



INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

VOL 04. ISSUE 02 . (2026)

<http://ijirjournal.com/>

Joint Configuration Optimization for Edge-Cloud Collaborative Inference in Visual IoT Systems

Margaret Wai-Kit Choi

School of Computer Science, Peking University, Beijing 100871, China

Linda Woo

School of Computer Science, Peking University, Beijing 100871, China

Abstract.

The rapid proliferation of visual Internet of Things applications, such as autonomous navigation, smart surveillance, and augmented reality, has driven the demand for real-time and highly accurate computer vision systems. However, executing complex deep neural networks exclusively on resource-constrained end devices is impractical due to severe computational and energy limitations. Relying solely on cloud computing introduces unacceptable communication latency and significant bandwidth consumption, which are detrimental to time-sensitive applications. To address these challenges, edge-cloud collaborative inference has emerged as a promising paradigm. In this architecture, deep learning models are dynamically partitioned between end devices, edge servers, and cloud servers. This paper presents a comprehensive framework for the joint configuration optimization of edge-cloud collaborative inference in visual Internet of Things systems. The proposed framework simultaneously optimizes multiple critical parameters, including camera frame rate, image capture resolution, neural network partition points, and intermediate feature map quantization levels. By formulating the configuration selection as a multidimensional optimization problem, we aim to minimize end-to-end inference latency and overall system energy consumption while strictly satisfying predefined baseline accuracy requirements. Due to the highly dynamic nature of network conditions and the non-convexity of the problem, we develop a deep reinforcement learning approach that adapts to time-varying channel states and server workloads. Extensive simulation results demonstrate that the proposed joint optimization strategy significantly outperforms conventional edge-only, cloud-only, and static partitioning baselines across various network environments. The findings provide critical insights into the design and deployment of efficient, scalable, and robust visual sensing networks for next-generation smart city infrastructure.

Keywords: *Visual Internet of Things, Edge Computing, Collaborative Inference, Deep Reinforcement Learning, Resource Optimization.*

1. Introduction

1.1 Background And Motivation

The Internet of Things has evolved significantly over the past decade, transitioning from simple scalar sensor networks to complex, high-bandwidth visual sensor networks. Visual Internet of Things systems deploy a vast array of cameras and optical sensors to capture rich multimedia data, enabling advanced applications such as intelligent traffic monitoring, facial recognition, industrial defect detection, and immersive augmented reality. These applications fundamentally rely on deep convolutional neural networks and transformer-based architectures to extract meaningful semantic information from raw video streams. While these deep learning models achieve unprecedented accuracy in object detection and image classification, they are notoriously computationally expensive and memory-intensive [1]. End devices in visual Internet of Things ecosystems typically possess limited battery life, restricted memory capacity, and constrained processing power, making it entirely unfeasible to execute state-of-the-art computer vision models locally. Traditionally, the paradigm for handling such computationally demanding tasks involved offloading the entirety of the captured raw video data to centralized cloud data centers. The cloud infrastructure provides virtually unlimited computational resources, allowing for the rapid execution of large-scale deep neural networks. However, this cloud-centric approach suffers from severe limitations in the context of modern visual applications. The continuous transmission of high-resolution video streams from thousands of end devices to remote cloud servers consumes an immense amount of network bandwidth [2]. Furthermore, the geographical distance between the data source and the cloud data center introduces substantial propagation delays, routing latency, and network jitter [3]. For mission-critical applications like autonomous driving or real-time industrial robotics, where decisions must be made within milliseconds, the latency introduced by cloud-only processing is entirely unacceptable. Furthermore, transmitting unencrypted or lightly encrypted raw video feeds over public backhaul networks raises significant privacy and data security concerns. To overcome the inherent limitations of cloud-only and local-only processing, the paradigm of multi-access edge computing has been integrated into visual Internet of Things architectures. Edge computing brings computational resources, storage capabilities, and network management functions closer to the data source, typically deploying edge servers at cellular base stations, local gateways, or on-premises enterprise servers [4]. By offloading computational tasks to the edge, systems can drastically reduce communication latency, alleviate backhaul network congestion, and enhance data privacy. However, edge servers still possess finite computational resources compared to hyperscale cloud data centers. During periods of peak demand, edge servers can easily become overloaded, leading to severe queueing delays that negate the initial benefits of edge offloading. This realization has catalyzed the development of edge-cloud collaborative computing, a hybrid architecture that synergistically leverages the low-latency characteristics of edge nodes and the immense computational power of the cloud [5]. In this collaborative paradigm for visual inference, a deep neural network is conceptually treated as a directed acyclic graph of computational layers. This graph can be partitioned into two or more segments. The initial layers, which typically extract low-level features such as edges and textures, are executed on the local camera or the nearby edge server. The intermediate feature maps generated by these early layers are then transmitted to the cloud server, where the remaining resource-intensive layers perform high-level semantic reasoning and final classification. This approach requires careful configuration to balance the tradeoffs between computation time, transmission time, and energy consumption.

