Classifying Wideband Acoustic Immittance Measurements with CNNs via Wasserstein GANs

Anisha Jain

Submitted to the Department of Mathematics of Smith College

in partial fulfillment of the requirements for the degree of

Bachelor of Arts

Luca Capogna and Susan Voss, Faculty Advisors

May 12, 2025

Contents

List of Figures

List of Tables

Abstract

Wideband Acoustic Immittance (WAI) refers to frequency-based measurements collected from the ear and serves as a non-invasive tool for auditory diagnosis. This thesis explores the classification of WAI measurements from infants with and without middle ear fluid using convolutional neural networks (CNNs). To address dataset imbalance and reduce classification bias, we employ Wasserstein generative adversarial networks (WGANs) to generate synthetic training samples. This thesis provides foundational background on deep learning concepts, including activation functions, convolutional operations, learning algorithms, and the universal approximation theorem. The thesis also presents key formulations in optimal transport—namely, the Monge and Kantorovich problems—alongside relevant measure theory concepts and the application of the Wasserstein distance in GANs.

Acknowledgements

I would like to thank Seyeon Lim and Iris Gao for their dedicated work in data management and model development during our special studies in the fall of 2024. Collaborating with them not only made this project possible, but also much more enjoyable. I am also deeply grateful to Dr. Susan Voss for her guidance in helping me understand the WAI database and for serving as my second advisor throughout this process. I am thankful for her insights and also for the chance to work on an interdisciplinary project that challenged and inspired me. Finally, I would like to thank Dr. Luca Capogna not only for supervising my thesis, but for his mentorship as I continue to grow into the world of math research. His support gave me the confidence to go after opportunities which I never would have imagined before.

Chapter 1

Introduction

Wideband acoustic immittance data (WAI) refer to many related quantities measured in response to acoustic signals applied to the ear. Research in WAI includes developing tools that automatically classify WAI to determine different diagnoses of the ear, as variations in WAI measurements across frequencies can be difficult to interpret with the naked eye. These diagnoses include identifying hearing loss that may be a result of fluid in the middle ear. In infants who are unable to communicate their experience in words, a tool which can classify an abnormality in WAI measurements will result in rapid and noninvasive diagnoisis.

For over two decades, researchers have collected wideband acoustic immittance (WAI) data from both normal and abnormal ears, typically relying on relatively small datasets and conventional statistical techniques like descriptive analysis and linear regression to explore how WAI varies with frequency. Such traditional approaches have not yielded reliable methods for

classifying ears as normal or abnormal. Recent advances in data availability through the WAI database managed by Dr. Voss [25] opens the door for using machine learning methods for this classification task.

1.1 Previous Work

Previous work has been done in building classification models for a particular ear diagnosis with good recall and accuracy. We evaluate results in classification by calculating the model's recall (the percentage of correctly classified inputs within a class), and accuracy (the percentage of correctly classified inputs overall); these metrics are discussed further in section 4.3. Grais et al. [14] provided the first example for using different machine learning models in classifying ears with a fluid build up in the ear. Specifically they used machine learning approaches such as K-nearest neighbors classifier, support vector machines, and random forest, as well as deep learning based classifiers, such as feedforward neural networks and convolutional neural networks. Ultimately, the CNN had the best performance out of these techniques, with an accuracy rate of 81%. This performance was slightly better than their feedforward network, with an accuracy network of 80%.

The network proposed in [14] had a significantly better performance with normal ears, with recall of 89% for normal ears, indicating that 89 % of normal ears were correctly classified. In contrast, abnormal ears had recall

of 70%. In order to improve performance, [14] suggest using larger data sets or augmenting datasets for the CNN to work better with small data sets. The studies in the papers by Sundgaard et al. [21] and Nie et al. [19] explored using data augmentation techniques to increase the effectiveness of classification networks.

Sundgaard et al. [21] developed a CNN to classify ears with fluid build up in the ear, attempting to distinguish between two subclasses in this diagnosis. In order to augment the dataset, they applied multiple different distortion techniques, introducing noise to produce new synthetic examples. These included adding Gaussian or exponential noise to the input, adding or multiply a constant across the entire input measurement, and setting all values in a randomly selected neighborhood of the input to the neighborhood's mean. Without augmentation, the accuracy of their classification network was 90%. With data augmentation, the resulting classification network reached accuracy rates of 92.6%, indicating that data augmentation can improve accuracy rates in classification.

Nie et al. [19] used transfer learning with data augmentation techniques to identify Otosclerosis from WAI frequency measurements. Similarly to [21], they constructed a dataset of synthetic data examples created by adding distortion to real data measurements. These included various strategies for adding random noise and additively combining randomly selected measurements. They then pre-trained a classification network on the synthetic

dataset, and then further trained the network on the real dataset. Without transfer learning, their CNNs acheived rates of 89.69% accuracy. With transfer learning, the CNNs achieved accuracy rates of 94.88%.

1.2 Our Approach

In this thesis, we aim to develop a deep learning classification network to identify normal and abnormal WAI absorbance measurements in infants. The preliminary success of [14], [19], and [21] in using machine learning methods indicates that it may be a successful tool for classifying abnormal and normal ears in infants. Particularly because neural networks are universal approximators they may be more able to represent a function that differentiates normal and abnormal ears than other traditional classification strategies such as regression. Neural networks will be discussed further in section 2.

The main problem we have to overcome is the scarcity of data available for abnormal ears compared to normal ears. The problem of training neural networks on unbalanced data sets is that the model may inherit bias present in the training data. To tackle this difficulty, we propose using generative artificial intelligence to augment the training set with synthetic data that balances the difference between normal and abnormal ears. Generative artificial intelligence refers to artificial intelligence that creates new con-

tent based on the data they are trained on. Both [19] and [21] indicate that data augmentation techniques have a significant positive impact in classification accuracy rates. Both publications use lightweight methods of linear combinations of data points and additive noise to create synthetic examples of data. In the age of generative artificial intelligence, synthetic data can also be generated efficiently using neural networks, which could potentially create more realistic and diverse examples that span the full distribution of real data. This thesis uses a model called Wasserstein Generative Adversarial Networks (WGANs) to generate synthetic examples of WAI data to increase the accuracy of the classification network.

WGANs are a variation on the generative adversarial network (GANs) originally proposed by Goodfellow et al. [13]. GANs involve two networks playing an adversarial game, where one network generates fake data to fool the other network, which tries to decide when data is real or fake. In playing this game against one another, both networks improve at their respective tasks. After training to an equilibrium, we can use the generator network to generate synthetic examples that are close to reality and are representative of the full distribution of data.

In practice, GANs are often instable and difficult to train. Arjovsky, Chintala, and Bottou [4] proposed using the Wasserstein distance to replace the loss function in GANs. The Wasserstein distance, coming from the field of optimal transport, represents how much effort is needed to transform the

distribution of synthetic data to the distribution of real data. The Wasserstein distance, under few conditions, is continuous and overall proves to be more theoretically grounded than the traditional loss functions in GANs. Wasserstein GANs and their mathematical foundations will be discussed further in section 3.

Our results show that the model trained on a hybrid data set composed of synthetic and real data, balanced so there are the same number of abnormal and normal measurements, will not classify according to any bias. The network's classification represents the true data distribution, as well as increases the accuracy for abnormal data points, but does not result in a significant increase in accuracy overall. To summarize, the ratio of abnormal versus normal ears predictions by our classification network is exactly the same ratio as in the testing data set with an accuracy rate of 85%.

Chapter 2

Deep Learning

Neural networks are the fundamental concept in deep learning, drawing inspiration from the structure and function of the human brain. They have the capacity to model many different functions, making them a flexible and powerful tool in countless applications. In this section I will describe the fundamental components of a neural network, show an important variant of a neural network called a convolutional neural network, and show how neural networks learn. Finally, I will provide a result that shows neural network's ability to express many different functions.

2.1 A Network's Architecture

The goal of a neural network is to estimate a function f^* based on a set of data [12]. We can represent the data $D = (X, Y)$. The tuple (x_i, y_i) represents one data point, where each x_i is a vector of input features and y_i is

the corresponding label for x_i. In our application, x_i is information about an ear, and y_i is the corresponding diagnosis label. Hence, our neural network is aiming to find the function f^* that accurately distinguishes between different diagnoses in Y to output y_i for a given x_i. Ultimately, we want our network f^* to accurately diagnose inputs that are not within the dataset but that are found in the real world. This concept is called *generalization.*

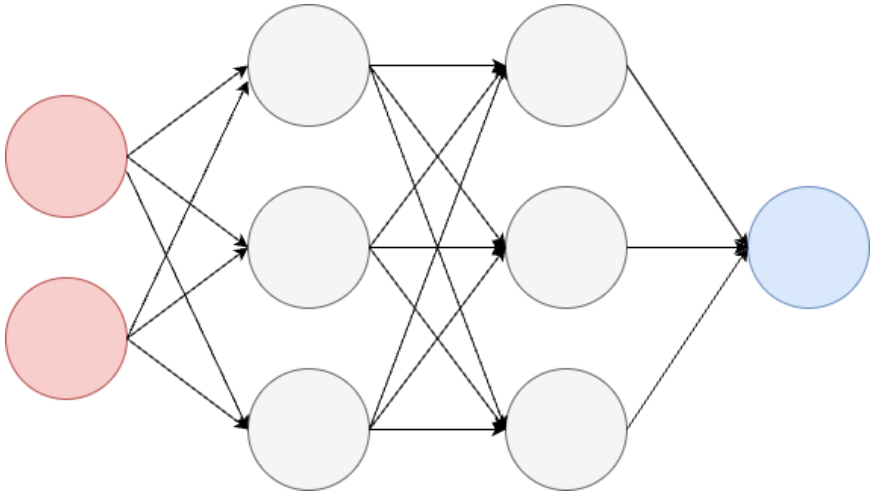


Figure 2.1: A fully connected Neural Network with two hidden layers.

A Neural Network is composed of many interconnected cells, which can be seen in the figure 2.1. The cells in the network cells are grouped into layers, and the number of layers in the models is called the *depth* of the network. Additionally, the number of cells in a single layer is its *width*, and the maximum width of the layers in a network is the network's width. The first and last layer of the neural network are called the input and output layer respectively, while the layers in between are called the network's *hidden layers.* In figure 2.1, the network has a depth of 4 and a width of 3, with two

hidden layers shown in gray, the input layer shown in red, and the output layer shown in blue.

Each individual cell performs a simple function which can be summarized generally as

$$\sigma(w \cdot x + b).$$

In a single cell, two steps occur. First, an affine function is applied to the input: the transformation w is applied to the input x and a bias term b is added. Second, a nonlinear function σ is applied to the result of the affine function.

More mathematically, a neural network is a parameterized family of functions $f(\theta; x) = y$, where $f(\theta; x)$ is a multiple composition of affine/-linear functions and nonlinear functions called *activation functions*. To give an example of the structure of the function, we will define some notation and then unpack its meaning. Let L be the number of layers in the network and let the input $x \in \mathbb{R}^n$. Define the *weights* matrices $W_i \in \mathbb{R}^{n_i \times n_{i-1}}, i = 1, \ldots, L$ and define the *bias* vector $b_i \in \mathbb{R}^{n_i}, i = 1, \ldots, L$. Then we define the parameter $\theta_i = \{W_i, b_i\}$, so the set of all weights and biases are denoted by θ. Also, let $\sigma : \mathbb{R}^{n_i} \to \mathbb{R}^{n_i}$ be some activation function, which we will describe in more detail later.

A full layer of the network f_i is denoted

$$f_i(\theta_i, x) = \sigma(W_i \cdot x + b_i).$$

In the layer, the activation function is applied element-wise to the result of the affine function performed between the weights and biases W_i, b_i and input x. For a network with a depth of $L = 2$, the entire network f can be written as

$$f(\theta; x) = f_2(\theta_2; f_1(\theta_1, x)) = \sigma(W_2 \cdot \sigma(W_1 \cdot x + b_1) + b_2). \tag{2.1}$$

The two layers in f are f_1 and f_2, and the full network is a composition of the two layers so that the output of f_1 is the input of f_2.

2.1.1 Activation Functions

Now we will describe in more detail what the nonlinear functions known as activation functions are in neural networks. The purpose of the activation function is to introduce non-linearity into neural networks, allowing them to be more expressive. Below we will describe three of the most commonly used activation functions: Sigmoid, ReLU, and Leaky ReLU.

- **Sigmoid**

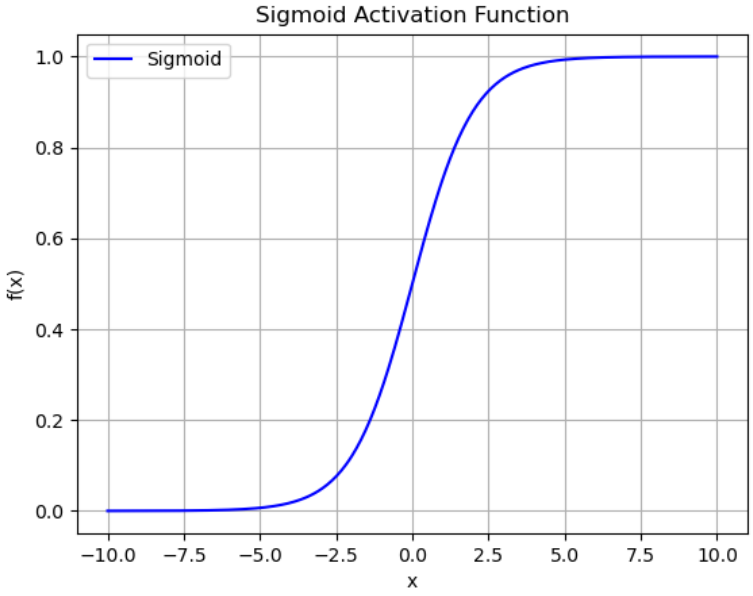


Figure 2.2: Sigmoid activation function.

The sigmoid function is defined as

$$\sigma_{\text{sigmoid}} = \frac{1}{1 + e^{-z}}.$$

This activation function is inspired by the idea of a neuron, "squashing" values between values generally 0 and 1, where 1 can be thought of as a neuron firing. It is a strictly increasing and differentiable function.

- **ReLU**

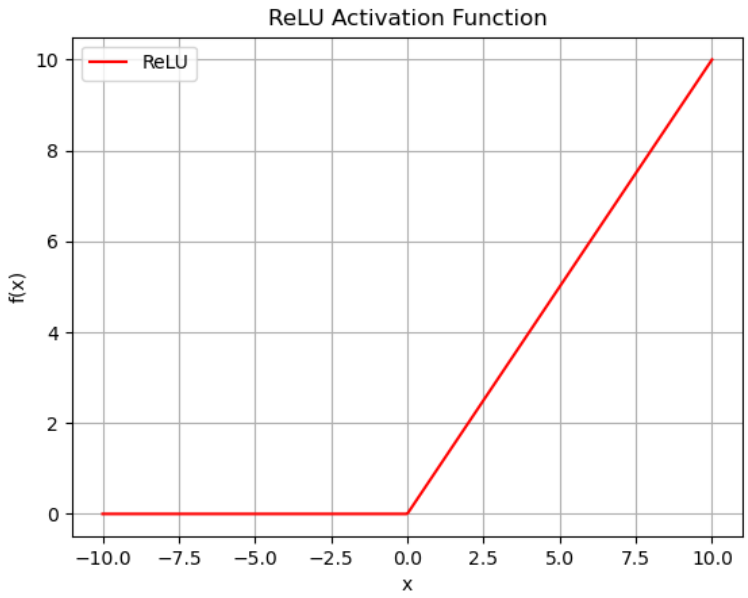


Figure 2.3: Rectified Linear Unit

The Rectified Linear Unit (ReLU) function, shown in 2.3, is a peice-wise linear function defined as

$$\sigma_{\text{ReLU}}(z) = \max(0, z).$$

In words, the function takes a non-negative input and outputs itself, and takes a negative input and outputs 0. The function has very straight forward gradient calculations, which is useful in optimization. Additionally, due to requiring only one step to compute, the using ReLU makes training fast. ReLU is often used in convolutional neural networks, discussed in the next section.

