CMPT 489 Project: Adversarial Machine Learning in Computer Vision

SHRAY KHANNA, Simon Fraser University
HAIPENG LI, Simon Fraser University
JOHN SWEENEY, Simon Fraser University

Adversarial machine learning is a very important and cutting-edge topic in the field of machine learning. Adversarial machine learning is a technique that attempts to fool machine learning models through malicious input. This paper conducts a survey, implementation, and defense against some of the most popular theoretical adversarial machine learning attacks. In the review section, there is an analysis and explanation of how an attack can theoretically be implemented. Implementation and the defense against implemented is conducted for selected attack methods. The methods analyzed and implemented demonstrate how differently models can act from what their designers intended under adversarial situations.

Additional Key Words and Phrases: adversarial machine learning, neural networks, computer vision, system security

ACM Reference Format:
Shray Khanna, Haipeng Li, and John Sweeney. 2019. CMPT 489 Project: Adversarial Machine Learning in Computer Vision. 1, 1 (December 2019), 11 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

State-of-the-art deep neural networks have achieved major breakthroughs in many research areas such as Computer Vision[13][12][9], Natural Language Processing[10][21], and Fraud Detection. [18] Machine learning techniques were initially designed for controlled environments in which the training and test data are from the same statistical distribution. In the real world, adversaries can manipulate input data to exploit vulnerabilities of machine learning algorithms thus compromising the security of the machine learning system. [7] Adversarial machine learning is the technique to fool models with malicious input.

Many implementations of deep neural networks in the real world are vulnerable to adversarial machine learning attacks. This paper focuses on adversarial machine learning attacks in the field of Computer Vision. Computer vision models are increasingly deployed in real-world environments like autonomous vehicles and next-generation employee-less stores, where robustness and security are crucial. In the case of a successful adversarial machine learning attack on an autonomous vehicle, there may be fatalities. This paper analyzes and implements published adversarial attacks on object classifications. In addition, novel attack defenses are implemented and explored.

Authors' addresses: Shray Khanna, Simon Fraser University, skhanna@sfu.ca; Haipeng Li, Simon Fraser University, haipengl@sfu.ca; John Sweeney, Simon Fraser University, sweeney@sfu.ca.

2 REVIEWS OF ATTACKS ON OBJECT CLASSIFICATION

2.1 Fast Gradient Sign Method (FGSM)

Machine learning models misclassify inputs even on a high test-score when small worst-case perturbations are fed to the model. The primary cause of Neural network behaving this way is the linearity in them. Szegedy et al [8] suggested a method to to efficiently compute an adversarial perturbation for a given image. Generally, machine learning models misclassify examples that are only a little different form the data trained on them. A linearity in high dimension space can cause adversarial examples.

Adversarial training gives a bonus to the regularization to control overfitting by adding a layer on top of strategies like dropout, pretraining and model averaging. A good performance on test set does not mean the model performs naturally well, when exposed to a fake example these models generally fail. This is upsetting as the most common way in computer vision is to use CNN features where Euclidean distance gives the perceptual distance. This resemblance is flawed as when images with immensely small distance lies in the wrong classes in network's representation.

Linear explanation of the adversarial examples comes from the linear models. The precision of the features are limited due to which a linear classifier responds same way to an input x as that of an adversarial input $x' = X + N$, where N is smaller than the precision of features. The classifier then assigns the same class to input x as that of input x'. Now, when weight vector gives dot product with the input to give a score for classifier we get:

$$W^T X' = W^T X + W^T .\eta$$

The perturbation now grows by $w^T .\eta$. To maximize this assign $\eta = sign(w)$. This way the adversarial example grows linearly with N (N being the input size). Szegedy et al[8] gives a way to compute these perturbations:

$$\eta = \epsilon .sign(\nabla_x J(\theta, x; y))$$

Where η is the perturbation, x the input to the model, y the targets associated with x (for machine learning tasks that have targets) and $J(\theta; x; y)$ be the cost used to train the neural network. They suggested to linearize the cost function around the current value of θ. This method is referred as fast gradient sign method.
This method generates the adversarial examples by FGSM which exploit the "linearity" of deep neural network models which were thought as non-linear in high dimensional space.

