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

[Deep Learning with Python, Second Edition] (https://www.manning.com/books/deep-learning-with-python-second-edition?a_aid=keras&a_bid=76564dff) has a companion notebook. All text paragraphs, figures, and pseudocode are removed, leaving only runnable code blocks and section headings for readability.

Categorizing movie reviews: An illustration of binary classification

The dataset from IMDB

The IMDB dataset is loading

importing a Keras-based IMDB dataset. We'll examine the 10,000 words here.

Creating training and test sets from the dataset.

```
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,
```

```
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.

"""### Getting the data ready

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)
```

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

"""### Getting the data ready

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

```
x_train[0]
```

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

```
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
```

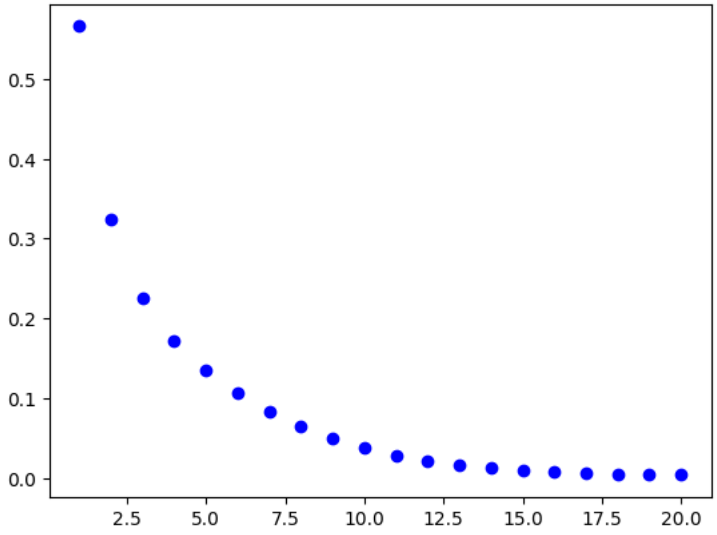
```
x_val = x_train[:10000]
partial_x_train = x_train[10000:]
y_val = y_train[:10000]
partial_y_train = y_train[10000:]
```

```
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()
```

