Assignment 1

Jielin Feng:480019484, Yifeng Gao:500083754, Zedong Wen:500334863

April 10, 2022

1 Introduction

1.1 Aim

This study aims to build a powerful and applicable Multilayer Neural Network which could classify data with 128 features into 10 given classes. 10 given classes are divided into [0,1,2,3,4,5,6,7,8,9]. The training set contains 50000 samples, which are used to build the multilayer neural network, and the test set contains 10000 samples, which are used to verify the performance of the neural network. In this study, firstly, it is aiming to find out the effect of the optimization method and regularization method listed from left to right: More than one hidden layer, Relu activation, Weight decay, Momentum in SGD, Dropout, Softmax and cross-entropy loss, Batch normalization and Gelu activation. Secondly, it is aiming in compare and evaluate the accuracy and computation time of each model after applying the methods listed above, and further discussed the advantages and disadvantages of using the specific method. Lastly, after each method is applied independently, it is expected to build up an overall best model that applies more than one optimization method or regularization method that let the model reach very good performance, the final model will be tuned with different parameters to find the most suitable parameters and performance will be discussed. The principle and performance evaluation of the final best model will be further discussed.

1.2 Why is this study important

The importance of this study is mainly reflected in the following aspects. Firstly, the compilation of multilayer neural networks based on basic computing tools such as Numpy, Scipy can deepen our understanding of this and make the basic knowledge firmer. The classification system constructed by multilayer neural networks can greatly improve the classification efficiency. A high-precision classifier can also improve the convenience of searching. For example, in typical image classification, it is convenient to query the desired image by placing images of the same category under the same search. Secondly, nine optimization or regularization methods are added to our deep learning structure. Improving the structure of deep learning is the development trend of deep learning. We need to know the possible improvement aspects of deep learning. Besides, the neural network of deep learning is an important modern technology, and it can be applied in various fields, such as image recognition and medical diagnosis.

2 Methods

2.1 Principle of different models

2.1.1 Based Model

The following models run in 2.1 are all based on a learning rate of 0.0001 and epochs of 150.

Design

Based model is a model with a 4-layer structure without any optimization method. In which the input layer contains 128 nodes; the two hidden layers in the middle contain 64 and 32 nodes respectively; since the data source contains 10 different classes from 0-9 , so the output layer contains 10 nodes. The activation function of both hidden layers is set to "tanh", and the activation function of the output layer is softmax.

Principle of softmax and cross-entropy

Softmax maps the outputs of multiple neurons to the (0,1) interval so that their sum is 1, which is in line with the characteristics of probability. We can find the point with the highest probability as the output of the final prediction. CrossEntropy describes the distance between two probability distributions. The smaller the cross entropy, the closer the two are. After the output of the softmax layer, we get the probability distribution, so it is used with CE. This makes it easy to compare subtle differences and, as a convex optimization function, facilitates gradient descent

2.1.2 Hidden-layer Model

Design

Hidden-layer Model adds two extra hidden layers based on the structure of the Based Model. This design is used to experiment with the effect of adding hidden layers on the performance of the final model. Hidden-layer Model has 6 layers, of which the input layer contains 128 nodes; the four hidden layers contain 64, 64, 32 and 32 nodes respectively, and the output layer contains 10 nodes; the activation functions of the four hidden layers are all set to " tanh". The rest of the settings are consistent with the Based Model.

Principle of Hidden-layer Model

The principle of adding hidden layers is that the increase of hidden layers can be used to fit non-linear functions. When the hidden layer is deeper, the more complex presentation can be learned. In theory, the deeper the layers, the stronger the ability to fit the function, and the more ideal data can be obtained. However, if the data is relatively simple, too deep layers may lead to over-fitting problems, which will increase the difficulty of training and cause the model to fail to converge. On the setting of neurons in the hidden layer, we design a decreasing structure of 64, 64, 32, 32. The reason for this setting is to let the hidden layer learn lower-level features in the front, and then feed back the low-level features to extract high-level features as the number of neurons in the hidden layer decreases.

2.1.3 Sigmoid Model

Design

Based on the Based Module, Sigmoid Model changes the activation function of the two hidden layers from "tanh" to "logistic". The remaining settings are consistent with Based Module.

Principle of Sigmoid Model

The principle of "sigmoid activation function" is to transform linearity into non-linearity. In the hidden layer, the activation function converts the value entering the neuron into a new output, which is passed to the next neuron. The sigmoid function will output the value in a range of 0-1 by $1/(1 + exp(-x))$. However, if the sigmoid function is used, the gradient might be very close to 0 in the backward propagation through multiple hidden layers. There is a high probability of gradient explosion and gradient vanishing.

2.1.4 Relu Model

Design

Based on the Based Module, Relu Model changes the activation function of the two hidden layers from "tanh" to "relu". The remaining settings are consistent with Based Module.

Principle of Relu Model

The principle of "relu activation function" is that relu is also a activation function. It converts the input into a range of max(0,x) output. This function keeps all positive values and replaces all negative values with 0. Its advantage is that it solves the gradient descent problem in a positive interval, has a faster calculation speed, and has a faster convergence speed. However, since it only retains all positive intervals, some neurons may not be updated.

2.1.5 Weight-decay Model

Design

On the basis of the Based Module, add Regularization methods weight decay. The remaining settings are consistent with the Based Module.

Principle of Weight-decay Model

The principle of weight decay which is so-called L2 regularization technique is Limiting the growth of the weights in the network, in which a term is added to the original loss function, and penalizing large weights. The purpose of weight decay is to adjust the model complexity to the loss function, which can prevent overfitting. This is because a smaller weight indicates a lower network complexity. Low-complexity networks tend to fit the data better, less possibility that overfitting might occur.

2.1.6 Momentum Model

Design

On the basis of the Based Module, the optimization method Momentum is applied to SGD. The remaining settings are consistent with the Based Module.

Principle of Momentum Model

Compared to Gradient Descent, which computes the gradient with all the data at once, SGD performs gradient updates for each sample. SGD is faster than Gradient Descent due to only one update at a time. However, SGD will be affected by more

noise, because the gradient is updated for each sample, and SGD cannot be in the optimal direction every iteration, which leads to the fact that although the training speed of SGD is fast, the accuracy cannot be guaranteed, and compared with Gradient Descent , the cost function will oscillate severely. Therefore, we applied momentum. If the current gradient in SGD is inconsistent with the historical update direction, the gradient will decay; if it is consistent with the historical update direction, the gradient in this direction will increase. This can make learning more stable, overcome the shortcomings of SGD to a certain extent, accelerate SGD through momentum, and suppress oscillations.

2.1.7 Dropout Model

Design

On the basis of the Based Module,add Regularization methods drop out. The remaining settings are consistent with the based Module.

Principle of Dropout Model

The principle of drop out is that in deep learning, when there are lots of parameters and few training samples, overfitting is easy to occur. And drop out can effectively alleviate the occurrence of overfitting. The principle is to set a probability "p" during forward propagation, so that the activation of neurons randomly stops working with the probability of "p", and only the remaining "p-1" is retained. Drop out reduces the complex supply relationship between neurons, In this way, the generalization performance of the model can be enhanced, and the model will not be too dependent on some local features, thereby improving the generalization performance and alleviating the problem of overfitting.

2.1.8 Batch-norm Model

Design

On the basis of the Based Model, Batch normalisation is added. The remaining settings are consistent with the Based Model.

Principle of Batch-norm Model

The principle of batch normalisation is that in the deep learning model, the parameters of each hidden layer are easy to change with the training, facing the problem of covariate shift, so that each layer has a different distribution, resulting in the parameters of the previous layer need to adapt to the new distribution. When the data entering the new distribution is input into the activation function, it will make the learning inefficient and may cause to gradient vanishing. Therefore, we add batch normalization. Batch normalization calculates the mean and variance of the data respectively, then normalizes it and finally calculate a scaling and translation Final distribution. Batch normalisation prevents the data from being distributed differently. It can improve the generalization ability of the model and effectively avoid the problem of gradient disappearance.

2.1.9 Mini-batch Model

Design

On the basis of the Based Module, set up mini-batch training. The remaining settings are the same as those of the Based Module.

Principle of Mini-batch Model

The Principle of Mini-batch SGD is that Unlike Stochastic GD, which updates the model for each training data, and Gradient Descent, which updates the model after all training data is calculated, mini-batch SGD sets the batch parameter and divides the training set into many batches, each of which contains a batch size numbers of data. mini-batch SGD calculates and updates parameters for each sub-batch of training data divided. Compared with Gradient Descent, mini-batch SGD has faster speed and better robustness. Compared with Stochastic GD, mini-batch SGD is more accurate and can avoid the problem of only finding local optimum to a certain extent.

2.1.10 Gelu Model

Design

Change the activation parameters based on the based model. Change all "tanh" to "gelu".

