Assignment 4: Neural Networks

1. Data processing:

I used the CSV format as it easier to work with and loaded the test set so that I can use it in place of the validation set for Hyper-parameter tuning validation accuracies.

2. Building and training a Baseline model

I designed the Network Architecture using my “NeuralNet” class as per the assignment specifications, but after choosing the hyperparameters I was going to tune, I made the number of neurons in the first layer be a variable so that it can be later changed during the Hyper-parameter experiments.

The same can be said for the **train_model** function it is designed in such a way that allows it to be re-used for later experiments, I believe this allows for a fair comparison between any model obtained from experimenting and the baseline model as they both would have gone through the same training and validation process and using the same random conditions such the “seed”.

The function has variable arguments such as “verbose”: for printing the progress for each epoch if set to True

and “return_history”: for returning the loss or accuracy history if it is set to True, otherwise only returning the final accuracy.
It trains for 10 epochs and takes the final accuracy as our baseline performance, in my case the final accuracy was also the best accuracy for the baseline model

The baseline final accuracy was 89.44%

3. Hyperparameter Optimization Experiment

The 3 Hyper-parameters choose to experiment with are:

○ learning rates = 0.005, 0.001, 0.0005

○ optimizers list = 'Adam', 'SGD', 'RMSprop','AdamW'

○ Number of neurons in the hidden layer one= 64, 256,512

Producing a total of 36 combinations/models: I chose these Hyper-parameters to account for not just the architectural element (number of neurons) of model building but also the optimization part of it via tuning the optimizers, so this choice reflects the balance of the two.

The list I chose for the number of neurons in the hidden layer one, was so that I could test both a number lower and higher than the number of neurons in the baseline model and then the addition of the 512 number tests for overfitting

<u>4. Part 4: Analysis and Report</u>

<u>Below are the results of the Hyperparameter Experiments using a Pandas data frame:</u>

```
Hyperparameter Experiment Results
*********************************************************
 learning_rate optimizer  neurons  accuracy
        0.0050      Adam       64     87.80
        0.0050      Adam      256     88.41
        0.0050      Adam      512     88.07
        0.0050       SGD       64     82.25
        0.0050       SGD      256     82.74
        0.0050       SGD      512     83.34
        0.0050   RMSprop       64     86.40
        0.0050   RMSprop      256     88.34
        0.0050   RMSprop      512     88.30
        0.0050     AdamW       64     87.54
        0.0050     AdamW      256     88.28
        0.0050     AdamW      512     88.23
        0.0010      Adam       64     88.73
        0.0010      Adam      256     89.30
        0.0010      Adam      512     89.37
        0.0010       SGD       64     67.76
        0.0010       SGD      256     68.39
        0.0010       SGD      512     68.34
        0.0010   RMSprop       64     88.92
        0.0010   RMSprop      256     88.98
        0.0010   RMSprop      512     89.43
        0.0010     AdamW       64     88.60
        0.0010     AdamW      256     89.32
        0.0010     AdamW      512     89.32
        0.0005      Adam       64     88.02
        0.0005      Adam      256     89.33
        0.0005      Adam      512     89.39
        0.0005       SGD       64     60.29
        0.0005       SGD      256     61.30
        0.0005       SGD      512     61.91
        0.0005   RMSprop       64     87.96
        0.0005   RMSprop      256     89.41
        0.0005   RMSprop      512     89.84
        0.0005     AdamW       64     88.16
        0.0005     AdamW      256     88.72
        0.0005     AdamW      512     89.52
*********************************************************
```

