Assignment 1
Linear and Logistic Regression

Linear Regression

1. **Explanation:**
    a. Solves a regression problem (i.e maps input features to a continuous output scalar, through generating a polynomial f(Xs) that fits the data)
2. **Hyperparameters**
    a. Learning rate
    b. Optimizer used
    c. Epochs (not tuned but we could have tuned it)
3. **Loss functions**
    a. We can not use the “Accuracy” metrics for measuring Linear regression performance as Linear regression uses linear activation function with only 1 neuron.
    b. Metric used is the mean squared error.
4.

Optm	RMSprop			Adagrad			Adam		
LR	0.1	0.01	0.001	0.1	0.01	0.001	0.1	0.01	0.001
Val Loss	3410	3600	3606	3648	440	153	3464	3728	3717
Figure	fig_R1	fig_R2	fig_R3	fig_G1	fig_G2	fig_G3	fig_A1	fig_A2	fig_A3

Optm	SGD	
LR	0.01	0.001

Mmtm	-	0.01	0.001	-	0.01	0.001
Val Loss	5.8	6.002	5.7	6.32	6.25	6.9
Train loss	0.0124	0.0102	0.0101			
Figure	fig S1	fig S2	fig S3	fig S4	fig S5	fig S6

5. Problems in Models

a. Overfitting: (RMSprop, Adagrad, Adam)

solved by early stopping (i.e decreasing the number of epochs) or by reducing the model complexity either by removing some layer/neurons -which doesn't apply in our case- or by adding dropout in features layer. Also we can add more data to our model. Other methods exist that don't apply to 1 neuron case.

b. Underfitting:

Solved by increasing the model complexity by adding more neurons/layers -which doesn't apply to our case-, or by transforming the model and increasing the space dimension (i.e adding X^2 , Cos(X), etc), and finally we can increase the number of epochs.

c. Overshooting:

Overshooting the optimal value of the parameters is solved by decreasing the learning rate.

Logistic Regression

1. **Explanation:**
    a. Solves a regression problem (i.e maps input features to a continuous output scalar, through generating a polynomial f(Xs) that fits the data)

2. **Hyperparameters**
    a. Learning rate
        i. 0.1, 0.01, 0.001
    b. Optimizer
        i. SGD, RMSprop, Adagrad, Adam
    c. Pre-processing
        i. MinMaxScalar, StandardScaler
    d. Parameter Initializers
        i. RandomNormal, RandomUniform
    e. Loss function
        i. Binary- Cross Entropy, Categorical Hinge, Mean Squared Error
    f. Epochs
        i. 500, 1000
    g. Batch_size (not tuned but we could have tuned it)
    h. Artificial features with and without interaction_only.

3. **This is the data collected, this part could be skipped since the analysis part contains the important details of these tests.**

    (A total of 132 tests were executed)

OPT	SGD							
LR	0.01				0.001			
PI	RU		RN		RU		RN	
Epochs	500	1000	500	1000	500	1000	500	1000
Val Acc-B	0.7895	0.7895	0.7895	0.7895	0.8158	0.7895	0.8158	0.7895
Val Acc-C	0.8421	0.8947	0.8421	0.8947	0.6842	0.5526	0.5263	0.7895

Val Acc-M	0.7895	0.7895	0.7895	0.7895	0.7632	0.8158	0.5263	0.7895
Train Acc-B	0.8502	0.8502	0.8458	0.8502	0.8414	0.8414	0.8458	0.8414
Train Acc-C	0.8722	0.8899	0.8767	0.8899	0.6652	0.5374	0.5374	0.8414
Train Acc-M	0.8458	0.8546	0.8414	0.8546	0.7797	0.8370	0.5374	0.8458
Test Acc-B	0.87	0.87	0.87	0.87	0.84	0.87	0.84	0.87
Test Acc-C	0.87	0.87	0.87	0.87	0.66	0.61	0.61	0.60
Test Acc-M	0.87	0.87	0.87	0.87	0.79	0.84	0.61	0.84
Figure	fig 1	fig 2	fig 3	fig 4	fig 5	fig 6	fig 7	fig 8

OPT	RMSProp											
LR	0.1				0.01				0.001			
PI	RN		RU		RN		RU		RN		RU	
Epochs	500	1000	500	1000	500	1000	500	1000	500	1000	500	1000
Val Acc-B	0.78	0.78	0.78	0.81	0.78	0.78	0.78	0.78	0.78	0.78	0.78	0.78
Val Acc-C	0.89	0.89	0.84	0.81	0.89	0.86	0.89	0.86	0.89	0.86	0.89	0.86
Val Acc-M	0.86	0.86	0.86	0.86	0.78	0.78	0.81	0.78	0.78	0.78	0.78	0.78
Train Acc-B	0.84	0.84	0.84	0.84	0.85	0.85	0.85	0.84	0.85	0.85	0.85	0.85
Train Acc-C	0.89	0.88	0.88	0.87	0.90	0.88	0.90	0.88	0.88	0.88	0.88	0.88
Train Acc-M	0.9	0.9	0.9	0.9	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85
Test Acc-B	0.87	0.87	0.87	0.87	0.87	0.87	0.87	0.87	0.87	0.87	0.87	0.87
Test Acc-C	0.87	0.87	0.79	0.89	0.87	0.89	0.87	0.89	0.87	0.87	0.87	0.87
Test	0.87	0.87	0.87	0.87	0.87	0.87	0.87	0.87	0.87	0.87	0.87	0.87

Acc-M												

OPT	Adagrad											
LR	0.1				0.01				0.001			
PI	RN		RU		RN		RU		RN		RU	
Epochs	500	1000	500	1000	500	1000	500	1000	500	1000	500	1000
Val Acc-B	0.78	0.78	0.78	0.78	0.78	0.78	0.78	0.78	0.78	0.78	0.84	0.84
Val Acc-C	0.89	0.89	0.89	0.89	0.78	0.84	0.81	0.84	0.87	0.81	0.76	0.78
Val Acc-M	0.78	0.78	0.78	0.78	0.78	0.78	0.78	0.78	0.76	0.76	0.78	0.81
Train Acc-B	0.84	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.83	0.83	0.81	0.81
Train Acc-C	0.89	0.89	0.89	0.89	0.85	0.85	0.85	0.85	0.81	0.83	0.77	0.80
Train Acc-M	0.85	0.85	0.85	0.85	0.84	0.83	0.85	0.83	0.78	0.84	0.77	0.82
Test Acc-B	0.87	0.87	0.87	0.87	0.87	0.87	0.87	0.87	0.82	0.82	0.84	0.82
Test Acc-C	0.87	0.87	0.87	0.87	0.84	0.87	0.84	0.87	0.82	0.82	0.82	0.82
Test Acc-M	0.87	0.87	0.87	0.87	0.87	0.87	0.87	0.87	0.82	0.82	0.82	0.82

OPT	Adam											
LR	0.1				0.01				0.001			
PI	RN		RU		RN		RU		RN		RU	
Epochs	500	1000	500	1000	500	1000	500	1000	500	1000	500	1000

Val Acc-B	0.78	0.78	0.78	0.81	0.78	0.78	0.78	0.78	0.78	0.78	0.78	0.78
Val Acc-C	0.84	0.84	0.84	0.86	0.89	0.89	0.89	0.86	0.89	0.89	0.89	0.89
Val Acc-M	0.81	0.89	0.89	0.86	0.78	0.78	0.81	0.78	0.78	0.78	0.78	0.78
Train Acc-B	0.86	0.84	0.86	0.83	0.85	0.85	0.84	0.84	0.85	0.85	0.85	0.84
Train Acc-C	0.88	0.86	0.86	0.87	0.89	0.89	0.89	0.90	0.88	0.89	0.88	0.89
Train Acc-M	0.91	0.88	0.88	0.9	0.85	0.85	0.86	0.85	0.85	0.85	0.85	0.85
Test Acc-B	0.87	0.87	0.87	0.87	0.87	0.87	0.87	0.87	0.87	0.87	0.87	0.87
Test Acc-C	0.87	0.87	0.87	0.87	0.87	0.87	0.87	0.87	0.87	0.87	0.87	0.87
Test Acc-M	0.84	0.87	0.84	0.87	0.87	0.87	0.87	0.87	0.87	0.87	0.87	0.87

Artificial Features				
Degree	Poly Degree 2		Poly Degree 3	
Interaction	Interaction only	All features	Interaction only	All features
Features total	92	105	378	560
Effect on best test Acc	Lower accuracy 0.89 -> 0.87	Lower accuracy 0.89 -> 0.87	Lower accuracy 0.89 -> 0.84 Higher on training	Lower accuracy 0.89 -> 0.87
Effect on best validation Acc	Lower accuracy 0.89 -> 0.78	Higher accuracy 0.89 -> 0.92	Lower accuracy 0.89 -> 0.81 Higher on training	Lower accuracy 0.89 -> 0.81 Over-fitting

4. Problems in Models

d. Overfitting:

solved by early stopping (i.e decreasing the number of epochs) or by reducing the model complexity either by removing some layer/neurons -which doesn't apply in our case- or by adding dropout in features layer. Also we can add more data to our model. Other methods exist that don't apply to 1 neuron case.

e. Underfitting:

Solved by increasing the model complexity by adding more neurons/layers -which doesn't apply to our case-, or by transforming the model and increasing the space dimension (i.e adding X^2 , Cos(X), etc), and finally we can increase the number of epochs.

f. Overshooting:

Overshooting the optimal value of the parameters is solved by decreasing the learning rate.

6. Performance Analysis

a. Best Accuracy

   i. Validation : 0.89 (colored green in the tables above)

   ii. Train : 0.91 (colored green in the tables above)

   iii. Test : 0.89 (colored green in the tables above)

b. Hyperparameters effect

   i. We can see that Categorical Hinge gave the highest accuracy with different optimizers, followed by Mean Squared Error which gave slightly better performance Categorical Hinge when using learning rate of 0.1 in Adam and RMSProp.

ii. Using 500 / 1000 epochs gave different results based on the loss function and optimizers used but overall we can see that using 500 epochs is more efficient.

    However in SGD, using 1000 epochs is more efficient.

iii. Highest performance for validation, train and test sets was when using RMSProp, also for both validation and training data Adam achieved the same results.
iv. Parameter initialization didn't affect much, but gave the best for tests with Random uniform and best for training with random normal.
v. Best 2 optimizers were Adam and RMSprop, and worst was SGD. Adagrad achieved the same best result on validation set only.

