Project Report

Dependable AI

Neehal Bajaj
B20AI026

10 May 2023

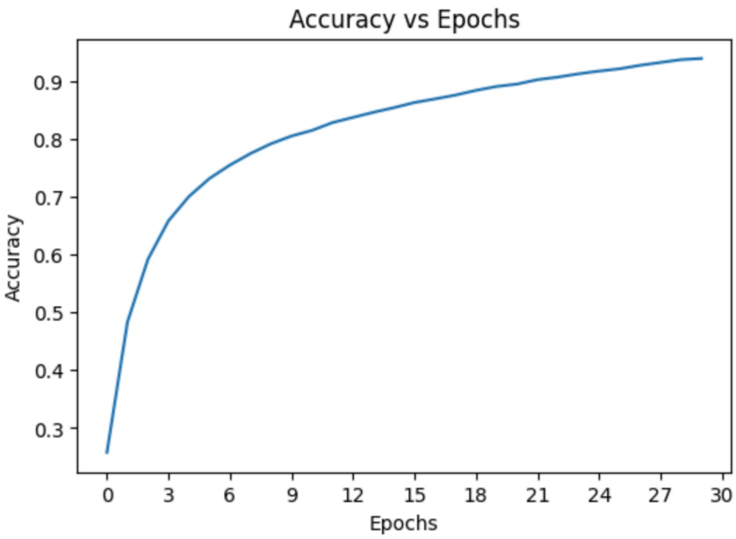
Randomised Masking Attack

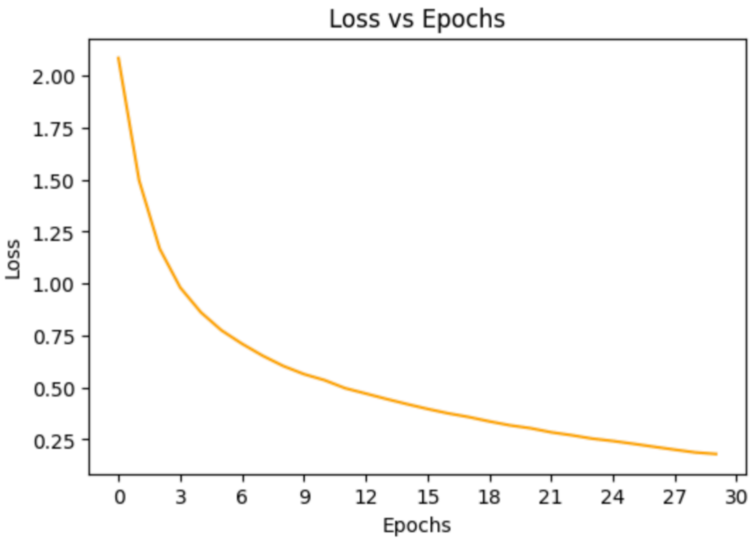
Overview

I've created a basic technique called Random Masking on a CIFAR-10 dataset, which includes randomly choosing pixels and setting them to zero or some other random value. The attacker iteratively perturbs the input data in small steps with the goal of finding the smallest possible perturbation that causes the model to misclassify the input; the idea is to disrupt the underlying patterns in the input data and force the model to learn more robust and resilient features than in PGD. To maximise the loss function while remaining within a fixed distance threshold, PGD iteratively distorts the input image. In contrast to random masking, which only requires picking and applying a random matrix from a given set to each channel of an image, PGD attacks can take a long time depending on a variety of factors such as the number of iterations, the size of the step size, and the distance threshold.

Approach

A) Training the CIFAR-10 dataset on the VGG-16 model with Cross Entropy Loss, SGD, and plotting loss versus epoch and accuracy versus epoch



