UNIVERSITY OF CALIFORNIA SAN DIEGO

Detecting, Diagnosing, Deflecting and Designing Adversarial Attacks

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

in

Computer Science

by

Yao Qin

2020

DEDICATION

To my family.

EPIGRAPH

A man should look for what is, and not for what he thinks should be.

Albert Einstein

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

First, I would like to thank my advisor Professor Garrison Cottrell, who offered me the great opportunity to join the family of Gary's Unbelievable Research Unit (GURU). He is very supportive of every decision that I made during my Ph.D. and gave me great freedom to pursue the research area that I am interested in. He could not be more supportive of my internships and attendance at the top conferences in machine learning. On the other hand, when I got lost in my research work, I will always go for his suggestions. Without his insightful guidance and incredible help, I would not have been able to mature into an independent researcher. Many thanks to Professors Manmohan Chandraker, Kamalika Chaudhuri, Lawrence Saul and Zhuowen Tu for serving on my thesis committee and offering their time and insightful comments. I also want to thank Professor Huchuan Lu for inspiring and guiding me to work on the research work related to AI while I was an undergraduate student.

During my Ph.D., I am very fortunate to have the chance to intern at different places and greatly broaden my horizon. I would like to thank Haifeng Chen and Dongjin Song who provide me the chance to intern at NEC Labs America. This is my first internship and from which I become more confident in doing independent research work. Great thanks to Aditya Nori and Antonio Criminisi, who are my mentors at Microsoft Research, Cambridge. They are always very appreciated for my research work and help me explore an incredible opportunity to better continue my Ph.D. I am also very grateful to Jingdong Wang for my internship at Microsoft Research, Asia. Lastly, I want to thank my internship mentors and collaborators at Google Brain, they are Geoffrey Hinton, Ian Goodfellow, Colin Raffel, Nicholas Frosst, Nicholas Carlini, Suharsh Sivakumar, Raghu Krishnamoorthi, etc. I could not be more fortunate to have the chance to work with these excellent researchers, who show me how to go for high-quality research work.

I have to give my special thanks to three people who play a pretty important role in offering me big opportunities during my Ph.D. career. First, I would like to thank Christopher Bishop, who is the director of the Microsoft Research Lab in Cambridge, UK. After a brief chat at the MSR reception in NIPS 2016, he contacted me the next day about a chance to intern at

MSR, Cambridge. I was really overwelmed by that unexpected favor since I was a very junior Ph.D. student at that time. At the end of my internship, Chris also offered me an incredible chance to continue my Ph.D. with the support of MSRC.

My sincere gratitude also goes to Ian Goodfellow, who shows me how to play the leading role in a research area and ignites me the interests in the research topic in this thesis. I could feel his help in many details and in every possible way. The biggest gift that he provides with me is the huge opportunity, including the chance to join the family of Red Team, the chance to work under the supervision of Geoffrey Hinton and the chance for further collaborations. Additionally, as a busy manager, I feel highly respected and valued because he would like to share his precious time for a weekly meeting and a weekly lunch.

I am also extremely fortunate and would like to give my special thanks to Geoffrey Hinton, who is my advisor while I was interning at Brain Toronto for half a year. It is almost every Ph.D. student's dream to work with such a big name in his/her research field and learn from him about the attitude towards research. When I had lunch with Geoff one-to-one on the first day of my internship, my dream came true and I could not enumerate everything that I learned from him!

I want to thank all the team members in Gary's Unbelievable Research Unit (GURU): Ben Cipollini, Davis Liang, Tomoki Tsuchida, Panqu Wang, Yufei Wang, Amanda Song, Yan Shu, etc. Many thanks to the graduate and visiting students at UCSD: Julaiti Alafate, Silvia Chyou, Kun Huang, Di Huang, Xun Huang, Wang-Cheng Kang, Guo Li, Zhengqin Li, Jianmo Ni, Meng Song, Tiancheng Sun, Saining Xie, Ling-Qi Yan, Songbai Yan, Luning Yang, Chicheng Zhang, Qianren Zhou, Shilin Zhu and many others for their suggestions and encouragement. I would greatly thank the Ph.D. students at UCSD: Sai Bi, Mengting Wan, Lifan Wu and Zexiang Xu for tons of meals together and their help in every details of my daily life. I am very grateful to get even closer with my old friend Minghan Song when she starts working at San Diego. I am also very fortunate to have my old friend Xiangchen Zhao at San Diego who truly shares my happiness and sadness and stays with me during the hardest time of my PhD. San Diego is more like my second hometown because of my lovely friends here.

During my internships, I am very fortunate to make friends with a lot of interesting people: Siddharth Ancha, Betty Chen, Ricky Tian Qi Chen, Qi Dou, Changwon Jang, Yang Li, S Mehdi Hosseini M, Yunchen Pu, Yang Song, Qi Sun, Yeming Wen, Yaohai Xu, Zhen Xu, Yuchen Yao, Xinchen Yan, Sherry Yang, Wenchao Yu, Chiyuan Zhang, Guodong Zhang, Ru Zhang, Yizhe Zhang, Judy Zhu and many others, who enlightened my various internships across the world. I also want to give my sincere thanks to my old friends Mengfei Tong, Qiuyi Wu, Bofan Wei, Peng Lu, Yiqun Xu, He Wang and many others who witnessed my growth and demonstrated to me how wonderful the friendship could be!

Finally, I also want to give my deepest thanks to my parents: Qin Qin and Quying Zhang and my sister Lu Qin for their love and support all the time. I could not be luckier to have them as my family.

Chapter 2 and Chapter 3, are based on the material as it appears in the Proceedings of the International Conference on Learning Representations (ICLR20) (Yao Qin, Nicholas Frosst, Sara Sabour, Colin Raffel, Garrison Cottrell and Geoffrey Hinton, “Detecting and Diagnosing Adversarial Images with Class-Conditional Capsule Reconstructions”). The dissertation author was the co-primary investigator and author of this paper.

Chapter 4, has been submitted for publication of the material as it may appear in the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR20) (Yao Qin, Nicholas Frosst, Colin Raffel, Garrison Cottrell and Geoffrey Hinton, “Deflecting Adversarial Attacks”). The dissertation author was the primary investigator and author of this paper.

Chapter 5, is based on the material as it appears in the Proceedings of the International Conference on Machine Learning (ICML19) (Yao Qin, Nicholas Carlini, Ian Goodfellow, Garrison Cottrell and Colin Raffel, “Imperceptible, Robust and Targeted Adversarial Examples for Automatic Speech Recognition”). The dissertation author was the primary investigator and author of this paper.

VITA

2011–2015	B.S. in Communications Engineering, Dalian University of Technology, China
2015–2017	M.S. in Computer Science, University of California San Diego, USA
2015–2020	Ph.D. in Computer Science, University of California San Diego, USA

PUBLICATIONS

Yao Qin*, Nicholas Frosst*, Sara Sabour, Colin Raffel, Garrison Cottrell and Geoffrey Hinton. "Detecting and Diagnosing Adversarial Images with Class-Conditional Capsule Reconstructions", in *International Conference on Learning Representations* (ICLR), 2020. (* Equal contribution)

Yao Qin, Nicholas Frosst, Colin Raffel, Garrison Cottrell and Geoffrey Hinton. "Deflecting Adversarial Attacks", submitted to *IEEE Conference on Computer Vision and Pattern Recognition* (CVPR), 2020.

Yao Qin, Nicholas Carlini, Ian Goodfellow, Garrison Cottrell and Colin Raffel. "Imperceptible, Robust and Targeted Adversarial Examples for Automatic Speech Recognition", in *International Conference on Machine Learning* (ICML), 2019.

Ian Goodfellow, **Yao Qin**, David Berthelot. "Evaluation Methodology for Attacks Against Confidence Thresholding Models." *arXiv Preprints*, 2018.

Yao Qin, Konstantinos Kamnitsas, Siddharth Ancha, Jay Nanavati, Garrison Cottrell, Antonio Criminisi and Aditya Nori. "Autofocus Layer for Semantic Segmentation", in *International Conference on Medical Image Computing & Computer Assisted Intervention* (MICCAI), 2018.

Yao Qin*, Mengyang Feng*, Huchuan Lu and Garrison Cottrell. "Hierarchical Cellular Automata for Visual Saliency", in *International Journal of Computer Vision* (IJCV), 2017. (* Equal contribution)

Yao Qin, Dongjin Song, Haifeng Chen, Wei Cheng, Guofei Jiang and Garrison Cottrell. "A Dual-Stage Attention-Based Recurrent Neural Network for Time Series Prediction", in *International Joint Conference on Artificial Intelligence* (IJCAI), 2017.

Qiuhui Pan, **Yao Qin**, Yiqun Xu, Mengfei Tong, Mingfeng He. "Opinion Evolution in Open Community", in *International Journal of Modern Physics C, 1750003*, 2016.

Yao Qin, Huchuan Lu, Yiqun Xu and He Wang. "Saliency Detection via Cellular Automata", in *Conference on Computer Vision and Pattern Recognition* (CVPR), 2015.

ABSTRACT OF THE DISSERTATION

Detecting, Diagnosing, Deflecting and Designing Adversarial Attacks

by

Yao Qin

Doctor of Philosophy in Computer Science

University of California San Diego, 2020

Professor Garrison Cottrell, Chair

There has been an ongoing cycle between stronger attacks and stronger defenses in the adversarial machine learning game. However, most of the existing defenses are subsequently broken by a more advanced defense-aware attack. This dissertation first introduces a stronger detection mechanism based on Capsule networks which achieves state-of-the-art detection performance on both standard and defense-aware attacks. Then, we diagnose the adversarial examples against our CapsNet and find that the success of the adversarial attack is proportional to the visual similarity between the source and target class (which is not the case for CNN-based networks). Pushing this idea further, we show how it is possible to pressure the attacker to produce an input that visually resembles the attacks target class, thereby deflecting the attack.

These deflected attack images thus can no longer be called adversarial, as our network classifies them the same way as humans do. The existence of the deflected adversarial attacks also indicates the ℓ_p norm is not sufficient to ensure the same semantic class. Finally, this dissertation discusses how to design adversarial attacks for speech recognition systems based on human perception rather than the ℓ_p-norm metric.

Chapter 1

Introduction

Adversarial examples are inputs to machine learning models that are specifically designed by an adversary to cause an incorrect output [63, 21]. Initial work on adversarial examples focused mainly on the domain of image classification, where adversarial examples can be constructed by imperceptibly modifying images to cause misclassification, and are practical in the physical world [33]. Since the discovery of the existence of the adversarial examples, some research work focused on either the creation of more robust models to **defend** against adversarial attacks [61, 41, 71], or designing a robust detection mechanism to **detect** adversarial attacks [22, 16, 43, 37, 50, 52]. Specifically, to defend against adversarial attacks is to correctly classified the adversarial input as the original class rather than the adversarial target class. Instead, the detection algorithms aim to distinguish adversarial attacks from real data and then flag the adversarial input. However, better defenses have led to the develpment of stronger attack algorithms to break these defenses [41, 10]. After several iterations of creating and breaking defenses, many state-of-the-art defenses and detection methods [52, 40, 37, 26] were broken subsequently with a more advanced defense-aware attack [25, 9, 2].

To end this ongoing cycle between stronger attacks and stronger defenses in the adversarial machine learning game, we first propose to detect adversarial examples or otherwise corrupted out-of-distribution images based on a class-conditional reconstruction of the input in Chapter 2. Then, to attack our own detection mechanism, we propose the Reconstructive Attack, which seeks both to cause a misclassification and a low reconstruction error. This reconstructive attack produces undetected adversarial examples but with much smaller success rate compared to the standard attacks. Among all these attacks, we find that Capsule networks (CapsNets) always perform better than convolutional networks.

Since adversarial examples raise questions about whether neural network models are sensitive to the same visual features as humans, we then diagnose the adversarial examples for CapsNets in Chapter 3 and find that the success of the reconstructive attack was proportional to the visual similarity between the source and target class. Additionally, the resulting perturbations can cause the input image to appear visually more like the target class and hence become nonadversarial. This is not the case for the CNN-based models. These extensive qualitative studies suggest that CapsNets use features that are more aligned with human perception and might have the potential to address the central issue raised by adversarial examples.

In Chapter 4, we present a new direction in defenses, which we argue is a step towards ending this cycle by **deflecting** adversarial attacks, i.e. by forcing the attacker to produce an input which visually resembles the adversarial target class. As we know for gradient-based attacks, the attackers are following the gradient with regard to the input to construct the adversarial example. Our deflecting model based on Capsule networks is able to force the attacker to follow the gradients pointing towards changing the perceptual class rather than an imperceptible adversarial noise. To this end, we first propose a stronger defense mechanism which combines three detection mechanisms to achieve state-of-the-art detection performance on both standard and defense-aware attacks. We then show that undetected attacks against our deflecting model are often classified as the adversarial target class by performing a human study where participants are asked to label the class of images produced by the attack. These attack images thus can no longer be called adversarial, as our network classifies them the same way as humans do.

In order to differentiate properties of adversarial examples on neural networks in general from properties which hold true only on images, it is important to study adversarial examples in different domains. So far, adversarial examples are known to exist on various domains, including image classification [5], speech recognition [11], reinforcement learning [26] and reading comprehension [29]. In Chapter 5, we focus on studying the adversarial example in the audio domain where Carlini and Wagner's work [11] showed that any given source audio sample can be perturbed slightly to attack an automatic speech recognition system into making a targeted transcription. In contrast to the adversarial examples in the image domain, current targeted adversarial examples applied to speech recognition systems are neither imperceptible nor practical in the real world: humans can easily identify the adversarial perturbations, and they are not effective when played over-the-air. In Chapter 5, we make advances on both of these fronts. First, we develop effectively imperceptible audio adversarial examples (verified through a human study) by leveraging the psychoacoustic principle of auditory masking, while retaining 100% targeted success rate on arbitrary full-sentence targets. Next, we make progress towards physical-world over-the-air audio adversarial examples by constructing perturbations which remain effective even after applying realistic simulated environmental distortions.

In Chapter 6, we summarize the contributions of this thesis and discuss the promising future work related to adversarial examples in order to increase the security of AI systems and to help us understand deep neural networks.

Chapter 2

Detecting Adversarial Attacks

2.1 Introduction

Adversarial examples [64] are inputs that are designed by an adversary to cause a machine learning system to make a misclassification. A series of studies on adversarial attacks has shown that it is easy to cause misclassifications using visually imperceptible changes to an image under ℓ_p-norm based similarity metrics [21, 34, 41, 10]. Since the discovery of adversarial examples, there has been a constant "arms race" between better attacks and better defenses. Many new defenses have been proposed [61, 20, 22, 43], only to be broken shortly thereafter [9, 2]. Currently, the most effective approach to reduce network's vulnerability to adversarial examples is "adversarial training", in which a network is trained on both clean images and adversarially perturbed ones [21, 41]. However, adversarial training is very time-consuming because it requires generating adversarial examples during training. It also typically only helps improve a network's robustness to adversarial examples that are generated in a similar way to those on which the network was trained. [24] showed that capsule models are more robust to simple adversarial attacks than CNNs but [44] showed that this is not the case for all attacks.

The cycle of attacks and defenses motivates us to rethink both how we can improve the general robustness of neural networks as well as the high-level motivation for this pursuit. One potential path forward is to detect adversarial inputs, instead of attempting to accurately classify them [58, 52]. Recent works [28, 18] argue that adversarial examples can exist within

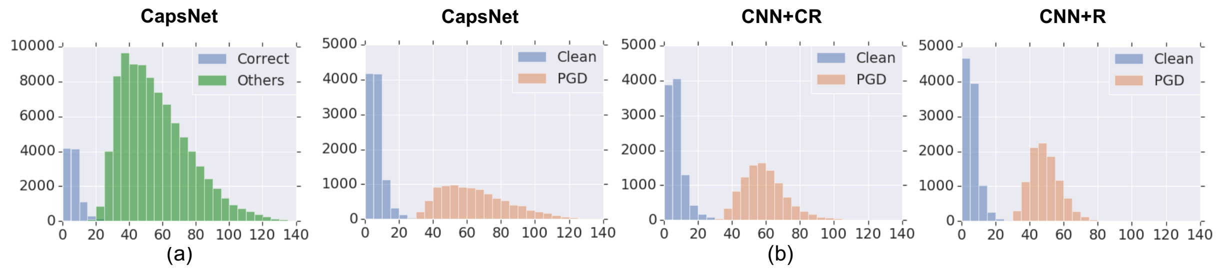


Figure 2.1. (a) The histogram of ℓ_2 distances between the input and the reconstruction using the correct capsule or other capsules in CapsNet on the real MNIST images. Notice the stark difference between the distributions of reconstructions of the capsule corresponding to the correct class and other capsules. (b) The histograms of ℓ_2 distances between the reconstruction and the input for real and adversarial images for the three models explored in this chapter on the MNIST dataset. We use PGD [41] with the ℓ_∞ bound $\varepsilon = 0.3$ to create the attacks.

the data distribution, which implies that detecting adversarial examples based on an estimate of the data distribution alone might be insufficient. Instead, in this chapter we develop methods for ***detecting*** adversarial examples by making use of *class-conditional reconstruction networks*. These sub-networks, first proposed by [54] as part of a Capsule Network (CapsNet), allow a model to produce a reconstruction of its input based on the identity and instantiation parameters of the winning capsule. Interestingly, we find that reconstructing an input from the capsule corresponding to the correct class results in a much lower reconstruction error than reconstructing the input from capsules corresponding to incorrect classes, as shown in Figure 2.1(a). Motivated by this, we propose using the reconstruction sub-network in a CapsNet as an attack-independent detection mechanism. Specifically, we reconstruct a given input from the resulting capsule pose parameters of the winning capsule and then detect adversarial examples by comparing the difference between the reconstruction distributions for natural and adversarial (or otherwise corrupted) images.

We extend this detection mechanism to standard convolutional neural networks (CNNs) and show its effectiveness against black box and white box attacks on three image datasets; MNIST, Fashion-MNIST and SVHN. We show that capsule models achieve the strongest attack detection rates and accuracy on these attacks. We then test our method against a stronger attack, the Reconstructive Attack, specifically designed to attack our detection mechanism by generating adversarial examples with a small reconstruction error. With this attack we are able to create undetected adversarial examples, but we show that this attack is less successful in fooling the classifier than a non-reconstructive attack.

In this chapter, we mainly focus on:

- proposing a class-conditional capsule reconstruction based detection method to detect standard white-box/black-box adversarial examples on multiple datasets. This detection mechanism is attack-agnostic and is successfully extended to standard convolutional neural networks.

- testing our detection mechanism on the corrupted MNIST dataset and show that it can work as a general out-of-distribution detector.

- designing the reconstructive attack, which is specifically designed to attack our detection mechanism but becomes less successful in fooling the classifier.

2.2 Related Work

Adversarial examples were first introduced in [5, 64], where a given image was modified by following the gradient of a classifier's output with respect to the image's pixels. [21] then developed the more efficient Fast Gradient Sign method (FGSM), which can change the label of the input image X with a similarly imperceptible perturbation which is constructed by taking an ε step in the direction of the gradient. Later, the Basic Iterative Method (BIM) [34] and Project Gradient Descent [41] can generate stronger attacks improved on FGSM by taking multiple steps in the direction of the gradient. In addition, [10] proposed another iterative optimization-based method to construct strong adversarial examples with small perturbations.

An early approach to reducing vulnerability to adversarial examples was proposed by [21], where a network was trained on both clean images and adversarially perturbed ones. Since then, there has been a constant “arms race” between better attacks and better defenses; [35] provide an overview of this field. However, many defenses against adversarial examples have been demonstrated to be an effect of “obfuscated gradients” and can be further circumvented under the white-box setting [2].

A recent thread of research focuses on the generation of (and defense against) adversarial examples which are not simply slightly-perturbed versions of clean images. For example, several approaches were proposed which utilize generative models to create novel images which appear realistic but which result in a misclassification [55, 27, 42]. These adversarial images are not imperceptibly close to some existing image, but nevertheless resemble members of the data distribution to humans and are strongly misclassified by neural networks. [53] also consider adversarial examples which are not the result of pixel-space perturbations by manipulating the hidden representation of a neural network in order to generate an adversarial example. Also they show that adversaries exist for a network with random weights. Therefore, susceptibility to adversarial attacks is not caused by learning and the convolution neural network architectures are fragile.

Another line of work attempts to circumvent adversarial examples by detecting them with a separately-trained classifier [20, 22, 43] or using statistical properties [23, 38, 16, 22]. However, many of these approaches were subsequently shown to be flawed [9, 2]. The most recent work in detecting adversarial examples [52] that has 99% true positive rate in CIFAR10 dataset [32] has also been fully bypassed by later work [25] which decreased the true positive rate to less than 2%.

Similar to our work, [58] also investigated the effectiveness of a class-conditional generative model as a defense mechanism for MNIST digits. However, we differ in some important ways. Their model is in some ways the opposite of ours - they first attempt to generate the input, and then make a classification on the resultant generations, whereas our method attempts to first classify the input, making use of an otherwise unchanged capsule classification model, and then generates the input from a high level representation. As such our method does not increase the computational overhead of classifying the input, compared to the approach of [58], which results in a 10x increase in computation. In addition, the work of [58] is only applied to MNIST, so our results on the more complex datasets represent an improvement.

2.3 Preliminaries

2.3.1 Adversarial Examples

Given a clean test image x, its corresponding label y, and a classifier $f(\cdot)$ which predicts a class label given an input, we refer to $x' = x + \delta$ as an adversarial example if it is able to fool the classifier into making a wrong prediction $f(x') \neq f(x) = y$. The small adversarial perturbation δ (where "small" is measured under some norm) causes the adversarial example x' to appear visually similar to the clean image x but to be classified differently. In the unrestricted case where we only require that $f(x') \neq y$, we refer to x' as an "untargeted adversarial example". A more powerful attack is to generate a "targeted adversarial example": instead of simply fooling the classifier to make a wrong prediction, we force the classifier to predict some targeted label $f(x') = t \neq y$. In this chapter, the target label t is selected uniformly at random as any label which is not the ground-truth correct label. As is standard practice in the literature, we test our detection mechanism on three ℓ_∞ norm based attacks (fast gradient sign method (FGSM) [21], the basic iterative method (BIM) [34], projected gradient descent (PGD) [41]) and one ℓ_2 norm based attack (Carlini-Wagner (CW) [10]).

2.3.2 Capsule Networks

Capsule Networks (CapsNets) are an alternative architecture for neural networks [54, 24]. In this work we make use of the CapsNet architecture detailed by [54]. Unlike a standard neural network which is made up of layers of scalar-valued units, CapsNets are made up of layers of capsules, which output a vector or matrix. Intuitively, just as one can think of the activation of a unit in a normal neural network as the presence of a feature in the input, the activation of a capsule can be thought of as both the presence of a feature and the pose parameters that represent attributes of that feature. A top-level capsule in a classification network therefore outputs both a classification and pose parameters that represent the instance of that class in the input. This high level representation allows us to train a reconstruction network.

2.3.3 Threat Model

In this chapter, we test our detection mechanism against both white-box and black-box attacks. For white-box attacks, the adversary has full access to the model as well as its parameters. In particular, the adversary is allowed to compute the gradient through the model to generate adversarial examples. To perform black-box attacks, the adversary is allowed to know the network architecture but not its parameters. Therefore, we retrain a substitute model that has the same architecture as the target model and generate adversarial examples by attacking the substitute model. Then we transfer these attacks to the target model. For ℓ_∞ based attacks, we always control the ℓ_∞ norm of the adversarial perturbation to be within a relatively small bound ε_∞, specific to each dataset.