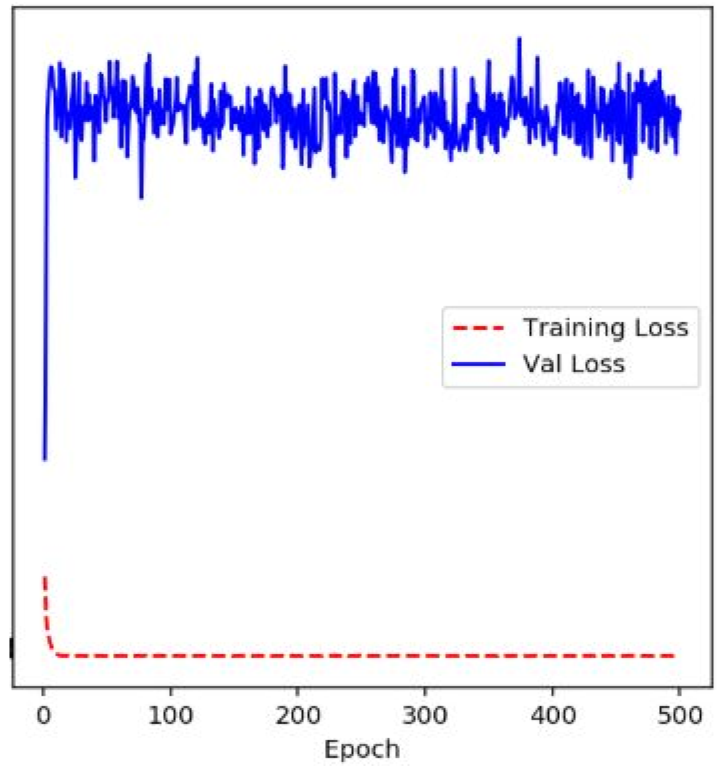
c. Artificial features
    i. Added polynomials of 2nd and 3rd degrees and 1st degree would only result in 1 new feature which most probably won't affect the result
    ii. Validation and training accuracy increased only when using polynomial of 2nd degree with 105 features

