https://colab.research.google.com/drive/1qeBwrosOfCNEeDJe8GFQ_wWOUw6gmru8 is the location of the original file.

data set is taken from above link

```
import tensorflow.keras as tf
from tensorflow.keras.datasets import imdb
(train_data, train_labels), (test_data, test_labels) = imdb.load_data(
    num_words=10000)
```

```
WARNING:tensorflow:From c:\Users\imtha\AppData\Local\Programs\Python\Python39\lib\site-packages\keras\src\losses.py:2976: The name tf.lc
```

reviewing

```
train_data[0]
```

```
[1,
 14,
 22,
 16,
 43,
 530,
 973,
 1622,
 1385,
 65,
 458,
 4468,
 66,
 3941,
 4,
 173,
 36,
 256,
 5,
 25,
 100,
 43,
 838,
 112,
 50,
 670,
 2,
 9,
 35,
 480,
 284,
 5,
 150,
 4,
 172,
 112,
 167,
 2,
 336,
 385,
 39,
 4,
 172,
 4536,
 1111,
 17,
 546,
 38,
 13,
 447,
 4,
 192,
 50,
 16,
 6,
 147,
 2025,
 19,
```

verify the category assigned to the initial review

```
train_labels[0]
```

```
max([max(sequence) for sequence in train_data])
```

```
     9999
```

"""Decoding and displaying movie reviews in text"""

```
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]])
```

As can be observed, the label is 1 and the initial review is favorable.

organizing the data

Multi-hot encoding is used to encode the integer sequences """

```
import numpy as np
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)
```

```
x_train[0]
```

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

developing the model

```
from tensorflow import keras
from tensorflow.keras import layers
# #Here I am  using two hidden layers, each with 16 nodes, and only one node in the output layer for either +ve or -ve output. ReLu is used
model = keras.Sequential([
    layers.Dense(16, activation="relu"),
    layers.Dense(16, activation="relu"),
    layers.Dense(1, activation="sigmoid")
])
```

```
     WARNING:tensorflow:From c:\Users\imtha\AppData\Local\Programs\Python\Python39\lib\site-packages\keras\src\backend.py:873: The name tf.ge
```

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

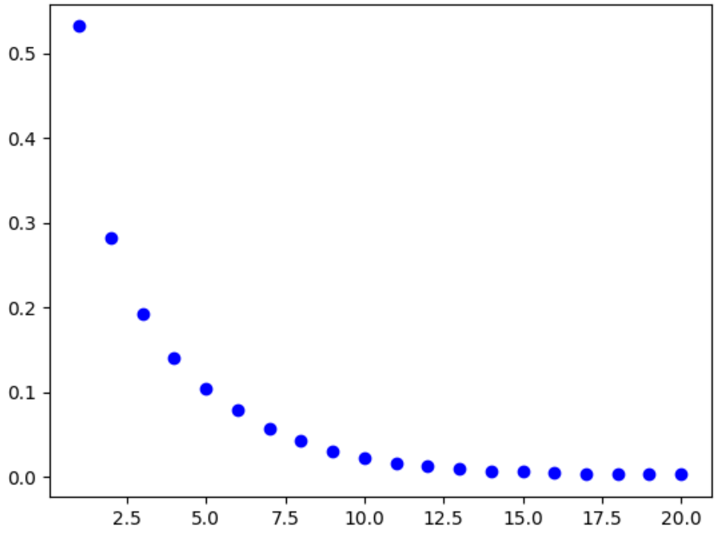
```
     WARNING:tensorflow:From c:\Users\imtha\AppData\Local\Programs\Python\Python39\lib\site-packages\keras\src\optimizers\__init__.py:309: Th
```

```
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=20,
                    batch_size=512,
                    validation_data=(x_val, y_val))

history_dict = history.history
history_dict.keys()
```