2.4 Detecting Adversarial Images by Reconstruction

2.4.1 Models

To detect adversarial images, we make use of the reconstruction network proposed in [54], which takes pose parameters v as input and outputs the reconstructed image $r(v)$. The reconstruction network is simply a fully connected neural network with two ReLU hidden layers with 512 and 1024 units respectively, with a sigmoid output with the same dimensionality as the dataset. The reconstruction network is trained to minimize the ℓ_2 distance between the input image and the reconstructed image. This same network architecture is used for all the models and datasets we explore. The only difference is what is given to the reconstruction network as input.

- **CapsNet**

  The reconstruction network of the CapsNet is *class-conditional*: It takes in the pose parameters of all the class capsules and masks all values to 0 except for the pose parameters of the predicted class. We use this reconstruction network for detecting adversarial attacks by measuring the Euclidean distance between the input and a class conditional reconstruction. Specifically, for any given input x, the CapsNet outputs a prediction $f(x)$ as well as the pose parameters v for all classes. The reconstruction network takes in the pose parameters and then selects the pose parameter corresponding to the predicted class, denoted as $v_{f(x)}$, to generate a reconstruction $r(v_{f(x)})$. Then we compute the ℓ_2 reconstruction distance $d(x) = \|r(v_{f(x)}), x\|_2$ between the reconstructed image and the input image, and compare it with a pre-defined detection threshold p (described below in Section 2.4.2). If the reconstruction distance $d(x)$ is higher than the detection threshold p, we flag the input as an adversarial example. Figure 2.1 (b) shows example histograms of reconstruction distances for natural images and typical adversarial examples.

- **CNN+CR**

  Although our strategy is inspired by the reconstruction networks used in CapsNets, the strategy can be extended to standard convolutional neural networks (CNNs). We create a similar architecture, CNN with conditional reconstruction (CNN+CR), by dividing the penultimate hidden layer of a CNN into groups corresponding to each class. The sum of each neuron group serves as the logit for that particular class and the group itself serves the same purpose as the pose parameters in the CapsNet. We use the same masking mechanism as [54] to select the pose parameter corresponding to the predicted label $v_{f(x)}$ and generate the reconstruction based on the selected pose parameters. In this way we extend the class-conditional reconstruction network to standard CNNs.

- **CNN+R**

  We can also create a more naïve implementation of our strategy by simply computing the reconstruction from the activations in the entire penultimate layer without any masking mechanism. We call this model the ”CNN+R” model. In this way we are able to study the effect of conditioning on the predicted class.

2.4.2 Detection Threshold

We find the threshold θ for detecting adversarial inputs by measuring the reconstruction error between a validation input image and its reconstruction. If the distance between the input and the reconstruction is above the chosen threshold θ, we classify the data as adversarial. Choosing the detection threshold θ involves a trade-off between false positive and false negative detection rates. The optimal threshold depends on the probability of the system being attacked. Such a trade-off is discussed by [17]. In our experiments we don't tune this parameter and simply set it as the 95th percentile of validation distances. This means our false positive rate on real validation data is 5%.

2.5 Experiments

In this section, we first introduce the evaluation metrics, test models and datasets that are used to evaluate the performance of our models. Then, we explain the implementation details to generate the adversarial attacks. Next, we demonstrate how reconstruction networks can detect standard white-box and naturally corrupted images. In addition, we introduce the “reconstructive attack”, which is specifically designed to circumvent our defense and show that it is a more powerful attack in this setting. We also report the performance of our detection methods towards black-box standard and reconstructive attacks. Finally, we show the detection performance on the CIFAR-10 dataset and discuss the effectiveness of the class-conditional information in the reconstruction network.

2.5.1 Evaluation Metrics

We use **Accuracy** to represent the proportion of clean examples that are correctly classified. In addition, We use **Success Rate** to measure the success of attacks. For targeted attacks, the success rate S_t is defined as the proportion of inputs which are classified as the target class, $S_t = \frac{1}{N}\sum_i^N (f(x_i') = t_i)$, while the success rate for untargeted attacks is defined as the proportion of inputs which are misclassified, $S_u = \frac{1}{N}\sum_i^N (f(x_i') \neq y_i)$. Previous works [9, 25] used **True Positive Rate** to measure the proportion of adversarial examples that are detected, which alone is insufficient to measure the ability of different detection mechanism because the unsuccessful adversarial examples do not have to be detected. Therefore, we propose to use **Undetected Rate**: the proportion of attacks that are successful and undetected to evaluate the detection mechanism. For targeted attacks, the undetected rate is defined as $R_t = \frac{1}{N}\sum_i^N (f(x_i') = t_i) \cap (d(x_i') \leq \theta)$, where $d(\cdot)$ computes the reconstruction distance of the input and p denotes the detection threshold introduced in section 2.4.2. Similarly, the undetected rate for untargeted attacks R_u can be defined as $R_u = \frac{1}{N}\sum_i^N (f(x_i') \neq y_i) \cap (d(x_i') \leq \theta)$. The smaller the undetected rate R_t or R_u is, the stronger the model is in detecting adversarial examples. The undetected rate can also be used to evaluate the attacks (higher is better). We also plot the **Undetected Rate vs. False Positive Rate** curve to compare the detection performance between different models, where **False Positive Rate** is defined as the proportion of clean examples that are misclassified as the adversarial example by the detection method.

2.5.2 Test Models and Datasets

In all experiments, all three models (CapsNet, CNN+R, and CNN+CR) have the same number of parameters and were trained with Adam [31] for the same number of epochs. The details of the model architectures on three datasets: MNIST [36], FashionMNIST [68], and SVHN [47] are shown in Figure 2.2. MNIST and FashionMNIST have exactly the same architectures while we use larger models for SVHN. Note that the only difference between

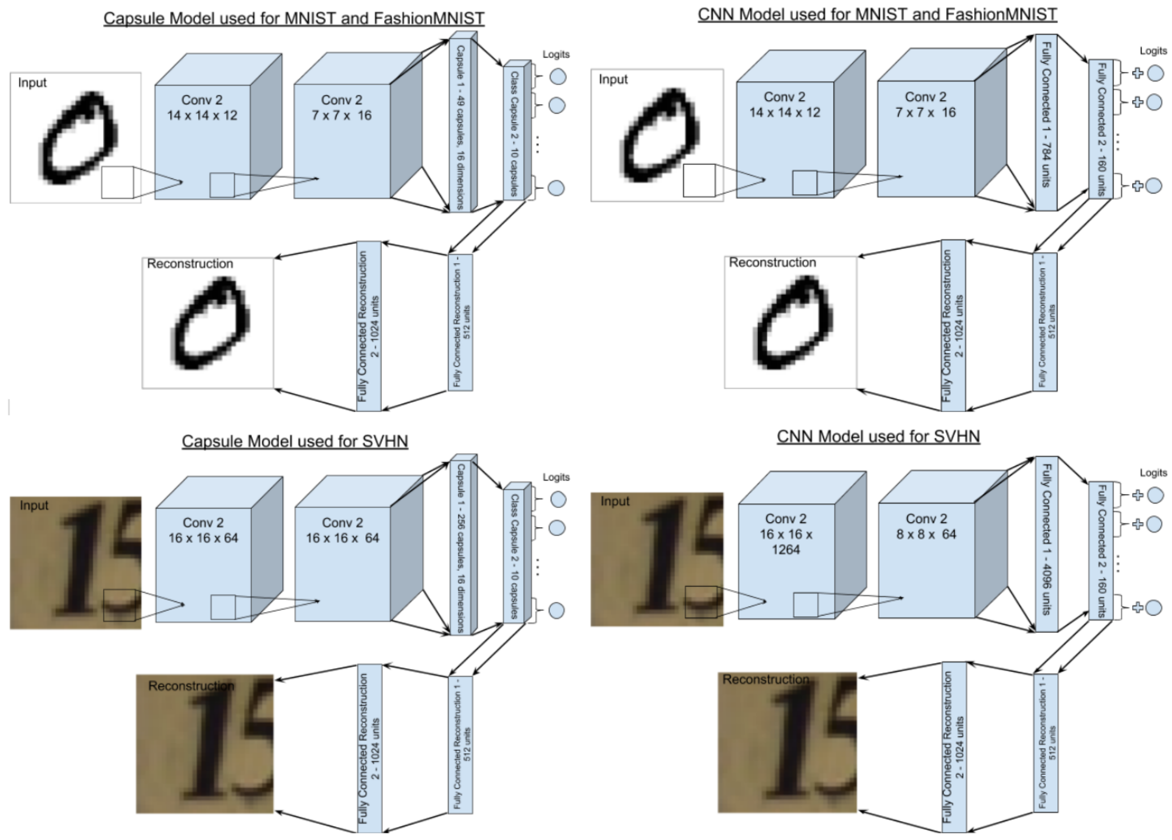

Capsule Model used for MNIST and FashionMNIST
Input
Conv 2 14 x 14 x 12
Conv 2 7 x 7 x 16
Capsule 1 - 49 capsules, 16 dimensions
Class Capsule 2 - 10 capsules
Logits
Reconstruction
Fully Connected Reconstruction 2 - 1024 units
Fully Connected Reconstruction 1 - 512 units
CNN Model used for MNIST and FashionMNIST
Input
Conv 2 14 x 14 x 12
Conv 2 7 x 7 x 16
Fully Connected 1 - 784 units
Fully Connected 2 - 160 units
Logits
Reconstruction
Fully Connected Reconstruction 2 - 1024 units
Fully Connected Reconstruction 1 - 512 units
Capsule Model used for SVHN
Input
Conv 2 16 x 16 x 64
Conv 2 16 x 16 x 64
Capsule 1 - 256 capsules, 16 dimensions
Class Capsule 2 - 10 capsules
Logits
Reconstruction
Fully Connected Reconstruction 2 - 1024 units
Fully Connected Reconstruction 1 - 512 units
CNN Model used for SVHN
Input
Conv 2 16 x 16 x 1264
Conv 2 8 x 8 x 64
Fully Connected 1 - 4096 units
Fully Connected 2 - 160 units
Logits
Reconstruction
Fully Connected Reconstruction 2 - 1024 units
Fully Connected Reconstruction 1 - 512 units

Figure 2.2. The architecture for the CapsNet, CNN+R and CNN+CR model used for our experiments on MNIST [36], FashionMNIST [68], and SVHN [47].

the CNN reconstruction (CNN+R) and the CNN conditional reconstruction (CNN+CR) is the masking procedure on the input to the reconstruction network based on the predicted class.

We display the test error rate for each model on these three datasets in Table 2.1. In general, all models achieved similar test accuracy. We did not do an exhaustive hyperparameter search on these models, instead we chose hyperparameters that allowed each model to perform roughly equivalently on the test sets.

2.5.3 Implementation Details

For all the ℓ_∞ based adversarial examples, the ℓ_∞ norm of the perturbations is bound by ε, which is set to 0.3, 0.1, 0.1 for MNIST, Fashion MNIST and SVHN dataset respectively following previous works [41, 61]. In FGSM based attacks, the step size c is 0.05. In BIM-based

Table 2.1. The test accuracy of each model when the input are clean test images in each dataset.

Dataset	CapsNet	CNN+CR	CNN+R
MNIST	99.4%	99.3%	99.4%
FashionMNIST	90.4%	90.5%	90.7%
SVHN	89.3%	90.7%	90.5%

[34] and PGD-based [41] attacks, the step size c is 0.01 for all the datasets and the number of iterations are 1000, 500 and 200 for MNIST, Fashion MNIST and SVHN dataset respectively. We choose a sufficiently large number of iterations to ensure the attacks has converged.

We use the publicly released code from the authors of [10] to perform the CW attack for our models. The number of iterations are set to 1000 for all three datasets.

Table 2.2. Success rate and undetected rate of white-box targeted and untargeted attacks. In the table, S_t/R_t is shown for targeted attacks and S_u/R_u is presented for untargeted attacks.

Networks	Targeted (%)				Untargeted (%)			
	FGSM	BIM	PGD	CW	FGSM	BIM	PGD	CW
MNIST Dataset								
CapsNet	3/0	82/0	86/0	99/2	11/0	99/0	99/0	100/19
CNN+CR	16/0	93/0	95/0	89/8	85/0	100/0	100/0	100/28
CNN+R	37/0	100/0	100/0	100/47	64/0	100/0	100/0	100/63
FASHION MNIST Dataset								
CapsNet	7/5	54/9	55/10	100/26	35/29	86/50	87/51	100/68
CNN+CR	19/13	89/28	89/28	87/37	74/33	100/25	100/24	100/72
CNN+R	23/16	98/19	98/19	99/81	62/48	100/35	100/34	100/87
SVHN Dataset								
CapsNet	22/20	83/45	84/46	100/90	74/67	99/70	99/68	100/94
CNN+CR	24/23	99/90	99/90	99/93	87/82	100/90	100/89	100/90
CNN+R	26/24	100/86	100/86	100/94	88/82	100/92	100/92	100/95

2.5.4 Standard White-box Attacks

We present the success and undetected rates for several targeted and untargeted attacks on MNIST, FashionMNIST, and SVHN in Table 2.2. Our method is able to accurately detect many

Table 2.3. Error rate/undetected rate on the Corrupted MNIST dataset.

Corruption	**Clean**	**Gaussian Noise**	**Gaussian Blur**	**Line**	**Dotted Line**	**Elastic Transform**
CapsNet	0.6/0.2	12.1/0.0	10.3/4.1	19.6/0.1	4.3/0.0	11.3/0.8
CNN+CR	0.7/0.3	9.8/0.0	6.7/4.2	17.6/0.1	4.2/0.0	11.1/1.1
CNN+R	0.6/0.4	6.7/0.0	8.9/6.4	18.9/0.1	3.1/0.0	12.2/2.1
Corruption	**Saturate**	**JPEG**	**Quantize**	**Sheer**	**Spatter**	**Rotate**
CapsNet	3.5/0.0	0.8/0.4	0.7/0.1	1.6/0.4	1.9/0.2	6.5/2.2
CNN+CR	1.5/0.0	0.8/0.5	0.9/0.1	2.1/0.4	1.8/0.4	6.1/1.6
CNN+R	1.2/0.0	0.7/0.5	0.7/0.2	2.2/0.7	1.8/0.4	6.5/3.4
Corruption	**Contrast**	**Inverse**	**Canny Edge**	**Fog**	**Frost**	**Zigzag**
CapsNet	92.0/0.0	91.0/0.0	21.5/0.0	83.7/0.0	70.6/0.0	16.9/0.0
CNN+CR	72.0/32.6	78.1/0.0	34.6/0.0	66.0/0.5	37.6/0.0	18.4/0.0
CNN+R	73.4/49.4	88.1/0.0	23.4/0.0	65.6/0.1	36.2/0.0	17.5/0.0

attacks with very low undetected rates. Capsule models almost always have the lowest undetected rates out of our three models. It is worth noting that this method performs best with the simplest dataset, MNIST, and that the highest undetected rates are found with the Carlini-Wagner attack on the SVHN dataset. This illustrates both the strength of this attack and a shortcoming of our defense, namely that our detection mechanism relies on ℓ_2 image distance as a proxy for visual similarity, and in the case of higher dimensional color datasets such as SVHN, this proxy is less meaningful.

2.5.5 Corruption Attacks

Recent work has argued that improving the robustness of neural networks to ℓ_p norm bounded adversarial attacks should not come at the expense of increasing error rates under distributional shifts that do not affect human classification rates and are likely to be encountered in the "real-world" [17]. For example, if an image is corrupted due to adverse weather, lighting, or occlusion, we might hope that our model can continue to provide reliable predictions or detect the distributional shift. We can test our detection method on its ability to detect these distributional shifts by making use of the Corrupted MNIST dataset [46]. This data set contains many visual transformations of MNIST that do not seem to affect human performance, but nevertheless are strongly misclassified by state-of-the-art MNIST models. Our three models can almost always detect these distributional shifts (in all corruptions CapsNets have either a small undetected rate or an undetected rate of 0). The error rate (the proportion of misclassified input) and undetected rate of three test models on the Corrupted MNIST dataset is shown in Table 2.3. We also visualize the examples from Corrupted MNIST dataset [46] and the corresponding reconstructed images for each model in Figure 2.3 and Figure 2.4.

2.5.6 Reconstructive Attacks

Thus far we have only evaluated previously-defined attacks. Following the suggestion in [9] that detection methods need to show effectiveness towards defense-aware attacks, we introduce an attack specifically designed to take into account our defense mechanism. In order to construct adversarial examples that cannot be detected by the network, we propose a two-stage optimization method to generate a “reconstructive attack”. Specifically, in each step, we first attempt to fool the network by following a standard attack which computes the gradient of the cross-entropy loss function with respect to the input. Then, in the second stage, we take the reconstruction error into account by updating the adversarial perturbation based on the ℓ_2 reconstruction loss. In this way, we endeavor to construct adversarial examples that can fool the network and also have a small ℓ_2 reconstruction error. The untargeted and targeted reconstructive attacks based are described in more detail below.

Untargeted Reconstructive Attacks

To construct untargeted reconstructive attacks, we first update the perturbation based on the gradient of the cross-entropy loss function following a standard FGSM attack [21], that is:

$$\delta \leftarrow \text{clip}_{\varepsilon}(\delta + c \cdot \beta \cdot \text{sign}(\nabla_{\delta} \ell_{net}(f(x+\delta), y))), \tag{2.1}$$

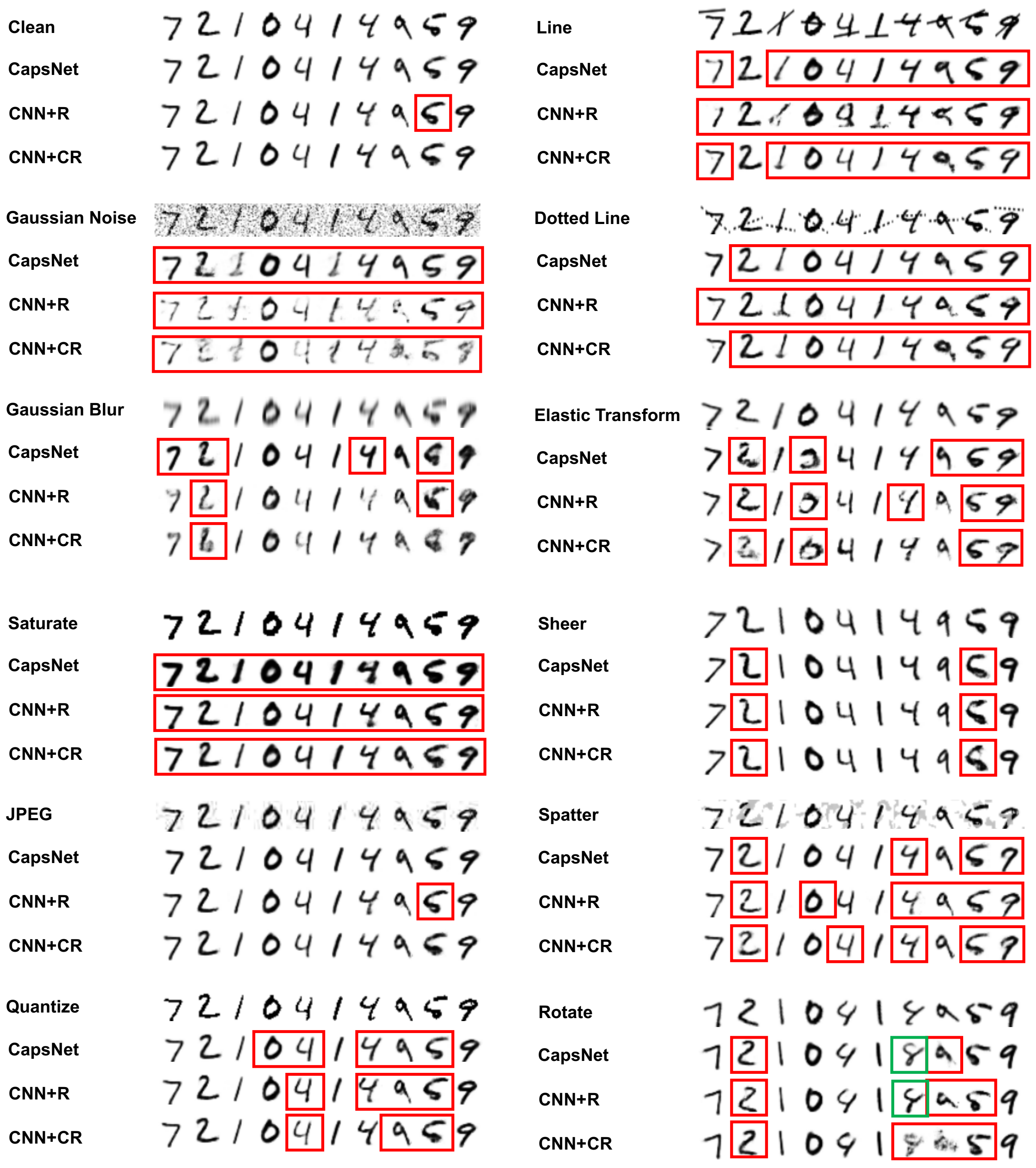


Figure 2.3. Examples of Corrupted MNIST and the reconstructed image for each model. A red box represents that this input is flagged as an adversarial example while a green box represents that this input has been misclassified and not been detected.

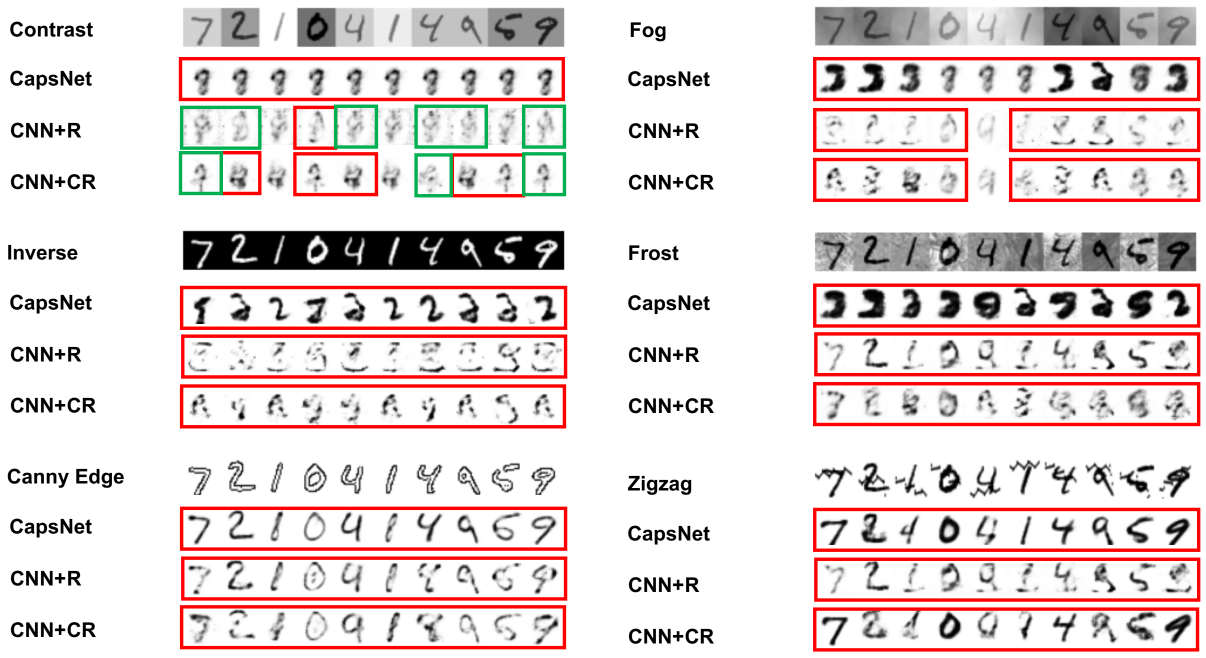


Figure 2.4. Examples of Corrupted MNIST and the reconstructed image for each model (continued). A red box represents that this input is flagged as an adversarial example while a green box represents that this input has been misclassified and not been detected.

where $\ell_{net}(f(\cdot),y)$ is the cross-entropy loss function, ε is the ℓ_∞ bound for our attacks, c is a hyperparameter controlling the step size in each iteration and β is a hyperparameter which balances the importance of the cross-entropy loss and the reconstruction loss (explained further below). In the second stage, we focus on constraining the reconstructed image from the newly predicted label to have a small reconstruction distance by updating δ according to

$$\delta \leftarrow \text{clip}_\varepsilon(\delta - c \cdot (1-\beta) \cdot \text{sign}(\nabla_\delta(\|r(v_{f(x+\delta)}) - (x+\delta)\|_2))), \tag{2.2}$$

where $r(v_{f(x+\delta)})$ is the class-conditional reconstruction based on the predicted label $f(x+\delta)$ in a CapsNet or CNN+CR network. The δ used here is the optimized δ from the first stage. $\|r(v_{f(x+\delta)}) - (x+\delta)\|_2$ is the ℓ_2 reconstruction distance between the reconstructed image and the input image. Since the CNN+R network does not use the class conditional reconstruction, we simply use the reconstructed image without the masking mechanism. According to Eqn 2.1

and Eqn 2.2, we can see that β balances the importance between the success rate of attacks and the reconstruction distance. This hyperparameter was tuned for each model and each dataset in order to create the strongest attacks.