Appendix

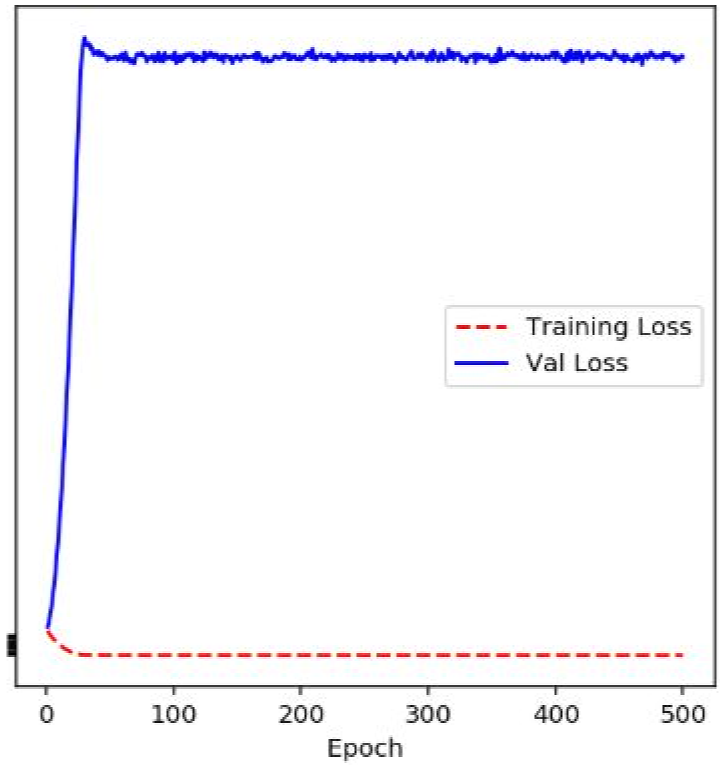
1. Linear Regression Sample Run

a. RMSprop

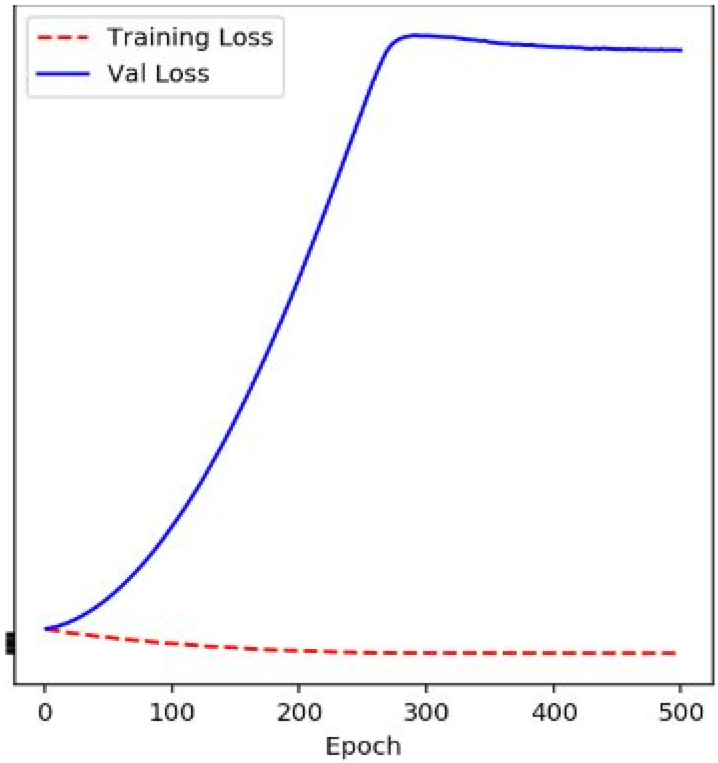
i. R_LR 0.1



ii. R_LR 0.01

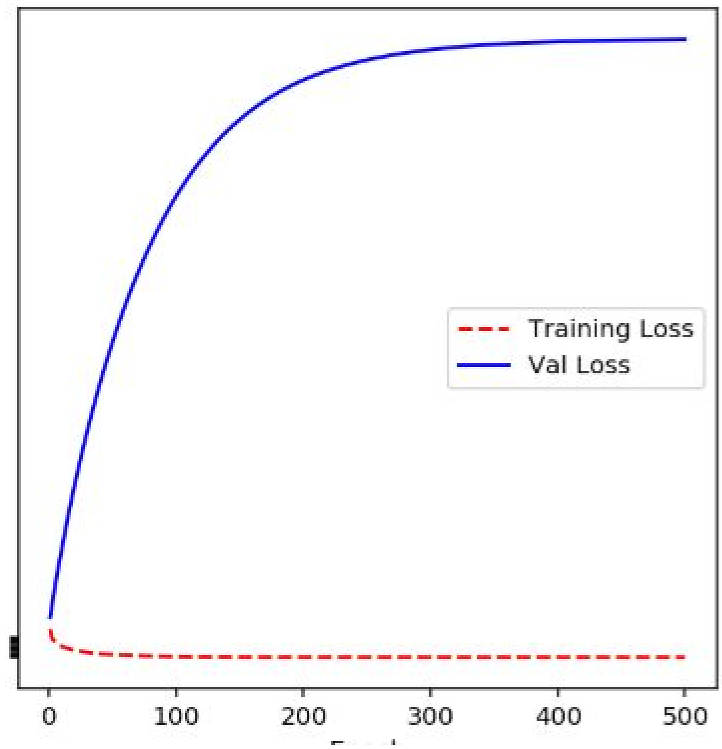


iii. R_LR 0.001

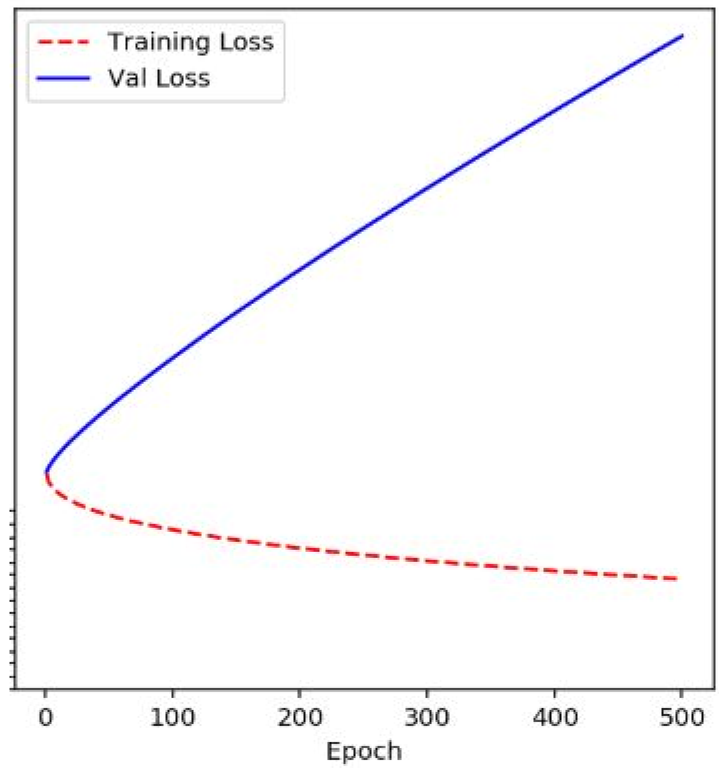


b. Adagrad

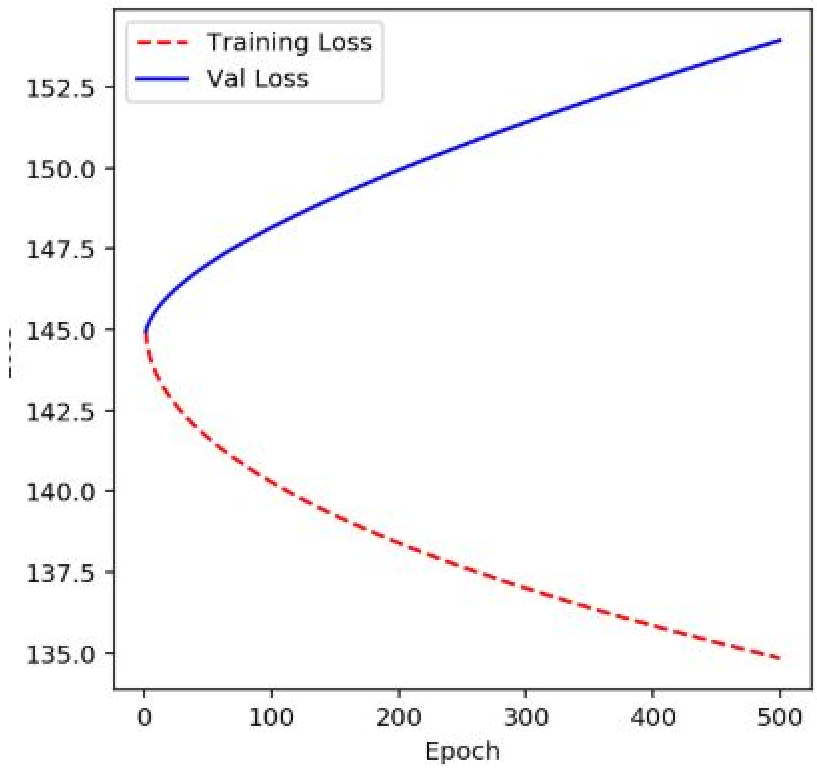
i. Adagrad LR 0.1



ii. Adagrad LR 0.01

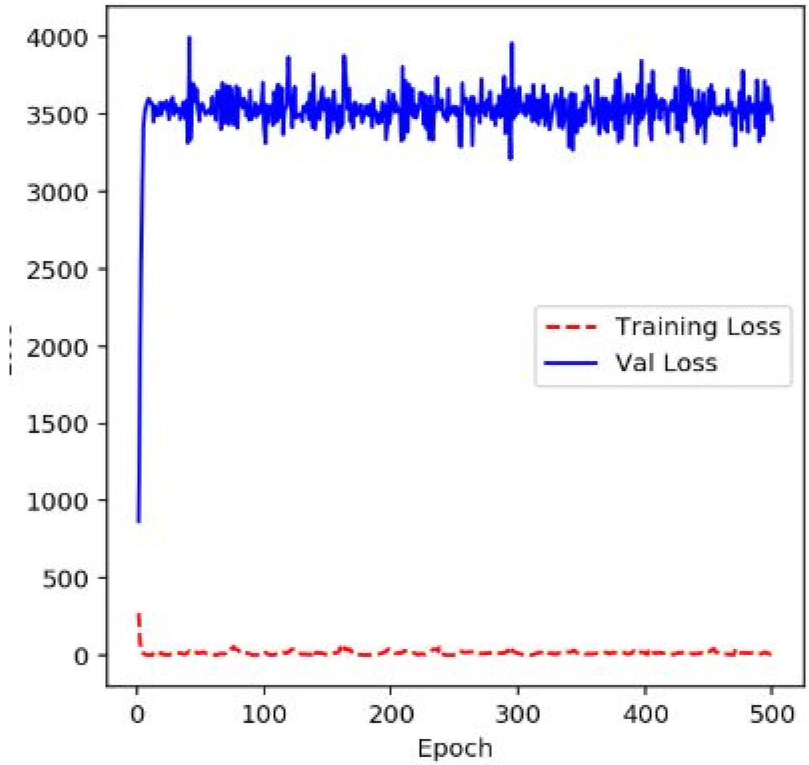


iii. Adagrad LR 0.001



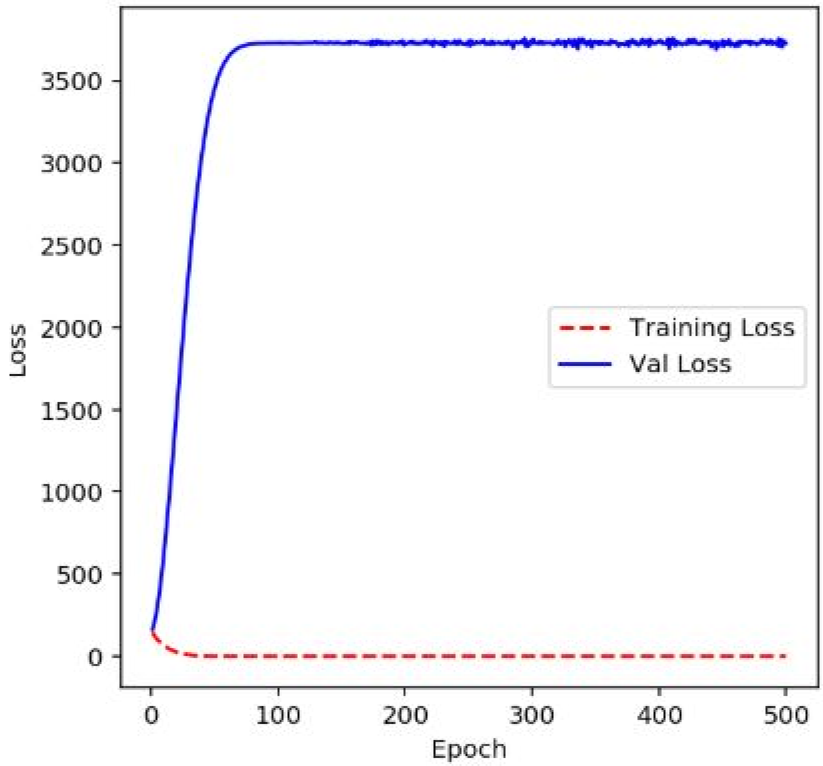
c. Adam

i. Adam LR 0.1

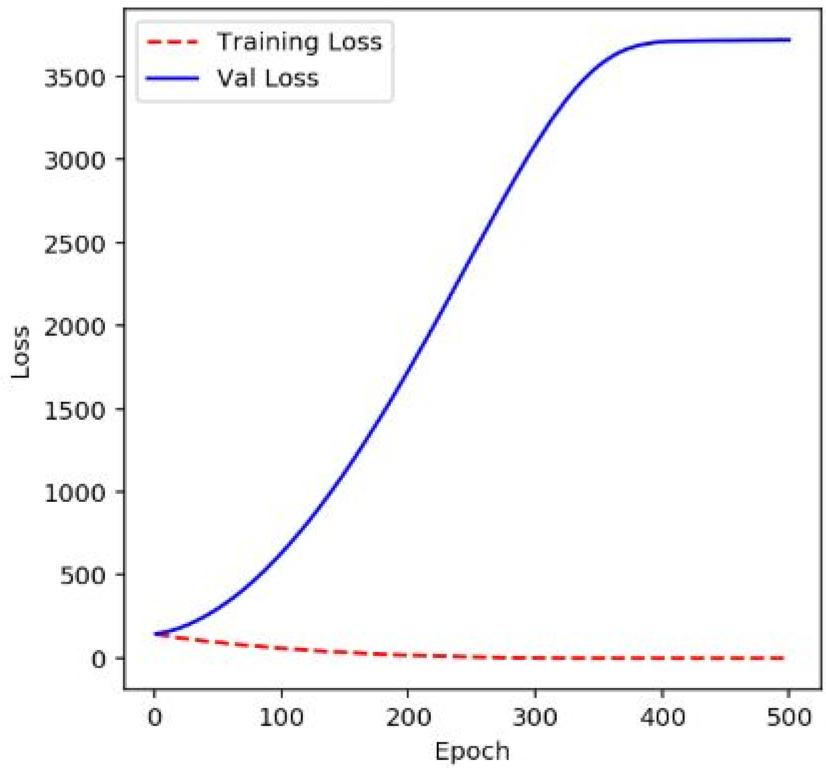


4000
3500
3000
2500
2000
1500
1000
500
0
0
100
200
300
400
500
Epoch
Training Loss
Val Loss

ii. Adam LR 0.01

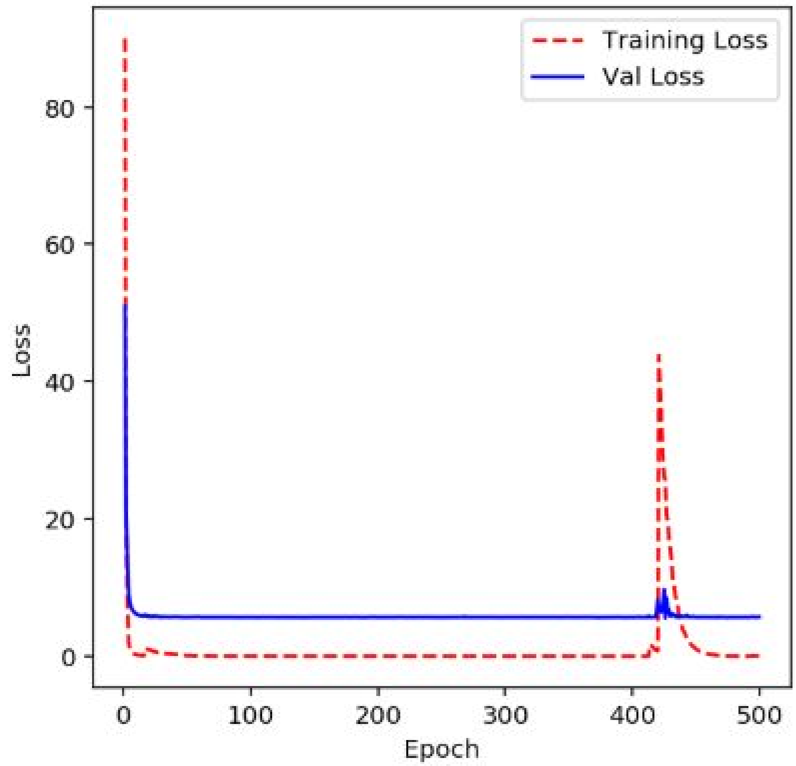


iii. Adam LR 0.001

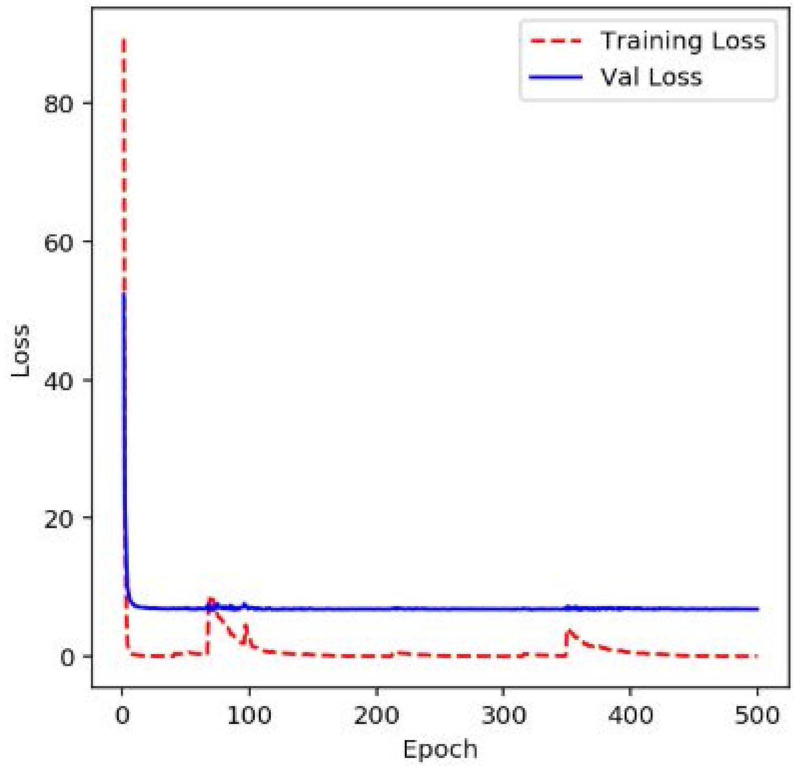


d. SGD

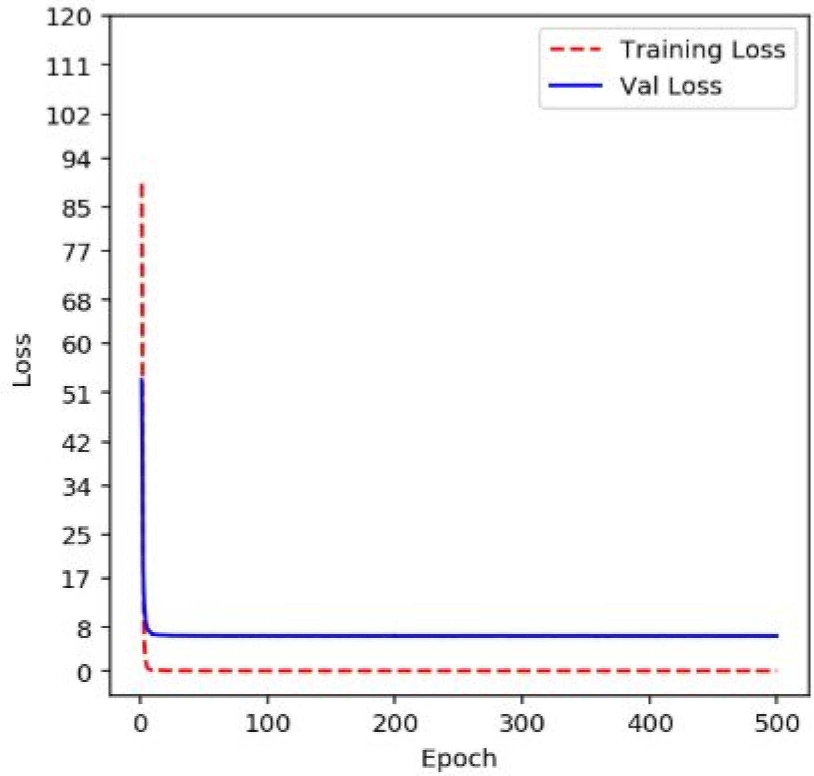
i. <u>SGD LR 0.01, no Momentum</u>



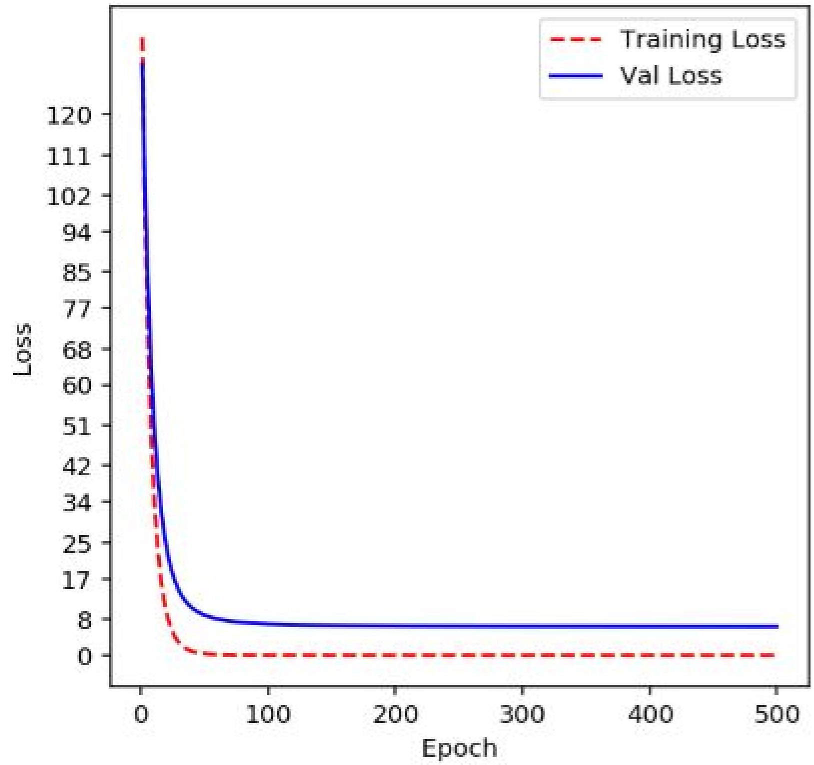
ii. SGD LR 0.01, Momentum 0.01



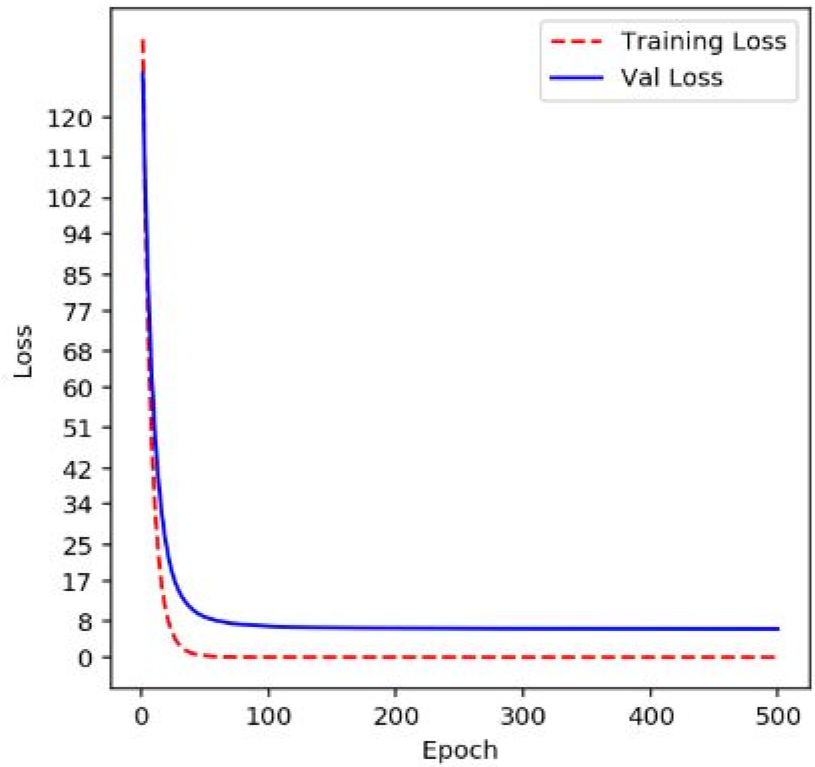
iii. SGD LR 0.01, Momentum 0.001



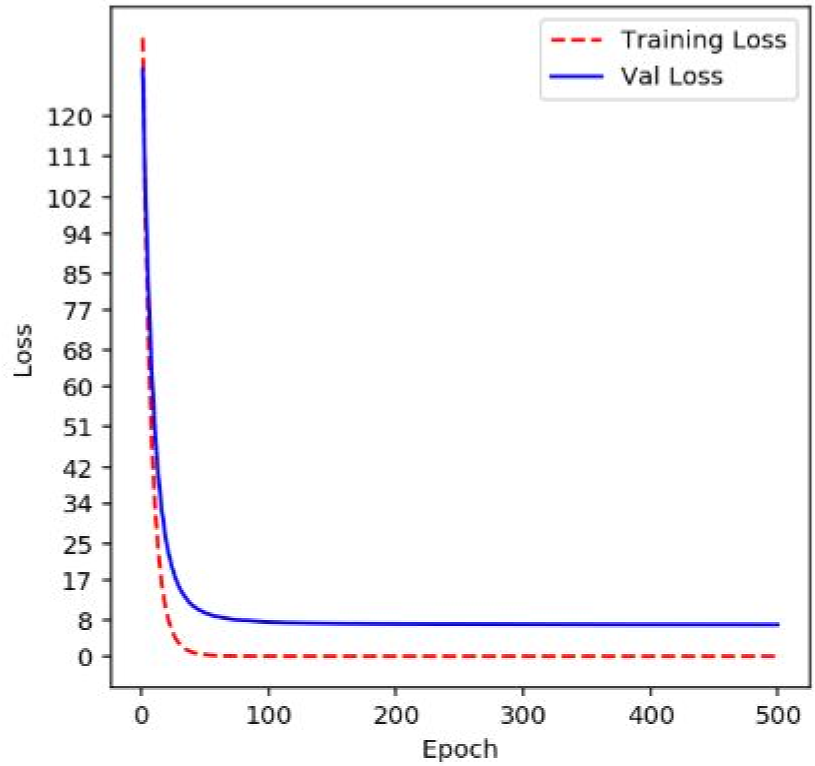
iv. SGD LR 0.001, no Momentum



v. SGD LR 0.001, Momentum 0.01



vi. SGD LR 0.001, Momentum 0.001

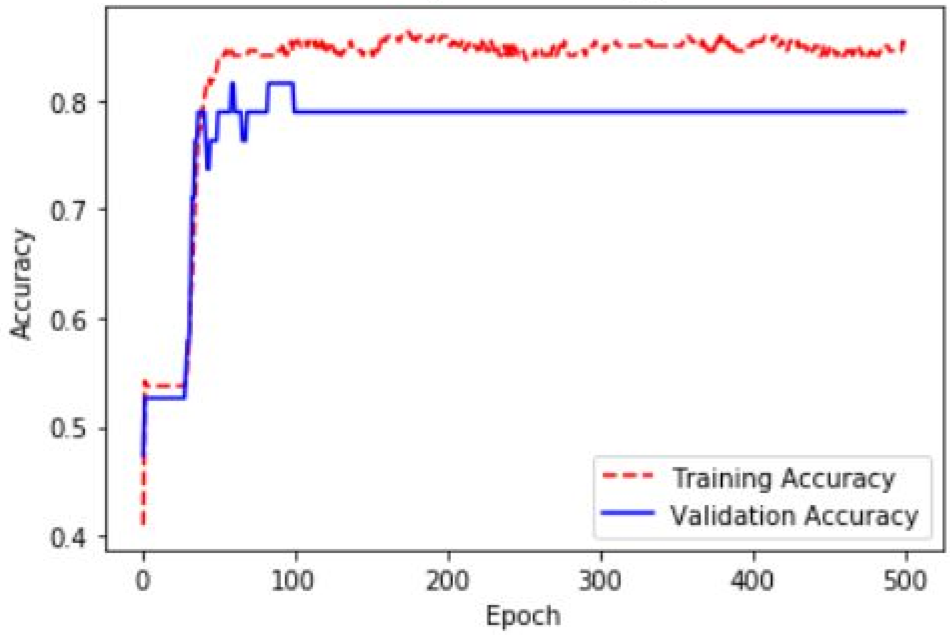


2. Logistic Regression

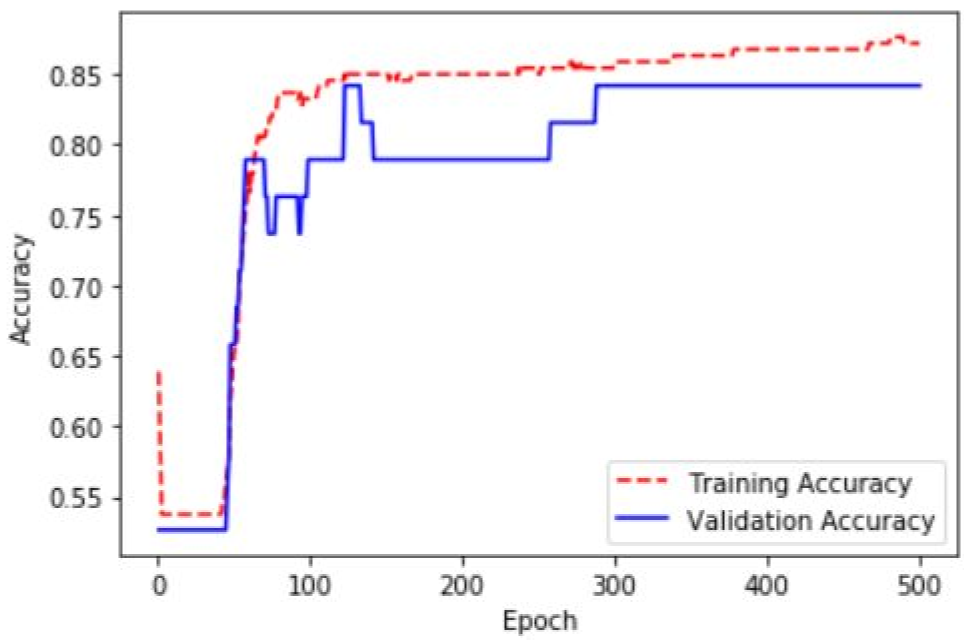
a. SGD-SIGMOID

i. <u>0.01-RU-500</u>

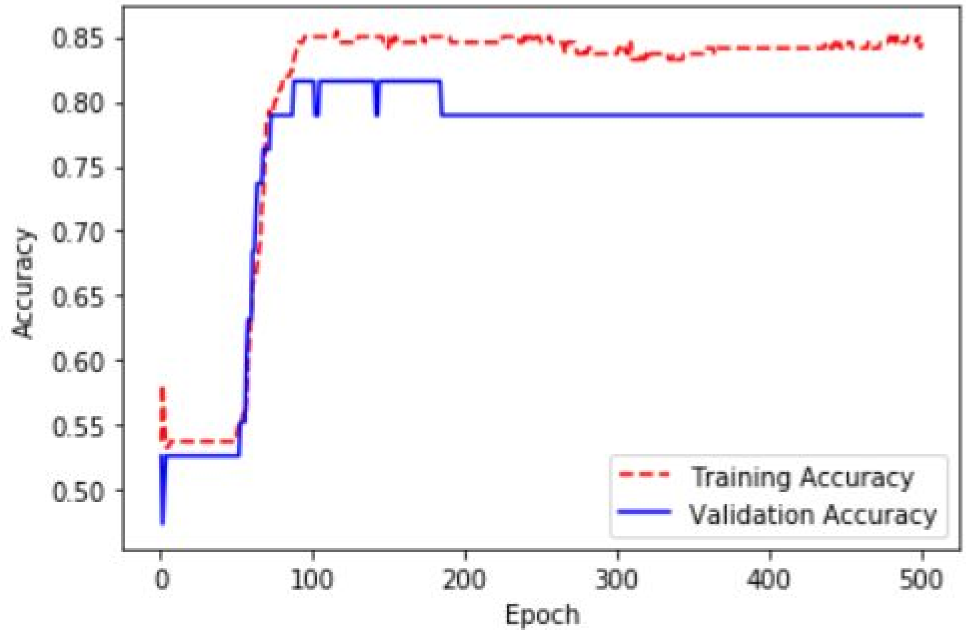
```
