Title
Secure and Private Machine Learning for Smart Devices

Author
ALZANTOT, MOUSTAFA FARID TAHA MOHAMMED

Publication Date
2019

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA
Los Angeles

Secure and Private Machine Learning for Smart Devices

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Computer Science

by

Moustafa Farid Taha Mohammed Alzantot

2019

ABSTRACT OF THE DISSERTATION

Secure and Private Machine Learning for Smart Devices

by

Moustafa Farid Taha Mohammed Alzantot
Doctor of Philosophy in Computer Science
University of California, Los Angeles, 2019
Professor Mani B. Srivastava, Chair

Nowadays, machine learning models and especially deep neural networks are achieving outstanding levels of accuracy in different tasks such as image understanding and speech recognition. Therefore, they are widely used in the pervasive smart connected devices, e.g., smartphones, security cameras, and digital personal assistants, to make intelligent inferences from sensor data. However, despite their high level of accuracy, researchers have recently found that malicious attackers can easily fool machine learning models. Therefore, this brings into question the robustness of machine learning models under attacks, especially in the context of privacy-sensitive and safety-critical applications.

In this dissertation, we investigate the security and privacy of machine learning models. First, we consider the problem of adversarial attacks that fool machine learning models under the practical setting where the attacker has limited information about the victim model and also restricted access to it. We introduce, *GenAttack*, an efficient method to generate adversarial examples against black-box machine learning. *GenAttack* requires 235 times fewer model queries than previous state-of-the-art methods while achieving a higher success rate in targeted attacks against the large scale Inception-v3 image classification models. We also show how *GenAttack* can be used to overcome a set of different recently proposed methods of model defenses. Furthermore, while prior research on adversarial attacks against machine learning models has focused only on image recognition models due to the challenges of attacking models of other data modalities such as text and speech, we show *GenAttack* can be extended to attack both speech recognition and text understanding models with a high success rate. We achieve 87% success rate against a speech command recognition model and

97% success rate against a natural language sentiment classification model.

In the second part of this dissertation, we focus on methods for improving the robustness of machine learning models against security and privacy threats. A significant limitation of deep neural networks is their lack of explanation for their predictions. Therefore, we present *NeuroMask*, an algorithm for generating accurate explanations of the neural network prediction results. Another serious threat against the voice-controlled devices is the *audio spoofing attacks*. We present a deep residual convolutional network for detecting two different kinds of attacks: the *logical access* attack and the *physical access* attack. Our model achieves 6.02% and 2.78% equal error rate (EER) on the evaluation datasets of the ASVSpoof2019 competition for the detection of the logical access, and physical access attacks, respectively. To alleviate the privacy concerns of unwanted inferences while sharing private sensor data measurements, we introduce, *PhysioGAN*, a *novel* model architecture for generating high-quality synthetic datasets of physiological sensor readings. Using evaluation experiments on two different datasets: ECG classification dataset and motion sensors for human activity recognition dataset, we show that compared to previous methods of training sensor data generative models *PhysioGAN* is capable of producing synthetic datasets that are both more accurate and more diverse. Therefore, synthetic datasets generated by *PhysioGAN* are a good replacement to be shared instead of the real *private* datasets with a moderate loss in their utility. Finally, we show how we apply the differential privacy techniques to extend the training of the generative adversarial networks to produce synthetic datasets with formal privacy guarantees.

The dissertation of Moustafa Farid Taha Mohammed Alzantot is approved.

Christina Panagio Fragouli

Guy Van den Broeck

Kai-wei Chang

Mani B. Srivastava, Committee Chair

University of California, Los Angeles

2019

To my parents and my grandmother,

whose unconditional love and support is the most precious thing in my life . . .

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

All praise be to Allah, the most gracious the most merciful, who has allowed me to perform this work. *"Praise to Allah, who has guided us to this; and we would never have been guided if Allah had not guided us."* [The holly Quran 7:43].

First,I would like to thank my Ph.D. advisor, Prof. Mani Srivastava, who has showered me with his support and encouragement during my entire Ph.D. journey. I admire his dedication to knowledge, education, and technology advances. Without his assistance, I wouldn't have been able to explore challenging research topics and ideas without fear of failure.

Next, I would like to thank my other committee members, Prof. Christina Panagio Fragouli, Prof. Guy Van den Broeck, and Prof. Kai-Wei Chang for their support and thoughtful feedback. I am also grateful for my former advisors: Prof. Moustafa Youssef, Prof. Fahim Kawsar for cultivating my interest in research and for their help with my Ph.D. admission, and for my mentors and hosts at the companies where I spent my summer internships: Google, Facebook, and Qualcomm.

The work presented in this dissertation wouldn't have been done the same way without the help I received from my co-authors and collaborators: Prof. Kai-Wei Chang, Prof. Simon Julier, Prof. Cho-Jui Hsieh, Prof. Santosh Kumar, Prof. Saman Zonouz, Dr. Supriyo Chakraborty, Dr. Gabriel Salles-Loustau, Yash Sharma, Ahmed Elgohary, and Huang Zhang. Among my collaborators, I would like to dedicate a special thanks to Dr. Supriyo Chakraborty (IBM Research) for his long term collaboration efforts and valuable advice. I am also grateful for my amazing lab-mates at the Networked and Embedded Sensing Lab (NESL): Dr. Luis Garcia, Dr. Bharathan Balaji, Dr. Paul Martin, Dr. Chenguang Shen, Dr. Salma Elmalki, Dr. Bo Jhang Ho, Dr. Fatima Anwar, Amr Elanwar, Tianwei Xing, Renju Liu, Sandeep Singh Sandha, EunSun Lee, Nathaniel Snyder, Joseph Noor, Ziqi Wang, Akash Deep Singh, and Brian Wang.

I would also like to thank my friends at UCLA: Dr. Ahmed Hareedy, Dr. Yasser Shoukry, Dr. Omar Hussein, Mohammed Karmoose, Yahya Eissa, Ahmed Alaa and Mahmoud Rashad for the fun moments we had together. I would also like to honor the memory of my late friend, *Dr. Moustafa Mahmoud Elhamshary*, who passed away in March 2018. I am sure he would have been so happy for me if he was still among us. He always has been a good friend and a great mentor.

Most important than all, I can't find the words to express the amount of gratitude I have for my beloved family. For my grandmother, who keeps me all day and night in her prayers. For my parents who gave me everything, they can give in this life. For my lovely sisters: Fatima, Samia, Amal, Eman, and brothers: Taha, Mohammed, Aly. And for my beautiful nieces and nephews who brought a lot of joy into my life. I can't be thankful enough for the amount of support all of them have been giving me during my entire life.

Finally, I would also like to express my gratitude to the funding agencies who supported my research and studies. The DAIS-ITA project has supported my research under Agreement Number W911NF-16-3-0001, the National Science Foundation under award # CNS-1705135 and OAC-1640813, and the NIH Center of Excellence for Mobile Sensor Data-to-Knowledge (MD2K) under award 1-U54EB020404-01. Any findings in this material are those of the author(s) and do not reflect the views of any of the above funding agencies.

VITA

2008	B.S. *(Distinction with honors)* Computer Engineering and Automatic Control, Tanta University, Egypt.
2014	Awarded the Common Market for Eastern and Southern Africa (COMESA) 2013 innovation award.
2014	Graduate Student Researcher, UCLA.
2016	Software Engineering Intern, Google Inc., Mountain view, California.
2017	Teaching Assistant, Computer Science Department, UCLA, California.
2017	M.Sc. (Computer Science), UCLA, Los Angeles, California.
2018	Software Engineering Intern, Facebook, Inc., Menlo Park, California.
2019	Awarded the 5th place award in the NIST Differential Privacy Synthetic Data Challenge

PUBLICATIONS

Moustafa Alzantot, Luis Garcia, and Mani B. Srivastava. *"PhysioGAN: Training High Fidelity Generative Model for Physiological Sensor Readings"*. Under review.

Moustafa Alzantot, Ziqi Wang, and Mani B. Srivastava. *"Deep Residual Neural Networks for Audio Spoofing Detection"*. The 20th Annual Conference of the International Speech Communication Association INTERSPEECH 2019, Graz, Austria, Sep. 15-19, 2019.

Moustafa Alzantot, Yash Sharma, Supriyo Chakraborty, Huan Zhang, Cho-Jui Hsieh, and Mani B Srivastava. *"Genattack: Practical black-box attacks with gradient-free optimization"*. Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2019, Prague, Czech Republic, July 13-17, 2019, ACM 2019.

Saman Aliari Zonouz, Gabriel Salles-Loustau, Mani Srivastava, and **Moustafa Alzantot**. *"Value-based Information Flow Tracking for Mobile, Wearable, and IOT Sensor Devices"*. US Patent Application No: 62/567,498. 2018.

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bojhang Ho, Kai-Wei Chang, and Mani B. Srivastava. *"Generating natural language adversarial examples"*. Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP 2018), Brussels, Belgium, Oct 31-Nov 4, 2019.

Moustafa Alzantot, Bharathan Balaji, and Mani B. Srivastava. *"Did you hear that? adversarial examples against automatic speech recognition"*. Presented at the NIPS 2017 Workshop on Machine Deception, Long beach, California, USA. Dec 8, 2017

Moustafa Alzantot, Yingnan Wang, Zhengshuang Ren, and Mani B. Srivastava. *"RSTensorflow: GPU enabled Tensorflow for Deep Learning on Commodity Android Devices"*. Proceedings of the 1st International Workshop on Deep Learning for Mobile Systems and Applications, ACM 2017.

Moustafa Alzantot, Supriyo Chakraborty, and Mani B. Srivastava. *"SenseGen: A Deep Learning Architecture for Synthetic Sensor Data Generation"*. Proceedings of the 2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops) (pp. 188-193). IEEE 2017.

CHAPTER 1

Introduction

Smart devices powered by machine learning and artificial intelligence play an increasingly expandin role in our daily lives. Recent statistics show that, in 2019, roughly a quarter of US adults use at least one wearable device [Wur]. Wearables are equipped with an array of sensors, e.g., cameras, microphones, and motion sensors, to sense the surrounding environment and perceive user actions. Smart home devices, e.g., Google Home, Amazon Echo, and Alexa, also rely on the microphone and other sensing modalities to interact with the user in a convenient natural way. A variety of innovative applications such as mobile health monitoring [WMH18, GDV18], self-driving cars [YZY18], biometrics [Che16], augmented reality games [AKP19], and others have taken advantage of the advances in machine learning research. Not only smart devices and their applications took advantage of the progress in machine learning research, but also machine learning community has gained a lot from the proliferation of smart devices. Smart devices with their connectivity and sensor-rich hardware represent a golden mine for large scale collection of datasets which are needed to train machine learning models. Therefore, we believe these mutual benefits have made a win-win partnership between intelligent IoT devices and machine learning models.

Nowadays, deep neural networks [LBH15] achieve state-of-the-art results in many tasks, including image recognition [SLJ15], speech recognition [HDY12, AAA16], text translation [WSC16], and sequential decision making such as playing games [MKS13, Gib16]. Nevertheless, despite their outstanding levels of accuracy, recently researchers have been concerned about the robustness of machine learning, especially deep neural networks, against malicious attacks. Recent studies have illustrated the vulnerability of deep neural networks (DNNs) to adversarial examples [SZS13, GSS14]. For instance, a virtually imperceptible perturbation to an image can cause

a well-trained image classification model to misclassify. Targeted adversarial examples can even cause misclassification to a chosen class picked by the attacker. Moreover, researchers have shown that these adversarial examples may remain effective in the physical world [KGB16a, AEI17]. The lack of robustness exhibited by DNNs to adversarial examples has raised serious concerns for safety-critical applications.

Furthermore, since machine learning models tend to *leak* information about their training datasets. Recent research [FJR15, SSS17, SZH18] have shown that attackers can *reverse engineer* models to reveal parts of the training dataset. The leakage of private training datasets presents a severe privacy threat for models trained on sensitive data such as the medical and physiological sensor readings. The adversary can also interfere with the model training process. Due to the cost of large-scale training models, the training process is often either fully or partially outsourced due to the lack of computation resources or necessary training data. This practice has been shown to introduce security risks known as *backdoor attacks* [GDG17]. For instance, an adversary may maliciously craft a *backdoored* model and publish it over the internet. The backdoored model will produce normally expected results except for the case when the input contains a specific trigger *designated* by the adversary. In the presence of this trigger, the *backdoored* model will produce misclassification *chosen by the adversary*. This threat is intensified by the inherent limitation of *deep neural networks* to explain their decisions [Lip16a]. Another severe attack that threatens the voice-controlled devices is the '*audio spoofing attack*' where the adversary may use computer-generated speech or replay a recording to gain illegitimate control over user devices. These attacks have advanced a lot over the past few years because speech synthesis [ODZ16, WWK16, JBW18] and voice conversion [TCS16, HLH18] have also progressed a lot over the past decade reaching the point where it has become very challenging to differentiate between their results and genuine users' speech.

The vulnerabilities of machine learning models against security attacks and privacy threats raises a big concern against the continuing adoption of deep learning in the context of ubiquitous smart devices.

1.1 Aim and Objective

Motivated by the above, in this dissertation, we investigate answers the following questions:

- **How can an adversary attack machine learning with limited information and restricted access?** Previous research efforts on adversarial attacks have mainly considered the *white-box* settings where an attacker possess complete information [GSS14, CW17a] about the victim model. In the more practical *black-box* setting, the attacker does not have access to this information. While recently researchers [CZS17a, PMG16] have also studied the *black-box* attack settings, the existing solutions do not scale well because they require a massive number of queries with the victim model. Therefore, these solutions are infeasible under practical attack scenario where the attacker has only restricted access to the victim models.

- **Could we generate *adversarial examples* against models outside the image recognition domain?** Most of the studies on how to generate adversarial examples have considered image recognition models. However, deep neural network models for other data modalities such as speech and text are now widely deployed on a vast majority of smart devices (e.g., phones and digital personal assistants). Therefore, we investigate the challenges of performing adversarial attacks against deep neural networks for data modalities other than images, mainly speech and text.

- **Could we increase the robustness of machine learning models against attacks?** We touch upon the aspect of improving the robustness against different attacks by addressing the issue of how to explain the deep neural network decisions and how to defend voice-controlled devices against audio spoofing attacks.

- **Could we use *generative models*, such as the *generative adversarial networks* to produce high fidelity synthetic physiological signals?** Generative adversarial networks have been one of the most active topics of research over the past few years. Nevertheless, they have been mainly applied to image generation and, to a lesser extent, text and tabular datasets. Only preliminary efforts [ACS17, EHR17] have been made to using them to real-valued time-series physiological sensor readings. The existing solutions suffer from producing samples that are

either inaccurate or lacking diversity. Therefore, we aim to study how to introduce a new approach for model training to generate high-quality synthetic datasets of sensor readings. The synthetic dataset can be utilized as a replacement for the original *private* data for privacy purposes. We study both aspects of how to maximize the utility of synthetic datasets and how to achieve strong formal guarantees that of the limits of private information disclosed by *generative model* training.

1.2 Contribution and Organization

The contribution of this dissertation is multifold covering both directions of research in both attacks and defenses of machine learning models. The following chapters of this dissertation are divided into two parts. The first part presents novel methods of adversarial attacks against machine learning models.

- **Chapter 2:** In this chapter, we introduces *GenAttack*, a *novel* algorithm for practical attacks against *black-box* machine learning models. We evaluate *GenAttack* against different image recognition models and datasets. Against the MNIST and CIFAR-10 classification models, *GenAttack* requires roughly 2000 times less number of interaction queries with the *victim* model than the previous state-of-the-art method of *black-box* adversarial attacks. Against the large scale, `Inception-v3` [SLJ15] ImageNet classification model, *GenAttack* requires roughly 235 times less number of queries while achieving higher attack success rate. We also show how *GenAttack* can bypass several different recently proposed defenses.

- **Chapter 3:** In this chapter, we extend *GenAttack* to target speech recognition models, which are now essential for the operation of smart voice-controlled devices such as phones, wearable, and home assistants. We demonstrate that *GenAttack* can cause targeted misclassification of *speech commands* with 87% success rate by adding small background noise. We conducted a human study to study the effect of added noise on human perception. We found that the adversarial noise did not alter 89% of human listeners decisions.

- **Chapter 4:** In this chapter, we present another extension of *GenAttack* to target *natural*

language understanding models. These models are now widely used in different applications such as the digital personal assistants *(e.g., Siri, Alexa, and Google assistant)* offering natural conversation-like communication with their users. Attacking *Natural language understanding* models has its own difficulties as we explain in details in Chapter 4. We conduct experiments against two different kinds of models, *sentiment analysis model*, and *textual entailment* model. *GenAttack* achieve success rates of 97% and 70%, respectively. We additionally demonstrate that 92.3% of the successful sentiment analysis adversarial examples are classified to their original label by 20 human annotators.

- **Chapter 5:** In this chapter, we introduce *NeuroMask* as a method for generating interpretable explanations of neural network results. We conduct experiments using different image classification models and show how *NeuroMask* can be used to identify the *backdoor trigger* of misclassified images by backdoored models.

- **Chapter 6:** In this chapter, we present our solution for the ASVSpoof2019 competition [con19], which aims to develop countermeasure systems that distinguish between audio spoofing attacks and genuine speeches. We develop a deep residual convolution neural network and evaluate model accuracy using three different sets of features. In the two tracks of the competition, *logical access attacks*, and *physical access attacks*, our model reduces the equal error rate (EER) metric of attack detection by 25% and 75%, respectively.

- **Chapter 7:** In this chapter, we introduce, *PhysioGAN*, a generative model to produce high fidelity synthetic physiological sensor data readings. Using experiments on two different real-world datasets: ECG Atrial Fibrillation (AFib) classification dataset, and Human Activity Recognition (HAR) from motion sensors dataset. Classification models trained on synthetic data generated by *PhysioGAN* have only 10%, and, 20% decrease in their classification accuracy than classification models trained on the real data. Compared to existing approaches, these results demonstrate that *PhysioGAN* is capable of generating higher levels of quality datasets. Therefore, the synthetic datasets generated by *PhysioGAN* can be used to replace the real *private* sensor data. This alleviates the issues of privacy risks of sharing physiological sensor data.

- **Chapter 8:** In the final chapter of this dissertation, we touch upon the idea how the training of generative models can be modified using differential privacy [Dwo11] to provide strong and formal privacy guarantees of the limits of private information disclosed in their synthetic datasets. The methods and algorithms we describe in our chapter were used in our *award winning* submissions in the 2018 Differential Privacy Synthetic Data Challenge [Ven19] where we were awarded three different prizes: *4th place award in round 1, 5th place award in round 3, and the open-source contribution award).*

1.3 Notation

In the remainder of this dissertation, we adopt the following mathematical notation:

- $\mathbb{R}$ denotes the set of real numbers.
- $\mathbb{R}^n$ denotes the set of vectors of n real numbers.
- $\mathbb{R}^{m \times n}$ denotes the set of $m \times n$ matrices of real numbers.
- $[0, 1]^n$ denotes the set of vectors of n real numbers between 0 and 1, inclusive.
- For each set $\mathcal{S}$, the symbol $|\mathcal{S}|$ denotes the cardinality of $\mathcal{S}$.
- For each vector $\mathbf{x} \in \mathbb{R}^n$, the symbols $\|\mathbf{x}\|_1$, $\|\mathbf{x}\|_2$, $\|\mathbf{x}\|_\infty$, and $\|\mathbf{x}\|_p$ denotes the L_1, L_2, L_∞, and L_p norms of $\mathbf{x}$, respectively.
- $\nabla f(\mathbf{x})$ denotes the gradient of function f when evaluated at the point $\mathbf{x}$.
- $\mathbb{E}_{\mathbf{x} \sim \mathbf{x}} f(\mathbf{x})$ represents the expected value of the function $f(\mathbf{x})$ computed over values of $\mathbf{x}$ drawn from the probability distribution $\mathbf{x}$.
- $\Pi_{\delta_{max}}(\mathbf{x}; \mathbf{o})$ where $\mathbf{x} \in \mathbb{R}^n$, and $\mathbf{o} \in \mathbb{R}^n$ denotes the projection operator that projects $\mathbf{x}$ on the hyper-sphere with radius δ_{max} centered at $\mathbf{o}$. This is achieved by clipping operation.

$$\mathbf{x}_j = \min\left(\max\left(\mathbf{x}_j, \mathbf{o}_j - \delta_{max}\right), \mathbf{o}_j + \delta_{max}\right) \quad \forall j = 1, ..., n$$

CHAPTER 2

GenAttack: Practical Black-box Attacks with Gradient-Free Optimization

In this chapter, we introduce *GenAttack*, an efficient method for generation of adversarial examples against black-box machine learning models.

2.1 Background

Deep neural networks (DNNs) have achieved state-of-the-art performance in various tasks in machine learning and artificial intelligence. Despite their effectiveness, recent studies have illustrated the vulnerability of DNNs to adversarial examples [SZS13, GSS14]. For instance, a virtually imperceptible perturbation to an image can lead a well-trained DNN to misclassify. Targeted adversarial examples can even cause misclassification to a chosen class. Moreover, researchers have shown that these adversarial examples are still effective in the physical world [KGB16a, AEI17], and can be crafted in distinct data modalities, such as in the natural language [ASE18], and speech [ABS17] domains. The lack of robustness exhibited by DNNs to adversarial examples has raised serious concerns for security-critical applications.

Nearly all previous work on adversarial attacks, [GSS14, CW17a, CSZ17, MFF16, GR14, KGB16a] has used gradient-based optimization in order to find successful adversarial examples. However, gradient computation can only be performed when the attacker has full knowledge of the model architecture and weights. Thus, these methods are only applicable in the *white-box* setting, where an attacker is given full access and control over a targeted DNN. However, when attacking real-world systems, one needs to consider the problem of performing adversarial attacks in the *black-box* setting, where nothing is revealed about the network architecture, parameters, or

training data. In such a case, the attacker only has access to the input-output pairs of the classifier. A popular approach in this setting have relied on attacking trained substitute networks, and hoping the generated examples transfer to the target model [PMG17]. This approach suffers from the inherent model mismatch between the substitute model to the target model, as well as the high computational cost required to train the substitute network. Recent works have used coordinate-based finite difference methods in order to directly estimate the gradients from the confidence scores, however the attacks are still computationally expensive, relying on optimization tricks to remain tractable [CZS17b]. Both approaches are query-intensive, thus limiting their practicality in real-world scenarios.

2.2 Contribution

Motivated by the above, we present *GenAttack*, a novel approach to generating adversarial examples without having to compute or even approximate the gradients, enabling efficient adversarial attacks to the black-box case. In order to perform *gradient-free optimization*, we adopt a population-based approach using genetic algorithms, iteratively evolving a population of feasible solutions until success. We also present a number of tricks that allows *GenAttack* to maintain its query-efficiency when attacking models trained on large-scale higher-dimensional datasets, such as ImageNet [DDS09].

Due to its gradient-free nature, *GenAttack* is robust to defenses which perform gradient masking or obfuscation [ACW18]. Thus, unlike many current black-box attack approaches, *GenAttack* can efficiently craft perturbations in the black-box setting to bypass some recently proposed defenses which manipulate the gradients.

We evaluate *GenAttack* using state-of-the-art image classification models, and find that the algorithm is successful at performing *targeted* black-box attacks with significantly less queries than current approaches. In our MNIST, CIFAR-10, and ImageNet experiments, *GenAttack* required roughly $\mathbf{2,126}$, $\mathbf{2,568}$, and $\mathbf{237}$ times less queries than the current state-of-the-art black-box attack, respectively. Additionally, we also demonstrate the success of *GenAttack* against state-of-the-art ImageNet defenses, such as ensemble adversarial training [TKP17], and randomized,

non-differentiable input transformation defenses [GRM17]. These results illustrate the power of *GenAttack*'s query efficiency and gradient-free nature.

In summary, we make the following contributions:

- We introduce *GenAttack*, a novel gradient-free approach for generating adversarial examples by leveraging population-based optimization. Our implementation is open-sourced[1] to promote further research in studying adversarial robustness.
- We show that in the restricted black-box setting, *GenAttack* using genetic optimization, as well as dimensionality reduction and adaptive parameter scaling, can generate adversarial examples which force state-of-the-art image classification models, trained on MNIST, CIFAR-10 and ImageNet, to misclassify examples to chosen target labels with significantly less queries than previous approaches.
- We further highlight the effectiveness of *GenAttack* by illustrating its success against a few state-of-the-art ImageNet defenses, namely ensemble adversarial training and randomized, non-differentiable input transformations. To the best of our knowledge, we are the first to present a successful black-box attack against these defenses.

The rest of this chapter is organized as follows: Section 2.3 provides a summary of related work. Section 2.4 formally defines the threat model for our attack. Section 2.5 discusses the details of *GenAttack*. Section 2.6 describes our evaluation experiments and their results. Finally, Section 2.7 concludes the chapter with discusses our main findings.

2.3 Related Work

In what follows, we summarize recent approaches for generating adversarial examples, in both the white-box and black-box cases, as well as defending against adversarial examples. Please refer to the cited works for further details.

[1] `https://github.com/nesl/adversarial_genattack`

2.3.1 White-box attacks & Transferability

In the *white-box* case, attackers have complete knowledge of and full access to the targeted DNN. In this scenario, the adversary is able to use backpropagation for gradient computation, which enables efficient gradient-based attacks. We briefly summarize a few important white-box attacks formulations below.

L-BFGS

[SZS13] used box-constrained L-BFGS to minimize the ℓ_2 norm of the added adversarial noise $||\delta||_2$ subject to $f(x+\delta) = l$ (prediction is class l) and $x + \delta \in [0,1]^m$ (input is within the valid pixel range), where $f : \mathbb{R}^m \to \{1, ..., k\}$ is the classifier, mapping a data example to a discrete label, $l \in \{1, ..., k\}$ is the target output label, and δ is the added noise.

FGSM & I-FGSM

[GSS14] proposed the Fast Gradient Sign Method (FGSM) to quickly generate adversarial examples. Under an L_∞ distortion constraint $\|\delta\|_\infty \leq \epsilon$, FGSM uses the sign of the gradient of the training loss J with respect to the original $\mathbf{x}_0$ and the true label l, to generate an adversarial example: $\mathbf{x} = \mathbf{x}_0 + \epsilon \cdot \text{sign}(\nabla J(\mathbf{x}_0, l))$, Similarly, targeted attacks can be implemented by computing the loss with respect to a specified target class t, and instead going in the direction of the negative gradient.

In [KGB16a], an iterative version of FGSM was proposed (I-FGSM), where FGSM is used iteratively with a finer distortion constraint, followed by an ϵ-ball clipping. In [MMS17a], project gradient descnet (PGD) is introduced, where I-FGSM is modified to incorporate random starts.

C&W & EAD

Instead of leveraging the training loss, C&W [CW17a] designed an L_2-regularized loss function based on the logit layer representation in DNNs for crafting adversarial examples. Handling the box constraint $\mathbf{x} \in [0,1]^p$ using a change of variables, they used Adam [KB14a] to minimize $c \cdot f(\mathbf{x}, t) + \|\mathbf{x} - \mathbf{x}_0\|_2^2$, where $f(x, t)$ is a loss function depending logit layer values and target

class t. EAD [CSZ17] generalizes the attack by minimizing an additional L_1 penalty, and has been shown to generate more robust and transferable adversarial examples [SC17, SC18, LCC18].

White-box attacks can also be used in black-box settings by taking advantage of *transferability* [LCL18]. Transferability refers to the property that adversarial examples generated using one model are often misclassified by another model [SZC18]. The substitute model approach to black-box attacks takes advantage of this property to generate successful adversarial examples, as we will discuss in the next subsection.

2.3.2 Black-box attacks

In the literature, the *black-box* attack setting has been referred to as the case where an attacker has free access to the input and output of a targeted DNN but is unable to perform back propagation on the network. Proposed approaches have relied on transferability and gradient estimation, and are summarized below.

Substitute Networks

Early approaches to black-box attacks made use of the power of free query to train a substitute model, a representative substitute of the targeted DNN [PMG17]. The substitute DNN can then be attacked using any white-box technique, and the generated adversarial examples are used to attack the target DNN. As the substitute model is trained to be representative of a targeted DNN in terms of its classification rules, adversarial examples of the substitute model are expected to be similar to adversarial examples of the corresponding targeted DNN. This approach, however, relies on the transferability property rather than directly attacking the target DNN, which is imperfect and thus limits the strength of the adversary. Furthermore, training a substitute model is computationally expensive and hardly feasible when attacking large models, such as Inception-v3 [SLJ15].

Zeroth Order Optimization (ZOO)

The zeroth order optimization (ZOO) attack builds on the C&W attack to perform black-box attacks [CZS17b], by modifying the loss function such that it only depends on the output of the DNN, and performing optimization with gradient estimates obtained via finite differences. ZOO, however, suffers from requiring a huge number of queries, since a gradient estimate requires 2 queries per each coordinate. Thus, attacking Inception-v3 [SLJ15] on the ImageNet dataset requires $299 \times 299 \times 3 \times 2 = 536,406$ queries per each optimization step. To resolve this issue, stochastic coordinate descent (SCD) is used, which only requires 2 queries per step. Still, convergence of SCD can be slow when the number of coordinates is large, thus reducing the dimensionality of the perturbation and using importance sampling are also crucial. Applying these techniques, unlike the substitute model approach, attacking Inception-v3 becomes computationally tractable. However, as we demonstrate in our experiments, the gradient estimation procedure is still quite query-inefficient, and thus impractical for attacking real-world systems.

In parallel works, [BRB18, IEA18, TTC18] have also studied the problem of efficiency and strength of black-box adversarial attacks, but our work remains unique in its goal and approach. [BRB18] focuses on attacking black-box models with only partial access to the query results. Notably, their method takes, on average, about 72x more queries than ours to achieve success against an undefended ImageNet model. [IEA18] and [TTC18] study the efficiency of black-box attacks under the same threat model we consider, however, both rely on *gradient estimation*, rather than *gradient-free optimization*. [IEA18] estimates the gradient of the expected value of the loss under a parameterized search distribution which can be seen as a finite differences estimate on a random gaussian basis. [TTC18] dispenses with ZOO's coordinate-wise estimation with a scaled random full gradient estimator. Though we treat both efforts as parallel work, for the sake of completeness, we provide a comparison between our "gradient-free" approach and the other query-efficient "gradient-estimation" approaches in our results.

2.3.3 Defending against adversarial attacks

Adversarial Training

Adversarial training is typically implemented by augmenting the original training dataset with the label-corrected adversarial examples to retrain the network. In [MMS17a], a high capacity network is trained against L_∞-constrained PGD, I-FGSM with random starts, yielding strong robustness in the L_∞ ball, but has been shown to be less robust to attacks optimized on other robustness metrics [SC17, SRB19, ZCS19]. In [TKP17], training data is augmented with perturbations transferred from other models, and was demonstrated to have strong robustness to transferred adversarial examples. We demonstrate in our experimental results that its less robust to query-efficient black-box attacks, such as *GenAttack*.

Gradient Obfuscation

It has been identified that many recently proposed defenses provide apparent robustness to strong adversarial attacks by manipulating the gradients to either be nonexistent or incorrect, dependent on test-time randomness, or simply unusable. Specifically, it was found in analyzing the ICLR 2018 non-certified defenses that claim white-box robustness, 7 of 9 relied on this phenomenon [ACW18]. It has also been shown that FGSM based adversarial training learns to succeed by making the gradients point in the wrong direction [TKP17].

One defense which relies upon gradient obfuscation is utilizing non-differentiable input transformations, such as bit-depth reduction, JPEG compression, and total variance minimization [GRM17]. In the *white-box* case, this defense can be successfully attacked with gradient-based techniques by replacing the non-differentiable transformation with the identity function on the backward pass [ACW18]. Though effective, this approach is not applicable in the *black-box* case, since the attacker requires knowledge of the non-differentiable component. We demonstrate in our experimental results that *GenAttack*, being gradient-free and thus impervious to said gradient manipulation, can naturally handle such procedures in the *black-box* case. Note that many black-box attacks that require gradient estimation including [CZS17b, IEA18, TTC18] cannot be directly applied when

non-differentiable input transformations exist.

2.4 Threat Model

We consider the following attack scenario. The attacker does not have knowledge about the network architecture, parameters, or training data. The attacker is solely capable of querying the target model as a black-box function:

$$f : \mathbb{R}^d \rightarrow [0,1]^K$$

where d is the number of input features and K is the number of classes. The output of function f is the set of model prediction scores. Note, that the attacker will *not* have access to intermediate values computed in the network hidden layers, including the logits.

The goal of the attacker is to perform a *targeted* attack. Formally speaking, given a benign input example $\mathbf{x}$ that is correctly classified by the model, the attacker seeks to find a perturbed example $\mathbf{x}_{adv}$ for which the network will produce the desired target prediction t chosen by the attacker from the set of labels $\{1..K\}$. Additionally, the attacker also seeks to minimize the L_p distance, in order to maintain the perceptual similarity between $\mathbf{x}_{orig}$ and $\mathbf{x}_{adv}$. i.e.,

$$\underset{c \in \{1..K\}}{\arg\max}\, f(\mathbf{x}_{adv})_c = t \quad \text{such that } ||\mathbf{x} - \mathbf{x}_{adv}||_p \leq \delta$$

where the distance norm function L_p is often chosen as L_2 or L_∞.

This threat model is equivalent to that of prior work in black-box attacks [CZS17b, PMG17], and is similar to the chosen-plain-text attack (CPA) in cryptography, where an attacker provides the victim with any chosen plain-text message and observes its encryption cipher output.

2.5 GenAttack Algorithm

GenAttack relies on genetic algorithms, which are population-based gradient-free optimization strategies. Genetic algorithms are inspired by the process of natural selection, iteratively evolving a population $\mathcal{P}$ of candidate solutions towards better solutions. The population in each iteration

is called a *generation*. In each generation, the quality of population members is evaluated using a *fitness* function. "Fitter" solutions are more likely to be selected for breeding the next generation. The next generation is generated through a combination of *crossover* and *mutation*. Crossover is the process of taking more than one parent solution and producing a child solution from them; it is analogous to reproduction and biological crossover. Mutation applies a small random perturbation to the population members, according to a small user-defined mutation probability. This is done in order to increase the diversity of population members and provide better exploration of the search space.

Algorithm 1 describes the operation of *GenAttack*. The algorithm input is the original example $\mathbf{x}_{orig}$ and the target classification label t chosen by the attacker. The algorithm computes an adversarial example $\mathbf{x}_{adv}$ such that the model classifies $\mathbf{x}_{adv}$ as t and $||\mathbf{x}_{orig} - \mathbf{x}_{adv}||_\infty \leq \delta_{max}$. We define the population size to be N, the *"mutation probability"* to be ρ, and the *"mutation range"* to be α.

GenAttack initializes a population of examples around the given input example $\mathbf{x}_{orig}$ by picking random examples from a uniform distributed defined over the sphere centered at the original example $\mathbf{x}_{orig}$, whose radius $= \delta_{max}$. This is achieved by adding random noise in the range $(-\delta_{max}, \delta_{max})$ to each dimension of the input vector $\mathbf{x}_{orig}$. Then repeatedly, until a successful example is found, each population members' fitness is evaluated, parents are selected, and `crossover` & `mutation` are performed to form the next generation.

Fitness function:

The subroutine `ComputeFitness` evaluates the fitness, i.e. quality, of each population member. As the fitness function should reflect the optimization objective, a reasonable choice would be to use the output score given to the target class label directly. However, we find it more efficient to also jointly motivate the decrease in the probability of other classes. We also find that the use of $\log$ proves to be helpful in avoiding numeric instability issues. Therefore, we chose the following function:

$$ComputeFitness(\mathbf{x}) = \log f(\mathbf{x})_t - \log \sum_{j=0, j\neq t}^{j=k} f(\mathbf{x})_c$$

Algorithm 1: Targeted Adversarial Attack using *GenAttack*

Input: original example $\mathbf{x}_{orig}$, target label t, maximum L_∞ distance δ_{max}, mutation-range α, mutation probability ρ, population size N, τ sampling temperature.

▷ ***Create initial generation***.

for $i = 1, ..., N$ in population **do**

$\mathcal{P}_i^0 \leftarrow \mathbf{x}_{orig} + \mathcal{U}(-\delta_{max}, \delta_{max})$

end for

for $g = 1, 2...G$ generations **do**

for $i = 1, ..., N$ in population **do**

$F_i^{g-1} = ComputeFitness(\mathcal{P}_i^{g-1})$

end for

$\mathbf{x}_{adv} = \mathcal{P}^{g-1}_{\arg\max_j F_j^{g-1}}$ ▷ ***Find the elite member.***

if $\arg\max_c f(\mathbf{x}_{adv})_c == t$ **then**

Return: $\mathbf{x}_{adv}$ ▷ ***Found successful attack***

end if

$\mathcal{P}_1^g = \{\mathbf{x}_{adv}\}$

▷ ***Compute Selection probabilities.***

$probs = Softmax(F^{g-1}/\tau)$

for $i = 2, ..., N$ in population **do**

Sample $parent_1$ from $\mathcal{P}^{g-1}$ according to $probs$

Sample $parent_2$ from $\mathcal{P}^{g-1}$ according to $probs$

$child = Crossover(parent_1, parent_2)$

$child_{mut}$ = $child + Bernoulli(\rho) * \hookrightarrow \mathcal{U}(-\alpha\,\delta_{max}, \alpha\,\delta_{max})$ ▷ ***Apply mutations and clipping.***

$child_{mut} = \Pi_{\delta_{max}}(child_{mut}, \mathbf{x}_{orig})$

▷ ***Add mutated child to next generation***.

$\mathcal{P}_i^g = \{child_{mut}\}$

end for

▷ ***adaptively update*** α***,*** ρ ***parameters***

ρ, α = UpdateParameters(ρ, α)

end for

Selection:

Population members at each iteration are ranked according to their fitness value. Members with higher fitness are more likely to be a part of the next generation while members with lower fitness are more likely to be replaced. We compute the probability of selection for each population member by computing the $\mathbf{Softmax}$ of the fitness values to turn them into a probability distribution. Then, we independently select random parent pairs among the population members according to that distribution. We also find it important to apply the elitism technique [BMP96], where the *elite* member, the one with highest fitness, of the current generation is guaranteed to become a member of the next generation.

Crossover operation:

After selection, parents are mated to produce members of the next generation. A child is generated by selecting each feature value from either $parent_1$ or $parent_2$ according to the selection probabilities $(p, 1-p)$ where p is defined as

$$p = \frac{fitness(parent\_1)}{fitness(parent\_1) + fitness(parent\_2)}$$

Mutation operation:

To promote diversity among the population members and exploration of the search space, at the end of each iteration, population members can be subject to mutation, according to probability ρ. Random noise uniformly sampled in the range $(-\alpha\,\delta_{\max}, \alpha\,\delta_{\max})$ is applied to individual features of the crossover operation results. Finally, clipping operator $\Pi_{\delta_{max}}$is performed to ensure that the pixel values are within the permissible L_∞ distance away from the benign example $\mathbf{x}_{orig}$.

2.5.1 Improving Query Efficiency

In this section, we present a couple of optimizations that we adopt in our *GenAttack* algorithm, and which contribute significantly to query efficiency.

2.5.1.1 Dimensionality Reduction:

On one hand, scaling genetic algorithms to explore high-dimensional search spaces (such as ImageNet models) efficiently requires a large population size in each generation. On the other hand, evaluating the fitness of each member implies additional costs in the form of new queries. Therefore, we limit *GenAttack* to operate using a relatively small (e.g., less than ten) population size. We provide more discussion on the trade-off between population size and number of queries in Section 2.6.3.

In addition, motivated by the work in [TTC18], we seek to address the challenge of scaling genetic algorithms (in turn *GenAttack*), by performing dimensionality reduction of the search space and defining adversarial noise in the lower dimensional space. To compute the adversarial example, we apply bilinear resizing (which is deterministic), to scale the noise up to same size as the input. Thus,

$$\mathbf{x}_{orig} \in [0,1]^d, \quad e_{adv} \in [0,1]^{d'}, \quad \mathbf{x}_{adv} = S(e_{adv}) + \mathbf{x}_{orig}$$

where e_{adv} is the learnt adversarial noise, S is the bilinear resizing operation, and d' is chosen such that $d' < d$. Effectively, this leads to a condensed adversarial noise vector, where one value of e_{adv} is used to perturb multiple neighbouring pixels of $\mathbf{x}_{orig}$ to produce $\mathbf{x}_{adv}$. We noticed that this approach significantly improves the query efficiency of *GenAttack* against high dimensional models, such as ImageNet models, while maintaining the attack success rate under the L_∞ constraint.

2.5.1.2 Adaptive Parameter Scaling

In order to lessen the sensitivity of genetic algorithms to hyperparameter values (e.g. mutation rate, population size, and mutation range), we use an annealing scheme where the algorithm parameters (namely the mutation rate ρ and mutation range α) are decreased gradually if the search algorithm is detected to be stuck for a number of sequential iterations without any further improvement in the objective function. Adaptive scaling alleviates the situation where a very high mutation rate may allow for an initially fast decrease in the objective function value, after which the algorithm may get stuck without achieving any further progress.

We employ exponential decay to update the parameter values

$$\rho = \max(\rho_{\text{min}}, 0.5 \times (0.9)^{\text{num\_plateaus}}) \tag{2.1}$$

$$\alpha = \max(\alpha_{\text{min}}, 0.4 \times (0.9)^{\text{num\_plateaus}}) \tag{2.2}$$

where ρ_{min} and α_{min} are chosen to be 0.1 and 0.15 respectively, and num_plateaus is a counter incremented whenever the algorithm does not improve the fitness of the population's elite member (highest fitness) for 100 consecutive steps.

2.6 Results

We evaluate *GenAttack* by running experiments attacking state-of-art MNIST, CIFAR-10, and ImageNet image classification models. For each dataset, we use the same models as used in the ZOO work [CZS17b]. For MNIST and CIFAR-10, the model accuracies are 99.5% and 80%, respectively. The reader can refer to [CW17a] for more details on the architecture of those models. For ImageNet, we use Inception-v3 [SLJ15], which achieves 94.4% top-5 accuracy and 78.8% top-1 accuracy. We compare the effectiveness of *GenAttack* to ZOO on these models in terms of the attack success rate (ASR), the runtime, and the median number of queries necessary for success. The runtime and query count statistics are computed over successful attacks only. A *single* query means an evaluation of the target model output on a *single* input image. Using the authors' code[2], we configure ZOO for each dataset based on the implementations the authors used for generating their experimental results [CZS17b]. We also evaluate against the state-of-the-art *white-box* C&W attack [CW17a], assuming direct access to the model, to give perspective on attack success rate.

