```
!pip install tensorflow

Requirement already satisfied: tensorflow in
/usr/local/lib/python3.10/dist-packages (2.17.0)
Requirement already satisfied: absl-py>=1.0.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=24.3.25 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (24.3.25)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1
in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.6.0)
Requirement already satisfied: google-pasta>=0.1.1 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: h5py>=3.10.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (3.11.0)
Requirement already satisfied: libclang>=13.0.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (18.1.1)
Requirement already satisfied: ml-dtypes<0.5.0,>=0.3.1 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (0.4.1)
Requirement already satisfied: opt-einsum>=2.3.2 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (3.3.0)
Requirement already satisfied: packaging in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (24.1)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!
=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (3.20.3)
Requirement already satisfied: requests<3,>=2.21.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (2.32.3)
Requirement already satisfied: setuptools in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (71.0.4)
Requirement already satisfied: six>=1.12.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (1.16.0)
Requirement already satisfied: termcolor>=1.1.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (2.4.0)
Requirement already satisfied: typing-extensions>=3.6.6 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (4.12.2)
Requirement already satisfied: wrapt>=1.11.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (1.16.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (1.64.1)
Requirement already satisfied: tensorboard<2.18,>=2.17 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (2.17.0)
Requirement already satisfied: keras>=3.2.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (3.4.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (0.37.1)
Requirement already satisfied: numpy<2.0.0,>=1.23.5 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (1.26.4)
Requirement already satisfied: wheel<1.0,>=0.23.0 in
```

```
/usr/local/lib/python3.10/dist-packages (from astunparse>=1.6.0-
>tensorflow) (0.44.0)
Requirement already satisfied: rich in /usr/local/lib/python3.10/dist-
packages (from keras>=3.2.0->tensorflow) (13.8.1)
Requirement already satisfied: namex in
/usr/local/lib/python3.10/dist-packages (from keras>=3.2.0-
>tensorflow) (0.0.8)
Requirement already satisfied: optree in
/usr/local/lib/python3.10/dist-packages (from keras>=3.2.0-
>tensorflow) (0.12.1)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0-
>tensorflow) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in
/usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0-
>tensorflow) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0-
>tensorflow) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0-
>tensorflow) (2024.8.30)
Requirement already satisfied: markdown>=2.6.8 in
/usr/local/lib/python3.10/dist-packages (from tensorboard<2.18,>=2.17-
>tensorflow) (3.7)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0
in /usr/local/lib/python3.10/dist-packages (from
tensorboard<2.18,>=2.17->tensorflow) (0.7.2)
Requirement already satisfied: werkzeug>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from tensorboard<2.18,>=2.17-
>tensorflow) (3.0.4)
Requirement already satisfied: MarkupSafe>=2.1.1 in
/usr/local/lib/python3.10/dist-packages (from werkzeug>=1.0.1-
>tensorboard<2.18,>=2.17->tensorflow) (2.1.5)
Requirement already satisfied: markdown-it-py>=2.2.0 in
/usr/local/lib/python3.10/dist-packages (from rich->keras>=3.2.0-
>tensorflow) (3.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in
/usr/local/lib/python3.10/dist-packages (from rich->keras>=3.2.0-
>tensorflow) (2.18.0)
Requirement already satisfied: mdurl~=0.1 in
/usr/local/lib/python3.10/dist-packages (from markdown-it-py>=2.2.0-
>rich->keras>=3.2.0->tensorflow) (0.1.2)
```

All the necessary modules are been imported.

```
import os
import numpy as np
import pandas as pd
```

```
import matplotlib.pyplot as pl
import tensorflow as tf

from keras import models, layers, optimizers, losses, metrics
from keras.models import Sequential
from keras.layers import Dense
from tensorflow.keras.utils import to_categorical
from keras.datasets import imdb


from keras.datasets import imdb
```

Downloading the IMDB Dataset

```
(training_data, training_labels), (testing_data, testing_labels) =
imdb.load_data(num_words=10000)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-
keras-datasets/imdb.npz
17464789/17464789 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
```

Vectorization of the Data

```
def vectorizingSeq(seq, dimension=10000):
    res = np.zeros((len(seq), dimension))
    for i, seqS in enumerate(seq):
        res[i, seqS] = 1.0
    return res
```

Model Building with 2 hidden layers, 16 hidden units, and RELU activation

```
Xtrain = vectorizingSeq(training_data)
Xtest = vectorizingSeq(testing_data)

Ytrain = np.asarray(training_labels).astype('float32')
Ytest = np.asarray(testing_labels).astype('float32')
print("Ytrain ", Ytrain.shape)
print("Ytest ", Ytest.shape)

Xval = Xtrain[:10000]
partial_Xtrain = Xtrain[10000:]
Yval = Ytrain[:10000]
partial_Ytrain = Ytrain[10000:]
```

```
Ytrain  (25000,)
Ytest  (25000,)
```

```
Xval = Xtrain[:10000]
partial_Xtrain = Xtrain[10000:]
```

```
Yval = Ytrain[:10000]
partial_Ytrain = Ytrain[10000:]
```

Model Building with two hidden layers, 16 hidden units, and the relu activation

```
mod_layer = models.Sequential()
mod_layer.add(layers.Dense(16, activation='relu',
input_shape=(10000,)))
mod_layer.add(layers.Dense(16, activation='relu'))
mod_layer.add(layers.Dense(1, activation='sigmoid'))
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/
dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim`
argument to a layer. When using Sequential models, prefer using an
`Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)
```

Model compilation using the binary cross entropy loss function and the rnsprob optimizer

```
mod_layer.compile(optimizer='rmsprop', loss='binary_crossentropy',
metrics=['accuracy'])

hist_mod_layer = mod_layer.fit(partial_Xtrain, partial_Ytrain,
epochs=20, batch_size=512, validation_data=(Xval, Yval))

res_mod_layer = mod_layer.evaluate(Xtest, Ytest)
print("_" * 100)
print("Test Loss and Accuracy")
print("res_mod_layer ", res_mod_layer)
hist_data = hist_mod_layer.history
hist_data.keys()
```

```
Epoch 1/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 7s 171ms/step - accuracy: 0.5450 - loss:
0.6658 - val_accuracy: 0.7761 - val_loss: 0.5366
Epoch 2/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 6s 36ms/step - accuracy: 0.8357 - loss:
0.4952 - val_accuracy: 0.8310 - val_loss: 0.4355
Epoch 3/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - accuracy: 0.9001 - loss:
0.3684 - val_accuracy: 0.8870 - val_loss: 0.3422
Epoch 4/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.9257 - loss:
0.2694 - val_accuracy: 0.8840 - val_loss: 0.3082
Epoch 5/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 92ms/step - accuracy: 0.9374 - loss:
0.2131 - val_accuracy: 0.8882 - val_loss: 0.2836
Epoch 6/20
```

