Imperative Learning: A Self-supervised Neuro-Symbolic Learning Framework for Robot Autonomy

Chen Wang[1], Kaiyi Ji[1], Junyi Geng[2], Zhongqiang Ren[3], Taimeng Fu[1], Fan Yang[4], Yifan Guo[5], Haonan He[3], Xiangyu Chen[1], Zitong Zhan[1], Qiwei Du[1], Shaoshu Su[1], Bowen Li[3], Yuheng Qiu[3], Yi Du[1], Qihang Li[1], Yifan Yang[1], Xiao Lin[1], and Zhipeng Zhao[1]

Abstract

Data-driven methods such as reinforcement and imitation learning have achieved remarkable success in robot autonomy. However, their data-centric nature still hinders them from generalizing well to ever-changing environments. Moreover, labeling data for robotic tasks is often impractical and expensive. To overcome these challenges, we introduce a new self-supervised neuro-symbolic (NeSy) computational framework, imperative learning (IL), for robot autonomy, leveraging the generalization abilities of symbolic reasoning. The framework of IL consists of three primary components: a neural module, a reasoning engine, and a memory system. We formulate IL as a special bilevel optimization (BLO), which enables reciprocal learning over the three modules. This overcomes the label-intensive obstacles associated with data-driven approaches and takes advantage of symbolic reasoning concerning logical reasoning, physical principles, geometric analysis, etc. We discuss several optimization techniques for IL and verify their effectiveness in five distinct robot autonomy tasks including path planning, rule induction, optimal control, visual odometry, and multi-robot routing. Through various experiments, we show that IL can significantly enhance robot autonomy capabilities and we anticipate that it will catalyze further research across diverse domains.

Keywords

Neuro-Symbolic AI, Self-supervised Learning, Bilevel Optimization, Imperative Learning

1 Introduction

With the rapid development of deep learning (LeCun et al. 2015), there has been growing interest in data-driven approaches such as reinforcement learning (Zhu and Zhang 2021) and imitation learning (Hussein et al. 2017) for robot autonomy. However, despite these notable advancements, many data-driven autonomous systems are still predominantly constrained to their training environments, exhibiting limited generalization ability (Banino et al. 2018; Albrecht et al. 2022). As a comparison, humans are capable of internalizing their experiences as abstract concepts or symbolic knowledge (Borghi et al. 2017). For instance, we interpret the terms "road" and "path" as symbols or concepts for navigable areas, whether it's a paved street in a city or a dirt trail through a forest (Hockley 2011). Equipped with these concepts, humans can employ spatial reasoning to navigate new and complex environments (Strader et al. 2024). This ability to apply spatial reasoning in unfamiliar contexts is a hallmark of human intelligence, one that remains largely missing from current data-driven autonomous robots (Garcez et al. 2022; Kautz 2022).

Though implicit reasoning abilities have garnered increased attention in the context of large language models (LLMs) (Lu et al. 2023; Shah et al. 2023a), robot autonomy systems still encounter significant challenges in achieving interpretable reasoning. This is particularly evident in tasks involving geometric, physical, and logical reasoning (Liu et al. 2023). Overcoming these obstacles and integrating interpretable symbolic reasoning into data-driven models, a direction known as neuro-symbolic (NeSy) reasoning, could remarkably enhance robot autonomy (Garcez et al. 2022).

While NeSy reasoning offers substantial potential, its specific application in the field of robotics is still in a nascent stage. A key reason for this is the emerging nature of the NeSy reasoning field itself, which has not yet reached a consensus on a rigorous definition (Kautz 2022). On one side, a narrow definition views NeSy reasoning as an amalgamation of neural methods (data-driven) and symbolic methods, which utilize formal logic and symbols for knowledge representation and rule-based reasoning. Alternatively, the broader definition expands the scope of what constitutes a "symbol". In this perspective, symbols are not only logical terms but also encompass any comprehensible, human-conceived concepts. This can include physical properties and semantic attributes, as seen in approaches involving concrete equations (Duruisseaux et al. 2023), logical programming (Delfosse et al. 2023), and programmable objectives (Yonetani et al. 2021a). This broad definition hence supports reasoning in both continuous space

[1]University at Buffalo, USA
[2]Pennsylvania State University, USA
[3]Carnegie Mellon University, USA
[4]ETH Zürich, Switzerland
[5]Purdue University, USA
This work was partially funded by the DARPA award HR00112490426.

Corresponding author:
Chen Wang, Spatial AI & Robotics (SAIR) Lab, Department of Computer Science and Engineering, University at Buffalo, Buffalo, NY 14260, USA.
Email: chenw@sairlab.org

(physical reasoning) and discrete space (logical reasoning). In this context, many literatures exemplified NeSy systems, e.g., model-based reinforcement learning (Moerland et al. 2023), physics-informed networks (Karniadakis et al. 2021), and learning-aided tasks such as control (O'Connell et al. 2022), task scheduling (Gondhi and Gupta 2017), and geometry analytics (Heidari and Iosifidis 2024).

In this article, we explore the broader definition of NeSy reasoning for robot autonomy and introduce a self-supervised NeSy learning framework, which will be referred to as *imperative learning* (IL). It is designed to be a unified framework following three principles: **(1) Data-driven and Scalable: Self-Supervised**. Large data-driven models have demonstrated promising performance in robotics. However, labeling data for robotic tasks is often significantly more costly than in computer vision, as it typically requires specialized equipment rather than simple human annotations (Ebadi et al. 2022). For instance, generating accurate ground truth for robot planning is particularly challenging due to the complexity of robot dynamics. As a result, to ensure the scalability of data-driven approaches in robot autonomy, the development of efficient self-supervised learning frameworks is essential. **(2) Generalizable and Interpretable: Neuro-Symbolic Learning with Physical and Logical Structure**. Robotic tasks often involve underlying structured knowledge governed by both physical laws and logical relationships (Dulac-Arnold et al. 2021), such as kinematic and logical constraints, task preconditions, and safety rules. Rather than relying purely on opaque neural models, IL integrates neural learning with symbolic reasoning components that encode these domain-specific structures. This hybrid design enables the system to make decisions that are easier to interpret, while also improving generalization to novel tasks and environments by leveraging prior knowledge about how the world works. **(3) Modular and Global Optimal: End-to-end Trainable**. A modular design reduces complexity by decomposing a system into independent yet interconnected components, making development, debugging, and interpretation more manageable. However, training neural and symbolic modules separately can lead to suboptimal integration, as errors may propagate and accumulate across components (Besold et al. 2021). To overcome this limitation, IL adopts a modular yet end-to-end trainable architecture, retaining the interpretability and flexibility of modular design, while enabling joint optimization across the entire pipeline.

IL is designed to embody the three core principles within a unified framework, inspired by a key observation: both neural models and symbolic methods can be formulated as optimization problems. The key distinction lies in their supervision requirements: neural models typically rely on labeled data for training, whereas symbolic reasoning models can often operate without explicit supervision. For instance, symbolic methods such as bundle adjustment (BA) for geometric reasoning, model predictive control (MPC) for physical reasoning, and A* search for logical reasoning can all be framed as optimization problems and solved without labeled data. IL leverages this property by enabling neural and symbolic components to mutually refine one another. To achieve this, IL is formulated as a special bilevel optimization (BLO), where gradients are back-propagated from self-supervised symbolic modules into the neural networks. This design creates a novel self-supervised learning paradigm, in which symbolic reasoning imperatively guides the learning process. The term "imperative" is thus used to emphasize this passive form of self-supervised, neuro-symbolic, and end-to-end learning.

In summary, the contribution of this article include

- We explore a self-supervised NeSy learning framework, which is referred to as imperative learning (IL) for robot autonomy. IL is formulated as a special BLO problem to enforce the network to learn symbolic concepts and enhance symbolic reasoning via data-driven methods. This leads to a reciprocal learning paradigm, which can avoid the sub-optimal solutions caused by composed errors of decoupled systems.
- We discuss several optimization strategies to tackle the technical challenges of IL. We present how to incorporate different optimization techniques into our IL framework including closed-form solution, first-order optimization, second-order optimization, constrained optimization, and discrete optimization.
- To benefit the entire community, we demonstrated the effectiveness of IL for several tasks in robot autonomy including path planning, rule induction, optimal control, visual odometry, and multi-robot routing. We released the source code at https://sairlab.org/iseries/ to inspire more robotics research using IL.

This article is inspired by and built on our previous works in different fields of robot autonomy including local planning (Yang et al. 2023), global planning (Chen et al. 2024), simultaneous localization and mapping (SLAM) (Fu et al. 2024), feature matching (Zhan et al. 2024), and multi-agent routing (Guo et al. 2024). These works introduced a prototype of IL in several different domains while failing to introduce a systematic framework for NeSy learning in robot autonomy. This article fills this gap by formally defining IL, exploring various optimization challenges of IL in different robot autonomy tasks, and introducing new applications such as rule induction and optimal control. Additionally, we introduce the theoretical background for solving IL based on BLO, propose several practical solutions to IL by experimenting with the five distinct robot autonomy applications, and demonstrate the superiority of IL over state-of-the-art (SOTA) methods in their respective fields.

2 Related Works

2.1 Bilevel Optimization

Bilevel optimization (BLO), first introduced by Bracken and McGill (1973), has been studied for decades. Classic approaches replaced the lower-level problem with its optimality conditions as constraints and reformulated the bilevel programming into a single-level constrained problem (Hansen et al. 1992; Gould et al. 2016; Shi et al. 2005; Sinha et al. 2017). More recently, gradient-based BLO has attracted significant attention due to its efficiency and effectiveness in modern machine learning and deep learning problems. Since this paper mainly focuses on the learning side, we will concentrate on gradient-based BLO methods, and briefly discuss their limitations in robot learning problems.

Methodologically, gradient-based BLO can be generally divided into approximate implicit differentiation (AID), iterative differentiation (ITD), and value-function-based approaches. Based on the explicit form of the gradient (or hypergradient) of the upper-level objective function via implicit differentiation, AID-based methods adopt a generalized iterative solver for the lower-level problem as well as an efficient estimate for Hessian-inverse-vector product of the hypergradient (Domke 2012; Pedregosa 2016; Liao et al. 2018; Arbel and Mairal 2022a). ITD-based methods approximate the hypergradient by directly taking backpropagation over a flexible inner-loop optimization trajectory using forward or backward mode of autograd (Maclaurin et al. 2015; Franceschi et al. 2017; Finn et al. 2017; Shaban et al. 2019; Grazzi et al. 2020). Value-function-based approaches reformulated the lower-level problem as a value-function-based constraint and solved this constrained problem via various constrained optimization techniques such as mixed gradient aggregation, log-barrier regularization, primal-dual method, and dynamic barrier (Sabach and Shtern 2017; Liu et al. 2020b; Li et al. 2020a; Sow et al. 2022; Liu et al. 2021b; Ye et al. 2022). Recently, large-scale stochastic BLO has been extensively studied both in theory and in practice. For example, Chen et al. (2021) and Ji et al. (2021) proposed a Neumann series-based hypergradient estimator; Yang et al. (2021), Huang and Huang (2021), Guo and Yang (2021), Yang et al. (2021), and Dagréou et al. (2022) incorporated the strategies of variance reduction and recursive momentum; and Sow et al. (2021) developed an evolutionary strategies (ES)-based method without computing Hessian or Jacobian.

Theoretically, the convergence of BLO has been analyzed extensively based on a key assumption that the lower-level problem is strongly convex (Franceschi et al. 2018; Shaban et al. 2019; Liu et al. 2021b; Ghadimi and Wang 2018; Ji et al. 2021; Hong et al. 2020; Arbel and Mairal 2022a; Dagréou et al. 2022; Ji et al. 2022a; Huang et al. 2022). Among them, Ji and Liang (2021) further provided lower complexity bounds for deterministic BLO with (strongly) convex upper-level functions. Guo and Yang (2021), Chen et al. (2021), Yang et al. (2021), and Khanduri et al. (2021) achieved a near-optimal sample complexity with second-order derivatives. Kwon et al. (2023); Yang et al. (2024) analyzed the convergence of first-order stochastic BLO algorithms. Recent works studied a more challenging setting where the lower-level problem is convex or satisfies Polyak-Lojasiewicz (PL) or Morse-Bott conditions (Liu et al. 2020b; Li et al. 2020a; Sow et al. 2022; Liu et al. 2021b; Ye et al. 2022; Arbel and Mairal 2022b; Chen et al. 2023; Liu et al. 2021c). More results on BLO and its analysis can be found in the survey (Liu et al. 2021a; Chen et al. 2022).

BLO has been integrated into machine learning applications. For example, researchers have introduced differentiable optimization layers (Amos and Kolter 2017), convex layers (Agrawal et al. 2019), and declarative layers (Gould et al. 2021) into deep neural networks. They have been applied to several applications such as optical flow (Jiang et al. 2020), pivoting manipulation (Shirai et al. 2022), control (Landry 2021), and trajectory generation (Han et al. 2024). However, systematic approaches and methodologies to NeSy learning for robot autonomy remain under-explored. Moreover, robotics problems are often highly non-convex, leading to many local minima and saddle points (Jadbabaie et al. 2019), adding optimization difficulties. We will explore the methods with assured convergence as well as those empirically validated by various tasks in robot autonomy.

2.2 Robot Learning Frameworks

We summarize the major robot learning frameworks, including imitation learning, reinforcement learning, and meta-learning. The others will be briefly mentioned.

Imitation Learning is a technique where robots learn tasks by observing and mimicking an expert's actions. Without explicitly modeling complex behaviors, robots can perform a variety of tasks such as dexterous manipulation (McAleer et al. 2018), navigation (Triest et al. 2023), and environmental interaction (Chi et al. 2023). Current research includes leveraging historical data, modeling multi-modal behaviors, employing privileged teachers (Kaufmann et al. 2020; Chen et al. 2020a; Lee et al. 2020), and utilizing generative models to generate data like generative adversarial networks (Ho and Ermon 2016), variational autoencoders (Zhao et al. 2023), and diffusion models (Chi et al. 2023). These advancements highlight the vibrant and ongoing exploration of imitation learning.

Imitation learning differs from regular supervised learning as it does not assume that the collected data is independent and identically distributed (iid), and relies solely on expert data representing "good" behaviors. Therefore, any small mistake during testing can lead to cascading failures. While techniques such as introducing intentional errors for data augmentation (Pomerleau 1988; Tagliabue et al. 2022; Codevilla et al. 2018) and expert querying for data aggregation (Ross et al. 2011) exist, they still face notable challenges. These include low data efficiency, where limited or suboptimal demonstrations impair performance, and poor generalization, where robots have difficulty adapting learned behaviors to new contexts or unseen variations due to the labor-intensive nature of collecting high-quality data.

Reinforcement learning (RL) is a learning paradigm where robots learn to perform tasks by interacting with their environment and receiving feedback in the form of rewards or penalties (Li 2017). Due to its adaptability and effectiveness, RL has been widely used in numerous fields such as navigation (Zhu and Zhang 2021), manipulation (Gu et al. 2016), locomotion (Margolis et al. 2024), and human-robot interaction (Modares et al. 2015).

However, RL also faces significant challenges, including sample inefficiency, which requires extensive interaction data (Dulac-Arnold et al. 2019), and the difficulty of ensuring safe exploration in physical environments (Thananjeyan et al. 2021). These issues are severe in complex tasks or environments where data collection is forbidden or dangerous (Pecka and Svoboda 2014). Additionally, RL often struggles with generalizing learned behaviors to new environments and tasks and faces significant sim-to-real challenges. It can also be computationally expensive and sensitive to hyperparameter choices (Dulac-Arnold et al. 2021). Moreover, reward shaping, while potentially accelerating learning, can inadvertently introduce biases or suboptimal policies by misguiding the learning process.

We notice that BLO has also been integrated into RL. For instance, Stadie et al. (2020) formulated the intrinsic rewards as a BLO problem, leading to hyperparameter optimization; Hu et al. (2024) integrated reinforcement and imitation learning under BLO, addressing challenges like coupled behaviors and incomplete information in multi-robot coordination; Zhang et al. (2020a) employed a bilevel actor-critic learning method based on BLO and achieved better convergence than Nash equilibrium in the cooperative environments. However, they are still within the framework of RL and no systematic methods have been proposed.

Meta-learning has garnered significant attention recently, particularly with its application to training deep neural networks (Bengio et al. 1991; Thrun and Pratt 2012). Unlike conventional learning approaches, meta-learning leverages datasets and prior knowledge of a task ensemble to rapidly learn new tasks, often with minimal data, as seen in few-shot learning. Numerous meta-learning algorithms have been developed, encompassing metric-based (Koch et al. 2015; Snell et al. 2017; Chen et al. 2020b; Tang et al. 2020; Gharoun et al. 2023), model-based (Munkhdalai and Yu 2017; Vinyals et al. 2016; Liu et al. 2020c; Co-Reyes et al. 2021), and optimization-based methods (Finn et al. 2017; Nichol and Schulman 2018; Simon et al. 2020; Singh et al. 2021; Bohdal et al. 2021; Zhang et al. 2024; Choe et al. 2024). Among them, optimization-based approaches are often simpler to implementation. They are achieving state-of-the-art results in a variety of domains.

BLO has served as an algorithmic framework for optimization-based meta-learning. As the most representative optimization-based approach, model-agnostic meta-learning (MAML) (Finn et al. 2017) learns an initialization such that a gradient descent procedure starting from this initial model can achieve rapid adaptation. In subsequent years, numerous works on various MAML variants have been proposed (Grant et al. 2018; Finn et al. 2019, 2018; Jerfel et al. 2018; Mi et al. 2019; Liu et al. 2019; Rothfuss et al. 2019; Foerster et al. 2018; Baik et al. 2020b; Raghu et al. 2019; Bohdal et al. 2021; Zhou et al. 2021; Baik et al. 2020a; Abbas et al. 2022; Kang et al. 2023; Zhang et al. 2024; Choe et al. 2024). Among them, Raghu et al. (2019) presents an efficient MAML variant named ANIL, which adapts only a subset of the neural network parameters. Finn et al. (2019) introduces a follow-the-meta-leader version of MAML for online learning applications. Zhou et al. (2021) improved the generalization performance of MAML by leveraging the similarity information among tasks. Baik et al. (2020a) proposed an improved version of MAML via adaptive learning rate and weight decay coefficients. Kang et al. (2023) proposed geometry-adaptive pre-conditioned gradient descent for efficient meta-learning. Additionally, a group of meta-regularization approaches has been proposed to improve the bias in a regularized empirical risk minimization problem (Denevi et al. 2018b, 2019, 2018a; Rajeswaran et al. 2019; Balcan et al. 2019; Zhou et al. 2019). Furthermore, there is a prevalent embedding-based framework in few-shot learning (Bertinetto et al. 2018; Lee et al. 2019; Ravi and Larochelle 2016; Snell et al. 2017; Zhou et al. 2018; Goldblum et al. 2020; Denevi et al. 2022; Qin et al. 2023; Jia and Zhang 2024). The objective of this framework is to learn a shared embedding model applicable across all tasks, with task-specific parameters being learned for each task based on the embedded features.

It is worth noting that IL is proposed to alleviate the drawbacks of the above learning frameworks in robotics. However, IL can also be integrated with any existing learning framework, e.g., formulating an RL method as the upper-level problem of IL, although out of the scope of this paper.

2.3 Neuro-symbolic Learning

As previously mentioned, the field of NeSy learning lacks a consensus on a rigorous definition. One consequence is that the literature on NeSy learning is scarce and lacks systematic methodologies. Therefore, we will briefly discuss two major categories: logical reasoning and physics-infused networks. This will encompass scenarios where symbols represent either discrete signals, such as logical constructs, or continuous signals, such as physical attributes. We will address other related work in the context of the five robot autonomy examples within their respective sections.

Logical reasoning aims to inject interpretable and deterministic logical rules into neural networks (Serafini and Garcez 2016; Riegel et al. 2020; Xie et al. 2019; Ignatiev et al. 2018). Some previous work directly obtained such knowledge from human expert (Xu et al. 2018; Xie et al. 2019, 2021; Manhaeve et al. 2018; Riegel et al. 2020; Yang et al. 2020) or an oracle for controllable deduction in neural networks (Mao et al. 2018; Wang et al. 2022; Hsu et al. 2023), refered as *deductive* methods. Representative works include DeepProbLog (Manhaeve et al. 2018), logical neural network (Riegel et al. 2020), and semantic Loss (Xu et al. 2018). Despite their success, deductive methods require structured formal symbolic knowledge from humans, which is not always available. Besides, they still suffer from scaling to larger and more complex problems. In contrast, *inductive* approaches induct the structured symbolic representation for semi-supervised efficient network learning. One popular strategy is based on forward-searching algorithms (Li et al. 2020b, 2022b; Evans and Grefenstette 2018; Sen et al. 2022), which is time-consuming and hard to scale up. Others borrow gradient-based neural networks to conduct rule induction, such as SATNet (Wang et al. 2019), NeuralLP (Yang et al. 2017), and neural logic machine (NLM) (Dong et al. 2019). In particular, NLM introduced a new network architecture inspired by first-order logic, which shows better compositional generalization capability than vanilla neural nets. However, existing inductive algorithms either work on structured data like knowledge graphs (Yang et al. 2017; Yang and Song 2019) or only experiment with toy image datasets (Shindo et al. 2023; Wang et al. 2019). We push the limit of this research into robotics with IL, providing real-world applications with high-dimensional image data.

Physical reasoning in NeSy learning refers to approaches that integrate physical knowledge into the learning process to improve interpretability, generalization, and safety. Two major paradigms in this space are physics-informed and physics-infused learning. Physics-informed learning typically incorporates physical laws into the training objective by adding physics-based loss terms or constraints, guiding neural networks to produce physically consistent outputs (Raissi et al. 2019; Karniadakis et al. 2021). For

example, in fluid dynamics, the Navier–Stokes equations can be enforced as soft penalties to regularize flow predictions (Sun and Wang 2020), and in structural mechanics, variational formulations are used to encode stress-strain behavior (Rao et al. 2021). In contrast, physics-infused learning embeds physical laws or physics-based models directly into the model architecture (Vargas Venegas and Huang 2023; Martin-Linares et al. 2023). For instance, Romero et al. (2023); Han et al. (2024) incorporated differentiable trajectory optimization into learning-based planning and control pipelines, allowing end-to-end learning with explicit physical reasoning; Zhao et al. (2024) programmed energy conservation laws into the model architecture for motion prediction in off-road driving. These methods tightly couple data-driven components with physical structure, enabling more expressive and interpretable models that can propagate gradients through physical interactions. We aim to incorporate physical reasoning via differentiable optimization within a unified neuro-symbolic learning framework for robot autonomy.

3 Imperative Learning

3.1 Structure

The proposed framework of imperative learning (IL) is illustrated in Figure 1, which consists of three modules, i.e., a neural learning system, a symbolic reasoning engine, and a memory module. Specifically, the neural system extracts high-level semantic attributes from raw sensor data such as images, LiDAR points, IMU measurements, and their combinations. These semantic attributes are subsequently sent to the reasoning engine, a symbolic process represented by physical principles, logical inference, analytical geometry, etc. The memory module stores the robot's experiences and acquired knowledge, such as data, symbolic rules, and maps about the physical world for either a long-term or short-term period. Additionally, the reasoning engine performs a consistency check with the memory, which will update the memory or make necessary self-corrections. Intuitively, this design has the potential to combine the expressive feature extraction capabilities from the neural system, the interpretability and the generalization ability from the reasoning engine, and the memorability from the memory module into a single framework. We next explain the mathematical rationale to achieve this.

3.2 Formulation

One of the most important properties of the framework in Figure 1 is that the neural, reasoning, and memory modules can perform reciprocal learning in a self-supervised manner. This is achieved by formulating this framework as a BLO. Denote the neural system as $\boldsymbol{z} = f(\boldsymbol{\theta}, \boldsymbol{x})$, where $\boldsymbol{x}$ represents the sensor measurements, $\boldsymbol{\theta}$ represents the perception-related learnable parameters, and $\boldsymbol{z}$ represents the neural outputs such as semantic attributes; the reasoning engine as $g(f, M, \boldsymbol{\mu})$ with reasoning-related parameters $\boldsymbol{\mu}$ and the memory system as $M(\boldsymbol{\gamma}, \boldsymbol{\nu})$, where $\boldsymbol{\gamma}$ is perception-related memory parameters (Wang et al. 2021a) and $\boldsymbol{\nu}$ is reasoning-related memory parameters. In this context, our

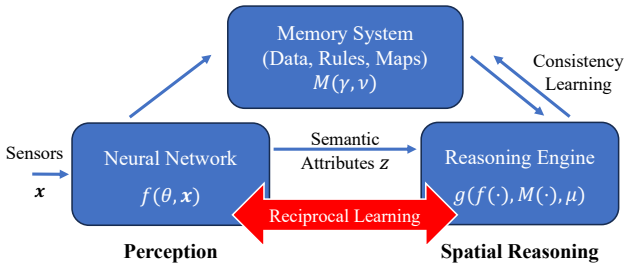


Figure 1. The framework of imperative learning (IL) consists of three primary modules including a neural perceptual network, a symbolic reasoning engine, and a general memory system. IL is formulated as a special BLO, enabling reciprocal learning and mutual correction among the three modules.

imperative learning (IL) is formulated as a special BLO:

$$\min_{\boldsymbol{\psi} \doteq [\boldsymbol{\theta}^\top, \boldsymbol{\gamma}^\top]^\top} U\left(f(\boldsymbol{\theta}, \boldsymbol{x}), g(\boldsymbol{\mu}^*), M(\boldsymbol{\gamma}, \boldsymbol{\nu}^*)\right), \tag{1a}$$

$$\text{s.t.} \quad \boldsymbol{\phi}^* \in \underset{\boldsymbol{\phi} \doteq [\boldsymbol{\mu}^\top, \boldsymbol{\nu}^\top]^\top}{\arg\min} L(f(\boldsymbol{\theta}, \boldsymbol{x}), g(\boldsymbol{\mu}), M(\boldsymbol{\gamma}, \boldsymbol{\nu})), \tag{1b}$$

$$\text{s.t.} \quad \xi(M(\boldsymbol{\gamma}, \boldsymbol{\nu}), \boldsymbol{\mu}, f(\boldsymbol{\theta}, \boldsymbol{x})) = \text{ or } \leq 0, \tag{1c}$$

where ξ is a general constraint (either equality or inequality); U and L are the upper-level (UL) and lower-level (LL) cost functions; and $\boldsymbol{\psi} \doteq [\boldsymbol{\theta}^\top, \boldsymbol{\gamma}^\top]^\top$ are stacked UL variables and $\boldsymbol{\phi} \doteq [\boldsymbol{\mu}^\top, \boldsymbol{\nu}^\top]^\top$ are stacked LL variables, respectively. Alternatively, U and L are also referred to as the *neural cost* and *symbolic cost*, respectively. As aforementioned, the term "*imperative*" is used to denote the passive nature of the learning process: once optimized, the neural system f in the UL cost will be driven to align with the LL reasoning engine g (e.g., logical, physical, or geometrical reasoning process) with constraint ξ, so that it can learn to generate logically, physically, or geometrically feasible semantic attributes or predicates. In some applications, $\boldsymbol{\psi}$ and $\boldsymbol{\phi}$ are also referred to as *neuron-like* and *symbol-like* parameters, respectively.

Self-supervised Learning As presented in Section 1, the formulation of IL is motivated by one important observation: many symbolic reasoning engines including geometric, physical, and logical reasoning, can be optimized or solved without providing labels. This is evident in methods like logic reasoning like equation discovery (Billard and Diday 2002) and A* search (Hart et al. 1968), geometrical reasoning such as bundle adjustment (BA) (Agarwal et al. 2010), and physical reasoning like model predictive control (Kouvaritakis and Cannon 2016). The IL framework leverages this phenomenon and jointly optimizes the three modules by BLO, which enforces the three modules to mutually correct each other. Consequently, all three modules can learn and evolve in a self-supervised manner by observing the world. However, it is worth noting that, although IL is designed for self-supervised learning, it can easily adapt to supervised or weakly supervised learning by involving labels either in UL or LL cost functions or both.

Memory The memory system within the IL framework is a general component that can retain and retrieve information online. Specifically, it can be any structure that is associated with `write` and `read` operations to retain and retrieve data (Wang et al. 2021a). A memory can be a neural network, where information is "written" into the parameters and is

Algorithm 1 The Algorithm of Solving Imperative Learning by Unrolled Differentiation or Implicit Differentiation.

1: **while** Not Convergent **do**
2: Obtain $\boldsymbol{\mu}_T, \boldsymbol{\nu}_T$ by solving LL problem (1b) (possibly with constraint (1c)) by a generic optimizer $\mathscr{O}$ with T steps.
3: Efficient estimation of UL gradients in (2) via
Unrolled Differentiation: Compute $\hat{\nabla}_{\boldsymbol{\theta}} U$ and $\hat{\nabla}_{\boldsymbol{\gamma}} U$ by having $\boldsymbol{\Phi}$ in (7) and $U(\boldsymbol{\psi}, \boldsymbol{\Phi}(\boldsymbol{\psi}))$ in (8) via AutoDiff.
Implicit Differentiation (Algorithm 2): Compute

$$\hat{\nabla}_{\boldsymbol{\psi}} U = \frac{\partial U}{\partial \boldsymbol{\psi}}\Big|_{\boldsymbol{\phi}_T} + \frac{\partial U}{\partial \boldsymbol{\phi}^*}\frac{\partial \boldsymbol{\phi}^*}{\partial \boldsymbol{\psi}}\Big|_{\boldsymbol{\phi}_T},$$

where the implicit derivatives $\frac{\partial \boldsymbol{\phi}^*}{\partial \boldsymbol{\psi}}$ can be obtained by solving a linear system via the LL optimality conditions.
4: **end while**

"read" through a set of math operations or implicit mapping, e.g., a neural radiance fields (NeRF) model (Mildenhall et al. 2021); It can also be a structure with explicit physical meanings such as a map created online, a set of logical rules inducted online, or even a dataset collected online; It can also be the memory system of LLMs in textual form, such as retrieval-augmented generation (RAG) (Lewis et al. 2020), which writes, reads, and manages symbolic knowledge. Hyperdimensional computing (Kleyko et al. 2022, 2023) also presents a promising direction for memory modeling, though it falls outside the scope of this work. In this article, we will explore several customized memory structures tailored to specific applications, as detailed in Section 4.