The authors also developed a "one-step target class" where instead of using true label l they used l_t which is the output class with least probability I_c. The authors also showed that the robustness of DNN is improved with use of FGSM and its variants. Training with adversarial objective function was found to be a good regularization (Yuan et al [22]). These results show that linearity is present even if it is not seen clearly and also models which optimize easily can be perturbed easily.

2.2 Basic Iterative Method (BIM) and Iterative Least Likely Class Method (ILCM)

Kurakin et al [2] proposed two methods for adversarial attacks on images which are basically an improvement of Fast Gradient Sign Method.
The idea behind Basic Iterative Method (BIM)[2] is to perturb images by taking a single large step in the direction that

increases the loss of the classier. This gradient method is more similar to gradient ascent as compare to descent.
The one-step methods perturb images by taking a single large step in the direction that increases the loss of the classier (i.e. one-step gradient ascent).As presented by Chakraborty et al[4] this method is an intuitive extension to iteratively take multiple small steps. while adjusting the direction after each step. The Basic Iterative Method does exactly that, and iteratively computes the following:

$$x_0^{adv} = x, \quad x_{N+1}^{adv} = Clip_{x,\epsilon}[x_N^{adv} + \alpha.sign(\nabla_x J(x_N^{adv}, y_{true}))]$$

where: x is Clean Input Image; x_i^{adv} - Adversarial Image at i^{th} step; J is the Loss Function; y_{true} is the Model Output for x; ϵ is the Tunable Parameter and α is the Step Size

The number of iterations are $min(\epsilon + 4, 1.25\epsilon)$. This method does not typically rely on any approximation of the model and produces additional harmful adversarial examples when run for more iterations.

Similar to extending the FGSM[8] to its 'one-step target class' variation, Kurakin et al.[2] also extended BIM to Iterative Least-likely Class Method (ILCM). The adversarial examples generated by the ILCM method has been shown to seriously affect the classication accuracy of a modern deep architecture Inception v3, even for very small values of ϵ, e.g.ϵ < 16.
The equation of ILCM is given by:

$$x_0^{adv} = x, \quad x_{N+1}^{adv} = Clip_{x,\epsilon}[x_N^{adv} - \alpha.sign(\nabla_x J(x_N^{adv}, y_{LL}))]$$

where: x is Clean Input Image; x_i^{adv} is the Adversarial Image at i^{th} step; J is the Loss Function; y_{LL} is for Least Likely Class; ϵ is the Tunable Parameter and α is the Step Size

For desired class, authors chose least likely class denoted by y_{LL} and is given by:

$$y_{LL} = arg\min_y[P(y/x)]$$

For a well-trained classifier, the least-likely class is usually highly dissimilar from the true class, so ILCM attack results in more interesting mistakes, such as mistaking a cat for a flower.

Loss in FGSM, BIM and ILCM is given by:

$$Loss = \frac{1}{(m-k) + \lambda.k} \left[\sum_{i \epsilon CLEAN} L(X_i/y_i) + \lambda \sum_{i \epsilon ADV} L(X_i^{adv}/y_i) \right]$$

where:
$L(X|y)$ is a loss on a single example X with true class y;m is total number of training examples in mini-batch; k is the number of training examples in mini-batch; λ is the parameter to control the relative weight of adversarial examples in the loss; The values we take are: $\lambda = 0.3$, $m = 32$ and $k = 16$

2.3 One pixel attack for fooling deep neural networks

Su et al [20] proposed a novel black-box method for generating one-pixel adversarial perturbations that work on any neural network. The attack relies on an algorithm called differential evolution (DE), which is a population based stochastic function minimizer for solving complex multi-modal optimization problems. [19]

The idea behind the one pixel attack is to maximize the classifier's score for an adversary's target class given the ability to only change a single pixel. The equation could also be written as an N pixel attack. The single pixel attack is described by the following equation where f is the image classifier, adv is the target class and vector e(x) is an additive adversarial perturbation according to x. [20]

$$\max_{e(x)} f_{adv}(x + e(x))$$

$$subject\ to\ ||\ e(x)\ ||= 1$$

The perturbation is encoded into an array for the candidate solution, which is optimized by differential evolution. [20] At each iteration the set of candidate solutions are produced by the DE formula:

$$x_i(g+1) = x_{r1}(g) + F(x_{r2}g - x_{r3}g), r1 \neq r2 \neq r3$$

At each time step, the best candidate solutions will be chosen up to a defined value.

One pixel attacks for fooling deep neural networks have the advantage of being difficult or even impossible for a human to discern the difference between the original and adversarial image. In addition, there does not currently exist a true defense that does not have a computational cost. [20]

2.4 Decision-Based Adversarial Attacks: Reliable Attacks Against Black-Box Machine Learning Models

Brendel et al [3] propose the Boundary Attack, a decision-based attack that begins from a large adversarial perturbation then to reduces the perturbation over multiple iterations while remaining adversarial. The attack is simple and unlike FGSM [8], DeepFool [15] and the One Pixel Attack [20] are easy to deploy against real-world examples. In addition, unlike these attacks, the boundary attack is a true black-box attack requiring no input from inside the model. Most prominent examples are web services that run user uploaded-images through a neural network, such as Google Images. The boundary attack is a decision-based attack. [3] There are no complete defenses against decision-based attacks because the attack can exploit all types of vulnerabilities; it is an extremely flexible attack as long as an initial adversarial image is locatable. Locating in initial adversarial image for the majority of models is trivial.

The boundary attack as its name describes tests the model for its boundary; how close can the the algorithm get to the original image, using a variety of methods, while the image remains adversarial.

The attack can be described by the following algorithm: [3]

Data: original image o, adversarial criterion c(.), decision of model d(.)
Result: adversarial example $\tilde{o}$ such that the distance $d(o, \tilde{o}) =|| o - \tilde{o} ||_2^2$
while k < maximum number of steps do
  draw random perturbation from proposal distribution $n_k \sim P(\tilde{o}^{k-1})$;
  if $\tilde{o}^{k-1} + n_k$ is adversarial then
    set $\tilde{o}^k = \tilde{o}^{k-1} + n_k$
  else

set $tildeo^k = \tilde{o}^{k-1}$
end
k = k + 1
end

The attack needs to be initialized from a state that is already adversarial. In addition, at every time-step we must efficiency draw possibilities from a proposal distribution. These possibilities much be within the input domain and reduce the distance of the perturbed image from the original image.

2.5 Universal Adversarial Perturbations

So far we have discussed methods like FGSM [8], DeepFool [15] and One-pixel Attack [20] compute perturbation on a single image. Moosavi-Dezfooli et al proposed Universal Adversarial Perturbations [14] to compute a "small universal perturbation" being able to fool the state-of-the-art neural networks on a set of images with high fooling rate. The perturbation is what the authors call "quasi-imperceptible" for human vision system.

Moosavi-Dezfooli et al proposes an iterative algorithm to compute the universal perturbation. For the formal definition of perturbation, we assume that the clean image is sampled from a distribution μ. A perturbation v is "universal" if it satisfies the following constraints:

$$P_{x\sim\mu}(\hat{k}(x+v) \neq \hat{k}(x)) \geq 1-\delta \quad s.t. \quad ||\rho||_p \leq \xi \tag{1}$$

where P denotes the probability, $\delta \in (0,1]$ quantifies the desired fooling rate for all images sampled from the distribution μ, $||.||_p$ denotes the l_p norm and ξ controls the magnitude of the perturbation vector v.

