

```
In [1]: import numpy as np
import matplotlib.pyplot as plt

from tensorflow import keras
from tensorflow.keras import layers
```

```
In [2]: # Load dataset (top 10,000 most frequent words)
num_words = 10000

(X_train, y_train), (X_test, y_test) = keras.datasets.imdb.load_data(num_words=num_

print("Training samples:", len(X_train))
print("Testing samples:", len(X_test))
```

Downloading data from <https://storage.googleapis.com/tensorflow/tf-keras-datasets/imdb.npz>

17464789/17464789 ————— 3s 0us/step

Training samples: 25000

Testing samples: 25000

```
In [3]: print("Sample review (encoded):", X_train[0])
print("Label:", y_train[0]) # 1 = positive, 0 = negative
```

Sample review (encoded): [1, 14, 22, 16, 43, 530, 973, 1622, 1385, 65, 458, 4468, 6, 3941, 4, 173, 36, 256, 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, 4, 3, 530, 38, 76, 15, 13, 1247, 4, 22, 17, 515, 17, 12, 16, 626, 18, 2, 5, 62, 386, 1, 2, 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, 1, 6, 82, 2, 8, 4, 107, 117, 5952, 15, 256, 4, 2, 7, 3766, 5, 723, 36, 71, 43, 530, 47, 6, 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, 447, 2, 113, 103, 32, 15, 16, 5345, 19, 178, 32]

Label: 1

```
In [5]: # Make all reviews same length
maxlen = 200

X_train = keras.preprocessing.sequence.pad_sequences(X_train, maxlen=maxlen)
X_test = keras.preprocessing.sequence.pad_sequences(X_test, maxlen=maxlen)
```

```
In [6]: model = keras.Sequential([
    layers.Embedding(input_dim=num_words, output_dim=128, input_length=maxlen),

    layers.Flatten(),

    layers.Dense(64, activation='relu'),
    layers.Dense(32, activation='relu'),

    layers.Dense(1, activation='sigmoid') # Binary classification
])
```

```
D:\DL\.venv\lib\site-packages\keras\src\layers\core\embedding.py:97: UserWarning: Argument `input_length` is deprecated. Just remove it.
warnings.warn(
```

```
In [7]: model.compile(
        optimizer='adam',
        loss='binary_crossentropy',
        metrics=['accuracy']
    )
```

WARNING:tensorflow:TensorFlow GPU support is not available on native Windows for TensorFlow >= 2.11. Even if CUDA/cuDNN are installed, GPU will not be used. Please use WSL2 or the TensorFlow-DirectML plugin.

```
In [8]: history = model.fit(
        X_train, y_train,
        epochs=5,
        batch_size=32,
        validation_split=0.2,
        verbose=1
    )
```

Epoch 1/5

**625/625** ————— **16s** 23ms/step - accuracy: 0.7880 - loss: 0.4221 - val\_accuracy: 0.8740 - val\_loss: 0.3010

Epoch 2/5

**625/625** ————— **15s** 23ms/step - accuracy: 0.9724 - loss: 0.0807 - val\_accuracy: 0.8434 - val\_loss: 0.4483

Epoch 3/5

**625/625** ————— **17s** 27ms/step - accuracy: 0.9957 - loss: 0.0129 - val\_accuracy: 0.8432 - val\_loss: 0.6783

Epoch 4/5

**625/625** ————— **17s** 27ms/step - accuracy: 0.9969 - loss: 0.0096 - val\_accuracy: 0.8194 - val\_loss: 0.8213

Epoch 5/5

**625/625** ————— **17s** 27ms/step - accuracy: 0.9908 - loss: 0.0253 - val\_accuracy: 0.8378 - val\_loss: 0.7287

```
In [9]: loss, accuracy = model.evaluate(X_test, y_test)

        print("Test Loss:", loss)
        print("Test Accuracy:", accuracy)
```

**782/782** ————— **3s** 3ms/step - accuracy: 0.8387 - loss: 0.7404

Test Loss: 0.7404467463493347

Test Accuracy: 0.8386800289154053

```
In [10]: predictions = model.predict(X_test)

        # Convert probabilities to binary output
        predicted_labels = (predictions > 0.5).astype(int)

        print("Predicted:", predicted_labels[:10].flatten())
        print("Actual:", y_test[:10])
```

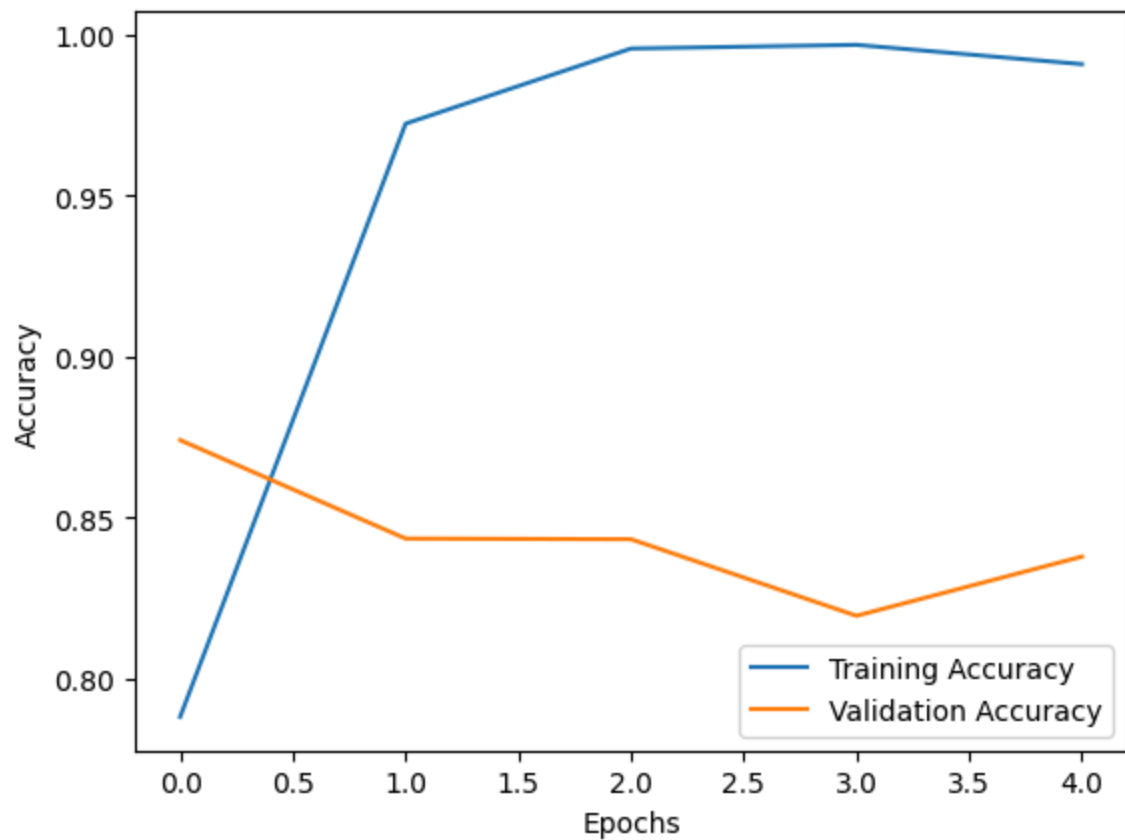
782/782 ————— 2s 3ms/step

Predicted: [0 1 1 0 1 0 1 0 1 1]

Actual: [0 1 1 0 1 1 1 0 0 1]

```
In [11]: plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')

plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```



In [ ]: