



Calhoun: The NPS Institutional Archive
DSpace Repository

Theses and Dissertations

1. Thesis and Dissertation Collection, all items

2022-09

ATTACKING NEURAL NETWORKS WITH HIGH ENTROPY INPUT SAMPLING

DeRidder, Daniel S.

Monterey, CA; Naval Postgraduate School

<http://hdl.handle.net/10945/71055>

This publication is a work of the U.S. Government as defined in Title 17, United States Code, Section 101. Copyright protection is not available for this work in the United States.

Downloaded from NPS Archive: Calhoun



Calhoun is the Naval Postgraduate School's public access digital repository for research materials and institutional publications created by the NPS community. Calhoun is named for Professor of Mathematics Guy K. Calhoun, NPS's first appointed -- and published -- scholarly author.

Dudley Knox Library / Naval Postgraduate School
411 Dyer Road / 1 University Circle
Monterey, California USA 93943

<http://www.nps.edu/library>



**NAVAL
POSTGRADUATE
SCHOOL**

MONTEREY, CALIFORNIA

THESIS

**ATTACKING NEURAL NETWORKS WITH HIGH
ENTROPY INPUT SAMPLING**

by

Daniel S. DeRidder

September 2022

Thesis Advisor:
Second Reader:

Armon C. Barton
Marko Orescanin

Approved for public release. Distribution is unlimited.

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			<i>Form Approved OMB No. 0704-0188</i>
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington, DC 20503.			
1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE September 2022	3. REPORT TYPE AND DATES COVERED Master's thesis	
4. TITLE AND SUBTITLE ATTACKING NEURAL NETWORKS WITH HIGH ENTROPY INPUT SAMPLING		5. FUNDING NUMBERS	
6. AUTHOR(S) Daniel S. DeRidder			
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) Deep learning is becoming a technology central to the safety and accuracy of many types of systems. Unfortunately, attackers can create adversarial examples that manipulate Deep Neural Networks (DNN) into making incorrect predictions by carefully crafting perturbations that, to humans, look indistinguishable from examples the DNN would classify correctly. Research shows that adversarial examples exist near the decision boundary. Decision-based attacks are designed to find adversarial examples by traversing the data manifold toward the decision boundary using iterative sampling without any knowledge of the model parameters or gradients. In this sense, decision-based attacks are very important, as they apply to many real-world attack scenarios. We propose a new decision-based attack, High Entropy Input Sampling (HEIS), that iteratively steps toward the decision boundary by using entropy over class predictions as a heuristic to find adversarial examples without any knowledge of the model gradients. Using HEIS, we were able to produce adversarial examples that reduced the accuracy of a CIFAR-10 DNN from 91% to 11% for epsilon=0.2 and reduced the accuracy of ResNet50, an ImageNet DNN, from 81% to 22% for epsilon=0.4. Furthermore, we discovered that the adversarial examples are highly transferable to other models, causing dramatic drops in accuracy among all models tested. Finally, we use HEIS to break three state-of-the-art neural network defenses.			
14. SUBJECT TERMS deep learning, neural networks, adversarial attacks, adversarial examples, deep neural networks, decision boundary, entropy		15. NUMBER OF PAGES 101	
		16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UU

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

**ATTACKING NEURAL NETWORKS WITH HIGH ENTROPY INPUT
SAMPLING**

Daniel S. DeRidder
Lieutenant, United States Navy
BS, University of Memphis, 2013

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

**NAVAL POSTGRADUATE SCHOOL
September 2022**

Approved by: Armon C. Barton
Advisor

Marko Orescanin
Second Reader

Gurminder Singh
Chair, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Deep learning is becoming a technology central to the safety and accuracy of many types of systems. Unfortunately, attackers can create adversarial examples that manipulate Deep Neural Networks (DNN) into making incorrect predictions by carefully crafting perturbations that, to humans, look indistinguishable from examples the DNN would classify correctly. Research shows that adversarial examples exist near the decision boundary. Decision-based attacks are designed to find adversarial examples by traversing the data manifold toward the decision boundary using iterative sampling without any knowledge of the model parameters or gradients. In this sense, decision-based attacks are very important, as they apply to many real-world attack scenarios. We propose a new decision-based attack, High Entropy Input Sampling (HEIS), that iteratively steps toward the decision boundary by using entropy over class predictions as a heuristic to find adversarial examples without any knowledge of the model gradients. Using HEIS, we were able to produce adversarial examples that reduced the accuracy of a CIFAR-10 DNN from 91% to 11% for $\epsilon=0.2$ and reduced the accuracy of ResNet50, an ImageNet DNN, from 81% to 22% for $\epsilon=0.4$. Furthermore, we discovered that the adversarial examples are highly transferable to other models, causing dramatic drops in accuracy among all models tested. Finally, we use HEIS to break three state-of-the-art neural network defenses.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

1	Introduction	1
1.1	Machine Learning Using Neural Networks	1
1.2	Reliability and Trustworthiness	2
1.3	Terms and Definitions	3
1.4	HEIS	5
2	Background and Related Work	7
2.1	Previous Research - Adversarial Attacks	7
2.2	Previous Research - Neural Network Defenses	12
3	Design and Methodology	17
3.1	Design	19
3.2	Methodology	25
4	Results	29
4.1	HEIS and CIFAR-10	29
4.2	HEIS and ImageNet	43
5	Discussion and Future Work	55
5.1	Discussion	55
5.2	Future Work	57
6	Conclusion	59
	Appendix: A	61
A.1	More HEIS Adversarial Examples	62
	List of References	75

List of Figures

Figure 2.1	Fast Gradient Sign Method (FGSM) Example	8
Figure 2.2	Transferable Adversarial Examples	9
Figure 2.3	Decision Boundary 2D Plot Example	10
Figure 2.4	Boundary Attack Methodology	12
Figure 2.5	Boundary Attack Example	12
Figure 2.6	Hypercube and Region Based Classification (RBC) Visualization	15
Figure 3.1	Iterative Hyperball Sampling and Testing	18
Figure 3.2	HEIS Multi-sampling Process	23
Figure 3.3	HEIS Entropy Calculation Process	24
Figure 4.1	CIFAR-10: Vanilla Accuracy vs μ_p for $\epsilon = 0.3$	30
Figure 4.2	CIFAR-10: Vanilla Accuracy vs P	31
Figure 4.3	CIFAR-10: Vanilla Accuracy vs ϵ (Gray Box)	32
Figure 4.4	CIFAR-10: Vanilla Accuracy vs ϵ (Black Box)	33
Figure 4.5	CIFAR-10: Average Entropy Values vs Steps	34
Figure 4.6	CIFAR-10 Benign Airplane $\epsilon = 0.031$	35
Figure 4.7	CIFAR-10 HEIS Airplane $\epsilon = 0.031$	35
Figure 4.8	CIFAR-10 Benign Ship $\epsilon = 0.1$	36
Figure 4.9	CIFAR-10 HEIS Ship $\epsilon = 0.1$	36
Figure 4.10	CIFAR-10 Benign Horse $\epsilon = 0.2$	37
Figure 4.11	CIFAR-10 HEIS Horse $\epsilon = 0.2$	37

Figure 4.12	CIFAR-10: HEIS Accuracy vs ϵ (all models, gray box)	39
Figure 4.13	CIFAR-10: HEIS Accuracy vs ϵ (all models, black box)	39
Figure 4.14	CIFAR-10: HEIS Accuracy vs ϵ (GR Model, gray box)	40
Figure 4.15	CIFAR-10: HEIS Accuracy vs ϵ (GR Model, black box)	40
Figure 4.16	CIFAR-10: HEIS Accuracy vs ϵ (AT Model, gray box)	41
Figure 4.17	CIFAR-10: HEIS Accuracy vs ϵ (AT Model, black box)	41
Figure 4.18	CIFAR-10: HEIS Accuracy vs ϵ (TRADES Model, gray box)	42
Figure 4.19	CIFAR-10: HEIS Accuracy vs ϵ (TRADES Model, black box)	42
Figure 4.20	ImageNet: ResNet50 Accuracy vs ϵ (Gray Box)	44
Figure 4.21	ImageNet Benign Cassette Player	46
Figure 4.22	ImageNet HEIS Cassette Player $\epsilon = 0.1$	46
Figure 4.23	ImageNet Benign Mirror	47
Figure 4.24	ImageNet HEIS Mirror $\epsilon = 0.2$	47
Figure 4.25	ImageNet Benign Chainsaw	47
Figure 4.26	ImageNet HEIS Chainsaw $\epsilon = 0.3$	47
Figure 4.27	ImageNet Benign English Springer	48
Figure 4.28	ImageNet HEIS English Springer $\epsilon = 0.4$	48
Figure 4.29	ImageNet Benign Church	48
Figure 4.30	ImageNet HEIS Church $\epsilon = 0.5$	48
Figure 4.31	ImageNet: Average Entropy Values vs Steps	49
Figure 4.32	ImageNet Accuracy vs S (all models)	50
Figure 4.33	ImageNet: HEIS Perturbation Visualization for Various S ($\epsilon = 0.5$)	51
Figure 4.34	HEIS Transfer Accuracy vs ϵ (MobileNetV2)	52
Figure 4.35	HEIS Transfer Accuracy vs ϵ (VGG16)	53

Figure 4.36	HEIS Transfer Accuracy vs ϵ (DenseNet201)	53
Figure A.1	Multi-class 3D Plot Example	61
Figure A.2	CIFAR-10 HEIS Adversarial Examples ($\epsilon = 0.031$)	62
Figure A.3	CIFAR-10 HEIS Adversarial Examples ($\epsilon = 0.1$)	62
Figure A.4	CIFAR-10 HEIS Adversarial Examples ($\epsilon = 0.2$)	63
Figure A.5	CIFAR-10 HEIS Adversarial Examples ($\epsilon = 0.3$)	63
Figure A.6	CIFAR-10 HEIS Adversarial Examples ($\epsilon = 0.4$)	64
Figure A.7	CIFAR-10 HEIS Adversarial Examples ($\epsilon = 0.5$)	64
Figure A.8	CIFAR-10 HEIS Adversarial Examples ($\epsilon = 0.6$)	65
Figure A.9	CIFAR-10 HEIS Adversarial Examples ($\epsilon = 0.7$)	65
Figure A.10	CIFAR-10 HEIS Adversarial Examples ($\epsilon = 0.8$)	66
Figure A.11	CIFAR-10 HEIS Adversarial Examples ($\epsilon = 0.9$)	66
Figure A.12	CIFAR-10 HEIS Adversarial Examples ($\epsilon = 1.0$)	67
Figure A.13	ImageNet HEIS Adversarial Examples ($\epsilon = 0.031$)	68
Figure A.14	ImageNet HEIS Adversarial Examples ($\epsilon = 0.1$)	68
Figure A.15	ImageNet HEIS Adversarial Examples ($\epsilon = 0.2$)	69
Figure A.16	ImageNet HEIS Adversarial Examples ($\epsilon = 0.3$)	69
Figure A.17	ImageNet HEIS Adversarial Examples ($\epsilon = 0.4$)	70
Figure A.18	ImageNet HEIS Adversarial Examples ($\epsilon = 0.5$)	70
Figure A.19	ImageNet HEIS Adversarial Examples ($\epsilon = 0.6$)	71
Figure A.20	ImageNet HEIS Adversarial Examples ($\epsilon = 0.7$)	71
Figure A.21	ImageNet HEIS Adversarial Examples ($\epsilon = 0.8$)	72
Figure A.22	ImageNet HEIS Adversarial Examples ($\epsilon = 0.9$)	72

Figure A.23 ImageNet HEIS Adversarial Examples ($\epsilon = 1.0$) 73

List of Tables

Table 3.1	CIFAR-10 Benign Accuracy	26
Table 3.2	ImageNet Benign Accuracy	27
Table 4.1	CIFAR-10 Before and After (Airplane, $\epsilon = 0.031$)	35
Table 4.2	CIFAR-10 Before and After (Ship, $\epsilon = 0.1$)	36
Table 4.3	CIFAR-10 Before and After (Horse, $\epsilon = 0.2$)	37
Table 4.4	ResNet50 Accuracy vs P for $\epsilon = 0.1$	45
Table 4.5	ImageNet Before and After (Cassette Player, $\epsilon = 0.1$)	46
Table 4.6	ImageNet: HEIS Transfer Accuracy	52

THIS PAGE INTENTIONALLY LEFT BLANK

List of Algorithms

1	HEIS Algorithm	22
---	--------------------------	----

THIS PAGE INTENTIONALLY LEFT BLANK

List of Acronyms and Abbreviations

ART	Adversarial Robustness Toolbox
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
DNN	Deep Neural Network
FGSM	Fast Gradient Sign Method
GPU	Graphics Processing Unit
HEIS	High Entropy Input Sampling
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
ML	Machine Learning
PGD	Projected Gradient Descent
RBC	Region Based Classification
RL	Reinforcement Learning
SVM	Support Vector Machine
TRADES	TRadeoff-inspired Adversarial DEfense via Surrogate-loss minimization
WRN	Wide Residual Network

THIS PAGE INTENTIONALLY LEFT BLANK

Acknowledgments

First and foremost, thank you to my amazing wife, Kristian, and our wonderful kids, Sam, Wyatt, and Dalton. Thank you for putting up with all the late nights, long hours, and stressed-out stages of living with a full-time master's student. Thank you to my advisor, Dr. Barton, for always providing helpful insights and double checking my work. Thank you to my fellow cohort members who always stuck together and helped each other out when needed. Finally, thank you to the Navy, that in its infinite wisdom, allowed my family and me the opportunity to live in beautiful Monterey, CA, for two years.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1: Introduction

1.1 Machine Learning Using Neural Networks

Machine learning (ML) is the science (and art) of programming computers to learn from data [1]. The main motivation behind machine learning is the fact that some problems cannot be solved by a traditional sequential computer program. We want a program or algorithm that will enable a machine to incrementally perform better at a specific task by providing it with data to learn from, instead of explicitly programming rules that it must follow. The goal of this process is to create machines that are trained and able to reliably perform their tasks (prediction, identification, tracking, etc.) in order to provide confident automation in areas that were not previously able to be automated.

There are a wide variety and types of machine learning, from Reinforcement Learning (RL) algorithms, to Linear Regression modeling, to Artificial Neural Networks (ANNs). This thesis will focus on the study of Deep Neural Networks (DNNs), which originated from the study of artificial neurons and ANNs. While a simple ANN might consist of a single or multi-layer perceptron¹, a DNN is an artificial neural network that has many, many layers of artificial neurons (tens or sometimes hundreds of layers). The term Deep Learning refers to the concept of ML using DNNs and is also the term given to the field of studying DNNs.

Deep Learning using DNNs has become more widely accessible and studied in the last few decades as computer processing hardware has increased in capability [2]. Advances in neural network architectures and the ability to train them has been made possible by the advent of multi-core processors, especially Graphics Processing Units (GPUs) that can perform thousands of computations in parallel. This significant boost in hardware performance brought DNN training that was once thought to be impossible into the realm of practical.

ML and DNNs are at the heart of a wide variety of technologies and various applications, including but not limited to

¹Perceptron: a handful of artificial neurons configured in one to several layers.

- Image/object classification
- Voice recognition
- Natural language processing
- Filtering spam emails
- Automatically flagging offensive online comments
- Detecting fraudulent financial activity
- Building an intelligent "bot" to play a game

In this thesis, the DNNs we utilize are designed, built, and trained for image recognition and classification using well-known data sets and architectures in order to provide as meaningful comparability as possible in our results.

1.2 Reliability and Trustworthiness

With the proliferation of ML and Deep Learning, the topics of reliability and trustworthiness inevitably come into focus, especially in the case of DNNs. Despite their widespread use and practical successes, it is still not completely understood how DNNs formulate their decision boundaries/regions or how they come up with their decisions [3]. This lack of understanding prompts questions like: How trustworthy are the results? How susceptible is the network to intentionally misleading or noisy data? How robust is the network against attacks? What does an attack against a Neural Network even look like? These questions have led to research on what is known as adversarial attacks and adversarial examples. An adversarial attack is an algorithm that seeks to find minimal perturbations to apply to input data with the goal of causing the DNN to output incorrect information [4]. Input data that contains these adversarial perturbations are called adversarial examples. An adversarial example is a piece of intentionally misleading data that is designed purposefully to cause a machine learning model to misclassify it while appearing non-malicious [5]. These adversarial examples often have perturbations that are very small, with individual pieces of the input data being only slightly altered from the original value. Thus, to the human eye, adversarial examples often look very similar to their original counterparts, and in many cases can even appear completely unaltered [4].

Research has shown that it is quite possible for a DNN, when given one of these adversarial examples, to output an incorrect classification with high confidence – even if the adversarial

example appears, to a human observer, exactly the same as the benign original [6], [7], [8].

Why study the creation and effectiveness of adversarial attacks and adversarial examples? DNNs have become increasingly common as modern decision-making tools [4]. The weaknesses of Neural Networks must be studied to explore their fragility, to set realistic expectations on their performance, and ultimately to provoke the creation of effective defenses. This is especially true when utilizing DNNs in applications where integrity, security, and trustworthiness of deployed systems are of top priority – such as autonomous cars, facial recognition systems, or unmanned systems.

1.3 Terms and Definitions

There are several types of adversarial attacks, but in general attacks are classified into several categories. It is helpful to become familiar with the concepts and terms that are commonly used among adversarial attack research. Also, it is common when working with image classification DNNs to use the terms "example", "image", and "sample" interchangeably. Throughout this thesis, they will be used as such.

1. Targeted vs Non-Targeted [9]
 - Targeted: attacks that are designed to produce inputs that are classified as a specific predetermined category.
 - Non-Targeted: attacks that do not care what the predicted category of the adversarial example is, as long as it is not the correct one.
2. Single-Step vs Iterative [10]
 - Single-Step: an attack that generates adversarial examples using a single calculation, e.g. calculating the loss gradient of a model at one time.
 - Iterative: an attack that generates adversarial examples over multiple steps, usually involving multiple predictions or passes through a model/network while maintaining and updating a perturbation value.
3. White Box vs Black Box vs Gray Box [4], [11]
 - White Box: An attack that has total access to the inner workings of the model. Any information about the model can be used in adversarial example creation, such as model gradients, confidence scores, training data, training algorithm,

model architecture, logits², etc.

- **Black Box:** An attack that has no knowledge of the model. Model vulnerability can only be assessed by observing past inputs and corresponding model outputs. In literature this is sometimes referred to as a "strict black box" attack.
- **Gray Box:** Limited access is allowed to the model, but not total internal access. In this thesis, when referring to "gray box", we assume the attack is allowed access to change the model inputs, observe and collect model outputs, and can query the model; yet the attack has no access to any internal parameters like those listed for "White Box".

There are also different general methodologies for adversarial example creation. Adversarial attacks like ours that are designed to cause the DNN to misclassify a given input are often divided into four categories [12]:

1. Gradient-based attacks

- These are some of the most common attacks and rely on detailed model information, specifically the gradient of the network weights with respect to an input image. This is explained later in Chapter 2, Section 2.1.1.

2. Score-based attacks

- These attacks rely on the predicted scores (e.g. class probabilities or logits) of the model. Conceptually, these attacks try to estimate the gradient from these scores.

3. Transfer-based attacks

- These attacks do not rely on information about the model itself, but rather on information from the data the model was trained on. It has been shown that substitute models trained on the same data can produce adversarial examples that are very effective on the attacked model [13].

4. Decision-based attacks

- These attacks rely solely on the final decision/prediction of the model. This is different from Score-based attacks in that they do not need confidence scores or logits from the model, only the model's final decision/classification.
- This is also arguably the most relevant type of attack in real-world ML applica-

²Logit: The vector of raw predictions that a classification model generates, which is typically fed to a normalization function (e.g. softmax).

tions where users are not likely to have access to training data, logits/confidence scores, or model gradients.

1.4 High Entropy Input Sampling (HEIS)

We call our proposed attack HEIS, it is a non-targeted, iterative, decision based black box attack, the methodology of which is explained in detail in Chapter 3. Decision-based attacks are explained in detail in Chapter 2, Section 2.1.3. These types of attacks aim to fool the DNN by finding adversarial examples that exist close to a DNN decision boundary. The advantage of decision-based attacks is that the attacker does not need access to the model parameters or data like a white box attack would require. This makes decision-based attacks more robust to common DNN defenses that aim to defend against gradient based attacks such as gradient masking and gradient regularization [12].

Most adversarial attacks create adversarial examples that cause the target system to misclassify the examples with high confidence [4], HEIS adversarial examples are classified with low confidence. While other decision based attacks rely on accuracy and/or image distance metrics [12], HEIS relies solely on an entropy calculation that incentivizes the algorithm to find adversarial examples that result in distributed model confidence across multiple classes. Our final ImageNet adversarial examples typically resulted in top-1 classification confidences no higher than 20-30%. We found this technique produces effective and highly transferable adversarial examples.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 2: Background and Related Work

Substantial research has been conducted to discover how adversarial attacks affect neural networks and also how to properly defend DNNs against threats [4], [11], [14]. Summarized here are a few well known attacks and defenses for neural networks and the general ideas behind how and why they work. Knowing how these types of attacks work will allow one to better understand the logic behind HEIS, which is covered in depth in Chapter 3.

