Name : Abrar Hussain
USN : 1RVU22CSE004

Ex No: 10 Date: 30-10-2024	Lab 10: Practice optimizing and deploying deep learning models

Objective:
The objective of this lab is to explore the process of optimizing and deploying neural networks in various formats using TensorFlow, TensorFlow Lite, ONNX (Open Neural Network Exchange), and TensorRT. This lab examines how each format affects model performance, especially inference time, by testing the models on the MNIST dataset.

Descriptions:
In this lab, three different model conversion techniques were applied to a Convolutional Neural Network (CNN) trained on the MNIST dataset. Each approach serves to optimize the model for specific deployment scenarios, such as mobile devices (TensorFlow Lite), cross-framework compatibility (ONNX), and inference speed improvement (TensorRT). After training, the models were tested to compare inference times in their original and optimized formats.

Steps to Build the Model:

Import Libraries:

Import necessary libraries including TensorFlow, TensorFlow Lite, ONNX, ONNX Runtime, and TensorRT. Additionally, libraries like NumPy, Matplotlib, and time are used for data handling and performance measurement.

Data Preparation:

Load the MNIST dataset and normalize the images to a [0, 1] scale. Reshape images to add a channel dimension as required by the CNN model.

```
mnist = tf.keras.datasets.mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0
```

Model Architecture:
Define a basic CNN with layers to process 28x28 grayscale MNIST images. The model includes:

- Two Conv2D layers for feature extraction.
- MaxPooling layers for downsampling.
- Dense layers for classification.

Name : Abrar Hussain
USN : 1RVU22CSE004

Model Training:

Compile and train the CNN model using TensorFlow with sparse categorical cross-entropy loss and accuracy as metrics. A brief training is performed to ensure the model is functional for conversion tests.

Model Conversion and Optimization

TensorFlow Lite Conversion:
Conversion: Convert the trained TensorFlow model to TensorFlow Lite format, enabling quantization for reduced model size and improved inference on mobile devices.

Testing: Load the converted model with TensorFlow Lite's interpreter and test its inference on sample images.

Results: Display the test image, the model's predicted class, and its confidence level. Plot the model's performance by comparing the output of the quantized model to the original TensorFlow model.

ONNX Conversion
Conversion: Use tf2onnx to convert the trained model to ONNX format. Check the model for compatibility using ONNX's validation tools.

```
onnx_model_path = "mnist_cnn_model.onnx"
spec = (tf.TensorSpec((None, 28, 28, 1), tf.float32, name="input"),)
model_proto,        _        =        tf2onnx.convert.from_keras(model,
input_signature=spec, output_path=onnx_model_path)
print(f"Model exported to {onnx_model_path}")
```

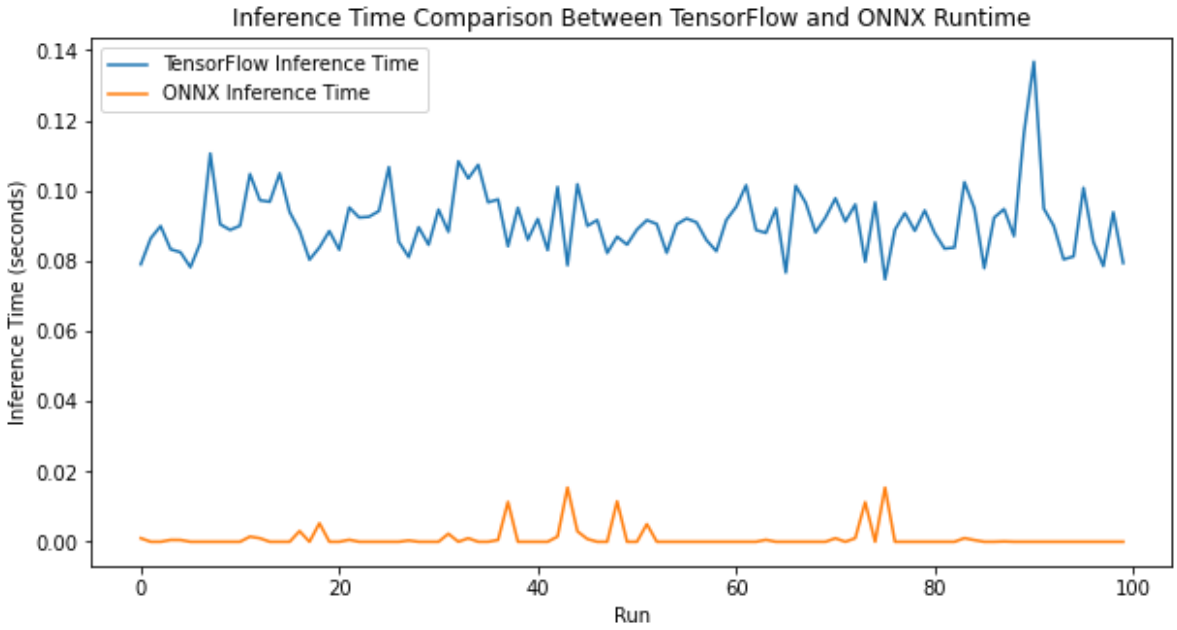
Inference with ONNX Runtime: Test the ONNX model with ONNX Runtime to compare inference speeds with TensorFlow.

```
def      measure_inference_time(tf_model,      ort_session,      x_test,
num_runs=100):
    tf_times = []
    onnx_times = []

    for _ in range(num_runs):
                                                    test_image       =
```

Name : Abrar Hussain
USN : 1RVU22CSE004

```
np.expand_dims(x_test[np.random.randint(len(x_test))], axis=0)

        start_time = time.time()
        tf_output = tf_model.predict(test_image)
        tf_times.append(time.time() - start_time)

        start_time = time.time()
                    onnx_output = ort_session.run(None, {'input':
test_image.astype(np.float32)})
        onnx_times.append(time.time() - start_time)

    return tf_times, onnx_times

tf_times, onnx_times = measure_inference_time(model, ort_session,
x_test)
```

Performance Comparison: Measure and plot inference times for both TensorFlow and ONNX Runtime models, comparing the average performance across multiple runs.



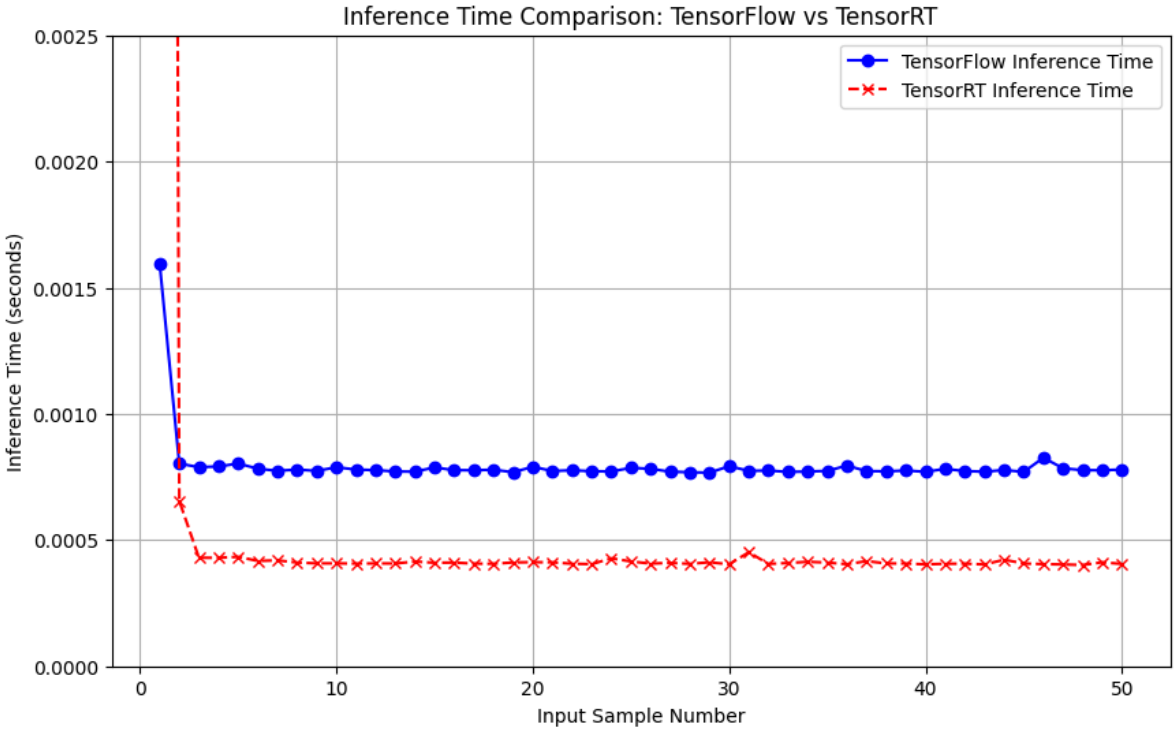
TensorRT Conversion

Conversion: Save the trained TensorFlow model and convert it to TensorRT format, optimizing it for high-performance inference.

Name : Abrar Hussain
USN : 1RVU22CSE004

Inference with TensorRT: Test the model's inference time with TensorRT and compare it to the original TensorFlow model.

Visualization



Github link :
https://github.com/abraaaar/RVU_BtechHons/tree/main/Deep%20Learning/Lab%2010