Towards Safe and Sample-efficient Learning for Autonomous Energy Systems

Submitted in partial fulfillment of the requirements for

the degree of

Doctor of Philosophy

in

Advanced Infrastructure Systems

Bingqing Chen

BEng, Civil Engineering, The University of Hong Kong
MSc, Structural Engineering, The University of Hong Kong
M.S., Machine Learning, Carnegie Mellon University

Carnegie Mellon University
Pittsburgh, PA

May, 2022

Keywords: Reinforcement Learning, Building Control, Distributed Energy Resources, Demand Flexibility, Inverter Control, Safety Verification, Constrained Reinforcement Learning, Autonomous Racing

Acknowledgments

First and foremost, I would like to thank my advisor Mario Bergés, and committee members Sean Qian, Soummya Kar, and Scott Moura for their feedback and suggestions.

Secondly, I would like to thank my significant other, Jonathan Francis, for being a constant throughout my PhD, and my long-time friend, Zicheng Cai, for helping me get on my feet with my first paper and continuous support for more than a decade. I would also like to thank my friends and collaborators, Weiran Yao, Jingxiao Liu, Yujie Wei, Priya Donti, Henning Lange, Xuecheng Liu, and research mentors, Sylvia Herbert, Kyri Baker, Luca Bondi, Samarjit Das, Ming Jin, Zhe Wang, Marco Pritoni, Tianzhen Hong.

Last but not the least, I would like to thank Zhiang Zhang for sharing his model on Intelligent Workspace, and `ecobee` and its customers for their data. I appreciate Michael Frenak's help in facilitating the experiments on Carnegie Mellon Campus. I am also grateful to Brandon Amos, Eric Burger, and Jaime Fisac, whose prior works are major source of inspiration for this thesis.

The research is funded by the U.S. Department of Energy's Office of Energy Efficiency and Renewable Energy (EERE) under the Building Technologies Office Award Number DE-EE0007682 and Carnegie Mellon University's College of Engineering Moonshot Award for Autonomous Technologies for Livability and Sustainability (ATLAS).

Abstract

Given the dire consequences of climate change, there are growing incentives to curb carbon emissions by reducing energy consumption and increasing the penetration of renewable energy generation, along with other measures to jointly combat this global challenge. In this thesis, we focus on learning-based controls to 1) reduce the energy consumption in buildings, and 2) to facilitate the integration of distributed energy resources (DERs).

Recently, there is increasing interest in applying reinforcement learning (RL) for energy systems operation given that 1) high-fidelity models for these system are resource-intensive to develop and not commonly available, 2) energy systems are heterogeneous and the solution for one system may not be transferable to others, and 3) some of these systems are undergoing transitions and thus the control should be adaptive and future-proof.

While RL is a promising solution, real-world applications of RL agents are numbered due to the facts that 1) RL agents generally take a long time to learn and reach acceptable performance and that 2) the actions by RL agents may not satisfy safety constraints posed by the underlying physical systems or the functional requirements. Thus, RL agents should learn safely and sample-efficiently to be practical for real-world energy systems.

Firstly, we address the challenge of sample complexity in *Research Question 1*, grounded in the application of improving energy efficiency in building operations. We expedite the learning process by warm-starting the RL agent with expert demonstrations and by incorporating domain knowledge on building thermodynamics in its policy. We validate that our proposed agent, Gnu-RL, can be deployed on real-world testbeds with satisfactory initial performance, and improve energy efficiency over time. In a notable experiment, Gnu-RL was deployed to operate a real-world testbed for three weeks, wherein it saved 16.7% of cooling demand compared to the existing controller while maintaining better thermal comfort. In comparison to existing methods, Gnu-RL is both practical and scalable as it only requires historical data and minimal engineering to be applied to other buildings.

Secondly, we focus on the application of facilitating the integration of DERs from both the demand side and the supply side. In *Research Question 2*, we utilize the inherent flexibility in a class of building loads — thermostatically controlled loads (TCLs), which accounts for 20% of the electricity consumption in the United States — to provide grid services. By characterizing the set of admissible action sequences (i.e. feasible for the TCLs and satisfying the end-use requirements) we propose a distributed control solution, COHORT, to coordinate a population of heterogeneous TCLs to jointly provide grid services. We demonstrate that COHORT is applicable to use cases including, but not limited to, generation following, minimizing ramping, and peak load curtailment. Aside from simulation studies, we validated COHORT in a hardware-in-the-loop simulation, including a real-world testbed and simulated instances of TCLs. During the 15-day experimental period, COHORT reduced daily peak loads by an average of 12.5% and maintained comfortable temperatures. COHORT is computationally scalable to both

large population sizes and long planning horizons, which unlock the potential to shift TCLs over extended periods of time, e.g. shifting wind and solar power from times when it might otherwise be curtailed to times it may be needed over the course of a day.

In *Research Question 3*, we extend *Research Question 2* to incorporate network constraints on top of device-level constraints. Specifically, we focus on controlling inverters, through which DERs are connected to the distribution networks, to ensure voltage constraints are not violated, as over-voltage has already become a common occurrence in areas with high renewable penetration. On the IEEE 37-bus feeder system, our proposed approach, PROF satisfies the voltage constraints 100% of the time, compared to 22% over-voltage violations incurred by a Volt/Var control strategy. Voltage support from inverters increase the hosting capacity of the existing networks and reduce the curtailment of renewable generation. Furthermore, as the renewable energy resources gradually replace fossil fuel ones over the course of coming decades, a learning-based control strategy can adapt to the transitioning power grid.

Finally, we consider the problem of power system operation in the abstracted form of high-dimensional, non-linear systems. To enforce safety constraints in performance-driven learning in the general case of high-dimensional, non-linear systems, we propose SAGE by incorporating Hamilton-Jacobi (HJ) reachability theory, a safety verification method for non-linear systems, into the constrained Markov decision process (CMDP) framework. Though HJ reachability is traditionally not scalable to high-dimensional systems, we demonstrate that with neural approximation, the HJ safety value can be learned directly on vision context—the highest-dimensional problem studied via the method, to-date. We evaluate our method on several benchmark tasks, including `Safety Gym` and `Learn-to-Race` (`L2R`), a recently-released high-fidelity autonomous racing environment. Our approach has significantly fewer constraint violations in comparison to other constrained RL baselines in `Safety Gym`, and achieves the new state-of-the-art results on the `L2R` benchmark task.

Contents

List of Tables

List of Figures

1. Introduction

There is increasing interest in applying learning-based control for energy systems operation, as 1) high-fidelity models for these system are resource-intensive to develop and not commonly available, 2) energy systems are heterogeneous and the solution for one system may not be transferable to others, and 3) some of these systems are undergoing transitions and thus the control should be adaptive and future-proof.

While reinforcement learning (RL) is a promising solution, real-world applications of RL agents are numbered due to the facts that 1) RL agents generally take a long time to learn and reach acceptable performance, and 2) that the actions during the learning phase may not satisfy functional requirements or be feasible for the underlying physical system. Thus, RL agents should learn safely and sample-efficiently to be practical for real-world energy systems.

In *Research Question 1*, we work towards the vision of RL agents learning to conserve energy in buildings, while maintaining a satisfactory comfort level. In *Research Question 2*, we extend the idea to a populations of buildings coordinating with each others to jointly provide grid services, while satisfying building-level constraints. In *Research Question 3*, the RL agent learns to control distributed energy resources (DERs) safely with consideration of both local and grid-level constraints. We also consider the abstracted problem of how to learn a safe set for high-dimensional, non-linear systems.

1.1. Industry Problems

Given the dire consequences of climate change, there is growing incentive to curb carbon emissions by 1) reducing energy consumption, and 2) increasing the penetration of renewable energy [168, 60, 179], along with other approaches to jointly combat this global challenge. In this thesis, we focus on improving upon the existing control strategies 1) to reduce the energy consumption in buildings, and 2) to facilitate the integration of variable renewable energy resources.

1.1.1. Energy-efficient Building Operation

Buildings are good candidates for energy-saving strategies due to the sheer size of their energy consumption. Concretely, buildings account for about 40% of the total energy and 70% of the electricity consumption in the United States [141, 91], and it is estimated that up to 30% of that energy usage may be reduced through advanced sensing and control strategies [92, 78]. Furthermore, replacing existing control logic with more intelligent ones requires almost no retrofit and would be a cost-effective solution.

However, this potential is largely untapped into. The majority of buildings systems today are still operated by rule-based and feedback controls, such as hysteresis or proportional-integral-derivative (PID) control [38], which depends on engineers to manually specify the control parameters and occupancy schedules. These control parameters and occupancy schedules are generally conservative, and as a result the building environments are often conditioned unnecessarily. Furthermore, these prescriptive and reactive

control strategies do not take into consideration predictive information on disturbances, such as weather and occupancy, leading to sub-optimal energy performance [126].

Case Study: To examine the energy performance of existing control, we conducted a case study in four rooms on Carnegie Mellon University (CMU) campus, during the fall semester (September 1 - December 10) in 2019. The testbed includes one classroom, one student lounge, and two conference rooms, ranging from 215.3 - 1025.5 ft^2. By adopting a reactive rule-based control (RBC), we decreased total heating and cooling demand by 14.2% in comparison to the weather normalized performance of the existing control over the same time period in 2018, while maintaining a similar level of thermal comfort (Figure 1.1). The reactive RBC simply replaced the design occupancy with the actual occupancy count from sensor measurements [163], and only used it to adjust the minimal airflow rate per the guidelines on demand-controlled ventilation in ASHRAE 62.1 [14].

The significant energy savings from refining a single setpoint showcase the conservativeness of existing control and reaffirm the large energy-saving potential in buildings. For instance, the existing control assume that each room operates under a fixed schedule from 6:00-22:00 regardless of weekday or weekend, which corresponds to 66.6% utilization (the dashed black line in Figure 1.1). In comparison to the actual utilization, the existing control unnecessarily conditions these rooms around 10 to 20% of the time.

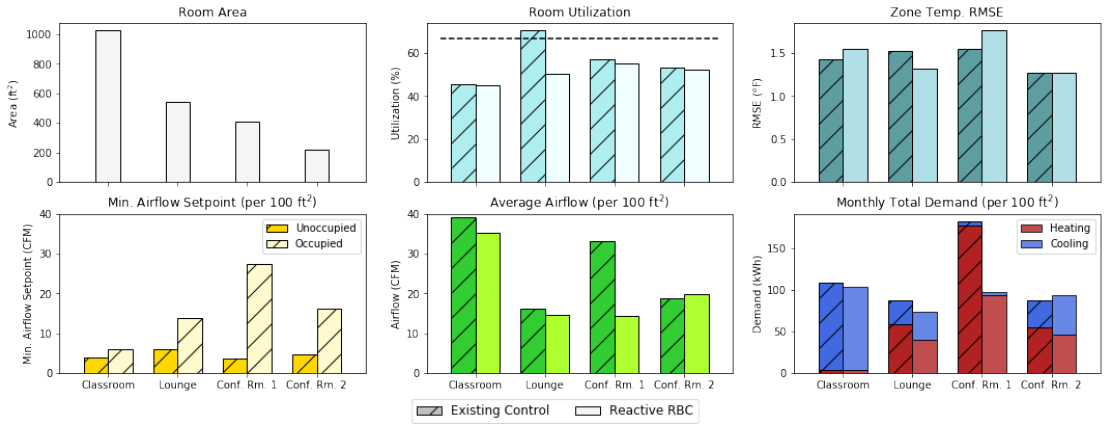


Figure 1.1: Performance of reactive RBC in comparison to the existing control. Thermal comfort is estimated by the deviation of actual zone temperature from its setpoint. By determining airflow rate based on actual occupancy count instead of a fixed schedule, there is on average 14.2% savings in total heating and cooling demand.

1.1.2. Renewable Energy Integration

There is growing incentive to replace traditional fossil fuel-based generation with renewable ones, e.g. solar, wind, hydro, and geothermal. For instance, the maximum hourly

penetration of utility-scale wind and solar generation reached 62.6% in California Independent System Operator (CAISO) in 2018 [198]. In contrast to hydro and geothermal energy resources, solar and wind ones are not constrained by local conditions, such as surface runoff and geology, but are uncertain and variable [135]. These variable renewable energy resources introduce unintended challenges for grid operators.

- The intermittent and variable nature of renewable generation makes it difficult to balance supply and demand of energy in the power grid. Traditionally, demand-side load is viewed as uncontrollable, while supply-side resources manage power generation to match it [165]. Such paradigm is no longer sufficient to accommodate an increasing penetration of renewable resources [118, 1], which is not only non-dispatchable, but also of uncertain and variable nature. This calls for new ways to match supply and demand over multiple timescales [135], i.e. from sub-second to inter-daily.

- High penetration of renewable generation without proper control leads to constraint violations in the distribution network. For instance, over-voltage has already become a common occurrence in areas with high renewable penetration [109, 196]. Thus, it is of growing importance to account for network constraints in distribution networks with high penetration of distributed energy resources[1] (DER) [98]. To alleviate voltage violations, the IEEE 1547.8-2018 standard [22] recommends a Volt/Var control strategy, in which the reactive power contribution of an inverter[2] is adjusted based on local voltage measurements. Unfortunately, such network-agnostic control based on local information only reduces, but does not avoid over-voltage and other violations.

- A related issue is that existing control in face of constraint violations lead to curtailment of available renewable generation. The original IEEE 1547-2003 standard specifies that inverters simply trip and disconnect from the grid when the voltage exceed limits, even when there is still behind-the-meter load [109]. The updated IEEE 1547-2018 takes a reactive power priority approach, meaning that the inverter will prioritize providing voltage support and curtail active power as needed.

  A case study of more than 1300 solar panel installations showed that on the worst day 5% of the customers experienced the most severe impact, with a generation loss of at least 16% [196]. Furthermore, the curtailment is distributed unevenly; customers far away from the distribution feeders suffered the most loss. In 2018, 461,043 MWh of renewable generation was curtailed in CAISO [1], sufficient to power a small country. A projection of the future power systems in the United

[1] Distributed energy resources refer to energy resources connected to distribution networks, including both generation units such as micro-turbines, photo-voltaic panels, and energy storage such as batteries [9].

[2] An inverter is a power electronics device that connects a DER with the distribution network. It convert native DC electricity into grid-compatible AC power, by controlling switching semiconductor devices at a fast timescale [135].

States with high renewable energy penetrations [151] show massive curtailment in off-peak load conditions.

