NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

IMAGE PERTURBATION GENERATION: EXPLORING NEW WAYS FOR ADVERSARIES TO INTERRUPT NEURAL NETWORK IMAGE CLASSIFIERS

by

Mitchell R. Graves

June 2020

Thesis Advisor: Robert L. Bassett
Second Reader: Robert A. Koyak

Approved for public release. Distribution is unlimited.

THIS PAGE INTENTIONALLY LEFT BLANK

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

1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** June 2020	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE IMAGE PERTURBATION GENERATION: EXPLORING NEW WAYS FOR ADVERSARIES TO INTERRUPT NEURAL NETWORK IMAGE CLASSIFIERS	**5. FUNDING NUMBERS**
6. AUTHOR(S) Mitchell R. Graves	
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)

Modern-day machine learning algorithms are vulnerable to adversarial perturbations. Before machine learning is employed in operational environments it is important for the Department of Defense to understand the vulnerabilities in adversarial domains. With an emphasis on applications to image classification, this thesis intends to investigate methods for constructing adversarial manipulations intended to fool a statistical classifier in order to establish best practices prior to employment in a mission-critical environment. The state-of-the-art method for generating adversarial perturbations utilizes an easy to obtain linear approximation of the defender's loss function and applies change to each pixel's value in the direction of the gradient by a pre-determined constant in order to maximize the defender's loss function. This results in misclassification of the image. This thesis aims to improve existing methods for developing adversarial inputs to image classifiers by attempting to find new ways to make perturbations less perceptible to human vision. We develop two new perturbation methods. Color Aware attempts to constrain perturbation changes to match human perception by mapping to CIELAB color space, a color space better suited to represent human vision, while Color & Edge Aware constrains perturbation changes in visually smooth regions in images.

14. SUBJECT TERMS adversarial, perturbations, image classifier, neural network	**15. NUMBER OF PAGES** 99
	16. PRICE CODE

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

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

IMAGE PERTURBATION GENERATION: EXPLORING NEW WAYS FOR ADVERSARIES TO INTERRUPT NEURAL NETWORK IMAGE CLASSIFIERS

Mitchell R. Graves
Captain, United States Marine Corps
BS, U.S. Naval Academy, 2013
MSE, University of Pennsylvania, 2014

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
June 2020

Approved by: Robert L. Bassett
Advisor

Robert A. Koyak
Second Reader

W. Matthew Carlyle
Chair, Department of Operations Research

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Modern-day machine learning algorithms are vulnerable to adversarial perturbations. Before machine learning is employed in operational environments it is important for the Department of Defense to understand the vulnerabilities in adversarial domains. With an emphasis on applications to image classification, this thesis intends to investigate methods for constructing adversarial manipulations intended to fool a statistical classifier in order to establish best practices prior to employment in a mission-critical environment. The state-of-the-art method for generating adversarial perturbations utilizes an easy to obtain linear approximation of the defender's loss function and applies change to each pixel's value in the direction of the gradient by a pre-determined constant in order to maximize the defender's loss function. This results in misclassification of the image. This thesis aims to improve existing methods for developing adversarial inputs to image classifiers by attempting to find new ways to make perturbations less perceptible to human vision. We develop two new perturbation methods. Color Aware attempts to constrain perturbation changes to match human perception by mapping to CIELAB color space, a color space better suited to represent human vision, while Color & Edge Aware constrains perturbation changes in visually smooth regions in images.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

THIS PAGE INTENTIONALLY LEFT BLANK

List of Figures

THIS PAGE INTENTIONALLY LEFT BLANK

List of Acronyms and Abbreviations

AI	Artificial Intelligence
CA	Color Aware
C& EA	Color & Edge Aware
CIE	Commission Internationale de l'Eclairage (International Commission on Illumination)
CIELAB	Commission Internationale de l'Eclairage L*a*b* colorspace
CNN	Convolutional Neural Network
DoD	Department of Defense
FGSM	Fast Gradient Sign Method
I-FGSM	Iterative-Fast Gradient Sign Method
L-BFGS	Limited-memory-Broyden-Fletcher-Goldfarb-Shanno
LED	Light Emitting Diode
MNIST	Modified National Institute of Standards and Technology
RGB	Red-Green-Blue

THIS PAGE INTENTIONALLY LEFT BLANK

Executive Summary

The proliferation of image sensors across modern battlefields leave the Department of Defense in situations where it cannot analyze all the data it collects. With a desired endstate of not missing any key intelligence, the Department of Defense is looking at opportunities to automate certain intelligence-gathering processes in order to sift through the vast amounts of collected information. Machine learning and artificial intelligence algorithms can process data in order to identify classes of objects seen in images and video feeds. Implementing a system like this may save many man hours, freeing analysts to focus on pressing issues.

Neural networks are the most popular modern-day image classifier and is the application that would be utilized in such a scenario. However, an issue recently discovered is that these networks are susceptible to adversarial perturbations, which are small changes to the image pixels, often imperceptible to the human eye, that cause the neural network to misclassify the image. Since these neural networks can be so easily fooled, the Department of Defense should be aware of the vulnerabilities when it utilizes these systems in a contested domain. This thesis takes a "Red Cell" perspective and explores new ways that hostile actors could break neural networks.

We have developed two new perturbation generation methods, Color Aware and Color & Edge Aware, that generate perturbations that are as disruptive as previous methods but are more imperceptible to human vision. Digital image data is stored in three-dimensional arrays. The depth dimension of one of these arrays is referred to as the "color channels", and humans perceive the combination of these values when we view images. The color channels are referred to the red, green, and blue (RGB) color planes because images are stored in RGB colorspace. Adversarial perturbations use the image's pixel data to influence misclassification in neural networks. The issue is that perturbations in RGB colorspace do not respect human vision. This means that a unit change between two colors in one part of the RGB colorspace may be more perceptible to human vision than that same change in another part of RGB colorspace. In order to ensure that perturbations remain consistent with human perception we convert image data from RGB colorspace to CIELAB colorspace. We call this perturbation generation process Color Aware. CIELAB colorspace is designed so that the Euclidean distance between any two colors will match human perception. By

developing the perturbation in CIELAB space, we ensure that pixel changes introduced to the original image will match human perception. Color Aware breaks neural networks like previous perturbation techniques but is more difficult for humans to detect.

Color & Edge Aware builds on Color Aware. Researchers have determined that perturbation artefacts are easily seen by humans in visually smooth, homogeneous regions of images. Color & Edge Aware utilizes edge detection filters to develop weights for areas of the images that contain edges. We use these weights in conjunction with our Color Aware algorithm, which limits change in visually smooth regions of images, and therefore make the perturbations even less perceptible to humans.

CHAPTER 1:
Introduction

If we continue to develop our technology without wisdom or prudence, our servant may prove to be our executioner.

- Omar Bradley (General, US Army)

This thesis aims to improve existing methods for developing adversarial inputs to image classifiers by attempting to find new ways to make perturbations less perceptible to human vision. Specifically, we aim at adding two constraints, the first attempts to constrain perturbation changes in CIELAB color space, which is endowed with a metric that better represents human visual perception, while the second limits perturbation changes to areas of an image that contain edges.

1.1 Motivation

Modern day machine learning algorithms are vulnerable to adversarial perturbations. Before the Department of Defense (DoD) can implement these new algorithms, it must understand the repercussions of employing them in a contested space. With an emphasis on applications to image classification, this thesis investigates methods for constructing adversarial manipulations intended to fool a statistical classifier. By serving as the"Red Cell," the purpose is to manipulate modern methods of image classification in order to identify their weaknesses. Knowledge of these limitations can then be used to establish best practices prior to employment of this technology in a mission-critical environment.

1.2 Military Relevance

Modern countries are in an artificial intelligence (AI) arms race. Russia recently announced that it is developing a missile that will make its own decisions, such as switching targets mid-flight (O'Connor 2017) . China's president, Xi Jinping, and the rest of their government leadership believe that being at the forefront in AI technology is crucial to the future of economic power competition and global military influence (Allen 2019). The United States

is also a participant in this AI arms race. As outlined in the National Defense Strategy, the "Department of Defense will invest broadly in the military application of autonomy, artificial intelligence, and machine learning" in order to develop advanced autonomous systems (Mattis 2018). Project Maven – made infamous because Google employees petitioned to abandon their work—was a project, paid for by the Pentagon, that utilized machine learning and artificial intelligence algorithms to identify different classes of objects in drone video (*Global News* 2018; Fryer-Biggs 2018; Pellerin 2017). While it stops short of weapons deployment, the program does allow users to gather information in real-time about objects in the video feeds, allowing the US Armed Forces to conduct reconnaissance and command and control more effectively and without human involvement. While the project does face political and social issues, it must be highlighted that the underlying systems that Project Maven are based on are not fully understood. Though the details of Project Maven are classified, there may be scenarios in which enemy actors are able to gain access to portions of or all of the data or algorithm used to accomplish its tasks. Before machine learning and related technologies are employed in operational environments it is important for the DoD to understand its vulnerabilities in contested domains. For machine learning, this means understanding the liability of the system when an adversary possesses technical insight on the workings of the algorithm.

The most advanced image classification algorithms have a success rate of nearly 97% for some datasets (human classification rate is only 90-95%) (Szegedy et al. 2016). With computers outperforming the average human, there are numerous applications of this technology within the Department of Defense. From automating parts of the intelligence process to reinforcing remote sentry posts, the return on investment can be high. However, military leaders and government acquisition officers should be wary of implementing a system like these before understanding the technologies' limitations. One such limitation is that adversarial perturbations to an image can change the classification of the output result, and in some cases, not distort the image enough for the human eye to notice (Goodfellow et al. 2014; Szegedy et al. 2013). An adversarial perturbation is a small change to an image which results in the image being misclassified by a given machine learning algorithm. If adversaries obtain friendly classifier models, it is possible to cause catastrophic failure with slight alterations to each image. This thesis introduces new constraints based on edges and image color channels in order to understand the limitations of modern techniques for generating

perturbations. Though current techniques for generating adversarial perturbations enjoy some success, many fail to accurately capture how a human eye detects differences between images. By experimenting with constraints based on edge-detection and perceived color distance, we attempt to cause misclassification using only perturbations which a human eye would not notice.

1.3 Technical Background

1.3.1 Neural Network Architecture

Project Maven's ability to classify objects from drone video feeds is a very advanced version of statistical classification. Statistical classification is the arrangement of various inputs into different categories based on their features. A classification system is trained with labeled data points and then utilized to determine the label that best matches new data. A common use for this concept is the application to digital image classification. To train an image classifier, many images are used to identify the relationship between an image's pixel values and its label (Papernot et al. 2017). For image classification, the label is most often the object depicted in the image. A modern day approach to statistical classification is the use of an artificial neural network. Originally inspired by the way a human brain operates, artificial neural networks are a series of algorithms that endeavor to recognize the underlying relationships between a sample's inputs and its label (Goodfellow et al. 2016). A convolutional neural network (CNN) is a specialized kind of neural network that is advantageous for image processing because it is used to process grid-like data (Krizhevsky et al. 2012). In image data storage, images are structured by storing their pixel values in a fixed format, e.g. height, width, and color channel. This storage technique creates a grid-like structure that is beneficial to convolutional techniques. Besides the input and the output layers, neural networks can contain hidden layers. The complexity and number of these layers is decided by the network's designer, often through trial and error, experience, end state objective, and accepted best practices. Research has shown that convolutional neural networks consistently outperform fully connected neural networks for image classification tasks and it is now a practiced norm for image processing (Nebauer 1998).

Other required elements of neural networks, not specific to CNNs, are activation functions. Activation functions are often placed after hidden layer outputs. These activation functions

promote non-linearity in a neural network. Without these activation functions, the network is just a large, complicated linear model. The combination of multiple hidden layers and multiple activation functions cause neural networks to produce output layers that are highly nonlinear relationships of their input layers (Szegedy et al. 2013). Because of the nonlinearity of the output, many were surprised to find that several state-of-the-art neural networks are vulnerable to adversarial perturbations (Szegedy et al. 2013). Researchers also found that one perturbation could cause a neural network to misclassify multiple classes. This suggests that these adversarial perturbations expose foundational blind spots in the neural network training algorithms (Goodfellow et al. 2014). While neural networks are nonlinear functions, research has shown that linear approximations are sufficient to create adversarial perturbations. See Figure 1.1 for an example of an adversarial image. Initially, this was unexpected because, as explained, the output is a nonlinear function of the input data; however, Goodfellow et al. discovered that the design of easy to train and optimize models results in a neural network which empirically has an accurate local linear approximation for its loss function (Goodfellow et al. 2014). This accurate linear approximation is where adversaries can make minute changes to the input (pixel values) that result in misclassification.

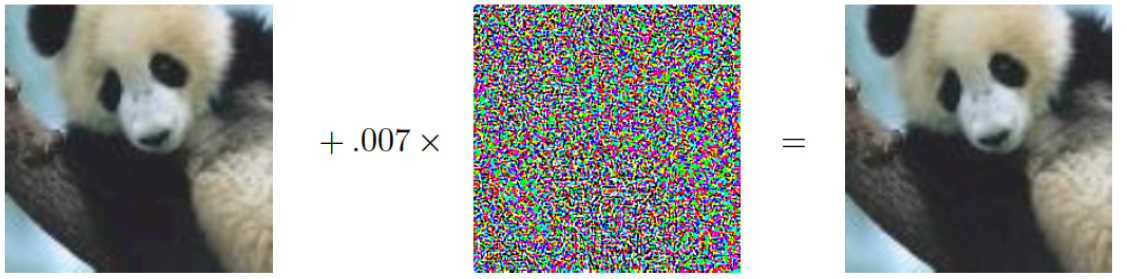

Figure 1.1. Example of a perturbation. Left: Original image classified as a panda. Middle: Small perturbation added to the original image, which when ran through an image classifier identified this noise as a "nematode" (which is a type of worm). Right: Image that looks like a panda for humans; yet the classifier has deemed that it is a gibbon. Adapted from Goodfellow et al. (2014).

1.3.2 Perturbation Traits

The framework for constructing adversarial perturbations starts with a battle between a defender and an adversary – one where the former wants to minimize a loss by choosing good parameters for classification while the latter wants to maximize this loss through perturbing the input data. Mathematically, this engagement takes place over the loss function. The adversary's goal is to create a small perturbation which maximizes the loss function and results in a misclassification. Since the discovery of adversarial perturbations for image classifiers, a considerable amount of research has been conducted in this area; however, most methods for generating adversarial perturbations can be described by a few key traits. These divisions are whether the attacks are targeted or untargeted and whether the technique to generate the attacks is one-step or iterative. A targeted attack attempts to have the perturbed image classified as a specific incorrect class, whereas an untargeted attack attempts to cause a misclassification without regard to a specific output class. Perturbation generation algorithms can create targeted or untargeted attacks, dependent on certain inputs. In untargeted attacks, the algorithm tries to shift the perturbed image's loss away from the correct label, while targeted attacks decrease loss for a desired label. The other significant trait that characterizes how a perturbation is generated is that they can either be generated through a one-step or iterative process.

These complex neural networks possess linear approximations that attackers attempt to estimate and alter in a very slight, precise way to create the perturbation. These approximations are calculated by computing the gradient of model's loss. The attackers then shift the image based on the gradient information. If this series of steps is only conducted one time, then the generation technique can be described as one-step. Many algorithms conduct this series multiple times and become iterative algorithms. One-step methods are quick and cheap computationally but often the perturbations generated are less refined. Conversely, iterative methods take longer to generate and use more memory, but these generated perturbations often create higher misclassification rates with higher confidence. In this thesis, we improve current perturbation techniques by modifying them to properly account for colors and edges. Our contributions apply to both targeted and untargeted attacks.

1.3.3 Choosing Our Constraints

Current efforts at generating new adversarial perturbations aim to cause misclassification while remaining imperceptible to the human eye (Szegedy et al. 2013; Carlini and Wagner 2017). Larger perturbations tend to improve adversarial strength in classifier networks, meaning that the confidence behind the misclassification is higher. When creating a perturbation scheme, the formulation is a compromise between two goals. If the formulation allows too much change, the misclassification has a high confidence; however, the perturbation is at risk of showing visible change, and vice versa. Treating the image as a vector containing its pixel values, using different ℓ_P-norms allows for a variety of different constraints on the magnitude of the perturbations. In our work, ℓ_p-norms measure the difference between perturbed images and the original input image and we introduce their use more thoroughly in Chapter 3. However, perturbations are normally on or close to this decision boundary (Croce and Hein 2019). The constraints are often made in the ℓ_0, ℓ_2, or ℓ_∞ space. Methods for generating perturbations typically assume that a perturbation with small ℓ_p-norm results in a perturbed image which is indistinguishable from the original by the human eye . However recent studies have questioned using ℓ_p-norms in Red-Green-Blue (RGB) colorspace as a measure of imperceptibility (Zhao et al. 2019).

Color images are stored in a computer as multidimensional arrays, in which there is a height, width, and color dimension. The color dimension is often stored in RGB colorspace, meaning that there are three color planes that contain, red, green, and blue values between 0-255. The RGB color space is an additive color set, meaning that with the addition of red, green, or blue colorants, any color can be created (Sharma 2018). When visualizing a digital color image, the color planes are stacked on top of each other and it is the combination of the three values in these planes that a human perceives. Figure 1.2 highlights, in 3-dimensional space, how the color planes interact.

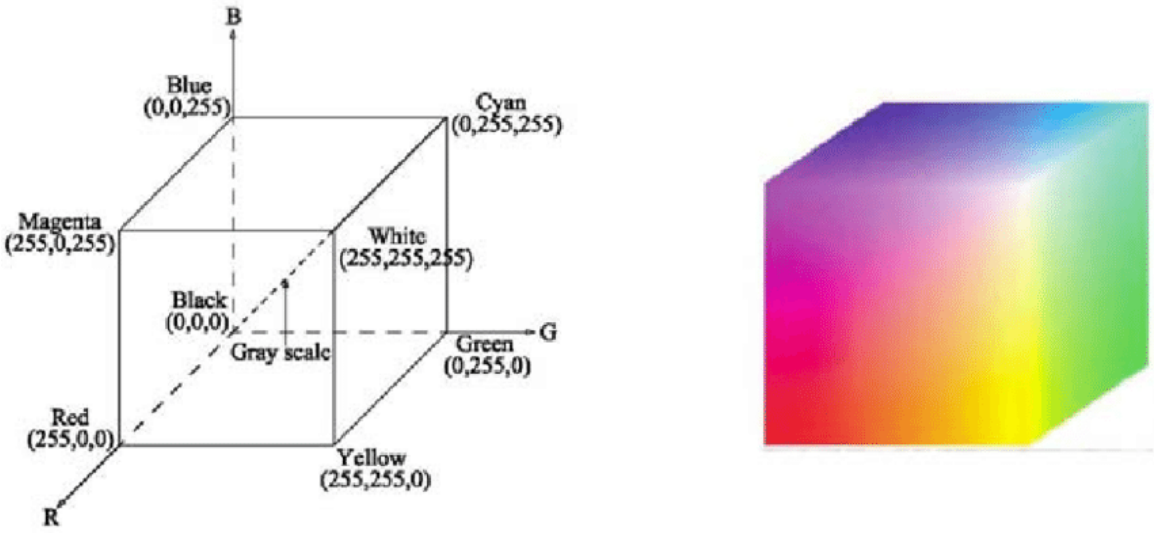

Figure 1.2. Visualization of RGB colorspace. Adapted from Martín and Santos (2011)

When generating perturbations using RGB colorspace, the ℓ_p-norms are calculated using the coordinate system seen on the left-hand side of Figure 1.2. As described earlier, humans perceive color from RGB space by the additive values in the red, green, and blue planes; however, the most popular perturbation techniques do not acknowledge that multiple entries in the image array correspond to the color at a single pixel. Many perturbations change pixel values based on the gradient of the loss with respect to the image array at each pixel value. Using the ℓ_∞ norm as an example, a pixel's red, green, and blue values can change independently up to a boundary, meaning the change from the original color to the perturbed color at a given pixel through all three color planes is a combination of the perturbations in each color plane. However, research has shown Euclidean distance in RGB space does not accurately measure perceived color distance by the human eye (Schanda 2007; Zhao et al. 2019). Thus, while certain perturbations are optimal in their respectively constrained ℓ_p-norm in RGB colorspace, the perturbations may not be optimal when preventing human detection. Figure 1.3 is a good example how color distances and human perception do not align in RGB colorspace. This misalignment motivates exploring perturbations in other colorspaces.

