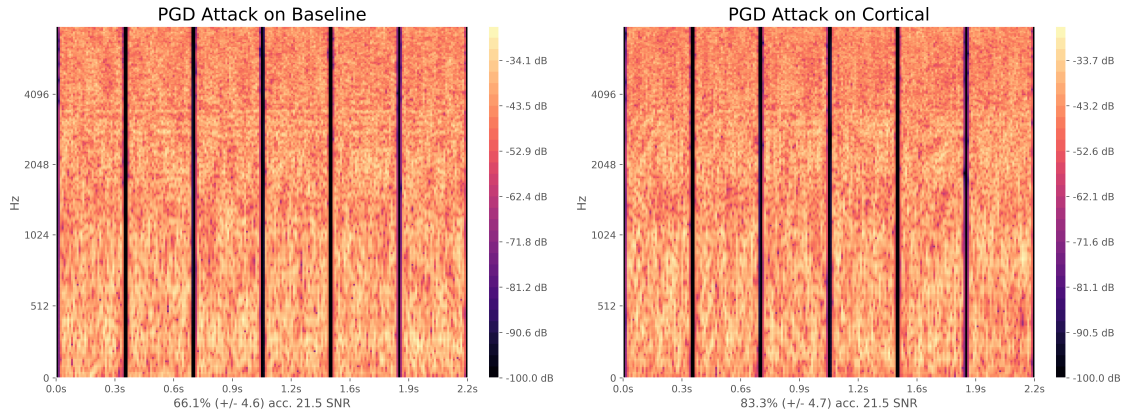
4.3 Experimental Results and Discussion

4.3.1 Data Sets

We created several monaural 15 hour training dataset and a 3 hour validation set to train our networks. We create our training and validation splits from different speakers of wakewords from the Kaggle Alexa dataset [113] and speakers in the Librispeech dataset [114]. For additional background noise we use the DEMAND dataset kitchen, living room, laundry room, hallway, office, cafeteria, cafe environments. We developed an algorithm to extract separate words from speech, which we use to isolate single words from the Librispeech dataset for negative example words. We use this algorithm and fine tune it manually when necessary on the Alexa dataset to extract start/end times positive words (but keep surrounding audio including noise), and we use it unsupervised on Librispeech to extract negative words. Positive words are augmented randomly to have duration between 400ms and 900ms (drawn from a distribution fitted on the training speakers), and to have a base frequency drawn from $\mathcal{U}([80, 350] \cap [f_{\text{source}} - 60, f_{\text{source}} + 60])$ Hz using the high quality C++ sound library Rubberband [115]. Negative words have modified length multiplied by $\mathcal{U}([0.8, 1.2])$, and modified pitch multiplied by $\mathcal{U}([0.8, 1.4])$ using the same library. To form a single training or validation sound clip, words are randomly selected to be positive or negative, and arranged with silence ($\mathcal{U}(50, 150)$ ms) to form a multi channel clip. Background noise form DEMAND is added as an additional channel. To form the single channel audio the channels are normalized to -15



(a) $\varepsilon = 0.015$ (b) $\varepsilon = 0.015$



(c) $\varepsilon = 0.00375$ (d) $\varepsilon = 0.00375$

Figure 4.6: This figure shows qualitative differences between the adversarial attacks on baseline and cortical networks for the FGSM and PGD attacks in the mel spectrogram domain. Each subfigure concatenates the best noise from 6 trials. We see a slightly higher dependence of modifying speaker frequencies in the baseline network over the cortical network. This is consistent with our hypothesis that the cortical representation of speech is more invariant to noise, at least for speaker frequencies (below 1kHz). In the PGD attacks we see more horizontal lines in the baseline, and some diagonal lines in the cortical. This is consistent with our expectation that cortical features filter out stationary noises.

dBFS and simulated in a bedroom or living room with absorption $\mathcal{U}([0.4, 0.9])$ with random distances from the simulated microphone [116]. Finally the single channel is normalized -15 dBFS and quantized to 16-bits.

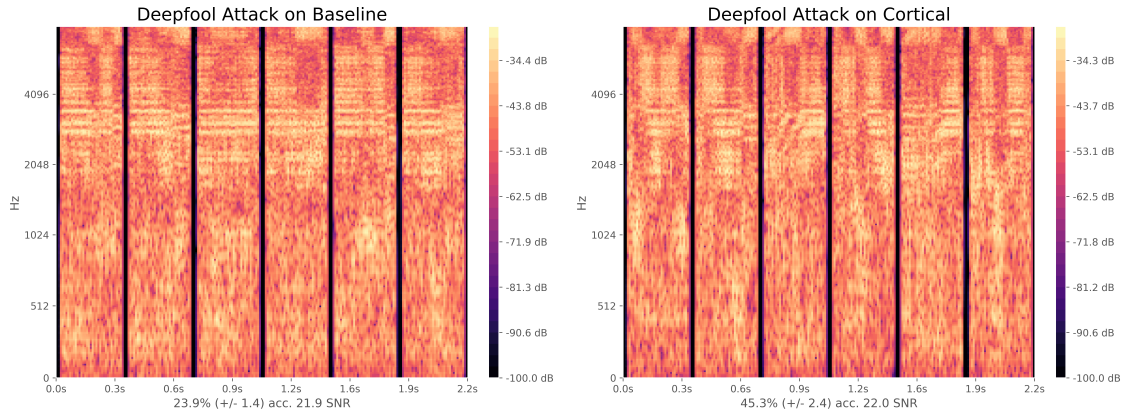
The word extraction algorithm. To extract words from audio in an unsupervised

method on Librispeech we used the following algorithm. Original audio was filtered for speech base frequencies $85 - 255$ Hz and normalized, then the energy of this signal was convolved with a 100ms Hann window at 25ms hop, giving an envelope for the signal's energy . Local min/max extrema were found, and segments were marked around triplets of low-high-low extrema points. Neighboring segments were joined if less than 350ms, and segments longer than 1.2s were discarded. Then words segments were sorted by the quietest silences preceding and following the word, and were extracted from the original unmodified audio.

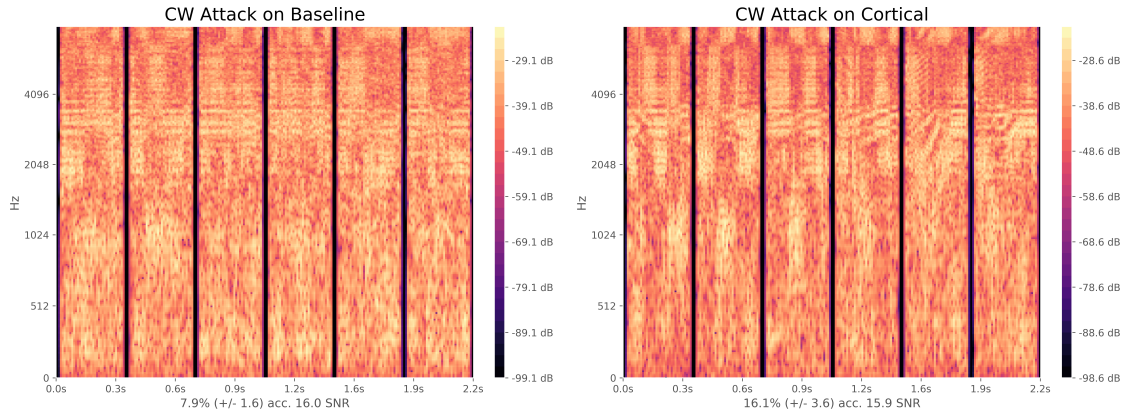
4.3.2 Analysis

The baseline and cortical networks were both able to train to similar competitive rates as seen in the Detection Error Tradeoff (DET) curves in Fig. 4.4. These curves are specific to the dataset that we created, but for comparison, the company PicoVoice has created a dataset in a similar manner to ours and tested the CMU Sphinx Open Source Toolkit PocketSphinx, the open source KITT.AI, and their own commercial product PicoVoice [117]. They advertise a miss rate of 6% at 1 false alarm per 10 hours for their own product, 32% for KITT.AI, and 48% for PocketSphinx. On our dataset our miss rate is 7% at 1 false alarm per 10 hours for our baseline and 10% for the cortical network. However this is an extreme mode of operation, at 1 false alarm per 1 hour is 5% for both models.

We evaluate the merit of the cortical network as a defense to adversarial examples on Universal FGSM, PGD, Deepfool, and Carlini-Wagner attacks [90, 91, 118,



(a) $\varepsilon = 0.005$ (b) $\varepsilon = 0.005$



(c) $\varepsilon = 0.01$ (d) $\varepsilon = 0.01$

Figure 4.7: This figure shows qualitative differences between the adversarial attacks on baseline and cortical networks for the Deepfool and Carlini Wagner attacks in the mel spectrogram domain. Each subfigure concatenates the best noise from 6 trials, and reports the SNR and accuracy on these 6 noises on the x-axis. The cortical network attacks appear to pulsate more in time while the baseline network attacks are more stationary. The cortical attacks also exhibit more diagonal lines at resonant frequencies.

119, 120, 121]. We use the standard implementation of these in the IBM Adversarial Robustness Toolbox (ART) [7]. The FGSM, Deepfool, and Carlini-Wagner attacks were performed for 4000 iterations and evaluated every 400, PGD was performed for 250 examples for 100 iterations each, evaluated every 25 examples. For each trial, the minimum accuracy noise was recorded on an eval set, and the average minimum accuracy of 12 trials over the three networks is show in Fig. 4.5.

The FGSM and PGD attacks have the slowest accuracy decay rate as the strength of the noise grows on the cortical network. This is likely due to the more dispersed patterns of attacks that these two methods create, shown in Fig. 4.6. Because of the STRFs selectivity for changing phoneme dynamics [102, 122], the cortical network should be more invariant to stationary noises as the human auditory cortex [99], and is only slowly impacted by this type of noise. We see the noises generated by the Deepfool and Carlini-Wagner attacks are much more textured in Fig. 4.7. For the same search in a given ℓ_∞ size ball, the cortical network requires a more elaborate noise than the baseline network off of the auditory spectrogram (and still performs better at the same SNR). The noise for the baseline is more stationary and has a higher impact on speaking frequencies. However, all the adversarial noises cannot be said to be imperceptible to the human ear, though speech is still clearly heard and understood by a human, the fricative like quality of the adversarial noises is perceptible even at 25dB SNR.

4.4 Conclusion & Further Work

We apply several white-box iterative optimization-based adversarial attacks to an implementation of Amazon Alexa's HW network, and a modified version of this network with integrated cortical representation prepossessing, and show that the cortical features help defend against universal adversarial examples. At the same level of distortion, the adversarial noises found for the cortical network are always less effective for universal attacks. Further work could start by refining the audi-

tory preprocessing, which here was limited to LFBE features emulating the cochlear filterbank. A more faithful auditory spectrogram would simulate the transduction of the hair cells and the reduction of the lateral inhibitory network (band specific non-linearities and additional filtering). We also constrained our analysis to highly effective but not imperceptible adversarial noise. Further work could more subtle noises that degrade performance gradually, enforcing imperceptibility with perceptual masking [93, 123].

Chapter 5: cGANs with Multi-Hinge Loss

Generative Adversarial Networks (GANs) [124] are an attractive approach to constructing generative models that mimic a target distribution, and have shown to be capable of learning to generate high-quality and diverse images directly from data [125]. Conditional GANs (cGANs) [126] are a type of GAN that use conditional information such as class labels to guide the training of the discriminator and the generator. Most frameworks of cGANs either augment a GAN by injecting (embedded) class information into the architecture of the real/fake discriminator [8], or by adding an auxiliary loss that is class based [9].

We describe an algorithm that uses both a projection discriminator and an auxiliary classifier with a loss that ensures generator updates are always class specific. Rather than training with a function that measures the information theoretic distance between the generative distribution and one target distribution, we generalize the successful hinge-loss [127] that has become an essential ingredient of state of the art GANs [125, 128] to the multi-class setting and use it to train a single generator-classifier pair [128]. While the canonical hinge loss made generator updates according to a class agnostic margin learned by a real/fake discriminator [127], our multi-class hinge-loss GAN updates the generator according to many

Figure 5.1: The proposed MHGAN generates images while leveraging the information from the margins of a classifier. All the images in this figure are conditionally sampled from our MHGAN.

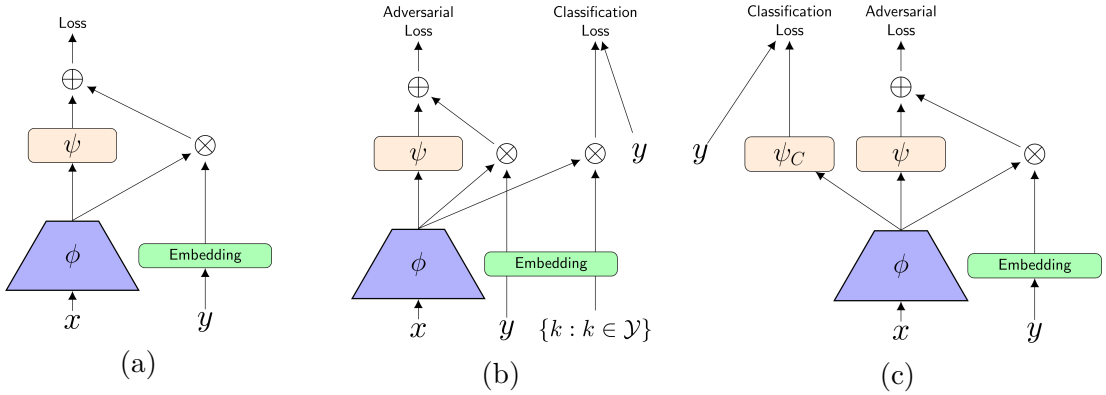


Figure 5.2: The architectures of the fully supervised GANs that we train. 5.2a shows the projection discriminator architecture [8] of our SAGAN. 5.2b shows how the projection discriminator could simultaneously be trained with an auxiliary classifier loss: we train MHSharedGAN this way. 5.2c Is an ACGAN [9] architecture that also contains a projection discriminator: we train our ACGAN and MHGAN with this architecture.

classification margins. With this modification, we are able to accelerate training compared to other GANs with auxiliary classifiers by performing only 1 D-step per G-step, and we improve Inception and Frechet Inception Distance Scores on Imagenet at 128×128 on a SAGAN baseline. We make our tensorflow code available at `https://github.com/ilyakava/gan`.

5.1 Background

A GAN [124] is a framework to train a generative model that maps random vectors $z \in \mathcal{Z}$ into data example space $x \in \mathcal{X}$ concurrently with a discriminative network that evaluates its success by judging examples from the dataset and generator as real or fake. The GAN was originally formulated as the minimax game:

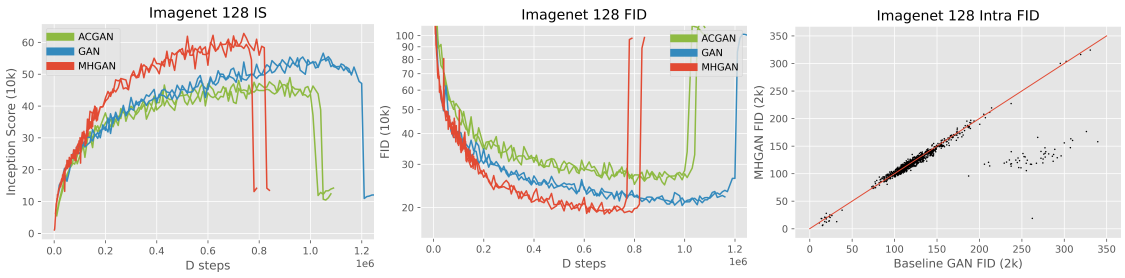
$$\begin{aligned} &\max_D E_{x\sim p_\mathrm{d}}[\log(D(x))] + E_{z\sim p_z}[\log(1 - D(G(z)))], \\ &\min_G E_{z\sim p_z}[\log(1 - D(G(z)))], \end{aligned} \tag{5.1}$$

where p_d is the real data distribution consisting of examples $x \in \mathcal{X}$, p_z is the latent distribution over the latent space $\mathcal{Z}$, $G : \mathcal{Z} \to \mathcal{X}$ is the generator neural network, and $D : \mathcal{X} \to [0, 1]$ is the discriminator neural network. The GAN model transfers the success of the deep discriminative model D to the generative model G and succeeds in generating impressive samples $G(z)$.

