Adversarial Machine Learning

Sanjay Thakur
sanjay.thakur@mail.mcgill.ca
260722338

Xoey Mengxi Zhang
xoey.zhang@mail.mcgill.ca
260587451

Srikanth Pasumarthy
srikanth.pasumarthy@mail.mcgill.ca
260728474

Abstract— **Machine Learning models have become pervasive in everyday problem solving and they are used for a great variety of tasks. However, our limited understanding of the vulnerabilities that these models have has exposed a big problem of model security and privacy. In this project, we implement various attacks (FGSM, JBSM, Black Box) and counter them with defense mechanisms, and conclude with proof and explanation of why these defenses are insufficient. We found that all observed attacks have no effect on kNN and in general Black Box attacks are not highly transferable to other models. Finally, defensive models' predictions become significantly less accurate when tested on legitimate inputs, as a trade-off to defending against adversarial attacks. Here's a video that summarizes our project: https://www.youtube.comwatch?v=f_g79JxyPjI.**

I. INTRODUCTION

Conforming to the goal of the project to "investigate safe and responsible methods of machine learning," we decided to study the topic adversarial machine learning. Adversarial attacks in machine learning, are corruptive input examples created to make even well-learned models flounder.

The recent breakthroughs in deep learning using feature representations to make prediction has unleashed unprecedented levels of intelligence to our ML algorithms. However, unlike human beings, these ML techniques fail to handle very trivial adversarial attacks, making these algorithms that have surpassed humans on many fronts extremely fragile and unreliable on others.

Most ML models have been designed with very weak or no threat model in mind, hence, they always end up failing dramatically when facing an adversary. A typical example would be an adversely crafted spam email evading the filter model. Another example would be an autonomous vehicle mistakenly reading traffic signals (see figure 1) [15].

The primary reason why these mistakes happen is because there exists easily exploitable loopholes in the trained models. Even worse, not only can an attacker create an input to fool one specific model, adversarial figures can be misclassified across a broad spectrum of differently trained models.

Our team will be studying and implementing various methods of creating such adversarial examples such as the Fast Gradient Sign Method, the Jacobian-based Saliency Map approach and the black-box approach, to create such fooling examples.

FGSM and JBSM attacks use the concept of gradients to get an idea of where the blind spots of the learning model are and then exploits that knowledge to modify the examples. As a result, the model's prediction is no better than a random prediction.

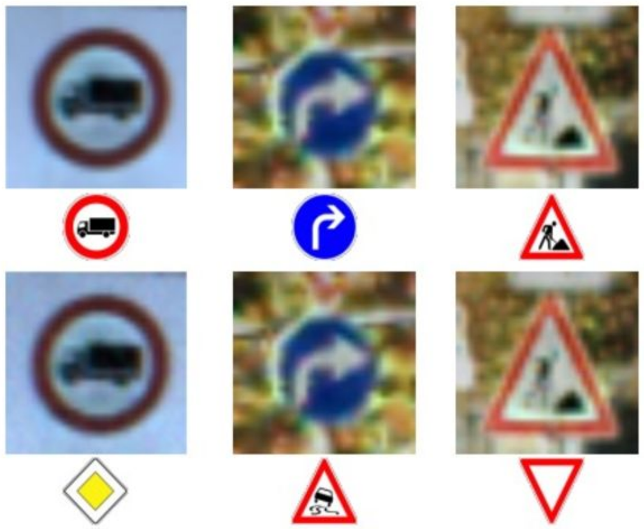


Fig. 1: Images with certain values of pixel can be wrongly read even by the most sophisticated models (last two rows) although they look identical to the naked eye [11].

The black-box approach generates adversarial examples by monitoring the labels predicted by the targeted model on selected examples, and then makes an estimate of how the model makes the prediction.

We'll then also implement a few state of the art defenses against those attacks such as adversarial training, defensive distillation.

Adversarial training is somewhat of a brute force method of training the model to classify all the possible adversarial examples correctly. This technique was considered computationally difficult, but Ian GoodFellow's paper showed how to achieve this cheaply [5].

Defensive Distillation (see Figure 2) is another technique that was introduced by Geoffrey Hinton, which essentially adds another smoother model on top of the ones we want to secure [6].

Lastly, we'll show by implementation why no defense technique until today is robust enough. So there are many different ways of attacking machine learning models and very few ways of defending them.

Given that this problem has some profound implications, it is paramount to push research in this direction. This will not

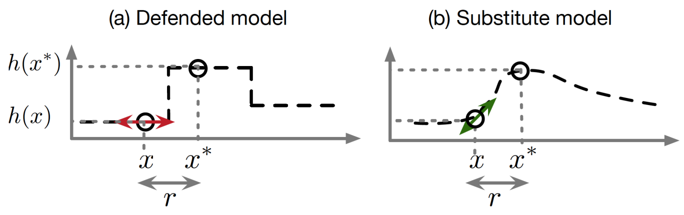


Fig. 2: The non-linearity of any model can be obscured which is known as gradient-masking. It protects the model from adversaries who use techniques like FGSM and JBSM to create adversarial examples for that model [4]

just give us another reason to understand in depth our elusive deep learning models, but also help us to find the loopholes in our real world ML based systems. One definitely would not want their self-driving car to flounder in decision making, not to mention the impetus it can give to other groundbreaking inventions such as that of Generative Adversarial Networks (GANs).

