Machine Learning, CSCI 5622, Assignment 3

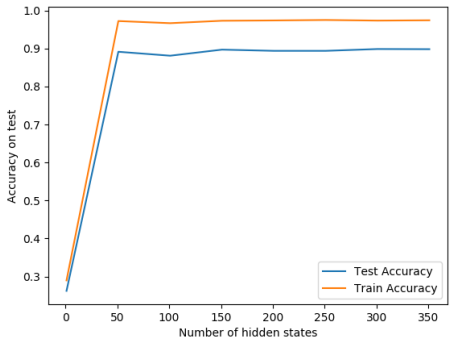
1. What is the structure of your neural network (for both tinyTOY and tinyMNIST dataset)? Show the dimensions of the input layer, hidden layer and output layer.

Dataset	Input Layer	Hidden Layer	Output Layer
tinyTOY	2	30	2
tinyMNIST	196	50	10

The neural network consists of an input layer, hidden layer and output layer. Number of nodes in input layer is equal to the number of features of the dataset. The dimension of the output layer is equal to the number of classes classified in the train data. By seeing the data, we know that tinyTOY has 2 features and tinyMNIST has 196 features. In tinyTOY, there are two classes and in tinyMNIST, as there are 10 numbers we need to classify, the dimension of the output layer is 10. The size of the hidden layer for tinyTOY is 30 and size for tinyMNIST is 50. The decision for hidden layer to be consisting of 50 nodes has been observed and explained in the following question.

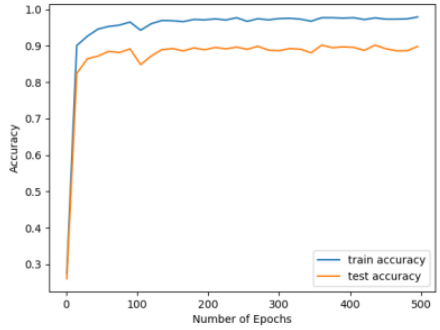
2. What the role of the size of the hidden layer on train and test accuracy (plot accuracy vs. size of hidden layer using tinyMNIST dataset)?

As shown in the graph below, as the dimension of hidden layer increases, the accuracy increases as an increase in number of nodes correspond to more learning in one epoch. But, after around 50 nodes, the accuracy stabilizes for both training and testing data. This means that after 50, increase in dimension of hidden layer does not increase the accuracy as the model is not learning more as nodes increase more than a threshold. We can also observe that accuracy on training set is more than the test accuracy. This is obvious because the model has already seen the train data and hence will classify the examples better than test data which is unseen.



3. How does the number of epochs affect train and test accuracy (plot accuracy vs. epochs using tinyMINST dataset)?

As seen in the graph below, accuracy of train and test both sharply increase as number of epochs increase. As at each epoch, the training examples are seen at a randomly different order, slight increases and declines in accuracy is observed. But after around 350 epochs, the accuracies stabilize. This pattern is observed because, after a large number of epochs, a large permutation of training example patterns has been seen by the model such that for following epochs, it does not learn any new patterns from the model.



4. Point out at least three layer types you used in your model. Explain what are they used for.

- **Convolutional Layer:** For every node in the convolutional layer, a weight matrix of the kernel size is initialized. For each stride of the input image, each cell of the input image is multiplied with each element of the weight matrix and summed up. Therefore the output of this layer is, for every node it is the matrix of sums of products for every stride. Each node in this layer is called a filter. The filters take a subset of input data by sweeping through the input data. This is also followed by a nonlinear activation. This layer is used to capture different patterns for every part of the image.
- **Max Pooling Layer:** At this layer, for every pool size dimension in the output of convolutional layer, the maximum value of the pool size is retained in the output of this layer. This layer effectively reduces the number of samples of the output of the previous convolutional layer. This operation reduces the number of operations required for the following layers.
- **Dense Layer:** This layer, also called a fully connected layer has a weight vector associated with every combination of number of nodes in previous layer and itself. It consists of a linear combination with the previous layer's output vector and weights of the dense layer. Linear combination is generally followed by a nonlinear activation. This layer is used to convert a vector of one dimension to another which is suited for the following layers.

5. How did you improve your model for higher accuracy?

- At first, I added a convolutional layer for the images, a flatten layer to convert it to 1D vectors as an input to a Dense layer for the output to classify 10 classes. The following tables show my experiments in choosing number of hidden nodes (filters), kernel size and strides for the convolutional layer. The best number of filters were 20, best dimension of kernel size is (10,10) and best strides after one sweep is (1,1).