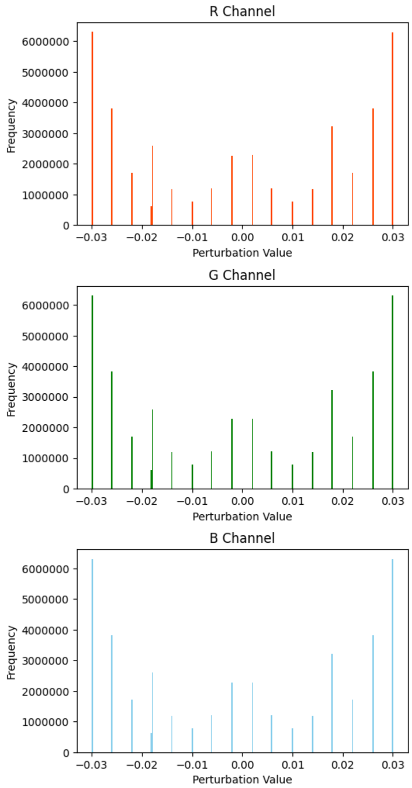


B)Creating a class that will perturb the dataset and return the perturbed images at every step of the attack

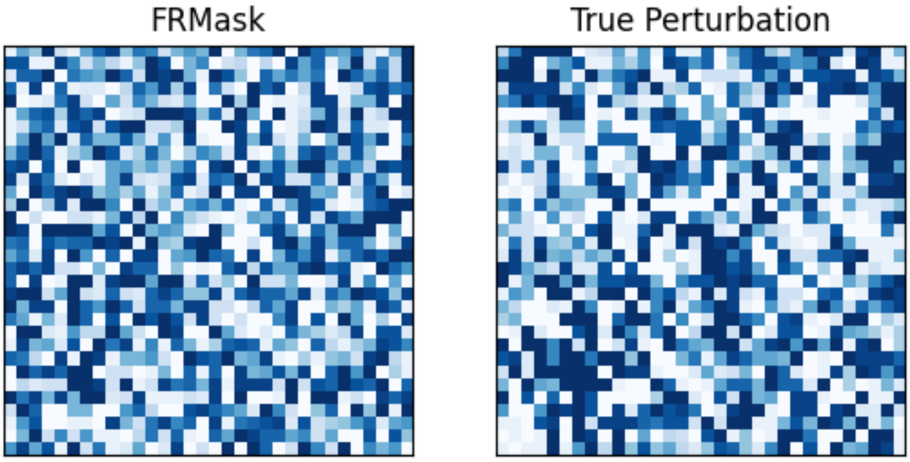
C)The function iterates over each batch of data in the loader and applies the specified attack to each batch of images. The final inventory of perturbed samples is then returned.

D)The samples are then initiated as final perturbation maps by employing job-lib's lightweight pipelining.

E) Using this lightweight pipelining, RGB Channels were acquired and subsequently plotted with the help of subplots were used to illustrate

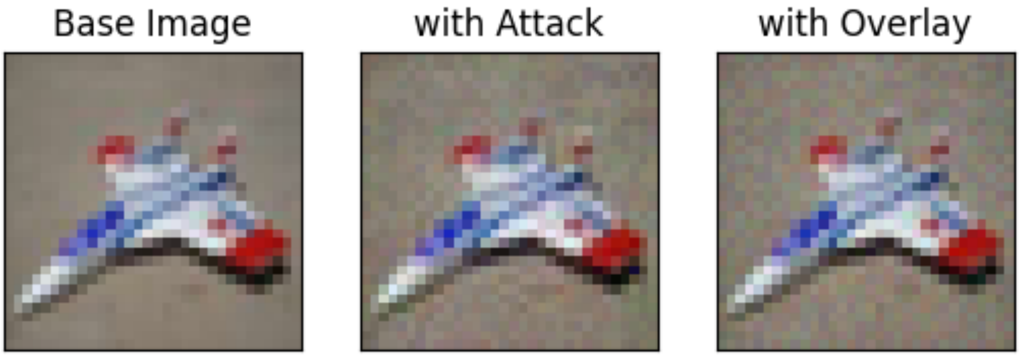


F) Using subplots, noise for true perturbation and Fixed Randomised Mask were plotted.



G) A second function is then defined that accepts an image and a set of weights, where each weight represents the probability of selecting a particular matrix for a specific channel. The modified image is then returned after the weights are applied to the image's weights.

