aerospace

MDPI

Article

GPU-Accelerated CNN Inference for Onboard DQN-Based Routing in Dynamic LEO Satellite Networks

Changgeun Yu [1], Daeyeon Kim [2], Heoncheol Lee [1,2,*] and Myonghun Han [3]

[1] School of Electronic Engineering, Kumoh National Institute of Technology, Gumi 39177, Republic of Korea
[2] Department of IT Convergence Engineering, Kumoh National Institute of Technology, Gumi 39177, Republic of Korea
[3] Agency for Defense Development, Daejeon 34186, Republic of Korea
* Correspondence: hclee@kumoh.ac.kr; Tel.: +82-54-478-7476

Abstract: This paper addresses the issue of onboard inference for AI-based routing algorithms in dynamic LEO (Low Earth Orbit) satellite networks. In dynamic LEO networks, it is essential to maintain communication performance across varying topologies while considering link disconnections and overcoming computational constraints for real-time inference on embedded boards. This paper proposes a GPU-based inference acceleration method to reduce the computation time required for real-time onboard inference of a Dueling DQN (Deep Q-Network)-based routing algorithm in dynamic LEO satellite networks. The approach is composed of memory management, low-level operations, and efficient indexing methods, which collectively enhance computational efficiency. As a result, the proposed method achieves approximately 2.4 times faster inference compared to conventional CPU-based approaches. Additionally, the kernel performance analysis reveals that the proposed method reaches 10% of the peak computational performance and 20% of the peak memory performance. This demonstrates the compatibility of the proposed method for integration with additional applications in the multitasking systems of LEO satellites.

Keywords: LEO satellite networks; routing; DQN; GPU; onboard computer; real-time inference

Citation: Yu, C.; Kim, D.; Lee, H.; Han, M. GPU-Accelerated CNN Inference for Onboard DQN-Based Routing in Dynamic LEO Satellite Networks. *Aerospace* **2024**, *11*, 1028. https://doi.org/10.3390/aerospace11121028

Academic Editor: Andrea Colagrossi

Received: 11 October 2024
Revised: 13 December 2024
Accepted: 14 December 2024
Published: 16 December 2024

1. Introduction

LEO satellites operate in orbits ranging from approximately 200 to 2000 km above Earth. Because of their close proximity to Earth, the signal transmission time between LEO satellites and Earth is very short. This provides a significant advantage over satellites at higher altitudes, as LEO satellites have very low latency, making applications such as real-time communication and remote operations feasible [1]. Additionally, for Earth observation, LEO satellites are capable of capturing high-resolution images. The amount of the Earth's surface captured by the satellite's camera depends on the altitude, with LEO satellites offering detailed information. They are commonly used in industries such as climate change monitoring and disaster response [2]. Another key feature of LEO satellites is that they are highly affected by Earth's gravity, which causes them to move relatively quickly. The lower the orbit, the shorter the time it takes for the satellite to complete one orbit, allowing it to pass the same location more frequently and enabling rapid global coverage. By operating LEO satellites in large numbers and forming a network, it is possible to overcome geographical constraints and provide flexible and reliable communication services [3]. Such LEO satellite networks are a core component in modern satellite communications, enabling stable communication in regions with inadequate infrastructure, such as oceans, mountainous areas, and developing countries [4]. For example, there is the Starlink network, an LEO satellite network developed by SpaceX with the goal of providing fast and reliable internet connections worldwide. This network is expected to bring various social and economic benefits beyond simply providing internet services [5].

However, despite the many advantages of LEO satellites, they face several challenges and limitations due to their unique environment in comparison to ground-based communication networks. LEO satellites orbit the Earth rapidly, which causes frequent changes in the connection status between communication nodes. This rapid mobility results in unstable satellite-to-satellite links, leading to frequent communication link disconnection [6]. In satellite communications, where reliable data transmission is critical, link disconnections can cause packet loss, reducing communication quality. This means that continuous reconnections with new satellites must be established due to the constantly changing topology of the satellite network [7]. Frequent updates to routing tables or configurations can introduce overhead, and particularly with variable latency, this can impact network stability. Furthermore, since LEO satellite networks must dynamically set up routes, designing effective routing protocols is challenging and becomes more complex as the number of satellites increases. Efficient routing in LEO satellite networks, which provide essential services to modern society, remains a major challenge, and various algorithmic studies are being conducted to address this issue.

While these approaches open up new possibilities beyond the limitations of traditional algorithms, LEO satellites face constraints in energy and computational resources. Satellites must orbit quickly, and to secure wide coverage, many satellites need to be deployed cost-effectively. Therefore, LEO satellite routing algorithms must consider the resource limitations of onboard computers and optimize energy consumption [8,9]. These limited computational resources and power supply pose significant constraints on performing large-scale data processing or complex computations. However, real-time inference still needs to be performed in such an environment, making computational time a critical factor. Real-time inference refers to the ability of the satellite system to process data and make decisions in real time, requiring a quick response. Optimization is necessary to avoid prolonged computation times, and methods to maximize the efficient use of limited resources are essential. For instance, parallel processing technologies, like GPUs or FPGAs, can be used to increase computation speed, or the algorithm itself can be optimized to reduce computational complexity. Therefore, this study proposes an efficient architecture leveraging GPU parallel processing technology to meet the constraints of limited computational resources and real-time inference in LEO satellites while effectively handling large-scale data processing and complex computations required for advanced routing algorithms. This approach not only addresses the technical limitations but also has the potential to revolutionize the range and quality of services that LEO satellite networks can provide in the future. Furthermore, the feasibility of this architecture is directly validated onboard, offering a practical and innovative solution for real-world applications. Our main contributions are as follows:

- Implementing a DQN-based inference process considering disconnection between LEO satellites in dynamic networks.
- Proposing a GPU-based parallelization method to accelerate the DQN-based inference process for real-time systems.
- Verifying the improved performance of the proposed method in real onboard systems with limited resources.

