Techniques for Cough Sound Analysis

Elliot Saba

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2018

Reading Committee:

Shwetak N. Patel, Chair

Les Atlas

Jon Froehlich

Program Authorized to Offer Degree:
Electrical Engineering

University of Washington

Abstract

Techniques for Cough Sound Analysis

Elliot Saba

Chair of the Supervisory Committee:
Professor Shwetak N. Patel
Department of Electrical Engineering

Coughing is a common symptom of pulmonary ailment and serves as a valuable measure when quantifying pulmonary health. This dissertation contains the results of the research and development of a set of techniques to enable researchers to investigate pulmonary health through cough sounds. A variety of signal processing and machine learning approaches are included, each with various performance and usability tradeoffs. We propose a novel algorithm that makes use of the best of the traditional signal processing approaches, combined with the recent advances in deep learning to provide new cough detection and classification results previously unattainable, especially when considered in the context of model runtime performance. We detail the construction a classifier for tuberculosis coughs, and develop a new tool to deal with bifurcated datasets we dub a discriminative adversarial network.

TABLE OF CONTENTS

Page

List of Figures iii

List of Tables vi

Chapter 1: Introduction 1
1.1 Overview and Motivation 1
1.2 Thesis Statement and Research Contributions 3

Chapter 2: Background and Related Work 5
2.1 Cough Recordings as a Clinical Tool 5
2.2 Overview of Cough Analysis 6
2.3 Related Cough Signal Processing 8
2.4 Deep Learning 16

Chapter 3: Techniques for Cough Sound Analysis 25
3.1 Dataset 25
3.2 Model Design Criteria 27
3.3 Dataset Loading 30
3.4 Feature Preprocessing 33
3.5 Detection Algorithm 35
3.6 Cough Classification 51

Chapter 4: Discriminative Adversarial Networks 55
4.1 Dataset Imbalance 55
4.2 Applying Adversarial Networks to Dataset Imbalance 58
4.3 A Synthetic Example 65
4.4 Expanding DANs to other problems 68

Chapter 5: Conclusion 70
5.1 Limitations of current work 73
5.2 Future Work 74

Bibliography 81

LIST OF FIGURES

Figure Number Page

2.1 From Korpás et al. [37], showing glottal position, recorded sound, airflow and pressure within the esophagus for two example coughs, segmented into three time regions. 6

2.2 The variation of sound recordings of the expulsive segment of a cough, from Korpás et al. [37] 7

2.3 Abeyratne et al. display the variation of coughs from patients with different pulmonary ailments [1]. 12

2.4 Diagram of a multilayer perceptron, graphically depicting the flow of information from an input layer of 5 neurons (x_i) to an intermediate layer of 3 neurons, to an output layer of two neurons (y_i). Each connecting line conains an independent weight that multiplies the value of the source node on the left, and is summed together with all other lines entering the destination node on the right. 19

2.5 Diagram of a convolutional neural network, graphically depicting a convolution between a 3-channel input tensor and a convolutional kernel in red. The convolutional kernel is shifted over the domain of the input tensor, calculating a weighted sum at each position and generating a single output pixel per shifted location in the input. The generated output plane of pixels is shown in blue. 22

3.1 Diagram of the feature calculation process. Cough events are annotated within a large audio file, windows of temporal data are taken, split into overlapping frames, then each frame is passed through a feature calculation routine. The "random cough" data selector chooses windows that are guaranteed to contain cough annotations, whereas the "random non-cough" data selector chooses windows at random. 31

3.2 Examples of cough signals visualized using the gammatone filterbank (GTFB) spectro-temporal decomposition. Lighter colors indicate higher signal energy within a time-frequency bin. 34

3.3 Frequency responses of 24-length gammatone filterbank outputs 36

3.4 Receiver Operating Characteristic curves for all baseline methods. In top left; all methods are compared, in the other three plots the same methods are separated out by learning method. Color indicates learning method, line style indicates featurespace transformation. 38

3.5 Receiver Operating Characteristic curve for a multilayer perceptron classifier paired with the MFCC, GTFB and LSP feature transformations. The best baseline method is plotted as well, for comparison. 40

3.6 Detection performance for the convolutional model operating upon GTFB features. Left; Instantaneous Receiver Operating Characteristic curve. Right; False alarm curve. 43

3.7 Detection performance for all three convolutional models operating upon GTFB features with varying levels of backpropagation learning allowed to change the GTFB calculation matrix. Right subplot is a zoomed version of Left. 44

3.8 Left; GTFB filter and tweaked version plotted on top of each other. Right; the same, but zoomed into the lower-frequency portions. 45

3.9 Left; GTFB spectrogram of a cough sample. Right; randomized GTFB spectrogram of the same cough sample. 46

3.10 Diagram of a frequency-variant convolutional neural network (FV-CNN), graphically depicting a convolution between a 3-channel input tensor and (for convenience) two different convolutional kernels in red. The convolutional kernels are shifted across the time dimension of the input tensor, with domain and range denoted by the dashed lines. A separate convolutional kernel will be used for the next row of frequency down, and a separate convolutional kernel will be used for the next row after that, continuing until all input data is mapped to an output. The generated output plane of pixels is shown in blue. 47

3.11 Detection performance of frequency-variant models in red, compared against all three non-frequency-variant convolutional models in black. Right subplot is a zoomed version of Left. 48

3.12 Left; False alarm curves for the convolutional model operating upon GTFB features and tweaked GTFB features. Right; False alarm curve after median filtering. 50

3.13 Receiver Operating Characteristic curve for the classification model in four different conditions: Trained on clinical data and tested on clinical data, trained on clinical data and tested on both datasets, trained on both datasets and tested on both datasets, and finally trained on both datasets but tested on only clinical data. 53

4.1 Cough classification architecture with DAN and CNN architectures visualized. Along the bottom lies the convolutional classification network that classifies cough type, with the outputs from the first layer feeding into a multilayer perceptron that classifies dataset source. 58
4.2 Example inputs to a linear classifier from two 62
4.3 Synthetic data examples used to train a convolutional neural network. 20×20 images are generated that contain random patterns with random perturbations. Left; a "fast" oscillating pattern. Right; a "slow" oscillating pattern. . 65
4.4 Left; an uncorrupted "fast" oscillating pattern. Right; a corrupted "fast" oscillating pattern. 66

LIST OF TABLES

Table Number Page

2.1 Summary of previous cough detection work showing the results, whether the method was partially or fully automated, the size of the dataset in number of coughs, and the methods employed. 14

2.2 Summary of previous cough classification work showing what classification task was attempted, the results, whether the method was partially or fully automatic, the size of the dataset in number of coughs, and the methods employed. 15

2.3 Example nonlinearities used in deep neural networks, showcasing their simplicity and easily computed derivatives. 21

3.1 Dataset breakdown by number of coughs captured, participant demographics, and total recording length in hours. 26

3.2 Simplified results for the baseline algorithm 37

3.3 The cough detection model architecture, with learned parameter distribution and runtimes measured per-layer. All runtimes measured on a Raspberry Pi 3 B+, using a batch size of 100, equal to processing a full second of audio at once. 42

4.1 The synthetic DAN evaluation architecture. Very similar to the cough classification model, except smaller. 67

Chapter 1

INTRODUCTION

1.1 Overview and Motivation

Pulmonary ailments account for four of the top ten causes for death worldwide and are especially prevalent in low-income countries [60]. It is estimated that over 1 billion people worldwide suffer from pulmonary disease [25]. Coughing is a symptom of many of these ailments, including (but not limited to) asthma, tuberculosis, cystic fibrosis, lower respiratory infection, chronic obstructive pulmonary disease, and over a hundred others [37]. Symptom tracking is an important part of the health care process at all stages, whether for screening the general public to find new cases, assessing new patients, or tracking long-term cases [43, 30]. Current systems to track pulmonary ailments through cough sounds include patient self-reporting, manual cough counting and analysis, and automated cough frequency trackers [54, 30, 32]. Self-reporting of cough frequency and cough characteristics has been shown to lack the accuracy necessary for usage in clinical situations [7]. Manual cough counting, due to the unpredictable and intermittent nature of coughs, can be a very time-intensive process requiring a dedicated listener to record all cough events over a time duration large enough to gather cough data for diagnostic purposes. Automated cough frequency trackers of varying accuracies for different applications exist [4, 5, 7, 9, 8, 45, 11, 24, 39, 46, 54, 59], each lying as a point within a large trade-off space with dimensions of automation, accuracy, runtime requirements and privacy considerations. A deeper inspection of related cough detection work is given in section 2.3.1.

Current tools for cough analysis focus on cough frequency as the primary metric to be tracked. While this metric is valuable [43, 30, 37], medical professionals hypothesize that as different pulmonary ailments can afflict the pulmonary system and air pathways in different

ways, the acoustic signature of coughs from patients with these ailments may change in both overt and subtle ways [37, 1]. One of the most common classifications is to draw a distinction between "wet" and "dry" coughs. A "wet" cough is one in which a pulmonary ailment has caused a buildup of *sputum*, a mixture of saliva and mucus, within the lungs that is ejected from the lungs during the expulsive phase of a cough. While there has been some preliminary study into the automatic classification of cough types into "wet" vs. "dry" classes [15, 57], there is still much work to be done towards creating a framework for cough sound analysis. We provide a more detailed view of related cough classification work in section 2.3.2.

To date, cough recording research has typically been separated into two distinct approaches: either attempting to classify a pre-segmented cough sound recorded in a clinical setting based upon certain cough characteristics [1, 2, 54, 57, 15], or detecting the presence of coughs in an ambulatory sensing modality such as giving the patient a recorder to carry on their person as they travel throughout the day [4, 5, 7, 9, 8, 45, 11, 24, 39, 46, 54, 59]. In the former, human effort is typically required to first locate and properly segment cough sounds for the cough classification algorithms. In the latter, the recorder may store a variety of signals such as audio data, ECG or other contact and non-contact sensors to be used for cough detection. The recordings are then analyzed by algorithm, human annotators or both, to result in an estimate of number of coughs per hour.

Cough sounds are not expected to become a primary diagnostic tool for pulmonary disease; however the official World Health Organization guidelines for the diagnosis of pulmonary ailments such as childhood pneumonia result in low enough performance that the addition of markers within cough signals has the potential to significantly increase diagnostic performance, especially within the context of the developing world [1]. Building an algorithm for automated cough analysis can therefore impact the medical world along multiple fronts. We list three potentially interesting applications of this technology, which will serve as motivating examples as we explore the tradeoff space in section TODO: X.X.X:

- *Mobile cough counter*: Implemented as a cough detection algorithm on a personal

mobile computation device with a microphone such as a mobile phone or smartwatch, this application would enable ubiquitous, personal pulmonary health tracking through constant cough detection. This application requires efficient algorithms so as to have as little impact on power usage as possible, to lengthen device lifetime, and to coexist with other classification algorithms such as personal digital assistants.

- *Listening station*: Implemented as a cough detection algorithm on a stationary device such as a smart speaker, this application would enable nighttime cough tracking for sleep studies and population cough counting for epidemiological studies where the cough station is located in public settings such as doctor's office waiting rooms. The ability for all computation to be performed locally on the device is critical for user privacy in this application, so that no raw audio is ever stored or transmitted outside of the device.

- *Clinical tool*: Implemented as a cough classification algorithm on a device such as a smartphone, this application would enable cheap screening methodologies for pulmonary ailments. We draw a clear distinction between cough detection algorithms, which typically must run faster than real-time and consume minimal computational resources for power and computational budget reasons, and cough classification algorithms which typically are only run once a cough has been identified, either manually or through a previously run cough detection algorithm. As such, cough classification algorithms do not have the same computational budgets applied to them and can run much slower than real-time.

1.2 Thesis Statement and Research Contributions

The state of current tools is insufficient to support the applications shown above, and as such new tools must be developed. This motivation directly introduces our thesis statement:

Cough sounds contain underutilized pulmonary health information

that can be extracted through novel applications of signal processing and deep learning.

The specific research contributions contained with this thesis are as follows:

- *Algorithms for cough detection*: This thesis work contains a state-of-the-art cough detection algorithm in terms of accuracy while simultaneously achieving excellent runtime performance, enabling new applications on embedded platforms.

- *Algorithms for cough classification*: By applying the same basic algorithm to cough classification instead of detection, we are able to build what is, to our knowledge, the first tuberculosis cough detector. We analyze its accuracy and provide guidelines for future work in extending capabilities for more effective cough sound analysis.

- *Techniques for highly imbalanced datasets*: The datasets collected in support of this dissertation are sub-optimal, and using them naively results in serious deficiencies in the trained models. We introduce novel methods to combat dataset imbalance that builds off of recent advances in the machine learning fields, applying it to our models with great effect.

This dissertation will begin with an overview of relevant cough detection and classification research in sections 2.3.1 and 2.3.2, then introduce necessary machine learning concepts in section 2.4. Introduction of our cough detection and classification algorithm will be given in chapter 3, with special attention paid to our handling of imbalanced datasets in chapter 4. We explicitly address the challenges of insufficient training data through a combination of models with a restricted set of parameters, data augmentation and adversarial networks to counter dataset imbalance. These chapters will analyze the results both in terms of detection/classification accuracy as well as runtime performance, with a final wrap-up in chapter 5.

Chapter 2
BACKGROUND AND RELATED WORK

Pulmonary research is a well-established field that has received a wealth of research into the maladies and ailments that can befall populations worldwide. A small portion of this research has been focused on the diagnostic implications of cough sounds and the systematic detection of them. We will give here a brief overview of background information as well as related work, split into categories based on the nature of the work and how it relates to the proposal at hand.

2.1 Cough Recordings as a Clinical Tool

Coughs, being rapid forced exhalatory maneuvers, have rich temporal and spectral representations which contain a wide variation across varying pulmonary conditions [37], giving rise to the supposition that these sounds may, through analysis, yield information about the pulmonary condition of the patient that coughed. Cough sounds can be segmented into three rough time periods, as visualized in Figure 2.1 [37]. The three time periods are labeled the inspiratory cough phase, the compressive cough phase, and the expulsive cough phase.

As cough sound recordings obtain data primarily within the expulsive cough phase, all cough recording papers restrict themselves to analysis within this temporal phase and do not attempt to segment out inspiratory or compressive cough phases. Within the expulsive cough phase, there is still a broad variation of cough sounds, as shown in Figure 2.2 [37], and Figure 2.3 [1]. We introduce this distinction to properly separate this work from the medical research that is interested in the mechanics of coughs during these pre-expulsive phases.

All cough classification efforts seek to quantify various properties of the expulsive cough phase, such as its "wetness" or "dryness", referring to a cough that does or does not produce

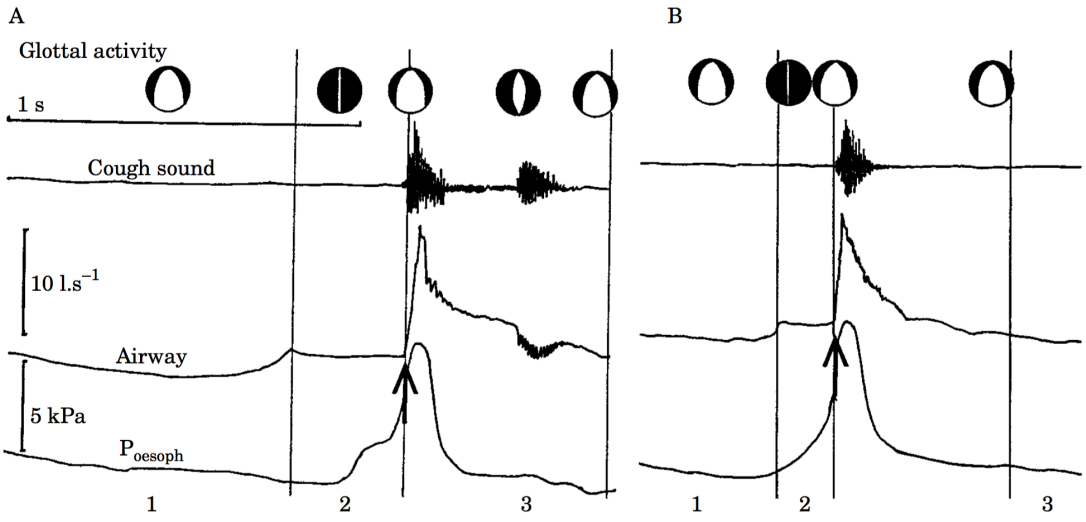

Figure 2.1: From Korpás et al. [37], showing glottal position, recorded sound, airflow and pressure within the esophagus for two example coughs, segmented into three time regions.

sputum when the patient coughs. This sputum (biological byproduct typically created due to the body's immune response to an infection within the lungs) causes a "rattling" sound within the patients chest as they cough. The effect ranges from subtle to extremely prominent depending on a variety of factors, one of which is the severity of pulmonary disease causing the buildup of sputum within the lungs in the first place, and serves as an important diagnostic component when trying to quantify the severity of a pulmonary ailment within a patient.

2.2 Overview of Cough Analysis

Cough sounds have long been known to be of diagnostic interest to pulmonologists. In 1996, Korpás et al. gave an overview of the analysis of cough sound records, or "tussiphonograms", as they are often referred to as [37]. Korpás performed basic signal processing operations upon the captured signals, inspecting the time domain waveforms and frequency domain periodograms in a clinical setting. Of particular interest is the conclusion that the cough and the results of spirometry tests contain separate pieces of information, as while administering

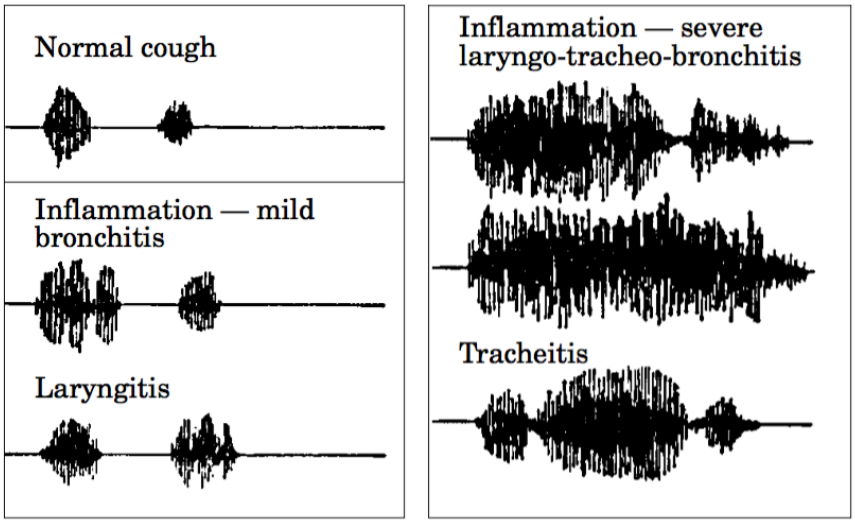

Figure 2.2: The variation of sound recordings of the expulsive segment of a cough, from Korpás et al. [37]

bronchodilating drugs to patients would alter the spirometry results, it would leave the cough analysis results relatively unchanged.

In 2014, Spinou and Birring gave another overview of the measurement and monitoring of cough, with an explicit emphasis on the tools available to medical professionals that wish to investigate cough sounds and their frequency [54]. Spinou and Birring note that beyond simply cough frequency there are other dimensions upon which information from coughs can be extractedl; however, all of the available systems that are capable of extracting this information are at best semi-automatic, and in the end require a human to perform the actual classification. Note that their analysis included only systems that can take ambulatory recordings and extract cough sounds from them, they did not include systems intended to classify pre-segmented cough sounds.

Drugman et al. investigated the sensors most likely to be able to extract information about coughs, evaluating contact and non-contact microphones, electrocardiography sensors, chest belts, accelerometers and thermistors placed over under the patient's nose [24]. Their analysis

determined that, given the choice of a single sensor, non-contact microphones contained significantly more information about the cough than any other sensor.

2.3 Related Cough Signal Processing

In this section we will give a brief overview of work that has attempted to detect, analyze or classify coughs using a variety of sensing modalities. We will report sensitivity (the ability for the algorithm to detect a cough sound when it actually occurred, also known as true positive rate) and specificity (the ability for the algorithm to reject a sound when it is actually not a cough sound, also known as true negative rate) for each comparison algorithm, as these metrics are the typical measurements to be used.

2.3.1 Cough Detection

Barry et al. used Linear Predictive Coding and Mel-Frequency Cepstral Coefficients to model the sound of coughs, and used a Probabilistic Neural Network to classify time windows as containing a cough or not [7]. This system, published as the *Hull Automatic Cough Counter* requires a human to perform the final cough count after being presented with the windows of time containing cough sounds, so as to deal with false positives and multiple coughs overlapping into a single classification. An hour of recorded audio could then be reviewed in, on average, 1 minute 35 seconds, substantially increasing the amount of data health care professionals could process.

Matos et al. fed Mel-Frequency Cepstral Coefficients (and derivatives of the same) into a Hidden Markov Model (HMM) to determine the location of coughs in the recordings [45]. Interestingly, the structure of the finite state machine that the HMM represented contained multiple (approximately 10) left-to-right sub-states for coughs, leading one to theorize that perhaps the HMM was able to learn some structure within the cough sounds themselves, with no domain knowledge other than the number of sub-states within the cough state.

Building off of the work of Matos et al., Birring et al. designed the *Leicester Cough Monitor*, an ambulatory system designed to use the same general system as previously published

by Matos et al. and validate its usage in a more naturalistic setting [11]. Using a portable recorder and a microphone that the user clips to their shirt, the patients are able to go about their lives while gaining a 6-hour audio recording of their daily dealings and, in particular, their coughs. These coughs are automatically analyzed by an offline processing algorithm, the results are reviewed by a human operator, then a second algorithm improves upon the original algorithm's detections. This second study vastly increased the number of coughs and the variety of sounds captured in the recordings as compared to the original study performed by Matos et al. Additionally, the authors investigated the repeatability of their results by testing patients with chronic cough disorders again three months after the fact and ensuring that the results were the same.

Barton et al. used a system published as *VitaloJAK*, a two-channel recording device that uses both a contact microphone on the chest as well as a non-contact microphone to simplify the signal processing challenge of disambiguating cough sounds from other acoustic events [9, 8]. The system then calculates various running statistics on the short-time spectrum of the signals, and captures regions of the signal that have high energy and high spectral center of gravity. Once these periods of high energy/spectral center of gravity are determined, the coughs are counted manually. This system, similar to that of Barry et al. [7] is primarily used as a kind of data compression system, heavily reducing the amount of data that a human operator must classify in order to obtain cough counts.

Amrulloh et al. used a wide variety of signal features (Mel-Frequency Cepstral Coefficients, Formant Frequency, Shannon Entropy, Zero-Crossing Rate, and Non-Gaussianity) fed into a Time-Delay Neural Network to segment cough sounds [5]. The authors expend much effort to accurately find not only the start times of coughs, but the end times as well, giving the ability to measure not only cough frequency but cough duration as well.

Monge-Alvarez et al. used Hu moments with a KNN classifier to obtain a sensitivity of 88% with a specificity of 96% [47]. Their contribution included a thorough analysis of why Hu moments, despite their computational complexity are a good foundational basis for cough detection, especially in the presence of noise. For our purposes, we discount Hu moments as

too computationally expensive to be used as a feature-space transformation (They were found to take an order of magnitude more computational complexity than a spectral estimation technique such as gammatone filterbanks, corroborated by [47]).

You et al. employed spectral subbands fed into linear SVMs to detect cough sounds [61]. Their detection work paid special attention to the performance of their classifiers within the presence of noise; synthesizing noise as well as collecting data with intentional sources of noise in teh background. They were able to achieve a sensitivity of 78% and a specificity of 88%.

Amoh et a. developed a deep learning system similar to that proposed later within this very dissertation [3]. Utilizing convolutional networks in a similar fashion to many recent computer vision models, they were able to obtain a sensitivity of 82% and a specificity of 93%. They also investigated using a recurrent neural architecture for variable-length segmentation, which did not perform quite as well, obtaining a sensitivity of 84% but a specificity of 75%.

Larson et al. used eigenvector decompositions of cough sounds fed into a Random Forest classifier to detect coughs [39]. Of particular note is the unusually high accuracy attained by the authors as well as the development of a "privacy-preserving" feature of the algorithm; the raw audio data, as part of the analysis process, is transformed through the eigenvector matrix built during training, destroying the intelligibility of any speech in the audio recording but retaining the cough sounds enough to perform cough detection. This is a feature many medical professionals desire, as it eliminates the privacy concerns of retaining large amounts of speech data of users that are not necessarily affiliated with the study at all.

2.3.2 Cough Classification

Many groups have done research throughout the past few decades to develop automatic systems to classify various attributes of coughs such as whether the cough is wet/dry or the intensity of the cough. Various diseases such as Asthma, Tuberculosis, Bronchitis, etc., can have very prominent effects on the pulmonary system and are therefore identifiable through the changes they affect upon the sounds of coughs. The identification of these changes to

a typical cough sound is the challenge of cough classification, a brief overview of which we include here:

Al-khassaweneh and Abdelrahman used Wigner Distribution functions and Wavelet Packet Transforms to analyze the time-frequency energy distributions of cough sounds to detect asthma [2]. Their analysis showed that asthmatic patients tended to have coughs with different energy signatures than non-asthmatic patients. In particular, asthmatic coughs had classifiably more energy, especially in the low frequency/long scale bins.

Subbaraj et al. employed a bandpass temporal energy estimate of cough recordings to classify the intensity of a cough [56]. They also investigate potential visualizations for their cough counting methods, mapping classified cough intensity versus time, to give a high level overview of a patient's cough activity over long periods of time. While the cough detection and wet/dry classification done is not automatic and requires human intervention, the intensity regression is.

Swarnkar et al. used a wide variety of signal processing methods to differentiate wet and dry coughs [57]. Their methods included analyzing spectral energy, temporal envelope, and time-independent waveform statistics such as kurtosis, fed into a Logistic Regression Model. Of particular note is the inconsistency of the human ground-truth scoring of coughs as wet/dry; The authors employed two domain experts to perform the scoring, and the two experts agreed on only 80% of all cough events. This underscores the difficulty of such a classifier; even among human domain experts, what constitutes a wet cough versus a dry cough is not completely well defined. The accepted approach in this and other works is to use only the events upon which a majority of expert annotators are in agreement, and to ignore the others.

Chatrzarrin et al. explicitly break cough sounds down into multiple temporal and spectral regions [15]. By inspecting the energy of the signal across time and in a few different frequency bands, the authors were able to identify patterns in the energy envelopes consistent in wet versus dry coughs. It is worth noting that in contrast to many other related works in this case only eight coughs were used for evaluation.

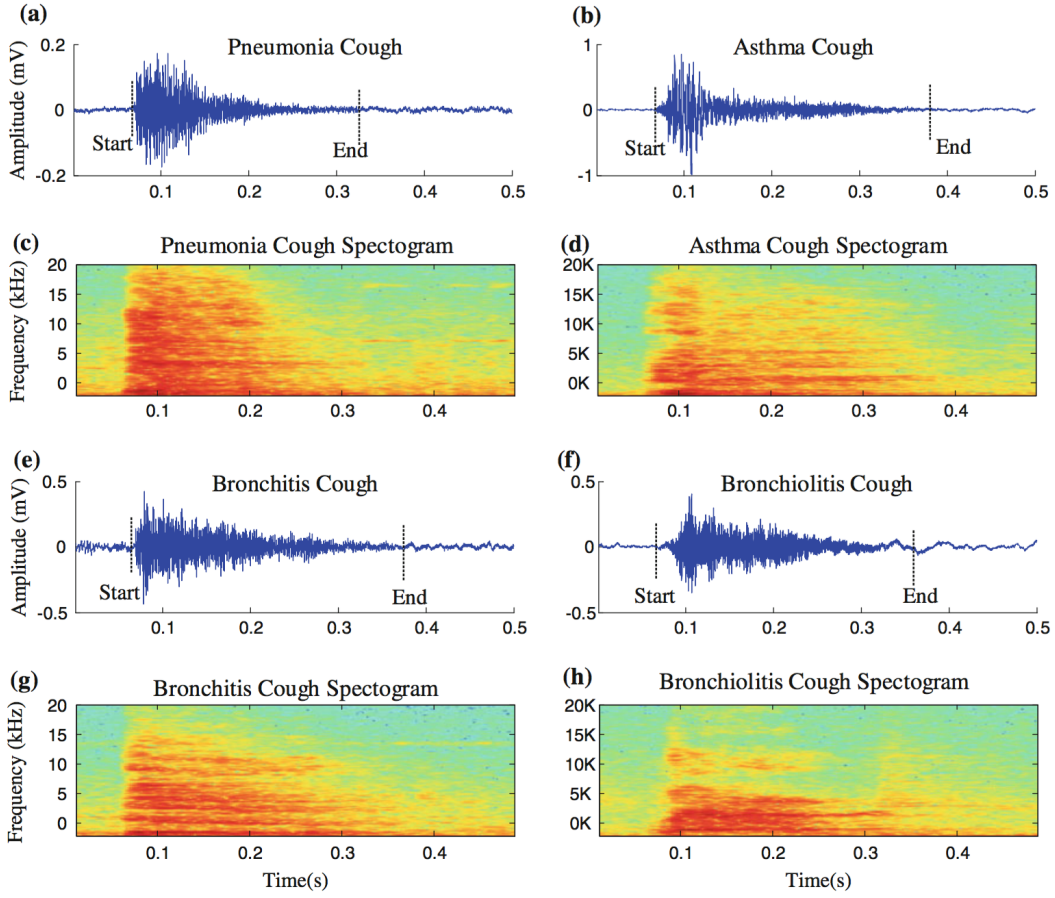

Figure 2.3: Abeyratne et al. display the variation of coughs from patients with different pulmonary ailments [1].

Abeyratne et al. analyzed cough sounds to diagnose asthma in pediatric patients [1]. Using a combination of time-series statistics (such as Non-Gaussianity score, kurtosis, etc..), formant-frequency tracking, general temporal-spectral energy-based features (MFCCs) and others, the authors built a logistic regression model to classify children as either Pneumonia or non-Pneumonia. They note the usefulness of cough sounds to diagnose a wide array of pulmonary ailments, providing a visualization of some of the variations within coughs of four different cough recordings, included here as Figure 2.3. Although their cough classification accuracies are not as high as many of the other references, their standard to measure against

and measurement point to beat are the World Health Organization guidelines for diagnosing pneumonia, which give (on the set of 91 patients) a sensitivity of 83% and a specificity of 47% for an overall accuracy of 75%. This underscores the relatively low bar that many of these algorithms need to overcome in order to be useful in the medical field.

Much more recently, Botha et al. built a system to detect tuberculosis using cough sounds [12]. Using a logistic regression on the log-power of overlapped spectral bands in combination with non-invasive clinical measurements (including mid upper arm circumference, body temperature, body mass index, eye conjunctiva and heart rate), they were able to obtain an impressive sensitivity of 82% with a specificity of 95%. Using only log spectral features, their sensitivity dropped to 60% with a specificity of 78%. A key difference between this work and others is that the control cough dataset consisted of forced coughs by users without any other recorded pulmonary ailments.

2.3.3 Relation to Thesis

These related works listed above, separated into the two broad categories of *detection* versus *classification* serve to provide a baseline and minimum bar of performance that our work must meet or exceed in order to be considered competitive. Summaries of this previous work are given in tables 2.1 and 2.2 to show the relative detection performance and collection of classification tasks previous work has attempted for quick and easy reference.

These collected works also illustrate the wide variety of research questions surrounding coughs as a clinical tool and underscore the interest the medical community has in performing this analysis. We also wish to explicitly point out the varying applications and subsequent performance levels expected of published research in these fields. For some cases, such as automatic ambulatory cough detection, specificity appears to be very desirable, as a key metric reported by almost every publication is the number of false alarms per hour. However, in the case of a nighttime cough tracker, sensitivity is not quite as critical for the algorithm to be effective, as the number and type of acoustic events requiring classification will be much more limited.

Publication	*Results*	*Auto*	*Coughs*	*Methods*
Barry [7]	Sensitivity: **80%** Specificity: **96%**	Partially	75	LPC/MFCC's with a PNN
Matos [45] **Birring [11]**	Sensitivity: **91%** Specificity: **99%**	Partially	1834	MFCC's with an HMM, followed by human-guided detection algorithms
Barton [8]	Sensitivity: **98%**	Partially	1932	Dual microphones with spectral energy thresholds and human operators
Amrulloh [5]	Sensitivity: **91%** Specificity: **97%**	Fully	1434	A variety of features with a TDNN, on pediatric coughs
Monge-Alvarez [47]	Sensitivity: **88%** Specificity: **96%**	Fully	N/A	Hu moments with a KNN classifier
You [61]	Sensitivity: **78%** Specificity: **88%**	Fully	N/A	Spectral subband features fed into linear SVMs
Amoh [3]	Sensitivity: **82%** Specificity: **93%**	Fully	627	STFT visually classified using CNNs
Larson [39]	Sensitivity: **92%** Specificity: **99%**	Fully	2558	Eigenvector features with a random forests classifier

Table 2.1: Summary of previous cough detection work showing the results, whether the method was partially or fully automated, the size of the dataset in number of coughs, and the methods employed.

Publication	*Task*	*Results*	*Auto*	*Coughs*	*Methods*
Al-khassaweneh [2]	Asthma	Sensitivity: **88%**	Fully	24	Spectral estimation with KNN
Subbaraj [56]	Intensity	Accuracy: **98%**	Partially	215	Temporal energy-based regression
Swarnkar [57]	Wet/Dry	Sensitivity: **55%** Specificity: **93%**	Fully	536	A variety of features with an LRM
Chatrzarrin [15]	Wet/Dry	Sensitivity: **100%** Specificity: **100%**	Fully	8	Spectral/temporal thresholding
Abeyratne [1]	Pneumonia	Sensitivity: **80%** Specificity: **73%**	Fully	440	A variety of features with an LRM
Botha [12]	Tuberculosis	Sensitivity: **82%** Specificity: **95%**	Fully	518	Log-spectral bands with an LRM and clinical metrics

Table 2.2: Summary of previous cough classification work showing what classification task was attempted, the results, whether the method was partially or fully automatic, the size of the dataset in number of coughs, and the methods employed.

2.4 Deep Learning

Our cough detection and classification algorithms make use of the recent advances in machine learning technologies yielded by the advent of "deep learning" [40, 53]. Historically, signal processing algorithms and machine learning models would be expert systems, designed by humans with deep domain knowledge taking advantage of physical or mathematical properties of the signals being analyzed in order to extract relevant information. Deep learning seeks to reduce the required expert knowledge by allowing computations of the correct type and shape to automatically learn the parameters for the computations through back-propagation [53]. We will give here a brief introduction to the deep learning paradigm, and discuss challenges faced when building these systems.

2.4.1 Neural Networks

Deep learning is built upon the research outcomes developed over the past half-century in investigating what are commonly known as "neural networks". Created as a simplistic mechanical biomimicry of neurons in the human brain, a "neural network" is a nonlinear function approximator. The simplest form of a neural network is the "perceptron"; an affine function of its inputs, passed through some form of nonlinearity:

$$f(X) = \sigma\left(\sum_i W_i X_i + b\right) \tag{2.1}$$

Equation 2.1 shows a perceptron (f) operating upon a vector of inputs (X) that are linearly combined with a vector of weights (W) and a bias term (b) [40]. This affine transformation results in a single value which is finally passed through the nonlinear warping function σ.

The function $f(X)$, typically viewed as a predictor or regressor of the inputs, is typically matched with an expected output (Y), and the parameters of the computation (W and b) must be set such that $f(X) = Y$). The benefit to this formulation is the ease with which it can be analyzed through the viewpoint of differentiable programming. Because $f(X)$ is

composed of operations with well-defined derivatives with respect to the inputs X, it is simple application of the chain rule to calculate $\frac{\partial f}{\partial W}$ and $\frac{\partial f}{\partial b}$ (assuming, for the moment, that $\sigma(x)$ is the identity function, that is, $\sigma(x) = x$):

$$\begin{aligned}\frac{\partial}{\partial W} f(X) &= \frac{\partial}{\partial W}\left(\sum_i W_i X_i + b\right) \\ &= \sum_i \frac{\partial}{\partial W}\left(W_i X_i\right) \\ &= \sum_i X_i \end{aligned} \tag{2.2}$$

$$\begin{aligned}\frac{\partial}{\partial b} f(X) &= \frac{\partial}{\partial b}\left(\sum_i W_i X_i + b\right) \\ &= \frac{\partial}{\partial b}\left(b\right) \\ &= 1 \end{aligned} \tag{2.3}$$

This allows us to manipulate W and b to effect a desired change in the output for a given input. By defining a "loss" function as a measurement of the difference between the current output of $f(X)$ and the desired output Y, then using that loss to inform how W and b should be modified in order to move the output of $f(X)$ closer to Y, we are able to "learn" optimal values of W and b:

$$\ell_2(f(X), Y) = \|f(X) - Y\|_2^2 \tag{2.4}$$

Minimizing $\ell_2(f(X), Y)$ across a wide set of $X \rightarrow Y$ mappings by tweaking the parameters W and b yields the least-squares solution to the optimization problem:

$$\begin{aligned} &\underset{W,b}{\text{minimize}}\ \ell_2(f(x), y) \\ \Rightarrow &\underset{W,b}{\text{minimize}}\ \left\|\sigma\left(\sum_i W_i X_i + b\right) - Y\right\|_2^2 \end{aligned} \tag{2.5}$$

This technique, known as *gradient descent*, lies at the heart of all neural network learning techniques, and is made possible by building computational models that are able to calculate

derivatives of the output with respect to all parameters being optimized. The operation of tracing derivatives backwards from the loss function to parameters is referred to as *back-propagation*. The typical formulation for gradient descent is to define a "training loop" wherein new $X \rightarrow Y$ mappings are used to calculate a loss, that loss is used to find new derivatives (also referred to as *gradients*) for parameters W and b, those derivatives are scaled by a learning rate (η) and finally accumulated into the parameters themselves, as shown in algorithm 1. Each iteration of the training loop (comprising a full pass over the entire training set S) is referred to as an *epoch*. Training time is typically measured in epochs, and model convergence criteria is also typically evaluated at epoch granularity.

ALGORITHM 1: Gradient descent training procedure

Input: X and Y from dataset S, parameters W and b, learning rate η

Output: Updated weights W and b

`// Loop over all entries in training set` S

for $X, Y \in S$ **do**

`// Calculate output using current` W_t `and` b_t

$\hat{Y} = f(X)$

`// Determine loss between calculated output and target output`

$L = \ell_2(\hat{Y}, Y)$

`// Update parameters using backpropagated gradients`

$W = W + \eta \left(\frac{\partial f}{\partial W} L \right)$

$b = b + \eta \left(\frac{\partial f}{\partial b} L \right)$

end

Although in its most simple formulation the entire training set S can be calculated on at once, this is rarely done in practice. Rather, the training set S is typically broken up into *minibatches* containing a limited number of training examples (sizes from 1 to 1024 are often seen in practice [40]). This method of operating upon and optimizing for a small subset of the full training set at a time is known as *stochastic gradient descent*, and is the foundation

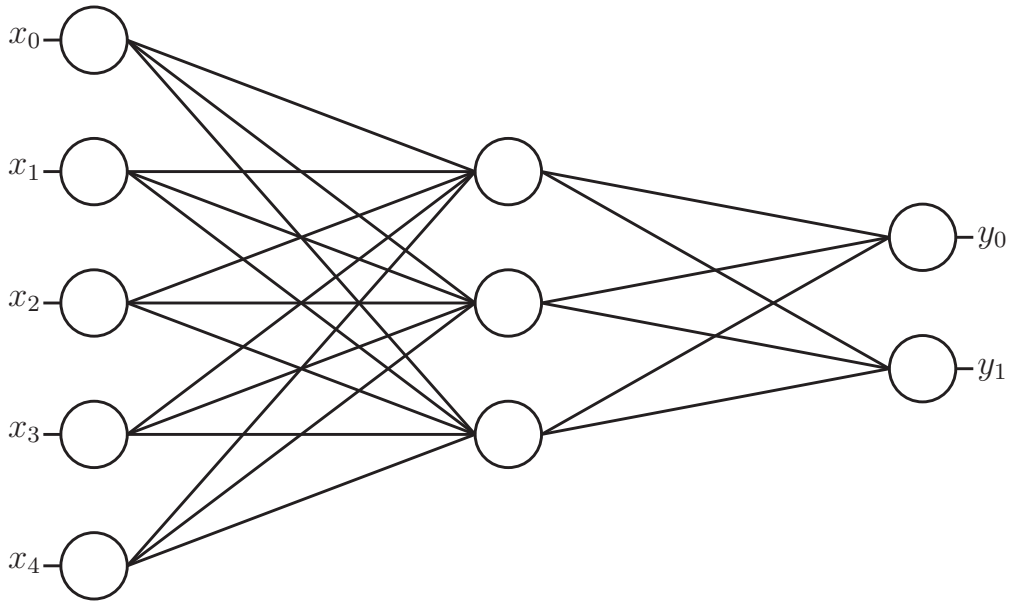

Figure 2.4: Diagram of a multilayer perceptron, graphically depicting the flow of information from an input layer of 5 neurons (x_i) to an intermediate layer of 3 neurons, to an output layer of two neurons (y_i). Each connecting line conains an independent weight that multiplies the value of the source node on the left, and is summed together with all other lines entering the destination node on the right.

for the learning methods used upon all neural networks in this dissertation.

2.4.2 Deeper Networks

A single perceptron, as given in equation 2.1, is capable of learning functions of very limited complexity. As stated above, the only functions capable of being learnt are affine transformations (with the exception that the nonlinearity σ, ignored so far, can add some vital complexity in and of itself, however we will soon see that σ is typically a very simple nonlinearity itself). In order to increase the complexity of functions that can be learnt by the neural network, we simply add another set of transformations on to the end of the first, cascading a similar set of operations onto the end of the first. This is shown graphically in

Figure 2.4, where multiple perceptrons have been grouped together to form a "multilayer perceptron" (MLP). Each node with lines feeding into it within the diagram denotes a single perceptron taking in the values from the previous set of nodes, multiplying those values by the "weights" implied by the graphical line, summing them together and then applying a nonlinearity to that sum. In Figure 2.4, there are two "layers" of perceptrons feeding from a layer of input nodes.

This stacking of operations shows the necessity of including a nonlinear term within the formulation of a perceptron; without a nonlinearity, stacks of affine operations could always be reduced to a single equivalent affine operation, limiting the scope of functions that could be learnt to simply affine ones. On the other hand, the nonlinear elements must be sufficiently simple that they have an easily computed derivative for backpropagation to be able to derive useful gradients for parameters throughout the neural network. With these constraints in mind, the families of neural network techniques naturally arise, with extremely simple nonlinearities such as the "ReLU", "ELU" and "tanh" functions used widely in contemporary machine learning literature [40]. These functions and their derivatives are shown in Table 2.3, demonstrating their simplicity.