- **Leaky RelU**

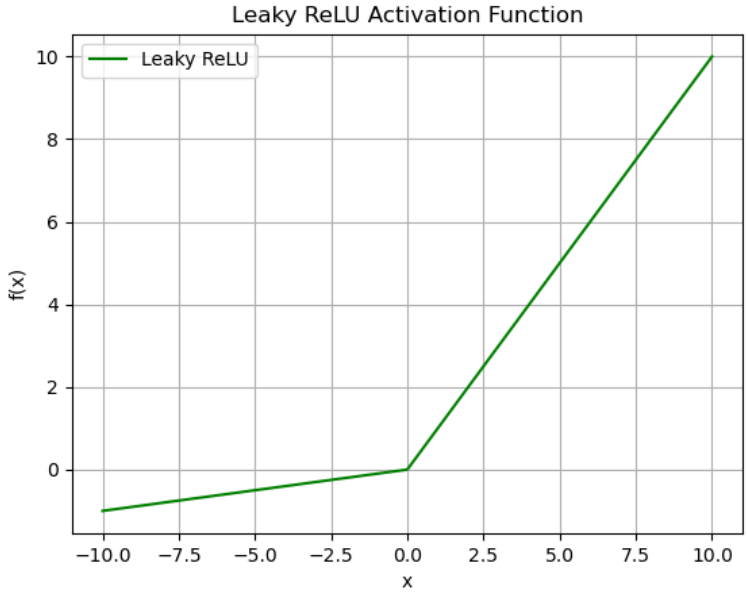


Figure 2.4: Leaky ReLU

Leaky ReLU is very similar to an ordinary ReLU function, except the function is non-zero when the input is negative. Rather, it sends negative inputs to some fraction α of the input. This can avoid a problem called the "Dying ReLU" problem, where if there are many negative inputs neurons fail to be activated. The leaky ReLU function is defined as

$$\sigma_{\text{Leaky ReLU}}(z) = \begin{cases} z, & \text{if } z > 0 \\ \alpha z, & \text{if } z \leq 0 \end{cases}$$

Activations in Neural Networks

In a single network, different activation functions may be used. For example, in one layer one could choose the ReLU function, and in another layer choose the sigmoid function. In addition, in the last layer of the network, there may or may not be an activation function depending on the desired format of the output. Choosing the best activation functions depends on the structure of the neural networks, as different activation functions may be better for their computational efficiency or experimental performance.

2.2 Convolutional Neural Networks

Convolutional neural networks (CNNs) are a type of neural networks, and are used in cases when data features have a spatial correlation within a data point. For example, if we consider an image as a data point, we know that a certain pixel within it is likely to be similar to the pixels near it. Another example is a single data point that is a month of stock prices; a certain day in that month will likely have similar prices to the days right before and after. In these examples, there is a grid-like structure where top, bottom, left, and right are clearly defined. Hence, this grid-like structure in data is when a CNN is useful.

2.2.1 Convolution operation

CNNs rely on an operation called a *convolution,* which for two real-valued functions $x(t), w(t)$ is defined as

$$s(t) = \int_{-\infty}^{\infty} x(a)w(t-a)da, \tag{2.2}$$

where $s(t)$ is the convolution of $x(t)$ and $w(t)$. The convolution operation is also notated with an asterisk, as $s(t) = (x*w)(t)$. Sometimes, the first function $x(t)$ is called the *input*, the second function $w(t)$ is called the *kernel*, and the output $s(t)$ is called the *feature map*.

Since data used in machine learning is discrete, we can discretize the convolution operation in (2.2),

$$s(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a), \tag{2.3}$$

which more accurately represents the convolution operation in machine learning applications. This can also be re-written in any number of dimensions for different applications; for example, in two dimensions for images. Usually, the input $x(t)$ represents a multi-dimensional array of input data, the kernel $w(t)$ represents an array of parameters that are learned by the network.

I will provide an example of the convolution operation to illustrate how the function takes advantage of spatial relationships within a data point. Let

$x(t)$ be a certain discrete function and let $w(t)$ be a certain blurring filter.

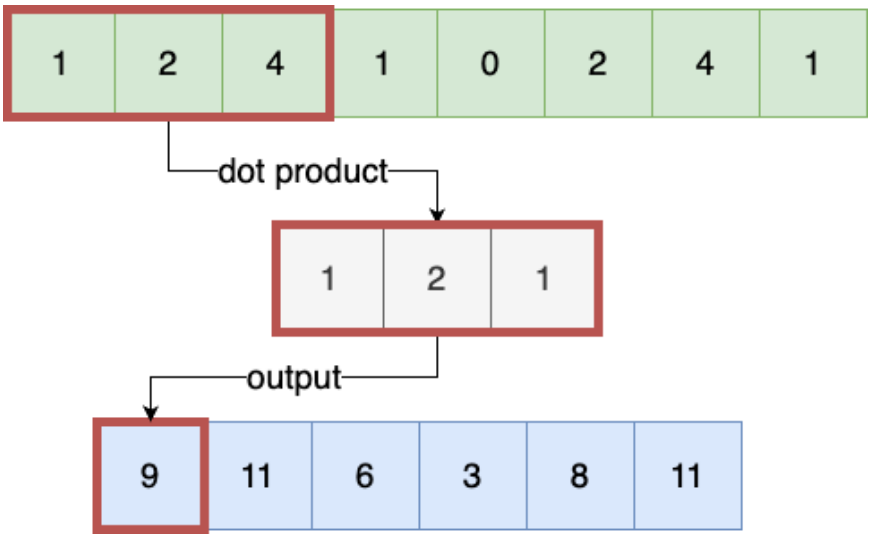


Figure 2.5: Convolution operation in 1-dimension.

When we apply the convolution in equation (2.3), we are taking a weighted sum of a small area of the data provided by $x(t)$, with the weights provided by $w(t)$. Through this operation, we are able to get a summary of the values in small neighborhoods of the data.

2.2.2 Convolutions in Neural Networks

A convolutional neural network is very similar to a regular neural network, except that in at least one layer, a convolution operation is used instead of a generic matrix multiplication operation [12]. The layers where a convolution is used is called a convolutional layer, and layers with the generic matrix multiplication operation are called dense layers. There are many benefits to using a convolution operation from a network architecture perspective.

First, a convolution can reduce the number of parameters required for

learning. Using a convolution requires the network to learn weights in the kernel $w(t)$. Since a useful kernel can be very small compared to the input, which can have hundreds if not millions of features, training convolution parameters requires far fewer operations compared to a generic neural network. This reduces computation time and space required for learning and inference.

Secondly, the convolution operator is equivariant with respect to translations. If we apply a translation function that shifts our input, the result of the convolution will be shifted in the same manner. Let $I(x-1, y)$ represent translating an image $I(x, y)$ and K be the convolution kernel. Equivariance in convolutions means that

$$(I * K)(x - 1, y) = I(x - 1, y) * K.$$

In words, this means that shifting the image and then convolving will produce the same result as convolving and then shifting the image. This allows the CNN to extract certain patterns that occur in the data in different places, like peaks in stock prices or edges in an image.

2.3 Learning

2.3.1 Loss Functions

Learning in deep learning involves finding the optimal function f^* that maps an input x_i to its correct label y_i (discussed in 2.1). In otherwords, learning is an optimization problem. We optimize on a certain function L, which we call the *Loss Function.* There are many different loss functions, but they represent a way to quantify the distance between the real data distribution and the learned distribution defined by f. Hence, the optimization problem involves finding the weights θ in f so that we minimize the error formulated by the loss function L. We can concretely describe the minimization problem as

$$\min_f \frac{1}{N} \sum_{i=1}^{N} L(f(\theta; x), y_i) + R(f), \tag{2.4}$$

where N is the number of data points used for training, $L()$ is the loss function for a single data point, $f()$ is the neural network, θ is the weights vector from the neural network, and $R(f)$ is some regularization term [27]. In broad terms, a regularization term penalizes the loss function to help choose weights which help the model generalize better.

There are many different classical loss functions used in machine learning, but in this section we will discuss two.

Mean Squared Error

Mean Squared Error (MSE) is a familiar loss function also used in regression tasks. The MSE is defined as

$$L_{MSE}(f(\theta; x), y) = \frac{1}{N}\sum_{i=1}^{N}(y_i - f(x_i))^2.$$

MSE is the average of the squared error between the true value y_i and the predicted value $f(x_i)$. When the errors are squared, those data points with more extreme errors will contribute more to the loss function than those with very small errors.

Binary Cross Entropy

Binary Cross Entropy, also referred to as Log Loss, is a loss function that is used in binary classification tasks. An example of binary classification would be a diagnostic test used to determine whether an individual does or does not have a certain diagnosis. We can label the outcomes "does have a diagnosis" as 1 and "does not have a diagnosis" as 0. The binary cross-entropy loss function is defined as

$$L_{BCE}(f(\theta; x), y) = -\frac{1}{N}\sum_{i=1}^{N} y_i \log(f(x_i)) + (1 - y_i)\log(1 - f(x_i)).$$

The first term in the log loss, $y_i \log(f(x_i))$ is non-zero when the real label $y_i = 1$; hence the term penalizes more if $f(x_i)$ is closer to 0 than 1. The

second term in the log loss, $(1 - y_i) \log(1 - f(x_i))$, is non-zero when the real label $y_i = 0$; similarly, the term increases the loss function when $f(x_i)$ is closer to 1 than 0. Moreover, log loss is a convex function, so it is more straightforward to find the loss function's global minimum.

2.3.2 Stochastic Gradient Descent

Finding the parameters θ that minimize the loss function is relatively easy when the number of parameters is small. However, in a neural network there may be hundreds, if not thousands of parameters, which makes it difficult to analytically calculate the exact minima of the loss function. Hence, we use an iterative approach called *gradient descent* that slowly adjusts the weights until a minima of the loss function is found [12].

In one iteration of gradient descent, we first aim to find the negative gradient of the loss function with respect to θ, $-\nabla L$. The negative gradient tells us the steepest method of descent to the minimum. After finding the gradient, we take one step towards the minima, proposing a new set of weights

$$\theta' = \theta - \lambda \nabla L, \tag{2.5}$$

where $\lambda \in \mathbb{R}^+$ represents the size of the step, called the *learning rate.* The learning rate is chosen to balance computation time and finding the minimum. A smaller learning rate means that the gradient descent algorithm will be less likely to overshoot the minima, but it will also take longer to

find the minimum in the first place. Additionally, a lower learning rate may force the algorithm to get stuck in a local minimum. Learning rates can also be variable.

In the proposed form of gradient descent, when calculating the gradient ∇L, we must evaluate the loss function over every point in the training dataset. This can be very expensive if the size of the dataset is very large. *Stochastic gradient descent* refers to a method where, rather than computing the gradient over every point in the dataset, we compute it over only one data point. However, this can result in gradients that vary greatly, which can impede convergence to the minimum. Usually, the gradient will be computed on a small random sample of data points. In practice, this results in models that converge quickly by using approximate gradients rather than exact ones. When the sample is larger than 1 but smaller than the entire dataset, the process is called *mini-batch stochastic gradient descent.*

Mini-batch can offer many advantages when training neural networks, but the primary consideration is how large of a batch to use. Larger batches will result in more accurate estimations of the gradient of the loss function, but will not provide as many savings on computation time. In contrast, smaller batches can result in very noisy gradients, which can add a regularizing effect but likely require more iterations to reach the minima. In order to compare training times we can use actual time passed, but we also introduce the following idea: one pass through the entire dataset using multiple

batches is called an *epoch*, and it is another way to measure the time it takes to train a neural network.

In general, optimizing neural networks is not an easy task, and there are many difficulties associated with gradient descent. Neural networks, with their many parameters and nonlinear activation functions, are almost always non-convex functions. Any minima which gradient descent explores is not guaranteed to be the global minimum. Additionally, in high dimensional non-convex functions, we may also encounter saddle points in gradient descent. Some solutions to this problem include using a noisy gradient to avoid getting stuck in these non-optimal critical points.

Another problem in gradient descent is known as the *vanishing and exploding gradient* problem, which we will discuss next in section 2.3.3 titled Backpropagation.

2.3.3 Backpropagation

Backpropagation is an algorithm for computing the gradient to update the weights in gradient descent. To illustrate backpropagation, return to the two layer example of a neural network in equation (2.1),

$$f(\theta; x) = f_2(\theta_2; f_1(\theta_1, x)) = \sigma(W_2 \cdot \sigma(W_1 \cdot x + b_1) + b_2),$$

where f_2, f_1 are the layers in the network f and $\theta = \{W_1, b_1, W_2, b_2\}$ are the parameters. For simplicity, assume that each layer only has one cell.

The first step in computing the gradient is the forward pass, where we compute $f(\theta; x_i)$ for all x_i in the training data. After this forward pass, we are able to compute the result of the loss function $L(f(\theta; x), y)$.

The second step in computing the gradient is the backward pass. Hence, we start with computing the gradient with respect to the output layer, $\frac{\partial L}{\partial f_2}$. Next we can compute the gradient on the input to the activation function in the second layer, $z_2 = W_2 f_1(\theta_1; x) + b_2$, which requires the chain rule.

$$\frac{\partial L}{\partial z_2} = \frac{\partial L}{\partial f_2}\sigma'(z_2) = \frac{\partial L}{\partial f_2}\sigma'(W_2 f_1(\theta_1; x) + b_2).$$

Next, compute the gradient with respect to W_2 and b_2 :

$$\frac{\partial L}{\partial W_2} = \frac{\partial L}{\partial f_2}\sigma'(W_2 f_1(\theta_1; x) + b_2) f_1(\theta_1; x)$$

$$\frac{\partial L}{\partial b_2} = \frac{\partial L}{\partial f_2}\sigma'(W_2 f_1(\theta_1; x) + b_2).$$

Now that we can get the gradients w.r.t to W_2 and b_2, we can update those parameters accordingly.

We continue on to find the gradients of the remaining weights. Find the partial with respect to f_1 :

$$\frac{\partial L}{\partial f_1} = \frac{\partial L}{\partial f_2}\sigma'(W_2 f_1(\theta_1; x) + b_2) W_2.$$

Next, we find the gradient with respect to the input of the activation function

in the first layer, $z_1 = W_1 x + b_1$.

$$\frac{\partial L}{\partial z_1} = \frac{\partial L}{\partial f_2} \sigma'(W_2 f_1(\theta_1; x) + b_2) W_2 \sigma'(W_1 x + b_1).$$

Finally, find the gradient with respect to the parameters W_1 and b_1.

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial f_2} \sigma'(W_2 f_1(\theta_1; x) + b_2) W_2 \sigma'(W_1 x + b_1) x.$$

$$\frac{\partial L}{\partial b_1} = \frac{\partial L}{\partial f_2} \sigma'(W_2 f_1(\theta_1; x) + b_2) W_2 \sigma'(W_1 x + b_1).$$

Through this process, we can find the gradients that are evaluated on the training data, which can be used to update the parameters W_2, b_2, W_1, b_1. The gradient is found in a recursive manner from back to front in the neural network, hence the name back propagation.

In the back propagation algorithm, a series of multiplications via the derivative chain rule occur to compute the gradient to update a single weight. If the quantities in the product are less than one, this can result in the gradient vanishing. In gradient descent, this results in very little change in the parameters and therefore very little further optimization of the loss function. Alternatively, if the quantities in the product are all greater than one, this can result in the exploding gradient problem. When the gradient becomes very large, learning becomes very unstable. [12] provides an example of the exploding gradient problem, called the cliff problem. When approaching a cliff from below or above, a few steps can result in a massive gradient that

loses all the progress gradient descent has made. Some solutions to the exploding and vanishing gradient problems include starting gradient descent with different parameters and clipping gradients to a certain interval.

2.4 Universal Approximation

Neural networks are a powerful tool because of their *expressiveness*: they have the ability to approximate many different functions. Cybenko [9] provides a proof for a neural network's ability to approximate universally, and we will provide the result here.

First we will define some notation. $I_n = [0,1]^n$ is an n-dimensional unit cube. $C(I_n)$ is the space of all continuous functions on I_n with the supremum norm $||\cdot||$. $M(I_n)$ is the space of all finite, signed regular Borel measures on I_n. Measures and measure theory will be discussed in the following chapter.

Cybenko's theorem applies to any function which is considered *discriminatory*. A continuous sigmoidal function is an example of a discriminatory function, which is defined as a function σ, where given a measure $\mu \in M(I_n)$ such that

$$\int_{I_n} \sigma(w^T x + b) d\mu, w \in \mathbb{R}^n, b \in \mathbb{R}$$

implies that $\mu = 0$.

Now we can provide Cybenko's universal approximation theorem.

Theorem 1. *Let σ be any continuous discriminatory function. Then finite sums of the form*

$$G(x) = \sum_{j=1}^{N} a_j \sigma(w_j^T x + b_j), \tag{2.6}$$

where $w_j \in \mathbb{R}^n$ and $a_j, b_j \in \mathbb{R}$, are dense in $C(I_n)$.

In other words, given any $\epsilon > 0$ and $f \in C(I_n)$, there is a sum $G(x)$ of the above form such that

$$|G(x) - f)| < \epsilon, \forall x \in I_n.$$

The theorem shows that a one layer neural network using a continuous sigmoidal activation function can approximate any Borel measurable function within an arbitrary bound, provided there are no restrictions placed on the number of cells or size of the weights. Note that any continuous function on a closed and bounded subset of $\mathbb{R}^n$ is Borel measurable, which theorem 1 shows can be approximated by a neural network.

Despite theorem 1 providing the strong conclusion of the existence of a neural network to represent any continuous function in $\mathbb{R}^n$, it does not prove that this neural network will be able to be learned. [12] provides two reasons for this. First the optimization algorithm used for training may not find the exact parameters which correspond to the optimal function. Second, the optimization algorithm might choose a different function altogether, by

way of overfitting to the data points used for training.

Additionally, theorem 1 does not place any bounds on the size of the optimal one layer neural network, and does not describe how wide it needs to be. Barron [5] provides a bound on the size of the network who's existence is shown in theorem 1, but shows in the worst case that an exponential number of cells is needed. This can result in an infeasibly wide neural network. Therefore, in most practical approaches, deeper rather than wider neural networks with many layers are used to represent the target function.

2.5 Problems in Generalization

In the previous section, we discussed how neural networks can be effective at many tasks due to their expressivity, but they still suffer with many of the difficulties that other traditional models do. In this section, we will describe two problems that are important when evaluating the capability of generalization in neural networks. This section is informed by Goodfellow, Bengio, and Courville [12].

2.5.1 Bias

Statistical bias refers to some kind of systematic error that causes the outcome to differ from the expected outcome. For example, suppose the training and testing sets for a certain neural network has 20% of class A and 80% of class B. If the network classifies the testing set as 20% of class A

and 80% of class B, that would be the expected and desired result. But if the network consistently classifies the testing set as 50% of class A and 50% of class B, we would say that the network is biased towards class A, and some intervention method is needed.

It is very important to identify bias, because in practice it can result in making incorrect decisions when attempting to generalize. This is because bias can hide or exaggerate relationships in data. Bias can be introduced in many ways, including selection bias where the sample is not representative of a population. In our application with classifying ears, despite having a representative training sample of measurements, some networks are significantly more likely to classify a measurement as normal than abnormal, indicating a bias. Our work attempts to address this bias by augmenting the dataset by generating synthetic examples.

2.5.2 Overfitting and Underfitting

Because of their expressivity, neural networks can be prone to overfitting. Overfitting occurs when the gap between the training error and validation error become very large. In other words, the model performs very well on the training dataset, but does not perform well on the testing dataset.

In contrast, underfitting can occur when the neural network is not trained long enough on the dataset. Hence the training error of the dataset is still high.

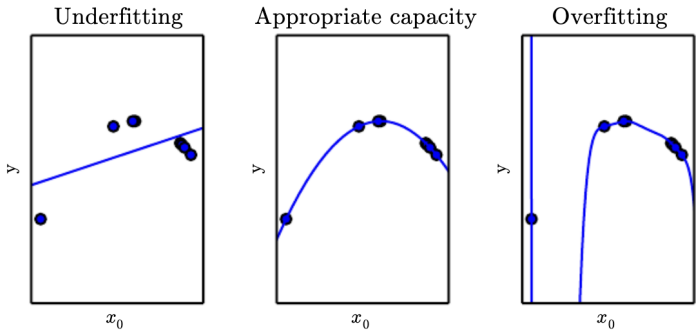


Figure 2.6: Underfitting, Overfitting [12].

Figure 2.6 shows three two dimensional graphs, where x_0 is the domain, y is the range. The points represent data points in the dataset and the three curves represent functions fit to the data. The first shows a model which is underfit, because it does not accurately represent the shape of the data. The middle shows a curve which is appropriately fit to the data, as it represents the shape of the data well. The third curve is overfit, as while it does conform to the shape of the data, it fits it too closely. Assuming that the population data fits the curve shown by the second graph well, neither the first or the third graph would be able to accurately generalize to data which had not been seen by the network before.

Chapter 3

Wasserstein GANs and their Mathematical Foundations

In this section, we will introduce some relevant definitions in measure theory, and describe two important optimal transport formulations including the Wasserstein distance. Lastly, we will show how Wasserstein distances can be used in Generative Adversarial Networks (GANs), a kind of generative artificial intelligence.

3.1 Introductory Measure Theory

Understanding optimal transport requires knowledge in measure theory. In this section, I will provide and explain definitions for concepts that will be utilized throughout this thesis. These definitions are provided through Wheeden and Zygmund [29], a textbook on analysis and measure theory, unless noted. First, I will go through measure theory in $\mathbb{R}^n$. Then I will

reframe the same ideas more abstractly to develop a more robust framework for optimal transport.

3.1.1 Measure theory in $\mathbb{R}^n$

To quantify regions in $\mathbb{R}^n$, we begin with one of the simplest building blocks: an interval. An *n-dimensional interval* is a special set that is a subset of $\mathbb{R}^n$ of the form $I = \{(x_1, ..., x_n) : a_k \leq x_k \leq b_k, k = 1, .., n\}$, where $a_k < b_k, k = 1, ..., n$.

Like an ordinary cube in $\mathbb{R}^3$, we also consider these n-dimensional intervals to have a volume. The *volume* $v(I)$ of an interval $I = \{(x_1, ..., x_n) : a_k \leq x_k \leq b_k, k = 1, .., n\}$ is

$$v(I) = \prod_{k=1}^{n} (b_k - a_k).$$

These volumes allow is to consider a size for these intervals. We will build on this notion of size for any arbitrary set. Consider closed n-dimensional intervals $I = \{(x_1, ..., x_n) : a_k \leq x_k \leq b_k, k = 1, .., n\}$ and their volumes $v(I) = \prod_{k=1}^{n} (b_k - a_k)$. To define the *outer measure* of an arbitrary subset $E \in \mathbb{R}^n$, enclose E by a countable collection S of intervals I_k. Let

$$\sigma(S) = \sum_{I_k \in S} v(I_k).$$

The outer measure of E, denoted $|E|_e$ is defined by

$$|E|_e = \inf \sigma(S),$$

where the infimum is taken over all such covers S of E. The outer measure allows us to measure any subset of real numbers, but the outer measure fails to have certain properties because of some sets that do not behave nicely. By restricting the domain of the sets where we conduct the outer measure, we develop certain properties; we will develop this idea further when discussing measure theory abstractly. Hence, we develop the idea of measurability for sets which are well-behaved.

A subset E of $\mathbb{R}^n$ is said to be *Lebesgue measurable,* or simply *measurable* if, given $\epsilon > 0$, there exists an open set G such that

$$E \subset G \text{ and } |G - E|_e < \epsilon.$$

Intuitively, this means that E is a well-behaved set in the sense that we can approximate it arbitrarily well with open sets. If E is measurable, its outer measure is called its *Lebesgue measure* or simply *measure*, and is denoted $|E|, |E| = |E|_e$ for all measurable E. In other words, the outer measure and Lebesgue measure are the same measurements, but the Lebesgue measure is conducted only on sets which are measurable.

With a notion of measure established for sets, we now turn to functions,

as problems in optimal transport involve functions. To ensure that integration behaves well, we require functions to interact properly with measurable sets, leading to the concept of Lebesgue measurability in functions.

Let f be a real-valued function defined on a set E in $\mathbb{R}^n$, where $-\infty \leq f(\boldsymbol{x}) \leq \infty$, $\boldsymbol{x} \in E$. Then f is called a *Lebesgue measurable function* on E, if for every finite a, the set

$$\{\boldsymbol{x} \in E : f(\boldsymbol{x}) > a\}$$

is a measurable subset of $\mathbb{R}^n$. In shorthand, we will use the abbreviation $\{f > a\}$ for $\{\boldsymbol{x} \in E : f(\boldsymbol{x}) > a\}$ to denote what is called a level set of f. As a varies, the level set describes how f is distributed.

Now that we have defined the measurability of functions, we can define their integral. There are many different and equivalent ways to define the Lebesgue integral, and we will begin with the idea that the integral of a (non-negative) function f should be represented by the volume under the curve of f. Let f be defined on a measurable subset $E \in \mathbb{R}^n$. Let $R(f, E)$ represent the *region of f over E,* and define it as

$$R(f, E) = \{(\boldsymbol{x}, y) \in \mathbb{R}^{n+1} : \boldsymbol{x} \in E, 0 \leq y \leq f(\boldsymbol{x})\}.$$

If R is measurable, then its measure is called the *Lebesgue integral of f over*

E:

$$|R(f,E)|_{n+1} = \int_E f(\boldsymbol{x})d\boldsymbol{x}.$$

Note that this definition is only for those functions which are non-negative, but since we are primarily working with probability distributions in optimal transport, which are non-negative by nature, this definition is suitable for developing an understanding of the Lebesgue integral.

3.1.2 Abstract Integration

We have developed an interpretation of measurability based on the basic idea of the interval in $\mathbb{R}^n$. Now we will develop a more general understanding of integration and measurability that can be applied to many different sets without such a geometric structure, especially those sets not in $\mathbb{R}^n$.

First we define the σ-*algebra*, which we can also name a *countably additive family of sets.* Let $\mathscr{L}$ be a fixed set and let Σ be a σ-algebra of subsets of $\mathscr{L}$. Then Σ satisfies the following properties:

1. $\mathscr{L} \in \Sigma$
2. if $E \in \Sigma$ then it's complement $E^C = \mathscr{L} - E \in \Sigma$
3. if $E_k \in \Sigma$ for $k = 1, 2, ...,$ then $\bigcup E_k \in \Sigma$.

In other words, these conditions say that Σ is nonempty, it is closed under complements, and it is closed under countable unions. We also call the elements E of Σ as Σ-*measurable sets.*

Next we will define some functions on a σ-algebra. If Σ is a σ-algebra then a real-valued function $\phi(E), E \in \Sigma$ is called an *additive set function* on Σ if

1. $\phi(E)$ is finite for every $E \in \Sigma$ and
2. $\phi(\bigcup E_k) = \sum \phi(E_k)$ for every countable family $\{E_k\}$ of disjoint sets in Σ

Another function $\mu(E)$ defined for $E \in \Sigma$ is called a *measure* on Σ if

1. $0 \leq \mu(E) \leq \infty$ and
2. $\mu(\bigcup E_k) = \sum \mu(E_k)$ for every countable family $\{E_k\}$ of disjoint sets in Σ.

Although a measure and additive set function are similar, the difference is that a measure is always nonnegative and can be infinite, while an additive set function can take any signed finite. With this, we provide the following definition.

Definition 1. If μ is a measure on the σ-algebra Σ on $\mathscr{L}$ then the triplet $(\mathscr{L}, \Sigma, \mu)$ is called a *measure space.*

Next we will move to developing the ideas of measurable functions and integration on those functions in an abstract setting. Let Σ be a fixed σ-algebra of subsets on $\mathscr{L}$, and let $f(x)$ be a real-valued function defined for x

in a measurable set E. We say that f is Σ-*measurable* or simply *measurable* if $\{x \in E : f(x) > a\}$ is measurable for $-\infty < a < \infty$.

Let f be a non-negative function on a measurable set E. Define the *integral of f over E with respect to μ* by

$$\int_E f d\mu = \sup \sum_j \left[\inf_{x \in E_j} f(x) \right] \mu(E_j).$$

where the supremum is taken over all decompositions $E = \bigcup E_j$ of E into the union of a finite number of disjoint measurable sets E_j. Another way to define the integral is

$$\int_E f d\mu = \int_E f^+ d\mu - \int_E f^- d\mu.$$

given that not both the integrals on the right hand side are ∞. We say that f is *integrable with respect to μ* over E if $\int_E f d\mu$ exists and is finite. When this is the case we write that $f \in L(E; \mu)$.

Now we will define the product of two measures. First we begin with the concept of a σ-algebra product. Let (Σ_1, X_1, μ_1) and (Σ_2, X_2, μ_2) be measurable spaces. Denote $\Sigma_1 \otimes \Sigma_2$ as the σ-algebra on the cartesian product $X_1 \times X_2 = \{(x_1, x_2) : x_1 \in X_1, x_2 \in X_1\}$. The *$\sigma$-algebra product* is

$$\Sigma_1 \otimes \Sigma_2 = \{(B_1, B_2) : B_1 \in \Sigma_1, B_2 \in \Sigma_2\}.$$

Definition 2. A *product measure* $\mu_1 \times \mu_2$ is defined on the measure space

$(\Sigma_1 \otimes \Sigma_2, X_1 \times X_2)$ and satisfies the property

$$(\mu_1 \times \mu_2)(B_1 \times B_2) = \mu_1(B_1)\mu_2(B_2).$$

Definition 4 is provided by Loève [17]. The product measure is helpful to describe the interaction between different measure spaces.

Next, a useful case of measure theory is in Borel measurability.

Definition 3. The smallest σ-algebra of subsets of $\mathbb{R}^n$ containing all open subsets of $\mathbb{R}^n$ is called the *Borel σ-algebra* $\mathscr{B}$ of $\mathbb{R}^n$, and the sets in $\mathscr{B}$ are called the Borel subsets of $\mathbb{R}^n$. Every Borel set is measurable, and a measure μ on the Borel subsets of $\mathbb{R}^n$ is called a *Borel measure.* A function f defined on a Borel set E is said to be *Borel measurable* if $\{f > a\}$ is a Borel set for every a.

Making use of the idea of Borel measurability, we will provide two definitions from Ambrosio, Brué, and Semola [3].

Definition 4. A *probability measure* μ is a real-valued nonnegative function where the Borel measure is equal to 1.

Definition 5. Given two measure spaces (Σ_1, X_1, μ) and (Σ_2, X_2), where the measure μ is defined as $\mu : \Sigma_1 \to \mathbb{R}^+$, and a Borel measurable function $f : X_1 \to X_2$, the *push forward* of μ by f for any measurable $E \in \Sigma_2$ is defined as

$$f_{\#}\mu(E) := \mu(f^{-1}(E)).$$

The push forward gives the measure of the pre-image of the input.

3.2 Optimal Transport Formulations

Suppose a group of bees are transporting nectar from flowers in set X to hives in set Y. The bees have limited energy, so there is a certain cost associated with moving nectar from flower x to hive y, $c(x, y)$, per unit of nectar. The problem then becomes which flowers should be supplying which hives.

Of course, bees are probably not using optimal transport to solve this question. Rather, it is said that the question of optimal transport arose during the Napoleonic wars, when Napoleon's troops were digging up roads and use the dug-up earth to build defenses and other works. Discussing this issue in 1781, Gaspard Monge developed the first formulation of the optimal transport problem [23]. But, Monge's formulation provided tricky in many cases. 150 years later, Leonid Kantorovich developed another, more general, formulation to the optimal transport problem.

Both Monge and Kantorovich's formulations are provided in this chapter section, and the equivalence of these two formulations are shown. We also recall the existence, uniqueness, and optimality of the two formulations under certain conditions, and justify the utility of the Kantorovich formulation over the Monge. Definitions and formulations are provided by Ambrosio,

Brué, and Semola [3].

3.2.1 Monge Formulation

In the Monge formulation of the optimal transport problem, we are considering moving probability mass from one probability measure (definition 4) μ on X to another probability measure ν on Y. Moving the mass has a certain cost $c(x, y) : X \times Y \to [0, \infty]$, where c is a Borel measurable function (definition 3) that represents the cost of moving a unit of mass from x to y. The cost function can take many forms, but the most common are the squared difference $c(x, y) = (x - y)^2$ and absolute difference $c(x, y) = |x - y|$, the former being preferred since it is a smooth function.

The last piece required for the Monge formulation is a transformation which maps X to Y, $T : X \to Y$, where the push measure $T_{\#}\mu = \nu$ (definition 5). This tells us that the measure $\nu(E) = \mu(f^{-1}(E)), E \in Y$.

The Monge formulation of the optimal transport problem is as follows:

$$\inf_{T} \int_{X} c(x, T(x)) d\mu(x). \tag{3.1}$$

In words, we want to find the transport map T which minimizes the average cost of transporting μ to ν.

3.2.2 Kantorovich formulation

The Kantorovich formulation of the optimal transport problem is a more general version of Monge's formulation. Rather than transport maps, the Kantorovich formulation relies on a transport plan. Given two probability measures μ on X and ν on Y, let π represent a Borel measure on the cartesian product $X \times Y = \{(x, y) : x \in X, y \in Y\}$, (definitions 4,3). The cartesian product $X \times Y$ has a projection into X called $p_X : X \times Y \to X$ and a projection into Y called $p_Y : X \times Y \to Y$. Let M be the set of all π defined as follows:

$$M(\mu, \nu) = \left\{ \pi \text{ s.t. } \begin{array}{c} p_X^{\#}\pi = \mu \\ p_Y^{\#}\pi = \nu \end{array} \right\}.$$

Hence, M is the set of all probability measures that project into (X, μ) or (Y, ν). M is called the set of *admissable measures* or *transport plans.*

Now we have all the pieces to show the Kantorovich formulation of optimal transport. Given a cost function $c(x, y) : X \times Y \to [0, \infty]$, the optimal transport problem is written as

$$\inf_{\pi \in M} \int_{X \times Y} c(x, y) d\pi(x, y). \tag{3.2}$$

The result of the infimum in (3.2) is called the *Wasserstein distance.*

The Kantorovich formulation has many advantages over the Monge for-

mulation of the optimal transport problem. Firstly, M is nonempty, since the product measure $\mu \times \nu \in M$, (definition 2), but this is not true for transport maps. Additionally, $M(\mu, \nu)$ is convex. Lastly, the Kantorovich formulation has allowed for many extensions on the optimal transport problem, including adding constraints (Martingale optimal transport formulation) and conducting optimal transport on more than 2 spaces (Multi-marginal optimal transportation).