The authors compute the universal perturbation by limiting its l_2 norm and l_∞ norm, and show that the upper bound of its norm is 4% perturbation of the respective image norm, which has achieved a significant spoofing ratio of 0.8 or higher for the state-of-the-art image classifier. Their iterative method of calculating perturbations is related to the DeepFool [15], which incrementally pushes a data point (or an image) to the decision boundary of its class. However, in this case, all training data points are pushed in turn to their respective decision boundaries, and the perturbations calculated on all images gradually accumulate by each time the accumulator is back-projected onto the desired sphere of radius p.

3 IMPLEMENTATING FGSM[8]

Fast Gradient Sign Method has the lowest time complexity as compared to other methods that we have surveyed. The reason behind that is because it deals with the sign of gradient and does not care about magnitude. This makes it faster at execution. The scaling factor ϵ (epsilon) is responsible for increasing or decreasing the noise (perturbation) in the input image. We tested this method on various scenarios with a wide variety of classes. Through our experiments, we've found out that this method has a limitation and performs differently for every state-of-the-art model. Fig 1 shows how this attack works on an input image with a small perturbation.

As you see that original image of Pig was reported with actual class (Pig) with a confidence of 60% but when we add a small perturbation, the model assigns it with the class Airliner with a confidence level of 99.4%. The model in the above example used is Resnet18.

We tested this method against Resnet18, Resnet101, Alexnet and Inception_v3 which are currently state-of-the-art models used for object classification and detection. After running a variety of test images we found that model has

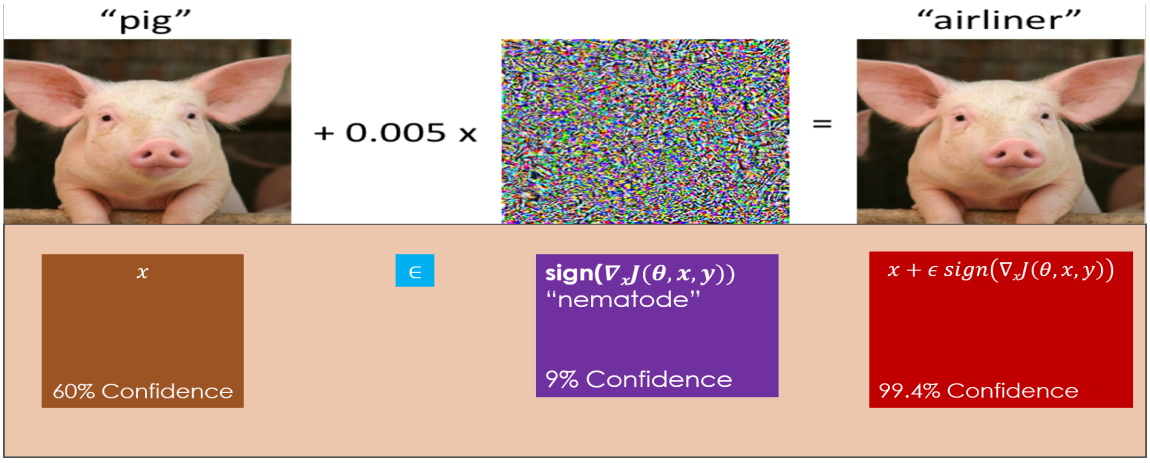


Fig. 1. FGSM attack on an input image of Pig. FGSM depends on the sign of the gradient of cost function and not the magnitude. Scale your perturbations using ϵ (Epsilon).

limitations to Deeper Layered Networks such as Resnet101. One reason behind this is that it computes perturbation for the entire input at once and thus, is not able to learn anything. We've shown a basic comparison of it's working on test images in Fig2.