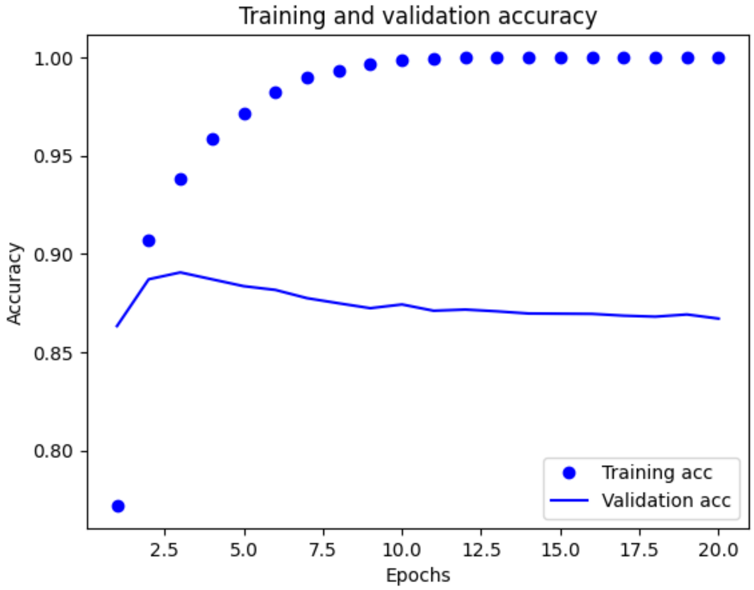
```
Epoch 1/20
WARNING:tensorflow:From c:\Users\imtha\AppData\Local\Programs\Python\Python39\lib\site-packages\keras\src\utils\tf_utils.py:492: The nam

WARNING:tensorflow:From c:\Users\imtha\AppData\Local\Programs\Python\Python39\lib\site-packages\keras\src\engine\base_layer_utils.py:384

30/30 [==============================] - 2s 33ms/step - loss: 0.5318 - accuracy: 0.7719 - val_loss: 0.3795 - val_accuracy: 0.8632
Epoch 2/20
30/30 [==============================] - 0s 9ms/step - loss: 0.2827 - accuracy: 0.9071 - val_loss: 0.2894 - val_accuracy: 0.8870
Epoch 3/20
30/30 [==============================] - 0s 9ms/step - loss: 0.1918 - accuracy: 0.9380 - val_loss: 0.2754 - val_accuracy: 0.8905
Epoch 4/20
30/30 [==============================] - 0s 10ms/step - loss: 0.1401 - accuracy: 0.9585 - val_loss: 0.2844 - val_accuracy: 0.8870
Epoch 5/20
30/30 [==============================] - 0s 10ms/step - loss: 0.1041 - accuracy: 0.9715 - val_loss: 0.3066 - val_accuracy: 0.8835
Epoch 6/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0781 - accuracy: 0.9819 - val_loss: 0.3346 - val_accuracy: 0.8816
Epoch 7/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0566 - accuracy: 0.9895 - val_loss: 0.3644 - val_accuracy: 0.8774
Epoch 8/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0419 - accuracy: 0.9930 - val_loss: 0.3990 - val_accuracy: 0.8748
Epoch 9/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0306 - accuracy: 0.9962 - val_loss: 0.4326 - val_accuracy: 0.8723
Epoch 10/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0223 - accuracy: 0.9985 - val_loss: 0.4586 - val_accuracy: 0.8742
Epoch 11/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0163 - accuracy: 0.9991 - val_loss: 0.4903 - val_accuracy: 0.8710
Epoch 12/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0119 - accuracy: 0.9997 - val_loss: 0.5161 - val_accuracy: 0.8716
Epoch 13/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0092 - accuracy: 0.9997 - val_loss: 0.5405 - val_accuracy: 0.8707
Epoch 14/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0071 - accuracy: 0.9999 - val_loss: 0.5659 - val_accuracy: 0.8696
Epoch 15/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0057 - accuracy: 0.9999 - val_loss: 0.5843 - val_accuracy: 0.8695
Epoch 16/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0047 - accuracy: 0.9999 - val_loss: 0.6045 - val_accuracy: 0.8694
Epoch 17/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0039 - accuracy: 0.9999 - val_loss: 0.6216 - val_accuracy: 0.8685
Epoch 18/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0033 - accuracy: 0.9999 - val_loss: 0.6389 - val_accuracy: 0.8680
Epoch 19/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0029 - accuracy: 0.9999 - val_loss: 0.6524 - val_accuracy: 0.8691
Epoch 20/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0025 - accuracy: 0.9999 - val_loss: 0.6692 - val_accuracy: 0.8670
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

```
import matplotlib.pyplot as plt
history_dict = history.history
loss_values = history_dict["loss"]
val_loss_values = history_dict["val_loss"]
epochs = range(1, len(loss_values) + 1)
plt.plot(epochs, loss_values, "bo", label="Training loss")
```

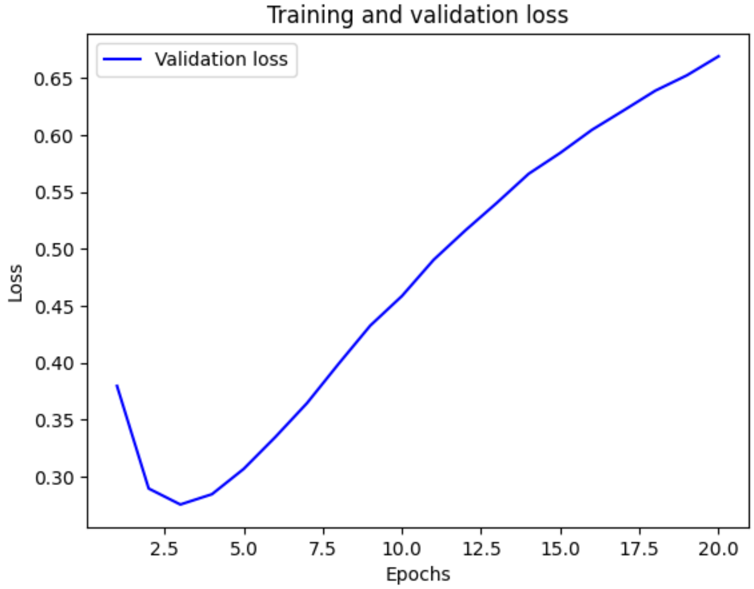
```
[<matplotlib.lines.Line2D at 0x235ba89f850>]
```



