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)
```

```
train_data[0]
```

```
    16,
    6,
    147,
    2025,
```

```
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]])
```

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

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))
```

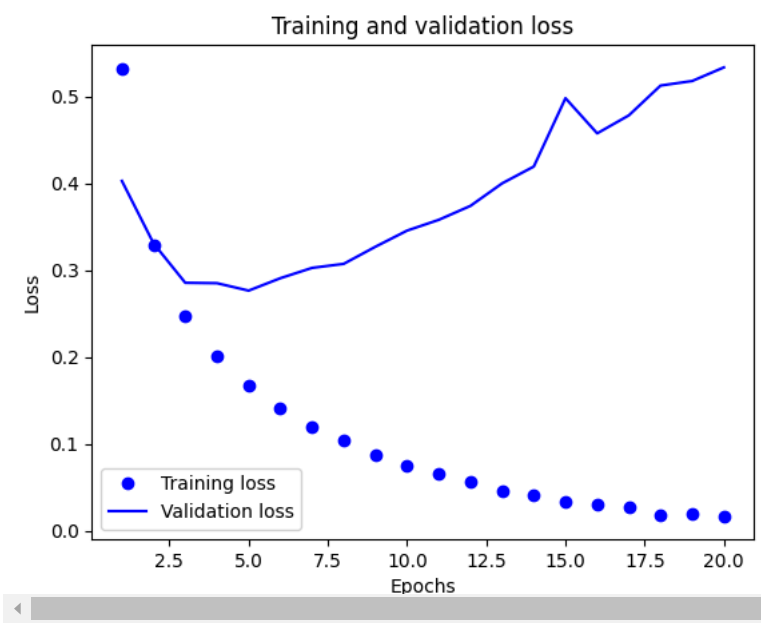
```
Epoch 1/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 5s 91ms/step - accuracy: 0.6659 - loss: 0.6054 - val_accuracy: 0.8649 - val_loss: 0.4024
Epoch 2/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 4s 39ms/step - accuracy: 0.8978 - loss: 0.3405 - val_accuracy: 0.8732 - val_loss: 0.3305
Epoch 3/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - accuracy: 0.9178 - loss: 0.2571 - val_accuracy: 0.8929 - val_loss: 0.2852
Epoch 4/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - accuracy: 0.9394 - loss: 0.1974 - val_accuracy: 0.8870 - val_loss: 0.2848
Epoch 5/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 51ms/step - accuracy: 0.9523 - loss: 0.1596 - val_accuracy: 0.8893 - val_loss: 0.2762
Epoch 6/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 40ms/step - accuracy: 0.9612 - loss: 0.1329 - val_accuracy: 0.8860 - val_loss: 0.2905
Epoch 7/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - accuracy: 0.9683 - loss: 0.1161 - val_accuracy: 0.8802 - val_loss: 0.3024
Epoch 8/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - accuracy: 0.9748 - loss: 0.0957 - val_accuracy: 0.8845 - val_loss: 0.3070
Epoch 9/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - accuracy: 0.9807 - loss: 0.0828 - val_accuracy: 0.8830 - val_loss: 0.3267
Epoch 10/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - accuracy: 0.9793 - loss: 0.0763 - val_accuracy: 0.8803 - val_loss: 0.3453
Epoch 11/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.9859 - loss: 0.0626 - val_accuracy: 0.8776 - val_loss: 0.3577
Epoch 12/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - accuracy: 0.9903 - loss: 0.0490 - val_accuracy: 0.8786 - val_loss: 0.3737
Epoch 13/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - accuracy: 0.9933 - loss: 0.0409 - val_accuracy: 0.8762 - val_loss: 0.3997
Epoch 14/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 51ms/step - accuracy: 0.9923 - loss: 0.0378 - val_accuracy: 0.8756 - val_loss: 0.4191
Epoch 15/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 53ms/step - accuracy: 0.9957 - loss: 0.0307 - val_accuracy: 0.8598 - val_loss: 0.4977
Epoch 16/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 37ms/step - accuracy: 0.9945 - loss: 0.0319 - val_accuracy: 0.8726 - val_loss: 0.4572
Epoch 17/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - accuracy: 0.9974 - loss: 0.0217 - val_accuracy: 0.8729 - val_loss: 0.4779
Epoch 18/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - accuracy: 0.9983 - loss: 0.0177 - val_accuracy: 0.8652 - val_loss: 0.5123
Epoch 19/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.9975 - loss: 0.0167 - val_accuracy: 0.8718 - val_loss: 0.5175
Epoch 20/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - accuracy: 0.9993 - loss: 0.0113 - val_accuracy: 0.8702 - val_loss: 0.5332
```

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

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

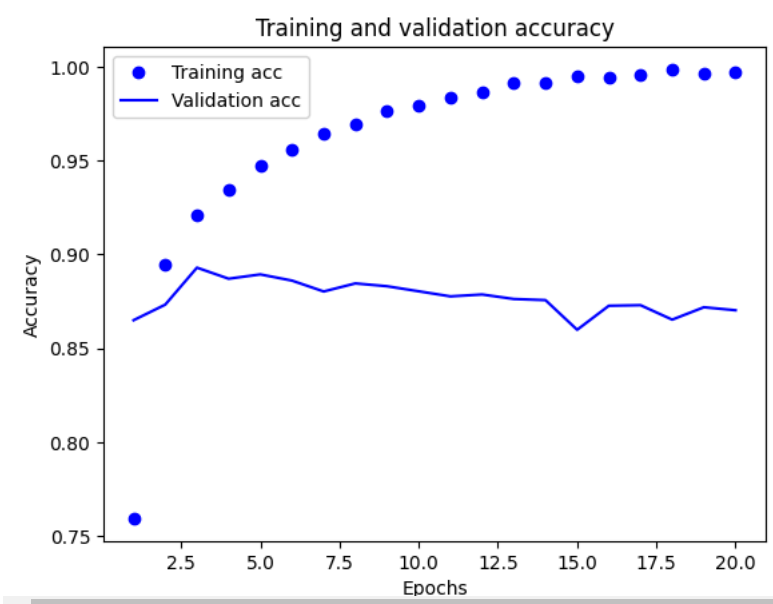
Plotting the training and validation loss

```
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 = model.evaluate(x_test, y_test)
```

```
Epoch 1/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 3s 26ms/step - accuracy: 0.7123 - loss: 0.5771
Epoch 2/4
```

```
49/49 ━━━━━━━━━━━━━━━━━━━━ 3s 31ms/step - accuracy: 0.9005 - loss: 0.3046
Epoch 3/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 27ms/step - accuracy: 0.9274 - loss: 0.2199
Epoch 4/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 3s 31ms/step - accuracy: 0.9321 - loss: 0.1894
782/782 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - accuracy: 0.8850 - loss: 0.2848
```

```
results_test
```

```
[0.2889654040336609, 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 ━━━━━━━━━━━━━━━━━━━━ 3s 33ms/step - accuracy: 0.7204 - loss: 1.1546
Epoch 2/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 3s 51ms/step - accuracy: 0.8782 - loss: 0.5987
Epoch 3/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 4s 36ms/step - accuracy: 0.8815 - loss: 0.5019
Epoch 4/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 32ms/step - accuracy: 0.8948 - loss: 0.4274
782/782 ━━━━━━━━━━━━━━━━━━━━ 3s 4ms/step - accuracy: 0.8751 - loss: 0.4355
```

```
results_val
```

```
[0.4309844970703125, 0.8774399757385254]
```

Using a trained model to generate predictions on new data

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

```
782/782 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step
array([[0.1914265 ],
       [0.99851084],
       [0.7675323 ],
       ...,
       [0.09908374],
       [0.0811417 ],
       [0.51199454]], dtype=float32)
```

