Defense Project Report

Team Emoji

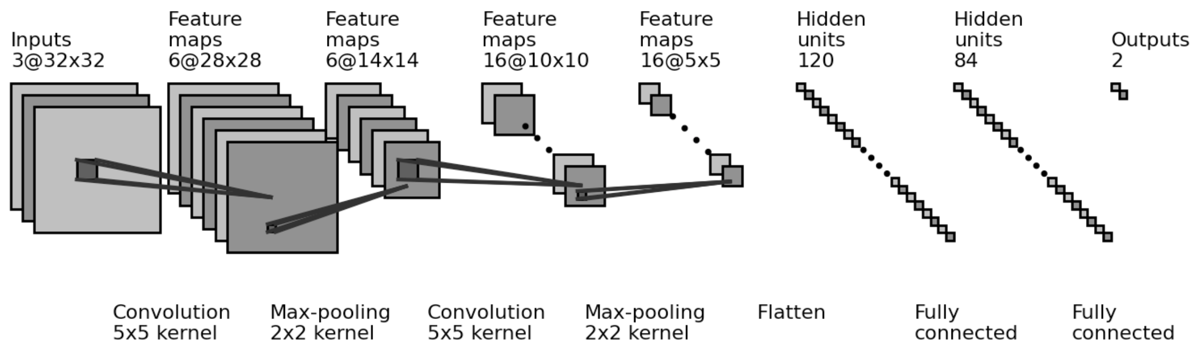
Name: Yifeng Xiong UCI Net ID: yifengx4
Name: Monica Zhou UCI Net ID: quanz13

Our Approach

Our defense project consists of two parts: detector and adversarial training. For the detector, we train a LeNet(6 layers) to distinguish between clean image and perturbed image, while for adversarial training, we add the perturbed image by FGSM(Fast Gradient Sign Method) into training. For levels of robustness, our method can defend against known attacks on unknown images, but it cannot work on unknown attacks.

Detector

Our detector is a 6 layer LeNet. The input is an image, the shape is [1, 3, 32, 32], and the output is an array, the shape is [1, 2]. Our detector would mark one image as perturbed if the second component is larger, and vice versa. We drew a simple picture for our neural network as below:



For the training data, we generate a large dataset with clean image and perturbed image. Then we shuffle the data to prevent overfitting. The following is the pseudocode for generating the whole dataset:

```
Input: trainloader(clean dataset)
Output: train_data(train dataset for the detector)
for inputs, labels in trainloader do
     add [inputs,[0]] into train_data #clean image
     add [perturb(inputs, labels),[1]] into train_data #perturbed
image
shuffle train_data
return train_data
```

Since we input one image a time, after one epoch, our detector gets 100% correct on test data. So we believe we do not need one more epoch. Since the perturb function is based on the main model, we first generate a clean defense project model with epoch 15, and use that model to train our detector. The reason we choose epoch 15 is that we get 49.7 for raw accuracy, which we believe is not overfit or underfit.

Adversarial Training

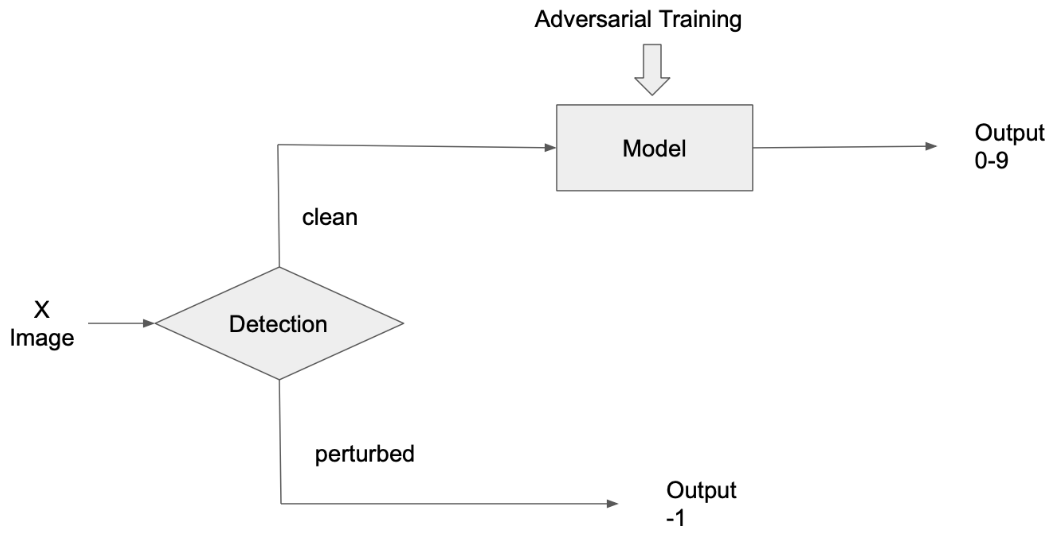
After we successfully get a strong detector, we start working on the adversarial training model. If the detector fails to detect the perturbed image, the model may still have the ability to defend. We add the perturbed data into training, and return the loss with the clean one.

The following is pseudocode:

```
Input: trainloader(training data)
for inputs, labels in trainloader do
     get outputs of inputs by model
     get CrossEntropyLoss for outputs
     perturb inputs
     get outputs of perturb inputs by model
     get CrossEntropyLoss for perturb outputs
     add loss together
     backward loss and optimize the model
```

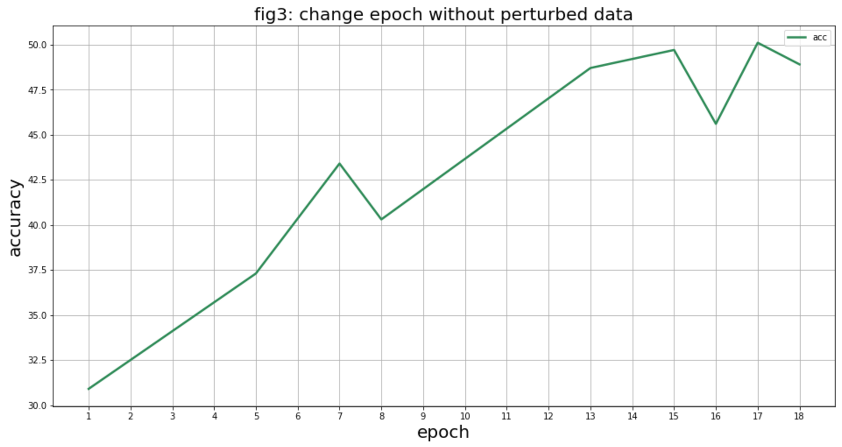
Summary of our model

We will input the image into the detector first, it will label -1 if it is perturbed. For those images our detector thinks are clean, we will then input it into the model after adversarial training. It will return a label which it believes the image belongs to. We provide a clear image to show our model.

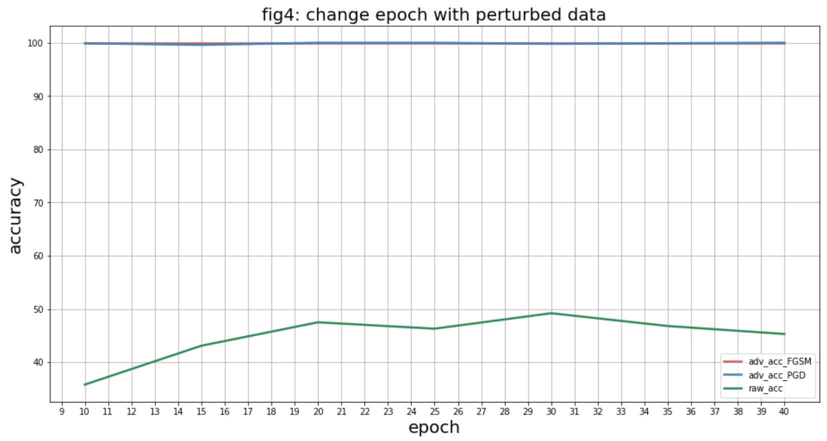


Analysis

Firstly, we try to train a general model without adversarial training, since we need this model during the training for our detectors. The only number we can change is the epoch. We have tried several cases, and finally we decided to pick 15.



After training the detectors, we go back to the original model and do adversarial training. This time, we add the perturbed data into the training. We also changed the epoch to get the best result we have locally.



Concluding Thoughts

Our model works pretty well locally, and for levels of robustness, our method can defend against known attacks on unknown images, but it cannot work on unknown attacks. The main reason is that we only add images perturbed by FGSM into training data for detectors. So our detectors can defend against attacks like FGSM, or PGD. The main challenge for this project is to implement a neural network by ourselves, since we have no experience before. We read many tutorials and watch some videos to get the background of neural networks. With the help of Professor, we successfully built a strong neural network. Our team works much better than last time, since we had a meeting at the start of this project, we discussed the project together, and we got some ideas from the discussion.