NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

REDUCING ADVERSARIAL FAILURES IN NEURAL NETWORKS USING "NONE OF THE ABOVE" CLASS PRIORS

by

Alexi N. Mendolia

December 2023

Thesis Advisor:	Patrick McClure
Second Reader:	Armon C. Barton

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** December 2023	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE
REDUCING ADVERSARIAL FAILURES IN NEURAL NETWORKS USING "NONE OF THE ABOVE" CLASS PRIORS

5. FUNDING NUMBERS

6. AUTHOR(S) Alexi N. Mendolia

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
N/A

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release. Distribution is unlimited.

12b. DISTRIBUTION CODE
A

13. ABSTRACT (maximum 200 words)

While machine learning presents an opportunity for increased automation in systems, machine-learning models are also subject to adversarial attacks. This thesis builds on previous methods for securing against adversarial examples by training a model with a "None of the Above" (NOTA) class. While classification models force categorization into one of a fixed number of classes, NOTA models implement an additional class allowing for the notion that some inputs will not "match" any of the given classes. While previous methods are largely successful in providing state of the art adversarial robustness, they are less successful against some of the more complex adversarial attack vectors. This thesis aims to increase adversarial robustness through a prior that biases predictions to be the NOTA class. We conduct a validation grid search to find the prior probability for a NOTA class over the CIFAR-10 image dataset that best decreases adversarial success. Through this work, we are able to provide a proof-of-concept that the addition of a NOTA-biased prior can decrease the adversarial success of some of the more complex evasion attacks. As the DOD moves to increase its use of machine learning models, these results will be increasingly important towards building models with adequate security.

14. SUBJECT TERMS machine learning classification, system safety, deep learning, artificial neural networks, adversarial examples, defense	**15. NUMBER OF PAGES** 57
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

REDUCING ADVERSARIAL FAILURES IN NEURAL NETWORKS USING "NONE OF THE ABOVE" CLASS PRIORS

Alexi N. Mendolia
Lieutenant, United States Navy
BS, United States Naval Academy, 2018

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
December 2023

Approved by: Patrick McClure
Advisor

Armon C. Barton
Second Reader

Gurminder Singh
Chair, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

While machine learning presents an opportunity for increased automation in systems, machine-learning models are also subject to adversarial attacks. This thesis builds on previous methods for securing against adversarial examples by training a model with a "None of the Above" (NOTA) class. While classification models force categorization into one of a fixed number of classes, NOTA models implement an additional class allowing for the notion that some inputs will not "match" any of the given classes. While previous methods are largely successful in providing state of the art adversarial robustness, they are less successful against some of the more complex adversarial attack vectors. This thesis aims to increase adversarial robustness through a prior that biases predictions to be the NOTA class. We conduct a validation grid search to find the prior probability for a NOTA class over the CIFAR-10 image dataset that best decreases adversarial success. Through this work, we are able to provide a proof-of-concept that the addition of a NOTA-biased prior can decrease the adversarial success of some of the more complex evasion attacks. As the DOD moves to increase its use of machine learning models, these results will be increasingly important towards building models with adequate security.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

1 Introduction **1**

1.1 Motivation 1

1.2 Problem Statement. 2

1.3 Research Questions 2

1.4 Thesis Organization 3

2 Background **5**

2.1 Neural Networks 5

2.2 Vulnerabilities in Neural Networks 5

2.3 Adversarial Defenses 8

2.4 NOTA Defenses and Attacks 9

2.5 Chapter Summary 13

3 Methods **15**

3.1 Model Methods 15

3.2 Prior Regularization 17

3.3 Model Architecture 18

3.4 Attack Vectors 20

3.5 Chapter Summary 26

4 Experiments **27**

4.1 Dataset Usage 27

4.2 Experiment Design 28

4.3 Performance Metrics 28

4.4 Results Documentation 28

4.5 Accuracy 30

4.6 Chapter Summary 31

5 Conclusion and Future Work **33**

5.1 Contributions 33
5.2 Future Work 33
5.3 Conclusion 34

List of References **35**

Initial Distribution List **39**

List of Figures

Figure 2.1 Basic design of a neural network 5

Figure 2.2 Example of an adversarial image. 6

Figure 2.3 Illustration of the padding class used in PadNet. 10

Figure 3.1 Illustration of Dropout. 16

Figure 3.2 Model Architecture printout for CIFAR-10 Models. 19

Figure 3.3 L2 Attack applied to the CIFAR-10 dataset 21

Figure 4.1 CIFAR-10 dataset. 27

Figure 4.2 ASR for all attacks on the CIFAR-10 dataset 30

THIS PAGE INTENTIONALLY LEFT BLANK

List of Tables

Table 4.1 Validation grid search results for μ_{NOTA} and prior weight for the CIFAR-10 dataset 28

Table 4.2 Attack success rate comparison for all models. 29

Table 4.3 Test Set Model Accuracy for all models 31

Table 4.4 Regular Class Accuracy (RCA) for all models 31

THIS PAGE INTENTIONALLY LEFT BLANK

List of Acronyms and Abbreviations

AI	Artificial Intelligence
ALP	Adversarial Logit Pairing
APGD	Auto Projected Gradient Descent
ASR	Adversarial Success Rate
C10	CIFAR-10
CNO	Chief of Naval Operations
CW	Carlini-Wagner
DoD	Department of Defense
DLR	Difference of Logits Ratio
DNN	Deep Neural Network
GSMA	Gradient-based Saliency-Map Attack
MAP	Maximum A Posteriori
ML	Machine Learning
MRNA	Multi-Modal NOTA-Aware Adversarial Mixup
NAM	Naive Adversarial Mixup
NN	Neural Network
NOTA	None of the Above
PGD	Projected Gradient Descent
RCA	Regular Class Accuracy

ReLU	Rectified Linear Unit
SRNA	Simple Reflexive NOTA-Aware Adversarial Mixup
TA	Training Accuracy
TI	Tiny ImageNet
USN	U.S. Navy
VA	Validation Accuracy
WD	Weight Decay
WRN	Wide Residual Network

Acknowledgments

First and foremost, I would like to thank my advisor, Dr. Patrick “Bayesian” McClure, for his wisdom, support, and boundless patience. Thank you for guiding me through many classes and this thesis process with sage advice and enthusiasm.

Next, I would like to thank CDR Ed Jatho. Thank you for allowing me to be a part of your team, and answering my many questions along the way. Your future students are lucky to have you.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1:
Introduction

Artificial Intelligence (AI) and Machine Learning (ML) are enablers; through the use of automation, we can more rapidly produce results while reducing the burden of work on humans. We have seen rapid growth in the focus on and use of automation in almost all industries, including the Department of Defense (DoD). Biden and Harris's 2022 National Security Strategy [1] specifically calls out the need for investment in trusted AI, while the Chief of Naval Operations (CNO)'s 2022 Navigation Plan [2] highlights how the U.S. Navy (USN) must leverage AI to accelerate decision-making and create strategic advantage. Through the judicious use of AI and ML models, we may imagine how the USN may be able to shift work to machines and increase productivity, affording Sailors the time and space to operate and think at a more strategic level.

1.1 Motivation

Automation provides DoD and USN leaders and decision-makers with many opportunities; however, as we implement automation into defense systems or other safety-critical systems, we must be aware of, and account for, the vulnerabilities in our systems and how we may defend them. Failures of automation exist and can have catastrophic or deadly effects. In 2016, the failure of a self-driving car to recognize a truck in oncoming traffic led to fatality of the driver [3]. We have also seen instances of racial biases in automated systems due to failures in training [3]; during the COVID-19 pandemic, automated health care management algorithms were trained on insufficient data that reflected overall inequities in the healthcare system, leading to instances of discrimination in the system's decision-making [4].

While the previous examples highlight the fragility of automated systems, we must also consider the deliberate exploitation of automated systems by adversaries. Adversarial attacks include performing slight modifications to a Deep Neural Network (DNN) model input in order to force an incorrect prediction. For datasets of images, there are many examples of these adversarial attacks, with perturbations indistinguishable to the human eye that may drastically alter the output of a classifier [5]. This notion of deceiving a classifier has

real-world applications; researchers at the Massachusetts Institute of Technology (MIT) and Northeastern University have used the concept of adversarial images to create tee-shirts with certain images on them such that they defeat intelligent person-detection systems [6]. We can imagine the detrimental implications of this type of deception against safety or security-critical systems if these systems are left undefended.

Certainly, these examples make a case for prudence and emphasize the need for trust in AI systems. In their essay on the vulnerabilities of AI and its applications to the military, Jatho and Kroll [7] highlight some of the challenges of AI and ML; however, they suggest that the use of adversarial ML is not an insurmountable problem. They liken adversarial attack to a modern-day camouflage, in that we must prepare for its use and build systems to be robust against it [7]. As we move across an age of increasing automation, decision-makers must assume that adversarial tools will be used and, therefore, must take caution to adequately secure their systems.

1.2 Problem Statement

While AI and ML are certainly necessary as the DoD moves toward a more technologically-advanced and capable military, we cannot operate automated systems without cautious work to secure vulnerabilities and defend against unknown attack vectors. This thesis will build from a pre-existing adversarial defense model built by Jatho [8]. This model adds an additional class (what he refers to as a "None of the Above," or "NOTA," class) with the idea of forcing adversarial examples into this class, rather than producing an incorrect class label. This thesis modifies his approach by using a parameter prior that biases predictions toward NOTA to reduce the adversarial success rate of attacks.