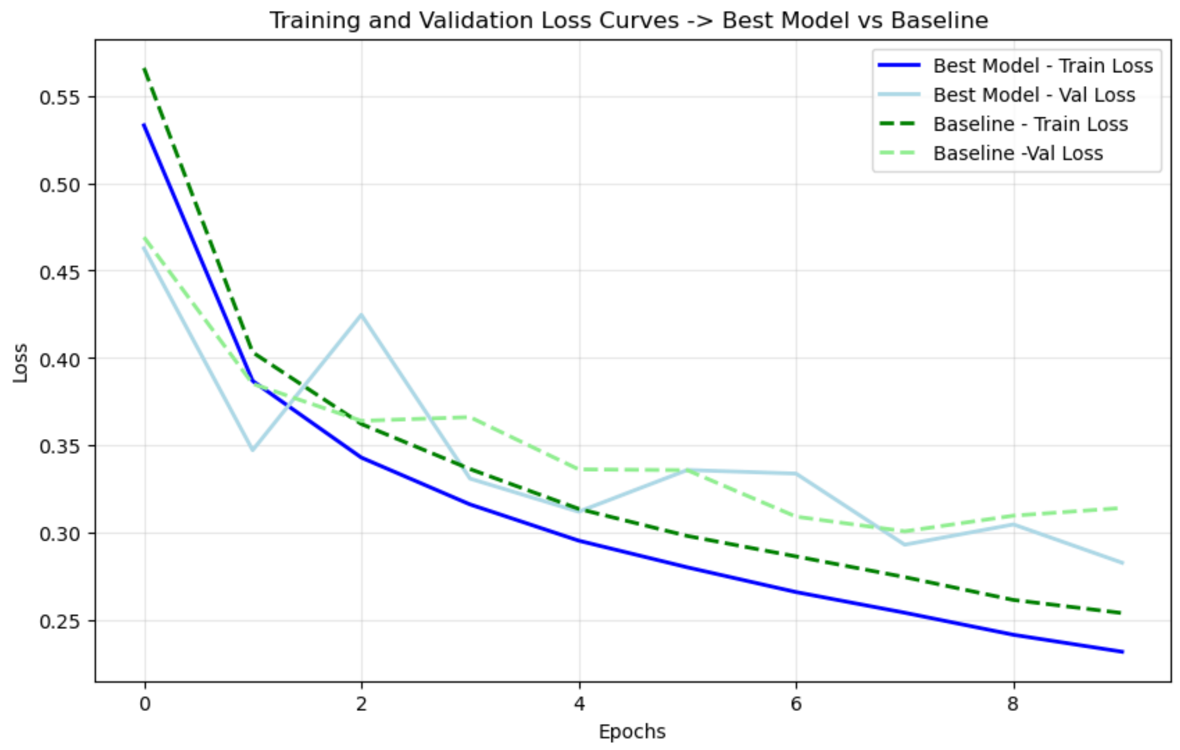
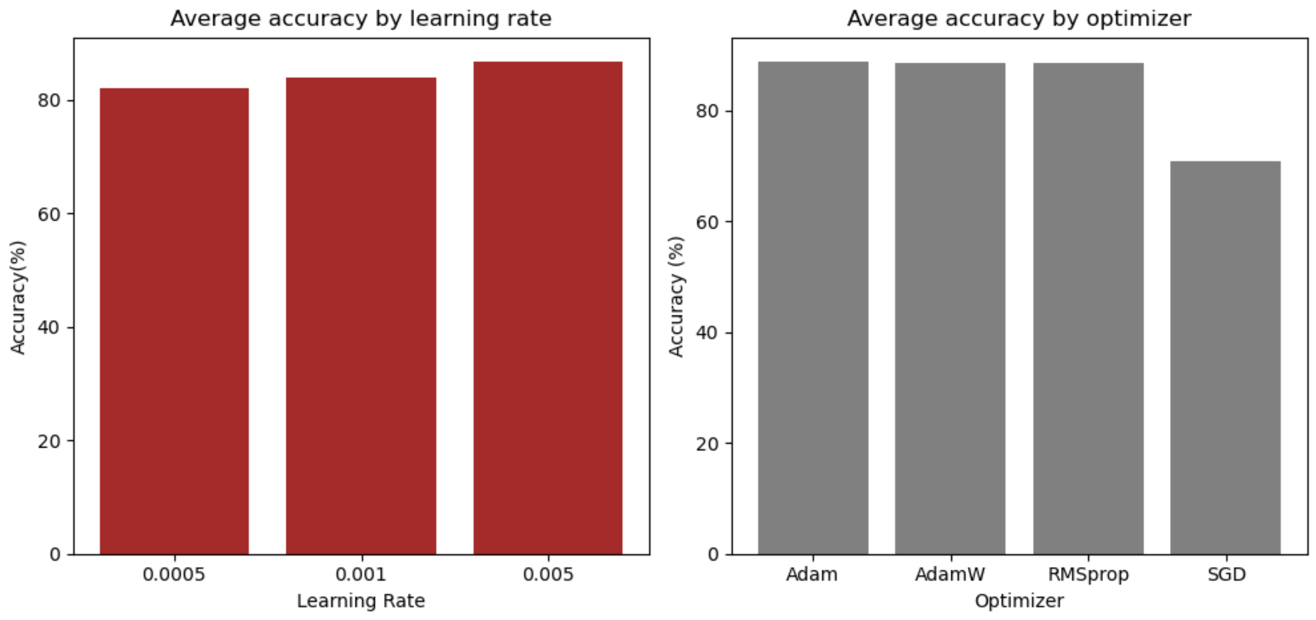


Figure A: Training & Validation loss curves for Baseline and Best overall model



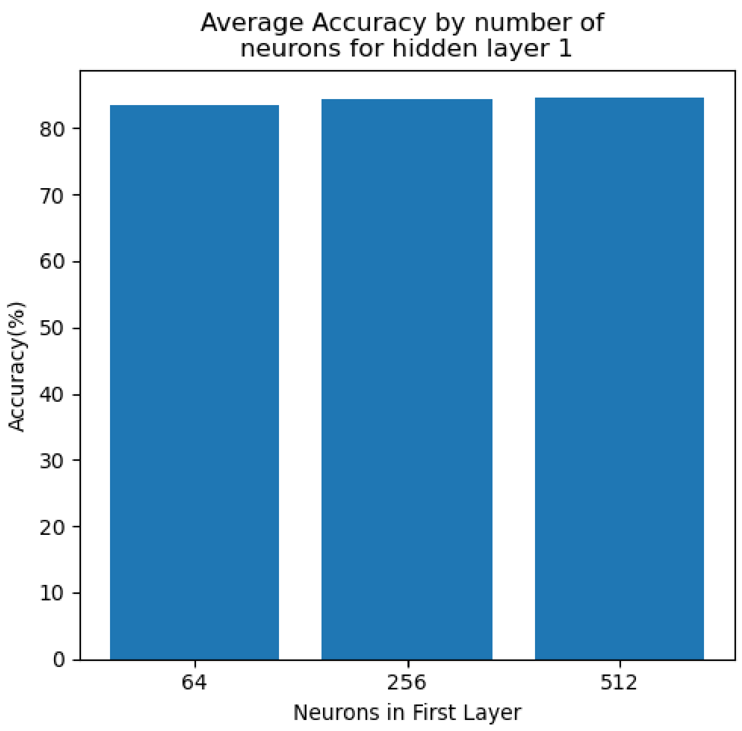


Figure B: Bar Charts that compares the performance of the different hyperparameter combinations by plotting the average accuracy by each hyperparameter setting

● Interpreting the results:

■ The combination of hyperparameters that produced the best model is as follows:

Learning rate = 0.0005, Optimizer = RMSprop and hidden layer neurons =512 with an accuracy of 89.84%. This model saw an improvement of 0.40% from the baseline accuracy

■ Hyperparameter Influence: (Please refer to both Figure B and Figure D (in the appendix))

a)The Hyperparameter that impacted performance the most was the learning rate, I learnt this from doing two different experiments, in the Appendix in figure D we can see that not only does the learning rate 0.1 perform poorly but the Adam and MSprop Optimizers have lower average accuracies in comparison to the SGD optimizer in response to using relatively bigger learning rates of 0.1 and 0.01

But this all changes after I tested for finer/smaller learning rates such as those in figure B in which all the learning rates have an average accuracy of 80+ and in response all the Optimizers except for the SGD also have very high average accuracies

It seems the learning rate also serves as a dependent variable for other fellow hyperparameters. Secondary to it is the optimizer Hyperparameter.

b) smaller learning rates even if it slows down training, offers greater stability and a greater chance of reaching the optimal performance. When I used larger learning rates, even my best model could not out-perform the baseline model. (see figure E in Appendix page8)

-> The adaptive optimizers such as RMSprop and Adam perform significantly better when using smaller learning rates because they dynamically adjust per parameter learning rate which lead to more stable convergence unlike the stiff SGD optimizer which when using smaller learning rates suffered as it got slower overall.

■ <u>Overfitting Analysis:</u> (Please refer to figure A in page 3)

The best model we got from hyperparameter tuning did not mitigate overfitting, because best model presents a larger gap between the training loss and validation loss curves than the baseline model and this means the best model has more overfitting than the baseline model even though the best model does have a lower validation loss in the early and late epochs of training .

