Experiment in Logistic and Softmax Regression to Classify Different Classes of Images

Dongze(Cody) Li
A16289926
dol005@ucsd.edu

Xiaoyan He
A16442577
x6he@ucsd.edu

Abstract

In this experiment we implement logistic regression and softmax regression to classify the 32 by 32 pixel images in CIFAR-10 dataset[3]. For logistic regression experiment, we used our logistic regression model to classify the images of airplane and dog as well as classify the images of cat and dog. For the first task, our model achieved the test accuracy of 71.2%. For the second task, the model make the prediction with test accuracy of 60%. More than that, we used softmax regression to classify all the images in the dataset. It reaches the test accuracy of 31.8% for our best performed model. Also, we found that the performance of the model is related to the similarity between the class of images. We also observed that the different combinations of hyperparameters have some impact of our model's performance.

1 Introduction

In this project, we developed some supervised machine learning method to help us to classify the 32 by 32 pixel images such as airplane, automobile, and cat in the CIFAR-10 dataset. To achieve this end, we write our own logistic to deal with binary classification and softmax regression to classify the images from multiple groups. All the data were convert into 1-dimension vector with each pixel treated as a feature, then pre-processed them by either min-max normalization or z-score normalization at the beginning. Also, for softmax regression, we transformed the label of each class to one-hot vector. In order to estimate our model's performance and help us to analyze the model efficiently, we applied 10-fold cross validation with the average validation loss and validation accuracy rate recorded into our training procedure. Moreover, to prevent the model from overfitting, we added an early stop function to stop training when the loss of validation set start to rise. More than that, we added a pipeline in our code to test the performance on different combinations of hyperparameters and return the best combination.

In logistic regression, we trained the model for the following two binary classification tasks: Airplane(class 0) vs. Dog(class 5) and Cat(class 3) vs. Dog(class 5). The final loss function and accuracy rate indicates that the logistic regression model have a better performance in Airplane vs. Dog than Cat vs. Dog with the hypothesis that airplane and dog has more difference in each feature.

Then for the classification among all the kinds of the images, we applied softmax regression. During this experiment, we found that the hyperparameters that achieved best result are different from those for logistic regression. Also, as we increased the number of groups that we trying to classify, the accuracy of our model's prediction decrease to the level of 31.8%.

Please notice that when tuning the hyperparameters, we turned on the early stop to find the hyperparameters with best performance. But when drawing the plot, we turned off the early stop to make the model train over 100 epochs for each cross validation fold. That makes the x-axis for each graph has the same scale, and we record the minimum validation loss point in situation that if we turn early stop on on the graph.

2 Related Work

2.1 Acknowledgement

We acknowledge the great help from our instructor Garrison Cottrell for briefly explained the concept of logistic regression, softmax regression and cross-entropy loss function [2]. We also thanks for the book "Dive into Deep Learning" [1] which gives a lot of insight on model construction. We also appreciate Krizhevsky for the dataset provided in his website [3][4].

2.2 Reference

[1] A. Zhang, Z.C. Lipton, M. Li, A.J. Smola. Dive into Deep Learning. *arXiv:2106.11342 [cs.LG], Jul 2022*

[2] G. Cottrell. *Lecture 2: Logistic Regression Multinomial Regression*, lecture notes, Department of Computer Science, University of California San Diego, Oct 2022.

[3] A. Krizhevsky. CIFAR-10 and CIFAR-100 Datasets, http://www.cs.toronto.edu/ kriz/cifar.html.

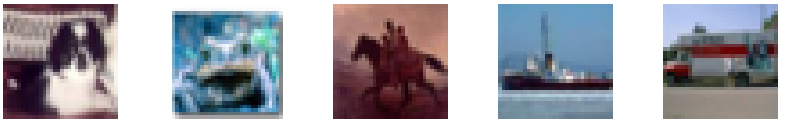
[4] A. Krizhevsky. Learning multiple layers of features from tiny images. Master's thesis, Department of Computer Science, University of Toronto, 2009.

3 Dataset

3.1 Data Description

The dataset we used in this programming assignment is CIFAR-10. It's the dataset that contains 60000 32 by 32 pixel color images in 10 different class with 6000 images for each class. Each class has a label from 0 to 9. The dataset is divided into five training sets and one test set. We randomly selected one image for each class and present them in the following figure 1.

(a) Class 0 (b) Class 1 (c) Class 2 (d) Class 3 (e) Class 4



(f) Class 5 (g) Class 6 (h) Class 7 (i) Class 8 (j) Class 9

Figure 1: Random selected sample from each class

3.2 Data Statistics

The test set contains 1000 images for each class which are randomly selected. The remaining images are randomly distributed into five training sets, and some training sets may contain more images from one class than another. Table 1 below demonstrates the size of train and test data for the different classes

Table 1: Data statistics

Class	Training Set 1	Training Set 2	Training Set 3	Training Set 4	Training Set 5
0	1005	984	994	1003	1014
1	974	1007	1042	963	1014
2	1032	1010	965	1041	952
3	1016	995	997	976	1016
4	999	1010	990	1004	997
5	937	988	1029	1021	1025
6	1030	1008	978	1004	980
7	1001	1026	1015	981	977
8	1025	987	961	1024	1003
9	981	985	1029	983	1022

3.3 Data Pre-processing

Before starting training our model, we firstly convert the images to vectors with the size 1024 by 1, since each image is 32 by 32 and we treated each pixel as a feature. For both logistic and softmax regression, we normalize each feature of the sample in two different ways, z-score normalization and min-max normalization. For z-score normalization, we subtract each sample by the mean of the dataset and then divide it by the standard deviation, as the equation (1) showed below.

$$f(x) = \frac{x - \mu}{\sigma} \tag{1}$$

For min-max-normalize, we subtract the sample by the minimum of the data then divide it by the difference between the maximum and minimum of the data, as the equation (2) showed below.

$$f(x) = \frac{x - min(x)}{max(x) - min(x)} \tag{2}$$

Moreover, for softmax regression, we transform the label for each class to one-hot vector that the kth element of the vector is 1 and 0 elsewhere, for converting kth class's label. In this experiment, we have 10 class in total, thus we will have a 10 by 1 vector with 1 in the index which represents the class's label and 0 else.

4 Logistic Regression

4.1 Description of the Model

Algorithm 1 Stochastic gradient descent with early stop

$w \leftarrow 0$ ▷ initialize weight
$idx \leftarrow 0$ ▷ initialize early stop index
for i in range of M **do** ▷ M is the number of epochs to train
  shuffle the dataset
  for j in 0 to N, in steps of B **do** ▷ N is number of example, B is batch size
    start = j
    end = $j + B$
    if model is logistic **then**
      choose sigmoid function as activation function $g(w)$
      $w \leftarrow w - learningrate * \sum_{n=start}^{end} \nabla E^n(g(w))$
    else if model is softmax **then**
      choose softmax function as activation function $g(w)$
      $w \leftarrow w - learningrate * \sum_{n=start}^{end} \nabla E^n(g(w))$
    end if
  end for
  test and record train and validation loss and accuracy
  if validation loss less than the best loss **then**
    $idx = i$
    $w_{best} = w$ ▷ Update the best weight
    Update best loss
  else if $i - idx > 10$ **then** ▷ condition for early stop
    break the loop ▷ early stop
  end if
end for
  return recorded loss and accuracy, best loss and accuracy, w_{best}

For first half of our project, we constructed logistic regression to do binary classification. It takes a d-dimensional vector $x \in \mathbb{R}^d$, where d is the number of features and gives an output either 1 or 0 to indicate if the model classify the input as the target class.

