ATTACKING COMPUTER VISION MODELS USING OCCLUSION ANALYSIS
TO CREATE PHYSICALLY ROBUST ADVERSARIAL IMAGES

A Thesis
presented to
the Faculty of California Polytechnic State University,
San Luis Obispo

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in Computer Science

by
Jacobsen Loh
June 2020
COMMITTEE MEMBERSHIP

TITLE: Attacking Computer Vision Models using Occlusion Analysis to Create Physically Robust Adversarial Images

AUTHOR: Jacobsen Loh

DATE SUBMITTED: June 2020

COMMITTEE CHAIR: Bruce DeBruhl, Ph.D.
Assistant Professor of Computer Science and Software Engineering

COMMITTEE MEMBER: Maria Pantoja, Ph.D.
Assistant Professor of Computer Science and Software Engineering

COMMITTEE MEMBER: Charles Birdsong, Ph.D.
Professor of Mechanical Engineering
ABSTRACT

Attacking Computer Vision Models using Occlusion Analysis to Create Physically Robust Adversarial Images

Jacobsen Loh

Self-driving cars rely on their sense of sight to function effectively in chaotic and uncontrolled environments. Thanks to recent developments in computer vision, specifically convolutional neural networks, autonomous vehicles have developed the ability to see at or above human-level capabilities, which in turn has allowed for rapid advances in self-driving cars. Unfortunately, much like humans being confused by simple optical illusions, convolutional neural networks are susceptible to simple adversarial inputs. As there is no overlap between the optical illusions that fool humans and the adversarial examples that threaten convolutional neural networks, little is understood as to why these adversarial examples dupe such advanced models and what effective mitigation techniques might exist to resolve these issues.

This thesis focuses on these adversarial images. By extending existing work, this thesis is able to offer a unique perspective on adversarial examples. Furthermore, these extensions are used to develop a novel attack that can generate physically robust adversarial examples. These physically robust instances provide a unique challenge as they transcend both individual models and the digital domain, thereby posing a significant threat to the efficacy of convolutional neural networks and their dependent applications.
ACKNOWLEDGMENTS

Academia is a challenging place, and I have no doubt that I owe much of my success and achievements to the help and support of my professors, friends, and family.

I would like to thank Ford Motor Company for sponsoring my research this past year, and providing time for research engineers to meet with me and other students on a weekly basis for discussions and questions.

Dr. Maria Pantoja and Dr. John Seng introduced me to deep learning and provided time and support to answer my questions about the field. Deep learning is a challenging field, and they helped me dip my toes in the water and learn to swim.

Two professors of particular recognition are Dr. Charles Birdsong and Dr. Bruce DeBruhl, who advised both my senior project and my master’s thesis. Their support and guidance has allowed me to transition from mechanical engineering to computer science. By finding my passion, I know I will be far more successful and happy in my career. I hope that they understand the impact they have had on me and how dramatically they improved my time at Cal Poly.

My friends, specifically Preston Quick and Austin Djang, have been invested in my success as a computer scientist from the beginning. From late night talks regarding my mechanical engineering doubts to senior project selection, they have both been immensely supportive of me. They helped me realize it is okay to change majors, no matter how late it might be.

Finally, I want to thank my family, who have always been so supportive of me. More specifically though, I want to thank my dad. His investment in my academic success goes back further than I can remember. With regards to my master’s degree and this thesis, he helped me learn how to conduct research and with understanding unfamiliar
math. We discussed the limitations of both existing work and my research so I could produce novel and interesting contributions. And at the end of the year, he helped me edit this thesis. I cannot state enough how much he has helped me grow into the academic and person I am today. So to him above all, thank you.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>LIST OF TABLES</th>
<th>x</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST OF FIGURES</td>
<td>xi</td>
</tr>
<tr>
<td>CHAPTER</td>
<td></td>
</tr>
<tr>
<td>1 Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Contributions</td>
<td>2</td>
</tr>
<tr>
<td>1.2 Chapters</td>
<td>3</td>
</tr>
<tr>
<td>2 Background</td>
<td>5</td>
</tr>
<tr>
<td>2.1 Neural Networks</td>
<td>5</td>
</tr>
<tr>
<td>2.1.1 Activation Functions</td>
<td>6</td>
</tr>
<tr>
<td>2.1.2 Loss Functions</td>
<td>8</td>
</tr>
<tr>
<td>2.1.3 Backpropagation</td>
<td>9</td>
</tr>
<tr>
<td>2.1.4 Regularization</td>
<td>10</td>
</tr>
<tr>
<td>2.2 Deep Learning</td>
<td>12</td>
</tr>
<tr>
<td>2.3 Convolutional Neural Networks</td>
<td>12</td>
</tr>
<tr>
<td>2.4 Feature Extraction and Feature Classification</td>
<td>14</td>
</tr>
<tr>
<td>2.5 Transfer Learning</td>
<td>15</td>
</tr>
<tr>
<td>2.6 Residual Networks</td>
<td>16</td>
</tr>
<tr>
<td>3 Related Works</td>
<td>19</td>
</tr>
<tr>
<td>3.1 Early Adversarial Examples</td>
<td>19</td>
</tr>
<tr>
<td>3.2 Physically Robust Adversarial Examples</td>
<td>20</td>
</tr>
<tr>
<td>3.3 Defenses to Adversarial Examples</td>
<td>20</td>
</tr>
<tr>
<td>4 Tools</td>
<td>22</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>7.3.2 Attacking a Model using Sliding Occlusions</td>
<td>54</td>
</tr>
<tr>
<td>7.4 Conclusions</td>
<td>60</td>
</tr>
<tr>
<td>8 Conclusions</td>
<td>61</td>
</tr>
<tr>
<td>8.1 Contributions</td>
<td>61</td>
</tr>
<tr>
<td>8.2 Discussions &amp; Future Work</td>
<td>62</td>
</tr>
<tr>
<td>BIBLIOGRAPHY</td>
<td>64</td>
</tr>
</tbody>
</table>
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Summary of tools used in this thesis</td>
<td>24</td>
</tr>
<tr>
<td>5.1</td>
<td>Robustness of trolley adversarial images generated on ResNet-50 across multiple models</td>
<td>31</td>
</tr>
<tr>
<td>5.2</td>
<td>Robustness of trolley adversarial images generated on ResNet-50 across multiple models after being transformed through the physical domain</td>
<td>31</td>
</tr>
<tr>
<td>5.3</td>
<td>Robustness of washer adversarial images generated on ResNet-50 across multiple models</td>
<td>33</td>
</tr>
<tr>
<td>5.4</td>
<td>Robustness of washer adversarial images generated on ResNet-50 across multiple models after being transformed through the physical domain</td>
<td>33</td>
</tr>
<tr>
<td>7.1</td>
<td>Sliding occlusion analysis of top two classes for the stop signs in Figure 7.5 (rows in this table correspond to rows in Figure 7.5)</td>
<td>56</td>
</tr>
<tr>
<td>7.2</td>
<td>Performance of adversarial stop signs against a ResNet-50 model</td>
<td>58</td>
</tr>
<tr>
<td>7.3</td>
<td>Performance of adversarial stop signs against a ResNet-50 model with some adversarial training</td>
<td>59</td>
</tr>
<tr>
<td>7.4</td>
<td>Performance of adversarial stop signs against a MobileNet model</td>
<td>59</td>
</tr>
</tbody>
</table>
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Simple fully-connected 3-layer neural network</td>
<td>6</td>
</tr>
<tr>
<td>2.2</td>
<td>A standard neural network (left) and a neural network after applying dropout (right)</td>
<td>11</td>
</tr>
<tr>
<td>2.3</td>
<td>Basic CNN architecture showing a image input layer, filter, and a convolutional layer</td>
<td>13</td>
</tr>
<tr>
<td>2.4</td>
<td>AlexNet architecture divided for training across two GPUs</td>
<td>13</td>
</tr>
<tr>
<td>2.5</td>
<td>Fundamental building block of residual networks</td>
<td>17</td>
</tr>
<tr>
<td>2.6</td>
<td>A 34-layer residual network (the left portion feeds into the right portion) designed for use with 1000 classes</td>
<td>18</td>
</tr>
<tr>
<td>5.1</td>
<td>Goodfellow et al.’s seminal FGSM example</td>
<td>27</td>
</tr>
<tr>
<td>5.2</td>
<td>Top-1 and top-5 accuracy of Inception v3 on images produced by various attack methods over a varying $\epsilon$</td>
<td>27</td>
</tr>
<tr>
<td>5.3</td>
<td>Clean image of a trolley (left) with a softmax activation 0.998, the corresponding perturbation scaled for visualization (middle), and the adversarial example (right) with a softmax activation of 1.0</td>
<td>30</td>
</tr>
<tr>
<td>5.4</td>
<td>Class probabilities during the Iterative FGSM attack</td>
<td>31</td>
</tr>
<tr>
<td>5.5</td>
<td>Clean image of a trolley (left), a Non-Iterative FGSM adversarial image targeting the passenger car class (middle), and an Iterative FGSM adversarial image targeting the passenger car class (right)</td>
<td>32</td>
</tr>
<tr>
<td>5.6</td>
<td>Clean image of a washer (left), a Non-Iterative FGSM adversarial image targeting the safe class (middle), and an Iterative FGSM adversarial image targeting the safe class (right)</td>
<td>32</td>
</tr>
<tr>
<td>5.7</td>
<td>Cropped photographs of the images in Figure 5.5</td>
<td>34</td>
</tr>
<tr>
<td>5.8</td>
<td>Cropped photographs of the images in Figure 5.6</td>
<td>34</td>
</tr>
<tr>
<td>5.9</td>
<td>Correlation matrix of sign maps during all iterations of an Iterative FGSM</td>
<td>35</td>
</tr>
</tbody>
</table>