3.2.3 Discrete examples of the Monge and Kantorovich Formulations

In machine learning applications we work with discretized data. First we provide the discrete example of the Monge formulation of optimal transport.

Consider two probability measures μ and ν on the discrete spaces $X = \{x_1, \dots x_N\}$ and $Y = \{y_1, \dots y_N\}$ respectively (definition 1). Define μ and ν as

$$\mu = \sum_{i=1}^{N} \frac{1}{N} \delta_{x_i}, \quad \nu = \sum_{i=1}^{N} \frac{1}{N} \delta_{y_i}.$$

where δ_{a_i} is defined as

$$\delta_{a_i}(A) = \begin{cases} 1, & \text{if } a_i \in A, \\ 0, & \text{if } a_i \notin A. \end{cases}$$

for some set A. Hence, this makes μ and ν two discrete uniform probability distributions.

Now consider the function $T : X \to Y$. In the discrete setting, T is a bijection if and only if the push forward $T_{\#}\mu = \nu$, since

$$T_{\#}\mu = \sum_{i=1}^{N} \frac{1}{N}\delta_{y_i} = \sum_{i=1}^{N} \frac{1}{N}\delta_{T(x_i)}.$$

In this case, given a cost c, the optimal transport map exists since there are a finite number of maps to transport X to Y.

We also provide another way to formulate the problem, from the Kantorovich perspective. Call π_{ij} the amount of mass sent from x_i to y_j. Since the amount of mass transferred to y_j must be $\frac{1}{N}$ by our probability measures, we know that $\sum_i \pi_{ij} = \frac{1}{N}$. Similarly, since each x_j must distribute an amount of mass equivalent to $\frac{1}{N}$, we know that $\sum_j \pi_{ij} = \frac{1}{N}$.

Call the set of all $N \times N$ matrices of π_{ij} which fulfill the above properties the set S. S is also called the set of *bi-stochastic matrices.* It is both compact and convex. Choose a subset $\tilde{S} \subset S$ such that

$$\tilde{S} = \{\pi \in S : \text{ there exists } T \text{ such that } \pi_{ij} = \frac{1}{N}\delta_{jT(i)}\},$$

where δ_{ij} is defined as

$$\delta_{ij} = \begin{cases} 1, & \text{if } i = j, \\ 0, & \text{if } i \neq j. \end{cases}$$

In other words, $\tilde{S}$ is the set of all bi-stochastic matrices where there exists a

bijection T between X and Y.

Given a cost function c, we can define

$$C(T) = \sum_{i=1}^{N} \frac{1}{N} c(x_i, T(x_i)), \quad C(\pi) = \sum_{i,j=1}^{N} c(x_i, y_j)\pi_{ij}.$$

as the cost from a transport map T or a bi-stochastic matrix π. Notice that $C(T) = C(\pi)$ when T is the transport map that induces π. Therefore, we can conclude that

$$\min_{T} C(T) = \min_{\pi \in \tilde{S}} C(\pi) \geq \min_{\pi \in S} C(\pi)$$

In words, we have shown that transport maps can be reforumulated as a bi-stochastic matrix problem. The idea of splitting probability mass as we did using a bi-stochastic matrix in this example is fleshed out in the following section.

3.2.4 Equivilance of transport maps and plans

Now we will show that the formulations in equations (3.1) and (3.2) are equivalent. Given a transport plan T, define the transport plan $\pi_T := (\text{id} \times T)_{\#}\mu$, where $(\text{id} \times T) : X \to X \times Y$, is the map $x \mapsto (x, T(x))$. For a certain cost function c, the cost of a transport plan is given by

$$C(\pi_T) = \int_{X \times Y} c(x, y) d\pi(x, y).$$

Substituting π_T, the cost of the transport plan is

$$C(\pi) = \int_{X \times Y} c(x, y) d((\text{id} \times T)_{\#}\mu)(x, y).$$

Integration of a function ϕ with respect to the push forward of a function $f_{\#}\mu$ can be rewritten as an integral of the composition $\phi \circ f$ with respect the measure μ. We can reformulate $C(\pi_T)$ as the following:

$$C(\pi_T) = \int_X c(x, T(x)) d\mu.$$

This is the same as the the cost of transport map T. We conclude that

$$C(\pi_T) = C(T).$$

Hence, there is always some transport plan that produces the same result as the optimal transport map. Thus

$$\inf (3.1) \geq \inf (3.2);$$

as in, the infimum of the Monge formulation is greater than or equal than the infimum of the Kantorovich formulation.

3.2.5 Uniqueness, Existence, and Optimality.

For both the Monge and Kantorovich formulations, we will recall uniqueness, existence, and optimality of the transport maps and plans. These prop-

erties only come about when we require some conditions on the construction on the transport maps and plans.

First, we will provide a result that shows the existence, uniqueness and optimality of transport maps on the real line $\mathbb{R}$ with measure μ where no single point $\{x\}, x \in \mathbb{R}$ has positive measure. When this is the case, we say that μ has no atom.

Theorem 2. *if μ, ν are probability measures on $\mathbb{R}$, and μ has no atom, then there exists a non-decreasing $T : \mathbb{R} \to \mathbb{R}$ pushing μ on to ν. Any other map S with these properties coincides with T on the support of μ, with at most countably many exceptions. If $c(x, y) = \phi(|y-x|)$ with $\phi : [0, \infty) \to [0, \infty)$ convex and nondecreasing, and if $C(T) < \infty$, then T is an optimal map. If ϕ is strictly convex, T is the unique optimal map.*

For existence, the theorem says that there will always exist a non-decreasing transport map $T : \mathbb{R} \to \mathbb{R}$ that pushes μ on to ν. We want to choose a non-decreasing transport map because it allows for the most efficient movement of mass. For uniqueness, the theorem also says that any other map that satisfying the conditions of existence will be equal to T almost everywhere. Finally, for optimality, the theorem says given a convex and non-decreasing function, then T is optimal, and if it is strictly convex, then T is uniquely optimal. This is because a convex function is guaranteed to have a minimum, but is not unique. A strictly convex function has a single unique minimum, and deviation from the minimum will always result in a larger cost.

Now we will recall the result for the existence of optimal plans for costs which are considered lower semi continuous. A real valued function $f : X \to \mathbb{R}$ is *lower semi continuous* if for all points x the values in a neighborhood N around x are not much lower than $f(x)$, $f(x_i) > y, \forall x_i \in N$. Additionally, we want to think of the space of all measures as a metric space, meaning that each measure has a distance to other measures in the space, namely the Wasserstein distance.

Definition 6. A *metric* on a set S is a function ρ which assigns to every element (a, b) of the Cartesian product $S \times S$ a real number $\rho(a, b)$ satisfying the following rules:

1. $\rho(a, b) \geq 0$ for all $a, b \in S$;
2. $\rho(a, b) = 0$ if and only if $a = b$;
3. $\rho(a, b) = \rho(b, a)$ for all $a, b \in S$;
4. $\rho(a, c) \leq \rho(a, b) + \rho(b, c)$ for all $a, b, c \in S$.

Definition 6 is provided by Cohen et al. [8].

Theorem 3. *Let X, Y be compact metric spaces and let $c : X \times Y \to [0, \infty]$ be lower semicontinuous. Then the minimum in equation* (3.2) *is attained.*

Although the full proof will not be provided, it relies on the compactness of the set of admissable plans M and that the function $\pi \mapsto C(\pi)$ is

lower semicontinuous, since there exists a minimum of lower semicontinuous functions on compact sets. The result tells us that there is always an optimal transport plan under mild assumption that the cost function is lower semicontinous. It also justifies the Kantorovich formulation over the Monge formulation, as the former applies in very general spaces.

3.3 Wasserstein GANs

Generative adversarial networks (GANs) were proposed by Goodfellow et al. [13] in 2014, and later Arjovsky, Chintala, and Bottou [4] incorporated Wasserstein distance into GANs. This section is informed by these two works, and theorems in this section are pulled from these two sources.

3.3.1 GANs Architecture

GANs consist of two neural networks, called the generator G and the discriminator D. They play a game, where the generator produces synthetic data points to trick the discriminator, which evaluates which points are fake and which are real. We will now put this more mathematically. Let $\mathcal{X}$ be a compact metric set and let Σ denote the σ-algebra of all Borel subsets of $\mathcal{X}$. Let Prob$(\mathcal{X})$ denote the set of all probability measures on $\mathcal{X}$. Then we can define a distributions $p_X \in$ Prob$(\mathcal{X})$ for a dataset $X \subset \mathcal{X}$. Using this dataset X, the purpose of the generator is to generate artificial points in the dataset's distribution p_X given some input noise z in distribution p_z. The

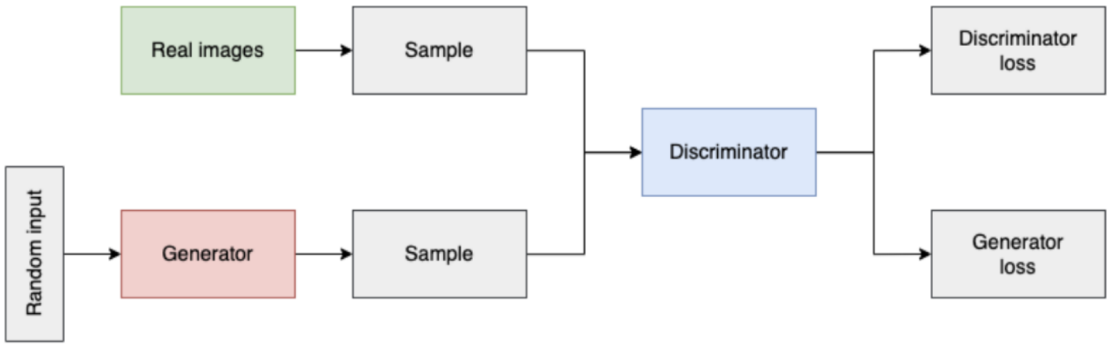


Figure 3.1: GANs Architecture

input z can be any kind of random noise, e.g. Gaussian or uniform. We can write the generator as $G(\theta_G; z)$, to show that G is parameterized by some weights θ_G. Hence, we call the distribution of the artificial data generated by G as $p_G \in \text{Prob}(\mathcal{X})$

The discriminator tries to distinguish between a real point in the dataset $x_{\text{real}} \in X$ and an artificially generated point, the output of $G(\theta_G; z)$, $x_{\text{synthetic}}$. We can write the discriminator as $D(\theta_D; x)$, where θ_D are the weights of D. Hence, $D(x)$ represents the probability that the input came from p_X rather than p_G. Figure 3.1 shows the flow of data in the GANs architecture.

Through training, the goal is to find a pair of discriminator and generator networks that reach an equilibrium. In training the discriminator we want to maximize the probability of assigning the correct labels to the outputs of G and points in X. Simultaneously, we want to train G so that p_X and p_G are very similar in order to fool the discriminator. Specifically, we want to maximize $\log(D(x))$ and minimize $\log(1 - D(G(z)))$. Hence, we can write the mini-max problem for optimizing GANs as

$$\min_G \max_D V(G, D) = \frac{1}{N} \sum_{i=1}^{N} \left[\log D(x_i) + \log(1 - D(G(z_i))) \right], \quad (3.3)$$

labeling the function we are optimizing over as V. In other words, V is an average over the expressions which the generator and discriminator seek to minimize and maximize, respectively. However, the optimal discriminator D^* given a fixed G has a closed formula, given by

$$D_G^*(x) = \frac{p_X(x)}{p_X(x) + p_G(x)}. \quad (3.4)$$

Using this formulation, this allows for re-writing (3.3) in terms of the *Kullback-liebler divergence,* notated $KL(p_1||p_2)$, *Jenson Shannon divergence,* notated by $JSD(p_1||p_2)$, which both provide a distance between two probability distributions. Hence, we provide the following theorem, which relies on the aforementioned reformulation.

Theorem 4. *The global minimum of the value function is achieved if and only if the* $p_G = p_X$.

Here we will not provide the full proof but instead discuss its implications. Considering equation (3.4), when $p_X = p_G$ exactly, then the value always outputted by D is $\frac{1}{2}$. This means the discriminator is unable to differentiate between real or fake data, since they match each other perfectly.

Additionally, the exact value of V simplifies to

$$V(G, D_G^*) = 2JSD(p_G||p_X) - \log(4).$$

Since the Jenson-Shannon divergence provides a distance between probability distributions, it always nonnegative, and equals 0 when $p_G = p_X$. Hence, the optimization problem amounts to minimizing the distance between p_G, p_X, and the minimum is reached when $p_G = p_X$.

Just like in section 2.3.3, we can use back propagation and gradient descent on V to find a solution to satisfy this problem. Additionally, because of the expressivity of neural networks discussed in 2.1, GANs can express many different functions making them effective in many different applications. However, training them can be difficult when trying to synchronize the learning of both the generator and the discriminator. For example, at the beginning of training the generator may generate outputs that are very different from the data set, which the discriminator may be able to reject very easily. In this case, rather than minimizing $\log D(x_i) + \log(1 - D(G(z)))$, a solution is maximizing $D(G(z))$ instead.

3.3.2 Wasserstein Distance in GANs

The paper Arjovsky, Chintala, and Bottou [4] developed the idea of using Wasserstein distance as the loss function in GANs, known as the WGANs. Note that in the WGAN, we often call the discriminator the critic. Using

Wasserstein distance is theoretically a better loss function than the KL divergence or Jenson-Shannon divergence.

Firstly, when given a sequence of probability distributions $p_n, n \in \mathbb{N}$, the Wasserstein distance will converge weakly where as the Jenson Shannon divergence and KL Divergence will converge strongly.

Theorem 5. *Let p be a probability distribution on a compact space $\mathcal{X}$ and let $p_n, n \in \mathbb{N}$, be a sequence of distributions on $\mathcal{X}$. Then considering all limits as $n \to \infty$,*

1. *If $KL(p_n||p) \to 0$ then $JSD(p_n||p) \to 0$.*

2. *$W(p_n, p) \to 0$ if and only if p_n converges to p in distribution.*

In this case, weak and strong refer to topological spaces, which we will not cover in this thesis. Wasserstein distance convergence as weak convergence means that there are more continuous functions to measure distance between probability distributions. Practically speaking, if G produces examples such that p_G is close to p_X then it will converge. This means that minimizing the Wasserstein distance will allow the generator to learn the ideal distribution.

Secondly, KL divergence and Jenson Shannon divergence are not continuous when the two probability distributions have disjoint supports. Additionally, the same conclusion holds when the intersection of the two supports is contained in a set with measure 0. In contrast, the Wasserstein

distance provides a better gradient than the KL divergence or Jenson Shannon divergence because it is continuous and differentiable under very few assumptions.

Theorem 6. *Let $G(\theta_G; z)$ be any feed forward network parameterized by θ_G and p_z be some distribution where the average value of z is finite. Then the Wasserstein distance $W(p_g, p_X)$ is continuous and differentiable everywhere.*

Although we will not show the full proof, it uses the bounded convergence theorem to show that W is continuous and shows that W is locally Lipschitz to show that W is differentiable, as gradients must be finite.

Definition 7. Let (X, d) and (Y, ρ) be two metric spaces with at least two points. A function $f : X \to Y$ is called *locally Lipschitz* if at each $x \in X$ there exists $\delta > 0$ such that the restriction of f to the open ball with radius δ and center x denoted by S is Lipschitz:

$$\sup \left\{ \frac{\rho(f(a), f(b))}{d(a, b)} : a \neq b, \{a, b\} \in S \right\} < \infty.$$

This definition is provided by Beer and Garrido [6]. Locally Lipschitz is a way to describe continuous functions. In practice, there are many ways to satisfy the Lipschitz constraint. In the original Wasserstein GANs, weights are clipped during training. Gulrajani et al. [16] proposes another method to maintain the Lipschitz constraint by adding penalty terms to gradients

instead. Gradient penalty rather than clipping may lead to better training stability.

Theorem 6 is not necessarily true for other distances such as the Jenson-Shannon distance or KL divergence. Of course, as discussed in section 2.3.3, the gradient of the loss function is employed in every step of training. And in fact, training by backpropagation is one of the advantages of GANs. Hence, as a continuous and differentiable function, the Wasserstein distance proves to be a more reasonable loss function than any other distance.

In the big picture, these theoretical guarantees provide multiple practical benefits. The continuity and differentiability of the Wasserstein distance means that it is possible to train the discriminator until optimality, as the gradient will always exist. Additionally, because the discriminator is able to train to optimality, the generator does not experience *mode collapse.* Mode collapse is when the generator only creates a few kinds of examples instead of a diverse set of examples from the whole distribution, as it becomes stuck in the local minima. But when the critic continuously improves, it will learn to reject the examples that the generator continues to provide. Hence we are able to use a better gradient for the generator instead of remaining in the local minima.