3.3 Optimization

BLO has been explored in frameworks such as meta-learning (Finn et al. 2017), hyperparameter optimization (Franceschi et al. 2018), and reinforcement learning (Hong et al. 2020). However, most of the theoretical analyses have primarily focused on their applicability to data-driven models, where 1st-order gradient descent (GD) is frequently employed (Ji et al. 2021; Gould et al. 2021). Nevertheless, many reasoning tasks present unique challenges that make GD less effective. For example, geometrical reasoning like BA requires 2nd-order optimizers (Fu et al. 2024) such as Levenberg-Marquardt (LM) (Marquardt 1963); multi-robot routing needs combinatorial optimization over discrete variables (Ren et al. 2023a). Employing such LL optimizations within the BLO framework introduces extreme complexities and challenges, which are still underexplored (Ji et al. 2021). Therefore, we will first delve into general BLO and then provide practical examples covering distinct challenges of LL optimizations in our IL framework.

The solution to IL (1) mainly involves solving the UL parameters $\boldsymbol{\theta}$ and $\boldsymbol{\gamma}$ and the LL parameters $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$. Intuitively, the UL parameters which are often neuron-like weights can be updated with the gradients of the UL cost U:

$$\begin{aligned}\nabla_{\boldsymbol{\theta}} U &= \frac{\partial U}{\partial f}\frac{\partial f}{\partial \boldsymbol{\theta}} + \frac{\partial U}{\partial g}\frac{\partial g}{\partial \boldsymbol{\mu}^*}\frac{\partial \boldsymbol{\mu}^*}{\partial \boldsymbol{\theta}} + \frac{\partial U}{\partial M}\frac{\partial M}{\partial \boldsymbol{\nu}^*}\frac{\partial \boldsymbol{\nu}^*}{\partial \boldsymbol{\theta}},\\ \nabla_{\boldsymbol{\gamma}} U &= \frac{\partial U}{\partial M}\frac{\partial M}{\partial \boldsymbol{\gamma}} + \frac{\partial U}{\partial g}\frac{\partial g}{\partial \boldsymbol{\mu}^*}\frac{\partial \boldsymbol{\mu}^*}{\partial \boldsymbol{\gamma}} + \frac{\partial U}{\partial M}\frac{\partial M}{\partial \boldsymbol{\nu}^*}\frac{\partial \boldsymbol{\nu}^*}{\partial \boldsymbol{\gamma}}.\end{aligned} \tag{2}$$

The key challenge of computing (2) is the implicit differentiation parts in blue fonts, which take the forms of

$$\frac{\partial \boldsymbol{\phi}^*}{\partial \boldsymbol{\psi}} = \begin{bmatrix} \frac{\partial \boldsymbol{\mu}^*}{\partial \boldsymbol{\theta}} & \frac{\partial \boldsymbol{\mu}^*}{\partial \boldsymbol{\gamma}} \\ \frac{\partial \boldsymbol{\nu}^*}{\partial \boldsymbol{\theta}} & \frac{\partial \boldsymbol{\nu}^*}{\partial \boldsymbol{\gamma}} \end{bmatrix}, \tag{3}$$

where $\boldsymbol{\mu}^*$ and $\boldsymbol{\nu}^*$ are the solutions to the LL problem. For simplicity, we can write (2) into the matrix form.

$$\nabla_{\boldsymbol{\psi}} U = \frac{\partial U}{\partial \boldsymbol{\psi}} + \frac{\partial U}{\partial \boldsymbol{\phi}^*}\frac{\partial \boldsymbol{\phi}^*}{\partial \boldsymbol{\psi}}, \tag{4}$$

where $\frac{\partial U}{\partial \boldsymbol{\psi}} = \left[(\frac{\partial U}{\partial f}\frac{\partial f}{\partial \boldsymbol{\theta}})^\top, (\frac{\partial U}{\partial M}\frac{\partial M}{\partial \boldsymbol{\gamma}})^\top\right]^\top$ and $\frac{\partial U}{\partial \boldsymbol{\phi}^*} = \left[(\frac{\partial U}{\partial g}\frac{\partial g}{\partial \boldsymbol{\mu}^*})^\top, (\frac{\partial U}{\partial M}\frac{\partial M}{\partial \boldsymbol{\nu}^*})^\top\right]^\top$. There are typically two methods for calculating those gradients, i.e., *unrolled differentiation* and *implicit differentiation*. We summarize a generic framework incorporating both methods in Algorithm 1 to provide a clearer understanding.

3.3.1 Unrolled Differentiation The method of unrolled differentiation is an easy-to-implement solution for BLO problems. It needs automatic differentiation (AutoDiff) through LL optimization. Specifically, given an initialization for LL variable $\boldsymbol{\phi}_0 = \boldsymbol{\Phi}_0(\boldsymbol{\psi})$ at step $t = 0$, the iterative process of unrolled optimization is

$$\boldsymbol{\phi}_t = \boldsymbol{\Phi}_t(\boldsymbol{\phi}_{t-1}; \boldsymbol{\psi}), \quad t = 1, \cdots, T, \tag{5}$$

where $\boldsymbol{\Phi}_t$ denotes an updating scheme based on a specific LL problem and T is the number of iterations. One popular updating scheme is based on the gradient descent:

$$\boldsymbol{\Phi}_t(\boldsymbol{\phi}_{t-1}; \boldsymbol{\psi}) = \boldsymbol{\phi}_{t-1} - \eta \cdot \frac{\partial L(\boldsymbol{\phi}_{t-1}; \boldsymbol{\psi})}{\partial \boldsymbol{\phi}_{t-1}}, \tag{6}$$

where η is a learning rate and the term $\frac{\partial L(\boldsymbol{\phi}_{t-1}; \boldsymbol{\psi})}{\partial \boldsymbol{\phi}_{t-1}}$ can be computed from AutoDiff. Therefore, we can obtain $\nabla_{\boldsymbol{\theta}} U(\boldsymbol{\theta})$, $\nabla_{\boldsymbol{\gamma}} U(\boldsymbol{\gamma})$ by substituting $\boldsymbol{\phi}_T = [\boldsymbol{\mu}_T, \boldsymbol{\nu}_T]$ approximately for $\boldsymbol{\phi}^* = [\boldsymbol{\mu}^*, \boldsymbol{\nu}^*]$ and the full unrolled system is defined as

$$\boldsymbol{\Phi}(\boldsymbol{\psi}) = (\boldsymbol{\Phi}_T \circ \cdots \circ \boldsymbol{\Phi}_1 \circ \boldsymbol{\Phi}_0)(\boldsymbol{\psi}), \tag{7}$$

where the symbol $\circ$ denotes the function composition. Therefore, we instead only need to consider an alternative:

$$\min_{\boldsymbol{\psi}} \; U(\boldsymbol{\psi}, \boldsymbol{\Phi}(\boldsymbol{\psi})), \tag{8}$$

where $\frac{\partial \boldsymbol{\Phi}(\boldsymbol{\psi})}{\partial \boldsymbol{\psi}}$ can be computed via AutoDiff instead of directly calculating the four terms in $\frac{\partial \boldsymbol{\phi}^*}{\partial \boldsymbol{\psi}}$ of (3).

3.3.2 Implicit Differentiation The method of implicit differentiation directly computes the derivatives $\frac{\partial \boldsymbol{\phi}^*}{\partial \boldsymbol{\psi}}$. We next introduce a generic framework for the implicit differentiation algorithm by solving a linear system from the first-order optimality condition of the LL problem, while the exact solutions depend on specific tasks and will be illustrated using several independent examples in Section 4.

Assume $\boldsymbol{\phi}_T = [\boldsymbol{\mu}_T, \boldsymbol{\nu}_T]$ is the outputs of T steps of a generic optimizer of the LL problem (1b) possibly under constraints (1c), then the approximated UL gradient (2) is

$$\hat{\nabla}_{\boldsymbol{\psi}} U \approx \frac{\partial U}{\partial \boldsymbol{\psi}}\Big|_{\boldsymbol{\phi}_T} + \frac{\partial U}{\partial \boldsymbol{\phi}^*}\frac{\partial \boldsymbol{\phi}^*}{\partial \boldsymbol{\psi}}\Big|_{\boldsymbol{\phi}_T}. \tag{11}$$

Then the derivatives $\frac{\partial \boldsymbol{\phi}^*}{\partial \boldsymbol{\psi}}$ are obtained via solving implicit equations from optimality conditions of the LL problem,

Algorithm 2 *Implicit Differentiation* via Linear System.

1: **Input:** The current variable ψ and the optimal variable ϕ^*.
2: **Initialization:** $k=1$, learning rate η.
3: **while** Not Convergent **do**
4: Perform gradient descent (or use conjugate gradient):

$$\boldsymbol{q}_k = \boldsymbol{q}_{k-1} - \eta\left(\boldsymbol{H}\boldsymbol{q} - \boldsymbol{v}\right), \tag{9}$$

where $\boldsymbol{H}\boldsymbol{q}$ is computed from fast Hessian-vector product.
5: **end while**
6: Assign $\boldsymbol{q} = \boldsymbol{q}_k$
7: Compute $\nabla_{\boldsymbol{\theta}} U$ and $\nabla_{\boldsymbol{\gamma}} U$ in (2) as:

$$\nabla_{\psi} U = \frac{\partial U}{\partial \psi} - \boldsymbol{q}^{\top} \cdot \boldsymbol{H}_{\phi^*\psi}, \tag{10}$$

where $\boldsymbol{q}^{\top} \cdot \boldsymbol{H}_{\phi^*\psi}$ is also from the fast Hessian-vector product.

i.e., $\frac{\partial \tilde{L}}{\partial \phi^*} = \mathbf{0}$, where $\tilde{L}$ is a generic LL optimality condition. Specifically, taking the derivative of equation $\frac{\partial \tilde{L}}{\partial \phi^*} = \mathbf{0}$ with respect to ψ on both sides leads to

$$\frac{\partial^2 \tilde{L}(\phi^*(\psi),\psi)}{\partial \phi^*(\psi)\partial \psi} + \frac{\partial^2 \tilde{L}(\phi^*(\psi),\psi)}{\partial \phi^*(\psi)\partial \phi^*(\psi)} \cdot \frac{\partial \phi^*(\psi)}{\partial \psi} = \mathbf{0}. \tag{12}$$

Solving the equation gives us the implicit gradients as

$$\frac{\partial \phi^*(\psi)}{\partial \psi} = -\underbrace{\left(\frac{\partial^2 \tilde{L}(\phi^*(\psi),\psi)}{\partial \phi^*(\psi)\partial \phi^*(\psi)}\right)^{-1}}_{\boldsymbol{H}^{-1}_{\phi^*\phi^*}} \underbrace{\frac{\partial^2 \tilde{L}(\phi^*(\psi),\psi)}{\partial \phi^*(\psi)\partial \psi}}_{\boldsymbol{H}_{\phi^*\psi}}. \tag{13}$$

This means we obtain the implicit gradients at the cost of an inversion of the Hessian matrix $\boldsymbol{H}^{-1}_{\phi^*\phi^*}$.

In practice, the Hessian matrix can be too big to calculate and store[†], but we could bypass it by solving a linear system. Substitute (13) into (4), we have the UL gradient

$$\nabla_{\psi} U = \frac{\partial U}{\partial \psi} - \underbrace{\frac{\partial U(\psi,\phi^*)}{\partial \phi^*}}_{\boldsymbol{v}^{\top}} \boldsymbol{H}^{-1}_{\phi^*\phi^*} \cdot \boldsymbol{H}_{\phi^*\psi} \tag{14a}$$

$$= \frac{\partial U}{\partial \psi} - \boldsymbol{q}^{\top} \cdot \boldsymbol{H}_{\phi^*\psi}. \tag{14b}$$

Therefore, instead of calculating the Hessian inversion, we can solve a linear system $\boldsymbol{H}\boldsymbol{q} = \boldsymbol{v}$ for $\boldsymbol{q}^{\top}$ by optimizing

$$\boldsymbol{q}^* = \arg\min_{\boldsymbol{q}} Q(\boldsymbol{q}) = \arg\min_{\boldsymbol{q}} \left(\frac{1}{2}\boldsymbol{q}^{\top}\boldsymbol{H}\boldsymbol{q} - \boldsymbol{q}^{\top}\boldsymbol{v}\right), \tag{15}$$

where $\boldsymbol{H}_{\phi^*\phi^*}$ is denoted as $\boldsymbol{H}$ for simplicity. The linear system (15) can be solved without explicitly calculating and storing the Hessian matrix by solvers such as conjugate gradient (Hestenes and Stiefel 1952) or gradient descent. For example, due to the gradient $\frac{\partial Q(\boldsymbol{q})}{\partial \boldsymbol{q}} = \boldsymbol{H}\boldsymbol{q} - \boldsymbol{v}$, the updating scheme based on the gradient descent algorithm is:

$$\boldsymbol{q}_k = \boldsymbol{q}_{k-1} - \eta\left(\boldsymbol{H}\boldsymbol{q}_{k-1} - \boldsymbol{v}\right). \tag{16}$$

This updating scheme is efficient since $\boldsymbol{H}\boldsymbol{q}$ can be computed using the fast Hessian-vector product, i.e., a Hessian-vector product is the gradient of a gradient-vector product:

$$\boldsymbol{H}\boldsymbol{q} = \frac{\partial^2 \tilde{L}}{\partial \phi \partial \phi} \cdot \boldsymbol{q} = \frac{\partial\left(\frac{\partial \tilde{L}}{\partial \phi} \cdot \boldsymbol{q}\right)}{\partial \phi}, \tag{17}$$

where $\frac{\partial \tilde{L}}{\partial \phi} \cdot \boldsymbol{q}$ is a scalar. This means the Hessian $\boldsymbol{H}$ is not explicitly computed or stored. We summarize this implicit differentiation in Algorithm 2. Note that the optimality condition depends on the LL problems. In Section 4, we will show that it can either be the derivative of the LL cost function for an unconstrained problem such as (27) or the Lagrangian function for a constrained problem such as (39).

Approximation Implicit differentiation is complicated to implement but there is one approximation, which is to ignore the implicit components and only use the direct part $\hat{\nabla}_{\psi} U \approx \frac{\partial U}{\partial \psi}\Big|_{\phi_T}$. This is equivalent to taking the solution ϕ_T from the LL optimization as constants in the UL problem. Such an approximation is more efficient but introduces an error term

$$\varepsilon \sim \left|\frac{\partial U}{\partial \phi^*}\frac{\partial \phi^*}{\partial \psi}\right|. \tag{18}$$

Nevertheless, it is useful when the implicit derivatives contain products of small second-order derivatives, which again depends on the specific LL problems.

It is worth noting that in the framework of IL (1), we assign perception-related parameters ψ to the UL neural cost (1a), while reasoning-related parameters ϕ to the LL symbolic cost (1b). This design stems from two key considerations: First, it can avoid involving large Jacobian and Hessian matrices for neuron-like variables such as $\boldsymbol{H}_{\phi^*\phi^*}$. Given that real-world robot applications often involve an immense number of neuron-like parameters (e.g., a simple neural network might possess millions), placing them in the UL cost reduces the complexity involved in computing implicit gradients necessitated by the LL cost (1b).

Second, perception-related (neuron-like) parameters are usually updated using gradient descent algorithms such as SGD (Sutskever et al. 2013). However, such simple first-order optimization methods are often inadequate for LL symbolic reasoning, e.g., geometric problems (Wang et al. 2023) usually need second-order optimizers. Therefore, separating neuron-like parameters from reasoning-related parameters makes the BLO in IL easier to solve and analyze. However, this again depends on the LL tasks.

Discussion on theoretical optimality and sufficient conditions. The guarantee of feasible solutions depends on the assumptions and conditions of the objective functions. For example, in unrolled differentiation (Section 3.3.1) and implicit differentiation (Section 3.3.2), if the LL function L in (1b) is twice differentiable and strongly convex with respect to ϕ with a unique solution, we expect to achieve first-order stationarity and global optimality for non-convex and convex UL objective functions, respectively.

The optimality for generic non-convex LL problems remain an open challenge and are sometimes NP-hard (Dempe et al. 2015) in BLO theory. This difficulty arises because non-uniqueness in the LL problem prevents the implicit differentiation theorem from guaranteeing the existence of implicit differentiability of ϕ^* with respect to the UL variables ψ. A weaker condition for establishing optimality is the Polyak-Lojasiewicz (PL) condition (Shen

[†]For instance, assume both UL and LU costs have a network with merely 1 million (10^6) parameters (32-bit float numbers), thus each network only needs a space of $10^6 \times 4\text{Byte} = 4\text{M}B$ to store, while their Hessian matrix needs a space of $(10^6)^2 \times 4\text{Byte} = 4\text{T}B$ to store. This indicates that a Hessian matrix cannot even be explicitly stored in the memory of a low-power computer, thus directly calculating its inversion is more impractical.

Table 1. Five examples in logic, planning, control, SLAM, and MTSP are selected to cover distinct scenarios of the LL problems for solving the BLO in imperative learning.

Section	Application	LL problem	LL solution	Type
Sec. 4.1	Planning	A*	Closed-form	(C)
Sec. 4.2	Logic	NLM	1st-order	(B)*
Sec. 4.3	Control	MPC	Constrained	(A)
Sec. 4.4	SLAM	PGO	2nd-order	(C)
Sec. 4.5	MTSP	TSP	Discrete	(A)

*All modules are trained, although a pre-trained feature extractor is given.

and Chen 2023), which allows for an overall non-convex landscape while maintaining certain convexity-like structures. Such structures have been observed in overparameterized neural networks and can guarantee an ε-approximate solution to the BLO problem (Liu et al. 2020a). We will show more detailed theoretical guarantees and their assumptions in the next section with respect to the types of LL optimization. To ensure the language remaining accessible and not overly theoretical, we will only provide summarized results and provide references to their sources. In this way, we hope it will be helpful for readers to easily find more relevant formal results when needed.

4 Applications and Examples

To showcase the effectiveness of IL, we will introduce five distinct examples in different fields of robot autonomy. These examples, along with their respective LL problem and optimization methods, are outlined in Table 1. Specifically, they are selected to cover distinct tasks, including path planning, rule induction, optimal control, visual odometry, and multi-agent routing, to showcase different optimization techniques required by the LL problems, including closed-form solution, first-order optimization, constrained optimization, second-order optimization, and discrete optimization, respectively. We will explore several memory structures mentioned in Section 3.2.

Additionally, since IL is a self-supervised learning framework consisting of three primary components, we have three kinds of learning types. This includes (**A**) given known (pre-trained or human-defined) reasoning engines such as logical reasoning, physical principles, and geometrical analytics, robots can learn a logic-, physics-, or geometry-aware neural perception system, respectively, in a self-supervised manner; (**B**) given neural perception systems such as a vision foundation model (Kirillov et al. 2023), robots can discover the world rules, e.g., traffic rules, and then apply the rules to future events; and (**C**) given a memory system, (e.g., experience, world rules, or maps), robots can simultaneously update the neural system and reasoning engine so that they can adapt to novel environments with a new set of rules. Note that if one of the modules has a good initialization, all the three modules can be further finetuned simultaneously. The five examples will cover all three learning types.

4.1 Closed-form Solution

We first illustrate the scenarios in which the LL cost L in (1b) has closed-form solutions. In this case, one can directly optimize the UL cost by solving

$$\min_{\boldsymbol{\theta},\boldsymbol{\gamma}} \quad U\left(f(\boldsymbol{\theta},\boldsymbol{x}), g(\boldsymbol{\mu}^*(\boldsymbol{\theta},\boldsymbol{\gamma})), M(\boldsymbol{\gamma},\boldsymbol{\nu}^*(\boldsymbol{\theta},\boldsymbol{\gamma}))\right), \tag{19}$$

where the LL solutions $\boldsymbol{\mu}^*(\boldsymbol{\theta},\boldsymbol{\gamma})$ and $\boldsymbol{\nu}^*(\boldsymbol{\theta},\boldsymbol{\gamma})$, that contain the implicit components $\frac{\partial \boldsymbol{\mu}^*}{\partial \boldsymbol{\theta}}$, $\frac{\partial \boldsymbol{\mu}^*}{\partial \boldsymbol{\gamma}}$, $\frac{\partial \boldsymbol{\nu}^*}{\partial \boldsymbol{\theta}}$, and $\frac{\partial \boldsymbol{\nu}^*}{\partial \boldsymbol{\gamma}}$, can be calculated directly due to the closed-form of $\boldsymbol{\mu}^*$ and $\boldsymbol{\nu}^*$. As a result, standard gradient-based algorithms can be applied in updating $\boldsymbol{\theta}$ and $\boldsymbol{\gamma}$ with gradients given by (2). In this case, there is **no** approximation error induced by the LL minimization, in contrast to existing widely-used implicit and unrolled differentiation methods that require ensuring a sufficiently small or decreasing LL optimization error for guaranteeing the convergence (Ji et al. 2021).

One possible problem in computing (3) is that the implicit components can contain expensive matrix inversions. Let us consider a simplified quadratic case in (1b) with $\boldsymbol{\mu}^* = \arg\min_{\boldsymbol{\mu}} \frac{1}{2}\|f(\boldsymbol{\theta},x)\boldsymbol{\mu} - y\|^2$. The closed-form solution takes the form of $\boldsymbol{\mu}^* = [f(\boldsymbol{\theta},x)^\top f(\boldsymbol{\theta},x)]^{-1}y$, which can induce a computationally expensive inversion of a possibly large matrix $f(\boldsymbol{\theta},x)^\top f(\boldsymbol{\theta},x)$. To address this problem, one can again formulate the matrix-inverse-vector computation $H^{-1}y$ as solving a linear system $\min_v \frac{1}{2}v^\top Hv - v^\top y$ using any optimization methods with efficient matrix-vector products.

Many symbolic costs can be effectively addressed through closed-form solutions. For example, both linear quadratic regulator (LQR) (Shaiju and Petersen 2008) and Dijkstra's algorithm (Dijkstra 1959) can be solved with a determined optimal solution. To demonstrate the effectiveness of IL for closed-form solutions, we next present two examples in path planning to utilize the neural model for reducing the search and sampling space of symbolic optimization.

Example 1: Path Planning

Path planning is a computational process to determine a path from a starting point to a destination within an environment. It typically involves navigating around obstacles and may also optimize certain objectives such as the shortest distance, minimal energy use, or maximum safety. Path planning algorithms are generally categorized into global planning, which utilizes global maps of the environment, and local planning, which relies on real-time sensory data. We will enhance two widely-used algorithms through IL: A* search for global planning and cubic spline for local planning, both of which offer closed-form solutions.

Example 1.A: Global Path Planning

Background The A* algorithm is a graph search technique that aims to find the global shortest feasible path between two nodes (Hart et al. 1968). Specifically, A* selects the path passing through the next node $n \in G$ that minimizes

$$C(n) = s(n) + h(n), \tag{20}$$

where $s(n)$ is the cost from the start node to n and $h(n)$ is a heuristic function that predicts the cheapest path cost from n to the goal. The heuristic cost can take the form of various metrics such as the Euclidean and Chebyshev distances. It can be proved that if the heuristic function $h(n)$ is admissible and monotone, i.e., $\forall m \in G, h(n) \le h(m) + d(m,n)$, where $d(m,n)$ is the cost from m to n, A* is guaranteed to find the optimal path without searching any node more than once

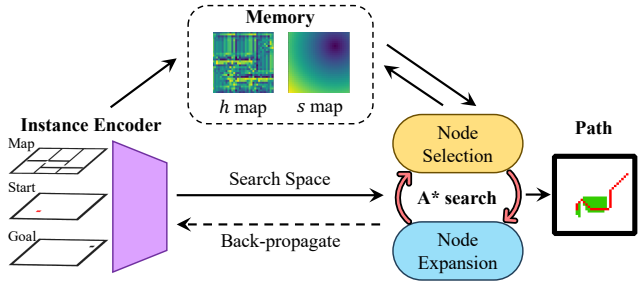


Figure 2. The framework of iA* search. The network predicts a confined search space, leading to overall improved efficiency. The A* search algorithm eliminates the label dependence, resulting in a self-supervised path planning framework.

(Dechter and Pearl 1985). Due to this optimality, A* became one of the most widely used methods in path planning (Paden et al. 2016; Smith et al. 2012; Algfoor et al. 2015).

However, A* encounters significant limitations, particularly in its requirement to explore a large number of potential nodes. This exhaustive search process can be excessively time-consuming, especially for low-power robot systems. To address this, recent advancements showed significant potential for enhancing efficiency by predicting a more accurate heuristic cost map using data-driven methods (Choudhury et al. 2018; Yonetani et al. 2021a; Kirilenko et al. 2023). Nevertheless, these algorithms utilize optimal paths as training labels, which face challenges in generalization, leading to a bias towards the patterns observed in the training datasets.

Approach To address this limitation, we leverage the IL framework to remove dependence on path labeling. Specifically, we utilize a neural network f to estimate the h value of each node, which can be used to reduce the search space. Subsequently, we integrate a differentiable A* module, serving as the symbolic reasoning engine g in (1b), to determine the most efficient path. This results in an effective framework depicted in Figure 2, which we refer to as imperative A* (iA*) algorithm. Notably, the iA* framework can operate on a self-supervised basis inherent in the IL framework, eliminating the need for annotated labels.

Specifically, the iA* algorithm is to minimize

$$\min_{\boldsymbol{\theta}} \quad U\left(f(\boldsymbol{x},\boldsymbol{\theta}), g(\boldsymbol{\mu}^*), M(\boldsymbol{\nu}^*)\right), \tag{21a}$$

$$\text{s.t.} \quad \boldsymbol{\mu}^*, \boldsymbol{\nu}^* = \arg\min_{\boldsymbol{\mu},\boldsymbol{\nu}} L(f, g(\boldsymbol{\mu}), M(\boldsymbol{\nu})), \tag{21b}$$

where $\boldsymbol{x}$ denotes the inputs including the map, start node, and goal node, $\boldsymbol{\mu}$ is the set of paths in the solution space, $\boldsymbol{\nu}$ is the accumulated h values associated with a path in $\boldsymbol{\mu}$, $\boldsymbol{\mu}^*$ is the optimal path, $\boldsymbol{\nu}^*$ is the accumulated h values associated with the optimal path $\boldsymbol{\mu}^*$, and M is the intermediate s and h maps.

The lower-level cost L is defined as the path length (cost)

$$L(\boldsymbol{\mu}, M, g) \doteq C_l(\boldsymbol{\mu}^*, M), \tag{22}$$

where the optimal path $\boldsymbol{\mu}^*$ is derived from the A* reasoning g given the node cost maps M. Thanks to its closed-form solution, this LL optimization is directly solvable. The UL optimization focuses on updating the network parameter $\boldsymbol{\theta}$ to generate the h map. Given the impact of the h map on the search area of A*, the UL cost U is formulated as a combination of the search area $C_a(\boldsymbol{\mu}^*, \boldsymbol{\nu}^*)$ and the path length $C_l(\boldsymbol{\mu}^*)$. This is mathematically represented as:

$$U(\boldsymbol{\mu}^*, M) \doteq w_a C_a(\boldsymbol{\mu}^*, \boldsymbol{\nu}^*) + w_l C_l(\boldsymbol{\mu}^*, M), \tag{23}$$

where w_a and w_l are the weights to adjust the two terms, and C_a is the search area computed from the accumulated h map with the optimal path $\boldsymbol{\mu}^*$. In the experiments, we define the s cost as the Euclidean distance. The input $\boldsymbol{x}$ is represented as a three-channel 2-D tensor, with each channel dedicated to a specific component: the map, the start node, and the goal node. Specifically, the start and goal node channels are represented as a one-hot 2-D tensor, where the indices of the "ones" indicate the locations of the start and goal node, respectively. This facilitates a more nuanced and effective representation of path planning problems.

Optimization As shown in (2), once the gradient calculation is completed, we backpropagate the cost U directly to the network f. This process is facilitated by the closed-form solution of the A* algorithm. For the sake of simplicity, we employ the differentiable A* algorithm as introduced by Yonetani et al. (2021a). It transforms the node selection into an `argsoftmax` operation and reinterprets node expansion as a series of convolutions, leveraging efficient implementations and AutoDiff in PyTorch (Paszke et al. 2019).

Intuitively, this optimization process involves iterative adjustments. On one hand, the h map enables the A* algorithm to efficiently identify the optimal path, but within a confined search area. On the other hand, the A* algorithm's independence from labels allows further refinement of the network. This is achieved by the back-propagating of the search area and length cost through the differentiable A* reasoning. This mutual connection encourages the network to generate increasingly smaller search areas over time, enhancing overall efficiency. As a result, the network inclines to focus on more relevant areas, marked by reduced low-level reasoning costs, improving the overall search quality.

Example 1.B: Local Path Planning

Background End-to-end local planning, which integrates perception and planning within a single model, has recently attracted considerable interest, particularly for its potential to enable efficient inference through data-driven methods such as reinforcement learning (Hoeller et al. 2021; Wijmans et al. 2019; Lee et al. 2024; Ye et al. 2021) and imitation learning (Sadat et al. 2020; Shah et al. 2023b; Loquercio et al. 2021). Despite these advancements, significant challenges persist. Reinforcement learning-based methods often suffer from sample inefficiency and difficulties in directly processing raw, dense sensor inputs, such as depth images. Without sufficient guidance during training, reinforcement learning struggles to converge on an optimal policy that generalizes well across various scenarios or environments. Conversely, imitation learning relies heavily on the availability and quality of labeled trajectories. Obtaining these labeled trajectories is particularly challenging for robotic systems that operate under diverse dynamics models, thereby limiting their broad applicability in flexible robotic systems.