- An inverter-dominated grid has low-inertia, and as a result reduced stability [135, 154]. Inverters do not have the inherent inertia from mechanical rotors in the case of synchronous machines [135, 154]. As a result, distribution network with high renewable penetration requires active control at much faster timescales [154].

1.2. Research Vision

In light of these industry challenges, this thesis presents contributions to achieve the following research vision.

Smart buildings learning to conserve energy, while maintaining occupants' comfort at a satisfactory level. It was envisioned in [161, 159, 160] more than 20 years ago that smart homes of the future have household appliances endowed with microprocessors, and communicates with each other to satisfy occupants' needs. The benefits of realizing this vision include improvement in energy efficiency and occupants' comfort [141, 24, 92]. Given the reduced cost of hardware and communication infrastructure, this vision seems extremely feasible today.

However, it was also pointed out that the bottleneck of this vision lies in the programming effort required [160]. In the context of a smart home, the controller needs to incorporate models of the thermodynamics and the mechanical system, take into consideration occupants' behaviour and preferences, and finally balance multiple competing objectives. Furthermore, building environments are heterogeneous, e.g. they have different layouts, system configurations, and occupancy patterns. Thus, the solution for one building is not directly transferable to another. Given the daunting task of implementing these considerations manually for individual buildings, it is asserted in [160] that *an intelligent environment must be adaptive.*

Research question 1 (Section 1.4.1) works towards this vision, by developing a practical and scalable reinforcement learning solution for building control—Gnu-RL [38, 42], to be described in Section 3. The proposed solution is practical, as it is sample efficient and can be deployed to real-world testbeds, using no prior information other than historical data. Since the proposed solution learns from and adapts to the environment autonomously, it is scalable to the heterogeneous building stock.

Building loads coordinating with each other to provide demand flexibility, while satisfying device-level constraints. To address the challenge of matching the supply and demand of energy, given the non-dispatchable and variable nature of renewable generation, a promising solution is to tap into the flexibility of demand-side resources to reduce, shift, or modulate their loads in response to price or control signals [165]. Such demand flexibility can be utilized to provide grid services[3], improve grid resiliency [165],

[3]Grid services refer to the services that support the delivery of electricity from the utility to the consumer and provide value through avoided electricity system costs [165].

and reduce operating costs [35].

Buildings are, again, ideal candidates due to their aggregate capacity. It is estimated that a fourth of all electricity demand in the United States will be dispatchable if buildings respond to dispatch signals [141]. Specifically, we focus on a class of flexible building loads—residential thermostatically controlled loads (TCLs), such as air conditioners (ACs), refrigerators, and electric water heaters, which account for about 20% of the electricity consumption in the United States [103]. They derive their flexibility through thermal inertia, and thus can provide grid services without compromising their end uses. Furthermore, 11% of the households in the United States are already equipped with smart thermostats [127], which provides a readily-available control and communication infrastructure.

Research question 2 (Section 1.4.2) improves upon existing methods, such that the proposed solution—COHORT [42] (Section 4) is scalable to long planning horizons, e.g. shifting wind and solar power from times when it might otherwise be curtailed to times it may be needed over the course of a day. This improvement unlocks the potential for TCLs to provide grid services that utilize buildings' thermal mass to shift loads over extended period of time, which is identified as the most promising use case for these loads by [92, 135, 186].

Inverters learning to control safely, i.e. satisfying both grid-level and device-level constraints. There are more than 2.3 million solar generators connected to the distribution networks in the United State today [152]. The growing penetration of grid-connected photo-voltaics (PVs), along with other DERs, calls for improved control strategies to integrate these DERs and to facilitate a seamless transition into the future power systems with high renewable penetration [135].

Recall that DERs, such as solar panels, micro-turbines, and batteries, are connected to the distribution networks via inverters. While inverters present challenges as they have low-inertia and behave differently from traditional synchronous machines, the silver lining is that they allow for fast actuation and flexible control. In fact, the control policy, instead of the inverter's physical properties, dictates the inverter's dynamics and interaction with the grid [135].

For *Research question 3* (Section 1.4.3), we develop a strategy, PROF (Section 5), to flexibly embed convex operational constraints into neural policies, such that the inverters learn to control safely, i.e. satisfying both inverter-level and grid-level constraints. Embedding safety guarantees in a learning-based solution instills confidence in industry practitioners.

A limitation of PROF is that it assumes the existence of a nominal model, with which to construct the safe set. Furthermore, the nominal model is not updated through interactions with the environment. For *Research question 3* (Section 1.4.3), we also consider the abstracted problem of learning about safety for a high-dimensional, nonlinear systems, and validate it in a challenging safety-critical task of autonomous racing (Section 6).

Being able to learning about safety through interactions with the environment enables

the inverters to adapt to the transitioning power grid autonomously, as renewable generation gradually replaces fossil fuel-based one over the course of coming decades, At the same time, DERs are installed by customers spontaneously and thus are opaque to distribution system operators (DSO) [196, 109].

1.3. Scope and Assumptions

In *Research Question 1* (Section 1.4.1), we focus on the energy efficiency of heating, ventilation, and air conditioning (HVAC) systems, which accounts for 30% of the energy consumed in buildings [141]. We envision the proposed solution (Section 3) to be transferable to other building systems. We assume operational data, particularly the state-action pairs, are logged in building automation system (BAS) for existing buildings.

In *Research Question 2* (Section 1.4.2), we focus on utilizing the demand flexibility in residential TCLs to provide grid services. This research question is network-agnostic, i.e. we do not consider the impact of TCL coordination on network constraints [185]. We adopt a direct load control-based approach, as opposed to methods based on economic incentives[4]. We expect the proposed solution (Section 4) to be applicable to other demand-side resources, such as batteries and thermal storage. We assume there exists the control and communication infrastructure (e.g. networked thermostats) for a load aggregator to have two-way communication with and be able to control a population of TCLs.

In *Research Question 3* (Section 1.4.3), we focus on operating inverters that connects DERs to the distribution networks, while satisfying voltage constraints. Since PV panels are the most prevalent DERs, we focus primarily on grid-connected PVs. We assume each inverter has the capabilities to communicate its states and take setpoints per IEEE 1547.1-2020 [22] guidelines for new installations.

1.4. Research Questions

We formulate the research questions that will be addressed in this thesis.

1.4.1. Research Question 1

What is a practical and scalable reinforcement learning solution that can be deployed in real-world building HVAC systems with satisfactory initial performance, without the need for high-fidelity simulation models? This research question aims to develop a practical and scalable solution for operating building HVAC systems by combining the strength of MPC and RL. MPC and RL were identified as options for optimal control of building systems in the seminal work by Michael C. Mozer [159] more than 20 years ago and are the two most popular approaches for HVAC control in literature. However, the majority of building stocks are still operated by on-off or PID control, despite the large energy saving potentials and successful demonstrations of advanced control strategies

[4]We refer interested readers to [35] for a comparison of direct load control vs. price-based methods.

reported in the literature. As discussed in Section 3.2, both approaches have drawbacks that limit their wide-spread adoption.

Challenge: MPC is not scalable due to the heterogeneity of building environments. MPC optimize an objective function iteratively over a receding time horizon. Despite many successful applications of MPC (e.g., [178, 149]), its wide-spread adoption has been limited by the need of accurate models [177, 126]. This is especially challenging as buildings are heterogeneous [147], e.g. they have different layouts, HVAC configurations, and occupancy patterns. Thus, custom models are required for each thermal zone or building. By some estimates, modeling can account for up to 75% of the time and resources required for implementing MPC in practice [182]. Another drawback is that MPC treats modeling and planning as two separate tasks. The quality of the model is evaluated by criteria such as prediction error, which does not necessarily lead to good control performance.

Challenge: RL is not practical for real-world building systems due to its sample complexity. RL, on the other hand, is a learning-based approach that adapts to different environments by learning a control policy through direct interaction with the environment [199]. RL has achieved remarkable success in mastering many highly complex tasks such as playing Go [158] and StarCraft [207]. It has also been shown to be a feasible approach for optimal control of HVAC systems [61, 145]. However, the flexibility of the RL framework comes at the cost of increased sample complexity [121]. Taking recent published results as examples, an RL agent may need 5 million interaction steps (47.5 years in simulation) to achieve the same performance as a feedback controller on an HVAC system [226]. Even after pre-training an RL agent on expert demonstrations, it may still require an additional year of training in simulation to achieve similar performance to a baseline controller [117].

While these results show that RL agents can be trained successfully in simulation, high-fidelity models are generally not available for individual thermal zones or buildings. Thus, training RL agents in simulation is not scalable and shifts the focus back to modeling (and calibration) [219].

1.4.2. Research Question 2

How to coordinate a large, heterogeneous population of TCLs to provide a wide range of grid services in a computationally-scalable manner, while incorporating detailed, system-specific models and constraints from individual loads? This research question aims to develop a practical, scalable, and versatile solution for coordinating a heterogeneous population of TCLs to provide grid services. There are two core difficulties with TCL coordination: large state-action space and non-convex constraints posed by individual loads. Building loads must be aggregated across a population to be a meaningful resource at the grid-level [165], which results in a problem with a large

state action space[5]. The set of all admissible power profiles of a TCL, defined here as flexibility [230], is combinatorial and non-convex, the size of which grows exponentially with the planning horizon[6].

As discussed in Section 4.2, none of the existing approach is computationally-scalable to both large population sizes and long planning horizons. Furthermore, we want to validate the proposed solution is practical for real-world systems, motivated by the observation from literature that there exist large discrepancy between performance reported in simulation and in real-world testbeds.

Challenge: No existing approach is computationally-viable for long planning horizons, when models and constraints for individual TCLs are considered. Centralized, distributed, and decentralized approaches have been proposed for TCL coordination (as summarized in Section 4.2). A distributed architecture, where each system is responsible for its own control and coordinates with others to jointly achieve a grid-level objective, can address many of the limitations of centralized and decentralized approaches. Most similar to our work, a distributed control solution is proposed in [31], which incorporates detailed models and constraints for individual loads and is, at the same time, scalable to large population sizes. However, it does not address the difficulty that the number of admissible action sequences grows exponentially with the planning horizon.

From an application perspective, the greatest potential exists in grid services that use the building's thermal mass to shift loads [186, 91]. However, load shifting typically entails a long planning horizon, e.g. day-ahead scheduling. Addressing this challenge unlocks the potential for a most promising use case.

Challenge: It is unclear how well simulation-based validation reflect performance on real-world systems. Given the practical difficulty of simultaneously controlling a large number of testbeds, the majority of works on this topic validated their approaches in simulation. The large discrepancy between performance reported in simulation and in real-world testbeds [162] motivates us to evaluate our proposed method realistically.

1.4.3. Research Question 3

There is growing interest in enforcing some notion of safety in RL algorithms, e.g., satisfying safety constraints, avoiding worst-case outcomes, or being robust to environmental stochasticity [87]. We focus on the notion of safety as satisfying constraints. Enforcing safety constraints in performance-driven learning boils down to two sub-problems [63, 106]: 1) safely optimizing a performance controller given a safety set, and 2) learning the safety set by interacting with the environment (safe exploration).

[5]To provide a concrete example, Pennsylvania-New Jersey-Maryland (PJM) Interconnection requires at least 0.1MW of capacity for an entity to participate in the auxiliary service market. Assuming that an AC consumes 1kW and can shift 10% of its power consumption without violating comfort, a load aggregator has to coordinate at least 1000 ACs to be eligible for the auxiliary service market.

[6]A TCL operates in discrete action space, i.e. *on* or *off*. To satisfy constraints on the state, i.e. the temperature, the actions need to be coupled over time through the thermodynamics [230].

Research Question 3.1: How to safely learn a controller for inverters, given a convex safety set characterizing the grid-level and inverter-level constraints? This sub-question assumes that a safety set is given a priori, and focuses on learning a performance controller given the safety set. To simplify the problem, the safety set is constructed via a linearized grid model with bounded linearization error [28]. The resulting safety set is convex and a conservative under-approximation of the true safety set.

Research Question 3.2: How to optimize a policy subject to safety constraints in a high-dimensional, non-linear system? While the linearized grid model simplifies the specification of the safety set, the constructed safety set is conservative and curtails renewable energy unnecessarily in comparison to the AC-OPF solution. We abstract the problem and raise the general question on how to construct the safety set for a general high-dimensional and non-linear system.

1.5. Organization

The rest of the thesis is organized as follows. Section 6.3 presents shared background knowledge. Section 3 summarizes the work that addresses *Research Question 1*. Section 4 presents the proposed approach to tackle *Research Question 2*. Section 5 and Section 6 discuss the solutions to *Research Question 3.1* and *Research Question 3.2*. Finally, Section 7 recaps the contributions and discusses directions for future work.

2. Preliminaries

This section presents shared background knowledge. Other background technical concepts will be described in individual sections.

Reinforcement Learning

RL learns an optimal control policy through direct interaction with the environment. The optimal control problem may be formulated as a Markov Decision Process[7] (MDP), as shown in Figure 2.1.

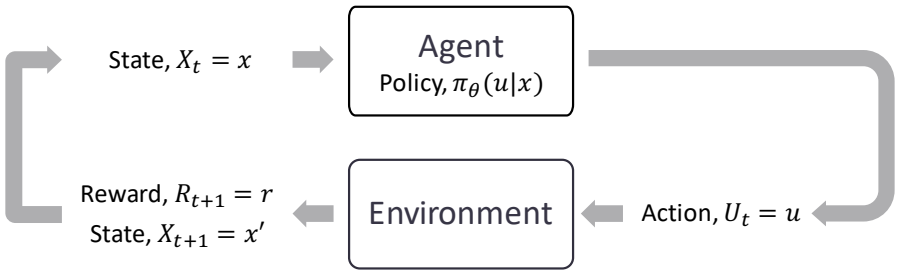


Figure 2.1: The agent-environment interaction in a Markov Decision Process

At each time step t, the agent selects an action $U_t = u \in \mathcal{U}$ given the current state $X_t = x \in \mathcal{X}$ based on its policy $\pi_\theta : \mathcal{X} \to \mathcal{P}(\mathcal{U})$ (Eq. 2.1a), where $\mathcal{X}$ is the state space, $\mathcal{U}$ is the action space. In modern RL, the policy is commonly approximated by a neural network parameterized by θ [156]. Using a neural network as a function approximator allows one to handle large state-space and generalize from past experiences [199]. When the agent takes the action u, there is a state transition $X_{t+1} = x'$ based the environment dynamics $f : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$ (Eq. 2.1b) and the agent receives a reward $R_{t+1} = r$ based on a reward function $R : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$. Unless stated otherwise, the cost is the negative of the reward (Eq. 2.1c).

$$u \sim \pi_\theta(u|x) = \mathbb{P}(U_t|X_t = x;\ \theta) \tag{2.1a}$$
$$x' \sim f(x,u) = \mathbb{P}(X_{t+1}|X_t = x, U_t = u) \tag{2.1b}$$
$$r = R(x,u) = -C(x,u) \tag{2.1c}$$

The objective of RL is to find a optimal policy π_{θ^*} that maximizes the expected total reward or, equivalently, minimizes the expected total cost.

$$\theta^* = \arg\max_\theta \mathbb{E}_{\pi_\theta}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k}\right] = \arg\min_\theta \mathbb{E}_{\pi_\theta}\left[\sum_{l=0}^{\infty} \gamma^l c_{k+l}\right] \tag{2.2}$$

[7] While Richard Bellman's notation is more commonly seen in RL literature (i.e. s-states, a-actions, r-rewards), we adopt Lev Pontryagin's notation (i.e. x-states, u-actions, c-costs) to be consistent with classical control literature.

