Distribution Agreement

In presenting this thesis as a partial fulfillment of the requirements for a degree from Emory University, I hereby grant to Emory University and its agents the non-exclusive license to archive, make accessible, and display my thesis in whole or in part in all forms of media, now or hereafter now, including display on the World Wide Web. I understand that I may select some access restrictions as part of the online submission of this thesis. I retain all ownership rights to the copyright of the thesis. I also retain the right to use in future works (such as articles or books) all or part of this thesis.

Jack Wessell April 10, 2023

The Evaluation of Distance Metrics for Generating Adversarial Perturbations from Univariate Time Series Data

By

Jack Wessell

Dr. Li Xiong

Computer Science

Dr. Li Xiong

Adviser

Dr. Manuela Manetta

Committee Member

Dr. Ymir Vigfusson

Committee Member

2023

The Evaluation of Distance Metrics for Generating Adversarial Perturbations from Univariate Time Series Data

By

Jack Wessell

Dr. Li Xiong

Adviser

An abstract of
a thesis submitted to the Faculty of Emory College of Arts and Sciences
of Emory University in partial fulfillment
of the requirements of the degree of
Bachelor of Science with Honors

Computer Science

2023

Abstract

The Evaluation of Distance Metrics for Generating Adversarial Perturbations from Univariate Time Series Data

By Jack Wessell

Recent research has shown that data can be manipulated so that when a machine learning model analyzes the data it will almost certainly classify the input incorrectly. To produce these inputs, an attacker adds a small amount of noise to the original example that forces the model to incorrectly classify the input. The goal when creating these examples is to cause the input to be incorrectly classified by the machine learning algorithm while being nearly imperceptibly different from their benign counterparts. In the context of image processing, the amount of noise added is typically calculated using a Euclidean or infinity norm regardless of the task performed by the target model such as image classification or image segmentation. However, despite the success of such distance metrics in the image processing domain, there has been little investigation into the efficacy of these measures in the domain of time series data, where these metrics are often applied by default. In this paper, I compare the effectiveness of generating adversarial examples with a variety of distance functions targeting deep learning models to determine which are the strongest approximations for human perception. I also utilize a reader study to provide statistical evidence for the superiority of one metric over another.

The Evaluation of Distance Metrics for Generating Adversarial Perturbations from Univariate Time Series Data

By

Jack Wessell

Dr. Li Xiong

Adviser

A thesis submitted to the Faculty of Emory College of Arts and Sciences
of Emory University in partial fulfillment
of the requirements of the degree of
Bachelor of Science with Honors

Computer Science

2023

Acknowledgements

Firstly, I would like to express my gratitude to my advisor Dr. Li Xiong. Dr. Xiong's machine learning class provided both my foundational knowledge of and passion for the field during my junior year. She also graciously accepted me into the AIMS research group where I have had the opportunity to hone my skills as a student and researcher while gaining exposure to fascinating problems alongside many talented, intelligent, and kind people. In addition to Dr. Xiong I would like to thank Wenjie Wang and Jian Lou, two of my colleagues in AIMS, who have supported me extensively throughout my time in the lab.

I would also like to thank Dr. Manetta and Dr. Vigfusson for the impact they have had on my undergraduate education. Both professors are not only incredibly knowledgeable in their respective fields and great lecturers but are also passionate about their students and what they take away from their courses. It is obvious from their courses that they truly care about each of their students, and I am incredibly grateful to have had the opportunity to meet them.

Lastly, I would like to give a special thanks to my parents, my girlfriend, and my friends for their continued support both throughout my thesis project and my undergraduate career. I would not be where I am today without their selflessness.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Adversarial Machine Learning

Machine learning models are playing an ever-growing role in the software that we frequently use. Such systems are at the heart of software such as voice recognition systems, automated fraud detection, and facial recognition. Additionally, with OpenAI's recent tool ChatGPT boasting the fastest growing user base in history, society's reliance on and interactions with machine learning models will continue to increase.

However, like much of our society's existing software and hardware, machine learning models are often vulnerable to attacks. Szegedy et. al [31] first noted machine learning models' vulnerability to artificially produced examples in the domain of image processing. They demonstrated that by adding generated noise to an image that was previously classified correctly by a model they could force the target model to classify the resulting input incorrectly. This noise can be so small that it is visually undetectable, as demonstrated in Figure 1.1. Since their paper was published in 2013, the domain of adversarial machine learning has matured considerably with researchers proposing a variety of attack methods and defenses for models trained on all forms of data. Recent work has produced attacks that can cause machine learning models to misclassify examples nearly 100% of the time [4], attacks against models performing different tasks such as image segmentation or malware detection [35] [13], and a variety of defense techniques [28].

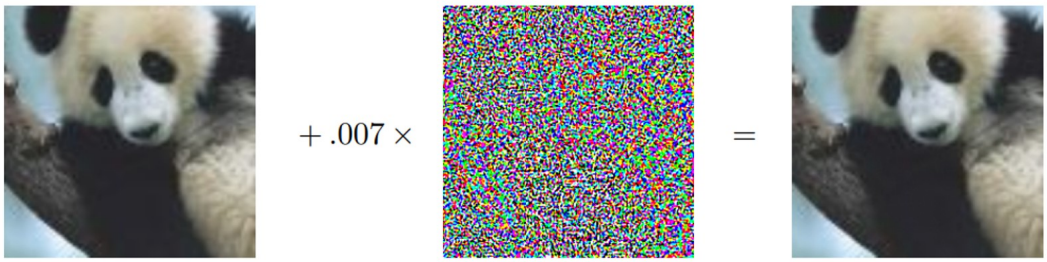

Figure 1.1: An example of an adversarial machine learning attack from [12]. Here, the added perturbation is so small it is undetectable by a human.

Regardless of the attack method used, the goal when attacking a machine learning system is always the same: take data that is correctly classified by the target model and add noise so that the resulting example, while imperceptibly different from the original, is classified incorrectly. For example, a malicious individual may deface a stop sign so that an autonomous vehicle's image processing system believes it to be a yield sign. Such an attack is not just theoretical. In fact, Eykholt et. al [8] showed that real-world stop signs can be defaced in such a way that is not suspicious to humans but are nonetheless classified incorrectly by machine learning models.

As our reliance on machine learning algorithms to make intelligent decisions for us grows, so does the potential damage that malicious attackers can cause by targeting such systems. Thus, it is of the utmost importance that we understand the vulnerabilities of the machine learning systems. By creating and testing adversarial machine learning techniques, we can develop algorithms that can defend against such attacks. In this way, we can better ensure the safety and security of the users of machine learning systems.

Gaps in Existing Works

For an attack against a machine learning model to be effective, it must be difficult for a human to detect. This is because examples that have been blatantly manipulated can easily be detected by humans. Much of the research regarding the similarity between

natural data and their artificially produced counterparts has been in the image processing domain where, as similarity defined by either the L_∞ or L_2 distance metric, has become the standard [4]. Additionally, the use of the L_∞ distance to measure similarity has been extended by default to adversarial machine learning research on models trained on other forms of data such as audio [5]. Thus, there has been little focus on determining which metrics are most suitable for measuring the size of the perturbation added to the data to produce adversarial counterparts in less-studied domains such as time series data.

Data in many domains can be represented as univariate time series data. Audio data or electrocardiogram (ECG) data are just two examples. Like image data, univariate time series data can be visualized (eg. ECG data) or listened to (eg. audio data) by an individual, which makes it useful for determining how similar the adversarial and benign examples are to each other from a human's perspective. However, univariate time series data has a form that is very different from that of image data, and thus definitions of distance that may be useful in the image processing domain may be ineffective in the domain of univariate time series data.

Finally, despite the importance of the imperceptibility of the perturbations to the efficacy of the attack, very few papers have employed studies to determine which distance metrics and adversarial techniques produce the least detectable perturbations. Therefore, there is little statistical evidence to support the superiority of one definition of distance over another, especially in the domain of univariate time series data.

Contributions

In this thesis, I seek to determine the efficacy of a variety of distance metrics for attacking machine learning models trained on time series data. In particular, I introduce the Wasserstein distance [32] as a superior metric by which to create more natural and less easily detectable perturbations for time series data. I utilize Electrocardiogram (ECG) time series data due to its following desirable properties:

1. ECG data is univariate in nature, allowing for the exact computation of the Wasserstein distance and its derivative while acting as a baseline for future research in multivariate time series data.
2. ECG data can be visually interpreted from its plot by a human, which allows us to draw conclusions about which distance metrics produce less detectable adversarial examples.
3. There is an inherent importance of and emphasis on security in ECG data due to it belonging to the medical field.

I also experiment with producing adversarial perturbations with commonly used metrics in addition to the Wasserstein-based distance functions. Using the resulting perturbations and a reader study, I attempt to determine which definition of distance produces the least detectable perturbations using a reader study consisting of medical professionals. The contributions to the field of adversarial machine learning in this thesis are summarized as the following:

1. I propose to use the Wasserstein distance to better capture the similarity between perturbed and clean examples in the context of univariate time series data.
2. I extend an optimal state-of-the-art attack to incorporate the Wasserstein distance for the first time.
3. I experimentally compare the approach to standard distance metrics such as the L_2 and L_∞ functions using real-world ECG datasets.
4. I employ a reader study to determine which methods produce the least detectable perturbations.
5. I perform additional studies on audio data to test the generalizability of the approach.

Chapter 2

Background

In this section, I start by introducing the mathematical underpinnings of the machine learning models utilized throughout this research as well as the general notation used to describe the techniques utilized throughout. Next, I briefly describe the distance metrics used, as they are the focus of the research. After that, I provide a short overview of some of the important results from the field of adversarial machine learning with which the reader should be familiar.

Neural Networks

Neural networks are a subset of machine learning models that have proven to be incredibly powerful tools for solving problems such as image classification [33], image segmentation [23], text classification [24], and text generation [10] just to name a few. All neural networks, regardless of their structure or the task they are designed to perform, consist of layers of a linear transformations followed by a nonlinear activation function such as the following ubiquitous Sigmoid function:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.1}$$

As neural networks become increasingly complex, they also tend to gain more layers, which is referred to as the model becoming deeper. Thus, many neural networks are often termed deep learning models. For the remainder of the paper, I will use the terms deep learning and machine learning interchangeably.

The task that a model is designed to perform is determined by the architecture of its layers. The following layer architectures are utilized extensively in our models:

1. Fully-Connected Layer:

   A fully-connected layer simply consists a collection of matrices that linearly transform the entire input vector into a vector of the target shape. This transformation is then followed by a non-linear transformation such as (2.1). Thus, in the context of time series data, a fully-connected layer processes a time series by weighting each individual time step and then combining all weighted time steps together to create a resulting feature vector.

2. Convolutional Layer:

   A convolutional layer consists of a much smaller matrix that slides over the input data and combines the value of only neighboring features into a single element in the feature vector. Thus, convolutional layers can learn location-invariant features such as edges in an image. Therefore, networks that include convolutional layers are well suited to handling data with local spacial or temporal dependencies. [20]

While there are a variety of other layers architectures commonly used for a variety of machine learning tasks such as recurrent units, long-short term memory cells, and transformers, the fully-connected and convolutional layers are the only ones utilized in the development of the models in this project.

Notation and Definitions

I will now briefly overview the notation utilized extensively throughout the paper:

1. Definition 1: A time series $X = [x_1, x_2, ..., x_n]$ is an ordered set of values with a length of n.
2. Definition 2: A dataset $D = \{(X_1, Y_1), (X_2, Y_2), ..., (X_m, Y_m)\}$ is a collection of ordered pairs of values where X_i is a time series and Y_i is its corresponding label.
3. Definition 3: A classifier C is a machine learning model trained to take as input a time series X_i and predict its corresponding label $C(X_i) = \hat{Y}_i$.
4. Definition 4: The logits of a trained classifier C when evaluated on X_i are denoted by $Z(X_i)$ and represent the output of the model before being transformed into a probability distribution of potential class labels.
5. Definition 5: An adversarial time series produced from X_i will be denoted X_i'.