As a summary of our results, we found that our kNN implementation was least affected by our different attack models, and that the Black Box attacks work well against the models they were created for, but are not easily transferrable to other models. We also noticed that defensive models' predictions become significantly less accurate when tested on legitimate inputs, as a trade-off to defending against adversarial attacks.

II. RELATED WORK

The domain in Machine Learning that involves attacking machine learning models and ways to defend it falls under the category of security and privacy of machine learning.

Szegedy et Al. [16] explains a few of the counter-intuitive properties of deep-learning models that result from the extreme difficulty of interpreting deep learning models. Their research also explains that it is the space, rather than the individual units, that contains the semantic information in the high layers of the neural network. The fact that the input-output mappings of the neural network is discontinuous to a fairly large extent, which can be used to craft imperceptible perturbations in the examples to get them misclassified (see Figure 3), is another incredible contribution of the paper.

Later, they also added that the specific nature of these perturbations is not a random artifact of learning: the same perturbation can cause a different network, that was trained on a different subset of the dataset, to misclassify the same input. This suggests that deep neural networks have intrinsic blind spots, and their structure is connected to the data distribution in a non-obvious way.

Szegedy et Al. [16] also shows simple cases of creating adversarial examples by perturbing the input to be predicted in small and often indistinguishable ways. The perturbation is tailored in such a manner that the test input starts shifting outside of its actual class boundaries (see Figure 4).

As a result the prediction made by that model becomes worse than a random prediction.

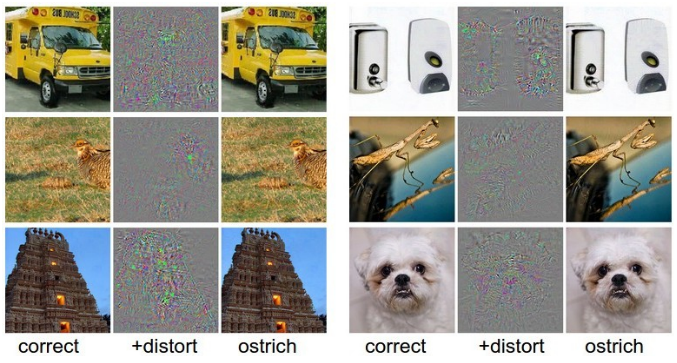
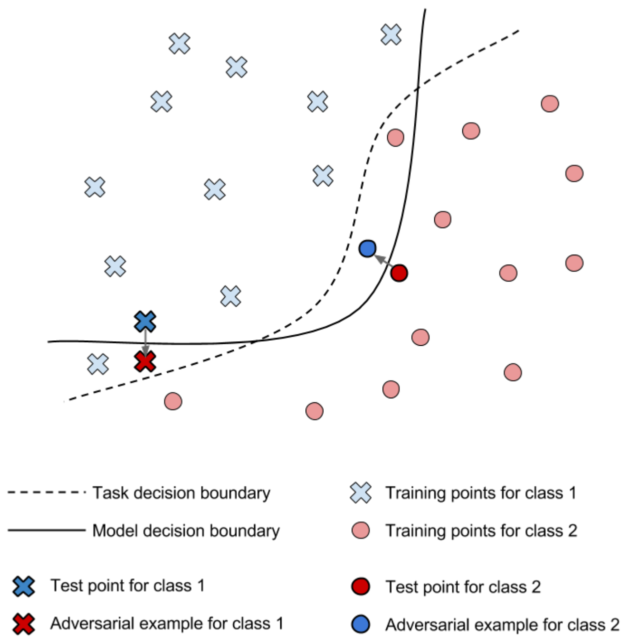


Fig. 3: Creating adversarial examples leveraging on non-linear decision boundaries of the DNN model [16].



Task decision boundary
Model decision boundary
Training points for class 1
Training points for class 2
Test point for class 1
Adversarial example for class 1
Test point for class 2
Adversarial example for class 2

Fig. 4: Adversarially tailored perturbation to make the model mis-classify the example whose class was to be predicted [9].

Ateniese et Al.'s research [2] shows a very simple technique to generate statistical information that a deployed model used for its training. Their technique used a meta-classifier to leak information which raises some serious privacy concerns with respect to confidentiality of the classifiers. It also shows how easy it is to devise method to figure out whether a given example was used for training or not.