In addition, we evaluate the effectiveness of *GenAttack* against ensemble adversarial training [TKP17], using models released by the authors at the following link[3]. Ensemble adversarial training is considered to be the state-of-art ImageNet defense against black-box attacks, proven to be effective at providing robustness against transferred attacks in hosted competitions [TKP17, KGB18, SLA18]. Finally, we evaluate against recently proposed randomized,

[2] `https://github.com/huanzhang12/ZOO-Attack`

[3] `https://github.com/tensorflow/models/tree/master/research/adv_imagenet_models`

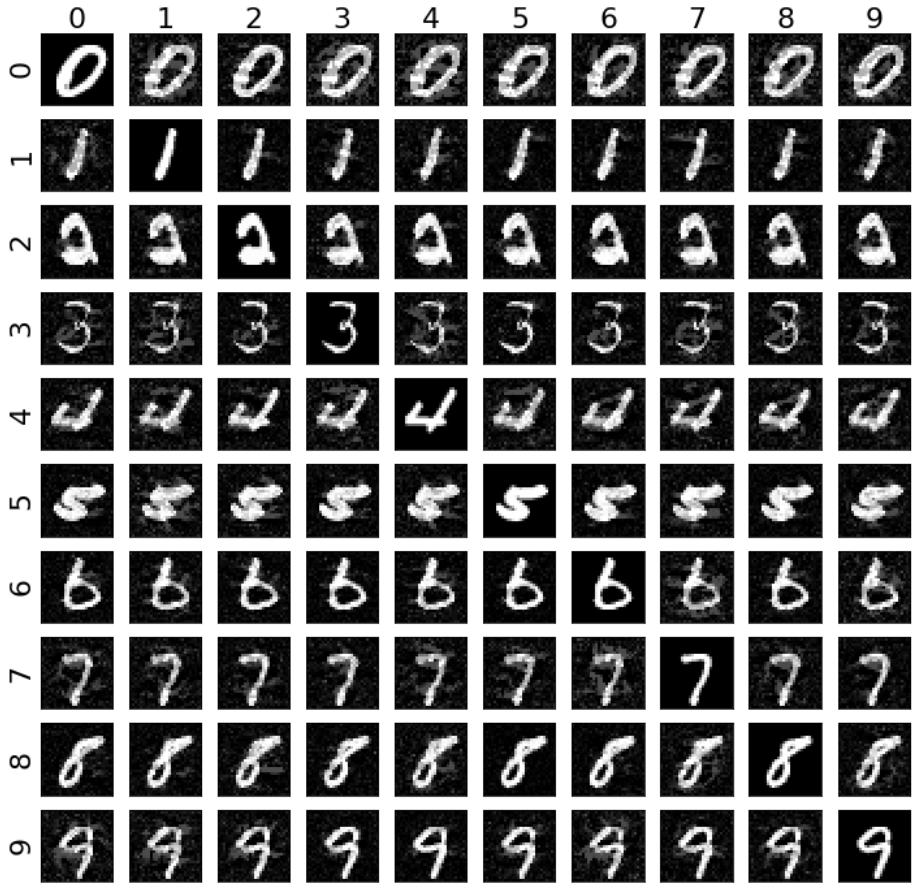


Figure 2.1: MNIST adversarial examples generated by *GenAttack*. Row label is the true label and column label is the target label.

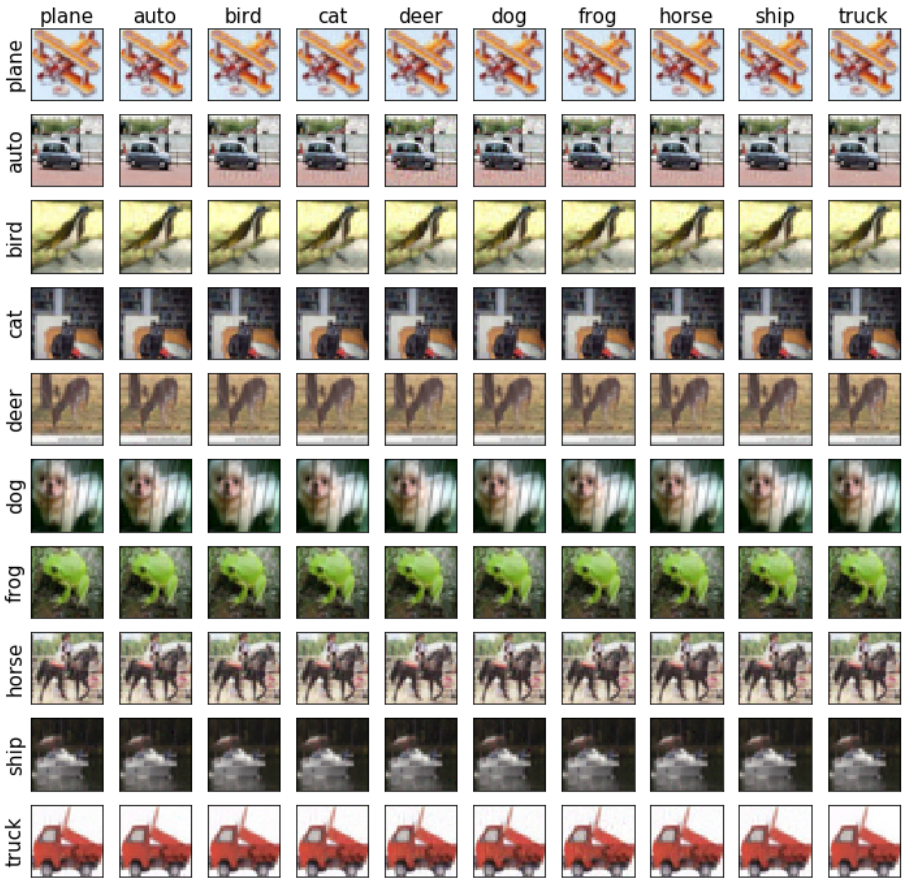


Figure 2.2: CIFAR-10 adversarial examples generated by *GenAttack*. Row label is the true label and column label is the target label.

non-differentiable input transformation defenses [GRM17] to test *GenAttack*'s performance against gradient obfuscation. We find that *GenAttack* can handle such defenses as-is due to its gradient-free nature.

Hyperparameters

For all of our MNIST and CIFAR-10 experiments, we limit *GenAttack* to a maximum of 100,000 queries, and fix the hyperparameters to the following values: mutation probability $\rho = 5\text{e}{-2}$, population size $N = 6$, and step-size $\alpha = 1.0$. For all of our ImageNet experiments, as the images are nearly 100x larger than those of CIFAR-10, we use a maximum of 1,000,000 queries and adaptively update the ρ and α parameters as discussed earlier in Section 2.5. In addition, we experimented both with and without dimensionality reduction ($d' = 96$). To match the mean L_∞ distortion computed over successful examples of ZOO, we set $\delta_{max} = \{0.3, 0.05, 0.05\}$, for our MNIST, CIFAR-10, and ImageNet experiments, respectively. As genetic algorithms have various tuning parameters, we conduct parameter sensitivity studies in Section 2.6.3. To encourage further work, our code is released as open source. [4]

2.6.1 Query Comparison

We compare *GenAttack* and ZOO by number of queries necessary to achieve success, and provide C&W white-box results to put the ASR in perspective. For MNIST, CIFAR-10, and ImageNet, we use 1000, 1000, and 100 randomly selected and correctly classified images from the test sets. For each image, we select a random target label.

2.6.1.1 MNIST and CIFAR-10:

Table 2.1 shows the results of our experiment. The results show that both ZOO and GenAttack can succeed on the MNIST and CIFAR-10 datasets, however *GenAttack* is 2,126 times and 2,568 times more efficient on each. A randomly selected set of MNIST and CIFAR-10 test images and their

[4] `https://github.com/nesl/adversarial_genattack.git`

	MNIST ($L_\infty = 0.30$)		CIFAR-10 ($L_\infty = 0.05$)	
	ASR	Queries	ASR	Queries
C&W	100%	–	100%	–
ZOO	98%	2,118,222	93.3%	2,064,798
GenAttack	100%	996	96.5%	804

Table 2.1: Attack success rate (ASR) and median number of queries for the C&W (white-box) attack, ZOO, and *GenAttack* with equivalent L_∞ distortion. Median of query counts is computed over successful examples. Number of queries is not a concern for white-box attacks.

associated adversarial examples targeted to each other label are shown in Figure 2.1 and Figure 2.2.

ImageNet:

Table 2.2 shows the results of our experiment against normally trained (InceptionV3) and ensemble adversarially trained (Ens4AdvInceptionV3) ImageNet models. To demonstrate the effect of dimensionality reduction and adaptive parameter scaling, we denote *GenAttack* without such tricks as "GA baseline". On ImageNet, ZOO is not able to succeed consistently in the targeted case, which is significant as it shows that *GenAttack* is efficient enough to effectively scale to ImageNet. Moreover, *GenAttack* is roughly 237 times more efficient than ZOO, and 9 times more query efficient than the GA baseline. A random example of results against Inception-v3 test image with its associated adversarial example is shown in Figure 2.3.

2.6.1.2 Comparison to parallel efforts in query efficient attacks

While preparing this manuscript, we became aware of parallel efforts that were also developed to address query-efficient adversarial attacks. For sake of completeness, we also present a comparison between our approach and other contributions. Unlike *GenAttack*, which performs gradient-free optimization, [TTC18] and [IEA18] propose more efficient gradient estimation procedures than ZOO. Table 2.3 shows a comparison between the results of the three methods. We notice that while

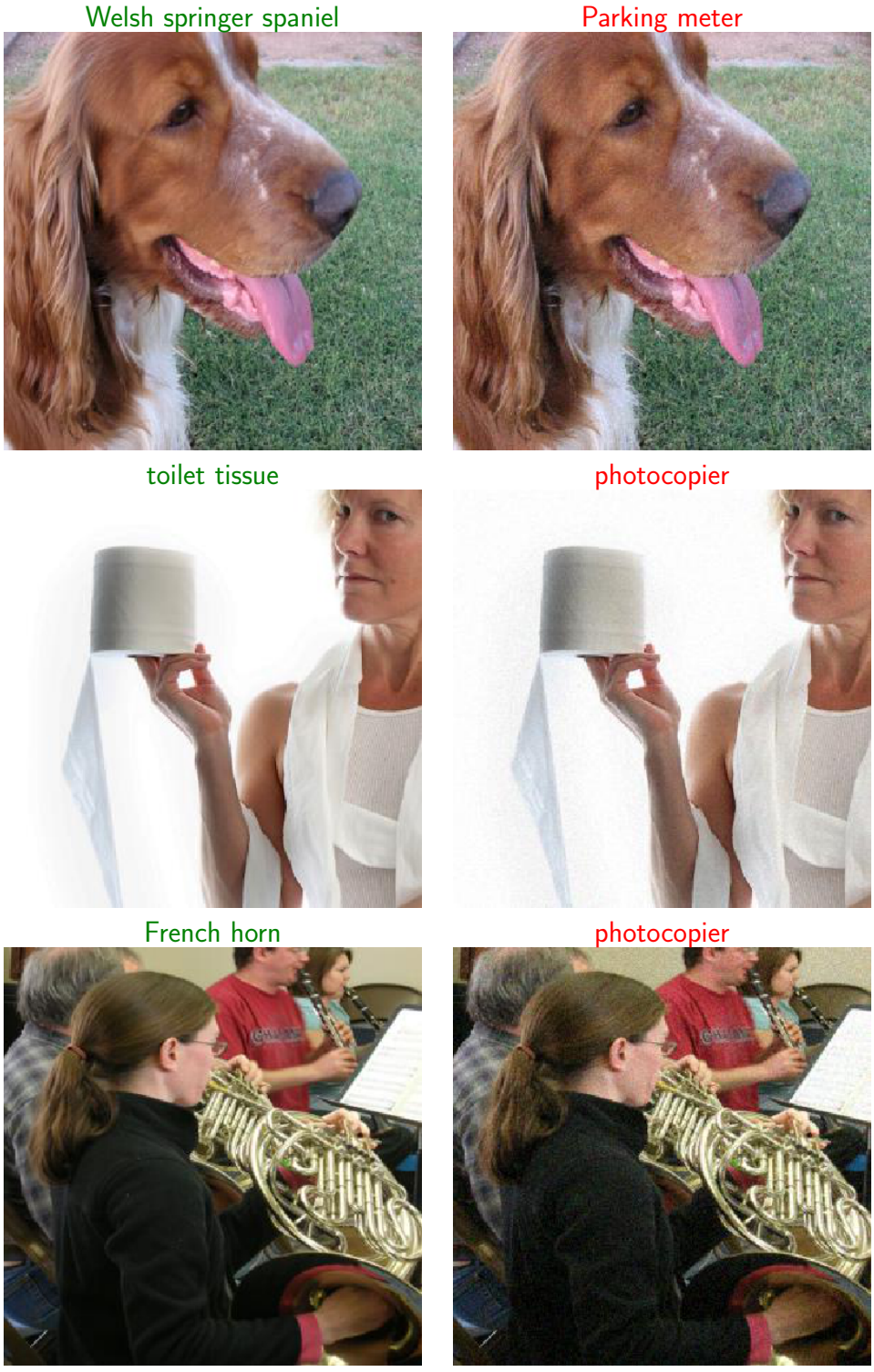


Figure 2.3: Random samples of adversarial examples generated by *GenAttack* against the `InceptionV3` model ($L_\infty = 0.05$). Left figure: original, right figure: adversarial example.

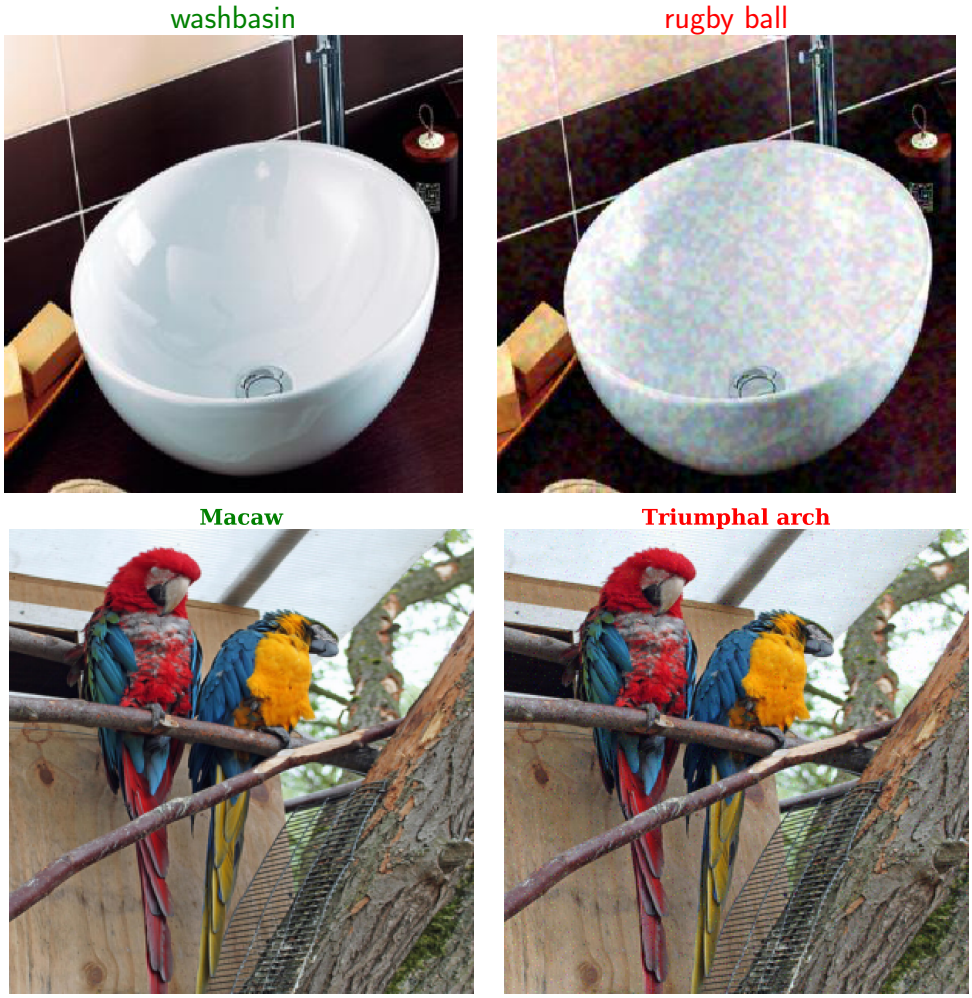


Figure 2.4: Adversarial example generated by *GenAttack* against the `Bit-depth Reduction` ($L_\infty = 0.05$) and `JPEG` compression defense ($L_\infty = 0.15$). Left figure: original, right figure: adversarial example.

	InceptionV3		Ens4AdvInceptionV3	
	ASR	Queries	ASR	Queries
C&W	100%	-	100%	-
ZOO	18%	2,611,456	6%	3,584,623
GA baseline	100%	97,493	93%	163,995
GenAttack	100%	11,081	95%	21,966

Table 2.2: Attack success rate (ASR) and median number of queries for the C&W (white-box) attack, ZOO, and *GenAttack* with equivalent L_∞ distortion (0.05). Median of query counts is computed over successful examples. GA baseline is *GenAttack* without the dimensionality reduction and adaptive parameter scaling tricks.

Attack	Queries Count	L_2-distance	L_∞-distance
GenAttack	**11,081**	2.3×10^{-4}	**0.05**
AutoZOOM [TTC18]	13,099	8.1×10^{-4}	0.75
Ilyas et al.[IEA18]	14,737	$\mathbf{1.9 \times 10^{-4}}$	**0.05**

Table 2.3: Comparison with parallel work against the ImageNet InceptionV3 model in terms of both median of query counts and per-pixel-L_2 and L_∞ distances between the original and adversarial images.

the three methods are all significantly more query efficient than the previous state-of-art (ZOO), *GenAttack* requires 25% fewer queries than the work of [IEA18] under the same L_∞ distance constraint at the cost of a slight increase in L_2 distance, mainly due to the use of dimensionality reduction. Also, *GenAttack* requires 15% less queries than [TTC18], even though [TTC18] has higher distortion in both L_∞ and L_2 distortion. Notably, [TTC18] has additional post-processing steps to reduce the amount of distortion but it significantly costs more queries. Therefore, we reported the number of queries and distortion distances at the first success for all attacks.

2.6.2 Attacking Defenses

In the following section, we present how *GenAttack* succeeds in breaking a set of state-of-the-art defense methods proposed to increase the robustness of models against adversarial attacks.

2.6.2.1 Attacking Ensemble Adversarial Training:

Ensemble adversarial training incorporates adversarial inputs generated on other already trained models into the model's training data in order to increase its robustness to adversarial examples [TKP17]. This has proven to be the most effective approach at providing robustness against transfer-based black-box attacks during the NIPS 2017 Competition. We demonstrate that the defense is much less robust against query-efficient black-box attacks, such as *GenAttack*.

We performed an experiment to evaluate the effectiveness of *GenAttack* against the ensemble adversarially trained model released by the authors [TKP17], namely `Ens4AdvInceptionV3`. We use the same 100 randomly sampled test images and targets used in our previous ImageNet experiments. We find that *GenAttack* is able to achieve 95% success against this strongly defended model, significantly outperforming ZOO. As seen, in Table 2.2, we compare the success rate and median query count between the ensemble adversarially trained and the vanilla Inception-v3 models. Our comparison shows that these positive results are yielded with only a limited increase in query count. We additionally note that the maximum L_∞ distortion used for evaluation in the NIPS 2017 competition varied between 4 and 16 (in a 0-255 scale), which when normalized equals 0.02 and 0.06, respectively. Our δ_{max} (0.05) falls in this range.

2.6.2.2 Attacking Non-Differentiable, Randomized Input Transformations:

Non-differentiable input transformations perform gradient obfuscation, relying upon manipulating the gradients to succeed against gradient-based attackers [ACW18]. In addition, randomized transformations increase the difficulty for the attacker to guarantee success. One can circumvent such approaches by modifying the core defense module performing the gradient obfuscation, however this is clearly not applicable in the black-box case.

	CIFAR-10		ImageNet	
	ASR	Queries	ASR	Queries
Bit depth	93%	2,796	95%	16,301
JPEG	88%	3,541	89%	23,822
TVM	70%	$5{,}888 \times 32$	–	–

Table 2.4: Evaluation of *GenAttack* against non-differentiable and randomized input transformation defenses. We use $L_\infty = 0.05$ for bit-depth, and $L_\infty = 0.15$ for JPEG and TVM experiments. For TVM, we compute the expectation of the fitness function by taking $t = 32$ queries.

In [GRM17], a number of input transformations were explored, including bit-depth reduction, JPEG compression, and total variance minimization. Bit-depth reduction and JPEG compression are non-differentiable, while total variance minimization introduces additional randomization and is quite slow, making it difficult to iterate upon. We demonstrate that *GenAttack* can succeed against these input transformations in the black-box case, due to its ***gradient-free*** and ***multi-modal*** population-based nature making it impervious to gradient obfuscation. To the best of our knowledge, we are the first to demonstrate a black-box algorithm which can bypass such defenses. Our results are summarized in Table 2.4.

For bit-depth reduction, 3 bits were reduced, while for JPEG compression, the quality level was set to 75, as in [GRM17]. *GenAttack* is able to achieve high success rate against both non-differentiable transformations, on both the CIFAR-10 and ImageNet datasets. A visual example of our results against JPEG compression is shown in Figure 2.4.

Total variance minimization (TVM) introduces an additional challenge as it is not only non-differentiable, but it also introduces randomization and is an exceedingly slow procedure. TVM randomly drops many of the pixels (dropout rate of 50%, as in [GRM17]) in the original image and reconstructs the input image from the remaining pixels by solving a denoising optimization problem. Due to randomization, the classifier returns a different score at each run for the same input, confusing the attacker. Succeeding against randomization requires more iterations, but iterating upon the defense is difficult due to the slow speed of TVM processing.

In the setting with randomization, the `ComputeFitness` function can be generalized to be

$$ComputeFitness(\mathbf{x}) = \mathop{\mathbb{E}}_{r}[\log f(\mathbf{x}, r)_t - \log \max_{c \neq t} f(\mathbf{x}, r)_c]$$

where $f(\mathbf{x}, r)$ is the randomization-defended model query function and r is the noise input to the TVM function, *GenAttack* can still handle this defense in the black-box case. The expectation is computed by querying the model t times (we used $t = 32$) for every population member to obtain a robust fitness score at the cost of an increased number of queries. Due to the computational complexity of applying TVM on each query, we performed the TVM experiment only using the CIFAR-10 dataset and achieved 70% success with L_∞ = 0.15. Due to the large randomization introduced by TVM, we counted an adversarial example as success only if it is classified as the target label three times in a row. Notably, TVM significantly decreases the model accuracy on clean inputs (e.g. in our CIFAR-10 experiments, from 80% to 40%) unless the model is re-trained with transformed examples [GRM17].

Comparison to ZOO and C&W:

Due to the non-differentiable nature of the input transformations, the C&W attack, a gradient-based attack, can not succeed without manipulating the non-differentiable component, as discussed in [ACW18]. In the *white-box* case, this method can be applied to yield high success rate, but is not applicable in the more restricted *black-box* case. In the black-box setting, ZOO achieved 8% and 0% against the non-differentiable bit-depth reduction and JPEG compression defenses on ImageNet, again demonstrating its impracticality.

2.6.3 Hyper-parameters values selection

Since genetic algorithms are traditionally sensitive to the choice of hyper-parameter values (e.g. population, mutation rate, etc.), we present a discussion regarding this effect, in the context of query efficiency, which leads to our selection of the hyper-parameter values listed in Section 2.6.

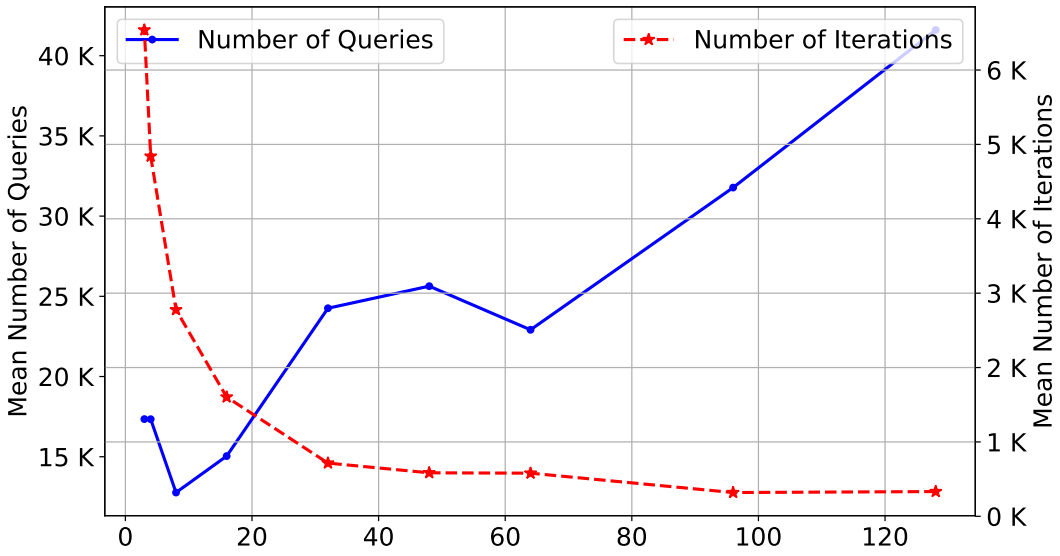


Figure 2.5: Effect of population size selection on both the speed of convergence and the number of queries.

Population size:

Large population size allows for increased population diversity and thus improved exploration of the search space within fewer iterations. However, since the evaluation of each population member costs one query, there is a trade-off in the selection of large population size to accelerate the algorithm success (in terms of the number of iterations), and the total number of queries spent. Figure 2.5 demonstrates this trade-off. On a set of 20 images, we recorded the mean number of queries as well as the number of iterations until success under different choices of population size. From this experiment, we conclude that the relatively small population size of six is a reasonable choice to balance between convergence speed and number of queries.

Mutation rate:

For the mutation rate ρ, we found that the best result can be achieved if we use an adaptive mutation rate which is gradually decayed according to Eq. (2.1) and (2.2) in Section 2.5. As shown in

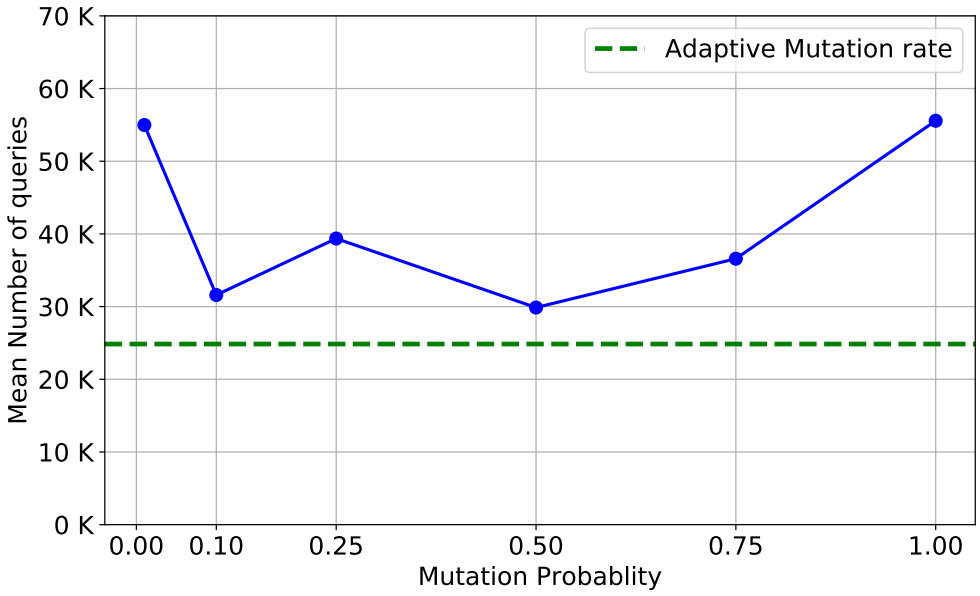


Figure 2.6: Effect of mutation probability on the number of queries required until success.

Figure 2.6, this method performs better than fixed valued mutation rates. Effectively, the adaptive mutation rate balances between exploration and exploitation by encouraging exploration initially, and then gradually increasing the exploitation rate as the algorithm approaches convergence near an optimal solution.

2.7 Conclusion and Future Work

In this chapter, we presented *GenAttack*, a powerful and efficient black-box attack which uses a gradient-free optimization scheme via adopting a population-based approach through genetic algorithms. We evaluated *GenAttack* by attacking well-trained MNIST, CIFAR-10, and ImageNet models, and found that *GenAttack* is successful at performing targeted black-box attacks against these models with not only significantly less queries than the previous state-of-the-art, but additionally can achieve a high success rate on ImageNet, which previous approaches are incapable of scaling to. Furthermore, we demonstrate that *GenAttack* can succeed against ensemble adversarial training, the state-of-the-art ImageNet defense, with only a limited increase in the number of queries.

Finally, we showed that *GenAttack* can succeed against gradient obfuscation, due to its gradient-free nature, namely through evaluating against non-differentiable input transformations, and can even succeed against randomized ones by generalizing the fitness function to compute an expectation over the transformation. To the best of our knowledge, this is the first demonstration of a black-box attack which can succeed against these state-of-the-art defenses. Our results suggest that population-based optimization opens up a promising research direction into effective gradient-free black-box attacks.

Future work: Our directions of the future work include studying how to efficiently generate adversarial examples when the *black-box* classifier returns only the *hard-label* prediction without probability scores. Also, we will investigate ideas for defenses against adversarial attacks. Also, we would like to consider the problem of how to

CHAPTER 3

Adversarial Attacks against Speech Recognition Systems

In this Chapter, we demonstrate how the *GenAttack* algorithm we introduced in Chapter 2 can be used to attack models for *automatic speech recognition*.

3.1 Background

Recent progress in machine learning and artificial intelligence is shaping the way we interact with our everyday devices. Speech based interaction is one of the most effective means and is widely used in personal assistants of smartphones (e.g. Siri, Google Assistant). These systems rely on running speech classification model to recognize the user's voice commands. Although traditional speech recognition models were based on hidden markov models (HMMs), deep learning models are currently the state of art for automatic speech recognition (ASR) [GMH13], [AAA16] and speech generation [ODZ16]. Despite their outstanding performance accuracies in many applications, recent research has shown that neural networks are easily fooled by malicious attackers who can force the model to produce wrong result or to even generate a targeted output value. This kind of attack known as adversarial examples has been demonstrated with high success against image recognition, and object detection models. However, to the best of our knowledge there have been no successful equivalent attacks against automatic speech recognition (ASR) models.

3.2 Contribution

In this Chapter, we present an attack approach that fools neural-network-based speech recognition model. Similar to adversarial example generation for images, the attacker will perturb benign

(*correctly classified*) audio files by adding a small amount of noise to cause the ASR model to mis-classify or produce a specific target output label. The added noise is small and will be observed by a human listening to the attack audio clip as background noise and will not change how a human recognizes the audio file. However, it will be sufficient to change the model prediction from the true label to another target label chosen by the attacker.

Existing methods for adversarial examples generation such as FGSM [GSS14], Jacobian-based Saliency Map Attack [PMJ16], DeepFool [MFF16], and Carlini [CW17a] depend on computing the gradient of some output of the network with respect to its input in order to compute the attack noise. For example, in the FGSM [GSS14] the adversarial noise is computed as:

$$\mathbf{x}_{adv} = \mathbf{x} + \epsilon \, sign(\nabla_{\mathbf{x}} J(\theta, \mathbf{x}, y))$$

The gradient needed to compute adversarial noise can be efficiently computed using backpropagation *assuming attacker knows model architecture and parameters*. However, backpropagation, *being based on the chain rule*, requires the ability to compute the derivative of each network layer output with respect to the layer inputs. While it is easily done in image recognition models where all layers in the pipeline are differentiable, it becomes problematic to apply same techniques for speech recognition models as they rely on the Mel Frequency Cepstral Coefficients (MFCCs) as features of the input audio data. Therefore, the first layers of an ASR model typically pre-process the raw audio by computing the spectrogram and the MFCC inputs. These two layers are not differentiable and there is no efficient way to compute the gradient through them. While the training process of the neural network does not require backpropagation because MFCCs are considered as model inputs, the generation of adversarial examples would require the gradient. Therefore, gradient-based methods [GSS14, PMJ16, CW17a, MFF16]) to generate adversarial noise are not directly applicable to speech recognition models based on MFCCs.

Our algorithm generates adversarial noise to perform targeted attacks on ASR. To avoid computing MFCC derivatives, we use a genetic algorithm which is a gradient-free optimization method. Our genetic algorithm based method does not require knowledge of the victim model architecture or parameters and can be used for black box attacks without training substitutive models. We evaluate our attack using the speech commands recognition model [SP15] and the speech commands

dataset [spe]. Our results show that targeted attacks succeed 87% of the time while adding noise to only the 8 least-significant-bits of a subset of samples in a 16 bits-per-sample audio file. We evaluate the effect of noise on human perception of the audio clip with a user study. Results show that the noise did not change the human decision in 89% of our samples and listeners still recognize the audio as its original label.

Adversarial Attacks on Audio:

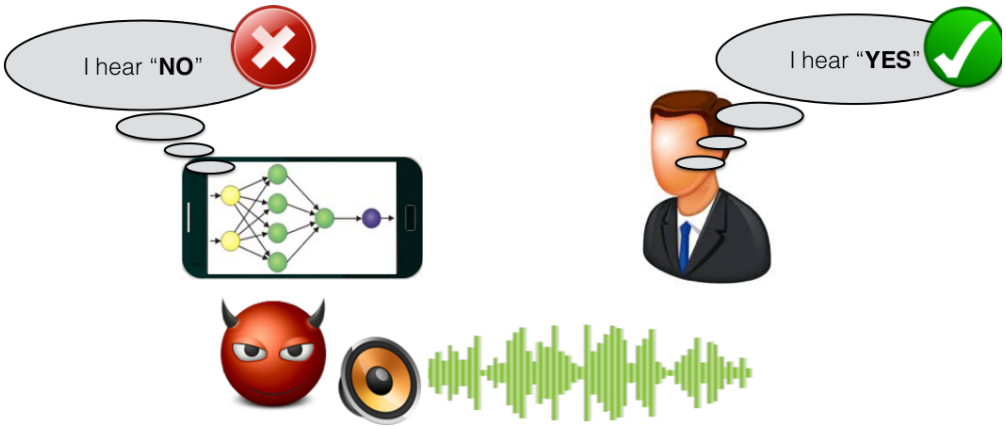


Figure 3.1: Adversarial attacks on speech commands: a malicious attacker adds small noise to the audio such that it is misclassified by the speech recognition model but does not change human perception.

Adversarial examples refer to inputs that are maliciously crafted by an attacker to fool machine learning models. Adversarial examples are typically generated by adding noise to the inputs that are correctly classified by the model, and the added noise should be imperceptible for humans. To create adversarial examples for speech recognition models an attacker takes a *legitimate* audio file perturbs it by adding an *imperceptible* noise that causes the machine learning speech recognition model to mis-classify the input *and possibly produce a desired target label*. We demonstrate this in Figure 3.1, where the attacker adds noise to an audio clip of the word "YES" that the machine learning model classifies as "NO" while the human still recognizes as "YES".

Prior Audio Attacks: While recent research uncovered potential attacks against speech recognition models, the demonstrated attacks do not represent an instantiation of adversarial examples [GSS14] as witnessed with image recognition models. Backdoor [RHR17] exploits the

non-linearities of microphones in smart devices to play audio at a frequency that is inaudible to humans (40 kHz), but creates a shadow in the audible range of the microphone. Backdoor harnesses this phenomenon to block microphone in places such as movie theaters. However, the attack requires an array of specialized high frequency speakers. DolphinAttack [SM17] exploits the same non-linearities in microphones to create commands audible to speech assistants but inaudible to humans. Notably, in both methods [RHR17, SM17] the attack sound is not heard by the human at all, while an adversarial example should be recognized by a human as benign while misclassified by the speech recognition model. The attack to closest adversarial examples is the "Hidden Voice Commands" by Carlini et al. [CMV16] that generates sounds that are unintelligible to human listeners but interpreted as commands by devices. Nevertheless, it does not represent an adversarial attack because the samples they generate are aimed to be 'unrecognizable' by humans, but it can still lead to suspicion. A more stealthy and powerful attack will maintain the listener interpretation of the attack samples as something benign.

3.3 Threat Model:

Following the same threat model as the one adopted in Chapter 2, a black-box threat model where the attacker knows nothing about the model architecture and parameter values, but is capable of querying the model results. Precisely, the victim model is used by the attacker as a black box function $f(\mathbf{x})$ while mounting his attacks. Such that: $f : \mathbb{X} \longrightarrow [0,1]^K$ where X is the space of all possible input audio files, and the output $[0,1]^K$ represent the prediction probability scores to each one of the possible K output labels. The output values are obtained from the final `Softmax` layer commonly used in classification models.

3.4 Generating Adversarial Speech Commands

We use gradient free genetic algorithm based approach to generate our adversarial examples as shown in 3. The algorithm accepts an original benign audio clip $\mathbf{x}$ and a target label t as its inputs. It creates a population of candidate adversarial examples by adding random noise to a subset of the

Algorithm 2: Generation of Targeted Adversarial Audio Files using *GenAttack*

Inputs : Original benign example $\mathbf{x}$

target classification label t

Output : Targeted attack example $\mathbf{x}_{adv}$

`/* Initialize the population of candidate solutions */`

population $\longleftarrow$ `InitializePopulation(`$\mathbf{x}$`)`

iter_num = 0

while $iter\_num < max\_iter$ **do**

scores $\longleftarrow$ `ComputeFitness(`population`)`

$\mathbf{x}_{adv}$ $\longleftarrow$ population $[argmax(scores)]$

if *argmax* $f(\mathbf{x}_{adv}) = t$ **then**

break `// Attack succeeded, Stop early.`

end

`/* Compute selection probabilities. */`

$select\_probs \longleftarrow Softmax(\frac{scores}{Temp})$

Next population $\longleftarrow$ { }

for $i \leftarrow 1$ **to** $size$ **do**

Select $parent_1$ from population according to probabilities $select\_probs$

Select $parent_2$ from population according to probabilities $select\_probs$

child = `Crossover(`$parent_1$, $parent_2$`)`

Next population = Next population $\bigcup\{child\}$

end

foreach *child of* Next population **do** `Mutate(`$child$`)`

population $\longleftarrow$ Next population

$iter\_num = iter\_num + 1$

end

return $\mathbf{x}_{adv}$

samples within the given audio clip. To minimize the noise effect on human perception, we add noise to only least-significant bits of a random subset of audio samples. We compute fitness score to each population member based on the prediction score of the target label and produce the next generation of adversarial examples from the current generation by applying selection, crossover and mutation. Selection means that population members with higher fitness value are more likely to become part of the next generation. Crossover takes pairs of population members and mixes them to generate a new 'child' that will be added to the new population. Finally, mutation adds random noise with very small probability to the child before passing it to the future generation. The algorithm iterates on this process for preset number of epochs or until the attack is found successful.

To assist other researchers to reproduce our results, we have made our implementation (with the same hyper-parameter values used for evaluation results reported in this chapter) available at `https://git.io/vFs8X`.

3.5 Evaluation

Speech Recognition Model: We evaluate our attack against the Speech Commands classification model [SP15] implemented in the TensorFlow [AAB16] software framework. This model is an efficient and light-weight keyword spotting model based on convolutional neural network and achieves 90% classification accuracy on the speech commands [spe] dataset. The speech commands dataset [spe] is a crowd-sourced dataset consisting of 65,000 audio files. Each file is a one second audio clip of single words like: "yes", "no", "up", "down", "left", "right", "on", "off", "stop", or "go".

