

## DL\_2

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


```
vocab_size = 10000 # Use top 10,000 words
maxlen = 200
```

```
(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words=vocab_size)
```

```
x_train = pad_sequences(x_train, maxlen=maxlen)
x_test = pad_sequences(x_test, maxlen=maxlen)
```

```
model = keras.Sequential([
    layers.Embedding(input_dim=vocab_size, output_dim=32, input_length=maxlen),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(1, activation='sigmoid') # Output layer with sigmoid for binary classification
])
```

```
⚡ /usr/local/lib/python3.11/dist-packages/keras/src/layers/core/embedding.py:90: UserWarning: Argument `input_length` is deprecated. Just
warnings.warn(
```



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

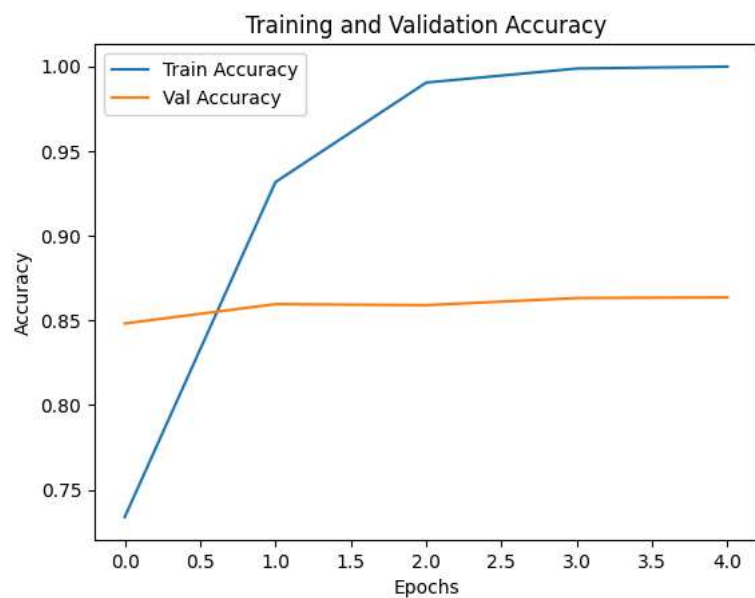
```
history = model.fit(x_train, y_train, epochs=5, batch_size=128, validation_split=0.2)
```

```
⚡ Epoch 1/5
157/157 ————— 8s 31ms/step - accuracy: 0.6210 - loss: 0.6201 - val_accuracy: 0.8482 - val_loss: 0.3490
Epoch 2/5
157/157 ————— 4s 23ms/step - accuracy: 0.9318 - loss: 0.1914 - val_accuracy: 0.8596 - val_loss: 0.3230
Epoch 3/5
157/157 ————— 5s 24ms/step - accuracy: 0.9911 - loss: 0.0518 - val_accuracy: 0.8590 - val_loss: 0.3896
Epoch 4/5
157/157 ————— 6s 31ms/step - accuracy: 0.9988 - loss: 0.0123 - val_accuracy: 0.8632 - val_loss: 0.4106
Epoch 5/5
157/157 ————— 4s 23ms/step - accuracy: 0.9996 - loss: 0.0050 - val_accuracy: 0.8636 - val_loss: 0.4341
```

```
test_loss, test_acc = model.evaluate(x_test, y_test, verbose=0)
print(f"\nTest Accuracy: {test_acc:.2f}")
```

```
⚡ Test Accuracy: 0.86
```

```
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Val Accuracy')
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.title("Training and Validation Accuracy")
plt.legend()
plt.show()
```



Start coding or [generate](#) with AI.