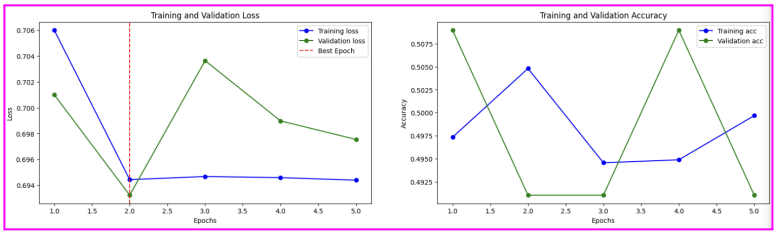
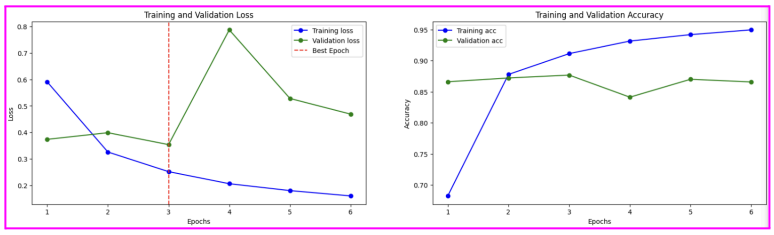
Training model: Hidden Layers=[1], Learning Rate=0.1, Input Dim=10000

Avg Validation Accuracy for this setting: 0.7802, Avg Best Epoch: 2

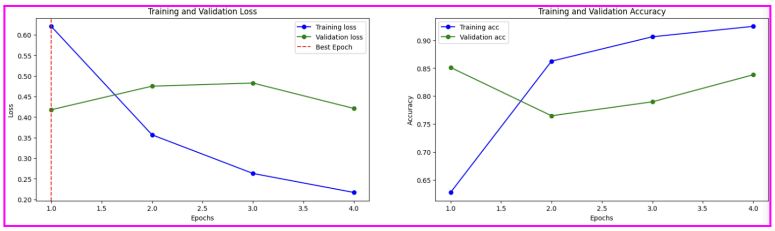
Fold 1



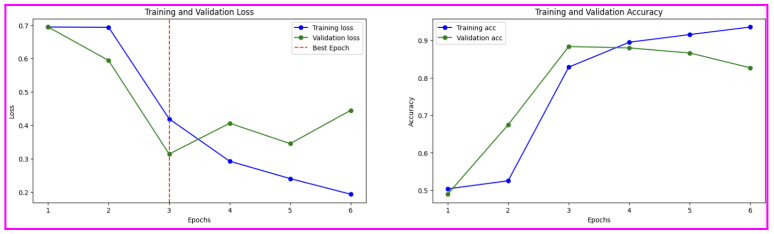
Fold 2



Fold 3



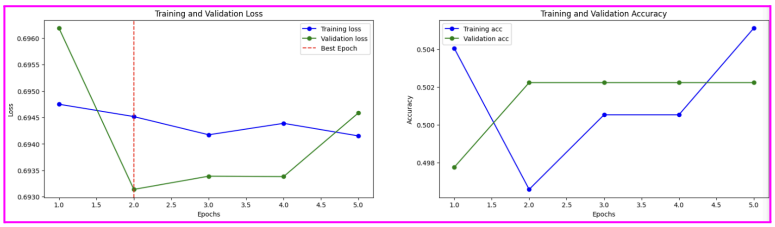
Fold 4



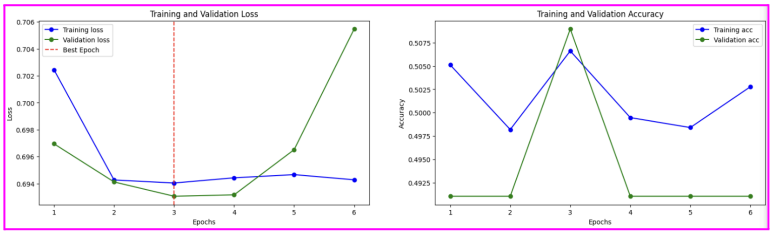
Training model: Hidden Layers=[1], Learning Rate=0.1, Input Dim=1000

Avg Validation Accuracy for this setting: 0.5949, Avg Best Epoch: 2

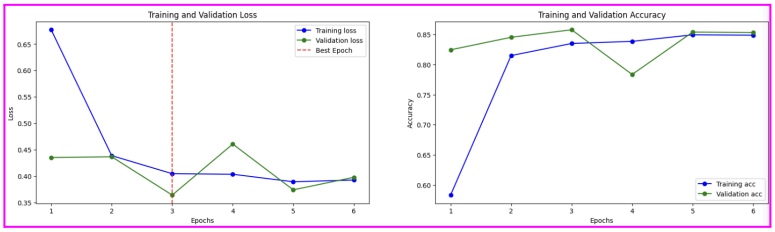
Fold 1



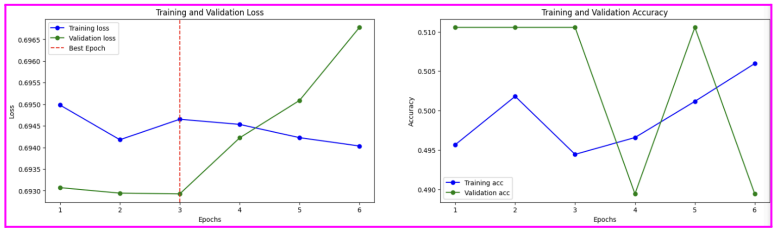
Fold 2



Fold 3



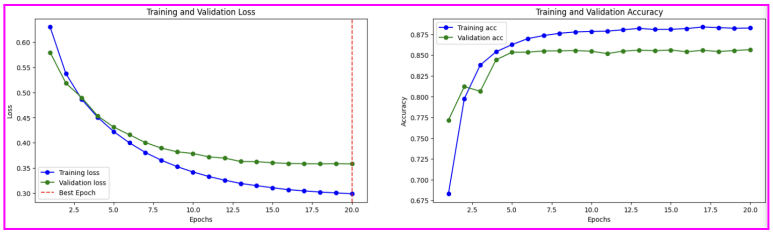
Fold 4



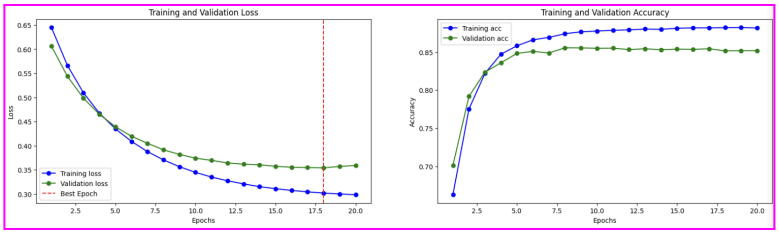
Training model: Hidden Layers=[1], Learning Rate=0.001, Input Dim=1000

Avg Validation Accuracy for this setting: 0.8590, Avg Best Epoch: 7

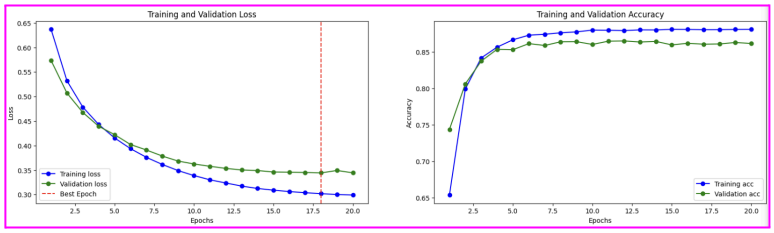
Fold 1



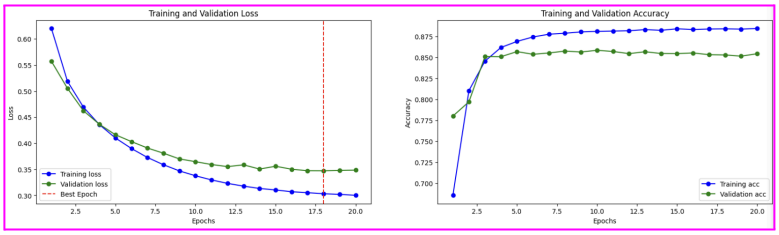
Fold 2



Fold 3



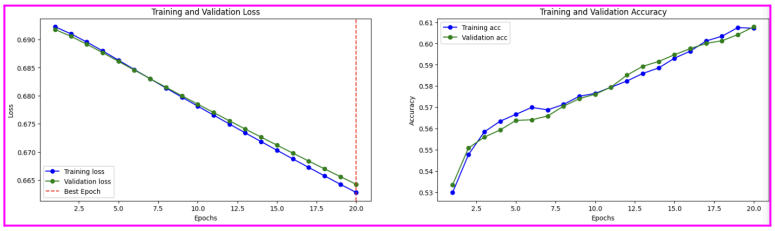
Fold 4



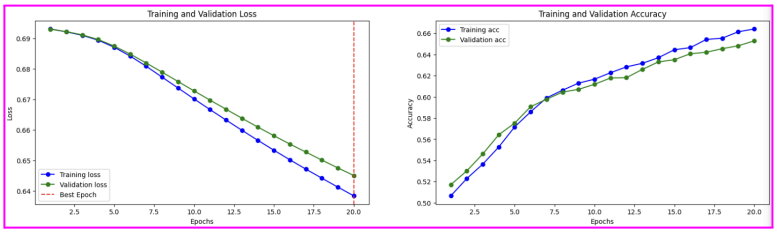
Training model: Hidden Layers=[1], Learning Rate=0.00001, Input Dim=10000

