NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

DETECTING AND DEFENDING AGAINST DIFFERENT FAMILIES OF ADVERSARIAL EXAMPLE ATTACKS

by

Shaun Kallis

June 2023

Thesis Advisor: Armon C. Barton
Second Reader: Neil C. Rowe

THIS PAGE INTENTIONALLY LEFT BLANK

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

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

4. TITLE AND SUBTITLE
DETECTING AND DEFENDING AGAINST DIFFERENT FAMILIES OF ADVERSARIAL EXAMPLE ATTACKS

5. FUNDING NUMBERS

6. AUTHOR(S) Shaun Kallis

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)

Adversarial example attacks alter an image so the image appears largely unaltered to human eyes, but image-recognition models will misclassify it. This is a common type of attack, against which there is currently no good general defense. Most state-of-the-art methods of detecting adversarial example attacks only consistently succeed in recognizing a few known attacks. These defenses do not generalize well to detecting other attacks, which means an adversary only needs to change their attack to leave us without robust abilities to detect attacks. Military intelligence increasingly relies on machine learning image recognition for analyzing satellite images. Finding defenses against these adversarial example attacks is important for ensuring our intelligence-gathering capabilities are not compromised. This thesis seeks to contribute models which will push the state of the art towards successful recognition of adversarial attacks regardless of which type of attack was used. Models we named 3-Mix were trained using combinations of different attacked images; other models were trained using SaliencyMix. These defenses were evaluated against ten attacks: PGD, auto-PGD, autoattack, square, Carlini L2 and L-inf, deepfool, elasticnet, JSMA, and boundary. On average the attack success rate against the best defense model was 0.12 for 3-Mix, 0.31 for SaliencyMix, and 0.77 for comparison model Mixup.

14. SUBJECT TERMS
adversarial examples, adversarial attacks, machine learning image recognition, convolutional neural network, CNN, deep neural network, DNN, SaliencyMix, 3-Mix, N-Mix, PadNet, NOTA, Adversarial Robustness Toolbox, ART, mixup, adversarial mixup, data augmentation

15. NUMBER OF PAGES
71

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

DETECTING AND DEFENDING AGAINST DIFFERENT FAMILIES OF ADVERSARIAL EXAMPLE ATTACKS

Shaun Kallis
Civilian, Non US Govt
BS, California State University, Monterey Bay, 2020

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
June 2023

Approved by: Armon C. Barton
Advisor

Neil C. Rowe
Second Reader

Gurminder Singh
Chair, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Adversarial example attacks alter an image so the image appears largely unaltered to human eyes, but image-recognition models will misclassify it. This is a common type of attack, against which there is currently no good general defense. Most state-of-the-art methods of detecting adversarial example attacks only consistently succeed in recognizing a few known attacks. These defenses do not generalize well to detecting other attacks, which means an adversary only needs to change their attack to leave us without robust abilities to detect attacks. Military intelligence increasingly relies on machine learning image recognition for analyzing satellite images. Finding defenses against these adversarial example attacks is important for ensuring our intelligence-gathering capabilities are not compromised. This thesis seeks to contribute models which will push the state of the art towards successful recognition of adversarial attacks regardless of which type of attack was used. Models we named 3-Mix were trained using combinations of different attacked images; other models were trained using SaliencyMix. These defenses were evaluated against ten attacks: PGD, auto-PGD, autoattack, square, Carlini L2 and L-inf, deepfool, elasticnet, JSMA, and boundary. On average the attack success rate against the best defense model was 0.12 for 3-Mix, 0.31 for SaliencyMix, and 0.77 for comparison model Mixup.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

THIS PAGE INTENTIONALLY LEFT BLANK

List of Figures

List of Tables

THIS PAGE INTENTIONALLY LEFT BLANK

List of Acronyms and Abbreviations

AE	adversarial examples
ART	adversarial robustness toolbox
ASR	attack success rate
CNN	convolutional neural network
CW	Carlini Wagner
DNN	deep neural network
FGSM	fast gradient sign method
GPU	graphics processing unit
JSMA	Jacobian saliency map attack
NOTA	none of the above
PGD	projected gradient descent
QR	quick response
ResNet	residual neural network
WRN	wide residual network

THIS PAGE INTENTIONALLY LEFT BLANK

Acknowledgments

This material is based upon activities supported by the National Science Foundation under Agreement No. 1565443. Any opinions, findings, and conclusions or recommendations expressed are those of the author and do not necessarily reflect the views of the National Science Foundation.

I would like to express my sincere gratitude to my advisor, Dr. Armon Barton, for his invaluable guidance, unwavering patience, and insightful contributions throughout the entire research process. His expertise and mentorship have been instrumental in shaping the direction and quality of this thesis. I extend my heartfelt appreciation to Georgios Andrianopoulos, whose early contributions laid the foundation for this research. His expertise and collaboration were vital in shaping the initial stages of this work. I am deeply grateful to my second reader, Professor Neil Rowe, for his meticulous review and constructive feedback. His invaluable suggestions and insights significantly enhanced the clarity and coherence of the writing. I also would like to acknowledge the exceptional education and guidance provided by all my professors at the Naval Postgraduate School.

I want to extend my sincere thanks to the Scholarship for Service program for making this educational pursuit possible. The financial support provided by the National Science Foundation and the Office of Personnel Management through the SFS program has been invaluable in facilitating my research and education.

I am grateful for the support and companionship of my fellow cohort members. Their encouragement and camaraderie have made this academic journey all the more fulfilling and enjoyable. I also deeply appreciate my friend Sierra for their unwavering encouragement and steadfast friendship.

Lastly, I want to convey my deepest gratitude to my family, especially my parents, grandmother, aunt Katherine and uncle Floyd. Their unconditional love, unwavering encouragement, and steadfast support have been the cornerstone of my accomplishments. Their unwavering belief in my abilities has been invaluable, and I am forever grateful for the guidance they have provided.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1:
The Proliferation of Image Recognition

The benefits of machine learning are increasingly recognized in society as its use increases, but the adoption of machine learning also brings security risks to its users. Computer vision is a subset of machine learning in which computers imitate functions of the human visual system. Image recognition is a subset of this field wherein a digital image is received by the computer and the computer processes the digital information in the image to identify what objects that data represents in the real-world. Image recognition is a well-developed and very useful aspect of machine learning. It has common uses in real world industry applications such as facial recognition for unlocking phones, quick response (QR) code recognition using cell-phone cameras, character recognition for scanning written or typed text and converting to digital text, agricultural use in identifying fruit which is ripe for picking or sorting produce, reading of scans to detect medical conditions, and self-driving vehicles identifying objects so the cars can navigate in the real-world. The U.S. government and military use image recognition for tasks such as facial recognition for criminal investigations, identifying and cataloging the military equipment of other countries from spy-satellite images, monitoring the U.S. border using images from drones, self-piloting military vehicles, and other applications which are not publicly disclosed.

Convolutional neural networks (CNNs) have been developed to be superb at image recognition tasks. CNNs are widely used for all the applications listed above and others, and as such have become an important digital tool. However, attacks against image recognition using adversarial examples (AE) are possible.

1.1 Problem Statement

Adversarial example attacks, sometimes called evasion attacks, add perturbation directly to an image such that a trained CNN will incorrectly classify the image, but to a human observer the image looks virtually indistinguishable from the original. Small changes to the input cause large changes to the output of the CNN. There are many evasion attack algorithms which use different techniques to fool established image recognition algorithms:

- Decision based which samples the image, then uses output to solve for perturbation.
- Gradient based which leverages gradient information to solve for perturbation.
- Score based which optimizes on the model output probability.
- White box wherein the model is known to the attacker.
- Black box where the attacker has limited model knowledge.

An adversary can add perturbation directly to an image or to objects in the real world. For example, researchers have demonstrated that by strategically adding stickers to a stop sign they could cause image recognition models to classify the sign as a yield sign. Adversaries may also fool the model by adding excessive perturbation, but if the perturbation is poorly concealed the attack will be less deceptive and easier to detect. In this paper we assume the attacker will attempt to keep their perturbation subtle to avoid detection and that they will be generating these adversarial examples to fool a fully trained model.

The general problem this thesis will seek to solve is that adversarial example attacks can lead to incorrect results in image recognition, and currently there is no defense robust enough to detect all attacks, virtually guaranteeing errors when a range of different attacks are applied.

If adversarial attack malware can be installed in the image processing software, or at another stage in a data pipeline, before image recognition tasks are performed, the CNN models trained to classify images will have their performance degraded or destroyed. Adversarial attacks pose a problem because if an adversary can implement these attacks in actively used systems without detection, the economic and strategic costs could be very large. If the attacks were to be done subtly, it may take a long time before anyone realizes the system has been attacked and how to fix the issue. To defend against this, one must develop systems to detect if an adversarial attack has been used so that an attack can be quickly recognized and rectified before significant damage is done. The earlier the problem is detected, the easier it is to mitigate the damage, to find the source of the problem, and to prevent the problem from happening in the future.

As the military becomes more reliant on image recognition, we must be sure these systems are robust against attacks. Military systems using automated guidance systems in autonomous vehicles, facial recognition for security, target tracking through cities by security cameras, and recognizing objects from satellite images all need a high degree of certainty they are correctly classifying the target. If a hostile governmental actor can access

and modify images on-the-line this could devastate the affected military capability, while potentially remaining undetected for months. Hence, it is important to develop defenses against such attacks.

Presently, the state of the art in detecting attacks will, at best, only be able to detect one or a tiny number of types of attack with a high degree of confidence. The lack of robustness in detection is a significant problem. Robustness can be defined as how well the model can prevent misclassifications from all types of adversarial attacks. A perfectly robust model detects all attacks, a brittle model detects few attacks. Systems cannot efficiently have dozens of models running to detect if an attack has occurred. Additionally, each new type of attack could be effectively a zero-day attack. Any effective adversarial image detector must be robust in detecting many attacks and should be general enough and robust enough that either new attacks will be detected or it will be substantially harder to create a new attack which remains undetected.

The work of Barton [1] in developing Padnet has shown great promise in this area. Padnet adds to the list of possible object classes one additional class for attacks. For example, with a normal image recognition model, an adversarial example image of a "cat" may be misclassified as a "dog." Training an image recognition model with Padnet would add an 11th class of object which would be labeled the none of the above (NOTA) class. In this case the Padnet image recognition model would see the same adversarial example image of a "cat" and classify the image as "NOTA" instead of as a "cat" or "dog." Any image being classified as "NOTA" could then trigger an alert which would permit further investigation, potentially enabling early detection of a devastating attack against an essential system. Linear interpolation was used to create the NOTA class in the original paper. Mixup and adversarial mixup are augmentation approaches that combine benign and adversarial examples, and have led to improvements in robustness. However, robustness across different attack types still needs to be improved on.

The specific solution proposed in this paper to solve the stated problem is to train a NOTA class by mixing benign samples with various adversarial examples generated from the same benign image. This will be done either by combining different attacks in the same image, or by using SaliencyMix to combine a benign image with a corresponding adversarial example. SaliencyMix is an augmentation strategy which adds the part of an image most salient to its

correct classification to another image and applies mixed labels (soft labeling). The goal in both cases is to cover as much of the multi-dimensional space as possible with the NOTA class, while still correctly identifying unaltered images. By combining many attacks in the training, we hope to greatly improve the robustness of the model against attacks not included in the training. To evaluate the robustness of our defense, we measure the performance of the defense against ten state-of-the-art attacks.

1.2 Research Questions

Other studies have been done on detecting an adversarial attack and have had a high degree of success in detecting if the named adversarial attack had been implemented. However, these have not generalized well into detecting if a different attack occurs which the model had not been trained to handle. Our contribution to the field is to work on methods of generalizing the detection to cover many varied attacks. Specifically, we ask:

1. Can we train a model that reliably categorizes adversarial examples as the NOTA class for multiple types of attacks in our test data?
   Preliminary evidence suggests that multiple attack types can be correctly classified as NOTA, but the reliability and robustness of the models have not been thoroughly tested.
2. Are some attacks more robust against detection than others?
   Answering this gives us further clues into how we can tailor the training of the model to optimize our measures of accuracy and robustness.
3. What impacts do different methods of generating the NOTA class have on the accuracy and robustness of the model?
   Experimenting with different methods of generating a NOTA class and testing the accuracy and robustness of the subsequent model is the primary focus of this thesis and provides substantive answers to the other questions. We find the new methods of generating a NOTA class to offer substantially more robustness than previous methods.