Model 2

```
# model2 with two hidden layer
from tensorflow import keras
from tensorflow.keras import layers

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

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 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 88ms/step - accuracy: 0.6669 - loss: 0.6099 - val_accuracy: 0.8690 - val_loss: 0.3829
Epoch 2/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 4s 37ms/step - accuracy: 0.9045 - loss: 0.3189 - val_accuracy: 0.8809 - val_loss: 0.3068
Epoch 3/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.9213 - loss: 0.2278 - val_accuracy: 0.8683 - val_loss: 0.3195
Epoch 4/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - accuracy: 0.9400 - loss: 0.1772 - val_accuracy: 0.8746 - val_loss: 0.3136
Epoch 5/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.9564 - loss: 0.1393 - val_accuracy: 0.8839 - val_loss: 0.2909
Epoch 6/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - accuracy: 0.9678 - loss: 0.1121 - val_accuracy: 0.8545 - val_loss: 0.4023
Epoch 7/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.9696 - loss: 0.0999 - val_accuracy: 0.8792 - val_loss: 0.3194
Epoch 8/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - accuracy: 0.9824 - loss: 0.0691 - val_accuracy: 0.8805 - val_loss: 0.3498
Epoch 9/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 53ms/step - accuracy: 0.9833 - loss: 0.0620 - val_accuracy: 0.8808 - val_loss: 0.3647
Epoch 10/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 49ms/step - accuracy: 0.9904 - loss: 0.0451 - val_accuracy: 0.8757 - val_loss: 0.4072
Epoch 11/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - accuracy: 0.9921 - loss: 0.0370 - val_accuracy: 0.8749 - val_loss: 0.4227
Epoch 12/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - accuracy: 0.9957 - loss: 0.0250 - val_accuracy: 0.8771 - val_loss: 0.4456
Epoch 13/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - accuracy: 0.9974 - loss: 0.0183 - val_accuracy: 0.8658 - val_loss: 0.5147
Epoch 14/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - accuracy: 0.9988 - loss: 0.0145 - val_accuracy: 0.8330 - val_loss: 0.7304
Epoch 15/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - accuracy: 0.9887 - loss: 0.0301 - val_accuracy: 0.8729 - val_loss: 0.5386
Epoch 16/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.9929 - loss: 0.0243 - val_accuracy: 0.8735 - val_loss: 0.5548
Epoch 17/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - accuracy: 0.9999 - loss: 0.0057 - val_accuracy: 0.8741 - val_loss: 0.5830
Epoch 18/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - accuracy: 0.9964 - loss: 0.0148 - val_accuracy: 0.8736 - val_loss: 0.6052
Epoch 19/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 52ms/step - accuracy: 0.9998 - loss: 0.0038 - val_accuracy: 0.8716 - val_loss: 0.6289
Epoch 20/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 51ms/step - accuracy: 0.9984 - loss: 0.0071 - val_accuracy: 0.8720 - val_loss: 0.6580
```

Model 2

```
#model2 with two hidden layer
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"])
```

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))
```

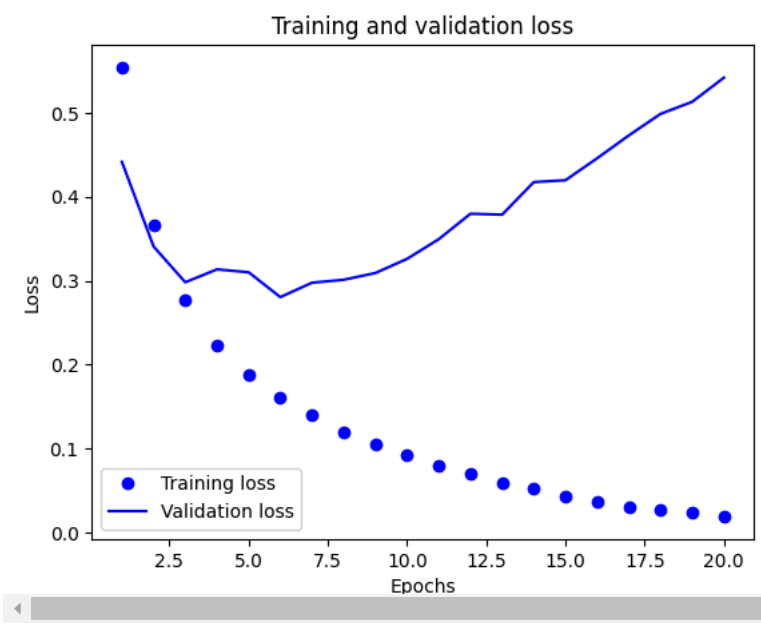
```
Epoch 1/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 4s 77ms/step - accuracy: 0.6822 - loss: 0.6169 - val_accuracy: 0.8496 - val_loss: 0.4416
Epoch 2/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 39ms/step - accuracy: 0.8796 - loss: 0.3860 - val_accuracy: 0.8785 - val_loss: 0.3407
Epoch 3/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - accuracy: 0.9090 - loss: 0.2841 - val_accuracy: 0.8875 - val_loss: 0.2981
Epoch 4/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - accuracy: 0.9275 - loss: 0.2252 - val_accuracy: 0.8705 - val_loss: 0.3135
Epoch 5/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.9370 - loss: 0.1901 - val_accuracy: 0.8733 - val_loss: 0.3100
Epoch 6/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 58ms/step - accuracy: 0.9447 - loss: 0.1649 - val_accuracy: 0.8870 - val_loss: 0.2804
Epoch 7/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 35ms/step - accuracy: 0.9616 - loss: 0.1333 - val_accuracy: 0.8827 - val_loss: 0.2976
Epoch 8/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.9661 - loss: 0.1177 - val_accuracy: 0.8852 - val_loss: 0.3011
Epoch 9/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - accuracy: 0.9680 - loss: 0.1090 - val_accuracy: 0.8837 - val_loss: 0.3091
Epoch 10/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - accuracy: 0.9771 - loss: 0.0861 - val_accuracy: 0.8814 - val_loss: 0.3260
Epoch 11/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - accuracy: 0.9797 - loss: 0.0767 - val_accuracy: 0.8798 - val_loss: 0.3494
Epoch 12/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - accuracy: 0.9836 - loss: 0.0664 - val_accuracy: 0.8723 - val_loss: 0.3797
Epoch 13/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.9867 - loss: 0.0581 - val_accuracy: 0.8794 - val_loss: 0.3787
Epoch 14/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - accuracy: 0.9902 - loss: 0.0470 - val_accuracy: 0.8690 - val_loss: 0.4174
Epoch 15/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 55ms/step - accuracy: 0.9932 - loss: 0.0412 - val_accuracy: 0.8755 - val_loss: 0.4197
Epoch 16/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 33ms/step - accuracy: 0.9948 - loss: 0.0342 - val_accuracy: 0.8726 - val_loss: 0.4459
Epoch 17/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - accuracy: 0.9958 - loss: 0.0290 - val_accuracy: 0.8727 - val_loss: 0.4731
Epoch 18/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - accuracy: 0.9951 - loss: 0.0270 - val_accuracy: 0.8691 - val_loss: 0.4987
Epoch 19/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - accuracy: 0.9961 - loss: 0.0229 - val_accuracy: 0.8697 - val_loss: 0.5131
Epoch 20/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.9987 - loss: 0.0170 - val_accuracy: 0.8684 - val_loss: 0.5419
```

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

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

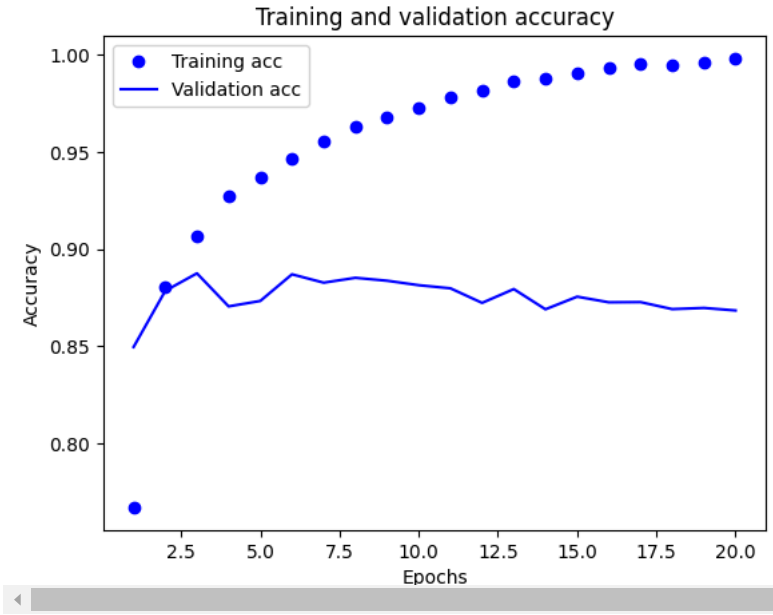
Plotting the training and validation loss

```
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 ━━━━━━━━━━━━━━━━━━━━ 3s 34ms/step - accuracy: 0.7373 - loss: 0.5656
Epoch 2/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 28ms/step - accuracy: 0.8973 - loss: 0.3237
```

```
Epoch 3/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - accuracy: 0.9145 - loss: 0.2559
Epoch 4/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - accuracy: 0.9258 - loss: 0.2197
782/782 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.8842 - loss: 0.2852
```

```
results_test
```

```
[0.28614529967308044, 0.8846399784088135]
```

```
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 32ms/step - accuracy: 0.9368 - loss: 0.1905
Epoch 2/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 40ms/step - accuracy: 0.9433 - loss: 0.1725
Epoch 3/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 25ms/step - accuracy: 0.9479 - loss: 0.1583
Epoch 4/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - accuracy: 0.9491 - loss: 0.1504
782/782 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.8764 - loss: 0.3072
```