Principle of Gelu Model

The principle of the gelu activation function is also to increase the nonlinear transformation. The full name of gelu is "gaussian error linear units", which is a high-performance neural network activation function. In the process of building a neural network, on the basis of nonlinearity, adding random regularization can improve the generalization ability of the model. In which gelu introduces the idea of random regularity under the premise of having an activation function. gelu is actually a combination of dropout, zoneout, and relus. gelu multiply the input by a mask consisting of 0, 1, and the generation of the mask is randomly dependent on the input according to probability. Assuming that the input is X and the mask is m, then m follows a Bernoulli distribution $\Phi(x)$, $\Phi(x) = P(X <= x)$, X follows a standard normal distribution $\Phi(x) = P(X <= x)$, X obeys standard normal distribution, this choice is because the input of neurons tends to be normal. The distribution is set so that when the input x decreases, the input has a higher probability of being dropped, so that the activation transformation will randomly depend on the input.

$$GELU(x) = xP(X <= x) = x\Phi(x) \tag{1}$$

2.2 Design of the best model

Our best model uses the learning rate of 0.1, epochs = 200, [128, 256, 128, 10] for the network. The activation function is relu, and the output layer uses softmax. In terms of optimization methods, minibatch,momentum, dropout and batch normalization. In terms of parameter setting, the batch size of minibatch is 256, the gamma of Momentum is 0.9, and the drop rate is 0.5. The reason why we do this is that weight decay didn't show a good effect of preventing overfitting in previous experiments. While we are using momentum and batchnorm, both of them have an acceleration effect. In case of large epochs, dropout is used to prevent over-fitting.

3 Experiments and results

3.1 Performance in terms of different evaluation metrics.

Models	Accuracy(in percentage)	Time(min)
Based Model	44.30	20.56
Hidden-layer Model	49.37	28.60
Sigmoid Model	40.60	21.15
Relu Model	50.91	37.65
Weight-decay Model	41.25	22.89
Momentum Model	45.74	23.83
Dropout Model	36.58	24.57
Batch-norm Model	48.34	28.32
Mini-batch Model	28.37	17.24
Gelu Model	50.66	26.50

Table 1: Ablation Studies Performance Table.

Models	learning rate	Accuracy(in percentage)
Baseline Model	0.1	47.96
Baseline Model	0.01	49.8
Base+weight	0.1	47.26
Base+weight	0.01	50.2
Base+drop	0.1	45.77
Base+drop	0.01	39.68
Base+BN	0.1	47.39
Base+BN	0.01	45.01

Table 2: methods comparison Table.

Models	hidden layer structure	Accuracy(in percentage)
Baseline Model	(256,128)	47.96
light layer	(64,32)	49.86
deep layer	(128,256,128)	45.44

Table 3: layers comparison Table.

3.2 Hyperparameter analysis

As can be seen from Table2, the experiments was devided into two groups for hyperparameter tuning of the learning rate. One group uses a learning rate of 0.01, and the other group uses a learning rate of 0.1. It can be seen from table and figure that the learning rate of 0.1 performs better. When the learning rate is 0.1, dropout shows the effect of preventing overfitting. It can be clearly seen that the model using dropout and other models In contrast, overfitting is well suppressed. When the learning rate is 0.01, the model does not fully converge. Therefore, our final model will adopt a learning rate of 0.1 to better reflect the effect of dropout to prevent overfitting.

3.3 Ablation studies

In order to better study the impact of the methods discussed in the method on the deep learning network architecture, we designed an ablation study. In this ablation study, the learning rate, epoch, and the output layer are all invariants. The learning rate of all experiments is 0.0001, epoch for all experiments are 150, and the activation function of the output layer of all models is softmax. Based Model is a model for comparison, which includes 4 layers and uses tanh as the activation function of the hidden layer. The remaining experiments add an optimization method or regularization method to the Based Model architecture to compare the impact of this method on the entire architecture. The accuracy and computation time performance of each model is shown in Table 1, cross entropy loss curve for each model is shown in Appendix.

3.3.1 Hidden-layer Model

As can be shown from Table 1,After increasing the hidden layer from 2 to 4, the accuracy increased from 44.30 to 49.37, and time increased from 20.56 to 28.60 compared to the original architecture. Cross-entropy loss curve is shown in Figure 2.It can be clearly seen that the Cross-entropy Loss curve of the Hidden-layer Model is smoother and has less jitter than the Based Model. Under the premise of 150 epochs, it has a similar convergence rate to the Based Model. And the Hidden-layer Model has a higher accuracy than the Hidden-layer Model. In general, Hidden-layer Model has a better performance than the Based Model. This is because deeper hidden layers can have better fitting ability and can learn more refined models. However, the deepening of the hidden layer also leads to more computing time.

