12-2020

A Targeted Adversarial Attack on Support Vector Machine Using the Boundary Line

Yessenia Rodriguez
The University of Texas Rio Grande Valley

A TARGETED ADVERSARIAL ATTACK ON SUPPORT VECTOR MACHINE USING THE BOUNDARY LINE

A Thesis

by

YESSENIA RODRIGUEZ

Submitted to the Graduate College of
The University of Texas Rio Grande Valley
In partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

December 2020

Major Subject: Computer Science

A TARGETED ADVERSARIAL ATTACK ON SUPPORT VECTOR MACHINE USING THE BOUNDARY LINE

A Thesis

by

YESSENIA RODRIGUEZ

COMMITTEE MEMBERS

Dr. Hansheng Lei
Chair of Committee

Dr. Fitratullah Khan
Committee Member

Dr. Mahmoud K. Quweider
Committee Member

December 2020

ABSTRACT

Rodriguez, Yessenia, A TARGETED ADVERSARIAL ATTACK ON SVM USING THE BOUNDARY LINE. Master of Science (MS), December 2020, 38 pp. 7 tables, 17 figures, references, 19 titles.

In this thesis, a targeted adversarial attack is explored on a Support Vector Machine (SVM). SVM is defined by creating a separating boundary between two classes. Using a target class, any input can be modified to cross the "boundary line," making the model predict the target class. To limit the modification, a percentage of an image of the target class is used to get several random sections. Using these sections, the input will be moved in small steps closer to the boundary point. The section that took the least number of steps to cause the model to predict the target class will be considered the optimal section. This method of attack can lead us to find the predominant features that the SVM uses to classify the target class. This knowledge can be used for further attacks and to find further vulnerabilities of the SVM model.

DEDICATION

The completion of my graduate studies would not have been possible without the love and support of my family. My mother, Adela Rodriguez, in particular, inspired, supported me, and pushed me to complete this degree. Thank you for your love and support throughout my whole life and encouraging me to continue to pursue higher education.

ACKNOWLEDGMENT

I will always be grateful to Dr. Hansheng Lei, chair of my thesis committee, for his help and support. He encouraged and guided me throughout the completion of my thesis. I am also thankful to Dr. Fitratullah Khan and Dr. Mahmoud K. Quweider, the members of my thesis committee, for all their support, advise and encouragement. Both Dr. Khan and Dr. Quweider served as my advisors and mentors throughout the years, and I would never have been able to complete this process without their guidance and assistance.

TABLE OF CONTENTS

LIST OF TABLES

Page

LIST OF FIGURES

Page

CHAPTER I

INTRODUCTION

Machine learning is a very popular topic due to its many applications. One main application is using machine learning models for classification. These models mathematically differentiate between classes observed in training data to classify new data. The mathematical approach used by the model changes based on the type of model and the parameters used. Support Vector Machine (SVM) is one such model; it finds a "boundary line" that separates two or more classes the best. The method and how an SVM works are illustrated in **Figure 3**. While these models have many applications and are being implemented, they have many vulnerabilities and have proven to not be very robust. A method to find these vulnerabilities and further study of these models to potentially improve the robustness is by attacking these models in what is known as an adversarial attack. In an adversarial attack, an input is minimally changed, ideally unnoticeable to the human eye. This change causes an incorrect classification or prediction from the model. Training the classification model using these *adversarial inputs* could create a more robust classification model that would be less vulnerable to other forms of attacks. Thus, research in this area and different methods of adversarial attacks have been proposed. There are two main elements to be considered when studying adversarial attacks. First, one must consider how much knowledge is known about the classification model. When nothing is known about the classification model, the attack is known as a *black-box attack*. On the other hand, if everything about the model can be known, the

attack is known as a *white-box attack*. Second element to consider is the exact goal of the attack. All adversarial attacks will lead to a wrong prediction or classification. However, the goal of one type of attack is for the classification model to always predict *one* class. This class is known as the *target class*, and this type of attack is called a *targeted adversarial attack*.

In this thesis, a targeted adversarial attack is proposed that targets the "boundary line" of the SVM, one of its weaknesses. For an SVM model, the goal of an *adversarial attack* is to move the input across the "boundary line," which would cause a misclassification. The attack proposed is tested on an SVM model used for image classification. The attack starts with an image of the *target class*, $class\_t$, which is referred to as t_point. This image will then be used to alter the input images that are not $class\_t$, resulting in the images to be misclassified as $class\_t$ by the model. To limit the change on the input images, t_point is not be used in its entirety, rather only a random section of t_point is used. The input images are modified incrementally into the section of t_point. The section and its size used by the attack contributes to the effectiveness of the attack. Thus, several percentages of t_point and several number of sections are tested to observe how the accuracy of the attack is affected. The attack also records the number of steps taken by each section to cause a misclassification. The section that took the least number of steps is believed to capture the "predominate features" of $class\_t$ or cover the "predominate features" of the original class. Using this assumption, a new model can be trained. Using the attack to recreate the original model or a similar one can lead to the discovery of more vulnerabilities and other attacks. The methodology of the proposed attack and its variation and applications are described in Chapter III – Methodology.

CHAPTER II

REVIEW OF LITERATURE

Much research has been done on adversarial attacks. This research tends to focus on models used for image classification and use either Neural Networks (NNs) or SVMs. Related works are reviewed, considering their similarities and differences along with what contributed to the success of the attack.

2. 1. Adversarial Attacks

Goodfellow et al. proposed an approach to generate adversarial attacks called Fast Gradient Sign Method (FGST) [5][7][10] that adds noise to the image based on the gradient. Other similar methods developed also use the gradient. One such method attempted to make the adversarial example generated more natural [10]. To accomplish this, Generative Adversarial Networks (GANs) method is used to generate noise more specific to the input, making it more natural. GANs, proposed by Goodfellow et al., is an approach used or modified to generate new data, including adversarial examples. One application for GANs outside of an adversarial attack is inpainting as proposed by Pathak [4]. Other studies focus on making less perceptible perturbations rather than natural ones [9] [11]. Zhou et al. proposed an attack in which the attack model that is penalized for altering data samples [9], minimizing the difference between the original and the adversarial example. Similarly, Su et al. limits the changes of the adversarial attack on the original examples [11]. However, Su et al. do this much more than Zhou et al. by limiting the attack to only one pixel. The benefit of this approach is the adversarial example is minimally different, demonstrating the

lack of robustness of the model and a vulnerability that could be easily exploited. However, in real life examples, the success of a limited adversarial attack decreases since the change is not enough to alter the model's prediction. Nevertheless, limited attacks have the advantage of potentially being more unperceivable to the human eye and less costly.

When generating adversarial examples, all these methods attempt to minimize the perturbation, noise, or change added to the image [7][10]. This is generally done by using a loss function and minimizing it in respect to the original and the adversarial example [8]. While this approach may work for models trained for image classification, attacks may generate adversarial examples more unnatural and with a more perceptible change for other media types. When proposing an adversarial attack on speech recognition, Qin et al. focused on audio analysis, its space and how it is perceived, to potentially make the perturbation more undetectable [8]. Their research suggests the idea of adversarial attacks using different approaches based on the type of data to have some merit. Their adversarial attacks were more natural and had smaller perturbation, or less distortion, when compared to another approach for generating audio adversarial examples [8].

Most adversarial attacks impact the input and are considered evasion attacks in which new inputs are perturbated in some minimal way to get the desired goal [1]. Biggio et al. attempt a different approach by modifying the label [3]. Their attack is considered a poison attack in which adversarial examples are introduced to the classification model during training, causing misclassification during testing [1][2]. Also, Biggio et al. attempt this by flipping or changing the label of some of the training samples [3]. While poison attacks can provide information about the classification model, it is not very practical since the likelihood of having access to a model during training is very low [2].

2. 2. Boundary Attack by Brendel et al.

