New Problems in Distributed Inference for DNN Models on Robotic IoT

(Anonymous Authors)

***Abstract*—The rapid advancements in machine learning (ML) techniques have led to significant achievements in various robotic tasks. Deploying these ML approaches on real-world robots requires fast and energy-efficient inference of their deep neural network (DNN) models. To our knowledge, distributed inference, which offloads a portion of the inference computation onto powerful GPU devices, has emerged as a promising optimization to improve inference performance in modern data centers. However, when deployed on real-world robots, existing parallel methods can not simultaneously meet the latency and energy requirements of robotic and raise significant challenges.**

This paper reveals and evaluates the problems hindering the application of these parallel methods in robotic IoT, including the in-applicability of data parallelism due to small batch sizes, the unacceptable communication overhead of tensor parallelism caused by frequent synchronization, and the transmission bottlenecks in pipeline parallelism that cannot be effectively mitigated through scheduling. By raising awareness of these new problems, we aim to stimulate research efforts toward finding a new parallel method that can accelerate distributed inference in robotic IoT.

I. Introduction

The rapid progress in machine learning (ML) techniques has led to remarkable achievements in various fundamental robotic tasks, such as object detection[14], [20], [22], robotic control[16], [31], [36], and environmental perception[4], [17], [34]. However, deploying these ML approaches on real-world robots requires fast and energy-efficient inference of their deep neural network (DNN) models, given the need for swift environmental responses and limited battery capacity of robots. Relying solely on additional computing accelerators on robots (e.g., GPU[24], FPGA[25], SoC[12]) increases energy consumption (e.g., 62% for [22] in our experiments) due to the computationally-intensive nature of DNN models, while placing the entire model to the cloud brings a long response delay.

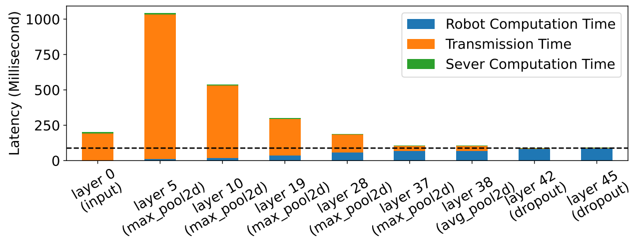
Distributed inference[**?**], which involves offloading a portion of the inference computation onto other powerful GPU devices, has emerged as a promising approach to meet the latency requirements of robotic applications and extend the battery lifetime of robots. By leveraging the high computing capabilities of powerful GPUs through Internet of Things for these robots (robotic IoT), distributed inference accelerates the inference process while alleviating the local computational burden, thereby reducing energy consumption. This paradigm has been widely adopted in data centers[**?**], where numerous GPUs are utilized for large model inference, such as in the case of ChatGPT[33].

However, all existing parallel methods for distributed inference in data center are ill-suited in robotic IoT. In data center, there are mainly three kinds of parallel methods: Data parallelism (DP) replicates the model across devices, and lets each replica handle one micro-batch (i.e., splits of a batch which is a set of training inputs for each device); Tensor parallelism (TP) splits a single DNN layer over devices; Pipeline parallelism (PP) places different layers of a DNN model over devices (layer partition) and pipelines the inference to reduce devices' idling time (pipeline execution). In this paper, we show that several problems are hindering existing parallel methods applying on robotic IoT.