Test fraction correct (NN-Score) = 0.39
Test fraction correct (NN-Accuracy) = 0.87
```




```
Test fraction correct (NN-Score) = 0.53
Test fraction correct (NN-Accuracy) = 0.87
```

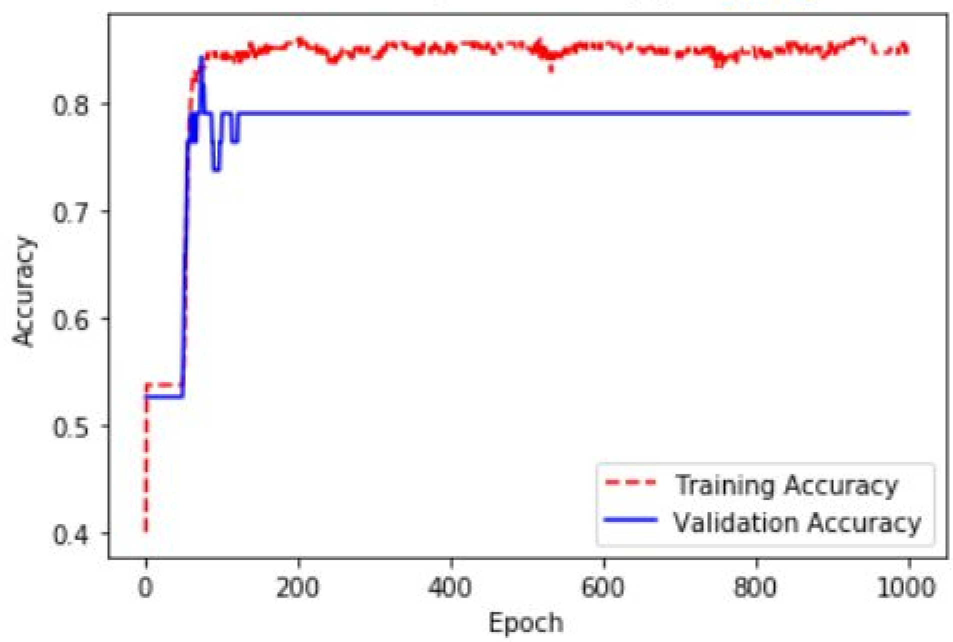


Test fraction correct (NN-Score) = 0.12
Test fraction correct (NN-Accuracy) = 0.87

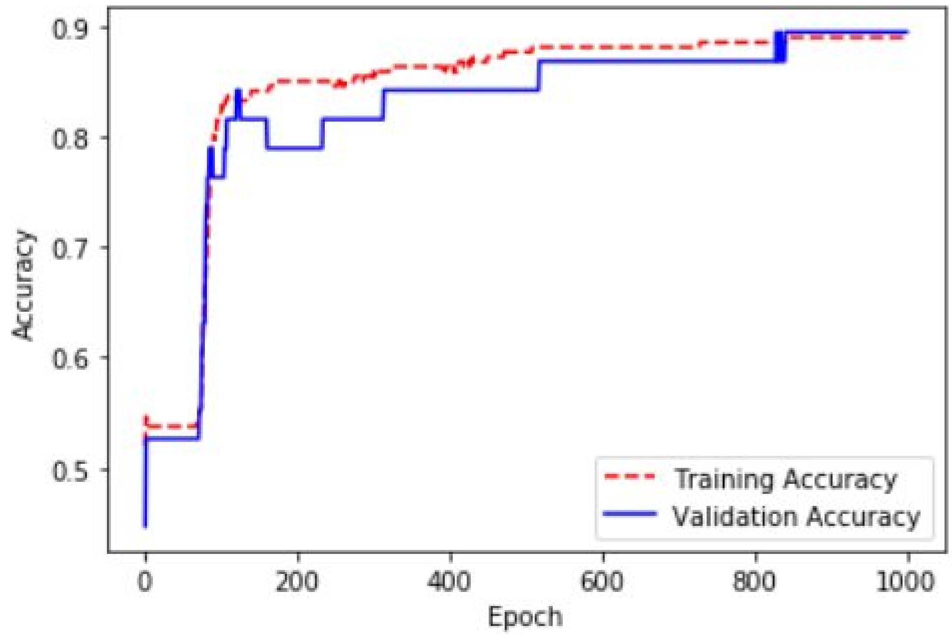


ii. <u>0.01-RU-1000</u>

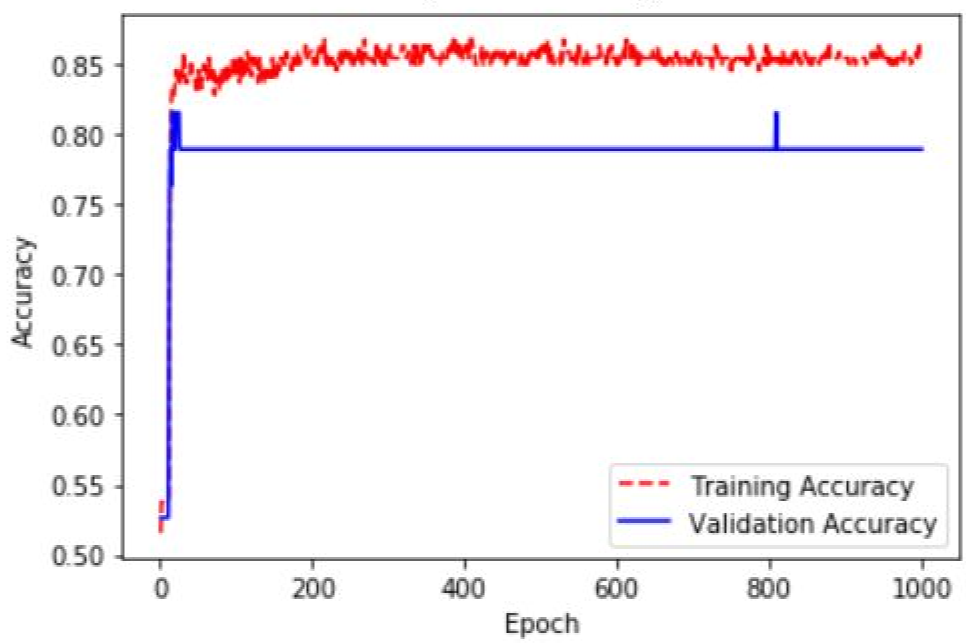
Test fraction correct (NN-Score) = 0.39
Test fraction correct (NN-Accuracy) = 0.87



Test fraction correct (NN-Score) = 0.53
Test fraction correct (NN-Accuracy) = 0.87

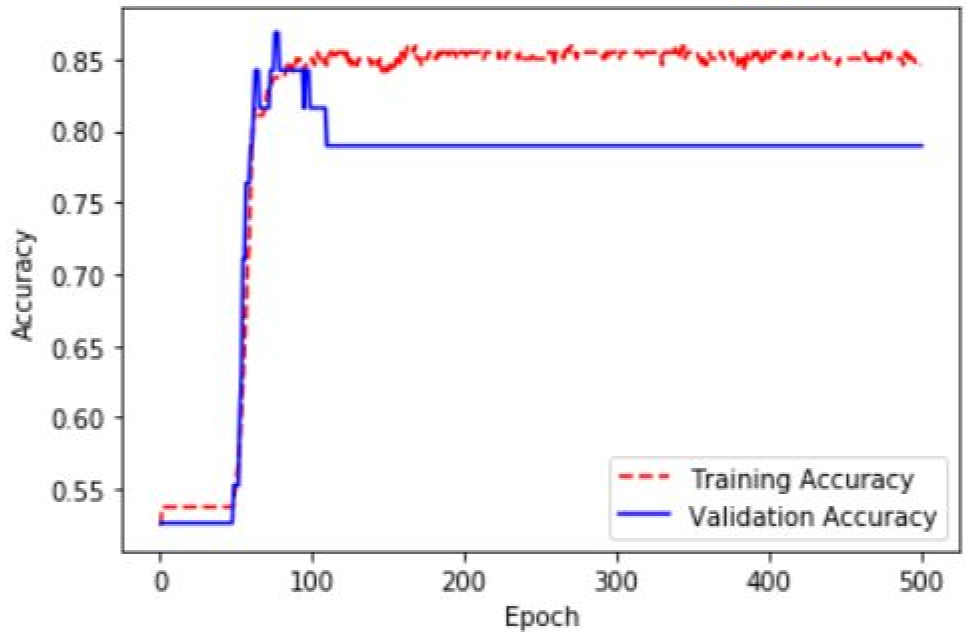


Test fraction correct (NN-Score) = 0.12
Test fraction correct (NN-Accuracy) = 0.87