2.1 Previous Research - Adversarial Attacks

2.1.1 Gradient Based Adversarial Attacks

A few well-known methods of creating adversarial examples for neural networks are Fast Gradient Sign Method (FGSM) from [6] and Projected Gradient Descent (PGD) from [15]. To understand how these types of attacks work, one must understand how a neural network learns by using gradient descent. Gradient descent is a generic optimization algorithm that is used to minimize some other function by iteratively moving in the direction of the steepest descent [1]. The descent of the function is determined by calculating the derivative of the function with respect to each of its coefficients (in neural networks the coefficients are the network weights). Imagine a ball placed on the side of a large valley. Where will the ball end up? At the deepest point in the valley, hopefully. That is what gradient descent is trying to accomplish. The slope of the valley represents the derivative of the function the network represents, and the starting position of the ball represents the initial neural network weights. Just like how we expect the ball to end up at the deepest point in the valley, gradient descent helps find the most optimal values for the neural network weights.

We randomly initialize the parameters of a network to find a starting point of the 'ball'. Then we calculate the slope at that position (via the partial derivative(s)) which in turn tells us how to tune the parameters in order to move the ball down the "valley." From there, we take a small step (step size is controlled by a parameter called the learning rate) to come up with a set of new parameters which gives us a new starting point. We simply 'move' in

the direction of the most negative slope - bringing us closer to the optimal values (i.e. the bottom of the valley). If we gradually and repeatedly perform these steps, we will eventually end up near the bottom of the valley (i.e. descend the gradient), and in the case of a neural network, this allows gradient descent to find the optimal network weights.

An adversarial attack like FGSM works by using the gradients of a neural network to create an adversarial example. For an input image, FGSM creates an adversarial example by examining the gradients of the network’s loss function (with respect to the image) to create a new image that minimizes the loss [6]. Since the loss is calculated across the entire input image, which can be broken up into multiple subsections (all the way down to individual pixels), it is easy to compute how much each portion of the input contributes to the overall loss by using the chain rule and finding the gradients. Then the adversarial example can be constructed by adding some small perturbation to the most effective section within the image. These perturbations are often small enough that the new adversarial example and the original input image may look identical to the human eye, but due to the perturbations being in just the right spot, it will fool the network into making wildly different classification decisions (see Figure 2.1) [6]. PGD is another gradient based attack that works very similarly to FGSM but uses multiple gradient steps per iteration to create adversarial examples [15].

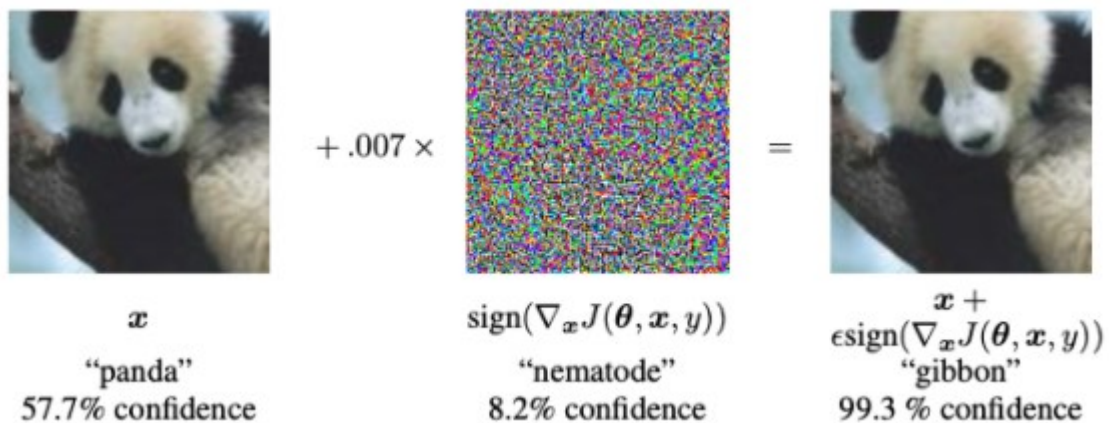


Figure 2.1. An FGSM example of an input image with added perturbations that cause a misclassification from 'panda' with 57% confidence to 'gibbon' with 99% confidence. Source: [6].

2.1.2 Transferable Adversarial Attacks

Research has shown that adversarial examples generated specifically for one model are often effective against other models that were designed for the same data set [4], [13]; this is known as transferability. A typical use-case for transferable adversarial attacks is where an attacker does not have full access to the model they wish to attack, but instead creates adversarial examples using a model that they do have access to – or one they created to mimic the target model. This surrogate model is trained using the same or similar data as the target, then used to create the adversarial examples. Liu et al. [13] showed that it is possible to create targeted and non-targeted adversarial examples that can transfer to a black-box system whose model, training data, and truthful label set is unknown to the attacker. Figure 2.2 shows adversarial examples that were created on their surrogate system, and then used to successfully fool a completely different model. This case is interesting in that their surrogate model was not trained using the same data set as the model being attacked, but still showed excellent transferability of the resultant adversarial examples.








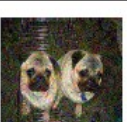
original image	true label	Clarifai.com results of original image	target label	targeted adversarial example	Clarifai.com results of targeted adversarial example
	viaduct	bridge, sight, arch, river, sky	window screen		window, wall, old, decoration, design
	hip, rose hip, rosehip	fruit, fall, food, little, wildlife	stupa, tope		Buddha, gold, temple, celebration, artistic
	dogsled, dog sled, dog sleigh	group together, four, sledge, sled, enjoyment	hip, rose hip, rosehip		cherry, branch, fruit, food, season
	pug, pug-dog	pug, friendship, adorable, purebred, sit	sea lion		sea seal, ocean, head, sea, cute

Figure 2.2. Original images and corresponding adversarial examples generated using an ensemble based approach to creating transferable examples. The images were successful against Clarifai.com, which is a black box image classification system. Source: [13].

2.1.3 Decision Based Attacks

One aspect of DNNs that is not yet fully understood is their decision boundaries and their geometrical properties [16]. Compared to other aspects of DNNs, decision boundary characterization is still in the early stages of study [17]. Decision based attacks typically rely on finding adversarial examples that lie close to a model's decision boundary in order to cause the model to produce incorrect outputs. For example, let's examine the decision boundary of a linear Support Vector Machine (SVM) as shown in Figure 2.3. The line between green and purple areas of the graph represent the model's decision boundary. Data given to the model for classification that falls within the green feature space will be classified as a triangle; data that falls within the purple feature space will be classified as a square. Data with features that lie too close to the decision boundary have a higher chance of being classified incorrectly, as you can see by the fact that there are several instances of green triangles in purple space and purple squares in green space. Decision based attacks try to intentionally create or find adversarial examples that lie close to a model's decision boundary.

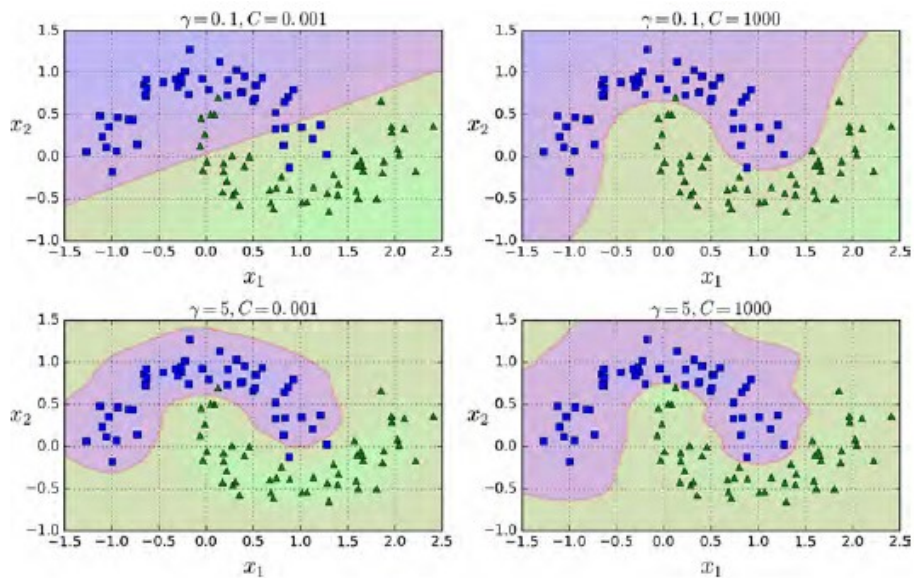


Figure 2.3. An example of several decision boundaries formed by linear SVM classifiers using an RBF kernel and various hyperparameters. Source: [1].

Since DNNs are typically very high-dimensional systems, it is not easy for humans to visualize or understand the decisions they make. To explain decision boundaries, it is

useful to utilize a simple model like a SVM with few dimensions, then simply point out that the concepts mathematically scale up to higher dimensions, even though they may not be directly observable. Decision boundaries get more complex as the dimensionality of the dataset and model goes up. Without showing any boundaries, Figure A.1 shows a plot of a multi-dimensional feature space where it becomes much harder to visualize and define boundaries between the different classes, since you must draw boundaries as three dimensional planes instead of two dimensional lines. Visualizing decision boundaries becomes even more perplexing, if not impossible, when the data dimensionality increases beyond three, as is usually the case when dealing with DNNs¹. However, it will still have decision boundaries in the feature space where it decides between one class or another. Finding input data that lies close to these boundaries is non-trivial and computationally expensive [16].

A decision based attack is an attack that relies solely on the final decision of the model to create adversarial examples [12]. The complexity of decision boundaries, however, makes decision based attacks difficult and requires multiple iterative steps, whereas some white box attacks can calculate adversarial examples in a single step (e.g. FGSM [15]). This is likely the reason why there are very few decision based attacks, despite their importance for real-world systems. In practice, attackers most likely will only have access to the model's final decision.

The original Boundary Attack, first proposed by Brendel et al. [12], was the first effective decision based attack. It works by starting with an adversarial example that already has a large perturbation (discovered by random sampling around the target), "then performs a random walk along the boundary between the adversarial and the non-adversarial region such that: (1) it stays in the adversarial region, and (2) the distance towards the target image is reduced." Figure 2.4 shows a nice graphical representation of this concept. Although surprisingly conceptually simple, since their Boundary Attack requires a lot of sampling and iterations, they discovered that their attack required many more iterations (roughly 75x more steps) to converge than popular gradient-based attacks. Although it is worth mentioning that at intermediate step counts, their attack produces adversarial examples that are easily recognizable by human eyes (see Figure 2.5). The Boundary Attack represents an extremely

¹Even simple 1-dimensional gray-scale MNIST images, where images have a square 28x28 shape, have an input space of $256^{(28*28)}$ (assuming pixel values are integer values between 0-255).

important technique that demonstrates an effective decision based attack that scales to more natural datasets such as ImageNet. This attack is able to produce adversarial examples that are effective against models that deploy defensive techniques as well as models in real-world black box settings [12].

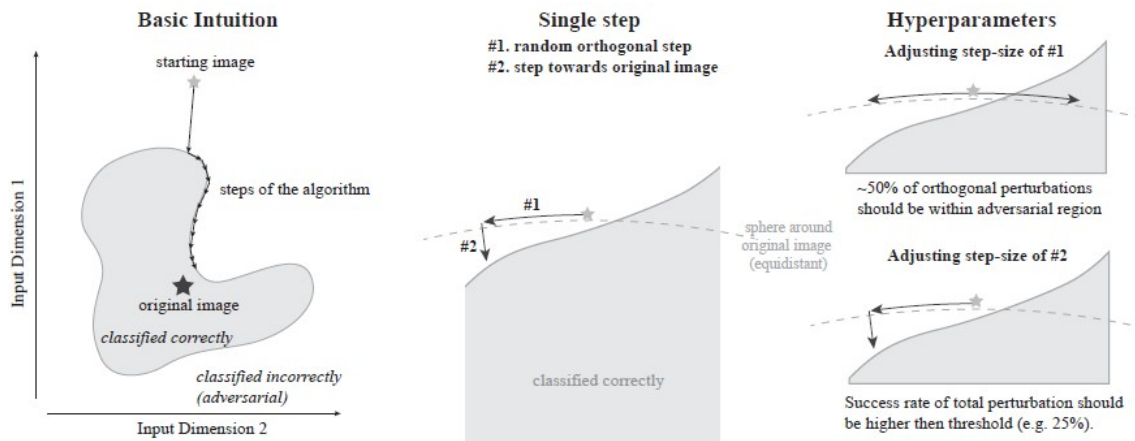


Figure 2.4. The Boundary attack starts with an adversarial example, then steps along the decision boundary so that (1) and (2) hold. Source: [12].

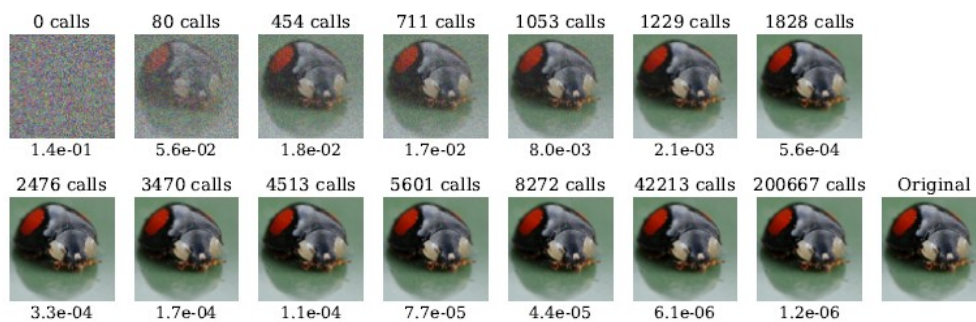


Figure 2.5. An example of the Boundary Attack shown at various iterations. Source: [12].

2.2 Previous Research - Neural Network Defenses

With the discovery of adversarial examples and adversarial attacks, there have been many techniques researched and developed to combat the effectiveness of adversarial examples on DNNs [4], [18], [19]. Techniques such as defensive distillation [20], [21], adversarial

training [9], [15], trading off between accuracy and robustness via surrogate-loss minimization (an algorithm called TRADES) [22], gradient regularization [23], and region-based classification [24] help DNNs operate as expected despite Adversarial Attacks.

2.2.1 Defensive Distillation

Papernot et al. first proposed using distillation as a defensive technique against adversarial examples [20]. Distillation is the term given to the process of training a (usually smaller) DNN using knowledge gained from a different (usually larger) DNN. The original motivation behind distillation was to reduce the computational complexity and architecture needs of a smaller DNN by utilizing knowledge gained from a larger DNN in order to deploy deep learning in resource-constrained environments, like edge computing or smart phones. Defensive distillation instead utilized "the knowledge extracted from a DNN to improve its own resilience to adversarial samples" [20]. Although defensive distillation was very promising, the defense has been shown to be easily defeated by more advanced adversarial attacks [21].

2.2.2 Adversarial Training

Adversarial training is the process in which a network is hardened against adversarial examples by including them in the training data set [7], or creating its own adversarial examples while training in order to better recognize them afterwards [6]. The motivation behind adversarial training is that the network would be taught how to correctly classify adversarial examples and thus be more robust against any future attacks. Although adversarial training has been shown to still be vulnerable to more sophisticated attacks, recent work has demonstrated increased model robustness against CIFAR-10 adversarial examples created using methods like PGD [15]. However, adversarial training has not been successfully proven robust for more complex, natural datasets like ImageNet [19].

2.2.3 TRadeoff-inspired Adversarial DEfense via Surrogate-loss minimization (TRADES)

Statistically, model robustness (performance against adversarial examples) and standard accuracy (performance against regular samples) can be at odds with each other [25]. Although this trade-off is well known and has been the subject of many empirical studies,

the underlying theory behind the cause of this trade-off is still largely unknown. Zhang et al. [22] created a novel defense called TRADES, in which they bound the robust error² using two terms: one that represents the natural error³ and one they call the boundary error⁴. The natural error is measured and optimized using a surrogate loss function. The entire robust error is then minimized using a differential upper bound on the gap between robust error and natural error that has been shown to be the tightest upper bound possible (using the theory of classification-calibrated loss). This approach is unique in that they used a surrogate loss function consisting of two terms: one that maximizes the standard accuracy and another regularization term that pushes the network's decision boundary away from the training data, which improves adversarial robustness. They discovered that their TRADES algorithm performs very well compared to other defenses under both white box and black box threat models, it even won the NeurIPS 2018 Adversarial Vision Challenge [22].

2.2.4 Gradient Regularization

Gradient regularization refers to a defense technique in which models are trained while being differentially penalized according to the degree to which small input changes alter its final predictions. In other words, the model is penalized when small input changes cause drastically different predictions. Models trained using this input gradient regularization have been shown to be more robust against adversarial examples, especially those created under a black box threat model [23].

2.2.5 Region Based Classification (RBC)

One technique designed to help DNNs classify adversarial examples is called RBC. What makes adversarial examples so effective is that they lie near a decision boundary for a DNN (the boundary between two or more classification outcomes), sometimes called a classification boundary between classification regions. Researchers discovered that if you sample from a hypercube⁵ around an adversarial example, most of those samples would still

²Robust error: the prediction error for adversarial examples.

³Natural error: the traditional classification error.

⁴Boundary error: characterizes how likely the input features are close to the decision boundary.

⁵A hypercube is an abstract, multi-dimensional "cube" centered around an example input. The other samples in this hypercube can be found by varying the parameters of the center example slightly so as to remain very similar to the original, but different enough to warrant a separate prediction by the DNN. Sometimes called a hyperball.

be correctly classified by the network [24]. In other words, the hypercube mostly overlaps with the correct side of the decision boundary (see Figure 2.6). Thus, RBC samples from a hypercube around whatever input it is given, classifies those samples, then makes its prediction on the original input based on the most common prediction among all the samples. This technique significantly reduces the effectiveness of some adversarial attacks without sacrificing prediction accuracy on non-adversarial examples [24].

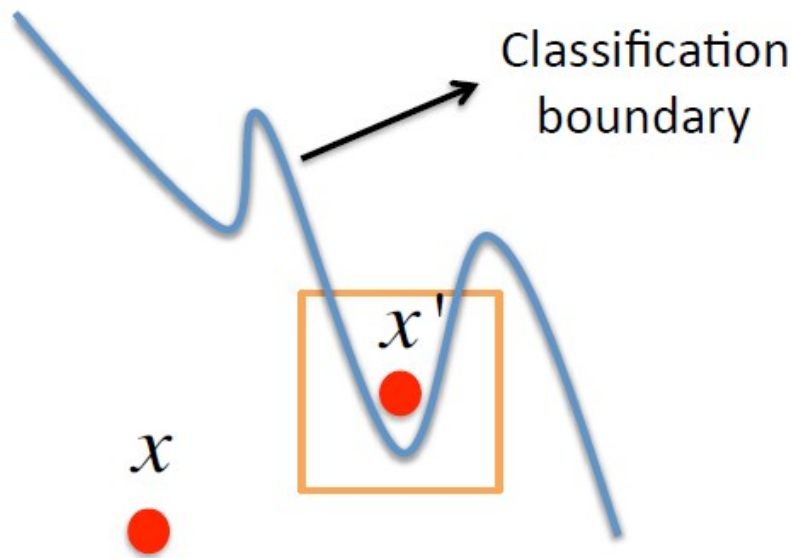


Figure 2.6. x is a benign example and x' is a corresponding adversarial example. The hypercube centered on x' intersects mostly with the correct classification region. Source: [24].

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 3: Design and Methodology

In this chapter we discuss the motivation behind our attack, the framework and hyperparameters used, and methodology of the algorithm. The basic principle behind HEIS is to start from a benign¹ input image and sample from a surrounding hyperball, with radius ϵ bounded by some l_p -norm, to find nearby samples that have a class distribution with high entropy when fed back through the model. The nearby sample that produces a class distribution with the highest entropy is then used as the beginning sample for the next round of the attack (see Figure 3.1). The motivating idea is that by randomly sampling from the feature space around the input and testing each of those samples on the model, we should be able to find samples that lie close to the model's decision boundaries, yet with small perturbation bounded by ϵ . We found that this theory does hold and HEIS is quite capable of generating adversarial examples that are very effective in both gray box and black box threat models.

¹Benign: an input with no adversarial perturbation(s); could also say "normal" or "regular".

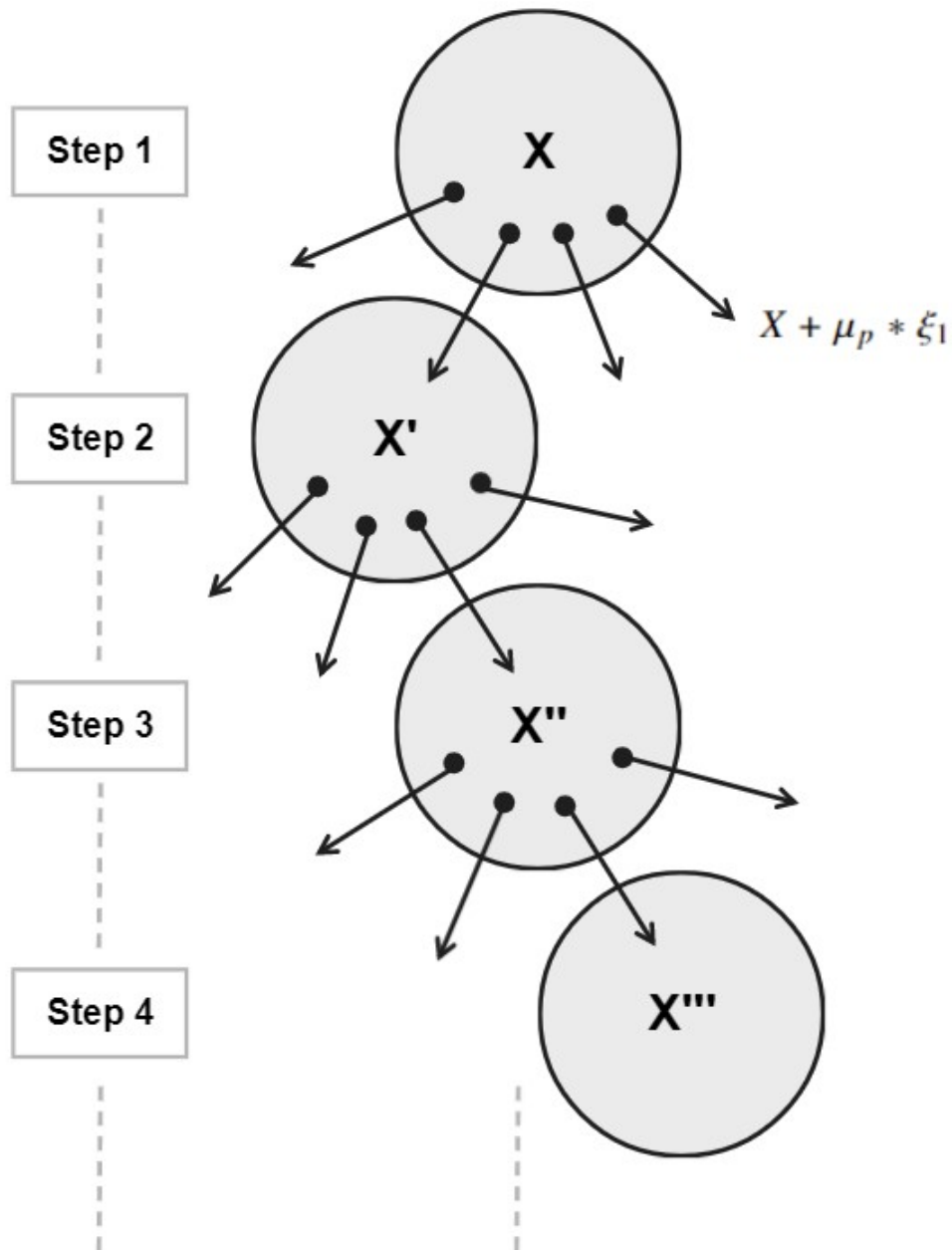


Figure 3.1. At step 1, X is the initial benign input. Only 4 samples for each iteration are shown here, found by calculating $X + \mu_p * \xi_1$ where ξ_1 is sampled from a Gaussian distribution, μ_p is a scaling coefficient, and $(\mu_p * \xi_1) \leq \epsilon$. Random sampling finds a nearby image in the feature space that returns some entropy over the class distribution when fed back through the model. The neighbor with the highest entropy is used as the starting point for the next step. The algorithm is explained in more detail in Section 3.1.2.

3.1 Design

The following are the general steps for conducting our attack:

1. Sample from a hyperball, with radius ϵ , around the current input to get several "Starting Points".
2. Sample from a hyperball around each "Starting Point" and pass those samples back through the model to examine the resulting class distributions.
3. Calculate the entropy of each class distribution to see which one is highest.
4. Choose the "Starting Point" whose class distribution led to the highest entropy as the beginning sample for the next round.

After several iterations of this process, the algorithm should theoretically find input data (i.e. adversarial examples) that lie close to the model's decision boundaries, having perturbation bounded by ϵ , and that result in a very high degree of uncertainty in the model's predictions. The attack relies almost solely on the entropy calculation. Entropy is a term for the measurement of uncertainty in a distribution given by:

$$H(X) := - \sum_{x \in X} p(x) \log(p(x))$$

Where X is a distribution and each event $x \in X$ occurs with probability $p(x)$. In our use, each event corresponds to one of the model's output classes. This entropy calculation results in higher values when the class distributions are more spread out, and lower values when the class distributions are centered on one or only a few classes. We can use this to indirectly measure the confidence of the model in its predictions without actually having access to the network's logits or confidence scores.

The lower the entropy value, we can assume that the model is more confident in its predictions. Conversely, the higher entropy values indicate the model's predictions were spread out over more classification categories, and thus had a higher degree of uncertainty about the input data. We utilize the resulting entropy values to guide the creation of our adversarial examples. HEIS iteratively finds examples with perturbation bounded by ϵ that cause the model to have high entropy over the predicted class distribution (i.e. lower classification confidence) thus creating adversarial examples that are highly effective in both gray box and

black box environments. To avoid the algorithm blindly adding noise at every step without increasing the effectiveness of the adversarial examples, we included the result of the previous step as one of the Starting Points in the current step. This means that the algorithm could potentially carry the same adversarial example forward through multiple steps if it fails to find a nearby sample that results in a higher entropy calculation. During testing, we often observed this phenomenon of HEIS opting for none of the Starting Points during one or several steps.

3.1.1 Hyperparameters

The method HEIS uses to create adversarial examples is fairly straightforward but proved to be computationally expensive in execution due to the multi-sampling process – similar to the Boundary Attack [12]. This type of experimentation required many tunable hyperparameters that could affect the process of creating adversarial examples. These parameters included:

1. Epsilon (ϵ)
 - Floating point value between zero and one (0-1).
 - Controls the clipping of adversarial examples during creation.
 - Utilized as a percentage of original data; i.e. an epsilon of 0.1 means no more than 10% change in any individual data point (i.e. pixel in an image).
 - For example, if the range of input pixels is between 0-255, then an epsilon value of 0.1 means that pixels will not be allowed to change by more than ± 25.5 .
2. Number of Steps (S)
 - A positive integer value.
 - Number of iterations of the algorithm to complete.
3. Starting Point Noise Coefficient (μ_p)
 - Floating point value between zero and one (0-1).
 - Coefficient that determines the amount of noise added when determining starting points.
4. Number of Starting Points (P)
 - A positive integer value.
 - Controls the number of starting point samples to use when attempting to find a nearby sample with the highest entropy.
5. Number of Entropy Samples (N)

- A positive integer value.
 - Number of samples to take when calculating entropy for an individual starting point.
6. Entropy Noise Coefficient (μ_e)
- Floating point value between zero and one (0-1).
 - Coefficient that determines the amount of noise added when taking samples for the entropy calculation.

The hyperparameter that we varied the most was epsilon (ϵ). As discussed in Chapter 4, some of the hyperparameters, like the starting point noise coefficient (μ_p) and number of starting points (P), did not seem to alter performance much. Others, like the number of entropy samples (N), we intentionally kept at a constant value in order to lower computational requirements; it also allowed us to more easily note meaningful performance changes when varying other hyperparameters. For all of the CIFAR-10 testing and most of the ImageNet testing, we used $S = 100$ steps².

3.1.2 Algorithm

The HEIS algorithm is surprisingly simple and fairly straightforward; Algorithm 1 shows the steps taken in creating adversarial examples. There is an outer **for** loop that runs the algorithm through S number of steps (line 1), creating P Starting Points at each step (line 3). Clipping is applied to each Starting Point to bound perturbation by ϵ (line 4). There is an inner **for** loop (line 6) that creates N samples from each starting point (line 8), passes them back through the model to create the class distributions (line 9), then calculates the entropy for each distribution (line 10). The last step is to simply choose the starting point that resulted in the highest entropy as the next $X_{current}$ (lines 12-13). Figure 3.1 shows a high-level representation of this process as the algorithm takes steps towards the decision boundary. Figure 3.2 shows how HEIS takes P starting points and N entropy samples by controlling the sampling process using μ_p and μ_e . Figure 3.3 shows the entropy calculation process for each starting point. The entries in array E align with their corresponding starting points in array X_{sp} .

Since HEIS is an iterative multi-sampling process, it can be computationally expensive for

²In Section 4.2, we tested a few different values of S on our ImageNet model.

Algorithm 1 HEIS

Variables	
X	the original sample
$X_{current}$	array of current images
X_{sp}	array of starting points (also images)
x_{max}/x_{min}	the max/min values of the input data
E	array of floating point values
$model$	the target ML model (in our case an image classification DNN)
Functions	
$clip(A, b, c)$	clips values in array A between b and c
$A.append(x)$	appends item x to the end of list A
$argmax(A)$	returns the index of the largest value in list A
$m.predict(X)$	uses model m to predict values of X

```
1: for  $steps = 1, 2, \dots, S$  do
2:    $\xi_1 \leftarrow N(\mu, \sigma^2)$                                  $\triangleright \xi_1 = \text{random normal noise}$ 
3:    $X_{sp} \leftarrow X_{current} + \mu_p * \xi_1$                      $\triangleright \text{get } P \text{ Starting Points}$ 
4:    $X_{sp} \leftarrow clip(X_{sp}, X + \epsilon, X - \epsilon)$            $\triangleright \text{clip by } \epsilon$ 
5:    $X_{sp} \leftarrow clip(X_{sp}, x_{max}, x_{min})$                  $\triangleright \text{clip by max/min of input data}$ 
6:   for  $X'$  in  $X_{sp}$  do
7:      $\xi_2 \leftarrow N(\mu, \sigma^2)$                                  $\triangleright \xi_2 = \text{random normal noise}$ 
8:      $X_{samples} \leftarrow X' + \mu_e * \xi_2$                      $\triangleright \text{get } N \text{ Entropy Samples}$ 
9:      $y_{pred} \leftarrow model.predict(X_{samples})$                $\triangleright y_{pred} = \text{class distribution}$ 
10:     $E.append(entropy(y_{pred}))$ 
11:  end for                                                     $\triangleright E \text{ now contains entropy values for each class distribution}$ 
12:   $max\_index \leftarrow argmax(E)$                                  $\triangleright \text{find highest value in } E$ 
13:   $X_{current} \leftarrow X_{sp}[max\_index]$                      $\triangleright \text{choose that Starting Point going forward}$ 
14: end for
```

larger numbers of steps and large batches of inputs. As you can see in Algorithm 1, there are two main **for** loops, and a sampling of the feature space happens inside of each loop. This means that the process does not scale easily to large batches of input data and can be memory intensive for data sets with large feature spaces. This is the main reason we limited some of our hyperparameter testing, as discussed in Chapter 4, Section 4.1.1.

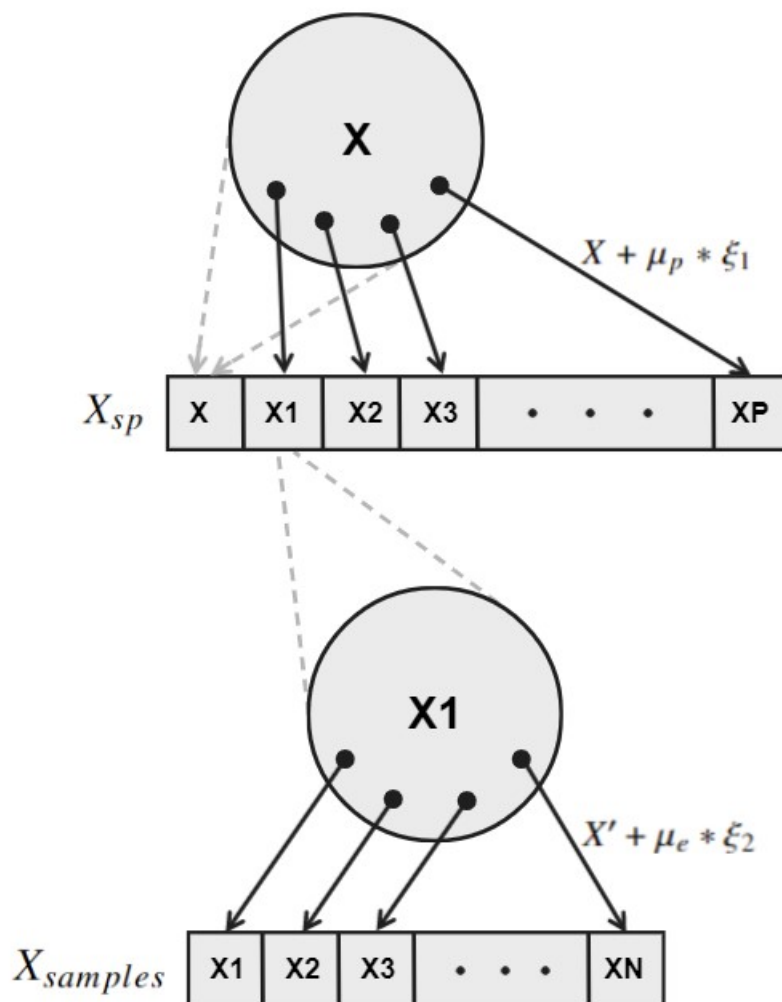


Figure 3.2. X is the current image – either the benign image at step 1 or the final result of a previous step. Random sampling around X , controlled by μ_p , gives P starting points (Algorithm 1, lines 2-3). The current image is always one of the starting points so the algorithm is not needlessly adding noise if the other starting points fail to produce a higher entropy. Random sampling around each starting point, controlled by μ_e , gives N entropy samples (Algorithm 1, lines 7-8).

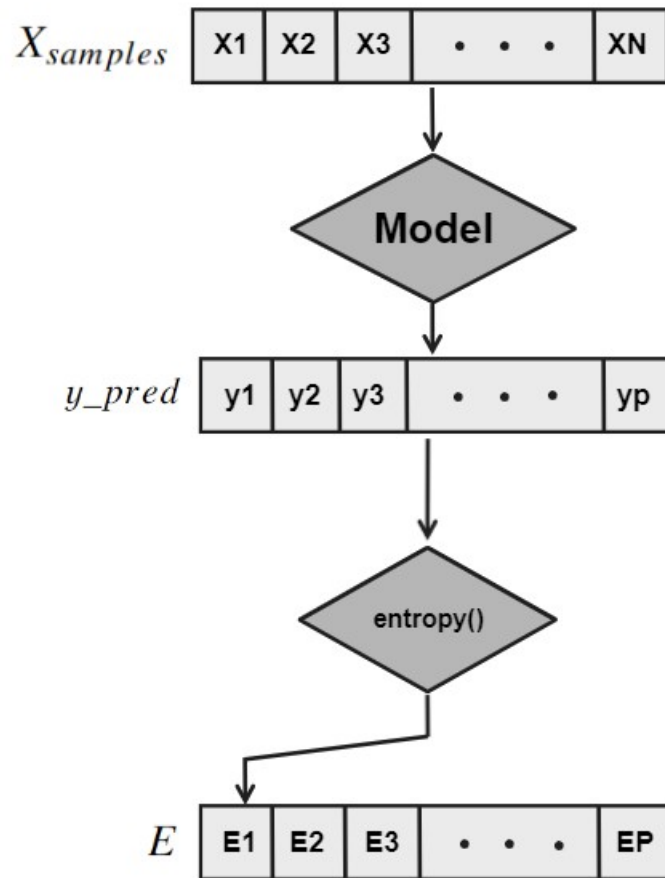


Figure 3.3. Entropy samples ($X_{samples}$) are passed through the Model to generate y_{pred} , the class distribution/predictions (Algorithm 1, line 9), which is fed through the entropy function (Algorithm 1, line 10). The entries in array E align with their corresponding starting points in array X_{sp} . The largest value in array E determines which starting point will be chosen as the beginning sample for the next step.

3.2 Methodology

To test our attack, we used two well-known datasets: CIFAR-10 and ImageNet. CIFAR-10 samples were pulled from the TensorFlow³ Keras Datasets module. ImageNet samples were pulled from the FastAI Imagenette dataset⁴, however, these images are now able to be pulled directly from the TensorFlow API using the Tensorflow_datasets package. Using well-known datasets is an easy and conventional way to test new attacks and defenses since there are an abundance of DNNs trained on them and most research involving neural network attacks and defenses use them.

3.2.1 CIFAR-10

CIFAR-10 is a collection of 32x32 color images, where each image belongs to one of 10 classes. It is commonly used in many machine learning applications to test computer vision algorithms since its low resolution allows researchers to quickly train DNNs for various purposes and to test new concepts. Our CIFAR-10 samples were pulled from the TensorFlow Keras Datasets module.

We tested the attack against four variations of CIFAR-10 DNNs, three of which were trained using some of the defensive techniques discussed in Chapter 2, Section 2.2. The baseline CIFAR-10 model we used was a Wide Residual Network (WRN) consisting of 38 layers and approximately 2.66 million trainable parameters. The variants of the model we used were able to achieve a high prediction accuracy against the CIFAR-10 dataset, shown in Table 3.1. The four variants of our CIFAR-10 WRN model, Vanilla, GR, AT, and TRADES are described below.

1. The Vanilla Model
 - No defensive techniques used in training.
2. The GR Model
 - Trained using Gradient Regularization as a defense.
 - Gradient regularization was discussed in Chapter 2, Section 2.2.4.
3. The AT Model
 - Trained using Adversarial Training as a defense.

³<https://www.tensorflow.org/>

⁴<https://github.com/fastai/imagenette>

- Adversarial training was discussed in Chapter 2, Section 2.2.2.

4. The TRADES Model

- Trained using TRADES as a defense.
- TRADES was discussed in Chapter 2, Section 2.2.3.

We utilized two versions of each of these four models for a total of eight models. All eight versions were architecturally identical and were all trained on CIFAR-10, however, they were all trained separately. For each variation of the model, one version was used during the adversarial example creation process (i.e. the gray box setting), then we tested the resulting adversarial examples on the second version of the model (i.e. the black box setting).

Table 3.1. Benign Accuracy of our CIFAR-10 Models

Model	Defense Technique	Top 1 Accuracy
Vanilla	None	91%
GR	Gradient Regularization	91%
AT	Adversarial Training	92%
TRADES	TRADES	87%

3.2.2 ImageNet

The original ImageNet dataset consists of 10,000,000+ images depicting 10,000+ object categories. However, the training dataset that is used in many ML and computer vision competitions, like the ImageNet Large Scale Visual Recognition Challenge (ILSVRC), is a subset of the original dataset and consists of approximately 1.2 million images divided into 1,000 categories. We used a smaller version of the ImageNet dataset, called Imagenette, which consists of images from only 10 categories that are easily classifiable by the DNNs we tested. This small subset of the ImageNet dataset is useful for testing new ideas, algorithms, and experiments since doing so on the entire ImageNet dataset is very time consuming and resource intensive. The images we used were 244x244 color images. Since our attack turned out to be quite resource intensive, we limited our dataset to 100 images from the Imagenette dataset, 10 images from each of the 10 classes. The accuracy numbers reported in this thesis are all based on this 100-image subset.

We created our ImageNet adversarial examples on a well-known ImageNet DNN: ResNet50,

first proposed by He et al. in [26]. ResNet50 consists of 50 network layers and just over 25.5 million trainable parameters. For the gray box attack setting, we tested the resulting adversarial examples back on the same ResNet50. Additionally, we tested the transferability of our adversarial examples against three other well-known DNNs: MobileNetV2⁵, VGG16⁶, and DenseNet201⁷ for the black box attack. These models, along with their pretrained ImageNet weights, are available directly through the TensorFlow API using the Keras Applications module. Using these ImageNet models and our Imagenette data set, Table 3.2 shows our baseline accuracy scores against benign images.

Table 3.2. Benign Accuracy of our ImageNet Models

Model	Top 1 Accuracy	Top 5 Accuracy
ResNet50	81%	94%
MobileNetV2	83%	98%
VGG16	73%	89%
DenseNet201	86%	96%

3.2.3 Comparing to Other Attacks

We tested all of our CIFAR-10 and ImageNet models against the Boundary Attack from [12], discussed in Chapter 2, Section 2.1.3. We utilized the Adversarial Robustness Toolbox (ART)⁸ Evasion Attacks module to implement the Boundary Attack. This module allows users to implement both the targeted and non-targeted versions of the attack; we opted to use the non-targeted version.

The Boundary Attack is also a black box decision based attack and thus uses a similar methodology to our attack, so we felt like a comparison of results between the two attacks would be the most reasonable to make. The difference between the Boundary Attack and HEIS is that the Boundary Attack starts with an adversarial example with a large perturbation that already classifies incorrectly and then iteratively reduces the distance between the adversarial example and a benign image (i.e. working from the outside in, you might say),

⁵MobileNetV2: first proposed by Sandler, Howard, Zhu, Zhmoginov, and Chen in [27]

⁶VGG16: first proposed by Simonyan and Zisserman in [28]

⁷DenseNet201: first proposed by Huang, Liu, Maaten, and Weinberger in [29]

⁸<https://adversarial-robustness-toolbox.org/>

while HEIS starts with a benign image and iteratively works its way "outward" to find an adversarial example that produces the highest entropy.

CHAPTER 4: Results

In this Chapter we will discuss and analyze the results of running HEIS on the DNNs and datasets discussed in Chapter 3. In the early stages of testing, we used the CIFAR-10 dataset to help guide our selection of meaningful hyperparameters to see what variations caused the most impact on the performance of the model against the adversarial examples. After thoroughly testing the attack on CIFAR-10, we tested it against the ImageNet DNNs. Our target ImageNet DNN, ResNet50, is larger and more complex than our CIFAR-10 models. Due to this size difference and more complex input data (244x244 images as opposed to 32x32), it took much longer to run the attack against ResNet50 than it did against the CIFAR-10 models; for this reason we did not explore as much of the hyperparameter space while testing against ImageNet.