5.1.1 Loss functions for training GANs

In this chapter we will use the word "loss" (and symbol L) for a quantity that we minimize when optimizing. For clarity we rewrite the GAN optimization problem in a more generic form [129] that is amenable to discussion and programming:

$$\begin{aligned} &\min_D E_{x\sim p_\mathrm{d}}[f(D(x))] + E_{z\sim p_z}[g(D(G(z)))], \\ &\min_G E_{z\sim p_z}[h(D(G(z)))]. \end{aligned} \tag{5.2}$$



(a) (b) (c)

Figure 5.3: Evaluation metrics for Imagenet-128. Inception Score and Frechet Inception Distance during training, and a comparison of Intra-class Frechet Inception distance for our best MHGAN model and the best baseline SAGAN we trained. In Figure 5.3c there are 622 classes whose intra-fid improve (each point below the red line is a class for which FID improves) and there are 46 classes in the lower right cluster.

Note the two minimums. We also don't restrict D to the unit interval but instead $D : \mathcal{X} \to \mathbb{R}$.

To regain the minimax objective in Eq. (5.1) we set $f(w) = \log(1 + e^{-w})$ and $h(w) = -g(w) = -w - \log(1+e^{-w})$ [129]. This choice minimizes the Jensen-Shannon divergence between p_{d} and p_{g} [124], which denotes the model distribution implicitly defined by $G(z), z \sim p_z$. The minimax objective however was difficult to train [130], and the study of various ways to measure the divergence or distance between p_{d} and p_{g} has been a source of improved loss functions that make training more stable and samples $G(z)$ of higher quality.

WGAN set $f(w) = -w$, $h(w) = -g(w) = -w$ and clipped the weights of D to greatly improved the ease and quality of training and reduced the mode dropping problem of GANs, and had the interpretation of minimizing the Wasserstein-1 distance between p_{d} and p_{g}. Minimizing Wassertstein-1 distance was shown to be a special instance of minimizing the integral probability metric (IPM) between p_{d} and p_{g}

[131], and inspired a mean and covariance feature matching IPM loss (McGAN)[131] following the empirical successes of the Maximum Mean Discrepancy objective [132] and feature matching [128].

The mean feature matching of McGAN has a geometric interpretation as well: the gradient updates of feature matching for generator updates are normal to the separatating hyperplane learned by the discriminator [127]. Using the SVM like hinge-loss choice of $f(w) = \max(0, 1 - w)$, $g(w) = \max(0, 1 + w)$ and $h(w) = -w$ [127, 133] the gradient are would be similar to those of McGAN. When combined with spectral normalization of weights in D [134], the hinge loss greatly improves performance, and has become a mainstay in recent state of the art GANs [8, 125, 135]. In this chapter we generalize this hinge loss to a multi-class setting.

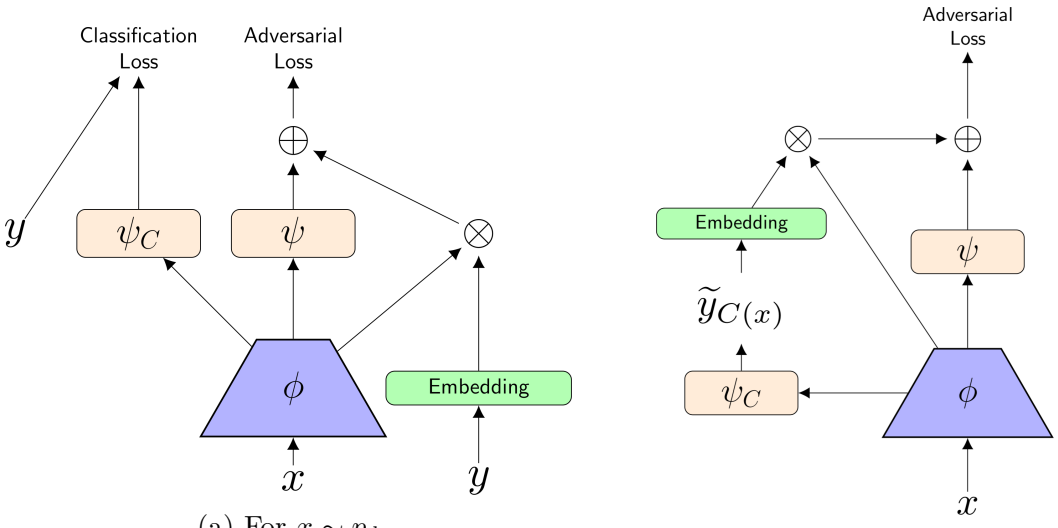
5.1.2 Supervised training for conditional GANs

Conditional GANs (cGANs) are a type of GAN that use conditional information [126] in the discriminator and generator. G and D become functions of the pairs $(z \sim p_z, y \sim p_{\mathrm{d}})$ and $(x, y) \sim p_{\mathrm{d}}$, where y is the conditional data, for example the class labels of an image. In a cGAN with a hinge loss, the discriminator would minimize L_D in Eq. (5.3), and the generator would minimize L_G in Eq. (5.3)

[125, 135].

$$
\begin{aligned}
L_D =& E_{(x,y)\sim p_{\mathrm{d}}}[\max(0, 1 - D(x, y))] \\
&+ E_{z\sim p_z, y\sim p_{\mathrm{d}}}[\max(0, 1 + D(G(z, y), y))] \\
=& L_{D\mathrm{real}} + L_{D\mathrm{fake}}, \\
L_G =& - E_{z\sim p_z, y\sim p_{\mathrm{d}}}[D(G(z, y), y)].
\end{aligned}
\tag{5.3}
$$

We briefly review some work on using conditional information to train the discriminator of GANs, as well as uses of classifiers.



(a) For $x \sim p_{\mathrm{d}}$

(b) For $x \sim p_{\mathrm{unlab}}$

Figure 5.4: The architectures of the semi supervised GANs that we train. 5.4a shows the auxiliary classifier architecture [8] of our SAGAN, the same as used for fully supervised experiments. When $x \sim p_{\mathrm{d}}$ the discriminator treats the example as shown in 5.4a. In the diagram the green "Embedding" block is the projection discriminator embedding for a class, ϕ is the discriminator up to the penultimate feature layer, ψ is a linear layer with scalar output, ψ_C is a linear layer with output size n classes. When $x \sim p_{\mathrm{unlab}}$ and the label is not available, it is treated as shown in 5.4b. This is similar to S^2GAN [10]. Both ACGAN-SSL and MHGAN-SSL use these architectures, the only difference between the two is the form of the classification loss for labeled examples.

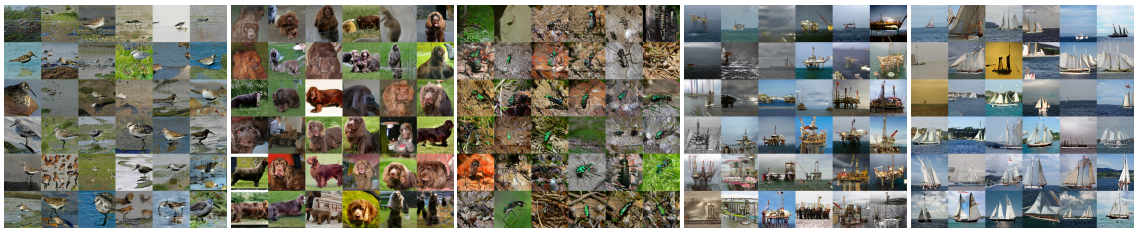
A projection discriminator [8] is a type of conditional discriminator that adds

the inner product between an intermediate feature and a class embedding to its final output, and proves highly effective when combined with spectral normalization in G [8, 125, 135]. Using multiple discriminator networks [136, 137] or multiple sub-networks of a single discriminator [138] has also been explored.

Several GANs have used a classifier in addition to, or in place of, a discriminator to improve training. CatGAN [139] replaces the discriminator with a K-class classifier trained with cross entropy loss that the generator tries to confuse. ALI [140] trains an encoder decoder pair with a discriminator, while also using the encoder for inference in a semi-supervised setting. ACGAN [9] uses an auxiliary classification network or extra classification layer appended to the discriminator, and adds the cross entropy loss from this network to the minimax GAN loss. Triple GAN [141] trains a classifier in addition to a discriminator and updates it with a special minimax type loss.

Improved GAN [128] originally proposed using a $K + 1$ classifier for semi-supervised learning (SSL) with feature matching loss, and others [142] have a used a similar setup for SSL GANs as well. The single conditional critic architecture of $D : (x, y) \to \mathbb{R}$ is swapped for the classifier architecture $C : x \to \mathbb{R}^{K+1}$, where there are K class labels and an extra label for fake images (the "+1") [128]. The Improved GAN trains this classifier architecture in a semi-supervised setting with log-likelihood loss, and trains the generator with a class agnostic mean feature matching loss. BadGAN [143] used the Improved GAN to achieve state of the art performance on semi-supervised learning classification and found the aim of having a low classification error on the K real classes is orthogonal to generating realistic ex-

(a) MHGAN IFID 134.82 (b) MHGAN IFID 126.52 (c) MHGAN IFID 158.00 (d) MHGAN IFID 147.14 (e) MHGAN IFID 132.38



(f) SAGAN IFID 287.50 (g) SAGAN IFID 275.03 (h) SAGAN IFID 339.48 (i) SAGAN IFID 307.16 (j) SAGAN IFID 296.28

Figure 5.5: Classes for which Intra-FID gets better for our proposed MHGAN. We show random samples from the 5 classes that improved the most in Figure 5.3c by points, excluding the trout class where SAGAN completely collapses.

amples. MarginGAN [144] similarly proposed a Triple GAN to train a high-quality Mean Teacher Shake-Shake classifier [145, 146] with a "bad" GAN by decreasing the margins of error of the cross entropy loss for generated images. Other approaches incorporate multiple adversarial games to get better SSL classification performance [147, 148]. There are few works which focus on improving the generator quality in a label limited setting. One approach has used two discriminators, one specializing on labeled and the other on unlabeled data [149]. Another approach investigated training an ACGAN and using pseudo labels from the classifier as inputs to the projection discriminator during training, but found that pre-labeling the unlabeled training data with a SSL classifier to be a better performing choice [10].

Our proposed multi-hinge GAN (MHGAN) uses a classifier like ACGAN [9] but instead of using a probabilistic cross entropy loss with the WGAN, we use a multi-

(a) MHGAN IFID 27.92 (b) MHGAN IFID 93.79 (c) MHGAN IFID 154.82 (d) MHGAN IFID 102.92 (e) MHGAN IFID 182.66

(f) SAGAN IFID 19.48 (g) SAGAN IFID 81.93 (h) SAGAN IFID 146.08 (i) SAGAN IFID 94.21 (j) SAGAN IFID 173.95

Figure 5.6: Classes for which Intra-FID gets worse for our proposed MHGAN. We show random samples from the 5 classes that worsened the most in Figure 5.3c by points. We did not observe any instances of mode collapse in MHGAN.

hinge loss similar to that in the discriminator. We demonstrate that adding this loss to the current state of the art SAGAN architecture with projection discrimination improves image quality and diversity, and trains stably at only 1 discriminator step per generator step for both supervised and semi-supervised settings. We compare our class specific modification with cross entropy loss [9] with projection discrimination, and with projection discrimination alone. Unlike cross-entropy, which has a basis in probabilistic quantities that are not present in WGANs, our multi-hinge loss is completely compatible with the WGAN formulation. We also find that the task of classification and discrimination should not share parameters, and show how to trade-off image diversity for quality in a shared parameter version MHSharedGAN.

5.2 Multi-hinge loss

We propose a multi-hinge loss that can be easily plugged into the popular and state of the art projection discriminator cGAN architecture of [8]. Our fully supervised formulation, motivated in Section 5.2.1, uses the auxiliary classifier setup seen in Figure 5.2c, but instead of using cross entropy as ACGAN [9] does to train this classifier we generalize the binary hinge loss [127, 133] to a multi-class hinge loss developed for SVMs [150], and we use it to train a spectrally normalized WGAN [134, 135, 151, 152]. Unlike other ACGANs, our multi-hinge loss formulation only requires 1 Critic step per 1 Generator step greatly speeding up training, and uses a single classifier for all classes in the dataset.

We denote the classifier function as $C : \mathcal{X} \to \mathcal{Y}$ and let $C_k(x)$ denote the kth element of the vector output of C for an example x, which represents the affinity of class k for x. The Crammer-Singer multi-hinge loss that we propose as an auxiliary term is:

$$
\begin{aligned}
L_{D\text{aux}} &= E_{(x,y)\sim p_{\text{d}}}[\max(0, 1 - C_y(x) + C_{\neg y}(x))] \\
L_{G\text{aux}} &= \\
&E_{\substack{z\sim p_z \\ y\sim p_{\text{d}}}}[\max(0, 1 - C_y(G(z, y)) + C_{\neg y}(G(z, y)))]
\end{aligned}
\tag{5.4}
$$

where $C_{\neg y}(x)$ is the classifier's highest affinity for any label that is "not y": $C_{\neg y}(x) =$

$\max_{k \neq y} C_k(x), y = 0, 1, \ldots, K$. We then train with the modified loss:

$$\begin{aligned} L_{\text{MH,D}} &= L_{D\text{real}} + L_{D\text{fake}} + L_{D\text{aux}}, \\ L_{\text{MH,G}} &= L_G + \lambda L_{G\text{aux}}. \end{aligned} \tag{5.5}$$

where $\lambda = 0.1$. The advantage of a conditional WGAN trained with the auxiliary terms in Eq. (5.4) is the main result of this chapter. In the following sections we discuss the motivation and advantage of training procedure. We also provide an SSL formulation in Section 5.2.2.

Method	**Imagenet-128**	
	IS	FID
Real data	156.40	
SAGAN	52.79	16.39
ACGAN	48.94	24.72
MHGAN	**61.98**	**13.27**
SAGAN 1M [135]	52.52	18.65
BigGAN 1M [125]	63.03	14.88

Table 5.1: Inception Scores and FIDs for supervised image generation on Imagenet-128. The models were chosen by maximizing the IS within 1M iterations. The BigGAN comparison we include is the one most similar to our setting (batch size 1024). Our SAGAN Baseline results are consistent with the results reported online with the implementation we use [1].

5.2.1 Motivation behind the multi-hinge loss

A class conditional discriminator should obviously not output "real" when it is conditioned on the wrong class. That is for a pair $(x, y) \sim p_{\text{d}}$ we expect if our discriminator loss is minimized then so is the quantity: $1 - D(x, y) + D(x, k), k \neq y$ (recall that in WGANs D is not a probability measure and may take values outside

$[0, 1]$). This quantity is positive for all k so long as the output of the discriminator conditioned on the correct label is larger by at least one than the discriminator conditioned on the rest of the labels. To explicitly enforce this margin, we could minimize the expectation:

$$E_{(x,y)\sim p_{\mathrm{d}}}[\max(0, 1 - D(x, y) + \max_{k \neq y} D(x, k))] \tag{5.6}$$

This form of a hinge-loss has been used by [150] in their formulation of efficient multi-class kernel SVMs. The ReLU $\max(0, \cdot)$ leads Eq. (5.6) to ignore cases where the correct decision is made with a margin more than 1. We design Eq. (5.4) to enforce the margins in Eq. (5.6).

Note that a projection discriminator cGAN implicitly has a classifier in it. The output of $D(x, k)$ projects the penultimate features onto an embedding for class k. Similarly the vector output of a typical classifier is a matrix multiply of the penultimate features with a n features $\times$ n class matrix. A projection discriminator $D(x, k)$ could be turned into a classifier $C(x)$ by using the entire matrix of class embeddings to output an affinity for every class. Creating a classifier this way doesn't increase the parameter count at all, and only increases computation in one layer of D by a constant factor (the number of classes) which we find is completely negligible for the large models typically trained for 64×64 images or greater. However, we find that this sharing of parameters between the classification task in Eq. (5.6) and the discrimination task around which the adversarial training centers is disadvantageous. As we will discuss, it is instead preferable to add an auxiliary classifier via an extra

final fully connected layer, the additional cost of a n features $\times$ n class matrix proves to be negligible in our experiments.

