Multi-layer Neural Network With KMNIST Classification

Dylan Haar dhaar@ucsd.edu
Taehyung Kim tak088@ucsd.edu
Tae Kun Kim tkkim@ucsd.edu
University of California, San Diego
La Jolla, CA, USA

Abstract

Neural networks are considered to be the state of the art of technologies, and in the center lies the network's ability to learn from its mistakes. What is more, the configurability of this epitome of human creation allows the technology to be versatile enough to be used in different settings. In this report, the team investigates how back propagation, when paired with momentum, regularization, and various activation functions, helps neural network improve its accuracy in identifying handwritten Japanese characters. With the best model we were able to get 88.18% test accuracy with following configurations ['layer specs': [784, 512, 10], 'activation': 'ReLU', 'learning rate': 0.0005, 'batch size': 128, 'epochs': 110, 'early stop': True, 'early stop epoch': 5, 'L2 penalty': 0.0001, 'momentum': True, 'momentum gamma': 0.9].

1 Back Propagation

To check that our implementation of the back propagation feature, we created a table to compare the gradient values used to update the weights in our code with the expected gradient values. We utilized $\epsilon = 10^-2$ to check our difference of gradient for weights and bias obtained after back propagation is within $O(\epsilon^2)$. The formula for the expected gradient values is as follows:

$$\frac{d}{dw}E^n(w) = \frac{E^n(w+\epsilon) - E^n(w-\epsilon)}{2\epsilon}$$

Based on the table, we can observe that our difference of gradient for weights and bias after back propagation is within $O(\epsilon^2)$, which proves that our implementation of back propagation is correct.

Layer	Kind	Index	Back Prop. Gradient	Approx. Gradient
Input to Hidden	Weight	0, 0	0.006513078233333693	0.006513078230696578
Input to Hidden	Weight	1, 0	0.009713369112478748	0.009713369103669933
Input to Hidden	Bias	0	0.13525812543502738	0.1352581015957366
Hidden to Output	Weight	0, 0	0.013943669035177568	0.013943968057583689
Hidden to Output	Weight	1, 0	0.012796905992820207	0.012797137139175518
Hidden to Output	Bias	0	0.012069697713757654	0.012069891650945408

Table 1: Empirical and theoretical gradient values

2 Model Training Procedure

Our neural network model is composed of 3 layers: input layer(784 units), hidden layer(128 units), output layer(10 units). Since our model had to perform classification problem with more than 2 categories, we used softmax function from hidden layer to output layer and ReLU activation function from Input layer to hidden layer. As the model contained hidden layer we had to use forward propagation and backward propagation while training procedure in order to update the weights of each layer, thereby increasing the model's performance.

Generally, we utilized minibatch stochastic gradient descent method which performs stochastic gradient descent on small set of data based on batch size of 128. We adopted momentum in the update rule so that we can speed up mini batch learning. We initialized momentum as 0.9, learning rate as 0.0005 and epoch as 100. Also, by utilizing early stopping method and patience value as 5, we stop training the model when the validation loss starts to rise. We wait for 5 iterations and on the 5th rise of validation loss we completely stopped training procedure to prevent overfitting the model.

Figure below shows that our model early stopped on 25th epoch although we tried to train the model for 100 epochs. The plots show that validation loss rising at the end of epochs which resulted early stopping and the accuracy rising significantly until 6 epochs. The final test accuracy was 84.12% and we believe that we can improve the accuracy with different activation functions or different learning rates, momentum, and regularization.

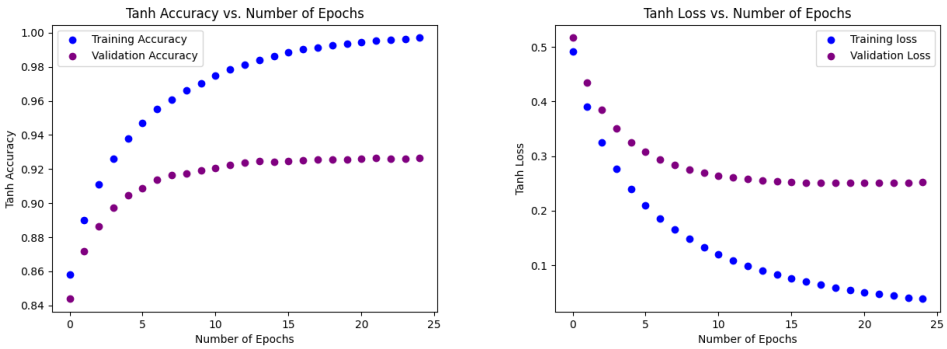

Figure 1: Training and validation accuracy and loss changes over epochs

3 Regularization

To observe how regularization can help with our network's accuracy, we first performed the L_1 regularization. Here, we trained our model over n epochs, where the model's validation loss started to go up. To be sure that the climbing validation loss was not a coincidence, we trained our model $0.1n$-many more epochs to observe that the validation loss continues to grow and ensure that the model must be stopped from training.

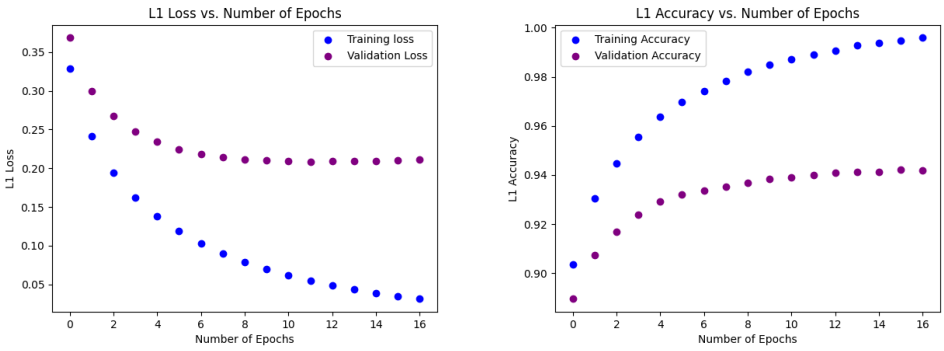

Figure 2: L1 regularization Loss and Accuracy

Next, to experiment the effects of different regularization methods, we performed L_2 regularization on the model.

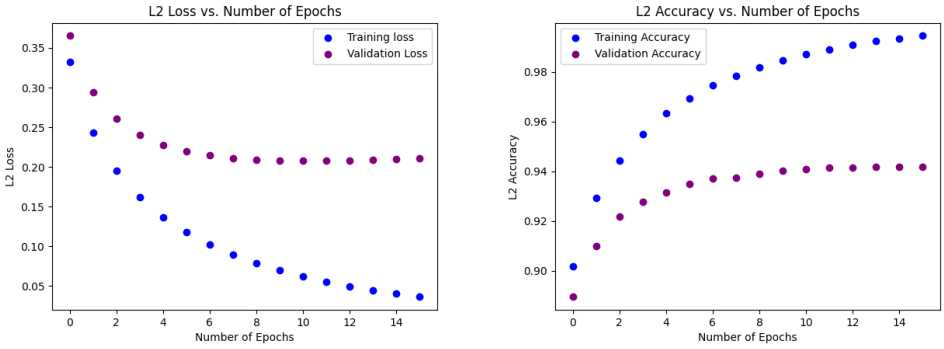

Figure 3: L2 regularization Loss and Accuracy

Here, we do not see a noticeable difference between L_1 and L_2 regularization methods.

4 Activation

In order to increase the accuracy of our model, we tried using different activation functions: sigmoid and ReLu.

Sigmoid: $f(z) = \frac{1}{1+e^{-}z}$

ReLu: f(z) = max(0, z)

When we used sigmoid activation function based on our baseline model, the test accuracy was 84.57% and ReLu activation function gave 86.65% accuracy on the test data set. Since our baseline model's test accuracy was 84.12%, both activation functions slightly improved the model's performance.

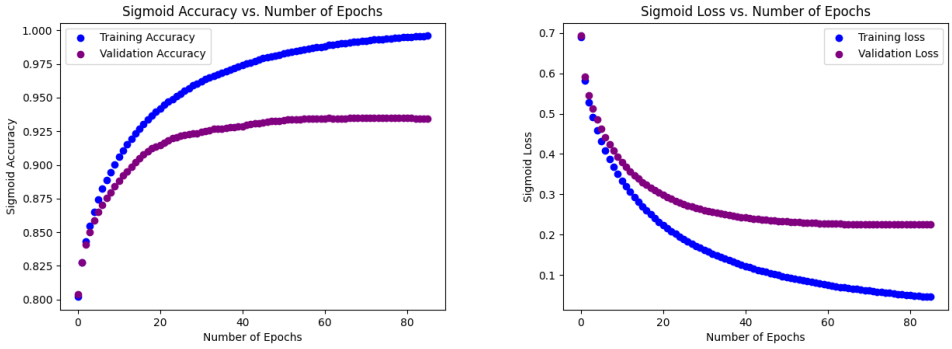

Figure 4: Sigmoid activation Accuracy and Loss

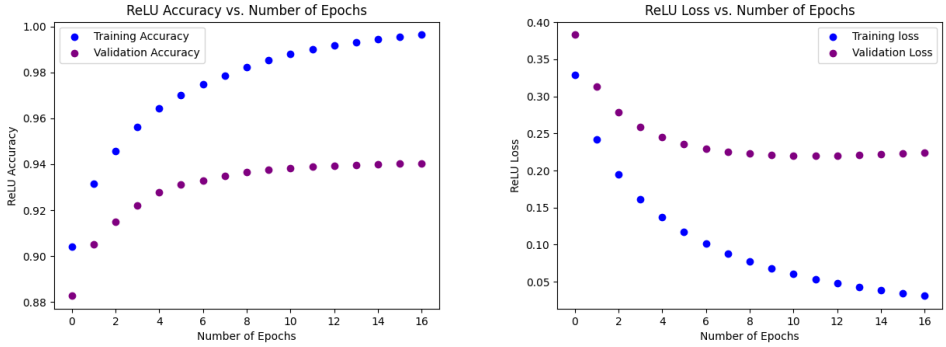

Figure 5: ReLu activation Accuracy and Loss

5 Network Topology

5.1 resizing hidden units

We changed the number of neurons in hidden layers to test if different number of neurons would increase model's performance. While doubling the number of neurons increased accuracy up to 85.1%, halving the number of neurons decreased accuracy to 80.8%. We obtained expected result because we originally thought that increasing the number of neurons would increase accuracy since it can perform more complicated computation with more weights and decreasing the number of neurons would decrease accuracy because of less weights in layer.

