HPC Project: Accelerating MNIST Classification

- Compared 4 versions: CPU, basic GPU, optimized GPU, Tensor Cores
- Goal: Maximize speed, preserve accuracy
- Significant performance gains with GPU acceleration

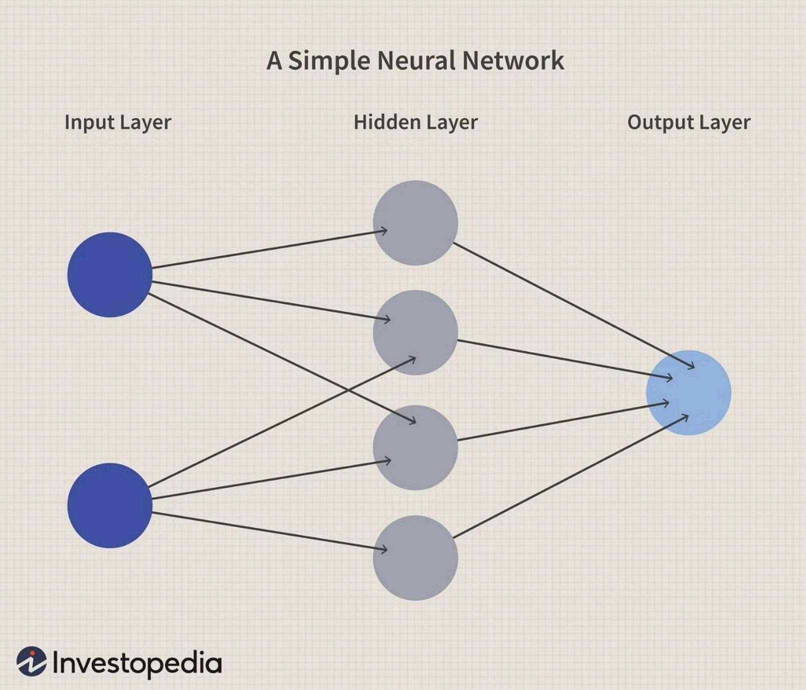
Project Overview & Team

Project Focus

- **Task**: MNIST digit classification (70,000 images)
- **Approach**: Optimize neural network across 4 versions
- **Goal**: Accelerate inference while maintaining accuracy

Team & Resources

- Ali Haider (22i-1210)
- Awais Khan (22i-0997)
- Muhammad Shayan Memon (22i-0773)



Neural Network Architecture

Input Layer

784 nodes (28x28 pixels, flattened)

Hidden Layer

128 nodes, ReLU activation

Output Layer

10 nodes, softmax (digits 0–9)

Training

Learning rate 0.01, 3 epochs, batch size 64 (V3/V4)

Implementation Versions

1. **V1: CPU Baseline**

   Sequential C implementation, 22.4s, 96.78% accuracy.

2. **V2: Naive GPU**

   CUDA port, static kernels, frequent memory transfers, 183.2s, 96.49% accuracy.

3. **V3: Optimized GPU**

   Dynamic configs, memory reuse, CUDA streams, 6.8s, 96.20% accuracy.

4. **V4: Tensor Core GPU**

   cuBLAS, TF32, fastest at 5.8s, 91.93% accuracy.

Key Optimizations by Version

Parallelism

Dynamic block/grid sizes, CUDA streams, shared memory softmax

Memory Handling

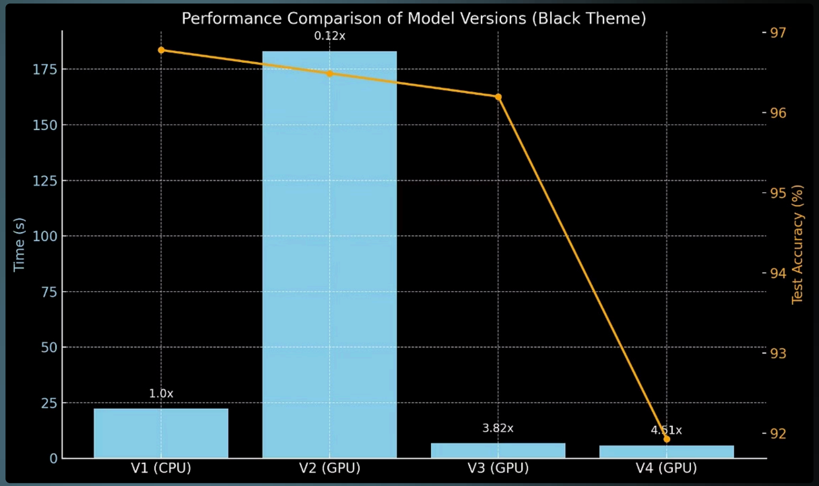
Pinned memory, memory reuse, reduced transfers (V3/V4)

Tensor Cores

cuBLAS with TF32, cublasGemmEx for fast matrix ops (V4)

Performance Results & Analysis

Version	Time (s)	Speedup	Test Acc.
V1 (CPU)	22.4	1.00x	96.78%
V2 (GPU)	183.2	0.12x	96.49%
V3 (GPU)	6.8	3.82x	96.20%
V4 (GPU)	5.8	4.51x	91.93%



Conclusion & Takeaways

1

GPU Acceleration Works

Proper parallelization (V3/V4) delivers major speedups for neural networks.

2

Optimization Matters

Naive GPU code (V2) can be slower than CPU if not tuned for memory and kernel efficiency.

3

Trade-offs

Tensor Cores (V4) offer speed but may reduce accuracy due to TF32 precision.