```
from 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
```

```
import numpy as np
def vectorize_sequences(sequences, dimension=10000):
  results = np.zeros((len(sequences), dimension))
  for i, sequence in enumerate(sequences):
     results[i, sequence] = 1.
  return results
x_train = vectorize_sequences(train_data)
x_test = vectorize_sequences(test_data)
```

```
from keras import models
from keras import layers
model = models.Sequential()
model.add(layers.Dense(16, activation='relu', input_shape=(10000,)))
model.add(layers.Dense(16, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `input_shape`/`
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

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

```
9999
```

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

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

```
from keras import optimizers
model.compile(optimizer=optimizers.RMSprop(learning_rate=0.001),
loss='binary_crossentropy',
metrics=['accuracy'])
```

```
from keras import losses
from keras import metrics
model.compile(optimizer=optimizers.RMSprop(learning_rate=0.001),
loss=losses.binary_crossentropy,
metrics=[metrics.binary_accuracy])
```

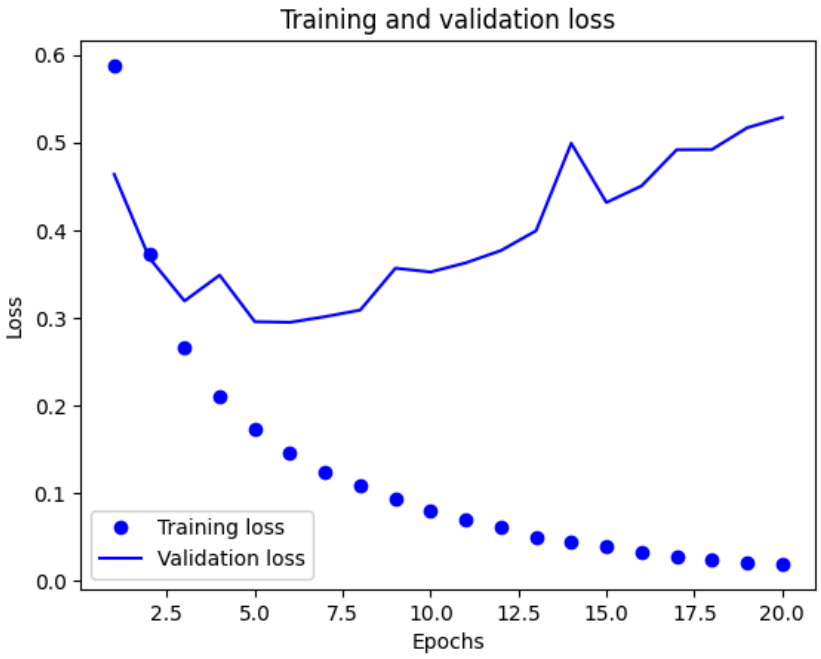
```
x_val = x_train[:10000]
partial_x_train = x_train[10000:]
y_val = y_train[:10000]
partial_y_train = y_train[10000:]
```

```
model.compile(optimizer='rmsprop',
loss='binary_crossentropy',
metrics=['acc'])
history = model.fit(partial_x_train,
partial_y_train,
epochs=20,
batch_size=512,
validation_data=(x_val, y_val))
```

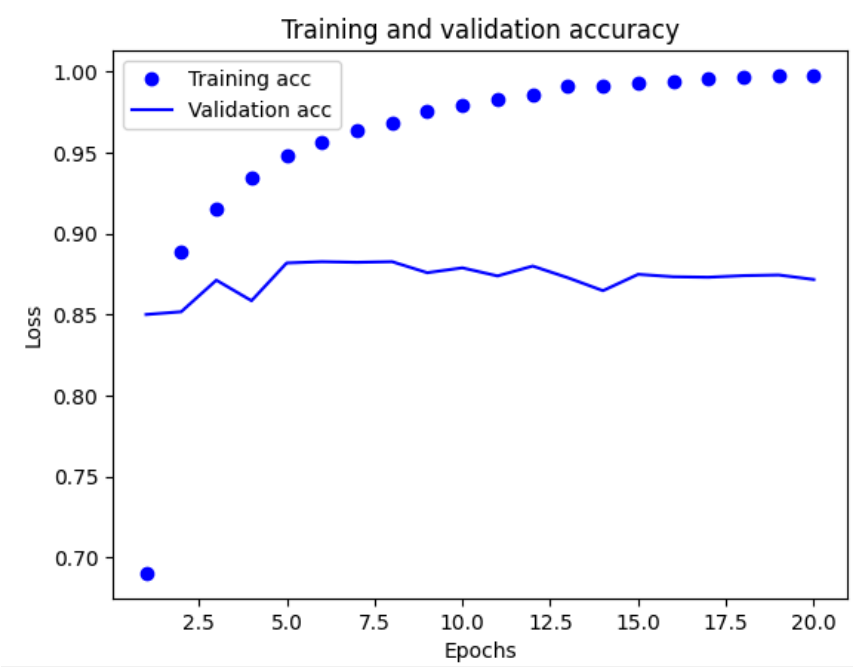
```
Epoch 1/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 4s 101ms/step - acc: 0.5932 - loss: 0.6439 - val_acc: 0.8499 - val_loss: 0.4636
Epoch 2/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 56ms/step - acc: 0.8807 - loss: 0.4032 - val_acc: 0.8516 - val_loss: 0.3683
Epoch 3/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 51ms/step - acc: 0.9142 - loss: 0.2756 - val_acc: 0.8711 - val_loss: 0.3192
Epoch 4/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 77ms/step - acc: 0.9358 - loss: 0.2118 - val_acc: 0.8584 - val_loss: 0.3487
Epoch 5/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 49ms/step - acc: 0.9469 - loss: 0.1775 - val_acc: 0.8817 - val_loss: 0.2956
Epoch 6/20
```

