Comparative Analysis of TensorFlow and PyTorch for MNIST Digit Classification

This project presents a comparative implementation and evaluation of two widely-used deep learning frameworks--TensorFlow and PyTorch--applied to the MNIST handwritten digit classification task. The primary objective is to assess performance, usability, and deployment capabilities of both frameworks under a controlled experimental setup.

Author

Vadlamudi Sai Krishna

Student ID: 12503080

Advanced Programming -- M.Sc. Program

Instructor: Prof. Tobias Schaffer

Submission Date: 07.07.2025

Project Overview

The study implements an identical feedforward neural network in both TensorFlow and PyTorch, trained on the MNIST dataset. Key evaluation metrics include:

- Training Time
- Test Accuracy
- Inference Time
- Ease of Deployment (TFLite / ONNX)

Additionally, each model is exported into a lightweight format suitable for deployment on embedded systems.

Model Architecture

- Input Layer: 784 nodes (28×28 pixels)
- Hidden Layer: 64 ReLU units
- Output Layer: 10 nodes (digit classes 0-9)

Environment and Tools

- Language: Python 3
- Frameworks: TensorFlow, PyTorch, NumPy

- Hardware:
  - CPU: Intel Core i7
  - GPU: NVIDIA Tesla T4 (Google Colab)
- Execution Platforms: Google Colab (cloud-based) and local environment

Repository Contents

- tensorflow_model.py - TensorFlow implementation
- pytorch_model.py - PyTorch implementation
- model.tflite - Exported TensorFlow Lite model
- model.onnx - Exported ONNX model from PyTorch
- Lab03_TensorFlow_vs_PyTorch.ipynb - Combined Jupyter notebook (optional)
- README.md - Project documentation

Summary of Results

Framework | Training Time | Test Accuracy | Inference Time

----------|---------------|---------------|----------------

TensorFlow | 24.97 seconds | 97.53% | 3.23 seconds

PyTorch | 46.25 seconds | 96.71% | 1.00 second

TensorFlow demonstrated faster training, while PyTorch achieved faster inference during evaluation.

Conclusion

- TensorFlow is more suitable for rapid prototyping and production deployment due to its high-level API and seamless model export (TFLite).
- PyTorch provides greater control, transparency, and flexibility, making it ideal for academic research and custom experimentation.
- Both frameworks achieved strong classification performance on MNIST, with only minor differences in accuracy.

References

- https://www.tensorflow.org
- https://pytorch.org/tutorials

- https://onnx.ai

- https://www.tensorflow.org/lite

Future Work

- Implement quantization techniques for model compression and deploy models on real edge devices

- Expand benchmarking to more complex datasets (e.g., CIFAR-10, ImageNet)

- Investigate distributed training and advanced optimization strategies