Adversarial Examples

JUSTIN WICKES*, Technische Universität Dortmund, Germany

In a world of growing reliance on machine learning and artificial intelligence, adversarial examples exploit a seemingly ubiquitous vulnerability within machine learning models. The danger adversarial examples have the potential to present is real. Thus research into the various adversarial attacks has been the focus of almost a decade's worth of research, as many have tried to understand these attacks in order to help prevent the danger they pose. In this paper, we will be exploring what an adversarial example is, why they are important, and the kinds of attacks that can be made with them.

ACM Reference Format:
Justin Wickes. 2022. Adversarial Examples. 1, 1 (August 2022), 6 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Adversarial Examples are malicious inputs constructed to be intentionally misclassified by a given machine learning model. A salient example would be that of a stop sign and a self-driving car [4]. The car processes and analyzes images of its surrounding to determine its behavior. When it identifies the image of a stop sign located in front of the car, it proceeds to stop, as any car should. Now if an attacker wanted to perform an adversarial attack on this self-driving car, they would have a few options. They could physically change the stop sign, so that it is no longer identifiable by the intelligence guiding the car. This could be achieved by physically marking the stop sign or even by placing a maliciously crafted sticker onto the sign. Alternatively the attacker could gain access to the car's image processing and change the images before they are analyzed. This way, the car could be fooled into driving past a stop sign without even physically marking said sign. This example illustrates quite well what an adversarial example is. It is an intentionally modified input, in this case an image of a stop sign, which is designed to be misclassified or otherwise misunderstood by the machine learning model. This example also can be used to clarify another aspect of adversarial examples, namely that their changes are deliberate. Simply painting the whole stop sign black would cause the car to no longer be able to identify the sign, but this would be ignoring an important aspect of adversarial examples, which is their indistinguishability from unmodified inputs.

In the aforementioned example, the goal is to modify the stop sign so it is misclassified by the car, but the modification should be as small as possible and ideally imperceptible to the human eye. This aspect of adversarial examples is an optimization problem. The attacker wants to create a modified input that is as close to indistinguishable from inputs that share it's correct label, while also having as high a chance of misclassification as possible. Making the stop sign look less and less like a stop sign makes it more likely that it is misclassfied, but also more likely that someone will identify that the sign or the car has been tampered with. This is where the intentionality of the perturbation is important. Simply painting over the sign or adding random noise to an image is not what makes a good adversarial example. All changes to the image should be deliberate and purposeful so as to achieve the desired optimization between chance of misclassification and inconspicuousness. Furthermore it is quite trivial for a machine learning model to simply remove or ignore random noise [10]. Many models are actually quite robust against noisey images, what is of interest, is that these same models can fall victim to adversarial examples.

The significance of adversarial examples should be becoming clear. Up until now we have only discussed how road signs could be tampered with to intentionally cause accidents and chaos, but adversarial examples are widely applicable. One study [1] has shown that one can create adversarial examples with 3D-printed objects. They produced a sea turtle that was perceived by a machine learning model as a rifle. It would not be inconceivable that the opposite could be achieved as well; creating a dangerous weapon, that when perceived by an artificial intelligence, is seen as harmless. Furthermore another study [2] has been done on adversarial examples for natural language processing systems, which proved that recorded audio could be changed slightly (by less than 1%), and the deep neural network would interpret it as a completely different phrase. Moreover they could change the audio to actually force the DNN to interpret it as a specific phrase of their choosing. The ramifications of these studies are clear to see; with more reliance on artificial intelligence and machine learning, we will become more vulnerable to these kinds of attacks. As A.I. becomes more ingrained in fundamental, everyday systems, the danger only grows. This is why the study of adversarial examples is so important.

For most of this paper we will be focusing on supervised learning models, specifically 2D image classifiers. This is where the most abundant research on adversarial examples is focused. Even with this restriction, there are still a plethora of adversarial attacks. All of these attacks have their own pros and cons and thus, separate use-cases. Selecting a specific attack tailored to the situation and the attacker's goals is almost always a fundamental decision for an adversarial attack. This paper is structured as follows: in section 2) we will examine the different threat models, as well as the difference between White-Box and Black-Box attacks. Then in section 3) we will review a few White-Box attacks and contrary to these approaches there are also Black-Box attacks, which we will discuss in section 4). We will conclude by examining the weaknesses of the proposed Black-Box attack from section 4) and then consider possible defenses.

2 THREAT MODELS

Threat models enumerate the goals and capabilities of adversaries in a target domain. What this means is a threat model indicates what the adversary wants to achieve and how much the adversary

*

Author's address: Justin Wickes, justin.wickes@tu-dortmund.de, Technische Universität Dortmund, Dortmund, NRW, Germany.

knows about the target machine learning model. A threat model is composed of an adversarial goal and the adversarial capabilities [6]. There are four adversarial goals and five adversarial capabilities which we will be discussing.

2.1 Adversarial Goals

The first goal is Confidence Reduction, which works to reduce how sure a given model is, that the input belongs to it's classification. This adversarial goal does not work to outright misclassify an adversarial example, but rather make a model less confident that it belongs to the class it was assigned. This is thus the least complex adversarial goal. The second goal is Misclassification, which tries to make the model assign the adversarial example to any class other than the original class. The third adversarial goal is Targeted Misclassification, which differentiates itself from the second goal by forcing the output classification to be a specific target class. The fourth and most complex goal is Source/Target Misclassification. This entails forcing the output classification of a specific input to be a specific target class. The difference between this goal and the third one is how the input is specified. For Source/Target Misclassification a specific input is chosen and modified to appear like the target class, whereas for Targeted Misclassification there is no specificity for the input. As long as it is misclassified as the targeted class, any possible input could be chosen and modified.

2.2 Adversarial Capabilities

Adversarial capabilities describe the amount of information about the target model, that is available to the adversary. The most knowledge an adversary can have is knowledge of both the architecture and training tools. This constitutes the first of the adversarial capabilities. With knowledge of both the architecture and training tools. The attacker has to access the training data, as well as functions and algorithms used for network training. They are able obtain knowledge about the DNN's architecture. This includes the activation functions of neurons, the number and type of layers, as well as bias matrices and weights. The attacker also knows which algorithm was used to train the network, including the associated loss function. This is the strongest adversary that can analyze the training data and simulate the deep neural network completely. However, in practical settings, knowledge of both the architecture and training tools is rare, and simply obfuscating what architecture and training tools were used is enough to stop these attacks.

The next capabilities are knowledge of either the architecture or the training data but not both. These attacks require less information and are thus easier to perform, but the adversarial examples crafted in the attack will likely be weaker (i.e. less likely to be misclassified) than attacks that use more information. Any attack that uses knowledge of the architecture, the training data, or both, is a White-Box attack. White-Box attacks have some amount of crucial internal information on the target model, this differentiates them from Black-Box attacks, which have no internal information. The next two capabilities constitute those of a Black-Box attack.

The next capability is that of an Oracle. The term oracle comes from the field of cryptography and defines the capability of using a given neural network to obtain output classifications when supplying inputs. Essentially the adversary can query or input into the neural network and the only knowledge they can obtain from this is what the input was classified as. There is no knowledge of architecture or training data, their only capability is to map some amount of inputs to their outputs classifications. All input features are usually known for image classifiers, but for other models it may be more difficult to find out what the inputs are. Thus ascertaining what all the input components are can be somewhat of a challenge in other contexts. This oracle capability can be further parameterized by the number of queries allowed/available. Querying all possible inputs and receiving their outputs proposes a few issues. While one could theoretically query every single possible input and receive their output classification, thus making a copy of the target neural network, it is unreasonable and impractical. If a machine learning model had N input components, with each component taking a discrete value among a set K of possible values. The number of inputs that would have to be queried is K^N. Computation time alone would be problematic, but additionally the more an adversary queries their target, the more likely they are to being discovered. Furthermore many API's have a set number of queries one user can make, before they are denied access. Such restrictions and limitations are crucial factors for Black-Box attacks using oracles.

The final capability is simply the ability to collect samples. The attacker has some number of input-output pairs and must use this information to execute a Black-Box attack. They cannot modify the inputs in any way, as opposed to the oracle. These adversarial attacks require large quantities of samples and even then are among the weakest attacks possible.

In summary, as knowledge over the target decreases, so does the strength of the attack and as the complexity of the goals increase, so does the difficulty of the attack. This should explain why there is not simply one catch-all attack that should be used all the time. Attacks have different use-cases that are determined by the adversary's goals and knowledge. So if an attacker were to be clever and efficient they would select different attacks depending on the scenario.

3 WHITE-BOX ATTACKS

There are many White-Box attacks that exist, for all kinds of use-cases, but for this section we will focus on three. These attacks are the Szegedy et al. method [10], the Fast Gradient Sign Method (FGSM) [3], and the Jacobian-based Saliency Map Attack (JSMA) [6]. Two of these attacks, FGSM and JSMA, are of particular note, because they will be used in the Black-Box attack described in the next section.

3.1 Szegedy et al. Method

$$c|r| + loss_f(x + r, l) \tag{1}$$

For this function x is an image/input. It represents a vector of pixels. Then r is the change of pixels to create an adversarial example, while l is the targeted outcome class. Finally c is used to balance the distance between images and predictions. The first term measures the distance between the adversarial example and the original image and the second term is the distance between the predicted outcome of the adversarial example and the desired class l. By minimizing the

function with respect to r, adversarial examples can be generated. This function is a perfect example of Targeted Misclassification and was the first method for creating adversarial examples [10].

3.2 Fast Gradient Sign Method

$$\delta_{\vec{x}} = \epsilon sgn(\nabla_{\vec{x}} c(F, \vec{x}, y)) \tag{2}$$

For this function, the goal is to create an adversarial example $\vec{x^*} = \vec{x} + \delta_{\vec{x}}$. In this instance, $\vec{x}$ is the input image, while $\delta_{\vec{x}}$ is the perturbation to be applied. $c(F, \vec{x}, y)$ is a cost function, where F represents the model's parameters, $\vec{x}$ is the input, and y is the label that was assigned to $\vec{x}$. The sign of the model's cost function gradient is represented with $sgn(\nabla_{\vec{x}} c(F, \vec{x}, y))$. Finally ϵ is the input variation parameter and controls the amplitude of the perturbation. The larger ϵ is, the more likely an adversarial sample will be misclassified, but the easier it will be for humans to detect that it is an adversarial sample. Normally this value lies between 0.25 to 0.4 [9] for most attacks, but it can be raised if required. This method applies an imperceptibly small perturbation to all pixels in the input image and is well suited for fast crafting of many adversarial samples. The one major downside being that the combined total perturbation over the whole image is quite large and thus this method of attack is potentially easier to detect than the JSMA. This function is also an example of a simple Misclassification, as no targets are specified.

3.3 Jacobian-based Saliency Map Attack

$$S(\vec{x}, t)[i] = \begin{cases} 0 \text{ if } \frac{\partial F_t}{\partial \vec{x}_i}(\vec{x}) < 0 \text{ or } \sum_{j \neq t} \frac{\partial F_j}{\partial \vec{x}_i}(\vec{x}) > 0 \\ \frac{\partial F_t}{\partial \vec{x}_i}(\vec{x}) \left| \sum_{j \neq t} \frac{\partial F_j}{\partial \vec{x}_i}(\vec{x}) \right| \text{ otherwise} \end{cases} \tag{3}$$

Similar to the last attack, the JSMA tries to craft an adversarial sample $\vec{x^*} = \vec{x} + \delta_{\vec{x}}$. This time however, it adds the perturbation $\delta_{\vec{x}}$ to a subset of the input components $\vec{x_i}$. This means the perturbation is only applied to a few pixels in the inputted 2D image, as opposed to the Fast Gradient Sign Method, where the perturbation is applied to all the pixels. F is the target model and it's parameters are required for this attack, similar to FGSM. $S(\vec{x}, t)[i]$ is the adversarial saliency value of component i for a target class t. Equation 3 is the definition of this saliency value, with $\left[\frac{\partial F_j}{\partial x_i}\right]_{ij}$ is the model's Jacobian matrix. The Jacobian matrix contains all partial derivatives of a vector function.

