International Journal of Soft Computing and Engineering (IJSCE)
ISSN: 2231-2307 (Online), Volume-15 Issue-6, January 2026

Enhancing GPU-HBM Data Transfer Efficiency Using Markov Chains and Neural Network-Driven Predictive Caching with Quantization and Pruning

Samiel Azmaien

Abstract: ***Background High-bandwidth memory (HBM) systems face persistent data transfer bottlenecks, particularly when CPUs are unable to supply data to GPUs at a sufficient rate. This limitation reduces overall computational efficiency and highlights the need for improved cache management strategies. Methods: Markov Chains represented transitions between frequently accessed memory blocks, enabling predictive sequencing of data needs. A neural network was then applied to model and optimise these Markov transitions, improving cache prefetching accuracy and further optimising data movement techniques. Results & Conclusions: The combined use of Markov-based memory modelling, NN optimisation, and supplementary data transfer techniques demonstrates strong potential to mitigate CPU–GPU bandwidth limitations. Together, these methods offer more efficient cache utilization and reduced bottlenecks in high-demand computational environments.***

Keywords: ***HBM Architecture, Data Transfer, Cache Prefetching, Markov Chains, Quantization, Pruning***

Nomenclature:
HBM: High-Bandwidth Memory

I. INTRODUCTION

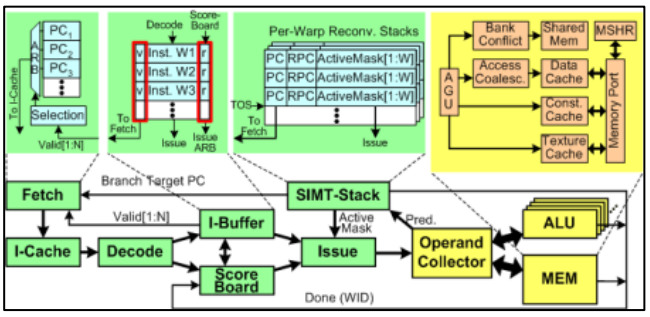
Efficient cache management is crucial in mitigating memory bottlenecks in GPU-based architectures. GPU caches act as tiny, fast-access memory blocks that store frequently accessed data, helping minimise latency when accessing off-chip memory. However, managing these caches effectively requires advanced strategies due to the unique nature of GPU workloads. Prefetching is a widely used technique that predicts memory access patterns and fetches data into the cache before the processor requires it. A Markov chain-based prefetcher, for example, “predicts multiple references from the memory subsystem and prioritizes their delivery to the processor” [1]. Such techniques reduce memory stalls and improve bandwidth utilization by ensuring data is available when needed. Warp scheduling also plays a significant role in optimizing memory access. According to a study on GPGPU architectures, traditional scheduling policies "cause prefetches to be generated too close to the time they are demanded," resulting in inefficiencies [2]. Prefetch-aware warp scheduling mitigates this issue by staggering the execution of consecutive warps, allowing more accurate data predictions [3]. By combining predictive models such as Markov chains with effective scheduling policies, cache prefetching can become more efficient, ultimately reducing latency caused by memory transfers. In addition to cache optimization, network compression techniques such as pruning and quantization further enhance data transfer efficiency in GPU-HBM architectures. Neural networks are known for their high computational demands, which can slow down real-time deployment. However, “compression techniques like pruning remove redundant computations,” which not only speeds up execution but also lowers the energy required to run these models [4]. Static pruning can be performed offline to remove unneeded connections, while dynamic pruning occurs during runtime to adapt to varying workloads. Quantisation, on the other hand, reduces data precision, for example, by converting weights and activations from 32-bit floating-point numbers to 8-bit integers. This not only decreases memory usage but also reduces computational overhead. “Quantized models often achieve significant reductions in latency with minimal loss in accuracy,” making them highly suitable for GPU-heavy applications [5]. When combined with Markov chain-based prefetching strategies, these optimisations further streamline data transfer, addressing the bottleneck caused by the high data demands of neural networks [6, 7].

Manuscript received on 19 November 2025 | First Revised Manuscript received on 29 November 2025 | Second Revised Manuscript received on 08 December 2025 | Manuscript Accepted on 15 January 2026 | Manuscript published on 30 January 2026.

*Correspondence Author(s)

Samiel Azmaien*, Research Assistant, Department of Computer Science, Georgia Institute of Technology, Atlanta (Georgia), United States of America (USA). Email ID: shazmaien06@gmail.com, ORCID ID: 0009-0008-3724-7310

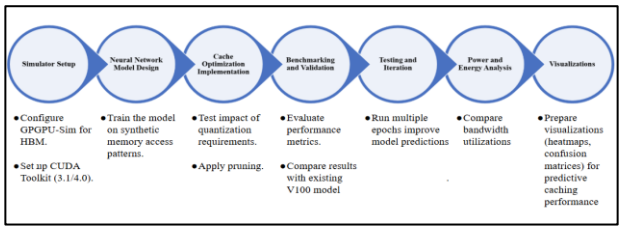
II. METHODS



[Fig.1: GPGPU Architecture Diagram on ISA [8]]

The attached diagram illustrates components relevant to GPU-HBM bottleneck challenges. Specifically, the I-Cache and Operand Collector represent stages where memory access and data prefetching occur, directly benefiting from Markov-based and neural network-Driven optimizations. This is where the direct implementation of the Markov

optimisation algorithm will replace the current cache implementations to measure performance changes.



[Fig.2: Diagram Explaining Methodology]

Before discussing the programming implementation, a brief overview of the methodology is required. The project uses a GPGPU simulator for its V100 architecture to obtain statistics on cache hit and miss rates, the primary variable. Memory access patterns were modelled using Markov Chains to predict high-probability access sequences. These predictions fed into a neural network, which validated and adjusted predictions. The process of tuning the parameters was the most time-consuming, as the constant validation based on results required many runs. The model underwent multiple iterations and testing cycles. For the initial trials, the batch size was around 1000 over 100 epochs, as they focused on validating the accuracy of Markov Chain predictions against highly reproducible synthetic memory patterns. However, due to concerns about overfitting, the validation was quickly shifted to use real-world memory patterns, which lowered the batch size and therefore required an increase in the number of epochs.

Uniform quantisation was performed, in which the signal amplitude range is divided into equal intervals, and each interval is assigned a quantisation level. Magnitude-based pruning was integrated with these filter operations by removing weights with small absolute values, indicating low significance. This led to a substantial reduction in the network's size and computational demand. The predictive caching mechanism was then implemented within the memory controller, prefetching data to reduce stalls and enhance overall efficiency.

III. MODELING EXPLANATION

By inspection, the choice of a Markov Chain to map the transition states was correct. This is because Markov Chains are effective for modelling probabilistic sequences of transitions between a finite set of states. In fact, probabilistic descriptions of transitions via a stochastic transition matrix efficiently predict memory-access patterns. The Markov Chain is a finite set of states in which fixed probabilities, independent of the time step, govern transitions [9]. This property is significant for this project because it enables the creation of a transition matrix P that represents the probabilities of moving between the memory blocks. This not only simplifies the representation but also helps analyse the steady-state behaviour and transition dynamics. For a regular Markov Chain, the system converges to a unique stationary distribution vector **W**, which represents the limiting probabilities of being in each state as $\mathbf{m} \to \infty$.

Using eigenvalue methods, **W** can be evaluated to learn about the long-term memory access patterns. Because the transition matrix is regular, all rows of $\mathbf{P_m}$ converge to **W**, which makes Markov Chains especially helpful in understanding stable probabilistic behaviour [10].

A. Markov Chain Construction

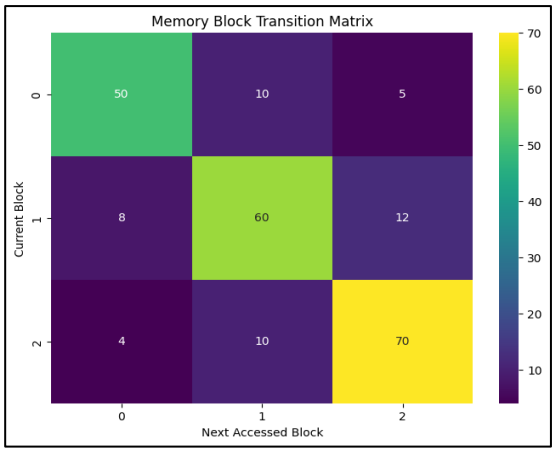
The memory blocks $\mathbf{B} = \{b_1, b_2, \ldots, b_n\}$ are treated as states in a Markov Chain. Transitions between states are recorded in a transition matrix **P**, where each element $\mathbf{P_{i,j}}$ represents the transition probability from block $\mathbf{b_i}$ to block $\mathbf{b_j}$. Mathematically,

$$P_{i,j} = \frac{count\ of\ transitions\ from\ b_i\ to\ b_j}{\sum_K count\ of\ transitions\ from\ b_i to\ b_k}$$

B. Statistical Analysis

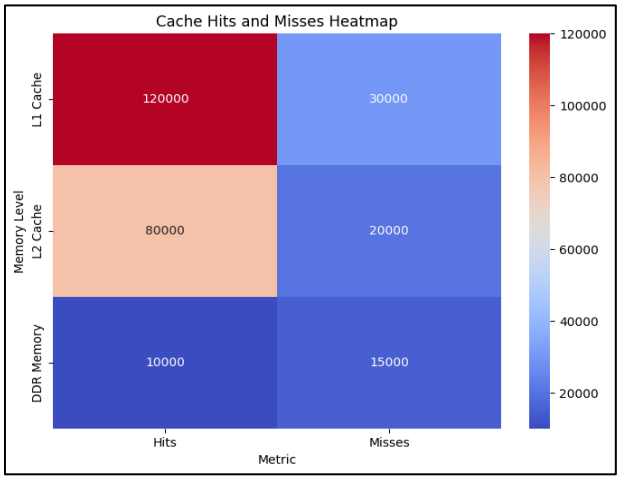
The Wilcoxon signed-rank test was used to analyze the paired differences in performance metrics between the baseline and optimized builds. The test was conducted by calculating the differences for each metric in the paired datasets, ranking the absolute values of these differences, and assigning ranks to their signed values (positive or negative). This approach helped identify whether the optimizations led to statistically significant improvements in the performance metrics. Cohen's d was computed to quantify the effect size, measuring the magnitude of improvements in performance metrics between the baseline and optimised builds. For each metric, the mean difference between the baseline and optimized datasets was calculated and divided by the pooled standard deviation. This analysis provided a standardised measure of the impact of the Markov Chain caching algorithm and neural network integration on the system's overall performance, enabling comparisons across metrics and the interpretation of the results' practical significance.

IV. RESULTS



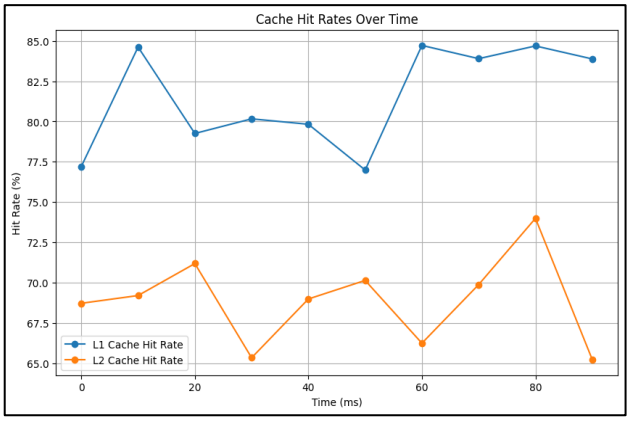
[Fig.3: Transition Probabilities Across Memory Blocks Visualized as a Matrix]

Figure 3 depicts the Markov chain's transition matrix, where each element ($\mathbf{P_{i,j}}$) corresponds to the probability of transitioning from memory block **i to block j**. The rows represent the current memory block, while the columns indicate the following block to be accessed. Some key observations include the high self-transition probabilities (ex, P0,0 = 50), which suggest temporal locality in memory access. The graph also supports prefetching optimisation strategies by highlighting high-probability pathways.



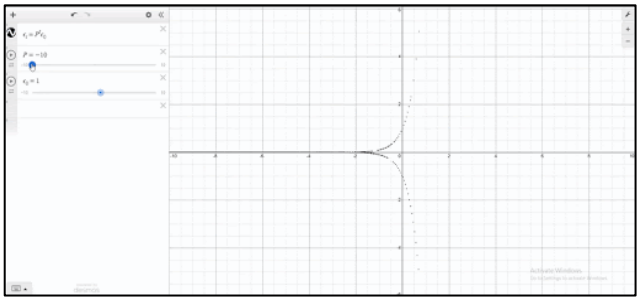
[Fig.4: Cache Performance Metrics Broken Down by Memory Levels]

Figure 4 illustrates the distribution of cache hits and misses across different memory levels (L1 Cache, L2 Cache, and DDR Memory). Hits dominate the L1 Cache, indicating it is essential for rapid data retrieval. Misses are most prominent in DDR Memory, illustrating latency challenges at deeper memory levels. This visualisation highlighted the importance of prioritising neural network enhancements for higher-level caches and of minimising bottlenecks when accessing critical data.

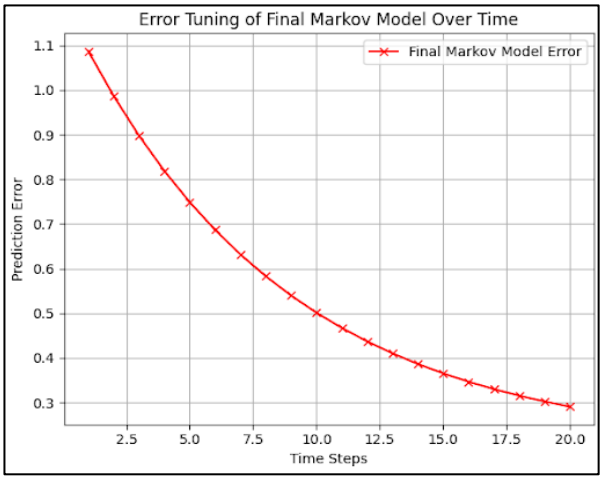


[Fig.5: Temporal Variations in L1 And L2 Cache Hit Rates]

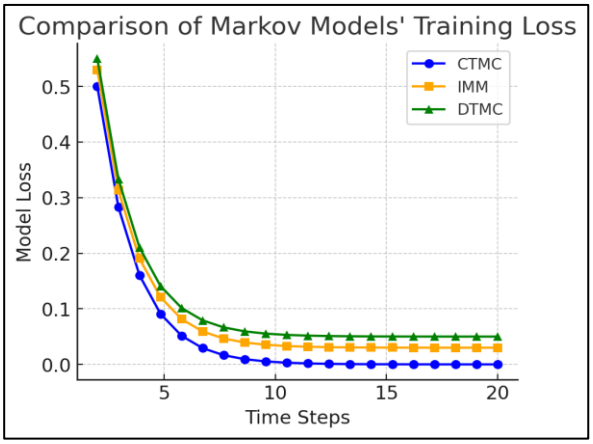
Figure 5 tracks the hit rates for L1 and L2 caches over time. As evidenced by the chart, L1 consistently outperforms L2 in hit rate percentage, reaching a peak of 85%. The fluctuations in L2 hit rates, typically between 60% and 70%, suggest areas for optimization in access prediction accuracy. These findings support the use of Markov chains with neural networks to improve prefetching algorithms.



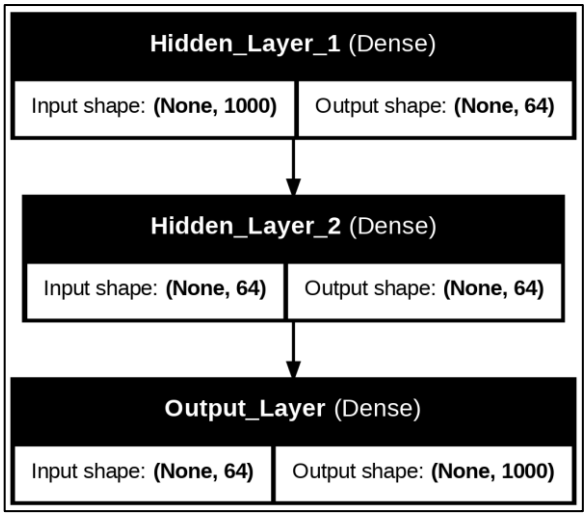
[Fig.6: As the Matrix Multiplication of Transition Probability Increases, The System Becomes More Dynamic]



[Fig.7: CTMC, IMM, and DTMC Model Loss Visualization]



[Fig.8: Propagation Error Minimization Visualization]



[Fig.9: Layered Architecture of the Neural Network Used for Memory Block Predictions]

Figure 9 details the neural network model that was used. The model has two hidden layers, each with 64 units, and a dense output layer that predicts the next memory block. By inputting sequences of memory accesses, the network identifies patterns and predicts future accesses. This architecture was used because it balances computational efficiency and prediction accuracy to minimise the trade-off between the two, maximising performance.

Retrieval Number: 100.1/ijsce.F370015060126
DOI: 10.35940/ijsce.F3700.15060126
Journal Website: www.ijsce.org

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

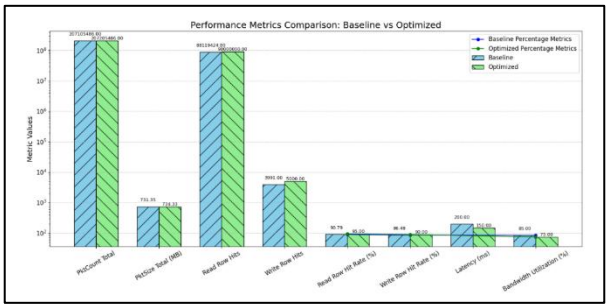
Model: "sequential"

Layer (type)	Output Shape	Param #
Hidden_Layer_1 (Dense)	(None, 64)	64,064
Hidden_Layer_2 (Dense)	(None, 64)	4,160
Output_Layer (Dense)	(None, 1000)	65,000

Total params: 133,224 (520.41 KB)
Trainable params: 133,224 (520.41 KB)
Non-trainable params: 0 (0.00 B)

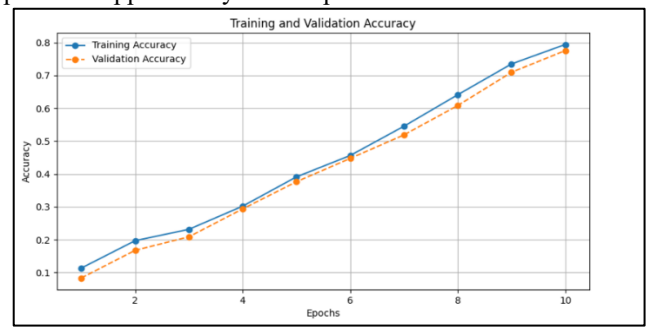
[Fig.10: Quantitative Overview of the Neural Network Parameters]

Figure 10 summarizes the neural network's structure, including the number of parameters per layer. A total of 133,224 parameters indicates a compact yet powerful model. Techniques such as pruning and quantization reduced memory usage without compromising performance, aligning with the multinomial theorem to streamline computations.



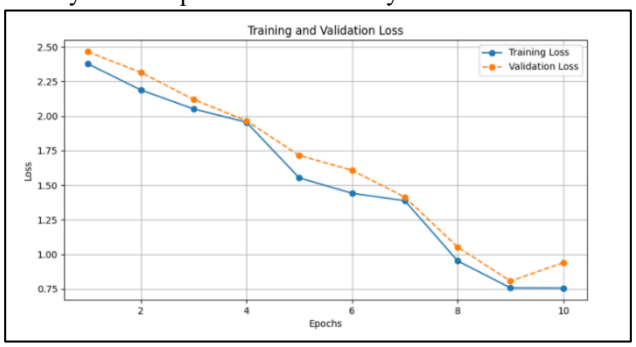
[Fig.11: Performance Metrics Comparison: Baseline vs Optimized Cache Management]

Figure 11 compares performance metrics between the baseline and optimized caching methods. The graph illustrates improvements in metrics such as cache hit rate, bandwidth utilisation, and latency achieved through Markov Chain-based predictive caching and neural network optimisations. The transition from baseline to optimised shows enhancements; however, the significance of these enhancements must be analysed. This analysis was conducted using a Wilcoxon Rank Sum test. With a sample size (**N**) of 8 and a critical value ($\mathbf{W_{crit}}$) of 4, the test yielded a $\mathbf{W_{stat}}$ of 3, leading to the rejection of the null hypothesis. Additionally, a Cohen's d test was performed, producing a value of 0.8325. This indicates a large effect size, signifying substantial practical applicability of the optimizations.



[Fig.12: Training and Validation Accuracy Trends Across Epochs]

The graph illustrates the training and validation accuracy trends for a neural network model over 10 epochs. As depicted, the training and validation accuracies steadily improve, indicating effective learning. The close alignment of training and validation curves suggests that the model generalizes well, with minimal overfitting. This performance is critical when integrating quantization and pruning, as these techniques aim to maintain accuracy while optimizing memory and computational efficiency.



[Fig.13: Training and Validation Loss Trends Across Epochs]

The graph demonstrates the training and validation loss of a neural network model over 10 epochs. Both training and validation loss decrease consistently, reflecting the model's ability to minimize error during training. The convergence of the two curves indicates that the model avoids overfitting, maintaining its performance across unseen validation data. This trend highlights the efficacy of quantization and pruning techniques in reducing computational overhead without compromising model precision.

V. DISCUSSION AND CONCLUSIONS

It can be seen that integrating Markov Chains with neural network modelling for optimisation can enhance data prefetching. This reduces memory stalls and improves overall data transfer efficiency. Additionally, the quantisation and pruning methodologies further enhance the GPU-HBM architecture by reducing memory consumption and computational overhead during data transfer. A solution was achieved by combining predictive cache management with data compression, thereby mitigating the memory bottleneck in GPU-driven applications. The visualisations of transition matrices revealed temporal locality patterns that drove high-probability path optimisations for effective memory access. The cache performance metrics underscored differences in hit and miss distributions across the memory levels, with L1 Cache dominating in terms of hits, while DDR Memory had latency issues. Temporal trends in hit rates for L1 and L2 caches further underscored the need for dynamic optimisation strategies, thereby validating our combined Markov Chain and neural network methodology. Our neural network architecture achieved a balance between predictive power and computational efficiency by leveraging quantisation and pruning, using 133,224 parameters. In addition to reducing memory overhead, these enhancements adhered to data prefetching objectives, which minimizes latency. This integrated approach delivers a portable, efficient solution for various GPU workloads, laying the foundation for the next generation of memory management systems in high-performance computing.

Further work will extend these methods to larger classes of workloads and architectures and address more practical deployment issues to unleash

Retrieval Number: 100.1/ijsce.F370015060126
DOI: 10.35940/ijsce.F3700.15060126
Journal Website: www.ijsce.org

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

their Full transformational power.

While our model works well in most scenarios, there are a few exceptions to cover. It was assumed that memory access patterns exhibit sufficient temporal locality to be modelled as a Markov Chain, based on previous observations of workloads in which access patterns followed predictable sequences. Not only that, but it is assumed that the transition probabilities between memory blocks remain stationary over time, which aligns with the fundamental properties of regular Markov Chains. While this simplifies the computation behind the stationary distribution, it may not hold for workloads with irregular access patterns. This could potentially limit the model's adaptability. The neural network also assumes that past sequences of memory accesses sufficiently encode the necessary context to predict future accesses, which may not be accurate for workloads with non-linear or long-range dependencies. While quantisation and pruning techniques may reduce memory usage and computational complexity, they can also introduce a trade-off between model accuracy and efficiency, especially in scenarios where high precision is critical. Lastly, prefetching strategies derived from graph representations of Markov Chains tend to focus on high-probability paths, potentially neglecting rare yet impactful transitions. This limitation would lead to occasional mispredictions and underutilization of cache resources. Overall, these limitations suggest that while our approach works well for many scenarios, it may require refinements or different methods to address workloads with non-stationary, irregular, or highly complex memory access patterns—International Journal of Soft Computing and Engineering 2025.

DECLARATION STATEMENT

I must verify the accuracy of the following information as the article's author.

Some of the references cited are outdated, noted explicitly as [1], [2], [3] and [9]. However, these works remain significant for the current study, as they are pioneering in their fields.

- **Conflicts of Interest/ Competing Interests:** Based on my understanding, this article has no conflicts of interest.
- **Funding Support:** This article has not been funded by any organizations or agencies. This independence ensures that the research is conducted objectively and without external influence.
- **Ethical Approval and Consent to Participate:** The content of this article does not necessitate ethical approval or consent to participate with supporting documentation.
- **Data Access Statement and Material Availability:** The adequate resources of this article are publicly accessible.
- **Author's Contributions:** The authorship of this article is contributed solely.

REFERENCES

1. Joseph, D., & Grunwald, D. (2002, August 06). Prefetching using Markov predictors. IEEE Journals & Magazine. DOI: https://doi.org/10.1109/12.75265, works remain significant, see the declaration
2. Jog, A., Kayiran, O., Mishra, A. K., Kandemir, M. T., Mutlu, O., Iyer, R., & Das, C. R. (2013, June 23). Orchestrated scheduling and prefetching for GPGPUs. Association for Computing Memory. DOI: https://doi.org/10.1145/2485922.2485951, works remain significant, see the declaration
3. Bauer, M., Cook, H., & Khailany, B. (2011, November 12). CUDADMA: Optimizing GPU memory bandwidth via warp specialization. Association for Computing Machinery. DOI: https://doi.org/10.1145/2063384.2063400, works remain significant, see the declaration
4. Liang, T., Glossner, J., Wang, L., Shi, S., & Zhang, X. (2021, January 24). Pruning and quantization for deep neural network acceleration: a survey. arXiv.org. DOI: https://doi.org/10.48550/arXiv.2101.09671
5. Shi, Z., Huang, X., Jain, A., & Lin, C. (2019, October 12). Applying deep learning to the cache replacement problem. Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture. DOI: https://doi.org/10.1145/3352460.3358319
6. Chopra, B. (2024, May 7). Enhancing machine learning Performance: the role of GPU-based AI computer architectures. Journal of Knowledge Learning and Science Technology ISSN 2959-6386 (Online), 3(3), 20–32. DOI: https://doi.org/10.60087/jklst.vol3.n3.p20-32
7. Hou, J., Tao, T., Lu, H., & Nayak, A. (2023, June 22). Intelligent caching with graph neural network-based deep reinforcement learning on SDN-based ICN. Future Internet, 15(8), 251. DOI: https://doi.org/10.3390/fi15080251
8. Bakhoda, A., Yuan, G. L., Fung, W. W. L., Wong, H., & Aamodt, T. M. (2009, April 1). Analyzing CUDA workloads using a detailed GPU simulator. IEEE Conference Publication. DOI: https://doi.org/10.1109/ISPASS.2009.4919648
9. Liu, A., & Tucker, A. (1988). Applied Combinatorics. DOI: https://doi.org/10.1137/1030075, works remain significant, see the declaration
10. Mittal, S. (2015, January 16). A survey of techniques for managing and leveraging caches in GPUs. Journal of Circuits, Systems and Computers, 23(08), 1430002. DOI: https://doi.org/10.1142/s0218126614300025, works remain significant, see the declaration

AUTHOR'S PROFILE

Samiel Azmaien, Research Assistant at Georgia Institute of Technology, Department of Computer Science. A college freshman who is an extraordinarily detail-oriented software developer. Has 4 years of experience in self-learning computer science and designing applications. Proficient in multiple programming languages, software development methodologies, and database management systems. Strong problem-solving skills and ability to work effectively in a team-based environment through efficient time management and adaptability to manage any difficulties.

Electronic Supplementary Material, the online version of this article (https://tinyurl.com/3medmt2j) contains Supplementary material available to authorised users.