Assignment-1

Harshith Reddy Suram

Introduction:
In this assignment, the focus is on improving the existing similar neural network model on IMDB dataset for sentiment classification. Therefore, in an attempt to find out how the modification of architectural changes and the training strategies affects the validation of the model and consequently the test accuracy, we shall make the following changes In our experiment. The modifications involve having more or less hidden layers, manipulating the number of hidden neurons, using different loss functions, varying activation functions, and using or not using the technique of regularization.

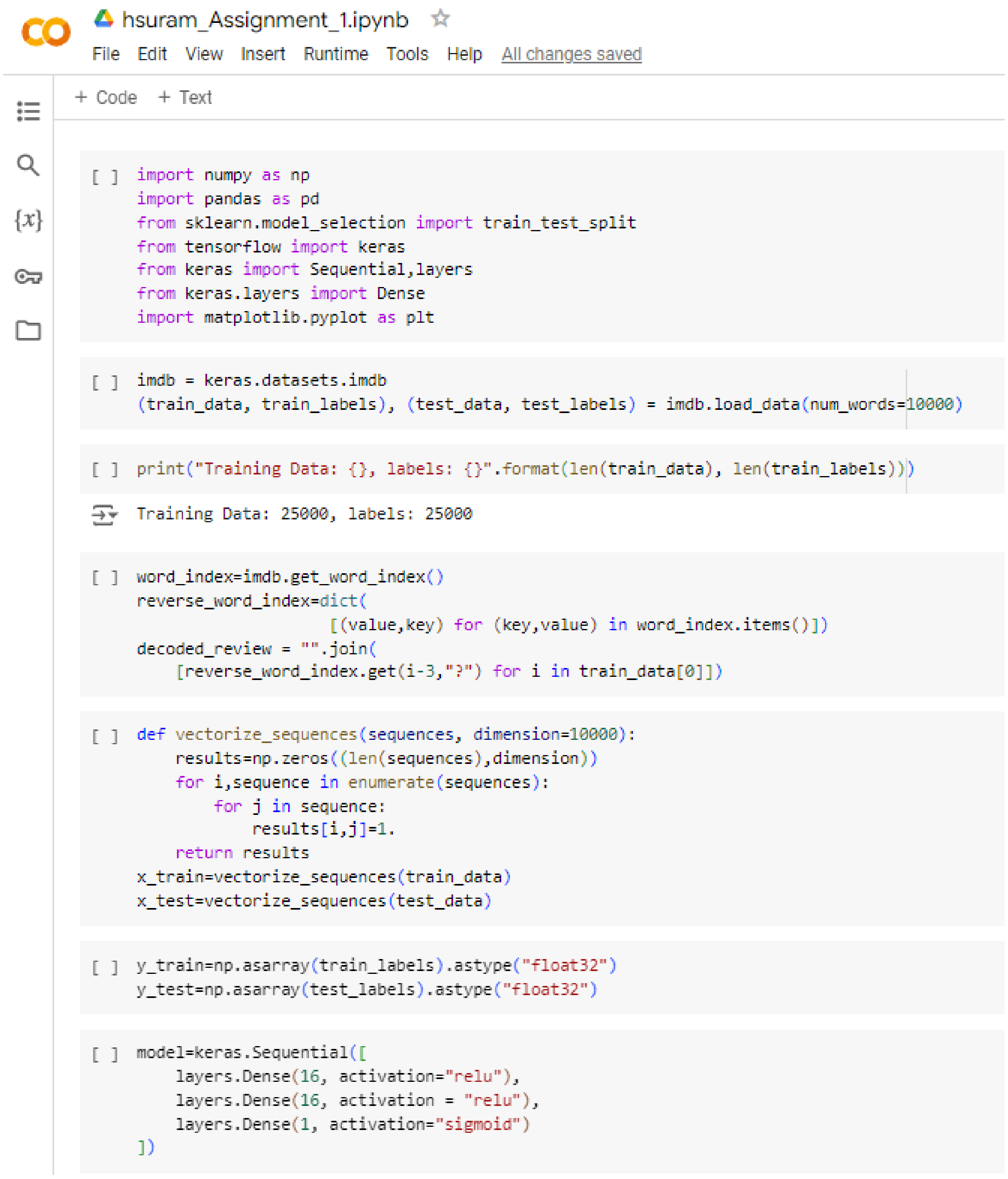
Problem Statement:
The goal of this assignment is to investigate various adjustments to an existing neural network model that might improve its performance on the IMDB sentiment analysis assignment. Specifically, the assignment includes:

1. Modifying the number of hidden layers to show how employing one or three hidden layers affects the model's validation and test accuracy.
2. Experimenting with different numbers of hidden units per layer (e.g., 32 and 64) to determine the best size for boosting model performance.
3. To examine the influence on model correctness, use the mean squared error (mse) loss function rather than the binary_crossentropy loss function.
4. To study performance differences, use the tanh activation function instead of the ReLU activation function.
5. Using regularization techniques such as dropout and L2 regularization to improve the model's generalizability and reduce overfitting.

Methodology:

We will train a neural network model to perform sentiment analysis on the IMDB movie review dataset using Keras. The process begins with importing necessary libraries and loading the IMDB dataset, which is split into training and test sets. The data is then preprocessed by converting reviews into binary vectors using a one-hot encoding method. A sequential neural network model is defined with two hidden layers, each containing 16 units with ReLU activation, and an output layer with a sigmoid activation function. The model is compiled with the Adam optimizer and binary_crossentropy loss. A validation set is created by splitting the training data, and the model is trained for 50 epochs. Training history, including loss and accuracy for both training and validation sets, is plotted to visualize the model's performance. The final model summary provides details on the structure and parameters of the neural network.

Code:



hsuram_Assignment_1.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
[ ] import numpy as np
    import pandas as pd
    from sklearn.model_selection import train_test_split
    from tensorflow import keras
    from keras import Sequential,layers
    from keras.layers import Dense
    import matplotlib.pyplot as plt
```

```
[ ] imdb = keras.datasets.imdb
    (train_data, train_labels), (test_data, test_labels) = imdb.load_data(num_words=10000)
```

```
[ ] print("Training Data: {}, labels: {}".format(len(train_data), len(train_labels)))
```

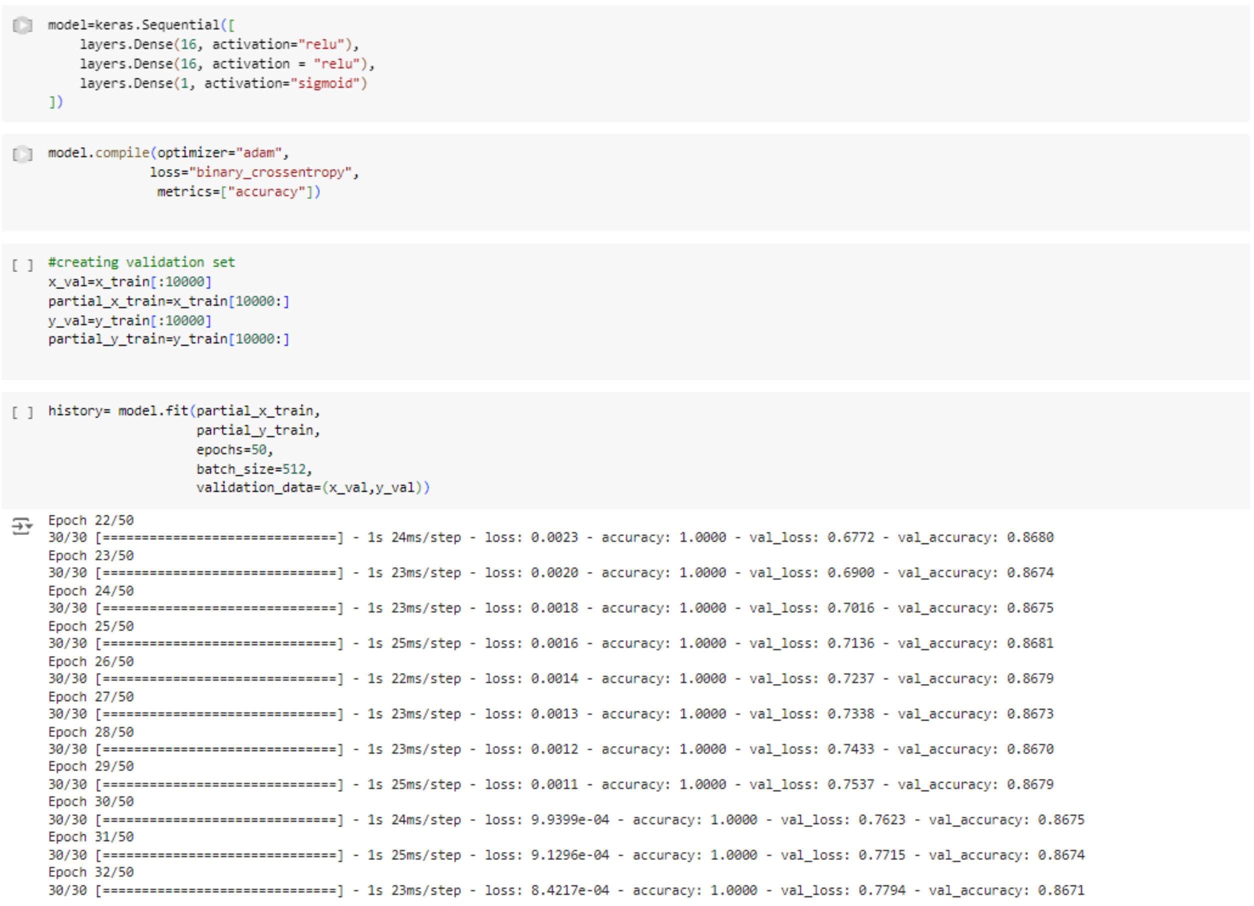
```
Training Data: 25000, labels: 25000
```