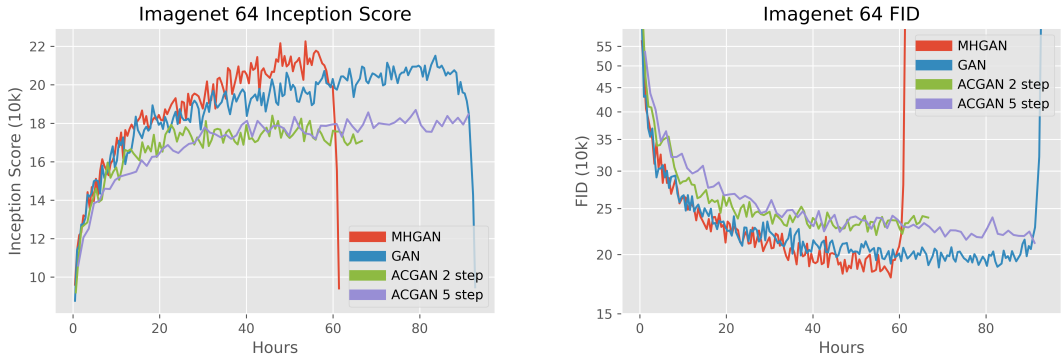
5.2.2 Semi-supervised learning

When additional unlabeled data $x \sim p_{\text{unlab}}$ is available, we find that learning with projection discrimination is not stable, that is bypassing the projection discriminator when a label is not available does not lead to successful training. So for semi-supervised settings we modify the training procedure in Eq. (5.5) by using pseudo-labels [10]. For the discriminator we add the term:

$$L_{D\text{unlab}} = E_{x \sim p_{\text{unlab}}}[\max(0, 1 - D(x, \widetilde{y}_C(x)))] \tag{5.7}$$

where $\widetilde{y}_C(x) = \arg\max_{k \in \mathcal{Y}} C_k(x)$, that is we depend on the classifier that we co-train with the discriminator. The loss for the generator is left unchanged. Overall for the semi-supervised setting we train with the losses:

$$\begin{aligned} L_{\text{MH,SSL,D}} =& \frac{L_{D\text{real}} + L_{D\text{unlab}}}{2} + L_{D\text{fake}} + L_{D\text{aux}}, \\ L_{\text{MH,SSL,G}} =& L_{\text{MH,G}}. \end{aligned} \tag{5.8}$$



(a) (b)

Figure 5.7: Inception Scores and FIDs (10k) for supervised image generation on Imagenet 64×64. 1 Million D-steps for SAGAN and MHGAN can be completed in about 48 hours.

A similar loss has previously been used with a WGAN co-trained with a cross entropy auxiliary classifier [10]

$$\begin{aligned} L_{\text{AC,SSL,D}} =& \frac{L_{D\text{real}} + L_{D\text{unlab}}}{2} + L_{D\text{fake}} \\ &+ E_{(x,y)\sim p_{\text{d}}}[\log C_y(x)], \\ L_{\text{AC,SSL,G}} =& L_G + \lambda E_{\substack{z\sim p_z \\ y\sim p_{\text{d}}}}[\log C_y(G(z,y))]. \end{aligned} \tag{5.9}$$

However we find that the consistency of hinge functions throughout the loss terms leads to more successful training.

Figure 5.4 shows how labeled and unlabeled data flow through our ACAN and MHGAN networks during semi-supervised learning. In Figure 5.4a is a projection discrimination network with an auxiliary classifier added. Figure 5.4b shows that for unlabeled data, the role of y is substituted with the pseudolabel $\widetilde{y}_{C(x)}$, and the classification loss is not trained.

5.3 Experiments

As our baseline, we use a spectrally normalized [134] SAGAN [135] architecture. We use this baseline from the publicly available tfgan implementation [1], and execute experiments on single v2-8 and v3-8 TPUs available on Google TFRC. This baseline was chosen for its exceptional performance [1]. On top of this baseline we implement a second baseline, ACGAN, and our MHGAN. The only architectural changes to SAGAN for both of these networks is that a single dense classification layer is added to the penultimate features. For these networks conditional information is given to G using class conditional BatchNorm [153, 154] and to D with projection discrimination [8]. A spectral norm is applied to both D and G during training [8, 135]. We train our SAGAN baseline with the hinge loss [127, 133] in Eq. (5.3). We train our proposed MHGAN with Eq. (5.5). To better see the advantage of our multi-hinge loss formulation we train an ACGAN baseline with the loss:

$$\begin{aligned} L_{\text{AC,D}} =& L_{D\text{real}} + L_{D\text{fake}} + E_{(x,y)\sim p_{\text{d}}}[\log C_y(x)], \\ L_{\text{AC,G}} =& L_G + \lambda E_{\substack{z\sim p_z \\ y\sim p_{\text{d}}}}[\log C_y(G(z,y))]. \end{aligned} \tag{5.10}$$

We also evaluate our multi-hinge formulation for semi-supervised settings and similarly train a MHGAN-SSL model with Eq. (5.8), and compare it with an ACGAN-SSL model trained with Eq. (5.9) (SAGAN was not able to train stably without an auxiliary classifier).

Method	**Imagenet 64x64** IS	FID
Real data	56.67	
Baseline 1M	19.60	15.49
MHGAN 1M	**22.16**	**13.29**

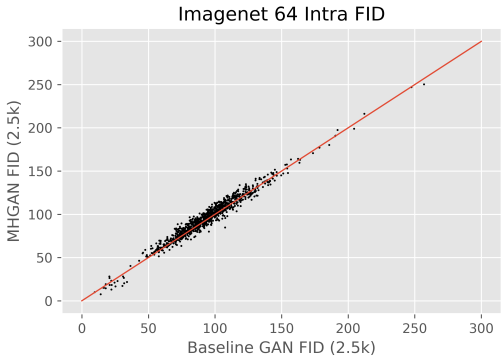
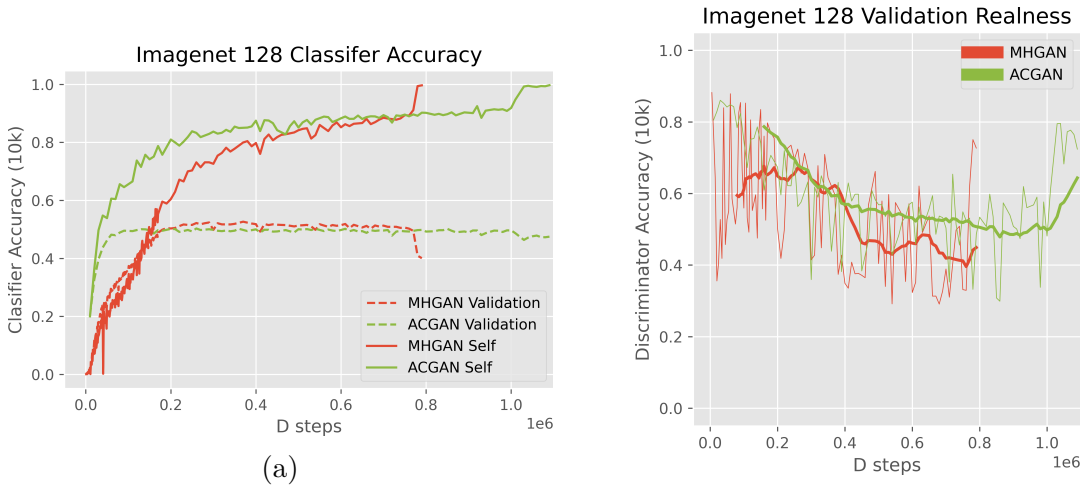


Table 5.2: Inception Scores and FIDs (50k) for supervised image generation on Imagenet 64×64. Though the margin of improvement is not as great as for Imagenet 128×128, adding the auxiliary MHGAN loss proves fruitful. Diversity is maintained at the same level (it improved in Imagenet 128×128).

For SAGAN, MHGAN, and MHGAN-SSL we optimize with size 1024 batches, learning rates of $1e-4$ and $4e-4$ for G and D, and 1 D step per G step. For ACGAN and ACGAN-SSL, we found training to be unsuccessful with only 1 D step per G step, so we train with 2 D steps per G step (training with more than 2 D steps did not improve results). For ACGAN-SSL we also used a learning rate of $5e-4$ for D and a z dimension of 120 instead of 128 in our other experiments [10]. The generator's auxiliary classifier loss weight is fixed at $\lambda = 0.1$ for all experiments. We use 64 channels and limit most of our experiments to 1,000,000 iterations. We use the Inception Score (IS) [128] and Frechet Inception Distance (FID) for quantitative evaluation of our generative models. During training we compute these scores with 10 groups of 1024 randomly generated samples using the official tensorflow implementations [1, 155], and for the final numbers in Table 5.1 we use 50k samples.



(a) (b)

Figure 5.8: Validation accuracy and self accuracy track IS and FID improvements and reach their own plateaus during training. Discriminator accuracy is very noisy during training, and behaves differently than classification accuracy, suggesting that the two tasks can be separately optimized.

5.3.1 Fully supervised image generation

In Figure 5.3 and Table 5.1 we present fully supervised results on the Imagenet dataset [156]. Previous work has noted a multitude of GAN algorithms train well on datasets of limited complexity and resolution but may not provide an indication that they can scale [157]. Thus we choose the largest and most diverse image data set commonly used to evaluate our GANs. Imagenet contains 1.3M training images and 50k validation images, each corresponding to one of 1k object classes. We resize the images to 128×128 for our experiments. On a single v3-8 TPU MHGAN and SAGAN complete 10k steps every 2 hours, and ACGAN completes 10k G-steps every 3.3 hours. Thus 1M MHGAN iterations takes 8.3 days.

We can see that the auxiliary MHGAN loss added to SAGAN trains a better GAN according to Inception score and FID. The classifier increases the fidelity of the

samples generated by the GAN without sacrificing diversity as shown in Figure 5.3c, where we plot the intra-class FID of SAGAN versus MHGAN for the best models we trained by overall IS. The mean class FID improves by 3.5%, and the class FID is lowered for 622 classes, and for 46 classes in the lower right cluster improves by an average of 50%. For the point below the cluster, mode collapse is prevented by MHGAN for the trout class. Some of these points in the cluster are shown in Figure 5.5, where we randomly sample images from classes where the FID score decreased (improved) from the SAGAN baseline to our MHGAN. We see in some of these cases that SAGAN is showing distortions or early signs of mode collapse, despite overall IS being at a high. We also show the opposite relation in Figure 5.6, where we randomly sample the classes for which FID increases (worsens) the most from SAGAN to MHGAN. From Figure 5.6 MHGAN doesn't show a worrying level of decreased diversity or mode collapse.

We also evaluated MHGAN on a smaller scale dataset of Imagenet at resolution 64×64. The conclusions from these experiments are the same but more muted compared to Imagenet at 128×128. Figures 5.7a and 5.7b show that MHGAN improves the FID and IS scores over SAGAN and ACGAN baselines, and in Table 5.2 are FID and IS numbers calculated over 50k images, and the Intra-FID comparison of SAGAN versus MHGAN which shows that diversity stayed the same between the two models.

5.3.2 Measuring the conditioning of the discriminator and generator

We attribute our model's success to the fact that it gradually incorporates class specific information into both the discriminator and the generator networks than just projection discrimination alone. When we have a classifier as in MHGAN and ACGAN it is straightforward to calculate the *Validation accuracy* and *Self accuracy*. Validation accuracy is the accuracy of the classifie on real validation data, the whole standard validation partition of the dataset is used. This measures how good the classifier learned is; if it starts to overfit the training data the we expect validation accuracy to decay. For self accuracy we test if $\arg\max_{k \in \mathcal{Y}} C_k(G(z, y))$ equals y. This measures how G incorporates the label information into its output, as measured by the concurrently trained C. This measures the self consistency of the GAN.

For the baseline SAGAN network with a projection discriminator, it is also possible to perform classification using the method mentioned in Section 5.2.1. That is our "classification" for an example x is $\arg\max_{k \in \mathcal{Y}} D(x, k)$, where each k is used as input to the projection discriminator layer that was trained.

We find that our suspicion from Section 5.2.1 is confirmed for the baseline projection discriminator model: for $(x, y) \sim p_{\mathrm{d}}$ it is not the case that $D(x, y) > D(x, k), k \neq y$ with high probability. In fact, we find that for all projection discriminators in our trained models that $D(x, y) > D(x, k), k \neq y$ uniformly and randomly.

However, the motivation behind Eq. (5.4) was to create a better GAN by training the discriminator to incorporate as much conditional information in the dataset. Though we find that the projection discriminator layer cannot be made

Method	**Imagenet-128 10%**	
	IS	FID
Real data	156.40	
ACGAN-SSL	32.38	26.12
MHGAN-SSL	**32.40**	**23.51**
S^2GAN-CO [10]	37.2	17.7
S^2GAN [10]	73.4	8.9

Table 5.3: Inception Scores and FIDs (50k) for semi-supervised image generation on Imagenet-128 trained with 10% of the data. The models were chosen by maximizing the IS within 1M iterations.

useful for classification, we can still incorporate more conditional information into the discriminator network through the embeddings it produces. In Figure 5.8a we plot validation and self accuracy for the models with a classifier (these metrics oscillate randomly near 1/ n class throughout training for SAGAN).

We see that validation performance (top-1 accuracy on Imagenet) is about the same for both models, converging around 50% and peaking at 50.32% for ACGAN and 52.66% for MHGAN. This is not competitive with purpose built classifiers which have different architectures and are typically much deeper than the discriminator of our GANs. Meanwhile classification accuracy on the training set hovers around the level of 98%. Self accuracy converges to 90% in both, and for obvious reasons goes to 100% when the generator experiences collapse (it is a curious detail that the SOTA top-1 classification accuracy on Imagenet has also been converging to 90% [158]). We leave it to future work to control self accuracy to be at the same level as validation accuracy, and for both to increase even more gradually, since this could prolong training and IS and FID improvements further. Intuitively there is a trade-off between fidelity (generated images looking like the right class) and diversity. In

Figure 5.8a we see that the absence of a logarithm, and the hinge function which limits learning on examples that are correct by a margin, leads to the more gradual incorporation of class specific information into the generator for MHGAN.

In Figure 5.8b we plot the discriminator's accuracy on the real validation set (the percentage of examples for which $D(x, y) > 0$). This is a very noisy metric during training, so we also plot the 100k itr moving average with the heavier line. It is interesting that this metric declines to around 50% as training progresses, since the same discriminator accuracy on the training set is stable at 91% on the training set for MHGAN, and 98% for SAGAN. Intuitively this indicates that D is memorizing the training set [125]. Yet since the classification accuracy remains stable or declines only slightly, this shows a disconnect between the classification and discrimination tasks.

5.3.3 The importance of separating classification and discrimination tasks

Earlier we mentioned that for all three models that we compare, the projection discriminator never incorporates class specific information; that the discriminator output has the highest affinity for the correct class about 1/ n class of the time. We observe that it is disadvantageous to try and train the projection discriminator to have the property $D(x, y) > D(x, k), k \neq y$ with high probability. Both MHGAN and ACGAN use an extra fully connected layer as an auxiliary classifier during training. Having this extra layer we saw that training with Eq. (5.5) improves both

quality and diversity as shown in Figure 5.3. It is also possible to follow the approach mentioned in Section 5.2.1 and share parameters between the projection discriminator and the classifier, and train with Eq. (5.5). We call this MHSharedGAN and illustrate this strategy in Figure 5.2b.

Training from scratch this way does not lead to satisfactory performance, but introducing this loss in the middle of SAGAN training produces interesting results (introducing it after 1M steps is also unsuccessful). After just 5k iterations after transitioning from training with Eq. (5.3) (SAGAN) to Eq. (5.5) (MHSharedGAN) Inception score skyrockets from 47.79 to 169.68 and Frechet Inception Distance drops to 8.87 (evaluated on 50k images). The catch is that unlike with MHGAN, there is a trade-off between quality and diversity being made when parameters are shared. This is illustrated in Figure 5.9, where we see the diversity of the generator drops drastically. Though the layer is spectrally normalized during training with 1 step of the power iteration method, during this second phase of training the spectrum of the projection discriminator (rank v.s. value plot of eigenvalues) changes from a sigmoid shape to a steep exponential shape, indicating that the projection has collapsed to a just a few dimensions.

We also performed the MHShared GAN finetuning on the SAGAN in Table 5.2 (trained for 1M steps) and after 15k steps the IS went to 30.47 and the FID went to 10.03 (and the validation classification accuracy went to 23.60% while the generator classification self accuracy went to 62.66%, both started at $1/n$ class). Again the diversity of the generator visibly declined; some examples of this are in Figure 5.11.