Section 2 introduces related work. In Section 3, we discuss grid-based MDPs (Markov decision processes), Dueling DQN, and the challenges of onboard real-time CNN (convolutional neural network) inference. Section 4 introduces GPU-based CNN acceleration techniques in detail, along with the proposed approach for onboard computation. In Section 5, we analyze routing simulation results and evaluate the acceleration performance of the proposed method.

2. Related Work

Table 1 provides a brief overview of related studies, categorized based on whether they consider LEO satellite network environments, reinforcement learning, parallelization, or onboard environments. Ref. [10] applies the Dijkstra algorithm for designing dynamic

routing algorithms for LEO satellites. The Dijkstra algorithm is a method for finding the shortest path from one node to all other nodes, and it can be applied to satellite networks for setting up communication paths between satellites. In this approach, the link database is periodically updated to compute the optimal path. The computed path information is then recorded in the routing table, and a routing algorithm is proposed that can quickly respond to changes in the link status. Ref. [11] focuses on predicting congestion states and distributing loads through link state analysis to optimize routing paths. To achieve this, it combines the GRouting algorithm with GNNs (Graph Neural Networks) and DRL (Deep Reinforcement Learning). A GNN is a deep learning model that learns the structural characteristics of each node and the relationships between nodes based on graph-structured data. DRL, using the learned network representation, dynamically selects the optimal routing paths between satellites. DRL identifies paths that minimize transmission delay while efficiently utilizing network resources. Refs. [12–14] apply reinforcement learning to routing algorithms, where the agent learns the optimal policy by interacting with the environment to maximize future cumulative rewards. These methods, based on immediate rewards, are being studied to quickly find optimal paths in highly dynamic LEO satellite networks. Compared to the DQN, the Dueling DQN employed in this paper demonstrates superior performance in complex environments, such as dynamic network environments. By separating the state-value function and the action-advantage function to calculate the Q-value, Dueling DQN achieves greater efficiency and stability in inference compared to DQN. This is particularly effective in environments where differences between actions are minimal, or state changes occur frequently. In LEO satellite onboard systems, where computational resources are limited and network topology often changes dynamically, Dueling DQN reduces the computational burden of insignificant actions, supporting rapid and reliable inference. This enhances the efficiency of communication path selection and resource management. While this approach overcomes the limitations of existing algorithms and proposes an efficient solution, it requires considerable computational resources. Considering the real-time inference process in LEO satellites, parallelization research is essential to address the constraints of limited computational resources.

Table 1. Overview of related works.

Related Works	LEO-SN	RL	Parallelization	Onboard
[10]	○	X (Dijkstra)	X	X
[11]	○	○ (GNN + DQN)	X	X
[12–14]	○	○ (RL)	X	X
[15,16]	○	○ (Dueling DQN)	○ (FPGA)	○
[17]	X	○ (RL)	○ (GPU)	○
Ours	○	○ (Dueling DQN)	○ (GPU)	○

Refs. [15,16] parallelize LEO satellite network routing algorithms using reinforcement learning on FPGAs to overcome onboard computational constraints. The FPGA's programmable logic (PL) optimizes hardware for computationally intensive tasks and accelerates parallel processing by leveraging techniques like pipelining. While FPGAs have the advantage of efficiently optimizing specific tasks, they may be less favorable in terms of scalability. Additionally, the complexity of hardware design and high initial costs are significant drawbacks. Ref. [17] utilizes GPU parallel processing capabilities to accelerate reinforcement learning algorithms. GPUs, with their thousands of cores, are well-suited for processing large volumes of data simultaneously, making them highly effective for parallel computation. The ability of GPUs to efficiently handle operations like matrix multiplications in reinforcement learning algorithms accelerates model training and inference. Furthermore, GPUs support high-performance software libraries and frameworks (e.g., CUDA and cuDNN), which significantly enhance development efficiency. Therefore,

this paper applies Dueling DQN to provide efficient routing in dynamic network environments and proposes a GPU-parallel processing-based architecture to meet the real-time inference requirements of OBC. The GPU-parallelized Dueling DQN-based routing algorithm, which has not been addressed in previous studies, was experimentally validated in an onboard environment. This study establishes a significant foundation for system implementation and further research. This approach not only enhances computational efficiency but also guarantees real-time routing and stability in resource-constrained environments.