Due to the ability to train the discriminator until optimality, the WGAN also proves to be more stable than the standard GAN. In the previous section we mentioned how it is difficult to synchronize the learning of the dis-

criminator with the learning of the generator. But when the discriminator is trained to optimality, then the generator is provided a loss like any other network and is able to learn upon that. Therefore, we do not need to balance or synchronize the learning of the generator and discriminator of a WGAN in the same manner as GANs. However, there are a few things that can reduce the stability of training WGANs, including a high learning rate.

Chapter 4

Application to WAI Diagnostics

4.1 The WAI Database

Wideband Acoustic Immitance (WAI) data serve as a noninvasive method in ear diagnostics. WAI data can be presented in many different formats which have significance to ear scientists such as absorbance or impedence, and these measurements are constructed from ear canal pressure measurements.

Dr. Susan Voss's lab in the Smith Engineering Department manages a database of WAI measurements that serve as a repository for WAI research in which measurements are submitted from many different academic sources. Measurements are also collected using different types of instruments, which create heterogeneity in the structure of measurements. Measurements are taken from humans at different stages of life (Infant, Adult, etc.) as well as from ears with different conditions such as fluid build-up. When there is a diagnosed condition within an ear, they are labeled *abnor-*

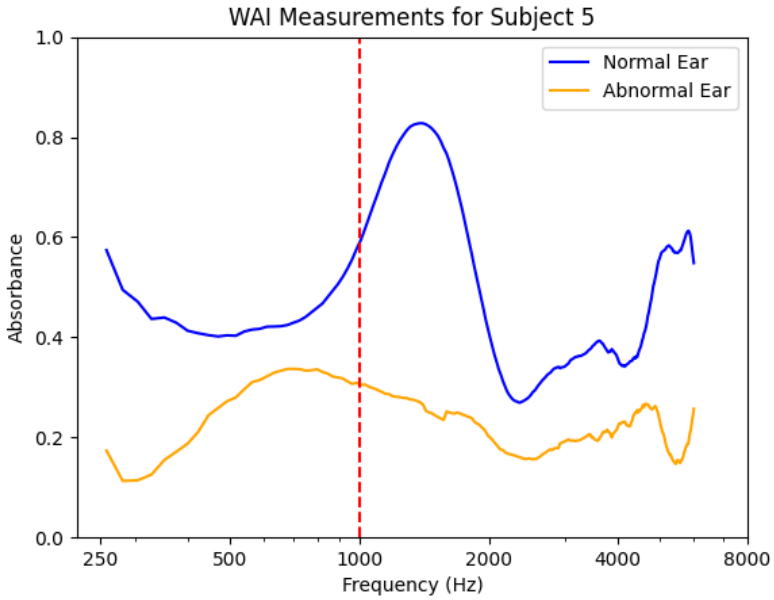


Figure 4.1: Two WAI measurements taken from the same infant, subject 5. The right ear is normal and the left ear is abnormal [24].

mal ears, and otherwise they are labeled *normal* ears.

The measurements depicted in this section are from normal and fluid filled infant ears in the paper Voss et al. [24]. Figure 4.1 is of an abnormal and normal ear absorbance measurement versus frequency. The abnormal measurement is taken from an infant (subject 5) from its left ear with a HearID instrument. The normal measurement is taken from the same infant but from its right ear.

Frequencies range from 200-8000 Hz depending on the tool used for measurement collection. In figure 4.1, both measurements are taken with a HearID instrument, which measures frequencies between 220 and 6000 Hz, with measurements taken at 246 unique frequencies.

Often, measurements below 1000 Hz are affected by acoustic leaks [15].

Leaks refer to noisy or faulty measurements that occur at very low frequencies. Hence their measurements at those frequencies can be considered corrupted and invalid. The following figure depicts a measurement with leaks.

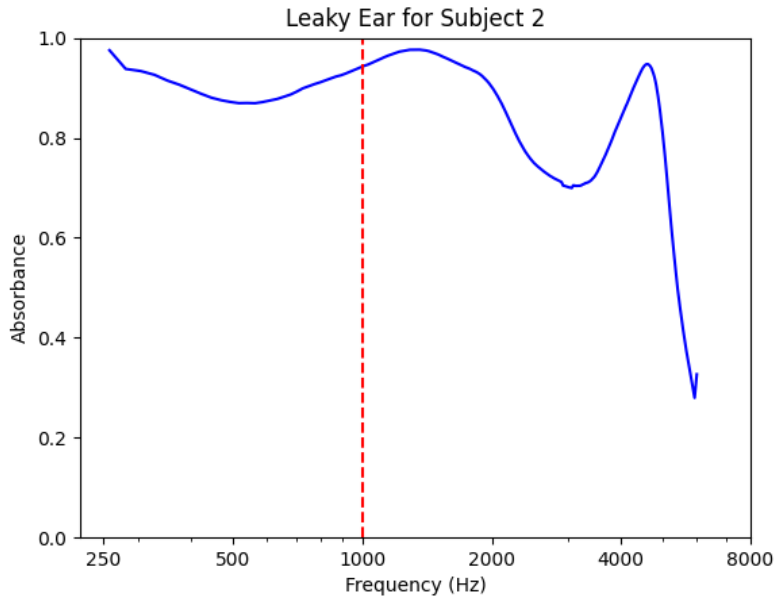


Figure 4.2: Leaky normal measurement of an Infant. The measurement is taken from subject 2 from [24]

Figure 4.2 shows a measurement taken from an infant's abnormal ear (subject 2), and the measurement is again taken with HearID. Notice that the measurement has a very high absorbance below 1000 hz compared to measurements in figure 4.1. According to Voss et al. [24], these features point towards a leaky measurement.

The example given for normal and abnormal ear in figure 4.1 look very different, but in general differences between the two classes may not be so stark. Even to the trained eye, it can be difficult to determine how exactly normal and abnormal ear measurements differ. Additionally, when there

is noise present in measurements, the difference between an abnormal and normal ears becomes increasingly unclear. Experts like Dr. Voss are often able to determine when measurements look roughly normal or abnormal, but not all medical professionals have access to this training. Hence, the need for an automated tool to differentiate between abnormal and normal ears will allow WAI measurements to be an effective, noninvasive measurement for a wider audience in ear diagnostic settings.

4.2 Data and Pre-processing

Due to the various sources of measurements in the WAI database, we chose to narrow the scope of this tool to a diagnostic tool for infant ears. Specifically we chose to use data from five separate publications: Aithal, Kei, and Driscoll [1], AlMakadma and Prieve [2], Ellison et al. [11], Sanford et al. [20], and Voss et al. [24].

Publication	**# Normal Ears**	**# Abnormal Ears**	**Instrument**
Aithal et. al (2014)	208	95	Titan
Al Makadma et. al (2024)	77	0	Titan, HearID
Ellison et. al (2012)	42	43	Titan
Sanford et. al (2009)	375	80	HearID
Voss et. al (2016)	22	15	HearID
TOTAL:	**724**	**233**	

Table 4.1: WAI Data Sources for infant ears.

These are the existing datasets that include absorbance measurements that are labeled as abnormal and normal for infants. They all have measurements over a wide frequency range. databases were chosen because of their number of abnormal ears as well as having a large number of frequencies in each measurement.

After selecting which published datasets will form our database, we cleaned the data to ensure that they have a consistent format for training. Measurements were taken by different instruments, two standard machines being Titan and HearID. The instrument used in each published dataset is listed in table 4.1. Titan measurements were between 220 to 8000 Hz, whereas HearID measurements were only up to 6000 Hz. Additionally, not only do different instruments collect a different number of frequency measurements for an ear, but the intervals of those frequencies are also different. Therefore we used a technique called interpolation by splines to create a consistent set of measurements for training.

Interpolation by splines is a method for constructing a smooth curve through a series of discrete data points by fitting piecewise low-degree polynomials on the discrete points [26]. These polynomials are fitted such that the curve and its first and second derivatives are continuous at the points where the segments meet. For a set of datapoints (x_i, y_i), the general for-

mula for cubic splines over the interval $[x_i, x_{i+1}]$ is given by

$$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3,$$

where coefficients a_i, b_i, c_i, d_i must be computed to ensure that the first and second derivative of the function are smooth. After interpolation, we can evaluate each piece-wise function on a given domain F such that the image $\bigcup S_i(F)$ essentially follows the curve of the original WAI absorbance measurement. Since the HearID measurements have more frequencies, we chose F as the HearID frequencies and interpolated over the measurements taken by Titan instruments. To interpolate the Titan data, we used the `scipy` function `InterpolatedUnivariateSplines` with a cubic spline.

We also removed seven measurements from the dataset shown in table 4.2, due to being identified as having too low absorbance in the frequency range between 1000-6000 Hz. This indicated a potential blocked probe when the measurement was collected, which result in a faulty data point.

After constructing a dataset with consistent frequency vectors, we also identified potential leaks in the dataset. Since leaky sections of the measurement are not representative of the actual state of the ear, we created a new dataset by truncating F to frequencies within the interval $(1000, 6000)$ Hz.

Lastly, we prepared an experimental dataset containing the slope m_i of

Paper Name	Subject	Session	Status	Reason
Aithal et. al (2014)	17	1	Normal	Probe
Aithal et. al (2014)	139	1	Abnormal	Probe
Aithal et. al (2014)	150	1	Abnormal	Probe
Aithal et. al (2014)	163	1	Abnormal	Probe
Ellison et. al (2012)	18	1	Normal	Probe
Ellison et. al (2012)	124	1	Abnormal	Probe

Table 4.2: Removed measurements from WAI dataset due to suspected blocked probe.

Dataset Characteristic	
# of Measurements	1492
# of Normal	1242
# of Abnormal	250
# Frequencies per Measurement	214

Table 4.3: Summary of Interpolated WAI Dataset with leaks and faulty data points removed.

each point x_i in a measurement:

$$m_i = \frac{x_i - x_{i-1}}{2}.$$

The resulting derivative vector represents the slope of the absorbance measurement with respect to frequency, $\frac{dA}{df}$. We stacked each measurement with the derivative so that each input to the CNN had the dimensions $2 \times n$.

Ultimately, we are working with a small dataset of 1492 measurements each with 214 frequencies collected from infant ears. Within the dataset, we have 1242 normal measurements and 250 abnormal measurements. The dataset characteristics are summarized in the table 4.3.

4.3 Evaluation methods

Here we will briefly describe the metrics we use for evaluating the performance of the neural networks.

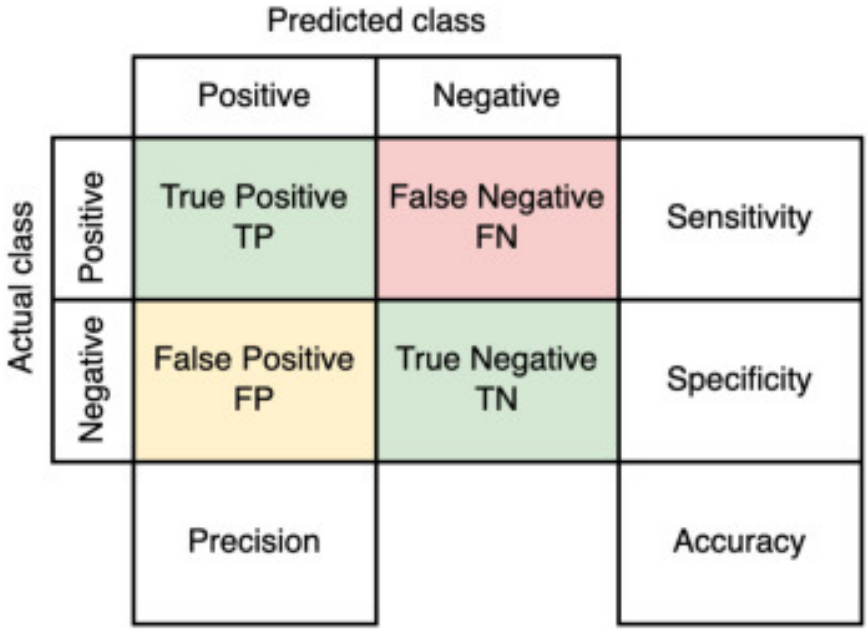


Figure 4.3: Confusion Matrix Explained [7]. In our application, a "positive" refers to an abnormal ear, and "negative" refers to a normal ear.

Confusion Matrix. A confusion matrix visualizes the four following elements in a binary classification task: true positives (TP), the correctly classified abnormal measurements; false positives (FP), the incorrectly classified normal measurements; true negatives (TN), the correctly classified normal measurements; and false negatives (FN), the incorrectly classified abnormal measurements. These are shown in the confusion matrix above.

Accuracy. Accuracy describes what percentage of points were classified correctly.

$$\text{Accuracy} = \frac{\text{TP} + \text{FP}}{\text{TP} + \text{FP} + \text{NP} + \text{NP}}.$$

Sensitivity. This is the ability of the network to correctly classify positives. It is also called recall. It answers the question: Of all the actual positives, how many were we able to identify?

$$\text{Sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}}.$$

Specificity. Specificity is the true negative rate. It answers the question: Of all the actual negatives, how many were we able to identify?

$$\text{Specificity} = \frac{\text{TN}}{\text{FP} + \text{TN}}.$$

Precision. Precision is the proportion of positive identifications that were actually correct. It answers the question: Of all the classified positives, how many were actually positive?

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}.$$

F1 Score. Provides a harmonic mean of both the sensitivity and precision. That way, we have one number that takes into account sensitivity and precision. The F1 score ranges from zero to one, and is used as a score for the model's performance with one being the best performance possible.

$$\text{F1} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}.$$

The most important metrics for evaluating these networks are the F1 score and the accuracy rate. In general, we want our model to be effective at classifying any measurement, but we also want to ensure that our networks are good at identifying abnormal ears, so specificity is also an important quantity.

4.4 CNN Results with Real Data

There does not exist any theorem or formula to determine the ideal architecture for a neural network, so the optimal architecture must be found through trial and error. We trained many different CNNs to determine what training schemes and architectures to use. In table 4.4, we provide the metrics for evaluating the classification power of different architectures. We tested different hyper-parameters, including: 1, 2, and 3 convolutional layers; 1, 2, and 3 dense layers; and kernel sizes of 3, 5, and 7. Each model was trained for 500 epochs with a fixed learning rate of 0.0001. These models were trained and tested on a imbalanced dataset of real ear measurements with no derivative. The table is sorted by F1 scores, with the 10 CNNs with the highest F1 scores shown. The top performers also have the highest training and testing accuracy.

CNN Layers	Dense Layers	Kernel Size	Training Accuracy	Test Accuracy	Sensitivity	Specificity	Precision	F1
3.0	2.0	3.0	0.957	**0.8696**	**0.5**	0.9402	0.9077	**0.9237**
1.0	1.0	5.0	0.913	0.8662	0.4167	**0.9522**	0.8951	0.9228
2.0	1.0	7.0	0.934	0.8662	0.4583	0.9442	0.9011	0.9222
2.0	1.0	5.0	0.9308	0.8662	0.4792	0.9402	0.9042	0.9219
2.0	2.0	5.0	0.935	0.8662	0.4792	0.9402	0.9042	0.9219
1.0	1.0	3.0	0.9109	0.8629	0.3958	0.9522	0.8918	0.921
1.0	1.0	7.0	0.9067	0.8629	0.4375	0.9442	0.8977	0.9204
1.0	2.0	5.0	0.9235	0.8629	0.4375	0.9442	0.8977	0.9204
2.0	3.0	7.0	0.9507	0.8629	0.4792	0.9363	0.9038	0.9198
3.0	1.0	3.0	0.935	0.8629	0.4792	0.9363	0.9038	0.9198

Table 4.4: Metrics for 10 best performing CNN architectures trained on real imbalanced absorbance data.

In the experiments above, we trained on the imbalanced dataset, reflecting the imbalance of abnormal and normal ears. We took the existing dataset of 1492 measurements and did an 80/20 split so that 80% of the data was used for training the CNN and and 20% of the data was reserved for testing. Within the dataset, we have 1242 normal ears and 250 abnormal ears, so in training we used 991 normal ears and 202 abnormal ears and in testing we reserved 48 abnormal ears and 251 normal ears. Because of the ratio between normal and abnormal ears in the imbalanced dataset, we would expect around one sixth of the testing data, approximately 48 measurements, to be classified as abnormal regardless of if the network is correct or not. However, in the experiments conducted for different architectures in table 4.4 we see that all networks have very low sensitivity compared to specificity, which indicates the models are much better at identifying normal measurements than abnormal measurements.

