As edge robotics advances, one central question persists: **Can we directly deploy current state-of-the-art robotics methods on edge devices, or do they need significant adaptation?** Edge robots, defined by their capability to process data locally with limited resources, face unique challenges that complicate the use of traditional robotics algorithms. In this post, we delve into why existing methods often fall short and explore the latest research on necessary adaptations for effective deployment.

1. The Limitations of Edge Robotics

Edge robots, like autonomous drones, mobile robots, and wearable robotic systems, must operate with limited resources in unpredictable environments. The primary constraints include:

- **Restricted Computation**: Unlike server-based systems, edge robots lack high-powered GPUs or large memory banks.
- **Energy Constraints**: Most edge robots rely on batteries, limiting the power available for processing.
- **Real-Time Demands**: Tasks such as navigation, mapping, and object detection require near-instant responses, often under severe latency constraints.

Current Methods: Bottlenecks in Deployability

The majority of state-of-the-art algorithms in robotics assume ample computational resources, making direct deployment on edge robots problematic. Below, we explore the specific issues with some of the most commonly used techniques.

2. The Challenges with Current Robotics Algorithms

Deep Reinforcement Learning (DRL)

Deep Reinforcement Learning (DRL) is a popular method for decision-making and navigation. However, several studies highlight the **resource intensity** of DRL:

- **Memory Usage**: DRL models often require millions of parameters to learn complex policies, which demands memory resources far beyond the capabilities of most edge devices. Deploying DRL-based policies without compression can lead to significant slowdowns, impacting real-time performance [^1].

- **Inference Latency**: DRL algorithms rely on deep neural networks that demand high inference speeds, especially in safety-critical tasks. Without access to GPUs, processing speeds are insufficient for real-time decision-making, leading to unreliable behavior.

Simultaneous Localization and Mapping (SLAM)

Simultaneous Localization and Mapping (SLAM) is fundamental for creating maps of unknown environments while tracking a robot's position. Modern SLAM techniques use a combination of **Visual SLAM (V-SLAM)** and **Lidar-based SLAM**:

- **Computational Complexity**: State-of-the-art SLAM algorithms involve dense feature extraction, point-cloud registration, and optimization steps that are computationally intensive. V-SLAM algorithms struggle to achieve adequate frame rates on edge devices without aggressive downsampling, affecting map accuracy [^2].

- **Energy Drain**: SLAM requires intensive use of sensors like LiDAR and stereo cameras, leading to rapid battery depletion. Energy-efficient SLAM alternatives often sacrifice mapping accuracy, creating a trade-off between precision and operational duration [^3].

Large-Scale Neural Networks for Vision

Robotics applications rely heavily on **Convolutional Neural Networks (CNNs)** for tasks like object detection, segmentation, and classification. State-of-the-art models such as ResNet, YOLO, and Mask R-CNN pose challenges on edge devices:

- **Model Size**: Modern CNNs can have millions of parameters. A standard YOLOv5 model, for example, has over 7 million parameters, consuming several hundred MBs of memory, far exceeding the storage capacity of most edge platforms.
- **Inference Throughput**: Deploying large-scale vision models on edge hardware results in substantial inference slowdowns, making them impractical for time-sensitive applications [^4].

Model Predictive Control (MPC) and Optimal Control

Model Predictive Control (MPC) is a widely-used method for trajectory planning and control in robotics:

- **Computational Demand**: MPC involves solving optimization problems at every control step, which is computationally prohibitive on low-power CPUs. Real-time MPC for high-dimensional systems requires dedicated hardware, which many edge devices lack [^5].
- **Sensitivity to Disturbances**: Adaptive MPC requires significant computational overhead to handle environmental uncertainties, which is challenging on edge platforms without sacrificing control performance.

3. Recent Research on Adaptations for Edge Deployability

Model Compression and Pruning

Compression Techniques like pruning, quantization, and neural architecture search (NAS) have been studied to reduce model sizes:

- **Pruning** involves removing unnecessary weights from neural networks, reducing memory footprints. Pruning can make complex models feasible for edge platforms, although at the cost of some accuracy [^6].
- **Quantization** reduces the precision of weights, decreasing computational load. Quantizing models can result in performance degradation, particularly in tasks requiring fine-grained accuracy, like robotic manipulation [^7].

TinyML: Creating Edge-Specific Models

TinyML refers to machine learning techniques explicitly designed for resource-constrained devices:

- Techniques like **Neural Architecture Search (NAS)** have been proposed to automatically discover lightweight architectures optimized for edge devices. TinyNAS models have demonstrated a

significant reduction in computational requirements while maintaining accuracy for specific tasks [^8].

- **Spiking Neural Networks (SNNs)**, which mimic the brain's spike-based information processing, have shown promise for power-efficient robotics. SNNs can achieve comparable performance to traditional CNNs for certain robotic tasks while consuming less power [^9].

Distributed Learning for Edge Swarms

Federated Learning is gaining attention for edge robotics, where multiple devices share updates to train models collaboratively without centralizing data:

- Federated learning has enabled swarms of drones to coordinate and adapt without sending raw data back to a central server, reducing communication overhead and preserving data privacy [^10].

4. The Need for New Paradigms in Robotics Algorithms

To meet the unique demands of edge robots, future research must focus on inherently lightweight and adaptive algorithms:

- **Dynamic Neural Networks** that adjust their complexity based on real-time resource constraints.
- **Event-Driven Processing**, where computations are triggered by events rather than continuously, reducing idle power consumption.
- **Bio-inspired Models** that emulate efficient natural processes, leading to inherently resource-efficient computation.

References

[^2]: Mur-Artal, R., & Tardós, J. D. (2017). ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras. *IEEE Transactions on Robotics*. [^3]: Zhang, Q., & Scaramuzza, D. (2020). Energy-Efficient Visual-Inertial Odometry for Micro Aerial Vehicles. *IEEE Robotics and Automation Letters*. [^4]: Han, S., Mao, H., & Dally, W. J. (2016). Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. *International Conference on Learning Representations (ICLR)*. [^5]: Kouvaritakis, B., & Cannon, M. (2016). Model Predictive Control: Classical, Robust, and Stochastic. *Springer*. [^6]: He, Y., Lin, J., & Wang, Z. (2018). AMC: AutoML for Model Compression and Acceleration on Mobile Devices. *European Conference on Computer Vision (ECCV)*. [^7]: Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., & Wu, Y. (2018). Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [^8]: Tan, M., & Le, Q. (2019). EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. *International Conference on Machine Learning (ICML)*. [^9]: Tavanaei, A., & Maida, A. (2019). Bio-Inspired Spiking Neural Networks for Object Recognition in Surveillance Systems. *Frontiers in Neuroscience*. [^10]: Konečný, J., McMahan, H. B., & Ramage, D. (2016). Federated Optimization: Distributed Machine Learning for On-Device Intelligence. *arXiv preprint arXiv:1610.02527*.