3.3.2 Sigmoid Model

As can be shown from Table 1,After changing all activation function in hidden layers from "tanh" to "logistic", the accuracy decreased from 44.30 to 40.60, and time increased from 20.56 to 21.15 compared to the original architecture. Cross-entropy loss curve is shown in Figure 3.The Sigmoid Model has a lower accuracy and compared with the Cross-entropy loss curve, the Sigmoid Model is smoother, and has a similar convergence speed to the Based Model under the premise of 150 epochs. In general, the Sigmoid Model does not perform as well as the Based Model. This is due to Based Model uses tanh as the activation function and outputs in range (-1, 1), while the Sigmoid Model uses logistic as the activation function and outputs the output of range (0, 1). After backward propagation, the gradient of the sigmoid activation function may be very close to 0, which may leads to suboptimal performance.

3.3.3 Relu Model

As can be shown from Table 1,After changing all activation function in hidden layers from "tanh" to "relu", the accuracy increased from 44.30 to 50.90, and time increased from 20.56 to 37.65 compared to the original architecture. Cross-entropy loss curve is shown in Figure 4.Compared with the Based Model, Relu Model has higher accuracy, and it can be seen from the Cross-entropy Loss curve that the Relu Model has a slightly faster convergence speed. Using Relu as the activation function generally gives

better performance. This is because the relu activation function converts all inputs to a positive interval, which has better generalization ability. And it can effectively avoid the problems of gradient explosion and gradient disappearance. But compared to Based Model, Relu Model has slower computation time

3.3.4 Weight-decay Model

As can be shown from Table 1,After applied Weight decay, the accuracy decreased from 44.30 to 41.25, and time increased from 20.56 to 22.89 compared to the original architecture. Cross-entropy loss curve is shown in Figure 5. Weight-decay is a kind of regularization, which essentially prevents the influence of model complexity on loss function. But our study, the model is simple, the over-fitting is not serious and complicated. Therefore, weight decay does not perform well in our model.

3.3.5 Momentum Model

As can be shown from Table 1,After applied Momentum in SGD, the accuracy increased from 44.30 to 45.74, and time increased from 20.56 to 23.83 compared to the original architecture. Cross-entropy loss curve is shown in Figure 6.Under the premise of 150 epochs, it can be seen that the Momentum Model has better accuracy and faster convergence speed than the Based Model. This may be because momentum takes into account the direction of the previous update to a certain extent when updating. Updates can be reduced for parameters that change direction. Therefore, Momentum Model is more stable, suppresses the oscillation, and speeds up the convergence speed to a certain extent.

3.3.6 Dropout Model

As can be shown from Table 1,After applied dropout regularization, the accuracy decreased from 44.30 to 36.59, and time increased from 20.56 to 24.57 compared to the original architecture. Cross-entropy loss curve is shown in Figure 7. Compared to the Based Model, Dropout Model is less accurate, and converges at a similar speed. The overall performance of the Dropout Model is not as good as the based Model. This is because the main role of dropout is to suppress overfitting, which requires fully convergence. The reason why dropout does not perform well in this model has a strong relationship with the number of epochs. If the number of epochs is increased, the usefulness of dropout will be more exerted.

3.3.7 Batch-Norm Model

As can be shown from Table 1,After applied Mini-batch SGD, the accuracy increase from 44.30 to 48.34, and time increased from 20.56 to 28.32 compared to the original architecture. Cross-entropy loss curve is shown in Figure 8.Overall, after applied Batch-normalisation, the Model perform better than the Based Model. Batch normalisation has the effect of accelerating convergence and suppressing overfitting.

3.3.8 Mini-batch Model

As can be shown from Table 1,After applied Mini-batch SGD, the accuracy decreased from 44.30 to 28.37, and time decreased from 20.56 to 17.24 compared to the original architecture. Cross-entropy loss curve is shown in Figure 9. Compared with the Based Model, Mini-batch Model has a very low accuracy, and can be seen from the Cross-entropy loss curve that the Mini-batch Model converges very slow. The reason for the poor performance of Mini-batch in this ablation study may be that the number of epochs is too small and the parameters of learning rate is too small. Because the number of updates per epoch of the Mini-batch Model is much smaller than that of the Based Model, it is difficult to produce good convergence speed and accuracy when the epoch and learning rate are the same with Based Model. In order to solve this problem, the number of epochs should be increased, and learning rate should be adjust to a larger size.

