100 Days with TensorFlow - One Day Per Day!

This document will summarize 100 lessons on learning TensorFlow, one lesson each day. The difficulty will gradually increase, and we will review progress weekly. The goal is to build a solid foundation in TensorFlow and work on a big project together as the lessons advance. Each lesson will include code examples and explanations.

Lesson 1: Introduction to TensorFlow Basics

In this lesson, we introduced TensorFlow and its use of tensors for handling data. Tensors are the fundamental building blocks in TensorFlow. A tensor is a generalization of vectors and matrices to potentially higher dimensions. TensorFlow represents all data in this multi-dimensional array format.

We started by printing the TensorFlow version and creating two constant tensors, using `tf.constant()`. Tensors in TensorFlow are immutable (their values can't change), and they can store data of any type like integers, floats, strings, or booleans.

We then performed a basic addition operation using `tf.add()`. TensorFlow provides built-in functions for performing mathematical operations on tensors, such as addition, multiplication, and matrix operations. The result of an operation is returned as another tensor.

In Part 2, we explored tensor shapes. A tensor's shape defines its dimensionality, or how many elements it contains in each dimension. A scalar (0-D tensor) has no dimensions, while a vector (1-D tensor) has one. A matrix (2-D tensor) has two dimensions, and higher-dimensional tensors can represent more complex data structures like batches of images.

Reshaping tensors is an important concept in TensorFlow. It allows you to change the shape of the tensor without changing its underlying data. For example, a tensor with 4 elements can be reshaped into a 2x2 matrix or a 1D vector. However, the total number of elements must remain the same.

In Part 3, we applied various tensor operations. TensorFlow supports arithmetic operations like addition, subtraction, multiplication, and division directly on tensors. We also performed matrix multiplication using `tf.matmul()`, which is useful in neural networks, where weights and inputs are often represented as matrices.

Finally, we discussed why reshaping is important. Reshaping tensors helps adjust the data's structure for specific machine learning models, batch processing, or to meet the input/output requirements of layers in neural networks. The number of elements must always stay the same when reshaping, and this allows for flexibility in handling different data forms without altering the actual data values.

Fun Facts

Did you know? When reshaping tensors, the total number of elements must remain the same! For example, a tensor with 4 elements can be reshaped into (2, 1, 2), but not (2, 1, 1), since the number of elements would not match. Tensors can be reshaped to fit various dimensions based on how you need to structure the data.

Code Used in Lesson 1

```
# Lesson 1, Part 1 - Basic TensorFlow Program

import tensorflow as tf
print("TensorFlow version:", tf.__version__)
```

```
tensor_a = tf.constant(5)

tensor_b = tf.constant(3)

result = tf.add(tensor_a, tensor_b)

print("Result of addition:", result)


# Lesson 1, Part 2 - Tensor Types and Shapes


int_tensor = tf.constant(10, dtype=tf.int32)

float_tensor = tf.constant(10.5, dtype=tf.float32)

string_tensor = tf.constant("Hello TensorFlow")

bool_tensor = tf.constant(True)

print("Integer Tensor:", int_tensor)

print("Float Tensor:", float_tensor)

print("String Tensor:", string_tensor)

print("Boolean Tensor:", bool_tensor)


scalar_tensor = tf.constant(42)

vector_tensor = tf.constant([1, 2, 3, 4])

matrix_tensor = tf.constant([[1, 2], [3, 4]])

print("Scalar Tensor:", scalar_tensor)

print("Vector Tensor:", vector_tensor)

print("Matrix Tensor:", matrix_tensor)


# Reshaping a tensor

reshaped_tensor = tf.reshape(vector_tensor, (2, 2))

print("Reshaped Tensor:", reshaped_tensor)
```

```
# Lesson 1, Part 3 - Tensor Operations


add_result = tf.add(tensor_a, tensor_b)

multiply_result = tf.multiply(tensor_a, tensor_b)

subtract_result = tf.subtract(tensor_a, tensor_b)

divide_result = tf.divide(tensor_a, tensor_b)

print("Addition Result:", add_result)

print("Multiplication Result:", multiply_result)

print("Subtraction Result:", subtract_result)

print("Division Result:", divide_result)


# Matrix multiplication

matrix_1 = tf.constant([[1, 2], [3, 4]])

matrix_2 = tf.constant([[5, 6], [7, 8]])

matrix_mult_result = tf.matmul(matrix_1, matrix_2)

print("Matrix Multiplication Result:", matrix_mult_result)


# Other operations

sum_result = tf.reduce_sum(matrix_1)

mean_result = tf.reduce_mean(matrix_1)

max_index = tf.argmax(vector_tensor)

print("Sum of elements:", sum_result)

print("Mean of elements:", mean_result)

print("Index of max value:", max_index)
```