Unified Local-Cloud Decision-Making via Reinforcement Learning

Kathakoli Sengupta, Zhongkai Shangguan, Sandesh Bharadwaj, Sanjay Arora[†], Eshed Ohn-Bar, and Renato Mancuso

Boston University [†]Red Hat

Abstract. Embodied vision-based real-world systems, such as mobile robots, require a careful balance between energy consumption, compute latency, and safety constraints to optimize operation across dynamic tasks and contexts. As local computation tends to be restricted, offloading the computation, i.e., to a remote server, can save local resources while providing access to high-quality predictions from powerful and large models. However, the resulting communication and latency overhead has led to limited usability of cloud models in dynamic, safety-critical, real-time settings. To effectively address this trade-off, we introduce UniLCD, a novel hybrid inference framework for enabling flexible local-cloud collaboration. By efficiently optimizing a flexible routing module via reinforcement learning and a suitable multi-task objective, UniLCD is specifically designed to support the multiple constraints of safety-critical end-to-end mobile systems. We validate the proposed approach using a challenging, crowded navigation task requiring frequent and timely switching between local and cloud operations. UniLCD demonstrates improved overall performance and efficiency, by over 23% compared to state-of-the-art baselines based on various split computing and early exit strategies. Our code is available at https://unilcd.github.io/.

Keywords: Task Offloading · Efficiency · Reinforcement Learning · Navigation

1 Introduction

We are currently undergoing a transformative societal phase as vision-based systems, such as mobile robots and personalized devices, transition from their controlled lab environments and into the real world. However, computational and energy constraints currently hinder the performance of deployed real-world systems, e.g., high-cost systems today may only be able to support lightweight neural network models with limited accuracy and runtime of about an hour before the on-board battery is depleted [22, 25, 38, 58, 62, 111]. Current efficiency bottlenecks may only become worse over time given ever-increasing sensor resolutions and larger models [2, 5, 53, 56, 79, 85].

To address the need for reliable and accurate inference, the currently prevailing approach for processing with increasingly powerful models involves sending data, e.g., an image, to remote cloud servers in order to offload computation [2, 5, 38, 51, 54, 106]. While this practice can benefit real-world systems through delivering high-quality model predictions, the transmission and cloud processing overhead incurs a high latency cost and is therefore not suitable for dynamic real-world agents, such as mobile

systems, which must continuously anticipate and respond to dynamic settings. Safety-critical systems with hard real-time constraints and responsiveness may solely rely on local processing, while leveraging strategies such as model pruning [23, 35, 43, 97] and quantization [44, 73, 89] to support embodied constraints, e.g., limited hardware and battery life. However, on-device lightweight models often suffer from significantly degraded accuracy and in turn can hinder safe decision-making, i.e., missing or falsely detecting nearby pedestrian. Thus, we seek to address a fundamental research question: How to realize robust vision-based systems that can be flexibly optimized for both *safety and real-time efficiency* while operating in dynamic real-world settings?

While shared local-cloud computation frameworks have been extensively studied by prior works, e.g., for various Internet of Things applications [24, 30, 31, 37, 37, 38, 51, 70, 106], the aforementioned methods rarely analyze safety-critical, real-time operation. Instead, standard off-loading frameworks rely on various ad-hoc and inadequate heuristics, e.g., based on a task-agnostic layer-wise split or simplistic statistics [37, 38], and thus cannot accommodate dynamic inference, i.e., based on the difficulty or safety-constraints of the current scenario. For instance, some scenarios may require split-second decisions, such as in the case of dense and dynamic scenarios with crowded surrounding pedestrians. In this work, we develop a more generalized paradigm enabling situation-specific collaboration between cloud computing and local inference to balance the energy cost while maintaining high system accuracy. Our method can be used to flexibly optimize for multiple constraints, i.e., latency, accuracy, efficiency, and safety, while also enabling systems with a highly sustainable operation (a critical need as compute-intensive systems become more widely adopted [82]).

Contributions: Towards facilitating real-time, cloud-enabled systems, we make three key contributions. First, we introduce a novel cloud-local hybrid collaboration framework, UniLCD. Our proposed reinforcement learning-based framework is inspired by the observation that small, low-power consumption models on edge devices are sufficient in some scenarios, while more complex conditions may call for powerful cloud resources. Our approach employs a conditional routing module that learns to dynamically route computation between local and cloud resources. Second, we demonstrate the importance of carefully designing the reward function in order to effectively balance over multiple constraints, i.e., energy consumption, latency, and safe performance in a context-dependent manner. Third, we validate our method and several state-of-the-art baselines by introducing a novel benchmark comprising challenging vision-based crowded navigation tasks that require seamless decision-making with frequent cloud-local switching. We demonstrate our approach to significantly outperform prior methods [37, 38] by 17% across multiple metrics, as well as a proposed ecological score. Our code and benchmark are available at https://unilcd.github.io/ for future researchers tackling multifaceted challenges in practical, real-world, vision-based systems.

2 Related Work

Cloud-Edge Collaborative Systems: The integration of cloud computing and edge devices has attracted increasing attention in recent years as it can potentially combine the

Table 1: Comparison with Prior Work. By holistically tackling aspects of system safety, efficiency, and latency, our framework is suitable for real-time decision-making in dynamic scenes.

Method	Low Latency	Edge Deployment	End-to-End Training	Situational	Embedding	High Accuracy	Real Time
On-Device	✓	✓	-	✓	✗	✗	✓
On-Cloud	✗	✗	-	✓	✗	✓	✗
Neurosurgeon [38]	✗	✓	✓	✓	✓	✓	✗
Dynamic [26]	✓	✓	✗	✗	✓	✗	✗
Selective Query [37]	✓	✓	✓	✗	✗	✓	✗
Compressive Offloading [100]	✗	✗	✓	✓	✓	✓	✗
Adaptive Offloading [88]	✓	✓	✓	✓	✗	✗	✗
Deep Sequential RL [90]	✓	✓	✓	✓	✓	✗	✗
UniLCD (ours)	✓	✓	✓	✓	✓	✓	✓

advantages from both [16, 24, 72, 76]. Leveraging the cloud's capacity to employ advanced hardware and deploy large models allows quick execution of computationally intensive tasks with accurate results, while edge devices enable real-time processing and diminish latency by accessing data close to the source. Thus, cloud-edge collaboration systems have been developed across various domains, such as autonomous driving [41] and smart cities [95]. Recently, Kag et al. [37] proposed a hybrid approach that learns to efficiently select queries for cloud processing. However, the method focuses on simplistic image classification domains without integrating sensitivity to computational delays. In contrast, our framework emphasizes the importance of *both prompt and accurate* cloud-edge collaboration and can thus more flexibly support broad applications, e.g., with handling of dynamic and complex safety-critical scenarios.

Real-Time Decision-Making: Real-time decision-making is critical for scenarios that require fast and accurate responses (e.g., autonomous driving [45, 46, 55, 102], energy grid management [80, 96], assistive technologies [15, 32, 86]). State-of-the-art models usually have billions of parameters, resulting in long inference times. Notably, on-going work is primarily concentrated on enhancing data and model processing speed [21, 92], as well as designing dedicated processing chips [39, 81]. In this work, we tackle a complementary direction of cloud-edge collaboration, where we explore the integration of the cloud's accelerated processing speeds with the deployment of edge devices. This integration aims to effectively mitigate latency concerns while ensuring the reliability of prompt decision-making, which generally results in efficiency and latency costs [4, 60, 71].