Distance Metrics

Now I will formally define the distance metrics utilized throughout the paper, starting with the L_p norms. The L_p distance between two vectors is denoted by $\|x - x'\|_p$ and is defined as the following:

$$\|x - x'\|_p = (\sum_{i=1}^{n} |x_i - x_i'|^p)^{1/p} \tag{2.2}$$

This project makes use of two L_p norms: the L_2 norm and the L_∞ norm. We will now explain these distance metrics in more detail:

1. The L_2 norm:

   The L_2 norm is calculated by setting $p = 2$ in (2.2) and is thus given by:

   $$\|x - x'\|_2 = (\sum_{i=1}^{n} (x_i - x_i')^2)^{1/2} \tag{2.3}$$

   The L_2 distance is the generalization of Euclidean distance in two dimensions to higher dimesions. In L_2 metric space, the distance between a natural and adversarial example is small if the difference between the individual elements x_i and x_i' is small.

2. The L_∞ norm:

   The L_∞ norm of a vector X can be calculated by the following formula:

$$L_\infty = max(x_1, x_2, ..., x_n) \tag{2.4}$$

   Thus, in the context of adversarial examples, the L_∞ distance between the natural and adversarial time series is simply the largest difference between any corresponding time step. While this may seem overly simplistic, the L_∞ norm has proven incredibly successful in the image processing domain and has even been argued to be the optimal choice for measuring similarity in that domain [4].

Both the L_2 and L_∞ norms are utilized throughout this project as a baseline for generating adversarial examples.

The Wasserstein distance is a metric that defines the similarity between two probability distributions P and Q as opposed to two vectors. In many ways, the Wasserstein metric captures ideas about similarity that more conventional distance metrics like the L_p norms do not. For example, in Figure 2.1, the Wasserstein distance between the two curves is very small, while the values of the L_p norms are much larger.

The value of the Wasserstein distance is the numerical cost of the optimal transport problem. In particular, its formula is given by the following, where J(P,Q) is the set of all joint distributions of P and Q and $X \sim P$ and $Y \sim Q$:

$$W_p(P, Q) = (\inf_{J \in J(P,Q)} \int \|x - y\|^p \, dJ(x, y))^{\frac{1}{p}} \tag{2.5}$$

The above formula does not reveal the intuition behind the Wasserstein metric, so the following analogy is often paired with it: imagine you have a mound of dirt with a shape of A and a hole in the ground of shape B. The value of the Wasserstein distance is the

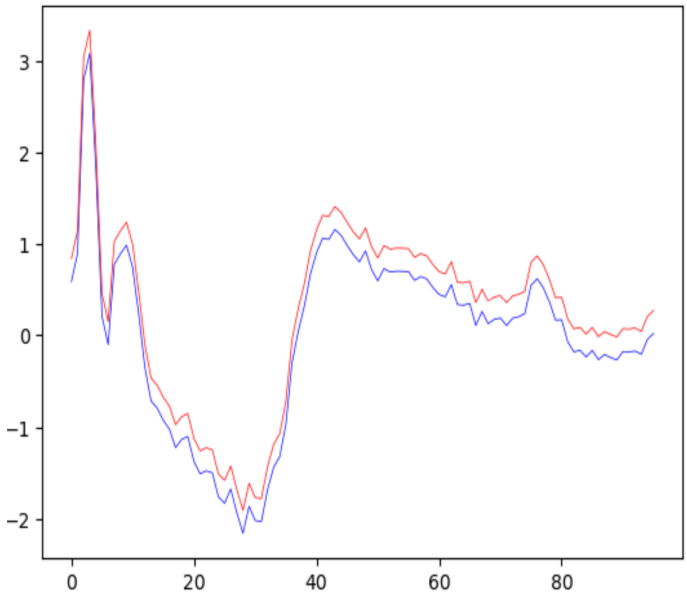

Figure 2.1: For these two visually similar curves the Wasserstein distance is small (0.25). However, the L_1 distance is 100 times larger at 24.0. The L_2 distance is better, but still 10 times larger at 2.45

amount of work required to ensure that all dirt from A fills in the holes of B. Thus, the Wasserstein distance is often referred to as the Earth Mover's distance.[1]

The formula in (2.3) cannot be calculated in general, let alone have the computable derivative that is required for producing adversarial examples. Fortunately, in the one dimensional discrete case, in which ECG data is included, (2.3) reduces the following computable, differentiable, and much more intuitive form:

$$W(u,v) = \sum_{i=1}^{n} |\sum_{j=1}^{i} u_j - \sum_{j=1}^{i} v_j| \tag{2.6}$$

1. Technically, the Wasserstein distance is undefined for two vectors in general due to them not both having the same sum (i.e, both the pile of dirt and the hole have the same volume). However, the Wasserstein distance has proven empirically to be an incredibly effective measure of similarity in a variety of fields in machine learning [1][34]

Thus, by utilizing the differentiability of the one-dimensional discrete case for the Wasserstein distance, I can utilize optimization techniques to produce adversarial examples using this metric.

Throughout the experiments I also utilize a handful of other metrics while collecting data, namely the L_1 distance, Frechet distance, and Dynamic-Time warping. However, these are simply used as benchmarks and not are used to produce adversarial examples. Thus, further discussion of their formulation is unnecessary. I refer the reader to [9] and [29] for their definitions.

Threat Models

Researchers in academia and industry alike are working on applying machine learning to many security-critical domains such as self-driving cars [14], automated speech recognition and detection of voice commands [36], and malware detection [6]. Thus, our ability to apply deep learning techniques safely and successfully to such tasks is dependent on our knowledge of the threats to which these models are vulnerable.

Almost all adversarial machine learning research operates under one of the following two threat models:

1. Black-box attack:

   Under the black-box model, it is assumed that the attacker has no access to any information about the model other than its output. The architecture, model weights, gradients, and training data are entirely unknown to the attackers.

2. White-box attack:

   The white-box threat model is the opposite of the black-box model. Under this set of assumptions, it is assumed that the attacker has access to all information about the model such as its parameters and the value of its gradients.

In this paper, all attacks and experiments will be performed under a white-box model. This is for two main reasons. First, building defenses under the assumption that a skilled hacker will be unable to access certain sensitive data can leave the software or hardware vulnerable when that assumption is broken. Second, black box attacks often reduce to white box attacks in practice. Recent work has shown that a programmer can often train a model locally (potentially with a different architecture) that is meant to replicate the target model. The programmer can then fine-tune the local model using the outputs of the target model as training labels, and by generating adversarial examples against the local model the attack can often fool the target model [26].

These threat models can be used to produce adversarial examples, which are data produced by combining data from a natural source with a perturbation so that the original and resulting data are perceptibly similar while being classified incorrectly by a target model. While the definition of similarity varies from domain to domain, models trained on a variety of data are vulnerable to adversarial examples.

In the realm of speech recognition, early research showed that it is possible to generate sounds that, while unintelligible to humans, can trigger voice commands on their devices [3]. This research focused on simple, short commands such as "Ok Google," or "Call 911" and mainly targeted more traditional learning algorithms such as Hidden Markov Models, but the security threats that such attacks pose should not be understated. More recent research has shown that by utilizing modern adversarial machine learning techniques attackers can add small perturbations to audio and cause accurate deep learning models such as Mozilla's DeepSpeech speech-to-text model to transcribe the audio into any arbitrary phrase [5]. Thus, studying the capabilities of adversarial machine learning and potential defenses is crucial to the security of this ever-growing field.

In the field of automated malware detection, recent research has shown results similar to other fields: only slight modifications to the malicious file can cause a highly accurate machine learning model to wrongly classify it as benign [13]. Such methods are

not only dangerous to automated malware detection software but make it entirely useless if defenses are not developed to protect against such attacks.

Given the plethora of threats our current deep learning models face, there has been an arms race to develop increasingly powerful attacks and defenses for such deep learning models. While there are a variety of novel and fascinating defense techniques such as defensive distillation [27], MagNet [22], and Defense-GAN [28], most defense techniques such as defensive distillation and MagNet are incapable of defending against the most powerful attack algorithms in a white-box setting [4][28].Unfortunately, training a GAN that is powerful enough to successfully defend a model from attacks is incredibly time consuming, resource intensive, and difficult to measure [28]. Thus, our paper is focused primarily on attack methods against undefended models. In the following section, I will briefly introduce a handful of the most common attack algorithms and then introduce the methods and experiments performed in this research project.

Chapter 3

Adversarial Machine Learning Attack Algorithms

L-BFGS

This attack algorithm is the original formulation presented by Szegedy et. al in their groundbreaking adversarial machine learning paper [31]. The optimization problem presented in the paper, which is focused on generating adversarial examples in the image processing domain, had the following formulation:

$$\begin{aligned} &\text{Minimize } \|\delta\|_2 \\ &\text{Such that } C(X+\delta) = t \\ &\text{and } X+\delta \in [0,1]^m \end{aligned} \tag{3.1}$$

Where X is an image unrolled into a vector of dimension m, δ represents the perturbation added to X to achieve the desired result, and t is the target class of X'. In practice, this optimization problem can be incredibly difficult to solve. So, Szegedy et. al use an optimization technique known as box-constrained L-BFGS to find an approximate solution

to the problem above by solving:

$$\begin{aligned} &\text{Minimize } c * \|\delta\|_2 + loss_C(X+\delta, t) \\ &\text{Such that } X + \delta \in [0,1]^m t \end{aligned} \tag{3.2}$$

In this formulation, $loss_C$ represents a loss function that measures whether the classifier C has produced the desired output. The authors then repeatedly solve this problem using many values of c, choosing the minimum value of c for which $C(X+\delta) = t$.

Fast Gradient Sign

The fast gradient sign method is an attack algorithm developed shortly after Szegedy et. al introduced their L-BFGS algorithm described above. Unlike the L-BFGS algorithm, the fast gradient sign method attempts to bound the perturbation introduced by δ with the L_∞ metric and is designed to be incredibly fast while forgoing optimal performance [12] . Thus, perturbations produced by the fast gradient sign method are not attempting to approximate the optimal perturbation. The problem solved by the fast gradient sign method has the following formulation:

$$X' = X - \epsilon * \text{sign}(\nabla loss_C(X, t)) \tag{3.3}$$

In this formulation, ϵ represents a small constant controlling by how much the values of X are updated. So, to produce the adversarial examples, the gradient of the loss function with respect to the input is calculated and the values of X are simply updated by epsilon in the direction of the gradient.

It is worth noting that Kurakin et. al introduced an iterative version of the fast gradient sign method [18]. The formulation is very similar to the original fast gradient sign method so I will not go over it here, but it was shown that the iterative fast gradient sign produces much better adversarial examples than its non-iterative counterpart.

