Fashion-MNIST dataset classification using convolutional neural network and multilayer perceptron

Chenglin, Zhou.(5411-8969) MLP implementation and hyperparameter selection, report writing

Zhengcheng, Song.(9455-5960) CNN implementation and hyperparameter selection, report writing

***Abstract*—Nowadays, deep learning becomes the mainstream method for image classification. In this report, we implement a multiclass classification task on the Fashion-MNIST dataset with Multilayer Perceptron (MLP) and Convolutional Neural Networks (CNN). A set of experiment have been done to select different hyperparameters properly. Finally, we generate a two hidden layers CNN model with 91.78% accuracy and a two hidden layers MLP model with 89.48% accuracy.**

***Index Terms*—CNN, MLP, hyperparameters, Fashion-MNIST**

I. Introduction

In this project, we respectively demonstrate a CNN model and a MLP model on a newly built multiclass image classification dataset.

A series of experiments are implemented to explore proper hyperparameters for good performance. After good results have been generated on validation dataset, we test our best models several times and reach average accuracy of 91.78% and 89.48% for CNN and MLP model. Other than proper selected hyperparameters, Batch normalization and Dropout could significantly improve the result in our experiments. In addition, it is proved under this specific project, CNN models performs better than MLP models.

In Methods, CNN and MLP models' architecture and hyperparameters used for testing are showed. Then, three methods to improve the performance are introduced—Dropout, Batch Normalization and Early Stop.

In Results, we first briefly present Fashion MNIST dataset and how we split data in it. Secondly, we show the experimental result of our models. At last, comparisons are done with CNN and MLP models.

In Discussion, we show how we select our hyperparameters in detail including choices of Dropout proportion, number of layers, number of units, learning rates, kernel sizes, Batch Normalization, and optimizer.

II. Methods

A. CNN model

For this project, both one hidden layer and two hidden layers CNNs have been implemented and tested. The two hidden layers network is selected due to better performance. The structure is shown in Figure 1.

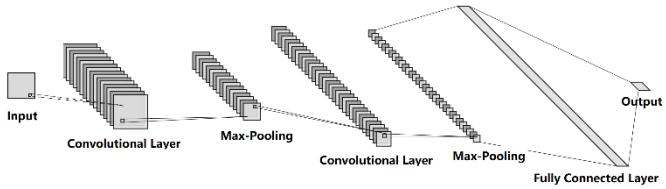

Fig.1.CNN structure.

For both convolutional layers, ReLu is used as activation function and Batch Normalization is used before activation. Dropout method is applied before output layer. The output layer is a fully connected layer with softmax activation function because of the multiclass classification task. Other parameters of the network structure are listed in Table I.

TABLE I. CNN PARAMETERS

Convolutional Layer	Kernel:3 × 3, stride = 1, padding = 2, maps = 24
Max-Pooling	2×2
Convolutional Layer	Kernel:3 × 3, stride = 1, padding = 0, maps = 48
Max-Pooling	2×2
Fully Connected Layer	Input Dimension= 720, Output Dimension= 10

According to experiment results, we select a learning rate of 0.001 and batch size of 100. The optimizer is Adam (beta1=0.9, beta2=0.999, epsilon=1e-8), and the loss function is cross-entropy function. Training will be stopped either 100 epochs is done, or an early stopping criterion is met.

B. MLP model

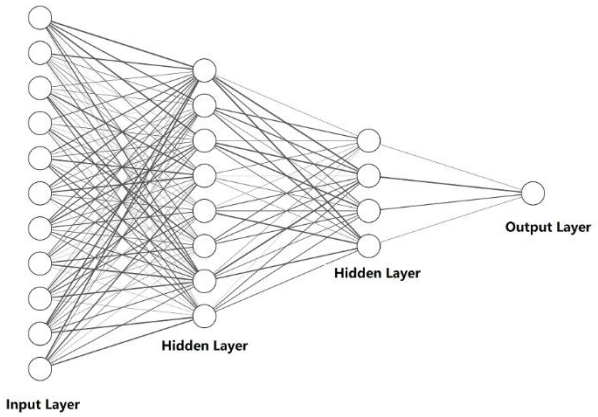

Fig.2.MLP structure.