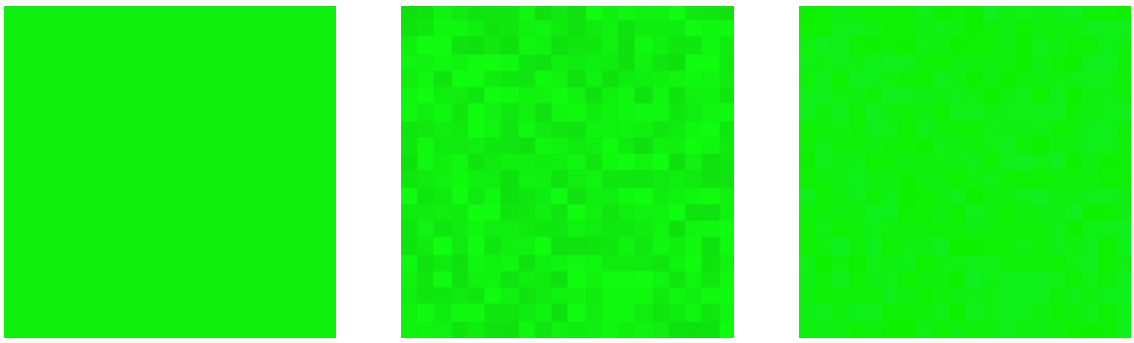

Figure 1.3. Example of how RGB colorspace does not respect human perceptions. Left: Original image (a 20×20 8-bit RGB image patch with color (15,240,15)). Middle: Image perturbed by adding noise in the G-channel, sampled from a uniform distribution in the range [-15,15]. This noise is perceptible by human vision. Right: Image perturbed by adding the identical noise, but in the B-channel. The B-channel perturbations are imperceptible (best viewed on screen). Adapted from Zhao et al. (2019).

Another issue with the RGB colorspace is that it is device-dependent. RGB is only a set of instructions for a device, and does not provide a reliable or precise specification of color (Sharma 2018). While the final exact color will look similar on many systems, there are actually many versions of RGB that depend on its device and configuration. A "device-independent" color system can be used to communicate information across different devices. For example, data that is stored in an independent color system and used in a RGB system gets translated locally. This allows data stored in an independent color system to be viewed the same on multiple systems, even if those systems have different RGB instructions. Mathematically, this idea is very important for our conversion from RGB colorspace because the algorithm now depends on which device images are captured. However, the conversion itself is less important than the conceptual understanding that RGB colorspace does not match human perception.

The structure of the human eye makes its perception of color different from a digital RGB representation. Rods and cones in our eyes enable humans to discern colors when different amounts of light hit an object. The type of light also affects the way humans perceive color; a yellow incandescent bulb will make a white piece of paper look more yellow than a modern day LED bulb. While RGB space could capture and project these color

differences, RGB space is not a representation of how color is created in the natural world; it is simply a communication of color through a digital medium. Tristimulus data is defined as the three basic dimensions, or colorimetric properties, of color characterization. These three dimensions, hue, saturation/chroma, and lightness, are perceptual attributes of color that allows the brain to interpret and conceptualize the signals it receives through visual stimuli (Schanda 2007; Sharma 2018). Color, a very subjective phenomenon, can differ between individuals, so the Commission Internationale de l'Eclairage (CIE) or in English – International Commission on Illumination – has developed methods to measure color and specifies today's color measurement systems (Sharma 2018; Schanda 2007). "CIE-based color systems are well accepted, visually-relevant, standardized, instrument-based, numerical methods, for color specification and today, form the basis for managing and monitoring all aspects of color management" (Sharma 2018). An important reason is that we are interested in CIE is that one of their colorspaces is designed so that colors are equally spaced to match human perception.

In 1905, Albert Munsell developed the Munsell Color System as a teaching aid to his art students. The premise around his color system was that each color was equally spaced with respect to human perception (Sharma 2018). Based off this color system, CIE developed a color system in 1931, and then adjusted again in 1976. CIE 1976 L*a*b* system, pronounced "l-star, a-star, b-star" is represented by a three-dimensional color diagram, seen in Figure 1.4. The L*a*b* system (also annotated as CIELAB and pronounced "see-lab") separates the color information into lightness (L*) and color information on a green/red (a*) and blue/yellow (b*) axis. The lightness of a color changes as we move vertically through this sphere, with L* of 0 representing black and L* of 100 specifying white (Sharma 2018).

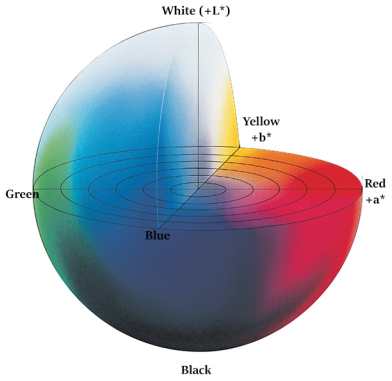

Figure 1.4. Visualization of L*a*b* colorspace. Source: Konica Minolta (2018)

Knowing that RGB space is not perceptually uniform to the human eye, and that L*a*b* colorspace is, we propose to create perturbations in this new colorspace. Our goal is to try to increase or maintain the confidence of misclassification while remaining imperceptible. To the best of our knowledge, this is the first perturbation generation technique that translates from RGB to CIELAB and then implements a ℓ_2 norm constraint across color channels. This ℓ_2 norm constraint in CIELAB space would limit changes in pixel values based on human perception. After the perturbation is generated and translated back to RGB colorspace, the perturbation should be less perceptible.. The ℓ_2 norm in CIELAB space is also known as $\Delta E_{a^*b^*}$. The terminology for "E" is derived from the German word "empfindung", which means a difference in sensation (Sharma 2018). The subscript after the ΔE annotates which version of measurement is being used. Zhao et al. generated a technique that constrains perturbations measuring the ΔE_{00}, also annotated as the CIEDE2000 color distance measurement (Zhao et al. 2019). This is CIE's newest ΔE measurement which aligns better to human perception than earlier versions of ΔE. In our work, we have decided not to fully implement the CIEDE2000 color distance measurement. Zhao et al. translate their RGB space into CIELAB to calculate their ΔE_{00} distance; however, their formulation is difficult to solve, and the proposed solution technique is a heuristic using an adaptive penalty (Zhao et al. 2019). Our algorithm translates the original image to CIELAB space where it remains for the optimization problem. Also, the addition of our ℓ_2-norm constraint across pixel values creates separable constraints where each pixel's perturbation can be computed independently, once a gradient has been calculated. This results in quick computation times.

To the best of our knowledge, Zhao et al. are the only authors to include perceptual distance in their work and acknowledge that ℓ_p-norms can be measured in CIELAB space. However, in order to reflect known human perception as close as possible, they chose to utilize the ΔE_{00} distance metric. Even though our uses of CIELAB space are slightly different, our second constraint was directly motivated by their recommended follow-on work.

Neither, traditional RGB nor Zhao et al.'s ΔE_{00} perturbations, in terms of human perception, perform well in visually smooth regions with low saturation, such as the background of the apples on the left of Figure 1.5. They deem important future work "to develop techniques that make perturbations imperceptible, or unnecessary, in those regions" (Zhao et al. 2019). As explained in Section 1.3.2, perturbations are often generated using the gradient of a classification-derived loss function to an input image. The right half of Figure 1.5 is a visual representation of the gradient of a classification-derived loss function with respect to the image on the left. While the colors of the image are chaotic, two traits should be noticed: 1) it is very busy and noisy, but change in color represents a change in gradient value, and 2) we can see from the gradient that the direction of maximum increase changes the apples differently than the background of the image. As perturbations are created, they are dependent on this gradient, so Zhao et al. claim that humans will first notice perturbations in visually smooth regions where the gradient of a classification-derived loss function with respect to the input image is noisy (Zhao et al. 2019). Current efforts at perturbation generation fail to constrain change in visually smooth portions of images.

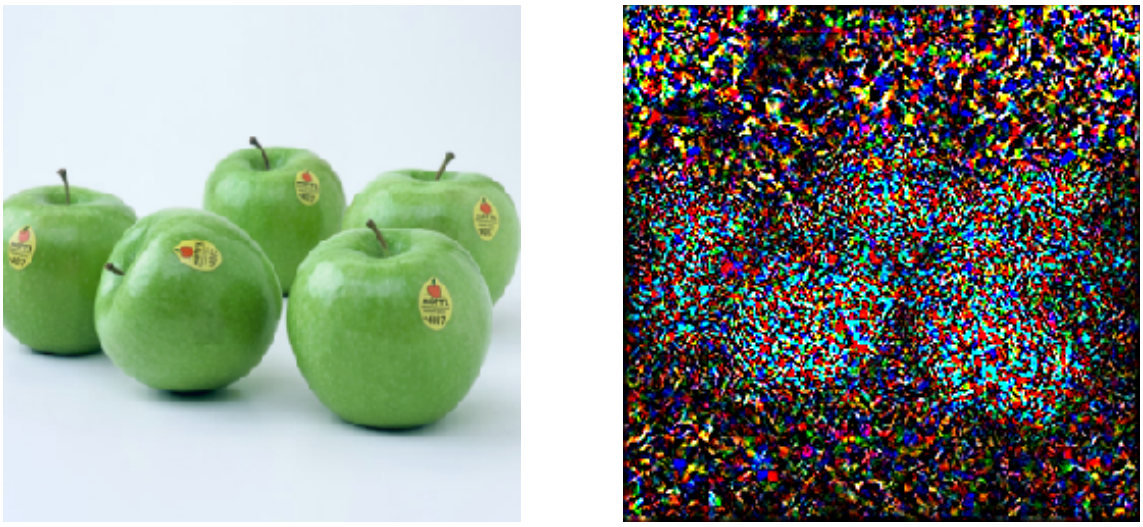

Figure 1.5. Left: Picture of apples. Right: Visualization of the gradient of the model with respect to the image data. Note that the area behind the apples, while it appears smooth to the human eye, contains large magnitudes in the gradient. (Source - ImageNet Database)

Being able to visually detect perturbations in smooth areas of images motivated us to develop a constraint that would only alter pixel values that were not in smooth regions. Utilizing conventional edge detection techniques from computer vision, we can determine if each pixel is on or near an edge and limit changes in the perturbation to these regions. Edge detection's goal is to outline boundaries in a digital image. Most edge detection algorithms work quickly on grayscale images. Instead of having RGB color planes and possessing color information, a grayscale image has one color plane that depicts only brightness information, making it monochromatic (Umbaugh 2010). A grayscale image can be constructed from a RGB image by taking the average of the color values across the RGB planes. Grayscale images are important because edge detection utilizes the difference of brightness in pixel values to determine if there is an edge present. The idea is that an edge can be found by comparing a pixel's brightness with the brightness of neighboring pixels. If two neighboring pixels possess the same brightness level (same shade of gray), then there is not likely an edge. If there is a large jump or fall in brightness from one pixel to the next, then that discontinuity defines an edge (Umbaugh 2010). For edge detection purposes, an ideal edge can be seen on the left hand side of Figure 1.6; however, the right side of the picture shows

how edges realistically appear in images.

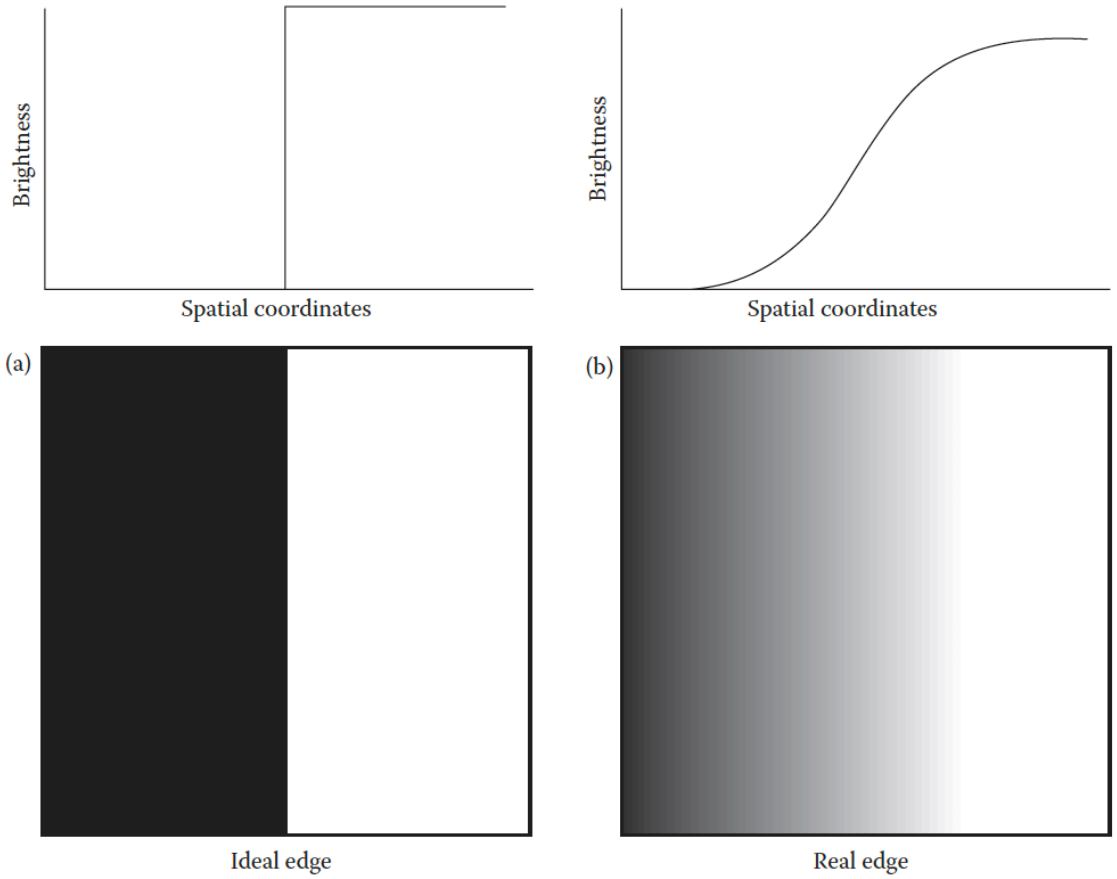

Figure 1.6. (a) Ideal edge versus (b) real edge. Source: Umbaugh (2010)

To detect edges for our constraints, we decided to use the Sobel Filter. The Sobel filter utilizes two kernels that are convolved over the image, horizontally and vertically. These filters can be seen below.

$$\text{VERTICAL EDGE KERNEL} = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

$$\text{HORIZONTAL EDGE KERNEL} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

Two masks are generated from the convolution of the grayscale image with these kernels. For each pixel in the image, these masks produce the values, s_1, which represents vertical edge presence and, s_2, which represents horizontal edge present. For each pixel, the overall edge magnitude can be calculated using the following equation:

$$\text{EDGE MAGNITUDE} = \sqrt{s_1^2 + s_2^2}$$

Figure 1.7 shows a grayscale image before and after the Sobel Filter is applied.

Figure 1.7. Left: Grayscale image of photo in Figure 1.5. Right: Sobel filter applied. (Image source - ImageNet Database)

The benefit from utilizing the Sobel filter is that it also returns edge magnitude. In order to minimize human ability to detect perturbations, we should limit changes to areas with higher edge magnitude. The stronger the edge, the larger perturbation allowed on that pixel. Generally, any off-the-shelf edge detector would be appropriate here, so we are not

necessarily restricted to the Sobel filter. However, edge detectors which return only binary pixel values indicating edges are inappropriate because they do not allow us to scale the magnitude of the perturbation with the edge magnitude. We show that this additional constraint can decrease perceptibility of perturbations by eliminating perturbations in visually smooth regions of images.

Chapter 2 of this thesis shows introductory neural networks and perturbation techniques in order to build toward modern day practices. It also introduces previous methods of image perturbation generation, which is crucial for comparison of our work in Chapter 3.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 2:
Neural Network and Perturbation Generation Literature Review

In the previous chapter, we introduced the concepts of neural networks and image perturbations as well as introduced motivation for our contributions. In this chapter, our goal is to dive deeper into neural network architecture and perturbation generation techniques. The first section will introduce a simple neural network and the Fast Gradient Sign Method (FGSM). The goal is to simplify the neural network and perturbation technique to its most important components to introduce the necessary mathematical concepts as clearly as possible. As the chapter progresses, we will build toward modern day practices. Sections 2.2 and 2.3 will focus on perturbation techniques that have achieved high recognition in this specific field. The goal of introducing these techniques is to build a fundamental understanding of how they work so that our work can be compared to their results. The perturbation techniques in these two sections are the box constrained Limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) algorithm, and the Carlini-Wagner ℓ_2-norm attack (Szegedy et al. 2013; Carlini and Wagner 2017).

2.1 Fast Gradient Sign Method

The Fast Gradient Sign Method, just as its name suggests, is a perturbation technique designed to be quick, cheap, and based on the direction of the gradient of a network's loss function. This perturbation is not necessarily supposed to produce an accurate looking perturbation (Carlini and Wagner 2017). The equation for an untargeted attack with their technique is:

$$x' = x + \epsilon \, \text{sign}(\nabla_x \text{Loss}(x, y; \theta)) \tag{2.1}$$

where x is the input image, x' is the perturbed image, ϵ is a magnitude that bounds change, y is the label of the input image, and θ are the weights from the neural network. Goodfellow et al. suggest that the linear approximation of the loss function makes it easy

to approximate these perturbations (Goodfellow et al. 2014). They declare that simply obtaining the direction of the gradient of the loss and changing the pixel values by some small magnitude, ϵ, that the neural network will not be able to identify the true class. This small shift in pixel values results in the loss shifting away from the true class.

As an adversary, we are trying to maximize problem 2.2. This constrained problem states that in developing a perturbation x' from the image x, we strive to maximize the model's loss subject to the constraint that no pixel is changed by more than the step size ϵ.

$$\begin{aligned} \underset{x'}{\text{maximize}} \quad & \text{Loss}(x', y; \theta) \\ \text{subject to} \quad & \|x - x'\|_\infty \leq \epsilon \end{aligned} \tag{2.2}$$

However, if we alter problem 2.2 by taking a first-order approximation of the objective, we obtain problem 2.3 where $\delta = x - x'$, which is the change the perturbation introduced to x.

$$\begin{aligned} \underset{\delta}{\text{maximize}} \quad & \nabla_x \text{Loss}(x, y; \theta)^T \delta \\ \text{subject to} \quad & \|\delta\|_\infty \leq \epsilon \end{aligned} \tag{2.3}$$

The solution to this problem is given by equation 2.1.

This algorithm is one of the first methods to quickly develop perturbations that consistently cause misclassifications in many models and follow-on research and studies consistently use FGSM as a comparison tool (Carlini and Wagner 2017; Zhao et al. 2019; Croce and Hein 2019; Nguyen et al. 2015; Kurakin et al. 2016; Papernot et al. 2017). Many state of the art perturbation generation methods utilize similar mathematical techniques to FGSM, which allows FGSM to serve an introduction to image perturbations generally. FGSM can be scaled in complexity; from iterative targeted attacks on a multiple class dataset to a one-step attack using binary classification. For this reason, the next section will focus on an image perturbation for a binary classification network. This is one of the most simple forms both neural networks and image perturbation techniques, and it will serve as foundation to introducing more complex perturbations.

2.1.1 The Fast Gradient Sign Method: Binary Classification

In order to exhibit FGSM in its simplest form, we must generate our perturbations on a simple neural network. Utilizing logistic regression in a single layer neural network will allow for binary classification. Binary means that the network is only trying to distinguish between two classes, and all the labels in the dataset are either 1's or 0's, where 1's are in the desired class and 0's are not. In a binary image classifier, each images' pixels are connected with weighted values to the activation function, which in the binary case is the sigmoid function, annotated $\sigma(z)$ where z is some scalar, and defined as:

$$\sigma(z) = \frac{e^z}{e^z + 1} = \frac{1}{1 + e^{-z}}$$

The sigmoid function takes the inputs from the fully connected layer and produces a probability between 0 and 1. In a binary classifier, a probability greater than $\frac{1}{2}$ results in labeling that image as part of the desired class, and a probability below $\frac{1}{2}$ means that the network will label the image as not in the desired class. Figure 2.1 shows an example of a layout for a fully connected, binary classifier neural network. Note that in this image, each $input_i$ represents one value of the input vector. Without the sigmoid function before the output, this figure would represent linear regression.

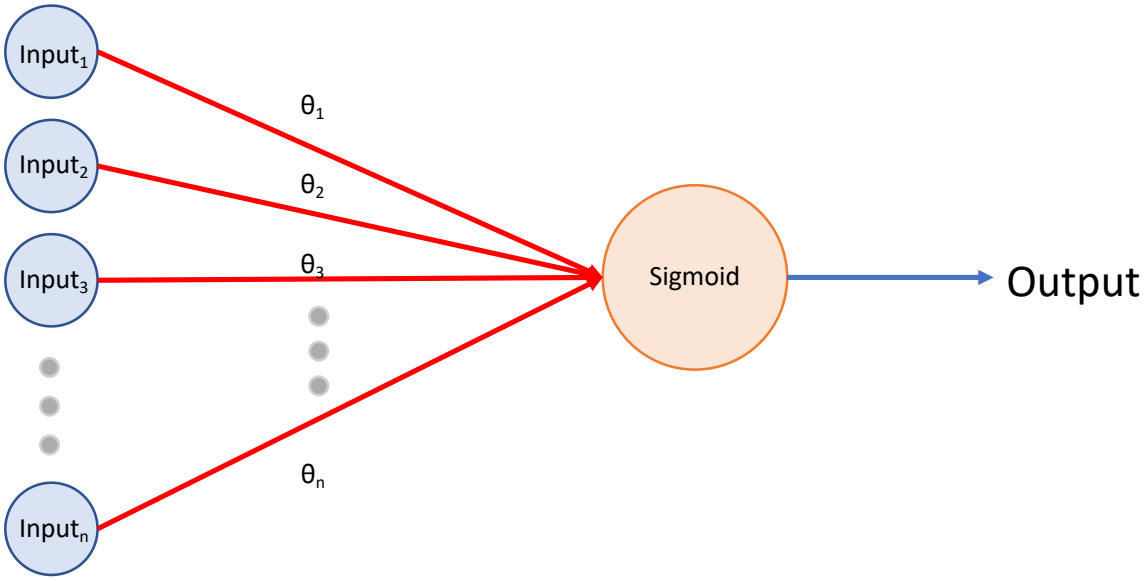

Figure 2.1. Representation of a fully connected neural network for binary classification.