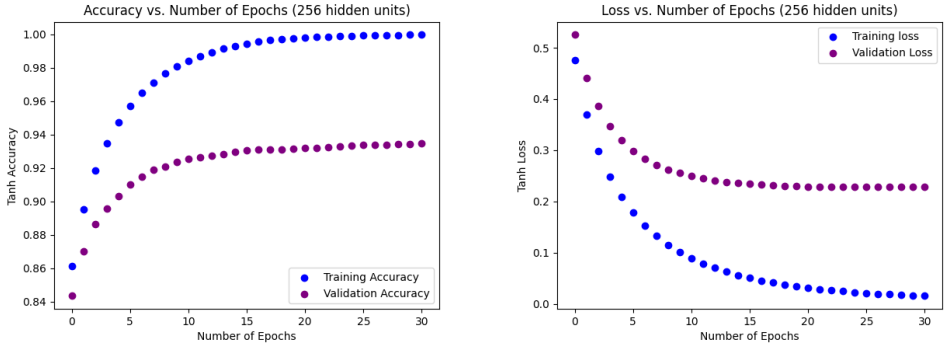

Figure 6: Changing number of neurons in hidden layer

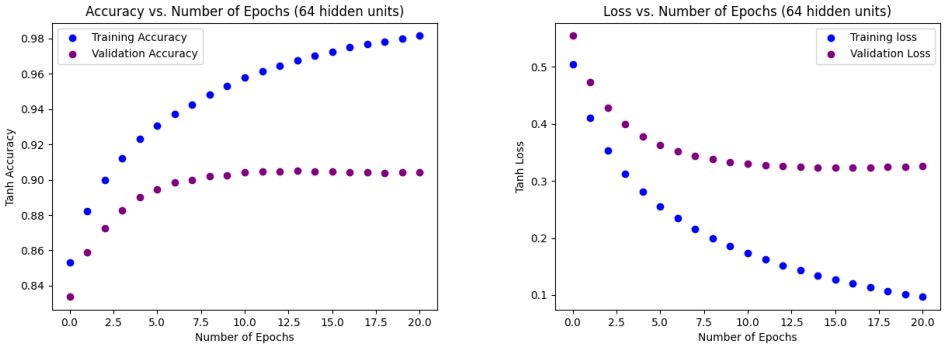

Figure 7: Changing number of neurons in hidden layer

5.2 changing number of hidden layers

We also increased the number of hidden layers to increase test accuracy of model. In order to have approximately same number of parameters compared to model with 1 hidden layer, we set number of units for each layer as [784, 110, 110, 10]. The test accuracy with 2 hidden layer was 83% which was less than expected.

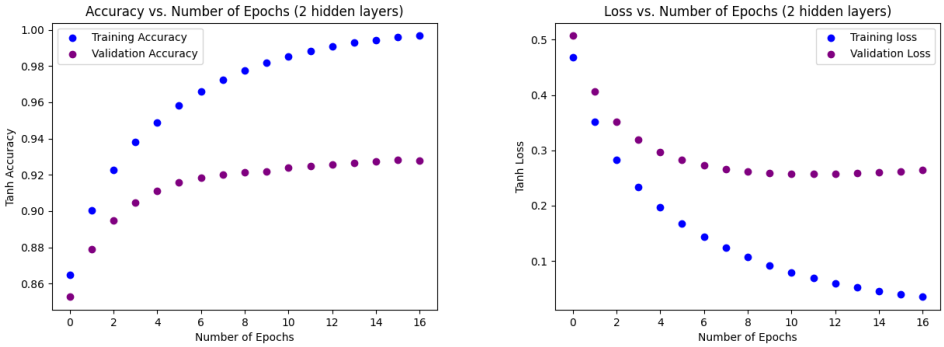

Figure 8: Increasing number of hidden layer

6 Team Contribution

Dylan Haar
Dylan was one of the key players in this project in that he structured and debugged most of the code work especially forward propagation and backward propagation that went into this project. Without Dylan, much of the project would not have been done.

Taehyung Kim
Taehyung was another key player in this project. Along with Dylan and Tae Kun, Taehyung coded the algorithms for training model and experiments. Moreover, Taehyung participated in creating the Latex file for this very report.

Tae Kun Kim
Tae Kun, along with the his teammates, contributed irreplaceable work to the team. He also co-wrote and debugged the overall neural network setup as well as its back propagation. He was able to help the team move forward by collaborating with his team members and making sure that all the code the team wrote together made sense in terms of math as well as the logistics of neural network.

7 References

[1] Gary Cottrell, Lecture3, BackProp: Representation, lecture notes, University of California, San Diego, delivered 2022.

[2] Gary Cottrell, Lecture 4A, A Few Notes on Improving Generalization, lecture notes, University of California, San Diego, delivered 2022.