```
results_val
```

```
[0.302819162607193, 0.8798400163650513]
```

Using a trained model to generate predictions on new data

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

```
782/782 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step
array([[0.24158992],
       [0.9997136 ],
       [0.84344923],
       ...,
       [0.11291283],
       [0.08470977],
       [0.6601336 ]], dtype=float32)
```

Model 3

```
# build the model with one hidden layer
from tensorflow import keras
from tensorflow.keras import layers

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

Compiling the model

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

Validating 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))
```

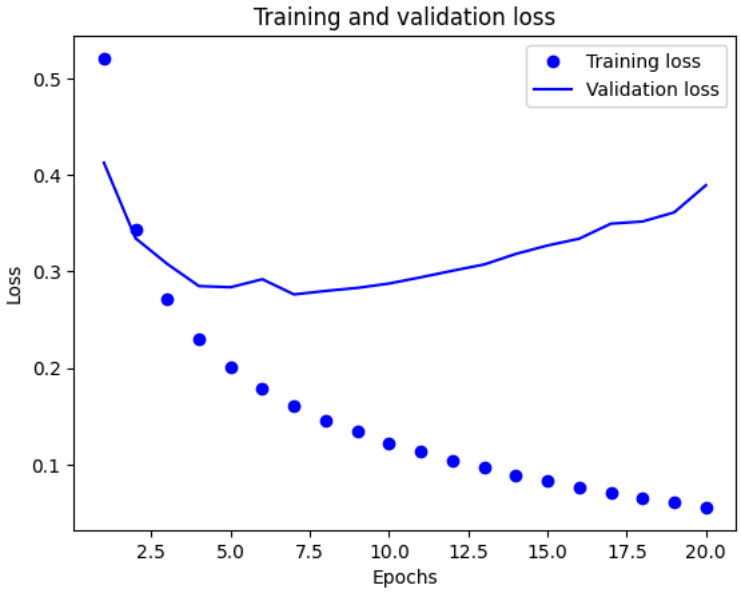
```
Epoch 1/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 74ms/step - accuracy: 0.6951 - loss: 0.5934 - val_accuracy: 0.8567 - val_loss: 0.4128
Epoch 2/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - accuracy: 0.8881 - loss: 0.3607 - val_accuracy: 0.8811 - val_loss: 0.3345
Epoch 3/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - accuracy: 0.9174 - loss: 0.2774 - val_accuracy: 0.8813 - val_loss: 0.3079
Epoch 4/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 52ms/step - accuracy: 0.9254 - loss: 0.2340 - val_accuracy: 0.8914 - val_loss: 0.2849
Epoch 5/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 37ms/step - accuracy: 0.9379 - loss: 0.1995 - val_accuracy: 0.8875 - val_loss: 0.2837
Epoch 6/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.9432 - loss: 0.1760 - val_accuracy: 0.8831 - val_loss: 0.2920
Epoch 7/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 43ms/step - accuracy: 0.9519 - loss: 0.1587 - val_accuracy: 0.8886 - val_loss: 0.2763
Epoch 8/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 59ms/step - accuracy: 0.9575 - loss: 0.1474 - val_accuracy: 0.8863 - val_loss: 0.2799
Epoch 9/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 36ms/step - accuracy: 0.9620 - loss: 0.1310 - val_accuracy: 0.8853 - val_loss: 0.2831
Epoch 10/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.9671 - loss: 0.1218 - val_accuracy: 0.8843 - val_loss: 0.2876
Epoch 11/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.9706 - loss: 0.1119 - val_accuracy: 0.8847 - val_loss: 0.2940
Epoch 12/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 58ms/step - accuracy: 0.9738 - loss: 0.1012 - val_accuracy: 0.8843 - val_loss: 0.3008
Epoch 13/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 45ms/step - accuracy: 0.9756 - loss: 0.0969 - val_accuracy: 0.8840 - val_loss: 0.3073
Epoch 14/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 37ms/step - accuracy: 0.9789 - loss: 0.0863 - val_accuracy: 0.8804 - val_loss: 0.3183
Epoch 15/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - accuracy: 0.9816 - loss: 0.0801 - val_accuracy: 0.8823 - val_loss: 0.3270
Epoch 16/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - accuracy: 0.9835 - loss: 0.0746 - val_accuracy: 0.8790 - val_loss: 0.3340
Epoch 17/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - accuracy: 0.9869 - loss: 0.0660 - val_accuracy: 0.8787 - val_loss: 0.3496
Epoch 18/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - accuracy: 0.9892 - loss: 0.0617 - val_accuracy: 0.8791 - val_loss: 0.3519
Epoch 19/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - accuracy: 0.9897 - loss: 0.0588 - val_accuracy: 0.8767 - val_loss: 0.3613
Epoch 20/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.9895 - loss: 0.0549 - val_accuracy: 0.8694 - val_loss: 0.3895
```

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

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

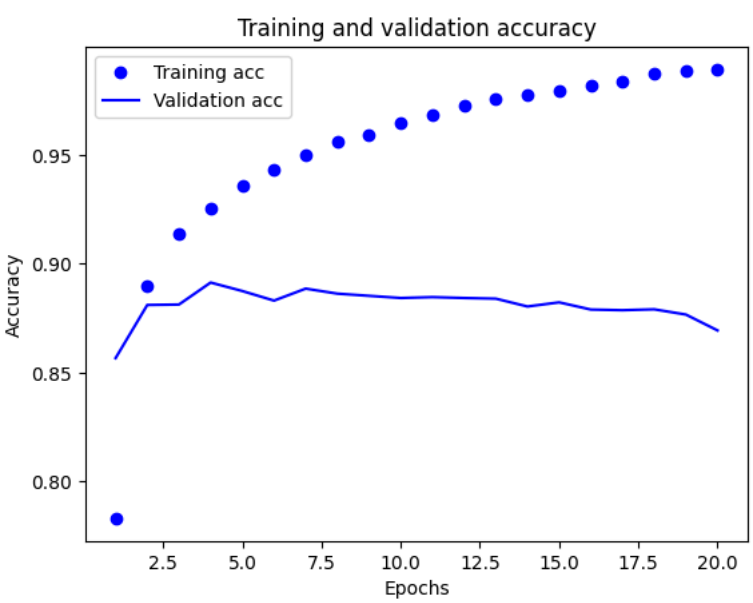
Plotting the training and validation loss

```
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 28ms/step - accuracy: 0.6708 - loss: 0.6048
```

```
Epoch 2/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 26ms/step - accuracy: 0.8963 - loss: 0.3294
Epoch 3/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - accuracy: 0.9264 - loss: 0.2237
Epoch 4/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 3s 26ms/step - accuracy: 0.9378 - loss: 0.1800
782/782 ━━━━━━━━━━━━━━━━━━━━ 3s 4ms/step - accuracy: 0.8780 - loss: 0.3117
```

```
results_test
```

```
[0.31133168935775757, 0.8793200254440308]
```

```
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: 0.9482 - loss: 0.1556
Epoch 2/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 3s 26ms/step - accuracy: 0.9514 - loss: 0.1406
Epoch 3/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 3s 37ms/step - accuracy: 0.9629 - loss: 0.1118
Epoch 4/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 26ms/step - accuracy: 0.9667 - loss: 0.1020
782/782 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.8678 - loss: 0.3888
```

```
results_val
```

```
[0.3852177858352661, 0.8696399927139282]
```

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

```
782/782 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step
array([[0.06419826],
       [0.99869645],
       [0.6238973 ],
       ...,
       [0.12326458],
       [0.0178102 ],
       [0.7137722 ]], dtype=float32)
```

Model 4

```
# model2 with two 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(1, activation="sigmoid")
])
```

Compiling the model

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

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:]
```

