Real-Time Adversarial attacks (Group – 28)

Jaydeep Singh Chouhan
IIT-Hyderabad
sm21mtech12005@iith.ac.in

Sahin Hossain Choudhury
IIT-Hyderabad
sm21mtech12002@iith.ac.in

Prakriti Sahu
IIT-Hyderabad
sm21mtech12009@iith.ac.in

Shubham Ramesh Dangat
IIT-Hyderabad
sm21mtech14003@iith.ac.in

Ravi Kumar
IIT-Hyderabad
sm21mtech12010@iith.ac.in

Abstract

Adversarial attack is a machine learning strategy that uses deceptive input to fool machine learning models. It comprises both the generation and detection of adversarial examples, which are inputs designed specifically to fool classifiers. Deep Neural networks (DNNs) are susceptible to adversarial examples, which are intentionally crafted inputs that cause DNNs to make incorrect predictions. Recent research has demonstrated that these techniques generalize to the physical domain, causing perturbations on physical objects to trick image classifiers under a range of real-world scenarios. Deep learning models applied in safety-critical cyber-physical systems are vulnerable to such attacks. Adversarial machine learning attacks have been intensively researched in some areas, such as image classification and spam detection. In this project, our team addresses the problem of adversarial attacks on real time state-of-art models. This paper reviews several techniques used for the same. Our team aims to implement one such technique and try to improvise upon the same.

1. Introduction

With machine learning rapidly becoming a big aspect of enterprises' value propositions, the necessity for organizations to secure them is increasing rapidly. As a result, Adversarial Machine Learning is rapidly becoming a significant field in the software development industry. There are numerous adversarial attacks that can be employed against machine learning systems. Many of these are based on deep learning systems as well as conventional machine learning models like Support Vector Machines (SVMs) and linear regression. Most adversarial attacks try to degrade classifier performance on certain tasks, basically to "fool" the machine learning system.

Adversarial examples are inputs to machine learning models that are purposefully created by an attacker to force the model to make a mistake. An adversarial example is a corrupted version of a valid input, where the corruption is performed by adding some perturbation to it. This unnoticed nuisance is intended to trick the classifier by increasing the likelihood of an erroneous class. The adversarial example is intended to appear "normal" to humans, yet it leads to misclassification by the targeted machine learning model.

In [1], The paper describes a method for generating adversarial patches for targets with a high intra-class variation, particularly people. The goal is to create a patch capable of successfully concealing a person from a person detector. Intruders can sneak around unobserved by holding a little cardboard plate in front of their body, directed at the surveillance camera, in an approach that could be used maliciously to defeat monitoring systems.

In [2], The paper discusses the idea of implementing the adversarial attacks on the face recognition systems by creating the eyeglass that helps the face to evade detection.

In [3] paper the DeepFool technique is proposed in the study to efficiently generate perturbations that fool deep networks and hence reliably evaluate the robustness of these classifiers. Extensive experimental findings show that our strategy beats recent methods for computing adversarial perturbations and making classifiers more robust.

In [4], The paper extends physical attacks to more difficult object detection models, a broader class of deep

learning algorithms extensively employed to recognize and classify many items in a scene. The paper improves on a recent physical attack on image classifiers by creating perturbed physical objects that object detection models either ignore or mislabeled. The paper utilizes a Disappearance Attack to make a Stop sign "disappear" according to the detector— either by covering the sign with an adversarial Stop sign poster or by affixing hostile stickers to the sign.

2. Literature Review

We reviewed several approaches for defined problem that we have listed below.

2.1. Approaches

2.1.1 DeepFool Algorithm for adversarial attack on image.

DeepFool for binary classifiers- This algorithm returns about the number of perturbations (noise) that needs to be added to the given image so that it gets misclassified to other class. It works on the basis of calculating the minimum amount of work that is needed to alter the given image so that it gets misclassified.

DeepFool for multiclass classifiers- Above algorithm can be easily extended to multiclass classifiers by simply extending the number of classes and finding the class that is closest to current class and perturbing our image to that class. This will lead in very a smaller number of perturbations required to be added to the image. This algorithm works on greedy approach so it might not give the optimal perturbations but it still yields very small perturbations which are believed to be good approximations of the minimal perturbations.

Both of the above algorithms are implemented by considering the L_2 norm as a distance metric, but the framework of these algorithms is not limited to just L_2 norm and the proposed algorithm can simply be adapted to find minimal adversarial perturbations for any L_p norm.

2.1.2 Physical Adversarial Examples to fool Object Detectors

This work extends physical attacks to more challenging object detection models, a broader class of deep learning algorithms commonly used to detect and classify multiple objects in a scene. Previous research has proposed more efficient algorithms for generating adversarial examples. All these works require that the attacker has "digital-level" access to an input, such as the ability to make arbitrary pixel-level modifications to a classifier's input image. For deep learning applications in Cyber physical systems (for example, in an autonomous vehicle), these attacks implicitly assume that the adversary controls the DNN's input system (e.g., a camera). A more powerful and realistic threat model would presume that the attacker only controls the physical layer, such as the surroundings or objects with which the system interacts.