Approach To address these challenges, we introduce IL to local planning and refer to it as imperative local planning (iPlanner), as depicted in Figure 3. Instead of predicting a continuous trajectory directly, iPlanner uses the network

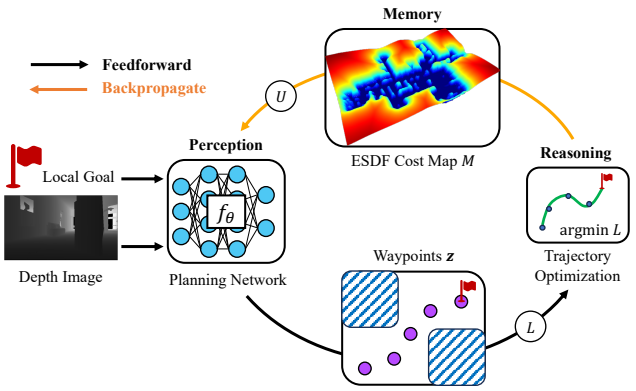


Figure 3. The framework of iPlanner. The higher-level network predicts waypoints, which are interpolated by the lower-level optimization to ensure path continuity and smoothness.

to generate sparse waypoints, which are then interpolated using a trajectory optimization engine based on a cubic spline. This approach leverages the strengths of both neural and symbolic modules: neural networks excel at dynamic obstacle detection, while symbolic modules optimize multi-step navigation strategies under dynamics. By enforcing the network output sparse waypoints rather than continuous trajectories, iPlanner effectively combines the advantages of both modules. Specifically, iPlanner can be formulated as:

$$\min_{\boldsymbol{\theta}} \quad U\left(f(\boldsymbol{x},\boldsymbol{\theta}), g(\boldsymbol{\mu}^*), M(\boldsymbol{\mu}^*)\right), \tag{24a}$$

$$\text{s.t.} \quad \boldsymbol{\mu}^* = \arg\min_{\boldsymbol{\mu}\in\mathbb{T}} L(\boldsymbol{z},\boldsymbol{\mu}), \tag{24b}$$

where $\boldsymbol{x}$ denotes the system inputs including a local goal position and the sensor measurements such as depth images, $\boldsymbol{\theta}$ is the parameters of a network f, $\boldsymbol{z} = f(\boldsymbol{x},\boldsymbol{\theta})$ denotes the generated waypoints, which are subsequently optimized by the path optimizer g, $\boldsymbol{\mu}$ represents the set of valid paths within the constrained space $\mathbb{T}$. The optimized path, $\boldsymbol{\mu}^*$, acts as the optimal solution to the LL cost L, which is defined by tracking the intermediate waypoints and the overall continuity and smoothness of the path:

$$L \doteq D(\boldsymbol{\mu},\boldsymbol{z}) + A(\boldsymbol{\mu}), \tag{25}$$

where $D(\boldsymbol{\mu},\boldsymbol{z})$ measures the path continuity based on the 1st-order derivative of the path and $A(\boldsymbol{u})$ calculates the path smoothness based on 2nd-order derivative. Specifically, we employ the cubic spline interpolation which has a closed-form solution to ensure this continuity and smoothness. This is also essential for generating feasible and efficient paths. On the other hand, the UL cost U is defined as:

$$U \doteq w_G C^G(\boldsymbol{\mu}^*, x_{\text{goal}}) + w_L C^L(\boldsymbol{\mu}^*) + w_M M(\boldsymbol{\mu}^*), \tag{26}$$

where $C^G(\boldsymbol{\mu}^*, x_{\text{goal}})$ measures the distance from the endpoint of the generated path to the goal x_{goal}, assessing the alignment of the planned path with the desired destination; $C^L(\boldsymbol{\mu}^*)$ represents the motion loss, encouraging the planning of shorter paths to improve overall movement efficiency; $M(\boldsymbol{\mu}^*)$ quantifies the obstacle cost, utilizing information from a pre-built Euclidean signed distance fields (ESDF) map to evaluate the path safety; and w_G, w_L, and w_M are hyperparameters, allowing for adjustments in the planning strategy based on specific performance objectives.

Optimization During training, we leverage the AutoDiff capabilities provided by PyTorch (Paszke et al. 2019) to solve the BLO in IL. For the LL trajectory optimization, we adopt the cubic spline interpolation implementation provided by PyPose (Wang et al. 2023). It supports differentiable batched closed-form solutions, enabling a faster training process. Additionally, the ESDF cost map is convoluted with a Gaussian kernel, enabling a smoother optimization space. This setup allows the loss to be directly backpropagated to update the network parameters in a single step, rather than requiring iterative adjustments. As a result, the UL network and the trajectory optimization can mutually improve each other, enabling a self-supervised learning process.

4.2 First-order Optimization

We next illustrate the scenario that the LL cost L in (1b) uses first-order optimizers such as GD. Because GD is a simple differentiable iterative method, one can leverage *unrolled optimization* listed in Algorithm 1 to solve BLO via AutoDiff. It has been theoretically proved by Ji et al. (2021, 2022a) that when the LL problem is strongly convex and smooth, unrolled optimization with LL gradient descent can approximate the hypergradients $\nabla_{\boldsymbol{\theta}}$ and ∇_{γ} up to an error that decreases linearly with the number of GD steps. For the *implicit differentiation*, we could leverage the optimality conditions of the LL optimization to compute the implicit gradient. To be specific, using Chain rule and optimality of $\boldsymbol{\mu}^*$ and $\boldsymbol{\nu}^*$, we have the stationary points

$$\frac{\partial L}{\partial \boldsymbol{\mu}}(\boldsymbol{\mu}^*,\boldsymbol{\nu}^*,\boldsymbol{\theta},\boldsymbol{\gamma}) = \frac{\partial L}{\partial \boldsymbol{\nu}}(\boldsymbol{\mu}^*,\boldsymbol{\nu}^*,\boldsymbol{\theta},\boldsymbol{\gamma}) = 0, \tag{27}$$

which, by taking differentiation over $\boldsymbol{\theta}$ and $\boldsymbol{\gamma}$ and noting the implicit dependence of $\boldsymbol{\mu}^*$ and $\boldsymbol{\nu}^*$ on $\boldsymbol{\theta}$ and $\boldsymbol{\gamma}$, yields

$$\underbrace{\frac{\partial^2 L}{\partial f \partial \boldsymbol{\mu}}\frac{\partial f}{\partial \boldsymbol{\theta}}}_{A} + \underbrace{\frac{\partial^2 L}{\partial \boldsymbol{\mu}^2}}_{B}\frac{\partial \boldsymbol{\mu}^*}{\partial \boldsymbol{\theta}} + \underbrace{\frac{\partial^2 L}{\partial \boldsymbol{\nu}\partial \boldsymbol{\mu}}}_{C}\frac{\partial \boldsymbol{\nu}^*}{\partial \boldsymbol{\theta}} = 0, \tag{28a}$$

$$\underbrace{\frac{\partial^2 L}{\partial f \partial \boldsymbol{\nu}}\frac{\partial f}{\partial \boldsymbol{\theta}}}_{\widetilde{A}} + \underbrace{\frac{\partial^2 L}{\partial \boldsymbol{\nu}^2}}_{\widetilde{B}}\frac{\partial \boldsymbol{\nu}^*}{\partial \boldsymbol{\theta}} + \underbrace{\frac{\partial^2 L}{\partial \boldsymbol{\mu}\partial \boldsymbol{\nu}}}_{\widetilde{C}}\frac{\partial \boldsymbol{\mu}^*}{\partial \boldsymbol{\theta}} = 0. \tag{28b}$$

Assume that the concatenated matrix $\begin{bmatrix} B, C \\ \widetilde{C}, \widetilde{B} \end{bmatrix}$ invertible at $\boldsymbol{\mu}^*,\boldsymbol{\nu}^*,\boldsymbol{\theta},\boldsymbol{\gamma}$. Then, it can be derived from the linear equations in (28a) that the implicit gradients $\frac{\partial \boldsymbol{\mu}^*}{\partial \boldsymbol{\theta}}$ and $\frac{\partial \boldsymbol{\nu}^*}{\partial \boldsymbol{\theta}}$

$$\begin{bmatrix} \frac{\partial \boldsymbol{\mu}^*}{\partial \boldsymbol{\theta}} \\ \frac{\partial \boldsymbol{\nu}^*}{\partial \boldsymbol{\theta}} \end{bmatrix} = -\begin{bmatrix} B, C \\ \widetilde{C}, \widetilde{B} \end{bmatrix}^{-1}\begin{bmatrix} A \\ \widetilde{A} \end{bmatrix}, \tag{29}$$

which, combined with (2), yields the UL gradient $\nabla_{\boldsymbol{\theta}} U$ as

$$\nabla_{\boldsymbol{\theta}} U = \frac{\partial U}{\partial f}\frac{\partial f}{\partial \boldsymbol{\theta}} + \begin{bmatrix} \frac{\partial U}{\partial \boldsymbol{\mu}^*} \\ \frac{\partial U}{\partial \boldsymbol{\nu}^*} \end{bmatrix}^{\top}\begin{bmatrix} \frac{\partial \boldsymbol{\mu}^*}{\partial \boldsymbol{\theta}} \\ \frac{\partial \boldsymbol{\nu}^*}{\partial \boldsymbol{\theta}} \end{bmatrix} \tag{30a}$$

$$= \frac{\partial U}{\partial f}\frac{\partial f}{\partial \boldsymbol{\theta}} - \underbrace{\begin{bmatrix} \frac{\partial U}{\partial \boldsymbol{\mu}^*} \\ \frac{\partial U}{\partial \boldsymbol{\nu}^*} \end{bmatrix}^{\top}\begin{bmatrix} B, C \\ \widetilde{C}, \widetilde{B} \end{bmatrix}^{-1}}_{D}\begin{bmatrix} A \\ \widetilde{A} \end{bmatrix} \tag{30b}$$

where the vector-matrix-inverse-product D can be efficiently approximated by solving a linear system w.r.t. a vector v

$$\min_{v} \quad \frac{1}{2} v^{\top}\begin{bmatrix} B, C \\ \widetilde{C}, \widetilde{B} \end{bmatrix} v + \begin{bmatrix} \frac{\partial U}{\partial \boldsymbol{\mu}^*} \\ \frac{\partial U}{\partial \boldsymbol{\nu}^*} \end{bmatrix}^{\top} v. \tag{31}$$

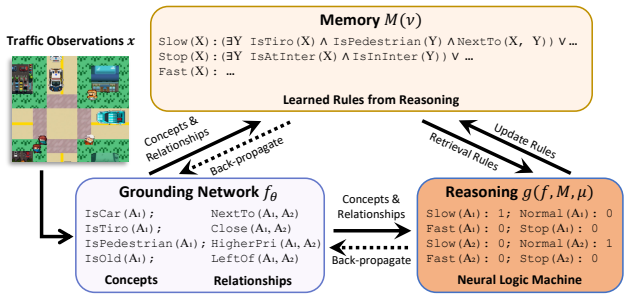


Figure 4. The iLogic pipeline, which simultaneously conducts rule induction and high-dimensional data grounding.

A similar linear system can be derived for computing $\nabla_\gamma U$. Then, the practical methods should first use a first-order optimizer to solve the lower-level problem in (1b) to obtain approximates $\hat{\boldsymbol{\mu}}$ and $\hat{\boldsymbol{\nu}}$ of the solutions $\boldsymbol{\mu}^*$ and $\boldsymbol{\nu}^*$, which are then incorporated into (30a) to obtain an approximate $\widehat{\nabla}_{\boldsymbol{\theta}} U$ of the upper-level gradient $\nabla_{\boldsymbol{\theta}} U$ (similarly for $\widehat{\nabla}_\gamma U$). Then, the upper-level gradient approximates $\widehat{\nabla}_{\boldsymbol{\theta}} U$ and $\widehat{\nabla}_\gamma U$ are used to optimize the target variables $\boldsymbol{\theta}$ and γ.

Approximation As mentioned in Section 3.3, one could approximate the solutions in (30a) by assuming the second-order derivatives are small and thus can be ignored. In this case, we can directly use the fully first-order estimates without taking the implicit differentiation into account as

$$\widehat{\nabla}_{\boldsymbol{\theta}} U = \frac{\partial U}{\partial f}\frac{\partial f}{\partial \boldsymbol{\theta}}, \quad \widehat{\nabla}_\gamma U = \frac{\partial U}{\partial M}\frac{\partial M}{\partial \gamma}, \tag{32}$$

which are evaluated at the approximates $\hat{\boldsymbol{\mu}}$ and $\hat{\boldsymbol{\nu}}$ of the LL solution $\boldsymbol{\mu}^*, \boldsymbol{\nu}^*$. We next demonstrate it with an example of inductive logical reasoning using first-order optimization.

Discussion on theoretical guarantee and optimality To achieve theoretical guarantees, one sufficient condition is that the LL function L in (1b) is twice differentiable and strongly convex respect to ϕ. The UL function U can be possibly non-convex. Based on (Ji et al. 2021, Theorem 1), the proposed first-order method, which solves the linear system in (31) and uses $\hat{\boldsymbol{\mu}}$ and $\hat{\boldsymbol{\nu}}$ of the solutions $\boldsymbol{\mu}^*$ and $\boldsymbol{\nu}^*$, can achieve the first-order ε-accurate stationary optimality, i.e., $\|\nabla_\psi\| \leq \varepsilon$, if we use $\log \frac{1}{\varepsilon}$-level steps for the approximation in (32). Moreover, when the UL function U in (19) is strongly convex with respect to θ and γ, we can further show that the obtained solution achieves the global optimality, based on (Ji and Liang 2021, Theorem 9).

Example 2: Inductive Logical Reasoning

Given few known facts, logical reasoning aims at deducting the truth value of unknown facts with formal logical rules (Iwańska 1993; Overton 2013; Halpern 2013). Different types of logic have been invented to address distinct problems from various domains, including propositional logic (Wang et al. 2019), linear temporal logic (Xie et al. 2021), and first-order-logic (FOL) (Cropper and Morel 2021). Among them, FOL decomposes a logic *term* into *lifted predicates* and *variables* with quantifiers, which has strong generalization power and can be applied to arbitrary entities with any compositions. Due to the strong capability of generalization, FOL has been widely leveraged in knowledge graph reasoning (Yang and Song 2019) and robotic task planning (Chitnis et al. 2022). However, traditional FOL requires a human expert to carefully design the predicates and rules, which is tedious. Automatically summarizing the FOL predicates and rules is a long-standing problem, which is known as inductive logic programming (ILP). However, existing works study ILP only in simple structured data like knowledge graphs. To extend this research into robotics, we will explore how IL can make ILP work with high-dimensional data like RGB images.

Figure 5. One snapshot of the visual action prediction task of LogiCity benchmark. The next actions of each agent are reasoned by our iLogic based on groundings and learned rules.

Background One stream of the solutions to ILP is based on forward search algorithms (Cropper and Morel 2021; Shindo et al. 2023; Hocquette et al. 2024). For example, Popper constructs answer set programs based on failures, which can significantly reduce the hypothesis space (Cropper and Morel 2021). However, as FOL is a combinatorial problem, search-based methods can be extremely time-consuming as the samples scale up. Recently, some works introduced neural networks to assist the search process (Yang et al. 2017; Yang and Song 2019; Yang et al. 2022b) or directly implicitly represent the rule (Dong et al. 2019). To name a few, NeurlLP re-formulates the FOL rule inference into the multi-hop reasoning process, which can be represented as a series of matrix multiplications (Yang et al. 2017). Thus, learning the weight matrix becomes equivalent to inducting the rules. Neural logic machines, on the other hand, designed a new FOL-inspired network structure, where the rules are implicitly stored in the network weights (Dong et al. 2019).

Despite their promising results in structured symbolic datasets, such as binary vector representations in BlocksWorld (Dong et al. 2019) and knowledge graphs (Yang et al. 2017), their capability of handling high-dimensional data like RGB images is rarely explored.

Approach To address this gap, we verify the IL framework with the visual action prediction (VAP) task in the LogiCity (Li et al. 2024), a recent logical reasoning benchmark. In VAP, a model must simultaneously discover traffic rules, identify the concept of agents and their spatial relationships, and predict the agents' actions. As shown in Figure 4, we utilize a grounding network f to predict the agent concepts and spatial relationships, which takes

the observations such as images as the model input. The predicted agent concepts and their relationships are then sent to the reasoning module for rule induction and action prediction. The learned rules from the reasoning process are stored in the memory and retrieved when necessary. For example, the grounding network may output concepts "`IsTiro(`A_1`)`", "`IsPedestrian(`A_2`)`," and their relationship "`NextTo(`A_1`, `A_2`)`". Then the reasoning engine applies the learned rules, "`Slow(`X`)` $\leftarrow$ ($\exists$ Y `IsTiro(`X`)` $\wedge$ `IsPedestrian(`Y`)` $\wedge$ `NextTo(`X`, `Y`)`) $\vee$ ...", stored in the memory module to infer the next actions "`Slow(`A_1`)`" or "`Normal(`A_2`)`", as is displayed in Figure 5. Finally, the predicted actions and the observed actions from the grounding networks are used for loss calculation. In summary, we formulate this pipeline in (33) and refer to it as imperative logical reasoning (iLogic):

$$\min_{\boldsymbol{\theta}} \; U(f(\boldsymbol{\theta},\boldsymbol{x}), g(\boldsymbol{\mu}^*), M(\boldsymbol{\nu}^*)), \tag{33a}$$

$$\text{s.t.} \;\; \boldsymbol{\mu}^*,\, \boldsymbol{\nu}^* = \operatorname*{arg\,min}_{\boldsymbol{\mu},\,\boldsymbol{\nu}} L(f, g(M,\boldsymbol{\mu}), M(\boldsymbol{\nu})). \tag{33b}$$

Specifically, in the experiments we use the same cross-entropy function for the UL and LL costs, i.e., $U \doteq L \doteq -\sum a_i^o \log a_i^p$, where a_i^o and a_i^p are the observed and predicted actions of the i-th agent, respectively. This results in a self-supervised simultaneous grounding and rule induction pipeline for logical reasoning. Additionally, f can be any grounding networks and we use a feature pyramid network (FPN) (Lin et al. 2017) as the visual encoder and two MLPs for relationships; g can be any logical reasoning engines and we use a neural logical machine (NLM) (Dong et al. 2019) with parameter $\boldsymbol{\mu}$; and M can be any memory modules storing the learned rules and we use the MLPs in the NLM parameterized by $\boldsymbol{\nu}$. More details about the task, model, and list of concepts are presented in the Section 5.2.

Optimization Given that gradient descent algorithms can update both the grounding networks and the NLM, we can apply the first-order optimization to solve iLogic. The utilization of task-level action loss enables efficient self-supervised training, eliminating the necessity for explicit concept labels. Furthermore, BLO, compared to single-level optimization, helps the model concentrate more effectively on learning concept grounding and rule induction, respectively. This enhances the stability of optimization and decreases the occurrence of sub-optimal outcomes. Consequently, the model can learn rules more accurately and predict actions with greater precision.

4.3 Constrained Optimization

We next illustrate the scenarios in which the LL cost L in (1b) is subject to the general constraints in (1c). We discuss two cases with equality and inequality constraints, respectively. Constrained optimization is a thoroughly explored field, and related findings were presented in (Dontchev et al. 2009) and summarized in (Gould et al. 2021). This study will focus on the integration of constrained optimization into our special form of BLO (1) under the framework of IL.

Equality Constraint In this case, the constraint in (1c) is

$$\xi(M(\boldsymbol{\gamma},\boldsymbol{\nu}),\boldsymbol{\mu},f) = 0. \tag{34}$$

Recall that $\boldsymbol{\phi} = [\boldsymbol{\mu}^\top, \boldsymbol{\nu}^\top]^\top$ is a vector concatenating the symbol-like variables $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$, and $\boldsymbol{\psi} = [\boldsymbol{\theta}^\top, \boldsymbol{\gamma}^\top]^\top$ is a vector concatenating the neuron-like variables $\boldsymbol{\theta}$ and $\boldsymbol{\gamma}$. Therefore, the implicit gradient components can be expressed as differentiation of $\boldsymbol{\phi}^*$ to $\boldsymbol{\psi}$, and we have

$$\nabla\boldsymbol{\phi}^*(\boldsymbol{\psi}) = \begin{pmatrix} \frac{\partial \boldsymbol{\mu}^*}{\partial \boldsymbol{\theta}} & \frac{\partial \boldsymbol{\mu}^*}{\partial \boldsymbol{\gamma}} \\ \frac{\partial \boldsymbol{\nu}^*}{\partial \boldsymbol{\theta}} & \frac{\partial \boldsymbol{\nu}^*}{\partial \boldsymbol{\gamma}} \end{pmatrix}. \tag{35}$$

Lemma 1. *Assume the LL cost $L(\cdot)$ and the constraint $\xi(\cdot)$ are 2^{nd}-order differentiable near $(\boldsymbol{\theta},\boldsymbol{\gamma},\boldsymbol{\phi}^*(\boldsymbol{\theta},\boldsymbol{\gamma}))$ and the Hessian matrix H below is invertible, we then have*

$$\nabla\boldsymbol{\phi}^*(\boldsymbol{\psi}) = H^{-1}L_{\boldsymbol{\phi}}^\top[L_{\boldsymbol{\phi}}H^{-1}L_{\boldsymbol{\phi}}^\top]^{-1}(L_{\boldsymbol{\phi}}H^{-1}L_{\boldsymbol{\phi}\boldsymbol{\psi}} - L_{\boldsymbol{\psi}}) - H^{-1}L_{\boldsymbol{\phi}\boldsymbol{\psi}}, \tag{36}$$

where the derivative matrices are given by

$$\begin{aligned} L_{\boldsymbol{\phi}} &= \frac{\partial \xi(M(\boldsymbol{\gamma},\boldsymbol{\nu}^*),\boldsymbol{\mu}^*,f)}{\partial \boldsymbol{\phi}}, \quad L_{\boldsymbol{\psi}} = \frac{\partial \xi(M(\boldsymbol{\gamma},\boldsymbol{\nu}^*),\boldsymbol{\mu}^*,f)}{\partial \boldsymbol{\psi}}, \\ L_{\boldsymbol{\phi}\boldsymbol{\psi}} &= \frac{\partial^2 L(f,g(\boldsymbol{\mu}^*),M(\boldsymbol{\gamma},\boldsymbol{\nu}^*))}{\partial \boldsymbol{\psi}\partial \boldsymbol{\phi}} - \lambda \frac{\partial^2 \xi(M(\boldsymbol{\gamma},\boldsymbol{\nu}^*),\boldsymbol{\mu}^*,f)}{\partial \boldsymbol{\psi}\partial \boldsymbol{\phi}}, \\ H &= \frac{\partial^2 L(f,g(\boldsymbol{\mu}^*),M(\boldsymbol{\gamma},\boldsymbol{\nu}^*))}{\partial^2 \boldsymbol{\phi}} - \lambda \frac{\partial^2 \xi(M(\boldsymbol{\gamma},\boldsymbol{\nu}^*),\boldsymbol{\mu}^*,f)}{\partial^2 \boldsymbol{\phi}}, \end{aligned} \tag{37}$$

and duel variable $\lambda \in \mathbb{R}$ satisfies $\lambda L_{\boldsymbol{\phi}} = \frac{\partial L\xi(f,g(\boldsymbol{\mu}),M(\boldsymbol{\gamma},\boldsymbol{\nu}))}{\partial \boldsymbol{\phi}}$.

Proof. First, since this is a constrained problem, one can define the Lagrangian function

$$\mathfrak{L}(\boldsymbol{\psi},\boldsymbol{\phi},\lambda) = L(f,g(\boldsymbol{\mu}),M(\boldsymbol{\gamma},\boldsymbol{\nu})) - \lambda\xi(M(\boldsymbol{\gamma},\boldsymbol{\nu}),\boldsymbol{\mu},f). \tag{38}$$

Assume $\boldsymbol{\phi}^*$ to be the solution given input $\boldsymbol{\psi}$. Then, a λ exists such that the Lagrangian has zero gradients at $\boldsymbol{\phi}^*$ and λ. Thus, taking derivatives w.r.t. $\boldsymbol{\phi}$ and λ yields

$$\begin{bmatrix} \frac{\partial L(f,g(\boldsymbol{\mu}^*),M(\boldsymbol{\gamma},\boldsymbol{\nu}^*))}{\partial \boldsymbol{\phi}} - \lambda \frac{\partial \xi(M(\boldsymbol{\gamma},\boldsymbol{\nu}^*),\boldsymbol{\mu}^*,f)}{\partial \boldsymbol{\phi}} \\ \xi(M(\boldsymbol{\gamma},\boldsymbol{\nu}^*),\boldsymbol{\mu}^*,f) \end{bmatrix} = \mathbf{0}. \tag{39}$$

Note that if $\lambda = 0$, then this means that the gradient $\frac{\partial L(f,g(\boldsymbol{\mu}^*),M(\boldsymbol{\gamma},\boldsymbol{\nu}^*))}{\partial \boldsymbol{\phi}} = \mathbf{0}$ (i.e., the solution of the unconstrained problems already satisfies the constraint). If $\lambda > 0$, then we can obtain from (39) that

$$\frac{\partial L(f,g(\boldsymbol{\mu}^*),M(\boldsymbol{\gamma},\boldsymbol{\nu}^*))}{\partial \boldsymbol{\phi}} = \lambda \frac{\partial \xi(M(\boldsymbol{\gamma},\boldsymbol{\nu}^*),\boldsymbol{\mu}^*,f)}{\partial \boldsymbol{\phi}}, \tag{40}$$

which means that the gradient of L w.r.t. $\boldsymbol{\phi}$ has the same direction as the gradient of constraint ξ w.r.t. $\boldsymbol{\phi}$ at the optimal solution $\boldsymbol{\mu}^*$ and $\boldsymbol{\nu}^*$. Taking the derivative of (39) w.r.t. $\boldsymbol{\psi}$ and noting that $\boldsymbol{\phi}^*$ and λ are functions of $\boldsymbol{\psi}$, thus

$$\frac{\partial^2 L}{\partial \boldsymbol{\psi}\partial \boldsymbol{\phi}} + \frac{\partial^2 L}{\partial^2 \boldsymbol{\phi}}\nabla\boldsymbol{\phi}^* - \frac{\partial \xi}{\partial \boldsymbol{\phi}}\nabla\lambda - \lambda\left(\frac{\partial^2 \xi}{\partial \boldsymbol{\psi}\partial \boldsymbol{\phi}} + \frac{\partial^2 \xi}{\partial^2 \boldsymbol{\phi}}\nabla\boldsymbol{\phi}^*\right) = \mathbf{0}, \tag{41a}$$

$$\frac{\partial \xi}{\partial \boldsymbol{\psi}} + \frac{\partial \xi}{\partial \boldsymbol{\phi}}\nabla\boldsymbol{\phi}^* = \mathbf{0}, \tag{41b}$$

which together indicates that

$$\nabla\boldsymbol{\phi}^* = H^{-1}L_{\boldsymbol{\phi}}^\top[L_{\boldsymbol{\phi}}H^{-1}L_{\boldsymbol{\phi}}^\top]^{-1}(L_{\boldsymbol{\phi}}H^{-1}L_{\boldsymbol{\phi}\boldsymbol{\psi}} - L_{\boldsymbol{\psi}}) - H^{-1}L_{\boldsymbol{\phi}\boldsymbol{\psi}}, \tag{42}$$

which completes the proof.

Inequality Constraint In this case, the constraint in (1c) is

$$\xi(M(\boldsymbol{\gamma},\boldsymbol{\nu}),\boldsymbol{\mu},f) \leq 0. \tag{43}$$

The analysis is similar to the equality case, with minor differences. Since the solution $\boldsymbol{\phi}^*$ with the inequality

constraint (43) is a local minimum of the original inequality-constrained problem, it is reasonable to assume that the inequality constraints are active at ϕ^*. Assume $\phi^*(\psi)$ exists, functions $L(\cdot)$ and $\xi(\cdot)$ are 2^{nd}-order differentiable in the neighborhood of $(\psi, \phi^*(\psi))$, and the inequality constraint is active at $\phi^*(\psi)$. Then, we can derive the same implicit derivative $\nabla\phi^*(\psi)$ as in (37) of the equality case, except that the dual variable $\lambda \leq 0$ and $\nabla\phi^*(\psi)$ needs to be computed using one-sided partial $\lambda = 0$ due to non-differentiability, meaning we approach the point from only one direction (either left or right) to define the gradient, as the function may have abrupt changes or 'kinks' at that point.

Proof. Because the inequality constraints are active at ϕ^* (i.e., $\xi(M(\gamma, \nu^*), \mu^*, f) = 0$), the remaining proof exactly follows the same steps as in the equality case.

While we have derived closed-form solutions for the equality constraint in (34) and the inequality constraint in (43), calculating the matrix inversions H^{-1} and $[L_\phi H^{-1} L_\phi^\top]^{-1}$ can be computationally expensive. To mitigate this, one may employ the approach discussed in Section 4.1, which involves approximating the matrix-inverse-vector product $H^{-1}y$ by solving the linear system $\min_v \frac{1}{2} v^\top H v - v^\top y$. This complicated computing process can sometimes be simplified by specific robot settings. We next present an example of robot control illustrating this process.

Discussion on theoretical guarantee and optimality When the Hessian matrices $\frac{\partial^2 L}{\partial^2 \phi}$ and $\frac{\partial^2 \xi}{\partial^2 \phi}$ of the LL function L and the constraint function ξ are invertible, we can guarantee the matrix H in (37) is invertible. One sufficient condition for such invertibility is that L and ξ are strongly convex. Motivated by the approximate Karush-Kuhn-Tucker (KKT) condition (Andreani et al. 2010), which is characterized as an optimality condition for nonlinear program, we can use the residual function in (Yao et al. 2024b, Eq. (15)) as an optimality measurement. As a result, the optimality can be established and guaranteed under these conditions.

Example 3: Model Predictive Control

Modeling and control of nonlinear dynamics precisely is critical for robotic applications such as aerial manipulation (He et al. 2023; Geng and Langelaan 2020) and off-road navigation (Triest et al. 2023). Although classic methods based on physics have excellent generalization abilities, they rely heavily on problem-specific parameter tuning. Methods like state feedback (Mellinger and Kumar 2011) and optimal control (Garcia et al. 1989; Ji et al. 2022b) require precise system modeling to predict and control dynamics effectively. However, it is challenging to accurately model unpredictable factors such as wind gusts, boundary layer effects, uneven terrain, or hidden dynamics of chaotic nonlinear effects.

Background In recent years, researchers have spent efforts to integrate physical modeling into data-driven methods and have gained remarkable success in the field of system control (O'Connell et al. 2022; Hu et al. 2023). For example, Amos et al. (2018) developed the differentiable model predictive control (MPC) to combine the physics-based modeling with data-driven methods, enabling learning dynamic models and control policies in an end-to-end manner (Jin et al. 2020). However, many prior studies depend on expert demonstrations or labeled data for supervised learning (Jin et al. 2022; Song and Scaramuzza 2022). While supervised learning yields effective results in the training datasets, it suffers from challenging conditions such as unseen environments and external disturbances.

