

A Comparative Analysis of Lab View and PyTorch for Machine Learning: The gap between Experimentation and Production

Archana Narayanan, Vishrut Jha, Joanne Anto
NightHack Technology, Bengaluru, Karnataka.

Abstract- This paper presents a comparative analysis of handwritten digit recognition performance between LabVIEW and PyTorch frameworks, utilizing a Convolutional Neural Network (CNN). The model is designed to classify digits from the MNIST dataset, which consists of 28×28 grayscale images of handwritten digits (0–9). The dataset includes 60,000 training images and 10,000 test images, providing a standardized benchmark for evaluating model performance. Metrics such as accuracy, training time, memory usage, and inference speed are evaluated. The results provide insight into the strengths and weaknesses of these frameworks in terms of efficiency, scalability, and usability. Results indicate that while both frameworks are effective, PyTorch offers faster training and inference, whereas LabVIEW demonstrates marginally better training accuracy.

Index Terms- Handwritten Digit Recognition, LabVIEW, PyTorch, Convolutional Neural Network (CNN), MNIST Dataset, Model Accuracy, Training Time, Memory Usage, Inference Speed, Efficiency, Scalability, Usability.

I. INTRODUCTION

Handwritten digit recognition is a classic problem in computer vision, widely used to evaluate machine learning frameworks. In this study, the MNIST dataset is used, a benchmark dataset comprising 28×28 grayscale images of handwritten digits ranging from 0 to 9. The dataset includes 60,000 training images and 10,000 test images, enabling robust training and evaluation of deep learning models.

A Convolutional Neural Network (CNN) was implemented to classify these digits, outputting probabilities for each of the 10 classes. This study compares the performance of the CNN model implemented in two different frameworks: LabVIEW and PyTorch. The comparison focuses on key metrics, including model accuracy, training efficiency, memory usage, and inference speed. By analyzing the results, the aim is to identify the strengths and limitations of each framework for deep learning applications.

II. CNN ARCHITECTURE

The CNN architecture used in this study is illustrated in Figure 1. It consists of the following components:

- **Input Layer:** Accepts grayscale images of size 28×28
- **Convolutional Layers:** Two convolutional layers, each with a kernel size of 3×3 , followed by ReLU activation functions.

- **Max Pooling Layers:** Two pooling layers with a window size of 2×2 for down-sampling.
- **Flatten Layer:** Transforms the 3D feature maps into a 1D vector.
- **Fully Connected Layers:**
 - The first fully connected layer has 200 neurons with ReLU activation.
 - Dropout with a rate of 50% is applied to reduce overfitting.
 - The final fully connected layer outputs 10 neurons (one for each class).
- **Output Layer:** Applies a Log Softmax activation function to generate class probabilities.

Both implementations used the same architecture to ensure a fair comparison.

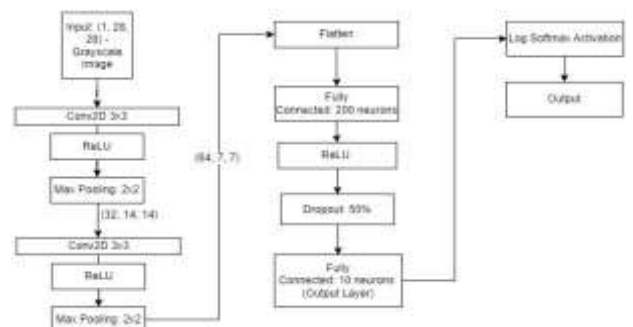


Figure 1: CNN Architecture Diagram

III. COMPARISON METRICS

1. Model Accuracy

The comparison of model accuracy between LabVIEW and PyTorch highlights their reliability in handwritten digit recognition. Both frameworks achieved high accuracy levels. The model accuracy for a given architecture appears largely independent of the deployment platform, whether it is PyTorch or LabVIEW. However, variations in accuracy values between LabVIEW and PyTorch can be observed across iterations and during the training process, including differences in the total number of epochs required to reach convergence, PyTorch converged at the 14th epoch and LabVIEW at the 12th epoch. The final test accuracy, training accuracy, and validation accuracy for LabVIEW and PyTorch implementations are summarized in Table 1.

