DL-LAB-8-Activities

1. Demonstrate **Text Classification** (sentiment analysis) using **CNN** on IMDB movie review dataset (positive review -1, negative review – 0).
2. Compare the Performance of Text Classification using CNN vs using SimpleRNN/LSTM on the same dataset. (Refer to last weeks exercises and use them here)
3. Combine CNN and LSTM for sequence classification. (Refer to tutorial)
4. Demonstrate Text Generation using SimpleRNN/LSTM, Characer-by-character. (Ref:
   Ref:https://www.tensorflow.org/text/tutorials/text_generation)

****

Exercise -1 (tutorial)

- **Data Preparation:** The raw text is first converted into sequences of integers, where each integer represents a word in the vocabulary. These sequences must be padded to a uniform length so the model can process them in batches.
- **Embedding Layer:** This layer transforms the integer sequences into dense, fixed-size vectors (word embeddings), which can capture semantic relationships between words.
- **Conv1D Layer:** This is the core convolutional layer for text. A filter slides over the embedded words (e.g., a window of 5 words at a time) to detect local patterns (like n-grams or phrases).
- **GlobalMaxPooling1D:** This pooling layer reduces the dimensionality of the feature maps by taking the maximum value from the entire sequence for each filter, identifying the most important features captured by the convolutional filters.
- **Dense and Dropout Layers:** These are standard fully connected layers for processing the extracted features and outputting the final classification probability. Dropout is used for regularization to prevent overfitting.
- **Compilation:** The model is compiled with an optimizer (adam) and a suitable loss function (binary_crossentropy for a two-class problem).

****

Exercise - 1 (CODE SNIPPET)

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, Conv1D, GlobalMaxPooling1D, Dense, Dropout
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.datasets import imdb

# --- 1. Load and Prepare Data ---
vocab_size = 10000  # Only consider the top 10k words
```

```
max_length = 500    # Pad/truncate all reviews to a fixed length of 500 words

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

# Pad sequences to a uniform length
x_train = pad_sequences(x_train, maxlen=max_length)
x_test = pad_sequences(x_test, maxlen=max_length)

# --- 2. Build the CNN Model ---
embedding_dim = 100
model = Sequential()
# Embedding layer converts word indices to dense vectors
model.add(Embedding(input_dim=vocab_size, output_dim=embedding_dim,
input_length=max_length))
# Conv1D layer applies 128 filters that look at 5 words at a time
model.add(Conv1D(filters=128, kernel_size=5, activation='relu'))
# Global max pooling extracts the most important features regardless of position
model.add(GlobalMaxPooling1D())
# Fully connected layers for classification
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5)) # Dropout helps prevent overfitting
# Final output layer with sigmoid activation for binary classification (0 or 1)
model.add(Dense(1, activation='sigmoid'))

# --- 3. Compile and Train ---
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
model.summary()

epochs = 5
batch_size = 128
history = model.fit(x_train, y_train,
                    epochs=epochs,
                    batch_size=batch_size,
                    validation_split=0.2, # Use 20% of data for validation
                    verbose=1)

# --- 4. Evaluate the Model ---
loss, accuracy = model.evaluate(x_test, y_test, verbose=0)
print(f'Test loss: {loss}')
print(f'Test accuracy: {accuracy}')
```

Exercise-3:

Refer to pdf: (CNN-LSTM-SEQUENCE-CLASSIFIACATION.pdf)