```
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 45ms/step - acc: 0.9572 - loss: 0.1452 - val_acc: 0.8825 - val_loss: 0.2949
Epoch 7/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 57ms/step - acc: 0.9652 - loss: 0.1242 - val_acc: 0.8821 - val_loss: 0.3013
Epoch 8/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 47ms/step - acc: 0.9707 - loss: 0.1065 - val_acc: 0.8825 - val_loss: 0.3089
Epoch 9/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 69ms/step - acc: 0.9791 - loss: 0.0866 - val_acc: 0.8757 - val_loss: 0.3566
Epoch 10/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 57ms/step - acc: 0.9801 - loss: 0.0805 - val_acc: 0.8787 - val_loss: 0.3523
Epoch 11/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 55ms/step - acc: 0.9842 - loss: 0.0687 - val_acc: 0.8737 - val_loss: 0.3627
Epoch 12/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 49ms/step - acc: 0.9873 - loss: 0.0575 - val_acc: 0.8798 - val_loss: 0.3767
Epoch 13/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 47ms/step - acc: 0.9929 - loss: 0.0439 - val_acc: 0.8726 - val_loss: 0.3993
Epoch 14/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 66ms/step - acc: 0.9918 - loss: 0.0454 - val_acc: 0.8646 - val_loss: 0.4990
Epoch 15/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 70ms/step - acc: 0.9911 - loss: 0.0429 - val_acc: 0.8747 - val_loss: 0.4315
Epoch 16/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 46ms/step - acc: 0.9957 - loss: 0.0295 - val_acc: 0.8732 - val_loss: 0.4504
Epoch 17/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 46ms/step - acc: 0.9969 - loss: 0.0239 - val_acc: 0.8729 - val_loss: 0.4916
Epoch 18/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 47ms/step - acc: 0.9968 - loss: 0.0242 - val_acc: 0.8739 - val_loss: 0.4918
Epoch 19/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 48ms/step - acc: 0.9971 - loss: 0.0210 - val_acc: 0.8743 - val_loss: 0.5166
Epoch 20/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 60ms/step - acc: 0.9969 - loss: 0.0195 - val_acc: 0.8715 - val_loss: 0.5284
```

```
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(history_dict['acc']) + 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_values = history_dict['acc']
val_acc_values = history_dict['val_acc']
plt.plot(epochs, acc_values, 'bo', label='Training acc')
plt.plot(epochs, val_acc_values, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```




```
model.compile(optimizer='SGD',
              loss='binary_crossentropy',
              metrics=['accuracy'])

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

```
Epoch 1/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 32ms/step - accuracy: 0.9483 - loss: 0.2192
Epoch 2/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 3s 31ms/step - accuracy: 0.9494 - loss: 0.2009
Epoch 3/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 26ms/step - accuracy: 0.9493 - loss: 0.1864
Epoch 4/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 3s 39ms/step - accuracy: 0.9498 - loss: 0.1711
782/782 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.8549 - loss: 0.4453
```

```
model = models.Sequential()
model.add(layers.Dense(16, activation='relu', input_shape=(10000,)))
model.add(layers.Dense(16, activation='relu'))
model.add(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 ━━━━━━━━━━━━━━━━━━━━ 4s 50ms/step - accuracy: 0.7378 - loss: 0.5525
Epoch 2/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 4s 32ms/step - accuracy: 0.8994 - loss: 0.2842
Epoch 3/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 3s 33ms/step - accuracy: 0.9266 - loss: 0.2115
Epoch 4/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 35ms/step - accuracy: 0.9401 - loss: 0.1757
782/782 ━━━━━━━━━━━━━━━━━━━━ 3s 4ms/step - accuracy: 0.8828 - loss: 0.2889
```