```
[ ] word_index=imdb.get_word_index()
    reverse_word_index=dict(
                        [(value,key) for (key,value) in word_index.items()])
    decoded_review = "".join(
        [reverse_word_index.get(i-3,"?") for i in train_data[0]])
```

```
[ ] def vectorize_sequences(sequences, dimension=10000):
        results=np.zeros((len(sequences),dimension))
        for i,sequence in enumerate(sequences):
            for j in sequence:
                results[i,j]=1.
        return results
    x_train=vectorize_sequences(train_data)
    x_test=vectorize_sequences(test_data)
```

```
[ ] y_train=np.asarray(train_labels).astype("float32")
    y_test=np.asarray(test_labels).astype("float32")
```

```
[ ] model=keras.Sequential([
        layers.Dense(16, activation="relu"),
        layers.Dense(16, activation = "relu"),
        layers.Dense(1, activation="sigmoid")
    ])
```



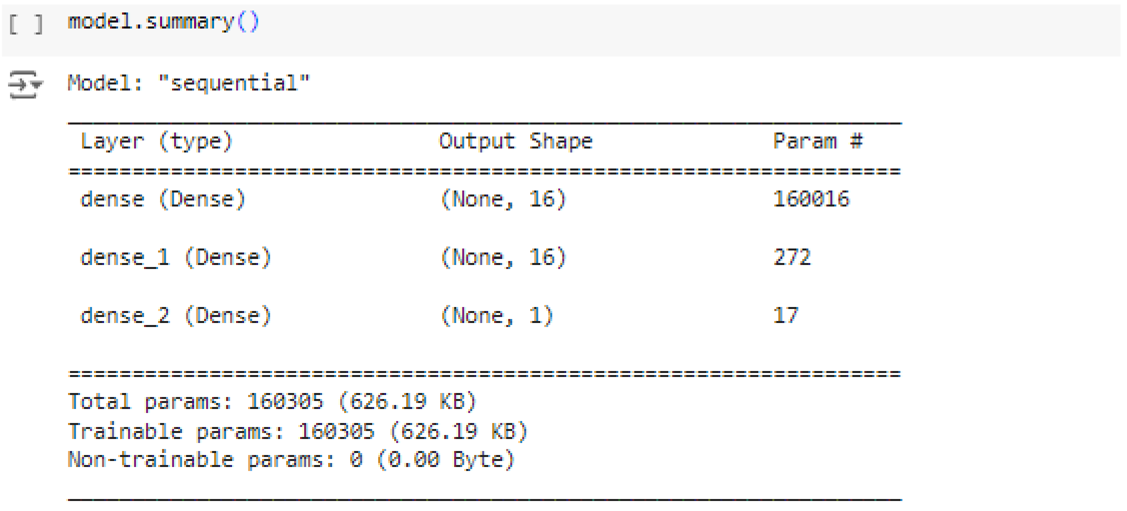
```
model=keras.Sequential([
    layers.Dense(16, activation="relu"),
    layers.Dense(16, activation = "relu"),
    layers.Dense(1, activation="sigmoid")
])
```

```
model.compile(optimizer="adam",
             loss="binary_crossentropy",
              metrics=["accuracy"])
```

```
[ ] #creating validation set
    x_val=x_train[:10000]
    partial_x_train=x_train[10000:]
    y_val=y_train[:10000]
    partial_y_train=y_train[10000:]
```

```
[ ] history= model.fit(partial_x_train,
                       partial_y_train,
                       epochs=50,
                       batch_size=512,
                       validation_data=(x_val,y_val))
```

```
Epoch 22/50
30/30 [==============================] - 1s 24ms/step - loss: 0.0023 - accuracy: 1.0000 - val_loss: 0.6772 - val_accuracy: 0.8680
Epoch 23/50
30/30 [==============================] - 1s 23ms/step - loss: 0.0020 - accuracy: 1.0000 - val_loss: 0.6900 - val_accuracy: 0.8674
Epoch 24/50
30/30 [==============================] - 1s 23ms/step - loss: 0.0018 - accuracy: 1.0000 - val_loss: 0.7016 - val_accuracy: 0.8675
Epoch 25/50
30/30 [==============================] - 1s 25ms/step - loss: 0.0016 - accuracy: 1.0000 - val_loss: 0.7136 - val_accuracy: 0.8681
Epoch 26/50
30/30 [==============================] - 1s 22ms/step - loss: 0.0014 - accuracy: 1.0000 - val_loss: 0.7237 - val_accuracy: 0.8679
Epoch 27/50
30/30 [==============================] - 1s 23ms/step - loss: 0.0013 - accuracy: 1.0000 - val_loss: 0.7338 - val_accuracy: 0.8673
Epoch 28/50
30/30 [==============================] - 1s 23ms/step - loss: 0.0012 - accuracy: 1.0000 - val_loss: 0.7433 - val_accuracy: 0.8670
Epoch 29/50
30/30 [==============================] - 1s 25ms/step - loss: 0.0011 - accuracy: 1.0000 - val_loss: 0.7537 - val_accuracy: 0.8679
Epoch 30/50
30/30 [==============================] - 1s 24ms/step - loss: 9.9399e-04 - accuracy: 1.0000 - val_loss: 0.7623 - val_accuracy: 0.8675
Epoch 31/50
30/30 [==============================] - 1s 25ms/step - loss: 9.1296e-04 - accuracy: 1.0000 - val_loss: 0.7715 - val_accuracy: 0.8674
Epoch 32/50
30/30 [==============================] - 1s 23ms/step - loss: 8.4217e-04 - accuracy: 1.0000 - val_loss: 0.7794 - val_accuracy: 0.8671
```



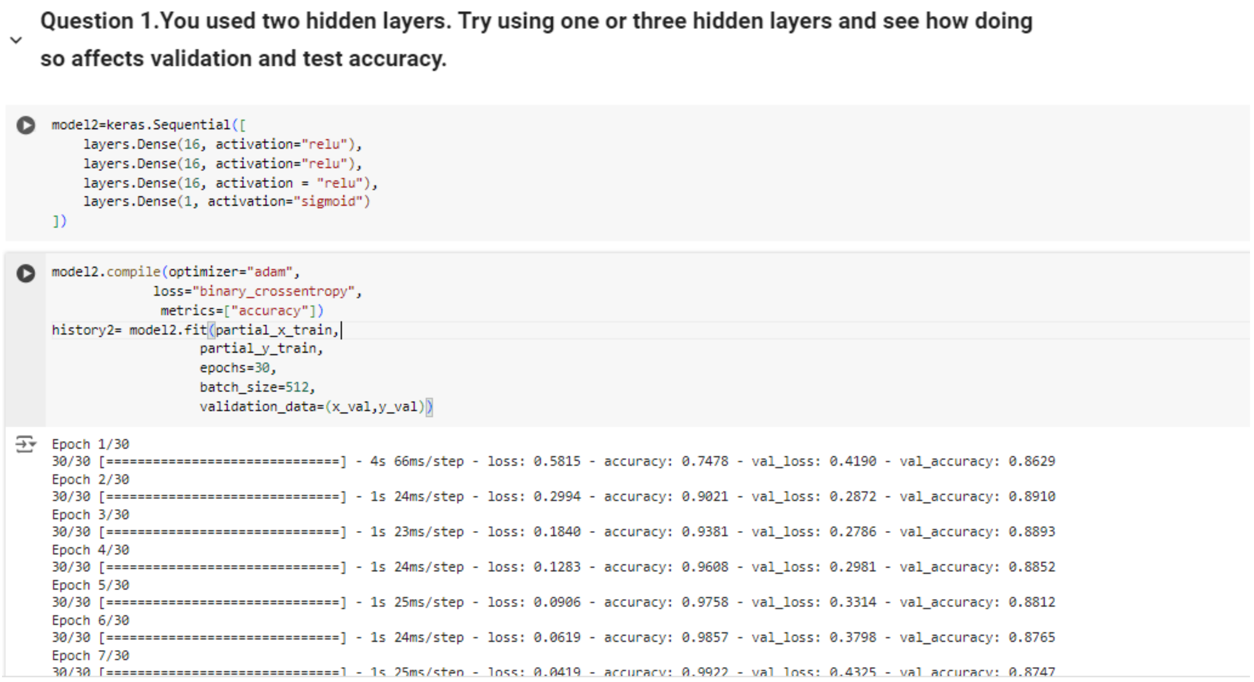
```
[ ] model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 16)                160016

 dense_1 (Dense)             (None, 16)                272

 dense_2 (Dense)             (None, 1)                 17

=================================================================
Total params: 160305 (626.19 KB)
Trainable params: 160305 (626.19 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Question-1:

Enhancements of the prior neural network model for sentiment analysis of the IMDB database involve the addition of a new hidden layer. The new model adopted here (model2) has three hidden layers, the first of which has 16 units, the second has 16 units as well, the third layer also has 16 units all those layers have ReLU activation functions in them and there is an output layer with sigmoid activations. The choice of correct data mining model is made when the application of the Adam optimizer and the binary cross entropy as the loss function is used, with accuracy as a measure of the quality. Specifically, it goes through 30 epochs training on the training data with the batch size of 512, while the performance is evaluated with the help of a validation dataset. As expected, the summary of the new model reveals the fact that the promotion of an additional hidden layer has led to the enhancement of the total number of parameters in an attempt to enhance the model’s ability to recognize intricate relationships within the given data.

Code:



Question 1.You used two hidden layers. Try using one or three hidden layers and see how doing so affects validation and test accuracy.

```
model2=keras.Sequential([
    layers.Dense(16, activation="relu"),
    layers.Dense(16, activation="relu"),
    layers.Dense(16, activation = "relu"),
    layers.Dense(1, activation="sigmoid")
])
```

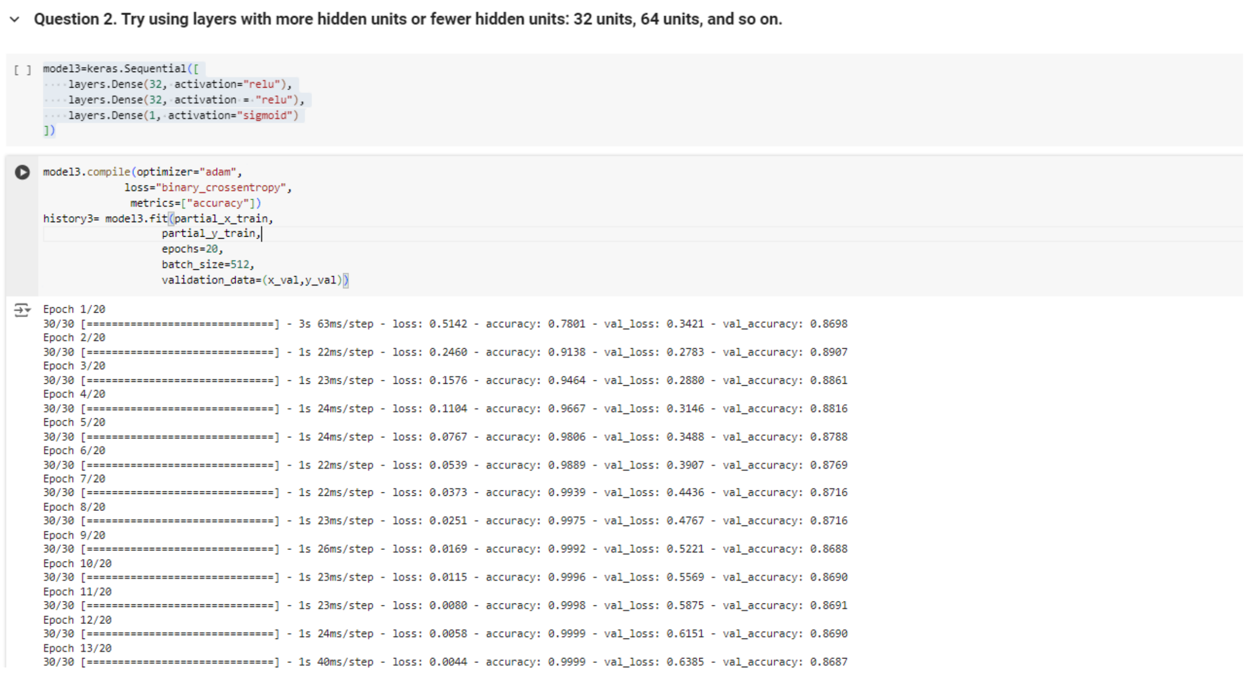
```
model2.compile(optimizer="adam",
             loss="binary_crossentropy",
              metrics=["accuracy"])
history2= model2.fit(partial_x_train,
                   partial_y_train,
                   epochs=30,
                   batch_size=512,
                   validation_data=(x_val,y_val))
```

```
Epoch 1/30
30/30 [==============================] - 4s 66ms/step - loss: 0.5815 - accuracy: 0.7478 - val_loss: 0.4190 - val_accuracy: 0.8629
Epoch 2/30
30/30 [==============================] - 1s 24ms/step - loss: 0.2994 - accuracy: 0.9021 - val_loss: 0.2872 - val_accuracy: 0.8910
Epoch 3/30
30/30 [==============================] - 1s 23ms/step - loss: 0.1840 - accuracy: 0.9381 - val_loss: 0.2786 - val_accuracy: 0.8893
Epoch 4/30
30/30 [==============================] - 1s 24ms/step - loss: 0.1283 - accuracy: 0.9608 - val_loss: 0.2981 - val_accuracy: 0.8852
Epoch 5/30
30/30 [==============================] - 1s 25ms/step - loss: 0.0906 - accuracy: 0.9758 - val_loss: 0.3314 - val_accuracy: 0.8812
Epoch 6/30
30/30 [==============================] - 1s 24ms/step - loss: 0.0619 - accuracy: 0.9857 - val_loss: 0.3798 - val_accuracy: 0.8765
Epoch 7/30
30/30 [==============================] - 1s 25ms/step - loss: 0.0419 - accuracy: 0.9922 - val_loss: 0.4325 - val_accuracy: 0.8747
```

Question-2:

We are using the IMDB dataset to train a new neural network model (model3) for sentiment analysis. In this model, the architecture consists of two hidden layers, each with 32 units and ReLU activation functions, followed by a single output layer with a sigmoid function. The model is built with the Adam optimizer and the binary_crossentropy loss function, with accuracy as a performance parameter. Training takes place on the partial training set for 20 epochs with a batch size of 512, and a validation set is utilized to check performance throughout training. This model seeks to assess the impact of increasing the number of units per hidden layer from 16 to 32 on the model's performance.

Code:


Question 2. Try using layers with more hidden units or fewer hidden units: 32 units, 64 units, and so on.

```
model3=keras.Sequential([
    layers.Dense(32, activation="relu"),
    layers.Dense(32, activation = "relu"),
    layers.Dense(1, activation="sigmoid")
])
```