The Carlini-Wagner Attack

The Carlini-Wagner (CW) attack is an optimization-based method of creating adversarial examples [4]. Furthermore, Carlini-Wagner is also a targeted attack like L-BFGS, meaning that the adversary is able to choose which class the model predicts for the adversarial example. The objective of the attack is the following constrained optimization problem:

$$\begin{aligned} &\text{Minimize } D(X, X+\delta) \\ &\text{Such that } C(X+\delta) = t \end{aligned} \tag{3.4}$$

Here, D is a distance function such as an L_p norm. This formulation is the same as that used for the L-BFGS algorithm except for the fact that here a general distance function is used in place of L_2 distance. However, where the CW attack distinguishes itself from L-BFGS is in how it reformulates the problem to make finding an approximate solution much more tractable. Instead of using box constrained L-BFGS to find a solution, Carlini and Wagner aimed to utilize an optimizer such as Adam, a gradient-based optimization algorithm, to produce adversarial examples. However, due to the nonlinear nature of neural networks, the constraint $C(x+\delta)$ is incredibly difficult to optimize over. Thus, Carlini and Wagner replace this constraint with a function f where $f(X+\delta) \leq 0$ if and only if $C(X+\delta) = t$. This reduces the above formulation to the following optimization problem:

$$\text{Minimize } loss(X) = D(X, X+\delta) + c * f(X+\delta) \tag{3.5}$$

In this formulation, $f(x)$ replaces the misclassification constraint as an objective function. Thus, by varying the parameter c the adversary can vary the relative weight of the similarity between the benign and adversarial examples and the misclassification of the model. Once this loss is calculated, the gradients with respect to the perturbation on X can be calculated and an optimizer such as Adam can be applied to minimize the loss function. This process allows for the creation of small perturbations that can fool the target model.

In the CW attack, the programmer has a variety of choices regarding both the distance metric and the objective function. I experiment with a wide variety of distance functions in this paper; however I use the following objective function exclusively:

$$f(x) = \max(\max_{i \neq t}(Z(x)_i) - Z(x)_t, -\kappa) \tag{3.6}$$

In this objective function, κ is a hyperparameter that the adversary is in control of alongside c. By increasing kappa we can effectively increase the required confidence level of our model when classifying the example as the target class. However, I found success sticking with very low values for kappa and seldom found it necessary to increase kappa beyond 2. Carlini and Wagner experiment with many objective functions in their original paper, however they find the most success with this formulation [4]. Thus, we will utilize this objective function throughout our experiments as well.

Due to its optimization-based approach as well as the introduction of the objective function, the CW attack is stronger than other attacks such as those introduced previously in this manuscript [28]. It is common for the CW attack to successfully cause the target model to misclassify the adversarial example 100% of the time, and the CW attack was also designed to break defensive distillation, which was the strongest defense against adversarial attacks at the time [4]. For these reasons, we will be utilizing the Carlini-Wagner attack exclusively in our experiments. It is worth noting, however, that the CW attack is slower than other attack formulations due to it being optimization based.

Chapter 4

Methods and Experiments

In this section, I will introduce our contributions and the structure of our algorithms, datasets, models, and experiments. Finally, I will introduce the structure of our reader study.

Carlini-Wagner attack formulation

As mentioned in previous sections, this work is primarily focused on determining which distance functions are the best metrics to optimize for when producing adversarial perturbations. The L_∞ metric acts as a baseline formulation against which the other metrics will be compared. I will be utilizing a Carlini-Wagner style attack to create adversarial examples with (3.2) as our objective function. I will be comparing the resulting adversarial examples produced using four different distance metrics.

L_∞ based perturbations

The first distance metric used to produce perturbations is the L_∞ distance as introduced in (2.4). However, the standard formulation for the L_∞ metric does not perform well under optimization. Since it only penalizes the term with the greatest absolute value, optimization algorithms often oscillate between two sub-optimal solutions. Thus, I utilize a common technique similar to that employed by Carlini and Wagner where instead of only penalizing the term that is largest in absolute value, all terms greater than a constant τ are

penalized [4]. Additionally, if all terms are smaller than τ, τ is multiplied by .9 and the algorithms continues. Thus, the minimization problem becomes the following when using the L_∞ norm as the distance metric:

$$\text{Minimize } \sum_i [(\delta_i - \tau)^+] + c * f(X + \delta) \tag{4.1}$$

This formulation makes using an efficient gradient-based optimizer such as Adam more effective when optimizing with respect to the L_∞ norm.

L_2 and Wasserstein based perturbations

Because the L_2 metric is fully differentiable, no additional steps are necessary to ensure that the optimizer can find a viable solution. Thus, loss function becomes:

$$\text{Minimize } L_2(X, X + \delta) + c * f(X + \delta) \tag{4.2}$$

Since the Wasserstein distance is also fully differentiable in the one-dimensional discrete case, it behaves similarly to the L_2 distance. Thus, the following formulation for the minimization problem works successfully:

$$\text{Minimize } W(X, X + \delta) + c * f(X + \delta) \tag{4.3}$$

Mixed loss function

The final distance function consists of a weighted combination of the L_2 and Wasserstein metrics. Since L_2 and Wasserstein both capture important but different aspects of similarity between curves, I was curious to see how the optimization problem behaved when both distance functions were present in the loss function. Thus, the minimization

problem becomes:

$$\begin{aligned} &\text{Minimize } D(X, X+\delta) + c * f(X+\delta) \\ &\text{Where } D(X, X+\delta) = (1-\alpha) * L_2(X, X+\delta) \\ &\qquad\qquad +\alpha * W(X, X+\delta) \end{aligned} \tag{4.4}$$

Here, alpha is a real number that is between 0 and 1 inclusive.

Now that I have defined each of the loss functions utilized throughout the paper, the algorithm used to perform the Carlini-Wagner attack can be constructed. The pseudocode is show in algorithm 1.

Algorithm 1 Carlini-Wagner Attack Algorithm

Input: original data X, distance metric D, optimizer O, target class t, target model C, iterations *l*, weight c, confidence κ

Output: Adversarial Example X'

$perturbation \leftarrow \mathcal{N}(0, 1)$
$i \leftarrow 0$
while $i \neq l$ **do**
    $Z = Logits(C(X + perturbation))$
    $f = \max(\max_{i \neq t}(Z_i) - Z_t, -\kappa)$
    $Distance = D(X, X + perturbation)$
    $loss = Distance + c * f$
    $O.minimize(loss, perturbation)$
    $i+ = 1$
end while
$X' \leftarrow X + perturbation$

Datasets and Models

All experiments are based on three datasets from the well-studied and publicly available University of California-Riverside time series classification (UCR) archive [7]. The three datasets are all selected from the ECG data category. I will now briefly introduce the three datasets: 1). ECG200 is a binary classification task in which the model is tasked with distinguishing between a normal heartbeat and Mydocardial Infarction. It contains 35 half-hour records sampled at a rate of 125 Hz. 2) ECG5000 is a mulit-class classification task with data collected from the BIDMC Congestive Heart Failure Database. The data

are collected from 20-hour long ECG readings sampled at 250 Hz from patients with severe congestive heart failure, and the class labels represent the severity of the heart failure for a total of five total classes. To create the dataset, the ECG reading was split into individual heartbeats and interpolated so that each heartbeat has constant length. 3). ECGFiveDays is also a binary classification task like ECG200. The data are all collected from the same 67-year old-male, and the classification task is to determine whether the ECG reading was collected on the first or last day of a five day period. The description of each dataset can be found in Table 4.1, where TrainSize is the number of time series examples in the training dataset, while TestSize is the number of time series examples in the test dataset.

Dataset Description				
Dataset	TrainSize	TestSize	Classes	SeqLen
ECG200	100	100	2	96
ECG5000	500	4500	5	160
ECGFiveDays	23	861	2	136

Table 4.1: Description of each dataset.

For our models, I adopt the well-studied model architectures from [15] as they are well reviewed and boast a wide variety of structures. The four model architectures we study are the Multi-Layer Perceptron (mlp) [11], Convolutional Neural Network (cnn) [25], Fully-convolutional network (fcn) [21], and a Residual Network (ResNet) [16]. The performance of each of these models on our target datasets can be found in table 4.2, while the model architectures can be found in table 4.3 [1].

Model Accuracy on Test datasets				
Dataset	MLP	FCN	CNN	ResNet
ECG200	0.916	0.9	0.83	0.89
ECG5000	0.931	0.939	0.928	0.934
ECGFiveDays	0.979	0.987	0.885	0.993

Table 4.2: Performance of each model on each dataset.

1. The ResNet architecture is included in the appendix

Model Architectures		
CNN	MLP	FCN
Conv1D(6, 7, 1)	Flatten	Conv1D(128, 8, 1)
Sigmoid	Dropout(0.1)	BatchNorm()
Avg. Pooling(3x3)	FC(500) + ReLU	ReLU
Conv1D(12, 7, 1)	Dropout(.2)	Conv1D(256, 5, 1)
Sigmoid	FC(500) + ReLU	BatchNorm()
Avg. Pooling(3x3)	Dropout(.2)	ReLU
Flatten	FC(500) + ReLU	Conv1D(128, 3, 1)
FC(10) + sigmoid	Dropout(.3)	BatchNorm()
	FC(10) + Softmax	ReLU
		GlobalAvgPooling()
		FC(10) + Softmax

Table 4.3: Model architectures for CNN, MLP, and FCN.

Experimental Setup

After the training and evaluation of the models, I began performing our experiments. For each dataset and model, I ran an experiment on each of the four distance function defined above (i.e L_2, L_∞, Wasserstein, and mixed). Thus, I ran a total of 48 experiments (3 datasets, 4 models, 4 loss functions). For all experiments I utilize (3.2) as our objective function.

All experiments were based on the Algorithm shown in Algorithm 1. For each experiment an Adam optimizer was utilized due to its well-documented empirical performance with a learning rate of 0.01. In early experiments I varied the value of the learning rate used, but found it had little effect on the resulting adversarial examples provided a sensible value was used. Finally, each perturbation underwent 50 iterations of optimization.

For the experiments with the L_2, L_∞, and Wasserstein distance functions I performed the above procedure while varying the values of c and κ. For early experiments I varied κ from 0 up to 20, however I found that for $\kappa > 2$ little performance gain was realized. Thus, in most experiments κ was set to either 0, 1, or 2. However, the value of c is

crucial in successfully creating adversarial examples that could feasibly fool a human. With a value of c too low, the optimizer will take very small steps in the direction of the target class and will be unable to force the model to classify the input incorrectly. With a value of c that is too high, the optimizer will take steps in an overly greedy manner and likely overstep a value close to the true minimum. Furthermore, the proper value of c varies with the dataset, model, and distance function used. Thus, I varied c from values as low as $5e^{-4}$ up to 20 depending on the experiment. For each set of parameters in an experiment, 100 random examples are taken from the target dataset's test set and used as the starting point for our adversarial experiments (in the case of ECG200, this is simply the entire test set).

The experiments utilizing the mixed loss function are very similar to those described above but contain one key difference. For our mixed function, κ is set to 0 for all experiments. Then, I ran an experiment using the procedure and parameters for c described above with the values for α being $[0, 0.1, 0.25, 0.5, 0.75, 0.9, 1]$. Other than this difference, the experiments were identical to the standard loss functions.

For all the experiments described above, the examples that successfully fooled the target models were saved along with the dataset they came from, the target model, the distance metric used, and the parameter settings. These adversarial examples were then used to create visualizations for use in the reader study.

Evaluation Metrics

For each adversarial example produced I measured the following 1). Whether or not it fooled the target model and 2). Its distance from the natural input using the L_2, L_1, Wasserstein, Dynamic Time Warping, and Frechet distance functions. Using these measures, we can calculate the following evaluation metrics:

1. The Attack Success Rate (ASR) is the number of examples that successfully trick the target model divided by the total number of examples.

2. Similarity is the average L_2, L_1, Wasserstein, Frechet distance [9], and Dynamic Time Warping distance [29]. We can use these values as approximations for the perceptibility of a perturbation.

Reader Study

There are two main ways by which one can show that a given distance metric is superior for generating adversarial examples. First, one could show that at the same ASR perturbations produced using one distance metric are less perceptible than those produced using another function. Or, second, one could show that one distance metric is more successful at evading defenses than another. As I do not consider defenses in this paper, I will be comparing the efficacy of the various distance metrics via visual similarity to their benign counterparts through a reader study.