3.5.1 Attack Results:

For the targeted attack experiment, we randomly select 500 audio clips from the dataset at 50 clips per labels (after we exclude the "silence" and "unknown" labels). We produce adversarial examples from each file such that it will be classified as a different target label. For example, for an audio clip of "yes", we produce adversarial examples that are targeted to be classified as "no", "up",

"down", "left", etc. This means for input audio clip we produce 9 adversarial examples leading to a total count of 4500 output files. Samples of our targeted attack output can be listened to at `https://git.io/vFs42`. Figure 3.2 shows the result of our targeted attack. Our algorithm

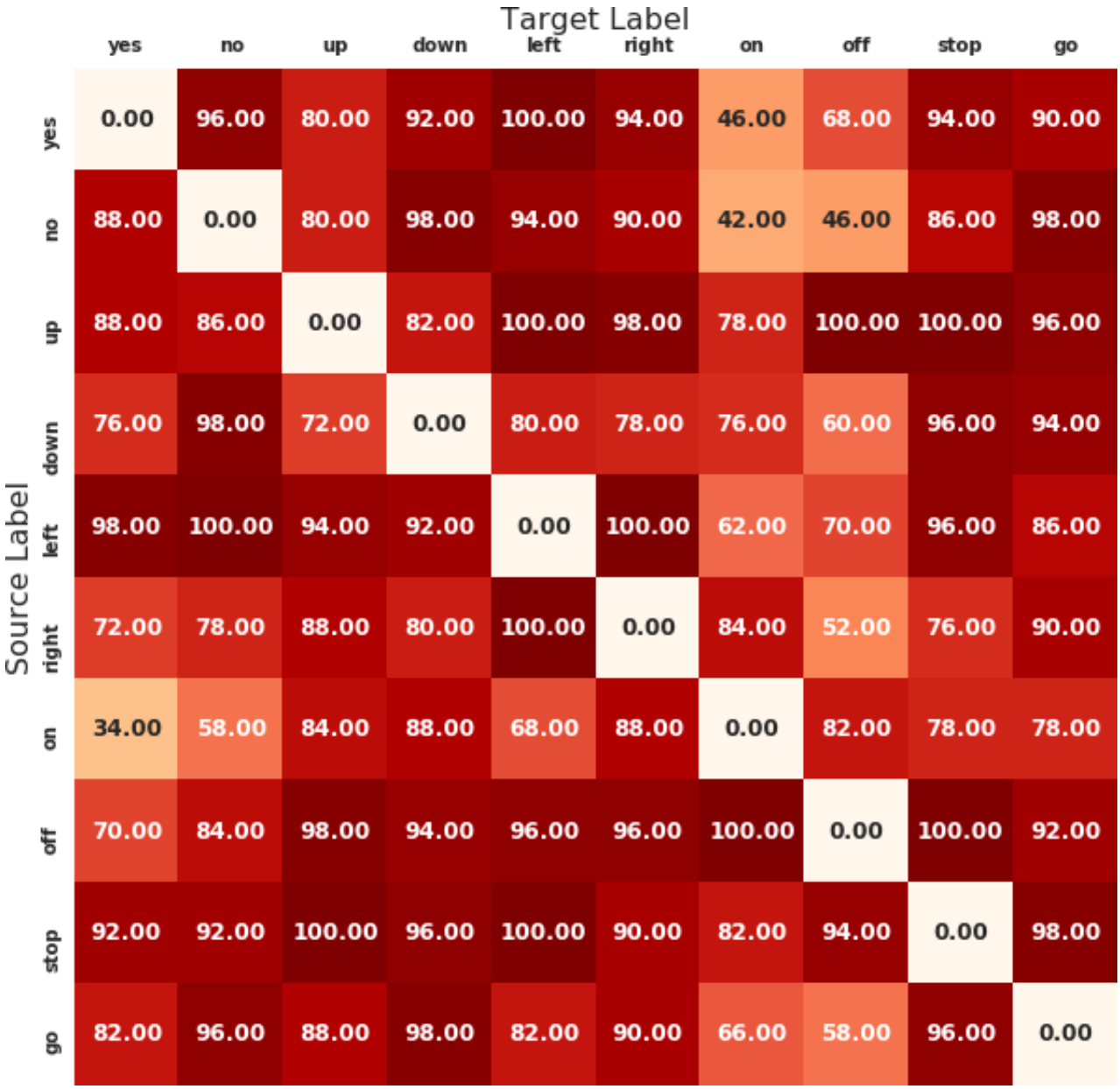


Figure 3.2: Percentage of success for every (source, target) targeted adversarial attack.

was successful 87% in performing targeted adversarial attack between any source-target pair. We limit the number of iterations in our algorithm to 500. If the algorithm fails to find a successful targeted attack within 500 iterations, we declare this as failure. The median time to generate an

adversarial audio file is 37 seconds on a desktop machine with Nvidia Titan X GPU. A more successful attack can be possible if we increase the limit of noise or number of iterations.

Attack labeled as source	Attack labeled as target	Attack labeled as other
89%	0.6%	9.4%

Table 3.1: Human perception of adversarial examples generated by our attack

3.5.2 Human Perception Results:

In order to assess the effect of added adversarial noise on human listeners, we conducted a human study where we recruited 23 participants, and we asked them to listen to and label successful adversarial audio clips we generated. In total, the study participants labeled 1500 audio clips. The participants were not told what is the source or target labels of the audio clips they were provided.

Results from our human experiment shown in Table 3.1 show that 89% of participants were not affected by the added noise and they still label the heard audio at the source label while the machine learning model labels all of them as the target label.

3.6 Conclusion and Future Work

In this section, we discuss the limitations and possible future directions for our study.

Using MFCC inversion for white box attack: Our attack algorithm does not require knowing the model architecture or its parameters and it only uses the victim model as a black box. In a white-box scenario where an attacker can utilize his knowledge about victim model, a stronger attack may be possible. However, this approach will face the hurdle of how to do back-propagation through the MFCC and spectrogram layer. One idea is to compute the adversarial noise with respect to the MFCC layer outputs as the classification model inputs, then use MFCC inversion [BD08] to reconstruct the adversarial audio. Further experiments should be done to evaluate the quality of this approach.

Evaluation against a larger ASR model and complete sentence generation: An interesting

question is if the more powerful state-of-art ASR models are also affected by adversarial examples, and whether we can generate adversarial sentences instead of just adversarial audio clips of single words.

Untargeted attacks: We reported the results of our targeted attacks where the attacker specifies the desired output label. In addition, we achieved 100% success rate with our untargeted attacks. Although the untargeted attack is considered a weaker type of attack, further study of the untargeted attacks can be useful to study model robustness against adversarial noise.

Over the air attack: In our evaluation, we assume that the attacker feeds the audio clip directly to the classification model. However, a more realistic and powerful attack will succeed even when we play the adversarial audio clip from the speaker while the victim model picks the audio from the microphone. This is harder to achieve, and we plan to study it in follow-up research.

CHAPTER 4

Generating Natural Language Adversarial Examples

In this Chapter, we demonstrate how the *GenAttack* algorithm can be modified to attack models for *natural language understanding*.

4.1 Background

Adversarial examples have been explored primarily in the image recognition domain. Examples have been generated through solving an optimization problem, attempting to induce misclassification while minimizing the perceptual distortion [SZS13, CW17a, CSZ17, SC17]. Due to the computational cost of such approaches, fast methods were introduced which, either in one-step or iteratively, shift all pixels simultaneously until a distortion constraint is reached [GSS14, KGB16b, MMS17b]. Nearly all popular methods are gradient-based.

Such methods, however, rely on the fact that adding small perturbations to many pixels in the image will not have a noticeable effect on a human viewer. This approach obviously does not transfer to the natural language domain, as all changes are perceptible. Furthermore, unlike continuous image pixel values, words in a sentence are discrete tokens. Therefore, it is not possible to compute the gradient of the network loss function with respect to the input words. A straightforward workaround is to project input sentences into a continuous space (e.g. word embeddings) and consider this as the model input. However, this approach also fails because it still assumes that replacing *every* word with words nearby in the embedding space will not be noticeable. Replacing words without accounting for syntactic coherence will certainly lead to improperly constructed sentences which will look odd to the reader.

Relative to the image domain, little work has been pursued for generating natural language adver-

sarial examples. Given the difficulty in generating semantics-preserving perturbations, distracting sentences have been added to the input document in order to induce misclassification [JL17].

4.2 Contribution

In our chapter, we attempt to generate semantically and syntactically similar adversarial examples, via word replacements, resolving the aforementioned issues. Minimizing the number of word replacements necessary to induce misclassification has been studied in previous work [PMS16], however without consideration given to semantics or syntactics, yielding incoherent generated examples.

In recent work, there have been a few attempts at generating adversarial examples for language tasks by using back-translation [IWG18], exploiting machine-generated rules [RSG18], and searching in underlying semantic space [ZDS18]. In addition, while preparing our submission, we became aware of recent work which target a similar contribution [KTL18, ERL18]. We treat these contributions as parallel work.

4.3 Attack Design

4.3.1 Threat model

We assume the attacker has black-box access to the target model; the attacker is not aware of the model architecture, parameters, or training data, and is only capable of querying the target model with supplied inputs and obtaining the output predictions and their confidence scores. This setting has been extensively studied in the image domain [PMG16, CZS17a, ASC18], but has yet to be explored in the context of natural language.

4.3.2 Algorithm

To avoid the limitations of gradient-based attack methods, we design an algorithm for constructing adversarial examples with the following goals in mind. We aim to minimize the number of modified

words between the original and adversarial examples, but only perform modifications which retain semantic similarity with the original and syntactic coherence. To achieve these goals, instead of relying on gradient-based optimization, we developed an attack algorithm that exploits population-based gradient-free optimization via genetic algorithms.

An added benefit of using gradient-free optimization is enabling use in the black-box case; gradient-reliant algorithms are inapplicable in this case, as they are dependent on the model being differentiable and the internals being accessible [PMS16, ERL18].

Genetic algorithms are inspired by the process of natural selection, iteratively evolving a population of candidate solutions towards better solutions. The population of each iteration is a called a *generation*. In each generation, the quality of population members is evaluated using a *fitness* function. "Fitter" solutions are more likely to be selected for breeding the next generation. The next generation is generated through a combination of *crossover* and *mutation*. Crossover is the process of taking more than one parent solution and producing a child solution from them; it is analogous to reproduction and biological crossover. Mutation is done in order to increase the diversity of population members and provide better exploration of the search space. Genetic algorithms are known to perform well in solving combinatorial optimization problems [AF94, Muh89], and due to employing a population of candidate solutions, these algorithms can find successful adversarial examples with fewer modifications.

`Perturb` Subroutine: In order to explain our algorithm, we first introduce the subroutine `Perturb`. This subroutine accepts an input sentence $\mathbf{x}_{cur}$ which can be either a modified sentence or the same as $\mathbf{x}_{orig}$. It randomly selects a word w in the sentence $\mathbf{x}_{cur}$ and then selects a suitable replacement word that has similar semantic meaning, fits within the surrounding context, and increases the $target$ label prediction score.

In order to select the best replacement word, `Perturb` applies the following steps:

- Computes the N nearest neighbors of the selected word according to the distance in the GloVe embedding space [PSM14]. We used euclidean distance, as we did not see noticeable improvement using cosine. We filter out candidates with distance to the selected word greater than δ. We use the counter-fitting method presented in [MST16] to post-process the

adversary's GloVe vectors to ensure that the nearest neighbors are synonyms. The resulting embedding is independent of the embeddings used by victim models.

- Second, we use the Google 1 billion words language model [CMS13] to filter out words that do not fit within the context surrounding the word w in $\mathbf{x}_{cur}$. We do so by ranking the candidate words based on their language model scores when fit within the replacement context, and keeping only the top K words with the highest scores.
- From the remaining set of words, we pick the one that will maximize the target label prediction probability when it replaces the word w in $\mathbf{x}_{cur}$.
- Finally, the selected word is inserted in place of w, and `Perturb` returns the resulting sentence.

The selection of which word to replace in the input sentence is done by random sampling with probabilities proportional to the number of neighbors each word has within Euclidean distance δ in the counter-fitted embedding space, encouraging the solution set to be large enough for the algorithm to make appropriate modifications. We exclude common articles and prepositions (e.g. a, to) from being selected for replacement.

Optimization Procedure: The optimization algorithm can be seen in Algorithm 3. The algorithm starts by creating the initial generation $\mathcal{P}^0$ of size S by calling the `Perturb` subroutine S times to create a set of distinct modifications to the original sentence. Then, the fitness of each population member in the current generation is computed as the target label prediction probability, found by querying the victim model function f. If a population member's predicted label is equal to the target label, the optimization is complete. Otherwise, pairs of population members from the current generation are randomly sampled with probability proportional to their fitness values. A new *child* sentence is then synthesized from a pair of parent sentences by independently sampling from the two using a uniform distribution. Finally, the `Perturb` subroutine is applied to the resulting children.

Algorithm 3: Algorithm for Finding Natural Language Adversarial Examples

for $i = 1, ..., S$ in population **do**

$\mathcal{P}_i^0 \leftarrow \texttt{Perturb}(\mathbf{x}_{orig}, target)$

end for

for $g = 1, 2...G$ generations **do**

for $i = 1, ..., S$ in population **do**

$F_i^{g-1} = f(\mathcal{P}_i^{g-1})_{target}$

end for

$\mathbf{x}_{adv} = \mathcal{P}^{g-1}_{\arg\max_j F_j^{g-1}}$

if $\arg\max_c f(\mathbf{x}_{adv})_c == t$ **then**

return $\mathbf{x}_{adv}$ ▷ ***Found successful attack***

else

$\mathcal{P}_1^g = \{\mathbf{x}_{adv}\}$

$p = Normalize(F^{g-1})$

for $i = 2, ..., S$ in population **do**

Sample $parent_1$ from $\mathcal{P}^{g-1}$ with probs p

Sample $parent_2$ from $\mathcal{P}^{g-1}$ with probs p

$child = Crossover(parent_1, parent_2)$

$child_{mut} = \texttt{Perturb}(child, target)$

$\mathcal{P}_i^g = \{child_{mut}\}$

end for

end if

end for

4.4 Experiments

To evaluate our attack method, we trained models for the sentiment analysis and textual entailment classification tasks. For both models, each word in the input sentence is first projected into a fixed 300-dimensional vector space using GloVe [PSM14]. Each of the models used are based on popular

Original Text Prediction = **Negative**. (Confidence = 78.0%)
*This movie had **terrible** acting, **terrible** plot, and **terrible** choice of actors. (Leslie Nielsen ...come on!!!) the one part I **considered** slightly funny was the battling FBI/CIA agents, but because the audience was mainly **kids** they didn't understand that theme.*
Adversarial Text Prediction = **Positive**. (Confidence = 59.8%)
*This movie had **horrific** acting, **horrific** plot, and **horrifying** choice of actors. (Leslie Nielsen ...come on!!!) the one part I **regarded** slightly funny was the battling FBI/CIA agents, but because the audience was mainly **youngsters** they didn't understand that theme.*

Original Text Prediction = **Positive**. (Confidence = 78%)
The promise of Martin Donovan playing Jesus was, **quite honestly**, enough to get me to see the film. Definitely worthwhile; clever and funny without overdoing it. The low quality filming was **probably** an appropriate effect but ended up being a little too jarring, and the ending sounded more like a PBS program than Hartley. Still, too many memorable lines and great moments for me to judge it harshly.
Adversarial Text Prediction = **Negative**. (Confidence = 59.9%)
The promise of Martin Donovan playing Jesus was, **utterly frankly**, enough to get me to see the film. Definitely worthwhile; clever and funny without overdoing it. The low quality filming was **presumably** an appropriate effect but ended up being a little too jarring, and the ending sounded more like a PBS program than Hartley. Still, too many memorable lines and great moments for me to judge it harshly.

Table 4.1: Examples of attack results for the sentiment analysis task. Modified words are highlighted in green and red for the original and adversarial texts, respectively.

Original Text Prediction = **Negative**. (Confidence = 74.30%)
Some **sort** of accolade must be given to 'Hellraiser: Bloodline'. It's **actually** out full-mooned Full Moon. It bears all the marks of, say, your 'demonic toys' or 'puppet master' series, without their **dopey**, uh, charm? Full Moon can get away with **silly** product because they know it's silly. These Hellraiser things, man, do they ever take themselves seriously. This **increasingly stupid** franchise (**though** not nearly as **stupid** as I am for having **watched** it) once made up for its low budgets by being stylish. Now it's just ish.
Adversarial Text Prediction = **Positive**. (Confidence = 51.03%)
Some **kind** of accolade must be given to 'Hellraiser: Bloodline'. it's **truly** out full-mooned Full Moon. It bears all the marks of, say, your 'demonic toys' or 'puppet master' series, without their **silly**, uh, charm? Full Moon can get away with **daft** product because they know it's silly. These Hellraiser things, man, do they ever take themselves seriously. This **steadily daft** franchise (**whilst** not nearly as **daft** as i am for having **witnessed** it) once made up for its low budgets by being stylish. Now it's just ish.

Original Text Prediction = **Negative**. (Confidence = 50.53%)
Thinly-cloaked retelling of the garden-of-eden story – nothing new, nothing shocking, although I feel that is what the filmmakers were going for. The idea is **trite**. Strong performance from Daisy Eagan, that's about it. I believed she was 13, and I was interested in her character, the rest left me cold.
Adversarial Text Prediction = **Positive**. (Confidence = 63.04%)
Thinly-cloaked retelling of the garden-of-eden story – nothing new, nothing shocking, although I feel that is what the filmmakers were going for. The idea is **petty**. Strong performance from Daisy Eagan, that's about it. I believed she was 13, and I was interested in her character, the rest left me cold.

Table 4.2: Additional examples of attack results for the sentiment analysis task. Modified words are highlighted in green and red for the original and adversarial texts, respectively.

Original Text Prediction: **Entailment** (Confidence = 86%)
Premise: *A runner wearing purple strives for the finish line.* **Hypothesis:** *A **runner** wants to head for the finish line.*
Adversarial Text Prediction: **Contradiction** (Confidence = 43%)
Premise: *A runner wearing purple strives for the finish line.* **Hypothesis:** *A **racer** wants to head for the finish line.*

Original Text Prediction: **Contradiction** (Confidence = 91%)
Premise: A man and a woman stand in front of a Christmas tree contemplating a single thought. **Hypothesis:** Two **people talk** loudly in front of a cactus.
Adversarial Text Prediction: **Entailment** (Confidence = 51%)
Premise: A man and a woman stand in front of a Christmas tree contemplating a single thought. **Hypothesis:** Two **humans chitchat** loudly in front of a cactus.

Table 4.3: Examples of attack results for the textual entailment task. Modified words are highlighted in green and red for the original and adversarial texts, respectively.

Original Text Prediction: **Contradiction** (Confidence = 94%)
Premise: A young girl wearing yellow shorts and a white tank top using a cane pole to fish at a small pond. **Hypothesis:** A girl wearing a **dress** looks off a **cliff**.
Adversarial Text Prediction: **Entailment** (Confidence = 40%)
Premise: A young girl wearing yellow shorts and a white tank top using a cane pole to fish at a small pond. **Hypothesis:** A girl wearing a **skirt** looks off a **ravine**.

Original Text Prediction: **Entailment** (Confidence = 86%)
Premise: A large group of protesters are walking down the street with signs. **Hypothesis:** Some people are holding up **signs** of protest in the street.
Adversarial Text Prediction: **Contradiction** (Confidence = 43%)
Premise: A large group of protesters are walking down the street with signs. **Hypothesis:** Some people are holding up **signals** of protest in the street.

Table 4.4: Additional examples of attack results for the textual entailment task. Modified words are highlighted in green and red for the original and adversarial texts, respectively.

	Sentiment Analysis			Textual Entailment	
	% success	% modified		% success	% modified
`Perturb` baseline	52%		19%	–	–
GenAttack	97%	14.7%		70%	23%

Table 4.5: Comparison between the attack success rate and mean percentage of modifications required by the genetic attack and perturb baseline for the two tasks.

open-source benchmarks, and can be found in the following repositories[1][2]. Model descriptions are given below.

Sentiment Analysis: We trained a sentiment analysis model using the IMDB dataset of movie reviews [MDP11]. The IMDB dataset consists of 25,000 training examples and 25,000 test examples. The LSTM model is composed of 128 units, and the outputs across all time steps are averaged and fed to the output layer. The test accuracy of the model is 90%, which is relatively close to the state-of-the-art results on this dataset.

Textual Entailment: We trained a textual entailment model using the Stanford Natural Language Inference (SNLI) corpus [BAP15]. The model passes the input through a ReLU "translation" layer [BAP15], which encodes the premise and hypothesis sentences by performing a summation over the word embeddings, concatenates the two sentence embeddings, and finally passes the output through 3 600-dimensional ReLU layers before feeding it to a 3-way softmax. The model predicts whether the premise sentence entails, contradicts or is neutral to the hypothesis sentence. The test accuracy of the model is 83% which is also relatively close to the state-of-the-art [CZL17].

4.4.1 Attack Evaluation Results

We randomly sampled 1000, and 500 *correctly classified* examples from the test sets of the two tasks to evaluate our algorithm. Correctly classified examples were chosen to limit the accuracy levels of

[1] `https://github.com/keras-team/keras/blob/master/examples/imdb_lstm.py`

[2] `https://github.com/Smerity/keras_snli/blob/master/snli_rnn.py`

the victim models from confounding our results. For the sentiment analysis task, the attacker aims to divert the prediction result from positive to negative, and vice versa. For the textual entailment task, the attacker is only allowed to modify the hypothesis, and aims to divert the prediction result from 'entailment' to 'contradiction', and vice versa. We limit the attacker to maximum $G = 20$ iterations, and fix the hyperparameter values to $S = 60$, $N = 8$, $K = 4$, and $\delta = 0.5$. We also fixed the maximum percentage of allowed changes to the document to be 20% and 25% for the two tasks, respectively. If increased, the success rate would increase but the mean quality would decrease. If the attack does not succeed within the iterations limit or exceeds the specified threshold, it is counted as a failure.

Sample outputs produced by our attack are shown in tables 4.1, 4.2, 4.3, and 4.4. Table 4.5 shows the attack success rate and mean percentage of modified words on each task. We compare to the `Perturb` baseline, which greedily applies the `Perturb` subroutine, to validate the use of population-based optimization. As can be seen from our results, we are able to achieve high success rate with a limited number of modifications on both tasks. In addition, the genetic algorithm significantly outperformed the `Perturb` baseline in both success rate and percentage of words modified, demonstrating the additional benefit yielded by using population-based optimization. Testing using a single TitanX GPU, for sentiment analysis and textual entailment, we measured average runtimes on success to be 43.5 and 5 seconds per example, respectively. The high success rate and reasonable runtimes demonstrate the practicality of our approach, even when scaling to long sentences, such as those found in the IMDB dataset.

Speaking of which, our success rate on textual entailment is lower due to the large disparity in sentence length. On average, hypothesis sentences in the SNLI corpus are 9 words long, which is very short compared to IMDB (229 words, limited to 100 for experiments). With sentences that short, applying successful perturbations becomes much harder, however we were still able to achieve a success rate of 70%. For the same reason, we didn't apply the `Perturb` baseline on the textual entailment task, as the `Perturb` baseline fails to achieve any success under the limits of the maximum allowed changes constraint.

4.4.2 User study

We performed a user study on the sentiment analysis task with 20 volunteers to evaluate how perceptible our adversarial perturbations are. Note that the number of participating volunteers is significantly larger than used in previous studies [JL17, ERL18]. The user study was composed of two parts. First, we presented 100 adversarial examples to the participants and asked them to label the sentiment of the text (i.e., positive or negative.) 92.3% of the responses matched the original text sentiment, indicating that our modification did not significantly affect human judgment on the text sentiment. Second, we prepared 100 questions, each question includes the original example and the corresponding adversarial example in a pair. Participants were asked to judge the similarity of each pair on a scale from 1 (very similar) to 4 (very different). The average rating is 2.23 ± 0.25, which shows the perceived difference is also small.

4.4.3 Adversarial Training

The results demonstrated in section 4.4.1 raise the following question: How can we defend against these attacks? We performed a preliminary experiment to see if adversarial training [MMS17b], the only effective defense in the image domain, can be used to lower the attack success rate. We generated 1000 adversarial examples on the cleanly trained sentiment analysis model using the IMDB training set, appended them to the existing training set, and used the updated dataset to adversarially train a model from scratch. We found that adversarial training provided no additional robustness benefit in our experiments using the test set, despite the fact that the model achieves near 100% accuracy classifying adversarial examples included in the training set. These results demonstrate the diversity in the perturbations generated by our attack algorithm, and illustrates the difficulty in defending against adversarial attacks. We hope these results inspire further work in increasing the robustness of natural language models.

4.5 Conclusion and Future Work

We demonstrate that despite the difficulties in generating imperceptible adversarial examples in the natural language domain, semantically and syntactically similar adversarial examples can be crafted using a black-box population-based optimization algorithm, yielding success on both the sentiment analysis and textual entailment tasks. Our human study validated that the generated examples were indeed adversarial and perceptibly quite similar. We hope our work encourages researchers to pursue improving the robustness of DNNs in the natural language domain.

CHAPTER 5

NeuroMask: Explaining Predictions of Deep Neural Networks through Mask Learning

In this Chapter, we demonstrate how the *NeuroMask* algorithm which can be used to explain the decisions of image recognition models.

5.1 Background

In the past decade, the world has witnessed a revolution in smart devices and machine intelligence. Computers in different forms and scales, ranging from servers to smart home devices, and from mobile phones to smart cars, are now achieving or exceeding human levels of autonomy and intelligence in certain specific situations. Much of this success has been made possible by the surge in the subset of machine learning algorithms known as Deep Neural Networks (DNNs). DNNs are a powerful way to learn approximations of the complex functions that underlie the process of decision making in many real-world applications (e.g. object recognition, image understanding, speech, and language understanding) that would have been hard for human and domain experts to tackle before. They also, most often, are trained using a domain-agnostic family of algorithms known as “back propagation" and “gradient descent" which require only using a labeled set of training examples and adjusting the model weight parameters to minimize an error, or *loss* function, defined over the training examples and model weights. The final set of weights are used in the model to carry out future predictions.

While the existing algorithms for training and building DNNs work effectively and *magically* to achieve unprecedented levels of accuracy in different application domains, they suffer from a major and, sometimes critical, limitation: DNNs lack the ability to provide explanations of how

the predicted outcomes were computed. This is extremely important. Humans often have a prior knowledge as to what are the important types of evidence or features which support a particular decision. A machine learning algorithm, however, models the correlations between input features and classes. It can learn relationships which are purely the result of noise, biases in training data, or even poorly formed machine learning problems. For example, a cancer diagnosis model from MRI images should highlight which part of the MRI image looks abnormal. A job applicant selection program should be able to explain why it has preferred a given applicant over another. The explanation is needed to ensure the fairness and correctness of the deployed models and that they are not using spurious features. The need for this was dramatically illustrated in a recent incident where Amazon deployed an experimental artificial intelligence recruiting tool which rated every applicant on a scale from one to five based on their application materials [REU18]. Not too long after the experiment began, it was discovered that the program would systemically assign CVs of women applicants lower ratings than those of men with similar qualifications. The sexist failing behavior of the program is of course not intended and could have been spotted earlier if the model had been able to explain its results. For these reasons, governments around the world have started to create regulations (e.g. the General Data Protection Regulation (GDPR) [Par16]) requiring organizations that use machine learning and artificial intelligence to make decisions affecting users (such as approval of loans, hiring, etc.) to also provide an explanation of their decision. In a similar effort, the US government has organized the DARPA's explainable Artificial Intelligence (XAI) [Gun17] to encourage and promote research efforts that address this challenging problem.

One method for understanding the performance of a learned algorithm is to look directly at its functional form. Some types of models, such as logistic regression and decisions trees, can sometimes be easy to understand. However, their results are often worse than those for DNNs. DNNs, on the other hand, learn models which are cascaded sequences of linear and non-linear operations (e.g. ReLU, tanh, sigmoid) which lead from the input to the final outcome. It is often impossible for a human to understand what evidence was used by the the model and how to reach its final conclusion. The limitation of DNNs to explain their outcomes and the evidence which supports them places barriers in their application in critical areas where an explanation is required and is as important as the outcome. This trade-off between model accuracy and interpretability

urges the need for a robust approach to generate explanations of DNNs predictions.

Motivated by the above, many authors have developed intepretability methods based on heat maps: the parts of an input image area assigned weights to show their relative importance in a DNN decision. While these efforts have lead to the creation of a number of methods (e.g. Saliency Map [SVZ13], LIME [RSG16], Smoothed Grad [STK17]) as a way to explain neural networks results, they are still either computationally inefficient or generate noisy results.

5.2 Contribution

Motivated by the above, we present *NeuroMask*, a new method that aims to generate better explanations by formulating the problem of generating explanations as an optimization problem to learn a 'mask' that hides parts of the input which are irrelevant to the model output and leaves important parts still visible to the model. We introduce an efficient algorithm for learning this 'mask' in Section 5.4. In addition, when applied to images, we also add additional constraints that make the explanation results more interpretable for a human observer by promoting the mask to hide/reveal contiguous parts of the input image instead of individual pixels. *NeuroMask* does not require any modification to the architecture or training algorithm of the DNN model and can be applied to produce explanations for the outputs of any pre-trained model.

We demonstrate the effectiveness of *NeuroMask* by showing its explanations for the predictions of state-the-art ImageNet [DDS09] and CIFAR-10 [KH09] image recognition models on different examples. Visual comparison between *NeuroMask* and explanations generated by other methods (Saliency Map [SVZ13], Smoothed Grad [STK17], Guided Backprop [SDB14], LIME [RSG16], and LRP-epsilon [BBM15a]) reflect the success of *NeuroMask* to produce high-quality explanations.

The rest of this chapter is organized as follows: Section 5.3 summarizes the related work. Section 5.4 describes the assumptions and implementation of the *NeuroMask* algorithm. Section 5.5 describes our evaluation experiments and provides visual examples of explanations generated by *NeuroMask*. Finally, Section 5.6 concludes the chapter and presents directions for our future work.

5.3 Related Work

Over the past few years, researchers have studied the problem of DNNs interpretability. While the definition of interpretability itself is still confusing as discussed in [Lip16b]. Most of the work [STK17, SVZ13, ZF14, STK17] use the term to refer how to explain the DNNs prediction results in terms of its own input which is the same definition we consider in this chapter.

The major existing methods for interpretability can coarsely be categorized according to how they work into one of the following categories:
(a) **Occlusion-based**: such as [ZF14] which systemically occlude different parts of the input image with a grey square and monitor the change of the prediction class probability. This method while being effective is computationally inefficient due to its brute-force nature that requires trying occlusion square at every possible position in the input image. It is also unsuitable when objects in the image have different scales and arbitrary shapes.
(b) **Gradient-based**: The Saliency Map [SVZ13] method relies on computing the gradient of the prediction class label with respect to the model input to estimate features importance. However, the results of Saliency Map are often noisy and hard to interpret. To improve further, The gradient-weighted action mapping (GRAD-CAM) [SCD17] uses the gradient of output label with respect to the final convolution layer to produce a coarse localization map highlighting important region in the image. Similarly, the SmoothGrad [STK17] improves the quality of saliency maps by reducing the visual noise by using a regularization technique. The Layerwise Relevance Propagation (LRP) and Deep LIFT have been recently proposed as an alternative method. The difference between these two methods and prior work is that they attempt to estimate the global importance of pixels, rather than the local sensitivity. LRP [BBM15b] was the first to propose the pixel-wise decomposition of classifiers in order to produce an explanation for a classification decision. They evaluate individual pixel contributions and produce "interpretable" heatmaps. Guided-Backprop [SDB14] uses backpropagation and deconvolution operation to invert the computation of the DNN in order to visualize the concepts represents by intermediate layers. Guided-Backprop can be considered as equivalent to computing gradients except for the case when the gradient becomes negative then it will be zeroed out.

(c) **Approximate local model-based**: such as Local Interpretable Model-agnostic Explanations (LIME) [RSG16] algorithm. LIME [RSG16] generates an explanation of a model prediction of a given input by training another model (*explainer*) which is selected from a group of intrinsically interpretable group of models (e.g. linear models, decision trees, etc.). The explainer is trained to approximate the model's decision surface around the provided input example. Training instances are obtained by drawing samples uniformly at random from the perturbed neighborhood of the input example. The perturbed samples are labeled with their prediction outcome from the given model and given the set of perturbed samples and their labels. The explainer is trained to mimic the decision surface of input model around the given example. LIME has been used to provide explanations for models of different data formats including text, tabular, and images. For images datasets, LIME uses super-pixels rather than individual pixels while producing the explainer training instances in order to produce a more interpretable explanation in terms of super-pixels. However, the reliance of superpixels sometimes can cause LIME to fail as we have observed in our experiments. It is also computationally inefficient due to the necessity of training the explainer model. During our literature survey to prepare this manuscript, we have found the approach of [FV17] to be the most similar to our idea. Both methods attempt to generate interpretable explanations of DNN decision by learning a mask that perturbs the input image. Nevertheless, our method is novel in its algorithm and cost function definition.

A more comprehensive review of literature in this topic can be found in [CTR17] and [GMR18].

5.4 Algorithm Design

The basic idea behind *NeuroMask* is that input features which are not strongly relevant to the model's classification decision can be suppressed from the input without affecting the model's output. In order to find out which pixels are influential/unimportant, *NeuroMask* maintains two clones of the given pre-trained classification model. As an input to one of them, we feed the input example for which we seek an explanation of the model's result. The image fed into the other model copy is the input example after suppressing part of its features. By comparing the outputs of the two copies, *NeuroMask* attempts to find which are the unimportant features that could be suppressed

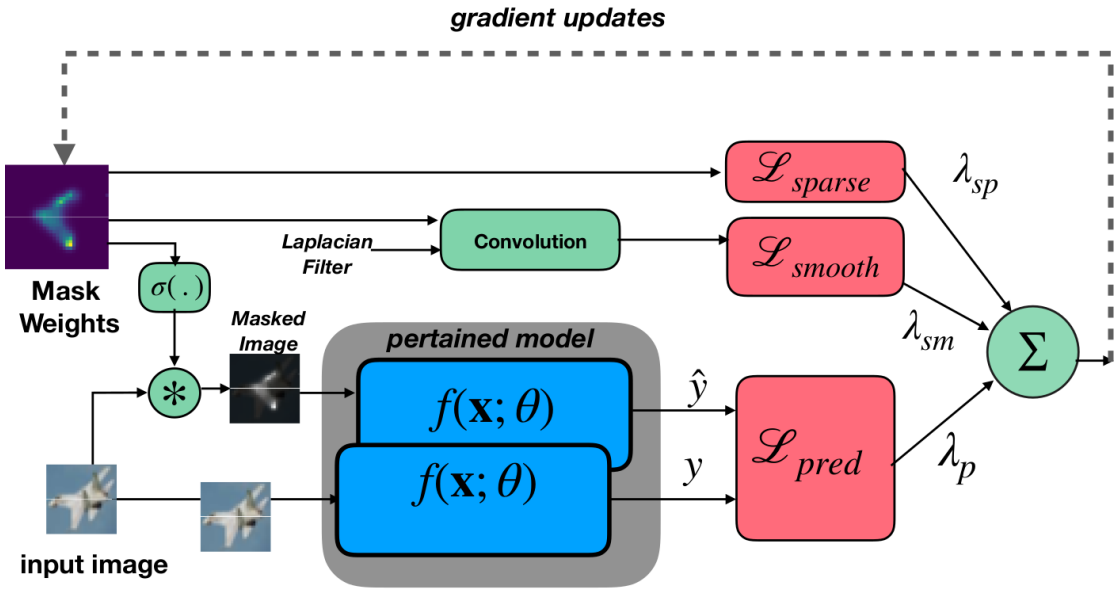


Figure 5.1: Interactions of different components in *NeuroMask*. The blue blocks are clones of the pre-trained models. The dotted line represent gradient updates the adjust the values of mask weights.

while maintaining close similarity between the outputs of the two model copies. Importantly, the output explanation should be *comprehensible* for a human observer. Toward this end, we design *NeuroMask* such that the explanation output is both *minimal* and *interpretable*. In this rest of this section, we give more details on how *NeuroMask* operates.

We formalize the problem of explaining the predictions of a DNN from a given example, as follows:
Given

- $\mathbf{x} \in \mathbb{R}^{H \times W \times 3}$: an input example representing RGB image whose height is H and width is W.
- $f(\mathbf{x}; \theta) : \mathbb{R}^{H \times W \times 3} \rightarrow [0, 1]^L$: a pre-trained image recognition model that maps input example to one of different L classification labels. θ denotes the set of model parameters.

The goal of *NeuroMask* is to learn the values of a single channel mask filter $\mathbf{m} \in [0, 1]^{H \times W}$ with the same height and width as the input image. Elements of the mask are real values the $[0, 1]$

corresponding to the relevance of the input image pixels at the same coordinate to the model prediction. The value 1 signifies a strongly relevant pixel and 0 deemed as an irrelevant pixel.

Figure 5.1 illustrates the different components of *NeuroMask* and how they interact with each other. At the heart of *NeuroMask*, there are two copies of the given pre-trained model (shown within the grey box). The two copies are identical and their weights remain frozen during the operation of *NeuroMask*. They only differ in the input applied to each one of them. Additionally, *NeuroMask* defines a set of trainable parameters which we refer to as 'mask weights' $\mathbf{W}$. The mask weights are initialized from a uniform random distribution, and transformed into a 'relevance mask' $\mathbf{m}$ using a `sigmoid` function. The relevance mask is what we need to compute by *NeuroMask* through the algorithm described in 3. One of the two model copies receives the input example as its input while the other receives the result of multiplying (*pixel-wise*) the input example by the relevance mask. Outputs from the two models are fed into a cost function $\mathcal{L}_{pred}$ the measures the distance between their predictions. In addition, the mask weights $\mathbf{W}$ goes into two additional cost terms $\mathcal{L}_{sparse}$ and $\mathcal{L}_{smooth}$ which are designed to improve the interpretability of the final mask weights. The weighted sum of these three terms ($\mathcal{L}_{pred}$, $\mathcal{L}_{sparse}$, $\mathcal{L}_{smooth}$) represent the total cost function $\mathcal{L}_{total}$ of *NeuroMask*.

$$\mathcal{L}_{total} = \lambda_p \ \mathcal{L}_{pred} + \lambda_{sp} \ \mathcal{L}_{sparse} + \lambda_{sm} \ \mathcal{L}_{smooth}$$

where λ_p, λ_{sp}, and λ_{sm} are weighting coefficient to balance between the different components of cost function.

We use the RMSProp [TH12] optimization algorithm to compute the final mask weights $\mathbf{W}^*$ that minimizes this cost function. Therefore, the final mask weights are defined by

$$\mathbf{W}^* = \arg\min_{\mathbf{W}} \quad \mathcal{L}_{total}(\mathbf{W}; \mathbf{x}, \theta)$$

The three components in *NeuroMask* cost function are defined according the the following:

- **Prediction cost** $\mathcal{L}_{pred}$: which measures the distance between the predictions of the two clones of the given model. It is defined as the cross-entropy between the two model copies outputs.

$$\mathcal{L}_{pred}(y, \hat{y}) = - \sum_{i \in \{1,2,..L\}} \mathbb{1}(i = \arg\max_{j \in \{1,2,..L\}} y_j) \ \log(\hat{y}_i)$$

- **Sparseness cost** $\mathcal{L}_{sparse}$: A high-quality explanation should be minimal. Rather than declaring all pixels in the input image as relevant to the model's prediction, we need to identify as small as possible subset of pixels that are considered the most relevant. This can be achieved by forcing the "relevance mask" $\mathbf{m}$ to be sparse matrix. The sparseness of $\mathbf{m}$ can be encouraged by defining $\mathcal{L}_{sparse}$ as a L_1 regularizer over the mask weights.

$$\mathcal{L}_{sparse}(\mathbf{W}) = \sum_{i=1}^{H}\sum_{j=1}^{W} |\mathbf{W}_{i,j} + \tau|$$

  This will force as many as possible elements of matrix $\mathbf{W}$ to be equal to $-\tau$. The τ (we pick its value equal 20) is chosen such that $\sigma(-\tau) \approx 0$.

- **Smoothness cost** $\mathcal{L}_{sparse}$: In addition to being minimal, it is also desirable for an explanation to be *interpretable*. I.e., the explanation would ideally be defined in terms of objects and object parts in the image rather than a subset of ungrouped pixels. Previous work (LIME, [RSG16]) addressed this requirement by expressing the explanation in terms of super-pixels which are patches of nearby pixels with similar color intensities, the approach would often fail when we have nearby objects with similar colors (as we show in our results section). Inspired by the work of [ZNZ18] that altered the training algorithm of convolutional networks so that convolution filters correspond to interpretable parts of objects in the image, we employ a similar constraint that encourages the mask weights to be smooth and therefore highlights groups of spatially co-located pixels. Accordingly, the definition of $\mathcal{L}_{sparse}$ is chosen to be the L_1 norm of the 2nd derivative of the mask weights, which are computed by convolving $\mathbf{W}$ with a discrete Laplacian filter f_s

$$\begin{aligned}\mathcal{L}_{smooth}(\mathbf{W}) =& |\nabla^2\mathbf{W}(x,y)|_1 = \left|\frac{\delta^2\mathbf{W}}{\delta x^2} + \frac{\delta^2\mathbf{W}}{\delta y^2}\right|_1, \\ =& |\mathbf{W} \circledast f_s|_1,\end{aligned}$$

  where $\circledast$ denotes the 2d convolution operation.

Algorithm 4 describes the steps to compute the explanation mask by minimizing the cost function.

Algorithm 4: Optimization algorithm to compute the explanation mask $\mathbf{m}$.

Input: input example $\mathbf{x} \in \mathbb{R}^{H \times W}$, a pre-trained prediction model $f : \mathbb{R}^d \rightarrow [0,1]^L$.

Output: m explanation mask showing the relevance of different image parts to the model decision.

$W \sim Uniform(-\tau, \tau)$ ▷ ***Initialize mask weights***

for $i = 1, ..., T$ **do**

  $\bar{\mathbf{x}} = \sigma(\mathbf{W}) \cdot \mathbf{x}$ ▷ ***Apply mask to input***

  $y = f(\mathbf{x}; \theta)$ ▷ ***Prediction of original input***

  $\hat{y} = f(\bar{\mathbf{x}}; \theta)$ ▷ ***Prediction of masked input***

  $\mathcal{L}_{pred} = -\sum_{i=1}^{L} \mathbb{1}(i = \arg\max_j y_j) \, \log(\hat{y}_i)$

  $\mathcal{L}_{sparse} = |\mathbf{W}|_1$

  $\mathcal{L}_{smooth} = |\mathbf{W} \circledast f_s|_1$

  $d\mathbf{W} = \nabla_{\mathbf{W}} (\lambda_p \mathcal{L}_{pred} + \lambda_{sp} \mathcal{L}_{sparse} + \lambda_{sm} \mathcal{L}_{smooth})$

  ▷ ***RMSProp optimizer step***

  $v_{dw} = \beta v_{dw} + (1 - \beta) dW^2$

  $W = W - \alpha \frac{dW}{\sqrt{dW} + \epsilon}$

end for

$\mathbf{m} = \sigma(\mathbf{W})$ ▷ ***Final mask***

5.5 Evaluation Results

We evaluate the effectiveness of *NeuroMask* by demonstrating the visual quality of the explanations it generates of the predictions made by pre-trained state-of-art image recognition models. These models are tested using the CIFAR-10 [KH09] and ImageNet [DDS09] datasets. We also compare *NeuroMask* outputs to the outputs to other state-of-the-art interpretability methods.

5.5.1 CIFAR-10 Results

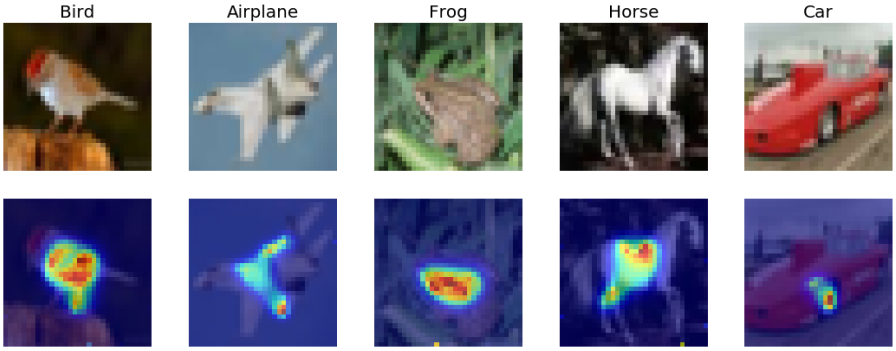


Figure 5.2: Using *NeuroMask* to explain the predictions on a pre-trained model for image recognition on randomly selected examples from the CIFAR-10 dataset.

The CIFAR-10 dataset [KH09] contains small images (32x32 pixels) for 10 different categories. We used a convolutional network from [CW17b] that reaches near to state-of-the-art (80%) classification accuracy on the CIFAR-10 dataset. We use *NeuroMask* to explain the classification model predictions on randomly selected images. As shown in figure 5.2, *NeuroMask* can accurately localize the object within the image and highlight its discriminative parts (for example; tires of the car and wings of the airplane). This indicates that *NeuroMask* is effective in explaining the decisions of the pre-trained model.

5.5.2 ImageNet Results

We also performed comparison to evaluate the quality of explanations produced by *NeuroMask* versus prominent state-of-the-art methods (Saliency Map [SVZ13], Smoothed Grad [STK17], Grad-

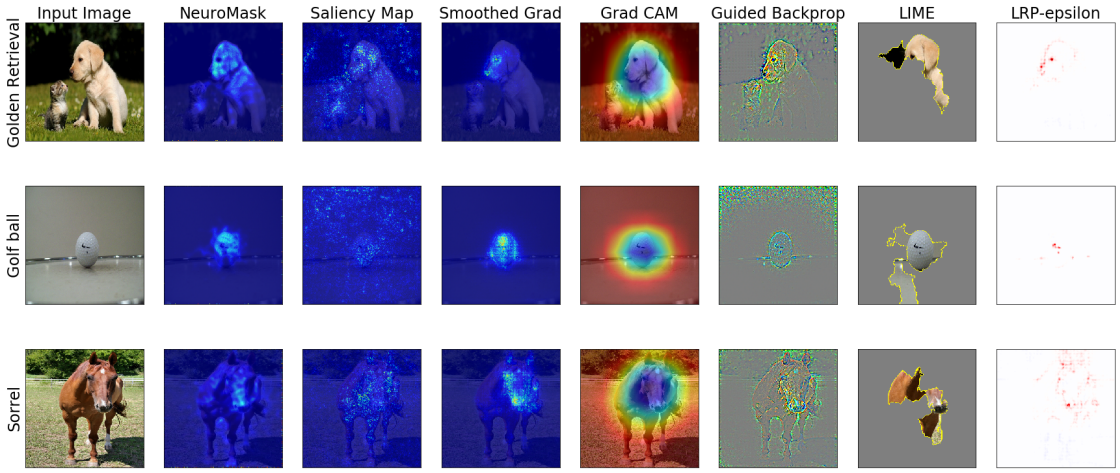


Figure 5.3: Qualitative evaluation of the explanations make by *NeuroMask* vs other state-of-the-art interpretability methods (Saliency Map [SVZ13], Smoothed Grad [STK17], Grad-CAM [SCD17], Guided Backprop [SDB14], LIME [RSG16], LRP-epsilon [BBM15a]) on images selected from ImageNet [DDS09] test dataset using Inception_v3 [SVI15] image recognition model.

CAM [SCD17], Guided Backprop [SDB14], LIME [RSG16], and LRP-epsilon [BBM15a]) for DNNs interpretability. We randomly select test images from the ImageNet [DDS09] test dataset. The ImageNet dataset contains large scale (299×299 pixels) images for 1000 different classes of images. In our experiments, We use the Inception-v3 [SVI15] as the pre-trained image recognition model (with 93.2% top-5 accuracy). The implementation of LIME [RSG16] was obtained from the author's Github repo[1], while for the remainder of methods we use the implementations provided by the iNNvestigate tool kit [ALS18].

Our remarks from the visual comparison are as follow: both *NeuroMask* and Smoothed Grad [STK17] produce explanations that are accurate and easy to interpret for a human observer. On the other hand, explanations produced by Saliency Map [SVZ13] and Guided Backprop [SDB14] have too much noise which makes them hard to understand. By contrast, Grad-CAM [SCD17] is too coarse grained while LRP-epsilon [BBM15a] is too conservative in what it highlights. Finally, LIME [RSG16] works well in some cases but fails in others (such as in the 2nd row) where its

[1] `https://github.com/marcotcr/lime`

reliance on superpixels causes it to declare the background as important as the golf-ball itself.

Based on these results, we conclude that *NeuroMask* has a lot of potential and promise to provide explanations for DNNs outcomes. Nevertheless, in the future, we plan to conduct more comprehensive evaluation studies that include doing user's surveys to analyze their feedback on the different interpretability results.

5.5.3 *NeuroMask* Learning Progress

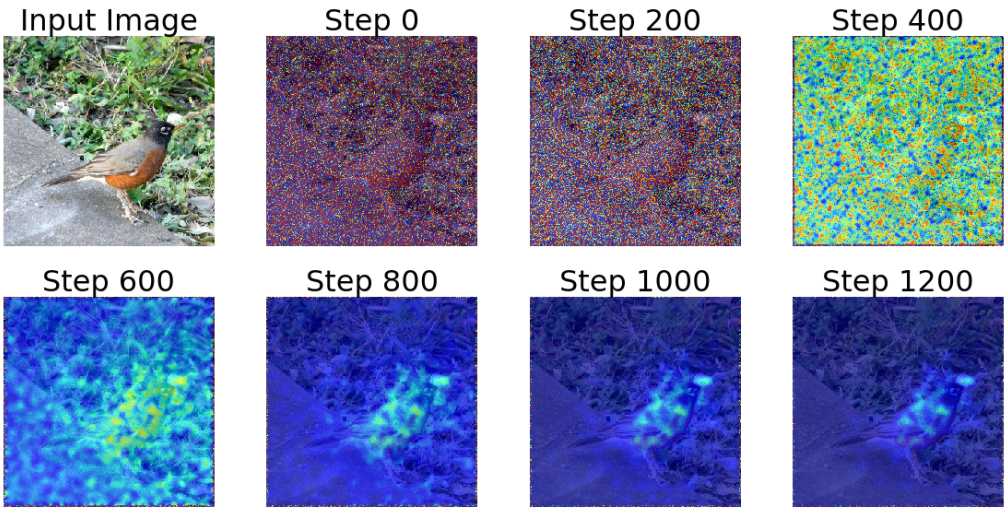


Figure 5.4: Evolution of the explanation by *NeuroMask* during the optimization process.

Figure 5.4 shows the progress of learning an interpretable explanation during the optimization process of *NeuroMask*. Since the mask weights are initialized as uniformly random, the mask initially looks like white noise (step 0) and then gradually it starts to focus more on the relevant part of the image as shown in Step 600. After that, the mask weights are refined to focus on important parts of the object which are most relevant to the classifier outcome.

5.5.4 Using *NeuroMask* to Detect *Backdoor* Triggers

AI models are vulnerable to *backdoor attacks*. An adversary may publish online trained models which have *injected* trojans into them. The *backdoored* model will produce normally expected results except for the case when the input has a specific *backdoor trigger*. This *backdoor trigger*

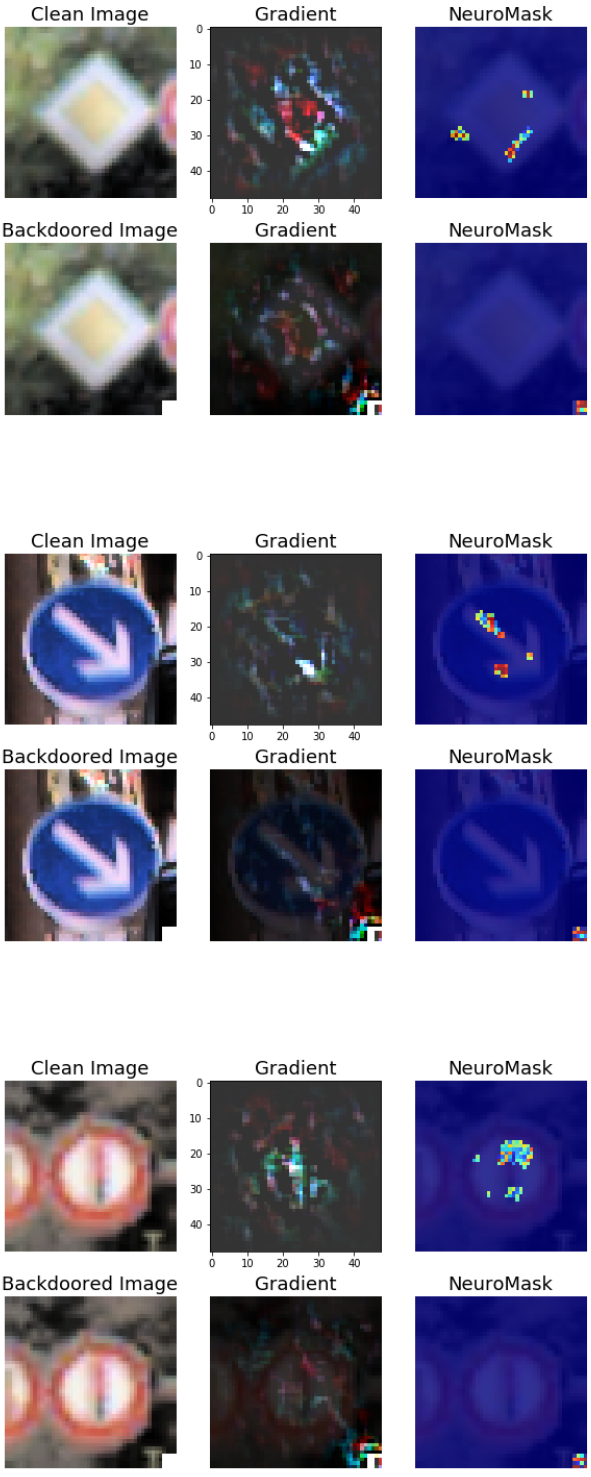


Figure 5.5: Different examples of Verifying *NeuroMask* explanation using a *backdoored* model using the *BadNet* [GDG17] attack. The top rows represent a *clean* correctly classified image while the bottom rows represents a *backdoored* image with the trojan trigger (*the white square at the bottom right corner*)

which is only known to the attacker will cause the model to misclassify. We perform experiments using the German Traffic Sign (GTSRB) [SSS11] dataset classification model. We train *backdoored* model on this dataset using the *BadNet* [GDG17] attack that poisons the training data with a *trojan trigger* causing the model to misclassify. The model test accuracy on normal *clean* images is 97%, but any image with the *white square* at the bottom right corner will be misclassified as *speed limit* sign. We use *NeuroMask* to generate model explanations for pairs of images before and after adding *backdoor trigger*. Figure 5.5 shows the generated explanations using both *gradient saliency map* and *netmask*. Top rows represent the *clean* correctly classified images while bottom rows represent the *backdoored* image . We notice that *NeuroMask* accurately identify the trigger as the explanation of prediction of the backdoored images.

5.6 Conclusion and Future Work

In this chapter, we presented *NeuroMask* a novel approach for generating explanations for the predictions of pre-trained deep neural networks. *NeuroMask* is model agnostic and can be used to explain the outputs of any image recognition models. The visual quality of explanations shows the success of *NeuroMask* to identify which parts of the input image were relevant to the classifier decision. Compared to explanations generated by other interpretability methods, we find *NeuroMask* to produce competitive explanations.

Future Work Our directions for future work include: extending *NeuroMask* to DNNs used for other tasks such as image captioning, and data modalities such as text and sound. We are also going to conduct users study surveys to understand their perception and feedback of the *NeuroMask* generated explanations.

CHAPTER 6

Deep Residual Neural Networks for Audio Spoofing Detection

In this Chapter, we introduce methods to defend the smart voice-controlled devices against *audio spoofing attacks*.

6.1 Background

Over the past decade, voice control has gained popularity as a practical and comfortable interface between users and smart devices. Due to the security and privacy sensitive nature of many applications (e.g., banking, health, and smart home) running on these devices, automatic speaker verification (ASV) [EKY13] techniques have emerged as a form of biometric identification of the speaker. However, ASV systems are threatened by replay [KEY17] and audio spoofing attacks where an attacker utilizes techniques such as voice conversion (VC) or speech synthesis (SS) to gain illegitimate control over user devices. Speech synthesis [ODZ16, WWK16, JBW18] and voice conversion [TCS16, HLH18] have also progressed a lot over the past decade reaching the point where it has become very challenging to differentiate between their results and genuine users' speech. To enhance reliability against attacks, we combine ASV systems with audio spoofing detection systems that compute countermeasure scores to distinguish between spoofed and bonafide (genuine) speech. The automatic speaker verification spoofing and countermeasure challenge (ASVSpoof [EKY13, WKE15, KEY17, con19]) competitions have emerged to assess the state-of-art methods for spoofing detection and promote further research in this critical challenge.

The first edition of the competition, ASVSpoof2015[WKE15], focused on *logical access* scenarios where the attacker is using text-to-speech (TTS) [ODZ16, HLH18, WWK16] and voice conversion (VC) [TCS16, HLH18] algorithms. The second edition of ASVSpoof competition,

ASVSpoof2017 [KEY17], focused on the *physical access* scenario where the attacker is performing *replay attack* by recording the genuine speech and then replay it to deceive the ASV system. The new edition of the competition, ASVSpoof2019 [con19], extends the previous versions in several directions. First, it considers all three major forms of attacks: SS, VC, and replay attacks. Besides, the latest and strongest spoof algorithms are used to generate more natural counterexamples for spoof detection systems. Finally, while the previous competitions used the equal error rate (EER) as an evaluation metric, ASVSpoof 2019 adopts a newly proposed tandem decision cost function (t-DCF) as its primary metric and leaves EER as a secondary metric.

6.2 Contribution

In this chapter, we present our models submitted for the ASVSpoof2019 competition [con19]. Inspired by the success of deep neural networks in many tasks [AAA16, SLJ15, EKN17], we pick a deep neural model as our model family. Among deep neural networks, convolutional networks have been the most successful in image classification [SLJ15], and have been recently applied to other data modalities such as Speech [AMJ14, AAA16], text [ZZL15] and ECG signals [RHH17]. We consider different feature extraction algorithms to convert the input (raw time-domain speech waveform) into a 2D feature representation. That 2D feature representation is fed as an input into our convolutional model. A practical challenge in training very deep (consisting of many layers) convolutional networks is vanishing gradients that makes it hard for lower-layers (closer to input) to receive useful update signals during the training [HZR16]. To overcome this issue, [HZR16] recently proposed an effective solution called *residual networks* which employ skip connections that act as shortcuts allowing training updates to back-propagate faster towards the lower layers during training. Therefore, we also consider adding residual links to improve and stabilize the training of our models. A detailed description of our model architecture is provided in Section 6.4.2. Finally, we show how the fusion of countermeasure (CM) scores produced by models trained on different features help to increase the accuracy of the spoofing detection.

Our contribution in this chapter is threefold. First, we design and implement a deep residual

convolutional network to perform audio spoofing detection. Our models are released as open source[1]. Second, we provide a comparison between the performance of three different feature extraction algorithms (MFCC, log-magnitude STFT, and CQCC). Third, we evaluate the performance of our residual network with varying choices of input features against the two attack scenarios of ASVSpoof2019 (logical access, and physical access) using both the development (including only *known* attacks) and evaluation datasets (including both *known* and *unknown* attacks).

The rest of this chapter is organized as follows. Section 6.3 provides a summary of related work. Section 6.4.1 describes the feature extraction module of the system. Section 6.4.2 then describes our model architecture design and implementation. Section 6.5 includes our experiment results. Finally, Section 6.6 concludes the chapter and points the future directions.

6.3 Related Work

While the participants of the previous ASVspoof2015 [WKE15] have built several powerful solutions against audio spoofing, the state-of-the-art of audio spoofing techniques, e.g., TTS [ODZ16, HLH18] and VC [LYT18], has also progressed a lot over the past four years. Likewise, this year's competition ASVSpoof2019 has a more realistic dataset for replay attacks compared to ASVSpoof2017 [KEY17]. Prominent previous approaches against *logical access* attacks include [VMO15] which used spectral-log-filter-bank and relative phase shift features as input to a model combining a deep neural network with support vector machine (SVM) classifier. [CQD15] proposed using a DNN to compute a representative spoofing vector (s-vector). Then it uses normalized Mahalanobis distance between the s-vector and the class representative vector to calculate countermeasure scores. [WYK15] uses relative phase information and group delay feature to train a Gaussian Mixture Model (GMM) for detecting spoofing attacks. Against *replay* attacks, [LNM17] have previously developed a deep learning model combining both CNN and RNN that lead to 6.73% EER on the ASVSpoof2017 evaluation dataset. In ASVSpoof2017, [CXZ17] also used a residual convolutional network, but with different an architecture and input features, to obtain 13.44% EER on the eval set.

[1] https://github.com/nesl/asvspoof2019

6.4 Model Design

The goal of ASVspoof challenge is to compute a countermeasure (CM) score for each input audio file. A high CM score indicates a bonafide speech, and a low CM score indicates a spoofing attack. We created a deep residual network that performs binary classification. To prepare the features as the convolutional network inputs, we process the raw audio waveform first a by a feature extraction step which we will discuss in the next section.

6.4.1 Feature Extraction

We prepare features from raw audio waveform by one of the following feature extraction algorithms: the Mel-Frequency Cepstral Coefficients (MFCCs), the Constant Q Cepstral Coefficients(CQCCs), and the Logarithmic Magnitude of Short-Time Fourier Transform(log-magnitude STFT).

Mel-frequency Cepstral Coefficients (MFCCs): MFCC is a widely used feature for speech recognition and other applications like music genre classification. MFCC is achieved by computing the short-time-Fourier-transform (STFT), then mapping the spectrum into Mel-Spectrum through a filter bank, and finally calculating a discrete cosine transform(DCT). We pick the first 24 coefficients. We also find the performance can be improved if we concatenate the MFCC with its first-order $\Delta MFCC$ and second derivative $\Delta^2 MFCC$ to produce our feature representation which is a 2D matrix whose x axis is the time and y axis is the 72 elements of $(MFCC, \Delta MFCC, \Delta^2 MFCC)$. This improvement is because derivatives of MFCC capture the dynamics in cepstral coefficients.

Constant Q Cepstral Coefficients(CQCCs): Instead of using STFT, the CQCC uses constant-Q transform(CQT) which was initially proposed for music processing. While STFT imposes a regularly spaced frequency bins, the CQT uses geometrically spaced frequency bins. Thus, it offers a higher frequency resolution at lower frequencies and higher temporal resolution at higher frequencies. To compute CQCC, after applying CQT, we calculate a power spectrum and take a logarithm. Then a *uniform re-sampling* is performed, followed by a DCT to get the CQCCs(which is also a 2D matrix). More details of CQCC can be found in [TDE17].

Logarithmic Magnitude of STFT: An advantage of deep learning models is their capabilities

of representation learning [BCV13, GBC16] by automatically learning high-level features from raw input data. This ability has led to neural models which process raw input images to outperform models dealing with human-engineered features. Inspired by this, we also train models with the log magnitude of STFT as the input. We first compute the STFT on hamming windows (window size = 2048 with 25% overlap). Then we calculate the magnitude of each component and convert it to log scale. The output matrix captures the time-frequency characteristics of the input audio waveform and is fed directly as an input to our neural model without any further transformations or conversions. While this input representation is rawer than either MFCC or CQCC, we rely on the representation learning abilities of neural networks to transform this input into higher-level representations within the hidden layers of our model.

6.4.2 Model Architecture

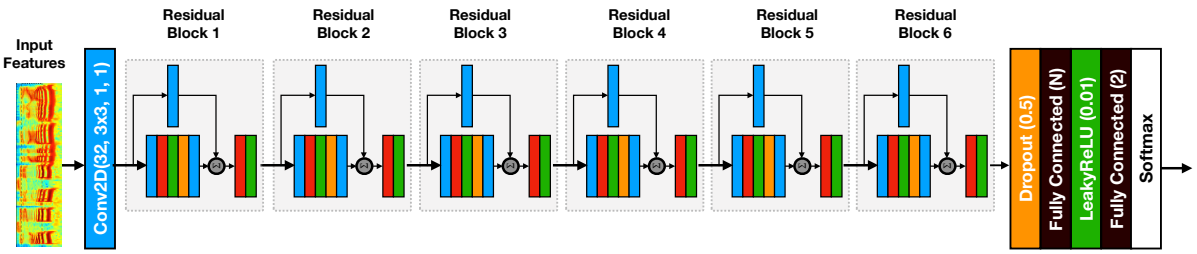


Figure 6.1: Model architecture for the `Spec-ResNet` model. Detailed structure of residual blocks is shown in 6.2.

We build three different models variants `MFCC-ResNet`, `CQCC-ResNet`, and `Spec-ResNet` which process MFCC, CQCC and log-magnitude STFT (which turns out to be a spectrogram) input features, respectively. The three variants have a nearly identical architecture, but they differ from each other in the input shape to accommodate the differences in the dimensions of input features, and consequentially also the number of units in the first fully connected layer which is after the last residual block, as we will explain later.

Figure 6.1 shows the architecture of the `Spec-ResNet` model which takes the log-magnitude STFT as input features. First, the input is treated as a single channel image and passed through a 2D

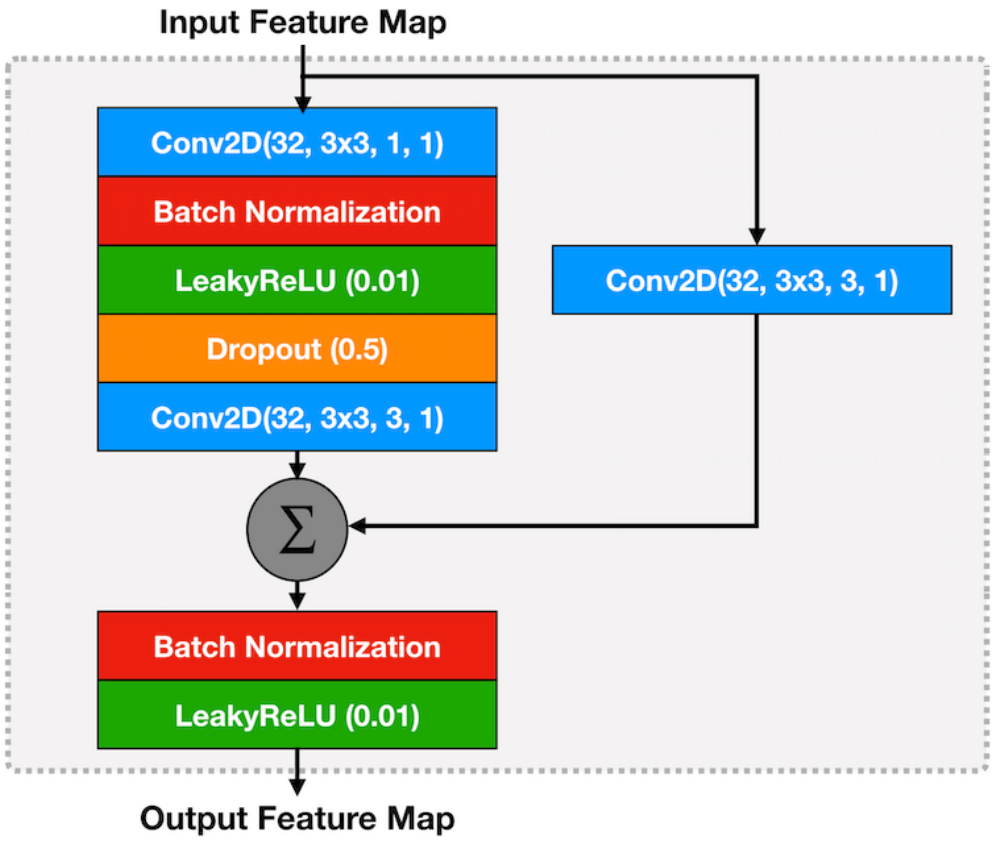


Figure 6.2: Detailed architecture of the convolution block with residual connection.

convolution layer with 32 filters, where filter size = 3×3, stride length = 1 and padding = 1. The output volume of the first convolution layer has 32 channels and is passed through a sequence of 6 *residual blocks*. The output from the last residual block is fed into a dropout layer [SHK14] (with dropout rate = 50%) followed by a hidden fully connected (FC) layer with leaky-ReLU [HZR15] activation function ($\alpha = 0.01$). Outputs from the hidden FC layer are fed into another FC layer with two units that produce classification logits. The logits are finally converted into a probability distribution using a final `softmax` layer.

The structure of a residual block is shown in Figure 6.2. Each residual block has a `Conv2D` layer (32 filters, filter size = 3×3, stride = 1, padding = 1) followed by a batch normalization layer [IS15], a leaky-ReLU activation layer [HZR15], a dropout (with dropout probability = 0.5) [SHK14], and another final `Conv2D` layer (also 32 filters and filter size = 3×3, but with stride = 3 and padding = 1). Dropout is used as a regularizer to reduce the model overfitting, and batch normalization [IS15] accelerates the network training progress. A skip-through connection is established by directly add the inputs to the outputs. To guarantee that the dimension agrees, we apply a `Conv2D` layer (32 filters, filter size = 3×3, stride = 3, padding = 1) on the bypass route. Finally, batch normalization [IS15] and leaky-ReLU non-linearlity are used to produce the residual block output.

All models are trained by minimizing a weighted cross-entropy loss function where the ratio of between weights assigned to genuine and spoofed examples are 9:1, in order to mitigate the imbalance in the training data distribution. The cost function is minimized using Adam optimizer [KB14b] with learning rate = 5×10^{-5} for 200 epochs with batch size = 32. After each epoch we save the model parameters, and we finally use the parameters with the best performance on the validation dataset.

The final countermeasure score (CM) is computed from the softmax outputs using the log-likelihood ratio.

$$CM(s) = \log(p(\text{bona fide}|s;\theta)) - \log(p(\text{spoof}|s;\theta))$$

where s is the given audio file and θ represents the model parameters.

6.5 Evaluation

We implemented our neural network model using PyTorch [PGC17] and trained our models using a desktop machine with TitanX GPU. Feature extraction was done using the `librosa` [MRL15] python library.[2].

6.5.1 Dataset and Baseline Models

The competition organizers provide a dataset of non-overlapping short audio files for each competition track. The bonafide voice clips come from 78 human (male and female) speakers. The dataset is divided into three partitions with disjoint sets of speakers: *training* (8 male, 12 female), *development* (4 male, 6 female), and *evaluation* (21 male, 27 female). The spoofed audio in the *logical access* scenario is generated using 17 different speech synthesis and voice conversion toolkits. Six of these attack types are considered *known* attacks and are used to generate the training and development datasets while the other 11 attacks are considered *unknown* and are used, along with two of the *known* attacks, to generate the evaluation dataset. For *physical access* scenario, replay attacks are recorded and replayed in the 27 different acoustic configurations and nine different settings (combinations of three categories of recording distance and three levels of replay device quality) [con19]. Evaluation data for the *physical access* are generated from different impulse responses and therefore represents *unknown* attacks. Distributions of training and validation datasets are shown in Table 6.1.

Baseline Models : For each track of the competition, the organizers have provided implementations for two baseline models which are using Gaussian mixture models (GMMs) [RR95, RQD00] using the Linear Frequency Cepstral Coefficients (LFCC) and CQCC features.

6.5.2 Evaluation Metrics

The evaluation scores are computed using the following metrics on both the development dataset (*known* attacks) and evaluation dataset (both *known* and *unknown* attacks):

t-DCF [KLD18]: the *tandem detection cost function* is the new primary metric in the ASVSpoof

[2]For the CQCC for which we used the MatLab code provided by competition organizers

	Logical access		Physical access	
Subset	Bona fide	Spoof	Bona fide	Spoof
Training	2,580	22,800	5,400	48,600
Development	2,5480	22,296	5,400	24,300

Table 6.1: Number of audio files in the ASVspoof 2019 dataset.

2019 challenge. It was proposed as a reliable scoring metric to evaluate the combined performance of ASV and CMs.

EER: the Equal Error Rate is used as a secondary metric. EER is determined by the point at which the miss (false negative) rate and false alarm (false positive) rate are equal to each other.

6.5.3 Results

Table 6.2 shows a comparison between the scores of our three model variants (`MFCC-Resnet`, `Spec-ResNet`, `CQCC-ResNet`) and the baseline algorithms (`LFCC-GMM`, and `CQCC-GMM`) on both the development and evaluation datatsets. *Fusion* represents the result of doing *weighted* average of the individual ResNet models' *CM* scores to provide a final *CM* score, where fusion weights are assigned based on the single model's performance on the validation dataset.

6.5.3.1 Logical Access Results

As shown in Table 6.2, Our `Spec-ResNet` and `CQCC-ResNet` have a significantly smaller t-DCF and EER scores than the baseline algorithms on the development set (*known* attacks) of the logical access scenario. The fusion of the models achieves a *perfect* score of zero EER and t-DCF on the development set. However, in the evaluation set results, our models outperform the baseline models only in the EER of `CQCC-ResNet` and t-DCF score of `MFCC-ResNet`. This highlights the difficulty of generalizing a spoofing detection system to *unknown* attack algorithms.

	Logical Access				Physical Access			
	Development		Evaluation		Development		Evaluation	
Model	t-DCF	EER	t-DCF	EER	t-DCF	EER	t-DCF	EER
Baseline LFCC-GMM	0.0663	2.71	0.2116	8.09	0.2554	11.96	0.3017	13.54
Baseline CQCC-GMM	0.0123	0.43	0.2366	9.57	0.1953	9.87	0.2454	11.04
MFCC-ResNet	0.1013	3.34	**0.2042**	9.33	0.3770	15.91	-	-
Spec-ResNet	0.0023	0.11	0.2741	9.68	0.0960	3.85	**0.0994**	**3.81**
CQCC-ResNet	0.0002	0.01	0.2166	**7.69**	0.1026	4.30	0.1070	4.43
Fusion	0.0000	0.00	**0.1569**	**6.02**	0.0581	2.65	**0.0693**	**2.78**

Table 6.2: t-DCF and EER scores for the different models as measured on the development and evaluation sets for both logical and physical access scenarios.

Nevertheless, our model fusion shows t-DCF $= 0.1569$ and EER $= 6.02$ which are approximately a 25% improvement over the best scores of baseline algorithms.

To provide a better analysis of the performance of our model against both *known* and *unknown* attacks, the t-DCF scores of our models against each attack type are shown in Figure 6.3. Attacks from A01 to A06 are *known* attacks (from the development set) while attacks from A07 to A19 are the 11 *unknown* and two *known* attacks (from the evaluation set). From Figure 6.3, we can see that our models still work well against most attack types except for only two types of the *unknown* attacks, namely A17 and A18. Both A17 and A18 are voice conversion algorithms, where A17 is based on waveform filtering and A18 is based on vocoders. In comparison to the baseline models, the `CQCC-GMM` model also perform poorly on A17(t-DCF=0.9820), which suggest that CQCC is easier to be deceived by waveform filtering based video conversion attacks. Both the `CQCC-GMM` and `LFCC-GMM` work fine on A18, so it is possible that ResNet is more vulnerable to vocoder based video conversion attacks.

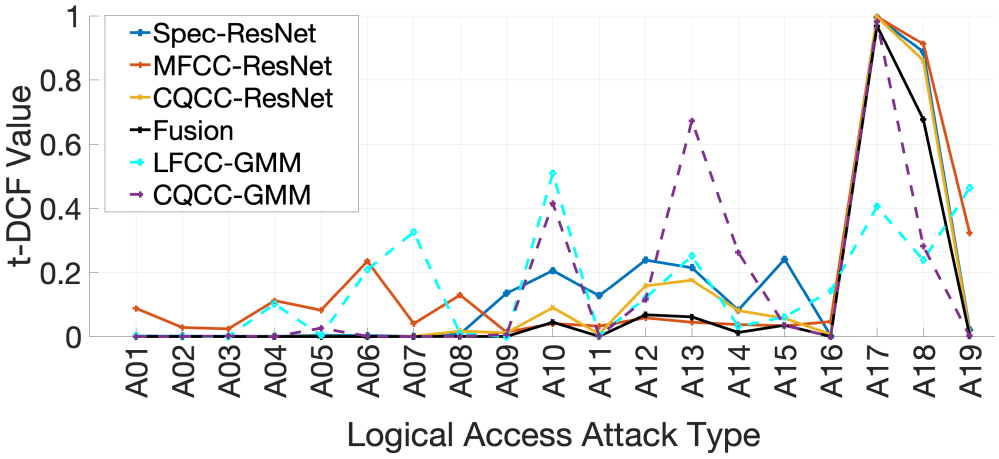


Figure 6.3: t-DCF scores of different models against different attack types (both TTS and VC) in the *logical access* scenario.

6.5.3.2 Physical Access Results

In the physical access scenario, both `Spec-ResNet` and `CQCC-ResNet` have significantly improved both the EER and t-DCF. As shown in Table 6.2, our best single model (`Spec-ResNet`)

is 50% and 60% better than the best baseline results according to the development set EER and t-DCF, respectively. According to the evaluation set scores, `Spec-ResNet` reduces the t-DCF and EER of baseline algorithms by 60% and 65%, respectively. Furthermore, the fusion of our models leads to 71% and 75% improvement.

Table 6.3 provides detailed results of model performance over different replay attack settings. Each setting is named with two letters. The first letter stands for the distance of the recording device from the bona-fide speaker. 'A' means 10-50 cm, 'B' means 50-100 cm, and 'C' means >100cm. The second letter indicates the quality of replay devices, where A means perfect, B means high, and C means low. From the results it is easy to see that, as the distance decreasing and recording device getting better, the anti-spoof task becomes more and more difficult. The worst results are achieved at setting 'AA'. Another thing to notice is that, while `Spec-ResNet` is generally performing better than `CQCC-ResNet` while in some cases like 'BB', 'BC', 'CB', and 'CC', `CQCC-ResNet` outperforms `Spec-ResNet`.

Generally, the system performs better on physical access scenarios that on logical access. This is probably caused by the challenge of generalization, as in logical access, most attacks in the testing dataset are diverse and unknown, while in physical access the features come from the replay channel properties and are easier to learn and generalize.

6.6 Conclusion and Future Work

In this chapter, we presented a novel audio spoofing detection system for both logical access and physical access scenarios. We provide comparisons between the performance of our model combined with three feature different feature extraction algorithms. According to the evaluation dataset scores, against replay attacks, the fusion of our models CM scores improves the t-DCF and EER metrics of baseline algorithm by 71% and 75% respectively. Also, against the TTS and VC attacks, our fusion of models improves the t-DCF and EER metrics by approximately 25% each.

Future work: In the future, we are going to study how to improve the generalization of our model against *unknown* attacks. One possible solution is to employ advanced fusion to build a 'wide-and-deep' network as proposed in [CKH16]. The key idea of this new proposal is to

Attack Type	CQCC-ResNet		Spec-ResNet		Fusion	
	t-DCF	EER	t-DCF	EER	t-DCF	EER
AA	0.2857	10.59	**0.2473**	**9.17**	0.1845	6.78
AB	0.0690	2.57	**0.0638**	**2.22**	0.0468	1.77
AC	0.0464	1.75	**0.0436**	**1.56**	0.0219	0.80
BA	0.1404	5.46	**0.1300**	**4.82**	0.0855	3.29
BB	**0.0295**	**1.18**	0.0374	1.34	0.0230	0.79
BC	**0.0213**	**0.84**	0.0240	0.86	0.0086	0.36
CA	0.1173	4.55	**0.1105**	**4.01**	0.0705	2.71
CB	**0.0266**	**1.00**	0.0342	1.19	0.0171	0.59
CC	**0.0209**	**0.82**	0.0254	0.87	0.0074	0.28

Table 6.3: Detailed comparison between the two best single models and the fusion model in Physical Access scenario under different *replay* attack settings.

concatenate the features from each model's last fully connected layers and use a shared softmax layer as the output layer. This might be able to train the networks to collaborate with each other and achieve a better fusion result.

CHAPTER 7

PhysioGAN: Training High Fidelity Generative Model for Physiological Sensor Readings

In this Chapter, we introduce *PhysioGAN* which is a generative model to produce high quality synthetic datasets of physiological sensors time-series data. *PhysioGAN* can be used to alleviate the privacy concerns that arises while sharing sensor data measurements.

7.1 Background

Improved techniques for training generative models is a rapidly growing area of research. Over the past few years, the machine learning research community has made significant leaps forward towards this goal. This wave of success has been mainly driven by the advent of new training techniques such as the variational autoencoder (VAE) [KW13] and the generative adversarial networks (GAN) [GPM14]. Through GANs and VAEs–as well as their improved versions [SGZ16, BDS18, ROV19]–we are now capable of producing high fidelity, large-scale images with unprecedented levels of quality. GANs and VAEs have also been proven useful in a variety of applications such as generating photorealistic super-resolution images from low-resolution images [LTH17], learning a disentangled latent space representation (which is valuable for content manipulation) [CDH16, HMP17], and generating realistic images from text descriptions [RAY16].

However, a majority of generative model research has focused on training models for images [RMC15, SGZ16, BDS18, ROV19] and, more recently, text datasets [YZW17, JDB18, WW18, ZLZ18]. Only a few pieces of work have studied how to learn a generative model for time series data such as sensor readings with limited success. Data from physiological sensors, e.g., electrocardiograms (ECGs) and fitness tracking sensors, are now prevalent in a lot of applications for health

monitoring and patient diagnosis. A good generative model for physiological sensor readings is an important key for many desired applications in the medical domain. To name a few *potential* applications, GANs have been successfully used to boost the performance of semi-supervised learning classification models that learn from a small set of labeled examples and a larger set of unlabeled examples on image datasets [SGZ16]. In the medical domain, semi-supervised learning is highly desirable since labeling medical sensors readings–e.g., whether an ECG signal segment is normal or abnormal–can be costly and doable only by medical professionals. Further, GANs have been utilized to address concerns of privacy in the context of machine learning. Since machine learning models store information of training data, it has been shown that they can be reverse-engineered by an attacker [FJR15, SSS17] to uncover sensitive information about the training data set. GANs have been used in combination with the differential privacy techniques [DR14] to train accurate models with strong privacy guarantees against this kind of attacks [PAE16]. It has been shown that GANs can produce synthetic datasets that can be used in place of the original real data while still being useful for performing analysis or even training newer models [AS19, JYS18]. These solutions are invaluable for researchers in the medical domain since the privacy-sensitive nature of medical datasets–along with their associated laws and regulations such as the 'Institution Review Board' (IRB)–prevents researchers from sharing the data they collect [SK11]. Unfortunately, the-state-of-the-art methods for training generative models on sensor data readings are still far away from being able to satisfy the requirements of these applications.

The few efforts that have explored training generative models for sensor readings have focused on simple tasks with *toy* datasets rather than meaningful, real-world tasks and datasets. For example, SenseGen [ACS17] uses a recurrent neural network to train a generative model for accelerometer sensor readings using the maximum-likelihood objective. However, SenseGen was only capable of performing *unconditional* generation and, thus, cannot control the attributes of the generator outputs. While the model can be easily extended to support conditional generation–as we will show in our experiments–we find that this training approach is not capable of delivering highly accurate conditional generation results.We also studied the conditional variational recurrent autoencoder (CVRAE) approach for training a generative model for sensor data. The variational autoencoder maximizes an *inexact lower-bound* of the likelihood of training data is generally better at generating

novel samples than traditional autoencoders [KW13]. Nevertheless, the accuracy of the conditional generation is also not high with this training approach. On the other hand, adversarially trained models using the GANs training framework are capable of producing more accurate conditionally generated samples. The RCGAN model [EHR17] has demonstrated how to use this approach to train a recurrent neural network generator with for conditional generation of medical sensors data. However, as we show in our evaluation section, we find that despite having a very high accuracy with the conditional generation, the RCGAN model suffers from a lack of *diversity*. We empirically showed that the RCGAN model produces samples that are very similar and nearly identical within each class. The lack of sample diversity in GANs models is a well-known problem known as *mode collapse* [TOB15]–which is currently being addressed by the machine learning community [SVR17]. Although mode collapse is not unique to generator models that produce sensor data, it is more severe in RGAN because it utilizes a recurrent generator [MPP17]. Since the *discriminator* used for RCGAN training does not provide any penalty when the generator produces repeated samples, the powerful recurrent generator tends to identify which subset of examples are good enough to fool the discriminator and simply repeats them–leading to a lack of sample diversity. Synthetic datasets that have samples suffering from either low generation accuracy or low diversity will have an equally poor performance when used as a replacement for real, *private* data. As such, a new training approach is needed that combines the merits of variational recurrent autoencoder approach with the GANs approach to produce a synthetic dataset that has both high conditional generation accuracy as well as a high diversity of samples.

7.2 Contribution

In this Chapter, we introduce *PhysioGAN*, a *novel* approach for training generative models to produce synthetic sensor readings. *PhysioGAN* consists of three different components: an *encoder*, a *decoder* and a *discriminator*. Together, the *encoder* and the *decoder* form a conditional variational recurrent autoencoder (CVRAE) similar to the CVRAE model–which we consider as a baseline. To improve the accuracy of conditional generation by the CVRAE, we introduce two additional training objectives provided by a *discriminator*: the *adversarial loss* and the *feature matching* loss.

The *discriminator* itself is trained as a multi-class classifier that predicts the class label of real data and attempts to identify “fake" samples produced by the *generator*. To address the issue of *mode collapse*, we introduce an additional *diversity loss* that urges the *generator* to maximize the mutual information between its output and the latent space noise used to generate them. Therefore, the *diversity loss* penalizes the *decoder*–which acts as a *generator*–when it generates identical samples.We improve the training stability by using an annealing approach where the model training cost function *softly* changes from a pure autoencoder loss to the new loss that combines the variational autoencoder loss with the *feature matching loss*, *diversity loss* and *adversarial loss*.