RL methods can be categorized as model-free and model-based based on whether a model of the environment is used. There are three common approaches to model-free RL, i.e. value-based methods, policy gradient methods, and actor-critic methods. Valued-based methods, e.g. Q-learning and its variants, learns the policy indirectly by learning a value function, e.g. $Q_\pi(x, u)$ (Eq. 2.3a) or $V_\pi(x)$ (Eq. 2.3b), and takes the action that maximizes the value function with exploration. γ is the discount factor. Alternatively, one may use the advantage function, $A_\pi(x, u)$ as given in Eq. 2.3c, which could be interpreted as how much a given action improve upon the policy's average behaviour. The value function may be updated via methods such as Bellman backup [199]. A major shortcoming of Q-learning is that it is only applicable to problems with discrete action spaces. Thus, for problems with continuous action space, each continuous action need to discretized into a number of discrete actions.

$$Q_{\pi_\theta}(x, u) = \mathbb{E}_{\pi_\theta}\big[\sum_{k=0}^{\infty} \gamma^k r_{t+k} | x_t = x, u_t = u\big] \tag{2.3a}$$

$$V_{\pi_\theta}(x) = \mathbb{E}_{\pi_\theta}\big[\sum_{k=0}^{\infty} \gamma^k r_{t+k} | x_t = x\big] \tag{2.3b}$$

$$A_{\pi_\theta}(x, u) = Q_{\pi_\theta}(x, u) - V_{\pi_\theta}(x) \tag{2.3c}$$

Policy gradients methods directly search for an optimal policy π_{θ^*}, using stochastic estimates of policy gradients (Eq. 3.1), and are applicable to problems with continuous action spaces. Some examples of policy gradient methods include REINFORCE [199], Truth Region Policy Optimization (TRPO) [188], and Proximal Policy Optimization (PPO) [190]

$$g \coloneqq \nabla_\theta \mathbb{E}_{\pi_\theta}\big[\sum_{k=0}^{\infty} \gamma^k r_{t+k}\big] \tag{2.4a}$$

$$\theta \leftarrow \theta + \alpha \hat{g} \tag{2.4b}$$

Actor-critic methods, e.g. Advantage Actor-Critic (A2C), Asynchronous Advantage Actor-Critic (A3C) [155], and Deep Deterministic Policy Gradient (DDPG) [**?**], are hybrids of the value-based and policy gradient approaches. Actor-critic methods use a policy network to select actions (the actor), and a value network to evaluate the action (the critic). A2C and A3C use a Q-network as critic and thus are only applicable to problems with discrete action spaces, i.e. the same drawback as Q-learning. DDPG, on the other hand, is applicable to problems with continuous action spaces. Although we classified PPO as a policy gradient method, one can and should incorporate a critic for variance reduction and more robust performance.

Alternatively, one can incorporate a model of the environment to improve the sample efficiency. Previously, it was believed that model-based RL could not perform as well as model-free RL asymptotically. But, recent work [53] has shown that model-based RL can

match the asymptotic performance of model-free RL algorithms, while being significantly more sample efficient. A common idea for model-based RL is to simultaneously learn a model of the environment and plan ahead based on the learned model. The classical Dyna-Q [199] is an example of such approach. Developing upon the idea, people have modeled the environment with Gaussian Process [122], locally linear models [214], and neural networks [53].

3. Gnu-RL: A practical and scalable solution for building control

This section summarize the work, published in [38, 39], to address Research Question 1. Also related to these work is [43], which studies the problem of off-policy evaluation in the context of building control.

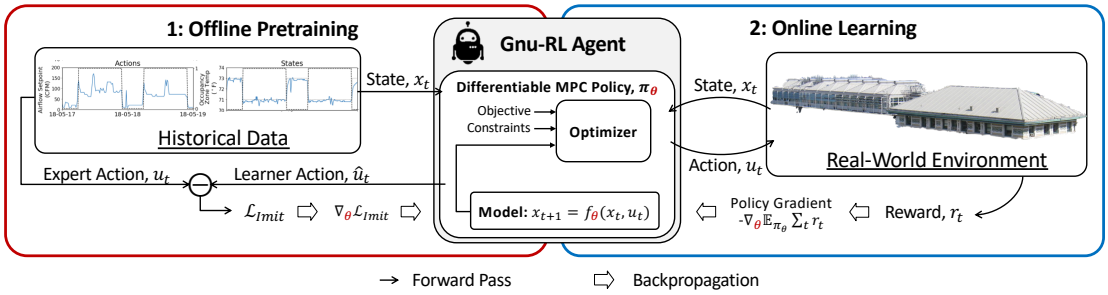


Figure 3.1: Gnu-RL Framework. The Gnu-RL agent utilizes a Differentiable MPC policy, which encodes domain knowledge on planning and system dynamics. In the offline pretraining phase, the agent is initialized by imitating historical data from the existing controller. In the online learning phase, the agent continues to improve its policy end-to-end using a policy gradient algorithm.

3.1. Introduction

To overcome the challenges as summarized in Section 1.4.1, we propose Gnu-RL[8]: a novel approach that enables practical deployment of RL for HVAC control and requires no prior information other than historical data from existing HVAC controllers (Figure 3.1). It leverages a recently-developed Differentiable MPC policy [12] that encodes domain knowledge on system dynamics and control, making it both sample-efficient and interpretable. Prior to any interactions with the environment, a Gnu-RL agent can be pre-trained on historical data through imitation learning, enabling it to match the behavior of the existing controller. Once it is put in charge of controlling the environment, the agent continues to improve its policy end-to-end, using Proximal Policy Optimization (PPO) [190], a policy gradient algorithm.

By integrating an policy gradient algorithm with a Differentiable MPC policy, our proposed approach combined the strength of both MPC and RL. Gnu-RL is the 1st solution that enable real-world deployment of RL agents for building HVAC control, without the resource-intensive process of developing high-fidelity simulation models. In comparison, we only need historical data, which is commonly logged in building automation system (BAS) and is a readily-available information source for existing buildings.

[8]The name Gnu comes from drawing an analogy between RL agents and animals. Gnus are among the most successful herbivores in the African savanna, and part of their success can be attributed to the precociality of their youngsters who are able to outrun predators within a day after their birth.

Gnu-RL was validated on three DOE commercial reference building models [64], a model of a 600m^2 multi-functional space [226] on CMU campus, and a real-world conference room (Section 3.5.3). We evaluated the performance of Gnu-RL via its energy-efficiency and thermal comfort, in comparison to existing control.

3.2. Related Work

In this section, we review literature on optimal control of building HVAC systems. Model predictive control (MPC) and reinforcement learning (RL) were identified as options for optimal control of building systems in [159] and are the two most popular approaches for HVAC control in the literature.

Model Predictive Control for HVAC. Model predictive control is a planning-based method that solves an optimal control problem iteratively over a receding time horizon. Some of the advantages of MPC are that it takes into consideration future disturbances and that it can handle handle multiple constraints and objectives, e.g. energy and comfort [126].

However, it can be argued that the main roadblock preventing widespread adoption of MPC is its reliance on a model [177, 126]. By some estimates, modeling can account for up to 75% of the time and resources required for implementing MPC in practice [182]. Because buildings are highly heterogeneous, a custom model is required for each thermal zone or buliding [147].

Privara et al. [177] identified two paradigms for modeling building dynamics: physical-based and statistical-based. Physical-based models, also referred to as white-box models, e.g. EnergyPlus, utilize physical knowledge and material properties of a building to build detailed representations of the building dynamics. One shortcoming is that such models are not control-oriented [16]. Nonetheless, it is not impossible to use such models for control. For instance, [229] used exhaustive search optimization to derive control policy for an EnergyPlus model. Furthermore, physical-based models require significant modeling effort to develop, because they have a large number of free parameters (e.g. 2,500 parameters for a medium-sized building [124]), and information required for determining these parameters are scattered in different design documents [97].

Statistical-based models assume a parametric model form, which may have physical underpinnings (i.e. grey-box models) or not (i.e. black-box models), and identifies model parameters directly from data. An example of grey-box models is the RC model, which draws an analogy to resistance and capacitance in electric circuits to describe thermodynamics [178]. While statistical-based modeling is potentially scalable, a practical problem is that the experimental conditions needed for accurate identification of building systems fall outside of normal building operations [5]. Alternatively, excitation signals from actuators may be used to help identify model parameters [178, 5, 15], which requires careful design of experiments [33] and may disturb normal operation. Even then, it was still difficult to identify some parameters even with supposedly rich excitation signals [5].

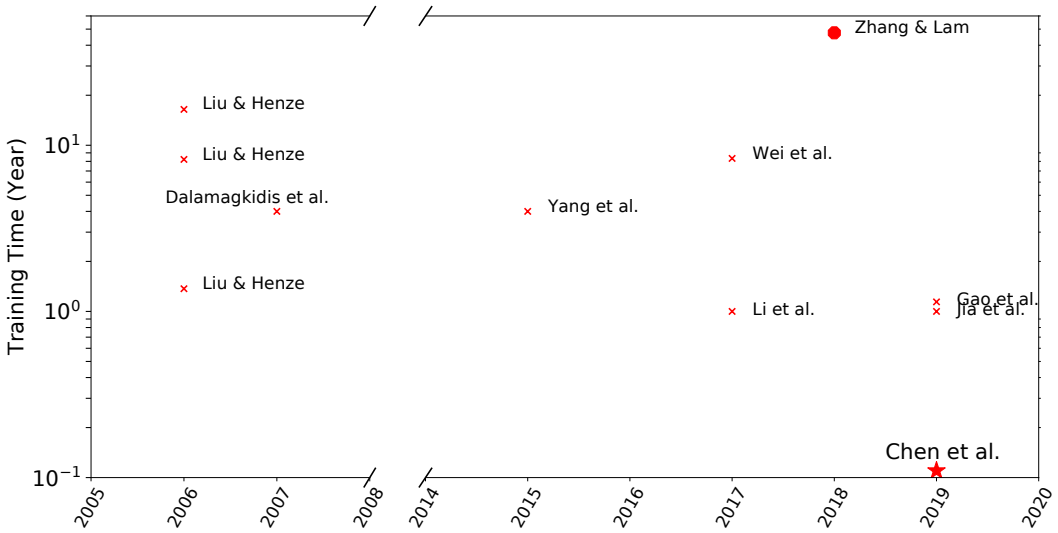


Figure 3.2: Summary of training time reported in the literature; By incorporating domain knowledge and expert demonstration, our proposed Gnu-RL agent drastically reduced training time compared to existing works.

Finally, there are many sources of stochasticity and uncertainty in building dynamics [149], adding to the modeling difficulty. Learning-based MPC [15] accounts for the stochasticity by modeling the building dynamics in a semi-parametric form, where the zone temperature evolves following a linear model and the internal thermal gain is learned from data as a non-parametric term. Learning-based MPC also decouples robustness and performance by using a statistical model to optimize performance and a deterministic model to impose comfort constraints. Adaptive MPC attempts to overcome some of these challenges by updating model parameters online with new observations. Here the objective is to estimate model parameters that minimize the difference between prediction and observation over time. Examples of this line of work include the use of Extended Kalman Filter [85] and Unscented Kalman Filter [149] to simultaneously estimate states and model parameters.

Reinforcement Learning for HVAC. The early works by [145, 61] demonstrated the potential of using RL for optimal HVAC control. However, practical application of RL was limited by its sample complexity, i.e. the long training time required to learn control strategies, especially for tasks associated with a large state-action space [145, 61]. In Figure 3.2, we summarized the training time reported in [145, 61, 219, 142, 215, 226, 117, 86]. We acknowledge the limitations of the summary. Firstly, the RL agents were evaluated in different environments of varying complexity, making direct comparison impossible. We only considered works that evaluated their agents in physics-based emulators, includ-

ing EnergyPlus, TRNSYS, and MATLAB Simulink. Another limitation is that most of these papers reported the total training time. A more meaningful evaluation metric may be the amount of time required to reach the same performance as a baseline controller. Despite these limitations, the observation here is that the amount of time used for training is typically in the order of years. In our work [38], we proposed Gnu-RL, a precocial agent that is capable of controlling HVAC system well at "birth".

To reduce sample complexity, researchers have adopted different approaches, such as injecting domain knowledge [172], pre-training the agent [142, 117], and incorporating a model of system dynamics [59, 164]. For example, [172] initialized a Q-learning algorithm with an MPC prior. However, they assumed a overly simple grey-box model, which has an analytical MPC solution. Such is generally not the case for practical application. Alternatively, [142, 117] used historical data to pre-train the agent. Specifically, [142] populated the replay memory with historical data and [117] used historical state-action pairs for expert demonstrations. Finally, [59, 164] used models to assist RL. [59] used a neural network (NN) based model to supply additional state-action pairs. [164] used a NN as a model for state transitions and incorporated multi-step planning into RL. The experiments in [164] showed that model-based RL was generally superior to model-free RL in terms of sample efficiency, energy performance, and occupants' comfort.