---

xi
5.10 Correlation matrix of sign maps during the final 10 stages of an Iterative FGSM 

5.11 Adversarial example (left) that is misclassified as a cucumber (0.9999) and the image after the liberal addition of noise (right) misclassified as a flowerpot (0.0714)

5.12 Clean example (left) and the image after the liberal addition of noise (right) misclassified as a park bench (0.2988)

6.1 RP$_2$ pipeline overview

6.2 Genuine graffiti (left) and an adversarial example (right) created by Eykholt et al.

6.3 Original image (top left), top 1% of gradients (top right), top 8% of gradients (bottom left), and top 32% of gradients (bottom right), which can be used to locate the weakest points of an image

6.4 Manually generated mask guided by the L$_1$ regularizations in Figure 6.3

6.5 Algorithmically generated adversarial example (left) constrained by the mask in Figure 6.4 and a manually created adversarial example (right) inspired by the algorithmically generated adversarial example

7.1 Original image (left), occlusion heat map (middle), and a composite image (right) to better visualize the image and heat map simultaneously (red indicates important regions while blue indicates unimportant regions)

7.2 Position of most damaging occlusion for the teapot in Figure 7.1

7.3 Original image (left), occlusion heat map (middle), and a composite image (right) to better visualize the image and heat map simultaneously (red indicates important regions while blue indicates unimportant regions)

7.4 A teapot without a handle (top row) and a teapot without a spout (bottom row)

7.5 Stop sign sliding occlusion analysis, which identifies the critical identifying features to be the regions underneath and between the letters “T” and “O”
7.6 Clean stop sign (left) and adversarial example (right) created using sliding occlusion analysis of multiple stop signs ............... 57

7.7 Top: Clean, Stop 1, Stop 2; Bottom: Stop 3, Stop 4 ............... 58

7.8 Eykholt et al. (left), Povolny et al. (middle), and image created in this thesis (right) ................................. 59
Chapter 1

INTRODUCTION

Autonomous vehicles have made rapid developments in recent years, moving from science fiction to reality. Though numerous technologies can be thanked for spurring these significant jumps towards a future with self-driving cars, perhaps no contribution has been greater than that of deep learning-based vision [1].

In a world built for and navigated by humans, the ability to see is of utmost importance. Until the last decade, the field of computer vision lagged well behind human performance and therefore struggled to be used in real-world applications; however, with the advent of convolutional neural networks and deep learning, computer vision has evolved from an error-prone technology to a tool with state-of-the-art capabilities exceeding those of a human [2].

From obstacle avoidance [1] to traffic signs [1], deep learning has given autonomous vehicles the eyes needed to navigate the world safely. Computer vision models are evolving to ensure they adapt to the needs of the real world.

In safety-critical applications such as autonomous vehicles, it is essential that safe operation can be guaranteed at all times. Errors by any component, including software, can produce disastrous consequences such as the loss of human life, which has already occurred during tests on public roads. Thus far these failures have only been the result of momentary lapses in performance, typically resulting from poor driving conditions. Therefore, the thought of an adversary deliberately attacking an autonomous vehicle’s vision systems should be of great concern, as a malicious actor could quickly wreak havoc [3]. While significant research has already been conducted
in various fields critical for autonomous vehicles, there is insufficient research in the realm of adversarial attacks against computer vision systems.

It turns out researchers have already identified vulnerabilities in computer vision systems (see Chapter 3). Of particular interest is the potential for an adversary to craft images that can confuse and mislead a model, similar to how humans are confused by optical illusions. These images, known as adversarial examples, vary from attacks modifying only a single pixel to attacks generating perceptible perturbations. Originally, these adversarial examples were primarily of interest to researchers for their ability to help illuminate the inner-workings of convolutional neural networks rather than as legitimate security concerns; however, in the past few years, researchers have discovered that these adversarial examples can be designed in such a way that they can exist the physical world (see Section 3.2). Perhaps even more concerning is the ability for these adversarial objects to be appear innocent while actually being catastrophic to computer vision models. This confirms the threat of adversarial examples, as it is now possible for an adversary to produce malicious objects that can be used to deliberately confuse and mislead models.

With these realizations, it is essential for researchers to discover the full potential of these attacks and develop effective defenses to ensure the security of safety-critical applications. Though significant work has already been done, there is still much to do before control can be fully relinquished to these systems.

1.1 Contributions

This thesis explores adversarial attacks against computer vision systems, in particular their application towards autonomous vehicles. Aside from summarizing the literature
pertaining to adversarial examples, this thesis reproduces existing work and makes novel contributions. Specifically, the contributions are:

- An in-depth study of the Fast Gradient Sign Method first proposed by Goodfellow et al. [4].
- A discussion of the limitations of the Robust Physical Perturbations (RP$_2$) attack pipeline developed by Eykholt et al. [5].
- Using occlusion analysis, introduced by Zeiler et al. [6], a novel attack is proposed.

1.2 Chapters

The organization of this thesis is as follows:

- **Chapter 2: Background**—A presentation of technical concepts fundamental to this thesis.
- **Chapter 3: Related Works**—A literature review of early adversarial examples, physically robust adversarial examples, and potential defense mechanisms to these attacks.
- **Chapter 4: Tools**—A brief overview of the hardware, software, and data sets used in the development of this thesis.
- **Chapter 5: Fast Gradient Sign Method**—An overview of the Fast Gradient Sign Method, as well as in-depth discussion and results that extend existing work.
• **Chapter 6: Physically Robust Perturbations**—An overview of the Robust Physical Perturbations (RP$_2$) attack pipeline and a discussion of its limitations.

• **Chapter 7: Sliding Occlusions**—An introduction to the occlusion analysis and a novel attack that uses this analysis.

• **Chapter 8: Conclusions**—A final discussion, summary of contributions, and recommendations for future work.
Chapter 2

BACKGROUND

Over the past decade, deep learning has transformed computer vision thereby allowing for rapid developments within the field; however, deep learning has a significant shortcoming in the form of adversarial examples [7]. These carefully crafted images and objects can produce significant misclassifications in deep learning models similar to a human’s confident misinterpretation of an optical illusion. This chapter will first discuss some of the fundamentals of deep learning that are used in this thesis. After introducing these fundamentals, advanced techniques that are used in this thesis will be introduced.

2.1 Neural Networks

Neural networks are the fundamental building blocks of deep learning models. These models consist of connected nodes (neurons) that pass signals through the model. By varying weights and biases within the model, different stimuli produce different outputs.

While neural networks seem to have roots in biological neural structures, they can be represented with simple linear algebra. The input to a node can be expressed by Equation 2.1 where $i$ indexes a node in the previous layer, $w$ is the weight for the connection between node $i$ and the current node, $x$ is the output signal from node $i$, and $b$ is the bias for the current node. This simple method for representing neural networks has many interesting properties, among them being fast computation due to their representations as matrices.
Figure 2.1: Simple fully-connected 3-layer neural network [8]

\[
\text{NodeInput} = \sum_i w_i x_i + b
\]

The final layer, typically known as the output layer, is then interpreted as the model’s prediction.

2.1.1 Activation Functions

Activation functions provide pseudo-thresholding capabilities to nodes. Strong inputs will cause the node to “fire” (an intuition drawn from biological neurons) while weak signals result in little to no output signal. A node produces its output by taking the sum of the inputs, passing it through the activation function, and then outputting this signal. These activation functions add non-linearity to a model, which in turn makes it possible for a model to act as universal approximators [9].

The sigmoid activation function, which was a popular activation function during the early development of neural networks, is shown in Equation 2.2. Sigmoids were popular due their intuitive nature of a neuron firing; however, they have fallen out of favor for two reasons. The first of these reasons is that they saturate and kill gradients,
which makes updating the model via backpropagation much more challenging. The second of these reasons is that sigmoid outputs are not zero-centered, which produces undesirable zig-zagging dynamics when updating model weights [8].

\[
\sigma(x) = \frac{1}{1 + e^{-x}}
\]  

(2.2)

The Rectified Linear Unit, or ReLU, activation function has replaced the sigmoid activation function as the go-to activation for hidden layers in neural networks. ReLU, which is calculated using Equation 2.3, is faster during training as it avoids the vanishing gradient issue produced by the sigmoid activation function. Additionally, ReLU and its derivative are faster to calculate as its operations are less expensive than those used in the sigmoid activation function [8].

\[
ReLU(x) = \max(0, x)
\]

(2.3)

The softmax activation function is used at the output layer of a model to normalize outputs; this is unlike the sigmoid and ReLU activation functions which are used in the hidden layers. Presented as Equation 2.4, the softmax function provides an intuitive sense for class probabilities by scaling the sum of the outputs to 1; however, it is important to note that these normalized probabilities are not true probabilities or confidences. For this thesis, the model output, which will be the output of the softmax function, will be referred to as a probability or confidence. The function preserves the ordering of the values. There are other functions that can be used in the final layer of a neural network, including monotonic normalization functions, but softmax is among the most popular and frequently used in existing neural networks.

7
\[ \text{Softmax}_j(z) = \frac{e^{z_j}}{\sum_k e^{z_k}} \]  

(2.4)

### 2.1.2 Loss Functions

During training, it is imperative to assess a model’s accuracy, which is then used to increase the model’s accuracy and ability to produce correct output. To achieve this, loss functions are used to provide a non-binary, quantitative assessment of the model’s accuracy. Using the calculated loss, a model can be updated via backpropagation to improve subsequent inferences.