5.3.4 Semi-supervised image generation

To demonstrate our multi-hinge loss in a semi-supervised setting we use the partially labeled Imagenet data set with a random selection of 10% of the samples from each class retaining their label publicly available on TFDS [159]. We perform experiments on this dataset resized to 128×128. We keep the same architecture choices described in Section 5.3 and train our MHGAN-SSL with Eq. (5.8) and compare to ACGAN-SSL trained with Eq. (5.9). We find in Figure 5.10 that MHGAN-SSL trains faster than ACGAN-SSL but reaches a similar level of performance in Table 5.3. Both networks have their validation accuracy go to 40%, and self accuracy above 80%. The speed of of MHGAN-SSL is an advantage over ACGAN-SSL, which has a similar loss formulation to S^2GAN-CO [10], but a smaller and simpler network. We leave it to future work to train the network in S^2GAN-CO with the MHGAN-SSL loss. However it has been shown that co-training with pseudo-labels is not as competitive as pre-labeling the unlabeled data with a separate classifier and then using a fully supervised ACGAN like loss (S^2GAN-CO vs S^2GAN). We also leave it to future work to use this pre-labeling strategy with MHGAN to improve on SSL GAN training.

Figure 5.12 shows that the best FID over time was more quickly achieved by MHGAN-SSL for 1 million iterations, though the performance is not as impressive as in the fully supervised case.

5.4 Conclusion

MHGAN is a powerful addition to projection discrimination and improves training with negligible additional computational cost. Tables 5.1 and 5.3 and Figure 5.3 show that the multi-hinge loss improves both the quality and diversity of generated images on Imagenet-128. We also show how classification and discrimination tasks should not be integrated too closely, and show how the multi-hinge loss can be used to trade diversity for image quality in MHSharedGAN. MHGAN is able to perform well in both fully supervised and semi-supervised settings, and learns a relatively accurate classifier concurrently with a high quality generator.

(a) SAGAN
trained for 580k steps.
IS 47.79 and FID 17.10 (50k).

(b) The model after 5k steps of MHSharedGAN training. IS 169.68 and FID 8.87 (50k).

Figure 5.9: Enforcing class fidelity without an auxiliary classifier, and training class margins to be respected in the projection discriminator using Eq. (5.4) instead leads to low diversity, but higher quality images.

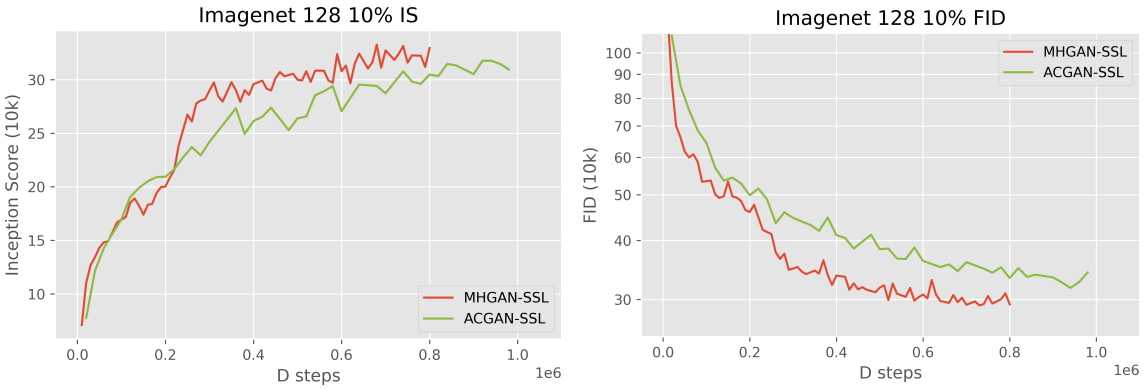
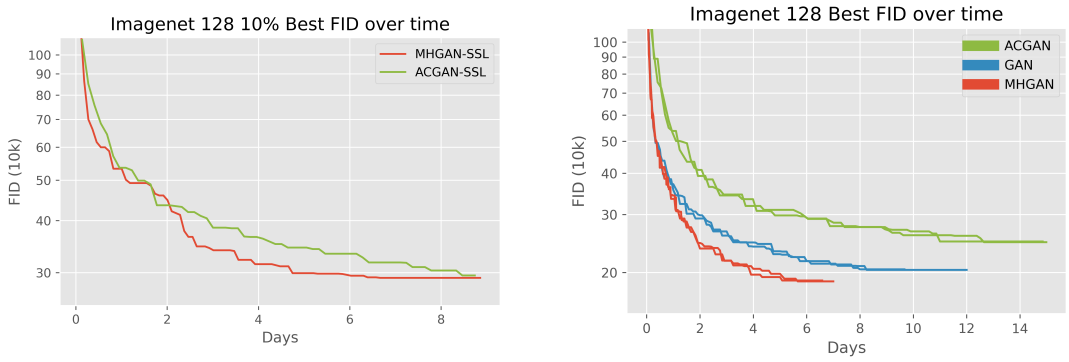


Figure 5.10: Inception Score and Frechet Inception distance when co-training a classifier in a semi-supervised setting of Imagenet-128 with 10% of labels. These experiments run at a similar rate to their fully supervised counterparts. MHGAN-SSL reaches IS and FID optima faster.

Figure 5.11: Before/After performing the MHSharedGAN finetuning to lower the diversity, but raise the quality (according to IS and FID metrics) of samples on Imagenet 64×64. The top row of images is sampled from SAGAN trained for 1M steps. And the bottom row is that same model after 15k steps using the MHSharedGAN strategy, where the projection discriminator weights are shared between a itself and a classifier optimized with multi-hinge loss.



(a) Semi-Supervised

(b) Fully Supervised

Figure 5.12: The best FID of our models plotted by duration of training. The multi-hinge loss accelerates training over SAGAN and ACGAN baselines in both fully supervised and semi-supervised settings.

Chapter 6: A study of quality and diversity in K+1 GANs

When GANs were first introduced [124], deep learning had already had striking successes with learning discriminative models while high quality deep generative models had yet to appear. The GAN framework combined these two models in an adversarial setting that allowed the successes of discriminative models to be used to train generative models that produced realistic samples. The first GAN pitted a two class real/fake discriminator against a generator network and real images [124]. Since then a multitude of innovations in the architectures of these networks, and loss functions for how to combine them, have flourished [129], with the chief goals of improving sample diversity and quality [160].

Many works have found new ways to incorporate class information into the GAN training process to improve image generation [8, 9, 139, 142, 161, 162]. In this chapter we focus on K+1 GANs which combine the task of real/fake discrimination with class discrimination directly [162]. K+1 GANs proved most successful in improving semi-supervised learning classification performance [143, 161, 162], but have more recently been challenged by a different class of methods [163, 164]. They have never been as widely successful as GANs with auxiliary classifiers (ACGAN) [9], or GANs that use class embeddings without the classification task [8, 125, 165]. Here

we investigate why the GAN architecture which most closely unifies real sample classification with fake sample classification has been overtaken by other methods. In the previous chapter we readily extended a binary formulation of the WGAN with hinge loss to a multi-class setting using inspiration from SVMs. In this chapter we take a more theoretical approach using a probabilistic interpretation of GANs.

We study a hypothesis testing inspired extension of the binary GAN into the K+1 setting [162], where K real distributions are classified against each other, and against a fake distribution. We show how the optimal discriminator/classifier in this setting is a straightforward generalization of the optimal binary GAN discriminator, and show that the formulation of the log likelihood loss prevents the generator from fully benefiting from supervision over the classes of real samples. Then we then introduce another generator loss criterion that uses dynamic labeling [161], and describe how it has increased class specificity over the K+1 GAN by emphasizing the modes of the real data. Finally we demonstrate with experiments on CelebA and CIFAR10 to what degree both GAN formulations have an effect on increasing the quality and diversity of samples over the original binary GAN and more popular ACGAN.

6.1 Binary GANs

The objective function of a binary GAN is chosen to be the log likelihood:

$$\begin{aligned} \mathrm{L}(d,g) =& p_0 \mathbb{E}[\log \mathbb{P}[D=0 \mid X] \mid H=0] + p_1 \mathbb{E}[\log \mathbb{P}[D=1 \mid X] \mid H=1] \\ =& p_0 \int \log(1-d(x)) f_0(x) dF(x) + p_1 \int \log d(x) f_1(x) dF(x) \\ =& p_0 \int \log(1-d(g(z))) f_0(g(z)) dF(g(z)) + p_1 \int \log d(x) f_1(x) dF(x) \end{aligned} \tag{6.1}$$

We use capitalized notation for the random variables on some probability triple $(\Omega, \mathcal{F}, \mathbb{P})$: $X(\omega) \in \mathbb{R}^k$ our observation (images for us), $H(\omega) \in \{0,1\}$ the always present underlying hypothesis or state of nature (the observation is from the real distribution or the fake distribution), and Z random noise. We follow the convention that the real hypothesis is 1 and the fake hypothesis is 0. Our discriminator $d(x)$ will model the probability that an observation is from the real distribution, so we also introduce a random variable $D(\omega) \in \{0,1\}$ for a decision to describe that our discriminator outputs the probability $d(x) = \mathbb{P}[D=1 \mid X=x, H]$ (at test time our strategy is implemented as $\arg\max\{d(x), 1-d(x)\}$, the formalism here is for the training of d and g). We assume the existence of two conditional probability distributions $F_h(x) = \mathbb{P}[X \leq x \mid H=h],\ h=0,1$ on $\mathbb{R}^k$ which are both absolutely continuous with respect to some distribution F on $\mathbb{R}^k$. We denote the fake and real densities induced by F_0, F_1 respectively as $f_h(x) = \mathbb{P}[X=x \mid H=h],\ h=0,1$. We denote the generator as the Borel measurable mapping $g : z \mapsto x$. The random

variable $g(Z)$ has a density which is set to $f_0(g(z))$. Where $p_h = \mathbb{P}[H = h], h = 0, 1$ is the prior on each hypothesis (often taken to be uniform). The discriminator seeks to maximize this objective while the generator seeks to minimize it. Maximizing $\mathrm{L}(d, g)$ over discriminators gives the pointwise solution $d^*(x) = p_1 f_1(x)/(p_0 f_0(x) + p_1 f_1(x))$. Thus, letting $f_{\text{avg}}(x) = p_0 f_0(x) + p_1 f_1(x)$ be the average of the densities and $f(\cdot, h) = f_h(\cdot) p_h, i = 0, 1$ be the joint densities:

$$\begin{aligned} \mathrm{L}(d^*, g) &= \int \log \frac{p_0 f_0(x)}{f_{\text{avg}}(x)} p_0 f_0(x) dF(x) + \int \log \frac{p_1 f_1(x)}{f_{\text{avg}}(x)} p_1 f_1(x) dF(x) \\ &= KL(f(\cdot, 0) \mid\mid f_{\text{avg}}(\cdot)) + KL(f(\cdot, 1) \mid\mid f_{\text{avg}}(\cdot)) \end{aligned} \tag{6.2}$$

The solution to $\min_g \mathrm{L}(d^*, g)$ when $p_0 = p_1 = 1/2$ is $f_0(g(z)) = f_0(x) = f_1(x)$. And more generally: $p_0 f_0(x) = p_1 f_1(x)$ [124].

6.2 K+1 GANs

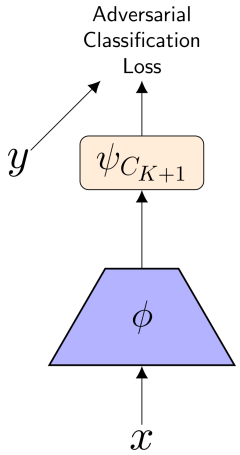


Figure 6.1: The $K + 1$ Improved GAN architecture. $\psi_{C_{K+1}}$ denotes the $K + 1$ classification layer. Note that there is no projection discriminator layer.

The generalization of the binary GAN [124], known as the K+1 GAN (and

also as LabelGAN [161]), was originally developed for semi-supervised learning with GANs [162]. This architecture is shown in Figure 6.1. We let $M = K + 1$ denote the total number of classes, K real classes plus the fake class, which we refer to as the zeroth class. This generalization echoes the form of the generalization of binary minimax hypothesis testing to be M-ary [166]. In this setting an observation $X(\omega)$ may occur under one of M hypotheses, for example an image X may be of a bird, a truck, a fake image, a frog, etc.. We generalize the discriminator to not just decide whether an example is from the real or fake hypothesis, but whether an example is from one of $M-1$ real data hypotheses or from an additional fake hypothesis. Now the state of nature take values $H(\omega) \in \{0, 1, \ldots, M-1\}$, and the same for our decision random variable $D(\omega)$. We use the convention that 0 still denotes the fake hypothesis, and $1, 2, \ldots, M-1$ denote the multiple real hypotheses (for example image classes). Our discriminator $d(x)$ outputs a vector in $[0, 1] \in \mathbb{R}^M$ of probabilities $d_m(x) = \mathbb{P}[D = m|X = x], m = 0, 1, \ldots, M-1$ to signify the probability that the decision should be m (at test time our decision strategy will simply be $\arg\max_m d_m(x)$). We denote the distributions of X under each hypothesis as $F_m(x) = \mathbb{P}[X \leq x|H = m], m = 0, 1, \ldots, M-1$, all absolutely continuous with respect to some distribution F, and all with densities f_m. Our generator g is unchanged from the binary GAN setting, and we again set the density of $g(Z)$ to f_0. The objective function of the binary GAN naturally generalizes to training all

outputs of the discriminator:

$$\begin{aligned}
\mathrm{L}_{\mathrm{K}+1}(d, g) &= \sum_{m=0}^{M-1} p_m \mathbb{E}[\log \mathbb{P}[D = m \mid X] \mid H = m] \\
&= \sum_{m=0}^{M-1} p_m \int \log d_m(x) f_m(x) dF(x) \\
&= \sum_{m=1}^{M-1} p_m \int \log d_m(x) f_m(x) dF(x) + p_0 \int \log d_0(g(z)) f_0(g(z)) dF(g(z))
\end{aligned} \tag{6.3}$$

We denote the fake and real densities as $f_h(x) = \mathbb{P}[X = x \mid H = h],\ h = 0, 1, \ldots, K$ and p_m is the prior on each hypothesis (often taken to be uniform).

Proposition 1. *For each $x \in \mathbb{R}^k$ the optimal vector valued decision strategy $d^*(x)$ exists and each of its M elements is given by $d_m^*(x) = p_m f_m(x) / \sum_{i=0}^{M-1} p_i f_i(x)$.*

This can be proved for discrete densities using the classic optimality criterion for differentiable convex objectives [167].

An optimality criterion for differentiable convex objectives [167]. Suppose f is differentiable (with domain Df) s.t. we have the first order condition $f(y) \geq f(x) + \nabla f(x)^T (y - x), \forall x, y \in Df$, then $x \in D^*$ (the feasible set) is optimal if and only if $\nabla f_0(x)^T (y - x) \geq 0, \forall y \in D^*$. $\nabla f_0(x)$ is the direction you can increase the objective function, it is the supporting hyperplane at the boundary point x.

Proof. (of Proposition 1)

$$\mathrm{L}_{\mathrm{K}+1}(d, g) = \int \sum_{m=0}^{M-1} p_m f_m(x) \log d_m(x) dF(x) \tag{6.4}$$

Fix x in $\sum_{m=0}^{M-1} p_m f_m(x) \log d_m(x)$, and consider the following convex optimization problem where $p_m f_m(x)$ becomes f_m and $d_m(x)$ becomes d_m:

$$\min_d L(d) = -\sum_{i=0}^{M-1} f_i \log d_i \ s.t. \ \sum d_i = 1, d \geq 0 \tag{6.5}$$

with $f_i \geq 0$ constants. Let c, d be vectors in the feasible set and let c_i, d_i denote their elements. First we check that we have the first order condition for c and d, noting that $\nabla L(d)_i = -\frac{f_i}{d_i}$ and that $\log x \leq x - 1, x \geq 0$:

$$-L(\frac{c}{d}) = \sum_{i=0}^{M-1} f_i \log \frac{c_i}{d_i} \leq \sum_{i=0}^{M-1} f_i(\frac{c_i}{d_i} - 1) = \sum_{i=0}^{M-1} \frac{f_i}{d_i}(c_i - d_i) = -\nabla L(d)^T(c - d) \tag{6.6}$$

since $L(d) - L(c) = -L(\frac{c}{d})$, this shows that $L(c) - L(d) \geq \nabla L(d)^T(c - d)$. Next we can get the solution d^* by looking at the iff condition. Suppose $\exists d^* \in D^* \ s.t. \ \nabla L(d^*) = -K\mathbf{1}$:

$$\begin{aligned} \nabla L(d^*)^T(c - d^*) &\geq 0 \\ -K\mathbf{1}(c - d^*) &\geq 0 \\ \sum c_i - \sum d_i^* &\leq 0 \end{aligned} \tag{6.7}$$

Indeed is true for all c in the feasible set. Therefore $d_i = f_i / \sum_{j=0}^{M-1} f_j$ is optimal. This solution is on the edge of the feasible set. Because $d_m^*(x) = p_m f_m(x) / \sum_{i=0}^{M-1} p_i f_i(x)$. d^* for each x this is an algebraic combination of Borel measurable functions, it is Borel measurable. $\square$

Note that this result easily simplifies to the binary GAN result when $M = 2$ and $p_0 = p_1 = 1/2$ [124]. Continuing to generalize the binary GAN game of $\min_g \max_d \mathrm{L}(d, g)$ we see:

$$\begin{aligned} \mathrm{L}_{\mathrm{K}+1}(d^*, g) &= \sum_{m=0}^{M-1} \int \log \frac{p_m f_m(x)}{f_{\mathrm{avg}}(x)} p_m f_m(x) dF(x) \\ &= \sum_{m=0}^{M-1} KL(f(\cdot, m) \mid\mid f_{\mathrm{avg}}(\cdot)) \end{aligned} \tag{6.8}$$

where $f_{\mathrm{avg}}(x) = \sum_{m=0}^{M-1} p_m f_m(x)$.