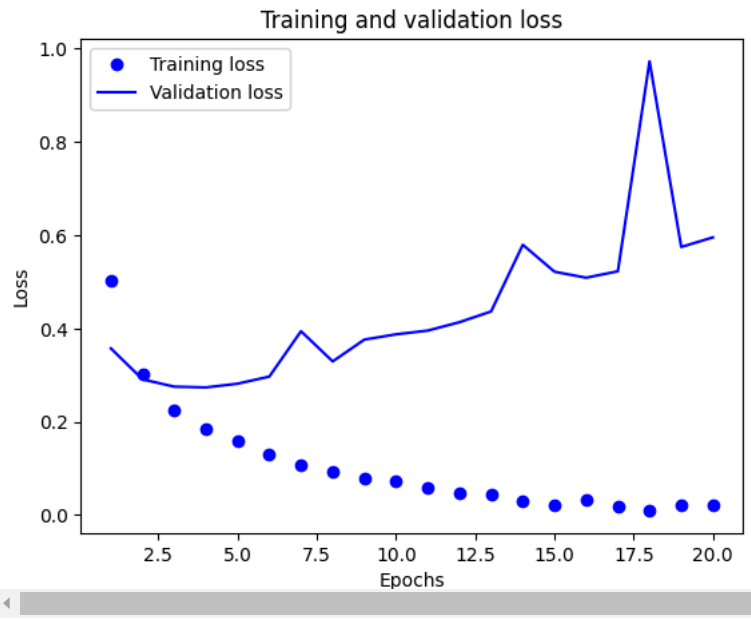
Training the model

```
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: 0.6891 - loss: 0.5859 - val_accuracy: 0.8754 - val_loss: 0.3573
Epoch 2/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 54ms/step - accuracy: 0.8906 - loss: 0.3168 - val_accuracy: 0.8884 - val_loss: 0.2909
Epoch 3/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 43ms/step - accuracy: 0.9218 - loss: 0.2312 - val_accuracy: 0.8916 - val_loss: 0.2753
Epoch 4/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 43ms/step - accuracy: 0.9353 - loss: 0.1898 - val_accuracy: 0.8877 - val_loss: 0.2737
Epoch 5/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 70ms/step - accuracy: 0.9510 - loss: 0.1504 - val_accuracy: 0.8852 - val_loss: 0.2816
Epoch 6/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 58ms/step - accuracy: 0.9633 - loss: 0.1198 - val_accuracy: 0.8823 - val_loss: 0.2969
Epoch 7/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 53ms/step - accuracy: 0.9688 - loss: 0.1006 - val_accuracy: 0.8595 - val_loss: 0.3941
Epoch 8/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 41ms/step - accuracy: 0.9676 - loss: 0.0947 - val_accuracy: 0.8820 - val_loss: 0.3294
Epoch 9/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 41ms/step - accuracy: 0.9791 - loss: 0.0734 - val_accuracy: 0.8698 - val_loss: 0.3761
Epoch 10/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 52ms/step - accuracy: 0.9806 - loss: 0.0696 - val_accuracy: 0.8779 - val_loss: 0.3875
Epoch 11/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 52ms/step - accuracy: 0.9898 - loss: 0.0464 - val_accuracy: 0.8745 - val_loss: 0.3954
Epoch 12/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 68ms/step - accuracy: 0.9938 - loss: 0.0362 - val_accuracy: 0.8783 - val_loss: 0.4134
Epoch 13/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 69ms/step - accuracy: 0.9955 - loss: 0.0292 - val_accuracy: 0.8765 - val_loss: 0.4365
Epoch 14/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 43ms/step - accuracy: 0.9968 - loss: 0.0224 - val_accuracy: 0.8533 - val_loss: 0.5794
Epoch 15/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 53ms/step - accuracy: 0.9959 - loss: 0.0234 - val_accuracy: 0.8652 - val_loss: 0.5217
Epoch 16/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 43ms/step - accuracy: 0.9956 - loss: 0.0233 - val_accuracy: 0.8754 - val_loss: 0.5088
Epoch 17/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 53ms/step - accuracy: 0.9994 - loss: 0.0119 - val_accuracy: 0.8750 - val_loss: 0.5226
Epoch 18/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 54ms/step - accuracy: 0.9996 - loss: 0.0096 - val_accuracy: 0.8160 - val_loss: 0.9726
Epoch 19/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 69ms/step - accuracy: 0.9806 - loss: 0.0525 - val_accuracy: 0.8739 - val_loss: 0.5747
Epoch 20/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 43ms/step - accuracy: 0.9977 - loss: 0.0121 - val_accuracy: 0.8734 - val_loss: 0.5952
```

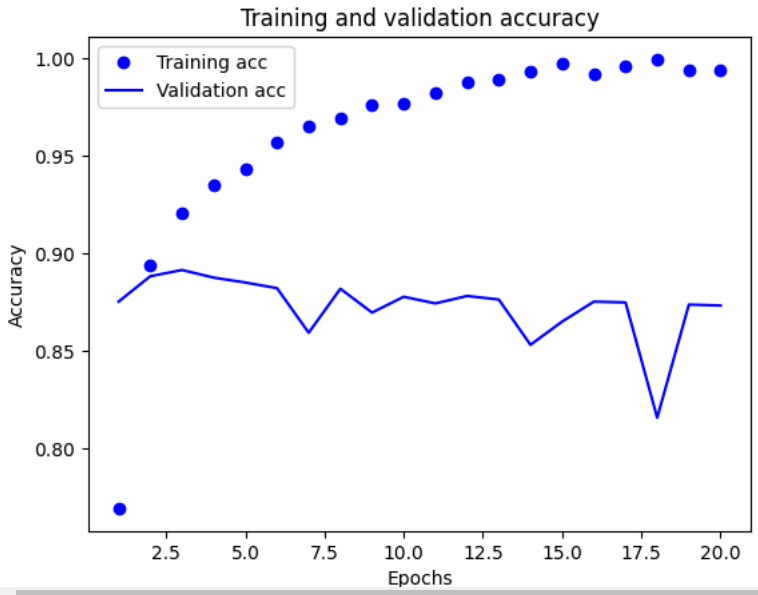
Plotting the training and validation loss

```
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(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 ━━━━━━━━━━━━━━━━━━━━ 3s 33ms/step - accuracy: 0.7365 - loss: 0.5487
Epoch 2/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 32ms/step - accuracy: 0.9025 - loss: 0.2725
Epoch 3/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 3s 39ms/step - accuracy: 0.9279 - loss: 0.2057
Epoch 4/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 4s 63ms/step - accuracy: 0.9371 - loss: 0.1721
782/782 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - accuracy: 0.8811 - loss: 0.2951
```

```
results_test
```

```
[0.2945205569267273, 0.8822399973869324]
```

```
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 ━━━━━━━━━━━━━━━━━━━━ 3s 49ms/step - accuracy: 0.9334 - loss: 0.1729
Epoch 2/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 37ms/step - accuracy: 0.9554 - loss: 0.1312
Epoch 3/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 35ms/step - accuracy: 0.9568 - loss: 0.1207
Epoch 4/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 33ms/step - accuracy: 0.9621 - loss: 0.1041
782/782 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - accuracy: 0.8680 - loss: 0.3747
```

```
results_val
```

```
[0.370725154876709, 0.8707200288772583]
```

Model 5

```
# model2 with two hidden layer
from tensorflow import keras
from tensorflow.keras import layers

model = keras.Sequential([
    layers.Dense(32, activation="relu"),



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

Compiling the model

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

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:]
```

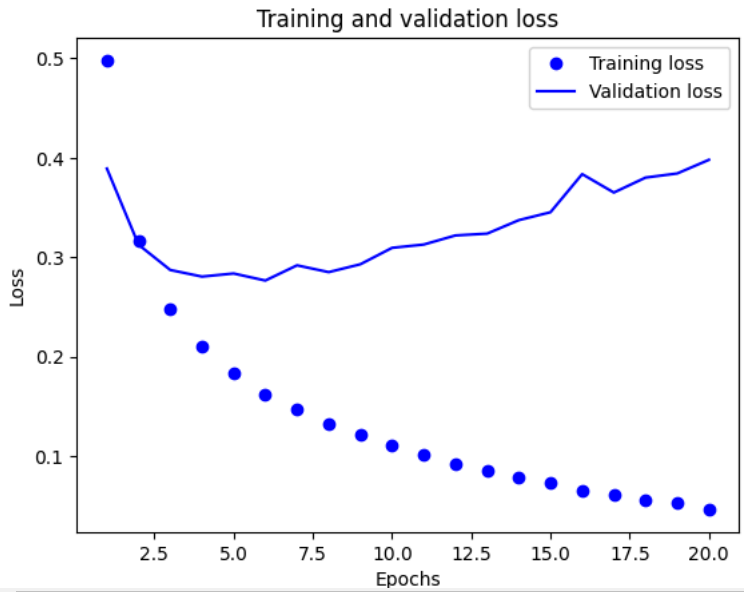
Training the model

```
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 85ms/step - accuracy: 0.6935 - loss: 0.5787 - val_accuracy: 0.8533 - val_loss: 0.3891
Epoch 2/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 53ms/step - accuracy: 0.8919 - loss: 0.3320 - val_accuracy: 0.8853 - val_loss: 0.3125
Epoch 3/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 42ms/step - accuracy: 0.9153 - loss: 0.2562 - val_accuracy: 0.8894 - val_loss: 0.2871
Epoch 4/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 54ms/step - accuracy: 0.9342 - loss: 0.2057 - val_accuracy: 0.8869 - val_loss: 0.2804
Epoch 5/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 67ms/step - accuracy: 0.9407 - loss: 0.1840 - val_accuracy: 0.8869 - val_loss: 0.2835
Epoch 6/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 42ms/step - accuracy: 0.9489 - loss: 0.1600 - val_accuracy: 0.8867 - val_loss: 0.2764
Epoch 7/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 48ms/step - accuracy: 0.9564 - loss: 0.1439 - val_accuracy: 0.8864 - val_loss: 0.2917
Epoch 8/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 51ms/step - accuracy: 0.9620 - loss: 0.1266 - val_accuracy: 0.8867 - val_loss: 0.2849
Epoch 9/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 50ms/step - accuracy: 0.9653 - loss: 0.1188 - val_accuracy: 0.8843 - val_loss: 0.2928
Epoch 10/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 49ms/step - accuracy: 0.9712 - loss: 0.1033 - val_accuracy: 0.8762 - val_loss: 0.3093
Epoch 11/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 59ms/step - accuracy: 0.9745 - loss: 0.0980 - val_accuracy: 0.8781 - val_loss: 0.3127
Epoch 12/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 42ms/step - accuracy: 0.9768 - loss: 0.0898 - val_accuracy: 0.8801 - val_loss: 0.3218
Epoch 13/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 42ms/step - accuracy: 0.9811 - loss: 0.0813 - val_accuracy: 0.8789 - val_loss: 0.3236
Epoch 14/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 41ms/step - accuracy: 0.9818 - loss: 0.0784 - val_accuracy: 0.8791 - val_loss: 0.3372
Epoch 15/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 41ms/step - accuracy: 0.9839 - loss: 0.0726 - val_accuracy: 0.8758 - val_loss: 0.3452
Epoch 16/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 55ms/step - accuracy: 0.9881 - loss: 0.0634 - val_accuracy: 0.8737 - val_loss: 0.3836
Epoch 17/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 66ms/step - accuracy: 0.9865 - loss: 0.0618 - val_accuracy: 0.8742 - val_loss: 0.3650
Epoch 18/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 66ms/step - accuracy: 0.9901 - loss: 0.0535 - val_accuracy: 0.8728 - val_loss: 0.3801
Epoch 19/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 42ms/step - accuracy: 0.9917 - loss: 0.0511 - val_accuracy: 0.8749 - val_loss: 0.3842
Epoch 20/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 41ms/step - accuracy: 0.9931 - loss: 0.0447 - val_accuracy: 0.8737 - val_loss: 0.3979
```

