Prediction Model for Modified National Institute of Standards and Technology Database via Multi-Layer Neural Network

Tianqi Zhao
PID: 14150941
email: tiz137@ucsd.edu

Siqi Huang
PID: A14143758
email:sih041@ucsd.edu

Alon Tron
PID: A16154189
email: atron@ucsd.edu

1. Abstract

In this project, our objective is to use a multi-layer neural network to identify the specific number displayed in the graphs of the dataset from Modified National Institute of Standards and Technology Database. We approach our goal using hidden layer implemented with back propagation algorithms and three different activation functions to choose for hidden layers: sigmoid, tanh and ReLU, while the output layer will always be the softmax activation function. After training, the model is expected to have approximately 90% chance to successfully recognize the number displayed in the test image.

2. Classification

2.1 Read MNIST data

Please see the coding file.

2.2 Implement Backpropagation

To compare the numerical approximation of the gradient and the gradient obtained in back propagation, we implemented a method:

def numerical_comparison(model, i, j, m, n, k, epsilon, X_train, y_train):

Which takes in the neuralnet model, i, j as coordinate for one weight (w1), m, n as coordinate for the second weight (w2), k as coordinate for one bias (b), and the training sample with target output. (Note: we used a network with only one hidden layer with 50 hidden units)

In the method, we iterated through the layers in the network. So it will firstly add epsilon to only w[i][j] (w1) in the input->hidden layer, and call forward_pass to calculate the loss. Then the method change w1 to original w1+epsilon in the layer, then calculate the loss and numerical approximation, saved to num_approx_w1. After recovering the original weights in the layer, the method does the same thing to w2 and b, changing only the specific w[m][n] and b[0][k], and calculate the numerical approximation. After calculating all the numerical approximations, the method changes all the weights and bias back to original, and call forward_pass and backward_pass, getting the specific d_w[i][j], d_w[m][n] and d_b[0][k] in the layer, storing in bp_grad_w1, bp_grad_w2 and bp_grad_b, respectively.

The method does the same thing to the hidden->output layer. Then we obtain a table for one hidden bias weight (b in the input_to_hidden_layer row), one output bias weight(b in the hidden_to_output_layer row), two input to hidden weights (w1, w2 in the input_to_hidden_layer row), two hidden to output weights (w1, w2 in the hidden_to_output_layer row), comparing their numerical approximation and backpropagation gradient in columns.

In **Table 2.2.1**, we called the method after training model for one mini batch, with epsilon = 0.01, w1=w[0][0], w2 = w[0][1], b = b[1] for both input->hidden and hidden->output layers. As we can see, the differences between num_ap (numerical approximation) and bp_grad (gradient obtained in bp) are significantly lower than the error bound 0.01^2. (The table is stored as part2b.csv)

2.3 Training with Vectorized Update

The training procedure for the neural network with one hidden layer of 50 units is as follows: Each image is a training point for the neural network. The matrix of each image is computed through the hidden layer with the activation function tanh into 50 units and outputted through the activation with softmax function into the vectorized output of 10 units.

The backpropagation procedure can be summarized into: forward_pass the training set, so the model will keep track of the input and output, and we obtain the delta for the out-most(sigmoid activation layer) layer. Then backward_pass the delta to update the gradient of weights, bias and input for each layer (not activation layer). Then we iterate through each layer, update the weights and bias with stored gradients. After update, we forward pass the validation data to obtain validation loss and accuracy.

In the training, we use 50 hidden units for the hidden layer with the activation function of tanh, a momentum of 0.9 , a learning rate of 0.08 with the highest epoch run of 300. With an early stopping at epoch 132, we obtain the best test accuracy of 90.4% at epoch 127, which is five epochs away from the one with the highest epoch. **Figure 2.3.1** shows the training accuracy and validation accuracy for our model and **Figure 2.3.2** shows the training and validation losses for our model.

2.4 Experiment with Regularization

In this training, we keep the learning rate, momentum and the hidden layer with the activation function of tanh of 50 hidden units as in 2.3, but use the highest epoch of 140 and a L2 regularization with a penalty of 0.001. The best test accuracy is around 90.65%, existing at the last epoch, which is around 0.25% higher than the one without. . **Figure 2.4.1** shows the training accuracy and validation accuracy for our model and **Figure 2.4.2** shows the training and validation losses for our model. In the process of training with a 10% increase in the number of training epochs from 2.3, we observe that there is no early stopping compared to the epoch number it stops in 2.3 and has the tendency to continue decrease in validation losses, which means that a higher accuracy might exist if we increase the number of training epochs.

2.5 Experiment with Activations

In this section, we train our model using two different activation functions for the hidden layer, which are sigmoid and ReLU. We keep the learning rate, momentum and the highest epoch of 300 from 2.3 and changes the activation function for the hidden layer.

First, we use the sigmoid function as our activation function for the hidden layer. **Figure 2.5.1.1** shows the training accuracy and validation accuracy for the current model and **Figure 2.5.1.2** shows the training and validation losses for the current model. We observe that with the same highest epoch as 2.3, the training does not early stop, which means that the losses continue to decrease and higher accuracy might occur. Without an early stop, we obtain our best accuracy as 92.89% at the last epoch, which achieve an even higher accuracy compared with that of tanh activation function for hidden layer. Also, compared with the plots of 2.3, both accuracy and losses curve are smoother.

Then, we have ReLU as our activation function for the hidden layer. **Figure 2.5.2.1** shows the training accuracy and validation accuracy for the current model and

Figure 2.5.2.2 shows the training and validation losses for the current model. We observe that with an early stop at epoch 165, we have our best accuracy as 72.62% at epoch 158, which has a much lower accuracy compared with the hidden layer using sigmoid and tanh. Also, unlike the plots for hidden layer with tanh and sigmoid function, both losses and accuracy curve fluctuates a lot as the number of epoch increases and does not smoothly increase

2.6 Experiment with Network Topology

In this section, we keep the learning rate, momentum and the highest epoch of 300 and activation function tanh for the hidden layer from 2.3. In the first part, we will use two different numbers of hidden unit for the hidden layer. In the second part, we will use two hidden layers.