Any M-ary problem can be converted to a binary problem by changing the task from deciding between $\{0, 1, \ldots, M-1\}$ to $\{0, \mathrm{not}0\}$, and deciding not0 is equivalent to deciding "real." The following Proposition 2 says that the loss for the optimal strategy for an M-ary problem will be lower bounded by that strategy applied to the corresponding binary problem.

Proposition 2. *Let* $\min_g \max_d \mathrm{L}_{\mathrm{K}+1}$ *be an M-ary problem with some densities* $f_m, m = 1, 2, \ldots, M-1$. *Let* d^* *be optimal in the sense of Proposition 1. Define a new binary GAN problem (we use* $\widetilde{\mathrm{var}}$ *to indicate a variable is for the new binary GAN problem)* $\min_{\widetilde{g}} \max_{\widetilde{d}} \mathrm{L}$ *with the real density* $\widetilde{f}_1$ *as the average of all the* $M-1$ *real density* $\widetilde{f}_1 = f_{\mathrm{real}}(x) = \sum_{m=1}^{M-1} p_m f_m(x)$. *Then the optimal discriminator* $\widetilde{d}^*$ *in this binary GAN problem will be* $1 - d_0^*(x)$ *and* $\min_g \mathrm{L}_{\mathrm{K}+1}(d^*, g) \geq \min_{\widetilde{g}} \mathrm{L}(1 - d_0^*, \widetilde{g}) = \mathrm{L}(1 - d_0^*, \widetilde{g}^*)$ *where* $\widetilde{g}^*$ *has the density* $p_0 \widetilde{f}_0(x) = f_{\mathrm{real}}(x)$.

Proof. The optimal discriminator for the binary GAN problem follows from Propo-

sition 1 and noting $1 - d_0^*(x) = f_{\text{real}}(x)/f_{\text{avg}}$. By the log-sum inequality:

$$\sum_{m=1}^{M-1} \log \frac{p_m f_m(x)}{f_{\text{avg}}(x)} p_m f_m(x) \geq \left(\sum_{m=1}^{M-1} p_m f_m(x) \right) \log \frac{\left(\sum_{m=1}^{M-1} p_m f_m(x) \right)}{(M-1) f_{\text{avg}}(x)}$$
$$= f_{\text{real}}(x) \log \frac{f_{\text{real}}(x)}{p_0 f_0(x) + f_{\text{real}}(x)} - f_{\text{real}}(x) \log(M-1) \tag{6.9}$$

where the last term has no dependence on g. Plugging in we see:

$$\min_g \mathrm{L}_{\mathrm{K}+1}(d^*, g) \geq \min_g \int \log \frac{p_0 f_0(x)}{f_{\text{avg}}(x)} p_0 f_0(x) dF(x) + \int \log \frac{f_{\text{real}}(x)}{f_{\text{avg}}(x)} f_{\text{real}}(x) dF(x) \tag{6.10}$$

where we recognize the right hand side is: $\min_g \mathrm{L}(1 - d_0^*, g)$, so we make the appropriate reparameterizations $f_0 \to \widetilde{f}_0,\ g \to \widetilde{g}$, and the minimum of the binary GAN problem was addressed in Section 6.1. Equality is achieved in the log-sum inequality when $p_m f_m(x), m = 1, 2, \ldots, M-1$ are equal. □

Of course in the original binary GAN setting of $M = 2$ and $p_0 = p_1 = 1/2$, both sides of the bound are identical. For $M > 2$ this lower bound is not tight at the minimum $f_0 = \widetilde{f}_0 = f_{\text{real}}/p_0$ and we cannot directly say that $\widetilde{g}^*$ from the constructed binary GAN problem minimizes the M-ary GAN problem. But the following remark will help provide evidence for the claim that indeed for $M > 2$ too $p_0 f_0(x) = f_{\text{real}}(x)$, that is that training the M-ary discriminator d to classify the real examples in parallel with telling fake examples apart from real examples does not change the generator learned.

Remark 1. *For discrete PMFs $f_m, m = 0, 1, \ldots, M-1$, we may write the following*

analogue of Equation (6.8)*:*

$$\min_{f_0} \mathrm{L}_{\mathrm{K+1}}(d^*, g) = \min_{f_0} \sum_{m=0}^{M-1} KL(f(\cdot, m) \mid\mid f_{\mathrm{avg}}(\cdot)) \tag{6.11}$$

which is a convex optimization problem. The empirical solution we observe is $p_0 f_0(x) \approx f_{\mathrm{real}}(x) = \sum_{m=1}^{M-1} p_m f_m(x)$.

6.2.1 Summary

The typical intuition in training neural networks is that learning from more data and sharing parameters across tasks leads to better performance. But here, surprisingly the theory states that by using class information in the $K+1$ GAN to train a discriminator that can discriminate between classes, and between fake images and each of the classes, is no better than ignoring the class information. That is, if we have labeled classes for data, but choose to throw it away by merging $f_1, \ldots, f_K$ into a single f_{real}, the generator of the binary GAN that we train should have the same distribution f_0 as if we trained the K+1 GAN. This result is also consistent with examining the gradients of the K+1 GAN which reveal a "overlaid-gradient" problem whereby the overall gradient w.r.t. a generated example is the same as that in the binary GAN [161].

The $K+1$ GAN's original purpose was however to improve the state of the art in semi-supervised classification [162], rather than improve generation of images (though human annotators were said to prefer images from a K+1 GAN). We note that in Equation (6.3) we omitted the part of the loss for unlabeled real images.

Nevertheless we have shown that counter-intuitively, training a fully supervised $K+1$ discriminator to classify the real examples in parallel with telling fake examples apart from real examples does not change the generator learned.

6.3 Dynamic Labeling and Mode Emphasizing GAN

One extension that increases the class specificity of the generator learned by a K+1 GAN is to introduce dynamic labeling [161]. Such a choice leads to f_0 converging to emphasize the modes of $f_1, \ldots, f_K$, rather than being the average of them f_{real}.

Looking at the last integral in Equation (6.3), we notice that the g^* that solves $\min_g \mathbb{E}[\log \mathbb{P}[D = 0 \mid X] \mid H = 0]$ also solves $\min_g -\mathbb{E}[\log \mathbb{P}[D \neq 0 \mid X] \mid H = 0] = \min_g -\mathbb{E}[\log \sum_{m>0} \mathbb{P}[D = m \mid X] \mid H = 0]$. This equivalence provides the intuition for why the density of the generator will be the average of all the real distributions. Let: $\mathrm{L}_{\text{Dyn}}(d, g) = -\mathbb{E}[\log \max_{l>0} \mathbb{P}[D = l \mid X] \mid H = 0]$. That is for each fake $g(z)$ we look at the "most likely mistake" the discriminator makes $d(g(z))$, and train g to emphasize it. For d we keep the K+1 GAN objective unchanged. If we let d^* be optimal in the sense of the previous section, then

$$\min_g \mathrm{L}_{\text{Dyn}}(d^*, g) = \min_g KL(f_0 \mid\mid \text{ptwise } \max_{m>0} f_m) - KL(f_0 \mid\mid f_{\text{avg}}) \tag{6.12}$$

Comparing this to Equation (6.8), our intuition expects $f_0(x) \approx \max_{m>0} f_m(x)$ with Dynamic Labeling instead of $f_0(x) = \sum_{m>0} f_m(x)$ in K+1 GAN for the following reason.

Remark 2. *For discrete PMFs* $f_m, m = 0, 1, \ldots, M-1$ *we may write the following analogue of Equation* (6.12)*:*

$$\min_{f_0} \mathrm{L}_{\mathrm{Dyn}}(d^*, g) = \min_{f_0} KL(f_0 \mid\mid \text{ptwise } \max_{m>0} f_m) - KL(f_0 \mid\mid f_{\mathrm{avg}}) \tag{6.13}$$

We note that

$$\min_{f_0} \mathrm{L}_{\mathrm{Dyn}}(d^*, g) \leq \min_{f_0} KL(f_0 \mid\mid \text{ptwise } \max_{m>0} f_m) \tag{6.14}$$

where the right hand side is a convex optimization problem. The empirical solution we observe $f_0(x) \approx \max_{m>0} f_m(x)$ *matching our intuition.*

Dynamic labeling GAN was found to be experimentally preferable to competing non-dynamically labeled GANs in the sense of Inception Score (IS) [161]. Our analysis here, which looks at the unconditional K+1 GAN with dynamic labeling, explains why but also reveals a potential failure mode. IS is proportional to the KL divergence of $\mathbb{P}(Y|X)$ and $\mathbb{P}(Y)$ for images X and labels Y from a pretrained Inception v3 network [162, 168]. Thus a generator that emphasizes all the modes (classes) of the real data distribution should generate easily recognizable images and have low entropy $\mathbb{P}(Y|X)$ and have a high entropy $\mathbb{P}(Y)$ by displaying all the classes, giving a high IS. However emphasizing the modes of the data could also result in lower observed diversity from finite samplings of g since the density of f_0 is concentrated around the modes. For this reason we also refer to this GAN as "Mode GAN".

6.4 Multi-hinge loss for K+1 GANs

With MHShared GAN in Chapter 5 we demonstrated a fine-tuning strategy that shares parameters between the projection discriminator for real/fake discrimination and a classifier layer for classification and can improve IS and FID of the generated samples but loose diversity. This provided some evidence that the tasks of classification and discrimination should not be too closely integrated for successfully training a GAN. In developing MHGAN we also experimented with $K+1$ [128] formulations of the loss where the discrimination and classification tasks were on equal footing. This provides another training strategy that brings the tasks of discrimination and classification close together as shown in Fig. 6.1.

More specifically, we denote the classifier architecture as $C : x \to \mathbb{R}^{K+1}$ where the class labels are $y \in \{1, \cdots, K\}$ and the label for fake images is 0. Note that while the previous networks had outputs which were modeled as probabilities, here the classifier only outputs affinities. $C_k(x)$ denote the kth element of the vector output of C for an example x, which represents the affinity of class k for x. For this loss we require a conditional generator, G becomes a function of the noise-label pair $(z \sim Z, y \sim \text{Unif}(1, \ldots, K))$. The multi-hinge loss for the classifier $L_{K+1,\text{MH}}$ is:

$$\begin{aligned} L_{K+1,\text{MH}-\text{C}} = &\sum_{h=1}^{K} \mathbb{E}[\max(0, 1 - C_m(X) + C_{\neg m}(X)) \mid H = h] \\ &+ \sum_{h=1}^{K} \mathbb{E}[\max(0, 1 + C_{\neg 0}(G(Z, h)) - C_0(G(Z, h))) \mid H = h] \end{aligned} \tag{6.15}$$

where $C_{\neg y}(x)$ is the classifier's highest affinity for any label that is "not y": $C_{\neg y}(x) =$

$\max_{k \neq y} C_k(x), y = 0, 1, \ldots, K$.

For the generator, we pair the class specific multi-hinge loss with a class agnostic feature matching loss:

$$\begin{aligned} L_{K+1,\text{MH}-\text{G}} &= \lambda\, L_{K+1,G\text{cs}} + L_{K+1,G\text{fm}} \\ &= \lambda \sum_{h=1}^{K} \mathbb{E}[\max(0, 1 - C_h(G(Z,h)) + C_{\neg h}(G(Z,h))) \mid H = h] \\ &\quad + \left\| \sum_{h=1}^{K} \mathbb{E}[C_{\text{feat}}(G(Z,h)) \mid H = h] - \mathbb{E}[C_{\text{feat}}(X)] \right\|_1 \end{aligned} \tag{6.16}$$

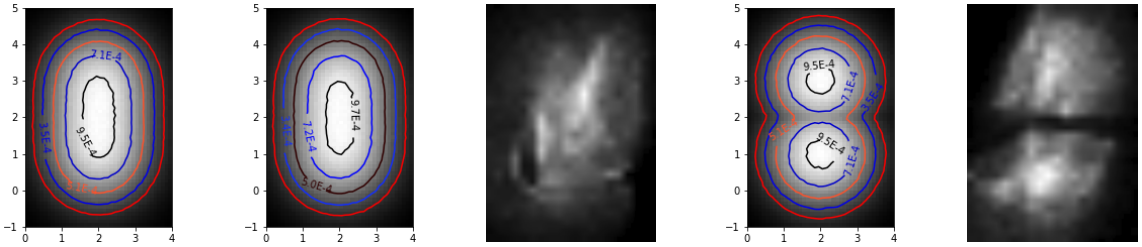
where C_{feat} denotes the features before the final classification layer of C. Substituting $L_{K+1,G\text{fm}}$ with a more WGAN like choice of $\mathbb{E}[\ \mathbb{E}[C_0(G(Z,h)) \mid H = h]\ ]$ did not train as stably. We abbreviate the $K+1$ classifier and generator pair with trained with Eqs. (6.15) and (6.16) as $K+1$ MHGAN.

6.5 Experiments and Discussion

6.5.1 Methods

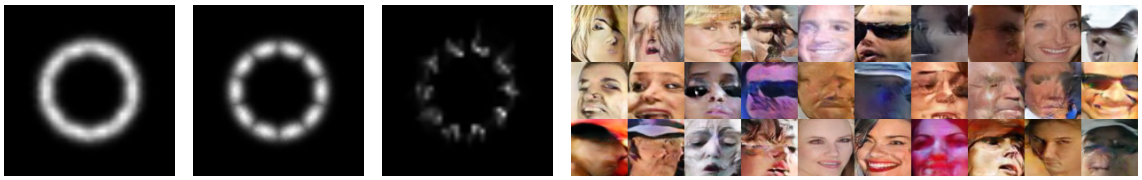
We demonstrate our findings on K+1 GANs using four datasets: synthetic 2D Gaussians, CelebA, and CIFAR10 and CIFAR100.

Overlapping Gaussians. In this dataset our observations $x \in \mathbb{R}^2$ are distributed normally. These data PMFs are plotted in Figure 6.2a. We also create a dataset of 10 highly overlapping 2D Gaussians whose means are distributed uniformly on a circle, samples from different Gaussians are different "classes". Samples from all the classes are shown in Figure 6.3a.



(a) Data, Averaged (b) Expected K+1 GAN (c) Learned K+1 GAN (d) Expected Mode GAN (e) Learned Mode GAN

Figure 6.2: Distributions of 4 generators on 2D Gaussian data on $(0, 4) \times (-1, 5)$. (b) shows the PMF that minimizes Equation (6.11), which is the distribution that we expect a GAN trained with L_{K+1} to learn. (c) shows the histogram of a generator trained with L_{K+1}. (d) shows the PMF that minimizes Equation (6.14), which is the distribution that we expect a GAN trained with L_{Dyn} to learn. (e) shows the histogram of a generator trained with L_{Dyn}.