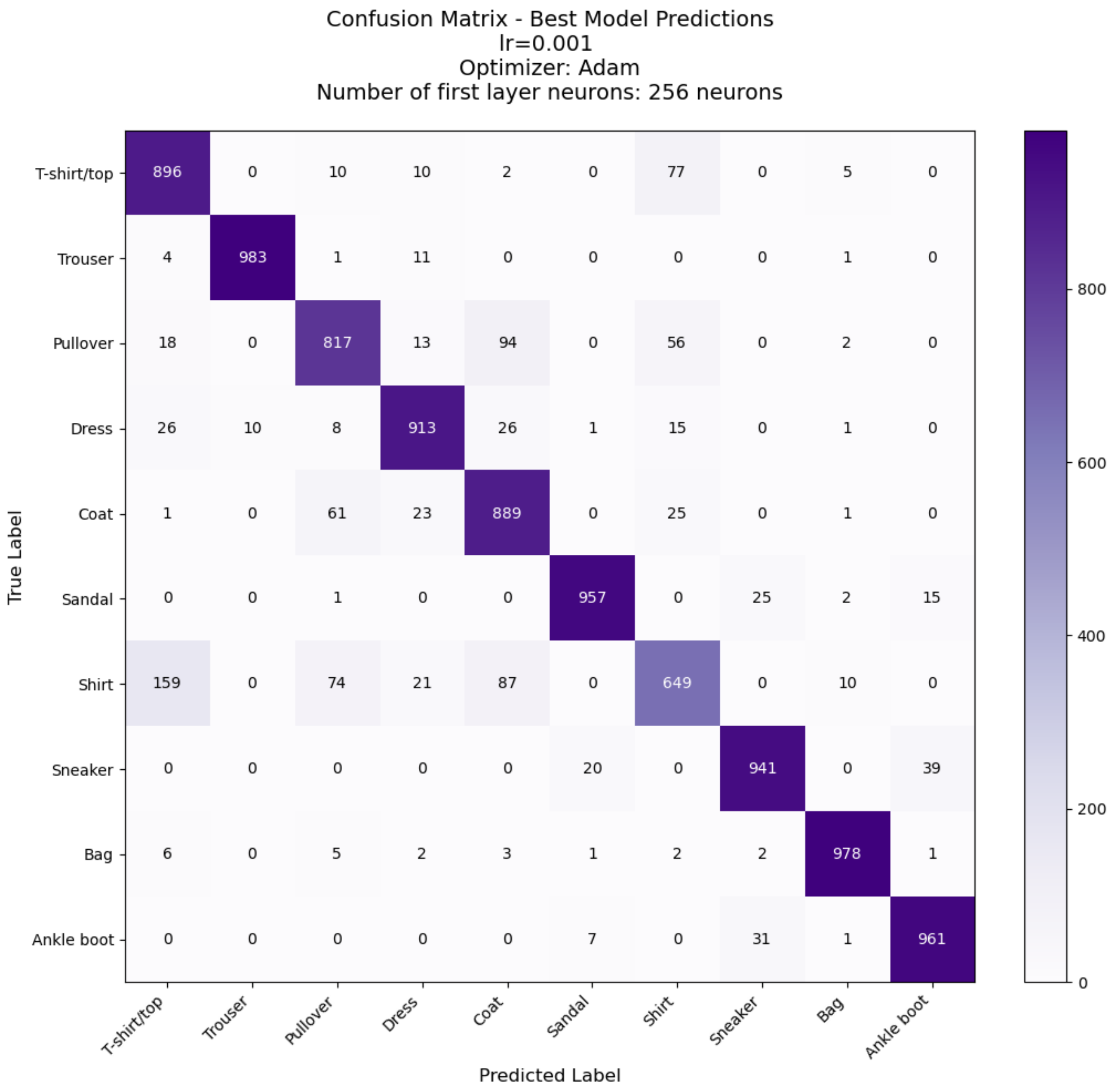
This could be due to the best model having 512 neurons for hidden layer one; this means the model may be over-parametrized is thus prone to overfitting.

■ <u>Error Analysis: (</u>Confusion matrix in the Appendix in figure C)

The two classes that are frequently confused are the T-shirt/top and the shirt classes; this isn't too surprising as the two clothing materials share similar structure as upper body clothes ,the model may find it hard to differentiate between the two as the visual differences such as collar structures ,or buttons are nuanced

In some cases, there is no visual difference, with the only difference between the two items being texture which isn't an easy feature to rely on for discriminating between the two.