Avg Validation Accuracy for this setting: 0.6016, Avg Best Epoch: 20

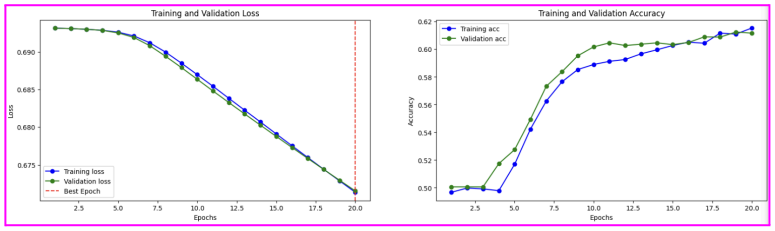
Fold 1



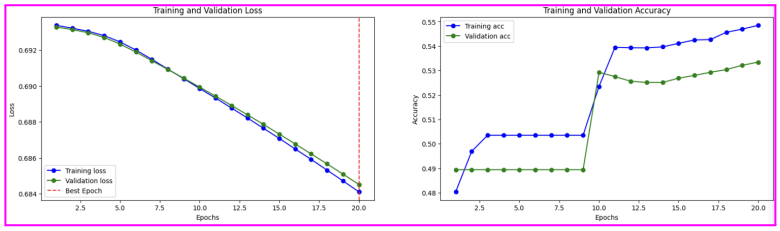
Fold 2



Fold 3



Fold 4

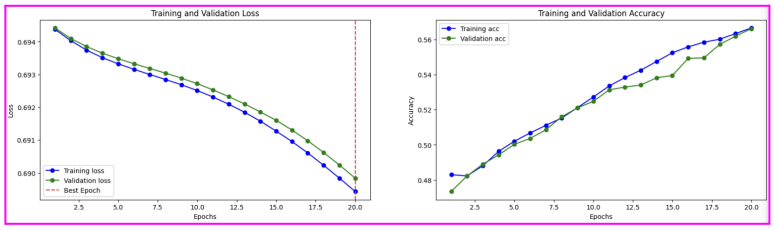


Training model: Hidden Layers=[1], Learning Rate=0.00001, Input Dim=1000

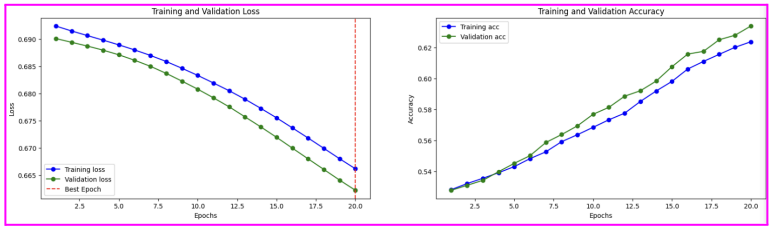
Avg Validation Accuracy for this setting: 0.5870, Avg Best Epoch: 20

Underfitted

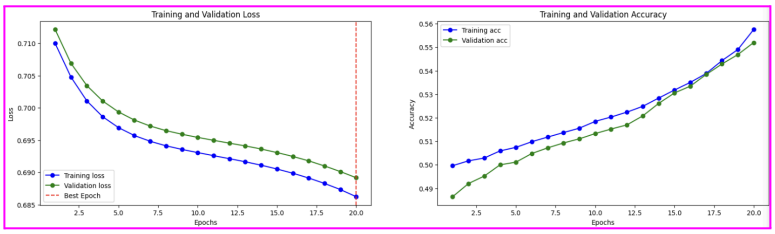
Fold 1



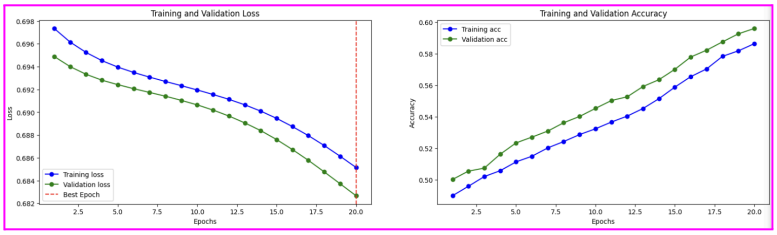
Fold 2



Fold 3



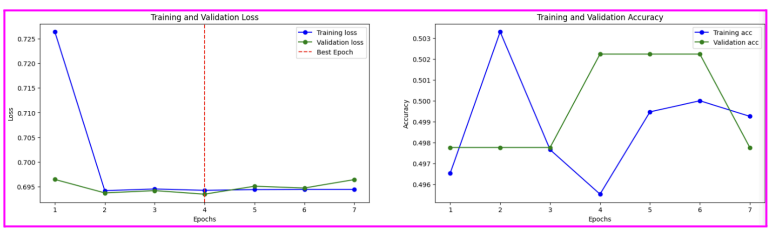
Fold 4



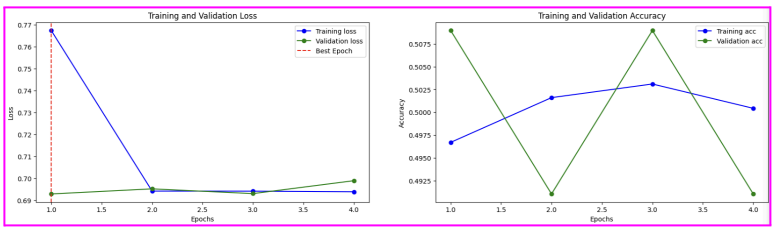
Training model: Hidden Layers=[16, 4], Learning Rate=0.1, Input Dim=10000

Avg Validation Accuracy for this setting: 0.5056, Avg Best Epoch: 2, underfitted

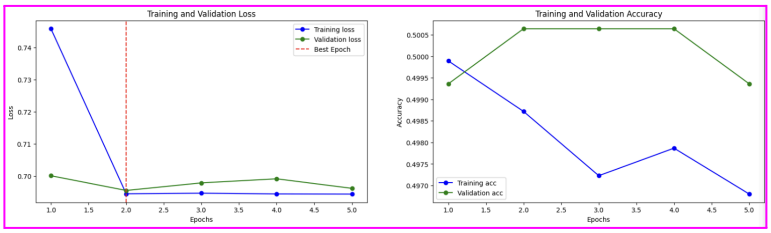
Fold 1



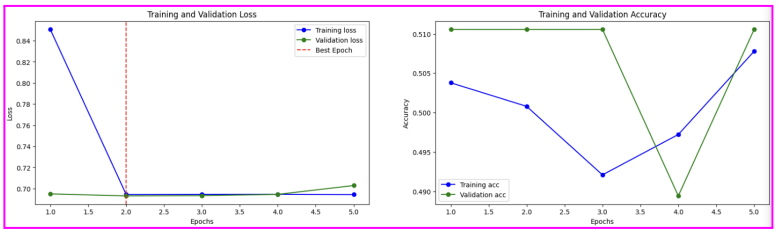
Fold 2



Fold 3



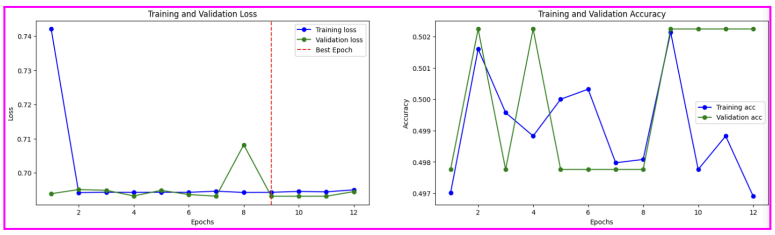
Fold 4



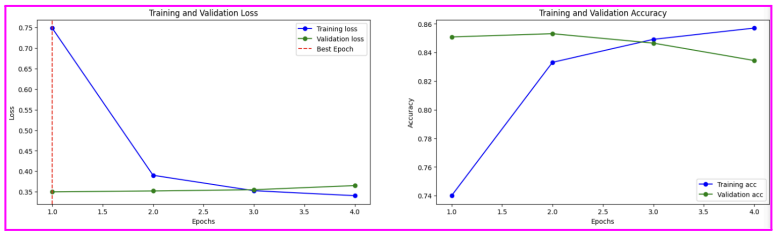
Training model: Hidden Layers=[16, 4], Learning Rate=0.1, Input Dim=1000

