COL774 Assignment - 2.1

Neural Networks

Divam Manchanda - 2022ME21336

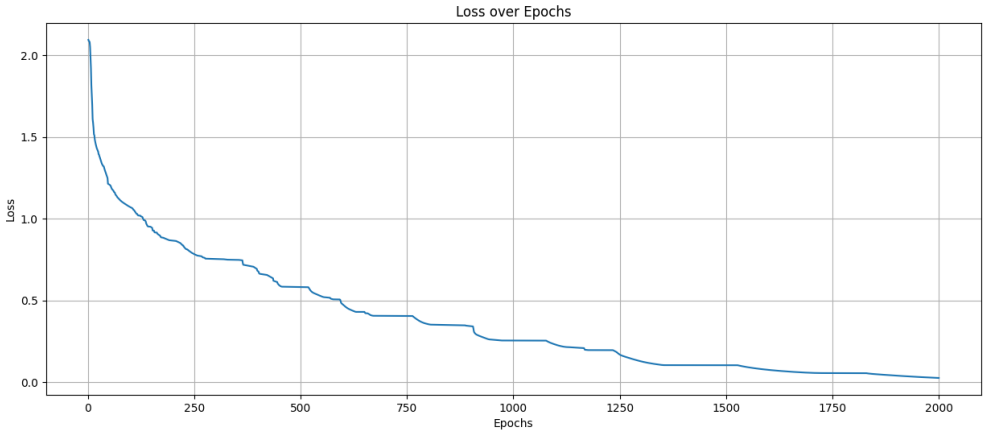
Yash Bansal - 2022CS51133

part c) Experimentation

Upon running RMSprop and momentum algorithms, we were able to achieve a minimum loss >2 for any hyperparameters that we experimented with; this was significantly surpassed by Adam Optimizer as even the untuned default parameters gave a loss under 2.

Hence, we extensively experimented with the hyperparameters for the Adam optimizer.

The highest-performing model obtained by us:-



Adam learning rates -

Initial - 0.001

after 500 epochs = 0.0005

after 1000 epochs = 0.0003

after 1500 epochs = 0.0001

batch size = 100

obtained a training loss of 0.07

The Process of getting to the model

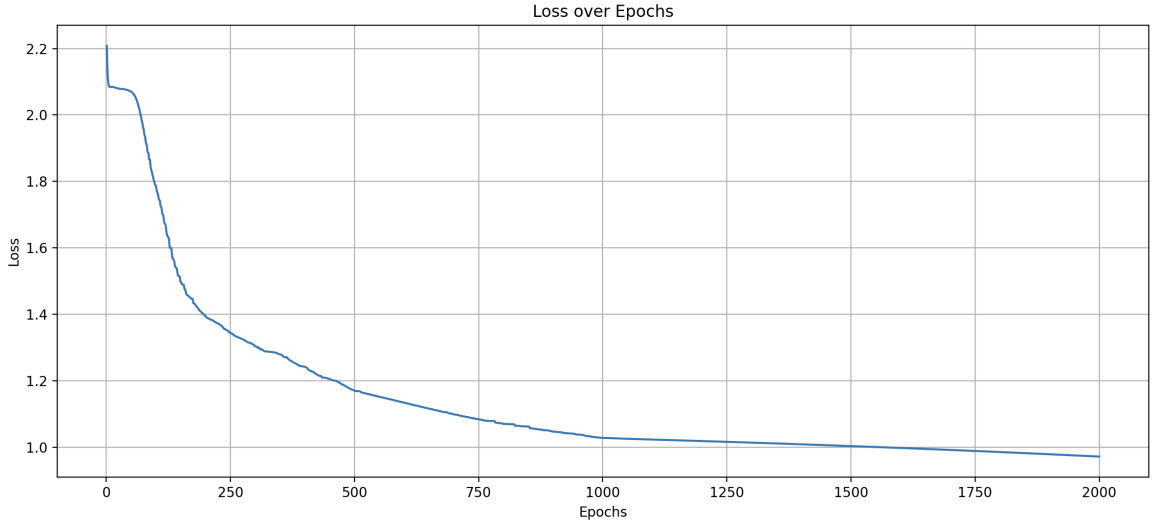
It was observed that :-

- Lower batch sizes give higher convergence per epoch
- A higher batch size gives a lower time per epoch

i.e., lowering batch size makes a single epoch run longer but then converges at a higher rate as well.

First, we stuck to full batch training to find the optimal parameters for the Adam optimizer, and then, with those parameters, tested for multiple batch sizes, where the optimal value for batch size was found to be 100.

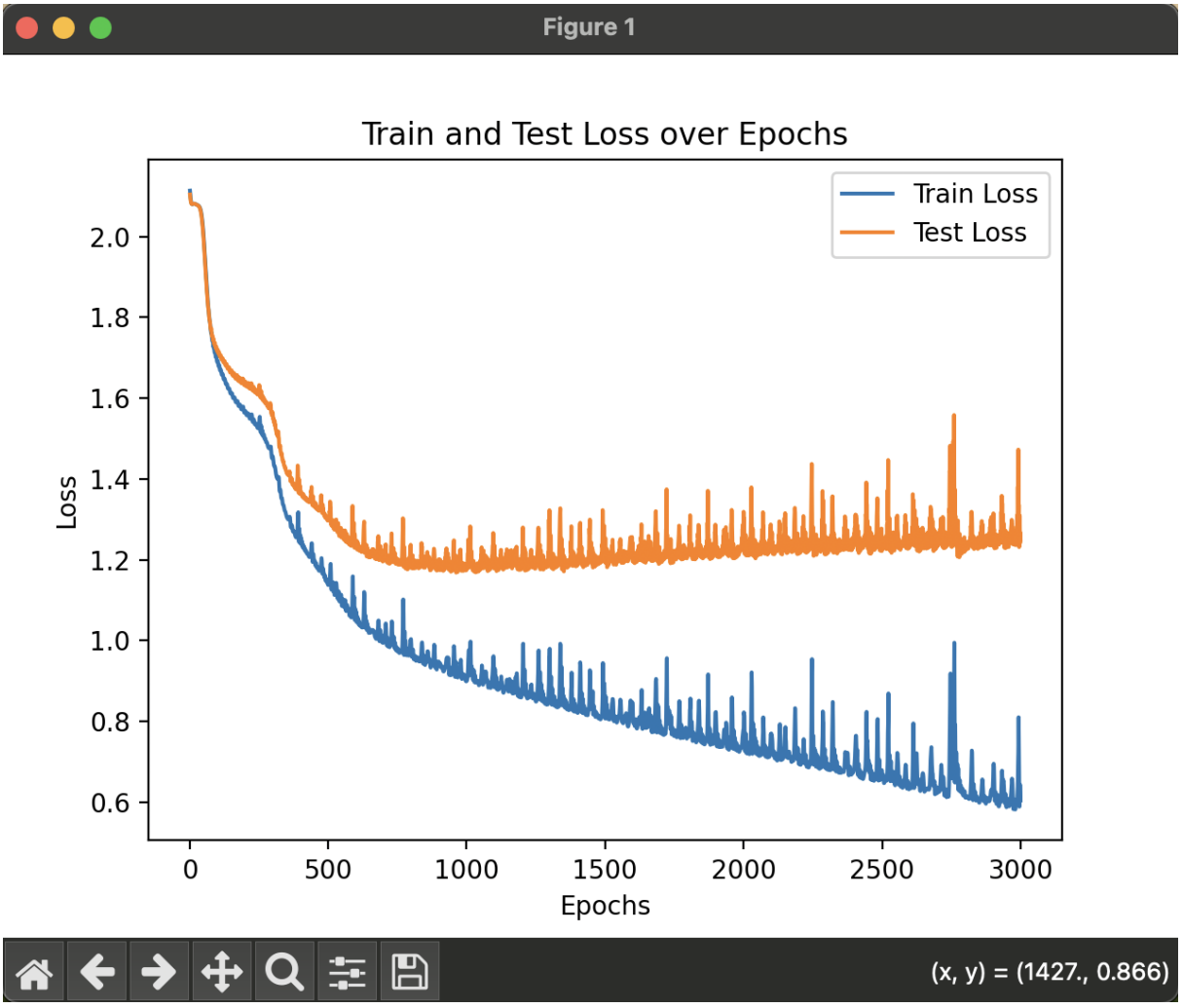
Following are the loss vs epoch graphs, all trained on a full batch size of 3200, for varying learning rates of the Adam optimizer



