Programming Homework 2 CIFAR-10 Classifier

Yuheng ZHAO

November 2022

1 Contents and File Structure

According to the requirements, this program includes:

1. image_classifier.py, source code of the cifar-10 image classification pipeline. We imply function runModel and by configuration, supports different types of neural networks to run the classification process, which includes **CNN**, **MLP** and **MLP with some modifications**(without non-linear ReLU activation function). We want to compare the different results of different network structures.
2. plot.py is used to plot the results of the classifier, including losses of each epoch under different network structures.
3. ./img folder contains the output of plot.py.
4. ./log folder stores the log data for the training process, including the average loss and accuracy of each iteration. The log files are organized by models, each model has one unique log file which is easy to identify. In this approach, epoch size is set to 20, and the last piece of the log data contains the final accuracy of the model.

2 Result

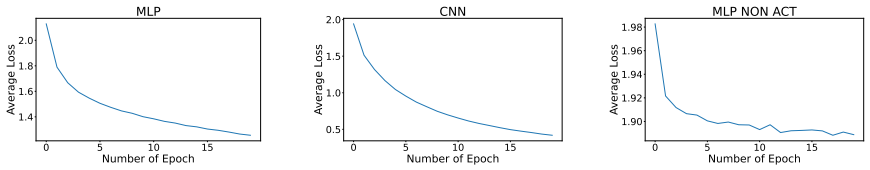
In this part we illustrate the different network structure of each model (see subsection 2.1), the result of model training, including the average loss of each epoch and the final accuracy (see subsection 2.2), in which the data can be found the data in ./log folder. We then discuss the difference of each model and the results in subsection 2.3.

2.1 Network Structure

The network structure of **MLP** and **CNN** is designed as required in the document, Table 1 shows the different model structure. ReLU function is used as activation function of MLP and CNN, we'll also discuss using the MLP model without the non-linear activation and see the difference in accuracy and loss.

Layer #	MLP (in dim → out dim)	CNN
1	Linear 3072→1024	Conv2D, in channel: 3, out channel: 64 kernel: 3×3, stride: 1, padding: 1
2	Linear 1024→512	Conv2D, in channel: 64, out channel: 128 kernel: 3×3, stride: 2, padding: 1
3	Linear 512→256	Conv2D, in channel: 128, out channel: 256 kernel: 3×3, stride: 2, padding: 1
4	Linear 256→128	Conv2D, in channel: 256, out channel: 256 kernel: 3×3, stride: 2, padding: 1
5	Linear 128→64	Linear 4096→1024
6	Linear 64→32	Linear 1024→1024
7	Linear 32→10	Linear 1024→10

Table 1: Network structure of different models.



(a) MLP (b) CNN (c) MLP without ReLU

Figure 1: Average loss of each epoch using different models.

2.2 Loss and Accuracy

We trained the models with SGD optimizer and a momentum of 0.9, a learning rate of 0.01. We report average loss after each epoch of the training data. Figure 1 shows the change of loss in each epoch. Among all the model we trained, the loss of each iteration decrease and finally converge at a low value as the number of iteration increases.

Table 2 shows the final accuracy of the models after 20 iterations, we can see that the final loss and accuracy varies significantly between different types of models.

	Fin. Train Loss	Accuracy
MLP	1.2556	52.68%
CNN	0.4194	82.61%
MLP without ReLU	1.8888	37.71%

Table 2: Final loss and accuracy of different models.

2.3 Discussions

In this program, we used CNN, fully-connected neural networks (MLP), and MLP without the non-linear activation function to finnish the image classifier on CIFAR-10 dataset. From the aforementioned result, CNN has a significant better performance of accuracy, which is 82%, much better than the two types of MLP, which are no more than 55%. This is probably because when doing image classification, fully connected layer use the whole input that might not work well for all kinds of images, in other words, fully connected layers are depend on the shape of the train images, which might not be a good thing for the over all model. Another possible reason is that fully connected layers have a larger number of weights thus highly prone to over-fitting whereas a single convolution operation reduces the number of parameters significantly which makes it less prone to over-fitting.

Also we can see that removing the non-linear activation function also reduce the final accuracy of the model. That's because the purpose of the activation function is to introduce non-linearity into the network, which means that the output cannot be reproduced from a linear combination of the inputs. So without a non-linear activation function the neural network (no matter how many layers it has), will behave as a single-layer perceptron, cuz summing those layers will just gives another linear combination.