Most of the adversarial attacks reviewed are gradient based, but using only the prediction, or decision, of the model may be more practical. This is what is known as a decision-based adversarial attack. Brendel et al. propose such an attack they refer to as the Boundary Attack [12]. The attack proposed by this thesis shares several similarities to this attack. This attack begins with an adversarial point. For an untargeted attack, the starting adversarial point is any point that is differently classified, whose distance from the original image would be minimized ideally. Perturbation or noise is added to the adversarial point to get it closer to the original image while no change in classification occurs. This is repeated, getting it closer and closer to the original image, until the adversarial point is classified as the original image. An independent and identically distributed Gaussian is used to obtain the perturbation which is projected onto a sphere around the original image. Using this sphere, the adversarial point is moved in one random direction and towards the original image. The sphere and direction change for each perturbation. This process is illustrated in [12, Fig. 2], provided as **Figure 1**.

The performance of the Boundary Attack seems very promising [12]. However, the cost of the attack, according to Brendel et al., is huge. This attack is compared to two other attacks in terms of cost – DeepFool (Rauber et al., 2017, as cited in [12]) and Carlini & Wagner attack (Carlini & Wagner, 2016a, as cited in [12]). Counting both forward and backward passes, DeepFool had 1,199,956 less passes and Carlini & Wagner had 1,168,000 less passes than the Boundary Attack. This is likely the result of restricting the attack to only use the prediction, or decision, of the model, and a similar occurrence can be observed in the attack proposed in this thesis. Additional number of steps needed may be attributed to the starting point being the adversarial point, not the original image. The final adversarial point needs to be as close to the

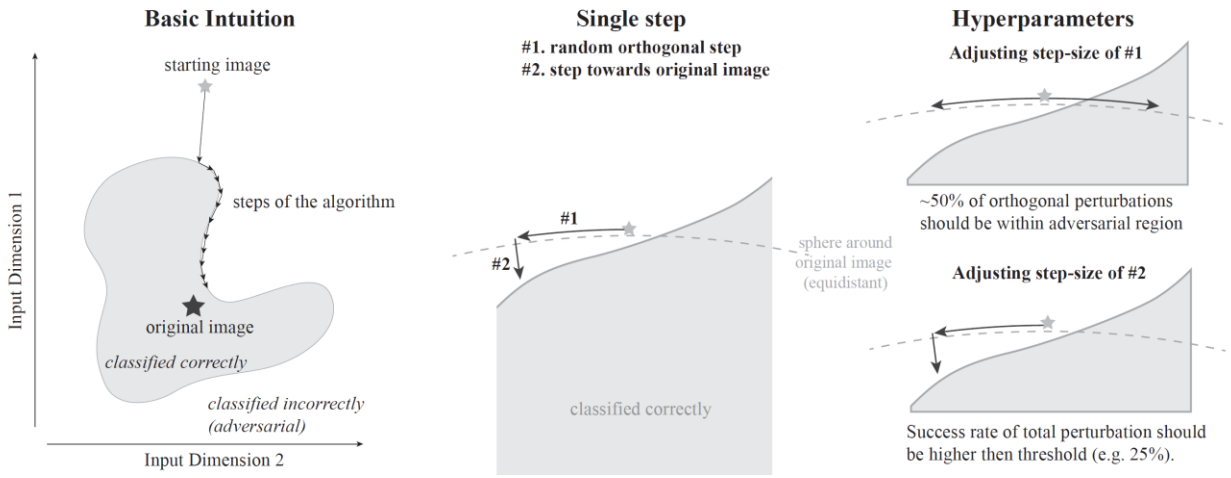

Figure 1. General Methodology of the Boundary Attack, obtained from [12]. This attack is proposed by Brendel et al.

original image as possible. For this reason, the starting adversarial point will need to undergo many changes to get it as close to *a differently classified image*, the original image. For this reason, the perturbations are performed on the original image in the attack proposed in this thesis, potentially reducing the number of perturbations needed. The perturbations performed are also different. The original point is moved closer to the target, or adversarial point, as well but in a more direct manner. Granted, this may potentially lead to taking a "longer route" or having a larger perturbation. To minimize the perturbation, only a random section of the point is perturbated. Thus, unlike in the Boundary Attack, the change is potentially more limited and ideally only on the predominate features of the original class. This is further explained in Chapter III - Methodology. Additionally, the Boundary Attack is applicable to both targeted and untargeted attacks. In this thesis, the attack is limited to a targeted attack. In a targeted attack, the starting adversarial point is a point belonging to the target class, which is used to guide the attack to the desired class. However, in an untargeted attack, the starting adversarial point could be any point that is not the original class. Thus, the success or the cost of the attack is dependent on the starting adversarial point selected. The process of this selection, according to Brendel et al., is to select a point in

which the distance $d(\boldsymbol{o}, \tilde{\boldsymbol{o}}) = \|\boldsymbol{o} - \tilde{\boldsymbol{o}}\|_2^2$ is minimized. However, the selection is based on the sample of images the adversarial point is selected from. To avoid any potential issues that could arise from incorrectly selecting the starting adversarial point, only a targeted attack is implemented in this thesis.

Jing et al. propose a guided decision-based attack that also shares similarities with the Boundary attack, but the starting adversarial point is obtained differently, attempting to reduce the cost of the attack [13]. They propose to not work with all the starting adversarial point, only a relevant component. This component is added to the original image, creating a new adversarial point. This reduces the cost of the attack to about 60%. The difference between the two attacks is illustrated [13, Fig. 1], which is provided as **Figure 2**. In this thesis, the perturbation is limited by working with only a portion of the adversarial point. However, Jing et al. work with a semantically relevant component, such as the head of a Persian cat in **Figure 2**. In this thesis, a random section is used instead. Granted, the ideal section is still searched for, but this section could contain non-semantically relevant pixels. The classification models are mathematically based, not semantically based. Thus, changing a portion of the input regardless of its semantic significance could still cause the model to misclassify it. This potential vulnerability is exploited by the attack proposed in this thesis.

2. 3. Defense Strategies

Like adversarial attacks, defenses against these attacks are also a popular area of research [1]. One common defense is referred to as adversarial training. Adversarial training is when inputs or adversarial examples are used to train, or re-train, the classification model [5]. Goodfellow et

Figure 2. Boundary Attack (Brendel et al.) vs Jing et al. Attack, obtained from [13]. Both attack methods are used to get close to the original image and can be compared in regards to the starting point and number of calls.

al. use their FGST for adversarial training and found that adversarial training worsens underfitting. Additionally, training a model for all possible inputs is not practical or realistic. Thus, other methods for defense or addressing these models' vulnerabilities are needed. Bhagoji et al. propose reducing the dimensionality of data to defend against adversarial attacks [1]. Classification models have a harder time finding the *predominate features* when working with high-dimensional data, making them vulnerable to adversarial attacks. It is found by Bhagoji et al. that the success of the attack decreases the more the dimension is reduced. However, the dimensionality reduction leads to a drop in accuracy based on the amount of reduction, making the defense counterproductive in certain cases.

Furthermore, there do exist several other defensive attacks. The effectiveness of them depend on the type of attack and the model being attacked. For gradient-based attacks, which are a popular type of adversarial attack, any mask or method used to hide the gradient could be used; one example is adding non-differentiable elements through saturated non-linearities [12]. A similar approach can also be used for other types of attacks like score-based attack, which use the predicted numerical values for guiding the attack. Additionally, the access to the model being attacked may

be limited. Thus, working with as little information from the model as possible may lead to better adversarial attacks in a practical setting.

CHAPTER III

METHODOLOGY

3. 1. Support Vector Machine

In Chapter I. Introduction, SVMs are introduced which is a machine learning model that can be used for classification. An SVM model differentiates two or more classes by finding the "boundary line" that separates them. This method is illustrated in **Figure 3**. The margin is the space between the classes, and the vector or line used to differentiate the classes, which is the center of the margin, is called the hyperplane. Depending on the number of classes, an SVM model may find more than one hyperplane, or "boundary line." The ideal hyperplane maximizes the margin and completely separates the two classes. However, the hyperplane may not account for all instances, and a class may have some points on the other side of the hyperplane, leading to misclassification. The hyperplane and margin of an SVM is vulnerable to a decision-based attack, more so than other machine learning models.