1.3 Research Questions

We hypothesize that a model prior that corresponds to high NOTA probability will lead to improved adversarial robustness, since the vast majority of input space likely corresponds to NOTA. Research questions being pursued in this work include:

1. What types of NOTA priors, if any, lead to improved adversarial robustness?
2. How effective are NOTA priors at defending against different types of adversarial attacks?

We have scoped this experiment to include the assessment of four models:

1. Jatho's Undefended Model [8]
2. NOTA-Prior Defended Model
3. Jatho's NOTA Model trained with adversarial examples [8]
4. NOTA with adversarial examples and NOTA-Prior Model

These models will be tested on the CIFAR-10 image dataset. Further, this project will be tested against three well-known categories of adversarial attacks: Carlini-Wagner attacks, AutoPGD, and AutoAttack.

1.4 Thesis Organization

We have introduced the motivation for this thesis and outlined the general research questions that are examined through this work. Chapter 2 gives background and defines adversarial attacks and defenses. This discussion includes a summary of prior work in adversarial defenses that inspire the work in this thesis. Chapter 3 defines our methods, specifically showing our model architecture and how we derived an equation for a NOTA-biased prior. It also summarizes all the attack vectors used. Chapter 4 summarizes our experiments, giving dataset considerations and analysis of results. We summarize our work and outline potential future work in Chapter 5.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 2:
Background

2.1 Neural Networks

A Neural Network (NN) is a function that computes an output from an input using one or more layers of artificial neurons [9], as shown in Figure 2.1. Through the Universal Approximation Theorem, we understand NNs to be powerful function approximators [10], [11]. Due to this characteristic, NNs can be used across a wide variety of applications.

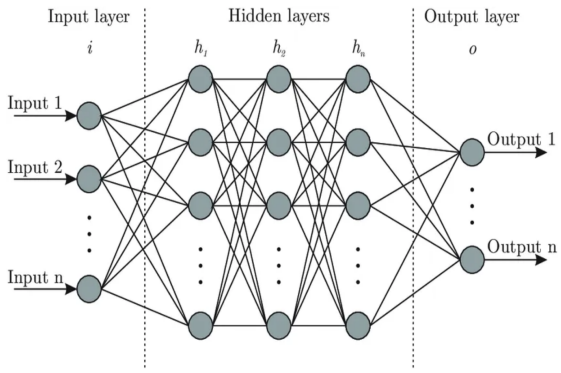

Figure 2.1. Basic design of a neural network. Source: [12]

2.2 Vulnerabilities in Neural Networks

As a widely used method for machine learning problems, NNs are also subject to exploitation. In [13], Chakraborty et al. defined three types of attacks against NNs: *Exploratory*, *Poisoning*, and *Evasion* attacks.

- Exploratory Attacks: The goal of exploratory attacks is to gather information about a learning algorithm without causing any changes in input or output of the model [14].
- Poisoning Attacks: These attacks inject malicious samples during the training process to compromise learning [13], [15].

- Evasion Attacks: The most common case of adversarial attacks, these attacks generally use gradient-based methods to manipulate a network's input toward a desired result such that they can "evade" the correct class label [5], [13].

In continuation of the work of Jatho [8], this thesis will focus solely on evasion attacks. Examples created using these attacks will be heretofore generalized as "adversarial examples" or "adversarial attacks."

2.2.1 Adversarial Example Definition

Discovered by Szegedy et al. [16] in 2015, adversarial attacks exploit inherent properties of NNs, ultimately forcing the network to deliver an incorrect output. Early work in this field focused on image classification and adversarial images for NNs. Figure 2.2 is an example of an adversarial image; Goodfellow et al. [17] altered the image of a panda (left), resulting in the change of the classifier's 57% confident prediction of 'panda' to the classifiers 99% confident, albeit incorrect, prediction of 'gibbon' (right). While the image on the left and the right look exactly the same to the human eye, the perturbation was enough to produce drastically different class labels, and perhaps more importantly, increased confidence toward the incorrect class label [17].

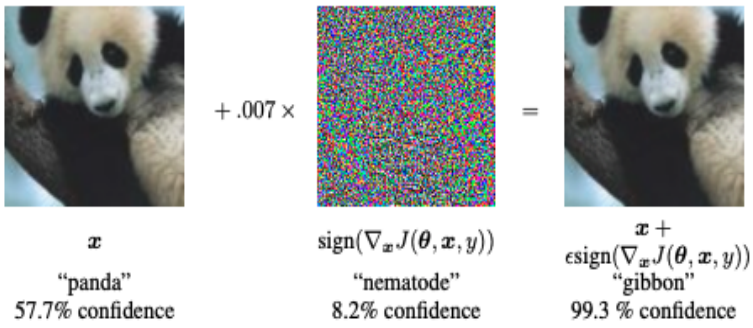

Figure 2.2. Example of an adversarial image. Source: [17]. This figure illustrates the imperceptible differences between the benign image on the left, and the adversarial image on the right.

The core reason why a NN is susceptible to these types of attacks is still not entirely clear. While original research by Szegedy et al. [16] asserted that the instability of these NNs was caused by discontinuity of input-output mappings learned by DNNs, later research conducted by Goodfellow et al. [17] asserted that the linearity of NNs makes them vulnerable to these attacks. Goodfellow et al. [17] explained adversarial examples as "a property of high-dimensional dot products" wherein the adversarial perturbation causes linear growth in the activation.

In their original work, Szegedy et al. [16] also found that the same adversarial examples will cause misclassification of the same inputs when used on separately-trained NNs, oftentimes with these separate NNs agreeing on class labels for the same examples [17]. This generalization property supports a linear view of adversarial examples, explained by Goodfellow et al. [17] as a result of "adversarial perturbations being highly aligned with the weight vectors of a model, and different models learning similar functions when trained to perform the same task."

2.2.2 Evasion Attacks

The methods by which to generate adversarial examples is another expanding area of research; the attacks used in this paper and their means of generating adversarial images are discussed in depth in Section 3.4. Specifically, we utilize the Carlini-Wagner attacks, Auto Projected Gradient Descent (APGD), and AutoAttack, which we consider to be some of the most commonly used evasion attacks.

Evasion attacks can be described using a few general categories, which will be referred to throughout this thesis.

White and Black Box Attacks: White box attacks have full access and knowledge of the model and its architecture, while black box attacks do not. We may consider black box attacks more difficult to perform, requiring generalizability using incomplete knowledge.

Untargeted and Targeted Attacks: Untargeted attacks aim to output incorrect predictions, but do not specify which predictions to make. Targeted attacks aim to output a particular, albeit incorrect, prediction.

2.3 Adversarial Defenses

A variety of mechanisms to defend against evasion attacks have been suggested since their discovery in 2015; however, this paper will focus specifically on the concept of adversarial training as the precursor to Jatho's novel approach [8]. We give examples of adversarial training, and break out adversarial logit pairing and generative methods as its sub-categories, before discussing the "NOTA" approach used in [8].

2.3.1 Adversarial Training

Goodfellow et al. [17] first introduced the concept of adversarial training, defining it as a data augmentation method that "uses inputs that are unlikely to occur naturally but that expose flaws in the ways that the model conceptualizes its decision function." In adversarial training, adversarial images are generated and added to the training data assigning these examples the true class label for the corresponding inputs used to generate the adversarial examples. Using adversarial training, the NN could ideally train itself to correctly recognize perturbed inputs as one of the already existing classes. In this way, the NN would be theoretically more robust against "outlier" inputs. For Figure 2.2, this means that a NN would be trained to recognize the perturbed image on the right to be part of the "panda" class rather than the "gibbon" class.

Research has since continued in this area, now attempting to build NNs to defeat more robust attacks. One such study was conducted by Madry et al. [18], by training a DNN classifier against the strongest adversary (at the time, Projected Gradient Descent (PGD)). Their approach was based on the idea of optimizing a saddle-point problem, and on building a network of a higher capacity than those used for clean examples only. While their MNIST dataset network achieved much higher accuracy than their CIFAR-10 dataset network, they assert that their findings suggest that adversarially-robust networks are achievable [18].

While adversarial training generally improves adversarial robustness, it is not adaptive to unknown attacks. A model must be trained with adversarial examples produced by all known attacks to be robust, a computationally expensive process. Because of this, adversarially-trained models will not be robust to previously unknown attacks [19]. Further, previous research [20] suggests adversarial training causes model overfitting for adversarial features, which reduces the generalizability of the model.

Adversarial Logit Pairing

Kannan et al. [21] introduced Adversarial Logit Pairing (ALP) as an enhancement to previously used Adversarial Training techniques. The authors describe that logit pairing "encourages the logits for two pairs of examples to be similar" [21]. For adversarial logit pairing, these examples are a clean image and a counterpart image that has been created as an adversarial image. The authors' contribution of an additional regularization term allows the model to define levels of similarity between clean and adversarial images, rather than solely assigning both the clean and adversarial images the same class label [21]. This approach allowed the authors to increase accuracy seen against white box and black box attacks [21]; however, ALP suffers many of the same pitfalls as adversarial training. Like adversarial training, ALP is a computationally expensive process that is not adaptive to new attack vectors [19].