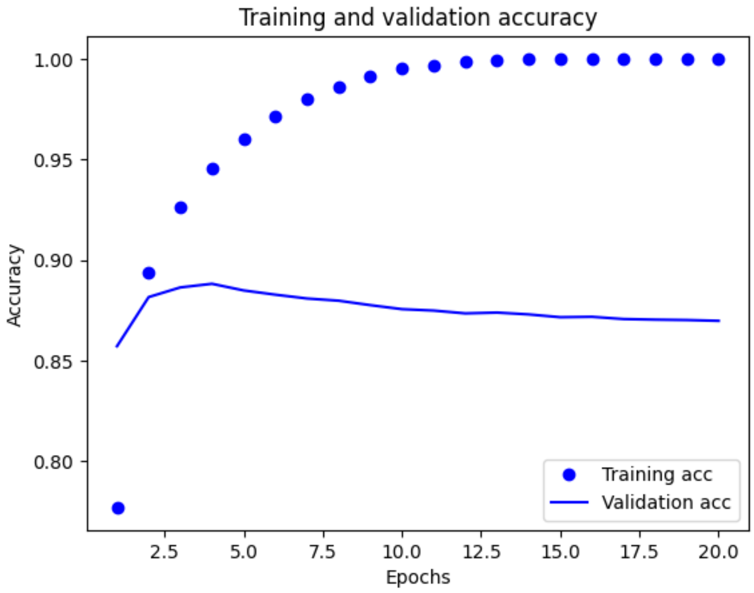
```
Epoch 1/20
WARNING:tensorflow:From c:\Users\imtha\AppData\Local\Programs\Python\Python39\lib\site-packages\keras\src\utils\tf_utils.py:492: The nam

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

30/30 [==============================] - 1s 23ms/step - loss: 0.5661 - accuracy: 0.7771 - val_loss: 0.4202 - val_accuracy: 0.8572
Epoch 2/20
30/30 [==============================] - 0s 9ms/step - loss: 0.3238 - accuracy: 0.8937 - val_loss: 0.3111 - val_accuracy: 0.8816
Epoch 3/20
30/30 [==============================] - 0s 9ms/step - loss: 0.2248 - accuracy: 0.9260 - val_loss: 0.2834 - val_accuracy: 0.8864
Epoch 4/20
30/30 [==============================] - 0s 8ms/step - loss: 0.1715 - accuracy: 0.9455 - val_loss: 0.2767 - val_accuracy: 0.8882
Epoch 5/20
30/30 [==============================] - 0s 8ms/step - loss: 0.1346 - accuracy: 0.9599 - val_loss: 0.2848 - val_accuracy: 0.8849
Epoch 6/20
30/30 [==============================] - 0s 8ms/step - loss: 0.1060 - accuracy: 0.9713 - val_loss: 0.3005 - val_accuracy: 0.8828
Epoch 7/20
30/30 [==============================] - 0s 8ms/step - loss: 0.0834 - accuracy: 0.9802 - val_loss: 0.3204 - val_accuracy: 0.8809
Epoch 8/20
30/30 [==============================] - 0s 8ms/step - loss: 0.0649 - accuracy: 0.9857 - val_loss: 0.3453 - val_accuracy: 0.8798
Epoch 9/20
30/30 [==============================] - 0s 8ms/step - loss: 0.0497 - accuracy: 0.9914 - val_loss: 0.3727 - val_accuracy: 0.8776
Epoch 10/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0378 - accuracy: 0.9950 - val_loss: 0.3998 - val_accuracy: 0.8756
Epoch 11/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0282 - accuracy: 0.9969 - val_loss: 0.4287 - val_accuracy: 0.8749
Epoch 12/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0209 - accuracy: 0.9985 - val_loss: 0.4591 - val_accuracy: 0.8735
Epoch 13/20
30/30 [==============================] - 0s 8ms/step - loss: 0.0158 - accuracy: 0.9994 - val_loss: 0.4824 - val_accuracy: 0.8739
Epoch 14/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0122 - accuracy: 0.9997 - val_loss: 0.5059 - val_accuracy: 0.8730
Epoch 15/20
30/30 [==============================] - 0s 8ms/step - loss: 0.0095 - accuracy: 0.9998 - val_loss: 0.5293 - val_accuracy: 0.8716
Epoch 16/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0076 - accuracy: 0.9999 - val_loss: 0.5504 - val_accuracy: 0.8718
Epoch 17/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0062 - accuracy: 0.9999 - val_loss: 0.5690 - val_accuracy: 0.8707
Epoch 18/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0052 - accuracy: 0.9999 - val_loss: 0.5865 - val_accuracy: 0.8704
Epoch 19/20
30/30 [==============================] - 0s 8ms/step - loss: 0.0044 - accuracy: 0.9999 - val_loss: 0.6028 - val_accuracy: 0.8702
Epoch 20/20
30/30 [==============================] - 0s 8ms/step - loss: 0.0038 - accuracy: 0.9999 - val_loss: 0.6186 - val_accuracy: 0.8698
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")
```

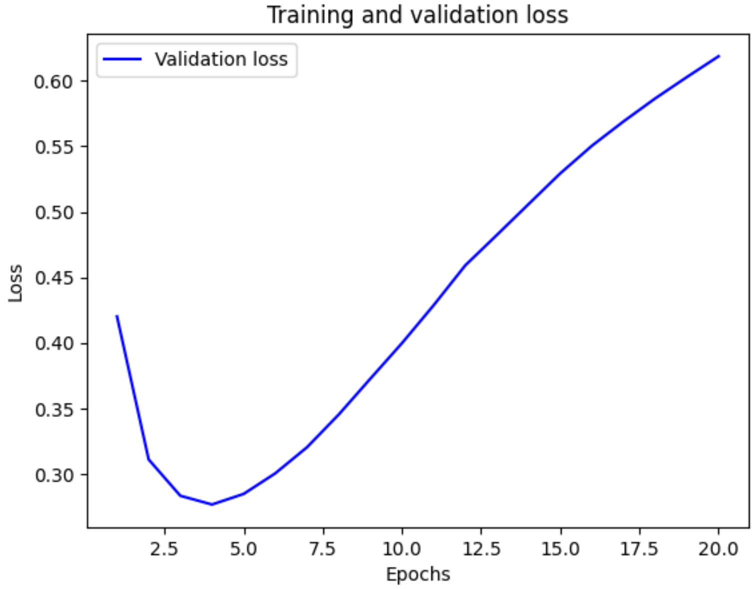
```
[<matplotlib.lines.Line2D at 0x1d71db6e6d0>]
```



```
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: 0.4762 - accuracy: 0.7980
Epoch 2/4
49/49 [==============================] - 0s 6ms/step - loss: 0.2418 - accuracy: 0.9144
Epoch 3/4
49/49 [==============================] - 0s 6ms/step - loss: 0.1746 - accuracy: 0.9395
Epoch 4/4
49/49 [==============================] - 0s 6ms/step - loss: 0.1380 - accuracy: 0.9552
782/782 [==============================] - 1s 1ms/step - loss: 0.3192 - accuracy: 0.8777
[0.3192020356655121, 0.8777199983596802]
```