Despite the numerous publications on this topic, applications beyond simulation are numbered. Of the publications referenced earlier, for example, only [145, 226] deployed their RL agents in real-world testbeds. However, they both assumed the existence of high-fidelity models for training their agents in simulation. Such approach shifts the focus back to modeling [219] and is not scalable. Similarly, [59] also reported a real-world deployment, but provided no quantitative results. More recently, Google announced a 40% energy consumption reduction in their data centers [76], but to our knowledge no technical publication of this achievement is available yet. It should be noted, though, that in the broader scope of building systems there are real-world applications of RL in systems other than HVAC, including water heaters [125] and lighting systems [169].

3.3. Preliminaries

We now present background technical concepts used by Gnu-RL.

3.3.1. Proximal Policy Optimization

Policy gradient methods directly optimize the policy π_θ to maximize the expected total reward (Eq. 3.1a). To do that, these methods compute an estimate of the policy gradient defined in Eq. 3.1b and optimize the objective with stochastic gradient ascent (Eq. 3.1c). Aside from the obvious benefit of being able to handle continuous action spaces, policy gradients methods have several advantages over value-based ones, as discussed in [199] . Firstly, policy gradient methods can learn both deterministic and stochastic policies, while there is no natural way to learn stochastic policy with value-based methods. Secondly, the policy may be a simpler function to approximate, and thus policy-based methods typically learn faster and yield a superior asymptotic policy. Finally, the choice

of policy parameterization is a natural way to inject domain knowledge into RL.

$$\theta^* = \arg\max_\theta \mathbb{E}_{\pi_\theta}\big[\sum_{k=0}^{\infty}\gamma^k r_{t+k}\big] \tag{3.1a}$$

$$g \coloneqq \nabla_\theta \mathbb{E}_{\pi_\theta}\big[\sum_{k=0}^{\infty}\gamma^k r_{t+k}\big] \tag{3.1b}$$

$$\theta \leftarrow \theta + \alpha\hat{g} \tag{3.1c}$$

A variety of policy gradient algorithms have been proposed in the literature. Perhaps, REINFORCE [199] is the most well-known one. However, it suffers from large performance variance and unstable policy updates [190]. PPO [190] is the most recent work in a line of research that improved upon vanilla policy gradient methods, including Natural Policy Gradients [120] and Trust Region Policy Optimization (TRPO) [188]. The intuition behind these methods is that in each update the policy π_θ should not change too much. This idea is most clear with the objective of TRPO (Eq. 3.2), which is to maximize an importance weighted advantage estimate, subject to a constraint on the size of the policy update.

$$\begin{aligned} \max_\theta \;\; & \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(u_t|x_t)}{\pi_{\theta_{old}}(u_t|x_t)}\hat{A}_t\right] \\ \text{s.t.} \;\; & \hat{\mathbb{E}}_t\left[\mathrm{KL}\left[\pi_\theta(\cdot|x_t), \pi_{\theta_{old}}(\cdot|x_t)\right]\right] \le \delta \end{aligned} \tag{3.2}$$

However, it is not straightforward to solve the optimization problem posed in Eq. 3.2, due to the constraint. PPO simplified the problem using a surrogate objective, given in Eq. 3.3a. $w_t(\theta)$ denotes the importance weighting of the policy after and before the update, i.e. $\frac{\pi_\theta(u_t|x_t)}{\pi_{\theta_{old}}(u_t|x_t)}$, and ϵ is a hyperparameter. For ease of notation in future discussion, we denote the negative of the objective as $\mathcal{L}_{PPO}$ (Eq. 3.3b). PPO is known to be stable and robust to hyperparameters and network architectures [190]. It was also shown to outperform methods, such as A3C [155] and TRPO [188].

$$\max_\theta \;\; \hat{\mathbb{E}}_t\Big[\min(w_t(\theta)\hat{A}_t, \mathrm{clip}(w_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\Big] \tag{3.3a}$$

$$\mathcal{L}_{PPO}(\theta) = -\hat{\mathbb{E}}_t\Big[\min(w_t(\theta)\hat{A}_t, \mathrm{clip}(w_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\Big] \tag{3.3b}$$

There are a number of options one can use for the advantage estimate $\hat{A}_t$, including total rewards (Eq. 3.4a), Q-function, advantage function, and k-step TD residual (Eq. 3.4b). [189] provides a thorough discussion on possible options and the bias-variance trade-off

of these options.

$$\hat{A}_{t,R} = \sum_{k=0}^{\infty} \gamma^k r_{t+k} \tag{3.4a}$$

$$\hat{A}_{t,TD} = -V_\pi(x_t) + \sum_{l=0}^{k-1} \gamma^l r_{t+l} + \gamma^k V_\pi(x_{t+k}) \tag{3.4b}$$

3.3.2. Differentiable MPC

Since the success of [156], it is common to approximate the policy $\pi_\theta(u|x)$ with a neural network. However, a generic neural network does not encode any domain knowledge on planning or system dynamics, which is abundant in existing HVAC control literature. We hypothesized that encoding such knowledge in the policy would expedite the learning process.

In this paper, we took advantage of a newly developed differentiable MPC policy [12] in place of a neural network. In the forward pass, the differentiable MPC policy solves a box-constrained linear-quadratic regulator (LQR) problem given in Eq. 3.5. Specifically, it finds the optimal trajectory, $\tau^*_{1:T} = \{x^*_t, u^*_t\}_{1:T}$, which minimizes the quadratic cost function over the planning horizon $1:T$ (Eq. 3.5a) and satisfies a linear model of the system dynamics (Eq. 4.1a). Furthermore, the Differentiable MPC implementation allows one to incorporate box constraints on the control action (Eq. 3.5c). It is also possible to use the differentiable MPC policy for non-quadratic cost and non-linear dynamics with local linear quadratic approximation.

$$\tau^*_{1:T} = \arg\min_{\tau_{1:T}} \sum_t \frac{1}{2} \tau_t^T C_t \tau_t + c_t^T \tau_t \tag{3.5a}$$

$$\text{s.t.} \quad x_1 = x_{\text{init}},\ x_{t+1} = F_t \tau_t + f_t \tag{3.5b}$$

$$\underline{u} \le u \le \bar{u} \tag{3.5c}$$

In the backward pass, the differentiable nature of the policy allows us to update the model parameters end-to-end. The learnable parameters are $\{C, c, F, f\}$. The derivatives of the loss with respect to the model parameters can be obtained by differentiating the Karush-Kuhn-Tucker (KKT) conditions of the problem given in Eq. 3.5, using the techniques developed in [13]. While the LQR problem (eq:lqr) is convex, optimizing an objective with respect to controller parameters is not [12]. This is analogous to the dual-estimation problem formulation in Adaptive MPC, i.e. simultaneously estimating states and parameters. Even for a linear system, the dual-estimation problem yields a non-convex problem [85].

3.4. Approach

The overall framework of our approach is summarized in Figure 3.1. To make our agent precocial, we took advantage of domain knowledge on HVAC control and expert demonstration from existing controllers. Specifically, a Gnu-RL agent utilizes a differentiable MPC policy, which encodes domain knowledge on planning and system dynamics. The training includes two phases: offline pretraining and online learning. In the offline pretraining phase, the agent is initialized by imitating the historical state-action pairs from the existing controller. Using this approach, the Gnu-RL agent learns to behave similarly to the existing controller, without any interaction with the environment. Thus, the pretrained agent is precocial and may be deployed into a real-world environment directly with minimal disturbance to normal operation. In the online learning phase, the agent interacts with the environment and improves its policy using a policy gradient algorithm. While the agent already performs reasonably well at the onset of online learning phase, it continues to fine-tune its policy based on new observations.

We first formulate the HVAC control problem, (Section 3.4.1) and then elaborate on the procedures used to train the agent during the offline pretraining phase (Section 3.4.2) and the online learning phase (Section 3.4.3).

3.4.1. Problem Formulation

We adapt the problem formulation in the Differentiable MPC policy (i.e., Eq. 3.5) to HVAC control. We use a linear model for the system dynamics, as shown in Eq. 3.6a. Though building thermodynamics are non-linear in nature, we assume that it may be locally linearized for the state-action space and the temporal resolution that we are interested in [177]. We define the state x_t as the zone temperature and the action u_t depends on the specific problem. One should choose the action such that it is consistent with the linear assumption or linearize it to be so. Besides the state x_t and the control action u_t, we also consider uncontrollable disturbances d_t, such as weather and internal thermal gains. We define the number of states, actions, and disturbances as m, n, p respectively. Thus, $x_t \in \mathbb{R}^m$, $u_t \in \mathbb{R}^n$, $d_t \in \mathbb{R}^p$, $A \in \mathbb{R}^{m \times m}$, $B_u \in \mathbb{R}^{m \times n}$, and $B_d \in \mathbb{R}^{m \times p}$. Eq. 3.6a can be written in the form of Eq. 4.1a, as shown in Eq. 3.6b. While the original formulation of Differentiable MPC policy learns f_t, we only learn B_d since the disturbances may be supplied by predictive models. Thus, the learnable parameters θ are $\{A, B_u, B_d\}$, which characterize the building thermodynamics. Compared to using a neural network policy, the number of free parameters is drastically reduced and the policy is interpretable to engineers. At each time step t, we provide the agent with predictive information on disturbance for the planning horizon, i.e. $d_{t:t+T-1}$.

$$x_{t+1} = Ax_t + B_u u_t + B_d d_t \tag{3.6a}$$

$$= \underbrace{\begin{bmatrix} A & B_u \end{bmatrix}}_{F} \underbrace{\begin{bmatrix} x_t \\ u_t \end{bmatrix}}_{\tau_t} + \underbrace{B_d d_t}_{f_t} \tag{3.6b}$$

Our objective (Eq. 3.7) is to minimize energy consumption, while maintaining thermal comfort. We balance relative importance of thermal comfort and energy with hyperparameter η. One may choose different values of η for occupied and unoccupied periods. We use the quadratic different difference between actual zone temperatures and setpoints as a proxy for thermal comfort, and thus $O_t = \eta_t I_m$ and $p_t = -\eta_t x_{t,\text{setpoint}}$. The cost with respect to actions may be defined based on the specific problem; some options may be $R_t = I_n$ or $s_t = \vec{1}$. Similarly, Eq. 3.7a can be written in the form of Eq. 3.5a, as shown in Eq. 3.7b. The Differentiable MPC allows for a learnable cost function, but we assume the cost function to be specified by engineers.

$$C_t(x_t, u_t) = \frac{1}{2} x_t^T O_t x_t + p_t^T x_t + \frac{1}{2} u_t^T R_t u_t + s_t^T u_t \tag{3.7a}$$

$$= \frac{1}{2} \underbrace{\begin{bmatrix} x_t^T & u_t^T \end{bmatrix}}_{\tau_t^T} \underbrace{\begin{bmatrix} O_t & 0 \\ 0 & R_t \end{bmatrix}}_{C_t} \underbrace{\begin{bmatrix} x_t \\ u_t \end{bmatrix}}_{\tau_t} + \underbrace{\begin{bmatrix} p_t^T & s_t^T \end{bmatrix}}_{c_t^T} \underbrace{\begin{bmatrix} x_t \\ u_t \end{bmatrix}}_{\tau_t} \tag{3.7b}$$

There are some finer points that we need to highlight. The Differentiable MPC policy outputs the optimal trajectory over the planning horizon, i.e. $\tau^*_{t:t+T-1} = \{x_t^*, u_t^*\}_{t:t+T-1}$. However, we only take the first optimal action u_t^* and re-plan at the next time step based on new observations. This avoids compounding model error over time. We use the re-planning procedure for both offline pretraining and online learning. Moreover, since the HVAC control problems we are interested in have continuous action spaces, we use a Gaussian policy [199] around the optimal action u_t^* (Eq. 3.8). σ can be interpreted as the amount of exploration.

$$\hat{u}_t \sim \pi_\theta(u|x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(u - u_t^*)^2}{2\sigma^2}\right) \tag{3.8}$$

It should be noted that [12] demonstrated the Differentiable MPC policy in the context of imitation learning, which is a supervised learning problem. Here we extended it to policy learning, which is generally considered as a harder problem.

3.4.2. Offline Pretraining

Imitation learning is a supervised approach for an agent to learn a policy. The premise is that it is easier for the expert to demonstrate the desired behaviour, compared to asking the expert to encode or fine-tune a policy [192]. In an HVAC control application, the existing controller can be considered as the expert and the historical state-action pairs logged in BAS as expert demonstrations. Specifically, the agent learns the mapping between states to actions, i.e. the policy $\pi_\theta(u|x)$, using expert demonstrations as ground truth.

In behaviour cloning, one minimizes the mean squared error (MSE) loss between the expert actions and learner actions. Since the Differentiable MPC policy also produces next-state predictions, we minimized the MSE loss between the states and actions from the expert and our agent simultaneously, as given by Eq. 3.9. u_t and $\hat{u}_t$ are the actions

from the expert and the learner respectively. x_{t+1} and $\hat{x}_{t+1}$ are the actual next state versus the next state predicted by the learner. The hyperparameter λ balances the relative importance of actions and next-state predictions. For instance, the engineer can choose a larger λ, if he has limited confidence on the actions of the existing controller. The procedures for offline pre-training are outlined in Algorithm 1. We can repeat the procedures in Algorithm 1 for a suitable number of epochs. For parameter selection, we make a train-test split over the expert demonstrations and select $\hat{\theta}$ with the smallest test loss.

$$\mathcal{L}_{\text{Imit}}(\theta) = \sum_t \lambda ||x_{t+1} - \hat{x}_{t+1}||_2^2 + ||u_t - \hat{u}_t||_2^2 \tag{3.9}$$

Algorithm 1: Offline Pre-training - Imitation Learning

Input: A Differentiable MPC policy π_θ;
Input: Expert demonstrations X, U;
Randomly initialize policy parameter $\theta = \{A, B_d, B_u\}$;
for *i = 0, . . . , # Episodes* **do**
  for *t = 0, . . . , # Steps* **do**
    $\hat{u}_t = \pi_\theta(x_t)$;
    $\hat{x}_{t+1} = f_\theta(x_t, u_t)$;
  end
  $\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}_{\text{Imit}}(\theta)$;
end
Output: A pre-trained policy, $\pi_{\tilde{\theta}}$

3.4.3. Online Learning

We adopt a policy gradient algorithm for this paper, because it integrates naturally with the Differentiable MPC policy. To elaborate, one can replace a neural network with a Differentiable MPC policy, and update model parameters, θ, using the same approach as laid out in Eq. 3.1. The procedures for online learning with PPO are outlined in Algorithm 2.

As mentioned in Section 3.3.1, there are a number of possible choices for advantage estimate $\hat{A}_t$. In our prior work [38], we used the total rewards (Eq. 3.4a) for ease of implementation. But, this option also results in the largest variance. On the other hand, using advantage function results in the smallest variance, but the advantage function must be learned first [189], which is problematic given we want our agent to be precocial. A good compromise may be the baselined version of total rewards given in Eq. 3.10, where π_0 refers to the policy from the existing controller and V_{π_0} is its value function. V_{π_0} can be learned offline based on historical data. Since V_{π_0} is not a function of θ, $\nabla_\theta V_{\pi_0} = 0$. Thus, offset a baseline from the total rewards reduces variance without introducing a bias. The intuition of this formulation is well-explained in [117]. Due to nature of HVAC control problem, the rewards fluctuate with weather and operation condition,

e.g. occupied and unoccupied, regardless of the policy. To reduce variance from using raw rewards, one can use offset the rewards by those that would have been obtained by the existing controller. This reformulates the original objective of maximizing expected total reward to improving upon the policy of the existing controller.

$$\hat{A}_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k} - V_{\pi_0}(x_t) \tag{3.10}$$

Algorithm 2: Online Learning - PPO (Modified from [190])

Input: A pretrained policy, $\pi_{\tilde{\theta}}$;
for i = *0, ..., # Episodes* **do**
  $\theta_{old} \leftarrow \theta$;
  for t = *0, ..., # Steps* **do**
    $\hat{u}_t = \pi_\theta(x_t)$;
    $x_{t+1}, r_{t+1} = \text{env.step}(\hat{u}_t)$;
  end
  Compute $\hat{A}_t$;
  With minibatch of size M;
    $\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}_{PPO}(\theta)$;
end

3.5. Experiments and Results

Gnu-RL was validated on three DOE commercial reference building models [64], a model of a 600m^2 multi-functional space [226] on CMU campus, and a real-world conference room.

3.5.1. Experiment 1: Simulation Study on Commercial Reference Buildings

We validated that our proposed approach is indeed a practical and scalable solution for HVAC control. We also show our approach is generalizable across different buildings, by applying it to the `warehouse`, the `small office`, and the `medium office` from the DOE commercial reference buildings [64]. We demonstrated that the Differentiable MPC policy is superior to a generic neural network policy, in that it is interpretable, more sample efficient, and has smaller performance variance. Specifically, we compared the Differentiable MPC policy to a long short-term memory (LSTM) network, with reference to [212]. Finally, we established two performance baselines for benchmarking the RL agents: an optimal LQR, i.e. the theoretical performance upper bound, and a PI controller, which is representative of controllers in existing buildings.

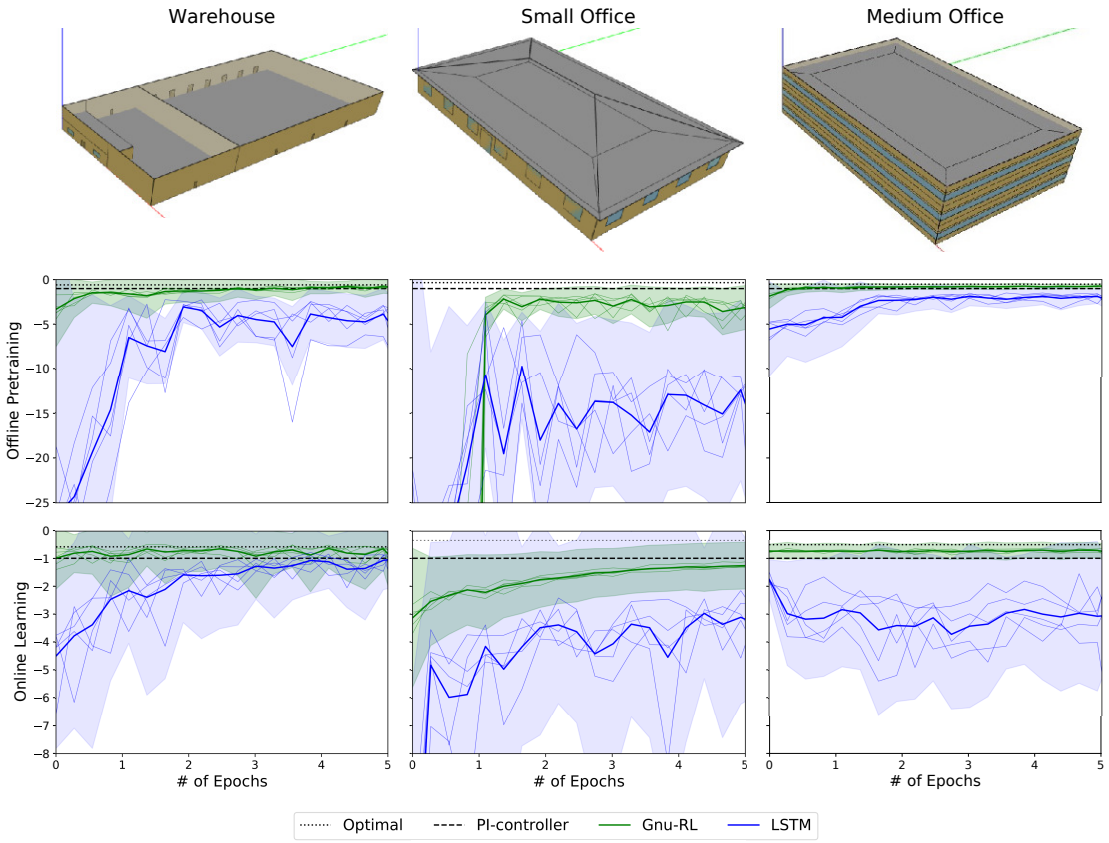


Figure 3.3: Summary of the experiments on the `warehouse`, the `small office`, and the `medium office`; Each experiment is repeated over 5 random seeds. The performance of our proposed Gnu-RL agent is compared to a LSTM policy, during offline pretraining and online learning. We also include two baselines: an optimal LQR and a PI controller. All rewards are normalized by that of the PI controller for the corresponding environment.

Simulation Environments. The simulation environments used in this experiment are based on the `warehouse`, the `small office`, and the `medium office` from DOE commercial reference buildings. We utilized OpenBuild [93]], a toolbox for co-simulation and controller design, for fast prototyping of controllable environments based on the EnergyPlus models. OpenBuild abstracts away the complexity of HVAC system and allows control over the heat flux to zones of the building directly. Furthermore, OpenBuild creates a linear state-space model of the building thermodynamics, such that the optimal performance may be calculated analytically to benchmark the performance of our proposed approach.

Each linear state-space model is created based on the RC modeling framework, where the building envelope is represented as a connected graph of thermal nodes. We denote the temperature at these thermal nodes as $z_t \in \mathbb{R}^l$, where l is the number of thermal nodes. The disturbance d_t to each thermal node includes weather and internal thermal gains from lighting, equipment, and occupancy. Both the RC model parameters and disturbances are calculated based on building and weather descriptions from the EnergyPlus files. We tabulated the dimensions of the state-space model created by OpenBuild in Table 3.1.

Table 3.1: Dimensions of the state-space models of the `warehouse`, the `small office`, and the `medium office`; Only the zone temperatures, i.e. the states, are observable, while the temperature at the other thermal nodes are not.

	Warehouse	**Small Office**	**Medium Office**
# of thermal nodes (l)	99	144	358
# of states (m)	2	5	15
# of actions (n)	2	5	15
# of disturbances (p)	27	57	85

We assume that only the zone temperatures x_t are observable. As one can see in Table 3.1 the number of zones is much smaller than that of the thermal nodes, i.e. $m << l$. The actions u_t are the heat flux to each zone, and thus $m = n$ and B_u is a square matrix for each environment. For all three buildings, we conducted the experiment on Typical Meteorological Year 3 (TMY3) weather sequence [217] in Chicago. For this experiment, we defined the cost function as in Eq. 3.11 and use a constant $\eta = 10$.

$$C_t(x_t, u_t) = \frac{\eta}{2}||x_t - x_{t,\text{setpoint}}||_2^2 + \frac{1}{2}||u_t||_2^2 \tag{3.11}$$

Implementation Details. All three environments used a 15-min simulation and control time step. Both the Gnu-RL agent and the LSTM agent plan ahead for 6 steps, i.e. a 1.5-hour planning horizon. We considered each calendar day as an episode and each calendar year as an epoch. To making training easier, particularly for the LSTM policy, we incorporated an episodic reset mechanism, i.e. the temperature at all thermal nodes was reset to setpoint at the beginning of each day. This prevents the agent from being stuck at undesirable state-space for excess amount of time. We used min-max normalization

to normalize all disturbance terms to 0-1. We also provided the agent with ground truth information on future disturbances.

RL for all our experiments[9] was implemented in PyTorch [171]. Following [12], we used RMSprop [203] as the optimizer for the Differentiable MPC policy and ADAM [128] as the optimizer for the LSTM policy. The LSTM policy has 2 layers, each with 32 units and ReLU activation. To have a fair comparison, the LSTM policy has the same input and output as the Differentiable MPC policy, i.e. input $= \{x_t, d_{t:t+T-1}\}$ and output $= \{u_{t:t+T-1}\}$.

All experiments were repeated over 5 random seeds. For offline pretraining, we used a learning rate of 1×10^{-3}, except for the Gnu-RL agent in the `small office`, where we used a learning rate of 1×10^{-2} due to the particularly bad initialization. For the Differentiable MPC policies, we initialized A and B_u to be identity matrix and randomly initialized B_d with a uniform distribution over [0, 0.1]. The LSTM policies were initialized by PyTorch defaults. While we minimizes the imitation loss during offline pretraining, we evaluated the performance directly by letting the agents control the environment. Specifically, we froze the policy every 100 episodes and let the agent control the environment with a reduced amount of exploration, i.e. $\sigma = 0.1$. We report the mean and standard deviation of the episodic rewards over 30 randomly sampled episodes.

For online learning, we used a learning rate of 2.5×10^{-4} for the Differentiable MPC policy, and a learning rate of 5×10^{-4} for the LSTM policy. We evaluated the performance following the same procedure as in offline pretraining, i.e. we evaluate the policy every 100 episodes and report the mean and standard deviation of rewards over 30 episodes. We re-scaled the reward to be around 1, for better performance [105]. For hyperparameters, we used $\gamma = 0.8$, $\epsilon = 0.1$, and M = 48. For the Differentiable MPC policy, we used a σ that linearly decayed from 1 to 0.1. For the LSTM policy, we let the neural network learn σ simultaneously.

The results for both offline pretraining and online learning are summarized in Figure 3.3, where we compared the performance of the Differentiable MPC policy with that of the LSTM policy. For each experiment, we average the mean and the standard deviation of episodic rewards over the 5 runs and show the confidence interval of one standard deviation around the mean. At the same time, we also plotted the performance of individual runs with a thinner line weight. For ease of comparison, we normalize all rewards by that of the PI controller for the corresponding environment.

We also compared the performance of RL agents with two baselines: an optimal LQR and a PI controller.

Optimal LQR: Since OpenBuild linearized the system dynamics, one can derive the optimal performance analytically for each environment with LQR. We assume the LQR has ground truth parameters of the model, full observability over all the thermal nodes z_t, and perfect predictive information of disturbances. These assumptions are not realistic. But, this provides us with a theoretical upper bound for the control performance.

PI Controller: For a more realistic performance baseline, we developed a PI controller

[9]The code is available at https://github.com/INFERLab/Gnu-RL.

for each zone with MATLAB PID tuner [153] based on ground truth model parameters. This is representative of controllers in existing buildings. Furthermore, we simulate state-action pairs with these PI controllers as expert demonstration.

Results: Offline Pretraining. In the offline pretraining phase, the agents were pretrained by imitating expert demonstration from the PI controllers. We trained all agents for 5 epochs, i.e. we go through 1-year worth of expert demonstration for 5 times. As shown in Figure 3.3, the performance had plateaued by then. By the end of offline pretraining, the Gnu-RL agents were performing similarly to the PI controllers. In fact, the Differentiable MPC policy outperformed the PI controller in the `warehouse` and the `medium office`. We hypothesize that the domain knowledge encoded in the Differentiable MPC policy enabled the agent to extrapolate beyond expert demonstration. While the Gnu-RL agent in the `small office` was not performing as well as the PI controller, it drastically improved upon its poor initial performance. Furthermore, given the same information, the Differentiable MPC policy achieved significantly better performance than its LSTM counterpart. This phenomenon was also observed in [12]. Due to its encoded knowledge, the Differentiable MPC policy was able to learn with lower sample complexity compared to a neural network. Finally, the Differentiable MPC policy has much smaller performance variance than the LSTM policy, which is desirable in practice. The same characteristics was also observed during online learning.

Note that the Differentiable MPC policies were initialized in the same way for the three environments, which worked well for the `warehouse` and the `medium office`, but not for the `small office`. This implies the initialization scheme should be based on the specific environment. Since the parameters could be trapped in local minimal, it is preferable to initialize as well as possible. Engineering estimates of the parameters, which have well-defined physical meaning, may be a more appropriate initialization scheme.

Results: Online Training. In the online training phase, the agents continue to improve their policies through direct interaction with the environment. We trained all agents for 5 epochs, i.e. we go through the TMY3 weather sequence 5 times. As shown in Figure 3.3, the Gnu-RL agents in the `warehouse` and the `medium office` were already performing better than the PI controller at the onset of online training phase and thus they basically provided energy savings and / or comfort improvement for free. On the other hand, the LSTM policy consistently under-performed the Gnu-RL agents over the 5-year training period. In fact, the best-performing LSTM policy only approached the performance of the PI controller in the `warehouse` at the end of 5-year training period. Theoretically, a sufficiently expressive neural network policy would eventually outperform the model-based Differentiable MPC policy. But, that is not meaningful for practical applications.

In the `warehouse`, the Gnu-RL agent improved its performance approaching the optimal. But, we also see fluctuations in performance. This may be a result of learning rate being too large, when the performance was already close to optimal. The performance of the Gnu-RL agent in the `small office` suffered due to the poor initialization.

Regardless, the agent improved its policy and approached the performance of the PI controller over time. This again highlights the importance of having a reasonably good initialization. Alternatively, other improvements may be available to expedite the learning. In the `medium office`, the performance curves for both the Gnu-RL agent and the LSTM policy were close to flat throughout the training period. We hypothesize the large state-action space of this environment makes convergence difficult [145].