Targeted Reconstructive Attacks

We perform a similar two-stage optimization to construct targeted reconstructive attacks, by defining a target label and attempting to maximize the classification probability of this label, and minimize the reconstruction error from corresponding capsule. In the first stage, we use a standard targeted attack to update the perturbation by computing the gradient of the cross-entropy loss function. Specifically, in the first stage of each step, we update the adversarial perturbation via:

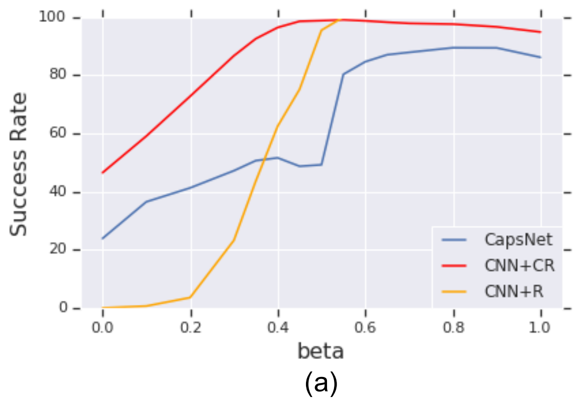
$$\delta \leftarrow \text{clip}_{\varepsilon}(\delta - c \cdot \beta \cdot \text{sign}(\nabla_{\delta} \ell_{net}(f(x+\delta), y))). \tag{2.3}$$

Then in the second stage, we focus on constraining the ℓ_2 distance to be small between the reconstructed and the input image. Here, we use the capsule corresponding to the targeted label to perform the reconstruction, that is:

$$\delta \leftarrow \text{clip}_{\varepsilon}(\delta - c \cdot (1-\beta) \cdot \text{sign}(\nabla_{\delta}(\|r_y(x+\delta) - (x+\delta)\|_2)), \tag{2.4}$$

where $r_y(x+\delta)$ is the reconstructed image that using label y to mask out the capsules/layers corresponding to all the other classes in CapsNet and CNN+CR models. Instead, CNN+R network uses the entire penultimate layer for reconstruction. Similarly, β is to balance the importance between the success rate and the detected rate of the constructed adversarial example. Figure 2.5 shows the plot of success rate and undetected rate versus the hyperparameter β which balances the importance between success rate and undetected rate in the targeted reconstructive PGD attacks on the MNIST dataset.

Because the targeted label is given, another way to construct targeted reconstructive attacks is to combine these two stages into one stage via minimizing the loss function $\ell =$



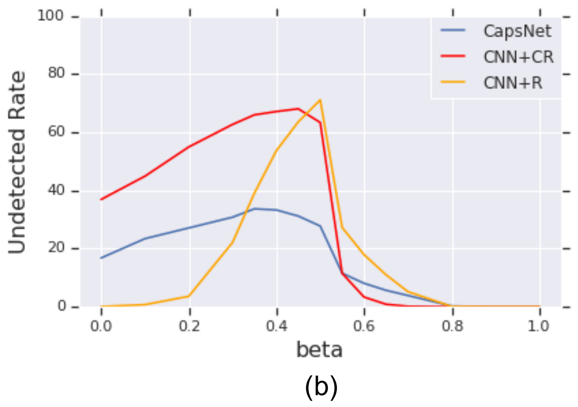


Figure 2.5. An example shows the plot of the success rate in (a) and undetected rate in (b) of targeted reconstructive PGD attack vesus the hyperparameter beta β for each model on the MNIST test set. We set the max ℓ_∞ norm $\varepsilon = 0.3$ to create the attacks.

$\beta \cdot \ell_{net}(f(x+\delta), y) + (1-\beta) \cdot \|r(v_{f(x+\delta)}) - (x+\delta)\|_2$. We implemented both of these targeted reconstructive attacks and found that the two-stage version is a stronger attack. Therefore, all the Reconstructive Attack experiments performed in this chapter are based on two-stage optimization.

We build our reconstructive attack based on the standard PGD attack, denoted as R-PGD, and test the performance of our detection models against this reconstructive attack in a white-box setting. Comparing Table 2.2 and Table 2.4, we can see that the Reconstructive Attack is significantly less successful at changing the models prediction (lower success rates than the standard attack). However, this attack is more successful at fooling our detection method. For all attacks and datasets the capsule model has the lowest attack success rate and the lowest undetected rate.

In addition, we report the undetected rate of the white-box targeted defense-aware R-PGD attack versus the False Positive Rate on the MNIST, Fashion-MNIST and SVHN datasets in Figure 2.6. We can clearly see that the undetected rate of the defense-aware attack against CapsNet is significantly smaller than the CNN-based networks, which suggests that CapsNets are more robust against adversarial attacks. Furthermore, CNN with class-conditional reconstruction (CNN+CR) has smaller undetected rate at the same False Positive Rate compared to the

Table 2.4. Success rate and the **worst case** undetected rate of white-box targeted and untargeted reconstructive attacks. Below S_t/R_t is shown for targeted attacks and S_u/R_u is presented for untargeted attacks.

Networks	Targeted (%)			Untargeted (%)		
	R-FGSM	R-BIM	R-PGD	R-FGSM	R-BIM	R-PGD
MNIST Dataset						
CapsNet	1.8/0.3	51.0/33.8	50.7/33.7	6.1/1.0	84.5/35.1	88.1/37.9
CNN+CR	7.6/0.5	98.0/68.1	98.6/68.1	41.7/3.2	96.5/86.8	99.4/87.7
CNN+R	16.9/3.3	86.3/65.9	95.5/71.2	25.9/8.1	82.9/67.8	95.1/70.5
FASHION MNIST Dataset						
CapsNet	6.5/5.8	53.3/28.4	53.7/29.8	33.3/29.9	85.3/75.9	84.9/75.5
CNN+CR	17.7/14.0	80.3/72.4	78.1/72.0	68.0/57.3	89.8/84.4	91.5/86.0
CNN+R	19.4/17.6	95.2/88.8	94.6/88.4	58.6/53.5	98.8/90.1	98.9/90.0
SVHN Dataset						
CapsNet	21.6/21.2	81.1/78.3	82.0/79.2	71.6/68.3	98.9/97.5	98.9/97.5
CNN+CR	24.2/22.6	98.5/97.6	99.0/97.9	86.0/82.3	99.9/99.5	99.9/99.5
CNN+R	26.6/25.8	99.6/99.4	99.5/99.3	87.1/84.5	100.0/99.9	100.0/99.9

CNN without class-conditional reconstruction (CNN+R), which suggests the class-conditional information is helpful in our models to improve the robustness against adversarial attacks.

2.5.7 Black-box Attacks

We also tested our detection mechanism results on black box attacks, shown in Table 2.5. Given the low undetected rates in the white-box settings, it is not surprising that our detection

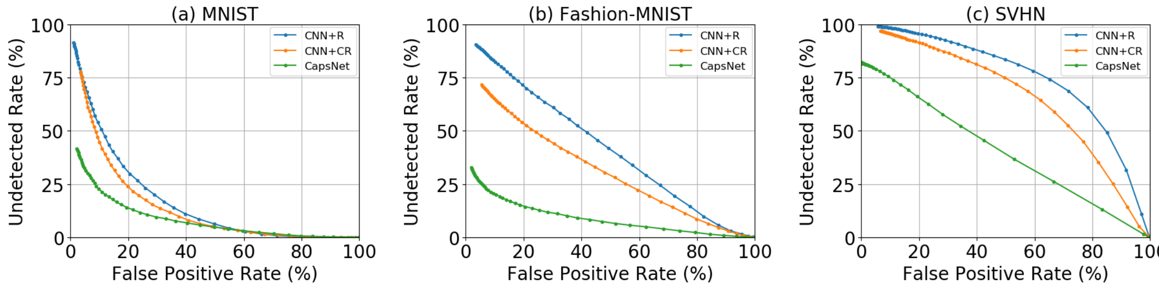


Figure 2.6. The undetected rate of the white-box targeted defense-aware R-PGD attack versus the False Positive Rate on the MNIST, Fashion-MNIST and SVHN datasets.

Table 2.5. Success rate and undetected rate of **black-box** targeted and untargeted attacks on the MNIST dataset. In the table, S_t/R_t is shown for targeted attacks and S_u/R_u is presented for untargeted attacks. All the numbers are shown in %.

Targeted	CapsNet	CNN-CR	CNN-R	Untargeted	CapsNet	CNN-CR	CNN-R
PGD	1.5/0.0	7.8/0.0	7.4/0.0	PGD	8.5/0.0	32.6/0.0	27.6/0.0
R-PGD	4.2/1.0	18.3/11.0	11.3/4.8	R-PGD	10.4/2.4	42.7/24.9	25.2/8.9

method is able to detect black box attacks as well. In fact, on the MNIST dataset the capsule model is able to detect all targeted and untargeted PGD attacks. Both the CNN-R and the CNN-CR models are able to detect the black box attacks as well, but with a relatively higher undetected rate.

2.5.8 CIFAR-10 Dataset

In order to show that our method based on CapsNet is capable to scale up to more complex datasets, we test our detection method with a deeper reconstruction network on CIFAR-10 [32]. The classification accuracy on the clean test dataset is 92.2%. In addition, we display the undetected rate of the white-box/black-box defense-aware R-PGD attack against CapsNets versus the False Positive Rate in Figure 2.7 (Left), where we can see a significant drop of the undetected rate of black-box R-PGD compared to the white-box setting. This indicates the CapsNets greatly reduce the attack transferability and the threat of black-box attacks.

Class-conditional Information

To investigate the effectiveness of the class-conditional information in the reconstruction network, we compare our CapsNet based on [54] with the other two variants of CapsNets: "CapsNet All" and "DeepCaps" [51]. In "CapsNet All", we remove the masking mechanism in the CapsNet and use all the capsules to do the reconstruction. In "DeepCaps", we extract the winning-capsule information as a single vector and used it as the input for the reconstruction network instead of using a masking mechanism to mask out the losing capsules information. In this way, the class information is not explicitly fed into the reconstruction network. As shown in Figure 2.7

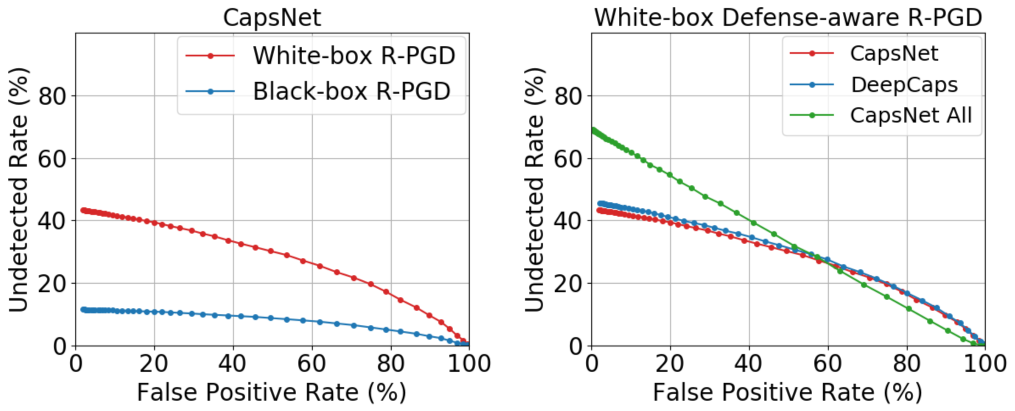


Figure 2.7. Left: The undetected rate of white-box/black-box defense-aware R-PGD versus the Fasle Positive Rate for the clean examples. The test model is our CapsNet. **Right**: The undetected rate of white-box defense-aware R-PGD versus the Fasle Positive Rate for the clean examples. The test model is our CapsNet using class-conditional reconstruction, "CapsNet All" using all capsule information, and the DeepCaps [51] using class-independent capsule information. The defense-aware R-PGD attack is tested on the CIFAR-10 dataset with $\epsilon_\infty = 8/255$.

(right), our CapsNet has the best detection performance (the lowest undetected rate at the same False Positive Rate) compared to the other two Capsule models. "DeepCaps" performs slightly worse that our "CapsNet" and "CapsNet All" has the worst detection performance. Therefore, we conclude that the class-conditional information used in the reconstruction network increases the model's robustness to adversarial attack. This also holds true to CNN-based networks because CNN+CR has a better detection performance than CNN+R, shown in Figure 2.6.

2.6 Discussion

Our detection mechanism relies on a similarity metric (i.e. a measure of reconstruction error) between the reconstruction and the input. This metric is required both during training in order to train the reconstruction network and during test time in order to flag adversarial examples. In the four datasets we have evaluated, the distance between examples roughly correlates with semantic similarity. This however might not the case for images in more complex dataset such as SUN dataset [69] and ImageNet [15], in which two images may be similar in terms of semantic content but nevertheless have significant ℓ_2 distance. A better similarity metric [67, 74] can be further explored to extend our methods to more complex problems. Furthermore our reconstruction network is trained on a hidden representation of one class but is trained to reconstruct the entire input. In datasets without distractors or backgrounds, this is not a problem. But in the case of ImageNet, in which the object responsible for the classification is not the only object in the image, attempting to reconstruct the entire input from a class encoding seems misguided.

2.7 Conclusion

We have presented a class-conditional reconstruction-based detection method that does not rely on a specific predefined adversarial attack. We have shown that by reconstructing the input from the internal class-conditional representation, our system is able to accurately detect black-box and white-box FGSM, BIM, PGD, and CW attacks. We then proposed a new attack to beat our defense - the Reconstructive Attack - in which the adversary optimizes not only the classification loss but also minimizes the reconstruction loss. We showed that this attack was able to fool our detection mechanism but with a much smaller success rate than a standard attack. Compared to CNN-based models, we showed that the CapsNet was able to detect adversarial examples with greater accuracy on all the datasets we explored.

This chapter is based on the material as it appears in the Proceedings of the International Conference on Learning Representations (ICLR20) (Yao Qin, Nicholas Frosst, Sara Sabour, Colin Raffel, Garrison Cottrell and Geoffrey Hinton, “Detecting and Diagnosing Adversarial Images with Class-Conditional Capsule Reconstructions”). The dissertation author was the co-primary investigator and author of this paper.

Chapter 3

Diagnosing Adversarial Attacks

3.1 Introduction

Adversarial examples raise questions about whether neural network models are sensitive to the same visual features as humans. In the previous Chapter, we find that CapsNets always perform better than convolutional networks among all the attacks. To explain the success of CapsNets over CNNs, we further ***diagnose*** the adversarial examples for CapsNets and find that 1) the success of the targeted reconstructive attack is highly dependent on the visual similarity between the source image and the target class. 2) many of the resultant attacks resemble members of the target class and so cease to be "adversarial" – i.e., they may also be misclassified by humans. These findings suggest that CapsNets with class conditional reconstructions have the potential to address the real issue with adversarial examples – networks should make predictions based on the same properties of the image as people use rather than using features that can be manipulated by an imperceptible adversarial attack. Based on extensive qualitative studies, we conclude that CapsNets is better at detecting adversarial examples compared to CNNs, suggesting the features captured by CapsNets are more aligned with human perception.

3.2 Visual Coherence of the Reconstructive Attack

The great success of CapsNet over CNN-based models motivates us to further diagnose the generated adversarial examples for CapsNets. If our true aim in adversarial robustness

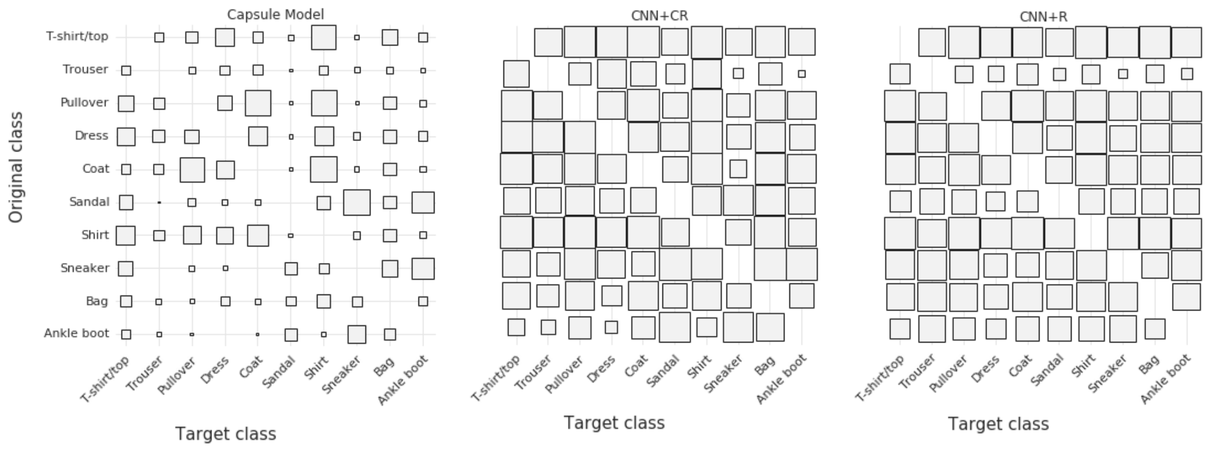


Figure 3.1. This diagram visualizes the adversarial success rates for each source/target pair for targeted R-PGD attacks on Fashion-MNIST with $\varepsilon_\infty = 25/255$. The size of the box at position x, y represents the success rate of adversarially perturbing inputs of class x to be classified as class y. Almost all source/target We can see that there is significantly higher variance for the CapsNet model than for the two CNN models.

research is to create models that make predictions based on reasonable and human-observable features, then we would prefer models that are more likely to misclassify a "shirt" as a "t-shirt" (in the case of FashionMNIST) than to misclassify a "bag" as a "sweater". For a model to behave ideally, the success of an adversarial perturbation would be related to the visual similarity between the source and the target class. By visualizing a matrix of adversarial success rates between each pair of classes (shown in Figure 3.1), we can see that for the capsule model there is a great variance between the source and target class pairs and that the success rate of attacks is highly related to the visual similarity of the classes. However, this is not the case for either of the other two CNN-based models.

Thus far we have treated all attacks as equal. However, a key component of an adversarial example is that it is visually similar to the source image, and that it does not resemble the adversarial target class. The adversarial research community makes use of a small epsilon bound as a mechanism for ensuring that the resultant adversarial attacks are visually unchanged from the source image. For standard attacks this heuristic is sufficient, because taking gradient steps in the image space in order to have a network misclassify an image normally results in

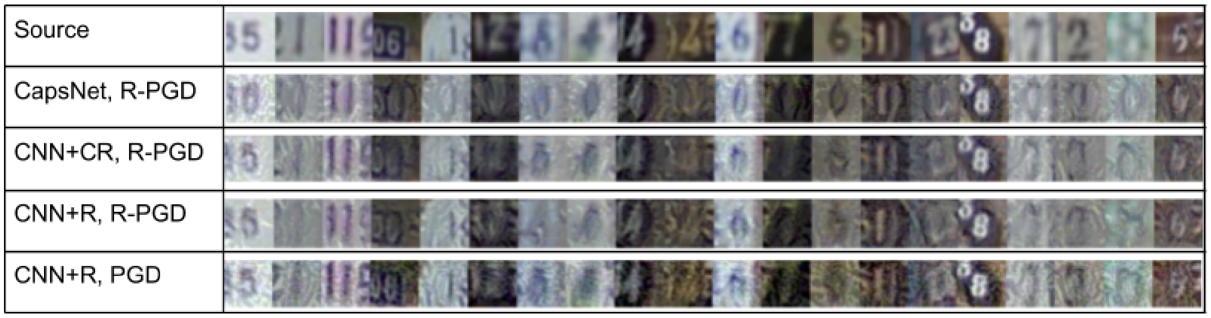


Figure 3.2. These are randomly sampled (not cherry picked) successful and undetected adversarial attacks created by R-PGD with a target class of 0 for each model on the SVHN dataset($\varepsilon_\infty = 25/255$). We can see that for the capsule model, many of the attacks are not "adversarial" as they resemble members of the target class.

something visually similar to the source image. But this is not the case for adversarial attacks which take the reconstruction error into account: As shown in Figure 3.2, when we use R-PGD to attack the CapsNet, many of the resultant attacks resemble members of the target class. In this way, they stop being "adversarial". As such, an attack detection method which does not detect them as adversarial is arguably behaving correctly. This puts the previously undetected rates presented earlier in a new light, and illustrates a difficulty in the evaluation of adversarial attacks and defenses. In addition, it should be noted that this phenomenon rarely occurs in a standard convolutional neural network, which suggests that the features captured by CapsNet are more aligned with human perception.

In Figure 3.3, we display sample images and the result of adversarially perturbing them with targeted R-PGD against the CapsNet model. We can see the visual similarity between many of the attacks and the target class.

