Omer Mustafa – i221180 Fahad Ahmad – i221087

https://github.com/fahadahmad9/HPC\_Project.git

Optimized cuda implementation for mnist neural network

HPC Project

**INTRODUCTION**

This report details the changes, speed improvements, and key optimizations across four versions (V1 to V4) of a neural network trained on the MNIST dataset. Execution times per epoch and hardware configurations are analyzed to explain performance trends.

**VERSION OVERVIEW**

1. **V1 (CPU Implementation)**

* Hardware: Ryzen 9 5900HS (CPU-only).
* Key Features:
  + Sequential processing with nested loops for matrix operations.
  + No GPU or parallelization.
  + Processes one sample at a time (no batching).
* Performance: 15 seconds/epoch on average.

1. **V2 (Initial GPU Port)**

* Hardware: RTX 3060 (GPU).
* Changes from V1:
  1. CUDA Integration:
     + Kernels for matrix multiplication, ReLU, softmax, and backpropagation.
     + Memory allocation and transfers between CPU/GPU.
  2. Per-Sample Processing:
     + Each sample is processed individually (no batch optimization).
     + Frequent host-device memory transfers for every sample.
  3. Naive Kernel Design:
     + Suboptimal grid/block configurations.
     + No utilization of advanced GPU features.
* Performance: 57 seconds/epoch (slower than V1).
* Speedup Analysis:
  1. Overhead from frequent memory transfers and inefficient kernel launches negated GPU parallelism benefits.

1. **V3 (Batch-Optimized GPU)**

* Hardware: RTX 3060 (GPU).
* Changes from V2:
  1. Batch Processing (BATCH\_SIZE = 128):
     + Aggregates gradients over a batch before updating weights.
     + Reduces kernel launches and memory transfers.
  2. Optimized Kernels:
     + Batched kernels for forward/backward passes (e.g., forward\_hidden\_batch, update\_W2\_batch).
     + Efficient grid/block configurations (e.g., dim3 hiddenBlock(256, 1)).
  3. Atomic Operations:
     + Custom atomicAddDouble for double-precision reductions.
  4. Memory Efficiency:
     + Preallocates device memory for batches, minimizing transfers.
* Performance: 7 seconds/epoch (8.14x speedup over V2).
* Speedup Drivers:
  1. Batch processing reduced kernel launch overhead.
  2. Parallel gradient aggregation and optimized memory access patterns.

1. **V4 (Tensor Core Experiment)**

* **V4’s Extreme Speedup**:
  + **100x faster than V1 (CPU)** and **46.6x faster than V3** due to:
    - Aggressive Tensor Core optimizations (e.g., forced FP16/FP32 mixed precision).
    - Radical kernel redesign (e.g., fused operations, reduced synchronization).
    - **Accuracy Collapse**: Training accuracy drops to **<30%** due to:
* **Numerical instability** from improper mixed-precision handling (e.g., loss scaling skipped).
* **Overly simplified model architecture** to meet Tensor Core constraints (e.g., reduced hidden layer size).
* **V3 vs. V4 Trade-off**:
* **Speed**: V4 achieves unprecedented speed (**0.15s/epoch**) but sacrifices learning capability.
* **Accuracy**: V4’s accuracy is unusable for MNIST, while V3 balances speed (7s) and accuracy (~94%).
* **Root Cause of V4’s Failure**:
  + **Improper Mixed-Precision Training**: Missing loss scaling or gradient clipping led to vanishing/exploding gradients.
  + **Tensor Core Misalignment**: Input/hidden layer dimensions (784/128) do not align with Tensor Core tiles (16x16), causing approximation errors.
  + **Over-Optimization**: Kernel fusion or loop unrolling likely introduced computation errors.

**PERFORMANCE SUMMARY**

| **Version** | **Hardware** | **Time/Epoch (Average)** | **Speedup vs. Previous** | **Key Change** |
| --- | --- | --- | --- | --- |
| V1 | Ryzen 9 5900HS | 15s | Baseline | CPU Sequential |
| V2 | RTX 3060 | 57s | 0.25x | Naive GPU Port |
| V3 | RTX 3060 | 7s | 8.14x | Batch Processing |
| V4 | RTX 3060 | 0.15s | 46.6x | Tensor Core |

**KEY TAKEAWAYS**

1. **V2 (GPU) Slower Than V1 (CPU):**
   * Lack of batching and excessive memory transfers made GPU overhead worse than CPU efficiency.
2. **V3’s Success:**
   * Batch processing and optimized kernels maximized GPU parallelism, achieving a 8.14x speedup over V2.
3. **V4’s Failure:**
   * Tensor Cores underperformed due to FP64 usage and unaligned matrix dimensions. FP16/FP32 would yield better results.