4.1 HEIS and CIFAR-10

We will first discuss the results of the hyperparameter testing we did for the CIFAR-10 models, then analyze the performance of HEIS against the undefended and defended models covered in Section 3.2.1. We utilized the undefended Vanilla model to test how each hyperparameter affected the performance of the adversarial examples created by HEIS. Once we decided on a set of hyperparameters, we tested HEIS against the Vanilla model and the models that incorporated defenses. We discovered that HEIS performs very well in a gray box setting and produces adversarial examples that are extremely effective in a black box setting as well, with high transferability to other models. In our gray box testing, HEIS performance was similar to, but slightly beaten by the Boundary Attack from [12]. In our black box testing, HEIS outperformed the Boundary Attack in all of our experiments.

4.1.1 Hyperparameter Testing

Recall from Chapter 3 that HEIS takes S Number of Steps, sampling from the input images at two different times: once to obtain P Starting Points, and then again to take N samples around each Starting Point. The first sampling is controlled by the Starting Point Noise Coefficient, μ_p , while the second is controlled by the Entropy Noise Coefficient, μ_e . We

intentionally used a fixed μ_e value of 0.1, which we found was sufficient to obtain reasonably diverse samples. The value for μ_p , however, controlled how different the Starting Points were allowed to be from the original data. A larger value for μ_p meant that the Starting Points, in terms of the average pixel distance, could be further away from the original sample. We wanted to test how varying this coefficient effected the overall performance of the resulting adversarial examples, as we would like to have adversarial examples that are not too different from the original image, but large enough to see the model’s accuracy drop. Our initial intuition was that a value of 0.1 would be sufficient, however we wanted to see what would happen if we used larger and smaller values. We tested several values for μ_p on the Vanilla model and, as you can see in Figure 4.1, we found that smaller μ_p values decreased the performance of the attack as expected, resulting in the model being able to classify the inputs at a correspondingly better rate, but larger coefficients did not improve performance enough to justify the added noise in the resulting adversarial examples. For this reason, we utilized a μ_p of 0.1 for the rest of our testing.

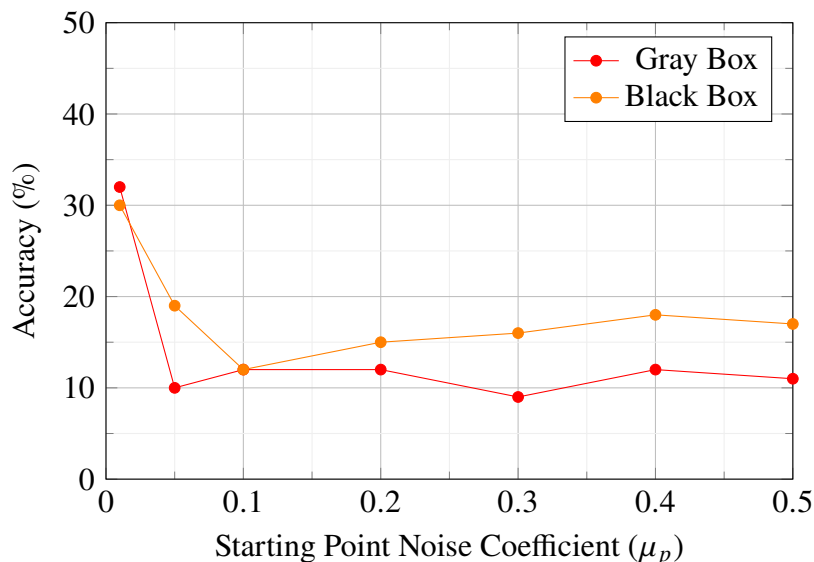


Figure 4.1. Vanilla Accuracy for $\epsilon = 0.3$

Next, we tested the effect of increasing or decreasing P , the number of Starting Points. Increasing this hyperparameter allows the algorithm to take more samples from the hyperball surrounding the input in hopes of finding the perfect one with the highest entropy to utilize in future steps. We hypothesized that increasing P would cause a corresponding increase in

the effectiveness of the resulting adversarial examples (meaning a drop in model accuracy). However, we discovered that increasing P only resulted in increased computation times and no desirable increase in attack performance (see Figure 4.2). For this reason, we used $P = 10$ for the rest of our testing¹.

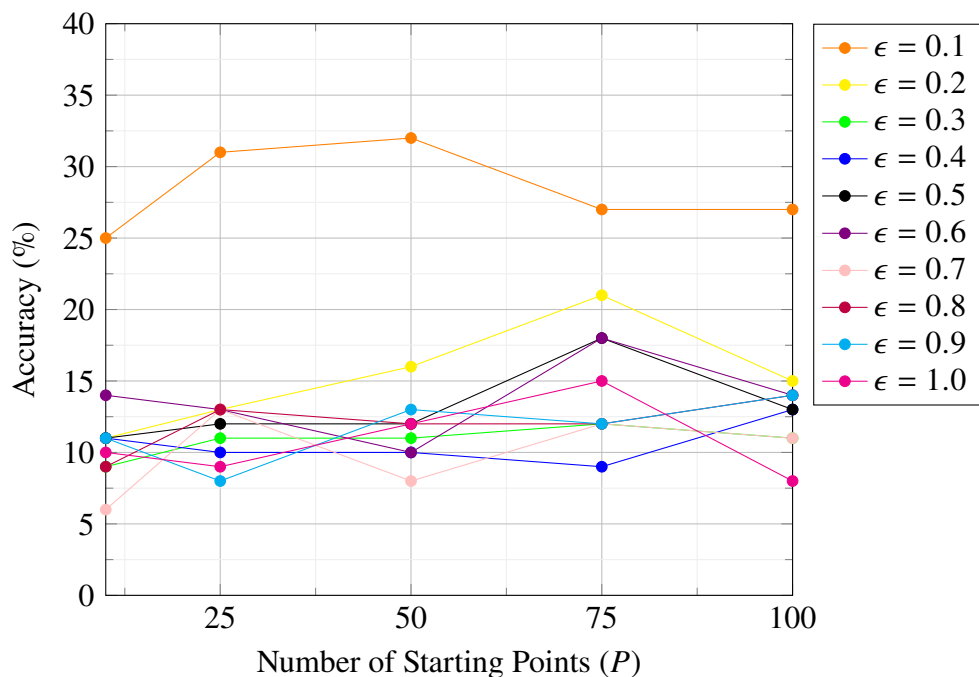


Figure 4.2. Vanilla Accuracy vs P

4.1.2 Undefended CIFAR-10 Performance

After settling on a baseline set of hyperparameters for HEIS, we ran the attack against the undefended CIFAR-10 Vanilla model. The Boundary Attack [12] performed better in gray box attacks when HEIS was constrained to ϵ values less than 0.2, but the performance of both attacks were similar when HEIS was allowed ϵ values of 0.2 and larger, with the Boundary Attack slightly outperforming HEIS. Our gray box attack results, shown in Figure 4.3, show that HEIS reduced Vanilla’s accuracy from 91% to $\sim 10\%$ for $\epsilon \geq 0.2$, compared to the Boundary Attack’s reduction to $\sim 7\%$. The Figures in this Section also show

¹We performed similar tests for our ImageNet Convolutional Neural Network (CNN) with similar results (see Section 4.2).

the "benign" data points – the Vanilla model’s accuracy against the original unperturbed CIFAR-10 dataset.

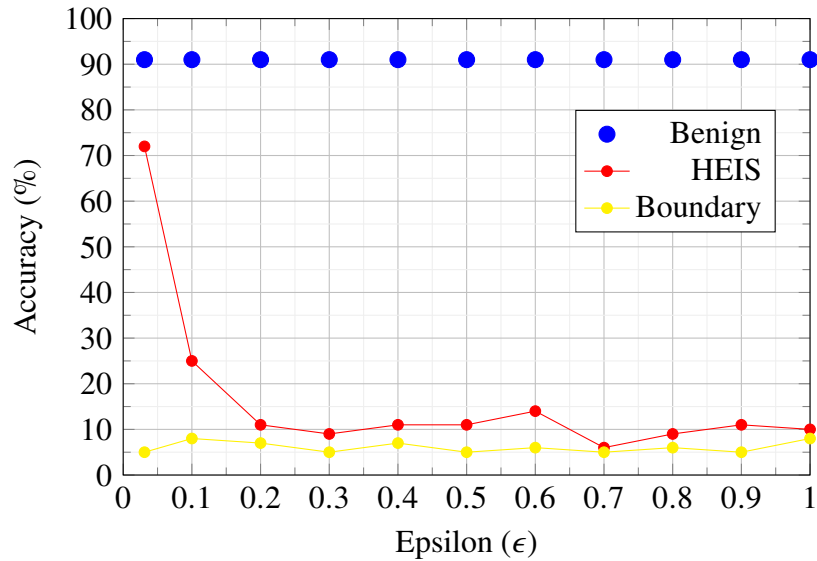


Figure 4.3. Vanilla Gray Box Accuracy

Although the Boundary Attack worked very well in gray box settings, the adversarial examples that it created were very specific to the network on which they were created. As shown in Figure 4.4, the Boundary Attack adversarial examples had almost no effect on the Vanilla model in a black box setting, despite the model having the exact same architecture and training set as the Vanilla model that was used to create them. HEIS was able to reduce black box accuracy from 91% to 15-20% for $\epsilon \geq 0.2$, while the Boundary Attack’s performance indicated almost no adverse effect and remained around 90%.

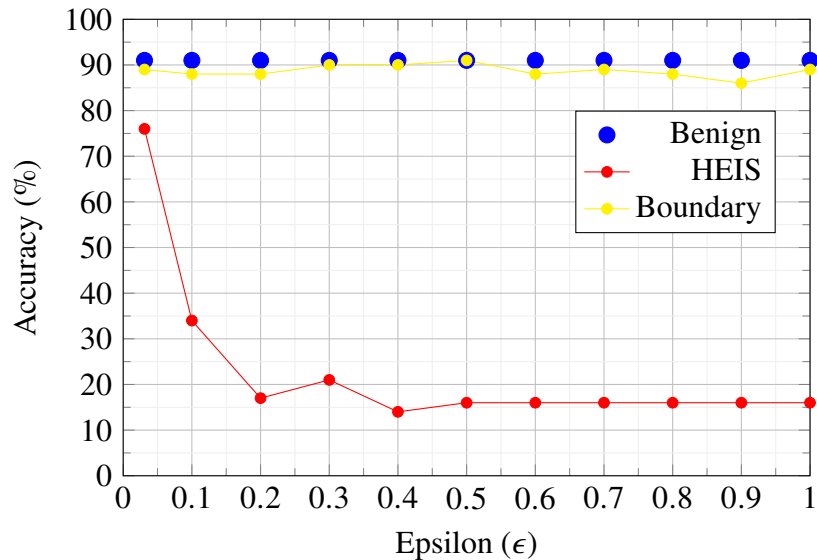


Figure 4.4. Vanilla Black Box Accuracy

4.1.3 CIFAR-10 Adversarial Examples

During testing, we investigated several instances of adversarial examples created using HEIS to see what they looked like when depicted visually, and also to examine model performance against single images to see if the desired effects were taking place. As predicted, we observed the confidence scores spreading out over multiple classes. For example, in Figures 4.6-4.11, HEIS not only caused the model to misclassify the image, but also created high uncertainty across all of the model outputs. Tables 4.1 and 4.2 show the network outputs for each individual classification category.

To see if this increase in class distribution entropy held true for the rest of the dataset, we calculated the average entropy across all of the adversarial examples at each step and plotted it in Figure 4.5. For most ϵ values, the entropy over class predictions had a steep increase from 38% to over 60% during the first 30 steps of the HEIS algorithm. After 30 steps, the entropy began to plateau with only slight increases after that. This indicates the algorithm indeed finds examples that exist near to adjacent decision boundaries of the model. We also saw this trend when we examined the entropy values for our ImageNet adversarial examples in Section 4.2. Since this plateau effect was common to both datasets, we decided to explore the effectiveness of HEIS for smaller values for S (number of steps) on our ImageNet DNN,

the results of which are covered in Section 4.2.2. The implication is very promising for the attack; if HEIS remains effective for significantly smaller values of S , then that bodes well for computational overhead and real-world applications for this iterative, multi-sampling process.

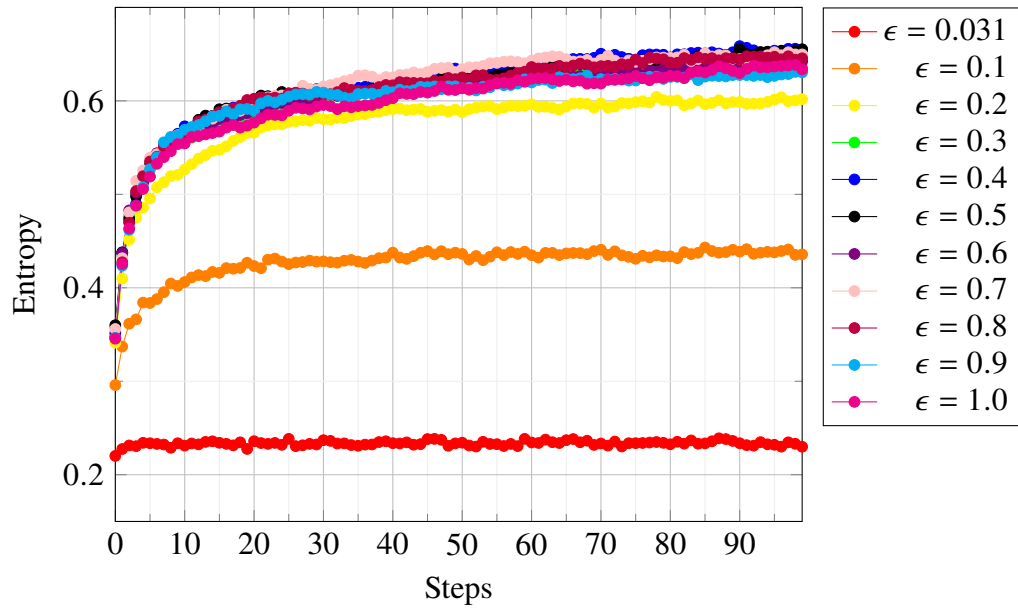


Figure 4.5. Average Entropy Values (TRADES model)

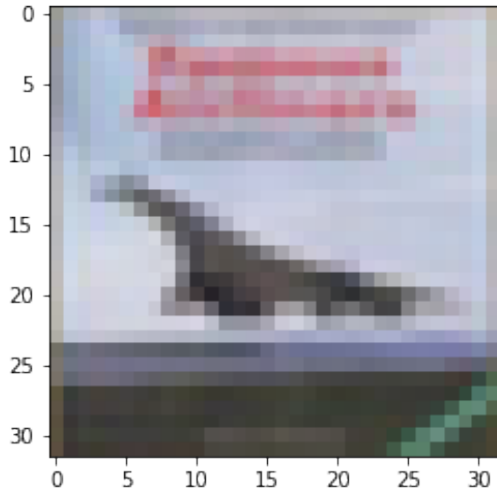


Figure 4.6. Airplane: 99.42%

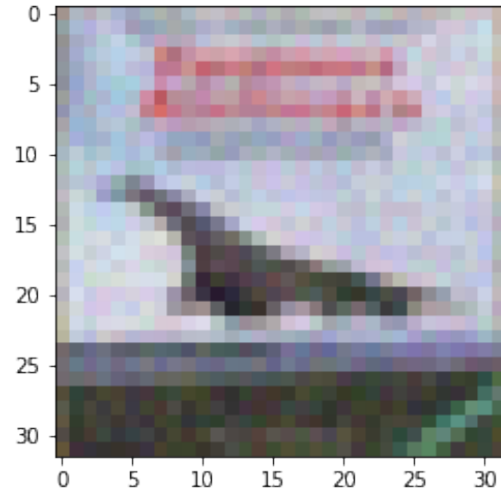


Figure 4.7. Bird: 82.81%
 Airplane: 5.89%, Ship: 4.73%
 $\epsilon = 0.031$

Table 4.1. CIFAR-10 airplane – before and after HEIS

Label	Benign	HEIS ($\epsilon = 0.031$)
Airplane	99.4247%	5.8900%
Automobile	0.0025%	0.1165%
Bird	0.1535%	82.8113%
Cat	0.0148%	1.2953%
Deer	0.0030%	1.1949%
Dog	0.0025%	1.4807%
Frog	0.0105%	0.6862%
Horse	0.3024%	0.5213%
Ship	0.0769%	4.7309%
Truck	0.0093%	1.2731%

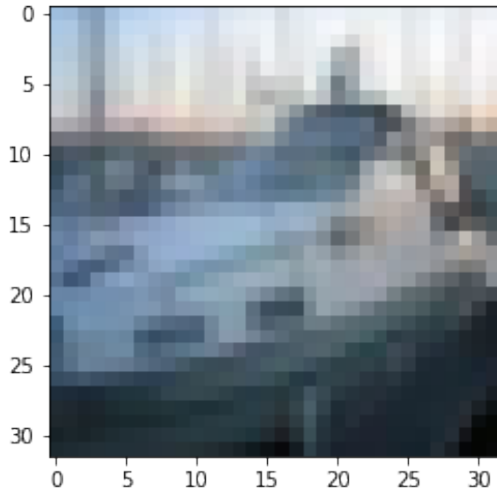


Figure 4.8. Ship: 85.63%
Automobile: 13.79%

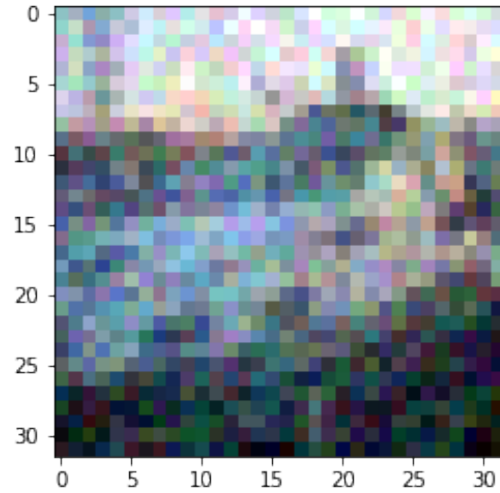


Figure 4.9. Cat: 65.06%
Frog: 20.86%, Deer: 7.89%
 $\epsilon = 0.1$

Table 4.2. CIFAR-10 ship – before and after HEIS

Label	Benign	HEIS ($\epsilon = 0.1$)
Airplane	0.1122%	0.8538%
Automobile	13.7933%	0.5392%
Bird	0.1218%	3.6012%
Cat	0.0556%	65.0599%
Deer	0.0673%	7.8953%
Dog	0.0151%	0.2392%
Frog	0.0808%	20.8602%
Horse	0.0085%	0.0721%
Ship	85.6333%	0.6374%
Truck	0.1122%	0.2418%

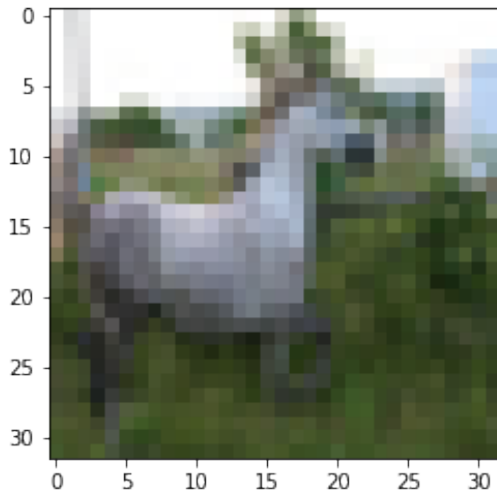


Figure 4.10. Horse: 99.45%

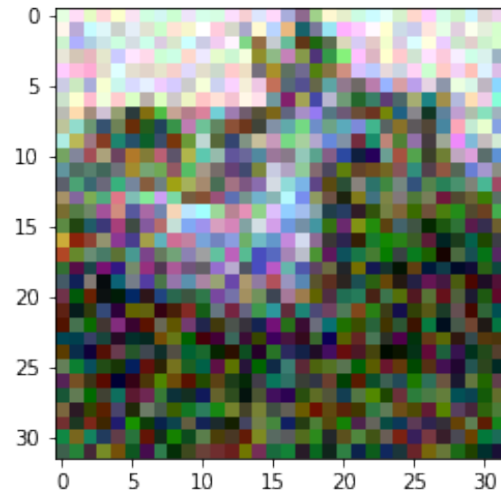


Figure 4.11. Cat: 76.19%
 Frog: 17.57%
 Deer: 4.61%
 $\epsilon = 0.2$

Table 4.3. CIFAR-10 horse – before and after HEIS

Label	Benign	HEIS ($\epsilon = 0.2$)
Airplane	0.0953%	0.5714%
Automobile	0.0119%	0.1172%
Bird	0.0324%	0.4680%
Cat	0.0482%	76.1925%
Deer	0.1704%	4.6138%
Dog	0.1590%	0.0479%
Frog	0.0047%	17.5749%
Horse	99.4500%	0.0100%
Ship	0.0110%	0.0176%
Truck	0.0171%	0.3869%

4.1.4 Defended CIFAR-10 Performance

For the defended models under gray box attack, the defensive techniques reduced the performance of HEIS, while the Boundary Attack [12] did not suffer as much loss in performance. However, for black box attacks, HEIS outperformed the Boundary Attack when testing the adversarial examples against our defended models for every value of ϵ except 0.031, where the performance was nearly identical. Recall from Section 3.2.1, the three defenses which we tested HEIS against were:

1. Gradient Regularization (GR model)
2. Adversarial Training (AT model)
3. TRADES (TRADES model)

As you can see in Figure 4.12, HEIS was able to drop the performance of the Vanilla model from 91% to 11% for $\epsilon \geq 0.2$ while the defended models were able to maintain accuracy scores in the 20-30% range, only dropping below 20% for $\epsilon \geq 0.6$. The Boundary Attack did not suffer such setbacks in gray box settings (Figures 4.14, 4.16, and 4.18) and was still able to achieve ~8-10% accuracy against all models. In black box tests (Figure 4.13), HEIS achieved accuracy drops from 91% to 17% on the Vanilla model for $\epsilon \geq 0.2$, while the defended models finally dipped below 30% accuracy for $\epsilon \geq 0.4$. However, the Boundary Attack adversarial examples had almost no effect against the black box defended models, and we observed only slight performance drops in Figures 4.15, 4.17, and 4.19, where the models were still able to achieve an accuracy above 80% for every test.