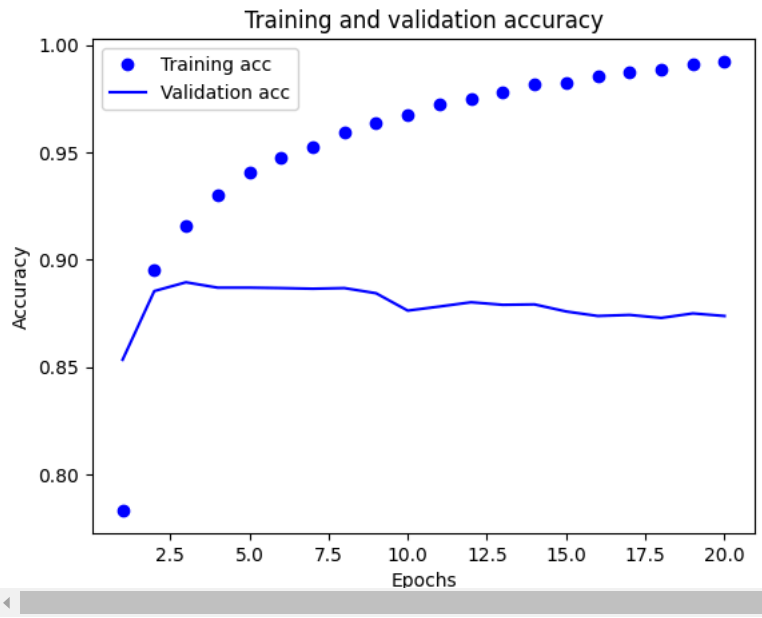
Plotting the training and validation loss

```
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(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 ━━━━━━━━━━━━━━━━━━━━ 3s 32ms/step - accuracy: 0.7227 - loss: 0.5540
Epoch 2/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 3s 34ms/step - accuracy: 0.8999 - loss: 0.2775
Epoch 3/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 3s 54ms/step - accuracy: 0.9259 - loss: 0.2087
Epoch 4/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 34ms/step - accuracy: 0.9415 - loss: 0.1690
782/782 ━━━━━━━━━━━━━━━━━━━━ 3s 4ms/step - accuracy: 0.8830 - loss: 0.2911
```

```
results_test
```

```
[0.2945205569267273, 0.8822399973869324]
```

```
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 31ms/step - accuracy: 0.9449 - loss: 0.1849
Epoch 2/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 31ms/step - accuracy: 0.9556 - loss: 0.1416
Epoch 3/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 4s 51ms/step - accuracy: 0.9629 - loss: 0.1169
Epoch 4/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 33ms/step - accuracy: 0.9668 - loss: 0.1041
782/782 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - accuracy: 0.8589 - loss: 0.4143
```

```
results_val
```

```
[0.4112447202205658, 0.8608800172805786]
```

Model 6

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

model = keras.Sequential([
    layers.Dense(32, activation="tanh", kernel_regularizer=regularizers.l2(0.01)),
    layers.Dense(32, activation="tanh", kernel_regularizer=regularizers.l2(0.01)),
    layers.Dense(1, activation="sigmoid")
])
```

compiling the model

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

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:]
```

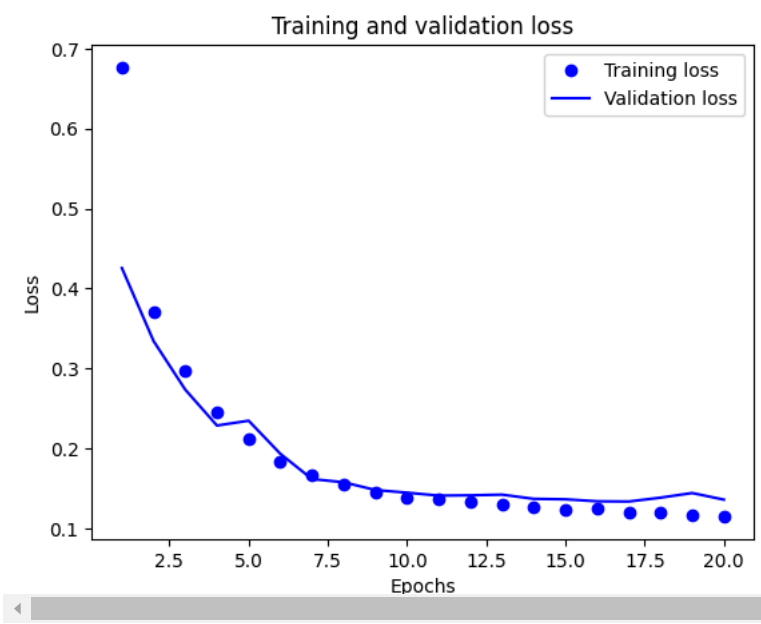
Training the model

```
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 105ms/step - accuracy: 0.6819 - loss: 0.8647 - val_accuracy: 0.8572 - val_loss: 0.4256
Epoch 2/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 4s 57ms/step - accuracy: 0.8600 - loss: 0.3923 - val_accuracy: 0.8373 - val_loss: 0.3342
Epoch 3/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 55ms/step - accuracy: 0.8707 - loss: 0.3069 - val_accuracy: 0.8568 - val_loss: 0.2734
Epoch 4/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 47ms/step - accuracy: 0.8802 - loss: 0.2522 - val_accuracy: 0.8666 - val_loss: 0.2286
Epoch 5/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 54ms/step - accuracy: 0.8748 - loss: 0.2183 - val_accuracy: 0.7916 - val_loss: 0.2346
Epoch 6/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 54ms/step - accuracy: 0.8685 - loss: 0.1920 - val_accuracy: 0.8340 - val_loss: 0.1932
Epoch 7/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 71ms/step - accuracy: 0.8775 - loss: 0.1704 - val_accuracy: 0.8732 - val_loss: 0.1616
Epoch 8/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 43ms/step - accuracy: 0.8793 - loss: 0.1552 - val_accuracy: 0.8629 - val_loss: 0.1576
Epoch 9/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 42ms/step - accuracy: 0.8948 - loss: 0.1422 - val_accuracy: 0.8730 - val_loss: 0.1479
Epoch 10/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 55ms/step - accuracy: 0.8913 - loss: 0.1373 - val_accuracy: 0.8756 - val_loss: 0.1446
Epoch 11/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 53ms/step - accuracy: 0.8981 - loss: 0.1312 - val_accuracy: 0.8784 - val_loss: 0.1409
Epoch 12/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 54ms/step - accuracy: 0.8962 - loss: 0.1312 - val_accuracy: 0.8722 - val_loss: 0.1413
Epoch 13/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 56ms/step - accuracy: 0.8996 - loss: 0.1281 - val_accuracy: 0.8691 - val_loss: 0.1421
Epoch 14/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 74ms/step - accuracy: 0.9049 - loss: 0.1240 - val_accuracy: 0.8779 - val_loss: 0.1368
Epoch 15/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 57ms/step - accuracy: 0.9138 - loss: 0.1198 - val_accuracy: 0.8758 - val_loss: 0.1363
Epoch 16/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 56ms/step - accuracy: 0.9068 - loss: 0.1204 - val_accuracy: 0.8839 - val_loss: 0.1338
Epoch 17/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 55ms/step - accuracy: 0.9174 - loss: 0.1161 - val_accuracy: 0.8800 - val_loss: 0.1335
Epoch 18/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 55ms/step - accuracy: 0.9098 - loss: 0.1182 - val_accuracy: 0.8703 - val_loss: 0.1383
Epoch 19/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 55ms/step - accuracy: 0.9088 - loss: 0.1174 - val_accuracy: 0.8526 - val_loss: 0.1441
Epoch 20/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 69ms/step - accuracy: 0.9146 - loss: 0.1137 - val_accuracy: 0.8715 - val_loss: 0.1358
```

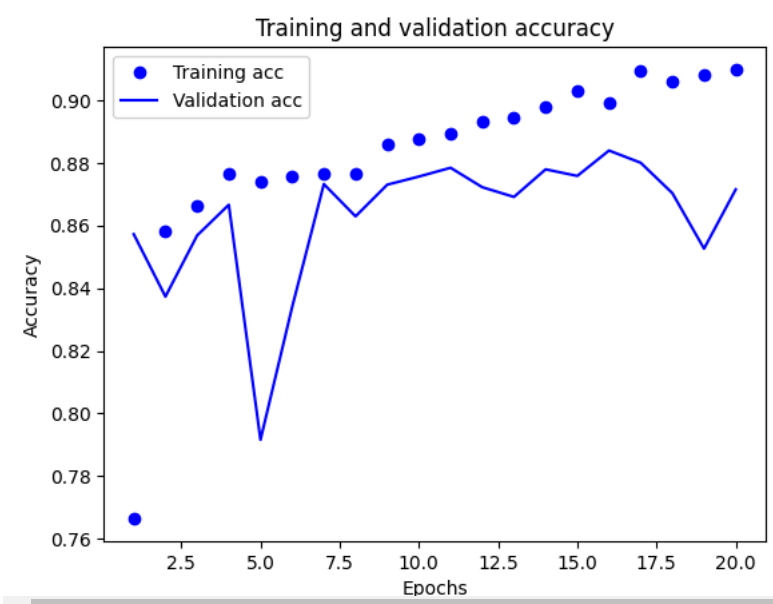
Plotting the training and validation loss

```
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

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

model = keras.Sequential([
    layers.Dense(32, activation="tanh", kernel_regularizer=regularizers.l2(0.01)),
    layers.Dense(32, activation="tanh", kernel_regularizer=regularizers.l2(0.01)),
    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 33ms/step - accuracy: 0.7472 - loss: 1.1504
Epoch 2/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 34ms/step - accuracy: 0.8802 - loss: 0.5987
Epoch 3/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 32ms/step - accuracy: 0.8894 - loss: 0.4860
Epoch 4/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 3s 33ms/step - accuracy: 0.8886 - loss: 0.4318
782/782 ━━━━━━━━━━━━━━━━━━━━ 3s 4ms/step - accuracy: 0.8815 - loss: 0.4235
```

```
results_test
```

```
[0.42044106125831604, 0.8821200132369995]
```

```
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 ━━━━━━━━━━━━━━━━━━━━ 3s 37ms/step - accuracy: 0.8856 - loss: 0.4201
Epoch 2/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 3s 42ms/step - accuracy: 0.8975 - loss: 0.3657
Epoch 3/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 32ms/step - accuracy: 0.9081 - loss: 0.3380
Epoch 4/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 33ms/step - accuracy: 0.9108 - loss: 0.3265
782/782 ━━━━━━━━━━━━━━━━━━━━ 3s 4ms/step - accuracy: 0.8813 - loss: 0.3756
```

```
results_val
```

```
[0.3736857771873474, 0.8839600086212158]
```

Model 7

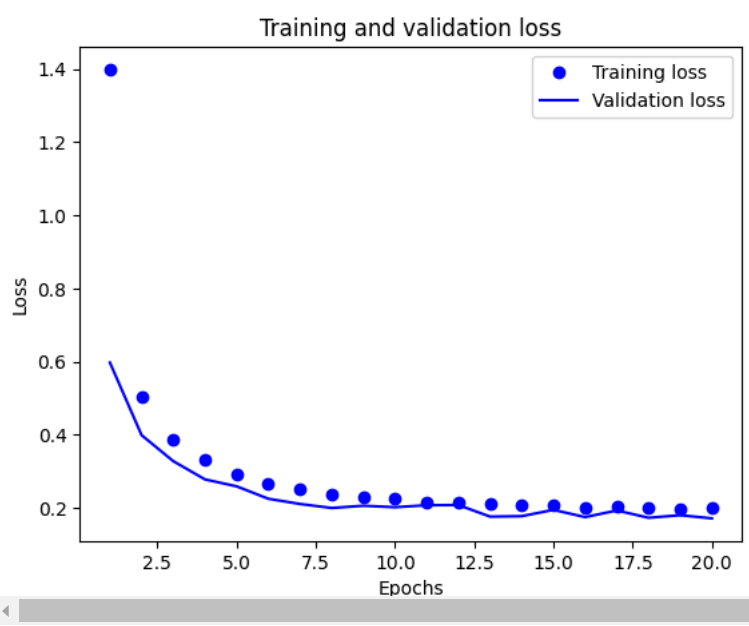
```
from tensorflow import keras
from tensorflow.keras import layers, regularizers

model = keras.Sequential([
    layers.Dense(126, activation="relu", kernel_regularizer=regularizers.l2(0.01)),
    layers.Dropout(0.5),
    layers.Dense(16, activation="relu", kernel_regularizer=regularizers.l2(0.01)),
    layers.Dropout(0.5),
    layers.Dense(16, activation="relu", kernel_regularizer=regularizers.l2(0.01)),
    layers.Dropout(0.5),
    layers.Dense(1, activation="sigmoid")
])

model.compile(optimizer="rmsprop",
              loss="mse",
              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 ━━━━━━━━━━━━━━━━━━━━ 6s 157ms/step - accuracy: 0.5718 - loss: 2.0112 - val_accuracy: 0.8237 - val_loss: 0.5970
Epoch 2/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 5s 154ms/step - accuracy: 0.7122 - loss: 0.5495 - val_accuracy: 0.8413 - val_loss: 0.3986
Epoch 3/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 111ms/step - accuracy: 0.7519 - loss: 0.4038 - val_accuracy: 0.8318 - val_loss: 0.3275
Epoch 4/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 5s 107ms/step - accuracy: 0.7713 - loss: 0.3418 - val_accuracy: 0.8487 - val_loss: 0.2777
Epoch 5/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 6s 132ms/step - accuracy: 0.7841 - loss: 0.2999 - val_accuracy: 0.8133 - val_loss: 0.2586
Epoch 6/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 4s 123ms/step - accuracy: 0.7901 - loss: 0.2702 - val_accuracy: 0.8605 - val_loss: 0.2247
Epoch 7/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 5s 126ms/step - accuracy: 0.8090 - loss: 0.2487 - val_accuracy: 0.8615 - val_loss: 0.2105
Epoch 8/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 5s 121ms/step - accuracy: 0.8056 - loss: 0.2381 - val_accuracy: 0.8648 - val_loss: 0.1995
```

```
Epoch 9/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 5s 124ms/step - accuracy: 0.8100 - loss: 0.2287 - val_accuracy: 0.8317 - val_loss: 0.2056
Epoch 10/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 6s 137ms/step - accuracy: 0.8056 - loss: 0.2253 - val_accuracy: 0.8272 - val_loss: 0.2015
Epoch 11/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 5s 125ms/step - accuracy: 0.8189 - loss: 0.2140 - val_accuracy: 0.8111 - val_loss: 0.2071
Epoch 12/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 6s 168ms/step - accuracy: 0.8074 - loss: 0.2167 - val_accuracy: 0.8006 - val_loss: 0.2075
Epoch 13/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 5s 150ms/step - accuracy: 0.8175 - loss: 0.2118 - val_accuracy: 0.8708 - val_loss: 0.1758
Epoch 14/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 4s 104ms/step - accuracy: 0.8305 - loss: 0.2024 - val_accuracy: 0.8646 - val_loss: 0.1772
Epoch 15/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 6s 141ms/step - accuracy: 0.8316 - loss: 0.2012 - val_accuracy: 0.8276 - val_loss: 0.1944
Epoch 16/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 5s 126ms/step - accuracy: 0.8295 - loss: 0.2011 - val_accuracy: 0.8651 - val_loss: 0.1745
Epoch 17/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 5s 105ms/step - accuracy: 0.8333 - loss: 0.2001 - val_accuracy: 0.8285 - val_loss: 0.1928
Epoch 18/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 6s 124ms/step - accuracy: 0.8317 - loss: 0.1998 - val_accuracy: 0.8660 - val_loss: 0.1727
Epoch 19/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 5s 123ms/step - accuracy: 0.8369 - loss: 0.1933 - val_accuracy: 0.8437 - val_loss: 0.1797
Epoch 20/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 7s 185ms/step - accuracy: 0.8247 - loss: 0.2013 - val_accuracy: 0.8685 - val_loss: 0.1711
```

