Deep Learning Homework # 1

*CNN experiment*

Jie yong Shin

data) I splited the fashion mnist dataset to train, validation and test datasets. I drawed graph with train and validation and used evaluation function to test datasets so the evaluated value will be placed at the below of train and validation graph. (figure [1])

model) To perform this experiment, I made a simple model first.This model is made with Keras Sequential class. The basic model configuration of layers is,

[input] - [colvolution] - [maxpool] - [convolution] - [maxpool] - [convolution] - [flatten] - [fully connected] - [soft max]

Used 'Adam' optimizer and since the data label is consisted of 0 to 9, I used 'spars categorical crossentropy' function for loss function. Did not used batch normalization and drop out until last experiment. To make sure, I set 100 epochs to every experiment. You can see it intuitively at figure [0].

a) mean subtraction and normalization (figure [a])

For Mean Subtraction, I subtracted average value of each validation data and train data. For normalization, I experimented MSCN(Mean Subtraction and Contrast Normalization) and scaling. For MSCN, I divided those subtracted values with standard deviation. For scaling, I divided data by 255 which is number of channels of gray scale. In train and validation graph, it is hard to notice big difference between those two ways but they have difference in test data evaluation. Mean Subtraction got 0.5 percent point and MSCN got 20 percent point of loss in test data evaluation. I also tried scaling which is generally used when classifying Fashion MNIST data. With this method, the validation data did not trained and got 28 percent point of loss in test data evaluation.

b) Xavier and He initialization

Since the best out put of normalization was Mean Subtraction, I used Mean subtraction to normalize input data. I used 'he\_normal' and 'glorot\_normal' functions from Keras. As you can see in figure [b], at train and validation loss graph, Xavier initialize appears overfitting and unstability of loss value. However, graph of He initializer got much stable graph with validation loss, and overfitting was much less than Xavier initializer. It is natural that these result appear because He initializer is supplement to Xavier initilizer.

c) change Network configuration

I used the model with He initialization which represented higher performance. Changed number of layers such that enable to use both same padding and valid padding. As a result, number of parameters increased more than 8 times bigger than original one. From about 93 thousand parameters to about 763 thousand parameters. You can see it by comparing figure [c0] and figure [c1]. Since I don't use drop out for now, this number of parameters occured lower performance. In figure [c3], as the epoch proceeds, loss and accuracy of validation shows noisy movements around the training loss. In addition, original model was stable with training but this model shows noisy movements with training as well. As we can see in first configuration changed model, too much of parameters can occur lower performance so I tryed another configuration. I tried to reduce the number of units and train them with fewer parameters while leaving the layer as it is. The number of parameters greatly reduced to about 42 thousand. It showed a very stable graph because it had an appropriate number of parameters. Training and validation accuracies were learning very narrowly, and loss also showed the best fit (figure [c3]).

d) experiment different optimization techniques (figure [d])

In the case of Adadelta, a continuous graph was drawn compared to Adam, so learning proceeded more stably, but accuracy itself showed lower performance. Adam introduces the first moment in RMSProp to achieve the combined effect of RMSProp and momentum, so it performed better than Adadelta. Nadam drew a higher accuracy and better learning graph than Adam. Nadam was able to perform higher than Adam by supplementing the momentum by adding NAG to Adam.

e) different activation functions

ReLU performed better than SelU or Leaky ReLU. (figure [e])

f) regularizations (figure [f])

For the weight decay of the experimental model, 0.0001 was applied to both the L1 and L2 regulations. First of all, when the regulation was not applied, there was a phenomenon that the validation loss bounces while epoch processes, but after the L1 regulation was applied, the big bouncing loss disappeared. The L2 regulation showed the less loss and insensitive graph than L1. Next, I tried to apply L1 and L2 at the same time. Although test evaluation was little bit better than the initial value, it showed lower performance at training and validation than using only L1 or L2. In the case of dropout, 0.5 was applied only once when passing the first dense layer after flattening (figure [f2]). It performs much lower at test than the L2 regulation, whether due to the dropout rate or the location of the dropout layer.