1.3 Thesis Organization

Chapter 1 has briefly introduced the emergent problem of attacks against image recognition. Chapter 2 discusses the fundamentals of image recognition and adversarial attacks against image recognition. This provides useful background information to help with understanding the later topics. Chapter 3 discusses the two methods which will be used to train the PadNet models, 3-Mix and SaliencyMix, which are designed to achieve our research objectives. Chapter 4 gives the results of the experiments. Chapter 5 offers conclusions and suggestions for future work.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 2:
Previous Work – A Survey of Similar Work in the Field

This section discusses previous work which led up to the research in this paper. First is a discussion of how image recognition developed to become an essential technology for many varied industries. Second is a discussion of the development of attacks against image recognition. Third, we will look at the latest methods of defending against attacks, particularly from the PadNet paper on which this research builds. The third section will also discuss the SaliencyMix algorithm which has been incorporated into developing one of the two defense methods.

Machine learning uses algorithms and models to enable computers to learn from data and make predictions on new data without being explicitly programmed with exactly how to make the predictions. The goal is to make correct predictions on new data. A feature is a measurable property of the input data, and the feature space is the set of all possible combinations and variations of these features. Models learn patterns within the feature space, and use these patterns to map input data to the corresponding labels or outputs through a function. When creating a model, the data is divided into training set, test set, and sometimes validation set. The training set is used to train the model by adjusting model parameters or weights to minimize the difference between predicted and actual labels. The test set is not seen by the model during training and is used to evaluate the model's performance on unseen examples, allow us to see if the model generalizes to new data. The validation set is used during training to fine-tune hyperparameters and prevent overfitting.

Hyperparameters control aspects of model training and include number of epochs, batch size and learning rate. An epoch is one pass through of the training data during training. The batch size is how many training examples are included in each iteration during an epoch. The learning rate is how fast the system adapts to new data by adjusting model parameters.

This thesis's research will use the CIFAR-10 dataset, which is well researched and carefully curated dataset [2]. This dataset contains $50,000$ training images and $10,000$ testing images, giving a total of $6,000$ images in each of the ten classes. Those ten classes of objects are

airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. Each image is sized 32x32 and is in color giving a total of 32x32x3 pixel values.

2.1 Image Recognition and Object Detection

Research into computer vision has been ongoing since the 1960s [3] but has seen large breakthroughs with the development of deep neural network (DNN)s and CNNs using back-propagation [4], [5].

2.1.1 Convolutional Neural Networks

Artificial neurons are simple digital models of biological neurons. Each artificial neuron receives digital inputs, performs computations on the inputs, and if an activation threshold is reached it produces a binary output. Deep neural networks arrange the neurons into multiple interconnected layers which allows the network to learn hierarchical representations of data, capturing and modeling of complex information from multiple inputs.

CNNs are deep neural networks used to find useful patterns in high-dimensional data such as images. In CNNs, groups of pixels must be evaluated in relation to other surrounding groups of pixels to identify patterns. Those patterns form maps which combine to determine how an image should be classified. Patterns between pixels can be found by applying predefined convolution kernels or masks which contain predefined patterns. If the filter's pattern matches with the pixel values at a given location on an image, a threshold is reached which contributes to a new layer of data, and it is combined with pattern matches over all the other pixel locations to build a feature map. Figure 2.1 represents how layers of abstraction lead to a classification of elephant from an input image of an elephant by determining increasingly complex features. Through this process a trained model is created which can be used to predict how a new image should be classified. The trained model will have an architecture defining the number and size of layers, the type and arrangement of its layers, and how data moves through the model. It will have weights determining the strength of connections between neurons, and biases which adjust the output of each neuron.

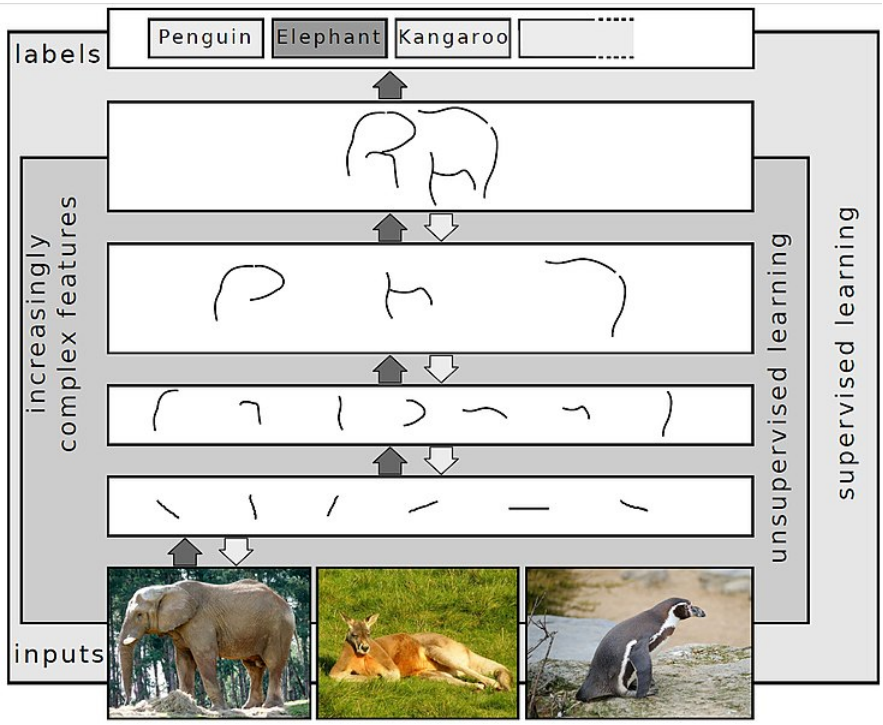

Figure 2.1. Combining data from convolution kernels in a CNN to recognize an object belonging to the elephant class. Source: [6]

2.1.2 Overfitting and Underfitting

Overfitting is when the machine-learning model fits too closely to the data points on which it is trained. This prevents the model from generalizing well to new data. For example, if the model memorizes every image of a cat instead of learning a general pattern for cat, then any new image of a cat will not be recognized as being one of the previously seen images and may be misclassified. Underfitting is when the model does not fit closely enough to the data the model has been trained on. For example, if the cats in a dataset were mostly black and only a few dogs were black, an underfitting model might incorrectly classify all black animals as cats.

2.1.3 Distance Metrics

Four common distance metrics, sometimes called norms, are L0, L1, L2, and L∞. These are different ways of measuring the distance between two vectors or the magnitude of a single vector. For example we can have vectors $a = [4, 1, 3]$ and $b = [3, 2, 5]$. L0 is the highest number of nonzero elements in either vector. L0 example distance is 3. L1 is the absolute value, the sum of absolute differences between the two vectors, sometimes called the Manhattan distance. L1 example distance is $|4 - 3| + |1 - 2| + |3 - 5| = 4$. L2 is the Euclidean distance, the root of the sum of squares. L2 example distance is $\sqrt{(4-3)^2 + (1-2)^2 + (3-5)^2} = \sqrt{6}$. L$\infty$ is the largest absolute value of any difference between the vectors. L∞ example distance is $|3 - 5| = 2$.

2.1.4 Dropout

Dropout is when some number of neurons are removed at random during a part of the training. This enables better regularization. Regularization is where we restrict a model in some way to prevent overfitting. L1 regularization adds a penalty term to the model's loss function based on the L1 norm of the model's weights. It encourages the model to have sparse weights by making the least important weights zero. Dropout is a type of L1 regularization because some features are being ignored. This helps prevent any feature from becoming too important and helps to eliminate the least important features.

2.2 Adversarial Attacks

Adversarial attacks add perturbation to an image, subtly changing the pixel values so that the image looks virtually the same to a human observer, but causing an image recognition model to misclassify the image. If an attacker gains unauthorized access to a system, adversarial attacks can be applied to images by attackers between the time the picture is taken and when the fully trained image recognition model is used to classify the image. An image that has been altered by one of these attacks is an adversarial example. Figure 2.2 first shows a benign image which is correctly classified and then shows adversarial examples generated by different adversarial attacks including DeepFool and Carlini Wagner (CW), with the misclassification each attack caused labeled below the image. If an attacker can change the image processing software to include an adversarial attack or can add an adversarial attack somewhere in the pipeline between the image being captured and

the image recognition model being applied to the image, this could cause the affected company or government millions of dollars or cost people their lives in the cases of medical scans or military intelligence. Because the images appear unchanged to human observers, image misclassifications could go unnoticed for a significant period causing the system's functionality to be degraded, and once the misclassifications are discovered finding the source of the problem could be a long expensive process. There are many types of adversarial attacks. The current state-of-the-art models are only able to reliably detect or prevent a tiny number of attacks. This means entities that wish to bypass defenses may only need to tweak their adversarial attacks or change their attack type to bypass current defenses. Thus, the defenses remains brittle.

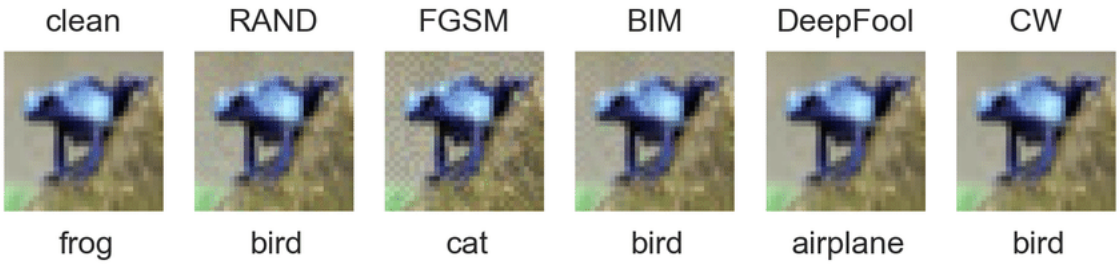

Figure 2.2. Adversarial example samples generated from the "clean" CIFAR-10 image. Source: [7]

Adversarial examples were first reported in a paper which showed that trained image recognition models were easily confused by barely noticeable changes to the image pixel values [8]. Since then new methods have been developed to create adversarial examples in the digital and in the physical world [9], [10]. These adversarial examples pose a threat to the safety of users and the security of the systems these models support.

To defend against attacks, a model must either correctly classify images in spite of adversarial examples, or it must detect adversarial examples and alert those in charge of the system. Previously explored methods of defending against adversarial examples included altering the training data or training procedure to improve robustness through regularization [8], [9], or train an adversarial example class using defensive distillation [11].

It is not safe to assume that object recognition will only ever occur in a closed and trusted environment where attackers will never be able to add code or alter the data pipeline. If even 1% of attacks get through, this can be devastating, causing car crashes for autonomous vehicles, misdiagnosis of life-threatening conditions, bad intelligence for the military, or economic harm through mistakes in agricultural systems. An adversary disabling a system would be noticed quickly and fixed, but using adversarial attacks could allow an adversary to degrade function, with the attack potentially going unnoticed and causing damage over an extended period.

The objective of the attacks being defended against in this paper is to alter pixel values, which are the inputs to the classifying model. The altered inputs cause a misclassification without substantially altering the image, such that a human checking the image would not suspect that the image has been attacked and would believe the misclassification is the result of a faulty model.

A white-box attack is where the attacker knows the architecture and parameters of a classifying model which lets them tailor their attack to maximize their effectiveness. With a black-box attack, the attacker lacks access to this information so attacks must be more general and are less likely to succeed. Nevertheless, black-box adversarial examples will often cause misclassifications of models which do not defend against the attack used to generate the adversarial example. A targeted attack specifies the class that the adversarial example should be misclassified as, while an untargeted attack succeeds if the adversarial example is misclassified as being in any class that is incorrect. The research in this paper will use untargeted black-box and untargeted white-box attacks.

Another way to attack the accuracy of image detection models is through a poisoning attack. This attacks the data the model will be trained on, or embeds malicious functionality into a model that will be used by someone else. If data comes from social media or some other untrusted source, an adversary can get wrongly labeled data or adversarial examples into their target's dataset. The adversary can cause targeted misclassification, neural trojans and other attacks [12]–[16].

2.2.1 Adversarial Robustness Toolbox

The adversarial robustness toolbox (ART) is an open-source Python library of attacks on machine learning and defenses against those attacks made available to the research community [12], [17]. This library allows users to benchmark novel attacks or novel defenses against current attacks and defenses. The library has 73 different attacks in four main categories: evasion attacks which are our focus, extraction attacks, inference attacks, and poisoning attacks. Carlini L2 [18], deepfool [19], and square attack [20] are used to generate the adversarial examples from which we create the our augmented datasets that are then used for training our models. To show the robustness of our defense, we evaluate it against the following state-of-the-art attacks implemented in the ART library: auto attack [21], projected gradient descent (PGD) [22], auto-PGD, Carlini L-2, Carlini L∞ [18], deepfool, fast gradient sign method (FGSM) [23], elastic net [24], and square attack. ART is maintained by IBM.

Normally in image classification a decision boundary occurs between each class, and the goal of each attack in the ART is to tweak pixel values so that humans barely register any change has been made, but the altered image's classification has been pushed from one side of a decision boundary to the other, so that an image classification model now misclassifies the image.

All the attacks in Sections 2.2.2–2.2.6 are in ART and are used to create adversarial examples which are either used to build the datasets on which our defensive models are trained, or to test the trained models. Except for Square attack (Section 2.2.3) which is a black-box attack and AutoAttack (Section 2.2.4) which uses Square attack, the rest of the attacks in Sections 2.2.2, 2.2.4–2.2.6 are white-box attacks.

2.2.2 Projected Gradient Decent

FGSM is an adversarial attack which adds perturbation to input data using the sign of the gradient of the loss function, scaled by a small value of ϵ [9]. ϵ is a value $[0 - 1]$ used to control the magnitude of perturbations added when creating an adversarial example. A smaller value of ϵ means less perturbation can be added. Projected gradient decent (PGD) is an effective way to find adversarial examples which builds on FGSM by adding multiple steps [22]. In each step PGD computes the gradient of the loss function with respect to the input and alters the input in the direction of the gradient while still bound by constraints.

These examples have their total perturbation limited (bounded) using the L2 or L∞ norms. For example, when the L2 norm is used, ϵ represents the maximum Euclidean distance between original and perturbed inputs.

2.2.3 Square Attack

Square attack is a black box attack with norm bounded perturbations [20]. Instead of relying on local gradient information it uses randomized search, so gradient masking does not affect it. In each iteration localized square-shaped updates are made at random positions to create perturbations approximately at the boundary of the feasible set. This attack can outperform some white box attacks.

2.2.4 Auto-PGD and AutoAttack

PGD is a popular method for testing adversarial robustness, but can fail for various reasons, including having a fixed step size and only using one type of loss measurement causing robustness to be overestimated. Croce and Hein [21] discuss Auto-PGD and AutoAttack which expand the capabilities of PGD. Auto-PGD removes the fixed step size problem by adding a momentum variable and at fixed intervals checks if the step size should be halved. This allows the algorithm to converge on a better local optimization. AutoAttack combines two variations of Auto-PGD with FAB [25] and Square Attack [20] to form an ensemble attack with attacks that generalize well across models and datasets.

2.2.5 Carlini L2 and L∞

Carlini Wagner (CW) attacks use either L0, L2, or L∞ distance metrics to create different adversarial attacks designed to find the least amount of perturbation which is still effective in causing misclassification [18]. L2 and L∞ are norms which measure the magnitude of matrices or vectors as discussed in 2.1.3. CW L2 attacks use the Euclidean norm to maximize distance while minimizing total perturbation, and L∞ maximizes on change to any pixel in the image while minimizing total perturbation. Each attack uses a function to minimizes the total distance between the original x and the adversarial example x' for all pixel values in the range [0,1].

2.2.6 JSMA

Jacobian saliency map attack (JSMA) forces misclassification by iteratively perturbing the parts of the image that are salient to misclassifying it as a new target class [26]. This iterative perturbing continues until a stopping condition is met, usually when the decision boundary has been crossed so that a model is likely to misclassify the image. Gamma measures what percentage of the features may be perturbed. For an image with 3072 pixel values ($32 * 32 * 3 = 3072$), a gamma of 0.1 means a maximum of 307 pixel values may be altered. Theta is the amount of perturbation that can be introduced at each modifiable feature per step.

2.3 Defending Against Attacks

This paper will build on previous work on mitigating evasion attacks on deep neural networks from Cao and Gong [27] which studied a general mitigation strategy for adversarial examples. This paper also builds on the work of Klingner et al. [28] which showed that it is possible for a CNN to detect if a certain type of evasion attack was used on an image.

The primary strategies for defending against attacks are runtime detection of adversarial examples and model hardening. Runtime detection seeks to classify adversarial example images as attacks on the model and alert those monitoring the system. This could be accomplished by generating adversarial images for an attack and training a model to recognize that attack, or it could use a more complicated process to determine if an attack has taken place [29].

Model hardening seeks to train models that will not be fooled by AEs and which will correctly classify the image regardless of whether the image has been modified by an attack. This can be done by generating AEs and training the model to classify them correctly [30].

2.3.1 Mixup

Mixup is a data augmentation defense approach which creates new training examples by linearly interpolating between two benign images, and between their corresponding labels using soft labeling [31]. This linear behavior reduces the amount of undesirable oscillations when making predictions outside the training examples. By blending features and labels of

different samples, a new model trained on this data learns more generalized representations improving robustness.

2.3.2 Adversarial Vertex Mixup

Adversarial Vertex Mixup is similar to Mixup but instead of mixing two benign images, a benign image is mixed with an adversarial example [32]. Additionally, instead of having the adversarial example perturbations be a set distance from the benign input, a scaling factor is multiplied with the distance of the perturbation from the source image to generate additional adversarial examples. This helps to reduce overfitting on adversarial features. Soft labeling with a label smoothing function is also used. The new model trained on this augmented dataset adjusts where decision boundaries are set between classes to be more robust against the attacks on which it is trained. Adversarial vertex mixup is among the most robust defenses currently available.

2.3.3 PadNet

The PadNet paper my research builds on is a form of runtime detection [1]. PadNet is the idea that rather than having a fine line for the decision boundary separating each class (see Figure 2.6), we can create a much wider border similar to a demilitarized zone. Boundary padding is an augmentation strategy which creates a padding class that acts as this demilitarized zone between benign classes. Anything in this padding region falls outside the correct classification areas and is labeled as an adversarial example. In some areas the boundary region between classes will be wider than in other areas. If there are n classes, this border area becomes the $(n + 1)$th class, and we assert that images falling within this widened decision boundary border area are likely adversarial attacks. Some parts of that border can be much wider than other parts because much of the information in the image has no value to determining its classification, and the decision boundary in these areas is somewhat arbitrary. The padding class between true classifications is termed the NOTA (none of the above) class. Figures 2.7 and 2.8 illustrate a simplified example of the concept. Figure 2.7 shows adversarial examples being generated and labeled NOTA. Figure 2.8 shows a new model trained on this data with a NOTA class boundary padding region.

Similar to mixup, adversarial mixup is an augmentation technique which mixes a benign sample x with its adversarial example counterpart x' and labels the resulting sample as

belonging to the NOTA class. Pixel values from the benign image (such as the clean image in Figure 2.2) are combined and averaged with the pixel values from one of the corresponding adversarial examples to create an adversarial mixup image.

Ordinarily, adversarial examples will push just enough past the decision boundary to get a false classification, but by adding the NOTA class, getting just past the old decision boundary will put the adversarial image within the NOTA class and thus lead to the image being labeled a potential adversarial example, which could trigger an alert to someone monitoring the system to investigate the image or to investigate the system if enough images are falling within the NOTA class. It will also make adversarial attacks harder to create because for an attack to get past the NOTA class to an incorrect classification requires the addition of significantly more perturbation. This added perturbation is likely to be much more noticeable to a human observer which forces an attack to be more easily discovered. A human could check the raw image data from the source, compare it to the image analyzed by the model and easily see that the image has been attacked, or they may simply observe the amount of noise in the image and recognize it as an attack. The NOTA class region is in multi-dimensional space rather than on a simple 2D or 3D graph, so the border can cover complex multi-dimensional terrain which is vulnerable to attacks which are too subtle for human eyes to easily notice. Figure 2.3 shows a t-SNE visualization of data sparse regions between classes. The t-SNE visualization in Figure 2.4 shows how the addition of a padding class, generated from adding Gaussian noise to benign images, populates the high dimensional space between classes. Figure 2.5 shows a t-SNE of how training the NOTA class using adversarial mixup samples fills in the data sparse regions near benign samples.

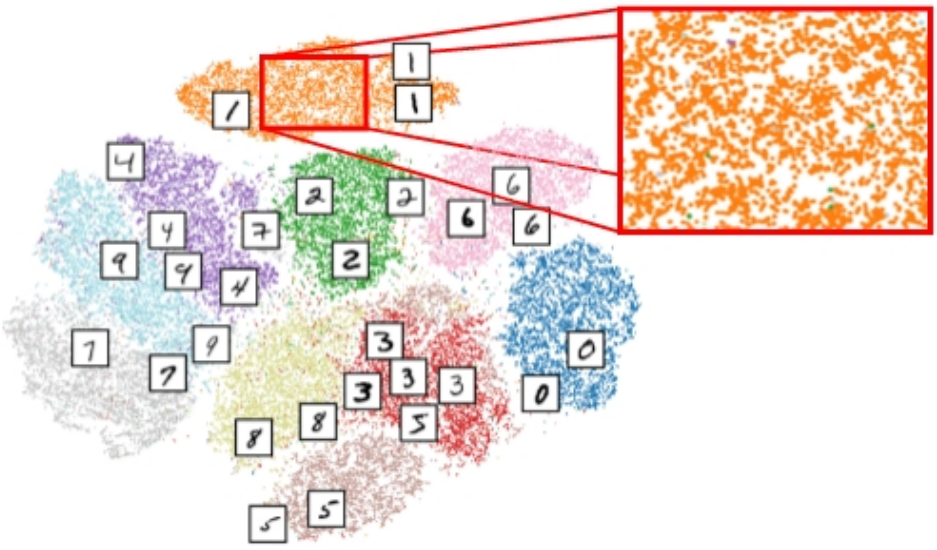

Figure 2.3. t-SNE visualization of MNIST classes before Boundary Padding or Adversarial Mixup. Source: [1]

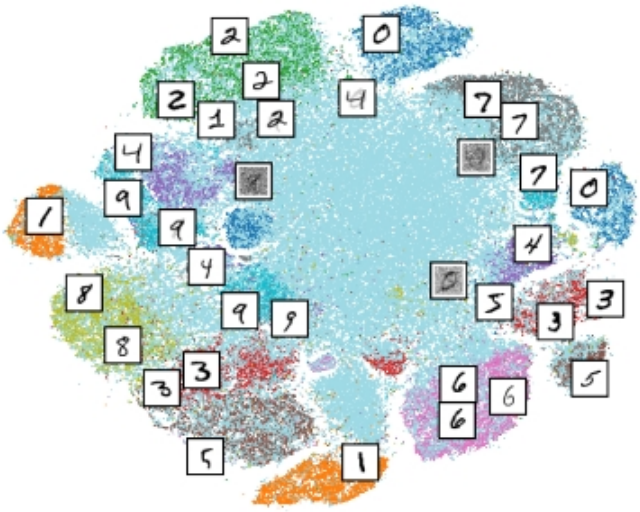

Figure 2.4. t-SNE visualization of MNIST classes after Boundary Padding. Source: [1]

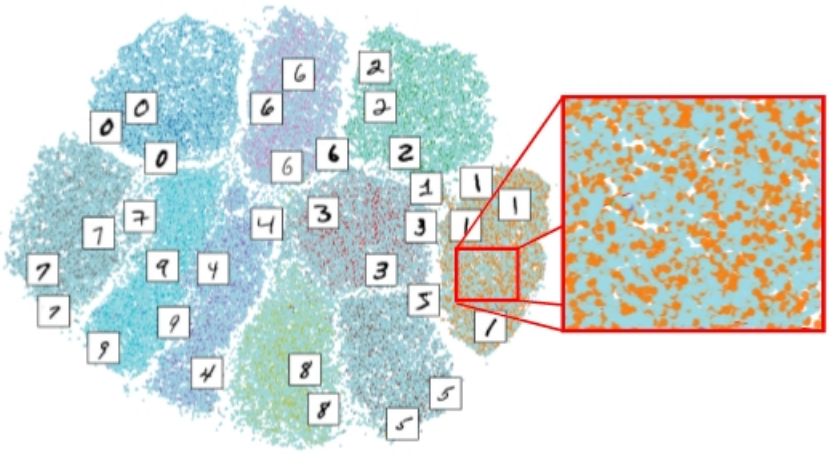

Figure 2.5. A t-SNE visualization of MNIST classes after Boundary Adversarial Mixup. Sparse regions near benign samples within the manifold are filled in with Adversarial Mixup samples. Source: [1]

The PadNet research has shown promising results by providing a more robust defense against attacks, but the research has focused on developing the NOTA class with individual attacks (using PGD to create the NOTA class, or using Carlini Wagner L2, or using Deepfool to create the NOTA class). In Chapter 3 of this paper, we discuss expanding this idea from mixing two images (benign sample x and adversarial example counterpart x') to mixing a benign image with n-number of adversarial examples generated using different attacks on the same benign image where n can be any number.