We evaluate *PhysioGAN* against four different baselines: the conditional-recurrent neural network generator (CRNN) (which is an extension of [ACS17] that allows for conditional generation), the conditional variational recurrent autoencoder (CVRAE), the conditional recurrent GANs (RCGAN) [EHR17], and a variation of RCGAN that has a modified *auto-regressive* generator (RCGAN-AR). We conduct our experiments using two real-world tasks and datasets. The first dataset is the “AFib classification dataset" [YPT18], which is a dataset of ECG signal segments. Each segment is labeled as either “Normal" or “Atrial Fibrillation (AFib)", which is a major kind of irregular heartbeat (also known as arrhythmia) that can lead heart failures and possibly death. The second dataset is a human activity recognition (HAR) dataset [AGO13] based on motion sensors such as the accelerometer and the gyroscope commonly found in wearable fitness tracking devices. The HAR dataset represents a dataset for multi-class classification with 6 different kinds of activities that can be grouped into two major groups. Because each group has 3 types of activities that are highly similar to each other and, learning how to conditionally generate samples is a difficult challenge. Further, the HAR dataset introduces the challenge of multi-channel data since each data sample has 6 different axes corresponding to correlated sensor readings. In addition to providing a visualization and qualitative comparison of samples produced by each model, we quantitatively evaluated the 5 models (*PhysioGAN* and the 4 other baselines) based on their *conditional generation accuracy*, the *diversity* of generated samples, as well as the *novelty* of samples to ensure that the model is not *simply* reproducing the same samples as those observed during the training. In addition to those metrics, we use an additional metric to measure the *utility* [EHR17, RV19] of the synthetic dataset produced by each generator. The utility of a synthetic dataset measures how well the dataset

can be used to train a classification model using *only* the synthetic data by validating its performance against the accuracy of a model trained on the *real dataset*.

Contributions Our contributions in this chapter are three-fold. First, we identify common issues that state-of-the-art models currently suffer from by evaluating existing approaches and available baselines of generative models for time-series sensor readings. Second, we provide a novel model training method, *PhysioGAN*, that combines both generative adversarial networks and variational recurrent autoencoders to train a generative model for sensor readings that produces samples with high accuracy and high diversity. Third, we evaluate *PhysioGAN* and other baselines on two different datasets and show that *PhysioGAN* is capable of producing a synthetic dataset that are both accurate and diverse compared to the synthetic datasets generated by other baseline algorithms. Therefore, on each of the AFib classification and HAR classification tasks, we can train classification models using *only synthetic data* produced by *PhysioGAN* with only 10% and 20% decrease in the classification accuracy than models trained on *real data*. the Finally, all of our model implementations and experiments are available as open-source at [1] to promote further research in this important direction of research.

The rest of this chapter is organized as follow: Section 7.3 provides a background on the different kinds of generative models such as GANs and VAEs and also summarizes the related work in training generative models for time-series sensor data readings. Section 7.4 describes our model architecture and training procedure details. Section 7.5 includes the results of our evaluation experiments. Finally, Section 7.6 concludes the chapter and discusses our future work.

7.3 Background and Related Work

In this section, we present the preliminary information necessary to understand the *PhysioGAN* model as well as the works directly related to the scope of this chapter.

[1] `https://github.com/nesl/physiogan`

7.3.1 Background

We first present an overview of the two state-of-the-art frameworks for training generative models: generative adversarial networks (GANs), and variational autoencoders (VAEs).

Generative Adversarial Networks

Generative adversarial networks (GANs) [GPM14] were presented as a framework for training generative models. GANs simultaneously train two separate models through an adversarial game. The first model–called the *generator*, $\mathbf{G}$–learns the distribution of training data. Instead of producing an explicit probability density value, the goal of the *generator* is to directly produce samples from the distribution it learned. The input to the *generator* $\mathbf{G}$ is a noise vector, $\mathbf{z}$, sampled from an arbitrary chosen prior noise distribution $p_{\mathbf{z}}(\mathbf{z})$, i.e., $\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})$. The noise distribution $p_{\mathbf{z}}(\mathbf{z})$ is typically chosen as the standard Gaussian distribution $\mathcal{N}(0, \mathbf{I})$. The *generator* function, $\mathbf{G}(\mathbf{z})$, translates that random noise into fake samples that match the real samples drawn from the training dataset. The second model is referred to as the *discriminator*, $\mathbf{D}$. The *discriminator* distinguishes between the fake samples produced by the *generator* and the real samples from the training dataset. $D(\mathbf{x})$ represents the probability that the input $\mathbf{x}$ is drawn from the real data distribution rather than coming from the generator outputs. The training objective of the *discriminator*, $\mathbf{D}$, is to increase its accuracy in distinguishing between those two sets of samples. On the other hand, the training objective of *generator*, $\mathbf{G}$, is to fool the *discriminator* by producing fake samples that look sufficiently realistic such that it becomes harder for the *discriminator* to identify them. This training procedure can be mathematically formalized as $\mathbf{D}$ and $\mathbf{G}$ playing a two-player min-max game with the following value function $V(\mathbf{G}, \mathbf{D})$:

$$\begin{aligned} \min_G \max_D V(G, D) = &\mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})}[\log D(\mathbf{x})] + \\ &\mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))] \end{aligned} \tag{7.1}$$

Conditional GANs [MO14] extend the original GANs [GPM14] models to generate samples that are conditioned on a given class label attribute $\mathbf{y}$. This can be achieved by feeding $\mathbf{y}$ as additional

input to both the the *discriminator* $\mathbf{D}$ and *generator* $\mathbf{G}$. Therefore, the objective function to the two-player min-max game becomes:

$$\min_G \max_D V(G, D) = \mathbb{E}_{\mathbf{x},\mathbf{y}\sim p_{\text{data}}(\mathbf{x}|\mathbf{y})}[\log D(\mathbf{x}|\mathbf{y})] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[\log(1 - D(G(\mathbf{z}|\mathbf{y})))] \tag{7.2}$$

Over the past few years, many extensions to GANs have been proposed [RMC15, SGZ16, ACB17a], and they have been successfully applied in a variety of domains such as generating realistic super-resolution images [LTH17], image in-painting [DU18], and image synthesis based on text description [RAY16, XZH18]. However, despite the recent success of GANs, successful training of GANs remains a challenge as it requires finding the Nash equilibrium between two non-cooperating players $\mathbf{G}$ and $\mathbf{D}$. The Nash equilibrium happens when the cost of each player is minimized with respect to its own parameters. However, since GANs training is done by applying gradient descent to alternately minimize both the discriminator loss and the generator loss, there is no guarantee that this training approach will converge as minimizing one of the losses may increase the other. Therefore, researchers have suggested various tricks to improve the stability of GANs training such as architecture guidelines for both generators and discriminators [RMC15], mini-batch discrimination [SGZ16], and historical averaging of model weights [SGZ16]. Wasserstein GAN [ACB17a, GAA17] is a recent improvement that replaces the *discriminator* by a critic and uses either weight clipping [ACB17a] or gradient-penalties [GAA17] to enforce a Lipschitz constraint to improve the training stability.

Variational AutoEncoders

In addition to GANs, the variational autoencoder (VAE) [KW13] is another state-of-the-art framework for training generative models. Unlike GANs, where the generator, $\mathbf{G}$, is trained to fool the discriminator, the objective of VAE training is based on maximum likelihood estimation. Intuitively, increasing the likelihood of the generator model to produce training data samples will also increase its capability of generating samples that are similar to the training data.

A major assumption VAEs make is that the data points $\mathbf{x}$ are generated in response to some latent code variable $\mathbf{z}$ that are drawn from a prior distribution $p_{\mathbf{z}}(\mathbf{z})$. According to the law of total

probability, the likelihood of one example $\mathbf{x}^{(i)}$ can be expressed as:

$$p_\theta(\mathbf{x}^{(i)}) = \int p_\theta(\mathbf{x}^{(i)}|\mathbf{z})\, p_{\mathbf{z}}(\mathbf{z})\, dz \tag{7.3}$$

where the model function $p_\theta(\mathbf{x}^{(i)}|\mathbf{z})$ acts as a *decoder* that produces the likelihood of sample $\mathbf{x}^{(i)}$ to be generated according to the latent space value of $\mathbf{z}$. When the decoder is implemented as a neural network, which is capable of being a universal function approximator, It can translate the arbitrarily chosen distribution of the latent space variables into the learned data distribution. The prior distribution of the noise $p_{\mathbf{z}}(\mathbf{z})$ is typically chosen as the standard Gaussian distribution $\mathcal{N}(0, \mathbf{I})$. However, this likelihood integral in equation 7.3 is intractable because there are many possible values of $\mathbf{z}$ and most of them will not have a significant likelihood of producing the given example $\mathbf{x}^{(i)}$. VAEs address this issue by introducing another network called the *inference* or *recognition* model, $q_\phi(\mathbf{z}|\mathbf{x})$, that approximates the true posterior $p_\theta(\mathbf{z}|\mathbf{x})$. VAEs use the expected log-likelihood of training samples under this approximate posterior:

$$\begin{aligned}
\log\big(p_\theta(\mathbf{x}^{(i)})\big) &= \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}\big[\log\ p_\theta(\mathbf{x}^{(i)})\big] \\
&= \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}\big[\log\ \frac{p_\theta(\mathbf{x}^{(i)}|\mathbf{z})p(\mathbf{z})}{p(\mathbf{z}|\mathbf{x}^{(i)})}\big] \\
&= \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}\left[\log\ \frac{p_\theta(\mathbf{x}^{(i)}|\mathbf{z})p(\mathbf{z})}{p(\mathbf{z}|\mathbf{x}^{(i)})}\frac{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}\right]
\end{aligned} \tag{7.4}$$

This equation can be simplified as:

$$\begin{aligned}
\log\big(p_\theta(\mathbf{x}^{(i)})\big) = &\Big[\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}(\log\ p_\theta(\mathbf{x}^{(i)}|\mathbf{z})) \\
&\quad - D_{KL}\big(q_\phi(\mathbf{z}|\mathbf{x}^{(i)})||p_{\mathbf{z}}(\mathbf{z})\big)\Big] \\
&+ D_{KL}\big(q_\phi(\mathbf{z}|\mathbf{x}^{(i)})||p_\theta(\mathbf{z}|\mathbf{x}^{(i)})\big)
\end{aligned} \tag{7.5}$$

where D_{KL} is the KL-divergence function that measures the distance between two probability distributions, i.e.,

$$D_{KL}(p||q) = \int p(x)\ \log\,(\frac{p(x)}{q(x)})\, dx$$

This allows for 7.5 to be rewritten as:

$$\begin{aligned}
\log\big(p_\theta(\mathbf{x}^{(i)})\big) = &\ \mathcal{L}_{ELBO}\big(\mathbf{x}^{(i)};\theta,\phi\big) \\
&+ D_{KL}\big(q_\phi(\mathbf{z}|\mathbf{x}^{(i)})||p_\theta(\mathbf{z}|\mathbf{x}^{(i)})\big)
\end{aligned} \tag{7.6}$$

Since The KL-divergence term is always non-negative, i.e. $D_{KL}(p,q) \geq 0$, the first right hand side term–called the *evidence lower bound* ($\mathcal{L}_{ELBO}$), will constitute a lower-bound for the log-likelihood function, i.e.,

$$\mathcal{L}_{ELBO}\big(\mathbf{x}^{(i)};\theta,\phi\big) \leq p_\theta(\mathbf{x})$$

This term $\mathcal{L}_{ELBO}$ is what the VAE will maximize as increasing a lower bound of log-likelihood will increase the log-likelihood. The difference between the lower bound $\mathcal{L}_{ELBO}$ and the true log-likelihood indicates the error due to replacing the exact intractable posterior $p_\theta(\mathbf{z}|\mathbf{x})$ by an the approximate posterior $q_\phi(\mathbf{z}|x)$–which is tractable to use with help of the '*recognition*' network. The training loss of the the *variational* autoencoder is equal to the negative of the *evidence lower bound*. Therefore, the the training loss is defined as:

$$\begin{gathered}\mathcal{L}_{vae}(\mathbf{x}^{(i)};\phi,\theta) = -\mathcal{L}_{ELBO}(\mathbf{x}^{(i)};\theta,\phi) \\ \Big[\overbrace{-\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}\big(\log\, p_\theta(\mathbf{x}^{(i)}|\mathbf{z})\big)}^{\text{Reconstruction loss}} + \overbrace{D_{KL}\big(q_\phi(\mathbf{z}|\mathbf{x}^{(i)})||p_{\mathbf{z}}(\mathbf{z})\big)}^{\text{Posterior loss}}\Big]\end{gathered} \tag{7.7}$$

The first part of the right-hand side in equation 7.7 represents the log-likelihood of the training sample $\mathbf{x}^{(i)}$ generated by the *decoder* network from a latent space input vector $\mathbf{z}$ sampled from the *recognition* network. This term represents the *reconstruction error* of the training example after being fed through the encoder-decoder networks. The second term of the right-hand side represents the KL-divergence between the distribution of the latent space values produced by the *recognition* network and a chosen prior distribution of the latent space values. Therefore, the KL-divergence term acts as a regularizer that encourages the *encoder* to produces latent space values that match the given prior. Typically, the latent space prior $p_{\mathbf{z}}(\mathbf{z})$ is chosen as an isotropic Gaussian.

Calculating the gradients of the $\mathcal{L}_{ELBO}$ requires backpropagation through the stochastic sampling of the *encoder* output. Therefore [KW13] introduced the technique known as the '*reparamterization trick*' where the approximate posterior sampling $\mathbf{z} \sim q_\phi\big(p_{\mathbf{z}}(\mathbf{z})|\mathbf{x}^{(i)}\big)$ is replaced with a differentiable transformation. The *recognition* network produces the mean, μ, and the standard deviation, σ, of the output distribution. The stochastic sampling of $\mathbf{z}$ can now be approximated using the following equation:

$$\mathbf{z} \sim q_\phi\big(p_{\mathbf{z}}(\mathbf{z})|\mathbf{x}^{(i)}\big) \approx \mu + \sigma \odot \epsilon \tag{7.8}$$

where $\epsilon \sim \mathcal{N}(0, \mathbf{I})$ is an auxiliary noise variable sampled from the standard Gaussian distribution.

7.3.2 Related Work

Generative models for data synthesis has been an active topic of research over the past few years. However, most of of the research focus has been on generating high fidelity images [BDS18, KLA19, ZGM18] and tabular datasets [AS19, PMG18, XV18, CBM17, YVE17]. For time-series datasets, the focus has been on the generation of natural language text [MKB10, YZW17, JDB18, HYL17, WW18] and music datasets [RER18]. Much less effort has been placed towards the generation of physiological sensor readings. In the rest of this section, we briefly discuss the related work on the generation of time-series and sensor readings.

Generative Models for Time-series Generation

The ability to generate high-quality human language text is essential for a variety of tasks such as machine translation and AI chatbots. RNNLM [MKB10] uses the maximum likelihood estimate (MLE) to train a recurrent neural network to predict the next word given the previous word. However, MLE is not a perfect training objective due to the *exposure bias* problem [Hus15] that leads to performance degradation at the generation time due to the discrepancy between the model inputs at training and inference. Scheduled sampling technique [BVJ15] was proposed to increase generation quality. However, it has been found that it will have the negative effect of decreasing sample diversity–leading to another issue known as *mode collapse* [HYL17, TOB15] where the model generates samples that are too similar to each other.

SeqGAN [YZW17] described an approach for training text generative models by modeling the *generator* as a stochastic policy agent of reinforcement learning (RL) which is trained using a policy gradient algorithm [SMS00]. TextGAN [ZGC16] uses a GANs framework to simultaneously train a recurrent neural network (RNN)-based *generator* with a convolutional neural network (CNN)-based *discriminator*. Instead of using the standard GANs training objective for a generator, TextGAN [ZGC16] uses feature matching [SGZ16] that matches the mean and variances of discriminator feature vectors between the real and synthetic sentences. A generative model

that combines both a variational autoencoder and a discriminator based adversarial training was introduced in [HYL17] to generate plausible text sequences whose attributes are controlled by learning a disentangled latent space representation. Our model design is inspired by this work due to the high quality of results it offered in controlled text generation. A more comprehensive literature review on using generative models for text can be found in [LZZ18, ZLZ18].

To generate time-series data outside the domain of text and music, SktechRNN [HE17] uses a recurrent neural network model to draw sketch-based drawings of common objects. Each drawing is represented as time-series of paint-brush strokes.

Generative Models for Sensor Readings Generation

In the following section, we provide an overview of previous work in the domain of generating synthetic sensor data readings.

Maximum Likelihood-based Models. SenseGen [ACS17] uses the maximum-likelihood objective to train a recurrent neural network with mixture density distribution (MDN) outputs to generate synthetic sensor readings. Their framework is trained and evaluated using accelerometer sensor dataset. However, SenseGen is not capable of performing *conditional* generation as it does not provide a mechanism to control the attributes of the generator results. Besides, models trained with only maximum-likelihood objective exhibit exposure bias [Hus15] that reduce the quality of the generated signal because the model is trained to predict the next step of sequence without being encouraged to model the holistic features of the signal. SenseGen [ACS17] did not have an evaluation for the quality or utility of generator results and used only visual quality to demonstrate the success of their model. To the best of our knowledge, there is no prior work in using VAEs to train generative models for sensor readings. VAEs have the nice capability of doing *inference* by *encoding* the input sample into a distributed latent space vector that captures global features of the signal. The ability to do *inference* jointly with *generation* not only improves the quality and diversity of generator results but also can be handy in applications such as sensor data imputation to fill missing segments of input examples.

Adversarially trained models. SensoryGAN [WCG18] uses the GANs adversarial training objec-

tive to train generative models for three kinds of human activities: staying, walking, and jogging. However, their technique requires training a separate generator with a different model architecture for each kind of activity rather than using a *single generator* with a conditioning input. This makes their approach less generalizable for other tasks and datasets. Also, in their experiments, they train models to generate only three human activities (*staying*, *walking*, *jogging*) which are considered *coarsely grained* and strongly dissimilar from each other due to the significant degree of difference in motion intensity. A more solid experiment would be to include activities that are considered similar with only *fine grained* differences such as *walking, walking upstairs, and walking downstairs*, or *normal* vs *abnormal* ECG signals we are doing in our experiments. The RCGAN [EHR17] uses a recurrent discriminator to train a recurrent generative model using the GAN training objective. However, experimentss of [EHR17] were only done on toy datasets and as we show later in our evaluation sections the synthetic samples produced RCGAN suffer from low diversity.

Compared to prior work, our work is the first of its kind that combines variational autoencoder with adversarial networks for the task of *conditional generation* of synthetic sensor readings. By using a novel training objective that combines variational autoencoder and generative adversarial networks, our model is able to generate high-quality synthetic datasets that are both accurate and diverse.

7.4 *PhysioGAN* Methodology

In the following section, we describe our model architecture and training algorithm.

7.4.1 Objective

Given a dataset of N labeled time-series sensor data readings, $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$, Our goal is to build a 'generator model' $\mathbf{G}$ that is capable of *conditionally generating* synthetic real-valued time-series sensor readings. The sensor readings may be multi-dimensional at each time step. For example, each time-step may consist of multiple values measured across the different channels of the same sensor or multiple sensors sampled in time-synchronized intervals. Ideally, the synthetic data

produced by the generator should look realistic and hard to distinguish from the real data sampled from the training set. They should also mimic the same distinguishing features and dynamics as the real data. Therefore, any analytic function computed over the synthetic data should return a value close to the returned value from the same function when computed over the real data. For instance, a machine learning classification model trained on a *synthetic* dataset produced by our generator should yield good accuracy when tested using samples from the *real* dataset.

7.4.2 Notation

Formally, the *generator* function can be defined as:

$$\begin{aligned} \tilde{\mathbf{x}} &= \mathbf{G}(\mathbf{z}, y; \theta) \\ \mathbf{G}(\mathbf{z}, y; \theta) &: \mathbb{R}^{N_z} \times \mathcal{Y} \rightarrow \{\mathbb{R}^{T \times N_d}\} \end{aligned} \tag{7.9}$$

where $\mathbf{z} \in \mathcal{R}^{N_z}$ is an input random noise vector sampled from an arbitrarily chosen prior distribution (e.g., standard Gaussian) used as a source of variation to the deterministic *generator* model function, and $y \in \mathcal{Y}$ is a latent code used to specify the desired attributes of the samples we want from generator output. For example, $\mathcal{Y}$ can be defined as

$$y \in \mathcal{Y} = \{\text{sitting}, \text{walking}, \text{running}\},$$

to represent the activity label when while a human activity classification dataset with those 3 classes of activities. θ represents the set of parameters of the generative models. Each example in the dataset $\mathbf{x}$ is a time-series with T time-steps, i.e.,

$$\mathbf{x} \in \mathcal{R}^{T \times N_d}$$

If we use the subscripted notation $\mathbf{x}_t$ to represent the time-series value at the single time step t, we can write $\mathbf{x}$ as

$$\mathbf{x} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T\}$$

. At each time step, $\mathbf{x}_t$ has N_d real-valued numbers representing the values across the different channels of sensor readings (or the values across different time-synchronized sensors). For example,

given a 3-axial accelerometer motion sensor, N_d will be equal to three–corresponding to the three X, Y, Z axes of the sensor readings. Thus, any $\mathbf{x_t}$ in this context can be represented as

$$\mathbf{x}_t = (\mathbf{x}_{t,1}, \mathbf{x}_{t,2}, ..., \mathbf{x}_{t,d}) \in \mathcal{R}^{N_d} \quad \forall t \in [1, 2, ..T]$$

. To summarize, equation 7.9 indicates that the generator learns how to translate an input noise vector and a conditional label into a time-series of real-valued numbers that looks realistic with respect to real sensor reading samples that match the designated condition label. Given this notation, we can now describe the model structure of *PhysioGAN*.

7.4.3 *PhysioGAN* Model Structure

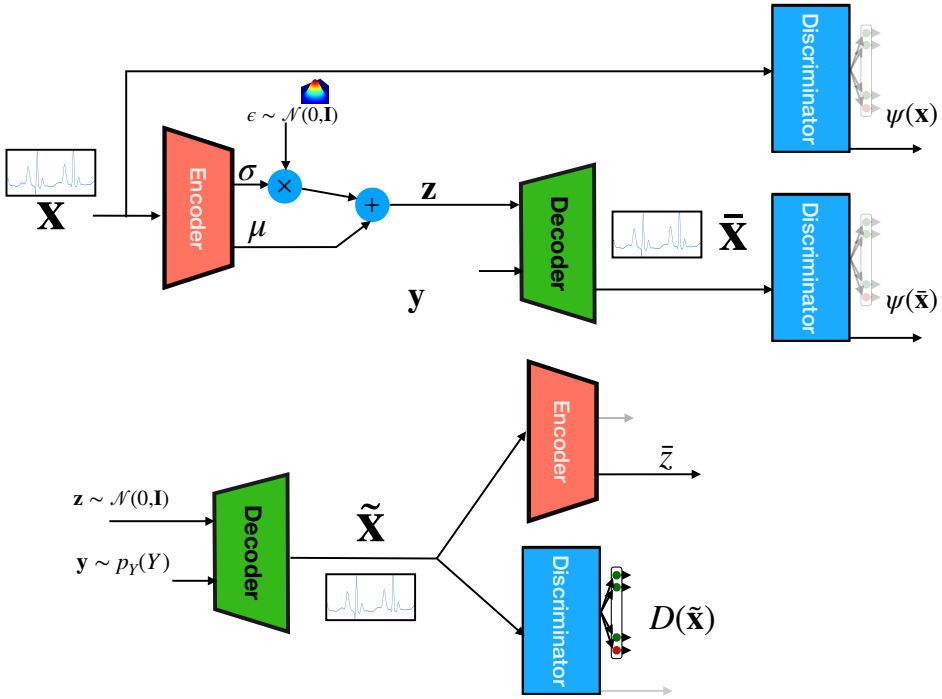


Figure 7.1: Overview of model design for *PhysioGAN*. *PhysioGAN* consist of an *encoder*, *decoder* and *discriminator* components. The different instances of the same component in this diagram are all sharing the same set of weights.

Our model takes advantage and draws inspirations from the recent success Generative adversarial networks (GAN) and variational autoencoders (VAE) have had for the generation of realistic time-

series values in text and sketch-drawing domains [BVV15, HE17, HYL17]. *PhysioGAN* consists of three major components: an *encoder*, a *decoder*, and a *discriminator*. Figure 7.1 depicts the overview of *PhysioGAN*'s model structure and the interaction between the different components. Together, the *encoder* and the *decoder* form a sequence-to-sequence autoencoder that learns to take an input example $\mathbf{x}$ and reconstructs it back into $\bar{\mathbf{x}}$ after it has been mapped into a fixed-length latent space vector $\mathbf{z}$. Both the *encoder* and the *decoder* are implemented as recurrent neural networks (RNNs). We utilize the VAE training objective–which maximizes the evidence lower bound $\mathcal{L}_{ELBO}$ in equation 7.7–since VAEs are better at generating new samples than traditional autoencoders [KW13]. Conditional generation from VAEs can be achieved by adding the attribute labels value y as an additional input to the *decoder*. However, the element-wise *reconstruction error* objective of the VAE does not encourage the *decoder* to maintain the holistic features of the output time-series that are unique for each class label. Therefore, to improve the quality of conditional generation, we pair the VAE *decoder* with a *discriminator* model. We extend the VAE *decoder* training objective–which acts as a *generator*–in two ways. First, we use the GANs training approach to train the *decoder* how to fool the *discriminator* into accepting the samples it generates as *realistic* and by using a multi-class classifier as our *discriminator* it also penalizes the generation of the samples that looks realistic but belong to a wrong class. Second, we extend the *reconstruction error* component of the training objective to include *feature matching* between the features computed by the *discriminator* on both the original input and reconstructed output of the discriminator. Feature matching encourages the decoder to not only preserve the element-wise similarity between autoencoder inputs and outputs but also to preserve the values of the global high-level features that are specific for each class label. While introducing the GAN objective using the *discriminator* into the *decoder* training helps to improve the quality of generated samples. GANs are known to suffer from the issue of '*mode collapse*' [TOB15] where the *generator* identifies which samples were able to fool the *discriminator* successfully and *lazily* generates repeated copies of them. In such a case, the model will exhibit *low diversity*, i.e., the model will generate samples that are too similar to each other. To alleviate this issue and improve the diversity of generated samples, we feed the synthetic samples produced by the decoder from a noise vector $\mathbf{z}$ back into the *encoder*, which reconstructs the noise vector $\hat{\mathbf{z}}$. Then, as an additional objective of the *decoder*

training, we introduce an additional loss term which measures the reconstruction error between $\bar{\mathbf{z}}$ and $\mathbf{z}$. This term encourages the *decoder* to produce samples that are unique for each $\mathbf{z}$ value so that the *encoder* will be able to approximately recover the value of $\mathbf{z}$ as $\bar{\mathbf{z}}$.

In the rest of this section, we describe the details of building individual components of *Physio-GAN* as well as how to train them.

Encoder Design

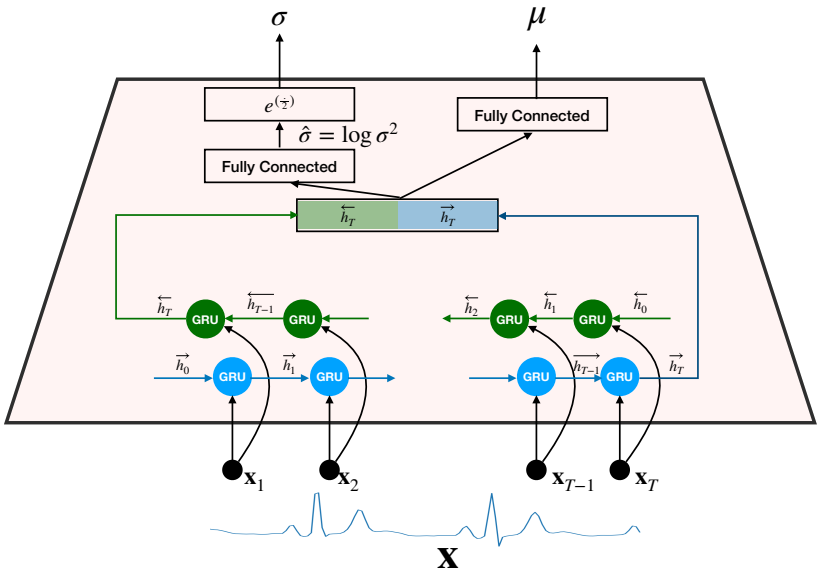


Figure 7.2: Architecture of the *Encoder* model.

The *encoder* translates the input time series sequence into a latent space vector sampled from a posterior distribution, i.e., $\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})$. As shown in Figure 7.2, our *encoder* model is implemented using a bidirectional recurrent neural network that accepts an input time-series sequence $\mathbf{x} \in \mathbb{R}^{T \times N_d}$ and produces a latent vector $\mathbf{z} \in \mathbb{R}^{N_z}$. The bidirectional recurrent neural network consists of two recurrent neural networks that process the input sequence in the forward and backward direction, respectively. Each one of those recurrent neural networks evolves a hidden state vector $\overrightarrow{h}$, $\overleftarrow{h}$ while processing the input sequence time-step by time-step. We use the Gated Recurrent Unit

(GRU) [CVB14, CGC15] implementation of the recurrent unit in our *encoder*.

$$\begin{aligned} \overrightarrow{o}_t, \overrightarrow{h}_t &= GRU(\mathbf{x}_t, \overrightarrow{h}_{t-1}) \\ \overleftarrow{o}_t, \overleftarrow{h}_t &= GRU(\mathbf{x}_t^{(reversed)}, \overleftarrow{h}_{t-1}) \end{aligned} \tag{7.10}$$

where $\mathbf{x}_t^{(reversed)}$ is the input time-series $\mathbf{x}$ after being reversed along the time-axis to be processed in the backward direction. The initial values for the hidden state vectors are zero vectors $\overrightarrow{h}_0 = \overleftarrow{h}_0 = \mathbf{0}$. After processing the whole sequence, we concatenate the final values of the GRU hidden state vectors. The concatenated final hidden state h_T represents a summary of the input sequence. h_T is projected through two fully connected layers to produce two vectors μ and $\hat{\sigma}$ which represent the mean and the logarithm of the variance of posterior distribution computed by the encoder. Each of μ and $\hat{\sigma}$ has a size of N_z. The log-variance output $\hat{\sigma}$ is converted into a non-negative standard deviation by the exponential operation.

$$\begin{aligned} h_T &= [\overrightarrow{h}_T; \quad \overleftarrow{h}_T] \\ \mu &= \mathbf{W}_\mu h_T + b_\mu \\ \hat{\sigma} &= \mathbf{W}_\sigma h_T + b_\sigma \\ \sigma &= e^{\frac{\hat{\sigma}}{2}} \end{aligned} \tag{7.11}$$

Finally, we use the re-parameterization trick [KW13] to approximate the probabilistic sampling of the encoder output $\mathbf{z} \sim \mathcal{N}(\mu, \sigma^2)$ by a differentiable transformation defined upon μ, σ and an auxiliary random variable ϵ.

$$\mathbf{z} = \mu + \sigma \odot \epsilon \quad \text{where } \epsilon \sim \mathcal{N}(0, \mathbf{I}) \tag{7.12}$$

Decoder Design

The decoder model $p_\theta(\mathbf{x}|\mathbf{z}, \mathbf{y})$ translates the pair of latent space noise vector $\mathbf{z}$ and condition label y into a time-series sequence of length T. As shown in Figure 7.3, the decoder is an auto-regressive [Gra13] recurrent neural network that produces an output sequence one-step at a time. At each time-step, the generated output is fed back as an input into the next time step. Therefore, the decoder output at time step t depends also on its own predictions at previous time-steps $< t$ in addition to the values of $\mathbf{z}$ and $\mathbf{y}$. Our decoder is built using a stack of three layers of gated recurrent

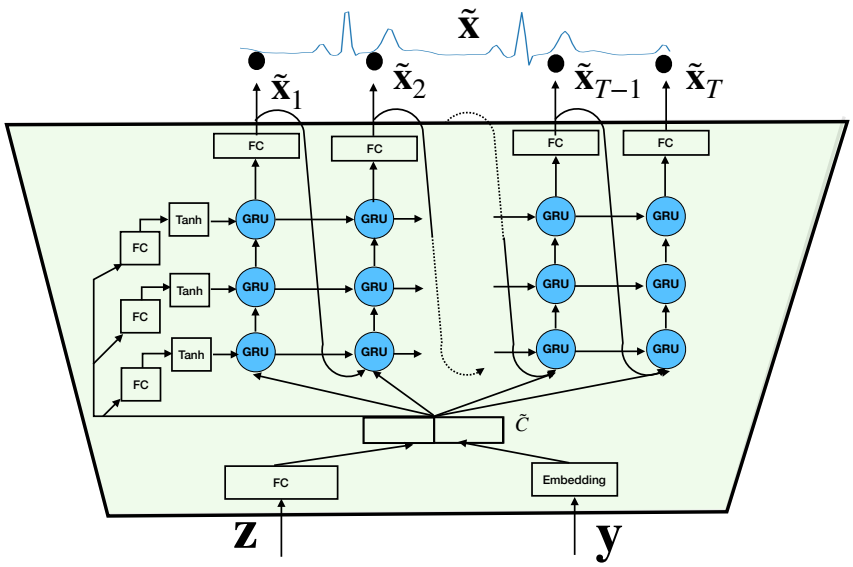


Figure 7.3: Architecture of the *Decoder* model.

units (GRU) neural networks. The vector s denotes the list of hidden states for the GRU units in the three layers and the vector $o^{(dec)}$ denotes the output of the last GRU layer. The initial state of the *decoder* GRU units s_0 is computed from the latent space vector $\mathbf{z}$ as in:

$$s_0 = tanh(\mathbf{W}_s\mathbf{z} + b_s) \tag{7.13}$$

At each time-step, the decoder takes its own generated value from the previous time-step $\tilde{\mathbf{x}}_{t-1}$ along with the current hidden state value s_{t-1} to produce an output $o_t^{(dec)}$ and an updated hidden state s_t. The last GRU layer output o_t is projected through a fully connected layer to produce the final generated value $\tilde{\mathbf{x}}_t \in \mathbb{R}^{N_d}$. We have found it useful to compute a context vector $\tilde{\mathbf{c}}$ by projecting both $\mathbf{z}$ and $\mathbf{y}$ through a fully connected layer and add this context vector the decoder input at each time-step. Using the context vector, $\tilde{\mathbf{c}}$, effectively adds a shortcut between the decoder output at each time-step and the encoder output $\mathbf{z}$ while processing long-sequences.

$$\begin{aligned} \tilde{\mathbf{c}} &= \mathbf{W}_c[\mathbf{z}\,;\mathbf{y}] + b_c \\ o_t^{(dec)}, s_t &= Dec([\tilde{\mathbf{x}}_{t-1}\,;\tilde{\mathbf{c}}], s_{t-1}) \\ \tilde{\mathbf{x}}_t &= W_o\, o_t^{(dec)} + b_o \end{aligned} \tag{7.14}$$

Discriminator Design

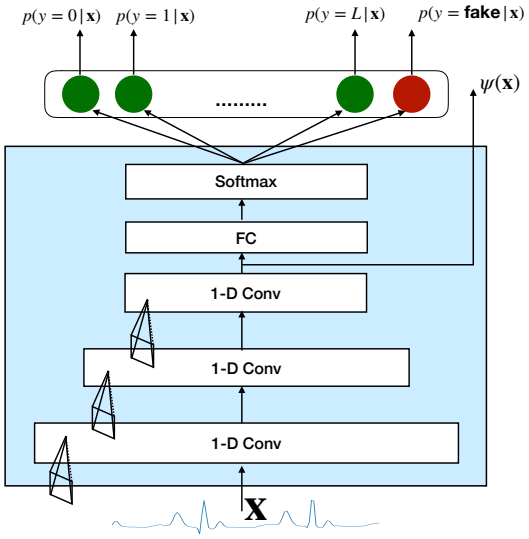


Figure 7.4: Architecture of the *Discriminator* model. The auxiliary output $\psi(\mathbf{x})$ is used to compute the *feature matching* loss.

The *discriminator* D is trained to distinguish between the samples produced by the *decoder* when we feed random noise vectors into it and the examples drawn from the real dataset. Through the feedback it provides for the *decoder* on the samples it generates, it forces the *decoder* to improve the quality of its generated samples until the *decoder* can produce samples that are sufficiently realistic to fool the *discriminator* into accepting them as if they were drawn from the real dataset. As previously suggested in [SGZ16], we use a $L + 1$ multi-class classifier instead of a binary classifier as our discriminator–where L is the number of class labels and the $L + 1$ class is the fake or "generated" class. Therefore, the goal of *discriminator* is to classify all samples produced by the *decoder* as label $L + 1$ and classify samples drawn from the real data as their correct label $\mathbf{y} \in 1, ..., L$. The goal of the decoder while generating a new sample given a latent space vector $\mathbf{z}$ and a condition label $by \in 1, ..., L$ is to fool the *discriminator* into believing this sample is a genuine sample that belongs to the desired class label $\mathbf{y}$.

As such, this modification changes the original conditional GANs min-max equation, shown

earlier in 7.2, into the following:

$$\begin{aligned}\min_G \max_D V(D,G) = &\mathbb{E}_{\mathbf{x},\mathbf{y}\sim p_{\text{data}}}[\log D(\mathbf{x})_{\mathbf{y}}] \\ &+ \mathbb{E}_{\substack{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z}) \\ \mathbf{y}\sim Cat(\{1,..,L\})}}[D(G(\mathbf{z},\mathbf{y}))_{L+1}] \\ &- \mathbb{E}_{\substack{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z}) \\ \mathbf{y}\sim Cat(\{1,..,L\})}}[D(G(\mathbf{z},\mathbf{y}))_{\mathbf{y}}]\end{aligned} \tag{7.15}$$

where, $D(\mathbf{x}) \in [0,1]^{L+1}$.

To realize the *discriminator*, we use the 1-D convolutional classification model shown in Figure 7.2. The *discriminator* consists of three convolution layers followed by a final fully connected layer with a `softmax` output. Each convolution layer has 32 filters with filter size = 3, and applied with stride = 3 and zero padding. Additionally, we regard the output of the last convolution layer (layer 3) as an auxiliary output which is going to be useful to implement the *feature matching* loss we describe in more details in Section 7.4.4.

7.4.4 Model Learning

The training of our model alternates between updating the *discriminator* and updating both the VAE *encoder* and *decoder*. For simplicity, we will refer to the training of both the *encoder* and the *decoder* as the *generator learning* because they are trained together–despite the fact that only the *decoder* is needed for generating samples after the training has been finished.

Further, the functions $\mathbf{E}(.,\phi),\ \mathbf{G}(.,\theta)\ \mathbf{D}(.,\theta^d)$ are used to refer to the *encoder*, *decoder*, and *discriminator*, receptively. Where the symbols ϕ, θ, and θ^d refer to the parameters of the three models, in the same order.

Discriminator Learning

The discriminator is trained to distinguish between the real data samples and those samples generated by the generator. It is a multi-class classifier with a $L+1$ probability distribution output where first $1 <= i <= L$ scores are the scores that the given sample is *real* and predicted as class label i. The last score $L+1$ is reserved to represent the probability that the given sample is '*fake*' or '*generated*'.

The adoption of a multi-class classifier instead of a binary classifier discriminator was proposed earlier by [SGZ16] in the scope of semi-supervised learning to let the *discriminator* provide a class-specific feedback signal to the generator outputs. Additionally, as noted by similar approach was used in [CDH16], this forces the *generator* to increase the mutual information between the GANs synthetic samples and the latent space condition codes. To train the discriminator, we sample batches of labeled examples from the training set, $\{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^{M}$, and create a set of fake examples by feeding into the *decoder* a set of randomly sampled pairs of latent space vectors and condition codes, $\{(\mathbf{z}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^{M}$. The objective of *discriminator* learning is to minimize the following cost function:

$$\begin{aligned} J(\theta_d) = \mathbb{E}_{\mathbf{x},\mathbf{y}\sim p_{\text{data}}}\big[-\log \mathbf{D}(\mathbf{x};\theta^d)_{\mathbf{y}}\big] \\ + \mathbb{E}_{\substack{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z}) \\ \mathbf{y}\sim Cat(\{1,..,L\})}}\big[-\log \mathbf{D}\big(\mathbf{G}(\mathbf{z},\mathbf{y};\theta);\theta^d\big)_{L+1}\big] \end{aligned} \tag{7.16}$$

Generator Learning

In the vanilla VAE model training, the training objective is defined based on the negative value of the evidence lower bound (ELBO), as shown in equation 7.7. Therefore, the original loss for VAE training is composed of two parts: the *reconstruction error* objective and the enforcement of smoothness on the latent space distribution of encoder outputs, making the encoder map examples into smooth regions in the latent space rather than single isolated points. This smoothness makes it more likely to produce realistic samples by feeding into the decoder values of latent space vectors sampled from the prior distribution $\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})$, i.e.,

$$\mathcal{L}_{vae} = \mathcal{L}_{recon} + \mathcal{L}_{posterior} \tag{7.17}$$

To improve the accuracy and diversity of the generated samples, we incorporate the discriminator feedback into the *encoder*, and *decoder* training objective by adding three more terms: feature matching $\mathcal{L}_{\text{features}}$, adversarial loss $\mathcal{L}_{adv}$, and the generator diversity $\mathcal{L}_{diverse}$. We explain the role and definition of each term as well as the equation of the total loss that combines them together.