MLP model have much similarity with CNN model. A two hidden layers MLP model is generated with Batch Normalization and Dropout in each hidden layer. The activation functions are also ReLu in hidden layers and softmax in output layer. The architecture is listed in Table II

TABLE II. MLP PARAMETERS

Fully Connected Layer	Input Dimension= 784 Output Dimension= 512
Fully Connected Layer	Input Dimension= 512 Output Dimension= 256
Fully Connected Layer	Input Dimension= 256 Output Dimension= 10

Other parameters are all the same with CNN model.

C. Dropout and Batch Normalization

With the increasement of model size and training epoch, overfitting occurs. Generally, overfitting happens when the model 'remembers' too much of the training dataset, even the noise in it. Hence, the model will perform much better in training data than test data. There are several techniques to handle this problem. In our models, Batch Normalization and Dropout have both been adopted.

Dropout temporarily remove randomly selected units from the architecture during training[1]. Thus, learning parameters is unlikely to be highly related to certain training data, and network can generalize better to avoid overfitting.

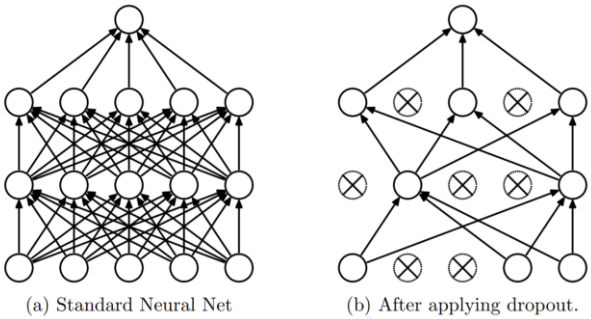

Fig.3.How to Dropout.[1]

Additionally, Dropout simulates model combination of multiple different models without actually implement multiple networks. Therefore, Dropout nearly always contributes to the improvement of performance as well.

Batch Normalization is a method to accelerate training and improve the capability of network by normalizing the input to layers[2]. The normalization is done through re-centering and re-scaling. Although Batch Normalization is widely used in many architectures, its mechanism is still under study. Different explanations[3][4] have been proposed towards how it works.

Moreover, whether to use Dropout together with Batch Normalization is argumentative as well. In some research[5], it is believed Dropout and Batch Normalization often lead to a worse performance when they are combined together.

On the other hand, some articles[6] show that using Batch Normalization alone cannot prevent overfitting. In our experiments, Dropout does a better job to prevent overfitting. Yet, combination of Dropout and Batch Normalization not only accelerate training but also improve performance than only adopting Dropout. Detailed result is in the discussion part of this report.

D. Early Stopping

Although Dropout is applied in MLP model, overfitting may still occur. Therefore, rather than simply stopping after certain epochs, we introduce an easy early stopping criterion. If the smallest validation loss does not update after 20 epochs, we stop training.

III. RESULTS

A. Datasets

Fashion MNIST dataset[7] comprises 28×28 grayscale images of 70,000 fashion products from 10 classes, with 7,000 images per class. In the dataset, there are 60,000 images as training set and 10,000 images as testing set. 10 labeled classes with a few images shown as example are presented below in Figure 4.

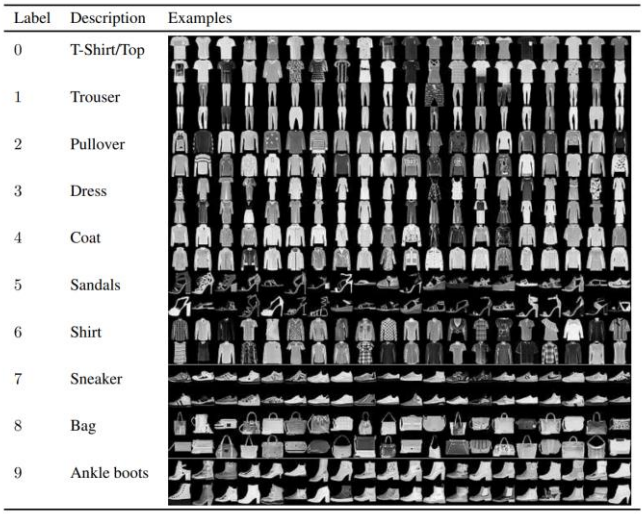

Fig.4. Fashion MNIST dataset[7]