My research in this thesis studies two methods of combining attacks to create a NOTA class. Both methods use Deepfool, Carlini Wagner L2 and Square attack to build NOTA classes which should be more resistant to attacks than when building the NOTA class using any single attack. This should ultimately provide a model which is robust against more types of attacks. Figure 2.9 illustrates that the goal of adding additional types of attacks is to widen the boundary padding creating a defense which is more robust against attacks. Figure 2.10 illustrates a what we seek to avoid, wherein using only one type of adversarial attack causes the boundary padding to concentrate in certain areas, leaving a decision boundary in other areas which is vulnerable to other types of attack.

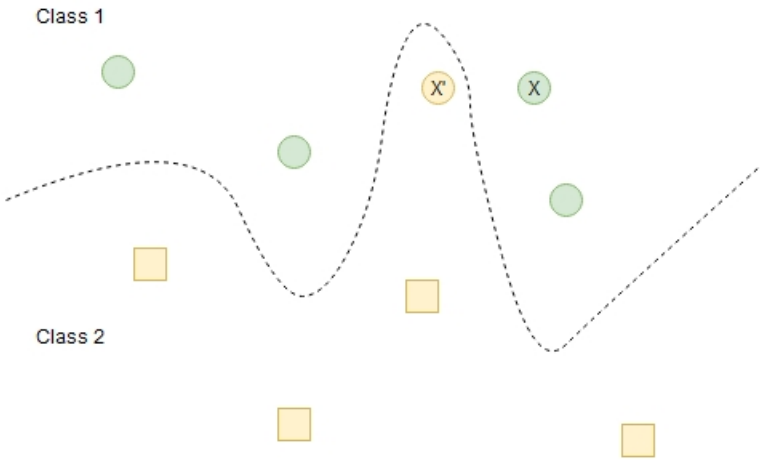

Figure 2.6. An adversarial example being misclassified.

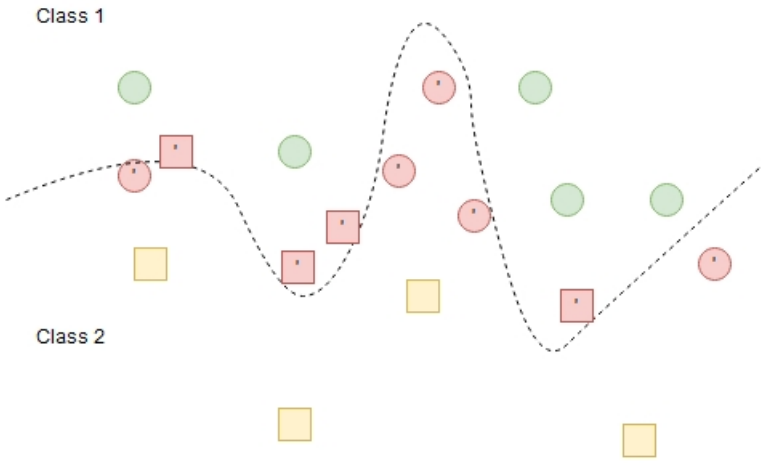

Figure 2.7. Many adversarial examples being generated which all cross the decision boundary.

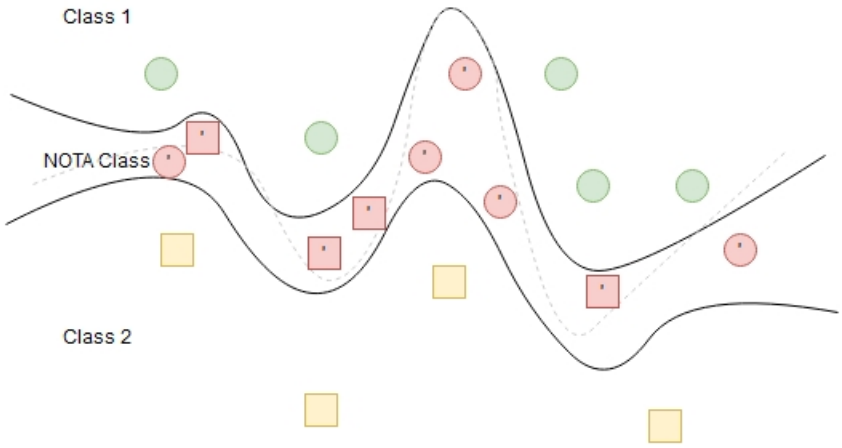

Figure 2.8. A NOTA class drawn around the initial adversarial examples.

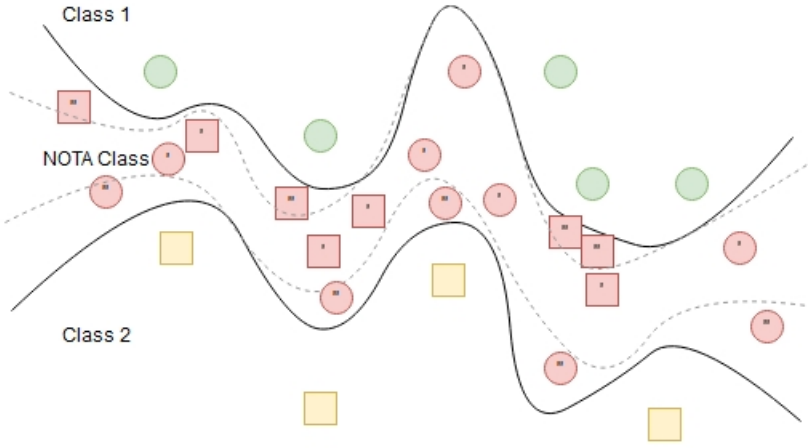

Figure 2.9. New adversarial examples included in the training to widen the NOTA class.

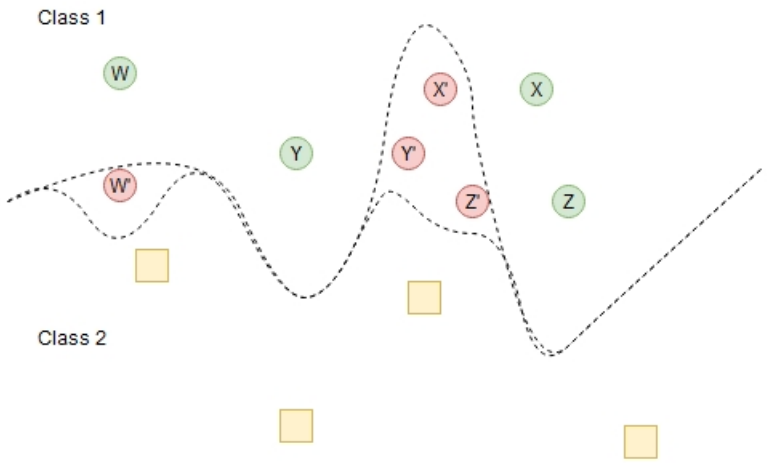

Figure 2.10. With only one type of adversarial attack there is a risk that the model will form pockets of NOTA class, leaving unprotected regions with only a decision boundary between true classes, which may be vulnerable to other types of attacks.

2.4 Data Augmentation Algorithms

Data augmentation artificially increases the diversity and size of the dataset by modifying or transforming copies of existing samples. The two algorithms in Sections 2.4.1 and 2.4.2 are relevant to the research done in this paper. These algorithms are data augmentation strategies used to regularize the model and reduce overfitting. They have an additional benefit of providing more robustness against adversarial examples.

2.4.1 CutMix

CutMix [33] replaces a section of an image with a section of another image. The label for the new image is adjusted to give weight to each label proportional to the number of pixels each image represents. This can lead to more robust results because it helps expand the dataset and prevents the model from being overly reliant on specific features. However, there is no

guarantee that relevant aspects of the image will be included in the new image because the area of focus is not always in a specific part of an image. It is possible that the relevant part of one image could be replaced with an irrelevant part of another image, creating a training example image which is actively detrimental to the training because it teaches the model to recognize irrelevant features of the image. For example, if an indoor cat image and a dog image get wrongly combined, the image could be labeled as 40% cat, 60% dog, but only show 40% couch, 60% grass. Then, if a cat is pictured near grass, it could be misclassified as a dog.

2.4.2 SaliencyMix

SaliencyMix is from a well know research paper, and has been shown to improve regularization and slightly increase robustness [34]. It improves on CutMix for model training outcomes. SaliencyMix maps the part of the image to which the image recognition model gives the highest probability of determining the image's classification. This part of the image is the most salient part. A rectangular area from the most salient part of an image replaces a rectangular area of the same size from another image with the label getting a proportional weight from each original image (soft labeling). Figure 2.11 demonstrates this process. Now the most relevant part of the cat, the head and some body, will replace the corresponding section removed from the dog image. This method of data augmentation has been shown to improve model prediction accuracy.

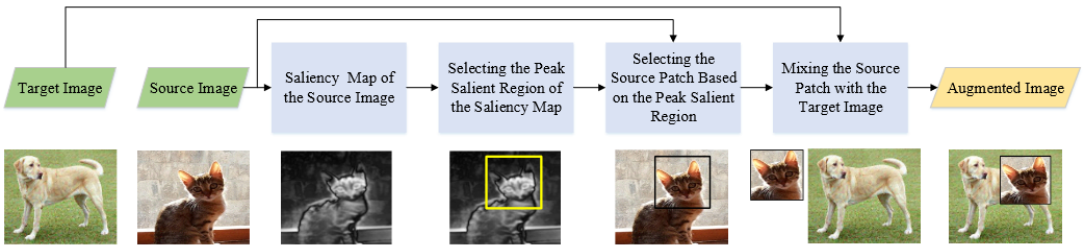

Figure 2.11. SaliencyMix data augmentation determining and extracting the salient part of the source image and patching it onto the target image. Source: [34]

In this paper's research, the most salient part of an adversarial example image will replace the removed section of the other image. This data augmentation strategy helps to improve regularization in models which detect adversarial images by helping to improve learning appropriate feature representation.

2.5 Threat Model

We assume the attacker has gained sufficient access and will use adversarial examples to attack a trained deep neural network. The attack is successful if and only if the trained model classifies the adversarial example x' as a class other than its benign class y or the padding class y_{NOTA}.

$$F(x') \neq y \wedge F(x') \neq y_{NOTA}$$

The attacker may use any attack mentioned in this paper, black-box or white-box. We assume that the attacker will seek to degrade the performance of the system and prevent their attacks from being detected. If an attack is discovered, steps would be taken to prevent future attacks and deny the attacker system access. If the performance of the system decreases, a human will investigate, so we assume the attacker will seek to keep from using obviously modified images in their attack to reduce the likelihood of their attacks being discovered. There are very obvious attacks such as patch attacks but these would be easily recognized when investigated. Although there are some attacks using high perturbation, we evaluate on small perturbation which is a more challenging problem.