For the reader study, I collected a group of 6 medical professionals from Emory University Hospital with experience in analyzing ECG data. Then, after all experiments had been completed and data had been recorded, I analyzed the results of the experiments using all four loss functions against the FCN, MLP, and ResNet models trained on the ECG5000 and ECG200 datasets. To limit the study to a reasonable length, I did not consider the CNN model due to its similarity to the FCN model, and dropped the ECGFiveDays dataset as I already included a binary classification task and felt that the classification problem it posed (i.e day ECG reading was taken) was not particularly relevant to the security of ECG data.

The study contained adversarial examples produced by the following loss functions: L_2, Wasserstein, L_∞, mixed $\alpha = .1$, mixed $\alpha = .25$, mixed $\alpha = .75$, and mixed $\alpha = .9$. For each combination of dataset, model, and loss function I chose 4-5 produced adversarial examples at random plus 13 randomly selected natural examples, giving a total of 218 examples in the study. To select the adversarial examples, I found the smallest value of c

for each model, dataset, and loss function for which the ASR fell into the range of .9-.95.[2] I chose this range as it is a high enough ASR to be considered a powerful attack. At ASR values of 100%, increasing the value of c will not change the ASR but will continue to increase the size of the perturbation. Thus, by limiting the experiment to this slightly lower range of ASR I can ensure a fairer comparison between the various distance metrics.

With the adversarial examples collected, each time series was plotted using Python's Matplotlib package with as a black line plot as shown in Figure 4.1. Then, each image was put into a Google form and the readers were simply asked to score the image as natural or computer generated.

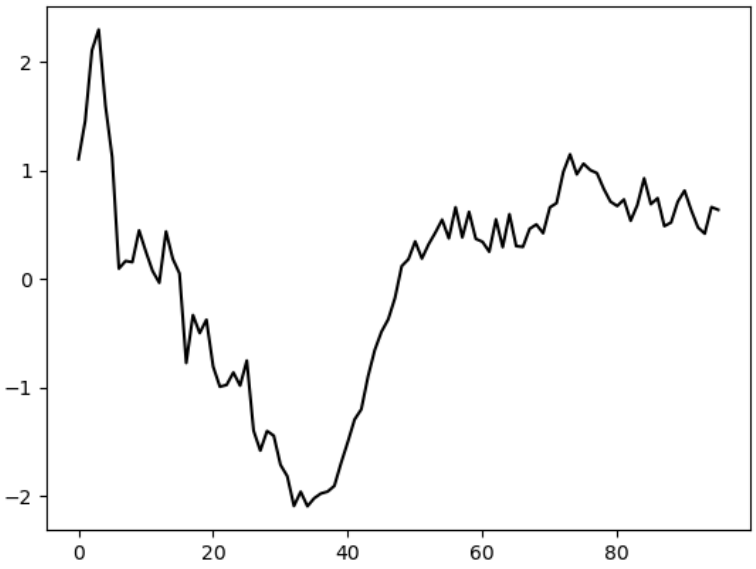

Figure 4.1: An example of one of the images shown to the subjects in the reader study. A plain black line is used so as to avoid any subconscious implications that a blue or red curve might suggest.

2. It is worth noting that for all loss functions the attack struggled to reach high attack success rates against the MLP model trained on the ECG5000 dataset. For this model, the examples were chosen at an ASR of .7.

Chapter 5

Results and Analysis

Non-Mixed Attacks

The standard loss functions analyzed in this paper include the L2, Linf, and Wasserstein distance functions. Figures 5.1, 5.2, and 5.3 demonstrate the effect that the parameter c in (3.5) has on the attack success rate given the various loss functions, models, and datasets under attack. Each figure considers only a single dataset but includes results for all non-mixed distance metrics and all models.

The general trend regardless of model, loss function, or dataset attacked is that as c increases the attack success rate increases as well, as was expected. Additionally, for most models, datasets, and loss functions I was able to find a value of c for which the ASR reached 100%. This highlights the effectiveness of the Carlini-Wagner attack and why I feel it represents the gold standard of adversarial attack techniques.

The first notable trend that can be gathered from Figures 5.1, 5.2, and 5.3 is that for all datasets and loss functions the mlp model is always the most difficult to attack. This can be seen most clearly in the graphs for the attack success rates of the L_∞ and L_2 attacks using the ECG5000 dataset in Figure 5.2.[1] This was surprising, as the mlp model did not

1. It should be noted that both of these attacks did reach a much ASR, namely around .7 for the L_∞ and .8 for the L_2 attacks. We did not include these values on the graph as it took roughly c = 20 for these attack success rates to be achieved, thus throwing off the scaling of the graphs

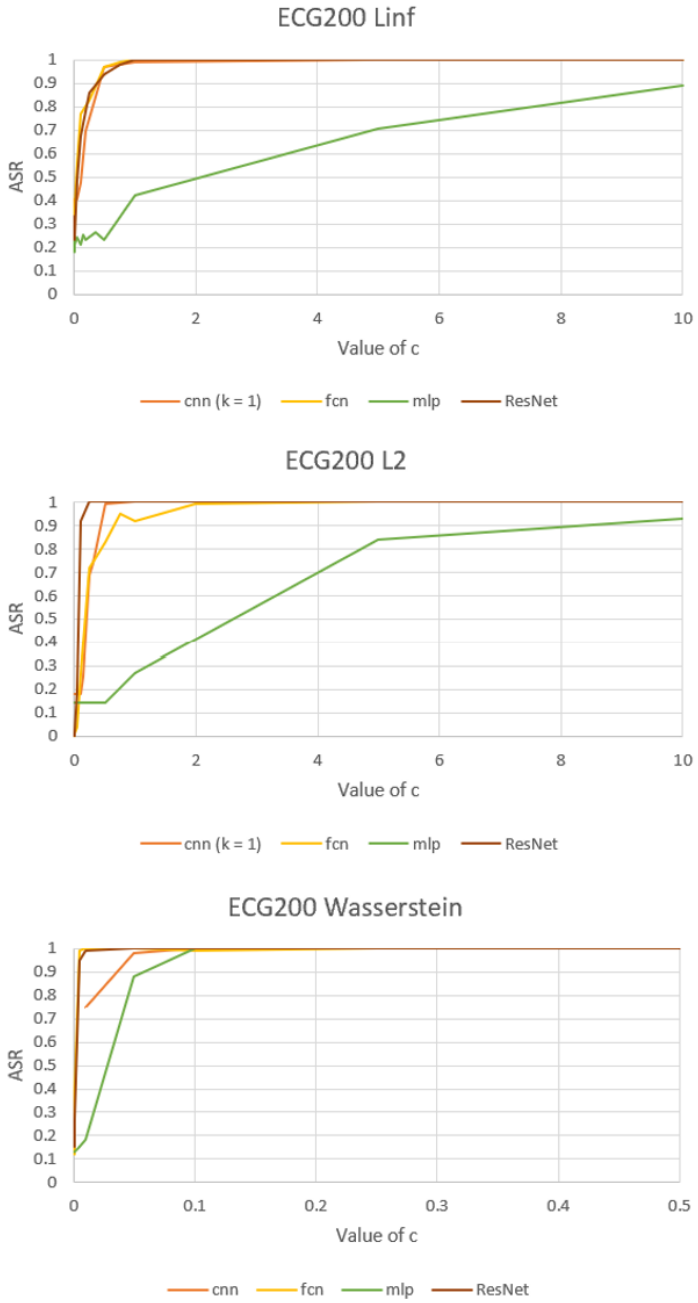

Figure 5.1: The ASR vs. c against the models trained on ECG200 data for the three standard distance metrics.

have any form of defenses that should make it more difficult to attack. One potential explanation for these results is the presence of the dropout layers in the mlp model that the other models lack. These layers set the outputs of randomly selected neurons in the layer to zero during the training process, thereby limiting the amount of information that later

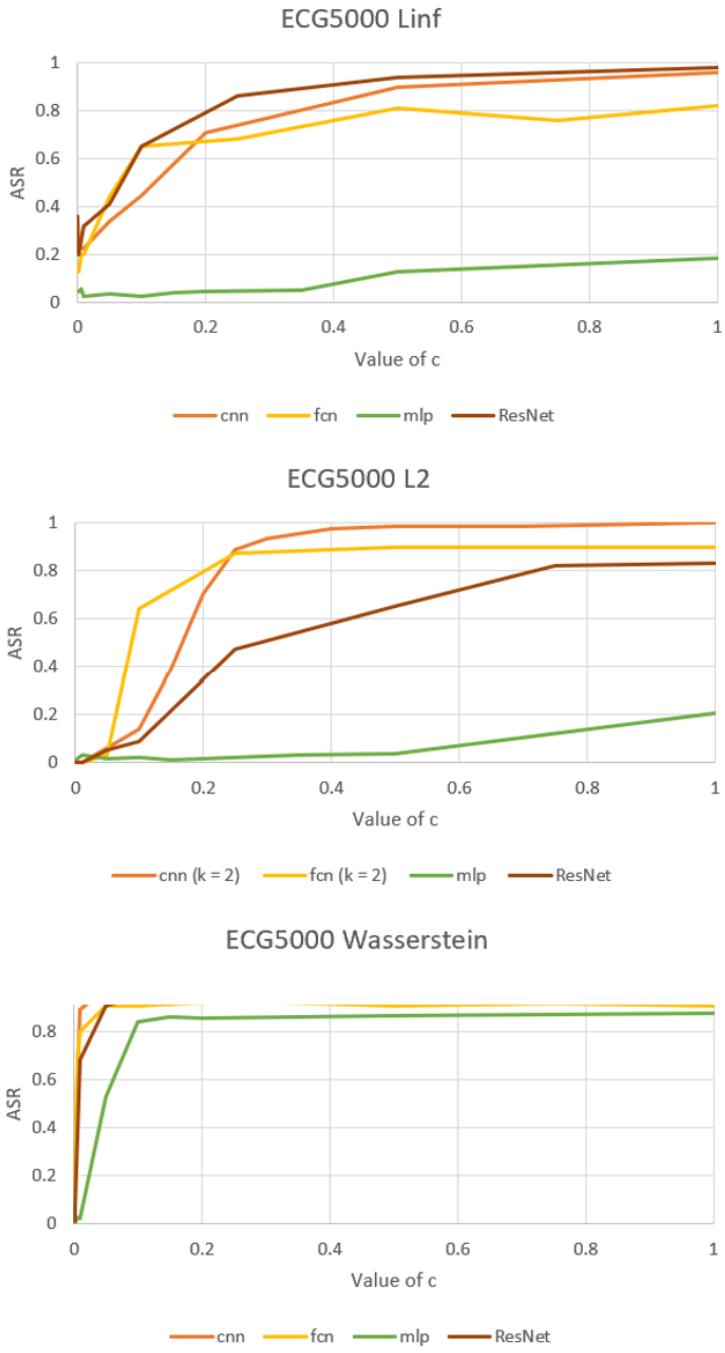

Figure 5.2: The ASR vs. c against the models trained on ECG5000 data for the three standard distance metrics.

layers have access to. This has been empirically shown to increase the robustness of the model and reduce overfitting, and perhaps it makes models more difficult to fool as well.

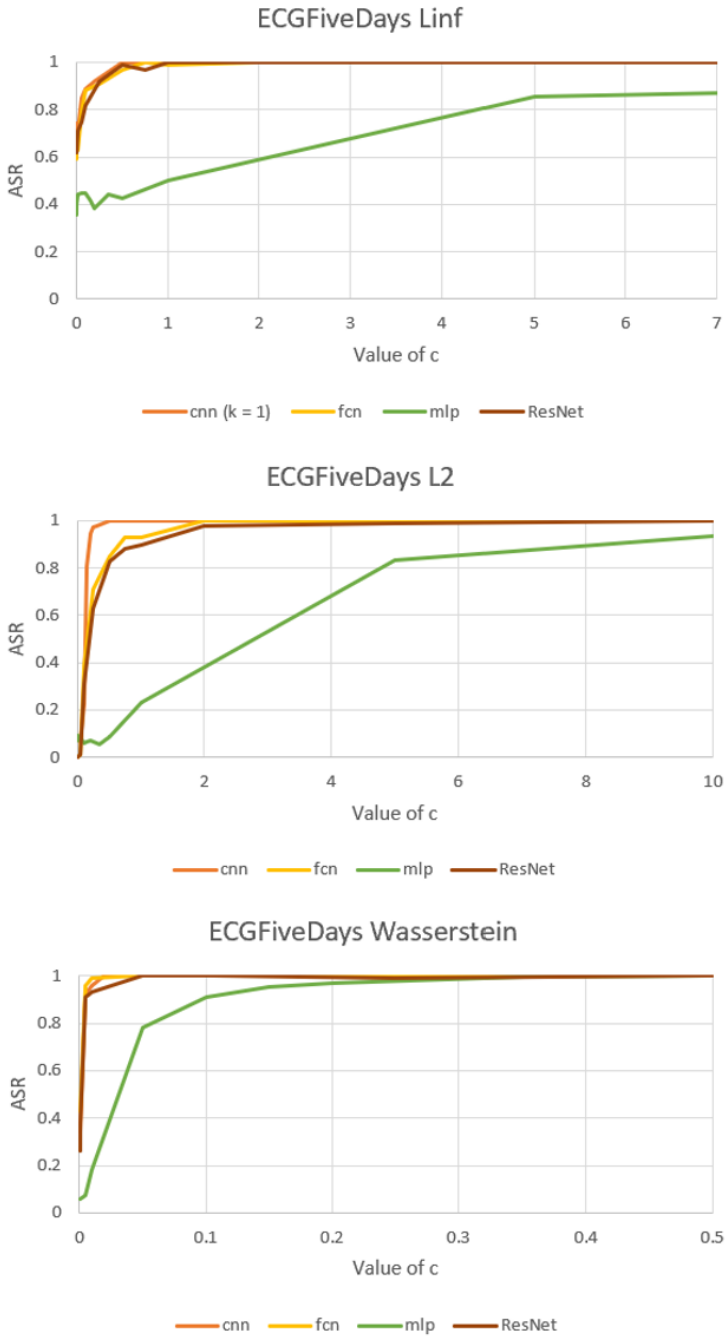

Figure 5.3: The ASR vs. c against the models trained on ECGFiveDays data for the three standard distance metrics.

One final trend worth noting that is visible from Figures 5.1, 5.2, and 5.3 is that the attacks performed using the Wasserstein distance reach 100% ASR with a comparatively much lower value of c. This can be seen across all datasets and models, and most notably

with the ECG5000 mlp model where the Wasserstein attack achieves an ASR of .9, which is the highest of all three attacks considered in this section. I have no reasonable explanation for why this is the case, but it could prove to be useful in real-world attacks where searching for a suitable value of c may not be feasible due to time or computation constraints.

Finally, in Figure 5.4 I have displayed the varying perturbations of the same instance of ECG200 data produced by different loss functions at the same attack success rate. From this figure, we can see that the L_∞ metric produces the most visible perturbation, while the L_2 adversarial example is the most similar to its benign counterpart. However, such visualizations are limited in that they provide only a small sample of the total results. This is why I relied on the results of the reader study to better determine which loss functions are the best proxies for how humans perceive similarity.

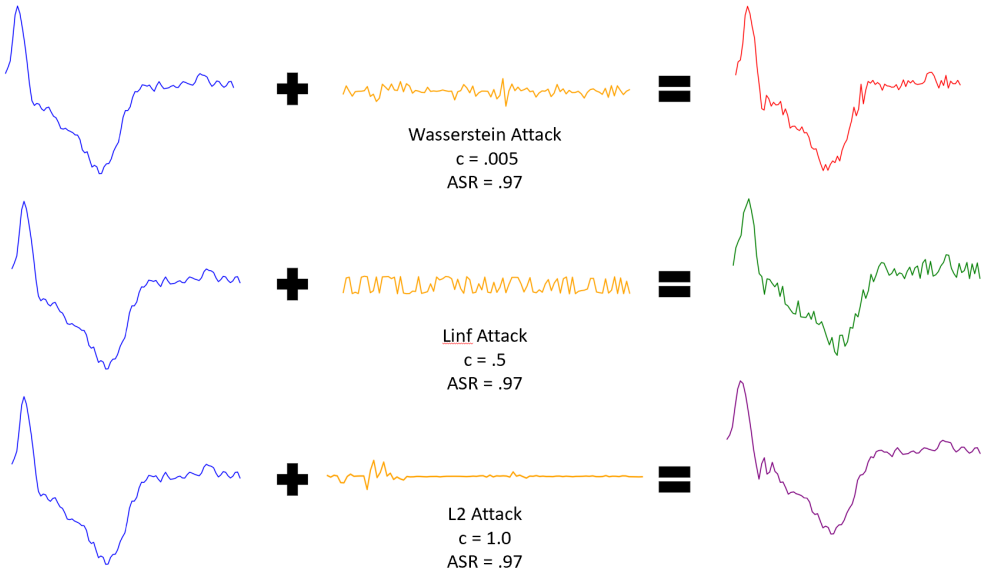

Figure 5.4: Perturbations produced with different attack methods at the same ASR on the same instance of ECG200 data.

Mixed Attacks

In the mixed attack, I produce adversarial examples while varying both the parameter α and the parameter c. My hypothesis was that, given the proper weight of α and c, the mixed loss function would combine the desirable aspects of both the L_2 and Wasserstein

distance functions, thus creating adversarial examples that are more similar to their natural counterparts than those created with other techniques.

Figure 5.5 displays the attack success rate of the mixed attack as the value of α increases. The rows correspond to the dataset used, while the columns correspond to ranges of values of c. The first column represents a low value of c, while the second column is a high c value. All four models are displayed in each graph. Similar to the non-mixed attack, the mixed attack can achieve a 100% attack success rate against nearly all models and datasets at certain values of c and α, with the mlp model trained on ECG5000 being the sole exception. The first notable trend seen here is that as the parameter α increases,

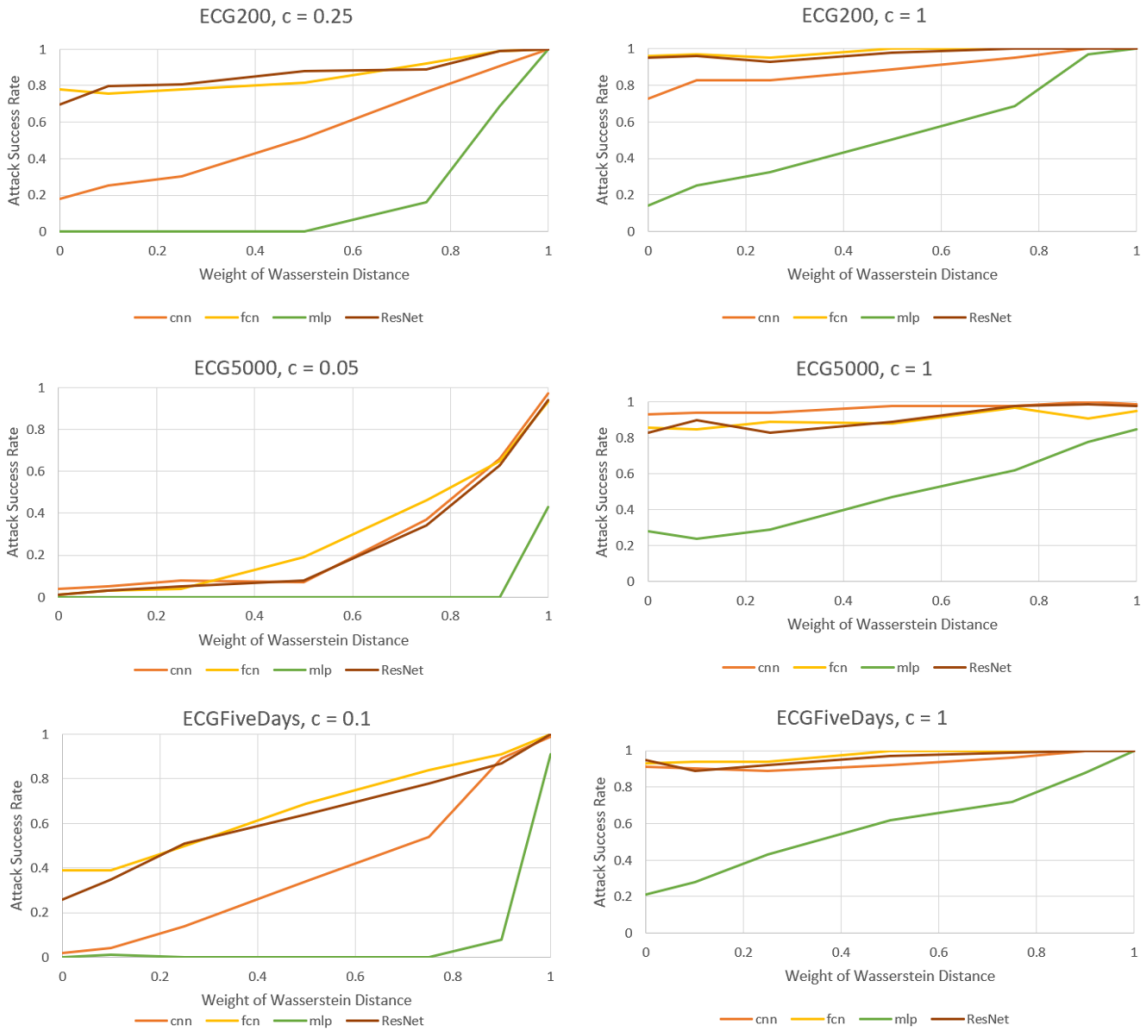

Figure 5.5: How the ASR of the mixed attack changes as the parameter alpha varies at a constant c. Row 1 represents ECG200, Row 2 ECG5000, and Row 3 ECGFiveDays.

the ASR of the attack also tends to increase. This is most visible in comparing the first column of Figure 5.5 to the second column, as at higher values of c the curves tend to flatten out due to the ability of the L_2 attack to achieve a 100% attack success rate as well. Additionally, as was the case with the non-mixed attacks the mlp model is notably more difficult to attack than the other models. It is also the only model for which the attack was unable to achieve a 100% attack success rate. Given the results of the of the non-mixed attack these results, while notable, are unsurprising.