When originally published, the FGSM was shown in a binary case on the Modified National Institute of Standards and Technology (MNIST) database (Goodfellow et al. 2014). The MNIST database has 70,000 images of handwritten numeric digits from 0-9, and is one of the well known data sets in the machine learning and image processing world. Even though the MNIST dataset contains ten classes, Goodfellow et al. utilized binary classification for its simplicity and ease of illustration (Goodfellow et al. 2014). We revisit the original examples provided in the FGSM paper in order to explain the general procedure for creating adversarial perturbations. We use logistic regression that distinguishes images that contain 3's from images that have 7's. Figure 2.2 is a selection of the 3's and 7's from the MNIST dataset that we use to create our two classes.

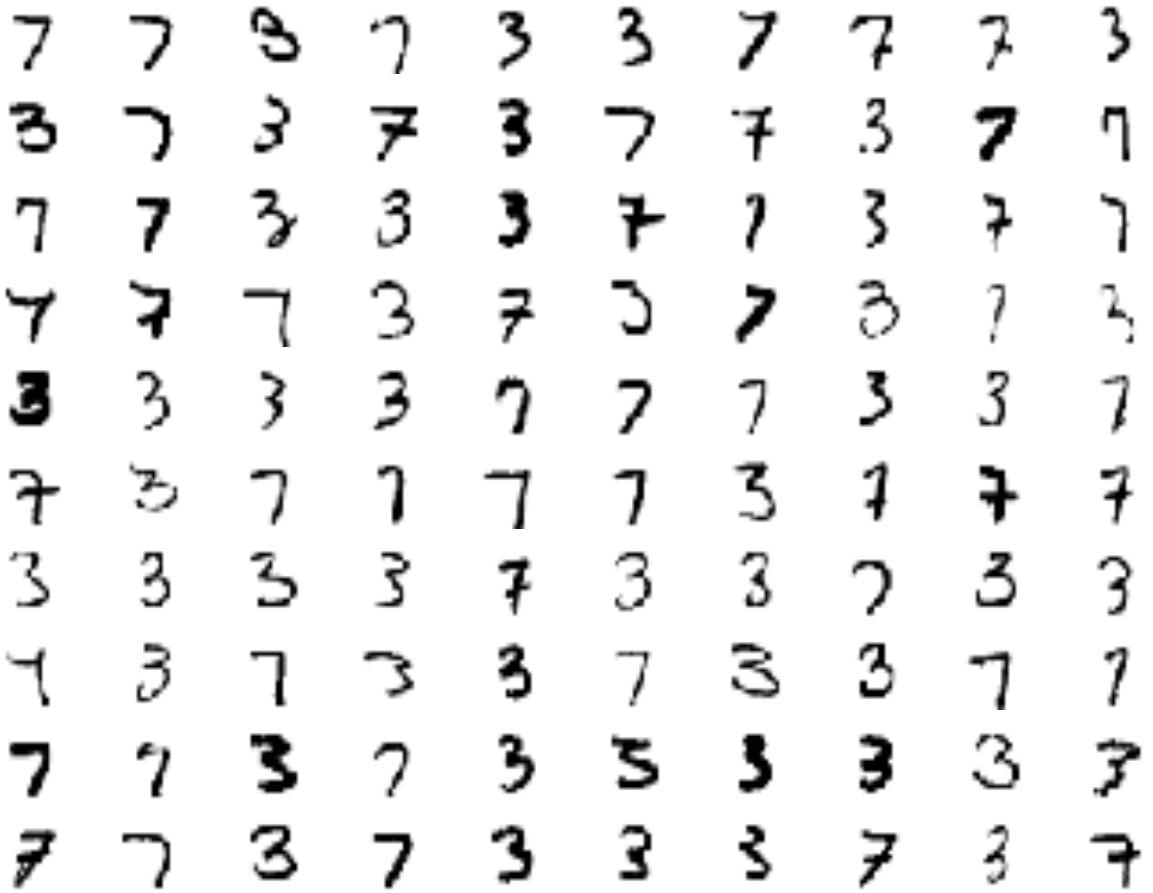

Figure 2.2. Examples of 3's and 7's from the MNIST data.

After training our neural network on the MNIST training set, we obtained a misclassification rate of only 3.09%; while the original paper had a 1.6% error rate (Goodfellow et al. 2014). In our network, 7's are represented by labels of 1 and 3's are represented by labels of 0. An example of what the network receives as an input and what it outputs can be seen in Figure 2.3

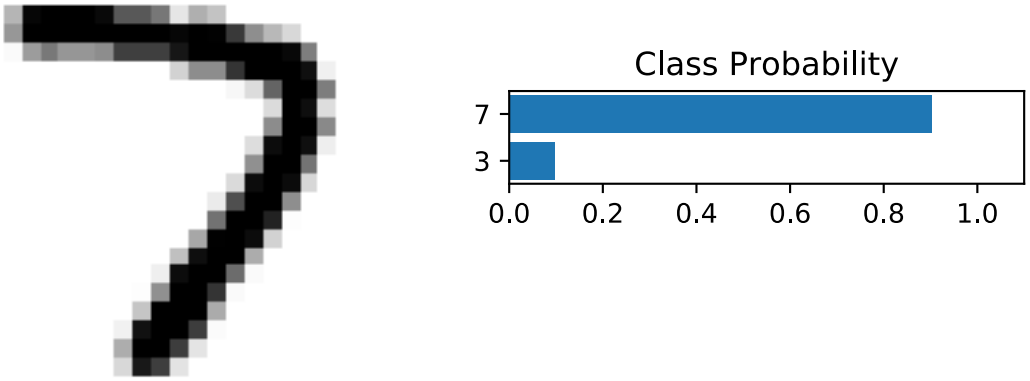

Figure 2.3. Left: Example 7 from the MNIST test set. Right: Output predicted class probabilities. In this example the network got the classification correct.

We assume we have a data sample structured $(x_1, y_1), ..., (x_n, y_n)$ such that $y_i \in \{0, 1\}$, where x_i represents an image, and y_i corresponds to its matching label. If we also have an estimated probability, denoted $\hat{p}_i$, of classifying x_i as $y_i = 1$ then we can determine the loss. Our network's classification is based off of binary cross entropy loss. Denoting binary cross entropy loss as BCE, we recall its definition.

$$\text{BCE}(\hat{\mathbf{p}}, \mathbf{y}) = -\sum_{i=1}^{n} y_i \log(\hat{p}_i) + (1 - y_i)\log(1 - \hat{p}_i)$$

This means for a single sample, i, if the true label is 1, then the loss is $-\log(\hat{p}_i)$ and if the true label is a 0, then the loss is $-\log(1 - \hat{p}_i)$. However, when we take this loss equation, and replace the estimated probability with the output of our sigmoid function, $\sigma(f(x_i))$, where $f(x_i)$ is the output from our logistic regression neural network for the i^{th} image, and then take log-likelihood of the sample (assuming independence across samples), we can derive the loss function for our model. We proceed by taking the log-likelihood of an i.i.d. Bernoulli sample $(f(x_1), y_1), ..., (f(x_n), y_n)$:

$$= \log\left(\prod_{i=1}^{n}\left(\sigma(f(x_i))^{y_i} \cdot (1 - \sigma(f(x_i)))^{1-y_i}\right)\right)$$

$$= \sum_{i=1}^{n} \Big(y_i \log(\sigma(f(x_i))) + (1 - y_i) \log(1 - \sigma(f(x_i))) \Big)$$

$$= \sum_{i=1}^{n} \Big(y_i (f(x_i) - \log(e^{f(x_i)} + 1)) + (1 - y_i) \cdot -(\log(e^{f(x_i)} + 1)) \Big)$$

$$= \sum_{i=1}^{n} y_i f(x_i) - \log(e^{f(x_i)} + 1) \tag{2.4}$$

Equation 2.4 is the log-likelihood function for the logistic regression model. If we simplify this and look at the equation for just one image x_i, and know that for logistic regression $f(x) = \theta^T x$ for $\theta \in \mathbb{R}^n$, the log-likelihood becomes:

$$\mathcal{L}(x_i, y_i; \theta) = y_i \theta^T x_i - \log(e^{\theta^T x_i} + 1), \tag{2.5}$$

where $\mathcal{L}$ is the symbol used for log-likelihood. If we take equation 2.5 and multiply it by -1, we can now minimize the negative log-likelihood instead of maximizing the log-likelihood. While mathematically the two are equivalent, minimizing the negative log-likelihood is referred to as the loss. By adding a negative in front, equation 2.5 becomes equation 2.6, which is the loss equation used in order to get the optimal perturbation.

$$-\mathcal{L}(x_i, y_i; \theta) = -\Big(y_i \theta^T x_i - \log(e^{\theta^T x_i} + 1) \Big) \tag{2.6}$$

Since equation 2.6 is the loss function for the model, instead of denoting it as a negative log-likelihood, $-\mathcal{L}(x_i, y_i; \theta)$, from here we denote it as $\text{Loss}(x_i, y_i; \theta)$. The FGSM leverages first order information from the loss function in order to create cheap and fast perturbations. This method takes the gradient of of equation 2.6 and uses it to construct the perturbation. While moving in the direction of the gradient would cause equation 2.6 to increase, the authors show that moving in the direction of the *sign* of the gradient vector is sufficient to cause effective perturbations. The step size, ϵ, determines the magnitude of change at each pixel. With this step size, the algorithm would then add or subtract from each pixel value

based on the gradient sign. Equation 2.7 shows the gradient of the model's loss with respect to the data.

$$\nabla_{x_i} \text{Loss}(x_i, y_i; \theta) = \left(- y_i + \frac{e^{\theta^T x_i}}{e^{\theta^T x_i} + 1} \right) \theta \tag{2.7}$$

By plugging in this representation of the gradient of the model's loss with respect to the input data back into equation 2.1, the full equation for FGSM becomes:

$$x' = x + \epsilon \, \text{sign}\left(\left(- y_i + \frac{e^{\theta^T x_i}}{e^{\theta^T x_i} + 1} \right) \theta \right)$$

However, unique to binary cross entropy is that the sign of the model weights is also the optimal perturbation. We will derive this fact and show that equation 2.8 holds true.

$$\epsilon \, \text{sign}\left(\left(- y_i + \frac{e^{\theta^T x_i}}{e^{\theta^T x_i} + 1} \right) \theta \right) = \epsilon \, \text{sign}(-y_i \theta) \tag{2.8}$$

Since the magnitude term (ϵ) is a constant on both sides of the equation, this term can be removed, and what we are left with is proving that the sign of θ is equal to the sign of the gradient. Looking at the second portion of the gradient, it is a version of the sigmoid function.

Recall that the sigmoid function has range $(0, 1)$, so the $\exp(\theta^T x_i)/(1 + \exp(\theta^T x_i))$ term can be written as some number $c \in (0, 1)$. This creates a situation where when $y_i = 1$, $-y_i + c$ will always be negative, and when $y_i = 0$, $-y_i + c$ will always be positive. By taking the sign of this expression, it simplifies to the statement that if the image is in the desired class, the gradient moves towards the undesired class, and vice versa. So for a logistic regression network, the FGSM equation for an optimal perturbed image can also be written:

$$x' = x + \epsilon \, \text{sign}(-y_i \theta)$$

The optimal perturbation, i.e. $\pm\text{sign}(\theta)$, for this logistic regression network can be seen in

Figure 2.4.

Figure 2.4. Optimal perturbation from the FGSM for this network

The perturbation in Figure 2.4, when multiplied by some step size ϵ and added to any image causes some level of misclassification, dependent of the magnitude of ϵ. Using the magnitude declared in the original paper, $\epsilon = .25$, Figure 2.5 shows the input from Figure 2.3 with the perturbation added. It also shows the output probabilities from the neural network, note that the network now misclassifies this image.

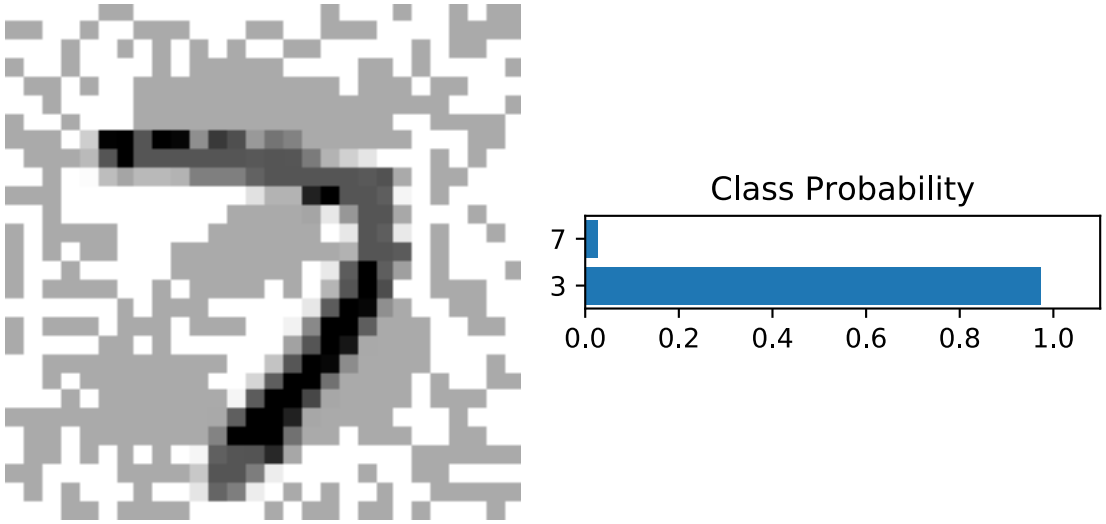

Figure 2.5. Left: Same 7 from Figure 2.3 but with the optimal perturbation developed by the FGSM. Right: Output predicted class probabilities. In this example the network classified the perturbed image with more confidence than the original.

Figure 2.6 recreates Figure 2.2 over multiple images. We keep $\epsilon = .25$. When applied across every image in the MNIST test set, we achieved a misclassification rate of 99.95% where the original paper state they also achieved a misclassification rate of 99.9% (Goodfellow et al. 2014).

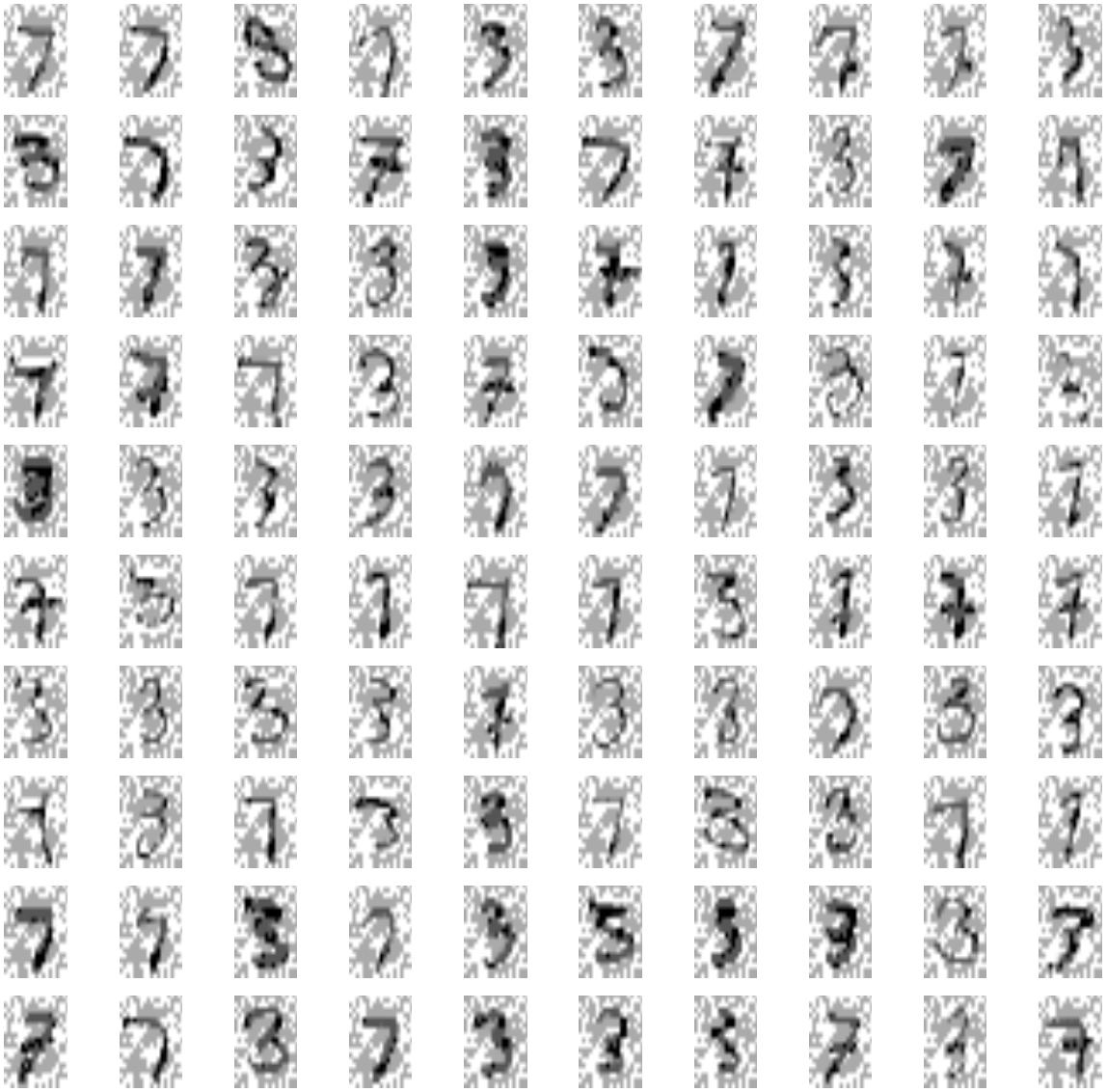

Figure 2.6. Examples of 3's and 7's in the MNIST data with FGSM perturbations.

2.1.2 Fast Gradient Sign Method Multinomial Case

While the binary image classifier is an introduction to learning image perturbations, we will also consider classification problems with more than two categories. The network that we want to attack is a multinomial classification network, meaning that there are more than

two classes to distinguish between. The goal of this section is to introduce the difference between binary and multinomial classifiers and show examples of targeted attacks. We will still be utilizing the MNIST dataset; however, we will be using the full range of values, 0-9, instead of limiting ourselves to 3's and 7's. Figure 2.7 shows a sample of images from the MNIST data set.

Figure 2.7. Example of MNIST data.

Since we will have more than one class in the multinomial setting, the main difference between the binary image classifier and the multinomial image classifier is the underlying loss function. For multinomial classification, we rely on cross entropy loss, annotated as CE and seen in equation 2.9:

$$\text{CE}(\hat{\mathbf{p}}, \mathbf{y}) = -\sum_{i=1}^{n}\sum_{j=1}^{N} y_{i,j}\log(\hat{p}_{i,j}) \tag{2.9}$$

where i is an image, j represents the classes, and N is the total number of classes. In machine learning, when training a multinomial classification neural network, the truth labels are one hot vectors, meaning that each observation y is 1 in a certain index and zero elsewhere. In this case the cross entropy loss can be reduced to $-\log(p_{i,j})$, for the index j where $y_{i,j} = 1$. Ultimately what this means is that the FGSM still works for multiple class problems, but the perturbation can no longer be simply the sign of the weights for the model; it has to be the sign of the gradient of the model's loss. And while the loss function implicitly relies on the weights, θ, from the perspective of the adversary they are fixed values. For notational convenience we omit θ from here on.

Another aspect of the model that requires modification in the multi-category setting is the activation function used on the output of the model. Recall that in binary classification, we utilized a sigmoid activation function at the end of our network. Sigmoid is useful for binary classification because it can easily determine two classes. For a multinomial network, we utilize the softmax as our activation function. The softmax is used on $N - class$ classifier and converts the logits to probabilities. Logits, denoted as a N x 1 vector, $\mathbf{z}$, are the outputs of the neural network prior to the final activation function. For each class in a network, the logit has to be converted to a probability using the softmax function seen in equation 2.10:

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{N} e^{z_j}} \text{ for } i = 1, ..., N \tag{2.10}$$

The softmax output generates the vector of labels $\hat{\mathbf{p}}$ where $0 \leq p_j \leq 1$ and $\sum_{j=1}^{N} p_j = 1$. This activation function is useful in $N - class$ classifiers because the output vector $\hat{\mathbf{p}}$ is treated as a probability distribution; meaning that $\hat{p}_j$ represents the probability that image

x is of class j. Ultimately, we are interested in the element of $\hat{\mathbf{p}}$ that contains the highest probability because this is the most likely class. To find the most likely class, as determined by the network, we use an $argmax$ function, which is a maximizer that finds the label and value of the highest probability.

With multiple classes, an adversary can conduct targeted attacks toward a certain class. Binary classification is a targeted and untargeted attack at the same time, because not only does a perturbation shift loss from the correct class, but it also specifically targets the other option. In a multiple class setting, when an untargeted attack occurs, the loss is shifted away from the correct label, but not towards any specific label. A targeted attack attempts to have the perturbed input classified as a specific class. This contrasts with the untargeted setting where any misclassification suffices. In the FGSM setting targeted perturbations are constructed as follows:

$$x' = x - \epsilon \text{ sign}(\nabla_x \text{Loss}(x, t)),$$

where t is the targeted class and is not x's true label. By subtracting the perturbation, which is now the gradient with respect to the data and a specific label, we are decreasing the loss for the targeted label, which from the perspective of the neural network, increases confidence for that class. Figure 2.8 is an example of an input image and output probabilities for this multinomial network. Figure 2.9 is that same image with a targeted attack attempting to classify the 1 as a 2.

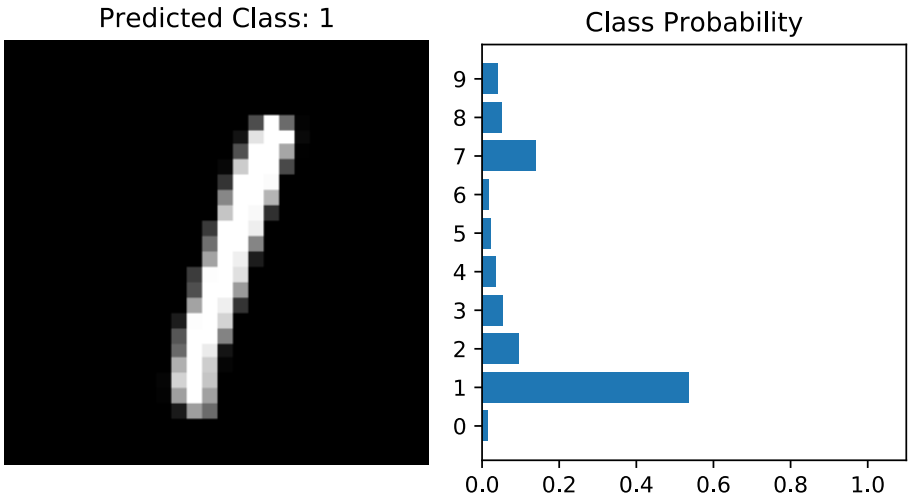

Figure 2.8. Left: Example of an inputted 1. Right: Output probabilities. The network classified this image correctly.

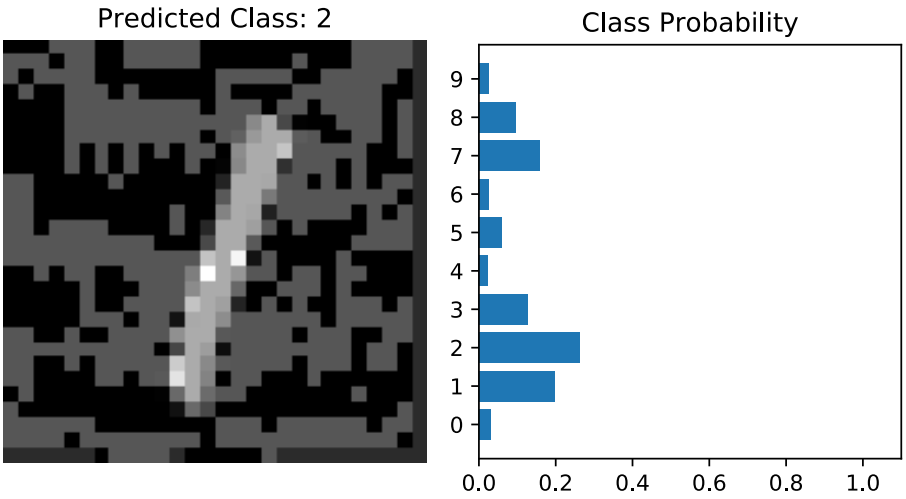

Figure 2.9. Left: Targeted attack against image from 2.8. Targeted class is a 2. Right: Output probabilities. The network now classified this image incorrectly and classified it as a 2.

An interesting feature about targeted attacks is that if one tries to take the gradient of the loss with the true label instead of an incorrect class, then the "perturbation" actually reinforces the confidence in the original class. Figure 2.10 shows an MNIST 2 along with the output

probabilities, while Figure 2.11 shows the same image but with a targeted attack against its true class. Remember, the perturbation that is introduced is a targeted attack against a 2. Notice the outputs actually classify this 2 at a higher probability than the 2 in Figure 2.10.

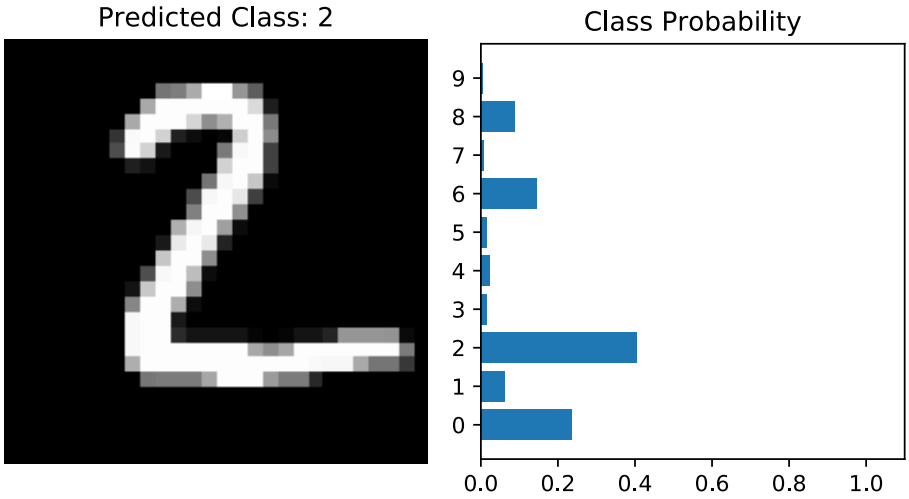

Figure 2.10. Left: Example of an inputted 2. Right: Output probabilities. The network classified this image correctly.

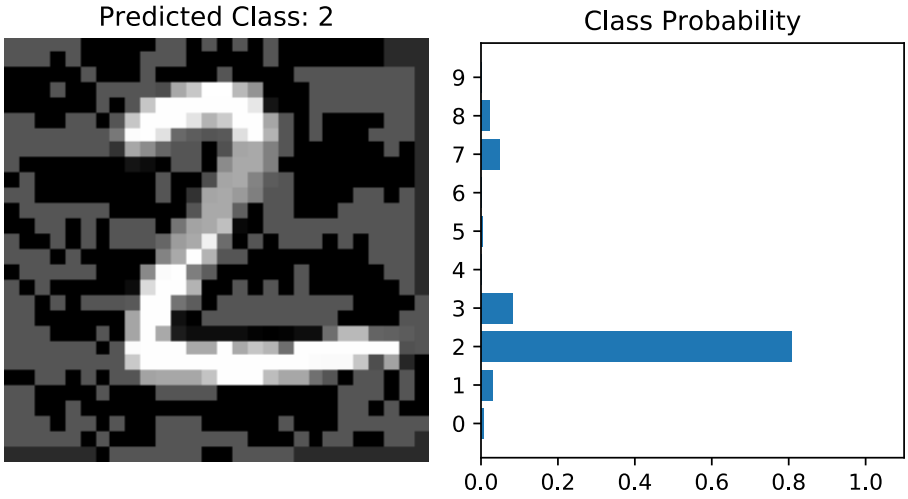

Figure 2.11. Left: Targeted attack against image from 2.10. Targeted class is a 2. Right: Output probabilities. The network now classified this stronger than the original image, even though it looks perturbed.

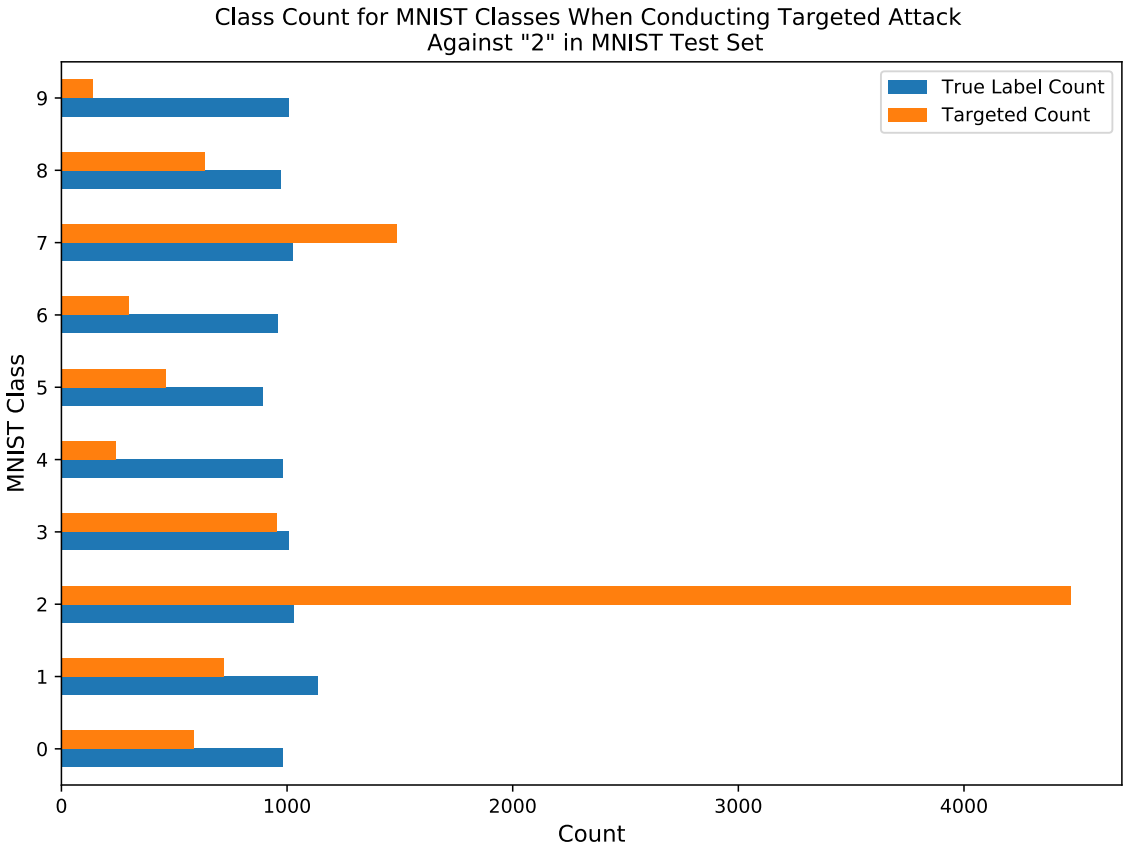

Figure 2.12. Targeted attack against class 2 using $\epsilon = .25$.

When conducting a targeted attack, obtaining a misclassification of the targeted class is not always easily obtained. Figure 2.12 highlights how a targeted attack shifts classification from true labels to the targeted class; however, notice how 7 also increases. In the MNIST dataset, 7's and 2's share a lot of the same features, as a result, sometimes a targeted 2 is classified as a 7. We highlight this to exemplify that not all classes can be easily targeted and misclassified.

While utilizing the MNIST dataset is useful for learning and testing new machine learning applications, the simplicity of the images allow perturbations to be easily seen. If ϵ is small enough, FGSM can maintain imperceptibility while causing misclassification. This final section highlights FGSM perturbations on real world photos and shows the effect of ϵ on classification and perceptibility of the perturbation in an image. We also showcase how an iterative version of FGSM differs from the one-step version. Figure 2.13 shows

the effects of increasing epsilon on one-step untargeted attacks. The difference between the original (top left) and the smallest perturbation (top middle) is almost imperceptible; however, as ϵ keeps increasing, the perturbation becomes quite noticeable. The title of each subplot in Figure 2.13 also reveals what the model classifies the image as with an appropriate probability. As seen, the model quickly identifies the perturbed image as a fireboat; however, as the step size increases, the confidence actually decreases. As the step moves away from the point in which the gradient was evaluated, the linear approximation of the loss becomes less accurate. A better way to approximate the model's loss is to take many smaller steps and redetermine the gradient, or direction of change, at each iteration. This is the Iterative-FGSM (I-FGSM).

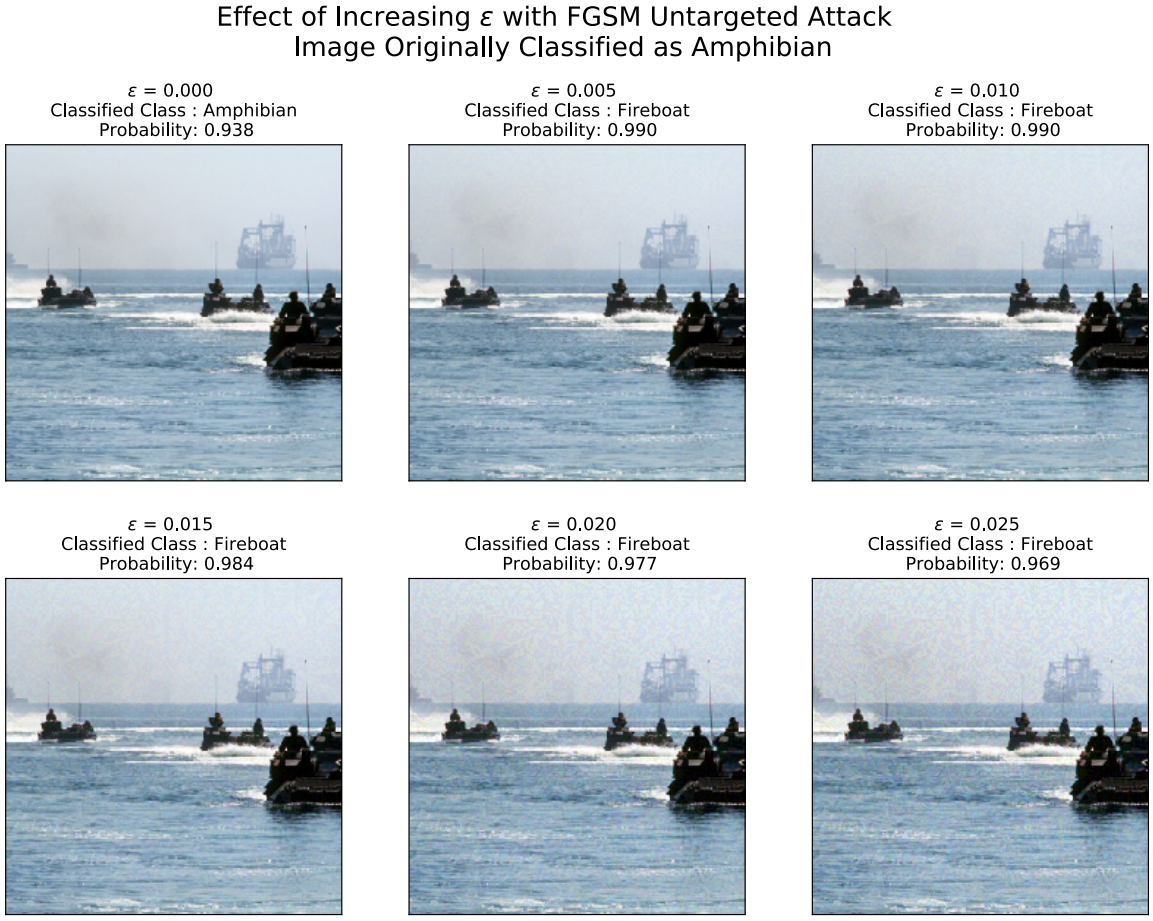

Figure 2.13. Using FGSM and increasing ϵ slightly causes quick misclassification in this case. However, as ϵ keeps increasing, the confidence of the misclassification actually decreases while the perturbation becomes quite noticeable. (Setting:image source - ImageNet; classifier - Inception V3)

Figure 2.14 shows how an iterative method can develop a better perturbation in terms of confidence of the misclassification. However, the perturbation is still visible during the fifth iteration. I-FGSM has potential to leave attackers vulnerable by utilizing an incorrect amount of iterations during an attack. FGSM forces a step on every iteration which can result in excessive change in the perturbation. This method lacks the advantages of modern optimizers which contain traits such as variable learning rates, line search, and momentum. In the next section, we introduce methods which utilize these tools in order to develop perturbations.

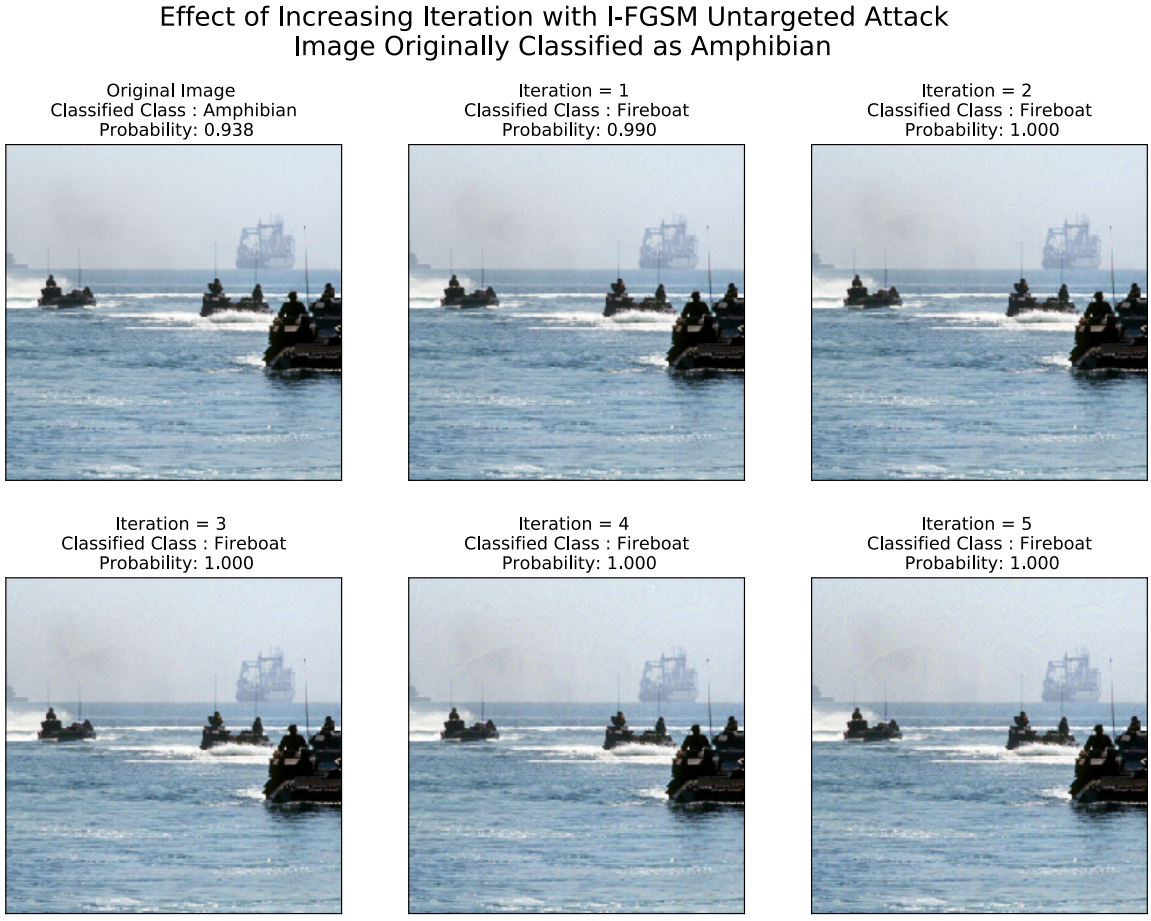

Figure 2.14. Shows the effect of increasing iterations while using the I-FGSM. For this example, $\epsilon = 0.005$. (Setting:image source - ImageNet; classifier - Inception V3)

2.2 L-BFGS

Prior to the FGSM, Szegedy et al. coined the term "adversarial examples" by introducing non-random imperceptible perturbations to a test image (Szegedy et al. 2013). In their paper, they generated their perturbations using a box-constrained Limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) algorithm (Szegedy et al. 2013). At a high level, L-BFGS can be summarized as follows. The BFGS portion is a Quasi-Newton method to solve unconstrained nonlinear optimization problems (Fletcher 2013). L-BFGS differs from the original algorithm because it periodically restarts the BFGS algorithm after a certain

number of iterations, thus utilizing less computer memory.