Nation-states could potentially use these attacks to degrade the automated analysis of spy satellite images. External threats such as hackers and insider threats such as manipulated or disgruntled employees could manipulate images in the data pipeline. Perturbations can be replicated in the real world to create camouflage, or to prevent recognition by surveillance systems, or malicious actors could use graffiti to cause self-driving cars to malfunction leading to deaths.

If our defense detects an attack it could alert the person monitoring the system such as the system administrator and ignore or reject that classification until a human makes a decision. In the case of an autonomous vehicle it could take precautions, alert the driver to the issue

and request they assess the situation and take control, and simultaneously it could alert the system administrator and potentially local authorities to investigate.

CHAPTER 3:
Methodology

In this chapter the motivation behind the defense is discussed, the framework chosen to execute it, and the methodology of the algorithm. The principle is to create a new class, in addition to the existing object classes, which represents the space of all attacks. This new class we call the NOTA (None of the Above) class.

3.1 Design

Our defense, the detection of the adversarial examples, relies on the following general steps:

1. Generate adversarial examples which are variants on true examples in the CIFAR-10 dataset.
2. Combine the adversarial examples with normal examples using N-Mix or SaliencyMix to create augmented datasets with the NOTA class label.
3. Train models using the augmented datasets.
4. Test the accuracy of the new models to ensure the original classes are still recognized with high accuracy. Evaluate the performance of the models against many varied adversarial examples, generated from different classes of adversarial attacks, to determine how robust the model is against varied attacks.

The first step was to create a dataset of images which have been attacked with adversarial example attacks using the Adversarial Robustness Toolbox (ART) [12], using the Carlini Wagner L2 [18], deepfool [19], and square attack [20] algorithms. The adversarial examples will be combined with the benign images using either the N-Mix or SaliencyMix algorithms, described in Section 3.2, to generate new datasets on which the models will be trained. A set of benign images with no perturbations will also be kept for training in a 1:1 ratio with the NOTA labeled images.

N-Mix combines adversarial examples from multiple types of attacks to create a composite of adversarial examples in each NOTA labeled image. In these experiments, N was always 3, but N could represent any number of attacks. This method, it was hoped, would generalize to

recognizing a broader range of adversarial examples than other detection methods. In another method we combined adversarial examples with benign images using SaliencyMix to try to create a tighter boundary around the correctly classified classes, hopefully improving detection accuracy. By creating small perturbations bounded by a small value of ϵ (0.031) in the adversarial examples we create a tighter boundary around the true classes.

Next, a CNN was trained on the augmented dataset that includes benign samples along with adversarial mixup samples. By training on the augmented dataset, we show during test time that the models maintain high benign accuracy while being able to detect AEs by classifying them as NOTA. The CNN will use graphics processing units (GPUs) to ensure fast processing of the data.

3.2 Algorithms

Each of the following algorithms will be used on the CIFAR-10 dataset because it is a well-researched dataset, thus making comparisons to other models much easier.

3.2.1 N-Mix

The idea behind N-Mix is taking the pixel values from a benign image and from n number of adversarial examples derived from the same base image, each modified with a different method from the ART library, and combining them with a weighted average. In our experiments, we tested 3-Mix algorithms where $n = 3$. The three adversarial examples combined with one corresponding benign image. For example, an image of a cat was combined with three copies of the same image of the cat which had each been attacked using Carlini Wagner L2, Deepfool, or Square attack. A weighted average was taken for each pixel value to create the new adversarial examples.

For creating N-Mix adversarial mixup samples, we first start with a benign sample x. Then, we create n adversarial examples from n different attacks denoted by $\{x'_1, x'_2, ...x'_n\}$. The N-Mix adversarial mixup sample is created by combining the benign sample with the n adversarial example counterparts.

We set a weight to the contributions of the benign and adversarial examples to cause the model to have linear behavior in-between benign samples and adversarial mixup samples.

To add this extra stochasticity, a variable α is drawn from the uniform distribution. Then, β is set to $1 - \alpha$. The weighted N-Mix sample is then given by:

$$x_{n-Mix} = \alpha x + \frac{\beta}{n} x'_1 + \frac{\beta}{n} x'_2 + \ldots + \frac{\beta}{n} x'_n \tag{3.1}$$

This we refer to as fixed-beta because the weight β is distributed equally to each adversarial example. In another experiment we added an element of randomness to what percentage of the weight β each adversarial image contributed. The weights w_1 through w_{n-1} are a random value between $\frac{1}{n-1}$ and $\frac{1}{n+1}$. To ensure the weights always add up to one, the last weight w_n is given by:

$$w_n = 1 - (w_1 + w_2 + \ldots + w_{n-1}) \tag{3.2}$$

Then the formula for generating variable-beta N-Mix adversarial examples is given by:

$$x_{n-Mix} = \alpha x + \frac{\beta}{n} w_1 x'_1 + \frac{\beta}{n} w_2 x'_2 + \ldots + \frac{\beta}{n} w_n x'_n \tag{3.3}$$

3.2.2 SaliencyMix

SaliencyMix will take the most salient part of the benign image (the part containing the most information which the CNN uses to determine its class) and replaces that section with the same section from an adversarial example. SaliencyMix by default also uses soft labeling which gives two weights for the label instead of one. One weight is for the proportion of the image that comes from the benign sample and one weight for the portion of the image that comes from the adversarial sample. For example, an image from class 2 with a NOTA labeled adversarial example may be labeled [.0 .0 .558 .0 .0 .0 .0 .0 .0 .0 .442]. The model generated with these adversarial examples we termed SaliencyMix multi-label. We also ran an experiment with a single label for SaliencyMix adversarial samples where the label was always set to be 100% in the NOTA class ([.0 .0 .0 .0 .0 .0 .0 .0 .0 .0 1.]).

SaliencyMix first uses the $cv2.saliency.StaticSaliencyFineGrained\_create()$ function to determine the salient part of the image and generate a map of that part so that part of the benign sample can be replaced. Let $I_s \in \mathbb{R}^{W \times H \times C}$ be the source image, W is width, H is height, and C is color. y_s is where patch will be cut. Saliency map detection is represented as $I_{vs} = f(I_s)$ where $I_{vs} \in \mathbb{R}^{W \times H}$ is the visual saliency map of the given source image I_s as shown in Figure 2.11 with function $f()$ representing the saliency detection model. Then we search for a pixel $I_{\text{vs}}^{i,j}$ in the saliency map that has the maximum intensity value. The i, j represent the x and y coordinates of the most salient pixel and can be found as $i, j = \arg\max(I_{vs})$.

A patch is selected, either by centering on the $I_{\text{vs}}^{i,j} - th$ pixel if possible, or keeping the $I_{\text{vs}}^{i,j} - th$ pixel on the selected patch to ensure the patch is selected from in the target and not from the background. Patch size is determined by ration λ. Once the patch is determined on the source it must be applied to the target. Let $I_t \in \mathbb{R}^{W \times H \times C}$ be the target image with label y_t. I_s and I_t are mixed to create augmented image I_a with label y_a. The mixing is defined as $I_a = M \odot I_s + M' \odot I_t$ where $M \in \{0, 1\}^{W \times H}$ represents a binary mask, M' is the compliment of M and $\odot$ represents the element-wise multiplication. M is set to 1 and M' set to 0 to create the training sample. The target image is an AE and the target label is the NOTA class. Multi-label which mixes the labels based on the size of the patches uses: $y_a = \lambda y_t + (1 - \lambda) y_s$ while single-label uses: $y_a = y_t$ where y_a is the label for the augmented sample and λ is the combination ratio.

CHAPTER 4:
Results

In this chapter the results of the 3-Mix NOTA aware models and the SaliencyMix NOTA aware models are discussed. Comparisons are made between the models and some common defense models as benchmarks. We are developing defenses against attacks and therefore want the accuracy of predictions to be high and the attack success rates to be low. The lower the rate of successful attacks, the more robust the model is in defending against those named attacks. The models were trained using adversarial examples from three types of attack and were tested against ten types of attacks to determine how well the defense generalizes.

4.1 Experimental Setup

For each of the 3-Mix models 50,000 adversarial examples from each of the three attacks were combined with 50,000 benign images to generate 50,000 3-Mix adversarial examples, each labeled as belonging to the NOTA class. The dataset used to generate for each of the 3-Mix models had 50,000 benign and 50,000 3-Mix images. In making comparisons, we wanted to ensure differences between SaliencyMix and 3-Mix were not due to choice of adversarial attack used to generated the SaliencyMixed images, and training time was prohibitive for training and testing many SaliencyMix models (taking over a week each). The decision was made to generate a dataset containing SaliencyMixed adversarial examples from each of the same three attacks as used for 3-Mix. A total of 150,000 SaliencyMixed images were generated, with 50,000 benign being mixed with 50,000 adversarial examples from each of the three attacks. Sampling from the 150,000 images to select 50,000 could have introduced issues with some classes potentially being poorly represented for some attacks, so we elected to use all 150,000 SaliencyMixed adversarial examples. 50,000 benign examples with each of the ten classes having 5,000 samples compared to 150,000 NOTA class samples could have led to problems with an imbalanced dataset. Three copies of each benign image were used to have a total of 150,000 SaliencyMixed adversarial examples and 150,000 benign images. In both cases there is a ratio of 1:1 benign images to adversarial examples.

To evaluate the 3-Mix and SaliencyMix data augmentation strategies for generating a NOTA aware model, we set up experiments to create variations of these datasets and trained a model for each.

After the datasets have been generated using SaliencyMix and N-Mix, training is consistent for both datasets. The TensorFlow model is trained for 1000 epochs with the option of early stopping [35]. The batch size is 128, the learning rate is 0.001. We use a wide residual network (WRN) architecture and SAM optimizer as part of our hyperparameters which are state of the art methods that give good results in image classification [36]. WRN solves two gradients to find the best area to converge on. The SAM optimizer in turn uses the Adam optimizer with a beta of 0.9. The model is trained using categorical cross-entropy and mean loss is used for the loss function. A wide residual neural network (ResNet) model is used for the CNN with width 6, depth 12, and 11 classes [37]. The input shape of [32, 32, 3] reflects the size of images in the CIFAR-10 dataset. The two models used for comparison (Section 4.2) were trained using the same architecture, optimizer, and other hyperparameters to ensure differences in results were due to algorithms rather than of differences in the architecture.

The trained models were then evaluated. Each model's accuracy in correctly classifying new benign images was checked. The adversarial attacks used in evaluation were PGD, AutoPGD, AutoAttack, Square attack, Carlini L2, Carlini L∞, deepfool, elasticnet, boundary attack, and eight versions of JSMA with different sets of parameters. One hundred adversarial examples were generated for each attack and the robust accuracy and attack success rate was calculated for each model on each attack. Trained models of mixup and adversarial vertex mixup were evaluated using the same method so the new models could be compared with state-of-the-art models with industry leading results.

With one exception, accuracy refers to classification accuracy, the proportion of benign images which a model correctly classifies. The exception is "robust accuracy" which specifically refers to how many of the adversarial examples were correctly classified as belonging to the base class y instead of a false class with adversarial label y'. Attack Success Rate (ASR) is the measure of what percentage of attacks were successful. For example if a model is fooled by 35 out of 100 adversarial examples, the ASR is 35%. It is desirable to have accuracy and robust accuracy at 1 (100% accuracy) and Attack Success Rate at 0. We expect

our robust accuracy to be low because our models are detectors which should predict y_{NOTA} instead of the base class y when attacked.