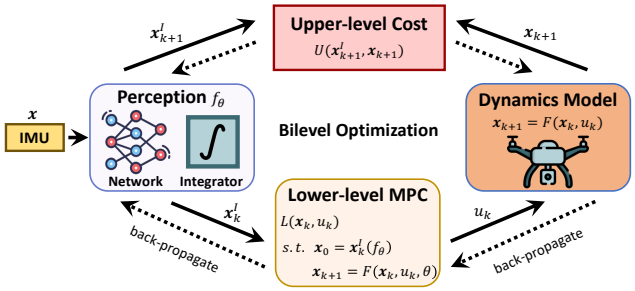


Figure 6. The framework of iMPC. The IMU model predicts the current state x^I_k. The differentiable MPC minimizes the lower-level cost L to get the optimal action u_k, which controls the dynamics model to the next predicted state (x_{k+1}) and actuates the real UAV to the next state measured by the IMU model (x^I_{k+1}). The upper-level optimization minimizes the discrepancy between the two states x_{k+1} and x^I_{k+1}.

Approach To eliminate dependence on human demonstrations, we incorporate differentiable MPC into the IL framework, hereafter termed imperative MPC (iMPC). We present a specific application, the IMU-based attitude control for unmanned aerial vehicles (UAVs) as an example, where a network denoises the raw IMU measurements and predicts the UAV attitude, which is then used by MPC for attitude control. It is worth noting that, although designed for attitude control, iMPC can be adjusted to other control problems.

As illustrated in Figure 6, iMPC consists of a perception network $f(\theta, x)$ and a physics-based reasoning engine based on MPC. Specifically, the encoder f takes raw sensor measurements, and here are IMU measurements x, to predict the HL parameters for MPC, and here is UAV's current state (attitude) x^I_k. The MPC takes the current attitude as initial state and solves for the optimal states and control sequence $\mu^* = \{\mu^*_k\}_{1:T} = \{x_k, u_k\}_{1:T}$ over a time horizon T under the constraints of system dynamics $F(\cdot)$, actuator limit, etc. Following the IL framework, the iMPC can be formulated as:

$$\min_{\theta} \quad U\left(f(x;\theta), g(\mu^*)\right), \tag{44a}$$

$$\text{s.t.} \quad \mu^* = \arg\min_{\mu} L(f, g(\mu)), \tag{44b}$$

$$\text{s.t.} \quad x_{k+1} = F(x_k, u_k), \quad k = 1, \cdots, T \tag{44c}$$

$$x_0 = x(t_0), \tag{44d}$$

$$x_k \in \mathcal{X};\ u_k \in \mathcal{U}, \tag{44e}$$

where the LL optimization is the standard MPC optimization to output an optimal control sequence $\mu^* = \{\mu_k\}_{1:T}$. Here, L is designed to minimize the tracking error and control effort:

$$L(\mu_k) \doteq \sum_{k=0}^{T-1} \left(\Delta\mu_k^\top Q_k \Delta\mu_k + p_k \Delta\mu_k \right), \tag{45}$$

where $\Delta\mu_k = \mu_k - \mu_k^{\text{ref}}$, μ_k^{ref} is the reference state trajectory, and Q_k and q_k are the weight matrices to balance tracking

performance and control effort, respectively. The optimal control action $\boldsymbol{u}_k$ will be commanded to both the modeled UAV (dynamics) and the real UAV, leading to the next state $\boldsymbol{x}_{k+1}$ and $\boldsymbol{x}^I_{k+1}$ (measured by the IMU network). The UL cost is defined as the Euclidean distance between $\boldsymbol{x}_{k+1}$ and $\boldsymbol{x}^I_{k+1}$

$$U(\boldsymbol{\theta}) \doteq \|\boldsymbol{x}^I_{k+1} - \boldsymbol{x}_{k+1}\|_2. \tag{46}$$

This discrepancy between the predicted states and the IMU network measurement reflects the models's imperfectness, indicating the learnable parameters $\boldsymbol{\theta}$ in both the network and the MPC module can be adjusted. Specifically, we adopt an IMU denoising model, AirIMU (Qiu et al. 2024) as the perception network, which is a neural model that can propagate system uncertainty through a time horizon. The differentiable MPC (Diff-MPC) from PyPose (Wang et al. 2023) is selected as the physics-based reasoning module.

Optimization Similar to other examples, the key step for the optimization is to compute the implicit gradient $\frac{\partial \boldsymbol{\mu}*}{\partial \boldsymbol{\theta}}$ from (37). Thanks to the Diff-MPC which uses a problem-agnostic AutoDiff implementation that applies one extra optimization iteration at the optimal point in the forward pass to enable the automatic gradient computation for the backward pass, our iMPC does not require the computation of gradients for the entire unrolled chain or analytical derivatives via implicit differentiation. This allows us to backpropagate the UL cost U through the MPC to update the network parameters $\boldsymbol{\theta}$ in one step. Additionally, iMPC not only avoids the heavy reliance on annotated labels leveraging the self-supervision from the physical model but also obtains dynamically feasible predictions through the MPC module. This results in a "physics-infused" network, enabling more accurate attitude control, as demonstrated in Section 5.3.

4.4 Second-order Optimization

We next illustrate the scenario that LL cost L in (1b) requires 2nd-order optimizers such as Gauss-Newton (GN) and Levenberg–Marquardt (LM). Because both GN and LM are iterative methods, one can leverage *unrolled differentiation* listed in Algorithm 1 to solve BLO. For the second strategy *implicit differentiation*, we could combine (30a) with a linear system solver in (31) in Section 4.2 to compute the implicit gradient $\nabla_{\boldsymbol{\theta}} U$ and $\nabla_{\boldsymbol{\gamma}} U$. Similar to the practical approach with LL first-order optimization, we can first use a second-order optimizer to solve the LL problem in (1b) to obtain approximates $\hat{\boldsymbol{\mu}}$ and $\hat{\boldsymbol{\nu}}$ of the solutions $\boldsymbol{\mu}^*$ and $\boldsymbol{\nu}^*$, which are then incorporated into (30a) to obtain an approximate $\widehat{\nabla}_{\boldsymbol{\theta}} U$ of the UL gradient $\nabla_{\boldsymbol{\theta}} U$ (similarly for $\widehat{\nabla}_{\boldsymbol{\gamma}} U$). Then, the UL gradient approximates $\widehat{\nabla}_{\boldsymbol{\theta}} U$ and $\widehat{\nabla}_{\boldsymbol{\gamma}} U$ are used to optimize the target variables $\boldsymbol{\theta}$ and $\boldsymbol{\gamma}$. The first-order approximation without any second-order derivative computations as in (32) can be also developed for a practical training speedup.

Discussion on theoretical guarantee and optimality The theoretical guarantee is similar to that for the first-order methods in Section 4.2, as long as the second-order LL optimizers such as Gauss-Newton and LM approximate $\boldsymbol{\mu}^*$ and $\boldsymbol{\nu}^*$ to the target accuracy ε. For example, if the LL problem is strongly convex, the second-order optimization can find an ε-accurate solution with an accelerated linear convergence rate (Ji and Liang 2021, Theorem 9 and 10).

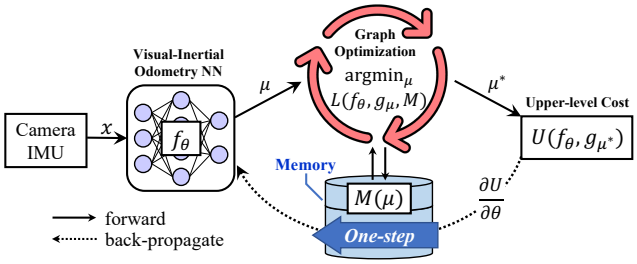


Figure 7. The framework of iSLAM. On the forward path, the odometry module f (front-end) predicts the robot trajectory $\boldsymbol{\mu}$. Then the pose graph optimization (back-end) minimizes the cost L in several iterations to get optimal poses $\boldsymbol{\mu}^*$. On the backward path, the cost U is back-propagated through the map M with a "one-step" strategy to update the network parameters $\boldsymbol{\theta}$.

Example 4: SLAM

Simultaneous localization and mapping (SLAM) is a critical technology in computer vision and robotics (Aulinas et al. 2008; Cadena et al. 2016). It aims to simultaneously track the trajectory of a robot and build a map of the environment. SLAM is essential in diverse robotic applications, such as indoor navigation (Wang et al. 2017a), underwater exploration (Suresh et al. 2020), and space missions (Dor et al. 2021). A SLAM system generally adheres to a front-end and back-end architecture. Specifically, the front-end is typically responsible for interpreting sensor data and providing an initial estimate of the robot's trajectory and the environment map; the back-end refines the initial estimate by performing global optimization to improve overall accuracy.

Background Recent advancements in the field of SLAM have shown that supervised learning-based methods can deliver remarkable performance in front-end motion estimation (Wang et al. 2021b; Teed and Deng 2021). These methods apply machine learning algorithms that depend on external supervision, usually in the form of a labeled dataset. Once trained, the model can make estimations without being explicitly programmed for the task. Meanwhile, geometry-based approaches remain crucial for the system's back-end, primarily tasked with reducing front-end drift (Qin et al. 2018; Xu et al. 2025). These techniques employ geometrical optimization, such as pose graph optimization (PGO) (Labbe and Michaud 2014), to maintain global consistency and minimize trajectory drift. However, their front-end and back-end components operate independently and are connected only in one direction. This means that the data-driven front-end model cannot receive feedback from the back-end for joint error correction. As a result, this decoupled approach might lead to suboptimal performance, adversely affecting the overall performance of the current SLAM systems.

Approach In response to the problem of suboptimal performance of the decoupled approach, we introduce IL into the SLAM system to jointly optimize the front-end and back-end components. Specifically, we reformulate SLAM as a BLO between the front-end and back-end as:

$$\min_{\boldsymbol{\theta}} \; U(f(\boldsymbol{\theta}, \boldsymbol{x}), L(\boldsymbol{\mu}^*)), \tag{47a}$$

$$\text{s.t.} \;\; \boldsymbol{\mu}^* = \arg\min_{\boldsymbol{\mu}} L(f, \boldsymbol{\mu}, M), \tag{47b}$$

where $f(\boldsymbol{\theta},\boldsymbol{x})$ is the front-end odometry network with parameter $\boldsymbol{\theta}$ and input $\boldsymbol{x}$; $L(f,\boldsymbol{\mu},M)$ is the back-end geometry-based reasoning (optimization) which enforces global consistency. Specifically, $\boldsymbol{\mu}$ is the poses and velocities to be optimized; M is a map (memory) to store the historical estimations; and U is the UL loss for model training. We omit unused symbols in (1), and let U depend directly on L instead of g as in the general form. This framework is illustrated in Figure 7, which we refer to as imperative SLAM (iSLAM).

The choice of front-end odometry network and the back-end optimization are flexible in iSLAM. Here we illustrate one example of using a stereo visual-inertial odometry network as the front-end and a pose-velocity graph optimization as the back-end (Fu et al. 2024). Specifically, the front-end odometry consists of two independent efficient modules, i.e., a stereo odometry network and an IMU network (Qiu et al. 2024). The stereo odometry network is responsible for pose estimation, while the IMU network estimates both pose and velocity. These estimations are then incorporated as constraints (edges) in the back-end pose-velocity graph, linking the pose and velocity nodes. This setup allows for the optimization of pose and velocity nodes to reduce inconsistency between the visual and inertial measurements. As a result, the LL cost L can be any estimation inconsistency between graph nodes:

$$L = \text{PVG}(\hat{p}_1^v \cdots \hat{p}_k^v, \hat{p}_1^i \cdots \hat{p}_k^i, \hat{v}_1^i \cdots \hat{v}_k^i), \tag{48}$$

where PVG denotes the summation of the inconsistencies calculated based on edge constraints in the pose-velocity graph; $\hat{p}_k^v$ and $\hat{p}_k^i$ are the estimates of transformations between the $(k-1)^{\text{th}}$ and k^{th} image frame from the stereo odometry network and the IMU network, respectively; $\hat{v}_k^i$ is the estimated delta velocity from the IMU network between the $(k-1)^{\text{th}}$ and k^{th} image frame. The pose and velocity nodes are combined to denote the LL variables $\boldsymbol{\mu} = \{p_0, p_1, \cdots, p_k, v_1, v_2, \cdots, v_k\}$. Intuitively, this LL solves the best poses and velocities based on both the visual and inertial observations, thus achieving better trajectory accuracy by combining the respective strengths of both camera and IMU.

In the process of UL optimization, the front-end model acquires knowledge from the feedback provided by the back-end. In this example, UL cost U is picked the same as L, although they are not necessarily equal in the general IL framework. Specifically, U is calculated on the optimal estimations $\boldsymbol{\mu}^*$ after PVGO and then back-propagated to networks f to tune its parameter $\boldsymbol{\theta}$ using gradient descent. More details about the iSLAM can be found at Section 5.4.

The structure of iSLAM results in a self-supervised learning method. At the back-end (lower-level), the robot's path is adjusted through PVGO to ensure geometric consistency between the visual and inertial motion estimations, whereas, at the front-end (upper-level), the model parameters are updated to incorporate the knowledge derived from the back-end, improving its performance and generalizability in unseen environments. This formulation under the IL framework seamlessly integrates the front-end and back-end into a unified optimization problem, facilitating reciprocal enhancement between the two components.

Optimization One can leverage Equation (29) to calculate the implicit gradients. However, due to the uniqueness of (47), we can apply an efficient approximation method, which we refer to as a "one-step" back-propagation strategy. It utilizes the nature of stationary points to avoid unrolling the inner optimization iterations. Specifically, according to the chain rule, the gradient of U with respect to $\boldsymbol{\theta}$ is

$$\frac{\partial U}{\partial \boldsymbol{\theta}} = \frac{\partial U}{\partial f}\frac{\partial f}{\partial \boldsymbol{\theta}} + \frac{\partial U}{\partial L^*}\left(\frac{\partial L^*}{\partial f}\frac{\partial f}{\partial \boldsymbol{\theta}} + \frac{\partial L^*}{\partial \boldsymbol{\mu}^*}\frac{\partial \boldsymbol{\mu}^*}{\partial \boldsymbol{\theta}}\right), \tag{49}$$

where L^* is the short note of lower-level cost at its solution $L(f, g(\boldsymbol{\mu}^*), M)$. Intuitively, $\frac{\partial \boldsymbol{\mu}^*}{\partial \boldsymbol{\theta}}$ embeds the gradients from the inner optimization iterations, which is computationally heavy. However, if we assume the optimization converges (either to the global or local optimal), we have a stationary point where $\frac{\partial L^*}{\partial \boldsymbol{\mu}^*} \approx 0$. This eliminates the complex gradient term $\frac{\partial \boldsymbol{\mu}^*}{\partial \boldsymbol{\theta}}$. In this way, we bypass the LL optimization iterations and thereby can back-propagate the gradients in one step. Note that the precondition for using this method is that U incorporates L as the sole term involving $\boldsymbol{\mu}^*$, otherwise it will introduce a constant error term on convergence, as analyzed in Equation (18).

4.5 Discrete Optimization

In the case where the LL cost L in (1b) involves discrete optimization, the semantic attributes $\boldsymbol{z}$ in Figure 1 are usually defined in a discrete space. However, the existence of discrete variables leads to extreme difficulty in solving IL, especially the computation of implicit gradient. To overcome this challenge, we will present several gradient estimators for $\nabla_{\boldsymbol{\theta}} g \doteq \nabla_\theta \mathbb{E}_{f(\boldsymbol{z}|\boldsymbol{\theta})}[g(\boldsymbol{z},\mu,M)]$, where the reasoning engine g functions over discrete latent variables $\boldsymbol{z}$. In this section, we assume the network f outputs the distribution function where $\boldsymbol{z}$ is sampled, i.e., $\boldsymbol{z} \sim f(\boldsymbol{z}|\boldsymbol{\theta})$. Note that the difference is that $\boldsymbol{z}$ is the input to g, while $\boldsymbol{\mu}$ is the variable to optimize. For simplicity, we omit M and $\boldsymbol{\mu}$ in the reasoning engine $g(\boldsymbol{z})$.

Log-derivative Estimator The log-derivative estimator, denoted as $\hat{g}_l'$, converts the computation of a non-expectation to an expectation using the log-derivative trick (LDT), which can then be approximated by random sampling. Specifically, it can be derived with the following procedure:

$$\hat{g}_l' \doteq \nabla_\theta \mathbb{E}_{f(\boldsymbol{z}|\boldsymbol{\theta})}[g(\boldsymbol{z})] \tag{50a}$$

$$= \nabla_\theta \int f(\boldsymbol{z}|\boldsymbol{\theta}) g(\boldsymbol{z}) d\boldsymbol{z} \tag{50b}$$

$$= \int \nabla_\theta f(\boldsymbol{z}|\boldsymbol{\theta}) g(\boldsymbol{z}) d\boldsymbol{z} \quad \text{(Leibniz rule)} \tag{50c}$$

$$= \int f(\boldsymbol{z}|\boldsymbol{\theta}) \frac{\nabla_\theta f(\boldsymbol{z}|\boldsymbol{\theta})}{f(\boldsymbol{z}|\boldsymbol{\theta})} g(\boldsymbol{z}) d\boldsymbol{z} \quad \left(\text{Multiply } \frac{f(\boldsymbol{z}|\boldsymbol{\theta})}{f(\boldsymbol{z}|\boldsymbol{\theta})}\right) \tag{50d}$$

$$= \int f(\boldsymbol{z}|\boldsymbol{\theta}) \nabla_\theta \log f(\boldsymbol{z}|\boldsymbol{\theta}) g(\boldsymbol{z}) d\boldsymbol{z} \quad \text{(The LDT)} \tag{50e}$$

$$= \mathbb{E}_{f(\boldsymbol{z}|\boldsymbol{\theta})}[\nabla_\theta \log f(\boldsymbol{z}|\boldsymbol{\theta}) g(\boldsymbol{z})] \tag{50f}$$

$$\approx \frac{1}{S}\sum_{i=1}^{S} \nabla_\theta \log f(\boldsymbol{z}_i|\boldsymbol{\theta}) g(\boldsymbol{z}_i), \quad \boldsymbol{z}_i \sim f(\boldsymbol{z}|\boldsymbol{\theta}), \tag{50g}$$

where $\boldsymbol{z}_i$ is an instance among S samplings. Note that $\nabla_\theta f(\boldsymbol{z}|\boldsymbol{\theta})$ is **not** a distribution function and thus (50c) is not an expectation and cannot be approximated by sampling. In contrast, (50e) is an expectation after the LDT is applied. The *log-derivative* estimator is a well-studied technique widely used in reinforcement learning (Sutton and Barto 2018). The advantage is that it is **unbiased** but the drawback is that it often suffers from **high variance** (Grathwohl et al. 2017).

Reparameterization Estimator An alternative approach is applying the *reparameterization trick*, which reparameterizes the non-differentiable variable $\boldsymbol{z}$ by a differentiable function $\boldsymbol{z} = T(\boldsymbol{\theta}, \boldsymbol{\varepsilon})$, where $\boldsymbol{\varepsilon} \sim \mathcal{N}(0,1)$ is a standard Gaussian random variable. Denote the estimator as $\hat{g}'_r$, then

$$\hat{g}'_r \doteq \mathbb{E}\left[\frac{\partial}{\partial \boldsymbol{\theta}} g(\boldsymbol{z})\right] = \mathbb{E}\left[\frac{\partial g}{\partial T}\frac{\partial T}{\partial \boldsymbol{\theta}}\right], \quad \boldsymbol{\varepsilon} \sim \mathcal{N}(0,1) \tag{51a}$$

$$\approx \frac{1}{S}\sum_{i=1}^{S} \frac{\partial g}{\partial T(\boldsymbol{\varepsilon}_s)}\frac{\partial T(\boldsymbol{\varepsilon}_i)}{\partial \boldsymbol{\theta}}, \quad \boldsymbol{\varepsilon}_i \sim \mathcal{N}(0,1) \tag{51b}$$

where $\boldsymbol{\varepsilon}_i$ is an instance among S samplings. Compared to the log-derivative estimator, the reparameterization estimator has a **lower variance** but sometimes introduces **bias** when model assumptions are not perfectly aligned with the true data, or if there are approximations in the model (Li et al. 2022a). Reparameterization trick has also been well studied and widely used in variational autoencoders (Doersch 2016).

Control Variate Estimator We next introduce a simple but effective variance-reduced estimator based on control variate (Nelson 1990), as defined in Lemma 2. It has been widely used in Monte Carlo estimation methods (Geffner and Domke 2018; Richter et al. 2020; Leluc et al. 2022).

Lemma 2. *A control variate is a zero-mean random variable used to reduce the variance of another random variable. Assume a random variable X has an **unknown** expectation α, and another random variable Y has a known expectation β. Construct a new random variable X^* so that*

$$X^* = X + \gamma(Y - \beta), \tag{52}$$

*where $\boldsymbol{c} \doteq Y - \beta$ is a control variate, then the new random variable X^* has α as its expectation but with a **smaller variance** than X, when the constant γ is properly chosen.*

Proof. The variance of the new variable X^* is

$$\mathrm{Var}(\mathrm{X}^*) = \mathrm{Var}(\mathrm{X}) + \gamma^2 \mathrm{Var}(\mathrm{Y}) + 2\gamma \mathrm{Cov}(\mathrm{X}, \mathrm{Y}), \tag{53}$$

where $\mathrm{Var}(\mathrm{X}) = \mathbb{E}\|\mathrm{X}\|^2 - \mathbb{E}^2\|\mathrm{X}\|$ denotes the variance of a random variable, and $\mathrm{Cov}(\mathrm{X}, \mathrm{Y}) = \mathbb{E}[(\mathrm{X} - \mathbb{E}\mathrm{X})^\top (\mathrm{Y} - \mathbb{E}\mathrm{Y})]$ denotes the covariance of two random variables. To minimize $\mathrm{Var}(\mathrm{X}^*)$, we differentiate (53) w.r.t. γ and set the derivative to zero. This gives us the optimal value of γ as

$$\gamma^* = -\frac{\mathrm{Cov}(\mathrm{X}, \mathrm{Y})}{\mathrm{Var}(\mathrm{Y})}. \tag{54}$$

Substitute (54) into (53), we have the minimum variance

$$\mathrm{Var}^*(\mathrm{X}^*) = (1 - \rho^2_{\mathrm{X},\mathrm{Y}})\mathrm{Var}(\mathrm{X}), \tag{55}$$

where $\rho_{X,Y} = \frac{\mathrm{Cov}(\mathrm{X},\mathrm{Y})}{\sqrt{\mathrm{Var}(\mathrm{X})\mathrm{Var}(\mathrm{Y})}}$ is the correlation of the two random variables. Since $\rho_{X,Y} \in [-1, 1]$, we have $\mathrm{Var}(\mathrm{X}^*) \leq \mathrm{Var}(\mathrm{X})$; when $\rho_{X,Y}$ is close to 1, $\mathrm{Var}(\mathrm{X}^*)$ will get close to 0. This means that as long as we can design a random variable Y that is correlated to X, we can reduce the estimation variance of X, which completes the proof.

In practice, we usually have no direct access to γ^*, and thus cannot guarantee the optimality of the control variate for arbitrarily chosen Y. However, without loss of generality, we can assume X and Y are positively correlated and solve the inequality $\gamma^2 \mathrm{Var}(\mathrm{Y}) + 2\gamma \mathrm{Cov}(\mathrm{X}, \mathrm{Y}) < 0$, then $\gamma \in (-\frac{2\mathrm{Cov}(\mathrm{X},\mathrm{Y})}{\mathrm{Var}(\mathrm{Y})}, 0)$. This means that as long as γ is within the range, $\mathrm{Var}(\mathrm{X}^*)$ is less than $\mathrm{Var}(\mathrm{X})$. When X and Y are highly correlated and are of similar range, $\mathrm{Cov}(\mathrm{X}, \mathrm{Y}) \approx \mathrm{Var}(\mathrm{Y})$, thus $\gamma^* = -1$ is the most common empirical choice.

Inspired by this control variate technique, we can construct a variance-reduced gradient estimator for $\nabla_{\boldsymbol{\theta}} g$ by designing a surrogate network (control variate) $s(\boldsymbol{z})$ approximating the reasoning process $g(\boldsymbol{z})$ (Grathwohl et al. 2017). Specifically, substitute $\gamma = -1$ into (52), we have $X^* = X - Y + \beta$, this gives us the new **control variate estimator** $\hat{g}'_c$:

$$\hat{g}'_c = \hat{g}'_l - \hat{g}'_s + s'(\boldsymbol{z}), \tag{56}$$

where $\hat{g}'_l$ is the sampling *log-derivative* estimator with high variance, $\hat{g}'_s$ denotes the *log-derivative* estimator of the surrogate network $s(\boldsymbol{z})$, and $s'(\boldsymbol{z})$ is another gradient estimator, which is the expectation of $\hat{g}'_s$. For example, when $s'(\boldsymbol{z})$ is a reparameterization estimator for $\hat{g}'_s$, a properly trained system will have $\hat{g}'_l$ and $\hat{g}'_s$ highly-correlated, $\mathbb{E}[\hat{g}'_s] = s'(\boldsymbol{z})$, then $\hat{g}'_c$ is an **unbiased** and **low-variance** estimator.

Using this control variate-based estimator, we can pass low-variance gradients through the semantic attribute $\boldsymbol{z}$ in the discrete space. This strategy does not assume the form of an LL optimization problem and thus can be applied to various applications. Note that the surrogate network is only needed for gradient estimation during training and is **unused** in the inference stage, meaning that the computational complexity is not increased. Next, we present an example using this technique to evaluate IL in the discrete space.

Example 5: Min-Max MTSP

The multiple traveling salesman problem (MTSP) seeks tours for multiple agents such that all cities are visited exactly once while minimizing a cost function defined over the tours. MTSP is critical in many robotic applications where a team of robots is required to collectively visit a set of target locations, e.g., unmanned aerial vehicles (Sundar and Rathinam 2013; Ma et al. 2019), automated agriculture (Carpio et al. 2021), warehouse logistics (Wurman et al. 2008; Ren et al. 2023b). The MTSP is renowned for its NP-hardness (Bektas 2006). Intuitively, MTSP involves many decision variables including assigning the cities to the agents and determining the visiting order of the assigned cities. For example, an MTSP with 100 cities and 10 agents involves $\frac{100!\times 99!}{10!\times 89!} \approx 10^{20000}$ possible solutions, while the number of atoms in the observable universe is estimated to be "merely" 10^{78} to 10^{82} (Gaztañaga 2023).

MTSP can be categorized by Min-Sum MTSP and Min-Max MTSP. Intuitively, Min-Sum MTSP aims to minimize the sum of tour lengths over all agents, while Min-Max MTSP aims to reduce the longest tour length. Though Min-Sum MTSP has been extensively studied (Bertazzi et al. 2015), Min-Max MTSP remains under-explored, and many techniques for Min-Sum MTSP may not directly apply to Min-Max MTSP. From the perspective of learning-based approaches, Min-Max MTSP is particularly challenging because the Min-Max operation is non-differentiable, which makes it harder to use gradient-based optimization and leads to a more complicated search for the global optimum. In this example, we will focus on the Min-Max MTSP, while our method may also apply to the Min-Sum MTSP. For the sake of brevity, we will use the abbreviation "MTSP" to refer to the Min-Max MTSP unless indicated otherwise.

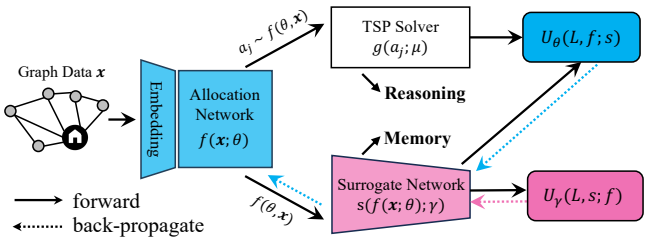


Figure 8. The framework of iMTSP. A surrogate network is introduced as the memory in the IL framework, constructing a low-variance gradient for the allocation network through the non-differentiable and discrete TSP solvers.

Algorithm 3 The iMTSP algorithm

Given MTSP Graph data $\boldsymbol{x}$, TSP solver g, and randomly initialized allocation and surrogate networks $f(\boldsymbol{\theta})$, $s(\gamma)$.
while not converged **do**
  $a_j \sim f(\boldsymbol{x};\boldsymbol{\theta})$; {Sampling allocation}
  $L \leftarrow \max_j D(g(a_j;\boldsymbol{\mu}))$; {$g$ is non-differentiable}
  $L' \leftarrow s(f(\boldsymbol{x};\boldsymbol{\theta});\gamma)$; {Cut gradients for L'}
  $\nabla\boldsymbol{\theta} \leftarrow [L - L']\frac{\partial}{\partial\boldsymbol{\theta}}\log f + \frac{\partial}{\partial\boldsymbol{\theta}}s(f;\gamma)$; {Compute $\frac{\partial U}{\partial\boldsymbol{\theta}}$}
  $\nabla\gamma \leftarrow \frac{\partial}{\partial\gamma}\frac{\partial U}{\partial\boldsymbol{\theta}}^2$; {Compute $\frac{\partial}{\partial\gamma}\mathrm{Var}(\frac{\partial \mathrm{U}}{\partial\boldsymbol{\theta}})$}
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha_1\nabla\boldsymbol{\theta}$; {Update allocation network}
  $\gamma \leftarrow \gamma - \alpha_2\nabla\gamma$; {Update surrogate network}
end while

Background Due to the high complexity, classic MTSP solvers such as Google's OR-Tools routing library meet difficulties for large-scale problems such as those with more than 500 cities (Furnon and Perron 2023). To overcome this issue, there has been a notable shift towards employing neural network models to tackle MTSP (Hu et al. 2020; Xin et al. 2021; Miki et al. 2018; Vinyals et al. 2015; Kool et al. 2018; Nazari et al. 2018; Park et al. 2021; Ouyang et al. 2021; Khalil et al. 2017; Costa et al. 2020; Wu et al. 2021). However, these methods still have fundamental limitations, particularly in their ability to generalize to new, unseen situations, and in consistently finding high-quality solutions for large-scale problems. Supervised models struggle with limited supervision on small-scale problems and lack feasible supervision for large-scale instances, leading to poor generalization (Xin et al. 2021; Miki et al. 2018; Vinyals et al. 2015). Reinforcement learning (RL)-based approaches usually exploit implementations of the policy gradient algorithm, such as the Reinforce algorithm (Williams 1992) and its variants. However, they often face the issue of slow convergence and highly sub-optimal solutions (Hu et al. 2020; Park et al. 2021). Other strategies like training greedy policy networks (Ouyang et al. 2021; Nazari et al. 2018; Kool et al. 2018; Khalil et al. 2017) and iteratively improving solutions (Costa et al. 2020; Wu et al. 2021) are also proposed but often get stuck at local optima.