3. Problem Description

In this paper, we apply reinforcement learning to a routing algorithm, emphasizing the need for accelerated routing to ensure that the LEO satellite's onboard computer (OBC) can meet real-time inference requirements. This goal is achieved through a custom-built simulation environment. Grid-based Markov Decision Processes (MDPs) are commonly used in reinforcement learning, where states and actions are defined in a grid-based environment to learn how to follow the optimal path to a goal [18]. In these structured grid layouts, the environment is divided into a grid, with each cell representing a specific state. The agent's goal is to move between cells or change states through actions, earning rewards and moving toward the goal state. This process is stochastic, meaning that while transitions between states are often deterministic in most grid-based problems, randomness can be introduced depending on the environment. These MDPs serve as an important tool for simulating the complex dynamics of routing. In this grid system, one state represents the condition of a particular location or satellite, and satellites move by selecting actions. This framework visually represents routing paths, obstacles, and communication coverage areas, constituting the environment of the LEO satellite network. It also integrates probabilistic transitions between states, meaning that the transitions are not fixed but can probabilistically change depending on the agent's actions and the environment. MDP modeling allows for the definition of the problem within the low Earth orbit satellite network. This model reflects issues like link disconnections caused by the dynamic environment. For instance, link disconnection regions (black cells) are treated as obstacles within the MDP, modeling areas where the agent cannot pass through. By integrating such environmental elements, the transition probabilities for all state–action pairs in the MDP model are calculated [19], and the success probability of each routing path is estimated, allowing for the selection of the optimal path. Considering the dynamic nature of the LEO constellation network, the MDP model, which integrates state transition probabilities, can adapt to the ever-changing environment and make optimal decisions. This proves to be an efficient routing algorithm.

Additionally, Dueling DQN is incorporated into the routing algorithm methodology. This reinforcement learning model differs from traditional Q-learning networks and DQNs [20]. In traditional methods, the Q-function estimates the reward values based on states and actions. The future reward obtained by taking a particular action in a given state is called the Q-value. However, calculating Q-values for all actions across all states can be computationally expensive, as it uses the same conditions for every estimation. To address this, Dueling DQN separates the learning process into two distinct components. The formula to calculate the output Q-value is as follows:

$$\mathrm{Q}(s,a) = \mathrm{V}(s) + \mathrm{A}(s,a) - \frac{1}{|\mathrm{A}|}\sum_{a'}\mathrm{A}(s,a') \tag{1}$$

where s is the current state and a and a' represent the current action and possible alternative actions, respectively. This formula calculates how much more beneficial a particular action is compared to other actions in a given state and adds the value of the state itself to get the final Q-value [21].

However, even with these advanced methodologies, satellite-borne systems on low-orbit satellites must process large amounts of data in real time with limited computational resources, memory, and other constraints, making it difficult for CNNs to perform in-

ferences efficiently. CNNs are capable of extracting local patterns or features through convolutional operations and handling complex data structures via hierarchical feature learning [22]. Nevertheless, traditional onboard computers in satellites struggle to achieve real-time performance with CNNs. In LEO satellites, where real-time processing is critical, if the computation time of a deep learning model conflicts with real-time requirements, other essential tasks on the satellite could be delayed, which would negatively impact the entire network system [23]. To address these challenges, it is necessary to simplify the computation. This simplification can be achieved through hardware acceleration. In particular, the parallel processing capabilities of GPUs, which are highly effective at accelerating AI-based algorithms, can offload intensive computations from onboard computers, enabling faster routing decisions.

Figure 1 provides a visual representation of how an agent performs routing using an embedded board with constrained onboard resources in an environment modeled by MDPs. First, the dynamic satellite environment is represented as a lattice map, with each cell corresponding to a state. Each cell represents either a communicable zone or an obstacle in the satellite routing problem. We applied the MDP model to simulate the dynamic environment. In this setup, the Dueling DQN routing algorithm is employed. While it is an effective routing method, it is computationally intensive, and the satellite's onboard computer struggles to handle the workload effectively. As a result, developing methods that consider the limited resources of onboard computers in the routing environment of low-orbit satellites remains a significant challenge. Traditional onboard systems on most satellites are currently limited in their ability to handle large-scale computations. Therefore, methodologies that reduce the computational burden through parallel processing, utilizing hardware like GPUs, are essential.

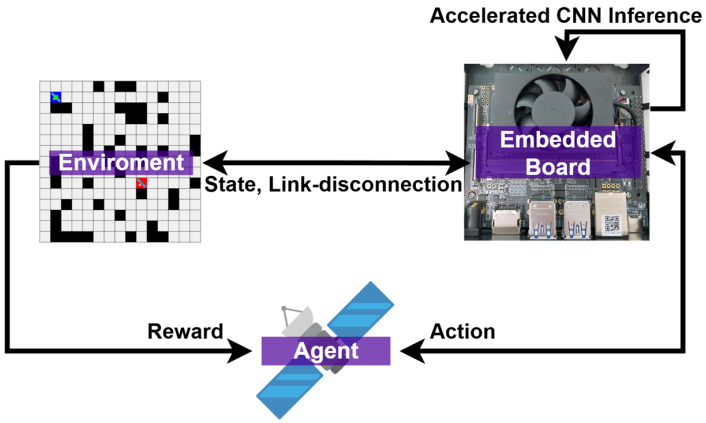
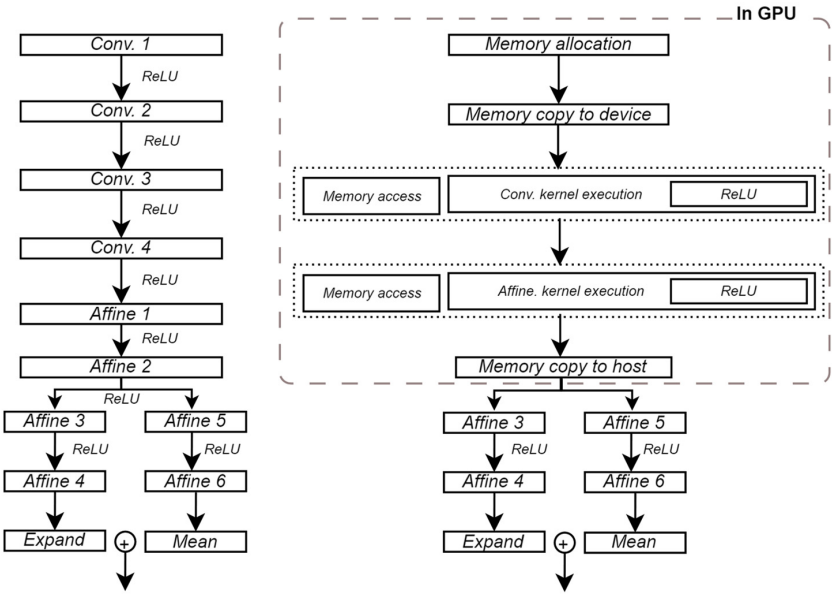