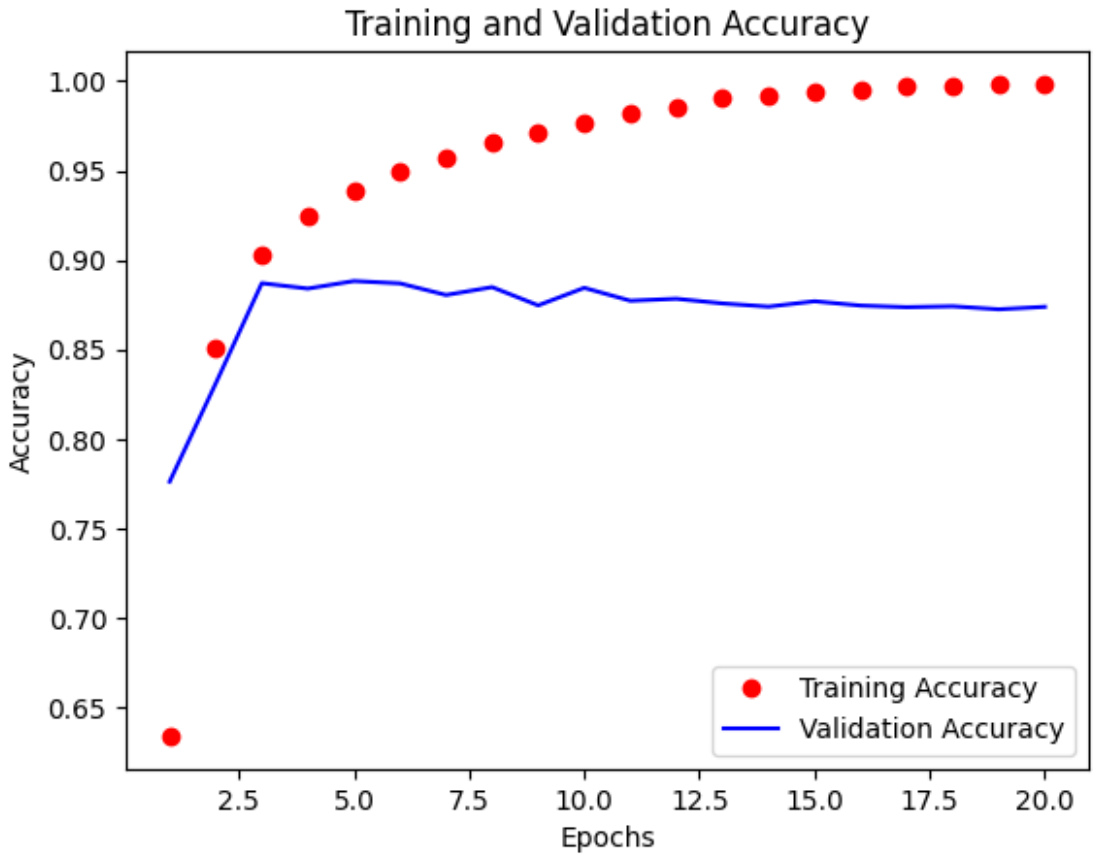
```
30/30 ━━━━━━━━━━━━━━━━━━━━ 4s 60ms/step - accuracy: 0.9531 - loss:
0.1737 - val_accuracy: 0.8869 - val_loss: 0.2807
Epoch 7/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 52ms/step - accuracy: 0.9606 - loss:
0.1407 - val_accuracy: 0.8804 - val_loss: 0.2966
Epoch 8/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 51ms/step - accuracy: 0.9687 - loss:
0.1171 - val_accuracy: 0.8848 - val_loss: 0.3035
Epoch 9/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - accuracy: 0.9741 - loss:
0.1026 - val_accuracy: 0.8746 - val_loss: 0.3500
Epoch 10/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 65ms/step - accuracy: 0.9768 - loss:
0.0918 - val_accuracy: 0.8844 - val_loss: 0.3207
Epoch 11/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 4s 108ms/step - accuracy: 0.9844 - loss:
0.0736 - val_accuracy: 0.8772 - val_loss: 0.3383
Epoch 12/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - accuracy: 0.9870 - loss:
0.0633 - val_accuracy: 0.8783 - val_loss: 0.3521
Epoch 13/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - accuracy: 0.9915 - loss:
0.0529 - val_accuracy: 0.8757 - val_loss: 0.3737
Epoch 14/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - accuracy: 0.9929 - loss:
0.0443 - val_accuracy: 0.8739 - val_loss: 0.3982
Epoch 15/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 53ms/step - accuracy: 0.9949 - loss:
0.0375 - val_accuracy: 0.8769 - val_loss: 0.4162
Epoch 16/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 56ms/step - accuracy: 0.9963 - loss:
0.0282 - val_accuracy: 0.8745 - val_loss: 0.4496
Epoch 17/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 36ms/step - accuracy: 0.9972 - loss:
0.0243 - val_accuracy: 0.8736 - val_loss: 0.4557
Epoch 18/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 48ms/step - accuracy: 0.9981 - loss:
0.0200 - val_accuracy: 0.8741 - val_loss: 0.4735
Epoch 19/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 42ms/step - accuracy: 0.9983 - loss:
0.0167 - val_accuracy: 0.8724 - val_loss: 0.4949
Epoch 20/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 57ms/step - accuracy: 0.9985 - loss:
0.0135 - val_accuracy: 0.8738 - val_loss: 0.5150
782/782 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - accuracy: 0.8568 - loss:
0.5650
_________________________________________________________________
_________________________________
```

```
Test Loss and Accuracy
res_mod_layer  [0.5546508431434631, 0.8611599802970886]

dict_keys(['accuracy', 'loss', 'val_accuracy', 'val_loss'])
```

Plot of the Training and Vaildation Accuracy

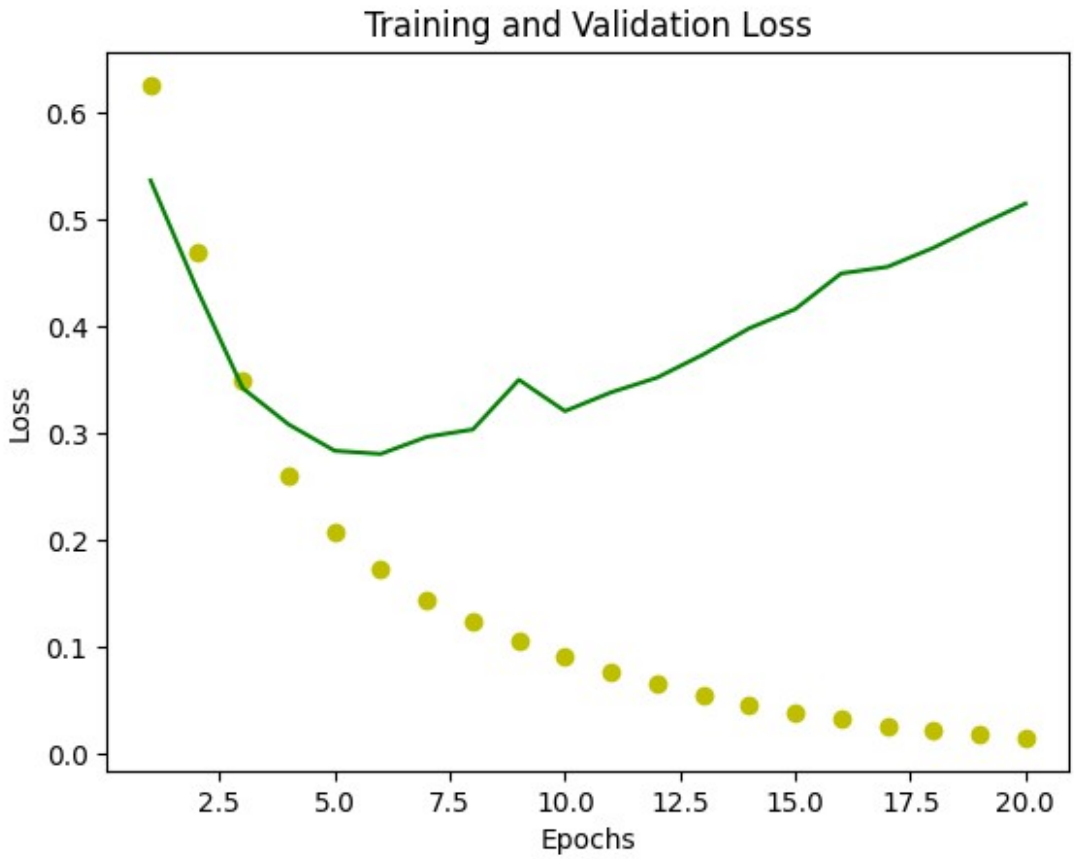
```
pl.clf()
accuracy_values = hist_data['accuracy']
val_accuracy_values = hist_data['val_accuracy']
epochs_range = range(1, (len(hist_data['accuracy']) + 1))
pl.plot(epochs_range, accuracy_values, 'ro', label='Training
Accuracy')
pl.plot(epochs_range, val_accuracy_values, 'b', label='Validation
Accuracy')
pl.title('Training and Validation Accuracy')
pl.xlabel('Epochs')
pl.ylabel('Accuracy')
pl.legend()
pl.show()
```




Plot of the Training and Validation Loss

```
pl.clf()
hist_data = hist_mod_layer.history
valueatLoss = hist_data['loss']
val_valueatLoss  = hist_data['val_loss']
epochs_range = range(1, (len(hist_data['loss']) + 1))
pl.plot(epochs_range, valueatLoss , 'yo', label='Training Loss')
pl.plot(epochs_range, val_valueatLoss, 'g', label='Validation Loss')
pl.title('Training and Validation Loss')
pl.xlabel('Epochs')
pl.ylabel('Loss')

Text(0, 0.5, 'Loss')
```




1.Model Building with one hidden layer with 16 hidden units and relu activation

```
mod_layer_1 = models.Sequential()
mod_layer_1.add(layers.Dense(16, activation='relu',
input_shape=(10000,)))
mod_layer_1.add(layers.Dense(1, activation='sigmoid'))
mod_layer_1.compile(optimizer='rmsprop', loss='binary_crossentropy',
metrics=['acc'])

history_1 = mod_layer_1.fit(partial_Xtrain, partial_Ytrain, epochs=20,
```

```
batch_size=512, validation_data=(Xval, Yval))

results_1 = mod_layer_1.evaluate(Xtest, Ytest)
print("_" * 100)
print("Test Loss and Accuracy")
print("results_1 ", results_1)
hist_dict_1 = history_1.history
hist_keys_1 = hist_dict_1.keys()

Epoch 1/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 7s 118ms/step - acc: 0.7036 - loss: 0.5789
- val_acc: 0.8732 - val_loss: 0.3826
Epoch 2/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 33ms/step - acc: 0.8996 - loss: 0.3309 -
val_acc: 0.8840 - val_loss: 0.3175
Epoch 3/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - acc: 0.9228 - loss: 0.2530 -
val_acc: 0.8854 - val_loss: 0.2971
Epoch 4/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - acc: 0.9327 - loss: 0.2165 -
val_acc: 0.8900 - val_loss: 0.2790
Epoch 5/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - acc: 0.9433 - loss: 0.1831 -
val_acc: 0.8886 - val_loss: 0.2755
Epoch 6/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 41ms/step - acc: 0.9482 - loss: 0.1644 -
val_acc: 0.8789 - val_loss: 0.2985
Epoch 7/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 63ms/step - acc: 0.9569 - loss: 0.1488 -
val_acc: 0.8849 - val_loss: 0.2882
Epoch 8/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 46ms/step - acc: 0.9619 - loss: 0.1307 -
val_acc: 0.8871 - val_loss: 0.2816
Epoch 9/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - acc: 0.9664 - loss: 0.1225 -
val_acc: 0.8769 - val_loss: 0.3050
Epoch 10/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 92ms/step - acc: 0.9668 - loss: 0.1162 -
val_acc: 0.8839 - val_loss: 0.3051
Epoch 11/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 76ms/step - acc: 0.9727 - loss: 0.1018 -
val_acc: 0.8850 - val_loss: 0.3023
Epoch 12/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 39ms/step - acc: 0.9772 - loss: 0.0926 -
val_acc: 0.8791 - val_loss: 0.3103
Epoch 13/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - acc: 0.9815 - loss: 0.0845 -
val_acc: 0.8825 - val_loss: 0.3161
Epoch 14/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 47ms/step - acc: 0.9817 - loss: 0.0778 -
```

```
val_acc: 0.8808 - val_loss: 0.3260
Epoch 15/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 53ms/step - acc: 0.9859 - loss: 0.0706 -
val_acc: 0.8805 - val_loss: 0.3355
Epoch 16/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 61ms/step - acc: 0.9875 - loss: 0.0674 -
val_acc: 0.8750 - val_loss: 0.3508
Epoch 17/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - acc: 0.9884 - loss: 0.0608 -
val_acc: 0.8773 - val_loss: 0.3536
Epoch 18/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - acc: 0.9898 - loss: 0.0570 -
val_acc: 0.8740 - val_loss: 0.3699
Epoch 19/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - acc: 0.9909 - loss: 0.0522 -
val_acc: 0.8751 - val_loss: 0.3758
Epoch 20/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - acc: 0.9913 - loss: 0.0495 -
val_acc: 0.8737 - val_loss: 0.3859
782/782 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - acc: 0.8592 - loss: 0.4196

──────────────────────────────────────────────────────────────────────
──────────────────
Test Loss and Accuracy
results_1  [0.41361203789711, 0.8636000156402588]
```

Model Building with three hidden layer with 16 hidden units and relu activation

```
mod_layer_2 = models.Sequential()
mod_layer_2.add(layers.Dense(16, activation='relu',
input_shape=(10000,)))
mod_layer_2.add(layers.Dense(16, activation='relu'))
mod_layer_2.add(layers.Dense(16, activation='relu'))
mod_layer_2.add(layers.Dense(1, activation='sigmoid'))
mod_layer_2.compile(optimizer='rmsprop', loss='binary_crossentropy',
metrics=['acc'])

history_2 = mod_layer_2.fit(partial_Xtrain, partial_Ytrain, epochs=20,
batch_size=512, validation_data=(Xval, Yval))

results_2 = mod_layer_2.evaluate(Xtest, Ytest)
print("_" * 100)
print("Test Loss and Accuracy")
print("results_2 ", results_2)
hist_dict_2 = history_2.history
hist_keys_2 = hist_dict_2.keys()

/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/
dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim`
argument to a layer. When using Sequential models, prefer using an
```

```
`Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)

Epoch 1/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 80ms/step - acc: 0.6167 - loss: 0.6433 -
val_acc: 0.8624 - val_loss: 0.4589
Epoch 2/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 39ms/step - acc: 0.8794 - loss: 0.3973 -
val_acc: 0.8838 - val_loss: 0.3256
Epoch 3/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - acc: 0.9160 - loss: 0.2638 -
val_acc: 0.8764 - val_loss: 0.3101
Epoch 4/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 40ms/step - acc: 0.9333 - loss: 0.2060 -
val_acc: 0.8850 - val_loss: 0.2881
Epoch 5/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - acc: 0.9479 - loss: 0.1591 -
val_acc: 0.8809 - val_loss: 0.3003
Epoch 6/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - acc: 0.9632 - loss: 0.1291 -
val_acc: 0.8773 - val_loss: 0.3152
Epoch 7/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 47ms/step - acc: 0.9662 - loss: 0.1137 -
val_acc: 0.8831 - val_loss: 0.3143
Epoch 8/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 39ms/step - acc: 0.9745 - loss: 0.0890 -
val_acc: 0.8719 - val_loss: 0.3588
Epoch 9/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - acc: 0.9810 - loss: 0.0761 -
val_acc: 0.8787 - val_loss: 0.3585
Epoch 10/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - acc: 0.9830 - loss: 0.0629 -
val_acc: 0.8766 - val_loss: 0.3806
Epoch 11/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - acc: 0.9847 - loss: 0.0565 -
val_acc: 0.8664 - val_loss: 0.4454
Epoch 12/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - acc: 0.9913 - loss: 0.0442 -
val_acc: 0.8724 - val_loss: 0.4360
Epoch 13/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - acc: 0.9924 - loss: 0.0350 -
val_acc: 0.8716 - val_loss: 0.4636
Epoch 14/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - acc: 0.9960 - loss: 0.0249 -
val_acc: 0.8615 - val_loss: 0.5537
Epoch 15/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - acc: 0.9957 - loss: 0.0237 -
val_acc: 0.8719 - val_loss: 0.5100
Epoch 16/20
```

```
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 40ms/step - acc: 0.9988 - loss: 0.0148 -
val_acc: 0.8679 - val_loss: 0.5653
Epoch 17/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 50ms/step - acc: 0.9848 - loss: 0.0422 -
val_acc: 0.8658 - val_loss: 0.5866
Epoch 18/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 33ms/step - acc: 0.9977 - loss: 0.0142 -
val_acc: 0.8694 - val_loss: 0.6034
Epoch 19/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - acc: 0.9989 - loss: 0.0087 -
val_acc: 0.8691 - val_loss: 0.6174
Epoch 20/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 54ms/step - acc: 0.9995 - loss: 0.0061 -
val_acc: 0.8508 - val_loss: 0.7753
782/782 ━━━━━━━━━━━━━━━━━━━━ 3s 4ms/step - acc: 0.8377 - loss: 0.8278
____________________________________________________________________________
_____________________________
Test Loss and Accuracy
results_2  [0.8084371089935303, 0.8406800031661987]
```