For the first part, with hidden unit of 50 for the hidden layer, we obtain the best test accuracy of 89.29% at epoch 105 for an early stopping at epoch 134. **Figure 2.6.1.1a** shows the training accuracy and validation accuracy for the current model and **Figure 2.6.1.1b** shows the training and validation losses for the current model. We observe that the test accuracy for this model is a little lower than that from 2.3. Also, we observe that the early stopping epoch stops more epochs after the epoch with the highest accuracy compared with that from 2.3. In the plots for accuracy and losses, the differences between the validation and training are closer as the number of epoch increases compared with that from 2.3. With hidden unit of 100 for the hidden layer, we obtain the best test accuracy of 91.13% at epoch 111 for an early stopping at epoch 114. **Figure 2.6.1.1a** shows the training accuracy and validation accuracy for the current model and **Figure 2.6.1.1b** shows the training and validation losses for the current model. We observe that the test accuracy for this model is higher than that from 2.3. Also, we observe that the early stopping epoch stops less epochs after the epoch with the highest accuracy compared with that from 2.3. In the plots for accuracy and losses, the differences between the validation and training are larger as the number of epoch increases compared with that from 2.3.

For the second part, to achieve an approximate amount of parameters as that for one hidden layer of 50 hidden units, we choose to have two hidden layers each with 45 hidden units. With the two hidden layers of hidden units 45, we obtain the best test accuracy of 91.10% at epoch 193 with an early stopping at epoch 197. **Figure 2.6.2.1** shows the training accuracy and validation accuracy for the current model and **Figure 2.6.2.2** shows the training and validation losses for the current model. We observe that the test accuracy for this model is higher than that from 2.3. Also, we observe that this model uses more epoch to reach its early stopping compared with that from 2.3. In the plots for accuracy and losses, the differences between the validation and training are similar as the number of epoch increases compared with that from 2.3. However, both have fluctuated more compared with that from 2.3

3. Contribution

The successful completion of the project is contributed to the collaboration and hard work of all three of the group members.With the skeleton of trainer written by Alon, instead of dividing the implementation into three separate pieces, the group members of the project discussed, implemented and debugged the backpropagation algorithm and trainer details together. The analysis of each problem is also the result of thorough discussion.

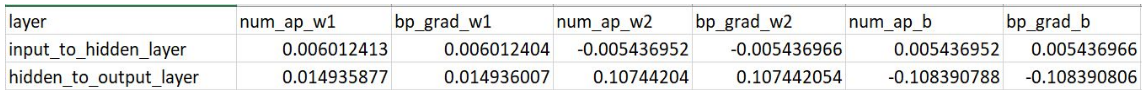
This is the personal contribution of each one of the team members:

Tianqi Zhao directly contributed to every part of the project. Tianqi implemented the Layer class, parts of the Neural Network class, Trainer part and most importantly solved many problems in the training implementation that we have been faced using her good mathematical understanding.

Siqi Huang contribution was implementing parts in the Activation and the Trainer classes. Siqi implemented big parts of the Trainer and used the different features to understand and investigate the different configurations and abilities of the neural network in order to write the report and address all the questions.

Alon Tron was writing part of the Activation class, Neural Network class and the Trainer. Alon started to build the Trainer from scratch and implemented its skeleton of which we added more features after. Alon also helped debugging and searching for the problems we had.

4. Citations, Figures, Tables, and References



layer	num_ap_w1	bp_grad_w1	num_ap_w2	bp_grad_w2	num_ap_b	bp_grad_b
input_to_hidden_layer	0.006012413	0.006012404	-0.005436952	-0.005436966	0.005436952	0.005436966
hidden_to_output_layer	0.014935877	0.014936007	0.10744204	0.107442054	-0.108390788	-0.108390806

Table 2.2.1 Comparison of Numerical Approx. and Gradient Calculated by BP. one hidden bias weight (b in the input_to_hidden_layer row), one output bias weight(b in the hidden_to_output_layer row), two input to hidden weights (w1, w2 in the input_to_hidden_layer row), two hidden to output weights (w1, w2 in the hidden_to_output_layer row), Epsilon = 0.01

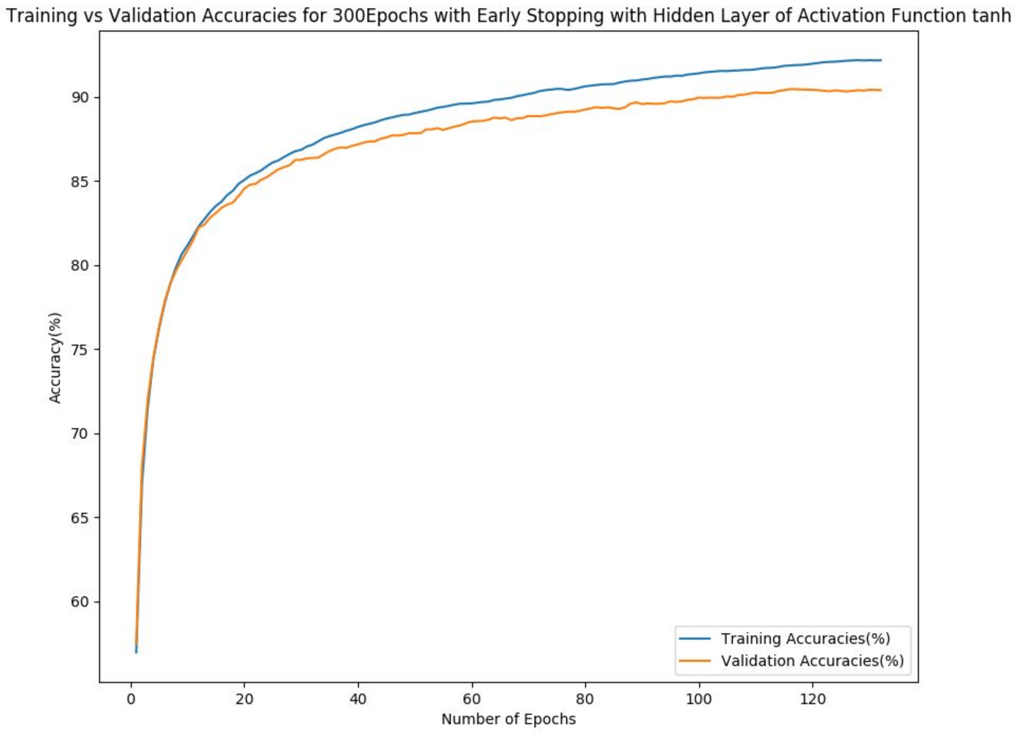


Figure 2.3.1 The above figure is the training accuracy vs validation accuracy for early stopping of 300 epochs with the multi-layer neural network of a momentum of 0.9, a learning rate of 0.08, one hidden layer of tanh activation function and an output layer of softmax activation function