The RP2 algorithm [6] is extended to give proof-of-concept attacks for object detection networks, a more diverse class of DNNs than image classifiers. A new physical attack on object detection networks using the new and improved algorithm is presented: the Disappearance Attack, which causes physical items to be disregarded by a detector.

Extended RP2 algorithm- There are three significant variations between the original RP2 [6] algorithm and this one:

a. Modified Adversarial Loss Function - The adversarial loss function is modified due to changes in the output behavior of classifiers and object detectors.
b. Synthetic Representation of New Physical Constraints - Additional constraints that the adversarial perturbation must be robust to are recognized, and these constraints are synthetically modelled.
c. Noise Smoothing using Total Variation - Rather than employing the 'p' norm, a smoothness constraint is introduced into the objective.

Methodology-

A Disappearance Attack causes a Stop sign to "disappear" according to the detector, either by covering it with an adversarial Stop sign poster or by adding adversarial stickers to the sign. The state-of-the-art YOLOv2 is used to demonstrate the results of the implementation of the Disappearance Attack. The faster R-CNN, a different object detection model, is also used to demonstrate the transferability of the adversarial perturbations.

A new Creation Attack is also implemented, in which innocuous physical stickers trick a model into identifying nonexistent objects. Comparatively, less time is spent improving the results of the Creation Attack and work on this attack is left for future experimentation.

2.1.3 Adversarial Patch Generation

This method is used to construct adversarial image patches for the real world that are universal (can attack

any scene), robust (can withstand a broad range of modifications), and targeted (can lead a classifier to output any target class).

In the adversarial examples, a variety of optimization algorithms such as L-BFGS, Fast Gradient Sign Method (FGSM), DeepFool, and Projected Gradient Descent (PGD) were used to modify pixels by a minimal amount. Furthermore, Jacobian-based saliency map approaches add a small patch to a fixed location in the image. The paper focuses on attacking and defending against slight or unnoticeable changes in the input. It provides an image-independent patch that is noticeable to a neural network. This form of patch is scene-independent, allowing attackers to generate a physical-world attack without prior knowledge of conditions such as illumination, camera angle, the type of classifier being targeted, or other items in the surroundings.

Methodology-

This method involves masking the patch to allow it to take any shape and then training a number of images. In addition, the image is optimized by employing gradient descent to apply a random translation, scaling, and rotation. A variation of the Expectation over Transformation (EOT) framework is used to train the patch.

2.1.4 Attack on Face-Recognition system

In paper [2] by Sharif et. at. focuses on targeting facial biometric systems in such a way that the attacks are both physically feasible and inconspicuous. They concentrate on targeting the state-of-the-art algorithms in this section. They categorize the attacks into two types- Dodging and Impersonation. Both are aimed towards facial recognition systems (FRS) that do multiclass classification. In this case, they assume a white-box scenario in which the user is aware of the architecture and settings of the system under attack. They talk about extending it to the black box. These cutting-edge algorithms are fooled using a pair of eyeglasses.

Methodology-

In 'Impersonation', advisory seeks to have a face recognized as a specific other face. An attacker who wishes to impersonate a target *t* needs to find how to perturb the input *x* via an addition *r* to maximize the probability of class c_t. In [7], Szegedy et at. defined this as minimizing the distance between $f(x+r)$ and h_t. Similarly, we define the optimization problem to be solved for 'Impersonation' as-

$$Argmin_r\ softmaxloss\ (f\ (x + r),\ c_t)$$

In other words, we seek to find a modification *r* of image *x* that will minimize the distance of the modified image to the target class c_t. The other approach used is 'Dodging' where the attacker seeks to have a faced recognized as any arbitrary face. To accomplish this goal, the attacker needs to find how to perturb the input *x* to minimize the probability of the class c_x. Such a perturbation *r* would maximize the value of $softmaxloss(f(x + r),\ c_x)$. To this end, we define the optimization problem for dodging as-

$$argmin_x\ (-\ SoftMax\ loss(f(x + r),\ c_x)$$

To solve this, they use gradient descent algorithm. To demonstrate the 'Dodging' and 'Impersonation' they use three DNNs: First, they use the DNN developed by Parkhi et al. (39-layer DNN for FRS), which we refer to as DNN_a. Second, they trained two DNNs (termed DNN_b and DNN_c) to recognize celebrities as well as people who were available to us in-person for testing real attacks. DNN_b and DNN_C trained by transfer learning.