3.3.9 Gelu Model

As can be shown from Table 1,After changing all activation function in hidden layers from "tanh" to "gelu", the accuracy increases from 44.30 to 50.66, and time increased from 20.56 to 26.50 compared to the original architecture. Cross-entropy loss curve is shown in Figure 10.Gelu model declined rapidly in the first 20epochs, because the activation function Gelu was inspired by relu and dropout, and the idea of random regularity was introduced into the activation. Gelu decides whether to abandon or keep the current neurons by inputting its own probability distribution. It can be understood that the input value is multiplied by 1 or 0 according to its situation. A more "mathematical" description is that for each input X, it obeys the standard normal distribution N(0, 1), and it will be multiplied by a Bernoulli distribution. With the decrease of X, the probability of it being zeroed will increase.

3.4 Comparison methods

In the methods model comparison, we set up four groups of experiments. The four groups of experiments were compared with the learning rate of 0.1 and 0.01 respectively. In all models, epochs is set to 100, with one input layer, two hidden layers and one output layer. There are 128, 256, 128 and 10 nodes respectively. The reason for this setting is that Model such as drop out don't perform well in ablation study. This is because the number of nodes in 128, 64, 32 and 10 nodes designs in ablation study is small, which leads to insufficient learning. Therefore, we decided to adopt more network structures of node to make it fully learn. Theoretically, a deeper hidden layer may have a better effect, but if the data structure is relatively simple, it may lead to over-fitting. All the hidden layers of the model use relu activation function, and the output layers use softmax.

In our model design experiment, Baseline model is mini-batch added momentum. Base+weight stands for Baseline model added weight decay. Base+drop stands for Baseline model added dropout. Base+drop stands for Baseline model added batch normalisation.

As can be seen from the table2, under the same learning rate, adding weight decay to the Base Model cannot improve the accuracy of the Base Model. For the case of

adding dropout and Batch Normalization, it can be seen that it can effectively avoid the over-fitting of the model, and Batch Normalization can accelerate the loss decline.

In table3 layer comparison, the learning rate we set is 0.1. The hidden layer structure of our Baseline model is the first layer 256 and the second layer 128. In order to compare the experiment, we set up two control groups. One is the light layer, its hidden layers are 64 and 32 respectively, and the other is the deep layer, its hidden layers are 128, 256 and 128 respectively.

3.5 Jusitification on the best model.

As Can be seen from Figure 13, in which is the cross enttopy of our Final best model. Accuracy is 48.5, and time is 100 minutes. due to the disscusions in previous content, we decide to used a model with relu activation function in hidden layer and softmax in output layer. we used momentum to accelerate, use dropout to prevent from overfitting and use Batch-normalisation to accelerate converge and prevent overfitting. We did not apply weight decay because weight decay did not seem to improve any in prevent overfitting in our situation.

4 Discussion and conclusion

Through the above experiments, we compared the optimization effects of different optimization models on the MLP model. Including the layers' design comparison of the network, the comparison of the regularization method, the comparison of the learning rate and the comparison of the epoch, etc., the best model was found. In experiments, we learned the role of various parameters and the combined effect of optimization methods but there is still an overfitting problem.

In the process of completing the study, we learned the knowledge of deep learning together, deepened our understanding of the knowledge learned in the classroom, and gained a deeper understanding of the foundation of deep learning. In the future, since for our final model, the final model still appears to meet the problem of overfitting, which may be due to the fact that the learning rate parameter tunning is still not accurate enough, and for dropout, momentum, weight decay and mini batch, we did not adjust the parameters in more detail, in the follow-up In our work, we will focus more on hyperparameter tuning for each method.