1.2 The Need For Joint Configuration Optimization

While edge-cloud collaborative inference provides a robust structural foundation for modern visual Internet of Things systems, realizing its full potential requires meticulous configuration and continuous optimization. The performance of a collaborative inference system is governed by a multitude of interacting parameters spanning the application layer, the computing layer, and the networking layer [6]. At the application layer, the configuration of the visual sensor itself plays a crucial role. The frame rate dictates how many images are captured and processed per second, directly influencing both the temporal resolution of the application and the workload generated. The spatial resolution of the captured images determines the size of the input tensor fed into the deep neural network. Higher resolutions generally yield better inference accuracy, particularly for detecting small objects at a distance, but they exponentially increase the computational burden and the size of the resulting feature maps [7]. At the computing layer, the primary configuration parameter is the selection of the neural network partition point. Choosing an early partition point means that a minimal amount of computation is performed locally or at the edge, but the intermediate feature maps that must be transmitted over the network may be excessively large. In fact, due to the nature of early convolutional layers, the data volume of intermediate feature maps often exceeds the size of the original compressed raw image [8]. Conversely, selecting a later partition point reduces the communication payload but places a heavier computational burden on the resource-constrained edge device. At the networking layer, intermediate data compression and quantization configurations are critical. The feature maps generated at the partition point consist of high-precision floating-point numbers. Transmitting these dense tensors over wireless channels introduces significant delay. Therefore, systems employ quantization techniques to reduce the precision of the feature maps, such as converting 32-bit floats to 8-bit integers, or employ spatial compression algorithms. While these techniques drastically reduce bandwidth requirements, aggressive compression inevitably degrades the final inference accuracy [9]. Optimizing any single parameter in isolation invariably leads to sub-optimal system performance. For instance, maximizing image resolution and frame rate to achieve the highest possible accuracy will flood the network with data, causing severe congestion, massive latency spikes, and rapid battery depletion on the end devices. Conversely, aggressively lowering the resolution and selecting a deep partition point to save bandwidth may render the edge server incapable of meeting real-time processing deadlines, while also dropping the inference accuracy below acceptable thresholds. Therefore, it is absolutely essential to formulate a joint configuration optimization strategy that simultaneously orchestrates the camera settings, the network partition point, and the data compression ratio. The primary contribution of this research is the development of a comprehensive, dynamic optimization framework that minimizes a holistic cost function encompassing both latency and energy consumption while strictly adhering to accuracy constraints. We introduce a novel state-space representation that captures the intricate dependencies between application-level configurations and network-level dynamics. By leveraging advanced reinforcement learning methodologies, our proposed system autonomously learns the optimal configuration policies under varying environmental conditions, without requiring prior knowledge of the underlying network transition probabilities.

2. Literature Review And Background

2.1 Evolution Of Computational Offloading In Iot

The concept of computational offloading has been a cornerstone of mobile and ubiquitous computing for several decades. Early research primarily focused on binary offloading decisions,

where an entire application or task was either executed entirely on the local device or transmitted entirely to a remote server. These systems typically utilized static profiling techniques to estimate the execution time and energy consumption of various software components. When the local device was deemed insufficient to meet application deadlines, or when the energy cost of local execution exceeded the cost of transmission, the task was encapsulated and migrated to the cloud. As applications grew more complex, particularly with the advent of deep learning, binary offloading became increasingly inadequate. Deep neural networks consist of millions of parameters and require billions of floating-point operations per inference pass. Researchers soon realized that the monolithic offloading of such models was highly inefficient. This led to the exploration of partial offloading, where applications were decomposed into independent or loosely coupled tasks. In the context of computer vision, partial offloading allowed early preprocessing steps, such as image cropping and background subtraction, to be performed locally, while the heavy classification algorithms were executed remotely. The introduction of mobile edge computing paradigms revolutionized partial offloading by providing intermediate computational tiers. The literature is replete with studies optimizing task allocation in multi-tier edge-cloud environments. Some researchers formulated the edge offloading problem as a mixed-integer non-linear programming problem, aiming to minimize the total energy consumption of all mobile users subject to strict latency constraints [10]. Others utilized game-theoretic approaches to resolve resource contention among multiple end devices competing for limited edge server capacity. Despite these advancements, many of these generalized task offloading frameworks treated computational tasks as black boxes, ignoring the unique structural properties of deep neural networks.