Figure 1. Flowchart of onboard routing inference systems.

4. Proposed Method

The main objective of this research is to present a novel system architecture that optimizes GPU performance to enhance the computational efficiency of large-scale data processing, considering the OBC of low-orbit satellites. As shown in Figure 2, the proposed system architecture is organized hierarchically. One of the primary advantages of this design is its ability to perform parallel computation. By leveraging the GPU's large number of compute cores, thousands of threads can execute simultaneously, enhancing parallel computation efficiency. In CNNs, many operations involve large matrix computations, and the GPU's multi-threaded structure allows these computations to be performed concurrently, achieving much faster speeds compared to a CPU. However, memory management is a crucial factor in performance optimization when performing GPU-based computations. In large-scale operations like CNNs, inefficient memory management creates bottlenecks, causing computational threads to wait unnecessarily for data, which hinders the GPU's ability to fully utilize its parallel processing power. To address this, we pre-allocated memory for weights and biases directly into the GPU's memory. Pre-allocating memory space

maximizes computational performance by minimizing runtime overhead from frequent memory allocation and deallocation. This also improves memory access speeds. Additionally, asynchronous operations are easier to implement when data is pre-allocated in device memory, as there is no need to transfer data during CNN operations. By parallelizing kernel execution and memory transfer, we further maximized the computational resources of the GPU.



(**a**) Conventional approach (**b**) Proposed approach

Figure 2. The comparison of the routing inference architectures in the conventional and proposed methods. (**a**) In the conventional approach, all blocks of the inference are conducted only sequentially on a CPU. (**b**) In the proposed approach, several blocks of the inference are conducted in parallel on a GPU, which can reduce the overall computation time.

4.1. Configuration of the CUDA Kernel

Our system primarily processes three-dimensional input and output arrays. To fully exploit the GPU's parallelism and enable efficient indexing, we adopted 2D threads combined with a 1D block structure aligned with the data structure. This approach effectively captures the inherent 2D spatial properties of the data, such as width and height, using 2D threads. The 1D block structure corresponds to the third dimension, allowing all elements in a 3D array to be indexed accurately and efficiently. Furthermore, maintaining independence between blocks maximizes computational parallelism.

Another feature of the kernel design is the fixed parallel operations on the output array. By fixed parallelism, we refer to a structure where each GPU thread independently processes a specific element of the output array. This one-to-one correspondence between each thread and an output element minimizes data dependency between threads, ensuring deterministic results by guaranteeing that each thread processes exactly one output element. Additionally, this focus on the output array makes it easy to scale the number of threads up or down based on the size and demand of the output data, supporting dynamic computational requirements.

An additional important aspect of the kernel design is the use of arrays for thread identifiers. This choice was made to efficiently handle CNN computations, where specific parts of the input are extracted with different weights. This structure increases the efficiency of parallel computations by allowing each thread to process multiple outputs simultaneously. Moreover, by utilizing the block index, we enabled computations to be performed only on specific blocks, improving efficiency by indexing multiple computed outputs at once. The

specific configuration of the CUDA kernel is shown in Algorithm 1. Here, we also present additional methods proposed in this paper.

Algorithm 1. Configuration of the CUDA kernel

```
1: Initialize thread indices and block index
2: Initialize sum array
3: Calculate output array
4: Initialize local input array
5: for i = 0 ~ input_channel
6:     for j = 0 ~ input_width
7:         for k = 0 ~ input_height
8:             Fill local input
9: Copy shared memory
10: for i = 0 ~ output_array
11:     Perform multiplication and add bias
12: ReLu
13: Write the output
```

4.2. Partitioned Weights for Low-Level Operations

In Figure 3, the weights are systematically divided. Due to the complexity of the input data and the structure of the CNNs, the weights are initially organized into a complex four-dimensional structure, with each dimension representing a specific property of the filter. This is necessary because the filter must learn both the spatial and channel dimensions from the input data simultaneously. To meet the efficient computational requirements of handling weights, we converted the filters into a 3D format for different types of filters, making them more manageable. This conversion provides two main benefits. First, it allows each block to take inputs more efficiently. Inputs in three dimensions are processed using weights also converted into three dimensions, enabling more efficient, low-level computation compared to the traditional structure. Second, we established a mechanism for shared access among all computational threads. Threads within the same block can access shared memory and jointly use the data, reducing memory bandwidth usage and minimizing access to global memory, which speeds up computation. This heuristic approach to weight granularity and structure is not just a superficial adjustment, as it greatly simplifies the indexing mechanism. Multiple threads can access the same data simultaneously, enhancing computation performance. This improved structure allows the system to handle low-level operations more effectively, giving threads faster access to data. The result of this optimization is a method that is fine-tuned for the stringent requirements of GPGPU computing.

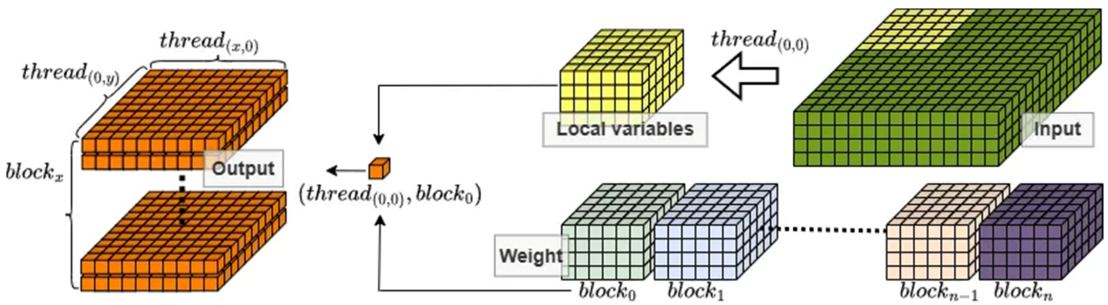


Figure 3. Optimize processing by splitting the weights into distinct blocks and overriding only the parts of the input that correspond to those weights.

4.3. Local Variables for Efficient Indexing

When performing CNNs, overlapping input data can occur, particularly during convolutional operations where filters slide over the input and create areas of overlap. This

overlap can lead to poor performance and increased memory usage, as threads are forced to process multiple non-essential components. To avoid these inefficiencies, we restructured our approach. As shown in Figure 3, we redefined only the portion of the input that corresponds to the weights as local variables. This approach significantly reduces redundant memory accesses because the appropriate weighting and multiplication operations are performed using the redefined input values, avoiding the need to fetch duplicate data. This optimization saves memory bandwidth and reduces the amount of computation caused by overlap, as fewer unnecessary operations are performed simultaneously. Since the number of inputs and kernels used in the Dueling DQN routing scheme for low-orbit satellites is relatively small, this de-duplication of input data improves overall performance.

4.4. Integrated ReLU for Minimize I/O Overheads

In Figure 2, examining the baseline confirms that convolution and ReLU operations are executed separately. In CNNs, ReLU, an activation function, is applied after the convolutional operation to introduce nonlinearity and make the feature map extracted by the filter more meaningful. The traditional approach treats these two processes as separate operations. However, this method posed challenges during kernel development. It led to unnecessary kernel calls and created a memory bottleneck since the GPU's computation speed is much faster than its memory access speed. Moreover, by performing the operations in separate kernels, we lost opportunities for parallelism, leading to computational inefficiency. To address this issue, we adopted a unified approach by combining convolution and ReLU into a single kernel, as shown in Figure 2, instead of separating the two operations. This unified method optimizes the execution process and significantly reduces unnecessary kernel calls, resulting in more efficient computation.

