# Import required libraries

import tensorflow as tf

from tensorflow import keras

from tensorflow.keras import layers

import matplotlib.pyplot as plt

import numpy as np

# Load IMDB dataset

vocab\_size = 10000 # Only consider the top 10,000 words

maxlen = 200 # Limit each review to 200 words

(x\_train, y\_train), (x\_test, y\_test) = keras.datasets.imdb.load\_data(num\_words=vocab\_size)

# Pad sequences to ensure uniform length

x\_train = keras.preprocessing.sequence.pad\_sequences(x\_train, maxlen=maxlen)

x\_test = keras.preprocessing.sequence.pad\_sequences(x\_test, maxlen=maxlen)

# Build the DNN model

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 for binary classification

])

# Compile the model

model.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['accuracy'])

# Train the model

history = model.fit(x\_train, y\_train, epochs=5, batch\_size=128, validation\_split=0.2)

# Evaluate on test data

loss, accuracy = model.evaluate(x\_test, y\_test)

print(f"\nTest Accuracy: {accuracy:.2f}")

# Plot accuracy and loss

plt.figure(figsize=(12, 5))

# Accuracy plot

plt.subplot(1, 2, 1)

plt.plot(history.history['accuracy'], label="Train Accuracy")

plt.plot(history.history['val\_accuracy'], label="Validation Accuracy")

plt.title("Accuracy over Epochs")

plt.xlabel("Epoch")

plt.ylabel("Accuracy")

plt.legend()

# Loss plot

plt.subplot(1, 2, 2)

plt.plot(history.history['loss'], label="Train Loss")

plt.plot(history.history['val\_loss'], label="Validation Loss")

plt.title("Loss over Epochs")

plt.xlabel("Epoch")

plt.ylabel("Loss")

plt.legend()

plt.tight\_layout()

plt.show()

# Word index decoding utility

word\_index = keras.datasets.imdb.get\_word\_index()

index\_word = {index + 3: word for word, index in word\_index.items()}

index\_word[0] = "<PAD>"

index\_word[1] = "<START>"

index\_word[2] = "<UNK>"

index\_word[3] = "<UNUSED>"

def decode\_review(sequence):

return ' '.join([index\_word.get(i, '?') for i in sequence])

# Clean decoding function (removes <PAD>, <START>)

def decode\_clean\_review(sequence):

cleaned\_tokens = [token for token in sequence if token > 3]

return decode\_review(cleaned\_tokens)

# Sample predictions on test data

print("\n\nSample Predictions on Test Data:\n")

num\_samples\_to\_show = 5

predictions = model.predict(x\_test[:num\_samples\_to\_show])

for i in range(num\_samples\_to\_show):

decoded = decode\_clean\_review(x\_test[i])

actual\_label = "Positive" if y\_test[i] == 1 else "Negative"

predicted\_label = "Positive" if predictions[i] >= 0.5 else "Negative"

print(f"Review {i+1}:")

print(decoded[:500] + "...\n")

print(f"Actual Label: {actual\_label}")

print(f"Predicted Label: {predicted\_label}")

print("-" \* 80)