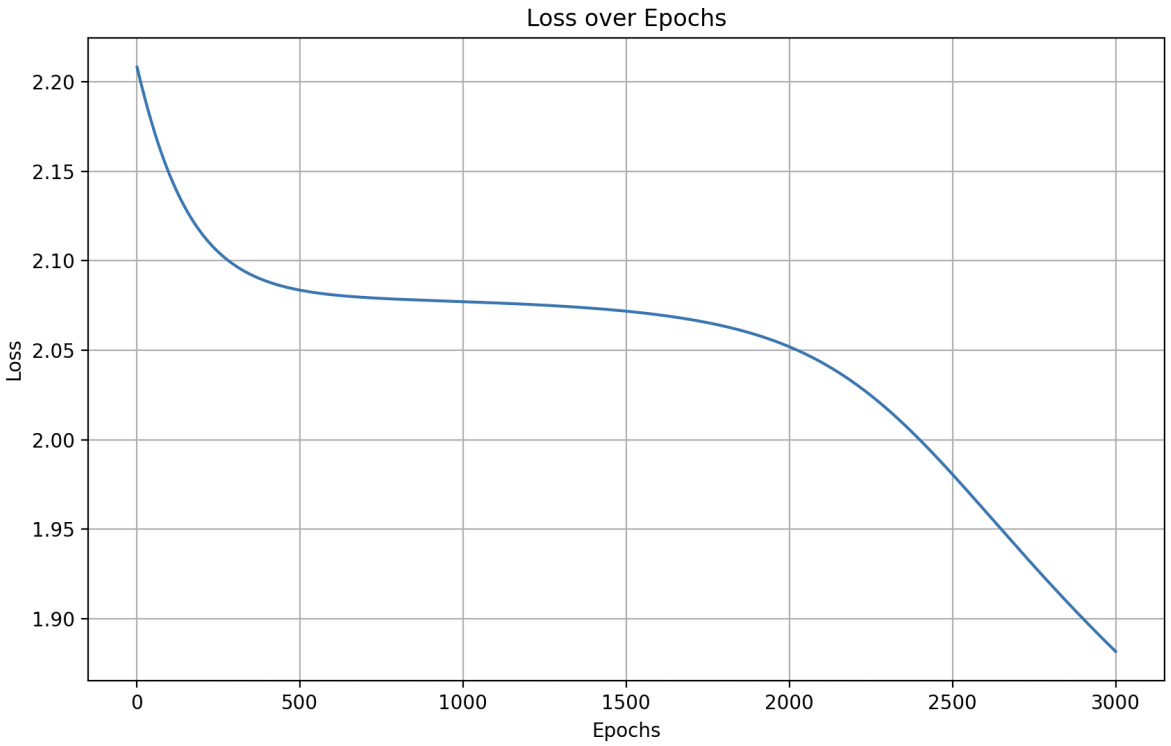
Adam learning rate -

initial 0.001

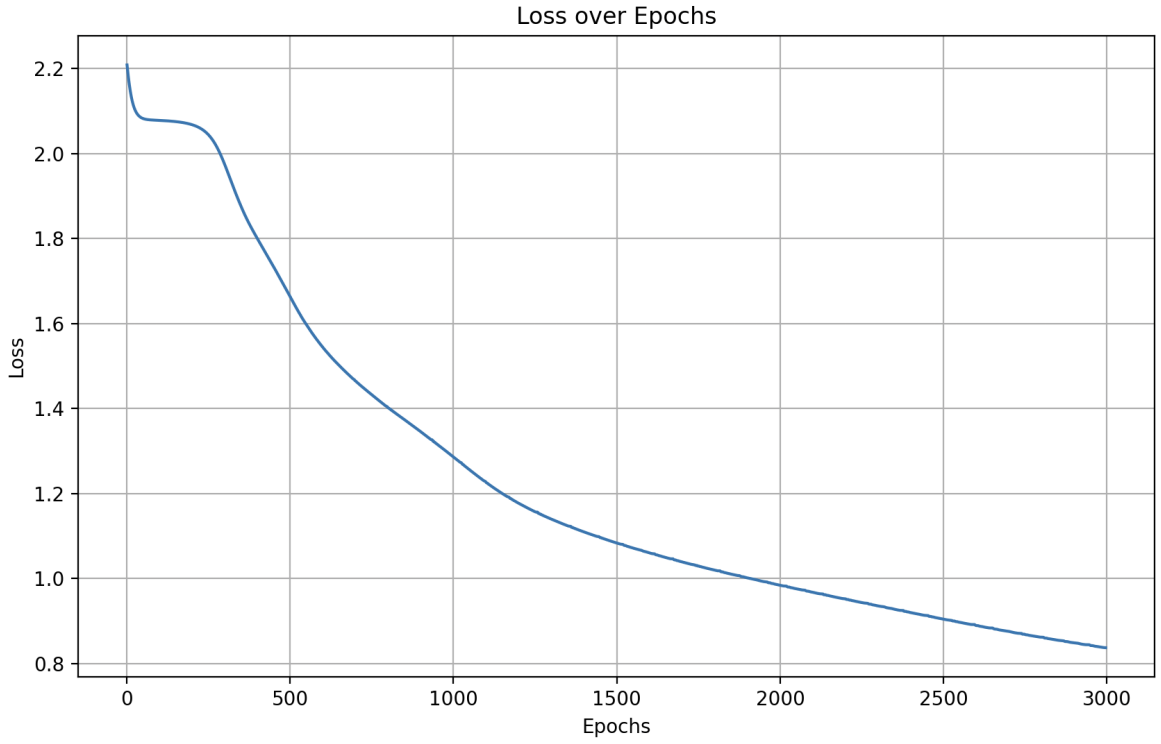