The activation function for logistic regression is sigmoid function follows the formula, $f(x) = 1/(1 - e^{-x})$. In our model, the input of sigmoid function is a scalar produced by the dot product of some weight vector $w \in \mathbb{R}^{d+1}$ where w_0 represent the bias and the sample vector x that x has been appended a leading 1 to present the bias term, and it returns value y between 0 and 1 that indicates the conditional probability the sample is in the target class given the sample x.

$$y = P(\mathcal{C}_1|x) = \frac{1}{1 + exp(-w^{\mathrm{T}}x)} = g(-w^{\mathrm{T}}x) \tag{3}$$

$$P(\mathcal{C}_0|x) = 1 - P(\mathcal{C}_1|x) = 1 - y \tag{4}$$

Since x in activation function is given, then g is a function about w. We use binary cross-entropy as our loss function which uses w as parameter.

$$E(w) = -\sum_{n=1}^{N} \{t^n \ln(y^n) + (1 - t^n) \ln(1 - y^n)\} \tag{5}$$

Then we minimize the loss function by updating w in each epoch through gradient descent.

$$w_{j,new} = w_{j,old} - \alpha \frac{\partial E(w)}{\partial w_j} = w_{j,old} + \alpha \sum_{n=1}^{N} (t^n - y^n) x_j^n \tag{6}$$

where α is the learning rate. Here, we applied 10-fold cross validation in our train procedure. In order to find the hyperparameters that gives the best validation loss, we recorded average training loss and validation loss and their accuracy as well as setting up early stop to prevent the model from overfitting. The early stop function

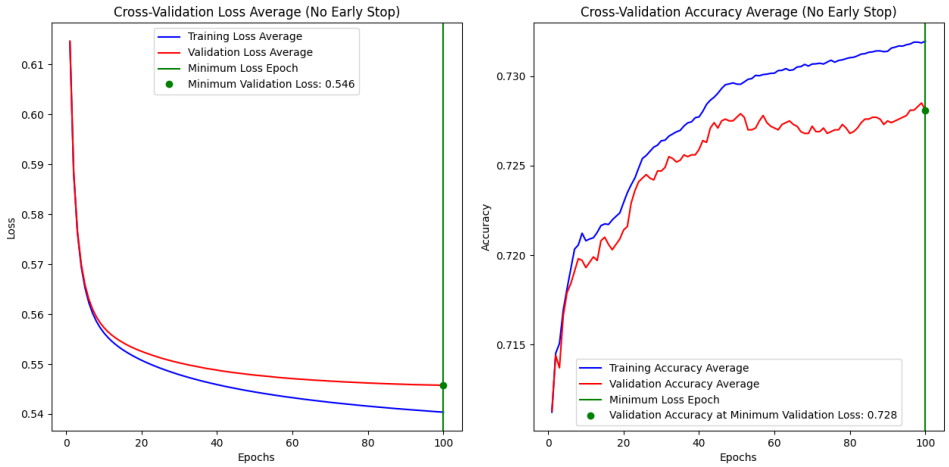
we setted up will stop the model from training if after reaching a local minimum in validation loss, the loss doesn't reach a new minimum in next 10 epochs.

In both the experiment of logistic regression and the softmax regression, we used loss to judge the performance of model rather than accuracy. Because the value activation function gives as out is a conditional probability of the sample belongs to a class. But the accuracy just shows if the model makes the correct classification. For example, one model mistakenly predict a class 1 image has 0.8 probability belongs to class 2, and another model predict a class 1 image has 0.52 probability belongs to class 2. Both model makes wrong prediction, but performance of each model are different, we can see the second model just make a slightly error. That what accuracy cannot tell, but loss can.

Beyond that, we also pick stochastic gradient descent as the method to train our weight vector. In each epoch, we generated a randomly permutation of indices of training set to shuffle the data then let our model go over and train through the shuffled training set in step of B, where B is the size of minibatch. Finally, we used the weight with best performance on validation set to test and report the averaged loss and accuracy. The pseudocode of stochastic gradient descent with early stop is shown as algorithm 1.

4.2 Results

4.2.1 Airplane (Class 0) vs. Dog (Class 5)



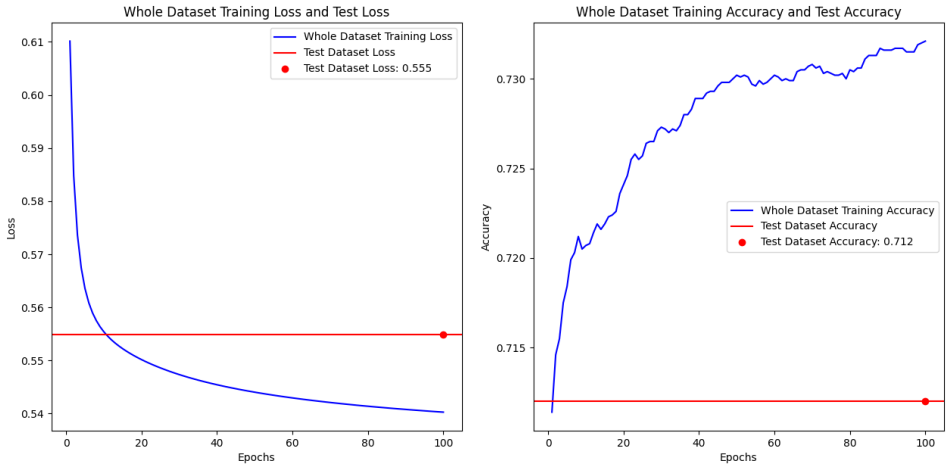
(a) Training and validation loss

(b) Training and validation accuracy

Figure 2: Training performance with learning rate as 1×10^{-6}, batch size as 1, z-score norm for class 0 vs. class 5

For the task of classify the images of airplane and dog, we achieved the best performance of model with learning rate as 1×10^{-6}, z-score normalization and batch size as 1. The averaged validation loss is 0.546 and the validation accuracy is 72.8%(See figure 2) with the test loss and accuracy as 0.555 and 71.2%(See figure 3).

As Figure 2 shows, both training and validation loss drop rapidly during the first 20 epochs and then begin to slow down. Beyond our expectation, the curve of validation loss does have a trend to go up which indicates the model may begin to overfit. Also, the validation loss curve looks smooth without any fluctuation in this case. With trying other combination of hyperparameters(we will show them in the next subsection), we tried to make a conclusion that the smaller learning rate and batch size help to reduce the fluctuation during the training process. Although there are some small fluctuations, the curves of training and validation accuracy generally show an upward trend, and we observed that validation accuracy increases slower and has more fluctuations.