Although hindered, HEIS still worked quite well on our defended models and showcases how effective black box attacks can be even against neural network defenses. Our GR model dropped from 90% accuracy to $\leq 32\%$ for $\epsilon \geq 0.2$ in our gray box tests and $\leq 23\%$ for $\epsilon \geq 0.4$ in our black box tests. The AT model dropped from 90% accuracy to $\leq 36\%$ for $\epsilon \geq 0.2$ in gray box testing and $\leq 25\%$ in black box testing for $\epsilon \geq 0.4$. The TRADES model dropped from 87% gray box accuracy to $\leq 33\%$ for $\epsilon \geq 0.2$ and $\leq 31\%$ black box accuracy for $\epsilon \geq 0.3$.

As a side note, the adversarial examples produced from running HEIS on these defended models appeared extremely similar to the ones that are displayed in Sections 4.1.3 and A.1.1, so we do not show them in this thesis.

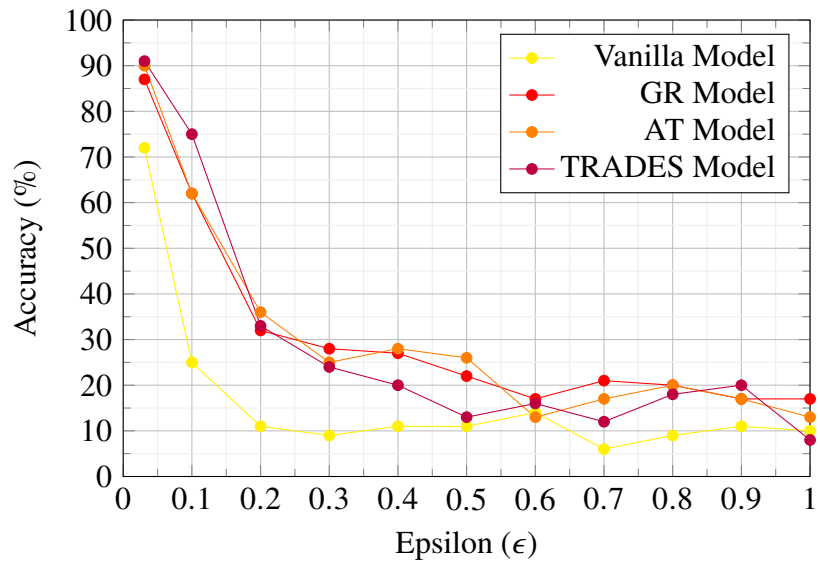


Figure 4.12. HEIS Gray Box Accuracy

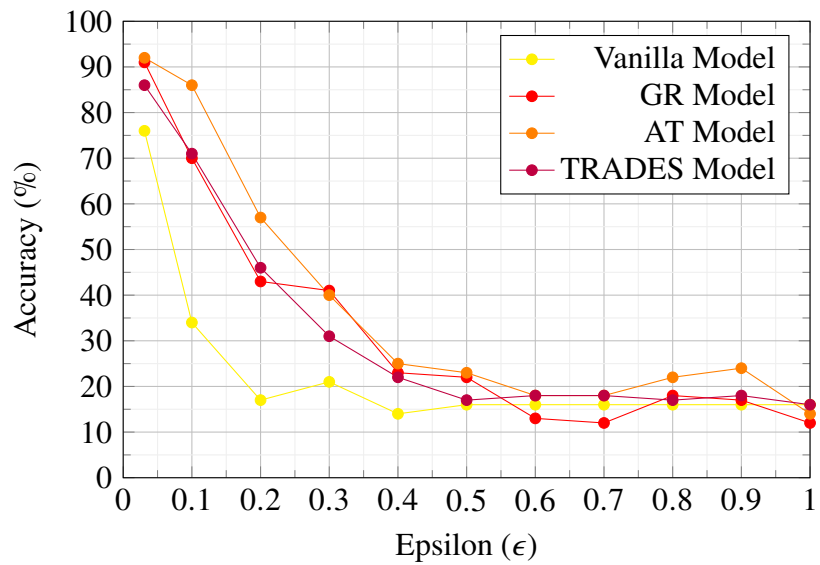


Figure 4.13. HEIS Black Box Accuracy

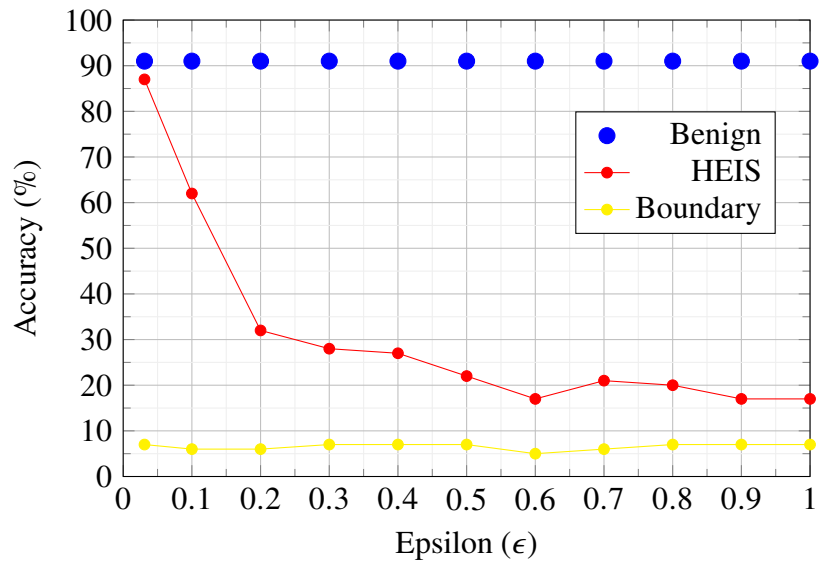


Figure 4.14. HEIS Gray Box Accuracy (GR Model)

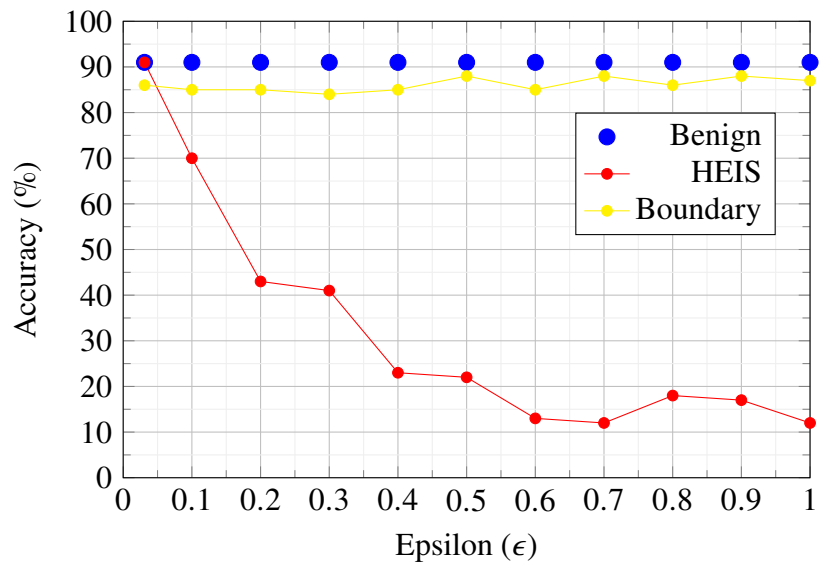


Figure 4.15. HEIS Black Box Accuracy (GR Model)

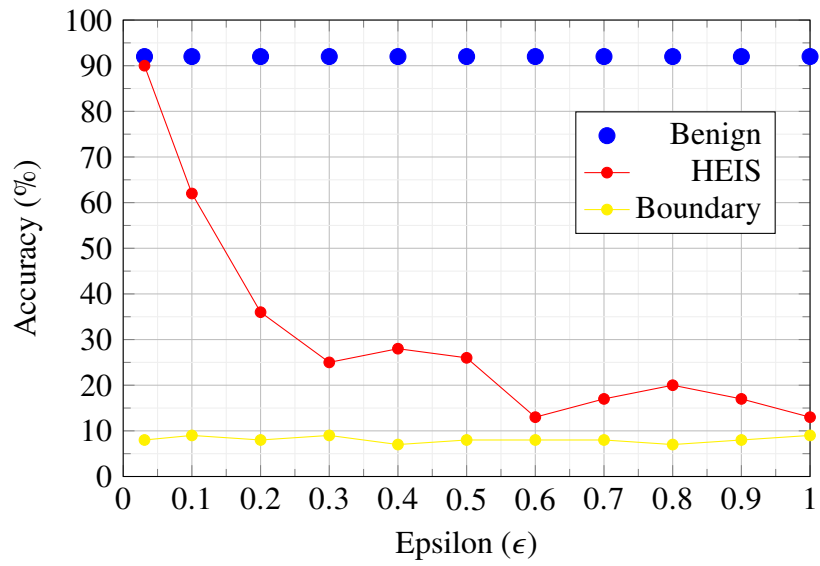


Figure 4.16. HEIS Gray Box Accuracy (AT Model)

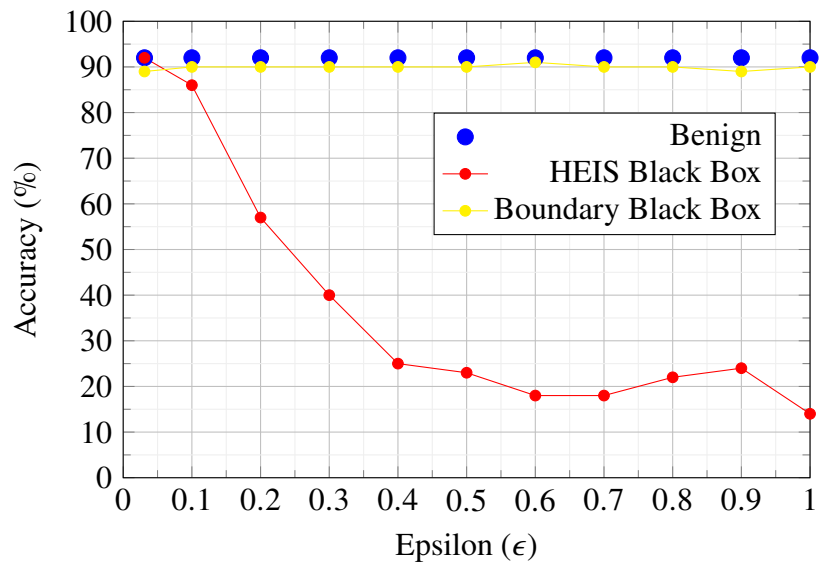


Figure 4.17. HEIS Black Box Accuracy (AT Model)

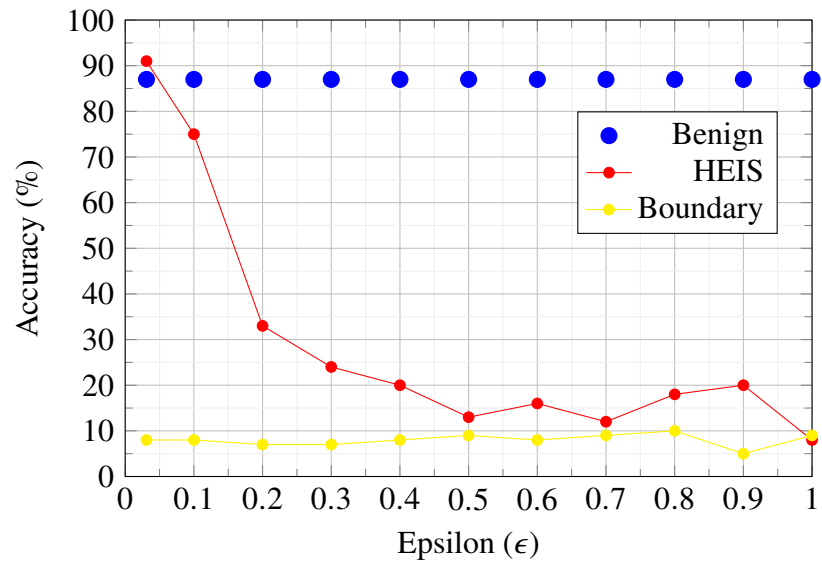


Figure 4.18. HEIS Gray Box Accuracy (TRADES Model)

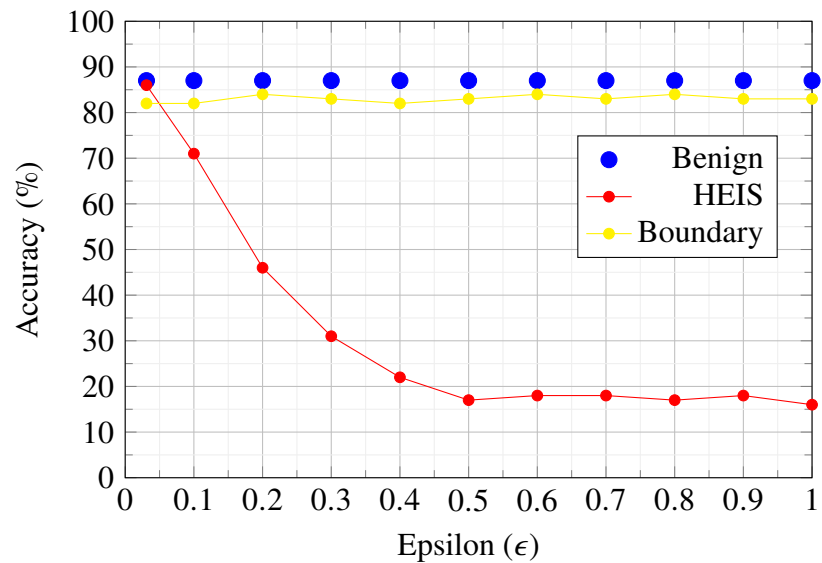


Figure 4.19. HEIS Black Box Accuracy (TRADES Model)

4.2 HEIS and ImageNet

In this section, we discuss the performance of HEIS against the ImageNet dataset. We used ResNet50 (from [26]) to create the adversarial examples, then tested their transferability against three other DNNs trained on ImageNet: MobileNetV2 from [27], VGG16 from [28], and DenseNet201 from [29]. We also tested the Boundary Attack [12] against ResNet50 in order to give a side-by-side comparison of performance against a known adversarial attack. Similarly to our CIFAR-10 testing, the Boundary Attack performed better against the target model during gray box tests in top-1 accuracy. However, we still observed significant performance degradation in both top-1 and top-5 accuracy² using HEIS adversarial examples for epsilon (ϵ) values as low as 0.2.

Since we only had one version of the ResNet50 model with preloaded ImageNet weights, we were not able to do the same black box testing we did before for the CIFAR-10 models where we had two separately trained models with identical architecture. Instead, we tested the HEIS and Boundary Attack adversarial examples created on ResNet50 against three other ImageNet DNNs to see how transferable they were. We discovered that the adversarial examples produced by HEIS transferred very well to the other ImageNet models and observed significant drops in top-1 and top-5 accuracy among all three of the other models, even for ϵ values as low as 0.2. This was not the case for the Boundary Attack adversarial examples, as they were only effective against the original target model, and had very little effect on the other ImageNet models. Section 4.2.3 explores the results of this transferability testing.

4.2.1 HEIS Performance

After running HEIS and the Boundary Attack against ResNet50, we noticed familiar trends in the performance of each. The Boundary Attack adversarial examples excelled against the model by avoiding the correct output class, although interestingly they only significantly reduced the top-1 accuracy while the top-5 accuracy remained virtually undisturbed. The HEIS adversarial examples, as hypothesized, were effective in increasing overall uncertainty by spreading out the model’s confidence scores across multiple classification outputs and were thus quite effective in reducing both top-1 and top-5 performance. In other words, HEIS

²Top-1 and top-5 accuracy refer to the highest (top-1) and five highest (top-5) class predictions the model produces for a given input.

was effective in not only obscuring the single correct output class, but produced adversarial examples that had such high uncertainty across enough classes that it even effected top-5 performance. This is exactly what we hoped to accomplish with HEIS and we think that these high entropy classification outputs cause the resulting adversarial examples to transfer well to other ImageNet models.

As you can see in Figure 4.20, HEIS was able to drop the performance of ResNet50 from 81% top-1 accuracy to below 50% for $\epsilon \geq 0.2$. For $\epsilon \geq 0.4$, the accuracy drops to ~20% for top-1 and ~30% for top-5. In contrast, the Boundary Attack only significantly dropped top-1 accuracy to ~9%, while the top-5 accuracy of ~92% is nearly unchanged from the model's performance on the original, unperturbed dataset.

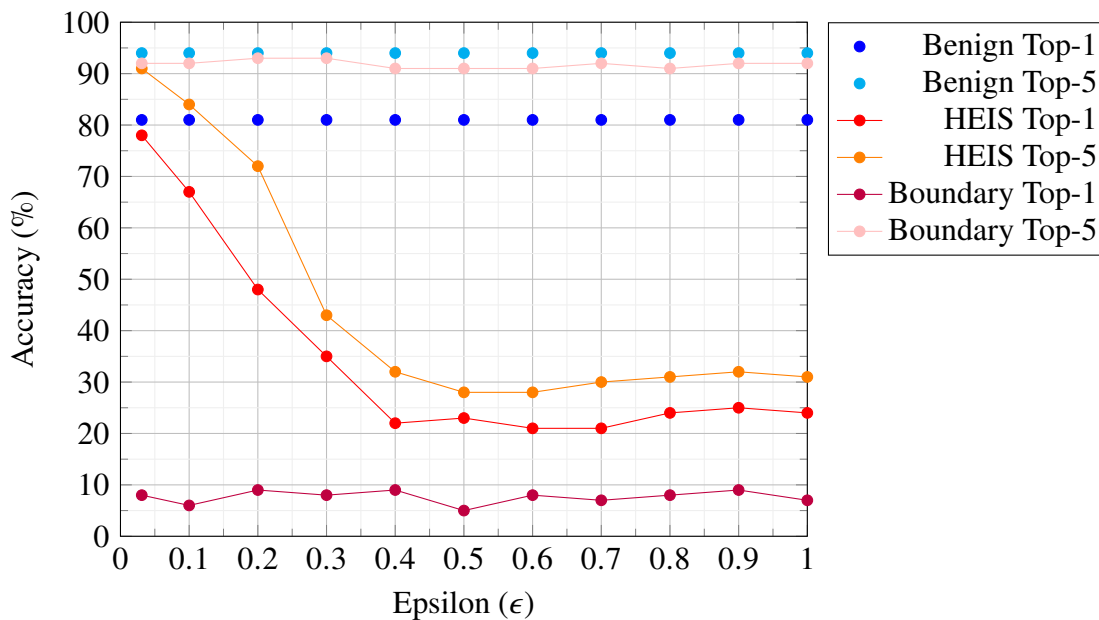


Figure 4.20. ResNet50 Gray Box Accuracy

An interesting trend to note in Figure 4.20 is that as ϵ values go above 0.4, the HEIS adversarial examples stop negatively impacting the accuracy of the models and the performance plateaus. This is also noticeable in the CIFAR-10 testing we performed if you examine Figures 4.12 and 4.13. This also holds true in Section 4.2.3 when we discuss the transferability of our ImageNet adversarial examples to other models. As we can see in Sections 4.1.3 and 4.2.2, when displaying the adversarial images and observing them with

the human eye, the images created using higher ϵ values are obviously noisier and have more visible perturbations than those with lower ϵ values (for instance, when comparing adversarial examples at $\epsilon = 0.1$ and $\epsilon = 0.5$). So why do the models not perform worse when ϵ values go above 0.4? We will hypothesize the reason behind this in Section 5.1.

Since ImageNet is a larger dataset, having 244x244 color images versus 32x32 colors images for CIFAR-10, and 1,000 classes instead of just 10, we wanted to run a quick test on the effect of increasing P , the number of starting points, to see if it helped HEIS achieve better results against ResNet50. Table 4.4³ shows the results of those tests, which were surprisingly similar to our CIFAR-10 test results. Increasing P had almost no effect on the performance of HEIS; perhaps more testing using higher values of P would have yielded different results, but this would have meant significantly increased computation time so we did not test any values above 100. For these reasons, we used $P = 10$ for the rest of our ImageNet testing.