```
plt.clf()
acc = history_dict["accuracy"]
val_acc = history_dict["val_accuracy"]
plt.plot(epochs, acc, "bo", label="Training acc")
plt.plot(epochs, val_acc, "b", label="Validation acc")
plt.title("Training and validation accuracy")
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.legend()
plt.show()
```



```
plt.plot(epochs, val_loss_values, "b", label="Validation loss")
plt.title("Training and validation loss")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend()
plt.show()
```



```
model = keras.Sequential([
    layers.Dense(16, activation="relu"),
    layers.Dense(16, activation="relu"),
    layers.Dense(1, activation="sigmoid")
])
#Here i am using three epochs to retrain the model here.
model.compile(optimizer="adam",
              loss="binary_crossentropy",
              metrics=["accuracy"])
model.fit(x_train, y_train, epochs=4, batch_size=512)
results = model.evaluate(x_test, y_test)

results
```

```
Epoch 1/4
49/49 [==============================] - 1s 6ms/step - loss: 0.4937 - accuracy: 0.7709
Epoch 2/4
49/49 [==============================] - 1s 14ms/step - loss: 0.2549 - accuracy: 0.9124
Epoch 3/4
49/49 [==============================] - 0s 6ms/step - loss: 0.1865 - accuracy: 0.9361
Epoch 4/4
49/49 [==============================] - 0s 5ms/step - loss: 0.1492 - accuracy: 0.9502
782/782 [==============================] - 1s 1ms/step - loss: 0.3060 - accuracy: 0.8808
[0.3060041666030884, 0.880840003490448]
```

```
model1_1 = keras.Sequential([
    layers.Dense(16, activation="relu"),
    layers.Dense(1, activation="sigmoid")
])
```

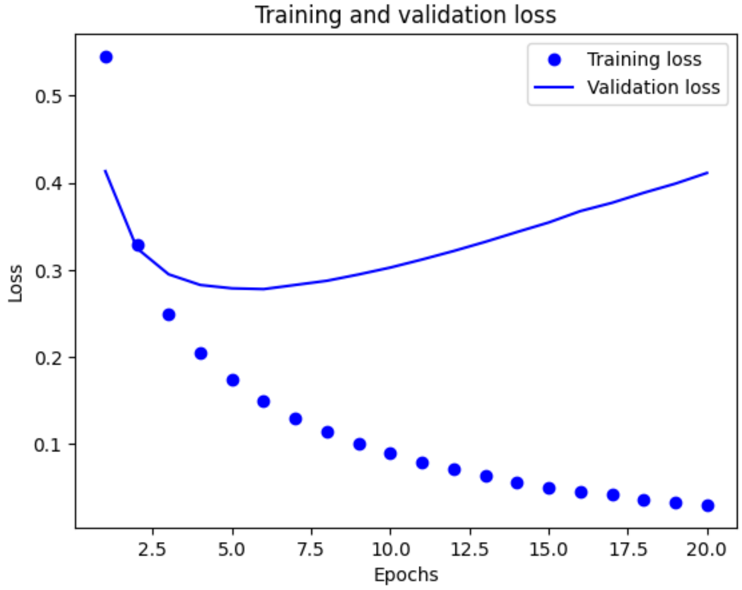
```
model1_3 = keras.Sequential([
    layers.Dense(16, activation="relu"),
    layers.Dense(16, activation="relu"),
    layers.Dense(16, activation="relu"),
    layers.Dense(1, activation="sigmoid")
])
```

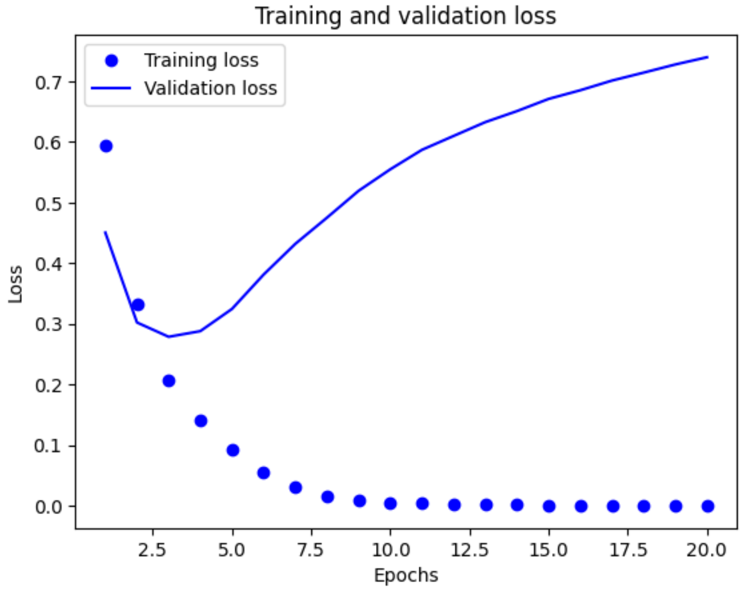
```
model1_1.compile(optimizer="adam",
              loss="binary_crossentropy",
              metrics=["accuracy"])

