This is a companion notebook for the book Deep Learning with Python, Second Edition. For readability, it only contains runnable code blocks and section titles, and omits everything else in the book: text paragraphs, figures, and pseudocode.

If you want to be able to follow what's going on, I recommend reading the notebook side by side with your copy of the book.

This notebook was generated for TensorFlow 2.6.

Getting started with neural networks: Classification and regression

Classifying movie reviews: A binary classification example

The IMDB dataset

Loading the IMDB dataset

Importing an IMDB dataset from Keras. Here, we'll look at the 10000 words.

Dividing the dataset into training and test sets.

```
from tensorflow.keras.datasets import imdb
(train_data, train_labels), (test_data, test_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
```

Simply printing the first review from the training dataset.

```
train_data[0]
```

```
 92,
 25,
 104,
 4,
 226,
 65,
 16,
 38,
 1334,
 88,
 12,
 16,
 283,
 5,
 16,
 4472,
 113,
 103,
 32,
 15,
 16,
 5345,
 19,
 178,
 321
```

checking the first review's label

```
train_labels[0]
```

```
1
```

```
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 seen, the first review is positive, and the label is 1.
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/imdb_word_index.json
1641221/1641221 [==============================] - 0s 0us/step
```

Preparing the data

Encoding the integer sequences via multi-hot encoding

```
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]
```

```
array([0., 1., 1., ..., 0., 0., 0.])
```

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

Building your model

Model definition

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

Compiling the model

Adam is used as the optimizer, and binary crossentropy is used as the loss function.

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

Validating your approach

Setting aside a validation set

```
x_val = x_train[:10000]
partial_x_train = x_train[10000:]
y_val = y_train[:10000]
partial_y_train = y_train[10000:]
```

Training your model

we're training our model with 20 epochs and 512 batches.

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

```
Epoch 1/20
30/30 [==============================] - 4s 90ms/step - loss: 0.5607 - accuracy: 0.7707 - val_loss: 0.4155 - val_accuracy: 0.8602
Epoch 2/20
30/30 [==============================] - 1s 41ms/step - loss: 0.3134 - accuracy: 0.9019 - val_loss: 0.3038 - val_accuracy: 0.8886
Epoch 3/20
30/30 [==============================] - 1s 49ms/step - loss: 0.2096 - accuracy: 0.9337 - val_loss: 0.2811 - val_accuracy: 0.8878
Epoch 4/20
30/30 [==============================] - 2s 58ms/step - loss: 0.1543 - accuracy: 0.9533 - val_loss: 0.2805 - val_accuracy: 0.8883
Epoch 5/20
30/30 [==============================] - 1s 40ms/step - loss: 0.1165 - accuracy: 0.9685 - val_loss: 0.2963 - val_accuracy: 0.8834
Epoch 6/20
30/30 [==============================] - 1s 33ms/step - loss: 0.0893 - accuracy: 0.9771 - val_loss: 0.3157 - val_accuracy: 0.8820
Epoch 7/20
30/30 [==============================] - 2s 51ms/step - loss: 0.0683 - accuracy: 0.9857 - val_loss: 0.3422 - val_accuracy: 0.8794
Epoch 8/20
30/30 [==============================] - 1s 42ms/step - loss: 0.0524 - accuracy: 0.9915 - val_loss: 0.3694 - val_accuracy: 0.8775
Epoch 9/20
30/30 [==============================] - 1s 35ms/step - loss: 0.0396 - accuracy: 0.9939 - val_loss: 0.4013 - val_accuracy: 0.8758
Epoch 10/20
30/30 [==============================] - 1s 34ms/step - loss: 0.0294 - accuracy: 0.9967 - val_loss: 0.4311 - val_accuracy: 0.8758
Epoch 11/20
30/30 [==============================] - 1s 33ms/step - loss: 0.0224 - accuracy: 0.9987 - val_loss: 0.4591 - val_accuracy: 0.8733
Epoch 12/20
30/30 [==============================] - 1s 32ms/step - loss: 0.0171 - accuracy: 0.9992 - val_loss: 0.4898 - val_accuracy: 0.8730
Epoch 13/20
30/30 [==============================] - 1s 36ms/step - loss: 0.0134 - accuracy: 0.9995 - val_loss: 0.5123 - val_accuracy: 0.8714
```

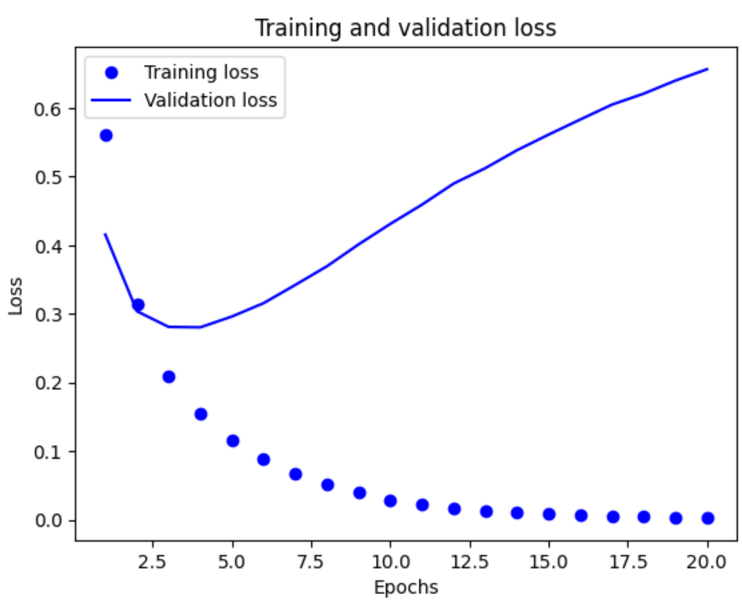
```
Epoch 14/20
30/30 [==============================] - 1s 49ms/step - loss: 0.0104 - accuracy: 0.9998 - val_loss: 0.5384 - val_accuracy: 0.8705
Epoch 15/20
30/30 [==============================] - 2s 58ms/step - loss: 0.0082 - accuracy: 0.9999 - val_loss: 0.5612 - val_accuracy: 0.8697
Epoch 16/20
30/30 [==============================] - 1s 38ms/step - loss: 0.0067 - accuracy: 0.9999 - val_loss: 0.5832 - val_accuracy: 0.8692
Epoch 17/20
30/30 [==============================] - 1s 41ms/step - loss: 0.0055 - accuracy: 0.9999 - val_loss: 0.6048 - val_accuracy: 0.8688
Epoch 18/20
30/30 [==============================] - 1s 34ms/step - loss: 0.0047 - accuracy: 0.9999 - val_loss: 0.6208 - val_accuracy: 0.8668
Epoch 19/20
30/30 [==============================] - 1s 35ms/step - loss: 0.0040 - accuracy: 1.0000 - val_loss: 0.6398 - val_accuracy: 0.8666
Epoch 20/20
30/30 [==============================] - 1s 32ms/step - loss: 0.0035 - accuracy: 1.0000 - val_loss: 0.6563 - val_accuracy: 0.8663
```