then 0.0001 after 1000 epochs



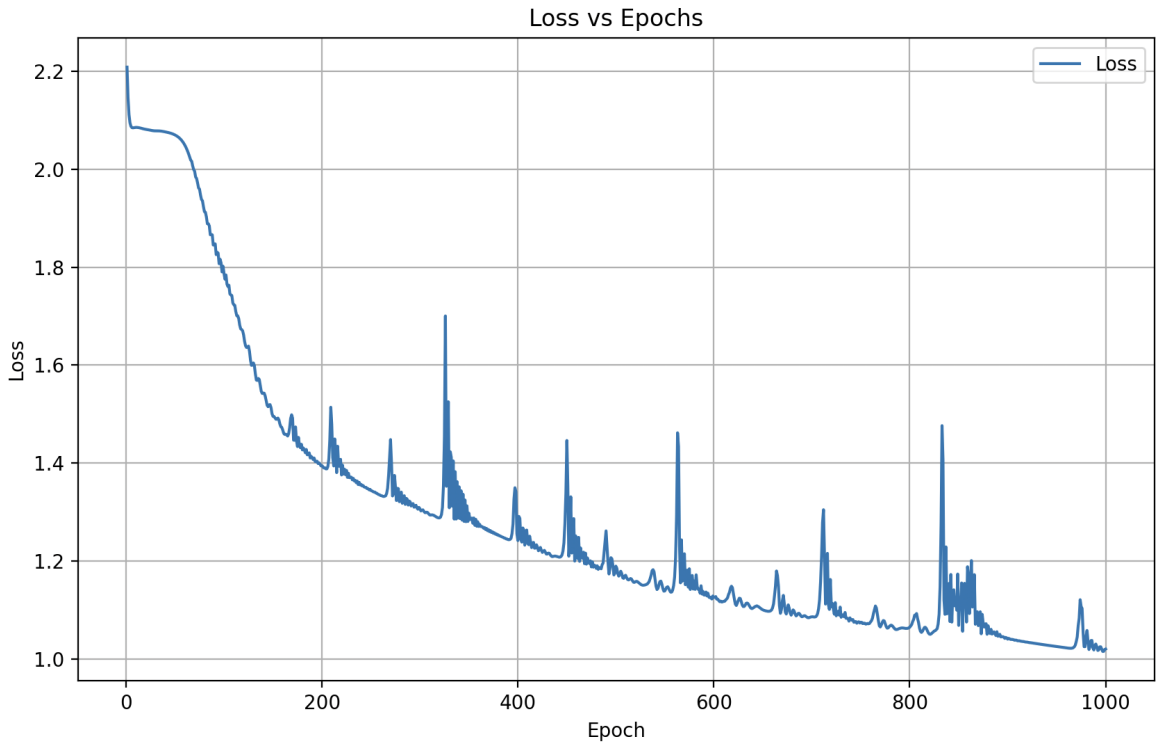
Train and test loss for default adam inbuilt in PyTorch.



