**Neural Network Report**

We experimented with 4 different types of networks:

1. 2 hidden layers with tanh as the activation
2. 2 hidden layers with sigmoid as the activation
3. 4 hidden layers with tanh as the activation
4. A NN as a combination of 123

This is the table of our results: [data](https://docs.google.com/spreadsheets/d/1ALcfI8z1_VICpVGFQuyhRFRaMKbaIsEjxwqmSBat23o/edit?usp=sharing)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Averages of | Training Accuracy | Training F-score | Test Accuracy | Test F-score |
| Network 1 | 0.8030 | 0.8051 | 0.8158 | 0.8265 |
| Network 2 | 0.7460 | 0.7456 | 0.7658 | 0.7658 |
| Network 3 | 0.8340 | 0.8484 | 0.8325 | 0.8366 |
| Network 4 | 0.8312 | 0.8336 | 0.8333 | 0.8302 |
| Best (N3) | 0.8474 | 0.8532 | 0.85 | 0.8543 |

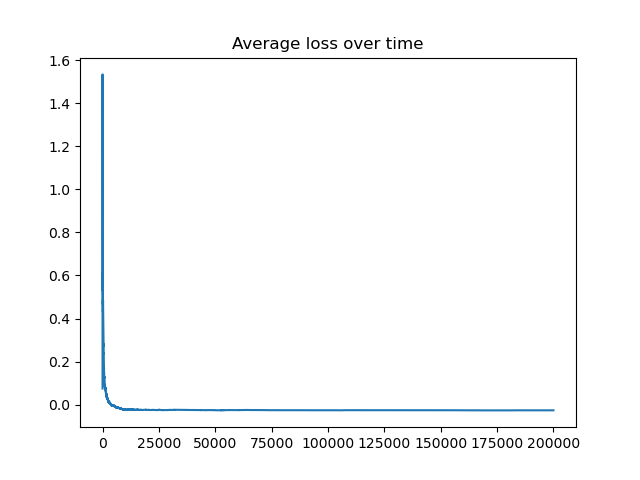
We thought that Network4 would correct the mistakes of these 3 and would produce a better result. But, we can see that there isn't much difference from Network3 and it is pulling most of the weight. And we got the best individual performance from Network3 of 85% final accuracy. Therefore, we chose Network 3 as our final model.

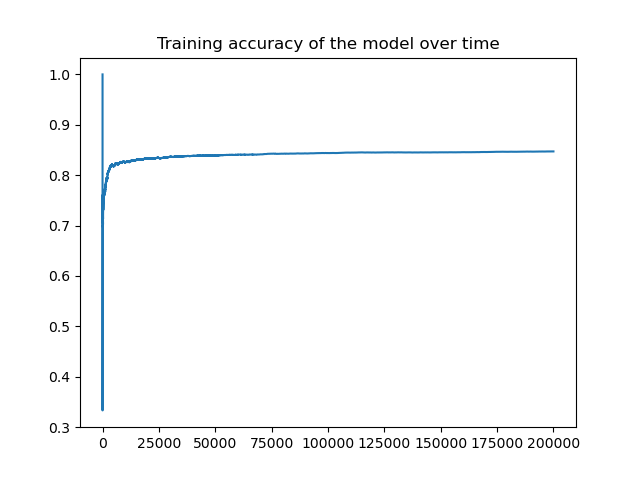
I did not have any experience in OOPs, so I wrote it with basic logic.

We found that the learning-rate of 0.0005 was giving the best results.

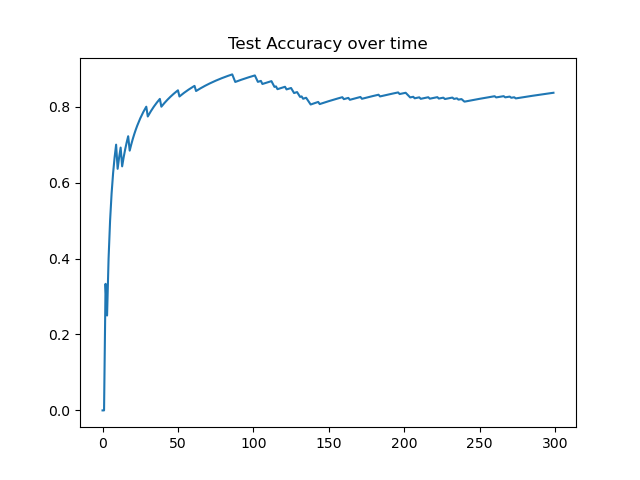
We were printing a graph of the avg prediction to confirm that the model was producing an average value of 0.5 .

I think our code did not reach accuracy over 90% because of some fault in our logic while coding the back-propagation part. It got really complicated with the matrix multiplication and we couldn’t go further.

We can clearly see that average loss rapidly approaches 0 and stays round it.



We can see that the training accuracy is approaching a limit at around 84% .



We split the data into 2 parts, 1160 for training and 300 for testing.

This is the test accuracy over time, which is around 83% .