Early-Exit Models: Early exit techniques are closely related to our work as they incorporate internal classifiers at various shallow layers, allowing the model to exit earlier during inference while still predicting the correct label, i.e., to save time and energy costs [34, 38, 47, 50, 74, 83, 98]. Yet, despite the fast inference, models can still suffer in latency and accuracy [38]. Li et al. introduced AppealNet [50], an architecture that efficiently processes deep learning tasks by predicting whether inputs can be managed by a resource-constrained edge device or need to be offloaded to a cloud-based model. However, such early exit strategies only addresses on-device efficiency without con-

sidering overall task and safety, leading to potentially unsafe and frequent navigation errors in our robot navigation task, as will be shown in our analysis in Sec 4.

Energy-Efficient Models: Our proposed method primarily aims to present timely results while minimizing the energy cost from both cloud servers and edge devices, which can not only help reduce carbon emissions [3, 11, 42, 109] but also increase battery life and reduce total overhead costs [20, 25, 54]. The promise of energy-efficient models has been substantially developed through various research fields, focusing on optimizing computational efficiency and resource utilization. One aspect of the field of energy-efficient models emphasizes hardware design [36, 66]. However, hardware development is often time-consuming and expensive. Another aspect provides insight into the efficacy of model pruning and quantization techniques that can minimize model size thus helping save energy [99, 110] consumption. In our work, we delve into resource allocation and optimization strategies, seeking to further improve energy efficiency.

Decision-Making Models for Mobile Systems: Robotic companions and assistants designed to operate in human-inhabited environments has made significant progress [8, 40, 103]. Imitation learning [33, 101, 104, 108] involves the system learning from observed behavior, typically demonstrated by a human or professional operator, to guide its decision-making process in mobile systems, such as autonomous vehicles [68, 84] or drones [93]. Reinforcement learning (RL) [63, 94] has also been applied to optimize decision-making processes, allowing robots to navigate efficiently while avoiding obstacles and optimizing path planning [6, 90, 107]. However, the wide-scale deployment of real-time vision-based navigation models on robots today is hindered by ever-increasing computational and hardware constraints [19, 52], making it difficult to deploy accurate but large models on systems. In our work, our RL-based framework can be used to query the cloud efficiently, i.e., to reduce overall energy cost without sacrificing local resources. [27, 57, 91, 100]. Our flexible framework is model and platform-agnostic, while automatically adapting across situations and communication settings.

3 Method

Our objective is to learn a *policy* that determines the dynamic allocation of local and cloud resources, optimizing energy efficiency and real-time performance. In the following, we discuss the main components of our method. First, we formulate our real-time task of navigating to a specific goal in Sec. 3.1. Next, we employ imitation learning to train two navigation policies tailored for local and cloud settings, respectively (Sec. 3.2). Third, we discuss how we train our sample-efficient *routing policy* via Proximal Policy Optimization (PPO) [78] (Sec. 3.3). In particular, we introduce our multi-objective reward, designed to optimize energy efficiency while maintaining navigation performance. Fig. 1 depicts an overview of our method.

3.1 Problem Formulation

We formulate our robot navigation task as a real-time sequential decision-making problem from a set of observations $\mathbf{o} = \{\mathbf{I}, \mathbf{p}\} \in \mathcal{O}$, comprising a front-view camera image

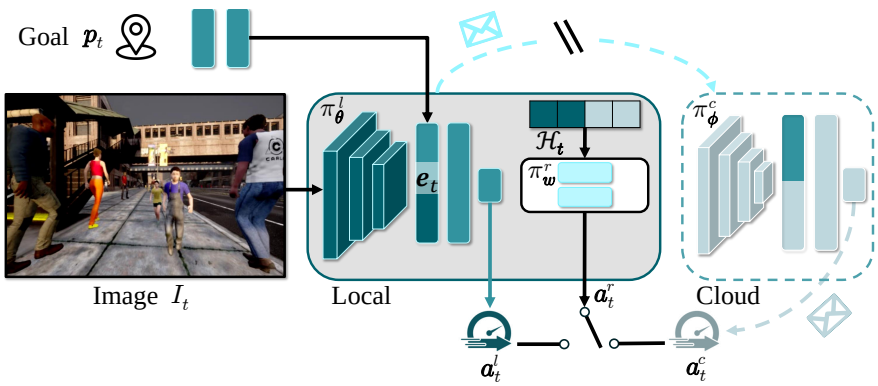

Fig. 1: Overview of UniLCD. Our system comprises a situational routing module, which takes the current embedding and a history of previous actions. Local actions are predicted by a pre-trained lightweight model that can be deployed efficiently on a the mobile system. The sample-efficient routing module, trained via RL, determines whether to implement the local action or transmit the scene embedding to the cloud server model, which is more accurate but computationally expensive and induces latency.

$\mathbf{I} \in \mathbb{R}^{W\times H\times 3}$, the next waypoint $\mathbf{p} \in \mathbb{R}^2$, to a set of actions $\mathbf{a} = \{d, v\} \in \mathcal{A}$, where $d \in \mathbb{R}$ represents the orientation and $v \in \mathbb{R}$ is the speed [13, 18]. Our objective is to obtain a mapping function $f_{\boldsymbol{\Theta}}: \mathcal{O} \rightarrow \mathcal{A}$ parameterized by $\boldsymbol{\Theta} = [\boldsymbol{\theta}, \boldsymbol{\phi}, \boldsymbol{\omega}]$ for generating actions at each time step to minimize the overall energy cost from both the local device and cloud server [9]. Our approach consists of three policies: a local and a cloud navigation policy $\pi^l_{\boldsymbol{\theta}}$ and $\pi^c_{\boldsymbol{\phi}}$ with weights $[\boldsymbol{\theta}, \boldsymbol{\phi}]$ that map the observations to actions, producing $\mathbf{a}^l_t$ and $\mathbf{a}^c_t$, respectively. Additionally, we introduce a routing policy $\pi^r_{\boldsymbol{\omega}}(\mathbf{o}|\pi^l_{\boldsymbol{\theta}}, \pi^c_{\boldsymbol{\phi}})$ parameterized by $\boldsymbol{\omega}$, to dynamically determine the optimal utilization of local resources versus offloading to the cloud server.

While our method can be optimized end-to-end, we initialize the two navigation policies in an imitation learning manner [13, 28, 69]. We also note that, given the constraints of local device hardware in our setting, the number of parameters for the local navigation policy is generally smaller than that of the cloud navigation policy. For comparative analysis and sample efficiency, after training the navigation policies, we freeze the parameters $\boldsymbol{\theta}, \boldsymbol{\phi}$ and train the routing policy $\pi^r_{\boldsymbol{\omega}}(\mathbf{o}|\pi^l_{\boldsymbol{\theta}}, \pi^c_{\boldsymbol{\phi}})$ based on the observations and the two navigation policies using residual RL. It's noteworthy that both the local navigation policy and the routing policy are deployed on a local device, sharing parameters for the initial layers of the neural network, as shown in Fig. 1.

Cloud-Aware Robot Navigation Task: Due to the challenges of obtaining sufficient data in the dynamic real-world scenario [12, 61, 87], we develop a simulation environment tailored for robot navigation through a crowded outdoor setting. Our environment is built upon CARLA (version 0.9.13) [18], an open-source simulator typically employed for testing autonomous driving algorithms. The information transmission between the local and the cloud server potentially introduces a stochastic delay [7, 29, 38, 105].