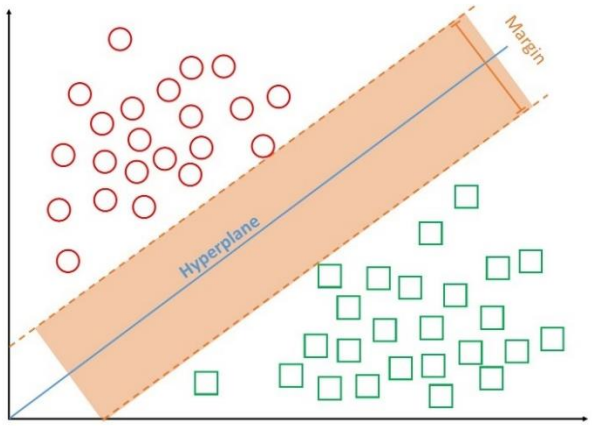

Figure 3. SVM Methodology. Two classes (circles and squares) separated by a hyperplane with a maximum margin.

This is mainly because the hyperplane can be easily found by moving an input in one direction, towards a target class for example. For a misclassification when using an SVM model, the objective is to cross the hyperplane. Additionally, finding the hyperplane can lead to potentially recreating the original SVM model, potentially leading to more attacks. This is the fundamental idea behind the attack proposed in this paper that is explained in Chapter III. Methodology – 3. 3. Section Boundary Attack.

When testing the proposed attack, an SVM model is created using MNIST dataset which is a large database of hand written digits [14]. This SVM model is a polynomial SVM. An SVM can have different kernels, or set of mathematical functions, that are used to find the hyperplane. Some kernels are linear, polynomial, radial basis function (RBF), and sigmoid. The data of the classes cannot always be separated by a line, as is shown in **Figure 4**. Thus, using different methods or kernels can allow for a more accurate hyperplane and a greater margin, resulting in a more accurate model. The polynomial SVM created for testing has an original accuracy of 98%.

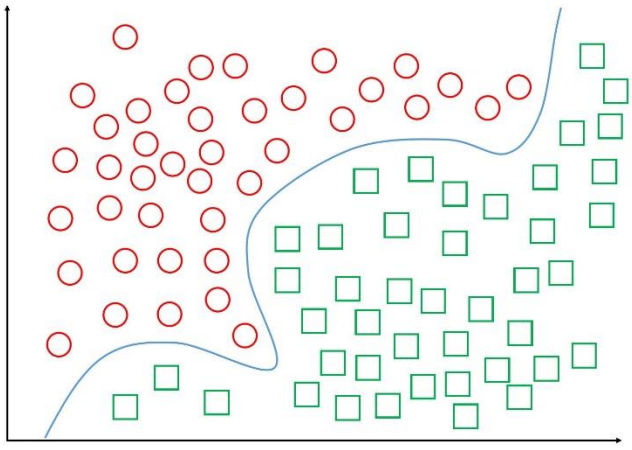

Figure 4. A Nonlinear SVM. Two classes (circles and squares) cannot be separated by a straight line, but a curved and irregular one.

956	0	2	0	0	4	2	1	2	0
0	1127	2	1	0	0	3	0	2	0
6	2	1006	0	2	1	3	9	3	0
0	2	3	984	0	6	0	6	5	4
1	0	3	0	966	0	4	0	0	8
2	0	0	8	1	869	4	1	5	2
4	4	2	0	3	6	937	0	2	0
0	15	8	1	1	0	0	995	0	8
1	1	2	5	5	5	0	3	949	3
3	6	1	4	14	5	0	6	1	969

Figure 5. Confusion Matrix for Original SVM

3. 2. Random Adversarial Sectioning

As previously stated, an input is moved in one direction towards the target class until crossing the hyperplane. However, this can lead to a great amount of change in the original input. To minimize and limit the effect of the attack, only random sections of the input are changed. This allows one to find the predominate features of the class the original input belongs to and the predominate features of the target class. The change is applied using an image that belongs to the target class. Random sections of this target class image are obtained. The number of sections obtained potentially affect the success of the attack. Also, the size of these sections can affect the success of the attack and the amount of change the original input undergoes. To test the effect of the number of sections, s, on the attack, experiments are conducted using 5, 10, 20, and 40 sections. To test the effect of the size of the sections, p, on the attack, experiments are conducted using 10%, 30% and 50% of the target class image.

To obtain a random section of the adversarial attack, the size of the sections, p, and the target class image, t_point, are used. The method is provided as **Algorithm 1**. The section obtained is a group of random adjacent pixels, starting with a random pixel, not a set shape. The section is actually a mask. The value of this mask is either 0 or 1, depending on whether the

corresponding pixel in t_point is part of the section or not. This mask is initiated to be the size of t_point and all its values are set to 0. Then, a random pixel is selected from t_point, which is referred to as r_pixel. Using the coordinates of r_pixel, the corresponding value, or pixel, in $mask$ is found and set to 1. Then, a new pixel is selected randomly from the adjacent pixels surrounding the non-zero pixels in $mask$ as is shown in **Figure 6c**. This new pixel is referred to as a_pixel. a_pixel is added to $mask$ similarly to how r_pixel is. A new a_pixel keeps on being selected and added until the section is of the desired size. This process is shown in **Figure 6d**. To keep track of the number of pixels that should be added to $mask$, a value called $sizeS$ is obtained using Formula 1, in which $sizeT$ refers to the size of t_point. The full process is shown in **Figure 6**.

$$sizeS = sizeT \times p \tag{1}$$

Algorithm 1 Generating Adversarial Section

Input: target class image t_point, size of section p

Output: random adversarial section $section$

$sizeS = t\_point.size * p$
Set r_pixel as a random value or pixel of t_point
Set $mask$ to be the same size as t_point with all the value being 0
$index = t\_point.index(r\_pixel)$ #coordinates
$mask[index] = 1$
$sizeS = sizeS - 1$
while $(sizeS > 0)$ **do**
    Set a_pixel to be a pixel or value of $mask$ that is an adjacent pixel or value of the nonzero values in $mask$
    $index = mask.index(a\_pixel)$
    $mask[index] = 1$
    $sizeS = sizeS - 1$
end while
return $mask$

This process is repeated s times to obtain the number of sections, or masks, desired. Sample random sections obtained from using the same t_point are provided as shown in **Figure 7**. In **Figure 7**, the corresponding pixels of t_point are shown that were obtained using the masks generated. For the generated sections, some may contain the predominate features of the target class while others may not. To find the ideal or optimal section, the sections generated are compared by considering the steps needed to cause a misclassification. The section with the least steps, or change to the original input, is the best section. These sections are used to recreate the SVM as is described in Chapter III – Methodology 3. 4. Recreate Original SVM.