Approach To overcome the above limitation, we reformulate MTSP as a BLO following the IL framework based on the control variate gradient estimator, as shown in Figure 8. This comprises a UL optimization, aiming to train an allocation network and a surrogate network assisting gradient estimation, and an LL optimization that solves decomposed single-agent TSPs. Specifically, the allocation network $f(\boldsymbol{x};\boldsymbol{\theta})$ takes the MTSP graph data $\boldsymbol{x}$ and produces a probability matrix, where its (i,j)-th entry represents the probability that city i is assigned to agent j. This decomposes the MTSP into multiple single-agent TSPs by random sampling the city allocation $a_j \sim f(\boldsymbol{x};\boldsymbol{\theta})$ for agent j. Then, a non-differentiable TSP solver g is employed to quickly find an optimal tour $\boldsymbol{\mu}$ for single-agent TSPs based on the allocation a_j, where $j = 1,\cdots,M$. During training, the surrogate network $s(\gamma)$, which serves as the memory module in the IL framework, is trained to have a low-variance gradient estimation through the non-differentiable reasoning engine g. This results in a self-supervised formulation for MTSP, which we refer to as imperative MTSP (iMTSP):

$$\min_{\boldsymbol{\theta}} U_{\boldsymbol{\theta}}\left(L(\boldsymbol{\mu}^*), f(\boldsymbol{\theta});\gamma\right); \quad \min_{\gamma} U_\gamma\left(L(\boldsymbol{\mu}^*), s(\gamma);\boldsymbol{\theta}\right) \tag{57a}$$

$$\text{s.t.} \quad \boldsymbol{\mu}^* = \arg\min_{\boldsymbol{\mu}} L(\boldsymbol{\mu}), \tag{57b}$$

$$L \doteq \max_j D(g(a_j;\boldsymbol{\mu})), \quad j = 1,2,\cdots,M, \tag{57c}$$

$$a_j \sim f(\boldsymbol{x};\boldsymbol{\theta}), \tag{57d}$$

where D computes the traveling cost given a tour, and L returns the maximum route length among all agents. The UL optimization consists of two separate components: the UL cost $U_{\boldsymbol{\theta}}$ is to optimize the allocation network $f(\boldsymbol{\theta})$, while U_γ is to optimize the surrogate network $s(\gamma)$. Inspired by the control variant gradient estimator (56) and to have a lower gradient variance, we design the UL costs $U_{\boldsymbol{\theta}}$ and U_γ as

$$U_{\boldsymbol{\theta}} \doteq L(\boldsymbol{\mu}^*)\log f(\boldsymbol{x};\boldsymbol{\theta}) - L'\log f(\boldsymbol{x};\boldsymbol{\theta}) + s(f(\boldsymbol{x};\boldsymbol{\theta});\gamma), \tag{58a}$$

$$U_\gamma \doteq \mathrm{Var}(\frac{\partial \mathrm{U}_{\boldsymbol{\theta}}}{\partial\boldsymbol{\theta}}), \tag{58b}$$

where L' is the output of the surrogate network (control variate) but is taken as a constant by cutting its gradients. The first instinct of training the surrogate network is to mimic the input-output relation of the TSP solver. However, highly correlated input-output mapping does not necessarily guarantee highly correlated gradients, which breaks the assumption of using control variate. To overcome this issue, U_γ is set as $\mathrm{Var}(\frac{\partial \mathrm{U}_{\boldsymbol{\theta}}}{\partial\boldsymbol{\theta}})$ to minimize its gradient variance.

Optimization We next present the optimization process for iMTSP. Intuitively, the gradient for the allocation network is the partial derivative of the UL cost $U_{\boldsymbol{\theta}}$ w.r.t. $\boldsymbol{\theta}$:

$$\frac{\partial U_{\boldsymbol{\theta}}}{\partial\boldsymbol{\theta}} = [L(\boldsymbol{\mu}^*) - L']\frac{\partial}{\partial\boldsymbol{\theta}}\log f(\boldsymbol{\theta}) + \frac{\partial}{\partial\boldsymbol{\theta}}s(f(\boldsymbol{\theta});\gamma), \tag{59}$$

which provides a low-variance gradient estimation through the non-differentiable TSP solver and the discrete decision space to the allocation network. Again, L and L' are treated as constants instead of functions of network parameters $\boldsymbol{\theta}$.

On the other hand, the surrogate network $s(f(\cdot),\phi)$ is separately optimized, and updated with a single sample variance estimator. Since $\mathrm{Var}(\frac{\partial \mathrm{U}}{\partial\boldsymbol{\theta}}) = \mathbb{E}[(\frac{\partial \mathrm{U}}{\partial\boldsymbol{\theta}})^2] - \mathbb{E}[(\frac{\partial \mathrm{U}}{\partial\boldsymbol{\theta}})]^2$, the gradient of the surrogate network is:

$$\frac{\partial U_\gamma}{\partial\gamma} = \frac{\partial}{\partial\gamma}\mathbb{E}[(\frac{\partial U_{\boldsymbol{\theta}}}{\partial\boldsymbol{\theta}})^2] - \frac{\partial}{\partial\gamma}\mathbb{E}[\frac{\partial U_{\boldsymbol{\theta}}}{\partial\boldsymbol{\theta}}]^2 = \frac{\partial}{\partial\gamma}\mathbb{E}[(\frac{\partial U_{\boldsymbol{\theta}}}{\partial\boldsymbol{\theta}})^2], \tag{60}$$

where $\frac{\partial}{\partial\gamma}\mathbb{E}[\frac{\partial U_{\boldsymbol{\theta}}}{\partial\boldsymbol{\theta}}]^2$ is always zero since $\frac{\partial U_{\boldsymbol{\theta}}}{\partial\boldsymbol{\theta}}$ in (59) is always an unbiased gradient estimator. This means that its expectation $\mathbb{E}[\frac{\partial U}{\partial\boldsymbol{\theta}}]$ will **not** change with γ. As a result,

in each iteration, we only need to alternatively compute (59) and (60) and then update the allocation and surrogate networks. In practice, $\mathbb{E}[(\frac{\partial U}{\partial \boldsymbol{\theta}})^2]$ is implemented as the mean of squared gradient $\frac{\partial U_{\boldsymbol{\theta}}}{\partial \boldsymbol{\theta}}$. We list a summarized optimization algorithm in Algorithm 3 for easy understanding.

5 Experiments

We next demonstrate the effectiveness of IL in robot autonomy by evaluating the five robotic examples, i.e., logical reasoning in autonomous driving, path planning for mobile robots, model predictive control for UAV, SLAM for mobile robots, and min-max MTSP for multi-agent collaboration, and comparing their performance with the SOTA algorithms in their respective fields.

5.1 Path Planning

In this section, we will conduct the experiments for iA* and iPlanner for global and local planning, respectively.

Example A: Global Path Planning

To demonstrate the effectiveness of IL in global planning, we conduct both qualitative and quantitative evaluations for iA*.

5.1.1 Baselines To evaluate the generalization ability of the proposed iA* framework, we select both classic and data-driven methods as the baselines. Specifically, we will compare iA* with the popular classic method, weighted A* (Pohl 1970) and the latest imitation learning-based methods including neural A* (Yonetani et al. 2021b), focal search with path probability map (FS + PPM) and weighted A* with correction factor (WA* + CF) (Kirilenko et al. 2023).

5.1.2 Datasets To evaluate iA* in diverse environments, we select three widely used datasets in path planning, Motion Planning (MP) (Bhardwaj et al. 2017), Matterport3D (Chang et al. 2017), and Maze (Ivanitskiy et al. 2023). Specifically, the MP dataset contains a set of binary obstacle maps from eight environments; the Matterport3D dataset provides diverse indoor truncated signed distance function (TSDF) maps; and the Maze dataset contains sophisticated mazes-type maps. To better evaluate the generalization ability, we randomly sample the start and goal positions following (Yonetani et al. 2021b) to provide more diverse maps. It is essential to note that all methods only use the MP dataset for training, and use Matterport3D and Maze for testing, which is to demonstrate their generalization abilities.

5.1.3 Metrics To evaluate the performance of different algorithms, we adopt the widely used metrics including the reduction ratio of node explorations (*Exp*) (Yonetani et al. 2021b) and the reduction ratio of time (*Rt*). *Exp* is the ratio of the reduced search area relative to the classical A*, which can be defined as $Exp = 100 \times \frac{S^* - S}{S^*}$, where S represents the search area of the method and S^* is the search area of classical A*. *Rt* is the ratio of saved time relative to the classical A*, which can be defined as $Rt = 100 \times \frac{t^* - t}{t^*}$, where t is the runtime of the method and t^* is the runtime of classical A*. Intuitively, *Exp* aims to measure the search efficiency while *Rt* aims to measure the runtime efficiency.

5.1.4 Quantitative Performance Since the classical A^* has a poor search efficiency for large maps, we resize the maps to different sizes, including 64×64, 128×128, and 256×256, to further demonstrate the efficiency of different algorithms.

Table 2. Path planning evaluation on the MP dataset. The symbol '–' denotes the method cannot provide meaningful results with that map size due to the large search space.

Metric	*Exp*	*Rt*	*Exp*	*Rt*	*Exp*	*Rt*
Map Size	64×64		128×128		256×256	
WA*	63.5%	47.0%	70.1%	55.4%	52.0%	54.4%
Neural A*	56.0%	24.5%	67.2%	42.1%	44.9%	50.0%
WA*+CF	68.8%	48.0%	–	–	–	–
FS+PPM	**88.3%**	**83.6%**	–	–	–	–
iA* (ours)	69.4%	52.8%	**75.9%**	**62.3%**	**56.1%**	**61.5%**

Table 3. Path planning evaluation on the Matterport3D dataset.

Metric	*Exp*	*Rt*	*Exp*	*Rt*	*Exp*	*Rt*
Map Size	64×64		128×128		256×256	
WA*	48.1%	36.8%	76.4%	79.6%	84.7%	81.5%
Neural A*	42.3%	28.4%	65.6%	50.1%	79.4%	63.9%
WA*+CF	57.2%	42.7%	–	–	–	–
FS+PPM	9.3%	−43.8%	–	–	–	–
iA* (ours)	**57.8%**	**44.7%**	**79.2%**	**81.8%**	**89.5%**	**87.8%**

Table 4. Path planning evaluation on the Maze dataset.

Metric	*Exp*	*Rt*	*Exp*	*Rt*	*Exp*	*Rt*
Map Size	64×64		128×128		256×256	
WA*	21.7%	27.4%	48.8%	11.7%	41.8%	29.5%
Neural A*	30.9%	40.8%	60.5%	32.8%	51.0%	38.9%
WA*+CF	**47.3%**	**45.3%**	–	–	–	–
FS+PPM	−8.9%	−105.0%	–	–	–	–
iA* (ours)	37.8%	44.9%	**64.8%**	**33.0%**	**51.5%**	**63.9%**

Table 2 summarizes the performance on the MP dataset. It can be seen that although iA* ranked 2nd for map size of 64×64, it achieves outstanding performance for larger maps with sizes 128×128 and 256×256. Specifically, it improves the runtime efficiency *Rt* by 23% for map size 256×256 compared to Neural A*, the SOTA method. This indicates that iA* has an advantage in larger maps, demonstrating the effectiveness of the upper-level predicted cost function for reducing the search space for LL path planning. We observe that FF+PPM and WA*+CF are limited to a fixed shape of input instances due to their integration of the attention mechanism. Their network architectures include fully connected (FC) layers, which require predefined input and output shapes. The number of parameters within these FC layers varies with different input and output shapes. Consequently, adjustments and re-training procedures are necessary to accommodate varying input instance shapes.

Table 3 summarizes the path planning performance on the Matterport3D dataset, where iA* outperforms all methods in all metrics and map sizes. Specifically, iA* achieves a 12.7% and a 37.4% higher performance in search efficiency *Exp* and runtime efficiency *Rt* than the SOTA method neural A*, respectively. It is worth noting that FS+PPM, the 1st ranked method on the 64×64 map category in the MP dataset, achieves the worst performance on the Matterport3D dataset, indicating that FS+PPM is fine-tuned overfitting to the MP dataset and cannot generalize to other types of maps.

Table 4 summarizes the path planning performance on the Maze dataset, which is believed to be the most difficult map

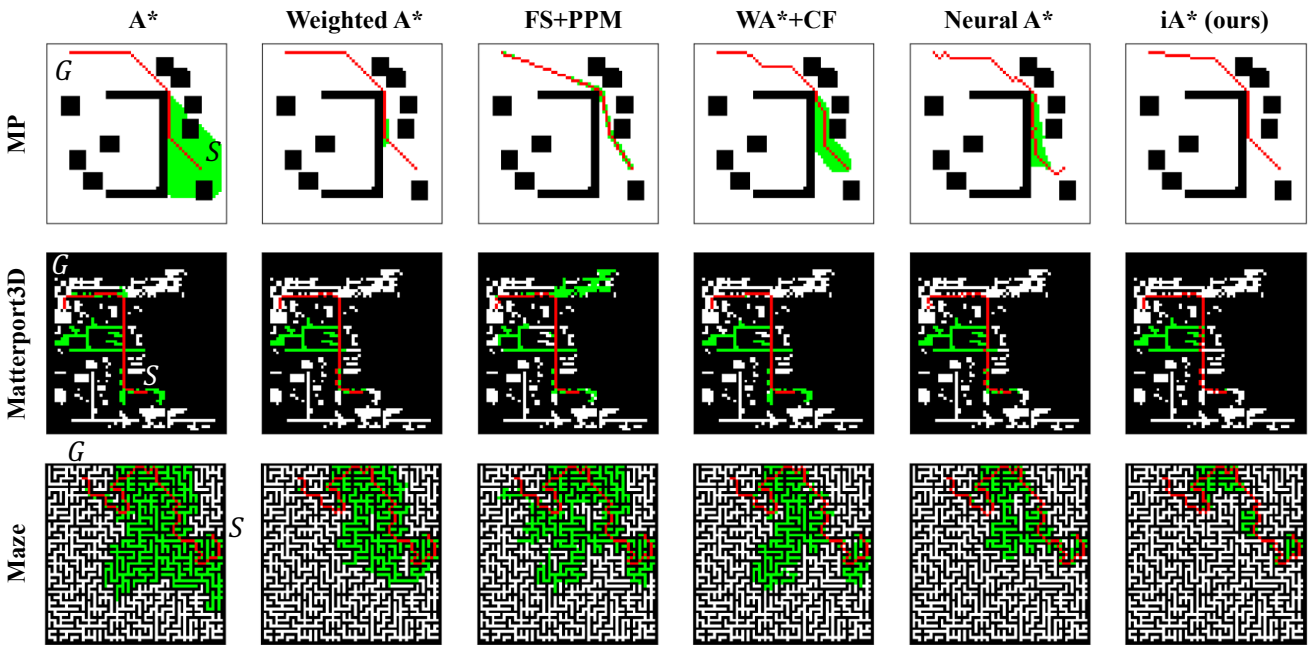


Figure 9. The qualitative results of path planning algorithms on three widely used datasets, including MP, Maze, and Matterport3D. The symbols S and G indicate the randomly selected start and goal positions. The optimal paths found by different path planning algorithms and their associated search space are indicated by red trajectories and green areas, respectively.

type due to its complicated routes. It can be seen that the FS+PPM algorithm performs poorly, even increasing 105% operation time and 8.9% search area than the classical A* search. Additionally, iA* shows the most significant runtime efficiency improvements on large maps, i.e., a 64.2% higher performance in *Rt* than Neural A*. This further demonstrates the generalization ability of the IL framework.

5.1.5 Qualitative Performance We next visualize their qualitative performance of all methods in Figure 9, where each row shows the representative examples from the MP, Matterport3D, and Maze datasets, respectively. Specifically, the green area and the red line represent the search space and optimal paths found by each method, respectively.

For the MP dataset, as shown in the 1st row of Figure 9, it is observed that all the methods have a smaller search space than the classical A* and our iA* has the smallest search space, indicating that iA* achieves the best overall efficiency. It is also observed that the path found by Neural A* is not smooth and all the other data-driven methods show slight differences in terms of path shape and length. This further indicates the effectiveness of our iA* framework.

The 2nd row of Figure 9 displays the path planning results from the Matterport3D dataset, which mainly focuses on indoor scenarios. It can be seen that the search space of FS+PPM and WA*+CF are larger than the other methods. It is worth noting that FS+PPM has a larger search space than classical A*. Besides, all the optimal paths found by different methods are close to the optimal path of classical A*.

In the 3rd row of Figure 9, we show the performance of all baseline methods for the maze-type maps from the Maze dataset. From these examples, it can be observed that all data-driven methods successfully improve the search efficiency than classical A*. Additionally, iA$*$ has the smallest search space, which means that it can even greatly improve the search efficiency of maze-level complicated maps.

Example B: Local Path Planning

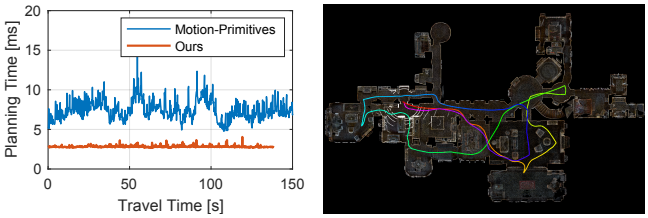
5.1.6 Baselines To evaluate the robustness and efficiency of IL in local path planning, we will compare iPlanner against both classic and data-driven methods. Specifically, we will compare with one of the SOTA classic planners, motion primitives planner (MPP) (Zhang et al. 2020b; Cao et al. 2022), which received the 1st place for most explored sectors in the DARPA Subterranean (SubT) challenge (DARPA). Additionally, we will also compare with data-driven methods, including the supervised learning (SL)-based method (Loquercio et al. 2021) and the reinforcement learning (RL)-based method (Hoeller et al. 2021), which will be simply denoted as SL and RL methods, respectively.

It is worth noting that MPP incorporates a 360° LiDAR as an onboard sensor, while iPlanner and the data-driven methods only use a depth camera, which has a narrower field of view (87°) and noisier range measurements (15 Hz). Notably, the RL method was trained for a downward-tilted (30°) camera, whereas the SL method used a front-facing camera setup. For fairness, iPlanner will be evaluated under both camera setups, i.e., iPlanner and iPlanner (Tilt), although it was only trained for the front-facing camera.

5.1.7 Platforms To thoroughly evaluate the performance of local planning algorithms, we select four distinct simulated environments set up by (Cao et al. 2022), including indoor, garage, forest, and Matterport3D (Chang et al. 2017). In each of the environments, 30 pairs of start and goal positions are randomly selected in traversable areas for a comprehensive evaluation. It is worth noting that none of these environments was observed by iPlanner and the baseline data-driven methods during training. Specifically, a

Table 5. The performance of SPL in local planning (Unit: %).

Method	Forest	Garage	Indoor	Matterport	Overall
MPP (LiDAR)	95.09	**89.42**	85.82	74.18	86.13
SL	65.58	46.70	50.03	28.87	47.80
RL (Tilt)	95.08	69.43	61.10	59.24	71.21
iPlanner (Tilt)	95.66	73.49	68.87	76.67	78.67
iPlanner	**96.37**	88.85	**90.36**	**82.51**	**89.52**



(a) Planning runtime. **(b)** Executed trajectory.

Figure 10. An example of real-time planning using iPlanner. (a) shows the comparison of planning time. (b) shows the trajectory executed by iPlanner and the outline of the environment.

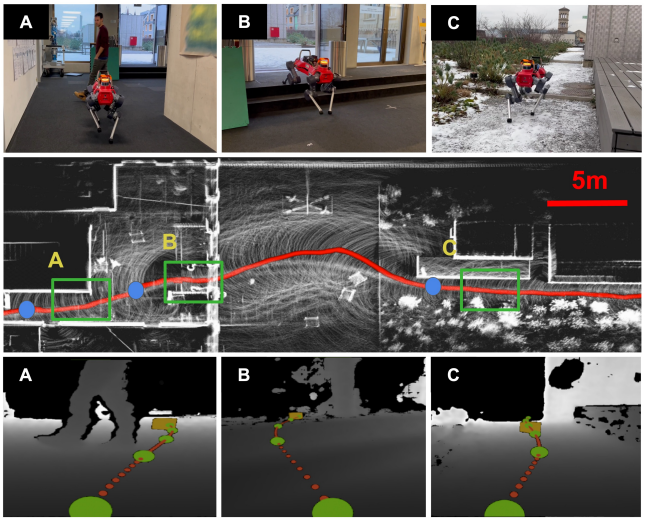


Figure 11. Real-world experiment for local path planning using iPlanner with a legged robot. The red curve indicates the robot's trajectory, beginning inside a building and then navigating to the outdoors. The robot follows a series of waypoints (blue) and plans in different scenarios marked by green boxes including (A) passing through doorways and avoid dynamic obstacles, (B) ascending stairs, and (C) circumvent outdoor static obstacles.

laptop with a 2.6 GHz Intel i9 CPU and an NVIDIA RTX 3080 GPU are used for running all methods.

5.1.8 Metrics The success rate weighted by path length (SPL) will be used for evaluation (Anderson et al. 2018). It has been widely adopted by various path planning systems (Yokoyama et al. 2021; Ehsani et al. 2024).

5.1.9 Quantitative Performance The overall performance of iPlanner and all baseline methods are detailed in Table 5. Notably, iPlanner demonstrates an average performance improvement of 87% over the SL method and 26% over the RL method in terms of SPL. Additionally, iPlanner even outperformed MPP which utilizes LiDAR for perception in most of the environments. This further demonstrates the robustness and reliability of our IL framework.

5.1.10 Efficiency Analysis We evaluate iPlanner's efficiency in Matterport3D. The robot follows 20 manual waypoints for global guidance but autonomously plans feasible paths and avoids local obstacles. As shown in Figure 10a, iPlanner achieves an average 3× speedup over MPP. The executed trajectory is shown in Figure 10b.

5.1.11 Real-world Tests We adopt an ANYmal-legged robot (Hutter et al. 2016) which is equipped with an NVIDIA Jetson Orin and Intel Realsense D435 front-facing depth camera. The real-world experiments involve both dynamic and static obstacles in indoor, outdoor, and mixed settings. The robot is given a sequence of waypoints by a human operator, as illustrated in Figure 11. Specifically, the robot goes through areas including passing through doorways, circumventing both static and dynamic obstacles, and ascending and descending stairs. These trials are designed to test the adaptability and reliability of iPlanner under varied and unpredictable environmental conditions. During the test, iPlanner can generate feasible paths online and guide the robot through all areas safely and smoothly, which demonstrates the generalization ability of our IL framework.

5.2 Logical Reasoning

We conduct the task of visual action prediction for multiple agents in traffic to assess the efficacy of our IL framework.

5.2.1 Datasets The VAP task in LogiCity (Li et al. 2024) integrates concept grounding and logical reasoning involving agent concepts and spatial relationships. It is particularly more challenging than other logical reasoning benchmarks with structured low-dimensional data, such as BlocksWorld (Dong et al. 2019) or knowledge graphs (Cropper and Morel 2021). The main reasons are that LogiCity requires learning abstract concepts from varying agent distributions in an urban setting; additionally, it also involves high-dimensional image data with diverse visual appearances, however, as a comparison, BlocksWorld only contains binary vectors.

To be specific, this benchmark comprises 2D RGB renderings for multiple types of agents whose actions are constrained by first-order logic (FOL) rules. The training set consists of 100 different urban environments and 12 agents, the validation set includes 20 environments and 14 agents, and the test set contains 20 environments and 14 agents. Each environment encompasses 100 stimulation steps, with agent types remaining consistent within each world of the same subset. Notably, the groups of 14 agents in the validation and test sets are distinct, which aims to demonstrate the models' generalization abilities. According to the difficulty, the task is divided into two modes, i.e., the *Easy* and *Hard*. In each timestep, agents perform one of four actions: `Slow`, `Normal`, `Fast`, and `Stop`. The *Easy* mode excludes the `Fast` action, while the *Hard* mode includes all four actions, leading to the actions constrained by more complicated rules.

Actions in the VAP task are imbalanced; for instance, `Fast` actions in the *Hard* model occur only about $1/4$ to $1/9$ as often as other actions, complicating the rule induction.

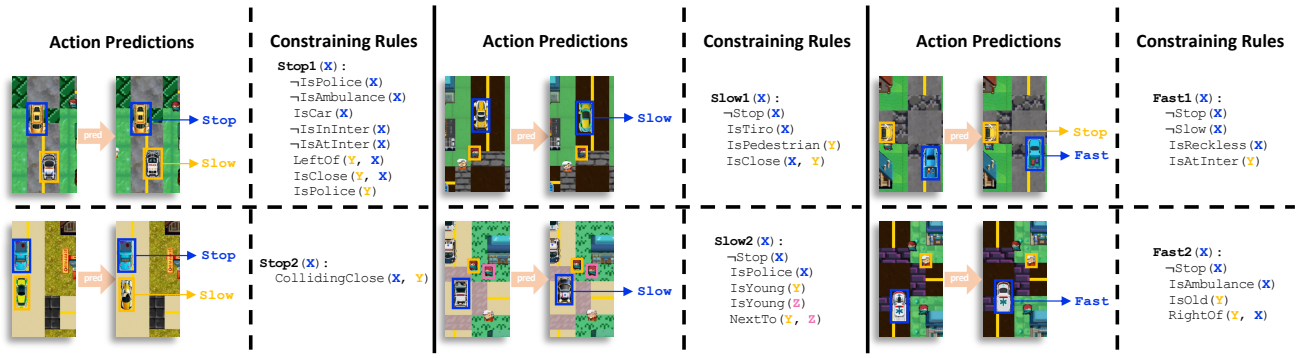


Figure 12. To predict each action, iLogic conducts rule induction with perceived groundings and the constraining rules exhibited on the right side and finally gets the accurate actions exhibited on the left side.

Table 6. Evaluation results of the *Easy* mode in LogiCity.

Metric / Action	Recall Rate Slow	Normal	Stop	wAcc Overall	Var(wAcc) Overall
ResNet+NLM	0.671	0.644	0.995	0.724	0.00128
iLogic (ours)	0.678	0.679	0.995	**0.740**	**0.00088**

Table 7. Evaluation results of the *Hard* mode in LogiCity.

Metric / Action	Recall Rate Slow	Normal	Fast	Stop	wAcc Overall	Var(wAcc) Overall
ResNet+NLM	0.552	0.575	0.265	0.999	0.400	0.00252
iLogic (ours)	0.459	0.598	0.338	0.999	**0.442**	**0.00083**

5.2.2 Metrics To evaluate the model performance thoroughly, we utilize the weighted accuracy (wAcc), the variance of wAcc, and the prediction recall rate of each action as the metrics. Particularly, wAcc places greater emphasis on infrequently occurring data, reflecting the generalization ability; the variance of wAcc implies the optimization stability of the algorithms; and the recall rate assesses the accuracy with which a model learns the rules constraining each action.

5.2.3 Baselines & Implementation Details Since very little literature has studied FOL reasoning on RGB images, we don't have many baselines. As a result, we adopt the method proposed in LogiCity (Li et al. 2024) that involves the SOTA visual encoder and reasoning engines. Particularly, we conduct experiments with two methods: (1) ResNet+NLM: ResNet as a visual encoder and NLM as the reasoning engine; (2) iLogic (ours): We train the "ResNet+NLM" within the IL framework, where the NLM is the LL problem. Specifically, the visual encoder includes a concept predictor and a relationship predictor. The concept predictor involves a ResNet FPN and an ROI-Align layer, followed by a 4-layer MLP to predict agent concepts (unary predicates). The relationship predictor includes a 3-layer MLP to encode paired agent information into agent relationships (binary predicates). The reasoning engine involves an NLM with breath as 3 and depth as 4. Two AdamW optimizers with a learning rate of 0.001 are adopted for the visual encoder and the reasoning engine, respectively.

5.2.4 Evaluation Results For fairness, we mitigate the effect of randomness on the training process by running 10 times for each experiment and reporting their average performance. As is displayed in Table 6 and Table 7, the prediction recall rate of most actions using iLogic is higher than or equal to those of the baseline in both *Easy* and *Hard* mode. This proves that our iLogic can learn rules that constrain most actions more effectively.

Given the imbalanced number of actions, it is challenging to induct all the constrained rules simultaneously. However, iLogic exhibits a marked improvement in wAcc, with 2.2% and 10.5% higher performance in *Easy* and *Hard* mode, respectively, demonstrating its stronger generalization ability. Besides, the variance of wAcc is 31.3% and 67.1% lower in *Easy* and *Hard* mode, respectively, which indicates higher optimization stability using the IL framework.

We present several qualitative examples in Figure 12, showing how actions are predicted from the learned rules. Since the actions are constrained by explicit rules in the LogiCity, a model has to learn these rules correctly to predict the actions. It can be seen that iLogic can induct the rules with perceived groundings on the right side and obtain the accurate actions exhibited on the left side of each example.

5.2.5 Efficiency Analysis Due to the complicated BLO, the training time for iLogic is about 40% longer than the supervised learning "ResNet + NLM", and this extra cost is well justified by the significant performance gain in Table 6 and Table 7. However, there is no additional computational cost in inference, with the system achieving a speed of approximately 19.78 frames/second on an RTX 3090 GPU.

5.3 Optimal Control

To validate the effectiveness of the iMPC framework, we test it on the attitude control for a quadrotor UAV. Specifically, we will show its ability to stabilize (hover) the UAV with different initial conditions under the effects of wind gusts.

5.3.1 Baselines To demonstrate the effectiveness of the IL framework for jointly optimizing the data-driven IMU model and MPC, we select four baselines including (1) IMU+MPC: classic IMU integrator with a regular Diff-MPC; (2) IMU$^+$+MPC: data-driven IMU model with a regular Diff-MPC; (3) IMU+MPC$^+$: classic IMU integrator with a Diff-MPC with learnable moment of inertia (MOI); and (4) iMPC (ours): IMU$^+$ with MPC$^+$ trained with IL framework. Specifically, "IMU$^+$" refers to the data-driven IMU denoising and integration model from (Qiu et al. 2024). Additionally, for the wind disturbance test, we select an RL

Table 8. The performance of UAV attitude control under different initial conditions. The "IMU" means the RMSE (unit: $\times 10^{-3}$rad) of attitude estimation for the corresponding method.