Filters	10	97.21	20	97.59	30	97.46
Kernel Size	1,1	97.67	5,5	97.42	10,10	97.98
Strides	1,1	98.67	2,2	97.59		

- For better accuracy, I added a maxpool layer after the convolutional layer. The following table illustrates the experiments conducted to choose the best pool size. The best maxpool size chosen was (5,5)

Pool Size	2,2	98.7	5,5	98.81	10,10	98.38

- After the maxpool layer, there was a necessity to add another layer for better generalization of the dataset and to achieve higher accuracy. I added a fully connected layer with 1000 nodes and a nonlinear activation function 'relu'. This layer helped in increasing the accuracy by 0.71 percent.
- The final accuracy for the CNN for MNIST data was found to be 99.09.

6. Try different activation functions and batch sizes. Show the corresponding accuracy

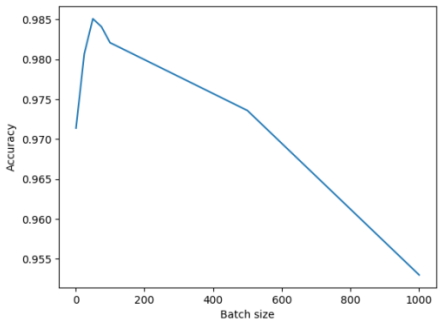
Activation functions have been used for both the convolutional and dense layer. The changing accuracies for change in activation functions have been tabulated by keeping the other parameters optimal according to the previous

experiments. The best activation functions for each of the corresponding layers have been selected for the model. The best activation for convolution layer is RELU and best activation for output dense layer is SOFTMAX.

Convolutional Layer	
RELU	98.21
SIGMOID	96.98
SOFTMAX	12.6

Output Layer	
RELU	14.67
SIGMOID	98.56
SOFTMAX	99.09

As shown in the graph below, as the batch size increases, the accuracy increases. But after an optimal batch size, as the batch size increases, the accuracy decreases. This is observed because the weights are updated once for a batch. And if the batch size is huge, the effect of each training example on the weights is reduced. The approximate optimal batch size is 128.

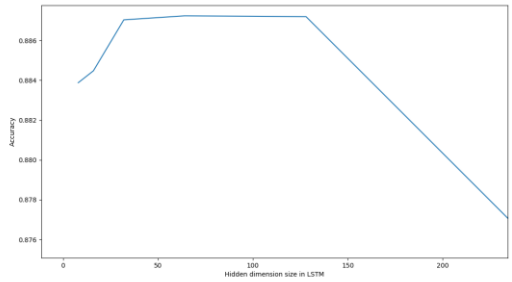


7. What is the purpose of the embedding layer? (Hint: think about the input and the output).

The output of the embedding layer is the input to the LSTM layer. It lets us treat individual words as units of meaning. The task at hand is to classify a review as positive or negative. Therefore, word meanings play an important role in this classification. For each word in the review, a vector is learned. These vectors correspond to the weight vector between the input layer and the embedding layer. As the weights vector should be of constant length, each review is padded or truncated to be of constant length. The embedding layer learns and transforms the semantic relationships between the words in the review. Instead of learning the vectors for the words, pretrained word embeddings can also be used which associates every word with a vector and acts as embedding layer in the pipeline.

8. What is the effect of the hidden dimension size in LSTM?

As seen in the graph below, as the number of hidden layer nodes increases, accuracy increases till around 30. The accuracy remains constant for further increase till 130 and decreases further. Therefore, the accuracy is maximum for the range of about 30 to 130 for this model. After a certain cap, the increase in hidden dimension does not affect the accuracy in a positive direction as a large number of nodes fail to learn the generalization of the features.



9. Replace LSTM with GRU and compare their performance.

LSTM provides an accuracy of 88.42 and replacing LSTM with GRU provides an accuracy of 88.53 percent.

10. Try to use pretrained word embeddings to initialize the embedding layer and see how that changes the performance.

The GloVe word to vector table has been used. Every word in the review was converted to a vector in reference to these mappings. The accuracy was found to be 88.73 percent. The GloVe file used is glove.6B.100d.txt which should be present in the same directory as the code for correct execution.