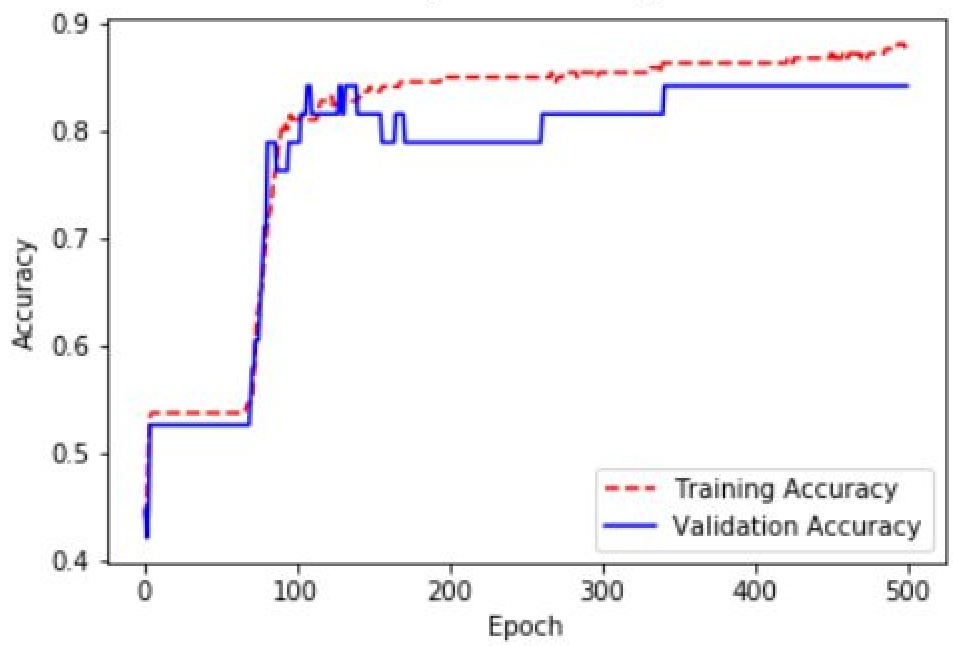


iii. 0.01-RN-500

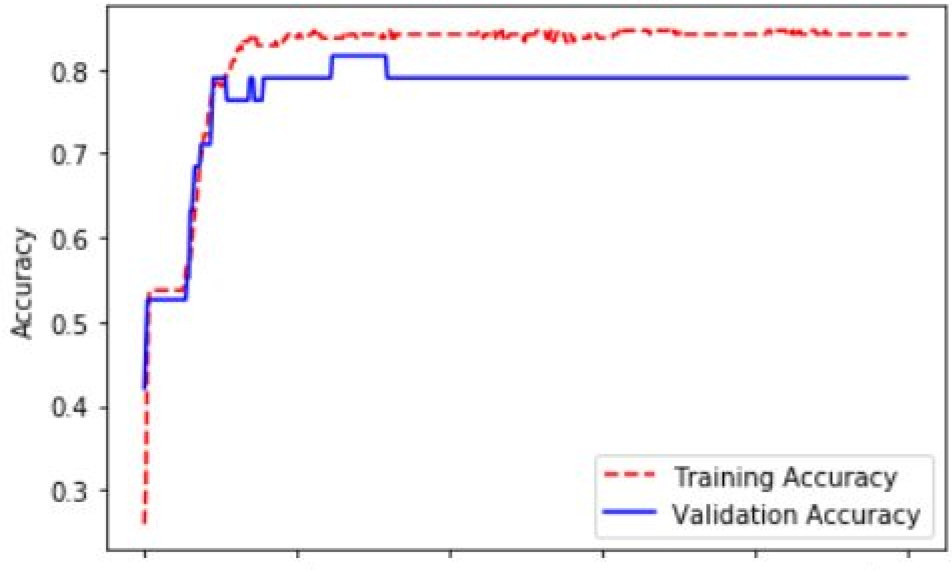
Test fraction correct (NN-Score) = 0.39
Test fraction correct (NN-Accuracy) = 0.87



Test fraction correct (NN-Score) = 0.53
Test fraction correct (NN-Accuracy) = 0.87

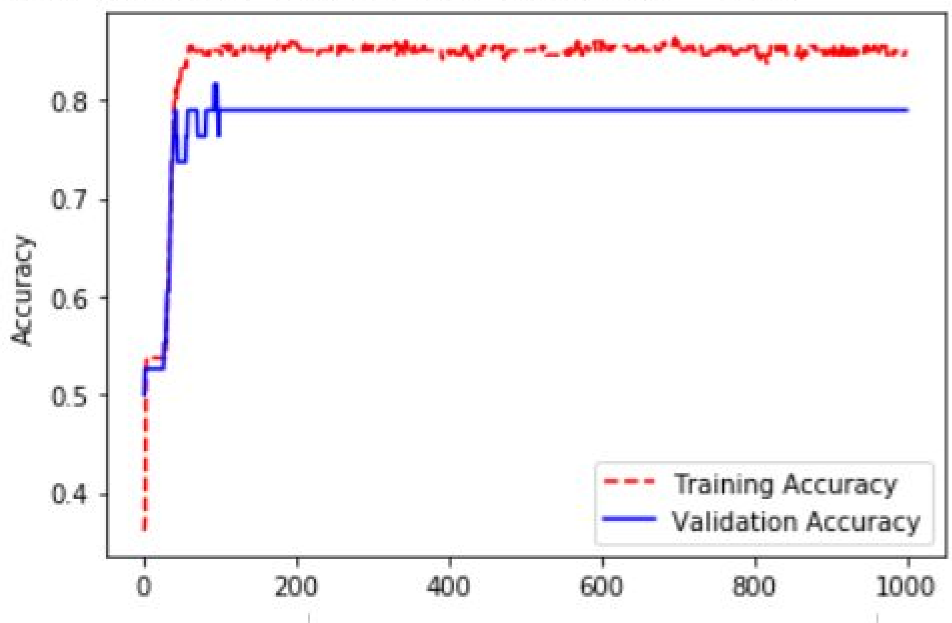


Test fraction correct (NN-Score) = 0.12
Test fraction correct (NN-Accuracy) = 0.87

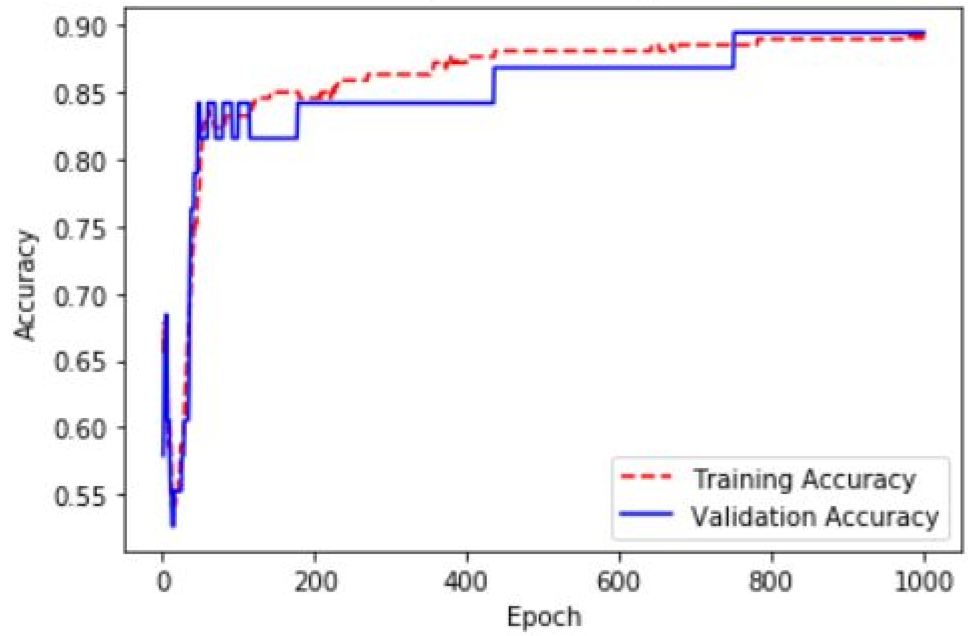


iv. <u>0.01-RN-1000</u>

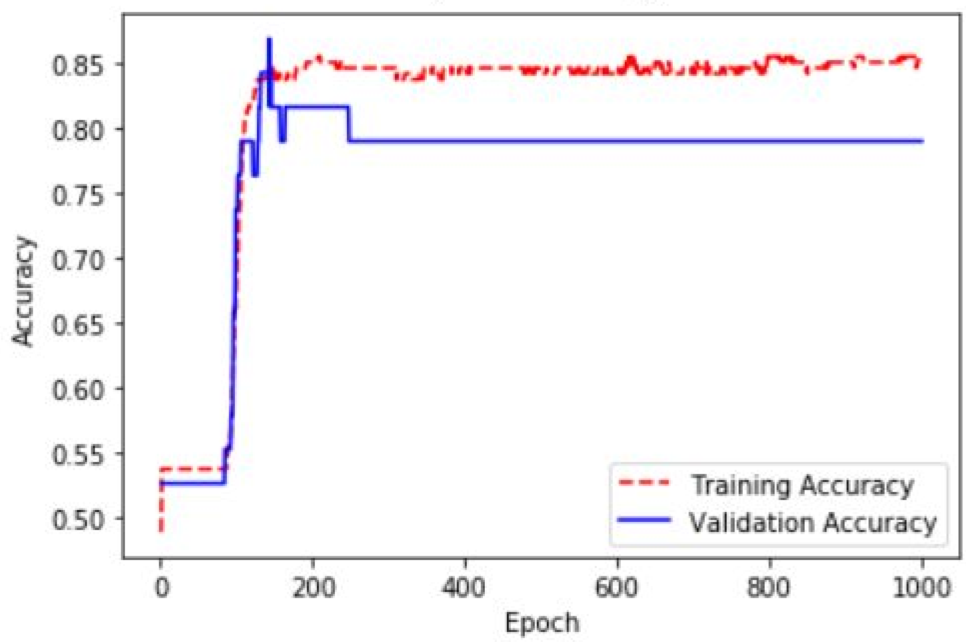
Test fraction correct (NN-Score) = 0.39
Test fraction correct (NN-Accuracy) = 0.87



Test fraction correct (NN-Score) = 0.53
Test fraction correct (NN-Accuracy) = 0.87

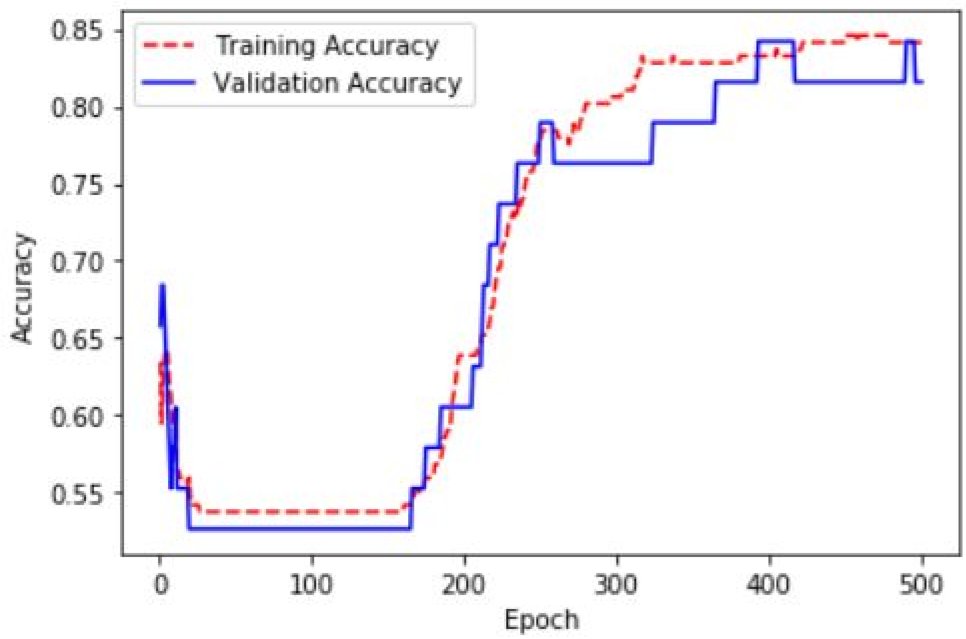


Test fraction correct (NN-Score) = 0.12
Test fraction correct (NN-Accuracy) = 0.87

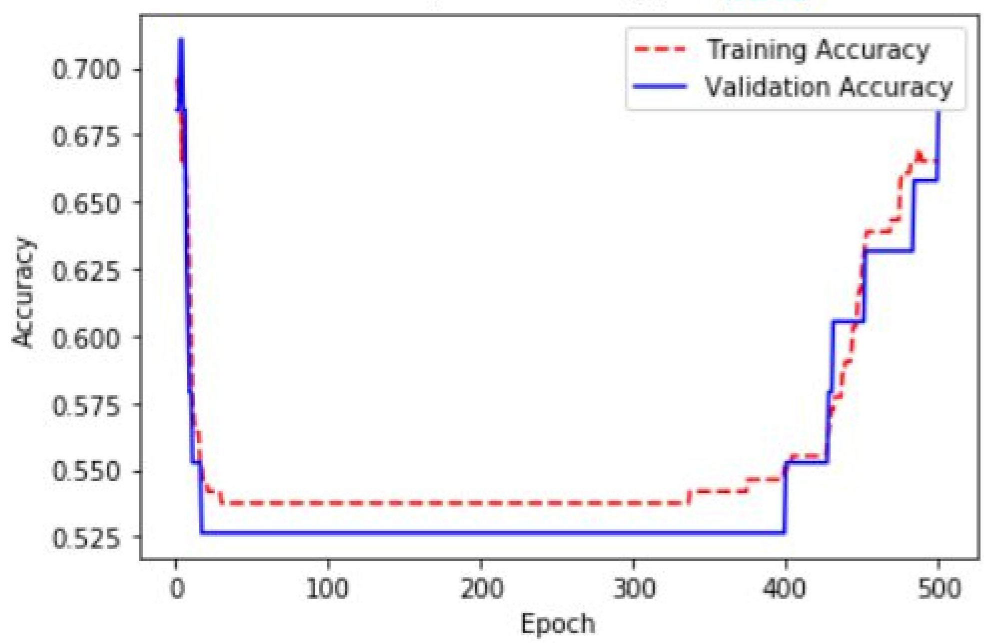


v. 0.001-RU-500

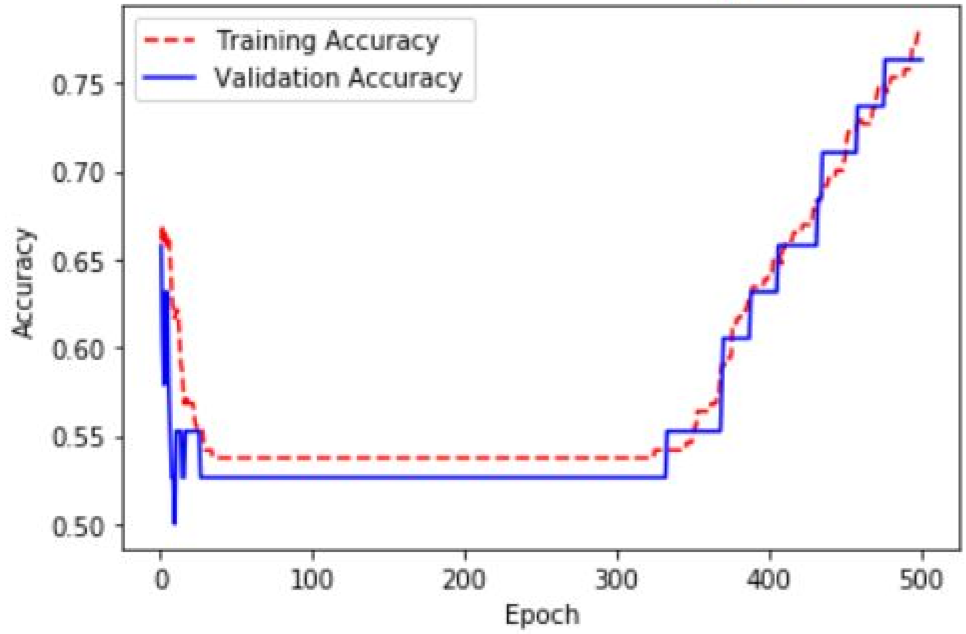
```
Test fraction correct (NN-Score) = 0.41
Test fraction correct (NN-Accuracy) = 0.84
```




```
Test fraction correct (NN-Score) = 0.87
Test fraction correct (NN-Accuracy) = 0.66
```

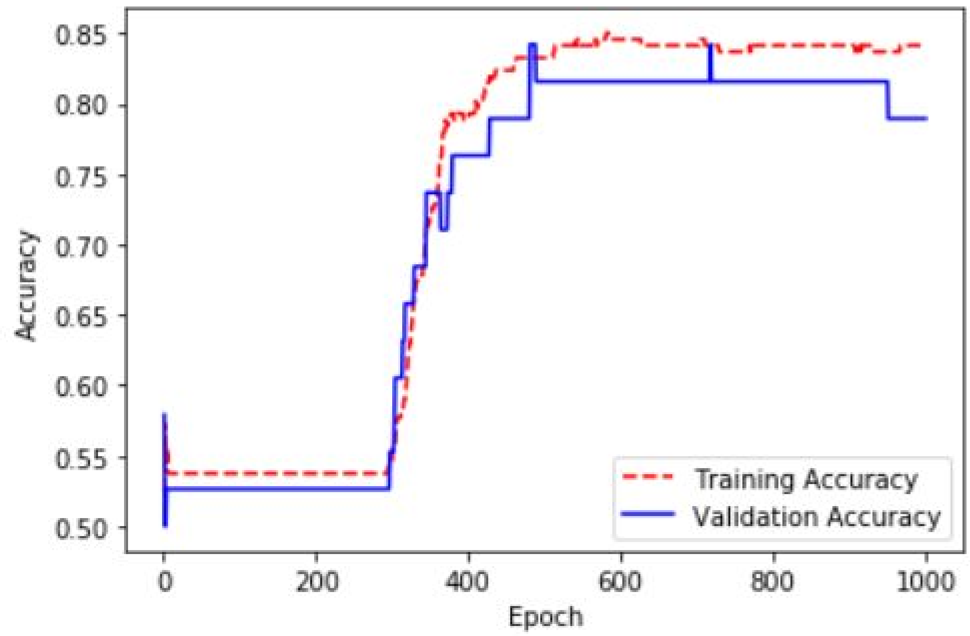


Test fraction correct (NN-Score) = 0.23
Test fraction correct (NN-Accuracy) = 0.79

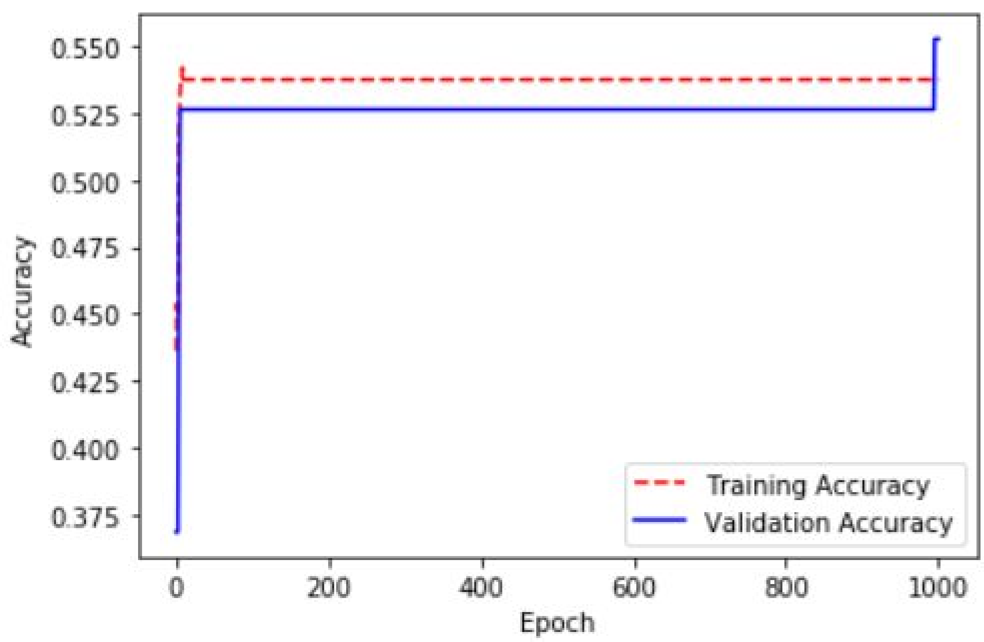


vi. 0.001-RU-1000

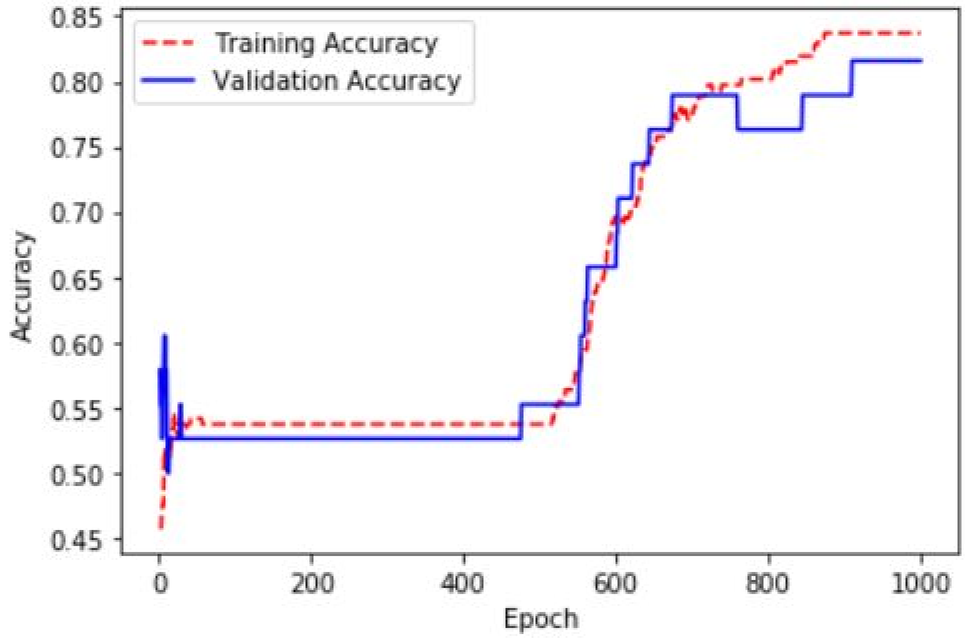
```
Test fraction correct (NN-Score) = 0.38
Test fraction correct (NN-Accuracy) = 0.87
```




```
Test fraction correct (NN-Score) = 0.87
Test fraction correct (NN-Accuracy) = 0.61
```

