ECE 520.650: Machine Intelligence Project 2

Project report written by Jay Paranjape & Arushi Sinha
May 4, 2023

1. Part A- Classification

1.1. Problem Statement

Build a Deep Convolutional Neural Network, train it on CIFAR-10 training set and test it on CIFAR-10 testing set. You can use any architectures learned in class or come up with your own architecture.

1.2. Data Set

This dataset consists of 60,000 RGB images of size 32x32. The images belong to objects of 10 classes such as frogs, horses, ships, trucks etc. The dataset is divided into 50,000 training images and 10,000 testing images. We are going to build a CNN model that can classify images of various objects. We have 10 classes of images: Airplane, Automobile, Bird, Cat, Deer, Dog, Frog, Horse, Ship, Truck. We used PyTorch library to get the pre-built CIFAR-10 dataset. We normalized this dataset to increase the computation speed. The images of each class can be visualized in fig 1

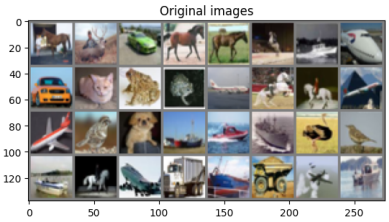


Figure 1: CIFAR-10 images

1.3. Deep Convolutional Neural Network Used: ResNet18

The CNN model used for image classification is ResNet18 RESNET18 has 18 layers with a 7x7 kernel as 1st layer. It has four layers of ConvNets that are identical. Each layer consists of two residual blocks. Each block consists of two weight layers with a skip connection connected to the output of the second weight layer with a ReLU. If the result is equal to the input of the ConvNet layer, then the identity connection is used. But, if the input is not similar to the output, then a convolutional pooling is done on the skip connection [7]. We used the pre-trained model of ResNet on PyTorch and fine-tuned it on the CIFAR-10 dataset by changing the final fully connected layer to have 10 classes instaed of 100. The loss function used is Cross Entropy loss and a Stochastic gradient descent optimizer is used.

- Accuracy found to be 96% on training dataset.
- Accuracy found to be 80% on test dataset.

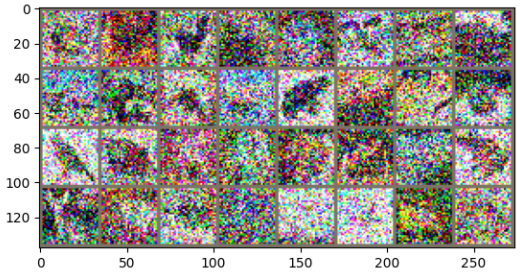
2. Part B- Adversarial Attack & Defense

2.1. Problem Statement

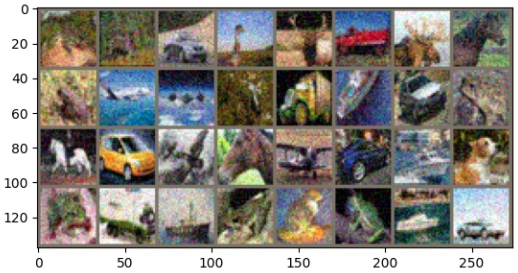
In this part, you will implement several attack models (noise, FGSM, PGD and C-W) on the MNIST data and implement one defense mechanism of your choice. Compare the drop in classification performance for three values of noise magnitude ϵ.

2.2. Attacks

The 4 perturbations used to attack our network are:



(a) Noise with stddev = 0.5



(b) Noise with stddev = 0.09

Figure 2: The attacks with less value of standard deviation appear similar to the real images but are less severe attacks in terms of accuracy drop

Stddev.	Accuracy	Drop
0.01	80%	0%
0.09	72%	8%
0.25	48%	32%
0.50	31%	49%

Table 1: Drop in performance of model on noisy test set with varying standard deviations of gaussian noise.

- **Additive Gaussian Noise Attack(AGNA):** It is a non-targeted black-box evasion attack. Here, the image is perturbed by adding gaussian noise until the model misclassifies the image [3]
- **FGSM (Fast Gradient Sign Attack):** It is a white-box attack It is designed to attack neural networks by leveraging the way they learn, gradients. This attack adjusts the input data to maximize the loss based on the same backpropagated gradients. In other words, the attack uses the gradient of the loss w.r.t the input data, then adjusts the input data to maximize the loss [2].
- **PGD (projected gradient descent):** It is the multi-step variant of FGSM, which is essentially a projected gradient descent on the negative loss function [4]
- **C-W (Carlini-Wagner):** CW finds the adversarial instance by finding the smallest noise $\delta \in R^{nxn}$ added to an image x that will change the classification to a class t [1].

2.3. Methodology

1. For the additive gaussian noise attack, we implement a pytorch transform where we add a gaussian random noise to the image. The mean of the noise variable is set to 0 and the standard deviation is chosen among 4 different values. With increasing standard deviation, the accuracy of the model on the adversarial test set decreased as noted in Table 1. However, the images became more and more noisy as seen in Figure 2.

2. For the rest of the attacks, we utilize the Foolbox Library[5, 6]. This shows the tradeoff between the attack effectiveness and visual quality of the attacked image.

3. We employ the given attacks with different values of epsilon and note the results. CW attack takes more time and compute over other methods and hence, we only attack 200 randomly selected test samples for this attack. For the rest of the attacks, we use the entire test set.

2.4. Drop in Classification Performance

We note the drop in performance of the originally trained ResNet 18 model when tested with the adversarial dataset, for each of the attacks. These results are tabulated in Table 2. For increasing values of epsilon, there is an increased drop in accuracy. However, the adversarial images do not appear visually lose to the original images as epsilon increases.

Epsilon	FGSM	PGD	C-W
0.002	54.36%	54.75%	58.5%
0.02	27.55%	24.31%	57.5%
0.2	6.67%	0.6%	53.5%

Table 2: The accuracy of the original model on the attacked test set. On the original test set, the model gave an accuracy of 80%

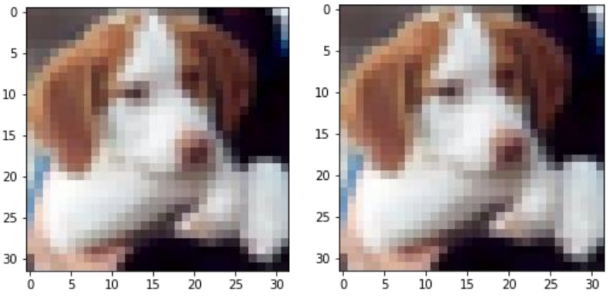
2.5. Defence

Adversarial defense is a technique with which we make our models robust to adversarial attacks. We show an effective technique against the Additive Gaussian Noise Attack with 0 mean and 0.09 standard deviation. We add gaussian noise with the same mean and variance to our training set and retrain the model on it. Thus, the model sees such noisy input during its training and learns its weights so as to minimize loss on these samples as well. This acts like a regularizer and improves the generalizability of the model on the adversarial test set while retaining performance on the original test set.

- On the original test set, the accuracy remains 80%.
- On the adversarial test set, the accuracy increases from 72% to 77%.

2.6. Some Visualizations:

1. Dog getting misclassified as cat after FGSM attack can be seen in figure 3



(a) (L) Original, (R) FGSM Attack

```
array([[[0.00100169, 0.00097981, 0.00098419],
        [0.00100169, 0.00090665, 0.00091067],
        [0.00100169, 0.00090665, 0.00091067],
        ...,
        [0.00092688, 0.00097981, 0.00098419],
        [0.00100172, 0.00097978, 0.00091064],
        [0.00100166, 0.00097984, 0.00098419]]

       [[0.00092688, 0.00097981, 0.00098419],
        [0.00100169, 0.00097981, 0.0009107 ],
        [0.00100169, 0.00097981, 0.00098416],
```

(b) Difference(Absolute Value)

Figure 3: Example of a FGSM attack

2. Black Box Additive Gaussian Noise Attacks on Some Test Set Images can be seen in figure 4

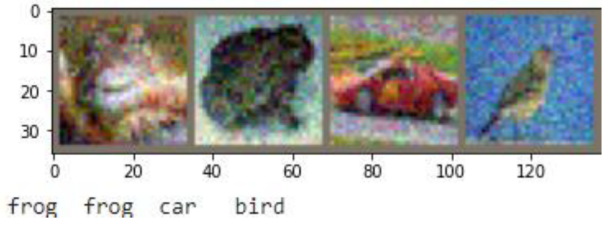


Figure 4: Gaussian Noise attacked images

3. Conclusion

- In this project, we first show the performance of ResNet18 on the CIFAR 10 dataset, followed by the effect of various adversarial attacks. We also employ adversarial training as a defense technique and show improvement in performance.

- We show the trade-off between the visual quality of the adversarial image and its negative effect on the model's performance. The higher the allowed epsilon, the higher the drop in accuracy and the lower the visual quality.
- Finally, we present some visualizations of how the attacked images look

References

[1] Learn the carlini and wagner's adversarial attack - mnist.

[2] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples, 2015.

[3] B. Li, C. Chen, W. Wang, and L. Carin. Certified adversarial robustness with additive noise, 2019.

[4] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.

[5] J. Rauber, W. Brendel, and M. Bethge. Foolbox: A python toolbox to benchmark the robustness of machine learning models. In *Reliable Machine Learning in the Wild Workshop, 34th International Conference on Machine Learning*, 2017.

[6] J. Rauber, R. Zimmermann, M. Bethge, and W. Brendel. Foolbox native: Fast adversarial attacks to benchmark the robustness of machine learning models in pytorch, tensorflow, and jax. *Journal of Open Source Software*, 5(53):2607, 2020.

[7] A. V. Sai Abhishek. Resnet18 model with sequential layer for computing accuracy on image classification dataset. 10:2320–2882, 07 2022.