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

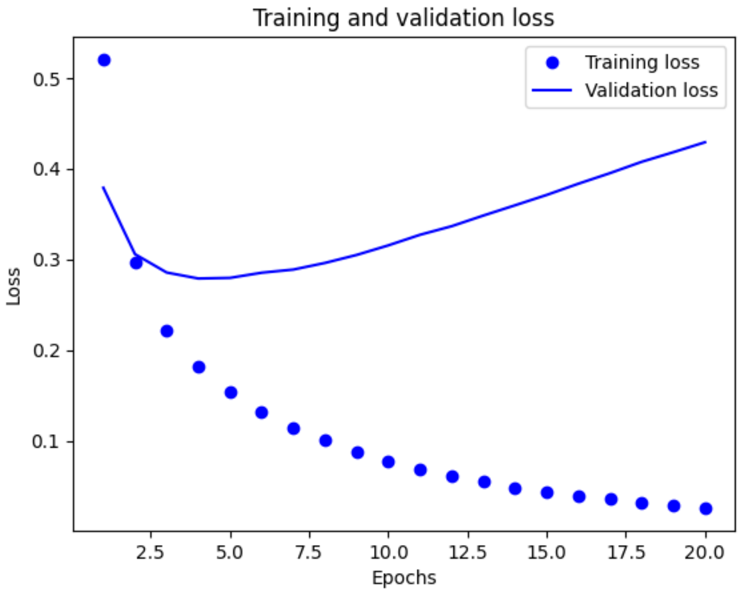
```
model1_3 = keras.Sequential([
    layers.Dense(16, activation="relu"),
    layers.Dense(16, activation="relu"),
    layers.Dense(16, activation="relu"),
    layers.Dense(1, activation="sigmoid")
])
```

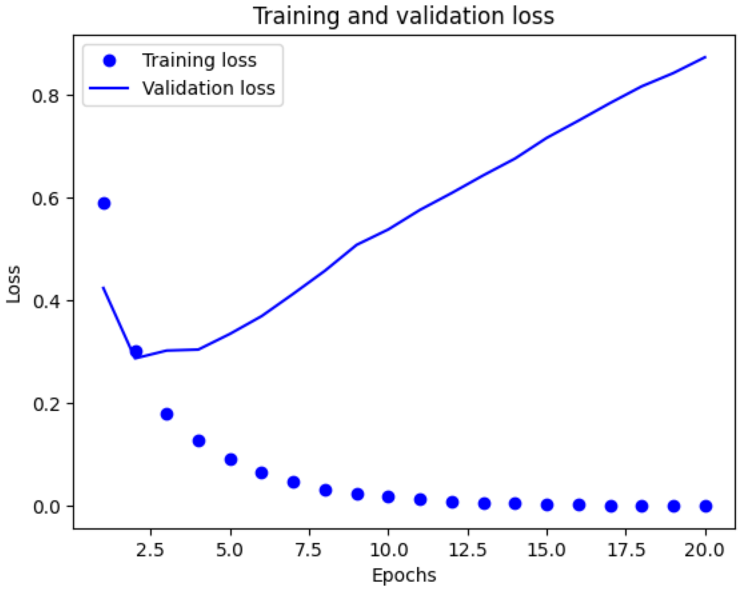
```
model1_1.compile(optimizer="adam",
              loss="binary_crossentropy",
              metrics=["accuracy"])

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

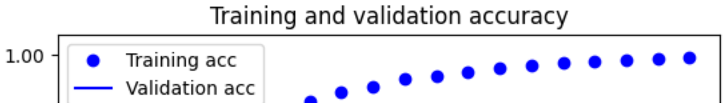
```
history1_1 = model1_1.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val, y_val))

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 [==============================] - 1s 18ms/step - loss: 0.5204 - accuracy: 0.7835 - val_loss: 0.3788 - val_accuracy: 0.8649
Epoch 2/20
30/30 [==============================] - 0s 8ms/step - loss: 0.2960 - accuracy: 0.9019 - val_loss: 0.3056 - val_accuracy: 0.8866
Epoch 3/20
30/30 [==============================] - 0s 8ms/step - loss: 0.2221 - accuracy: 0.9292 - val_loss: 0.2854 - val_accuracy: 0.8904
Epoch 4/20
30/30 [==============================] - 0s 8ms/step - loss: 0.1819 - accuracy: 0.9450 - val_loss: 0.2788 - val_accuracy: 0.8890
Epoch 5/20
30/30 [==============================] - 0s 8ms/step - loss: 0.1531 - accuracy: 0.9571 - val_loss: 0.2795 - val_accuracy: 0.8886
Epoch 6/20
30/30 [==============================] - 0s 8ms/step - loss: 0.1322 - accuracy: 0.9651 - val_loss: 0.2852 - val_accuracy: 0.8851
Epoch 7/20
30/30 [==============================] - 0s 8ms/step - loss: 0.1148 - accuracy: 0.9706 - val_loss: 0.2887 - val_accuracy: 0.8849
Epoch 8/20
30/30 [==============================] - 0s 7ms/step - loss: 0.1001 - accuracy: 0.9778 - val_loss: 0.2960 - val_accuracy: 0.8827
Epoch 9/20
30/30 [==============================] - 0s 7ms/step - loss: 0.0877 - accuracy: 0.9825 - val_loss: 0.3048 - val_accuracy: 0.8822
Epoch 10/20
30/30 [==============================] - 0s 7ms/step - loss: 0.0775 - accuracy: 0.9852 - val_loss: 0.3153 - val_accuracy: 0.8811
Epoch 11/20
30/30 [==============================] - 0s 7ms/step - loss: 0.0686 - accuracy: 0.9889 - val_loss: 0.3270 - val_accuracy: 0.8786
Epoch 12/20
30/30 [==============================] - 0s 8ms/step - loss: 0.0618 - accuracy: 0.9901 - val_loss: 0.3365 - val_accuracy: 0.8789
Epoch 13/20
30/30 [==============================] - 0s 8ms/step - loss: 0.0552 - accuracy: 0.9921 - val_loss: 0.3482 - val_accuracy: 0.8798
Epoch 14/20
30/30 [==============================] - 0s 8ms/step - loss: 0.0486 - accuracy: 0.9941 - val_loss: 0.3595 - val_accuracy: 0.8789
Epoch 15/20
30/30 [==============================] - 0s 7ms/step - loss: 0.0436 - accuracy: 0.9952 - val_loss: 0.3709 - val_accuracy: 0.8766
Epoch 16/20
30/30 [==============================] - 0s 8ms/step - loss: 0.0392 - accuracy: 0.9962 - val_loss: 0.3833 - val_accuracy: 0.8765
Epoch 17/20
30/30 [==============================] - 0s 7ms/step - loss: 0.0354 - accuracy: 0.9971 - val_loss: 0.3949 - val_accuracy: 0.8755
Epoch 18/20
30/30 [==============================] - 0s 8ms/step - loss: 0.0317 - accuracy: 0.9976 - val_loss: 0.4074 - val_accuracy: 0.8744
Epoch 19/20
30/30 [==============================] - 0s 8ms/step - loss: 0.0286 - accuracy: 0.9985 - val_loss: 0.4180 - val_accuracy: 0.8729
Epoch 20/20
30/30 [==============================] - 0s 7ms/step - loss: 0.0259 - accuracy: 0.9991 - val_loss: 0.4290 - val_accuracy: 0.8721
Epoch 1/20
30/30 [==============================] - 1s 21ms/step - loss: 0.5886 - accuracy: 0.7651 - val_loss: 0.4240 - val_accuracy: 0.8533
Epoch 2/20
30/30 [==============================] - 0s 8ms/step - loss: 0.3027 - accuracy: 0.9019 - val_loss: 0.2871 - val_accuracy: 0.8878
Epoch 3/20
30/30 [==============================] - 0s 9ms/step - loss: 0.1797 - accuracy: 0.9395 - val_loss: 0.3023 - val_accuracy: 0.8837
Epoch 4/20
30/30 [==============================] - 0s 8ms/step - loss: 0.1279 - accuracy: 0.9591 - val_loss: 0.3043 - val_accuracy: 0.8839
Epoch 5/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0904 - accuracy: 0.9749 - val_loss: 0.3348 - val_accuracy: 0.8820
Epoch 6/20
30/30 [==============================] - 0s 8ms/step - loss: 0.0649 - accuracy: 0.9857 - val_loss: 0.3693 - val_accuracy: 0.8800
Epoch 7/20
30/30 [==============================] - 0s 8ms/step - loss: 0.0465 - accuracy: 0.9919 - val_loss: 0.4128 - val_accuracy: 0.8764
Epoch 8/20
30/30 [==============================] - 0s 8ms/step - loss: 0.0332 - accuracy: 0.9957 - val_loss: 0.4576 - val_accuracy: 0.8767
Epoch 9/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0239 - accuracy: 0.9976 - val_loss: 0.5079 - val_accuracy: 0.8735
```