In a Jacobian-based Saliency Map Attack, input components are sorted in decreasing order by their saliency value and perturbed until the sample is misclassified. In the discipline of Computer Vision, a saliency map describes a mapping of an image that indicates where peoples eyes are drawn. It is used to locate visual "hotspots". In a similar way, this algorithm creates a sort of saliency map, which measures not where human eyes are drawn, but how important a given part of an image is for deciding it's output classification. This attack perturbs the most deciding pixels first, and it perturbs them by a large amount in comparison to FGSM. The greater the perturbation per component, the less components must be perturbed. The input variation parameter ϵ is often $\epsilon = 1$ for this attack. Υ is the distortion and is another important parameter, because it quantifies how many input components have been perturbed. As one might notice, a large difference between FGSM and JSMA is how their perturbations are applied. In JSMA a few pixels are changed greatly, so that overall the perturbation over the whole image remains low. In this way, JSMA attacks are likely harder for a human to notice. The Jacobian-based Saliency Map Attack is, however, more complex than the FGSM, meaning it has a higher computing cost. This attack also constitutes a Source/Target Misclassification, which is another major difference from FGSM.

Hopefully by explaining these attacks, it becomes more clear why cleverly selecting your attack is important when crafting adversarial examples. These are all White-Box attacks, yet they have very different use-cases.

4 BLACK-BOX ATTACK

Now we can begin to tackle the Black-Box attack [9]. To review: a Black-Box attack has no detailed knowledge of the target model's architecture, parameters, or training data. The adversary only has knowledge of an oracle O, which can access the label $\tilde{O}(\vec{x})$ assigned to any chosen input $\vec{x}$. To solve this, we first create a substitute for the target system, which we can attack with one of the previously discussed White-Box attacks. This solves the lack of information on the architecture and parameters of the model. To solve the lack of training data, we use a synthetic dataset using synthetic inputs and the outputs of the target model. In this way, the lack of information about the target can be overcome.

This Black-Box attack has three restrictions: its only capability is to observe the Oracle O, the number of labels it queries on the target remote system is limited, and it applies and scales to multiple machine learning classifier types.

4.1 Substitute

An approximation F of the target model T is created. Determining the actual architecture of this approximation is not a limiting factor, as we have access to the oracle O and thus have knowledge of the expected input and output. Consequently an architecture fitting to the input-output relation can be chosen. Within [9] it was found that learning a substitute model that mimics the remote system's accuracy is unimportant, what is of importance is mimicking the decision boundaries. The decision boundaries describe an area in a problem space where the output classifier is ambiguous. It is the boundary between decision regions. If we look at our stop sign example from the beginning, a decision boundary would be the area where the model is unsure if the sign it's processing is a stop sign or a yield sign. Thus everything on one side of the decision boundary is stop signs and everything on the other is yield signs. Creating perturbations that push valid inputs towards decision boundaries is how one creates an adversarial example. In summary what is important is not the actual accuracy of the substitute F in comparison to the target T, but rather how F groups its outputs.

These substitutes can be learned with a Deep Neural Network or Logistic Regression. The attack can be used against DNNs, as well

as other classifier types, namely Logistic Regression, SVM, Decision Tree, and Nearest Neighbors.

4.2 Transferability

Transferability is the property that explains how making a White-Box attack on a substitute model would create adversarial examples that are misclassified by the target. Transferability describes is a property that describes the following phenomena: adversarial examples trained on a machine learning model may be misclassified by another model even if their architectures significantly differ. Why adversarial examples are transferable is still in debate, but the property itself is undoubtedly real.

There are two kinds of transferability [8]: Intra-technique transferability and Cross-technique transferability. Intra-technique transferability is defined across models trained with the same machine learning technique but different parameter initializations or datasets. All models (DNN, LR, SVM, kNN, DT) are found vulnerable to this type of transferability, but the phenomenon is stronger for differentiable models like deep neural networks and logistic regression than for support vector machines, decision trees, and and nearest neighbors [8]. Decision trees, while still vulnerable to intra-techique transferability, seem to be the most resilient to it. Cross-technique transferability considers models trained using two different techniques. Deep neural networks and nearest neighbors are particularly resilient to this form of transferability, whereas logistic regression, support vector machines, decision trees, and ensembles of models are considerably more vulnerable to Cross-technique transferability [8].

This is how adversarial examples created from one model may be used to attack another and it is why this particular Black-Box attack works at all. There would be no use in training a substitute model if the adversarial attacks made against it could not be used to attack another model. As such this property of transferability is critical to this Black-Box attack, and consequently is the pairing of substitute model and target model of great importance. This is because some machine learning techniques are more resilient to certain kinds of transferability, as such it is often within the adversary's interest to train multiple substitute models, each with different architecture or parameters. This way, they can simply select the substitute that produces the most often misclassified adversarial examples.

4.3 Synthetic Data

As previously stated, the adversary has no knowledge of the training data used in the target model, so synthetic data must be created. This data is created using a heuristic that measures the directions in which the the target model's output begins to vary, using the substitute's Jacobian matrix. An original training set is selected and synthetic data is continually added, until the substitute mimics the target's decision boundaries. This is called Jacobian-based Dataset Augmentation.

Algorithm 1 - Substitute DNN Training: for oracle $\tilde{O}$, a maximum number max_ρ of substitute training epochs, a substitute architecture F, and an initial training set S_0.

Input: $\tilde{O}$, max_ρ, S_0, λ
1: Define architecture F
2: **for** $\rho \in 0 \,..\, max_\rho - 1$ **do**
3: *// Label the substitute training set*
4: $D \leftarrow \left\{(\vec{x}, \tilde{O}(\vec{x})) : \vec{x} \in S_\rho\right\}$
5: *// Train F on D to evaluate parameters θ_F*
6: $\theta_F \leftarrow \text{train}(F, D)$
7: *// Perform Jacobian-based dataset augmentation*
8: $S_{\rho+1} \leftarrow \{\vec{x} + \lambda \cdot \text{sgn}(J_F[\tilde{O}(\vec{x})]) : \vec{x} \in S_\rho\} \cup S_\rho$
9: **end for**
10: **return** θ_F

[8]
(4)

As previously stated, the goal of this algorithm is to augment images from the original training set to be classified closer to decision boundaries. This algorithm will be explained in the following steps:

(1) The initial collection of training data. This data does not need to come from the targeted model's training distribution. Samples should however be chosen intelligently, collecting a certain number of samples per output classification is a strong general strategy.
(2) Select the architecture to be trained as F (the substitute)
(3) For max_ρ Epochs following steps are repeated:
   (a) Query $\tilde{O}(\vec{x})$ to label every $\vec{x}$ from the initial training set
   (b) Train architecture using training set with classical machine learning techniques. One would train this substitute as if it were any other machine learning model.
   (c) Perform the Jacobian-based Dataset Augmentation and augment training set to produce larger set with more points. This set better represents the model's decision boundaries.

The Jacobian-based Dataset Augmentation in Figure 4 on line 8 evaluates the sign of the Jacobian matrix dimension correlative to the label assigned to input $\vec{x}$ by the oracle. The λ describes the step size in the sensitive direction and it has been shown [9], that increasing the step size decreases transferability. As transferability is crucial to this attack's strategy, the value λ was later altered to be periodic, which also increased the accuracy of the substitute while also keeping transferability at an acceptable level. Consequently, after every period, the step size alternates between being positive and negative.

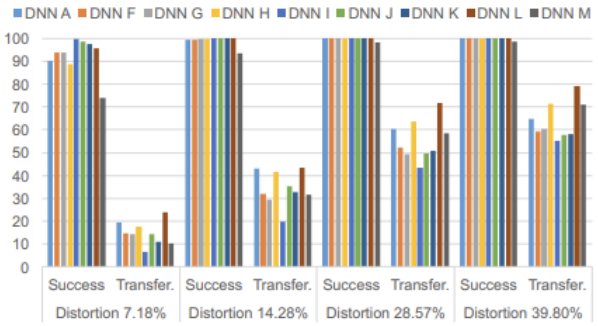
To further reduce querying when applicable, Reservoir Sampling [11] was introduced. This is useful in realistic scenarios and for paid APIs where the number of queries is limited before exceeding quota or when being detected is a high risk. Reservoir sampling is used to select κ new inputs before a Jacobian-based dataset augmentation. Without reservoir sampling, the training data grows exponentially, literally doubling every iteration. This means that with high max_ρ, the queries directed at the target will grow very quickly. Reservoir sampling allows for the first few iterations of the Substitute DNN

Training to proceed unimpeded. The dataset, for the first few iterations, can grow exponentially. Then after a set iteration has been reached, reservoir sampling is used to limit the growth of the dataset. This method of sampling also ensures that each input has an equal chance to be chosen for Jacobian-based Dataset Augmentation in the next iteration.

After the substitute is trained and has similar decision boundaries to the target, a White-Box attack can be performed on the substitute. The adversarial examples that were crafted by the attack, can then be used on the remote system with a high chance of misclassification.

4.4 Results

In the study [9], three attacks of note were performed. One was on the remote platform of MetaMind who were using a Deep Learning based machine learning technique. In the Black-Box attack, MetaMind was queried 6,400 times during the training of the substitute. Adversarial samples directed to MetaMind were misclassified at a rate of 84.24%. Following that they proceeded to attack Amazon who had trained their model using logistic regression. Amazon was queried 800 times and the crafted adversarial examples were misclassified 96.19% of the time. Finally they attacked Google, whose machine learning technique could not be ascertained. They were queried 2,000 times and achieved a misclassification rate of 88.94%. While these misclassification rates are certainly impressive, the fact that each remote platform was queried a different amount of times is questionable. One of the results from these attacks was the realization that the defender may increase the attacker's cost by training models with higher input dimensionality or modeling complexity. These two factors are what cause adversaries to need to make so many queries while training their substitutes. While it could be argued that a more complex target requires more queries and thus the vastly differing number of queries in these attacks is justifiable, it could also equally be argued that the remote systems were simply queried until the misclassification rate seemed impressive. There is a case to be made here about the dishonesty of the results.



[9] (5)

Further results include the finding that FGSM and JSMA have similar transferability rates when similar amounts of perturbation are applied. The distortion Υ in the JSMA algorithm increases transferability, the higher it is. A 29% distortion improves adversarial sample transferability significantly. Additionally it was found that increasing the number of training iterations does not improve adversarial sample transferability, if the substitute has reached an asymptotic accuracy. Furthermore substitutes using logistic regression perform measurably better against logistic regression and support vector machine trained models compared to substitutes using deep neural networks.

5 WEAKNESSES AND DEFENSES

There are two weaknesses of this particular Black-Box attack. One was previously mentioned, namely that models with higher complexity or input dimensionality require more queries to train the substitute. This can be slightly mitigated by the adversary by choosing not to query from a single user, and instead distributing oracle queries among a set of colluding users, thus making them harder to detect. The other weakness is that this method of attack can only learn substitutes with DNN or LR techniques. As discussed in the section on transferability, nearest neighbor models have a resiliency to Cross-technique transferability. Being that only DNN or LR substitutes can be learned, every attack on a kNN model would hinge very strongly on Cross-technique transferability for the adversarial examples trained on the substitutes. As such kNN models as well as sufficiently complex models can be difficult to attack using this method.

There are a few possible defenses against this Black-Box attack, but no complete and comprehensive defense is known. One of the defenses introduced in the paper [9] is Defensive Distillation [7], which is a kind of Gradient Masking. A gradient in mathematics is the derivative of a function with more than one input variable. As with any derivative it measures the rate of change. For this reason the gradient is important, because locating high rates of change can often lead to decision boundaries. Gradient Masking and thus Defensive Distillation try to smoothen out the gradient, to make it harder to identify where high rates of change are. Defensive Distillation has been shown to work in White-Box settings in repelling a FGSM attack. However it does not work against this Black-Box attack. The reason is that the substitute doesn't interact with the gradient at all, it only queries the oracle for labels. Consequently the target model may be distilled and thus be masking it's gradient, but the substitute is not distilled. Thus a White-Box attack can still successfully be performed on the substitute and Gradient Masking does nothing to hinder the success of the adversarial examples crafted from said attack. Due to this, Gradient Masking and Defensive Distillation are unsuccessful at defending against this Black-Box attack.