3. 3. Section Boundary Attack

Once the section is obtained, an input, i, can be perturbated. The difference between the

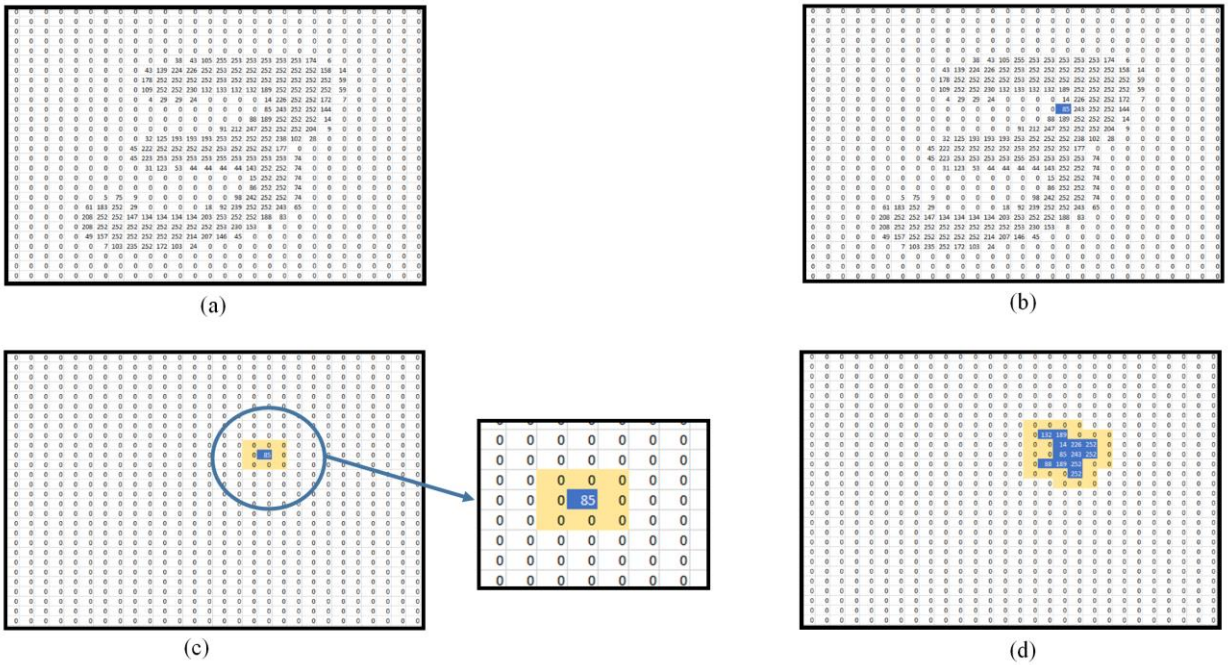

Figure 6. Random Adversarial Sectioning. (a) The target class image and its pixel values: In this case, the target class is the number *3*. (b) The target class image and one random pixel selected, r_pixel. r_pixel is shown in blue to differentiate it from the other pixels in the target class image. (c) The surrounding pixels of r_pixel are shown and highlighted in light yellow. From these pixels, one random pixel is selected, a_pixel. (d) This process is repeated until the size desired is reached. A greater section, that is still not complete, is shown with a blue background, and the adjacent pixels of the section are highlighted in light yellow.

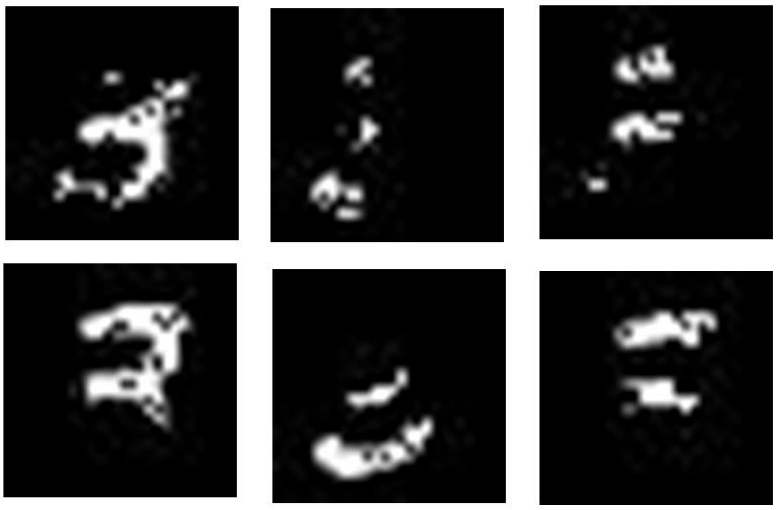

Figure 7. Random Adversarial Sections. These are some examples of random adversarial sections generated using the same t_point when the target class is the number *3* and the size of the section is 30% of the original image, t_point.

section of t_point obtained using the $mask$ generated and i is considered the direction. i is moved in the said direction in small steps, with the size of the step being θ, which is defined by **Formula 2**. The potential adversarial point is obtained using **Formula 3**. To initiate $adPoint$, it is set to i. In **Formula 3**, Δi is the step with the direction and size, θ. This is defined in **Formula 4**. To limit the perturbation to only the desired section, the value of the mask is multiplied with the direction and step. If the pixel should not be permutated, it is not part of the section, the mask value for that pixel should be 0, resulting in zero change when multiplied. Otherwise, a perturbation occurs since the mask's value is 1. Additionally, the perturbation is limited to a small step by using θ. **Formula 3** is implemented until the decision of the SVM model changes to the target class, referred to as $class\_T$ or θ becomes 1.

$$\theta = \frac{1}{N}, with\ N\ being\ a\ whole\ number \tag{2}$$

$$adPoint = adPoint + \Delta i \tag{3}$$

$$\Delta i = \theta \overline{mask_j}\left(\overline{t\_point_j} - \overline{i_j}\right) \tag{4}$$

The full attack is described in **Algorithm 2**. The number of times Formula 3 is implemented is considered the number of steps taken by the attack, referred to as k in **Algorithm 2**. To test all the sections generated, $S = (mask_1, mask_2, \dots, mask_s)$, and find the optimal section, **Algorithm 2** is implemented for each section. After all the sections have resulted in a misclassification, the number of steps each section took is compared, and the section with the least number of steps is considered the optimal section. A section in S is referred to as $mask_\alpha$, and the steps $mask_\alpha$ took is referred to as $step_\alpha$. For this process, **Algorithm 3** is used, which calls **Algorithm 2**. Granted, the process of obtaining random sections and finding the optimal one leads to calling on the SVM many times and adds time to the attack. To limit this, the attack is trained for a specific SVM model and using a specific t_point. After the "training data," or inputs, for which **Algorithm 3** is applied, all the optimal sections are averaged together to form one universal section. This universal section should ideally cause any input to be classified $classT$ in the least number of steps. The section or mask consists of zeros and ones. To average it, all the masks are added and divided, resulting in numbers between 0 and 1. These values are now used as weights as is shown in **Formula 5**. This process is described in **Algorithm 4**. For **Algorithm 4**, **Algorithm 3** is modified slightly to not only return the adversarial example with the least amount of perturbation, called $adPoint_i$ in **Algorithm 4**, but also the section used to create that adversarial example, which is referred to as $mask_i$. Using the universal section generated, all other inputs are perturbated to cause the inputs to be classified as $class\_t$. **Formula 5**, provided below, shows how the universal section generated is used to perturbate additional inputs.

$$adPoint = x + sectionUniv(t\_point - x) \tag{5}$$

Algorithm 2 Generating an Adversarial Example

Input: the decision function $f(.)$, the original point or image i, the adversarial section $section$, the adversarial mask $mask$, the target class $class\_t$

Output: an adversarial example $adPoint$, the steps taken k

1: Set the step or size of perturbation θ (e.g. $\theta = 0.01$)
2: $\Delta i = \theta(section - i)$ for $mask$ non-zero values
3: $adPoint = i$
4: $k = 0$:
5: **while** $(k < N)$ **do**
6: $adPoint = adPoint + \Delta i$
7: **if** $f(adPoint) = class\_t$ **then**
8: return $adPoint, k$
9: **end if**
10: $k = k + 1$
11: **end while**

Algorithm 3 Generating a Cost-Effective Adversarial Example

Input: the decision function $f(.)$, the original point or image i, list of adversarial sections S, the target class $class\_t$

Output: an adversarial example with the smallest number of steps $min\_adPoint$

1: $minSteps = N$ //obtained from θ formula
2: $min\_adPoint = i$
3: **for** $mask_\alpha$ in S **do**
4: Call algorithm 2 to generate adversarial example for i using $mask_\alpha$
5: Set $step_\alpha$ to be the number of steps returned by algorithm 2
6: Set ad_point_α to be the adversarial example returned by algorithm 2
7: **if** $minSteps > step_\alpha$ **then**
8: $minSteps = step_\alpha$
9: $min\_adPoint = ad\_point_\alpha$
10: **end if**
11: **end for**
12: return $min\_adPoint$

In **Formula 5**, x is the input being perturbated, and $sectionUniv$ is the universal section generated. The difference between t_point and x is still calculated to get the correct direction for each input. **Formula 5** is applied when x does not belong to the class $class\_t$.