(a) Test loss

(b) Test accuracy

Figure 3: Test performance with learning rate as 1×10^{-6}, batch size as 1, z-score norm for class 0 vs. class 5

The final test result indicates that our model has a loss of 0.555 and accuracy of 71.2% on test set. Figure 4 is the visualization of the weight vector. Here, we reshape the 1024 by 1 vector into a 32 by 32 matrix to visualize it. We can see that the model puts the most positive weight in the middle part of the image and some weight on top and two bottom corners as well. Also, it shows the model put a lot of negative weight on left and right top corner. We think one possible reason for such weight distribution is that for the images in class 0, the airplane often occur in the middle of part of the image, thus a lot of features here are different from those images of dog. Thus, these features are most helpful for the model to identify if the image is belong to the class of airplane.

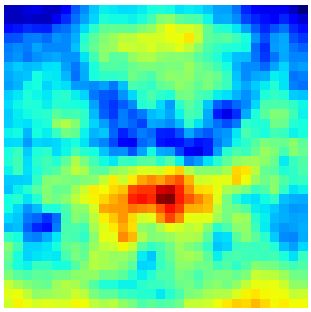
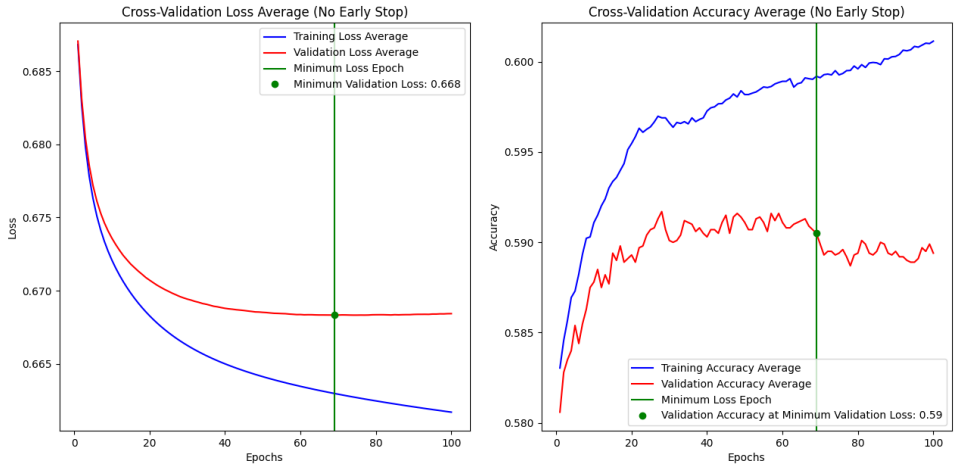


Figure 4: Final weight vector visualization for class 0 vs. class 5

4.2.2 Cat (Class 3) vs. Dog (Class 5)

We used the same hyperparameters as above to achieve the best performance. It gives the result that validation loss is 0.668 and validation accuracy is 59%(See figure 5). Its performance on test set is 0.667 as loss and test accuracy is around 60%(See figure 6). One difference between this experiment and the last experiment is that the validation loss triggered early stop and we observed that unlike class 0 vs. class 5 which validation loss keep going down, the validation loss here seems to converge around 0.66. Also, the validation accuracy curve has more fluctuations than that in class 0 vs. class 5.

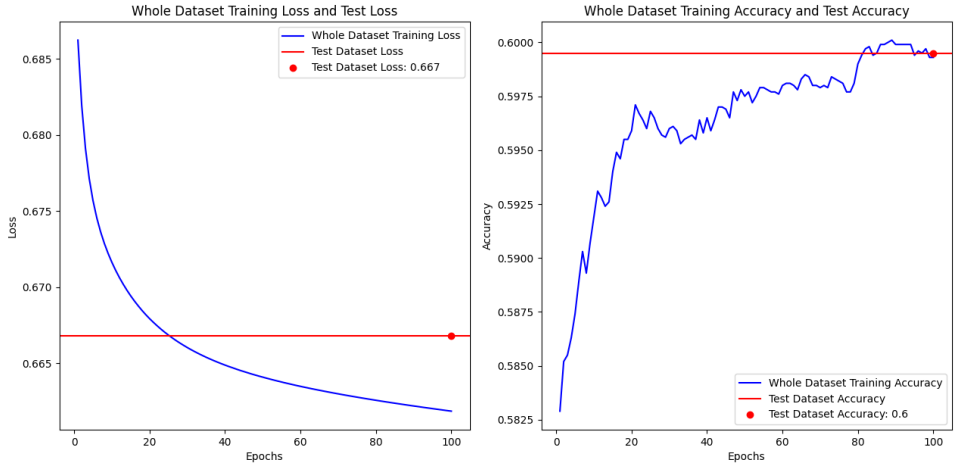
One possible reason for such result is that classifying cat and dog are much harder than classifying airplane and dog. There are more similar features between cat and dog even people cannot easily distinguish them if the picture is not clear enough, thus the result here has a high loss and lower accuracy. Moreover, from the plots in figure 5, we can see if we don't add an early stop, the model will begin to overfit which will lead to a poor performance in prediction and classification.



(a) Training and validation loss

(b) Training and validation accuracy

Figure 5: Training performance with learning rate as 1×10^{-6}, batch size as 1, z-score norm for class 3 vs. class 5



(a) Test loss

(b) Test accuracy

Figure 6: Test performance with learning rate as 1×10^{-6}, batch size as 1, z-score norm for class 3 vs. class 5

As figure 7 showed in below, compare to the figure 4 which is the visualization of weight in classification between airplane and dog, it has the red part in the top middle. One hypothesis about this difference between two weight vector is that there are significant different features between cat's and dog's head and the top middle part might be the a frequent place that cat's and dog's head appear in the image in this dataset.

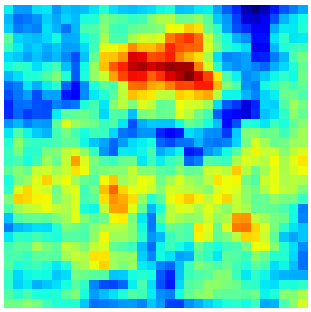
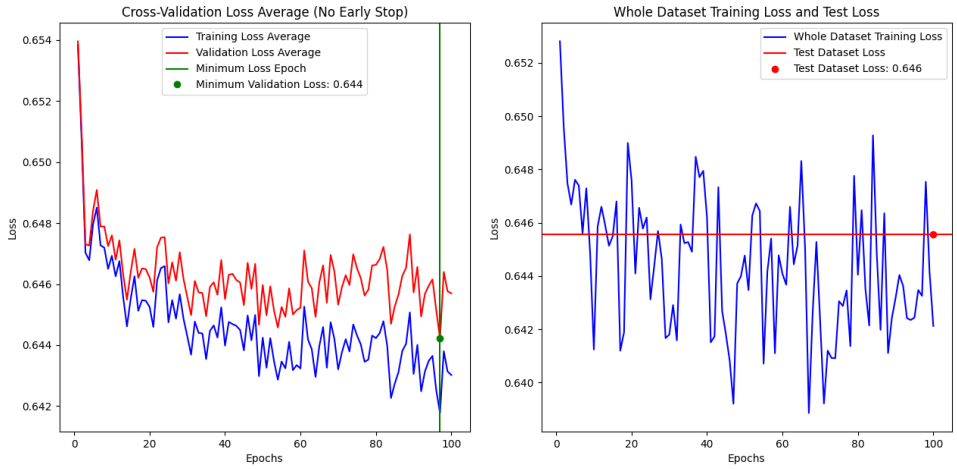