The first of these loss functions is the categorical hinge loss, which is presented in Equation 2.5. Categorical hinge loss, also known as the max-margin loss, provides a loss value ranging from 1 to 0. This loss function is typically reserved for support vector machines (SVMs); however, I found it to produce stronger gradients when vanishing gradients became an issue with the categorical cross-entropy loss function.

\[ \text{CategoricalHinge}(y_{true}, y_{pred}) = \max(1 - y_{true} \times y_{pred}, 0) \]  

(2.5)

The second of these loss functions is the categorical cross-entropy loss function, which is presented in Equation 2.6. The categorical cross-entropy loss function aims to minimize the cross-entropy between the true and predicted probability distributions, which in the case of classification is achieved by placing all predicted mass at the position of the true class [8]. This loss function is commonly used for multi-class classification models.
2.1.3 Backpropagation

The underlying mathematics that enable neural networks to learn is backpropagation. After a forward pass of the model, a loss value is calculated using the chosen loss function. The gradient of this loss function can be expressed as $\nabla Loss(x)$, which informs how to change the loss function’s input for a maximum change in the loss function’s output.

Of course, the input to the loss function cannot randomly change; namely, the loss function’s input must change due to changes in the previous layers. These gradients can repeat with each layer’s derivative being calculated by $\frac{\partial Layer_i}{\partial Layer_{i-1}}$. To update the weights in layer $i$, the gradients from each layer are chained together using the chain rule. This allows one to determine how the weights in layer $i$ should be updated to minimize the loss calculated at the end of the model. These chained gradients are expressed as Equation 2.7.

$$
\frac{\partial Loss}{\partial weights_i} = \frac{\partial Loss}{\partial Layer_n} \frac{\partial Layer_n}{\partial Layer_{n-1}} \ldots \frac{\partial Layer_i}{\partial weights_i}
$$

(2.7)

By finding the gradient of the loss with respect to the weights in each layer, each layer can be updated to minimize the loss for the given training example. By repeating this process with many training examples, the model’s performance is improved.
Backpropagation is a standard part of deep learning libraries, meaning that while understanding these mechanics is important, it is not usually necessary to implement them in order to train a model.

2.1.4 Regularization

Neural networks have the ability to approximate any function [9]; however, when presented with a limited data set, models can accidentally learn the data set rather than the true distribution the model is supposed to approximate. This issue, known as overfitting, is a common and serious issue when training models. Many regularization methods exist for handling overfitting such as L1 regularization, L2 regularization, and dropout.

L1 and L2 regularization depend on the L1 and L2 distances, which are given by equations 2.8 and 2.9, where \( V_1 \) and \( V_2 \) are individual vectors. While these functions can be used for measuring distances between two arbitrary vectors, they are more commonly used for measuring the growth of weights in a model or for quantifying the difference between two images.

\[
L_1(V_1, V_2) = \sum_i |V_1[i] - V_2[i]| 
\]  
(2.8)

\[
L_2(V_1, V_2) = \sqrt{\sum_i (V_1[i] - V_2[i])^2} 
\]  
(2.9)

For regularization, these functions are used to penalize weights. This is done by finding a L1 or L2 distance between a model’s weights and a zero vector of the same size and then adding this value, multiplied by some tunable parameter, to the model’s
loss function. The training of the model will then try to minimize the loss function, which will pressure the model to keep weights small and thereby avoid overfitting the data while simultaneously improving its overall accuracy.

L1 and L2 differ in how they penalize differences. L2 penalizes individual differences much more significantly than the L1 distance, which leads to the L1 distance favoring sparseness while L2 favors uniformity.

Dropout, a more advanced regularization technique that was first presented by Hinton et al. [10][11], is another technique used to avoid overfitting. To prevent nodes from co-adapting to the data set too much, dropout randomly removes nodes and their connections temporarily during training as seen in Figure 2.2. An intuitive explanation for why this works is that applying dropout during training produces a model that is an average of many other models where strong connections are consistent across models while weak or noisy connections are are lost in the dropout process.

![Figure 2.2: A standard neural network (left) and a neural network after applying dropout (right) [11]](image)

Srivastava et al. found that dropout provides substantial benefits over other regularization methods, and that a dropout rate of 0.5 is optimal for many models and tasks. It should also be noted that dropout only occurs during training; during inference all
nodes are active, which means validation and testing accuracy often exceed training accuracy.

2.2 Deep Learning

Deep learning is characterized by the depth, or number of layers, of a model. By developing models that are composed of multiple processing layers (often containing 10-20 layers [8]), deep learning allows models to learn representations of data with multiple layers of abstraction [12]. The ability to extract and classify learned features is what provides deep learning models with this ability to perform well in the field of computer vision as images typically consist of abstractions that can be learned by these multi-layer models, which is demonstrated by the ability of deep learning models to dramatically surpass the capabilities of traditional computer vision algorithms [13]. Furthermore, while deep learning models are very time consuming to train, they are often very fast at test time, which is typically desirable in practice [8].

2.3 Convolutional Neural Networks

While neural networks have demonstrated significant success and are believed to universal approximators, they can become overwhelmingly large when used for images. Images, which can contain anywhere from thousands to millions of pixels, are very large input vectors. When scaling a neural network to process these inputs, the number of trainable parameters becomes overwhelmingly large.

Convolutional neural networks (CNNs) make the explicit assumption that the input layer is an image. By doing so, the number of parameters for the model can be greatly reduced, which in turn allows for deeper models. As seen in Figure 2.3, CNNs use 3D
volumes of neurons that convolve across layers. The ability to find patterns within an image grows with the model’s depth, which means models with more layers can find increasingly complex patterns.

Convolutional neural networks conclude with one or more dense, or fully-connected, layers. These layers coalesce the the patterns found in the image into the probabilities that the model outputs. AlexNet [14], the first highly successful CNN, is shown in Figure 2.4.

Convolutional neural networks have a number of additional layers and parameters that can be used to improve a model’s performance, but they are outside the purview
of this thesis. The documentation for a deep learning library will contain a list of available layers, at which point a researcher can determine which layers may be useful for their application.

2.4 Feature Extraction and Feature Classification

Early computer vision required domain-specific knowledge when developing image classification algorithms. Designers typically developed feature extraction algorithms such as histogram of oriented gradients (HOG). These feature extractors were manually developed and selected for certain image domains, which proved to be quite challenging. Once an image was reduced via feature extraction, this feature vector was classified using a simple classifier such as a support vector machine (SVM). By coupling these algorithms, image classification algorithms could be created.

Convolutional neural networks have changed image classification. Rather than a two-stage approach, which required significant algorithm development using domain-specific knowledge, developers can use end-to-end machine learning for image classification.

A convolutional neural network used for image classification can be thought of as two stages: feature extraction and feature classification. The first of these stages, feature extraction, consists of the convolutional layers comprised primarily of convolving filters. The second of these stages, feature classification, consists of fully-connected layers that classify the extracted features into class probabilities.
2.5 Transfer Learning

As previously discussed, one of the major challenges when developing deep learning models is the need for substantial computing power and access to large data sets. This differs from humans who can learn tasks (e.g. image classification) in very little time with very little data. Humans can achieve this by utilizing existing knowledge of a domain (e.g. what features are common and distinguishing among animals) to quickly learn new classifications.

This inspires transfer learning, a technique that allows deep learning models to learn faster with less data [15][16]. When implementing transfer learning, a developer retrains an existing, pre-trained model on a different data set. Retraining, which involves training weights that are initialized to weights trained on a different data set, can be applied to some or all of the model’s weights. Layers that are not updated are considered “frozen” or “untrainable” while layers that are updated are considered “unfrozen” or “trainable.”

There are four scenarios guiding how many layers should be frozen during the training process [8]. These decisions are made based on the similarity between the old and new data sets as well as the size of the new data set.

- **Small and similar** Due to the limited data, overfitting is a concern. Since the data set is similar to the original, the previously learned features will likely work well for the new data set. Thus, retrain the classification layers.

- **Small and different** Due to the limited data, overfitting is a concern. Since the data set is different, the classifier and some feature extraction layers should be tuned to better match the new task.
• **Large and similar** With a large data set available, overfitting is less likely to occur. All layers can be adjusted to better fit the new task.

• **Large and different** With a large data set, overfitting is less likely to occur. Additionally, it will be desirable to retrain all layers, including the feature extraction layers, to adjust to the new task.

### 2.6 Residual Networks

Researchers have empirically determined that increasing the depth of a model improves the model as shallow models are subsets of deeper models. Unfortunately, due to the difficulty of training extremely deep models, deeper models can have poor performance unless they are designed and trained correctly.

It is confusing that a deeper model would not perform as well as its shallower counterpart. After all, a deep model consisting of a shallow model followed by a series of identity functions (until the depth of the deeper model is obtained) should have the same performance as the shallower model. This unexpected result is due to the vanishing gradient problem [17], which is when the backpropagated gradients become so small that they cannot be used to update the model’s weights. When the gradients are unable to be backpropagated appropriately, the model’s performance suffers.

Residual networks (ResNets) presented by He et al [18] aim to solve this issue by allowing signals to take shortcuts through the model as demonstrated in Figure 2.5. These shortcuts enable the signals to pass forward through the identity function, and then for gradients to backpropagate through without vanishing.

Figure 2.6 shows ResNet-34, a 34-layer residual network. Larger ResNets with depths of 50, 101, and 152 are expanded by increasing the block sizes (i.e. changing the
Figure 2.5: Fundamental building block of residual networks [18]

number of layers in Figure 2.5). This allows for a dramatic increase in the number of layers in a ResNet while ensuring there is still a somewhat direct path back to the model’s input via shortcut connections.
Figure 2.6: A 34-layer residual network (the left portion feeds into the right portion) designed for use with 1000 classes [18]
This chapter introduces related papers and developments that are related to the discovery of adversarial examples, the development of physically robust adversarial examples, and proposed defenses and their efficacy against these attacks. Namely, this chapter is three sections: early adversarial examples, physically robust adversarial examples, and defenses against adversarial examples.

