N.V.Nithin Kumar

vnamburi@gitam.in

Tensors - Tensors represent deep learning data. They are multidimensional arrays, used to store multiple dimensions of a dataset. Each dimension is called a feature. For example, a cube storing data across an X, Y, and Z access is represented as a 3-dimensional tensor. Tensors can store very high dimensionality, with hundreds of dimensions of features typically used in deep learning applications.

```
#importing libraries
import tensorflow as tf
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.utils import to_categorical


# Load the text data
text = open('/content/pg5200.txt').read()


# Tokenize the text
tokenizer = Tokenizer()
tokenizer.fit_on_texts([text])
encoded = tokenizer.texts_to_sequences([text])[0]


# Generate training data
sequences = []
for i in range(1, len(encoded)):
    sequence = encoded[i-1:i+3]
    sequences.append(sequence)
X = []
y = []
for seq in sequences:
    X.append(seq[:-1])
    y.append(seq[-1])
X = tf.keras.preprocessing.sequence.pad_sequences(X, maxlen=3)
y = to_categorical(y, num_classes=len(tokenizer.word_index)+1)


# Define the model architecture
model = tf.keras.Sequential([
    tf.keras.layers.Embedding(input_dim=len(tokenizer.word_index)+1, output_dim=64, input_len
    tf.keras.layers.LSTM(128),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(len(tokenizer.word_index)+1, activation='softmax')
])



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



# Train the model
history = model.fit(X, y, epochs=25, verbose=1)
```

```
Epoch 1/25
791/791 [==============================] - 18s 16ms/step - loss: 6.4199 - accuracy: 0.05
Epoch 2/25
791/791 [==============================] - 13s 17ms/step - loss: 6.0637 - accuracy: 0.05
Epoch 3/25
791/791 [==============================] - 13s 16ms/step - loss: 5.9044 - accuracy: 0.05
Epoch 4/25
791/791 [==============================] - 14s 17ms/step - loss: 5.6735 - accuracy: 0.08
Epoch 5/25
791/791 [==============================] - 14s 17ms/step - loss: 5.3985 - accuracy: 0.10
Epoch 6/25
791/791 [==============================] - 14s 18ms/step - loss: 5.1786 - accuracy: 0.12
Epoch 7/25
791/791 [==============================] - 14s 18ms/step - loss: 4.9979 - accuracy: 0.13
Epoch 8/25
791/791 [==============================] - 15s 19ms/step - loss: 4.8298 - accuracy: 0.14
Epoch 9/25
791/791 [==============================] - 16s 20ms/step - loss: 4.6684 - accuracy: 0.15
Epoch 10/25
791/791 [==============================] - 13s 16ms/step - loss: 4.5220 - accuracy: 0.16
Epoch 11/25
791/791 [==============================] - 13s 16ms/step - loss: 4.3813 - accuracy: 0.17
Epoch 12/25
791/791 [==============================] - 13s 16ms/step - loss: 4.2373 - accuracy: 0.18
Epoch 13/25
791/791 [==============================] - 12s 15ms/step - loss: 4.1020 - accuracy: 0.19
Epoch 14/25
791/791 [==============================] - 13s 17ms/step - loss: 3.9608 - accuracy: 0.21
Epoch 15/25
791/791 [==============================] - 13s 16ms/step - loss: 3.8193 - accuracy: 0.22
Epoch 16/25
791/791 [==============================] - 13s 16ms/step - loss: 3.6900 - accuracy: 0.23
Epoch 17/25
791/791 [==============================] - 13s 17ms/step - loss: 3.5531 - accuracy: 0.25
Epoch 18/25
791/791 [==============================] - 13s 16ms/step - loss: 3.4219 - accuracy: 0.26
Epoch 19/25
791/791 [==============================] - 13s 17ms/step - loss: 3.2957 - accuracy: 0.28
Epoch 20/25
791/791 [==============================] - 13s 17ms/step - loss: 3.1771 - accuracy: 0.29
Epoch 21/25
791/791 [==============================] - 13s 16ms/step - loss: 3.0552 - accuracy: 0.31
Epoch 22/25
791/791 [==============================] - 13s 16ms/step - loss: 2.9442 - accuracy: 0.32
```

```
Epoch 23/25
791/791 [==============================] - 13s 16ms/step - loss: 2.8403 - accuracy: 0.34
Epoch 24/25
791/791 [==============================] - 13s 16ms/step - loss: 2.7466 - accuracy: 0.36
Epoch 25/25
791/791 [==============================] - 13s 16ms/step - loss: 2.6546 - accuracy: 0.37
```

```
# Evaluate the model on test data
test_text = "How about if I sleep a"
tokenized_text = tokenizer.texts_to_sequences([test_text])[0]
encoded_text = tf.keras.preprocessing.sequence.pad_sequences([tokenized_text], maxlen=3)
predicted_probs = model.predict(encoded_text)[0]
predicted_word_index = tf.argmax(predicted_probs).numpy()
predicted_word = tokenizer.index_word[predicted_word_index]
```

```
1/1 [==============================] - 1s 937ms/step
```

```
# Print the predicted next word
print(predicted_word)
```

```
little
```