Initial	Methods	*ST* (s)	*RMSE* (°)	*SSE* (°)	IMU
10°	IMU+MPC	0.287	0.745	0.250	7.730
	IMU$^+$+MPC	0.281	0.692	0.193	6.470
	IMU+MPC$^+$	0.283	0.726	0.216	7.370
	iMPC (ours)	**0.275**	**0.691**	**0.185**	**6.310**
15°	IMU+MPC	0.330	0.726	0.262	8.390
	IMU$^+$+MPC	0.299	0.691	0.213	7.020
	IMU+MPC$^+$	0.321	0.721	0.230	7.980
	iMPC (ours)	**0.296**	**0.688**	**0.201**	**6.800**
20°	IMU+MPC	0.336	0.728	0.263	8.370
	IMU$^+$+MPC	0.318	0.685	0.217	7.270
	IMU+MPC$^+$	0.324	0.715	0.224	8.090
	iMPC (ours)	**0.317**	**0.684**	**0.208**	**7.010**

approach with a commonly used approach, proximal policy optimization (PPO) (Schulman et al. 2017) as the baseline.

5.3.2 Implementation Details Due to efficiency considerations, we use a lightweight network for the data-driven IMU model, which includes a 2-layer multi-layer perceptron (MLP) as the encoder to get the feature embeddings from both the accelerometer and the gyroscope. Two decoders are implemented each comprising two linear layers to address the specific corrections needed for the accelerometer and the gyroscope. These decoders are dedicated to accurately predicting the necessary adjustments for each sensor type, thereby enhancing the overall precision of the IMU readings in dynamic conditions. For both the encoder and the decoder, we use `GELU` as the activation function for its effectiveness in facilitating smoother non-linear transformations. For RL, we take pre-integrated IMU measurements as the observation, output UAV action (thrust and torques) in OpenAI gym (Brockman et al. 2016), and designed the reward to minimize the difference between the current and target UAV pose.

5.3.3 Simulation Environment To accurately measure the performance of different controllers, we build a simulation environment for a quadrotor UAV with a standard 6 degree-of-freedom (DoF) dynamics model running at 1 KHz (Gabriel et al. 2007; Abdellah and Abdelaziz 2004) and an onboard IMU sensor running at 200 Hz. To simulate real-world effects, such as environmental disturbance, actuator uncertainty, sensor noise, and other unknown behaviors, we inject zero-mean Gaussian noise with a standard deviation of 1e-4 to the control input and zero-mean Gaussian noise with a standard deviation of 8.73e-2 to the UAV attitude at 1KHz. Additionally, we employ the sensor noise models documented by the `Epson G365` IMU, including initial bias, bias instability, and random walk for both the accelerometer and gyroscope. This produces the same noise level as a typical UAV tracking system using a visual-inertial odometry system (Xu et al. 2023) for attitude estimation.

5.3.4 Metrics To evaluate the performance of a controller, we use three widely-used metrics, including settling time (*ST*), root-mean-square error (*RMSE*), and steady-state error (*SSE*). Specifically, *ST* is the time for the UAV to enter and remain within $\pm 1.5°$ of its final steady attitude, measuring how quickly the system settles; *RMSE* is the root-mean-square of the difference between estimated and the desired attitude; and *SSE* is the absolute difference between the steady and desired attitude, evaluating the control accuracy.

Table 9. The learned UAV MOI error using iMPC under different initial conditions with an initial MOI of 50% offset.

Initial Offset	Initial	Error	Initial	Error	Initial	Error
50%	10°	2.67%	15°	3.41%	20°	2.22%

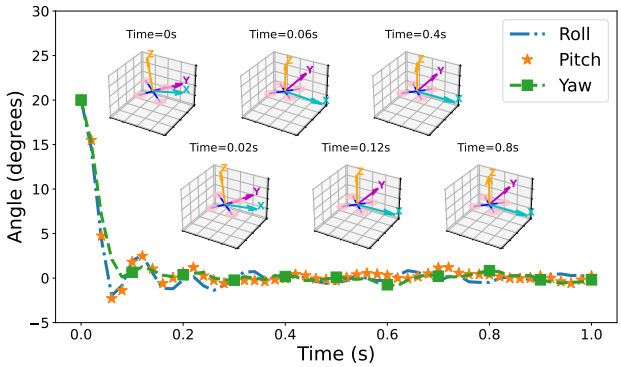
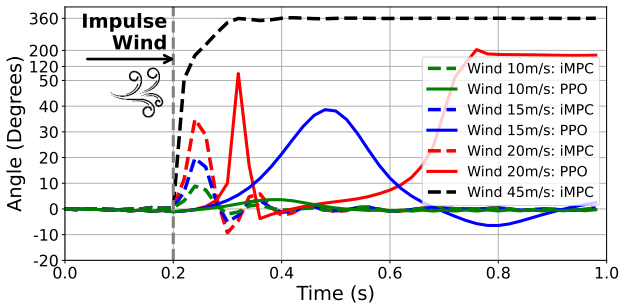


Figure 13. The UAV attitude quickly returns to a stable (zero) state for an initial condition of 20° using iMPC as the controller.

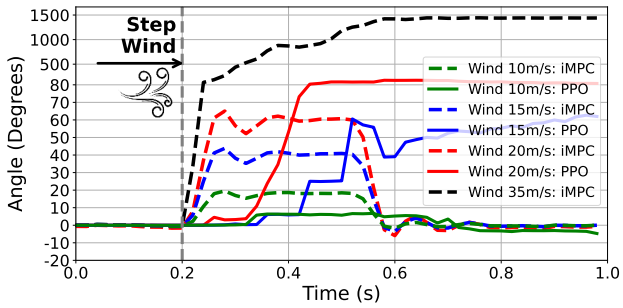
5.3.5 Robustness to Initial Conditions We first evaluate the performance of the UAV attitude control under different initial conditions. The performance of different controllers is shown in Table 8 with an initial attitude of 10° to 20° of all three Euler angles (roll, pitch, and yaw). Each experiment was repeated 10 times to ensure the accuracy and reliability of the results. It is clear that our iMPC can stabilize the UAV for all initial conditions within the test range. It is also worth noting that all methods show a stable performance with a small standard deviation thus we omit those numbers in the table. Compared to the baseline methods, iMPC ends up with less *SSE*, *RMSE*, and *ST*. Figure 13 shows an example that the UAV attitude quickly returns to a stable (zero) state for an initial condition of 20° using our iMPC.

To investigate how the lower-level MPC helps the upper-level IMU learning, we show the IMU attitude estimation performance in the last column of Table 8. We can see that due to the physics knowledge from dynamics and as well as the lower-level MPC, the denoising and prediction performance of the IMU module get significantly improved compared to the separately trained case. Therefore, the imperative learning framework enforces the improvement of both the IMU network and the final control performance.

Since the UAV MOI is learned in iMPC, it is important to show the final estimated MOI is close to the real value after the IL learning process. To illustrate this, we set an initial MOI with 50% offset from its true value and show the final estimated MOI error using iMPC in Table 9. It can be seen that iMPC can learn the MOI with a final error of less than 3.5%. Additionally, if we jointly consider the performance in Table 8 and Table 9 and compare iMPC with IMU$^+$+MPC, and IMU+MPC$^+$ with IMU+MPC, where "MPC$^+$" indicates the learned MOI is in the control loop, we can conclude that



(a) UAV pitch angle response when encountering an **impulse wind** disturbance at 0.2 s for different speeds. PPO loses control at a wind speed of 15 m/s, while iMPC loses control at 45 m/s.



(b) UAV pitch angle response when the encountering a **step wind** disturbance at 0.2 s and lasting for 0.3 s for different speeds. PPO loses control at a wind speed of 15 m/s, while iMPC loses control at 35 m/s.

Figure 14. Control performance of iMPC and PPO under different levels of wind disturbance.

Table 10. UAV attitude control performance under **impulse wind**, where '✗' means losing control at that wind speed.

Wind	Methods	*ST* (s)	*RMSE* (°)	*SSE* (°)	IMU
10 m/s	IMU+MPC	0.406	0.360	0.169	9.190
	IMU$^+$+MPC	0.392	0.355	0.142	9.010
	IMU+MPC$^+$	0.406	0.360	0.168	9.180
	RL (PPO)	0.574	1.730	1.426	✗
	iMPC (ours)	**0.368**	**0.353**	**0.132**	**8.660**
15 m/s	IMU+MPC	0.454	0.328	0.120	8.160
	IMU$^+$+MPC	0.450	0.316	0.065	7.730
	IMU+MPC$^+$	0.454	0.325	0.117	8.140
	RL (PPO)	✗	✗	✗	✗
	iMPC	**0.430**	**0.312**	**0.060**	**7.570**
20 m/s	IMU+MPC	0.486	0.361	0.186	6.550
	IMU$^+$+MPC	0.476	0.356	0.139	6.330
	IMU+MPC$^+$	0.484	0.361	0.185	6.520
	RL (PPO)	✗	✗	✗	✗
	iMPC	**0.470**	**0.354**	**0.135**	**6.160**

Table 11. UAV attitude control performance under **step wind**, where '✗' means losing control at that wind speed.

Wind	Methods	*ST* (s)	*RMSE* (°)	*SSE* (°)	IMU
10 m/s	IMU+MPC	0.596	0.396	0.189	6.870
	IMU$^+$+MPC	0.594	0.373	0.183	6.110
	IMU+MPC$^+$	0.588	0.392	0.179	6.830
	RL (PPO)	0.754	1.882	1.478	✗
	iMPC (ours)	**0.574**	**0.352**	**0.168**	**5.990**
15 m/s	IMU+MPC	0.684	0.337	0.172	7.190
	IMU$^+$+MPC	0.670	0.309	0.167	6.740
	IMU+MPC$^+$	0.684	0.337	0.171	7.180
	RL (PPO)	✗	✗	✗	✗
	iMPC (ours)	**0.620**	**0.297**	**0.149**	**6.730**
20 m/s	IMU+MPC	0.704	0.376	0.240	6.300
	IMU$^+$+MPC	0.690	0.317	0.173	6.120
	IMU+MPC$^+$	0.704	0.372	0.253	6.300
	RL (PPO)	✗	✗	✗	✗
	iMPC (ours)	**0.676**	**0.311**	**0.150**	**6.110**

a better learned MOI can lead to a better attitude control performance, with smaller SSE and settling time.

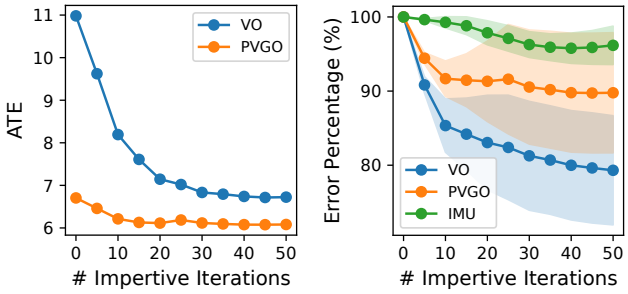
5.3.6 Robustness to Wind Disturbance We next evaluate the control performance under the wind disturbance to validate the robustness of the proposed approach. Specifically, we inject the wind disturbance by applying an external force $F = \frac{1}{2} \cdot C_d \cdot \rho \cdot A \cdot V^2$ and torque $\tau = r \times F$ to the nominal system based on the drag equation, where C_d is the drag coefficient, ρ is the air density, A is the frontal area of the object facing the wind, V is the wind speed, r is the length of lever arm. Hence, for a fixed UAV configuration, different wind speeds V lead to different disturbance levels. We conduct two sets of experiments: in the first set, we add impulse wind gust to the UAV in the direction of opposite heading during the hover with varying wind speed $10/15/20/45$ m/s; in the second set, a step wind is added to the system in the same direction during hover and lasts for 0.3 s for different wind speed $10/15/20/35$ m/s. The external wind starts at 0.2 s after entering a steady hover. During RL (PPO) training, we apply external wind disturbances of up to 10m/s. In contrast, our hybrid approach with MPC provides robustness without requiring exposure to a wide range of wind conditions during training.

The pitch angle responses of the UAV using iMPC under the two different wind disturbances are shown in Figure 14a and Figure 14b, respectively. It can be seen that the wind affects the UAV attitude with different peak values under different wind speeds. The vehicle can resist wind and return to hover quickly even under 20m/s^2 impulse or step wind. Note that the maximum wind speed the system can sustain under step wind is smaller than that under impulse wind due to the long duration of the disturbance. When the wind speed is too high, the UAV can no longer compensate for the disturbance and the UAV flipped and finally crashed.

Detailed comparisons with other approaches are presented in Table 10 and Table 11 for the two scenarios with different disturbances, respectively. It is evident that iMPC leads to better attitude control performance with smaller *SSE* and a faster *ST*. Particularly compared to IMU+MPC and IMU$^+$+MPC, iMPC reduces noises with a more accurate attitude state prediction. Additionally, iMPC further reduces the IMU prediction error in terms of RMSE. Therefore, even in the presence of significant external wind disturbances, iMPC can still control the vehicle attitude robustly. This demonstrates that the IL framework can simultaneously enhance both the MPC parameter learning and IMU denoising and prediction. Compared to the RL baseline PPO, iMPC demonstrates significantly better performance, especially under wind disturbances that were not encountered during training. While PPO performs

Table 12. The average RMSE drifts on KITTI dataset. r_{rel} is rotational RMSE drift (°/100 m), t_{rel} is translational RMSE drift (%) evaluated on various segments with length 100–800 m.

Methods	Inertial	Supervised	r_{rel}	t_{rel}
DeepVO	✗	✓	5.966	5.450
UnDeepVO	✗	✗	2.394	5.311
TartanVO	✗	✓	3.230	6.502
DROID-SLAM	✗	✓	**0.633**	5.595
iSLAM (VO Only)	✗	✗	1.101	**3.438**
Wei et al. (2021)	✓	✗	0.722	5.110
Yang et al. (2022a)	✓	✓	0.863	2.403
DeepVIO	✓	✗	1.577	3.724
iSLAM (ours)	✓	✗	**0.262**	**2.326**



(a) Decrease of the ATE. **(b)** Decrease of error percentage.

Figure 15. (a) The ATE of our VO and PVGO w.r.t. number of imperative iterations. (b) The decrease in error percentage. The error before imperative learning is treated as 100%. The ATE is used for VO and PVGO to calculate the percentage, while the relative displacement error is used for IMU. The solid lines are mean values while the transparent regions are the variances.

similarly under wind conditions seen during training, as shown in Figure 14a and Figure 14b, it completely loses control of the UAV when faced with out-of-distribution disturbances. These results further highlight the robustness of the IL framework, benefiting from a hybrid approach that integrates prior physical knowledge.

5.3.7 Efficiency Analysis Compared to a conventional approach of IMU+MPC, iMPC achieves similar efficiency because the MPC dominates the total computation time and the overhead from the learning-based IMU inference is negligible. In fact, iMPC runs at 50 Hz control frequency with a 200 Hz IMU measurement, which is at the same level as standard MPC. On the other hand, when compared to RL, although the inference time of iMPC is longer as expected due to the additional requirement of solving an online optimization, such physics-based optimization also brings prior knowledge into the training process, thereby overcoming the typical low sampling efficiency issue of RL. In fact, the experiment shows the PPO needs an extra 221.28% time for training due to the low sampling efficiency.

5.4 SLAM

We next provide a comprehensive evaluation of the iSLAM framework in terms of estimation accuracy and runtime. We will also provide assessments on the unique capabilities brought by IL, including the mutual enhancement between the front-end and the back-end of a SLAM system and its self-supervised adaptation to new environments.

5.4.1 Datasets To ensure thorough evaluation, we adopt two widely-used datasets, including KITTI (Geiger et al. 2013) and EuRoC (Burri et al. 2016). They have diverse environments and motion patterns: KITTI incorporates high-speed long-range movements in driving scenarios, while EuRoC features aggressive motions in indoor environments.

5.4.2 Metrics Following prior works (Qiu et al. 2022), we use the widely-used absolute trajectory error (ATE), relative motion error (RME), and root mean square error (RMSE) of rotational and translational drifts as evaluation metrics.

5.4.3 Accuracy Evaluation We first assess the localization accuracy of iSLAM on the KITTI dataset, which has been widely used in previous works on various sensor setups. To facilitate a fair comparison, in Table 12, we evaluate our standalone VO component against existing VO networks, including DeepVO (Wang et al. 2017b), UnDeepVO (Li et al. 2018), TartanVO (Wang et al. 2021b), and DROID-SLAM (Teed and Deng 2021), and compare the full iSLAM to other learning-based visual-inertial methods, which are Wei et al. (2021), Yang et al. (2022a), and DeepVIO (Han et al. 2019). Sequences 00 and 03 are omitted in our experiment since they lack completed IMU data. Notably, some methods such as DeepVO and (Yang et al. 2022a) were supervisedly trained on KITTI, while our iSLAM was self-supervised. Even though, iSLAM outperforms all the competitors. Additionally, it is noteworthy that our base model, TartanVO, doesn't exhibit the highest performance due to its lightweight design. Nevertheless, through imperative learning, we achieve much lower errors with a similar model architecture.

We also test iSLAM on the EuRoC benchmark, where the presence of aggressive motion, substantial IMU drift, and significant illumination changes pose notable challenges to SLAM algorithms (Qin et al. 2018). However, both our standalone VO and the full iSLAM generalize well to EuRoC. As shown in Table 13, iSLAM achieves an average ATE 62% lower than DeepV2D (Teed and Deng 2018), 76% lower than DeepFactors (Czarnowski et al. 2020), and 42% lower than TartanVO (Wang et al. 2021b).

5.4.4 Efficiency Analysis Efficiency is a crucial factor for SLAM systems in real-world robot applications. Here we conduct the efficiency assessments for iSLAM on an RTX4090 GPU. Our stereo VO can reach a real-time speed of 29-31 frames per second (FPS). The scale corrector only uses about 11% of inference time, thus having minimal impact on the overall efficiency. The IMU module achieves an average speed of 260 FPS, whereas the back-end achieves 64 FPS, when evaluated independently. The entire system operates at around 20 FPS. Note that the imperative learning framework offers broad applicability across different front-end and back-end designs, allowing for great flexibility in balancing accuracy and efficiency. We next measure the runtime of our "one-step" back-propagation strategy against the conventional unrolling approaches (Tang and Tan 2018; Teed and Deng 2021) during gradient calculations. A significant runtime gap is observed: our "one-step" strategy is on average 1.5× faster than the unrolling approach.

Table 13. Absolute Trajectory Errors (ATE) on the EuRoC dataset.

Methods	MH01	MH02	MH03	MH04	MH05	V101	V102	V103	V201	V202	V203	**Avg**
DeepV2D	0.739	1.144	0.752	1.492	1.567	0.981	0.801	1.570	**0.290**	2.202	2.743	1.354
DeepFactors	1.587	1.479	3.139	5.331	4.002	1.520	0.679	0.900	0.876	1.905	**1.021**	2.085
TartanVO	0.783	**0.415**	0.778	1.502	1.164	0.527	0.669	0.955	0.523	0.899	1.257	0.869
iSLAM (VO Only)	0.320	0.462	0.380	0.962	0.500	0.366	0.414	0.313	0.478	0.424	1.176	0.527
iSLAM (ours)	**0.302**	0.460	**0.363**	**0.936**	**0.478**	**0.355**	**0.391**	**0.301**	0.452	**0.416**	1.133	**0.508**

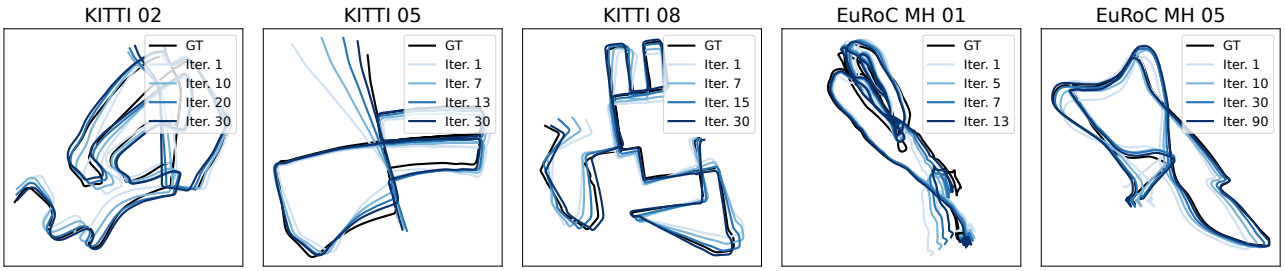


Figure 16. The predicted trajectories from the front-end are improved concerning the number of imperative iterations in iSLAM.

5.4.5 Effectiveness Validation We next validate the effectiveness of imperative learning in fostering mutual improvement between the front-end and back-end components in iSLAM. The reduction of ATE and error percentage w.r.t. imperative iterations is depicted in Figure 15. One imperative iteration refers to one forward-backward circle between the front-end and back-end over the entire trajectory. As observed in Figure 15a, the ATE of both VO and PVGO decreases throughout the learning process. Moreover, the performance gap between them is narrowing, indicating that the VO model has effectively learned geometry knowledge from the back-end through BLO. Figure 15b further demonstrates that imperative learning leads to an average error reduction of 22% on our VO network and 4% on the IMU module after 50 iterations. Meanwhile, the performance gain in the front-end also improves the PVGO result by approximately 10% on average. This result confirms the efficacy of mutual correction between the front-end and back-end components in enhancing overall accuracy.

The estimated trajectories are visualized in Figure 16. As observed, the trajectories estimated by the updated VO model are much closer to the ground truth, which further indicates the effectiveness of the IL framework for SLAM.

5.5 Min-Max MTSP

We next demonstrate our iMTSP in generalization ability, computational efficiency, and convergence speed.

5.5.1 Baselines To thoroughly evaluate the performance of different methods for MTSP, we compare our iMTSP with both classic methods and learning-based methods. This includes a well-known meta-heuristic method implemented by the Google OR-Tools routing library (Furnon and Perron 2023) and an RL-based method (Hu et al. 2020), which will be referred to as Google OR-Tools and RL baseline for abbreviation, respectively. Specifically, Google OR-Tools can provide near-optimal solutions[‡] to MTSPs with a few hundred cities, while the RL baseline can solve larger-scale MTSPs such as with one thousand cities. We cannot compare with other methods such as (Liang et al. 2023; Gao et al. 2023; Park et al. 2021) because they either don't provide source code or cannot be applied to our settings.

5.5.2 Datasets Most of the existing approaches can handle MTSPs with about 150 cities, but their performance will significantly compromise with 400 or more cities. To demonstrate iMTSP's ability to generalization and handle large-scale problems, we conduct experiments on training sets with 50 to 100 cities but test the models on problems with 400 to 1000 cities. We believe this challenging setting can reflect the generalization ability of the proposed models.

Due to the uniqueness of MTSP, we cannot deploy MTSP algorithms on such a large-scale robot team, thus we build a simulation for comparison. Specifically, all the data instances are generated by uniformly sampling points in a unit rectangular so that both x and y coordinates are in the range of 0 to 1. The test set consists of i.i.d. samples but with a larger number (400 to 1000) of cities as aforementioned. Note that the proposed approach is applicable to general graphs with arbitrary costs assigned to edges, although our dataset is constrained in the 2-D space.

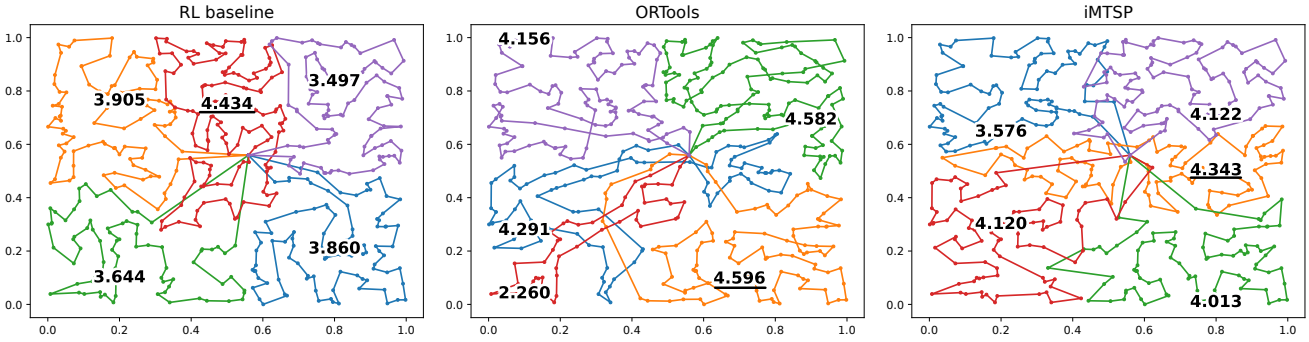
5.5.3 Implementation Details The structure of our allocation network is similar to that in (Hu et al. 2020), where the cities and agents are embedded into 64 dimensional vectors. The surrogate network is a three-layer MLP whose hidden dimension is 256 and whose activation function is "`tanh`". Additionally, Google OR-Tools provides the TSP solver in our iMTSP. It uses the "Global Cheapest Arc" strategy to generate its initial solutions and uses "Guided local search" for local improvements. We use the best-recommended settings in Google OR Tools for all experiments. All tests were conducted locally using the same desktop-level computer.

5.5.4 Quantitative Performance We next present quantitative evidence of iMTSP's superior solution quality, particularly in maximum route length, as detailed in Table 14. It can be seen that iMTSP achieves up to 80% shorter maximum

[‡]Google OR-Tools is a meta-heuristic and cannot guarantee the optimality of solutions. However, it is efficient for small-scale problems and wildly used for academic and commercial purposes. Thus, in our experiments, it can be seen as an "ideal" solver providing "optimal" solutions.

Table 14. Performance of different methods on large-scale MTSPs. iMTSP significantly outperforms the Google OR-Tools (Furnon and Perron 2023) and provides an average of $3.2 \pm 0.01\%$ shorter maximum route length than the RL baseline (Hu et al. 2020).

Training Setting			# Test Cities							Avg gap
Model	# Agents	# Cities	400	500	600	700	800	900	1000	
Google OR-Tools	10	–	10.482	10.199	12.032	13.198	14.119	18.710	18.658	277.1%
RL baseline	10	50	3.058	3.301	3.533	3.723	3.935	**4.120**	**4.276**	0.2%
iMTSP (ours)	10	50	**3.046**	**3.273**	**3.530**	**3.712**	**3.924**	4.122	4.283	0
Google OR-Tools	10	–	10.482	10.199	12.032	13.198	14.119	18.710	18.658	290.1%
RL baseline	10	100	3.035	3.271	3.542	3.737	3.954	4.181	4.330	4.3%
iMTSP (ours)	10	100	**3.019**	**3.215**	**3.424**	**3.570**	**3.763**	**3.930**	**4.042**	0
Google OR-Tools	15	–	10.293	10.042	11.640	13.145	14.098	18.471	18.626	349.9%
RL baseline	15	100	2.718	2.915	3.103	3.242	3.441	3.609	3.730	6.3%
iMTSP (ours)	15	100	**2.614**	**2.771**	**2.944**	**3.059**	**3.221**	**3.340**	**3.456**	0



(a) RL baseline. **(b)** OR-Tools with 1800s runtime. **(c)** iMTSP.

Figure 17. The performance visualization of the baselines and iMTSP for a problem instance with a central depot, 5 agents, and 500 cities. The numbers denote the length of the route, and different colors represent different routes. The longest tour lengths are underlined and iMTSP has the best solution and has fewer sub-optimal patterns like circular partial routes or long straight routes.

Table 15. Computing time of iMTSP dealing with 1000 cities. Due to a similar network structure, the RL baseline (Hu et al. 2020) has roughly the same runtime efficiency compared with ours, while Google OR-Tools (Furnon and Perron 2023) cannot find feasible solutions within a given time budget.

Number of Agents	5	10	15
Computing Time	4.85 s	1.98 s	1.35 s

route length than Google OR-Tools which cannot converge to a local minimum within the given 300 seconds time budget.

On the other hand, iMTSP provides a better solution in most cases than the RL baseline. On average, iMTSP has $3.2 \pm 0.01\%$ shorter maximum route length over the RL baseline. When the models are trained with 100 cities, the difference between the route lengths monotonically increases from 0.4% to 8.0% and from 3.4% to 8.9%, respectively with 10 and 15 agents. The results demonstrate that with the lower-variance gradient provided by our control variate-based optimization algorithm, iMTSP usually converges to better solutions when being trained on large-scale instances.

5.5.5 Efficiency Analysis Since iMTSP contains both a data-driven allocation network and classic TSP solvers, it is important to identify the bottleneck of the architecture for future improvements. As in Table 15, with 5, 10, and 15 agents, the computing time of our model are 4.85 s, 1.98 s, and 1.35 s, respectively, to solve one instance with 1000 cities. Note that the computing time decreases as the number of agents increases. This result indicates that the major computational bottleneck is the TSP solvers rather than the allocation network because more agents indicate more parameters in the allocation network but less average number of cities per agent. One possible direction to further reduce the computing time of iMTSP is to create multiple threads to run the TSP solvers in parallel since the TSPs in the lower-level optimization of iMSTP are independent.

5.5.6 Qualitative Performance We next conduct qualitative analysis for the MTSP solvers in Figure 17, where different colors indicate different routes. It is observed that sub-optimal patterns exist for both baselines but very few of them exist in that of iMTSP. For example, we can observe circular partial routes for Google OR-Tools such as the green and purple routes, where the route length can be reduced by simply de-circle at the overlapping point. This observation provides further intuitions about the advantages of iMTSP.

5.5.7 Gradient Variance As demonstrated in Section 4.5, our control variate-based iMTSP framework is expected to have a smaller gradient variance than the RL baseline. We verify such a hypothesis by explicitly recording the mini-batch gradient variance during the training process. The experiment is conducted twice with 10 agents respectively visiting 50 and 100 cities. The training data with a batch size of 512 is divided into several mini-batches containing 32 instances. We compute and store the gradients for every

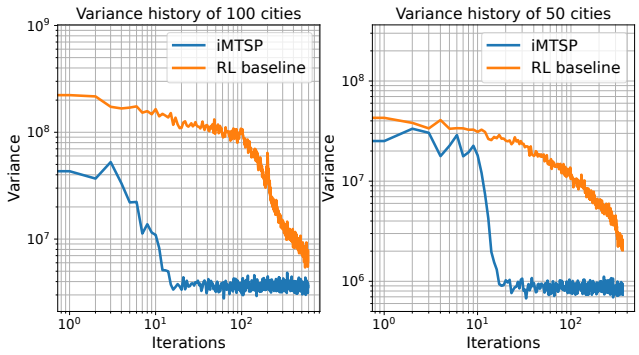


Figure 18. The gradient variance history of our method and the RL baseline during the training process with 50 and 100 cities. The vertical axis is the logarithmic variance of the allocation network gradient, and the horizontal axis denotes the number of training iterations. Our control-variate-base gradient estimator converges about 20× faster than the RL baseline.

mini-batch, and then calculate their variance for the entire batch. The parameters of the allocation network are still updated with the average gradient of the whole batch. The variance of the gradient with respect to the training process is shown in Figure 18, where the horizontal axis represents the number of training iterations and the vertical axis denotes the mean logarithm gradient variance. It is worth noting that the gradient of our iMTSP converges about 20× faster than the RL baseline (Hu et al. 2020), which verifies the effectiveness of our control variate-based BLO process.

6 Conclusions & Discussions

We introduced imperative learning, a self-supervised neuro-symbolic learning framework aimed at enhancing robot autonomy. Imperative learning provides a systematic approach to developing neuro-symbolic systems by integrating the expressiveness of neural methods with the generalization capabilities of symbolic methods. We presented five distinct applications of robot autonomy: path planning, rule induction, optimal control, visual odometry, and multi-agent routing. Each application exemplifies different optimization techniques for addressing lower-level problems, namely closed-form solutions, first-order optimization, second-order optimization, constrained optimization, and discrete optimization, respectively. These examples demonstrate the versatility of imperative learning and we expect they can stimulate further research in the field of robot autonomy.

Similar to other robot learning frameworks, imperative learning has its drawbacks and faces numerous unresolved challenges. Robotics often involves highly nonlinear real-world problems, where theoretical assumptions for bilevel optimizations do not always hold, leading to a lack of theoretical guarantees for convergence and stability, despite its practical effectiveness in many tasks. Additionally, unlike data-driven models, the training process for bilevel optimization requires careful implementation. Moreover, applying imperative learning to new problems is not straightforward. Researchers need to thoroughly understand the problems to determine the appropriate allocation of tasks to the respective modules (neural, symbolic, or memory), based on the strengths of each module. For instance, a neural model might excel in tasks involving dynamic obstacle detection and classification, while a symbolic module could be more suitable for deriving and optimizing multi-step navigation strategies based on rules and logical constraints.

We believe that the theoretical challenges of imperative learning will also inspire new directions and topics for fundamental research in bilevel optimization. For example, second-order bilevel optimization remains underexplored in machine learning due to time and memory constraints. However, its relevance is increasingly critical in robotics, where second-order optimization is essential for achieving the required accuracy in complex tasks. Furthermore, handling lower-level constraints within general bilevel optimization settings presents significant challenges and remains an underdeveloped area. Recent advancements, such as those based on proximal Lagrangian value functions (Yao et al. 2024a, 2023), offer potential solutions for constrained robot learning problems. We intend to investigate robust approaches and apply these techniques to enhance our experimental outcomes for imperative learning. Additionally, we plan to develop heuristic yet practical solutions for bilevel optimization involving discrete variables, and we aim to refine the theoretical framework for these discrete scenarios with fewer assumptions, leveraging recent progress in control variate estimation.

To enhance the usability of imperative learning for robot autonomy, we will extend PyPose (Wang et al. 2023; Zhan et al. 2023), an open-source Python library for robot learning, to incorporate bilevel optimization frameworks. Additionally, we will provide concrete examples of imperative learning using PyPose across various domains, thereby accelerating the advancement of robot autonomy.

Acknowledgements

This work was supported by the DARPA award HR00112490426. Any opinions, findings, conclusions, or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of DARPA.

References

Abbas M, Xiao Q, Chen L, Chen PY and Chen T (2022) Sharp-maml: Sharpness-aware model-agnostic meta learning. In: International conference on machine learning. PMLR, pp. 10–32.

Abdellah M and Abdelaziz B (2004) Dynamic feedback controller of euler angles and wind parameters estimation for a quadrotor unmanned aerial vehicle. In: 2004 IEEE International Conference on Robotics and Automation (ICRA). IEEE.

Agarwal S, Snavely N, Seitz SM and Szeliski R (2010) Bundle adjustment in the large. In: European conference on computer vision. Springer, pp. 29–42.

Agrawal A, Amos B, Barratt S, Boyd S, Diamond S and Kolter JZ (2019) Differentiable convex optimization layers. Advances in neural information processing systems 32.

Albrecht J, Fetterman A, Fogelman B, Kitanidis E, Wróblewski B, Seo N, Rosenthal M, Knutins M, Polizzi Z, Simon J and Qiu K (2022) Avalon: A benchmark for rl generalization using procedurally generated worlds. Advances in Neural Information Processing Systems 35: 12813–12825.

Algfoor ZA, Sunar MS and Kolivand H (2015) A comprehensive study on pathfinding techniques for robotics and video games. International Journal of Computer Games Technology 2015: 7–7.

Amos B, Jimenez I, Sacks J, Boots B and Kolter JZ (2018) Differentiable mpc for end-to-end planning and control. Advances in neural information processing systems 31.

Amos B and Kolter JZ (2017) Optnet: Differentiable optimization as a layer in neural networks. In: International Conference on Machine Learning. PMLR, pp. 136–145.

Anderson P, Chang A, Chaplot DS, Dosovitskiy A, Gupta S, Koltun V, Kosecka J, Malik J, Mottaghi R, Savva M et al. (2018) On evaluation of embodied navigation agents. arXiv preprint arXiv:1807.06757 .

Andreani R, Martínez JM and Svaiter BF (2010) A new sequential optimality condition for constrained optimization and algorithmic consequences. SIAM Journal on Optimization 20(6): 3533–3554.

Arbel M and Mairal J (2022a) Amortized implicit differentiation for stochastic bilevel optimization. In: International Conference on Learning Representations (ICLR).

Arbel M and Mairal J (2022b) Non-convex bilevel games with critical point selection maps. Advances in Neural Information Processing Systems 35: 8013–8026.

Aulinas J, Petillot Y, Salvi J and Lladó X (2008) The slam problem: a survey. Artificial Intelligence Research and Development : 363–371.

Baik S, Choi M, Choi J, Kim H and Lee KM (2020a) Meta-learning with adaptive hyperparameters. Advances in neural information processing systems 33: 20755–20765.

Baik S, Hong S and Lee KM (2020b) Learning to forget for meta-learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2379–2387.

Balcan MF, Khodak M and Talwalkar A (2019) Provable guarantees for gradient-based meta-learning. In: International Conference on Machine Learning (ICML). pp. 424–433.

Banino A, Barry C, Uria B, Blundell C, Lillicrap T, Mirowski P, Pritzel A, Chadwick MJ, Degris T, Modayil J et al. (2018) Vector-based navigation using grid-like representations in artificial agents. Nature 557(7705): 429–433.

Bektas T (2006) The multiple traveling salesman problem: an overview of formulations and solution procedures. omega 34(3): 209–219.

Bengio Y, Bengio S and Cloutier J (1991) Learning a synaptic learning rule. In: IEEE International Joint Conference on Neural Networks (IJCNN).

Bertazzi L, Golden B and Wang X (2015) Min–max vs. min–sum vehicle routing: A worst-case analysis. European Journal of Operational Research 240(2): 372–381.

Bertinetto L, Henriques JF, Torr P and Vedaldi A (2018) Meta-learning with differentiable closed-form solvers. In: International Conference on Learning Representations (ICLR).

Besold TR, d'Avila Garcez A, Bader S, Bowman H, Domingos P, Hitzler P, Kühnberger KU, Lamb LC, Lima PMV, de Penning L et al. (2021) Neural-symbolic learning and reasoning: A survey and interpretation 1. In: Neuro-Symbolic Artificial Intelligence: The State of the Art. IOS press, pp. 1–51.

Bhardwaj M, Choudhury S and Scherer S (2017) Learning heuristic search via imitation. In: Conference on Robot Learning. PMLR, pp. 271–280.

Billard L and Diday E (2002) Symbolic regression analysis. In: Classification, clustering, and data analysis: recent advances and applications. Springer, pp. 281–288.

Bohdal O, Yang Y and Hospedales T (2021) Evograd: Efficient gradient-based meta-learning and hyperparameter optimization. Advances in Neural Information Processing Systems 34: 22234–22246.

Borghi AM, Binkofski F, Castelfranchi C, Cimatti F, Scorolli C and Tummolini L (2017) The challenge of abstract concepts. Psychological Bulletin 143(3): 263.

Bracken J and McGill JT (1973) Mathematical programs with optimization problems in the constraints. Operations Research 21(1): 37–44.

Brockman G, Cheung V, Pettersson L, Schneider J, Schulman J, Tang J and Zaremba W (2016) Openai gym. arXiv preprint arXiv:1606.01540 .

Burri M, Nikolic J, Gohl P, Schneider T, Rehder J, Omari S, Achtelik MW and Siegwart R (2016) The euroc micro aerial vehicle datasets. The International Journal of Robotics Research 35(10): 1157–1163.

Cadena C, Carlone L, Carrillo H, Latif Y, Scaramuzza D, Neira J, Reid I and Leonard JJ (2016) Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. IEEE Transactions on robotics 32(6): 1309–1332.

Cao C, Zhu H, Yang F, Xia Y, Choset H, Oh J and Zhang J (2022) Autonomous exploration development environment and the planning algorithms. In: 2022 International Conference on Robotics and Automation (ICRA). IEEE, pp. 8921–8928.

Carpio RF, Maiolini J, Potena C, Garone E, Ulivi G and Gasparn A (2021) Mp-stsp: A multi-platform steiner traveling salesman problem formulation for precision agriculture in orchards. In: International Conference on Robotics and Automation. IEEE, pp. 2345–2351.

Chang A, Dai A, Funkhouser T, Halber M, Niebner M, Savva M, Song S, Zeng A and Zhang Y (2017) Matterport3d: Learning from rgb-d data in indoor environments. In: 2017 International Conference on 3D Vision (3DV). IEEE Computer Society, pp. 667–676.

Chen C, Chen X, Ma C, Liu Z and Liu X (2022) Gradient-based bi-level optimization for deep learning: A survey. arXiv preprint arXiv:2207.11719 .

Chen D, Zhou B, Koltun V and Krähenbühl P (2020a) Learning by cheating. In: Conference on Robot Learning. PMLR, pp. 66–75.

Chen J, Zhan LM, Wu XM and Chung Fl (2020b) Variational metric scaling for metric-based meta-learning. In: Proceedings of the AAAI conference on artificial intelligence, volume 34. pp. 3478–3485.

Chen L, Xu J and Zhang J (2023) On bilevel optimization without lower-level strong convexity. arXiv preprint arXiv:2301.00712 .

Chen T, Sun Y and Yin W (2021) A single-timescale stochastic bilevel optimization method. arXiv preprint arXiv:2102.04671 .

Chen X, Yang F and Wang C (2024) iA*: Imperative learning-based a* search for pathfinding. arXiv preprint arXiv:2403.15870 URL https://arxiv.org/abs/2403.15870.

Chi C, Feng S, Du Y, Xu Z, Cousineau E, Burchfiel B and Song S (2023) Diffusion policy: Visuomotor policy learning via action diffusion. arXiv preprint arXiv:2303.04137 .

Chitnis R, Silver T, Tenenbaum JB, Lozano-Perez T and Kaelbling LP (2022) Learning Neuro-Symbolic Relational Transition Models for Bilevel Planning. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 4166–4173.

Choe S, Mehta SV, Ahn H, Neiswanger W, Xie P, Strubell E and Xing E (2024) Making scalable meta learning practical. Advances in neural information processing systems 36.

Choudhury S, Bhardwaj M, Arora S, Kapoor A, Ranade G, Scherer S and Dey D (2018) Data-driven planning via imitation learning. The International Journal of Robotics Research 37(13-14): 1632–1672.

Co-Reyes JD, Feng S, Berseth G, Qui J and Levine S (2021) Accelerating online reinforcement learning via model-based meta-learning. In: Learning to Learn-Workshop at ICLR 2021.

Codevilla F, Müller M, López A, Koltun V and Dosovitskiy A (2018) End-to-end driving via conditional imitation learning. In: 2018 IEEE international conference on robotics and automation (ICRA). IEEE, pp. 4693–4700.

Costa PRdOd, Rhuggenaath J, Zhang Y and Akcay A (2020) Learning 2-opt heuristics for the traveling salesman problem via deep reinforcement learning. In: Asian Conference on Machine Learning. PMLR, pp. 465–480.

Cropper A and Morel R (2021) Learning Programs by Learning from Failures. Machine Learning : 801–856.

Czarnowski J, Laidlow T, Clark R and Davison AJ (2020) Deepfactors: Real-time probabilistic dense monocular slam. IEEE Robotics and Automation Letters 5(2): 721–728.

Dagréou M, Ablin P, Vaiter S and Moreau T (2022) A framework for bilevel optimization that enables stochastic and global variance reduction algorithms. arXiv preprint arXiv:2201.13409 .

(DARPA) DARPA (2019) Darpa subterranean challenge. https://www.darpa.mil/program/darpa-subterranean-challenge. Accessed: 2024-06-22.

Dechter R and Pearl J (1985) Generalized best-first search strategies and the optimality of a. Journal of the ACM (JACM) 32(3): 505–536.

Delfosse Q, Shindo H, Dhami D and Kersting K (2023) Interpretable and explainable logical policies via neurally guided symbolic abstraction. arXiv preprint arXiv:2306.01439 .

Dempe S, Kalashnikov V, Pérez-Valdés GA and Kalashnykova N (2015) Bilevel programming problems. Energy Systems. Springer, Berlin 10(978-3): 53–56.

Denevi G, Ciliberto C, Grazzi R and Pontil M (2019) Learning-to-learn stochastic gradient descent with biased regularization. arXiv preprint arXiv:1903.10399 .

Denevi G, Ciliberto C, Stamos D and Pontil M (2018a) Incremental learning-to-learn with statistical guarantees. arXiv preprint arXiv:1803.08089 .

Denevi G, Ciliberto C, Stamos D and Pontil M (2018b) Learning to learn around a common mean. In: Advances in Neural Information Processing Systems (NeurIPS). pp. 10169–10179.

Denevi G, Pontil M and Ciliberto C (2022) Conditional meta-learning of linear representations. Advances in Neural Information Processing Systems 35: 253–266.

Dijkstra E (1959) A note on two problems in connexion with graphs. Numerische Mathematik 1(1): 269–271.

Doersch C (2016) Tutorial on variational autoencoders. arXiv preprint arXiv:1606.05908 .

Domke J (2012) Generic methods for optimization-based modeling. In: Artificial Intelligence and Statistics (AISTATS). pp. 318–326.

Dong H, Mao J, Lin T, Wang C, Li L and Zhou D (2019) Neural Logic Machines. In: Proceedings of the International Conference on Learning Representations (ICLR). pp. 1–10.

Dontchev AL, Rockafellar RT and Rockafellar RT (2009) Implicit functions and solution mappings: A view from variational analysis, volume 616. Springer.

Dor M, Skinner KA, Driver T and Tsiotras P (2021) Visual slam for asteroid relative navigation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2066–2075.

Dulac-Arnold G, Levine N, Mankowitz DJ, Li J, Paduraru C, Gowal S and Hester T (2021) Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. Machine Learning 110(9): 2419–2468.

Dulac-Arnold G, Mankowitz D and Hester T (2019) Challenges of real-world reinforcement learning. arXiv preprint arXiv:1904.12901 .

Duruisseaux V, Duong TP, Leok M and Atanasov N (2023) Lie group forced variational integrator networks for learning and control of robot systems. In: Learning for Dynamics and Control Conference. PMLR, pp. 731–744.

Ebadi K, Bernreiter L, Biggie H, Catt G, Chang Y, Chatterjee A, Denniston CE, Deschênes SP, Harlow K, Khattak S, Nogueira L, Palieri M, Petráček P, Petrlík M, Reinke A, Krátký V, Zhao S, akbar Agha-mohammadi A, Alexis K, Heckman C, Khosoussi K, Kottege N, Morrell B, Hutter M, Pauling F, Pomerleau F, Saska M, Scherer S, Siegwart R, Williams JL and Carlone L (2022) Present and future of slam in extreme underground environments. arXiv preprint arXiv:2208.01787 .

Ehsani K, Gupta T, Hendrix R, Salvador J, Weihs L, Zeng KH, Singh KP, Kim Y, Han W, Herrasti A et al. (2024) Spoc: Imitating shortest paths in simulation enables effective navigation and manipulation in the real world. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 16238–16250.

Evans R and Grefenstette E (2018) Learning explanatory rules from noisy data. Journal of Artificial Intelligence Research 61: 1–64.

Finn C, Abbeel P and Levine S (2017) Model-agnostic meta-learning for fast adaptation of deep networks. In: International conference on machine learning. PMLR, pp. 1126–1135.

Finn C, Rajeswaran A, Kakade S and Levine S (2019) Online meta-learning. In: International Conference on Machine Learning (ICML). PMLR, pp. 1920–1930.

Finn C, Xu K and Levine S (2018) Probabilistic model-agnostic meta-learning. In: Advances in Neural Information Processing Systems (NeurIPS). pp. 9516–9527.

Foerster J, Farquhar G, Al-Shedivat M, Rocktäschel T, Xing E and Whiteson S (2018) DiCE: The infinitely differentiable monte carlo estimator. In: International Conference on Machine Learning (ICML). pp. 1529–1538.

Franceschi L, Donini M, Frasconi P and Pontil M (2017) Forward and reverse gradient-based hyperparameter optimization. In:

International Conference on Machine Learning (ICML). pp. 1165–1173.

Franceschi L, Frasconi P, Salzo S, Grazzi R and Pontil M (2018) Bilevel programming for hyperparameter optimization and meta-learning. In: International conference on machine learning. PMLR, pp. 1568–1577.

Fu T, Su S, Lu Y and Wang C (2024) iSLAM: Imperative SLAM. IEEE Robotics and Automation Letters (RA-L) URL https://arxiv.org/abs/2306.07894.

Furnon V and Perron L (2023) Google or-tools routing library. URL https://developers.google.com/optimization/routing/.

Gabriel H, Huang H, Waslander S and Tomlin C (2007) Quadrotor helicopter flight dynamics and control: Theory and experiment. In: 2007 AIAA Guidance, Navigation and Control Conference and Exhibit. p. 6461.

Gao H, Zhou X, Xu X, Lan Y and Xiao Y (2023) AMARL: An attention-based multiagent reinforcement learning approach to the min-max multiple traveling salesmen problem. IEEE Transactions on Neural Networks and Learning Systems .

Garcez Ad, Bader S, Bowman H, Lamb LC, de Penning L, Illuminoo B, Poon H and Zaverucha CG (2022) Neural-symbolic learning and reasoning: A survey and interpretation. Neuro-Symbolic Artificial Intelligence: The State of the Art 342(1): 327.

Garcia CE, Prett DM and Morari M (1989) Model predictive control: Theory and practice—a survey. Automatica 25(3): 335–348.

Gaztañaga E (2023) The mass of our observable universe. Monthly Notices of the Royal Astronomical Society: Letters 521(1): L59–L63.

Geffner T and Domke J (2018) Using large ensembles of control variates for variational inference. Advances in Neural Information Processing Systems 31.

Geiger A, Lenz P, Stiller C and Urtasun R (2013) Vision meets robotics: The kitti dataset. The International Journal of Robotics Research 32(11): 1231–1237.

Geng J and Langelaan JW (2020) Cooperative transport of a slung load using load-leading control. Journal of Guidance, Control, and Dynamics 43(7): 1313–1331.

Ghadimi S and Wang M (2018) Approximation methods for bilevel programming. arXiv preprint arXiv:1802.02246 .

Gharoun H, Momenifar F, Chen F and Gandomi A (2023) Meta-learning approaches for few-shot learning: A survey of recent advances. ACM Computing Surveys .

Goldblum M, Reich S, Fowl L, Ni R, Cherepanova V and Goldstein T (2020) Unraveling meta-learning: Understanding feature representations for few-shot tasks. In: International Conference on Machine Learning. PMLR, pp. 3607–3616.

Gondhi NK and Gupta A (2017) Survey on machine learning based scheduling in cloud computing. In: Proceedings of the 2017 international conference on intelligent systems, metaheuristics & swarm intelligence. pp. 57–61.

Gould S, Fernando B, Cherian A, Anderson P, Cruz RS and Guo E (2016) On differentiating parameterized argmin and argmax problems with application to bi-level optimization. arXiv preprint arXiv:1607.05447 .

Gould S, Hartley R and Campbell D (2021) Deep declarative networks. IEEE Transactions on Pattern Analysis and Machine Intelligence 44(8): 3988–4004.

Grant E, Finn C, Levine S, Darrell T and Griffiths T (2018) Recasting gradient-based meta-learning as hierarchical bayes. arXiv preprint arXiv:1801.08930 .

Grathwohl W, Choi D, Wu Y, Roeder G and Duvenaud D (2017) Backpropagation through the void: Optimizing control variates for black-box gradient estimation. arXiv preprint arXiv:1711.00123 .

Grazzi R, Franceschi L, Pontil M and Salzo S (2020) On the iteration complexity of hypergradient computation. In: Proc. International Conference on Machine Learning (ICML).

Gu S, Holly E, Lillicrap TP and Levine S (2016) Deep reinforcement learning for robotic manipulation. arXiv preprint arXiv:1610.00633 1: 1.

Guo Y, Ren Z and Wang C (2024) iMTSP: Solving min-max multiple traveling salesman problem with imperative learning. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). URL https://arxiv.org/abs/2405.00285.

Guo Z and Yang T (2021) Randomized stochastic variance-reduced methods for stochastic bilevel optimization. arXiv preprint arXiv:2105.02266 .

Halpern DF (2013) Thought and knowledge: An introduction to critical thinking. Psychology press.

Han L, Lin Y, Du G and Lian S (2019) Deepvio: Self-supervised deep learning of monocular visual inertial odometry using 3d geometric constraints. In: 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, pp. 6906–6913.

Han Z, Xu L and Gao F (2024) Learning to plan maneuverable and agile flight trajectory with optimization embedded networks. arXiv preprint arXiv:2405.07736 .

Hansen P, Jaumard B and Savard G (1992) New branch-and-bound rules for linear bilevel programming. SIAM Journal on Scientific and Statistical Computing 13(5): 1194–1217.

Hart PE, Nilsson NJ and Raphael B (1968) A formal basis for the heuristic determination of minimum cost paths. IEEE transactions on Systems Science and Cybernetics 4(2): 100–107.

He G, Jangir Y, Geng J, Mousaei M, Bai D and Scherer S (2023) Image-based visual servo control for aerial manipulation using a fully-actuated uav. In: 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, pp. 5042–5049.

Heidari N and Iosifidis A (2024) Geometric deep learning for computer-aided design: A survey. arXiv preprint arXiv:2402.17695 .

Hestenes MR and Stiefel E (1952) Methods of conjugate gradients for solving. Journal of research of the National Bureau of Standards 49(6): 409.

Ho J and Ermon S (2016) Generative adversarial imitation learning. Advances in neural information processing systems 29.

Hockley A (2011) Wayfaring: Making lines in the landscape. PhD Thesis, Brunel University.

Hocquette C, Niskanen A, Järvisalo M and Cropper A (2024) Learning MDL Logic Programs from Noisy Data. In: Proceedings of the AAAI Conference on Artificial Intelligence (AAAI). pp. 10553–10561.

Hoeller D, Wellhausen L, Farshidian F and Hutter M (2021) Learning a state representation and navigation in cluttered and

dynamic environments. IEEE Robotics and Automation Letters 6(3): 5081–5088.

Hong M, Wai H, Wang Z and Yang Z (2020) A two-timescale framework for bilevel optimization: Complexity analysis and application to actor-critic. arxiv e-prints, art. arXiv preprint arXiv:2007.05170 .

Hsu J, Mao J and Wu J (2023) Ns3d: Neuro-symbolic grounding of 3d objects and relations. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2614–2623.

Hu Y, Geng J, Wang C, Keller J and Scherer S (2023) Off-policy evaluation with online adaptation for robot exploration in challenging environments. IEEE Robotics and Automation Letters .

Hu Y, Yao Y and Lee WS (2020) A reinforcement learning approach for optimizing multiple traveling salesman problems over graphs. Knowledge-Based Systems 204: 106244.

Hu Z, Shishika D, Xiao X and Wang X (2024) Bi-cl: A reinforcement learning framework for robots coordination through bi-level optimization. arXiv preprint arXiv:2404.14649 .

Huang F and Huang H (2021) Biadam: Fast adaptive bilevel optimization methods. arXiv preprint arXiv:2106.11396 .

Huang M, Ji K, Ma S and Lai L (2022) Efficiently escaping saddle points in bilevel optimization. arXiv preprint arXiv:2202.03684 .

Hussein A, Gaber MM, Elyan E and Jayne C (2017) Imitation learning: A survey of learning methods. ACM Computing Surveys (CSUR) 50(2): 1–35.

Hutter M, Gehring C, Jud D, Lauber A, Bellicoso CD, Tsounis V, Hwangbo J, Bodie K, Fankhauser P, Bloesch M et al. (2016) Anymal-a highly mobile and dynamic quadrupedal robot. In: 2016 IEEE/RSJ international conference on intelligent robots and systems (IROS). IEEE, pp. 38–44.

Ignatiev A, Morgado A and Marques-Silva J (2018) Pysat: A python toolkit for prototyping with sat oracles. In: International Conference on Theory and Applications of Satisfiability Testing. Springer, pp. 428–437.

Ivanitskiy MI, Shah R, Spies AF, Räuker T, Valentine D, Rager C, Quirke L, Mathwin C, Corlouer G, Behn CD and Fung SW (2023) A configurable library for generating and manipulating maze datasets.

Iwańska L (1993) Logical reasoning in natural language: It is all about knowledge. Minds and Machines 3: 475–510.

Jadbabaie A, Sra S, Jegelka S, Rakhlin A and States MIOTCU (2019) Foundations of scalable nonconvex optimization .

Jerfel G, Grant E, Griffiths TL and Heller K (2018) Online gradient-based mixtures for transfer modulation in meta-learning. arXiv preprint arXiv:1812.06080 .

Ji K and Liang Y (2021) Lower bounds and accelerated algorithms for bilevel optimization. arXiv preprint arXiv:2102.03926 .

Ji K, Liu M, Liang Y and Ying L (2022a) Will bilevel optimizers benefit from loops. Advances in Neural Information Processing Systems (NeurIPS) .

Ji K, Yang J and Liang Y (2021) Bilevel optimization: Convergence analysis and enhanced design. In: International conference on machine learning. PMLR, pp. 4882–4892.

Ji T, Geng J and Driggs-Campbell K (2022b) Robust model predictive control with state estimation under set-membership uncertainty. In: 2022 IEEE Conference on Decision and Control (CDC). IEEE.

Jia C and Zhang Y (2024) Meta-learning the invariant representation for domain generalization. Machine Learning 113(4): 1661–1681.

Jiang S, Campbell D, Liu M, Gould S and Hartley R (2020) Joint unsupervised learning of optical flow and egomotion with bi-level optimization. In: 2020 international conference on 3D vision (3DV). IEEE, pp. 682–691.

Jin W, Murphey TD, Kulić D, Ezer N and Mou S (2022) Learning from sparse demonstrations. IEEE Transactions on Robotics .

Jin W, Wang Z, Yang Z and Mou S (2020) Pontryagin differentiable programming: An end-to-end learning and control framework. Advances in Neural Information Processing Systems 33: 7979–7992.

Kang S, Hwang D, Eo M, Kim T and Rhee W (2023) Meta-learning with a geometry-adaptive preconditioner. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 16080–16090.

Karniadakis GE, Kevrekidis IG, Lu L, Perdikaris P, Wang S and Yang L (2021) Physics-informed machine learning. Nature Reviews Physics 3(6): 422–440.

Kaufmann E, Loquercio A, Ranftl R, Müller M, Koltun V and Scaramuzza D (2020) Deep drone acrobatics. arXiv preprint arXiv:2006.05768 .

Kautz H (2022) The third ai summer: Aaai robert s. engelmore memorial lecture. AI Magazine 43(1): 105–125.

Khalil E, Dai H, Zhang Y, Dilkina B and Song L (2017) Learning combinatorial optimization algorithms over graphs. Advances in Neural Information Processing Systems 30.

Khanduri P, Zeng S, Hong M, Wai HT, Wang Z and Yang Z (2021) A near-optimal algorithm for stochastic bilevel optimization via double-momentum. Advances in Neural Information Processing Systems (NeurIPS) 34: 30271–30283.

Kirilenko D, Andreychuk A, Panov A and Yakovlev K (2023) Transpath: learning heuristics for grid-based pathfinding via transformers. In: Proceedings of the AAAI Conference on Artificial Intelligence, volume 37. pp. 12436–12443.

Kirillov A, Mintun E, Ravi N, Mao H, Rolland C, Gustafson L, Xiao T, Whitehead S, Berg AC, Lo WY et al. (2023) Segment anything. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 4015–4026.

Kleyko D, Rachkovskij D, Osipov E and Rahimi A (2023) A survey on hyperdimensional computing aka vector symbolic architectures, part ii: Applications, cognitive models, and challenges. ACM Computing Surveys 55(9): 1–52.

Kleyko D, Rachkovskij DA, Osipov E and Rahimi A (2022) A survey on hyperdimensional computing aka vector symbolic architectures, part i: Models and data transformations. ACM Computing Surveys 55(6): 1–40.

Koch G, Zemel R and Salakhutdinov R (2015) Siamese neural networks for one-shot image recognition. In: ICML Deep Learning Workshop, volume 2.

Kool W, Van Hoof H and Welling M (2018) Attention, learn to solve routing problems! arXiv preprint arXiv:1803.08475 .

Kouvaritakis B and Cannon M (2016) Model predictive control. Switzerland: Springer International Publishing 38.