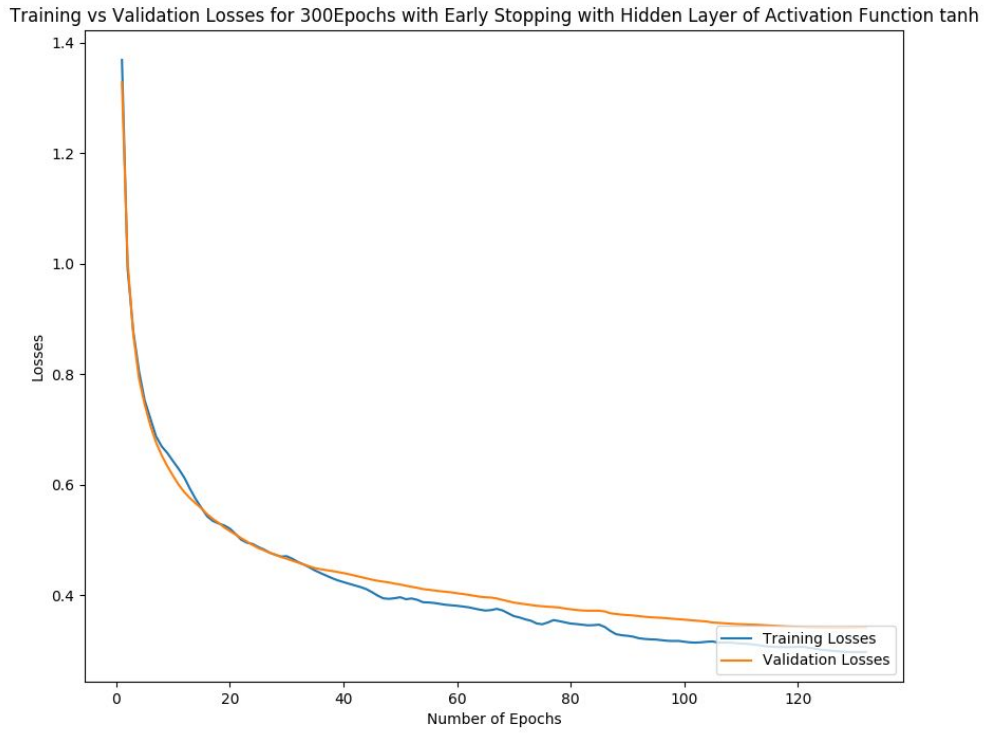


Figure 2.3.2 The above figure is the training losses and validation losses for early stopping of 300 epochs with the multi-layer neural network of a momentum of 0.9, a learning rate of 0.08, one hidden layer of tanh activation function and an output layer of softmax activation function.

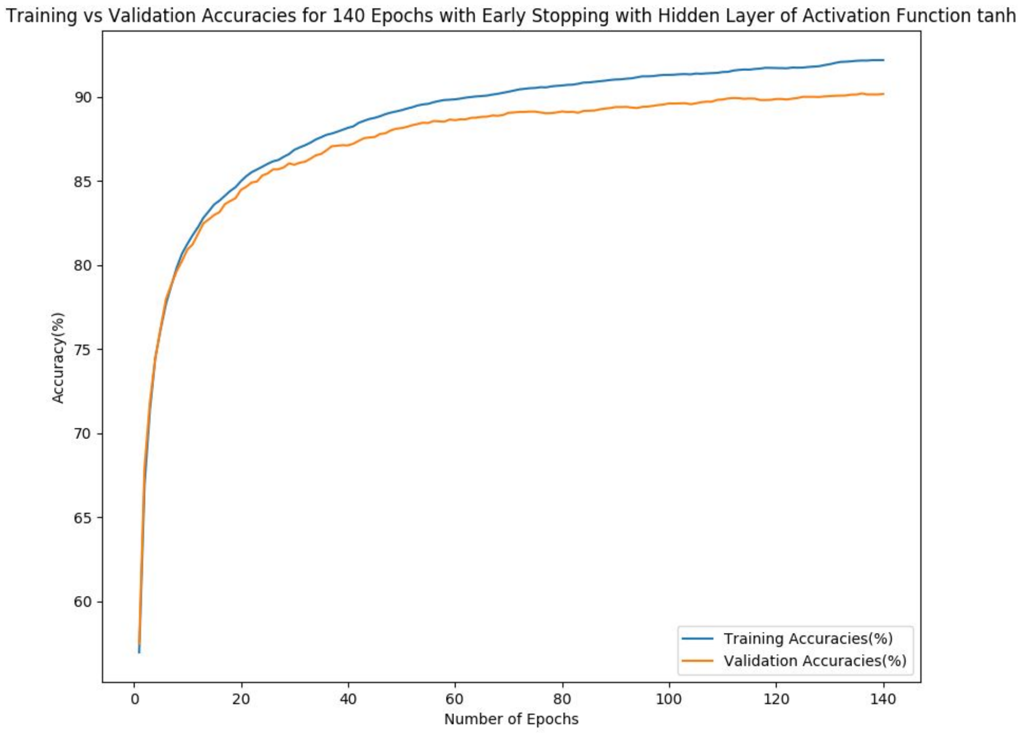


Figure 2.4.1 The above figure is the training accuracy vs validation accuracy for early stopping of 140 epochs with the L2 regularized multi-layer neural network of a momentum of 0.9, a learning rate of 0.08, one hidden layer of tanh activation function and an output layer of softmax activation function

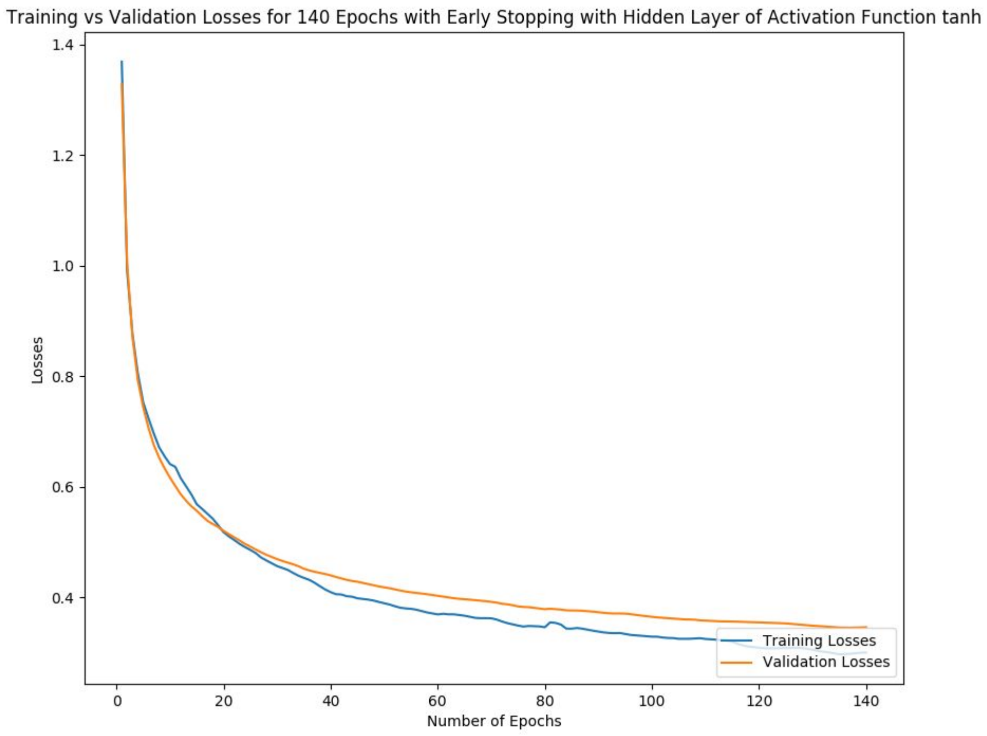


Figure 2.4.2 The above figure is the training losses and validation losses for early stopping of 140 epochs with the L2 regularized multi-layer neural network of a momentum of 0.9, a learning rate of 0.08, one hidden layer of tanh activation function and an output layer of softmax activation function.

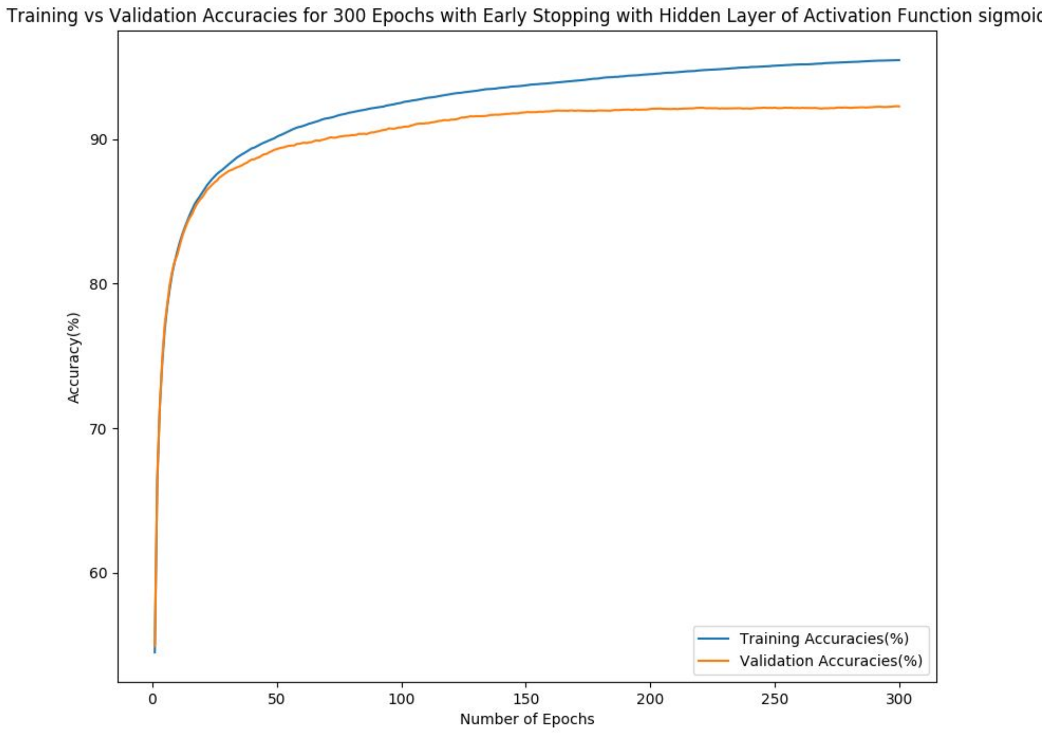


Figure 2.5.1.1 The above figure is the training accuracy vs validation accuracy for early stopping of 300 epochs with the multi-layer neural network of a momentum of 0.9, a learning rate of 0.08, one hidden layer of sigmoid activation function and an output layer of softmax activation function

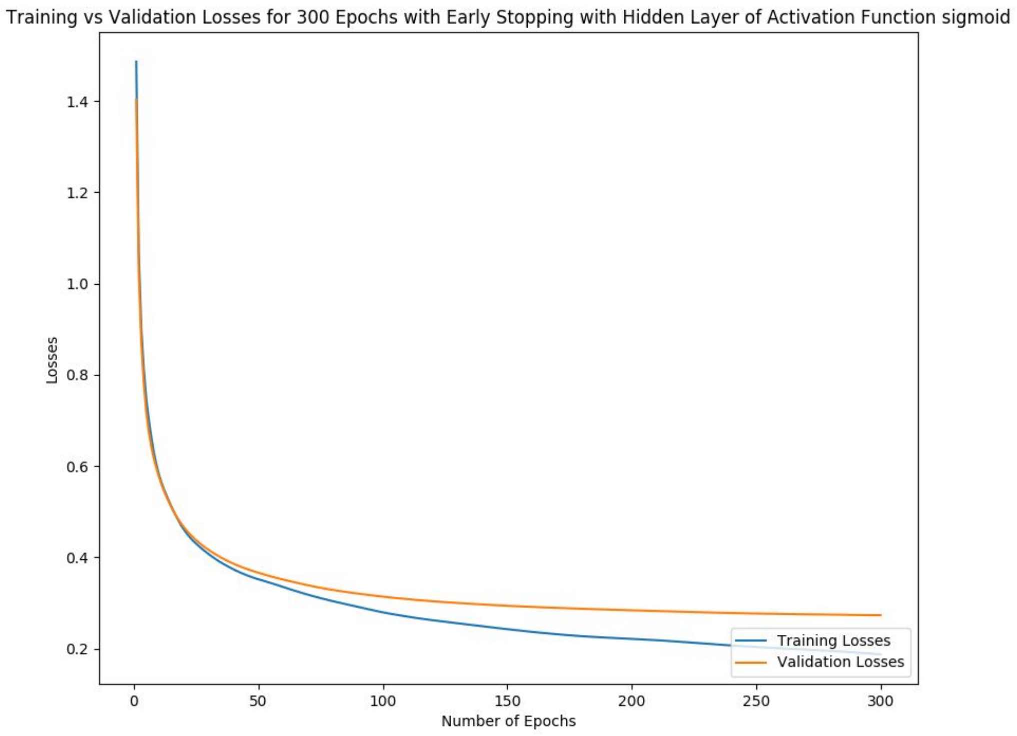


Figure 2.5.1.2 The above figure is the training losses and validation losses for early stop of 300 epochs with the multi-layer neural network of a momentum of 0.9, a learning rate of 0.08, one hidden layer of sigmoid activation function and an output layer of softmax activation function.

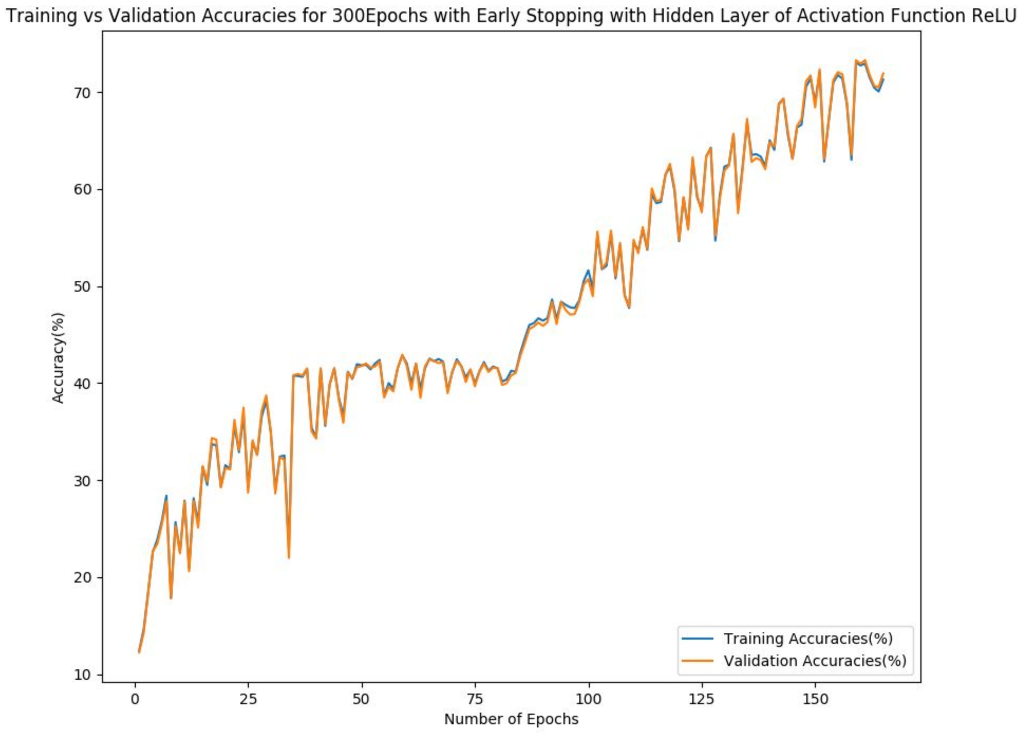


Figure 2.5.2.1 The above figure is the training accuracy vs validation accuracy for early stopping of 300 epochs with the multi-layer neural network of a momentum of 0.9, a learning rate of 0.08, one hidden layer of ReLU activation function and an output layer of softmax activation function