2.Now with 32 hidden units and one hidden layer.

```
mod_layer_3 = models.Sequential()
mod_layer_3.add(layers.Dense(32, activation='relu',
input_shape=(10000,)))
mod_layer_3.add(layers.Dense(1, activation='sigmoid'))
mod_layer_3.compile(optimizer='rmsprop', loss='binary_crossentropy',
metrics=['acc'])

hist_3 = mod_layer_3.fit(partial_Xtrain, partial_Ytrain, epochs=20,
batch_size=512, validation_data=(Xval, Yval))

res_3 = mod_layer_3.evaluate(Xtest, Ytest)
print("_" * 100)
print("Test Loss and Accuracy")
print("res_3 ", res_3)
hist_dict_3 = hist_3.history
hist_keys_3 = hist_dict_3.keys()

Epoch 1/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 4s 95ms/step - acc: 0.6913 - loss: 0.5788 -
val_acc: 0.8664 - val_loss: 0.3805
Epoch 2/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 41ms/step - acc: 0.8964 - loss: 0.3316 -
val_acc: 0.8729 - val_loss: 0.3241
Epoch 3/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 43ms/step - acc: 0.9196 - loss: 0.2529 -
val_acc: 0.8849 - val_loss: 0.2889
Epoch 4/20
```

```
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 43ms/step - acc: 0.9322 - loss: 0.2100 -
val_acc: 0.8879 - val_loss: 0.2784
Epoch 5/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 44ms/step - acc: 0.9438 - loss: 0.1804 -
val_acc: 0.8860 - val_loss: 0.2806
Epoch 6/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 74ms/step - acc: 0.9487 - loss: 0.1613 -
val_acc: 0.8822 - val_loss: 0.2885
Epoch 7/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 63ms/step - acc: 0.9575 - loss: 0.1434 -
val_acc: 0.8837 - val_loss: 0.2865
Epoch 8/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 42ms/step - acc: 0.9641 - loss: 0.1280 -
val_acc: 0.8844 - val_loss: 0.2857
Epoch 9/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 45ms/step - acc: 0.9671 - loss: 0.1200 -
val_acc: 0.8839 - val_loss: 0.2920
Epoch 10/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 42ms/step - acc: 0.9684 - loss: 0.1091 -
val_acc: 0.8766 - val_loss: 0.3324
Epoch 11/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 43ms/step - acc: 0.9698 - loss: 0.1028 -
val_acc: 0.8829 - val_loss: 0.3061
Epoch 12/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 69ms/step - acc: 0.9769 - loss: 0.0918 -
val_acc: 0.8828 - val_loss: 0.3206
Epoch 13/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 58ms/step - acc: 0.9790 - loss: 0.0861 -
val_acc: 0.8768 - val_loss: 0.3256
Epoch 14/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 42ms/step - acc: 0.9817 - loss: 0.0775 -
val_acc: 0.8685 - val_loss: 0.3704
Epoch 15/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 53ms/step - acc: 0.9830 - loss: 0.0738 -
val_acc: 0.8787 - val_loss: 0.3410
Epoch 16/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 44ms/step - acc: 0.9853 - loss: 0.0657 -
val_acc: 0.8778 - val_loss: 0.3509
Epoch 17/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 44ms/step - acc: 0.9877 - loss: 0.0592 -
val_acc: 0.8769 - val_loss: 0.3627
Epoch 18/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 41ms/step - acc: 0.9887 - loss: 0.0558 -
val_acc: 0.8721 - val_loss: 0.3761
Epoch 19/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 57ms/step - acc: 0.9903 - loss: 0.0525 -
val_acc: 0.8735 - val_loss: 0.3819
Epoch 20/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 59ms/step - acc: 0.9913 - loss: 0.0485 -
```

```
val_acc: 0.8749 - val_loss: 0.4022
782/782 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - acc: 0.8623 - loss: 0.4341
_____
_____
Test Loss and Accuracy
res_3  [0.4239153265953064, 0.8664799928665161]
```