Hinton et Al.'s paper [3] articulates various ways of poisoning the training data set for the SVM so as to maximize its prediction error. Such poisoning attacks injects crafted examples into the training set that are created by exploiting the fact that SVMs assume its input to have even data distribution. These attacks use gradient ascent on the convex loss functions to find the exact set of points that, when perturbed, will cause the greatest harm to the models performance, at inference.

Goodfellow et Al.'s paper [5] showed that the speculative causes of extreme nonlinearity of deep neural networks, perhaps combined with insufficient model averaging and insufficient regularization of the purely supervised learning problem, are unnecessary. Linear behavior in high-dimensional spaces is sufficient to cause adversarial examples (see Figure 5). This view enabled them to design a fast method of generating adversarial examples that makes adversarial training practical. They also showed that adversarial training can provide an additional regularization benefit beyond those provided by using dropout alone. Generic regularization strategies such as dropout, pre-training, and model averaging do not confer a significant reduction in a models vulnerability to adversarial examples, but changing to nonlinear model families (such as RBF networks) can do so.

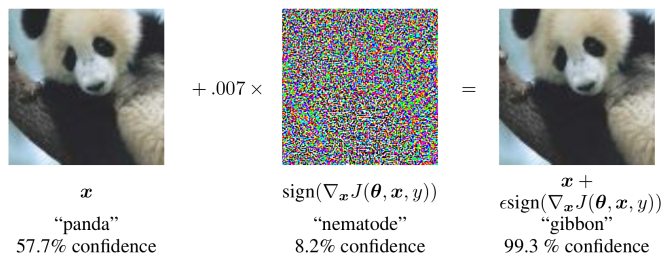

$+ .007 \times$
$=$
$\boldsymbol{x}$
"panda"
57.7% confidence
$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"nematode"
8.2% confidence
$\boldsymbol{x} + \epsilon\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"gibbon"
99.3 % confidence

Fig. 5: Adversarial examples can be generated using the simple fact that DNN are linear in higher dimensions [4]

Papernot et Al.'s work [11] shows another effective way of making attacks without knowing the training inputs or design beforehand. It creates a black-box that observes the labels that a DNN creates on chosen inputs. The attack strategy consists in training a local model to substitute for the target DNN, using inputs synthetically generated by an adversary and labeled by the target DNN. The local substitute was used to craft adversarial examples, which were misclassified by the targeted DNN such as MetaMind, an online deep learning API and other models hosted by Amazon and Google. This black-box attack strategy is capable of evading defense strategies previously found to make adversarial example crafting harder.

Papernot et Al. in another paper [13] shows how to use defensive distillation to mitigate attacks made by fast gradient sign method [5] and Jacobian-based iterative attack [14]. This technique is inspired by Hinton et Al.'s work [6], which uses knowledge compression techniques from a multitude of DNN models in the form of an ensemble.

Finally, Papernot's "Distillation as a Defense to Adversarial Perturbations against Deep Neural Networks" [15] gives a more insightful take-aways on defensive distillation by investigating its generalizability and robustness properties, and shows that dramatic protection can be achieved by using the distillation process. It reduces the gradients applied in adversarial sample creation by a factor of 10^{30} [15].

III. PROBLEM REPRESENTATION

A common way to find perturbations that force models to make wrong predictions is to compute adversarial examples. This involves using very slight perturbations that are often indistinguishable to humans, yet force machine learning models to produce drastically erroneous predictions. Fortunately, many attacks based on adversarial examples require the adversary to have knowledge of the machine learning model's parameters to solve the optimization problem for the input perturbation. However, there are also more realistic threat models where the strategy is to compute the approximate decision boundary by observing the predictions of chosen inputs, then craft the adversarial examples that will be misclassified by the observed model. In the wake of such threats, it is important to build robust ML models to tackle adversarial examples. We tried to study and reproduce some of the defense techniques that are available in the literature along with crafting adversarial examples.

To demonstrating these attacks and defenses, we used the MNIST database of handwritten digits [8]. There are 60,000 training samples and 10,000 test samples in the dataset, where each data sample is a matrix of dimensions 28×28 representing a gray-scale figure as shown in Figure 6.

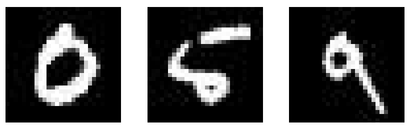


Fig. 6: Handwritten digits

IV. METHODOLOGY

The Convolutional Neural Network (CNN) architecture we used for all the experiments has the same architecture: [INPUT - CONV1 - ACTIVATION - CONV2 - ACTIVATION - CONV3 - AVTIVATION - DROPOUT - FC]. In more detail:

- INPUT will hold the raw pixels of the figure (28×28).
- CONV1 layer has 64 filters, whereas CONV2 and CONV3 have 128 filters.
- ACTIVATION function used is *Relu*, which takes element wise maximum *max(0,x)*.
- FC fully connected feed forward network computes the class scores resulting in class scores of size 1×10

We adopted the following methodology: after training the CNN, we first test with legitimate (or raw) test examples, then corrupt the test set by adding perturbations using different attack techniques. Then, we compare the accuracy scores and visual depiction of perturbations in each of the attack scenarios. Later on, we attempt to make the CNN more robust to adversarial examples by using some defense mechanisms. The results of these techniques are presented in Section V. But first, we explain the implementation for each mechanism.