Using a universal section, execution time of the attack is decreased, but may negatively affect the accuracy and effectiveness of the attack. Averaging an image tends to create blurry images, which may lead to a section with unclear predominate features of $class\_t$. Additionally, the change to the input is greater and may extend to the whole image. To improve this approach. **Algorithm 5** is created. In **Algorithm 5**, the optimal sections for one class are averaged together, resulting in a class section. Thus, for each class in the "training data" that is not $classT$, a class section is obtained. Using **Algorithm 5**, the attack, after being trained, would only need to call on

the SVM once to obtain the original class of the input to know which class section to use. Once the class section is known,

the new input is perturbated using **Formula 5**, with the class section as $sectionUniv$.

3. 4. Recreating Original SVM

As previously mentioned, this attack uses random adversarial sections of the target class, $class\_t$, to perturbate new inputs. It is believed that these random adversarial sections

Algorithm 4 Generating a Universal Adversarial Section for Attack

Input: the decision function $f(.)$, list of training inputs I, list of adversarial sections S, the target class $class\_t$

Output: an adversarial universal section $sectionUniv$

1: Set $sumSection$ to be the same size as t_point with all the values being 0
2: **for** i in I **do**
3: Call algorithm 3 to generate adversarial example for i using sections S
4: Set $section_i$ to be the optimal section returned by algorithm 3
5: Set $adPoint_i$ to be the adversarial example returned by algorithm 3
6: $sumSection = sumSection + section_i$
7: **end for**
8: $sectionUniv = sumSection/(size\ of\ I)$
9: return $sectionUniv$

Algorithm 5 Generating Class Adversarial Section for Attack

Input: the decision function $f(.)$, list of training inputs I, list of adversarial sections S, the target class $class\_t$

Output: a dictionary containing the adversarial class section $sectionClass$

1: Set $sectionClass$ to be a dictionary with the size of 10 with the keys being the class labels (e.g. 0, 1, etc.) and the values being empty sections - the same size as t_point with all the values being 0
2: Set $countClass$ to be a dictionary with the size of 10 with the keys being the class labels (e.g. 0, 1, etc.) and the values being all 0
3: **for** i in I **do**
4: Call algorithm 3 to generate adversarial example for i using sections S
5: Set $section_i$ to be the optimal section returned by algorithm 3
6: Set $adPoint_i$ to be the adversarial example returned by algorithm 3
7: $key = f(i)$
8: $sectionClasss[key] = sectionClass[key] + section_i$
9: $countClass[key] = countClass[key] + 1$
10: **end for**
11: Divide all
12: return $sectionClass$

contain the predominate features of $class\_t$ or cover the predominate features of the input's class. Granted, the random adversarial section may contain other elements that do not greatly identify $class\_t$ or the class of the input. Therefore, the optimal section, the section that requires the least amount of change to the input is believed to be the section of interest. This section likely results in a misclassification sooner because it contains the predominate features of $class\_t$, or it corresponds to and masks the predominate features of the class the input belongs to. Using this assumption, the original SVM can be recreated. The input and the section of the input that corresponds to the optimal adversarial section is used as training data to classify the class they belong to. Furthermore, the original adversarial image, t_point, and the optimal section is used as training data to classify the target class, $class\_t$. In addition, the adversarial example generated using the optimal section is used as training data as well for $class\_t$. This approach allows one to use a new set of data to recreate the original SVM or create a similar SVM.

Recreating the original SVM allows for a wider range of attacks. It is discussed in Chapter II. Review of Literature – 2. 2. Boundary Attack by Brendel et. al that using only the decision of an SVM is more practical. This is because, generally, only the decision of a classification model is available. Thus, being able to recreate the original model can be very useful. Having access to the complete model allows the application of more attacks such as gradient attacks, which are very common and effective adversarial attacks.

CHAPTER IV

RESULTS

In this chapter, the results of several experiments conducted for the algorithms and the Section Boundary Attack are provided and described. The effectiveness of the attack is evaluated based on the accuracy of the SVM. The attack and algorithms are also evaluated based on execution time and the number of calls made to the SVM model. These experiments are conducted using the MNIST dataset [14] and the number *3* as the target class. For all these experiments, the step taken is $\theta = 0.01 = 1/N = 1/100$.

4. 1. Section Boundary Attack

In this section, the proposed boundary attack is implemented without using training data or obtaining a universal section or class section. Thus, for these experiments, **Algorithm 3** is used. The number of sections, s, and the size of the sections, p, are changed to observe the effect on the attack as previously mentioned in Chapter III. Methodology – Random Adversarial Sectioning. Experiments are conducted using 5, 10, and 20 as the value for s and 10%, 30% and 50% as the value for p. The result of these experiments is provided in **Table 1**, **Table 2**, **Figure 8**, **Figure 9**, and **Figure 10**.

In **Table 1**, one can observe the accuracy of the SVM and how the Section Boundary Attack affected its performance. The original SVM model has an accuracy of 98%. As can be seen in **Table 1**, the accuracy decreases by at least 7%, for $s = 10$ and $p = 10\%$, and decreases to a

maximum of 59%, for $s = 20$ and $p = 50\%$. The number of sections, s, does not appear to greatly impact the accuracy of the SVM. However, its effect on the runtime is significant, increasing it by about 10 seconds as can be observed in **Table 3**. The great increase on time is likely because the attack, **Algorithm 3** is repeated for each section, adding time to the overall attack. From **Table 2**, the calls made by the optimal section, the section requiring the least number of steps, is about 80 overall. Thus, for each section the SVM model is called about 80 to N times regardless of the size of the section, p, or the number of sections, s. Based on this, it is concluded that it is the number of sections, s, that truly affects the running time. The size of the sections with respect to the target class image, p, greatly affects the accuracy of the attack as can be observed in **Table 1**. When $p = 10\%$, the accuracy of the SVM does not decrease greatly, but it decreases by half when $p = 30\%$ and when $p = 50\%$. This is likely because the change made to the image is on a greater scale – a greater portion of the image. Granted, the section should still

Accuracy of SVM After Attack			
	$p = 10\%$	$p = 30\%$	$p = 50\%$
$s = 5$	86%	54%	49%
$s = 10$	91%	52%	48%
$s = 20$	87%	54%	39%

Table 1. Accuracy of SVM After Attack. The accuracy of the SVM when the attack is implemented is provided. This allows one to determine the effect on the accuracy, yielding the accuracy of the attack. s is the number of sections tested, and p is the size of the sections with respect to the target class image.

Average Calls for Optimal Sections			
	$p = 10\%$	$p = 30\%$	$p = 50\%$
$s = 5$	88.79	82.94	78.61
$s = 10$	87.86	82.01	76.29
$s = 20$	86.77	77.88	72.79

Table 2. Average Calls for Optimal Sections. The average calls, or iterations, for the attack for # of inputs is provided. The maximum number of calls possible is N. s is the number of sections tested, and p is the size of the sections in respect to the target class image.

Execution Time of Attack			
	$p = 10\%$	$p = 30\%$	$p = 50\%$
$s = 5$	13 sec	13 sec	13 sec
$s = 10$	22 sec	22 sec	22 sec
$s = 20$	32 sec	34 sec	33 sec

Table 3. Execution Time of Attack. The average execution time, or runtime, for the attack for # of inputs is provided. The execution time is rounded to the nearest second. s is the number of sections tested, and p is the size of the sections with respect to the target class image.

952	0	0	25	0	2	0	0	1	0
0	1042	0	82	0	1	1	0	9	0
4	0	769	251	2	0	1	2	3	0
0	0	0	1007	0	0	0	0	2	1
2	0	5	20	944	1	2	0	4	4
8	0	2	422	1	448	4	0	7	0
6	2	3	24	4	2	915	0	2	0
1	7	9	183	2	0	0	815	2	9
3	0	1	98	3	2	0	1	866	0
4	6	2	46	19	3	0	3	3	923

Figure 8. Confusion Matrix for Attacked SVM with $s = 20$ and $p = 10\%$

876	0	3	83	0	0	14	0	1	3
0	15	1	937	0	0	1	0	181	0
1	0	428	593	3	0	4	0	3	0
0	0	0	1010	0	0	0	0	0	0
1	0	1	141	773	0	16	0	48	2
3	0	1	722	2	143	1	0	19	1
0	0	1	267	1	1	683	0	5	0
2	0	3	906	4	0	1	64	18	30
1	0	0	277	0	0	0	0	690	0
2	0	5	251	11	0	2	0	50	688

Figure 9. Confusion Matrix for Attacked SVM with $s = 20$ and $p = 30\%$

707	0	18	134	0	0	0	0	35	86
0	0	0	1059	0	0	0	0	76	0
0	0	366	356	1	0	0	0	7	2
0	0	0	1010	0	0	0	0	0	0
0	0	9	475	159	0	0	0	32	307
0	0	0	736	0	55	0	0	96	5
0	0	65	334	4	0	405	0	113	33
0	0	1	976	0	0	0	0	7	44
0	0	0	329	0	0	0	0	645	0
0	0	1	452	0	0	0	0	3	553

Figure 10. Confusion Matrix for Attacked SVM with $s = 20$ and $p = 50\%$

not be too large to minimize the amount of change done on the input. The effect of p on the runtime, however, is very minimal that it does not significantly affect the runtime as is shown in **Table 3**.

In addition, the goal of the attack is to cause misclassification, specifically for all inputs to be classified as the target class, which is the number *3* in this case. **Figure 8**, **Figure 9**, and **Figure 10** are the confusion matrices for the original SVM when testing the adversarial examples generated for $s = 20$. Based on the figures, the attack is more successful for some classes, numbers. Additionally, the attack misclassified on various occasions, but not by classifying as the target class, number *3*. This can be observed in the case of class 4, number *4*, which the attack leads to a classification of class 9, number *9*, for most of its inputs in **Figure 10**. This implies class 9, number *9*, is in between the target class and class 4, number *4*. Thus, when moving inputs belonging to class 4, number *4*, in the direction of the target class, the inputs move into and must cross class 9, number *9*, leading number *4* to be classified as number *9* as a result. Finally, the classes with the least misclassification are class 0, number *0*, and class 8, number *8*. It is likely the predominate features of these two classes are very distinct, making it harder to cause misclassification. The classes with the most misclassification are class 1, number *1*, class 5,

number *5*, and class 7, number *7*. It is likely that these classes are the closest to the target class, making them more susceptible to the attack. Using the figures, it can be concluded that about 90% of the misclassifications are incorrectly classified as the target class. When observing the confusion matrices for $s = 5$, which are provided as **Figure 11, 12, and 13**, the percentage of misclassifications classified as the target class varies more, depending on the size of the section, p. Therefore, one can conclude that the number of sections, s, does affect the accuracy of the attack in relation to classifying inputs as the target class.

While conducting these experiments, the average time to obtain a random adversarial

838	0	2	20	1	7	63	0	42	7
0	1086	0	7	17	0	6	0	19	0
1	1	803	112	8	0	10	0	95	2
0	0	1	1003	2	0	0	0	3	1
0	0	1	1	969	1	7	0	0	3
1	1	0	134	6	691	14	0	39	6
2	2	0	4	4	2	935	0	9	0
0	3	14	46	70	0	0	482	22	391
0	0	0	51	11	0	0	0	911	1
2	0	0	17	65	2	1	0	12	910

Figure 11. Confusion Matrix for Attacked SVM with $s = 5$ and $p = 10\%$

857	0	1	21	0	0	0	0	23	78
0	3	0	839	0	0	0	14	278	1
0	0	577	399	2	0	1	0	42	11
0	0	0	1006	0	0	0	0	2	2
1	0	9	43	417	0	0	0	5	507
2	0	0	404	1	102	0	0	201	182
20	0	34	35	39	1	575	0	198	56
0	0	3	716	1	0	0	86	3	219
0	0	0	48	0	0	0	0	909	17
0	0	1	106	0	0	0	0	0	902

Figure 12. Confusion Matrix for Attacked SVM with $s = 5$ and $p = 30\%$

814	0	0	57	0	0	0	0	47	62
0	0	0	982	0	0	0	0	152	1
1	0	428	562	1	0	0	0	33	7
0	0	0	1007	0	0	0	0	1	2
1	0	3	186	323	0	0	0	8	461
0	0	0	533	2	91	0	0	172	94
13	0	16	117	28	0	497	0	204	83
0	0	1	847	0	0	0	6	3	171
0	0	0	89	1	0	0	0	862	22
0	0	1	154	0	0	0	0	0	854

Figure 13. Confusion Matrix for Attacked SVM with $s = 5$ and $p = 50\%$

section is 0.1023 seconds. The time to generate the sections does not change greatly based on the size of the section, p. Additionally, the time to generate the sections is not added to the runtimes provided in **Table 3**. Thus, a greater number of sections, s, needs additional time to generate more random adversarial sections. In these experiments, the largest value of s used is 20, adding about 2 seconds to the runtime for generating the sections.

4. 2. Universal Section Attack

In this section, a universal section is generated and implemented as is described in **Algorithm 4** in Chapter III. Methodology – 3. 3. Section Boundary Attack. The number of sections, s, and the size of the sections, p, are 5 and 30%, respectively. The value for both is determine based on the results from the experiments described above in Chapter IV. Results – 4. 1. Section Boundary Attack, specifically considering the running time. For these experiments, two sets of data are required – training data and testing data. This data is obtained from the testing data [14]. Different number of training data is used to observe its effect on the accuracy. The training data contained 5,000 samples, 100 samples, and 500 samples. The testing data used

to test the universal section consists of 5,000 samples for all the different collection of training data. The result of these experiments is provided in **Table 4** and **Figure 14**.

As can be observed in **Table 4**, the execution time for the attack decreases from 128,604

Size of Training Data	Accuracy	Execution Time	Training Time
5,000	95%	30,534.69 sec	30,452.28 sec
100	55%	722.77 sec	634.25 sec
500	89%	3,161.43 sec	3,080.14 sec

Table 4. Universal Section Accuracy and Execution Time

374	0	4	34	6	1	0	0	39	62
0	0	2	381	0	0	0	0	181	0
0	0	376	105	1	0	1	0	10	0
0	0	0	504	0	0	0	0	3	3
0	0	2	17	353	0	0	0	21	89
0	0	1	200	11	37	0	0	106	81
10	0	129	28	73	0	165	0	88	3
0	0	5	281	1	0	0	20	9	200
0	0	0	11	0	0	0	0	469	5
1	0	0	20	6	0	0	0	5	457

Figure 14. Confusion Matrix for Universal Attacked SVM when 100 Samples are used

seconds, when no training is done, to 30,534.69 seconds when $s = 5$ and $p = 30\%$ and a universal section is used, decreasing about 37% when the complete testing data [14] is used. Additionally, only 88.52 seconds is used to generate adversarial examples for 5,000 samples and the accuracy decreases to half when only using 100 samples for training. Thus, training the Section Boundary Attack leads to a faster attack. Regarding the accuracy of the attack, it is about the same when training consists of 100 samples. However, using 500 or 5,000 samples greatly decreases the effectiveness of the attack, with the accuracy of the SVM decreasing by only 10% and 4%, respectively. This is likely because the universal section is an average of all the directions possible to reach the target class. When an image is averaged together, it becomes blurry [4]. This also

happens with the universal section, blurring the predominate features of the target class. Based on **Figure 14**, one can observe the attack is more successful for class 1, number *1*, class 5, number *5*, and class 7, number *7*. This corresponds with the results documented in Chapter 1V. Results – 4. 1. Section Boundary Attack. Additionally, some classes, like class 6, have large misclassification, but the misclassification is not a result of being classified as the target class. This can also be observed in class 5 and class 7. These two classes are mostly classified as the target class but are also misclassified as other classes by a great number of samples.