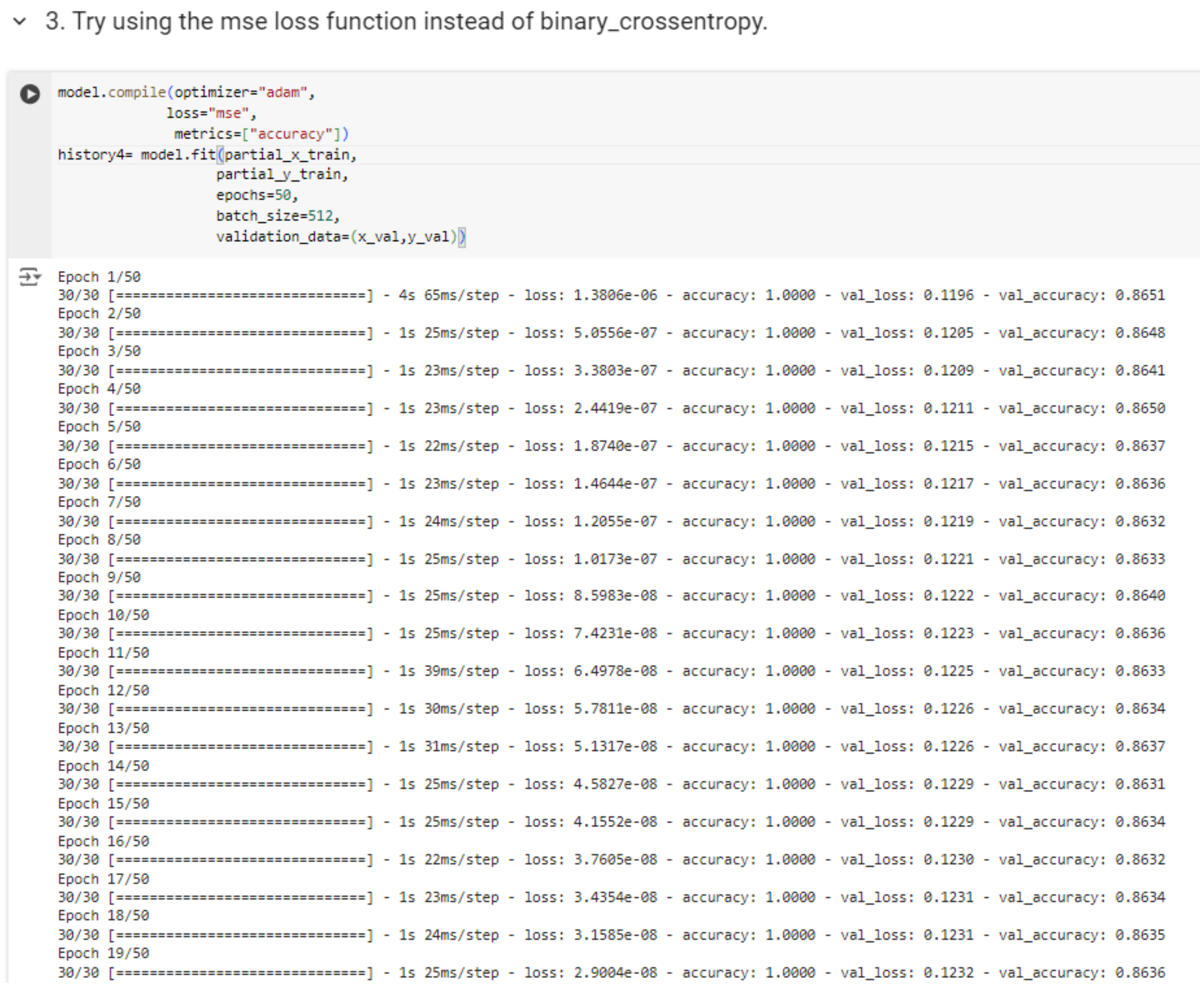
```
model3.compile(optimizer="adam",
             loss="binary_crossentropy",
              metrics=["accuracy"])
history3= model3.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val,y_val))
```

```
Epoch 1/20
30/30 [==============================] - 3s 63ms/step - loss: 0.5142 - accuracy: 0.7801 - val_loss: 0.3421 - val_accuracy: 0.8698
Epoch 2/20
30/30 [==============================] - 1s 22ms/step - loss: 0.2460 - accuracy: 0.9138 - val_loss: 0.2783 - val_accuracy: 0.8907
Epoch 3/20
30/30 [==============================] - 1s 23ms/step - loss: 0.1576 - accuracy: 0.9464 - val_loss: 0.2880 - val_accuracy: 0.8861
Epoch 4/20
30/30 [==============================] - 1s 24ms/step - loss: 0.1104 - accuracy: 0.9667 - val_loss: 0.3146 - val_accuracy: 0.8816
Epoch 5/20
30/30 [==============================] - 1s 24ms/step - loss: 0.0767 - accuracy: 0.9806 - val_loss: 0.3488 - val_accuracy: 0.8788
Epoch 6/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0539 - accuracy: 0.9889 - val_loss: 0.3907 - val_accuracy: 0.8769
Epoch 7/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0373 - accuracy: 0.9939 - val_loss: 0.4436 - val_accuracy: 0.8716
Epoch 8/20
30/30 [==============================] - 1s 23ms/step - loss: 0.0251 - accuracy: 0.9975 - val_loss: 0.4767 - val_accuracy: 0.8716
Epoch 9/20
30/30 [==============================] - 1s 26ms/step - loss: 0.0169 - accuracy: 0.9992 - val_loss: 0.5221 - val_accuracy: 0.8688
Epoch 10/20
30/30 [==============================] - 1s 23ms/step - loss: 0.0115 - accuracy: 0.9996 - val_loss: 0.5569 - val_accuracy: 0.8690
Epoch 11/20
30/30 [==============================] - 1s 23ms/step - loss: 0.0080 - accuracy: 0.9998 - val_loss: 0.5875 - val_accuracy: 0.8691
Epoch 12/20
30/30 [==============================] - 1s 24ms/step - loss: 0.0058 - accuracy: 0.9999 - val_loss: 0.6151 - val_accuracy: 0.8690
Epoch 13/20
30/30 [==============================] - 1s 40ms/step - loss: 0.0044 - accuracy: 0.9999 - val_loss: 0.6385 - val_accuracy: 0.8687
```

Question-3:

We altered the original neural network model for sentiment analysis on the IMDB dataset by altering the loss function to mean squared error (mse). The model architecture stays unchanged, with two hidden layers, each with 16 units with ReLU activation functions, and an output layer with a sigmoid activation function. The model is built utilizing the Adam optimizer, with mse as the loss function and accuracy as the performance metric. Training is carried out on the partial training set for 50 epochs with a batch size of 512, and the validation set is used to assess the model's performance while training. This experiment will look into how employing mse instead of binary_crossentropy affects the model's training and validation accuracy.

Code:



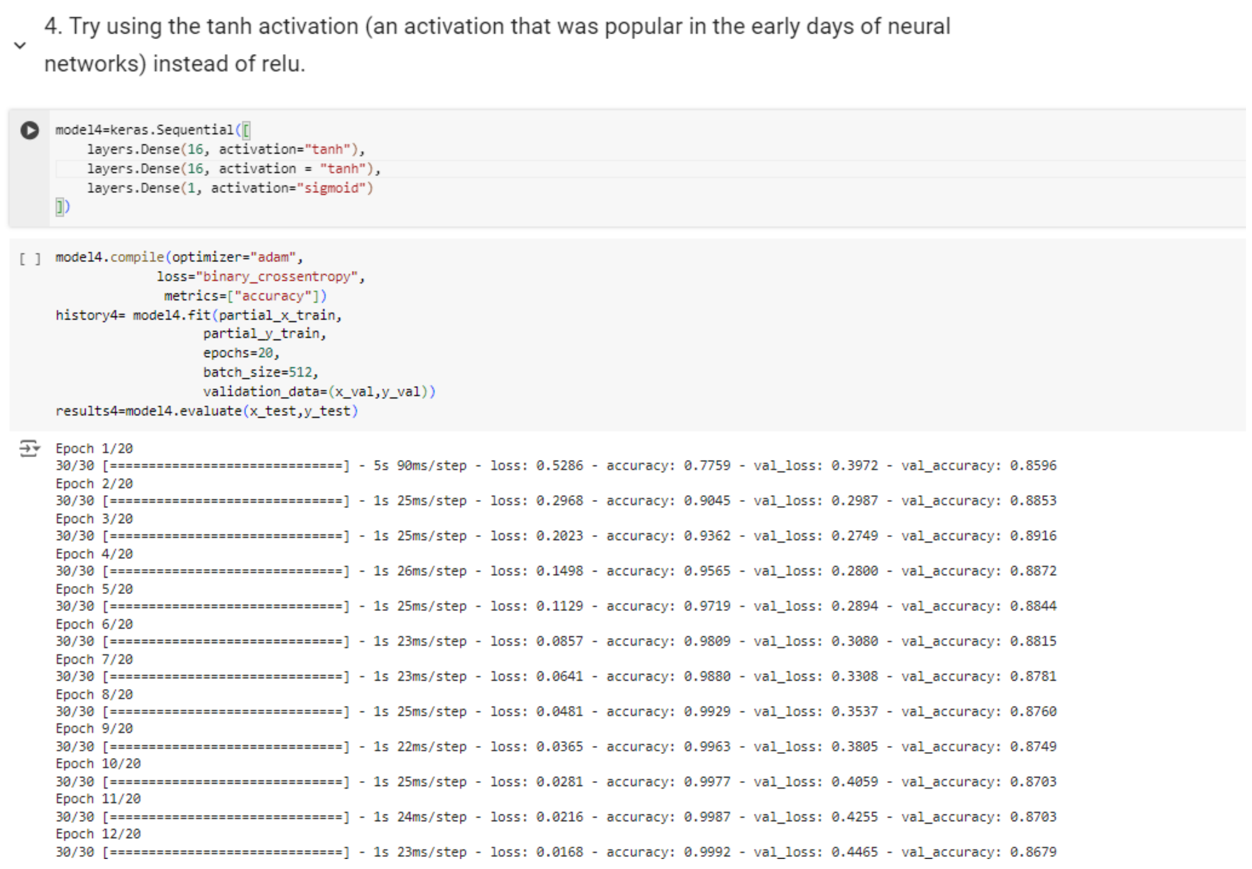
3. Try using the mse loss function instead of binary_crossentropy.

```
model.compile(optimizer="adam",
             loss="mse",
              metrics=["accuracy"])
history4= model.fit(partial_x_train,
                   partial_y_train,
                   epochs=50,
                   batch_size=512,
                   validation_data=(x_val,y_val))
```

```
Epoch 1/50
30/30 [==============================] - 4s 65ms/step - loss: 1.3806e-06 - accuracy: 1.0000 - val_loss: 0.1196 - val_accuracy: 0.8651
Epoch 2/50
30/30 [==============================] - 1s 25ms/step - loss: 5.0556e-07 - accuracy: 1.0000 - val_loss: 0.1205 - val_accuracy: 0.8648
Epoch 3/50
30/30 [==============================] - 1s 23ms/step - loss: 3.3803e-07 - accuracy: 1.0000 - val_loss: 0.1209 - val_accuracy: 0.8641
Epoch 4/50
30/30 [==============================] - 1s 23ms/step - loss: 2.4419e-07 - accuracy: 1.0000 - val_loss: 0.1211 - val_accuracy: 0.8650
Epoch 5/50
30/30 [==============================] - 1s 22ms/step - loss: 1.8740e-07 - accuracy: 1.0000 - val_loss: 0.1215 - val_accuracy: 0.8637
Epoch 6/50
30/30 [==============================] - 1s 23ms/step - loss: 1.4644e-07 - accuracy: 1.0000 - val_loss: 0.1217 - val_accuracy: 0.8636
Epoch 7/50
30/30 [==============================] - 1s 24ms/step - loss: 1.2055e-07 - accuracy: 1.0000 - val_loss: 0.1219 - val_accuracy: 0.8632
Epoch 8/50
30/30 [==============================] - 1s 25ms/step - loss: 1.0173e-07 - accuracy: 1.0000 - val_loss: 0.1221 - val_accuracy: 0.8633
Epoch 9/50
30/30 [==============================] - 1s 25ms/step - loss: 8.5983e-08 - accuracy: 1.0000 - val_loss: 0.1222 - val_accuracy: 0.8640
Epoch 10/50
30/30 [==============================] - 1s 25ms/step - loss: 7.4231e-08 - accuracy: 1.0000 - val_loss: 0.1223 - val_accuracy: 0.8636
Epoch 11/50
30/30 [==============================] - 1s 39ms/step - loss: 6.4978e-08 - accuracy: 1.0000 - val_loss: 0.1225 - val_accuracy: 0.8633
Epoch 12/50
30/30 [==============================] - 1s 30ms/step - loss: 5.7811e-08 - accuracy: 1.0000 - val_loss: 0.1226 - val_accuracy: 0.8634
Epoch 13/50
30/30 [==============================] - 1s 31ms/step - loss: 5.1317e-08 - accuracy: 1.0000 - val_loss: 0.1226 - val_accuracy: 0.8637
Epoch 14/50
30/30 [==============================] - 1s 25ms/step - loss: 4.5827e-08 - accuracy: 1.0000 - val_loss: 0.1229 - val_accuracy: 0.8631
Epoch 15/50
30/30 [==============================] - 1s 25ms/step - loss: 4.1552e-08 - accuracy: 1.0000 - val_loss: 0.1229 - val_accuracy: 0.8634
Epoch 16/50
30/30 [==============================] - 1s 22ms/step - loss: 3.7605e-08 - accuracy: 1.0000 - val_loss: 0.1230 - val_accuracy: 0.8632
Epoch 17/50
30/30 [==============================] - 1s 23ms/step - loss: 3.4354e-08 - accuracy: 1.0000 - val_loss: 0.1231 - val_accuracy: 0.8634
Epoch 18/50
30/30 [==============================] - 1s 24ms/step - loss: 3.1585e-08 - accuracy: 1.0000 - val_loss: 0.1231 - val_accuracy: 0.8635
Epoch 19/50
30/30 [==============================] - 1s 25ms/step - loss: 2.9004e-08 - accuracy: 1.0000 - val_loss: 0.1232 - val_accuracy: 0.8636
```

Question-4:

This code creates and trains a new neural network model (model4) for sentiment analysis using the IMDB dataset, with a focus on modifying the activation function of the hidden layers. The architecture is made up of two hidden layers, each with 16 units and employing the tanh activation function, followed by an output layer with the sigmoid activation function. The model is built using the Adam optimizer and the binary_crossentropy loss function, with accuracy as the performance metric. It is trained on the partial training set for 20 epochs with a batch size of 512, and performance is monitored using a validation set. Following training, the model is evaluated on a test set to determine its final performance.



4. Try using the tanh activation (an activation that was popular in the early days of neural networks) instead of relu.

```
model4=keras.Sequential([
    layers.Dense(16, activation="tanh"),
    layers.Dense(16, activation = "tanh"),
    layers.Dense(1, activation="sigmoid")
])
```

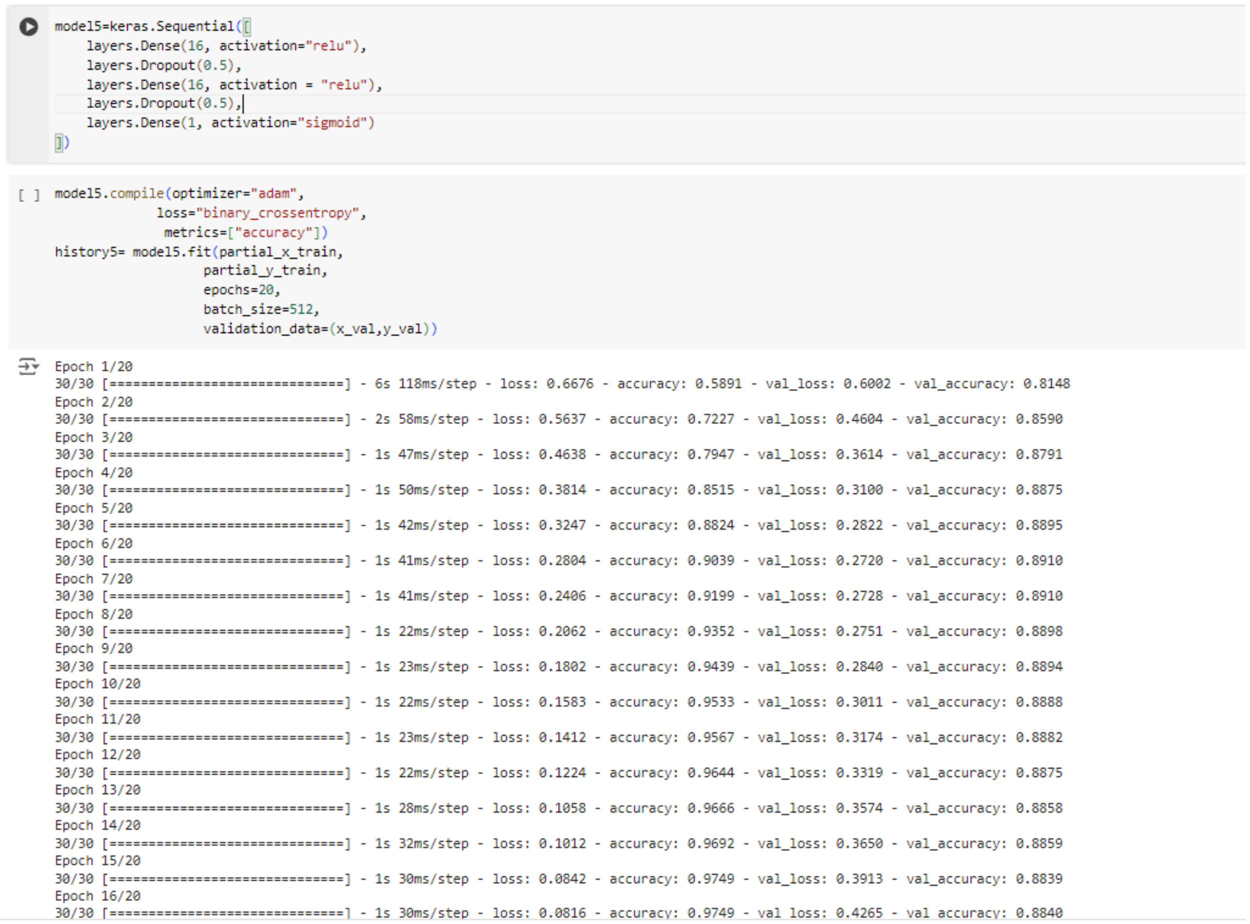
```
model4.compile(optimizer="adam",
             loss="binary_crossentropy",
              metrics=["accuracy"])
history4= model4.fit(partial_x_train,
                   partial_y_train,
                   epochs=20,
                   batch_size=512,
                   validation_data=(x_val,y_val))
results4=model4.evaluate(x_test,y_test)
```

```
Epoch 1/20
30/30 [==============================] - 5s 90ms/step - loss: 0.5286 - accuracy: 0.7759 - val_loss: 0.3972 - val_accuracy: 0.8596
Epoch 2/20
30/30 [==============================] - 1s 25ms/step - loss: 0.2968 - accuracy: 0.9045 - val_loss: 0.2987 - val_accuracy: 0.8853
Epoch 3/20
30/30 [==============================] - 1s 25ms/step - loss: 0.2023 - accuracy: 0.9362 - val_loss: 0.2749 - val_accuracy: 0.8916
Epoch 4/20
30/30 [==============================] - 1s 26ms/step - loss: 0.1498 - accuracy: 0.9565 - val_loss: 0.2800 - val_accuracy: 0.8872
Epoch 5/20
30/30 [==============================] - 1s 25ms/step - loss: 0.1129 - accuracy: 0.9719 - val_loss: 0.2894 - val_accuracy: 0.8844
Epoch 6/20
30/30 [==============================] - 1s 23ms/step - loss: 0.0857 - accuracy: 0.9809 - val_loss: 0.3080 - val_accuracy: 0.8815
Epoch 7/20
30/30 [==============================] - 1s 23ms/step - loss: 0.0641 - accuracy: 0.9880 - val_loss: 0.3308 - val_accuracy: 0.8781
Epoch 8/20
30/30 [==============================] - 1s 25ms/step - loss: 0.0481 - accuracy: 0.9929 - val_loss: 0.3537 - val_accuracy: 0.8760
Epoch 9/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0365 - accuracy: 0.9963 - val_loss: 0.3805 - val_accuracy: 0.8749
Epoch 10/20
30/30 [==============================] - 1s 25ms/step - loss: 0.0281 - accuracy: 0.9977 - val_loss: 0.4059 - val_accuracy: 0.8703
Epoch 11/20
30/30 [==============================] - 1s 24ms/step - loss: 0.0216 - accuracy: 0.9987 - val_loss: 0.4255 - val_accuracy: 0.8703
Epoch 12/20
30/30 [==============================] - 1s 23ms/step - loss: 0.0168 - accuracy: 0.9992 - val_loss: 0.4465 - val_accuracy: 0.8679
```