Generative (Diffusion) Methods

The research area of diffusion models has grown concurrently with the area of adversarial training; in order to create adversarial examples for models to train on, various uses of generative models have been suggested. Having evolved since 2015, the most recent research in using diffusion models to improve adversarial training combines a variety of methods to build robust models without having to train on external datasets [22]. While their work achieves state of the art robust accuracy, the authors suggest that learning efficiency is a task for further development [22].

2.4 NOTA Defenses and Attacks

The NOTA class, also called a "padding class," was built to support the addition of an "artificial padding class" into the classification space, reasoning that the classifier will more reasonably distinguish between true and adversarial datapoints [15]. The idea of this class is similar to adversarial training; however, a true class label is not generated. Adversarial samples are generated and labelled as NOTA.

Jatho [8] notes, "with carefully crafted NOTA augmentation examples, we can continuously 'seed' the benign data-point-sparse space between data-dense regions of a classifier's input space throughout training and leverage the DNN to classify this vast space as NOTA." The

idea of the NOTA class is that a DNN could clearly define the "boundaries between and adjacent to classes," and that all other input space would be generalized as NOTA [8].

We explain the conceptual foundation for Jatho's [8] NOTA class, Barton's [19] boundary padding method, and describe how the NOTA apporach was adapted to create Naive Adversarial Mixup (NAM), an algorithm for generating and seeding NOTA examples into classification space.

2.4.1 Boundary Padding

The concept of a NOTA class was inspired by Barton's concept of "Defensive Padding" [19]. Built to remedy the issues of computationally expensive adversarial training, Barton's work "PadNet" adds in a "padding class" between the existing input examples from different classes [19].

This concept is illustrated in Figure 2.3 [19]. Given prior research suggesting adversarial examples exist near decision boundaries [23], padding class examples were built to be near the decision boundaries between classes [19].

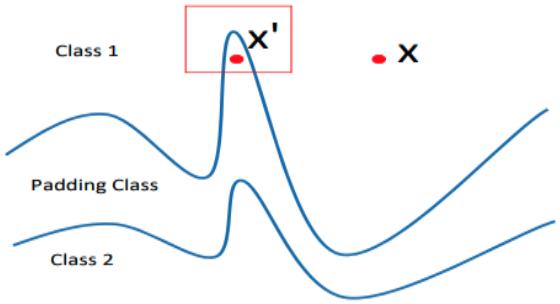

Figure 2.3. Illustration of the padding class used in PadNet. Source: [19]

PadNet was evaluated on the MNIST and CIFAR-10 datasets, and was shown to outperform robustness metrics of ALP and better adapted to different attacks [19]. This serves as the basis for future work by Barton, Jatho, and Berzins [15] into the concept of a barrier class, which would become the None of the Above (NOTA) class used by Jatho [8].

2.4.2 NOTA Attacks

If a NOTA or padding class were assumed to be built into a neural network, one may also assume that attacks would be adapted to account for this class. To build a more robust defense and test for accuracy against NOTA-aware attacks, Jatho adapted the stopping criteria, target selection, and objective function for the attacks given in the Adversarial Robustness Toolbox [24] to be aware of a NOTA class [8]. Specifically, Jatho uses the following NOTA-adapted attacks:

- Carlini-Wagner suite (L_2 and L_∞)
- APGD Cross-Entropy and Difference of Logits Ratio (DLR)
- Square Attack
- DeepFool
- AutoAttack (Untargeted)
- Gradient-based Saliency-Map Attack (GSMA)

Summaries of Jatho's changes are included in Section 3.4.7. Jatho [8] tested his defenses against both the original attacks and adapted attacks.

2.4.3 Naive Adversarial Mixup (NAM)

The NAM algorithm [8] creates samples of a NOTA class. It uses a linear combination of images, inspired by Mixup [25], to help create NOTA examples [8]. The idea of this approach is to "mix" a correctly labelled training set example with an adversarial example counterpart, creating another image that is then labelled as NOTA [8]. We may express the linear combination of inputs as:

$$X_{NAM} = \lambda x + (1 - \lambda) X_{ADV}, \tag{2.1}$$

where X_{ADV} is the adversarial example and λ is the weight used. NAM specifically produces two types of NOTA examples: mean NAM, which sets λ to 0.5, and uniform NAM, which randomizes λ between 0.05 and 0.95 [8].

Naive Adversarial Mixup against Original Attacks

Jatho tested NAM using the CIFAR-10 and CIFAR-100 datasets to find the Attack Success Rate (ASR), or fraction of successful examples over total examples [8]. For the CIFAR-10 dataset, all attacks showed reduction of ASR from undefended models to NAM-defended models, with the exception of the Square Attack L_2 model, which showed an increase of ASR of 3%. Further, while the NAM defense reduced ASR for Square Attack L_∞ and GSMA, the reduction of ASR from undefended CIFAR-10 models to defended models was limited to 5% or less. For the CIFAR-100 dataset, all attacks similarly showed the reduction of ASR from undefended models to NAM-defended models, with the exception of the Square Attack L_2 model, which showed an increase of ASR of 7% [8].

Naive Adversarial Mixup against NOTA-Aware Attack

The results of NAM against NOTA-aware attacks were mixed. For the CIFAR-10 dataset, some of the NOTA-aware attacks retained similar ASRs to their NOTA-unaware counterparts, but the CW L_2 attack, APGD DLR (L_2 and L_∞) attack, and untargeted AutoAttack (L_2 and L_∞) resulted in ASRs similar to the undefended model. For the CIFAR-100 dataset, the change in results was less drastic between the NOTA-unaware and NOTA-aware attacks, but for the most part still resulted in higher ASRs against the NOTA-aware attacks. For the CIFAR-100 dataset, we also note that the ASRs for the NOTA-aware APGD and untargeted AutoAttack were between 98% and 99% [8].

2.4.4 Modified Adversarial Mixup

Jatho creates two "reflexive" models for Adversarial Mixup. Simple Reflexive NOTA-Aware Adversarial Mixup (SRNA) [8] adds to the NAM approach by switching between two different losses. One loss seeks to maximize cross entropy loss for the original ('clean') class, and the other loss seeks to maximize cross entropy loss for the NOTA class. Jatho describes this method as "alternating between pushing away from the true label class to plant NOTA, and pushing away from existing NOTA to plant NOTA where it does not exist" [8].

Multi-Modal NOTA-Aware Adversarial Mixup (MRNA) builds on SRNA by increasing the number of loss functions used and increasing the ratio of NOTA examples to benign

examples. Jatho uses Adaptive Cross Entropy Loss [26] and Adaptive Difference Logits Ratio Loss [26] to create NOTA examples.

For both datasets, we see that both SRNA and MRNA achieved considerable reduction in ASR in comparison to the other analyzed models (for both NOTA-unaware and aware attacks). For CIFAR-10, both SRNA and MRNA significantly decreased ASR compared to the original NAM defense against most NOTA-aware attacks, with exceptions for the CW L_∞ attack, APGD CE attacks (L_2 and L_∞) and Square Attack (L_2). We note poor performance (small decrease in ASR) for APGD DLR L_∞ and for untargeted AutoAttack (L_∞) for the SRNA NAM models.

For the CIFAR-100 dataset, SRNA and MRNA are both successful compared to the original NAM against NOTA-aware attacks; however, we see a significant increase in ASR from NAM to SRNA for the Carlini-Wagner L2 attack and an increase in ASR from NAM to SRNA and MRNA for Square Attack (L_∞). Depending on the attack used, SRNA and MRNA have comparable performance, with SRNA generally performing slightly better for the Carlini-Wagner L_∞, APGD Cross Entropy, Square, and GSMA attacks for both CIFAR-10 and CIFAR-100 datasets [8]. *This thesis will utilize the SRNA model.*

We note that between the undefended models and the reflexive models, there is a small loss in model accuracy. For CIFAR-10, we see a 0.81% deduction in accuracy from an undefended model to a SRNA model, while for CIFAR-100, there is a 1.42% reduction in accuracy. We consider these losses to be acceptable, given the increased adversarial robustness.

2.5 Chapter Summary

This chapter presents an introduction to adversarial attacks, particularly evasion attacks, and defenses, based on the creation of adversarial examples during training. As part of this, we introduced NOTA defenses, which try to learn models that classify adversarial examples as a "None of the Above" class. This thesis builds on this by introducing NOTA-biased parameter priors, which will be explained in the next two chapters.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 3:
Methods

3.1 Model Methods

To build our model, we utilize a variety of methods, including MAP estimation, ReLU and softmax activation functions, and Dropout. These methods allow us to build a NN robust enough to support accurate multi-class image classification.