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

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

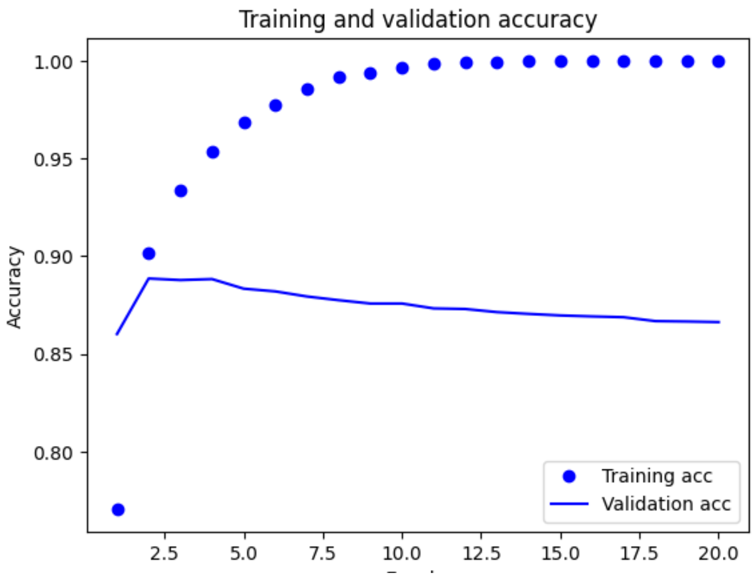
Plotting the training and validation loss

```
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")
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()
```



Plotting the training and validation accuracy

```
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()
```



Validation accuracy starts to decline around the third epoch.

Retraining a model from scratch

```
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)
```

```
Epoch 1/4
49/49 [==============================] - 2s 33ms/step - loss: 0.4773 - accuracy: 0.8111
Epoch 2/4
49/49 [==============================] - 2s 36ms/step - loss: 0.2437 - accuracy: 0.9132
Epoch 3/4
49/49 [==============================] - 1s 27ms/step - loss: 0.1790 - accuracy: 0.9363
Epoch 4/4
49/49 [==============================] - 1s 28ms/step - loss: 0.1421 - accuracy: 0.9517
782/782 [==============================] - 2s 2ms/step - loss: 0.3079 - accuracy: 0.8802
```

```
results
```

```
[0.3079053461551666, 0.8801599740982056]
```