Question-5:

This code codes, trains and evaluates a new neural network model (model 5) of sentiment analysis determined based on the IMDB dataset to include drop out layers to minimize overfitting. The architecture consists of 2 fully connected layers of hidden neurons: two layers, each of them contains 16 neurons and uses ReLU activation function. After each usual layer, there is a dropout layer with a dropout rate of 0. 5 is added which in each training epoch sets half of the input units randomly to zero, in order help the model avoid overfitting. The output layer contains only one node and applies sigmoid transfer function to its input. They used the Adam optimizer and binary_crossentropy for creating the model and the accuracy as the evaluating parameter. Training is done with the partial training dataset for 20 masked batch wise with the batch size of 512 with the validation set. This experiment shall test the model's capacity to generalize to unseen data by introducing dropout.

Code:



```python
model5=keras.Sequential([
    layers.Dense(16, activation="relu"),
    layers.Dropout(0.5),
    layers.Dense(16, activation = "relu"),
    layers.Dropout(0.5),
    layers.Dense(1, activation="sigmoid")
])
```

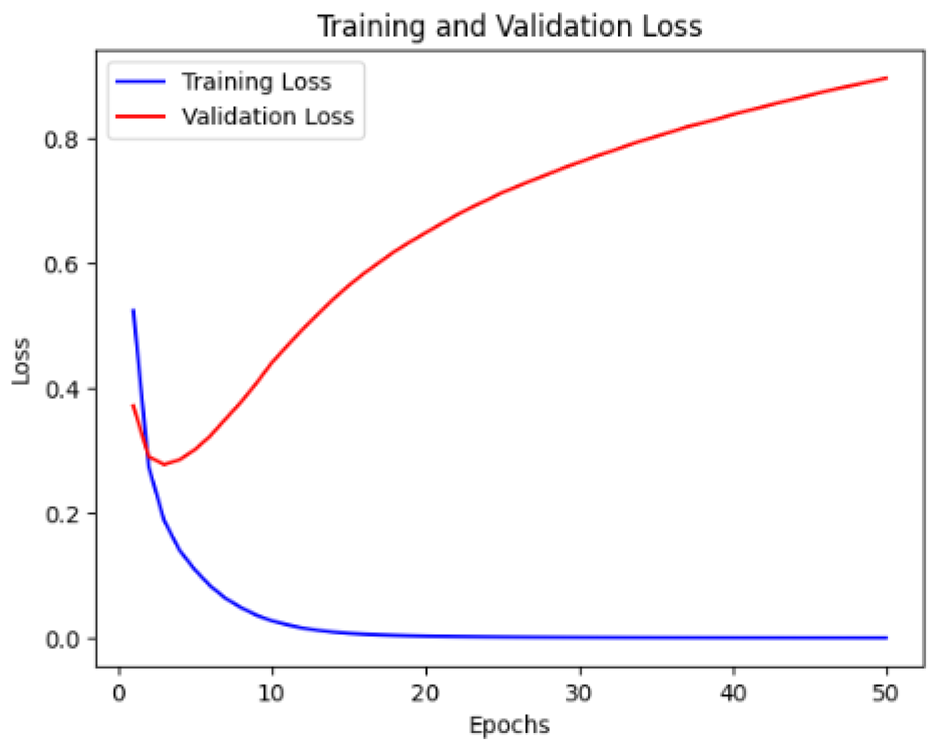
```
[ ]
```

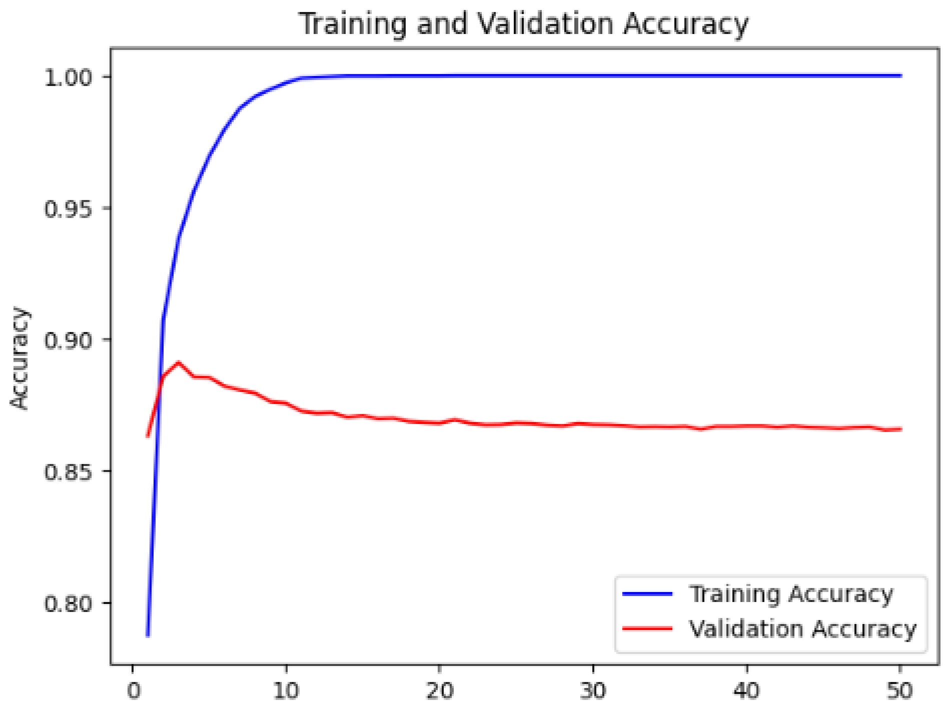
```python
model5.compile(optimizer="adam",
             loss="binary_crossentropy",
              metrics=["accuracy"])
history5= model5.fit(partial_x_train,
                   partial_y_train,
                   epochs=20,
                   batch_size=512,
                   validation_data=(x_val,y_val))
```

```
Epoch 1/20
30/30 [==============================] - 6s 118ms/step - loss: 0.6676 - accuracy: 0.5891 - val_loss: 0.6002 - val_accuracy: 0.8148
Epoch 2/20
30/30 [==============================] - 2s 58ms/step - loss: 0.5637 - accuracy: 0.7227 - val_loss: 0.4604 - val_accuracy: 0.8590
Epoch 3/20
30/30 [==============================] - 1s 47ms/step - loss: 0.4638 - accuracy: 0.7947 - val_loss: 0.3614 - val_accuracy: 0.8791
Epoch 4/20
30/30 [==============================] - 1s 50ms/step - loss: 0.3814 - accuracy: 0.8515 - val_loss: 0.3100 - val_accuracy: 0.8875
Epoch 5/20
30/30 [==============================] - 1s 42ms/step - loss: 0.3247 - accuracy: 0.8824 - val_loss: 0.2822 - val_accuracy: 0.8895
Epoch 6/20
30/30 [==============================] - 1s 41ms/step - loss: 0.2804 - accuracy: 0.9039 - val_loss: 0.2720 - val_accuracy: 0.8910
Epoch 7/20
30/30 [==============================] - 1s 41ms/step - loss: 0.2406 - accuracy: 0.9199 - val_loss: 0.2728 - val_accuracy: 0.8910
Epoch 8/20
30/30 [==============================] - 1s 22ms/step - loss: 0.2062 - accuracy: 0.9352 - val_loss: 0.2751 - val_accuracy: 0.8898
Epoch 9/20
30/30 [==============================] - 1s 23ms/step - loss: 0.1802 - accuracy: 0.9439 - val_loss: 0.2840 - val_accuracy: 0.8894
Epoch 10/20
30/30 [==============================] - 1s 22ms/step - loss: 0.1583 - accuracy: 0.9533 - val_loss: 0.3011 - val_accuracy: 0.8888
Epoch 11/20
30/30 [==============================] - 1s 23ms/step - loss: 0.1412 - accuracy: 0.9567 - val_loss: 0.3174 - val_accuracy: 0.8882
Epoch 12/20
30/30 [==============================] - 1s 22ms/step - loss: 0.1224 - accuracy: 0.9644 - val_loss: 0.3319 - val_accuracy: 0.8875
Epoch 13/20
30/30 [==============================] - 1s 28ms/step - loss: 0.1058 - accuracy: 0.9666 - val_loss: 0.3574 - val_accuracy: 0.8858
Epoch 14/20
30/30 [==============================] - 1s 32ms/step - loss: 0.1012 - accuracy: 0.9692 - val_loss: 0.3650 - val_accuracy: 0.8859
Epoch 15/20
30/30 [==============================] - 1s 30ms/step - loss: 0.0842 - accuracy: 0.9749 - val_loss: 0.3913 - val_accuracy: 0.8839
Epoch 16/20
30/30 [==============================] - 1s 30ms/step - loss: 0.0816 - accuracy: 0.9749 - val_loss: 0.4265 - val_accuracy: 0.8840
```