3.1.1 MAP Estimation

When training models, we used Maximum A Posteriori (MAP) estimation, which seeks to find the most likely model parameters given the training data. We can define our posterior $p(\theta|D)$ using the likelihood $p(D|\theta)$, the prior $p(\theta)$, and the evidence $p(D)$ as

$$p(\theta|D) = \frac{p(D|\theta)p(\theta)}{p(D)}. \tag{3.1}$$

The goal of MAP estimation can be described mathematically as

$$\underset{\theta}{\operatorname{argmax}}\ p(\theta|D) = \underset{\theta}{\operatorname{argmax}}\ \log\ p(D|\theta) + \log p(\theta). \tag{3.2}$$

This may also be represented as

$$\underset{\theta}{\operatorname{argmax}}\ p(\theta|D) = \underset{\theta}{\operatorname{argmin}} - \log\ p(D|\theta) - \log\ p(\theta). \tag{3.3}$$

For the negative log likelihood term, $-\log p(D|\theta)$, we use the cross entropy loss, since we are performing classification for one-hot encoded targets. The cross entropy loss between a one-hot vector, y, and a prediction, $p(y|x, \theta)$, is defined by

$$-\sum_{j=1}^{c} p(y_j) \log p(y = j|x, \theta). \tag{3.4}$$

3.1.2 ReLU Activation

Our model uses the Rectified Linear Unit (ReLU) activation function [27]. This activation function represents a piecewise function that sets input values less than zero to zero [28]. Mathematically, this is defined per

$$f(x) = max(0, x) = \begin{cases} x_i & x_i \geq 0 \\ 0 & x_i < 0. \end{cases} \tag{3.5}$$

3.1.3 Softmax Activation

The final layer in our network is a softmax activation layer [8]. Softmax functions are typically used in multi-class classification problems as it computes the probability of predicting each class given a logit [29] [28]. Mathematically, this can be written as

$$p(y = j|x, \theta) = \frac{e^{l_j}}{\sum_{k=0}^{c-1} e^{l_k}}. \tag{3.6}$$

In this equation, c represents the number of classes used and l_j represents the logit for each input class, j [29] [30].

3.1.4 Dropout

Our NN models utilize conventional dropout [31] during training. The idea of dropout is that each neuron will be dropped (i.e. multiply the output of some neurons by zero) p_{drop} percent of the time during training and will use the expected output value of each neuron during testing, as illustrated in Figure 3.1 [31].

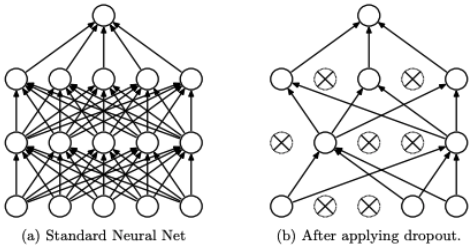

Figure 3.1. Dropout illustration. Source: [31].

3.2 Prior Regularization

A model prior defines which parameters to bias towards in the absence of data. By implementing a parameter prior, we can bias a model towards a specified behavior.

3.2.1 Complexity Priors

Typically, models will implement standard complexity priors in order to weight each class with equal probability. This is generally done through the use of a mean-zero Gaussian prior. A mean-zero Gaussian prior minimizes parameters, so the resulting prior term is $\frac{1}{2\sigma^2}\sum_{i=1}|\theta_i|^2$. The resulting optimization for finding the parameters that maximize our posterior is

$$\underset{\theta}{\text{argmin}} - logp(\theta|D) = \underset{\theta}{\text{argmin}} - \log p(D|\theta) + \frac{1}{2\sigma^2}\sum_{i=1}|\theta_i|^2. \tag{3.7}$$

Standard complexity priors are used in the vanilla model and the NOTA model from [8].

3.2.2 NOTA Priors for Adversarial Defense

We hypothesize that for NOTA-models, the majority of examples in input space will be classified as NOTA. This means that a model prior that corresponds to high NOTA probability might lower the Adversarial Success Rate (ASR).

We implement this prior by maintaining our mean-zero Gaussian priors for our non-NOTA classes, such that $p(b_i) \sim \mathcal{N}(0, \sigma^2_{prior})$. We then add in one more element to our prior vector to represent our NOTA class, using a Gaussian prior weighted using μ_{NOTA}, such that $p(b_{NOTA}) \sim \mathcal{N}(\mu_{NOTA}, \sigma^2_{prior})$. For the CIFAR-10 dataset, this means we used a prior vector with ten zeroes, and an eleventh non-zero number, representing μ_{NOTA}.

Using the proposed NOTA prior, the softmax logit for the NOTA class is $l_{NOTA} = w_{NOTA} \cdot x_l + b_{NOTA}$, where w_{NOTA} represents the weight matrix for the NOTA class applied to the input to the linear layer before the softmax x_l, and b_{NOTA} represents the bias for the NOTA class. Incorporating our NOTA logit into the classic softmax function leads to

$$p(y = NOTA|x, \theta) = \frac{e^{l_{NOTA}}}{(\sum_{k=0}^{c-1} e^{l_k}) + e^{l_{NOTA}}}. \tag{3.8}$$

As discussed in the complexity prior section, the most likely value for the non-NOTA logits under the standard prior complexity is zero. This allows us to easily control the probability of predicting NOTA using the most likely logits under the prior by setting the mean of the prior for the NOTA logit bias.

We may then compute our softmax function in terms of μ_{NOTA}, resulting in

$$p(y = NOTA|x, \theta^*) = \frac{e^{\mu_{NOTA}}}{c + e^{\mu_{NOTA}}}, \tag{3.9}$$

where θ^* represents the most likely θ under $p(\theta)$. Given a desired $p(y = NOTA|x, \theta^*)$, we can then find μ_{NOTA} using

$$\mu_{NOTA} = \log \frac{c * p(y = NOTA|x, \theta^*)}{1 - p(y = NOTA|x, \theta^*)}. \tag{3.10}$$

We conduct a grid search over a range of values for $p(y = NOTA|x, \theta^*)$, in order to find the μ_{NOTA} which both lowers ASR and leads to an acceptable accuracy on validation data. Additionally, we grid search over prior variances, which control the Weight Decay (WD) coefficient. We select the final value using the same criteria as for μ_{NOTA}. Performance metrics are further detailed in Section 4.3.

3.3 Model Architecture

This project builds the vanilla model used in [8], which is a Wide Residual Network (WRN) [32]. The layers of this vanilla model are listed in Figure 3.2, as well as the layers required to implement a NOTA-biased prior, as discussed in Section 3.2.

3.3.1 Wide ResNets

Our models are WRNs, as defined in [32]. The idea of WRN models is that a decreased depth and increased width residual network will be faster to train and increase model accuracy [32]. The models in [8] specifically use the WRN configuration used in [33]. For the CIFAR-10 (C10) dataset, a ResNet depth of 12 and ResNet width of 6 is used (WRN-12-6).

3.3.2 Model Printout

The model architectures for our NOTA-prior defended CIFAR-10 models are shown in Figure 3.2. Note that the vanilla model and the NOTA with adversarial examples architectures are the same, while the NOTA-prior only model and NOTA with adversarial examples and NOTA-prior model architectures are the same. The NOTA-prior models see the addition of a Batch Normalization layer and a final Add Layer. In this Add layer, we combine the final Dense layer with our prior vector. We may then apply our Softmax activation to our combined layer for our final output. Note that this model printout is for the CIFAR-10 dataset (a WRN-12-6 architecture), and will change depending on the size of the dataset used.

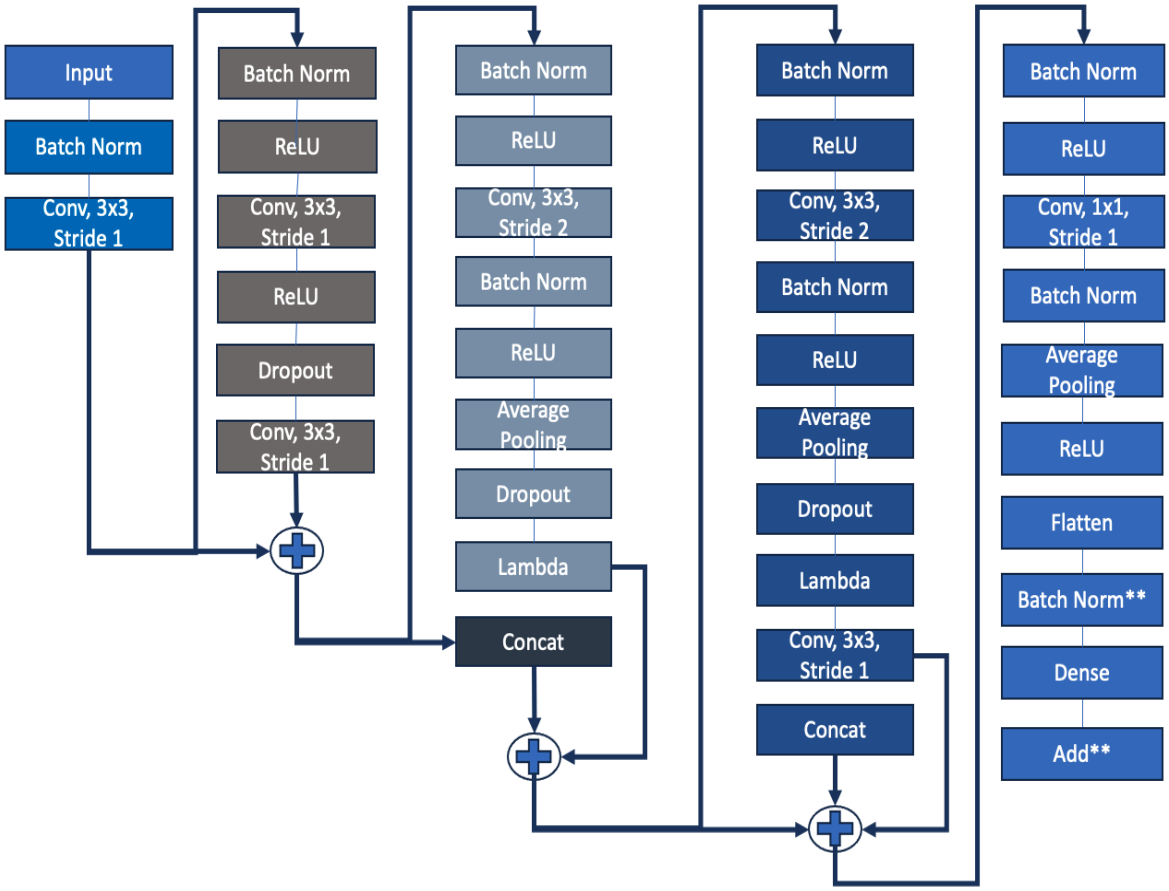

Figure 3.2. Model Architecture Printout for CIFAR-10 Models. **Indicates layers added for NOTA-prior models.