Algorithm 1 UniLCD's Routing Policy Training with Reinforcement Learning

Input: Image $\mathbf{I}$, next waypoint $\mathbf{p}$, local policy $\pi^l_{\boldsymbol{\theta}}$, cloud policy $\pi^c_{\boldsymbol{\phi}}$
Initialize: Number of iterations T, history $\mathcal{H}$, routing policy $\pi^r_{\boldsymbol{\omega}}$, reply buffer $\mathcal{S}$
Collect on policy samples:
for t $= 1$ **to** T **do**
    Obtain local action $\mathbf{a}^l_t$ and embeddings $\mathbf{e}_t$ using local policy $\pi^l_{\boldsymbol{\theta}}(\mathbf{I}_t, \mathbf{p}_t)$
    Append $(\mathbf{a}^l_t, 0)$ to history $\mathcal{H}_t$
    if $\pi^r_{\boldsymbol{\omega}_t}(\mathcal{H}_t, \mathbf{e}_t) = 0$ **then** $\mathbf{a}_t = \mathbf{a}^l_t$
    else
        Send $\mathbf{e}_t$ to cloud, $\mathbf{a}_t = \pi^c_{\boldsymbol{\phi}}(\mathbf{I}_t, \mathbf{p}_t)$
        Update last value of $\mathcal{H}_t$ to $(\mathbf{a}_t, 1)$
    end if
    Compute instant reward using Eq. (2)
    if Arrived destination **then** break
    end if
    Update replay buffer $\mathcal{S} = \mathcal{S} \cup \{\mathbf{I}_t, \mathbf{p}_t, \mathcal{H}_t, r_t\}$
    Update routing policy parameters with PPO
end for

Toward Energy-Efficiency: The primary objective of our research is to enhance energy efficiency while preserving task performance [30, 38, 70]. In our study, the navigation task is offloaded to the cloud for processing only when local computation is insufficient for achieving optimal task performance. However, communication bottlenecks necessitate prioritizing local processing to conserve energy and minimize latency. Offloading to the cloud is reserved for situations where only the cloud can meet the task performance requirements, such as challenging scenarios like navigating in bad weather [65] or crowded environments. Consequently, we leverage a routing network trained with PPO and suitable for running on local devices. This modular architecture is then optimized for energy efficiency, as discussed below.

3.2 Learning Local and Cloud Policies

In our work, we follow the standard imitation learning approach to train our navigation policies in an offline manner. Specifically, we first collect a dataset $\mathcal{D} = \{\mathbf{I}_i, \mathbf{p}_i, \mathbf{a}_i\}_{i=1}^N$ on diverse and complex routes and weathers using CARLA [18, 67] to simulate real-world scenarios to provide the basis for our navigation policies.

As shown in Fig. 1, both the local and cloud navigation policies comprise (i) a visual semantics feature extractor module for obtaining embeddings from input images, (ii) a multilayer perceptron to extract the feature associated with the robot's imminent goal point, and (iii) a goal-conditional module that takes concatenation of image embeddings and goal features to predict robot actions, encompassing both direction d and speed v. To share common features and accommodate local computing resource constraints, we first train a robust cloud policy with a large neural network. Subsequently, we freeze the parameters of the first few layers of the pre-trained cloud policy to serve as a shared feature extractor for the local policy. Additional fully connected layers are then added to the extracted features to form the local policy. Therefore, the local policy is significantly

smaller compared to the cloud policy. Only the parameters of the fully connected layer in the local policy are optimized, leading to reduced computational consumption and improved efficiency in both training and inference time. The learning objective for both local and cloud policies is achieved by minimizing the $\mathcal{L}_1$ distance:

$$\text{minimize}\, \mathbb{E}_{(\mathbf{I},\mathbf{p},\mathbf{a})\sim D}[\mathcal{L}_1(\mathbf{a}, \pi(\mathbf{I}, \mathbf{p}))] \tag{1}$$

where π represents navigation policy $\{\pi^l_{\boldsymbol{\theta}}, \pi^c_{\boldsymbol{\phi}}\}$.

3.3 Learning a Routing Policy

To effectively balance cloud and local computing resources while achieving good performance at the same time, our study proposes a *routing policy* that seamlessly switches between local and cloud. We follow the standard PPO training process as shown in Algorithm 1 and the formulation of our routing policy is introduced below.

State Space: We denoted the current state as $\mathbf{s}_t = \{\mathbf{e}_t, \mathcal{H}_t\}$, where $\mathbf{e}_t$ represents the image embeddings and goal features extracted from the shared feature extractor employed by both local and cloud policies, as discussed in Sec. 3.2, and $\mathcal{H}_t = \{(\mathbf{a}_i, \mathbb{1}_i)\}^t_{t-k}$ is a sequence of the last k history actions, where $\mathbf{a}_i$ represents previous navigation actions and $\mathbb{1}_i$ is an indicator function indicating whether the action is obtained locally or from the cloud.

Action Space: The action produced by our router policy is a binary discrete value, indicating whether to accept the local navigation policy $\pi^l_{\boldsymbol{\theta}}$ or transmit the embeddings $\mathbf{e}_t$ to the cloud navigation policy $\pi^c_{\boldsymbol{\phi}}$.

Task Reward: It is challenging to optimize task performance and energy efficiency while considering the sub-optimal actions generated from a small local model and latency-induced cloud model. We design our reward function encompassing five key components: geodesic reward r_{geo}, speed reward r_{speed}, energy disadvantage bonus r_{energy}, extreme action clip r_{action}, and collision penalty $r_{collision}$. The overall reward function is defined as

$$r = (r_{geo} \cdot r_{speed} \cdot r_{energy} \cdot r_{action})^{\alpha} - r_{collision} \tag{2}$$

where $\alpha = 1/4$ is a scaling factor that scales the overall reward between $[0, 1]$. By using a multiplicative objective, we ease optimization across different instantaneous components of the reward, i.e., if one of the terms is low, the entire reward is affected. Hence, the multiplicative overall reward objective can also be scale invariant to some extent (however, clipping one of the reward terms can change its importance). In our analysis, we find it to improve optimization and reduce the need for careful hyper-parameter tuning in multi-objective RL. For collision reward, we treat the term separately as it is sparsely observed over selected frames as a large negative reward, as defined below.

Geodesic Reward: We introduce r_{geo} to align the robot's trajectory with the predefined path, penalizing the robot for movements that deviate from the path.

$$r_{geo} = (1 - \tanh(d_{geo})) \tag{3}$$

where d_{geo} represents the Euclidean distance from the robot's current position to the nearest waypoint on the pre-defined path.

Speed Reward: The speed reward, denoted as r_{speed}, is designed to motivate robots to reach their destination in the shortest time. Considering the impact of GPU resource utilization on task execution duration, speed reward encourages quick task completion thus minimizing the overall computing resource usage. The maximum speed of a robot in our environment is set to $m_v = 1.5m/s$ according to [49, 64], and we define the speed reward as

$$r_{speed} = v/m_v \tag{4}$$

where v represents the current speed.

Energy Disadvantage: In our pursuit of minimizing energy consumption as the primary objective, and considering the significantly lower energy cost of local computation compared to cloud transmission and computation, we introduce penalties when the robot seeks cloud assistance. Similar to the speed component, we formulate the normalized energy disadvantage reward as

$$r_{energy} = 1 - e/m_e \tag{5}$$

where m_e is the maximum energy cost at one step.