Figure 7: Final weight vector visualization for class 3 vs. class 5

4.3 Discussions

4.3.1 Airplane (Class 0) vs. Dog (Class 5)

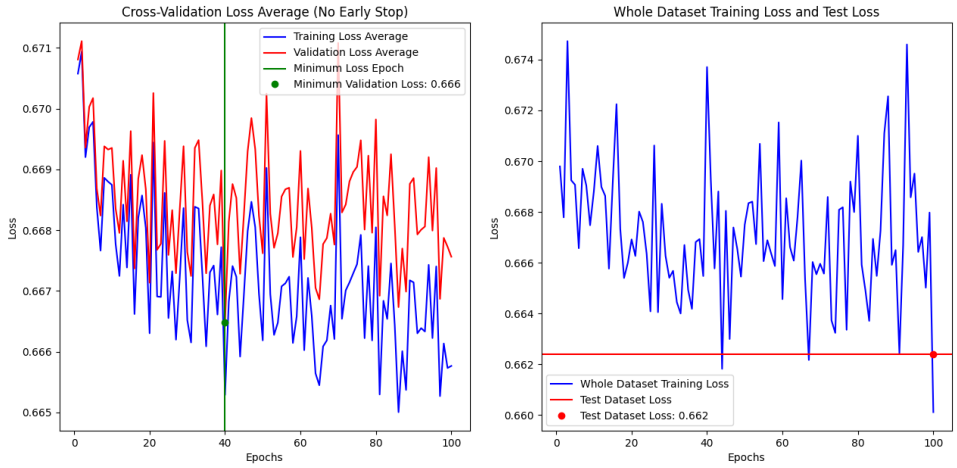


(a) Training and validation loss with batch size 5

(b) Test loss with batch size 5

Figure 8: Training and test performance with learning rate as 1×10^{-6}, batch size as 5, z-score norm for class 0 vs. class 5

For logistic regression, we also tried some other combinations of hyperparameters. First, we considered the batch size of 5 (See figure 8) and 10 (See figure9) and hold every thing else constant. We found that as the increasing in batch size, it takes less time to train over 100 epochs. Then, we found that although across the whole training process the model didn't overfit, the loss and prediction accuracy became unstable as we can see more fluctuation in the plots. We can see that the as the batch size increase, there are more fluctuation by comparing figure 8 and figure 9.

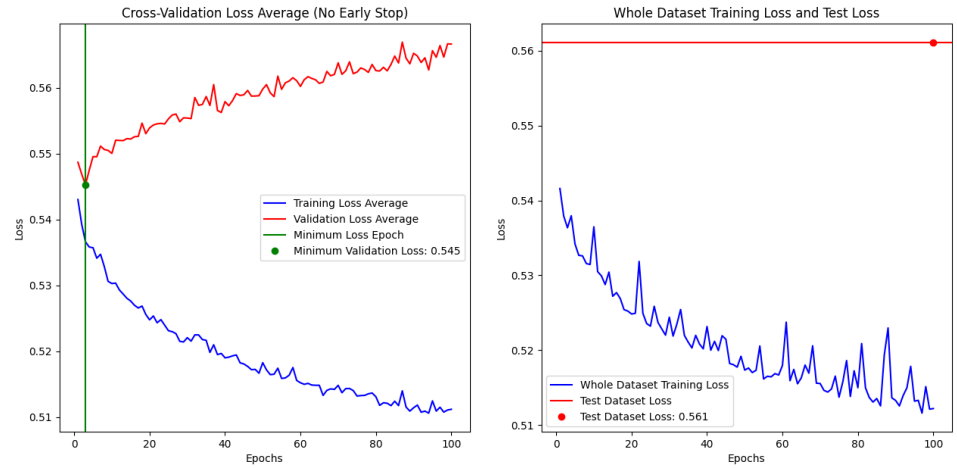


(a) Training and validation loss with batch size 10

(b) Test loss with batch size 10

Figure 9: Training and test performance with learning rate as 1×10^{-6}, batch size as 10, z-score norm for class 0 vs. class 5

Also, increasing in the batch size gives a worse performance in validation loss and test loss. Unlike the validation loss curve of the best performed hyperparameters that decreases significantly until it reaches around 0.55, the validation loss curve for batch size of 5 and 10 don't show such trend. In figure 8 (a), we observed that there is a significant drop in validation at the beginning and then oscillates around the level of 0.646. And in figure 9 (a), it's hard to observe a big drop in loss and it has more greater fluctuations. From these observation, we think the loss and fluctuations will increase as we increase the batch size.



(a) Training and validation loss with learning rate 1×10^{-4}

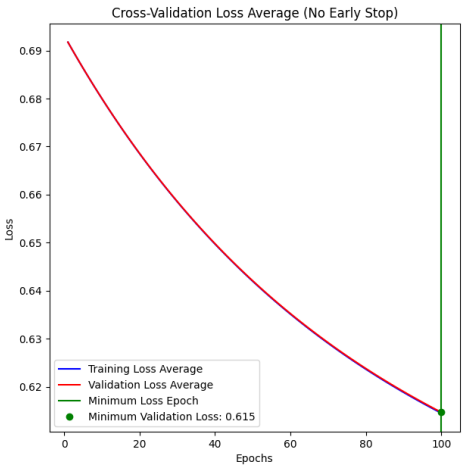
(b) Test loss with learning rate 1×10^{-4}

Figure 10: Training and test performance with learning rate as 1×10^{-4}, batch size as 1, z-score norm for class 0 vs. class 5

The next combination of hyperparameters we tried is learning rate of 1×10^{-4} and holding others as the same as those give the best performance. It gives the similar validation and test performance as the best

hyperparameters. But the major difference between this combination and the best hyperparameters is that it triggered the early stop before 20 epochs.

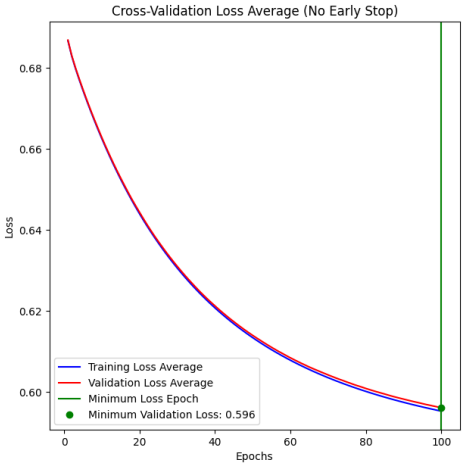
As figure 10 (a) shows, after few epochs the training loss curve keeps decreasing and the validation loss curve begins to increase in contrast which means if we don't stop the model it will began to overfit. We can also see the curve oscillates at some points. If we want to approach the minimum point as close as we can, we should avoid to set a too large learning rate since it make the model stop training just after a few epochs and we can't make the model become more precise in prediction.