A neural network is a function $F(x) = y$ where the input $x \in \mathbb{R}^n$ produces an output $\hat{p} \in \mathbb{R}^N$, where n is the dimension of images and N is the number of classes. If we state that the $C(x) = argmax_i F(x)_i$, Szegedy et al.'s ideal problem for a targeted attack is the constrained optimization problem seen in problem 2.11.

$$\begin{aligned} \underset{x'}{\text{minimize}} \quad & ||x - x'||_2^2 \\ \text{subject to} \quad & C(x') = t \\ & x' \in [0,1]^n \end{aligned} \tag{2.11}$$

However, because this problem is very hard to solve, they instead solve problem 2.12,

$$\begin{aligned} \underset{x'}{\text{minimize}} \quad & c \cdot ||x - x'||_2^2 \; + \; \text{Loss}(F(x'), t) \\ \text{subject to} \quad & x' \in [0,1]^n \end{aligned} \tag{2.12}$$

where $\text{Loss}(F(x'), t)$ is a function that takes positive values (Carlini and Wagner 2017). In our application of their algorithm, we use cross entropy loss. The choice of the penalty parameter c is important for generating good perturbations; Carlini and Wagner suggest to "repeatedly solve this optimization problem for multiple values of c, adaptively updating c using bisection search or any other method for one-dimensional optimization" (Carlini and Wagner 2017).

Just as we did with FGSM, we highlight how L-BFGS perturbations change over iteration count. Unlike FGSM, which forces a step size of ϵ for every pixel, the step size for L-BFGS differs because the optimization algorithm allows shorter steps in order to find a better solution. The ability for the algorithm to take a smaller step when the larger step returns a non-optimal value is known as line search. Figure 2.15 shows how each iteration increases the misclassification confidence until iteration four to five, where no more improvement can be made. Unlike I-FGSM, as seen in Figure 2.14, where additional iterations adds perceptibility, that does not happen with L-BFGS because of the line search feature. Another striking difference with this algorithm is that the perturbation remains

imperceptible as the iterations increase. Figure 2.16 shows for each iteration the top 5 classes from the model; in this case, one class (Fireboat) overwhelmingly dominates all others from the first iteration.

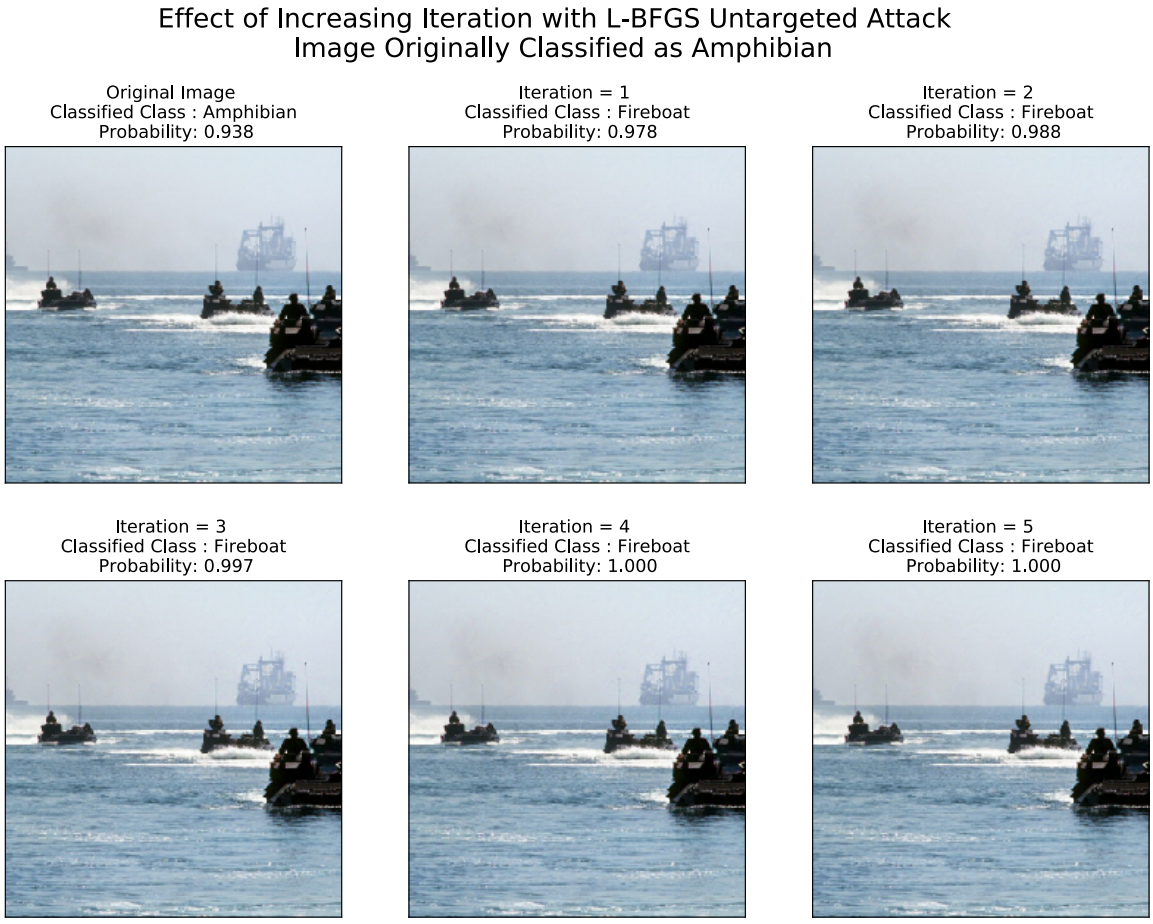

Figure 2.15. As iterations increase using L-BFGS, the perturbation actually settles and finds a perturbation that maximizes misclassification confidence. This is different than (Setting: image source - ImageNet; classifier - Inception V3)

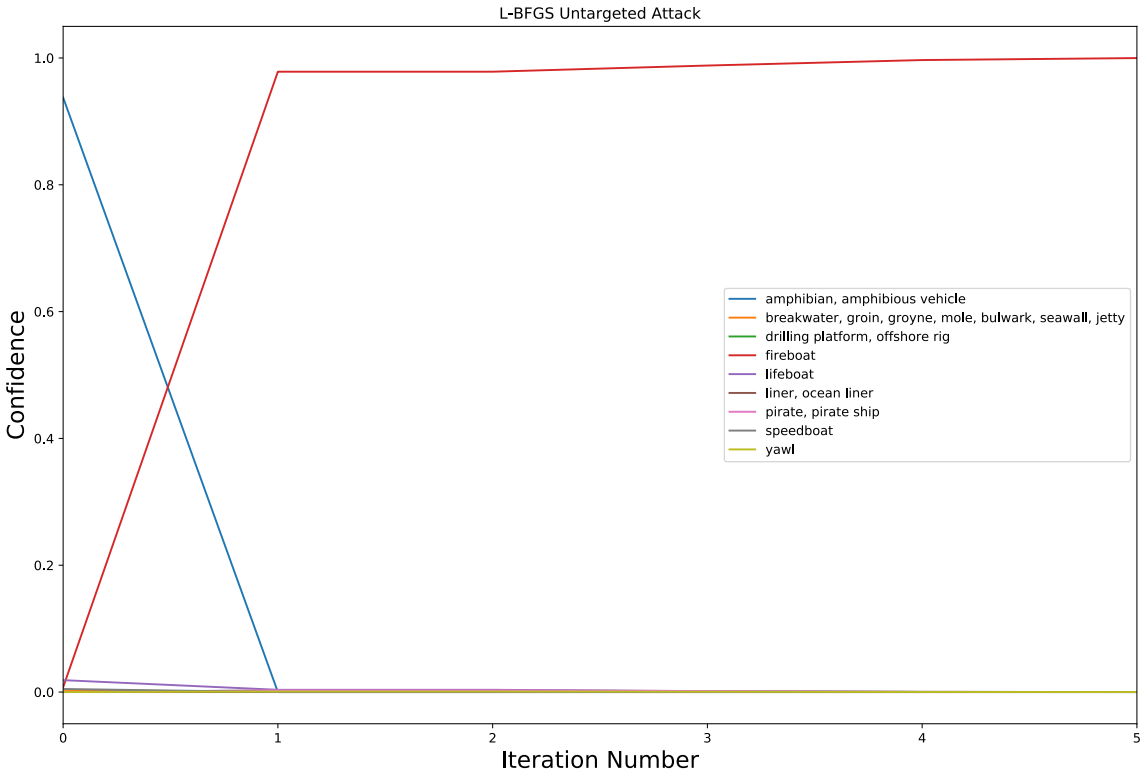

Figure 2.16. Confidence of top 5 classes for untargeted perturbations as iterations are increased.

2.3 Introduction to Carlini-Wagner ℓ_2 Attack

When adversarial perturbations were discovered, researchers in the field were surprised that a combination of small pixel changes could cause a misclassification in neural network based classifiers (Szegedy et al. 2013). This raised concerns about robustness of neural networks and was the motivation for Carlini and Wagner to test how small perturbations affected misclassification. Their goal was to create a perturbation that would find the minimum amount of change to cause a misclassification. Even though they developed their attacks to test the robustness of neural networks, these attacks showed enough promise that they have become benchmark comparisons in the field of image perturbation generation.

Carlini and Wagner started with nearly the same optimization problem presented by Szegedy et al. four years prior seen in problem 2.13. Given that $x \in [0,1]^n$ is an image and δ

represents the perturbation introduced:

$$\begin{aligned} \underset{\delta}{\text{minimize}} \quad & D(x, x+\delta) \\ \text{subject to} \quad & C(x+\delta) = t \\ & x+\delta \in [0,1]^n \end{aligned} \tag{2.13}$$

where t is a targeted class and $D(x)$ represents a function to measure some form of distance. The second constraint in problem 2.13 is the introduction of the box constraint in order to produce a valid image. Normally, 8-bit images contain numbers from 0-255 for their pixel values; however, these are frequently scaled to between 0 and 1. The box constraint ensures that when a perturbation is added to an image, the new values are still within the bounds 0-1, i.e. $0 \leq x + \delta \leq 1$.

Carlini and Wagner state that problem 2.13 is very difficult to solve because $C(x + \delta) = t$ is highly nonlinear, so they decide to formulate it in a way better for optimization. They introduce the function f such that $C(x + \delta) = t$ if and only if $f(x + \delta) \leq 0$. In their paper, they outline seven choices for f. However, for their ℓ_2 attack, Carlini and Wagner utilize $f(x') = max(max\{Z(x')_i : i \neq t\} - Z(x')_t, -\kappa)$, where $Z(x)$ is the output of the neural network except the softmax function (the logits) (Carlini and Wagner 2017). We emulate their choice of f in our implementation of their code. The variable κ allows control of the confidence for which misclassification occurs. Carlini and Wagner set $\kappa = 0$ for their attacks but this parameter creates a side benefit that encourages the solver to find x' that will be classified as class t with higher confidence (Carlini and Wagner 2017). Like Carlini and Wagner, we maintain $\kappa = 0$ for our replication of their attacks, not only to maintain purity with their original attack, but also because no other attack tested in this paper has a parameter that is comparable. Thus, the original optimization problem can be reduced to problem 2.14:

$$\begin{aligned} \underset{\delta}{\text{minimize}} \quad & D(x, x+\delta) \\ \text{subject to} \quad & f(x+\delta) \leq 0 \\ & x+\delta \in [0,1]^n \end{aligned} \tag{2.14}$$

However, like Szegedy et al., Carlini and Wagner decide to add a penalty to simplify the optimization problem, as seen in problem 2.15:

$$\begin{aligned} &\underset{\delta}{\text{minimize}} \quad D(x, x+\delta) \; + \; c \cdot f(x+\delta) \\ &\text{subject to} \quad x + \delta \in [0,1]^n \end{aligned} \tag{2.15}$$

Lastly, as seen in problem 2.16 reforming the metric D into an ℓ_2-norm, the problem becomes, given x, find δ that solves:

$$\begin{aligned} &\underset{\delta}{\text{minimize}} \quad ||\delta||_2 \; + \; c \cdot f(x+\delta) \\ &\text{subject to} \quad x + \delta \in [0,1]^n \end{aligned} \tag{2.16}$$

Carlini and Wagner investigate three separate approaches for this problem: projected gradient descent, clipped gradient descent, and change of variables (Carlini and Wagner 2017). This last approach, change of variables, is the novel part of their work that really stands out. They propose that instead of optimizing over δ as defined in the last problem, that instead a new variable w be introduced. By using equation 2.17 for δ_i they apply change-of-variables and optimize over w.

$$\delta_i = \frac{1}{2}(tanh(w_i) + 1) - x_i \tag{2.17}$$

Since $-1 \leq tanh(w_i) \leq 1$ that it follows the previously defined box constraint $0 \leq x + \delta \leq 1$, and thus a solution will automatically be valid.

Replacing δ with the new variable equation and utilizing a ℓ_2-norm distance metric, their ℓ_2 targeted attack, as seen in problem 2.18, becomes, given x, and a target class t such that $t \neq C(x)$, find w that solves:

$$\begin{aligned} &\underset{w}{\text{minimize}} \quad ||\frac{1}{2}(tanh(w_i) + 1) - x_i||_2^2 \; + \; c \cdot f(\frac{1}{2}(tanh(w_i) + 1)) \\ &\text{subject to} \quad x' \in [0,1]^n \end{aligned} \tag{2.18}$$

The constant c in the penalty portion of the objective function above is crucial because when $c = 0$ the gradient descent will not move away from the original image. Yet, comparable to a large ϵ in FGSM, a large c value will cause the first steps of the optimization to behave in an overly-greedy way where f dominates the objective function and the distance metric is undervalued. This results in poor perturbations. In order to obtain the correct value of c, the algorithm conducts twenty iterations of a binary line search (Carlini and Wagner 2017). For each c selected the algorithm then runs 10,000 iterations of the Adam optimizer. Ultimately, the smallest perturbation that causes misclassification is returned.

When developing perturbations that cause misclassification, Carlini and Wagner's ℓ_2 algorithm executes as advertised. However, the execution of the algorithm makes it very hard to compare to our perturbation method or other methods used in this paper. Since c is optimized and then there is an optimizer ran for each c value found, the computation time for one perturbation is much larger than other perturbation methods. We attempted to find a constant value of c that would work for all images and allow us to compare Carlini-Wagner's attack one iteration at a time; however, we obtained average results with some images and dismal with others. One of the main reasons is that the value of c is so particular, that if it is not optimized, then their objective function is not properly balanced. While perturbations can be one-step or iterative, Carlini-Wagner's algorithm is drastically impeded when implemented in a one-step method.

Carlini Wagner ℓ_2 Untargeted

1. Value: 0.9668 Tank
2. Value: 0.0017 Amphibian
3. Value: 0.0005 Half track
4. Value: 0.0002 Modem
5. Value: 0.0002 Aircraft carrier
6. Value: 0.0002 Disk brake
7. Value: 0.0002 Menu
8. Value: 0.0002 Fountain
9. Value: 0.0001 Bib
10. Value: 0.0001 Stingray

1. Value: 0.4863 Amphibian
2. Value: 0.4555 Tank
3. Value: 0.0022 Half track
4. Value: 0.0004 Military uniform
5. Value: 0.0004 Robin
6. Value: 0.0004 Bulletproof vest
7. Value: 0.0003 Aircraft carrier
8. Value: 0.0003 Black swan
9. Value: 0.0003 Stingray
10. Value: 0.0003 Wok

Figure 2.17. Top: Original image with top 10 classes. Bottom: untargeted perturbation with top 10 classes. (Setting:image source - ImageNet; classifier - Inception V3, perturbation - Carlini Wagner ℓ_2: iterations = 20, learning rate = .01)

Figure 2.18. Top: Original image with top 10 classes. Bottom: Targeted perturbation against half-track with top 10 classes. (Setting:image source - ImageNet; classifier - Inception V3, perturbation - Carlini Wagner ℓ_2: iterations = 20, learning rate = .01)

For this reason, we decide to use Carlini-Wagner's methods for qualitative and quantitative comparison of individual images. We cannot exhibit the change of perturbation per iteration, such as in Figures 2.13, 2.14, and 2.15, without drastically misrepresenting the

original purpose of the algorithm. Figures 2.17 and 2.18 only show input and output with classification for Carlini-Wagner's ℓ_2 attack. To implement Carlini and Wagner's ℓ_2 attack, we used code from an open source repository (Wu 2018). In these figures, it should be noted that the classification of the targeted image is not as high compared to other perturbation methods. The reason for this is that $\kappa = 0$, so the algorithm is trying to find the perturbation that barely causes misclassification. Lastly, for these perturbations, we used twenty iterations for each c value because it saves a lot of time and computing power, but as can be seen in Figures 2.17 and 2.18 the perturbations still cause misclassification, it just takes a fraction of the time to obtain.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 3:
Methodology and Results

In this chapter our goal is to highlight our contributions to the field of image perturbations. The first section shows the tools we utilized as well as methods and results for both our new constraints. The last section highlights the difference between our methods and previous perturbation techniques, as well as provides recommendation for future work .

3.1 Methodology

In order to produce results that are comparable to other works, we decided to use the ImageNet dataset for our images and Inception Version 3 (V3) as our classifier model. The ImageNet dataset was originally designed to be a compilation of realistic photos inspired to aid the computer image and vision research field. The data set is organized by groups of words and phrases called 'synonym sets' or 'synsets' (Deng et al. 2009). These sets of words act as the labels for the images. The designers of the dataset have organized, on average, 1,000 images for each label. Figure 3.1 is an example of some of the photos with their paired labels. The other reason we were drawn to the ImageNet dataset is that it is a common sandbox used throughout the field of image perturbation generation (Szegedy et al. 2013; Carlini and Wagner 2017; Goodfellow et al. 2014).

Utilizing such a large and complex dataset like ImageNet requires an image classifier that can differentiate the multiple classes and return a high accuracy. Inception V3 boasts over an 80% classification rate with the ImageNet dataset (Szegedy et al. 2016). Another benefit is the complexity of the network. If our final goal is to aid the DoD, our perturbation must be successful against a network that would realistically be employed by the government. Inception V3's ability to decipher and classify the ImageNet dataset exemplifies its application potential. The last benefit of using Inception V3 is that Carlini and Wagner used this classifier with the ImageNet dataset for their attacks. Utilizing the same classifier and dataset allows us a direct comparison to their work.

We utilized the Python language, and specifically the PyTorch library, to accomplish our work. PyTorch is a library designed around the idea of open source machine learning with

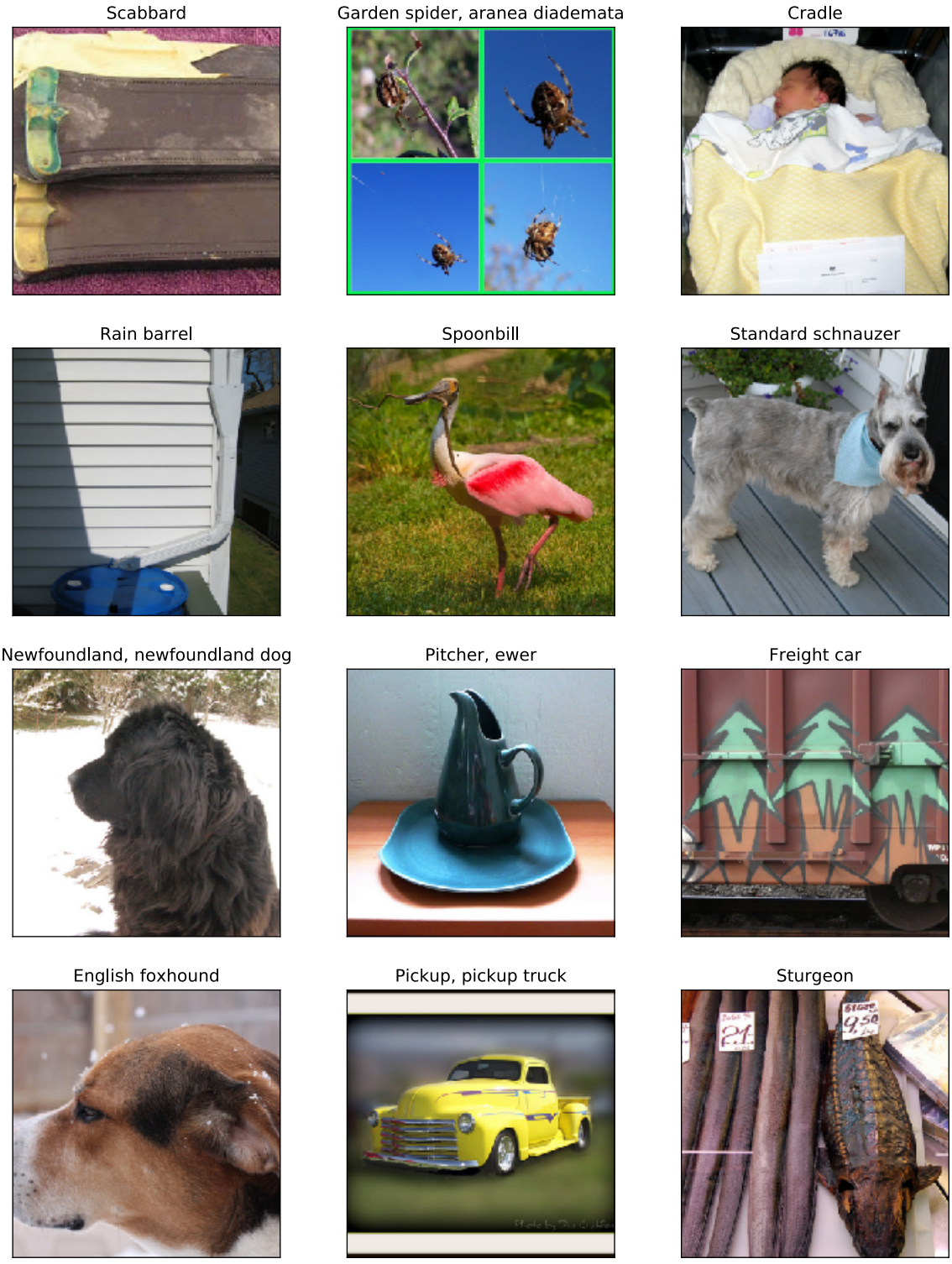

Figure 3.1. Example of ImageNet photos with their respective labels. (Source: ImageNet Database)

a goal of expediting the process from prototypes to production employment (Paszke et al. 2017). A couple of the main benefits from utilizing PyTorch is that it allows for quick adjustments for experimentation and pre-trained networks. The way the PyTorch library is written allows us to convert quickly to CIELAB space and back for our perturbations; this task would be more strenuous in another library. Specifically, PyTorch has enabled our research because it has tools designed to easily extract gradient information from a neural network. Our second advantage, and a huge time saver, is that PyTorch contains pre-trained neural networks, to include Inception V3. The ability to load a pre-trained Inception V3 network saves us time and computing power because training the network would require processing millions of images from ImageNet.

3.1.1 Color Aware

As introduced in Section 1.3.3, our goal is to achieve a perturbation generation technique that is more imperceptible to the human eye than previous methods. In this section, we introduce our *Color Aware (CA)* perturbation technique. This technique differentiates itself from other perturbation generation methods by leveraging a conversion to CIELAB space where both gradient approximation and optimization occur. The code we used to convert RGB data to CIELAB space was obtained from an open source repository (Zhang et al. 2017). While in CIELAB colorspace, we also add a ℓ_2-norm constraint across the color channels in order to capitalize on CIELAB colorspace's unique property of matching human perception. This ℓ_2-norm is denoted as $\|\cdot\|_{2,c}$, where c indicates that the norm is taken over the color channel dimension for that given image.

An adversary conducting a targeted attack will attempt to maximize problem 3.1. Similar to FGSM problem 2.2, this constrained problem develops a perturbation in CIELAB space x'_{LAB} from an image whose data has been translated from RGB to CIELAB, x_{LAB}. However, the step size ϵ is constrained by an ℓ_2-norm across the color channel.

$$\begin{aligned} \underset{x'_{LAB}}{\text{maximize}} \quad & \text{Loss}(x'_{LAB}, t) \\ \text{subject to} \quad & \|x - x'\|_{2,c} \leq \epsilon \end{aligned} \tag{3.1}$$

However, if we alter problem 3.1 by taking a first-order approximation of the objective, we

obtain problem 3.2.

$$\begin{aligned} \underset{\delta}{\text{maximize}} \quad & \nabla_{x_{LAB}} \text{Loss}(x_{LAB}, t)^T \delta \\ \text{subject to} \quad & \|\delta\|_{2,c} \leq \epsilon \end{aligned} \tag{3.2}$$

Our implementation of problem 3.2 can be seen in Algorithm 1.

Algorithm 1 Color Aware

Input:

X_{RGB} : Input image in RGB colorspace

t : Target Label

K : Number of Iterations

ϵ: ℓ_2-norm bound

Output:

X'_{RGB} : Perturbed Image

Initialize $\delta = 0$

$X_{LAB} \leftarrow X_{RGB}$ ▷ Convert RGB Image to LAB

for $k \leftarrow$ 1 to K **do**

    $g \leftarrow \nabla_{X_{LAB}} \text{Loss}(F(X_{LAB} + \delta), t)$

    $\delta \leftarrow \delta + \frac{\epsilon \cdot g}{\|g\|_{2,c}}$

end for

$X'_{LAB} \leftarrow X_{LAB} + \delta$

$X'_{RGB} \leftarrow X'_{LAB}$ ▷ Convert LAB Perturbation to RGB

$X'_{RGB} \leftarrow clip(X'_{RGB}, 0, 1)$ ▷ If perturbation extends past the boundary, adjust the values back to nearest boundary value

return X'_{RGB}

The way the constraints are written in the current algorithm make it comparable to FGSM. However our algorithm applies the process in CIELAB space and we apply an ℓ_2-norm constraint across the color channels, whereas FGSM uses ℓ_∞ norm. Figure 3.2 showcases one example of our perturbation for a targeted attack.

Color Aware

1. Value: 0.9492 Tank
2. Value: 0.0040 Amphibian
3. Value: 0.0020 Half track
4. Value: 0.0010 Cannon
5. Value: 0.0006 Bulletproof vest
6. Value: 0.0005 Military uniform
7. Value: 0.0005 Cliff dwelling
8. Value: 0.0004 Projectile
9. Value: 0.0004 Missile
10. Value: 0.0003 Aircraft carrier

1. Value: 0.9895 Amphibian
2. Value: 0.0059 Tank
3. Value: 0.0009 Half track
4. Value: 0.0003 Jeep
5. Value: 0.0002 Bulletproof vest
6. Value: 0.0002 Military uniform
7. Value: 0.0000 Assault rifle
8. Value: 0.0000 Terrapin
9. Value: 0.0000 Rifle
10. Value: 0.0000 Projectile

Figure 3.2. Top: Original image classified with top 10 classifications. Bottom: Color Aware targeted attack with target label amphibian and top ten classes. (Setting: image source - ImageNet; classifier - Inception V3; perturbation - Color Aware: $\epsilon = .1$, iterations $= 5$)

3.1.2 Color and Edge Aware

Color & Edge Aware is a combination of our Color Aware algorithm and an additional constraint based on edge magnitudes. As can be seen in Figure 3.3, perturbations are much harder for the human eye to detect when there is texture in the image. Our second contribution adds a constraint based on the amount of edge present at each pixel location in the image as determined by an edge filter. The objective of this additional constraint is to limit perturbations in visually smooth portions of the image. As explained in Section 1.3.3 we utilize an off-the-shelf Sobel filter from *scikit-image* library in Python; however, any edge detection algorithm can suffice (van der Walt et al. 2014). Given a grayscale image, a Sobel filter when convolved over the image, outputs an image where each pixel's value represents the edge magnitude. The higher the magnitude, the more edge is present in that pixel. These edge magnitudes allow us to scale the magnitude of the perturbation with the edge magnitude at each pixel. The stronger the edge, the larger the perturbation allowed on that respective pixel.

Figure 3.3. Left: Image that has been perturbed by FGSM. Middle: Enlarged section of sky. Note how the visibility of the perturbation is evident in homogeneous smooth region. Right: Enlarged section of water. Notice how the perturbation is harder to detect because of the texture in this section of the image. (Setting: image source - ImageNet; classifier - Inception V3; perturbation - FGSM: $\epsilon = .015$, iterations $= 1$)

The flow of this algorithm is unique because we use the original data twice, once to calculate the edges, and then again when the perturbation is generated. The summary of Color & Edge Aware is visualized in Figure 3.4.

Algorithm 2 Color & Edge Aware

Input:

X_{RGB} : Input image in RGB colorspace
t : Target Label
K : Number of Iterations
ϵ: ℓ_2-norm bound

Output:

X'_{RGB} : Perturbed Image

Initialize $\delta = 0$
$X_{GS} \leftarrow X_{RGB}$ ▷ Convert RGB to Grayscale
$w \leftarrow f_{Edge}(X_{GS})$ ▷ Generate edge weights utilizing an edge detection filter
$X_{LAB} \leftarrow X_{RGB}$ ▷ Convert RGB Image to LAB
for $k \leftarrow$ 1 to K **do**
    $g \leftarrow \nabla_{X_{LAB}} \text{Loss}(F(X_{LAB} + \delta), t)$
    $\delta \leftarrow \delta + \frac{w \cdot \epsilon \cdot g}{\|g\|_{2,c}}$ ▷ Add the weights as a constraint
end for
$X'_{LAB} \leftarrow X_{LAB} + \delta$
$X'_{RGB} \leftarrow X'_{LAB}$ ▷ Convert LAB Perturbation to RGB
$X'_{RGB} \leftarrow clip(X'_{RGB}, 0, 1)$
return X'_{RGB}

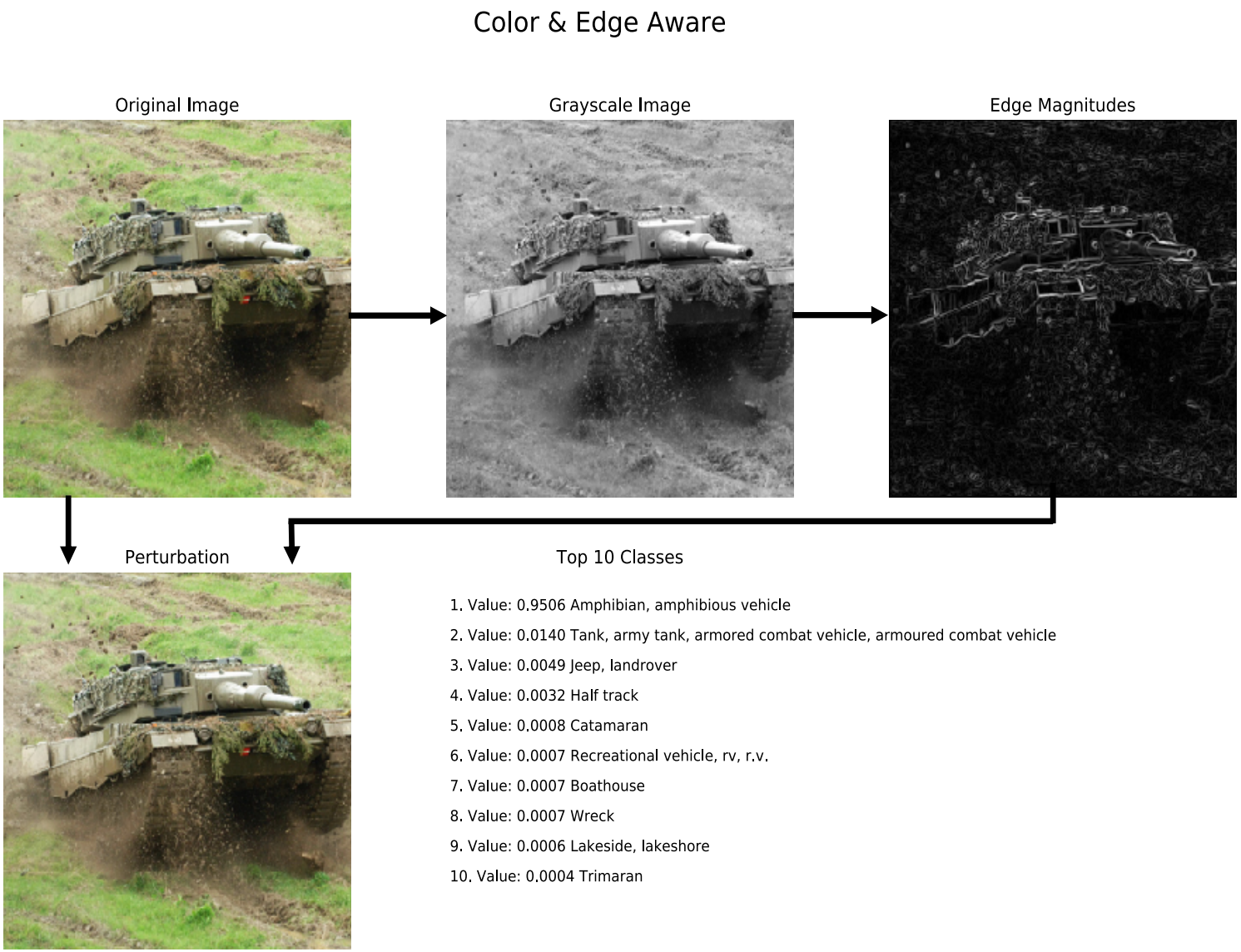

Figure 3.4. Visual flow of Color & Edge Aware. This algorithm is similar to Color Aware except we introduce the edge magnitudes as a constraint. For correct classification of the original image see Figure 3.2 (Setting: image source - ImageNet; classifier - Inception V3; perturbation - Color & Edge Aware: $\epsilon = 1$, iterations $= 5$)

3.2 Results

3.2.1 Comparing Color Aware to Color & Edge Aware

With the exception of the edge magnitude weighting, Color Aware and Color & Edge Aware are the same algorithm. From the perspective of misclassification confidence, adding the edge constraint does not make Color and Edge Aware better than Color Aware. This is because the Color & Edge Aware perturbation method imposes stronger constraints than the Color Aware method. However, Color & Edge Aware eliminates perturbations in homogeneous regions of the image, thereby, creating perturbations that are more imperceptible to humans. The issue is that with the added constraint, we cannot use the same parameters to create the perturbation. Specifically, the step size for Color & Edge Aware needs to be larger than Color Aware. Examining Figures 3.2 and 3.4 the confidence of the misclassification is similar; but it's important to note that the setting at which the algorithms were executed are different. Specifically, Color Aware was run five times with an $\epsilon = .1$, whereas Color & Edge Aware was run five times with a $\epsilon = 1$. In order to maintain similar iterations between the two algorithms, we relax the step size for Color & Edge Aware.

Figure 3.5 highlights how Color & Edge Aware lags Color Aware in confidence of misclassification per iteration if the parameters are left the same. Increasing the step size, ϵ, for Color & Edge Aware, as seen in Figure 3.6, allows it to achieve nearly the same level of confidence as Color Aware in the same amount of iterations.

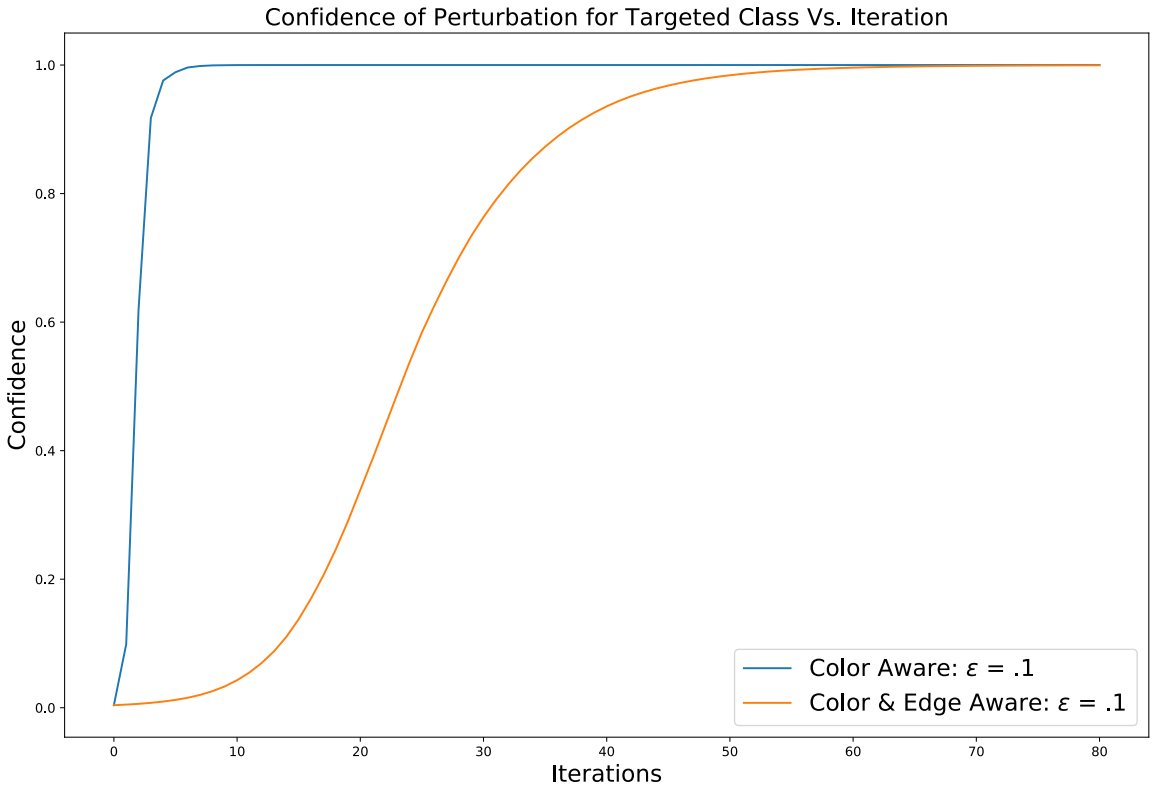

Figure 3.5. Plot of label confidence for a targeted attack over iteration for Color Aware and Color & Edge Aware. Color Aware can cause a targeted misclassification with high confidence within 5 iterations. For this particular image, it takes Color & Edge Aware approximately 50 iterations to get near the same misclassification confidence.(Setting: $\epsilon = .1$ for both perturbation techniques)

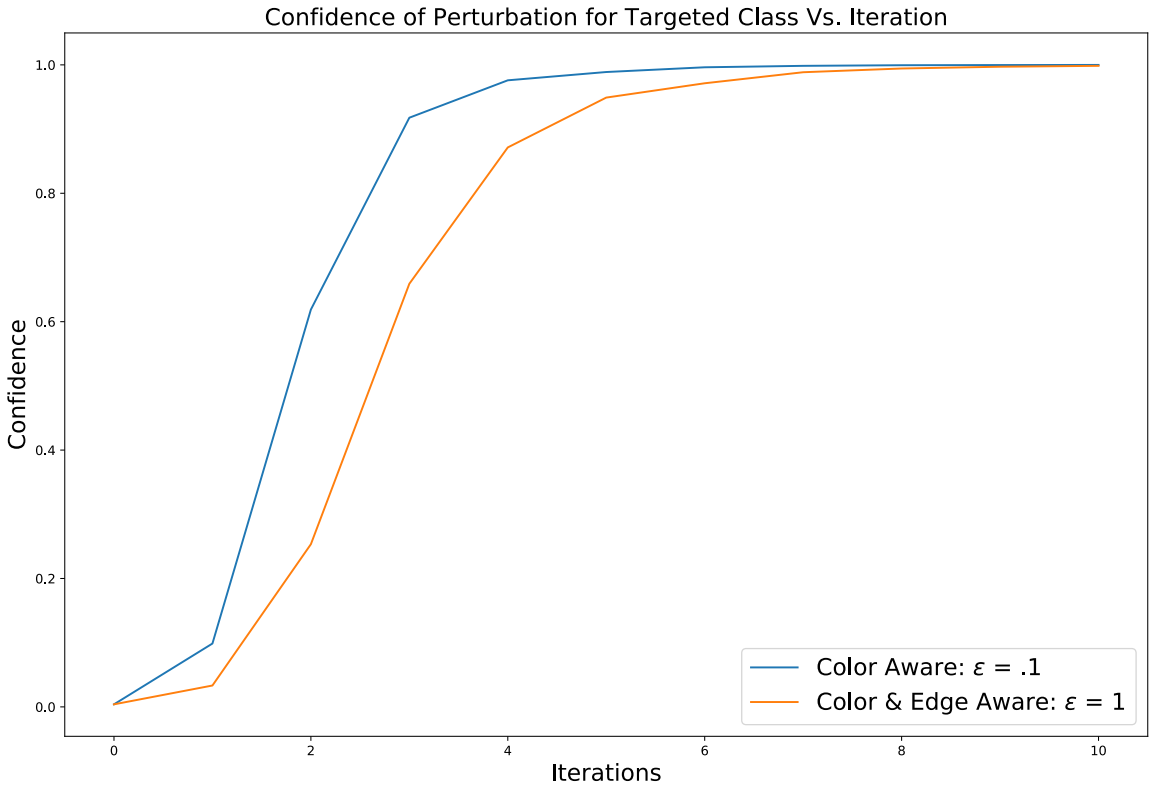

Figure 3.6. Plot of label confidence for a targeted attack over iteration for Color Aware and Color & Edge Aware. Increasing ϵ for Color & Edge Aware decreases the iterations needed for high misclassification. (Setting: $\epsilon = 1$ for C&EA, and $\epsilon = .1$ for CA)

Figure 3.7 shows an unperturbed image from the ImageNet validation set as well as both our Color Aware and Color & Edge Aware targeted attacks. Once again, it should be pointed out that both our perturbation techniques were successful conducting a targeted attack from class tank to class amphibian, with near the same confidence level. The perturbations, unless closely examined, are imperceptible from the original. What is different, as annotated above each perturbation, are the ℓ_1 and ℓ_2-norms. The ℓ_1-norm reflects the sum of the absolute change in pixel values from the original image, while the ℓ_2-norm reflects the root of the sum of the squared changes in pixel values from the original image. In general, the ℓ_2-norm is more sensitive to outliers. Thus the ℓ_2-norm reflects the trend in Color & Edge Aware to change fewer pixels but makes bigger changes in those pixels. The larger ϵ value results in larger ℓ_2-norms compared to Color Aware. Likewise, with unconstrained pixels, Color

Aware quickly develops a larger ℓ_1-norm but its ℓ_2-norm is less than Color & Edge Aware's. However, these values do not directly relate to perceptibility of a perturbation, they are merely metrics to help comprehend what is happening when the perturbation is too small to visually detect.