Extreme Action Clip: Due to the property that $tanh$ converges at extreme action values, leading to a potential risk of converging to local maxima, we introduce an action clipping mechanism to address this concern. Specifically, actions exceeding the maximum allowed value will be directly assigned a reward of 0. This precautionary measure is extended to both standardized directional and speed actions, as shown in Eq. (6)

$$r_{action} = \mathbb{1}(|r_{speed}| < \epsilon) \cdot \mathbb{1}(|\frac{d}{d_m}| < \epsilon) \tag{6}$$

where $\epsilon = 0.97$ is the threshold for clipping the actions, d is the next rotation angle suggested by the navigation model chosen and d_m is the maximum possible rotation, ensuring that extreme actions do not unduly influence the reward function.

Collision Disadvantage: Robot navigation needs to emphasize safety and endeavor to avoid collisions with obstacles, ensuring the successful accomplishment to the destination. To underscore the significance of collision avoidance, we include an episodic termination mechanism to emphasize the severity of collisions. Specifically, a substantially higher negative penalty $r_{collision}$ is assigned and the episode is terminated whenever a collision happens.

4 Experiments

4.1 Implementation Details

Data Collection: To learn the local and cloud navigation policies using imitation learning, we collect crowd navigation data from our CARLA environment. We simulate

environments with varying densities by introducing five, 15, 30, and 70 pedestrians into the ego robot's path, corresponding to low, medium, dense, and crowd-density settings, respectively. This data is then aggregated to form a comprehensive training dataset for both local and cloud navigation policies. Our data collection approach employs a heuristic policy: we establish 10 different paths with predefined waypoints, which the robot follows from the starting position to the destination. The robot avoids static obstacles and halts when other pedestrians block its path, resuming moving once the path is clear.

Local and Cloud Policies: As discussed in Sec. 3.2, our local and cloud policies share a common feature extractor that processes a 480×480 image and a 2D imminent position vector. Specifically, we utilize the initial layers of RegNet [77] to extract the image features, and two fully connected (FC) layers to capture the position features. For the local policy, the image feature is flattened and concatenated with the position feature, followed by additional FC layers to generate the local actions. For the cloud policy, both the image feature and position feature are transmitted from the local to the cloud. The image feature is then processed by the remaining layers of RegNet [77] and concatenated with the position feature. An MLP is built on top of the combined features to generate the cloud actions.

Routing Policy: Our routing policy follows the standard PPO training process, with a policy network that decides whether to route locally or to the cloud, and a value network that evaluates the chosen actions by estimating the value function. Both the policy and value networks are MLPs with two hidden layers, with the hidden size being 16 for the policy network and 256 for the value network.

Training Protocol: We use AdamW [59] optimizer and train for 200 epochs with a learning rate of 0.0001 using our robot navigation dataset for both local and cloud navigation policies. The routing policy is trained for $1{,}000$ episodes, each consisting at most $1{,}500$ steps, with a discount factor γ set to 0.99. Episodes are truncated if the agent collides with other objects, completes the designated number of steps, reaches the predefined destination, or deviates more than three meters from the predefined route.

4.2 Evaluation Metrics

Task Performance Evaluation: We adapt CARLA's evaluation metrics [1] to assess the navigation performance of our ego robot. In addition to commonly used metrics (e.g., success rate, route completion, etc.), we propose *Navigation Score*, denoted as NS, similar to the driving score [1], by normalizing the infraction counts per meter. NS is defined as

$$\text{NS} = RC \cdot {P_I}^{IC} \cdot P_{RD} \tag{7}$$

where RC is route completion, P_I represents the infraction penalty for collisions, IC represents the number of robot collisions per meter, and P_{RD} represents the penalty for route deviation. Following the CARLA leaderboard settings, P_I is set to 0.5, and P_{RD} is defined as

$$P_{RD} = \begin{cases} 0.8, & \text{if } RD > \epsilon_{RD} \\ 1.0, & \text{otherwise} \end{cases} \tag{8}$$

where $\epsilon_{RD} = 1.5$m is the route deviation threshold.

Energy Evaluation: In addition to the navigation-specific evaluations, we introduce metrics to assess the computational and communication overhead incurred during the execution of our algorithm. Specifically, the total energy consumption for each episode is defined as

$$\text{Energy} = E_{local} \cdot N_{local} + E_{cloud} \cdot N_{cloud} \tag{9}$$

where N_{local} and N_{cloud} represent the number of steps the local and cloud models are chosen by the routing policy in one episode, respectively. E_{local} and E_{cloud} are the energy costs for processing one observation locally or on the cloud server. We refer to the energy values suggested by [38] and scale them according to our model and image size. In our study, $E_{local} = 0.15J$ is the computation energy cost of the local policy, and $E_{cloud} = 1.5J$ comprises both the computation and communication energy costs of the cloud policy. Further details regarding the calculation can be found in the supplementary material.

Ecological Navigation Score: As our primary objective is to optimize the overall energy consumption while ensuring robot navigation performance, we introduce an *Ecological Navigation Score* (ENS) that balances navigation performance and efficiency considerations. The ENS is defined as

$$\text{ENS} = P_E \cdot \text{NS} \tag{10}$$

where P_E is the penalty term for energy consumption, and defined as

$$P_E = 1 - \frac{\text{Energy}}{N_E} \tag{11}$$

where $N_E = (E_{local} + E_{cloud}) \cdot (N_{local} + N_{cloud})$ is a normalization factor.

Run-Time: Real-time decision-making is crucial for robot navigation. In our study, the robots need to respond instantly to environmental changes and swiftly switch between local and cloud-based navigation policies to adjust path planning. To demonstrate this real-time response, we report our run-time Frames Per Second (FPS), which includes both the model processing time and the communication time between a local device and the cloud server.

4.3 Results

We use the CARLA simulator (version 0.9.13) [18] to validate the effectiveness of our proposed method. We tested our algorithm on five different routes in CARLA's Town 10, each with 30 episodes. In testing time, we explored diverse weather conditions (including hard rain, sunny, wet, and sunset) and varying traffic density to simulate realistic conditions [14]. The maximum length of the routes is 40 meters. In this section, we first compare UniLCD with state-of-the-art local-cloud collaboration solutions [37, 38, 88, 90, 100]. Subsequently, we evaluate the performance of UniLCD with various sizes of local models and data-transmission settings. Third, we conduct ablation studies across different crowd-density settings. Finally, we systematically assess

Table 2: Comparing UniLCD with Baselines. We categorize our comparative analysis between individual models (Local and Cloud only), baselines (Deep Learning and RL-based computational offloading research), and our proposed UniLCD variations, which show progressive improvement. † denotes methods transmitting to the cloud raw input data, i.e., instead of an embedding. ENS is Ecological Navigation Score (%), NS is Navigation Score (%), SR is Success Rate (%), RC is Route Completion (%), Infract. is Infraction Rate (/m), Energy is measured in Joules per meter (J/m) and FPS is Frames Per Second.