2.2 Deep Neural Network Partitioning Strategies

The structural characteristics of modern deep convolutional neural networks make them exceptionally well-suited for layer-level partitioning. A typical network consists of a linear or branched sequence of layers, including convolutional layers, pooling layers, activation functions, and fully connected layers. Each layer takes the output tensor of the previous layer as its input, performs a specific mathematical transformation, and produces an output feature map. Early pioneering work in collaborative inference introduced the concept of neurosurgeon approaches, which dynamically profiled the execution time and output data size of each layer in networks like AlexNet and VGG. These studies demonstrated that the data volume does not decrease monotonically as it traverses the network [11]. Often, the first few convolutional layers expand the dimensionality of the data, resulting in intermediate feature maps that are significantly larger than the input image. Only after several pooling and strided convolution operations does the data volume decrease sufficiently to warrant transmission. Consequently, selecting the optimal partition point requires an exhaustive analysis of the per-layer computational latency versus the transmission latency associated with the layer's output size. Further advancements in model partitioning explored the concept of early-exit architectures. In these designs, auxiliary classifier branches are attached to intermediate layers of the primary backbone network. If the confidence score of an early classifier exceeds a predefined threshold, the inference process is terminated immediately, and the result is returned. This mechanism drastically reduces both computation and communication for easily recognizable inputs. However, integrating early exits into an edge-cloud collaborative framework complicates the configuration space, as the system must now determine not only where to partition the main network but also how to calibrate the confidence thresholds for the early exits located at the edge.

2.3 Joint Optimization Approaches And Their Limitations

In recent years, the academic community has recognized that optimizing neural network partitioning in a vacuum is insufficient for robust visual Internet of Things operations. The configuration of the multimedia source itself exerts a profound impact on downstream processing. Several studies have attempted to integrate video encoding parameters into the offloading decision matrix. Researchers have proposed frameworks that dynamically adjust the video encoding bitrate and the edge computing resource allocation to maximize the quality of experience for real-time video analytics [12]. These studies typically employ Lyapunov optimization techniques to decouple the long-term temporal dependencies of queueing dynamics, transforming the problem into a series of independent deterministic optimization steps. While effective for continuous video streams, these approaches often rely on traditional video codecs like H.264, which are designed for human visual perception rather than machine vision. Compressing raw video for machine vision presents a different set of challenges. Lossy compression introduces artifacts that can severely degrade the accuracy of downstream deep learning models. To mitigate this, recent literature has shifted towards feature map compression. Instead of compressing the raw image, the system computes the early layers locally and then applies quantization or spatial pooling to the resulting high-dimensional tensors [13]. Quantization maps the 32-bit floating-point activations to lower-precision formats, such as 8-bit or even 4-bit integers. Despite these individual advancements, comprehensive joint optimization frameworks remain scarce. Existing literature generally suffers from three major limitations. First, most studies optimize a limited subset of parameters, such as only partitioning and bandwidth allocation, or only resolution and frame rate, ignoring the complex cross-layer interactions. Second, many proposed algorithms rely on simplistic, static models of the network environment, assuming constant channel capacities and deterministic server processing rates, which fail to capture the stochastic nature of real-world deployments [14]. Third, they frequently employ heuristic search algorithms that scale poorly as the configuration space expands, resulting in optimization times that are too slow for dynamic runtime adaptation [15]. This paper directly addresses these gaps by proposing a holistic, learning-based approach that simultaneously navigates the entire configuration space under dynamic environmental conditions.

3. System Architecture And Problem Formulation

3.1 Hierarchical Visual Iot Architecture

The proposed system architecture is modeled as a hierarchical, three-tier topology consisting of end devices, multi-access edge computing nodes, and a centralized cloud data center. This architecture is designed to facilitate dynamic, distributed inference for a vast array of concurrent visual streams. The first tier comprises a multitude of visually enabled end devices, such as wireless security cameras, autonomous drones, and wearable augmented reality headsets. These devices are equipped with integrated image sensors, localized processing units (such as low-power neural processing units or microcontrollers), and wireless communication modules. Their primary function is to capture environmental data, perform preliminary data processing, and execute the initial layers of the deep neural network if a local partition point is selected. The second tier consists of edge servers strategically deployed at the network periphery. These servers are typically co-located with cellular base stations, wireless access points, or specialized local gateways. Edge servers possess moderate computational capabilities, bridging the gap between the severely constrained end devices and the massive cloud infrastructure. In our collaborative inference model, the edge servers receive intermediate feature maps or compressed video streams from the end

devices. Depending on the system configuration, the edge server may execute the intermediate layers of the neural network and forward deeper feature maps to the cloud, or it may complete the inference process entirely if the required latency is exceedingly strict and the model complexity permits. The third tier is the centralized cloud server, representing a practically infinite pool of computational and storage resources. The cloud server hosts the complete, unpartitioned deep learning models and handles the heaviest computational workloads, particularly the complex fully connected layers and sophisticated semantic reasoning modules. The communication links between the tiers are heterogeneous. The connection between the end devices and the edge servers relies on wireless access networks, such as 5G cellular links or Wi-Fi 6, which are highly susceptible to interference, fading, and bandwidth fluctuations. The backhaul connection between the edge servers and the cloud data center typically utilizes high-capacity wired fiber-optic links. While the bandwidth of the backhaul is generally large, the geographical distance introduces an unavoidable propagation delay that must be accounted for in the system latency model.