Reconstruction loss $\mathcal{L}_{recon}$ For any given single example from the training set $(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})$, the reconstruction error measures how well the *decoder* can recover the original input $\mathbf{x}^{(i)}$ after it

has been compressed into the latent space code $\mathbf{z}$ produced by the *encoder*. For sensor data readings with real values, we assume that the *decoder* output is a Gaussian distribution with fixed variance. Therefore, the log-likelihood of *decoder* output is proportional to the mean-squared error between the original reading value $\mathbf{x}^{(i)}$ and its own reconstruction through the autoencoder, i.e., $\bar{\mathbf{x}}^{(i)} = \mathbf{G}(\mathbf{E}(\mathbf{x};\phi),\mathbf{y};\theta)$. Therefore,

$$\begin{aligned}\mathcal{L}_{recon}(\mathbf{x}^{(i)},\mathbf{y}^{(i)};\phi,\theta) &= -\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}(\log\ p_\theta(\mathbf{x}^{(i)}|\mathbf{z},\mathbf{y}^{(i)})) \\ &\propto \left[\frac{1}{T}\ \frac{1}{N_d}\ \left\|\mathbf{x}-\mathbf{G}\big(\mathbf{E}(\mathbf{x}^{(i)};\phi),\mathbf{y}^{(i)};\theta\big)\right\|_2\right]\end{aligned} \tag{7.18}$$

Posterior loss $\mathcal{L}_{posterior}$ the Kullback-Leibler Divergence loss $\mathcal{L}_{kl}$ in Equation 7.7 enforces a prior over the latent space distribution. When this prior distribution of latent space is selected to be IID Gaussian with zero mean and unit variance, the Kullback-Leibler divergence loss can be computed in the closed form [KW13]:

$$D_{KL}\left(p(\mathbf{z}|\mu,\sigma^2)||\mathcal{N}(0,\mathbf{I})\right) = -\frac{1}{2}\frac{1}{N_z}(1+\log\ \sigma^2-\mu^2-\sigma^2) \tag{7.19}$$

where μ and $\hat{\sigma}$ are, respectively, the mean and log-variance of the posterior distribution outputted by the encoder network q_ϕ, as we described in Section 7.4.3.

A common issue in VAE training is suffering from *posterior collapse* where the *decoder* ignores the latent space code $\mathbf{z}$. As reported by previous research [BVV15, **?**], This more likely when the *decoder* is by itself a powerful model such as the RNN *decoder* in our case. The VAE *posterior collapse* happens during the early steps of training when the model finds it is easier to bring down the KL-divergence component of Equation 7.18 rather than the reconstruction error. Therefore, $\mathcal{L}_{kl}$ goes rapidly down to nearly zero, after that the *decoder* is optimized by itself to minimize the *reconstruction error* while ignoring the *encoder* output. Thus, there will be no gradient signal passed between the two models, i.e., the *encoder* and *decoder* have no influence on each other [BVV15]. To address this issue, we use the 'free bits' [KSJ, ROP19] method that modifies VAE loss such that:

$$\mathcal{L}_{posterior}(\mathbf{x},\mathbf{y};\phi) = max\Big(D_{KL}\big(p(\mathbf{z}|\mu,\sigma^2)||\mathcal{N}(0,\mathbf{I})\big)-\delta,0\Big), \tag{7.20}$$

where the KL-divergence is minimized only once until it surpasses a given threshold δ–which we pick as $\delta = 0.1$. To ensure that the model learns how to pack useful information between the

encoder and the decoder, we use a cost annealing scheme [BVV15] that assigns a high weight to the *reconstruction loss* and a nearly zero weight to the *posterior loss* at the early steps of training. Then, we gradually and smoothly increase the weight of *posterior loss* while decreasing the weight of the *reconstruction loss*. This way encourages the model to pack useful information between the encoder and the decoder through the latent space code $\mathbf{z}$. The annealing scheme of the training cost is described in more details at the end of Section in equation 7.25.

Feature matching loss $\mathcal{L}_{feats}$ The original reconstruction loss of the VAE training is based on the element-wise distance between the two vectors of the original input $\mathbf{x}^{(i)}$ and its reconstruction $\bar{\mathbf{x}}^{(i)}$ in the raw data space. This objective, however, leads to outputs that are blurry and lack fine details. Feature matching [SGZ16] encourages the model to reduce the distance between the higher levels features of the original input $\psi(\mathbf{x}^{(i)})$ and those of its reconstruction $\psi(\bar{\mathbf{x}}^{(i)})$. This encourages the model to maintain the holistic attributes of the data points while providing robustness against noise as well as in-variance against transformations such as signal shift. The operator ψ may be provided by either a domain-specific feature extraction algorithm or by taking the values of one of the hidden layers in a classifier. Since our *discriminator* model is trained as a multi-class classifier to predict the correct label of input examples, we reuse the *discriminator* as a feature extractor and, accordingly, the ψ is chosen to be the output values of the last convolution layer in the *multi-class discriminator* model. Therefore, the features reconstruction loss is defined as:

$$\begin{aligned} \mathcal{L}_{feats}(\mathbf{x}, \mathbf{y};\ \phi, \theta) =& \frac{1}{d} \left\| \psi(\mathbf{x}) - \psi(\mathbf{x}) \right\|_2 \\ & \text{where, } \bar{\mathbf{x}} = \mathbf{G}\big(\mathbf{E}(\mathbf{x};\ \phi), \mathbf{y};\ \theta\big) \end{aligned} \tag{7.21}$$

Adversarial Loss $\mathcal{L}_{adv}$ The adversarial training loss of the *decoder* is based on the feedback it receives from the *discriminator* on its generated samples. This directs the *generator (i.e. the Decoder)* to learn how to improve the quality of its generation by matching the class condition label code $\mathbf{y}$, i.e.,

$$\begin{aligned} \mathcal{L}_{adv}(\mathbf{z}, \mathbf{y}; \theta) =& -\log \mathbf{D}\big(\mathbf{G}(\mathbf{z}, \mathbf{y};\ \theta);\ \theta_d\big)_{\mathbf{y}} \\ & \text{where } \mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z}), \mathbf{y} \sim Cat(\{1, .., L\}) \end{aligned} \tag{7.22}$$

Diversity loss $\mathcal{L}_{diverse}$ Training GANs requires finding the Nash equilibrium between two non-cooperating adversaries. However, this process is known to unstable for training GANs as the *discriminator* and *generator* may train in orbits without convergence. This is due to the fact that gradient-descent optimization is not well suited for the task of finding the Nash equilibrium. One common symptom of GAN training failure is *mode collapse*, where the *generator* produces repeated samples that are essentially replicas of instances that were successful in fooling the discriminator. After the *discriminator* identifies that these samples are *fake*, the *generator* will pick another mode to repeat, and so on. This prevents the *generator* from producing samples with high diversity. This issue may be amplified in our model due to the *posterior collapse* where the decoder may depend on itself as a powerful generative model and ignores the latent code $\mathbf{z}$ it receives from the encoder during the training. The *diversity loss* penalizes this situation by the forcing the *decoder* to utilize the latent code vector $\mathbf{z}$. To compute, the *diversity loss*, we use the *encoder* to reconstruct the the latent code vector from samples generated by the decoder. The *diversity loss* is defined as the mean-squared-error between the original latent code $\mathbf{z}$ and its reconstruction by the *encoder* $\bar{\mathbf{z}}$. At the case when the *decode* is suffering from 'mode collapse', it will ignore the latent code and produces identical samples. In such a case, the *encoder* will be unable to recover the latent code from the samples, leading to a high penalty for the *decoder*. The *diversity loss* is defined as:

$$\begin{aligned}\mathcal{L}_{diverse}(\mathbf{z},\mathbf{y};\ \phi,\theta) &= \left\|\mathbf{E}(\mathbf{G}(\mathbf{z},\mathbf{y};\theta);\phi)-\mathbf{z}\right\|_2 \\ &\text{where } \mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z}), \mathbf{y}\sim Cat(\{1,..,L\})\end{aligned} \tag{7.23}$$

Total training cost for generator The summarize, the total training cost of the *encoder* and *decoder* models is defined as:

$$\begin{aligned}J_{total}(\phi,\theta) =& \mathbb{E}_{\mathbf{x},\mathbf{y}\sim p_{\text{data}}}\Big[\eta_t\ \mathcal{L}_{recon}(\mathbf{x},\mathbf{y};\phi,\theta) \\ &+\big(1-\eta_t\big)\big(\beta\mathcal{L}_{posterior}(\mathbf{x},\mathbf{y};\phi)+\lambda_f\mathcal{L}_{feats}(\mathbf{x},\mathbf{y};\phi,\theta)\big)\Big] \\ &+\mathbb{E}_{\substack{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})\\ \mathbf{y}\sim Cat(\{1,..,L\})}}\Big[\big(1-\eta_t\big)\Big(\lambda_a\ \mathcal{L}_{adv}(\mathbf{z},\mathbf{y};\theta)+\lambda_d\ \mathcal{L}_{diverse}(\mathbf{z},\mathbf{y};\theta)\Big)\Big]\end{aligned} \tag{7.24}$$

Where the $\beta=0.2, \lambda_f=1, \lambda_a=1, \lambda_d=0.2$ are weighting coefficients empirically chosen to balance the values of the different loss components. The η_t is a decay function chosen to be the

'inverse sigmoid decay' [BVJ15] with $k = 200$.

$$\eta_t = max\big(\frac{k}{k + \exp(t/k)}, 0.1\big) \tag{7.25}$$

The goal of η_t is to focus the training at the early step on only the *reconstruction loss* $\mathcal{L}_{recon}$. Then gradually, add the other losses and decrease the importance of *reconstruction loss*. We have empirically found this technique improves the stability of training. At the early steps, the output of the *generator* will be too different from the real data and easy for the *discriminator* to distinguish, leading to a saturation of the *adversarial loss*. We avoid this by focusing more on the *reconstruction loss* at the early steps and then introduce the *adversarial loss* after the model has started to produce sensible outputs. Also, the annealing scheme helps to avoid the *posterior collapse* issue we discussed earlier in this Section. The gradual increase of the importance of the *posterior loss*, $\mathcal{L}_{posterior}$, lets the model focuses first on using the latent space value $\mathbf{z}$ to store useful information in order to minimize the reconstruction loss then gradually starts to minimize the *posterior loss*, $\mathcal{L}_{posterior}$, to match the latent space prior distribution.

Model Summary

PhysioGAN consists of three different models: *encoder*, *decoder* and *discriminator*. We extend the vanilla variational autoencoder training objective by including additional terms to improve the quality and diversity of generated samples. The procedure for training of *PhysioGAN* is given in Algorithm 5.

7.5 Evaluation

In the following section, we describe our experiments and evaluation results. We used two different datasets: ECG signal classification, and activity classification from motion sensors. The dataset are described in details in section 7.5.1. We compare the quality of the synthetic data generated by our model against different baselines, which are described in section 7.5.2. Choosing the right metric for evaluating the quality of generative models is still an active topic of research. In addition to the visual quality of samples, most of metrics currently in use are specific to the kind of data being

Algorithm 5: *PhysioGAN* Model Training Algorithm

Require: a dataset of labeled training examples $\mathcal{D}^{train} = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}\}_{i=1}^{N}$.

t = 0
Initialize the weights of *encoder* ϕ, *decoder* θ, and *discriminator* θ_d with random weights.
for `number of training epochs` **do**
    $t = t + 1$.
    Compute η_t according to equation 7.25.
    for each batch $B_d = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^{M}$ in training data $\mathcal{D}^{train}$ **do**
        Sample a batch of latent variables $B_{z,c} = \{(\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z}), \mathbf{y} \sim Cat(\{1, .., L\}))\}$.
        Use the training data batch B_d and the latent variables batch $B_{z,c}$ to update the *discriminator* weights θ_d to minimize the cost given in equation 7.16.
        Use the training data batch B_d to compute equations 7.19, 7.20, and 7.21.
        Sample another batch of latent variables $B'_{z,c} = \{(\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z}), \mathbf{y} \sim Cat(\{1, .., L\}))\}$.
        Use $B'_{z,c}$ to compute equations 7.22, and 7.23.
        Update the the *encoder* weights ϕ and *decoder* weights θ to minimize the total cost given in equation 7.24.
    end for
end for

generated such as the Inception Score [SGZ16] and The Fréchet Inception distance (FID) [HRU17] of image generation, and the perplexity score [CBR98] for text generation. In this chapter, we evaluate the quality of generated sensor data using metrics that measure their conditional generation accuracy score (Section 7.5.4), diversity of samples score (Section 7.5.5), and novelty of samples (Section 7.5.6). Besides, we use an application-specific metrics of the overall quality of the generated data. An evaluation criteria, we use the *synthetic dataset utility* (Section 7.5.7) which measures how well suited are the *generated samples* to be used for training classification models that are evaluated on *real* samples test data.

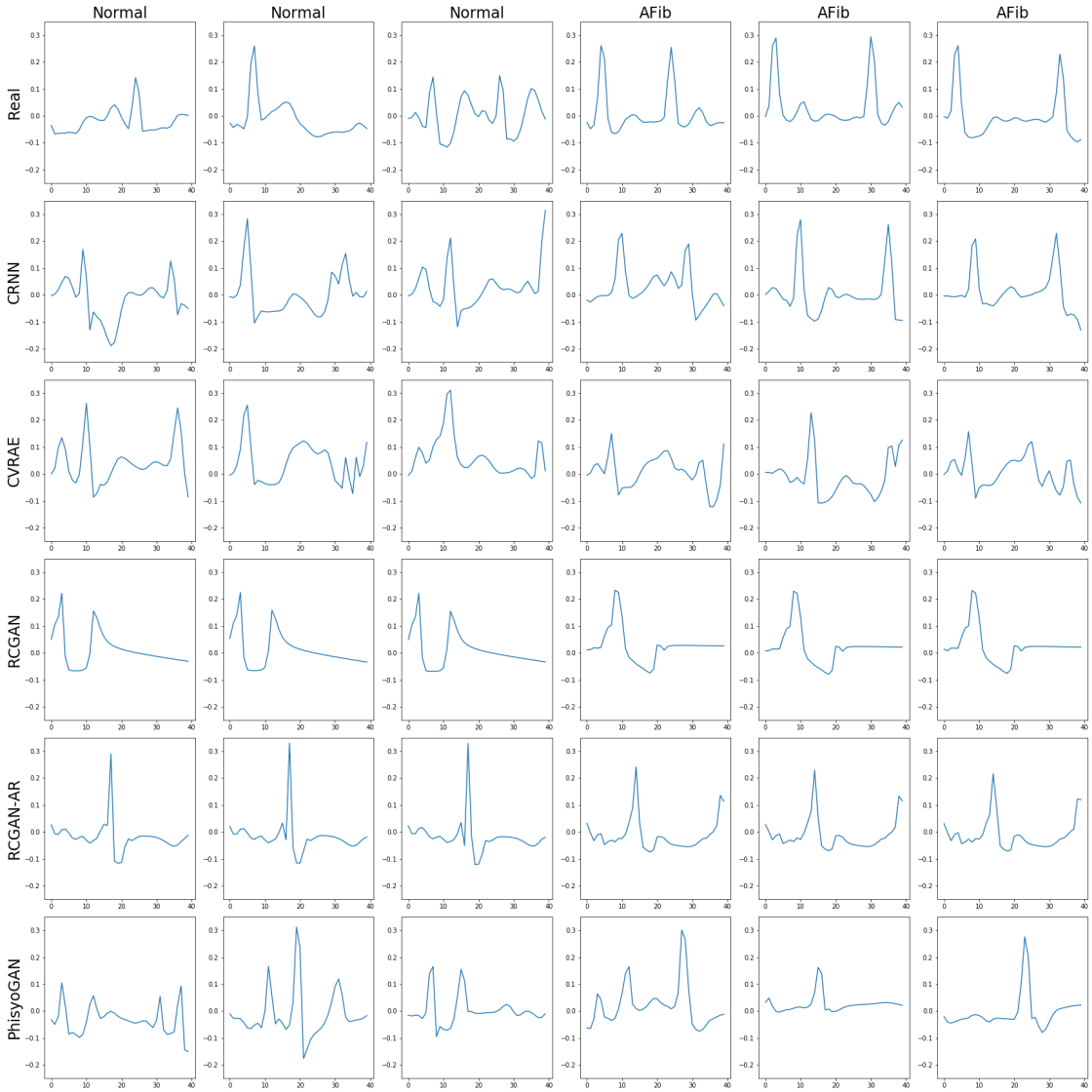


Figure 7.5: Random samples of the real data (top row) and synthetic data (bottom rows) generated by *PhysioGAN* and other baseline models the AFib classification ECG dataset. The title of each column indicate the class label of the samples.

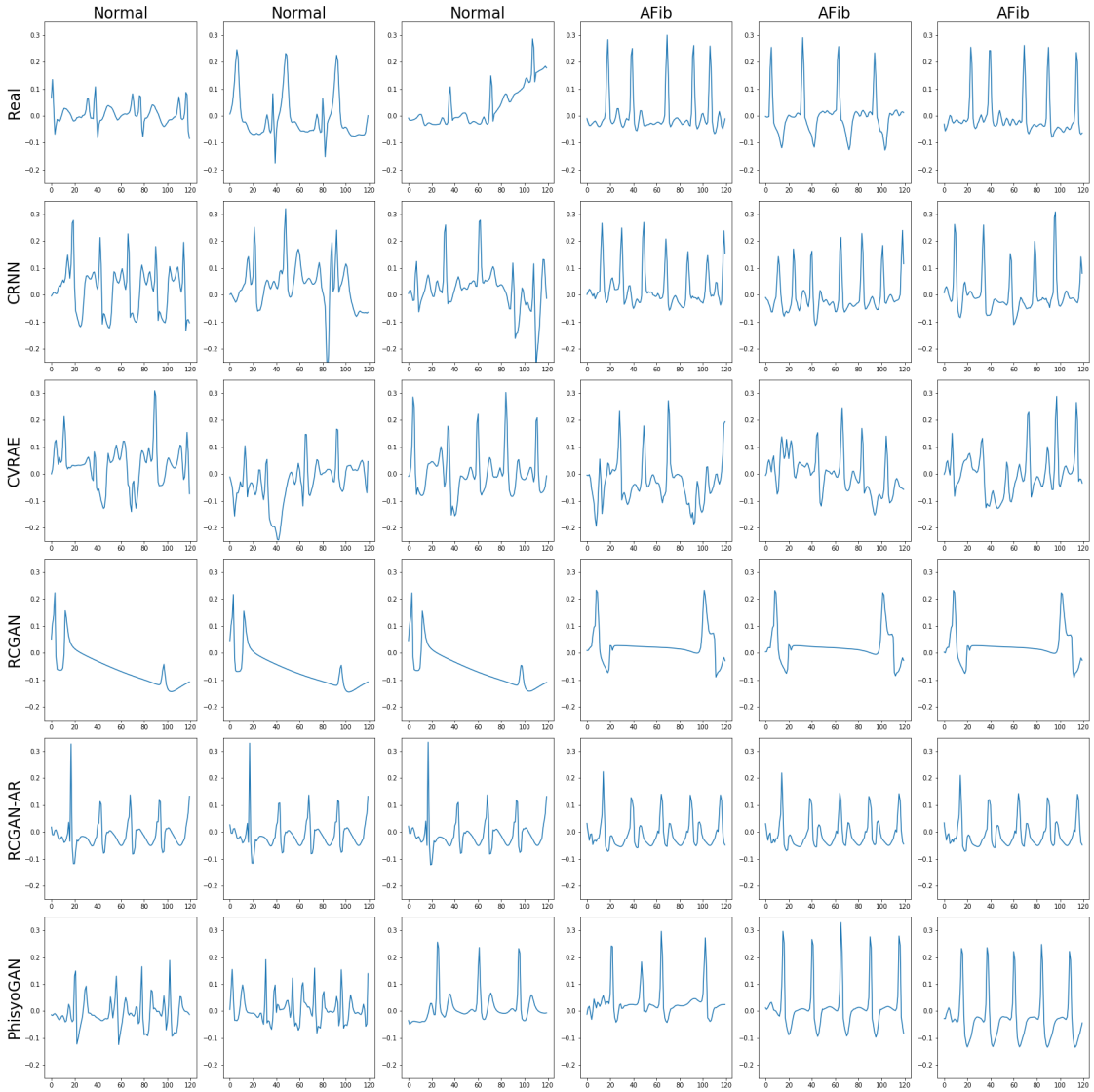


Figure 7.6: Random samples of the real data (top row) and synthetic data (bottom rows) generated by *PhysioGAN* and other baseline models the AFib classification ECG dataset. Each sample has 120 time-steps.The title of each column indicate the class label of the samples.

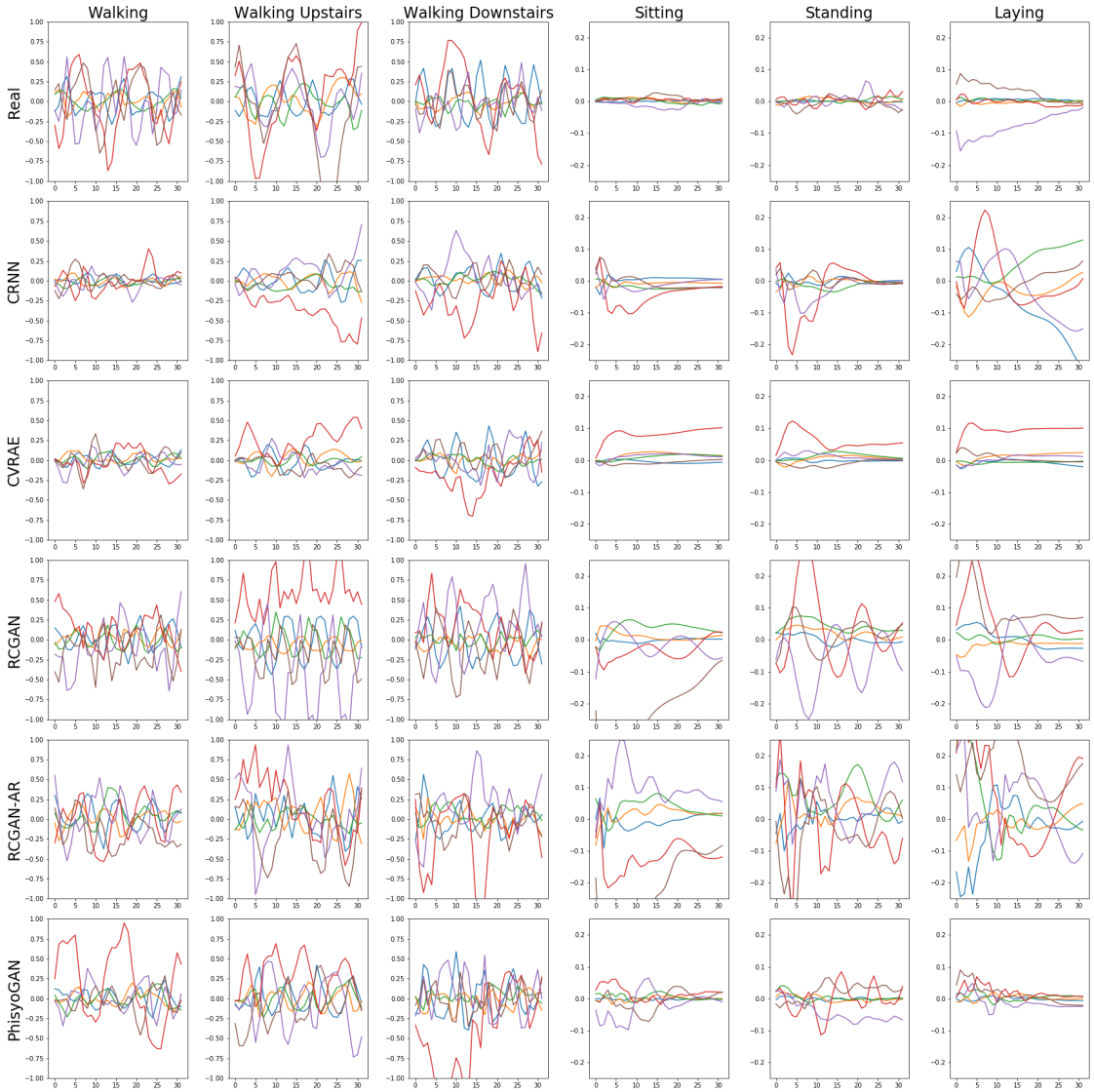


Figure 7.7: Random samples of the real data (top row) and synthetic data (bottom row) generated by *PhysioGAN* and other baseline models on the HAR dataset. The title of each column indicate the class label of the samples.

7.5.1 Datasets

ECG dataset for AFib classification:

Atrial Fibrillation (AFib) is an irregular heartbeat (arrhythmia) disorder. It is considered the most common arrhythmia type, occurring in 1-2% of the general world population and leads to a significant increase in the risks of death, strokes, hospitalization, and heart failure [ECM10]. The Electrocardiography (ECG) signal is considered the most common method for arrhythmia classification. The arrhythmia detection dataset of [YPT18] contains 1000 fragments of ECG signals from 45 persons taken from the MIT-BIH Arrhythmia Database. The dataset fragments correspond to 17 classes including the *Normal Sinus Rhytm* (NSR), the *Atrial Fibrillation* (AFib) and 15 others. In our study, we focus on learning how to generate only (NSR) and AFib classes because they are the most important rhythms and because the other rhythms had a significantly smaller number of examples in the dataset. Therefore, we use the 418 examples that correspond to NSR and AFib classes. Each sample corresponds to 10 seconds of ECG samples recorded at 300 Hz sampling rate. For computational efficiency, we sub-sample the signal to 30 Hz. We split the data into train and test subsets using 75%, 25% split ratios. We train a recurrent neural network-based classification model on the dataset, after sub-sampling, which achieves 97.14% classification accuracy, which is on-par with the state-of-the-art classification accuracy reported by [YPT18]. Random samples of the dataset are shown in the top row of Figure 7.5.

Motion Sensors for Human Activity Recognition

Human activity recognition is important for many reasons, such as elderly fall detection [CKC06] and the assessment of Parkinson disease patients [ABN19]. Activity recognition in wearable devices relies on using the embedded motion sensors such as accelerometer and gyroscope. The UCI *human activity recognition* HAR dataset [AGO13] includes 10,299 examples collected from smartphone attached to the waist of 30 volunteers while performing six different activities: walking, walking upstairs, walking downstairs, sitting, standing, and laying. The dataset is split into training and test sets using 70%, 30% split ratios. We trained a recurrent neural network-based classification model on the dataset, which achieves 89.74% test set classification accuracy. We use this dataset as

an example for learning how to generate a *multi-class* and *multi-dimensional* (each time step has six values corresponding to the X, Y, Z axis values for each of the accelerometer and gyroscope sensors) time-series data. Samples of the dataset are shown in the top row of Figure 7.7.

Notably, the dataset we have used in our experiments is considered more challenging than those used in the original experiments of previous work. For example, [ACS17] did not perform conditional generation or utility evaluation. [EHR17] performed experiments on only toy dataset (e.g., sin-waves) and short low-frequency frequency simple classification tasks. Likewise, [WCG18] only studies the generation of accelerometer data while considering only highly dissimilar classes (e.g., only *walking* vs. *standing*) while we consider the more challenging case of generating six classes including classes are highly similar to each other (e.g., *walking*, *walking upstairs*, and *walking downstairs*).

7.5.2 Baseline algorithms

In comparison to the vast amount of research done on image and text generation, significantly less success has been made towards the conditional generation of high-quality synthetic sensor dataset. Among the notable efforts in this space that we are aware of is the work of [ACS17] trains a maximum-likelihood based recurrent neural network for *unconditional* generation of accelerometer sensor readings. Both the work of [EHR17] and [WCG18] uses adversarial training to train a recurrent neural network of producing real-valued time-series values. We include those methods as baselines and compare their performances against PhisyoGAN according to the measures of condition accuracy, diversity, and novelty and synthetic data utility.

We use the following baseline models:

- **CRNN** *(Conditional RNN-Model)*: This is an auto-regressive [Gra13] recurrent neural network model similar which is trained by maximizing the likelihood of predicting each next time-step values given the previous values. This baseline can be considered as an extension to [ACS17] with the additional support of performing conditional generation which was achieved by conditioning the RNN at each step on the latent code, $[z, c]$, in the same way as our decoder introduced earlier in equation 7.14.

- **CVRAE** *(Conditional Variational Recurrent Auto-Encoder)*: This is a conditional variational autoencoder with a recurrent encoder and recurrent decoder. The architecture of *encoder* and *decoder* were same as those used in our model. But the training objective is different. The training objective used for *CVRAE* is the vanilla conditional VAE training loss introduced earlier in 7.18.

- **RCGAN** *(Recurrent Conditional GAN)*: This model mimics the structure and training method of the conditional recurrent generative model introduced in [EHR17]. It consists of a recurrent neural network generator which is trained with the GANs training objective shown in equation 7.15. Notably, this model is not auto-regressive and the RNN input at each time-step dependent only on the latent space codes $[z, c]$ but not on the previous predictions made by the generator.

- **RCGAN-AR** *(Recurrent Conditional GAN-Auto-Regressive)*: This baseline extends the **RCGAN** model by introducing feedback connections that go from the generator output at one time-step to its input in the next time-step. Therefore, the generator behaves exactly like our decoder, shown in equation 7.14, but it is still trained with the GANs training objective 7.15.

To make a fair comparison between all models, the architecture of *CRNN*, *RCGAN*, *RCGAN-AR* is identical for the architecture of our *decoder*, described earlier in Section 7.4.3. While the baseline *CVRAE* has the same architecture of its encoder and decoder as the encoder and decoder of *PhysioGAN*. Also, models that required a *discriminator* for adversarial training (*RCGAN*, *RCGAN-AR*) were trained using the same multi-class convolutional discriminator that used to train *PhysioGAN*, which we described earlier in Section 7.4.3. Therefore, the five models are only different in their model training technique.

7.5.3 Evaluation Results

Figures 7.5, 7.6, and 7.7 provides a visual illustration of randomly selected samples from the real data and randomly selected samples of the synthetic data generated by *PhysioGAN* for each class of the ECG AFib classification and the human activity recognition (HAR) classification datasets,

respectively. Since it is more challenging to rely on visual inspection of sensor measurements than images and text as an evaluation metric. Therefore, in the rest of this section, we introduce test results that measure the different aspects of synthetic data quality: accuracy, diversity, and novelty of the synthetic samples. In addition to the aforementioned evaluation criteria, We also conduct additional task-specific metrics: the *synthetic dataset utility* which evaluates the quality of the synthetic dataset to replace the *real training data* to train classification models on them.

7.5.4 Conditional Generation Accuracy Score

Model	HAR Dataset	AFib Dataset
Real Data	89%	97%
CRNN	76.0%	67%
CVRAE	72.0%	67%
RCGAN	100%	100%
RCGAN-AR	82%	100%
PhysioGAN	90%	94%

Table 7.1: The *conditional generation score* of synthetic dataset produced by each generative model. The first row indicates the accuracy of the *oracle* model which is trained a training dataset from the real data.

The *conditional generation accuracy score* evaluates the rate by which the generated sensor readings match the class label that the generator was conditioned upon to generate those samples. To compute the score value, for each dataset, we train a high accuracy model on the *real dataset* and use this model as a *oracle* that predicts a classification label of each synthetic sample. To evaluate each generative model, we generate a large set of synthetic samples (with size = 10 times the size of

the real training data) produced by that model and use the *Oracle* model to predict a label for those samples. The rate by which the *Orcale* predictions matches the *condition* code of the generated samples represents the accuracy of conditional generation. The results of *conditional generation score* are shown in 7.1.

The result from Table 7.1 that the models trained with adversarial training (i.e., RCGAN, RCGAN-AR, and *PhysioGAN*) have a significantly higher *conditional generation score* than models trained with maximum-likelihood (i.e., CRNN, and CVRAE). This indicates that adversarial-trained generative models are more likely to produce samples that will look, according to the *oracle* model, as the class they were supposed to match. However, the *conditional generation score* is not a sufficient metric to assess the quality of the generative models because it does not assess the *intra-class* diversity of generated samples. Neither, it does evaluate the novelty of the generated samples to inspect whether or not the generative model is memorizing samples from training data. Therefore, we introduce two other metrics: the *diversity* and *novelty* scores.

7.5.5 Diversity of Samples Score

Mode collapse is a common pitfall for GANs [TOB15]. It is defined by the case when the *generator* produce synthetic samples that are very similar to each other. On the other hand, we want the *generator* to produce samples that are accurate, diverse, and novel from those in the training dataset. In previous research, [WW18] defined a score metric to evaluate the diversity of generated text. In the same way, we define a *diversity score* of the synthetic dataset according to the equation in 7.26.

$$
\begin{aligned}
Diversity\big(\mathcal{S}^{(i)}\big) =& \frac{1}{\Lambda}\ min\ \Big\{\texttt{DTW}(\mathcal{S}^{(i)}, \mathcal{S}^{(j)})\Big\}_{j=1}^{j=|\mathcal{S}|, j\neq i} \\
\text{Where, } \Lambda =& \frac{1}{|\mathcal{D}|}\sum_{i=1}^{i=|\mathcal{D}|} min\ \Big\{\texttt{DTW}(\cdot^{(i)}, \mathcal{D}^{(j)})\Big\}_{j=1}^{j=|\mathcal{D}|, j\neq i}
\end{aligned}
\tag{7.26}
$$

Given a set of synthetic examples $\mathcal{S}$ and another set of real data examples $\mathcal{D}$, the *diversity score* of an individual sample $S^{(i)}$ is defined as the distance the sample i^{th} sample and its nearest neighbor in the synthetic dataset $\mathcal{S}$. Distances are measured using the dynamic time warping (DTW) [SC07] distance measure, which is a reliable measure of time-series dissimilarity due to robustness to minor translations and variations. The *diversity score* of the whole dataset $\mathcal{S}$ is the average of

the *diversity score* assigned for individual samples. In order to have a normalized score value where, as a reference, the *diversity score* of the original real dataset $\mathcal{D}$ is equal to 1 we divide the*diversity score* of each sample in a synthetic dataset by the normalizer Δ which is the average dynamic-time-warping distance between each example from the real dataset $\mathcal{D}$ and its nearest neighbor from the same dataset.

The results for computing the diversity scores on the synthetic datasets produced by *PhysioGAN* and the baseline generative models are shown in Table 7.2. The result shows that *PhysioGAN* has a significantly higher diversity of generated samples than the other methods trained with adversarial training RCGAN, and RCGAN-AR which reflects how *PhysioGAN* had much less mode collapse then the other models that relied only on the vanilla *adversarial training* objective.

Model	HAR Dataset	AFib Dataset
Real data	1.00	1.00
CRNN	0.43	1.03
CVRAE	0.38	1.01
RGAN	0.27	0.06
RGAN-AR	0.23	0.07
PhysioGAN	1.14	0.87

Table 7.2: The *diversity scores* of synthetic dataset produced by each generative model. The first row indicates the score of the *real* data.

7.5.6 Novelty of Samples Score

Generative models are desired to learn the underlying distribution of the training dataset and produce samples that are both *novel* and *realistic* rather than over-fitting the real dataset set. We extend the

idea of [WW18] to evaluate the novelty of the synthetic samples generated by the models under our study using the *novelty score* shown in 7.27.

$$
\begin{aligned}
Novelty\big(\mathcal{S}^{(i)}\big) &= \frac{1}{\Lambda}\, min\, \big\{\texttt{DTW}(\mathcal{S}^{(i)}, \mathcal{D}^{(j)})\big\}_{j=1}^{j=|\mathcal{D}|} \\
\text{Where, } \Lambda &= \frac{1}{|\mathcal{D}|}\sum_{i=1}^{i=|\mathcal{D}|} min\, \big\{\texttt{DTW}(\mathcal{D}^{(i)}, \mathcal{D}^{(j)})\big\}_{j=1}^{j=|\mathcal{D}|, j\neq i}
\end{aligned}
\tag{7.27}
$$

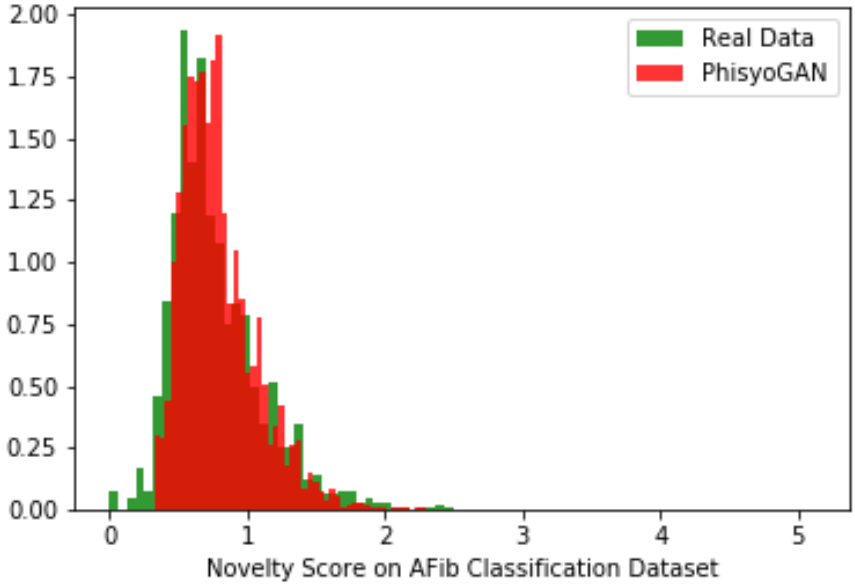


Figure 7.8: Distribution of novelty scores for *PhysioGAN* synthetic samples on the AFib classification dataset.

Given a dataset of synthetic samples $\mathcal{S}$ and the training dataset $\mathcal{D}$ that was used to train the model which produced $\mathcal{S}$, the novelty score of an individual sample $\mathcal{S}^{(i)}$ is measured as its distance to the nearest neighbor from samples in $\mathcal{D}$. Distances are measured using the dynamic time warping [SC07]. Novelty scores are also normalized by dividing their value upon the average distance to nearest neighbor between samples in $\mathcal{D}$ and each other. Therefore, a novelty score equal to zero indicates that the *synthetic* sample is a replica of another sample in the training set. When the novelty score of a *synthetic sample* is equal to one, this indicates that the sample is, on average, as close to samples in the training data as samples from the training data are close to each other. The novelty score of the entire synthetic dataset $\mathcal{S}$ is the mean value of novelty score for each individual sample in $\mathcal{S}$.

Model	HAR Dataset	AFib Dataset
Real data	1.00	1.00
CRNN	1.09	1.33
CVRAE	1.33	1.52
RCGAN	1.67	1.15
RCGAN-AR	1.75	1.00
PhysioGAN	1.35	1.02

Table 7.3: The *novelity scores* of synthetic dataset produced by each generative model. The first row indicates the score of the *real* data.

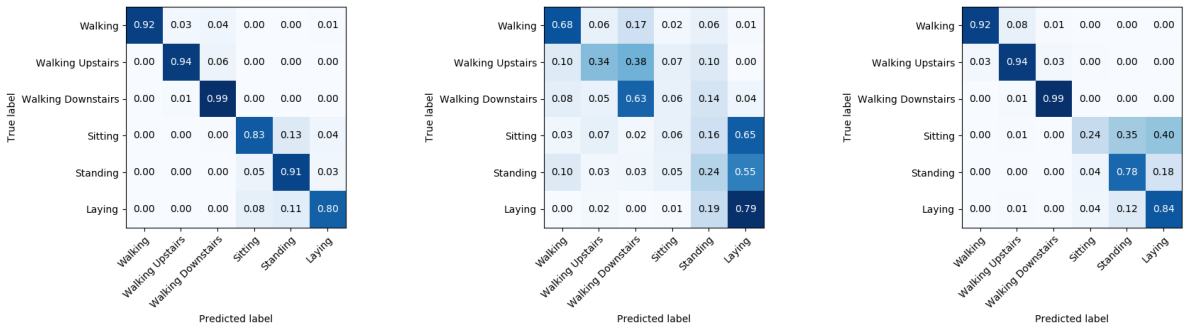
The novelty scores of *PhysioGAN* and the other baseline generative model training methods are shown in Table 7.3. We notice that *PhysioGAN* has less *novelty score* than other methods which may indicate that samples produced by *PhysioGAN* are more similar to the training data samples. To investigate whether or not *PhysioGAN* is memorizing samples from the training data, we compare the distribution of novelty scores for individual synthetic samples produced by *PhysioGAN* on each dataset and the distribution of *novelty scores* for real data in Figure 7.8. From the figure, we conclude that *PhysioGAN* is not memorizing the training data samples because rare samples have *novelty score* close to zero.

7.5.7 Utility of Synthetic data

Since datasets of physiological and medical sensor readings are often considered privacy-sensitive, laws and regulations impose a lot of constraints on how this data can be shared. This introduces a challenge for research teams who collect datasets and are willing to share it with other researchers or the public audience. As an alternative, those researchers may resort into generating *synthetic dataset* that does not belong to real patients but are produced with a *generative model* trained on

real patients data. The utility of synthetic dataset under this situation is reflected by how good are they to be used in lieu of the real data in the downstream task (commonly a classification task).

Rather than evaluating the quality of a conditional generative model based on measuring the aspects of accuracy, and diversity. An alternative way is to measure how good are they to produce data suitable for a downstream task. Even though the downstream is generally unknown at the model training time, a good generative model that learns the generating distribution of the data should be able to produce data that are as good to be used in any downstream task as the training real data. Following this approach, a recently proposed metric for the quality of conditional generative models is the *classification accuracy score* [RV19]. The *classification accuracy score* evaluates conditional generative models by training a classification model using *only* synthetic data and validates the accuracy of the trained model on real test data. This idea was also proposed earlier in [EHR17] under the name of 'TSTR' (Train on Synthetic Test on Real). Despite their success in producing synthetic images that look very realistic and natural, the evaluation of the BigGAN [BDS18] state-of-the-art generative model for ImageNet images shows that top-1 and top-5 accuracy of models trained on synthetic data are 27.9% and 41.6% less than models trained on real data [RV19]. It was also observed that *classification accuracy score* for a variety of generative models does not correlate with other metrics such as the Inception score [SGZ16], and FID [HRU17] which indicates the challenging nature of how to evaluate generative models.



(a) Trained on Real Data (b) Trained on CRNN (c) Trained on *PhysioGAN*

Figure 7.9: The confusion matrix of different classification models trained on real data (shown left), and classification models trained on synthetic datasets produced by CRNN (shown in the middle), and *PhysioGAN* (shown right).