Method	**ENS↑**	**NS↑**	**SR↑**	**RC↑**	**Infract.↓**	**Energy↓**	**FPS↑**
† Cloud-Only [77]	0.00	96.47	93.33	98.50	0.03	36.49	7.11
Local-Only [75]	63.43	67.33	0.00	75.23	0.16	4.33	65.40
Baseline Methods:							
Compressive Offloading [100]	13.98	80.16	0.00	80.16	0.00	90.66	1.82
† Selective Query [37]	24.14	61.28	0.00	82.68	0.11	45.35	18.14
† Adaptive Offloading [88]	37.42	40.37	70.00	94.05	1.22	4.80	30.14
Neurosurgeon [38]	39.85	63.10	0.00	80.54	0.03	28.31	12.53
SPINN [48]	36.31	72.75	60.00	92.73	0.35	18.94	20.37
Deep Sequential RL [90]	58.84	61.83	0.00	79.36	0.36	3.77	77.94
UniLCD Module Ablations:							
† Standard Reward	48.35	54.99	0.00	75.23	0.13	3.57	50.20
† Standard Reward w/ History	50.04	57.21	10.00	77.71	0.12	8.38	49.07
† Our Reward (Eq. (2))	48.30	79.90	56.66	91.15	0.19	21.72	16.05
† Our Reward w/ History	71.70	87.71	83.33	94.66	0.11	7.83	33.98
Our Reward (Eq. (2))	57.20	87.39	60.00	91.10	0.06	6.60	**12.49**
Our Reward w/ History	**85.97**	**94.58**	**93.33**	**95.90**	**0.02**	**2.90**	26.49

the significance of each reward term by omitting one at a time and analyzing their respective impacts, thereby reinforcing the foundational rationale of our reward design. The last two ablation studies can be found in our supplementary.

Comparing UniLCD with Baselines: Tab. 2 depicts our main results, comparing against multiple baselines, including local and cloud-only models. We emphasize that prior work does not usually consider safety-critical, real-time tasks such as social navigation. As shown in Tab. 2, several baselines can achieve a viable route completion (RC) score, however, this comes at a cost, e.g., high infraction rates. This includes early-exit baselines, which provide a limited offloading mechanism. We show the ENS metric to drop to zero when assessing the high energy-consuming cloud model exclusively, demonstrating that conducting all computations in the cloud with the computation-heavy model is energy-consuming. Moreover, the local-only model demonstrates a poor navigation score of 67.33% and a high collision rate of 0.16, indicating that low-accuracy, device-only models are more susceptible to navigation errors. The baseline models show comparable ENS scores to the local-only method but do not improve much in other metrics. Early-exit strategies such as SPINN [48] and Neurosurgeon [38] address on-device efficiency with improved ENS scores up to 39.85%, but perform worse than offloading-based methods in other metrics. Furthermore, we compare our method with DNN-based offloading (Selective Query [37]), which fails to improve due to the dynamic nature of the task and over-reliance on the cloud. This particular baseline also requires transmitting raw data to the cloud, which is inefficient. The best performing

baseline, of Deep Sequential RL [90], achieves an ENS of 58.84%. UniLCD achieves a state-of-the-art results, with an ENS of 85.97%, achieving state-of-the-art performance. Our analysis also highlights the need for cloud-edge collaborative methods that can combine both the strengths of fast local inference and high-capacity cloud computation.

Impact of Reward Design: Tab. 2 also shows ablation of UniLCD using various rewards. We evaluate UniLCD using the standard reward design (e.g., [111]), where individual reward terms are added. We initially find the additive reward to result in comparable performance to our reward design (ENS of 48.35% compared to 48.30%). However, when added history, the additive reward does not improve, while ours is shown to outperform significantly (ENS of 50.04% for standard reward compared to 71.70% with ours). We attribute this to the normalization effect of the multiplicative function during training. This eases optimization, whereas, in additive overall reward, the contribution of each term may need to be carefully optimized. While it is possible that careful hyperparameter optimization can match these results, this highlights the flexibility and generalization of our objective, which performs well across different model inputs. This finding also uncovers the sensitivity of current methods for the reward design. Finally, we utilize an embedding from the local model as input to UniLCD to provide the necessary context for collision handling and transmit it to the cloud, i.e., instead of raw data to reduce energy costs. This improves overall performance, achieving an ENS of 85.97%, mostly due to reduced energy consumption, while maintaining high performance. The embedding-based communication architecture also reduces collisions due to a higher FPS, resulting in a higher NS.

Local Policy Backbone Ablations: Tab. 3 shows ablations of UniLCD for local backbones of various sizes. We run experiments with UniLCD that transmit raw image data to the cloud using local backbones such as MobileNetV2 (medium), MobileNetV3small (small), and MobileVIT (tiny), coupled with a RegNet (cloud) model. As an imitation model, the medium backbone demonstrates task performance closest to the cloud, with minimal instances of infractions followed by the small and tiny backbones. The UniLCD with history model consistently favors local at instances where both local and cloud models have similar capacity, i.e., the medium backbone. The algorithm successfully sustains an impressive average route completion rate of 94.66% with occasional cloud support and significantly reduces energy consumption to 7.83 J/m using a comparatively weaker local model with a smaller backbone. Further, testing local policy incapable of path-following or collision avoidance with a tiny backbone reveals a complete reliance on cloud assistance but results in the successful completion of all episodes.

We also run ablations of UniLCD that transmit embeddings to the cloud. The local models for these experiments are trained by freezing pre-trained cloud weights where stage 1 is fine-tuned with the feature extractor containing the early layers of the cloud model and stage 2 is fine-tuned with the feature extractor containing the final layers of the cloud model. Observations indicate that UniLCD with history model achieves an ENS of 85.97% solely with the stage 1 backbone. It is noteworthy that both models exhibit a bias towards local processing when a backbone comparable to cloud performance (stage 2) is employed, resulting in the highest ENS of 86.78%, and a route completion

Table 3: Local Policy Backbone Ablations. We test the task performance in the dense crowd setting for UniLCD and UniLCD w/ History. † denotes methods transmitting to the cloud raw input data, i.e., instead of an embedding. For UniLCD transmitting data, we use three dimensions of local models: Tiny, Small, and Medium. For UniLCD transmitting embedding, we use local models trained up to two stages: Stage 1 trained with fewer parameters and Stage 2 trained with more parameters. Params is Number of Parameters in each local model (M), ENS is Ecological Navigation Score (%), NS is Navigation Score (%), SR is Success Rate (%), RC is Route Completion (%), Infract. is Infraction Rate (/m), Energy is measured in Joules per meter (J/m), and FPS is Frames Per Second.

Local Model Size	**Params**	**ENS↑**	**NS↑**	**SR↑**	**RC↑**	**Infract.↓**	**Energy↓**	**FPS↑**
UniLCD:								
† Tiny	1.37	12.80	93.29	90.00	97.93	0.07	34.01	7.35
† Small	2.54	48.30	79.90	56.66	91.15	0.19	21.72	16.05
† Medium	3.50	50.92	87.87	80.00	95.50	0.12	21.19	15.10
UniLCD w/ History:								
† Tiny	1.37	0.00	91.27	**93.33**	**98.50**	0.11	36.52	6.42
† Small	2.54	71.70	87.71	83.33	94.66	0.11	7.83	33.98
† Medium	3.50	73.46	83.86	90.00	96.22	0.15	5.22	11.53
UniLCD:								
Stage 1	0.53	57.20	87.39	60.00	91.10	0.06	6.60	12.49
Stage 2	0.95	74.12	81.54	93.33	91.10	0.16	1.80	**65.40**
UniLCD w/ History:								
Stage 1	0.53	85.97	94.58	**93.33**	95.99	**0.02**	2.90	26.49
Stage 2	0.95	**86.78**	**95.47**	**93.33**	**98.15**	0.04	**1.77**	36.50

of 98.15% with minimal infractions, averaging 0.04 per meter. Overall, these findings underscore UniLCD's proficiency in leveraging cloud resources only when local processing is inadequate.

