COMP5212 Machine Learning
Programming Homework 2

LO, Li-yu
20997405
e-mail: lloac@connect.hkust.hk

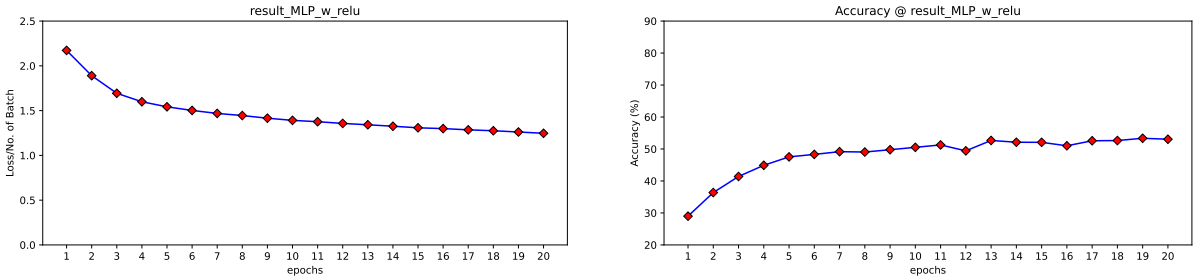
November 15, 2023

Abstract

In this assignment, we aim to train a multi-class classifier on CIFAR-10 dataset with following models: (1) **Multi-Layer Perceptron (MLP)** and (2) **Convolutional Neural Network (CNN)**. In particular, We adopt PyTorch library as our main programming framework. Below then reports the training results. Note that all the training was carried on GeForce RTX 3080 Ti.

Multi-Layer Perceptron

We first discuss the model using multi-layer perceptron. In which, 7 fully connected NN layers are deployed with ReLU activation function. The input/output sizes of each layer is defined to reduce the dimension equally after one NN + activation layer, from initial data input size 3072 to final output size 10. The input/output sizes of each layer are therefore: 3072/2633, 2633/2194, 2194/1755, 1755/1316, 1316/877, 877/438, and 438/10. As for training, we utilize SGD-M, with *learning rate* $\alpha = 0.01$, momentum coefficient $\beta = 0.9$ and *epoch* $= 20$. On the other hand, the cost function is cross-entropy. Below is the training results and calculated accuracy:



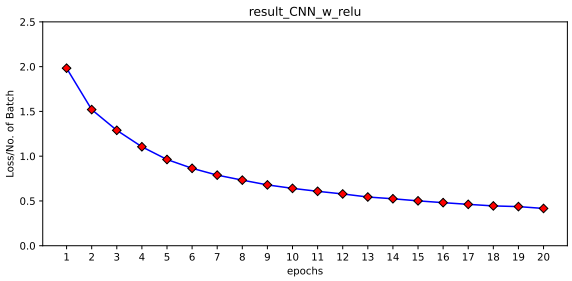
(a) MLP Loss v. Epochs

(b) MLP Accuracy

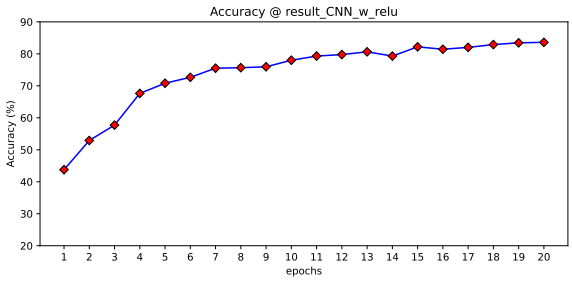
Figure 1: Multi-Layer Perceptron training results.

Convolutional Neural Network

As mentioned, a classifier with convolutional neural network is also trained. In which, 4 convolutional layers and 3 fully connected layers, with ReLU activation function is adopted. Particularly, to fulfill the training requirement, where the input size of 1^{st} fully connected layer should be 4096, each CNN layer information is as followed: (1) 2D convolution kernel 5×5/input channel 3/output channel 66, (2) 2D convolution kernel 5×5/input channel 66/output channel 128/stride (2,2)/padding (1,1), (3) 2D convolution kernel 5×5/input channel 128/output channel 192, and (4) 2D convolution kernel 5×5/input channel 192/output channel 256/stride (2,2)/padding (1,1). As for the fully connected layers, the input/output size of each FLC layer information is as followed: 4096/2734, 2734/1372, and 1372/10. Similar to MLP, we utilize SGD-M, with *learning rate* $\alpha = 0.01$, $\beta = 0.9$ and *epoch* $= 20$. On the other hand, the cost function is cross-entropy. Below is the training results and calculated accuracy:



(a) CNN Loss v. Epochs



(b) CNN Accuracy

Figure 2: Convolutional Neural Network training results.

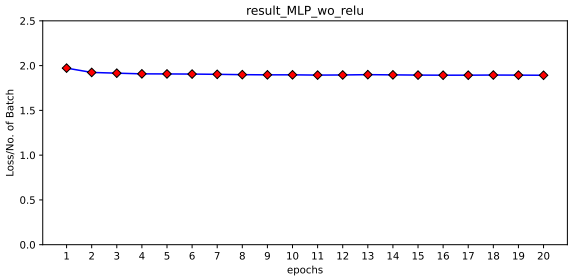
On MLP v. CNN

The difference in the performance of the two models are obvious. From the reported training results, it could be seen that CNN has a superior model accuracy, where it reaches around 85%, where MLP only achieves 50%, while having less loss. With a same training epochs, MLP was trained faster with 267 seconds, whereas CNN was with 308.

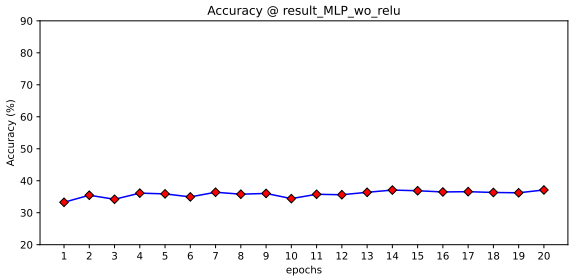
NNs without Activation Layers

The last section presents the training results when we remove the activation layers, i.e., ReLU in this case. Below first presents training results of the two.

Multi-Layer Perceptron:



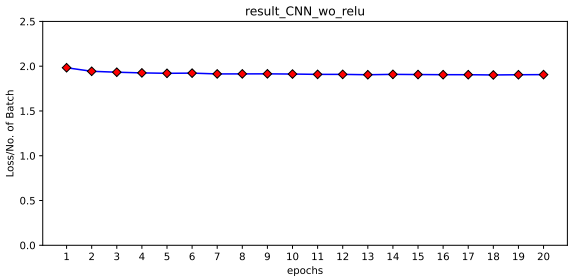
(a) MLP (no ReLU) Loss v. Epochs



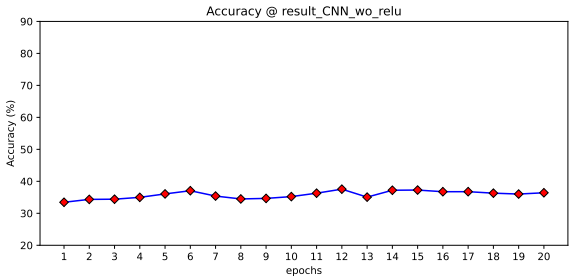
(b) MLP (no ReLU) Accuracy

Figure 3: Multi-Layer Perceptron with no activation layer training results.

Convolutional Neural Network:



(a) CNN (no ReLU) Loss v. Epochs



(b) CNN (no ReLU) Accuracy

Figure 4: Convolutional Neural Network with no activation layer training results.

From above, as observed, without the activation layers, the performance drops drastically in both MLP and CNN. The main reason behind this due to the fact that, without the non-linearity induced by ReLU, the models are simply linear. Training the two models with such

settings could therefore be seen as a linear regression models with a higher dimension. And therefore, that is why although CNN achieves higher accuracy in the original settings, under the circumstances where the ReLU layers were removed, it behaves as poorly as MLP, achieving an accuracy between 30 % and 40 %.