Appendices

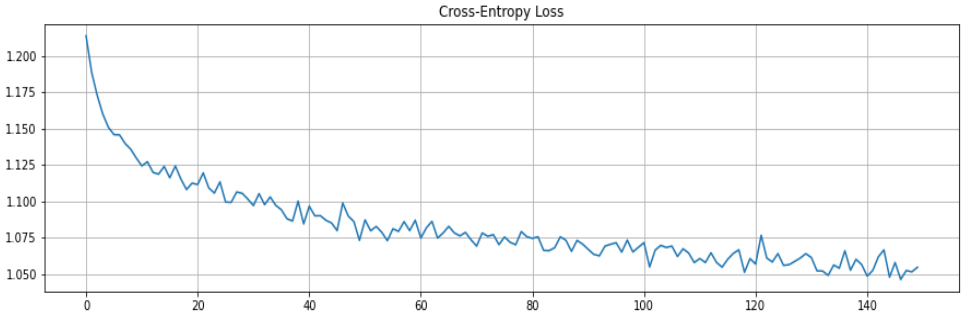

Figure 1: cross entropy loss of Based Model

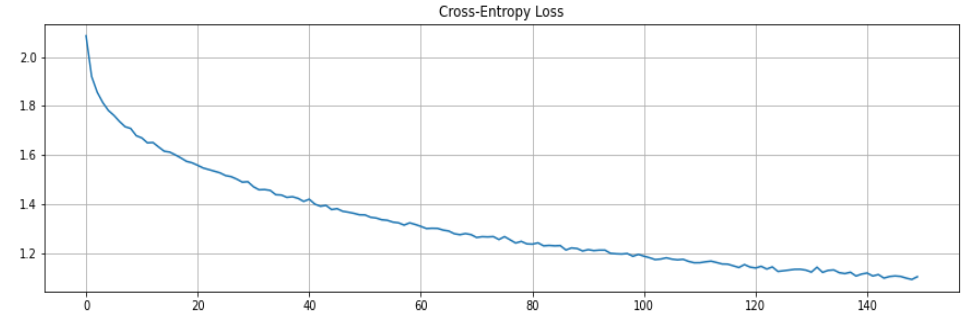

Figure 2: cross entropy loss of Hidden-layer Model

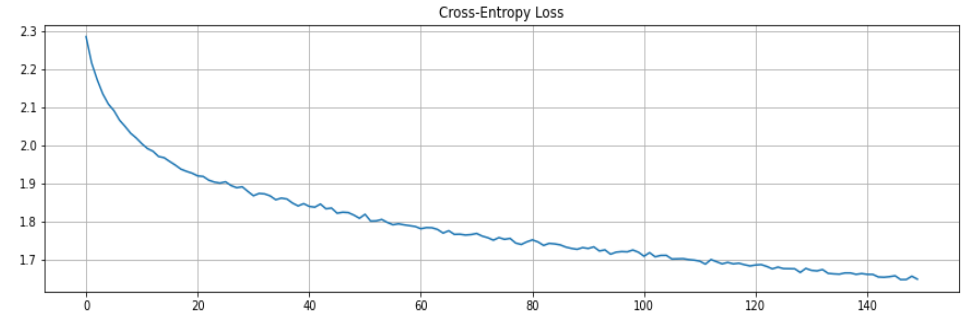

Figure 3: cross entropy loss of Sigmoid Model

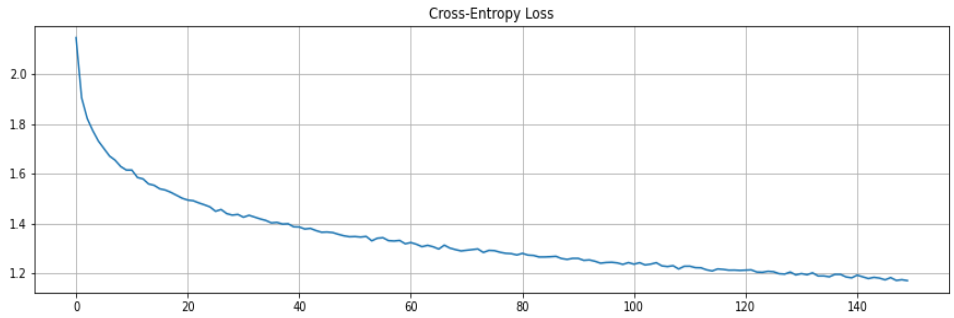

Figure 4: cross entropy loss of Relu Model

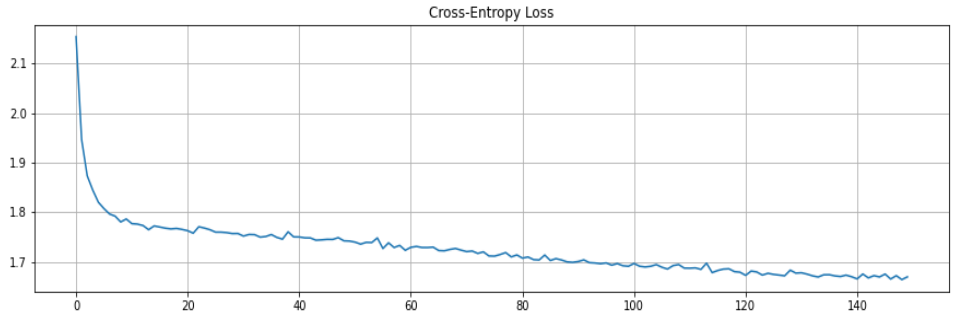