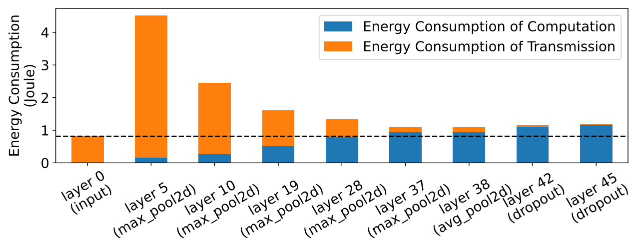
Problem 1 (DP). The small batch sizes inherent to robotic IoT applications (typically 1, 3, etc.) hinder the computation of micro-batches, rendering DP inapplicable for robotic IoT. In data centers, DP is feasible due to the large batch sizes employed (e.g., 16 or 32 images), allowing for the division of inputs into micro-batches that still contain several complete inputs (e.g., 2 or 4 images). However, in the context of robotic IoT, real-time performance is crucial, necessitating immediate inference upon receiving inputs, which typically have smaller batch sizes (e.g., 1 or 3 images). Further splitting these inputs would result in micro-batches containing incomplete inputs (e.g., 1/4 or 1/2 of an image), which cannot be computed in parallel to speed up inference.

Problem 2 (TP). TP requires frequent synchronization among devices, leading to unacceptable communication overhead in robotic IoT. By partitioning parameter tensors of a layer across GPUs, TP allows for concurrent computation on different parts of this tensor but requires an all-reduce synchronization (i.e., TP.sync[**?**]) to combine computation results from different devices, which entails significant communication overhead. Consequently, TP is used mainly for large layers that are too large to fit in one device in data centers and require dedicated high-speed interconnects (e.g., 400 Gbps for NVLink[15]) even within data centers. On the contrary, robots must prioritize seamless mobility and primarily depend on wireless connections, which inherently possess limited bandwidth, as described in Sec.II-A, making all-reduce synchronization an unacceptable overhead (e.g., commonly hundreds of milliseconds for each layer of VGG19[29] in our experiments).

Consequently, existing distributed inference approaches[**?**] in robotic IoT primarily adopt the PP paradigm and focus on layer partition of PP to achieve fast and energy-efficient infer-



(a) Inference Latency



(b) Energy consumption

Fig. 1: Our experiments on VGG19[29] reveal the comprehensive performance of various layer partition methods. The X-axis of the graph represents different layer partition scheduling schemes, where 'layer i' signifies that all layers up to and including the i-th layer are computed on the robot, while the subsequent layers are processed on the server. Notice that different hardware conditions, network conditions and DNN model structure will lead to different performance, making this field an attractive area for a wide range of research.

ence. This is because the PP paradigm in data centers consists of layer partition and pipeline execution, where the pipeline execution of PP enhances inference throughput rather than reducing the completion time of a single inference[**?**], which is the most critical requirement in robotic IoT. Based on the fact that the amounts of output data in some intermediate layers of a DNN model are significantly smaller than that of its raw input data[13], DNN layer partition methods constitute various trade-offs between computation and transmission, taking into account application-specific inference speed requirements and energy consumption demands, as shown in Fig.1.

Problem 3 (PP). Existing methods based on PP face significant challenges due to transmission bottlenecks in robotic IoT, which cannot be effectively mitigated through PP scheduling. Firstly, PP is unable to overlap the transmission and computation phases within the same inference, as it can only overlap these phases across multiple inferences via pipeline execution, which increases throughput but not the speed of a single inference[**?**]. Moreover, due to the limited bandwidth of robotic IoT, the transmission phase can be a substantial bottleneck (e.g., up to 70% of inference time and 50% of energy consumption in our experiments). Secondly, adjusting the scheduling scheme to reduce the traffic volume is not an ideal solution for robotic IoT, as allocating more DNN model layers to robots, while reducing the transmission volume and associated transmission time, increases energy consumption and under-utilizes the powerful GPU devices for accelerating inference, and may not necessarily reduce overall inference time.

In this paper, we take the first step to reveal and evaluate the problems hindering existing parallel methods for distributed inference applying on robotic IoT. These findings aim to raise research effort to find a new parallel method to speed up distributed inference on robotic IoT, so that the DNN models deployed on real-world robots can achieve fast and energy-efficient inference, and it will nurture diverse ML applications deployed on mobile robots in the field.

The rest of the paper is organized as follows: the chapter II provides background; the chapter III describes in detail about the problems; the chapter IV; provides with evaluation results; the chapter V concludes.