```
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()
```

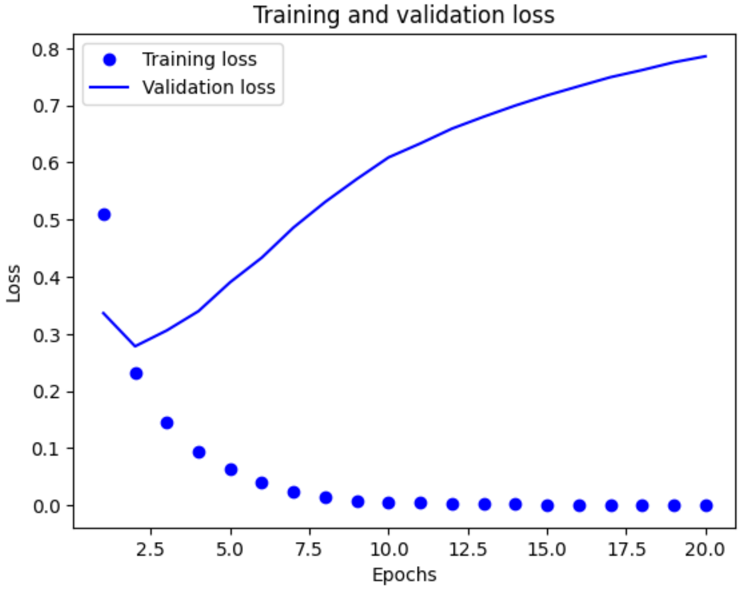

Training and validation loss
Training loss
Validation loss
Loss
0.5
0.4
0.3
0.2
0.1
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Epochs


Training and validation loss
Training loss
Validation loss
Loss
0.8
0.6
0.4
0.2
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Epochs


Training and validation accuracy
1.00
Training acc
Validation acc