They also focus on physical realizability by utilizing facial accessories such as eye glasses. To facilitate physical realizability of the glasses generated, the focus is on enhancing perturbations' robustness, smoothness, and printability. To find such perturbations, they define the non-printability score (NPS) of images to be high for images that contain unreproducible colors, and low otherwise.

They also try to attack black box model (Face++ FRS), for which they use particle swarm optimization method. In the end, they tried attacking a real time face detector Viola-Jones Face Detector. There are limitations in the work like variations in imaging conditions that they investigated in this work, which are narrower than can be encountered in practice. For instance, they controlled lighting by taking images in a room that does not have external windows.

2.1.5 FGSM- Explaining and Harnessing Adversarial Examples

The cause of adversarial examples was a mystery, and speculations suggested it was due to extreme nonlinearity of deep neural networks, combined with insufficient model averaging and insufficient regularization of the purely supervised learning problem. The study shows that these speculations do not hold true and linear behavior in high-dimensional spaces was enough to cause adversarial examples. This helped to design a fast method of generating adversarial examples that makes adversarial training practical. The study shows that adversarial training can provide an additional regularization benefit than by using dropout alone.

2.1.6 One Pixel Attack for fooling DNNs

The research studies an attack in a very restricted scenario where just one pixel can be changed. A novel method is proposed for producing one-pixel adversarial perturbations based on differential evolution (DE) for this purpose. Because of the inherent features of DE, it requires less adversarial information (a blackbox attack) and can mislead more types of networks. Some newly proposed detection methods have demonstrated high accuracy in identifying adversarial perturbation. They do, however, introduce another layer of preprocessing, which increases the system's response time and can be inefficient when dealing with adversarial scenarios that run in real time with a high frame rate. Furthermore, the effect of preprocessing on classification accuracy is still poorly understood. In practice, detecting adversarial perturbation can be useful. However, the basic issue remains unresolved: neural networks are still unable to distinguish similar images, or even minor adversarial perturbation. The goal of proposing novel attack methods is to highlight the existence of various types of vulnerabilities and the corresponding understanding.

2.1.7 Iterative Method

The paper demonstrates that machine learning systems are vulnerable to adversarial examples even in physical world scenario. This is demonstrated by feeding adversarial images captured with a cell phone camera to an ImageNet Inception classifier and analyzing the system's classification accuracy. It is discovered that a considerable proportion of adversarial examples are incorrectly identified even when seen through a camera.

2.2. Findings as per literature surveys

The results from 2.1.1 were collected by testing the DeepFool algorithm on deep convolutional neural networks architectures applied to MNIST, CIFAR-10, and ImageNet image classification datasets. The following results were obtained as shown in Table 1.

The work in 2.1.2 shows that the existing algorithms can be adapted to produce physical attacks on object detectors in highly variable environmental conditions, shown in table 2 and 3.

The paper in 2.1.3 tested the attack and compared it with two whitebox attacks (whitebox-single model, whitebox-ensemble), blackbox attacks and control patch (Real Toaster). The conclusion was that the attack worked on real world scenario and can be disguised as an innocuous sticker. The paper motivated as if generating small or imperceptible perturbations to a natural image is not concerned by potential malicious attackers. Even if humans are able to notice these patches, they may not understand the intent of the patch and instead view it as a form of art.

The paper reviewed in 2.1.4 carried several experiments, and the following results were achieved. The article replicated dodging and impersonation attempts in tests 1 and 2 by allowing the attacker to disturb any pixel on her face. The attacker was successful in all of her attempts. Furthermore, the antagonistic examples discovered in these circumstances are most likely unnoticeable to humans. A pixel that overlay the face had a mean perturbation of 1.7, with a standard deviation of
0.93. The study claims that the types of attacks investigated in trials 1 and 2 are far from feasible. Because the perturbations are too tiny and lack structure, they may be too hard to physically implement, e.g., it is likely impossible to transform a human face in exactly the way required by the attack. Furthermore, the amount by which pixels are perturbed is frequently much smaller than the mistake that happens when printing a perturbation and then capturing it with a scanner or camera. The results demonstrate that the average inaccuracy per pixel in a perturbation when printed and scanned is 22.9, with a standard deviation of 38.22. As a result, even if the attacker properly recognizes the perturbation, she is unlikely to trick the system. To that purpose, the article mimics evading and impersonation attacks in tests 3–8 by perturbing only the eyeglass frames of each subject's eyeglasses. Except for experiment 6, the attacker was able to avoid detection or impersonate targets in all attempts. Impersonators in experiment 6 were successful in around 91.67 percent of their attempts to mislead DNNA utilizing disturbed eyewear. They suggest that due to the vast number of classes recognized by DNNA, other classes occupy the picture space between some attacker-target pairs. As a result, mimicking these targets necessitates navigating many classification boundaries, making impersonation assaults more challenging.