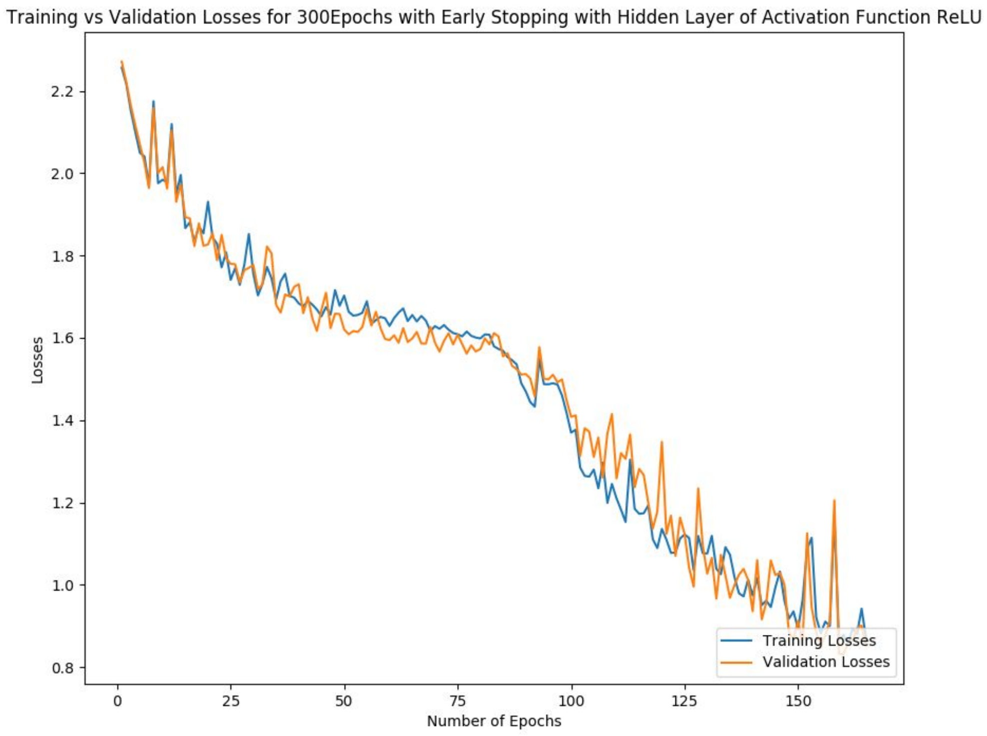


Figure 2.5.2.2 The above figure is the training losses and validation losses for early stop of 300 epochs with the multi-layer neural network of a momentum of 0.9, a learning rate of 0.08, one hidden layer of ReLU activation function and an output layer of softmax activation function.

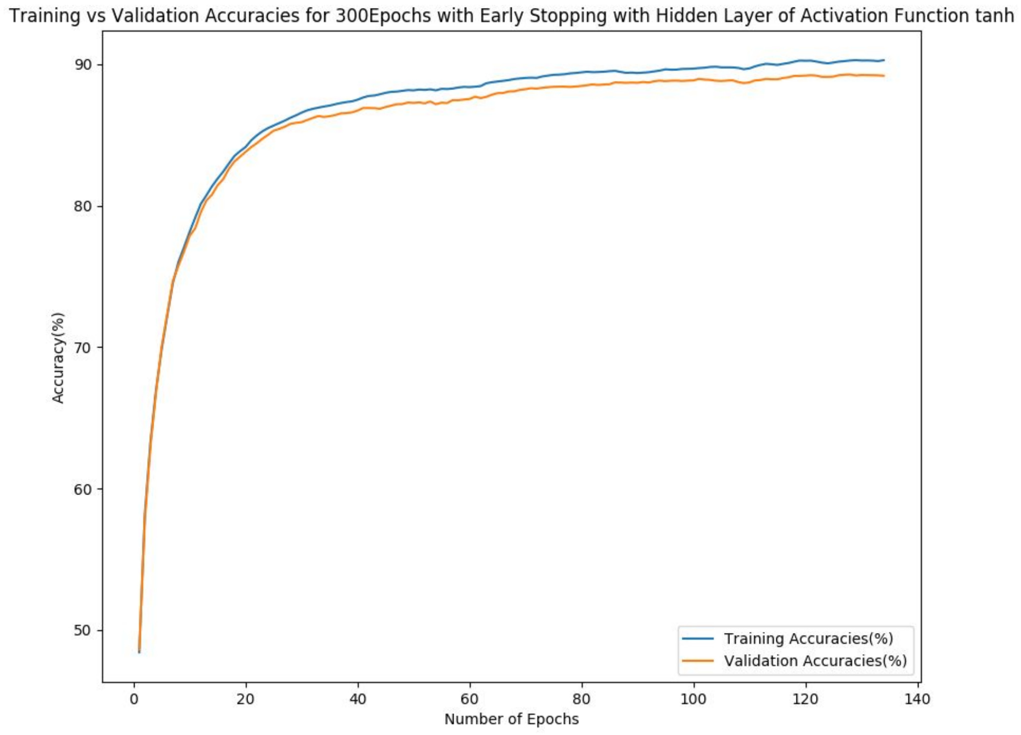


Figure 2.6.1.1a The above figure is the training accuracy vs validation accuracy for early stop of 300 epochs with the multi-layer neural network of a momentum of 0.9, a learning rate of 0.08, one hidden layer of tanh activation function of hidden units 25 and an output layer of softmax activation function