Table 1: Model Accuracy

Metric	LabVIEW Value	PyTorch Value	Unit	Description
Final Test Accuracy	99.22	99.38	%	Accuracy achieved on the test set.
Training Accuracy (Final)	99.39	99.21	%	Accuracy on training set at final epoch.
Validation Accuracy (Final)	99.40	99.43	%	Accuracy on validation set at final epoch.

2. Training Time

LabVIEW exhibited longer training durations per epoch, taking 46.5 seconds compared to PyTorch's 33.6 seconds. While PyTorch converged at the 14th epoch and LabVIEW at the 12th epoch, PyTorch demonstrated faster training and inference speeds overall. This can be attributed to its optimized libraries and efficient hardware utilization, resulting in a lower average time per batch (0.0716 seconds) and reduced total training time (470.4 seconds) compared to LabVIEW's 558 seconds.

Table 2: Training Time

Metric	LabVIEW Value	PyTorch Value	Unit
Time per Epoch	46.5 (12 epoch)	33.6 (14 epoch)	Seconds/Epoch
Total Training Time	558	470.4	Seconds
Average Time per Batch	0.104	0.0716	Seconds/Batch

3. Memory Usage

PyTorch exhibited higher memory utilization, leveraging GPU acceleration, whereas LabVIEW had lower but CPU-centric usage. The memory usage during the training process reflects the operations involved in both the forward and backward passes of the model. In each epoch, memory is used to store intermediate activations during the forward pass, compute gradients during backpropagation, and update the model's weights. The peak memory usage occurs during training as the model processes each batch of data, performing computations across the convolutional and fully connected layers, while gradients are also stored for the backpropagation step. During the validation phase, memory usage is slightly reduced since no gradients are calculated, but it still requires memory to compute the forward pass on the validation set and track the loss and accuracy metrics. The observed memory utilization, therefore, reflects the cumulative requirements of these operations, with memory usage being highest during training and slightly lower during validation.

Table 3: Memory Usage

Metric	LabVIEW Value	PyTorch Value	Unit
Maximum Memory Usage	396.4 MB	641.47 MB	MB
Average Memory Usage	383 MB	595.9 MB	MB

4. Inference Speed

Inference times for PyTorch were significantly faster, making it preferable for real-time applications.

Table 4: Inference Speed

Metric	LabVIEW Value	PyTorch Value	Unit
Time per Batch (Inference)	0.57	0.012685	Seconds/Batch
Time per Sample (Inference)	0.00253	0.000791	Seconds/Sample

IV. LOSS AND ACCURACY TRENDS

1. Loss Over Epochs

The loss trends for LabVIEW and PyTorch reveal key differences in their training behavior and performance. PyTorch demonstrates a smooth and consistent decline in both training and validation losses, converging steadily by the 14th epoch. The losses stabilize with minimal gaps between them, indicating efficient learning and good generalization without overfitting. In contrast, LabVIEW shows an initial rapid decline in loss but experiences a noticeable spike midway through training, suggesting potential instability or interruptions during the process.

Although the losses eventually stabilize, LabVIEW converges slightly earlier, around the 12th epoch, but with a slower rate of overall loss reduction.

PyTorch Loss over Epochs

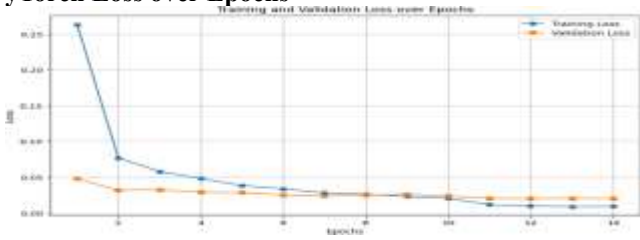


Figure 2: PyTorch Loss Over Epochs

LabVIEW Loss over Epochs

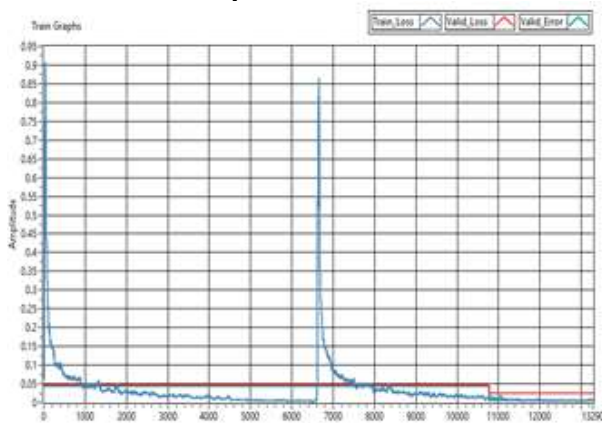


Figure 3: LabVIEW Loss Over Epochs

2. Validation Accuracy Over Epochs

The Validation Accuracy over Epochs plot highlights the performance comparison between PyTorch and LabVIEW. Initially, PyTorch experienced a slower start, showing a lower accuracy compared to LabVIEW during the first few epochs. However, as training progresses, PyTorch steadily improves and achieves a convergence by the 14th epoch and LabView by 12th epoch.

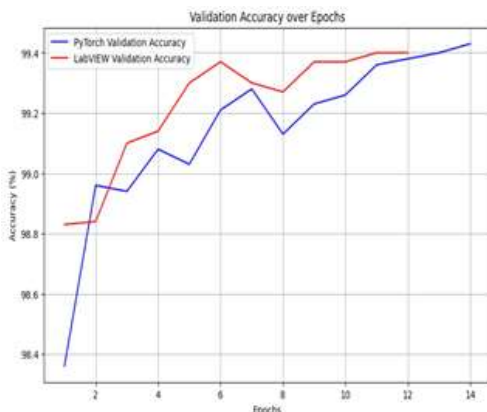


Figure 4: Validation Accuracy Over Epochs

V. CONCLUSION

This study highlights the trade-offs between LabVIEW and PyTorch for handwritten digit recognition using a CNN model.

Training Efficiency: PyTorch was faster in training and inference, making it suitable for large-scale datasets and real-time applications. PyTorch significantly reduced total training time to 470.4 seconds from LabVIEW's 558 seconds. It also cut the time per epoch by 27.7%, allowing for faster training cycles and quicker model improvements.

Memory Usage: LabVIEW was more memory-efficient but at the cost of slower execution. While PyTorch uses more memory on average (595.9 MB vs. LabVIEW's 383 MB), this extra usage helps speed up computation and improve processing times.

Inference Speed: PyTorch significantly improves inference time, reducing batch processing from 0.57 seconds to just 0.012 seconds, making it ideal for real-time applications that need quick decisions.

Resource Utilization: PyTorch proved to be more efficient in utilizing available resources, using less CPU time during both training and inference compared to LabVIEW, which indicates better optimization for hardware resources.

Usability: PyTorch provided superior scalability and debugging tools, making it easier to extend to other tasks and debug complex models.

Future work could involve extending this comparison to other deep learning tasks and incorporating energy consumption metrics.

REFERENCES

1. Seng, L. Ming, et al. "MNIST handwritten digit recognition with different CNN architectures." J. Appl. Technol. Innov. 5.1 (2021): 7-10.
2. Peng, Jiao. "Research on the application of convolutional neural network based on PyTorch framework in handwritten digit recognition." International Conference on Image, Signal Processing, and Pattern Recognition (ISPP 2024). Vol. 13