As per the results from 2.1.5, adversarial examples are a property of high-dimensional dot products. They are caused by models that are too linear rather than too nonlinear. The generalization of adversarial examples across models can be explained by adversarial perturbations being highly aligned with a model's weight vectors and different models learning similar functions when trained to accomplish the same objective. The direction of the perturbation is more important than the specific point in space. Adversarial perturbations generalize across different clean examples since it is the direction that matters most. A variety of methods for producing adversarial examples were introduced. It has been demonstrated that adversarial training can result in greater regularization than dropout. Models that are simple to optimize are equally simple to perturb. Linear models are incapable of withstanding adversarial perturbation. Only structures with a hidden layer (following universal approximator theorem) can be trained to withstand adversarial perturbation. RBF networks can withstand adversarial examples. Models trained to model the input distribution are not robust to adversarial examples. Ensembles are vulnerable to adversarial examples.

Classifier	**Test error**	$\hat{\rho}_{adv}$ [DeepFool]	Time	$\hat{\rho}_{adv}$ [8]	time	$\hat{\rho}_{adv}$ [9]	time
LeNet (MNIST)	1%	2.0×10^{-1}	110 ms	1.0	20 ms	2.5×10^{-1}	> 4 s
FC500-150-10 (MNIST)	1.7%	1.1×10^{-1}	50 ms	3.9×10^{-1}	10 ms	1.2×10^{-1}	> 2 s
NIN (CIFAR-10)	11.5%	2.3×10^{-2}	1100 ms	1.2×10^{-1}	180 ms	2.4×10^{-2}	>50 s
LeNet (CIFAR-10)	22.6%	3.0×10^{-2}	220 ms	1.3×10^{-1}	50 ms	3.9×10^{-2}	>7 s
CaffeNet (ILSVRC2012)	42.6%	2.7×10^{-3}	510 ms*	3.5×10^{-2}	50 ms*	-	-
GoogLeNet (ILSVRC2012)	31.3%	1.9×10^{-3}	800 ms*	4.7×10^{-2}	80 ms*	-	-

Table 1: The adversarial robustness of different classifiers on different datasets. The time required to compute one sample for each method is given in the time columns. The times are computed on a Mid-2015 MacBook Pro without CUDA support. The asterisk marks determine the values computed using a GTX 750 Ti GPU.

YOLOv2	**Poster**	**Sticker**
Indoors	202/236 (85.6%)	210/247 (85.0%)
Outdoors	156/215 (72.5%)	146/230 (63.5%)

Table 2: Attack success rate for the disappearance attack on YOLO v2. The paper tested a poster perturbation, where a true-sized print is overlaid on a real Stop sign, and a sticker attack, where the perturbation is two rectangles stuck to the surface of the sign. The table cells show the ratio: number of frames in which a Stop sign was not detected / total number of frames, and a success rate, which is the result of this ratio.

Faster R-CNN	**Poster**	**Sticker**
Indoors	189/220 (85.9%)	146/248 (58.9%)
Outdoors	84/209 (40.2%)	47/249 (18.9%)

Table 3: Attack success rate for the disappearance attack on Faster R-CNN. We tested a poster perturbation, where the entire Stop sign is replaced with a true-sized print, and a sticker attack, where the perturbation is two rectangles stuck to the surface of the sign. The table cells show the ratio: number of frames in which a Stop sign was not detected / total number of frames, and a success rate, which is the result of this ratio.

Experiment #	**Area perturbed**	**Goal**	**Model**	**# Attackers**	**Success rate**
1	Entire face	Dodging	DNN_A	20	100.00%
2	Entire face	Impersonation	DNN_A	20	100.00%
3	Eyeglass frames	Dodging	DNN_A	20	100.00%
4	Eyeglass frames	Dodging	DNN_B	10	100.00%
5	Eyeglass frames	Dodging	DNN_C	20	100.00%
6	Eyeglass frames	Impersonation	DNN_A	20	91.67%
7	Eyeglass frames	Impersonation	DNN_B	10	100.00%
8	Eyeglass frames	Impersonation	DNN_C	20	100.00%

Table 4: A summary of the digital-environment experiments attacking DNN_A, DNN_B , and DNN_C under the white-box scenario. In each attack we used three images of the subject that we sought to misclassify; the reported success rate is the mean success rate across those images.

3. Preliminary Results

We experimented with three different types of adversarial attacks as follows-

3.1 One Pixel attack

We performed attack on image in Fig. 1 which was predicted as frog before attack with probability 0.352779 and after attack it was misclassified as bird with probability 0.90834 as shown in Fig. 2.