Avg Validation Accuracy for this setting: 0.5916, Avg Best Epoch: 4

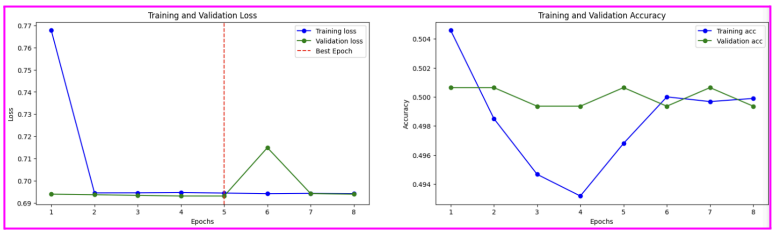
Fold 1



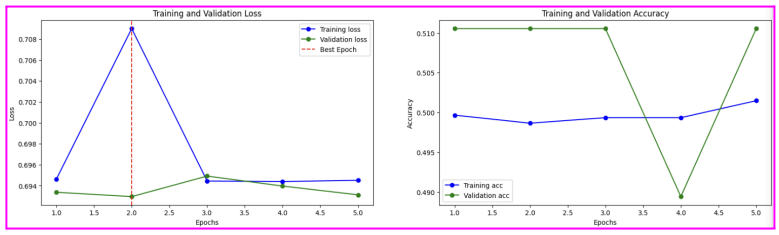
Fold 2



Fold 3



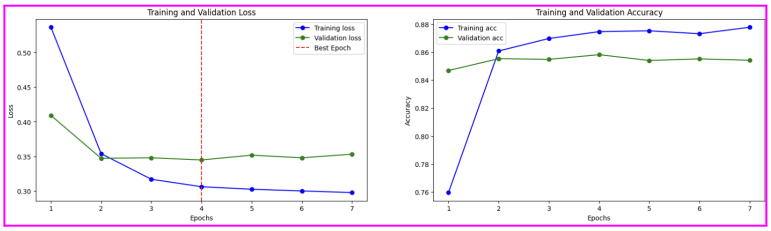
Fold 4



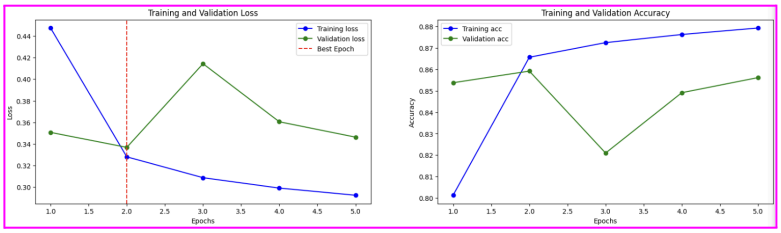
Training model: Hidden Layers=[16, 4], Learning Rate=0.001, Input Dim=1000

Avg Validation Accuracy for this setting: 0.8607, Avg Best Epoch: 2

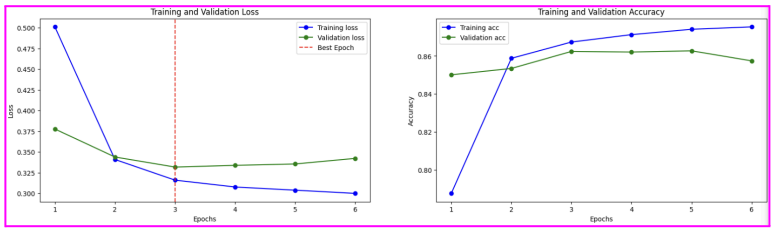
Fold 1



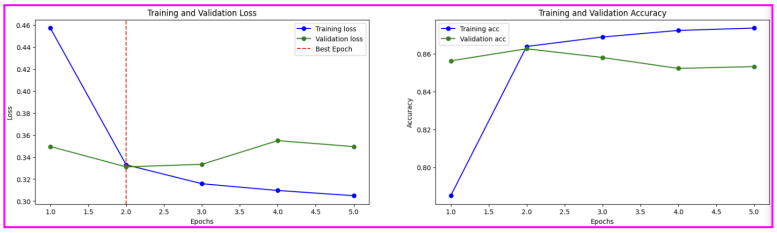
Fold 2



Fold 3



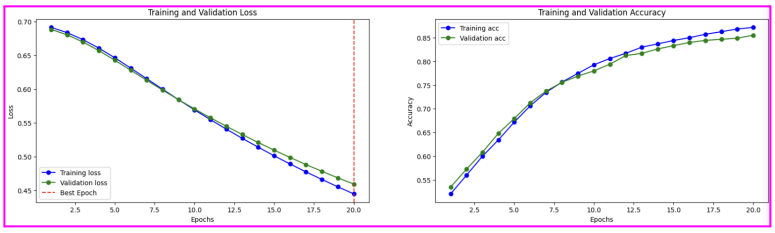
Fold 4



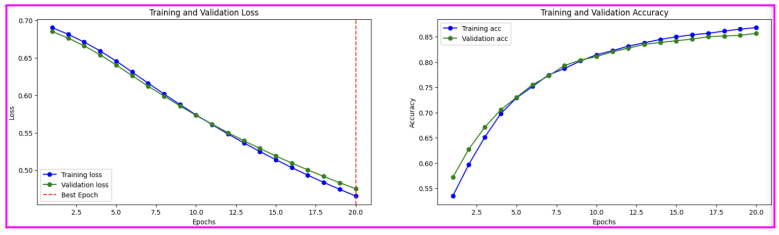
Training model: Hidden Layers=[16, 4], Learning Rate=1e-05, Input Dim=10000

Avg Validation Accuracy for this setting: 0.8180, Avg Best Epoch: 20

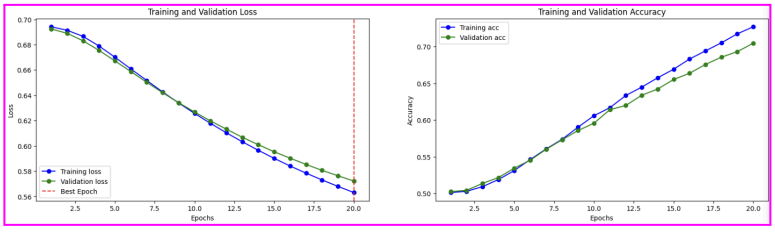
Fold 1



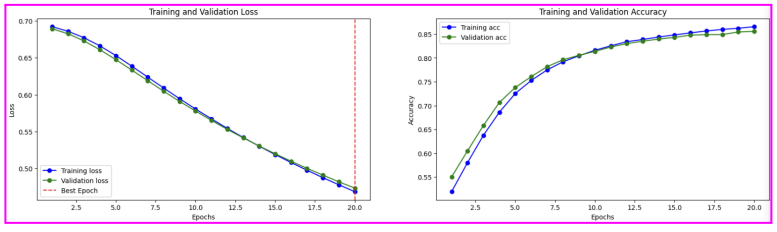
Fold 2



Fold 3



Fold 4

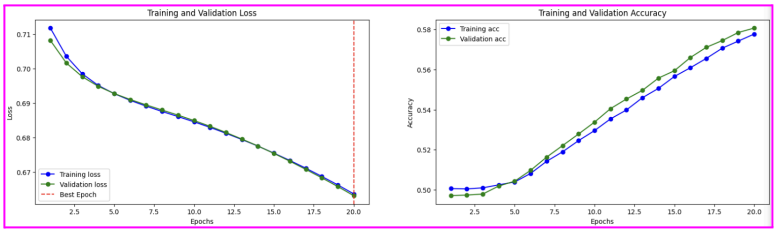


Training model: Hidden Layers=[16, 4], Learning Rate=1e-05, Input Dim=1000

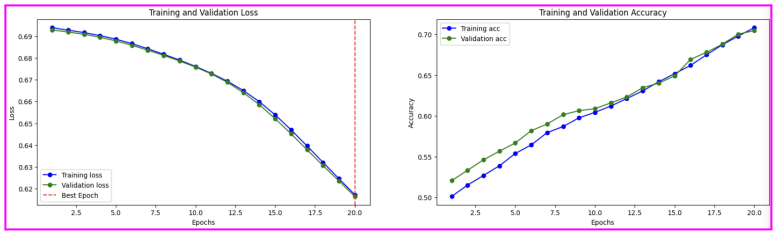
Avg Validation Accuracy for this setting: 0.6738, Avg Best Epoch: 20