Now with 64 units and one hidden layer

```
mod_layer_4 = models.Sequential()
mod_layer_4.add(layers.Dense(64, activation='relu',
input_shape=(10000,)))
mod_layer_4.add(layers.Dense(1, activation='sigmoid'))
mod_layer_4.compile(optimizer='rmsprop', loss='binary_crossentropy',
metrics=['acc'])

hist_4 = mod_layer_4.fit(partial_Xtrain, partial_Ytrain, epochs=20,
batch_size=512, validation_data=(Xval, Yval))

results_4 = mod_layer_4.evaluate(Xtest, Ytest)
print("_" * 100)
print("Test Loss and Accuracy")
print("results_4 ", results_4)
hist_dict_4 = hist_4.history
hist_keys_4 = hist_dict_4.keys()

Epoch 1/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 5s 135ms/step - acc: 0.6763 - loss: 0.5714
- val_acc: 0.8556 - val_loss: 0.3800
Epoch 2/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 63ms/step - acc: 0.8919 - loss: 0.3141 -
val_acc: 0.8830 - val_loss: 0.3021
Epoch 3/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 63ms/step - acc: 0.9188 - loss: 0.2427 -
val_acc: 0.8824 - val_loss: 0.2936
Epoch 4/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 62ms/step - acc: 0.9280 - loss: 0.2052 -
val_acc: 0.8888 - val_loss: 0.2772
Epoch 5/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 66ms/step - acc: 0.9461 - loss: 0.1714 -
val_acc: 0.8868 - val_loss: 0.2835
Epoch 6/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 85ms/step - acc: 0.9510 - loss: 0.1564 -
val_acc: 0.8766 - val_loss: 0.3034
Epoch 7/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 99ms/step - acc: 0.9569 - loss: 0.1392 -
val_acc: 0.8701 - val_loss: 0.3404
Epoch 8/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 4s 61ms/step - acc: 0.9573 - loss: 0.1313 -
```

```
val_acc: 0.8826 - val_loss: 0.2956
Epoch 9/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 65ms/step - acc: 0.9677 - loss: 0.1098 -
val_acc: 0.8836 - val_loss: 0.3000
Epoch 10/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 60ms/step - acc: 0.9699 - loss: 0.1025 -
val_acc: 0.8828 - val_loss: 0.3144
Epoch 11/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 92ms/step - acc: 0.9739 - loss: 0.0940 -
val_acc: 0.8737 - val_loss: 0.3311
Epoch 12/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 4s 60ms/step - acc: 0.9782 - loss: 0.0845 -
val_acc: 0.8796 - val_loss: 0.3273
Epoch 13/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 60ms/step - acc: 0.9769 - loss: 0.0810 -
val_acc: 0.8794 - val_loss: 0.3388
Epoch 14/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 59ms/step - acc: 0.9855 - loss: 0.0652 -
val_acc: 0.8656 - val_loss: 0.3870
Epoch 15/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 62ms/step - acc: 0.9795 - loss: 0.0721 -
val_acc: 0.8785 - val_loss: 0.3663
Epoch 16/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 78ms/step - acc: 0.9849 - loss: 0.0625 -
val_acc: 0.8787 - val_loss: 0.3665
Epoch 17/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 100ms/step - acc: 0.9892 - loss: 0.0535
- val_acc: 0.8747 - val_loss: 0.3791
Epoch 18/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 5s 80ms/step - acc: 0.9912 - loss: 0.0464 -
val_acc: 0.8753 - val_loss: 0.3943
Epoch 19/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 64ms/step - acc: 0.9919 - loss: 0.0451 -
val_acc: 0.8737 - val_loss: 0.4020
Epoch 20/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 69ms/step - acc: 0.9949 - loss: 0.0378 -
val_acc: 0.8714 - val_loss: 0.4175
782/782 ━━━━━━━━━━━━━━━━━━━━ 3s 4ms/step - acc: 0.8582 - loss: 0.4539
```

---

```
Test Loss and Accuracy
results_4  [0.4481642246246338, 0.8614400029182434]
```

3.Using the MSE Loss function

```
mod_layer_5 = models.Sequential()
mod_layer_5.add(layers.Dense(16, activation='relu',
input_shape=(10000,)))
mod_layer_5.add(layers.Dense(1, activation='sigmoid'))
```

```
mod_layer_5.compile(optimizer='rmsprop', loss='mse', metrics=['acc'])

hist_5 = mod_layer_5.fit(partial_Xtrain, partial_Ytrain, epochs=20,
batch_size=512, validation_data=(Xval, Yval))

res_5_units = mod_layer_5.evaluate(Xtest, Ytest)
print("_" * 100)
print("Test Loss and Accuracy")
print("res_5_units ", res_5_units)
hist_dict_5 = hist_5.history
hist_dict_5.keys()
```

```
Epoch 1/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 71ms/step - acc: 0.6994 - loss: 0.2038 -
val_acc: 0.8637 - val_loss: 0.1261
Epoch 2/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 57ms/step - acc: 0.8820 - loss: 0.1117 -
val_acc: 0.8771 - val_loss: 0.1047
Epoch 3/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 79ms/step - acc: 0.9048 - loss: 0.0869 -
val_acc: 0.8873 - val_loss: 0.0931
Epoch 4/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - acc: 0.9258 - loss: 0.0712 -
val_acc: 0.8896 - val_loss: 0.0885
Epoch 5/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - acc: 0.9359 - loss: 0.0617 -
val_acc: 0.8892 - val_loss: 0.0858
Epoch 6/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - acc: 0.9436 - loss: 0.0556 -
val_acc: 0.8883 - val_loss: 0.0845
Epoch 7/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - acc: 0.9513 - loss: 0.0498 -
val_acc: 0.8876 - val_loss: 0.0850
Epoch 8/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - acc: 0.9568 - loss: 0.0442 -
val_acc: 0.8842 - val_loss: 0.0837
Epoch 9/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - acc: 0.9610 - loss: 0.0409 -
val_acc: 0.8811 - val_loss: 0.0870
Epoch 10/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - acc: 0.9632 - loss: 0.0405 -
val_acc: 0.8845 - val_loss: 0.0840
Epoch 11/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 58ms/step - acc: 0.9663 - loss: 0.0379 -
val_acc: 0.8733 - val_loss: 0.0923
Epoch 12/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 83ms/step - acc: 0.9686 - loss: 0.0349 -
val_acc: 0.8827 - val_loss: 0.0868
Epoch 13/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 4s 62ms/step - acc: 0.9742 - loss: 0.0322 -
```

```
val_acc: 0.8770 - val_loss: 0.0888
Epoch 14/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 4s 97ms/step - acc: 0.9766 - loss: 0.0295 -
val_acc: 0.8818 - val_loss: 0.0861
Epoch 15/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 4s 58ms/step - acc: 0.9755 - loss: 0.0289 -
val_acc: 0.8747 - val_loss: 0.0934
Epoch 16/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 51ms/step - acc: 0.9760 - loss: 0.0284 -
val_acc: 0.8798 - val_loss: 0.0873
Epoch 17/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - acc: 0.9818 - loss: 0.0257 -
val_acc: 0.8780 - val_loss: 0.0879
Epoch 18/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - acc: 0.9818 - loss: 0.0248 -
val_acc: 0.8789 - val_loss: 0.0894
Epoch 19/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - acc: 0.9842 - loss: 0.0228 -
val_acc: 0.8762 - val_loss: 0.0919
Epoch 20/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - acc: 0.9820 - loss: 0.0243 -
val_acc: 0.8751 - val_loss: 0.0902
782/782 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - acc: 0.8650 - loss: 0.0987
____________________________________________________________________
_____________________________
Test Loss and Accuracy
res_5_units  [0.09694524109363556, 0.868399977684021]

dict_keys(['acc', 'loss', 'val_acc', 'val_loss'])
```

