Further exploration: A Data-Driven Approach to Understanding Road Safety in NYC

[1] Shubo Pang
Columbia University

sp4410@columbia.edu

I. Introduction

Modern urban traffic systems face significant challenges in managing large-scale data efficiently and ensuring real-time responses to dynamic traffic conditions. This project aims to leverage the computational power of GPU clusters and deep learning frameworks to address these challenges effectively.

The primary goal of this study is to optimize the processing of massive traffic datasets, predict accident-prone areas, and enhance real-time traffic management capabilities. By implementing advanced methods such as CUDA-enabled parallel processing and PyTorch frameworks, this research explores the application of GPU acceleration to reduce latency and improve the efficiency of traffic data analysis and model training.

The research includes the following key components:

1.Development of a severity mapping system using GPU-based parallel computing to analyze and visualize traffic accident data.

2.Application of deep learning models such as ResNet to classify and predict traffic sign categories with high accuracy.

3.Exploration of high-performance computing frameworks to improve the scalability and reliability of real-time hazard prediction systems.

Through these efforts, this study aims to contribute to the development of intelligent transportation systems, enhancing safety and efficiency in urban traffic management.

II. Statement

A. Milestone Task Review

Based on the results generated in Milestone 4, including the heatmap and severity mapping, we will conduct further discussion.

B. Optimization and Acceleration with GPU

As for the severity mapping part of the code, we can utilize PyTorch tensors and place them on GPU devices to perform these computations in parallel. The source code and scripts related to the analysis can be found on GitHub repository[1].

In the Milestone 3 visualization of the "Average Number of Crashes per Hour of Day," we observed that traffic accidents peak between 14:00 and 18:00, with a decreasing trend on both sides. Therefore, we centered the traffic accidents from 14:00 to 18:00 and generated a bar chart of traffic accidents over time, fitting the data to a normal distribution. Taking 16:00 as the axis of symmetry, the highest point of each bar in the histogram is fitted using the least squares method, and the resulting normal distribution is shown in Figure 5.

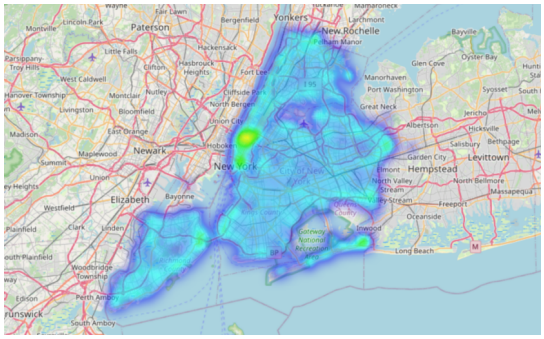

Fig. 1. heatmap result

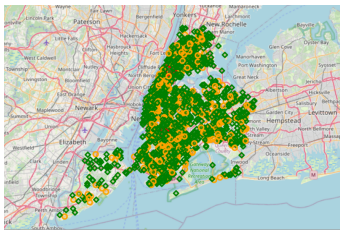

Fig. 2. severity mapping result

C. Advantages of Using GPU Clusters

The advantages of using GPUs are as follows:

1.Remarkable Parallel Computing Capability In this code, the first thing we notice is the GPU's powerful parallel computing capability when handling large-scale data. For instance, when determining the severity of each accident during the classification process, a CPU would need to iterate through the DataFrame row by row to determine the severity of each incident. In contrast, a GPU can load the entire dataset into memory at once and utilize its numerous computing cores to process multiple data points simultaneously. This parallel operation significantly reduces processing time. Particularly when dealing with large datasets, the advantages of GPUs become even more evident. From the results, the GPU's processing time is reduced to mere seconds, whereas the CPU might require several minutes, showcasing the GPU's outstanding performance in parallel computing.

```
# Use GPU to compute marker categories
print("Using GPU acceleration to generate marker types...")
markers = torch.empty(len(data_geo_sampled), dtype=torch.int32, device='cuda')
markers[killed > 0] = 2  # Red triangle
markers[(killed == 0) & (injured > 0)] = 1  # Orange circle
markers[(killed == 0) & (injured == 0)] = 0  # Green square
```

Fig. 3. GPU Optimization and Acceleration for Severity Mapping Visualization

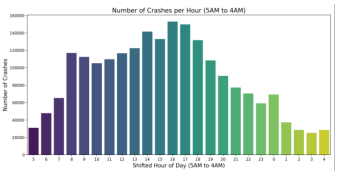

Fig. 4. The traffic accident chart with 14H-18H centered

2.Efficient Resource Utilization GPUs not only operate faster but also utilize system resources more efficiently. In the code example, once we convert the data into PyTorch tensors and move them to the GPU, all subsequent computations are performed directly on the GPU, minimizing the number of data transfers between the CPU and GPU. Additionally, GPUs can provide consistent high throughput without interfering with other applications, which is particularly important for scenarios with high real-time requirements.

3.Accelerating Machine Learning Model Training and Inference Beyond simple condition evaluations, GPUs can significantly accelerate the training and inference processes of machine learning models. Although the current code does not involve complex model training, in practical applications, GPUs can greatly reduce the training time for deep learning or traditional machine learning algorithms (such as decision trees and support vector machines). For this case, if future plans include extending the functionality to predict traffic accident risks based on historical data, GPUs will undoubtedly be a critical tool for accelerating this process.

The source code and scripts related to the analysis can be found on GitHub repository[1], this code demonstrates the advantages of GPU parameters in data processing tasks, particularly in accelerating complex data analysis and machine learning workflows. By comparing the processing time, performance metrics, and visualization results of the CPU and GPU, we can clearly observe the efficiency improvements and technical advantages brought by the GPU.

D. Applications of Introducing GPU and HPC (High-Performance Computing) Frameworks

GPU and HPC (High-Performance Computing) frameworks can quickly generate notifications for pedestrians and vehicles at intersections based on the corresponding normal distribution of accident occurrences at specific times. The specific applications are as follows:

1. Accident-Prone Area Alerts

Real-Time Data Processing: Utilize GPU acceleration to rapidly process real-time data from multiple sensors (such as cameras, radars, etc.) to identify potential hazards or accident-prone areas.

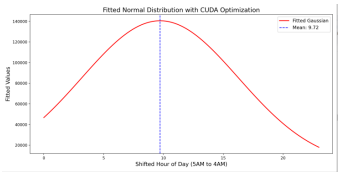

Fig. 5. Normal distribution fitted using the least squares method

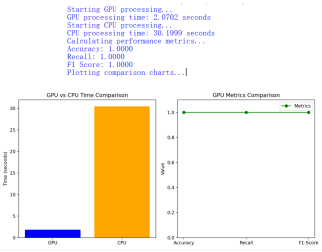

Fig. 6. Comparison of Processing Efficiency Between CPU and GPU

Large-Scale Data Analysis: GPUs can accelerate the analysis of historical accident data and pattern recognition, enabling the training of complex statistical models to more accurately identify high-risk locations.

Implementation Approach: Leverage deep learning frameworks (e.g., TensorFlow, PyTorch) in combination with GPU acceleration to build convolutional neural networks (CNNs) or other types of neural network models. These models can be used for tasks such as image classification and object detection, helping to automatically identify accident-prone areas.

2. Notification Services

Framework Support: TensorFlow and PyTorch both offer strong GPU support, which can significantly accelerate the training of neural networks and the inference speed on large-scale datasets.

Real-Time Response: When vehicles approach known accident-prone areas or areas predicted to have potential risks, GPUs can accelerate the execution of algorithms like decision trees or support vector machines (SVMs) to quickly generate warning signals.

Personalized Recommendations: Based on the driver's historical behavior and personal preferences, notifications can be customized. This requires extensive personalized data processing, which GPUs can accelerate.

Implementation Approach: Develop a real-time monitoring system capable of receiving data from in-vehicle sensors and evaluating the risk level of the current driving environment using pre-trained models. Once abnormal situations are detected, the system can immediately send warning notifications to the driver.

Real-Time Data Processing: Utilize GPU acceleration to rapidly process real-time data from multiple sensors (such as cameras, radar, etc.) to identify potential hazards or accident-prone areas. Large-Scale Data Analysis: Analyze historical accident data and perform pattern recognition using GPUs to

accelerate the training of complex statistical models, enabling more accurate identification of high-risk locations.

Implementation Approach: Use deep learning frameworks (such as TensorFlow or PyTorch) combined with GPU acceleration to build Convolutional Neural Networks (CNNs) or other neural network models for tasks like image classification and object detection, assisting in the automatic identification of accident-prone areas.

2. Alert Services

GPU-Enhanced Frameworks: Both TensorFlow and PyTorch provide robust GPU support, significantly accelerating the training of neural networks and inference speed on large datasets.

Instant Response: When a vehicle approaches a known accident-prone area or a potential accident is predicted, GPUs can accelerate the execution of algorithms like decision trees or Support Vector Machines (SVMs) to quickly generate warning signals.

Personalized Recommendations: Customize alert content based on a driver's historical behavior and personal preferences. This requires extensive personalized data processing, which GPUs can efficiently accelerate.

Implementation Approach: Develop a real-time monitoring system that receives data from in-vehicle sensors and evaluates the risk level of the current driving environment using pre-trained models. When abnormal situations are detected, the system immediately sends warning notifications to the driver.

E. Code Examples of GPU Clusters for Early Hazard Prediction

We use CUDA and GPU technology to accelerate model training and inference, applying the RESNET model in traffic sign recognition scenarios and improving computational efficiency through torch.compile. The source code and scripts related to the analysis can be found on GitHub repository[1].

Testing speed limit signs, we obtained the results shown in

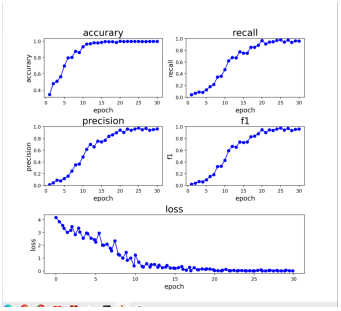

Fig. 7. Output Results of the SGD Optimizer

Figure 8. Testing pedestrian warning signs, we obtained the results shown in Figure 9.

From this, we conclude:

1. Applicability and Advantages of the SGD Optimizer

From the charts, it can be observed that the SGD optimizer demonstrates the following characteristics in traffic sign classification tasks:

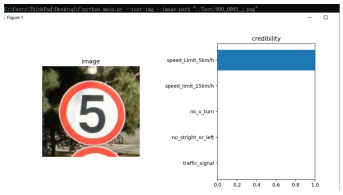

Fig. 8. Predicting Speed Limit Signs

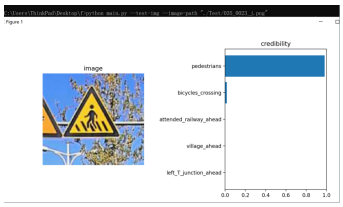

Fig. 9. Predicting pedestrian warning signs

Convergence and Accuracy Performance:

Around epoch 20, various metrics (accuracy, recall, precision, and F1) gradually stabilize and approach 1.0. The loss value also decreases rapidly and enters a steady state. This indicates that, with sufficient training, the model can achieve very high prediction accuracy, making it highly suitable for traffic sign classification tasks that require high precision. Strong Generalization Capability:

The minimal curve fluctuations indicate that the optimization path of the SGD optimizer has good stability and generalization capability. This is especially important in scenarios where the distribution of training and test datasets may differ, such as dynamic traffic environments.

Memory Efficiency:

The SGD optimizer uses less GPU memory, enabling GPU resources to be utilized more efficiently for large-batch training tasks (e.g., increasing batch size). These characteristics demonstrate that SGD is an optimization method suitable for tasks requiring high precision, particularly when using the ResNet model for classification in traffic management.

2. The Role of GPU Acceleration in Real-Time Traffic Sign Classification

Improvements Through CUDA and torch.compile:

CUDA acceleration takes full advantage of the parallel computing power of GPUs, significantly improving the speed of model training and inference.

Using torch.compile to compile the model optimizes the computation graph and reduces Python interpreter overhead, making the training process more efficient.

For real-time traffic sign classification tasks, this optimization meets the requirements of traffic management systems for low latency and high throughput.

The Ability to Predict Hazards in Advance:

The second and third images demonstrate the model's accurate predictions for specific traffic signs (e.g., pedestrian and slowdown signs).

This high-accuracy classification capability can be used to anticipate potential hazard scenarios, such as pedestrian cross-

ings near crosswalks or complex road conditions requiring vehicles to slow down.

3. The Synergy Between SGD Optimizer and GPU Clusters

The advantages of the SGD optimizer in convergence performance and generalization capability make it highly suitable for large-scale training on GPU clusters. The following techniques can further enhance efficiency:

Mixed Precision Training:

Utilize FP16 and FP32 for mixed computation, improving training speed while saving memory resources.

Increased Batch Size:

Increasing the batch size enables more efficient use of GPU resources, reducing the number of iterations per epoch.

Multi-GPU Distributed Training:

Implementing distributed training on GPU clusters can further shorten training time.

4. Application Scenarios for Real-Time Hazard Prediction

From the classification results in the images, it can be seen that the model achieves very high prediction accuracy and confidence (credibility) for traffic signs. For example:

In the first image, the model's confidence in classifying the pedestrian sign is close to 1.0, indicating that it can quickly and accurately identify pedestrian scenarios. This is critical for alerting drivers in advance and enabling them to take appropriate measures.

In the second image, the model's confidence in classifying the "slow down" sign is also very high, demonstrating the model's ability to recognize complex road conditions and assist traffic management systems in adjusting traffic flow in real-time.

5. Comprehensive Discussion: The Contribution of GPU Cluster Acceleration to Traffic Management

Using CUDA and GPU cluster acceleration can greatly reduce training time and enable real-time inference, allowing traffic management systems to quickly perceive and respond to road hazards. In traffic sign classification tasks, the combination of ResNet and SGD optimizer not only improves the model's final accuracy but also ensures its generalization capability, performing better in real-world traffic scenarios.

The high accuracy and confidence of the model's classification results further validate the importance of GPU clusters in building real-time traffic monitoring and hazard prediction systems.

Further Advantages Discovered:

1. Advantages of Using GPUs for Processing – Accelerated Training Speed

Accelerated Gradient Descent Optimization:

From the output charts, it is evident that training loss decreases significantly in each epoch. This demonstrates the role of GPUs in large-scale parallel computation, which accelerates the gradient descent optimization process.

Enhanced Real-Time Capability:

The powerful parallel computing capabilities of GPUs enable models to iterate quickly, allowing them to adapt to new data distributions faster. This is especially critical for applications requiring real-time responses.

Rapid Improvement in Training Accuracy:

Performance Metric Enhancement: With increasing epochs, metrics such as accuracy, recall, and F1-score gradually converge and reach high levels (close to 1). This not only confirms the positive impact of GPUs on model training performance but also highlights their efficiency in multi-batch parallel computation. Model Stability: The rapid improvement in these metrics indicates that models trained on GPUs can quickly learn the underlying patterns in the data, ensuring reliable predictive capabilities in real-world applications.

2. Practical Applications of GPU Clusters – Improved Training Efficiency with GPUs

Real-Time Monitoring and Processing:

As illustrated in the uploaded document, GPU acceleration significantly improves the training and inference speed of deep learning models, providing a solid foundation for real-time vehicle data monitoring and processing.

The Necessity of Optimization:

The training curves (accuracy and loss) clearly demonstrate the importance of GPU acceleration for model optimization, especially when dealing with complex tasks.

Demand for Large-Scale Data Processing:

Massive Video and Image Data:

Vehicle cameras generate a large volume of video and image data every second, which traditional CPUs struggle to process efficiently. The parallel computing power of GPU clusters enables real-time analysis of these massive datasets, handling tasks such as object detection, traffic flow monitoring, and driving behavior analysis.

Edge Computing Advantages:

Integrating GPUs directly into vehicles allows for preliminary data processing locally, reducing latency and alleviating the burden on cloud servers.

3. Transition from Vehicle Cameras to GPU Clusters – GPU Integration in Modern Vehicles

Real-Time Recognition and Detection:

In modern vehicles, cameras integrated with GPUs can perform inference tasks through centralized GPU clusters, such as real-time traffic light recognition and pedestrian detection, ensuring timely safety alerts for drivers.

Resource Sharing and Collaborative Operations:

With GPU clusters, computing resources can be shared among different vehicles, enhancing the intelligence of the entire transportation system. For example, multiple vehicles can form a distributed GPU network through V2V communication to collaboratively address complex traffic scenarios.

In summary,the parallel computing power of GPU clusters is indispensable for large-scale data processing, particularly in traffic scenarios. It not only improves data processing speed but also enhances model stability and accuracy.

Centralized Architecture: Vehicle cameras can achieve a centralized architecture for data collection and real-time processing through GPU clusters, contributing to the efficiency of intelligent transportation systems.

Chart Support: The metrics in the charts (accuracy, recall, loss curves) illustrate the speed of model training and inference

using GPUs, as well as their potential effectiveness in real-time scenarios.

In conclusion, leveraging the powerful computational capabilities and efficient parallel processing mechanisms of GPU clusters not only significantly enhances the speed and quality of model training but also ensures the effective execution of various real-time tasks in intelligent transportation systems. This technology's applications are not limited to data processing within a single vehicle but can be extended to the entire vehicle network environment, promoting collaboration and information sharing among vehicles. This further advances the construction and development of smart cities.

F. Outlook

Intelligent transportation systems will increasingly rely on efficient data processing capabilities to analyze massive amounts of information in real time. With the advancement of autonomous driving technology and vehicle-to-everything (V2X) communication, the demand for rapidly and accurately handling complex traffic scenarios is becoming more urgent. GPUs (Graphics Processing Units), with their exceptional parallel computing capabilities and high-speed floating-point operations, have become a critical component of modern intelligent transportation systems.

Building large-scale GPU clusters can centralize the processing of data from multiple sources, such as information collected by in-vehicle sensors, cameras, radars, and other devices, enabling faster and more accurate decision-making support. By applying optimization techniques—such as using `torch.compile` to reduce the overhead of operation interpretation, increasing batch size for more efficient GPU utilization, enabling `pin_memory` and `num_workers` to accelerate data transfer, and adopting mixed-precision training to enhance training speed and reduce memory usage—these measures not only improve the efficiency of model training and inference but also ensure effective in-vehicle data preprocessing and emergency responses, reducing latency and enhancing the robustness and reliability of the system.

Furthermore, with V2V (vehicle-to-vehicle) communication, multiple vehicles can form a distributed GPU network to share computing resources and collaboratively tackle complex traffic scenarios. This integration not only enhances the intelligence of individual vehicles but also promotes the coordinated operation of the entire road traffic system, laying a solid foundation for the development of smart cities and autonomous driving technologies.

ACKNOWLEDGMENT

We would like to express our gratitude to the Department of Civil Engineering and Engineering Mechanics at Columbia University for providing experimental equipment and technical support. We also extend our thanks to the Northeast Big Data Innovation Hub for their valuable contributions and feedback. Lastly, special thanks to Professor Sharon Di for their her systematic course structure and rigorous teaching methodology.

REFERENCES

[1] GitHub Repository: https://github.com/sp4410/f/tree/main