Figure 5: cross entropy loss of Weight-decay Model

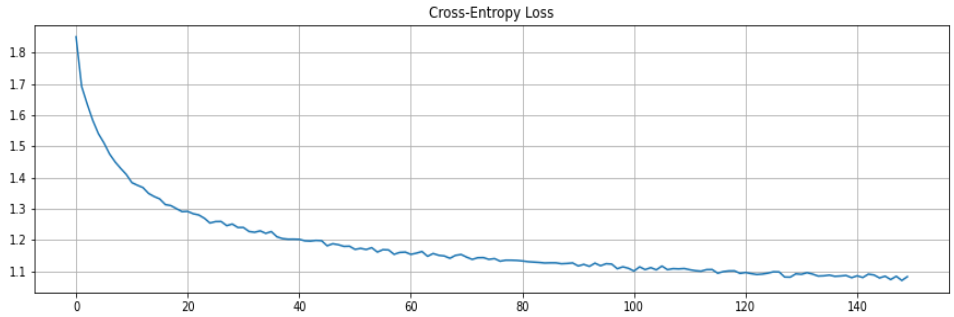

Figure 6: cross entropy loss of Momentum Model

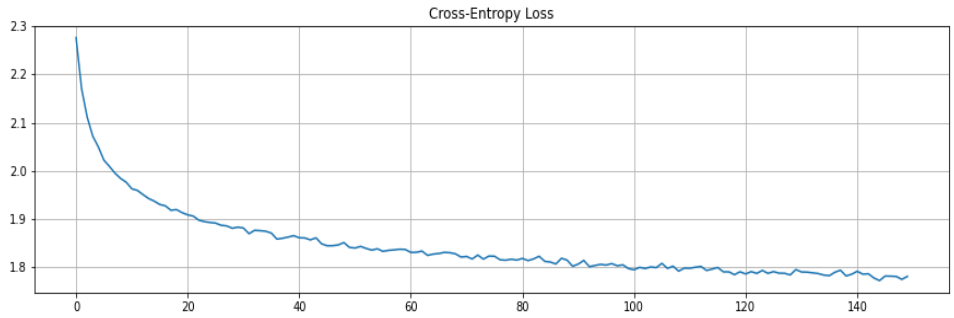

Figure 7: cross entropy loss of Dropout Model

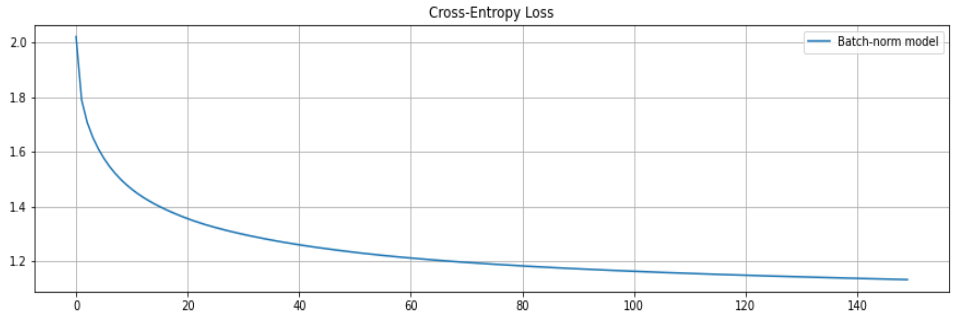

Figure 8: cross entropy loss of Batch-norm Model

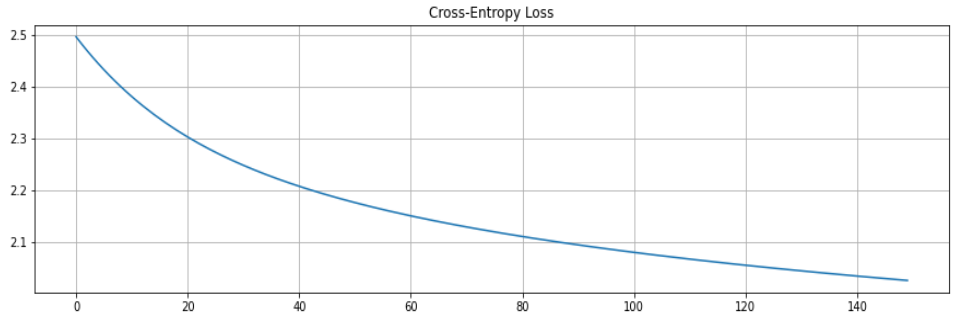

Figure 9: cross entropy loss of Mini-batch Model