3.1 Early Adversarial Examples

In 2014, Szegedy et al. found that hardly perceptible perturbations, which can be found by maximizing the network’s prediction error, can produce misclassifications by the network [7]. It was then discovered that perturbing just a single pixel can result in misclassifications [19]. Additionally, researchers found that it is possible to produce artificial images that are completely unrecognizable to humans while being classified with extremely high confidence by deep neural networks [20]. The Fast Gradient Sign Method, presented by Goodfellow et al., produces adversarial examples by adding imperceptible perturbations to images [4]. Furthermore, they argue that these gradient-based techniques are highly effective due to the linearity of neural networks, despite the early explanations that argued adversarial examples were the product of highly non-linear, overfitted models.

Significantly, Szegedy et al. found that adversarial examples generated on one network can produce misclassifications on a different network [7]. It was also found that these perturbations are more robust to transferability if generated by a single-step attack.
rather than a multi-step attack [21]. These discoveries were fundamental to the development of black-box attacks, in which an attacker approximates a model and then attacks the substitute model [22].

3.2 Physically Robust Adversarial Examples

Early adversarial examples assumed attackers had the ability to feed data directly into a machine learning model. While this was effective for demonstrating the existence of adversarial examples, the threat of these attacks was limited by the fact attackers typically do not have direct access to a model. Kurakin et al. were able to demonstrate that simple adversarial examples were able to survive the physical transformations that result from photographing a printed adversarial example [23]. This work was expanded on by Athalye et al., who successfully manufactured the first 3D adversarial object [24]. Additionally, Eykholt et al. developed physically robust traffic signs that were carefully crafted such that the perturbations mimic graffiti, which enables the attack to “hide in the human psyche” [5].

3.3 Defenses to Adversarial Examples

To mitigate the damage caused by these attacks, researchers have explored potential techniques to avoid misclassifying adversarial examples. One such approach is defensive distillation [25], which relies on neural network distillation [26]; however, defensive distillation has since been found to be ineffective against adversarial examples [27]. Another technique, which uses an ensemble of weak defenses in an attempt to develop a strong defense [28], is believed to be an insufficient defense against adversarial examples [29]. Another potential defense mechanism is training on adversarial examples; however, it was observed that such models will ultimately perform better
on adversarial examples than on clean examples [21]. Finally, some researchers believe that due to the fact deep neural networks are theoretically universal approximators [9], efforts should be focused on improving the training of neural networks, not on developing defense mechanisms [30].

At this time, there are no effective defenses against adversarial examples.
Deep learning is a computationally-intensive field that requires significant hardware, software, and data to achieve good results. This chapter will detail these tools—hardware, software, and data—that were used in the development and analysis of this thesis.

### 4.1 Computing Platform

Deep learning requires significant hardware resources to develop and utilize models. All the deep learning work for this thesis was conducted in Cal Poly’s Massively Parallel Accelerated Computing (MPAC) Lab. The MPAC lab offers nearly 40 machines that feature 28-core Intel Xeon CPUs running at 2.30 GHz. Each of these machines also contains an NVIDIA GeForce GTX 980 GPU with 3593 MB of memory. Finally, each machine offers 2 TB of local disk space that can be used for storing models and data sets.

### 4.2 Software

Python is the dominant programming language in the machine learning community due to its high-level nature and wealth of libraries available, which makes prototyping and developing models and pipelines very easy. This thesis used Python 3 in order to utilize the latest releases of libraries.
4.2.1 Deep Learning

The deep learning frameworks used are TensorFlow [31] and Keras [32]. TensorFlow is an open-source machine learning library developed by Google that uses dataflow graphs to represent computation [33]. TensorFlow can be easily installed to run with both CPU and GPU, which makes learning much faster. Keras, a modular neural network library that can run on top of TensorFlow, offers users a high-level API to the TensorFlow library [34]. This makes it easy to build, modify, train, and test models in Python.

Together, these libraries offer an easy way to implement deep learning techniques. These libraries allow developers to spend less time implementing the minutiae of deep learning and more time on making cutting-edge developments.

4.2.2 Image Manipulation

Image manipulation was achieved using OpenCV [35], an open-source computer vision library written in C and C++ with APIs for Python and C++ [36]. OpenCV offers many functions for image processing and manipulation. OpenCV was used to produce pseudo-natural transformations to images. These pseudo-natural transformations could then be used to assess the robustness of attacks in the physical world.

4.2.3 Analysis

Data analysis and figure generation was achieved using Matplotlib, pandas, and seaborn. These Python libraries make it easy to extract meaningful conclusions from data and produce complex figures.
4.2.4 Data & Models

The models used in this thesis were trained on the ImageNet [37] and German Traffic Sign Recognition Benchmark [38] data sets. These data sets are open source and commonly used in research, which make them good benchmarks for this work. Note that the models trained on the ImageNet data set were borrowed pre-trained and borrowed from the Keras model zoo.

4.3 Summary

Table 4.1 provides a summary of the tools used in this thesis.

<table>
<thead>
<tr>
<th>Type</th>
<th>Resource</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardware</td>
<td>Cal Poly MPAC Lab</td>
</tr>
<tr>
<td>Software</td>
<td>Python 3, TensorFlow, Keras, OpenCV, Matplotlib, pandas, seaborn</td>
</tr>
<tr>
<td>Data</td>
<td>ImageNet, GTSRB</td>
</tr>
<tr>
<td>Models</td>
<td>Keras model zoo (ResNet-50, VGG16, MobileNet)</td>
</tr>
</tbody>
</table>
This chapter will outline the Fast Gradient Sign Method (FGSM) developed by Goodfellow et al. [4]. FGSM has become immensely popular for developing adversarial examples, and has been used to create perturbations in other attacks. Understanding and implementing the FGSM technique is essential to success in exploring and developing additional attacks.

5.1 Background

An early and significant contribution to the field of adversarial examples for computer vision models was the introduction of the Fast Gradient Sign Method. In the corresponding paper, Goodfellow et al. contradicted existing literature by arguing that machine learning models are vulnerable to adversarial examples due to their linear nature. By adopting the viewpoint that a target model is linear in nature, one can quickly generate adversarial examples. More specifically, this linear assumption allows one to exploit Equation 5.1, where $w$ is some weight vector, $x$ is the input image, and $\eta$ is the target perturbation.

$$w^T \tilde{x} = w^T x + w^T \eta$$  \hspace{1cm} (5.1)

Due to the large number of input values that make up an image, many small adjustments to a perturbation can accumulate thereby allowing for large changes in a model’s output activation. Thus, a collection of seemingly insignificant changes
in the input pixels can produce a significant change in the model's output, thereby producing a misclassification.

The question then becomes how one should generate perturbation \( \eta \). Unlike updating a model's weights, which is expressed by Equation 5.2 where \( \theta \) is the model’s parameters, \( x \) is the input image, \( y \) is the target model output, \( J(\theta, x, y) \) is the loss function used to train the model, and \( \epsilon \) is a scaling value, Goodfellow et al. proposed FGSM, which is expressed by Equation 5.3. The difference between these equations is that Equation 5.2 updates a model’s parameters while Equation 5.3 updates the input to the model.

\[
\nabla_\theta J(\theta, x, y) \tag{5.2}
\]

\[
\eta = \epsilon \text{sign}(\nabla_x J(\theta, x, y)) \tag{5.3}
\]

Goodfellow et al. found that FGSM reliably causes a wide variety of models to misclassify inputs. One of the most well-known FGSM examples is shown in Figure 5.1, in which an image of a panda is misclassifed as a gibbon (a type of ape) with higher confidence even than the original panda.

Interestingly, the Fast Gradient Sign Method is more effective than a Fast Gradient Method (remove the \( \text{sign}(\cdot) \) function from Equation 5.3). This is likely the result of forcing small gradients, which are typically far more numerous than large gradients, to make larger contributions to input image, and thereby the network’s overall activation.
Figure 5.1: Goodfellow et al.’s seminal FGSM example [4]

![Image of Goodfellow et al.’s FGSM example]

Figure 5.2: Top-1 and top-5 accuracy of Inception v3 on images produced by various attack methods over a varying $\epsilon$ [23]

Kurakin et al. also introduced two similar methods similar to FGSM for generating adversarial examples: Iterative Fast Gradient Sign Method and Iterative Least-Likely Class Method [23]. The Iterative FGSM attack, which is a multi-step attack, recalculates the perturbation $\eta$ is recalculated. The Iterative Least-Likely Class Method, is similar to the Iterative FGSM attack, but it differs by being an untargeted attack that simply aims to destroy the confidence in the actual class. The effectiveness of these methods, which is shown in Figure 5.2, is significantly higher than for the basic FGSM introduced in Equation 5.3.
As iterative methods are multi-step attacks that produce much finer perturbations by exploiting subtleties in a model, iterative methods are typically less effective at higher values of $\epsilon$ in addition to being less likely to transfer between models or be physically robust.

5.2 Verification of Existing Work

Iterative FGSM, presented as Algorithm 1, is easily implemented using deep learning libraries. Furthermore, due to optimizations within these libraries, a forward pass through a model and the corresponding backpropagation are very fast, which makes generating adversarial examples quite fast (anywhere from a few seconds for easy images to a few minutes for challenging images).

Algorithm 1: Iterative Fast Gradient Sign Method

Result: $adv\_img$

$adv\_img = clean\_img$;

$target = one\_hot(target\_class, num\_classes)$;

for $epoch$ in epochs do

    loss = loss\_function(target, model.output);
    grads = gradients(loss, model.input);
    signs = -1 * signs(grads);
    $adv\_img = \text{clip}(adv\_img + \epsilon \times signs, lower\_bound, upper\_bound)$;

end

return $adv\_img$;

Though not stated by Goodfellow et al. or Kurakin et al., this thesis found that the ability to convert images into adversarial examples depends significantly on the model being attacked and the model’s confidence in the target class. If a model is extremely confident in the correct class, it makes generating adversarial examples
quite challenging. For instance, perturbing a trolley to a passenger car is quite easy (the two classes are semantically similar with a set of similar or shared features) while perturbing a trolley into a cucumber is quite challenging (there is little overlap in semantic meaning or key features).

Additionally, this thesis observed that though the categorical cross-entropy loss function is most effective for perturbing images into neighboring classes (e.g. trolley to passenger car), the categorical hinge loss is more effective for perturbing images into distance classes (e.g. trolley to cucumber). This was due to the gradients from the categorical cross-entropy loss being flattened during backpropagation and completely vanishing in the case of challenging images, making it impossible to attack these challenging examples with the cross-entropy loss function.

Figure 5.3 is a sample output from the implemented pipeline targeting the ResNet-50 model from the Keras model zoo, which is trained on the ImageNet data set. The clean image (left) is easily classified as a trolley by both a human and the target model (softmax activations of 0.998 for trolley, 1.05e-08 for cucumber). The carefully crafted noise (middle)—dramatically scaled for contrast—is meaningless to both a human and the model (the top classification for the noise is a softmax activation of 0.362 for a park bench). The adversarial image (right) is easily classified as a trolley by the human; however, it is confidently misclassified by the target model as a cucumber (softmax activations of 2.22e-13 for trolley, 1.0 for cucumber).

The probability of the target class over these iterations is shown in Figure 5.4. More specifically, the figure shows the probabilities of any classes that were the top class at any point during the attack. This indicates that the image of the trolley is almost immediately destroyed, at which point the model cannot make any prediction with significant confidence; however, at the conclusion of the attack, the cucumber class quickly emerges as the dominant class.
Initial experiments regarding the robustness of these adversarial images across models and the physical domain were conducted using the trolley images in Figure 5.5. The methodology of these experiments is as follows. First, images were generated by applying both the Non-Iterative FGSM and Iterative FGSM until the images were classified by the ResNet-50 model as adversarial examples. Second, the images were passed to the ResNet-50, VGG16, and MobileNet models from the Keras model zoo. Finally, the images were displayed on a smart phone, photographed by another smart phone, and cropped before being classified again. This photographic transformation provides changes in contrast, brightness, blur, noise, and data (lossy JPEG encoding).

The results of these initial experiments, which are shown in Table 5.1 (robustness across models) and Table 5.2 (robustness across models and the physical domain) disagreed with those found by Kurakin et al. [23]; however, additional experiments were able to find some agreement with Kurakin et al.’s results. This discrepancy appears to be due to the confidence in the initial classification.

Rather than using the trolley image, which is confidently classified by the classifier, the attack utilizes a less confident image, the washing machine from Kurakin et al. [23], which is shown in Figure 5.6. It was observed that using an image with a lower

Figure 5.3: Clean image of a trolley (left) with a softmax activation 0.998, the corresponding perturbation scaled for visualization (middle), and the adversarial example (right) with a softmax activation of 1.0
Table 5.1: Robustness of trolley adversarial images generated on ResNet-50 across multiple models

<table>
<thead>
<tr>
<th>Model</th>
<th>Clean Image</th>
<th>Non-Iterative FGSM</th>
<th>Iterative FGSM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Trolley</td>
<td>Passenger Car</td>
<td>Trolley</td>
</tr>
<tr>
<td>ResNet-50</td>
<td>0.979</td>
<td>0.019</td>
<td>5.75e-4</td>
</tr>
<tr>
<td>VGG16</td>
<td>0.994</td>
<td>0.004</td>
<td>0.976</td>
</tr>
<tr>
<td>MobileNet</td>
<td>0.995</td>
<td>0.004</td>
<td>0.996</td>
</tr>
</tbody>
</table>

Table 5.2: Robustness of trolley adversarial images generated on ResNet-50 across multiple models after being transformed through the physical domain

<table>
<thead>
<tr>
<th>Model</th>
<th>Clean</th>
<th>Non-Iterative</th>
<th>Iterative</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Trolley</td>
<td>Passenger Car</td>
<td>Trolley</td>
</tr>
<tr>
<td>ResNet-50</td>
<td>0.942</td>
<td>0.056</td>
<td>0.456</td>
</tr>
<tr>
<td>VGG16</td>
<td>0.997</td>
<td>0.002</td>
<td>0.997</td>
</tr>
<tr>
<td>MobileNet</td>
<td>0.979</td>
<td>0.020</td>
<td>0.991</td>
</tr>
</tbody>
</table>
clean confidence increases the likelihood of adversarial examples successfully transferring across models (Table 5.3) in addition to surviving transformations through the physical domain (Table 5.4).

Figure 5.6: Clean image of a washer (left), a Non-Iterative FGSM adversarial image targeting the safe class (middle), and an Iterative FGSM adversarial image targeting the safe class (right)

While the target class is not always the maximum class, and occasionally the actual class remains the dominant class, these attacks typically prove to be sufficiently effective to reduce the actual class dramatically (to the point where it is indistinguishable from other classes), even across models and through physical transformations.
Table 5.3: Robustness of washer adversarial images generated on ResNet-50 across multiple models

<table>
<thead>
<tr>
<th>Model</th>
<th>Clean Washer</th>
<th>Clean Safe</th>
<th>Non-Iterative Washer</th>
<th>Non-Iterative Safe</th>
<th>Iterative Washer</th>
<th>Iterative Safe</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet-50</td>
<td>0.689</td>
<td>0.001</td>
<td>0.190</td>
<td>2.38e-08</td>
<td>0.001</td>
<td>1.0</td>
</tr>
<tr>
<td>VGG16</td>
<td>0.504</td>
<td>0.004</td>
<td>0.927</td>
<td>0.051</td>
<td>0.878</td>
<td>0.897</td>
</tr>
<tr>
<td>MobileNet</td>
<td>0.987</td>
<td>0.176</td>
<td>0.136</td>
<td>0.878</td>
<td>0.0579</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.4: Robustness of washer adversarial images generated on ResNet-50 across multiple models after being transformed through the physical domain

<table>
<thead>
<tr>
<th>Model</th>
<th>Clean Washer</th>
<th>Clean Safe</th>
<th>Non-Iterative Washer</th>
<th>Non-Iterative Safe</th>
<th>Iterative Washer</th>
<th>Iterative Safe</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet-50</td>
<td>0.692</td>
<td>0.021</td>
<td>0.065</td>
<td>0.060</td>
<td>0.019</td>
<td>0.834</td>
</tr>
<tr>
<td>VGG16</td>
<td>0.275</td>
<td>0.010</td>
<td>0.250</td>
<td>0.050</td>
<td>0.148</td>
<td>0.524</td>
</tr>
<tr>
<td>MobileNet</td>
<td>0.803</td>
<td>0.019</td>
<td>0.148</td>
<td>0.508</td>
<td>0.267</td>
<td></td>
</tr>
</tbody>
</table>

5.3 Extension of Existing Work

While Goodfellow et al. [4] and Kurakin et al. [23] provide substantial developments regarding FGSM, they provide little data to support any argument regarding the linearity or robustness of the attack. The goal of this section is to provide that data.

5.3.1 Linearity of Iterative FGSM

Goodfellow et al. claim that it is the linear nature of neural networks that results in the vulnerability to adversarial examples. To assess the linearity of the Iterative FGSM attack, the correlation of sign maps over the course of the attack was plotted, which is shown in Figure 5.9.

Sign maps are defined to be the sign of the gradients in Equation 5.3. Then, the total number of matching signs is divided by the total number of pixels in the sign map, which produces the correlation.
A sign map correlation matrix is presented in Figure 5.9, which shows little correlation between sign maps. A value of 0.5 suggests no correlation, which is the dominant value throughout the correlation matrix; however, in the last few iterations, the correlation values increase slightly. These last iterations are shown in Figure 5.10 in an effort to improve readability.

It is interesting to note that the beginning and end of the attack have higher correlation values. Considering the probabilities over the course of the attack (see Figure 5.4 for the class probabilities during this attack), it seems clear that the correlations of the sign maps go through three stages. First, the sign maps are highly correlated as the confidence in the actual class quickly diminishes. Next, the sign maps have little correlation as the probabilities and perturbations wander aimlessly in search for
Figure 5.9: Correlation matrix of sign maps during all iterations of an Iterative FGSM

Figure 5.10: Correlation matrix of sign maps during the final 10 stages of an Iterative FGSM
the target class. The final stage is when the sign maps become highly correlated once more and the confidence in the target class rapidly increases.

Additional studies to find localized correlations between sign maps yielded noise. Overall, it seems that while there are pockets of linearity, the majority of the domain is non-linear, making it challenging to effectively navigate the space while relying on the linearity of neural networks. This provides context for Goodfellow et al.’s claim.

5.3.2 Robustness of FGSM

Another question regarding the adversarial examples generated by the FGSM attacks is their robustness. Though much of this has been addressed in Section 5.1 and Section 5.2, this section will provide new and more targeted studies of robustness.

The first of these studies is the effect of random noise on an adversarial example. This is motivated by the observation that perturbations appear (to a human) to be random noise. With this in mind, this study iteratively generates and applies randomly generated sign maps to an image while observing the class probabilities.