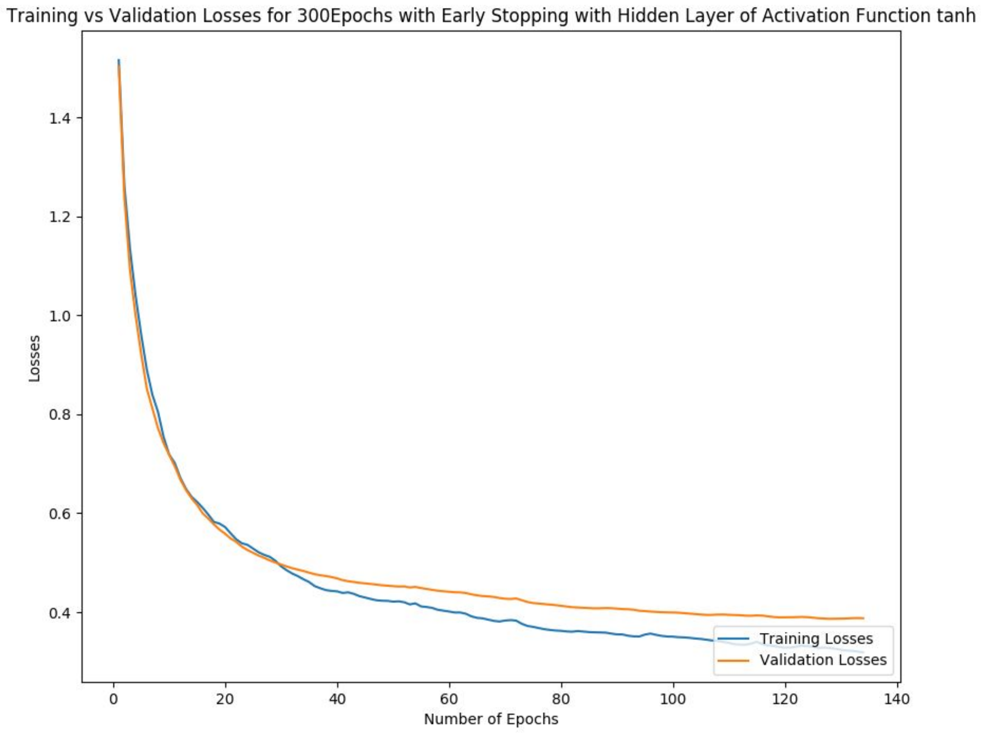


Figure 2.6.1.1b The above figure is the training losses and validation losses for early stop of 300 epochs with the multi-layer neural network of a momentum of 0.9, a learning rate of 0.08, one hidden layer of tanh activation function of hidden units 25 and an output layer of softmax activation function.

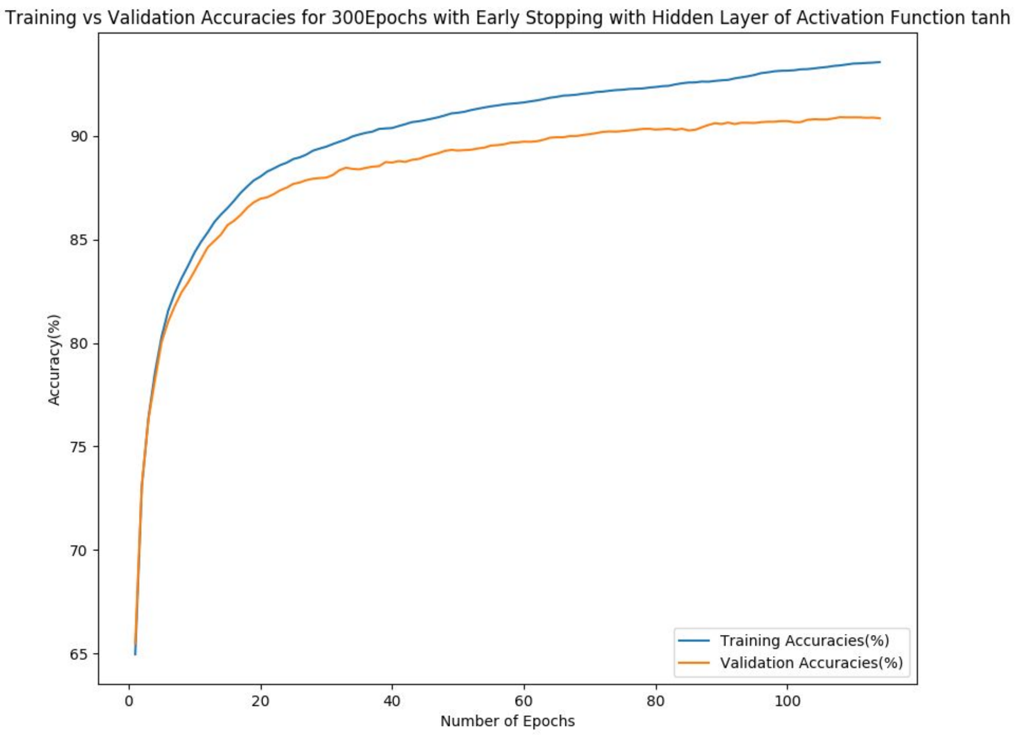


Figure 2.6.1.2a The above figure is the training accuracy vs validation accuracy for early stop of 300 epochs with the multi-layer neural network of a momentum of 0.9, a learning rate of 0.08, one hidden layer of tanh activation function of hidden units 100 and an output layer of softmax activation function

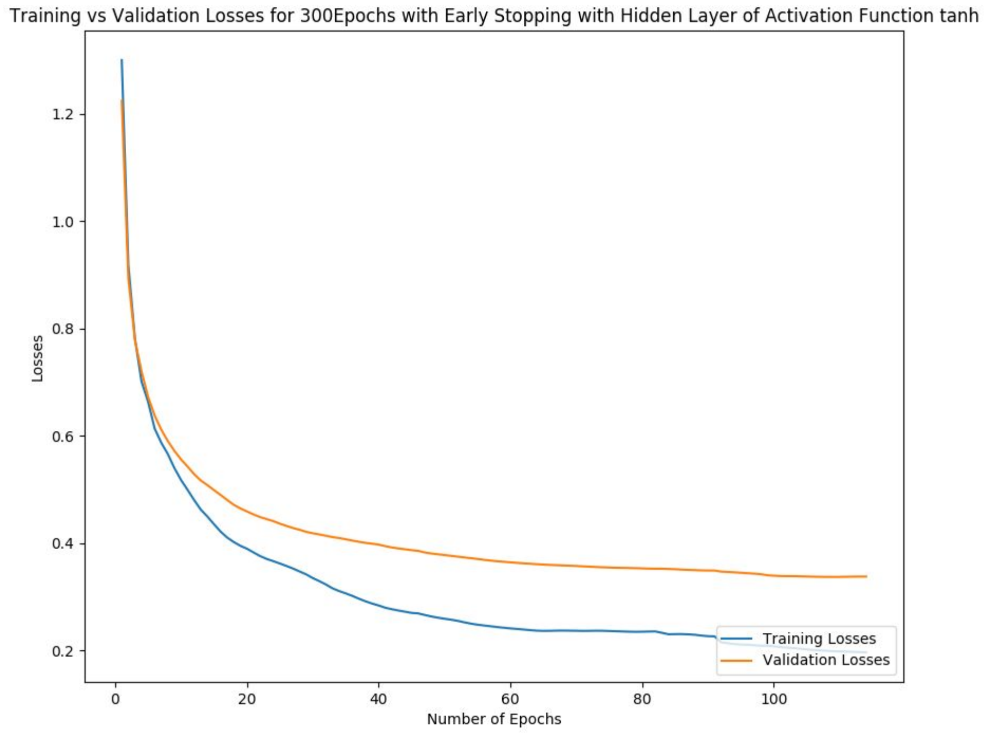


Figure 2.6.1.2b The above figure is the training losses and validation losses for early stop of 300 epochs with the multi-layer neural network of a momentum of 0.9, a learning rate of 0.08, one hidden layer of tanh activation function of hidden units 100 and an output layer of softmax activation function.