Kwon J, Kwon D, Wright S and Nowak R (2023) A fully first-order method for stochastic bilevel optimization. arXiv preprint arXiv:2301.10945 .

Labbe M and Michaud F (2014) Online global loop closure detection for large-scale multi-session graph-based slam. In: 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, pp. 2661–2666.

Landry B (2021) Differentiable and Bilevel Optimization for Control in Robotics. Stanford University.

LeCun Y, Bengio Y and Hinton G (2015) Deep learning. nature 521(7553): 436–444.

Lee J, Bjelonic M, Reske A, Wellhausen L, Miki T and Hutter M (2024) Learning robust autonomous navigation and locomotion for wheeled-legged robots. Science Robotics 9(89): eadi9641.

Lee J, Hwangbo J, Wellhausen L, Koltun V and Hutter M (2020) Learning quadrupedal locomotion over challenging terrain. Science robotics 5(47): eabc5986.

Lee K, Maji S, Ravichandran A and Soatto S (2019) Meta-learning with differentiable convex optimization. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Leluc R, Portier F, Segers J and Zhuman A (2022) A quadrature rule combining control variates and adaptive importance sampling. Advances in Neural Information Processing Systems 35: 11842–11853.

Lewis P, Perez E, Piktus A, Petroni F, Karpukhin V, Goyal N, Küttler H, Lewis M, Yih Wt, Rocktäschel T et al. (2020) Retrieval-augmented generation for knowledge-intensive nlp tasks. Advances in Neural Information Processing Systems 33: 9459–9474.

Li B, Li Z, Du Q, Luo J, Wang W, Xie Y, Stepputtis S, Wang C, Sycara KP, Ravikumar PK, Gray A, Si X and Scherer S (2024) LogiCity: Advancing neuro-symbolic ai with abstract urban simulation. In: Advances in Neural Information Processing Systems (NeurIPS). URL https://arxiv.org/abs/2411.00773.

Li J, Gu B and Huang H (2020a) Improved bilevel model: Fast and optimal algorithm with theoretical guarantee. arXiv preprint arXiv:2009.00690 .

Li Q, Huang S, Hong Y, Chen Y, Wu YN and Zhu SC (2020b) Closed loop neural-symbolic learning via integrating neural perception, grammar parsing, and symbolic reasoning. In: International Conference on Machine Learning. pp. 5884–5894.

Li R, Wang S, Long Z and Gu D (2018) Undeepvo: Monocular visual odometry through unsupervised deep learning. In: 2018 IEEE international conference on robotics and automation (ICRA). IEEE, pp. 7286–7291.

Li Y (2017) Deep reinforcement learning: An overview. arXiv preprint arXiv:1701.07274 .

Li Z, Wang T, Lee JD and Arora S (2022a) Implicit bias of gradient descent on reparametrized models: On equivalence to mirror descent. Advances in Neural Information Processing Systems 35: 34626–34640.

Li Z, Yao Y, Chen T, Xu J, Cao C, Ma X and L\"{u} J (2022b) Softened symbol grounding for neuro-symbolic systems. In: International Conference on Learning Representations.

Liang H, Ma Y, Cao Z, Liu T, Ni F, Li Z and Hao J (2023) SplitNet: a reinforcement learning based sequence splitting method for the MinMax multiple travelling salesman problem. In: Proceedings of the AAAI Conference on Artificial Intelligence. pp. 8720–8727.

Liao R, Xiong Y, Fetaya E, Zhang L, Yoon K, Pitkow X, Urtasun R and Zemel R (2018) Reviving and improving recurrent back-propagation. In: Proc. International Conference on Machine Learning (ICML).

Lin TY, Dollár P, Girshick R, He K, Hariharan B and Belongie S (2017) Feature Pyramid Networks for Object Detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2117–2125.

Liu C, Zhu L and Belkin M (2020a) Toward a theory of optimization for over-parameterized systems of non-linear equations: the lessons of deep learning. arXiv preprint arXiv:2003.00307 7.

Liu H, Socher R and Xiong C (2019) Taming MAML: Efficient unbiased meta-reinforcement learning. In: International Conference on Machine Learning. PMLR, pp. 4061–4071.

Liu R, Gao J, Zhang J, Meng D and Lin Z (2021a) Investigating bi-level optimization for learning and vision from a unified perspective: A survey and beyond. IEEE Transactions on Pattern Analysis and Machine Intelligence 44(12): 10045–10067.

Liu R, Liu X, Yuan X, Zeng S and Zhang J (2021b) A value-function-based interior-point method for non-convex bi-level optimization. In: International Conference on Machine Learning (ICML).

Liu R, Liu Y, Zeng S and Zhang J (2021c) Towards gradient-based bilevel optimization with non-convex followers and beyond. Advances in Neural Information Processing Systems (NeurIPS) 34: 8662–8675.

Liu R, Mu P, Yuan X, Zeng S and Zhang J (2020b) A generic first-order algorithmic framework for bi-level programming beyond lower-level singleton. In: International Conference on Machine Learning (ICML).

Liu W, Daruna A, Patel M, Ramachandruni K and Chernova S (2023) A survey of semantic reasoning frameworks for robotic systems. Robotics and Autonomous Systems 159: 104294.

Liu X, Wu J and Chen S (2020c) Efficient hyperparameters optimization through model-based reinforcement learning and meta-learning. In: 2020 IEEE 22nd International Conference on High Performance Computing and Communications; IEEE 18th International Conference on Smart City; IEEE 6th International Conference on Data Science and Systems (HPCC/SmartCity/DSS). IEEE, pp. 1036–1041.

Loquercio A, Kaufmann E, Ranftl R, Müller M, Koltun V and Scaramuzza D (2021) Learning high-speed flight in the wild. Science Robotics 6(59): eabg5810.

Lu Y, Feng W, Zhu W, Xu W, Wang XE, Eckstein M and Wang WY (2023) Neuro-symbolic procedural planning with commonsense prompting. In: International Conference on Learning Representations.

Ma Y, Zhang H, Zhang Y, Gao R, Xu Z and Yang J (2019) Coordinated optimization algorithm combining GA with cluster for multi-UAVs to multi-tasks task assignment and path planning. In: International Conference on Control and Automation. IEEE, pp. 1026–1031.

Maclaurin D, Duvenaud D and Adams R (2015) Gradient-based hyperparameter optimization through reversible learning. In: International Conference on Machine Learning (ICML). pp. 2113–2122.

Manhaeve R, Dumancic S, Kimmig A, Demeester T and De Raedt L (2018) Deepproblog: Neural probabilistic logic programming. Advances in neural information processing systems 31.

Mao J, Gan C, Kohli P, Tenenbaum JB and Wu J (2018) The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. In: International Conference on Learning Representations.

Margolis GB, Yang G, Paigwar K, Chen T and Agrawal P (2024) Rapid locomotion via reinforcement learning. The International Journal of Robotics Research 43(4): 572–587.

Marquardt DW (1963) An algorithm for least-squares estimation of nonlinear parameters. Journal of the society for Industrial and Applied Mathematics 11(2): 431–441.

Martin-Linares CP, Psarellis YM, Karapetsas G, Koronaki ED and Kevrekidis IG (2023) Physics-agnostic and physics-infused machine learning for thin films flows: modelling, and predictions from small data. Journal of Fluid Mechanics 975: A41.

McAleer S, Agostinelli F, Shmakov A and Baldi P (2018) Solving the rubik's cube without human knowledge. arXiv preprint arXiv:1805.07470 .

Mellinger D and Kumar V (2011) Minimum snap trajectory generation and control for quadrotors. In: 2011 IEEE international conference on robotics and automation. IEEE, pp. 2520–2525.

Mi F, Huang M, Zhang J and Faltings B (2019) Meta-learning for low-resource natural language generation in task-oriented dialogue systems. In: International Joint Conference on Artificial Intelligence (IJCAI). AAAI Press.

Miki S, Yamamoto D and Ebara H (2018) Applying deep learning and reinforcement learning to traveling salesman problem. In: International Conference on Computing, Electronics and Communications Engineering. IEEE, pp. 65–70.

Mildenhall B, Srinivasan PP, Tancik M, Barron JT, Ramamoorthi R and Ng R (2021) Nerf: Representing scenes as neural radiance fields for view synthesis. Communications of the ACM 65(1): 99–106.

Modares H, Ranatunga I, Lewis FL and Popa DO (2015) Optimized assistive human–robot interaction using reinforcement learning. IEEE transactions on cybernetics 46(3): 655–667.

Moerland TM, Broekens J, Plaat A, Jonker CM et al. (2023) Model-based reinforcement learning: A survey. Foundations and Trends® in Machine Learning 16(1): 1–118.

Munkhdalai T and Yu H (2017) Meta networks. In: International Conference on Machine Learning (ICML).

Nazari M, Oroojlooy A, Snyder L and Takác M (2018) Reinforcement learning for solving the vehicle routing problem. Advances in Neural Information Processing Systems 31.

Nelson BL (1990) Control variate remedies. Operations Research 38(6): 974–992.

Nichol A and Schulman J (2018) Reptile: a scalable metalearning algorithm. arXiv preprint arXiv:1803.02999 .

Ouyang W, Wang Y, Han S, Jin Z and Weng P (2021) Improving generalization of deep reinforcement learning-based TSP solvers. In: Symposium Series on Computational Intelligence. IEEE, pp. 1–8.

Overton WF (2013) Competence and procedures: Constraints on the development of logical reasoning. In: Reasoning, necessity, and logic. Psychology Press, pp. 1–32.

O'Connell M, Shi G, Shi X, Azizzadenesheli K, Anandkumar A, Yue Y and Chung SJ (2022) Neural-fly enables rapid learning for agile flight in strong winds. Science Robotics 7(66): eabm6597.

Paden B, Čáp M, Yong SZ, Yershov D and Frazzoli E (2016) A survey of motion planning and control techniques for self-driving urban vehicles. IEEE Transactions on intelligent vehicles 1(1): 33–55.

Park J, Bakhtiyar S and Park J (2021) Schedulenet: Learn to solve multi-agent scheduling problems with reinforcement learning. arXiv preprint arXiv:2106.03051 .

Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, Killeen T, Lin Z, Gimelshein N, Antiga L, Desmaison A, Kopf A, Yang E, DeVito Z, Raison M, Tejani A, Chilamkurthy S, Steiner B, Fang L, Bai J and Chintala S (2019) Pytorch: An imperative style, high-performance deep learning library. Advances in neural information processing systems 32.

Pecka M and Svoboda T (2014) Safe exploration techniques for reinforcement learning–an overview. In: Modelling and Simulation for Autonomous Systems: First International Workshop, MESAS 2014, Rome, Italy, May 5-6, 2014, Revised Selected Papers 1. Springer, pp. 357–375.

Pedregosa F (2016) Hyperparameter optimization with approximate gradient. In: International Conference on Machine Learning (ICML). pp. 737–746.

Pohl I (1970) Heuristic search viewed as path finding in a graph. Artificial intelligence 1(3-4): 193–204.

Pomerleau DA (1988) Alvinn: An autonomous land vehicle in a neural network. Advances in neural information processing systems 1.

Qin T, Li P and Shen S (2018) Vins-mono: A robust and versatile monocular visual-inertial state estimator. IEEE Transactions on Robotics 34(4): 1004–1020.

Qin X, Song X and Jiang S (2023) Bi-level meta-learning for few-shot domain generalization. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 15900–15910.

Qiu Y, Wang C, Wang W, Henein M and Scherer S (2022) AirDOS: Dynamic slam benefits from articulated objects. In: International Conference on Robotics and Automation (ICRA). URL https://arxiv.org/abs/2109.09903.

Qiu Y, Wang C, Zhou X, Xia Y and Scherer S (2024) AirIMU: Learning uncertainty propagation for inertial odometry. arXiv preprint arXiv:2310.04874 URL https://arxiv.org/abs/2310.04874.

Raghu A, Raghu M, Bengio S and Vinyals O (2019) Rapid learning or feature reuse? towards understanding the effectiveness of MAML. International Conference on Learning Representations (ICLR) .

Raissi M, Perdikaris P and Karniadakis GE (2019) Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. Journal of Computational physics 378: 686–707.

Rajeswaran A, Finn C, Kakade SM and Levine S (2019) Meta-learning with implicit gradients. In: Advances in Neural Information Processing Systems (NeurIPS). pp. 113–124.

Rao C, Sun H and Liu Y (2021) Physics-informed deep learning for computational elastodynamics without labeled data. Journal of Engineering Mechanics 147(8): 04021043.

Ravi S and Larochelle H (2016) Optimization as a model for few-shot learning. In: International Conference on Learning

Representations (ICLR).

Ren Z, Rathinam S and Choset H (2023a) Cbss: A new approach for multiagent combinatorial path finding. IEEE Transactions on Robotics .

Ren Z, Rathinam S and Choset H (2023b) CBSS: A new approach for multiagent combinatorial path finding. IEEE Transactions on Robotics 39(4): 2669–2683. DOI:10.1109/TRO.2023.3266993.

Richter L, Boustati A, Nüsken N, Ruiz F and Akyildiz OD (2020) Vargrad: a low-variance gradient estimator for variational inference. Advances in Neural Information Processing Systems 33: 13481–13492.

Riegel R, Gray A, Luus F, Khan N, Makondo N, Akhalwaya IY, Qian H, Fagin R, Barahona F, Sharma U, Ikbal S, Karanam H, Neelam S, Likhyani A and Srivastava S (2020) Logical neural networks. arXiv preprint arXiv:2006.13155 .

Romero A, Song Y and Scaramuzza D (2023) Actor-critic model predictive control. arXiv preprint arXiv:2306.09852 .

Ross S, Gordon G and Bagnell D (2011) A reduction of imitation learning and structured prediction to no-regret online learning. In: Proceedings of the fourteenth international conference on artificial intelligence and statistics. JMLR Workshop and Conference Proceedings, pp. 627–635.

Rothfuss J, Lee D, Clavera I, Asfour T and Abbeel P (2019) Promp: Proximal meta-policy search. arXiv preprint arXiv:1810.06784 .

Sabach S and Shtern S (2017) A first order method for solving convex bilevel optimization problems. SIAM Journal on Optimization 27(2): 640–660.

Sadat A, Casas S, Ren M, Wu X, Dhawan P and Urtasun R (2020) Perceive, predict, and plan: Safe motion planning through interpretable semantic representations. In: European Conference on Computer Vision. Springer, pp. 414–430.

Schulman J, Wolski F, Dhariwal P, Radford A and Klimov O (2017) Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347 .

Sen P, de Carvalho BW, Riegel R and Gray A (2022) Neuro-symbolic inductive logic programming with logical neural networks. In: Proceedings of the AAAI Conference on Artificial Intelligence, volume 36. pp. 8212–8219.

Serafini L and Garcez Ad (2016) Logic tensor networks: Deep learning and logical reasoning from data and knowledge. arXiv preprint arXiv:1606.04422 .

Shaban A, Cheng CA, Hatch N and Boots B (2019) Truncated back-propagation for bilevel optimization. In: International Conference on Artificial Intelligence and Statistics (AISTATS). pp. 1723–1732.

Shah D, Osiński B, ichter b and Levine S (2023a) Lm-nav: Robotic navigation with large pre-trained models of language, vision, and action. In: Conference on Robot Learning. PMLR, pp. 492–504.

Shah D, Sridhar A, Bhorkar A, Hirose N and Levine S (2023b) Gnm: A general navigation model to drive any robot. In: 2023 IEEE International Conference on Robotics and Automation (ICRA). IEEE, pp. 7226–7233.

Shaiju A and Petersen IR (2008) Formulas for discrete time lqr, lqg, leqg and minimax lqg optimal control problems. IFAC Proceedings Volumes 41(2): 8773–8778.

Shen H and Chen T (2023) On penalty-based bilevel gradient descent method. arXiv preprint arXiv:2302.05185 .

Shi C, Lu J and Zhang G (2005) An extended kuhn–tucker approach for linear bilevel programming. Applied Mathematics and Computation 162(1): 51–63.

Shindo H, Pfanschilling V, Dhami DS and Kersting K (2023) α ilp: thinking visual scenes as differentiable logic programs. Machine Learning 112(5): 1465–1497.

Shirai Y, Jha DK, Raghunathan AU and Romeres D (2022) Robust pivoting: Exploiting frictional stability using bilevel optimization. In: 2022 International Conference on Robotics and Automation (ICRA). IEEE, pp. 992–998.

Simon C, Koniusz P, Nock R and Harandi M (2020) On modulating the gradient for meta-learning. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VIII 16. Springer, pp. 556–572.

Singh R, Bharti V, Purohit V, Kumar A, Singh AK and Singh SK (2021) Metamed: Few-shot medical image classification using gradient-based meta-learning. Pattern Recognition 120: 108111.

Sinha A, Malo P and Deb K (2017) A review on bilevel optimization: from classical to evolutionary approaches and applications. IEEE Transactions on Evolutionary Computation 22(2): 276–295.

Smith C, Karayiannidis Y, Nalpantidis L, Gratal X, Qi P, Dimarogonas DV and Kragic D (2012) Dual arm manipulation—a survey. Robotics and Autonomous systems 60(10): 1340–1353.

Snell J, Swersky K and Zemel R (2017) Prototypical networks for few-shot learning. In: Advances in Neural Information Processing Systems (NIPS).

Song Y and Scaramuzza D (2022) Policy search for model predictive control with application to agile drone flight. IEEE Transactions on Robotics 38(4): 2114–2130.

Sow D, Ji K, Guan Z and Liang Y (2022) A constrained optimization approach to bilevel optimization with multiple inner minima. arXiv preprint arXiv:2203.01123 .

Sow D, Ji K and Liang Y (2021) On the convergence theory for hessian-free bilevel algorithms. In: Advances in Neural Information Processing Systems.

Stadie B, Zhang L and Ba J (2020) Learning intrinsic rewards as a bi-level optimization problem. In: Conference on Uncertainty in Artificial Intelligence. PMLR, pp. 111–120.

Strader J, Hughes N, Chen W, Speranzon A and Carlone L (2024) Indoor and outdoor 3d scene graph generation via language-enabled spatial ontologies. IEEE Robotics and Automation Letters .

Sun L and Wang JX (2020) Physics-constrained bayesian neural network for fluid flow reconstruction with sparse and noisy data. Theoretical and Applied Mechanics Letters 10(3): 161–169.

Sundar K and Rathinam S (2013) Algorithms for routing an unmanned aerial vehicle in the presence of refueling depots. IEEE Transactions on Automation Science and Engineering 11(1): 287–294.

Suresh S, Sodhi P, Mangelson JG, Wettergreen D and Kaess M (2020) Active slam using 3d submap saliency for underwater volumetric exploration. In: 2020 IEEE International Conference on Robotics and Automation (ICRA). IEEE, pp. 3132–3138.

Sutskever I, Martens J, Dahl G and Hinton G (2013) On the importance of initialization and momentum in deep learning. In: International conference on machine learning. PMLR, pp. 1139–1147.

Sutton RS and Barto AG (2018) Reinforcement learning: An introduction. MIT press.

Tagliabue A, Kim DK, Everett M and How JP (2022) Efficient guided policy search via imitation of robust tube mpc. In: 2022 International Conference on Robotics and Automation (ICRA). IEEE, pp. 462–468.

Tang C and Tan P (2018) Ba-net: Dense bundle adjustment network. arXiv preprint arXiv:1806.04807 .

Tang H, Li Z, Peng Z and Tang J (2020) Blockmix: meta regularization and self-calibrated inference for metric-based meta-learning. In: Proceedings of the 28th ACM international conference on multimedia. pp. 610–618.

Teed Z and Deng J (2018) Deepv2d: Video to depth with differentiable structure from motion. arXiv preprint arXiv:1812.04605 .

Teed Z and Deng J (2021) DROID-SLAM: Deep Visual SLAM for Monocular, Stereo, and RGB-D Cameras. Advances in neural information processing systems .

Epson G365 (2020) Epson Electronics M-G365 Datasheet.

Thananjeyan B, Balakrishna A, Nair S, Luo M, Srinivasan K, Hwang M, Gonzalez JE, Ibarz J, Finn C and Goldberg K (2021) Recovery rl: Safe reinforcement learning with learned recovery zones. IEEE Robotics and Automation Letters 6(3): 4915–4922.

Thrun S and Pratt L (2012) Learning to learn. Springer Science & Business Media.

Triest S, Castro MG, Maheshwari P, Sivaprakasam M, Wang W and Scherer S (2023) Learning risk-aware costmaps via inverse reinforcement learning for off-road navigation. In: 2023 International Conference on Robotics and Automation (ICRA). IEEE.

Vargas Venegas C and Huang D (2023) Physics-infused reduced-order modeling of aerothermal loads for hypersonic aerothermoelastic analysis. AIAA journal 61(3): 1002–1020.

Vinyals O, Blundell C, Lillicrap T and Wierstra D (2016) Matching networks for one shot learning. In: Advances in Neural Information Processing Systems (NIPS).

Vinyals O, Fortunato M and Jaitly N (2015) Pointer networks. Advances in Neural Information Processing Systems 28.

Wang C, Gao D, Xu K, Geng J, Hu Y, Qiu Y, Li B, Yang F, Moon B, Pandey A, Aryan, Xu J, Wu T, He H, Huang D, Ren Z, Zhao S, Fu T, Reddy P, Lin X, Wang W, Shi J, Talak R, Cao K, Du Y, Wang H, Yu H, Wang S, Chen S, Kashyap A, Bandaru R, Dantu K, Wu J, Xie L, Carlone L, Hutter M and Scherer S (2023) PyPose: A library for robot learning with physics-based optimization. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). URL https://arxiv.org/abs/2209.15428.

Wang C, Wang W, Qiu Y, Hu Y, Kim S and Scherer S (2021a) Unsupervised online learning for robotic interestingness with visual memory. IEEE Transactions on Robotics (TRO) URL https://arxiv.org/abs/2111.09793.

Wang C, Yuan J and Xie L (2017a) Non-iterative slam. In: International Conference on Advanced Robotics (ICAR). IEEE, pp. 83–90. URL https://arxiv.org/abs/1701.05294.

Wang PW, Donti P, Wilder B and Kolter Z (2019) Satnet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver. In: International Conference on Machine Learning. pp. 6545–6554.

Wang R, Mao J, Hsu J, Zhao H, Wu J and Gao Y (2022) Programmatically grounded, compositionally generalizable robotic manipulation. In: International Conference on Learning Representations.

Wang S, Clark R, Wen H and Trigoni N (2017b) Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks. In: 2017 IEEE international conference on robotics and automation (ICRA). IEEE, pp. 2043–2050.

Wang W, Hu Y and Scherer S (2021b) Tartanvo: A generalizable learning-based vo. In: Conference on Robot Learning. PMLR, pp. 1761–1772.

Wei P, Hua G, Huang W, Meng F and Liu H (2021) Unsupervised monocular visual-inertial odometry network. In: Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence. pp. 2347–2354.

Wijmans E, Kadian A, Morcos A, Lee S, Essa I, Parikh D, Savva M and Batra D (2019) Dd-ppo: Learning near-perfect pointgoal navigators from 2.5 billion frames. arXiv preprint arXiv:1911.00357 .

Williams RJ (1992) Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine Learning 8: 229–256.

Wu Y, Song W, Cao Z, Zhang J and Lim A (2021) Learning improvement heuristics for solving routing problems. IEEE Transactions on Neural Networks and Learning Systems 33(9): 5057–5069.

Wurman PR, D'Andrea R and Mountz M (2008) Coordinating hundreds of cooperative, autonomous vehicles in warehouses. AI magazine 29(1): 9–9.

Xie Y, Xu Z, Kankanhalli MS, Meel KS and Soh H (2019) Embedding symbolic knowledge into deep networks. Advances in neural information processing systems 32.

Xie Y, Zhou F and Soh H (2021) Embedding symbolic temporal knowledge into deep sequential models. In: International Conference on Robotics and Automation. pp. 4267–4273.

Xin L, Song W, Cao Z and Zhang J (2021) NeuroLKH: Combining deep learning model with Lin-Kernighan-Helsgaun heuristic for solving the traveling salesman problem. Advances in Neural Information Processing Systems 34: 7472–7483.

Xu J, Zhang Z, Friedman T, Liang Y and Broeck G (2018) A semantic loss function for deep learning with symbolic knowledge. In: International Conference on Machine Learning. PMLR, pp. 5502–5511.

Xu K, Hao Y, Yuan S, Wang C and Xie L (2023) AirVO: An illumination-robust point-line visual odometry. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). URL https://arxiv.org/abs/2212.07595.

Xu K, Hao Y, Yuan S, Wang C and Xie L (2025) AirSLAM: An efficient and illumination-robust point-line visual slam system. IEEE Transactions on Robotics (TRO) URL https://arxiv.org/abs/2408.03520.

Yang F, Wang C, Cadena C and Hutter M (2023) iPlanner: Imperative path planning. In: Robotics: Science and Systems (RSS). URL https://arxiv.org/abs/2302.11434.

Yang F, Yang Z and Cohen WW (2017) Differentiable learning of logical rules for knowledge base reasoning. Advances in neural information processing systems 30.

Yang J, Ji K and Liang Y (2021) Provably faster algorithms for bilevel optimization. Advances in Neural Information Processing Systems 34: 13670–13682.

Yang M, Chen Y and Kim HS (2022a) Efficient deep visual and inertial odometry with adaptive visual modality selection. In: Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXVIII. Springer, pp. 233–250.

Yang Y, Kerce JC and Fekri F (2022b) Logicdef: An interpretable defense framework against adversarial examples via inductive scene graph reasoning. In: Proceedings of the AAAI Conference on Artificial Intelligence, volume 36. pp. 8840–8848.

Yang Y and Song L (2019) Learn to explain efficiently via neural logic inductive learning. arXiv preprint arXiv:1910.02481 .

Yang Y, Xiao P and Ji K (2024) Achieving $O(\varepsilon^{-1.5})$ complexity in Hessian/Jacobian-free stochastic bilevel optimization. Advances in Neural Information Processing Systems 36.

Yang Z, Ishay A and Lee J (2020) Neurasp: Embracing neural networks into answer set programming. In: Bessiere C (ed.) Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20. International Joint Conferences on Artificial Intelligence Organization, pp. 1755–1762. DOI:10.24963/ijcai.2020/243.

Yao W, Yin H, Zeng S and Zhang J (2024a) Overcoming lower-level constraints in bilevel optimization: A novel approach with regularized gap functions. arXiv preprint arXiv:2406.01992 .

Yao W, Yu C, Zeng S and Zhang J (2023) Constrained bi-level optimization: Proximal lagrangian value function approach and hessian-free algorithm. In: The Twelfth International Conference on Learning Representations.

Yao W, Yu C, Zeng S and Zhang J (2024b) Constrained bi-level optimization: Proximal lagrangian value function approach and hessian-free algorithm. arXiv preprint arXiv:2401.16164 .

Ye J, Batra D, Das A and Wijmans E (2021) Auxiliary tasks and exploration enable objectgoal navigation. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 16117–16126.

Ye M, Liu B, Wright S, Stone P and Liu Q (2022) Bome! bilevel optimization made easy: A simple first-order approach. In: Conference on Neural Information Processing Systems (NeurIPS). p. 1.

Yokoyama N, Ha S and Batra D (2021) Success weighted by completion time: A dynamics-aware evaluation criteria for embodied navigation. In: 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, pp. 1562–1569.

Yonetani R, Taniai T, Barekatain M, Nishimura M and Kanezaki A (2021a) Path planning using neural a* search. In: International conference on machine learning. PMLR, pp. 12029–12039.

Yonetani R, Taniai T, Barekatain M, Nishimura M and Kanezaki A (2021b) Path planning using neural a* search. In: International conference on machine learning. PMLR, pp. 12029–12039.

Zhan Z, Gao D, Lin YJ, Xia Y and Wang C (2024) iMatching: Imperative correspondence learning. In: European Conference on Computer Vision (ECCV). URL https://arxiv.org/abs/2312.02141.

Zhan Z, Li X, Li Q, He H, Pandey A, Xiao H, Xu Y, Chen X, Xu K, Cao K, Zhao Z, Wang Z, Xu H, Fang Z, Chen Y, Wang W, Fang X, Du Y, Wu T, Lin X, Qiu Y, Yang F, Shi J, Su S, Lu Y, Fu T, Dantu K, Wu J, Xie L, Hutter M, Carlone L, Scherer S, Huang D, Hu Y, Geng J and Wang C (2023) PyPose v0.6: The imperative programming interface for robotics. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) Workshop. URL https://arxiv.org/abs/2309.13035.

Zhang B, Luo C, Yu D, Li X, Lin H, Ye Y and Zhang B (2024) Metadiff: Meta-learning with conditional diffusion for few-shot learning. In: Proceedings of the AAAI Conference on Artificial Intelligence, volume 38. pp. 16687–16695.

Zhang H, Chen W, Huang Z, Li M, Yang Y, Zhang W and Wang J (2020a) Bi-level actor-critic for multi-agent coordination. In: Proceedings of the AAAI Conference on Artificial Intelligence, volume 34. pp. 7325–7332.

Zhang J, Hu C, Chadha RG and Singh S (2020b) Falco: Fast likelihood-based collision avoidance with extension to human-guided navigation. Journal of Field Robotics 37(8): 1300–1313.

Zhao TZ, Kumar V, Levine S and Finn C (2023) Learning fine-grained bimanual manipulation with low-cost hardware. arXiv preprint arXiv:2304.13705 .

Zhao Z, Li B, Du Y, Fu T and Wang C (2024) PhysORD: A neuro-symbolic approach for physics-infused motion prediction in off-road driving. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). URL https://arxiv.org/abs/2404.01596.

Zhou F, Wu B and Li Z (2018) Deep meta-learning: Learning to learn in the concept space. arXiv preprint arXiv:1802.03596 .

Zhou P, Yuan X, Xu H, Yan S and Feng J (2019) Efficient meta learning via minibatch proximal update. In: Advances in Neural Information Processing Systems (NeurIPS). pp. 1532–1542.

Zhou P, Zou Y, Yuan XT, Feng J, Xiong C and Hoi S (2021) Task similarity aware meta learning: Theory-inspired improvement on maml. In: Uncertainty in artificial intelligence. PMLR, pp. 23–33.

Zhu K and Zhang T (2021) Deep reinforcement learning based mobile robot navigation: A review. Tsinghua Science and Technology 26(5): 674–691.