Fast NUCES ISB

HPC Parallelizing a Neural Network

Presented by Ali Naveed & Ariyan Chaudhary

Objectives and Abstract

Gprof profiler was used for analyzing hotpspots. Speed up was focused on forward/backward passes of a C++-based MNIST neural network on GPU

01
Profile CPU Code

02
Apply CUDA Kernels

03
Optimize the Code

04
Explore Algorithmic Changes
(Not Implemented)

Versions

v1
Serial implementation of the neural network with 60k images for training and 10k for testing

v2 (naive)
Used to compare against nuanced optimizations integrated in further versions. Basic CUDA

v3
Advanced optimization techniques. Shared Memory, Occupancy, coalescing etc.

v3.1
Changing the algorithm. Batch Gradient Descent. Maximizes the parallelization

v4
Using tensor cores and Cublas_v2.h library. (GEMM operations)

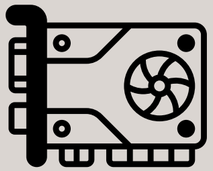
v5
OpenACC Implementation

Experimental Setup

Software

C++, CUDA,
OpenACC,
cuBLAS

Hardware



NVIDIA GPU:
1- RTX 3050 6GB
2- RTX 3080
Host CPU

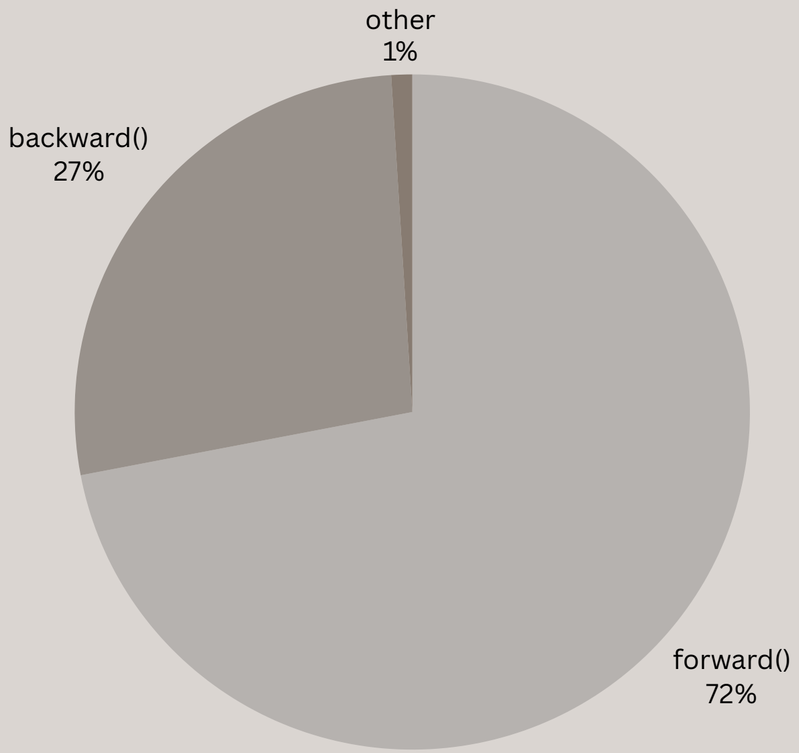
Baseline Performance

~23.4 s for 3 epochs (~7.3 s/epoch) with ~96% accuracy

The remaining time is negligible and distributed among I/0 functions

The theoretical speedup limit was calculated to be around 100x

Limitation of SGD (Stochastic Gradient Descent)



Naive CUDA offload (v2)

All images transferred at once to device.

