Lab Report: **TensorFlow and PyTorch**

Adham Ali Mohamed Abdelfattah
Student ID: 22404698
Course: Advanced Programming
Instructor: Prof. Tobias Schaffer

June 14, 2025

1 Introduction

This lab investigates and compares the TensorFlow and PyTorch deep learning frameworks. The goal is to train and evaluate a simple model on the MNIST dataset, export it to lightweight formats (TFLite and ONNX), and assess its performance with ONNX Runtime.

2 Methodology

A feedforward neural network was implemented separately in both TensorFlow and PyTorch. Each model was trained on the MNIST dataset for 5 epochs, and the evaluation was based on accuracy and inference speed. Models were then exported to TFLite (TensorFlow) and ONNX (PyTorch) formats.

2.1 Software and Hardware Used

- Programming language: Python 3.12
- Libraries: TensorFlow 2.16.2, PyTorch 2.2.2, ONNX, ONNXRuntime, TorchVision, Matplotlib
- Hardware: Apple MacBook Air (Intel Core i5, CPU only)

2.2 Code Repository

The full source code is publicly available on GitHub at:

`https://github.com/adhamali74/TensorFlow_and_PyTorch`

This repository includes:

- Source code for TensorFlow and PyTorch models
- Exported models in TFLite and ONNX formats

- Inference code using ONNX Runtime
- Logs and training results
- Lab report and project README

2.3 Code Implementation

TensorFlow Example:

```
model = tf.keras.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28)),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(10)
])
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(
                  from_logits=True),
              metrics=['accuracy'])
model.fit(x_train, y_train, epochs=5)
```

PyTorch Example:

```
class SimpleNN(nn.Module):
    def __init__(self):
        super(SimpleNN, self).__init__()
        self.fc1 = nn.Linear(28*28, 128)
        self.relu = nn.ReLU()
        self.fc2 = nn.Linear(128, 10)
    def forward(self, x):
        x = x.view(-1, 28*28)
        x = self.relu(self.fc1(x))
        return self.fc2(x)
```

3 Results

- TensorFlow test accuracy: **97.14%**
- PyTorch test accuracy: **97.39%**
- ONNX inference time: **0.0452 seconds**

4 Challenges, Limitations, and Error Analysis

4.1 Challenges Faced

- Exporting to TFLite required a workaround due to missing input signature compatibility.
- PyTorch ONNX export needed careful input shape formatting.

4.2 Error Analysis

- Initial TensorFlow TFLite conversion errors due to ___get_save_spec not found.
- ONNXRuntime needed consistent image format (NCHW).

4.3 Limitations of the Implementation

- Training was done on CPU only (Intel i5 chip), so execution was slower.
- Only basic feedforward networks were compared.

5 Discussion

Both frameworks provided comparable accuracy, but differed in usability. TensorFlow was more complex to export to TFLite, while PyTorch's ONNX integration was smoother. ONNXRuntime outperformed both native runtimes in inference speed.

6 Conclusion

This lab demonstrated successful training, evaluation, and export of deep learning models in TensorFlow and PyTorch. PyTorch showed faster export ease, and ONNXRuntime provided the best inference speed, highlighting the value of format interoperability.

7 References

- TensorFlow: `https://www.tensorflow.org/`
- PyTorch: `https://pytorch.org/`
- ONNX: `https://onnx.ai/`