B. *Data Splitting*

Fashion-MNIST dataset have already split test data out, so we only need to divide validation dataset out of training dataset. 2/3 of data will be used for training, and 1/3 of data will be used for cross validation. We shuffle all the training data and select same number of exemplars for each class across the dataset.

C. *Results*

To avoid randomness in training, 10 CNN models are trained under same hyperparameters. Figure 5 shows the Validation loss during training for 10 models.

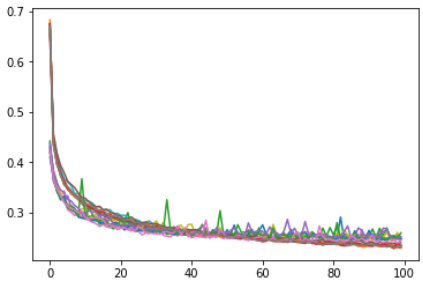

Fig.5. Validation loss during training for 10 models

The average test accuracy reaches 91.78% for CNN models. Figure 6 is one confusion matrix of 10 CNN's test results.

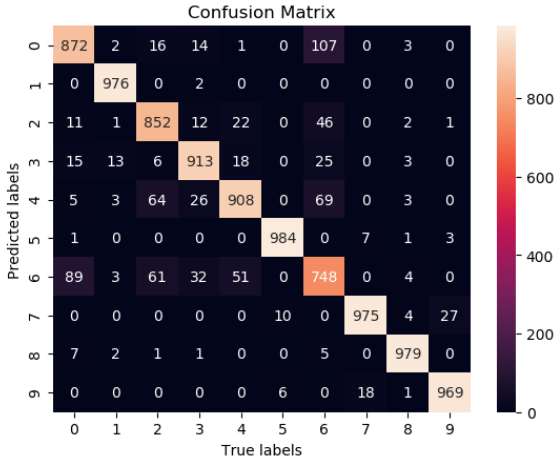

Fig.6. One confusion matrix of CNN's test results

For MLP model, test procedure remains the same. The average test accuracy reaches 89.48%. Figure 7 is one confusion matrix of 10 MLP's test results.

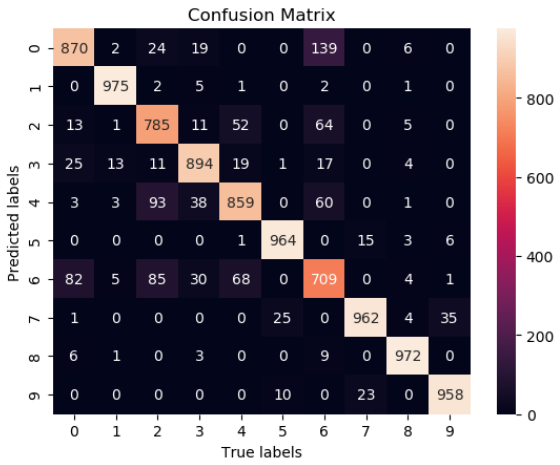

Fig.7. One confusion matrix of MLP's test results

D. *Comparison*

CNN models not only have a better overall performance, but also have better accuracy for each class. The variance of loss of CNN is smaller than MLP, which indicate CNN's performance is more stable. Moreover, in experiment, CNN shows higher Robustness. The change of hyperparameters causes much less influence on CNN structure. Intuitive comparison can be found in Discussion part.

With the acceleration of GPU training and the development of packaged neural network tools. The advantage of MLP—easier to compute and implement—become insignificant. The exceed of overall performance makes CNN much popular than MLP nowadays.

IV. Discussion

In this part of the report, we are going to discuss how hyperparameters influence the performance of both models and show how the hyperparameters is selected in our models. When we alternative one hyperparameter, other parameters remain the same.

