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
17465344/17464789 [==============================] - 0s 0us/step
17473536/17464789 [==============================] - 0s 0us/step

word_index = imdb.get_word_index()
reverse_word_index = dict([(value, key) for (key, value) in word_index.items()])

rn=0
decoded_review = ' '.join([reverse_word_index.get(i - 3, '?') for i in train_data[rn]])
print(train_labels[rn])
decoded_review

1
' this film was just brilliant casting location scenery story direction everyone's really suited the part they played and you could just imagine being there rob...rt ? is an amazing actor and now the same being director ? father came from the same scottish island as myself so i loved the fact there was a real connection with this film the witty remarks throughout the film were great it was just brilliant so much that i bought the film as soon as it was released for ? and would recommend it to everyone to watch and the fly fishing was amazing really cried at the end it

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

from tensorflow import keras
from tensorflow.keras import models
from tensorflow.keras import layers

model = keras.Sequential([
layers.Dense(5, activation='relu'),
layers.Dense(5, activation='relu'),
})
```python
layers.Dense(5, activation='relu'),
  layers.Dense(5, 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:]

model.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['acc'])
history = model.fit(partial_x_train, partial_y_train, epochs=14, batch_size=256, validation_data=(x_val, y_val))
results = model.evaluate(x_test, y_test)

results
```

```
Epoch 1/14
59/59 [==================================] - 3s 27ms/step - loss: 0.6405 - acc: 0.6897 - val_loss: 0.5670 - val_acc: 0.7914
Epoch 2/14
59/59 [==================================] - 1s 20ms/step - loss: 0.4694 - acc: 0.8598 - val_loss: 0.4082 - val_acc: 0.8796
Epoch 3/14
59/59 [==================================] - 1s 21ms/step - loss: 0.3091 - acc: 0.9133 - val_loss: 0.3036 - val_acc: 0.8900
Epoch 4/14
59/59 [==================================] - 1s 19ms/step - loss: 0.2137 - acc: 0.9306 - val_loss: 0.2760 - val_acc: 0.8909
Epoch 5/14
59/59 [==================================] - 1s 19ms/step - loss: 0.1633 - acc: 0.9572 - val_loss: 0.2966 - val_acc: 0.8872
Epoch 6/14
59/59 [==================================] - 1s 19ms/step - loss: 0.1063 - acc: 0.9666 - val_loss: 0.3203 - val_acc: 0.8822
Epoch 7/14
59/59 [==================================] - 1s 19ms/step - loss: 0.0859 - acc: 0.9749 - val_loss: 0.3424 - val_acc: 0.8819
Epoch 8/14
59/59 [==================================] - 1s 20ms/step - loss: 0.0703 - acc: 0.9799 - val_loss: 0.3661 - val_acc: 0.8810
Epoch 9/14
59/59 [==================================] - 1s 20ms/step - loss: 0.0569 - acc: 0.9843 - val_loss: 0.4131 - val_acc: 0.8764
Epoch 10/14
59/59 [==================================] - 1s 20ms/step - loss: 0.0452 - acc: 0.9892 - val_loss: 0.4342 - val_acc: 0.8778
Epoch 11/14
59/59 [==================================] - 1s 21ms/step - loss: 0.0357 - acc: 0.9920 - val_loss: 0.4751 - val_acc: 0.8756
Epoch 12/14
59/59 [==================================] - 1s 20ms/step - loss: 0.0273 - acc: 0.9945 - val_loss: 0.5394 - val_acc: 0.8685
Epoch 13/14
59/59 [==================================] - 1s 21ms/step - loss: 0.0205 - acc: 0.9963 - val_loss: 0.5765 - val_acc: 0.8701
Epoch 14/14
782/782 [==================================] - 2s 2ms/step - loss: 0.6480 - acc: 0.8533
[0.6479699015617371, 0.8532800078392029]
```
```python
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['acc']
val_acc = history_dict['val_acc']
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()
```