3.3 Conclusion

In this chapter, we further explain the success of CapsNet by qualitatively showing that the success of the reconstructive attack was proportional to the visual similarity between the target class and the source class for the CapsNet. In addition, we showed that images generated by this reconstructive attack to attack the CapsNet are not typically adversarial, i.e. many of

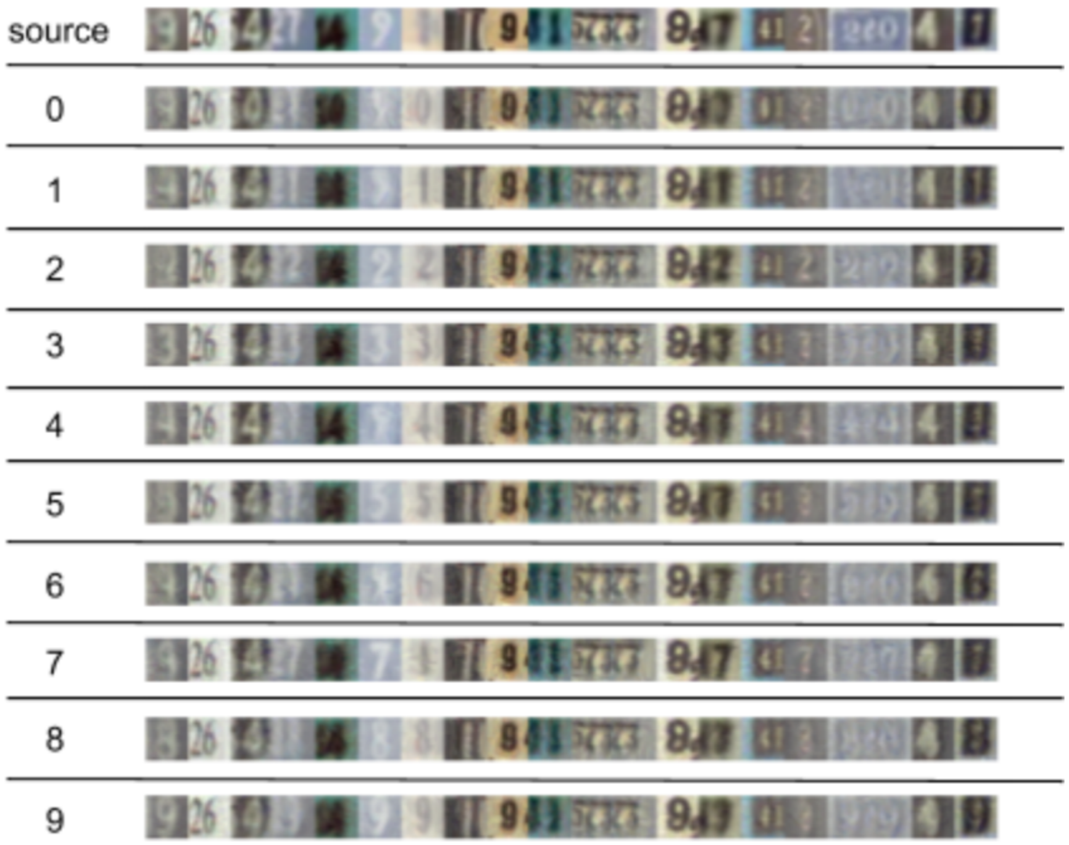


Figure 3.3. These are randomly sampled (not cherry picked) inputs (top row) and the result of adversarially perturbing them with targeted R-PGD against the CapsNet model (other rows). Many of these attacks are not successful.

the resultant attacks resemble members of the target class even with a small ℓ_∞ norm bound. These are not the case for the CNN-based models. The extensive qualitative studies indicate that the gradient of the reconstructive attack may be better aligned with the true data manifold, and implies that the capsule model relies on visual features similar to those used by humans. We believe this is a step towards solving the true problem posed by adversarial examples.

This chapter is based on the material as it appears in the Proceedings of the International Conference on Learning Representations (ICLR20) (Yao Qin, Nicholas Frosst, Sara Sabour, Colin Raffel, Garrison Cottrell and Geoffrey Hinton, "Detecting and Diagnosing Adversarial Images with Class-Conditional Capsule Reconstructions"). The dissertation author was the co-primary investigator and author of this paper.

Chapter 4

Deflecting Adversarial Attacks

4.1 Introduction

Adversarial attacks have been the subject of constant research since they were first discovered [64, 21, 34, 41]. Most of this research has been focused on the creation of more robust models to **defend** against adversarial attacks [61, 41, 71], where the input image is correctly classified as the original class rather than the adversarial target class, as illustrated in Figure 4.1 (a). However, better defenses have led to the development of stronger attack algorithms to break these defenses [41, 10, 13, 2]. After several iterations of creating and breaking defenses, some research focused on adversarial attack **detection** [22, 16, 43, 37, 50, 52]. Detection algorithms aim to distinguish adversarial attacks from real data and then flag the adversarial input, instead of attempting to correctly classify such inputs, as shown in Figure 4.1 (b). However, this strategy fell into the same creating/breaking cycle: Many state-of-the-art methods [52, 40, 37] claiming to detect adversarial attacks were broken shortly after publication with a defense-aware attack [25, 9, 2]. We attempt to get ahead of this cycle by focusing on the **deflection** of adversarial attacks, shown in Figure 4.1 (c): If the result of the adversarial optimization of an image looks to a human like the adversarial target class rather than its original class, then the image can hardly be called adversarial anymore. We call such attacks “deflected”. Some examples are shown in Figure 4.2. Although visually there is a big difference between clean input and deflected attacks, we have confirmed the maximal adversarial perturbation is bounded by 16/255 via reading the

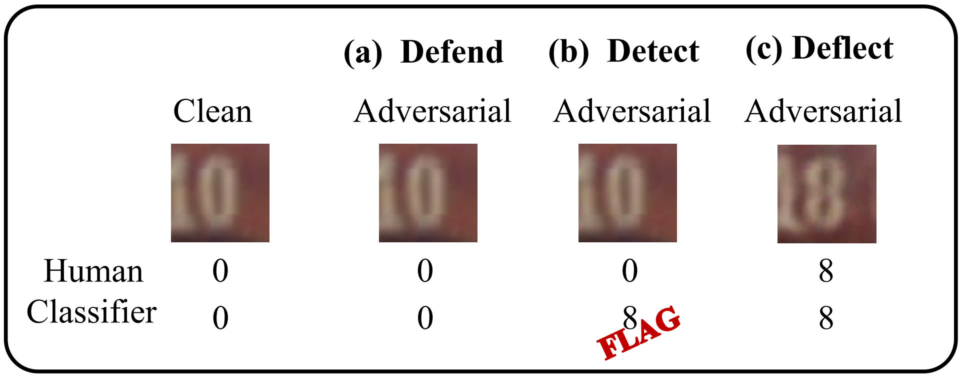


Figure 4.1. Different results of an adversarial attack against three different defense approaches. The original class is 0 and the adversarial target class is 8.

clean image and its corresponding deflected attack and computing the difference of their pixel value.

In this chapter, we propose a network and detection mechanism based on Capsule layers [54, 50] that either detects attacks accurately or, for undetected attacks, often causes the attacker to produce images that resemble the target class (thereby deflecting them). Our network architecture is made up of two components: A capsule classification network that classifies the input, and a reconstruction network that reconstructs the input conditioned on the pose parameters of the predicted capsule. Apart from the classification loss and ℓ_2 reconstruction loss used in [54, 50], we introduce an extra cycle-consistency training loss which constrains the classification of the winning capsule reconstruction to be the same as the classification of the original input. This new auxiliary training loss encourages the reconstructions to more closely match the class-conditional distribution and helps the model detect and deflect adversarial attacks.

In addition, we propose two new attack-agnostic detection methods based on the discrepancy between the winning-capsule reconstruction of clean and adversarial inputs. We find that a detection method that combines ours with the one proposed by [50] performs best. We show that this method can accurately detect white-box and black-box attacks based on three different distortion metrics (EAD [13], CW [10] and PGD [41]) on both the SVHN and CIFAR-10 datasets.

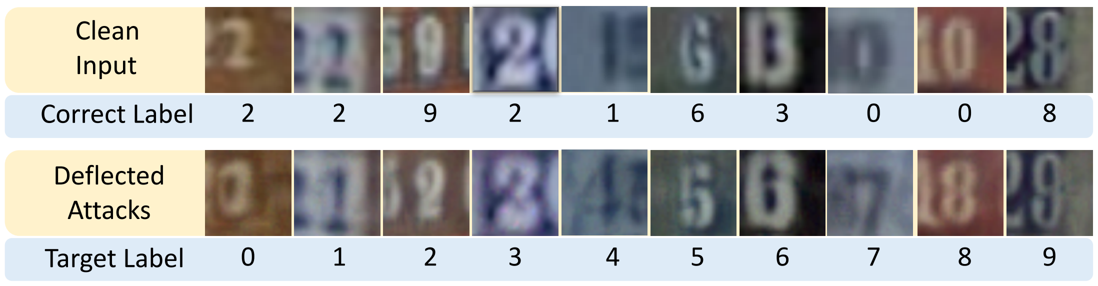


Figure 4.2. Deflected adversarial attacks on the SVHN dataset. These images were generated by a defense aware attack and the maximal adversarial perturbation is bounded by 16/255.

Following the suggestions in [2, 9], we also propose defense-aware attacks for our new detection method. We find that our detection methods significantly outperform state-of-the-art methods on defense-aware attacks. Finally, we perform a human study to verify that many of the undetected adversarial attacks against our model have been successfully deflected, i.e. adversarial images from both defense-aware and standard attacks against our detection mechanism are frequently classified as the target class by humans. In contrast, successful attacks against baseline models do not have this property.

To summarize, in this chapter:

- We introduce the notion of **deflecting adversarial attacks**, which presents a step towards ending the battle between attacks and defenses.
- We propose a new cycle-consistency loss which trains a CapsNet to encourage the winning-capsule reconstruction to closely match the class-conditional distribution and show that this can help detect and deflect adversarial attacks.
- We introduce two attack-agnostic detection methods based on the discrepancy between the winning-capsule reconstruction of the clean and adversarial inputs, and design a defense-aware attack to specifically attack our detection mechanisms.
- We show through extensive experiments on SVHN and CIFAR-10 that our detection mechanism can achieve state-of-the-art performance in detecting white-/black-box standard

and defense-aware attacks.

- We perform a human study to show that our approach, unlike previous methods, is able to deflect a large percentage of undetected adversarial attacks.

4.2 Network Architecture

In order to design a model that is strong enough to deflect adversarial attacks, we build our network based on CapsNet [54]. Figure 4.3 shows the pipeline of our network architecture.

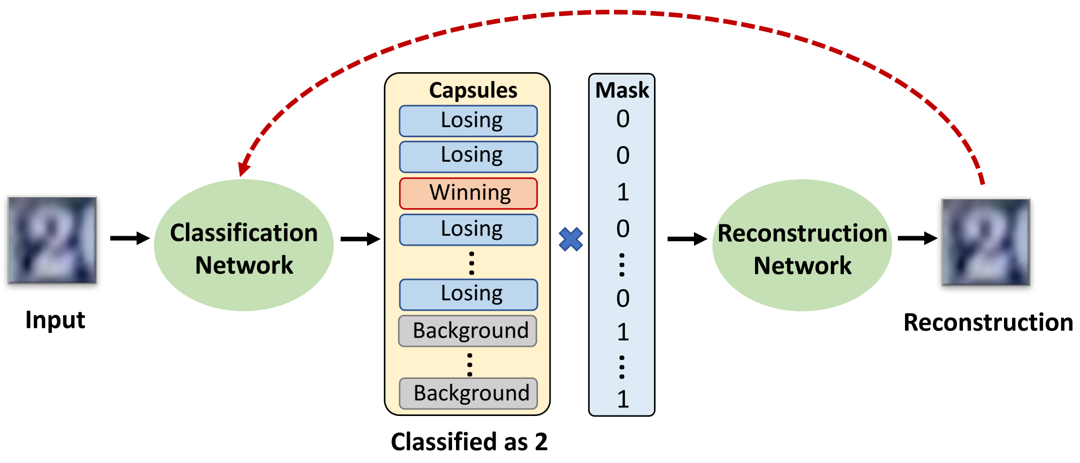


Figure 4.3. The network architecture with cycle-consistent winning capsule reconstructions.

The final layer of our classifier is a Capsule layer ("CapsLayer" for short) which includes both class capsules and background capsules. These capsules are intended to encode feature attributes corresponding to the class and the background respectively. Given an input x, the output of a CapsLayer is a prediction $f(x)$ and a pose parameter v for all the classes and the background, where v_i denotes the pose parameter for class i. As in the initial Capsules proposed in [54], the magnitude of the activation vector of a capsule encodes the existence of an instance of the class and the orientation of the activation vector encodes instantiation parameters of the instance, such as its pose. Therefore, the magnitudes of the capsules' activations are used to perform classification while the activation vector of the winning class capsule together with the activation vectors of the background capsules are used as the input to the reconstruction network. We use $r(v_{i=f(x)})$ and $r(v_{i\neq f(x)})$ to represent the reconstruction from the winning capsule and a losing capsule respectively. The reconstruction network uses the activations of all the background capsules as well as the activation of one class capsule but we omit this to simplify the notation. The details of the network architecture used in this chapter are provided in Table 4.1 and Table 4.2.

Table 4.1. The network architecture for the SVHN dataset.

	Layer Name	Configurations
Classification Network	Conv	filter size: 3x3, number of filters: 64x4, stride size: 1x1, activation: leaky relu
	Conv	filter size: 3x3, number of filters: 64x8, stride size: 1x1, activations: leaky relu
	Avg Pooling	pool size: 2x2, stride size: 2x2
	Conv	filter size: 3x3, number of filters: 64x2, stride size: 1x1, activation: leaky relu
	Conv	filter size: 3x3, number of filters: 64x4, stride size: 1x1, activation: leaky relu
	Avg Pooling	pool size: 2x2, stride size: 2x2
	Conv	filter size: 3x3, number of filters: 64x1, stride size: 1x1, activation: leaky relu
	Conv	filter size: 3x3, number of filters: 64x2, stride size: 1x1, activation: leaky relu
	CapsLayer	number of input capsules: 16, input atoms: 512, number of output capsules: 25, output atoms: 4, number of dynamic routing: 1
Reconstruction Network	fully connected	input size: 100, output size:1024
	fully connected	input size: 1024, output size:16384
	deconv	filter size: 4x4, number of filters: 64, stride size: 2x2
	deconv	filter size: 4x4, number of filters: 32, stride size: 2x2
	conv	filter size: 4x4 number of filters: 3, stride size: 1x1, activation: sigmoid

Table 4.2. The network architecture for the CIFAR-10 dataset.

	Layer Name	Configurations
Classification Network	Conv	filter size: 3x3, number of filters: 128x4, stride size: 1x1, activation: leaky relu
	Conv	filter size: 3x3, number of filters: 128x8, stride size: 1x1, activations: leaky relu
	Avg Pooling	pool size: 2x2, stride size: 2x2
	Conv	filter size: 3x3, number of filters: 128x2, stride size: 1x1, activation: leaky relu
	Conv	filter size: 3x3, number of filters: 128x4, stride size: 1x1, activation: leaky relu
	Avg Pooling	pool size: 2x2, stride size: 2x2
	Conv	filter size: 3x3, number of filters: 128x1, stride size: 1x1, activation: leaky relu
	Conv	filter size: 3x3, number of filters: 128x2, stride size: 1x1, activation: leaky relu
	CapsLayer	number of input capsules: 16, input atoms: 512, number of output capsules: 25, output atoms: 8, number of dynamic routing: 1
Reconstruction Network	fully connected	input size: 200, output size:1024
	fully connected	input size: 1024, output size:16384
	deconv	filter size: 4x4, number of filters: 64, stride size: 2x2
	deconv	filter size: 4x4, number of filters: 32, stride size: 2x2
	conv	filter size: 4x4 number of filters: 3, stride size: 1x1, activation: sigmoid

4.2.1 Cycle-consistent winning-capsule reconstructions

The CapsNet [54] is trained with two loss terms: a marginal loss for the classification and an ℓ_2 reconstruction loss. To encourage the reconstruction to more closely match the class conditional distribution and help the model detect and deflect adversarial attacks, we additionally incorporate an extra cycle-consistency loss ℓ_{cyc} which constrains the reconstruction from the winning capsule to be classified as the same class as the input, formulated as:

$$\ell_{cyc} = \ell_{net}(f(r(v_{i=f(x)})), f(x)), \tag{4.1}$$

where ℓ_{net} is the cross-entropy loss function and $i \in \{0, 1, \ldots, n\}$, n denotes the number of classes in the dataset. This can be achieved by feeding the reconstruction corresponding to the winning capsule back into the classification network, shown as the dotted red line in Figure 4.3. This extra training loss together with our Cycle-consistent Detector (introduced in Section 4.3.3) can help detect adversarial attacks. In addition, since the winning-capsule reconstructions are optimized to more closely match the class conditional data distribution, it becomes easier for our model to deflect adversarial attacks.

4.3 Detection Methods

In this chapter, we use three reconstruction-based detection methods to detect standard attacks. They are: **G**lobal **T**hreshold **D**etector (GTD), first proposed in [50], **L**ocal **B**est **D**etector (LBD) and **C**ycle-**C**onsistency **D**etector (CCD).

4.3.1 Global Threshold Detector

When the input is adversarially perturbed, the classification given to the input may be incorrect, but the reconstruction is often blurry and therefore the distance between the adversarial input and the reconstruction is larger than that would be expected from normal input. This allows us to detect the input as adversarial with the Global Threshold Detector. This method, proposed in [50], measures the reconstruction error between the input and its reconstruction from the winning capsule. If the reconstruction error is greater than a global threshold θ, that is:

$$\|r(v_{i=f(x)}) - x\|_2 > \theta, \tag{4.2}$$

then the input is flagged as an adversarial example.

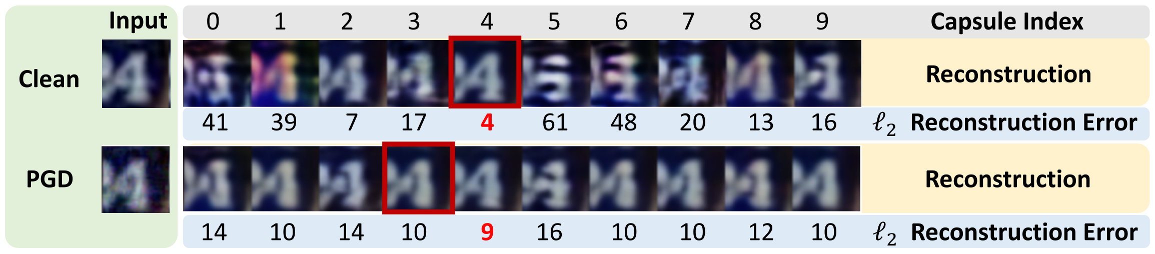


Figure 4.4. An example of a clean input, an adversarial example generated via a PGD attack, and the reconstructions for the clean and adversarial inputs from each class capsule. The reconstruction corresponding to the winning capsule is surrounded by a red box. Under each reconstruction is its ℓ_2 reconstruction error; the smallest reconstruction error is highlighted in red. Both the clean input and its winning capsule reconstruction are classified as '4'. The PGD attack is classified as the target class '3' but its winning capsule reconstruction is classified as '4'.

4.3.2 Local Best Detector

When the input is a clean image, the reconstruction error from the winning capsule is smaller than that of the losing capsules, where an example is shown in the first row of Figure 4.4. This is likely because the ℓ_2 reconstruction objective only minimizes the reconstruction from the winning capsule during training. However, when the input is an adversarial example, the reconstruction from the capsule corresponding to the correct label can be even closer to the input compared to the reconstruction corresponding to the winning capsule (see the second row in Figure 4.4). Therefore, we propose the "Local Best Detector" (LBD) to detect such adversarial images whose reconstruction error from the winning capsule is not the smallest, that is:

$$\text{argmin}_j \|r(v_j) - x\|_2 \neq f(x), \quad j \in \{0, 1, \ldots, n\}, \tag{4.3}$$

where n is the number of classes in the dataset.

4.3.3 Cycle-Consistency Detector

If the input is a clean image, the reconstruction from the winning capsule will resemble the input. Our model should ideally assign the same class to the reconstruction of the winning capsule as the clean input. This behavior is reinforced by training with the cycle-consistency loss.

For example, as shown in Figure 4.4 both the clean input and its winning-capsule reconstruction are classified as 4. However, when the input is an adversarial example which is perceptually indistinguishable from the clean image but causes the model to predict the target class, the reconstruction of the winning capsule often appears closer to the clean input and/or is blurry. As a result, the reconstruction of the winning capsule is often not classified as the target class. As shown in Figure 4.4, the adversarial input has been classified as the target class “3” while the reconstruction corresponding to the winning capsule is classified as “4”. Therefore, the Cycle-Consistency Detector (CCD) is designed to flag the input as an adversarial example if the input x and its reconstruction of the winning capsule $r(v_{i=f(x)})$ are not classified as the same class:

$$f(r(v_{i=f(x)})) \neq f(x). \tag{4.4}$$

In this chapter, we use these three detectors together to detect adversarial examples. In other words, we flag any input as adversarial if it is classified as adversarial by any of the detection mechanisms. As a result, an adversarial input can only go undetected if it passes all three detection mechanisms.

4.4 The Defense-Aware CC-PGD Attack

In order for an attack mechanism to generate an adversarial example $x' = x + \delta$ (where δ is a small adversarial perturbation) that can both cause a misclassification and is not detected by our detection mechanisms, the constructed adversarial attack must:

- successfully fool the classifier: $f(x') = t$ and $f(x) \neq t$, where t is the target class.
- avoid being detected by the Global Threshold Detector (GTD), the attack needs to constrain the reconstruction of the winning capsule to be close to the input.
- fool the Local Best Detector (LBD), the attack should encourage the reconstructions from all the losing capsules to be far away from the input to ensure the reconstruction error of

the winning capsule is the smallest.

- circumvent the Cycle-Consistency Detector (CCD) by fooling the classifier into making the target prediction when it is fed the winning-capsule reconstruction of the adversarial input, that is: $f(r(v_{i=f(x')})) = f(x') = t$.

To generate such an attack, we follow [50] and devise attacks which consist of two stages at each gradient step. The first stage attempts to fool the classifier by following a standard attack (e.g., a standard PGD attack) which follows the gradient of the cross-entropy loss function with respect to the input. Then, in the second stage, we focus on fooling the detection mechanisms by taking the reconstruction error and cycle-consistency into consideration. This can be formulated as minimizing the reconstruction loss ℓ_r, which consists of three components: the reconstruction loss corresponding to the Global Threshold Detector ℓ_g, the reconstruction loss corresponding to the Local Best Detector ℓ_l and the cycle-consistency classification loss corresponding to the Cycle-Consistency Detector ℓ_{cyc}. Specifically, the reconstruction loss is defined as:

$$\begin{aligned} \ell_r(x') &= \alpha_1 \cdot \ell_g(x') + \alpha_2 \cdot \ell_l(x') + \alpha_3 \cdot \ell_{cyc}(x') \\ &= \alpha_1 \cdot \|r(v_{i=f(x')}) - x'\|_2 - \alpha_2 \cdot \frac{\sum_{k \neq f(x')}^{n} \|r(v_k) - x'\|_2}{n-1} \\ &\quad + \alpha_3 \cdot \ell_{net}(f(r(v_{i=f(x')})), f(x')) \end{aligned} \tag{4.5}$$

where $x' = x + \delta$ is the adversarial example, n is the number of the classes in the dataset, $\|r(v_{i=f(x')}) - x'\|_2$ is the winning-capsule reconstruction error and $\|r(v_{k\neq f(x')}) - x'\|_2$ is the losing-capsule reconstruction error. The hyperparameters α_1, α_2 and α_3 are used to balance the importance of attacking each detector. Then, the adversarial perturbation can be updated in the second stage as:

$$\delta \leftarrow \text{clip}_{\varepsilon_\infty}(\delta - c \cdot \text{sign}(\nabla_\delta(\ell_r(x+\delta)))), \tag{4.6}$$

where ε_∞ is the ℓ_∞ norm bound and c is the step size in each iteration.

4.5 Experiments

Now that we have proposed our new defense model, we first verify its detection performance on the SVHN and CIFAR-10 datasets on a variety of attacks. Then, we use a human study to demonstrate that our model frequently causes attacks to be deflected.

4.5.1 Evaluation Metrics and Datasets

In this chapter, we use **Accuracy** to represent the proportion of clean examples that are correctly classified by our network. We use **Success Rate** to measure the performance of an attack, which is defined as the proportion of adversarial examples that successfully fool the classifier into making the targeted prediction. In order to evaluate the performance of different detection mechanisms, we report both **False Positive Rate (FPR)** and **Undetected Rate**. The False Positive Rate is the proportion of clean examples that are flagged as an adversarial example by the detection mechanism. The Undetected Rate, first proposed in [50], denotes the proportion of adversarial examples that successfully fool the classifier and also go undetected. Finally, we perform a human study in Section 4.6 in order to show that our model is able to effectively deflect adversarial attacks.

4.5.2 Training Details and Test Accuracy

We set the batch size to be 64 and the learning rate to 0.0001 to train the network on SVHN. For CIFAR-10, the batch size is set to be 128 and the learning rate is 0.0002. We use the Adam optimizer [31] to train all models. The cycle-consistency loss ℓ_{cyc} is multiplied with 0.0005 before being added to the margin loss and the ℓ_2 reconstruction loss used as in the original CapsNets [54].

We test our deflecting models on the SVHN [47] and CIFAR-10 datasets [32]. The classification accuracy on the clean test set is 96.5% on SVHN and 92.6% on CIFAR-10, which show that our deflecting models are reasonably good at classifying clean images.

4.5.3 Threat Model

In this chapter, we consider two commonly used threat models: white-box and black-box. For white-box attacks, the adversary has full knowledge of the network architecture and parameters and is allowed to construct the adversarial attack by computing the gradient of model's output with respect to its input. In the black-box setting, the adversary is aware of the network architecture of the target model but does not have direct access to the model's parameters.

4.5.4 Adversarial Attacks

Following the suggestions in [7], we test our attack-agnostic detection mechanisms on three standard targeted attacks based on different distance metrics:

- ℓ_1 norm-based EAD [13],
- ℓ_2 norm-based CW [10],
- ℓ_∞ norm-based PGD [41].

In addition, we follow the suggestions in [9] to report the performance of our detection mechanisms against defense-aware attacks. We use CC-PGD (described in Section 4.4) as our defense-aware attack. For the ℓ_∞ norm-based attacks, we set the maximal perturbation ϵ_∞ to be 16/255 on SVHN and 8/255 on CIFAR-10 as is typically used [6, 41].

To generate EAD and CW attacks, we follow the previous work [13, 10] to set the binary search steps to be 9, maximum iterations to be 1000 and learning rate to be 0.01. To construct ℓ_∞ norm-based attacks (PGD and our defense-aware CC-PGD), we use a step size 0.01 (2.55/255) in each iteration as [41].

4.5.5 Sanity Checks for PGD and CC-PGD Attack

In this section, we perform basic sanity checks to ensure the adversarial attacks are correctly implemented and our proposed defense-aware CC-PGD is tuned well. In this section, we test attacks against our proposed deflecting model on the CIFAR-10 dataset.

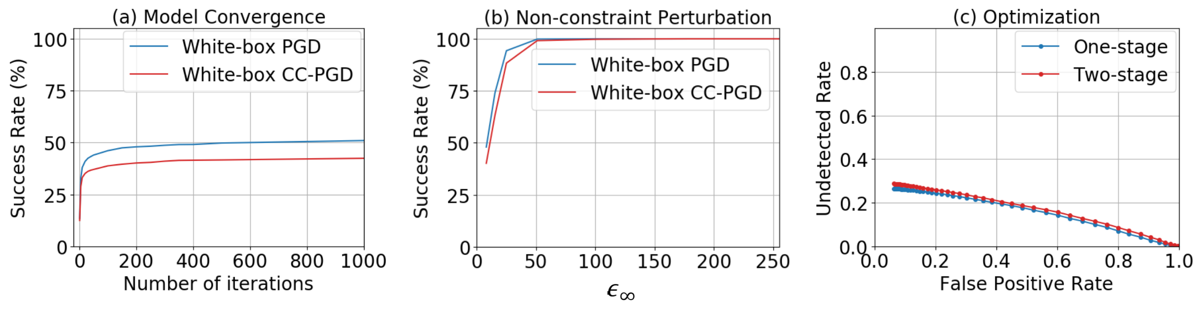


Figure 4.5. (a) The success rate of white-box PGD and CC-PGD changes as the number of iterations increases for our deflecting model on CIFAR-10 dataset. (b) The success rate of white-box PGD and CC-PGD changes as ε_∞ increases for our deflecting model on CIFAR-10 dataset. (c) The Undetected Rate of the defense-aware attack CC-PGD optimized by a two-stage optimization and one-stage optimization vs. False Positive Rate for the clean data on the CIFAR-10 dataset.

- **Convergence of attacks.**

  Figure 4.5 (a) shows the success rate of white-box PGD and CC-PGD varies as the number of iterations increases on the CIFAR-10 dataset. We can see that the attacker has almost plateaued after 200 iterations. Therefore, we set the total number of attack steps to be 200 in generating PGD and CC-PGD attack for efficiency.

- **100% success rate with non-constraint ℓ_∞ norm.**

  In Figure 4.5 (b), we show that the success rate of white-box PGD and CC-PGD varies as the ℓ_∞ bound of the adversarial perturbation ε_∞ increases. We can see that when ε_∞ is greater than 50/255, the success rate is 100%. However, when ε_∞ is set to be 8/255 (which is typically used [6, 41]), the attack success rate against our deflecting model is below 50%.

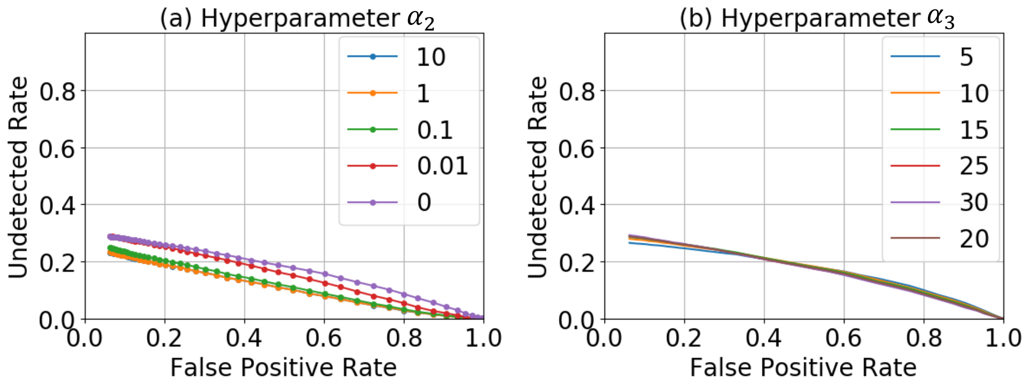


Figure 4.6. The undetected rate of our white-box defense-aware CC-PGD attack versus False Positive Rate (FPR) for clean input on the CIFAR-10 dataset when we change the hyperparameter α_2 in (a) and hyperparameter α_3 in (b). These hyperparameters control the importance of attacking each detector in Eqn. 5.13.

- **Two-stage optimization**

  To demonstrate the effectiveness of our used two-stage optimization in generating defense-aware CC-PGD, we compare the attack performance of two-stage optimization introduced in Section 4.4 and a one-stage optimization which uses a single loss function which combines the cross-entropy loss to fool the classifier with the reconstruction loss ℓ_r in Eqn. 5.13 to fool the detectors. In Figure 4.5 (c), we construct the defense-aware CC-PGD against our deflecting model on the CIFAR-10 dataset using one-stage and two-stage optimization respectively. We can see that the defense-aware CC-PGD attack that is optimized by the two-stage optimization has a higher undetected rate than that optimized by the one-stage optimization. Therefore, we use two-stage optimization in all the experiments to construct CC-PGD attack.

- **Hyperparameters**

  In Eqn. 5.13, the hyperparameters α_1, α_2 and α_3 are used to balance the importance of attacking each detector. We set $\alpha_1 = 1$ and then show the attack performance when we change α_2 (see Figure 4.6 (a)) and α_3 (see Figure 4.6 (b)). We can see that when we set $\alpha_2 = 0$, the attack performance is the best (higher undetected rate at a low False

Positive Rate). In addition, the attack performance of our CC-PGD is not sensitive to the hyperparameter α_3. Therefore, we simply set $\alpha_3 = 20$, which is slightly better at a low False Positive Rate. In all the following experiments, we set the hyperparameter α_1, α_2 and α_3 in Eqn. 5.13 to be 1, 0 and 20 respectively to balance the importance among three detectors in generating our defense-aware CC-PGD. Since the Cycle-Consistency Detector is the most effective detector (discussed later in Section 4.5.6), we assign a much higher weight to α_3, which controls the importance of attacking Cycle-Consistency Detection in generating our defense-aware CC-PGD attack. In addition, we observe that increasing α_2 (controlling the importance of attacking the Local Best Detector) leads to a decrease of the attack performance). Therefore, α_2 is set to be 0. This might result from the contradiction between minimizing the winning-capsule reconstruction and maximizing the losing-capsule reconstruction, where they share the background capsule information. Lastly, α_1 is set to be a very small value as 1 for the best attack performance for CC-PGD.

The parameter that balances the importance of the two stages in CC-PGD is empirically set to be 0.5 on SVHN and 0.75 for the first stage and 0.25 for the second stage on CIFAR-10.

4.5.6 Ablation Study

- **Detection methods**

In this section, we study the effectiveness of our proposed detection mechanisms: Local Best Detector (LBD) and Cycle-Consistency Detector (CCD) and compare them with the Global Threshold Detector (GTD) from [50].

Since the False Positive Rate (FPR) of clean input flagged by the Global Threshold Detector (GTD) varies as the chosen global threshold, in Figure 4.7 we plot the undetected rate of white-box adversarial attacks flagged by different detectors versus the False Positive Rate (FPR) of the clean input. The global threshold θ is chosen from the range [0, 20] with a step size of 0.4. We can clearly see that: 1) A single Global Threshold Detector (GTD) proposed in [50] is not enough to effectively detect adversarial attacks. 2) In the

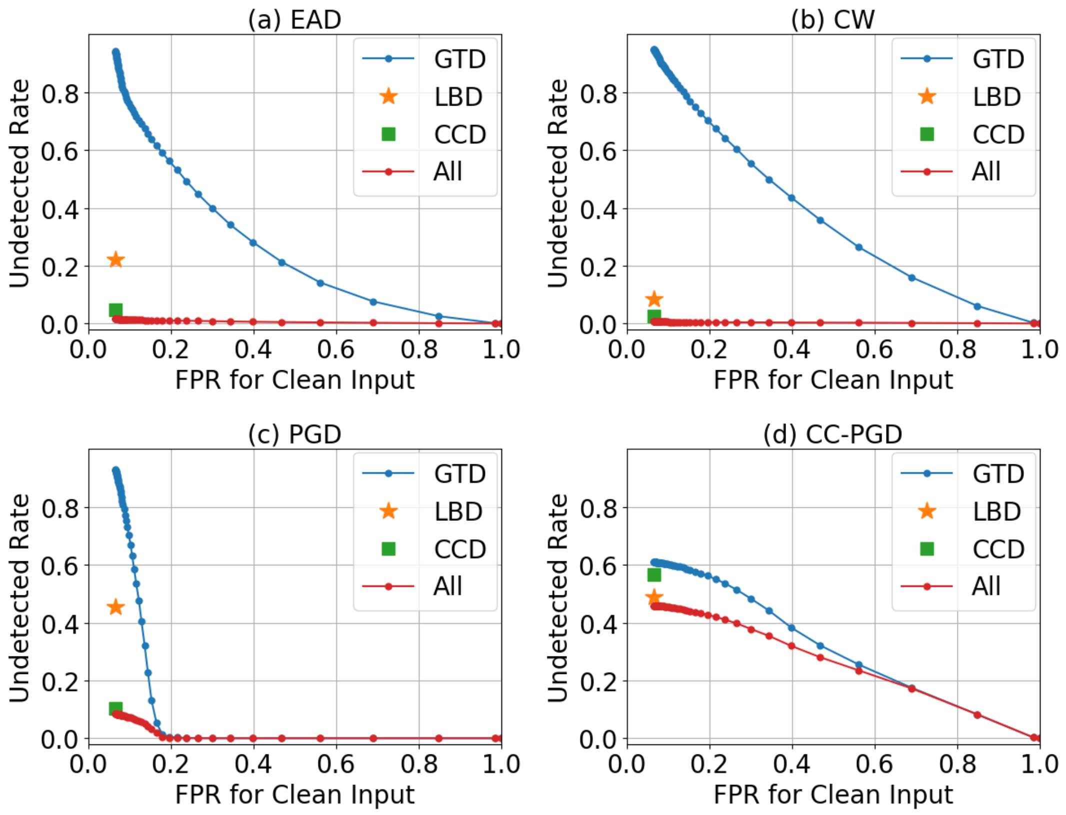


Figure 4.7. The Undetected Rate of different detectors for white-box attacks versus False Positive Rate (FPR) for clean input on the SVHN dataset. "All" denotes GTD, LBD and CCD are all used to detect adversarial attacks. The better detection mechanism has a smaller FPR for clean input and smaller undetected rate for attacks.

standard EAD, CW and PGD attack, the CCD is the most effective detector at a low False Positive Rate. However, it becomes less effective than LBD when the inputs are created with the defense-aware CC-PGD attack which is designed to specifically attack the three detection mechanisms. 3) In all the attacks, the combination of all three detectors always performs the best. Therefore, we only report the performance of the undetected rate of the combination of all three detectors in the following experiments.

- **Cycle-consistency loss**

To demonstrate the effectiveness of the proposed cycle-consistency loss, we construct a baseline Capsule model that has the same network architecture as our deflecting model but is trained without the extra cycle-consistency loss. The False Positive Rate of the Cycle-Consistency Detector on the CIFAR-10 test set is 33.46%, which represents that 33.46% of the clean test images are incorrectly flagged as an adversarial example by the Cycle-Consistency Detector. This means the Cycle-Consistency Detector is not suitable for a model that is trained without cycle-consistency loss. Therefore, to compare the detection performance between the baseline Capsule model and our deflecting model, we use a combined Global Threshold Detector (GTD) and Local Best Detector (LBD) for the baseline Capsule model and all three detectors for the deflecting model. The undetected rate of the white-box defense-aware attack versus the False Positive Rate (FPR) of the clean input on the CIFAR-10 dataset is shown in Figure 4.8, where we can see that our deflecting model together with all three detectors has a better detection performance compared to the baseline model trained without the cycle-consistency loss.

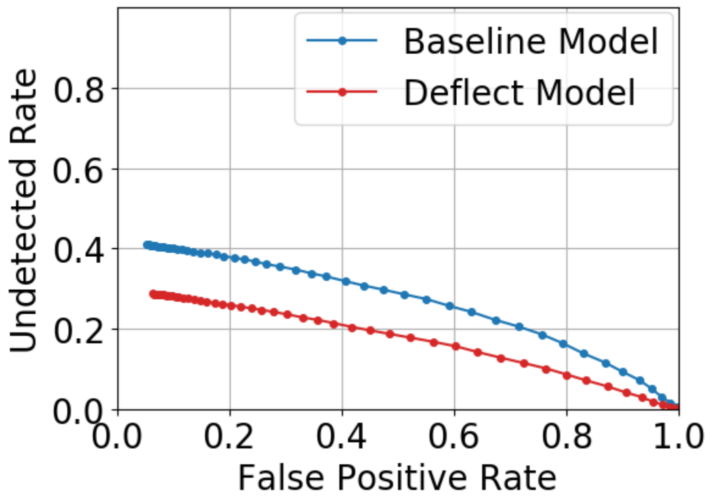


Figure 4.8. Ablation study for cycle-consistency loss. The Undetected Rate of the defense-aware attack vs. False Positive Rate for baseline Capsule model trained without cycle-consistency loss and our deflecting model on the CIFAR-10 dataset. GTD and LBD are used to detect adversarial examples in baseline Capsule model. GTD, LBD and CCD are all used to detect adversarial attacks for our deflecting model.

4.5.7 Detection of White-box Attacks

Before showing that our defense produces deflected attacks, we must first validate that it improves detection performance. Therefore, we test our model on standard and defense-aware attacks and compare it with state-of-the-art detection methods in this section.

- **Standard attacks**

  As shown in Figure 4.9, our detection method has a very small undetected rate for standard white-box attacks on both the SVHN and CIFAR-10 dataset. For PGD attacks, we achieve an undetected rate below 10% with a small False Positive Rate on the SVHN dataset. The undetected rate for white-box PGD is around 22% with the smallest False Positive Rate on the CIFAR-10 dataset. These demonstrate that our detection mechanism is very effective in detecting standard white-box attacks that are based on different ℓ_p norms.

- **Defense-aware attacks**

  Following the suggestions in [9], we test our detection mechanism in the setting where the adversary is fully aware of the defense ("defense-aware attacks") using the CC-PGD attack. Since the PGD attack is stronger than EAD and CW, the first stage of our CC-PGD attack is to construct an adversarial image via standard PGD and then, in the second stage, take the reconstruction error and cycle-consistency into consideration in order to fool the detection methods. In Figure 4.9 we can clearly see the undetected rate of CC-PGD increases compared to a standard PGD attack. However, there is a significant performance drop in the success rate of White-box CC-PGD (from PGD: 96.0% to CC-PGD: 69.0% on SVHN) as shown in Table 4.3. This indicates that the adversary needs to sacrifice some success rate in order not to be detected by our detection mechanism.

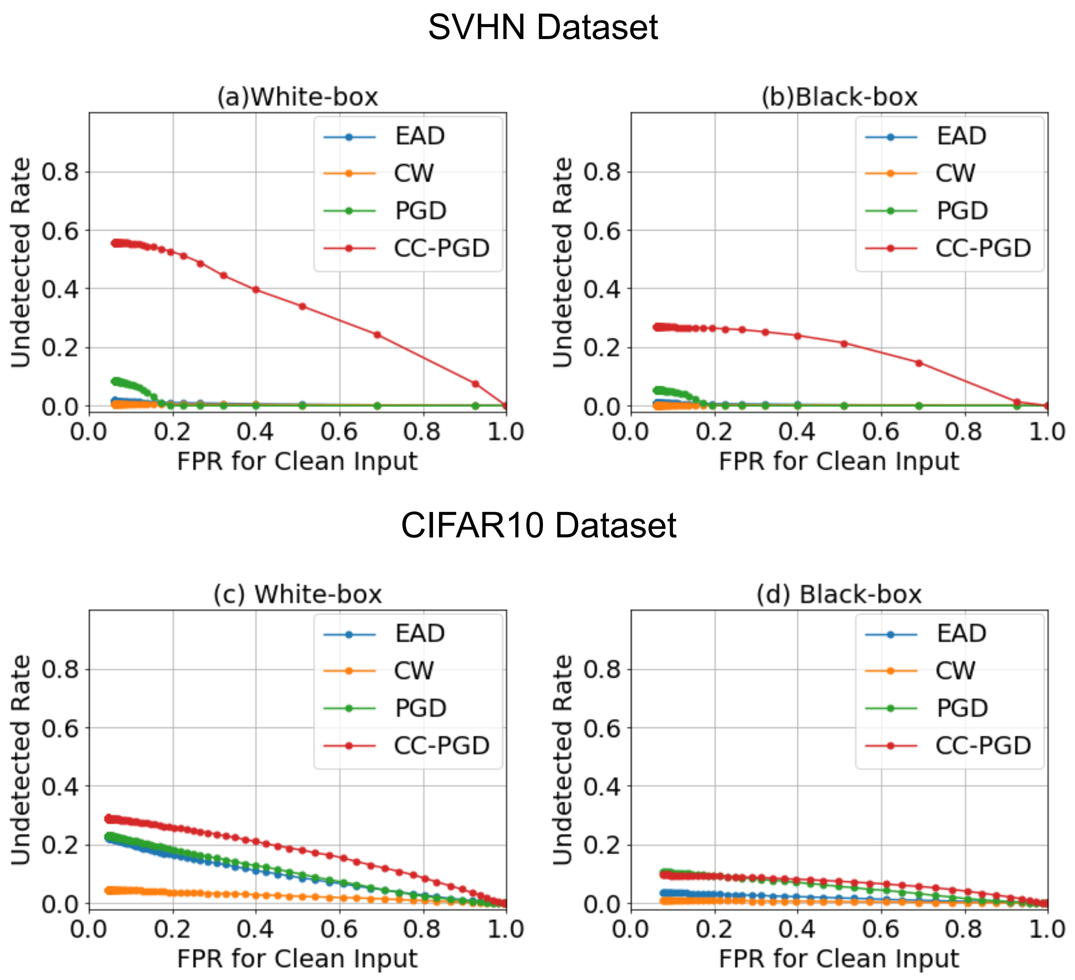


Figure 4.9. The Undetected Rate for white-box and black-box attacks versus False Positive Rate (FPR) for clean input on the SVHN and CIFAR-10 datasets. The strongest attack has the largest area under the line.

Table 4.3. Success rate of the white-box and black-box attacks for our deflecting model on the SVHN and CIFAR-10 dataset.

Dataset	EAD		CW		PGD		CC-PGD	
	White	Black	White	Black	White	Black	White	Black
SVHN	100.0%	10.1%	97.6%	1.7%	96.0%	28.7%	69.0%	37.0%
CIFAR-10	100.0%	6.9%	78.0%	1.6%	49.3%	15.5%	46.8%	12.9%

Table 4.4. Comparison of the Undetected Rate of the state-of-the-art detection methods on the CIFAR-10 dataset. For all the models, the maximum ℓ_∞ perturbation is $\varepsilon_\infty = 8/255$ of the pixel dynamic range and the False Positive Rate of the clean input are 5%. The best detection performance are highlighted in **bold**. (Smaller numbers indicate better detection performance.)

Detection Methods	Statistical Test	Classifier-based	Ours
CW	0.1%	**0.0**%	4.6%
Defense-aware PGD	97.8%	98.4%	**28.9**%

- **Comparison with State-of-the-Art Detection Methods**

  We compare our detection methods with the most recent statistical test-based detection method [52] and a classifier-based detection method proposed in [25]. In Table 4.4, we can see that although the statistical test [52] and the classifier-based detection method [25] can detect standard attacks successfully, they both fully fail against defense-aware attacks[1]. In contrast, our proposed reconstruction-based detection mechanism has the best undetected rate in detecting defense-aware adversarial attacks and a very small undetected rate of 4.6% in detecting CW attacks.

4.5.8 Detection of black-box Attacks

To study the effectiveness of our detection mechanisms, we also test our models on black-box attacks. In Figure 4.9 we can see an over 50% performance drop in the undetected rate when the inputs are black-box CC-PGD attacks on both datasets. The highest undetected rate of a black-box attack is around 13% on the CIFAR-10 dataset, which demonstrates that our detection mechanism can successfully detect black-box defense-aware attacks. In addition, the great gap of the success rate between white-box and black-box attacks shown in Table 4.3 indicates our defense model significantly reduces the transferability of all kinds of adversarial attacks.

[1]The numbers of statistical test and classifier-based detection in the Table 4.4 are extracted from [25]. Since the success rate of the attacks are close to 100%, the undetected rate is roughly (1 - True Positive Rate).

4.5.9 Examples of Adversarial Attacks and Reconstructions

We display successful adversarial attacks but detected by our detection mechanism, and display all the reconstructions when the input are EAD attacks (on the left) and CW attacks (on the right) in Figure 4.10 for the SVHN dataset. We also show the successful and detected adversarial PGD attacks (on the left) and our CC-PGD attacks (on the right) in Figure 4.11 for CIFAR-10 dataset.