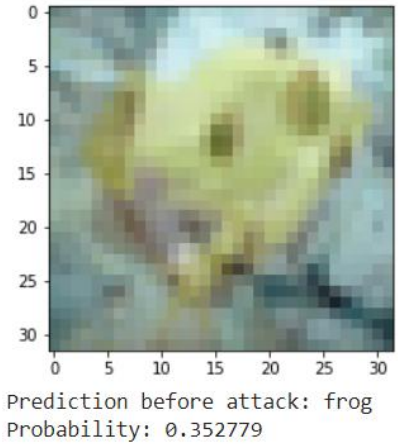


Fig. 1: One pixel attack (Original Image)

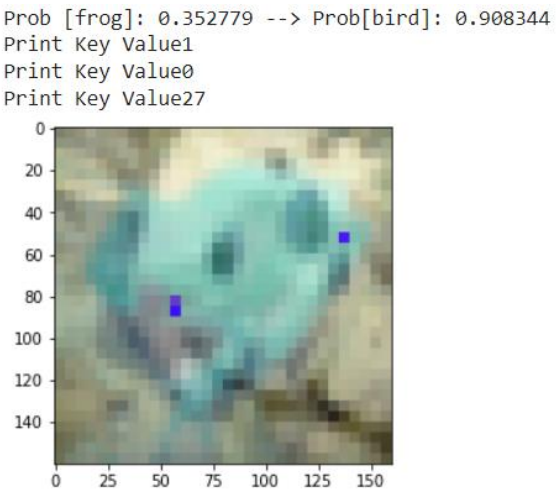


Fig. 2: One pixel attack (After attack)

3.2 Fast Gradient Sign Method

We performed attack on image in Fig. 3 which was predicted as airplane before attack and after attack it was misclassified as frog as shown in Fig. 4.

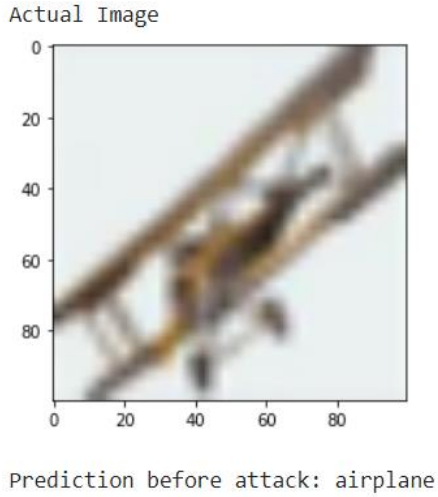


Fig 3: Fast gradient Sign Method attack (Original Image)

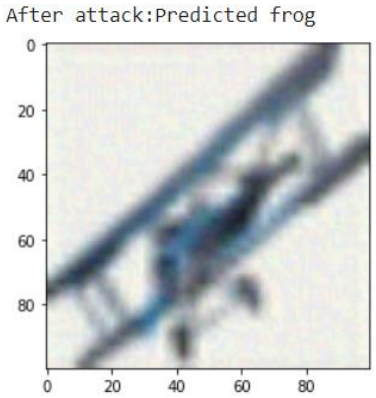


Fig. 4: Fast gradient Sign Method attack (After attack)

3.3 Iterative Method

We performed attack on image in Fig. 5 which was predicted as airplane before attack and after attack it was misclassified as automobile as shown in Fig. 6.

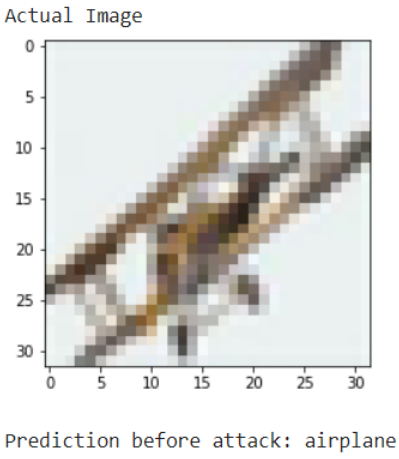


Fig 5: Iterative attack (Original Image)

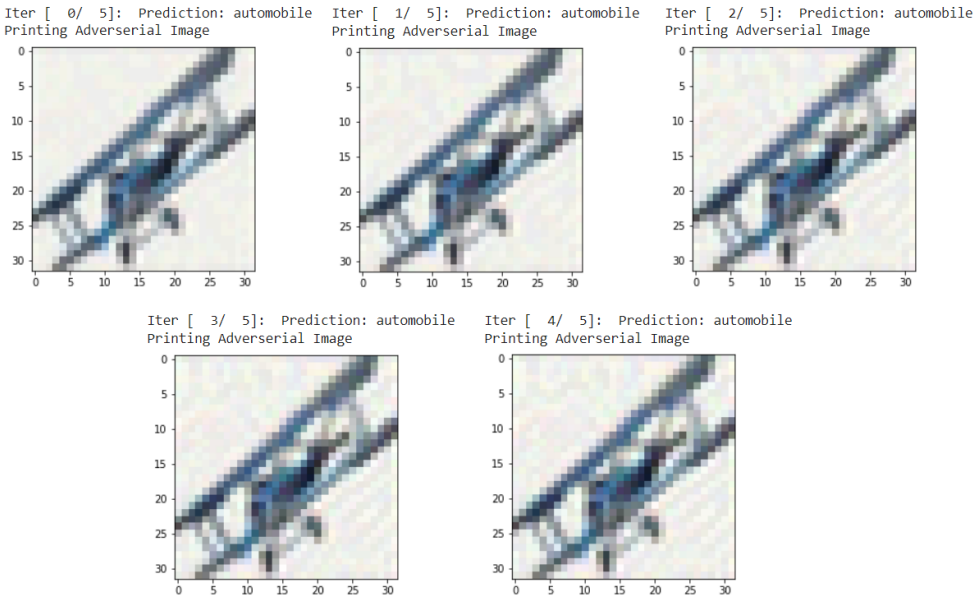
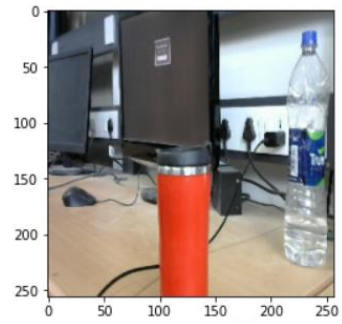


Fig 6: Iterative attack (After attack)

4. Final Results

We implemented the above attacks on sample of images collected by us. Following are the results obtained for various models. Also, we implemented the adversarial patch attack by generating patch of school bus and goldfish, results of which are included below.

4.1 One Pixel Attack



Prediction before attack: carton

Fig 7: One Pixel (before attack)

Prob [carton]: 0.064119
Prob [carton]: 0.126223 --> Prob[shower curtain]: 0.096113
Print Key Value27

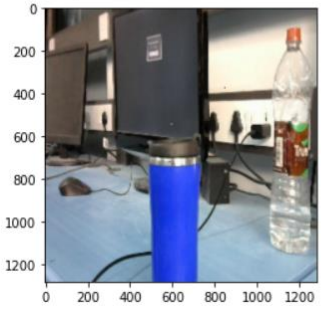
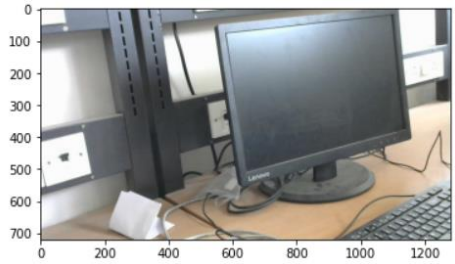


Fig 8: One Pixel (after attack)

4.2 FGSM attack



Prediction before attack: monitor

Fig 9: FGSM attack (before attack)

After attack:Predicted scale, weighing machine

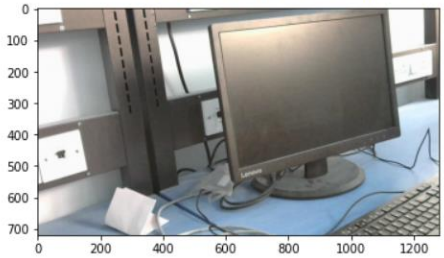


Fig 10: FGSM attack (after attack)

4.3 Iterative attack

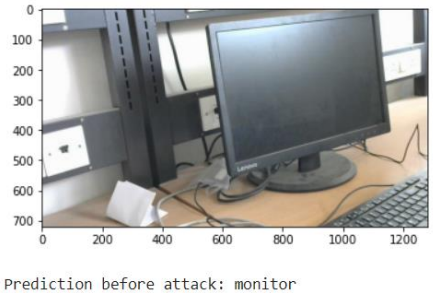


Fig 11: Iterative attack (before attack)