```
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()
```

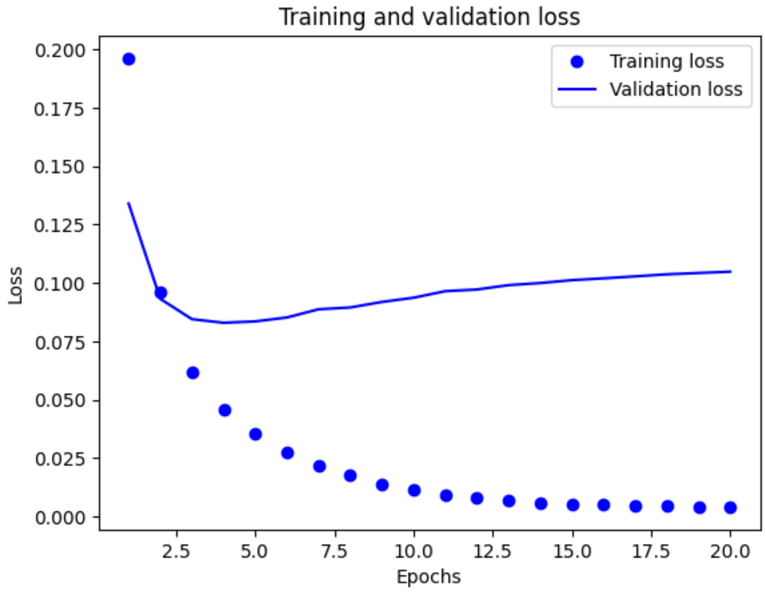
```
Epoch 1/20
30/30 [==============================] - 1s 22ms/step - loss: 0.5097 - accuracy: 0.79
Epoch 2/20
30/30 [==============================] - 0s 10ms/step - loss: 0.2316 - accuracy: 0.91
Epoch 3/20
30/30 [==============================] - 0s 9ms/step - loss: 0.1457 - accuracy: 0.949
Epoch 4/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0951 - accuracy: 0.971
Epoch 5/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0626 - accuracy: 0.984
Epoch 6/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0391 - accuracy: 0.991
Epoch 7/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0227 - accuracy: 0.997
Epoch 8/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0136 - accuracy: 0.999
Epoch 9/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0084 - accuracy: 0.999
Epoch 10/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0057 - accuracy: 0.999
Epoch 11/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0041 - accuracy: 0.999
Epoch 12/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0029 - accuracy: 1.000
Epoch 13/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0022 - accuracy: 1.000
Epoch 14/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0018 - accuracy: 1.000
Epoch 15/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0014 - accuracy: 1.000
Epoch 16/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0012 - accuracy: 1.000
Epoch 17/20
30/30 [==============================] - 0s 9ms/step - loss: 9.9245e-04 - accuracy: 1
Epoch 18/20
30/30 [==============================] - 0s 8ms/step - loss: 8.4754e-04 - accuracy: 1
Epoch 19/20
30/30 [==============================] - 0s 8ms/step - loss: 7.3224e-04 - accuracy: 1
Epoch 20/20
30/30 [==============================] - 0s 8ms/step - loss: 6.4065e-04 - accuracy: 1
```



Training and validation accuracy

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


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()
```

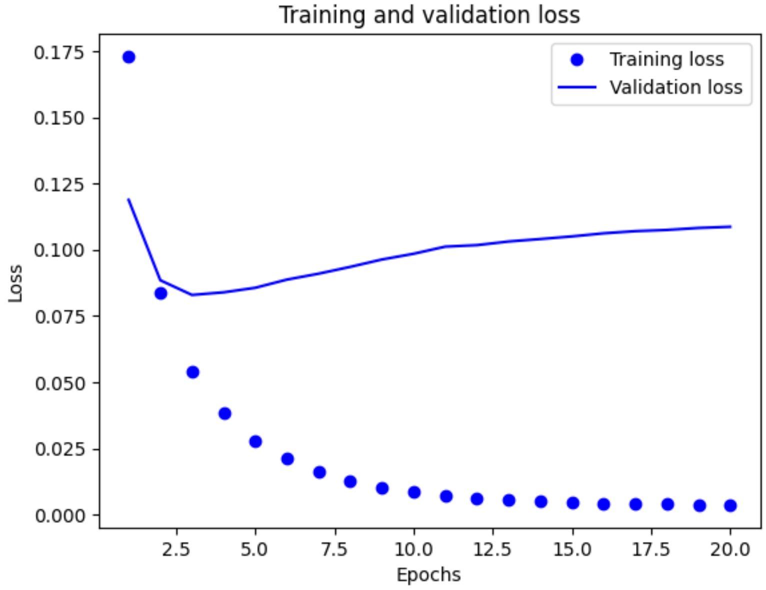
```
Epoch 1/20
30/30 [==============================] - 1s 23ms/step - loss: 0.1962 - accuracy: 0.75
Epoch 2/20
30/30 [==============================] - 0s 8ms/step - loss: 0.0963 - accuracy: 0.894
Epoch 3/20
30/30 [==============================] - 0s 8ms/step - loss: 0.0620 - accuracy: 0.933
Epoch 4/20
30/30 [==============================] - 0s 8ms/step - loss: 0.0456 - accuracy: 0.953
Epoch 5/20
30/30 [==============================] - 0s 8ms/step - loss: 0.0353 - accuracy: 0.968
Epoch 6/20
30/30 [==============================] - 0s 8ms/step - loss: 0.0275 - accuracy: 0.977
Epoch 7/20
30/30 [==============================] - 0s 8ms/step - loss: 0.0216 - accuracy: 0.983
Epoch 8/20
30/30 [==============================] - 0s 8ms/step - loss: 0.0177 - accuracy: 0.987
Epoch 9/20
30/30 [==============================] - 0s 8ms/step - loss: 0.0140 - accuracy: 0.990
Epoch 10/20
30/30 [==============================] - 0s 8ms/step - loss: 0.0113 - accuracy: 0.992
Epoch 11/20
30/30 [==============================] - 0s 7ms/step - loss: 0.0093 - accuracy: 0.994
Epoch 12/20
30/30 [==============================] - 0s 7ms/step - loss: 0.0080 - accuracy: 0.994
Epoch 13/20
30/30 [==============================] - 0s 8ms/step - loss: 0.0068 - accuracy: 0.995
Epoch 14/20
30/30 [==============================] - 0s 7ms/step - loss: 0.0059 - accuracy: 0.996
Epoch 15/20
30/30 [==============================] - 0s 8ms/step - loss: 0.0053 - accuracy: 0.996
Epoch 16/20
30/30 [==============================] - 0s 7ms/step - loss: 0.0049 - accuracy: 0.996
Epoch 17/20
30/30 [==============================] - 0s 7ms/step - loss: 0.0046 - accuracy: 0.996
Epoch 18/20
30/30 [==============================] - 0s 8ms/step - loss: 0.0044 - accuracy: 0.996
Epoch 19/20
30/30 [==============================] - 0s 8ms/step - loss: 0.0043 - accuracy: 0.996
Epoch 20/20
30/30 [==============================] - 0s 7ms/step - loss: 0.0041 - accuracy: 0.996
```



Training and validation accuracy

```
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()
```

```
Epoch 1/20
30/30 [==============================] - 1s 22ms/step - loss: 0.1730 - accuracy: 0.79
Epoch 2/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0839 - accuracy: 0.902
Epoch 3/20
30/30 [==============================] - 0s 8ms/step - loss: 0.0540 - accuracy: 0.940
Epoch 4/20
30/30 [==============================] - 0s 8ms/step - loss: 0.0381 - accuracy: 0.961
Epoch 5/20
30/30 [==============================] - 0s 8ms/step - loss: 0.0279 - accuracy: 0.976
Epoch 6/20
30/30 [==============================] - 0s 8ms/step - loss: 0.0212 - accuracy: 0.983
Epoch 7/20
30/30 [==============================] - 0s 8ms/step - loss: 0.0162 - accuracy: 0.987
Epoch 8/20
30/30 [==============================] - 0s 8ms/step - loss: 0.0126 - accuracy: 0.990
Epoch 9/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0101 - accuracy: 0.992
Epoch 10/20
30/30 [==============================] - 0s 8ms/step - loss: 0.0084 - accuracy: 0.993
Epoch 11/20
30/30 [==============================] - 0s 8ms/step - loss: 0.0072 - accuracy: 0.994
Epoch 12/20
30/30 [==============================] - 0s 8ms/step - loss: 0.0061 - accuracy: 0.995
Epoch 13/20
30/30 [==============================] - 0s 8ms/step - loss: 0.0054 - accuracy: 0.995
Epoch 14/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0049 - accuracy: 0.996
Epoch 15/20
30/30 [==============================] - 0s 8ms/step - loss: 0.0046 - accuracy: 0.996
Epoch 16/20
30/30 [==============================] - 0s 8ms/step - loss: 0.0043 - accuracy: 0.996
Epoch 17/20
30/30 [==============================] - 0s 8ms/step - loss: 0.0040 - accuracy: 0.996
Epoch 18/20
30/30 [==============================] - 0s 8ms/step - loss: 0.0038 - accuracy: 0.996
Epoch 19/20
30/30 [==============================] - 0s 8ms/step - loss: 0.0037 - accuracy: 0.996
Epoch 20/20
30/30 [==============================] - 0s 8ms/step - loss: 0.0036 - accuracy: 0.996
```



```
<matplotlib.legend.Legend at 0x1d720932a00>
```

Training and validation accuracy