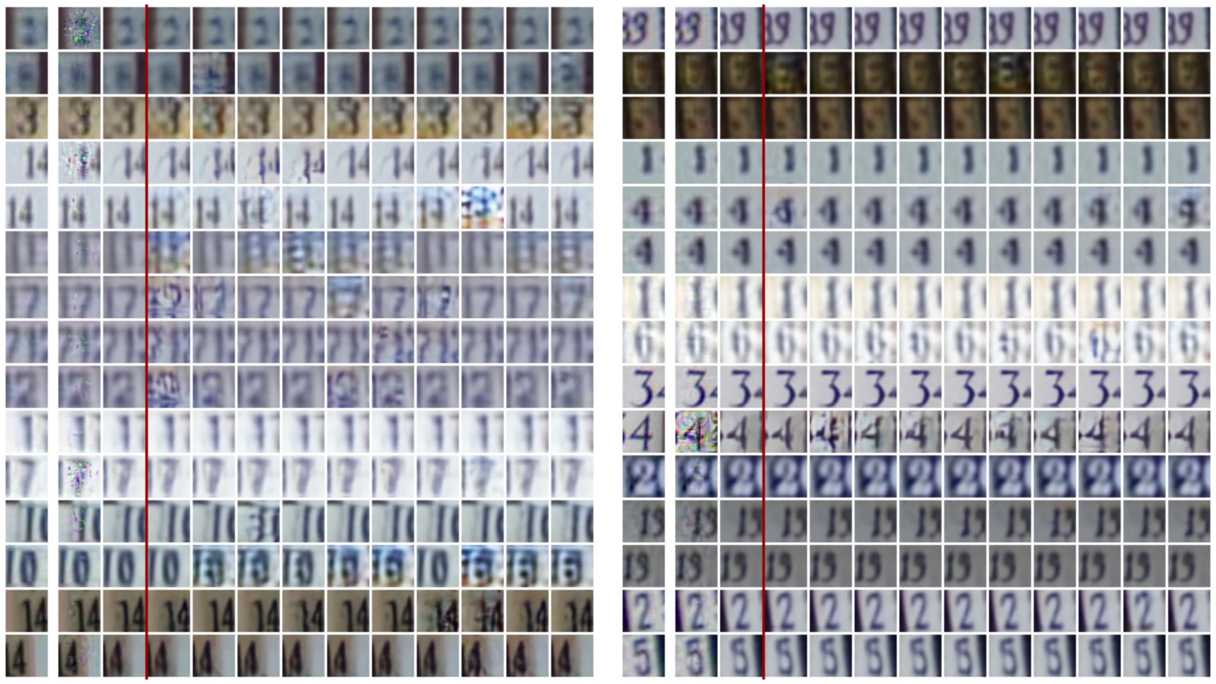


Figure 4.10. Successful but detected adversarial EAD attacks (on the left) and CW attacks (on the right) and the corresponding capsule reconstructions on SVHN. The first column is the clean input, the second column is the adversarial example, the third column is the winning-capsule reconstruction, the last ten columns are the reconstructions corresponding to class 0 to 9.

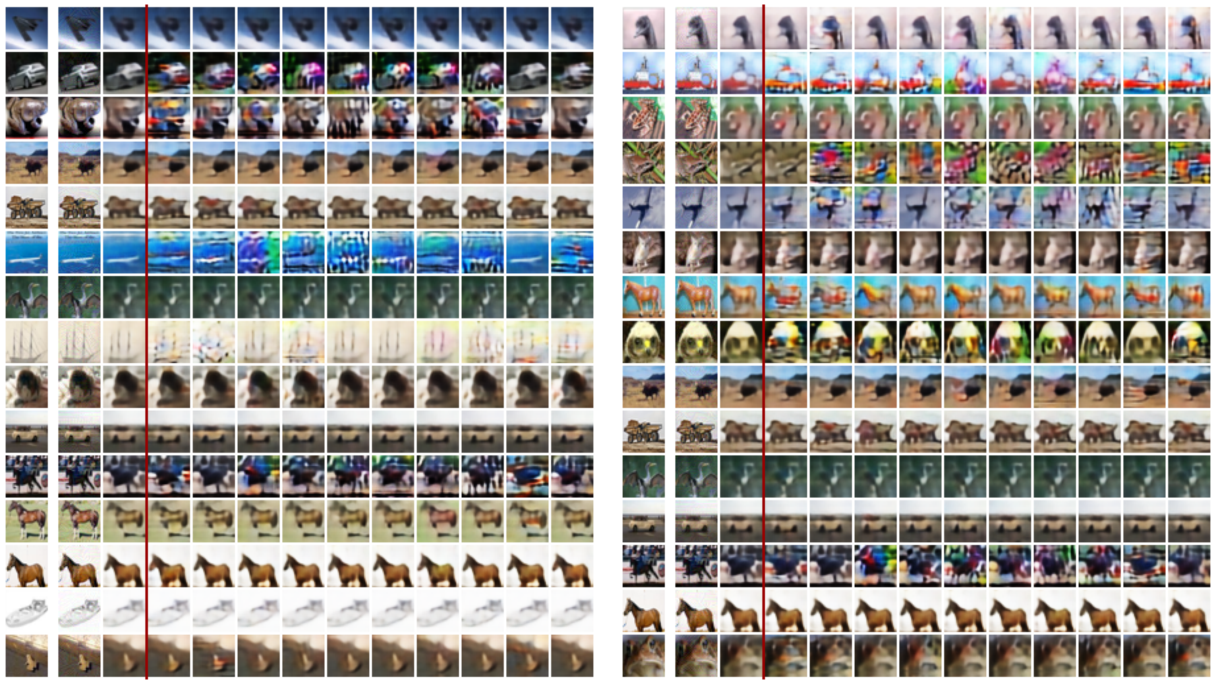


Figure 4.11. Successful but detected adversarial PGD attacks (on the left) and our CC-PGD attacks (on the right) and the corresponding capsule reconstructions on CIFAR-10. The first column is the clean input, the second column is the adversarial example, the third column is the winning-capsule reconstruction, the last ten columns are the reconstructions corresponding to class 0 to 9. The maximal ℓ_∞ bound to the adversarial perturbation is 8/255.

4.6 Deflected Attacks

The numbers that presented earlier in this chapter have implicitly assumed all adversarial attacks still resemble the initial class, and therefore classifying them as the target class would constitute a mistake. This assumption may not be true in practice. We have discussed the ability of our model to deflect adversarial attacks by having adversarial gradients aligned with the class conditional data distribution, thereby making adversarial attacks resemble the target class. In order to quantify these claims we need to evaluate human performance on the adversarial attacks against our model.

4.6.1 Human Study on SVHN

In order to validate our claim that our method can deflect adversarial attacks, we performed a human study. We made use of the Amazon Mechanical Turk web service to recruit participants and asked people to label SVHN digits. Each time, they were shown a single image which was randomly sampled from the following five different sets: 1) clean images from the SVHN test set, 2) the undetected and successful black-box PGD and CC-PGD adversarial attacks against our deflecting model, 3) the undetected and successful white-box PGD and CC-PGD adversarial attacks against our deflecting model, 4) the successful black-box PGD attacks generated to attack a standard CNN classifier[2], 5) the successful white-box PGD attacks for the CNN classifier. The maximal adversarial perturbation of all the ℓ_∞ norm-based attacks are bounded by the same $\varepsilon_\infty = 16/255$. The recruiters were asked to classify each image as a digit between 0 and 9. If multiple digits occurred in one image, we asked people to label the digit closest to the center of the image. We did not limit the time that people could spend in labeling each image and we did not explain the purpose of this study to the users other than it was a research study. In this way, we had 1500 images labeled in total and each image was labeled by five different users. We then calculated the percentage of uniformly labeled images that were classified as either the original class or the adversarial target class. The results are summarized in Figure 4.12.

We can see that 69.7% of successful and undetected black-box attacks against our model were classified as the adversarial target. This means that when our defense is attacked with adversarial attacks generated within a standard ℓ_∞ bound, not only are the results visibly different than the source image, they resemble the target class. In this way, these attacks are successfully deflected and can hardly be said to be adversarial, as the network is classifying them the same way our human testers classified them. This is not the case for the baseline CNN model, where only 14.3% of the successful black-box PGD attacks were labeled as the target class. In addition, compared to the white-box attacks, more undetected and successful adversarial attacks generated

[2]The CNN classifier has the same network architecture as the classification network in our deflecting model except that we replace the CapsLayer with a convolutional layer.

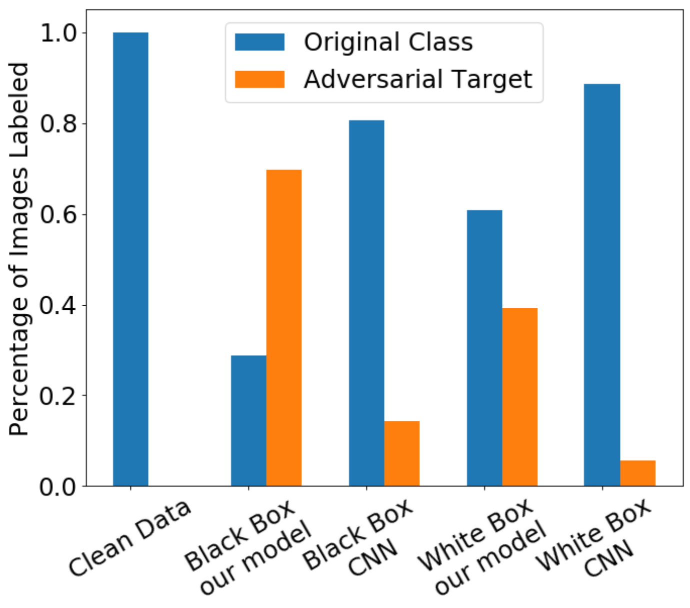


Figure 4.12. The human study results on SVHN. The maximal ℓ_∞ perturbation is 16/255.

under the black-box setting are deflected to resemble the target class. This suggests that to attack our deflecting model in a more practical and challenging setting (black-box), the attack ends up being deflected in order not to be detected. Some examples of deflected adversarial attacks on SVHN are shown in Figure 4.13.

4.6.2 Deflected Attacks on CIFAR-10

To show that our model can effectively deflect adversarial attacks on the CIFAR-10 dataset, we have chosen a deflected adversarial attack for each class with a maximal ℓ_∞ norm as 25/255, displayed in Figure 4.13. It is apparent that the clean input has been perturbed to have the representative features of the target class, in order to fool both the classifier and our detection mechanisms. As a result, these adversarial attacks are also successfully deflected by our model. Unlike SVHN, for which human evaluators reliably classified the attacks as the target label, the generated adversarial attacks against our deflecting model on the CIFAR-10 do not reliably resemble the target class, though they are much harder to identify than the clean data.

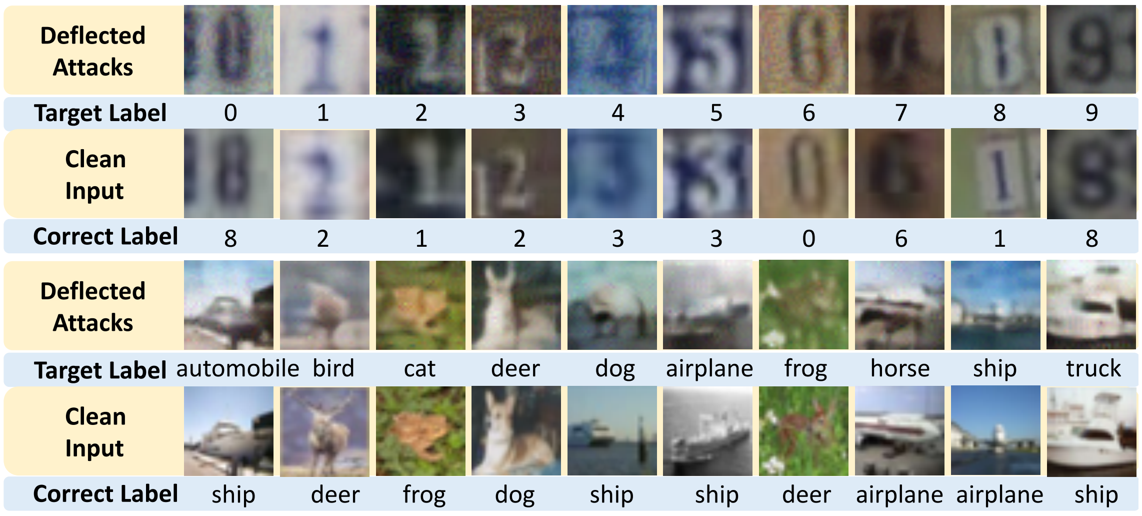


Figure 4.13. Deflected adversarial attacks on SVHN and CIFAR-10. The maximal ℓ_∞ perturbation is 16/255 for SVHN and 25/255 for CIFAR-10.

4.7 Conclusion

In this chapter, we introduce a new approach which presents a step towards ending the battle between defenses and attacks by deflecting adversarial attacks. To this end, we propose a new cycle-consistency loss to encourage the winning-capsule reconstruction of the CapsNet to closely match the class-conditional distribution. By making use of the three detection mechanisms, we are able to detect standard adversarial attacks based on three different distance metrics with a low False Positive Rate on the SVHN and CIFAR-10 datasets. To specifically attack our detection mechanisms, we propose a defense-aware attack and find that our model achieves drastically lower undetected rates for defense aware attacks compared to state-of-the-art methods. In addition, a large percentage of the undetected and successful attacks are deflected by our model to resemble the adversarial target class, so they cannot be considered as adversarial any more. This is verified by a human study showing that 70% of the successful and undetected black-box adversarial attacks are classified unanimously by humans as the target class on the SVHN dataset.

This chapter has been submitted for publication of the material as it may appear in the

Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR20) (Yao Qin, Nicholas Frosst, Colin Raffel, Garrison Cottrell and Geoffrey Hinton, “Deflecting Adversarial Attacks”). The dissertation author was the primary investigator and author of this paper.

Chapter 5

Designing Adversarial Attacks

5.1 Introduction

Adversarial examples [63] are inputs that have been specifically designed by an adversary to cause a machine learning algorithm to produce a misclassification [5]. Initial work on adversarial examples focused mainly in the domain of image classification. In order to differentiate properties of adversarial examples on neural networks in general from properties which hold true only on images, it is important to study adversarial examples in different domains. Indeed, adversarial examples are known to exist on domains ranging from reinforcement learning [26] to reading comprehension [29] to speech recognition [11]. This paper focuses on the latter of these domains, where [11] showed that any given source audio sample can be perturbed slightly so that an automatic speech recognition (ASR) system would transcribe the audio as any different target sentence.

To date, adversarial examples on ASR differ from adversarial examples on images in two key ways. First, adversarial examples on images are imperceptible to humans: it is possible to generate an adversarial example without changing the 8-bit brightness representation [63]. Conversely, adversarial examples on ASR systems are often perceptible. While the perturbation introduced is often small in magnitude, upon listening it is obvious that the added perturbation is present [57]. Second, adversarial examples on images work in the physical world [33] (e.g., even when taking a picture of them). In contrast, adversarial examples on ASR systems do not yet work in such an "over-the-air" setting where they are played by a speaker and recorded by a microphone.

In this chapter, we improve the construction of adversarial examples on ASR and match the power of attacks on images by developing adversarial examples which are imperceptible, and make steps towards robust adversarial examples[1].

In order to generate imperceptible adversarial examples, we depart from the common ℓ_p distance measure widely used for adversarial example research. Instead, we make use of the psychoacoustic principle of auditory masking, and only add the adversarial perturbation to regions of the audio where it will not be heard by a human, even if this perturbation is not "quiet" in terms of absolute energy.

Further investigating properties of adversarial examples which appear to be different from images, we examine the ability of an adversary to construct physical-world adversarial examples [33]. These are inputs that, even after taking into account the distortions introduced by the physical world, remain adversarial upon classification. We make initial steps towards developing audio which can be played over-the-air by designing audio which remains adversarial after being processed by random room-environment simulators [56].

Finally, we additionally demonstrate that our attack is capable of attacking a modern, state-of-the-art Lingvo ASR system [59].

5.2 Related Work

We build on a long line of work studying the robustness of neural networks. This research area largely began with [5, 63], who first studied *adversarial examples* for deep neural networks.

This paper focuses on adversarial examples on automatic speech recognition systems. Early work in this space [19, 14] was successful when generating *untargeted* adversarial examples that produced incorrect, but arbitrary, transcriptions. A concurrent line of work succeeded at

[1]The project webpage is at http://cseweb.ucsd.edu/~yaq007/imperceptible-robust-adv.html

generating targeted attacks in practice, even when played over a speaker and recorded by a microphone (a so-called "over-the-air" attack) but only by both (a) synthesizing completely new audio and (b) targeting older, traditional (i.e., not neural network based) speech recognition systems [8, 73, 60].

These two lines of work were partially unified by [11] who constructed adversarial perturbations for speech recognition systems targeting arbitrary (multi-word) sentences. However, this attack was neither effective over-the-air, nor was the adversarial perturbation completely inaudible; while the perturbations it introduces are very quiet, they can be heard by a human (see § 5.7.2). Concurrently, the CommanderSong [72] attack developed adversarial examples that are effective over-the-air but at a cost of introducing a significant perturbation to the original audio.

Following this, concurrent work with ours develops attacks on deep learning ASR systems that either work over-the-air or are less obviously perceptible.

- [70], create adversarial examples which can be played over-the-air. These attacks are highly effective on short two- or three-word phrases, but not on the full-sentence phrases originally studied. Further, these adversarial examples often have a significantly larger perturbation, and in one case, the perturbation introduced had a *higher* magnitude than the original audio.

- [57] work towards developing attacks that are less perceptible through using "Psychoacoustic Hiding" and attack the Kaldi system, which is partially based on neural networks but also uses some "traditional" components such as a Hidden Markov Model instead of an RNN for final classification. Because of the system differences we can not directly compare our results to that of this paper, but we encourage the reader to listen to examples from both papers.

Our concurrent work manages to achieve both of these results (almost) simultaneously: we generate adversarial examples that are both nearly imperceptible and also remain effective after simulated distortions. Simultaneously, we target a state-of-the-art network-based ASR system, Lingvo, as opposed to Kaldi and generate full-sentence adversarial examples as opposed to targeting short phrases.

A final line of work extends adversarial example generation on ASR systems from the white-box setting (where the adversary has complete knowledge of the underlying classifier) to the black-box setting [30, 66] (where the adversary is only allowed to query the system). This work is complementary and independent of ours: we assume a white-box threat model.

5.3 Background

5.3.1 Problem Definition

Given an input audio waveform x, a target transcription y and an automatic speech recognition (ASR) system $f(\cdot)$ which outputs a final transcription, our objective is to construct an imperceptible and targeted adversarial example x' that can attack the ASR system when played over-the-air. That is, we seek to find a small perturbation δ, which enables $x' = x + \delta$ to meet three requirements:

- **Targeted**: the classifier is fooled so that $f(x') = y$ and $f(x) \neq y$. Untargeted adversarial examples on ASR systems often only introduce spelling errors and so are less interesting to study.

- **Imperceptible**: x' sounds so similar to x that humans cannot differentiate x' and x when listening to them.

- **Robust**: x' is still effective when played by a speaker and recorded by a microphone in an over-the-air attack. (We do not achieve this goal completely, but do succeed at simulated environments.)

5.3.2 ASR Model

We mount our attacks on the **Lingvo** classifier [59], a state-of-the-art sequence-to-sequence model [62] with attention [4] whose architecture is based on the Listen, Attend and

Spell (LAS) model [12]. It feeds filter bank spectra into an encoder consisting of a stack of convolutional and LSTM layers, which conditions an LSTM decoder that outputs the transcription. The use of the sequence-to-sequence framework allows the entire model to be trained end-to-end with the standard cross-entropy loss function. The Lingvo system achieves the state-of-the-art performance for automatic speech recognition. The detailed parameter settings of the Lingvo classifier is introduced in [59].

The key improvement of LAS over previous CTC-based models (*e.g.,* DeepSpeech) is that it can outputs the character sequences without making any independence assumption between the characters. Each character is predicted based on a probability distribution conditioned on all the characters seen previously. Therefore, the LAS model can output the full transcription without specific post processing.

5.3.3 Threat Model

In this chapter, as is done in most prior work, we consider the white box threat model where the adversary has full access to the model as well as its parameters. In particular, the adversary is allowed to compute gradients through the model in order to generate adversarial examples.

When we mount over-the-air attacks, we do not assume we know the exact configurations of the room in which the attack will be performed. Instead, we assume we know the *distribution* from which the room will be drawn, and generate adversarial examples so as to be effective on any room drawn from this distribution.

5.3.4 Adversarial Example Generation

Adversarial examples are typically generated by performing gradient descent with respect to the input on a loss function designed to be minimized when the input is adversarial [63]. Specifically, let x be an input to a neural network $f(\cdot)$, let δ be a perturbation, and let $\ell(f(x), y)$ be a loss function that is minimized when $f(x) = y$. Most work on adversarial examples focuses on minimizing the max-norm ($\|\cdot\|_\infty$ norm) of δ. Then, the typical adversarial example generation algorithm [63, 10, 41] solves

$$\begin{aligned}&\text{minimize } \ell(f(x+\delta),y)+\alpha\cdot\|\delta\|\\&\text{such that } \|\delta\|<\varepsilon\end{aligned}$$

(where in some formulations $\alpha = 0$). Here, ε controls the maximum perturbation introduced.

To generate adversarial examples on ASR systems, [11] set ℓ to the CTC-loss and use the max-norm which has the effect of adding a small amount of adversarial perturbation consistently throughout the audio sample.

5.4 Imperceptible Adversarial Examples

Unlike on images, where minimizing ℓ_p distortion between an image and the nearest misclassified example yields a visually indistinguishable image, on audio, this is not the case [57]. Thus, in this work, we depart from the ℓ_p distortion measures and instead rely on the extensive work which has been done in the audio space for capturing the human perceptibility of audio.

5.4.1 Psychoacoustic Models

A good understanding of the human auditory system is critical in order to be able to construct imperceptible adversarial examples. In this chapter, we use *frequency masking*, which refers to the phenomenon that a louder signal (the "masker") can make other signals at nearby frequencies (the "maskees") imperceptible [45, 39]. In simple terms, the masker can be seen as creating a "masking threshold" in the frequency domain. Any signals which fall under this threshold are effectively imperceptible.

Because the masking threshold is measured in the frequency domain, and because audio signals change rapidly over time, we first compute the short-time Fourier transform of the raw audio signal to obtain the spectrum of overlapping sections (called “windows”) of a signal. The window size N is 2048 samples which are extracted with a “hop size” of 512 samples and are windowed with the modified Hann window. We denote $s_x(k)$ as the kth bin of the spectrum of frame x.

Then, we compute the log-magnitude power spectral density (PSD) as follows:

$$p_x(k) = 10\log_{10}\left|\frac{1}{N}s_x(k)\right|^2. \tag{5.1}$$

The normalized PSD estimate $\bar{p}_x(k)$ is defined by [39]

$$\bar{p}_x(k) = 96 - \max_k\{p_x(k)\} + p_x(k) \tag{5.2}$$

5.4.2 Masking Threshold

In this section, we detail how we compute the frequency masking threshold for constructing imperceptible adversarial examples. This procedure is based on psychoacoustic principles which were refined over many years of human studies. For further background on psychoacoustic models, we refer the interested reader to [39, 45].

Given an audio input, in order to compute its masking threshold, first we need to identify the maskers, whose normalized PSD estimate $\bar{p}_x(k)$ must satisfy three criteria: 1) they must be local maxima in the spectrum; 2) they must be higher than the threshold in quiet; and 3) they have the largest amplitude within 0.5 Bark (a psychoacoustically-motivated frequency scale) around the masker’s frequency. Then, each masker’s masking threshold can be approximated using the simple two-slope spread function, which is derived to mimic the excitation patterns of maskers. Finally, the global masking threshold $\theta_x(k)$ is a combination of the individual masking threshold as well as the threshold in quiet via addition (because the effect of masking is additive in the logarithmic domain).