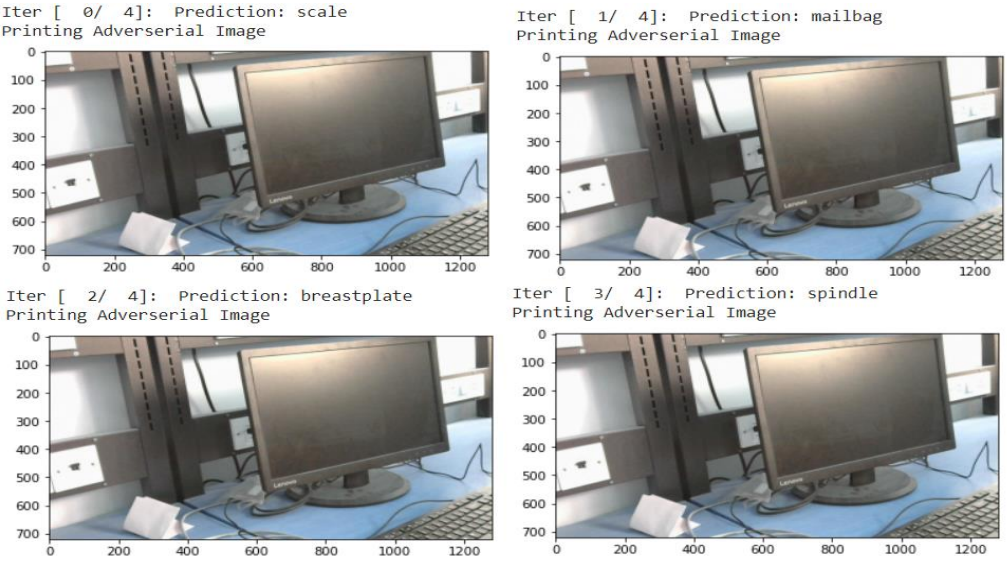


Fig 12: Iterative attack (after attack)

4.4 Adversarial Patch attack

Following are the patches for performing attacks. These patches are added over the image to fool the model into predicting incorrect class.

Fig 13: Goldfish and school bus patch

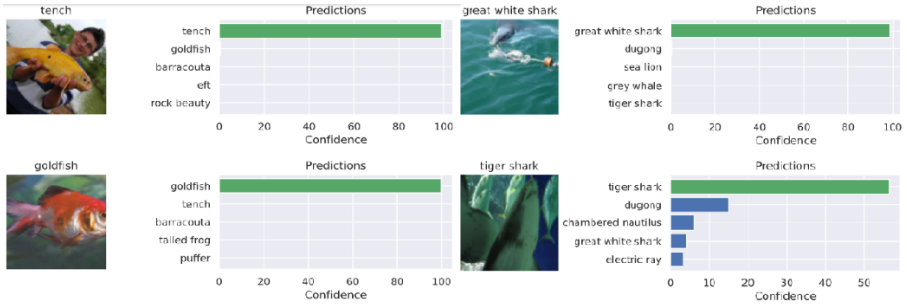


Fig 14: Adversarial patch attack (before attack)

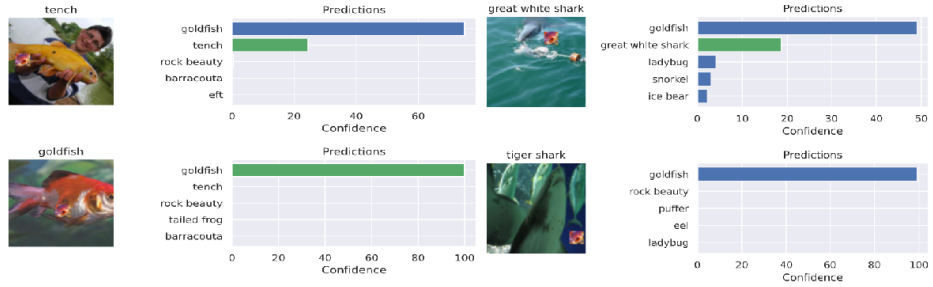


Fig 15: Patch attack using Goldfish patch.

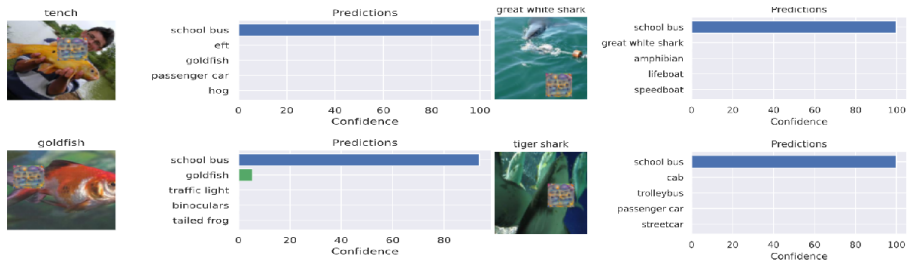


Fig 16: Patch attack using School bus patch

In addition to this, we also implemented defense against FGSM attack which is shown below.

```
Epsilon: 0      Test Accuracy = 9708 / 10000 = 0.9708
Epsilon: 0.007  Test Accuracy = 9701 / 10000 = 0.9701
Epsilon: 0.01   Test Accuracy = 9650 / 10000 = 0.965
Epsilon: 0.02   Test Accuracy = 9602 / 10000 = 0.9602
Epsilon: 0.03   Test Accuracy = 9552 / 10000 = 0.9552
Epsilon: 0.05   Test Accuracy = 9380 / 10000 = 0.938
Epsilon: 0.1    Test Accuracy = 8520 / 10000 = 0.852
Epsilon: 0.2    Test Accuracy = 5006 / 10000 = 0.5006
Epsilon: 0.3    Test Accuracy = 2484 / 10000 = 0.2484
```