3.4 Attack Vectors

In this section, we discuss the various types of attacks used in this thesis. These attacks represent the most common evasion attacks. We also note how these attacks are adapted by Jatho [8], in order to account for the addition of a NOTA class.

3.4.1 Threat Model

This project maintains the same assumptions as written in [8]. This project does not consider attack vectors that occur during the training phase, including those that involve manipulation of training data. Specifically, this project focuses on evasion attacks. Two examples of how evasion attacks might be implemented are: 1) physical attacks, where adversaries alter physical objects (e.g., the adversarial tee-shirts discussed in Section 1.1), and 2) insider threats, where attackers with access may change a model's input to cause misclassification, as illustrated in Section 2.2.1 [8].

3.4.2 Carlini-Wagner (CW) Attacks

One of the most popular forms of adversarial attacks, Carlini-Wagner attacks [34], attempt to find the smallest noise that leads to an adversarial example. These attacks typically use L_p norms to measure how much adversarial noise, δ, is added to a clean example. The specific L_p norms used in this thesis are the L_2 and L_∞ distance. This thesis uses CW Untargeted attacks, which find adversarial noise using:

$$\underset{\delta}{\operatorname{argmin}} \ \| \delta \|_p + \beta * \operatorname{ReLU}(l_y - \max_{i \neq y} l_i + \gamma), \tag{3.11}$$

where l_y is the logit for the true label, $\max_{i \neq y} l_i$ is the highest logit across all non-y classes, β is the coefficient for our constraint term, and γ is our confidence (how much we want to push into another class region). CW attacks iteratively try to find the smallest positive β that leads to an adversarial example. Because this process is iterative, it is also relatively computationally expensive.

Distance Metrics

CW Attacks use L_p norms to measure the distance between an unmodified input (x) and a similar, modified input (x'), with $\delta = x' - x$ [34].

L_2 **Distance:** Carlini and Wagner [34] define the L_2 distance as the Euclidean distance to an adversarial example, given by

$$L_2(x) = \left(\sum_{i=0}^{d-1} |x_i|^2 \right)^{\frac{1}{2}}. \tag{3.12}$$

This project utilizes a standard maximum L_2 distance of 0.5 [8]. In Figure 3.3, the resulting L_2 adversarial images for the CIFAR-10 dataset from [34] showcase the imperceptible changes from the initial images.

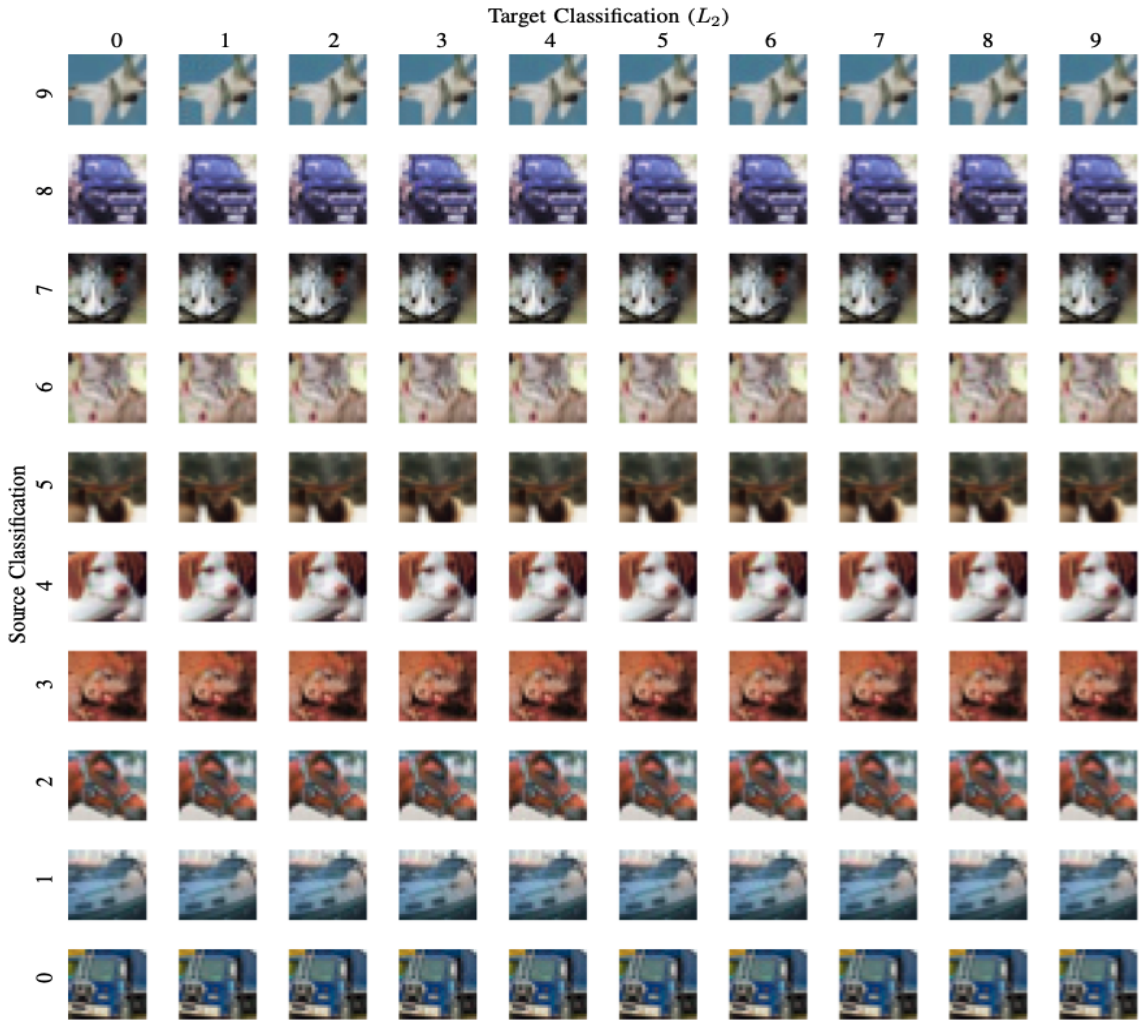

Figure 3.3. L2 attack applied to the CIFAR-10 dataset. Source: [34]

L_∞ **Distance:** L_∞ distance is the "maximum change to any of the coordinates" [34]. This can be defined as

$$\| \sigma \|_\infty = \max(|\sigma_1|, ..., |\sigma_n|). \tag{3.13}$$

In terms of L_p norms, we define the L_∞ distance as the limit of L as p goes toward infinity,

$$\lim_{p \to \infty} L_p(\sigma) = \left(\sum_{i=0}^{d-1} |\sigma_i|^p \right)^{\frac{1}{p}}. \tag{3.14}$$

This project utilizes a standard maximum L_∞ distance of 0.031 [8].

3.4.3 Auto-PGD

Auto-PGD, or APGD [35], is an update to the PGD attack [18] that uses an adaptive learning rate and momentum. Specifically, APGD uses automatic learning rate annealing, halving the learning rate each iteration that loss does not improve [35].

In addition to performing APGD using a cross-entropy loss, [35] performs APGD with a DLR loss. Mathematically, the untargeted DLR loss is defined as

$$DLR(l) = \frac{l_y - \max_{i \neq y} l_i}{l_{max} - l_{third}}, \tag{3.15}$$

where l_y is the logit for the true (correct) class, l_i is the highest incorrect class logit, l_{max} is the highest logit, and l_{third} is the third highest logit [35]. The use of DLR loss allows us to constrain logits beyond just the logit for the target class.

For the targeted DLR loss, we want to push toward a given target class. Therefore, we change our loss to

$$DLR_{targ}(l) = \frac{l_y - l_t}{l_{max} - \frac{l_{third} + l_{fourth}}{2}}. \tag{3.16}$$

3.4.4 DeepFool Attack

This thesis also uses the DeepFool [36] attack as a part of AutoAttack. The idea of DeepFool is to iteratively find the shortest distance to a local linear approximation of the decision boundary [36]. The pseudocode from [36] is shown in Algorithm 1.