Fold 1

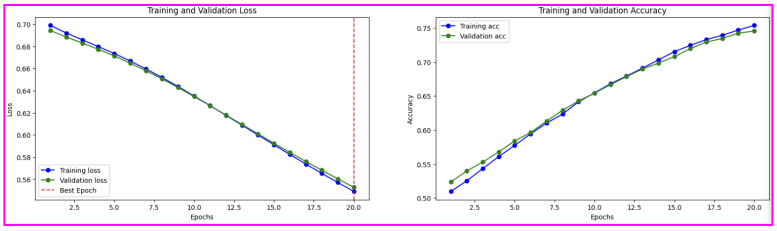
Fold 2

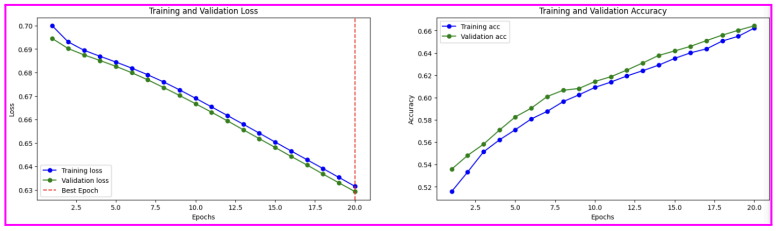


Fold 3



Fold 4

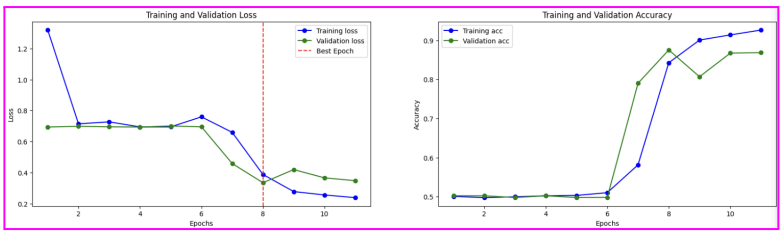




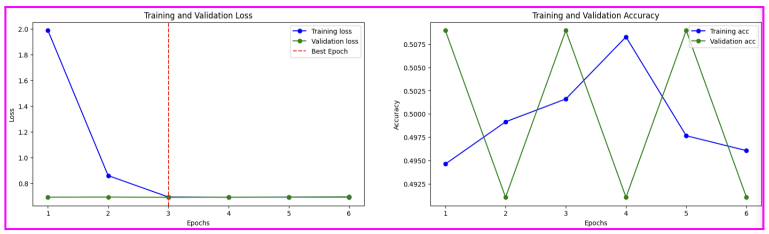
Training model: Hidden Layers=[32, 16, 8, 4], Learning Rate=0.1, Input Dim=10000

Avg Validation Accuracy for this setting: 0.5989, Avg Best Epoch: 4

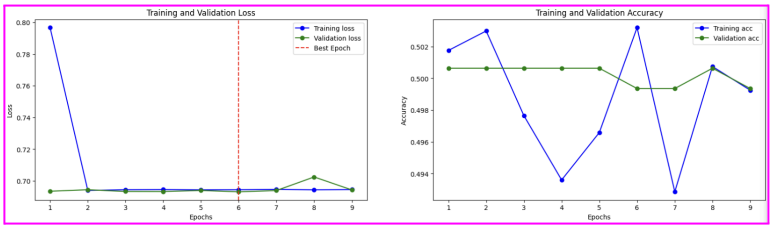
Fold 1



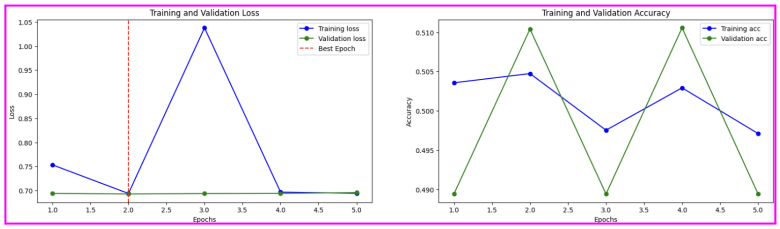
Fold 2



Fold 3



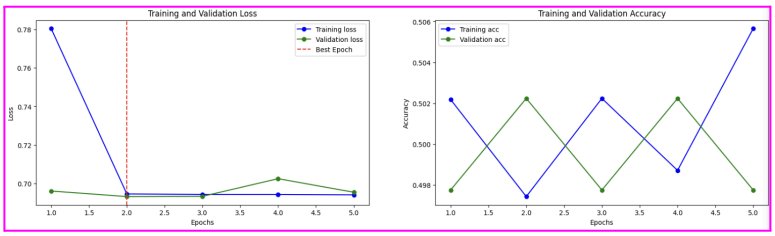
Fold 4



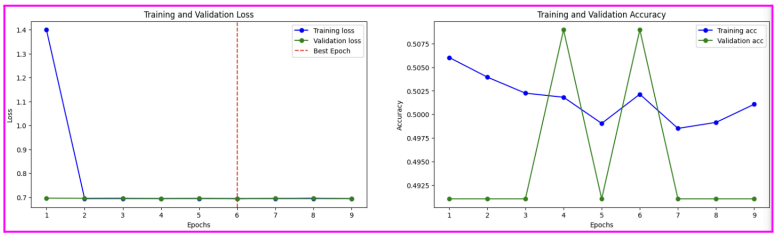
Training model: Hidden Layers=[32, 16, 8, 4], Learning Rate=0.1, Input Dim=1000

Avg Validation Accuracy for this setting: 0.5056, Avg Best Epoch: 3

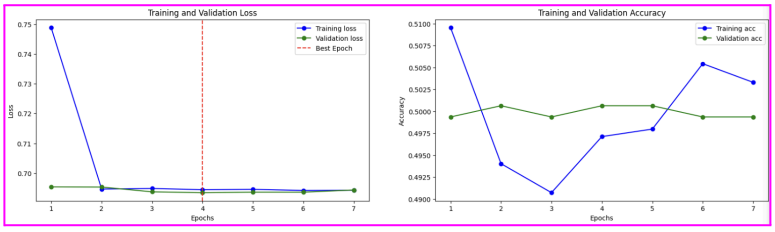
Fold 1



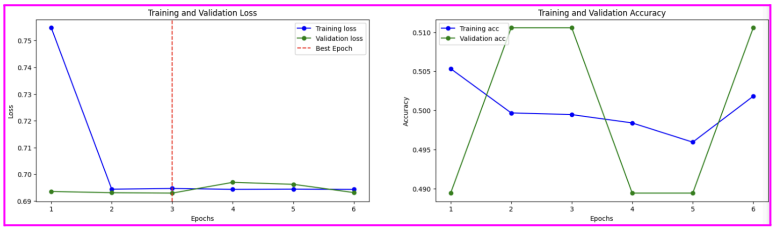
Fold 2



Fold 3



Fold 4

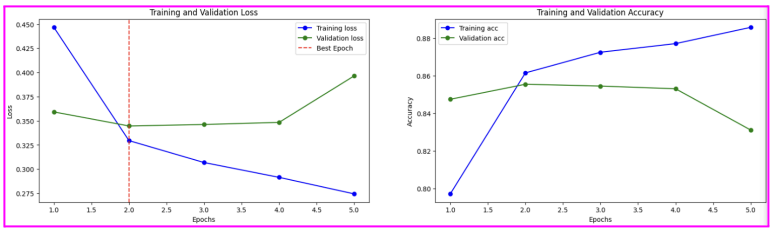


Training model: Hidden Layers=[32, 16, 8, 4], Learning Rate=0.001, Input Dim=1000

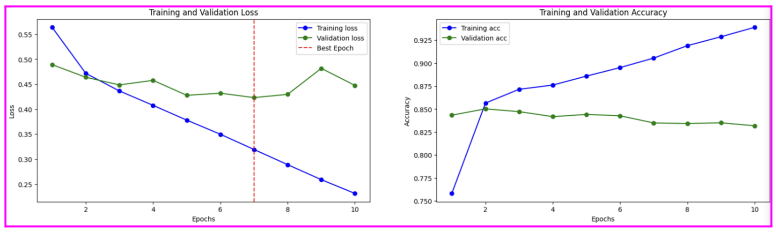
Avg Validation Accuracy for this setting: 0.8572, Avg Best Epoch: 3

Fold 1

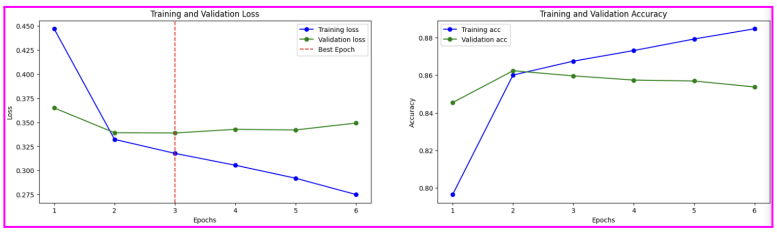
Fold 2

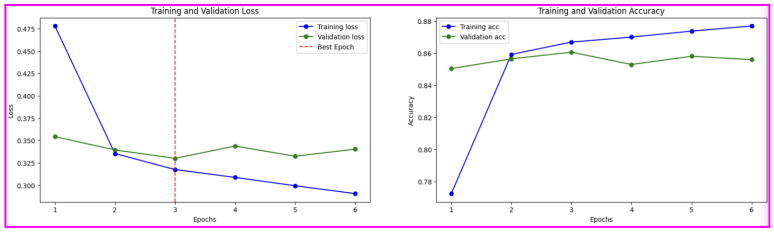


Fold 3



Fold 4

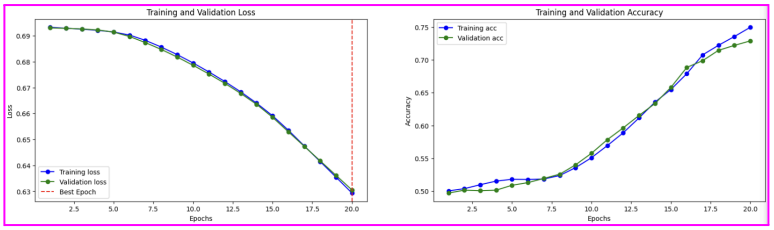




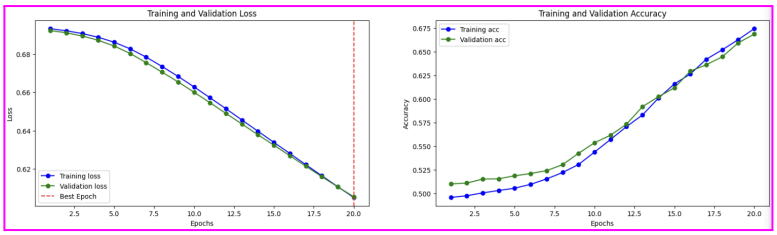
Training model: Hidden Layers=[32, 16, 8, 4], Learning Rate=1e-05, Input Dim=10000

Avg Validation Accuracy for this setting: 0.7542, Avg Best Epoch: 20

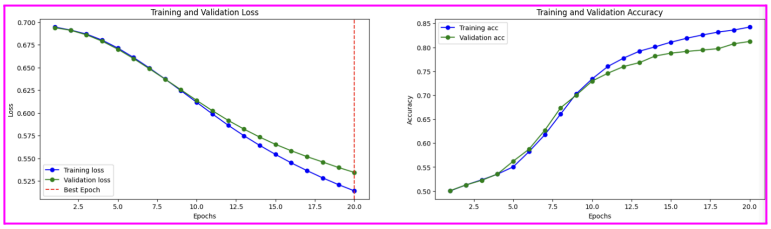
Fold 1



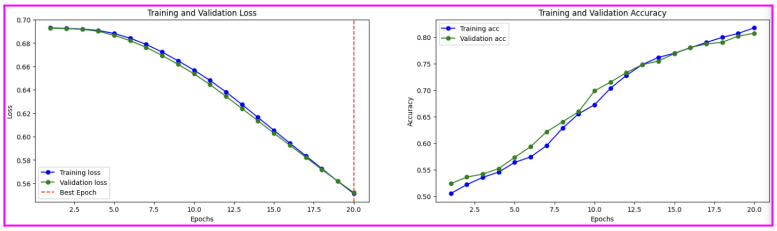
Fold 2



Fold 3



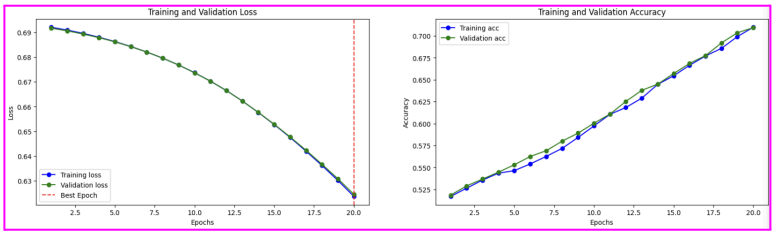
Fold 4



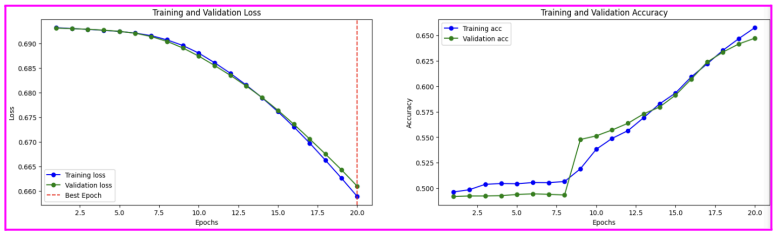
Training model: Hidden Layers=[32, 16, 8, 4], Learning Rate=1e-05, Input Dim=1000

Avg Validation Accuracy for this setting: 0.6969, Avg Best Epoch: 20

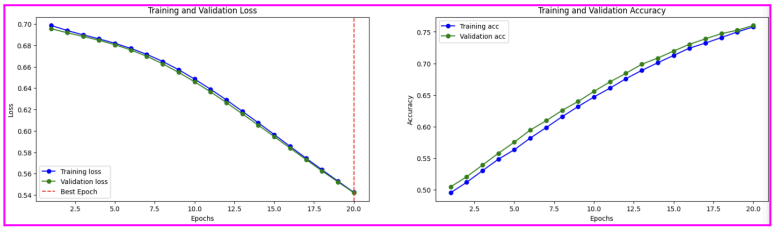
Fold 1



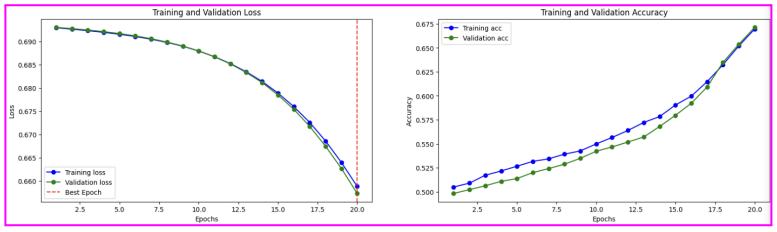
Fold 2



Fold 3



Fold 4



In general, the options with the best generalization were those with more layers ([32, 16, 8, 4]), a larger input size (10000), and a smaller learning rate (0.00001).

Top Hyperparameter Settings based on the accuracy averaged over the 4 folds (Validation Ranking)

Layers=**[16, 4]**, LR=**0.001**, Input=**10000**, Accuracy=0.8929, Best Epoch=2

Layers=**[1]**, LR=**0.001**, Input=**10000**, Accuracy=0.8864, Best Epoch=5

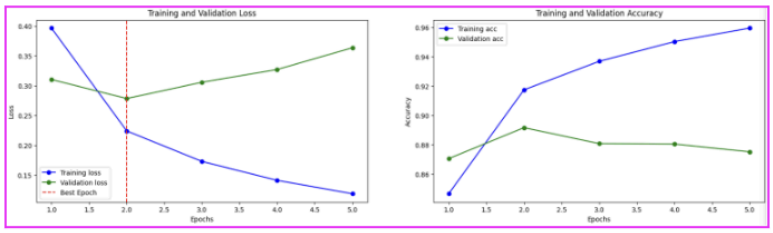
Layers=[**32, 16, 8, 4**], LR=**0.001**, Input=**10000**, Accuracy=0.8847, Best Epoch=2

The models with the best accuracy all appeared to be overfitted so I chose the epoch where the validation loss began to level out and the test accuracy began to pull away from the validation accuracy.

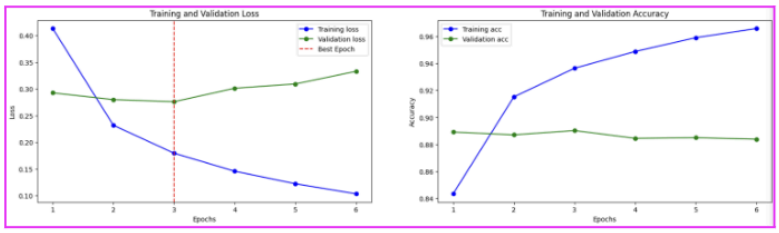
Training model: Hidden Layers=[16, 4], Learning Rate=0.001, Input Dim=10000

Avg Validation Accuracy for this setting: 0.8929, Avg Best Epoch: 2

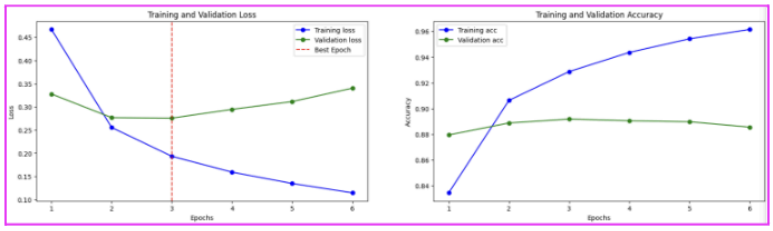
Fold 1



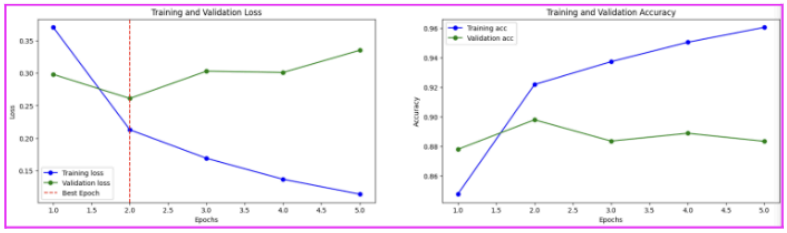
Fold 2



Fold 3



Fold 4



Performance: Accuracy = 89%, same as the validation accuracy.

```
results
```

```
[0.2902035117149353, 0.8855199813842773]
```

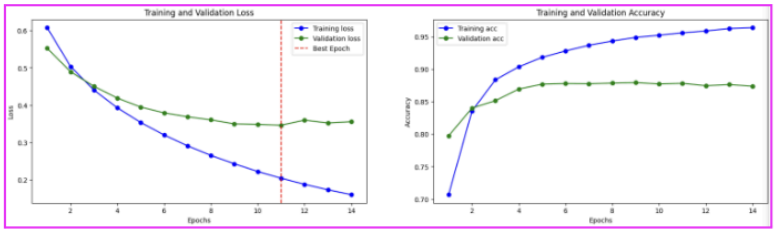
```
array([[0.14189863],
       [0.99921095],
       [0.95651525],
       ...,
       [0.0696547 ],
       [0.06919444],
       [0.3590736 ]],
```

As you can see, the network is very confident for many samples (>0.9 or <0.1) with only a few samples in between.

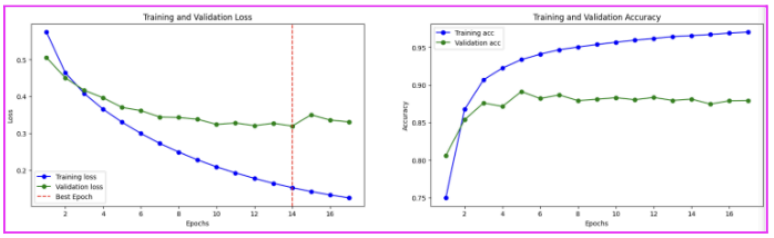
Training model: Hidden Layers=[1], Learning Rate=0.001, Input Dim=10000

Avg Validation Accuracy for this setting: 0.8864, Avg Best Epoch: 5

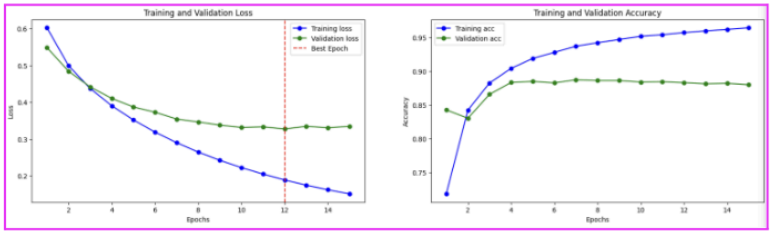
Fold 1



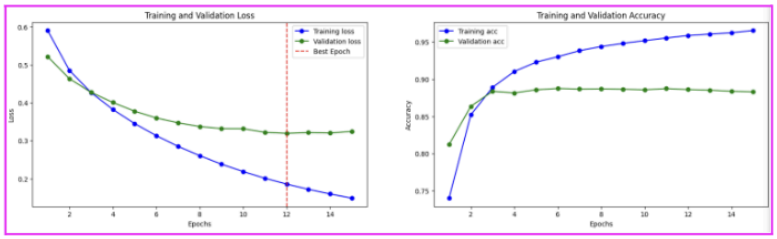
Fold 2



Fold 3



Fold 4



Performance: Accuracy = 88% which is only slightly less than the validation accuracy of 89%

```
results
```

```
[0.3645864725112915, 0.8773999810218811]
```

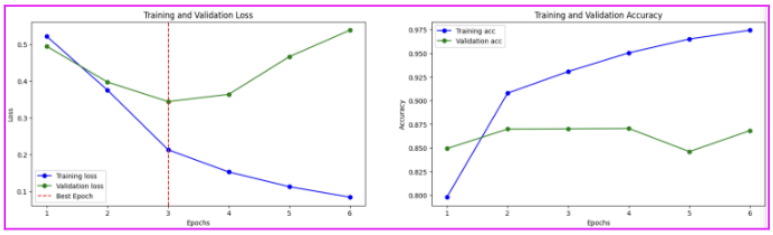
```
array([[0.24943775],
       [0.71941286],
       [0.52248615],
       ...,
       [0.15743315],
       [0.12823163],
       [0.3851985 ]],
```

As you can see, the network doesn't appear to be as confident with many samples between 0.3-0.7)

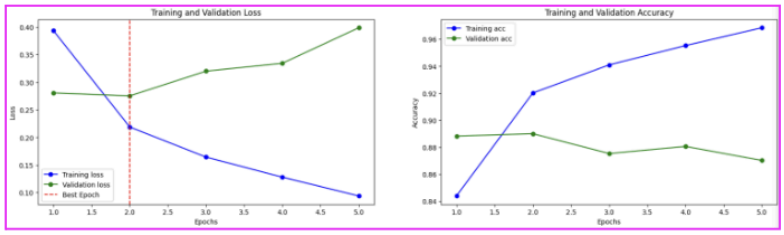
Training model: Hidden Layers=[32, 16, 8, 4], Learning Rate=0.001, Input Dim=10000

Avg Validation Accuracy for this setting: 0.8847, Avg Best Epoch: 2

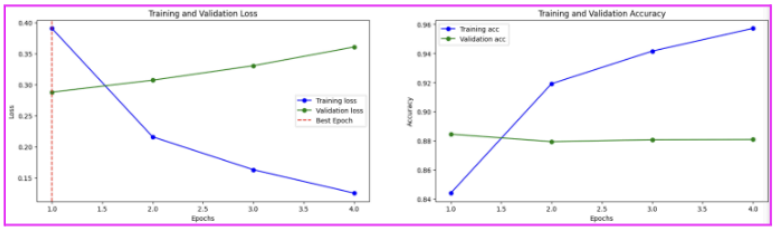
Fold 1



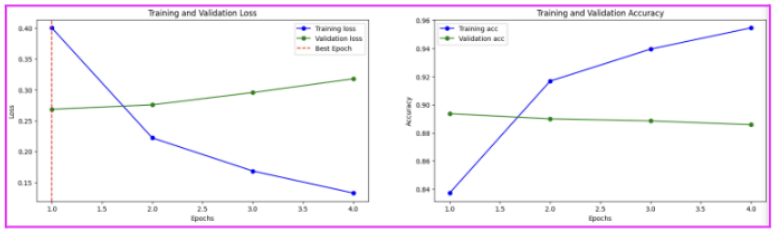
Fold 2



Fold 3



Fold 4



Performance: Accuracy = 0.88%, same as validation accuracy.

```
results
```

```
[0.30974918603897095, 0.8769999742507935]
```

```
array([[0.08396875],
       [0.999911  ],
       [0.47640088],
       ...,
       [0.08595792],
       [0.03928249],
       [0.19819334]]
```

As you can see, the network is very confident for some samples (e.g. 0.999 or 0.04) but less confident for others (e.g. 0.476)

PROBLEM #2

Set of 12 options to consider and validate:

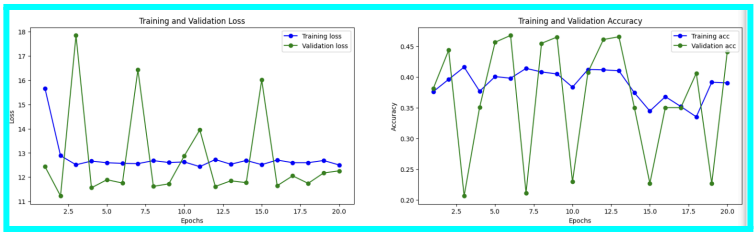
```
# Define hyperparameters to tune
news_hidden_layer_options = [
    [4],                # 1 layer, 4 nodes
    [16, 4],            # 2 layers, 16 nodes & 4 nodes
    [32, 16, 8],        # 3 layers, 32 → 16 → 8 nodes
]
news_learning_rate_options = [0.1, 0.01, 0.001]
news_batch_size_options = [128, 256, 512]
news_regularizers_options = [0.1, 0.01, 0.001]
```

Options illustrating underfitting, overfitting, and good generalization:

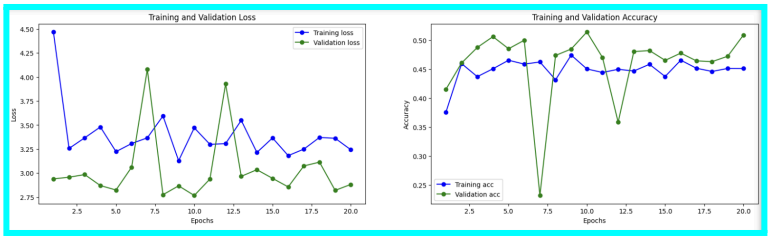
Underfitting was seen in options with fewer layers, larger regularizer (0.1), larger learning rate (0.1). The plots illustrate a small gap between training and validation performance and neither metric improves significantly with more epochs.

Underfitting

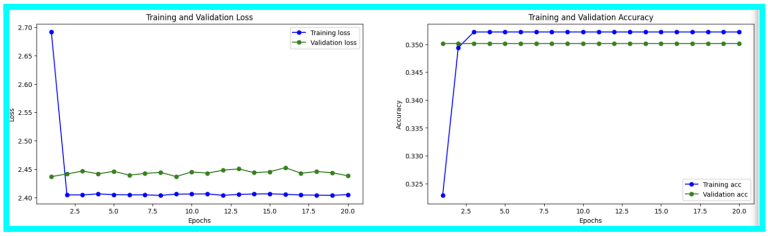
Hidden Layers=[4], LR=0.1, Batch Size=128, regularizer=0.1



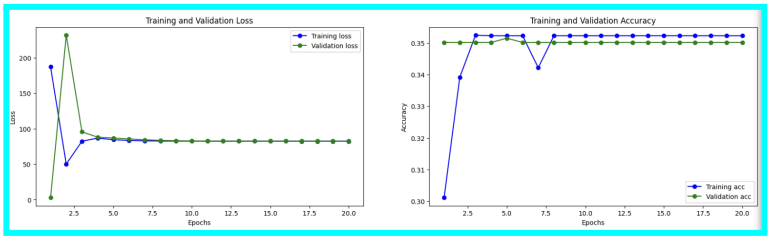
Hidden Layers=[4], LR=0.1, Batch Size=256, regularizer = 0.01



Hidden Layers=[16, 4], LR=0.1, Batch Size=128, regularizer=0.1



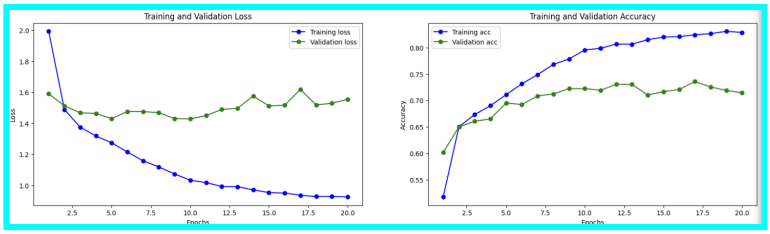
Hidden Layers=[32, 16, 8], LR=0.1, Batch Size=512, regularizer=0.1



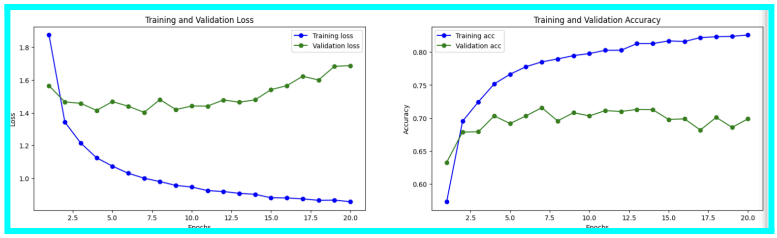
Overfitting was seen in options with a smaller regularizer (0.001). The plots illustrate a growing gap between training and validation performance where training metrics continue to improve while validation metrics plateau or degrade.

Overfitting

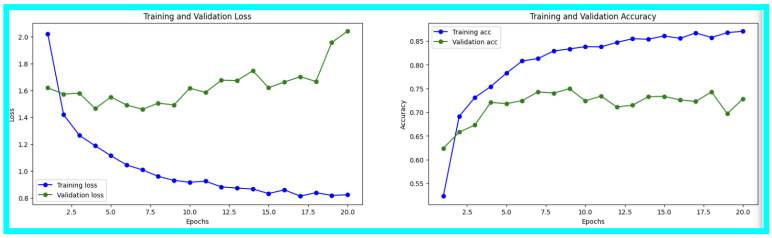
Hidden Layers=[4], LR=0.01, Batch Size=128, regularizer=0.001



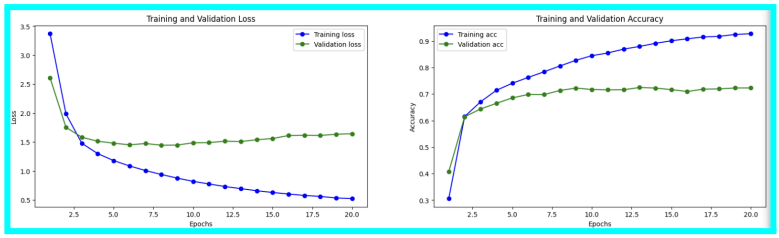
Hidden Layers=[16, 4], LR=0.01, Batch Size=128, regularizer=0.001



Hidden Layers=[32,16,8], LR=0.01, Batch Size=128, regularizer = 0.001



Hidden Layers=[32,16,8], LR=0.001, Batch Size=128, regularizer = 0.001

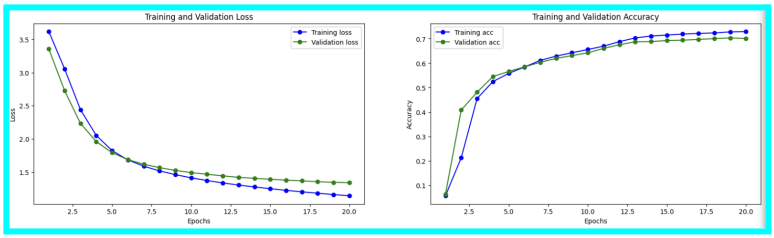


The options with the best generalization were those with more layers ([32, 16, 8]) and a smaller learning rate (0.001) The plots below demonstrate both training and validation metrics improve and converge until they plateau at good values.

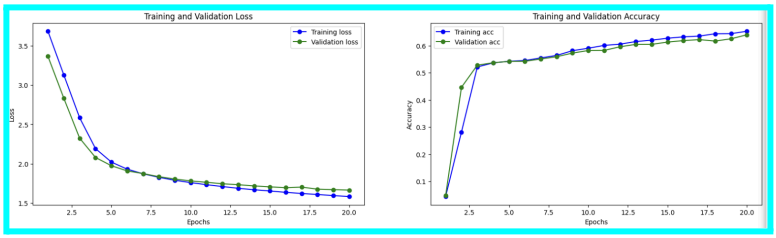
Good Generalization

Hidden Layers=[4], LR=0.001, Batch Size=128, regularizer = 0.001

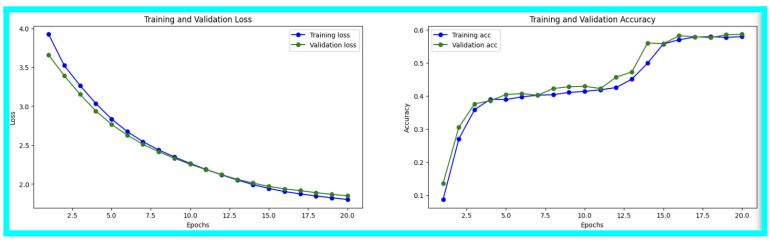
Hidden Layers=[16, 4], LR=0.001, Batch Size=128, regularizer = 0.01

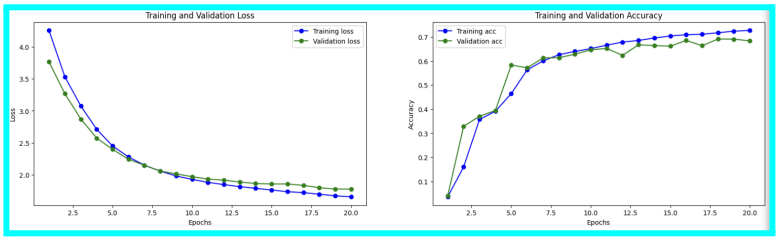


Hidden Layers=[16, 4], LR=0.001, Batch Size=512, regularizer=0.01



Hidden Layers=[32, 16, 8], LR=0.001, Batch Size=512, regularizer=0.01

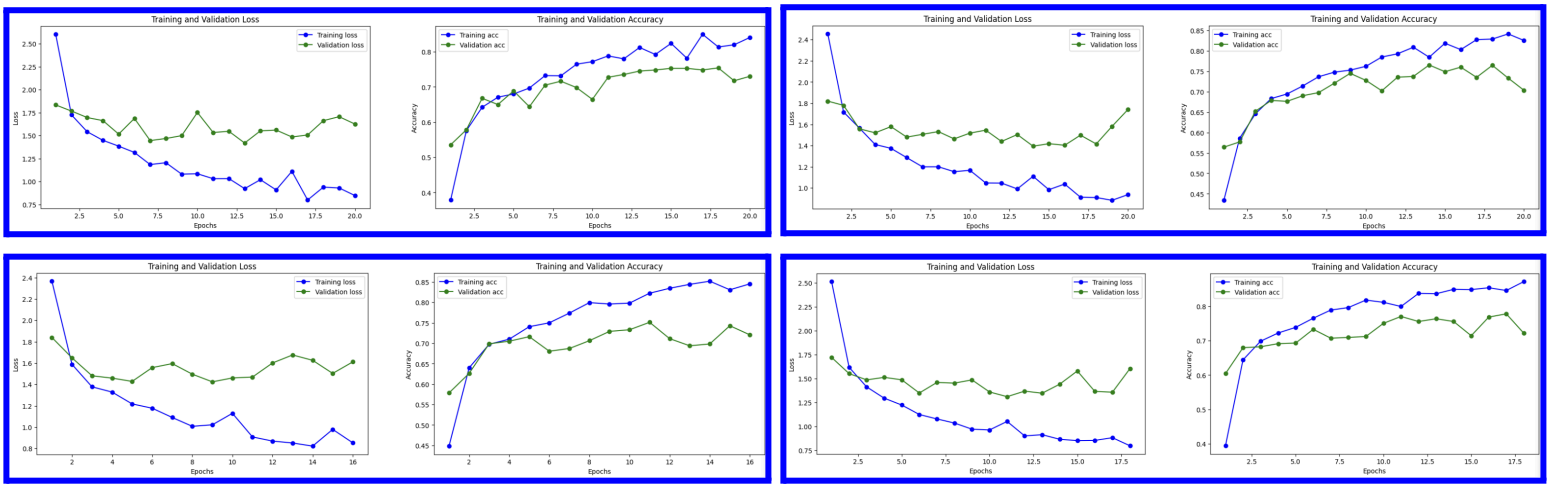




Top Hyperparameter Settings based on the accuracy averaged over the 4 folds (Validation Ranking)
Layers=[32, 16, 8], LR=0.01, batch_size=**512**, regularizer=0.001, Accuracy=0.7623
Layers=[32, 16, 8], LR=**0.01**, batch_size=**128**, regularizer=0.001, Accuracy=0.7605
Layers=[32, 16, 8], LR=**0.001,** batch_size=**128**, regularizer=0.001, Accuracy=0.7413

Like the imdb best hyperparameter settings, the options with the best accuracy all appear to be overfitted. So once again, I chose the epoch where the validation loss began to level out and the test accuracy began to pull away from the validation accuracy.