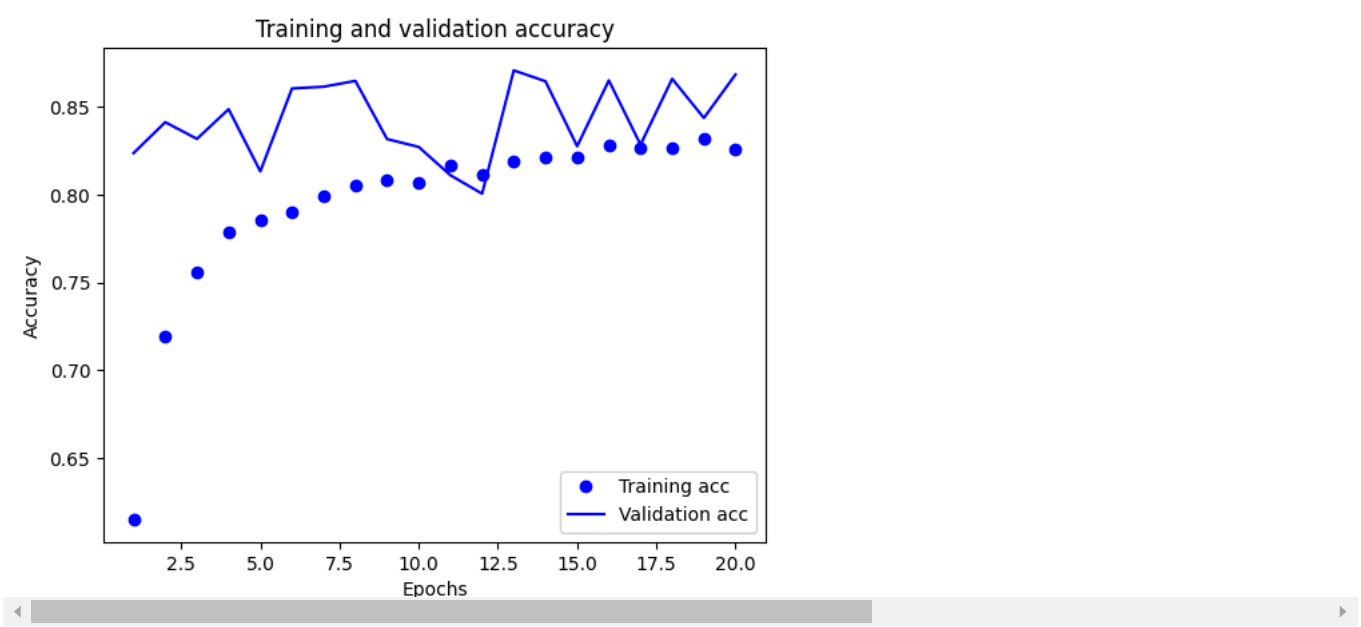
Plotting the training and validation loss

```
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(126, activation="relu", kernel_regularizer=regularizers.l2(0.01)),
    layers.Dropout(0.5),
    layers.Dense(16, activation="relu", kernel_regularizer=regularizers.l2(0.01)),
    layers.Dropout(0.5),
    layers.Dense(16, activation="relu", kernel_regularizer=regularizers.l2(0.01)),
    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 ━━━━━━━━━━━━━━━━━━━━ 7s 109ms/step - accuracy: 0.5463 - loss: 1.6771
Epoch 2/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 10s 95ms/step - accuracy: 0.6983 - loss: 0.4215
Epoch 3/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 5s 90ms/step - accuracy: 0.7456 - loss: 0.3196
Epoch 4/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 4s 82ms/step - accuracy: 0.7720 - loss: 0.2662
782/782 ━━━━━━━━━━━━━━━━━━━━ 7s 9ms/step - accuracy: 0.8133 - loss: 0.2313
```

```
results_test
```

```
[0.22963492572307587, 0.8169599771499634]
```

```
model.compile(optimizer="rmsprop",
              loss="mse",
              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 ━━━━━━━━━━━━━━━━━━━━ 7s 102ms/step - accuracy: 0.7573 - loss: 0.2526
Epoch 2/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 4s 78ms/step - accuracy: 0.7878 - loss: 0.2290
Epoch 3/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 4s 81ms/step - accuracy: 0.7920 - loss: 0.2216
Epoch 4/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 5s 85ms/step - accuracy: 0.8101 - loss: 0.2125
782/782 ━━━━━━━━━━━━━━━━━━━━ 6s 8ms/step - accuracy: 0.8507 - loss: 0.1864
```

```
results_val
```

```
[0.18642626702785492, 0.8517600297927856]
```

Model 8

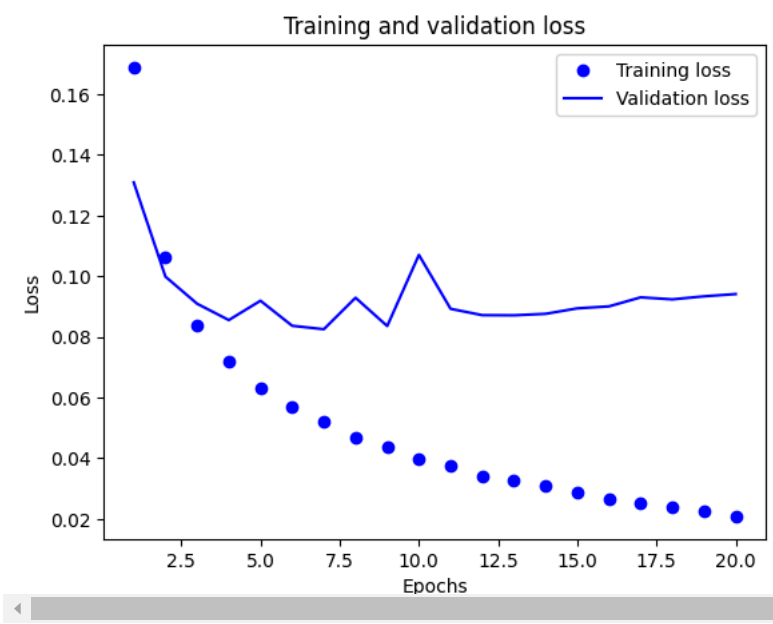
```
from tensorflow import keras
from tensorflow.keras import layers, regularizers

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


model.compile(optimizer="rmsprop",
              loss="mse",
              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 81ms/step - accuracy: 0.7012 - loss: 0.1978 - val_accuracy: 0.8406 - val_loss: 0.1309
Epoch 2/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 43ms/step - accuracy: 0.8765 - loss: 0.1122 - val_accuracy: 0.8794 - val_loss: 0.0999
Epoch 3/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 71ms/step - accuracy: 0.9045 - loss: 0.0857 - val_accuracy: 0.8840 - val_loss: 0.0909
Epoch 4/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 59ms/step - accuracy: 0.9226 - loss: 0.0709 - val_accuracy: 0.8868 - val_loss: 0.0856
Epoch 5/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 54ms/step - accuracy: 0.9293 - loss: 0.0635 - val_accuracy: 0.8749 - val_loss: 0.0919
Epoch 6/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 41ms/step - accuracy: 0.9389 - loss: 0.0568 - val_accuracy: 0.8865 - val_loss: 0.0837
Epoch 7/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 52ms/step - accuracy: 0.9420 - loss: 0.0516 - val_accuracy: 0.8848 - val_loss: 0.0825
Epoch 8/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 55ms/step - accuracy: 0.9494 - loss: 0.0456 - val_accuracy: 0.8734 - val_loss: 0.0929
Epoch 9/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 54ms/step - accuracy: 0.9525 - loss: 0.0436 - val_accuracy: 0.8838 - val_loss: 0.0836
Epoch 10/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 72ms/step - accuracy: 0.9639 - loss: 0.0366 - val_accuracy: 0.8543 - val_loss: 0.1070
Epoch 11/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 44ms/step - accuracy: 0.9608 - loss: 0.0370 - val_accuracy: 0.8758 - val_loss: 0.0893
Epoch 12/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 53ms/step - accuracy: 0.9667 - loss: 0.0329 - val_accuracy: 0.8807 - val_loss: 0.0872
Epoch 13/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 53ms/step - accuracy: 0.9668 - loss: 0.0330 - val_accuracy: 0.8802 - val_loss: 0.0871
Epoch 14/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 54ms/step - accuracy: 0.9711 - loss: 0.0291 - val_accuracy: 0.8801 - val_loss: 0.0876
Epoch 15/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 44ms/step - accuracy: 0.9741 - loss: 0.0271 - val_accuracy: 0.8788 - val_loss: 0.0894
Epoch 16/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 42ms/step - accuracy: 0.9752 - loss: 0.0256 - val_accuracy: 0.8794 - val_loss: 0.0901
Epoch 17/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 54ms/step - accuracy: 0.9800 - loss: 0.0225 - val_accuracy: 0.8789 - val_loss: 0.0930
Epoch 18/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 42ms/step - accuracy: 0.9805 - loss: 0.0223 - val_accuracy: 0.8770 - val_loss: 0.0924
Epoch 19/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 42ms/step - accuracy: 0.9802 - loss: 0.0213 - val_accuracy: 0.8771 - val_loss: 0.0934
Epoch 20/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 45ms/step - accuracy: 0.9831 - loss: 0.0191 - val_accuracy: 0.8766 - val_loss: 0.0941
```