In our next experiment, we tried different input formats to the CNN. We tried training a network on the original imbalanced data, the experimental dataset which included the derivative of each measurement in the imbalanced dataset, and balanced data where we removed normal data points so the number in each class was equal. For all these networks, we trained a CNN with a drop-out rate of 0.1, learning rate of 0.0001, CNN layers 3, dense layers 3, and kernel size of 3, for 500 epochs.

In figure 4.4, we see that only 21 ears are classified as abnormal by the network trained on imbalanced absorbance data. As in our previous experiments, this indicates a bias towards classifying data as normal. As a diagnostic tool for abnormal ears, this would result in missing more than 50% of abnormalities in baby ears.

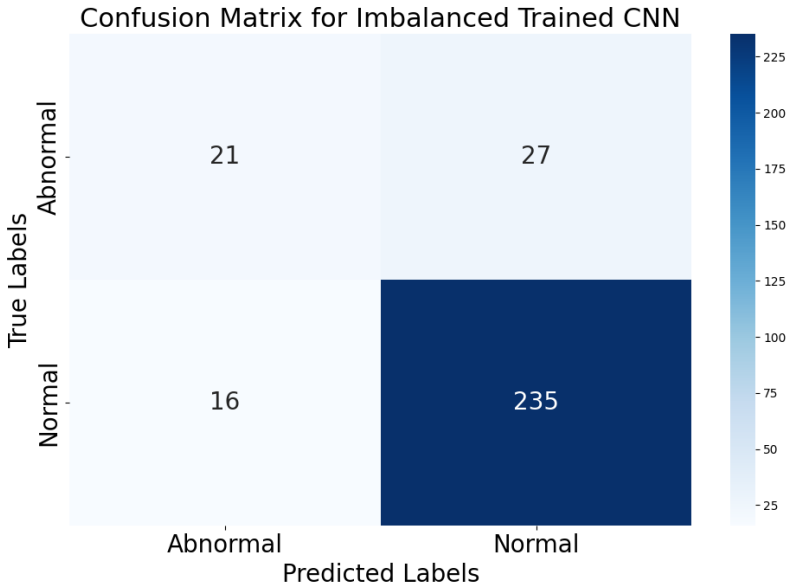


Figure 4.4: Confusion matrix for network trained on imbalanced absorbance training dataset with drop-out rate of 0.1, learning rate of 0.0001, CNN layers 3, dense layers 3, and kernel size of 3, for 500 epochs.

Next we attempted fully balancing the training dataset. We randomly selected 202 normal ears from the set of normal training measurements and trained a new CNN with the same hyperparameters on those 202 normal ears and the previously selected 202 abnormal ears. We reserved the same 48 abnormal ears and 251 normal ears for testing. In 4.5 we provide the confusion matrix for this new network. We see that 63 ears are classified as abnormal, showing that the network is now biased towards an abnormal classification. Yet 18 real abnormal ears, nearly 30% of the testing abnormal set, are still classified as incorrectly.

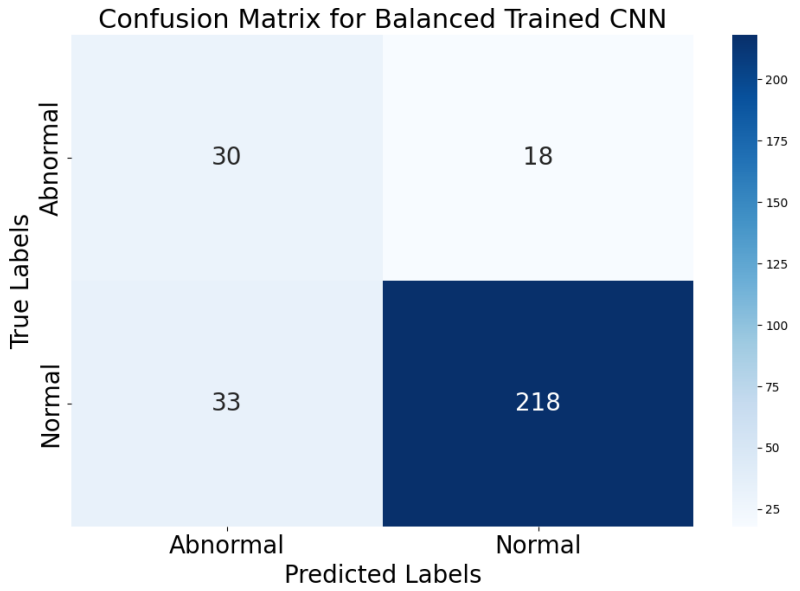


Figure 4.5: Confusion matrix for network trained on balanced absorbance training dataset with drop-out rate of 0.1, learning rate of 0.0001, CNN layers 3, dense layers 3, and kernel size of 3, for 500 epochs.

The results in figures 4.4 and 4.5 indicate both that a significant imbalance between the two classes results in a bias towards normal ears and that a lack of abnormal data inhibits the CNN from understanding the true struc-

ture of abnormal data. We cannot simply reduce the number of normal data for training because more normal data points become classified as abnormal.

The results of the three data structure experiments are in table 4.5. In the table we see that the derivative experiment did not result in an increase in performance.

Structure	Training Accuracy	Test Accuracy	Sensitivity	Specificity	Precision	F1
Imbalanced	0.9518	0.8562	0.4375	0.9363	0.8969	0.9162
Balanced	0.8731	0.8294	0.625	0.8685	0.9237	0.8953
Derivative	0.9497	0.8462	0.4375	0.9243	0.9098	0.9098

Table 4.5: Performance metrics for networks trained on imbalanced, balanced, and imbalanced with derivative training datasets.

In our next experiments, we will show the results of the top five architectures in the following experiments. We tested different learning rates, including 0.001, 0.0001, and 0.00001, shown in in table 4.6. We also tried scheduled learning rates, with decay at a rate of 0.8 every 100 epochs, shown in table 4.7. Again, these models were trained for 500 epochs.

In general, scheduled learning rates did not provide much benefit to the accuracy an F1 scores of the networks. Overall, a lower learning rate resulted in better F1 scores, but there appeared to be a trade off between the specificity and sensitivity. For lower learning rates, there were lower sensitivities but higher specificities. The opposite was true for higher learning rates. This indicates that the middle learning rate may be most effective.

Next we tested different training schemes. First we tested different training epoch lengths, including 250, 500, 1000, and 2000 epochs, shown in

Learning Rate	CNN Layers	Dense Layers	Kernel Size	Training Accuracy	Test Accuracy	Sensitivity	Specificity	Precision	F1
1e-05	1.0	1.0	5.0	0.8889	0.8696	0.4167	0.9562	0.8955	0.9249
1e-05	2.0	1.0	7.0	0.8889	0.8696	0.4167	0.9562	0.8955	0.9249
1e-05	2.0	1.0	5.0	0.8931	0.8696	0.4167	0.9562	0.8955	0.9249
1e-05	2.0	2.0	5.0	0.892	0.8696	0.4167	0.9562	0.8955	0.9249
1e-05	3.0	2.0	3.0	0.8931	0.8662	0.4167	0.9522	0.8951	0.9228
0.0001	1.0	1.0	5.0	0.9088	0.8662	0.4167	0.9522	0.8951	0.9228
0.0001	2.0	2.0	5.0	0.9423	0.8629	0.4583	0.9402	0.9008	0.9201
0.0001	2.0	1.0	5.0	0.9308	0.8595	0.4583	0.9363	0.9004	0.918
0.0001	2.0	1.0	7.0	0.9298	0.8595	0.4583	0.9363	0.9004	0.918
0.0001	3.0	2.0	3.0	0.9455	0.8562	0.5	0.9243	0.9062	0.9152
0.001	3.0	2.0	3.0	0.9927	0.8495	0.4792	0.9203	0.9023	0.9112
0.001	2.0	1.0	7.0	0.9874	0.8495	0.5208	0.9124	0.9087	0.9105
0.001	1.0	1.0	5.0	0.9602	0.8428	0.4792	0.9124	0.9016	0.9069
0.001	2.0	1.0	5.0	0.9717	0.8294	0.5	0.8924	0.9032	0.8978
0.001	2.0	2.0	5.0	0.979	0.8027	0.4583	0.8685	0.8934	0.8808

Table 4.6: Performance metrics for best performing networks with different fixed learning rates, trained on imbalanced absorbance data.

Schedule Learning Rate	CNN Layers	Dense Layers	Kernel Size	Training Accuracy	Test Accuracy	Sensitivity	Specificity	Precision	F1
1e-05	3.0	2.0	3.0	0.8931	0.8696	0.4167	0.9562	0.8955	0.9249
1e-05	2.0	1.0	7.0	0.8899	0.8696	0.4167	0.9562	0.8955	0.9249
1e-05	2.0	1.0	5.0	0.8836	0.8696	0.4167	0.9562	0.8955	0.9249
1e-05	2.0	2.0	5.0	0.8931	0.8696	0.4167	0.9562	0.8955	0.9249
1e-05	1.0	1.0	5.0	0.8857	0.8662	0.3958	0.9562	0.8922	0.9231
0.001	1.0	1.0	5.0	0.9549	0.8662	0.4583	0.9442	0.9011	0.9222
0.0001	2.0	2.0	5.0	0.9329	0.8662	0.4583	0.9442	0.9011	0.9222
0.0001	1.0	1.0	5.0	0.9025	0.8629	0.3958	0.9522	0.8918	0.921
0.0001	2.0	1.0	5.0	0.9329	0.8629	0.4167	0.9482	0.8947	0.9207
0.0001	2.0	1.0	7.0	0.9266	0.8629	0.4167	0.9482	0.8947	0.9207
0.001	2.0	2.0	5.0	0.9853	0.8462	0.5	0.9124	0.9051	0.9087
0.001	3.0	2.0	3.0	0.9927	0.8462	0.5417	0.9044	0.9116	0.908
0.0001	3.0	2.0	3.0	0.9497	0.8395	0.4167	0.9203	0.8919	0.9059
0.001	2.0	1.0	7.0	0.9895	0.8328	0.4792	0.9004	0.9004	0.9004
0.001	2.0	1.0	5.0	0.9853	0.8328	0.4792	0.9004	0.9004	0.9004

Table 4.7: Performance metrics for best performing networks with different scheduled learning rates, trained on imbalanced absorbance data.

Epochs	CNN Layers	Dense Layers	Kernel Size	Training Accuracy	Test Accuracy	Sensitivity	Specificity	Precision	F1
250.0	1.0	1.0	5.0	0.8994	0.8696	0.4375	0.9522	0.8985	0.9246
250.0	2.0	1.0	7.0	0.9099	0.8662	0.3958	0.9562	0.8922	0.9231
500.0	1.0	1.0	5.0	0.9036	0.8662	0.4167	0.9522	0.8951	0.9228
500.0	2.0	1.0	5.0	0.9266	0.8662	0.4583	0.9442	0.9011	0.9222
1000.0	1.0	1.0	5.0	0.9224	0.8662	0.4583	0.9442	0.9011	0.9222
1000.0	2.0	1.0	7.0	0.9539	0.8629	0.4167	0.9482	0.8947	0.9207
500.0	2.0	1.0	7.0	0.9266	0.8629	0.4167	0.9482	0.8947	0.9207
250.0	2.0	2.0	5.0	0.9172	0.8629	0.4167	0.9482	0.8947	0.9207
2000.0	1.0	1.0	5.0	0.9476	0.8629	0.4792	0.9363	0.9038	0.9198
250.0	2.0	1.0	5.0	0.9046	0.8595	0.375	0.9522	0.8885	0.9192
250.0	3.0	2.0	3.0	0.9256	0.8595	0.4167	0.9442	0.8943	0.9186
500.0	2.0	2.0	5.0	0.9486	0.8595	0.4583	0.9363	0.9004	0.918
1000.0	2.0	1.0	5.0	0.9539	0.8595	0.4583	0.9363	0.9004	0.918
1000.0	2.0	2.0	5.0	0.9727	0.8595	0.4792	0.9323	0.9035	0.9176
2000.0	2.0	1.0	7.0	0.979	0.8495	0.4583	0.9243	0.8992	0.9116
500.0	3.0	2.0	3.0	0.9444	0.8495	0.4792	0.9203	0.9023	0.9112
1000.0	3.0	2.0	3.0	0.9801	0.8428	0.4375	0.9203	0.8953	0.9077
2000.0	2.0	1.0	5.0	0.9654	0.8395	0.4792	0.9084	0.9012	0.9048
2000.0	3.0	2.0	3.0	0.9927	0.8395	0.5	0.9044	0.9044	0.9044
2000.0	2.0	2.0	5.0	0.9822	0.8294	0.5208	0.8884	0.9065	0.8974

Table 4.8: Performance metrics for best performing networks with different epoch lengths, trained on imbalanced absorbance data.

table 4.8.

In general we see that very long training lengths results in higher training accuracy, but does not necessarily result in improved accuracy when testing. This points to overfitting to the data in the training set, as the networks are not able to generalize well compared to networks with lower training accuracy. However, longer training lengths result in higher sensitivities in general. Through these results, we determined that 500 epochs are generally sufficient to learn from the amount of data present, as well as that too many epochs can result in overfitting.

Batch Size	CNN Layers	Dense Layers	Kernel Size	Training Accuracy	Test Accuracy	Sensitivity	Specificity	Precision	F1
32.0	2.0	1.0	5.0	0.9361	**0.8729**	**0.5**	0.9442	0.908	**0.9258**
128.0	2.0	1.0	5.0	0.9172	0.8662	0.3958	0.9562	0.8922	0.9231
32.0	2.0	1.0	7.0	0.9423	0.8662	0.4792	0.9402	0.9042	0.9219
64.0	2.0	1.0	5.0	0.9245	0.8629	0.3958	0.9522	0.8918	0.921
128.0	1.0	1.0	5.0	0.9046	0.8629	0.4167	0.9482	0.8947	0.9207
64.0	2.0	1.0	7.0	0.9214	0.8629	0.4583	0.9402	0.9008	0.9201
64.0	1.0	1.0	5.0	0.9067	0.8595	0.375	0.9522	0.8885	0.9192
32.0	1.0	1.0	5.0	0.9088	0.8595	0.375	0.9522	0.8885	0.9192
64.0	3.0	2.0	3.0	0.9476	0.8629	0.5208	0.9283	0.9102	0.9191
128.0	2.0	2.0	5.0	0.9256	0.8595	0.3958	0.9482	0.8914	0.9189
64.0	2.0	2.0	5.0	0.9444	0.8595	0.4583	0.9363	0.9004	0.918
128.0	2.0	1.0	7.0	0.9109	0.8562	0.4167	0.9402	0.8939	0.9165
128.0	3.0	2.0	3.0	0.9455	0.8528	0.3958	0.9402	0.8906	0.9147
32.0	3.0	2.0	3.0	0.9675	0.8428	0.4167	0.9243	0.8923	0.908
32.0	2.0	2.0	5.0	0.9612	0.8395	0.5	0.9044	0.9044	0.9044

Table 4.9: Performance metrics for best performing networks with different batch sizes, trained on imbalanced absorbance data.

We also tested different batch sizes, 32, 64 and, 128, on a model with learning rate of 0.0001, dropout rate of 0.1, 3 convolutional layers, 2 dense layers, and a kernel size of 3, shown in table 4.9. This CNN was again trained with 500 epochs. We determined that batch size does not necessarily provide returns in the F1 score. In some runs, a batch size of 32 results in very high F1 scores, but in some runs it results in lower F1 scores.

After these experiments, we identified our best performing network in table 4.10. We also include the following graphs for the best performer: a confusion matrix accompanied with the graphs of the classified measurements, loss graph for the network, and accuracy graph. This network correctly classifies more measurements than any other model.

Hyperparameter	
Batch size	32
Epochs	500
Dropout Rate	0.1
CNN Layers	2
Dense Layers	1
Kernel Size	5
Fixed Learning rate	0.0001

Table 4.10: Parameters for the overall best performing CNN.

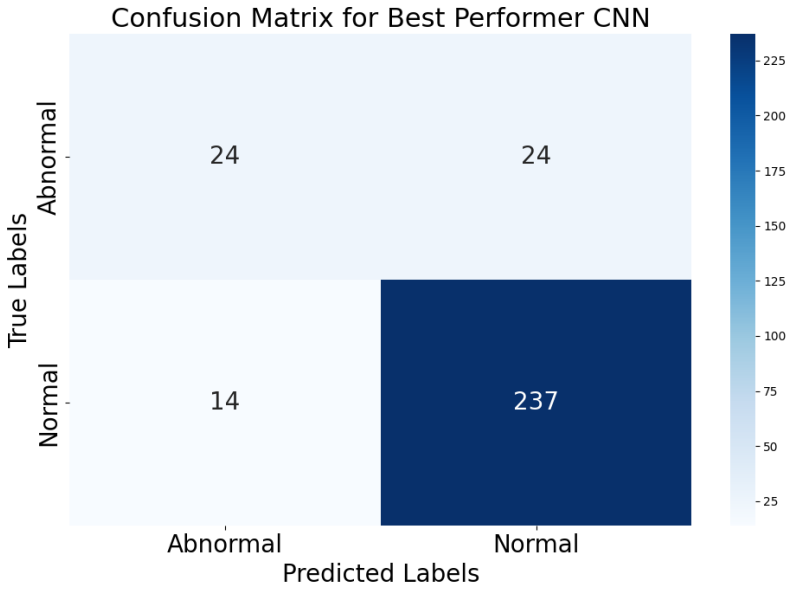


Figure 4.6: Confusion Matrix for the best CNN trained with parameters in table 4.10.