Figure 5.6 displays perturbations produced using the mixed attack at varying levels of α and at similar attack success rates. All perturbations are plotted at the same scale. From these visualizations, we can see similar results to the perturbations shown in the section on the non-mixed attacks: the CW attack using the L_2 norm produces smaller perturbations than when using the Wasserstein distance. Interestingly, there does not seem to be a linear relationship between the value of α and the size of the perturbation. For example, the perturbations produced by the CW attack with $\alpha = .5$ and $\alpha = .9$ are much smaller than those produced when $\alpha = .75$ and $\alpha = .25$ and are arguably smaller than the perturbation produced when using purely the L_2 metric. As mentioned previously, these

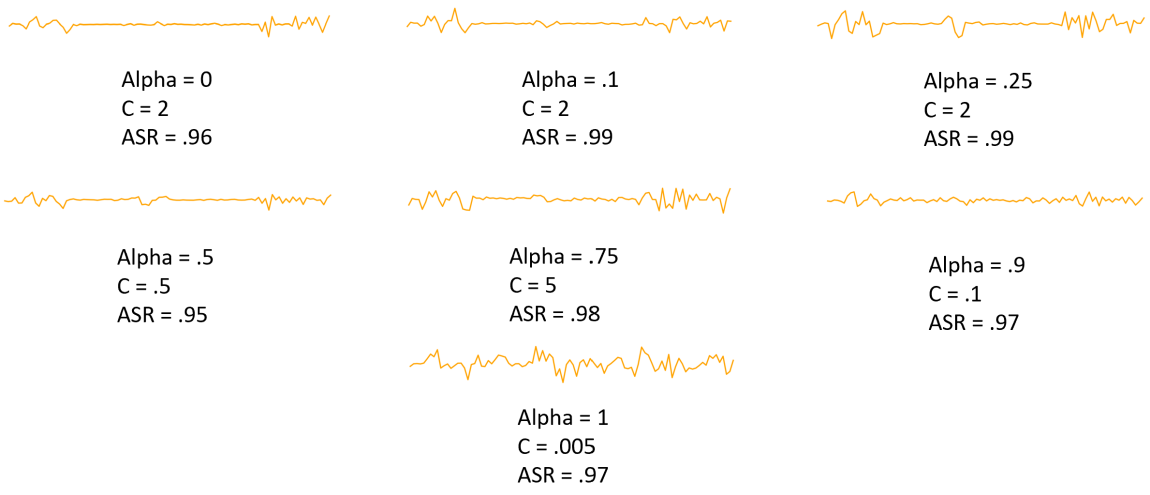

Figure 5.6: Perturbations produced with varying values of α at a similar ASR on the same instance of ECG200 data.

results provide little evidence to support the superiority of one value of α over another or for the mixed CW attack over using a standard loss function. For example, the superiority of perturbation produced when $\alpha = .5$ over $\alpha = .75$ in figure 5.6 could be a result of a lucky random initialization of the perturbation before gradient descent is performed. Therefore, while the visualizations are promising, we must rely on the results of the reader study to make claims about the produced perturbations

Evaluation Metrics

Table 5.1 displays a subset of the results of the experiments targeting each model trained on the ECG200 dataset (the results for the ECG5000 and ECGFiveDays experiments will be relegated to the appendix to avoid clutter in this section as the results were largely similar). I selected data from experiments with an ASR that fell in the range of .89-.95. I chose to not select for 100% ASR for the same reasons as described in the reader study design section. If there were multiple experiments that achieved the same selected ASR, the one with the smallest distances was chosen. Then, I calculated the average perturbation size using only the adversarial examples that successfully tricked the target model. Therefore, the examples that don't successfully trick the model are not included, so the bias as a result of differing ASRs has largely been accounted for. From Table 5.1 we can see the following:

1. For the FCN and ResNet models, a mixed attack formulation achieves the smallest average L_2, L_1, and DTW distance.
2. For all model architectures the Wasserstein based attack achieves the smallest average Wasserstein distance, while for all architectures the L_∞ experiments achieve the smallest average Frechet distance.
3. When the L_2 metric outperforms the mixed attacks, it is by a very small margin as shown by the $\alpha = .5$ experiment for the MLP and CNN models.

ECG200 CNN model results					
Loss function	*Average* L_2	*Average* L_1	*Average Wass.*	*Average DTW*	*Average Frechet*
L2	**0.012**	**0.064**	0.039	**4.414**	0.214
Linf	0.029	0.147	0.079	10.286	**0.203**
Wasserstein	0.079	0.197	**0.006**	12.386	0.566
Mixed .1	0.026	0.080	0.046	5.201	0.272
Mixed .25	0.033	0.090	0.048	5.731	0.302
Mixed .5	0.022	0.074	0.040	5.001	0.251
Mixed .75	0.047	0.120	0.046	7.389	0.357
Mixed .9	0.028	0.097	0.024	6.542	0.280
ECG200 FCN model results					
L2	0.079	0.185	0.066	12.484	0.465
Linf	0.080	0.256	0.096	17.236	**0.337**
Wasserstein	0.079	0.189	**0.006**	13.823	0.664
Mixed .1	0.043	0.145	0.063	10.708	0.438
Mixed .25	0.043	**0.142**	0.059	**10.436**	0.438
Mixed .5	0.087	0.210	0.066	14.125	0.541
Mixed .75	**0.042**	0.146	0.039	10.687	0.465
Mixed .9	0.091	0.222	0.033	14.490	0.533
ECG200 MLP model results					
L2	**0.005**	**0.044**	0.026	**4.028**	0.209
Linf	0.015	0.108	0.051	9.119	**0.168**
Wasserstein	0.028	0.120	**0.005**	9.343	0.422
Mixed .1	0.023	0.095	0.045	7.972	0.350
Mixed .25	0.032	0.116	0.051	9.564	0.386
Mixed .5	0.006	0.048	0.025	4.279	0.238
Mixed .75	0.020	0.090	0.033	7.491	0.328
Mixed .9	0.008	0.059	0.019	5.171	0.239
ECG200 ResNet model results					
L2	0.023	0.088	0.043	7.123	0.366
Linf	0.019	0.120	0.059	9.671	**0.178**
Wasserstein	0.027	0.117	**0.005**	9.138	0.418
Mixed .1	0.008	0.050	0.028	4.380	0.277
Mixed .25	0.011	0.057	0.030	4.813	0.289
Mixed .5	0.020	0.081	0.036	6.716	0.365
Mixed .75	**0.006**	**0.047**	0.021	**4.050**	0.251
Mixed .9	0.008	0.057	0.018	4.997	0.263

Table 5.1: Evaluation metrics for each loss function and model on the ECG200 dataset.

It is important to remember that while attacking the mlp model the L_2 and L_∞ attacks were unable to achieve a 100% attack success rate, while the formulations that included the Wasserstein distance were. Therefore, these results suggest that the mixed distance

functions are the most effective at both achieving a high ASR while also producing small perturbations.

These results provide preliminary evidence for the superiority of the mixed attacks over the other formulations in most cases. Because the mixed distance functions often achieve lower average L_1, L_2, and DTW distances as well as the second lowest average Wasserstein distance, it may be the case that adversarial perturbations created using the mixed distance functions are the most similar to their benign counterparts. However, it is important to note that each of these distance metrics only approximate similarity from a human's perspective. It is possible that some of these metrics may be unrelated to how humans perceive similarity between curves. Therefore, while these results should not be discredited, it is important to consider them alongside the results of the reader study.

Applications to Audio Data: Preliminary Studies

In addition to my experiments with ECG data, I had originally planned to run a similar albeit smaller scale experiment with audio data to broaden the scope of our study. While this portion of the study never made it to fruition, I will include my progress here as a baseline for future work.

I utilized the AudioMNIST [2] dataset for the source of the audio data. AudioMNIST consists of 30000 audio samples of people of varying nationality, age, and gender saying the digits 0-9. The classification task is therefore to determine which digit the speaker reads in the audio clip. To create the training data, I randomly selected 10500 of the audio clips at a sampling rate of 6500 Hz due to memory constraints. Each audio clip was represented as a vector of 16-bit signed integers. I then split the data into a training, validation, and test dataset for model training, fine-tuning, and testing.

For the classifier, I utilized a cnn model like that in the ECG data but with a slightly different structure as shown in Table 5.2. The model achieved an accuracy of 96.6% on the test dataset. Once the model was trained, I began to run similar experiments

AudioMNIST CNN Structure
Conv1D(25, 30, 1)
ReLU
Avg. Pooling(3x3)
Conv1D(50, 30, 1)
ReLU
Global Avg. Pooling
FC(10)
SoftMax

Table 5.2: Model architecture for the AudioMNIST classifier.

to those described above using algorithm 1. The one difference between the audio and ECG experiments is that after the creating the adversarial audio sample I clipped the values of data to respect the integrality constraints. While this is a naive approach, it had little effect on the ability of the examples to fool the target model.

The preliminary experiments consisted of creating adversarial examples using the L_2, Wasserstein, and mixed distance metrics. The highest ASR achieved using the L_2 metric was 69%, while the highest achieved using both the Wasserstein and mixed loss function was 77%. However, in [19] the researchers successfully fool Mozilla's DeepSpeech 100% of the time, so the results should be taken with a grain of salt. It is possible that the CW attack against models trained on audio data require more tuning for c, κ, and the learning rate. In addition, it is possible that the integer representation of the audio data resulted in scaling issues when the loss was calculated. Nonetheless, Figure 5.7 depicts comparisons between the waveforms of the natural audio examples and their corresponding adversarial examples. While hard to determine from the images, much of the noise added was in the background. While I did not make much progress with the audio experiment, there are many directions that future work can take. Recent research has experimented with loss functions that more closely approximate how humans distinguish sounds such as measuring the loudness of the perturbation in decibels [5] or by disguising the perturbation as natural background noise such as a coffee shop or a train stop [19]. Additionally,

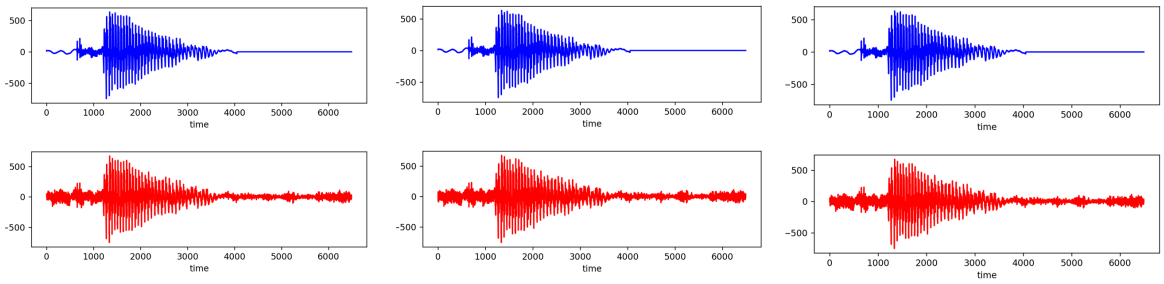

Figure 5.7: Waveforms of an audio sample in blue and their corresponding adversarial example in red. All are produced using the Wasserstein distance.

defenses such as WaveGuard [17] have been developed to specifically protect models that are trained on audio data.

Reader Study Results and Discussion

As mentioned previously, the reader study consisted of 13 natural examples of ECG data and 206 adversarial examples created from the fcn, mlp, and ResNet models trained on the ECG200 and ECG5000 datasets using the loss functions shown in Table 5.3. These examples were evaluated independently by a group of 6 medical professionals, who were asked to predict whether a given example was natural or artificial.

Table 5.3 displays the results of the reader study grouped by the distance function that was used to create the example. For each loss function, 5.3 displays the number of readers who guessed that the example was natural, the total number of predictions, and the sample standard deviation calculated from the data. From the table, we can see that the examples that were the most effective at fooling the readers were those created using the Wasserstein distance, while the worst performing were the examples created using the L_2 metric. Additionally, there was an increasing trend in the percentage of reader fooled as the parameter α increased in the mixed examples.

After collecting the data shown in Table 5.3, a two-sample t-test was performed on many of the pairs of loss functions to determine if the differences shown in the table were statistically significant. However, performing the statistical tests revealed that the results

Reader Study Results								
Loss Function	L_2	L_∞	Wasserstein	$\alpha = .1$	$\alpha = .25$	$\alpha = .75$	$\alpha = .9$	Natural
Number Predicted Natural	72	75	77	76	77	78	81	40
Total Predictions	180	174	168	180	174	180	180	78
Percent Natural	0.400	0.431	0.458	0.422	0.443	0.433	0.450	0.513
S.S.D of Percentage Natural	0.189	0.164	0.141	0.174	0.232	0.184	0.170	0.159

Table 5.3: Reader study results for each group of adversarial images, which shows how many readers predicted the adversarial examples to be natural.

in the table are not statistically significant at the 95% confidence level. For example, the confidence level that the L_2 and Wasserstein experiments produced different results is only .2. Meanwhile, the confidence level for the L_2 and mixed experiment with $\alpha = .9$ is only .29. Therefore, while the results may point to the Wasserstein metric more accurately approximating human perception, the evidence is limited at best. Thus, more research is necessary in order to determine which distance metric is the most likely to fool a discerning reader.

