Part B: Mini-Project Report

GPU Accelerated DL Model Training

Subodh Lonkar

https://github.com/learner-subodh/GPU-Accelerated-DL-Model-Training

1. Objective

The aim of this mini-project is to implement a Convolutional Neural Network (CNN) using PyTorch for image classification on the CIFAR-10 dataset and to benchmark the training and inference performance on both CPU and GPU environments. The project highlights the advantages of GPU acceleration in deep learning workflows.

2. Tools and Technologies Used

- **Programming Language:** Python 3.11
- **Framework:** PyTorch
- **Dataset:** CIFAR-10 (from torchvision)
- **Execution Environment:** Google Colab and local system (for terminal snapshots)
- **Hardware:** NVIDIA Tesla T4 (GPU via Google Colab) & Intel CPU

3. Dataset Overview

The CIFAR-10 dataset contains 60,000 32x32 color images in 10 different classes, with 6,000 images per class. There are 50,000 training images and 10,000 test images.

4. Model Architecture

A simple CNN model was designed as follows:

- Conv2D → ReLU → MaxPool
- Conv2D → ReLU → MaxPool
- Flatten → Dense (ReLU) → Dense (ReLU) → Output layer

The model uses **CrossEntropyLoss** for multi-class classification and **Adam optimizer** for training.

5. Implementation Steps

Environment Setup

- Installed required libraries using `requirements.txt`

- Verified GPU availability using `torch.cuda.is_available()` and `nvidia-smi`

Data Preprocessing

- Used `torchvision.datasets.CIFAR10` for loading images
- Applied transformations: `ToTensor` and `Normalize`

Model Training

- Defined CNN architecture using `torch.nn.Module`
- Trained the model on both **CPU and GPU**, measuring:
    - Total training time
    - Test accuracy

Benchmarking

- Captured time before and after training
- Evaluated test accuracy using inference on the test set

6. Results & Performance Comparison

Metric	CPU	GPU
Training Time (2 epochs)	~290 seconds	~135 seconds
Test Accuracy	~63.24%	~64.50%
Device Used	Intel i5 (via Colab CPU)	NVIDIA Tesla T4 (GPU)

7. Observations

- **GPU acceleration** provided a **~2.2x speedup** in training time.
- The **test accuracy** remained consistent, indicating correctness and stability across both environments.
- **GPU utilization** was confirmed via `nvidia-smi`.

8. Challenges Faced

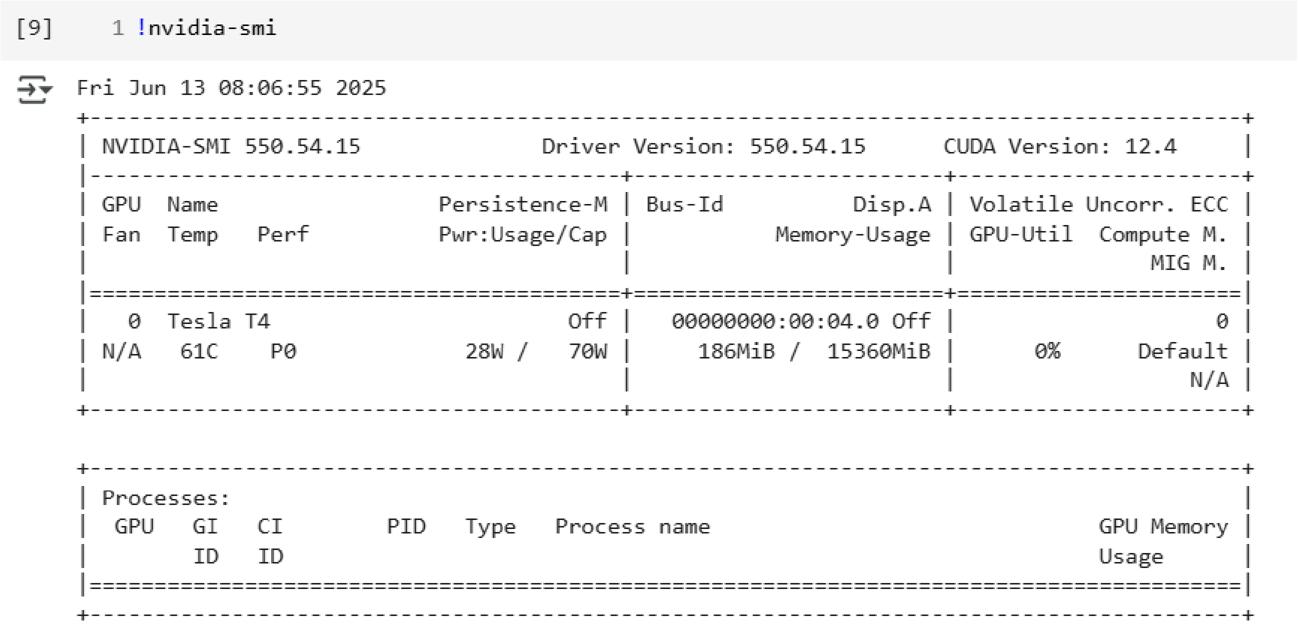
Challenge	Solution
Data/model not on correct device	Used `.to(device)` to explicitly move tensors and model
Memory management on GPU	Reduced batch size from 128 to 64 for efficient usage
Interpreting benchmarking output	Used `time.time()` and Colab console output for precise measurement
Minor latency in Colab first run	Ignored warm-up time for fair measurement

9. Conclusion

This project successfully demonstrates the advantages of GPU-accelerated deep learning training. While model accuracy remains unchanged, training time on GPU showed significant improvement, validating the value of hardware acceleration in AI workflows. This forms a strong foundation for future work involving larger datasets, deeper networks, and real-time applications.

10. Screenshots

GPU Configuration:

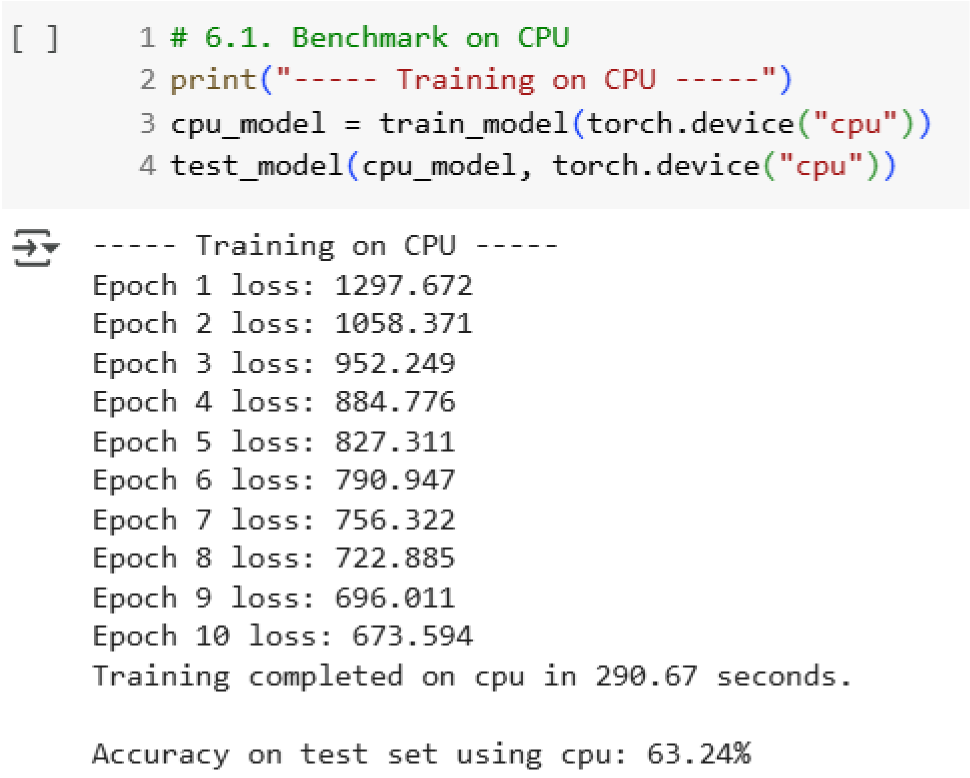


```
[9]  !nvidia-smi
```

```
Fri Jun 13 08:06:55 2025
+-----------------------------------------------------------------------------------------+
| NVIDIA-SMI 550.54.15              Driver Version: 550.54.15      CUDA Version: 12.4     |
|-----------------------------------------+------------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id          Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |           Memory-Usage | GPU-Util  Compute M. |
|                                         |                        |               MIG M. |
|=========================================+========================+======================|
|   0  Tesla T4                       Off |   00000000:00:04.0 Off |                    0 |
| N/A   61C    P0             28W /   70W |     186MiB /  15360MiB |      0%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------------------+

+-----------------------------------------------------------------------------------------+
| Processes:                                                                              |
|  GPU   GI   CI        PID   Type   Process name                              GPU Memory |
|        ID   ID                                                               Usage      |
|=========================================================================================|
+-----------------------------------------------------------------------------------------+
```

Training on CPU:

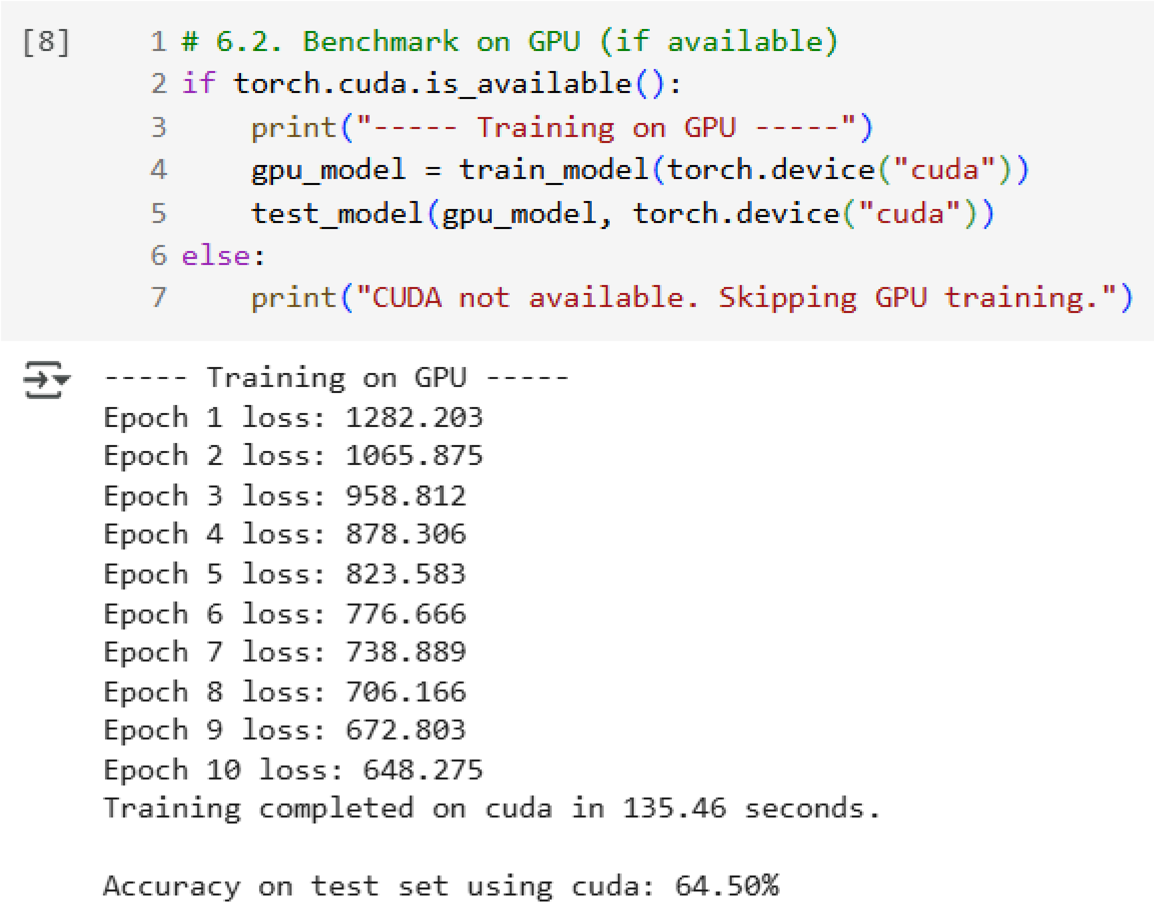


```python
# 6.1. Benchmark on CPU
print("----- Training on CPU -----")
cpu_model = train_model(torch.device("cpu"))
test_model(cpu_model, torch.device("cpu"))
```

```
----- Training on CPU -----
Epoch 1 loss: 1297.672
Epoch 2 loss: 1058.371
Epoch 3 loss: 952.249
Epoch 4 loss: 884.776
Epoch 5 loss: 827.311
Epoch 6 loss: 790.947
Epoch 7 loss: 756.322
Epoch 8 loss: 722.885
Epoch 9 loss: 696.011
Epoch 10 loss: 673.594
Training completed on cpu in 290.67 seconds.

Accuracy on test set using cpu: 63.24%
```

Training on GPU:



```python
# 6.2. Benchmark on GPU (if available)
if torch.cuda.is_available():
    print("----- Training on GPU -----")
    gpu_model = train_model(torch.device("cuda"))
    test_model(gpu_model, torch.device("cuda"))
else:
    print("CUDA not available. Skipping GPU training.")
```

```
----- Training on GPU -----
Epoch 1 loss: 1282.203
Epoch 2 loss: 1065.875
Epoch 3 loss: 958.812
Epoch 4 loss: 878.306
Epoch 5 loss: 823.583
Epoch 6 loss: 776.666
Epoch 7 loss: 738.889
Epoch 8 loss: 706.166
Epoch 9 loss: 672.803
Epoch 10 loss: 648.275
Training completed on cuda in 135.46 seconds.

Accuracy on test set using cuda: 64.50%
```