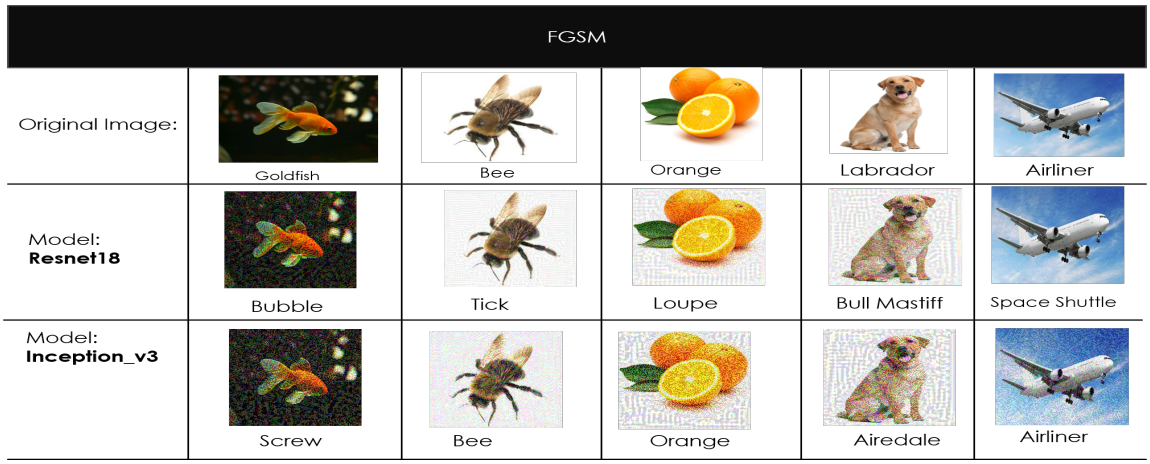


Fig. 2. FGSM attack on Resnet18 (model trained on 18 Residual layers) and Inception_v3 (better and deeper layered model).

Fig 2 shows that even the perturbation added to the input image to show the noisy images which fooled the model. As you can see we had to add a lot of perturbation for Inception_v3 as compared to Resnet18. When we used Resnet18, the amount of perturbation needed was relatively low for instance images of Bee and Airliner are almost the same. We have used the maximum possible value for ϵ in images of Orange and Airliner, after this value the image becomes very hazy. The Inception_v3 model is still predicting the correct class even after having the maximum perturbation.

4 UNIVERSAL PERTURBATION ATTACK AND DEFENSE

4.1 Overview

"Universal Perturbations" can be applied to "any" image to trick a state-of-the-art network classifier, seriously threatening the success of deep learning in practice. We first implement a Universal Perturbation Attack and then propose a defense framework to effectively protect the network from such attack. Our approach learns a Perturbation Filtering Network (PFN) as filtering layers to a targeted model, such that no modification needs to be done on the target model. The PFN is trained using real and synthetic image-agnostic perturbations. Our experiments show that our framework can reduce the fooling rate of unseen adversarial perturbations from 81.1% to as low as 2.0%.

4.2 Proposed Approach

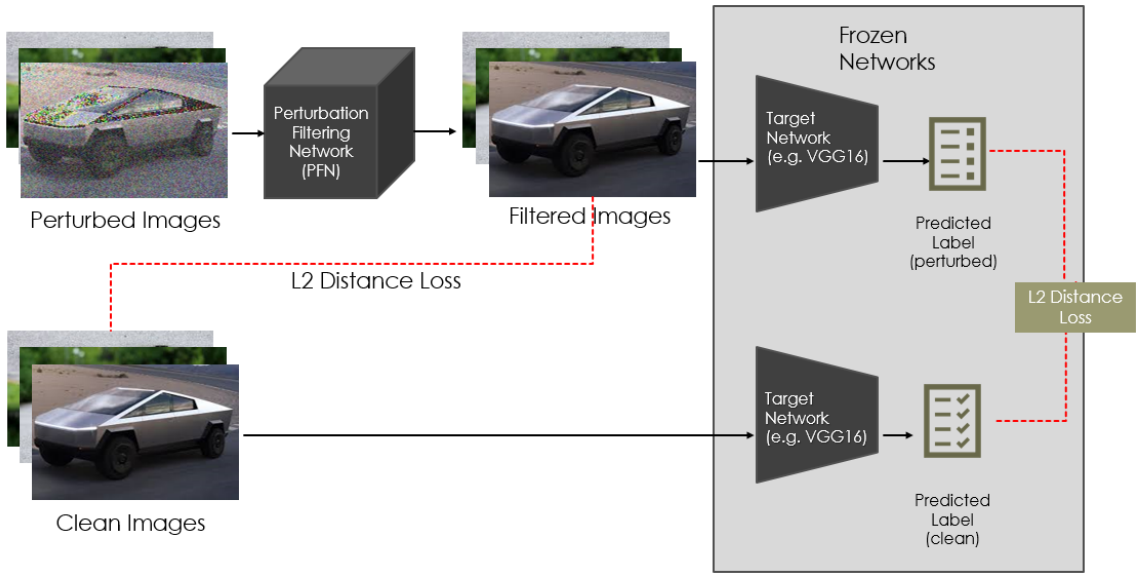


Fig. 3. We propose PFN: From the clean data, image-agnostic perturbations are computed and augmented with the synthetic perturbations. Both the clean and perturbed images are fed into the Perturbed Filtering Network (PFN). PFN is trained by connecting it to the first layer of the target network in order to keep the parameters of the target network unchanged during training.

We propose Perturbation Filtering Network (PFN), which is trained as perturbation filtering layers attached to the first layer of the targeted network. We define the combination of PFN and target network as joint network. Fig. 3 shows the network structure of joint network. We train the PFN using both clean and adversarial examples to ensure that the image transformation learned by our network is not biased towards the adversarial examples.

We use a combined loss function: L_{rec} and L_{lab}. L_{rec} is the loss between the output image $\hat{\mathbf{I}}_\mathbf{i}$ of PFN (we call it filtered image) and the clean image $\mathbf{I}_\mathbf{i}$ in pixel-level L2 distance. L_{lab} is the L2 loss between the predicted probability label vector $\mathbf{p}_\mathbf{i}$, $\hat{\mathbf{p}}_\mathbf{i}$ by target model and joint model respectively. These losses are defined as follows:

$$L_{rec} = \frac{1}{N}\sum_{i=1}^{N}\sqrt{(\mathbf{I}_\mathbf{i} - \hat{\mathbf{I}}_\mathbf{i})^2} \tag{2}$$

$$L_{lab} = \frac{1}{N}\sum_{i=1}^{N}\sqrt{(\mathbf{p_i} - \hat{\mathbf{p}}_\mathbf{i})^2} \tag{3}$$

Unlike the defense network proposed by N Akhtar [1], which minimizes the cross-entropy loss [6] of the predicted labels, we minimize the sum of L_{rec} and L_{lab} during training using weight parameters α and β to control the contribution of two losses:

$$L_{total} = \alpha L_{rec} + \beta L_{lab} \tag{4}$$

We train the PFN using both clean and adversarial examples to ensure that the image transformation learned by our network is not biased towards the adversarial examples. The PFN is implemented as 10-ResNet blocks [9] sandwiched by convolution layers. The 224x224x3 input image is fed to Conv 3x3, stride = 1, feature maps = 64, "same" convolution; followed by 10 ResNet blocks, where each block consists of two convolution layers with ReLU activations [16]. We minimize Eq. 4 with ADAM optimizer [11]. We set the learning rate to 0.005. We used mini-batch size of 16, and trained the PFN for a given targeted network for at least 3 epochs.

5 ATTACK & DEFENSE EXPERIMENTS

5.1 Dataset

Our dataset is based on Natural Images [17] - a compiled dataset of 6899 images from 8 distinct classes (airplane, car, cat, dog, flower, fruit, motorbike, person), which is used for training a classifier as target model. When we do the attack and defense, we use a subset of Natural Images dataset which contains 1600 images for training (200 images per class) and around 640 images (around 80 images per class) for testing. Note that this is a relatively small dataset considering our limited computing power. In the future we plan to use the original ImageNet dataset or its subset for the experiments.