3.5.2. Experiment 2: Simulation Study on Intelligent Workspace

We also validated our approach in a simulation environment with detailed HVAC system. Specifically, we trained and evaluated our agent using the EnergyPlus model from [226], which was modeled after the Intelligent Workspace (IW) on Carnegie Mellon University (CMU) campus. For offline pretraining, we used a baseline P-controller for expert demonstration and simulated the state-actions pairs under the TMY3 weather sequence, from Jan. 1st to Mar. 31st. We pretrained our agent on the simulated state-action pairs. For online learning, We deployed our agent in the simulation environment, using the weather sequence in 2017 from Jan. 1st to Mar. 31st. Since the simulation environment, the state-action space, and the weather sequence for training and testing are the same as those in [226], our results are directly comparable. However, [226] assumed the existence of a high-fidelity model for training, while we only assumed the existence of historical data from the existing controller.

To understand how our approach compare to MPC, we compared imitation learning with system identification during the offline pretraining stage, and policy gradient methods with Adaptive MPC during the online learning stage. In the offline pretraining phase, we initialized our agent with imitation learning. In comparison, it is possible to initialize the agent with system identification using the same information. System identification is the class of methods that estimate model parameters of a dynamic system based on input and output signals [146]. Specifically, prediction error methods (PEM) look for parameters that minimize the difference between predicted states and observed states. For online learning, we compared our approach to Adaptive MPC. RL algorithms update model parameters end-to-end, with the objective of maximizing expected reward. On the other hand, it is also possible to update parameters online using Adaptive MPC, with the objective of minimizing prediction error [85, 149].

Simulation Testbed. The IW (Figure 3.4) is a 600m^2 multi-functional space, including a classroom, a common area, and offices. We used the same EnergyPlus model used in [226], which was calibrated against operational data. In this experiment, we controlled the water-based radiant heating system. Figure 3.4b shows a schematic of the system and the control logic. The hot water is supplied by a district heating plant. The supply water (SW) flow is kept constant and the supply water temperature is controlled by a P-controller to maintain zone temperature. We trained our agent to control the supply water temperature in place of the existing P-controller during the heating season. The allowable range of supply water temperature is 20-65$^\circ C$. The variables considered for this problem are listed in Table 3.2. The cost function used for this experiment and the

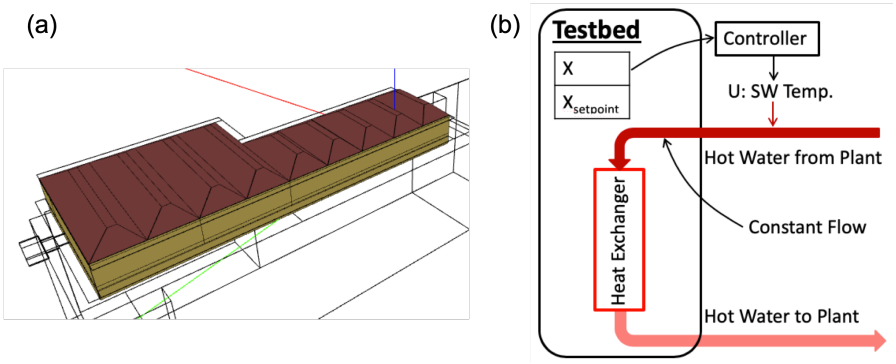


Figure 3.4: Simulation testbed based on Intelligent Workspace. (a) is a geometric view of the EnergyPlus model rendered by OpenStudio [99], and (b) is a schematic of the water-based radiant heating system

real-world experiment presented in Section **??** in given in Eq. 3.12.

$$C_t(x_t, u_t) = \frac{\eta_t}{2}||x_t - x_{t,\text{setpoint}}||_2^2 + \frac{1}{2}||u_t||_1 \tag{3.12}$$

Table 3.2: The state, action, and disturbance terms defined for the simulation study on Intelligent Workspace, a 600m^2 multi-functional space

X - State	Zone Temperature ($°C$)
U - Control Action	SW Temperature ($°C$)
D - Disturbance	Outdoor Air Temperature ($°C$)
	Outdoor Air Relative Humidity (%)
	Diffuse Solar Radiation (W/m^2)
	Direct Solar Radiation (W/m^2)
	Occupancy Flag
	Wind Speed (m/s)

Implementation Details. The implementation details are the same as Section 3.5.1, unless specified otherwise. We used the OpenAI Gym [30] wrapper for EnergyPlus developed in [226] to interface with the simulation environment. The EnergyPlus model has a 5-minute simulation time step. Following [226], each action was repeated for 3 times (a 15-min control time step). The agent plans ahead for 12 steps (a 3-hour planning horizon). We shifted the 20-65$°C$ range of supply water temperature setpoint to 0-45$°C$ for the control action. We used $\eta = 3$ during occupied periods and 0.1 during unoccupied periods. For offline pre-training, we used a learning rate of 1×10^{-4} and a λ of 100. λ was adjusted so that loss from states and loss from actions were about the same magnitude. During online training, we used a learning rate of 5×10^{-4}.

Results: Offline Pretraining. Note that the existing P-controller for supply water temperature operates 24/7, which is not the intended behaviour for our agent and is not a fair comparison.[10] Instead, we modified the existing P-controller to be operational only during occupied periods, and call it the baseline P-controller. We simulated state-action pairs using the baseline P-controller under TMY3 weather sequence from Jan. 1st to Mar. 31st, as expert demonstrations.

We compared the performance of initializing model parameters, i.e. $\theta = \{A, B_u, B_d\}$, with imitation learning and system identification. We used PEM for system identification, as described in [177]. We assumed the same model (Eq. 3.6a) and used the same time series for both initialization schemes. We evaluated the performance of the two initialization schemes by letting the pretrained agents control the simulation environment under the TMY3 weather sequence, with fixed parameters. Figure 3.5 shows the behavior of the initialized agents over a five-day period, neither of which had interacted with the environment before. The agent initialized with imitation learning behaved similarly to the baseline P-controller and tracked temperature setpoint well. The agent initialized with system identification, however, consistently underestimated the amount of heating required, despite its small prediction error (RMSE = 0.15 $°C$).

The poor performance of the agent initialized with system identification is not surprising, as the experimental conditions required for accurate identification of building systems fall outside normal building operations [5]. In practice, excitation signals from actuators were often necessary to identify model parameters [178, 5]. However, such procedure requires careful design of experiments [5] and may disturb normal operation. Instead, we successfully initialized the agent with imitation learning on observational data, which neither required experimentation nor disturbed occupants. The superior performance of imitation learning can be attributed to the fact that the agent imitated the expert on top of learning system dynamics. Learning how the expert would have acted under a given circumstance was directly relevant to the control task.

Results: Online Training. After pretraining, our agent controlled the environment using the actual weather sequence in 2017. The left hand side of Figure 3.6 shows the behaviour of Gnu-RL at the onset of training for a four-day period. Gnu-RL already knew how to track temperature setpoint as well as the baseline P-controller, despite the fact that it had not interacted with the environment before. In comparison, a recent publication on the same environment [226] took 47.5 years in simulation to achieve similar performance to the existing controller.

The results with comparison to [226] are tabulated in Table 3.3. The heating demand and predicted percent dissatisfied (PPD) were calculated by EnergyPlus. Similar to [226], we only considered PPD during occupied periods. Our agent saved 6.6% energy and better maintained comfort compared to the best performing RL agent in [226]. To understand where the energy savings come from, we show a close-up view of the state-action pairs over a single day on the right hand side of Figure 3.6. While the baseline

[10] To illustrate the control strategy used by the existing P-controller, we included the data traces from the actual system from Jan. 1 to Jan.4, 2017 in Figure 3.6.

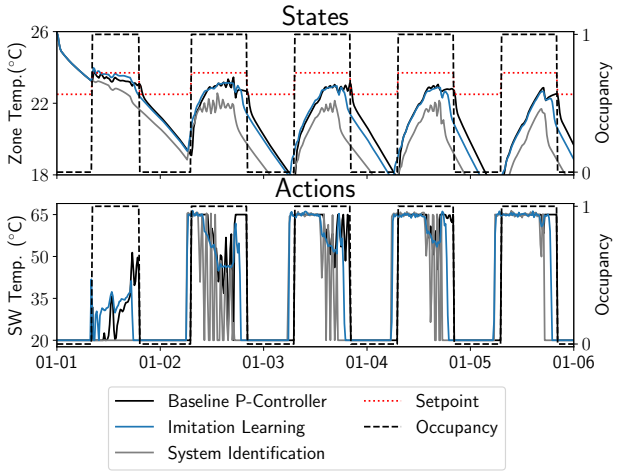


Figure 3.5: Comparison of two initialization schemes: imitation learning vs. system identification (evaluated on TMY3 weather sequence); The agent initialized with imitation learning behaved similarly to the baseline P-controller, while the agent initialized with system identification consistently underestimated the heat requirement. Both agents were initialized with the same information, i.e. export demonstration from the baseline P-controller.

P-controller heats up the space following a fixed occupancy schedule, Gnu-RL preheats the space prior to occupancy and lets temperature float towards the end of occupancy. This explains the savings with respect to the baseline P-controller. It is worth noting that the preheating behaviour was not present in the baseline P-controller. The knowledge embedded in the Differentiable MPC policy enabled our agent to extrapolate beyond expert demonstration.

Our approach finds model parameters that maximize expected reward using a policy gradient algorithm. Alternatively, Adaptive MPC updates model parameters online by minimizing prediction error. We compare the performance of two approaches with their respective objectives: minimizing prediction error (Eq. 3.13) and maximizing expected reward (Eq. 3.3). To minimize prediction error, we use the same procedures as in Algorithm 2, but use $\mathcal{L}_{PEM}$ in place of $\mathcal{L}_{PPO}$. Both agents are initialized with the same parameters from imitation learning.

$$\mathcal{L}_{\text{PEM}}(\theta) = \sum_{t} (x_{t+1} - \hat{x}_{t+1})^2 \tag{3.13}$$

Figure 3.7 compares the performance of optimizing two different objectives over time. Because the rewards are also a function of the weather sequence, we show the difference between rewards from the agent and that from the baseline P-controller, which we call residual reward. While optimizing $\mathcal{L}_{PEM}$ resulted in consistently smaller prediction error, optimizing $\mathcal{L}_{PPO}$ resulted in larger overall reward. Table 3.3 also shows that the

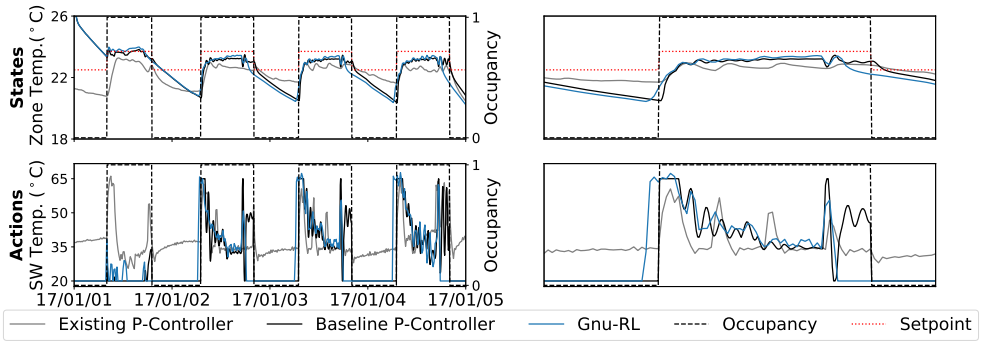


Figure 3.6: Performance of the Gnu-RL agent at the onset of deployment; The Gnu-RL already tracked temperature setpoint as well as the baseline P-controller.

PEM agent failed to maintain comfort, despite its small prediction error. One way to interpret this result is that minimizing prediction error is only a surrogate for learning a control policy [12]. It is clear from the comparison that small prediction error does not necessarily translate to good control performance. We observed a similar result in Section 3.5.2. Another common observation from both comparisons is that it is highly-effective to directly optimize the task objective, whether it is imitation or control.

Table 3.3: Comparison of performance during online learning phase

	Heating Demand (kW)	**PPD Mean (%)**	**PPD SD (%)**
Existing P-Controller [226]	43709	9.45	5.59
Agent #6 [226]	37131	11.71	3.76
Baseline P-Controller	35792	9.71	6.87
Gnu-RL	34687	9.56	6.39
Gnu-RL + $\mathcal{L}_{PEM}$	24901	18.77	12.48

3.5.3. Experiment 3: Real-World Deployment

Given the promising results in our simulation studies, we repeated our experiment in a real-world conference room on campus, during Jun. 5th-Jun. 25st, 2019 to validate that our approach can make possible real-world deployment of RL for HVAC control with no prior information other than historical data. While the procedure here follows the same framework, there are additional challenges from a real-world deployment. Firstly, the existing controller in our testbed is not able to track temperature setpoint well. Thus, our agent needed to learn from sub-optimal expert moves. Secondly, real-world deployment demands a higher-level of robustness compared to simulation study. For instance, the agent's intended actions are not necessarily the same as the actions taken, e.g. there is

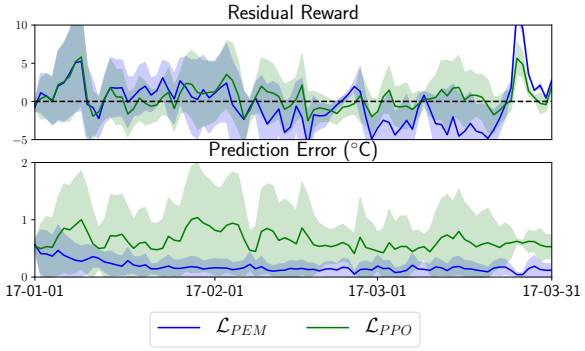


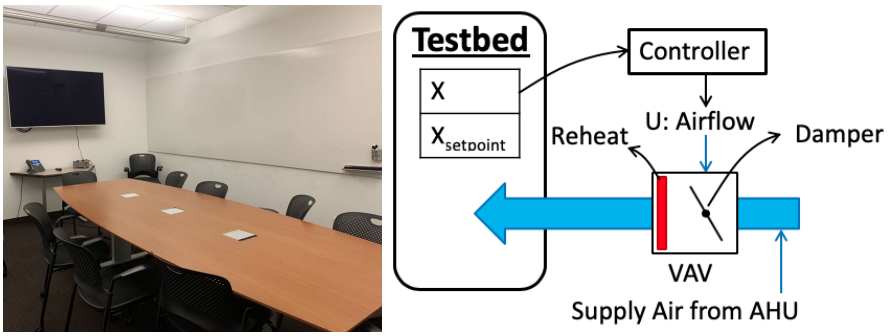
Figure 3.7: Comparison of two approaches Policy Gradient ($\mathcal{L}_{PPO}$) vs. Adaptive MPC ($\mathcal{L}_{PEM}$); While optimizing $\mathcal{L}_{PEM}$ resulted in consistently smaller prediction error, optimizing $\mathcal{L}_{PPO}$ resulted in larger overall reward.

a 1-2 minute delay with the BAS interface. Finally, RL is sensitive to hyperparameters and other implementation details [105]. However, it is difficult to fine-tune these design choices in a real-world deployment. We resorted to using the implementation details that worked well for the simulation study (Section 3.5.2) unless specified otherwise, although the implementation details may not be optimal for this specific problem.