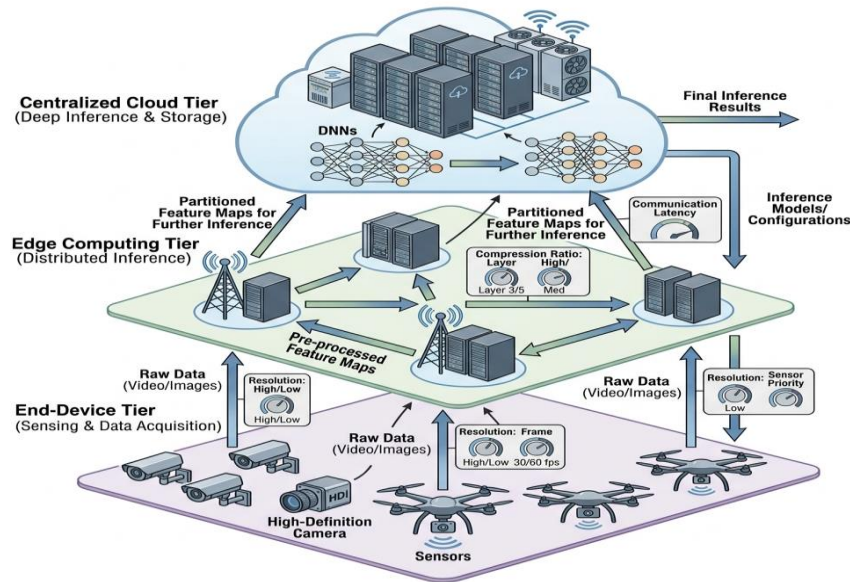


Figure 1 Comprehensive Schematic of Hierarchical Edge

3.2 System Dynamics And Parameter Definitions

The operational dynamics of the proposed system are governed by a continuous cycle of data capture, configuration selection, local processing, transmission, and remote execution. To formalize the problem, we define a set of critical configuration variables that the central orchestrator or the distributed agents must determine for each time interval. The first configuration variable is the spatial resolution of the captured visual frame. The system can select from a discrete set of predefined resolutions. Higher resolutions preserve fine-grained spatial details, which are imperative for recognizing distant or small objects, but they exponentially increase the number of pixels to be processed by the initial convolutional layers. The second configuration variable is the frame rate, defining the temporal resolution of the visual stream. The frame rate determines the total number of inference requests generated per second. A higher frame rate ensures smooth tracking of fast-moving objects but significantly increases the workload on both the local processors and the transmission queues. The third configuration variable is the neural network partition strategy. The targeted deep neural network is modeled as a sequence of discrete computational blocks. The partition variable determines exactly after which specific block the local computation terminates. The feature maps produced by this terminal block are then prepared

for transmission. If the partition variable is set to zero, no local processing occurs, and the raw image is transmitted. If the variable is set to the final layer, the entire inference process is executed locally. The fourth configuration variable relates to intermediate data compression. Specifically, we define a quantization parameter that dictates the numerical precision of the transmitted feature maps. By reducing the precision from standard floating-point to lower bit-width integers, the system can proportionally reduce the transmission payload size. The performance of the system under any given configuration is measured across three primary dimensions: end-to-end latency, total energy consumption, and inference accuracy. The end-to-end latency is an accumulation of multiple distinct delay components. It begins with the local processing latency on the end device, which is a function of the selected resolution and the partition point. This is followed by the wireless transmission latency, determined by the size of the quantized feature maps, the selected frame rate, and the current wireless channel capacity. Upon reaching the edge server, the data experiences queueing latency, dependent on the current workload of the edge node, followed by the edge processing latency. A similar sequence of transmission, queueing, and processing delays occurs if further offloading to the cloud is required. Energy consumption is predominantly incurred at the resource-constrained end devices. The energy model incorporates the baseline power consumption of keeping the sensor active, the computational energy spent executing the local layers of the neural network, and the communication energy consumed by the wireless transceiver when transmitting the intermediate data. The inference accuracy is a complex, non-linear function of the chosen resolution, the quantization level, and potentially the frame rate (if temporal information is critical to the specific model). Estimating this accuracy requires extensive offline profiling of the deep neural network under various combinations of input degradation and feature map compression.