- **Step 1: Identifications of Maskers**

  In order to compute the frequency masking threshold of an input signal $x(n)$, where $0 \leq n \leq N$, we need to first identify the maskers. There are two different classes of maskers: tonal and nontonal maskers, where nontonal maskers have stronger masking effects compared to tonal maskers. Here we simply treat all the maskers as tonal ones to make sure the threshold that we compute can always mask out the noise. The normalized PSD estimate of the tonal maskers $\bar{p}_x^m(k)$ must meet three criteria. First, they must be local maxima in the spectrum, satisfying:

$$\bar{p}_x(k-1) \leq \bar{p}_x^m(k) \leq \bar{p}_x(k+1), \tag{5.3}$$

  where $0 \leq k < \frac{N}{2}$.

  Second, the normalized PSD estimate of any masker must be higher than the threshold in quiet ATH(k), which is:

$$\bar{p}_x^m(k) \geq \text{ATH}(k), \tag{5.4}$$

  where ATH(k) is approximated by the following frequency-dependency function:

$$\text{ATH}(f) = 3.64(\frac{f}{1000})^{-0.8} - 6.5\exp\{-0.6(\frac{f}{1000} - 3.3)^2\} + 10^{-3}(\frac{f}{1000})^4. \tag{5.5}$$

  The quiet threshold only applies to the human hearing range of 20Hz $\leq f \leq$ 20kHz. When we perform short time Fourier transform (STFT) to a signal, the relation between the frequency f and the index of sampling points k is

$$f = \frac{k}{N} \cdot f_s, \quad 0 \leq f < \frac{f_s}{2} \tag{5.6}$$

  where f_s is the sampling frequency and N is the window size.

Last, the maskers must have the highest PSD within the range of 0.5 Bark around the masker's frequency. Human's main hearing range between 20Hz and 16kHz is divided into 24 non-overlapping critical bands, whose unit is Bark, varying as a function of frequency f as follows:

$$b(f) = 13\arctan(\frac{0.76f}{1000}) + 3.5\arctan(\frac{f}{7500})^2. \tag{5.7}$$

As the effect of masking is additive in the logarithmic domain, the PSD estimate of the masker is further smoothed with its neighbors by:

$$\bar{p}_x^m(\bar{k}) = 10\log_{10}[10^{\frac{\bar{p}_x(k-1)}{10}} + 10^{\frac{\bar{p}_x^m(k)}{10}} + 10^{\frac{\bar{p}_x(k+1)}{10}}] \tag{5.8}$$

- **Step 2: Individual masking thresholds**

An individual masking threshold is better computed with frequency denoted at the Bark scale because the spreading functions of the masker would be similar at different Barks. We use $b(i)$ to represent the Bark scale of the frequency index i. There are a number of spreading functions introduced to imitate the characteristics of maskers and here we choose the simple two-slope spread function:

$$\mathrm{SF}[b(i), b(j)] = \begin{cases} 27[b(j) - b(i)], & \text{if } [b(j) - b(i)] \leq 0. \\ G(b(i)) \cdot [b(j) - b(i)], & \text{otherwise} \end{cases} \tag{5.9}$$

where $G(b(i)) = [-27 + 0.37\max\{\bar{p}_x^m(b(i)) - 40, 0\}]$ and $b(i)$ and $b(j)$ are the Bark scale of the masker at the frequency index i and the maskee at frequency index j respectively. Then, $T[b(i), b(j)]$ refers to the masker at Bark index $b(i)$ contributing to the masking effect on the maskee at Bark index $b(j)$. Empirically, the threshold $T[b(i), b(j)]$ is calculated by:

$$T[b(i), b(j)] = \bar{p}_x^m(b(i)) + \Delta_m[b(i)] + \mathrm{SF}[b(i), b(j)], \tag{5.10}$$

where $\Delta_m[b(i)] = -6.025 - 0.275b(i)$ and SF$[b(i), b(j)]$ is the spreading function.

- **Step 3: Global masking threshold**

  The global masking threshold is a combination of individual masking thresholds as well as the threshold in quiet via addition. The global masking threshold at frequency index i measured with Decibels (dB) is calculated according to:

$$\theta_x(i) = 10\log_{10}[10^{\frac{ATH(i)}{10}} + \sum_{j}^{N_m} 10^{\frac{T[b(j),b[i]]}{10}}], \tag{5.11}$$

  where N_m is the set of all the selected maskers. The computed θ_x is used as the frequency masking threshold for the input audio x to construct imperceptible adversarial examples.

When we add the perturbation δ to the audio input x, if the normalized PSD estimate of the perturbation $\bar{p}_\delta(k)$ is under the frequency masking threshold of the original audio $\theta_x(k)$, the perturbation will be masked out by the raw audio and therefore be inaudible to humans. The normalized PSD estimate of the perturbation $\bar{p}_\delta(k)$ can be calculated via:

$$\bar{p}_\delta(k) = 96 - \max_k\{p_x(k)\} + p_\delta(k). \tag{5.12}$$

where $p_\delta(k) = 10\log_{10}|\frac{1}{N}s_\delta(k)|^2$ and $p_x(k) = 10\log_{10}|\frac{1}{N}s_x(k)|^2$ are the PSD estimate of the perturbation and the original audio input.

5.4.3 Optimization with Masking Threshold

- **Loss function**

  Given an audio example x and a target phrase y, we formulate the problem of constructing an imperceptible adversarial example $x' = x + \delta$ as minimizing the loss function $\ell(x, \delta, y)$, which is defined as:

$$\ell(x, \delta, y) = \ell_{net}(f(x+\delta), y) + \alpha \cdot \ell_\theta(x, \delta) \tag{5.13}$$

where ℓ_{net} requires that the adversarial examples fool the audio recognition system into making a targeted prediction y, where $f(x) \neq y$. In the Lingvo model, the simple cross entropy loss function is used for ℓ_{net}. The term ℓ_θ constrains the normalized PSD estimation of the perturbation $\bar{p}_\delta(k)$ to be under the frequency masking threshold of the original audio $\theta_x(k)$. The hinge loss is used here to compute the loss for masking threshold:

$$\ell_\theta(x,\delta) = \frac{1}{\lfloor \frac{N}{2} \rfloor + 1} \sum_{k=0}^{\lfloor \frac{N}{2} \rfloor} \max\left\{\bar{p}_\delta(k) - \theta_x(k), 0\right\}, \tag{5.14}$$

where N is the predefined window size and $\lfloor x \rfloor$ outputs the greatest integer no larger than x. The adaptive parameter α is to balance the relative importance of these two criteria.

- **Stability**

The existence of the log function in the threshold $\theta_x(k)$ and the normalized PSD estimate of the perturbation $\bar{p}_\delta(k)$ leads to instability during back-propagation. Therefore, we remove the term $10\log_{10}$ in the PSD estimate of $p_\delta(k)$ and $p_x(k)$ and then they become:

$$p_\delta(k) = \left|\frac{1}{N} s_\delta(k)\right|^2, \quad p_x(k) = \left|\frac{1}{N} s_x(k)\right|^2 \tag{5.15}$$

and the normalized PSD of the perturbation turns into

$$\bar{p}_\delta(k) = \frac{10^{9.6} p_\delta(k)}{\max_k\{p_x(k)\}}. \tag{5.16}$$

Correspondingly, the threshold $\theta_x(k)$ becomes:

$$\theta_x(k) = 10^{\frac{\theta_x}{10}} \tag{5.17}$$

- **Two Stage Attack**

  Empirically, we find it is difficult to directly minimize the masking threshold loss function via backpropagation without any constraint on the magnitude of the perturbation δ. This is reasonable because it is very challenging to fool the neural network and limit a very large perturbation to be under the masking threshold in the frequency domain at the same time. In contrast, if the perturbation δ is relatively small in magnitude, then it will be much easier to push the remaining distortion under the frequency masking threshold.

  Therefore, we divide the optimization into two stages: the first stage of optimization focuses on finding a relatively small perturbation to fool the network (as was done in prior work [11]) and the second stage makes the adversarial examples imperceptible.

  In the first stage, we set α in Eqn 5.13 to be zero and clip the perturbation to be within a relatively small range. As a result, the first stage solves:

$$\begin{aligned} &\text{minimize } \ell_{net}(f(x+\delta), y) \\ &\text{such that } \|\delta\| < \varepsilon \end{aligned} \tag{5.18}$$

  where $\|\delta\|$ represents the $\|\cdot\|_\infty$ max-norm of δ. Specifically, we begin by setting $\delta = 0$ and then on each iteration:

$$\delta \leftarrow \text{clip}_\varepsilon(\delta - lr_1 \cdot \text{sign}(\nabla_\delta \ell_{net}(f(x+\delta), y))), \tag{5.19}$$

  where lr_1 is the learning rate and and $\nabla_\delta \ell_{net}$ is the gradient of ℓ_{net} with respect to δ. We initially set ε to a large value and then gradually reduced during optimization following [11].

  The second stage focuses making the adversarial examples imperceptible, with an *unbounded* max-norm; instead, δ is only constrained by the masking threshold constraints.

Specifically, initialize δ with δ_{im}^* optimized in the first stage and then on each iteration:

$$\delta \leftarrow \delta - lr_2 \cdot \nabla_\delta \ell(x, \delta, y), \qquad (5.20)$$

where lr_2 is the learning rate and $\nabla_\delta \ell$ is the gradient of ℓ with respect to δ. The loss function $\ell(x, \delta, y)$ is defined in Eqn. 5.13. The parameter α that balances the network loss $\ell_{net}(f(x+\delta), y)$ and the imperceptibility loss $\ell_\theta(x, y)$ is initialized with a small value, *e.g.*, 0.05, and is adaptively updated according to the performance of the attack. Specifically, every twenty iterations, if the current adversarial example successfully fools the ASR system (i.e. $f(x+\delta) = y$), then α is increased to attempt to make the adversarial example less perceptible. Correspondingly, every fifty iterations, if the current adversarial example fails to make the targeted prediction, we decrease α. We check for attack failure less frequently than success (fifty vs. twenty iterations) to allow more iterations for the network to converge.

5.4.4 Implementation Details

In order to construct imperceptible adversarial examples, we divide the optimization into two stages. In the first stage, the learning rate lr_1 is set to be 100 and the number of iterations T_1 is 1000 as [11]. The max-norm bound ε starts from 2000 and will be gradually reduced during optimization. In the second stage, the number of iterations T_2 is 4000. The learning rate lr_2 starts from 1 and will be reduced to be 0.1 after 3000 iterations. The adaptive parameter α which balances the importance between ℓ_{net} and ℓ_θ begins with 0.05 and gradually updated based on the performance of adversarial examples. Adam optimizer [31] is used to construct the imperceptible adversarial examples. Algorithm **??** shows the details of the two-stage optimization.

5.5 Robust Adversarial Examples

5.5.1 Acoustic Room Simulator

In order to improve the robustness of adversarial examples when playing over-the-air, we use an acoustic room simulator to create artificial utterances (speech with reverberations) that mimic playing the audio over-the-air. The transformation function in the acoustic room simulator, denoted as t, takes the clean audio x as an input and outputs the simulated speech with reverberation $t(x)$. First, the room simulator applies the classic Image Source Method introduced in [1, 56] to create the room impulse response r based on the room configurations (the room dimension, source audio and target microphone's location, and reverberation time). Then, the generated room impulse response r is convolved with the clean audio to create the speech with reverberation, to obtain $t(x) = x * r$ where $*$ denotes the convolution operation. To make the generated adversarial examples robust to various environments, multiple room impulse responses r are used. Therefore, the transformation function t follows a chosen distribution $\mathscr{T}$ over different room configurations.

5.5.2 Optimization with Reverberations

In this section, our objective is to make the perturbed speech with reverberation (rather than the clean audio) fool the ASR system. As a result, the generated adversarial examples $x' = x + \delta$ will be passed through the room simulator first to create the simulated speech with reverberation $t(x')$, mimicking playing the adversarial examples over-the-air, and then the simulated $t(x')$ will be fed as the new input to fool the ASR system, aiming at $f(t(x')) = y$. Simultaneously, the adversarial perturbation δ should be relatively small in order not to be audible to humans.

In the same manner as the Expectation over Transformation in [3], we optimize the expectation of the loss function over different transformations $t \sim \mathscr{T}$ as follows:

$$\begin{aligned} &\text{minimize } \ell(x,\delta,y) = \mathop{\mathbb{E}}_{t \sim \mathscr{T}} \left[\ell_{net}(f(t(x+\delta)),y)\right] \\ &\text{such that } \|\delta\| < \varepsilon. \end{aligned} \tag{5.21}$$

Rather than directly targeting $f(x+\delta) = y$, we apply the loss function l_{net} (the cross entropy loss in the Lingvo network) to the classification of the transformed speech $f(t(x+\delta)) = y$. We approximate the gradient of the expected value via independently sampling a transformation t from the distribution $\mathscr{T}$ at each gradient descent step.

In the first I_{r_1} iterations, we initialize ε with a sufficiently large value and gradually reduce it following [11]. We consider the adversarial example successful if it successfully fools the ASR system under a single random room configuration; that is, if $f(t(x+\delta)) = y$ for just one $t(\cdot)$. Once this optimization is complete, we obtain the max-norm bound for δ, denoted as ε_r^*. We will then use the perturbation δ_r^* as an initialization for δ in the next stage.

Then in the following I_{r_2} iterations, we finetune the perturbation δ with a much smaller learning rate. The max-norm bound ε is increased to $\varepsilon_r^{**} = \varepsilon_r^* + \Delta$, where $\Delta > 0$, and held constant during optimization. During this phase, we only consider the attack successful if the adversarial example successfully fools a set of randomly chosen transformations $\Omega = \{t_1, t_2, \cdots, t_M\}$, where $t_i \sim \mathscr{T}$ and M is the size of the set Ω. The transformation set Ω is randomly sampled from the distribution $\mathscr{T}$ at each gradient descent step. In other words, the adversarial example $x' = x + \delta$ generated in this stage satisfies $\forall t_i \in \Omega$, $f(t_i(x+\delta)) = y$. In this way, we can generate robust adversarial examples that successfully attack ASR systems when the exact room environment is not known ahead of time, whose configuration is drawn from a pre-defined distribution.

It should be emphasized that there is a tradeoff between imperceptibility and robustness (as we will show experimentally in Section 5.7.2). If we increase the max amplitude of the perturbation ε_r^{**}, the robustness can always be further improved. Correspondingly, it becomes much easier for humans to perceive the adversarial perturbation and alert the ASR system.

In order to keep these adversarial examples mostly imperceptible, we therefore limit the ℓ_∞ amplitude of the perturbation to be in a reasonable range.

- **Implementation Details**

  To develop the robust adversarial examples that could work after being played over-the-air, we also optimize the adversarial perturbation in two stages. The first stage intends to find a relatively small perturbation while the second stage focuses on making the constructed adversarial example more robust to random room configurations. The learning rate lr_1 in the first stage is 50 and δ will be updated for 2000 iterations. The max-norm bound ε for the adversarial perturbation δ starts from 2000 as well and will be gradually reduced. In the second stage, the number of iterations is set to be 4000 and the learning rate lr_2 is 5. In this stage, ε is fixed and equals the optimized ε_r^* in the first stage plus Δ. The size of the transformation set Ω is set to be $M = 10$.

5.6 Imperceptible and Robust Attacks

By combining both of the techniques we developed earlier, we now develop an approach to generate both imperceptible and robust adversarial examples. This can be achieved by minimizing the loss

$$\ell(x,\delta,y) = \mathop{\mathbb{E}}_{t\sim\mathscr{T}}\left[\ell_{net}(f(t(x+\delta)),y) + \alpha\cdot\ell_\theta(x,\delta)\right], \tag{5.22}$$

where the cross entropy loss function $\ell_{net}(\cdot)$ is again the loss used for Lingvo, and the imperceptibility loss $\ell_\theta(\cdot)$ is the same as that defined in Eqn 5.14. Since we need to fool the ASR system when the speech is played after random perturbations, the cross entropy loss $\ell_{net}(f(t(x+\delta)),y)$ forces the transcription of the transformed adversarial example $t(x+\delta)$ to be y (again, as done earlier).

To further improve these adversarial examples to be imperceptible, we optimize $\ell_\theta(x,\delta)$ to constrain the perturbation δ to fall under the masking threshold of the clean audio in the frequency domain. This is much easier compared to optimizing the hinge loss $\ell_\theta(t(x),t(\delta)) = \max\{\bar{p}_{t(\delta)}(k) - \theta_{t(x)}(k), 0\}$ because the frequency masking threshold of the clean audio $\theta_x(k)$ can be pre-computed while the masking threshold of the speech with reverberation $\theta_{t(x)}(k)$ varies with the room reverberation r. In addition, optimizing $\ell_\theta(x,\delta)$ and $\ell_\theta(t(x),t(\delta))$ have similar effects based on the convolution theorem that the Fourier transform of a convolution of two signals is the pointwise product of their Fourier transforms. Note that the speech with reverberation $t(x)$ is a convolution of the clean audio x and a simulated room reverberation r, hence:

$$\mathscr{F}\{t(x)\} = \mathscr{F}\{x * r\} = \mathscr{F}\{x\} \cdot \mathscr{F}\{r\} \tag{5.23}$$

where $\mathscr{F}$ is the Fourier transform, $*$ denotes the convolution operation and $\cdot$ represents the pointwise product. We apply the short-time Fourier transform to the perturbation and the raw audio signal first in order to compute the power spectral density $\bar{p}_{t(\delta)}$ and the masking threshold $\theta_{t(x)}$ in the frequency domain. Since most of the energy in the room impulse response falls within the spectral analysis window size, the convolution theorem in Eqn 5.23 is approximately satisfied. Therefore, we arrive at:

$$(\bar{p}_{t(\delta)} - \theta_{t(x)}) \approx (\bar{p}_\delta - \theta_x) \cdot \mathscr{F}\{r\}. \tag{5.24}$$

As a result, optimizing the imperceptibility loss $\ell_\theta(x,\delta)$ can help in finding the optimal δ and in constructing the imperceptible adversarial examples that can attack the ASR systems in the physical world.

Specifically, we will first initialize δ with the perturbation δ_r^{**} that enables the adversarial examples to be robust in Section 5.5. Then in each iteration, we randomly sample a transformation

t from the distribution $\mathscr{T}$ and update δ according to:

$$\delta \leftarrow \delta - lr_3 \cdot \nabla_\delta \left[\ell_{net}(f(t(x+\delta),y)) + \alpha \cdot \ell_\theta(x,\delta))\right], \tag{5.25}$$

where lr_3 is the learning rate and α, a parameter that balances the importance of the robustness and the imperceptibility, is adaptively changed based on the performance of adversarial examples. Specifically, if the constructed adversarial example can successfully attack a set of randomly chosen transformations, then α will be increased to focus more on imperceptibility loss. Otherwise, α is decreased to make the attack more robust to multiple room environments.

- **Implementation Details**

  To construct imperceptible and robust adversarial examples, we begin with the robust adversarial examples generated in Section. 5.5. In the first stage, we focus on reducing the imperceptibility by setting the initial α to be 0.01 and the learning rate is set to be 1. We update the adversarial perturbation δ for 4000 iterations. If the adversarial example successfully attacks the ASR system in 4 out of 10 randomly chosen rooms, then α will be increased by 2. Otherwise, for every 50 iterations, α will be decreased by 0.5.

  In the second stage, we focus on improving the less perceptible adversarial examples to be more robust. The learning rate is 1.5 and α starts from a very small value of 0.00005. The perturbation will be further updated for 6000 iterations. If the adversarial example successfully attacks the ASR system in 8 out of 10 randomly chosen rooms, then α will be increased by 1.2.

Table 5.1. Examples of the original and targeted phrases on the LibriSpeech dataset.

Original phrase 1	the more she is engaged in her proper duties the less leisure will she have for it even as an accomplishment and a recreation
Targeted phrase 1	old will is a fine fellow but poor and helpless since missus rogers had her accident
Original phrase 2	a little cracked that in the popular phrase was my impression of the stranger who now made his appearance in the supper room
Targeted phrase 2	her regard shifted to the green stalks and leaves again and she started to move away

5.7 Evaluation

5.7.1 Datasets and Evaluation Metrics

- **Datasets**

  We use the LibriSpeech dataset [48] in our experiments, which is a corpus of 16KHz English speech derived from audiobooks and is used to train the Lingvo system [59]. We randomly select 1000 audio examples as source examples, and 1000 separate transcriptions from the test-clean dataset to be the targeted transcriptions. We ensure that each target transcription is around the same length as the original transcription because it is unrealistic and overly challenging to perturb a short audio clip (*e.g.,* 10 words) to have a much longer transcription (*e.g.,* 20 words). Examples of the original and targeted transcriptions are available in Table 5.1.

- **Evaluation Metrics**

  For automatic speech recognition, we evaluate our model using the standard word error rate (WER) metric, which is defined as WER $= \frac{S+D+I}{N_W} \times 100\%$, where S, D and I are the number of substitutions, deletions and insertions of words respectively, and N_W is the total number of words in the reference.

  We also calculate the success rate (sentence-level accuracy) as Accuracy $= \frac{N_s}{N_a} \times 100\%$, where N_a is the number of audio examples that we test, and N_s is the number of audio

Table 5.2. Sentence-level accuracy and WER for 1000 clean and (imperceptible) adversarially perturbed examples, fed without over-the-air simulation into the Lingvo model. In "Clean", the ground truth is the original transcription. In"Adversarial", the ground truth is the targeted transcription.

Input	Clean	Adversarial
Accuracy (%)	58.60	100.00
WER (%)	4.47	0.00

examples that are correctly transcribed. Here, "correctly transcribed" means the original transcription for clean audio and the targeted transcription for adversarial examples.

5.7.2 Imperceptibility Analysis

To attack the Lingvo ASR system, we construct 1000 imperceptible and targeted adversarial examples, one for each of the examples we sampled from the LibriSpeech test-clean dataset. Table5.2 shows the performance of the clean audio and the constructed adversarial examples. We can see that the word error rate (WER) of the clean audio is just 4.47% on the 1000 test examples, indicating the model is of high quality. Our imperceptible adversarial examples perform even better, and reach a 100% success rate.

Qualitative Human Study

Of the 1000 examples selected from the test set, we randomly selected 100 of these with their corresponding imperceptible adversarial example. We generate then generate an adversarial example using the prior work of [11] for the same target phrase; this attack again succeeds with 100% success. We perform three experiments to validate that our adversarial examples are imperceptible, especially compared to prior work.

- **Experimental Design.**

  We recruit 80 users online from Amazon Mechanical Turk. We give each user one of the three (nearly identical) experiments, each of which we describe below. In all cases, the experiments consist of 20 “comparisons tasks”, where we present the evaluator with some audio samples and ask them questions (described below) about the samples. We ask the user to listen to each sample with headphones on, and answer a simple question about the audio samples (the question is determined by which experiment we run, as given below). We do not explain the purpose of the study other than that it is a research study, and do not record any personally identifying information.[2] We randomly include a small number of questions with known, obvious answers; we remove 3 users from the study who failed to answer these questions correctly.

In all experiments, users have the ability to listen to audio files multiple times when they are unsure of the answer, making it as difficult as possible for our adversarial examples to pass as clean data. Users additionally have the added benefit of hearing 20 examples back-to-back, effectively “training” them to recognize subtle differences. Indeed, a permutation test finds users are statistically significantly better at distinguishing adversarial examples from clean audio during the second half of the experiment compared to the first half of the experiment, although the magnitude of the difference is small: only by about 3%.

Figure 5.1 summarizes the statistical results we give below.

- **Experiment 1: clean or noisy.**

  We begin with what we believe is the most representative experiment of how an attack would work in practice. We give users one audio sample and ask them to tell us if it has *any* background noise (e.g., static, echoing, people talking in the background).