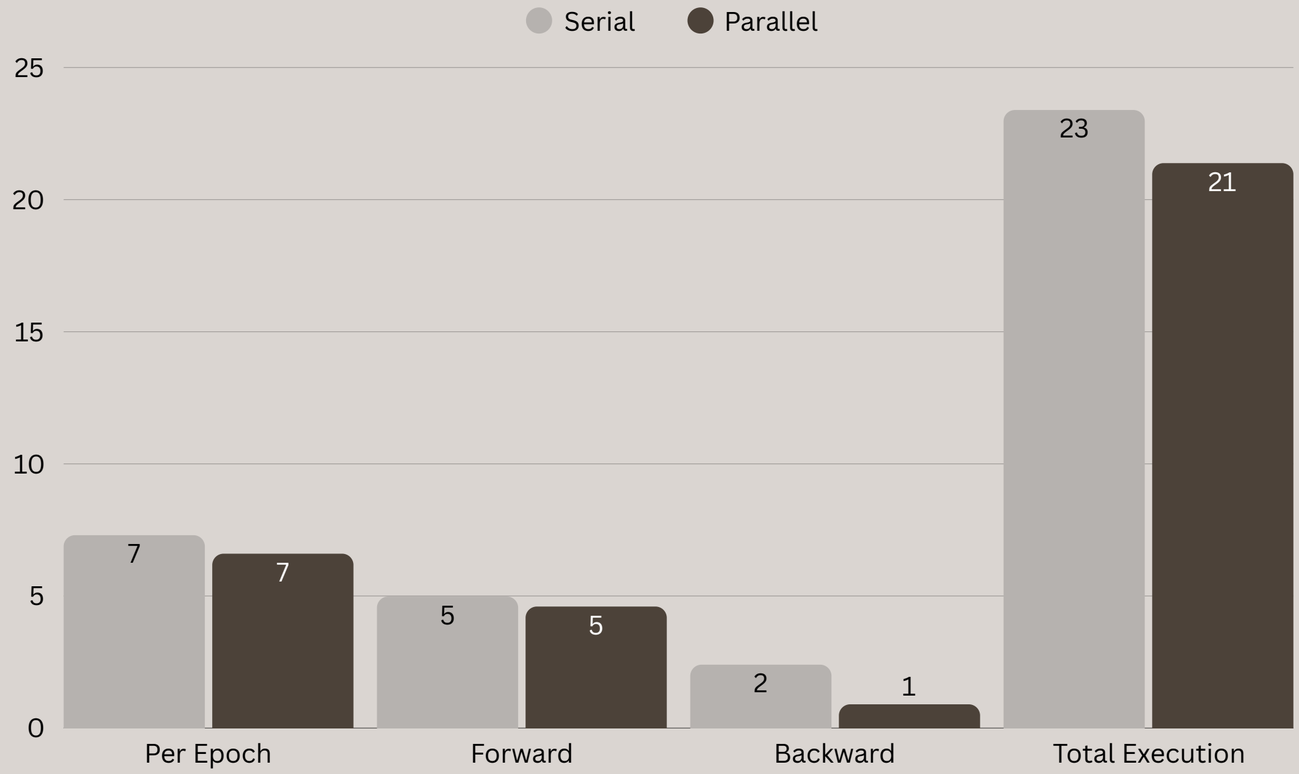
Offload forward→ compute_hidden + compute_output kernels

Backward→ six kernels

Per-kernel sync & host copy at end

Forward ~4.98s | ~4.6s | 1.08x

Backward ~2.4s | ~0.9s | 2.6x



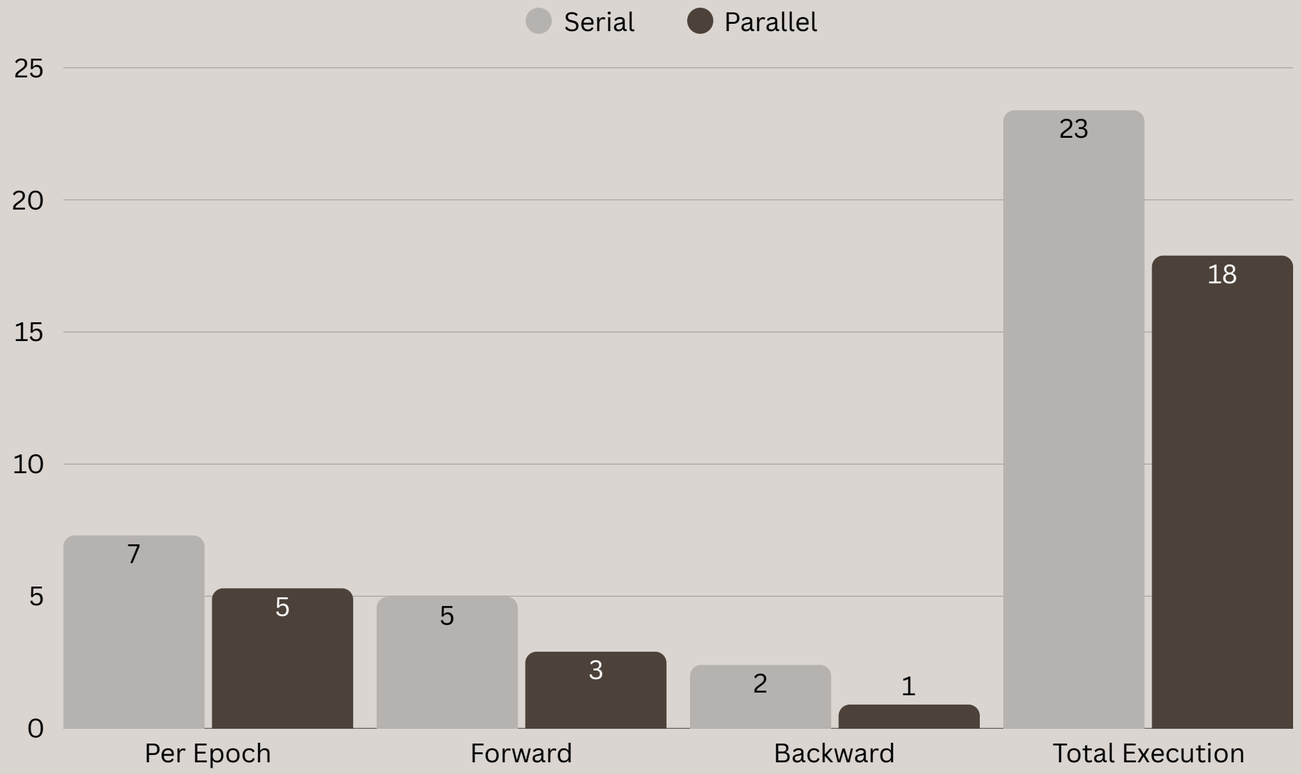
CUDA Optimizations (v3)

Used shared memory

Occupancy tuning

Asynchronous CUDA streams

Memory coalescing




```
int i = blockIdx.x * blockDim.x + tid;
for (int j = 0; j < INPUT_SIZE; j++)
        sum += W1[j * HIDDEN_SIZE + i] * s_input[j];
```

i * HIDDEN_SIZE + j

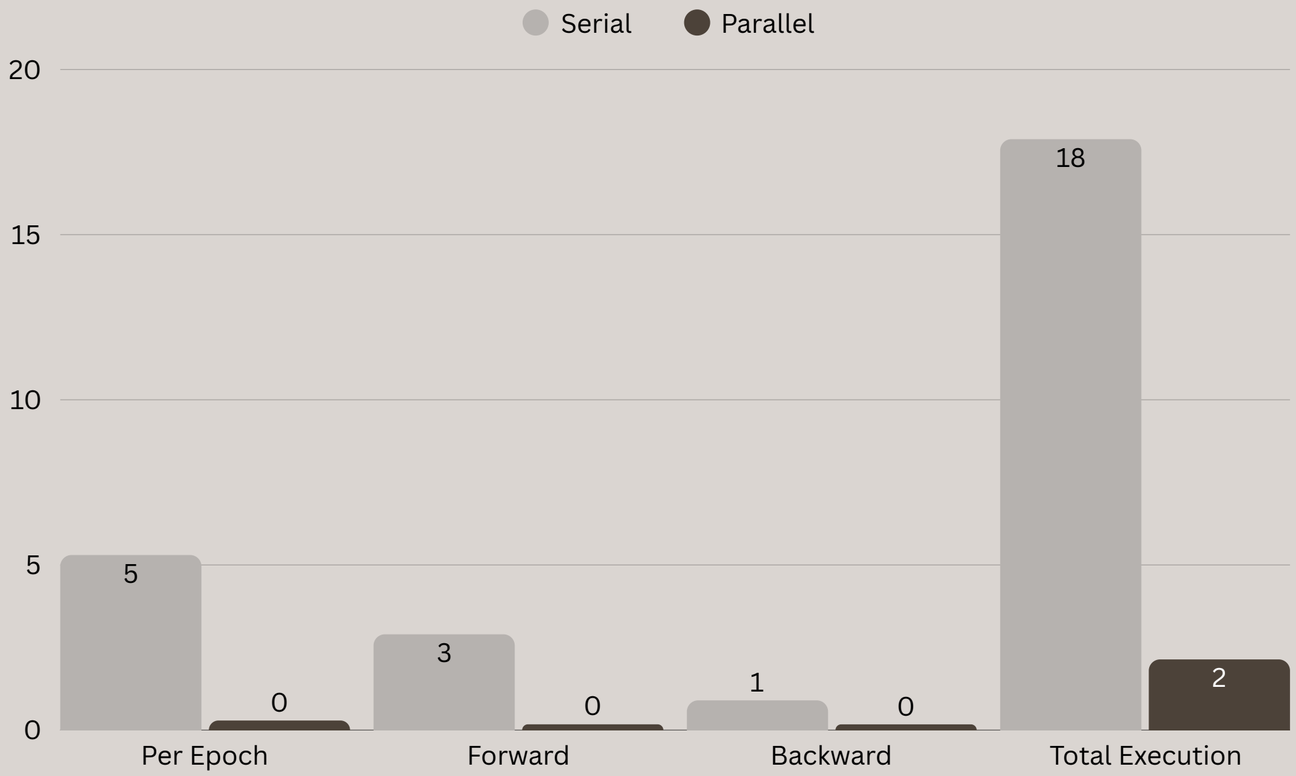
Version 3.1 – Mini-Batch Parallelism

Focuses on the algorithm used in the Neural Network training called Stochastic Gradient Descent (SGD)

This strategy provides poor parallelization room.

Batch of images processing at the same time

Speed up from Serial:
23.39/2.14 → ~10.9x



Version 4 – Tensor Core via cuBLAS

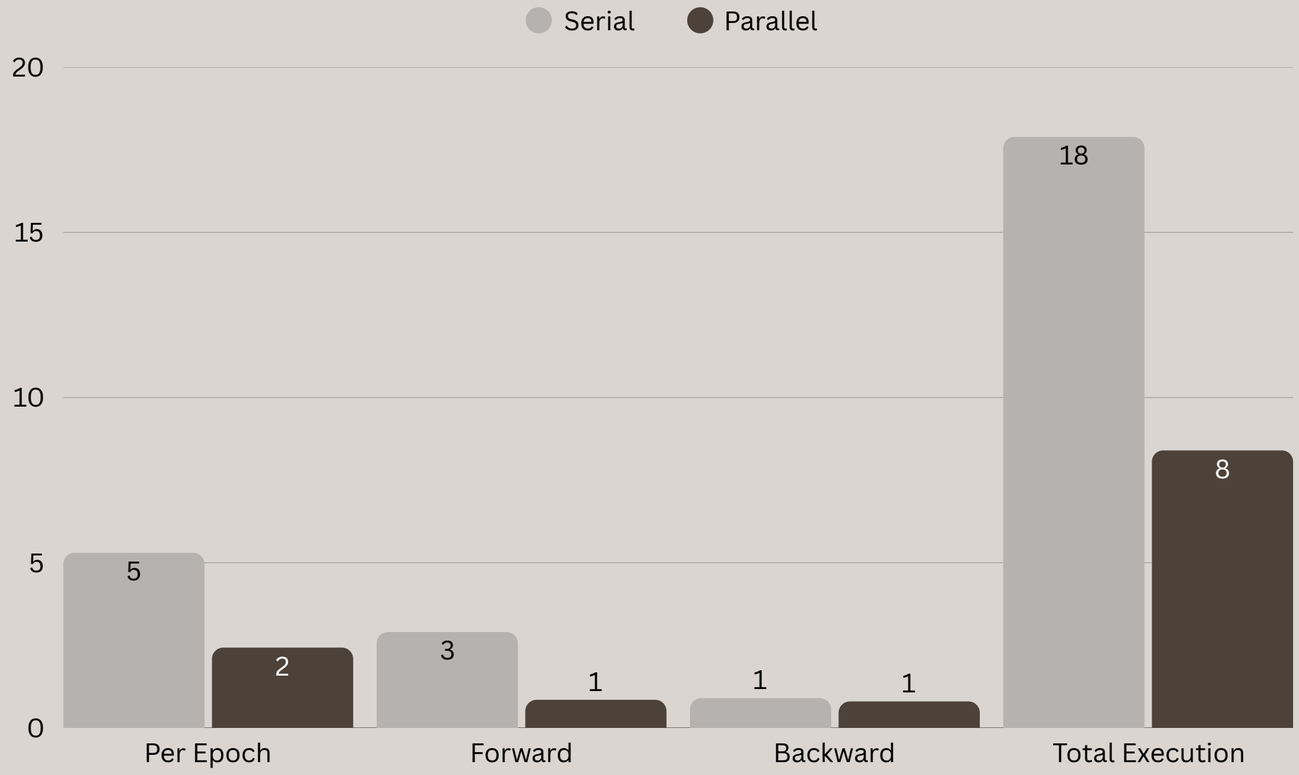
GEMM (General Matrix Multiplication)

Replace manual mat-muls with cublasSgemm on NVIDIA tensor cores

Achieves ~2.8× speedup vs serial; forward ~3.6×, backward ~1.0×

Provides better speedup

If paired with mini-batch would provide estimated speed up of 50x



Observations

Calling the kernels hundred of thousands of times put a bottleneck at the acceleration which could not be overcome without changing the algorithm itself

Summary of Speedups

Version	Per-Epoch (s)	Speedup vs Serial	Key Gain Factor
Serial	7.8	1×	–
V2	6.6	1.1×	Basic CUDA offload
V3	5.3	1.6× (fwd)	Shared mem, streams
V3.1	0.43	10.9×	Mini-batch parallelism
V4	2.8	2.8×	Tensor cores via cuBLAS
V5	13.3	0.57×	OpenACC pragmas

Thank You