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

```
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
```

```
train_data[0]
```

```
        4472,
        113,
        103,
        32,
        15,
        16,
        5345,
        19,
        178,
        32]
```

```
train_labels[0]
```

```
1
```

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

```
9999
```

Decoding reviews back to 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]])
```

```
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 1

```
from tensorflow import keras
from tensorflow.keras import layers

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

Compiling the model

```
model.compile(optimizer="rmsprop",
              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

```
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 ━━━━━━━━━━━━━━━━━━━━ 3s 65ms/step - accuracy: 0.6735 - loss: 0.6138 - val_accuracy: 0.8671 - val_loss: 0.4079
Epoch 2/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - accuracy: 0.8922 - loss: 0.3523 - val_accuracy: 0.8741 - val_loss: 0.3259
Epoch 3/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - accuracy: 0.9197 - loss: 0.2527 - val_accuracy: 0.8848 - val_loss: 0.2939
Epoch 4/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - accuracy: 0.9399 - loss: 0.1982 - val_accuracy: 0.8770 - val_loss: 0.3034
Epoch 5/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.9451 - loss: 0.1678 - val_accuracy: 0.8825 - val_loss: 0.2983
Epoch 6/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 47ms/step - accuracy: 0.9620 - loss: 0.1326 - val_accuracy: 0.8861 - val_loss: 0.2837
Epoch 7/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 60ms/step - accuracy: 0.9681 - loss: 0.1149 - val_accuracy: 0.8741 - val_loss: 0.3189
Epoch 8/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 60ms/step - accuracy: 0.9714 - loss: 0.0992 - val_accuracy: 0.8823 - val_loss: 0.3066
Epoch 9/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 60ms/step - accuracy: 0.9791 - loss: 0.0831 - val_accuracy: 0.8822 - val_loss: 0.3227
Epoch 10/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 49ms/step - accuracy: 0.9849 - loss: 0.0687 - val_accuracy: 0.8727 - val_loss: 0.3534
Epoch 11/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 60ms/step - accuracy: 0.9878 - loss: 0.0563 - val_accuracy: 0.8754 - val_loss: 0.3644
Epoch 12/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 60ms/step - accuracy: 0.9894 - loss: 0.0511 - val_accuracy: 0.8759 - val_loss: 0.3755
Epoch 13/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 68ms/step - accuracy: 0.9930 - loss: 0.0412 - val_accuracy: 0.8768 - val_loss: 0.4066
Epoch 14/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 51ms/step - accuracy: 0.9950 - loss: 0.0348 - val_accuracy: 0.8727 - val_loss: 0.4208
Epoch 15/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 49ms/step - accuracy: 0.9972 - loss: 0.0273 - val_accuracy: 0.8759 - val_loss: 0.4410
Epoch 16/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 43ms/step - accuracy: 0.9964 - loss: 0.0258 - val_accuracy: 0.8622 - val_loss: 0.5433
Epoch 17/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 43ms/step - accuracy: 0.9958 - loss: 0.0237 - val_accuracy: 0.8727 - val_loss: 0.4797
Epoch 18/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 41ms/step - accuracy: 0.9988 - loss: 0.0147 - val_accuracy: 0.8729 - val_loss: 0.5019
Epoch 19/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 53ms/step - accuracy: 0.9987 - loss: 0.0145 - val_accuracy: 0.8709 - val_loss: 0.5205
Epoch 20/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 49ms/step - accuracy: 0.9995 - loss: 0.0098 - val_accuracy: 0.8661 - val_loss: 0.5863
```

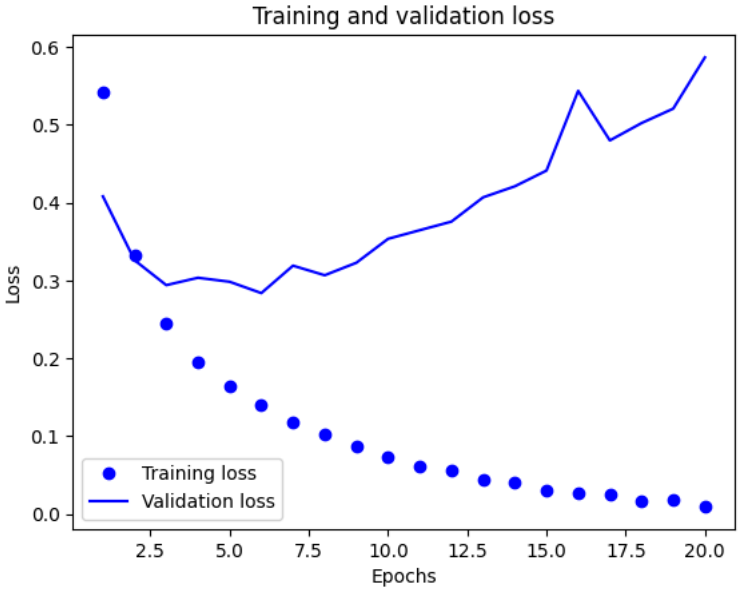
```
history_dict = history.history
history_dict.keys()
```

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

Plotting the training and validation loss

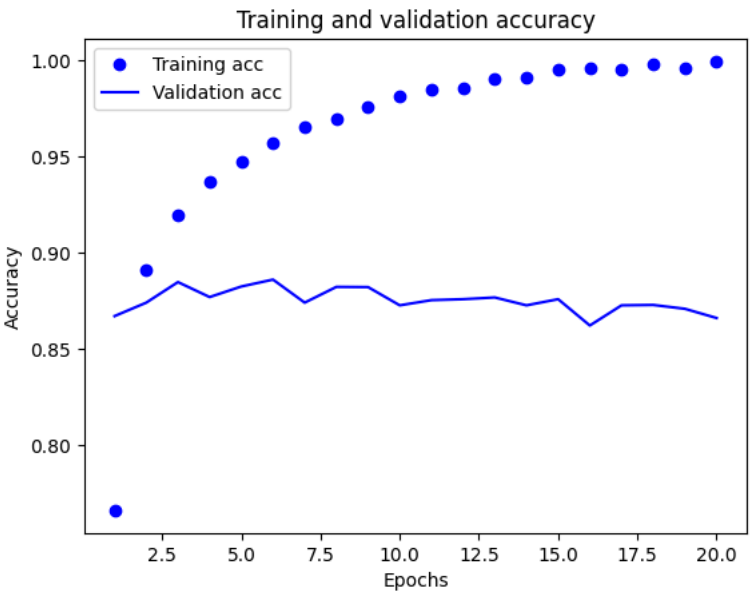
```
import matplotlib.pyplot as plt
history_dict = history.history
loss_values = history_dict["loss"]
val_loss_values = history_dict["val_loss"]
epochs = range(1, len(loss_values) + 1)
plt.plot(epochs, loss_values, "bo", label="Training loss")
plt.plot(epochs, val_loss_values, "b", label="Validation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



Retraining a model from scratch

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

```
    layers.Dense(1, activation="sigmoid")
])
model.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])
model.fit(x_train, y_train, epochs=4, batch_size=512)
results_test = model.evaluate(x_test, y_test)
```

```
Epoch 1/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 24ms/step - accuracy: 0.7292 - loss: 0.5621
Epoch 2/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - accuracy: 0.9020 - loss: 0.2871
Epoch 3/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 23ms/step - accuracy: 0.9218 - loss: 0.2192
Epoch 4/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - accuracy: 0.9370 - loss: 0.1780
782/782 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - accuracy: 0.8834 - loss: 0.2867
```

```
results_test
```

```
[0.28576093912124634, 0.8852800130844116]
```

```
model.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])
model.fit(x_train, y_train, epochs=4, batch_size=512)
results_val = model.evaluate(x_test, y_test)
```

```
Epoch 1/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 25ms/step - accuracy: 0.9388 - loss: 0.1716
Epoch 2/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - accuracy: 0.9531 - loss: 0.1426
Epoch 3/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - accuracy: 0.9582 - loss: 0.1268
Epoch 4/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 3s 34ms/step - accuracy: 0.9598 - loss: 0.1181
782/782 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - accuracy: 0.8661 - loss: 0.3711
```

```
results_val
```

```
[0.368953138589859, 0.8681600093841553]
```

Using a trained model to generate predictions on new data

```
model.predict(x_test)
```

```
782/782 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step
array([[0.06319276],
       [0.9999145 ],
       [0.15988427],
       ...,
       [0.08197054],
       [0.0265181 ],
       [0.5328368 ]], dtype=float32)
```

Model 2

```
from tensorflow import keras
from tensorflow.keras import layers

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

Compiling the model

```
model.compile(optimizer="rmsprop",
              loss="mse",
              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

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

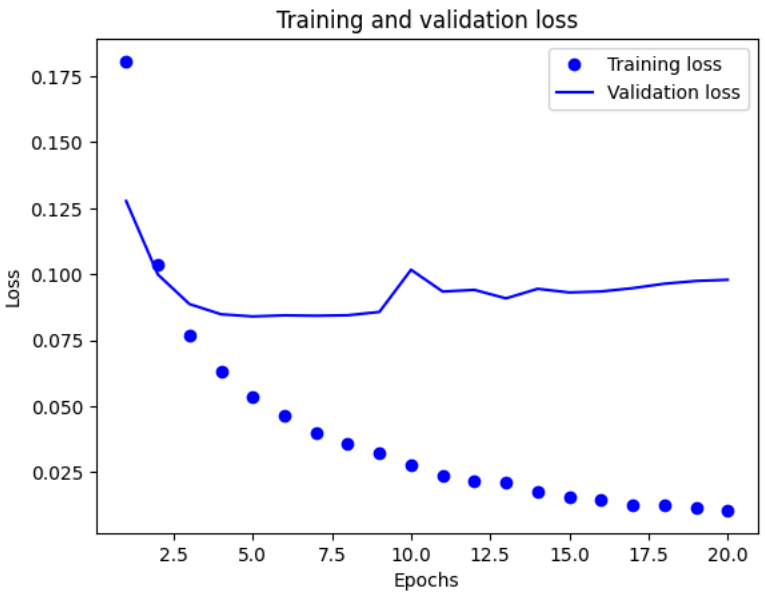
Show hidden output

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

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

Plotting the training and validation loss

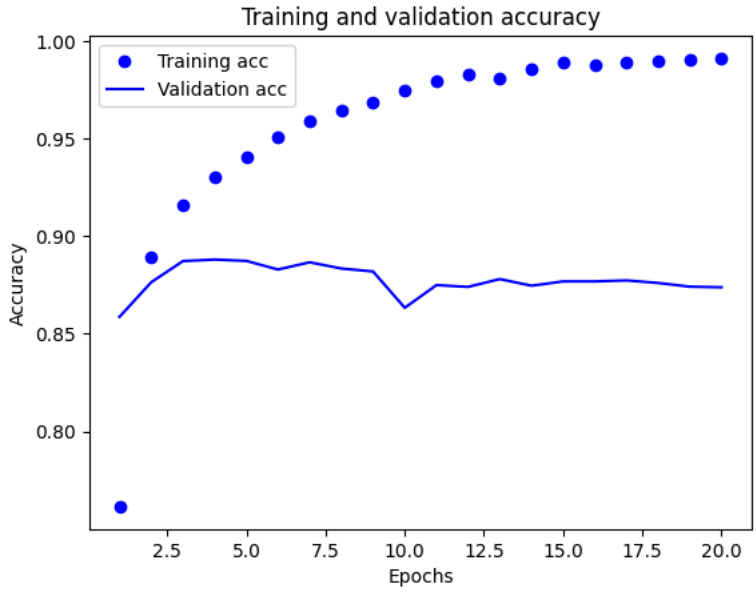
```
import matplotlib.pyplot as plt
history_dict = history.history
loss_values = history_dict["loss"]
val_loss_values = history_dict["val_loss"]
epochs = range(1, len(loss_values) + 1)
plt.plot(epochs, loss_values, "bo", label="Training loss")
plt.plot(epochs, val_loss_values, "b", label="Validation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



Retraining a model from scratch

```
model = keras.Sequential([
    layers.Dense(16, activation="relu"),
    layers.Dense(1, activation="sigmoid")
])
model.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])
model.fit(x_train, y_train, epochs=4, batch_size=512)
results_test = model.evaluate(x_test, y_test)
```

```
Epoch 1/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 25ms/step - accuracy: 0.7556 - loss: 0.5253
Epoch 2/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - accuracy: 0.9020 - loss: 0.2917
Epoch 3/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 29ms/step - accuracy: 0.9221 - loss: 0.2281
Epoch 4/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 36ms/step - accuracy: 0.9363 - loss: 0.1929
782/782 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - accuracy: 0.8835 - loss: 0.2872
```

```
results_test
```

```
[0.28754010796546936, 0.8842399716377258]
```

```
model.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])
model.fit(x_train, y_train, epochs=4, batch_size=512)
results_val = model.evaluate(x_test, y_test)
```

```
Epoch 1/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 25ms/step - accuracy: 0.9400 - loss: 0.1790
Epoch 2/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - accuracy: 0.9484 - loss: 0.1582
Epoch 3/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - accuracy: 0.9539 - loss: 0.1459
Epoch 4/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 3s 37ms/step - accuracy: 0.9539 - loss: 0.1389
782/782 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - accuracy: 0.8755 - loss: 0.3235
```

```
results_val
```

```
[0.32372936606407166, 0.8751199841499329]
```

Using a trained model to generate predictions on new data

```
model.predict(x_test)
```

```
782/782 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step
array([[0.10797434],
       [0.9999238 ],
       [0.37158963],
       ...,
       [0.10583925],
       [0.03413222],
       [0.45355377]], dtype=float32)
```

Model 3

```
from tensorflow import keras
from tensorflow.keras import layers

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

Compiling the model

```
model.compile(optimizer="rmsprop",
              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 the model

```
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 ━━━━━━━━━━━━━━━━━━━━ 3s 69ms/step - accuracy: 0.7026 - loss: 0.5903 - val_accuracy: 0.8571 - val_loss: 0.4068
Epoch 2/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 36ms/step - accuracy: 0.8911 - loss: 0.3516 - val_accuracy: 0.8765 - val_loss: 0.3340
Epoch 3/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.9154 - loss: 0.2721 - val_accuracy: 0.8884 - val_loss: 0.2945
Epoch 4/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - accuracy: 0.9295 - loss: 0.2244 - val_accuracy: 0.8892 - val_loss: 0.2789
Epoch 5/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - accuracy: 0.9410 - loss: 0.1908 - val_accuracy: 0.8854 - val_loss: 0.2780
Epoch 6/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 60ms/step - accuracy: 0.9521 - loss: 0.1658 - val_accuracy: 0.8875 - val_loss: 0.2737
Epoch 7/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 43ms/step - accuracy: 0.9568 - loss: 0.1480 - val_accuracy: 0.8854 - val_loss: 0.2856
Epoch 8/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.9640 - loss: 0.1307 - val_accuracy: 0.8872 - val_loss: 0.2794
Epoch 9/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - accuracy: 0.9673 - loss: 0.1157 - val_accuracy: 0.8851 - val_loss: 0.2883
Epoch 10/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.9729 - loss: 0.1053 - val_accuracy: 0.8830 - val_loss: 0.2938
Epoch 11/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - accuracy: 0.9764 - loss: 0.0951 - val_accuracy: 0.8809 - val_loss: 0.3036
Epoch 12/20
```

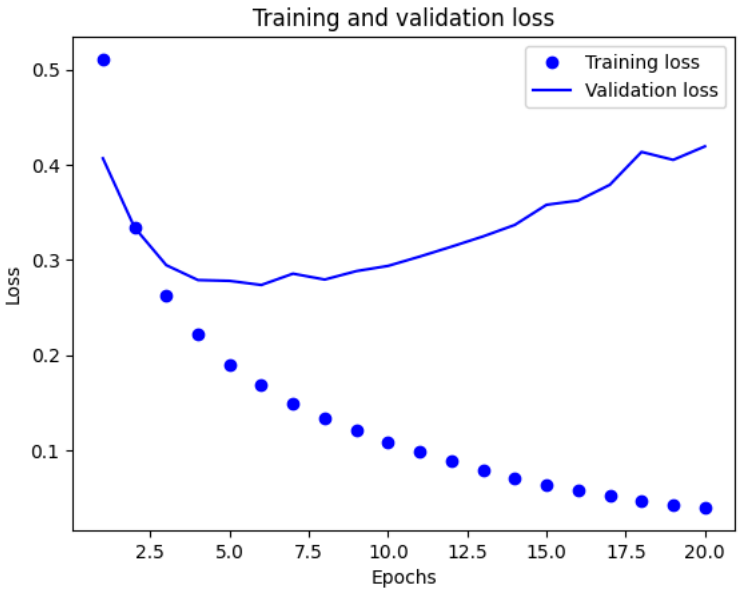
```
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.9785 - loss: 0.0858 - val_accuracy: 0.8784 - val_loss: 0.3139
Epoch 13/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.9822 - loss: 0.0769 - val_accuracy: 0.8810 - val_loss: 0.3247
Epoch 14/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - accuracy: 0.9863 - loss: 0.0669 - val_accuracy: 0.8805 - val_loss: 0.3368
Epoch 15/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - accuracy: 0.9895 - loss: 0.0594 - val_accuracy: 0.8774 - val_loss: 0.3580
Epoch 16/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 47ms/step - accuracy: 0.9895 - loss: 0.0559 - val_accuracy: 0.8762 - val_loss: 0.3624
Epoch 17/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 33ms/step - accuracy: 0.9917 - loss: 0.0476 - val_accuracy: 0.8741 - val_loss: 0.3790
Epoch 18/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - accuracy: 0.9932 - loss: 0.0433 - val_accuracy: 0.8721 - val_loss: 0.4135
Epoch 19/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 80ms/step - accuracy: 0.9946 - loss: 0.0401 - val_accuracy: 0.8730 - val_loss: 0.4052
Epoch 20/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 108ms/step - accuracy: 0.9942 - loss: 0.0355 - val_accuracy: 0.8722 - val_loss: 0.4194
```

