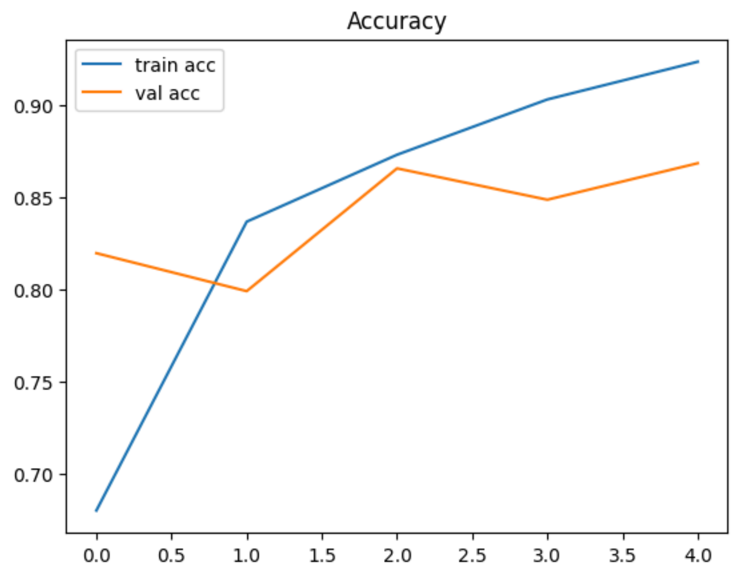
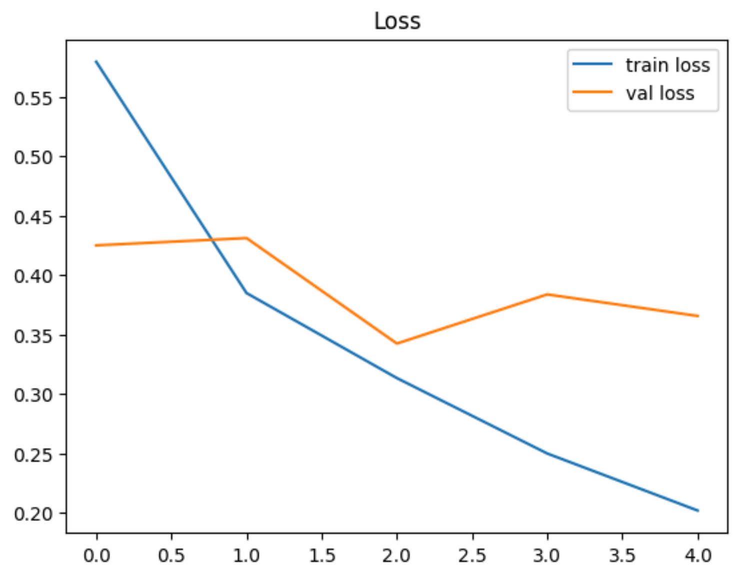
```
import numpy as np
from keras.datasets import imdb
from keras.preprocessing.sequence import pad_sequences
from keras.models import Sequential
from keras.layers import Embedding, SimpleRNN, Dense
import matplotlib.pyplot as plt

# Task 1: Data Preparation
num_words = 10000
maxlen = 200

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

# Explore data
print(f"Number of training samples: {len(x_train)}")
print(f"Average review length: {np.mean([len(x) for x in x_train])}")
print(f"Label distribution: {np.bincount(y_train)}")

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

# Visualize a sample
print("Original review:", x_train[0][:10])
print("Padded review:", x_train_padded[0][:10])

# Task 2: Model Building
model = Sequential()
model.add(Embedding(input_dim=10000, output_dim=32, input_length=200))
model.add(SimpleRNN(32))
model.add(Dense(1, activation='sigmoid'))

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

# Task 3: Training and Evaluation
history = model.fit(x_train_padded, y_train, epochs=5, batch_size=128, validation_split=0.2)

# Plot accuracy/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 on test set
test_loss, test_acc = model.evaluate(x_test_padded, y_test)
print(f"Test accuracy: {test_acc:.4f}")
```

```
17464789/17464789 ━━━━━━━━━━━━━━━━━━━━ 2s 0us/step
Number of training samples: 25000
Average review length: 238.71364
Label distribution: [12500 12500]
Original review: [1, 14, 22, 16, 43, 530, 973, 1622, 1385, 65]
Padded review: [  5  25 100  43 838 112  50 670   2   9]
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/embedding.py:90: UserWarning: Argument `input_length` is deprecated. Ju
  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)
Epoch 1/5
157/157 ━━━━━━━━━━━━━━━━━━━━ 13s 67ms/step - accuracy: 0.5951 - loss: 0.6489 - val_accuracy: 0.8194 - val_loss: 0.4250
Epoch 2/5
157/157 ━━━━━━━━━━━━━━━━━━━━ 20s 63ms/step - accuracy: 0.8382 - loss: 0.3852 - val_accuracy: 0.7988 - val_loss: 0.4311
Epoch 3/5
157/157 ━━━━━━━━━━━━━━━━━━━━ 10s 63ms/step - accuracy: 0.8727 - loss: 0.3158 - val_accuracy: 0.8654 - val_loss: 0.3424
Epoch 4/5
157/157 ━━━━━━━━━━━━━━━━━━━━ 10s 64ms/step - accuracy: 0.9079 - loss: 0.2409 - val_accuracy: 0.8484 - val_loss: 0.3837
Epoch 5/5
157/157 ━━━━━━━━━━━━━━━━━━━━ 9s 55ms/step - accuracy: 0.9227 - loss: 0.2047 - val_accuracy: 0.8682 - val_loss: 0.3656
```





```
782/782 ━━━━━━━━━━━━━━━━━━━━ 8s 10ms/step - accuracy: 0.8565 - loss: 0.3861
Test accuracy: 0.8574
```

```
import numpy as np
import matplotlib.pyplot as plt
from keras.datasets import imdb
from keras.preprocessing.sequence import pad_sequences
from keras.models import Sequential
from keras.layers import Embedding, SimpleRNN, LSTM, Dense, Dropout
from keras.optimizers import RMSprop

# Load and preprocess data
num_words = 10000
maxlen = 200

(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words=num_words)
x_train = pad_sequences(x_train, maxlen=maxlen)
x_test = pad_sequences(x_test, maxlen=maxlen)

def plot_history(history, title):
    plt.figure(figsize=(12, 5))
    plt.subplot(1, 2, 1)
    plt.plot(history.history['accuracy'], label='Train Acc')
    plt.plot(history.history['val_accuracy'], label='Val Acc')
    plt.title(f'{title} - Accuracy')
    plt.legend()

    plt.subplot(1, 2, 2)
    plt.plot(history.history['loss'], label='Train Loss')
    plt.plot(history.history['val_loss'], label='Val Loss')
    plt.title(f'{title} - Loss')
    plt.legend()
    plt.show()

# 1. Change SimpleRNN units from 32 to 64
model_rnn_64 = Sequential([
    Embedding(input_dim=10000, output_dim=32, input_length=200),
    SimpleRNN(64),
    Dense(1, activation='sigmoid')
])
model_rnn_64.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['accuracy'])
history_rnn_64 = model_rnn_64.fit(x_train, y_train, epochs=5, batch_size=128, validation_split=0.2)
plot_history(history_rnn_64, "SimpleRNN (64 units)")

# 2. Replace SimpleRNN with LSTM
model_lstm = Sequential([
    Embedding(input_dim=10000, output_dim=32, input_length=200),
    LSTM(32),
    Dense(1, activation='sigmoid')
])
model_lstm.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['accuracy'])
history_lstm = model_lstm.fit(x_train, y_train, epochs=5, batch_size=128, validation_split=0.2)
plot_history(history_lstm, "LSTM (32 units)")

# 3. Reduce vocabulary size to 5000 and 2000
for vocab_size in [5000, 2000]:
    (x_train_small, y_train_small), (x_test_small, y_test_small) = imdb.load_data(num_words=vocab_size)
    x_train_small = pad_sequences(x_train_small, maxlen=maxlen)
    x_test_small = pad_sequences(x_test_small, maxlen=maxlen)

    model_vocab = Sequential([
        Embedding(input_dim=vocab_size, output_dim=32, input_length=200),
        SimpleRNN(32),
        Dense(1, activation='sigmoid')
    ])
    model_vocab.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['accuracy'])
    history_vocab = model_vocab.fit(x_train_small, y_train_small, epochs=5, batch_size=128, validation_split=0.2)
    plot_history(history_vocab, f"SimpleRNN (vocab size = {vocab_size})")

# 4. Use dropout=0.2 in the RNN layer
model_dropout = Sequential([
    Embedding(input_dim=10000, output_dim=32, input_length=200),
    SimpleRNN(32, dropout=0.2),
    Dense(1, activation='sigmoid')
])
model_dropout.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['accuracy'])
history_dropout = model_dropout.fit(x_train, y_train, epochs=5, batch_size=128, validation_split=0.2)
plot_history(history_dropout, "SimpleRNN (with Dropout)")
```