Table 4.4. ResNet50 accuracy for $\epsilon = 0.1$.

P	Top 1 Accuracy	Top 5 Accuracy
10	73%	82%
25	70%	83%
50	70%	83%
100	73%	82%

4.2.2 ImageNet Adversarial Examples

In this section we explore what our ImageNet adversarial examples look like and what kind of response we get from ResNet50 when making predictions against individual samples. Figures 4.21 and 4.22 show an example of an image before and after running HEIS at $\epsilon = 0.1$. As you can see, while the perturbations did cause ResNet50 to miss the top-1 category, it was still able to get the correct class in the top-5. This is obviously an image that ResNet50 struggles with, as even its benign prediction confidence is only 35%, but it is still an interesting case as you can clearly see in Table 4.5 that the predictions after HEIS have increased the model’s confidence in almost every category except the ones it was the most sure of before.

³Recall from Section 3.2.2 that ResNet50’s baseline accuracy against this dataset is 81% for top-1 and 94% for top-5.

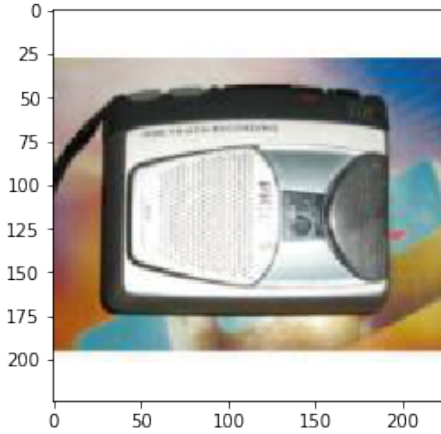


Figure 4.21. Cassette: 35.13%
 Spotlight: 18.42%
 Car mirror: 12.37%

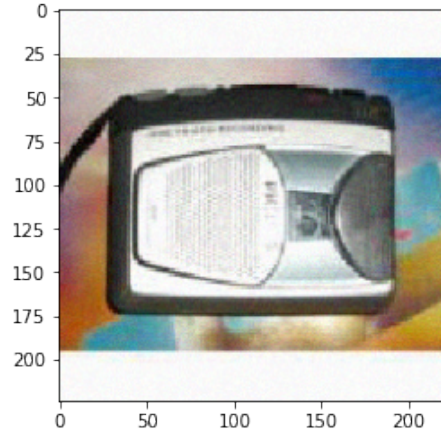


Figure 4.22. Polaroid: 36.25%
 Cassette: 22.12%
 Microwave: 15.31%
 $\epsilon = 0.1$

Table 4.5. ImageNet cassette player – before and after HEIS

Label	Benign	HEIS ($\epsilon = 0.1$)
Cassette player	35.1274%	22.1206%
Spotlight	18.4222%	5.5675%
Car mirror	12.3761%	0.5946%
Polaroid camera	7.3647%	36.2499%
Loupe	3.3186%	0.3931%
Microwave	0.4415 %	15.3066%
Television	0.2580%	1.9116%
Space heater	0.0665%	1.8408%
Oscilloscope	0.1602%	1.7111%
Radio	0.4684%	1.2323%

In Figures 4.23-4.30, we see that as we increase ϵ above 0.2, HEIS starts causing ResNet50 to become increasingly unconfident in predictions it was quite sure of before, in some cases even causing the DNN to begin crossing interesting decision boundaries; such as from long metallic objects like chainsaws and space shuttles to canines, or from canines to fowl, or

from buildings to animals, etc. More ImageNet HEIS adversarial examples can be seen in Appendix A, Section A.1.2.

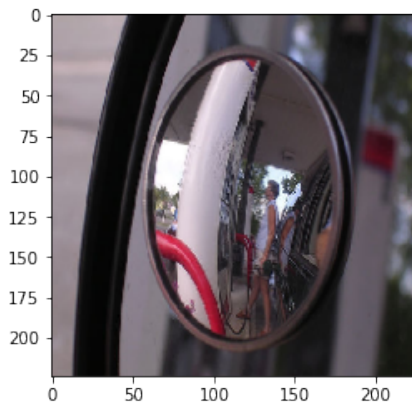


Figure 4.23. Car mirror: 93.31%
Sunglasses: 2.04%
Grille: 0.68%

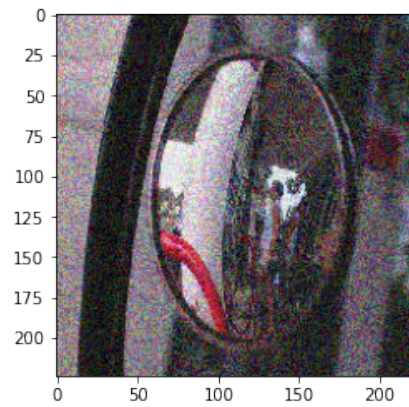


Figure 4.24. Iron: 17.24%
Car mirror: 14.92%
Disk Brake: 9.46%
 $\epsilon = 0.2$

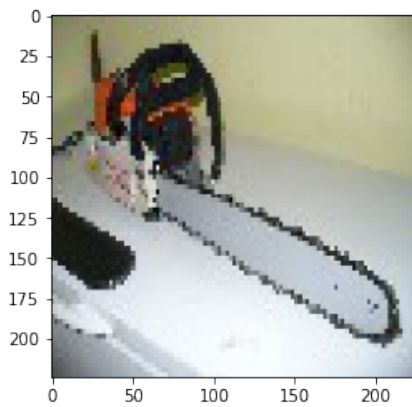


Figure 4.25. Chainsaw: 50.04%
Space shuttle: 24.26%
Projectile: 2.01%

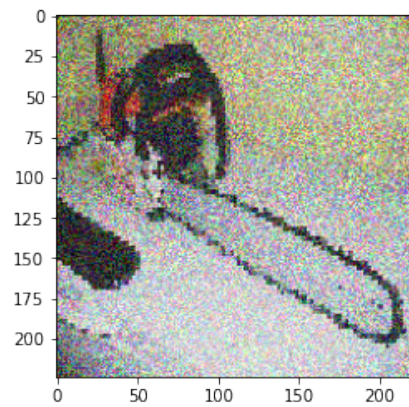


Figure 4.26. Briard: 13.28%
Chainsaw: 11.31%
Schipperke: 9.58%
 $\epsilon = 0.3$

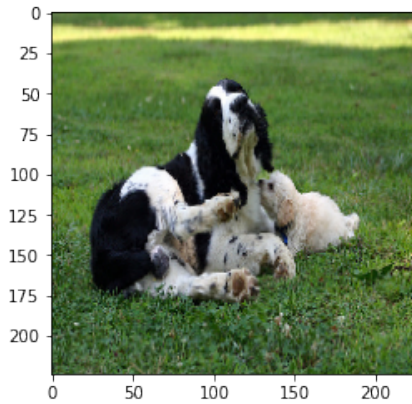


Figure 4.27. Cocker spaniel: 51.68%
 Springer: 17.22%
 Poodle: 13.41%

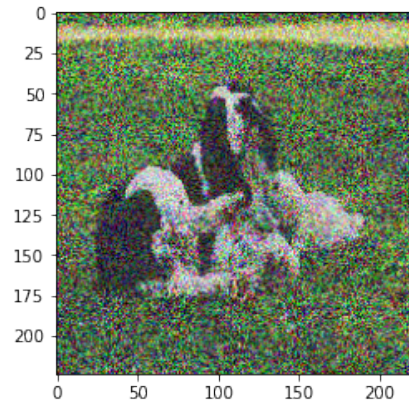


Figure 4.28. Goose: 19.24%
 Swan: 9.55%
 Badger: 7.73%
 $\epsilon = 0.4$

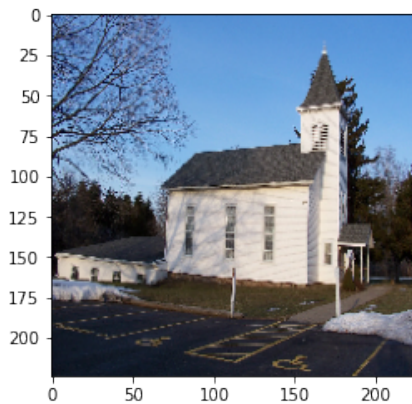


Figure 4.29. Church: 95.51%
 Monastery: 1.05%

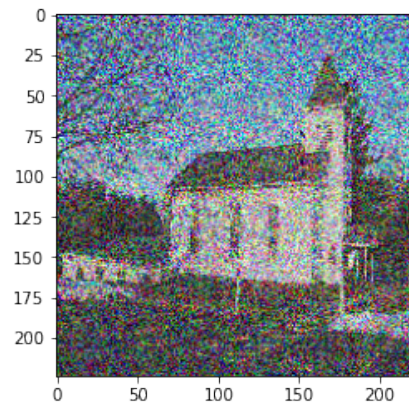


Figure 4.30. Hare: 30.92%
 Fountain: 27.22%
 Stone wall: 6.59%
 $\epsilon = 0.5$

Just like we did for CIFAR-10 in Section 4.1, we wanted to see if this uncertainty trend held true for our entire ImageNet dataset. We calculated the average entropy across all of

the adversarial examples at each step and plotted it in Figure 4.31. Again we see a definite upward trend in the average entropy, indicating that even for this more complex model and dataset – HEIS indeed finds examples that exist near to adjacent decision boundaries. However, just like our CIFAR-10 testing, it ceases to increase much after 30-40 steps. Since this plateau effect was common across both datasets, this caused us to wonder if maybe HEIS could perform just as well with significantly lower step counts. If so, the attack becomes much more computationally feasible and realizable for real world applications, as there may be limits on how often systems allow high frequency interactions of the type common to black box attacks.

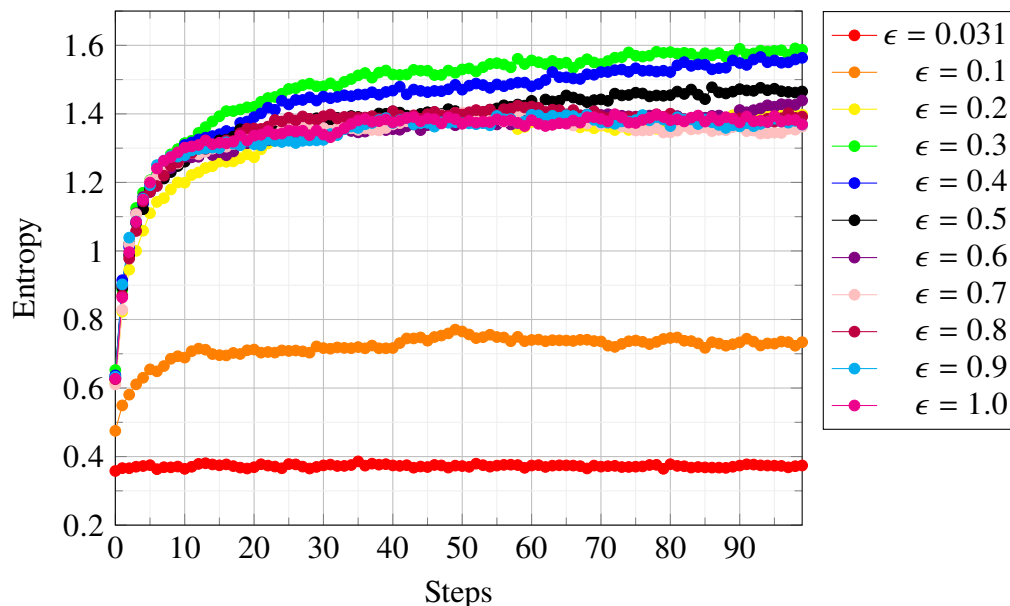


Figure 4.31. Average Entropy Values (ResNet50)

We tested HEIS ($\epsilon = 0.5$) against ResNet50 for smaller step sizes. We found that while it did give us our expected decrease in computation time, as you can see in Figure 4.32 there is a positive correlation between the effectiveness of the adversarial examples and the number of steps HEIS is allowed to take. Although it is worthwhile to point out that even with drastically reduced step counts, the ResNet50 HEIS adversarial examples still caused significant drops in performance among all four models. For example, at only five steps, ResNet50 dropped from 81% top-1 accuracy down to 48%. Those same adversarial examples then caused MobileNetV2 to drop from 83% to 30% in top-1 accuracy. At 15 steps,

ResNet50 dropped to 36% top-1 accuracy, and testing those adversarial examples against VGG16 dropped its performance from 73% top-1 accuracy down to 34%. DenseNet201, our top ImageNet performer, finally dropped below 50% top-1 accuracy (from 86%) for $S \geq 25$. The other positive outcome when dropping the number of steps, besides decreased computation time, is that it results in adversarial examples that are visibly less perturbed when viewed with the human eye (see Figure 4.33).

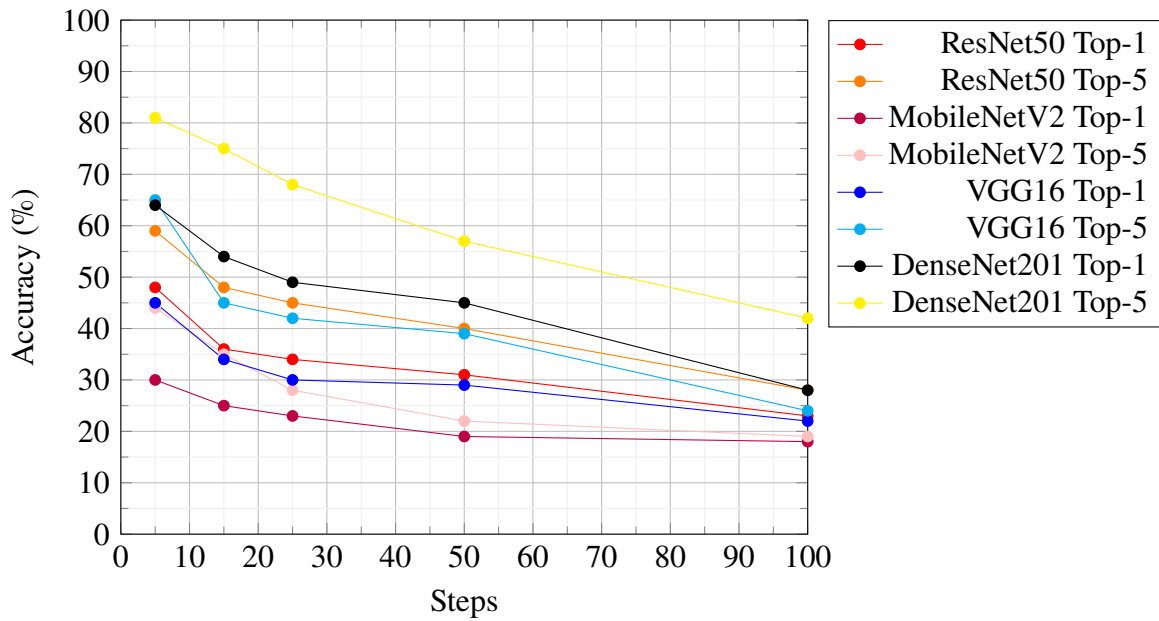


Figure 4.32. ImageNet Accuracy (all models) for $\epsilon = 0.5$

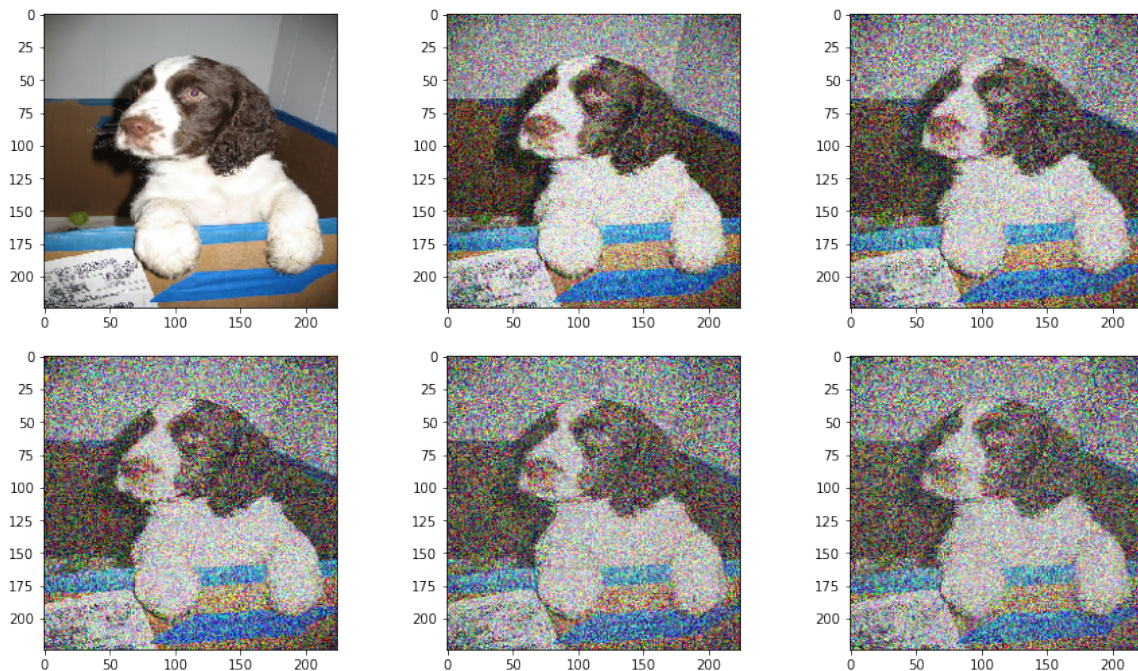


Figure 4.33. HEIS adversarial examples produced for increasing values of S ($\epsilon = 0.5$). The top left image is the original ImageNet sample, then from left to right $S = [5, 15, 25, 50, 100]$

4.2.3 HEIS Transferability

The most effective use of HEIS was in the transferability of its adversarial examples to the other models, which is where the Boundary Attack [12] severely underperformed. After running HEIS and the Boundary attack against ResNet50, we took the resulting adversarial examples and passed them through our three other ImageNet models. Although these models are all trained on the same dataset, they are architecturally quite different from each other. Figures 4.34, 4.35, and 4.36 show the results of these tests against MobileNetV2, VGG16, and DenseNet201 respectively. As we observed in Section 4.2.2, HEIS produces adversarial examples that create a high degree of uncertainty among the model’s output classes. We discovered that this led to the creation of adversarial examples that were very transferable to our other ImageNet models and were even able to degrade their performance in both top-1 and top-5 accuracy. This is quite an interesting result. Since the ResNet50 adversarial examples were effective at degrading both the top-1 and top-5 accuracy of all three other ImageNet DNN, it suggests that HEIS creates adversarial examples that are able to obscure

features in the input data that multiple DNNs relied on for classification.

Table 4.6 highlights some of the most notable drops in performance among our other ImageNet models when classifying our ResNet50 adversarial examples. MobileNetV2 dropped 66 percentage points in top-1 accuracy for $\epsilon \geq 0.3$, VGG16 dropped 53 percentage points and DenseNet201 dropped 56 percentage points for $\epsilon \geq 0.5$. As you can see in Table 4.6, similarly significant drops were also observed in top-5 accuracy.