	HAR Dataset		AFib Dataset
Training Dataset	RNN	SVM	RNN
Real Data	97%	83.4%	0.96 (97.0%)
CRNN	32.4%	35.9%	0.46 (54.2%)
CVRAE	46.3%	43.7%	0.50 (48.7%)
RCGAN	30.5%	37.6%	0.52 (55.2%)
RCGAN-AR	29.1%	26.9%	0.48 (50.4%)
PhysioGAN	77.8%	65.1%	0.87 (88.6%)

Table 7.4: Accuracy scores for classification models trained on *synthetic* datasets generated by different generative models. The first row indicates the accuracy of the same model when trained on real data. For the AFib dataset, we use the area-under-the curve (AUC) score because the test dataset is highly imbalanced and show the accuracy between parentheses.

On the ECG dataset, we train a recurrent neural network classification model. While on the HAR dataset, we train two different classification models, a deep-recurrent neural network-based model, and traditional SVM classification model with human-engineered features, selected from best performing work. Results for training those models on both the real data and synthetic dataset are shown in Table 7.4. The first row indicates the accuracy of the models trained on real data for each task. The rows of generative models show the accuracy of classification models trained by their synthetic datasets when evaluated on *real test dataset*. To train models on synthetic data, we sample a set of generator results with size = 10 times the size of the real training dataset. We have noticed that increasing the size of synthetic data will increase the test score of the models they are trained on. The results in Table 7.4 show that *PhysioGAN* have significantly higher utility than the other generative model training methods. Also in Figure 7.9 we show the confusion matrix of classification models trained on real and synthetic datasets, the results show that models trained on samples produced *PhysioGAN* can achieve high accuracy in classification between different classes compared.This is because *PhysioGAN* generates samples that have a balance between their *generation accuracy* and *diversity*.

7.6 Conclusion and Future Work

In this Chapter, we presented *PhysioGAN*, a novel model structure that combines both the variational autoencoder and generative adversarial networks in order to learn how to generate synthetic samples of physiological sensors time-series data *conditioned* on their class labels. We evaluate *PhysioGAN* against different baseline training methods using metrics for the *accuracy*, *diversity*, and *novelity*. On two different datasets, our evaluation results show that *PhysioGAN* is capable of producing high quality *novel* samples that are both accurate and diverse. Furthermore, synthetic datasets produced by *PhysioGAN* can be used to replace the real dataset in training of classification model with a moderate decrease in the classification accuracy. As a result, *PhysioGAN* alleviates the privacy concerns that arise while sharing the privacy-sensitive physiological sensors data.

Future directions of work: In the future, we will explore other applications of the *PhysioGAN* such as learning a *disentangeld representation* of sensor data and use this representation to manipu-

late sensor data to prevent inferences of sensitive attributes while maintaining others non-sensitive inferences. Also, we will study how to train *PhysioGAN* with *differential privacy* to offer stronger and formal guarantees of the limits of information leaked from examples in the training dataset. We touch upon this approach, briefly, in the next chapter.

CHAPTER 8

Differentially Private Release of Private Datasets using Generative Adversarial Networks

In this Chapter, we introduce techniques for using *generative adversarial networks* to produce *differentially private* synthetic datasets that protects personally identifiable information while maintaining a dataset's utility for analysis.

8.1 Background

The NIST: Differential Privacy Synthetic Data Challenge [nis] was a three rounds competition organized the National Institute of Standards and Technology (NIST) during the period between Oct 2018 and April 2019. Participants were asked to create new methods, or improve existing methods of data deidentification while preserving the dataset's utility for analysis. Our team UCLANESL[1] has participated in round 1 and round 3 of the competition and achieved three prizes: 4^{th} place in first round, the 5^{th} place in the third round, and open-source contribution award[2].

The goal of the challenge competition is to produce Differentially private synthetic datasets that preserve the utility of the original datasets on different tasks like using them to train clustering and classification models. The generation of the synthetic dataset should satisfy differential privacy guarantees so that it does not leak entries from the original dataset. Such a method for utility preserving synthetic dataset generation with differential privacy guarantees would be of extreme

[1]Team members are : Prof. Mani Srivastava (Rounds 1 and 3), Moustafa Alzantot (Rounds 1 and 3), Dr. Supriyo Chakraborty (Round1), and Nathaniel Snyder (Round 1).

[2]Prizes announcement: `https://www.nist.gov/communications-technology-laboratory/pscr/team-uclanesl`

value to provide researchers with datasets to work with when the original datasets are sensitive to the privacy of the individuals who have contributed to it or when regulations prohibit the release of these data such medical records datasets.

Our team submission combines together two state-of-art methods for synthetic data generation and accounting of differential privacy loss. Namely, the WGAN [ACB17b] method for training GAN (generative adversarial networks) models and the moments accountant [ACG16] approach for calculating the bound of the privacy loss.

8.2 Contribution

The contribution of this chapter is to describe the algorithms we used in our *award winning* submissions of the NIST differential privacy synthetic data challenge. Furthermore, our implementation of the methods described in this Chapter is available as open source at [3].

The rest of this Chapter is organized as follows: Section 8.3 describes our algorithm and model trainng technique. Section 8.3.4 proves that our training algorithm satisfies the differential privacy requirements. Section 8.4 presents our evaluation results.

8.3 Methodology

Our approach to solving the competition challenge is the following: We use the provided real dataset to train a generative model (GAN) using the improved Wasserstein GAN training algorithm. After training, the generator model of our GAN can be used to produce synthetic dataset by feeding noise samples from a prior distribution (e.g. normal distribution) into it. Compared to the original GAN formulation, the WGAN produces higher quality results and it is much more stable to train. In order to satisfy the differential privacy constraints, the training of our WGAN models is made private by sanitizing the gradient before applying gradient descent steps. Sanitization of gradient occurs in two steps: first, the l_2 norm of the computed gradient is clipped to ensure that the gradient function

[3] `https://github.com/nesl/nist_differential_privacy_synthetic_data_challenge`

has bounded l_2 sensitivity. Second, we apply the Gaussian mechanism by adding Gaussian noise to the sum of clipped gradient. During the model training, we track the total privacy loss by using the moments accountant method [ACG16].

In the rest of this Section, we first present brief background about both Wasserstein GANs 8.3.1 and differential privacy 8.3.2, then we discuss how we utilized differential privacy in the Wasserstein GAN model training in Section 8.3.3.

8.3.1 Wasserstein Generative Adversarial Networks

Deep learning models using neural networks have achieved huge success in many tasks (e.g. image recognition, object detection, speech recognition, etc.). Deep learning models can be categorized as either discriminative models or generative models. Discriminative models attempt to learn the mapping between inputs and the target variables. Formally, they learn to compute $P(y|x;\theta)$. Where x is the model input, y is the output and θ denotes the model parameters. On the other hand, generative models attempt to learn the joint distribution between the input and output variables i.e. $P(x,y;\theta)$.

The first wave of deep learning advancements in the past decade has been focused on discriminative models (e.g. image recognition and speech classification) because they are easier to train and achieve good results on. While training generative models was still considered relatively harder to make progress on. Especially when the input data dimensions are large then training a generative model using maximum likelihood needs huge amounts of training data (due to the curse of dimensionality) and involves approximations of many intractable probabilistic computations that lead to poor quality results.

Generative Adversarial Networks A breakthrough happened in 2014 when researchers introduced a new approach known as *Generative Adversarial networks* (GANs) [GPM14, Goo16]. The goal of the training is to learn a probability distribution $p_g(\mathbf{x};\theta_G)$ that matches the distribution of input training data $p_{data}(\mathbf{x})$. While GANs do not provide an explicit representation of the learned probability distribution, we can use a GAN to generate samples from the distribution it has learned.

This is useful in the case when the goal is to generate synthetic dataset which makes them a suitable technique for the purpose of our competition.

The basic idea of adversarial networks is to maintain two neural network models D, and G that compete against each other. The $D(\mathbf{x}; \theta_D)$ model is termed the *discriminator* and accepts an input that is either coming from the real data distribution $\mathbf{x} \sim p_{data}(\mathbf{x})$ or from the outputs of the generator model $\mathbf{x} \sim p_g(\mathbf{x}; \theta_G)$. The objective of model D training is to increase its accuracy in making a difference between inputs that are coming from the real data distribution and those that are coming from the generator model outputs. In original GAN[GPM14] formulation the output of D model is a probability value in the $[0, 1]$ range.

The generator model $G(\mathbf{z}; \theta_G)$ takes an input noise vector sampled from a given prior distribution (e.g. normal or uniform) to produce fake samples that mimic the samples in the real training dataset. The objective of the model G training is to confuse the model D and limits its ability to make distinction between real examples and generator outputs.

Formally, G and D play the following min-max game with value function $V(D, G)$

$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{data}(\mathbf{x})}[log\, D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_z(\mathbf{z})}[log(1 - D(G(\mathbf{z})))]$$

When both G and D are differentiable functions (e.g. neural networks), the two models are training algorithm alternates between updating D, and G and the parameters of each model are updated using gradient descent and backpropagation.

It has been empirically found [Goo16] that it is better to train model G to maximize $\mathbb{E}_{\mathbf{z} \sim p_z(\mathbf{z})}[logD(G(\mathbf{z}))]$ instead of minimizing $\mathbb{E}_{\mathbf{z} \sim p_z(\mathbf{z})}[log(1 - D(G(\mathbf{z})))]$ in order to provider stronger gradient updates to improve the θ_G parameters in the early stages of training. This technique is known as the log D trick. One important aspect of the GAN training approach: only the model D is trained directly using examples from the real dataset, while G model is trained through the feedback it receives from the D model outptus on the samples it generates.

Wasserstein GAN Although GANs constitute a significant improvement than previous methods for training generative models, researchers faced a major challenge in scaling up GANs and stabilizing their training on big and high dimensional datasets. GANs suffered from instability in

training that prevented the competing models D, and G from reaching the Nash equilibrium of the min-max game so that both model the quality of D and G oscillates.It has also been found that when G model produces samples that are far from the true data samples, the discriminator saturates and results in vanishing gradient that limits the chances of G model to improve. This problem has motivated researchers to develop many alternatives and tricks to make GAN training more stable and improve the equality of results[SGZ16, RMC15, AB17, SVR17, MPP16].

A notable fix to GAN adversarial training stability issues and improve the results quality is the Wasserstein GAN [ACB17b]. In the original GAN formulation, the generator attempts to minimize the Jenson Shannon Divergence (JSD) between p_g and p_d. Rather than minimizing the JSD, Wasserstein GAN changes the GAN training loss so that it minimizes an efficient approximation of the Earth Movers (EM) distance between the the two distributions $p_{data}(\mathbf{x})$, and $p_g(\mathbf{x};\theta_G)$.

The Earth-Mover distance of Waserstein-1 distance is defined as:

$$W(P_{data}(\mathbf{x}), P_g(\mathbf{x};\theta)) = \inf_{\gamma \in \Pi(P_{data}(\mathbf{x}), P_g(\mathbf{x};\theta))} \mathbb{E}_{(x,y)\sim\gamma}\big[\, \|x-y\| \,\big] \,, \tag{8.1}$$

where $\Pi(P_{data}(\mathbf{x}), P_g(\mathbf{x};\theta))$ is the set of all joint distributions $\gamma(x,y)$ whose marginals are respectively $P_{data}(\mathbf{x})$ and $P_g(\mathbf{x};\theta)$. Intuitively, the EM distance is the "cost" of the optimal transport plan between the two distributions where $\gamma(x,y)$ represents how much "mass" must be transported from x to y in order to transform the distributions $P_{data}(\mathbf{x})$ into the distribution $P_g(\mathbf{x};\theta)$.

EM distance has a number of appealing properties than the JSD that leads to an improved and more stable training of the WGAN models. The D model in W-GANs is termed as *critic* instead of *discriminator*, because its output value is no longer required to represent a probability value and therefore no longer constrained to be within the $[0, 1,]$ range. Unlike the JSD distance, EM distance does not saturate when the D model produces good results and that allows WGAN model to train D model till optimality and after that, it can be used to provide useful gradient to train the generator model. Therefore, it is no longer needed to put a lot of effort maintaining a balance between D and G while training which was a reason for a lot of troubles in the original GAN training. Instead, in $WGAN$ the better the critic D, the better gradients it can provide as feedback to train the generator G.

To avoid intractable infimum operation in EM distance definition, WGAN[ACB17b] employs (the Kantrovitch-Rubinstein duality [Vil08]) to compute it with following formula instead:

$$W(P_{data}(\mathbf{x}), P_g(\mathbf{x};\theta)) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x\sim P_{data}(\mathbf{x})}[f(x)] - \mathbb{E}_{x\sim P_g(\mathbf{x};\theta)}[f(x)] \tag{8.2}$$

Therefore, if we have a family of functions $w \in \mathcal{W}$ that are all K-Lipschitz for some K, the following equation becomes equal to Wasserstein distance multiplied by a multiplicative factor K.

$$\max_{w\in\mathcal{W}} \mathbb{E}_{x\sim P_{data}(\mathbf{x})}[f_w(x)] - \mathbb{E}_{z\sim p(z)}[f_w(G(z;\theta))]$$

This equation can be approximated when the neural network parameters w are lying in a compact space. In order to ensure that parameters lie in a compact space, WGAN[ACB17b] enforces this by clamping the model weights to a fixed box (say $\mathcal{W} = [-0.01, 0.01]^l$) after each gradient update.

The min-max game between the two networks becomes :

$$\min_{\theta} \max_{w\in\mathcal{W}} \mathbb{E}_{x\sim P_{data}(\mathbf{x})}[f_w(x)] - \mathbb{E}_{z\sim p(z)}[f_w(G(z;\theta))]$$

The WGAN training algorithm is provided in 6.

In Algorithm 6, f_w is the critic model function with parameters w, and G is the generator function with parameters θ. The critic is trained multiple for multiple steps (n_{critic}) before training the generator in order to reach its optimality and improve the gradients it can provide to improve the generator. The weight clipping of critic parameters w in line 7, ensure that the weights parameters lie in a compact space and the $RMSProb$[HSS12] is an extension to gradient descent that adaptively adjusts the learning rate for each dimension in the weight vector.

8.3.2 Differential Privacy

Differential privacy is a privacy model that provides privacy guarantees for algorithms on aggregate databases. Intuitively, it provides a promise made by the data holder to the data subject: "you will not be affected by allowing your data to be used in the released results of statistical queries no matter what other auxiliary information the attacker may have".

Algorithm 6: Algorithm for training WGAN

Require: : α, the learning rate. c, the clipping parameter. m, the batch size. n_{critic}, the number of iterations of the critic per generator iteration.

Require: : w_0, initial critic parameters. θ_0, initial generator's parameters.

while θ has not converged **do**
  for $t = 0, ..., n_{\text{critic}}$ **do**
    Sample $\{x^{(i)}\}_{i=1}^{m} \sim P_{data}(\mathbf{x})$ a batch from the real data.
    Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
    $g_w \leftarrow \nabla_w \left[\frac{1}{m}\sum_{i=1}^{m} f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^{m} f_w(G(z^{(i)};\theta))\right]$
    $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$
    $w \leftarrow \text{clip}(w, -c, c)$
  end for
  Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
  $g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^{m} f_w(G(z^{(i)};\theta))$
  $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$
end while

Differential Privacy : An randomized algorithm $\mathcal{M} : \mathcal{D}^n \rightarrow \mathcal{R}$ is (ϵ, δ)-differentially private if for all neighbouring databases d, d' such that $||d - d'|| \leq 1$:

$$Pr[\mathcal{M}(d) \in S] \leq e^{\epsilon} Pr[M(d') \in S] + \delta$$

for all $S \in \mathcal{R}$. Typically δ is less than the inverse of any polynomial in the size of the database. If $\delta = 0$, then $\mathcal{M}$ is ϵ-differentially private. The smaller the value of ϵ, the stricter privacy guarantee is offered by $\mathcal{M}$.

Post-Processing property [DR14] if $\mathcal{M} : \mathcal{D}^n \rightarrow \mathcal{R}$ is randomized algorithm taht is an ϵ-differentially private mechanism by using:-differentially private. Let $f : \mathcal{R} \rightarrow \mathcal{R}'$ be an arbitrary data-independent randomized algorithm. Then $f \circ \mathcal{M} : \mathcal{D}^n \rightarrow \mathcal{R}'$ is an (ϵ, δ)-differentially private.

Laplace Mechanism [DR14] Any given function $f : \mathcal{D}^n \rightarrow \mathcal{R}$ can be approximated via an ϵ-differentially private mechanism by using:

$$\mathcal{M}(x) = f(x) + Lap(\frac{\Delta_f}{\epsilon})$$

where $Lap(\sigma)$ is a value sampled from Laplace distribution with zero mean and σ scale, and Δ_f is the sensitivity of f defined as

$$\Delta_f = \max_{\substack{x,y \in \mathcal{D}^n \\ ||x-y||=1}} ||f(x) - f(y)||_1$$

Gaussian Mechanism[DR14] Another alternative to adding Laplacian noise is to add Guassian noise. In this case rather than scaling noise to l_1 sensitivity Δ_f, we instead scale the noise to the l_2 sensitivity $\Delta_2(f)$. Therefore, any given function $f : \mathcal{D}^n \rightarrow \mathcal{R}$ can be approximated via an ϵ-differentially private mechanism by using:

$$\mathcal{M}(x) = f(x) + \mathcal{N}(0, \Delta_2^2(f) \cdot \sigma^2)$$

where $\Delta_2(f)$ is l_2 sensitivity of function f

$$\Delta_2(f) = \max_{\substack{x,y \in \mathcal{D}^n \\ ||x-y||=1}} ||f(x) - f(y)||_2$$

A single application of Guassian mechanism to a function f of l_2 sensitivity $\Delta_2(f)$ satisfies (ϵ, δ) differential privacy if $\sigma \geq \frac{\sqrt{2 \log \frac{1.25}{\delta}}}{\epsilon}$

Sequential Composition and Privacy Accountant [McS09] : The sequential composition property of differential privacy states that for any sequence of computations that each of them is differentially private in isolation will also provide differential privacy. If each of $\mathcal{M}_i$ is ϵ_i differentially-private then the sequence of $\mathcal{M}_i$ provides $\sum_i \epsilon_i$-differential privacy. [McS09] proposes the use of **privacy accountant** to maintain track of the total privacy loss in complex algorithms that consist of individual differentially-private computation steps.

Strong Composition Theorem: The lemma 2.3[BST14, DRV10] can be used to compute the total privacy loss of T steps of adaptive composition. When $\epsilon, \delta' \geq 0$, The class of ϵ-differentially

private algorithms satisfies (ϵ', δ')-differential privacy under T-fold adaptive composition for $\epsilon' = \sqrt{2 * T \ln(\frac{1}{\delta'})}\epsilon + T\epsilon(e^{\epsilon} - 1)$

Moments Accountant Method: Like privacy accountant method, the moments accountant also keeps track of the privacy loss for sequence of computations. However, the moments accountant uses the tail bound of log moments of privacy loss random variable to obtain a more strict bound for privacy loss. For any two neighbouring data-sets $d, d` \in \mathcal{D}^n$, a mechanism $\mathcal{M}$, auxiliary input $\mathbf{aux}$, and an outcome $o \in \mathcal{R}$, the privacy loss at o is defined as

$$c(o; \mathcal{M}, \mathbf{aux}, d, d') \triangleq \log \frac{Pr[\mathcal{M}(\mathbf{aux}, d) = o]}{Pr[\mathcal{M}(\mathbf{aux}, d') = o]}$$

where the $\mathbf{aux}$ input for the k^{th} mechanism $\mathcal{M}_{\|}$ includes the output of all previous mechanisms. Therefore, allowing for composition of mechanisms.

For any given mechanism, the λ^{th} moment $\alpha_{\mathcal{M}}(\lambda; \mathbf{aux}, d, d')$ is the log of the moment generating function evaluated at value λ

$$\alpha_{\mathcal{M}}(\lambda; \mathbf{aux}, d, d') \triangleq log\ \mathbb{E}_{o \sim M(\mathbf{aux}, \mathbf{d})}[exp(\lambda c(o; \mathcal{M}, \mathbf{aux}, d, d')]$$

If we let $\alpha_{\mathcal{M}}(\lambda)$ to be the maximum taken over all possible $\mathbf{aux}$ and all neigbhouring databases d, d'

$$\alpha_{\mathcal{M}}(\lambda) = \max_{\mathbf{aux}, d, d'} \alpha_{\mathcal{M}}(\lambda; \mathbf{aux}, d, d')$$

Then $\alpha_{\mathcal{M}}$ has two important properties:

- **Composability**: suppose a mechanism $\mathcal{M}$ consists of sequence of adaptive mechanisms $\mathcal{M}_1, ..\mathcal{M}_k$ where $M_i : \Pi_{j=1}^{i-1} \mathcal{R}_j \times \mathcal{D} \rightarrow \mathcal{R}_i$, then for any λ

$$\alpha_M(\lambda) \leq \sum_{i=1}^{k} \alpha \mathcal{M}_i(\lambda)$$

- **Tail bound**: For any $\epsilon > 0$, the mechanism $\mathcal{M}$ is (ϵ, δ)-differentially private for

$$\delta = \min_{\lambda} \exp(\alpha_{\mathcal{M}}(\lambda) - \lambda\epsilon)$$

The proof for the two properties is provided in [ACG16].

As a result, we can compute the overall bound $\alpha_{\mathcal{M}}$ by summing the bounds of each step $\alpha_{\mathcal{M}_i}$. Then, we can use the tail bound property to convert the moments bound to (ϵ, δ)-differential privacy guarantee.

8.3.3 Wasserstein GAN Training with Differential Privacy

Now, we discuss about how utilize the *differential privacy* technique to train our WGAN model. Algorithm 7 describes the procedure for training our differentially-private WGAN.

Compared to the original WGAN training algorithm given in 6, we add the following steps:
Compute Per-Example Gradient: Rather than computing the mini batch gradient value, we need to compute the gradient of loss function with respect to the *critic* parameters per each example $x^{(j)}$ in the training data $g_w(x^{(j)})$.
$x^{(j)}$

Gradient Norm Clipping: Since choosing the amount of noise we add while applying the Gaussian mechanism depends on the l_2 sensitivity of the input function. We clip the l_2 norm of each gradient so that the maximum gradient norm becomes c_p.

Adding Noise: We apply Gaussian mechanism to santize the gradients of individual examples by adding noise $\mathcal{N}(0, \sigma^2 c_p^2 \mathbf{I})$ to the sum of clipped per-example gradients. Therefore, the aggregated gradient for each step becomes differential private. In order to achieve (ϵ, δ) differential privacy for each step, it suffice to chose σ to be $\geq \frac{\sqrt{2 \log \frac{1.25}{\delta}}}{\epsilon}$[DR14].

Using Moments Account: We use the moments accountant to track the privacy loss of the whole training algorithm by through composition of bounds of the log of moments of privacy loss per each step.

Algorithm 7: Algorithm for training (ϵ, δ)-differentially private WGAN

Require: : α, the learning rate. c, the clipping parameter. m, the batch size. n_{critic}, the number of iterations of the critic per generator iteration. Noise scale σ, group size L, Gradient Norm bound c_p

Require: : w_0, initial critic parameters. θ_0, initial generator's parameters.

for $t \in [T]$ **do**

  for $i = 0, ..., n_{\text{critic}}$ **do**

    Pick a random sample $L_{t,i} = \{x^{(j)}\}_{j=1}^{L} \sim P_{data}(\mathbf{x})$ from the real data.

    Sample $\{z^{(j)}\}_{j=1}^{L} \sim p(z)$ a batch of samples from prior noise.

    Compute the per-example gradient

    $g_w(x^{(j)}) = \nabla_w f_w(x^{(j)})$ for $x^{(j)} \in L_{t,i}$

    $g_w(z^{(j)}) = \nabla_w f_w(G(z^{(j)}; \theta))$ for $j \in [L]$.

    Clip gradients

    $\bar{g}_w(x^{(j)}) = g_w(x^{(j)}) / max(1, \frac{||g_w(x^{(j)})||_2}{c_p})$ for $x^{(j)} \in L_{t,i}$

    $\bar{g}_w(z^{(j)}) = g_w(z^{(j)}) / max(1, \frac{||g_w(z^{(j)})||_2}{c_p})$ for $j \in [L]$

    Add Noise

    $\tilde{g}_w = \frac{1}{L} \left(\sum_{j=1}^{L} \bar{g}_w(x^{(j)}) + \mathcal{N}(0, \sigma^2 c_p^2 \mathbf{I}) \right) - \frac{1}{L} \sum_{j=1}^{L} \bar{g}_w(z^{(j)})$

    $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, \tilde{g}_w)$

    $w \leftarrow \text{clip}(w, -c, c)$

  end for

  Sample $\{z^{(j)}\}_{j=1}^{m} \sim p(z)$ a batch of samples from prior noise.

  $g_\theta \leftarrow -\nabla_\theta \frac{1}{m} \sum_{j=1}^{m} f_w(G(z^{(j)}; \theta))$

  $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$

end for

8.3.4 Proof of Privacy

Our proof of privacy for algorithm 7 follows from the following:

Differential Privacy Proof for Critic:

To ensure that *critic* is differentially-private, we changed the WGAN training algorithm 6 into algorithm 7 by adding the following steps:

- We apply gradient clipping to bound the l_2 sensitivity of the gradient computed for each example.
- We then add noise to the sum of the clipped gradients using the Gaussian mechanism. In above step, we bound the per-sample gradient to have a maximum l_2 norm equal to c_p. This ensures that the sum of clipped gradients will also be l_2 bounded for any two neighbouring data-sets (using Cauchy-Schwartz inequality) with the norm bound equal to c_p (assuming bounded differential privacy).
- We use the moments accountant method to select the proper scale of Gaussian noise that we add to the gradients.

By bounding the l_2 sensitivity of the gradient updates and applying Gaussian mechanism, each update step becomes (ϵ, δ)-differentially private when the noise scale σ is at least $\frac{\sqrt{2 \log \frac{1.25}{\delta}}}{\epsilon}$[DR14] with respect to the training lot. Since the lot itself is a random sample of the training database, the privacy amplification theorem [KLN11, BKN10] states that each step will be $(O(q\epsilon), q\delta)$-differentially private with respect to the training database where sampling ratio $q = \frac{L}{N}$.

Using the moments accountant[ACG16] we can compute the total privacy loss of the training algorithm to show that our final model is $(O(q\,\epsilon\,\sqrt{T\,n_{critic}}), \delta)$-differentially private. **The proof of this bound can be found in [ACG16]**. It is important to notice that the only restrictions for the results of sequential composition to hold is that the internal randomness of the noise generated at each gradient update step is independent for each repeated gradient sanitization step and that that real training data remains hidden from the adversary [KOV17]. It does not matter even if the

adversary (which is the discriminator in our case) can control which part of the dataset to access or to make any choices based on previous outcomes [KOV17].

When (ϵ, δ)-privacy budget is very small. We prefer to use a privacy accountant based on the strong composition theorem [BST14] rather than the moments accountant [ACG16]. Even though the bounds of strong-composition theorem accumulate faster, it allows us to run training for few steps before running out of privacy budget. **The proof of correctness the privacy-bounds computed by strong composition theorem can be found in [BST14] (Lemma 2.3)**.

Differential Privacy Proof for Generator:
In WGAN training, only the *critic* parameters w are updated directly using the real dataset entries. While the generator parameter θ updates are updated as a function of the critic output values. Therefore, if the critic itself is differentially private, the generator will also be differentially private **(Using the post-processing invariance property of differential privacy[DR14])**.

8.3.5 Tracking Privacy Loss

In our implementation, we use the privacy accountant method of [ACG16], to keep track of the accumulated privacy loss spending after each training step that updates the *critic* weight values. If the accumulated privacy loss spent exceeds the maximum allowed limits of (ϵ, δ), then we abort the training early and don't perform any further updates to the *generator* model.

We employ two state-of-art methods to implement a privacy accountant that tracks the total privacy budget spent after each step of training our models.

- **moments accountant** [ACG16] which tracks privacy loss using the tail bound of the log of moments of privacy loss random variable.
- **amortized accountant** which uses privacy amplification theorem [BKN10, KLN11] and strong composition theorem[BST14, DRV10] to provide an amortized bound of privacy loss.

Both methods are known to provide the state-of-art tight bounds of privacy loss tracking. Although the moments accountant [ACG16] provides a tighter bound. Although the growth of

privacy loss calculated with moments accountant is much less than the growth of privacy loss calculated by strong-composition theorem, We have found that when ϵ_{total} is very small, the bounds computed by strong composition theorem are actually less than the bounds of moments accountant. Therefore, our calculations of privacy loss uses the **amortized accountant** when privacy loss $\epsilon \leq 0.7$, otherwise we use the **moments accountant**.

8.3.6 Data Pre-Processing and Post-Processing

The datasets given to us during the match 3 competition was taken from the Public Use Microdata Sample (PUMS) of the 1940 USA Census Data for the State of Colorado, fetched from the IPUMS USA Website. In total, 662k records and 139 columns of data in CSV format. In this section, we describe how to pre-process the data to prepare them for our model training and how we post-process the model generation results.

8.3.6.1 Pre-processing

In order to prepare the input data for our model training, we apply the following transformations for the input data columns. Each column is considered as one the three following types:

- **Categorical**: Categorical columns are pre-processed by applying a one-hot encoding to their values. The transformation happens through two steps:

    1. Assume the column has N distinct values, then we map the values for the columns to values in the range $\{0, 1, ..., N-1\}$. This process is needed to prepare the values for the next step of one-hot-encoding. The number of distinct values N for each columns is identified using either of the following ways:

        - From the **data-specs.json** metadata file. We use the *maxval* property to identify the maximum value of the column then the number of distinct values is equal to $maxval + 1$.
        - From the codebook **codebook.cbk** file that was included in the competitor pack with description of the possible set of values for each column.

- For the **MIGSEA5** and **OCC** columns, due to the high sparsity of the column values we identify the set of distinct values from the tables available on IPUMS website:
  `https://usa.ipums.org/usa/volii/seacodes.shtml`
  `https://usa.ipums.org/usa/volii/occ1940.shtml`
- For the set of geographic state-dependent columns (**SEA, METAREA, METAREAD, COUNTY, CITY**), we collect the number of distinct values from the input data.

2. Apply one hot encoding to the column value.

- **numeric (*integer*)**: numeric columns are used as it is without any transformations.
- **numeric or not-available**: for columns which have 'not available' (i.e. 99999) codes we transform each column into two columns which are : a boolean column which denotes which the column value is available or not. A numeric column which is equal to the input column value if it is available (i.e. not equal to the 9999 code) or a zero otherwise.

8.3.6.2 Post-processing

After sampling synthetic data from the generator model, we apply post-processing step to format the data in the same format as input data. The post-processing step does not require information from the input data therefore it satisfies the **post-processing in-variance property of differential privacy**.

Post-processing takes place by either or the following ways depending on the column type:

- **Categorical**: For categorical columns, we treat the generator output as a probability distribution over the set of distinct column values. The post processing happens in two steps

  1. We randomly sample a value according the softmax distribution produced by the generator. This produces an integer value in the [0, N-1] range where N is the number of distinct values for the data column.
  2. We map the [0, N-1] values to the set of distinct values for the data column in the inverse

direction of mapping that we did during the pre-processing. *mapping between [0, N-1] range and the set of distinct values was computed in pre-processing step.*

- **numeric (*integer*)**: numeric columns are used as it is without any transformations. We clip the column value to the [0, $maxval$] range where $maxval$ is identified from the metadata **data-specs.json** file.

- **numeric or not-available**: for these columns, the generator model produces a boolean value specifying whether the column has a valid value or *not-available*. If that value is true, then the post-proceed value is $maxval$. The second output is a numeric value. The numeric value is clipped to [0, $maxval$] range and if the boolean value

8.3.6.3 Using public data

As required by the competition rules, our solution does not use any public datasets to training the generator. Once the generator model is trained from input data using differentially-private training algorithm. The generator model is capable of producing new synthetic data.

- We assume that the metadata of each column (i.e. data provided in **data-specs.json**) such the maximum value of each column are to be considered public information.

- Also we also consider as public information, the **codebook.cbk** provided in the competitor pack and we use it to identify the set of possible values for categorical columns while doing data pre-processing.

- For the **MIGSEA5** and **OCC** columns, we identify the set of possible values (*but not the column values distribution*) from IPUMS website : `https://usa.ipums.org/usa/volii/seacodes.shtml` and `https://usa.ipums.org/usa/volii/occ1940.shtml`.

- Since the final testing dataset will be for different states other than Colorado which was used during the provisional testing phase, As posted in the competition forum (`https://apps.topcoder.com/forums/?module=Thread&threadID=935216`) for the geographic

state-dependent columns (**SEA**, **METAREA**, **METAREAD**, **COUNTY**, **CITY**), we collect the set of distinct values for each of these columns from the input data, but we don't collect any information about the distribution of values.

8.3.7 Running Our Solution

To promote further research in this important area, we made all our experiments and implementation code publicly available as open-source at `https://github.com/nesl/nist_differential_privacy_synthetic_data_challenge`. Our implementation uses the `TensorFlow` open-source deep learning framework which is more efficiently run using GPU enabled machines using. It can be also run in a container using `NvidiaDocker`. The runtime is signficantlly improved when running on a GPU- powered machine. It has been tested to work successfully on a machine with Titan X GPU.

Steps to install `NvidiaDocker` can be found in `https://github.com/NVIDIA/nvidia-docker`

After unziping the given zip file. First, build the docker image.

```
docker build -t uclanesl_wgan .
```

then, run the image while giving the appropriate argument values.

then, run the image while giving the appropriate argument values.

```
docker run --runtime=nvidia -v $(pwd):/x uclanesl_wgan /x/data.
    csv /x/output.csv /x/data-specs.json  epsilon [delta]
```

where

- **data.csv**: input dataset file.
- **output.csv**: output file.
- **data-specs.json**: metadata file.
- **epsilon** ϵ privacy budget.

- **delta**: **(optional)** value of δ parameter. If it is not given, then δ is chosen to be $\frac{1}{n^2}$ where n is the size of private dataset.

8.4 Evaluation Results

8.4.1 Datasets

The competition dataset was taken from the Public Use Microdata Sample (PUMS) of the 1940 USA Census Data for the State of Colorado. In total the dataset had 662,000 rows and 139 columns.

During the final evaluation rounds, we submitted our code for final scoring where the organizers used other datasets, *unknown to us* to evaluate our submission.

8.4.2 Evaluation Metrics

The utility of synthetic datasets is determined based on three scoring methods. The three scores was later averaged together to produce the final score. The description of scoring methods is given below:

8.4.2.1 Scoring Method 1

Scoring method one is based on the difference of density distributions for the original and privatized datasets. It performs bucketizaiton of different sets of 3 randomly selected columns. The absolute difference between density distribution of values in the *original* and *privatized* dataset is computed. Since density distribution is between 0 and 1, the absolute difference s will be between 0 and 2. This absolute difference value s is transformed into the score S which is in the range [0, 1000,1000] using the equation

$$S = 1000000 * (1 - \frac{s}{2})$$

A set of 100 tests is created (by choosing different sets of 3 columns), and the final score is the average of scores obtained from individual tests.a

8.4.2.2 Scoring Method 2

To compute the score of scoring method 2, A set of tests are generated. Each test consists of different columns selected at random with each column has 33% chance to be selected. To conduct the test, the scoring code will pick a random subset of possible values for each column included in the test. A dataset record will satisfy a test rule, if it has records that are within the ranges defined by the columns of all columns in a test set. The tests are generated guarantees that original dataset always have at least one record that satisfy each test.

The way tests are generated guarantees that in the original dataset there was at least a single record matching test rules.

The score of method two for the i-th test is defined on the missmatch between the number of rows satisifying the test from the *original* dataset $f_{o,i}$ and those satisfying the test from the *privatized* dataset $f_{p,i}$. The missmatch score is computed according the the following equations:

$$\begin{aligned} d_i =& \ln(\max(f_{p,i}, 10^{-6})) - \ln(f_{o,i}) \\ \delta =& \sqrt{\frac{1}{N}\sum_{i=1}^{N} d_i^2} \\ score =& 10^6 \ \max\left(0, 1 + \frac{\delta}{\ln(10^{-3})}\right) \end{aligned}$$

where $N = 300$, the total number of tests.

8.4.2.3 Scoring Method 3

Scoring method 3 was calculated based on comparing the gender pay gap for each city in the original dataset. The first component is based on the mean-square deviation between Gini indices obtained for the original and privatized dataset, averaged over the cities present in the original dataset. The second component ranks the cities by the gender pay gap, and calculate the score component based on the mean-square deviation between the resulting city ranks in the original and privatized datasets. The overall score from the third method is the average between these two score components and normalized to the [0; 1 000 000].

8.4.3 Evaluation Results

	Score 1	Score 2	Score 3	Average
No Privacy	550679.36	467903.27	690474.40	569685.68
$\epsilon = 10.0$	674409.70	649732.57	704674.97	704674.97
$\epsilon = 3.0$	637677.48	608017.73	738434.54	661376.58
$\epsilon = 1.0$	636014.04	603058.82	805215.98	681429.61
$\epsilon = 0.3$	457963.07	494882.80	814078.46	588974.78
$\epsilon = 0.1$	342467.27	355310.08	753246.78	483674.71

Table 8.1: Utility scores of the synthetic datasets gen different levels of privacy loss.

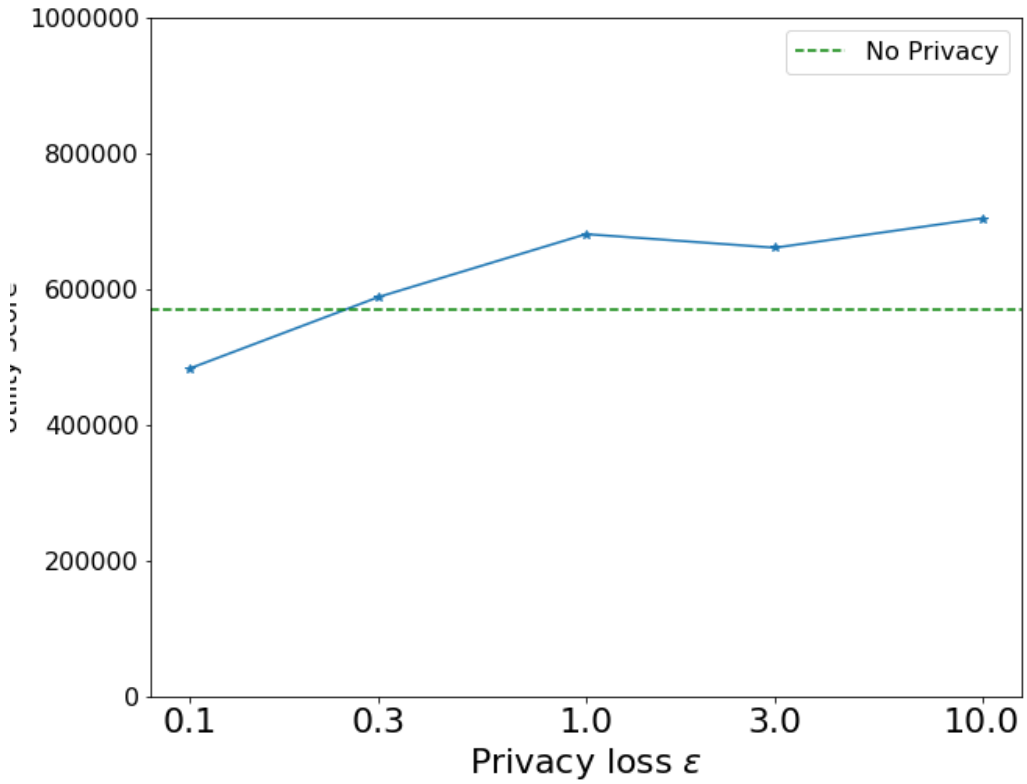


Figure 8.1: Utility scores of synthetic datasets under different levels of privacy loss.

Table 8.1 shows the evaluation results of the different scores using our submission under

different values of the privacy loss ϵ. The first row represents the scores achieved from a GAN model trained without differential privacy.

Figure 8.1 illustrates the relationship between the maximum privacy loss ϵ and the final score (average of the three scoring methods). Interestingly, we observe that for values $\epsilon > 1$, the privatized synthetic datasets have higher utility score than the synthetic dataset produced by GAN model trained without differential privacy. This might be because the small noise added at this levels of privacy helps to regularize the model training and prevent the *generator* for overfitting to the *critic* causing 'mode collapse' [TOB15] which is a common pitfall for generative models.

8.5 Conclusion and Future Work

In this chapter, we have presented the algorithms and privacy proofs for training Wasserstein GAN models with differential privacy. Methods described in this chapter were used in the submissions of team UCLANESL in the NIST synthetic dataset differential privacy challenge. All models and experiments are also available as open-source.

Future work: Our future work directions including performing analysis into how to further increase the utility score and how to apply the methods described in this chapter in the generation of the synthetic time-series sensor data (e.g., physiological sensors).

CHAPTER 9

Conclusion and Future Work

The primary goal of this dissertation was to, in the context of smart devices and sensor data, *study the vulnerability of machine learning models against attacks, how to make them more robust and how to protect the privacy of individuals contributing their sensor data to train machine learning models.*

We introduced, *GenAttack*, new algorithm for query-efficient generation of *adversarial examples* under the black-box threat model. We demonstrated the query efficiency of *GenAttack* by experiments against different image classification models for different datasets, MNIST, CIFAR-10 and ImageNet, and comparing the results to ZOO attack, the previous state-of-the-art black-box attack algorithm. We have also shown that adversarial examples are not limited to the computer vision domain but can also be performed against different kinds of models which process other data types such as speech and text. In Chapter 3, we demonstrated how to use *GenAttack* algorithm to perform *targeted classification attack* against speech classification model with 87% attack success rate. The magnitude of *adversarial noise* added by *GenAttack* to the original audio waveform is limited therefore it does not affect the classification outcome made by a human listener to the adversarial audio clips. In Chapter 4, we described how to craft adversarial examples against natural language understanding models. The attack algorithm replaces *few* words of the original text by another word in order to cause *targeted* miss-classification attack. The new replacement words are chosen to preserve the *semantic* and syntactic integrity of the modified text. Our experiments have demonstrated that we can attack and IMDB movie review classification model and a textual entailment model with 97% and 70% attack success rate, respectively.