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

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

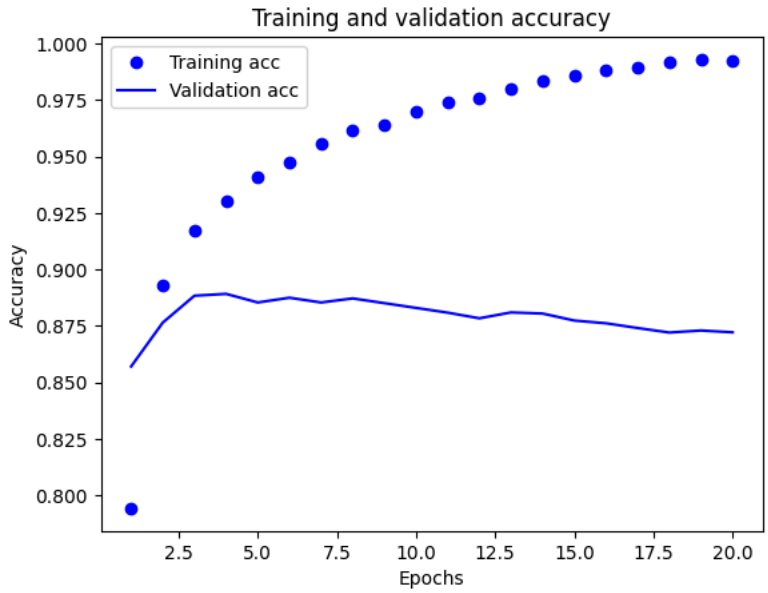
Plotting the training and validation loss

```
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()
```



Retraining a model from scratch

```
model = keras.Sequential([
    layers.Dense(16, activation="relu"),
    layers.Dense(16, activation="relu"),
    layers.Dense(16, activation="relu"),
    layers.Dense(1, activation="sigmoid")
])
model.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])
model.fit(x_train, y_train, epochs=4, batch_size=512)
results_test = model.evaluate(x_test, y_test)
```

```
Epoch 1/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 3s 29ms/step - accuracy: 0.7356 - loss: 0.5685
Epoch 2/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 3s 28ms/step - accuracy: 0.8995 - loss: 0.2856
Epoch 3/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 27ms/step - accuracy: 0.9184 - loss: 0.2139
Epoch 4/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 35ms/step - accuracy: 0.9389 - loss: 0.1732
782/782 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.8625 - loss: 0.3551
```

```
results_test
```

```
[0.3480762541294098, 0.8647599816322327]
```

```
model.fit(x_train, y_train, epochs=4, batch_size=512)
results_val = model.evaluate(x_test, y_test)
```

```
Epoch 1/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - accuracy: 0.9448 - loss: 0.1536
Epoch 2/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step - accuracy: 0.9571 - loss: 0.1311
Epoch 3/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - accuracy: 0.9630 - loss: 0.1131
Epoch 4/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - accuracy: 0.9675 - loss: 0.0982
782/782 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.8709 - loss: 0.3815
```

```
results_val
```

```
[0.3773863613605499, 0.8717600107192993]
```

Using a trained model to generate predictions on new data

```
model.predict(x_test)
```

```
782/782 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step
array([[0.07496766],
       [0.99994016],
       [0.6939374 ],
       ...,
       [0.06111119],
       [0.02972515],
       [0.83877164]], dtype=float32)
```

Model 4

```
# cresate model with 32 units in the hidden layer
from tensorflow import keras
from tensorflow.keras import layers

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

Compiling the model

```
model.compile(optimizer="rmsprop",
              loss="mse",
              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 the model

```
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 ━━━━━━━━━━━━━━━━━━━━ 5s 104ms/step - accuracy: 0.6877 - loss: 0.2121 - val_accuracy: 0.8556 - val_loss: 0.1247
Epoch 2/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 4s 55ms/step - accuracy: 0.8769 - loss: 0.1078 - val_accuracy: 0.8705 - val_loss: 0.1009
Epoch 3/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 54ms/step - accuracy: 0.9064 - loss: 0.0774 - val_accuracy: 0.8846 - val_loss: 0.0878
Epoch 4/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 67ms/step - accuracy: 0.9241 - loss: 0.0636 - val_accuracy: 0.8898 - val_loss: 0.0833
Epoch 5/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 71ms/step - accuracy: 0.9414 - loss: 0.0504 - val_accuracy: 0.8606 - val_loss: 0.1037
Epoch 6/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 41ms/step - accuracy: 0.9519 - loss: 0.0432 - val_accuracy: 0.8830 - val_loss: 0.0888
Epoch 7/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 42ms/step - accuracy: 0.9614 - loss: 0.0362 - val_accuracy: 0.8705 - val_loss: 0.0982
Epoch 8/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 41ms/step - accuracy: 0.9652 - loss: 0.0319 - val_accuracy: 0.8503 - val_loss: 0.1167
```

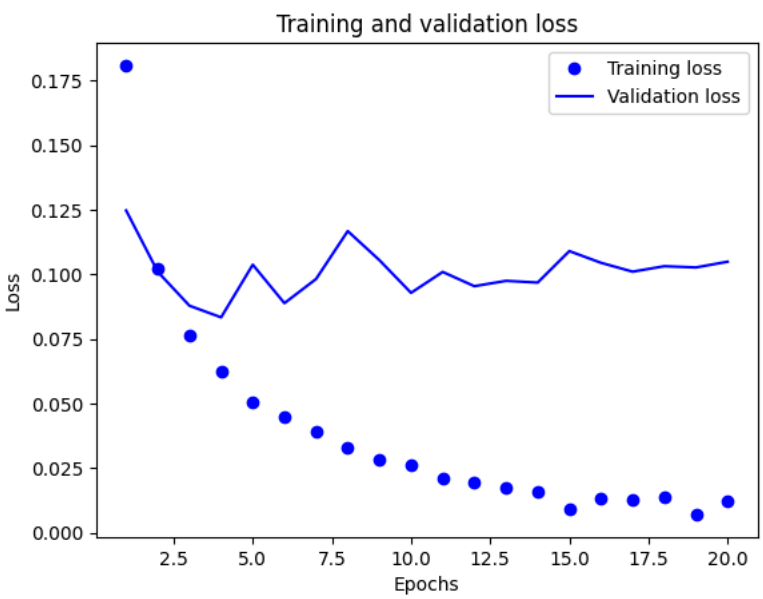
```
Epoch 9/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 53ms/step - accuracy: 0.9647 - loss: 0.0307 - val_accuracy: 0.8618 - val_loss: 0.1055
Epoch 10/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 42ms/step - accuracy: 0.9683 - loss: 0.0279 - val_accuracy: 0.8772 - val_loss: 0.0928
Epoch 11/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 71ms/step - accuracy: 0.9805 - loss: 0.0196 - val_accuracy: 0.8714 - val_loss: 0.1009
Epoch 12/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 43ms/step - accuracy: 0.9846 - loss: 0.0160 - val_accuracy: 0.8775 - val_loss: 0.0954
Epoch 13/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 43ms/step - accuracy: 0.9858 - loss: 0.0140 - val_accuracy: 0.8765 - val_loss: 0.0974
Epoch 14/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 54ms/step - accuracy: 0.9875 - loss: 0.0131 - val_accuracy: 0.8782 - val_loss: 0.0968
Epoch 15/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 53ms/step - accuracy: 0.9916 - loss: 0.0098 - val_accuracy: 0.8673 - val_loss: 0.1089
Epoch 16/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 53ms/step - accuracy: 0.9769 - loss: 0.0198 - val_accuracy: 0.8713 - val_loss: 0.1044
Epoch 17/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 42ms/step - accuracy: 0.9778 - loss: 0.0197 - val_accuracy: 0.8760 - val_loss: 0.1010
Epoch 18/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 72ms/step - accuracy: 0.9932 - loss: 0.0071 - val_accuracy: 0.8768 - val_loss: 0.1031
Epoch 19/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 64ms/step - accuracy: 0.9932 - loss: 0.0075 - val_accuracy: 0.8747 - val_loss: 0.1026
Epoch 20/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 43ms/step - accuracy: 0.9933 - loss: 0.0068 - val_accuracy: 0.8728 - val_loss: 0.1048
```