Table 4.6. ImageNet: HEIS Transfer Accuracy

Model	Top 1 Drop	Top 5 Drop	ϵ
MobileNetV2	83% \rightarrow 17%	98% \rightarrow 20%	≥ 0.3
VGG16	73% \rightarrow 20%	89% \rightarrow 25%	≥ 0.5
DenseNet201	86% \rightarrow 30%	96% \rightarrow 45%	≥ 0.5

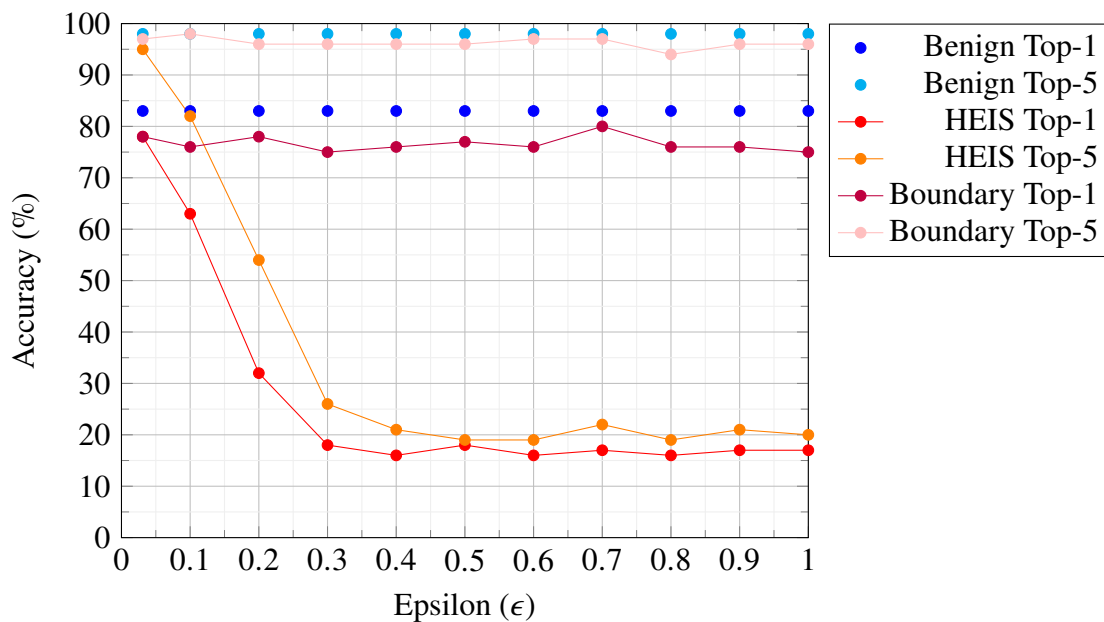


Figure 4.34. HEIS Transfer Accuracy (MobileNetV2)

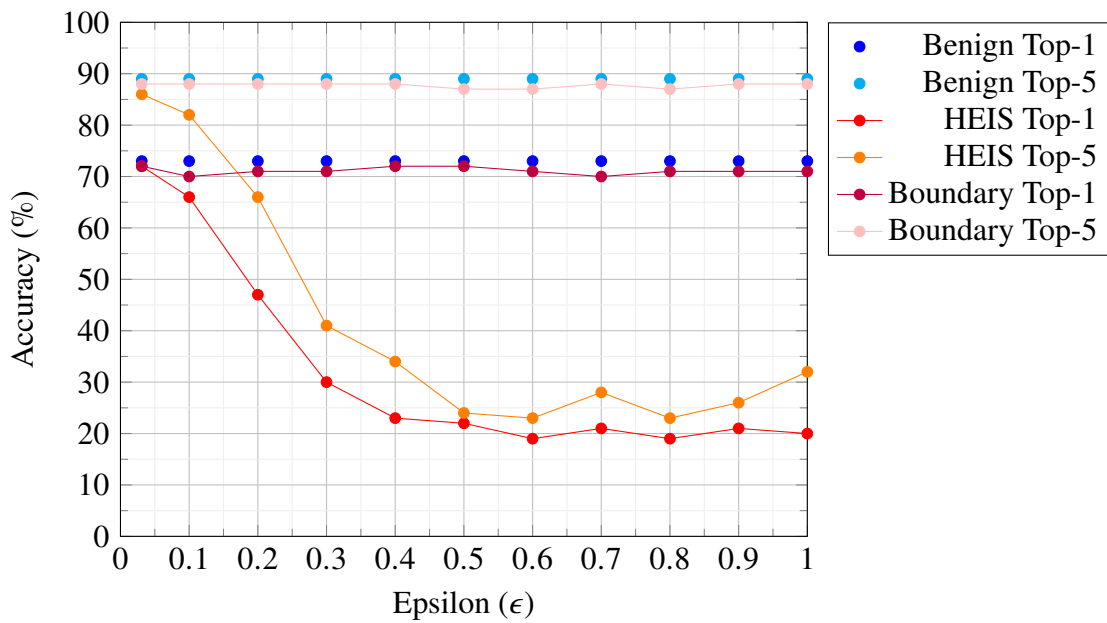


Figure 4.35. HEIS Transfer Accuracy (VGG16)

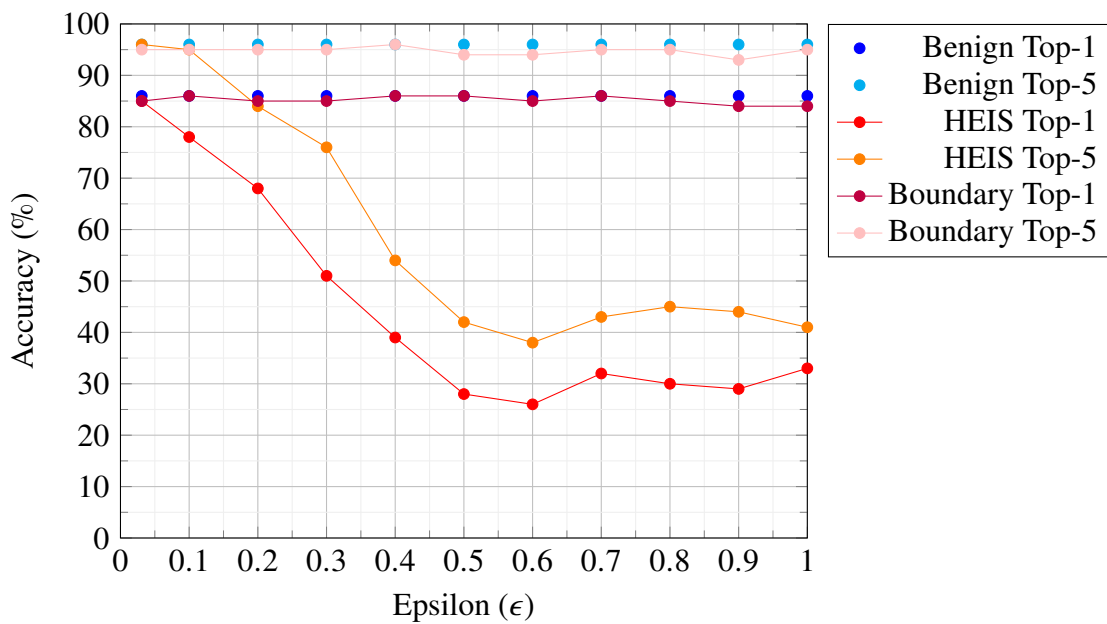


Figure 4.36. HEIS Transfer Accuracy (DenseNet201)

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 5: Discussion and Future Work

5.1 Discussion

5.1.1 Real World Applications

In real world applications of adversarial attacks, it is most likely the case that attackers will not have access to the ML system or model that they are trying to defeat. It is in these black box environments where attacks can also be the most difficult, since you cannot know for sure what data the system may have been trained on. In HEIS, we introduce a decision based attack that requires no knowledge of the model or the training data, and without the need for a surrogate model. The practicality associated with strict black box attacks like HEIS cannot be understated, any model can be attacked and the attack does not rely on gradients or approximations or prior knowledge.

While the Boundary Attack from [12] is a phenomenal and practical black box attack, we found that its adversarial examples do not generalize well and are only effective against the target model, and then only against top-1 accuracy. HEIS creates high entropy, multi-class uncertainty among the target model's class predictions, and the adversarial examples transfer extremely well to other models. It is reasonable to assume that adversarial examples created from complex, natural datasets like ImageNet could also be effective against real-world models without knowing the actual training data.

5.1.2 HEIS Perturbations

As observed in Sections 4.1.3 and 4.2.2, one of the main drawbacks of HEIS is that it adds a decent amount of visible perturbation to the image. The images begin to become unrecognizable after ϵ values of around 0.5 for CIFAR-10 and are noticeably noisy in the ImageNet adversarial examples for ϵ values as small as 0.4¹. This is because, as explained in Section 3.1.1, our clipping hyperparameter, ϵ , controls the percentage HEIS allows the

¹More CIFAR-10 and ImageNet adversarial examples can be seen in Appendix A.

pixel values to stray from their original value. For example, if $\epsilon = 0.5$, the algorithm is allowed to alter the data by up to 50%. If the pixel range is 0-255, an individual pixel could now be anything between ± 127 of its original value (clipped between 0 and 255, of course). However, it is not uncommon for adversarial attacks to result in visibly obvious perturbations [13], [15]. Although these perturbations are obvious to the human observer, many ML applications are designed to augment or replace humans with the goal of increased automation and efficiency – so in many cases, it may not matter whether or not the adversarial examples have visual perturbations, as they may never be seen by an actual human. This is why black box attacks, especially decision based attacks, remain very relevant to DNN research.

In Section 4.2.1, we discovered that for $\epsilon \geq 0.4$, the HEIS adversarial examples stopped negatively effecting performance and instead the model's accuracy plateaued. We believe this to be a side effect from the methodology of the attack. Since the entropy calculation is the only metric HEIS uses to choose adversarial examples (and not accuracy), we observed that the algorithm would often iterate through multiple steps where it could not find samples with higher entropy than the one from the previous step. This typically occurred around 30-50 steps. At this point the class distributions usually contained small "peaks" around two or three classes and the rest of the categories were lower (although still much higher than for a benign image). In our Vanilla CIFAR-10 model, these peaks were more often than not centered around the "Cat" and "Frog" classes (observable in Figures 4.9 and 4.11). Perhaps the "Cat" and "Frog" decision regions in the network were the largest? Or the decision boundaries for those two classes coincided with most other boundaries? In any case, this outcome became the most common which means that is where the entropy calculation maximized. With only 10 output classes, even a network that constantly outputs one class for every input is still correct 10% of the time, as we see in Section 4.1. In our ImageNet testing these "peaks" were not as noticeable since there were 1000 different output classes.

5.1.3 Unanswered Questions

During experimentation, we witnessed a few surprising observations. The most surprising was the fact that increasing the value of P , the number of starting points, did not improve the performance of the adversarial examples. Would increasing P beyond 100 have given better results? What about a combination of increasing P and μ_p to allow the algorithm to

search a wider space around the current image?

Could HEIS be integrated with another type of attack, like PGD, to create a more robust, multifaceted attack? Or create a hybrid attack that results in adversarial examples that are less perturbed yet still highly transferable? There are several possible avenues for this type of research; one possibility could be to create HEIS adversarial examples and then use them as the starting point for a gradient-based attack, or the reverse – taking adversarial examples from another attack and using them as the starting point for HEIS with small step counts.

5.2 Future Work

Our first suggestion for further HEIS experimentation is to run the attack against ImageNet DNNs that were trained using defensive techniques discussed in Chapter 2. In Section 4.1.4 we found that some of these techniques reduced the effectiveness of our CIFAR-10 adversarial examples, so it would be interesting to see how robust a more complex DNN like ResNet50 would be against HEIS if it were fortified by such defenses. Another avenue for future research is to compare HEIS against other boundary and decision-based attacks – not just the Boundary Attack from [12]. Comparing against multiple attacks under controlled conditions using similar or identical parameters is the easiest way to convey attack effectiveness.

For high ϵ values, HEIS produces adversarial images that are easily recognizable by the human observer as having visual perturbations. For low ϵ values, HEIS creates adversarial examples with visually less perturbation, except the examples are not as effective compared to examples with higher ϵ values. There are several avenues of testing and experimentation that could possibly improve or assist HEIS in creating stronger adversarial examples with less perturbation.

- When sampling, only add partial noise to smaller areas within the images, instead of across the entire image. Since HEIS takes a large number of samples, restricting the amount of noise and where within the image it gets added might reduce the overall amount of noise produced in the final adversarial examples. This should work since the algorithm could still theoretically find the desired high-entropy samples given enough iterations.

- Clip pixel values in only a positive or negative direction instead of both (e.g. only allow pixel values to get larger or smaller but not both).
- Increasing P and/or μ_p to increase the sample space from which HEIS chooses its starting points.
- Decrease the amount of steps. Although this tactic was explored briefly in this thesis, perhaps combining it with any of the previous suggestions might lead to better overall results.

Perhaps there is potential for an adaptive version of HEIS that varies the amount of noise added when sampling for starting points or entropy samples. One possible implementation could be to increase the amount of noise allowed in the sampling process by adaptively updating μ_p and/or μ_e if the attack progresses through multiple iterations yet fails to find samples with higher entropy.

CHAPTER 6: Conclusion

Despite their high performance in computer vision applications, ML systems like DNNs are still susceptible to noisy or intentionally perturbed data. This threat is even more concerning in safety critical applications or when incorporated into secure systems. Types of systems like those utilized in autonomous cars, facial recognition systems, and unmanned aerial vehicles can be vulnerable to adversarial attacks, which can have devastating real-world consequences if not handled correctly by the controlling software. In this thesis, we present a technique that achieves excellent black box results and creates adversarial examples that exhibit high transferability to other models.

Without having any knowledge of the inner workings of any model, HEIS was able to drop the accuracy of our Vanilla CIFAR-10 model, a DNN with 38 layers and 2.66 million trainable parameters, from 91% to 11% for $\epsilon = 0.2$; and drop the accuracy of ResNet50, a DNN with 50 layers and 25.5 million trainable parameters, to below 50% in both top-1 and top-5 metrics using ϵ values as low as 0.2 and 0.3. What's more, these adversarial examples worked splendidly against other DNNs whereas another popular decision based attack, the Boundary Attack [12], did not.

As the use of ML systems increases, people need to understand the limitations of tools like DNNs and realize how fragile they can be. Previous research has shown that adversarial attacks are easily realized in the physical world [5], [30] and cyberspace [31]. As Jatho and Kroll point out in [32], "We must be extremely judicious about when, where and how we employ these technologies", for AI applications are often "wildly optimistic" and can "inflate our expectations of what this technology can do." This is especially true in critical applications where unintentional vulnerabilities can cause irreparable harm, especially when dealing with human safety, security applications, or military and defense operations. It is our hope that adversarial research like ours will continue to expose vulnerabilities in ML systems in order to inspire increasingly robust defenses that enable them to continue to operate in the real world undeterred.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX: A

This appendix contains extra visuals for the reader that are not necessarily important to add into the body of the thesis. For example, Section A.1 shows the same adversarial examples created by HEIS across multiple values of ϵ to show how much visual perturbation is produced.

3D Plot of Multiple Classes in a Dataset

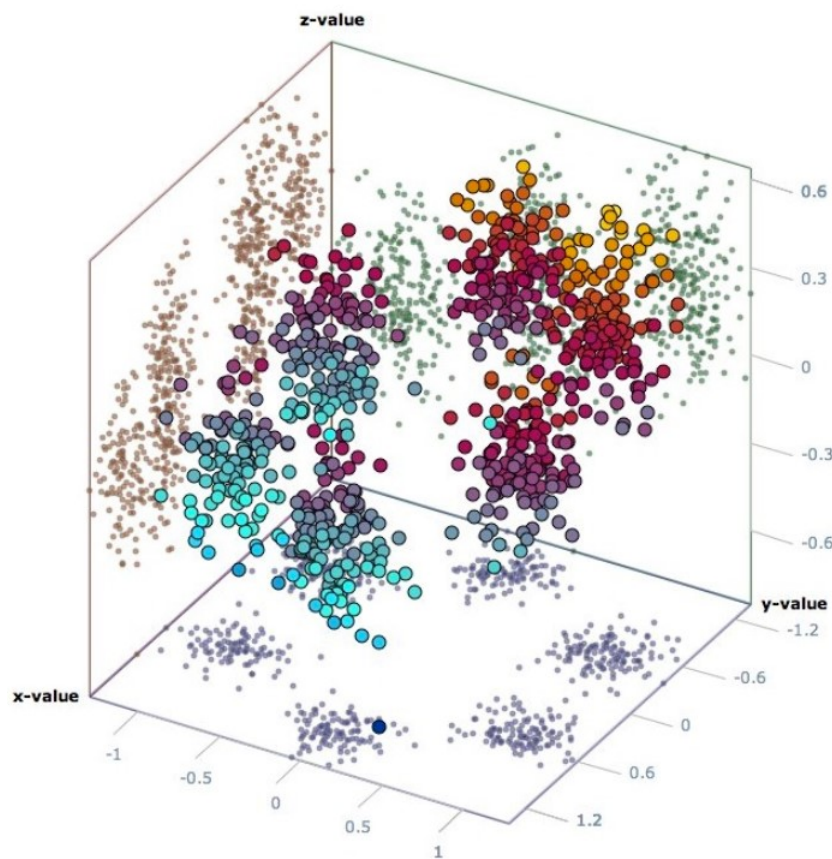


Figure A.1. 3-dimensional plot showing multiple classes. Drawing a boundary between two or more classes proves challenging with increased dimensionality as boundaries here become three dimensional planes instead of two dimensional lines. Source: <https://www.doka.ch/Excel3Dscatterplot.htm>

A.1 More HEIS Adversarial Examples

A.1.1 More CIFAR-10 Adversarial Examples

Epsilon: 0.031



Figure A.2. CIFAR-10 HEIS Adversarial Examples ($\epsilon = 0.031$)

Epsilon: 0.1



Figure A.3. CIFAR-10 HEIS Adversarial Examples ($\epsilon = 0.1$)

Epsilon: 0.2



Figure A.4. CIFAR-10 HEIS Adversarial Examples ($\epsilon = 0.2$)

Epsilon: 0.3

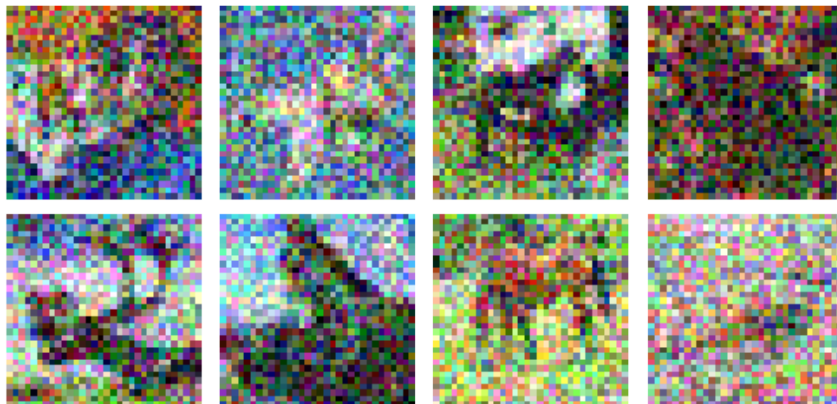


Figure A.5. CIFAR-10 HEIS Adversarial Examples ($\epsilon = 0.3$)

Epsilon: 0.4

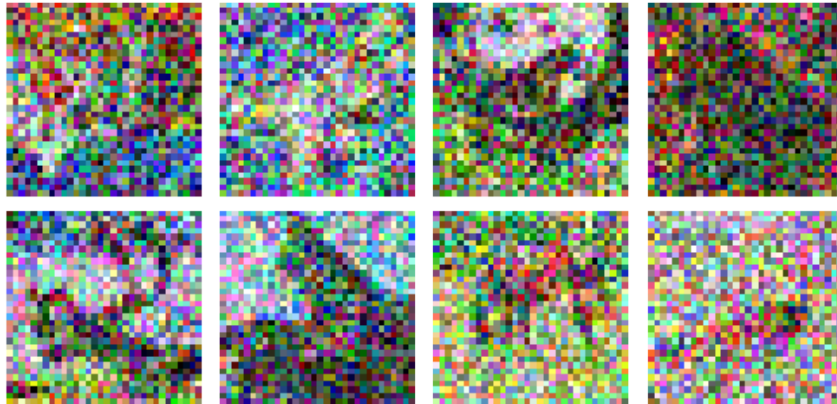


Figure A.6. CIFAR-10 HEIS Adversarial Examples ($\epsilon = 0.4$)

Epsilon: 0.5

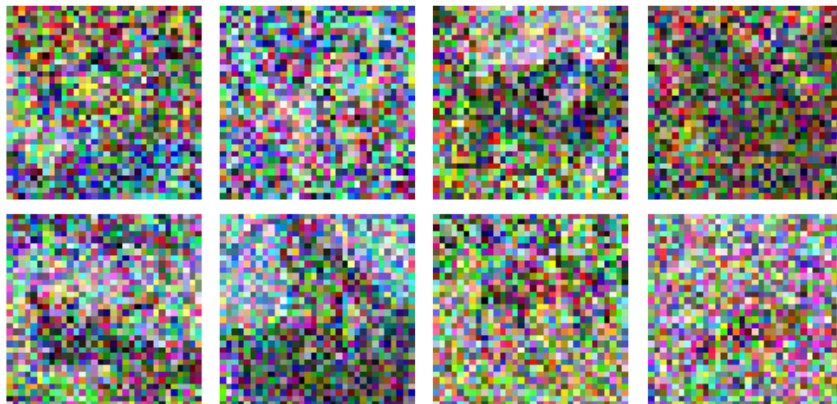


Figure A.7. CIFAR-10 HEIS Adversarial Examples ($\epsilon = 0.5$)

Epsilon: 0.6

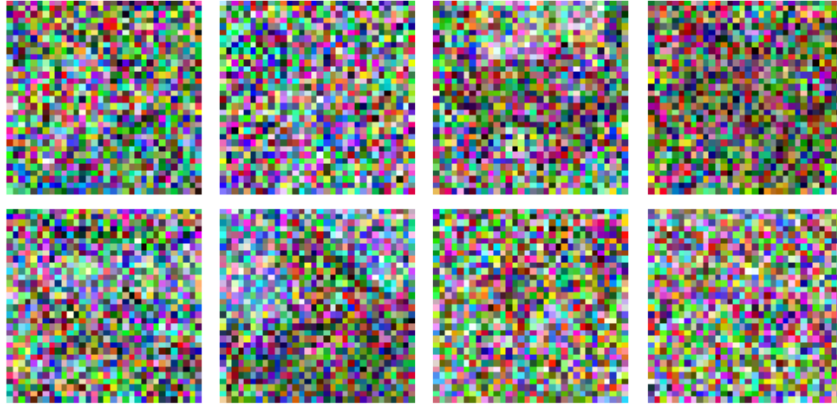


Figure A.8. CIFAR-10 HEIS Adversarial Examples ($\epsilon = 0.6$)

