```
from keras.datasets import imdb
from keras.preprocessing.sequence import pad_sequences
import numpy as np
import matplotlib.pyplot as plt

# Load dataset
(vocab_size, maxlen) = (10000, 200)
(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words=vocab_size)

# Explore dataset
print("Number of training samples:", len(x_train))
print("Number of test samples:", len(x_test))
print("Average review length:", np.mean([len(i) for i in x_train]))
print("Label distribution:", np.unique(y_train, return_counts=True))

# Pad sequences
x_train = pad_sequences(x_train, maxlen=maxlen)
x_test = pad_sequences(x_test, maxlen=maxlen)

# Visualize padded vs. original
print("Original:", x_train[0][-20:])
print("Padded:", x_train[0])
```

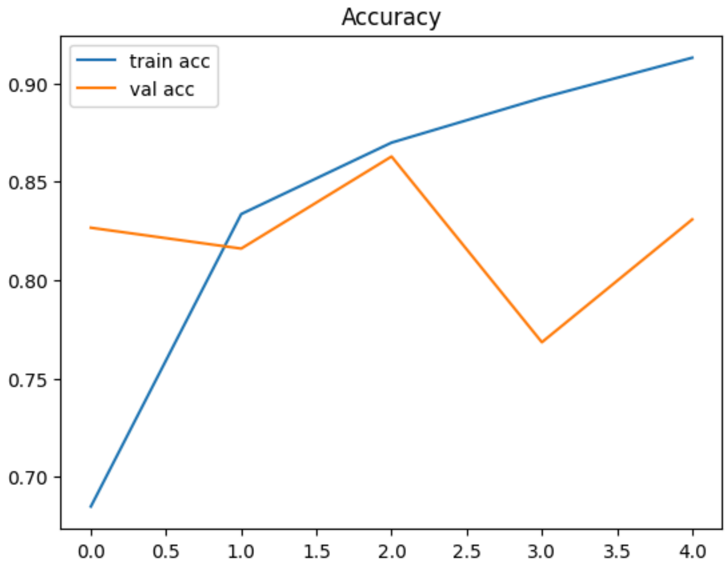
```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/imdb.npz
17464789/17464789 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
Number of training samples: 25000
Number of test samples: 25000
Average review length: 238.71364
Label distribution: (array([0, 1]), array([12500, 12500]))
Original: [  65   16   38 1334   88   12   16  283    5   16 4472  113  103   32
   15   16 5345   19  178   32]
Padded: [   5   25  100   43  838  112   50  670    2    9   35  480  284    5
  150    4  172  112  167    2  336  385   39    4  172 4536 1111   17
  546   38   13  447    4  192   50   16    6  147 2025   19   14   22
    4 1920 4613  469    4   22   71   87   12   16   43  530   38   76
   15   13 1247    4   22   17  515   17   12   16  626   18    2    5
   62  386   12    8  316    8  106    5    4 2223 5244   16  480   66
 3785   33    4  130   12   16   38  619    5   25  124   51   36  135
   48   25 1415   33    6   22   12  215   28   77   52    5   14  407
   16   82    2    8    4  107  117 5952   15  256    4    2    7 3766
    5  723   36   71   43  530  476   26  400  317   46    7    4    2
 1029   13  104   88    4  381   15  297   98   32 2071   56   26  141
    6  194 7486   18    4  226   22   21  134  476   26  480    5  144
   30 5535   18   51   36   28  224   92   25  104    4  226   65   16
   38 1334   88   12   16  283    5   16 4472  113  103   32   15   16
 5345   19  178   32]
```

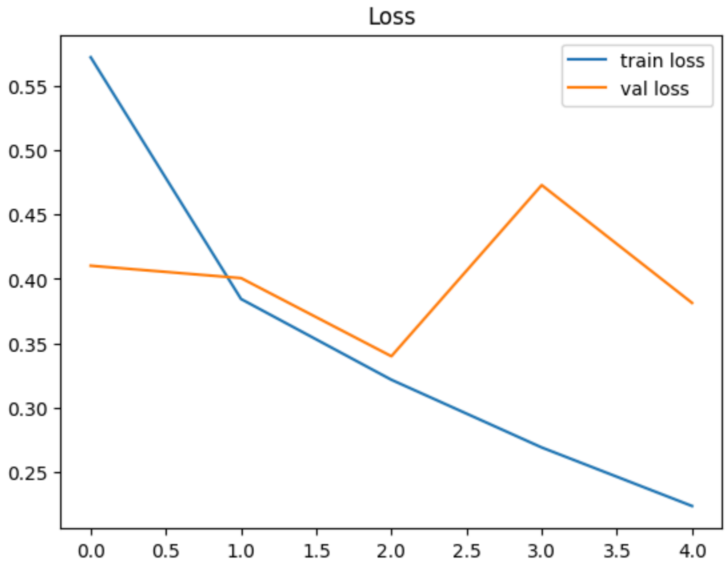
```
from keras.models import Sequential
from keras.layers import Embedding, SimpleRNN, Dense

model = Sequential([
    Embedding(input_dim=10000, output_dim=32, input_length=200),
    SimpleRNN(32),
    Dense(1, activation='sigmoid')
])

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