Performance over Training Progress: We also investigate the evolution of the ENS task metric over the training progress as shown in Fig. 2. Remarkably, our model and reward outperform the standard reward even at the onset of training despite not requiring careful hyper-parameter tuning. Nonetheless, as training progresses, the difference in the baselines becomes even more apparent. The ENS performance is presented over training iterations and averaged over 10 tests, i.e., using different environmental seeds. Results for performance in different environments and settings (e.g., crowd density) can be found in the supplementary. In our comparative analysis, we note that as the complexity of the scenario increases, our baseline approaches show a decline in task performance. This decline is evidenced by an increase in route deviation and infraction rates. While UniLCD can easily navigate comparatively simpler scenarios, its infraction rate increases only slightly. This observation suggests the algorithm's robust adaptability for effective navigation in diverse and intricate environments.

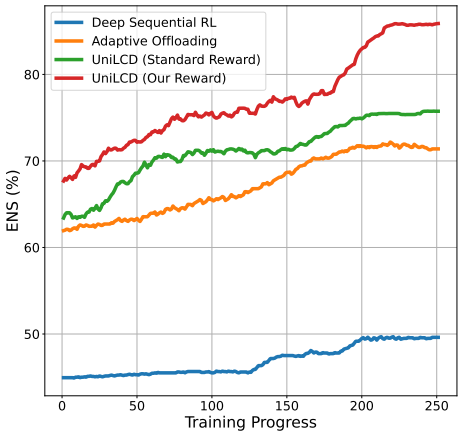

Fig. 2: Training Progress Results. We evaluate performance for different models, including UniLCD trained with a standard, carefully tuned, additive reward vs. UniLCD with the proposed reward function. We find consistently improved performance throughout the entire model training process. Results are shown for averaging across 10 evaluation seeds.

5 Conclusion and Future Work

We envision large-scale robot agents that collaborate and actively execute tasks in complex real-world scenarios. To this end, we design a PPO-based routing policy that learns to seamlessly switch between local and cloud policies depending on the situation and task objective of the observations. Specifically, we address the challenging task of real-time social robot navigation. We demonstrate that the proposed approach results in reduced overall energy cost while maintaining robust navigation performance. Our simulation environment can be used to facilitate more research into both *effective and efficient* navigation in the future. Thus, we aim to facilitate the development of more sustainable and robust societal-scale AI-based systems. Moreover, our approach can be extended to various platforms and real-time decision-making tasks that require both seamless communication and high performance. While we take a step towards quantifying trade-offs in real-time, safety-critical systems in challenging simulation settings, the next step would be to analyze UniLCD within diverse real-world environments and ambient settings (e.g., different communication settings, larger cloud models). Finally, although we consider the overarching objective of minimizing processing computation across both local and cloud devices, other aspects of efficiency can also be considered. For instance, maintaining local infrastructure often requires additional battery and hardware resources, which can be better optimized with cloud-based solutions especially when leveraging large, high-capacity models [10, 17].

Acknowledgments

We thank the Red Hat Collaboratory (award #2024-01-RH07) for supporting this research, and Bassel Mabsout for helpful discussions regarding the problem formulation.

References

1. Carla autonomous driving leaderboard. https://leaderboard.carla.org/ (2022)
2. Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F.L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al.: GPT-4 technical report. arXiv (2023)
3. Alharbi, H.A., Aldossary, M.: Energy-efficient edge-fog-cloud architecture for iot-based smart agriculture environment. Access (2021)
4. Aujla, G.S., Kumar, N., Zomaya, A.Y., Ranjan, R.: Optimal decision making for big data processing at edge-cloud environment: An sdn perspective. Trans. Ind. Inform. (2017)
5. Brohan, A., Brown, N., Carbajal, J., Chebotar, Y., Chen, X., Choromanski, K., Ding, T., Driess, D., Dubey, A., Finn, C., et al.: RT-2: Vision-language-action models transfer web knowledge to robotic control. CoRL (2023)
6. Caruso, M., Regolin, E., Camerota Verdù, F.J., Russo, S.A., Bortolussi, L., Seriani, S.: Robot navigation in crowded environments: A reinforcement learning approach. Machines (2023)
7. Charneski, A.: Modeling network latency. Online (2015)
8. Chen, C., Liu, Y., Kreiss, S., Alahi, A.: Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning. In: ICRA (2019)
9. Chen, X., Zhang, J., Lin, B., Chen, Z., Wolter, K., Min, G.: Energy-efficient offloading for dnn-based smart iot systems in cloud-edge environments. TPDS (2021)
10. Cheng, Y., Wang, D., Zhou, P., Zhang, T.: Model compression and acceleration for deep neural networks: The principles, progress, and challenges. IEEE Signal Process. Mag. (2018)
11. Chin, R., Morcos, A.S., Marculescu, D.: Pareco: Pareto-aware channel optimization for slimmable neural networks. In: ICML (2020)
12. Chung, M.J., Wang, M.J.J.: The change of gait parameters during walking at different percentage of preferred walking speed for healthy adults aged 20–60 years. Gait & posture (2010)
13. Codevilla, F., Müller, M., López, A., Koltun, V., Dosovitskiy, A.: End-to-end driving via conditional imitation learning. In: ICRA (2018)
14. Codevilla, F., Santana, E., López, A.M., Gaidon, A.: Exploring the limitations of behavior cloning for autonomous driving. In: ICCV (2019)
15. Dang, B., Zhao, W., Li, Y., Ma, D., Yu, Q., Zhu, E.Y.: Real-time pill identification for the visually impaired using deep learning. arXiv (2024)
16. Ding, S., Lin, D.: Dynamic task allocation for cost-efficient edge cloud computing. In: SCC (2020)
17. Dong, Q., Chen, X., Satyanarayanan, M.: Creating edge ai from cloud-based llms. In: IWMCSA (2024)
18. Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., Koltun, V.: Carla: An open urban driving simulator. In: CoRL (2017)
19. Dudek, G., Jenkin, M.: Computational principles of mobile robotics. CUP (2024)
20. El Haber, E., Nguyen, T.M., Ebrahimi, D., Assi, C.: Computational cost and energy efficient task offloading in hierarchical edge-clouds. In: PIMRC (2018)
21. Frantar, E., Alistarh, D.: Spdy: Accurate pruning with speedup guarantees. In: ICML (2022)
22. Fu, Z., Kumar, A., Malik, J., Pathak, D.: Minimizing energy consumption leads to the emergence of gaits in legged robots. In: CoRL (2021)
23. Gamanayake, C., Jayasinghe, L., Ng, B.K.K., Yuen, C.: Cluster pruning: An efficient filter pruning method for edge ai vision applications. JSTSP (2020)
24. Gan, Y., Pan, M., Zhang, R., Ling, Z., Zhao, L., Liu, J., Zhang, S.: Cloud-device collaborative adaptation to continual changing environments in the real-world. In: CVPR (2023)