4.2 Comparison Model Results

First, to get a benchmark against which results can be compared, two previously trained models representing defenses considered strong (Mixup and Adversarial Vertex Mixup) were evaluated using the same evaluation program with which the SaliencyMix and 3-Mix models were evaluated. First the program evaluates the accuracy with which each model successfully classifies benign images. Then the program creates adversarial examples using ten different attacks and tests the rate at which attacks are successful in fooling the models, the rate at which these attacks cause each model to incorrectly classify the adversarial examples. The last attack, JSMA, is run 8 times using two different values of theta (0.1 and 0.01) and 4 different values of gamma (0.1, 0.2, 0.5, and 1).

On benign samples Mixup had an accuracy of 0.92 and Adversarial Vertex Mixup had an accuracy of 0.87 in correctly classifying the images. Image classification accuracy on benign images for all evaluated models is in Table 4.1. In the following tables (Table 4.2, Table 4.3, and Table 4.4) "robust accuracy" is a measure of how many images were correctly classified as belonging to their original class in spite of the image being attacked. "attack success rate (ASR)" is a measure of how often the model was fooled by the attack and gave an incorrect classification. It is desirable to have robust accuracy at 1 (100% accuracy), and Attack Success Rate at 0 meaning all attacks failed.

These models correctly classified some images that had been attacked as seen in the robust accuracy column of Table 4.2, but on average did a poor job of defending against most attacks. Not including JSMA the average attack success rate was 0.749 for Mixup and 0.781 for Adversarial Vertex Mixup. Including JSMA with $theta = 0.1$ and $gamma = 0.1$ the averages were 0.771 for Mixup and 0.777 for Adversarial Vertex Mixup. Figure 4.1 displays the ASR for each attack against the models. Figure 4.2 displays the ASR for the JSMA attack variations against the models.

Table 4.1. Table of accuracy in correctly classifying benign images

Model	Accuracy
Mixup	0.92
Adversarial Vertex Mixup	0.87
3-Mix variable beta	0.89
3-Mix fixed beta	0.87
SaliencyMix single label	0.83
SaliencyMix multi-label	0.88

Table 4.2. Table of results for comparison models, Mixup and Adversarial Vertex Mixup (t=theta, g=gamma)

Attack	Mixup		Adversarial Vertex Mixup	
	Robust Accuracy	ASR	Robust Accuracy	ASR
autoattack	0	1	0	1
PGD	0.33	0.67	0.27	0.73
square	0.81	0.19	0.45	0.55
carliniL2	0.02	0.98	0.04	0.96
carliniL∞	0	1	0.39	0.61
deepfool	0	1	0.09	0.91
auto-PGD	0.97	0.03	0.57	0.43
elasticnet	0	1	0	1
boundary	0.13	0.87	0.16	0.84
JSMA t.1 g.1	0.03	0.97	0.26	0.84
JSMA t.1 g.2	0.02	0.98	0.1	0.9
JSMA t.1 g.5	0.01	0.99	0.01	0.99
JSMA t.1 g1.0	0	1	0.01	0.99
JSMA t.01 g.1	0.1	0.9	0.24	0.76
JSMA t.01 g.2	0.07	0.93	0.15	0.85
JSMA t.01 g.5	0.07	0.93	0.02	0.98
JSMA t.01 g1.0	0	1	0.01	0.99

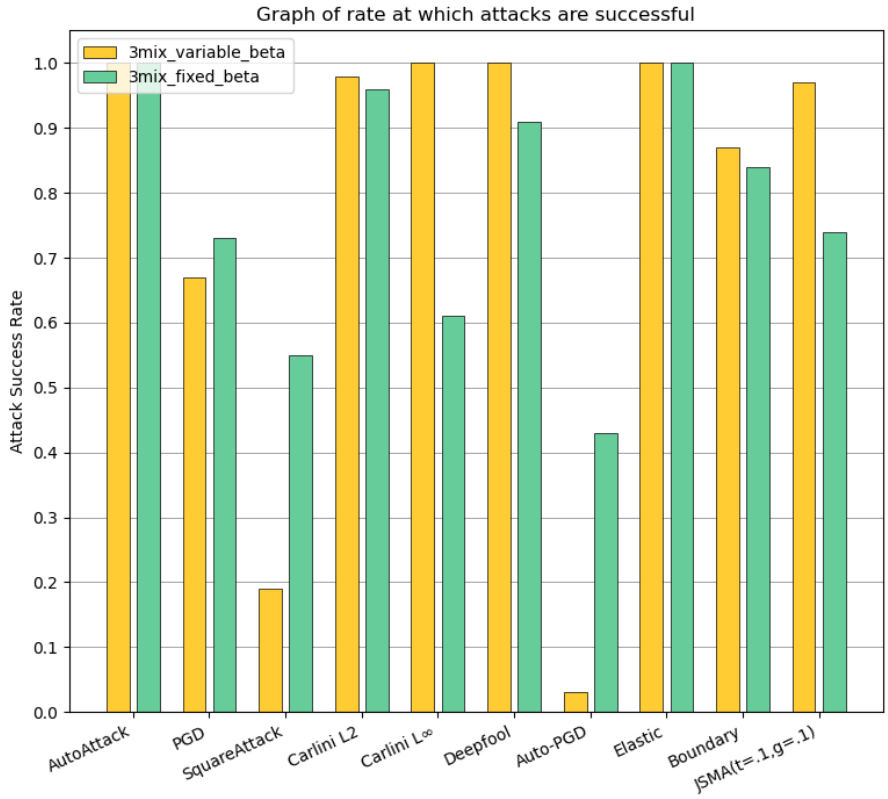

Figure 4.1. Graph showing how successful each attack was in fooling the mixup and adversarial vertex models

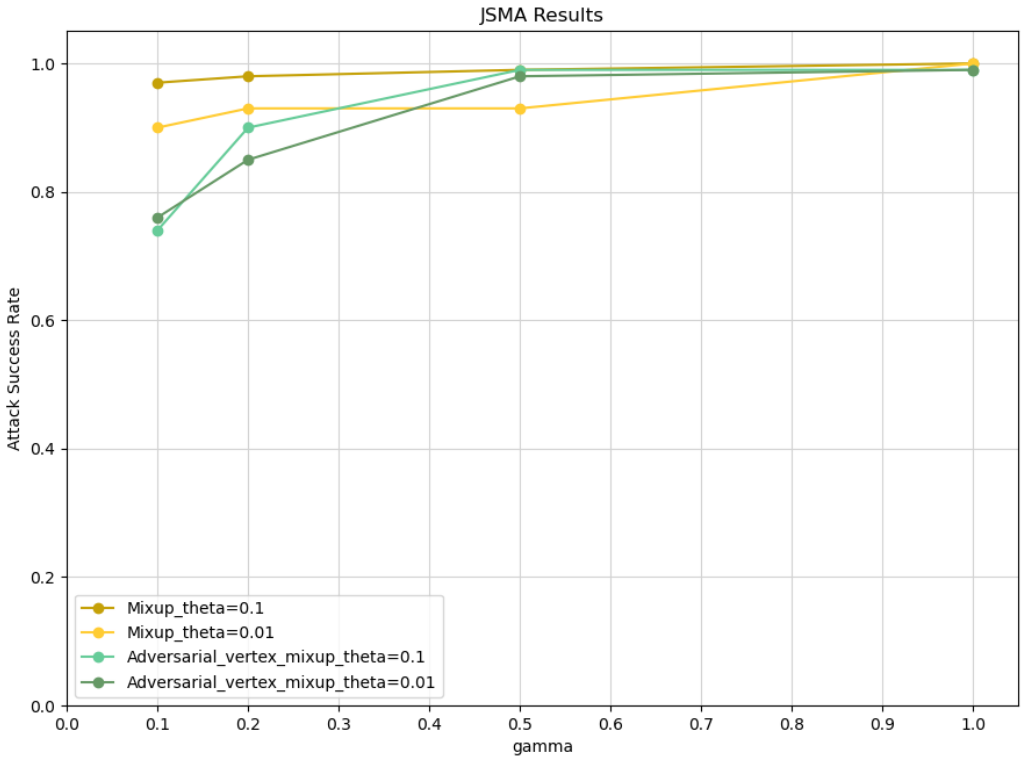

Figure 4.2. ASR for JSMA attacks on the comparison models

4.3 SaliencyMix Results

Table 4.3 displays the results for the SaliencyMix models. Figure 4.3 displays the ASR for each attack against the models. Figure 4.4 displays the ASR for the JSMA attack variations against the models.

Across all the attacks, excluding JSMA, the average attack success rate for single-label was 0.344, and for multi-label it was 0.313. Including JSMA with $theta = 0.1$ and $gamma = 0.1$ the averages were 0.405 for SaliencyMix single-label and 0.374 for SaliencyMix multi-label. While there were significant differences in how successful each defense was against each attacks, the averages across the ten attacks were very similar. These scores show a significant reduction in attack success rates when contrasted with the comparison models. Figure 4.5

shows the SaliencyMix results next to the Comparison model results. The JSMA results are also significantly improved when contrasted with the comparison models.

Table 4.3. Table of results for SaliencyMix (t=theta, g=gamma)

Attack	SalMix single label		SalMix multi-label	
	Robust Accuracy	ASR	Robust Accuracy	ASR
autoattack	0	0.11	0	0.96
PGD	0.15	0.07	0.01	0.67
square	0.19	0.07	0.16	0.2
carliniL2	0	0.59	0.06	0.16
carliniL∞	0	0.59	0	0.16
deepfool	0.04	0.44	0	0.19
auto-PGD	0.37	0.26	0.22	0.2
elasticnet	0	0.67	0.02	0.14
boundary	0.04	0.3	0.04	0.14
JSMA t.1 g.1	0	0.96	0	0.92
JSMA t.1 g.2	0	0.52	0	1
JSMA t.1 g.5	0.15	0.85	0	0
JSMA t.1 g1.0	0	0.89	0	0.84
JSMA t.01 g.1	0.03	0.66	0.039	0.75
JSMA t.01 g.2	0	0.74	0	0.96
JSMA t.01 g.5	0.04	0.41	0.04	0.73
JSMA t.01 g1.0	0	0.81	0.06	0.94

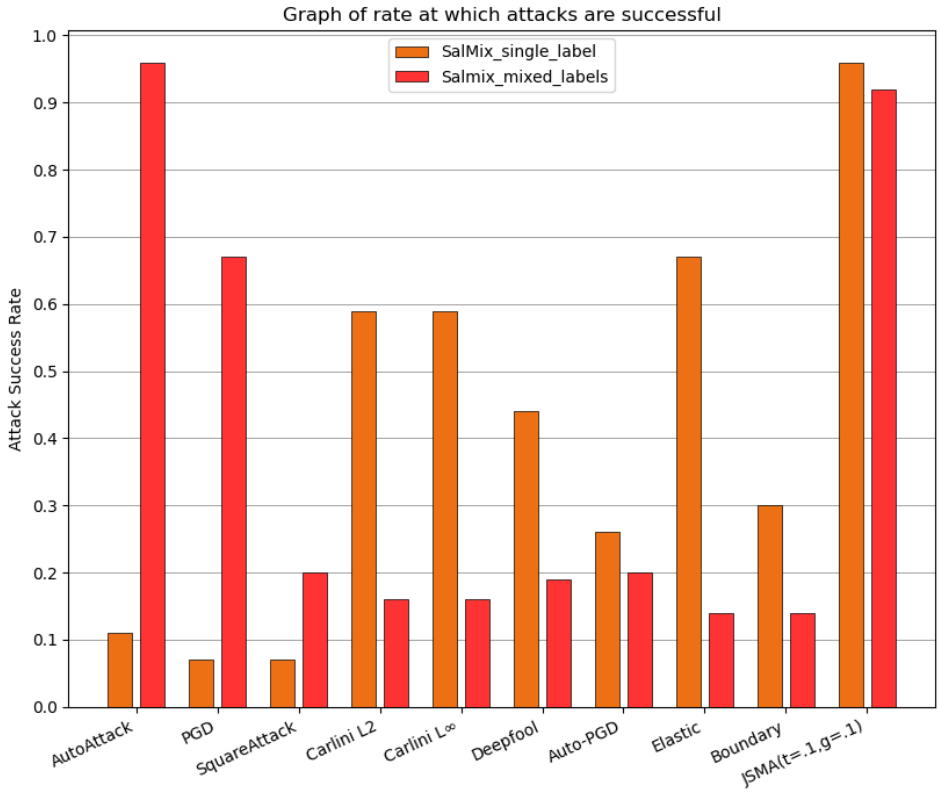

Figure 4.3. Attack success rate for each attack in fooling the SaliencyMix models