Building your model

1 using one or three hidden layers, and see how doing so

affects validation and test accuracy.

```
#I am creating a model with just 1 hidden layer and the ReLu activation function.
model1_1 = keras.Sequential([
    layers.Dense(16, activation="relu"),
    layers.Dense(1, activation="sigmoid")
])
```

```
# I am using three hidden layers here, with ReLu activation function and sigmoid for output layer.
model1_3 = keras.Sequential([
    layers.Dense(16, activation="relu"),
    layers.Dense(16, activation="relu"),
    layers.Dense(16, activation="relu"),
```

```
    layers.Dense(1, activation="sigmoid")
])


#Adam and binary crossentropy are used in both scenarios (3 and 1 layers)
model1_1.compile(optimizer="adam",
              loss="binary_crossentropy",
              metrics=["accuracy"])


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



# model fitting with 20 epochs and 512 batch size
history1_1 = model1_1.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val, y_val))
```

```
Epoch 1/20
30/30 [==============================] - 4s 109ms/step - loss: 0.5741 - accuracy: 0.7357 - val_loss: 0.4392 - val_accuracy: 0.8503
Epoch 2/20
30/30 [==============================] - 1s 43ms/step - loss: 0.3491 - accuracy: 0.8931 - val_loss: 0.3345 - val_accuracy: 0.8821
Epoch 3/20
30/30 [==============================] - 1s 36ms/step - loss: 0.2585 - accuracy: 0.9230 - val_loss: 0.3004 - val_accuracy: 0.8854
Epoch 4/20
30/30 [==============================] - 1s 33ms/step - loss: 0.2099 - accuracy: 0.9386 - val_loss: 0.2833 - val_accuracy: 0.8896
Epoch 5/20
30/30 [==============================] - 1s 38ms/step - loss: 0.1765 - accuracy: 0.9501 - val_loss: 0.2800 - val_accuracy: 0.8857
Epoch 6/20
30/30 [==============================] - 1s 35ms/step - loss: 0.1518 - accuracy: 0.9581 - val_loss: 0.2789 - val_accuracy: 0.8872
Epoch 7/20
30/30 [==============================] - 2s 52ms/step - loss: 0.1313 - accuracy: 0.9659 - val_loss: 0.2807 - val_accuracy: 0.8871
Epoch 8/20
30/30 [==============================] - 1s 42ms/step - loss: 0.1134 - accuracy: 0.9729 - val_loss: 0.2854 - val_accuracy: 0.8854
Epoch 9/20
30/30 [==============================] - 2s 55ms/step - loss: 0.0984 - accuracy: 0.9782 - val_loss: 0.2924 - val_accuracy: 0.8840
Epoch 10/20
30/30 [==============================] - 1s 47ms/step - loss: 0.0859 - accuracy: 0.9833 - val_loss: 0.3012 - val_accuracy: 0.8823
Epoch 11/20
30/30 [==============================] - 1s 35ms/step - loss: 0.0750 - accuracy: 0.9870 - val_loss: 0.3114 - val_accuracy: 0.8806
Epoch 12/20
30/30 [==============================] - 1s 33ms/step - loss: 0.0662 - accuracy: 0.9897 - val_loss: 0.3213 - val_accuracy: 0.8823
Epoch 13/20
30/30 [==============================] - 1s 30ms/step - loss: 0.0580 - accuracy: 0.9919 - val_loss: 0.3320 - val_accuracy: 0.8814
Epoch 14/20
30/30 [==============================] - 1s 35ms/step - loss: 0.0513 - accuracy: 0.9934 - val_loss: 0.3418 - val_accuracy: 0.8798
Epoch 15/20
30/30 [==============================] - 1s 43ms/step - loss: 0.0456 - accuracy: 0.9949 - val_loss: 0.3530 - val_accuracy: 0.8788
Epoch 16/20
30/30 [==============================] - 1s 42ms/step - loss: 0.0407 - accuracy: 0.9961 - val_loss: 0.3632 - val_accuracy: 0.8775
Epoch 17/20
30/30 [==============================] - 1s 42ms/step - loss: 0.0364 - accuracy: 0.9969 - val_loss: 0.3741 - val_accuracy: 0.8771
Epoch 18/20
30/30 [==============================] - 1s 36ms/step - loss: 0.0328 - accuracy: 0.9974 - val_loss: 0.3853 - val_accuracy: 0.8766
Epoch 19/20
30/30 [==============================] - 1s 32ms/step - loss: 0.0293 - accuracy: 0.9986 - val_loss: 0.3946 - val_accuracy: 0.8747
Epoch 20/20
30/30 [==============================] - 1s 47ms/step - loss: 0.0264 - accuracy: 0.9991 - val_loss: 0.4056 - val_accuracy: 0.8753
```

```
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 [==============================] - 6s 158ms/step - loss: 0.5474 - accuracy: 0.7514 - val_loss: 0.3953 - val_accuracy: 0.8553
Epoch 2/20
30/30 [==============================] - 2s 58ms/step - loss: 0.2852 - accuracy: 0.9023 - val_loss: 0.2877 - val_accuracy: 0.8885
Epoch 3/20
30/30 [==============================] - 2s 55ms/step - loss: 0.1842 - accuracy: 0.9375 - val_loss: 0.2795 - val_accuracy: 0.8907
Epoch 4/20
30/30 [==============================] - 1s 31ms/step - loss: 0.1300 - accuracy: 0.9589 - val_loss: 0.2990 - val_accuracy: 0.8853
Epoch 5/20
30/30 [==============================] - 1s 31ms/step - loss: 0.0928 - accuracy: 0.9728 - val_loss: 0.3279 - val_accuracy: 0.8825
Epoch 6/20
```

```
30/30 [==============================] - 1s 32ms/step - loss: 0.0648 - accuracy: 0.9839 - val_loss: 0.3715 - val_accuracy: 0.8806
Epoch 7/20
30/30 [==============================] - 1s 33ms/step - loss: 0.0455 - accuracy: 0.9896 - val_loss: 0.4202 - val_accuracy: 0.8763
Epoch 8/20
30/30 [==============================] - 2s 51ms/step - loss: 0.0313 - accuracy: 0.9945 - val_loss: 0.4667 - val_accuracy: 0.8740
Epoch 9/20
30/30 [==============================] - 1s 41ms/step - loss: 0.0202 - accuracy: 0.9980 - val_loss: 0.5151 - val_accuracy: 0.8716
Epoch 10/20
30/30 [==============================] - 2s 53ms/step - loss: 0.0135 - accuracy: 0.9993 - val_loss: 0.5579 - val_accuracy: 0.8698
Epoch 11/20
30/30 [==============================] - 1s 40ms/step - loss: 0.0092 - accuracy: 0.9997 - val_loss: 0.5984 - val_accuracy: 0.8676
Epoch 12/20
30/30 [==============================] - 1s 51ms/step - loss: 0.0066 - accuracy: 0.9999 - val_loss: 0.6347 - val_accuracy: 0.8682
Epoch 13/20
30/30 [==============================] - 2s 65ms/step - loss: 0.0050 - accuracy: 0.9999 - val_loss: 0.6665 - val_accuracy: 0.8678
Epoch 14/20
30/30 [==============================] - 1s 37ms/step - loss: 0.0039 - accuracy: 0.9999 - val_loss: 0.6968 - val_accuracy: 0.8663
Epoch 15/20
30/30 [==============================] - 1s 51ms/step - loss: 0.0031 - accuracy: 0.9999 - val_loss: 0.7203 - val_accuracy: 0.8676
Epoch 16/20
30/30 [==============================] - 1s 40ms/step - loss: 0.0026 - accuracy: 0.9999 - val_loss: 0.7444 - val_accuracy: 0.8673
Epoch 17/20
30/30 [==============================] - 1s 33ms/step - loss: 0.0022 - accuracy: 0.9999 - val_loss: 0.7664 - val_accuracy: 0.8659
Epoch 18/20
30/30 [==============================] - 1s 32ms/step - loss: 0.0018 - accuracy: 0.9999 - val_loss: 0.7875 - val_accuracy: 0.8655
Epoch 19/20
30/30 [==============================] - 1s 35ms/step - loss: 0.0016 - accuracy: 0.9999 - val_loss: 0.8074 - val_accuracy: 0.8649
Epoch 20/20
30/30 [==============================] - 1s 41ms/step - loss: 0.0014 - accuracy: 0.9999 - val_loss: 0.8239 - val_accuracy: 0.8649
```

plotting training vs validation loss

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

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

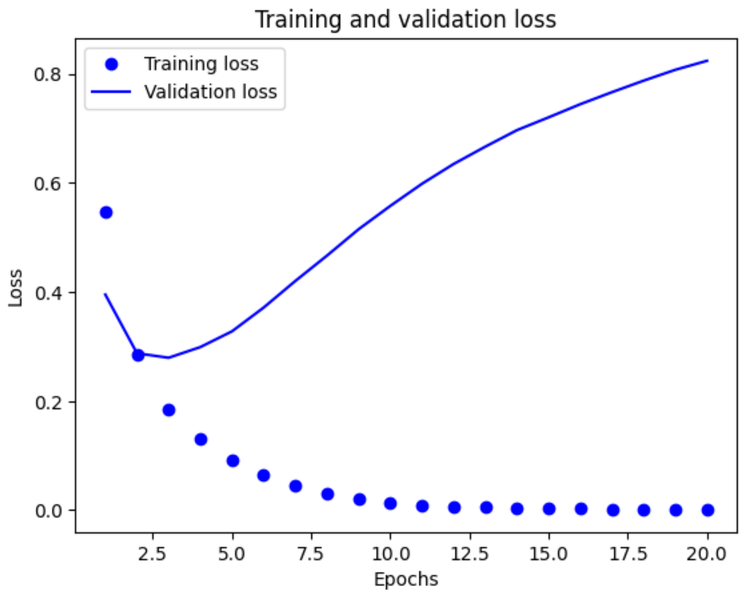
```
historyp1_3 = history1_1.history
historyp1_3.keys()
```