3.3 The Optimization Objective

The fundamental goal of the proposed framework is to discover an optimal configuration policy that dynamically maps the current state of the environment to the best possible set of configuration variables. The objective function is constructed as a weighted sum of the system-wide end-to-end latency and the energy consumption of the end devices. These two metrics often exhibit conflicting behaviors. For example, processing more layers locally might save transmission energy but significantly increase computational latency due to the weak processors on the end devices. The optimization is heavily constrained. Most importantly, the system must guarantee a minimum threshold of inference accuracy. Configurations that drop the resolution too low or apply excessively aggressive quantization to meet latency deadlines are strictly penalized or rendered invalid if the resulting accuracy falls below the application's required baseline. Furthermore, the system must respect the queue stability constraints at the edge servers; configurations that consistently generate data faster than the edge can process it will lead to buffer overflows and packet loss, destabilizing the entire visual inference pipeline. The complexity of this joint optimization problem, characterized by discrete decision variables, non-linear system dynamics, and coupled constraints, renders traditional convex optimization techniques entirely inapplicable.

4. Optimization Methodology And Algorithm Design

4.1 State Space And Action Space Representation

Given the NP-hard nature of the formulated joint configuration optimization problem and the stochastic behavior of the underlying communication networks, we cast the problem as a Markov Decision Process. This formulation allows us to leverage the powerful function approximation capabilities of deep reinforcement learning to derive near-optimal policies. The state space must

comprehensively capture the operating environment at the beginning of each decision epoch. We define the state vector to include multiple critical observations. First, it includes the estimated wireless channel bandwidth for each end device, which fluctuates due to user mobility and environmental interference. Second, it incorporates the current queue lengths at both the local communication buffers on the end devices and the task processing queues at the edge servers. These queue lengths serve as vital indicators of network congestion and computational bottlenecks. Third, the state vector includes the remaining battery capacity of the end devices, allowing the optimization algorithm to adapt its energy conservation strategies based on the urgency of power preservation. The action space corresponds directly to the configuration variables defined in the system architecture. For each active visual stream, the learning agent must output a multidimensional action vector encompassing the chosen resolution, the frame rate, the network partition point, and the quantization level. Because these variables are inherently discrete, standard continuous-action reinforcement learning algorithms cannot be applied directly without significant modification. To accurately evaluate the consequences of the selected actions, we define a highly tailored reward function. The reward is inversely proportional to the objective cost function described previously. Specifically, the system receives a positive baseline reward for successfully completing an inference task. From this baseline, we subtract penalty terms corresponding to the measured end-to-end latency and the energy consumed during the cycle. Crucially, the reward function incorporates a severe step-function penalty if the expected accuracy of the chosen configuration falls below the mandatory threshold. This harsh penalty mechanism strongly discourages the reinforcement learning agent from exploring regions of the action space that violate the fundamental quality of service requirements.

Table Caption MUST be immediately above the table.

Table 1 Configuration Parameters and Action Space Definitions

Parameter Category	Variable Name	Discrete Values Available	Description
Application	Image Resolution	224x224, 448x448, 896x896	Spatial size of the visual input fed into the model.
Application	Frame Rate	5, 15, 30	Number of inference requests generated per second.
Computing	Partition Point	Block 1 to Block 12	The neural network layer after which offloading occurs.
Networking	Quantization	8-bit, 16-bit, 32-bit	Numerical precision applied to intermediate feature maps.

4.2 Proposed Deep Reinforcement Learning Framework

To navigate this complex, multi-dimensional discrete action space, we design a customized Deep Q-Network variant enhanced with multi-agent coordination capabilities. In a large-scale visual Internet of Things deployment, relying on a single centralized agent to determine the configurations for thousands of devices simultaneously leads to an unmanageable explosion in the action space dimensionality. Therefore, we adopt a decentralized execution paradigm where each edge server acts as a semi-autonomous learning agent, managing the configuration of the end

devices within its immediate coverage area. The core architecture of our learning agent relies on deep neural networks to approximate the optimal action-value function. The input to the neural network is the current environmental state vector. The network is composed of multiple fully connected hidden layers, utilizing rectified linear unit activation functions to capture the intricate non-linear relationships between network conditions and configuration performance. The output layer of the network provides the estimated expected cumulative reward for every possible combination of discrete configuration actions. To ensure stable and efficient learning, our algorithm incorporates several advanced techniques from the reinforcement learning literature. We utilize experience replay, where the agent stores its state transitions, actions, rewards, and subsequent states in a large memory buffer. During the training phase, the agent samples random mini-batches from this buffer to update its neural network weights. This process breaks the temporal correlation between consecutive experiences and significantly reduces the variance of the gradient updates. Furthermore, we employ a target network architecture. Instead of constantly updating the neural network used to calculate the target Q-values, we maintain a separate copy of the network that is updated only periodically. This stabilization technique prevents the learning process from diverging or oscillating uncontrollably, which is a common issue in deep reinforcement learning when the target values shift constantly. During the operational phase, the system follows an epsilon-greedy exploration strategy. Most of the time, the agent exploits its learned knowledge by selecting the configuration combination that yields the highest predicted action-value. However, with a small probability, the agent selects a completely random configuration. This continuous exploration ensures that the system can adapt to sudden, unforeseen changes in the environment, such as the sudden deployment of new edge infrastructure or severe, long-lasting degradation in wireless channel quality. The training process is conducted continuously in the background, allowing the policy to evolve and refine itself over the lifetime of the visual Internet of Things system.