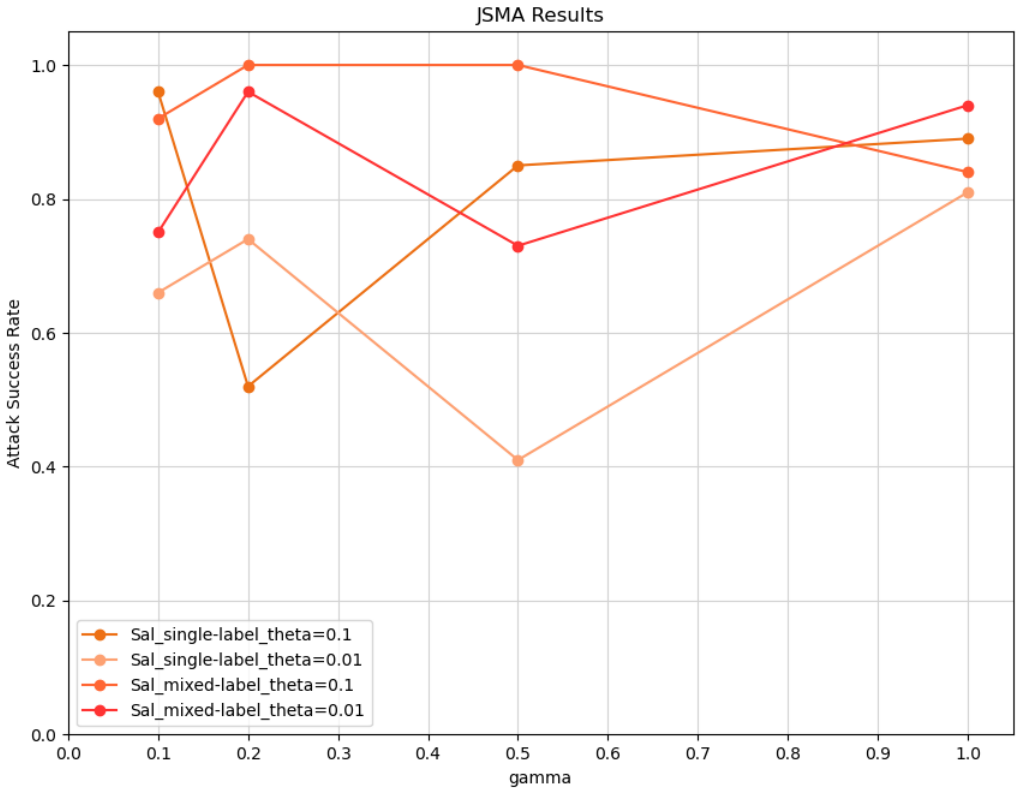

Figure 4.4. Results of JSMA attacks on SaliencyMix models with different values of gamma and theta

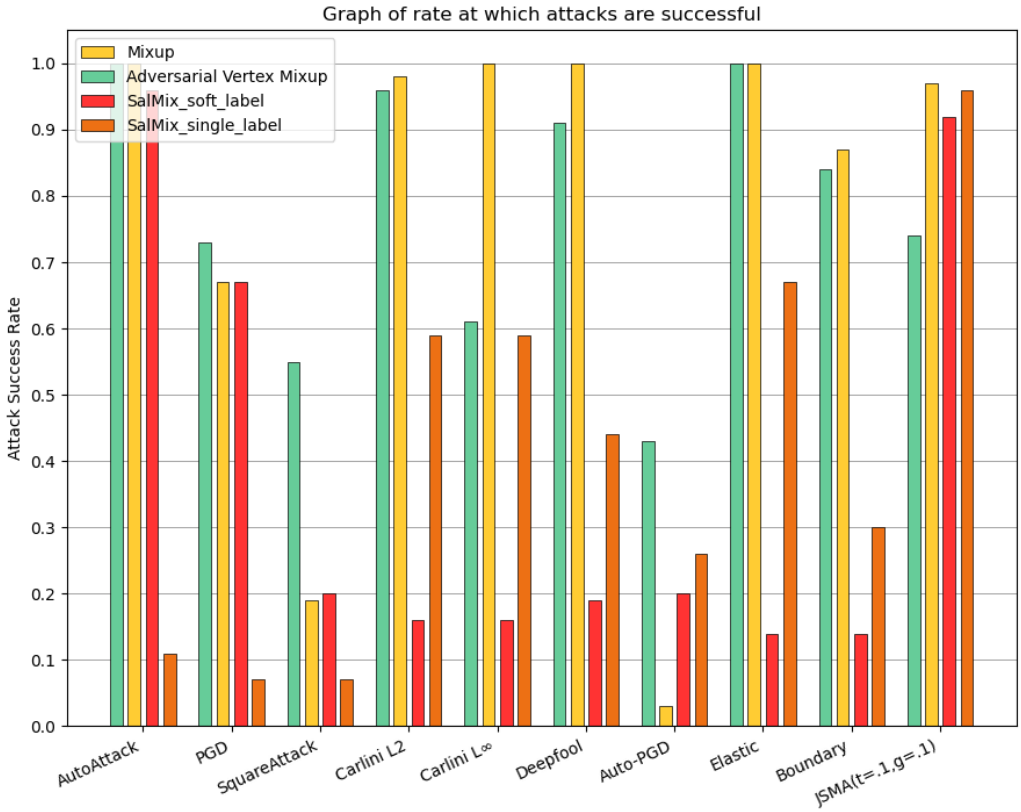

Figure 4.5. Comparison of attack success rates on mixup, adversarial vertex mixup, and SaliencyMix models

4.4 3-Mix Results

Not including JSMA the average attack success rate was 0.023 for 3-Mix with variable-beta and 0.096 for 3-Mix with fixed-beta. Both averages are better than the averages for either of the SaliencyMix models or the Comparison models. Including JSMA with $theta = 0.1$ and $gamma = 0.1$ the averages were 0.115 for 3-Mix with variable-beta and 0.173 for 3-Mix with fixed-beta. Figure 4.6 shows a side by side comparison of the ASR results from the SaliencyMix and 3-Mix models. For the most part the 3-Mix models have significantly better results.

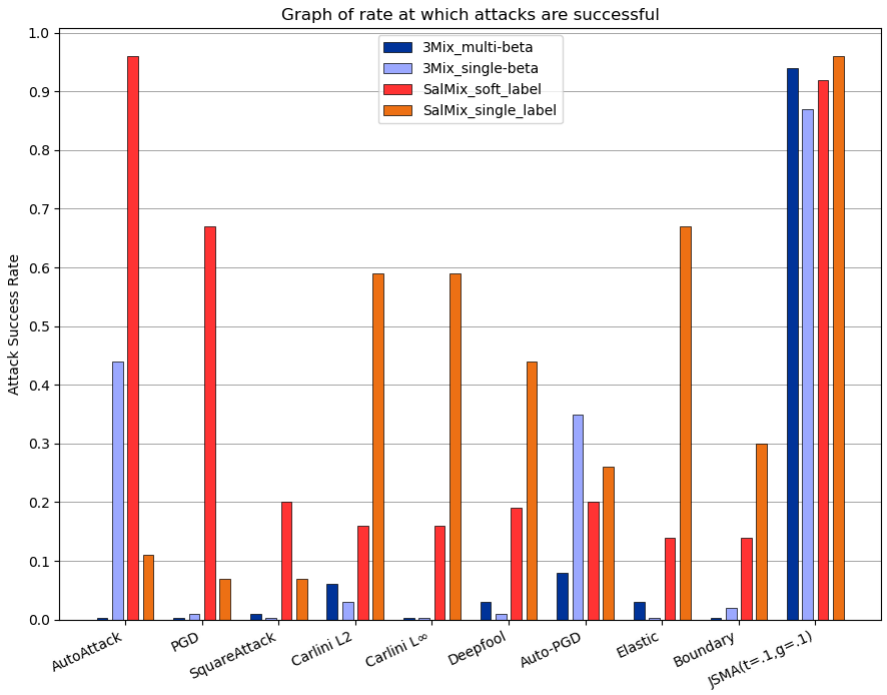

Figure 4.6. Side by side comparison of attack success rate on SaliencyMix and 3-Mix models

For the 3-Mix models the average attack success rates over the JSMA results were 0.845 for variable-beta and 0.808 for fixed-beta. Figure 4.7 show all JSMA results for the 3-Mix models. For SaliencyMix the JSMA averages were 0.893 for multi-label and 0.73 for single-label. For the Comparison models the JSMA average were 0.963 for mixup and 0.9 for Adversarial vertex mixup. None of these models had high degree of robustness against the JSMA attacks although SaliencyMix NOTA aware models and 3-Mix NOTA aware models performed slightly better than the comparison models. Figure 4.8 shows JSMA results for both 3-Mix and SaliencyMix.

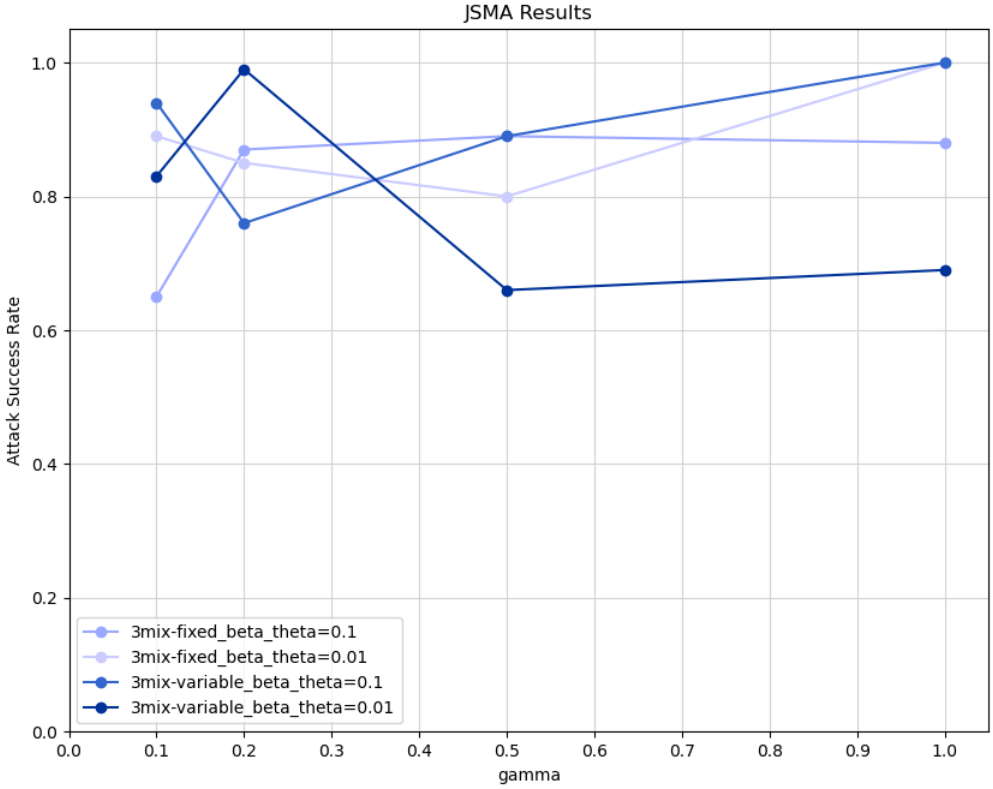

Figure 4.7. Results of JSMA attacks on 3-Mix models with different values of gamma and theta