Though it is observed that these randomly generated sign maps can reduce an adversarial example’s target probability and eventually remove the adversarial image from the target class, the probability of the actual class is never recovered. For instance, Figure 5.11 shows an adversarial examples before and after the application of approximately 20 iterations of random noise. When the target probability (cucumber) is lost due to the noise, the original probability (trolley) is minimal and the model cannot reliably suggest the image on the right is a trolley.

As a control, this same procedure is applied to the clean image of the trolley. The clean image proves to be significantly more robust to noise and only deteriorates out
Figure 5.11: Adversarial example (left) that is misclassified as a cucumber (0.9999) and the image after the liberal addition of noise (right) misclassified as a flowerpot (0.0714)

of the original class (trolley) after more than 50 iterations of randomly generated sign maps; however, due to the increased number of iterations, the image is significantly damaged, which can be seen in Figure 5.12. Overall, this suggests that adversarial examples are quite fragile compared to their clean counterparts as the adversarial images are merely false representations of objects.

Figure 5.12: Clean example (left) and the image after the liberal addition of noise (right) misclassified as a park bench (0.2988)

Another interesting question regarding the robustness of FGSM attacks is what happens if FGSM is applied to an adversarial example in an attempt to take the image back to its original class. Even after an adversarial example has been crafted such that the softmax activation for the target class is 1.0 (perhaps taking dozens of iterations), the confidence in the adversarial class quickly diminished back to the true class
during an FGSM attack attempting to take the adversarial example back to its true class. While this is not a potential defense technique, it is an interesting observation as it suggests FGSM targets subtleties in the model that are destroyed by reversing FGSM rather than obfuscating and destroying the key features used in classification.

The final study regarding the robustness of images generated by the FGSM attack is the effect of simple transformations on an adversarial example. The effects of these transformations—rotation, brightness, Gaussian noise, Gaussian blur, and contrast—are as follows.

- **Rotation**: The Non-Iterative FGSM images were typically able to survive small angle rotational transformations. Rotation was the only transformation that could recover the original class.

- **Brightness**: The target class was lost in all adversarial images; however, the actual class was not recovered and the model was unable to make a confident decision regarding any class.

- **Noise**: The target class was lost in all adversarial images; however, the actual class was not recovered and the model was unable to make a confident decision regarding any class.

- **Blur**: The target class was lost in all adversarial images; however, the actual class was not recovered and the model was unable to make a confident decision regarding any class.

- **Contrast**: The target class was lost in some but not all adversarial images. Additionally, the actual class was recovered in some cases.

These transforms not only provide some insight to the robustness of adversarial examples in the physical world, but they suggest that the FGSM attacks target weaknesses
in the model’s training set or parameters rather than fundamental attacks to the semantic meaning of the image. This is supported by the fact that rotations of only a few degrees have the ability to reduce images back to their original class.

Additionally, these studies suggest that popular preprocessing techniques used in classical computer vision such as contrast optimization and blurring may provide little value in defending against adversarial examples.

Finally, note that the reason for the lack of quantitative data for these transformations is due to the overwhelming number of variables across these transformations. Between different transformation types, numerous variables (angles, kernels, distributions, and scaling values), and testing across numerous images, it is challenging to find a meaningful way of testing and presenting these transformations. Rather, select parameters were chosen to get a sense of the effects of the transformations, which were then summarized in a bulleted form.

5.3.3 Perturbations

Two interesting questions arise regarding the perturbations themselves used in generating adversarial examples, which relate to the greater question of whether perturbations modify the semantic meaning of an image or whether they attack subtleties in a network.

The first of these questions is whether perturbations developed on one image can be used to damage other images. This experiment was conducted by perturbing an image towards a target class while modifying a second image with the same perturbations. Not only were the perturbations ineffective at helping the second image reach the target class, they proved to do little to the true class of the second image. These
results seem to support the idea that perturbations are unique to the image and not the underlying object or model.

The second question is whether interesting properties can be discovered by classifying perturbations. Unfortunately, classifying perturbations and sign maps provides little meaningful information. Not only are these inputs not classified as the original or target classes, but the classifications are typically the same classifications given to randomly generated perturbations, which agrees with Goodfellow et al. [4]. Namely, it seems that to a model, stand-alone perturbations are meaningless to a model until they are added to the image for which they were generated.

Considering the observations that FGSM perturbations are ineffective on other images and have no interesting classification themselves, it seems perturbations produced by FGSM do not change the semantic meaning of images. This is particularly intriguing as these attacks still have the potential to carry across models, which suggests that FGSM attacks are still capable of finding underlying visual clues that are being leverage by multiple models. Consequently, this is promising for generating physically robust attacks.

5.4 Conclusions

The family of Fast Gradient Sign Method attacks provide intriguing insights to neural networks. Developing successful attacks requires deep understanding of the target by the adversary, and the FGSM attacks provide unique insight to the linearity of models and their decision space.

More importantly, the FGSM attacks are a critical baseline for generating more sophisticated adversarial examples. FGSM provides a reliable way of perturbing images,
and advanced techniques discussed in later sections provide ways of controlling perturbations during the course of an attack.
A natural question after the development of the Fast Gradient Sign Method is how can perturbations be designed to be physically robust? As attackers rarely have direct access to a model’s input (and if they do, there are more effective attacks than feeding in adversarial inputs), adversaries have few options for attacking a model. In the case of computer vision models, this is typically in the form of providing adversarial inputs to a camera, which is then passed to the target model; however, this presents the challenge of developing adversarial inputs that can survive the physical transformations associated with manufacturing and photographing. In an effort to overcome this challenge, Eykholt et al. [5] proposed Robust Physical Perturbations (RP$_2$).

6.1 Background

Seeking to create adversarial examples that can survive changing conditions and remain effective at fooling a classifier, the RP$_2$ pipeline developed by Eykholt et al. considers environmental conditions, spatial constraints, physical limits on imperceptibility, and the effects of fabrication error on the adversarial image. An overview of the RP$_2$ pipeline is given in Figure 6.1.

The attack begins by sampling numerous views of the same object from a distribution, which allows the attack to generalize to the physical object rather than attacking a single view. Next, backpropagated gradients are regularized using L1 regularization to locate the most vulnerable regions of a sign. These regions are identified and
used to create masks to limit and control the portions of a sign that are perturbed. Next, perturbations are applied and constrained using either the L2 or L∞ norms in addition to other factors such as the Non-Printability Score (NPS) [39]. Finally, the adversarial sign is manufactured and evaluated.

A simple equation for generating perturbations is formalized as Equation 6.1, where \( \lambda \) is a hyperparameter to control the regularization of the perturbation, \( \delta \) is the perturbation, \( p \) denotes the distance function to be used for the norm of \( \delta \), \( J(\cdot, \cdot) \) is the loss function, \( f_\theta(\cdot) \) is the target model, \( x \) is the input image, and \( y^* \) is the target label.

\[
\arg\min_\delta \lambda \|\delta\|_p + J(f_\theta(x + \delta), y^*)
\]  

Equation 6.1 is then extended to generate robust, spatially-constrained perturbations. This is formalized by Equation 6.2, where \( M_x \) is the perturbation mask to control the locations of the perturbation, \( NPS \) is the Non-Printability score, \( E \) is the expected value of \( x_i \) that one might expect to find in the distribution of physical and digital
transformations $X^V$, $x_i$ is an image in the distribution, and $T(\cdot)$ is the alignment function that transforms the perturbation in the same way the object in $x_i$ is transformed (e.g. if the object in $x_i$ is rotated, the perturbation is rotated as well).

$$\arg\min_{\delta} \|M_x \cdot \delta\|_p + NPS + \mathbb{E}_{x_i \sim X^V} J\left(f_\theta(x_i + T_i(M_x \cdot \delta)), y^*\right)$$

(6.2)

By controlling the locations of the perturbations and regularizing them, Eykholt et al. aim to create perturbations that “hide in the human psyche.” An example of this is shown in Figure 6.2, which shows genuine graffiti (left) and an adversarial example (right), crafted by Eykholt et al., that aims to mimic genuine graffiti. By doing this, the attack is less likely to be detected and removed by humans.

![Figure 6.2: Genuine graffiti (left) and an adversarial example (right) created by Eykholt et al. [5]](image)

6.2 Verification of Existing Work

The first step of the RP$_2$ is to locate the weak portions of the image. This is done by using the L1 regularization to locate the most vulnerable portions of the image, as demonstrated in Figure 6.3. By locating these weak points in the image, masks can be created to target these regions, which maximizes the effect of the perturbations while minimizing the amount of the image that needs to be perturbed.
Using Figure 6.3, a mask can be manually created to target the most vulnerable portions of the image. This mask is seen in Figure 6.4.

Figure 6.4: Manually generated mask guided by the $L_1$ regularizations in Figure 6.3

This mask was then used to attack the original image shown in Figure 6.3. The algorithmically generated adversarial example is the left image in Figure 6.5. This image was then used to inspire the manual creation of a generalized, manufacturable
image, which is shown in right image in Figure 6.5. The softmax activations for 90 kph (not the true class) are remarkably high: 0.947 for the algorithmically generated image and 0.686 for the manually created image.

![Image](image-1.png)

**Figure 6.5:** Algorithmically generated adversarial example (left) constrained by the mask in Figure 6.4 and a manually created adversarial example (right) inspired by the algorithmically generated adversarial example

Unfortunately, despite repeating this process for several images and generalizing the results, the adversarial examples proved not to be robust. When rotated slightly, the images quickly deteriorated back into the stop sign class.

Note that in some cases, the adversarial images generated using the RP$_2$ pipeline transferred well across models causing misclassifications with extremely high softmax activations; however, this was rare and suggests the attack is targeting specific models rather than generalized weaknesses in the object.

### 6.3 Limitations of Existing Work

This section will address the numerous shortcomings of Eykholt et al.’s RP$_2$ pipeline. These shortcomings not only make the pipeline challenging to implement and execute, but they also reduce the effectiveness of the attack across models and in the physical domain.
The first issue facing the RP$_2$ pipeline is its reliance on backpropagating gradients. Though this seems like it is not an issue (after all, models are trained using backpropagation), attacking strong models can be challenging or even impossible due to vanishing gradients. If a strong model predicts a stop sign with a softmax activation of 1.0, then the backpropagated gradients will be 0. This makes it impossible to generated L$_1$ masks or the subsequent perturbations necessary to execute the attack. Even in cases where the softmax activation is not 1.0, it is possible to have vanishing gradients that arrive at the input layer (the image) as 0.

The second limitation of the RP$_2$ approach is related to pixelation and the Non-Printability Score (NPS). Even though NPS can be used to better manufacture adversarial examples to represent their digital counterparts, the NPS does not regulate or reduce the pixelation observed in the adversarial examples. This pixelation exploits subtleties in the model, rather than finding robust, homogeneous perturbations that can survive physical transformations. While this makes for fast and effective digital attacks (e.g. Fast Gradient Sign Method), these results quickly fail when faced with physical transformations.

Another of RP$_2$’s limitations pertains to the constraints placed on the attack pipeline. While these constraints are impressively well-designed to overcome many of the challenges faced when generating physically robust adversarial images, they are overly restrictive and make executing an attack challenging or impossible. Finding a way to relax these constraints would be highly beneficial to attackers as this would increase the flexibility of the attack.

The final limitation of the RP$_2$ attack pipeline is the number of stages. As discussed in Section 3.1, multi-step attacks are more likely to seek the subtleties of a model while single-step attacks are more likely to find general weaknesses that will then
carry over between models and into the physical domain. Eykholt et al.’s work uses a myriad of steps, which strongly suggests the attack is inherently weak.

6.4 Conclusions

Eykholt et al.’s RP$_2$ pipeline offers valuable contributions to the field of physically robust adversarial generation. In an attempt to overcome the demands of developing physically robust attacks, Eykholt et al. draw from a distribution of images, which is used to develop more general attacks. Furthermore, they propose the use of masks to control and localize the attack to the weakest regions of the object, which can be identified by using the L$_1$ regularization. Despite these contributions, the attack is inherently limited due to its multi-step approach, which in turn reduces its real-world effectiveness due to the inability to attack strong models, its challenging implementation, and its overly aggressive constraints.
Although computers have achieved performance to rival that of humans on image classification tasks, deep learning researchers lack understanding as to how machine learning models produce their output. More specifically, researchers are confused by which features cause strong signals in a model to produce classifications. One method of studying the spatially localized features used by a model is to use occlusion analysis, a technique similar to eye tracking [40] on humans. This chapter will introduce sliding occlusions, which were first proposed by Zeiler et al. [6]. Then a novel attack will be presented, whose attack pipeline depends on these sliding occlusions.

7.1 Background

Zeiler et al. first introduced sliding occlusions as a technique to analyze the sensitivity of classifiers. By occluding portions of an image, one can observe which portions of an image are important to the correct classification (e.g. the object itself) and which portions are expendable (e.g. the background). Furthermore, this analysis not only reveals the features important for classification, but it can help provide confirmation that a model is identifying the desired objects rather than surrounding contextual clues commonly associated with an object (e.g. green grass behind a soccer ball).
7.2 Verification of Existing Work

Though images can be studied by occluding a targeted portion of the image, a systematic approach provides a comprehensive study of an entire image. This approach, outlined by Algorithm 2, provides researchers with the ability to see the features and characteristics critical to classification by a model.

Algorithm 2: Sliding Occlusion Heat Maps and Composite Images

Result: heat_map, composite_img

prob_sum = zeros(img.shape)
count = zeros(img.shape)

for row = 0 to height step stride do
    for col = 0 to width step stride do
        occluded_img = copy(img)
        occluded_img[row:row+size, col:col+size] = 0
        prediction = predict(occluded_img)[true_label]
        prob_sum[row:row+size, col:col+size] += prediction
        count[row:row+size, col:col+size] += 1
    end
end

heat_map = normalize(prob_sum / count)
composite_img = heat_map * img

return heat_map, composite_img

The sliding occlusion implementation used in this thesis produced the images in Figure 7.1, which shows the original images (left column), occlusion heat maps (middle column), and composite images (right column). Not only does Figure 7.1 provide a sanity check that the model is looking for the primary object in the image, it also highlights the key parts of the object that led to strong classifications.
As observed in Figure 7.1, the teapot is most damaged by occluding the spout. The most damaging of these occlusions, as shown in Figure 7.2, reduced the model’s teapot softmax activation from 0.706 to 0.011, with the dominant class being a combination lock (softmax activation of 0.430). Note that the heat maps and composite images are normalized to increase the contrast between critical and non-critical regions of the image.
Note that it is possible to change the size of the occlusion during occlusion analysis. While larger occlusions can help find larger features during analysis by occluding large features in their entirety, they also provide lower resolution in the resulting heat maps and composite images as there are fewer possible positions for the occlusion. In contrast, small occlusions provide a fine resolution; however, they are limited in their ability to obscure large features, meaning the heat map and composite image may not properly represent the original image. The occlusion size is a parameter that can be adjusted by a researcher. It is best to simply sweep through a few different values as this provides a comprehensive overview and takes little time, but occlusions with dimensions roughly 1/8th of the original image are likely close to optimal.

Figure 7.2: Position of most damaging occlusion for the teapot in Figure 7.1

7.3 Extension of Existing Work

Though Zeiler et al. [6] first introduce the idea of occlusions to measure the sensitivity of a convolutional neural networks to different parts of an input image, they do nothing beyond using this as a tool as a sanity check that their model is detecting target objects instead of the surrounding context. This section presents novel uses of these sliding occlusions, including a novel attack that aims to deliberately occlude critical components of an image.
One significant use of the sliding occlusion analysis is the ability to identify object features that are key to correct classification. Consider teapots, which were first introduced in Figure 7.1. One might wonder, what are the characteristics of a teapot that allow a model to correctly identify it? By running the sliding occlusion analysis on multiple pictures of teapots, one can begin to see patterns and draw conclusions regarding strong features. An analysis of teapots, shown in Figure 7.3, indicates that the spout of a teapot is a key characteristic used in classification. Furthermore, the analysis seems to suggest that the knob, handle, and void between the two are other useful features for identifying teapots. It is hypothesized that these features provide distinction from other related ImageNet classes such as coffee pots, Crock pots, and frying pans, which do not have overhead handles. This information is valuable as it provides insight to characteristics that are not typically considered by a human, but are ultimately valuable to an attacker as they are key to the model’s success.

To further explore this hypothesis, Figure 7.4 is provided. These teapots, which are missing a handle or a spout, are classified with high softmax activations (0.791 and 0.964). Particularly in the case of the teapot without a spout (top row of Figure 7.4), this suggests that the model is identifying multiple features, some of which are not typically considered by a human. It is worth noting though that these teapots are significantly less robust to occlusions than the previously presented teapots, which is likely due to their missing key features.

In conclusion, this section has demonstrated that sliding occlusions can be used to locate key features used for classification on an object. By running this analysis on multiple images of the same class, the key features for the class distribution can be
identified, which can be useful information for designers or adversaries when training or attacking models.

7.3.2 Attacking a Model using Sliding Occlusions

One application of occlusion attacks is against traffic sign classifiers that might be used in autonomous vehicles. To explore this application, three different models were
trained on the German Traffic Sign Recognition Benchmark (GTSRB) data set using transfer learning and the initial ImageNet weights from the Keras Model zoo. These models are a ResNet-50 model, a ResNet-50 model with some adversarial training, and a MobileNet model. The test accuracies for these models are 98.58%, 99.12%, and 96.86%, respectively. Note that the ResNet-50 models perform significantly better than the MobileNet model.

Stop signs, which are safety-critical traffic signs, are considered in this untargeted attack. To find the critical parts of a stop sign, the sliding occlusion analysis, shown in Figure 7.5, was conducted on three stop signs to better model the stop sign distribution. This analysis revealed two areas that are most important for the correct classification of stop signs: the area underneath the text “STOP” and the area between and on the letters “T” and “O”. This suggests that occluding these regions will
make correct classification much more challenging for a classifier. The extent of the
damage caused by these occlusions is summarized in Table 7.1.

Figure 7.5: Stop sign sliding occlusion analysis, which identifies the critical
identifying features to be the regions underneath and between the letters
“T” and “O”

Table 7.1: Sliding occlusion analysis of top two classes for the stop signs
in Figure 7.5 (rows in this table correspond to rows in Figure 7.5)

<table>
<thead>
<tr>
<th>Clean Image</th>
<th>Occluded Image</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top</td>
<td>Second</td>
</tr>
<tr>
<td>Class</td>
<td>Prob.</td>
</tr>
<tr>
<td>Stop</td>
<td>0.944</td>
</tr>
<tr>
<td>Stop</td>
<td>0.630</td>
</tr>
<tr>
<td>Stop</td>
<td>0.706</td>
</tr>
</tbody>
</table>

56
To generate the adversarial stop sign, an unperturbed stop sign is loaded into an image processing program. The design of this occlusion is as follows. First, a black occluding region was added underneath the letters “T” and “O” to obscure these critical regions. Next, a cartoon truck was added between the letters “T” and “O” to occlude the other critical region as determined by the sliding occlusion analyses of stop signs. Finally, dashed yellow lines were added to the black occluding region to give the impression the attack is artwork or graffiti rather than deliberate damage to a sign. Note that the key regions were determined by the sliding occlusions attack; the method of obscuring these regions was artistic in order to allow the attack to “hide in the human psyche” [5].