5.2 Train a Classifier as Target Model

We conduct experiments on two target models: In experiment 1 we attack then the Resnet18 M_1 [9] pre-trained on ImageNet [5] , which has 1000 classes. In experiment 2 we trained Resnet18 M_2 on our dataset - Natural Images which has 8 classes. We are able to train M_2 with 99% accuracy on Natural Images dataset. Note that M_1 is not trained on Natural Images dataset, so each image will be classified as one of 1000 classes from ImageNet, which is not be as accurate as M_2. We show the difference in next section.

5.3 Compute Universal Perturbation

We compute 12 different Universal Perturbation on a subset of Natural Images dataset with 1600 training images against both M_1 and M_2 using our implementation of [14]. We notice that the attack on M_1 is much easier than M_2 with 92% fooling rate in 3 epochs for the former but only 81.1% fooling rate in 8 epochs for the latter. This implies that the Universal Perturbation Attack has better success rate for a model with more output classes. We show the perturbed images for experiment 2 in the second column of Fig 4.

5.4 Defense Against Universal Perturbation

We train the PFN against the attack on model M_1 and M_2 by minimizing Eq. 4 where $\alpha = 40$ and $\beta = 1$. 10 different Universal Perturbations are used for training and 2 for testing. We evaluated the performance by *fooling rate*, which is

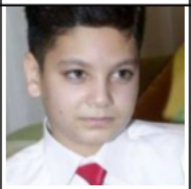
	Original	Perturbed	Trained with L_{total}	Trained with L_{rec}	Trained with L_{lab}
Fooling Rate	0%	81.1%	2.0%	3.3%	5.1%
Images, predicted labels and confidence	 Airplane 0.97	 Fruit 0.54	 Airplane 0.75	 Airplane 0.73	 Airplane 0.62
Images, predicted labels and confidence	 Person 0.99	 Fruit 0.67	 Person 0.99	 Person 0.94	 Person 0.90

Fig. 4. Representative examples of the defense results. First, second columns show the original images and images with universal perturbation. The last three columns compare the filtered image quality, fooling rates and prediction by our PFN trained with L_{total}, L_{rec}, L_{lab} respectively.

the percentage of perturbed images successfully fool the target network by making it predict wrong labels. In experiment 1 we are able to reduce the fooling rate from 92% to 33%, and in experiment 2 from 81.1% to 2.0% in 3 epochs. We can see our methods are particularly effective on experiment 2 with the ResNet trained on our small dataset. As mentioned in previous section, this dataset is hard to fool using Universal Perturbation. So it is not surprising that it is easy to achieve such a good defense result. Some representative examples are demonstrated in the first three columns in Fig 4. Note that the fooling rates are tested on testing dataset, which means the network never saw the perturbation nor the images before during testing phase, which shows that our defense framework has very good generalizability.

We conduct an ablation study to validate each term of our loss function in Eq. 4, shown in last two columns in Fig 4. Comparing the three methods with different loss function, we can clearly see the superiority of our loss function L_{total}, which produces better image quality which is the closest to the original image, lower fooling rate and higher confidence predicted by the target model. When trained without L_{rec} (5th column in Fig 4), the color filtered images are brighter than original images as a result of lacking the constraint of recovering the original images in the loss function. When trained with only L_{rec} without L_{lab} (4th column in Fig 4), the color distribution of filtered images are much closer to the original images, however, it still gives worse fooling rate with lower prediction confidence. By combining L_{rec} and L_{lab} we are able to get the best results. See our supplementary material for more results.

6 CONCLUSION

The examples that were presented in this paper show how many modern machine learning algorithms are vulnerable in surprising ways. The failure of simple machine learning models, when subjected to the attacks, demonstrate how differently the models act from what their designers intended. Fortunately, there are excellent solutions for the specific attacks described in this paper that make the risk of a particular type of attack negligible or substantially lower the

chance of success. However, the defenses we proposed are not completely adaptive and doesn't perform well on certain dataset; it may leave another vulnerability open to attackers who know the defense being applied.