model1_3.compile(optimizer="adam",
              loss="binary_crossentropy",
              metrics=["accuracy"])
```

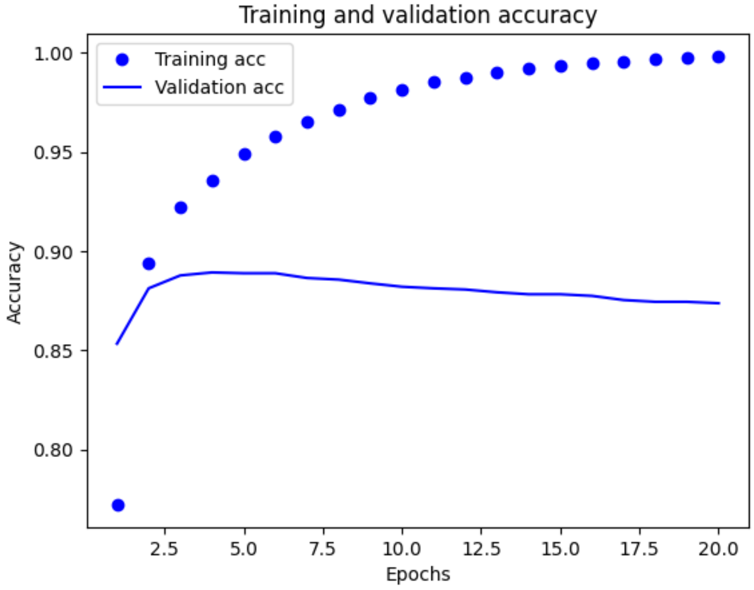
```
history1_1 = model1_1.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val, y_val))

history1_3 = model1_3.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val, y_val))
```

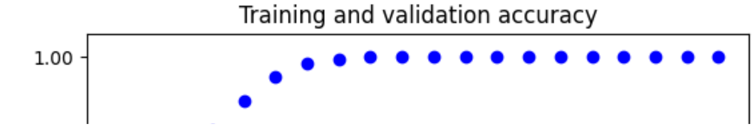
```
Epoch 1/20
30/30 [==============================] - 2s 30ms/step - loss: 0.5452 - accuracy: 0.7723 - val_loss: 0.4134 - val_accuracy: 0.8533
Epoch 2/20
30/30 [==============================] - 0s 9ms/step - loss: 0.3295 - accuracy: 0.8937 - val_loss: 0.3249 - val_accuracy: 0.8812
Epoch 3/20
30/30 [==============================] - 0s 10ms/step - loss: 0.2496 - accuracy: 0.9221 - val_loss: 0.2950 - val_accuracy: 0.8877
Epoch 4/20
30/30 [==============================] - 0s 10ms/step - loss: 0.2055 - accuracy: 0.9355 - val_loss: 0.2828 - val_accuracy: 0.8892
Epoch 5/20
30/30 [==============================] - 0s 10ms/step - loss: 0.1737 - accuracy: 0.9491 - val_loss: 0.2789 - val_accuracy: 0.8888
Epoch 6/20
30/30 [==============================] - 0s 10ms/step - loss: 0.1497 - accuracy: 0.9577 - val_loss: 0.2780 - val_accuracy: 0.8888
Epoch 7/20
30/30 [==============================] - 0s 10ms/step - loss: 0.1299 - accuracy: 0.9651 - val_loss: 0.2828 - val_accuracy: 0.8864
Epoch 8/20
30/30 [==============================] - 0s 10ms/step - loss: 0.1145 - accuracy: 0.9712 - val_loss: 0.2876 - val_accuracy: 0.8856
Epoch 9/20
30/30 [==============================] - 0s 10ms/step - loss: 0.1010 - accuracy: 0.9770 - val_loss: 0.2949 - val_accuracy: 0.8837
Epoch 10/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0893 - accuracy: 0.9812 - val_loss: 0.3028 - val_accuracy: 0.8820
Epoch 11/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0795 - accuracy: 0.9850 - val_loss: 0.3120 - val_accuracy: 0.8812
Epoch 12/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0710 - accuracy: 0.9871 - val_loss: 0.3218 - val_accuracy: 0.8806
Epoch 13/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0637 - accuracy: 0.9898 - val_loss: 0.3323 - val_accuracy: 0.8792
Epoch 14/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0569 - accuracy: 0.9923 - val_loss: 0.3436 - val_accuracy: 0.8782
Epoch 15/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0509 - accuracy: 0.9934 - val_loss: 0.3545 - val_accuracy: 0.8782
Epoch 16/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0457 - accuracy: 0.9947 - val_loss: 0.3676 - val_accuracy: 0.8774
Epoch 17/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0418 - accuracy: 0.9955 - val_loss: 0.3771 - val_accuracy: 0.8753
Epoch 18/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0372 - accuracy: 0.9966 - val_loss: 0.3886 - val_accuracy: 0.8744
Epoch 19/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0336 - accuracy: 0.9975 - val_loss: 0.3991 - val_accuracy: 0.8744
Epoch 20/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0304 - accuracy: 0.9981 - val_loss: 0.4114 - val_accuracy: 0.8737
Epoch 1/20
30/30 [==============================] - 2s 28ms/step - loss: 0.5946 - accuracy: 0.7209 - val_loss: 0.4504 - val_accuracy: 0.8425
Epoch 2/20
30/30 [==============================] - 0s 10ms/step - loss: 0.3326 - accuracy: 0.8908 - val_loss: 0.3023 - val_accuracy: 0.8842
Epoch 3/20
30/30 [==============================] - 0s 10ms/step - loss: 0.2078 - accuracy: 0.9285 - val_loss: 0.2787 - val_accuracy: 0.8901
Epoch 4/20
30/30 [==============================] - 0s 10ms/step - loss: 0.1405 - accuracy: 0.9546 - val_loss: 0.2880 - val_accuracy: 0.8879
Epoch 5/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0932 - accuracy: 0.9729 - val_loss: 0.3246 - val_accuracy: 0.8818
Epoch 6/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0555 - accuracy: 0.9875 - val_loss: 0.3813 - val_accuracy: 0.8766
Epoch 7/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0304 - accuracy: 0.9955 - val_loss: 0.4319 - val_accuracy: 0.8782
Epoch 8/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0165 - accuracy: 0.9982 - val_loss: 0.4751 - val_accuracy: 0.8735
Epoch 9/20
```

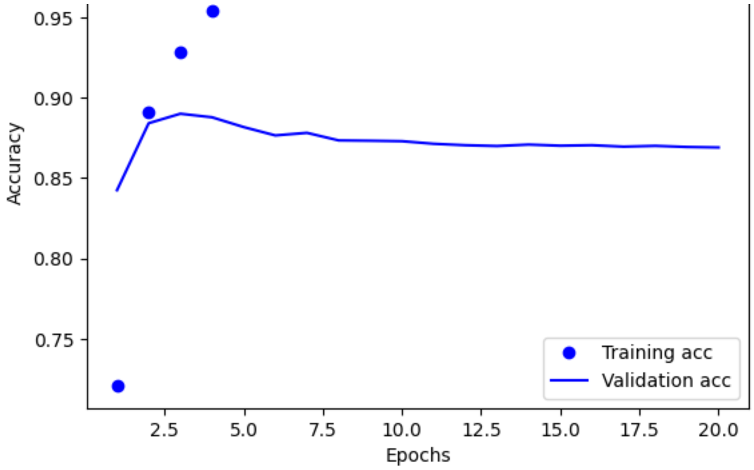
```
historyp1_1 = history1_1.history
historyp1_1.keys()

historyp1_3 = history1_1.history
historyp1_3.keys()

historyp1_1 = history1_1.history
loss_values1 = historyp1_1["loss"]
val_loss_values1 = historyp1_1["val_loss"]
epochs = range(1, len(loss_values) + 1)
plt.plot(epochs, loss_values1, "bo", label="Training loss")
plt.plot(epochs, val_loss_values1, "b", label="Validation loss")
plt.title("Training and validation loss")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend()
plt.show()

historyp1_3 = history1_3.history
loss_values3 = historyp1_3["loss"]
val_loss_values3 = historyp1_3["val_loss"]
epochs = range(1, len(loss_values) + 1)
plt.plot(epochs, loss_values3, "bo", label="Training loss")
plt.plot(epochs, val_loss_values3, "b", label="Validation loss")
plt.title("Training and validation loss")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend()
plt.show()

plt.clf()
acc1 = historyp1_1["accuracy"]
val_acc1 = historyp1_1["val_accuracy"]
plt.plot(epochs, acc1, "bo", label="Training acc")
plt.plot(epochs, val_acc1, "b", label="Validation acc")
plt.title("Training and validation accuracy")
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.legend()
plt.show()