5. Experimental Results

5.1. Experimental Setups

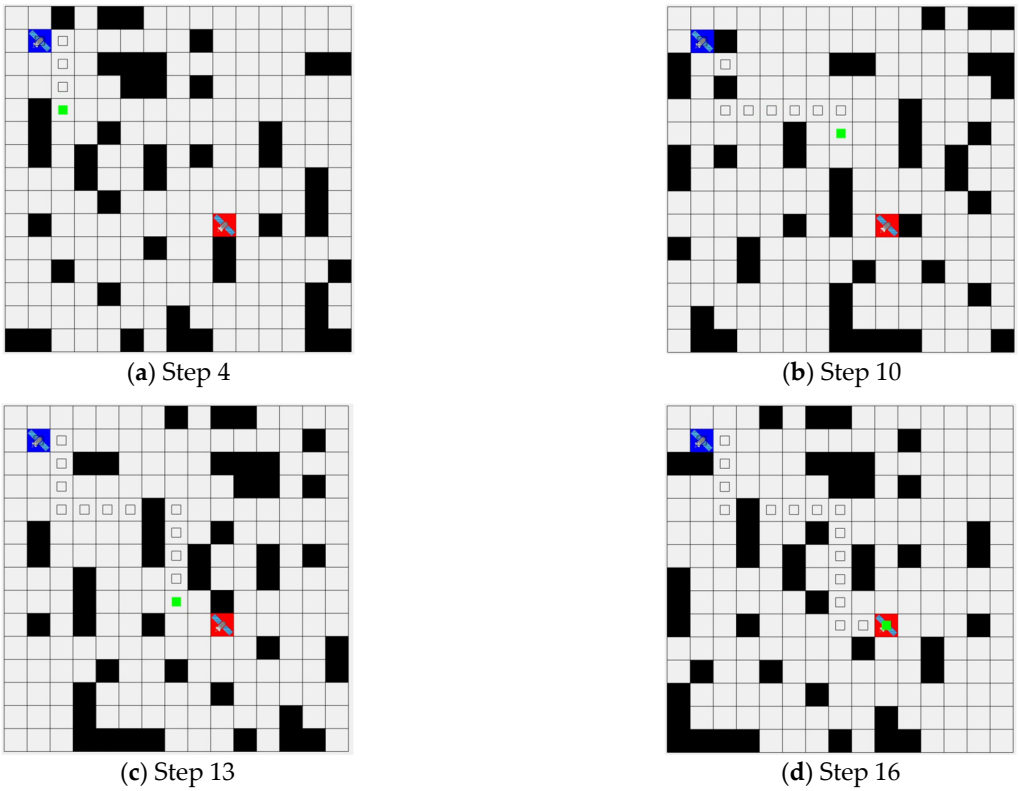
To accelerate the Dueling DQN routing algorithm for low-orbit satellites in an onboard environment, we used the NVIDIA Jetson Orin NX 16 GB embedded board to closely mimic the limited computational capacity of an onboard system across all phases of the application. This approach ensures consistency while utilizing state-of-the-art features. The technical specifications of this board are shown in Table 2. Based on the table, the NVIDIA Jetson Orin NX offers strong AI and GPU performance, allowing it to efficiently handle the high-density computations required by reinforcement learning algorithms like Dueling DQN. It also ensures that power consumption, which is critical for LEO satellites, remains within acceptable limits. The experimental environment provided a dynamic testing environment consisting of four channels: agent location, target location, obstacles, and boundaries, within a 15×15 grid world environment. This grid world introduced an additional layer of complexity, where disconnected regions autonomously moved one position to the left at every discrete time step, mimicking the dynamic challenges present in real-world environments. In this environment, we proposed a system that leverages GPU acceleration for CNN inference in the Dueling DQN routing scheme deployed on the embedded board. The kernel was implemented using the PyCUDA library to ensure smooth operation within the Python environment, and PyTorch v1.12.0 was selected as the deep learning framework for this research.

Table 2. NVIDIA Jetson Orin NX 16 GB board technical specification.

Specification	Detail
AI Performance	100TOPs
GPU	1024-core NVIDIA Ampere GPU featuring 32 Tensor Cores
CPU	8-core Arm Cortex-A78AE v8.2 64-bit CPU
Memory	16 GB 128-bit LPDDR5 102.4 GB/s
Power	10 W–25 W

5.2. Result and Analysis

Figure 4 shows the inference results of a model trained in the grid environment following Dueling DQN principles. Blue indicates the starting position, red represents the goal, black denotes obstacles (link disconnections), green dots mark the current position, and small squares trace the path. In Figure 4b–d, although there are obstacles on the path due to the dynamic environment, they can be ignored as they represent sidesteps from points where the satellite was not affected. Demonstrating excellent navigation capabilities, the model skillfully avoids obstacles that would cause link disconnections or pose dynamic challenges, reaching the specified destination in just 16 steps. These results emphasize not only the efficiency of the trained model but also its ability to effectively navigate complex scenarios. This stems from the pre-separation of state values and action advantages in Dueling DQN, which minimizes the agent's unnecessary path exploration and demonstrates its applicability to real-time routing problems. Particularly in dynamic environments, Dueling DQN excels at adapting to rapid changes and efficiently evaluating and selecting optimal actions, showcasing superior routing performance.



(**a**) Step 4 (**b**) Step 10

(**c**) Step 13 (**d**) Step 16

Figure 4. Inference results of the Dueling Deep Q-Network-based routing simulation. (Blue is the starting position, red is the target position, black is the obstacles (link breaks), the green dot is the current position of the satellite, and the small square you follow is the path.)

Figure 5 compares the overall CNN inference performance times for the routing algorithm applied in the onboard environment. It shows the difference in inference time between the conventional CPU-based computation commonly used in OBCs and the GPU-based computation using the proposed methodology. Each value is based on 10 runs of the applied method. The baseline refers to CNN inference using only loops, while MKL-Optimized represents CNN inference using only the PyTorch framework, with both methods relying solely on the CPU for computation. MKL-Optimized (GPU) refers to the process of transferring the mentioned MKL method to the GPU for computation. This method was performed for comparison with the proposed GPU architecture. The Parallelized Main refers to the parallelization of the main layer using the GPU, which

shows a performance improvement of 1.590 times faster than MKL-Optimized (GPU). This demonstrates that MKL-Optimized (GPU) shows lower performance due to the lack of parallel processing optimization, while the proposed method exhibits superior computational efficiency, making it a more suitable approach for meeting real-time requirements. Finally, Parallelized Main, Sub involves the parallelization of both the main and sub-layers, resulting in the lowest inference time. The performance improvement between the CPU-based computation and the main layer parallelization is shown by the red line, with a 2.205 times faster performance.