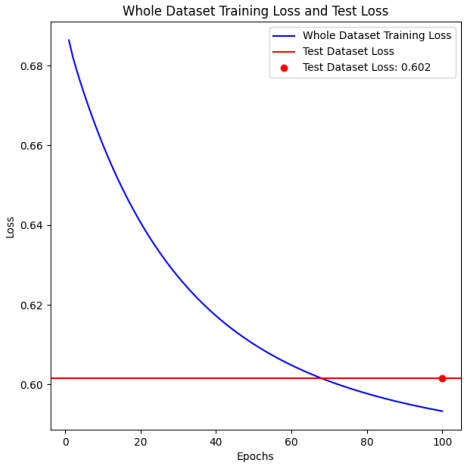
(a) Training and validation loss with learning rate 1×10^{-8}

(b) Test loss with learning rate 1×10^{-8}

Figure 11: Training and test performance with learning rate as 1×10^{-8}, batch size as 1, z-score norm for class 0 vs. class 5



(a) Training and validation loss with min-max normalization

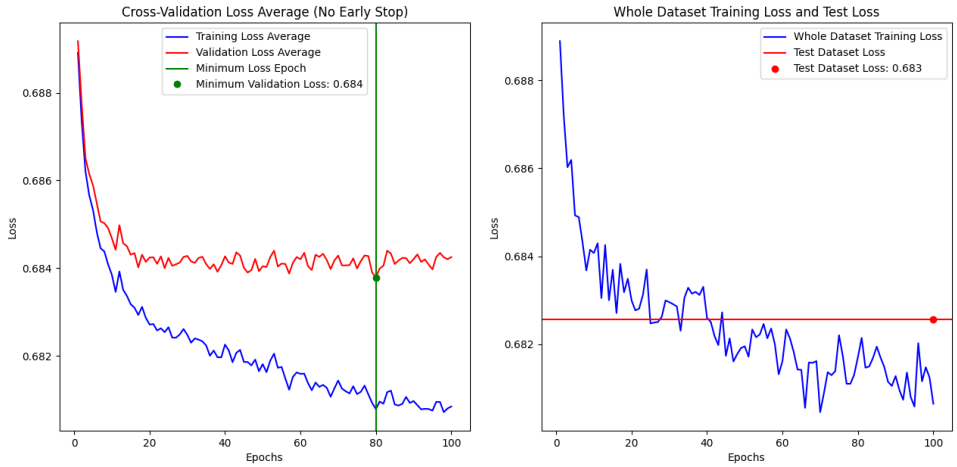


(b) Test loss with min-max normalization

Figure 12: Training and test performance with learning rate as 1×10^{-6}, batch size as 1, min-max norm for class 0 vs. class 5

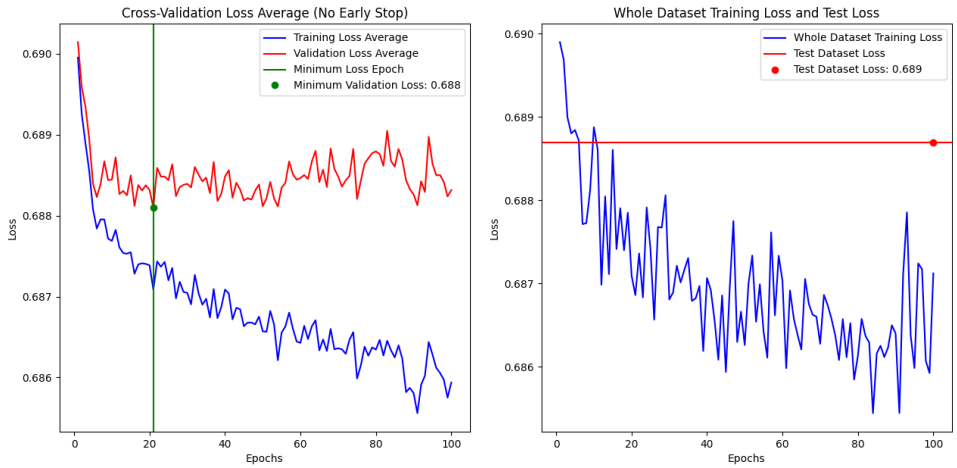
Figure 11 and figure 12 show very similar patter. Figure 11 shows the training and test loss with learning rate 1×10^{-8} and holds other as same as the best performed hyperparameters. Figure 12 is about when we change the normalization from z-score to min-max the performance of the model. In both figures, we observed that the training loss and validation loss curve almost overlap everywhere. Also both models give the similar result of validation and test loss which is around 0.6. It indicates that over 100 epochs the model still doesn't converge. For the learning rate of 1×10^{-8}, it tells that too small learning rate will cause the problem of not converging to the global minimum. As well, figure 12 implies that holding all else same, z-score normalization has a larger convergence rate than min-max normalization.

4.3.2 Cat (Class 3) vs. Dog (Class 5)



(a) Training and validation loss with batch size 5 (b) Test loss with batch size 5

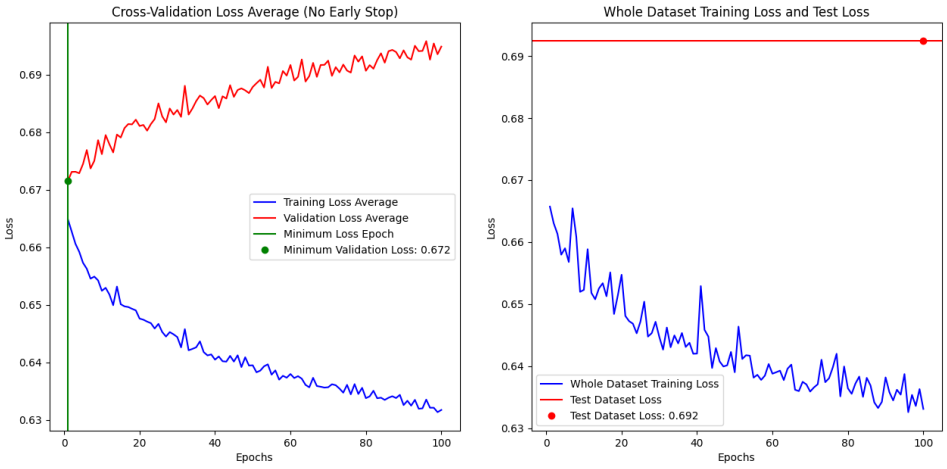
Figure 13: Training and test performance with learning rate as 1×10^{-6}, batch size as 5, z-score norm for class 3 vs. class 5



(a) Training and validation loss with batch size 10 (b) Test loss with batch size 10

Figure 14: Training and test performance with learning rate as 1×10^{-6}, batch size as 10, z-score norm for class 3 vs. class 5

For the experiment of classification of cat and dog, we did the same thing as the classification of airplane and dog. We have the similar finds as the what we states before, as figure 13 and 14. One major difference is that the fluctuations here are much smaller for both batch size of 5 and batch size of 10. Also the loss for both batch sizes are closer to the best value compare to those of the classification of the airplane and dog. One hypothesis for such performance is that it's harder for the model to classify between cat and dog even the optimized model can only classify with the accuracy of 60%. Thus, given any sample of cat or dog after dot product with the weight vector, the sigmoid function might more likely return the value around 0.5 which mean our model are not sure that the image is dog or cat, therefore the loss will also in the range around $\ln(0.5)$ which is around 0.68. So the curve here has less fluctuation. In contrast, when training the logistic regression to classify the airplane and dog, sigmoid function is more likely to return more extreme value, therefore causes more fluctuations in loss curve.

