Comprehensive Proposal: High-Performance CUDA-Based CNN for E-commerce Sales Forecasting

Mehdi Khameedeh, Salar Rahanama

December 31, 2025

1 Project Abstract

This project proposes the implementation of a Convolutional Neural Network (CNN) for sales forecasting on the Brazilian E-Commerce Public Dataset (Olist Store). While inspired by the predictive accuracy goals of Singh et al. (2020), this work focuses on the **GPU Systems** challenge of training a network from scratch using CUDA. We aim to demonstrate that performance-driven design—specifically via memory tiling, coalesced access, and occupancy tuning—can significantly optimize the training pipeline while maintaining functional correctness.

2 Dataset Selection and Preprocessing

The research utilizes the **Brazilian E-Commerce Public Dataset by Olist** (100,000 orders).

- **Feature Engineering:** We will extract temporal features (day of week, seasonality) and product metadata (category, price). These are mapped into a 1D or 2D tensor format suitable for convolutional kernels.
- **CPU-Side Preparation:** Normalization and batching will be handled on the host (CPU). Data will be pinned in memory (`cudaHostAlloc`) to maximize the transfer rate (PCIe bandwidth) to the device.

3 Detailed CNN Architecture

The network will be a regression-based CNN designed to predict a continuous value (Total Sales).

1. **Input Layer:** $N \times C \times L$ (Batch size, Channels, Length of time window).
2. **Convolutional Layer (1D):** 32 filters, 3×1 kernel size. *CUDA implementation will use Shared Memory to cache weights.*
3. **Activation Layer:** ReLU, implemented as a point-wise parallel kernel.
4. **Pooling Layer:** 1D Max-Pooling (2×1 stride) to reduce feature map variance.
5. **Fully Connected (FC) Layer:** Maps high-level features to a single scalar. Implemented via GEMM (General Matrix Multiply) logic.

4 CUDA Implementation & System Optimization

The "Core Training Pipeline" will be written entirely in CUDA C++.

4.1 Kernel Design

- **Forward Pass:** Convolution will be implemented using a **tiling strategy**. By loading input patches into `__shared__` memory, we reduce the number of redundant Global Memory reads by a factor of the kernel size.
- **Backward Pass (Backprop):** We will implement kernels to compute the gradient of the loss with respect to weights ($\frac{\partial L}{\partial W}$). This involves a transposed convolution-like operation on the GPU.
- **Weight Update:** An SGD kernel will perform $W = W - \eta \cdot \nabla W$ as a single-pass element-wise operation.

4.2 Performance-Driven Design Choices

- **Memory Coalescing:** We will ensure that data layouts (Nabla/Tensor formats) allow for 128-byte aligned accesses, ensuring a single memory transaction per warp.
- **Occupancy Maximization:** We will analyze register usage per thread. If register pressure is high, we will tune the `maxrregcount` to allow more active warps on the SM.
- **Arithmetic Intensity:** We will aim to increase the ratio of floating-point operations to memory access by fusing the ReLU activation kernel with the Convolution kernel.

5 Anticipated Bottlenecks

1. **Memory Bandwidth:** The Olist dataset involves many small transactions. Frequent global memory access during backpropagation is expected to be the primary bottleneck.
2. **Atomic Contention:** When calculating gradients across a large batch, `atomicAdd` operations in global memory may lead to serialization. We will explore **warp-level primitives** (`__shfl_down_sync`) to perform partial reductions within registers first.

6 Evaluation and Reporting

6.1 Functional Correctness

We will plot a **Loss vs. Epoch** curve. Success is defined by the Mean Squared Error (MSE) decreasing and converging, matching or exceeding the predictive trends noted in the Singh et al. paper [1].

6.2 Performance Metrics

Using **NVIDIA Nsight Systems** and **Compute**, we will report:

- **Kernel Timings:** Latency of individual forward vs. backward kernels.
- **Effective Bandwidth:** (Bytes read + Bytes written) / Time.

- **Roofline Analysis:** Plotting our kernels against the GPU's theoretical peak to see if we are compute-bound or memory-bound.

References

[1] Singh, K., Booma, P. M., & Eaganathan, U. (2020). E-Commerce System for Sale Prediction Using Machine Learning Technique. *Journal of Physics: Conference Series*.

[2] NVIDIA. (2023). *CUDA C++ Programming Guide*.