Algorithm 1: DeepFool: multi-class case [36]

input: Image x, classifier f ;
output: Perturbation $\hat{r}$;

$y_{\text{pred}}(x) = \text{argmax}(f(x))$;
Initialize $x_0 \leftarrow x, i \leftarrow 0$;
while $y_{pred}(x_i) = y_{pred}(x_0)$ **do**
  for $y \neq y_{pred}(x_0)$ **do**
    $w'_y \leftarrow \nabla f_y(x_i) - \nabla f_{y_{\text{pred}}(x_0)}(x_i)$;
    $f'_y \leftarrow f_y(x_i) - f_{y_{\text{pred}}(x_0)}(x_i)$;
  $\hat{l} \leftarrow \text{argmin}_{y \neq y_{\text{pred}}(x_0)} \dfrac{|f'_y|}{||w'_y||_2}$;
  $r_i \leftarrow \dfrac{|f'_{\hat{l}}|}{||w'_{\hat{l}}||_2^2} w'_{\hat{l}}$;
  $x_{i+1} \leftarrow x_i + r_i$;
  $i \leftarrow i + 1$;
return: $\hat{r} = \sum_i r_i$

3.4.5 Square Attack

Square Attack [37] is an evasion attack that does not use gradient methods, but utilizes random search over adversarial noise to find better adversarial examples. In the case of untargeted Square Attack, this search would check to see if the noise increases the loss for the correct label. In the case of targeted Square Attack, the search checks if noise decreases the loss for the target label. To implement this attack, the height and width and location for a rectangle are randomly chosen, as well as a change (bounded by an ϵ) [37]. The pseudocode from [37] is shown in Algorithm 2.

Per [37] [8], this attack seeks to optimize

$$\min_{\hat{x} \in [0,1]^d} \mathcal{L}(f(x'), y), s.t. \parallel x' - x \parallel_p \leq \epsilon. \tag{3.17}$$

Algorithm 2: Square Attack via random search [37]

input: classifier f, point $x \in \mathbb{R}^d$, image size w, number of color channels c, l_p-radius ϵ, label $y \in [1, ..., K]$, number of iterations N ;

output: approximate minimizer $x' \in \mathbb{R}^d$ of the problem stated in equation 3.17 ;

$x' \leftarrow init(x)$, $l^* \leftarrow \mathcal{L}(f(x), y)$, $i \leftarrow 1$;

while $i \leq N$ ***and*** x' *is not adversarial* **do**

  $h^{(i)} \leftarrow$ side length of the square to modify (according to a schedule);

  $\delta\ P(\epsilon, h^{(i)}, w, c, x', x)$ (see paper for sampling distributions.);

  $x'_{new} \leftarrow$ Project $x' + \delta$ onto $\{z \in \mathbb{R}^d : ||z - x||_p \leq \epsilon\} \cap [0, 1]^d$;

  $l_{new} \leftarrow \mathcal{L}(f(x'_{new}), y)$;

  if $l_{new} < l^*$ **then**

    $x' \leftarrow x'_{new}$;

    $l^* \leftarrow l_{new}$;

  $i \leftarrow i + 1$

return: x'

3.4.6 AutoAttack

AutoAttack sequentially applies $APGD_{CE}$, APG_{DLR}, the FAB attack (replaced by DeepFool in the Adversarial Robustness Toolbox [24]), and the Square Attack, in order to create a parameter-free attack [35]. This thesis tests both untargeted and targeted AutoAttack. Targeted AutoAttack first sequentially conducts untargeted versions of each attack listed, and if unsuccessful, sequentially conducts targeted versions of each attack listed.

3.4.7 NOTA-Adapted Attacks

This thesis utilizes NOTA-aware versions of the attacks listed in Section 3.4 when attacking NOTA-defended models. These attacks are built to prevent a model from predicting NOTA, which makes them better than the standard attacks for evaluating the robustness of our defense. Summaries of the proposed updates from [8] to each attack vector are listed below.

NOTA-Aware Carlini-Wagner Attacks

Jatho [8] adapts the CW attacks by updating the loss function so that an adversarial example is found with a class that is not equal to the true class or the NOTA class. The updated untargeted loss function is

$$\underset{\delta}{\text{argmin}} \ \| \delta \|_p + \beta * \text{ReLU}\left(\max_{j=y \ or j=NOTA} l_j \ - \max_{i \neq y \ and \ i \neq NOTA} l_i \right), \tag{3.18}$$

where the ReLU term represents us pushing away from both the true and NOTA classes so that they are not selected [8].

NOTA-Aware APGD Attack

Jatho's NOTA-Adapted APGD attack [8] ensures we do not select NOTA as our target. The implementation also ensures that a NOTA prediction is not considered a success; the attack does not stop if NOTA is predicted. For APGD-DLR, NOTA is not used in any of the logits used to calculate loss [8].

NOTA-Aware DeepFool Attack

As Jatho describes in [8], a DeepFool attack updated to account for NOTA removes NOTA as a potential target for selection and updates the stopping criteria so that the attack will not stop on a NOTA-prediction unless it reaches maximum iterations.

NOTA-Aware Square Attack

To update the Square Attack, Jatho [8] updated the stopping criteria to ensure it would not stop on the true class or the NOTA class. Additionally, Jatho updated the attack so that NOTA could not be chosen as the correct label in the case where labels were not given.

NOTA-Aware AutoAttack

NOTA-Aware AutoAttack consists of NOTA-Aware attacks for APGD, DeepFool, and Square Attack (described above).

3.5 Chapter Summary

This chapter defines the model architecture and methods used in this thesis. We first explain the model architecture, a WRN model with dropout. We explain the use of MAP and ReLU and softmax activation functions. We then explain the use of complexity priors in vanilla models and how we adapt a complexity prior to be biased toward the NOTA class. A full model printout is included, emphasizing the difference between the vanilla and NOTA-prior defended models. Additionally, we explain each of the attack vectors used and how they were adapted to be aware of a NOTA class.

CHAPTER 4:
Experiments

4.1 Dataset Usage

This thesis uses the CIFAR-10 dataset [38], a relatively small image dataset, in order to understand if a NOTA-biased prior will create any noticeable change in model performance. This dataset uses 60,000 32x32 RGB images over ten classes (6,000 images per class), shown in Figure 4.1.

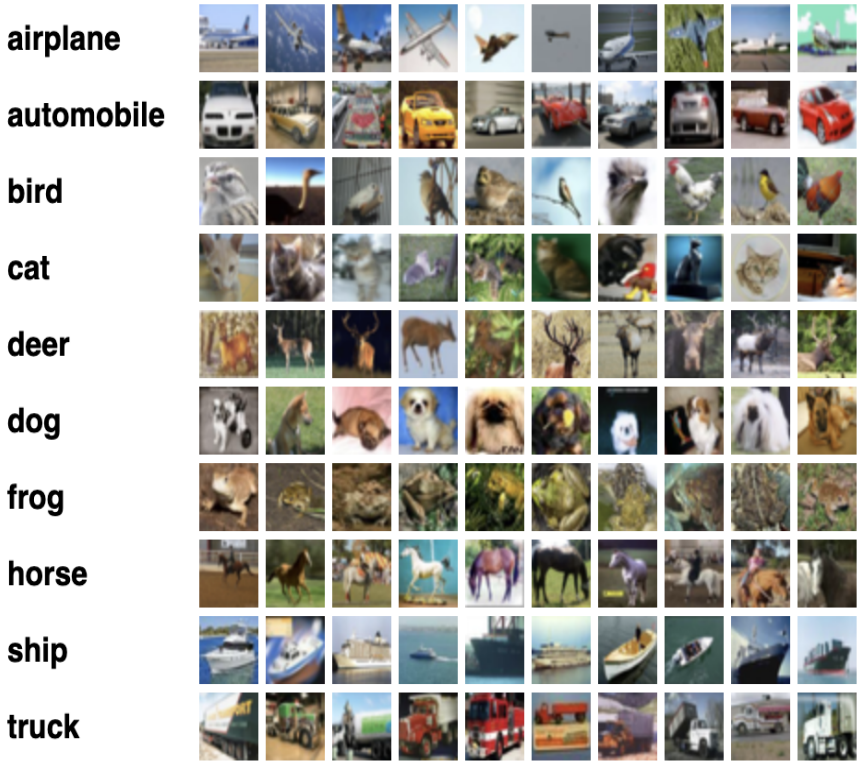

Figure 4.1. CIFAR-10 dataset images. Source: [38]

4.2 Experiment Design

This thesis computes four different types of models, as listed in Section 1.3. We compare the NOTA-prior only model to the vanilla model and compare the NOTA Examples with Prior model to the NOTA Examples model, using the performance metrics discussed in Section 4.3. Similarly, we compare the NOTA Examples with Prior model to the vanilla model to understand how our defense works broadly compared to an undefended model.

4.3 Performance Metrics

Our training script returns ASR, Training Accuracy (TA), and Validation Accuracy (VA) for the models it builds. ASR is defined as $\frac{\text{successful examples}}{\text{total examples}}$ [8], while TA and VA are defined as $\frac{\text{correct predictions}}{\text{total predictions}}$ for the training set and validation set, respectively [39].