4.Using the tanh activation

```
mod_layer_6 = models.Sequential()
mod_layer_6.add(layers.Dense(16, activation='tanh',
input_shape=(10000,)))
mod_layer_6.add(layers.Dense(1, activation='sigmoid'))
mod_layer_6.compile(optimizer='rmsprop', loss='binary_crossentropy',
metrics=['acc'])

hist_6 = mod_layer_6.fit(partial_Xtrain, partial_Ytrain, epochs=20,
batch_size=512, validation_data=(Xval, Yval))

res_6_units = mod_layer_6.evaluate(Xtest, Ytest)
print("_" * 100)
print("Test Loss and Accuracy")
print("res_6_units ", res_6_units)
hist_dict_6 = hist_6.history
hist_dict_6.keys()
```

```
Epoch 1/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 5s 146ms/step - acc: 0.6962 - loss: 0.5867
- val_acc: 0.8566 - val_loss: 0.4162
Epoch 2/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 39ms/step - acc: 0.8887 - loss: 0.3637 -
val_acc: 0.8736 - val_loss: 0.3400
Epoch 3/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - acc: 0.9162 - loss: 0.2780 -
val_acc: 0.8864 - val_loss: 0.3009
Epoch 4/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 45ms/step - acc: 0.9326 - loss: 0.2280 -
val_acc: 0.8839 - val_loss: 0.2928
Epoch 5/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 88ms/step - acc: 0.9393 - loss: 0.2000 -
val_acc: 0.8889 - val_loss: 0.2746
Epoch 6/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 4s 117ms/step - acc: 0.9489 - loss: 0.1734
- val_acc: 0.8892 - val_loss: 0.2731
Epoch 7/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 97ms/step - acc: 0.9539 - loss: 0.1548 -
val_acc: 0.8876 - val_loss: 0.2726
Epoch 8/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 53ms/step - acc: 0.9584 - loss: 0.1423 -
val_acc: 0.8865 - val_loss: 0.2768
Epoch 9/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 43ms/step - acc: 0.9661 - loss: 0.1236 -
val_acc: 0.8872 - val_loss: 0.2812
Epoch 10/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - acc: 0.9691 - loss: 0.1131 -
val_acc: 0.8836 - val_loss: 0.2926
Epoch 11/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - acc: 0.9712 - loss: 0.1025 -
val_acc: 0.8806 - val_loss: 0.3139
Epoch 12/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 53ms/step - acc: 0.9776 - loss: 0.0901 -
val_acc: 0.8806 - val_loss: 0.3082
Epoch 13/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 48ms/step - acc: 0.9816 - loss: 0.0808 -
val_acc: 0.8785 - val_loss: 0.3221
Epoch 14/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 45ms/step - acc: 0.9844 - loss: 0.0729 -
val_acc: 0.8786 - val_loss: 0.3302
Epoch 15/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 64ms/step - acc: 0.9869 - loss: 0.0663 -
val_acc: 0.8775 - val_loss: 0.3413
Epoch 16/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 53ms/step - acc: 0.9879 - loss: 0.0608 -
val_acc: 0.8769 - val_loss: 0.3550
Epoch 17/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 46ms/step - acc: 0.9895 - loss: 0.0539 -
```

```
val_acc: 0.8753 - val_loss: 0.3841
Epoch 18/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 45ms/step - acc: 0.9893 - loss: 0.0500 -
val_acc: 0.8752 - val_loss: 0.3961
Epoch 19/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 43ms/step - acc: 0.9921 - loss: 0.0444 -
val_acc: 0.8752 - val_loss: 0.3962
Epoch 20/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 52ms/step - acc: 0.9939 - loss: 0.0399 -
val_acc: 0.8730 - val_loss: 0.4097
782/782 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - acc: 0.8597 - loss: 0.4501
```

```
Test Loss and Accuracy
res_6_units  [0.4406522214412689, 0.8638399839401245]

dict_keys(['acc', 'loss', 'val_acc', 'val_loss'])
```

5.Using drop out for three hidden layers with 64 units, MSE loss function and RELU

```
mod_layer_7 = models.Sequential()
mod_layer_7.add(layers.Dense(16, activation='relu',
input_shape=(10000,)))
mod_layer_7.add(layers.Dropout(0.5))
mod_layer_7.add(layers.Dense(16, activation='relu'))
mod_layer_7.add(layers.Dropout(0.5))
mod_layer_7.add(layers.Dense(16, activation='relu'))
mod_layer_7.add(layers.Dropout(0.5))
mod_layer_7.add(layers.Dense(1, activation='sigmoid'))
mod_layer_7.compile(optimizer='rmsprop', loss='mse', metrics=['acc'])

hist_7 = mod_layer_7.fit(partial_Xtrain, partial_Ytrain, epochs=20,
batch_size=512, validation_data=(Xval, Yval))

res_7_units = mod_layer_7.evaluate(Xtest, Ytest)
print("_" * 100)
print("Test Loss and Accuracy")
print("res_7_units ", res_7_units)
hist_dict_7 = hist_7.history
hist_dict_7.keys()

Epoch 1/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 5s 92ms/step - acc: 0.5186 - loss: 0.2493 -
val_acc: 0.6613 - val_loss: 0.2330
Epoch 2/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 33ms/step - acc: 0.5917 - loss: 0.2312 -
val_acc: 0.8394 - val_loss: 0.1753
Epoch 3/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - acc: 0.6930 - loss: 0.1970 -
val_acc: 0.8556 - val_loss: 0.1365
```

```
Epoch 4/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - acc: 0.7530 - loss: 0.1681 -
val_acc: 0.8665 - val_loss: 0.1114
Epoch 5/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - acc: 0.7997 - loss: 0.1487 -
val_acc: 0.8824 - val_loss: 0.0941
Epoch 6/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - acc: 0.8408 - loss: 0.1245 -
val_acc: 0.8762 - val_loss: 0.0933
Epoch 7/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - acc: 0.8557 - loss: 0.1131 -
val_acc: 0.8780 - val_loss: 0.0902
Epoch 8/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 43ms/step - acc: 0.8817 - loss: 0.0976 -
val_acc: 0.8841 - val_loss: 0.0849
Epoch 9/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 49ms/step - acc: 0.8895 - loss: 0.0893 -
val_acc: 0.8878 - val_loss: 0.0849
Epoch 10/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 34ms/step - acc: 0.9040 - loss: 0.0793 -
val_acc: 0.8851 - val_loss: 0.0879
Epoch 11/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - acc: 0.9109 - loss: 0.0728 -
val_acc: 0.8870 - val_loss: 0.0901
Epoch 12/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - acc: 0.9178 - loss: 0.0680 -
val_acc: 0.8865 - val_loss: 0.0896
Epoch 13/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - acc: 0.9237 - loss: 0.0621 -
val_acc: 0.8866 - val_loss: 0.0927
Epoch 14/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - acc: 0.9254 - loss: 0.0602 -
val_acc: 0.8812 - val_loss: 0.0968
Epoch 15/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - acc: 0.9353 - loss: 0.0550 -
val_acc: 0.8871 - val_loss: 0.0949
Epoch 16/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - acc: 0.9320 - loss: 0.0548 -
val_acc: 0.8853 - val_loss: 0.0947
Epoch 17/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - acc: 0.9313 - loss: 0.0534 -
val_acc: 0.8864 - val_loss: 0.0959
Epoch 18/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 45ms/step - acc: 0.9387 - loss: 0.0481 -
val_acc: 0.8856 - val_loss: 0.0974
Epoch 19/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 36ms/step - acc: 0.9419 - loss: 0.0451 -
val_acc: 0.8857 - val_loss: 0.0980
Epoch 20/20
```

```
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - acc: 0.9405 - loss: 0.0461 -
val_acc: 0.8845 - val_loss: 0.0987
782/782 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - acc: 0.8769 - loss: 0.1050
_____
Test Loss and Accuracy
res_7_units  [0.10332717001438141, 0.8790799975395203]

dict_keys(['acc', 'loss', 'val_acc', 'val_loss'])
```