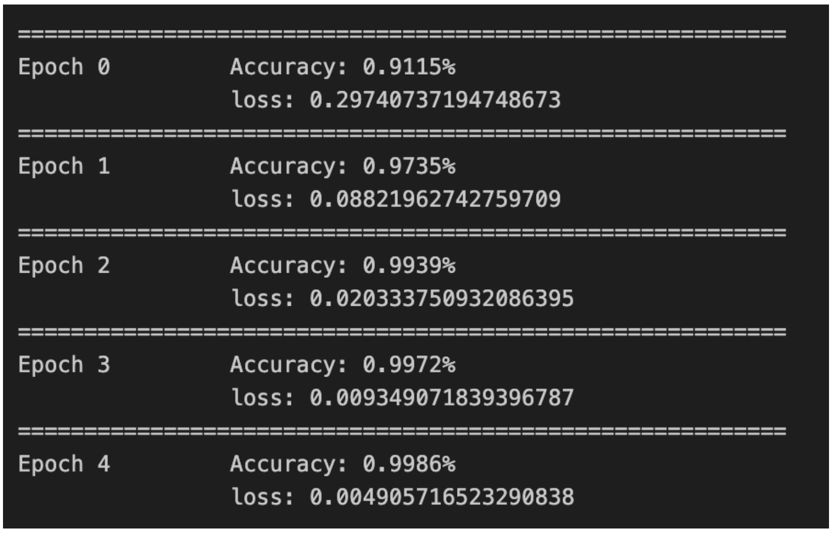
H) We then plot base images, adversal images obtained from the preceding pipelining, and finally images with the assistance of the modified images obtained from the preceding function.



I)We obtain the adversial mask by subtracting the adversial images from the base image.

J) We define an adversarial training function, which performs standard training on the original batch of images and labels for each batch. In addition, the function computes the model's accuracy on the original sample of images and labels, as well as on the adversarially modified images.

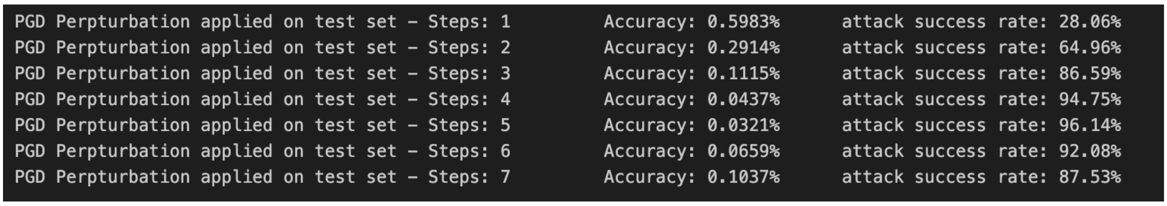
K) We then select a new CIFAR-10 train and test set and conduct adversarial training, obtaining the accuracies for 5 epochs.



```
===========================================================
Epoch 0         Accuracy: 0.9115%
                loss: 0.29740737194748673
===========================================================
Epoch 1         Accuracy: 0.9735%
                loss: 0.08821962742759709
===========================================================
Epoch 2         Accuracy: 0.9939%
                loss: 0.020333750932086395
===========================================================
Epoch 3         Accuracy: 0.9972%
                loss: 0.009349071839396787
===========================================================
Epoch 4         Accuracy: 0.9986%
                loss: 0.004905716523290838
```

L) We define an evaluate function that applies a pre-trained model to a specified new data injector and returns the model's accuracy on the perturbed images. Then, we define a PGD attack class and a pre-trained model. It generates adversarial examples by executing a PGD attack with the number of specified stages.

M) Finally, we attack the model to determine how it degrades the model's accuracy.



```
PGD Perpturbation applied on test set - Steps: 1        Accuracy: 0.5983%      attack success rate: 28.06%
PGD Perpturbation applied on test set - Steps: 2        Accuracy: 0.2914%      attack success rate: 64.96%
PGD Perpturbation applied on test set - Steps: 3        Accuracy: 0.1115%      attack success rate: 86.59%
PGD Perpturbation applied on test set - Steps: 4        Accuracy: 0.0437%      attack success rate: 94.75%
PGD Perpturbation applied on test set - Steps: 5        Accuracy: 0.0321%      attack success rate: 96.14%
PGD Perpturbation applied on test set - Steps: 6        Accuracy: 0.0659%      attack success rate: 92.08%
PGD Perpturbation applied on test set - Steps: 7        Accuracy: 0.1037%      attack success rate: 87.53%
```

References & Links

A) Reference:- https://openreview.net/forum?id=SkgkJn05YX

B) Github Repository:- https://github.com/nbj18/Randomised-Attack