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

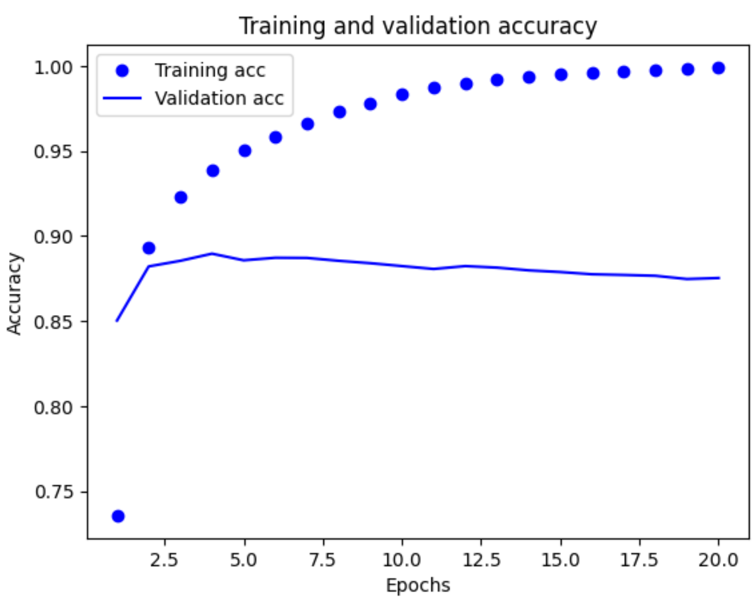
```
historyp1_1 = history1_1.history
loss_values1 = historyp1_1["loss"]
val_loss_values1 = historyp1_1["val_loss"]
epochs = range(1, len(loss_values) + 1)
plt.plot(epochs, loss_values1, "bo", label="Training loss")
plt.plot(epochs, val_loss_values1, "b", label="Validation loss")
plt.title("Training and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

Training and validation loss

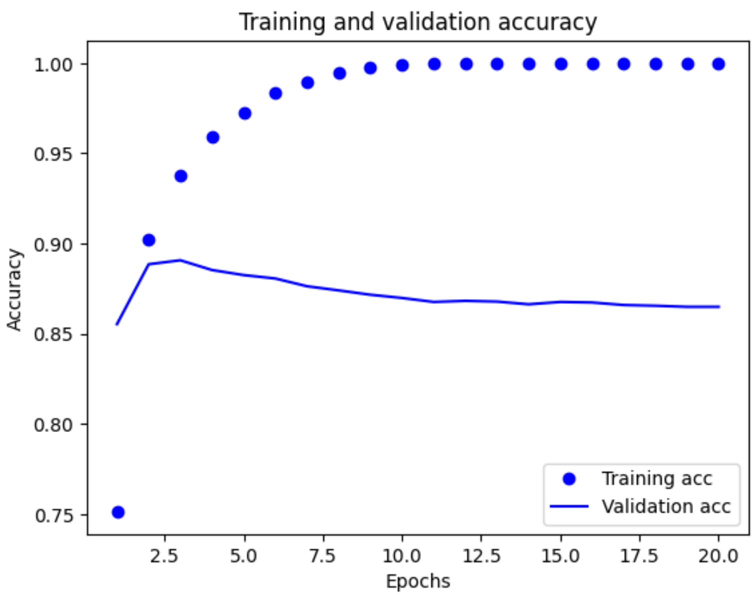
```
historyp1_3 = history1_3.history
loss_values3 = historyp1_3["loss"]
val_loss_values3 = historyp1_3["val_loss"]
epochs = range(1, len(loss_values) + 1)
plt.plot(epochs, loss_values3, "bo", label="Training loss")
plt.plot(epochs, val_loss_values3, "b", label="Validation loss")
plt.title("Training and 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()
```



2 For the hidden layers we are using nodes 32 units, 64 units

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

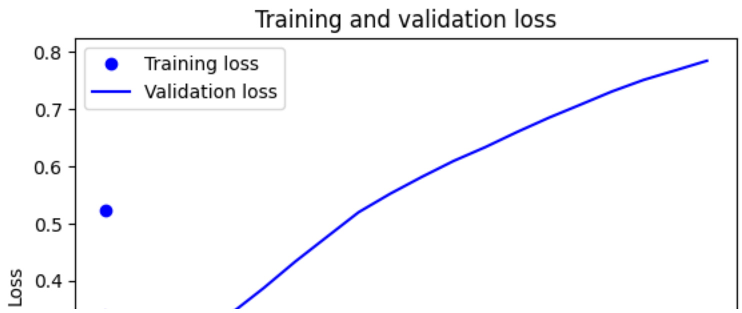
```
model2.compile(optimizer="adam",
              loss="binary_crossentropy",
              metrics=["accuracy"])
```

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

```
Epoch 1/20
30/30 [==============================] - 3s 74ms/step - loss: 0.5223 - accuracy: 0.7798 - val_loss: 0.3463 - val_accuracy: 0.8686
Epoch 2/20
30/30 [==============================] - 1s 40ms/step - loss: 0.2485 - accuracy: 0.9105 - val_loss: 0.2777 - val_accuracy: 0.8898
Epoch 3/20
30/30 [==============================] - 1s 39ms/step - loss: 0.1621 - accuracy: 0.9453 - val_loss: 0.2841 - val_accuracy: 0.8873
Epoch 4/20
30/30 [==============================] - 2s 55ms/step - loss: 0.1130 - accuracy: 0.9649 - val_loss: 0.3099 - val_accuracy: 0.8815
Epoch 5/20
30/30 [==============================] - 2s 52ms/step - loss: 0.0786 - accuracy: 0.9786 - val_loss: 0.3436 - val_accuracy: 0.8809
Epoch 6/20
30/30 [==============================] - 1s 42ms/step - loss: 0.0530 - accuracy: 0.9888 - val_loss: 0.3869 - val_accuracy: 0.8760
Epoch 7/20
30/30 [==============================] - 1s 40ms/step - loss: 0.0341 - accuracy: 0.9945 - val_loss: 0.4333 - val_accuracy: 0.8727
Epoch 8/20
30/30 [==============================] - 2s 61ms/step - loss: 0.0217 - accuracy: 0.9975 - val_loss: 0.4765 - val_accuracy: 0.8713
Epoch 9/20
30/30 [==============================] - 2s 51ms/step - loss: 0.0129 - accuracy: 0.9997 - val_loss: 0.5196 - val_accuracy: 0.8677
Epoch 10/20
30/30 [==============================] - 1s 38ms/step - loss: 0.0080 - accuracy: 0.9998 - val_loss: 0.5520 - val_accuracy: 0.8689
Epoch 11/20
30/30 [==============================] - 1s 38ms/step - loss: 0.0054 - accuracy: 0.9999 - val_loss: 0.5815 - val_accuracy: 0.8679
```

```
Epoch 12/20
30/30 [==============================] - 1s 46ms/step - loss: 0.0038 - accuracy: 0.9999 - val_loss: 0.6093 - val_accuracy: 0.8682
Epoch 13/20
30/30 [==============================] - 1s 41ms/step - loss: 0.0027 - accuracy: 0.9999 - val_loss: 0.6334 - val_accuracy: 0.8674
Epoch 14/20
30/30 [==============================] - 1s 41ms/step - loss: 0.0020 - accuracy: 1.0000 - val_loss: 0.6598 - val_accuracy: 0.8661
Epoch 15/20
30/30 [==============================] - 2s 58ms/step - loss: 0.0014 - accuracy: 1.0000 - val_loss: 0.6847 - val_accuracy: 0.8670
Epoch 16/20
30/30 [==============================] - 1s 40ms/step - loss: 0.0011 - accuracy: 1.0000 - val_loss: 0.7076 - val_accuracy: 0.8660
Epoch 17/20
30/30 [==============================] - 2s 55ms/step - loss: 8.0906e-04 - accuracy: 1.0000 - val_loss: 0.7308 - val_accuracy: 0.8663
Epoch 18/20
30/30 [==============================] - 2s 75ms/step - loss: 6.2698e-04 - accuracy: 1.0000 - val_loss: 0.7510 - val_accuracy: 0.8660
Epoch 19/20
30/30 [==============================] - 2s 51ms/step - loss: 4.9709e-04 - accuracy: 1.0000 - val_loss: 0.7677 - val_accuracy: 0.8662
Epoch 20/20
30/30 [==============================] - 2s 57ms/step - loss: 4.0365e-04 - accuracy: 1.0000 - val_loss: 0.7845 - val_accuracy: 0.8651
```

```python
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()
```



3 using the mse loss function instead of binary_crossentropy.

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

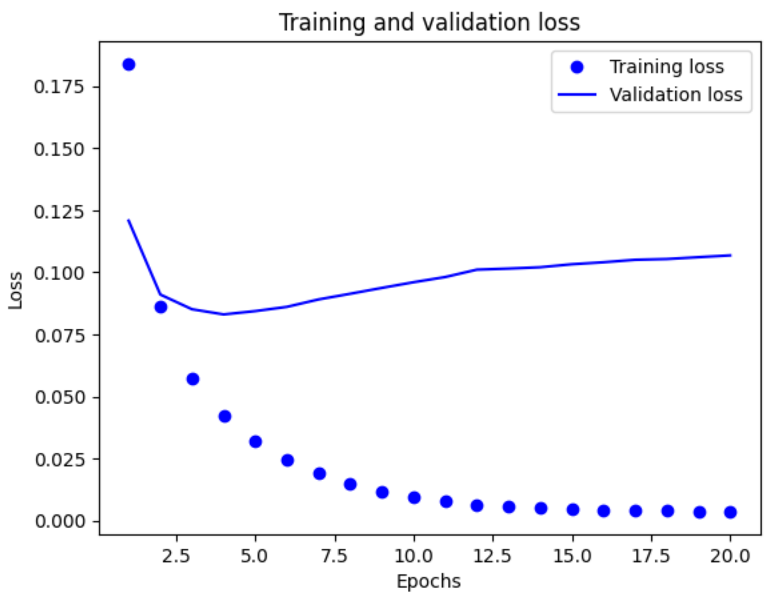
```
#So Here, I used the MSE loss function instead of the binary cross entropy that he has previously used.
model3.compile(optimizer="adam",
              loss="mse",
              metrics=["accuracy"])
```

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

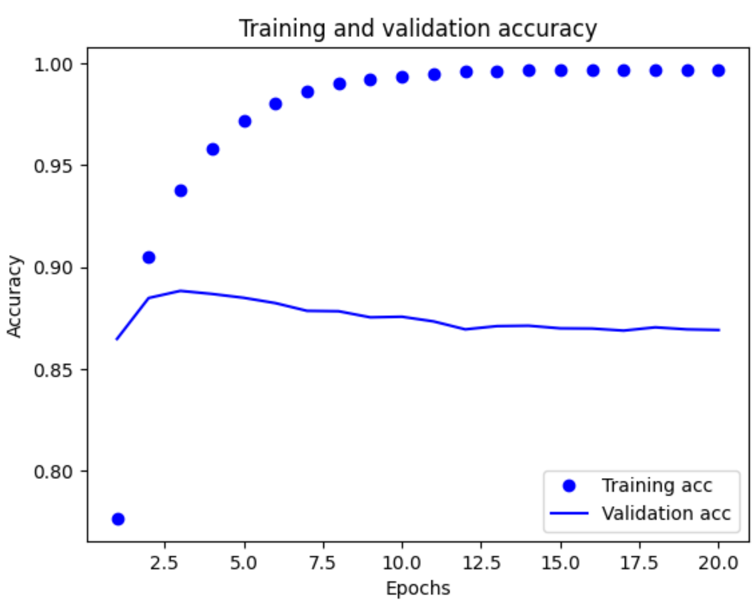
```
Epoch 1/20
30/30 [==============================] - 3s 78ms/step - loss: 0.1840 - accuracy: 0.7766 - val_loss: 0.1208 - val_accuracy: 0.8647
Epoch 2/20
30/30 [==============================] - 1s 50ms/step - loss: 0.0861 - accuracy: 0.9047 - val_loss: 0.0911 - val_accuracy: 0.8848
Epoch 3/20
30/30 [==============================] - 2s 56ms/step - loss: 0.0573 - accuracy: 0.9379 - val_loss: 0.0851 - val_accuracy: 0.8883
Epoch 4/20
30/30 [==============================] - 2s 63ms/step - loss: 0.0424 - accuracy: 0.9579 - val_loss: 0.0830 - val_accuracy: 0.8868
Epoch 5/20
30/30 [==============================] - 1s 41ms/step - loss: 0.0319 - accuracy: 0.9716 - val_loss: 0.0844 - val_accuracy: 0.8849
Epoch 6/20
30/30 [==============================] - 1s 34ms/step - loss: 0.0246 - accuracy: 0.9803 - val_loss: 0.0861 - val_accuracy: 0.8823
Epoch 7/20
30/30 [==============================] - 1s 32ms/step - loss: 0.0190 - accuracy: 0.9861 - val_loss: 0.0891 - val_accuracy: 0.8785
Epoch 8/20
30/30 [==============================] - 1s 31ms/step - loss: 0.0149 - accuracy: 0.9901 - val_loss: 0.0914 - val_accuracy: 0.8783
Epoch 9/20
30/30 [==============================] - 1s 37ms/step - loss: 0.0118 - accuracy: 0.9922 - val_loss: 0.0937 - val_accuracy: 0.8753
Epoch 10/20
30/30 [==============================] - 1s 36ms/step - loss: 0.0095 - accuracy: 0.9935 - val_loss: 0.0960 - val_accuracy: 0.8756
Epoch 11/20
30/30 [==============================] - 1s 32ms/step - loss: 0.0079 - accuracy: 0.9948 - val_loss: 0.0981 - val_accuracy: 0.8733
Epoch 12/20
30/30 [==============================] - 1s 31ms/step - loss: 0.0065 - accuracy: 0.9958 - val_loss: 0.1011 - val_accuracy: 0.8694
Epoch 13/20
30/30 [==============================] - 1s 38ms/step - loss: 0.0056 - accuracy: 0.9963 - val_loss: 0.1015 - val_accuracy: 0.8710
Epoch 14/20
30/30 [==============================] - 2s 57ms/step - loss: 0.0050 - accuracy: 0.9965 - val_loss: 0.1020 - val_accuracy: 0.8712
Epoch 15/20
30/30 [==============================] - 2s 63ms/step - loss: 0.0046 - accuracy: 0.9965 - val_loss: 0.1033 - val_accuracy: 0.8699
Epoch 16/20
30/30 [==============================] - 1s 34ms/step - loss: 0.0043 - accuracy: 0.9966 - val_loss: 0.1041 - val_accuracy: 0.8698
Epoch 17/20
30/30 [==============================] - 1s 41ms/step - loss: 0.0041 - accuracy: 0.9966 - val_loss: 0.1050 - val_accuracy: 0.8688
Epoch 18/20
30/30 [==============================] - 1s 38ms/step - loss: 0.0040 - accuracy: 0.9966 - val_loss: 0.1054 - val_accuracy: 0.8704
Epoch 19/20
30/30 [==============================] - 1s 31ms/step - loss: 0.0038 - accuracy: 0.9967 - val_loss: 0.1061 - val_accuracy: 0.8694
Epoch 20/20
30/30 [==============================] - 1s 31ms/step - loss: 0.0036 - accuracy: 0.9968 - val_loss: 0.1068 - val_accuracy: 0.8691
```

```
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()
```

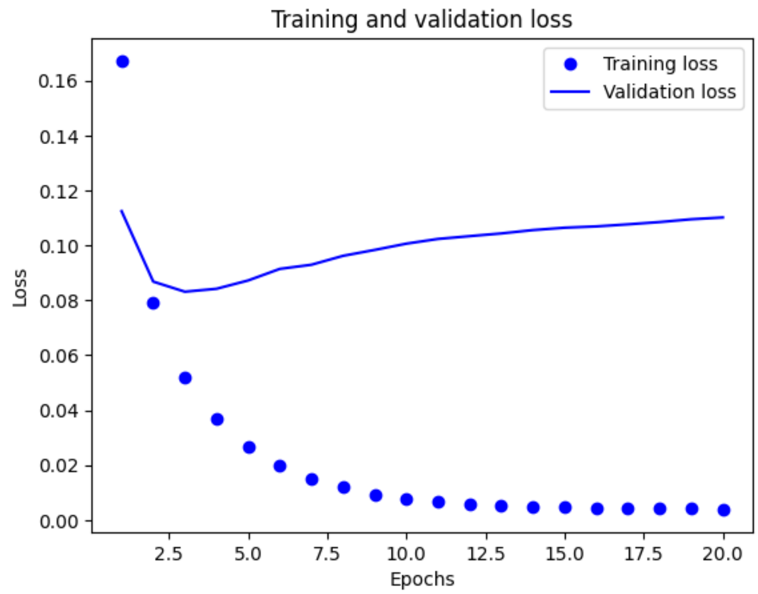


4 I am using tanh activation instead of relu.

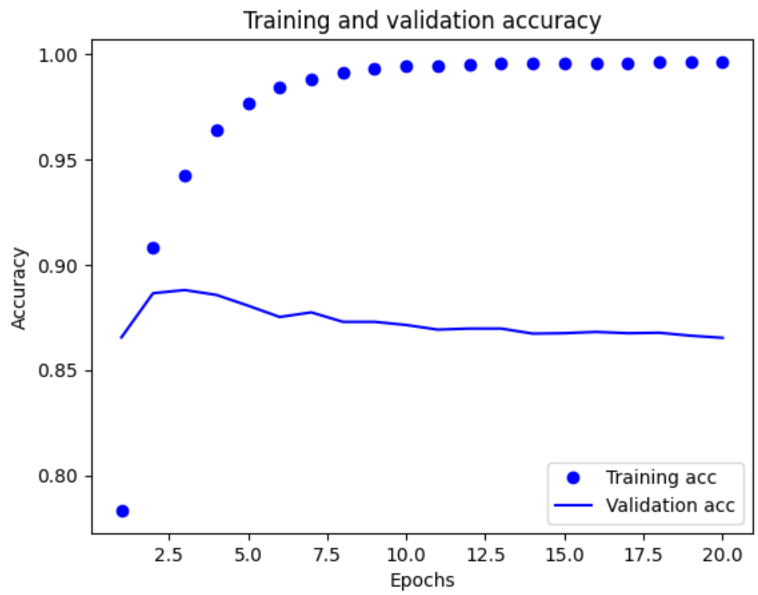
```python
model4 = keras.Sequential([
    layers.Dense(16, activation="tanh"),
    layers.Dense(16, activation="tanh"),
    layers.Dense(1, activation="sigmoid")
])


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))
```

```
Epoch 1/20
30/30 [==============================] - 3s 81ms/step - loss: 0.1672 - accuracy: 0.7833 - val_loss: 0.1124 - val_accuracy: 0.8655
Epoch 2/20
30/30 [==============================] - 2s 56ms/step - loss: 0.0792 - accuracy: 0.9085 - val_loss: 0.0868 - val_accuracy: 0.8865
Epoch 3/20
30/30 [==============================] - 1s 41ms/step - loss: 0.0520 - accuracy: 0.9423 - val_loss: 0.0831 - val_accuracy: 0.8880
Epoch 4/20
30/30 [==============================] - 1s 35ms/step - loss: 0.0368 - accuracy: 0.9639 - val_loss: 0.0842 - val_accuracy: 0.8857
Epoch 5/20
30/30 [==============================] - 1s 39ms/step - loss: 0.0267 - accuracy: 0.9765 - val_loss: 0.0872 - val_accuracy: 0.8806
Epoch 6/20
30/30 [==============================] - 1s 42ms/step - loss: 0.0198 - accuracy: 0.9841 - val_loss: 0.0914 - val_accuracy: 0.8752
Epoch 7/20
30/30 [==============================] - 1s 35ms/step - loss: 0.0152 - accuracy: 0.9885 - val_loss: 0.0929 - val_accuracy: 0.8774
Epoch 8/20
30/30 [==============================] - 1s 32ms/step - loss: 0.0119 - accuracy: 0.9915 - val_loss: 0.0962 - val_accuracy: 0.8729
Epoch 9/20
30/30 [==============================] - 1s 43ms/step - loss: 0.0092 - accuracy: 0.9934 - val_loss: 0.0983 - val_accuracy: 0.8729
Epoch 10/20
30/30 [==============================] - 2s 53ms/step - loss: 0.0076 - accuracy: 0.9944 - val_loss: 0.1006 - val_accuracy: 0.8714
Epoch 11/20
30/30 [==============================] - 2s 62ms/step - loss: 0.0066 - accuracy: 0.9948 - val_loss: 0.1023 - val_accuracy: 0.8692
Epoch 12/20
30/30 [==============================] - 1s 46ms/step - loss: 0.0058 - accuracy: 0.9954 - val_loss: 0.1033 - val_accuracy: 0.8697
Epoch 13/20
30/30 [==============================] - 2s 53ms/step - loss: 0.0053 - accuracy: 0.9957 - val_loss: 0.1043 - val_accuracy: 0.8697
Epoch 14/20
30/30 [==============================] - 1s 37ms/step - loss: 0.0049 - accuracy: 0.9959 - val_loss: 0.1056 - val_accuracy: 0.8673
Epoch 15/20
30/30 [==============================] - 1s 32ms/step - loss: 0.0047 - accuracy: 0.9959 - val_loss: 0.1064 - val_accuracy: 0.8675
Epoch 16/20
30/30 [==============================] - 1s 38ms/step - loss: 0.0045 - accuracy: 0.9960 - val_loss: 0.1069 - val_accuracy: 0.8681
Epoch 17/20
30/30 [==============================] - 1s 37ms/step - loss: 0.0044 - accuracy: 0.9960 - val_loss: 0.1076 - val_accuracy: 0.8675
Epoch 18/20
30/30 [==============================] - 1s 34ms/step - loss: 0.0042 - accuracy: 0.9961 - val_loss: 0.1085 - val_accuracy: 0.8677
Epoch 19/20
30/30 [==============================] - 1s 32ms/step - loss: 0.0041 - accuracy: 0.9962 - val_loss: 0.1095 - val_accuracy: 0.8663
Epoch 20/20
30/30 [==============================] - 1s 39ms/step - loss: 0.0039 - accuracy: 0.9965 - val_loss: 0.1101 - val_accuracy: 0.8653
```

```python
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()
```




```
<matplotlib.legend.Legend at 0x7eae20aac910>
```



5 In our network I am using Dropout Technique.

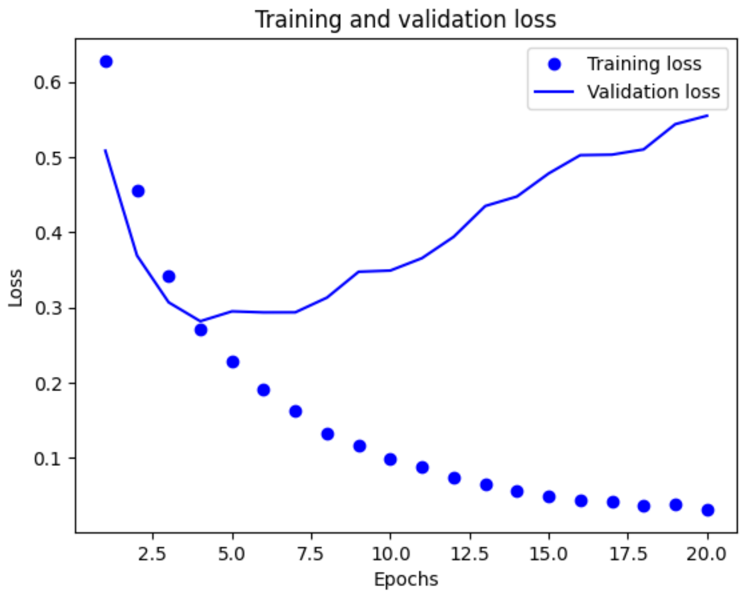
Double-click (or enter) to edit

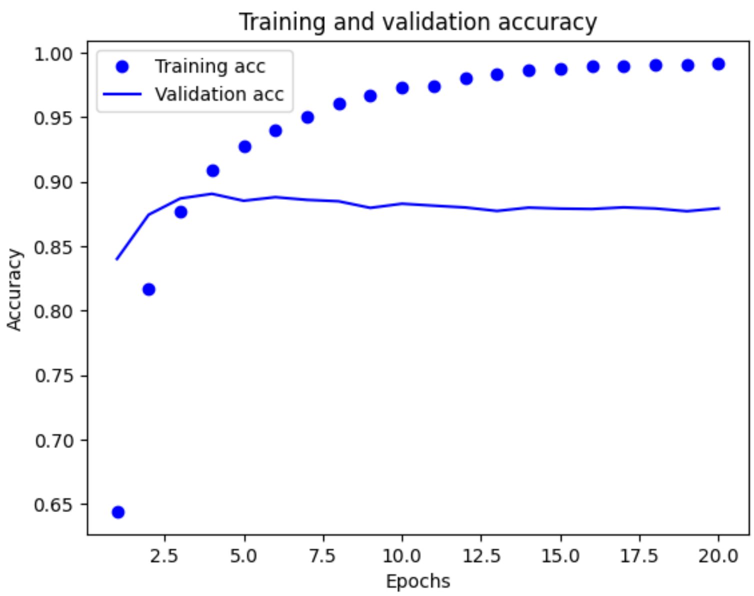
```
#I am using the dropout method with two hidden layers that have the ReLu activation function.
from tensorflow import keras
from tensorflow.keras import layers
model5 = keras.Sequential([
    layers.Dense(16, activation="relu"),
    layers.Dropout(0.5),
    layers.Dense(16, activation="relu"),
    layers.Dense(1, activation="sigmoid")
])


model5.compile(optimizer="adam",
              loss="binary_crossentropy",
              metrics=["accuracy"])


hist5 = model5.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
```

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

```
Epoch 1/20
30/30 [==============================] - 4s 99ms/step - loss: 0.6277 - accuracy: 0.6443 - val_loss: 0.5083 - val_accuracy: 0.8402
Epoch 2/20
30/30 [==============================] - 1s 33ms/step - loss: 0.4561 - accuracy: 0.8172 - val_loss: 0.3690 - val_accuracy: 0.8745
Epoch 3/20
30/30 [==============================] - 1s 42ms/step - loss: 0.3427 - accuracy: 0.8771 - val_loss: 0.3068 - val_accuracy: 0.8871
Epoch 4/20
30/30 [==============================] - 1s 42ms/step - loss: 0.2712 - accuracy: 0.9093 - val_loss: 0.2817 - val_accuracy: 0.8907
Epoch 5/20
30/30 [==============================] - 1s 43ms/step - loss: 0.2284 - accuracy: 0.9275 - val_loss: 0.2949 - val_accuracy: 0.8853
Epoch 6/20
30/30 [==============================] - 2s 52ms/step - loss: 0.1910 - accuracy: 0.9398 - val_loss: 0.2935 - val_accuracy: 0.8881
Epoch 7/20
30/30 [==============================] - 2s 56ms/step - loss: 0.1631 - accuracy: 0.9504 - val_loss: 0.2935 - val_accuracy: 0.8860
Epoch 8/20
30/30 [==============================] - 2s 58ms/step - loss: 0.1320 - accuracy: 0.9609 - val_loss: 0.3131 - val_accuracy: 0.8849
Epoch 9/20
30/30 [==============================] - 1s 34ms/step - loss: 0.1160 - accuracy: 0.9673 - val_loss: 0.3475 - val_accuracy: 0.8798
Epoch 10/20
30/30 [==============================] - 1s 47ms/step - loss: 0.0988 - accuracy: 0.9727 - val_loss: 0.3490 - val_accuracy: 0.8830
Epoch 11/20
30/30 [==============================] - 1s 34ms/step - loss: 0.0877 - accuracy: 0.9744 - val_loss: 0.3655 - val_accuracy: 0.8815
Epoch 12/20
30/30 [==============================] - 2s 53ms/step - loss: 0.0740 - accuracy: 0.9801 - val_loss: 0.3938 - val_accuracy: 0.8801
Epoch 13/20
30/30 [==============================] - 1s 44ms/step - loss: 0.0646 - accuracy: 0.9839 - val_loss: 0.4348 - val_accuracy: 0.8774
Epoch 14/20
30/30 [==============================] - 1s 42ms/step - loss: 0.0559 - accuracy: 0.9863 - val_loss: 0.4474 - val_accuracy: 0.8800
Epoch 15/20
30/30 [==============================] - 1s 43ms/step - loss: 0.0493 - accuracy: 0.9872 - val_loss: 0.4780 - val_accuracy: 0.8792
Epoch 16/20
30/30 [==============================] - 2s 53ms/step - loss: 0.0443 - accuracy: 0.9895 - val_loss: 0.5023 - val_accuracy: 0.8789
Epoch 17/20
30/30 [==============================] - 1s 50ms/step - loss: 0.0415 - accuracy: 0.9900 - val_loss: 0.5030 - val_accuracy: 0.8802
Epoch 18/20
30/30 [==============================] - 1s 41ms/step - loss: 0.0374 - accuracy: 0.9907 - val_loss: 0.5100 - val_accuracy: 0.8793
Epoch 19/20
30/30 [==============================] - 1s 41ms/step - loss: 0.0379 - accuracy: 0.9903 - val_loss: 0.5435 - val_accuracy: 0.8772
Epoch 20/20
30/30 [==============================] - 1s 41ms/step - loss: 0.0318 - accuracy: 0.9919 - val_loss: 0.5547 - val_accuracy: 0.8794
```

```
#Creating training vs. validation graphs Training vs. validation accuracy and loss
import matplotlib.pyplot as plt
histp5 = hist5.history
loss_values = histp5["loss"]
val_loss_values = histp5["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 = histp5["accuracy"]
val_acc = histp5["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()
```


Training and validation loss
Training loss
Validation loss
Loss
Epochs
0.6
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


Training and validation accuracy
Training acc
Validation acc
Accuracy
Epochs
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0