5. Experimental Results And Discussion

5.1 Simulation Setup And Evaluation Methodology

To rigorously evaluate the performance of the proposed joint configuration optimization framework, we developed a comprehensive simulation environment integrating network dynamics and actual deep learning execution profiles. The network topology is simulated using a custom discrete-event simulator that models wireless channel fading, bandwidth contention, and packet-level queueing dynamics. For the visual inference workload, we selected a widely adopted object detection model, YOLOv4-tiny, which is heavily utilized in real-time visual Internet of Things applications due to its balance of speed and accuracy. The model was trained and profiled on the standard COCO dataset. To obtain realistic latency and energy consumption measurements, we profiled the execution of every individual layer of the model across different hardware profiles. The end device capabilities were modeled after a Raspberry Pi 4 equipped with an edge tensor processing unit. The edge server was modeled as an industrial-grade workstation with a high-performance graphics processing unit, while the cloud tier was represented by server-grade hardware with virtually negligible processing latency compared to the communication overhead. We conducted offline profiling to construct lookup tables mapping every combination of image resolution, frame rate, and feature map quantization to its corresponding baseline inference accuracy. This empirical profiling guarantees that our simulation accurately reflects the non-linear degradation of machine vision performance under aggressive resource constraints. To provide a robust comparative analysis, we benchmarked our proposed deep reinforcement learning joint

optimization approach against four distinct baseline strategies commonly found in the literature. The first baseline is a purely Local Execution strategy, where the entire model runs on the end device, minimizing communication but maximizing computational delay. The second baseline is a Cloud-Only Offloading strategy, where raw compressed frames are transmitted to the cloud, minimizing local computation but incurring massive transmission latency. The third baseline is a Static Partitioning method, which utilizes a fixed resolution and fixed neural network split point that performed best on average during pre-deployment testing. The fourth baseline is an Independent Optimization heuristic that sequentially optimizes the partition point, then the resolution, and finally the quantization, without considering their joint interactions.

5.2 Performance Evaluation And Comprehensive Analysis

The primary metric for our evaluation is the ability of the system to maintain low end-to-end latency while adhering strictly to energy budgets and accuracy constraints. We subjected the simulated system to varying network conditions, scaling the average available wireless bandwidth from severely constrained environments to high-capacity scenarios.

Table Caption MUST be immediately above the table.

Table 2 Comparative Performance Metrics under Average Network Load

Optimization Strategy	Avg. Latency (ms)	Avg. Energy per Frame (mJ)	Accuracy Violation Rate (%)
Local Execution	450.2	85.4	0.0
Cloud-Only Offloading	310.8	32.1	0.0
Static Partitioning	245.5	45.6	12.4
Independent Optimization	185.3	41.2	5.8
Proposed Joint Optimization	124.7	38.5	0.2

The results presented in the accompanying data clearly demonstrate the superiority of the proposed joint optimization framework. Under average network loads, our approach achieved a remarkable reduction in average end-to-end latency. Compared to the Cloud-Only baseline, the latency was reduced by more than half. This is primarily because the deep reinforcement learning agent intelligently learned to partition the neural network at optimal points, executing the initial feature extraction locally and significantly reducing the volume of data transmitted over the wireless bottleneck. When comparing the energy consumption, the Proposed Joint Optimization strategy achieved a finely tuned balance. While the Cloud-Only approach inherently minimizes local energy consumption by avoiding computation, it completely fails to meet strict latency deadlines. Conversely, Local Execution drained the end device batteries rapidly. Our joint approach dynamically adjusted the resolution and quantization levels, allowing it to maintain a low energy footprint while still leveraging the edge servers for heavy computation. The most critical advantage of the proposed method is observed in the accuracy violation rate. The Static Partitioning baseline suffered from a high rate of accuracy violations. When the network bandwidth suddenly dropped, the static method was forced to drop frames or buffer them indefinitely, failing the real-time requirements. The Independent Optimization heuristic performed better but occasionally fell into local optima where an aggressive quantization choice to save bandwidth unexpectedly pushed the inference accuracy below the required threshold. In contrast, our deep reinforcement learning agent maintained an accuracy violation rate of nearly zero. By exploring the entire joint

configuration space during training, the agent implicitly learned the complex multidimensional boundaries of the accuracy constraints, ensuring that it only selected configurations that were guaranteed to meet the application requirements.