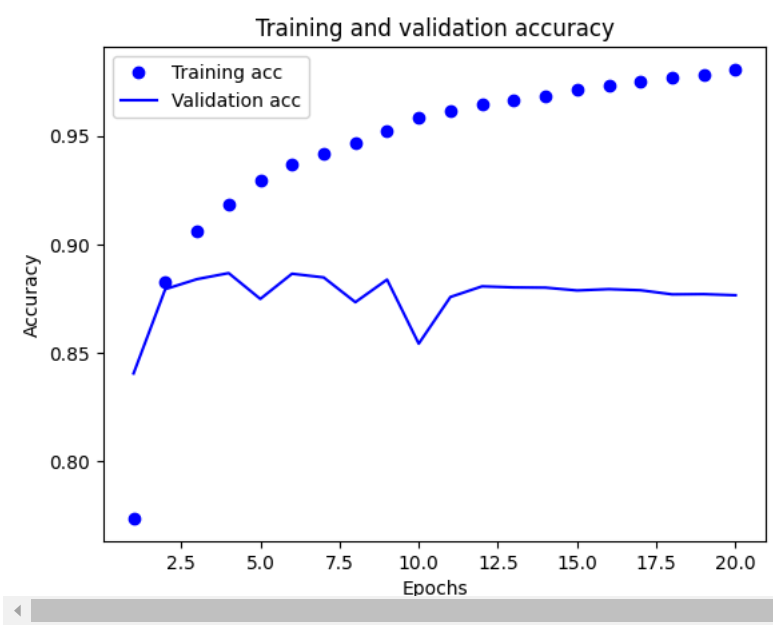
```
history_dict = history.history
history_dict.keys()
```

```
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",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 ━━━━━━━━━━━━━━━━━━━━ 4s 43ms/step - accuracy: 0.7208 - loss: 0.3512
Epoch 2/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step - accuracy: 0.8630 - loss: 0.1588
Epoch 3/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - accuracy: 0.8697 - loss: 0.1497
Epoch 4/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 3s 28ms/step - accuracy: 0.8771 - loss: 0.1444
782/782 ━━━━━━━━━━━━━━━━━━━━ 3s 4ms/step - accuracy: 0.8426 - loss: 0.1554
```

```
results_test
```

```
[0.15385255217552185, 0.8468400239944458]
```

```
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: 0.8719 - loss: 0.1440
Epoch 2/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 3s 26ms/step - accuracy: 0.8728 - loss: 0.1400
Epoch 3/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - accuracy: 0.8730 - loss: 0.1391
Epoch 4/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - accuracy: 0.8756 - loss: 0.1378
782/782 ━━━━━━━━━━━━━━━━━━━━ 3s 4ms/step - accuracy: 0.8587 - loss: 0.1462
```

```
results_val
```

```
[0.1460731476545334, 0.8588799834251404]
```

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

```
782/782 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step
array([[0.37469065],
       [0.9418483 ],
       [0.51682514],
       ...,
       [0.2202851 ],
       [0.20810063],
       [0.40521508]], dtype=float32)
```

Model 9

```
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")
])
```

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

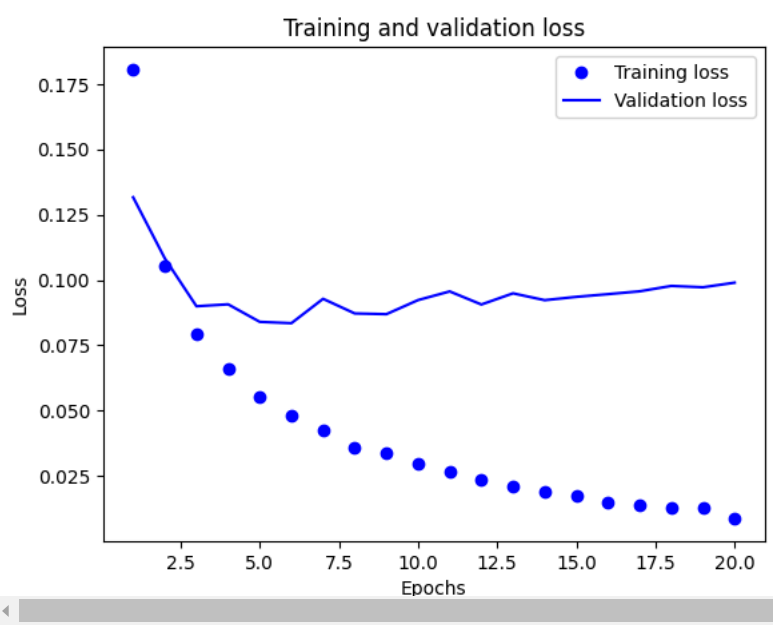
```
Epoch 1/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 5s 84ms/step - accuracy: 0.6932 - loss: 0.2098 - val_accuracy: 0.8603 - val_loss: 0.1316
Epoch 2/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 63ms/step - accuracy: 0.8837 - loss: 0.1114 - val_accuracy: 0.8637 - val_loss: 0.1083
Epoch 3/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 37ms/step - accuracy: 0.9052 - loss: 0.0834 - val_accuracy: 0.8876 - val_loss: 0.0899
Epoch 4/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - accuracy: 0.9251 - loss: 0.0673 - val_accuracy: 0.8782 - val_loss: 0.0907
Epoch 5/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.9391 - loss: 0.0555 - val_accuracy: 0.8860 - val_loss: 0.0840
Epoch 6/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.9515 - loss: 0.0469 - val_accuracy: 0.8858 - val_loss: 0.0835
Epoch 7/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - accuracy: 0.9606 - loss: 0.0392 - val_accuracy: 0.8715 - val_loss: 0.0928
Epoch 8/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - accuracy: 0.9665 - loss: 0.0355 - val_accuracy: 0.8790 - val_loss: 0.0872
Epoch 9/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - accuracy: 0.9695 - loss: 0.0320 - val_accuracy: 0.8819 - val_loss: 0.0869
Epoch 10/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 42ms/step - accuracy: 0.9722 - loss: 0.0292 - val_accuracy: 0.8744 - val_loss: 0.0923
Epoch 11/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 34ms/step - accuracy: 0.9777 - loss: 0.0253 - val_accuracy: 0.8717 - val_loss: 0.0956
```

```
Epoch 12/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.9796 - loss: 0.0238 - val_accuracy: 0.8750 - val_loss: 0.0906
Epoch 13/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.9848 - loss: 0.0192 - val_accuracy: 0.8765 - val_loss: 0.0949
Epoch 14/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.9856 - loss: 0.0184 - val_accuracy: 0.8772 - val_loss: 0.0923
Epoch 15/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - accuracy: 0.9882 - loss: 0.0160 - val_accuracy: 0.8762 - val_loss: 0.0935
Epoch 16/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - accuracy: 0.9895 - loss: 0.0144 - val_accuracy: 0.8754 - val_loss: 0.0946
Epoch 17/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.9900 - loss: 0.0132 - val_accuracy: 0.8767 - val_loss: 0.0957
Epoch 18/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - accuracy: 0.9931 - loss: 0.0104 - val_accuracy: 0.8721 - val_loss: 0.0977
Epoch 19/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 42ms/step - accuracy: 0.9924 - loss: 0.0099 - val_accuracy: 0.8753 - val_loss: 0.0972
Epoch 20/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 59ms/step - accuracy: 0.9933 - loss: 0.0089 - val_accuracy: 0.8728 - val_loss: 0.0990
```

```
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()
```