To validate that Gnu-RL indeed make possible real-world deployment of RL agents using only historical data, we deployed it in a conference room on CMU campus for a three-week period, during Jun. 5th-Jun. 25st, 2019. The Gnu-RL agent was pretrained on a month of historical data from the summer in 2017 and 2018. Throughout the three-week experiment, the Gnu-RL agent continuously improved its policy, and learned to maintain thermal comfort well despite the complex occupancy pattern by the end of the experiment.

Testbed. The conference room is a 20m^2 single-zone space (Figure 3.8a) controlled by a variable air volume (VAV) box. Figure 3.8b shows a schematic of the HVAC system and the control logic. In the cooling season, the VAV box discharges a variable volume of cool air into the room. The cool air is supplied by an air handling unit (AHU) at $55°F$. In this experiment, we controlled the amount of airflow that was supplied to the room. We let the existing PID controller determine the damper position to meet our proposed airflow setpoint. The VAV box is also equipped with a hot-water-based reheat coil, which was kept closed throughout the experiment for energy efficiency. The variables used in the problem is listed in Table 3.4. In the existing control logic, the maximum allowable airflow is 200 CFM, and the minimum allowable values are 10 CFM for unoccupied periods and 35 CFM for occupied periods. We followed the same upper and lower bounds for our control action.

Results: Offline Pre-training. We used the time sequences from May 1 to August 31 in 2017 and 2018 for training and testing respectively. We manually selected days where



(a) A photo of the testbed (b) A schematic of the VAV under control

Figure 3.8: Real-world Testbed

Table 3.4: The state, action, and disturbance defined for controlling the conference room

X - State	Indoor Air Temperature ($^\circ F$)
U - Control Action	Supply Airflow (CFM)
D - Disturbance	Outdoor Air Temperature ($^\circ F$)
	Discharge Air Temperature ($^\circ F$)
	Occupancy Flag
	Occupancy Count

the controller tracks the temperature reasonably well: 20 days from 2017 for training and 13 days from 2018 for testing. Again, We pre-trained our agent using imitation learning. We iterated over the training set for 20 epochs and picked the set of parameters with smallest test loss. The MSE were 0.1 and 0.028 for the state and normalized action, respectively.

Results: Online Learning. Figure 3.9 shows how our agent's behavior evolved over the three-week experiment period. Each snapshot shows the state-action pairs over a one-day period. Initially (6/10), our agent knew to pre-cool the space before scheduled occupancy, but it tended to overshoot. At Week 2 (6/17), the agent was no longer overshooting. But, it consistently underestimated the amount of cooling required to maintain temperature. By the end of the experiment (6/24), the agent was tracking setpoint reasonably well despite the varying number of occupants. Also shown in Figure 3.9, there were quite a few discrepancies between the meeting schedule and the real-time occupancy counts. Other than that, there were also counting errors from the occupancy sensor. For instance, there is a positive cumulative counting error towards the end of 6/17.

The performance of our agent with comparison to the existing controller, which follows a fixed occupancy schedule from 6am to 10pm, is summarized in Table 3.5. The Gnu-RL agent saved 16.7% of total cooling demand compared to the existing control strategy, while tracking temperature setpoint significantly better. It should be noted that the

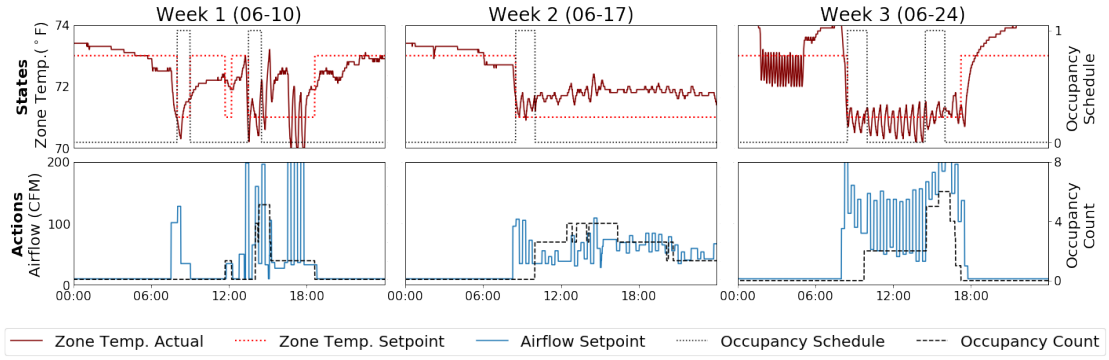


Figure 3.9: Performance of the Gnu-RL agent during a three-week real-world deployment; The Gnu-RL agent continuously improved its policy over time.

Table 3.5: Summary of results for real-world deployment

		Cooling Demand (kWh)	**OAT Mean** ($^\circ F$)	**OAT SD** ($^\circ F$)	**IAT RMSE** ($^\circ F$)
Existing Controller	Jun. 2017	169.4	69.6	6.9	2.4
	Jun. 2018	130.7	71.9	7.1	2.7
	Normalized	99.4	-	-	-
Gnu-RL		82.8	69.9	6.2	1.02

cooling demand is not proportional to the energy consumption. The total cooling demand[11] of the existing controller is calculated using the historical data from Jun. 2017 and 2018 and is normalized for the duration of the experiment and outdoor air temperature (OAT), following the Weather Normalized Method suggested by Energy Star [195]. Since it is difficult to calculate predicted percentage of dissatisfied (PPD) [77], a thermal comfort metric, for the real-world deployment, we used the RMSE between indoor air temperature (IAT) and setpoint as a proxy for comfort and evaluate it only during occupied periods.

3.6. Discussion and Conclusions

We proposed Gnu-RL, a precocial RL agent that is capable of controlling HVAC at "birth". To achieve this, we bootstrapped our agent with domain knowledge and expert demonstration. We demonstrated in both simulation studies and a real-world deployment that Gnu-RL had reasonably good initial performance and continued to improve over time. Firstly, we demonstrated that the Gnu-RL agent is scalable, by applying it to three different buildings. Furthermore, we showed that the Differentiable MPC pol-

[11] The cooling demand is calculated as $\dot{Q} = c\dot{m}\Delta T$, where Q is the cooling demand, c is the specific heat of air, m is the amount of airflow, and ΔT is the difference between mixed air temperature and supply air temperature from the AHU. Mixed air is the mixture of recirculation air and outdoor air.

icy is superior to a LSTM policy in that it is interpretable, more sample-efficient, and has smaller performance variance. In another simulation study, we benchmarked our approach to a recent publication, and our agent saved 6.6% energy compared to the best performing RL agent in [226], while maintaining occupants' comfort better. We also compared our approach to alternatives, i.e. system identification and Adaptive MPC, and demonstrated that it is more effective to optimize task objectives end-to-end. In the real-world conference room, where Gnu-RL was deployed for a three-week period, it saved 16.7% of cooling demand compared to the existing controller, while tracking the temperature setpoint better.

All the energy savings were achieved without the need for a high-fidelity model. Thus, to use our approach in practice, an engineer only needs to specify the state, action, and disturbance terms of interest and define the cost function. The only prior information we used was historical data from the existing controllers. While we discussed our approach in the context of HVAC, it is readily transferable to the control of other building systems. Furthermore, the requirement of historical data does not preclude the usage of this method on new buildings. Since, there are only a small number of free parameters and these parameters have well-defined physical meaning, it is straightforward to initialize these parameters with engineering calculations.

In summary, our proposed approach, Gnu-RL, was shown to be a promising practical and scalable RL solution for HVAC control. However, there are many potential improvements to explore in future work, starting by relaxing some of the assumptions made here. For example, we assumed that future occupancy information was available, which is seldom the case. As future work, we will incorporate a probabilistic occupancy model into the RL framework. For reference, [139] presented the close connection between RL and probabilistic graphical models. Similarly, we assumed that building systems can be locally linearized. The assumption worked for the problems we considered, but it may or may not extrapolate to more complex problems.

Additionally, we identified a few directions for further research. Firstly, there is a need for standardized evaluation, including common simulation testbeds, baseline controllers, and evaluation procedures, such that researchers can compare their results on equal footings. As a step towards this direction, we conducted a simulation study in the same environment as in [226], along with the same state-action space and weather sequence, making our results comparable. We also make our code publicly available at https://github.com/INFERLab/Gnu-RL. Regarding evaluation procedures, we refer readers to [105], which provided a thorough discussion on the challenges in reproducing RL results, along with recommendations.

Secondly, there is a need to develop offline evaluation procedures for pretrained agents [71]. To elaborate, in our real-world deployment, we could only observe the imitation loss from our agent after offline pretraining. Yet, it was an indirect proxy for control performance. In fact, our agent tended to overshoot at initialization, contrary to our expectation based on the imitation loss. Thus, there is a need to evaluate the performance of pretrained agents without access to the environment. This is important in practice to reassure building owners / operators the expected performance of a novel controller before deploying it in a real building.

Thirdly, there are a number of engineering decisions one needs to make when applying our approaches, ranging from initialization of model parameters, the hyperparameter η that balances energy and comfort, and other hyperparameters used during online learning. Furthermore, one cannot expect to do hyperparameter selection for real buildings in the same way as in simulation. We made those decisions based on implementation details used in the literature and engineering judgments. While these decisions generally worked well in our experiments, they do not guarantee good results (recall the experiment on the `small office` in Section 3.5.1). More experiments on different environments may lead to additional insights on how to make these decisions intelligently.

Finally, control problems with larger state-action spaces generally require longer learning/training time [145]. This is observed in our experiment on the `medium office`. In existing buildings, HVAC controllers operates independently based on their local information. But, for scenarios where whole-building control is necessary, there may be a need for multi-agent RL, where the large original problem is subdivided into more tractable sub-problems.

4. COHORT: Coordination Of HeterOgeneous Residential TCLs

This section summarize the work, published in [42], to address Research Question 2 (Section 1.4.2). [46] is a precursor of [42], and [213] is an extension of it, where the modeling aspect is published as a simulation environment for RL research.

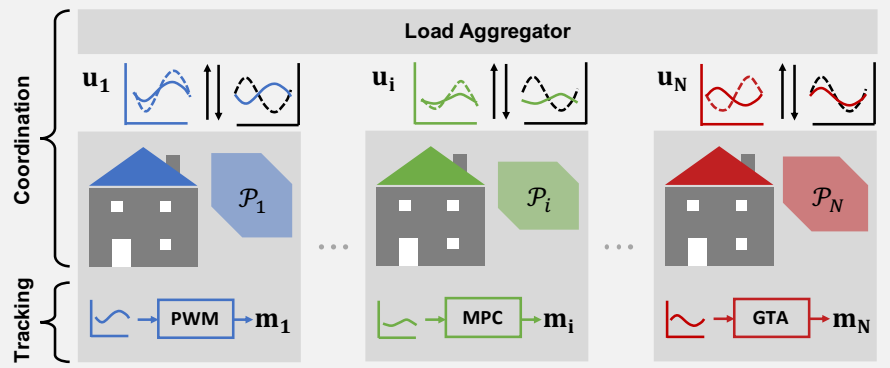


Figure 4.1: COHORT. The load aggregator coordinates a population of TCLs to jointly optimize a grid-level objective, while satisfying each TCL's constraints, characterized by the set $\mathcal{P}_i$. The load aggregator and each TCL coordinates at the level of its power trajectory, $\mathbf{u_i}$, until a consensus is reached among the population. Each TCL is responsible for its own control and tracks $\mathbf{u_i}$ locally with its preferred strategy.

4.1. Introduction

Growing peak demand in some regions and increasing penetration of renewable generation in others are presenting challenges for grid operators to balance supply and demand [75, 118]. Traditionally, demand-side load is viewed as uncontrollable, while supply-side resources manage power generation to match it. An emerging paradigm is to tap into the flexibility of demand-side resources to reduce, shift, or modulate their loads in response to price or control signals [165]. Such demand flexibility can be utilized to provide grid services, improve grid resiliency [165], and reduce operating costs [35].

In this work, we focus on load control[12][35] of residential thermostatically controlled loads (TCLs), such as air conditioners (ACs), refrigerators, and electric water heaters, which account for about 20% of the electricity consumption in the United States (US) [103]. Due to their inherent flexibility through thermal inertia, they can provide grid services without compromising their end uses. However, there are two challenges to utilizing TCL flexibility. Firstly, TCLs must be aggregated across a population to be a meaningful resource at the grid-level [165], which results in a control problem with a

[12] As opposed to methods based on economic incentives. We refer interested readers to [35] for a comparison of load control vs. price-based methods.

large state-action space. Secondly, the constraints posed by each TCL are combinatorial and thus non-convex [31], due to the fact that a TCL operates in discrete action space, i.e., *on* or *off*.

We present a novel framework (Figure 4.1) for the Coordination Of HeterOgeneous Residential Thermostatically controlled loads (COHORT) to jointly optimize a grid-level objective, while satisfying each TCL's end-use requirements and operational constraints. To effectively handle the large state-action space, we adopt a distributed control architecture, where each TCL is responsible for its own control and coordinates with others to find a grid-level solution. Similar to [31], we decompose the grid-level problem into smaller subproblems and coordinate their solutions using the alternating direction method of multipliers (ADMM) [29]. The advantages of the distributed architecture compared with centralized and decentralized approaches are elaborated in Section 4.2. To address the second challenge, we characterize each TCL's flexibility, i.e., the set of all admissible power profiles [230], as a convex set through relaxation. As a result, COHORT is computationally viable for tasks with long planning horizons (e.g., 24 hours), thereby addressing a limitation of [31]—namely that its computational cost grows exponentially with the planning horizon. After coordination, we use low-frequency pulse width modulation (PWM), inspired by [32], to translate the solution of the convex-relaxed problem back to *on*/*off* activation. Since the coordination process makes no assumption on each TCL's control scheme, the TCL may opt for alternative strategies with reference-tracking capability, such as model predictive control (MPC) and global thermostat adjustment (GTA).

