Accelerating Deep Learning: PyTorch Performance on CPU vs. GPU

Harman Preet Singh

harman.pnahal@gmail.com

Abstract

This study investigates the performance gains of training neural networks in PyTorch using GPU acceleration (CUDA) compared to CPU-based training. Using a range of model architectures — Feedforward Neural Networks (FNN), Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), and Generative Adversarial Networks (GAN) — we measure training time differences between an NVIDIA RTX 3060 Laptop GPU and an AMD Ryzen 5 5600H CPU. The results highlight significant speed improvements for most models when trained on the GPU, though with some notable variations across architectures. These findings improve insights into the efficiency of GPU utilisation for deep learning tasks and its effectiveness across different neural network types.

Contents

1. Introduction

Background

The use of neural networks in machine learning has expanded significantly, with applications ranging from image recognition and natural language processing to generative modelling. As the complexity and size of these models grow, so does the computational demand for training them effectively. Traditionally, central processing units (CPUs) have been used for model training, but graphics processing units (GPUs) are increasingly leveraged for their parallel processing capabilities, allowing for accelerated training, particularly in deep learning frameworks like TensorFlow and PyTorch. Understanding the impact of performances differences between these hardware options is essential for developers and researchers aiming to optimise model training time.

Objective

This study aims to assess the performance differences between training PyTorch models on a GPU (using NVIDIA's CUDA framework) and a CPU. Specifically, we measure training times across several types of neural network architectures, including Feedforward Neural Networks (FNN), Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), and Generative Adversarial Networks (GAN). The goal is to identify which architectures benefit the most from GPU acceleration, as well as any unexpected performance results, such as observed inefficiencies or unexpected gains in specific setups.

Hardware constraints

The hardware used in this study includes an NVIDIA RTX 3060 Laptop GPU and an AMD Ryzen 5 5600H CPU. While the difference in these devices' architectures may introduce some bias, they were selected as the available resources on a standard consumer laptop. The comparison may not be fully generalised to more powerful or comparable hardware setups but provides a practical view of performance on widely available consumer-grade hardware. This report addresses these limitations by focusing on relative improvements in performance rather than absolute benchmarking against high-performance CPUs or GPUs.

2. Experimental setup

Hardware specifications

To evaluate the performance differences between training neural networks on a CPU versus a GPU, the experiments were conducted using consumer-grade hardware. The hardware specifications are as follows:

- **GPU**: NVIDIA RTX 3060 Laptop GPU, configured with CUDA for GPU-accelerated computations, with 6GB of GDDR6 memory and parallel processing capabilities suited for deep learning tasks
- **CPU**: AMD Ryzen 5 5600H, a six-core processor with 12 threads and a base clock speed of 3.3 GHz, reaching up to 4.2 GHz with max boost

These hardware choices allow for a practical comparison between CPU and GPU training on standard, accessible hardware but may not reflect the absolute speedups possible with higher-end GPUs or CPUs

Software and libraries

- **PyTorch**: version 2.5.0
- **CUDA Toolkit**: version 12.6
- **Python**: version 3.12.7
- **Operating system**: The experiments were conducted on Windows 10, though no significant OS-based performance differences are expected for this comparison

These software configuration ensure reproducibility and compatibility with PyTorch's GPU-accelerated libraries, enabling a straightforward comparison between CPU and GPU training

Neural network architectures

To evaluate GPU vs. CPU performance across a range of neural network types, we selected the following architectures:

- **Feedforward Neural Network (FNN)**: A simple, fully connected architecture used in foundational machine learning tasks. The FNN was trained for 100 epochs with a batch size of 128, chosen to assess the base-level efficiency of each device for a relatively straightforward model
- **Convolutional Neural Network**: A commonly used architecture in image classification, trained on the MNIST dataset for 3 epochs with a batch size of 64. The CNN was selected to

assess the comparative efficiency of each device for models that benefit from spatial data processing.

- **Recurrent Neural Network (RNN)**: A model commonly used in sequential data tasks, also trained on the MNIST dataset for 3 epochs with a batch size of 64. This architecture was included to evaluate each device's performance with models that require recurrent operations over sequences.
- **Generative Adversarial Network (GAN)**: A generative model used to produce new data samples from a given distribution, trained on the MNIST dataset for 3 epochs with a batch size of 64. GANs were included to test both devices' ability to handle the complex adversarial training process typical of generative models

Training parameters

Each model was trained with specific parameters to standardise comparisons. Key training parameters included:

- **Epochs**: The number of epochs varied by model type to ensure meaningful training time without excessive computation. For FNN, 100 epochs were chosen to capture training speed on a simpler model, while CNN, RNN, and GAN were trained for 3 epochs on the MNIST dataset, balancing sufficient training time with practical run times for comparison
- **Batch size**: Batch size was set at 128 for FNN and 64 for CNN, RNN, and GAN to ensure each model had adequate data for training while remaining compatible with the memory limits of the GPU.
- **Dataset**: The MNIST dataset, a widely used dataset of handwritten digits, was selected for its compatibility across different neural network types and for its moderate size, making it feasible for both CPU and GPU training without excessive memory demands.

This setup enables a consistent basis for measuring and comparing CPU and GPU performance across multiple neural network architectures, allowing us to analyse both expected speed improvements and any architectural variations in performance between devices.

3. Results

This section presents the training time results for each neural network architecture on both the CPU (AMD Ryzen 5 5600H) and GPU (NVIDIA RTX 3060 Laptop GPU). Training time differences are reported as both absolute times and relative percentage changes, highlighting the comparative efficiency of CUDA-accelerated GPU training versus CPU-only training. Each subsection includes a brief analysis of the results.

3.1 Feedforward Neural Network (FNN)

For the Feedforward Neural Network, training was conducted for 100 epochs, with a batch size of 128.

device	**training time**	**difference**
CPU	~21.9s	
GPU (CUDA)	~15.9s	37.27% faster

The FNN model exhibited a ~37% reduction in training time when trained on the GPU compared to the CPU. This improvement aligns with expectations, as GPUs can handle the parallel processing needs of FNNs efficiently due to their fully connected nature.

3.2 Convolutional Neural Networks (CNN)

The Convolutional Neural Network (CNN) was trained on the MNIST dataset for 3 epochs, using a batch size of 64.

device	**training time**	**difference**
CPU	~58.1s	
GPU (CUDA)	~47.3s	22.90% faster

The GPU offered a ~22% reduction in training time for the CNN model. While still significant, the relative improvement is lower than that seen with the FNN. This result is likely due to the convolutional operations, which benefit from GPU acceleration but require specific optimisations to fully leverage GPU power.

3.3 Recurrent Neural Network (RNN)

For the Recurrent Neural Network (RNN), training was performed on the MNIST dataset for 3 epochs, with a batch size of 64.

device	**training time**	**difference**
CPU	~65.2s	
GPU (CUDA)	~45.8s	42.35% faster

The RNN model demonstrated a substantial speed improvement of ~42% on the GPU. This considerable reduction in training time is likely due to the GPU's ability to handle parallel operations in recurrent layers, which process sequential data with high computational demand. The result suggest that RNNs, even with sequential dependencies, can benefit considerably from GPU acceleration, particularly when batch sizes allow for parallelisation across multiple sequences.

3.4 Generative Adversarial Network (GAN)

The Generative Adversarial Network (GAN) was trained on the MNIST dataset for 3 epochs, using a batch size of 64.

device	**training time**	**difference**
CPU	~51.7s	
GPU (CUDA)	~55.2s	6.40% slower

Surprisingly, the GAN model trained slightly slower on the GPU, showing a ~6% increase in training time compared to the CPU. This unexpected result may stem from the specific architectural requirements of GANs, which involve both a generator and discriminator network competing with each other. Since GAN training is adversarial and can involve irregular computation flows between the two networks, the model may not have fully leveraged GPU parallelism. The slower performance could also be due to overhead from GPU memory transfers or the relatively limited computational demand of the GAN model on the MNIST dataset.

Summary of results

In summary, training on the GPU consistently yielded faster results for the FNN, CNN, and RNN architectures, with speed improvements ranging from approximately 23% to over 42%. However, the GAN model did not benefit from GPU training in this setup, suggesting that GANs may require specific optimisations or higher computational loads to achieve improved performance on GPUs.

4. Future work

This study highlights both the advantages and limitations of GPU acceleration for training different neural network architectures. To further refine our understanding of CPU versus GPU training performance, several areas of future exploration are suggested:

4.1 Additional architectures

Future work could include Autoencoders and Transformer models to assess performance differences for architectures with unique computational demands. Autoencoders, which involve both encoding and decoding processes, could reveal GPU advantages in parallel processing, while Transformers, known for heavy computational requirements may provide insights into the efficiency gains offered by GPUs for attention-based models.

4.2 Scaling up with larger datasets

While this study used the MNIST dataset due to its accessibility and moderate size, more complex datasets such as CIFAR-10 or ImageNet could be used to evaluate whether larger, more computationally intensive data leads to even greater relative gains for GPU training. Additionally, training on more complex datasets would further test each architecture's ability to utilise GPU memory and processing power effectively.

4.3 Benchmarking on comparable hardware

To provide a more equitable comparison, future experiments could involve CPUs and GPUs of comparable performance tiers. This would allow for a more controlled evaluation of architectural efficiency without potential hardware imbalance affecting the results.

4.4 Exploring optimisation techniques

Investigating additional optimisation techniques such as mixed-precision training, model pruning, or CUDA-specific optimisations, may provide insights into how GPU training times can be further improved for certain architectures, especially those like GANs that showed less benefit from GPU acceleration in this study.

5. Conclusion

This study compared training times for various neural network architectures — Feedforward Neural Network (FNN), Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), and Generative Adversarial Network (GAN) — on a CPU (AMD Ryzen 5 5600H) and a GPU (NVIDIA RTX 3060 Laptop GPU) using the PyTorch framework. The results indicated that GPU acceleration yielded significant improvements for most architectures, with the GPU achieving training speed-ups of 22-42% for the FNN, CNN, and RNN models. However, the GAN model performed slightly slower on the GPU, an unexpected result likely due to its unique adversarial structure and less GPU-optimised computation flow.

These findings suggest that while GPUs generally provide substantial benefits for neural network training, specific architectures like GANs may require targeted optimizations to fully leverage GPU power. Additionally, the degree of performance improvement observed varies significantly with the network structure, highlighting the importance of considering model type and computational needs when selecting hardware for deep learning tasks.

Overall, this study underscores the value of GPU acceleration for deep learning, particularly for computationally intensive architectures. It also reveals that hardware considerations are not universally applicable across all model types and that understanding these differences can help practitioners make informed decisions when deploying models on different devices. Future research in this area could provide more insights into optimizing neural network training, further refining the balance between training speed, hardware costs, and energy efficiency.