25. Hadidi, R., Cao, J., Xie, Y., Asgari, B., Krishna, T., Kim, H.: Characterizing the deployment of deep neural networks on commercial edge devices. In: IISWC (2019)
26. Han, Y., Huang, G., Song, S., Yang, L., Wang, H., Wang, Y.: Dynamic neural networks: A survey. PAMI (2021)
27. Hanyao, M., Jin, Y., Qian, Z., Zhang, S., Lu, S.: Edge-assisted online on-device object detection for real-time video analytics. In: INFOCOM (2021)
28. Hawke, J., Shen, R., Gurau, C., Sharma, S., Reda, D., Nikolov, N., Mazur, P., Micklethwaite, S., Griffiths, N., Shah, A., et al.: Urban driving with conditional imitation learning. In: ICRA (2020)
29. Hooghiemstra, G., Van Mieghem, P.: Delay distributions on fixed internet paths. Delft University of Technology, report (2001)
30. Hu, L., Sun, G., Ren, Y.: Coedge: Exploiting the edge-cloud collaboration for faster deep learning. Access (2020)
31. Hu, X., Wang, L., Wong, K.K., Tao, M., Zhang, Y., Zheng, Z.: Edge and central cloud computing: A perfect pairing for high energy efficiency and low-latency. Trans. Wirel. (2019)
32. Huang, Z., Shangguan, Z., Zhang, J., Bar, G., Boyd, M., Ohn-Bar, E.: ASSISTER: Assistive navigation via conditional instruction generation. In: ECCV (2022)
33. Hussein, A., Gaber, M.M., Elyan, E., Jayne, C.: Imitation learning: A survey of learning methods. CSUR (2017)
34. Jazbec, M., Timans, A., Veljković, T.H., Sakmann, K., Zhang, D., Naesseth, C.A., Nalisnick, E.: Fast yet safe: Early-exiting with risk control. arXiv (2024)
35. Jiang, Y., Wang, S., Valls, V., Ko, B.J., Lee, W.H., Leung, K.K., Tassiulas, L.: Model pruning enables efficient federated learning on edge devices. Trans. Neural Netw. Learn. Syst. (2022)
36. Judd, P., Albericio, J., Hetherington, T., Aamodt, T.M., Moshovos, A.: Stripes: Bit-serial deep neural network computing. In: MICRO (2016)
37. Kag, A., Fedorov, I., Gangrade, A., Whatmough, P., Saligrama, V.: Efficient edge inference by selective query. In: ICLR (2022)
38. Kang, Y., Hauswald, J., Gao, C., Rovinski, A., Mudge, T., Mars, J., Tang, L.: Neurosurgeon: Collaborative intelligence between the cloud and mobile edge. SIGARCH (2017)
39. Karras, K., Pallis, E., Mastorakis, G., Nikoloudakis, Y., Batalla, J.M., Mavromoustakis, C.X., Markakis, E.: A hardware acceleration platform for AI-based inference at the edge. CSSP (2020)
40. Katyal, K.D., Hager, G.D., Huang, C.M.: Intent-aware pedestrian prediction for adaptive crowd navigation. In: ICRA (2020)
41. Kim, S.W., Ko, K., Ko, H., Leung, V.C.: Edge-network-assisted real-time object detection framework for autonomous driving. Network (2021)
42. Kimovski, D., Mathá, R., Hammer, J., Mehran, N., Hellwagner, H., Prodan, R.: Cloud, fog, or edge: Where to compute? Internet Comput. (2021)
43. Kong, Z., Dong, P., Ma, X., Meng, X., Sun, M., Niu, W., Shen, X., Yuan, G., Ren, B., Qin, M., et al.: SPViT: Enabling faster vision transformers via soft token pruning. arXiv (2021)
44. Kryzhanovskiy, V., Balitskiy, G., Kozyrskiy, N., Zuruev, A.: Qpp: Real-time quantization parameter prediction for deep neural networks. In: CVPR (2021)
45. Lai, L., Ohn-Bar, E., Arora, S., Yi, J.S.K.: Uncertainty-guided never-ending learning to drive. In: CVPR (2024)
46. Lai, L., Shangguan, Z., Zhang, J., Ohn-Bar, E.: XVO: Generalized visual odometry via cross-modal self-training. In: ICCV (2023)
47. Laskaridis, S., Kouris, A., Lane, N.D.: Adaptive inference through early-exit networks: Design, challenges and directions. In: IWEMDL (2021)
48. Laskaridis, S., Venieris, S.I., Almeida, M., Leontiadis, I., Lane, N.D.: Spinn: synergistic progressive inference of neural networks over device and cloud. In: MobiCom (2020)

49. Levine, R.V., Norenzayan, A.: The pace of life in 31 countries. J. Cross-Cult. Psychol. (1999)
50. Li, M., Li, Y., Tian, Y., Jiang, L., Xu, Q.: Appealnet: An efficient and highly-accurate edge/cloud collaborative architecture for dnn inference. In: DAC (2021)
51. Li, X., Dang, Y., Aazam, M., Peng, X., Chen, T., Chen, C.: Energy-efficient computation offloading in vehicular edge cloud computing. Access (2020)
52. Li, Z., Ren, T., He, X., Liu, C.: Red: A systematic real-time scheduling approach for robotic environmental dynamics. In: RTSS (2023)
53. Lin, B., Zhu, B., Ye, Y., Ning, M., Jin, P., Yuan, L.: Video-LLaVA: Learning united visual representation by alignment before projection. arXiv (2023)
54. Lin, B., Huang, Y., Zhang, J., Hu, J., Chen, X., Li, J.: Cost-driven off-loading for dnn-based applications over cloud, edge, and end devices. Trans. Ind. Inform. (2019)
55. Lin, Y., Zhang, J.w., Liu, H.: Deep learning based short-term air traffic flow prediction considering temporal–spatial correlation. Aerosp. Sci. Technol. (2019)
56. Liu, H., Li, C., Wu, Q., Lee, Y.J.: Visual instruction tuning. NeurIPS (2023)
57. Liu, L., Li, H., Gruteser, M.: Edge assisted real-time object detection for mobile augmented reality. In: ICMCN (2019)
58. Liu, R., Xu, X., Shen, Y., Zhu, A., Yu, C., Chen, T., Zhang, Y.: Enhanced detection classification via clustering svm for various robot collaboration task. arXiv (2024)
59. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. arXiv (2017)
60. Masoudi, M., Cavdar, C.: Device vs edge computing for mobile services: Delay-aware decision making to minimize power consumption. Trans. Mob. Comput. (2020)
61. Mavrogiannis, C., Baldini, F., Wang, A., Zhao, D., Trautman, P., Steinfeld, A., Oh, J.: Core challenges of social robot navigation: A survey. T-HRI (2023)
62. Miki, T., Lee, J., Hwangbo, J., Wellhausen, L., Koltun, V., Hutter, M.: Learning robust perceptive locomotion for quadrupedal robots in the wild. Sci. Robot. (2022)
63. Moerland, T.M., Broekens, J., Plaat, A., Jonker, C.M., et al.: Model-based reinforcement learning: A survey. Found. Trends Mach. Learn. (2023)
64. Mohler, B.J., Thompson, W.B., Creem-Regehr, S.H., Pick, H.L., Warren, W.H.: Visual flow influences gait transition speed and preferred walking speed. Exp. Brain Res. (2007)
65. Muşat, V., Fursa, I., Newman, P., Cuzzolin, F., Bradley, A.: Multi-weather city: Adverse weather stacking for autonomous driving. In: CVPR (2021)
66. Nguyen, D.T., Nguyen, T.N., Kim, H., Lee, H.J.: A high-throughput and power-efficient fpga implementation of yolo cnn for object detection. VLSI (2019)
67. Ohn-Bar, E., Prakash, A., Behl, A., Chitta, K., Geiger, A.: Learning situational driving. In: CVPR (2020)
68. Pan, Y., Cheng, C.A., Saigol, K., Lee, K., Yan, X., Theodorou, E., Boots, B.: Agile autonomous driving using end-to-end deep imitation learning. arXiv (2017)
69. Pan, Y., Cheng, C.A., Saigol, K., Lee, K., Yan, X., Theodorou, E.A., Boots, B.: Imitation learning for agile autonomous driving. Int. J. Robot. Res. (2020)
70. Park, S.H., Jeong, S., Na, J., Simeone, O., Shamai, S.: Collaborative cloud and edge mobile computing in c-ran systems with minimal end-to-end latency. Trans. Signal Inf. Process. (2021)
71. Park, S., Kwon, D., Kim, J., Lee, Y.K., Cho, S.: Adaptive real-time offloading decision-making for mobile edges: deep reinforcement learning framework and simulation results. Appl. Sci. (2020)
72. Penmetcha, M., Min, B.C.: A deep reinforcement learning-based dynamic computational offloading method for cloud robotics. Access (2021)
73. Polino, A., Pascanu, R., Alistarh, D.: Model compression via distillation and quantization. In: ICLR (2018)

74. Qendro, L., Campbell, A., Lio, P., Mascolo, C.: Early exit ensembles for uncertainty quantification. In: ML4H (2021)
75. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: Mobilenetv2: Inverted residuals and linear bottlenecks. In: CVPR (2018)
76. Satyanarayanan, M., Beckmann, N., Lewis, G.A., Lucia, B.: The role of edge offload for hardware-accelerated mobile devices. In: IWMCSA (2021)
77. Schneider, N., Piewak, F., Stiller, C., Franke, U.: Regnet: Multimodal sensor registration using deep neural networks. In: IV (2017)
78. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv (2017)
79. Shah, D., Osiński, B., Levine, S., et al.: LM-Nav: Robotic navigation with large pre-trained models of language, vision, and action. In: CoRL (2023)
80. Shangguan, Z., Lin, L., Wu, W., Xu, B.: Neural process for black-box model optimization under bayesian framework. arXiv (2021)
81. Shao, Y.S., Clemons, J., Venkatesan, R., Zimmer, B., Fojtik, M., Jiang, N., Keller, B., Klinefelter, A., Pinckney, N., Raina, P., et al.: Simba: Scaling deep-learning inference with multi-chip-module-based architecture. In: MICRO (2019)
82. Sudhakar, S., Sze, V., Karaman, S.: Data centers on wheels: emissions from computing onboard autonomous vehicles. Micro (2022)
83. Tang, S., Wang, Y., Kong, Z., Zhang, T., Li, Y., Ding, C., Wang, Y., Liang, Y., Xu, D.: You need multiple exiting: Dynamic early exiting for accelerating unified vision language model. In: CVPR (2023)
84. Teng, S., Chen, L., Ai, Y., Zhou, Y., Xuanyuan, Z., Hu, X.: Hierarchical interpretable imitation learning for end-to-end autonomous driving. T-IV (2022)
85. Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al.: LLaMA: Open and efficient foundation language models. arXiv (2023)
86. Treuillet, S., Royer, E.: Outdoor/indoor vision-based localization for blind pedestrian navigation assistance. Int. J. Image Graph. (2010)
87. Tsai, C.E., Oh, J.: A generative approach for socially compliant navigation. In: ICRA (2020)
88. Van Tam, N., Hieu, N.Q., Van, N.T.T., Luong, N.C., Niyato, D., Kim, D.I.: Adaptive task offloading in coded edge computing: A deep reinforcement learning approach. COMML (2021)
89. Wang, J.: Lightweight and real-time object detection model on edge devices with model quantization. In: JPCS (2021)
90. Wang, J., Hu, J., Min, G., Zhan, W., Ni, Q., Georgalas, N.: Computation offloading in multi-access edge computing using a deep sequential model based on reinforcement learning. Commun. Mag. (2019)
91. Wang, J., Hu, J., Min, G., Zomaya, A.Y., Georgalas, N.: Fast adaptive task offloading in edge computing based on meta reinforcement learning. TPDS (2020)
92. Wang, K., Liu, Z., Lin, Y., Lin, J., Han, S.: Haq: Hardware-aware automated quantization with mixed precision. In: CVPR (2019)
93. Wang, T., Chang, D.E.: Robust navigation for racing drones based on imitation learning and modularization. In: ICRA (2021)
94. Wiering, M.A., Van Otterlo, M.: Reinforcement learning. Adapt. Learn. Optim. (2012)
95. Wu, H., Zhang, Z., Guan, C., Wolter, K., Xu, M.: Collaborate edge and cloud computing with distributed deep learning for smart city internet of things. IoT-J (2020)
96. Wu, N., Wang, H.: Deep learning adaptive dynamic programming for real time energy management and control strategy of micro-grid. J. Clean. Prod. (2018)
97. Wu, T., Song, C., Zeng, P.: Model pruning based on filter similarity for edge device deployment. Front. Neurorobot. (2023)

98. Xia, G., Bouganis, C.S.: Window-based early-exit cascades for uncertainty estimation: When deep ensembles are more efficient than single models. In: ICCV (2023)
99. Yang, T.J., Chen, Y.H., Sze, V.: Designing energy-efficient convolutional neural networks using energy-aware pruning. In: CVPR (2017)
100. Yao, S., Li, J., Liu, D., Wang, T., Liu, S., Shao, H., Abdelzaher, T.: Deep compressive offloading: Speeding up neural network inference by trading edge computation for network latency. In: SenSys (2020)
101. Zhang, J., Huang, Z., Ohn-Bar, E.: Coaching a teachable student. In: CVPR (2023)
102. Zhang, J., Huang, Z., Ray, A., Ohn-Bar, E.: Feedback-guided autonomous driving. In: CVPR (2024)
103. Zhang, J., Zheng, M., Boyd, M., Ohn-Bar, E.: X-World: Accessibility, vision, and autonomy meet. In: ICCV (2021)
104. Zhang, J., Zhu, R., Ohn-Bar, E.: SelfD: Self-learning large-scale driving policies from the web. In: CVPR (2022)
105. Zhang, W., He, J.: Modeling end-to-end delay using pareto distribution. In: ICIMP (2007)
106. Zhang, X., Zhang, H., Zhou, X., Yuan, D.: Energy minimization task offloading mechanism with edge-cloud collaboration in iot networks. In: VTC (2021)
107. Zhu, K., Zhang, T.: Deep reinforcement learning based mobile robot navigation: A review. TST (2021)
108. Zhu, R., Huang, P., Ohn-Bar, E., Saligrama, V.: Learning to drive anywhere. CoRL (2023)
109. Zhu, W., Rosendo, A.: Psto: Learning energy-efficient locomotion for quadruped robots. Machines (2022)
110. Zhu, Y., Mottaghi, R., Kolve, E., Lim, J.J., Gupta, A., Fei-Fei, L., Farhadi, A.: Target-driven visual navigation in indoor scenes using deep reinforcement learning. In: ICRA (2017)
111. Zhuang, Z., Fu, Z., Wang, J., Atkeson, C., Schwertfeger, S., Finn, C., Zhao, H.: Robot parkour learning. In: CoRL (2023)