Lab03: TensorFlow vs PyTorch **TH Deggendorf - Campus Cham** **Instructor: Prof. Tobias Schaffer** This lab explores the implementation, training, evaluation, and deployment of a simple neural network using **TensorFlow** and **PyTorch**, allowing a side-byside comparison of both deep learning frameworks. Objective - Build and train a neural network model using TensorFlow and PyTorch - Compare training time and performance - Convert trained models to lightweight formats for embedded deployment: - TensorFlow → TensorFlow Lite (`.tflite`) - PyTorch → ONNX (`.onnx`) Contents

tensorflow_model.py # TensorFlow model implementation and training

— pytorch_model.py # PyTorch model implementation and training
├— logs/ # Training and inference logs
— model.tflite # TensorFlow Lite converted model
— model.onnx # ONNX exported PyTorch model
— Lab03_TensorFlow_vs_PyTorch.ipynb # Reference notebook (with partial implementation)
comparison_report.md # Summary comparison of frameworks
README.md # This file
yaml
Copy code
Dataset
- **MNIST**: 28x28 grayscale images of handwritten digits (10 classes: 0–9)
Model Architecture

- 1. Flatten input: `28x28 → 784`
- 2. Dense layer: `64 units`, ReLU activation
- 3. Output layer: `10 units`
 - **Softmax** (TensorFlow)
 - **Logits** (PyTorch)

Tasks

Task 1: Model Implementation & Training

- Normalize and load MNIST data
- Implement the architecture using:
- `tf.keras.Sequential` in TensorFlow
- Custom `nn.Module` class in PyTorch
- Train both models for **5 epochs**
- Record and compare **training time**

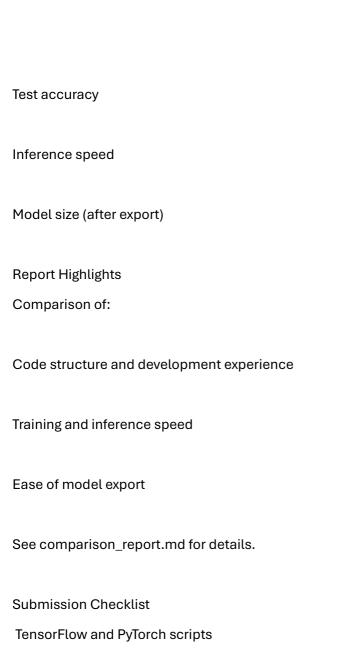
Task 2: Inference & Evaluation

- Evaluate models on test data
- Measure **accuracy** and **inference time**

```
- Use:
 - `model.evaluate()` in TensorFlow
 - `model.eval()` + `torch.no_grad()` in PyTorch
### Task 3: Model Conversion
#### TensorFlow → TensorFlow Lite
```python
converter = tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model = converter.convert()
with open('model.tflite', 'wb') as f:
 f.write(tflite_model)
PyTorch → ONNX
python
Copy code
dummy_input = torch.randn(1, 784)
torch.onnx.export(model, dummy_input, "model.onnx",
 input_names=["input"], output_names=["output"])
```

**Evaluation Metrics** 

Training time



Training and inference logs

model.tflite and model.onnx files

Comparison report

Author

Praneeth Katkam

Student at TH Deggendorf – Campus Cham

Course: Applied AI for Digital Production Management

Instructor: Prof. Tobias Schaffer