II. Background

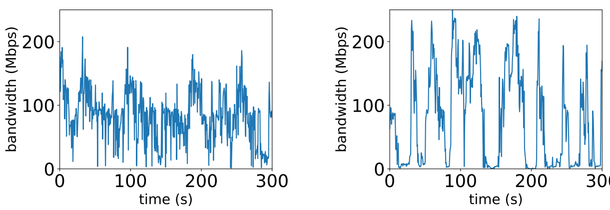
A. Characteristics of Robotic IoT Networks

Wireless transmission of Robotic IoT. In real-world robotic IoT scenarios, devices often need to navigate and move around for tasks such as search and exploration. While wireless networks provide the required high mobility, they also come with limited bandwidth. For instance, Wi-Fi 6, the most advanced Wi-Fi technology, offers a maximum theoretical bandwidth of 1.2 Gbps for a single stream[19]. Furthermore, not only the limited hardware resources on the robot can not fully play the potential of Wi-Fi 6[35], but also the actual available bandwidth of wireless networks is often reduced in practice due to factors such as movement of the devices[21], [26], occlusion from by physical barriers[6], [28], and preemption of the wireless channel by other devices[2], [27].

To demonstrate the instability of wireless transmission in real-world situations, we conducted a robot surveillance experiment using four-wheel robots navigating around several given points at 5~40cm/s speed in our lab (indoors) and campus garden (outdoors). The hardware and wireless network settings are as described in Sec. IV. We believe our setup represents the state-of-the-art (SOTA) computation and communication capabilities of robotic IoT devices. We saturated the wireless network connection with iperf [1] and recorded the average bandwidth capacity between these robots every 0.1s for 5 minutes.

The results, shown in Fig.2, indicate only 93 Mbps and 73 Mbps average bandwidth capacity for indoor and outdoor scenarios. The outdoor environment exhibited higher instability, with bandwidth frequently dropping to extremely low values around 0 Mbps. This can be attributed to the lack of walls in open outdoor areas to reflect wireless signals, and the presence of obstacles, such as trees, between communicating robots. Consequently, fewer signals can be received in outdoor areas compared to indoor environments.

In summary, the wireless transmission of robotic IoT systems is constrained by limited bandwidth, both due to the



(a) Indoors

(b) Outdoors

Fig. 2: The instability of wireless transmission in robotic IoT networks.

theoretical upper limit of wireless transmission technologies and the practical instability of wireless networks.

Comparison with Datacenter Networks. Compared to robotic IoT networks, data center networks (used for large model inference, e.g., ChatGPT[33]) are wired and typically exhibit higher bandwidth capacity and lower bandwidth fluctuation. In data center networks, GPU devices are connected using high-speed networking technologies, such as InfiniBand[30] or PCIe[15], which offer bandwidths ranging from 40 Gbps to 500 Gbps. Bandwidth fluctuation in these networks is primarily caused by congestion on intermediate switches and can be mitigated through traffic scheduling on switches[23].

B. Other methods on robotic IoT

Compressed communication. Compressed communication, which is essential for practical distributed inference over wireless networks, significantly reduces communication traffic volume through techniques such as quantization[5], [9], [10] and model distillation[11], [18], [32]. Quantization reduces the numerical precision of model weights and activations, thereby minimizing the memory footprint and computational requirements of deep learning models. This process typically involves converting high-precision (e.g., 32-bit) floating-point values to lower-precision (e.g., 8-bit) fixed-point representations, with minimal loss of model accuracy. Model distillation[11], [18], [32], on the other hand, involves training a smaller, more efficient "student" model" to mimic the behavior of a larger, more accurate "teacher" model by minimizing the difference between the student model's output and the teacher model's output. The distilled student model retains much of the teacher model's accuracy while requiring significantly fewer resources. As they achieve faster inference speed by modifying the model and potentially sacrificing accuracy, these model compression methods are orthogonal to distributed inference and complement distributed inference, which realizes fast inference without loss of accuracy by intelligently scheduling computation tasks across multiple devices.