(a) Data (b) f_0 Exp. (c) f_0 (d)

Figure 6.3: Mode emphasis of Dynamic Labeling GAN. ((a)) shows 10 overlapping classes of 2D Gaussian data arranged on a circle within $(0, 4) \times (-1, 5)$. ((b)) shows the PMF that is the solution of the convex upper-bound of Equation (6.12), this is the distribution that we expect a GAN trained with L_{Dyn} to learn. ((c)) shows the histogram of a generator trained with L_{Dyn}. ((d)) shows a random selection 10 images from each GAN that have a MTCNN confidence of 0. The first row from is from a binary GAN, the second row is from ACGAN, and the third row from Dynamic Labeling GAN. This shows that even if modes are emphasized, this does not necessarily mean samples are not realized in what could be considered undesirable low density regions of f_{real}.

CelebA. CelebA is a dataset with 40 binary attribute annotations and 5 landmark locations, the images we use are 64×64 and have the background cropped. From the landmarks we can estimate the face rotation from the ratio of eye distance and nose to mouth distance (using the rule of thirds). To train our supervised models Mode GAN and ACGAN we split the dataset into 5 categories: Non-rotated female/male with mouth open/closed are 4, and all rotated faces (more than 15 degrees) are the

5th.

CIFAR10 & CIFAR100. CIFAR10 is a 32×32 dataset with 10 classes, and CIFAR100 is the same dataset with the labels specified into 100 classes. We use all 50,000 images for training, and the 10 classes are the labels for our supervised models.

We train three SN GAN networks [165] (Binary GAN, ACGAN, Mode GAN) with default hyperparameters and only vary the loss function used. We also train two additional SN GAN networks (Mode GAN and K+1 GAN) that also used projection discrimination [134]. Finally we train conditional architecture [135] variants of GAN, ACGAN, and K+1 MHGAN for CIFAR100. All networks were trained for 200k iters for CelebA and 500k iters for CIFAR10 and CIFAR100.

6.5.2 Discussion

When the form of the true densities $f_m(x)$ are known, we can observe the learned density of the generator f_0 via the histogram of its samples. For the Dynamically Labeled GAN we see the PMF that minimizes the convex upper-bound of Equation (6.12) in Figure 6.3b emphasizes the modes of the data. In Figure 6.3c we see that the learned g from minimizing Equation (6.12) while training a GAN indeed emphasizes the modes of the data: the samples are in high density around the means of each class, and are low density in the overlap between classes. However, this simple example does not generalize neatly to real data. We may expect an analogous situation in the domain of faces to be that few samples exist between

the "means" of the 5 classes we chose: i.e. no interpolations between males facing left and females facing right. But we see in Figure 6.3d that Dynamically Labeled GAN generates unwanted low quality samples just like the baseline Binary GAN and ACGAN networks.

To measure the quality of our CelebA GAN's output, we use the confidence output of MTCNN, a publicly available pretrained face detector, to measure the frequency of bad generator samples. In Figure 6.3d are examples of images that MTCNN assigns 0 confidence output by all 3 GANs. For images that are far away from the modes of the CelebA data density, for example images that blur together multiple face orientations into 1 image, MTCNN will assign a low confidence. The right hand side of Figure 6.4 shows out of 100,000 samples generated by each GAN, the percentage of images with MTCNN confidence greater than $c = 0, 1 - 10^{-1}, \ldots, 1 - 10^{-4}$ is always largest in Mode GAN, but only by a small margin. And as shown in the figure, all the networks produce a score 0 face 4-6 times every 10000 images.

The left hand side of Figure 6.4 shows that performance was tied for the three networks on CelebA and CIFAR10. Adding projection discrimination (PJ) to the networks yeilded a much greater difference in IS performance than the changes in loss function. We show some samples from the CIFAR10 GANs in Figure 6.5, which are hard to qualitatively rank.

To measure diversity within a batch of generated images, we embed the images in a feature space with an auxiliary network, calculate all the pairwise distances, and record the mean. In Figure 6.5d we see Mode GAN always has a higher mean

	CelebA FID	CIFAR-10 IS
SN GAN	4.13	8.03
SN ACGAN	5.07	8.02
SN Mode GAN	3.65	8.13
PJ SN Mode GAN		8.50
PJ SN K+1 GAN		8.43

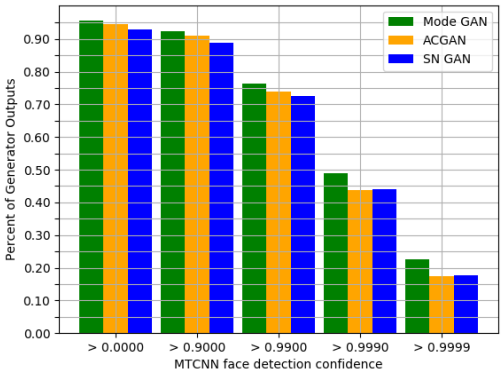


Figure 6.4: On the left are FIDs image generation on CelebA and Inception scores for CIFAR-10. On the right is the percentage of 100,000 samples generated by each GAN for which MTCNN has greater than c confidence that a face is present, where $c = 0, 1 - 10^{-1}, 1 - 10^{-2}, 1 - 10^{-3}, 1 - 10^{-4}$. We use the publicly available pretrained MTCNN face detector network to compute the confidence scores.

(a) SNGAN (b) ACGAN (c) Mode GAN (d) CIFAR10

Figure 6.5: Generated images on CIFAR10 for the three methods we compare. Images are randomly selected from each generator and sorted by class by an auxiliary DenseNet121 classifier with 95% test accuracy into the 10 classes of CIFAR10: plane, car, bird, cat, deer, dog, frog, horse, ship, truck. In (d) we plot the mean pairwise distance of the outputs of the three GANs. That is for each image in a batch of size in the range $n * 128, n = 1, 2, \ldots, 79$ an embedding was generated by an auxiliary network, and then the pairwise Euclidean distances were computed. For each of these 79 batches of increasing size the mean embedding distance per generator was recorded. In each plot the average of 10 trials is shown with the standard deviation shaded. The embedding network was the 1,024 dimension penultimate layer of a DenseNet-121.

pairwise distance only within a margin of error. Thus we don't observe that fitting the modes of real data yields a big impact on quality or diversity of samples.

At low resolutions (48×48 and below) we observed that such models are

able to train from scratch and obtain competitive IS and FID scores, but that fidelity came at the cost of diversity there too. In Figure 6.6 we show that $K+1$ MHGAN can perform well from the standpoint of Inception Score, less so from the standpoint of FID, but increases the intra-class FID by 29%. Unifying the classifier and the discriminator leads to an excellent Inception Score for CIFAR100, and a good FID, but the mean IFID score goes from 74.70 for the SAGAN to 96.32 for the K+1 MHGAN, indicating a drop in diversity. MHGAN led to an increase in quality and diversity in generated images. But Similar to MHSharedGAN, MH K+1 Improved GAN shows that sharing parameters between the tasks of classification and discrimination leads to a less desirable GAN.

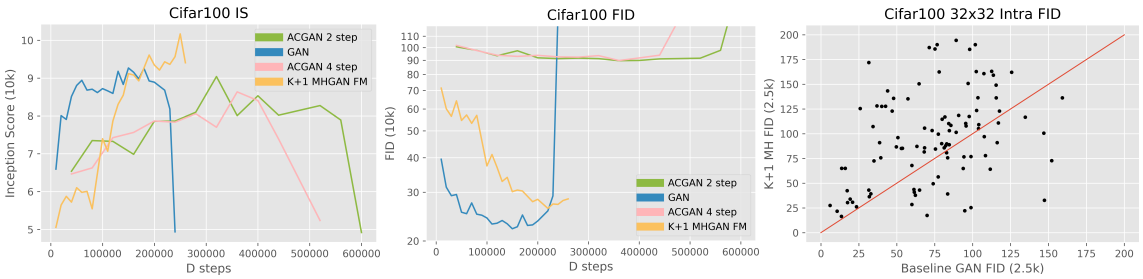


Figure 6.6: Comparing MH K+1 Improved GAN with SAGAN and ACGAN baselines on CIFAR100.

Our experiments show that classification should be left as an auxiliary task to discrimination, and not combined with it by sharing parameters too closely.

6.6 Conclusion

We studied the K+1 GAN paradigm which generalizes the canonical true/fake GAN by training a generator with a K+1-ary classifier instead of a binary discriminator. We showed how the standard formulation of the K+1 GAN does not take

advantage of class information fully and show how its learned generative data distribution is no different than the distribution that a traditional binary GAN learns. We then investigated another GAN loss function that dynamically labels its data during training and showed how this leads to learning a generative distribution that emphasizes the target distribution modes, but we saw that this has a tenuous link to increased quality and diversity of generated samples. We also modified the multi-hinge loss of Chapter 5 for K+1 GANs, and accumulated more evidence that classification should remain a separate task from real/fake discrimination for the best quality and diversity GAN generator.

Chapter 7: Summary and Suggestions for Future Work

In Chapters 2 and 3 we studied the tradeoffs between learnable and classical bases. In Chapter 2 we addressed source separation of widely varying sound events in audio recordings. Recent deep learning approaches have achieved impressive performance on speech enhancement and separation tasks. However, these approaches have not been investigated for separating mixtures of arbitrary sounds of different types, a task we refer to as *universal sound separation*, and it is unknown how performance on speech tasks carries over to non-speech tasks. We compared two paradigms of source separation models, class-based masking and multiple-instance segmentation, on large datasets we created explicitly for exploring the capacity of models traditionally successful on speech data. Finding that multiple-instance segmentation is better suited to real world tasks, we investigated the space of mask-based separation architectures, varying both the overall network architecture and the framewise analysis-synthesis basis for signal transformations. These network architectures include convolutional long short-term memory networks and time-dilated convolution stacks inspired by the recent success of time-domain enhancement networks like ConvTasNet. For the latter architecture, we also proposed novel modifications that further improve separation performance. In terms of the framewise

analysis-synthesis basis, we explored both a short-time Fourier transform (STFT) and a learnable basis, as used in ConvTasNet. For both of these bases, we also examined the effect of window size. In particular, for STFTs, we found that longer windows (25-50 ms) work best for speech/non-speech separation, while shorter windows (2.5 ms) worked best for arbitrary sounds. For learnable bases, shorter windows (2.5 ms) worked best on all tasks. Surprisingly, for universal sound separation, STFTs outperform learnable bases.

In Chapter 3 we investigated new techniques that are able to capture the special physical properties of hyperspectral data for hyperspectral image (HSI) classification, the one of the most active tasks in geoscience and remote sensing. The special property of this data over color images is an additional spectral dimension that has great physical relevance. In the aim of classifying what material lies in a single pixel within a HSI cube, the spectrum of that pixel should contain the spectral signature of the material. Confounding noise factors can then be accounted for by investigating neighboring pixels. This means a feature extractor for HSI data should jointly extract spectral-spatial features. Simultaneously HSI data is also greatly limited, so a feature extractor must not need more than a handful of examples per material for training to be applicable in real world scenarios. We built a bridge between time-frequency methods that decompose spectra into multi-spectral bands, with hierarchical neural networks that incorporate spatial information across scales and model multiple levels of dependencies between spectral features. To accommodate for the low training sample scenario we investigated the use of the Fourier scattering transform, which is an amalgamation of time-frequency representations with

neural network architectures, to advance the state of the art in spectral-spatial classification. We tested the proposed three dimensional Fourier scattering method on standard hyperspectral datasets, and present results that indicate that the Fourier scattering transform is highly effective at representing spectral data when compared with other state-of-the-art spectral-spatial classification methods.

In Chapter 4 we used a similar multi-scale approach to develop a defense against audio adversarial attacks. We propose a unification of a computational model of speech processing in the brain with commercial wake-word networks to create a cortical network, and show that it can increase resistance to adversarial noise without a degradation in performance. We integrated knowledge from the cortical representation of speech studied in Neuroscience to develop a defense against adversarial attacks on audio. We enhanced a neural networks used for wake-word detection by integrating this biological representation for sound processing, and found that the accuracy of the architecture did not degrade, while resistance to adversarial noise increased several times.

We also examined how semantic information can be further exploited in generative tasks in Chapters 5 and 6. In Chapter 5 we proposed a new algorithm to incorporate class conditional information into the discriminator of GANs via a multi-class generalization of the commonly used Hinge loss. Like in SVMs we showed how to control the margins that a spectrally constrained Wasserstein GAN learns. Our approach is in contrast to most GAN frameworks in that we train a classifier and conditional discriminator with the same hinge loss style loss function, instead of an auxiliary cross entropy classifier. We also used the classifier to evaluate

the conditioning of the discriminator, and show that the consistency between the class conditioned generator inputs and class specific generator outputs is improved without our loss. We showed the dangers of connecting the tasks of discrimination and classification too closely, and the resulting trade-off between class-fidelity (and therefore quality) and diversity. With our multi-hinge loss modification we were able to improve the state of the art Inception Scores and Frechet Inception Distances on the popular vision Imagenet-128 dataset.

In Chapter 6 we continued to look at how closely classification and discrimination can be combined in GANs. We studied the multiclass image generation problem from a fundamental hypothesis testing framework, taking a probabilistic interpretation of GANs. We introduced theory for the K+1 GAN network and training paradigm, and found that a single GAN cannot generate class specific samples when conditioned with a straightforward cross entropy loss. We also study an adapted loss that emphasizes the class modes of the target distribution through dynamic labeling (Mode GAN), which is manifested by the generator learning a distribution with decreased density near class boundaries. This chapter gathered more evidence to confirm the principle learned in Chapter 5 that discrimination and classification cannot be combined too closely, as it results in trading diversity for quality for GANs.

We now discuss some further methods to exploit the semantic content of data.

7.1 Open Issues and Directions for Future Work

7.1.1 Adversarial Audio Attacks on Smaller Networks

We applied our cortical defense to network a network with minimal preprocessing, but nonetheless reportedly used by Amazon Alexa [108]. Some other commercial voice assistants such as Google Assistant make miniaturizing changes to their wake-word detection and speech recognition algorithms to save on power [169, 170]. These include local dynamic normalization instead of global normalization, quantized weights, and smaller features [170]. It has not been investigated what effect these choices make on the robustness or vulnerability of their ASR systems. There are opportunities for developing new adversarial attacks that exploit these attributes, and it is also important to evaluate defenses that are compatible with these low-power architecture choices.

Our cortical defense focused on integrating what is known about the auditory cortex into a defense, but there is also an opportunity to refine the auditory spectrogram. The techniques to computationally mimick the transduction of the hair cells and the reduction of the lateral inhibitory network are differentiable and their inclusion could be investigated in their impact on adversarial examples.

7.1.2 Transfer learning with Multispectral and Hyperspectral Imagery

An advantage of the method we presented in Chapter 3 is its compatibility with conventional deep learning implementations. This readily allows for a shift

from the pre-processing based classification with a linear SVM we presented to an end-to-end feature extraction with a classification deep network. This has the potential to improve classification performance further as both the classification and feature filters will be learned for each dataset, and opens the door for integrating our method with other deep learning techniques like transfer learning or meta learning. Our method focused on the limited data scenario, but recently there have been larger labeled satellite imagery datasets made available with RGB and Multi-spectral data [171]. An investigation of the ability of FST networks to generalize between geographically distant locations, as well as their ability to transfer networks trained on RGB to more bands, remains to be evaluated.

7.1.3 ModeGAN with Adversarial distributions

Adversarial examples are commonly understood to be a threat to Deep Networks, but they can also be harnessed as additional training examples to improve image recognition models [143, 172]. In general having more training data prevents overfitting in deep networks, but when the additional data comes from different distributions, the performance of the model may be greatly reduced as a consequence of added robustness. Some methods source additional training data from other domains or generate synthetic training data, and then apply techniques that lessen the impact of the domain shift in the data on the model [173]. Since adversarial examples come from a different data distribution, special techniques such as auxiliary network structures have been developed to lessen this domain shift [172]. The Mode GAN developed in Chapter 6 has the ability to emphasize the modes of all

distributions it is trained on which resulted in fidelity in its generator. Adversarial examples could be searched for in the proximity of these modes to find the "most likely" adversaries a network might encounter. The impact of this kind of GAN guided adversarial training has yet to be studied.

Bibliography

[1] tensorflow/gan, June 2020. original-date: 2019-04-29T16:24:10Z.

[2] Yi Luo and Nima Mesgarani. Conv-tasnet: Surpassing ideal time–frequency magnitude masking for speech separation. *IEEE/ACM transactions on audio, speech, and language processing*, 27(8):1256–1266, 2019.

[3] Kevin Wilson, Michael Chinen, Jeremy Thorpe, Brian Patton, John Hershey, A. Rif Saurous, Jan Skoglund, and F. Richard Lyon. Exploring tradeoffs in models for low-latency speech enhancement. In *Proc. IWAENC*, September 2018.

[4] Scott Wisdom, John R Hershey, Kevin Wilson, Jeremy Thorpe, Michael Chinen, Brian Patton, and Rif A Saurous. Differentiable consistency constraints for improved deep speech enhancement. *Proc. ICASSP*, May 2019.

[5] Alexander LeNail. Nn-svg: Publication-ready neural network architecture schematics. *Journal of Open Source Software*, 4(33):747, 2019.

[6] Richard Lyon. *Human and Machine Hearing: Extracting Meaning from Sound*. Cambridge University Press, 2017.

[7] Maria-Irina Nicolae, Mathieu Sinn, Minh Ngoc Tran, Beat Buesser, Ambrish Rawat, Martin Wistuba, Valentina Zantedeschi, Nathalie Baracaldo, Bryant Chen, Heiko Ludwig, Ian Molloy, and Ben Edwards. Adversarial robustness toolbox v1.2.0. *CoRR*, 1807.01069, 2018.

[8] Takeru Miyato and Masanori Koyama. cGANs with projection discriminator. In *International Conference on Learning Representations*, 2018.

[9] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier GANs. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2642–2651, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

[10] Mario Lucic, Michael Tschannen, Marvin Ritter, Xiaohua Zhai, Olivier Bachem, and Sylvain Gelly. High-fidelity image generation with fewer labels. *arXiv preprint arXiv:1903.02271*, 2019.

[11] Xugang Lu, Yu Tsao, Shigeki Matsuda, and Chiori Hori. Speech enhancement based on deep denoising autoencoder. In *Proc. Interspeech*, pages 436–440, March 2013.

[12] Felix J. Weninger, John R. Hershey, Jonathan Le Roux, and Björn Schuller. Discriminatively trained recurrent neural networks for single-channel speech separation. In *Proc. GlobalSIP*, December 2014.

[13] Yong Xu, Jun Du, Li-Rong Dai, and Chin-Hui Lee. An experimental study on speech enhancement based on deep neural networks. *IEEE Signal Processing Letters*, 21(1):65–68, 2014.

[14] Hakan Erdogan, John R Hershey, Shinji Watanabe, and Jonathan Le Roux. Phase-sensitive and recognition-boosted speech separation using deep recurrent neural networks. In *Proc. ICASSP*, pages 708–712, April 2015.

[15] Felix Weninger, Hakan Erdogan, Shinji Watanabe, Emmanuel Vincent, Jonathan Le Roux, John R Hershey, and Björn Schuller. Speech enhancement with lstm recurrent neural networks and its application to noise-robust asr. In *International Conference on Latent Variable Analysis and Signal Separation*, pages 91–99. Springer, 2015.

[16] John R Hershey, Zhuo Chen, Jonathan Le Roux, and Shinji Watanabe. Deep clustering: Discriminative embeddings for segmentation and separation. In *Proc. ICASSP*, pages 31–35, March 2016.

[17] Yusuf Isik, Jonathan Le Roux, Zhuo Chen, Shinji Watanabe, and John R Hershey. Single-channel multi-speaker separation using deep clustering. In *Proc. Interspeech*, September 2016.

[18] Dong Yu, Morten Kolbæk, Zheng-Hua Tan, and Jesper Jensen. Permutation invariant training of deep models for speaker-independent multi-talker speech separation. In *Proc. ICASSP*, pages 241–245, March 2017.

[19] Morten Kolbæk, Dong Yu, Zheng-Hua Tan, Jesper Jensen, Morten Kolbaek, Dong Yu, Zheng-Hua Tan, and Jesper Jensen. Multitalker speech separation with utterance-level permutation invariant training of deep recurrent neural networks. *IEEE/ACM Transactions on Audio, Speech and Language Processing*, 25(10):1901–1913, 2017.

[20] DeLiang Wang and Jitong Chen. Supervised Speech Separation Based on Deep Learning: An Overview. In *arXiv preprint arXiv:1708.07524*, 2017.

[21] Zhong-Qiu Wang, Jonathan Le Roux, and John R Hershey. Alternative objective functions for deep clustering. In *Proc. ICASSP*, pages 321–325, April 2018.

[22] Po-Sen Huang, Minje Kim, Mark Hasegawa-Johnson, and Paris Smaragdis. Deep learning for monaural speech separation. In *Proc. ICASSP*, pages 1562–1566, May 2014.

[23] Zhong-Qiu Wang, Jonathan Le Roux, and John R Hershey. Multi-channel deep clustering: Discriminative spectral and spatial embeddings for speaker-independent speech separation. *Proc. ICASSP*, 2018.

[24] Andreas Jansson, Eric Humphrey, Nicola Montecchio, Rachel Bittner, Aparna Kumar, and Tillman Weyde. Singing voice separation with deep U-Net convolutional networks. *Proc. ISMIR*, October 2017.

[25] Y Cem Subakan and Paris Smaragdis. Generative adversarial source separation. In *Proc. ICASSP*, pages 26–30, April 2018.

[26] E. M. Grais and M. D. Plumbley. Combining fully convolutional and recurrent neural networks for single channel audio source separation. In *Proc. AES*, May 2018.

[27] Q. Kong, Y. Xu, W. Wang, and M. D. Plumbley. A joint separation-classification model for sound event detection of weakly labelled data. In *Proc. ICASSP*, pages 321–325, April 2018.

[28] Naoya Takahashi, Nabarun Goswami, and Yuki Mitsufuji. MMDenseLSTM: An efficient combination of convolutional and recurrent neural networks for audio source separation. *arXiv preprint arXiv:1805.02410*, 2018.

[29] Emad M Grais and Mark D Plumbley. Single channel audio source separation using convolutional denoising autoencoders. In *Proc. GlobalSIP*, pages 1265–1269, November 2017.

[30] Pritish Chandna, Marius Miron, Jordi Janer, and Emilia Gómez. Monoaural audio source separation using deep convolutional neural networks. In *Proc. LVA/ICA*, pages 258–266, February 2017.

[31] Yi Luo, Zhuo Chen, and Nima Mesgarani. Speaker-independent speech separation with deep attractor network. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(4):787–796, 2018.

[32] Yi Luo, Zhuo Chen, John R Hershey, Jonathan Le Roux, and Nima Mesgarani. Deep clustering and conventional networks for music separation: Stronger together. In *Proc. ICASSP*, pages 61–65, March 2017.

[33] E. Vincent, J. Barker, S. Watanabe, J. Le Roux, F. Nesta, and M. Matassoni. The second 'CHiME' speech separation and recognition challenge: Datasets, tasks and baselines. In *Proc. ICASSP*, pages 126–130, May 2013.

[34] Pro Sound Effects Library, 2018. Available from http://www.prosoundeffects.com, accessed: 2018-06-01.

[35] Martín Abadi et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[36] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[37] J Le Roux, S Wisdom, H Erdogan, and JR Hershey. SDR – half-baked or well done? *Proc. ICASSP*, May 2019.

[38] Jonathan Le Roux, Nobutaka Ono, and Shigeki Sagayama. Explicit consistency constraints for STFT spectrograms and their application to phase reconstruction. In *Proc. SAPA*, pages 23–28, September 2008.

[39] Hongyi Zhang, Yann N Dauphin, and Tengyu Ma. Fixup initialization: Residual learning without normalization. *arXiv preprint arXiv:1901.09321*, 2019.

[40] Alexander F. H. Goetz. Three decades of hyperspectral remote sensing of the Earth: A personal view. *Remote Sensing of Environment*, 113:S5–S16, September 2009.

[41] N. Skytland. What is NASA doing with Big Data today? *open.NASA.gov*, 2012.

[42] L. He, J. Li, C. Liu, and S. Li. Recent advances on spectralâĂŞspatial hyperspectral image classification: An overview and new guidelines. *IEEE Transactions on Geoscience and Remote Sensing*, 56(3):1579–1597, March 2018.

[43] Jacopo Acquarelli, Elena Marchiori, Lutgarde M. C. Buydens, Thanh Tran, and Twan van Laarhoven. Spectral-Spatial Classification of Hyperspectral Images: Three Tricks and a New Learning Setting. *Remote Sensing*, 10(7):1156, July 2018.

[44] Y. Chen, H. Jiang, C. Li, X. Jia, and P. Ghamisi. Deep Feature Extraction and Classification of Hyperspectral Images Based on Convolutional Neural Networks. *IEEE Transactions on Geoscience and Remote Sensing*, 54(10):6232–6251, October 2016.

[45] Fei Deng, Shengliang Pu, Xuehong Chen, Yusheng Shi, Ting Yuan, and Shengyan Pu. Hyperspectral Image Classification with Capsule Network Using Limited Training Samples. *Sensors (Basel, Switzerland)*, 18(9), September 2018.

[46] Xiaorui Ma, Jie Geng, and Hongyu Wang. Hyperspectral image classification via contextual deep learning. *EURASIP Journal on Image and Video Processing*, 2015(1):20, July 2015.

[47] Heming Liang and Qi Li. Hyperspectral Imagery Classification Using Sparse Representations of Convolutional Neural Network Features. *Remote Sensing*, 8(2):99, January 2016.

[48] T. C. Bau, S. Sarkar, and G. Healy. Hyperspectral region classification using a three-dimensional gabor filterbank. *IEEE Transactions on Geoscience and Remote Sensing*, 48(9):3457–3464, 2010.

[49] S. Jia, L. Shen, and Q. Li. Gabor Feature-Based Collaborative Representation for Hyperspectral Imagery Classification. *IEEE Transactions on Geoscience and Remote Sensing*, 53(2):1118–1129, February 2015.

[50] L. He, J. Li, A. Plaza, and Y. Li. Discriminative Low-Rank Gabor Filtering for SpectralâĂŞSpatial Hyperspectral Image Classification. *IEEE Transactions on Geoscience and Remote Sensing*, 55(3):1381–1395, March 2017.

[51] Xiangyong Cao, Lin Xu, Deyu Meng, Qian Zhao, and Zongben Xu. Integration of 3-dimensional discrete wavelet transform and Markov random field for hyperspectral image classification. *Neurocomputing*, 226:90–100, February 2017.

[52] G. Franchi and J. Angulo. A deep spatial/spectral descriptor of hyperspectral texture using scattering transform. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 3568–3572, September 2016.

[53] S. Jia, J. Hu, Y. Xie, L. Shen, X. Jia, and Q. Li. Gabor Cube Selection Based Multitask Joint Sparse Representation for Hyperspectral Image Classification. *IEEE Transactions on Geoscience and Remote Sensing*, 54(6):3174–3187, June 2016.

[54] L. Shen and S. Jia. Three-Dimensional Gabor Wavelets for Pixel-Based Hyperspectral Imagery Classification. *IEEE Transactions on Geoscience and Remote Sensing*, 49(12):5039–5046, December 2011.

[55] Y. Qian, M. Ye, and J. Zhou. Hyperspectral Image Classification Based on Structured Sparse Logistic Regression and Three-Dimensional Wavelet Texture Features. *IEEE Transactions on Geoscience and Remote Sensing*, 51(4):2276–2291, April 2013.

[56] Yuan Yan Tang, Yang Lu, and Haoliang Yuan. Hyperspectral image classification based on three-dimensional scattering wavelet transform. *IEEE Transactions on Geoscience and Remote sensing*, 53(5):2467–2480, 2015.

[57] Wojciech Czaja and Weilin Li. Analysis of time-frequency scattering transforms. *Applied and Computational Harmonic Analysis*, 2017.

[58] Wojciech Czaja and Weilin Li. Rotationally invariant time-frequency scattering transforms. *Journal of Fourier Analysis and Applications*, To Appear, 2020.

[59] Erchan Aptoula, Murat Can Ozdemir, and Berrin Yanikoglu. Deep Learning With Attribute Profiles for Hyperspectral Image Classification. *IEEE Geoscience and Remote Sensing Letters*, 13(12):1970–1974, December 2016.

[60] Weiwei Song, Shutao Li, Leyuan Fang, and Ting Lu. Hyperspectral Image Classification With Deep Feature Fusion Network. *IEEE Transactions on Geoscience and Remote Sensing*, 56(6):3173–3184, June 2018.

[61] Stéphane Mallat. Group invariant scattering. *Communications on Pure and Applied Mathematics*, 65(10):1331–1398, 2012.

[62] Yanhui Guo, Siming Han, Han Cao, Yu Zhang, and Qian Wang. Guided filter based Deep Recurrent Neural Networks for Hyperspectral Image Classification. *Procedia Computer Science*, 129:219–223, January 2018.

[63] Wei Hu, Yangyu Huang, Li Wei, Fan Zhang, and Hengchao Li. Deep Convolutional Neural Networks for Hyperspectral Image Classification, 2015.

[64] P. Zhong, Z. Gong, S. Li, and C. SchÃűnlieb. Learning to Diversify Deep Belief Networks for Hyperspectral Image Classification. *IEEE Transactions on Geoscience and Remote Sensing*, 55(6):3516–3530, June 2017.

[65] Pai-Hui Hsu. Evaluating the Initialization Methods of Wavelet Networks for Hyperspectral Image Classification. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 41B7:83–89, June 2016.

[66] Hongmin Gao, Shuo Lin, Yao Yang, Chenming Li, and Mingxiang Yang. Convolution Neural Network Based on Two-Dimensional Spectrum for Hyperspectral Image Classification, 2018.

[67] Wojciech Czaja, Weilin Li, and Ilya Kavalerov. Scattering transforms and classification of hyperspectral images. In David W. Messinger and Miguel Velez-Reyes, editors, *Algorithms and Technologies for Multispectral, Hyperspectral, and Ultraspectral Imagery XXIV*, page 17, Orlando, United States, May 2018. SPIE.

[68] Pai-Hui Hsu and Hsiu-Han Yang. Hyperspectral image classification using wavelet networks. In *2007 IEEE International Geoscience and Remote Sensing Symposium*, pages 1767–1770, July 2007.

[69] Hyungtae Lee and Heesung Kwon. Going Deeper with Contextual CNN for Hyperspectral Image Classification. *IEEE Transactions on Image Processing*, 26(10):4843–4855, October 2017.

[70] A. Mughees, A. Ali, and L. Tao. Hyperspectral image classification via shape-adaptive deep learning. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 375–379, September 2017.

[71] P S Ashitha, V. Sowmya, and K. P. Soman. Classification of hyperspectral images using scattering transform. *International Journal of Scientific & Engineering Research*, 5(7):5, 2014.

[72] X. Yang, Y. Ye, X. Li, R. Y. K. Lau, X. Zhang, and X. Huang. Hyperspectral Image Classification With Deep Learning Models. *IEEE Transactions on Geoscience and Remote Sensing*, 56(9):5408–5423, September 2018.

[73] Sara Sabour, Nicholas Frosst, and Geoffrey E. Hinton. Dynamic Routing Between Capsules. *arXiv:1710.09829 [cs]*, October 2017. arXiv: 1710.09829.

[74] Nikhila Haridas, V. Sowmya, and K. P. Soman. Comparative Analysis of Scattering and Random Features in Hyperspectral Image Classification. *Procedia Computer Science*, 58:307–314, January 2015.

[75] L. He, Y. Li, X. Li, and W. Wu. SpectralâĂŞspatial classification of hyperspectral images via spatial translation-invariant wavelet-based sparse representation. *IEEE Transactions on Geoscience and Remote Sensing*, 53(5):2696–2712, 2015.

[76] Joan Bruna and Stéphane Mallat. Invariant scattering convolution networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1872–1886, 2013.

[77] C. L. Bennett, M. R. Carter, D. J. Fields, and F. D. Lee. Infrared hyperspectral imaging results from vapor plume experiments. *Proc. SPIE*, 2480(19):435–444, 1995.

[78] E. Neil. Lewis, Patrick J. Treado, Robert C. Reeder, Gloria M. Story, Anthony E. Dowrey, Curtis. Marcott, and Ira W. Levin. Fourier transform spectroscopic imaging using an infrared focal-plane array detector. *Analytical Chemistry*, 67(19):3377–3381, 1995.

[79] Karlheinz Gröchenig. *Foundations of time-frequency analysis*. Springer Science & Business Media, 2013.

[80] Joakim Andén and Stéphane Mallat. Deep scattering spectrum. *IEEE Transactions on Signal Processing*, 62(16):4114–4128, 2014.

[81] Scattering transforms in python. `https://github.com/ilyakava/pyfst`. Accessed: 2020-02-06.

[82] Marion F. Baumgardner, Larry L. Biehl, and David A. Landgrebe. 220 band aviris hyperspectral image data set: June 12, 1992 indian pine test site 3, Sep 2015.

[83] F. Dell' Acqua, P. Gamba, and A. Ferrari. Exploiting spectral and spatial information for classifying hyperspectral data in urban areas. In *IGARSS 2003. 2003 IEEE International Geoscience and Remote Sensing Symposium. Proceedings (IEEE Cat. No.03CH37477)*, volume 1, pages 464–466 vol.1, July 2003.

[84] A. L. Neuenschwander, M. M. Crawford, and M. J. Provancha. Mapping of coastal wetlands via hyperspectral aviris data. In *Geoscience and Remote Sensing Symposium Proceedings, 1998. IGARSS '98. 1998 IEEE International*, volume 1, pages 189–191 vol.1, Jul 1998.

[85] A. L. Neuenschwander, M. M. Crawford, and S. Ringrose. Monitoring of seasonal flooding in the okavango delta using eo-1 data. In *IEEE International Geoscience and Remote Sensing Symposium*, volume 6, pages 3124–3126 vol.6, 2002.

[86] Hyperspectral remote sensing scenes. `http://www.ehu.eus/ccwintco/index.php/Hyperspectral_Remote_Sensing_Scenes`. Accessed: 01-01-2018.

[87] Demo dffn. `https://github.com/weiweisong415/Demo_DFFN`. Accessed: 2010-02-06.

[88] Jie Liang, Jun Zhou, Yuntao Qian, Lian Wen, Xiao Bai, and Yongsheng Gao. On the Sampling Strategy for Evaluation of Spectral-Spatial Methods in Hyperspectral Image Classification. *IEEE Transactions on Geoscience and Remote Sensing*, 55(2):862–880, February 2017.

[89] Nicholas Carlini and David Wagner. Audio Adversarial Examples: Targeted Attacks on Speech-to-Text. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 1–7, May 2018.

[90] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal Adversarial Perturbations. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 86–94, Honolulu, HI, July 2017. IEEE.

[91] Hokuto Hirano and Kazuhiro Takemoto. Simple iterative method for generating targeted universal adversarial perturbations. *Proceedings of 25th International Symposium on Artificial Life and Robotics (AROB 25th 2020)*, pages 426–430, November 2019. arXiv: 1911.06502.

[92] Sajjad Abdoli, Luiz G. Hafemann, Jerome Rony, Ismail Ben Ayed, Patrick Cardinal, and Alessandro L. Koerich. Universal Adversarial Audio Perturbations. *arXiv:1908.03173 [cs, eess, stat]*, September 2020. arXiv: 1908.03173.

[93] Yao Qin, Nicholas Carlini, Garrison Cottrell, Ian Goodfellow, and Colin Raffel. Imperceptible, Robust, and Targeted Adversarial Examples for Automatic Speech Recognition. In *International Conference on Machine Learning*, pages 5231–5240. PMLR, May 2019. ISSN: 2640-3498.

[94] Lea Schonherr, Thorsten Eisenhofer, Steffen Zeiler, Thorsten Holz, and Dorothea Kolossa. Imperio: Robust Over-the-Air Adversarial Examples for Automatic Speech Recognition Systems. *arXiv:1908.01551 [cs, eess]*, April 2020. arXiv: 1908.01551.

[95] Juncheng Li, Shuhui Qu, Xinjian Li, Joseph Szurley, J. Zico Kolter, and Florian Metze. Adversarial Music: Real world Audio Adversary against Wake-word Detection System. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d\textquotesingle AlchÃl-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 11908–11918. Curran Associates, Inc., 2019.

[96] Nai Ding, Aniruddh D. Patel, Lin Chen, Henry Butler, Cheng Luo, and David Poeppel. Temporal modulations in speech and music. *Neuroscience and Biobehavioral Reviews*, 81(Pt B):181–187, October 2017.

[97] Nima Mesgarani, Stephen V. David, Jonathan B. Fritz, and Shihab A. Shamma. Mechanisms of noise robust representation of speech in primary auditory cortex. *Proceedings of the National Academy of Sciences*, 111(18):6792–6797, May 2014. Publisher: National Academy of Sciences Section: Biological Sciences.

[98] Alexander J. E. Kell and Josh H. McDermott. Invariance to background noise as a signature of non-primary auditory cortex. *Nature Communications*, 10(1):3958, September 2019. Number: 1 Publisher: Nature Publishing Group.

[99] Bahar Khalighinejad, Jose L. Herrero, Ashesh D. Mehta, and Nima Mesgarani. Adaptation of the human auditory cortex to changing background noise. *Nature Communications*, 10(1):2509, June 2019. Number: 1 Publisher: Nature Publishing Group.

[100] Taishih Chi, Powen Ru, and Shihab A. Shamma. Multiresolution spectrotemporal analysis of complex sounds. *The Journal of the Acoustical Society of America*, 118(2):887–906, August 2005.

[101] Powen Ru. Multiscale Multirate Spectro-Temporal Auditory Model, 2001.

[102] Nima Mesgarani and Shihab Shamma. Speech processing with a cortical representation of audio. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5872–5875, May 2011. ISSN: 2379-190X.

[103] Nima Mesgarani, Stephen V. David, Jonathan B. Fritz, and Shihab A. Shamma. Phoneme representation and classification in primary auditory cortex. *The Journal of the Acoustical Society of America*, 123(2):899–909, February 2008. Publisher: Acoustical Society of America.

[104] S Shamma and Huib Versnel. Ripple analysis in ferret primary auditory cortex iii. prediction of unit responses to arbitrary spectral profiles. Technical report, University of Maryland, College Park, 1995.

[105] N. Mesgarani, M. Slaney, and S.A. Shamma. Discrimination of speech from nonspeech based on multiscale spectro-temporal Modulations. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(3):920–930, May 2006. Conference Name: IEEE Transactions on Audio, Speech, and Language Processing.

[106] Joakim Anden, Vincent Lostanlen, and Stephane Mallat. Joint time-frequency scattering for audio classification. In *2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6, September 2015. ISSN: 2378-928X.

[107] Nima Mesgarani and Shihab Shamma. Denoising in the Domain of Spectrotemporal Modulations. *EURASIP Journal on Audio, Speech, and Music Processing*, 2007(1):042357, November 2007.

[108] Jinxi Guo, Kenichi Kumatani, Ming Sun, Minhua Wu, Anirudh Raju, Nikko StrÃűm, and Arindam Mandal. Time-Delayed Bottleneck Highway Networks Using a DFT Feature for Keyword Spotting. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5489–5493, April 2018. ISSN: 2379-190X.

[109] Sankaran Panchapagesan, Ming Sun, Aparna Khare, Spyros Matsoukas, Arindam Mandal, Bjorn Hoffmeister, and Shiv Vitaladevuni. Multi-task learning and weighted cross-entropy for dnn-based keyword spotting. In *Interspeech 2016*, pages 760–764, 2016.

[110] Ming Sun, Varun Nagaraja, BjÃűrn Hoffmeister, and Shiv Vitaladevuni. Model Shrinking for Embedded Keyword Spotting. In *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, pages 369–374, December 2015.

[111] testandroidtests. Library Catalog: github.com.

[112] jhautry/echo-dot. Library Catalog: github.com.

[113] Alexa Dataset.

[114] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: an asr corpus based on public domain audio books. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 5206–5210. IEEE, 2015.

[115] Rubber Band Audio Time Stretcher Library.

[116] Robin Scheibler, Eric Bezzam, and Ivan DokmaniÄĞ. Pyroomacoustics: A Python package for audio room simulations and array processing algorithms. *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 351–355, April 2018. arXiv: 1710.04196.

[117] Picovoice/wake-word-benchmark, September 2020. original-date: 2018-04-25T00:12:46Z.

[118] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *CoRR*, abs/1412.6572, 2015.

[119] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.

[120] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2574–2582, June 2016. ISSN: 1063-6919.

[121] Nicholas Carlini and David Wagner. Towards Evaluating the Robustness of Neural Networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 39–57, May 2017. ISSN: 2375-1207.

[122] N. Mesgarani, C. Cheung, K. Johnson, and E. F. Chang. Phonetic Feature Encoding in Human Superior Temporal Gyrus. *Science*, 343(6174):1006–1010, February 2014.

[123] Joseph Szurley and J. Zico Kolter. Perceptual based adversarial audio attacks. *CoRR*, abs/1906.06355, 2019.

[124] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Proceedings of the 27th International Conference on Neural Information Processing Systems-Volume 2*, pages 2672–2680. MIT Press, 2014.

[125] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *International Conference on Learning Representations*, 2019.

[126] Mehdi Mirza and Simon Osindero. Conditional Generative Adversarial Nets. *arXiv:1411.1784 [cs, stat]*, November 2014. arXiv: 1411.1784.

[127] Jae Hyun Lim and Jong Chul Ye. Geometric GAN. *arXiv:1705.02894 [cond-mat, stat]*, May 2017. arXiv: 1705.02894.

[128] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in neural information processing systems*, pages 2234–2242, 2016.

[129] Hao-Wen Dong and Yi-Hsuan Yang. Towards a Deeper Understanding of Adversarial Losses. *arXiv:1901.08753 [cs, stat]*, January 2019. arXiv: 1901.08753.

[130] Martin Arjovsky and Leon Bottou. Towards Principled Methods for Training Generative Adversarial Networks. In *International Conference on Learning Representations*, 2017.

[131] Youssef Mroueh, Tom Sercu, and Vaibhava Goel. McGan: Mean and covariance feature matching GAN. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2527–2535, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

[132] Chun-Liang Li, Wei-Cheng Chang, Yu Cheng, Yiming Yang, and Barnabás Póczos. Mmd gan: Towards deeper understanding of moment matching network. In *Advances in Neural Information Processing Systems*, pages 2203–2213, 2017.

[133] Dustin Tran, Rajesh Ranganath, and David Blei. Hierarchical implicit models and likelihood-free variational inference. In *Advances in Neural Information Processing Systems*, pages 5523–5533, 2017.

[134] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*, 2018.

[135] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 7354–7363, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

[136] Tu Nguyen, Trung Le, Hung Vu, and Dinh Phung. Dual discriminator generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2670–2680, 2017.

[137] Ishan Durugkar, Ian Gemp, and Sridhar Mahadevan. Generative Multi-Adversarial Networks. In *International Conference on Learning Representations*, 2017.

[138] Yao Ni, Dandan Song, Xi Zhang, Hao Wu, and Lejian Liao. Cagan: Consistent adversarial training enhanced gans. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 2588–2594. International Joint Conferences on Artificial Intelligence Organization, 7 2018.

[139] Jost Tobias Springenberg. Unsupervised and Semi-supervised Learning with Categorical Generative Adversarial Networks. *arXiv:1511.06390 [cs, stat]*, November 2015. arXiv: 1511.06390.

[140] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Alex Lamb, Martin Arjovsky, Olivier Mastropietro, and Aaron Courville. Adversarially Learned Inference. In *International Conference on Learning Representations*, 2017.

[141] LI Chongxuan, Taufik Xu, Jun Zhu, and Bo Zhang. Triple generative adversarial nets. In *Advances in neural information processing systems*, pages 4088–4098, 2017.

[142] Augustus Odena. Semi-supervised learning with generative adversarial networks. *arXiv preprint arXiv:1606.01583*, 2016.

[143] Zihang Dai, Zhilin Yang, Fan Yang, William W Cohen, and Ruslan R Salakhutdinov. Good Semi-supervised Learning That Requires a Bad GAN. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6510–6520. Curran Associates, Inc., 2017.

[144] Jinhao Dong and Tong Lin. Margingan: Adversarial training in semi-supervised learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 10440–10449. Curran Associates, Inc., 2019.

[145] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Advances in neural information processing systems*, pages 1195–1204, 2017.

[146] Ilya Kavalerov. ilyakava/MarginGAN, August 2020. original-date: 2020-08-26T14:59:06Z.

[147] Zhijie Deng, Hao Zhang, Xiaodan Liang, Luona Yang, Shizhen Xu, Jun Zhu, and Eric P Xing. Structured generative adversarial networks. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 3899–3909. Curran Associates, Inc., 2017.

[148] Zhe Gan, Liqun Chen, Weiyao Wang, Yuchen Pu, Yizhe Zhang, Hao Liu, Chunyuan Li, and Lawrence Carin. Triangle generative adversarial networks. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5247–5256. Curran Associates, Inc., 2017.

[149] Kumar Sricharan, Raja Bala, Matthew Shreve, Hui Ding, Kumar Saketh, and Jin Sun. Semi-supervised conditional gans. *arXiv preprint arXiv:1708.05789*, 2017.

[150] Koby Crammer and Yoram Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of machine learning research*, 2(Dec):265–292, 2001.

[151] Martin Arjovsky, Soumith Chintala, and Leon Bottou. Wasserstein GAN. *arXiv:1701.07875 [cs, stat]*, December 2017. arXiv: 1701.07875.

[152] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in neural information processing systems*, pages 5767–5777, 2017.

[153] Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur. A Learned Representation For Artistic Style. In *International Conference on Learning Representations*, 2017.

[154] Harm De Vries, Florian Strub, Jérémie Mary, Hugo Larochelle, Olivier Pietquin, and Aaron C Courville. Modulating early visual processing by language. In *Advances in Neural Information Processing Systems*, pages 6594–6604, 2017.

[155] bioinf-jku/TTUR, November 2019. original-date: 2017-06-26T13:32:12Z.

[156] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[157] Mario Lucic, Karol Kurach, Marcin Michalski, Sylvain Gelly, and Olivier Bousquet. Are GANs Created Equal? A Large-Scale Study. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 700–709. Curran Associates, Inc., 2018.

[158] Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V Le. Self-training with noisy student improves imagenet classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10687–10698, 2020.

[159] TensorFlow Datasets, a collection of ready-to-use datasets. `https://www.tensorflow.org/datasets`.

[160] Martin Arjovsky and Leon Bottou. Towards Principled Methods for Training Generative Adversarial Networks. *arXiv:1701.04862 [cs, stat]*, January 2017. arXiv: 1701.04862.

[161] Zhiming Zhou, Han Cai, Shu Rong, Yuxuan Song, Kan Ren, Weinan Zhang, Jun Wang, and Yong Yu. Activation maximization generative adversarial nets. In *International Conference on Learning Representations*, 2018.

[162] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved Techniques for Training GANs. *arXiv:1606.03498 [cs]*, June 2016. arXiv: 1606.03498.

[163] David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin A Raffel. Mixmatch: A holistic approach to semi-supervised learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 5050–5060. Curran Associates, Inc., 2019.

[164] Kihyuk Sohn, David Berthelot, Chun-Liang Li, Zizhao Zhang, Nicholas Carlini, Ekin D. Cubuk, Alex Kurakin, Han Zhang, and Colin Raffel. FixMatch: Simplifying Semi-Supervised Learning with Consistency and Confidence. *arXiv:2001.07685 [cs, stat]*, January 2020. arXiv: 2001.07685.

[165] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral Normalization for Generative Adversarial Networks. *arXiv:1802.05957 [cs, stat]*, February 2018. arXiv: 1802.05957.

[166] H Vincent Poor. *An introduction to signal detection and estimation*. Springer Science & Business Media, 2013.

[167] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[168] Shane Barratt and Rishi Sharma. A Note on the Inception Score. *arXiv:1801.01973 [cs, stat]*, June 2018. arXiv: 1801.01973.

[169] Theodore Bluche, Mael Primet, and Thibault Gisselbrecht. Small-Footprint Open-Vocabulary Keyword Spotting with Quantized LSTM Networks. *arXiv:2002.10851 [cs]*, February 2020. arXiv: 2002.10851.

[170] P. Warden and D. Situnayake. *TinyML: Machine Learning with TensorFlow Lite on Arduino and Ultra-Low-Power Microcontrollers*. O'Reilly Media, Incorporated, 2019.

[171] Gencer Sumbul, Marcela Charfuelan, Begüm Demir, and Volker Markl. Bigearthnet: A large-scale benchmark archive for remote sensing image understanding. In *IGARSS 2019-2019 IEEE International Geoscience and Remote Sensing Symposium*, pages 5901–5904. IEEE, 2019.

[172] Cihang Xie, Mingxing Tan, Boqing Gong, Jiang Wang, Alan L. Yuille, and Quoc V. Le. Adversarial examples improve image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[173] Swami Sankaranarayanan, Yogesh Balaji, Carlos D. Castillo, and Rama Chellappa. Generate to adapt: Aligning domains using generative adversarial networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.