A. *Dropout*

As we applying Dropout in both networks, a new hyperparameter is introduced. Theoretically, no Dropout or the partition of Dropout is too small will lead to overfitting. On the contrary, too many Dropouts lower the performance and underfitting may happen. Moreover, for different size of models, Dropout rates may vary to get better result.

Additionally, these results answer the question that could Batch Normalization be used alone without Dropout. Experiment indicates that only use Batch Normalization cannot prevent overfitting.

For the final model, we use 0.6 Dropout rate.

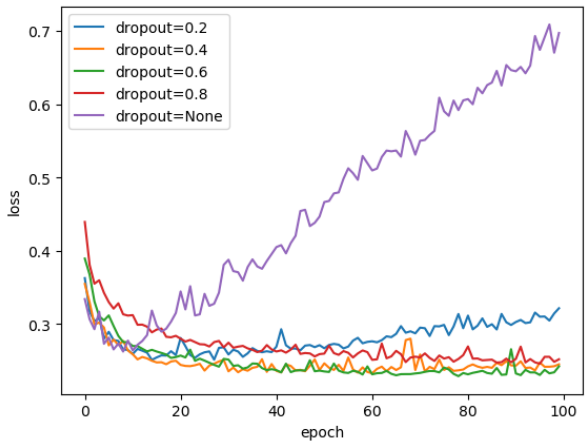

Fig.8.Result of different Dropout proportion in CNN.

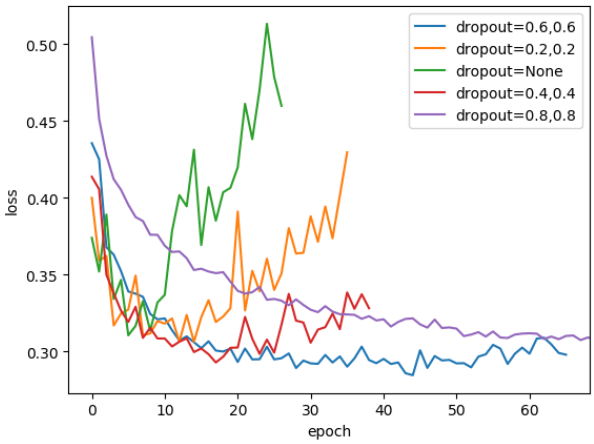

Fig.9.Result of different Dropout proportion in MLP.

B. *Number of Layers*

According to former homework, two layers architecture performs much better than one-layer structures. One advantage of the two layers structure is that it is easier to train. Hence, to test different models with some standards, we select number of units so they have similar number of trainable parameters in CNN models. For MLP, different units have been selected in one-layer structure.

Two layers architecture performs much better according to Figure 10 and Figure 11. We select two layers model for our final models.

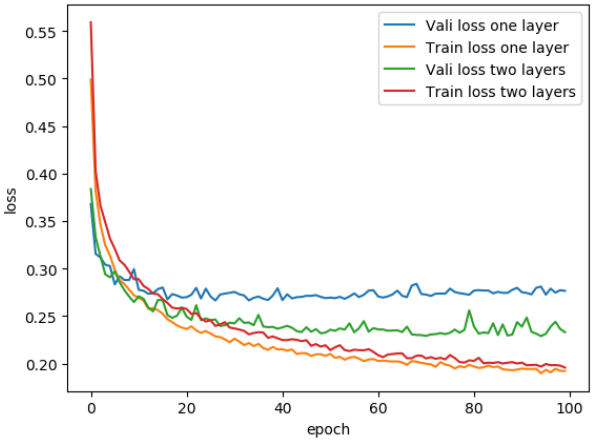

Fig.10.Result of different layers in CNN.

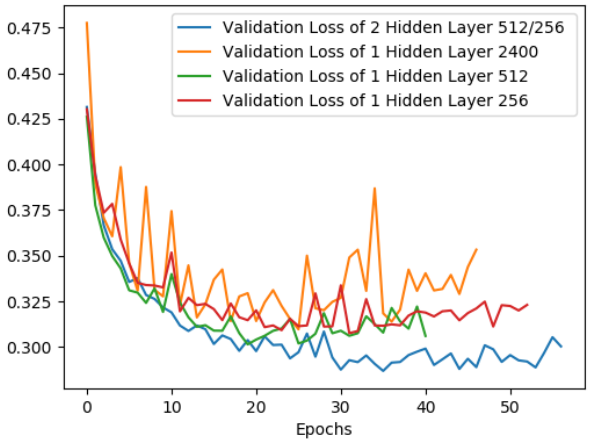

Fig.11.Result of different layers in MLP.

C. *Number of Units*

As the former result showing, two hidden layers architecture performs better. Hence, we only discuss how number of units in two hidden layers models affect the result.

Apparently, fewer units cannot select and learn enough features to do the classification. In contrast, too many units not only increase training time but also cause overfitting. Though we could increase Dropout partition to handle overfitting, Dropout too many units lead to worse result as we discussed before.

Figure shows three representative models in our experiment. The units in each layer are (10,20), (16,32) and (36,72).

As we can see, properly larger model leads to better performance. However, a model too large will either overfitting or performs worse according to Dropout proportion.

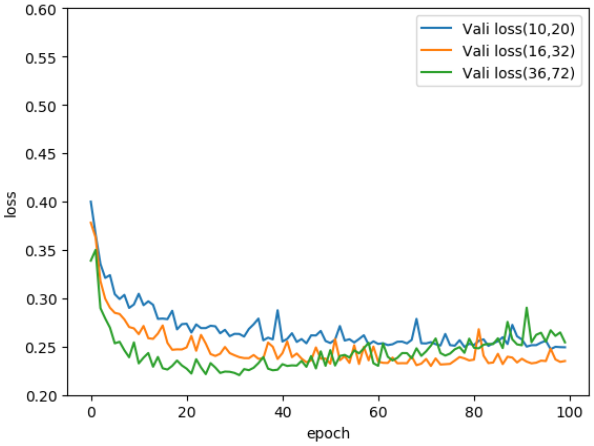

Fig.12.Result of different number of units in CNN.

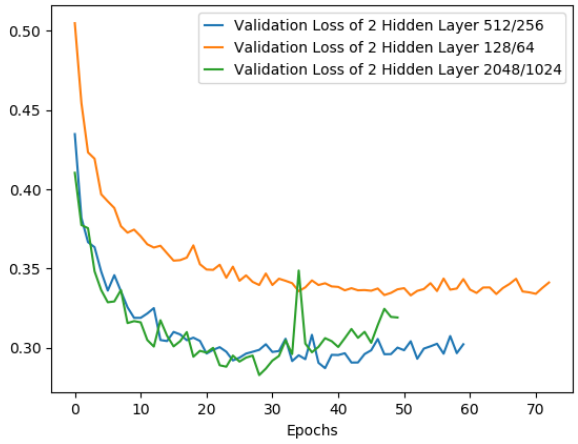

Fig.13.Result of different number of units in MLP.

D. *Kernel Size in CNN*

Large kernel size in CNN would overlook at smaller details. Therefore, it may lose some significant features, but it may also reduce the disrupting of noise. In contrast, smaller kernel size will extract more detailed information.

Figure 14 shows the different result of different kernel size. We choose 3 as our kernal size.

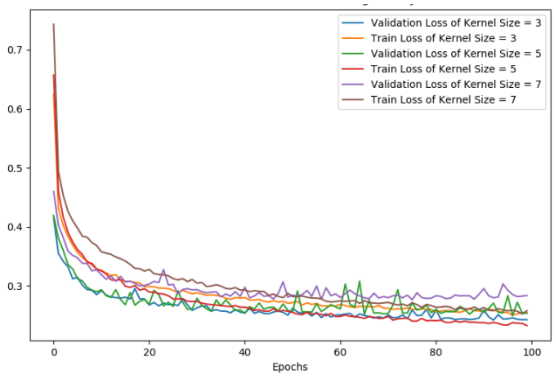

Fig.14.Result of different kernel size in CNN.

E. *Learning Rate*

When learning rate is less than 0.001, there is no significant difference of performance. However, when learning rate equals to 0.01. The loss becomes unstable and hard to convergent.

Hence, we choose 0.001 as our learning rate.

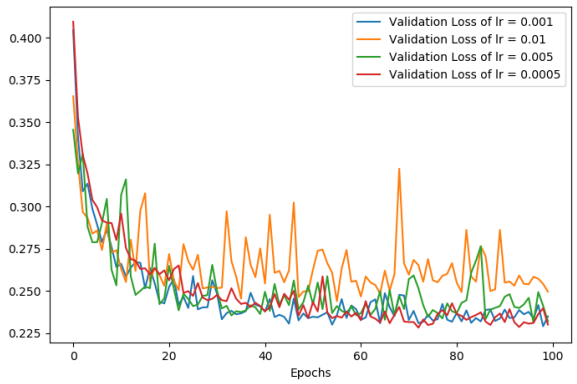

Fig.15.Use different learning rate in CNN.

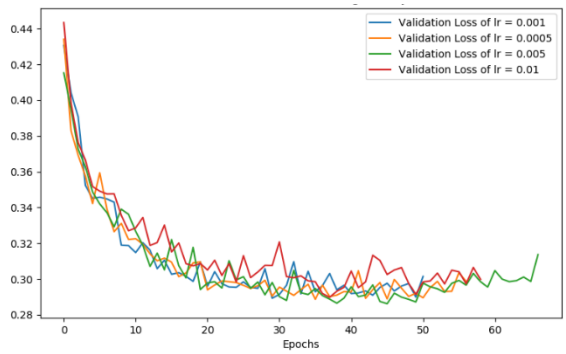

Fig.16.Use different learning rate in MLP.

F. *Batch Normalization*

The argument about whether to use Batch Normalization is introduced in Methods. However, in this specific experiment, it is better to use both Dropout and Batch Normalization.

Figure 17 and 18 shows that Batch Normalization could apparently increase the performance.

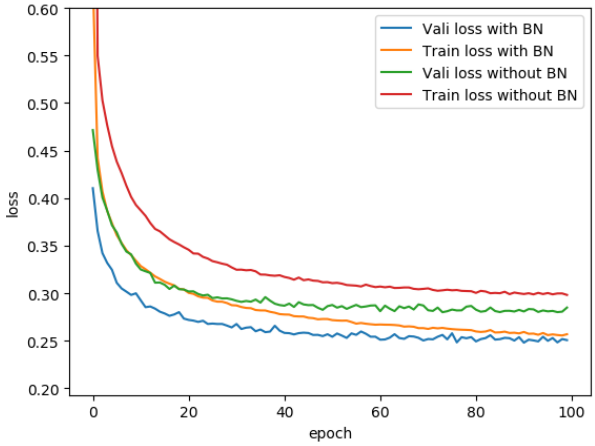

Fig17.Loss of CNN with or without batch normalization.

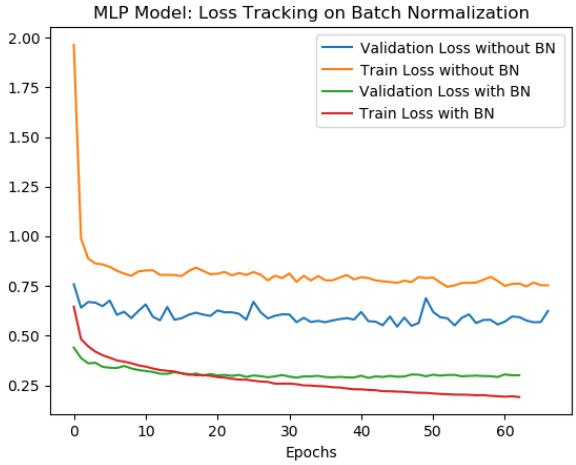

Fig.18.Loss of MLP with or without batch normalization.

G. *Optimizer*

SGD is easy to fall into local optimal point. SGD with momentum and Adam have more chance to avoid this kind of situation. Comparing Adam and SGD with momentum, Adam

convergents faster and oscillatate greater within a reasonable range. We believe that this charactristic of Adam could help it to find better optimal point.