Job scheduling. Significant research efforts have been devoted to exploring inference parallelism and unleashing the potential of layer partition to accelerate DNN inference, such as job scheduling, which determines the scheduling mode of different inference tasks based on their delay and energy consumption requirements, as well as the load condition of various devices. For instance, [3], [7] support online scheduling of offloading inference tasks based on the current network and resource status of mobile systems while meeting user-defined energy constraints to extend battery life and enhance the capabilities of mobile device. [8] focused on optimizing DNN inference workloads in cloud computing using a deep reinforcement learning based scheduler for QoS-aware scheduling of heterogeneous servers, aiming to maximize inference accuracy and minimize response delay. While these methods aim at overall optimization in multi-task scenarios involving multi-robots at edge side, they do not address the optimization of single inference tasks and are thus orthogonal to distributed inference for a single inference, where improved distributed inference can provide faster and more energy-efficient inference for them.

III. Problems in Parallel Inference Strategies for DNN Models on Robotic IoT

A. Existing parallel inference strategies in data center

Data parallelism. Data parallelism in distributed inference is an approach that divides the input data into smaller subsets (micro-batch) and processes them concurrently across multiple devices, thus leveraging their combined computational power. This parallel processing of data samples allows for faster inference speeds, . When data parallelism introduces increased throughput, efficient resource utilization, and the ability to handle large-scale data sets, it also introduces challenges such as communication overhead for synchronizing and potential difficulties in maintaining effectiveness across devices with various available computational resources. Its scalability [8] is bounded by communication on low-end networks and the size of a total batch (a set of data for producing each parameter update).

Tensor parallelism. Tensor parallelism in distributed inference is an approach that partitions input and parameter tensors of a layer (e.g., transformer multi-head self-attention layer [30]) across GPUs, allowing for concurrent computation on different parts of this tensor. TP requires an all-reduce communication (i.e., TP.sync) to aggregate the tensors between repeated blocks, which typically lies on the critical path and is network-intense. Therefore, TP is usually deployed across GPUs within the same server to use fast intra-server GPU-to-GPU links (e.g., NVLink [42]). TP is mainly adopted as a complementary technique to help existing parallel training systems [8], [19], [25], [41] to support larger transformer layers.

Pipeline parallelism.

B. Existing distributed inference on robotic IoT

1) Layer partition: Hu et al. [23] designed a dynamic DNN surgery strategy to partition DNN inference between the cloud and edge at the granularity of the DNN layers. This strategy reduced the system latency and improved throughput by limiting data transmission, but it paid less attention to the offloading problem for DNN layers. Mohammed et al. [19] proposed an adaptive DNN partition scheme and a distributed algorithm based on the matching game method, where the DNN layers were offloaded to fog nodes. Neurosurgeon [4] claimed that excessive latency and energy consumption were generated when uploading massive data of DNNs to the cloud via the wireless network. To cope with this problem, a lightweight scheduler was designed to partition DNN-based applications automatically between end devices and the cloud at the granularity of DNN layers.

2) Energy consumption: Most of the aforementioned work tried to reduce the system latency in cloud/edge environments [16], [18], [23], but they did not consider reducing the system energy consumption of offloading DNN layers with deadline constraints. An application partitioning algorithm was presented in [20] for pursuing a trade-off between energy consumption and data transmission in dynamic mobile environments. Teerapittayanon et al. [21] proposed distributed DNNs over the cloud, edge, and IoT devices. They considered the data transmission cost but not the layer execution consumption, while deploying distributed DNNs in the cloud-edge environments. In the previous work [6], a cost-driven offloading scheme was designed for DNN-based smart IoT systems with deadline constraints over the cloud, edge, and IoT devices, where a discrete PSO algorithm was developed to reduce the system cost of executing DNN layers and transferring data. Different from this work, we further consider the energy consumption of each participating server and IoT device, and introduce the layer partition operations into the offloading decision-making process for DNN layers.

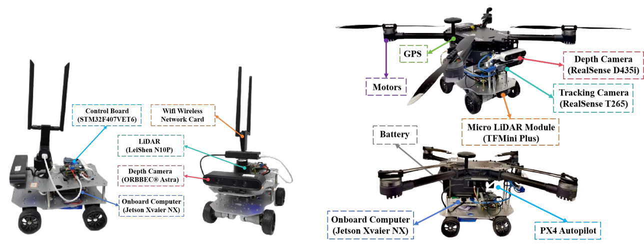
In general, most of these work focused on the offloading problem in distributed inference. However, it is still an open issue to optimize the system energy consumption when offloading DNN layers with deadline constraints in robotic IoT.

C. Dilemma on Inference Time

D. Dilemma on Energy Consumption

IV. Evaluation

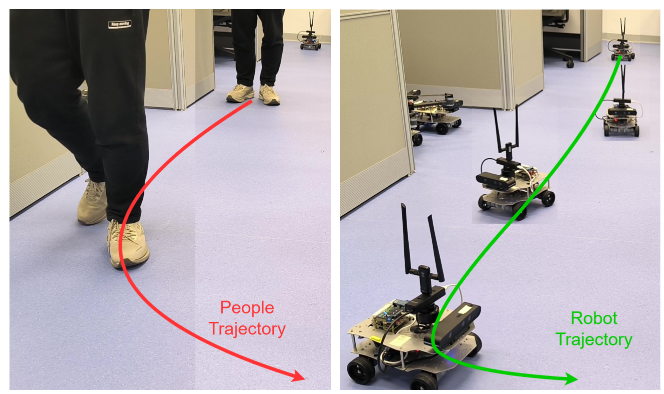
Testbed. The evaluation was conducted on a custom four-wheeled robot (Fig 3a), and a custom air-ground robot(Fig 3b). They are equipped with a Jetson Xavier NX[24] 8G onboard computer serving as the ROS master. The system runs Ubuntu 18.04 and utilizes a SanDisk 256G memory card, with ROS2 Galactic installed for application development and a dual-band USB network card (MediaTek MT76x2U) for wireless connectivity. The Jetson Xavier NX interfaces with a Leishen N10P LiDAR, ORBBEC Astra depth camera, and an STM32F407VET6 controller via USB serial ports. Both LiDAR and depth cameras facilitate environmental perception, enabling autonomous navigation, obstacle avoidance, and SLAM mapping. The host computer processes environmental information in ROS2 Galactic, performing path planning, navigation, and obstacle avoidance before transmitting velocity and control data to corresponding ROS topics. The controller then subscribes to these topics, executing robot control tasks.



(a) Four-wheeled robot (b) Air-ground robot

Fig. 3: The detailed composition of the robot platforms

Real-world Robotic Applications. We evaluated two kinds of typical real-world applications on robots: a real-time people-tracking robotic application on our four-wheeled robot as depicted in Fig 4 and a autonomous navigation on our air-ground robot as depicted in Fig 5.



(a) Targeted people (b) Robot moving trajectory

Fig. 4: A real-time people-tracking robotic application on our robot based on a well-known human pose estimation ML model, KAPAO[22].

Fig. 5: By predicting occlusions in advance, AGRNav[31] gains an accurate perception of the environment and avoids collisions, resulting in efficient and energy-saving paths.

V. Conclusion

The conclusion goes here.

bandwidth upper bound.

which type.

Acknowledgment

The authors would like to thank...

References

[1] iPerf - Download iPerf3 and original iPerf pre-compiled binaries.

[2] Adame, T., Carrascosa-Zamacois, M., and Bellalta, B. Time-sensitive networking in ieee 802.11 be: On the way to low-latency wifi 7. *Sensors 21*, 15 (2021), 4954.

[3] Altamimi, M., Abdrabou, A., Naik, K., and Nayak, A. Energy cost models of smartphones for task offloading to the cloud. *IEEE Transactions on Emerging Topics in Computing 3*, 3 (2015), 384–398.

[4] Cao, A.-Q., and de Charette, R. Monoscene: Monocular 3d semantic scene completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022), pp. 3991–4001.

[5] Défossez, A., Adi, Y., and Synnaeve, G. Differentiable model compression via pseudo quantization noise. *arXiv preprint arXiv:2104.09987* (2021).

[6] Ding, M., Wang, P., López-Pérez, D., Mao, G., and Lin, Z. Performance impact of los and nlos transmissions in dense cellular networks. *IEEE Transactions on Wireless Communications 15*, 3 (2015), 2365–2380.

[7] Elgazzar, K., Martin, P., and Hassanein, H. S. Cloud-assisted computation offloading to support mobile services. *IEEE Transactions on Cloud Computing 4*, 3 (2014), 279–292.

[8] Fang, Z., Yu, T., Mengshoel, O. J., and Gupta, R. K. Qos-aware scheduling of heterogeneous servers for inference in deep neural networks. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management* (2017), pp. 2067–2070.

[9] Gheorghe, , and Ivanovici, M. Model-based weight quantization for convolutional neural network compression. In *2021 16th International Conference on Engineering of Modern Electric Systems (EMES)* (2021), IEEE, pp. 1–4.

[10] Gong, C., Chen, Y., Lu, Y., Li, T., Hao, C., and Chen, D. Vecq: Minimal loss dnn model compression with vectorized weight quantization. *IEEE Transactions on Computers 70*, 5 (2020), 696–710.

[11] Gou, J., Yu, B., Maybank, S. J., and Tao, D. Knowledge distillation: A survey. *International Journal of Computer Vision 129*, 6 (2021), 1789–1819.

[12] Honkote, V., Kurian, D., Muthukumar, S., Ghosh, D., Yada, S., Jain, K., Jackson, B., Klotchkov, I., Nimmagadda, M. R., Dattawadkar, S., et al. 2.4 a distributed autonomous and collaborative multi-robot system featuring a low-power robot soc in 22nm cmos for integrated battery-powered minibots. In *2019 IEEE International Solid-State Circuits Conference-(ISSCC)* (2019), IEEE, pp. 48–50.

[13] Hu, C., Bao, W., Wang, D., and Liu, F. Dynamic adaptive dnn surgery for inference acceleration on the edge. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications* (2019), IEEE, pp. 1423–1431.

[14] Joseph, K. J., Khan, S., Khan, F. S., and Balasubramanian, V. N. Towards open world object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2021), pp. 5830–5840.

[15] Li, A., Song, S. L., Chen, J., Li, J., Liu, X., Tallent, N. R., and Barker, K. J. Evaluating modern gpu interconnect: Pcie, nvlink, nv-sli, nvswitch and gpudirect. *IEEE Transactions on Parallel and Distributed Systems 31*, 1 (2019), 94–110.

[16] Li, Q., Gama, F., Ribeiro, A., and Prorok, A. Graph neural networks for decentralized multi-robot path planning. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2020), IEEE, pp. 11785–11792.

[17] Li, Y., Yu, Z., Choy, C., Xiao, C., Alvarez, J. M., Fidler, S., Feng, C., and Anandkumar, A. Voxformer: Sparse voxel transformer for camera-based 3d semantic scene completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2023), pp. 9087–9098.

[18] Lin, T., Kong, L., Stich, S. U., and Jaggi, M. Ensemble distillation for robust model fusion in federated learning. *Advances in Neural Information Processing Systems 33* (2020), 2351–2363.

[19] Liu, R., and Choi, N. A first look at wi-fi 6 in action: Throughput, latency, energy efficiency, and security. *Proceedings of the ACM on Measurement and Analysis of Computing Systems 7*, 1 (2023), 1–25.

[20] Liu, S., Li, X., Lu, H., and He, Y. Multi-object tracking meets moving uav. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2022), pp. 8876–8885.

[21] Masiukiewicz, A. Throughput comparison between the new hew 802.11 ax standard and 802.11 n/ac standards in selected distance windows. *International Journal of Electronics and Telecommunications 65*, 1 (2019), 79–84.

[22] McNally, W., Vats, K., Wong, A., and McPhee, J. Rethinking keypoint representations: Modeling keypoints and poses as objects for multi-person human pose estimation. In *European Conference on Computer Vision* (2022), Springer, pp. 37–54.

[23] Noormohammadpour, M., and Raghavendra, C. S. Datacenter traffic control: Understanding techniques and tradeoffs. *IEEE Communications Surveys & Tutorials 20*, 2 (2017), 1492–1525.

[24] NVIDIA. The world's smallest ai supercomputer. https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-xavier-series/, 2024.

[25] Ohkawa, T., Yamashina, K., Kimura, H., Ootsu, K., and Yokota, T. Fpga components for integrating fpgas into robot systems. *IEICE TRANSACTIONS on Information and Systems 101*, 2 (2018), 363–375.

[26] Pei, Y., Mutka, M. W., and Xi, N. Connectivity and bandwidth-aware real-time exploration in mobile robot networks. *Wireless Communications and Mobile Computing 13*, 9 (2013), 847–863.

[27] Ren, Y., Tung, C.-W., Chen, J.-C., and Li, F. Y. Proportional and preemption-enabled traffic offloading for ip flow mobility: Algorithms and performance evaluation. *IEEE Transactions on Vehicular Technology 67*, 12 (2018), 12095–12108.

[28] Sarkar, N. I., and Mussa, O. The effect of people movement on wi-fi link throughput in indoor propagation environments. In *IEEE 2013 Tencon-Spring* (2013), IEEE, pp. 562–566.

[29] Simonyan, K., and Zisserman, A. Very deep convolutional networks for large-scale image recognition, 2015.

[30] Wang, H., Potluri, S., Luo, M., Singh, A. K., Sur, S., and Panda, D. K. Mvapich2-gpu: optimized gpu to gpu communication for infiniband clusters. *Computer Science-Research and Development 26*, 3 (2011), 257–266.

[31] Wang, J., Sun, Z., Guan, X., Shen, T., Zhang, Z., Duan, T., Huang, D., Zhao, S., and Cui, H. Agrnav: Efficient and energy-saving autonomous navigation for air-ground robots in occlusion-prone environments. In *IEEE International Conference on Robotics and Automation (ICRA)* (2024).

[32] Wang, L., and Yoon, K.-J. Knowledge distillation and student-teacher learning for visual intelligence: A review and new outlooks. *IEEE transactions on pattern analysis and machine intelligence 44*, 6 (2021), 3048–3068.

[33] Wu, T., He, S., Liu, J., Sun, S., Liu, K., Han, Q.-L., and Tang, Y. A brief overview of chatgpt: The history, status quo and potential future development. *IEEE/CAA Journal of Automatica Sinica 10*, 5 (2023), 1122–1136.

[34] Xia, Z., Liu, Y., Li, X., Zhu, X., Ma, Y., Li, Y., Hou, Y., and Qiao, Y. Scpnet: Semantic scene completion on point cloud. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2023), pp. 17642–17651.

[35] Yang, X., Lin, H., Li, Z., Qian, F., Li, X., He, Z., Wu, X., Wang, X., Liu, Y., Liao, Z., et al. Mobile access bandwidth in practice: Measurement, analysis, and implications. In *Proceedings of the ACM SIGCOMM 2022 Conference* (2022), pp. 114–128.

[36] Yang, Y., Juntao, L., and Lingling, P. Multi-robot path planning based on a deep reinforcement learning dqn algorithm. *CAAI Transactions on Intelligence Technology 5*, 3 (2020), 177–183.