Layers=[32, 16, 8], LR=0.01, batch_size=512, regularizer=0.001, Accuracy=0.7623, best epoch = 5



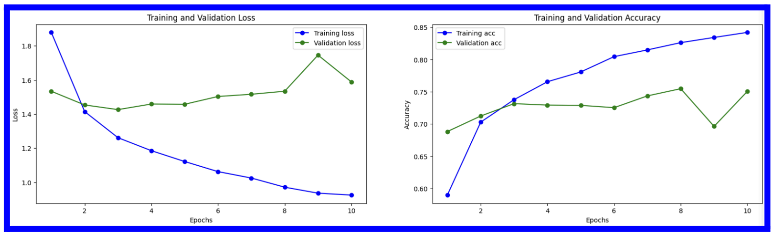
Performance: Accuracy = 88% which is significantly better than the validation accuracy of 76%

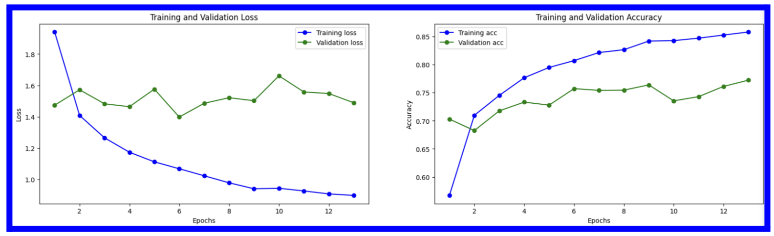
```
results
```

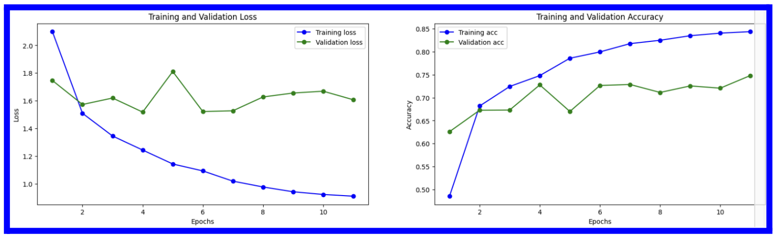
```
[0.30974918603897095, 0.8769999742507935]
```

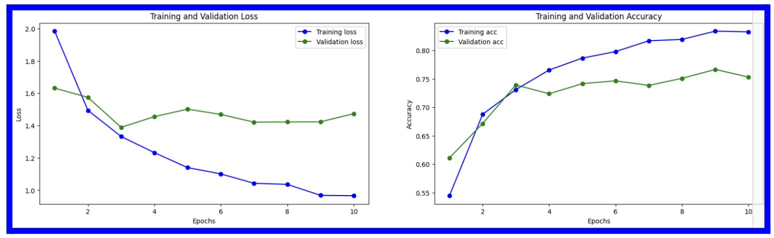
```
array([[8.56445622e-05, 4.28081639e-02, 5.55160705e-06,
        1.17410445e-05, 8.89988485e-07, 3.29604768e-06],
       [3.50066955e-04, 8.15151632e-01, 5.82974113e-04,
        1.53802000e-06, 2.90200433e-06, 1.73249107e-04],
       [7.72958854e-04, 7.24152684e-01, 2.14734767e-03,
        6.89178478e-06, 1.37599345e-05, 3.68236710e-04],
       ...,
       [4.42234959e-05, 1.68160796e-02, 8.41969666e-07,
        2.19003414e-05, 4.15692114e-07, 1.83577185e-06],
       [3.64404498e-03, 3.71500701e-01, 1.31716495e-02,
        3.34909244e-04, 1.94294116e-04, 6.87504769e-04],
       [6.24066405e-03, 2.28766978e-01, 4.06282321e-02,
        6.12961361e-04, 1.10586605e-03, 3.35597363e-03]],
```

Layers=[32, 16, 8], LR=0.01, batch_size=128, regularizer=0.001, Accuracy=0.7605, best epoch = 3







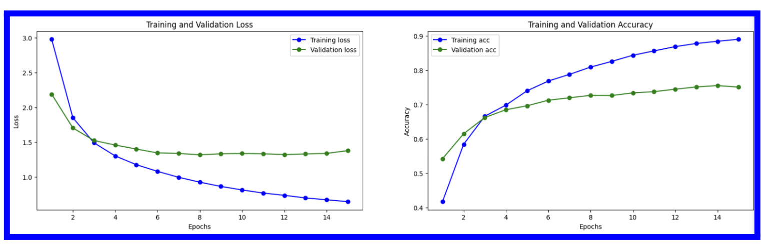


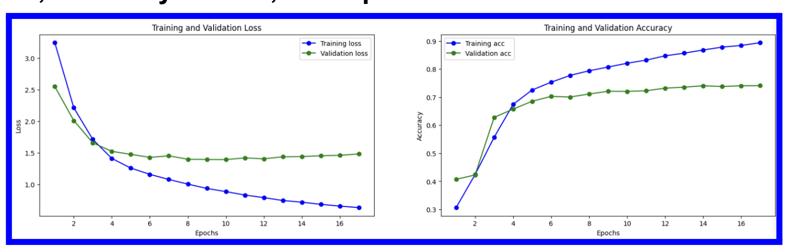
Performance: Accuracy = 0.73% which is slightly worse than the validation accuracy of 76%

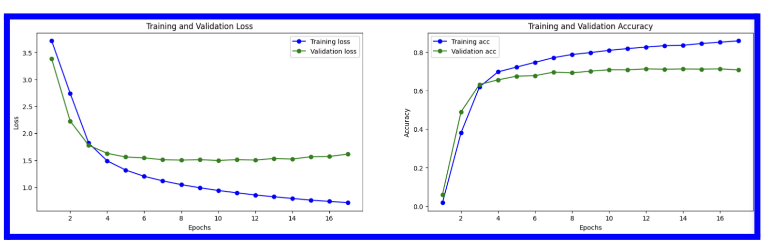
```
results
[1.392417550086975, 0.7315226793289185]
```

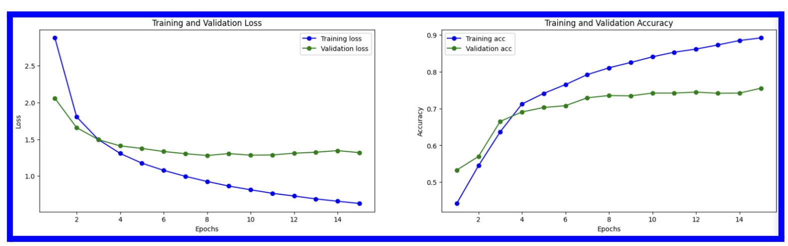
```
array([[2.95825402e-07, 8.66205955e-05, 1.22865793e-04,
         1.49838195e-06, 1.50309415e-05, 2.05926345e-10],
        [6.56168386e-02, 3.54681388e-02, 1.49964662e-02,
         1.17414747e-03, 2.96229846e-04, 2.22951663e-03],
        [2.23931461e-03, 4.90569234e-01, 2.21348740e-02,
         9.66696243e-05, 4.84103548e-05, 1.05371466e-03],
        ...,
        [2.78621201e-06, 8.59474298e-04, 1.08804507e-03,
         1.13618098e-05, 9.86702507e-05, 1.14573488e-08],
        [1.15348946e-03, 1.22784795e-02, 1.64755601e-02,
         8.06712778e-04, 1.40864321e-03, 7.62545187e-06],
        [5.76257566e-03, 2.40567386e-01, 1.47802413e-01,
         3.76560900e-04, 7.85422671e-05, 3.79834928e-05]],
```

Layers=[32, 16, 8], LR=0.001, batch_size=128, regularizer=0.001, Accuracy=0.7413, best epoch=4









Performance: Accuracy = 70% which is slightly worse than the validation accuracy of 74%

```
results
[1.4262136220932007, 0.700801432132721]
```

```
array([[2.60958877e-05, 1.51621953e-05, 1.20951128e-07,
         3.35221957e-05, 5.41410554e-06, 2.98007762e-06],
        [8.70595220e-03, 2.10271075e-01, 3.84616368e-02,
         1.01698664e-04, 3.12489647e-05, 1.51924731e-04],
        [1.57440379e-02, 1.35708719e-01, 4.48180884e-02,
         4.27692570e-03, 1.46066351e-03, 2.26336974e-03],
        ...,
        [6.93599504e-05, 1.60146999e-04, 2.38511166e-06,
         3.27714806e-04, 4.83546028e-05, 2.48255747e-05],
        [1.72600138e-03, 1.80407404e-03, 6.41465303e-05,
         8.00862967e-04, 2.74599472e-04, 2.44319177e-04],
        [1.03018875e-03, 7.44916275e-02, 1.69521000e-03,
         1.38882641e-03, 1.72279091e-04, 2.41441870e-04]]
```