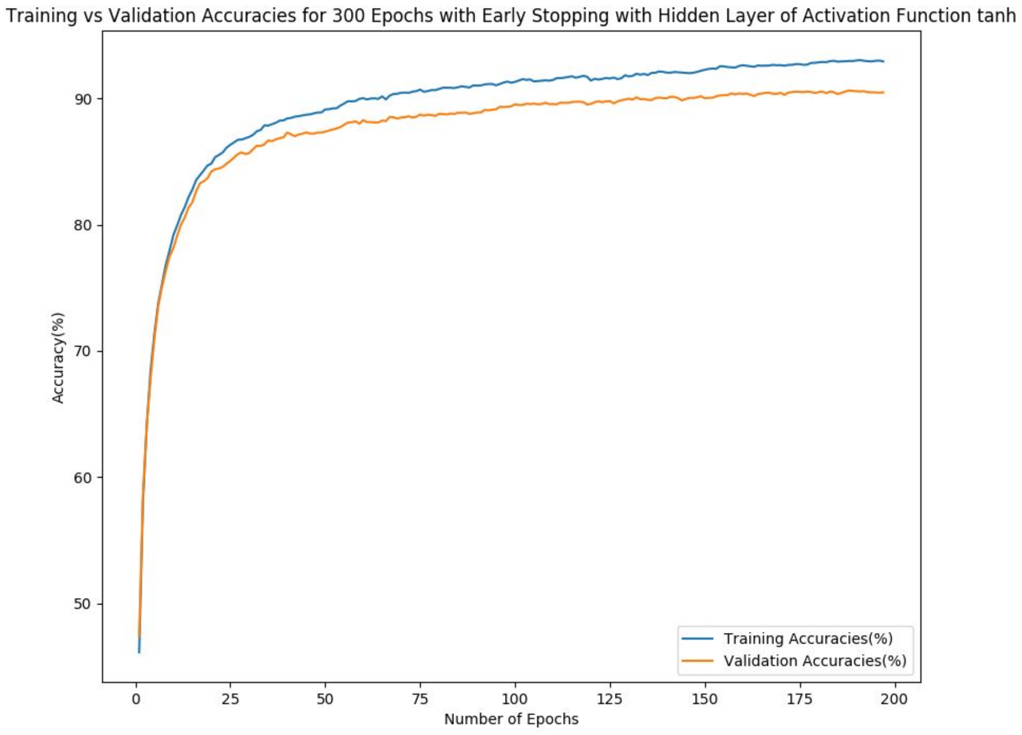


Figure 2.6.2.1 The above figure is the training accuracy vs validation accuracy for early stop of 300 epochs with the multi-layer neural network of a momentum of 0.9, a learning rate of 0.08, two hidden layers of tanh activation function of hidden units 45 and an output layer of softmax activation function

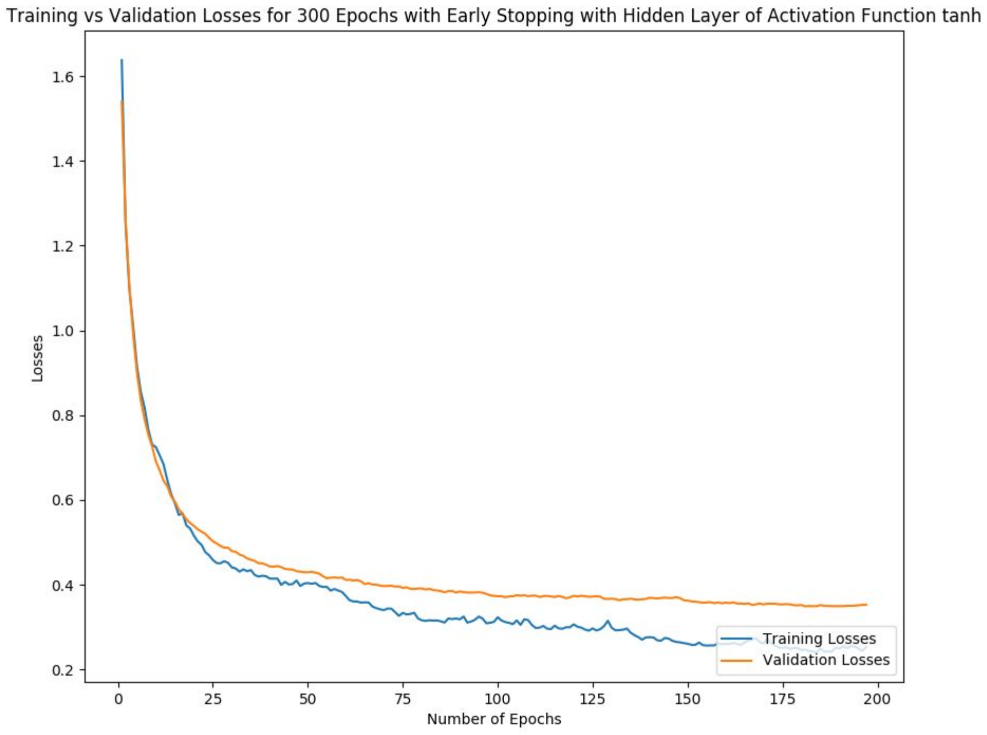


Figure 2.6.2.2 The above figure is the training losses and validation losses for early stop of 300 epochs with the multi-layer neural network of a momentum of 0.9, a learning rate of 0.08, two hidden layers of tanh activation function of hidden units 45 and an output layer of softmax activation function