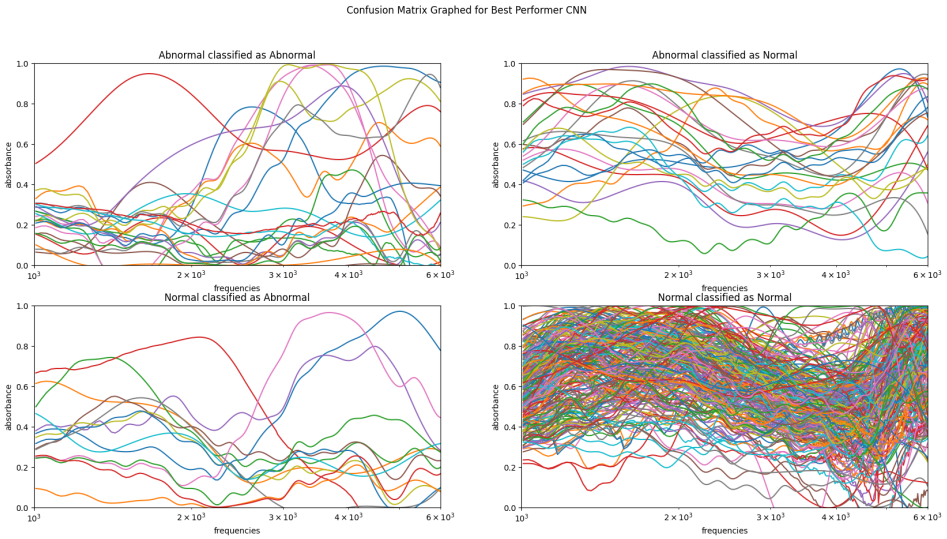


Figure 4.7: Graphed testing data corresponding to the confusion matrix for the best performing CNN.

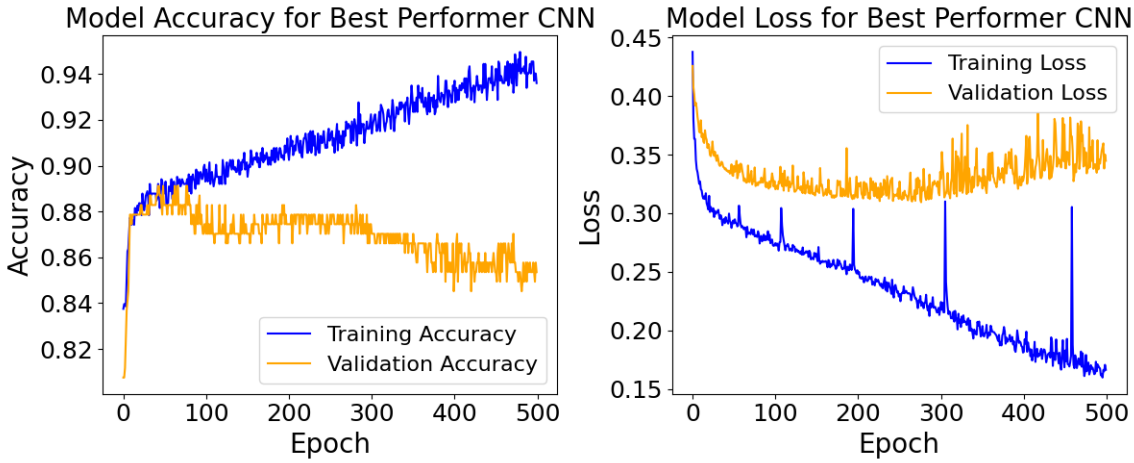


Figure 4.8: Accuracy and Loss graphs for the best performing CNN.

Figure 4.6 indicates that there are still 38 measurements that are being incorrectly classified. The measurements are visualized in figure 4.7. Figure 4.8 indicate that the network still has quite a bit to learn in the data, since accuracy and loss in blue are still increasing. However as train time goes on, the generalization gap and the validation loss increases. Though it does

not impact the result on the test data negatively, it indicates that the model is very prone to overfitting. The CNN would benefit from training on more data than what currently exists in the dataset.

The full network and training scheme can be found in the code supplement.

4.5 WGAN Results

Our implementation of WGAN is informed by the Keras implementation in Nain [18]. The code can be found in the code supplement. The WGAN is structured in an adversarial structure between two neural networks, the generator and the discriminator as discussed in section 3.3.2. Like in the CNN used for diagnosis in the previous section, the generator and discriminator are convolutional neural networks to extract information in small neighborhoods of the input. Specifically, we used a generator with 2 convolution layers and 1 dense layer, and a critic with 2 convolution layers and 1 dense layers, and both models had kernel sizes of 3. Unless stated otherwise, all models used learning rates of 0.0002 for both the generator and critic.

The implementation uses the technique called gradient penalty to maintain the Lipchitz continuity condition mentioned in Gulrajani et al. [16], discussed in section 3.3.2. Gradient Penalty allows the discriminator to converge more neatly, as opposed to the technique of weight clipping discussed

in [4].

When training the WGAN we want to train to optimality. As mentioned in [4], in WGAN we can train the critic until optimality and then the generator will improve. In the following figures we show the loss function graphs for both the generator (orange) and critic (blue) for two different epochs to determine when the critic is fully trained. The WGANs are trained on the abnormal WAI data.

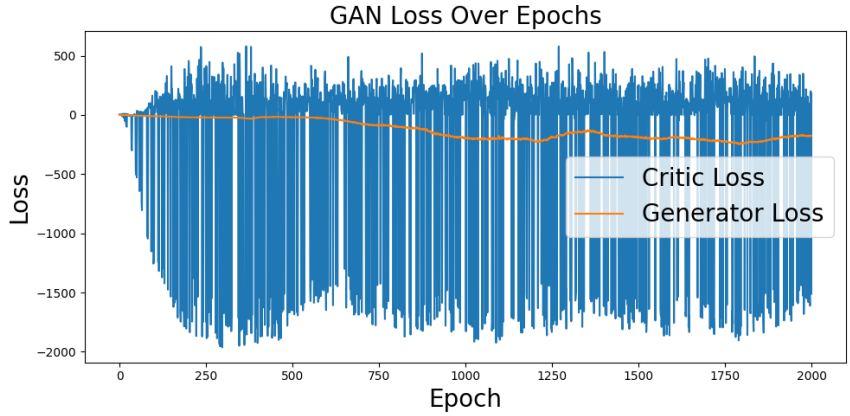


Figure 4.9: WGAN trained for 2000 epochs on abnormal data.

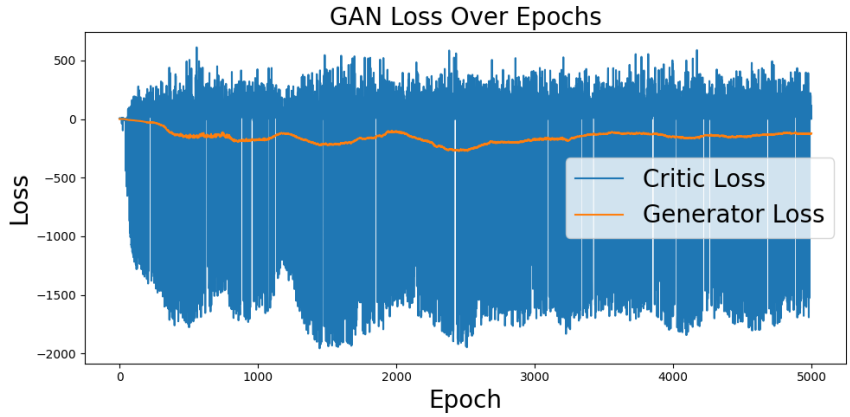


Figure 4.10: WGAN trained for 5000 epochs on abnormal data.

In the loss function graphs, the critic loss, shown in blue, is highly un-

stable. Although we see a steep decrease until 250 epochs, the blue critic loss function oscillates heavily. In contrast, the generator loss, shown in orange, is very stable. Additionally, the model which was trained for longer produced better results.

Arjovsky, Chintala, and Bottou [4] indicate that a higher learning rate in the critic can result in the critic unable to train to optimality. Figures 4.9 and 4.10 show networks trained with learning rates of 0.0002. In the following graphs we show the loss function when the learning rate is decreased by ten-fold.

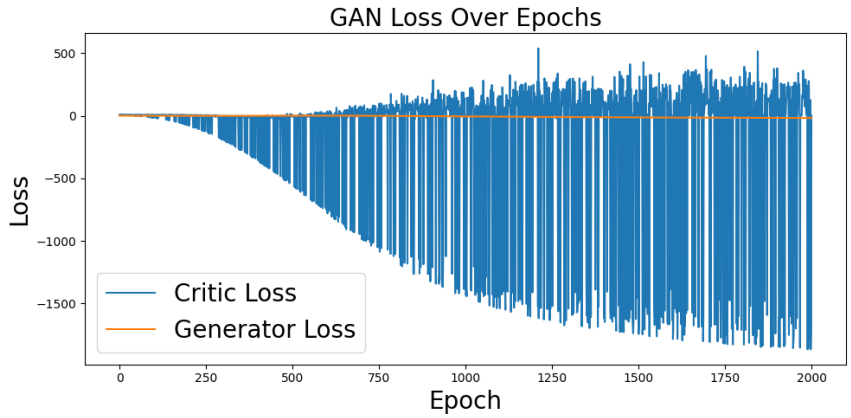


Figure 4.11: Training WGAN for 2000 epochs on abnormal data with learning rate 0.00002.

In 4.11 we show the loss function of the WGAN using a learning rate of 0.00002 and trained for 2000 epochs. Although the critic loss is highly oscillatory, the bottom curve of the graph is more stable than in WGANs using learning rates of 0.0002. This resulted in generators that could not produce meaningful results. Even when trained for longer, this model still did not

produce meaningful results. We show examples of generated WAI data that does not appear meaningful in the examples of synthetic data, 4.5.1.

Finally, we trained new WGANs on the normal WAI data to produce normal synthetic data to augment our dataset.

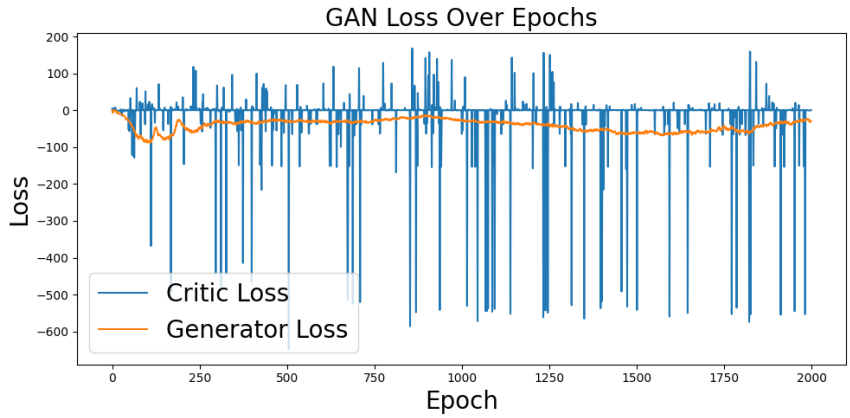


Figure 4.12: WGANs trained for 2000 epochs, learning rate of 0.0002, generating normal data.

When training on normal data, the critic loss was significantly less oscillatory, but still oscillated in an extreme manner.

We also tried different methods similar to section 4.4 such as changing the kernel size and increasing the number of convolutional layers. Kernel size did not result in significant change in the quality of the images. Increasing the number of convolutional layers to at least three in both the generator and the critic produced cleaner results.

4.5.1 Synthetic Data

We generated examples of both normal and abnormal data. We applied a smoothing convolution filter to remove excess noise. Dr. Voss stated from a subjective perspective that the synthetic data appears to be representative of real WAI data. However, there is ultimately no sure way to determine whether a normal or abnormal measurement looks like an ear with or without fluid. In the following figures 4.13 and 4.14 we show a comparison between 25 generated synthetic data points compared to 25 randomly selected real data. We show that the standard deviation and the mean of the synthetic data is comparable visually but further statistical testing is necessary to determine if their difference is significant.

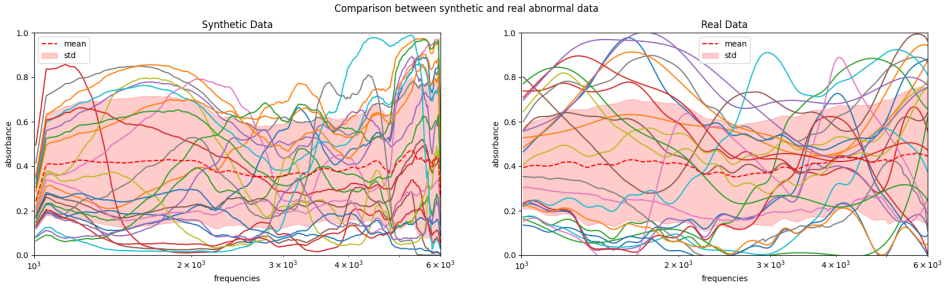


Figure 4.13: 25 randomly selected abnormal synthetic generated by WGAN vs. 25 randomly selected real abnormal data, with mean and standard deviation of full datasets.

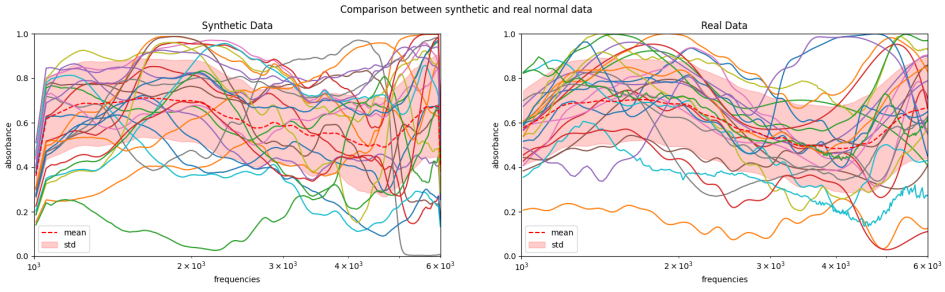


Figure 4.14: 25 randomly selected normal synthetic generated by WGAN vs. 25 randomly selected real normal data, with mean and standard deviation of full datasets

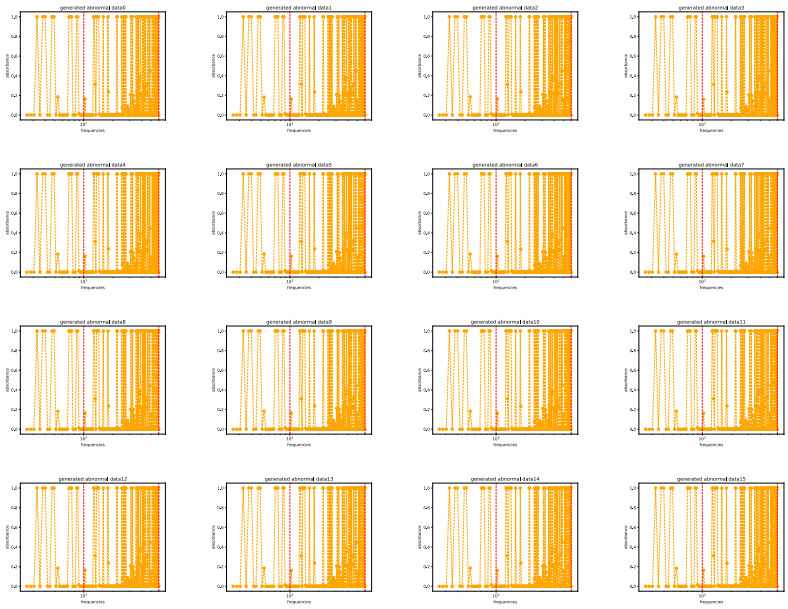


Figure 4.15: Synthetic data from WGANs which does not converge.

We also provide examples of the generated abnormal measurements from a non-converging model as reference. When we refer to a model which does not produce meaningful results, those generated examples in figure 4.15 are one case of this.

The full generated synthetic datasets can be found in the code supplement.

4.6 CNN Results with Synthetic Data

We tried four experiments with synthetic data to understand how data augmentation impacted the CNN accuracy. These models were trained with the same parameters as the best performing network in section 4.4 summarized

in table 4.10.