Figure 5.8 depicts two bar charts that show the number of adversarial examples that fall into the following categories: no readers fooled, one reader fooled, two readers fooled, and so on. In the top chart the examples created using the L_2 and L_∞ metric skew farther to the left than those created using the Wasserstein distance. Thus, readers may be more likely to correctly predict that such examples are artificial. Furthermore, the bottom chart provides similar evidence as the distributions skew farther right as α increases.

However, like the conclusions gathered from Table 5.2, the evidence provided by Figure 5.8 for the superiority over the Wasserstein or mixed distance functions over standard loss functions is very limited. Ideally, a Chi-Squared test would be performed to determine if the observed distributions are significantly different from a baseline, which would likely be chosen as the L_∞ metric due to its ubiquity in adversarial machine learning attacks. However, the Chi-Squared test is unreliable unless there are counts of at least 5 in each category for the base line distribution, so this test would provide little additional evidence for whether one distance metric is superior to another. Thus, more research is

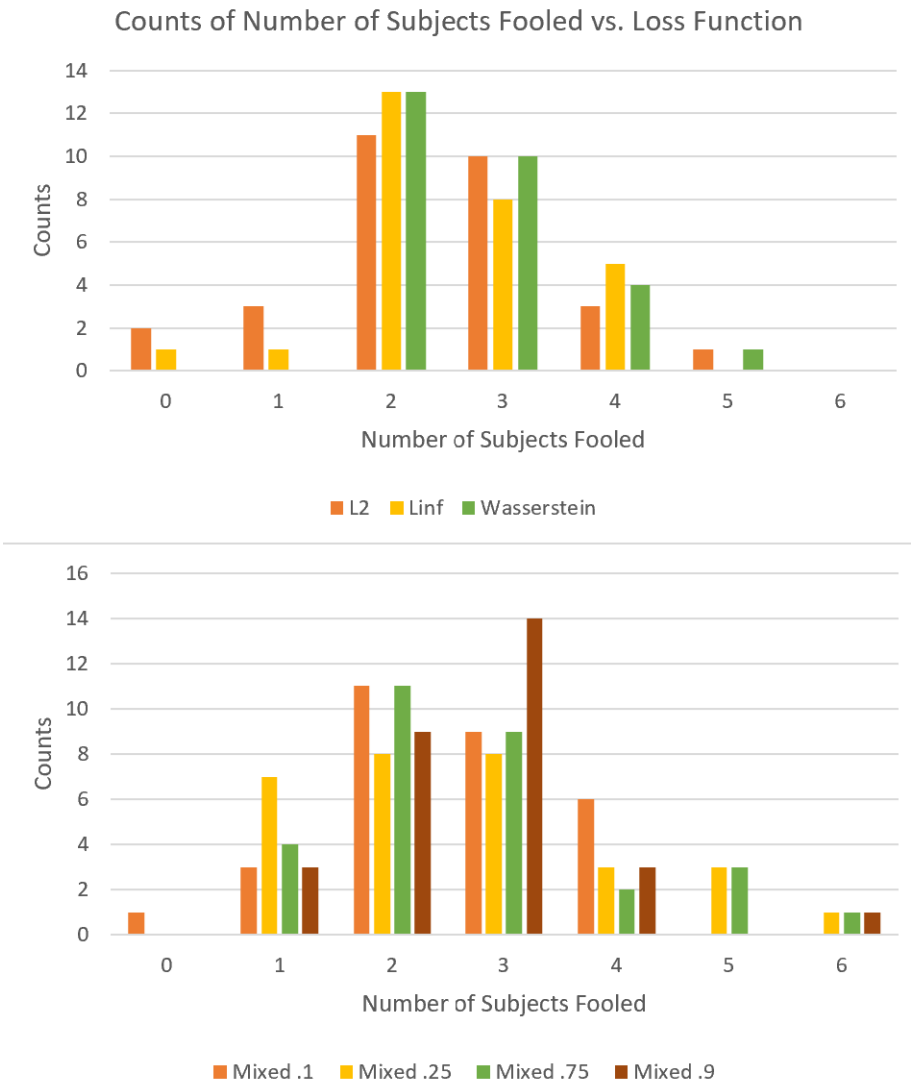

Figure 5.8: A comparison of the distributions for the many loss functions used. Each category is the number of readers fooled for an individual example.

necessary in order to make a mathematically justifiable claim. For example, future work could provide the users with the natural example in addition to the adversarial examples and ask them to grade the amount of manipulation that was required to produce the adversarial data. This design would remove any guesswork from the study that was likely present in our design.

Chapter 6

Conclusions, Limitations, and Future Work

Recent work has demonstrated that deep learning models can be vulnerable to maliciously produced inputs that are designed to be misclassified by the target model. The threat of adversarial attacks thus severely limits the domains in which deep learning can safely be deployed. This fact necessitates research into attack strategies in order to discover the most powerful and hard-to-detect attacks so that our construction of defenses for deep learning models can be properly informed.

In this thesis, I explore the vulnerability of models trained on univariate time series data to adversarial attacks. I focus primarily on ECG data due to its ability to be visualized and interpreted by humans as well as the sensitive nature inherent to medical data. In particular, I utilized a Carlini-Wagner style attack using a variety of different distance functions in order to determine which produces the most visually similar adversarial examples. I compared the L_∞, L_2, Wasserstein, and a novel distance function consisting of a weighted linear combination of the L_2 and Wasserstein distances. I ran a plethora of experiments targeting four different models trained on three different datasets with a CW attack utilizing these four distance functions. From these experiments, I

collected adversarial ECG plots produced by each of these methods. A cohort of medical professionals was used to determine which looked the most natural to trained physicians.

While the reader study did not produce statistically significant results, there was preliminary evidence that distance functions that included a Wasserstein term produced less detectable adversarial examples. These preliminary results in conjunction with the quantitative results of the experiments suggest that the novel mixed distance function may produce minimal adversarial perturbations while achieving a high attack success rate. However, future work should expand on the user study performed in this thesis in order to create an experiment that would allow for more dfinitive conclusions. First, the readers should be shown a larger collection of adversarial examples than those used in our study. This would allow for a larger sample size and give the study greater statistical power. Additionally, and likely more importantly, after completing the study one of the readers informed us that physicians have very little experience analyzing single-heartbeat ECG readings. Thus, the data may have been too unfamiliar to the physicians for them to make informed decisions, which is supported by the 51% accuracy of the physicians on the natural data (Table 5.3). Thus, future work utilizing ECG data should make use of a larger collection of examples taken from ECG readings consisting of more than one heartbeat.

For future work that expands on this project, I have the following suggestions:

1. Compare the results of attacks against models when defenses are in place.
2. Expand to other forms of univariate time series data. For example, audio data and stock prices, which are both interpretable by humans, and also have inherent security risks.
3. Explore the defensive potential of dropout layers in deep neural networks. While they alone cannot prevent attacks as shown, they may prove valuable as part of a larger defensive ecosystem.

4. Experiment with multivariate time series data and similarity when using dimensionality reduction techniques such as principal component analysis (PCA).
5. Experiment with a variety of loss functions using different attack styles that prioritize speed over optimal performance such as the Fast-Gradient Sign Method (FGSM).

Perhaps the most important direction for future work is to determine whether a Carlini-Wagner style attack utilizing the loss functions and method described in this paper could be deployed in real life against a classifier. While the CW attack using the mixed loss function is very powerful and can cause a model to misclassify 100% of the time, this means nothing if the attack is too slow to be deployed on a real-world system without raising human suspicion. Thus, future experiments should determine whether or not such attacks are viable in a real-world scenario.

Appendix A

ResNet Architecture

ResNets are a common neural network architecture for a variety of deep learning tasks and have achieved high levels of performance on both image and time series processing tasks [16]. Their defining feature is the residual connection, also known as skip connections, which allow data to be fed into layers deep in the network without passing through the earlier layers. Skip connections have been found to reduce training time while increasing performance especially on problems where fine-grained details are incredibly important such as segmentation [30].

For the experiments, I utilized a ResNet architecture consisting of 3 blocks. The structure of these blocks is shown in Table A.1, with the only difference between the 1st block and the 2nd and 3rd blocks is the number of feature maps in the convolutional layers. At the beginning of each block, the input data is copied. The first copy goes through the block in a sequential fashion, while the second copy skips to the end of the block. Once the block produces its output, the output and the original input are added together element-wise. This feature vector then has a ReLU function applied to it and is fed in similar fashion to the next residual block. This repeats until after the 3rd residual block, where a standard fully connected layer is applied in the same fashion as the previously described models. Figure A.1 depicts this architecture in a slightly larger ResNet model. While this is not the exact same as the model used throughout our experiments, the core structure is the same.

Residual Block Architecture
Conv1D(64, 8, 1)
Batch Normalization
Relu
Conv1D(64, 5, 1)
Batch Normalization
ReLU
Conv1D(64, 3, 1)
Batch Normalization

Table A.1: Architecture for a singular residual block. The model contains 3 in total

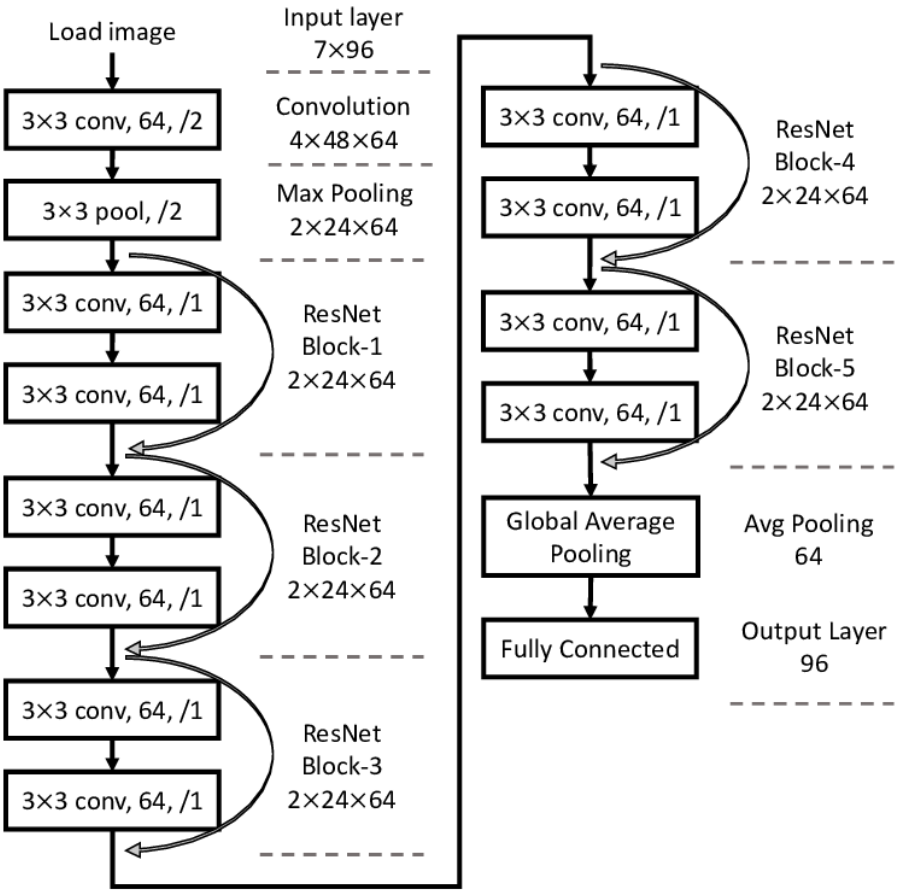

Figure A.1: A visualization of a commonly used pretrained ResNet model known as ResNet-12.