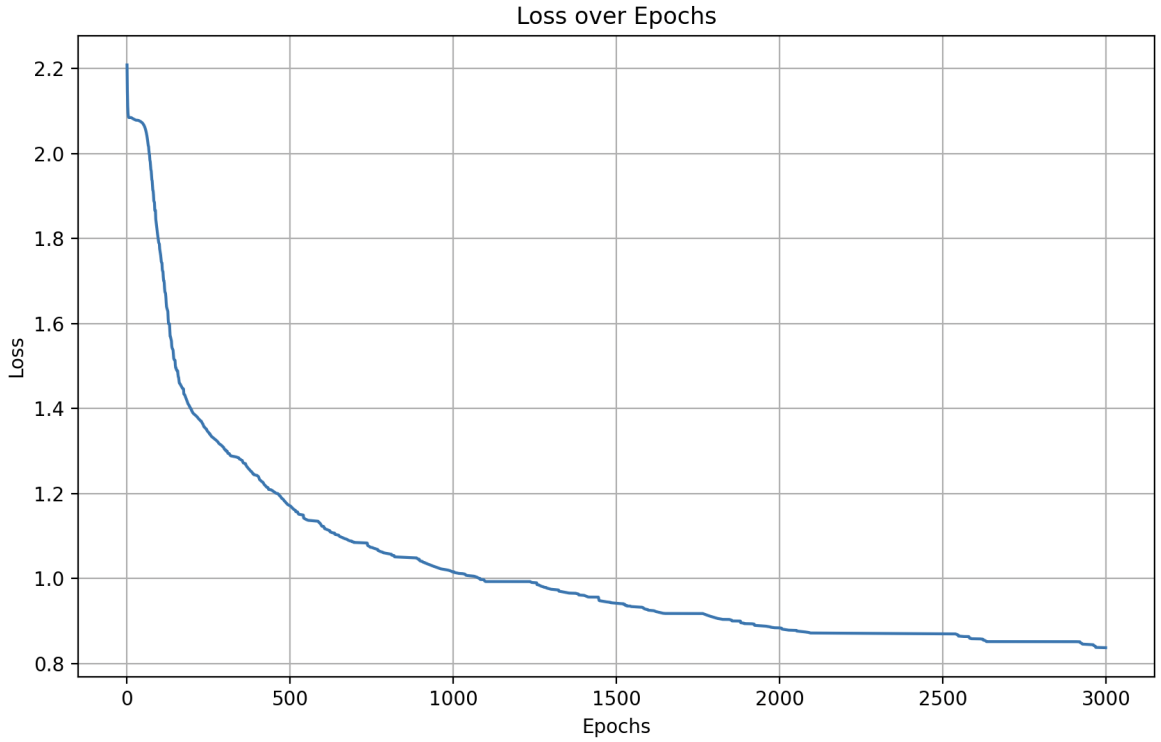
adam learning rate = 0.00001



adams learning rate = 0.0001



adams learning rate =0.001



train_loss_adams_truncated for learning rate = 0.001

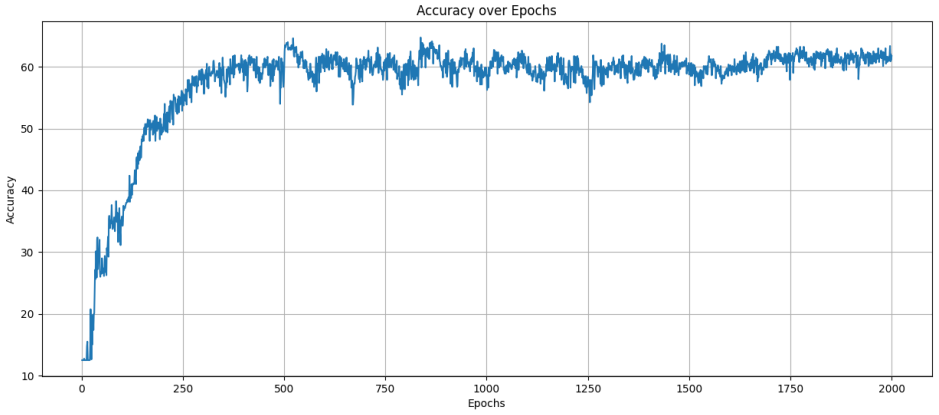
general observations -

- Higher learning rates give optimal faster initial convergence but then greater instability and fluctuation later hence, scheduling has been down where alpha decreases with epochs, therefore obtaining a faster convergence in the earlier stages and good stability in the later stage of the training phase
- Therefore, in the final model, these were kept to be:
  - Initial - 0.001
  - after 500 epochs = 0.0005
  - after 1000 epochs = 0.0003
  - after 1500 epochs = 0.0001
- Again, as mentioned earlier, batch sizes were varied for the above learning rates, and 100 was seen to perform best.
- The optimal loss and corresponding loss vs epoch graph obtained for these values given above is the first graph in this report