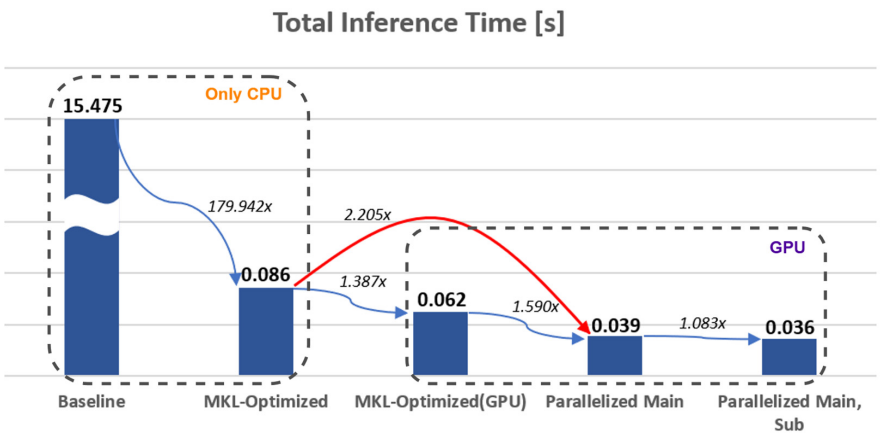


Figure 5. Comparative analysis of the inference operation times between the baseline and the proposed system.

Figure 6 presents the metrics that evaluate the extent to which the kernel utilizes peak performance. These metrics were obtained using Nsight Compute, a GPU profiling tool provided by NVIDIA, which offers valuable insights into the performance of GPU kernels. The measurements indicate that most kernels achieve around 10% of compute peak performance and 20% of memory peak performance. While low peak performance metrics may initially seem suboptimal, in the context of resource-constrained environments like satellite onboard systems, they highlight the potential for integrating additional applications. In these environments, low compute and memory peak performance can prove advantageous. Low-orbit satellites are multitasking systems that not only manage routing but also perform real-time data processing, satellite health monitoring, and communications management. Therefore, the observed performance, while low, is a positive factor in evaluating the system's suitability for multitasking and concurrent operations within the satellite's overall mission objectives.

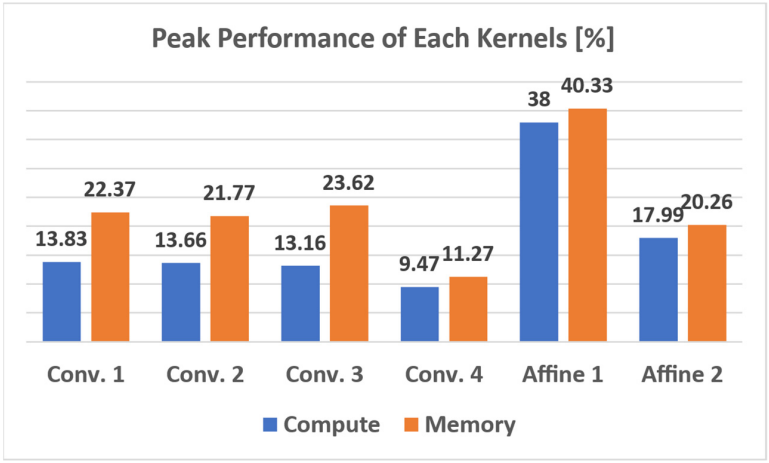


Figure 6. The peak performance profiling results of the proposed kernels.

6. Conclusions

This paper proposes a GPU-accelerated CNN inference architecture for DQN-based routing in dynamic LEO satellite networks. The proposed architecture combines advanced routing algorithms capable of adapting to dynamic environments, such as link disconnections, with GPU-based acceleration techniques to effectively address computational complexity. Experimental results demonstrate that the proposed Dueling DQN-based routing algorithm achieves high exploration performance in dynamic environments with link disconnections. Furthermore, it significantly outperforms existing methodologies, proving its suitability for real-time inference in LEO satellite environments. These results confirm that the proposed architecture meets real-time exploration requirements by enabling faster routing in dynamic environments, including link disconnections. This leads to substantial improvements in onboard system efficiency and performance by reducing energy consumption, enhancing reliability, and providing additional computational capacity. Such technical advantages not only support stable and efficient communication services in LEO satellite networks but also lay a critical foundation for developing real-time routing and optimization solutions and advancing technologies for future onboard applications.

Author Contributions: C.Y., D.K. and H.L.: methodology, software, validation, and writing—original draft preparation. M.H.: conceptualization, formal analysis, and writing—review and editing. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Agency for Defense Development, funded by the Korean government (Defense Acquisition Program Administration) (UI220033VD).

Data Availability Statement: Data is contained within the article.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Hassan, N.U.L.; Huang, C.; Yuen, C.; Ahmad, A.; Zhang, Y. Dense Small Satellite Networks for Modern Terrestrial Communication Systems: Benefits, Infrastructure, and Technologies. *IEEE Wirel. Commun.* **2020**, *27*, 96–103.
2. Giuffrida, G.; Fanucci, L.; Meoni, G.; Batič, M.; Buckley, L.; Dunne, A.; van Dijk, C.; Esposito, M.; Hefele, J.; Vercruyssen, N.; et al. The Φ-Sat-1 Mission: The First On-Board Deep Neural Network Demonstrator for Satellite Earth Observation. *IEEE Trans. Geosci. Remote Sens.* **2022**, *60*, 1–14.
3. Yang, Y.; Mao, Y.; Ren, X.; Jia, X.; Sun, B. Demand and key technology for a LEO constellation as augmentation of satellite navigation systems. *Satell. Navig.* **2024**, *5*, 11.
4. Osoro, O.B.; Oughton, E.J.; Wilson, A.R.; Rao, A. Sustainability assessment of Low Earth Orbit (LEO) satellite broadband mega-constellations. *arXiv* **2023**, arXiv:2309.02338.
5. Shaengchart, Y.; Kraiwanit, T. Starlink satellite project impact on the Internet provider service in emerging economies. *Res. Glob.* **2023**, *6*, 100132. [CrossRef]
6. Zhang, J.; Cai, Y.; Xue, C.; Xue, Z.; Cai, H. LEO Mega Constellations: Review of Development, Impact, Surveillance, and Governance. *Space Sci. Technol.* **2022**, *2022*, 9865174. [CrossRef]
7. Bi, Y.; Han, G.; Xu, S.; Wang, X.; Lin, C.; Yu, Z.; Sun, P. Software Defined Space-Terrestrial Integrated Networks: Architecture, Challenges, and Solutions. *IEEE Netw.* **2019**, *33*, 22–28. [CrossRef]
8. Chen, F.; Wang, Q.; Ran, Y. Dynamic Routing Algorithm for Maximizing Battery Life in LEO Satellite Networks. In Proceedings of the 2022 IEEE 8th International Conference on Computer and Communications (ICCC), Chengdu, China, 9–12 December 2022; Volume 8; pp. 671–676.
9. Song, Z.; Hao, Y.; Liu, Y.; Sun, X. Energy-Efficient Multiaccess Edge Computing for Terrestrial-Satellite Internet of Things. *IEEE Internet Things J.* **2021**, *8*, 14202–14218. [CrossRef]
10. Lu, Z.; Zhi, R.; Ma, W. Quick Routing Response to Link Failure in Low-Earth Orbit Satellite Networks. In Proceedings of the 2022 IEEE 8th International Conference on Computer and Communications (ICCC), Chengdu, China, 9–12 December 2022; Volume 8; pp. 690–695.
11. Wang, H.; Ran, Y.; Zhao, L.; Wang, J.; Luo, J.; Zhang, T. GRouting: Dynamic Routing for LEO Satellite Networks with Graph-based Deep Reinforcement Learning. In Proceedings of the 2021 4th International Conference on Hot Information-Centric Networking (HotICN), Nanjing, China, 25–27 November 2021; Volume 4; pp. 123–128.
12. Wang, X.; Dai, Z.; Xu, Z. LEO Satellite Network Routing Algorithm Based on Reinforcement Learning. In Proceedings of the 4th International Conference on Electronics Technology (ICET), Chengdu, China, 7–10 May 2021; Volume 4; pp. 1105–1109.

13. Tsai, K.-C.; Yao, T.-J.; Huang, P.-H.; Huang, C.-S.; Han, Z.; Wang, L.-C. Deep Reinforcement Learning-Based Routing for Space-Terrestrial Networks. In Proceedings of the 2022 IEEE 96th Vehicular Technology Conference (VTC2022-Fall), London, UK, 26–29 September 2022; Volume 96; pp. 1–5.
14. Zuo, P.; Wang, C.; Yao, Z.; Hou, S.; Jiang, H. An Intelligent Routing Algorithm for LEO Satellites Based on Deep Reinforcement Learning. In Proceedings of the 2021 IEEE 94th Vehicular Technology Conference (VTC2021-Fall), Norman, OK, USA, 27–30 September 2021; Volume 94; pp. 1–5.
15. Kim, H.; Park, J.; Lee, H.; Won, D.; Han, M. An FPGA-Accelerated CNN with Parallelized Sum Pooling for Onboard Realtime Routing in Dynamic Low-Orbit Satellite Networks. *Electronics* **2024**, *13*, 2280. [CrossRef]
16. Kim, D.; Lee, H.; Won, D.; Han, M. FPGA-based Inference Parallelization for Onboard RL-based Routing in Dynamic LEO Satellite Networks. *Int. J. Aeronaut. Space Sci.* **2024**, *25*, 1135–1145.
17. Dalton, S.; Frosio, I.; Garland, M. Accelerating reinforcement learning through GPU Atari emulation. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 19773–19782.
18. Chen, S.; Zhang, X.; Chen, X.; Li, Z.; Wang, Y.; Lin, Q.; Xu, J. Reinforcement Re-ranking with 2D Grid-based Recommendation Panels. *arXiv* **2022**, arXiv:2204.04954.
19. Nawaz, F.; Ornik, M. Multiagent, Multitarget Path Planning in Markov Decision Processes. *IEEE Trans. Autom. Control* **2023**, *68*, 7560–7574. [CrossRef]
20. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [CrossRef] [PubMed]
21. Cao, J.; Wang, X.; Wang, Y.; Tian, Y. An improved Dueling Deep Q-network with optimizing reward functions for driving decision method. *Proc. Inst. Mech. Eng. Part D J. Automob. Eng.* **2023**, *237*, 2295–2309. [CrossRef]
22. Alzubaidi, L.; Zhang, J.; Humaidi, A.J.; Al-Dujaili, A.; Duan, Y.; Al-Shamma, O.; Santamaría, J.; Fadhel, M.A.; Al-Amidie, M.; Farhan, L. Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions. *J. Big Data* **2021**, *8*, 53. [CrossRef] [PubMed]
23. Al Homssi, B.; Al-Hourani, A.; Wang, K.; Conder, P.; Kandeepan, S.; Choi, J.; Allen, B.; Moores, B. Next Generation Mega Satellite Networks for Access Equality: Opportunities, Challenges, and Performance. *IEEE Commun. Mag.* **2022**, *60*, 18–24. [CrossRef]