Lab Work 5 Report
Madiyar Kaliyev

1. Loading the IMDB Dataset:
The code begins by importing the necessary modules and loading the IMDB dataset using Keras. The dataset consists of 25,000 movie reviews labeled as positive or negative sentiment. It is preprocessed, and only the top 10,000 most frequent words are considered. The dataset is split into training and testing sets.

```
from keras import models
from keras import layers
from keras.datasets import imdb
import numpy as np

# Load the IMDB dataset
(train_data, train_labels), (test_data, test_labels) =
imdb.load_data(num_words=10000)
```

2. Vectorizing Sequences:
A function vectorize_sequences is defined to convert sequences into a binary matrix, which can be used as input to a neural network. This function ensures that the input data is suitable for training.

```
# Vectorize the sequences
def vectorize_sequences(sequences, dimension=10000):
    results = np.zeros((len(sequences), dimension))
    for i, sequence in enumerate(sequences):
        results[i, sequence] = 1.
    return results
```

3. Building and Compiling the Model:
The neural network model is constructed using the Sequential API from Keras. It consists of an input layer, two hidden layers with ReLU activation functions, and an output layer with a sigmoid activation function. The model is compiled with the Adam optimizer, binary crossentropy loss, and accuracy as the evaluation metric.

```
# Build the neural network
model = models.Sequential()
model.add(layers.Dense(16, activation='relu', input_shape=(10000,)))
model.add(layers.Dense(16, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
```

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

4. Training the Model:
The model is trained on the training data using the fit method. The training process is monitored, and the history is stored for later visualization. The validation split is set to 20%, and the training is performed for 10 epochs with a batch size of 10.

```
# Train the model on the IMDb dataset
history = model.fit(x_train, y_train, epochs=10, batch_size=10,
validation_split=0.2)
accuracy= model.evaluate(x_test, y_test)
```

5. Adding Regularization:
L1 and L2 regularization are introduced to the model to observe their impact on performance. The regularization is applied to the hidden layers, and the model is recompiled and retrained with the regularization terms.

```
from keras import regularizers

# Build the neural network
model = models.Sequential()
model.add(layers.Dense(16,
activation='relu',kernel_regularizer=regularizers.l1(0.01),
input_shape=(10000,)))
model.add(layers.Dense(16, activation='relu',
kernel_regularizer=regularizers.l2(0.01)))
model.add(layers.Dense(1, activation='sigmoid'))


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

# Train the model on the IMDb dataset
```

```
history = model.fit(x_train, y_train, epochs=10, batch_size=10,
validation_split=0.2)
accuracy= model.evaluate(x_test, y_test)
```

6. Plotting Accuracy and Loss:
Matplotlib is used to visualize the training and validation accuracy as well as the training and validation loss over epochs.

```
from matplotlib import pyplot as plt

#plotting train and validation accuracy

plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()
```

```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()
```

```
from matplotlib import pyplot as plt

#plotting train and validation accuracy

plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()
```

```
from matplotlib import pyplot as plt

#plotting train and validation accuracy

plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()
```