Stacks of multilayer perceptrons have been used to great effect as general-purpose nonlinear function approximators. The ability to combine stacks of relatively simple nonlinear pieces to model more complex nonlinear functions has unlocked many new applications and yielded impressive results on machine learning tasks from many separate fields. The counterpoint to this is that the straightforward fully-connected layers of neurons shown in Figure 2.4 cause an explosion in the number of parameters that must be learned from training data. The number of parameters in each layer grows as $O\left(n^2\right)$ in the number of input variables n. In order for a statistically-constrained model such as a neural network to converge, the number of training examples used to train must exceed the number of degrees of freedom within the model by a sufficient threshold to constrain the parameters meaningfully. Data augmentation techniques such as perturbing inputs [58] or model augmentation techniques such as adding dropout [55] and introducing regularization constraints [38] can alleviate

Function	***Definition***	***Derivative***
ReLU [40]	$\begin{cases} x, & \text{for } x \geq 0 \\ 0, & \text{for } x < 0 \end{cases}$	$\begin{cases} 1, & \text{for } x \geq 0 \\ 0, & \text{for } x < 0 \end{cases}$
ELU [19]	$\begin{cases} x, & \text{for } x \geq 0 \\ \alpha\left(e^x - 1\right), & \text{for } x < 0 \end{cases}$	$\begin{cases} 1, & \text{for } x \geq 0 \\ \alpha\left(e^x - 1\right) - 1, & \text{for } x < 0 \end{cases}$
Leaky ReLU [44]	$\begin{cases} x, & \text{for } x \geq 0 \\ \alpha x, & \text{for } x < 0 \end{cases}$	$\begin{cases} 1, & \text{for } x \geq 0 \\ \alpha, & \text{for } x < 0 \end{cases}$
Sigmoid [40]	$\frac{1}{1+e^{-x}}$	$\frac{1}{1+e^{-x}}\left(1 - \frac{1}{1+e^{-x}}\right)$

Table 2.3: Example nonlinearities used in deep neural networks, showcasing their simplicity and easily computed derivatives.

some of the need for vast quantities of data; however there remains an intrinsic relationship between the number of parameters within a model and the number of training examples required for a model to stably converge. In this thesis, we explicitly address the problem of insufficient data through a combination of models with a restricted set of parameters, data augmentation, and adversarial networks.

2.4.3 Convolutional Neural Networks

Convolutional neural networks [6, 41] (CNNs) provide an excellent foundation for building detection and classification systems for signals that have information that is "localized". First introduced three decades ago [6], they have found a resurgence in recent years in the computer vision field for their impressive ability to learn joint localized probability distributions in images. A single object within the image will be represented in pixels that are neighboring each other, therefore a network that seeks to extract information about that object may not need to simultaneously draw information from all pixels at once, but may be able to restrict

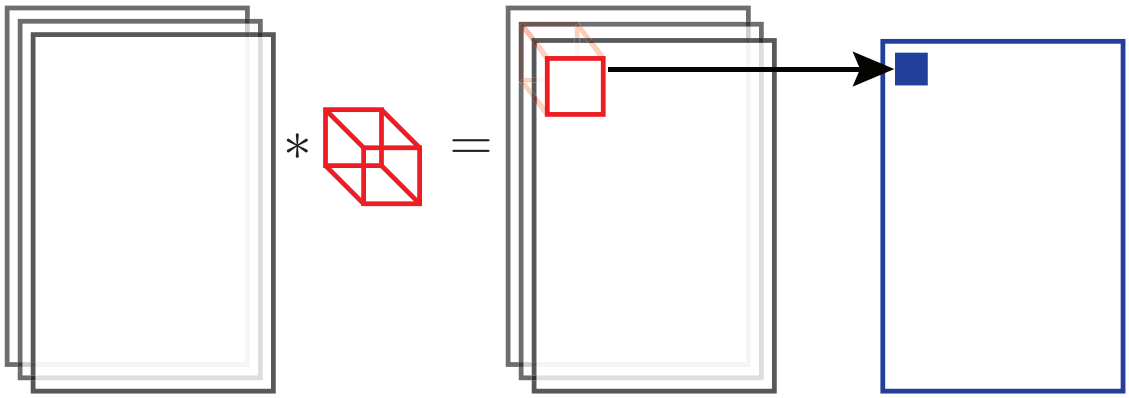

Figure 2.5: Diagram of a convolutional neural network, graphically depicting a convolution between a 3-channel input tensor and a convolutional kernel in red. The convolutional kernel is shifted over the domain of the input tensor, calculating a weighted sum at each position and generating a single output pixel per shifted location in the input. The generated output plane of pixels is shown in blue.

its view to small neighborhoods of input. A convolutional neural network finds shift-invariant joint probability distributions inherent within the input signal that can be used by further networks in later layers. This can be viewed as a kind of "pattern matching", and maps very well to the traditional signal processing operation of using convolutional operations to create a matched filter, where the patterns being searched for in a CNN are learned through backpropagation and gradient descent. Mathematically, the pattern match probabilities are calculated through the fundamental convolution equation:

$$\mathbf{O}(x, y, j) = \sum_{i=1}^{m} \sum_{u,v=0}^{s-1} \Big(\mathbf{K}(u, v, i, j) \mathbf{I}(x + u, y + v, i) \Big) + \mathbf{B}(j) \tag{2.6}$$

Where the input tensor $\mathbf{I} \in \mathbb{R}^{w \times h \times m}$ is transformed by the kernel tensor $\mathbf{K} \in \mathbb{R}^{s \times s \times m \times n}$, added to the bias tensor $\mathbf{B} \in \mathbb{R}^{n}$ to create the output tensor $\mathbf{O} \in \mathbb{R}^{w \times h \times n}$, and each element of the output tensor $\mathbf{O}$ represents the likelihood of a particular pattern existing at that location within the input tensor. The parameters w and h represent the input tensor width and height respectively, while m and n represent the channels of the input and output

tensors. We note that the learned parameters **K** are independent of input data size and are typically orders of magnitude smaller than the data they operate on. The number of learned parameters also does not scale with input size, it is a function only of the number of patterns to be sifted out of the input image, and the size of pattern to be determined. This greatly reduces the amount of data necessary for training, and has found large success in computer vision applications. Convolutional networks serve to continually extract information and reduce the dimensionality of input data until a final detection/classification stage which is applied once the data rank is small enough to be amenable to more traditional neural network architectures, such as fully connected layers.

2.4.4 Adversarial Networks

Generative adversarial networks (GANs) are a deep learning method for transforming one signal domain to match the distribution of another [28]. GANs have been proven to be a state-of-the-art method for matching the distribution of datasets. For example, in a process known as style transfer, GANs are used to reconstruct an image in a manner stylistically similar to a target image [34]. A GAN operates by taking a standard deconvolutional network that operates upon noise as an input and produces an image and pairs it with a discriminator network that is trained to distinguish generated (fake) images from the target images. During training, the accuracy of the discriminator is used as a loss term for the generative network. The loss produced by the discriminator effectively forces the generative network to learn the distribution of the target dataset in order to consistently fool the discriminator.

In this thesis, we take the concept of a GAN and introduce a small modification we call the discriminative adversarial network (DAN). We apply a DAN to ensure that our classification algorithm learns features independent of data collection site, an essential detail when attempting to fuse two highly imbalanced data collections. This DAN operates in opposition to the classification network, forcing portions of the classification network to not make use of patterns or information within the input data that could be used to determine which data collection the data originated from. In essence, it forces the model to ignore

certain pieces of information that would otherwise heavily bias the model predictions. The DAN will be described in greater detail in Chapter 4.

Chapter 3

TECHNIQUES FOR COUGH SOUND ANALYSIS

As stated in section 1.2, a key research contribution of this dissertation is the development of algorithms for cough detection and cough classification. In this chapter, we will detail the development of these algorithms, comparing different methods and relating them to the applications listed in chapter 1. We will compare and contrast previous methodologies with two new algorithms for cough detection, paying special attention to the design criteria listed in section 3.2, and then discuss performance of one of these detection architectures upon a cough classification task.

3.1 Dataset

Our machine learning models are trained upon what is, to our knowledge, the largest cough sound database collected in the world. The dataset was collected from two separate sites: an *Ambulatory* dataset and a *Clinical* dataset. Although each were collected for distinct purposes, we will show that the combination of both poses challenges and provides opportunities for developing more robust and useful machine learning models than either dataset used independently. The audio recordings in both cases were annotated by a team of expert annotators trained to detect cough sounds.

The *ambulatory* dataset was collected at the University of Washington and consists of 64.3 hours of audio taken from 17 subjects (7 female) already known to exhibit cough symptoms before enrollment; 8 participants were diagnosed with a cold, 5 with chronic cough due to various reasons including smoking, 3 with asthma and 1 with allergies. A total of 2420 coughs are represented in this dataset. The *ambulatory* dataset was collected by participants wearing a smartphone as a personal recorder device on their persons and recording 3-6 hours of data

	Coughs (TB)	Participants (Female)	Total Length	Combined Cough Length
Clinical	**2273** (**1924**)	**84** (**34**)	**82.5** hrs	**13.5** mins
Ambulatory	**2420** (**0**)	**17** (**7**)	**64.3** hrs	**11.1** mins
Total	**4693** (**1924**)	**101** (**41**)	**147.7** hrs	**24.6** mins

Table 3.1: Dataset breakdown by number of coughs captured, participant demographics, and total recording length in hours.

as they went about their daily activities. Represented within this dataset are a multitude of other audio sources such as human speech, motor vehicles, laughter, and sounds as the mobile device moved about.

The *clinical* dataset was collected at the Desmond Tutu HIV Foundation through a partnership with the Gates Foundation and consists of 52.6 hours of audio taken from 56 participants (27 female), 45 of whom were previously diagnosed with pulmonary tuberculosis. A total of 2273 coughs are represented in this dataset, with 1924 of those coughs produced by TB-positive subjects. The *clinical* dataset was collected by participants sitting in a closed room with a TV playing in the background for 1 hour. Represented within this dataset are human speech, TV noise, and various background sounds such as door closing and a whirring fan.

For cough classification tasks, it is imperative that mixing data from separate data collections does not cause the machine learning model to simply distinguish which site a recording came from due to the model learning some commonality within datasets such as the acoustics of the room. This would cause the model to learn about the environment the cough sounds were collected within as opposed to the cough sounds themselves. As detailed in the paragraphs above, our datasets are a prime example of unbalanced datasets where this would be a problem, as all tuberculosis coughs are contained within the *clinical* dataset, and almost all of the control coughs are contained within the *ambulatory* dataset. To address this imbalance, we will utilize a Discriminative Adversarial Network (DAN) to ensure that

the machine learning model learns more than just differences in data collection sites when training classification models. This is described in greater detail in chapter 4.

All data collections were approved by the Institutional Review Board at their respective collection sites.

3.2 Model Design Criteria

Section 2.3.1 listed multiple research efforts to build automated cough detection algorithms. We note that while there are a wide variety of different methods surveyed, there exist broad similarities, especially among systems that seek to disambiguate coughs from other common sonic events. All systems follow the same basic pattern: to chop streams of audio up into small time segments (referred to as *frames*), perform some type of feature-space transformation/dimensionality reduction upon each frame individually, then feed those features into a machine learning model to classify that collection of frames as either a cough or a non-cough event. While variations upon this theme abound, this technique has found wide adoption within the signal processing and machine learning communities for its applicability to almost any signal detection/classification problem.

There are a few fundamental assumptions inherent in applying a machine learning model to a new problem:

- First, it is assumed that the machine learning task is possible; that the machine learning model contains the necessary mathematical expressiveness to separate, regress or otherwise process the input information into the desired output. A counter-example to this is attempting to use a linear model to approximate an extremely nonlinear phenomenon; in general it is impossible for the model to faithfully approximate such a function.

- Second, it is assumed that the statistically-determined elements of the machine learning model (the *parameters*) are given enough training data to be able to converge to reasonable values. A counter-example to this is attempting to train a linear model

containing 100 parameters on a dataset with only 10 input/output mappings; with such a dearth of data, the parameters within the model will be grossly underspecified, and in general will not perform well on new data it has not trained on.

- Third, it is assumed that the machine learning model's output can be obtained in an environment that is useful to the application at hand. A counter-example to this is building a model that cannot be executed within the resource constraints of a mobile or embedded platform that the application requires.

It is extremely important to ensure that these assumptions are given due consideration when applying machine learning to a new problem, and cough sound analysis is no exception.

Traditional signal processing addresses the first assumption through domain-specific feature engineering. As an example, an impressive amount of literature exists on speech signal processing, and many techniques exist for decomposing speech signals across various axes of information. Linear Predictive Coding (LPC) models the vocal tract as a series of resonant cavities, decoupling the effect of the vocal cords from the shaping influence of the vocal tract. Mel-Frequency Cepstral Coefficients (MFCCs) use a biologically-inspired spectral decomposition to break audio signals down into an unevenly sampled time-frequency distribution that mimics the human auditory peripheral system's response to sound. Both of these techniques significantly reduce the complexity of speech signals from the standpoint of a machine learning model. Information previously scattered throughout the time-domain signal is extracted and collected into bins in frequency, or as coefficients of a much lower-dimensional polynomial. This information extraction is hand-designed by human experts that have knowledge of the signals being analyzed and are thus able to make inferences about the nature of the information being analyzed and which pieces of information are most important to extract, which can be discarded, and what format of information would be most advantageous to present to a machine learning algorithm. This hand-tuned feature extraction serves to simultaneously reduce input dimensionality (thereby reducing the number of parameters that must be learned within a model) and preprocess data into a simpler format for the machine

learning model, reducing the necessary complexity of the model itself.

The advent of deep learning brings with it new capabilities for automation in feature selection and design. Automatic feature extraction is an area of active research, attempting to take the ideas of backpropagation and gradient descent to their logical extremes and learn parameters for the entire pipeline of operations, from raw audio data input to model output. Efforts towards this have shown impressive results in some domains by allowing backpropagation to alter feature-space transformations that may not otherwise have been optimal with regards to the eventual output of the network. This approach is able to reduce or even eliminate human effort in designing the first stages of the machine learning pipeline, however it inherently increases the number of parameters that must be learned, thereby increasing the amount of data necessary to satisfy the second assumption listed above.

Finally, our intended applications listed in chapter 1 require an efficient, small-memory-footprint solution to cough detection. Many methods have either prohibitive computational costs or memory requirements that far exceed the limits imposed by modern embedded systems. Solutions to address these issues range from limiting the expressiveness of a model through architectural design constraints, to altering the nature of a computation by, for example, approximating coefficients with lower precision numeric types thereby saving on both memory and computational resources.

In this dissertation, we will explore the design space of cough processing algorithms along the axis of application-specific feature design versus deep learning. In particular, we will investigate the benefits and disadvantages of using deep learning based techniques to perform this kind of auditory analysis. We begin by creating a "baseline" algorithm. This algorithm is designed to mimic previous work so as to give us an idea of what algorithms similar to those previously published would attain on our datasets. We then show the naive application of deep learning to our problem space, identify issues with this approach and return with a more targeted application of deep learning that performs much better.

3.3 Dataset Loading

Previous work in this space has utilized a wide variety of feature-space transformations including LPC, LSP, MFCCs and simple energy-based measures. Sections 3.4 and 3.5.1 will show comparisons between the performances of these different features with various learning models. We detail here the data loading pipeline that is common across all features, so as to minimize differences between model and feature types when comparing approaches.

The data recordings being analyzed and learnt from within this dissertation are multiple-hour long audio recordings that have been annotated with cough event locations. For all model training, the input dataset (unless otherwise specified, this is the combination of the *clinical* and *ambulatory* data collections) was partitioned by participant and randomly split into 5 folds, one of which was set aside to be used as the test set to measure model accuracy. This was done so as to ensure that cough sounds from a participant are not found within both the training and testing sets. This mimics a deployment scenario, where all data being classified is from participants that the model did not train on. The traditional signal processing and machine learning data loading methodology would be to collate time windows that are interesting into a matrix of feature vectors, then perform machine learning upon that matrix. This works well on machine learning methods that operate until convergence upon a dataset small enough to be held within memory all at once, such as is typical with random forests, support vector machines, etc... This does not work as well with deep learning methods, as the iterative learning loop is made more explicit for the machine learning researcher, and as such it has become common practice to not simply iterate over the same data points over and over, but to instead introduce data augmentation techniques such as adding random noise to training data, randomly permuting the order in which samples are drawn from underlying distributions, and to in general add as much variation into training data as is possible to increase generalizability of the learned model.

To strike a middle ground between these two philosophies, we build an abstraction that grabs random windows of time from the dataset that are guaranteed to overlap at least one

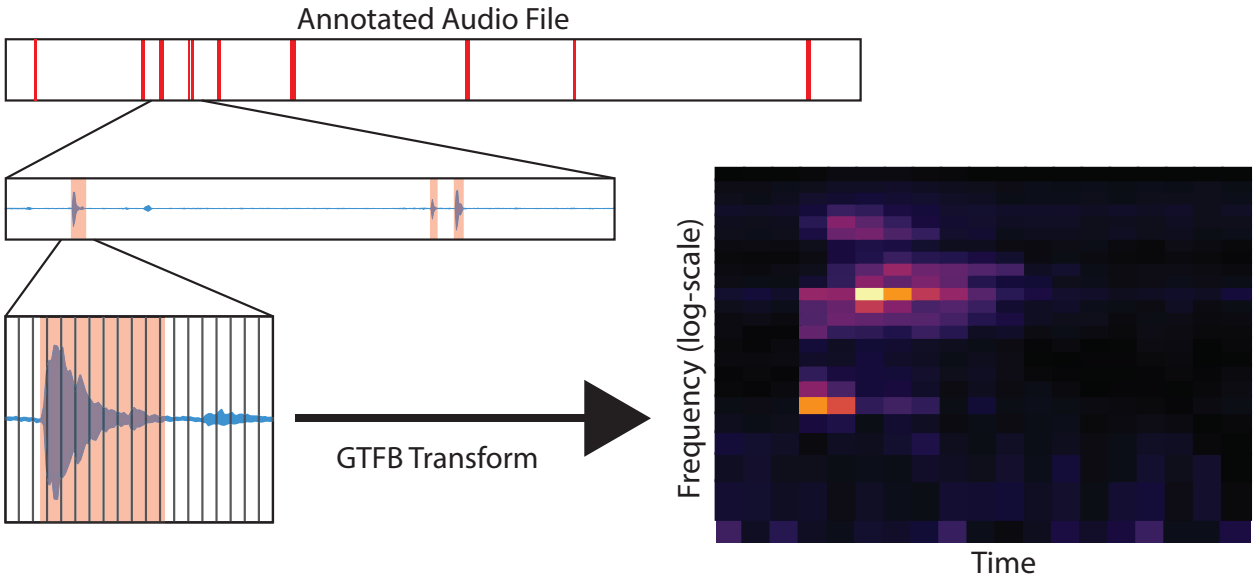

Figure 3.1: Diagram of the feature calculation process. Cough events are annotated within a large audio file, windows of temporal data are taken, split into overlapping frames, then each frame is passed through a feature calculation routine. The "random cough" data selector chooses windows that are guaranteed to contain cough annotations, whereas the "random non-cough" data selector chooses windows at random.

cough event, and add small amounts of noise to them. We refer to this as the "random cough" data selection process. We also build an abstraction that grabs windows completely at random, which will contain silence, fan noise, human speech, etc... We refer to this as the "random non-cough" data selection process. In this dissertation, the windows of time extracted are of length 200ms, and each window is then broken up into frames of 20ms length with 50% overlap, resulting in a collection of 19 separate 20ms-long frames per window. When streaming a file through a classification algorithm, 100 such windows will be classified per second as the earliest frame is discarded and a new frame is appended to the end of the window, shifting it forward 10ms in time. These frames are then individually passed through the chosen feature preprocessing steps. This is shown graphically in Figure 3.1, where an

annotated audio file is shown at the top of the figure, with cough annotations depicted in red. In the bottom left, an extracted time window is shown that contains a cough annotation. The time window is sliced up into overlapping frames, which are then transformed by the gammatone filterbank transformation and depicted in a time-frequency representation on the bottom right of Figure 3.1.

When training a traditional machine learning algorithm, a feature matrix is collected to contain $5N$ samples from the "random cough" data selection process, and $5N$ samples from the "random non-cough" data selection process, where N is the number of unique cough sounds held within the training set. When training a neural network, the explicitly iterative training process (as given in Algorithm 1) makes it simple and natural to constantly sample new random cough and non-cough samples, and so each minibatch of new data fed into the network for training is comprised of half "random cough" samples and half "random non-cough" samples. This results in a training dataset balanced between cough samples and non-cough samples, avoiding the highly imbalanced classes that would occur if the entire dataset were naively used for training. When testing a model, the entire test set is used for validation, and so all performance numbers are indicative of model performance on whole recordings, not on randomly sampled windows of time.

The neural network training loop naturally revisits every cough sound multiple times as it works towards model convergence. This is not explicitly determined, but is implicit in the random sampling of cough sounds used when assembling each minibatch of data for the network to train upon. Using the scheme described above, traditional machine learning methods receive a limited amount of the randomized and augmented data benefits that the neural network training scheme provides. In this dissertation, a value of $5N$ was chosen as a good balancing point, considering training memory and computational requirements (This implies that each cough sound is represented 5 times on average within the training dataset, with each separate instance augmented separately).

3.4 Feature Preprocessing

We evaluated a number of different feature-space transformations including linear predictive coding (LPC) [52], line spectral pairs (LSP) [33, 50], Mel-frequency cepstral coefficients (MFCC) [23], outputs of a gammatone filterbank (GTFB) [51] as well as the raw short-time Fourier transform (STFT). These transformations represent a variety of different ways of decomposing a signal that contains information inherent in oscillations, reverberations, and other physical phenomena. This feature extraction is a critical step to ensure that the machine learning models converge stably and quickly. Without this dimensionality reduction, the number of parameters that must be learned through back-propagation would far outstrip the amount of data available to train on.

The very act of dimensionality reduction through feature extraction implies that there is an assumption being made regarding the nature of the signal being processed. These feature choices, both within this dissertation and in related work, show the assumptions being made by the feature designer: that a prominent feature of cough sounds may be linear correlation between time samples imposed by the vocal tract's filtering of the lung's exhalation sound (LPC, LSP), or perhaps the energy distribution across frequency and time (GTFB, STFT). While we performed a search across feature choices for comparison purposes in the baseline algorithm evaluation in section 3.5.1, for all further work we chose to use GTFB features as the basis for our later algorithms.