VA is the metric by which we define our grid search; we take the model which gives the best combination of lowest ASR and highest VA to run against our test script. The parameters for the models chosen are given in Table 4.1. Our test script returns model accuracy and Regular Class Accuracy (RCA), both further discussed in Section 4.5.

4.4 Results Documentation

For each model type, we conducted validation grid search in order to find the best values for μ_{NOTA} and weight decay that resulted in lowest ASR while retaining validation accuracy. These values are recorded in Table 4.1.

Table 4.1. Validation grid search results for μ_{NOTA} and prior weight for the CIFAR-10 dataset

Parameters from Chosen Models		
	μ_{NOTA}	WD
Prior-Only Model	15	$1x10^{-6}$
NOTA Examples + Prior Model	16	$1x10^{-2}$

We note that a larger value for μ_{NOTA} and a stronger prior is required for the NOTA with Examples model. In the following sections, we highlight the results from our test script on each of these models using the parameters above.

4.4.1 Prior Defended Models

The numerical results for NOTA-prior only models are shown in Table 4.2. These are also shown graphically in Figure 4.2. With the addition of the prior only, we see reduction of ASR for CW L_∞, APGD CE and DLR L_2 by at least 12 to 14% compared to the vanilla model. We also note that the ASR increases by 4% for the CW L_2 models from vanilla model to the prior-only model.

4.4.2 NOTA with Examples and Prior Defended Models

The results for all models are shown in Figure 4.2. While there is little to no change in attack success from the NOTA with adversarial examples to NOTA with adversarial examples and prior models for the Carlini-Wagner attacks, APGD CE attacks, and targeted AutoAttack examples, we see a 9% decrease in ASR for the APGD DLR L_2 attack, a 59% decrease in ASR for the APGD DLR L_∞ attack, and a 13% decrease in ASR for untargeted AutoAttack L_∞ examples. These results indicate that the addition of a NOTA-biased prior provide an additional layer of robustness for the NOTA models against more complex attacks.

We note that the CW attacks already see a very low ASR, perhaps explaining why we do not see further reduction in ASR. We also note that the targeted AutoAttack is the strongest attack, explaining why we still see a near 100% ASR.

Table 4.2. Attack success rate comparison for all models. Note that 'NOTA Ex' represents Jatho's NOTA with Examples model, while 'Ex + Prior' represents the NOTA with Examples model with the addition of a NOTA Prior.

	Carlini-Wagner		APGD				AutoAttack			
			CE		DLR		Untargeted		Targeted	
	L_2	L_∞	L_2	L_∞	L_2	L_∞	L_2	L_∞	L_2	L_∞
Vanilla Det. ASR	0.96	1	1	1	0.92	0.99	1	1	1	1
Prior-Only Det. ASR	1	0.88	0.86	1	0.78	0.98	0.97	1	0.99	1
NOTA Ex Det. ASR	0.09	0.05	0.05	0.05	0.88	0.95	0.86	0.92	0.99	1
Ex + Prior Det. ASR	0.08	0.07	0.06	0.06	0.79	0.36	0.8	0.79	0.99	0.98

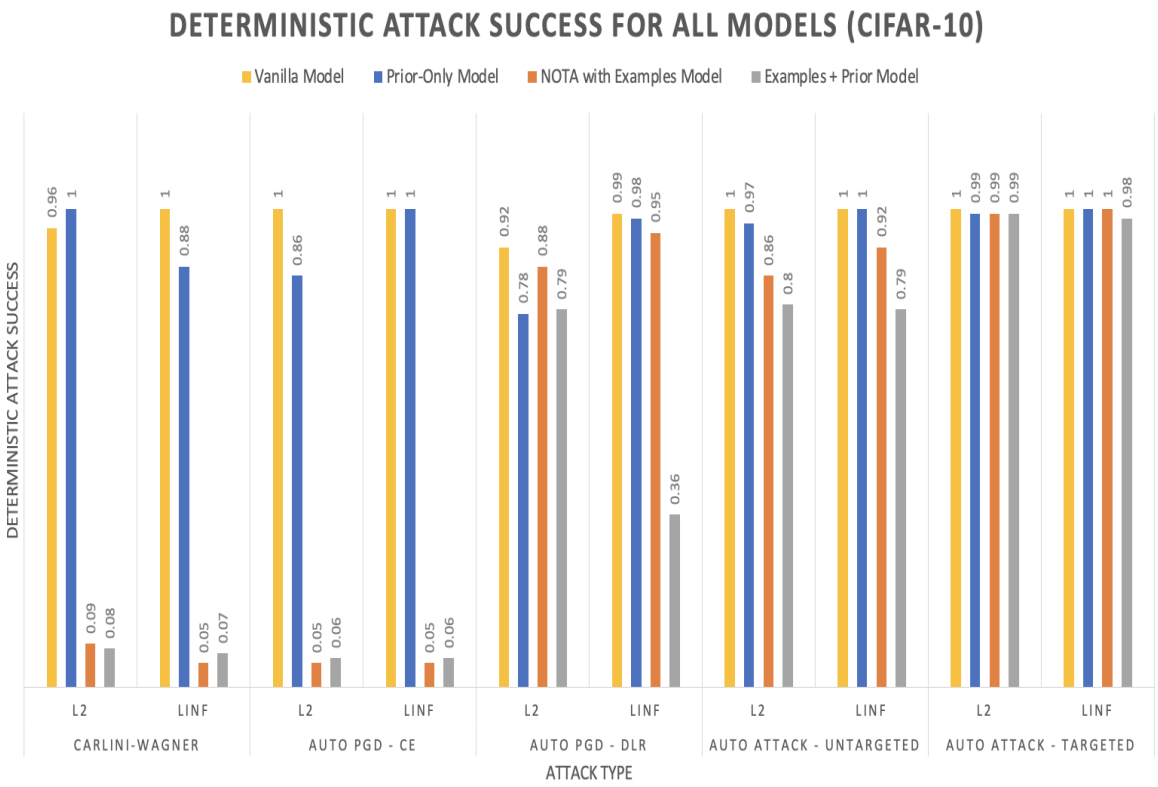

Figure 4.2. ASR for all attacks on the CIFAR-10 dataset

4.5 Accuracy

Our models output two different types of accuracy: test set accuracy and regular class accuracy. RCA computes the amount of samples placed in the correct class. We analyze these metrics to determine how the addition of a NOTA-prior affects overall accuracy of the model and if it changes if examples are given a correct class label, or appropriately sorted into the NOTA class.

4.5.1 Test Set Accuracy

Table 4.3 showcases the test set accuracies for all models. We note overall loss of accuracy across all models, with test set accuracy dropping less than 1% between the vanilla model and NOTA Examples and NOTA-prior model. We characterize this loss as acceptable, given the increased model robustness.

Table 4.3. Test Set Model Accuracy for all models

Model	Deterministic Accuracy
Vanilla Model	0.9259
Prior-Only Model	0.9207
NOTA Examples Model	0.9169
NOTA Examples + Prior Model	0.9191

4.5.2 Regular Class Accuracy

Table 4.4 showcases the change in RCA for all models. We note a general trend of near-zero RCA, with the exception of the RCAs for the Carlini-Wagner attack models. The Carlini-Wagner NOTA models show RCAs over 0.9, indicating that the NOTA defense leads to examples that are correctly classified, while the near-zero RCAs for APGD and AutoAttack NOTA models suggests that examples are predominantly classified as NOTA. We consider both of these results favorable, with their differences showcasing the differences between attacks used. While the Carlini-Wagner attacks minimize noise, the other attacks bound noise, leading to differences in the adversarial examples generated.

Table 4.4. Regular Class Accuracy (RCA) for all models

	Carlini-Wagner		APGD				AutoAttack			
			CE		DLR		Untargeted		Targeted	
	L_2	L_∞	L_2	L_∞	L_2	L_∞	L_2	L_∞	L_2	L_∞
Vanilla RCA	0.04	0	0	0	0.08	0.01	0	0	0	0
Prior-Only RCA	0	0.12	0.04	0	0.22	0.02	0.03	0	0.01	0
NOTA Ex. RCA	0.91	0.95	0	0	0	0	0.14	0.07	0.01	0
NOTA Ex. + Prior RCA	0.92	0.93	0	0	0	0	0.18	0.21	0.01	0.02

4.6 Chapter Summary

This chapter summarizes our dataset usage and implementation of our four different models. Results are given for prior-only models, as well as NOTA with adversarial examples and prior models. This chapter summarizes how the addition of a NOTA-prior to a NOTA with adversarial examples model can increase adversarial robustness against some attacks, while suffering only a very minor reduction in accuracy.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 5:
Conclusion and Future Work

As ML becomes more integrated into safety and security-critical systems, building models with robust security and defense mechanisms is increasingly important. The addition of a NOTA class and a complexity prior biased toward the NOTA class can provide an additional measure of defense against adversarial attacks.

5.1 Contributions

This thesis provides a proof-of-concept of the efficacy of a NOTA-biased prior towards reducing the effectiveness of evasion attacks. We derive a complexity prior that is biased toward the NOTA class and conduct validation grid search over the prior and prior variance in order to find a combination of parameters that best decreases adversarial success.