one layer with drop out of 0.5

```
mod_layer_8 = models.Sequential()
mod_layer_8.add(layers.Dense(16, activation='relu',
input_shape=(10000,)))
mod_layer_8.add(layers.Dropout(0.5))
mod_layer_8.add(layers.Dense(1, activation='sigmoid'))
mod_layer_8.compile(optimizer='rmsprop', loss='binary_crossentropy',
metrics=['acc'])

hist_8 = mod_layer_8.fit(partial_Xtrain, partial_Ytrain, epochs=20,
batch_size=512, validation_data=(Xval, Yval))

res_8_units = mod_layer_8.evaluate(Xtest, Ytest)
print("_" * 100)
print("Test Loss and Accuracy")
print("res_8_units ", res_8_units)
hist_dict_8 = hist_8.history
hist_dict_8.keys()

Epoch 1/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 61ms/step - acc: 0.6568 - loss: 0.6229 -
val_acc: 0.8561 - val_loss: 0.4504
Epoch 2/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 36ms/step - acc: 0.8430 - loss: 0.4356 -
val_acc: 0.8745 - val_loss: 0.3706
Epoch 3/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - acc: 0.8780 - loss: 0.3561 -
val_acc: 0.8846 - val_loss: 0.3266
Epoch 4/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - acc: 0.8920 - loss: 0.3068 -
val_acc: 0.8788 - val_loss: 0.3121
Epoch 5/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - acc: 0.9096 - loss: 0.2745 -
val_acc: 0.8913 - val_loss: 0.2867
Epoch 6/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 60ms/step - acc: 0.9162 - loss: 0.2464 -
val_acc: 0.8910 - val_loss: 0.2787
Epoch 7/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 60ms/step - acc: 0.9239 - loss: 0.2290 -
```

```
val_acc: 0.8891 - val_loss: 0.2743
Epoch 8/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 34ms/step - acc: 0.9336 - loss: 0.2074 -
val_acc: 0.8865 - val_loss: 0.2736
Epoch 9/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - acc: 0.9344 - loss: 0.1950 -
val_acc: 0.8877 - val_loss: 0.2714
Epoch 10/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - acc: 0.9442 - loss: 0.1757 -
val_acc: 0.8828 - val_loss: 0.2826
Epoch 11/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - acc: 0.9454 - loss: 0.1655 -
val_acc: 0.8872 - val_loss: 0.2730
Epoch 12/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - acc: 0.9567 - loss: 0.1521 -
val_acc: 0.8882 - val_loss: 0.2837
Epoch 13/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - acc: 0.9571 - loss: 0.1405 -
val_acc: 0.8866 - val_loss: 0.2834
Epoch 14/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - acc: 0.9643 - loss: 0.1288 -
val_acc: 0.8833 - val_loss: 0.2904
Epoch 15/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 53ms/step - acc: 0.9646 - loss: 0.1230 -
val_acc: 0.8815 - val_loss: 0.2973
Epoch 16/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 45ms/step - acc: 0.9680 - loss: 0.1161 -
val_acc: 0.8829 - val_loss: 0.2965
Epoch 17/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - acc: 0.9665 - loss: 0.1122 -
val_acc: 0.8851 - val_loss: 0.3074
Epoch 18/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - acc: 0.9718 - loss: 0.1022 -
val_acc: 0.8853 - val_loss: 0.3231
Epoch 19/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - acc: 0.9721 - loss: 0.1005 -
val_acc: 0.8838 - val_loss: 0.3252
Epoch 20/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - acc: 0.9739 - loss: 0.0915 -
val_acc: 0.8827 - val_loss: 0.3287
782/782 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - acc: 0.8712 - loss: 0.3563
_____
Test Loss and Accuracy
res_8_units  [0.3515060245990753, 0.8748000264167786]

dict_keys(['acc', 'loss', 'val_acc', 'val_loss'])
```

Using the L1 regulariser

```python
from tensorflow import keras
from tensorflow.keras import layers, regularizers,models

mod_layer_9 = models.Sequential()
mod_layer_9.add(layers.Dense(64, activation='relu',
activity_regularizer=regularizers.L1(0.01), input_shape=(10000,)))
mod_layer_9.add(layers.Dense(1, activation='sigmoid'))
mod_layer_9.compile(optimizer='rmsprop', loss='binary_crossentropy',
metrics=['acc'])

hist_9 = mod_layer_9.fit(partial_Xtrain, partial_Ytrain, epochs=20,
batch_size=512, validation_data=(Xval, Yval))

res_9_units = mod_layer_9.evaluate(Xtest, Ytest)
print("_" * 100)
print("Test Loss and Accuracy")
print("res_9_units ", res_9_units)
hist_dict_9 = hist_9.history
hist_dict_9.keys()
```

```
Epoch 1/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 4s 103ms/step - acc: 0.4954 - loss: 3.2740
- val_acc: 0.4958 - val_loss: 0.7110
Epoch 2/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 4s 68ms/step - acc: 0.5038 - loss: 0.6984 -
val_acc: 0.4965 - val_loss: 0.7012
Epoch 3/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 60ms/step - acc: 0.5009 - loss: 0.6938 -
val_acc: 0.4958 - val_loss: 0.6995
Epoch 4/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 60ms/step - acc: 0.5086 - loss: 0.6931 -
val_acc: 0.4953 - val_loss: 0.6990
Epoch 5/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 94ms/step - acc: 0.4989 - loss: 0.6932 -
val_acc: 0.4951 - val_loss: 0.6990
Epoch 6/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 89ms/step - acc: 0.5010 - loss: 0.6932 -
val_acc: 0.4952 - val_loss: 0.6990
Epoch 7/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 4s 61ms/step - acc: 0.5059 - loss: 0.6931 -
val_acc: 0.4953 - val_loss: 0.6990
Epoch 8/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 63ms/step - acc: 0.5093 - loss: 0.6931 -
val_acc: 0.4959 - val_loss: 0.6990
Epoch 9/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 59ms/step - acc: 0.5122 - loss: 0.6930 -
val_acc: 0.4956 - val_loss: 0.6990
Epoch 10/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 73ms/step - acc: 0.5014 - loss: 0.6932 -
val_acc: 0.4953 - val_loss: 0.6990
```

```
Epoch 11/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 4s 120ms/step - acc: 0.5005 - loss: 0.6932
- val_acc: 0.4953 - val_loss: 0.6990
Epoch 12/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 70ms/step - acc: 0.5112 - loss: 0.6930 -
val_acc: 0.4953 - val_loss: 0.6990
Epoch 13/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 61ms/step - acc: 0.5060 - loss: 0.6931 -
val_acc: 0.4950 - val_loss: 0.6990
Epoch 14/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 61ms/step - acc: 0.5061 - loss: 0.6931 -
val_acc: 0.4951 - val_loss: 0.6990
Epoch 15/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 62ms/step - acc: 0.5053 - loss: 0.6931 -
val_acc: 0.4950 - val_loss: 0.6990
Epoch 16/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 109ms/step - acc: 0.5004 - loss: 0.6932
- val_acc: 0.4950 - val_loss: 0.6990
Epoch 17/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 4s 60ms/step - acc: 0.4991 - loss: 0.6932 -
val_acc: 0.4950 - val_loss: 0.6990
Epoch 18/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 62ms/step - acc: 0.5021 - loss: 0.6932 -
val_acc: 0.4950 - val_loss: 0.6990
Epoch 19/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 60ms/step - acc: 0.5061 - loss: 0.6931 -
val_acc: 0.4950 - val_loss: 0.6990
Epoch 20/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 89ms/step - acc: 0.5084 - loss: 0.6930 -
val_acc: 0.4953 - val_loss: 0.6990
782/782 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - acc: 0.5078 - loss: 0.6935
_____
_____
Test Loss and Accuracy
res_9_units  [0.693612277507782, 0.5004799962043762]

dict_keys(['acc', 'loss', 'val_acc', 'val_loss'])
```