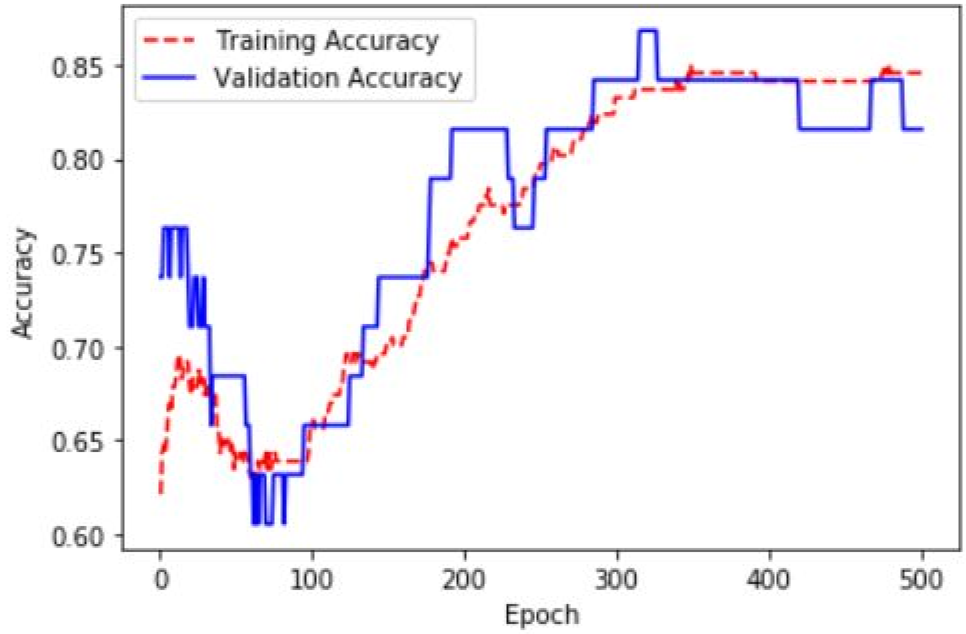


```
Test fraction correct (NN-Score) = 0.15
Test fraction correct (NN-Accuracy) = 0.84
```

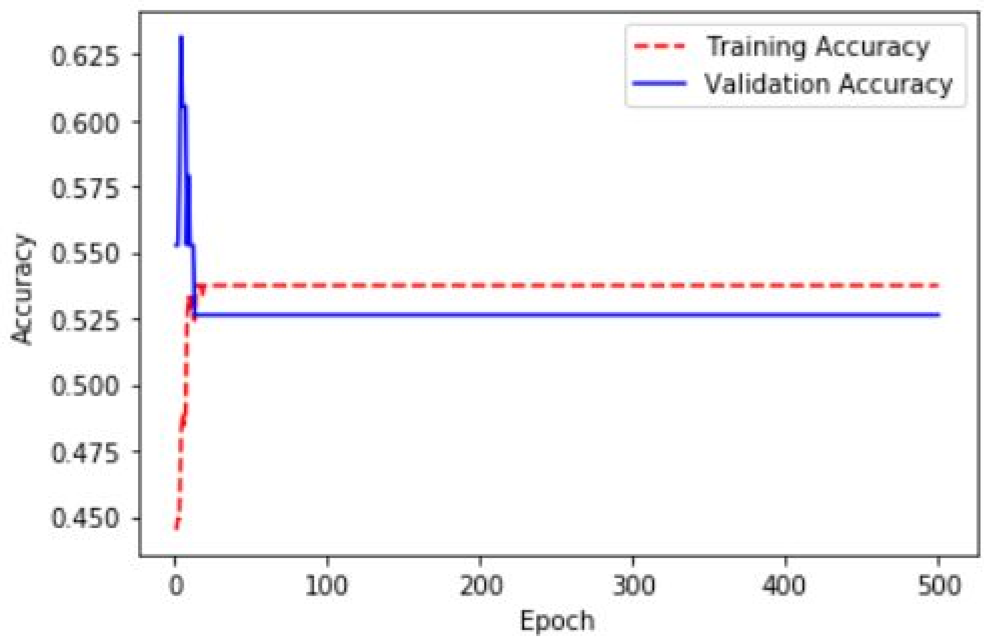


vii. <u>0.001-RN-500</u>

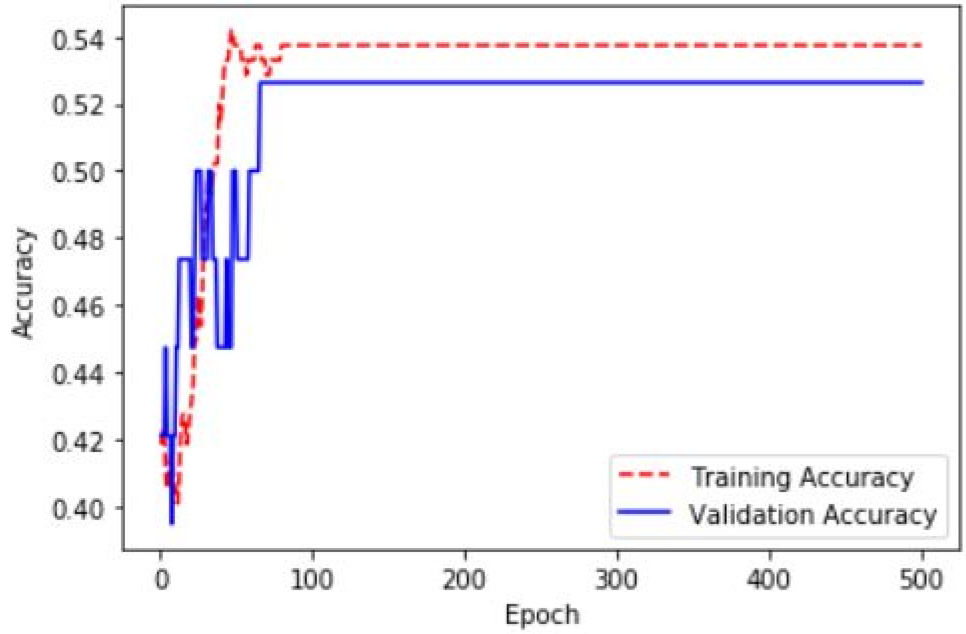
Test fraction correct (NN-Score) = 0.39
Test fraction correct (NN-Accuracy) = 0.84



Test fraction correct (NN-Score) = 0.89
Test fraction correct (NN-Accuracy) = 0.61

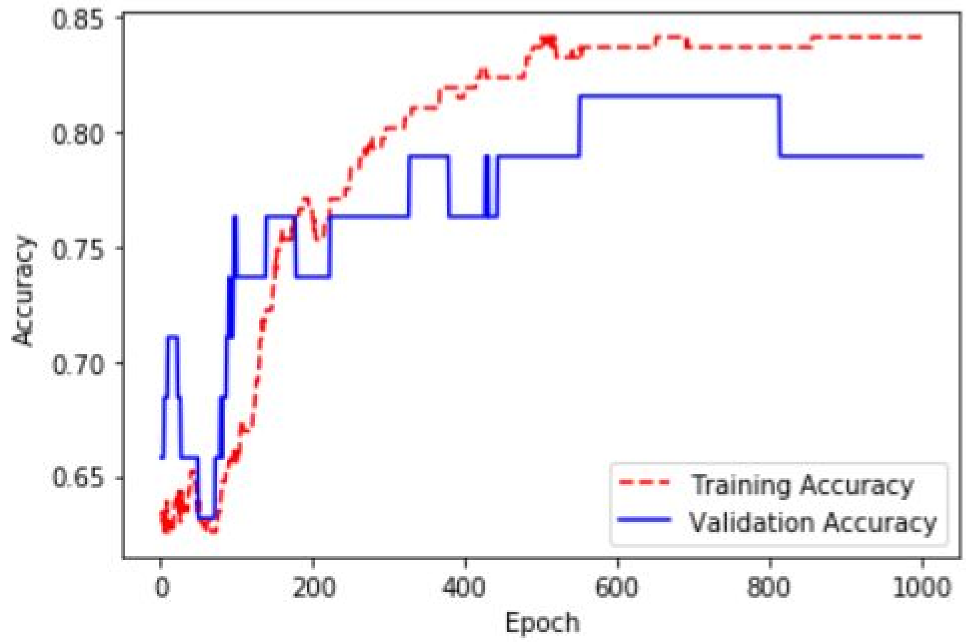


```
Test fraction correct (NN-Score) = 0.25
Test fraction correct (NN-Accuracy) = 0.61
```

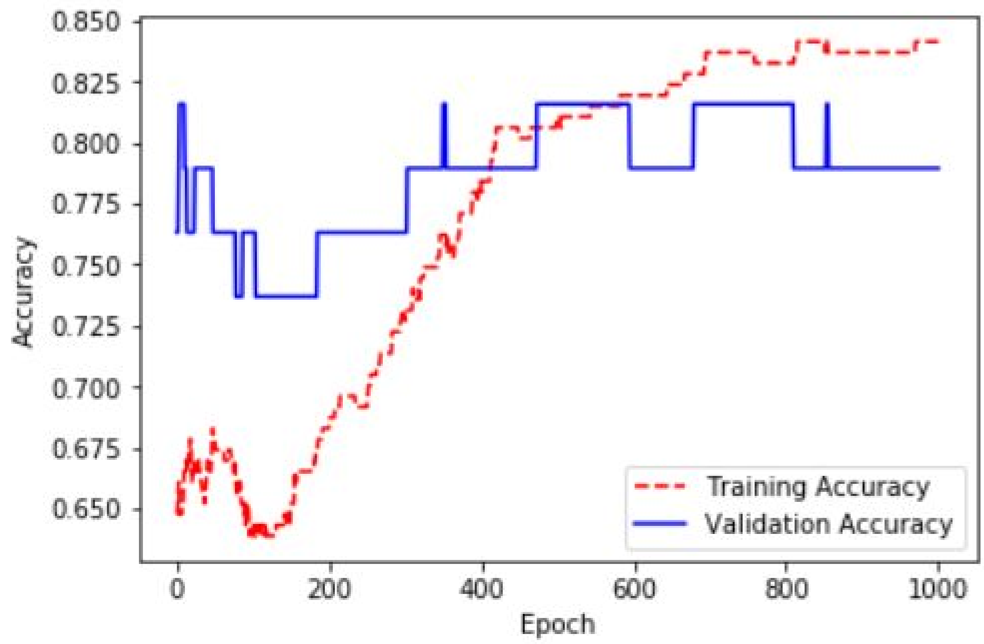


viii. 0.001-RN-1000

```
Test fraction correct (NN-Score) = 0.38
Test fraction correct (NN-Accuracy) = 0.87
```




```
Test fraction correct (NN-Score) = 0.60
Test fraction correct (NN-Accuracy) = 0.84
```



```
Test fraction correct (NN-Score) = 0.16
Test fraction correct (NN-Accuracy) = 0.84
```