Figure 7.6: Clean stop sign (left) and adversarial example (right) created using sliding occlusion analysis of multiple stop signs

To evaluate this attack, the images were displayed on a 24-inch monitor and photographed at an angle from approximately 5 feet away. These images were then cropped and fed into the classifier as shown in Figure 7.7. The attack proved to be highly effective against the classifier, with the adversarial image producing a misclassification with a top class of 80 kph. Not only is the model fooled, but the top class is functionally quite different from a stop sign, meaning vehicle behavior is significantly
Table 7.2: Performance of adversarial stop signs against a ResNet-50 model

<table>
<thead>
<tr>
<th></th>
<th>Top Class</th>
<th>Top Prob.</th>
<th>Second Class</th>
<th>Second Prob.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clean</td>
<td>Stop</td>
<td>1.0</td>
<td>Rnd.Abt.</td>
<td>1.95e-11</td>
</tr>
<tr>
<td>Stop 1</td>
<td>Rnd.Abt.</td>
<td>0.763</td>
<td>Stop</td>
<td>0.233</td>
</tr>
<tr>
<td>Stop 2</td>
<td>30 kph</td>
<td>0.917</td>
<td>Left/Straight</td>
<td>0.019</td>
</tr>
<tr>
<td>Stop 3</td>
<td>30 kph</td>
<td>0.812</td>
<td>Left/Straight</td>
<td>0.078</td>
</tr>
<tr>
<td>Stop 4</td>
<td>60 kph</td>
<td>0.901</td>
<td>80 kph</td>
<td>0.025</td>
</tr>
</tbody>
</table>

different for an 80 kph sign than a yield sign. An intersection with vehicles traveling passing through at 80 kph instead of stopping would quickly become catastrophic. The results of this attack are summarized in Table 7.2, Table 7.3, and Table 7.4.

Figure 7.7: Top: Clean, Stop 1, Stop 2; Bottom: Stop 3, Stop 4

Finally, Figure 7.8 shows the images generated by Eykholt et al. [5], Povolny et al. [41], and this thesis. These images indicate that all researchers are finding the same universal weaknesses in stop signs, which suggests the sliding occlusion attack would in fact succeed against the other models used by Eykholt et al. and Povolny et al.
Table 7.3: Performance of adversarial stop signs against a ResNet-50 model with some adversarial training

<table>
<thead>
<tr>
<th>Class</th>
<th>Prob.</th>
<th>Class</th>
<th>Prob.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clean Stop</td>
<td>0.759</td>
<td>Misc. (42)</td>
<td>0.104</td>
</tr>
<tr>
<td>Stop 1 Stop</td>
<td>1.0</td>
<td>Misc. (42)</td>
<td>5.95e-06</td>
</tr>
<tr>
<td>Stop 2 50 kph</td>
<td>0.445</td>
<td>Rd. Abt.</td>
<td>0.056</td>
</tr>
<tr>
<td>Stop 3 50 kph</td>
<td>0.800</td>
<td>20 kph</td>
<td>0.033</td>
</tr>
<tr>
<td>Stop 4 Keep Right</td>
<td>1.0</td>
<td>Left/Straight</td>
<td>2.61e-10</td>
</tr>
</tbody>
</table>

Table 7.4: Performance of adversarial stop signs against a MobileNet model

<table>
<thead>
<tr>
<th>Class</th>
<th>Prob.</th>
<th>Class</th>
<th>Prob.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clean Stop</td>
<td>0.973</td>
<td>Yield</td>
<td>0.024</td>
</tr>
<tr>
<td>Stop 1 Stop</td>
<td>0.798</td>
<td>Yield</td>
<td>0.051</td>
</tr>
<tr>
<td>Stop 2 Stop</td>
<td>0.414</td>
<td>Yield</td>
<td>0.118</td>
</tr>
<tr>
<td>Stop 3 80 kph</td>
<td>0.443</td>
<td>Stop</td>
<td>0.328</td>
</tr>
<tr>
<td>Stop 4 Yield</td>
<td>0.467</td>
<td>Stop</td>
<td>0.345</td>
</tr>
</tbody>
</table>

Figure 7.8: Eykholt et al. [5] (left), Povolny et al. [41] (middle), and image created in this thesis (right)

Sliding occlusion analysis offers a simple way to generate adversarial examples. While these are not targeted attacks that seek to maximize a certain class, they succeed in effectively reducing the confidence in the actual class. In the discussed example of stop signs, locating and obfuscating critical regions can quickly lead to hazardous misclassifications. Furthermore, due to the coarseness of this attack (unlike, FGSM or RP$_2$), this attack is quite robust to transformations through the physical domain. This
is demonstrated by evaluating the designed adversarial stop sign by photographing it and classifying it, which produces across three different models.

Additionally, this attack can be distinguished from the RP\textsubscript{2} presented by Eykholt et al. by a few key differences:

- The attack is greatly generalized, to ensure its success in the physical domain and across models
- Adversarial examples are generated in a single step, meaning the attack:
  - Targets fundamental weaknesses that exist across models
  - Is physically robust
  - Is simple to implement and execute
- The generated adversarial images are easy to manufacture for use in the physical world

7.4 Conclusions

The chapter introduced and verified the sliding occlusion analysis first introduced by Zeiler et al. More importantly, this chapter introduces a novel attack that utilizes this sliding occlusion analysis. This attack, which is easy to implement and execute, is physically robust across models and the physical domain, due in part to its single-step nature, but also by targeting large features in the target object that are universal across classifiers.
Autonomous vehicles owe much of their recent success to the advancements made in computer vision. The deep learning models used to visually perceive the surroundings are incredibly powerful, often meeting or exceeding human performance. Unfortunately, the existence of adversarial examples poses a significant threat to these models and their efficacy.

These adversarial examples can effectively fool convolutional neural networks into confidently making incorrect predictions, similar to how humans confidently misinterpret optical illusions; however, while humans may find optical illusions amusing to observe, these adversarial examples can result in catastrophic failures in autonomous vehicles. Therefore, it is important to research and understand these attacks in an effort to better defend against them.

Early work in this field found adversarial examples that existed solely in the digital domain; however, further work revealed that these attacks have the ability to transcend models and leap into the physical world. This increases the severity of this threat, as adversaries now have the ability to remotely attack models despite never having access to the target system.

8.1 Contributions

This thesis explores adversarial attacks against computer vision models, particularly as they pertain to autonomous vehicles. Following the discovery by Szegedy et al.
that hardly perceptible perturbations could produce misclassifications in convolutional neural networks, researchers have explored methods of generating adversarial examples including methods of making these attacks physically robust. The contributions of this thesis are as follows:

- An in-depth study of the Fast Gradient Sign Method first proposed by Goodfellow et al. [4]. This study explores the linearity of the Iterative Fast Gradient Sign Method proposed by Kurakin et al. [23], the robustness of the Fast Gradient Sign Method, and properties of the generated perturbations.

- A discussion of the limitations of the Robust Physical Perturbations (RP$_2$) attack pipeline developed by Eykholt et al. [5].

- Using occlusion analysis, introduced by Zeiler et al. [6], a novel attack is proposed

### 8.2 Discussions & Future Work

The contributions of this thesis are significant; however, there are still many interesting questions that must be answered. The first recommendation for future work is a conclusive verification of Eykholt et al.’s RP$_2$ attack pipeline. Despite its success for Eykholt et al., it seems unlikely that this attack performs well across models due to its multi-step approach. Additionally, the attack seems to target models with lower confidence, which means these models are more easily confused and attacked.

To better understand the work contained in this thesis, researchers should explore the inner-workings of the model during these attacks. How are the activations different when responding to adversarial examples featuring occlusions compared to clean images? Is it possible to replace the final softmax layer with a different activation
function that will reveal when a model is confused and unable to make a confident decision? Unfortunately, the softmax activation function forces a decision by normalizing the output, which hides the fact the model is confused.

Building on the work presented in this thesis, another opportunity for future work is the development of a sliding occlusion attack that is targeted. The novel attack proposed in this paper is untargeted and carries well across models and into the physical domain; however, it is limited in its ability for an adversary to control the model’s prediction. If this work is conducted, it will be important to remember that universal robustness is a key feature of this attack, and that this feature must be maintained in the targeted attack.

It is also unclear how well this work extends to object detection models. Researchers should explore the effectiveness of this attack against object detection models to determine whether this attack is effective against only classification models or whether these attacks extend to detection models.

Additionally, researchers should explore the effectiveness of this attack against non-mappable objects. As traffic signs are stationary, they can be easily mapped; however, non-mappable objects such as pedestrians might be susceptible to this attack.

Finally, occlusion analysis should be used to train better models. The occlusion analysis featured in Chapter 7 indicates that models focus on limited key features. In the case of stop signs, these features seem to poorly represent the target class. This is important not only to defend against this novel attack, but to defend better against adversarial examples. After all, as researchers have already suggested, the best defense against adversarial examples will not be a particular defense mechanism, but rather a more robust model that better interprets the target distributions.
BIBLIOGRAPHY


[32] François Chollet et al. Keras. https://keras.io, 2015.


Accessorize to a crime: Real and stealthy attacks on state-of-the-art face 
recognition. In Proceedings of the 2016 acm sigsac conference on computer 

[40] Andrew T Duchowski. Eye tracking methodology. Theory and practice, 

[41] Steve Povolny and Shivangee Trivedi. Model Hacking ADAS to Pave Safer 
Roads for Autonomous Vehicles. mcafee.com/blogs/mcafee-labs, February 
2020.