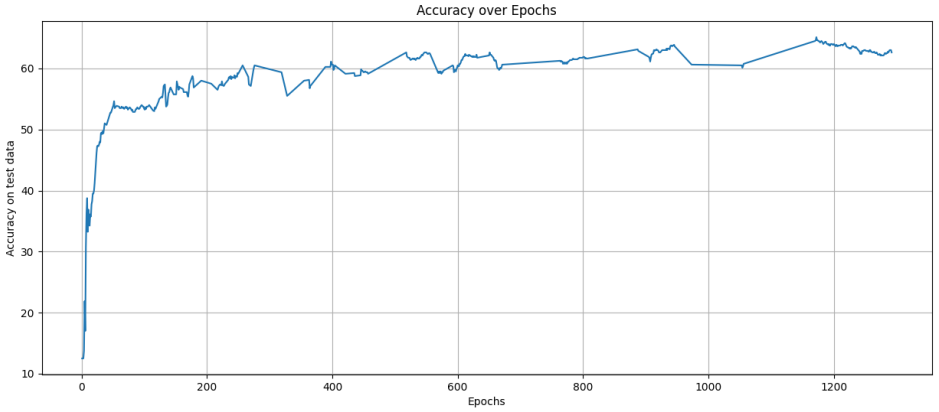
PS: we also tried shuffling the training data, but no improvements were obtained by doing so

part d)

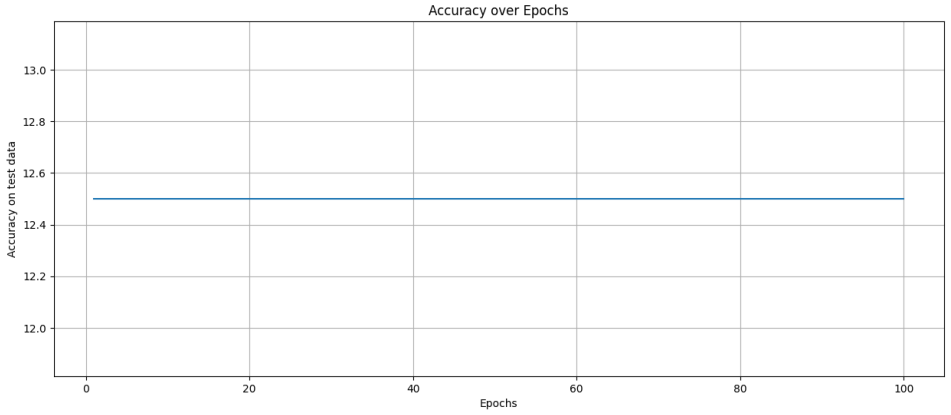
- In the 15-minute time constraint, we could successfully run only 1500 epochs
- We initially introduced a single new hidden lair in our neural network, with the same activations as in part c, and no increase in performance received therefore, the general architecture was kept the same as that of part c
- Next, we changed the activations for all nodes, and received the following results:



activation: leaky RELU - max accuracy around 55-60 around 1500 epochs



activation: sigmoid - crossed accuracy of 60% in only 1200 epochs



Activation: RELU - suffers vanishing gradients, as can be seen above

Therefore, the optimal activation giving the highest accuracy (64%) - sigmoid

finally, we experimented with regularization: by adding an L2 norm of weights in our cross-entropy loss, maximum accuracy obtained with the same algorithms on the test set was found to be ~55%; hence, regularization was dropped, and early stopping was implemented.

It can be seen above in part c) that up until 1500 epochs, overfitting is not prevalent in the model and also that the train set has achieved its minima already in all cases.

Hence, stopping at the 1500th epoch, (which takes up the entire computing time), is able to return values close to the minima

(given that in part d) graphs above, the test accuracy is still increasing in the neighborhood of 1500 epochs; we receive further evidence that overfitting hasn't occurred on the training set and that running 1500 epochs is sufficient early stopping)

Ideal optimizers and their corresponding hyperparameters are the same as part c) since they gave us the highest outputs we observed.

The final best accuracy obtained in part d after plugging in the above parameters ~64%.