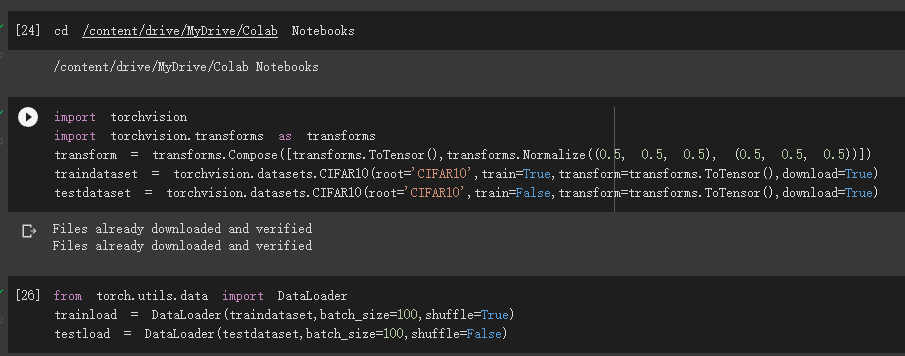
EN.520.650 class project 2 report

Qihua Gong

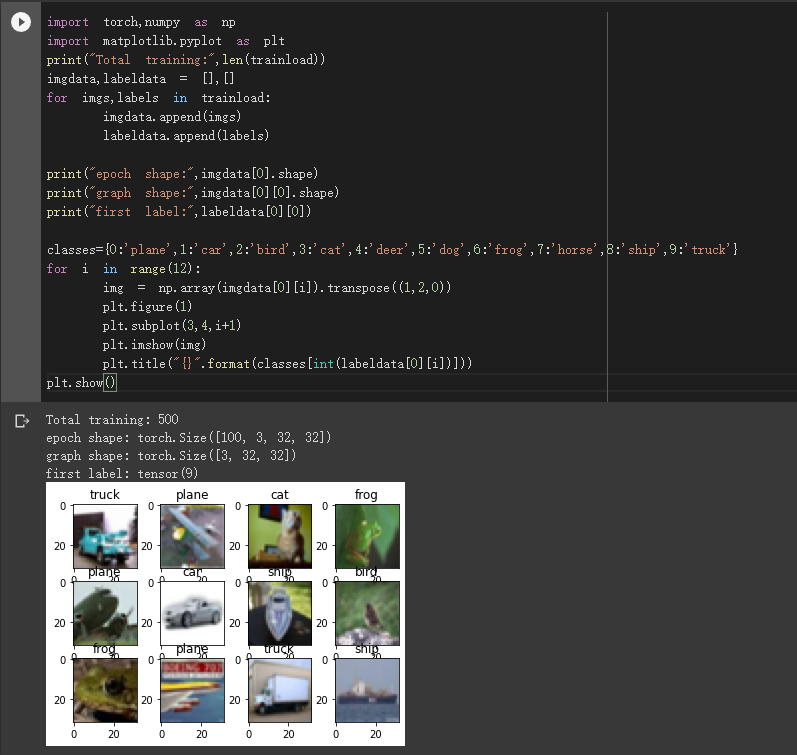
Part1:

The part 1 of this project is about the convolutional neural network training on CIFAR-10 data set. I will use the architecture Lenet5 and Pytorch library in python to help build the training model. CIFAR-10 is a small image classification data set consisting of 60,000 32x32 color images, each of which corresponds to a category, but these 60,000 images have a total of 10 categories. After importing the data set for the above framework, I will try to do a classification and recognition of the image data set. After completing the learning, the test epoch will be compared and loss data which will visually express the training results.

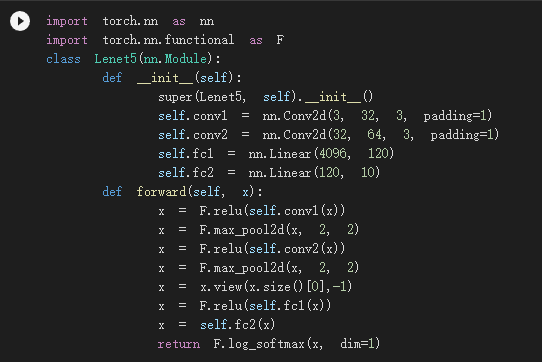
The first is to download and process the data set into a dictionary for our convenience. Originally, we need to use the official given code and use the pickle module of python to handle it. Here we can directly use the built-in tool of pytorch to download and get training set and test set. Then we use the DataLoader function to implement dynamic data extraction and loading.



Then I wrote a small part to verify the work of the imported data and to title the classification. I print the training size, first epoch size, first picture in this epoch to check. Also, I draw the table to check the picture classification well.



Next, we need to create a convolutional neural network model for training. Here we choose the Lenet5 architecture. The design of lenet5 is composed of two parts- convolutional layer and linear. The convolutional layer is used to extract the features of the image, and then feed the features to the linear layer for learning and training. I designed the network structure as a process of two convolutions and then pooling. The first convolution uses 32 convolution kernels to obtain 32 feature maps, and then performs down-sampling pooling to obtain 32 16x16 feature maps. The second convolution uses 64 convolution kernels to obtain 64 feature maps, and then pooling processing to obtain 64 8x8 feature maps. For the layer, the number of input neurons is the 64x8x8=4096 feature points after the above 64 8x8 feature maps are flattened, the number of hidden layers is set to 1024, and the output layer is 10 neurons, corresponding to the probability of 10 categories. The number of convolution kernels and the number of hidden layers need to be tested. I have referred to some other network models. I use these to set the class Lenet5 and forward passing the layer as relu, pool, relu, pool and then linear.

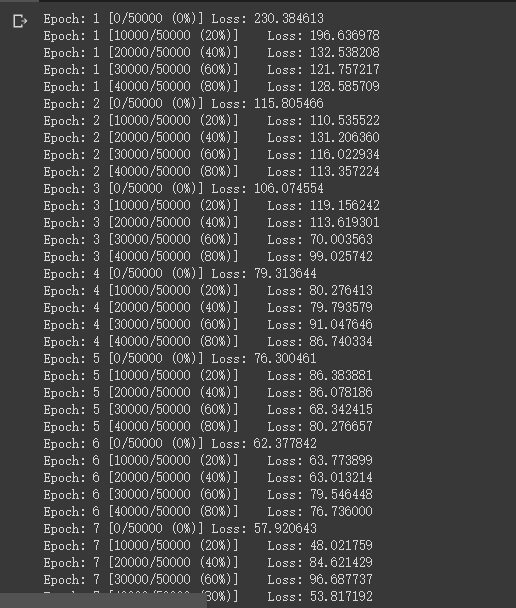


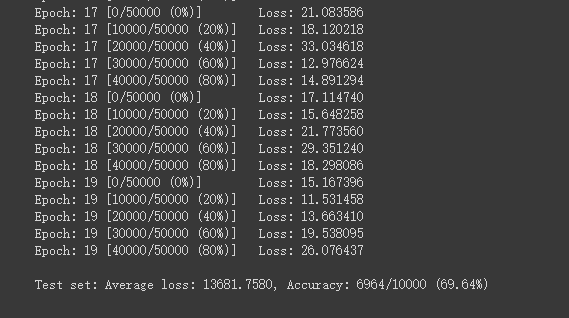
After establishing the network, we are going to train the model. The method is very simple, apply the training function, the forward calculation process, cleared gradient, reverse the gradient optimization process, error back propagation calculation, and finally the gradient is updated and the training model is saved.



We also need to set up a check function to verify the test epoch and calculate the loss. I used the cross entropy method of the torch.nn module to determine the calculation standard of loss, and also used the Adam method in the Optim optimizer of the torch.nn module.

Finally, set the loop and model storage location and start training. Here’s the result I got. Beside, I use the gpu training.





After the 19 epoch training, the lenet5 framework's classification and recognition of cifa10 images accuracy reached nearly 70%, which is a very satisfactory result.

Make a simple result test, select a small batch of data for prediction, take out the first set of data, return the image data and labels, and then pass the image data into the model for prediction. We can get the result:

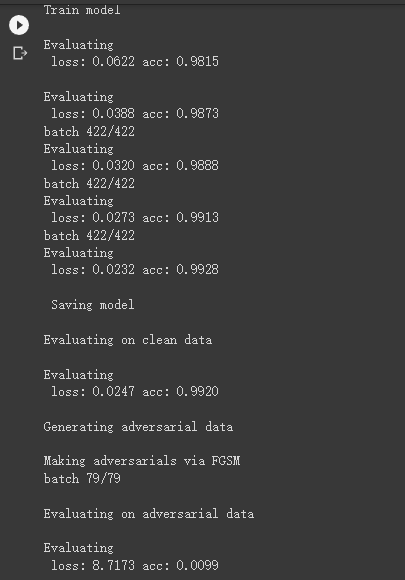


After consulting some online materials, the convolutional neural network designed by lenet5 is relatively shallow, and the prediction accuracy is not too high.

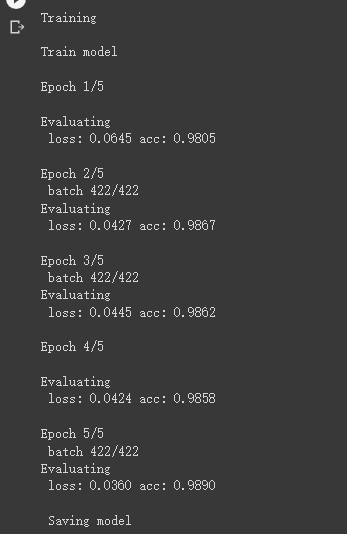
Part2:

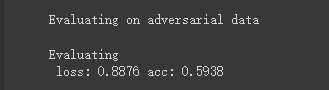
In the part 2 of this project, we are going to implement several attack models on the Mnist dataset and try the defense mechanism. The methods and implementation of the three attacks are very simple, that is, first use the tensorflow platform to train a non-disturbing model on the mnist data set. There are already integrated mnist datasets and training codes on tensorflow, here we only need to set the parameters of the network. Then add the attacking algorithm to train a new model and compare the results of the final recognition accuracy of the two models. Let's discuss the three attacks one by one.

First is the fgsm attack model. When training the classification model normally, the network learns features based on the input image, and then passes through the softmax layer to get the classification probability, then the loss function calculates the loss value based on the classification probability and the true label, returns the loss value and calculates the gradient (that is, gradient back propagation), and finally the network parameters are updated based on the calculated gradient. The purpose of updating the network parameters is to make the loss value smaller and smaller, so that the probability of the correct classification of the model is higher and higher. The purpose of the fgsm attack is not to modify the parameters of the classification network, but to modify the pixel value of the input image so that the modified image can disrupt the classification of the classification network. Then, according to the training process of the classification model, the loss value can be returned to the input. image and compute gradients. We only need to add the calculated gradient direction to the input image, so that the loss value of the modified image passing through the classification network is larger than the loss value of the unmodified image passing through the classification network. probability is reduced. The attack model of fgsm on tensorflow is easy to find, and we apply it to training here after some minor modifications. Finally, the figure below shows the accuracy and loss of network model recognition after normal training and fgsm attack by comparison.



Second is the cw attack model. A CW attack is a Target Attack that adds imperceptible perturbations to attack examples, causing the model to give a false label with high confidence. The algorithm regards the adversarial sample as a variable. The principle of the specific operation is to make the difference between the adversarial sample and the corresponding clean sample as small as possible, or to make the adversarial sample make the model misclassify, and the probability of the wrong class is as high as possible. . The advantage of the cw attack is that the disturbance added by the adversarial samples generated by the CW attack is almost invisible to the human eye. On the contrary, the disturbance generated by the adversarial samples generated by FGSM and PGD is relatively vague. And CW's attack effect is better, it can successfully attack many defense methods, and we can also adjust the confidence level by ourselves. The following is the result after the cw attack. Maybe because the amount of model data is relatively small, I did not see a big difference here.

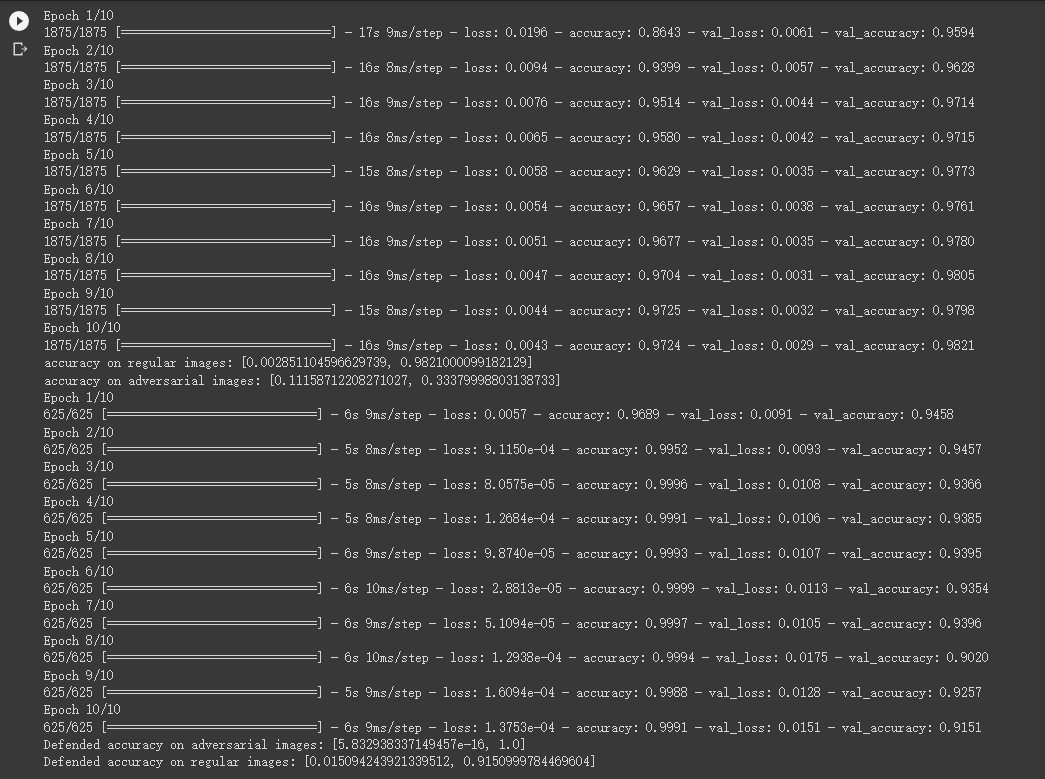




The third one here is the deepfool attack method. DeepFool attack is an iterative attack method. There are many mature encapsulation codes on the Internet. The DeepFool class contains two methods, one is the initialization of hyperparameters, and the other is to generate adversarial samples. Deepfool is a classic adversarial attack method, which defines the sample robustness and model robustness mirrors for the first time, and it can accurately calculate the perturbation of deep classifiers to large-scale datasets, thereby reliably quantifying the robustness of the classifier. It can also be said that it is an algorithm optimized for robustness evaluation based on fgsm. This algorithm can generate the least disturbance based on the gradient iterative method and can have a high attack accuracy. The following figure is the result after applying the deepfool algorithm.



Finally, here is an example to defend against attacks. I use the fgsm attack as a sample attack, and the defense method is the classic fgsm adversarial training. Adversarial training is an intuitive defense method against adversarial examples, which attempts to improve the robustness of neural networks by training with adversarial examples. The internal maximization optimization problem is to find the most effective adversarial examples, which can be achieved by carefully designed adversarial attacks. The external minimization optimization problem is the standard training procedure for loss function minimization. The final network should be resistant to adversarial attacks that generate adversarial samples used in the training phase. This kind of targeted training is actually the most effective defense against a single attack, but it lacks randomness and generalization. In other words, it cannot face multiple attack methods. Below is the result of the adversarial training of fgsm, we can see that the fgsm attack reduced the original recognition rate from 98% to 35%, and then after the adversarial training, the recognition rate returned from 35% to 90%.



In addition, there is a comparison of the accuracy after the change of the 𝜖 value required by the job. The result is definitely that the larger the value of 𝜖, the lower the accuracy. Because the value of 𝜖 relates to the size of the linear disturbance, the larger the linear disturbance, the greater the degree of attack, and the more serious the natural decline of the recognition rate. Here we show the effect of changing the value of 𝜖 in the attack model. The 𝜖 increase from 0.05:

