CS 689 Homework3 part 2 - Animesh Sengupta

Q5

A: [[0.9935],
[-0.0818],
[1.2234],
[1.7484],
[-0.3340]]

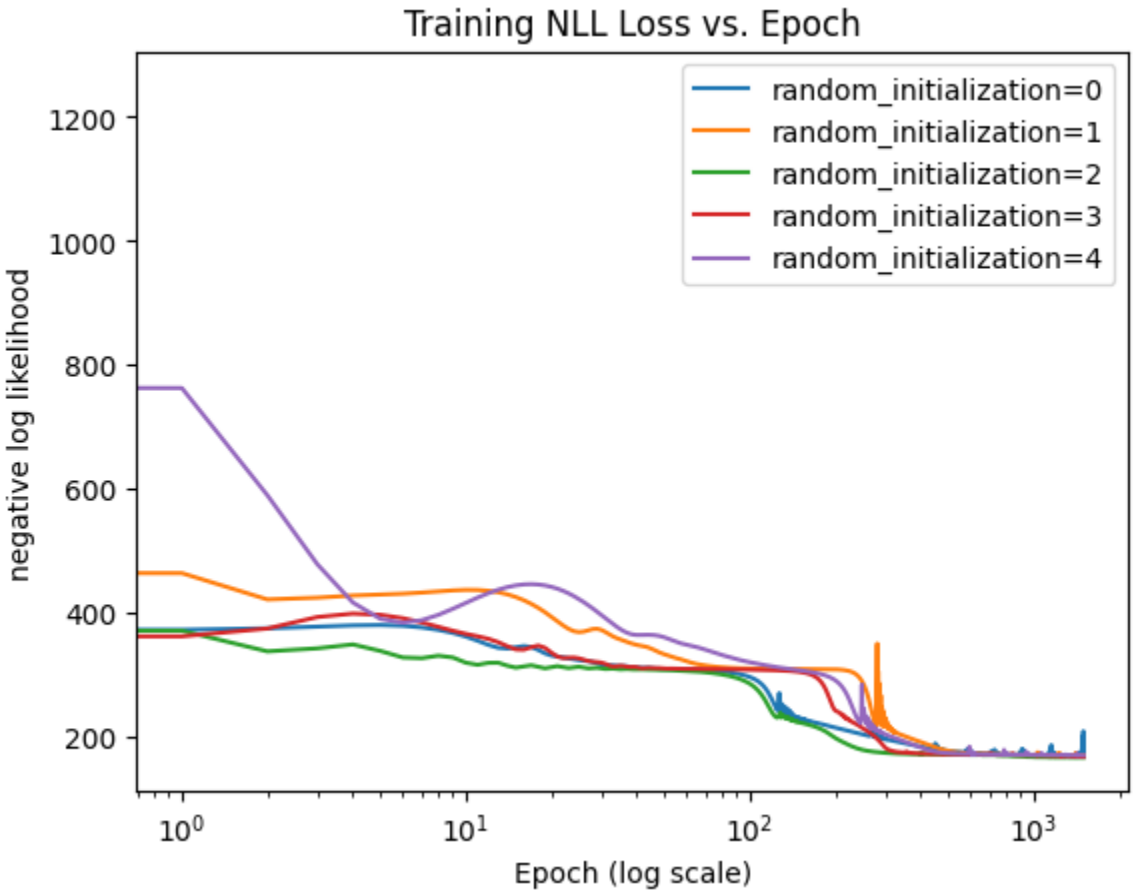
B:

[[1.7839],
[1.4218],
[1.8674],
[2.0829],
[1.3326]]

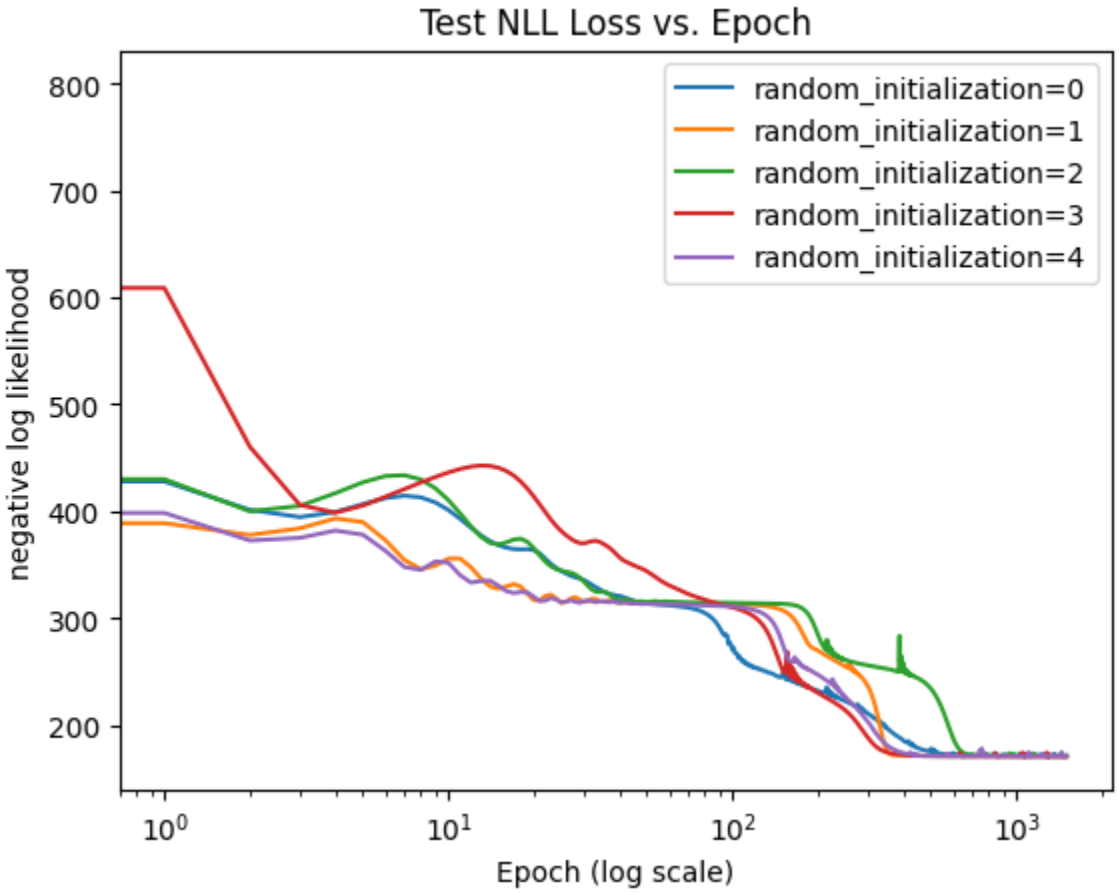
C:

609.3236694335938

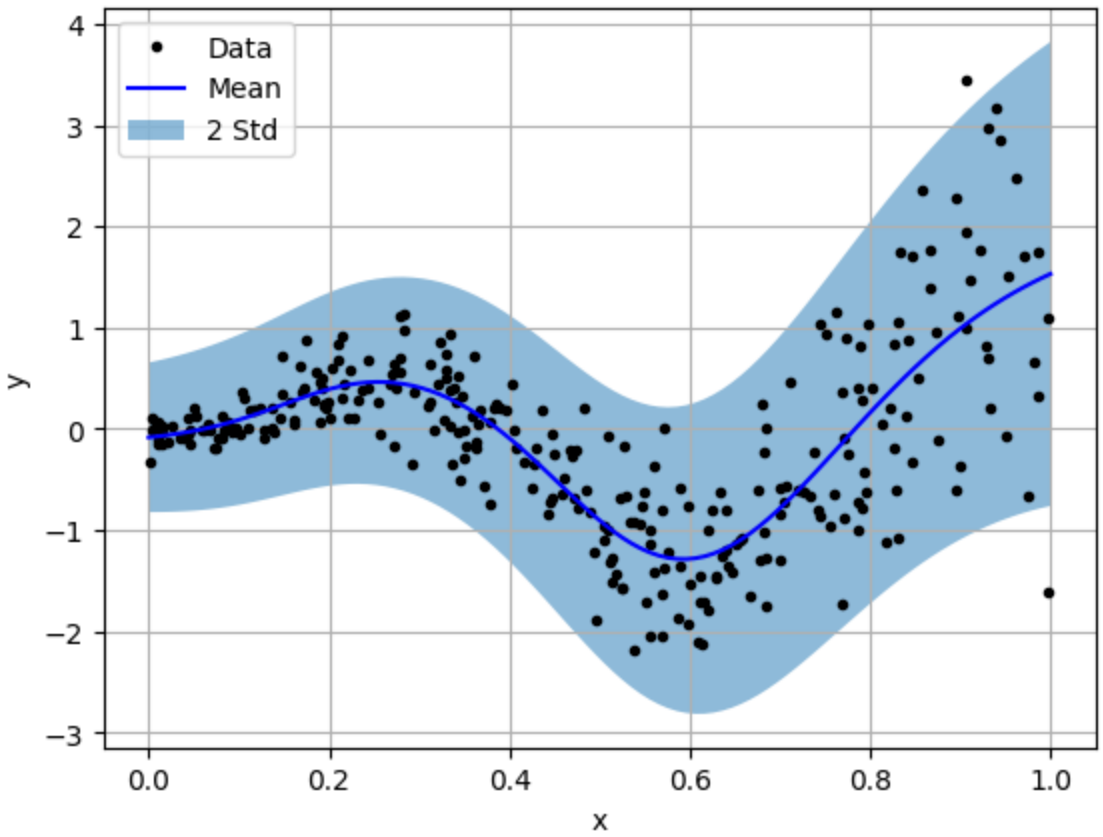
D:



E:


Test NLL Loss vs. Epoch
random_initialization=0
random_initialization=1
random_initialization=2
random_initialization=3
random_initialization=4
negative log likelihood
800
700
600
500
400
300
200
10^0
10^1
10^2
10^3
Epoch (log scale)

F:



Q6

A

I have experimented with 3 models , they're Multilayer perceptron i.e Fully connected simple neural network, Recurrent neural network and Transformers architecture. RNN preserves the hidden state and adds that as an input to the next layer. The Transformer consists of an encoder and a decoder, each made up of multiple layers of self-attention and feed-forward neural networks. For both RNN and transformer, was modelled to be many to one model. The input was passed by embedding layers to both RNN and transformer. For each model we chose MSE loss as it would be best to find the accuracy of our prediction of expression evaluation.

Hyper parameterx

Learning rate=1e-4

Epoch =1000

Batch size -1024

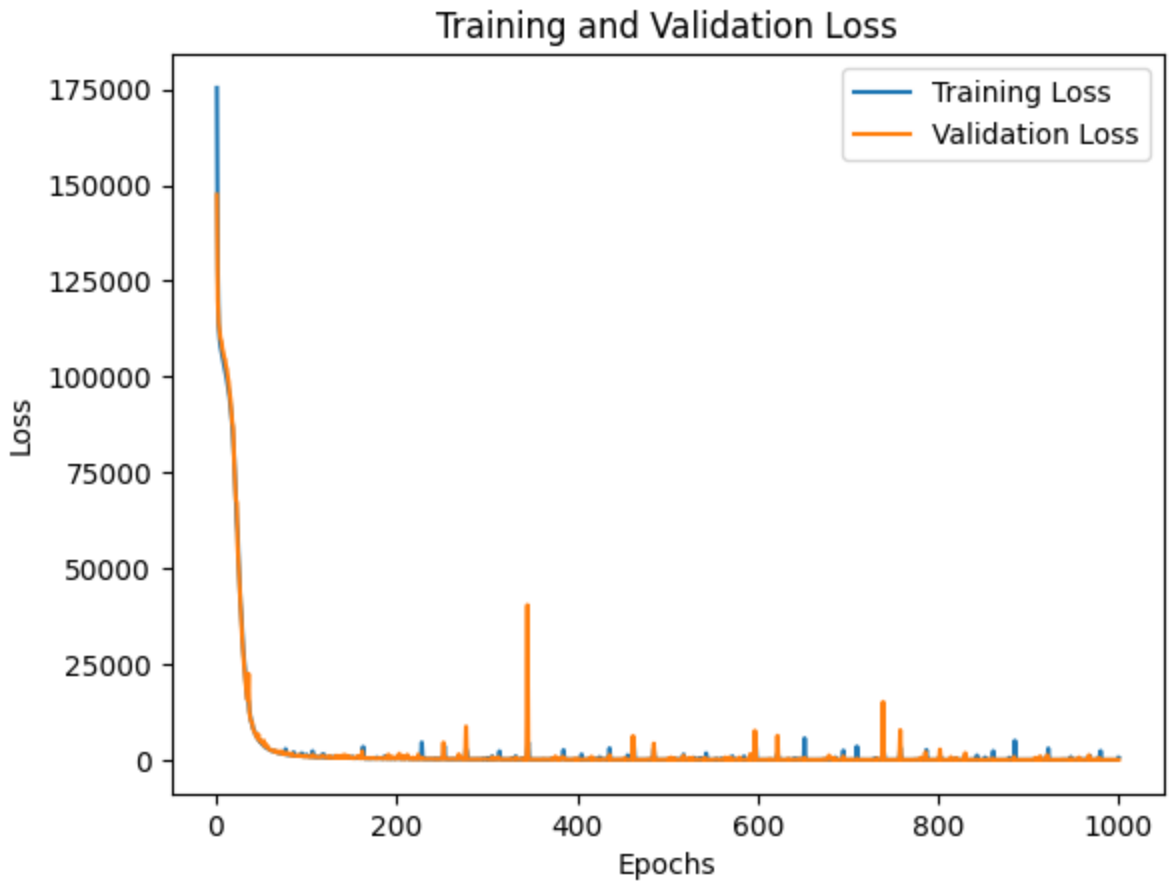
We chose learning rate as 1e-4, epochs as 1000 as it took a lot of time to converge and there was a lot of oscillations in loss vs epoch graphs. Batch size 1024 was taken because we have 90000 data and we wanted faster compute with accurate gradients.

MLP had hidden layer dimension unit as an hyper parameter.
The **embedding size** was chosen to be 16 since we only have 13 characters, just a bit more complicated than one-hot encoding.
The **num of hidden layers** was a hyper parameter which we selected using grid search for both rnn and transformer.
Hidden size for RNN was also selected using gridsearch and best value was 64.
Additionally transformers had **num of multiattention heads(Nhead)** and **depth of model** which is the same as embedding size**. Nhead** control the self-attention unit , choice was between 2,4 and 8 since our sequence task is not complicated and only length is 7.

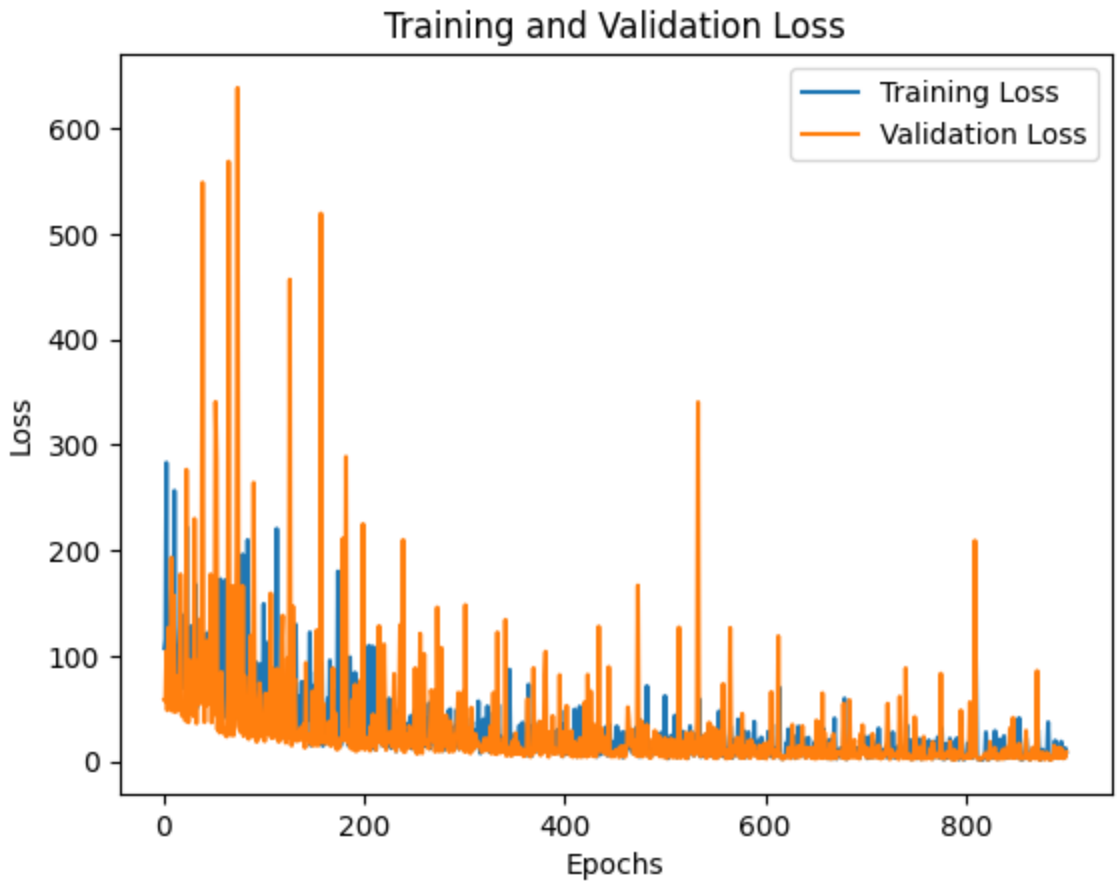
B.
The approach for selecting hyperparameters for MLP and Rnn+transformers were entirely different.
Since MLP is not very complicated , we used the grid search to identify the ideal learning rate and the batch size. The ideal was lr=1e-3 and batch size as 128 The number of epochs needed were smaller as it able to converge around 500. We used the grid search to find the hidden unit dimension between 64 - 1024 in powers of 2. Least complicated model performed the best with 128 as hidden unit.

For the RNN and transformer, the most important part was to chose the embedding size. Higher embedding vector complicated the model way too much and slowed down with much returns. A embedding dimension of 16 was chosen which is much closer to 13 as it will be somewhat similar to one hot encoding vector and thus would be able to capture the variation appropriately. These two complicated models took a lot of time to converge hence 1000 epoch was chosen with learning rate as 1e-4 due to heavy oscillations in lower learning rate in epoch vs loss graphs
The batch size was chosen to be 1024 to have faster convergence and larger steps while increasing computing speed. Since we had 90000 as data size , we dont need to go with smaller batches. Hence 1024 was chosen so that we could have 50-60 batches in training.
Next for rnn we needed to chose the num_layers and hidden_size. We used the grid search to compute these two hyperparameters. Since this was a less complicated model we chose 2-8 as hidden layers and 32-128 for hidden size.
For transformers , we needed to chose the num_layers and the number of multiattention heads. Again we used grid search to find the optimal values.

C.
The mlp training/validation loss vs epochs

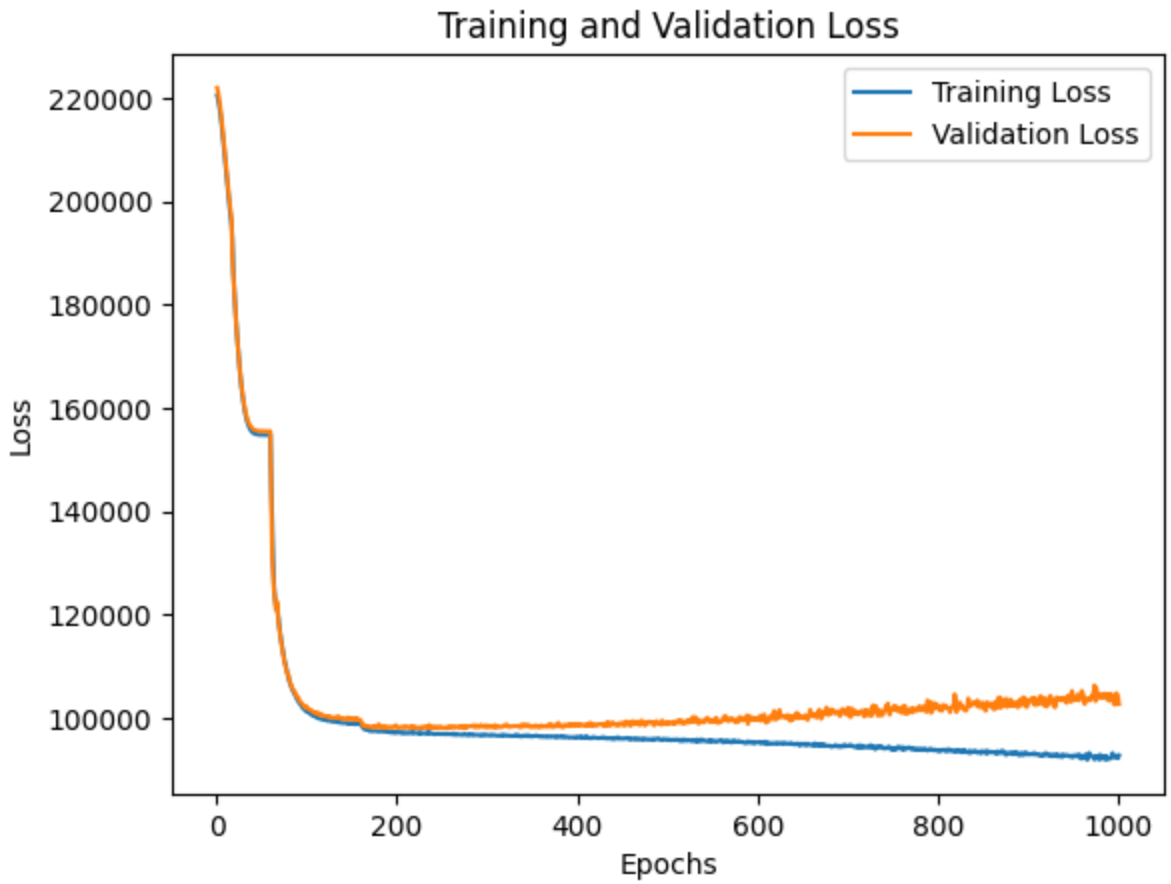


The rnn training/validation loss vs epochs, first 100 loss clipped



The transformer training/validation loss vs epochs

As you can see , the transformers performed the worst out of all the models. I suspect, even 1000 epochs werent enough to converge. The best loss was generated by RNN and MLP. although for RNN we had a lot of oscillations in the loss , hence after reducing the learning rate, we got more smoother curve.



Hence we chose RNN as the best model since the validation loss was converging to zero. The hyperparameter chosen were

```
# best model = RNN
# with hyperparameters
# num_epochs = 1000
# batch_size = 1024
# learning_rate = 1e-5
# hidden_size = 64
# num_layers = 4
# embedding_size=16
```

D

The best MSE loss for test came out to be 0.97 for RNN model.