Epsilon: 0.7

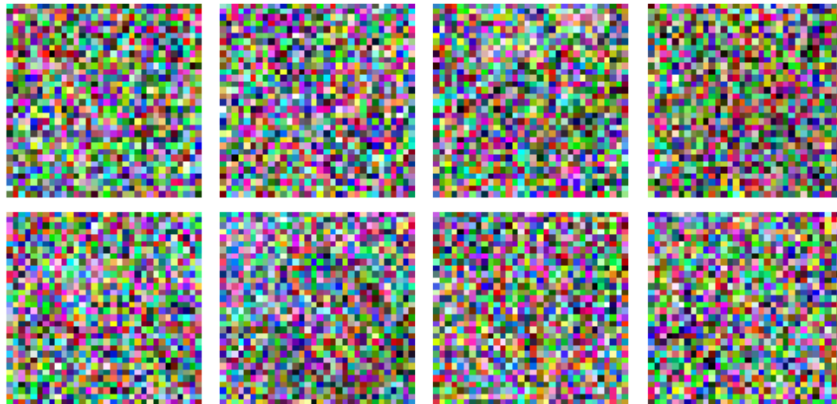


Figure A.9. CIFAR-10 HEIS Adversarial Examples ($\epsilon = 0.7$)

Epsilon: 0.8

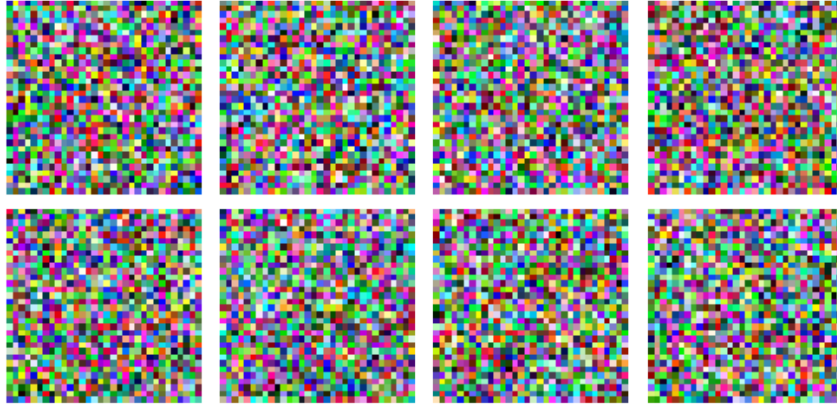


Figure A.10. CIFAR-10 HEIS Adversarial Examples ($\epsilon = 0.8$)

Epsilon: 0.9

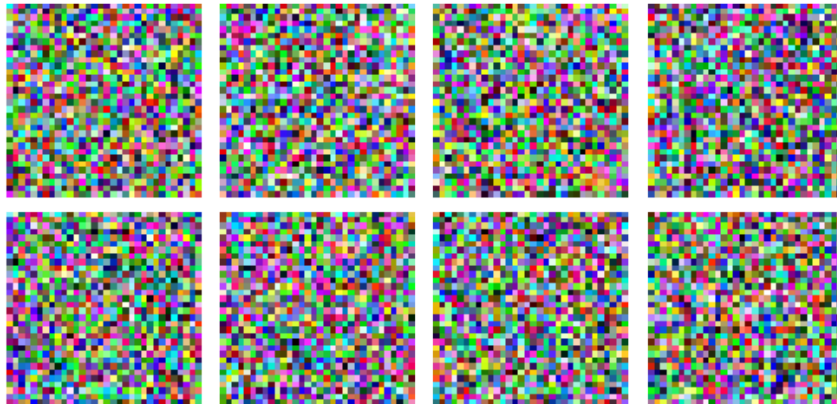


Figure A.11. CIFAR-10 HEIS Adversarial Examples ($\epsilon = 0.9$)

Epsilon: 1.0

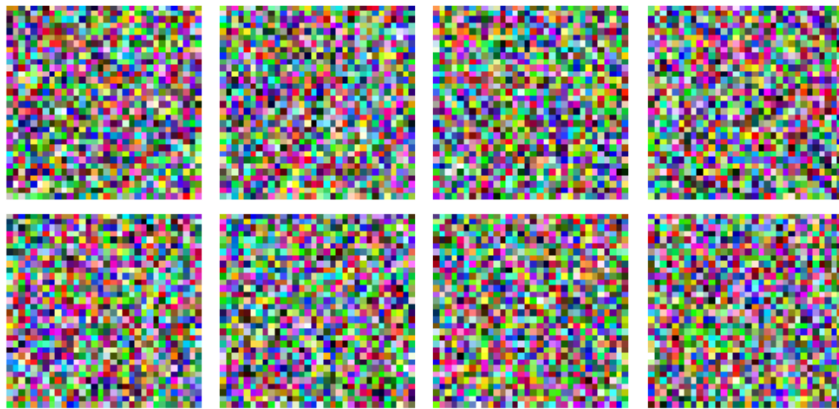


Figure A.12. CIFAR-10 HEIS Adversarial Examples ($\epsilon = 1.0$)

A.1.2 More ImageNet Adversarial Examples

Epsilon: 0.031



Figure A.13. ImageNet HEIS Adversarial Examples ($\epsilon = 0.031$)

Epsilon: 0.1

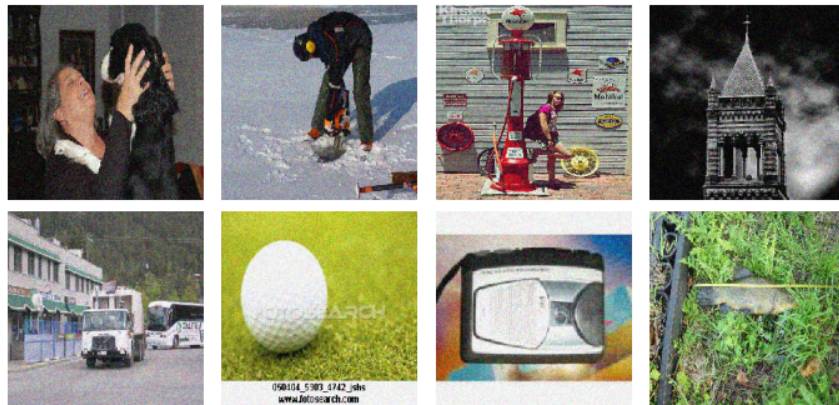


Figure A.14. ImageNet HEIS Adversarial Examples ($\epsilon = 0.1$)

Epsilon: 0.2

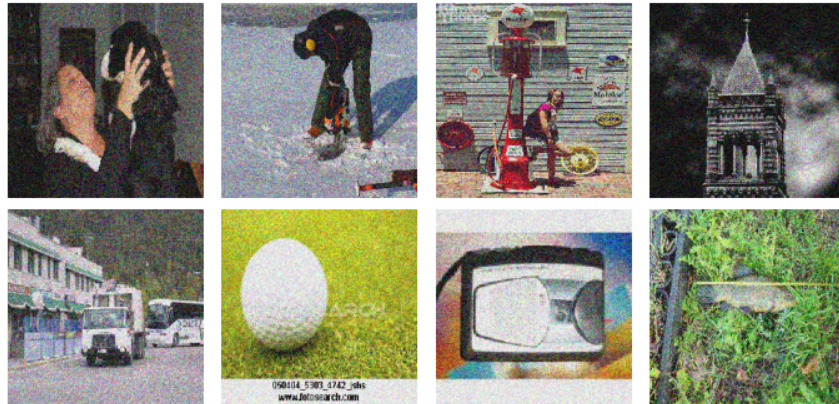


Figure A.15. ImageNet HEIS Adversarial Examples ($\epsilon = 0.2$)

Epsilon: 0.3



Figure A.16. ImageNet HEIS Adversarial Examples ($\epsilon = 0.3$)

Epsilon: 0.4

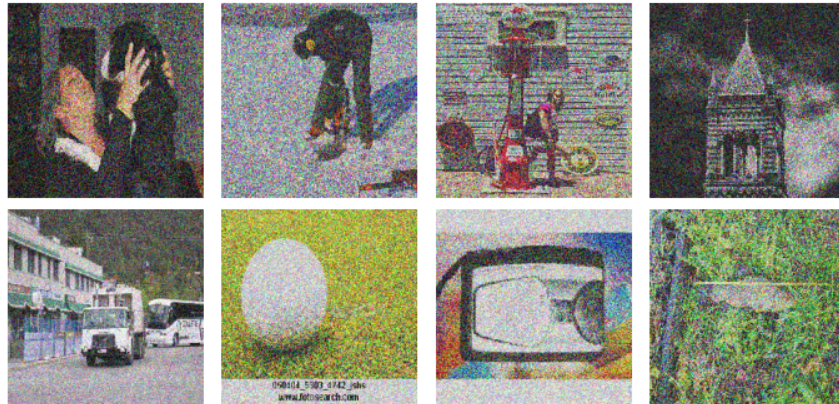


Figure A.17. ImageNet HEIS Adversarial Examples ($\epsilon = 0.4$)

Epsilon: 0.5

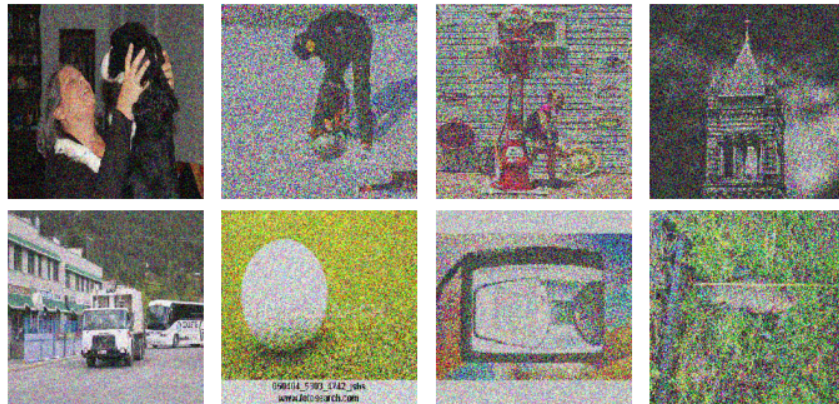


Figure A.18. ImageNet HEIS Adversarial Examples ($\epsilon = 0.5$)

Epsilon: 0.6

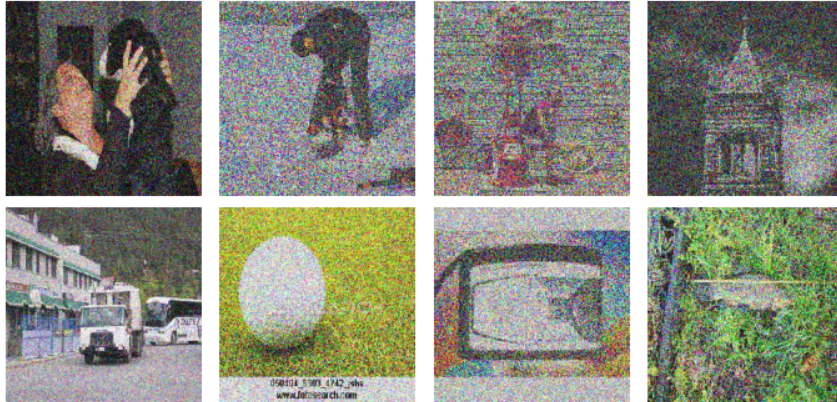


Figure A.19. ImageNet HEIS Adversarial Examples ($\epsilon = 0.6$)

Epsilon: 0.7

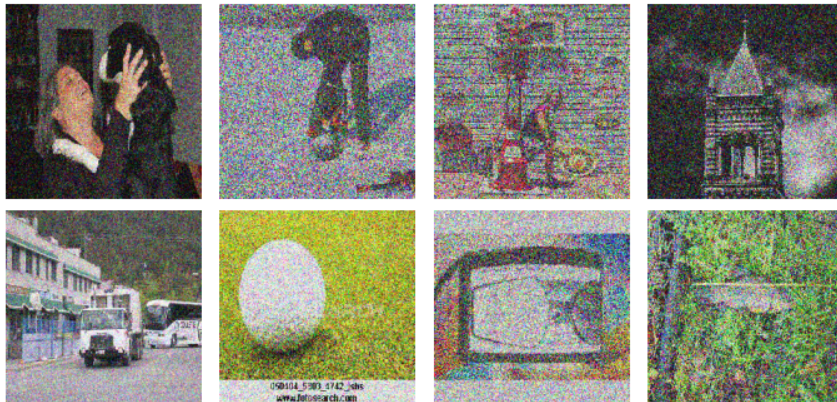


Figure A.20. ImageNet HEIS Adversarial Examples ($\epsilon = 0.7$)

Epsilon: 0.8

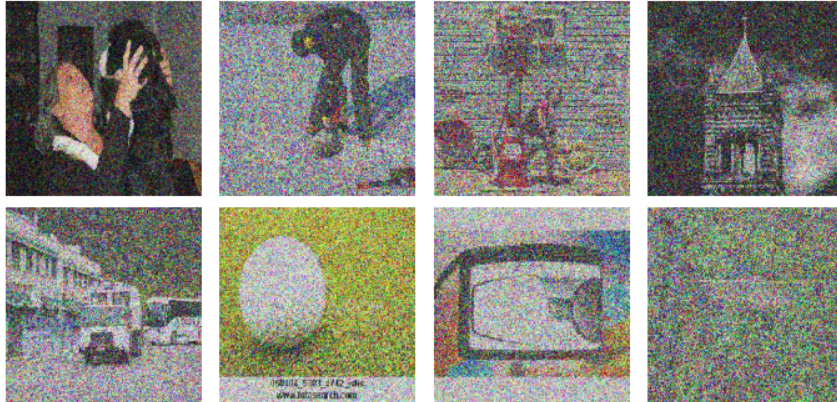


Figure A.21. ImageNet HEIS Adversarial Examples ($\epsilon = 0.8$)

Epsilon: 0.9



Figure A.22. ImageNet HEIS Adversarial Examples ($\epsilon = 0.9$)

Epsilon: 1.0

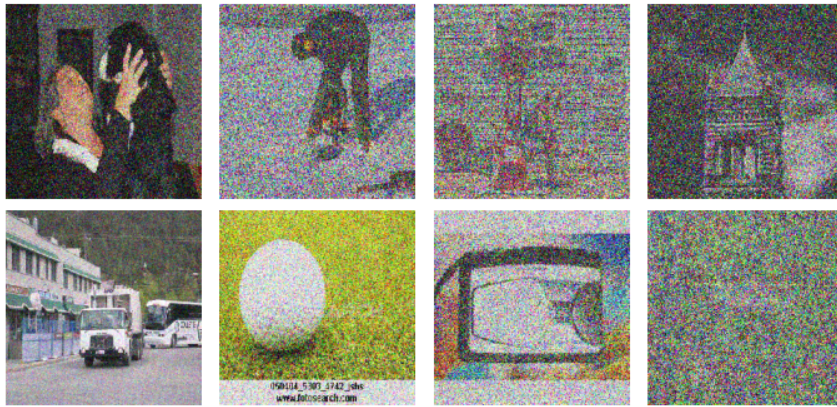


Figure A.23. ImageNet HEIS Adversarial Examples ($\epsilon = 1.0$)

THIS PAGE INTENTIONALLY LEFT BLANK

List of References

- [1] A. Geron, *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*, 2nd ed. Sebastopol, CA, USA: O'Reilly Media, 2019.
- [2] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2015.
- [3] H. Karimi, T. Derr, and J. Tang, "Characterizing the decision boundary of deep neural networks," *CoRR*, 2019 [Online]. doi: <https://doi.org/10.48550/arXiv.1912.11460>.
- [4] A. Chakraborty, M. Alam, V. Dey, A. Chattopadhyay, and D. Mukhopadhyay, "Adversarial attacks and defences: A survey," *CoRR*, 2018 [Online]. url: <http://arxiv.org/abs/1810.00069>.
- [5] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," 2017 [Online]. doi: <https://doi.org/10.48550/arXiv.1607.02533>.
- [6] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *CoRR*, 2015 [Online]. doi: <https://doi.org/10.48550/arXiv.1412.6572>.
- [7] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *CoRR*, 2014 [Online]. doi: <https://doi.org/10.48550/arXiv.1312.6199>.
- [8] M. Sharif, S. Bhagavatula, L. Bauer, and M. K. Reiter, "A general framework for adversarial examples with objectives," *ACM Transactions on Privacy and Security*, vol. 22, no. 3, June 2019 [Online]. url: <https://mahmoods01.github.io/files/tops19-adv-ml.pdf>.
- [9] A. Shafahi, M. Najibi, A. Ghiasi, Z. Xu, J. Dickerson, C. Studer, L. S. Davis, G. Taylor, and T. Goldstein, "Adversarial training for free!" 2019 [Online]. doi: <https://doi.org/10.48550/arXiv.1904.12843>.
- [10] Y. Lin, H. Zhao, Y. Tu, S. Mao, and Z. Dou, "Threats of adversarial attacks in dnn-based modulation recognition," in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, 2020, pp. 2469–2478.
- [11] I. J. Goodfellow, "Defense against the dark arts: An overview of adversarial example security research and future research directions," *CoRR*, 2018 [Online]. doi: <https://doi.org/10.48550/arXiv.1806.04169>.

- [12] W. Brendel, J. Rauber, and M. Bethge, “Decision-based adversarial attacks: Reliable attacks against black-box machine learning models,” 2017 [Online]. doi: <https://doi.org/10.48550/arXiv.1712.04248>.
- [13] Y. Liu, X. Chen, C. Liu, and D. Song, “Delving into transferable adversarial examples and black-box attacks,” 2016 [Online]. doi: <https://doi.org/10.48550/arXiv.1611.02770>.
- [14] N. Akhtar and A. Mian, “Threat of adversarial attacks on deep learning in computer vision: A survey,” *IEEE Access*, vol. 6, pp. 14 410–14 430, 2018.
- [15] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, “Towards deep learning models resistant to adversarial attacks,” 2019 [Online]. doi: <https://doi.org/10.48550/arXiv.1706.06083>.
- [16] H. Karimi and J. Tang, “Decision boundary of deep neural networks: Challenges and opportunities,” in *Proceedings of the 13th International Conference on web search and data mining (WSDM ’20)*. ACM, 2020, pp. 919–920.
- [17] W. He, B. Li, and D. Song, “Decision boundary analysis of adversarial examples,” in *International Conference on Learning Representations*, 2018 [Online]. url: <https://openreview.net/forum?id=BkpiPMbA->.
- [18] A. Barton, “Defending neural networks against adversarial examples,” Ph.D. dissertation, The University of Texas at Arlington, Arlington, TX, USA, December 2018.
- [19] F. Tramèr, N. Carlini, W. Brendel, and A. Madry, “On adaptive attacks to adversarial example defenses,” *CoRR*, 2020 [Online]. doi: <https://doi.org/10.48550/arXiv.2002.08347>.
- [20] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, “Distillation as a defense to adversarial perturbations against deep neural networks,” in *2016 IEEE Symposium on Security and Privacy (SP)*, 2016, pp. 582–597.
- [21] N. Carlini and D. Wagner, “Towards evaluating the robustness of neural networks,” in *2017 IEEE Symposium on Security and Privacy (SP)*, 2017, pp. 39–57.
- [22] H. Zhang, Y. Yu, J. Jiao, E. P. Xing, L. E. Ghaoui, and M. I. Jordan, “Theoretically principled trade-off between robustness and accuracy,” *CoRR*, 2019 [Online]. doi: <https://doi.org/10.48550/arXiv.1901.08573>.
- [23] A. S. Ross and F. Doshi-Velez, “Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients,” *CoRR*, vol. abs/1711.09404, 2017 [Online]. doi: <https://doi.org/10.48550/arXiv.1711.09404>.

- [24] X. Cao and N. Z. Gong, “Mitigating evasion attacks to deep neural networks via region-based classification,” 2017 [Online]. doi: <https://doi.org/10.48550/arXiv.1709.05583>.
- [25] D. Tsipras, S. Santurkar, L. Engstrom, A. Turner, and A. Madry, “Robustness may be at odds with accuracy,” 2018 [Online]. doi: <https://doi.org/10.48550/arXiv.1805.12152>.
- [26] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” *CoRR*, 2015 [Online]. doi: <https://doi.org/10.48550/arXiv.1512.03385>.
- [27] M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L. Chen, “Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation,” in *CVPR*, 2018.
- [28] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in *3rd International Conference on Learning Representations*, Y. Bengio and Y. LeCun, Eds., San Diego, CA, USA, May 2015.
- [29] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2261–2269.
- [30] A. Athalye, L. Engstrom, A. Ilyas, and K. Kwok, “Synthesizing robust adversarial examples,” *CoRR*, 2017 [Online]. doi: <https://doi.org/10.48550/arXiv.1707.07397>.
- [31] N. Papernot, P. D. McDaniel, I. J. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, “Practical black-box attacks against deep learning systems using adversarial examples,” *CoRR*, 2016 [Online]. doi: <https://doi.org/10.48550/arXiv.1602.02697>.
- [32] E. Jatho and J. A. Kroll, “Artificial intelligence: Too fragile to fight?” U.S. Naval Institute, Booz Allen Hamilton, February 2022 [Online]. url: <http://hdl.handle.net/10945/68820>.

THIS PAGE INTENTIONALLY LEFT BLANK

Initial Distribution List

1. Defense Technical Information Center
Ft. Belvoir, Virginia
2. Dudley Knox Library
Naval Postgraduate School
Monterey, California