Testing using the CIFAR-10 dataset, we are able to show that the addition of a NOTA-prior to Jatho's [8] NOTA model trained with adversarial examples increases adversarial robustness. Specifically, the NOTA-prior increases adversarial robustness against APGD-DLR and untargeted AutoAttack vectors, two strong attacks that were very successful against the previous models.

5.2 Future Work

Additional work is ongoing for this project. This work includes the testing of the Tiny ImageNet (TI) [40] dataset, a larger dataset of 64x64x3 images in 200 classes. The results from this dataset are meant to allow us to understand if there is a noticeable difference in results across datasets, and how the size and number of classes in a dataset may change the effectiveness of a NOTA defense. Testing of this dataset will allow us to understand more completely how to most effectively implement a NOTA-prior across a variety of inputs.

Other future work may include investigating the use of NOTA-priors with stochastic models. In particular, models that utilize the various Bayesian methods, namely Monte Carlo Dropout and Laplace Approximation, used in [8].

5.3 Conclusion

By implementing this small change to our models, we believe we can increase overall robustness against stronger attack vectors, while minimally affecting model accuracy. While further analysis is still needed to understand how a NOTA-biased prior affects larger datasets, this thesis provides a proof-of-concept for the increased robustness that may be provided from a NOTA-biased prior. Given the increasing need for trustworthy AI and ML systems, we believe this work contributes an additional measure of defense against adversarial ML.

List of References

[1] J. Biden, "National Security Strategy of the United States of America," Washington, DC, USA, 2022. Available: https://www.whitehouse.gov/wp-content/uploads/2022/10/Biden-Harris-Administrations-National-Security-Strategy-10.2022.pdf

[2] M. Gilday, "Chief of Naval Operations Navigation Plan 2022," Washington, DC, USA, 2022. Available: https://media.defense.gov/2022/Jul/26/2003042389/-1/-1/1/NAVIGATION%20PLAN%202022_SIGNED.PDF

[3] C. Q. Choi, "7 Revealing Ways AIs fail," *IEEE Spectrum*, Sep. 21 2021. Available: https://spectrum.ieee.org/ai-failures/

[4] B. Chakravorti, "Why AI Failed to Live Up to Its Potential During the Pandemic," *Harvard Business Review*, Mar. 17, 2022. Available: https://hbr.org/2022/03/why-ai-failed-to-live-up-to-its-potential-during-the-pandemic/

[5] R. Awasthi, "Breaking Deep Learning with Adversarial examples using Tensorflow," Available: https://cv-tricks.com/how-to/breaking-deep-learning-with-adversarial-examples-using-tensorflow/ (2023/01/15), 2017.

[6] K. Xu, G. Zhang, S. Liu, Q. Fan, M. Sun, H. Chen, P.-Y. Chen, Y. Wang, and X. Lin, "Adversarial t-shirt! evading person detectors in a physical world," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part V 16*. Springer, 2020, pp. 665–681.

[7] E. Jatho and J. Kroll, "Artificial Intelligence: Too Fragile to Fight?" *U.S. Naval Institute Proceedings*, vol. 148, no. 2, Feb. 2022. Available: https://www.usni.org/magazines/proceedings/2022/february/artificial-intelligence-too-fragile-fight

[8] E. W. J. III, "Finding and Fixing Fragility in Machine Learning," PhD dissertation, Dept. of Comp. Sci., Naval Postgraduate School, Monterey, CA USA, June 2023, Available: https://calhoun.nps.edu/handle/10945/72193.

[9] P. Gudikandula, "A Beginner Intro to Neural Networks," Medium, March 19, 2019. Available: https://purnasaigudikandula.medium.com/a-beginner-intro-to-neural-networks-543267bda3c8/.

[10] M. Sahay, "Neural Networks and the Universal Approximation Theorem," TowardsDataScience, Jun 6, 2020. Available: https://towardsdatascience.com/neural-networks-and-the-universal-approximation-theorem-8a389a33d30a/.

[11] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[12] L. Shukla, "Designing Your Neural Networks," TowardsDataScience, Sept 23, 2019. Available: https://towardsdatascience.com/designing-your-neural-networks-a5e4617027ed/.

[13] A. Chakraborty, M. Alam, V. Dey, A. Chattopadhyay, and D. Mukhopadhyay, "Adversarial attacks and defences: A survey," *arXiv*, 2018, Available: https://doi.org/10.48550/arXiv.1810.00069.

[14] M. Rigaki and S. Garcia, "A survey of privacy attacks in machine learning," *ACM Comput. Surv.*, vol. 56, no. 4, nov 2023. Available: https://doi.org/10.1145/3624010

[15] A. Barton and I. Jatho, Edgar, "Defending Against Adversarial Examples in Deep Neural Network Classifiers," NAVAIR, 2021-12-31, prepared for: NAVAIR. Available: http://hdl.handle.net/10945/68624

[16] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv*, 2013, Available: https://doi.org/10.48550/arXiv.1312.6199.

[17] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv*, 2014, Available: https://doi.org/10.48550/arXiv.1412.6572.

[18] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv*, 2017, Available: https://doi.org/10.48550/arXiv.1706.06083.

[19] A. Barton, "Defending Neural Networks Against Adversarial Examples," PhD dissertation, Dept. of Comp. Sci., The University of Texas at Arlington, Arlington, TX, December 2018, Available: https://rc.library.uta.edu/uta-ir/bitstream/handle/10106/27743/BARTON-DISSERTATION-2018.pdf?sequence=1.

[20] S. Lee, H. Lee, and S. Yoon, "Adversarial vertex mixup: Toward better adversarially robust generalization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 272–281.

[21] H. Kannan, A. Kurakin, and I. Goodfellow, "Adversarial logit pairing," *arXiv*, 2018, Available: https://doi.org/10.48550/arXiv.1803.06373.

[22] Z. Wang, T. Pang, C. Du, M. Lin, W. Liu, and S. Yan, "Better diffusion models further improve adversarial training," *arXiv*, 2023, Available: https://doi.org/10.48550/arXiv.2302.04638.

[23] X. Cao and N. Z. Gong, "Mitigating evasion attacks to deep neural networks via region-based classification," in *Proceedings of the 33rd Annual Computer Security Applications Conference*, 2017, pp. 278–287.

[24] M.-I. Nicolae, M. Sinn, M. N. Tran, B. Buesser, A. Rawat, M. Wistuba, V. Zantedeschi, N. Baracaldo, B. Chen, H. Ludwig *et al.*, "Adversarial robustness toolbox v1. 0.0," *arXiv*, 2018, Available: https://doi.org/10.48550/arXiv.1807.01069.

[25] W. E. Zhang, Q. Z. Sheng, A. Alhazmi, and C. Li, "Adversarial attacks on deep-learning models in natural language processing: A survey," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 11, no. 3, pp. 1–41, 2020.

[26] F. Croce and M. Hein, "Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks," in *International conference on machine learning*. PMLR, 2020, pp. 2206–2216.

[27] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.

[28] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, "Activation functions: Comparison of trends in practice and research for deep learning," *arXiv*, 2018, Available: https://doi.org/10.48550/arXiv.1811.03378.

[29] K. E. Koech, "Softmax Activation Function — How It Actually Works," Towards-DataScience, Sep 30, 2020. Available: https://towardsdatascience.com/softmax-activation-function-how-it-actually-works-d292d335bd78.

[30] "Module 2: Maximum Likelihood Estimation and Maximum a Posterior Estimation," class notes for Bayesian Methods for Neural Networks, Dept. of Comp. Sci., Naval Postgraduate School, Monterey, CA, USA, winter 2023.

[31] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014. Available: http://jmlr.org/papers/v15/srivastava14a.html

[32] S. Zagoruyko and N. Komodakis, "Wide residual networks," *arXiv*, 2016, Available: https://doi.org/10.48550/arXiv.1605.07146.

[33] M. D. McDonnell, "Training wide residual networks for deployment using a single bit for each weight," *arXiv*, 2018, Available: https://doi.org/10.48550/arXiv.1802.08530.

[34] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *2017 ieee symposium on security and privacy (sp)*. IEEE, 2017, pp. 39–57.

[35] F. Croce and M. Hein, "Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks," in *Proceedings of the 37th International Conference on Machine Learning* (Proceedings of Machine Learning Research), H. D. III and A. Singh, Eds. PMLR, 13–18 Jul 2020, vol. 119, pp. 2206–2216. Available: https://proceedings.mlr.press/v119/croce20b.html

[36] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2574–2582.

[37] M. Andriushchenko, F. Croce, N. Flammarion, and M. Hein, "Square attack: a query-efficient black-box adversarial attack via random search," in *European conference on computer vision*. Springer, 2020, pp. 484–501.

[38] A. Krizhevsky and G. Hinton, "Learning Multiple Layers of Features from Tiny Images," University of Toronto, Toronto, Ontario, Tech. Rep. 0, 2009.

[39] A. Geron, *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*, 2nd ed. O'Reilly Media, Inc., 2019.

[40] Y. Le and X. Yang, "Tiny imagenet visual recognition challenge," *CS 231N*, vol. 7, no. 7, p. 3, 2015.

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California