The GTFB provides a signal representation that offers locality; information from a single physical phenomenon (such as a cough, a door slamming, etc...) will be "localized" within the time-frequency distribution, according to the manner in which the sound was generated. This provides a natural fit for the convolutional networks discussed within section 2.4.3, as the property of locality, combined with the dimensionality reduction of a dyadic filterbank yields a signal representation that can then be fed into CNNs very easily. Signal processing literature is littered with cases where dyadic filterbanks provide signal compression advantages, from the mathematical basis of Wavelets [22] to the empirically-determined

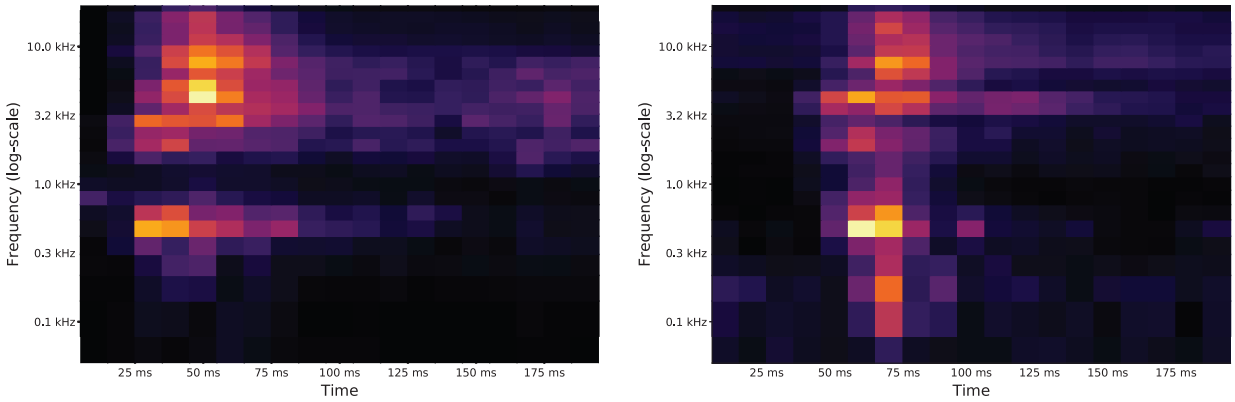

Figure 3.2: Examples of cough signals visualized using the gammatone filterbank (GTFB) spectro-temporal decomposition. Lighter colors indicate higher signal energy within a time-frequency bin.

nonuniform spacing of the Mel frequency scale [23]. It is hypothesized that dyadic filterbanks work well with natural signals such as the human voice due to how these signals are generated. Assuming a sum-of-products signal model where there is an internal excitation that contains some sort of carrier signal which is then modulated by the vocal tract, there arises a natural relationship between absolute frequency and the bandwidth of a particular carrier's energy within the time-frequency space. This natural relationship lends itself well to analysis through dyadic filterbanks. Cough signals do not contain strongly-defined carriers per se, however as wideband signals with large swaths of high frequency content coupled with lower frequency components due to occasional voicing and rattling of the pulmonary system, this signal decomposition still works well. Spectral techniques such as MFCCs do not maintain locality in the same way as techniques such as the STFT and GTFB, and as such do not perform as well when classified upon with CNNs. Raw STFTs, on the other hand, do not perform the dimensionality reduction needed to create the efficient, low-parameter-count model to satisfy the design requirements listed in chapter 1.

An example output of the GTFB is shown in Figure 3.2, with two example coughs visualized within the spectro-temporal plane. Lighter colors signify greater energy concentration

within a particular time-frequency bin. As shown in Figure 3.2 regions of interest can be clearly identified as large, coherent "blobs" in frequency and time.

The first step of our signal processing pipeline is therefore set to be:

$$\begin{aligned} X[i] &= \mathcal{F}\left(x\left[\frac{iK}{2} : \frac{(i+2)K}{2}\right] \cdot h\right) \\ \hat{X}[i] &= WX[i] \\ x \in \mathbb{R}^N, \quad h \in \mathbb{R}^K, \quad W &\in \mathbb{R}^{K \times L} \end{aligned} \tag{3.1}$$

Where $x[i]$ represents the raw input time signal at time point i, $X[i]$ represents a spectral slice of x, $\hat{X}$ represents the output features, $x[start : end]$ represents the sub-sequence of x starting at *start* and ending at *end* and $\mathcal{F}$ is the Fourier transform operator. K is the length of a single "frame" of audio (in this dissertation, this is always 20 milliseconds), L is the number of filters within the GTFB filterbank, and W represents the coefficients of that filterbank, mapping from a Fourier transform output of length K to the filterbank output of length L. As a concrete example, in this dissertation, all data files are audio recordings sampled at 48 KHz, with a frame length K of 960 samples, and a filterbank size L of 24. The frequency response for a 24-length filterbank is visualized in Figure 3.4.

3.5 Detection Algorithm

In this section we first compare against baseline algorithms, then introduce our own novel cough detection algorithm based on recent advancements of convolutional neural networks, fulfilling one part of the research outcomes listed in section 1.2.

3.5.1 Baseline Models

Previous cough detection work generally centers around chopping the input signal up into frames of audio, which are each passed through a feature-space transformation. Transformed frames are then concatenated into a feature vector and passed into a machine learning algorithm which is trained to determine the proper class (e.g. *cough* vs. *non-cough*) from that

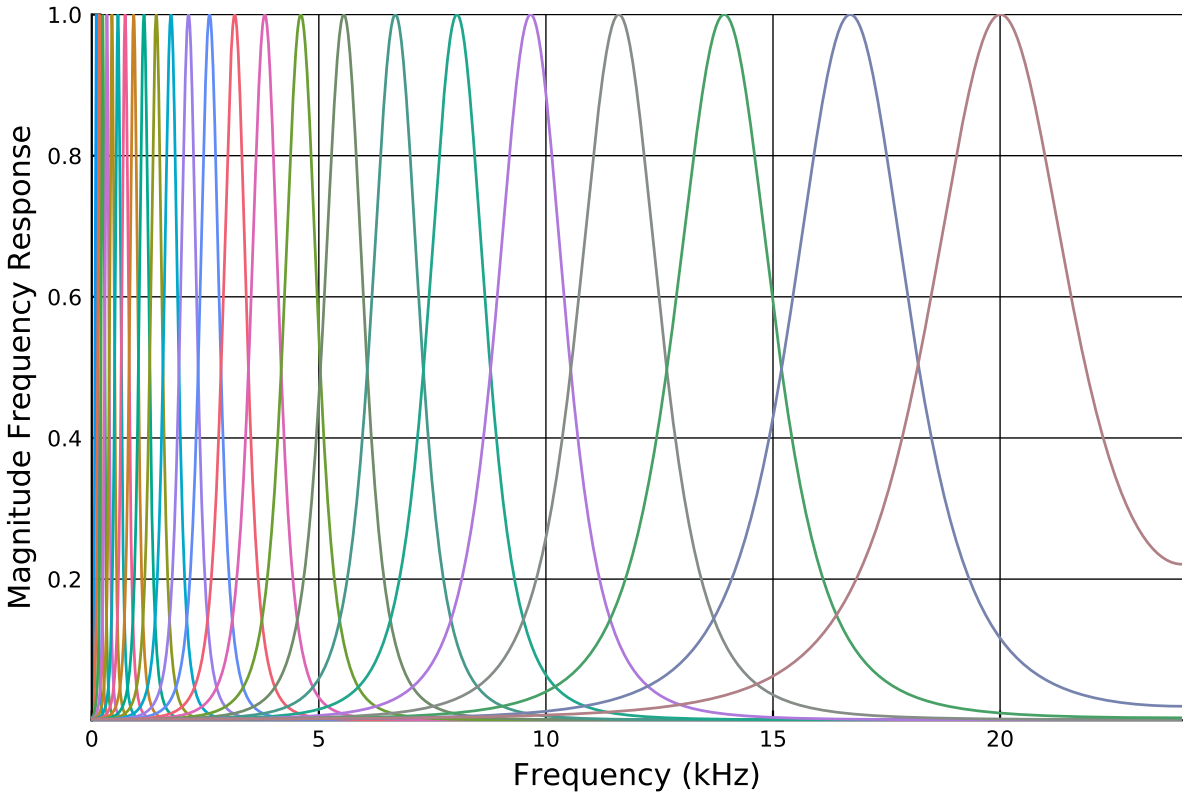

Figure 3.3: Frequency responses of 24-length gammatone filterbank outputs

feature vector. There are a wide range of feature-space transformations and machine learning algorithms used in previous work. So as to faithfully illustrate the differences between these variations, in our baseline algorithm we train models upon the MFCC [23], LSP [33] and GTFB [51] feature-space transformations using the K-nearest neighbors [21], support vector machines [29] and random forests [13] machine learning algorithms. This spread of algorithms gives us a solid foundation to be able to compare our later algorithms against the families of previous work, and also gives us an intuition on which pieces of information may be useful within the signal. Simplified results are given in Table 3.5.1, showing the accuracies at a 90% true positive rate for different pairings of learning algorithms and feature choices.

We note here first that we do not report results for simply classifying upon the raw STFT. This is due to the extremely large dimensionality of the STFT; while there are

	kNN [21]	**RF** [13]	**SVM** [29]
LSP [33]	Accuracy: **50.8%**	Accuracy: **68.7%**	Accuracy: **60.5%**
MFCC [23]	Accuracy: **89.7%**	Accuracy: **72.6%**	Accuracy: **88.8%**
GTFB [51]	Accuracy: **69.7%**	Accuracy: **90.1%**	Accuracy: **87.6%**

Table 3.2: Simplified results for the baseline algorithm

many ways to reduce that dimensionality, we choose the GTFB as the representative STFT dimensionality reduction technique. We also do not classify upon LPC coefficients directly, instead transforming to LSP before classification. This is mainly to save space; the LPC results are very similar to the LSP results and so we omit them.

Overall, the best performing of the baseline algorithms displayed here was GTFB features combined with a random forests classifier, achieving an overall accuracy of **90.1%** at a true positive rate of 90%, as shown in table 3.5.1. Other learning algorithms combined with the GTFB and MFCC features achieved similar if slightly worse performances, with k nearest neighbors and MFCCs taking second place with an accuracy of **89.7%** and support vector machines on MFCCs taking third place with an accuracy of **88.8%**. Of particular note is that line spectral pairs seem to do extremely poorly, especially when compared to previous work. After some analysis, this is due to the fact that we are not instituting any kind of energetic filter upon the incoming signal. Previous work would only classify signals of sufficient energy, automatically rejecting signals that failed to reach a certain loudness threshold. In this work, we sample truly randomly from our dataset, which includes many instances of noise and silence. These additional periods cause a large perturbation within the LSP signal domain, which causes the classification task to become much more difficult, and so all three classification methods fail to learn on the LSP dataset, despite an aggressive hyperparameter search. We further note that including an energy coefficient alongside the LSP coefficients was insufficient to aid the learning algorithms. Eliminating quiet sections from the dataset did work however, allowing the SVM and random forests learning techniques to achieve

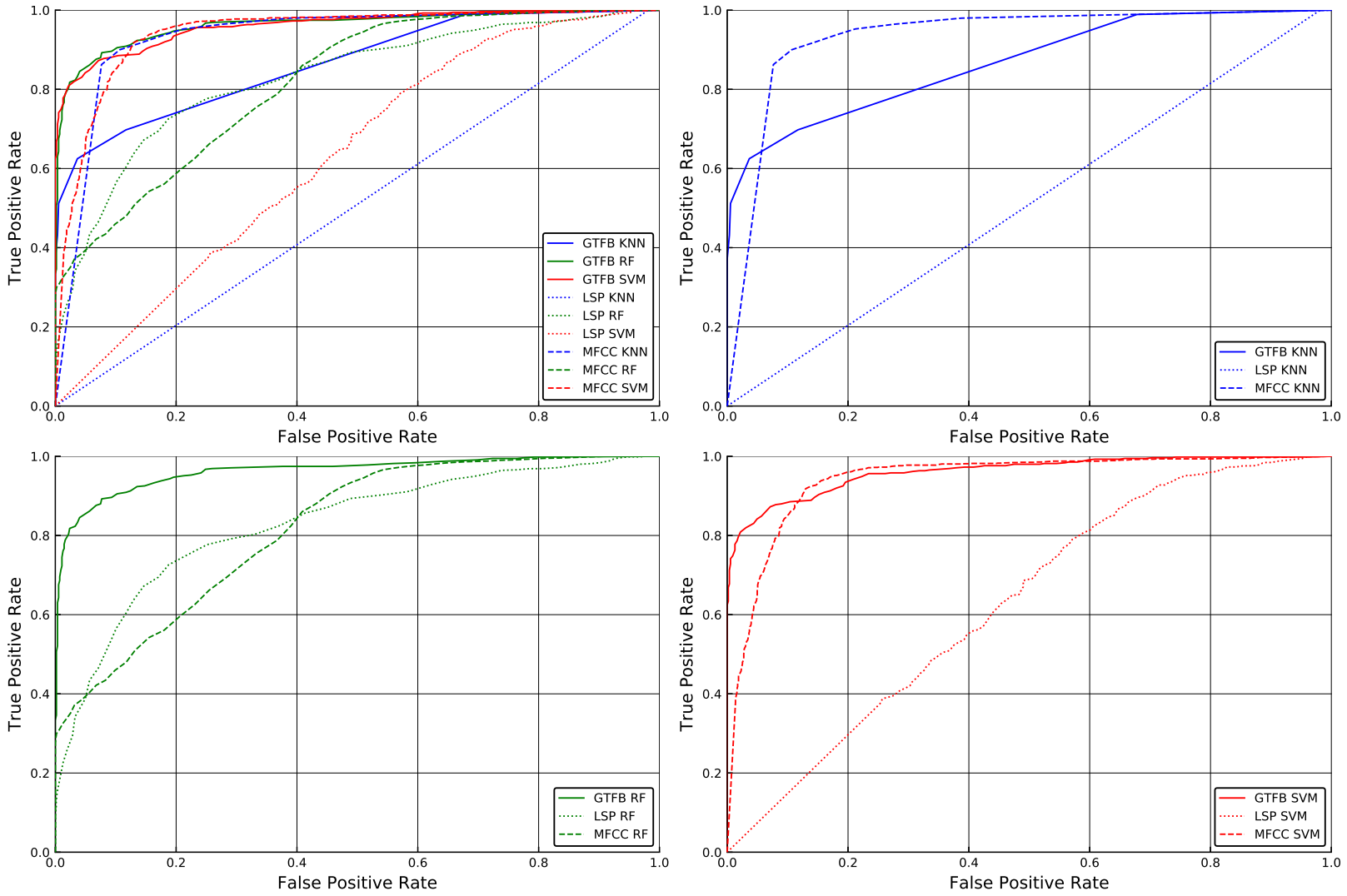

Figure 3.4: Receiver Operating Characteristic curves for all baseline methods. In top left; all methods are compared, in the other three plots the same methods are separated out by learning method. Color indicates learning method, line style indicates featurespace transformation.

overall accuracies of around 80% on those altered datasets. As these energy-thresholded LSP results are not directly comparable to our later algorithms and are not the best of all baseline algorithms, we will not discuss them further.

Figure 3.5.1 shows ROC curves for all baseline methods, with color encoding learning method, and line style encoding featurespace transformation choice. In the top left is shown all methods compared to eachother, then each learning method is split out into a separate plot in the other three quadrants of the figure. These results are slightly worse than those reported in the literature we are comparing against, however as our dataset is challenging and large, we consider these results reasonable and a good baseline to compare against.

3.5.2 Multilayer Perceptron Model

As a first attempt at applying deep learning to this problem, we employ the simplest neural network architecture available as a classifier; the multilayer perceptron. This model performs affine transformations upon the input, passing the result through a nonlinearity between each affine transformation, working its way down from the input dimensionality to an output rank of two. Similar to the architecture shown in Figure 2.4, the MLP classifier uses three layers, linearly decreasing in size from the input dimensionality (in the case of GTFB and MFCC features 456, in the case of LSP features 247) to output dimensionality (2). Between each layer is placed a LeakyReLU [44] nonlinearity with $\alpha = 0.1$. The two output neurons are fed through a softmax [40] activation function to transform arbitrary outputs into proper probabilities, and the two outputs are then taken to be *cough* vs. *non-cough* probabilities.

ROC curves for the MLP model paired with the MFCC, GTFB and LSP feature transformations are given in Figure 3.5.2. The accuracies for these three feature transformations at a true positive rate of 90% were found to be **45.1%**, **80.5%** and **76.8%**, respectively. These models do not compare very favorably with the baseline models. This is most likely due to the fact that a three-layer multilayer perceptron does not have much ability to fit highly nonlinear functions. As described in section 2.4, the main contribution of deep learning to the machine learning field is the ability to stack multiple simple nonlinearities on top of each

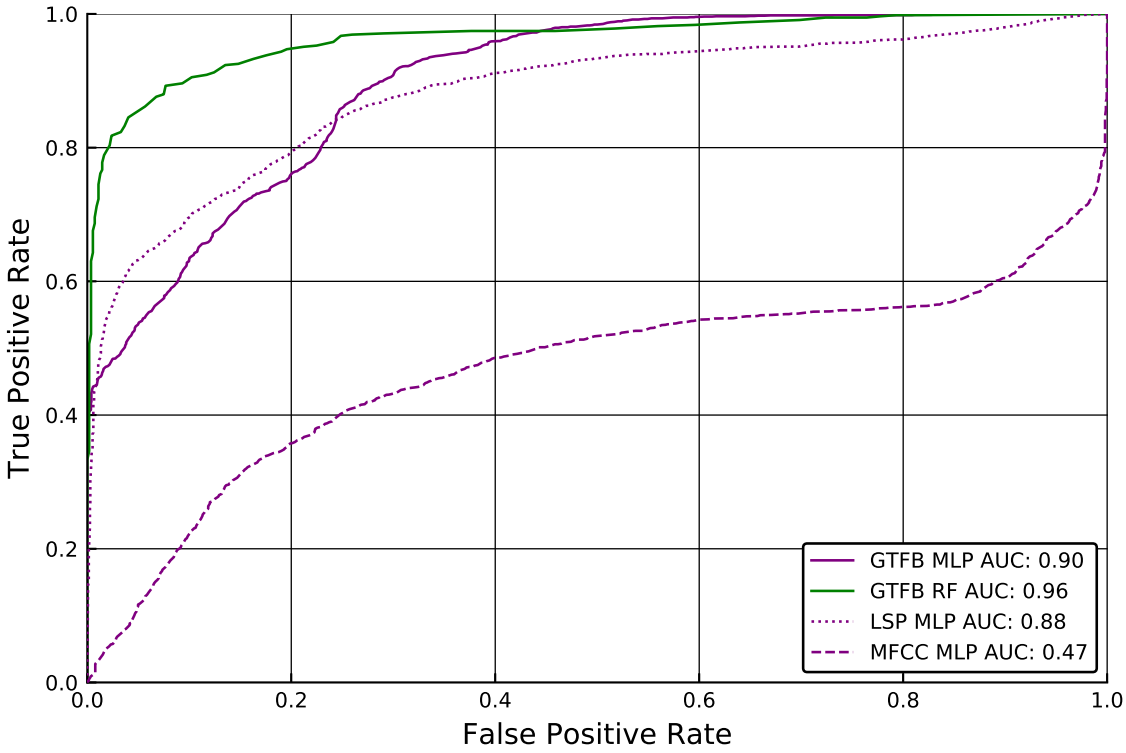

Figure 3.5: Receiver Operating Characteristic curve for a multilayer perceptron classifier paired with the MFCC, GTFB and LSP feature transformations. The best baseline method is plotted as well, for comparison.

other to learn highly nonlinear functions. This is limited by the tension between the depth of the model (as depth increases, the ability for the model to fit arbitrarily nonlinear functions increases) and the number of parameters within the model (as the number of parameters increases, the amount of data necessary to converge to a stable model also increases).

The layer sizes of the GTFB model step down from 456 to 305, to 13, to 2, resulting in a total number of parameters of **186511**. This is a large number of parameters for a dataset containing less than 5000 cough samples, and although multiple coefficients are produced from each cough sample, and the cough samples themselves are augmented through the addition of noise and random shifting, these models still must make aggressive use of

dropout and weight regularization in order to avoid overfitting. In the end, adding further layers onto the MLP does not help much as the model begins to overfit drastically, and the model flounders in its lack of data. To build deeper models and thereby learn the nonlinear function that maps from featurespace representation to cough detector, we will need to change the fundamental architecture of our model.

3.5.3 Convolutional Model

Multilayer perceptrons, while quite effective, are extremely data-hungry, as their parameter counts explode quadratically with the rank of the input feature vector. For this reason, we decided to investigate models with lower parameter counts that could more readily reap the benefits of the stacked nonlinearities of a deep learning model. Encouraged by the GTFB results, which have given promising results for MLP, random forests and SVM classifiers, we exploited the locality inherent in cough sounds when represented in time-frequency distributions by learning a convolutional model off of the outputs of the GTFB.

After feature preprocessing, the input to the machine learning model can be viewed as an image with a single channel, the result of applying the gammatone filterbank to the input audio data, broken up into overlapping time frames. An example of this decomposition is shown in Figure 3.2, where two exemplary cough signals are shown side-by-side with color mapping the energy at each point in time and frequency. Note that the cough "event" manifests itself as a "blob", which is the technical term for a localized event in the time-frequency domain.

Convolutional networks can be stacked on top of each other to find larger and more complex patterns within data. We employ stacks of batch normalization [31] (BN), Convolutional neural network (CNN) and LeakyReLU (LReLU) [44] layers to create "blocks" of convolutional kernels. Each block contains three consecutive sequences of a BN-CNN-LReLU layer grouping. We separate these blocks with max pooling layers to reduce data dimensionality, culminating in a global average pooling layer [42] and a final fully connected layer with softmax activation to output probabilities for the two classes (*cough* vs. *non-cough*).

Layer	**Output Shape**	**Parameters**	**Runtime**
Feature preprocessing (GTFB)	$24 \times 19 \times 1$	0 (0.0%)	36.9ms (13.4%)
3x BN-CNN-LReLU	$24 \times 19 \times 8$	1316 (26.2%)	153.2ms (55.6%)
Max Pooling	$12 \times 9 \times 8$	0 (0.0%)	4.7ms (1.7%)
3x BN-CNN-LReLU	$12 \times 9 \times 8$	1848 (36.7%)	64.3ms (23.3%)
Max Pooling	$6 \times 4 \times 8$	0 (0.0%)	1.0ms (0.4%)
3x BN-CNN-LReLU	$6 \times 4 \times 8$	1848 (36.7%)	15.2ms (5.5%)
Global Average Pooling	$1 \times 1 \times 8$	0 (0.0%)	0.2ms (0.1%)
Fully Connected	2	18 (0.0%)	0.1ms (0.0%)
Total		5030	275.6ms

Table 3.3: The cough detection model architecture, with learned parameter distribution and runtimes measured per-layer. All runtimes measured on a Raspberry Pi 3 B+, using a batch size of 100, equal to processing a full second of audio at once.

Throughout the network, all convolutional layers output eight channels, and all kernel sizes are 3×3.

Our models have been designed from the ground-up for usage in embedded processing environments, with low computational resources required and an emphasis on models that can be deployed onto resource constrained devices. Table 3.5.3 gives detection model timings as measured on a Raspberry Pi 3 B+. The timings were calculated using the built-in profiling mechanisms of MXNet [16]. Feature preprocessing and the first convolutional block account for the majority of the runtime requirements, adding up to nearly 70% of the total CPU time used per batch. The numbers reported in Table 3.5.3 are representative of calculations with a batch size of 100, which corresponds to processing a full second of audio at once.

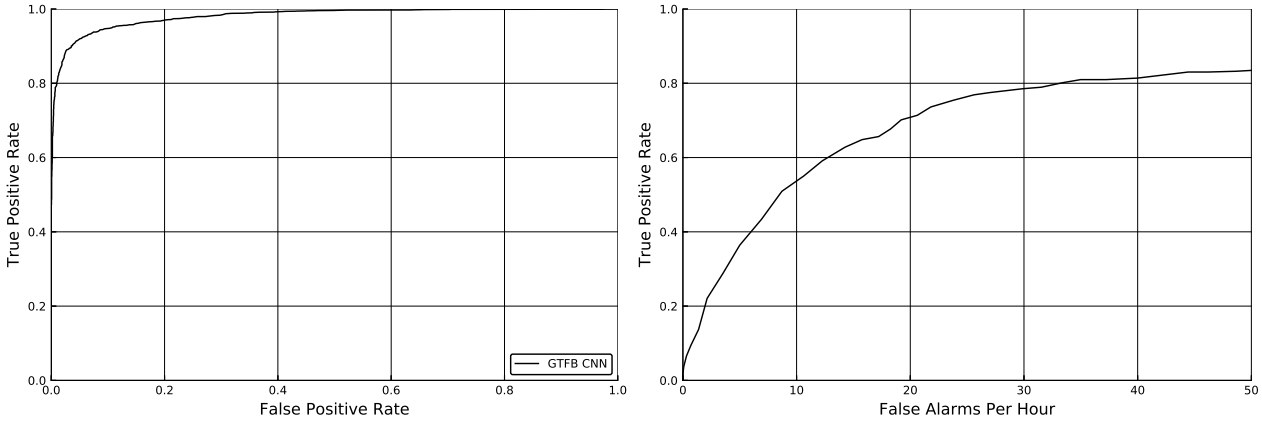

Figure 3.6: Detection performance for the convolutional model operating upon GTFB features. Left; Instantaneous Receiver Operating Characteristic curve. Right; False alarm curve.

This reduces library overhead to less than 1% of total runtime and is a realistic latency target for all applications considered within this dissertation. We highlight the low memory requirements of the trained detection model: The model consists of **12504** static parameters, (including, for example, GTFB values) and **5030** learned model parameters, yielding a total model memory footprint of approximately **70 kB**. Peak memory usage by the model during classification with a batch size of 100 remains less than **1.5 MB**, easily fitting within even the smallest embedded DSP platforms. This is in strong contrast to many previous works that depend on machine learning models such as random forests or support vector machines, which routinely require much more memory than is available in embedded devices.

Figure 3.5.3 shows an ROC curve displaying the convolutional model's performance. The accuracy of the model was found to be **93.2%**, achieving a false positive rate of less than **3.5%** for a **90%** true positive rate. This performance is calculated by running the full recordings of all files within the test set through the model and comparing to the ground truth annotations, meaning that for the purposes of evaluation, in this case the true negative condition outweighs the true positive condition by over 300×.

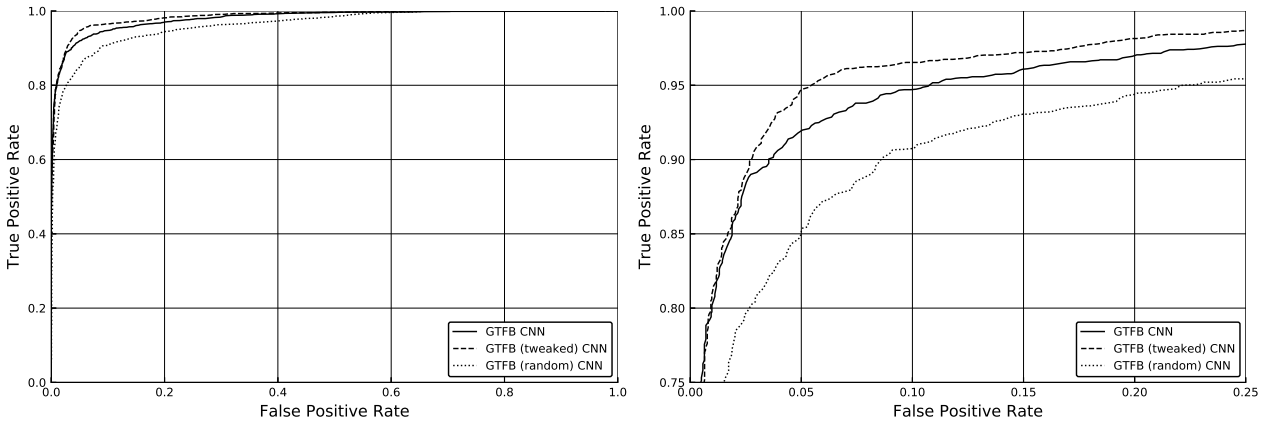

Figure 3.7: Detection performance for all three convolutional models operating upon GTFB features with varying levels of backpropagation learning allowed to change the GTFB calculation matrix. Right subplot is a zoomed version of Left.

3.5.4 Pushing Backpropagation Further Back

In our quest to use the tools of deep learning in ever greater ways, we pushed the concept of backpropagation back even farther toward the raw audio input. We represented the gammatone filterbank as a learnable matrix multiplication off of input Fourier transform magnitudes. Represented this way, the GTFB transform is merely a 24×481 real-valued matrix, (for 24 output filters, assuming a time frame length of 960 samples, and that only the positive Fourier coefficients are used). We performed two experiments, first seeding this spectral transformation matrix with GTFB values, then allowing backpropagation to tweak the values. Second, initializing the matrix with random values just like all other portions of the network. In both cases, this extra 24×481 set of parameters is learned as a shared weight applied to each incoming frame of audio to generate the temporal-spectral "image" that is fed into the convolutional network.

Results for both of these experiments are given as ROC curves in Figure 3.5.4. The figure shows that the ability to subtly change the GTFB matrix affords the "tweaked" model the ability to eke out a few more percentage points of accuracy. The overall changes made to the

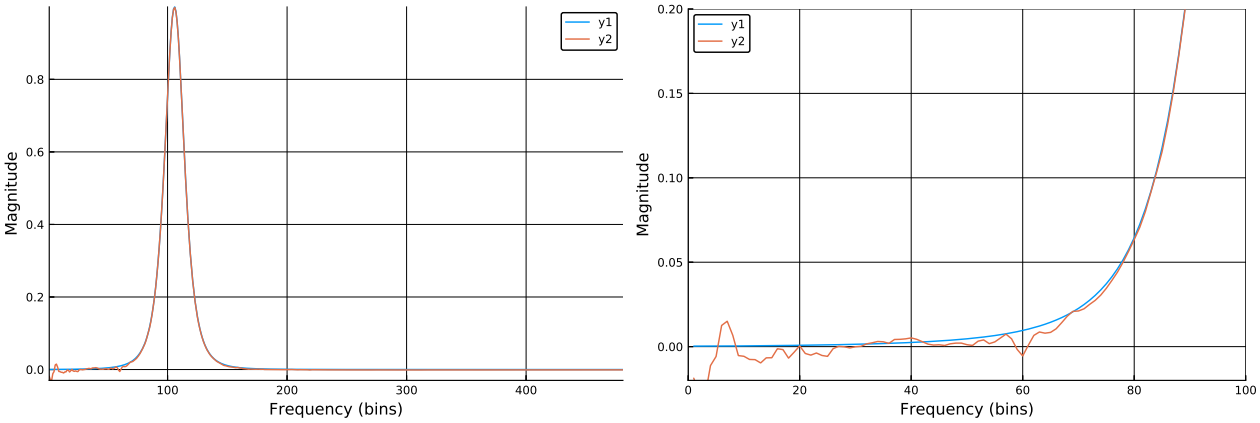

Figure 3.8: Left; GTFB filter and tweaked version plotted on top of each other. Right; the same, but zoomed into the lower-frequency portions.

filterbank are extremely minor, overall less than 1.5% deviation from the original filterbank. Figure 3.5.4 shows the filter deviations for a single exemplary filterbank. We note that most of the filter deviations seem to show lower frequencies being introduced into higher frequency bands, which seems to indicate that the model is finding correlations between low and high frequencies in cough sounds. This runs somewhat counter to the "localization" arguments espoused in section 3.5.3, leading us to grudgingly accept that nature rarely makes things easy. The reality of low and high frequency correlation is causing the filterbank abstraction to break down slightly here, however due to our introduction of backpropagation in even this portion of the network, it appears that the model is capable of dealing with this.

For the "randomized" GTFB filterbank (not really a gammatone filterbank at all, but merely a random matrix that is used to transform the spectral magnitudes fed into the network) there are unfortunately no simple answers as to what is learned. The "filters" within the filterbank are, to our eyes, very similar to the gaussian noise that they were initialized with. The network, however, is able to learn patterns within this data, as shown in Figure 3.5.4. In an effort to understand what these randomized filters are learning, we visualized a "spectrogram" similar to what is shown in Figure 3.2. The result is shown in Figure 3.5.4,

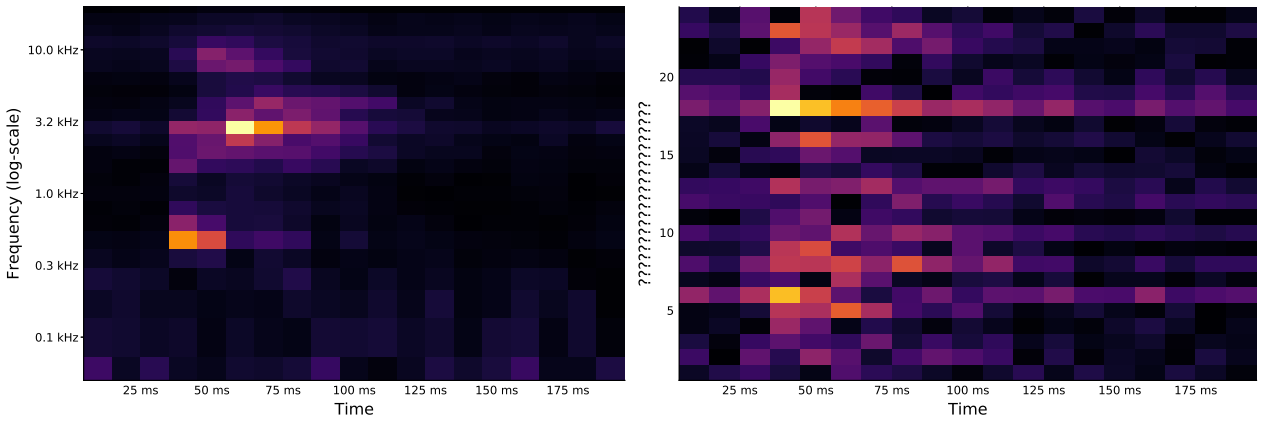

Figure 3.9: Left; GTFB spectrogram of a cough sample. Right; randomized GTFB spectrogram of the same cough sample.

where a cough sample is shown transformed through the typical GTFB transform resulting in a "GTFB spectrogram", and this is juxtaposed against the "spectrogram" that is the result of running that same cough sample through the randomized matrix after training it on cough samples. We note that the matrix does seem to have some concept of frequency locality; it has not completely scrambled the spectral energy, which forms the basis for the convolutional kernels to be able to find patterns within these images. Regardless, this model does not perform nearly as well as the vanilla GTFB or tweaked GTFB models, and so it will be ignored for now until future work can investigate methods of coercing this model toward even greater efficacy.

3.5.5 Introducing Frequency Variance

The astute reader may notice that while the fundamental proposition of a convolutional neural network is learning of shift-invariant probability distributions, our signals are only truly shift invariant along the time axis. There is no reason why frequency shift invariance should be considered a necessary property of the system, however eliminating it will increase the parameter count somewhat, decreasing the ratio of parameters to training examples.

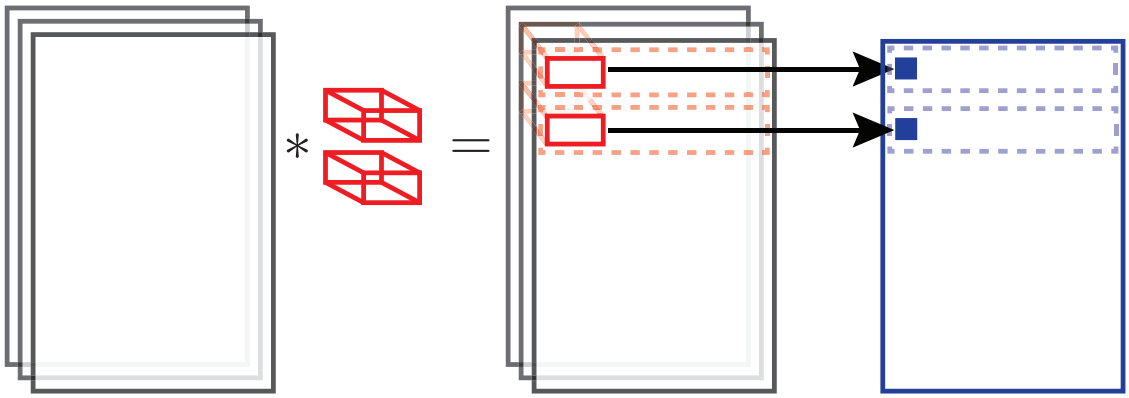

Figure 3.10: Diagram of a frequency-variant convolutional neural network (FV-CNN), graphically depicting a convolution between a 3-channel input tensor and (for convenience) two different convolutional kernels in red. The convolutional kernels are shifted across the time dimension of the input tensor, with domain and range denoted by the dashed lines. A separate convolutional kernel will be used for the next row of frequency down, and a separate convolutional kernel will be used for the next row after that, continuing until all input data is mapped to an output. The generated output plane of pixels is shown in blue.

This is because the probability distribution that the model must learn is now predicated on frequency, and thus instead of learning one 3×3 probability distribution that is shifted across all of time and frequency, it now must learn N 3×3 probability distributions that are each shifted across time only, where N is the number of filters within the gammatone filterbank. This is visualized in Figure 3.10, showing how the convolutional kernels shift only in time, and a separate kernel is used for each slice of frequency. This formulation increases the number of parameters of the model from **5030** up to **63948**, an increase of just over $12\times$.

This increase in parameters gives the model slightly more flexibility at the cost of requiring more training data to converge to a stable result. When trained upon the dataset in the same way as our other convolutional models, we find that the performance of a frequency-variant convolutional model is able to match that of the frequency-invariant model with

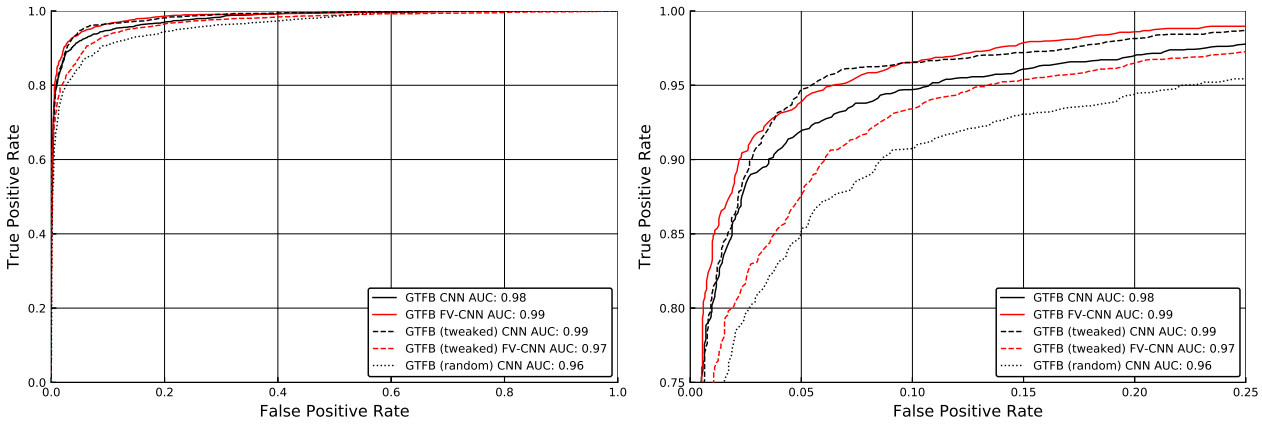

Figure 3.11: Detection performance of frequency-variant models in red, compared against all three non-frequency-variant convolutional models in black. Right subplot is a zoomed version of Left.

tweaked GTFB filters. Allowing backpropagation to tweak the GTFB filterbanks for this frequency-variant convolutional model did not increase accuracy but rather hindered it. We hypothesize this is due to the parameter count (now up to **74892**) beginning to eclipse the amount of training data available. The results for both of these experiments are shown as ROC curves in Figure 3.5.5, with black denoting frequency-invariant models, red denoting frequency-variant models, and line texture denoting how the featurespace transformation was learned (if at all).

Our interpretation of these results is that the restriction of convolutional kernels to a single location in frequency allows the kernels learned in low frequencies to differ from those learned in high frequencies, which turns out to be a competitive advantage. We experimented with both 3×3 kernels (results shown in Figure 3.5.5) as well as 3×1 kernels (patterns across time only, not in frequency; results not shown). We found that a 3×1 kernel was insufficient for proper learning; e.g. it is truly necessary to learn joint time-frequency probability distributions, and the best results were found when those joint time-frequency probability distributions were further conditioned on absolute frequency. Inspecting the changes the

backpropagation algorithm made to the filters when allowed to tweak them showed that the changes were similar to those made when using a frequency-invariant model: high frequuency bands began to couple in small, unpredictable ways with lower frequency bands.

3.5.6 Temporal Smoothing/Segmentation

The results in sections 3.5.3 and 3.5.4 refer only to instantaneous detection accuracy. There is no concept of context or temporal information beyond the 19 frames of data extending out to either side, and as with any instantaneous estimator, its noisyness through time can be greatly improved upon through temporal smoothing. We evaluated two methods for this; a simple median filter applied to the classification outputs, and a simple recurrent neural network composed of stacked gated recurrent units [17] applied to the same outputs. In all cases, the median filter was found to outperform the recurrent neural networks, and so we will refrain from delving deeper into the results in this dissertation. We do wish to note that the recurrent neural networks were not significantly worse than the median filter, and that they were not trained jointly with the convolutional model. We hypothesize that one direction for future work is to investigate whether jointly training the recurrent neural network on top of the convolutional detection model (rather than simply training the recurrent neural network on the static outputs of a previously-run detection model) would increase performance.

We used a median filter of length 9 to smooth contiguous detection results and ensure that noisy estimators did not generate spurious coughs in the presence of noise. We took this smoothed output and coalesced contiguous classification results into discrete cough events. These cough events were then directly compared to the ground truth annotations, with a single overlapping classification counted as a true positive, any other kind of detection (completely non-overlapping with a ground truth annotation, or multiple overlapping classifications) as a false positive. The lack of any kind of overlapping detected cough event for a ground truth annotation was taken as a false negative, and we did not track true negatives. These event detection rules form the basis for our analysis on "false alarms per hour", where this metric simply reports the number of false positive cough events per hour of recording.

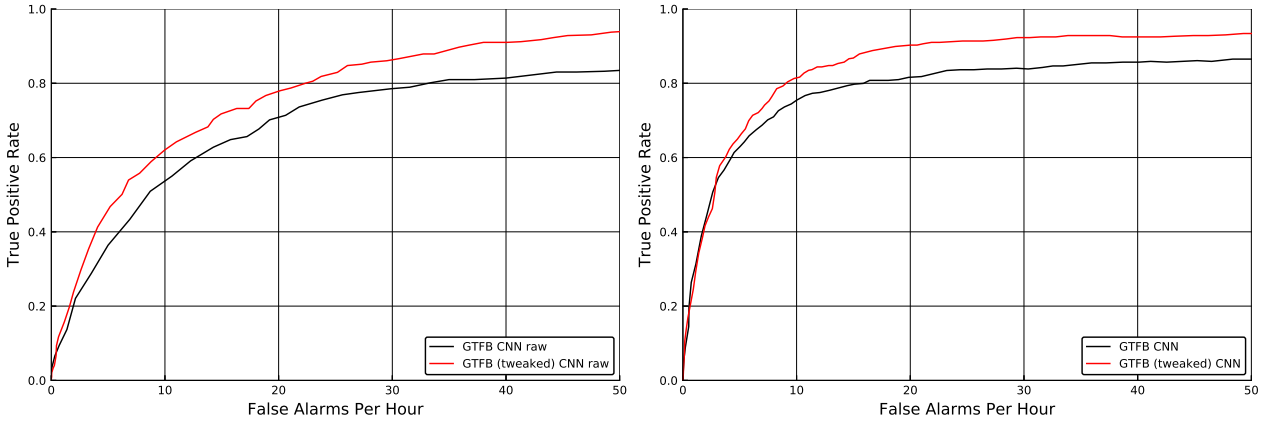

Figure 3.12: Left; False alarm curves for the convolutional model operating upon GTFB features and tweaked GTFB features. Right; False alarm curve after median filtering.

Using this methodology, we were able to obtain an **80%** true positive rate with less than **10** false alarms per hour, as measured on our testing dataset. This is shown in the right-hand side of Figure 3.5.6.

3.5.7 End-to-End Deep Learning Models

To complete the effort of pushing the deep learning paradigm ever further, we attempted to completely rid ourselves of a feature preprocessing step at all, and use an end-to-end deep learning pipeline, where every operation, from raw input signal to finished classification probabilities is differentiable and learned. While our previous experiments were met with some success, this level of deep learning was found to avoid convergence very reliably. In particular, the densely packed nature of temporal data requires extremely large amounts of training data, far beyond what we have been able to collect in this dissertation work. A promising direction of future work is to apply transfer learning to a pretrained auditory model (such as WaveNet [49]) to bend its audio-processing abilities towards cough sound analysis and see if it is able to improve upon the fundamental assumptions made when we chose to use the GTFB as our signal representation.

3.6 Cough Classification

A primary research outcome of this dissertation is to answer the question of whether information about cough sounds can be extracted in such a way as to enable automatic cough classification based upon disease. Through a collaboration with the Gates Foundation and the Desmond Tutu HIV Centre in South Africa, we have access to a dataset of cough sounds that contain many coughs from patients diagnosed with tuberculosis. We build a cough classification model and analyze the results for accuracy. Runtime performance is less of a concern for cough classification as the model need only be evaluated once a cough event has been known to occur. As the architecture of the cough classification model is very similar to the architecture of the cough detection model however, the runtime performance is very good.

3.6.1 Cough Classification Model

The classification model for classifying coughs into *tuberculosis* vs. *control* coughs is extremely similar to the detection network. The differences are that the feature preprocessing stage generates 32 filterbank outputs instead of 24, the network contains 4 stacked blocks of convolutions rather than 3, and each convolutional layer generates 6 channels instead of 8. These changes, while relatively minor, allow the classification network greater latitude in building complex patterns out of the stacks of convolutional kernels. This in turn significantly boosts the classification accuracy while effecting a relatively minor impact on runtime performance without increasing the number of parameters significantly. We note that the runtime of this model is similar to that of the detection model; however, as runtime is rarely a concern for classification models we will forgo any analysis of our cough classification model performance.

As mentioned in Section 3.1, the datasets for cough classification are highly imbalanced; due to differences across the datasets, it is necessary to take steps to prevent the classification model from simply learning the difference between the *clinical* and *ambulatory* datasets

rather than learning the difference between a *tuberculosis* cough and a *control* cough. To do this, we employ a Discriminative Adversarial Network (DAN) to take the outputs of the first convolutional "block", and attempt to classify the input data sample as either stemming from the *clinical* dataset or the *ambulatory* dataset. This is shown graphically in Figure 4.1, with the typical convolutional network classifying what type of cough was input, and a second DAN classifying the dataset from which the cough originated. The DAN takes the form of a 3-layer multilayer perceptron with linearly decreasing layer sizes and LeakyReLU activations between layers. The last layer feeds into a softmax activation that predicts the dataset for the given input cough samples. The ability of the DAN to correctly classify an input data sample is then pitted against the ability of the main network to classify an input cough. The network loss L is therefore a function of both the classifier network loss L_C, and the discriminator loss L_D as shown in the training procedure given in Algorithm 2. This addition of a second network suppresses the ability of the classification network to extract features that distinguish one dataset from the other, forcing it to instead rely on patterns that are present in both datasets. For more information on DANs, see chapter 4

3.6.2 Classification Results

As mentioned previously, the dataset our models are trained upon (summarized in table 3.1) is bifurcated; half of the data comes from a clinical setting, whereas the other half comes from an ambulatory setting. Additionally, all tuberculosis data comes from the clinical dataset, however very little of the non-tuberculosis data comes from the clinical dataset. This presents a difficult problem to the machine learning optimizer: if trained upon only clinical data it is able to learn, however with so few non-tuberculosis (referred to as *control*) cough samples, it is unable to learn a very good classifier, and if tested upon data from outside of the clinic, it performs very badly. These two conditions can be seen in the Figure 3.6.2, in the *Clinical on Clinical* and *Clinical on Both* conditions. It is painfully obvious that the *Clinical on Both* results show a detector that has learned from a very disproportionate population, and as such is classifying many more patients as having tuberculosis than should be the case.

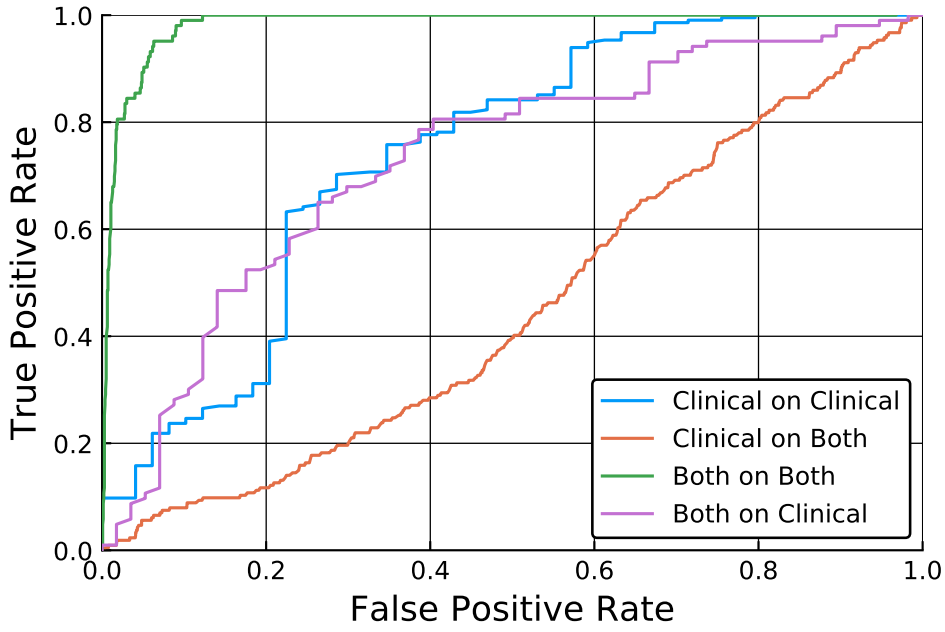

Figure 3.13: Receiver Operating Characteristic curve for the classification model in four different conditions: Trained on clinical data and tested on clinical data, trained on clinical data and tested on both datasets, trained on both datasets and tested on both datasets, and finally trained on both datasets but tested on only clinical data.

To remedy this, we attempt to introduce the ambulatory dataset's cough samples into the training set, however this presents the machine learning model with an easy escape. The model can simply classify all samples from the clinical dataset as tuberculosis samples, and thereby avoid having to learn how to differentiate cough sounds, but rather learn how to differentiate which data collection site a sample came from. This is a perfect situation in which to use a DAN, to use adversarial networks to filter out the information that could be used to learn to discriminate dataset provenance.

We applied a DAN to our model as shown in Figure 4.1, and used it to suppress dataset provenance information within the cough samples. This allowed our model to train upon both datasets, greatly increasing the model's ability to detect a control cough without harming its

ability to detect a tuberculosis cough. In particular, in Figure 3.6.2, the *Both on Both* line shows the performance of a model trained on both datasets and then tested on both, which does quite well, however the true test is to take that model and test it only on clinical data to prove that it is not, in fact, simply predicting tuberculosis for all samples within the clinical dataset. As can be seen by the *Both on Clinical* condition in 3.6.2, this is not the case. We can therefore be certain that our model has learned more about what non-tuberculosis coughs sound like, without harming its conception of what a tuberculosis cough sounds like.

A current state-of-the-art sputum testing system called the Gene Xpert is used in clinics worldwide to screen for tuberculosis in patients already identified to have some kind of pulmonary ailment. The sputum test requires the patient to cough up a sample of sputum and insert it into the machine for genetic testing. This process is time consuming and the machine itself is expensive. When tested by a third party, the Gene Xpert was found to achieve a sensitivity of **52%** with a sensitivity of **95%** [48]. We hypothesize that this cough system could therefore be used as an initial screening tool to catch users with tuberculosis in low-resource clinics that cannot afford a Gene Xpert machine (which retails for over $17,000 USD as of the time of this writing) before flagging the patients for further testing.

We note here that a tuberculosis cough tends to be quite "wet", often producing sputum in vast quantities. To contrast tuberculosis coughs with those in our control group, the majority of all non-tuberculosis cough sounds are "dry", however a large minority of the control coughs were also wet due to preexisting conditions. Additionally, not all tuberculosis coughs captured within the dataset were wet. We unfortunately do not have ground truth data on which coughs were wet and which were dry across both tuberculosis and control coughs; it would be illuminating to determine how correlated the tuberculosis cough classification model's output is with the ground truth labels of coughs based on wetness. A promising direction of future work is to collect further datasets of mixed coughs for other pulmonary ailments with distinctive cough sounds and to integrate those with these two datasets to verify that the models are able to learn subtle differences in coughs beyond simply "tuberculosis" versus "non-tuberculosis", as these models have.

Chapter 4

DISCRIMINATIVE ADVERSARIAL NETWORKS

In this chapter, we will briefly describe a new application of transfer learning toward the problem of dataset imbalance. This application is dubbed a "discriminative adversarial network" and addresses concerns against model overfitting in networks trained on disparate datasets. This fulfills one part of the research outcomes listed in section 1.2.

4.1 Dataset Imbalance

Many machine learning algorithms are not guaranteed to be globally optimal; they employ stochastic training methods that converge to a local optimum, but there is no guarantee that the converged to values are the best possible values for that model. Neural networks are a conspicuous member of this class of algorithms, convergence to suboptimal results is not a rare occurrence when training deep neural networks, and techniques to work around these issues abound. One of these techniques is to maintain a balanced dataset. The fundamental presumption behind stochastic gradient descent (as briefly described in section 2.4) is that each minibatch that is trained upon (in this dissertation, a minibatch is approximately 64 training samples, although values from 1 to 1024 are common [10]) is representative of the overall population of the training set. This assumption is an important one, as if a learning algorithm is given only negative examples for too many iterations, it is natural for the gradient descent algorithms to begin compromising the performance of the portions of the network responsible for positive examples.

This can be seen mathematically, as the optimization problem being solved (Equation 2.5, which is shown specialized for a perceptron) is gradient descent, which will always seek to minimize the loss function as much as possible. The update applied to the weights of the

model W, which we will refer to as ∂W can therefore be decomposed:

$$\partial W = \partial W^+ + \partial W^- \tag{4.1}$$

Where ∂W^+ corresponds to the component that would update weights more toward classifying inputs as a positive class, and ∂W^- corresponds to the component that would update weights more toward classifying inputs as a negative class. This decomposition is trivially accomplishable by simply performing a forward pass of only positive examples within a minibatch, calculating the gradients of the model weights with respect to those outputs, storing those gradients as ∂W^+ and doing the same for all negative examples and storing those gradients as ∂W^-. By training on minibatches with representation of both positive and negative classes, the summation of the two components will update the network along the shared subspace that the optimization algorithm has deemed is beneficial for the network across both classes. While it is true that the loss surface being optimized over is high-dimensional and extremely nonlinear (due to the many nonlinearities imposed by the depth of the network being optimized), and many local optima will exist, a fundamental assumption of gradient descent is that the model will, eventually, converge to an optimum point at which $\partial W^+ + \partial W^-$ cancel each other out and no more progress can be made. The benefit to ensuring that both ∂W^+ and ∂W^- are present in each minibatch is to ensure that each "step" taken by the optimizer moves along the direction in state space that is actually desired, whereas taking many steps wholly in the ∂W^+ direction, followed by a step wholly in the ∂W^- direction runs the risk of falling into local optima from which it is difficult for the optimizer to escape. While sufficiently large steps taken by the optimizer are always able to escape from these local optima that bias strongly toward one class, in practice it is difficult for the model training loop to detect when this happens and increase the learning rate η to deal with it.

When datasets are large and balanced (e.g. the number of training examples per class are roughly equal) simply randomly sampling from the datasets is enough to guarantee rough balancing of minibatches. This approach does not work for our use cases here, as the nature

of the data we are working with is that there is over 350× as much non-cough data as there is cough data (see section 3.1 for a breakdown of the dataset). This led us to design the minibatch balancing strategy described in section 3.3 that ensures that roughly 1/2 of all training instances within every minibatch are cough sounds. This successfully addressed the issue of balance between cough/non-cough examples when training cough detection models. For our cough classification models however, there exists another sort of imbalance: *data collection site imbalance*.

Inspecting table 3.1, it is simple to see that all tuberculosis cough samples come from the *clinical* dataset, however the great majority of non-tuberculosis coughs come from the *ambulatory* dataset. While balancing non-tuberculosis coughs against tuberculosis coughs when building a cough classifier is straightforward (we simply sample random coughs from the two subsets of coughs, as described in section 3.6) there remains a fundamental danger within our model training that the updates applied to the model weights W will be solely based upon the data collection site and not the actual cough sound itself. We confirmed that this is indeed an issue by training classifier networks on the dataset and noting that the network misclassified every single non-tuberculosis cough sample from the *clinical* dataset as a tuberculosis cough. We hypothesize this is because gradient descent is overwhelmed by the component of the weight gradients that is due to data recording site differences. This leads the model to learn to tell the difference between data recording sites rather than to tell the difference between a tuberculosis cough and a non-tuberculosis cough.

Viewed in this manner, the solution is simple: we must disincentivize the gradient descent algorithm from learning to cue off of the wrong pieces of information. To do so, we must alter the optimization to pay less attention to gradients common to dataset differences, and more attention to gradients common to class differences. While, ideally, there would be no dataset differences and this would not be a problem, the reality is that dataset collection is difficult, and building more robust learning methods is desirable.

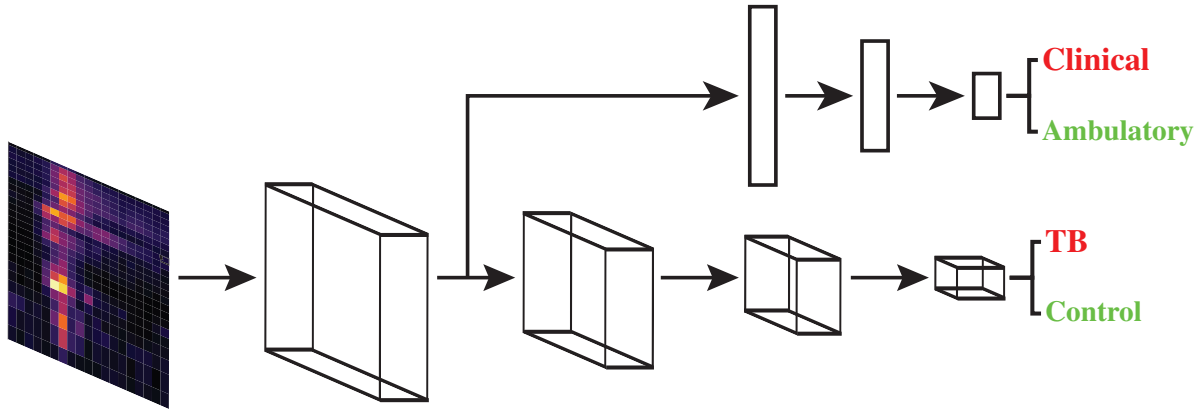

Figure 4.1: Cough classification architecture with DAN and CNN architectures visualized. Along the bottom lies the convolutional classification network that classifies cough type, with the outputs from the first layer feeding into a multilayer perceptron that classifies dataset source.

4.2 Applying Adversarial Networks to Dataset Imbalance

The fundamental proposition of adversarial networks is that it is possible to "pit" one network against another in order to increase performance of the second network. In the case of generative adversarial networks (GANs) training examples are synthesized by a network in order to cause misclassification within a second classifier network. We take this same proposition and bend it toward training models that are "blind" to certain attributes of signals; in particular we need to reduce the effect that imbalanced datasets have upon model training.

To do so, in addition to our typical convolutional classification network that seeks to classify coughs as *TB* or *control*, we set up a discriminator network that seeks to classify an audio recording as *clinical* or *ambulatory*. By jointly training both networks, we are able to learn to classify cough sounds while penalizing the model for being able to distinguish which dataset a particular sample was collected from. Figure 4.1 shows the adversarial architecture used in section 3.6 graphically. Along the bottom of the figure is shown the convolutional

ALGORITHM 2: Classification and DAN training procedure

Input: Sample X with Label Y from dataset S, learning rate η_t, DAN strength parameter λ.

Output: Updated weights W_D and W_C of discriminator and classifier networks.

$\hat{Y}, \hat{S} = \text{Forward}(X)$;

`// Update Discriminator`

$L_D = \text{SoftmaxCrossEntropy}(S, \hat{S})$;

$\frac{\partial L_D}{\partial W_D} = \text{Backward}(L_D, W_D)$;

${W_D}^{t+1} = \text{UpdateParameters}(W_D, \frac{\partial L_D}{\partial W_D}, \eta_t)$;

`// Update Classifier`

$L_C = \text{SoftmaxCrossEntropy}(Y, \hat{Y})$;

$L = L_C - \lambda L_D$;

$\frac{\partial L}{\partial W_C} = \text{Backward}(L, W_C)$;

${W_C}^{t+1} = \text{UpdateParameters}(W_C, \frac{\partial L}{\partial W_C}, \eta_t)$;

classification network, while along the top is shown the discriminative portion of the network.

Of particular note within Figure 4.1 is that there is a small section of the overall network that is shared between the two classification pipelines. The first block of convolutional layers (the second row in table 3.5.3) is shared by both networks, and it is within this block of parameters that the DAN has its effect. Because there are now two networks imposing their respective gradients upon the weights of the first block of layers, we can decompose the gradients upon that first layer as follows:

$$\partial W = \partial W^{\text{TB}} + \partial W^{\text{Control}} + \partial W^{\text{Clinical}} + \partial W^{\text{Ambulatory}} \tag{4.2}$$

The purpose of a DAN is to learn a TB/control classifier that is able to perform well while the clinical/ambulatory discriminator is unable to accurately determine the source dataset of a sample. Algorithm 2 shows the forward/backward propagation recipe to utilize the DAN alongside a classifier network. In summary, first we pass a minibatch of training data through the model, resulting in estimates from the classifier network ($\hat{Y}$) and the discriminator network ($\hat{S}$). The discriminator network then takes a single optimizer step,

updating the discriminator weights W_D, which explicitly do *not* contain the shared weights within the first layer. This update allows the discriminator network to improve its ability to extract information from the output of the first block of layers, attempting to estimate which data collection site this sample was taken from (symbolized as S within algorithm 2). Next, the classifier is updated, however it is not simply the loss from the classifier network (L_C) that is used; the loss from the discriminator network (L_D) is introduced as well, with a mixing factor of λ. The set of parameters being updated by this combination of losses (W_C) contains the full classification network, including the shared first block of operations. The loss from the discriminator network L_D will therefore attribute gradients only to that shared first block. The rest of the classifier network is effectively trained normally; it is only the gradients of that first block of shared operations that are modified by this training recipe.

Returning our attention to equation 4.2, we are able to rephrase the gradient update of the first block of operations as:

$$\partial W = \partial W^{\text{classifier}} - \lambda W^{\text{discriminator}} \tag{4.3}$$

The purpose of adding the second term to this gradient is to attempt to "cancel out" a subspace of the weights W. To be concrete: we wish to eliminate the subspace of W that allows information about which data collection site a sample originated from to flow through the rest of the network. If the weights W can be made to successfully filter that information out while still retaining the information necessary to classify a sample as TB or control, then our optimization process has succeeded. The method by which this is done is to construct a network that extracts the clinical vs. ambulatory information from the input data. The gradients being pushed backward through the discriminative network onto the shared first block represent the network's best guess as to what filters must be available within that first block in order to correctly discriminate what data collection site a sample was recorded at. By multiplying by $-\lambda$, the gradients we add on actively suppress the subspaces within the filters with a tunable strength.

4.2.1 A Linear Example

To take a linear example, let us imagine that a fully-connected network is performing a classification task upon an input vector from a bifurcated dataset similar to the tuberculosis dataset. There are significant differences between the signals taken from the two datasets; in this example, let us assume that the previous layers of a processing network (the result of which we refer to as x) contains a strong marker for dataset provenance. In this example, we assume that the last element within x when a data instance originates from dataset collection site A is always very large, whereas the last element from each vector when the data instance originates from dataset collection site B is always very small. A visualization of this is given in Figure 4.2, where an example x_A and x_B (representing input that originates from from data collection sites A and B, respectively) are shown. The last element of x_A is highlighted in red, showing a very clear marker that can be used to discriminate which dataset a particular x vector originates from. Let us further imagine that the true classes are highly correlated with the data collection site; positive examples come from data collection site A approximately 80% of the time. We concern ourselves with only the last classification layer, to illustrate how gradients might interact to effect the input x.

Formally, let us take our classification model to be:

$$\hat{y} = \text{softmax}\left(Wf(x)\right) \tag{4.4}$$
$$x \in \mathbb{R}^{10}, \quad W \in \mathbb{R}^{2\times 10}$$

Where x is our input vector, itself an output from upstream layers in the network, $\hat{y}$ represents the estimated output class probabilities, softmax$(\cdot)$ is the softmax activation function for rescaling arbitrary values into proper class probabilities, and W is the matrix of weights for the linear classifier. For our purposes, we will ignore the softmax$(\cdot)$, and view this as merely a linear operation, assuming that we know how to backpropagate losses through the softmax function.

If we take the gradients of W, we can clearly see that ∂W will be be composed of two separate pieces, the "true" classification gradients that will learn filters to extract information

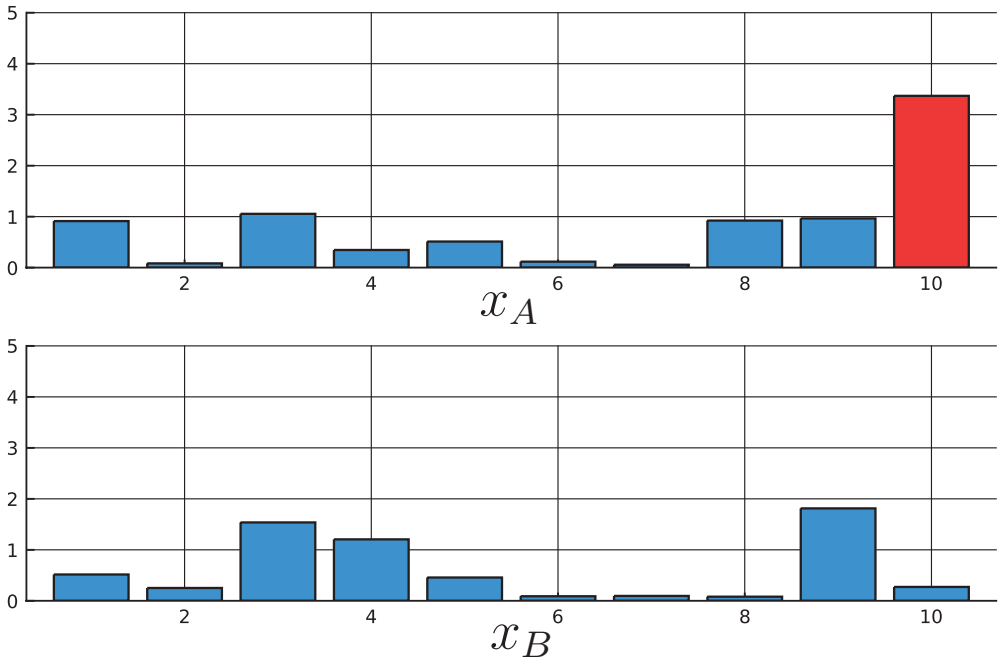

Figure 4.2: Example inputs to a linear classifier from two

from the first nine elements of x, and the "false" gradient that comes from that last element of x:

$$\partial W = \partial W^{\text{classifier}} + \partial W^{\text{AB}} \tag{4.5}$$

For the purposes of simplicity, we will assume that these two vectors are entirely orthogonal in that the last element of $\partial W^{\text{classifier}}$ is zero, and the first nine elements of ∂W^{AB} are zero. This implies that there is no information within the last element of x that has any bearing upon the true classification task; it is based upon data collection site (which itself is highly correlated with the true classification labels). By decomposing the gradients in such a way, we have an intuitive understanding of what a properly generalized model should do; ignore the last element of x and pay attention to the subtler patterns within the first nine elements of x, as the last element of x will be correct only 80% of the time.

The reason this does not happen automatically is simple; the loss function surface that

gradient descent follows can be dominated by the large values given by that last element of x. The gradients must be separated and the ∂W^{AB} portion nullified for the learning algorithm to pay attention only to the "true" information available for classification.

In order to learn this separation of gradients, we will set up a second linear discriminator network, which we will denote with D, which (in this example) is simply another 10×2 matrix that is intended to discriminate which data collection site x came from. In this example, it is plain to see that the gradients upon the last element of x induced by D will of course be very large. The gradients upon the last element of x induced by W will also be large, but in this example they will be (on average) 80% the size of the gradients imposed by D.

Within this example, we assume that x is the output from a previous layer (in our larger examples, x would represent the output of the shared first block of convolutional layers). The purpose of this formulation is to apply gradients upon x to suppress the information being unwittingly transmitted through x, overwhelming the gradient descent optimizer with deceptive gradients. By applying the update rule from **??**, we subtract out a gradient that is more fully correlated with the dataset influence than the classifier gradient. In effect, we are able to isolate the effect of the dataset-correlated independent variable and subtract that out of the gradients being computed by the classifier.

This is, of course, a simplified view of reality. Exact subtraction does not typically work, and iterative methods (using λ as a suppression factor) is necessary. The value of λ controls how much to warp the error surface by subtracting the discriminative loss from the overall classification loss. In our experiments, we have found this to be tightly coupled with the relative sizes of the gradients, which is a difficult quantity to foresee. In general, small values of λ such as 0.1, coupled with a small learning rate have yielded the best results on real-world datasets such as our cough dataset. A synthetic example is given in 4.3, that shows example data that, when trained with a DAN, is able to achieve higher accuracy than would otherwise be possible using naive learning methods.

4.2.2 Training Considerations

When training a DAN, it is important to avoid several practical pitfalls:

- *Mirror-architectures* The classifier and discriminator networks should not be exact copies of eachother. In cases where the classifier and discriminator networks are predicting the same values 80% of the time, the networks can converge in ways that cause the gradients to be very similar a large proportion of the time. This in turn causes the subtraction $\partial W^{\text{classifier}} - \lambda W^{\text{discriminator}}$ to do nothing but reduce the magnitude of the update to the weights. This by itself would not be a large problem, however especially for larger values of λ it is easy to run into machine precision issues.

  For these reasons, it is recommended to design the DAN to be of a different style of architecture than the classifier network. As an example, in Figure 4.1 is demonstrated a convolutional network that uses a multilayer perceptron-based discriminator. By architecting it in this way, we have made it much more difficult for the gradients to align and cause these precision issues.

- *Gradient flip-flops* As these networks are trained in opposition to each other, (the discriminator attempts to correctly discern dataset provenance, the classifier's update step actively prevents this by changing the weights of the shared network section) it is entirely possible to observe wildly fluctuating network performance. This is indicative of a learning rate that is too high. The discriminative network may successfully learn to take advantage of one manifestation of dataset provenance within the input signal x, and the classifier's update step may then over-correct by subtracting too much from a particular weight. The DAN is then free to simply flip its weights in response, and the network is left in more or less the same state it was at before. For this reason, we recommend training with lower learning rates when training a DAN than would otherwise be used, to allow the discriminative and classification networks to converge both separately and jointly, with a minimum of oscillation in network state space.

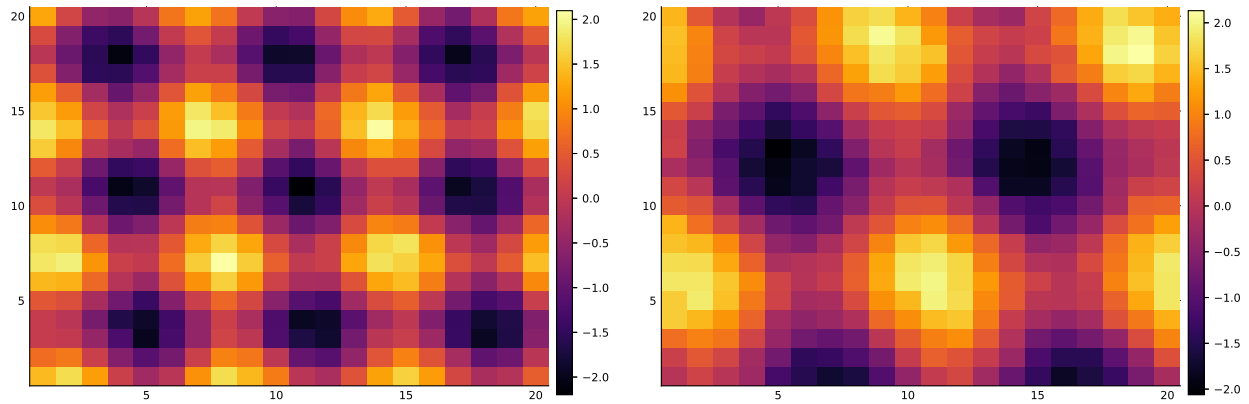

Figure 4.3: Synthetic data examples used to train a convolutional neural network. 20×20 images are generated that contain random patterns with random perturbations. Left; a "fast" oscillating pattern. Right; a "slow" oscillating pattern.

4.3 A Synthetic Example

To give a synthetic example to better understand the mechanics of discriminative adversarial networks, we will generate a dataset with what we will call "correlated corruption". We generate random images containing a "fast" oscillating pattern or a "slow" oscillating pattern, along with random perturbations including added noise, skew, and oscillation rate. Examples of these images are shown in Figure 4.3.

Tightly correlated with the class of the image (e.g. "fast" versus "slow") is a corruption that increases the top half of the image by 100, dwarfing the typical perturbations. Figure 4.4 shows a comparison of corrupted and uncorrupted "fast" oscillating patterns. Note that the dynamic range given by the colorbar on the right of Figure 4.4 is much wider in the corrupted example.

We setup the data generation such that 80% of all "fast" oscillation instances are corrupted, and 20% of all "slow" oscillation instances are corrupted. This gives us a toy dataset to experiment with, to evaluate how easily learning algorithms get caught in the local minima afforded by networks that take advantage of the corruption to win a large reduction in loss.

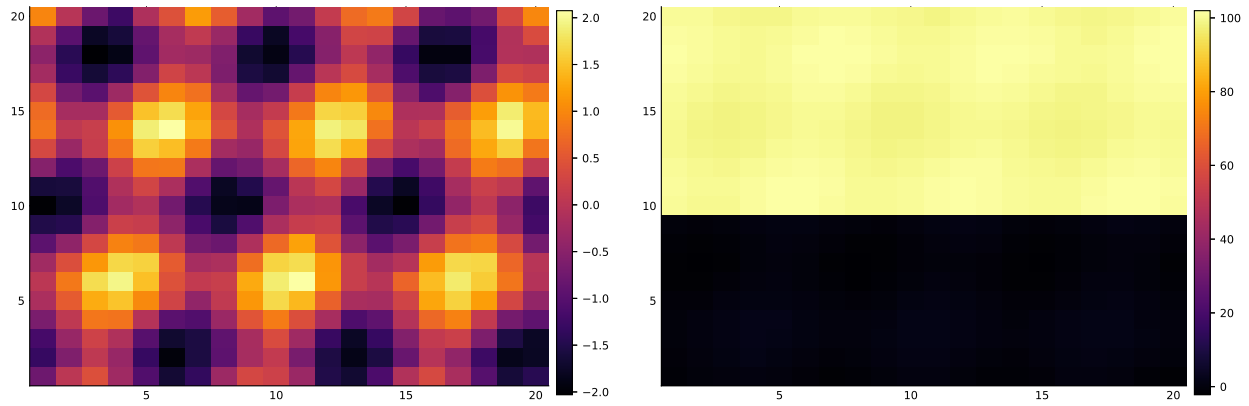

Figure 4.4: Left; an uncorrupted "fast" oscillating pattern. Right; a corrupted "fast" oscillating pattern.

We freely admit the inadequacies of this synthetic dataset; the local minima can be avoided through multiple alternative methods such as using a more intelligent optimizer (ADAM [35] is able to reliably escape the local minima) or even just low learning rates and a little luck in initial weight distribution. Regardless, this simple dataset is sufficient to illustrate the mechanics of discriminative adversarial networks, and so we forge ahead.

4.3.1 Model Setup

We develop a convolutional architecture very similar to our cough classification model, complete with a multilayer perceptron-based discriminator. Table 4.3.1 gives a listing of the convolutional side of the architecture. Our parameter choices for the convolutional model are much simpler than the cough classification model. Only 2 channels, a limited number of convolutional layers, and a reduced input dimensionality all serve to keep the complexity of the model low for easy experimentation.

Just as in the cough classification architecture shown in Figure 4.1, we attach a multilayer perceptron to the outputs of the first "block" of convolutional layers. In this synthetic example, we build a two-layer MLP off of that first block of shared weights, and train it exactly as specified in 2. Our training hyperparameters are to define one epoch as 1000

Layer	**Output Shape**	**Parameters**
3x BN-CNN-LReLU	$20 \times 20 \times 2$	106 (37.9%)
Max Pooling	$10 \times 10 \times 2$	0 (0.0%)
2x BN-CNN-LReLU	$10 \times 10 \times 2$	84 (30.0%)
Max Pooling	$5 \times 5 \times 2$	0 (0.0%)
2x BN-CNN-LReLU	$5 \times 5 \times 2$	84 (30.0%)
Global Average Pooling	$1 \times 1 \times 2$	0 (0.0%)
Fully Connected	2	6 (2.1%)
Total		280

Table 4.1: The synthetic DAN evaluation architecture. Very similar to the cough classification model, except smaller.

minibatches of size 32, and to train for one epoch with learning rate $\eta = 0.01$, and a further epoch with learning rate $\eta = 0.001$.

4.3.2 Synthetic Results

When training the model with $\lambda = 0$ (equivalent to a purely convolutional classifier with no discriminator network at all), we obtain the expected result of classifying at 80% accuracy. When comparing the network predictions against corrupted/non-corrupted labels instead of fast/slow oscillation labels, we discover that the network has indeed fallen prey to the large gradients and local minima imposed by the large corruptions of data. The "accuracy" of the network's outputs, when compared to the corrupted/non-corrupted labels is 100%, whereas the "accuracy" of of the network's outputs when compared to the true labels is stuck at the correlation factor of 80%.

To engage the DAN, we begin increasing the λ parameter so as to warp the cost function of the network's classification task. We performed a search over λ values, with smaller values having no effect (overall network performance being untouched) until λ reached values close to 0.5, at which point network convergence started to become more erratic. We continued to raise the λ parameter until we reached a value of 3.0, at which point the network began to reliably converge to **99.9%** accuracy. This consequently means that the accuracy of the network's output when compared to the corrupted/non-corrupted labels has dropped to 80%, as expected. This value of λ is much higher than what was used in the cough classification work. We hypothesize this is due to the fact that the gradients in the synthetic case are due to a corruption that is many times larger in magnitude than the signal being analyzed, whereas in the case of cough classification, the corrupting signal magnitudes are much closer to equal. In essence, the cost function in this synthetic example requires much more warping than in the cough classification task.

As this method is attempting to "dodge" local minima, and given that neural networks are randomly initialized, this solution is similarly stochastic in nature. It is not unusual for this technique to fail some percentage of the time, although in our experiments good choices of λ help. This is reminiscent of the struggles that multilayer perceptrons faced before the advent of Glorot [27] initialization greatly increased the reliability of convergence to a good minimum. While we have determined some few elements necesasry to increase the likelihood of convergence, future investigation is needed to determine the ingredients to maximize the probability of success when training a DAN.

4.4 Expanding DANs to other problems

The basic idea of a discriminative adversarial network as a tool to warp the error function of a machine learning model is applicable far beyond simply cough sounds or even imbalanced datasets. It is applicable within any context where information should be stripped out of a network and that information is available as a part of the "ground truth" annotations within the dataset. As an example similar to what we have already seen, one could imagine

a dataset of images collected from patients attempting to classify a health condition from pictures of their skin. Many confounding factors could exist within this dataset, including lighting conditions, camera manufacture, motion blur from unsteady data collection, etc... While such confounding factors could be dealt with through careful experimental design, perhaps the most important to be robust against would be skin tone. Many deep learning systems have been shown to be "biased" by choice of dataset and implicit assumptions made by the researchers designing such systems [14, 36]. DANs are a natural tool for attempting to fight this kind of bias within deep learning systems. A skin health classification system that may make mistakes due to the skin tone of the user could be made more resilient against such issues by building a DAN made to explicitly determine the skin tone of the user. As is the case with DANs, the gradients from this discriminator would then be used to suppress any information used to make that classification, such that the primary classifier network would become less sensitive to such variations in input.

As another example, one could imagine a reinforcement learning system that has a particularly difficult to control environment, and those environmental factors somehow end up disrupting or degrading its ability to learn. For instance a self-driving vehicle that learns to be more cautious around blue vehicles because a large part of its test set had aggressive drivers that just happened to use a blue sedan. The color of the car should not cause an autonomous vehicle to change its behavior, and so DANs offer an easy way to turn ground truth labels (these training recordings contain the aggressive driver) into gradients (these recordings create these gradients upon the computer vision system) into altered models (this model is no longer capable of telling the difference between the blue training car and the red training car). This shows most clearly the power of DANs: to build links between undesirable signal attributes and ground truth labels, which can then be used to suppress those undesirable signal attributes within later processing stages.

Chapter 5

CONCLUSION

In conclusion, we have given in this dissertation an overview of cough sound processing work and proposed novel methods to both make use of domain-specific knowledge as it can be applied to these signals, as well as explored some applications of deep learning in pushing these signal processing models beyond what they would otherwise be capable of. In particular, we investigated the inherent tension between the attraction of using deep learning and backpropagation to learn ever more complex nonlinear mappings for classification and detection versus the reality of imperfect and limited datasets and computational power. We began with a comparison of baseline methods, using a variety of featurespace transformations and learning methods taken from the relevant related works. We showed that for cough detection, the best methods clustered together in terms of performance, with two featurespace transformations (GTFB and MFCC) giving good results. We hypothesized that the limiting factor was therefore the learning method stacked on top of the featurespace transformation and so began building neural network models to learn the complex nonlinear relationships between input features and output cough detection probabilities.

Our first efforts failed to produce real fruit and indeed were much worse than the original learning methods. We determined that this was due to the fact that our deep learning models were exploding in learned parameter count and thus limited our viable model depth, due to the rapidly increasing number of parameters. To combat this, we exploited the locality of the GTFB transformation, and began applying convolutional networks as a highly parameter-restricted pattern recognizer. This allowed us to build significantly deeper models, which in turn enabled the learning system stacked on top of GTFB features to achieve much higher detection accuracy than the baseline methods. This validated our initial hypothesis that

the underlying featurespace transformation retained more information than the traditional learning systems were extracting. This implies that the learning methods used in previous works (such as support vector machines, random forests, K nearest neighbors, etc...) were not capable of fitting themselves to the task at hand well enough to extract all necessary information from the input features. This directly supports the thesis statement of this dissertation, demonstrating that the deep learning models we learned have been able to extract more information about cough sounds than the previous works.

Unsatisfied with this result, we next began pushing backpropagation further back, learning upon the GTFB filter functions themselves. This increased the number of parameters being learned but gave the model an ability to experiment a bit more in featurespace to try and determine whether there was more information that could be extracted by altering the featurespace transformation while holding the overall architecture of the rest of the model steady. The results showed that yes, indeed, further improvements could be made by allowing the filterbank to be tweaked by backpropagation, although their values did not change significantly from their original filters. We see two possible reasons for this: the first, and least likely, is that the gammatone filterbanks are an optimal featurespace transformation for this particular model architecfture and task. The second, and more likely, is that there is not enough data to push the model out of the local minimum it begins inside of when initialized with the GTFB filterbank.

To test these hypotheses, we experimented with randomly initializing the featurespace transformation matrix and training that toward some kind of filterbank transformation. The results showed that the system does not gravitate toward a gammatone-filterbank-like system, however neither does it attain the same accuracy as either the plain GTFB transformation or the "tweaked" GTFB transformation. This speaks directly to one of the fundamental axes of this dissertation; that in the presence of limited datasets, domain-specific knowledge can help a learned system bootstrap itself closer to an optimal result than brute-force learning could otherwise accomplish. We hypothesize that with a larger dataset, the randomly-initialized featurespace transformation would be able to converge to a state that

would meet or exceed the performance of the other two methods.

We reanalyzed our assumption carried over from the world of computer vision that assumed shift invariance in both dimensions of our input data, and created a frequency-variant convolutional model that was able to perform slightly better than the vanilla convolutional model. Our analysis of this is that despite requiring more time and data to train due to the increased parameter counts, the frequency-variant models show significant promise. We attempted to combine the GTFB tweaking through backpropagation and frequency-variant models, however the number of parameters rose too high and the model failed to converge to an optimal result.

We also developed a tool to deal with the fact that the datasets used by this work were sub-optimal, primarily due to the correlation of dataset provenance and cough class. Discriminative adversarial networks, as an extension of the adversarial learning field have applicability far beyond the world of cough classification, and can be employed anywhere a confounding variable might cause issues within a machine learning model. We showed their use on our bifurcated cough classification dataset, being used to ignore the acoustic differences between data samples collected from the two dataset locations, and gave an example on synthetic data showing the failure modes of the DANs and how to train them in the presence of challenging data. We furthermore expanded on the applicability of this learning technique to other problem domains, and showed the general usefulness of the DAN technique in selecting what pieces of the input a deep learning model should attend to.

Finally, a real-time cough detector was deployed onto a Raspberry Pi-based system and empirical measurements were taken to ensure that these models lived up to their promise in terms of memory and computational resource limits. Our results showed that the models were more than capable of being run at greater than realtime speeds on very limited hardware, and that architectures similar to this could be deployed to a very broad class of devices. We hypothesize that there exist still many opportunities for further performance enhancements to continue to reduce computational load and power requirements, however the targets for such work are very application-dependent.

5.1 Limitations of current work

We identify here a few key limitations of the work presented within this dissertation.

- *Language model*: Comparing our cough detection and classification models to the large body of work that has been developed for automatic speech recognition, there is a conspicuous absence of a "language model": the piece of a learning system that is able to make predictions of future events given knowledge of past events. While we briefly mentioned that attempts to create an RNN-based temporal smoother were unsuccessful, we believe that a system similar in spirit to those investigated here should be able to have a significant effect on cough detection accuracy. This belief comes from the observation that coughs are often bunched together and even the inter-cough timing can often be predicted, so a layer of temporal memory more intelligent than a median filter (such as an RNN or an HMM) should be able to extract that information.

- *DAN convergence*: The convergence of discriminative adversarial networks appears to be highly stochastic in nature. This is likely a function of the ill-conditioning that is forced upon the gradients backpropagating through the two models and being forced into opposition with eachother in the shared sections of the model. In some cases, DANs may be unable to learn anything useful because the error function is unable to be warped into a shape that allows gradient descent to find a superior minimum. In order for DANs to truly become a powerful tool, it will be necessary to better understand the nature of this lack of convergence and to develop tools that can better guide and shape the application of DANs to real-world problems.

- *Confounding variables within datasets*: The datasets used within this dissertation, while much better than any public dataset to our knowledge, suffer from various forms of imbalance. As repeatedly mentioned throughout, there is the problem of tuberculosis coughs being very tightly correlated with data collection location. It is also important to note that while many of the non-tuberculosis coughs are from sputum-producing

pulmonary ailments, we have not quantified exactly how many of the control coughs are "wet". Classifying a cough sound as "wet" vs. "dry" is itself a non-trivial task that would require a consensus vote between experts, and without this it is difficult to quantify the effect that the "wet" vs. "dry" confounding variable is having upon the tuberculosis classification accuracy.

5.2 Future Work

We identify here four main thrusts of future work: increasing cough classification performance, broadening classification targets, variable-length inputs and further improving model runtime performance.

- *Increasing cough classification performance*: The cough classification results contained within this dissertation could be improved by reducing the number of confounding factors within the dataset. There remain many sources of variance within the cough sounds (stage of tuberculosis infection, amount of sputum being produced per cough, exactly which pulmonary ailments are represented within the control coughs, level of background noise) that currently amortized across all results. It would be very illuminating to examine model performance and training behavior when these sources of variance within the cough signals are removed, and would likely allow identification of confounding factors that signficiantly contribute to the false positives of the cough classification algorithm. We expect this analysis to result in findings such as that extremely wet coughs from late-stage TB are more easily classified than early-stage TB where sputum production is not as intense, or that the loud background noise from fans in the cough collection chamber significantly reduce classification accuracy. From these findings, a new dataset collection (or a dataset sub-sampling) could be designed to train models in a more ideal environment, yielding a more accurate cough classifier.

- *Broadening classification targets*: The cough classification models currently only disambiguate *tuberculosis* coughs versus all other pulmonary ailments, however there are

many other diseases which could benefit from acoustic classification. It would be very valuable to investigate which pulmonary ailments have distinctive acoustic signatures by training models upon cough sounds from participants with a wide variety of pulmonary ailments and attempting to classify them using models based off of this dissertation's results. This would provide not only a valuable diagnostic tool but also a dataset for experimentation with more complex deep learning models. As noted previously, a chief limiting factor of learning these deep models is the lack of large amounts of data. Up to this point, there is no large, high-quality, public database of cough sounds that can be used for acoustic model development research. Such a dataset would aid not only in construction of new healthcare tools, but also the development of new acoustic analysis tools and the field of machine learning in general.

It remains an open question whether a "forced" cough contains the same pulmonary information as a natural cough. The data collection process for natural coughs is time-consuming and uncertain; some patients cough over a hundred times per hour, whereas others cough merely once or twice an hour. Further investigation is warranted to force users to cough on-demand, creating an acoustic event that may contain diagnostically relevant information, and to see if machine learning models can be trained upon such a dataset to extract that information. The work from [12] serves as a good initial result in this direction, it is worth investigating to ensure that forced tuberculosis coughs contain the same pulmonary information as natural coughs, and as such can be used to classify a forced cough as a tuberculosis cough. This would obviate the need for patient screening that entails an hour-long process waiting for a natural cough to occur.

- *Variable-length inputs* Cough sounds are not fixed-length events. Their duration can last from less than a quarter of a second to well over 5 seconds. This means that the fixed-length input to our convolutional models either must be large enough so as to capture all relevant information for long cough events (and capture much irrelevant information for short cough events) or limit their scope and exclude much relevent

information for long cough events. In this dissertation, we opted for the latter and found that we were still able to outperform previous methods by examining relatively short segments of time (200ms) and combining multiple readings together with a median filter. However it stands to reason that there may be significant benefits to building a recurrent convolutional model that is able to ingest variable-length sequences and detect cough sounds with no pre-set limit on length. Such systems are still an area of active research and training methodologies have not yet crystalized around a well-accepted best practice. We hypothesize that this could be a strong contribution not only to the field of acoustic pulmonary health sensing but also the field of acoustic machine learning as well.

- *Improving model runtime performance*: There remain many low-hanging fruits for improving model runtime performance. Starting from a high-level view of the model, further architectural improvements are likely possible. Separable convolutions [18] have been shown to have a negligible impact on model accuracy on many computer vision tasks while boasting an impressive speedup by factorizing convolutional kernels and restricting their rank within an intermediate operation. Similar transformations could be made upon these cough detection and classification models to further reduce parameter count and computational complexity.

  On most embedded platforms, integer arithmetic is significantly more time and power efficient than floating-point. In the extreme, binarization of network weights [20] has become possible through advancements in training techniques and model architectures. Accordingly, investigation into the level of quantization that this network can support may yield further model compression and performance enhancement. Previous works have shown that many pieces of information within the network can be significantly compressed or outright discarded depending on the importance the network places upon that weight [26]. Similar transformations could be enacted upon this network to create a heterogenously quantized version of the model that has been optimally

compressed for the particular task of cough detection or classification.

Finally, at the lowest level, the hardware platform being executed upon plays a heavy role in the overall performance of the system. Persistent recording and acoustic analysis is become ever more pervasive, with smartphones, smartwatches, home speaker systems, TVs and even cars running background machine learning models to detect wakewords, accept voice commands and detect other elements within the acoustic environment. These applications are enabled by low-power application-specific integrated circuits (ASICs) that are able to run machine learning models with much greater efficiency than a typical general-purpose CPU would be able to. Programming these ASICs takes significant engineering effort, partly because they are often proprietary and access to documentation and toolchains is restricted, but also because they are not yet within the realm of the hobbyist hacker and so the effort to open them up to general repurpose has so far been avoided. This may soon change however, as the chips become further commoditized and as machine learning technologies continue to have the barrier to entry lowered by researchers, corporations and other interested parties. The logical conclusion is that a device more purpose-built for machine learning than the Raspberry Pi 3 B+ could be coerced to run our machine learning models at far greater efficiency than even the Raspberry Pi. This would be ideal for all battery-powered applications of cough counting and classification, such as on-phone detection, mobile detection stations, etc...

Finally, discriminative adversarial networks are a general technique that can be utilized anywhere there are hidden variables correlated with desired data classes that cannot be easily disentangled. Harnessing the expressivity of deep learning to estimate and then remove the influence of these confounding variables is a powerful technique, however there remains further validation work that can be done to learn the limits of this technique and its applicability to new settings and data types. We look forward to other researchers incorporating this technique into their learning strategies to be able to fold in more disparate datasets than

would otherwise be possible with traditional learning methods.

ACNOWLEDGEMENTS

This Ph.D. would never have happened without the support of an entire village of supporters, advocates, tutors and peers. My thanks cannot be contained within a few pages of text and extends far beyond these few names listed here, but I do my best to highlight some of the pillars of my life these past years:

First and foremost, to my family; Susan, George, James, Elise, Max and Erika. Thank you for your continual support, love and gentle needling. I would not be here today if it weren't for you all.

To my labmates and mentors in the ISDL starting nearly a decade ago; thank you for tricking me into thinking that grad school was at all a good idea. Brian, Pascal, Xing, Kai, Scott, Tommy, Greg, you guided, encouraged and elevated my thinking in ways that I didn't know existed. Above all, you showed me that my education was merely leaving me with more questions than answers, and that is a gift I will carry with me for the rest of my life.

To Les; thank you for not firing me when I crashed the SSLI cluster after two weeks on the job. Your continual vigorous search for mathematical truth has shaped me into the thinker I am today more than any class or tutelage ever did.

To my labmates and mentors in LABSN; thank you for easing my transition into graduate life and responsibilities. Eric, Ross, Dan, Katherine, Mark and Jonathan. You all helped model to me what "true science" looks like, and how to straddle science and technology.

To KC; thank you for your patient tutelage, incessant drive to push my work beyond what I would do just by myself, for continually reaching out to me as a human being. I still owe you one incorrectly-made bubble tea.

To my labmates and mentors in the Ubicomp lab; thank you for making work more like a playground than a pit of despair. Eric, Sidhant, Gabe, Mayank, Keyu, Tien. Your

understated assumption that I knew what I was doing filled me with confidence when I was still struggling to find my place in a new lab. Your overstated mockery and well-meaning verbal abuse warmed me in times when I was feeling chilled. Eric, Edward, Ruth, Josh, Lilian, Alex, Jake, Farshid, Hanchuan, Mohit, CJ, Manuja. I never would have puzzled my way through a Ph.D. without you all. Whether we were groaning about professors that inexplicably hated us, folding paper cranes out of hard feedback, philosophizing about life and our place in it, or just fishing for cats together, I could not ask for a better group of people to grow with. *Quid facit quod flexuris.*

To Shwetak; thank you for giving me the freedom to explore, the space to fail, and for always looking out for the best opportunities for me in my personal growth.

To my extended family; Friday nights, 4:12, Living Water, Potato Clan, my cousins. Only one element of that list technically belongs, but this is *my* acknowledgements section, so I'm going to say what I want to say. You all helped keep me sane through the various ups and downs of life over the course of this winding journey, eccentricities notwithstanding.

BIBLIOGRAPHY

[1] Udantha R. Abeyratne, Vinayak Swarnkar, Amalia Setyati, and Rina Triasih. Cough Sound Analysis Can Rapidly Diagnose Childhood Pneumonia. *Annals of Biomedical Engineering*, 41(11):2448–2462, November 2013.

[2] Mahmood Al-khassaweneh and Ra'ed Bani Abdelrahman. A signal processing approach for the diagnosis of asthma from cough sounds. *Journal of Medical Engineering & Technology*, 37(3):165–171, April 2013.

[3] J. Amoh and K. Odame. Deep Neural Networks for Identifying Cough Sounds. *IEEE Transactions on Biomedical Circuits and Systems*, 10(5):1003–1011, October 2016.

[4] Justice Amoh and Kofi Odame. Technologies for developing ambulatory cough monitoring devices. *Critical Reviews^{TM} in Biomedical Engineering*, 41(6), 2013.

[5] Yusuf A. Amrulloh, Udantha R. Abeyratne, Vinayak Swarnkar, Rina Triasih, and Amalia Setyati. Automatic cough segmentation from non-contact sound recordings in pediatric wards. *Biomedical Signal Processing and Control*, 21:126–136, August 2015.

[6] Les E. Atlas, Toshiteru Homma, and Robert J. Marks II. An Artificial Neural Network for Spatio-Temporal Bipolar Patterns: Application to Phoneme Classification. In D. Z. Anderson, editor, *Neural Information Processing Systems*, pages 31–40. American Institute of Physics, 1987.

[7] Samantha J. Barry, Adrie D. Dane, Alyn H. Morice, and Anthony D. Walmsley. The automatic recognition and counting of cough. *Cough*, 2:8, 2006.

[8] Antony Barton, Patrick Gaydecki, Kimberley Holt, and Jaclyn A Smith. Data reduction for cough studies using distribution of audio frequency content. *Cough (London, England)*, 8:12, December 2012.

[9] Antony James Barton. Signal Processing Techniques for Data Reduction and Event Recognition in Cough Counting, November 2013.

[10] Yoshua Bengio. Practical recommendations for gradient-based training of deep architectures. *arXiv:1206.5533 [cs]*, June 2012. arXiv: 1206.5533.

[11] S. S. Birring, T. Fleming, S. Matos, A. A. Raj, D. H. Evans, and I. D. Pavord. The Leicester Cough Monitor: preliminary validation of an automated cough detection system in chronic cough. *The European Respiratory Journal*, 31(5):1013–1018, May 2008.

[12] G. H. R. Botha, G. Theron, R. M. Warren, M. Klopper, K. Dheda, P. D. van Helden, and T. R. Niesler. Detection of tuberculosis by automatic cough sound analysis. *Physiological Measurement*, 39(4):045005, April 2018.

[13] Leo Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, October 2001.

[14] Joy Buolamwini and Timnit Gebru. Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification. page 15.

[15] H. Chatrzarrin, A. Arcelus, R. Goubran, and F. Knoefel. Feature extraction for the differentiation of dry and wet cough sounds. In *2011 IEEE International Symposium on Medical Measurements and Applications*, pages 162–166, May 2011.

[16] Tianqi Chen, Mu Li, Yutian Li, Min Lin, Naiyan Wang, Minjie Wang, Tianjun Xiao, Bing Xu, Chiyuan Zhang, and Zheng Zhang. MXNet: A Flexible and Efficient Machine Learning Library for Heterogeneous Distributed Systems. *arXiv:1512.01274 [cs]*, December 2015. arXiv: 1512.01274.

[17] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *arXiv:1406.1078 [cs, stat]*, June 2014. arXiv: 1406.1078.

[18] François Chollet. Xception: Deep Learning with Depthwise Separable Convolutions. *arXiv:1610.02357 [cs]*, October 2016. arXiv: 1610.02357.

[19] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). *arXiv:1511.07289 [cs]*, November 2015. arXiv: 1511.07289.

[20] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. BinaryConnect: Training Deep Neural Networks with binary weights during propagations. *arXiv:1511.00363 [cs]*, November 2015. arXiv: 1511.00363.

[21] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Trans. Inf. Theor.*, 13(1):21–27, September 2006.

[22] I. Daubechies. *Ten Lectures on Wavelets*. Society for Industrial and Applied Mathematics, 1992.

[23] S. Davis and P. Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(4):357–366, August 1980.

[24] T. Drugman, J. Urbain, N. Bauwens, R. Chessini, C. Valderrama, P. Lebecque, and T. Dutoit. Objective Study of Sensor Relevance for Automatic Cough Detection. *IEEE Journal of Biomedical and Health Informatics*, 17(3):699–707, May 2013.

[25] Donald Enarson. *Respiratory diseases in the world: realities of today - opportunities for tomorrow*. European Respiratory Society, Sheffield, 2013. OCLC: 868996973.

[26] Josh Fromm, Shwetak Patel, and Matthai Philipose. Heterogeneous Bitwidth Binarization in Convolutional Neural Networks. *arXiv:1805.10368 [cs]*, May 2018. arXiv: 1805.10368.

[27] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. page 8.

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

[29] Marti A. Hearst. Support vector machines. *IEEE Intelligent Systems*, 13(4):18–28, July 1998.

[30] J. Y. Hsu, R. A. Stone, R. B. Logan-Sinclair, M. Worsdell, C. M. Busst, and K. F. Chung. Coughing frequency in patients with persistent cough: assessment using a 24 hour ambulatory recorder. *European Respiratory Journal*, 7(7):1246–1253, July 1994.

[31] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv:1502.03167 [cs]*, February 2015. arXiv: 1502.03167.

[32] Richard S. Irwin, Michael H. Baumann, Donald C. Bolser, Louis-Philippe Boulet, Sidney S. Braman, Christopher E. Brightling, Kevin K. Brown, Brendan J. Canning, Anne B. Chang, Peter V. Dicpinigaitis, Ron Eccles, W. Brendle Glomb, Larry B. Goldstein, LeRoy M. Graham, Frederick E. Hargreave, Paul A. Kvale, Sandra Zelman Lewis, F. Dennis McCool, Douglas C. McCrory, Udaya B. S. Prakash, Melvin R.

Pratter, Mark J. Rosen, Edward Schulman, John Jay Shannon, Carol Smith Hammond, Susan M. Tarlo, and American College of Chest Physicians (ACCP). Diagnosis and management of cough executive summary: ACCP evidence-based clinical practice guidelines. *Chest*, 129(1 Suppl):1S–23S, January 2006.

[33] F. Itakura. Line spectrum representation of linear predictor coefficients of speech signals. *The Journal of the Acoustical Society of America*, 57(S1):S35–S35, April 1975.

[34] Yongcheng Jing, Yezhou Yang, Zunlei Feng, Jingwen Ye, Yizhou Yu, and Mingli Song. Neural style transfer: A review. *arXiv preprint arXiv:1705.04058*, 2017.

[35] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]*, December 2014. arXiv: 1412.6980.

[36] Svetlana Kiritchenko and Saif M. Mohammad. Examining Gender and Race Bias in Two Hundred Sentiment Analysis Systems. May 2018.

[37] J. Korpáš, J. Sadloňová, and M. Vrabec. Analysis of the cough sound: an overview. *Pulmonary pharmacology*, 9(5):261–268, 1996.

[38] Anders Krogh and John A. Hertz. A Simple Weight Decay Can Improve Generalization. In J. E. Moody, S. J. Hanson, and R. P. Lippmann, editors, *Advances in Neural Information Processing Systems 4*, pages 950–957. Morgan-Kaufmann, 1992.

[39] Eric C. Larson, TienJui Lee, Sean Liu, Margaret Rosenfeld, and Shwetak N. Patel. Accurate and privacy preserving cough sensing using a low-cost microphone. In *Proceedings of the 13th international conference on Ubiquitous computing*, pages 375–384. ACM, 2011.

[40] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:436 EP –, 05 2015.

[41] Yann LeCun, Bernhard E. Boser, John S. Denker, Donnie Henderson, R. E. Howard, Wayne E. Hubbard, and Lawrence D. Jackel. Handwritten Digit Recognition with a Back-Propagation Network. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems 2*, pages 396–404. Morgan-Kaufmann, 1990.

[42] Min Lin, Qiang Chen, and Shuicheng Yan. Network In Network. *arXiv:1312.4400 [cs]*, December 2013. arXiv: 1312.4400.

[43] Robert G. Loudon and Linda C. Brown. Cough Frequency in Patients with Respiratory Disease. *American Review of Respiratory Disease*, 96(6):1137–1143, December 1967.

[44] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier Nonlinearities Improve Neural Network Acoustic Models. *International Conference on Machine Learning*, page 6, 2013.

[45] S. Matos, S. S. Birring, I. D. Pavord, and H. Evans. Detection of cough signals in continuous audio recordings using hidden Markov models. *IEEE Transactions on Biomedical Engineering*, 53(6):1078–1083, June 2006.

[46] K. McGuinness, K. Holt, R. Dockry, and J. Smith. P159 Validation of the VitaloJAK™ 24 Hour Ambulatory Cough Monitor. *Thorax*, 67(Suppl 2):A131–A131, December 2012.

[47] J. Monge-Alvarez, C. Hoyos-Barcelo, P. Lesso, and P. Casaseca-de-la Higuera. Robust Detection of Audio-Cough Events using local Hu moments. *IEEE Journal of Biomedical and Health Informatics*, pages 1–1, 2018.

[48] Vladyslav Nikolayevskyy, Paolo Miotto, Edita Pimkina, Yanina Balabanova, Irina Kontsevaya, Olga Ignatyeva, Alessandro Ambrosi, Girts Skenders, Arvydas Ambrozaitis, Alexander Kovalyov, Anna Sadykhova, Tatiana Simak, Andrey Kritsky, Svetlana Mironova, Olesya Tikhonova, Yulia Dubrovskaya, Yulia Rodionova, Daniela Cirillo, and Francis Drobniewski. Utility of propidium monoazide viability assay as a biomarker for a tuberculosis disease. *Tuberculosis*, 95(2):179–185, March 2015.

[49] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. WaveNet: A Generative Model for Raw Audio. *arXiv:1609.03499 [cs]*, September 2016. arXiv: 1609.03499.

[50] K. K Paliwal. On the use of line spectral frequency parameters for speech recognition. *Digital Signal Processing*, 2(2):80–87, April 1992.

[51] R. D. Patterson, Ian Nimmo-Smith, John Holdsworth, and Peter Rice. An efficient auditory filterbank based on the gammatone function. In *a meeting of the IOC Speech Group on Auditory Modelling at RSRE*, volume 2, 1987.

[52] Joseph W. Picone. Signal modeling techniques in speech recognition. *Proceedings of the IEEE*, 81(9):1215–1247, 1993.

[53] Juergen Schmidhuber. Deep Learning in Neural Networks: An Overview. *Neural Networks*, 61:85–117, January 2015. arXiv: 1404.7828.

[54] Arietta Spinou and Surinder S. Birring. An update on measurement and monitoring of cough: what are the important study endpoints? *Journal of Thoracic Disease*, 6(7):S728–S734, October 2014.

[55] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. page 30.

[56] S. Subburaj, L. Parvez, and T. G. Rajagopalan. Methods of Recording and Analysing Cough Sounds. *Pulmonary Pharmacology*, 9(5):269–279, October 1996.

[57] Vinayak Swarnkar, Udantha R. Abeyratne, Anne B. Chang, Yusuf A. Amrulloh, Amalia Setyati, and Rina Triasih. Automatic Identification of Wet and Dry Cough in Pediatric Patients with Respiratory Diseases. *Annals of Biomedical Engineering*, 41(5):1016–1028, May 2013.

[58] Luke Taylor and Geoff Nitschke. Improving Deep Learning using Generic Data Augmentation. *arXiv:1708.06020 [cs, stat]*, August 2017. arXiv: 1708.06020.

[59] B. H. Tracey, G. Comina, S. Larson, M. Bravard, J. W. López, and R. H. Gilman. Cough detection algorithm for monitoring patient recovery from pulmonary tuberculosis. In *2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 6017–6020, August 2011.

[60] World Health Organization. The top 10 causes of death, January 2017.

[61] Mingyu You, Zeqin Liu, Chong Chen, Jiaming Liu, Xiang-Huai Xu, and Zhong-Min Qiu. Cough detection by ensembling multiple frequency subband features. *Biomedical Signal Processing and Control*, 33:132–140, March 2017.