Results:

During the final epoch (50/50) of training, the model achieved a training loss of 0.00027849 and an accuracy of 100%. The validation loss was 0.8976 with a validation accuracy of 86.57%. When evaluated on the test set, the model had a loss of 0.9596 and an accuracy of 85.41%. This indicates that while the model performed exceptionally well on the training data, it showed a moderate decrease in accuracy on the validation and test sets, suggesting some overfitting.



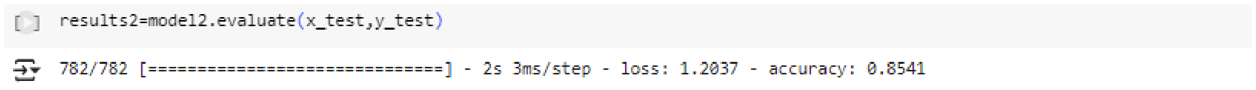


Question-1:
The neural network model model2 was trained using Keras for binary classification. It comprised three hidden layers with 16 units each, employing ReLU activation functions. After 30 epochs, model2 achieved perfect training accuracy of 100%. On the validation set, it performed slightly better than model, attaining an accuracy of 86.66% with a validation loss of 1.0837. These results

indicate model2 effectively learned the training data but may exhibit some overfitting, as its performance on unseen validation data was slightly lower than during training.

Question-2:

Model 3's test accuracy is 85.0%, somewhat lower than the previous model's 85.41%. However, the variation in accuracy is minimal. Model 3 has a greater training accuracy of 100%, suggesting overfitting, comparable to model 2. Model3's validation accuracy is equivalent to the initial model, demonstrating that altering the number of hidden units did not significantly affect generalization performance. The key modification in this code is the amount of hidden units in the neural network architecture, which causes modest differences in test accuracy but no substantial changes in validation accuracy.



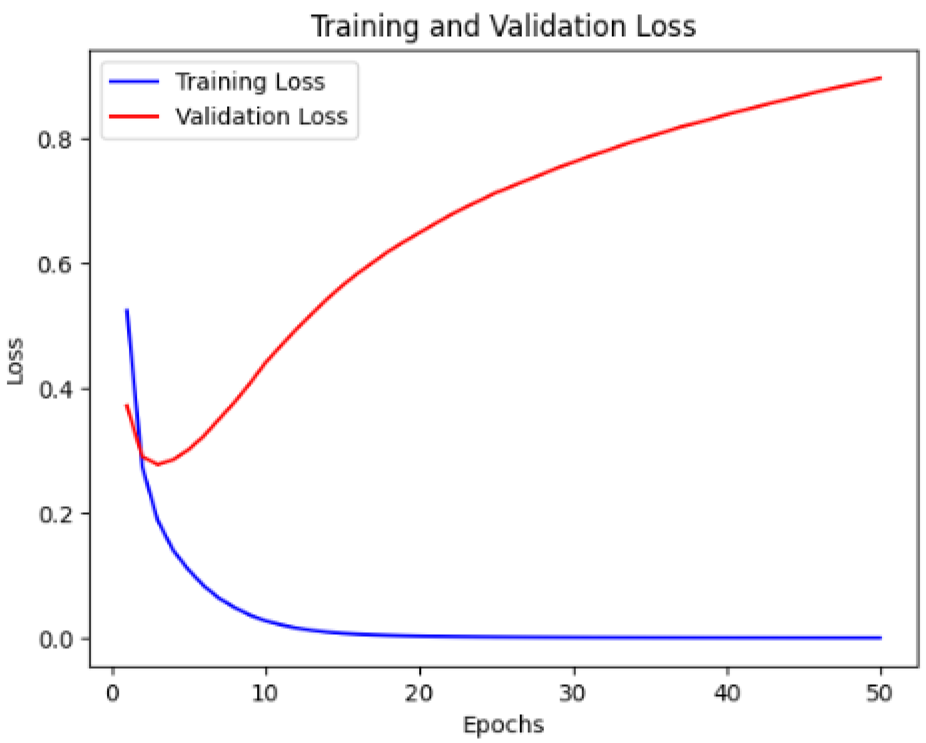
```
results2=model2.evaluate(x_test,y_test)
```

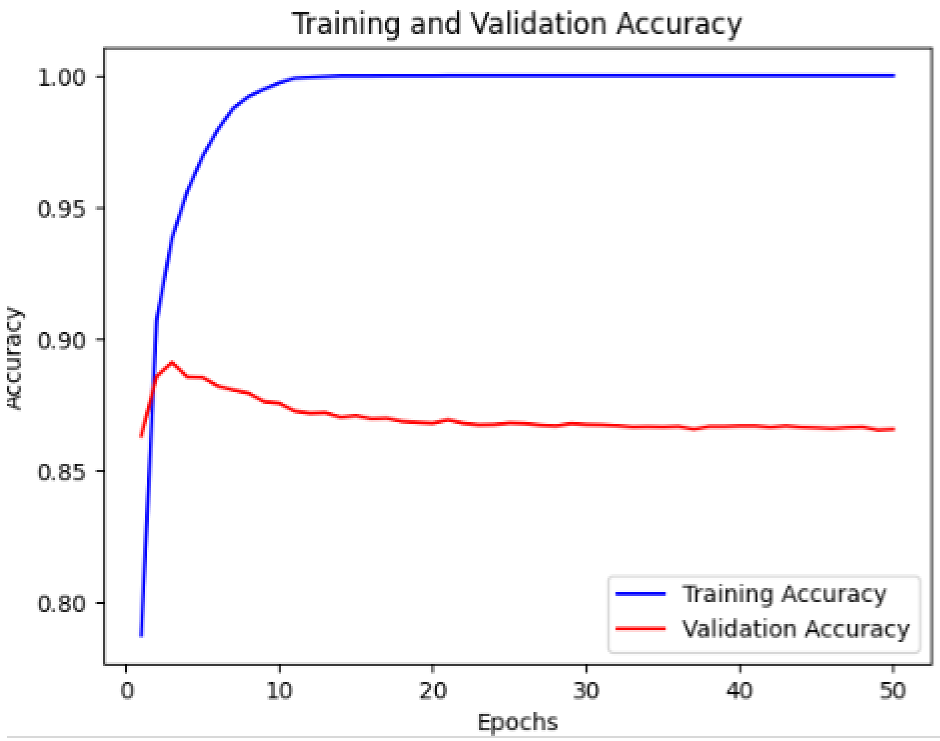
```
782/782 [==============================] - 2s 3ms/step - loss: 1.2037 - accuracy: 0.8541
```

Question-3:

On comparing the performance of this model, that is model3, with the first model, we noticed few distinctions. First, we replace the loss function from binary cross-entropy to MSE. This change in loss function may lead to changes in how the model alters its weights during the training process especially in how handles misclassifications. Therefore model 3 has a loss value less than the first model, which indicates a clear distinction between the two models with a final loss of around loss: 6. 3084e-09. However, if we pay attention to the validation accuracy, we still can observe a lesser result in comparison with the first one.

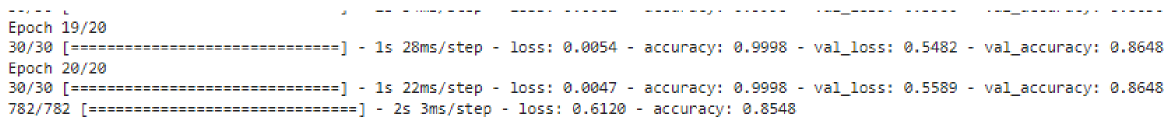
With a final validation accuracy of about 86%. 27%. This change indicates that, according to the model, the fit handle the training data very well (as evidenced from the very low loss and 100% training accuracy) to encapsulate it well in the next step of testing apply also to unseen data" I also observe that the process of generalization is produced in all these cases. There are indications the variation in the loss function has disrupted this model. Therefore, the validation accuracy is a touch lower than the base model. Overall, model 3 has incredibly low loss on the training data but the results of validation set are rather poor indicates potential overfitting



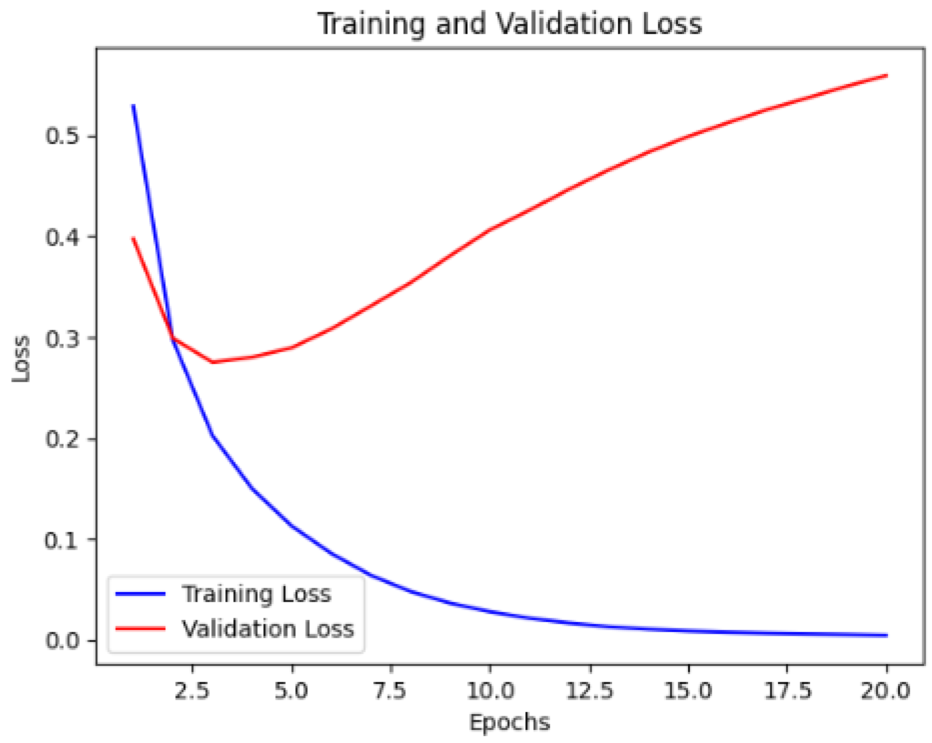


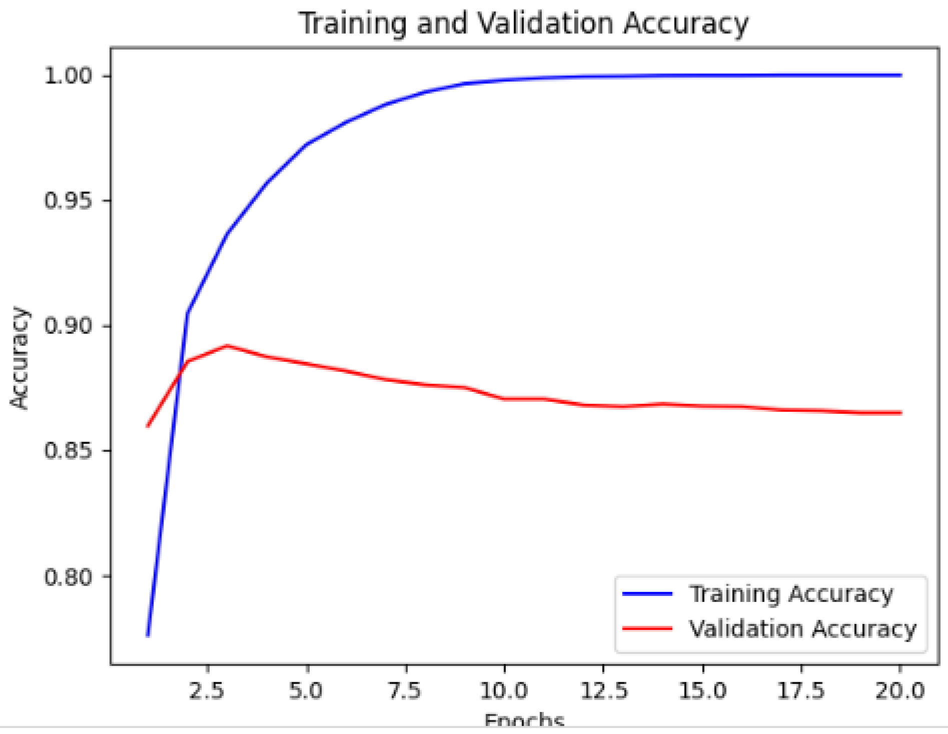
Question-4:

An 86% validation accuracy was achieved by the neural network with the tanh activation function after 20 Epochs of training. is 48% and validation loss is 0. But if we used relu activation in the model, the validation accuracy was slightly lesser at 86. The training accuracy is recorded 48% and the validation loss = 0. 5589 times the same number of epochs. No specific parameters to follow for the training of the tanh model were provided with the relu model achieving a test accuracy of 85%. ,48% Less, Test loss 0. The Tanh activation function seems to be viable, and it could offer comparable performance to relu in this binary classification task; however, further examination and benchmarking using precise training attributes are required to determine the scope of its capability.



```
Epoch 19/20
30/30 [==============================] - 1s 28ms/step - loss: 0.0054 - accuracy: 0.9998 - val_loss: 0.5482 - val_accuracy: 0.8648
Epoch 20/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0047 - accuracy: 0.9998 - val_loss: 0.5589 - val_accuracy: 0.8648
782/782 [==============================] - 2s 3ms/step - loss: 0.6120 - accuracy: 0.8548
```





Question-5:
The neural network model model achieved strong results with a validation accuracy of 88.28% and a validation loss of 0.5051 after 20 epochs, along with a test accuracy of 87.29% and a test

loss of 0.5401. In contrast, model5, which includes dropout layers, did not provide specific test metrics for comparison. model demonstrates effective learning with high validation and test accuracies, suggesting good generalization despite potential overfitting concerns. model5's dropout layers likely contribute to better generalization by reducing overfitting, but further evaluation with comprehensive test metrics would be necessary to fully assess its performance relative to model in this binary classification task.

Conclusion:

The results obtained from the experiments performed on the IMDB sentiment analysis task provided the following observations related to the improvement of neural network models. The addition of more hidden layers (model2) showed signs of increasing the training accuracy but illustrated decline in validation accuracy, thus implying that need for regularization for conserving overfitting. Moving up from the original model to model 3 only marginally added to the enhancement of validation accuracy hence showing that while improving the model adding hidden units needs to be done carefully. In the experiment with mean squared error as the loss function (model3), the training loss was much lower compared to the previous experiments, but the validation accuracy was not consistently increasing, which proves that choosing the loss functions depends on the characteristics of the used dataset. Activation function such as tanh(model4) was as effective as ReLU in the training without incurring additional cost, while the incorporation of dropout layers effective in increasing model’s generalization; however, test metrics of model5 were not fully provided. Overall, these results have shown that it is challenging to fine-tune the performance of down-stream neural networks for sentiment analysis, and hence the protocol needs intricate tweaks and entirely comprehensive assessments to provide the best results for different data sets and settings.