A. Tools used

1) Cleverhans [10]:: A Python library to benchmark machine learning systems' vulnerability to adversarial examples.
2) Tensorflow [1]:: TensorFlow$^{\text{TM}}$ is an open source software library for numerical computation using data flow graphs.

B. ATTACKS

1) ***Fast Gradient Sign Method:*** The basic idea of Fast Gradient Sign Method (FGSM) [5] is to add a perturbation η to each pixel of input x to create an adversarial example, $\tilde{x} = x+\eta$. If η is smaller than the precision of features, then it is rational for the classifier to respond to x and $\tilde{x}$ in a similar way as long as $\| \eta \|_{\infty} < \epsilon$ where, ϵ is small enough to be discarded by human eye (in case of figures). Now consider a linear model with parameters w and an adversarial example $\tilde{x}$:

$$w^T \tilde{x} = w^T x + w^T \eta \tag{1}$$

The adversarial perturbation causes the activation to grow by $w^T \eta$. We can maximize this increase by keeping a max constraint on η nevertheless (so that the adversarial resulting figure is still the same to human perception) by assigning $\eta = sign(w)$. If w has n-dimensions then the activation $(w^T \tilde{x})$ will grow $\propto \epsilon n$. Then for large n or higher dimensions, we can make infinitesimal changes to to the input that add up to one large change to the output. In Abadi et Al.'s work [1], it is hypothesized that CNN is too linear to resist linear adversarial perturbation shown in Eqn 1.
In case of FGSM, the perturbation is calculated as below:

$$\eta = \epsilon sign(\nabla_x J(\theta, x, y)) \tag{2}$$

where, θ be CNN model parameters, x is input to model, y is the prediction of classifier for the input x, $J(\theta, x, y)$ is the loss function. It is pretty intuitive to see that higher the ϵ, higher the perturbation and greater chance of erroneous perception. As the attacker, I would not want my adversarial example to be noticeably different to the human eye, but still want to confuse the classifier. Therefore, for the perturbations shown in Figure 7, the attacker would likely pick either 0.05 or 0.1 for ϵ.

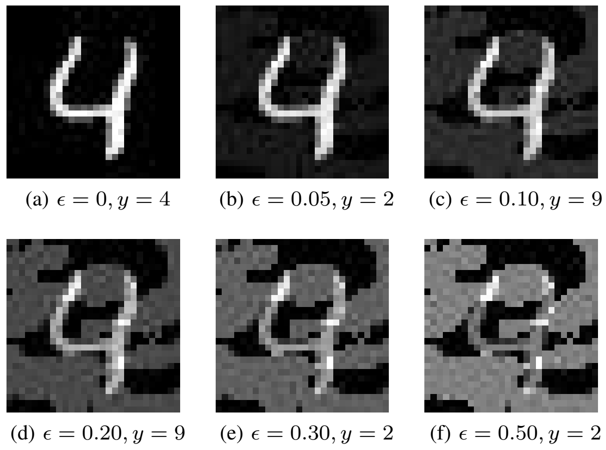

(a) $\epsilon = 0, y = 4$ (b) $\epsilon = 0.05, y = 2$ (c) $\epsilon = 0.10, y = 9$
(d) $\epsilon = 0.20, y = 9$ (e) $\epsilon = 0.30, y = 2$ (f) $\epsilon = 0.50, y = 2$

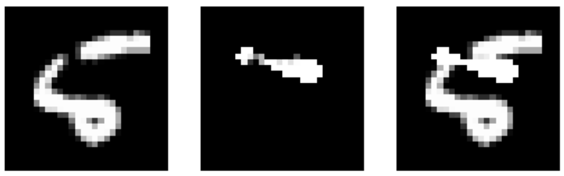
Fig. 7: Perturbed figures for different ϵ

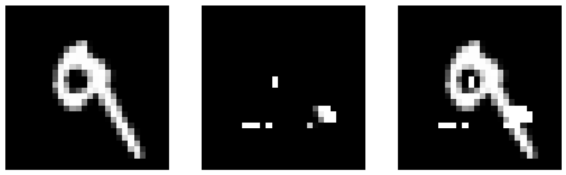
2) ***Jacobian Based Saliency Map:*** In computer vision, saliency map is an figure that shows each picture's unique qualities: it represents visual importance of a figure. In Jacobian based Saliency Map (JBSM) based attack [12], the forward derivative (Jacobian) of CNN is evaluated to construct an adversarial saliency map. This saliency map identifies the set of input pixels relevant to adversary's goals, perturbing these pixels quickly leads to desired adversarial goals. Similar to the previous attack, this approach also assumes the knowledge of CNN's architecture and weight parameters to compute the Jacobian. The computation of perturbations can be summarized as follows:

- The Jacobian component corresponding to each pair of output class and input pixel is computed. With 28 $\times$ 28 pixels, the Jacobian has $10 \times 28 \times 28$ floating point numbers.
- Where large positive value of particular Jacobian entry J_{ijk} means that, increasing the jk^{th} pixel value of figure would lead to higher probability of classifying the figure as i^{th} class and vice-versa.
- Computing Jacobian is the **first stage**, where the idea is to identify the features we should prioritize when crafting the perturbation that will result in misclassification.
- In the **second stage** of the process, we determine best pixels for our adversary goal: to misclassify an input sample in a chosen target class by applying the smallest number of perturbations possible to its input features.
- The adversarial saliency score is computed for each pixel by taking negative of product of the gradient of the target class and the sum of the gradients of all other classes.
- The saliency scores of each pixel are compared and two types of pixels are selected for perturbing: a pixel that has no impact on the target class but highly negative impact on all other classes, and a pixel that has a highly positive impact on the target class and a moderate to low positive impact on all other classes.
- In the **third stage**, we simply maximize the value of the pixel we identified in the prior stage, i.e, in case of binary figure we either set the identified pixels to 0 or 1.
- The last two stages can be visualized using Fig 8. Where, left most figure is input digit, middle figure shows the pixels selected using saliency scores for perturbation, right most figure shows the perturbed figure.

In Fig 9, a 10 $\times$ 10 grid is presented, where an ij^{th} figure represents that input class is i and figure is perturbed so that it can be misclassified as j (target class).

3) ***Black Box:*** Both the attack strategies we have seen so far require the adversary to know the internals of the model, but the Black Box [11] attack strategy as the name suggests


(a) input = 5, target = 8


(b) input = 9, target = 6

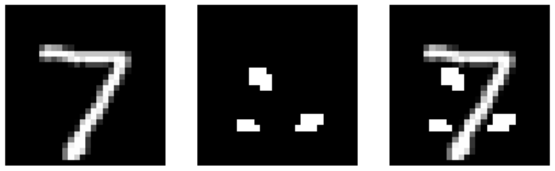

(c) input = 7, target = 6

Fig. 8: Perturbations using JSMA

doesn't require knowledge of the parameters of the model, all it needs is to observe the labels given CNN for chosen inputs. The approach of this attack is to consists of training a local model to substitute for the target CNN, and uses local substitute to craft adversarial examples. The steps performed to implement Black Box attack can be illustrated as follows:

- **Initialization**: Adversary collects small set S_0 of inputs, representative of input domain, 10 figures of handwritten digits 0 through 9. These examples do not need to originate from the same distribution where the targeted Oracle O was trained.
- **Architecture Selection**: The adversary selects an architecture to be trained as the substitute F.
- The adversary iteratively trains the substitutes F_i by repeating the following for $i = 1, 2, ..., i_{max}$:
  - **Labeling**: Query for the labels $O(x) \forall x \in S_i$ from the Oracle O.
  - **Training**: Train the substitute F_i using the data (x, $O(x)$).
  - **Augmentation**: Apply *Jacobian-based dataset Augmentation* technique to produce a larger substitute dataset S_{i+1}.
- **Adversarial sample crafting**: Once a substitute is trained the adversarial samples can be crafted by using either of the two attack techniques detailed in previous two Sections IV-B.1, IV-B.2.

After training the substitute CNN model, we crafted our adversarial samples using Fast Gradient Sign Method (with ϵ = 0.3) whose results can be visualized in Figure 10 for each of the handwritten digit.

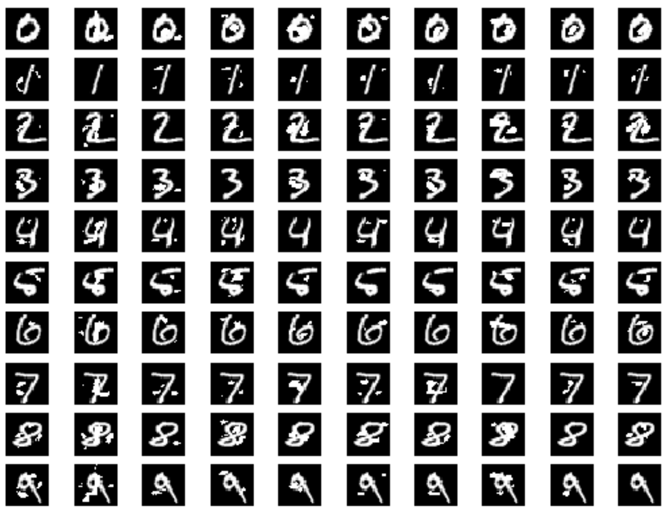


Fig. 9: JSMA attack: Grid Visualization

C. DEFENSES

Most adversarial example construction techniques use the gradient of the model to make an attack. In other words, they look at a picture of a handwritten digit "4", and perform tests to find out the directions that can be taken in the picture space to increase the probability of the "5" class, and then give the image a small push (in other words, they perturb the input) in that direction. The new, modified figure will now be mis-recognized as a "5". We want to create defenses that will counter this a much as possible.

1) ADVERSARIAL TRAINING: The two types of defensive strategies are: (1) *reactive* where one seeks to detect adversarial examples when it happens, (2) *proactive* where one makes the model itself more robust. Adversarial training, as the name suggests tries to make the model more robust by training with adversarial examples. Simply put, it is a brute force solution where the defense is to simply generate many adversarial examples using both FGSM and JSMA, and explicitly train the Convolution Neural Network to not be fooled by them. As expected, the more epochs of training with adversarial examples used, the better the accuracy on adversarial and legitimate examples. We used 6 epochs in our implementation. To cross validate the robustness we trained and tested model with different adversarial examples generated using different ϵ in FGSM approach.

2) DEFENSIVE DISTILLATION: There are many ways to smoothen the decision boundary of the classifier. We use variation in temperature constant in the softmax function to do so. Figure 11 gives a high level overview of our implementation as it was suggested in Papernot et Al.'s work [15].
First an initial network F is trained with a softmax temperature of T. The output probabilities are fed into the distilled network on the same data.

To implement this defense, we used a NN with one hidden layer and 10 perceptrons in both the protective and to be

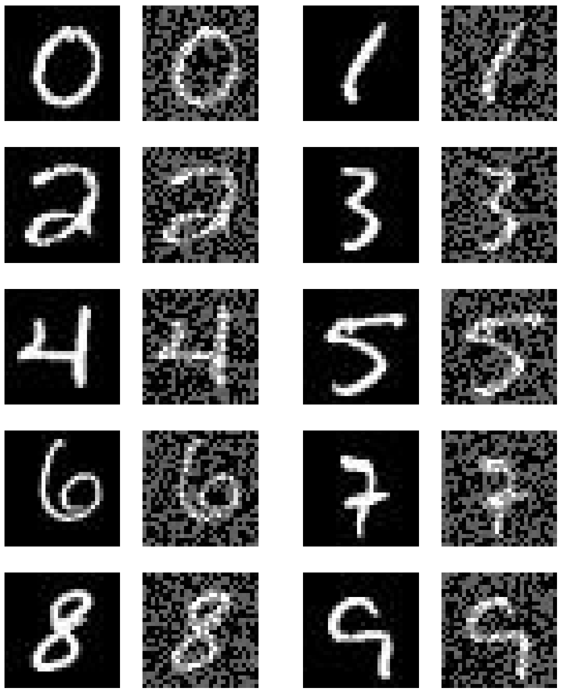


Fig. 10: Black Box attack: Initialized inputs and Crafted Adversarial examples

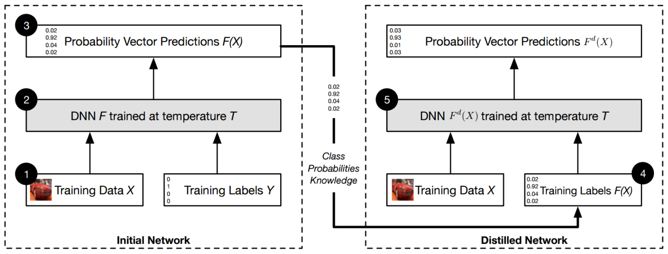


Fig. 11: Two separate networks are used instead of just one where the first one essentially smoothens out the class vector so that attacks like JSMA and FGSM can be mitigated [15].

protected models. We introduced the notion of temperature in the softmax function as suggested by [15], so that our softmax now becomes

$$F(X) = \frac{e^{\frac{z_i(X)}{T}}}{\sum_{l=0}^{N-1} e^{\frac{z_i(X)}{T}}} \quad \forall i \in \{0, 1, ...N-1\} \tag{3}$$

The first layer smoothens the training labels which gets used by the true predictive model which we wish to defend.

V. EMPIRICAL RESULTS

A. ATTACKS

We experimented with three attacks, and in all three cases there is significant drop in accuracy of the predictions by CNN. The summary of the results is shown in Table I where it compares accuracies of the CNN model with legitimate and adversarial examples crafted by the three methods.

TABLE I: Accuracy of different methods on Legitimate and Adversarial Test examples

	Legitimate examples	Adversarial examples
FGSM	97.26 %	2 %
JBSM	93.05 %	4 %
Black Box	95.20 %	8.16 %

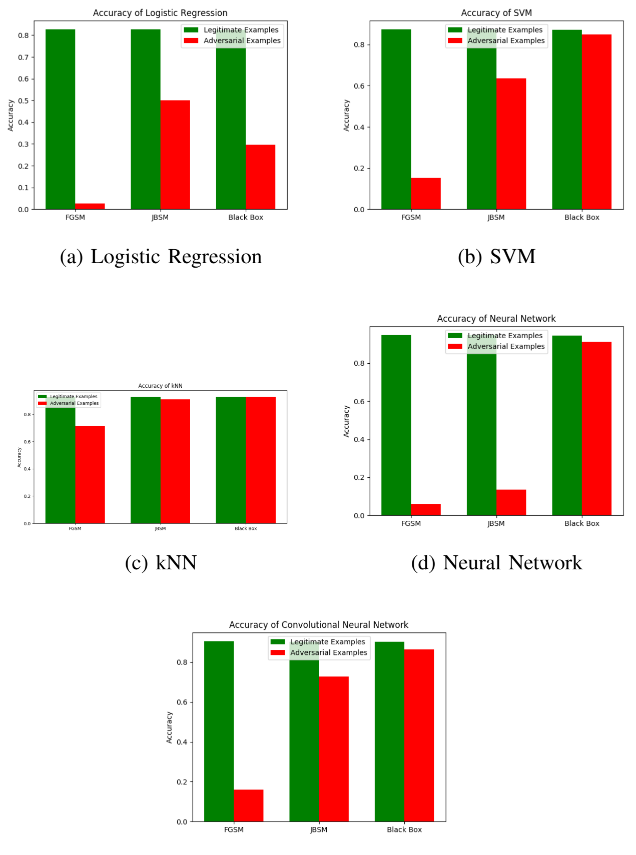
Please note that the accuracy of JBSM method on adversarial examples is the average accuracy rate of adversarial examples. Since, in JBSM adversarial examples are crafted for a specific target we took the average of accuracies for each particular target class and obtained 4%. The ϵ used for perturbation in FGSM and Black Box approach is 0.2 and the average rate of perturbation in case of JBSM is 3%. The visual appeal of these perturbations can be seen in Figures 7,9,10.

An important metric used to measure the performance of attack mechanisms is transferability. Both the Fast Gradient Sign Method and Jacobian Based Saliency Map approaches require internal details of the models to craft adversarial examples, therefore it would be interesting to see the performance of adversarial examples on completely different models, ones where the adversary do not consider when crafting the malicious examples. To find out, we tested these adversarial examples on Logistic Regression, SVM, KNN, Feed-Forward Neural Network and CNN of different architectures. We also chose the hyper-parameters and architectures of these models randomly, instead of selecting those that yield the best performance, since the model that we used to test legitimate and adversarial samples is still going to be the same, and we are more interested in the comparison rather than the actual performance. The comparison is shown in Figure 12 where, each bar graph depicts the results of particular machine learning method as noted, first two bars belong to accuracies of FGSM, second ones belong to JBSM, and the third one to Black Box attack. Also the green bar shows accuracy on legitimate examples for reference and the red bar shows accuracy with adversarial examples generated by respective methods.

B. DEFENSES

It is established in literature [11] that adversarial examples are hard to defend against because it is hard to construct a theoretical model of the adversarial example crafting process. This is evident when we experimented few of the defense mechanisms. In adversarial training the CNN model is trained with adversarial examples along with regular training data as this is expected to increase the robustness of the model. We trained our CNN model with adversarial examples generated by FGSM method with $\epsilon = 0.2$ and tested with adversarial examples crafted by different mechanisms. The results of this experiment (after 10 epochs) are presented in the Table III.

The result generated from using defensive distillation is shown in table II.



(a) Logistic Regression (b) SVM

(c) kNN (d) Neural Network

(e) Convolutional Neural Network

Fig. 12: Comparison of accuracies by different models with legitimate, adversarial examples

TABLE II: Result from using Defensive Distillation

	Legitimate examples	Adversarial examples
With Defensive Distillation	67	23
Without defensive distillation	91	8

It is apparent from the results that, although there is a compromise on the performance on genuine data by a defended model, the defense gives considerable levels of security against the gradient-based attacks.

VI. DISCUSSION

A. ANALYSIS

It is evident from the results shown in Table I that the attack mechanisms are very effective on models that are used to calculate the vicious gradients. Even the Black Box attack, which works on a substitute model, has an successful attack of 8%.

However, when it comes to transferability of these adversarial examples to completely different machine learning models, the adversarial examples perform very differently, as shown in Figure 12. The adversarial examples generated by FGSM is clearly the winner by significantly reducing accuracy in all the models except kNN. Additionally, the Black Box attack, which is much more realistic, does not show enough transferability to cause damage to machine learning model predictions except that of Logistic Regression (Fig 12.a). One very interesting observation from the experiment is that, surprisingly, adversarial examples do not really affect the kNN's predictions. This could be because all other methods (Logistic Regression, SVM, Neural Network, Convolutional Neural Network) have inherent linearity in them which these adversarial attacks exploit.