In order to aid the reader in viewing the perturbations, Figure 3.8 has enlarged sections from each image from Figure 3.7. To an untrained observer, the enlarged sections may still look the same, especially if it is not being viewed on a screen. However, the bottom left image in Figure 3.8, the Color Aware perturbation, does have visible perturbations, specifically visible in the sky or on the side of the building. Figure 3.9 shows just the change of pixel values. Anything that is gray means the pixel values were not changed. Pixels that appear to have distorted color are pixels where the perturbation has changed value. When comparing the left and right images in Figure 3.9 it should be apparent that the way the perturbation methods changed pixel values of the original image is different. Specifically, the Color & Edge Aware changes to the original image are exactly where our Sobel edge detection constraint allowed change, amongst the edges. Figure 3.10 shows the edge constraints of the enlarged section for reference.

Original Image

Classify: tank
Probability: 0.802

Color Aware: ℓ_1 Norm: 462.492
ℓ_2 Norm: 1.069

Classify: amphibian
Probability: 0.991

Color & Edge Aware: ℓ_1 Norm: 345.491
ℓ_2 Norm: 1.619

Classify: amphibian
Probability: 0.987

Figure 3.7. Top: Original image and classification. Bottom Left: Color Aware targeted perturbation and classification. Bottom Right: Color & Edge Aware targeted perturbation and classification. (Setting: image source - ImageNet; classifier - Inception V3; perturbation - Color Aware: $\epsilon = .1$, iterations $= 3$; perturbation - Color & Edge Aware: $\epsilon = 1$, iterations $= 6$)

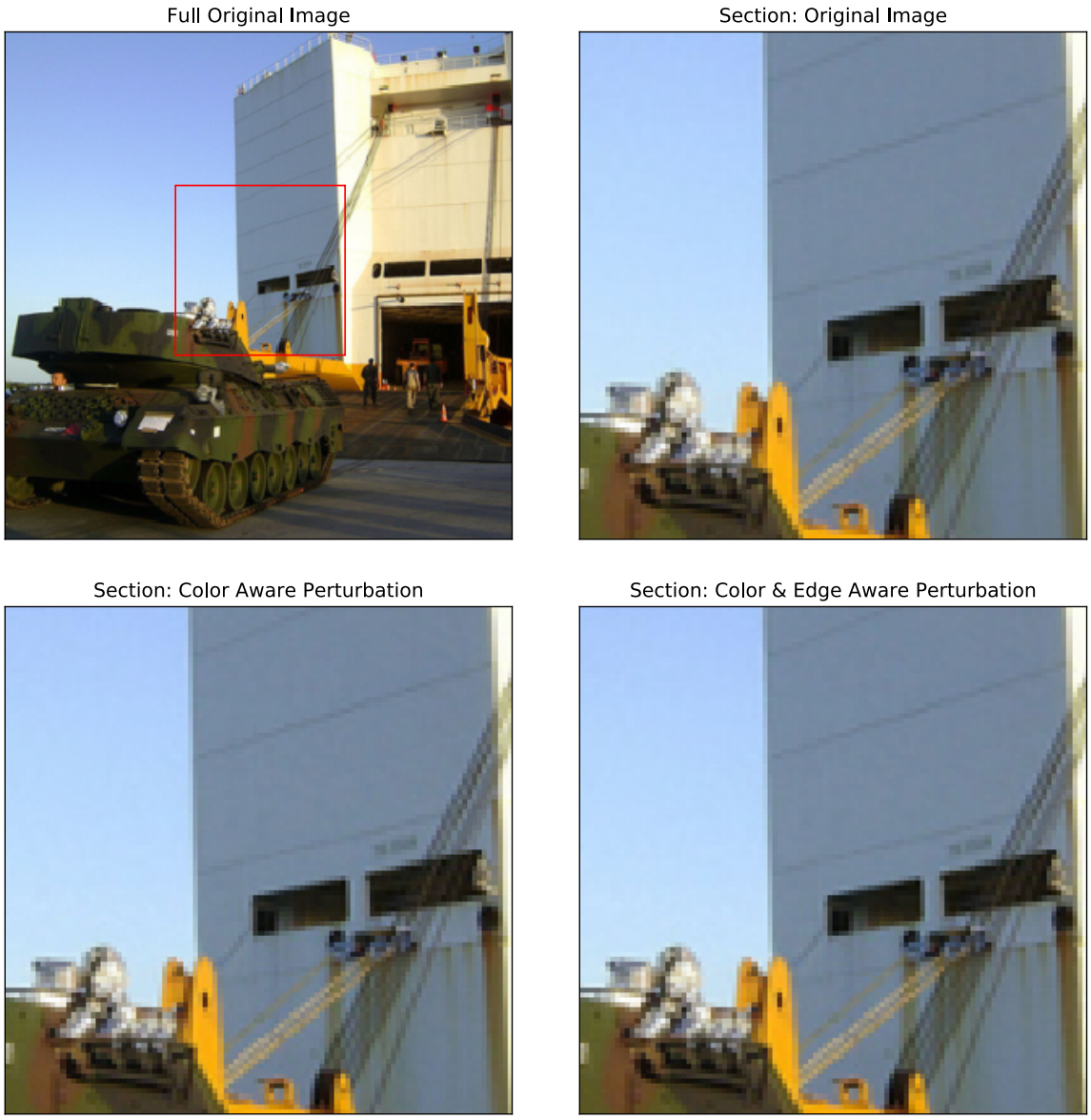

Figure 3.8. Original image from Figure 3.7 with enlarged sections from original image and each perturbation. (Setting: image source - ImageNet)

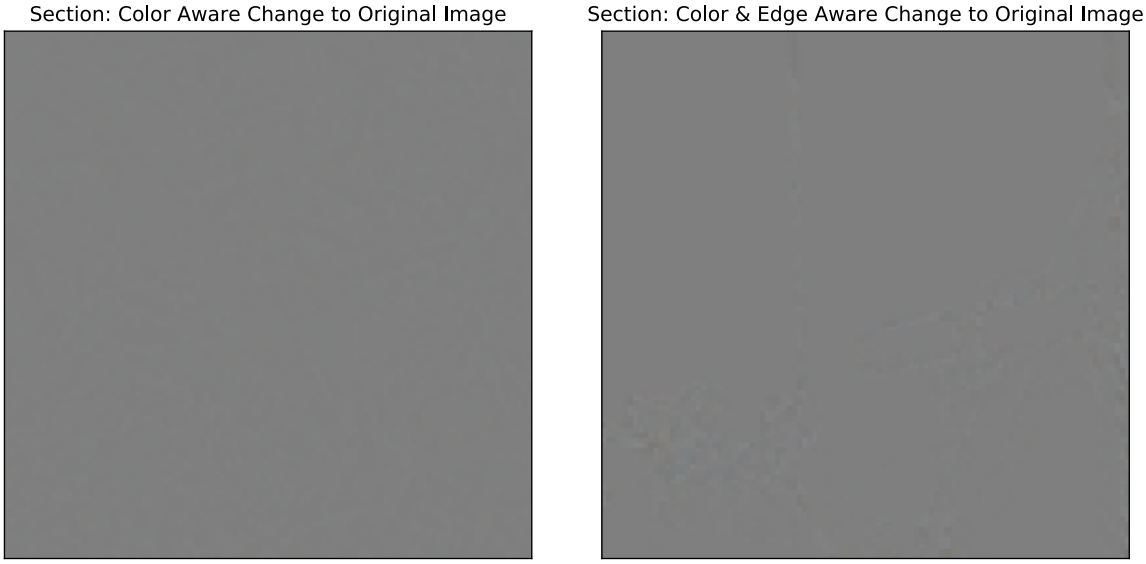

Figure 3.9. Left: Change that Color Aware made to original image from Figure 3.7. Change from Color Aware is mostly uniform across the image. Right: Change that Color & Edge Aware made to original image. In this image it is evident that the Color & Edge Aware constraint from the Sobel Filter worked. Note that the change is grouped in areas that match the Sobel filter in Figure 3.10 Flat gray represents no change. (Best viewed on a screen)

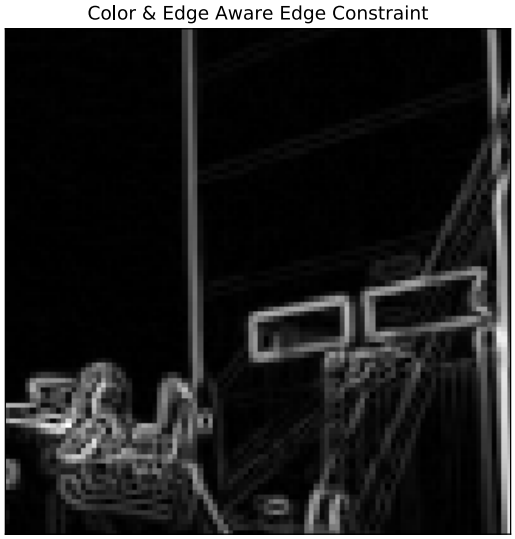

Figure 3.10. Original image from Figure 3.7 with enlarged sections from original image and each perturbation. (Setting: image source - ImageNet, edge detection filter - Sobel)

From a human perceptibility perspective, regions with edges hide perturbations much better than regions without. Figure 3.11 shows both Color Aware and Color & Edge Aware perturbations after six iterations with $\epsilon = 1$. Earlier we had to increase ϵ for Color & Edge Aware in order to produce similar misclassification confidence in the same number of iterations. Here we explore this topic further by pushing the boundaries with Color Aware and replicate the step size give to Color & Edge Aware. The left image in 3.11 shows a clearly perceptible perturbation because a lot of change has been made to the image. The remarkable aspect of Figure 3.11 is that some pixels in the Color & Edge Aware perturbation on the right experienced the same amount of change; yet, they are imperceptible. We believe this confirms that humans perceive perturbations in smooth, homogeneous regions of the image more easily than along edges.

Figure 3.11. Left: Color Aware perturbation Right: Color & Edge Aware perturbation (Setting: image source - ImageNet, classifier - Inception V3, perturbation - CA and C&EA both with 6 iterations and $\epsilon = 1$)

Even though we implemented the edge magnitude constraint on our own perturbation generation method Color Aware, we believe that an edge-weighting constraint could be applied to other perturbation methods. We will discuss this further in 3.2.3.

3.2.2 Comparing to Other Perturbation Generation Techniques

As mentioned in Chapter 2, we will compare Color Aware and Color & Edge Aware to I-FGSM, L-BFGS, and Carlini-Wagner's ℓ_2 Attack. Since perceptibility is subjective, it is difficult to rigorously test our claim that our contributions generate perturbations which are less perceptible to the human eye. Instead we provide examples that highlight each attack technique for both targeted and untargeted attacks, which can be seen in Figures 3.12 and 3.13. We note that the results in Figures 3.12 and 3.13 are representative of the trends seen throughout our research. Next to each example, we provide misclassification class, misclassification confidence, and the ℓ_1 and ℓ_2-norms. We hope that the ℓ_1 and ℓ_2-norms help the reader better understand how much change a perturbation has done to an image. While no ℓ_p-norm correlates directly to perceptibility, we hope it adds clarity to how the

images are manipulated.

In Figure 3.12 an untargeted attack is successful for each perturbation method. Four out of the five classified as a breakwater, which is a jetty, or a manmade piece of land that protects a harbor, while Carlini-Wagner classifies as a fireboat. After examining the original image, even though it was classified as a submarine with very high confidence, it is understandable how the perturbation can make the neural network shift towards a breakwater. The low silhouette of the submarine is similar in profile to jetties that the network saw in training. By increasing loss for submarines, the network chooses the next most-likely class. In the case of the Carlini-Wagner attack, it returns the perturbation that caused misclassification but with the least amount of changes as measured by ℓ_2-norm, which explains the lower misclassification confidence.

Figure 3.13 is similar to Figure 3.12; however, instead of allowing an untargeted attack, we attempt to have the network identify this submarine as a speedboat. As can be seen in the figure, each algorithm successfully had the network classify as a speedboat, with all but Carlini-Wagner having high misclassification confidence.

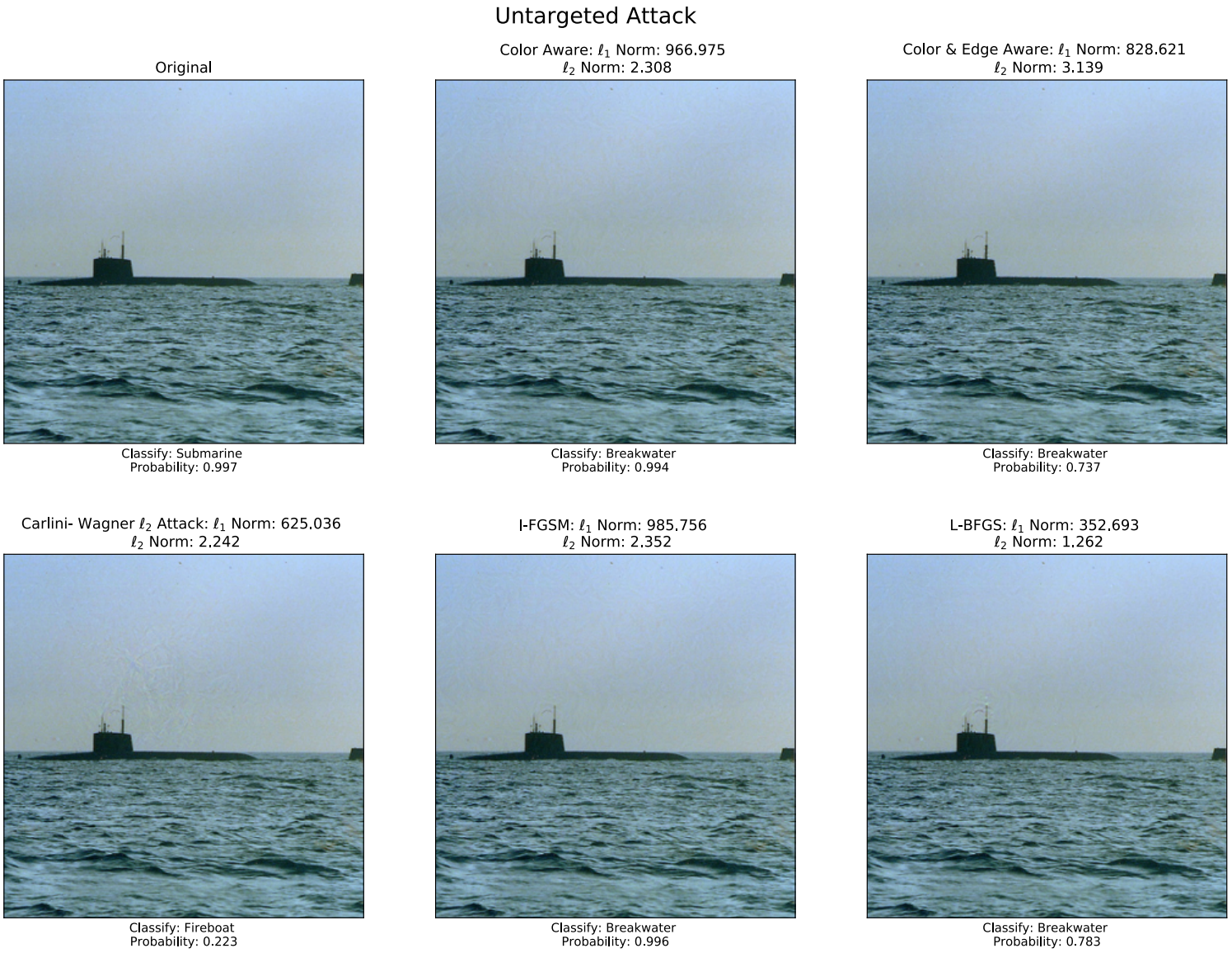

Figure 3.12. Visualization of different perturbation algorithms' untargeted attacks. (Setting: image source - ImageNet, classifier - Inception V3, perturbations - CA: $\epsilon = .1$, iterations = 10 - C&EA: $\epsilon = 1.5$, iterations = 10 - CW ℓ_2: learning rate = 1e-2, search steps = 20 - I-FGSM: $\epsilon = .001$, iterations = 10 - L-BFGS: iterations = 10)

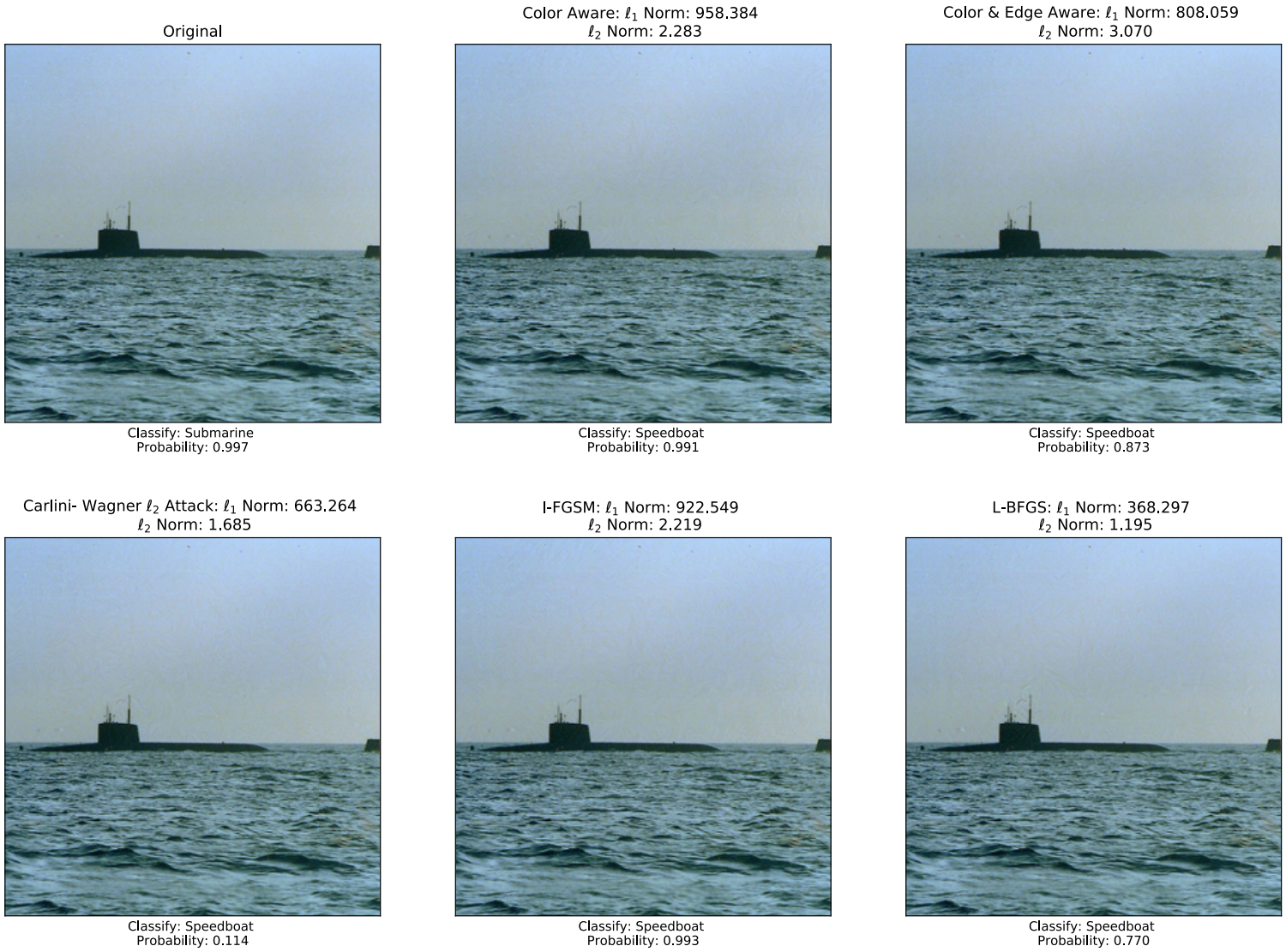

Figure 3.13. Misclassification confidence versus iterations for different perturbation methods. (Setting: image source - ImageNet; classifier - Inception V3; perturbations - CA: ϵ = .1, iterations = 10 - C&EA: ϵ = 1.5, iterations = 10 - I-FGSM: ϵ = .001, iterations = 10 - L-BFGS: iterations = 10)

Each perturbation result in Figures 3.12 and 3.13 is representative of the many results we obtained when perturbing other images. However, the perturbations in these figures are only a snapshot in time of the perturbation process. These images do not demonstrate the temporal element of the perturbation process, i.e. which algorithms gain misclassification the with the fewest iterations. Figure 3.14 shows misclassification confidence for the targeted label as a function of the number of iterations used. Carlini-Wagner's attack is not represented in the figures. Recall from section 2.3 that Carlini-Wagner's method was never intended to be used iteratively and for this reason we leave it out of these comparisons. For Figure 3.14, the original image used was the same submarine image from Figures 3.12 and 3.13 and the targeted label was once again a speedboat. Figure 3.14 highlights how Color & Edge Aware trails Color Aware, as first seen in Figure 3.5; however, it also shows the similarity between I-FGSM and Color Aware. This is no surprise because mathematically, the two algorithms are similar.

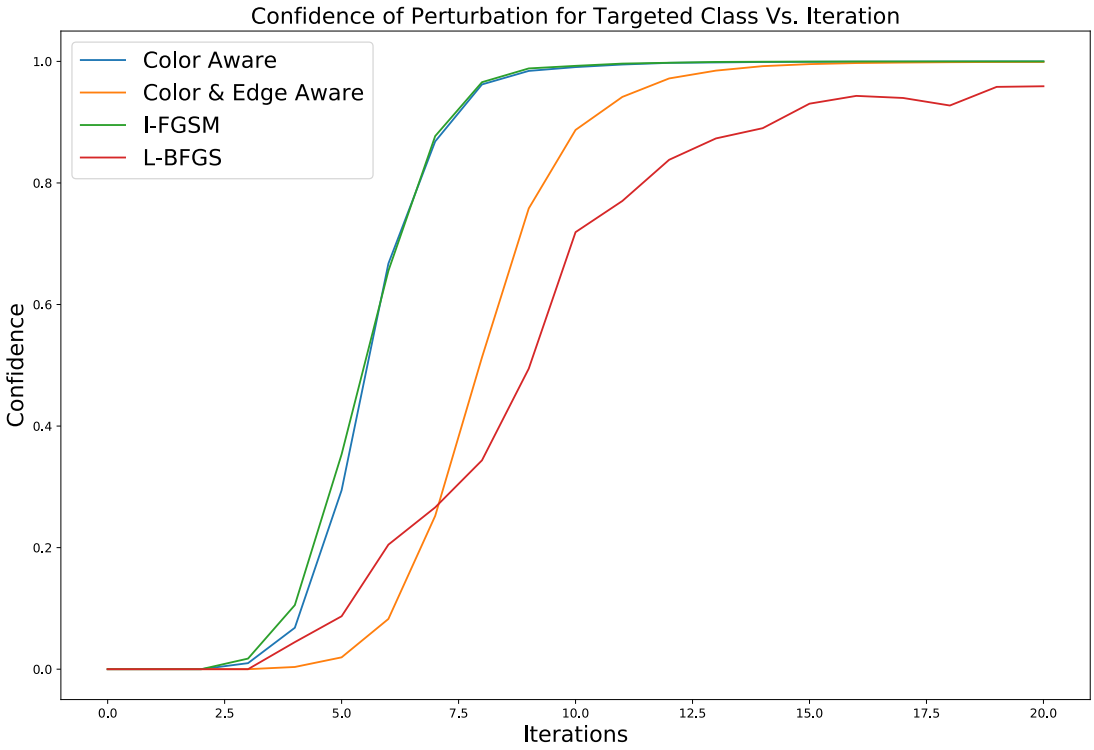

Figure 3.14. Misclassification confidence versus iterations. (Setting: perturbations - CA: $\epsilon = .1$, iterations = 20 - C&EA: $\epsilon = 1.5$, iterations = 20 - I-FGSM: $\epsilon = .001$, iterations = 20 - L-BFGS: iterations = 20)

Figures 3.15 and 3.16 shed some light on how L-BFGS perturbation method works. In order to develop these graphs, we captured the ℓ_1 and ℓ_2-norms at each iteration represented in Figure 3.14. We then plotted confidence of the targeted label on the y-axis and an ℓ_p-norm on the x-axis. These parameterized curves are of the form $(x(t), y(t))$, where t = iterations, $x(t) = \ell_p$-norm, and $y(t)$ = confidence of the targeted class. Evident in both Figures 3.15 and 3.16 is that the L-BFGS perturbation has a different trend than the other three perturbation methods. The unique structure of the L-BFGS curve illustrates that this perturbation method has a fundamentally different structure than the others. Since I-FGSM, Color Aware, and Color & Edge Aware force a step size for each iteration, the ℓ_1 and ℓ_2-norms will increase each iteration. However, L-BFGS's ability to conduct line search, or the ability to take a shorter step if the output is optimal, allows this algorithm to find a perturbation that has high misclassification but small ℓ_1 and ℓ_2-norms, with respect to the other methods. Since the graphs only represent the confidence of the targeted label, the flat red line along the first part of the x-axes is from the network not even identifying the targeted label after the first iteration of L-BFGS. However, upon subsequent iterations, L-BFGS increases confidence but goes down in ℓ_1 and ℓ_2-norms.

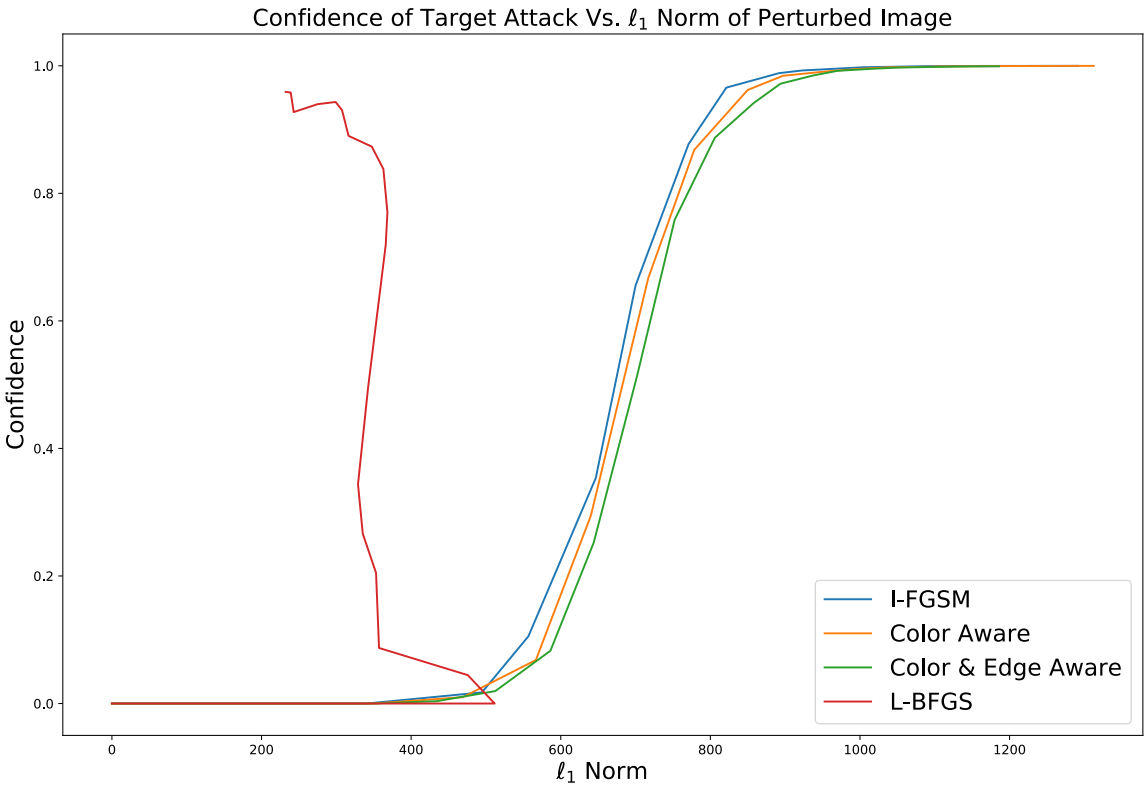

Figure 3.15. Misclassification confidence versus ℓ_1-norm. (Setting: perturbations - CA: $\epsilon = .1$, iterations $= 20$ - C&EA: $\epsilon = 1.5$, iterations $= 20$ - I-FGSM: $\epsilon = .001$, iterations $= 20$ - L-BFGS: iterations $= 20$)

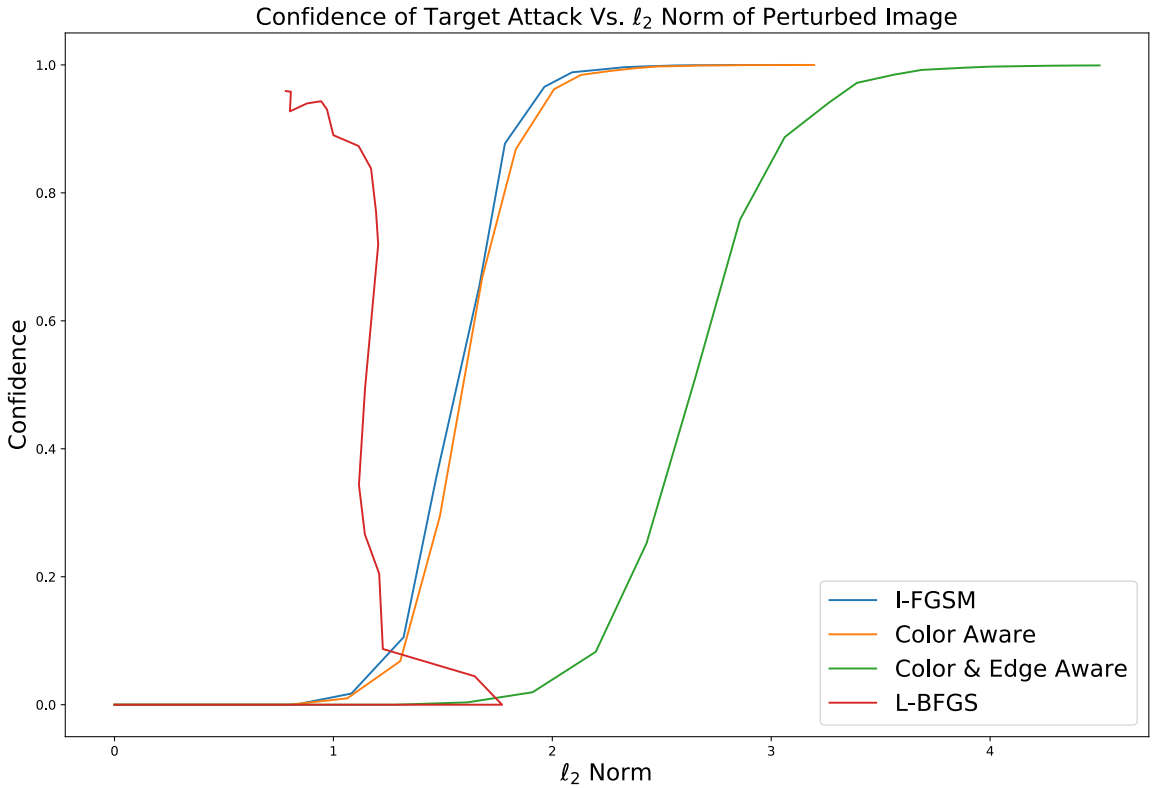

Figure 3.16. Misclassification confidence versus ℓ_2-norm. (Setting: perturbations - CA: $\epsilon = .1$, iterations $= 20$ - C&EA: $\epsilon = 1.5$, iterations $= 20$ - I-FGSM: $\epsilon = .001$, iterations $= 20$ - L-BFGS: iterations $= 20$)

Besides L-BFGS, the other most evident change between the two figures is that Color & Edge Aware has a drastically increased ℓ_2-norm compared to any other method, but this is expected due to the sensitivity of the ℓ_2-norm to outliers However, it should be pointed out that a larger ℓ_2-norm does not mean it is more perceptible to humans. In Figure 3.17 we compare I-FGSM and Color & Edge Aware. In this example, we conduct a targeted attack make an image of an aggressive tank appear as a benign horse cart to the neural network. Both attacks are successful, but note the ℓ_2-norms for each image. The Color & Edge Aware perturbation has a larger ℓ_2-norm; however, the I-FGSM perturbation contains artefacts of the perturbation peeking through in the upper left-hand corner of the image, whereas Color & Edge Aware does not. This highlights how our perturbation method perturbs areas specifically where human vision cannot perceive change.

Original

Classify: Tank
Probability: 0.954

I-FGSM: ℓ_1 Norm: 2016.684
ℓ_2 Norm: 4.489

Classified Class : Horse cart
Probability: 0.998

Color & Edge Aware: ℓ_1 Norm: 1149.846
ℓ_2 Norm: 4.738

Classified Class : Horse cart
Probability: 1.000

Figure 3.17. Perturbation perceptibility with similar ℓ_2-norms. Notice how the upper left portion of the I-FGSM has perturbation artifacts. (Setting: image source - ImageNet, classifier - Inception V3, perturbations - C&EA: ϵ = 5, iterations = 6 - I-FGSM: ϵ = .005, iterations = 3)

3.2.3 Conclusion and Future Work

We have shown two new ways, Color Aware and Color & Edge Aware, to improve adversarial inputs to image classifiers by making perturbation less perceptible to humans. Color Aware converts image RGB data to CIELAB space and adds an ℓ_2-norm across the color channels. These additional constraints assure that the change introduced by the perturbation at each pixel value matches human perception. Color & Edge Aware builds off of Color Aware. We introduced the edge magnitude constraint which limits change in visually smooth, homogenous regions of the image. We then increased the step size for Color & Edge Aware in order to accommodate the extra constraint. When comparing our algorithms to previous perturbation generation techniques, ours is similar in performance and design to FGSM; however, we believe we can implement our ideas with other methods.

There are a number of future directions which can be taken to improve upon this work. First, our general idea of applying perturbations in CIELAB space can be applied to many other perturbation methods, and it remains to be seen how this modification would affect their performance. Second, we also expect that our weighting of the perturbation constraints with respect to edge magnitudes could be applied to other perturbation methods. This could be done for penalized methods as well, thereby making these perturbation techniques less detectable to the human eye. For example one could extend our optimization method to L-BFGS instead of proceeding to the boundary of the constraint set in each iteration. This could have the advantages like line search from L-BFGS but also the advantages that CIELAB space provides.

Another promising area of future work would be to apply our methods to networks that have defense mechanisms in place. Defense distillation is a technique used when training a neural network to make it more robust to adversarial perturbations (Papernot et al. 2016). By introducing perturbations in the training process, the neural network's classifier is more resilient to perturbations. We want to investigate whether Color & Edge Aware's perturbation can fool a neural network that has defense mechanisms included in its training. Color & Edge Aware's edge constraint means no image or classes of images are perturbed the same way, every image is unique. If we pair this with the larger step size that Color & Edge Aware takes, we believe there is a possibility that Color & Edge Aware's perturbation could still fool a neural network whose defense mechanisms were trained with previous perturbation techniques.

Another possible advantage from taking large step sizes with Color & Edge Aware is that the perturbation may survive image compression. Researchers have discovered that successful adversarial perturbations from FGSM with small ϵ values do not cause misclassification after be compressed and recomposed (Dziugaite et al. 2016). The small noise introduced throughout the image is lost in the compression process. Compression of images is a real world issue, and adversaries have to create perturbations that can survive this image manipulation. Because of its large step size, we believe that it is worth investigating to see if Color & Edge Aware's perturbation survives image compression.

THIS PAGE INTENTIONALLY LEFT BLANK

List of References

Allen GC (2019) *Understanding China's AI Strategy: Clues to Chinese Strategic Thinking on Artificial Intelligence and National Security* (Center for a New American Security Washington, DC).

Carlini N, Wagner D (2017) Towards Evaluating the Robustness of Neural Networks. *2017 IEEE Symposium on Security and Privacy (SP)*, 39–57 (IEEE).

Croce F, Hein M (2019) Sparse and Imperceivable Adversarial Attacks. *Proceedings of the IEEE International Conference on Computer Vision*, 4724–4732.

Deng J, Dong W, Socher R, Li LJ, Li K, Fei-Fei L (2009) ImageNet: A Large-Scale Hierarchical Image Database. *CVPR09*.

Dziugaite GK, Ghahramani Z, Roy DM (2016) A Study of the Effect of JPG Compression on Adversarial Images. *arXiv preprint arXiv:1608.00853* .

Fletcher R (2013) *Practical Methods of Optimization* (Hoboken: Wiley), 2nd ed. edition.

Fryer-Biggs Z (2018) Inside the Pentagon's Plan to Win Over Silicon Valley's AI Experts. *Wired* (December 21), https://www.wired.com/story/inside-the-pentagons-plan-to-win-over-silicon-valleys-ai-experts/.

Global News (2018) What is Project Maven? The Pentagon AI project Google Employees Want Out Of (April 5), https://globalnews.ca/news/4125382/google-pentagon-ai-project-maven.

Goodfellow I, Bengio Y, Courville A (2016) *Deep Learning* (MIT press).

Goodfellow IJ, Shlens J, Szegedy C (2014) Explaining and Harnessing Adversarial Examples. *arXiv preprint arXiv:1412.6572* .

Konica Minolta (2018) What is CIE 1976 Lab Color Space? Accessed March 28, 2020, https://sensing.konicaminolta.asia/what-is-cie-1976-lab-color-space/.

Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems*, 1097–1105.

Kurakin A, Goodfellow I, Bengio S (2016) Adversarial Examples in the Physical World. *arXiv preprint arXiv:1607.02533* .

Martín JC, Santos AG (2011) The Influence of Water Repellent Products on the Chromatic Modifications of the Ceramic Brick. *Materiales de Construcción* 61(304):597–611.

Mattis J (2018) Summary of the 2018 national defense strategy of the united states of america. Technical report, Department of Defense Washington United States.

Nebauer C (1998) Evaluation of Convolutional Neural Networks for Visual Recognition. *IEEE Transactions on Neural Networks* 9(4):685–696.

Nguyen A, Yosinski J, Clune J (2015) Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 427–436.

O'Connor T (2017) Russia's Military Challenges U.S. And China By Building A MissileThat Makes Its Own Decisions. *Newsweek* (July 20), https://www.newsweek.com/russia-military-challenge-us-china-missile-own-decisions-639926.

Papernot N, McDaniel P, Goodfellow I, Jha S, Celik ZB, Swami A (2017) Practical Black-Box Attacks Against Machine Learning. *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, 506–519.

Papernot N, McDaniel P, Wu X, Jha S, Swami A (2016) Distillation as a Defense to Adversarial Perturbations Against Deep Neural Networks. *2016 IEEE Symposium on Security and Privacy (SP)*, 582–597 (IEEE).

Paszke A, Gross S, Chintala S, Chanan G, Yang E, DeVito Z, Lin Z, Desmaison A, Antiga L, Lerer A (2017) Automatic D fifferentiation in PyTorch. *NIPS-W*.

Pellerin C (2017) Project Maven to Deploy Computer Algorithms to War Zone by Year's End. *Department of Defense* (July 21), https://www.defense.gov/Explore/News/Article/Article/1254719/project-maven-to-deploy-computer-algorithms-to-war-zone-by-years-end/.

Schanda J (2007) *Colorimetry: Understanding the CIE System* (John Wiley & Sons).

Sharma A (2018) *Understanding Color Management* (John Wiley & Sons).

Szegedy C, Vanhoucke V, Ioffe S, Shlens J, Wojna Z (2016) Rethinking the Inception Architecture for Computer Vision. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2818–2826.

Szegedy C, Zaremba W, Sutskever I, Bruna J, Erhan D, Goodfellow I, Fergus R (2013) Intriguing Properties of Neural Networks. *arXiv preprint arXiv:1312.6199* .

Umbaugh SE (2010) *Digital Image Processing and Analysis: Human and Computer Vision Applications with CVIPtools* (CRC press).

van der Walt S, Schönberger JL, Nunez-Iglesias J, Boulogne F, Warner JD, Yager N, Gouillart E, Yu T, the scikit-image contributors (2014) scikit-image: Image Processing in Python. *PeerJ* 2:e453, ISSN 2167-8359, URL http://dx.doi.org/10.7717/peerj.453.

Wu K (2018) PYTORCH-CW2. GitHub. https://github.com/kkew3/pytorch-cw2.

Zhang R, Zhu JY, Isola P, Geng X, Lin AS, Yu T, Efros AA (2017) Real-Time User-Guided Image Colorization with Learned Deep Priors. *ACM Transactions on Graphics (TOG)* 9(4).

Zhao Z, Liu Z, Larson M (2019) Towards Large yet Imperceptible Adversarial Image Perturbations with Perceptual Color Distance. *arXiv preprint arXiv:1911.02466* .

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