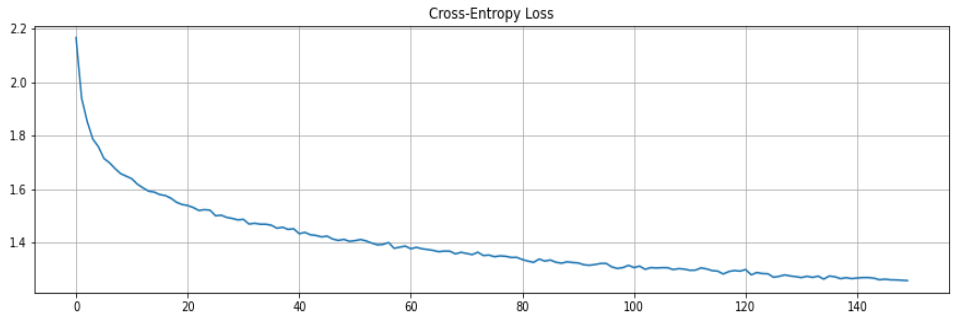

Figure 10: cross entropy loss of Gelu Model

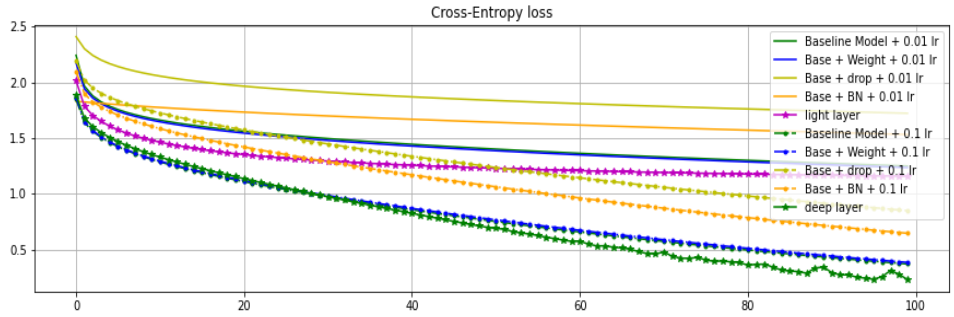

Figure 11: cross entropy loss of comparison methods

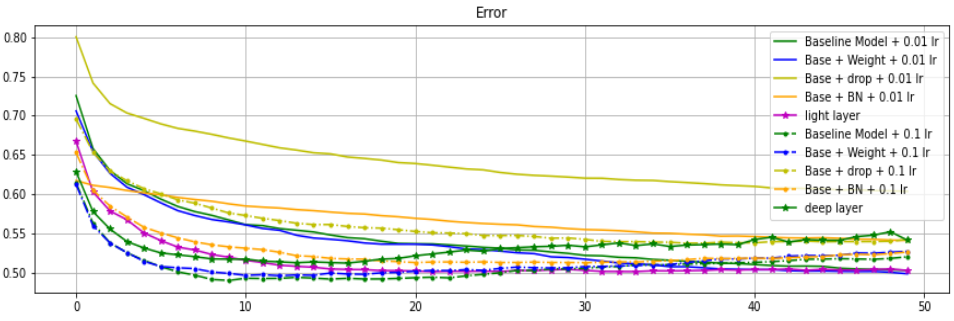

Figure 12: error of comparison methods

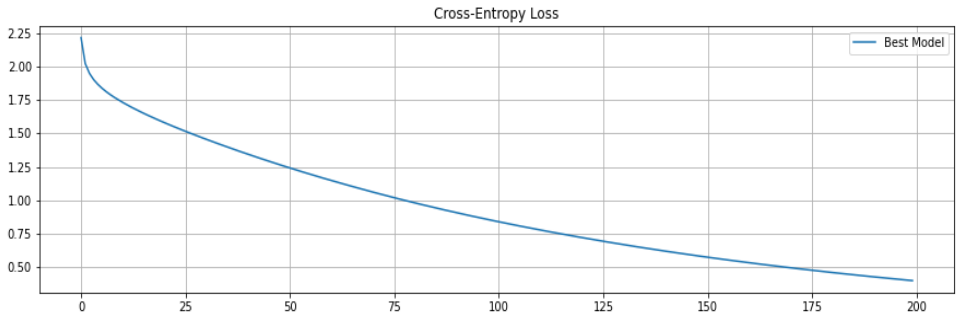

Figure 13: Cross entropy of best model

-software and hardware colab and Tesla K80