Appendix B

ECGFiveDays and ECG5000 Evaluation Metrics

In this section I will be displaying tables like Table 5.1 containing the measurements for the ECG5000 and ECGFiveDays datasets. The procedure used to generate these tables is the same as for Table 5.1 and the conclusions that can be drawn are largely the same. They are relegated to the appendix simply to improve the readability and flow of the thesis. It is worth noting that for the mlp model trained on the ECGFiveDays dataset, the mixed attack with $\alpha = .1$ did not achieve an ASR in the range of .89-.95, so a lower value was taken. Additionally, the mixed experiment with $\alpha = .75$ was taken at 100% ASR as its next highest ASR was far lower than the .89-.95 ASR range. Additionally, for the mlp model trained on the ECG5000 dataset the L_2 and L_∞ based attacks only reached a 70% ASR. Thus, to make a fair comparison, all examples were sampled from near this ASR for all attacks.

ECGFiveDays CNN model results					
Loss function	*Average* L_2	*Average* L_1	*Average Wass.*	*Average DTW*	*Average Frechet*
L2	0.005	0.040	0.024	3.652	0.158
Linf	0.014	0.101	0.052	9.867	0.156
Wasserstein	0.025	0.115	**0.005**	12.183	0.356
Mixed .1	0.005	0.042	0.026	3.833	0.170
Mixed .25	0.008	0.052	0.028	4.745	0.208
Mixed .5	0.004	0.037	0.019	3.438	0.145
Mixed .75	0.004	0.039	0.016	3.624	0.150
Mixed .9	**0.003**	**0.034**	0.008	**3.204**	**0.121**
ECGFiveDays FCN model results					
L2	0.042	0.128	0.087	13.383	0.324
Linf	0.046	0.193	0.116	20.664	**0.248**
Wasserstein	0.070	0.183	**0.007**	18.771	0.664
Mixed .1	**0.024**	0.105	0.069	**10.682**	0.260
Mixed .25	**0.024**	**0.104**	0.068	10.710	0.269
Mixed .5	0.056	0.158	0.100	16.551	0.369
Mixed .75	0.085	0.205	0.117	21.437	0.432
Mixed .9	0.026	0.106	0.032	10.752	0.321
ECGFiveDays MLP model results					
L2	**0.003**	**0.036**	0.022	**4.101**	0.208
Linf	0.010	0.086	0.041	9.664	**0.144**
Wasserstein	0.026	0.115	**0.005**	12.567	0.432
Mixed .1*	0.004	0.045	0.026	5.086	0.253
Mixed .25	0.005	0.049	0.027	5.445	0.270
Mixed .5	0.004	0.042	0.023	4.719	0.242
Mixed .75*	**0.003**	0.040	0.019	4.507	0.236
Mixed .9	0.004	0.042	0.015	4.668	0.239
ECGFiveDays ResNet model results					
L2	0.003	0.032	0.023	3.533	0.194
Linf	0.014	0.106	0.054	11.264	**0.164**
Wasserstein	0.025	0.114	**0.005**	12.409	0.433
Mixed .1	0.004	0.034	0.024	3.728	0.216
Mixed .25	**0.002**	**0.027**	0.019	**2.986**	**0.164**
Mixed .5	0.003	0.032	0.021	3.530	0.200
Mixed .75	0.003	0.032	0.018	3.519	0.188
Mixed .9	0.003	0.036	0.014	4.003	0.202

Table A.2: Evaluation metrics for each loss function and model on the ECGFiveDays dataset.

ECG5000 CNN model results					
Loss function	*Average* L_2	*Average* L_1	*Average Wass.*	*Average DTW*	*Average Frechet*
L2	**0.032**	**0.113**	0.054	**8.647**	**0.279**
Linf	0.058	0.215	0.096	19.478	0.298
Wasserstein	0.098	0.219	**0.005**	20.705	0.616
Mixed .1	0.061	0.153	0.071	12.649	0.386
Mixed .25	0.083	0.186	0.080	15.504	0.458
Mixed .5	0.061	0.157	0.063	12.669	0.391
Mixed .75	0.110	0.233	0.082	19.723	0.533
Mixed .9	0.138	0.275	0.060	23.659	0.577
ECG5000 FCN model results					
L2	0.221	**0.360**	0.144	**34.724**	0.785
Linf	**0.190**	0.391	0.147	37.036	**0.665**
Wasserstein	0.356	0.446	**0.011**	42.628	1.289
Mixed .1	0.217	0.365	0.139	35.422	0.749
Mixed .25	0.289	0.422	0.152	40.739	0.848
Mixed .5	0.436	0.557	0.197	53.907	1.004
Mixed .75	0.550	0.660	0.242	61.867	1.032
Mixed .9	0.371	0.502	0.098	46.449	0.959
ECG5000 MLP model results*					
L2	0.114	0.238	0.104	27.836	0.720
Linf	0.085	0.256	0.109	29.439	**0.464**
Wasserstein	0.149	0.287	**0.023**	32.422	0.922
Mixed .1	**0.062**	**0.159**	0.076	**18.589**	0.569
Mixed .25	0.072	0.171	0.076	20.030	0.609
Mixed .5	0.103	0.222	0.096	25.739	0.667
Mixed .75	0.101	0.221	0.084	25.483	0.691
Mixed .9	0.168	0.318	0.099	37.132	0.782
ECG5000 ResNet model results					
L2	0.039	0.136	0.072	14.708	0.479
Linf	0.042	0.178	0.085	19.195	**0.281**
Wasserstein	0.113	0.247	**0.012**	26.161	0.827
Mixed .1	0.027	**0.108**	0.056	**11.525**	0.387
Mixed .25	**0.024**	**0.108**	0.055	11.685	0.403
Mixed .5	0.041	0.137	0.068	14.625	0.478
Mixed .75	0.040	0.138	0.057	14.818	0.486
Mixed .9	0.053	0.163	0.049	17.142	0.554

Table A.3: Evaluation metrics for each loss function and model on the ECG5000 dataset.

Bibliography

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 214–223. PMLR, 06–11 Aug 2017.

[2] Sören Becker, Marcel Ackermann, Sebastian Lapuschkin, Klaus-Robert Müller, and Wojciech Samek. Interpreting and explaining deep neural networks for classification of audio signals. *CoRR*, abs/1807.03418, 2018.

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

[4] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks, 2016.

[5] Nicholas Carlini and David A. Wagner. Audio adversarial examples: Targeted attacks on speech-to-text. *CoRR*, abs/1801.01944, 2018.

[6] George E. Dahl, Jack W. Stokes, Li Deng, and Dong Yu. Large-scale malware classification using random projections and neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 3422–3426, 2013.

[7] Hoang Anh Dau, Eamonn Keogh, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, Yanping, Bing Hu, Nurjahan Begum, Anthony Bagnall, Abdullah Mueen, Gustavo Batista, and Hexagon-ML. The ucr time series classification archive, October 2018. https://www.cs.ucr.edu/ eamonn/time$_s eries_d ata_2$018.

[8] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. Robust physical-world

attacks on deep learning visual classification. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1625–1634, 2018.

[9] Chenglin Fan and Benjamin Raichel. Computing the fréchet gap distance. *Discrete & Computational Geometry*, 65:1244–1274, 2020.

[10] Noureen Fatima, Ali Shariq Imran, Zenun Kastrati, Sher Muhammad Daudpota, and Abdullah Soomro. A systematic literature review on text generation using deep neural network models. *IEEE Access*, 10:53490–53503, 2022.

[11] M.W Gardner and S.R Dorling. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric Environment*, 32(14):2627–2636, 1998.

[12] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples, 2014.

[13] Kathrin Grosse, Nicolas Papernot, Praveen Manoharan, Michael Backes, and Patrick D. McDaniel. Adversarial perturbations against deep neural networks for malware classification. *CoRR*, abs/1606.04435, 2016.

[14] Abhishek Gupta, Alagan Anpalagan, Ling Guan, and Ahmed Shaharyar Khwaja. Deep learning for object detection and scene perception in self-driving cars: Survey, challenges, and open issues. *Array*, 10:100057, 2021.

[15] Jonathan Weber Lhassane Idoumghar Hassan Ismail Fawaz, Germain Forestier and Pierre-Alain Muller. Deep learning for time series classification: a review, March 2019. https://doi.org/10.1007/s10618-019-00619-1.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[17] Shehzeen Hussain, Paarth Neekhara, Shlomo Dubnov, Julian McAuley, and Farinaz Koushanfar. Waveguard: Understanding and mitigating audio adversarial examples, 2021.

[18] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world, 2016.

[19] Siddique Latif, Rajib Rana, and Junaid Qadir. Adversarial machine learning and speech emotion recognition: Utilizing generative adversarial networks for robustness, 2018.

[20] Zhouhan Lin, Roland Memisevic, and Kishore Konda. How far can we go without convolution: Improving fully-connected networks, 2015.

[21] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation, 2014.

[22] Dongyu Meng and Hao Chen. Magnet: a two-pronged defense against adversarial examples. *CoRR*, abs/1705.09064, 2017.

[23] Shervin Minaee, Yuri Boykov, Fatih Porikli, Antonio Plaza, Nasser Kehtarnavaz, and Demetri Terzopoulos. Image segmentation using deep learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(7):3523–3542, 2022.

[24] Shervin Minaee, Nal Kalchbrenner, Erik Cambria, Narjes Nikzad, Meysam Chenaghlu, and Jianfeng Gao. Deep learning–based text classification: A comprehensive review. *ACM Comput. Surv.*, 54(3), apr 2021.

[25] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks, 2015.

[26] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples, 2016.

[27] Nicolas Papernot and Patrick D. McDaniel. On the effectiveness of defensive distillation. *CoRR*, abs/1607.05113, 2016.

[28] Pouya Samangouei, Maya Kabkab, and Rama Chellappa. Defense-gan: Protecting classifiers against adversarial attacks using generative models. *CoRR*, abs/1805.06605, 2018.

[29] Pavel Senin. Dynamic time warping algorithm review. 01 2009.

[30] Lamia H. Shehab, Omar M. Fahmy, Safa M. Gasser, and Mohamed S. El-Mahallawy. An efficient brain tumor image segmentation based on deep residual networks (resnets). *Journal of King Saud University - Engineering Sciences*, 33(6):404–412, 2021.

[31] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks, 2013.

[32] S. S. Vallender. Calculation of the wasserstein distance between probability distributions on the line. *Theory of Probability & Its Applications*, 18(4):784–786, 1974.

[33] Pin Wang, En Fan, and Peng Wang. Comparative analysis of image classification algorithms based on traditional machine learning and deep learning. *Pattern Recognition Letters*, 141:61–67, 2021.

[34] Eric Wong, Frank R. Schmidt, and J. Zico Kolter. Wasserstein adversarial examples via projected sinkhorn iterations. *CoRR*, abs/1902.07906, 2019.

[35] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Yuyin Zhou, Lingxi Xie, and Alan L. Yuille. Adversarial examples for semantic segmentation and object detection. *CoRR*, abs/1703.08603, 2017.

[36] Wei Zhang, Xiaodong Cui, Ulrich Finkler, Brian Kingsbury, George Saon, David Kung, and Michael Picheny. Distributed deep learning strategies for automatic speech recognition. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5706–5710, 2019.