plt.clf()
acc3 = historyp1_3["accuracy"]
val_acc3 = historyp1_3["val_accuracy"]
plt.plot(epochs, acc3, "bo", label="Training acc")
plt.plot(epochs, val_acc3, "b", label="Validation acc")
plt.title("Training and validation accuracy")
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.legend()
plt.show()
```


Training and validation loss
Training loss
Validation loss
Loss
0.5
0.4
0.3
0.2
0.1
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Epochs


Training and validation loss
Training loss
Validation loss
Loss
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Epochs

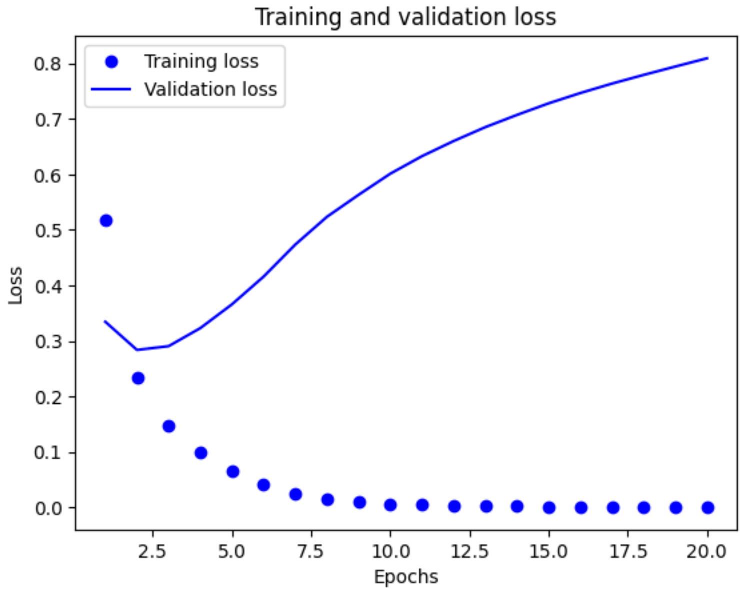

Training and validation accuracy
Training acc
Validation acc
Accuracy
1.00
0.95
0.90
0.85
0.80
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Epochs

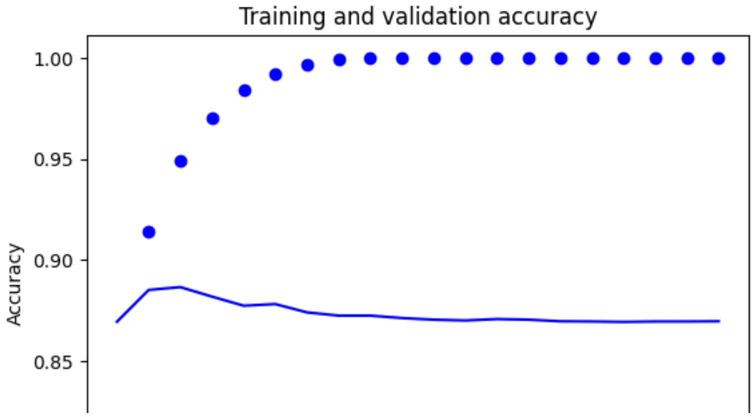

Training and validation accuracy
1.00

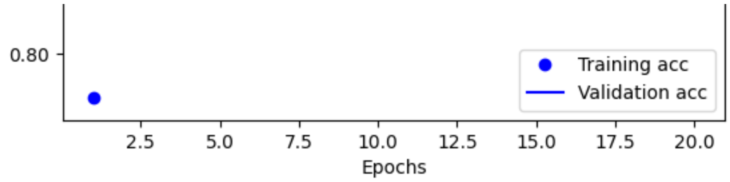


```
model2 = keras.Sequential([
    layers.Dense(32, activation="relu"),
    layers.Dense(64, activation="relu"),
    layers.Dense(1, activation="sigmoid")
])

model2.compile(optimizer="adam",
              loss="binary_crossentropy",
              metrics=["accuracy"])

hist2 = model2.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val, y_val))

histp2 = hist2.history
loss_values = histp2["loss"]
val_loss_values = histp2["val_loss"]
epochs = range(1, len(loss_values) + 1)
plt.plot(epochs, loss_values, "bo", label="Training loss")
plt.plot(epochs, val_loss_values, "b", label="Validation loss")
plt.title("Training and validation loss")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend()
plt.show()

plt.clf()
acc = histp2["accuracy"]
val_acc = histp2["val_accuracy"]
plt.plot(epochs, acc, "bo", label="Training acc")
plt.plot(epochs, val_acc, "b", label="Validation acc")
plt.title("Training and validation accuracy")
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.legend()
plt.show()
```