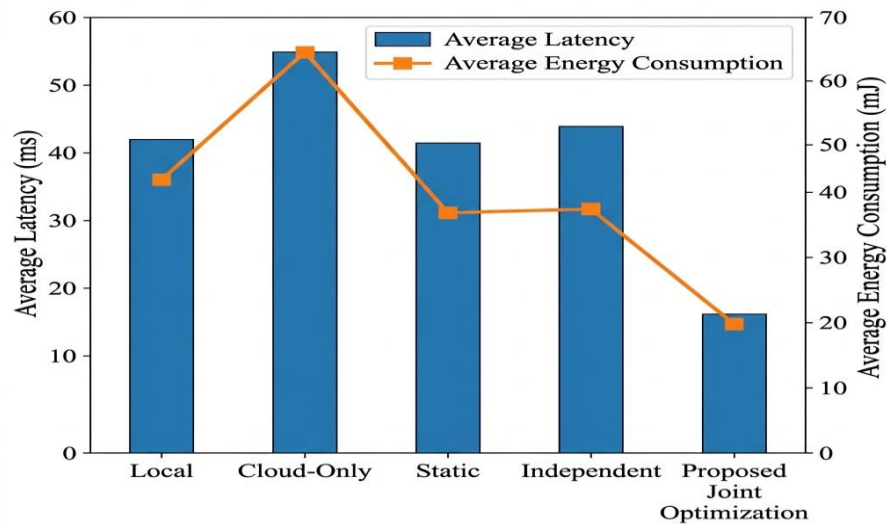


Figure 2 End

Further analysis under highly dynamic network conditions revealed the robustness of the reinforcement learning approach. When simulated network congestion was introduced, temporarily slashing the available bandwidth by seventy percent, the system immediately adapted. The agent autonomously shifted its policy from using high resolutions and late partition points to utilizing lower resolutions, highly quantized feature maps, and earlier partition points, effectively migrating the computational burden back to the local device to avoid the collapsed communication link. Once the congestion cleared, the system seamlessly reverted to utilizing the edge resources, demonstrating its capacity for continuous, real-time adaptation in unpredictable visual Internet of Things environments.

6. Conclusion And Future Directions

6.1 Summary Of Contributions

This paper has presented a comprehensive and highly adaptable framework for the joint configuration optimization of edge-cloud collaborative inference within visual Internet of Things systems. Recognizing the profound limitations of isolated parameter tuning, we developed a multidimensional optimization model that simultaneously orchestrates camera frame rates, image capture resolutions, dynamic neural network partition points, and intermediate data quantization levels. By framing this complex coordination challenge as a Markov Decision Process, we successfully applied advanced deep reinforcement learning techniques to navigate the discrete, non-convex action space. Through extensive simulation incorporating realistic deep learning execution profiles and stochastic network dynamics, we demonstrated that our proposed joint optimization strategy significantly outperforms existing baseline methods. The framework successfully minimized end-to-end inference latency and constrained local energy consumption without compromising the strict accuracy baselines required by critical computer vision applications. The ability of the decentralized learning agents to autonomously adapt to fluctuating wireless channel capacities and edge server workloads proves the viability of this approach for large-scale, real-world deployment in smart cities and automated industrial environments.

6.2 Limitations And Future Work

Despite the significant performance improvements achieved, the current framework presents certain limitations that warrant further investigation. Primarily, the current model assumes a relatively static physical topology where end devices remain connected to the same edge server for the duration of the decision epoch. In highly mobile scenarios, such as vehicle-to-infrastructure visual networks, the rapid handover between different edge nodes introduces severe disruptions to the collaborative inference pipeline. Future research must expand the state space to account for user mobility and incorporate predictive hand-over management into the joint optimization decision matrix. Additionally, the reliance on continuous, centralized training data collection raises potential privacy concerns, even when raw video is not transmitted. The metadata regarding device usage patterns and configuration states could theoretically be exploited. Therefore, integrating privacy-preserving machine learning paradigms, specifically federated learning, into the training of the reinforcement learning agents represents a critical next step. By keeping the replay buffers and raw experiences strictly localized on the edge servers and only aggregating model gradients in the cloud, future iterations of this system can achieve the same level of intelligent configuration optimization while fundamentally guaranteeing data privacy across the visual Internet of Things ecosystem.

References

- Yang, Z., Liang, B., & Ji, W. (2021). An intelligent end-edge-cloud architecture for visual IoT-assisted healthcare systems. *IEEE internet of things journal*, 8(23), 16779-16786.
- Yang, W., Qin, Y., & Yang, Z. (2022). A reinforcement learning based data storage and traffic management in information-centric data center networks. *Mobile Networks and Applications*, 27(1), 266-275.
- Wang, S., Zhang, X., Wang, P., & Li, X. (2026). Design of Magnetic Sensor Array-Based PUFs for IoT Security. *IEEE Transactions on Instrumentation and Measurement*, 75, 1-9.
- Yao, Y., Lu, Z., Chen, G., Huang, C., Feng, C., & Quek, T. Q. (2026). Interference Management in ISAC-SAGINs Based on Transformer-Enabled Mean-Field Reinforcement Learning Method. *IEEE Transactions on Wireless Communications*.
- Chen, W., Li, X., Liu, X., & Yin, L. (2015). Study of self-calibrating MEMS accelerometers. *AIP Advances*, 5(4).
- Zhang, W., Li, M., & Chen, X. (2026). Distributed Edge-Fog-Cloud Architecture for Intelligent Fault Diagnosis of Permanent Magnet Synchronous Motors Using Temporal Fusion Transformers and Federated Learning. *Fundamental Scientific Reports in Multidisciplinary Areas*, 2 (02), 121-131.
- Li, X., Chen, W., Liu, X., & Ren, M. (2015). A high-order sigma-delta accelerometer interface circuit. *IEICE Electronics Express*, 12(2), 20141035-20141035.
- Xia, L., Yang, M., Wang, J., Yan, Z., Ren, Y., Yu, G., & Lei, K. (2025, September). Mamba4Net: Distilled Hybrid Mamba Large Language Models For Networking. In *2025 IEEE 33rd International Conference on Network Protocols (ICNP)* (pp. 1-10). IEEE.
- Yang, W., Qin, Y., Jiang, Z., & Chu, X. (2021, June). Traffic Management for Distributed Machine Learning in RDMA-enabled Data Center Networks. In *ICC 2021-IEEE International Conference on Communications* (pp. 1-6). IEEE.
- Yang, Z., Ji, W., & Wang, Z. (2024). Adaptive joint configuration optimization for collaborative inference in edge-cloud systems. *Science China Information Sciences*, 67(4), 149103.
- Li, X., Yin, L., Chen, W., Gao, Z., & Liu, X. (2017). A high-resolution tunneling magneto-resistance sensor interface circuit. *Modern Physics Letters B*, 31(04), 1750030.

- Zhao, Q., & Huang, Y. (2026). DAREA: A Dynamic AI-Driven Architecture for Real Estate Asset Tokenization on Blockchain. *Fundamental Scientific Reports in Multidisciplinary Areas*, 2(02), 182-192.
- Yang, Z., Ji, W., Guo, Q., & Wang, Z. (2023, October). Javp: Joint-aware video processing with edge-cloud collaboration for dnn inference. In *Proceedings of the 31st ACM International Conference on Multimedia* (pp. 9152-9160).
- Yang, Y., Ma, X., Li, C., Zheng, Z., Zhang, Q., Huang, G., ... & Zhao, Q. (2021). Believe what you see: Implicit constraint approach for offline multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 34, 10299-10312.
- Li, X., Wang, P., Li, G., Ni, L., & Zhang, Y. (2023). Design of interface circuits and lightweight PUF for TMR sensors. *IEEE Sensors Journal*, 23(11), 11754-11761.
- Yao, Y., Zhu, Z., Miao, P., Cheng, X., Shu, F., & Wang, J. (2025). Optimizing hybrid RIS-aided ISAC systems in V2X networks: A deep reinforcement learning method for anti-eavesdropping techniques. *IEEE Transactions on Vehicular Technology*, 74 (6), 9224-9239.
- Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., McCauley, M., et al. (2010). Spark: Cluster computing with working sets. In *Proceedings of HotCloud*.
- Yao, Y., Sun, J., Miao, P., Chen, G., & Tafazolli, R. (2026). Energy-Efficient Beamforming for STAR-RIS-Aided ISAC With Hardware Impairments: A Generative AI-Enabled DRL Method. *IEEE Transactions on Wireless Communications*.
- Chen, L., Ge, L., Jiang, X., Jiang, Z., & Li, H. (2026). Tensor-Based Target Sensing for Resource-Irregular ISAC Systems. *IEEE Transactions on Signal Processing*.
- Wan, H., Zhang, Y., Wang, J., Wu, D., Li, M., Chen, X., ... & Ji, X. (2025). Toward universal embodied planning in scalable heterogeneous field robots collaboration and control. *Journal of Field Robotics*, 42(5), 2318-2336.
- Xu, C., Wang, C., & Li, Z. (2026). H2-LBM: A Hierarchical Hybrid Deep Reinforcement Learning Framework for L7 Load Balancing and Global Traffic Scheduling in Multi-Cloud LLM Serving. *Fundamental Scientific Reports in Multidisciplinary Areas*, 2 (02), 109-120.
- Hu, Y., Yang, Z., Zhao, C., & Ji, W. (2026, May). Adaptive guidance semantically enhanced via multimodal llm for edge-cloud object detection. In *ICASSP 2026-2026 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 12962-12966). IEEE.