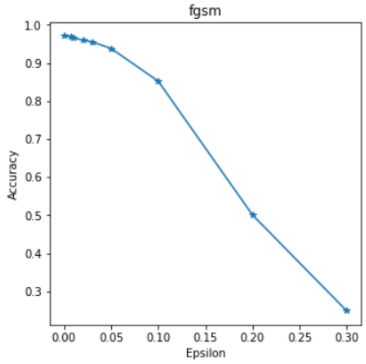


Fig 17: Accuracy dropping from 97% to 24% without defense.

```
Epsilon: 0      Test Accuracy = 9033 / 10000 = 0.9033
Epsilon: 0.007  Test Accuracy = 9050 / 10000 = 0.905
Epsilon: 0.01   Test Accuracy = 9048 / 10000 = 0.9048
Epsilon: 0.02   Test Accuracy = 9050 / 10000 = 0.905
Epsilon: 0.03   Test Accuracy = 9027 / 10000 = 0.9027
Epsilon: 0.05   Test Accuracy = 8990 / 10000 = 0.899
Epsilon: 0.1    Test Accuracy = 8952 / 10000 = 0.8952
Epsilon: 0.2    Test Accuracy = 8832 / 10000 = 0.8832
Epsilon: 0.3    Test Accuracy = 8801 / 10000 = 0.8801
```

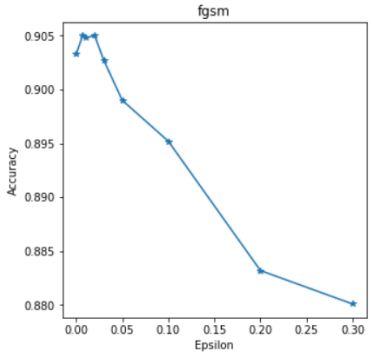


Fig 18: Accuracy dropping from 90% to 88% with defense (using defensive distillation).

Implementing Transferability property: We implemented above attacks on various pretrained models (resnet18, alexnet, vgg16, squeezenet, densenet, inception, googlenet, shufflenet, mobilenet_v2, mobilenet_v3_large,

mobilenet_v3_small, resnext50_32x4d, wide_resnet50_2, mnasnet, efficientnet_b0, efficientnet_b1, efficientnet_b2, efficientnet_b3, efficientnet_b4, efficientnet_b5, efficientnet_b6, efficientnet_b7, regnet_y_400mf, regnet_y_800mf, regnet_y_1_6gf, regnet_y_3_2gf, regnet_y_8gf, regnet_y_16gf, regnet_y_32gf). These models were also fooled by perturbed images, resulting in misclassification of the images

Refer below given code link for code and results.

Code Link: https://github.com/shubham-dangat/Deep_Learning_Project/tree/main

5. Conclusion

From all the papers reviewed, we concluded that there are several ways to generate/implement adversarial attacks on real-time detection/recognition models. These attacks vary from pixel manipulation to physical patches that can be generated to fool these models. The approaches reviewed vary from software to physical objects. Our team implemented three types of such attacks (One pixel attack, FGSM and Iterative method) and reviewed the literature for the same. We implemented these attacks and also tested our attacks on various pretrained models.

References

[1] Simen Thys, Wiebe Van Ranst, Toon Goedemé. Fooling automated surveillance cameras: adversarial patches to attack person detection, 18 Apr 2019.

[2] M. Sharif, S. Bhagavatula, L. Bauer, and M. K. Reiter. Accessorize to a crime: Real and stealthy attacks on state-ofthe-art face recognition. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pages 1528–1540. ACM, 2016.

[3] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 2574–2582, 2016.

[4] D. Song, K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, F. Tramer, A. Prakash, and T. Kohno. Physical adversarial examples for object detectors. In 12th {USENIX} Workshop on Offensive Technologies ({WOOT} 18), 2018.

[5] adversarial examples for object detectors. In 12th {USENIX} Workshop on Offensive Technologies ({WOOT} 18), 2018

[6] T. B. Brown, D. Mane, A. Roy, M. Abadi, and J. Gilmer.

[7] ´Adversarial patch. arXiv preprint arXiv:1712.09665, 2017.

[8] https://arxiv.org/abs/1707.08945

[9] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, Goodfellow, and R. Fergus. Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199, 2013.

[10] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. In International Conference on Learning Representations, 2015.

[11] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, J. Goodfellow, and R. Fergus. Intriguing properties of neural networks. In International Conference on Learning Representations (ICLR), 2014.

[12] Su, J., Vasconcellos Vargas, D., and Kouichi, S., “One pixel attack for fooling deep neural networks”, arXiv e-prints, 2017.

[13] Goodfellow, I. J., Shlens, J., and Szegedy, C., “Explaining and Harnessing Adversarial Examples”, arXiv e-prints, 2014.

[14] Kurakin, A., Goodfellow, I., and Bengio, S., “Adversarial examples in the physical world”,arXiv e-prints, 2016.