Using the L2 regulariser

```
mod_layer_10 = models.Sequential()
mod_layer_10.add(layers.Dense(64, activation='relu',
activity_regularizer=regularizers.L2(0.01), input_shape=(10000,)))
mod_layer_10.add(layers.Dense(1, activation='sigmoid'))
mod_layer_10.compile(optimizer='rmsprop', loss='binary_crossentropy',
metrics=['acc'])

hist_10 = mod_layer_10.fit(partial_Xtrain, partial_Ytrain, epochs=20,
batch_size=512, validation_data=(Xval, Yval))
```

```
res_10_units = mod_layer_10.evaluate(Xtest, Ytest)
print("_" * 100)
print("Test Loss and Accuracy")
print("res_10_units ", res_10_units)
hist_dict_10 = hist_10.history
hist_dict_10.keys()
```

```
Epoch 1/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 5s 94ms/step - acc: 0.5066 - loss: 1.2568 -
val_acc: 0.5029 - val_loss: 0.6975
Epoch 2/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 4s 62ms/step - acc: 0.5141 - loss: 0.6944 -
val_acc: 0.4977 - val_loss: 0.6951
Epoch 3/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 61ms/step - acc: 0.5126 - loss: 0.6932 -
val_acc: 0.4975 - val_loss: 0.6946
Epoch 4/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 86ms/step - acc: 0.5005 - loss: 0.6932 -
val_acc: 0.4950 - val_loss: 0.6945
Epoch 5/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 5s 69ms/step - acc: 0.5069 - loss: 0.6931 -
val_acc: 0.4965 - val_loss: 0.6944
Epoch 6/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 59ms/step - acc: 0.4979 - loss: 0.6932 -
val_acc: 0.4963 - val_loss: 0.6942
Epoch 7/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 59ms/step - acc: 0.5088 - loss: 0.6931 -
val_acc: 0.4957 - val_loss: 0.6941
Epoch 8/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 64ms/step - acc: 0.5056 - loss: 0.6931 -
val_acc: 0.4959 - val_loss: 0.6940
Epoch 9/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 4s 100ms/step - acc: 0.5072 - loss: 0.6931
- val_acc: 0.4956 - val_loss: 0.6940
Epoch 10/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 4s 62ms/step - acc: 0.5015 - loss: 0.6932 -
val_acc: 0.4951 - val_loss: 0.6940
Epoch 11/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 60ms/step - acc: 0.5023 - loss: 0.6931 -
val_acc: 0.4952 - val_loss: 0.6940
Epoch 12/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 60ms/step - acc: 0.5012 - loss: 0.6932 -
val_acc: 0.4950 - val_loss: 0.6940
Epoch 13/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 60ms/step - acc: 0.4972 - loss: 0.6932 -
val_acc: 0.4950 - val_loss: 0.6940
Epoch 14/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 4s 100ms/step - acc: 0.5046 - loss: 0.6931
- val_acc: 0.4952 - val_loss: 0.6939
```

```
Epoch 15/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 61ms/step - acc: 0.5007 - loss: 0.6932 -
val_acc: 0.4949 - val_loss: 0.6940
Epoch 16/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 63ms/step - acc: 0.5058 - loss: 0.6931 -
val_acc: 0.4951 - val_loss: 0.6939
Epoch 17/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 65ms/step - acc: 0.4997 - loss: 0.6932 -
val_acc: 0.4947 - val_loss: 0.6940
Epoch 18/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 60ms/step - acc: 0.5036 - loss: 0.6931 -
val_acc: 0.4951 - val_loss: 0.6940
Epoch 19/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 89ms/step - acc: 0.5031 - loss: 0.6931 -
val_acc: 0.4954 - val_loss: 0.6939
Epoch 20/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 88ms/step - acc: 0.5021 - loss: 0.6931 -
val_acc: 0.4943 - val_loss: 0.6940
782/782 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - acc: 0.5080 - loss: 0.6931
____________________________________________________________________
____________________
Test Loss and Accuracy
res_10_units  [0.6932082176208496, 0.5007200241088867]

dict_keys(['acc', 'loss', 'val_acc', 'val_loss'])
```

Using adam optimizer inplace of rmsprop

```
mod_layer_11 = models.Sequential()
mod_layer_11.add(layers.Dense(64, activation='relu',
activity_regularizer=regularizers.L1(0.01), input_shape=(10000,)))
mod_layer_11.add(layers.Dense(1, activation='sigmoid'))
mod_layer_11.compile(optimizer='rmsprop', loss='binary_crossentropy',
metrics=['acc'])

hist_11 = mod_layer_11.fit(partial_Xtrain, partial_Ytrain, epochs=20,
batch_size=512, validation_data=(Xval, Yval))

res_11_units = mod_layer_11.evaluate(Xtest, Ytest)
print("_" * 100)
print("Test Loss and Accuracy")
print("res_11_units ", res_11_units)
hist_dict_11 = hist_11.history
hist_dict_11.keys()

Epoch 1/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 4s 107ms/step - acc: 0.5063 - loss: 3.7010
- val_acc: 0.4959 - val_loss: 0.7167
Epoch 2/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 92ms/step - acc: 0.5044 - loss: 0.7000 -
```

```
val_acc: 0.4947 - val_loss: 0.7029
Epoch 3/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 4s 58ms/step - acc: 0.5068 - loss: 0.6937 -
val_acc: 0.4948 - val_loss: 0.7007
Epoch 4/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 59ms/step - acc: 0.4983 - loss: 0.6932 -
val_acc: 0.4941 - val_loss: 0.7005
Epoch 5/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 64ms/step - acc: 0.5019 - loss: 0.6931 -
val_acc: 0.4941 - val_loss: 0.7005
Epoch 6/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 67ms/step - acc: 0.5054 - loss: 0.6931 -
val_acc: 0.4942 - val_loss: 0.7005
Epoch 7/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 97ms/step - acc: 0.5078 - loss: 0.6931 -
val_acc: 0.4942 - val_loss: 0.7005
Epoch 8/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 60ms/step - acc: 0.4994 - loss: 0.6932 -
val_acc: 0.4942 - val_loss: 0.7006
Epoch 9/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 62ms/step - acc: 0.5026 - loss: 0.6931 -
val_acc: 0.4943 - val_loss: 0.7006
Epoch 10/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 60ms/step - acc: 0.4976 - loss: 0.6932 -
val_acc: 0.4944 - val_loss: 0.7006
Epoch 11/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 62ms/step - acc: 0.5044 - loss: 0.6931 -
val_acc: 0.4943 - val_loss: 0.7006
Epoch 12/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 105ms/step - acc: 0.5070 - loss: 0.6931
- val_acc: 0.4944 - val_loss: 0.7006
Epoch 13/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 75ms/step - acc: 0.5077 - loss: 0.6931 -
val_acc: 0.4944 - val_loss: 0.7006
Epoch 14/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 63ms/step - acc: 0.5092 - loss: 0.6930 -
val_acc: 0.4943 - val_loss: 0.7006
Epoch 15/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 61ms/step - acc: 0.4994 - loss: 0.6932 -
val_acc: 0.4945 - val_loss: 0.7006
Epoch 16/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 61ms/step - acc: 0.5034 - loss: 0.6931 -
val_acc: 0.4943 - val_loss: 0.7006
Epoch 17/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 63ms/step - acc: 0.5057 - loss: 0.6931 -
val_acc: 0.4943 - val_loss: 0.7006
Epoch 18/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 4s 122ms/step - acc: 0.5041 - loss: 0.6931
- val_acc: 0.4942 - val_loss: 0.7006
```

```
Epoch 19/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 65ms/step - acc: 0.5059 - loss: 0.6931 -
val_acc: 0.4944 - val_loss: 0.7006
Epoch 20/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 97ms/step - acc: 0.5018 - loss: 0.6932 -
val_acc: 0.4944 - val_loss: 0.7006
782/782 ━━━━━━━━━━━━━━━━━━━━ 4s 5ms/step - acc: 0.5078 - loss: 0.6936
_________________________________________________________________________
_____________________________
Test Loss and Accuracy
res_11_units  [0.6937016248703003, 0.5006399750709534]

dict_keys(['acc', 'loss', 'val_acc', 'val_loss'])
```