  As a baseline, users believed that 19% of original clean audio samples contained some amount of noise, and 66% of users believed that the adversarial examples generated by [11] contained some amount of noise. In comparison, only 23% of users believe that the adversarial examples we generate contain any noise, a result that is not statistically

[2]Unfortunately, for this reason, we are unable to report aggregate statistics such as age or gender, slightly harming potential reproducibility.

significantly different from clean audio ($p > .05$). That is, when presented with just one audio sample in isolation, users do not believe the adversarial examples we generate are any noisier than the clean samples.

- **Experiment 2: identify the original.**

  We give users two audio samples and inform them that one of the audio samples is a modified version of the other; we ask the user to select the audio sample corresponding to the one which sounds like the *more natural* audio sample. This setup is much more challenging: when users can listen to both the before and after, it is often possible to pick up on the small amount of distortion that has been added. When comparing the original audio to the adversarial examples generated by [11], the evaluator chose the original audio 82% of the time. When we have the evaluator compare the imperceptible adversarial examples we generate to those of [11], our imperceptible examples are selected as the better audio sample 83% of the time—a difference that is not statistically distinguishable from comparing against the clean audio.

  However, when directly comparing the adversarial examples we generate to the clean audio, users prefer the clean audio still 66% of the time. Observe that the baseline percentage, when the samples are completely indistinguishable, is 50%. Thus, users only perform 16% better than random guessing at distinguishing our examples from clean examples.

- **Experiment 3: identical or not.**

  Finally, we perform the most difficult experiment: we present users with two audio files, and ask them if the audio samples are identical, or if there are *any* differences. As the baseline, when given the same audio sample twice, users agreed it was identical 85% of the time. (That is, in 15% of cases the evaluator wrongly heard a difference between the two samples.) When given a clean audio sample and comparing it to the audio generated by [11], users only believed them to be identical 24% of the time. Comparing clean audio to the adversarial examples we generate, user believed them to be completely identical 76% of the time, 3× more often than the adversarial examples generated by the baseline, but below the 85%-identical value for actually-identical audio.

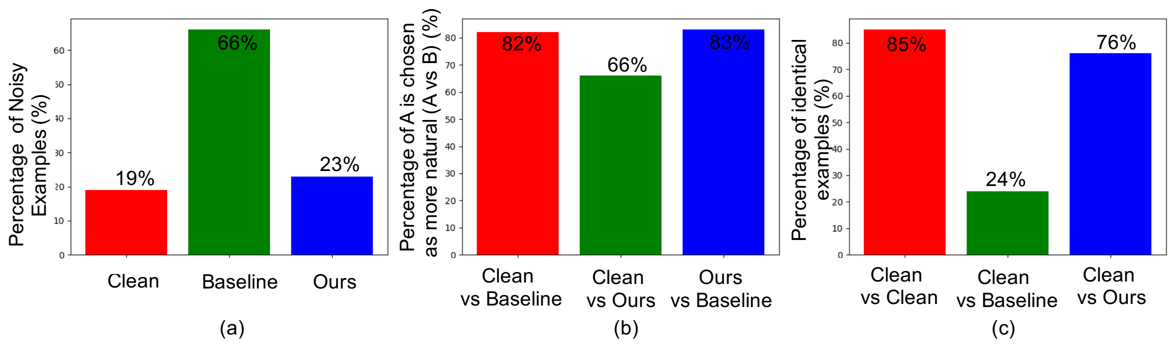


Figure 5.1. Results of human study for imperceptibility. Here baseline represents the adversarial example generated by [11], and ours denotes the imperceptible adversarial example generated following the algorithm in Section. 5.4.

5.7.3 Robustness Analysis

To mount our simulated over-the-air attacks, we consider a challenging setting that the exact configuration of the room in which the attack will be performed is unknown. Instead, we are only aware of the distribution from which the room configuration will be drawn. First, we generate 1000 random room configurations sampled from the distribution as the training room set. The test room set includes another 100 random room configurations sampled from the same distribution. Adversarial examples are created to attack the Lingvo ASR system when played in the simulated test rooms. We randomly choose 100 audio examples from LibriSpeech dataset to perform this robustness test.

As shown in Table 5.3, when fed non-adversarial audio played in simulated test rooms, the WER of the Lingvo ASR degrades to 15.42% which suggests some robustness to reverberation. In contrast, the success rate of adversarial examples in [11] and our imperceptible adversarial examples in Section 5.4 are 0% in this setting. The success rate of our robust adversarial examples generated based on the algorithm in Section 5.5 is over 60%, and the WER is smaller

Table 5.3. Sentence-level accuracy and WER for 100 clean and adversarially perturbed examples, fed with over-the-air simulation into the Lingvo model. The ground truth for "clean" inputs is the original transcription while the ground truth is the targeted transcription for the adversarial inputs. The perturbation is bounded by $\|\delta\| < \varepsilon_r^* + \Delta$.

Input	Clean	Robust ($\Delta = 300$)	Robust ($\Delta = 400$)	Imperceptible & Robust
Accuracy (%)	31.37	62.96	64.64	49.65
WER (%)	15.42	14.45	13.83	22.98

than that of the clean audio. Both the success rate and the WER demonstrate that our constructed adversarial examples remain effective when played in the highly-realistic simulated environment.

In addition, the robustness of the constructed adversarial examples can be improved further at the cost of increased perceptibility. As presented in Table 5.3, when we increase the max-norm bound of the amplitude of the adversarial perturbation $\varepsilon_r^{**} = \varepsilon_r^* + \Delta$ (Δ is increased from 300 to 400), both the success rate and WER are improved correspondingly. Since our final objective is to generate imperceptible and robust adversarial examples that can be played over-the-air in the physical world, we limit the max-norm bound of the perturbation to be in a relatively small range to avoid a huge distortion toward the clean audio.

To construct imperceptible as well as robust adversarial examples, we start from the robust attack ($\Delta = 300$) and finetune it with the imperceptibility loss. In our experiments, we observe that 81% of the robust adversarial examples [3] can be further improved to be much less perceptible while still retaining high robustness (around 50% success rate and 22.98% WER).

Qualitative Human Study

We run identical experiments (as described earlier) on the robust and robust & imperceptible adversarial examples.

In **experiment 1**, where we ask evaluators if there is any noise, only 6% heard any noise

[3]The other 19% adversarial examples lose the robustness because they cannot successfully attack the ASR system in 8 randomly chosen training rooms in any iteration during optimization.

on the clean audio, compared to 100% on the robust (but perceptible) adversarial examples and 83% on the robust and imperceptible adversarial examples. [4]

In **experiment 2**, where we ask evaluators to identify the original audio, comparing clean to robust adversarial examples the evaluator correctly identified the original audio 97% of the time versus 89% when comparing the clean audio to the imperceptible and robust adversarial examples.

Finally, in **experiment 3**, where we ask evaluators if the audio is identical, the baseline clean audio was judged different 95% of the time when compared to the robust adversarial examples, and the clean audio was judged different 71% of the time when compared to the imperceptible and robust adversarial examples.

In all cases, the imperceptible and robust adversarial examples are statistically significantly less perceptible than just the robust adversarial examples, but also statistically significantly more perceptible than the clean audio. Directly comparing the imperceptible and robust adversarial examples to the robust examples, evaluators believed the imperceptible examples had less distortion 91% of the time.

Clearly the adversarial examples that are robust are significantly easier to distinguish from clean audio, even when we apply the masking threshold. However, this result is consistent with work on adversarial examples on images, where completely imperceptible physical-world adversarial examples have not been successfully constructed. On images, physical attacks require over $16\times$ as much distortion to be effective on the physical world (see, for example, Figure 4 of [33]).

5.8 Conclusion

In this chapter, we successfully construct imperceptible adversarial examples (verified by a human study) for automatic speech recognition based on the psychoacoustic principle

[4]Evaluators stated they heard noise on clean examples $3\times$ less often compared to the baseline in the prior study. We believe this is due to the fact that when primed with examples which are obviously different, the baseline becomes more easily distinguishable.

of auditory masking, while retaining 100% targeted success rate on arbitrary full-sentence targets. Simultaneously, we also make progress towards developing robust adversarial examples that remain effective after being played over-the-air (processed by random room environment simulators), increasing the practicality of actual real-world attacks using adversarial examples targeting ASR systems.

We believe that future work is still required: our robust adversarial examples do not play fully over-the-air, despite working in simulated room environments. Resolving this difficulty while maintaining a high targeted success rate is necessary for demonstrating a practical security concern.

As a final contribution of potentially independent interest, this work demonstrates how one might go about constructing adversarial examples for non-ℓ_p-based metrics. Especially on images, nearly all adversarial example research has focused on this highly-limited distance measure. Devoting effort to identifying different methods that humans use to assess similarity, and generating adversarial examples exploiting those metrics, is an important research effort we hope future work will explore.

This chapter is based on the material as it appears in the Proceedings of the International Conference on Machine Learning (ICML19) (Yao Qin, Nicholas Carlini, Ian Goodfellow, Garrison Cottrell and Colin Raffel, "Imperceptible, Robust and Targeted Adversarial Examples for Automatic Speech Recognition"). The dissertation author was the primary investigator and author of this paper.

Chapter 6

Conclusion

In this thesis, we mainly focused on adversarial defenses in the image domain and adversarial attacks in the audio domain. In the image domain, we proposed a class-conditional reconstruction-based detection method that does not rely on a specific predefined adversarial attack. To specifically attack the detection method, we design a new defense-aware attack in which the adversary not only optimizes the classification loss but also takes the reconstruction error into consideration. We have shown that our detection method together with a Capsule network can detect standard and defense-aware attacks very well. To further diagnose the adversarial examples against Capsule networks and convolutional based networks, we qualitatively showed that the success of the reconstructive attack was proportional to the visual similarity between the target class and the source class for the CapsNet, which is not the case for convolutional based networks. This indicates that the Capsule model relies on visual features similar to those used by humans and has the potential to help us have a better understanding of how neural networks work.

In addition, we propose a deflecting model which presents a new notion of deflecting adversarial examples, which is a step towards ending the cycle between stronger defenses and attacks in the adversarial game. To deflect adversarial examples is to pressure the attacker to make the adversarial input resemble the target class and therefore, the adversarial input stops being adversarial. The existence of deflected adversarial examples shows that the most commonly used ℓ_p norm constraint to the adversarial perturbation does not ensure imperceptibility because the deflected adversarial examples are within a standard small ℓ_p norm distance to the original image but are classified by humans as a different class.

In the audio domain, we successfully construct imperceptible adversarial examples (verified by a human study) for automatic speech recognition based on the psychoacoustic principle of auditory masking, while retaining a 100% targeted success rate on arbitrary full-sentence targets. Simultaneously, we also make progress towards developing robust adversarial examples that remain effective after being played over-the-air (processed by random room environment simulators), increasing the practicality of actual real-world attacks using adversarial examples targeting ASR systems.

We believe that future work on adversarial examples is still required and below we mainly discuss three promosing directions:

- **Generalization**

  Most existing research work on adversarial examples has focused on the image domain. The properties of image adversaial examples have not yet been tested to hold true in other domains, *e.g.*, audio or text domains. For example, transferability, a fact that frequently adversarial examples designed for one network will work against another network, is a property that significantly complicates finding robust defenses [49]. This should be further studied to see if it can be successfully generalized to other domains. In addition, we must seek methods that can combine domain-specific knowledge and the lessons learned from image adversarial examples. For instance, when the input is discrete (text domain) rather than continuous (image and audio domain), how to define, design and defend against adversarial examples?

- **Robustness**

  Designing more robust adversarial examples and more robust defense models is critical for real-world applications. For example, our imperceptible adversarial examples do not work while playing over-the-air. To construct robust adversarial examples, we can incorporate the real room impulses, e.g., the BUT dataset [65], to help create adversarial examples that remain effective while playing over-the-air. Resolving this difficulty while maintaining a high target success rate is necessary for demonstrating a practical security concern. In addition, our detection mechanism relies on a similarity metric (ℓ_2 reconstruction error) between the reconstruction and the input. This metric is required both during training in order to train the reconstruction network and during test time in order to flag adversarial examples. In the datasets we have evaluated, the distance between examples roughly correlates with semantic similarity. This is not the case, however, for images in more complex datasets such as ImageNet [15], in which two images may be similar in terms of semantic content but nevertheless have significant ℓ_2 distance. This issue will need to be resolved for this method to scale up to more complex and practical datasets.

- **Understanding**

  In this work, we have shown that Capsule networks rely on features that are more aligned to human perception compared to convolutional based networks. Devoting efforts to study the adversarial examples against these two different network architectures and find out the underlying explanation for the superior performance is an interesting future direction. We believe that it could also help us have a better understanding of how neural networks work. In addition, the deflected adversarial examples have the potential to fall into the same distribution as the clean data. Therefore, constructing a dataset of deflected adversarial examples is an interesting direction that we hope future work could explore to see if it can serve as a data augmentation method and benefit few-shot learning tasks.

Bibliography

[1] Jont B Allen and David A Berkley. Image method for efficiently simulating small-room acoustics. *The Journal of the Acoustical Society of America*, 65(4):943–950, 1979.

[2] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *arXiv preprint arXiv:1802.00420*, 2018.

[3] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples. In *ICML*, 2018.

[4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[5] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 387–402. Springer, 2013.

[6] Jacob Buckman, Aurko Roy, Colin Raffel, and Ian Goodfellow. Thermometer encoding: One hot way to resist adversarial examples. 2018.

[7] Nicholas Carlini, Anish Athalye, Nicolas Papernot, Wieland Brendel, Jonas Rauber, Dimitris Tsipras, Ian Goodfellow, and Aleksander Madry. On evaluating adversarial robustness. *arXiv preprint arXiv:1902.06705*, 2019.

[8] Nicholas Carlini, Pratyush Mishra, Tavish Vaidya, Yuankai Zhang, Micah Sherr, Clay Shields, David Wagner, and Wenchao Zhou. Hidden voice commands. In *USENIX Security Symposium*, pages 513–530, 2016.

[9] Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 3–14. ACM, 2017.

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

[11] Nicholas Carlini and David A. Wagner. Audio adversarial examples: Targeted attacks on speech-to-text. *2018 IEEE Security and Privacy Workshops (SPW)*, pages 1–7, 2018.

[12] William Chan, Navdeep Jaitly, Quoc Le, and Oriol Vinyals. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 4960–4964. IEEE, 2016.

[13] Pin-Yu Chen, Yash Sharma, Huan Zhang, Jinfeng Yi, and Cho-Jui Hsieh. Ead: elastic-net attacks to deep neural networks via adversarial examples. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

[14] Moustapha Cisse, Yossi Adi, Natalia Neverova, and Joseph Keshet. Houdini: Fooling deep structured prediction models. *arXiv preprint arXiv:1707.05373*, 2017.

[15] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[16] Reuben Feinman, Ryan R. Curtin, Saurabh Shintre, and Andrew B. Gardner. Detecting adversarial samples from artifacts. *arXiv preprint arXiv:1703.00410*, 2017.

[17] Justin Gilmer, Ryan P. Adams, Ian Goodfellow, David Andersen, and George E. Dahl. Motivating the rules of the game for adversarial example research. *arXiv preprint arXiv:1807.06732*, 2018.

[18] Justin Gilmer, Luke Metz, Fartash Faghri, Samuel S. Schoenholz, Maithra Raghu, Martin Wattenberg, and Ian Goodfellow. Adversarial spheres. *arXiv preprint arXiv:1801.02774*, 2018.

[19] Yuan Gong and Christian Poellabauer. Crafting adversarial examples for speech paralinguistics applications. *arXiv preprint arXiv:1711.03280*, 2017.

[20] Zhitao Gong, Wenlu Wang, and Wei-Shinn Ku. Adversarial and clean data are not twins. *arXiv preprint arXiv:1704.04960*, 2017.

[21] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[22] Kathrin Grosse, Praveen Manoharan, Nicolas Papernot, Michael Backes, and Patrick McDaniel. On the (statistical) detection of adversarial examples. *arXiv preprint arXiv:1702.06280*, 2017.

[23] Dan Hendrycks and Kevin Gimpel. Early methods for detecting adversarial images. *arXiv preprint arXiv:1608.00530*, 2016.

[24] Geoffrey E. Hinton, Sara Sabour, and Nicholas Frosst. Matrix capsules with EM routing. 2018.

[25] Hossein Hosseini, Sreeram Kannan, and Radha Poovendran. Are odds really odd? bypassing statistical detection of adversarial examples. *arXiv preprint arXiv:1907.12138*, 2019.

[26] Sandy Huang, Nicolas Papernot, Ian Goodfellow, Yan Duan, and Pieter Abbeel. Adversarial attacks on neural network policies. *arXiv preprint arXiv:1702.02284*, 2017.

[27] Andrew Ilyas, Ajil Jalal, Eirini Asteri, Constantinos Daskalakis, and Alexandros G. Dimakis. The robust manifold defense: Adversarial training using generative models. *arXiv preprint arXiv:1712.09196*, 2017.

[28] Saumya Jetley, Nicholas A. Lord, and Philip HS Torr. With friends like these, who needs adversaries? *arXiv preprint arXiv:1807.04200*, 2018.

[29] Robin Jia and Percy Liang. Adversarial examples for evaluating reading comprehension systems. *arXiv preprint arXiv:1707.07328*, 2017.

[30] Shreya Khare, Rahul Aralikatte, and Senthil Mani. Adversarial black-box attacks for automatic speech recognition systems using multi-objective genetic optimization. *arXiv preprint arXiv:1811.01312*, 2018.

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

[32] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[33] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.

[34] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.

[35] Alexey Kurakin, Ian Goodfellow, Samy Bengio, Yinpeng Dong, Fangzhou Liao, Ming Liang, Tianyu Pang, Jun Zhu, Xiaolin Hu, Cihang Xie, et al. Adversarial attacks and defences competition. *arXiv preprint arXiv:1804.00097*, 2018.

[36] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[37] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *Advances in Neural Information Processing Systems*, pages 7167–7177, 2018.

[38] Xin Li and Fuxin Li. Adversarial examples detection in deep networks with convolutional filter statistics. In *Proceedings of the IEEE International Conference on Computer Vision*, 2017.

[39] Yiqing Lin and Waleed H Abdulla. Principles of psychoacoustics. In *Audio Watermark*, pages 15–49. Springer, 2015.

[40] Xingjun Ma, Bo Li, Yisen Wang, Sarah M Erfani, Sudanthi Wijewickrema, Grant Schoenebeck, Dawn Song, Michael E Houle, and James Bailey. Characterizing adversarial subspaces using local intrinsic dimensionality. *arXiv preprint arXiv:1801.02613*, 2018.

[41] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[42] Dongyu Meng and Hao Chen. Magnet: a two-pronged defense against adversarial examples. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 135–147. ACM, 2017.

[43] Jan Hendrik Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. On detecting adversarial perturbations. *arXiv preprint arXiv:1702.04267*, 2017.

[44] Felix Michels, Tobias Uelwer, Eric Upschulte, and Stefan Harmeling. On the vulnerability of capsule networks to adversarial attacks. *arXiv preprint arXiv:1906.03612*, 2019.

[45] Joan L Mitchell. Introduction to digital audio coding and standards. *Journal of Electronic Imaging*, 13(2):399, 2004.

[46] Norman Mu and Justin Gilmer. Mnist-c: A robustness benchmark for computer vision. In *ICML 2019 Workshop on Uncertainty and Robustness in Deep Learning*, 2019.

[47] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, page 5, 2011.

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

[49] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*, 2016.

[50] Yao Qin, Nicholas Frosst, Sara Sabour, Colin Raffel, Garrison Cottrell, and Geoffrey Hinton. Detecting and diagnosing adversarial images with class-conditional capsule reconstructions. *arXiv preprint arXiv:1907.02957*, 2019.

[51] Jathushan Rajasegaran, Vinoj Jayasundara, Sandaru Jayasekara, Hirunima Jayasekara, Suranga Seneviratne, and Ranga Rodrigo. Deepcaps: Going deeper with capsule networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10725–10733, 2019.

[52] Kevin Roth, Yannic Kilcher, and Thomas Hofmann. The odds are odd: A statistical test for detecting adversarial examples. *arXiv preprint arXiv:1902.04818*, 2019.

[53] Sara Sabour, Yanshuai Cao, Fartash Faghri, and David J. Fleet. Adversarial manipulation of deep representations. *arXiv preprint arXiv:1511.05122*, 2015.

[54] Sara Sabour, Nicholas Frosst, and Geoffrey E. Hinton. Dynamic routing between capsules. In *Advances in Neural Information Processing Systems*, pages 3856–3866, 2017.

[55] Pouya Samangouei, Maya Kabkab, and Rama Chellappa. Defense-gan: Protecting classifiers against adversarial attacks using generative models. *arXiv preprint arXiv:1805.06605*, 2018.

[56] Robin Scheibler, Eric Bezzam, and Ivan Dokmanić. Pyroomacoustics: A python package for audio room simulation and array processing algorithms. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 351–355. IEEE, 2018.

[57] Lea Schönherr, Katharina Kohls, Steffen Zeiler, Thorsten Holz, and Dorothea Kolossa. Adversarial attacks against automatic speech recognition systems via psychoacoustic hiding. *arXiv preprint arXiv:1808.05665*, 2018.

[58] Lukas Schott, Jonas Rauber, Wieland Brendel, and Matthias Bethge. Robust perception through analysis by synthesis. *arXiv preprint arXiv:1805.09190*, 2018.

[59] Jonathan Shen, Patrick Nguyen, Yonghui Wu, Zhifeng Chen, Mia X Chen, Ye Jia, Anjuli Kannan, Tara Sainath, Yuan Cao, Chung-Cheng Chiu, et al. Lingvo: a modular and scalable framework for sequence-to-sequence modeling. *arXiv preprint arXiv:1902.08295*, 2019.

[60] Liwei Song and Prateek Mittal. Inaudible voice commands. *arXiv preprint arXiv:1708.07238*, 2017.

[61] Yang Song, Taesup Kim, Sebastian Nowozin, Stefano Ermon, and Nate Kushman. Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. *arXiv preprint arXiv:1710.10766*, 2017.

[62] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112, 2014.

[63] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[64] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[65] Igor Szoke, Miroslav Skacel, Ladislav Mosner, Jakub Paliesek, Jan Cernocky, et al. Building and evaluation of a real room impulse response dataset. *arXiv preprint arXiv:1811.06795*, 2018.

[66] Rohan Taori, Amog Kamsetty, Brenton Chu, and Nikita Vemuri. Targeted adversarial examples for black box audio systems. *arXiv preprint arXiv:1805.07820*, 2018.

[67] Lucas Theis, Aäron van den Oord, and Matthias Bethge. A note on the evaluation of generative models. *arXiv preprint arXiv:1511.01844*, 2015.

[68] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[69] Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3485–3492. IEEE, 2010.

[70] Hiromu Yakura and Jun Sakuma. Robust audio adversarial example for a physical attack. *arXiv preprint arXiv:1810.11793*, 2018.

[71] Yuzhe Yang, Guo Zhang, Dina Katabi, and Zhi Xu. Me-net: Towards effective adversarial robustness with matrix estimation. *arXiv preprint arXiv:1905.11971*, 2019.

[72] Xuejing Yuan, Yuxuan Chen, Yue Zhao, Yunhui Long, Xiaokang Liu, Kai Chen, Shengzhi Zhang, Heqing Huang, Xiaofeng Wang, and Carl A Gunter. Commandersong: A systematic approach for practical adversarial voice recognition. *arXiv preprint arXiv:1801.08535*, 2018.

[73] Guoming Zhang, Chen Yan, Xiaoyu Ji, Tianchen Zhang, Taimin Zhang, and Wenyuan Xu. Dolphinattack: Inaudible voice commands. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 103–117. ACM, 2017.

[74] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 586–595, 2018.