4. 3. Class Section Attack

In this section, **Algorithm 5**, described in Chapter III. Methodology – 3. 3. Section Boundary Attack, is implemented. For these experiments s and p are 5 and 30%, respectively, as is done in the previous experiments. Additionally, the size of the training data used to generate the class section is set to 100, 500, and 5,000, as is done in the previous experiments, to also observe its effect on accuracy. These similarities between the previous experiments and the experiments discussed in this section are implmented to be able to accurately compare the two approaches. The accuracy for all the classes is provided in **Table 5**. The data provided in **Table 6** document the accuracy of each class for the original SVM. The confusion matrix for the SVM under the attack proposed as **Algorithm 5** is provided as **Figure 15**.

Based on **Table 5** and **Table 6**, the accuracy of the attack increases compared to when a universal section is used, going from 99% to 46% for the best results of the attack observed during experiments. In comparison, the universal section causes the accuracy of the SVM go from 99% to 55% in the best results observed. The effect of the class sections on the classes' accuracy seems to fluctuate. For example, class 8, number *8*, seems to not greatly be affected by the class section boundary attack when only 100 samples are used to generate the class sections. However, when

500 samples are used, a much greater decrease in accuracy for class 8 can be observed. On the other hand, the accuracy for class 4, number *4*, and class 6, number *6*, increases with the increase of samples used to generate the class sections. This difference in performance likely has to do with how close the classes are to the target class and the number of samples for each class contained in the training data. For this reason, the class distribution for the training data is provided in **Table 7**. According to **Table 7**, class 8 has 2 samples when only 100 samples are used for training. Thus, the class section for class 8 is generated using only 2 samples. When 500 samples are used for training, class 8 has 40 samples. The addition of samples likely contributed to the success of the attack. This can also be observed with other classes.

In the experiments described in Chapter IV. Results – 4. 1. Section Boundary Attack, the number 0 and 9 classes has the least misclassification observed. These two classes also have less misclassification than other classes, particularly class 0, number *0*. For the overall attack's goal, to classify all inputs as the target class, the implementation of training and class sections lead to 53.72% misclassification. 81.5% of misclassifications are classified as the target class, so the overall goal is met. Granted, there are many instances of misclassification in which the input

Class	Accuracy – 100 Samples	Accuracy – 500 Samples	Accuracy – 5,000 Samples
Number 0	87%	72%	99%
Number 1	0%	0%	81%
Number 2	69%	45%	97%
Number 3	99%	100%	99%
Number 4	16%	79%	95%
Number 5	0%	5%	89%
Number 6	11%	78%	98%
Number 7	0%	0%	97%
Number 8	96%	19%	99%
Number 9	96%	66%	96%
Total	47%	46%	95%

Table 5. Class Section Attacked SVM Accuracy

Class	Accuracy
Number 0	99%
Number 1	99%
Number 2	98%
Number 3	98%
Number 4	99%
Number 5	97%
Number 6	99%
Number 7	98%
Number 8	99%
Number 9	99%

Table 6. Accuracy of Original SVM per Class

Size of Training Data	Class 0	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7	Class 8	Class 9
5,000	466	593	530	497	510	465	452	507	478	502
100	8	14	8	11	14	7	10	15	2	11
500	42	68	55	43	56	52	41	51	40	52

Table 7. Class Distribution for Training Samples

372	0	116	17	0	11	0	1	3	0
0	0	0	559	0	1	0	0	4	0
0	0	226	273	0	0	0	0	2	1
0	0	0	509	0	0	0	0	0	1
0	0	2	65	381	0	0	3	3	28
0	0	1	373	0	20	0	0	40	2
0	0	27	36	9	36	387	0	1	0
0	0	0	322	0	0	0	0	0	194
0	0	1	387	3	2	0	0	92	0
0	0	3	157	3	1	0	0	0	325

Figure 15. Confusion Matrix for Class Section Attacked SVM when 500 Samples were used

is not classified as the target class as is observed in Chapter IV. Results – 4. 1. Section Boundary Attack.

4. 4. Recreating SVM Experiments

For testing the recreation of the SVM, a selection of inputs is used to build the training data for the new SVM. The number of inputs, or samples in ad_train, is 5,000, and the training data generated from those inputs is about 18,500 samples. Using this generated training data, a new SVM is built. Afterwards, the two SVMs are compared by using a different selection of inputs, ad_test. Both the original and recreated SVM are tested using ad_test and the confusion matrices for both SVMs are provided as **Figure 16** and **Figure 17**. The accuracy for the original SVM is 99%, and the accuracy for the new SVM is 93%. For the individual classes, the accuracy decreases to a maximum of 11.63%, for class 9. Other classes with more than 5% decrease in accuracy are class 2, class 4, class 5, and class 8. Class 2, class 4, class 8, and class 9 proved to be more resistant to the Section Boundary Attack as discussed in Chapter IV. Results – 4. 1. Section Boundary Attack and observed in **Figure 10**. Thus, it makes sense that using the adversarial example provided as training samples for the target class would decrease the accuracy. The original SVM still classifies the adversarial examples generated for this classes as their original classes about 50% of the time. This difference in classification for the adversarial examples for these classes causes a significant difference in accuracy between the two SVMs. Nevertheless, based on these results, it can be concluded that the SVMs are quite similar. Granted,

514	0	0	0	0	2	2	1	1	0
0	560	0	0	0	0	2	0	2	0
4	1	493	0	0	0	1	2	1	0
0	1	1	500	0	0	0	1	4	3
1	0	1	0	477	0	0	0	0	3
1	0	0	5	0	424	3	0	2	1
0	0	0	0	0	3	493	0	0	0
0	2	4	0	0	0	0	508	0	2
0	0	0	0	2	1	0	0	481	1
0	0	0	1	3	1	0	2	0	482

Figure 16. Confusion Matrix for Original SVM using Half of Testing Data

504	1	3	3	0	7	0	2	0	0
0	551	1	2	0	1	2	0	7	0
6	3	457	5	11	1	2	6	10	1
0	2	18	470	0	7	0	4	9	0
1	2	2	4	462	2	4	1	1	3
3	6	7	16	4	382	12	3	3	0
1	3	4	0	5	1	479	2	1	0
0	1	5	2	1	7	0	479	3	18
3	3	2	12	2	3	2	2	455	1
3	0	0	9	24	8	0	11	1	433

Figure 17. Confusion Matrix for Recreated SVM

the SVMs are not identical, but they should behave similarly for the majority of inputs. Thus, an attack that works for the recreated SVM is likely to also be effective to attack the original SVM.

CHAPTER V

CONCLUSION & FUTURE WORK

5. 1. Conclusion

In conclusion, the Section Boundary Attack can generate adversarial examples using only a section of a sample point from the target class. The average misclassification caused by the attack is 37.78%, about 80% when a universal section is used, and about 63.67% when class sections are used for the experiments described in Chapter IV Results. The runtime is 22.67 seconds when the Section Boundary Attack is implemented, which may not be very practical or effective. However, generating a universal section or class section allows adversarial examples to be generated in less than a second, only needing time to "train" or generate the universal or class sections. When considering runtime and accuracy, it is clear that generating and implementing class sections is the best method of attack.