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/embedding.py:90: UserWarning: Argument `input_length` is deprecated. Just
  warnings.warn(
Model: "sequential"
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━┓
┃ Layer (type)                    ┃ Output Shape           ┃       Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━┩
│ embedding (Embedding)           │ ?                      │   0 (unbuilt) │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ simple_rnn (SimpleRNN)          │ ?                      │   0 (unbuilt) │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense (Dense)                   │ ?                      │   0 (unbuilt) │
└─────────────────────────────────┴────────────────────────┴───────────────┘
 Total params: 0 (0.00 B)
 Trainable params: 0 (0.00 B)
 Non-trainable params: 0 (0.00 B)
```

```
history = model.fit(x_train, y_train, epochs=5, batch_size=128, validation_split=0.2)

# Plot accuracy and loss
plt.plot(history.history['accuracy'], label='train acc')
plt.plot(history.history['val_accuracy'], label='val acc')
plt.title('Accuracy')
plt.legend()
plt.show()

plt.plot(history.history['loss'], label='train loss')
plt.plot(history.history['val_loss'], label='val loss')
plt.title('Loss')
plt.legend()
plt.show()

# Evaluate
test_loss, test_acc = model.evaluate(x_test, y_test)
print("Test Accuracy:", test_acc)
```

```
Epoch 1/5
157/157 ━━━━━━━━━━━━━━━━━━━━ 13s 71ms/step - accuracy: 0.5949 - loss: 0.6456 - val_accuracy: 0.8266 - val_loss: 0.4102
Epoch 2/5
157/157 ━━━━━━━━━━━━━━━━━━━━ 19s 59ms/step - accuracy: 0.8201 - loss: 0.4093 - val_accuracy: 0.8160 - val_loss: 0.4006
Epoch 3/5
157/157 ━━━━━━━━━━━━━━━━━━━━ 10s 55ms/step - accuracy: 0.8706 - loss: 0.3207 - val_accuracy: 0.8628 - val_loss: 0.3399
Epoch 4/5
157/157 ━━━━━━━━━━━━━━━━━━━━ 11s 61ms/step - accuracy: 0.8930 - loss: 0.2710 - val_accuracy: 0.7684 - val_loss: 0.4727
Epoch 5/5
157/157 ━━━━━━━━━━━━━━━━━━━━ 11s 64ms/step - accuracy: 0.9095 - loss: 0.2331 - val_accuracy: 0.8308 - val_loss: 0.3813
```





```
782/782 ━━━━━━━━━━━━━━━━━━━━ 9s 11ms/step - accuracy: 0.8371 - loss: 0.3845
Test Accuracy: 0.8365600109100342
```

```
SimpleRNN(64)
```

```
<SimpleRNN name=simple_rnn_1, built=False>
```

```
from keras.layers import LSTM

model = Sequential([
    Embedding(input_dim=10000, output_dim=32, input_length=200),
    LSTM(32),
    Dense(1, activation='sigmoid')
])
```

```
# Change vocab_size to 5000 or 2000 in imdb.load_data
from keras.layers import Dropout

model = Sequential([
    Embedding(input_dim=10000, output_dim=32, input_length=200),
    SimpleRNN(32, dropout=0.2),
    Dense(1, activation='sigmoid')
])
```