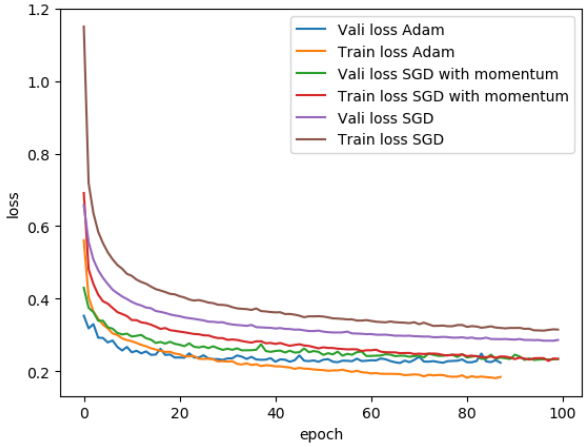

Fig.19.Use different optimizer in CNN.

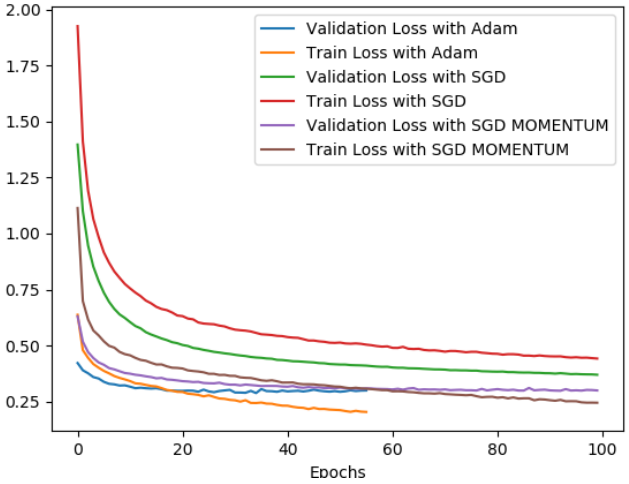

Fig.20.Use different optimizer in MLP.

V. Conclusion

A. *Preventing overfitting*

Dropout does a decent job for preventing overfitting, and it could also improve performance. Under some specific circumstance, early stopping could be adopted as well to make sure overfitting will not happen. However, in our models, Batch Normalization make no contributions to prevent overfitting.

B. *Batch normalization*

Though its mechanism is still under study, Batch Normalization could significantly help models to get better results. Dropout and Batch Normalization being adopted together bring the performance to a new level in this project although whether should models use them together remains controversial

C. *CNN and MLP*

There is no doubt CNN's performance fully exceed MLP in this project. With the fast development of GPUs and deep learning software. CNN may replace MLP in most of areas. However, MLP is still easier to apply and to compute, so this model may still be chosen under specific circumstances.

References

[1] Srivastava, Nitish & Hinton, Geoffrey & Krizhevsky, Alex & Sutskever, Ilya & Salakhutdinov, Ruslan. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research. 15. 1929-1958.

[2] Ioffe, Sergey & Szegedy, Christian. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.

[3] Santurkar, Shibani & Tsipras, Dimitris & Ilyas, Andrew & Madry, Aleksander. (2018). How Does Batch Normalization Help Optimization? (No, It Is Not About Internal Covariate Shift).

[4] Yang, Greg & Pennington, Jeffrey & Rao, Vinay & Sohl-Dickstein, Jascha & Schoenholz, Samuel. (2019). A Mean Field Theory of Batch Normalization.

[5] Li, Xiang & Chen, Shuo & Hu, Xiaolin & Yang, Jian. (2018). Understanding the Disharmony between Dropout and Batch Normalization by Variance Shift.

[6] Zhang, Chiyuan & Bengio, Samy & Hardt, Moritz & Recht, Benjamin & Vinyals, Oriol. (2016). Understanding deep learning requires rethinking generalization.

[7] Xiao, Han & Rasul, Kashif & Vollgraf, Roland. (2017). Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms.