DL Assignment 3

Name : Ankita Singh || Roll no: 57

Problem Statement: Binary classification using Deep Neural Networks Example: Classify movie reviews into positive" reviews and "negative" reviews, just based on the text content of the reviews. Use IMDB dataset

In [3]:
```
import numpy as np
from tensorflow import keras
from tensorflow.keras.datasets import imdb
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Embedding, Flatten
```

In [4]:
```
# Set parameters
max_words = 10000 # Consider only the top 10,000 most frequently occurring words
max_length = 250 # Limit the review length to 250 words
embedding_size = 50 # Dimensionality of the word embeddings
```

In [5]:
```
# Load the IMDB dataset
(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words=max_words)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/imdb.npz
17464789/17464789 ━━━━━━━━━━━━━━━━━━━━ 4s 0us/step
```

In [6]:
```
x_train = keras.preprocessing.sequence.pad_sequences(x_train, maxlen=max_length)
x_test = keras.preprocessing.sequence.pad_sequences(x_test, maxlen=max_length)
```

In [7]:
```
model = Sequential()
```

In [8]:
```
model.add(Embedding(max_words, embedding_size, input_length=max_length))
```

```
/home/admin14/.local/lib/python3.10/site-packages/keras/src/layers/core/embedding.py:90: UserWarning: Argument `
input_length` is deprecated. Just remove it.
  warnings.warn(
```

In [9]:
```
model.add(Flatten())
```

In [11]:
```
model.add(Dense(128, activation='relu'))
```

In [12]:
```
model.add(Dropout(0.5))
```

In [13]:
```
model.add(Dense(1, activation='sigmoid'))
```

In [14]:
```
# Compile the model
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

In [19]:
```
batch_size = 32
epochs = 5
```

In [20]:
```
model.fit(x_train, y_train, batch_size=batch_size, epochs=epochs, validation_data=(x_test, y_test))
```

```
Epoch 1/5
782/782 ━━━━━━━━━━━━━━━━━━━━ 18s 23ms/step - accuracy: 0.8970 - loss: 0.3020 - val_accuracy: 0.8674 - val_loss:
0.3325
Epoch 2/5
782/782 ━━━━━━━━━━━━━━━━━━━━ 18s 23ms/step - accuracy: 0.9615 - loss: 0.1432 - val_accuracy: 0.8566 - val_loss:
0.3927
Epoch 3/5
782/782 ━━━━━━━━━━━━━━━━━━━━ 18s 23ms/step - accuracy: 0.9802 - loss: 0.0868 - val_accuracy: 0.8498 - val_loss:
0.4999
Epoch 4/5
782/782 ━━━━━━━━━━━━━━━━━━━━ 18s 23ms/step - accuracy: 0.9876 - loss: 0.0597 - val_accuracy: 0.8480 - val_loss:
0.5403
Epoch 5/5
782/782 ━━━━━━━━━━━━━━━━━━━━ 18s 23ms/step - accuracy: 0.9901 - loss: 0.0486 - val_accuracy: 0.8488 - val_loss:
0.6000
```

Out[20]:
```
<keras.src.callbacks.history.History at 0x75f3f099c7c0>
```

In [21]:
```
loss, accuracy = model.evaluate(x_test, y_test)
```

```
782/782 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8499 - loss: 0.5982
```

In [22]:
```
print("Test Loss:", loss)
print("Test Accuracy:", accuracy)
```

```
Test Loss: 0.599955677986145
Test Accuracy: 0.8487600088119507
```

```
In [ ]:
```