```
Epoch 1/20
30/30 [==============================] - 2s 38ms/step - loss: 0.5169 - accuracy: 0.7781 - val_loss: 0.3344 - val_accuracy: 0.8695
Epoch 2/20
30/30 [==============================] - 0s 12ms/step - loss: 0.2345 - accuracy: 0.9138 - val_loss: 0.2836 - val_accuracy: 0.8852
Epoch 3/20
30/30 [==============================] - 0s 12ms/step - loss: 0.1482 - accuracy: 0.9492 - val_loss: 0.2905 - val_accuracy: 0.8866
Epoch 4/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0981 - accuracy: 0.9703 - val_loss: 0.3230 - val_accuracy: 0.8819
Epoch 5/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0646 - accuracy: 0.9839 - val_loss: 0.3659 - val_accuracy: 0.8774
Epoch 6/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0409 - accuracy: 0.9917 - val_loss: 0.4159 - val_accuracy: 0.8782
Epoch 7/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0257 - accuracy: 0.9965 - val_loss: 0.4736 - val_accuracy: 0.8741
Epoch 8/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0153 - accuracy: 0.9993 - val_loss: 0.5236 - val_accuracy: 0.8725
Epoch 9/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0091 - accuracy: 0.9999 - val_loss: 0.5632 - val_accuracy: 0.8725
Epoch 10/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0060 - accuracy: 0.9999 - val_loss: 0.6012 - val_accuracy: 0.8713
Epoch 11/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0043 - accuracy: 0.9999 - val_loss: 0.6328 - val_accuracy: 0.8705
Epoch 12/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0032 - accuracy: 0.9999 - val_loss: 0.6601 - val_accuracy: 0.8701
Epoch 13/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0024 - accuracy: 1.0000 - val_loss: 0.6851 - val_accuracy: 0.8708
Epoch 14/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0019 - accuracy: 1.0000 - val_loss: 0.7071 - val_accuracy: 0.8705
Epoch 15/20
30/30 [==============================] - 0s 14ms/step - loss: 0.0016 - accuracy: 1.0000 - val_loss: 0.7280 - val_accuracy: 0.8697
Epoch 16/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0013 - accuracy: 1.0000 - val_loss: 0.7465 - val_accuracy: 0.8696
Epoch 17/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0011 - accuracy: 1.0000 - val_loss: 0.7636 - val_accuracy: 0.8694
Epoch 18/20
30/30 [==============================] - 0s 13ms/step - loss: 9.5230e-04 - accuracy: 1.0000 - val_loss: 0.7792 - val_accuracy: 0.8696
Epoch 19/20
30/30 [==============================] - 0s 12ms/step - loss: 8.2709e-04 - accuracy: 1.0000 - val_loss: 0.7943 - val_accuracy: 0.8696
Epoch 20/20
30/30 [==============================] - 0s 11ms/step - loss: 7.2530e-04 - accuracy: 1.0000 - val_loss: 0.8091 - val_accuracy: 0.8697
```





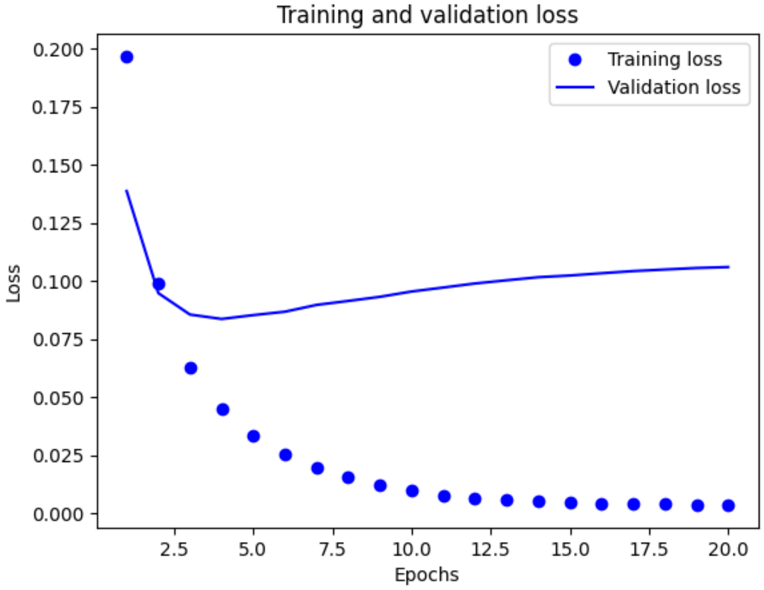


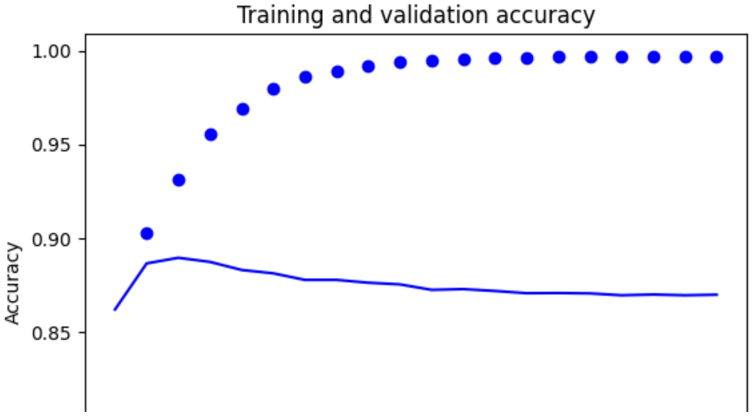
```
model3 = keras.Sequential([
    layers.Dense(16, activation="relu"),
    layers.Dense(16, activation="relu"),
    layers.Dense(1, activation="sigmoid")
])
```

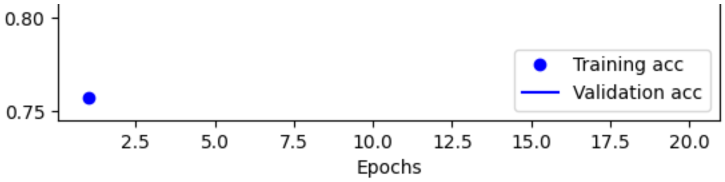
making use of binary_crossentropy in place of the MSE loss function

```
model3.compile(optimizer="adam",
              loss="mse",
              metrics=["accuracy"])

hist3 = model3.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val, y_val))

histp3 = hist3.history
loss_values = histp3["loss"]
val_loss_values = histp3["val_loss"]
epochs = range(1, len(loss_values) + 1)
plt.plot(epochs, loss_values, "bo", label="Training loss")
plt.plot(epochs, val_loss_values, "b", label="Validation loss")
plt.title("Training and validation loss")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend()
plt.show()

plt.clf()
acc = histp3["accuracy"]
val_acc = histp3["val_accuracy"]
plt.plot(epochs, acc, "bo", label="Training acc")
plt.plot(epochs, val_acc, "b", label="Validation acc")
plt.title("Training and validation accuracy")
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.legend()
plt.show()
```

```
Epoch 1/20
30/30 [==============================] - 2s 31ms/step - loss: 0.1967 - accuracy: 0.7573 - val_loss: 0.1387 - val_accuracy: 0.8619
Epoch 2/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0990 - accuracy: 0.9030 - val_loss: 0.0948 - val_accuracy: 0.8865
Epoch 3/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0626 - accuracy: 0.9316 - val_loss: 0.0855 - val_accuracy: 0.8895
Epoch 4/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0450 - accuracy: 0.9553 - val_loss: 0.0836 - val_accuracy: 0.8873
Epoch 5/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0335 - accuracy: 0.9687 - val_loss: 0.0853 - val_accuracy: 0.8830
Epoch 6/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0256 - accuracy: 0.9799 - val_loss: 0.0868 - val_accuracy: 0.8812
Epoch 7/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0199 - accuracy: 0.9859 - val_loss: 0.0897 - val_accuracy: 0.8777
Epoch 8/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0154 - accuracy: 0.9893 - val_loss: 0.0914 - val_accuracy: 0.8777
Epoch 9/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0123 - accuracy: 0.9921 - val_loss: 0.0932 - val_accuracy: 0.8762
Epoch 10/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0096 - accuracy: 0.9939 - val_loss: 0.0955 - val_accuracy: 0.8753
Epoch 11/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0078 - accuracy: 0.9951 - val_loss: 0.0972 - val_accuracy: 0.8724
Epoch 12/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0067 - accuracy: 0.9957 - val_loss: 0.0989 - val_accuracy: 0.8728
Epoch 13/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0058 - accuracy: 0.9961 - val_loss: 0.1003 - val_accuracy: 0.8718
Epoch 14/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0051 - accuracy: 0.9965 - val_loss: 0.1016 - val_accuracy: 0.8706
Epoch 15/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0046 - accuracy: 0.9967 - val_loss: 0.1023 - val_accuracy: 0.8707
Epoch 16/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0043 - accuracy: 0.9967 - val_loss: 0.1033 - val_accuracy: 0.8705
Epoch 17/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0040 - accuracy: 0.9969 - val_loss: 0.1043 - val_accuracy: 0.8695
Epoch 18/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0038 - accuracy: 0.9969 - val_loss: 0.1049 - val_accuracy: 0.8699
Epoch 19/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0037 - accuracy: 0.9969 - val_loss: 0.1056 - val_accuracy: 0.8695
Epoch 20/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0036 - accuracy: 0.9969 - val_loss: 0.1060 - val_accuracy: 0.8698
```







insted of relu, using tanh activation

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

model4.compile(optimizer="adam",
              loss="mse",
              metrics=["accuracy"])

hist4 = model4.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val, y_val))

histp4 = hist4.history
loss_values = histp4["loss"]
val_loss_values = histp4["val_loss"]
epochs = range(1, len(loss_values) + 1)
plt.plot(epochs, loss_values, "bo", label="Training loss")
plt.plot(epochs, val_loss_values, "b", label="Validation loss")
plt.title("Training and validation loss")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend()
plt.show()

plt.clf()
acc = histp4["accuracy"]
val_acc = histp4["val_accuracy"]
plt.plot(epochs, acc, "bo", label="Training acc")
plt.plot(epochs, val_acc, "b", label="Validation acc")
plt.title("Training and validation accuracy")
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.legend()
```