Using a random section, the perturbation is limited. However, using several sections to generate the best adversarial example still leads to promising results. Jing et al. propose a guided decision-based attack that also does not work with all the starting adversarial point, or target point [13]. However, they use a more search-oriented approach to find a semantically relevant component that does not include background elements, like the head of the Persian cat in **Figure 2**. In this thesis, it is shown that a random section, regardless of its semantic meaning, can still lead to misclassification and easily generate adversarial examples even if the section consists of background elements. Granted, the change performed by the attack proposed in this thesis is much

greater than the attack proposed by Jing et al. [13], but the number of calls is less, and potentially the execution time as well. Additionally, a clear vulnerability of SVM models is still found and explored. The recreation of the SVM, while not perfect, may likely lead to other vulnerabilities being exploited, providing additional benefits of using the attack proposed in this thesis. Despite the benefit, further work still needs to be done to improve this attack and further research in this area is needed to potentially lead to more secure and robust classification models.

5. 2. Future Work

For the Section Boundary Attack, the adversarial examples generated are not very natural. Additionally, while only a small section of the input is changed, the perturbation on the input is still noticeable to the human eye at times. This is likely because a direct approach is used to go from input to adversarial example, leading the attack to make more changes than may be necessary. This is attempted to be limited by decreasing the size of the sections. However, the accuracy of the attack suffers as a compromise. One reason for this may be that empty sections are being used – the margins and background of the image, the input. Thus, the accuracy of the attack may benefit from constricting the random adversarial section to the center of the image, where the main subject of an image tends to be found. Other methods to isolate the main subject to limit the random adversarial section may also result in more natural adversarial examples and better accuracy. When considering the MNIST dataset, one option is to constrict the random adversarial section to nonzero elements.

Furthermore, the step or size of perturbation θ for the experiments is set to $\theta = 0.01 = 1/100$. This causes the calls to be limited to a maximum of 100 for each input. However, it may have contributed to the unnaturalness of the changes. Smaller steps lead to less change, potentially making it less imperceptible to the human eye. However, due to concerns about the execution time,

the step size is set to $\theta = 0.01$. Thus, finding ways to reduce the execution time while being able to decrease the step size could be another method to improve the performance of the attack.

Lastly, the attack needs to be implemented and tested for more datasets to see its performance and compare how working with larger images or more classes may affect the effectiveness of the attack. As stated in Chapter II Review of Literature – 4. 3. Defense Strategies, models have a harder time finding the *predominate features* when working with high-dimensional data, making them vulnerable to adversarial attacks [1]. For this reason, changing the dimension of the data may likely affect the accuracy of the attack. Thus, the attack must be tested on a variety of data.

REFERENCES

[1] N. Bhagoji, D. Cullina, and P. Mittal. "Dimensionality Reduction as a Defense against Evasion Attacks on Machine Learning Classifiers," 2017. [Online]. Available: https://www.semanticscholar.org/paper/Dimensionality-Reduction-as-a-Defense-against-on-Bhagoji-Cullina/10bd926253cbf5829ee92e927127641b69546e65

[2] Biggio, B. Nelson, and P. Laskov. "Poisoning attacks against support vector machines," 2012. [Online]. Available: https://arxiv.org/abs/1206.6389

[3] Biggio, B. Nelson, and P. Laskov. "Support Vector Machines Under Adversarial Label Noise" in *Asian Conference on Machine Learning,* Nov. 2011, pp. 97-112.

[4] Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. "Context Encoders: Feature Learning by Inpainting" in *CVPR 2016*, pp. 2536-2544.

[5] J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and Harnessing Adversarial Examples. Presented at ICLR 2015. [Online]. Available: https://arxiv.org/abs/1412.6572

[6] J. Wu and R. Fu. "Universal, transferable and targeted adversarial attacks," 2019. [Online]. Available: https://arxiv.org/abs/1908.11332

[7] K. Ren, T. Zheng, Z. Qin, and X. Liu. "Adversarial Attacks and Defenses in Deep Learning," *Engineering*, vol. 6, issue 3, pp, 346-360, Jan. 2020, doi: https://doi.org/10.1016/j.eng.2019.12.012

[8] Y. Qin, N. Carlini, I. Goodfellow, G. Cottrell, and C. Raffel. “Imperceptible, Robust, and Targeted Adversarial Examples for Automatic Speech Recognition” in *International Conference on Machine Learning*, May 2019, pp. 5231-5240.

[9] Y. Zhou, M. Kantarcioglu, B. Thuraising. “Adversarial Support Vector Machine Learning” in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, Aug. 2012, pp. 1059-1067.

[10] Z. Zhao, D. Dua, and S. Singh. Generating Natural Adversarial Examples. Presented at ICLR 2018. [Online]. Available: https://arxiv.org/abs/1710.11342

[11] J. Su, D. V. Vargas and K. Sakurai, "One Pixel Attack for Fooling Deep Neural Networks," in *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 5, pp. 828-841, Oct. 2019, doi: 10.1109/TEVC.2019.2890858.

[12] W. Brendel, J. Rauber, and M. Bethge. Decision-Based Adversarial Attacks: Reliable Attacks Against Black-Box Machine Learning Models. Presented at ICLR 2018. [Online]. Available: https://arxiv.org/abs/1712.04248

[13] H. Jing, C. Meng, X. He and W. Wei, "Black Box Explanation Guided Decision-Based Adversarial Attacks," *2019 IEEE 5th International Conference on Computer and Communications (ICCC)*, Chengdu, China, 2019, pp. 1592-1596, doi: 10.1109/ICCC47050.2019.9064243.

[14] Y. LeCun, C. Cortes, and C. J.C. Burges, “The MNIST Datasbase,” November, 1989. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[15] J. Wu, and R. Fu. “Universal, transferable and targeted adversarial attacks,” 2019. [Online]. Available: https://arxiv.org/abs/1908.11332

[16] M. Usama, M. Asim, S. Latif, J. Qadir and Ala-Al-Fuqaha, "Generative Adversarial Networks For Launching and Thwarting Adversarial Attacks on Network Intrusion Detection Systems," *2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC)*, Tangier, Morocco, 2019, pp. 78-83, doi: 10.1109/IWCMC.2019.8766353.

[17] Z. Zhu, Y. Lu and C. Chiang, "Generating Adversarial Examples By Makeup Attacks on Face Recognition," *2019 IEEE International Conference on Image Processing (ICIP)*, Taipei, Taiwan, 2019, pp. 2516-2520, doi: 10.1109/ICIP.2019.8803269.

[18] D. J. Miller, Z. Xiang and G. Kesidis, "Adversarial Learning Targeting Deep Neural Network Classification: A Comprehensive Review of Defenses Against Attacks," in Proceedings of the IEEE, vol. 108, no. 3, pp. 402-433, March 2020, doi: 10.1109/JPROC.2020.2970615.

[19] Y. Liu, S. Mao, X. Mei, T. Yang and X. Zhao, "Sensitivity of Adversarial Perturbation in Fast Gradient Sign Method," *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*, Xiamen, China, 2019, pp. 433-436, doi: 10.1109/SSCI44817.2019.9002856.

BIOGRAGPHY SKETCH

Yessenia Rodriguez was born in McAllen, Texas. In 2016, she Yessenia graduated from Brownsville Early College High School and entered The University of Texas Rio Grande Valley. There she completed a Bachelor of Science in Computer Science in 2018. Yessenia continued her education by pursing a Master of Science in Computer Science at the University of Texas Rio Grande Valley. During the summer of 2019, Yessenia was employed by the Upward Bound Program of the University of Texas Rio Grande Valley as an instructor for a STEM Summer Internship Program. Then, in 2020, Yessenia was employed by the University as a graduate assistant aiding professors in their courses, developing material, and teaching the 1101 course for computer science. In Fall 2020, Yessenia graduated from the University of Texas Rio Grande Valley, completing a Master of Science in Computer Science. Yessenia can be contacted by email at yessenia.yess98@icloud.com.