CONTRIBUTION

Here are the contributions of each group member in the project:

Shray Khanna: Surveyed two papers: Fast Gradient Sign Method (FGSM[8]) and Basic Iterative Method (and Iterative Least Likely Class Method [2]). Implemented the first method (FGSM) on several models to test the results. Made and presented the final presentation. Wrote partial documentation.

Haipeng Li: Surveyed and implemented the Universal Perturbation Attack [14]. Proposed and implemented the defense against Universal Perturbation Attack: Perturbation Filtering Network (PFN). Made slides for the initial presentation and presented in both presentations. Wrote partial documentation.

John Sweeney: Surveyed two papers: the One Pixel Attack[20] and Decision-Based Adversarial Attacks: Reliable Attacks Against Black-Box Machine Learning Models. [3] Partially implemented the One Pixel Attack[20], before it was decided as a group to remove One Pixel Attack Attack due to the defense being redundant. Presented on One Pixel Attack for final presentation. Wrote the abstract, introduction and conclusion for the paper.

REFERENCES

[1] Naveed Akhtar, Jian Liu, and Ajmal Mian. 2018. Defense against universal adversarial perturbations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3389–3398.

[2] Ian J. Goodfellow Alexey Kurakin and Samy Bengio. 2016. Adversarial examples in the physical world. *CoRR* abs/1607.02533 (2016). http://arxiv.org/abs/1607.02533

[3] Wieland Brendel, Jonas Rauber, and Matthias Bethge. 2017. Decision-Based Adversarial Attacks: Reliable Attacks Against Black-Box Machine Learning Models. arXiv:stat.ML/1712.04248

[4] Anirban Chakraborty, Manaar Alam, Vishal Dey, Anupam Chattopadhyay, and Debdeep Mukhopadhyay. 2018. Adversarial Attacks and Defences: A Survey. *CoRR* abs/1810.00069 (2018). arXiv:1810.00069 http://arxiv.org/abs/1810.00069

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 248–255.

[6] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. Deep learning. MIT press.

[7] Ian Goodfellow, Patrick McDaniel, and Nicolas Papernot. 2018. Making Machine Learning Robust Against Adversarial Inputs. In *Communications of the ACM*. 56–66.

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

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[10] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[11] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. 1097–1105.

[13] Yann LeCun, Koray Kavukcuoglu, and Clément Farabet. 2010. Convolutional networks and applications in vision. In *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*. IEEE, 253–256.

[14] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. 2017. Universal adversarial perturbations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1765–1773.

[15] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. 2016. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2574–2582.

[16] Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*. 807–814.

[17] Prasun Roy, Subhankar Ghosh, Saumik Bhattacharya, and Umapada Pal. 2018. Effects of Degradations on Deep Neural Network Architectures. *arXiv preprint arXiv:1807.10108* (2018).

[18] SamanehSorournejad, Zahra Zojaji, Reza Ebrahimi Atani, and Amir Hassan Monadjemi. 2016. A Survey of Credit Card Fraud Detection Techniques: Data and Technique Oriented Perspective. arXiv:1611.06439v1

[19] Rainer Storn and Kenneth Price. 1997. Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. *Journal of Global Optimization* 11, 4 (01 Dec 1997), 341–359. https://doi.org/10.1023/A:1008202821328

[20] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. 2019. One Pixel Attack for Fooling Deep Neural Networks. *IEEE Transactions on Evolutionary Computation* 23, 5 (Oct 2019), 828–841. https://doi.org/10.1109/tevc.2019.2890858

[21] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. 3104–3112.

[22] Xiaoyong Yuan, Pan He, Qile Zhu, Rajendra Rana Bhat, and Xiaolin Li. 2017. Adversarial Examples: Attacks and Defenses for Deep Learning. *CoRR* abs/1712.07107 (2017). arXiv:1712.07107 http://arxiv.org/abs/1712.07107