The next defense, which is more successful, is adversarial training [3] [5]. Adversarial training is the act of retraining a given machine learning model with adversarial examples mixed in with the training data, however the adversarial examples are assigned their correct labels. This teaches the model to ignore the intentional perturbations in the adversarial examples and learn only from "robust" features. Training with a significant enough ϵ value ($\epsilon \geq 3$)[9], drastically decreases the misclassification rate of adversarial examples and renders the Black-Box attack ineffective. A major issue with adversarial training, however, is that it only defends a given machine learning model against the same attacks used to craft the examples originally included in the training data. For example, one could train their model on adversarial examples crafted using the Fast Gradient

Sign Method, but their model would only learn to defend against adversarial examples with ubiquitous small perturbations. So if an adversary staged a Black-Box attack and used the Jacobian-based Saliency Map Attack to craft adversarial examples, then the defender would be practically defenseless. Their model would be trained to be robust against widespread small perturbations, so when faced with extremely selective large perturbations, it would be as if no adversarial training had been performed at all. For this reason, a good defense would be to train on adversarial examples from an ensemble of substitutes of the original model. This way, different attacks can be performed on each substitute and thus a variety of adversarial examples can be crafted. The machine learning model would become robust against a large set of adversarial attacks and therefore increase the chances of a successful defense.

The final defense would be any comprehensive form of Input Modification or Denoiser. Such defenses clean an input of noise and perturbation before it is passed onto the model to be analyzed. Such a defense could possibly remove the relevant perturbations of an adversarial example before they are misclassified or otherwise harm the machine learning model.

6 CONCLUSION

The Black-Box attack introduced by Papernot et al. is a breakthrough in adversarial attacks. While the honesty of some of the reported figures could be called into question, this Black-Box attack nonetheless shows that attacks using severely limited knowledge are possible. While the number of queries required to produce effective adversarial examples can be quite large, and even though the attack depends heavily upon the substitute and target model requiring significant transferability, their paper is still a large step forward in the subject area of adversarial examples. Before their paper there were no Black-Box attacks that were nearly as reliable as the one introduced by Papernot and his colleagues. There were only really White-Box attacks. Now there are a plethora of papers discussing various Black-Box attacks, many of them drawing inspiration from Papernot and his coworkers. Thus while certainly having flaws, their paper shows how adversarial examples can theoretically be applied in more practical settings, where information is scarce.

Although the majority of successful adversarial attacks have been performed in studies and many of the more ambitious designs [12] have yet to see widespread use, adversarial examples are becoming increasingly more relevant. There have yet to be any examples of real adversarial attacks that have caused major harm, and for this reason the study and understanding of adversarial examples is so important. The danger they pose is not behind us, but rather in front of us, in the future. Eventually we will become more and more reliant on artificial intelligence and machine learning and it is a simple eventuality that adversarial examples will be crafted to mislead these systems. In order to better understand what kinds of attacks are possible and how to defend against them, we must continue to deepen our understanding of these tools.

REFERENCES

[1] Anish Athalye, et al. 2018. Synthesizing Robust Adversarial Examples. (2018).
[2] Nicholas Carlini, et al. 2018. Audio Adversarial Examples: Targeted Attacks on Speech-to-Text. (2018).
[3] Ian J. Goodfellow, et al. 2015. Explaining and Harnessing Adversarial Examples. (2015).
[4] Christopher Molnar. 2022. *Interpretable Machine Learning*.
[5] Aleksander Mądry, et al. 2019. Towards Deep Learning Models Resistant to Adversarial Attacks. (2019).
[6] Nicolas Papernot, et al. 2015. The Limitations of Deep Learning in Adversarial Settings. (2015).
[7] Nicolas Papernot, et al. 2016. Distillation as a Defense to Adversarial Perturbations against Deep Neural Networks. (2016).
[8] Nicolas Papernot, et al. 2016. Transferability in Machine Learning: from Phenomena to Black-Box Attacks using Adversarial Samples. (2016).
[9] Nicolas Papernot, et al. 2017. Practical Black-Box Attacks against Machine Learning. (2017).
[10] Christian Szegedy, et al. 2014. Intriguing properties of neural networks. (2014).
[11] Jeffrey S. Vitter. 1985. Random sampling with a reservoir. (1985).
[12] Zhe Zhou, et al. 2018. Invisible Mask: Practical Attacks on Face Recognition with Infrared. (2018).