Appendix:



True Label \ Predicted Label	T-shirt/top	Trouser	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankle boot
T-shirt/top	896	0	10	10	2	0	77	0	5	0
Trouser	4	983	1	11	0	0	0	0	1	0
Pullover	18	0	817	13	94	0	56	0	2	0
Dress	26	10	8	913	26	1	15	0	1	0
Coat	1	0	61	23	889	0	25	0	1	0
Sandal	0	0	1	0	0	957	0	25	2	15
Shirt	159	0	74	21	87	0	649	0	10	0
Sneaker	0	0	0	0	0	20	0	941	0	39
Bag	6	0	5	2	3	1	2	2	978	1
Ankle boot	0	0	0	0	0	7	0	31	1	961

Figure C: Confusion matrix from my best model as obtained from hyperparameter tuning experiments

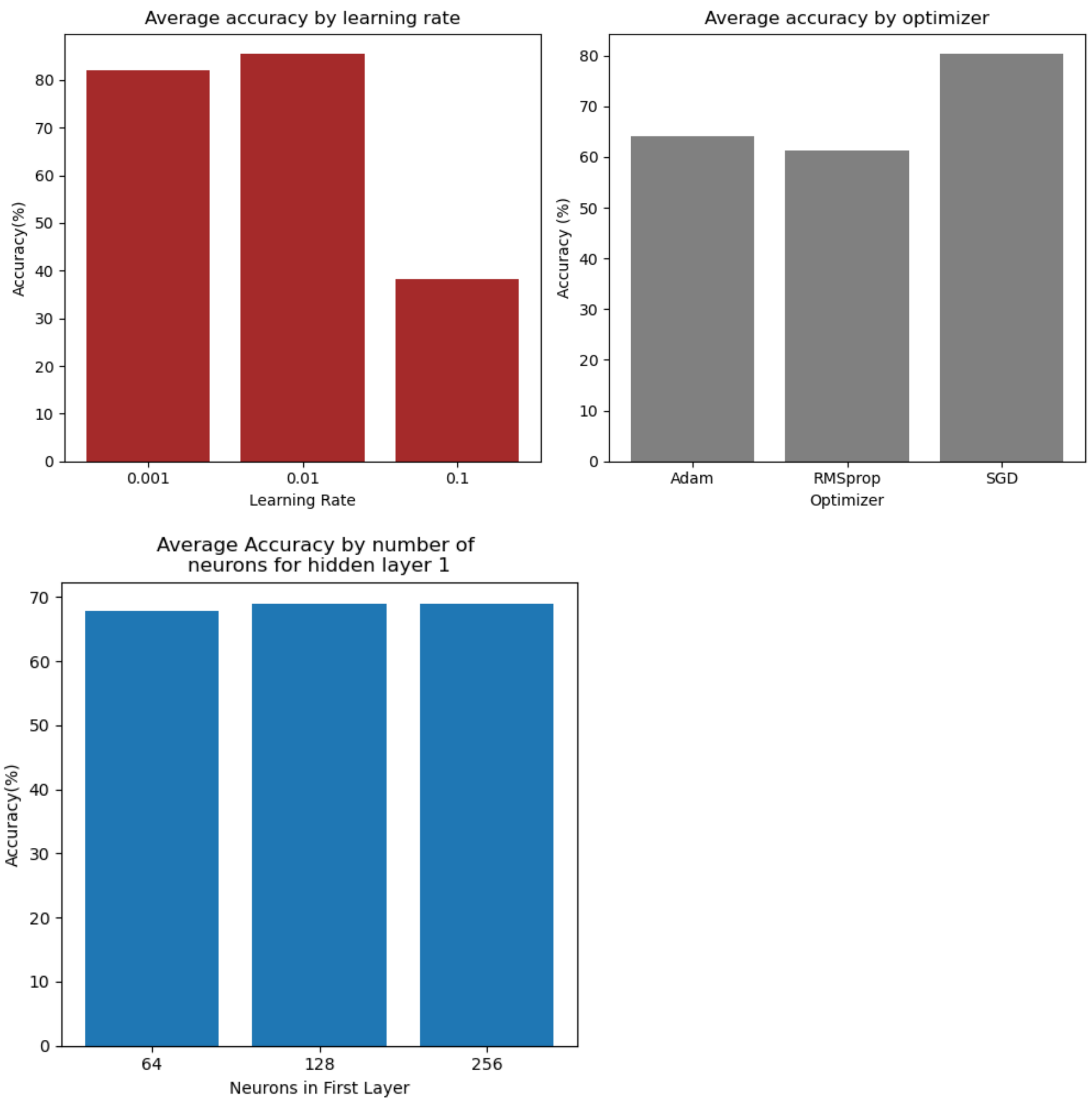
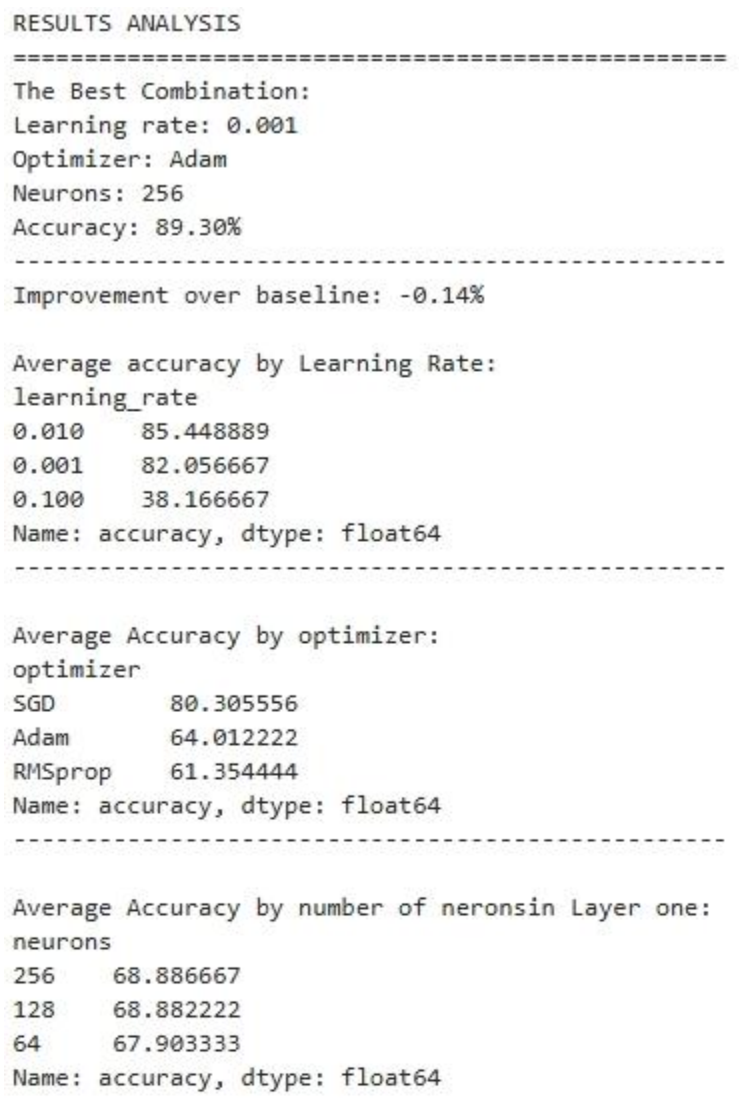


Figure D: Bar Charts that compares the performance of the different hyperparameter combinations by plotting the average accuracy by each hyperparameter setting (from my early experiments using the bigger learning rates: 0.001, 0.01 & 0.1)



```
RESULTS ANALYSIS
==================================================
The Best Combination:
Learning rate: 0.001
Optimizer: Adam
Neurons: 256
Accuracy: 89.30%
--------------------------------------------------
Improvement over baseline: -0.14%

Average accuracy by Learning Rate:
learning_rate
0.010    85.448889
0.001    82.056667
0.100    38.166667
Name: accuracy, dtype: float64
--------------------------------------------------

Average Accuracy by optimizer:
optimizer
SGD        80.305556
Adam       64.012222
RMSprop    61.354444
Name: accuracy, dtype: float64
--------------------------------------------------

Average Accuracy by number of neronsin Layer one:
neurons
256    68.886667
128    68.882222
64     67.903333
Name: accuracy, dtype: float64
```

Figure E: Results from my early experiments of using bigger learning rates (resulting in a best model that was still not performing as good as the baseline model)