```
import numpy as np
from tensorflow.keras.datasets import imdb
from tensorflow.keras import layers,models
```

```
(train_data,train_lables),(test_data,test_lables)=imdb.load_data(num_words=10000)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/imdb.r
17464789/17464789 [==============================] - 0s 0us/step
```

```
train_lables[0]
```

```
1
```

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

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/imdb_w
1641221/1641221 [==============================] - 0s 0us/step
```

```
decode=' '.join([rev_word_index.get(i-3,'?')for i in train_data[0]])
decode
```

```
'? this film was just brilliant casting location scenery story direction everyone's rea
lly suited the part they played and you could just imagine being there robert ? is an a
mazing actor and now the same being director ? father came from the same scottish islan
d as myself so i loved the fact there was a real connection with this film the witty re
marks throughout the film were great it was just brilliant so much that i bought the fi
lm 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 was so sad and you know what they sa
y if you cry at a film it must have been good and this definitely was also ? to the two
```

```
def sequence_vec(sequences,dimensions=10000):
  results=np.zeros((len(sequences),dimensions))
  for i,seq in enumerate(sequences):
    results[i,seq]=1
  return results
```

```
x_train=sequence_vec(train_data)
x_test=np.asarray(train_lables).astype('float32')
y_train=sequence_vec(test_data)
y_test=np.asarray(test_lables).astype('float32')
```

```
x_train[0]
```

```
array([0., 1., 1., ..., 0., 0., 0.])
```

```
x_test[0]
```

```
1.0
```

```
y_train[0]
```

```
array([0., 1., 1., ..., 0., 0., 0.])
```

```
y_test[0]
```

```
0.0
```

```
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'])
```

```
x_val=x_train[:10000]
x_partial=x_train[10000:]
x_test_val=x_test[:10000]
x_test_partial=x_test[10000:]
```

```
model.fit(x_partial,x_test_partial,epochs=30,validation_data=(x_val,x_test_val),batch_size=5
```

```
Epoch 3/30
30/30 [==============================] - 1s 40ms/step - loss: 0.2479 - Accuracy: 0.9
Epoch 4/30
30/30 [==============================] - 1s 48ms/step - loss: 0.2032 - Accuracy: 0.9
Epoch 5/30
30/30 [==============================] - 1s 35ms/step - loss: 0.1680 - Accuracy: 0.9
Epoch 6/30
30/30 [==============================] - 1s 37ms/step - loss: 0.1437 - Accuracy: 0.9
Epoch 7/30
30/30 [==============================] - 1s 39ms/step - loss: 0.1243 - Accuracy: 0.9
Epoch 8/30
30/30 [==============================] - 1s 50ms/step - loss: 0.1051 - Accuracy: 0.9
```

```
30/30 [==============================] - 1s 36ms/step - loss: 0.0694 - Accuracy: 0.98
Epoch 12/30
30/30 [==============================] - 1s 36ms/step - loss: 0.0581 - Accuracy: 0.98
Epoch 13/30
30/30 [==============================] - 1s 35ms/step - loss: 0.0480 - Accuracy: 0.98
Epoch 14/30
30/30 [==============================] - 1s 34ms/step - loss: 0.0431 - Accuracy: 0.98
Epoch 15/30
30/30 [==============================] - 1s 35ms/step - loss: 0.0348 - Accuracy: 0.99
Epoch 16/30
30/30 [==============================] - 1s 36ms/step - loss: 0.0320 - Accuracy: 0.99
Epoch 17/30
30/30 [==============================] - 1s 36ms/step - loss: 0.0219 - Accuracy: 0.99
Epoch 18/30
30/30 [==============================] - 1s 36ms/step - loss: 0.0207 - Accuracy: 0.99
Epoch 19/30
30/30 [==============================] - 2s 57ms/step - loss: 0.0183 - Accuracy: 0.99
Epoch 20/30
30/30 [==============================] - 1s 48ms/step - loss: 0.0151 - Accuracy: 0.99
Epoch 21/30
30/30 [==============================] - 1s 33ms/step - loss: 0.0122 - Accuracy: 0.99
Epoch 22/30
30/30 [==============================] - 1s 36ms/step - loss: 0.0096 - Accuracy: 0.99
Epoch 23/30
30/30 [==============================] - 1s 47ms/step - loss: 0.0112 - Accuracy: 0.99
Epoch 24/30
30/30 [==============================] - 1s 36ms/step - loss: 0.0053 - Accuracy: 0.99
Epoch 25/30
30/30 [==============================] - 1s 36ms/step - loss: 0.0074 - Accuracy: 0.99
Epoch 26/30
30/30 [==============================] - 1s 35ms/step - loss: 0.0074 - Accuracy: 0.99
Epoch 27/30
30/30 [==============================] - 1s 36ms/step - loss: 0.0032 - Accuracy: 0.99
Epoch 28/30
30/30 [==============================] - 1s 35ms/step - loss: 0.0088 - Accuracy: 0.99
Epoch 29/30
30/30 [==============================] - 1s 33ms/step - loss: 0.0023 - Accuracy: 0.99
Epoch 30/30
30/30 [==============================] - 2s 57ms/step - loss: 0.0047 - Accuracy: 0.99
<keras.src.callbacks.History at 0x7b41bb2952a0>
```

```
loss1,acc1=model.evaluate(x_train,x_test)
print("Training accuracy :",acc1)
```

```
782/782 [==============================] - 2s 2ms/step - loss: 0.3043 - Accuracy: 0.9470
Training accuracy : 0.9470400214195251
```

```
loss2,acc2=model.evaluate(y_train,y_test)
print("Test accuracy :",acc2)
```

```
782/782 [==============================] - 2s 3ms/step - loss: 0.8125 - Accuracy: 0.8580
Test accuracy : 0.8580399751663208
```

Start coding or generate with AI.