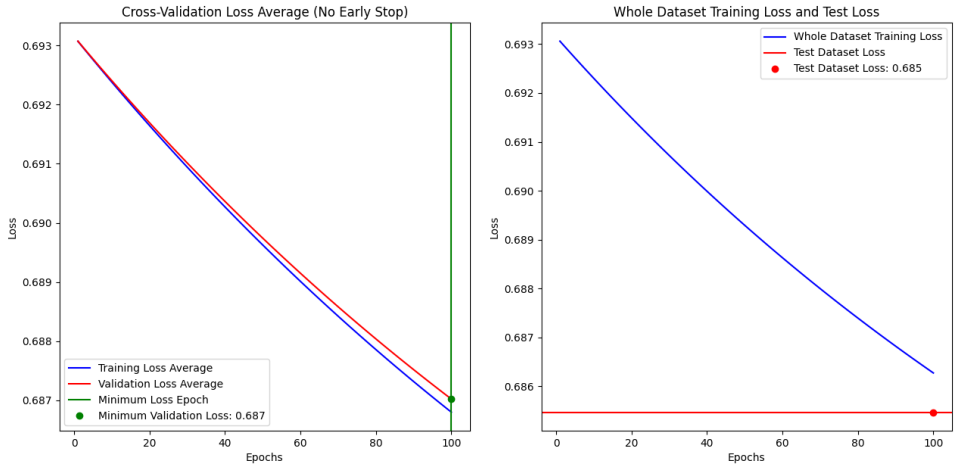


(a) Training and validation loss with learning rate 1×10^{-4}

(b) Test loss with learning rate 1×10^{-4}

Figure 15: Training and test performance with learning rate as 1×10^{-4}, batch size as 1, z-score norm for class 3 vs. class 5

Figure 15 shows the similar patterns as figure 10 as we increase the learning rate to 1×10^{-4}. When we increase the learning rate, it's more likely for the model to begin overfit. But the minimum validation loss before early stop is still close to the best value.

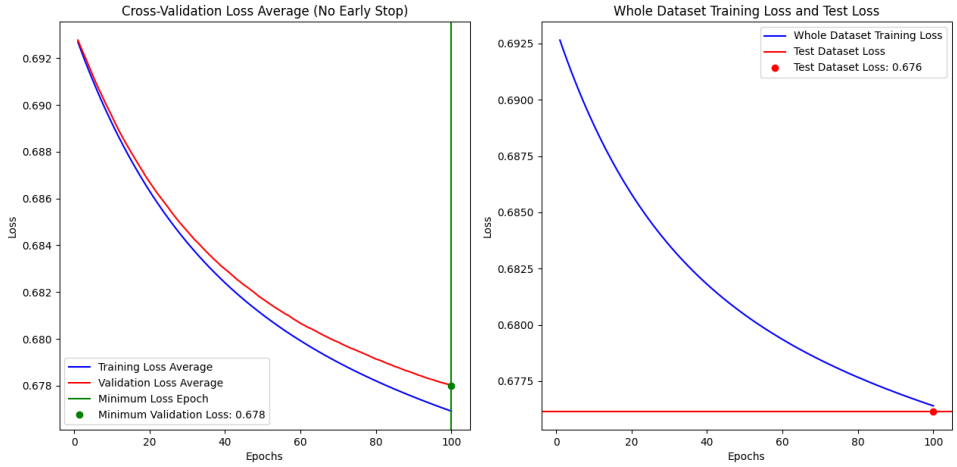


(a) Training and validation loss with learning rate 1×10^{-8}

(b) Test loss with learning rate 1×10^{-8}

Figure 16: Training and test performance with learning rate as 1×10^{-8}, batch size as 1, z-score norm for class 3 vs. class 5

And figure 16 and figure 17 also give the similar result. Both decreasing the learning rate too much or changing to min-max normalization caused the model didn't converge to the global minimization. One thing slightly different from those plot in classification of airplane and dog is that the training loss and validation loss do not overlap everywhere.



(a) Training and validation loss with min-max normalization

(b) Test loss with min-max normalization

Figure 17: Training and test performance with learning rate as 1×10^{-6}, batch size as 1, min-max norm for class 3 vs. class 5

5 Softmax Regression

5.1 Description of the Model

Another part of our project is implementing softmax regression to classify all the data in the set. It's the generalization of logistic regression for multiple classes which is 10 in our case. Given an input vector $x \in \mathbb{R}^d$, softmax regression model will return 10-dimensional vector with k^{th} element represents the conditional probability that sample x is belong to the class k given that sample x.

The activation function is shown as equation (7) below:

$$y_k^n = \frac{exp(w_k^{\mathrm{T}} x^n)}{\sum_{k'} exp(w_k^{\mathrm{T}} x^n)} \tag{7}$$

where y_k^n is the k^{th} element of vector y^n and represents the probability that sample x^n belong to class k. The model makes prediction based on this probability, that is classifying the sample into the group that has the highest probability it belongs to.

The cross entropy function is given as equation (8):

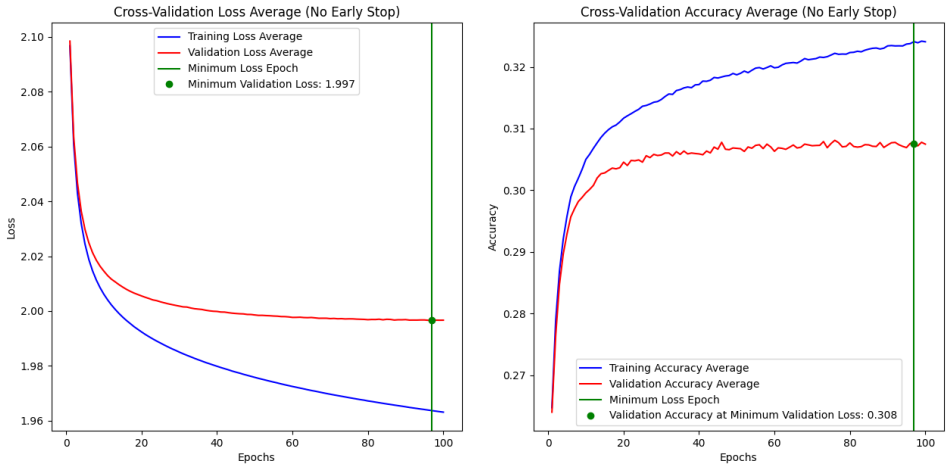
$$E = -\sum_n \sum_{k=1}^{c} t_k^n \ln y_k^n \tag{8}$$

where t_k^n is the k^{th} element of one-hot encoding vector of target label of sample x^n. Similar to the work we did in logistic regression, we used 10-fold cross validation with early stop and stochastic gradient descent(see algorithm 1) to train our weight matrix which are as same as what we used in logistic regression. The equation for gradient descent is given as equation (9):

$$w_{(j,k),new} = w_{(j,k),old} - \alpha \sum_{n=1}^{N} \frac{\partial E^n(w)}{\partial w_{j,k}} = w_{(j,k),old} + \alpha \sum_{n=1}^{N} (t_k^n - y_k^n) x_j^n \tag{9}$$

We also used the best performed weight matrix in cross validation to report the loss and accuracy on test set.

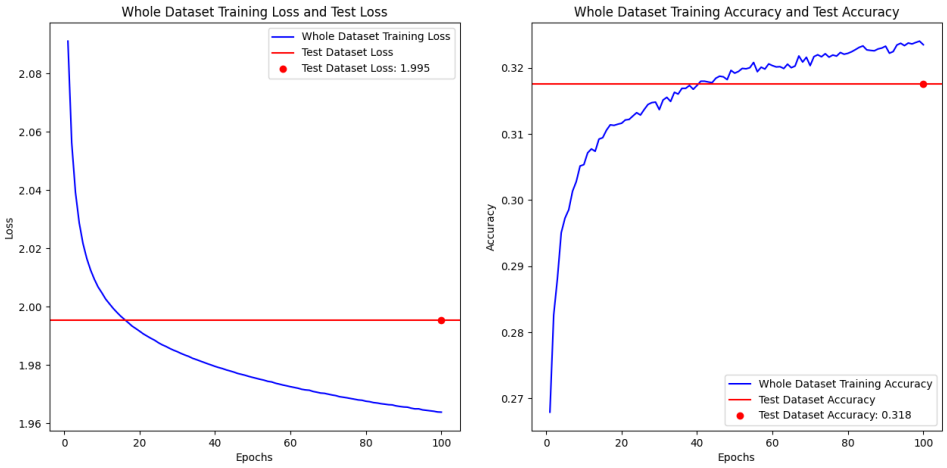
5.2 Results



(a) Training and validation loss

(b) Training and validation accuracy

Figure 18: Training performance with learning rate as 3×10^{-6}, batch size as 15, z-score norm

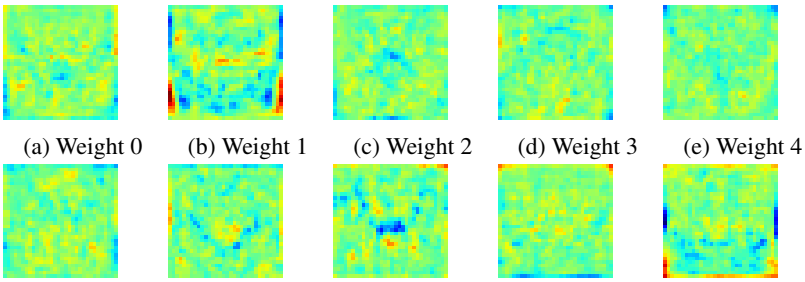


(a) Test loss

(b) Test accuracy

Figure 19: Test performance with learning rate as 3×10^{-6}, batch size as 15, z-score norm

By using the best performed hyperparameters, it achieves test loss of 1.995 and test accuracy as 31.8% with validation loss as 1.997. From figure 18, we can see that the validation loss curve has a trend to converge around 2, and doesn't show a trend of overfit through 100 epochs. Both training and validation loss curve decrease fast in the first 20 epochs, and then while the training loss curve keeps decreasing the validation loss curve stays around 2. The accuracy curves look like the flip of the loss curve. The training accuracy keeps increasing, while validation accuracy converges around 0.308. The test performance in accuracy reaches our expectation which is over 30%.



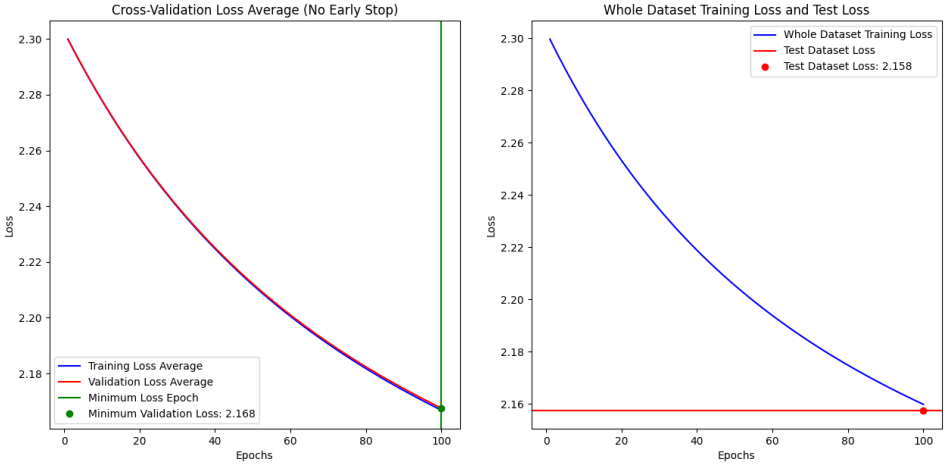
(a) Weight 0 (b) Weight 1 (c) Weight 2 (d) Weight 3 (e) Weight 4

(f) Weight 5 (g) Weight 6 (h) Weight 7 (i) Weight 8 (j) Weight 9

Figure 20: Weight matrix visualization

The weight matrix in softmax regression is 10 by 1024, each column of the weight matrix dot products with the sample vector to give the conditional probability of such sample belongs to the class through softmax function. To visualize the weight matrix, we reshape it into 10 32 by 32 matrix and shows as figure 20. We found that weight 1 and weight 9 has some similar pattern. Note that weight 1 is the second column of weight matrix w_2 since we index the classes and weight vectors that represents them from 0. Weight 1 is used to give the probability of sample belongs to class 1 given such sample, and similar for weight 9. We found that class 1 is the image of automobile and class 9 is truck, they share more common features than other classes, therefore the visualizations of those two columns have more similar pattern.

5.3 Discussions

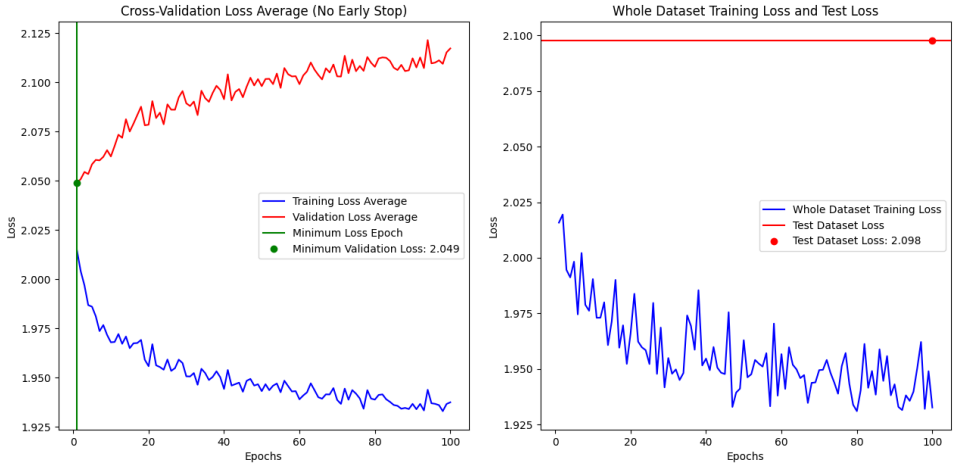


(a) Training and validation loss with learning rate 1×10^{-8}

(b) Test loss with learning rate 1×10^{-8}

Figure 21: Training and test performance with learning rate as 1×10^{-8}, batch size as 15, z-score norm

We also tried learning rate of 1×10^{-8}. As figure 21 shows, we have the same problem as what we showed in the part of logistic regression. The curve shows that our softmax model didn't converge yet and we have a test loss around 2.158 which is higher than our optimized value to indicate that the model didn't reach the minimum.

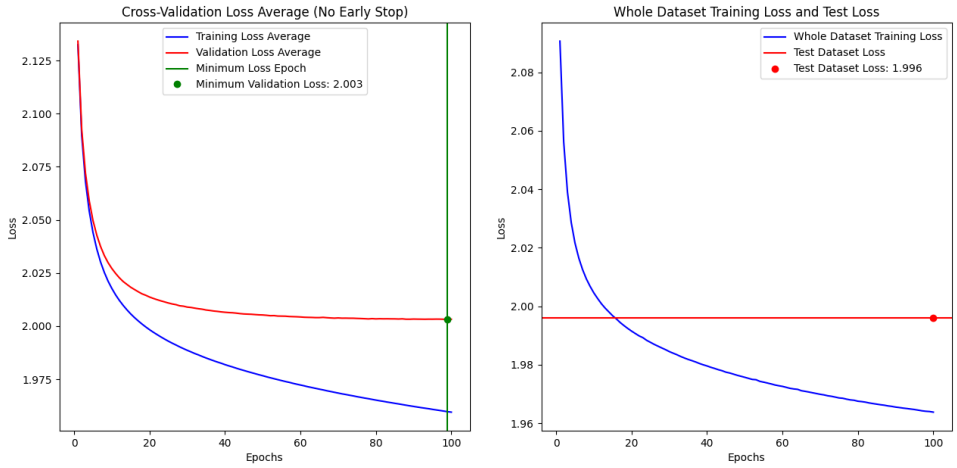


(a) Training and validation loss with learning rate 1×10^{-4}

(b) Test loss with learning rate 1×10^{-4}

Figure 22: Training and test performance with learning rate as 1×10^{-4}, batch size as 15, z-score norm

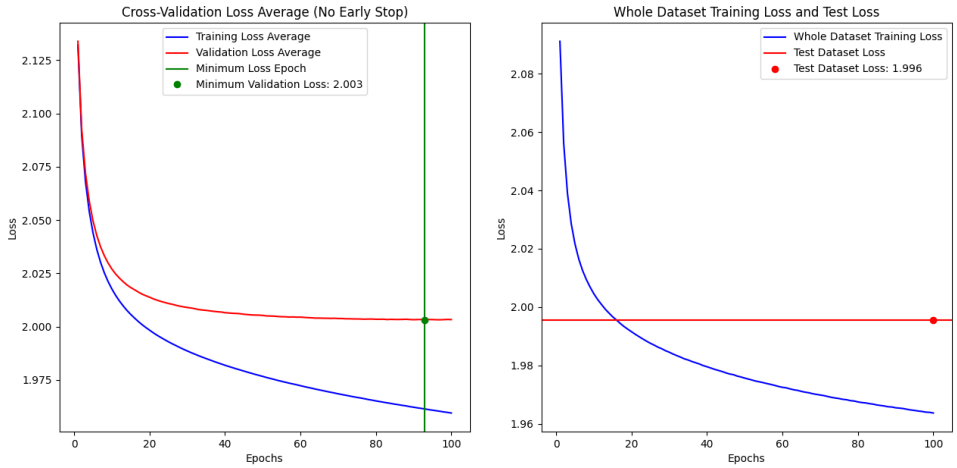
Figure 22 demonstrates the same problem as logistic regression when we increase the learning rate to a relative large value in this case is 1×10^{-4}. We can see that the model starts to overfit after the first epoch. The plot test performance shows the problem even clearer as the model performed well on training set and perform badly on the test set. Therefore, we think although we have early stop, we should still avoid use too large learning rate to make the model trained over enough epochs to increase the precision.



(a) Training and validation loss with batch size 1

(b) Test loss with batch size 1

Figure 23: Training and test performance with learning rate as 3×10^{-6}, batch size as 1, z-score norm



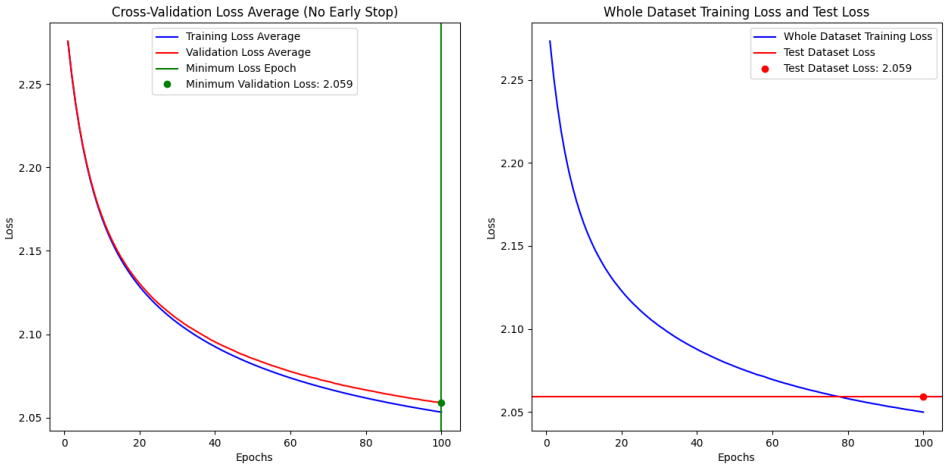
(a) Training and validation loss with batch size 200

(b) Test loss with batch size 200

Figure 24: Training and test performance with learning rate as 3×10^{-6}, batch size as 200, z-score norm

We also tuned the batch size. Through all the batch size, there is few difference in performance. We used the plots of batch size as 1 and 200 which are the smallest and largest batch size we tried. They are shown as figure 23 and 24. Unlike logistic regression, we see that the batch size have a little impact on the validation and test

loss, one major difference is that for batch size of 200, it triggered early stop early than batch size of 1 and 15. One hypothesis we made about this fact is that the dataset we used to train softmax regression is much larger than what we used to train logistic regression, thus the size of minibatch has less affect on training.



(a) Training and validation loss with min-max norm

(b) Test loss with min-max norm

Figure 25: Training and test performance with learning rate as 3×10^{-6}, batch size as 15, min-max norm

The plot (figure 25) of min-max normalization is similar to the those we did in logistic regression. It shows that the model didn't reach the minimum point as we change the z-score normalization to min-max and hold everything else constant. Combining the similar results in logistic regression, we concluded that the at same learning rate and batch size, using z-score normalization converges make the model converge to minimum faster than using min-max normalization.

6 Team contributions

Xiaoyan and Dongze equally distributed all the work in data preprocess and network construction. We didn't specifically assign the job for each of us. We took turns that one wrote a test and the other person made the test pass in the process of implementing data preprocess methods. We came up with the structure of network together, and implemented logistic regression and softmax regression together. During this process, we met together in library, constructed the network together, took turn to serve as the primary programmer when constructed netwrok, and kept actively communication. During hyperparameters tuning, we both ran same amount of load during the two weeks.

Beyond the equally distributed works in data preprocessing, Xiaoyan is the primary programmer for setting up training and testing method and the assistant of Dongze for setting up cross-validation method. Also, Xiaoyan took more jobs on composed the report.

Dongze is the primary programmer for setting up cross-validation, initialization and also take care of the main.py file. Dongze also assist Xiaoyan to set up the training and testing method. Also, Dongze took more jobs on visualization.

But both of us did some jobs on visualization and reports. Both of us agreed that we contributed equally in this project.