First we tried balancing the training dataset by matching the number of abnormal data points with real normal data points by adding randomly selected synthetic data. Next, we tried imbalanced datasets. We increased the number of normal and abnormal data so that there was a mix of synthetic and real data for each class. Third, we used all synthetic data, matching the ratio in the previous imbalanced experiment between normal and normal. Lastly, we conducted transfer learning by taking the model trained on all synthetic data and further training on only the real data.

In the following table we show the results of the four experiments on the same reserved real testing data set in figure 4.4.

Training Scheme	Training Accuracy	Test Accuracy	Sensitivity	Specificity	Precision	F1
Transfer	0.9748	0.8595	0.4583	0.9363	0.9004	0.918
Balanced	0.9886	0.8562	0.5417	0.9163	0.9127	0.9145
Imbalanced - M	0.998	0.8495	0.4583	0.9243	0.8992	0.9116
Imbalanced - S	1.0	0.6756	0.6667	0.6773	0.914	0.778

Table 4.11: Performance metrics for CNNs trained on four different training data sets/schemes. In the imbalanced data sets, we use S to denote a fully synthetic set and M to denote a mixed data set.

Training on the fully synthetic imbalanced data did not result in identifying many normal data points correctly. Transfer learning and training on the balanced dataset were very close to training on the imbalanced mixed dataset. We show the confusion matrix in figure 4.16 for the network with the balanced network trained on a mix of synthetic and real data.

Compared to the network trained on the imbalanced real data set shown

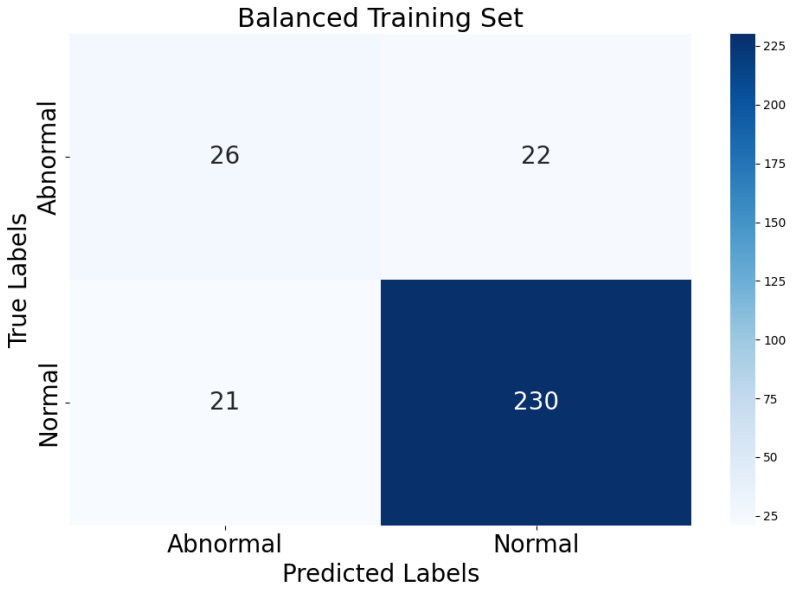


Figure 4.16: Confusion matrix of testing set of real data for CNN trained on balanced data with added synthetic abnormal data.

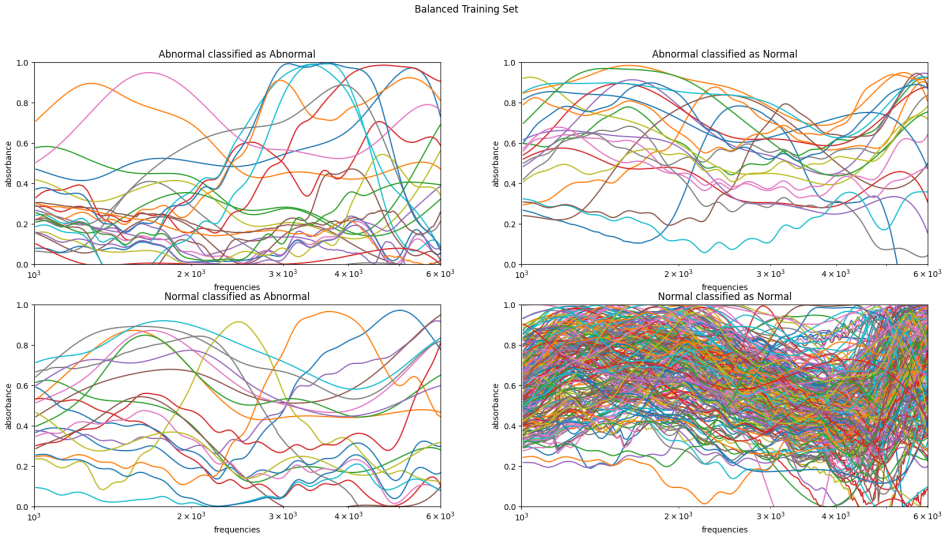
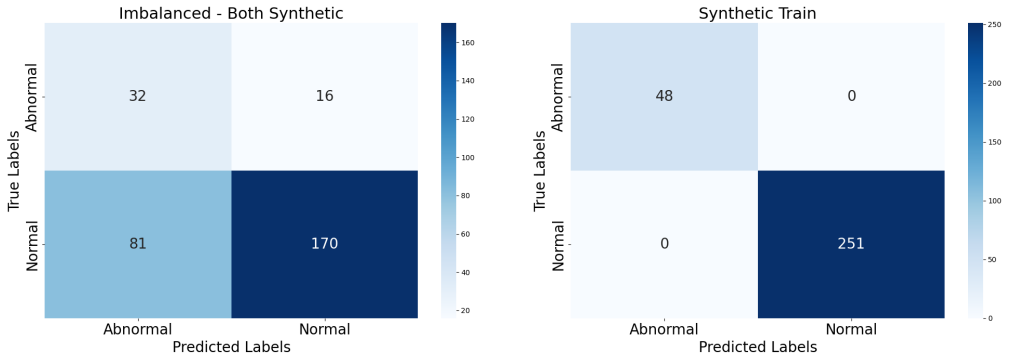


Figure 4.17: Testing set of real data visualized for CNN trained on balanced data with added synthetic abnormal data.

in the confusion matrix by figure 4.4 or the network trained on real balanced data shown in the confusion matrix in figure 4.5, the figure 4.16 shows that including the synthetic data results in unbiasing the network, as the ratio between the number of measurements classified abnormal versus normal reflects the real dataset imbalance. The network misclassifies about the same number of abnormal and normal data, as shown by the confusion matrix. The network trained on some synthetic data is able to classify more abnormal data points accurately, but the trade off is the network also classifies more normal measurements incorrectly. So while the network is able to unbias and accurately classify 10% more abnormal data points than the best performer, it results in a decrease in accuracy by 3% as more normal data points are classified incorrectly. In figure 4.17, we show the visualizations of the measurements in the test set by their classification. The false positives may indicate that fluid was present in those baby ears, but it did not result in hearing loss.

We also show below the confusion matrix on the real imbalanced testing set and a fully synthetic testing data set with the same ratio as the real imbalanced dataset between normal and abnormal data points. This confusion matrix is for the network trained on all synthetic data.

Figure 4.18b shows a perfectly classified testing dataset. This indicates that the CNN is correct and able to distinguish between two classes. Figure 4.18a also indicates that the synthetic data does not fully capture the distri-



(a) Real testing set result

(b) Synthetic testing set result

Figure 4.18: Confusion matrices for network trained with all synthetic data.

bution of abnormal data, since while many of the abnormal data points are classified, 16 are not. Additionally, 81 normal data points are classified as normal, so the synthetic trained model is heavily biased towards the abnormal classification. This means that while the synthetic data may appear to be representative of real data, there may be some characteristics that WGANs is not capturing that are imperceptible to the eye.

4.7 Conclusion

Our goal was to develop a classification model to identify abnormal and normal WAI measurements. We conducted many experiments on hyperparameters such as fixed and scheduled learning rates, epoch length, number of convolutional layers, number of dense layers and batch sizes. We also conducted experiments on the input data structure of the CNN, including training sets that were balanced, imbalanced, as well as an imbalanced data set containing the derivative of the absorbance measurements.

We identified the parameters for the most effective CNN in classifying normal versus abnormal ear data, achieving an overall accuracy of more than 87%. However, this model exhibited a significant class imbalance, favoring the classification of normal ears, as reflected in a sensitivity of only 50% for abnormal samples.

To address this, we generated synthetic data using Wasserstein GANs and evaluated the impact of this augmentation strategy. While the overall accuracy remained relatively stable at 85%, the introduction of synthetic data significantly mitigated the model's bias, as the number of classified abnormals to normals reflected the ratio in the testing set. Additionally, the recall for abnormal ears increased by 5%. These results demonstrate that generative augmentation can improve fairness in classification without significantly compromising accuracy.

4.8 Future Work

Currently, the classification network only classifies based on the absorbance measurement. However, other measurements such as impedance magnitude, impedance angle, and ear canal static pressure could also be used in the classification network, given that enough of these data types are available. By providing more information to the network, this may result in a more accurate classification model. Additionally, it is possible that some normal

data points may have had fluid within the ear despite subjects being able to hear normally. Understanding if all examples are representative of the existence of fluid will allow us to understand the scope of classification of the network. To this end, unsupervised methods in machine learning may be effective in classifying the data. Lastly, implementing a transformer model rather than a CNN may result in improved results as transformers can capture finer information in the dataset [22].

Since the WGANs was unable to fully capture the full distribution of WAI data, further work in synthetic data generation is necessary. We noted that the loss graphs of the critic were highly unstable despite tweaks in the learning rate and batch size, and the model often did not converge. Strategies to improve the stability could be used to improve synthetic generation. For example, Wei et al. [28] use certain condition checks on the loss to ensure that the Lipschitz continuity condition is satisfied. Additionally, it is possible that not enough data exists to create representative results of the data distribution using GANs. Using data augmentation techniques in Nie et al. [19] and Sundgaard et al. [21] to increase the number of examples for training the WGANs may result in better results. Lastly, different generative models such as diffusion models may be able to create more representative synthetic examples. Although requiring more computational power, diffusion models have produced clearer and more representative synthetic data, for example in [10].

Bibliography

[1] S. Aithal, J. Kei, and C. Driscoll. "Wideband absorbance in Australian Aboriginal and Caucasian neonates". In: *Journal of the American Academy of Audiology* 25.5 (May 2014), pp. 482–494. DOI: 10.3766/jaaa.25.5.7.

[2] H.A. AlMakadma and B.A. Prieve. "Refining Measurements of Power Absorbance in Newborns: Probe Fit and Intrasubject Variability". In: *Ear and Hearing* 42.3 (May 2021), pp. 531–546. DOI: 10.1097/AUD.0000000000000954.

[3] L. Ambrosio, E. Brué, and D. Semola. *Lectures on Optimal Transport*. UNITEXT. Springer International Publishing, 2021. ISBN: 9783030721626. URL: https://books.google.com/books?id=vcI5EAAAQBAJ.

[4] Martin Arjovsky, Soumith Chintala, and Léon Bottou. *Wasserstein GAN*. 2017. arXiv: 1701.07875 [stat.ML]. URL: https://arxiv.org/abs/1701.07875.

[5] A.R. Barron. "Universal approximation bounds for superpositions of a sigmoidal function". In: *IEEE Transactions on Information Theory* 39.3 (1993), pp. 930–945. DOI: 10.1109/18.256500.

[6] Gerald Beer and M. Isabel Garrido. "Locally Lipschitz functions, cofinal completeness, and UC spaces". In: *Journal of Mathematical Analysis and Applications* 428.2 (2015), pp. 804–816. ISSN: 0022-247X. DOI: https://doi.org/10.1016/

j.jmaa.2015.02.085. URL: https://www.sciencedirect.com/science/article/pii/S0022247X15002139.

[7] Rohit Kumar Bondugula, Nitin Sai Bommi, and Siba K. Udgata. "An efficient multi-stage ensemble deep learning framework for diagnosing infectious diseases". In: *Decision Analytics Journal* 11 (2024), p. 100458. ISSN: 2772-6622. DOI: https://doi.org/10.1016/j.dajour.2024.100458. URL: https://www.sciencedirect.com/science/article/pii/S2772662224000626.

[8] David Cohen et al. *Math 281 Definitions and Theorems, derived from A Modified Moore Method for Teaching Undergraduate Mathematics.* 1982.

[9] George Cybenko. "Approximation by superpositions of a sigmoidal function." In: *Control Signal Systems 2* (1989). DOI: \url{https://doi.org/10.1007/BF02551274}.

[10] Prafulla Dhariwal and Alex Nichol. "Diffusion Models Beat GANs on Image Synthesis". In: *CoRR* abs/2105.05233 (2021). arXiv: 2105.05233. URL: https://arxiv.org/abs/2105.05233.

[11] J.C. Ellison et al. "Wideband acoustic transfer functions predict middle-ear effusion". In: *Laryngoscope* 122.4 (Apr. 2012), pp. 887–894. DOI: 10.1002/lary.23182.

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

[13] Ian J. Goodfellow et al. *Generative Adversarial Networks*. 2014. arXiv: 1406.2661 [stat.ML]. URL: https://arxiv.org/abs/1406.2661.

[14] Emad M. Grais et al. "Analysing wideband absorbance immittance in normal and ears with otitis media with effusion using machine learning". In: *Scientific Reports* 11.1 (2021), p. 10643. DOI: 10.1038/s41598-021-89588-4. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8137706/.

[15] Katherine A Groon et al. “Air-leak effects on ear-canal acoustic absorbance”. In: *Ear and Hearing* 36.1 (Jan. 2015). Research supported by NIH (grants P30 DC004662, T35 DC008757, R01 DC008318), pp. 155–163. ISSN: 1538-4667. DOI: 10.1097/AUD.0000000000000077.

[16] Ishaan Gulrajani et al. *Improved Training of Wasserstein GANs*. 2017. arXiv: 1704.00028 [cs.LG]. URL: https://arxiv.org/abs/1704.00028.

[17] Michel Loève. *Probability Theory*. Springer, 1977.

[18] Aakash Kumar Nain. *WGAN-GP overriding Model.train$_s$tep*. 2020. URL: https://keras.io/examples/generative/wgan_gp/.

[19] Leixin Nie et al. “Classification of Wideband Tympanometry by Deep Transfer Learning With Data Augmentation for Automatic Diagnosis of Otosclerosis”. In: *IEEE Journal of Biomedical and Health Informatics* 26.2 (2022), pp. 888–897. DOI: 10.1109/JBHI.2021.3093007.

[20] C.A. Sanford et al. “Sound-conduction effects on distortion-product otoacoustic emission screening outcomes in newborn infants: test performance of wideband acoustic transfer functions and 1-kHz tympanometry”. In: *Ear and Hearing* 30.6 (Dec. 2009). Erratum in: Ear Hear. 2014 Mar-Apr;35(2):288, pp. 635–652. DOI: 10.1097/AUD.0b013e3181b61cdc.

[21] Josefine Vilsbøll Sundgaard et al. “A Deep Learning Approach for Detecting Otitis Media From Wideband Tympanometry Measurements”. In: *IEEE Journal of Biomedical and Health Informatics* 26.7 (2022), pp. 2974–2982. DOI: 10.1109/JBHI.2022.3159263. URL: https://ieeexplore.ieee.org/document/9735386.

[22] Ashish Vaswani et al. “Attention Is All You Need”. In: *CoRR* abs/1706.03762 (2017). arXiv: 1706.03762. URL: http://arxiv.org/abs/1706.03762.

[23] Cédric Villani. “The founding fathers of optimal transport”. In: *Optimal Transport: Old and New*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 29–37. ISBN: 978-3-540-71050-9. DOI: 10.1007/978-3-540-71050-9_3. URL: https://doi.org/10.1007/978-3-540-71050-9_3.

[24] S.E. Voss et al. “Reflectance Measures from Infant Ears With Normal Hearing and Transient Conductive Hearing Loss”. In: *Ear and Hearing* 37.5 (Sept. 2016), pp. 560–571. DOI: 10.1097/AUD.0000000000000293.

[25] Susan Voss. *Wideband Acoustic Immittance Database*. 2014. URL: doi.org/10.35482/egr.001.2022.

[26] Kai Wang. *A STUDY OF CUBIC SPLINE INTERPOLATION*. 2013. URL: https://www2.rivier.edu/journal/ROAJ-Fall-2013/J784-Wang_cubic-splines.pdf.

[27] Qi Wang et al. “A Comprehensive Survey of Loss Functions in Machine Learning.” In: *Annals of data science* (2020). DOI: \url{https://doi.org/10.1007/s40745-020-00253-5}.

[28] Xiang Wei et al. “Improving the Improved Training of Wasserstein GANs: A Consistency Term and Its Dual Effect”. In: *CoRR* abs/1803.01541 (2018). arXiv: 1803.01541. URL: http://arxiv.org/abs/1803.01541.

[29] Richard L. Wheeden and Antoni Zygmund. *Measure and Integral: An Introduction to Real Analysis*. Boca Raton, FL: CRC Press, 1977. ISBN: 9780367258247.