High-Performance MNIST Digit Classification

This presentation explores how to accelerate a neural network using CUDA for classifying handwritten digits from the MNIST dataset.

By:

- Rana Bilal Akbar
- Maaz Khan
- Mehboob Ali

Project Objective

Goal

Build a fast and accurate neural network for MNIST digit classification using the GPU.

Techniques

Parallel kernels, memory optimization, batch training, CUDA Programming.

MNIST Dataset

Training

60,000 images.

Testing

10,000 images.

Format

28x28 grayscale pixels, 784-dimensional input vector.

Neural Network Architecture

1

Input

784 neurons.

2

Hidden

128 neurons with ReLU activation.

3

Output

10 neurons with softmax activation.

```
void backward_batch(NeuralNetworkDevice* net, dou
    double* d_output_batch;
    double* d_hidden_batch;

    cudaMallocAsync(&d_output_batch, batch_size *
    cudaMallocAsync(&d_hidden_batch, batch_size *

    cudaMemsetAsync(d_output_batch, 0, batch_size
    cudaMemsetAsync(d_hidden_batch, 0, batch_size
    dim3 grid1(batch_size);
    dim3 block1(OUTPUT_SIZE);
    layerGradientBatch<<<grid1, block1, 0, stream

    dim3 grid2(batch_size);
    dim3 block2(HIDDEN_SIZE);
    hiddenLayerGradientBatch<<<grid2, block2, 0,

    dim3 grid3(OUTPUT_SIZE);
    dim3 block3(HIDDEN_SIZE);
    updateWeights2Batch<<<grid3, block3, 0, strea

    dim3 grid4(HIDDEN_SIZE);
    dim3 block4(INPUT_SIZE);
    updateWeights1Batch<<<grid4, block4, 0, strea

    cudaFreeAsync(d_output_batch, stream);
    cudaFreeAsync(d_hidden_batch, stream);
}
```

Implementation Details

Kernels

Custom CUDA kernels for all layers.

Memory

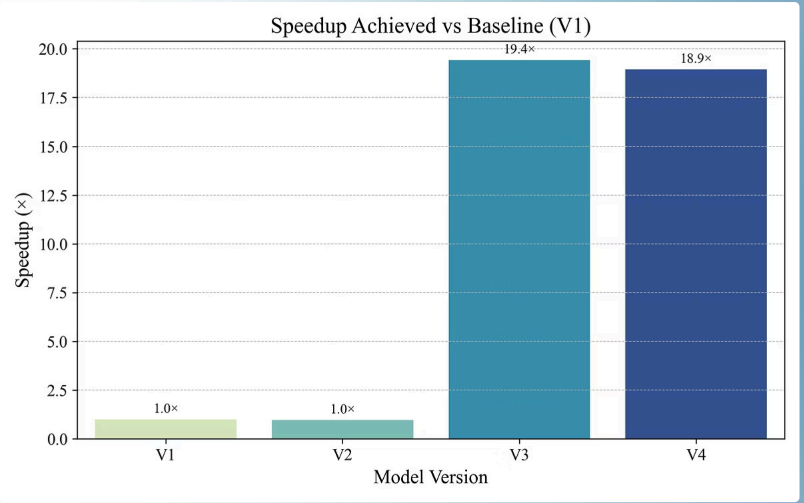
Pinned host memory, asynchronous transfers.

Optimization

Shared memory, batch processing, streams.

Performance Comparison

Version	Training Time	Accuracy
V1 (CPU)	Very Slow	~97%
V2 (GPU)	Faster	~97%
V3 (GPU)	Much Faster	~91.76%
V4 (GPU)	Fastest	~91%



```
./v2.exe
MNIST Neural Network2

Epoch 1 - Loss: 0.2314 - Train Accuracy: 93.17% - Time: 7.816s
Epoch 2 - Loss: 0.0992 - Train Accuracy: 97.05% - Time: 8.067s
Epoch 3 - Loss: 0.0689 - Train Accuracy: 97.92% - Time: 7.942s
Total training time: 23.824s
```

Version 2: GPU Forward Pass

CUDA Kernel

Each thread calculates one neuron's output.

Memory

GPU memory for weights and activations.

Version 3: Full GPU Backpropagation

Kernels

Backward passes for both layers.

Atomic Operations

Safe parallel weight updates.

Batch Training

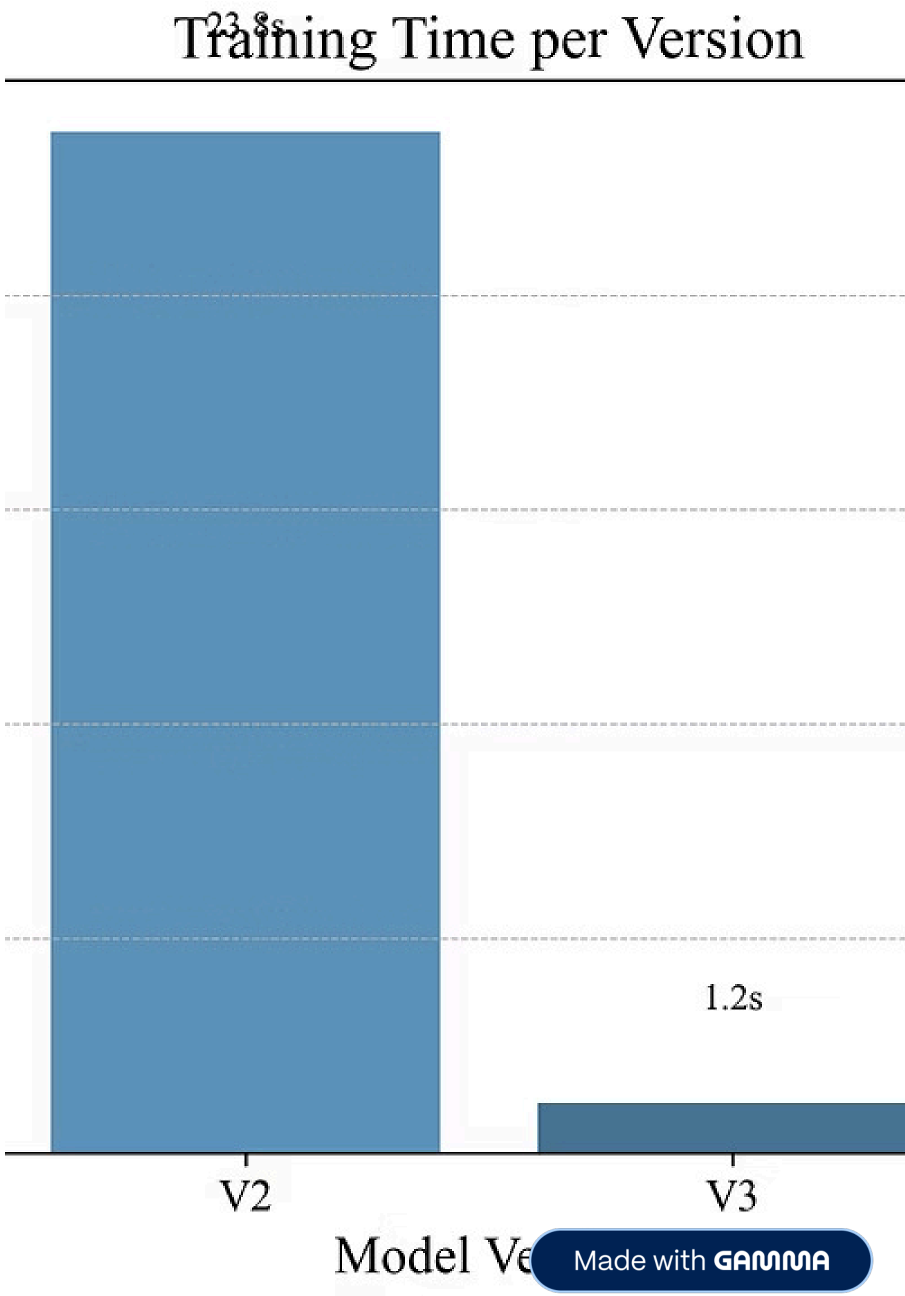
Improved throughput.

CUDA Streams

Overlapping memory transfer and computation.

CUDA Shared Memory

Easier access



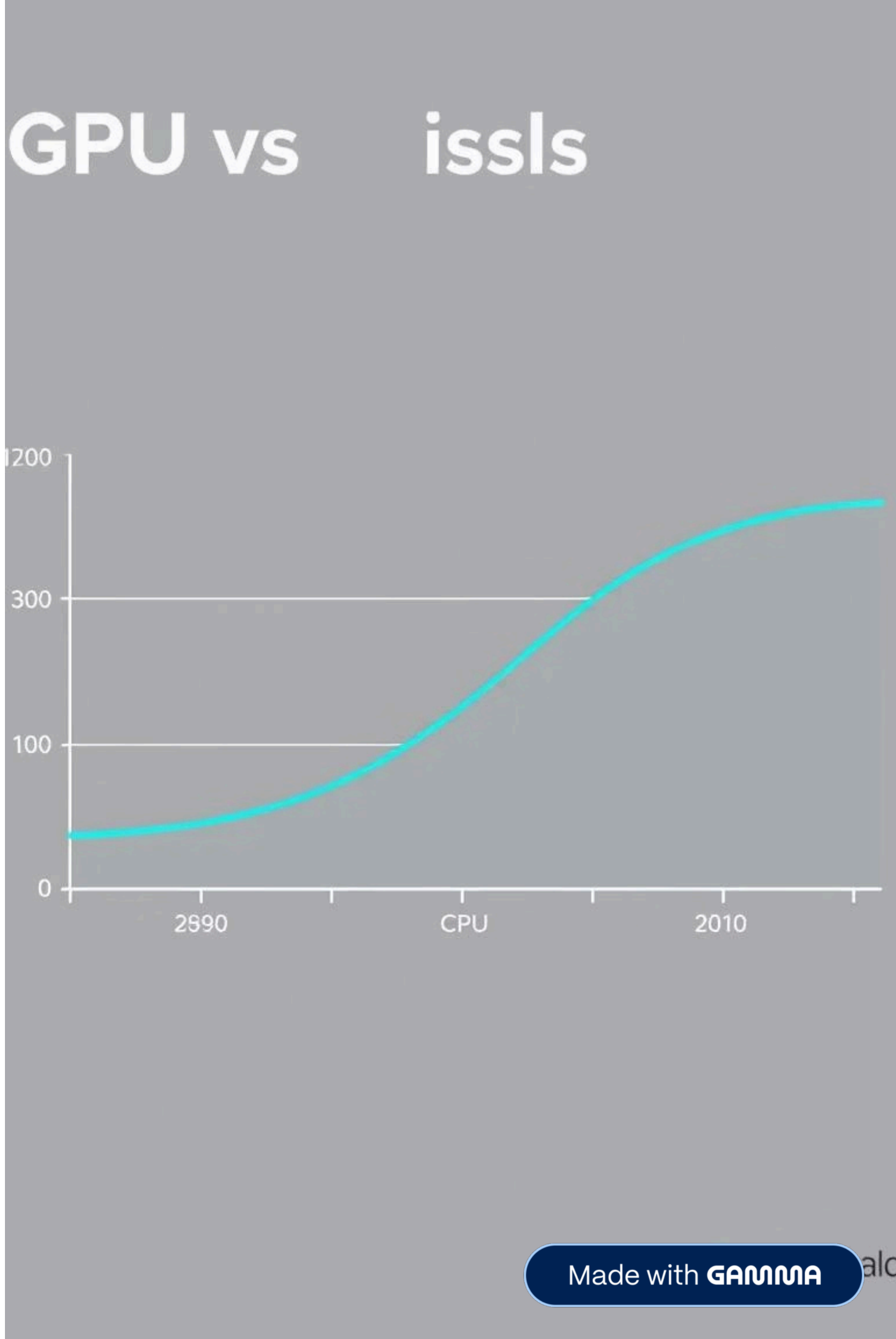
Conclusion

Results

Significant performance gains with GPU acceleration.

Impact

GPU programming is powerful for deep learning.



Made with GAMMA