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

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

Plotting the training and validation loss

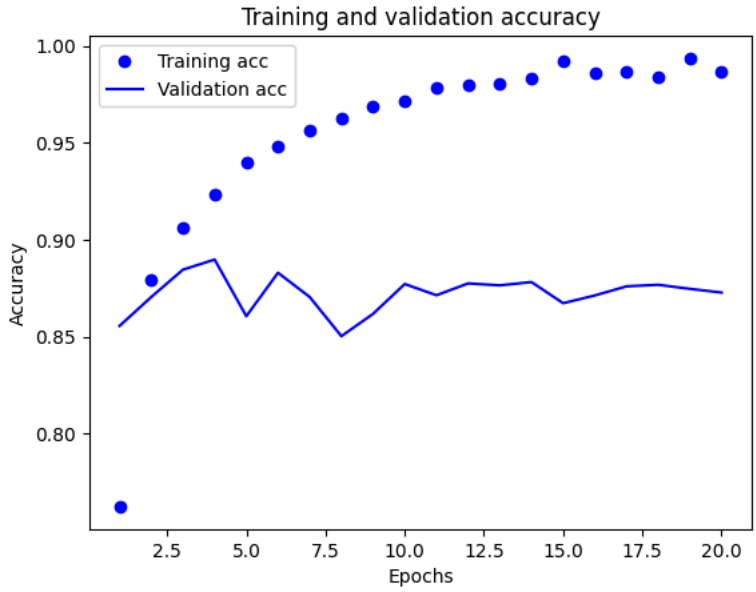
```
import matplotlib.pyplot as plt
history_dict = history.history
loss_values = history_dict["loss"]
val_loss_values = history_dict["val_loss"]
epochs = range(1, len(loss_values) + 1)
plt.plot(epochs, loss_values, "bo",  label="Training loss")
plt.plot(epochs, val_loss_values, "b", label="Validation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



Retraining a model from scratch

```
model = keras.Sequential([
    layers.Dense(32, activation="relu"),
    layers.Dense(1, activation="sigmoid")
])
model.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])
model.fit(x_train, y_train, epochs=4, batch_size=512)
results_test = model.evaluate(x_test, y_test)
```

```
Epoch 1/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 30ms/step - accuracy: 0.7488 - loss: 0.5342
Epoch 2/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 3s 33ms/step - accuracy: 0.8991 - loss: 0.2900
Epoch 3/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 3s 41ms/step - accuracy: 0.9219 - loss: 0.2263
Epoch 4/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 30ms/step - accuracy: 0.9295 - loss: 0.2011
782/782 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.8847 - loss: 0.2811
```

```
results_test
```

```
[0.28151941299438477, 0.8855599761009216]
```

```
model.fit(x_train, y_train, epochs=4, batch_size=512)
results_val = model.evaluate(x_test, y_test)
```

```
Epoch 1/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 30ms/step - accuracy: 0.9360 - loss: 0.1774
Epoch 2/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 3s 45ms/step - accuracy: 0.9442 - loss: 0.1598
Epoch 3/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 30ms/step - accuracy: 0.9512 - loss: 0.1457
Epoch 4/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 34ms/step - accuracy: 0.9553 - loss: 0.1353
782/782 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.8768 - loss: 0.3088
```

```
results_val
```

```
[0.3073391318321228, 0.8791599869728088]
```

Using a trained model to generate predictions on new data

```
model.predict(x_test)
```

```
782/782 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step
array([[0.13780561],
       [0.9999626 ],
       [0.48642397],
       ...,
       [0.11182618],
       [0.04618336],
       [0.54677534]], dtype=float32)
```

Model 5

```
# creating the model with 64 units in hidden layer
from tensorflow import keras
from tensorflow.keras import layers

model = keras.Sequential([
    layers.Dense(64, activation="tanh"),
    layers.Dense(64, activation="tanh"),
    layers.Dropout(0.5),
    layers.Dense(1, activation="sigmoid")
])
```

Compiling the model

```
model.compile(optimizer="rmsprop",
              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 the model

```
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 ━━━━━━━━━━━━━━━━━━━━ 5s 119ms/step - accuracy: 0.6784 - loss: 0.5856 - val_accuracy: 0.8613 - val_loss: 0.3443
Epoch 2/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 4s 72ms/step - accuracy: 0.8872 - loss: 0.2892 - val_accuracy: 0.8457 - val_loss: 0.3635
Epoch 3/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 70ms/step - accuracy: 0.9231 - loss: 0.2069 - val_accuracy: 0.8831 - val_loss: 0.2963
Epoch 4/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 92ms/step - accuracy: 0.9260 - loss: 0.1823 - val_accuracy: 0.8802 - val_loss: 0.3113
Epoch 5/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 4s 69ms/step - accuracy: 0.9495 - loss: 0.1350 - val_accuracy: 0.8743 - val_loss: 0.3428
Epoch 6/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 67ms/step - accuracy: 0.9617 - loss: 0.1090 - val_accuracy: 0.8741 - val_loss: 0.3608
Epoch 7/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 60ms/step - accuracy: 0.9648 - loss: 0.0973 - val_accuracy: 0.8254 - val_loss: 0.6193
Epoch 8/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 4s 98ms/step - accuracy: 0.9628 - loss: 0.1055 - val_accuracy: 0.8742 - val_loss: 0.4243
Epoch 9/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 69ms/step - accuracy: 0.9826 - loss: 0.0559 - val_accuracy: 0.8709 - val_loss: 0.4638
Epoch 10/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 61ms/step - accuracy: 0.9859 - loss: 0.0485 - val_accuracy: 0.8700 - val_loss: 0.4866
Epoch 11/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 71ms/step - accuracy: 0.9876 - loss: 0.0423 - val_accuracy: 0.8686 - val_loss: 0.5209
Epoch 12/20
```

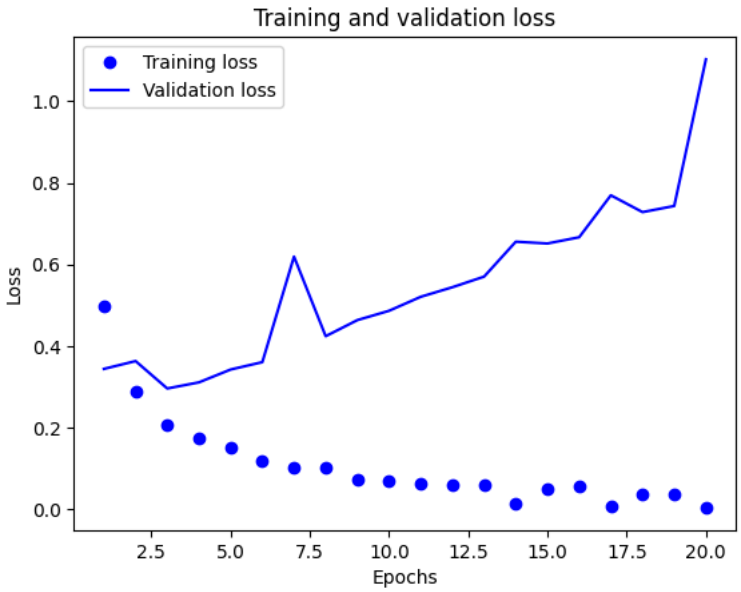
```
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 67ms/step - accuracy: 0.9890 - loss: 0.0421 - val_accuracy: 0.8722 - val_loss: 0.5443
Epoch 13/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 96ms/step - accuracy: 0.9956 - loss: 0.0230 - val_accuracy: 0.8701 - val_loss: 0.5703
Epoch 14/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 68ms/step - accuracy: 0.9984 - loss: 0.0146 - val_accuracy: 0.8657 - val_loss: 0.6558
Epoch 15/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 65ms/step - accuracy: 0.9848 - loss: 0.0531 - val_accuracy: 0.8688 - val_loss: 0.6515
Epoch 16/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 65ms/step - accuracy: 0.9956 - loss: 0.0208 - val_accuracy: 0.8691 - val_loss: 0.6667
Epoch 17/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 66ms/step - accuracy: 0.9997 - loss: 0.0073 - val_accuracy: 0.8567 - val_loss: 0.7695
Epoch 18/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 59ms/step - accuracy: 0.9804 - loss: 0.0621 - val_accuracy: 0.8673 - val_loss: 0.7283
Epoch 19/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 115ms/step - accuracy: 0.9938 - loss: 0.0235 - val_accuracy: 0.8650 - val_loss: 0.7433
Epoch 20/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 66ms/step - accuracy: 0.9997 - loss: 0.0043 - val_accuracy: 0.8277 - val_loss: 1.1026
```

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

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

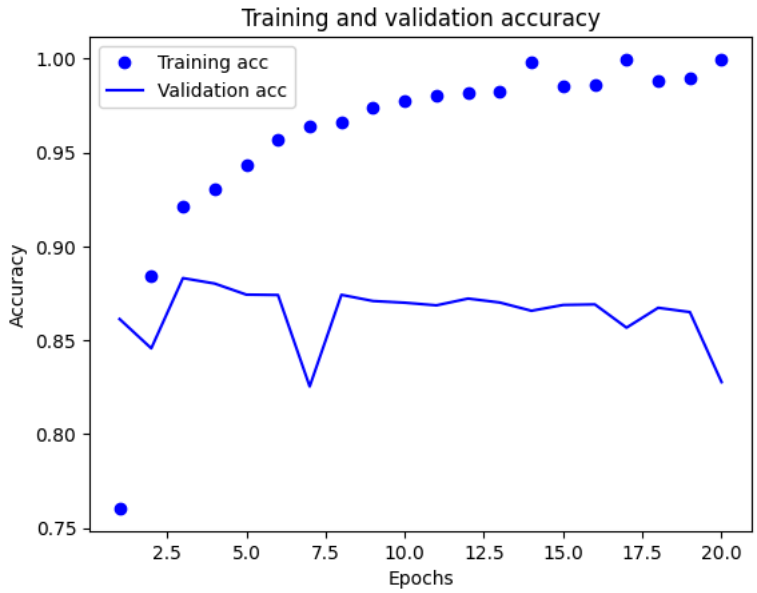
Plotting the training and validation loss

```
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()
```



Retraining a model from scratch

```
model = keras.Sequential([
    layers.Dense(64, activation="relu"),
    layers.Dense(1, activation="sigmoid")
])
model.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])
model.fit(x_train, y_train, epochs=4, batch_size=512)
results_test = model.evaluate(x_test, y_test)
```

```
Epoch 1/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 4s 44ms/step - accuracy: 0.7316 - loss: 0.5293
Epoch 2/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 3s 51ms/step - accuracy: 0.8947 - loss: 0.2773
Epoch 3/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 5s 56ms/step - accuracy: 0.9235 - loss: 0.2181
Epoch 4/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 41ms/step - accuracy: 0.9320 - loss: 0.1889
782/782 ━━━━━━━━━━━━━━━━━━━━ 3s 4ms/step - accuracy: 0.8770 - loss: 0.3021
```

```
results_test
```

```
[0.2976061999797821, 0.8805999755859375]
```

```
model.fit(x_train, y_train, epochs=4, batch_size=512)
results_val = model.evaluate(x_test, y_test)
```

```
Epoch 1/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 44ms/step - accuracy: 0.9317 - loss: 0.1843
Epoch 2/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 3s 44ms/step - accuracy: 0.9413 - loss: 0.1651
Epoch 3/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 3s 55ms/step - accuracy: 0.9502 - loss: 0.1462
Epoch 4/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 5s 55ms/step - accuracy: 0.9490 - loss: 0.1390
782/782 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - accuracy: 0.8768 - loss: 0.3184
```

```
results_val
```

```
[0.3156396150588989, 0.8794800043106079]
```

Using a trained model to generate predictions on new data

```
model.predict(x_test)
```

```
782/782 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step
array([[0.12260523],
       [0.99998844],
       [0.6606853 ],
       ...,
       [0.09730489],
       [0.07166199],
       [0.73363024]], dtype=float32)
```

Model 6

```
# creating the model with MSE loss function
from tensorflow import keras
from tensorflow.keras import layers, regularizers

model = keras.Sequential([
    layers.Dense(16, activation="relu", kernel_regularizer=regularizers.l2(0.01)),
    layers.Dense(1, activation="sigmoid")
])
```

Compiling the model

```
model.compile(optimizer="rmsprop",
              loss="mse",
              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 the model

```
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 94ms/step - accuracy: 0.7097 - loss: 0.3924 - val_accuracy: 0.8342 - val_loss: 0.1888
Epoch 2/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 4s 56ms/step - accuracy: 0.8625 - loss: 0.1697 - val_accuracy: 0.8679 - val_loss: 0.1567
Epoch 3/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 52ms/step - accuracy: 0.8808 - loss: 0.1503 - val_accuracy: 0.8723 - val_loss: 0.1504
Epoch 4/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 41ms/step - accuracy: 0.8847 - loss: 0.1434 - val_accuracy: 0.8469 - val_loss: 0.1569
Epoch 5/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - accuracy: 0.8784 - loss: 0.1433 - val_accuracy: 0.8524 - val_loss: 0.1529
Epoch 6/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.8799 - loss: 0.1404 - val_accuracy: 0.8531 - val_loss: 0.1522
Epoch 7/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - accuracy: 0.8833 - loss: 0.1390 - val_accuracy: 0.8524 - val_loss: 0.1509
Epoch 8/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 39ms/step - accuracy: 0.8890 - loss: 0.1353 - val_accuracy: 0.8246 - val_loss: 0.1618
Epoch 9/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 45ms/step - accuracy: 0.8742 - loss: 0.1399 - val_accuracy: 0.8721 - val_loss: 0.1413
Epoch 10/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 47ms/step - accuracy: 0.8839 - loss: 0.1353 - val_accuracy: 0.8669 - val_loss: 0.1420
```

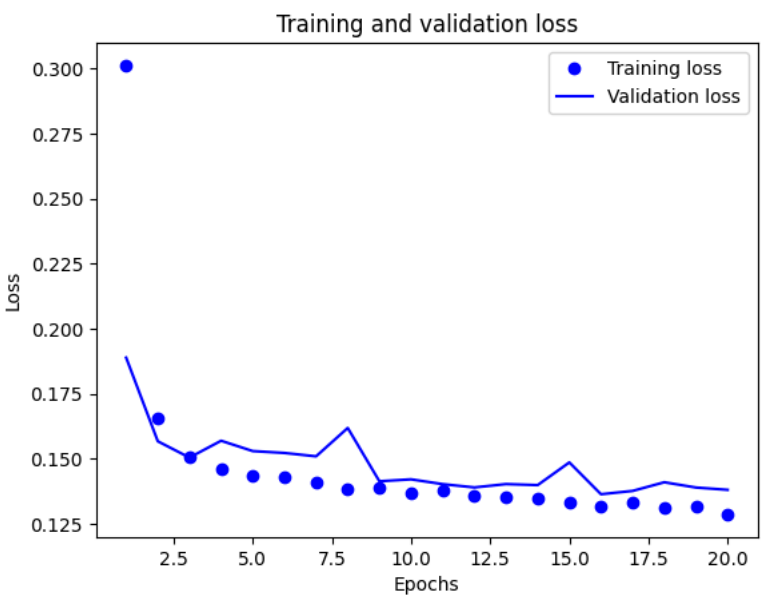
```
Epoch 11/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 46ms/step - accuracy: 0.8871 - loss: 0.1344 - val_accuracy: 0.8723 - val_loss: 0.1402
Epoch 12/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 35ms/step - accuracy: 0.8868 - loss: 0.1316 - val_accuracy: 0.8712 - val_loss: 0.1389
Epoch 13/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.8895 - loss: 0.1314 - val_accuracy: 0.8676 - val_loss: 0.1402
Epoch 14/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - accuracy: 0.8832 - loss: 0.1322 - val_accuracy: 0.8660 - val_loss: 0.1398
Epoch 15/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - accuracy: 0.8904 - loss: 0.1300 - val_accuracy: 0.8465 - val_loss: 0.1485
Epoch 16/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - accuracy: 0.8855 - loss: 0.1299 - val_accuracy: 0.8744 - val_loss: 0.1363
Epoch 17/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 54ms/step - accuracy: 0.8822 - loss: 0.1316 - val_accuracy: 0.8703 - val_loss: 0.1376
Epoch 18/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - accuracy: 0.8895 - loss: 0.1283 - val_accuracy: 0.8604 - val_loss: 0.1409
Epoch 19/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 39ms/step - accuracy: 0.8833 - loss: 0.1303 - val_accuracy: 0.8645 - val_loss: 0.1389
Epoch 20/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 55ms/step - accuracy: 0.8930 - loss: 0.1268 - val_accuracy: 0.8663 - val_loss: 0.1380
```

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

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

Plotting the training and validation loss

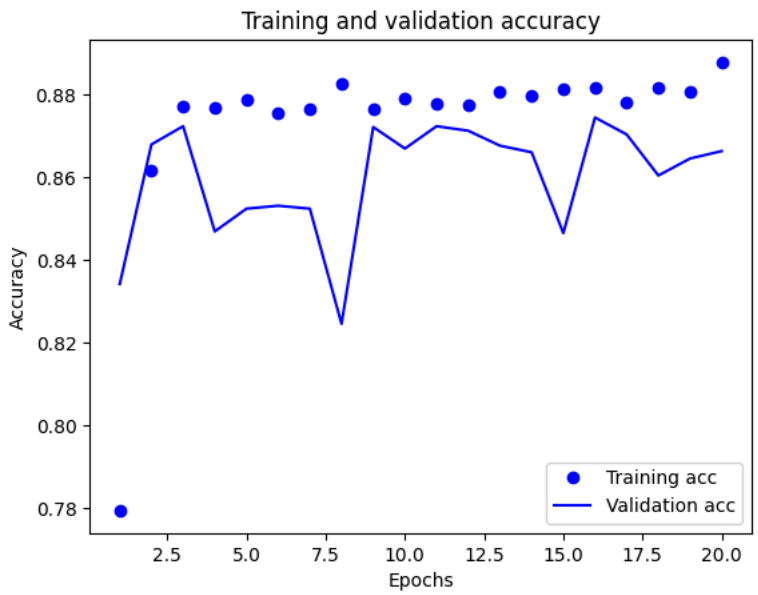
```
import matplotlib.pyplot as plt
history_dict = history.history
loss_values = history_dict["loss"]
val_loss_values = history_dict["val_loss"]
epochs = range(1, len(loss_values) + 1)
plt.plot(epochs, loss_values, "bo", label="Training loss")
plt.plot(epochs, val_loss_values, "b", label="Validation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



Retraining a model from scratch

```
model = keras.Sequential([
    layers.Dense(16, activation="relu"),
    layers.Dense(1, activation="sigmoid")
])
model.compile(optimizer="rmsprop",
              loss="mse",
              metrics=["accuracy"])
model.fit(x_train, y_train, epochs=4, batch_size=512)
results_test = model.evaluate(x_test, y_test)
```

```
Epoch 1/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 25ms/step - accuracy: 0.7371 - loss: 0.1955
Epoch 2/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 3s 25ms/step - accuracy: 0.8860 - loss: 0.1055
Epoch 3/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - accuracy: 0.9068 - loss: 0.0816
Epoch 4/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 23ms/step - accuracy: 0.9210 - loss: 0.0703
782/782 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - accuracy: 0.8854 - loss: 0.0882
```

```
results_test
```

```
[0.08774887770414352, 0.8858799934387207]
```

```
model.fit(x_train, y_train, epochs=4, batch_size=512)
results_val = model.evaluate(x_test, y_test)
```

```
Epoch 1/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - accuracy: 0.9290 - loss: 0.0627
Epoch 2/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - accuracy: 0.9375 - loss: 0.0571
Epoch 3/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 23ms/step - accuracy: 0.9421 - loss: 0.0536
Epoch 4/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - accuracy: 0.9433 - loss: 0.0510
782/782 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.8836 - loss: 0.0853
```

```
results_val
```

```
[0.08411918580532074, 0.8862400054931641]
```

Using a trained model to generate predictions on new data

```
model.predict(x_test)
```

```
782/782 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step
array([[0.17853478],
       [0.9993706 ],
       [0.8565252 ],
       ...,
       [0.1577102 ],
       [0.11842143],
       [0.5935936 ]], dtype=float32)
```

Model 7

```
from tensorflow import keras
from tensorflow.keras import layers

model = keras.Sequential([
    layers.Dense(32, activation="tanh"),
    layers.Dense(32, activation="tanh"),
    layers.Dense(32, activation="tanh"),
    layers.Dropout(0.5),
    layers.Dense(1, activation="sigmoid")
])
```

Compiling the model

```
model.compile(optimizer="rmsprop",
              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 the model

```
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 83ms/step - accuracy: 0.6915 - loss: 0.5648 - val_accuracy: 0.8772 - val_loss: 0.3179
Epoch 2/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 65ms/step - accuracy: 0.9006 - loss: 0.2718 - val_accuracy: 0.8817 - val_loss: 0.2914
Epoch 3/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 55ms/step - accuracy: 0.9225 - loss: 0.2028 - val_accuracy: 0.8772 - val_loss: 0.2974
Epoch 4/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 43ms/step - accuracy: 0.9453 - loss: 0.1534 - val_accuracy: 0.8847 - val_loss: 0.3072
Epoch 5/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 57ms/step - accuracy: 0.9601 - loss: 0.1203 - val_accuracy: 0.8573 - val_loss: 0.4421
Epoch 6/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 54ms/step - accuracy: 0.9634 - loss: 0.1060 - val_accuracy: 0.8763 - val_loss: 0.3590
Epoch 7/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 55ms/step - accuracy: 0.9789 - loss: 0.0734 - val_accuracy: 0.8722 - val_loss: 0.4070
Epoch 8/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 42ms/step - accuracy: 0.9792 - loss: 0.0707 - val_accuracy: 0.8749 - val_loss: 0.4464
Epoch 9/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 71ms/step - accuracy: 0.9794 - loss: 0.0709 - val_accuracy: 0.8734 - val_loss: 0.4753
Epoch 10/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 52ms/step - accuracy: 0.9867 - loss: 0.0505 - val_accuracy: 0.8688 - val_loss: 0.5129
Epoch 11/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 45ms/step - accuracy: 0.9887 - loss: 0.0413 - val_accuracy: 0.8669 - val_loss: 0.5488
Epoch 12/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 42ms/step - accuracy: 0.9955 - loss: 0.0251 - val_accuracy: 0.8696 - val_loss: 0.5721
Epoch 13/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 55ms/step - accuracy: 0.9987 - loss: 0.0141 - val_accuracy: 0.8074 - val_loss: 1.0479
```

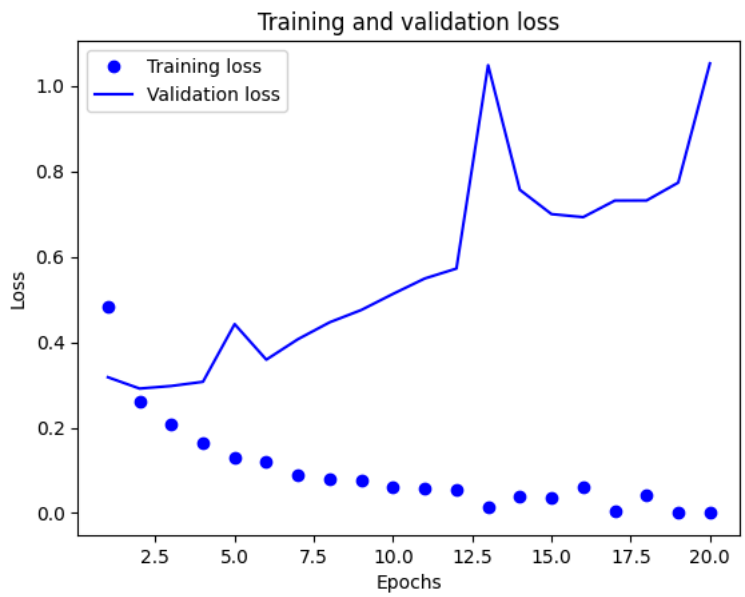
```
Epoch 14/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 45ms/step - accuracy: 0.9692 - loss: 0.0958 - val_accuracy: 0.8483 - val_loss: 0.7564
Epoch 15/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 68ms/step - accuracy: 0.9804 - loss: 0.0644 - val_accuracy: 0.8625 - val_loss: 0.6995
Epoch 16/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 56ms/step - accuracy: 0.9974 - loss: 0.0130 - val_accuracy: 0.8683 - val_loss: 0.6924
Epoch 17/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 62ms/step - accuracy: 0.9997 - loss: 0.0049 - val_accuracy: 0.8666 - val_loss: 0.7312
Epoch 18/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 53ms/step - accuracy: 0.9973 - loss: 0.0117 - val_accuracy: 0.8655 - val_loss: 0.7313
Epoch 19/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 42ms/step - accuracy: 1.0000 - loss: 0.0029 - val_accuracy: 0.8660 - val_loss: 0.7732
Epoch 20/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 54ms/step - accuracy: 1.0000 - loss: 0.0017 - val_accuracy: 0.8451 - val_loss: 1.0522
```

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

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

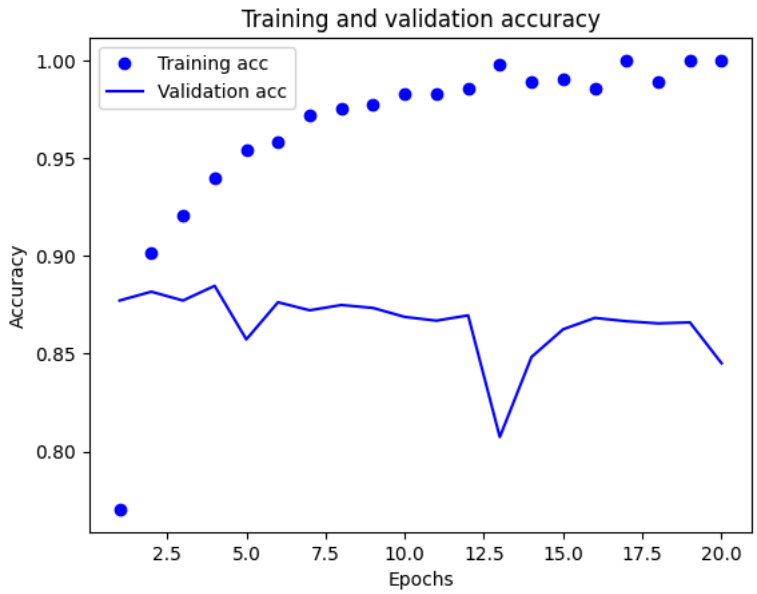
Plotting the training and validation loss

```
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()
```



Retraining a model from scratch

```
model = keras.Sequential([
    layers.Dense(16, activation="tanh"),
    layers.Dense(1, activation="sigmoid")
])
model.compile(optimizer="rmsprop",
              loss="mse",
              metrics=["accuracy"])
model.fit(x_train, y_train, epochs=4, batch_size=512)
results_test = model.evaluate(x_test, y_test)
```

```
Epoch 1/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 30ms/step - accuracy: 0.7315 - loss: 0.1896
Epoch 2/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 35ms/step - accuracy: 0.8957 - loss: 0.0983
Epoch 3/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 28ms/step - accuracy: 0.9138 - loss: 0.0769
Epoch 4/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 29ms/step - accuracy: 0.9230 - loss: 0.0660
782/782 ━━━━━━━━━━━━━━━━━━━━ 3s 4ms/step - accuracy: 0.8846 - loss: 0.0870
```

```
results_test
```

```
[0.08682823181152344, 0.8843200206756592]
```

```
model.fit(x_train, y_train, epochs=4, batch_size=512)
results_val = model.evaluate(x_test, y_test)
```

```
Epoch 1/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 34ms/step - accuracy: 0.9338 - loss: 0.0590
Epoch 2/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - accuracy: 0.9410 - loss: 0.0517
Epoch 3/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 29ms/step - accuracy: 0.9471 - loss: 0.0479
Epoch 4/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - accuracy: 0.9514 - loss: 0.0447
782/782 ━━━━━━━━━━━━━━━━━━━━ 3s 4ms/step - accuracy: 0.8824 - loss: 0.0865
```

```
results_val
```

```
[0.08560524135828018, 0.8847200274467468]
```

Using a trained model to generate predictions on new data

```
model.predict(x_test)
```

```
782/782 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step
```

```
array([[0.11478019],
       [0.9995104 ],
       [0.8290724 ],
       ...,
       [0.1266484 ],
       [0.07466183],
       [0.6199398 ]], dtype=float32)
```

Model 8

```
# creating the model with regularization (L2)
from tensorflow import keras
from tensorflow.keras import layers, regularizers

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

Compiling the model

```
model.compile(optimizer="rmsprop",
              loss="mse",
              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 the model

```
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 92ms/step - accuracy: 0.6664 - loss: 0.2103 - val_accuracy: 0.8640 - val_loss: 0.1154
Epoch 2/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 5s 80ms/step - accuracy: 0.8676 - loss: 0.1061 - val_accuracy: 0.8832 - val_loss: 0.0914
Epoch 3/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 79ms/step - accuracy: 0.9087 - loss: 0.0755 - val_accuracy: 0.8624 - val_loss: 0.1000
Epoch 4/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 66ms/step - accuracy: 0.9195 - loss: 0.0647 - val_accuracy: 0.8856 - val_loss: 0.0840
Epoch 5/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 66ms/step - accuracy: 0.9349 - loss: 0.0538 - val_accuracy: 0.8836 - val_loss: 0.0835
Epoch 6/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 66ms/step - accuracy: 0.9467 - loss: 0.0453 - val_accuracy: 0.8855 - val_loss: 0.0844
Epoch 7/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 61ms/step - accuracy: 0.9592 - loss: 0.0381 - val_accuracy: 0.8825 - val_loss: 0.0854
Epoch 8/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 4s 94ms/step - accuracy: 0.9652 - loss: 0.0324 - val_accuracy: 0.8624 - val_loss: 0.1036
Epoch 9/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 4s 59ms/step - accuracy: 0.9554 - loss: 0.0361 - val_accuracy: 0.8721 - val_loss: 0.0958
Epoch 10/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 67ms/step - accuracy: 0.9682 - loss: 0.0279 - val_accuracy: 0.8786 - val_loss: 0.0906
Epoch 11/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 67ms/step - accuracy: 0.9720 - loss: 0.0258 - val_accuracy: 0.8804 - val_loss: 0.0920
Epoch 12/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 4s 107ms/step - accuracy: 0.9818 - loss: 0.0184 - val_accuracy: 0.8796 - val_loss: 0.0911
Epoch 13/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 4s 65ms/step - accuracy: 0.9837 - loss: 0.0170 - val_accuracy: 0.8775 - val_loss: 0.0939
Epoch 14/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 71ms/step - accuracy: 0.9829 - loss: 0.0172 - val_accuracy: 0.8789 - val_loss: 0.0933
Epoch 15/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 66ms/step - accuracy: 0.9840 - loss: 0.0155 - val_accuracy: 0.8795 - val_loss: 0.0946
```

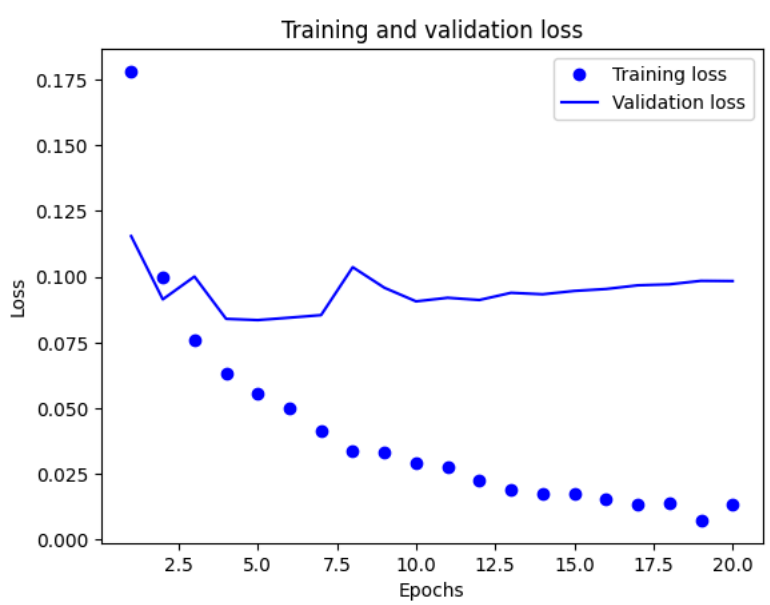
```
Epoch 16/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 98ms/step - accuracy: 0.9890 - loss: 0.0116 - val_accuracy: 0.8784 - val_loss: 0.0953
Epoch 17/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 4s 64ms/step - accuracy: 0.9920 - loss: 0.0092 - val_accuracy: 0.8775 - val_loss: 0.0967
Epoch 18/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 64ms/step - accuracy: 0.9929 - loss: 0.0084 - val_accuracy: 0.8775 - val_loss: 0.0971
Epoch 19/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 67ms/step - accuracy: 0.9942 - loss: 0.0071 - val_accuracy: 0.8766 - val_loss: 0.0984
Epoch 20/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 4s 111ms/step - accuracy: 0.9889 - loss: 0.0105 - val_accuracy: 0.8776 - val_loss: 0.0983
```

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

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

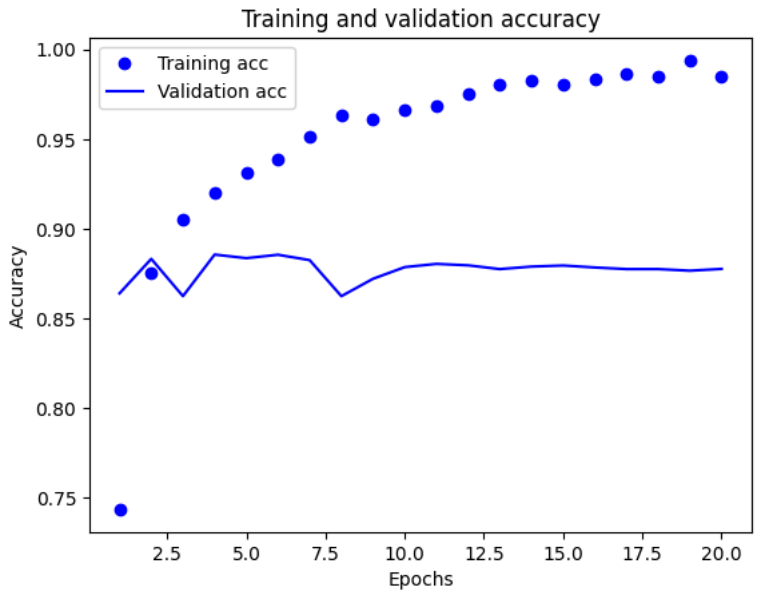
Plotting the training and validation loss

```
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()
```



Retraining a model from scratch

```
model = keras.Sequential([
    layers.Dense(16, activation="relu",kernel_regularizer=regularizers.l2(0.01) ),
    layers.Dense(1, activation="sigmoid")
])
model.compile(optimizer="rmsprop",
              loss="mse",
              metrics=["accuracy"])
model.fit(x_train, y_train, epochs=4, batch_size=512)
results_test = model.evaluate(x_test, y_test)
```

```
Epoch 1/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 27ms/step - accuracy: 0.7243 - loss: 0.3482
Epoch 2/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - accuracy: 0.8674 - loss: 0.1573
Epoch 3/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - accuracy: 0.8678 - loss: 0.1491
Epoch 4/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - accuracy: 0.8680 - loss: 0.1455
782/782 ━━━━━━━━━━━━━━━━━━━━ 3s 4ms/step - accuracy: 0.8615 - loss: 0.1490
```

```
results_test
```

```
[0.1488877832889557, 0.8609200119972229]
```

```
model.fit(x_train, y_train, epochs=4, batch_size=512)
results_val = model.evaluate(x_test, y_test)
```

```
Epoch 1/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - accuracy: 0.8746 - loss: 0.1414
Epoch 2/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - accuracy: 0.8773 - loss: 0.1384
Epoch 3/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - accuracy: 0.8792 - loss: 0.1365
Epoch 4/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - accuracy: 0.8694 - loss: 0.1380
782/782 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - accuracy: 0.8671 - loss: 0.1407
```

```
results_val
```

```
[0.1401102989912033, 0.8674399852752686]
```

Using a trained model to generate predictions on new data

```
model.predict(x_test)
```

```
782/782 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step
```

```
array([[0.39365995],
       [0.9412366 ],
       [0.6159313 ],
       ...,
       [0.22851022],
       [0.25822031],
       [0.39816388]], dtype=float32)
```

Model 9

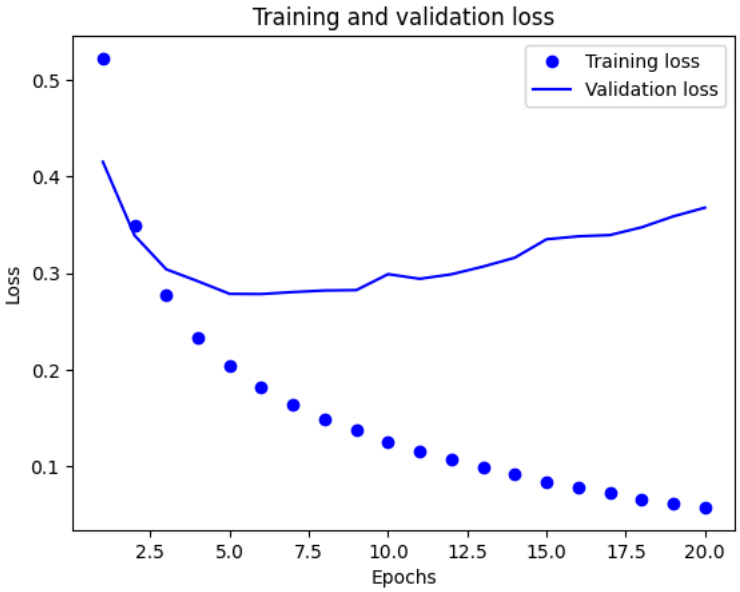
```
from tensorflow import keras
from tensorflow.keras import layers

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


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


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))
```

```
Epoch 1/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 78ms/step - accuracy: 0.7046 - loss: 0.5896 - val_accuracy: 0.8633 - val_loss: 0.4151
Epoch 2/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 46ms/step - accuracy: 0.8936 - loss: 0.3639 - val_accuracy: 0.8793 - val_loss: 0.3389
Epoch 3/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 38ms/step - accuracy: 0.9165 - loss: 0.2799 - val_accuracy: 0.8855 - val_loss: 0.3039
Epoch 4/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 48ms/step - accuracy: 0.9270 - loss: 0.2388 - val_accuracy: 0.8856 - val_loss: 0.2915
Epoch 5/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 34ms/step - accuracy: 0.9340 - loss: 0.2020 - val_accuracy: 0.8881 - val_loss: 0.2785
Epoch 6/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - accuracy: 0.9465 - loss: 0.1777 - val_accuracy: 0.8868 - val_loss: 0.2783
Epoch 7/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.9538 - loss: 0.1589 - val_accuracy: 0.8851 - val_loss: 0.2804
Epoch 8/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - accuracy: 0.9577 - loss: 0.1439 - val_accuracy: 0.8849 - val_loss: 0.2820
Epoch 9/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - accuracy: 0.9606 - loss: 0.1326 - val_accuracy: 0.8850 - val_loss: 0.2824
Epoch 10/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 54ms/step - accuracy: 0.9641 - loss: 0.1256 - val_accuracy: 0.8835 - val_loss: 0.2989
Epoch 11/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 34ms/step - accuracy: 0.9680 - loss: 0.1159 - val_accuracy: 0.8830 - val_loss: 0.2941
Epoch 12/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - accuracy: 0.9721 - loss: 0.1059 - val_accuracy: 0.8838 - val_loss: 0.2988
Epoch 13/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.9764 - loss: 0.0931 - val_accuracy: 0.8820 - val_loss: 0.3067
Epoch 14/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - accuracy: 0.9773 - loss: 0.0901 - val_accuracy: 0.8804 - val_loss: 0.3159
Epoch 15/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - accuracy: 0.9824 - loss: 0.0795 - val_accuracy: 0.8750 - val_loss: 0.3350
Epoch 16/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - accuracy: 0.9827 - loss: 0.0786 - val_accuracy: 0.8760 - val_loss: 0.3381
Epoch 17/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - accuracy: 0.9857 - loss: 0.0680 - val_accuracy: 0.8797 - val_loss: 0.3394
Epoch 18/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - accuracy: 0.9869 - loss: 0.0652 - val_accuracy: 0.8787 - val_loss: 0.3474
Epoch 19/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - accuracy: 0.9889 - loss: 0.0617 - val_accuracy: 0.8785 - val_loss: 0.3587
Epoch 20/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 50ms/step - accuracy: 0.9905 - loss: 0.0547 - val_accuracy: 0.8758 - val_loss: 0.3677
```

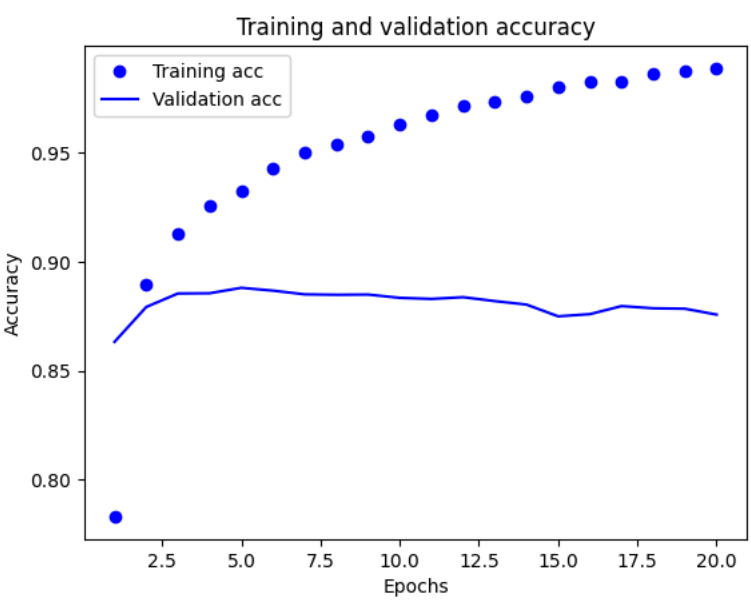
```
history_dict = history.history
history_dict.keys()
```

```
dict_keys(['accuracy', 'loss', 'val_accuracy', 'val_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()
```




```
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()
```



```
model = keras.Sequential([
    layers.Dense(16, activation="relu"),
    layers.Dropout(0.5),
    layers.Dense(1, activation="sigmoid")
])
model.compile(optimizer="rmsprop",
              loss="mse",
              metrics=["accuracy"])
model.fit(x_train, y_train, epochs=4, batch_size=512)
results_test = model.evaluate(x_test, y_test)
```

```
Epoch 1/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 25ms/step - accuracy: 0.6708 - loss: 0.2077
Epoch 2/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - accuracy: 0.8539 - loss: 0.1238
Epoch 3/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 3s 29ms/step - accuracy: 0.8829 - loss: 0.1010
Epoch 4/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - accuracy: 0.8966 - loss: 0.0885
782/782 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - accuracy: 0.8873 - loss: 0.0868
```

```
results_test
```

```
[0.08605113625526428, 0.8882399797439575]
```

```
model.fit(x_train, y_train, epochs=4, batch_size=512)
results_val = model.evaluate(x_test, y_test)
```

```
Epoch 1/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - accuracy: 0.9086 - loss: 0.0774
Epoch 2/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - accuracy: 0.9174 - loss: 0.0709
Epoch 3/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - accuracy: 0.9260 - loss: 0.0649
Epoch 4/4
```