TABLE III: Effect of Adversarial Training on examples generated by different mechanisms

Without Adversarial Training	Legitimate	FGSM	JSMA	Black Box
97.94 %	79.72 %	10.55 %	22.22 %	60.85 %

Speaking of defense mechanisms, it is clearly proven from the results in Table III that generating attacks is much easier than setting up defenses against such attacks. In fact, when the model is trained with adversarial examples to make it more robust, the accuracy on legitimate examples drops from 97.94% to 79.72% during adversarial training and from 91% to 67% during defense using defensive distillation. The gains on accuracy with adversarial examples, FGSM (2% $\rightarrow$ 10.55%), JBSM (4% $\rightarrow$ 22.22%), Black Box (8.16% $\rightarrow$ 60.85%), Defensive-Distillation(8% to 23%) in comparison.

B. CONTRIBUTION TO KNOWLEDGE

We have conducted the study in an investigative and experimental fashion and got some interesting results. We think with the present state of the art Machine Learning attack mechanisms, there isn't a need to press the panic button yet. Even though FGSM does have enough transferability to damage the accuracies of machine learning models, there isn't a strong economic incentive to do attacks with FGSM as it doesn't force the classifier to mis-classify as a particular target class instead just makes it mis-classify as any other incorrect class. The JBSM attack mechanism, which is aimed at creating adversarial examples for a target class, doesn't perform well with a different CNN architecture (Fig 12.e), and the Black Box attack was defended by Adversarial Training with a reasonable results (Table III).
Perhaps the biggest takeaway from our study is discovery that none of these attacks have any affect on kNN (Fig 12.e), which gives a hint on the direction to be explored for defense mechanisms against adversarial examples.

C. LIMITATIONS

We used only the MNIST dataset for experiments and lot of literature in the area also focuses on image based datasets. It would be interesting to see the results of adversarial attacks with other type of datasets other than images. Also, the attacks were all created in a post-training scenario, and we didn't consider the poisoning of training set scenarios.

D. Future Work

With the results we have obtained through our experiments and implementations, we wonder if it is possible to conceive a perfect defense that allows almost no decrease to the classification of legitimate inputs, but that will at the same time defend against adversarial attacks, in somewhat of a human fashion. Perhaps with more research and a combination of different ways to identify features, we can construct a model that is the best of both worlds.

It was also noticed that different researchers have profound differences in their understanding of the reason behind blind-spots, especially in deep networks. This calls for future intensive research in making reasonable and valid hypothesis on the behavior of deep learning models.

Other domains which might get affected by similar attacks should not be neglected as well. Feiri et Al. [7] and Papernot et Al.[15] have mentioned similar attacks that have possible negative effects on readings obtained from LiDAR or other light sensors, which have now become a core component of the sensors in autonomous vehicles, and can cause catastrophic consequences if not addressed.

VII. Statement of Contributions

Srikanth Pasumarthy worked on implementing the attacks and defense techniques.

Sanjay Thakur implemented defensive-distillation and helped Xoey with creating the content for the video.

Xoey Zhang created the video and helped with writing the report.

We hereby state that all the work presented in this report is that of the authors.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] Giuseppe Ateniese, Giovanni Felici, Luigi V. Mancini, Angelo Spognardi, Antonio Villani, and Domenico Vitali. Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers. *CoRR*, abs/1306.4447, 2013.

[3] B. Biggio, B. Nelson, and P. Laskov. Poisoning Attacks against Support Vector Machines. *ArXiv e-prints*, June 2012.

[4] Ian Goodfellow and Nicolas Papernot. Is attacking machine learning easier than defending it?, February 2017.

[5] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[6] G. Hinton, O. Vinyals, and J. Dean. Distilling the Knowledge in a Neural Network. *ArXiv e-prints*, March 2015.

[7] Michael Feiri Frank Kargl Jonathan Petit, Bas Stottelaar. Remote attacks on automated vehicles sensors: Experiments on camera and lidar, February 2017.

[8] Yann LeCun, Corinna Cortes, and Christopher J.C. Burges. The mnist database of handwritten digits. http://yann.lecun.com/exdb/mnist/.

[9] Nicolas Papernot and Ian Goodfellow. Breaking things is easy, February 2017.

[10] Nicolas Papernot, Ian Goodfellow, Ryan Sheatsley, Reuben Feinman, and Patrick McDaniel. cleverhans v1.0.0: an adversarial machine learning library. *arXiv preprint arXiv:1610.00768*, 2016.

[11] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pages 506–519. ACM, 2017.

[12] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*, pages 372–387. IEEE, 2016.

[13] Nicolas Papernot and Patrick D. McDaniel. On the effectiveness of defensive distillation. *CoRR*, abs/1607.05113, 2016.

[14] Nicolas Papernot, Patrick D. McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. *CoRR*, abs/1511.07528, 2015.

[15] Nicolas Papernot, Patrick D. McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. *CoRR*, abs/1511.04508, 2015.

[16] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *CoRR*, abs/1312.6199, 2013.