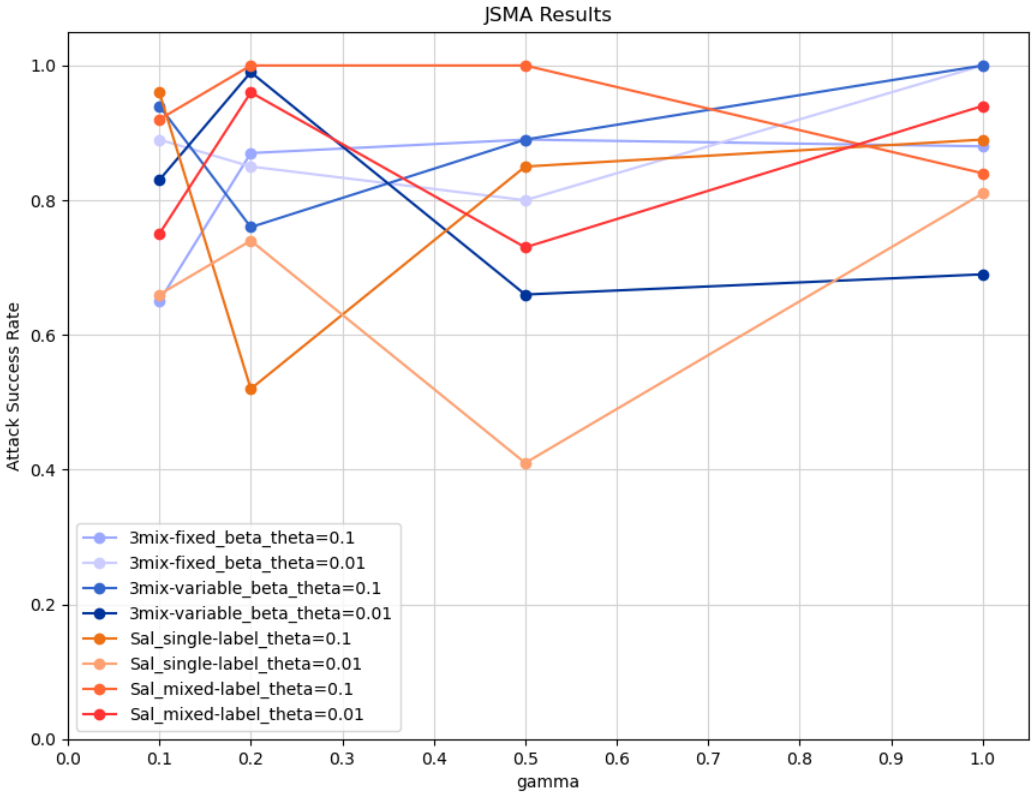

Figure 4.8. Results of JSMA attacks with different values of gamma and theta for SaliencyMix and 3-Mix models

Table 4.4. Table of results for 3-Mix (t=theta, g=gamma)

Attack	3mix variable beta		3mix fixed beta	
	Robust Accuracy	ASR	Robust Accuracy	ASR
autoattack	0	0	0	0
PGD	0	0	0	0
square	0	0.01	0	0
carliniL2	0	0.06	0.01	0.06
carliniL∞	0	0	0	0
deepfool	0	0.03	0	0
auto-PGD	0	0.08	0	0.05
elasticnet	0	0.03	0	0.03
boundary	0	0	0	0
JSMA t.1 g.1	0.01	0.94	0.01	0.65
JSMA t.1 g.2	0	0.76	0	0.87
JSMA t.1 g.5	0	0.89	0	0.89
JSMA t.1 g1.0	0	1	0	0.88
JSMA t.01 g.1	0.02	0.83	0.06	0.89
JSMA t.01 g.2	0.01	0.99	0	0.85
JSMA t.01 g.5	0	0.66	0.01	0.8
JSMA t.01 g1.0	0	0.69	0	1

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 5:
Conclusion and Future Work

Using datasets with multiple types of attacks to train a NOTA aware model will create a more robust defense than previous models were able to achieve. Combining the attacks into a single image using 3-Mix produced the best results. SaliencyMix still produced good results with each adversarial example only having one type of attack. Previous models only blocked around 25% of the surveyed attacks. The new 3-Mix models block around 85% of the surveyed attacks.

Lessons learned: Creating a pre-trained dataset is much faster than training a dataset on the fly each time an experiment is run. This saved considerable time. However, it may have reduced the robustness because the same adversarial examples are seen repeatedly in training. In future work this could be somewhat offset by further increasing the number of adversarial examples in the dataset. Given more processing time and power, generating new AE adapted to the model at intervals in the training (perhaps every 300 epochs) could help to widen the NOTA class making it more effective, more closely resembling the goal illustrated in figure 2.9.

In the future 3-Mix could use different combinations of attacks. Including a JSMA attack could be particularly useful. 3-Mix could also be expanded to N-Mix by including more attacks as discussed in section 3.2.1. The greater the number of different attacks the NOTA class is trained on, the more robust it is likely to be. Hopefully this robustness will extend to blocking newly created attacks in the future, or at least make it more difficult to create new attacks. Although the fixed-beta performed better than the variable-beta in my experiments, it may still be worthwhile to explore different implementations of adjusting the randomness on the beta weights in each N-Mix image to see how the robustness is effected.

THIS PAGE INTENTIONALLY LEFT BLANK

List of References

[1] A. Barton, "Defending neural networks against adversarial examples," Ph.D. dissertation, The University of Texas at Arlington, Dec. 2018 [Online]. Available: https://rc.library.uta.edu/uta-ir/bitstream/handle/10106/27743/BARTON-DISSERTATION-2018.pdf

[2] A. Krizhevsky, V. Nair, and G. Hinton, "The cifar-10 dataset." Accessed May 30, 2023 [Online]. Available: https://www.cs.toronto.edu/~kriz/cifar.html

[3] S. A. Papert, "The summer vision project," Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA, USA, Rep. AIM-100, Jul. 1966 [Online]. Available: https://dspace.mit.edu/handle/1721.1/6125

[4] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," in *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015 [Online]. Available: https://doi.org/10.1038/nature14539

[5] L. Jiao *et al.*, "A survey of deep learning-based object detection," in *CoRR*, 2019 [Online]. Available: http://arxiv.org/abs/1907.09408

[6] H. Schulz and S. Behnke, "Deep learning," in *KI - Künstliche Intelligenz*, vol. 26, no. 4, pp. 357–363, Nov. 2012 [Online]. Available: https://doi.org/10.1007/s13218-012-0198-z

[7] Y. Song, T. Kim, S. Nowozin, S. Ermon, and N. Kushman, "Pixeldefend: Leveraging generative models to understand and defend against adversarial examples," in *CoRR*, 2017 [Online]. Available: http://arxiv.org/abs/1710.10766

[8] C. Szegedy *et al.*, "Intriguing properties of neural networks," in *arXiv preprint arXiv:1312.6199*, 2013 [Online]. Available: https://doi.org/10.48550/arXiv.1312.6199

[9] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *International Conference on Learning Representations (ICLR)*, Mar. 2015 [Online]. Available: https://doi.org/10.48550/arXiv.1412.6572

[10] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," in *CoRR*, 2016 [Online]. Available: http://arxiv.org/abs/1607.02533

[11] N. Papernot, P. D. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *CoRR*, 2015 [Online]. Available: http://arxiv.org/abs/1511.04508

[12] M. Nicolae *et al.*, "Adversarial robustness toolbox v0.2.2," in *CoRR*, 2018 [Online]. Available: http://arxiv.org/abs/1807.01069

[13] M. Barreno, B. Nelson, A. D. Joseph, and J. D. Tygar, "The security of machine learning," in *Machine Learning*, vol. 81, pp. 121–148, May 2010 [Online]. Available: https://doi.org/10.1007/s10994-010-5188-5

[14] T. Gu, B. Dolan-Gavitt, and S. Garg, "Badnets: Identifying vulnerabilities in the machine learning model supply chain," in *CoRR*, 2017 [Online]. Available: http://arxiv.org/abs/1708.06733

[15] L. Muñoz-González *et al.*, "Towards poisoning of deep learning algorithms with back-gradient optimization," in *CoRR*, 2017 [Online]. Available: http://arxiv.org/abs/1708.08689

[16] B. Biggio, L. Didaci, G. Fumera, and F. Roli, "Poisoning attacks to compromise face templates," in *2013 International Conference on Biometrics (ICB)*, pp. 1–7, Jun. 2013 [Online]. Available: https://doi.org/10.1109/ICB.2013.6613006

[17] Python Software Foundation, "Homepage." Accessed May 30, 2023 [Online]. Available: https://www.python.org/

[18] N. Carlini and D. A. Wagner, "Towards evaluating the robustness of neural networks," in *CoRR*, 2016 [Online]. Available: http://arxiv.org/abs/1608.04644

[19] S. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," in *CoRR*, 2015 [Online]. Available: http://arxiv.org/abs/1511.04599

[20] M. Andriushchenko, F. Croce, N. Flammarion, and M. Hein, "Square attack: a query-efficient black-box adversarial attack via random search," in *CoRR*, 2019 [Online]. Available: http://arxiv.org/abs/1912.00049

[21] F. Croce and M. Hein, "Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks," in *CoRR*, 2020 [Online]. Available: https://arxiv.org/abs/2003.01690

[22] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *International Conference on Learning Representations (ICLR)*, Sep. 2019 [Online]. Available: https://doi.org/10.48550/arXiv.1706.06083

[23] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *ICML 2020*, Mar. 2015 [Online]. Available: https://doi.org/10.48550/arXiv.1412.6572

[24] P.-Y. Chen, Y. Sharma, H. Zhang, J. Yi, and C.-J. Hsieh, "Ead: Elastic-net attacks to deep neural networks via adversarial examples," in *AAAI 2018*, 2018 [Online]. Available: https://doi.org/10.48550/arXiv.1709.04114

[25] F. Croce and M. Hein, "Minimally distorted adversarial examples with a fast adaptive boundary attack," in *CoRR*, 2019 [Online]. Available: http://arxiv.org/abs/1907.02044

[26] N. Papernot, P. D. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *CoRR*, 2015 [Online]. Available: http://arxiv.org/abs/1511.07528

[27] X. Cao and N. Z. Gong, "Mitigating evasion attacks to deep neural networks via region-based classification," in *CoRR*, 2017 [Online]. Available: http://arxiv.org/abs/1709.05583

[28] M. Klingner, V. R. Kumar, S. Yogamani, A. Bär, and T. Fingscheidt, "Detecting adversarial perturbations in multi-task perception," in *IROS 2022*, 2022 [Online]. Available: https://doi.org/10.48550/arXiv.2203.01177

[29] Y. Kantaros *et al.*, "Visionguard: Runtime detection of adversarial inputs to perception systems," in *CoRR*, 2020 [Online]. Available: https://arxiv.org/abs/2002.09792

[30] S. Ma, Y. Liu, G. Tao, W.-C. Lee, and X. Zhang, "Nic: Detecting adversarial samples with neural network invariant checking," in *26th Annual Network and Distributed System Security Symposium (NDSS)*, Feb. 2019 [Online]. Available: https://doi.org/10.14722/ndss.2019.23415

[31] H. Zhang, M. Cissé, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," in *CoRR*, 2017 [Online]. Available: http://arxiv.org/abs/1710.09412

[32] S. Lee, H. Lee, and S. Yoon, "Adversarial vertex mixup: Toward better adversarially robust generalization," in *CoRR*, 2020 [Online]. Available: https://arxiv.org/abs/2003.02484

[33] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo, "Cutmix: Regularization strategy to train strong classifiers with localizable features," in *CoRR*, 2019 [Online]. Available: http://arxiv.org/abs/1905.04899

[34] A. F. M. S. Uddin, M. S. Monira, W. Shin, T. Chung, and S. Bae, "Saliencymix: A saliency guided data augmentation strategy for better regularization," in *CoRR*, 2020 [Online]. Available: https://arxiv.org/abs/2006.01791

[35] Google Brain Team, "Tensorflow homepage." Accessed May 30, 2023 [Online]. Available: https://www.tensorflow.org/

[36] J. Kwon, J. Kim, H. Park, and I. K. Choi, "ASAM: adaptive sharpness-aware minimization for scale-invariant learning of deep neural networks," in *CoRR*, 2021 [Online]. Available: https://arxiv.org/abs/2102.11600

[37] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CoRR*, 2015 [Online]. Available: http://arxiv.org/abs/1512.03385

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California