We also, in Chapter 5, we proposed *NeuroMask* algorithm that can be used to generate explanations for the neural network result. The explanation of a neural network is useful to gain

more trust in the classification outcome. We then, in Chapter 6, focused on how to make smart devices that rely on speech input, such as the smart home devices, more robust against *audio spoofing attacks*. In Chapter 7, we introduced, *PhysioGAN*, a *model* architecture and training technique to generate synthetic samples of physiological sensor data *conditioned* on their class labels. Compared to existing methods of training generative models for physiological sensors data, *PhysioGAN* generates samples that exhibit a good balance between their *accuracy* and *diversity*. Therefore, it can be used to replace the real dataset of collected physiological sensors data to protect the privacy of the individuals from identifications or unwanted inferences. Finally, in chapter 8 we showed how to utilize the techniques of *differential privacy* to train generative models with a formal proof of their disclosure of private information from the training dataset. As the limits of the disclosure of private information can be quantified using the privacy loss parameter.

9.1 Directions for Future Work:

Due to the increasing role played by smart devices in our daily lives, and the wide adoption of machine learning models in these devices. The goals of studying the vulnerability of machine learning against the different security and privacy attacks and how to increase their robustness remain interesting questions for further research.

- **Query efficient adversarial attacks against hard-label black box models:** In this dissertation, we presented a query efficient method to generate adversarial attacks against black-box models that return *prediction scores* of classification labels for the input examples. A more strict model to consider is to attack machine learning models in the hard-label black box where the model outcome is only the prediction label without providing prediction confidence scores. The hard-label black-box setting is very challenging because the attacker has very limited and sparse feedback from the victim model on the effect of changes made to the adversarial noise. Recently, researchers [BRB18, CLC18] have proposed attack methods that succeed under this setting. However, these methods are still inefficient in terms of the number of required queries to the victim model.

- **Physical world attacks against speech recognition model:** In this dissertation, we demonstrated successful attacks against speech classification models. We have found that *adversarial audio clips* we generate are successful under the *direct model access* setting where the audio clip is fed directly into the victim audio classification model. However, we have noticed that the *adversarial audio clips* are no longer successful when we play them from a speaker device and record them over-the-air by a microphone and feed that recording into the victim model. Audio adversarial attacks that remain successful in the physical world setting would be considered a serious threat against smart devices because they allow attackers an easier channel to attack voice-controlled devices.

- **Defenses against adversarial examples:** We have shown that the different kinds (image, speech, and text) of classification models are vulnerable to adversarial examples of attacks. We have also demonstrated that some of the recently proposed defenses can be broken. The search for a robust defense mechanism remains an important goal of research to pursue. Ideally, the robustness of the defense technique should be *formally certified* to prevent being broken by future attacks. Recent research such as [WK17, WZC18, LAG18] has provided promising directions to pursue this goal.

- **Generation and evaluation of explanations of deep neural network models results:** We have proposed, *NeuroMask*, a technique to generate explanations of deep neural network image classification results. The explanations are important to gain more trust in the neural network results. In the future, I would like to extend this work by applying it to other kinds of neural network models (e.g., text and speech models). Also, I would like to study how to provide a quantitative metric to compare the results of *NeuroMask* against the explanations generated by other solutions.

- **Generation of differentially-private synthetic datasets of physiological sensors:** We have presented, *PhysioGAN*, as an approach to build and train generative models to produce synthetic samples of physiological sensors data. Compared to prior methods of generating synthetic sensor data, samples generated by *PhysioGAN* have high utility because they can be used to replace the real datasets in training machine learning with a moderate decrease in

their classification accuracies. A future direction of research I would like to pursue is to train *PhysioGAN* with differential privacy to quantify the limits of private identifiable information from training examples that are disclosed by *PhysioGAN*.

REFERENCES

[AAA16] Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, et al. "Deep speech 2: End-to-end speech recognition in english and mandarin." In *International conference on machine learning*, pp. 173–182, 2016.

[AAB16] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. "Tensorflow: Large-scale machine learning on heterogeneous distributed systems." *arXiv preprint arXiv:1603.04467*, 2016.

[AB17] Martin Arjovsky and Léon Bottou. "Towards principled methods for training generative adversarial networks." *arXiv preprint arXiv:1701.04862*, 2017.

[ABN19] Somayeh Aghanavesi, Filip Bergquist, Dag Nyholm, Marina Senek, and Mevludin Memedi. "Motion sensor-based assessment of Parkinson's disease motor symptoms during leg agility tests: results from levodopa challenge." *IEEE journal of biomedical and health informatics*, 2019.

[ABS17] Moustafa Alzantot, Bharathan Balaji, and Mani Srivastava. "Did you hear that? adversarial examples against automatic speech recognition." *Machine Deception Workshop, Neural Information Processing Systems (NIPS) 2017*, 2017.

[ACB17a] Martin Arjovsky, Soumith Chintala, and Léon Bottou. "Wasserstein gan." *arXiv preprint arXiv:1701.07875*, 2017.

[ACB17b] Martin Arjovsky, Soumith Chintala, and Léon Bottou. "Wasserstein gan." *arXiv preprint arXiv:1701.07875*, 2017.

[ACG16] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. "Deep learning with differential privacy." In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 308–318. ACM, 2016.

[ACS17] Moustafa Alzantot, Supriyo Chakraborty, and Mani Srivastava. "Sensegen: A deep learning architecture for synthetic sensor data generation." In *2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pp. 188–193. IEEE, 2017.

[ACW18] A. Athalye, N. Carlini, and D. Wagner. "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples." *arXiv preprint arXiv:1802.00420*, 2018.

[AEI17] A. Athalye, L. Engstrom, A. Ilyas, and K. Kwok. "Synthesizing robust adversarial examples." *arXiv preprint arXiv:1707.07397*, 2017.

[AF94] Edward J Anderson and Michael C Ferris. "Genetic algorithms for combinatorial optimization: the assemble line balancing problem." *ORSA Journal on Computing*, **6**(2):161–173, 1994.

[AGO13] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, and Jorge Luis Reyes-Ortiz. "A public domain dataset for human activity recognition using smartphones." In *Esann*, 2013.

[AKP19] Kati Alha, Elina Koskinen, Janne Paavilainen, and Juho Hamari. "Why do people play location-based augmented reality games: A study on Pokémon GO." *Computers in Human Behavior*, **93**:114–122, 2019.

[ALS18] Maximilian Alber, Sebastian Lapuschkin, Philipp Seegerer, Miriam Hägele, Kristof T. Schütt, Grégoire Montavon, Wojciech Samek, Klaus-Robert Müller, Sven Dähne, and Pieter-Jan Kindermans. "iNNvestigate neural networks!" *CoRR*, **abs/1808.04260**, 2018.

[AMJ14] Ossama Abdel-Hamid, Abdel-rahman Mohamed, Hui Jiang, Li Deng, Gerald Penn, and Dong Yu. "Convolutional neural networks for speech recognition." *IEEE/ACM Transactions on audio, speech, and language processing*, **22**(10):1533–1545, 2014.

[AS19] Moustafa Alzantot and Mani Srivastava. "Differential Privacy Synthetic Data Generation using WGANs.", 2019.

[ASC18] M. Alzantot, Y. Sharma, S. Chakraborty, and M. Srivastava. "GenAttack: Practical Black-box Attacks with Gradient-Free Optimization." *arXiv preprint arXiv:1805.11090*, 2018.

[ASE18] Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. "Generating Natural Language Adversarial Examples." *EMNLP: Conference on Empirical Methods in Natural Language Processing*, 2018.

[BAP15] Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. "A large annotated corpus for learning natural language inference." *arXiv preprint arXiv:1508.05326*, 2015.

[BBM15a] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation." *PloS one*, **10**(7):e0130140, 2015.

[BBM15b] Sebastian Bach, Alexander Binder, GrÃ©goire Montavon, Frederick Klauschen, Klaus-Robert MÃŒller, and Wojciech Samek. "On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation." *PLOS ONE*, **10**(7):1–46, 07 2015.

[BCV13] Yoshua Bengio, Aaron Courville, and Pascal Vincent. "Representation learning: A review and new perspectives." *IEEE transactions on pattern analysis and machine intelligence*, **35**(8):1798–1828, 2013.

[BD08] Laura E Boucheron and Phillip L De Leon. "On the inversion of mel-frequency cepstral coefficients for speech enhancement applications." In *Signals and Electronic Systems, 2008. ICSES'08. International Conference on*, pp. 485–488. IEEE, 2008.

[BDS18] Andrew Brock, Jeff Donahue, and Karen Simonyan. "Large scale gan training for high fidelity natural image synthesis." *arXiv preprint arXiv:1809.11096*, 2018.

[BKN10] Amos Beimel, Shiva Prasad Kasiviswanathan, and Kobbi Nissim. "Bounds on the sample complexity for private learning and private data release." In *Theory of Cryptography Conference*, pp. 437–454. Springer, 2010.

[BMP96] Dinabandhu Bhandari, CA Murthy, and Sankar K Pal. "Genetic algorithm with elitist model and its convergence." *International journal of pattern recognition and artificial intelligence*, **10**(06):731–747, 1996.

[BRB18] Wieland Brendel, Jonas Rauber, and Matthias Bethge. "Decision-based adversarial attacks: Reliable attacks against black-box machine learning models." *International Conference on Learning Representations (ICLR)*, 2018.

[BST14] Raef Bassily, Adam Smith, and Abhradeep Thakurta. "Differentially private empirical risk minimization: Efficient algorithms and tight error bounds." *arXiv preprint arXiv:1405.7085*, 2014.

[BVJ15] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. "Scheduled sampling for sequence prediction with recurrent neural networks." In *Advances in Neural Information Processing Systems*, pp. 1171–1179, 2015.

[BVV15] Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. "Generating sentences from a continuous space." *arXiv preprint arXiv:1511.06349*, 2015.

[CBM17] Edward Choi, Siddharth Biswal, Bradley Malin, Jon Duke, Walter F Stewart, and Jimeng Sun. "Generating multi-label discrete patient records using generative adversarial networks." *arXiv preprint arXiv:1703.06490*, 2017.

[CBR98] Stanley F Chen, Douglas Beeferman, and Ronald Rosenfeld. "Evaluation metrics for language models." In *DARPA Broadcast News Transcription and Understanding Workshop*, pp. 275–280. Citeseer, 1998.

[CDH16] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. "Infogan: Interpretable representation learning by information maximizing generative adversarial nets." In *Advances in neural information processing systems*, pp. 2172–2180, 2016.

[CGC15] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. "Gated feedback recurrent neural networks." In *International Conference on Machine Learning*, pp. 2067–2075, 2015.

[Che16] Adam J Cheyer. "Device access using voice authentication.", February 16 2016. US Patent 9,262,612.

[CKC06] Jay Chen, Karric Kwong, Dennis Chang, Jerry Luk, and Ruzena Bajcsy. "Wearable sensors for reliable fall detection." In *2005 IEEE Engineering in Medicine and Biology 27th Annual Conference*, pp. 3551–3554. IEEE, 2006.

[CKH16] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. "Wide & deep learning for recommender systems." In *Proceedings of the 1st workshop on deep learning for recommender systems*, pp. 7–10. ACM, 2016.

[CLC18] Minhao Cheng, Thong Le, Pin-Yu Chen, Jinfeng Yi, Huan Zhang, and Cho-Jui Hsieh. "Query-efficient hard-label black-box attack: An optimization-based approach." *arXiv preprint arXiv:1807.04457*, 2018.

[CMS13] Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. "One billion word benchmark for measuring progress in statistical language modeling." *arXiv preprint arXiv:1312.3005*, 2013.

[CMV16] Nicholas Carlini, Pratyush Mishra, Tavish Vaidya, Yuankai Zhang, Micah Sherr, Clay Shields, David Wagner, and Wenchao Zhou. "Hidden Voice Commands." In *USENIX Security Symposium*, pp. 513–530, 2016.

[con19] ASVspoof consortium. "ASVspoof 2019: Automatic Speaker Verification Spoofing and Countermeasures Challenge Evaluation Plan." 2019.

[CQD15] Nanxin Chen, Yanmin Qian, Heinrich Dinkel, Bo Chen, and Kai Yu. "Robust deep feature for spoofing detection—The SJTU system for ASVspoof 2015 challenge." In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[CSZ17] P. Y. Chen, Y. Sharma, H. Zhang, J. Yi, and C. Hsieh. "EAD: Elastic-net attacks to deep neural networks via adversarial examples." *arXiv preprint arXiv:1709.0414*, 2017.

[CTR17] Supriyo Chakraborty, Richard Tomsett, Ramya Raghavendra, Daniel Harborne, Moustafa Alzantot, Federico Cerutti, Mani Srivastava, Alun Preece, Simon Julier, Raghuveer M Rao, et al. "Interpretability of deep learning models: a survey of results." In *2017 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computed, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation*, pp. 1–6. IEEE, 2017.

[CVB14] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. "On the properties of neural machine translation: Encoder-decoder approaches." *arXiv preprint arXiv:1409.1259*, 2014.

[CW17a] N. Carlini and D. Wagner. "Towards evaluating the robustness of neural networks." *arXiv preprint arXiv:1608.04644*, 2017.

[CW17b] Nicholas Carlini and David Wagner. "Towards evaluating the robustness of neural networks." In *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 39–57. IEEE, 2017.

[CXZ17] Zhuxin Chen, Zhifeng Xie, Weibin Zhang, and Xiangmin Xu. "ResNet and Model Fusion for Automatic Spoofing Detection." In *INTERSPEECH*, pp. 102–106, 2017.

[CZL17] Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. "Enhanced lstm for natural language inference." In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pp. 1657–1668, 2017.

[CZS17a] P. Chen, H Zhang, Y. Sharma, J. Yi, and C. Hseih. "ZOO: Zeroth Order Optimization Based Black-box Attacks to Deep Neural Networks without Training Substitute Models." *arXiv preprint arXiv:1708.03999*, 2017.

[CZS17b] P. Y. Chen, H. Zhang, Y. Sharma, J. Yi, and C. Hsieh. "Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models." In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pp. 15–26. ACM, 2017.

[DDS09] J. Deng, W. Dong, R. Socher, J. Li, K. Li, and L. Fei-Fei. "Imagenet: A large-scale hierarchical image database." In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 248–255. IEEE, 2009.

[DR14] Cynthia Dwork, Aaron Roth, et al. "The algorithmic foundations of differential privacy." *Foundations and Trends® in Theoretical Computer Science*, **9**(3–4):211–407, 2014.

[DRV10] Cynthia Dwork, Guy N Rothblum, and Salil Vadhan. "Boosting and differential privacy." In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pp. 51–60. IEEE, 2010.

[DU18] Ugur Demir and Gozde Unal. "Patch-based image inpainting with generative adversarial networks." *arXiv preprint arXiv:1803.07422*, 2018.

[Dwo11] Cynthia Dwork. "Differential privacy." *Encyclopedia of Cryptography and Security*, pp. 338–340, 2011.

[ECM10] Developed with the special contribution of the European Heart Rhythm Association (EHRA), Endorsed by the European Association for Cardio-Thoracic Surgery (EACTS), Authors/Task Force Members, A John Camm, Paulus Kirchhof, Gregory YH Lip, Ulrich Schotten, Irene Savelieva, Sabine Ernst, Isabelle C Van Gelder, et al. "Guidelines for the management of atrial fibrillation: the Task Force for the Management of Atrial Fibrillation of the European Society of Cardiology (ESC)." *European heart journal*, **31**(19):2369–2429, 2010.

[EHR17] Cristóbal Esteban, Stephanie L Hyland, and Gunnar Rätsch. "Real-valued (medical) time series generation with recurrent conditional gans." *arXiv preprint arXiv:1706.02633*, 2017.

[EKN17] Andre Esteva, Brett Kuprel, Roberto A Novoa, Justin Ko, Susan M Swetter, Helen M Blau, and Sebastian Thrun. "Dermatologist-level classification of skin cancer with deep neural networks." *Nature*, **542**(7639):115, 2017.

[EKY13] Nicholas WD Evans, Tomi Kinnunen, and Junichi Yamagishi. "Spoofing and countermeasures for automatic speaker verification." In *Interspeech*, pp. 925–929, 2013.

[ERL18] J. Ebrahimi, A. Rao, D. Lowd, and D. Dou. "HotFlip: White-Box Adversarial Examples for Text Classification." *ACL'18; arXiv preprint arXiv:1712.06751*, 2018.

[FJR15] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. "Model inversion attacks that exploit confidence information and basic countermeasures." In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pp. 1322–1333. ACM, 2015.

[FV17] Ruth C Fong and Andrea Vedaldi. "Interpretable explanations of black boxes by meaningful perturbation." In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3429–3437, 2017.

[GAA17] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. "Improved training of wasserstein gans." In *Advances in Neural Information Processing Systems*, pp. 5767–5777, 2017.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[GDG17] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. "Badnets: Identifying vulnerabilities in the machine learning model supply chain." *arXiv preprint arXiv:1708.06733*, 2017.

[GDV18] Melanie RF Gropler, Aarti S Dalal, George F Van Hare, and Jennifer N Avari Silva. "Can smartphone wireless ECGs be used to accurately assess ECG intervals in pediatrics? A comparison of mobile health monitoring to standard 12-lead ECG." *PloS one*, **13**(9):e0204403, 2018.

[Gib16] Elizabeth Gibney. "Google AI algorithm masters ancient game of Go." *Nature News*, **529**(7587):445, 2016.

[GMH13] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. "Speech recognition with deep recurrent neural networks." In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pp. 6645–6649. IEEE, 2013.

[GMR18] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. "A survey of methods for explaining black box models." *ACM computing surveys (CSUR)*, **51**(5):93, 2018.

[Goo16] Ian Goodfellow. "NIPS 2016 tutorial: Generative adversarial networks." *arXiv preprint arXiv:1701.00160*, 2016.

[GPM14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative adversarial nets." In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

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

[Gra13] Alex Graves. "Generating sequences with recurrent neural networks." *arXiv preprint arXiv:1308.0850*, 2013.

[GRM17] C. Guo, M. Rana, and L. van der Maaten. "Countering adversarial images using input transformations." *arXiv preprint arXiv:1711.00117*, 2017.

[GSS14] I. Goodfellow, J. Shlens, and C. Szegedy. "Explaining and harnessing adversarial examples." *arXiv preprint arXiv:1412.6572*, 2014.

[Gun17] David Gunning. "Explainable artificial intelligence (xai)." *Defense Advanced Research Projects Agency (DARPA), nd Web*, 2017.

[HDY12] Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Brian Kingsbury, et al. "Deep neural networks for acoustic modeling in speech recognition." *IEEE Signal processing magazine*, **29**, 2012.

[HE17] David Ha and Douglas Eck. "A neural representation of sketch drawings." *arXiv preprint arXiv:1704.03477*, 2017.

[HLH18] Wen-Chin Huang, Chen-Chou Lo, Hsin-Te Hwang, Yu Tsao, and Hsin-Min Wang. "Wavenet vocoder and its applications in voice conversion." *ROCLING 2018*, p. 96, 2018.

[HMP17] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. "beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework." *ICLR*, **2**(5):6, 2017.

[HRU17] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. "Gans trained by a two time-scale update rule converge to a local nash equilibrium." In *Advances in Neural Information Processing Systems*, pp. 6626–6637, 2017.

[HSS12] Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. "Neural networks for machine learning lecture 6a overview of mini-batch gradient descent." *Cited on*, p. 14, 2012.

[Hus15] Ferenc Huszár. "How (not) to train your generative model: Scheduled sampling, likelihood, adversary?" *arXiv preprint arXiv:1511.05101*, 2015.

[HYL17] Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. "Toward controlled generation of text." In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1587–1596. JMLR. org, 2017.

[HZR15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification." In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

[HZR16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[IEA18] A. Ilyas, L. Engstrom, A. Athalye, and J. Lin. "Black-box Adversarial Attacks with Limited Queries and Information." *arXiv preprint arXiv:1804.08598*, 2018.

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

[IWG18] Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. "Adversarial Example Generation with Syntactically Controlled Paraphrase Networks." In *Proceedings of NAACL*, 2018.

[JBW18] Lauri Juvela, Bajibabu Bollepalli, Xin Wang, Hirokazu Kameoka, Manu Airaksinen, Junichi Yamagishi, and Paavo Alku. "Speech waveform synthesis from MFCC sequences with generative adversarial networks." In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5679–5683. IEEE, 2018.

[JDB18] Felix Juefei-Xu, Rahul Dey, Vishnu Naresh Boddeti, and Marios Savvides. "RankGAN: A Maximum Margin Ranking GAN for Generating Faces." In *Asian Conference on Computer Vision*, pp. 3–18. Springer, 2018.

[JL17] Robin Jia and Percy Liang. "Adversarial examples for evaluating reading comprehension systems." *arXiv preprint arXiv:1707.07328*, 2017.

[JYS18] James Jordon, Jinsung Yoon, and Mihaela van der Schaar. "PATE-GAN: generating synthetic data with differential privacy guarantees." 2018.

[KB14a] D. Kingma and J. Ba. "Adam: A method for stochastic optimization." *arXiv preprint arXiv:1412.6980*, 2014.

[KB14b] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization." *arXiv preprint arXiv:1412.6980*, 2014.

[KEY17] Tomi Kinnunen, Nicholas Evans, Junichi Yamagishi, Kong Aik Lee, Md Sahidullah, Massimiliano Todisco, and Héctor Delgado. "ASVspoof 2017: Automatic speaker verification spoofing and countermeasures challenge evaluation plan." *Training*, **10**(1508):1508, 2017.

[KGB16a] A. Kurakin, I. Goodfellow, and S. Bengio. "Adversarial examples in the physical world." *arXiv preprint arXiv:1607.02533*, 2016.

[KGB16b] A. Kurakin, I. Goodfellow, and S. Bengio. "Adversarial machine learning at scale." *ICLR'17; arXiv preprint arXiv:1611.01236*, 2016.

[KGB18] A. Kurakin, I. Goodfellow, S. Bengio, Y. Dong, F. Liao, M. Liang, T. Pang, J. Zhu, X. Hu, C. Xie, J. Wang, Z. Zhang, Z. Ren, A. Yuille, S. Huang, Y. Zhao, Y. Zhao, Z. Han, J Long, Y. Berdibekov, T. Akiba, S. Tokui, and M. Abe. "Adversarial attacks and defences competition." *arXiv preprint arXiv:1804.00097*, 2018.

[KH09] A. Krizhevsky and G. Hinton. "Learning multiple layers of features from tiny images." 2009.

[KLA19] Tero Karras, Samuli Laine, and Timo Aila. "A style-based generator architecture for generative adversarial networks." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4401–4410, 2019.

[KLD18] Tomi Kinnunen, Kong Aik Lee, Héctor Delgado, Nicholas Evans, Massimiliano Todisco, Md Sahidullah, Junichi Yamagishi, and Douglas A Reynolds. "t-DCF: a detection cost function for the tandem assessment of spoofing countermeasures and automatic speaker verification." *arXiv preprint arXiv:1804.09618*, 2018.

[KLN11] Shiva Prasad Kasiviswanathan, Homin K Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. "What can we learn privately?" *SIAM Journal on Computing*, **40**(3):793–826, 2011.

[KOV17] Peter Kairouz, Sewoong Oh, and Pramod Viswanath. "The composition theorem for differential privacy." *IEEE Transactions on Information Theory*, **63**(6):4037–4049, 2017.

[KSJ] Diederik P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. "Improving variational inference with inverse autoregressive flow.(nips), 2016." *URL http://arxiv. org/abs/1606.04934*.

[KTL18] V. Kuleshov, S. Thakoor, T. Lau, and S. Ermon. "Adversarial Examples for Natural Language Classification Problems." *OpenReview submission OpenReview:r1QZ3zbAZ*, 2018.

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

[LAG18] Mathias Lecuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. "Certified robustness to adversarial examples with differential privacy." *arXiv preprint arXiv:1802.03471*, 2018.

[LBH15] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning." *nature*, **521**(7553):436, 2015.

[LCC18] P. H. Lu, P. Y. Chen, K. C. Chen, and C. M. Yu. "On the limitation of MagNet defense against L1-based adversarial examples." *arXiv preprint arXiv:1805.00310*, 2018.

[LCL18] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. "Delving into transferable adversarial examples and black-box attacks." *International Conference on Learning Representations (ICLR)*, 2018.

[Lip16a] Zachary C Lipton. "The mythos of model interpretability." *arXiv preprint arXiv:1606.03490*, 2016.

[Lip16b] Zachary Chase Lipton. "The Mythos of Model Interpretability." *CoRR*, **abs/1606.03490**, 2016.

[LNM17] Galina Lavrentyeva, Sergey Novoselov, Egor Malykh, Alexander Kozlov, Oleg Kudashev, and Vadim Shchemelinin. "Audio Replay Attack Detection with Deep Learning Frameworks." In *Interspeech*, pp. 82–86, 2017.

[LTH17] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. "Photo-realistic single image super-resolution using a generative adversarial network." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4681–4690, 2017.

[LYT18] Jaime Lorenzo-Trueba, Junichi Yamagishi, Tomoki Toda, Daisuke Saito, Fernando Villavicencio, and Zhenhua Kinnunen, Tomi a Ling. "The voice conversion challenge 2018: Promoting development of parallel and nonparallel methods." *arXiv preprint arXiv:1804.04262*, 2018.

[LZZ18] Sidi Lu, Yaoming Zhu, Weinan Zhang, Jun Wang, and Yong Yu. "Neural text generation: past, present and beyond." *arXiv preprint arXiv:1803.07133*, 2018.

[McS09] Frank D McSherry. "Privacy integrated queries: an extensible platform for privacy-preserving data analysis." In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, pp. 19–30. ACM, 2009.

[MDP11] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. "Learning Word Vectors for Sentiment Analysis." In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[MFF16] S. Moosavi-Dezfooli, A. Fawzi, and P. Frossard. "Deepfool: a simple and accurate method to fool deep neural networks." In *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, number EPFL-CONF-218057, 2016.

[MKB10] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. "Recurrent neural network based language model." In *Eleventh annual conference of the international speech communication association*, 2010.

[MKS13] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. "Playing atari with deep reinforcement learning." *arXiv preprint arXiv:1312.5602*, 2013.

[MMS17a] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. "Towards deep learning models resistant to adversarial attacks." *arXiv preprint arXiv:1706.06083*, 2017.

[MMS17b] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. "Towards deep learning models resistant to adversarial attacks." *arXiv preprint arXiv:1706.06083*, 2017.

[MO14] Mehdi Mirza and Simon Osindero. "Conditional generative adversarial nets." *arXiv preprint arXiv:1411.1784*, 2014.

[MPP16] Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. "Unrolled generative adversarial networks." *arXiv preprint arXiv:1611.02163*, 2016.

[MPP17] Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. "Unrolled Generative Adversarial Networks." 2017.

[MRL15] Brian McFee, Colin Raffel, Dawen Liang, Daniel PW Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. "librosa: Audio and music signal analysis in python." In *Proceedings of the 14th python in science conference*, pp. 18–25, 2015.

[MST16] Nikola Mrkšić, Diarmuid O Séaghdha, Blaise Thomson, Milica Gašić, Lina Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. "Counter-fitting word vectors to linguistic constraints." *arXiv preprint arXiv:1603.00892*, 2016.

[Muh89] Heinz Mühlenbein. "Parallel genetic algorithms, population genetics and combinatorial optimization." In *Workshop on Parallel Processing: Logic, Organization, and Technology*, pp. 398–406. Springer, 1989.

[nis] "Differential Privacy Synthetic Data Challenge." `https://challenge.gov/a/buzz/nist-pscr/differential-privacy-synthetic-data-challenge`.

[ODZ16] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. "Wavenet: A generative model for raw audio." *arXiv preprint arXiv:1609.03499*, 2016.

[PAE16] Nicolas Papernot, Martín Abadi, Ulfar Erlingsson, Ian Goodfellow, and Kunal Talwar. "Semi-supervised knowledge transfer for deep learning from private training data." *arXiv preprint arXiv:1610.05755*, 2016.

[Par16] Parliament and Council of the European Union. "General data protection regulation.", 2016.

[PGC17] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. "Automatic differentiation in pytorch." 2017.

[PMG16] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. Celik, and A. Swami. "Practical Black-Box Attacks against Machine Learning." *arXiv preprint arXiv:1602.02697*, 2016.

[PMG17] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami. "Practical black-box attacks against machine learning." In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pp. 506–519. ACM, 2017.

[PMG18] Noseong Park, Mahmoud Mohammadi, Kshitij Gorde, Sushil Jajodia, Hongkyu Park, and Youngmin Kim. "Data synthesis based on generative adversarial networks." *Proceedings of the VLDB Endowment*, **11**(10):1071–1083, 2018.

[PMJ16] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B.; Celik, and A. Swami. "The limitations of deep learning in adversarial settings." *arXiv preprint arXiv:1511.07528*, 2016.

[PMS16] N. Papernot, P. McDaniel, A. Swami, and R. Harang. "Crafting Adversarial Input Sequences for Recurrent Neural Networks." *arXiv preprint arXiv:1604.08275*, 2016.

[PSM14] Jeffrey Pennington, Richard Socher, and Christopher Manning. "Glove: Global vectors for word representation." In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543, 2014.

[RAY16] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. "Generative adversarial text to image synthesis." *arXiv preprint arXiv:1605.05396*, 2016.

[RER18] Adam Roberts, Jesse Engel, Colin Raffel, Curtis Hawthorne, and Douglas Eck. "A hierarchical latent vector model for learning long-term structure in music." *arXiv preprint arXiv:1803.05428*, 2018.

[REU18] REUTERS news. "Amazon scraps secret AI recruiting tool that showed bias against women." https://www.reuters.com/article/us-amazon-com-jobs-automation-insight/amazon-scraps-secret-ai-recruiting-tool-that-showed-bias-against-women-idUSKCN1MK08G, 2018.

[RHH17] Pranav Rajpurkar, Awni Y Hannun, Masoumeh Haghpanahi, Codie Bourn, and Andrew Y Ng. "Cardiologist-level arrhythmia detection with convolutional neural networks." *arXiv preprint arXiv:1707.01836*, 2017.

[RHR17] Nirupam Roy, Haitham Hassanieh, and Romit Roy Choudhury. "Backdoor: Making microphones hear inaudible sounds." In *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services*, pp. 2–14. ACM, 2017.

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

[ROP19] Ali Razavi, Aäron van den Oord, Ben Poole, and Oriol Vinyals. "Preventing posterior collapse with delta-vaes." *arXiv preprint arXiv:1901.03416*, 2019.

[ROV19] Ali Razavi, Aaron van den Oord, and Oriol Vinyals. "Generating Diverse High-Fidelity Images with VQ-VAE-2." *arXiv preprint arXiv:1906.00446*, 2019.

[RQD00] Douglas A Reynolds, Thomas F Quatieri, and Robert B Dunn. "Speaker verification using adapted Gaussian mixture models." *Digital signal processing*, **10**(1-3):19–41, 2000.

[RR95] Douglas A Reynolds and Richard C Rose. "Robust text-independent speaker identification using Gaussian mixture speaker models." *IEEE transactions on speech and audio processing*, **3**(1):72–83, 1995.

[RSG16] Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. ""Why Should I Trust You?": Explaining the Predictions of Any Classifier." *CoRR*, **abs/1602.04938**, 2016.

[RSG18] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Semantically Equivalent Adversarial Rules for Debugging NLP Models." In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pp. 856–865, 2018.

[RV19] Suman Ravuri and Oriol Vinyals. "Classification Accuracy Score for Conditional Generative Models." *arXiv preprint arXiv:1905.10887*, 2019.

[SC07] Stan Salvador and Philip Chan. "Toward accurate dynamic time warping in linear time and space." *Intelligent Data Analysis*, **11**(5):561–580, 2007.

[SC17] Y. Sharma and P. Y. Chen. "Attacking the Madry defense model with L1-based adversarial examples." *arXiv preprint arXiv:1710.10733*, 2017.

[SC18] Y. Sharma and P. Y. Chen. "Bypassing feature squeezing by increasing adversary strength." *arXiv preprint arXiv:1803.09868*, 2018.

[SCD17] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. "Grad-cam: Visual explanations from deep networks via gradient-based localization." In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 618–626, 2017.

[SDB14] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. "Striving for simplicity: The all convolutional net." *arXiv preprint arXiv:1412.6806*, 2014.

[SGZ16] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. "Improved techniques for training gans." In *Advances in neural information processing systems*, pp. 2234–2242, 2016.

[SHK14] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. "Dropout: a simple way to prevent neural networks from overfitting." *The Journal of Machine Learning Research*, **15**(1):1929–1958, 2014.

[SK11] George Silberman and Katherine L Kahn. "Burdens on research imposed by institutional review boards: the state of the evidence and its implications for regulatory reform." *The Milbank Quarterly*, **89**(4):599–627, 2011.

[SLA18] Y. Sharma, T. Le, and M. Alzantot. "CAAD 2018: Generating Transferable Adversarial Examples." *arXiv preprint arXiv:1810.01268*, 2018.

[SLJ15] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. "Going deeper with convolutions." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

[SM17] Liwei Song and Prateek Mittal. "Inaudible Voice Commands." *arXiv preprint arXiv:1708.07238*, 2017.

[SMS00] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. "Policy gradient methods for reinforcement learning with function approximation." In *Advances in neural information processing systems*, pp. 1057–1063, 2000.

[SP15] Tara N Sainath and Carolina Parada. "Convolutional neural networks for small-footprint keyword spotting." In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[spe] "Speech Commands Dataset." `https://research.googleblog.com/2017/08/launching-speech-commands-dataset.html`.

[SRB19] Lukas Schott, Jonas Rauber, Matthias Bethge, and Wieland Brendel. "Towards the first adversarially robust neural network model on MNIST." *International Conference on Learning Representations (ICLR)*, 2019.

[SSS11] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. "The German Traffic Sign Recognition Benchmark: A multi-class classification competition." In *IJCNN*, volume 6, p. 7, 2011.

[SSS17] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. "Membership inference attacks against machine learning models." In *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 3–18. IEEE, 2017.

[STK17] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. "Smoothgrad: removing noise by adding noise." *arXiv preprint arXiv:1706.03825*, 2017.

[SVI15] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. "Rethinking the Inception Architecture for Computer Vision." *CoRR*, **abs/1512.00567**, 2015.

[SVR17] Akash Srivastava, Lazar Valkov, Chris Russell, Michael U Gutmann, and Charles Sutton. "Veegan: Reducing mode collapse in gans using implicit variational learning." In *Advances in Neural Information Processing Systems*, pp. 3308–3318, 2017.

[SVZ13] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. "Deep inside convolutional networks: Visualising image classification models and saliency maps." *arXiv preprint arXiv:1312.6034*, 2013.

[SZC18] Dong Su, Huan Zhang, Hongge Chen, Jinfeng Yi, Pin-Yu Chen, and Yupeng Gao. "Is Robustness the Cost of Accuracy?–A Comprehensive Study on the Robustness of 18 Deep Image Classification Models." In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 631–648, 2018.

[SZH18] Ahmed Salem, Yang Zhang, Mathias Humbert, Pascal Berrang, Mario Fritz, and Michael Backes. "Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models." *arXiv preprint arXiv:1806.01246*, 2018.

[SZS13] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, and I. Goodfellow. "Intriguing properties of neural networks." *arXiv preprint arXiv:1312.6199*, 2013.

[TCS16] Tomoki Toda, Ling-Hui Chen, Daisuke Saito, Fernando Villavicencio, Mirjam Wester, Zhizheng Wu, and Junichi Yamagishi. "The Voice Conversion Challenge 2016." In *Interspeech*, pp. 1632–1636, 2016.

[TDE17] Massimiliano Todisco, Héctor Delgado, and Nicholas Evans. "Constant Q cepstral coefficients: A spoofing countermeasure for automatic speaker verification." *Computer Speech & Language*, **45**:516–535, 2017.

[TH12] Tijmen Tieleman and Geoffrey Hinton. "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude." *COURSERA: Neural networks for machine learning*, **4**(2):26–31, 2012.

[TKP17] F. Tramèr, A. Kurakin, N. Papernot, D. Boneh, and P. McDaniel. "Ensemble adversarial training: Attacks and Defenses." *arXiv preprint arXiv:1705.07204*, 2017.

[TOB15] Lucas Theis, Aäron van den Oord, and Matthias Bethge. "A note on the evaluation of generative models." *arXiv preprint arXiv:1511.01844*, 2015.

[TTC18] Chun-Chen Tu, Paishun Ting, Pin-Yu Chen, Sijia Liu, Huan Zhang, Jinfeng Yi, Cho-Jui Hsieh, and Shin-Ming Cheng. "AutoZOOM: Autoencoder-based Zeroth Order Optimization Method for Attacking Black-box Neural Networks." *arXiv preprint arXiv:1805.11770*, 2018.

[Ven19] Brianna Vendetti. "2018 Differential Privacy Synthetic Data Challenge.", Aug 2019.

[Vil08] Cédric Villani. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008.

[VMO15] Jesus Villalba, Antonio Miguel, Alfonso Ortega, and Eduardo Lleida. "Spoofing detection with DNN and one-class SVM for the ASVspoof 2015 challenge." In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[WCG18] Jiwei Wang, Yiqiang Chen, Yang Gu, Yunlong Xiao, and Haonan Pan. "SensoryGANs: An Effective Generative Adversarial Framework for Sensor-based Human Activity Recognition." In *2018 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2018.

[WK17] Eric Wong and J Zico Kolter. "Provable defenses against adversarial examples via the convex outer adversarial polytope." *arXiv preprint arXiv:1711.00851*, 2017.

[WKE15] Zhizheng Wu, Tomi Kinnunen, Nicholas Evans, Junichi Yamagishi, Cemal Hanilçi, Md Sahidullah, and Aleksandr Sizov. "ASVspoof 2015: the first automatic speaker verification spoofing and countermeasures challenge." In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[WMH18] Johan Wannenburg, Reza Malekian, and Gerhard P Hancke. "Wireless capacitive-based ECG sensing for feature extraction and mobile health monitoring." *IEEE Sensors Journal*, **18**(14):6023–6032, 2018.

[WSC16] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. "Google's neural machine translation system: Bridging the gap between human and machine translation." *arXiv preprint arXiv:1609.08144*, 2016.

[Wur] Yoram Wurmser. "Advanced Wearables Pick Up Pace as Fitness Trackers Slow." `https://www.emarketer.com/content/wearables-2019`.

[WW18] Ke Wang and Xiaojun Wan. "SentiGAN: Generating Sentimental Texts via Mixture Adversarial Networks." In *IJCAI*, pp. 4446–4452, 2018.

[WWK16] Zhizheng Wu, Oliver Watts, and Simon King. "Merlin: An Open Source Neural Network Speech Synthesis System." In *SSW*, pp. 202–207, 2016.

[WYK15] Longbiao Wang, Yohei Yoshida, Yuta Kawakami, and Seiichi Nakagawa. "Relative phase information for detecting human speech and spoofed speech." In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[WZC18] Tsui-Wei Weng, Huan Zhang, Hongge Chen, Zhao Song, Cho-Jui Hsieh, Duane Boning, Inderjit S Dhillon, and Luca Daniel. "Towards fast computation of certified robustness for relu networks." *arXiv preprint arXiv:1804.09699*, 2018.

[XV18] Lei Xu and Kalyan Veeramachaneni. "Synthesizing tabular data using generative adversarial networks." *arXiv preprint arXiv:1811.11264*, 2018.

[XZH18] Tao Xu, Pengchuan Zhang, Qiuyuan Huang, Han Zhang, Zhe Gan, Xiaolei Huang, and Xiaodong He. "Attngan: Fine-grained text to image generation with attentional generative adversarial networks." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1316–1324, 2018.

[YPT18] Özal Yıldırım, Paweł Pławiak, Ru-San Tan, and U Rajendra Acharya. "Arrhythmia detection using deep convolutional neural network with long duration ECG signals." *Computers in biology and medicine*, **102**:411–420, 2018.

[YVE17] Alexandre Yahi, Rami Vanguri, Noémie Elhadad, and Nicholas P Tatonetti. "Generative Adversarial Networks for Electronic Health Records: A Framework for Exploring and Evaluating Methods for Predicting Drug-Induced Laboratory Test Trajectories." *arXiv preprint arXiv:1712.00164*, 2017.

[YZW17] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. "Seqgan: Sequence generative adversarial nets with policy gradient." In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[YZY18] Zhengyuan Yang, Yixuan Zhang, Jerry Yu, Junjie Cai, and Jiebo Luo. "End-to-end multi-modal multi-task vehicle control for self-driving cars with visual perceptions." In *2018 24th International Conference on Pattern Recognition (ICPR)*, pp. 2289–2294. IEEE, 2018.

[ZCS19] Huan Zhang, Hongge Chen, Zhao Song, Duane Boning, Inderjit S Dhillon, and Cho-Jui Hsieh. "The Limitations of Adversarial Training and the Blind-Spot Attack." *International Conference on Learning Representations (ICLR)*, 2019.

[ZDS18] Zhengli Zhao, Dheeru Dua, and Sameer Singh. "Generating natural adversarial examples." *arXiv preprint arXiv:1710.11342*, 2018.

[ZF14] Matthew D Zeiler and Rob Fergus. "Visualizing and understanding convolutional networks." In *European conference on computer vision*, pp. 818–833. Springer, 2014.

[ZGC16] Yizhe Zhang, Zhe Gan, and Lawrence Carin. "Generating text via adversarial training." In *NIPS workshop on Adversarial Training*, volume 21, 2016.

[ZGM18] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. "Self-attention generative adversarial networks." *arXiv preprint arXiv:1805.08318*, 2018.

[ZLZ18] Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. "Texygen: A benchmarking platform for text generation models." In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pp. 1097–1100. ACM, 2018.

[ZNZ18] Quanshi Zhang, Ying Nian Wu, and Song-Chun Zhu. "Interpretable convolutional neural networks." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8827–8836, 2018.

[ZZL15] Xiang Zhang, Junbo Zhao, and Yann LeCun. "Character-level convolutional networks for text classification." In *Advances in neural information processing systems*, pp. 649–657, 2015.