COHORT can incorporate detailed, system-specific dynamics and constraints of individual TCLs. At the same time, its computational cost scales well with both population size and planning horizon. As a result, COHORT is generalizable to a wider variety of grid objectives, compared to existing methods, which we demonstrate through three distinct use cases: generation following, minimizing ramping, and peak load curtailment. As a proof of concept, we evaluated COHORT in simulation, targeting challenges arising from increasing penetration of renewable generation [118] under, generation following and minimizing ramping, based on load profiles from California Independent System Operator (CAISO) [115].

Then, we validated that COHORT is practical for real-world TCLs. To do that, we developed a hardware-in-the-loop (HIL) simulation, including a real-world residential AC controlled via a smart thermostat, and simulated instances of ACs modeled after real-world data traces. In a 15-day experimental period, COHORT reduced daily peak loads by 12.5% on average based on load profiles from Pennsylvania-New Jersey-Maryland (PJM) Interconnection [175], while maintaining comfort in the real-world testbed.

4.2. Related Work

We review existing work on TCL control, and include work on other flexible loads if the methodology is relevant.

Architectures for TCL control. The primary challenge for jointly controlling a large number of TCLs is the large state-action space. To address this challenge, a popular approach is to develop an aggregate model for the population and control the population in a centralized manner. Examples of such aggregate model include the state bin transition model [130, 225] and the virtual battery model [103, 230]. However, these aggregate models depend on the assumptions that each system may be characterized by a 1^{st}- (or 2^{nd}- [225]) order linear model, and that all systems in the population share the same model structure and control scheme. These aggregate models have low fidelity and do not capture system-specific dynamics. Specifically, aggregate TCL modeling is ill-suited for predicting long-term responses—a pre-requisite for tasks with long planning horizons, such as load shifting [31]. Alternatively, one can jointly control building loads as a centralized MPC problem [209], but, while this approach allows for incorporation of detailed building models and system-specific constraints, it is computationally expensive and also raises privacy concers, as it requires each building to share an excessive amount of information with the load aggregator, including: thermal models, system specifications, control logic, and occupants' usage pattern and comfort preferences.

Aside from the centralized architecture, decentralized control [204] and distributed control [58, 150, 31] approaches have also been proposed in the literature. Taking advantage of the fact that system frequency is a universally-available indicator for supply-demand imbalance, [204] determines the action of each TCL with a power response model based on locally-available information. The key advantage of a decentralized control approach is that each system can be controlled based on local information, without any communication. However, this characteristic also constrains the applications of decentralized control to frequency or voltage regulation and real-time load shaping [31].

In a distributed architecture, which we adopt in this work, each system is responsible for its own control, and it coordinates with others to jointly achieve a grid-level objective. In [150], the distributed MPC scheme allocated the aggregate load to TCL clusters following a time-invariant weight. However, this allocation scheme does not account for the fact that the flexibility available at each building is time-varying, and thus does not fully utilize the aggregate flexibility [209]. [**?**] used a similar distributed MPC approach, but adaptively learned the allocation scheme with an evolutionary strategy. Most similar to this work is [31], which also used ADMM for distributed optimization. A major advantage is that the computation is distributed to and parallelized at each TCL. A significant limitation of [31] the computational cost grows exponentially with the planning horizon. Specifically, it represented each TCL as a set of feasible state-action trajectories, the size of which is N_a^T, where N_a is the number of alternative actions considered at each time-step, and T is the number of time-steps in the planning horizon. Given that the trajectories depends on initial state and future disturbances, the trajectories need to be unrolled at each time-step. Another limitation is that the solution is only optimal with respect to the set of trajectories under consideration.

Experimental Validation. The majority of works on this topic validated their approaches in simulation. In particular, works such as [130, 103, 204, 230, 150, 31] validated

their approach on population of TCLs simulated with 1^{st}-order linear thermal model, using model parameters sampled from assumed distributions. It is unclear how well such validation reflects performance on real-world systems. [218] demonstrated 1^{st}- and 2^{nd}-order models failed to accurately capture the thermodynamics of an individual electric water heater. Furthermore, there is a large difference between performance reported in simulation and in real-world testbeds. For instance, [103] reported a maximum absolute percentage error of less than 1% tracking a 4s frequency regulation signal in a simulation study. In comparison, in a real-world experiment on 300 residential ACs, [162] reported a median absolution percentage error of 6.7% executing 1-hour demand response (DR) events, which is arguably a much simpler task. Such discrepancy calls for more realistic evaluation. Other attempts at realistic evaluation include [56] which developed linear models based on configurations of real households, and [166], which used a co-simulation environment with EnergyPlus models.

Optimization Objectives. A myriad of grid-level objectives have been discussed in the literature, such as: energy cost minimization [56], DR events [225, 204, 166, 162], frequency regulation [103, 230], generation following [150, 31], reference tracking [130], and peak load reduction [58]. However, these works generally formulate their approaches based on their specific use case, without discussing their generalizability to other applications.

4.3. Preliminaries

We now present background technical concepts used by COHORT, including TCL modeling (Section 4.3.1) and ADMM (Section 4.3.2).

4.3.1. TCL Model and Flexibility

Here, we introduce the modeling of an individual TCL and define its flexibility. The contents is largely inspired by [230], from which we made modifications based on our problem.

System Dynamics. The temperature dynamics of an individual TCL is commonly modeled with Eq. 4.1a [130, 103, 230], where T_t is the TCL temperature, $T_{a,t}$ is the ambient temperature, and $m_t \in \{0, 1\}$ is the binary control variable representing the operating state, i.e., *on* or *off*, at time t. The negative sign associated with the control action assumes the TCL is operating in cooling mode, which could be changed to a positive sign to reflect heating. P_m is the rated power. Denoting the thermal resistance and capacitance of the TCL as R and C respectively, the model parameters can be calculated as: $a = \exp\{-\Delta T/(RC)\}$ and $b_t = \eta_t R$, where ΔT is the time-step and η_t is the time-varying coefficient of performance (COP). While the thermodynamics is linear, it is difficult to analyze the system dynamics in Eq. 4.1a due to the discrete control variable m_t. A common approach is to apply convex relaxation to the discrete control variable, which results in a linear system approximation (Eq. 4.1b) [130, 103,

230]. The new control variable $u_t \in [0, P_m]$, i.e., power consumption of the TCL, is continuous instead of discrete. This approach is justified as the aggregate behavior of a TCL population can be approximated accurately by Eq. 4.1b [230].

$$T_{t+1} = aT_t + (1-a)(T_{a,t} - b_t m_t P_m) \tag{4.1a}$$

$$T_{t+1} = aT_t + (1-a)(T_{a,t} - b_t u_t) \tag{4.1b}$$

The TCL dynamics over a planning horizon, $t : t+T-1$, is thus characterized by Eq. 4.2a (or more concisely Eq. 4.2b), where $\mathbf{B_u} = \text{diag}(-(1-a)b_t, \ldots, -(1-a)b_{t+T-1})$. Throughout this work, boldface lower-case letters, e.g., $\mathbf{x}$, are vectors, and boldface upper-case letters, e.g., $\mathbf{A}$, are matrices; $\mathbf{x}$, $\mathbf{x}_0$, $\mathbf{u} \in \mathbb{R}^T$ and $\mathbf{A}$, $\mathbf{B_u} \in \mathbb{R}^{T \times T}$. Denoting the number of disturbance terms as l, we have $\mathbf{D} \in \mathbb{R}^{T \times l}$ and $\mathbf{b_d} \in \mathbb{R}^l$. In this case, the disturbance term only includes the ambient temperature, $l = 1$.

$$\underbrace{\begin{bmatrix} 1 & & & \\ -a & 1 & & \\ & \ddots & \ddots & \\ & & -a & 1 \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} T_{t+1} \\ T_{t+2} \\ \vdots \\ T_{t+T} \end{bmatrix}}_{\mathbf{x}} = \underbrace{\begin{bmatrix} aT_t \\ 0 \\ \vdots \\ 0 \end{bmatrix}}_{\mathbf{x}_0} + \mathbf{B_u} \underbrace{\begin{bmatrix} u_t \\ u_{t+1} \\ \vdots \\ u_{t+T-1} \end{bmatrix}}_{\mathbf{u}} + \underbrace{\begin{bmatrix} T_{a,t} \\ T_{a,t+1} \\ \vdots \\ T_{a,t+T-1} \end{bmatrix}}_{\mathbf{D}} \underbrace{[1-a]}_{\mathbf{b_d}} \tag{4.2a}$$

$$\mathbf{Ax} = \mathbf{x}_0 + \mathbf{B_u u} + \mathbf{D b_d} \tag{4.2b}$$

Constraints. Each TCL needs to satisfy the end-use requirements and respect the operational constraints. In this case, we require the TCL temperature to be within the deadband, i.e., $T_t \in [T_{sp} - \Delta, T_{sp} + \Delta]$, where T_{sp} is the setpoint and Δ is half of the deadband. At the same time, the system needs to be operating with in its power limits, i.e., $P_t \in [0, P_m]$. Combining the system dynamics given in Eq. 4.2b, the aforementioned constraints can be written as Eq. 4.3, where $\underline{\mathbf{u}} = [0]$, $\overline{\mathbf{u}} = [Pm]$, $\underline{\mathbf{x}} = [T_{sp} - \Delta]$, and $\overline{\mathbf{x}} = [T_{sp} + \Delta]$.

$$\underline{\mathbf{u}} \leq \mathbf{u} \leq \overline{\mathbf{u}}; \quad \underline{\mathbf{x}} \leq \mathbf{A}^{-1}(\mathbf{x}_0 + \mathbf{B_u u} + \mathbf{D b_d}) \leq \overline{\mathbf{x}}; \tag{4.3}$$

Flexibility. We adopt the definition of flexibility proposed in [230], where the flexibility of a system is the set of all admissible power profiles. Our formulation of Eq. 4.4 is generalized from that in [230], to incorporate non-linear systems. $\mathcal{P}$ denotes the flexibility of the system, $\mathcal{T}(x_k, u_k)$ denotes the state transition function, and $\underline{x}_k$, $\overline{x}_k$, $\underline{u}_k$, and $\overline{u}_k$ are the lower and upper bounds for state and action of the given system at time k. An important intuition is that the flexibility is coupled over time through the thermodynamics [230]:

$$\mathcal{P} = \left\{ [u_{t:t+T-1}] \;\middle|\; \begin{array}{l} x_{k+1} = \mathcal{T}(x_k, u_k); \\ \underline{u}_k \leq u_k \leq \overline{u}_k; \;\; \forall k \in \{t, \ldots, t+T-1\} \\ \underline{x}_{k+1} \leq x_{k+1} \leq \overline{x}_{k+1}; \end{array} \right\} \tag{4.4}$$

For the specific case of TCL, which follows the linear dynamics in Eq. 4.1b, the

flexibility $\mathcal{P}$ of a TCL can be expressed as Eq. 4.5.

$$\mathcal{P} = \mathcal{U} \cap \mathcal{X} \tag{4.5}$$

where $\mathcal{U} = \{\mathbf{u} | \underline{\mathbf{u}} \leq \mathbf{u} \leq \overline{\mathbf{u}}\}$ and $\mathcal{X} = \{\mathbf{x} | \underline{\mathbf{x}} \leq \mathbf{A^{-1}}(\mathbf{x_0} + \mathbf{B_u u} + \mathbf{D b_d}) \leq \overline{\mathbf{x}}\}$, as derived in Eq. 4.3. Note that $\mathcal{P}$ boils down to a set of linear inequalities, which is geometrically interpreted as a polytope[13] [230].

4.3.2. ADMM

ADMM is a well-established distributed convex optimization algorithm, which decomposes a large problem into smaller subproblems and coordinates the solutions to find a global optimum [29]. Generally, ADMM solves problems in the form of Eq. 4.6.

$$\begin{aligned} \min_{u,v} \quad & f(u) + g(v) \\ \text{s.t.} \quad & Au + Bv = c \end{aligned} \tag{4.6}$$

Specifically, we introduce the application of ADMM to a canonical problem: the *sharing problem*, as given in Eq. 4.7, where f_i is a local objective for agent i, and g is the global objective—defined as a function of the aggregate of all decision variables from the agents:

$$\min_{u_i} \quad \sum_{i}^{N} f_i(u_i) + g(\sum_{i}^{N} u_i) \tag{4.7}$$

By introducing a copy of the decision variable u_i as v_i, the sharing problem can be written in a ADMM-compatible form (Eq. 4.8):

$$\begin{aligned} \min_{u,v} \quad & \sum_{i}^{N} f_i(u_i) + g(\sum_{i}^{N} v_i) \\ \text{s.t.} \quad & u_i - v_i = 0, \quad i = 1, \ldots, N \end{aligned} \tag{4.8}$$

The update rules for solving the problem are given in Eq. 4.9, where w is the dual variable, ρ is a hyperparameter, and the superscript (k) denotes the value of a variable at the k^{th} iteration. We elaborate on the intuition behind these update rules here. The sharing problem can be interpreted as the agents coordinating their decisions so as to strike a balance between the local and the global objectives. Hence, u_i and v_i are each agent's solutions to its local problem and the global objective, respectively. The dual variable w_i, as calculated in Eq. 4.9c, is the cumulative disagreement between u_i and v_i. Thus, w_i, which we also call the incentive variable, communicates how to adjust each agent's solutions such that they would agree, i.e., $u_i = v_i$. Thus, in the *u-update* step (Eq. 4.9a), each agent solves its local problem, while mindful of its solution to the global problem. Similarly, in the *v-update* step (Eq. 4.9b), the agents jointly optimize the global objective, while ensuring their decisions are close to those of their local problems.

[13] A polytope can be characterized as a set $\mathcal{S} = \{x \in \mathbb{R}^n | Ax \leq b\}$.

$$u_i^{(k+1)} = \arg\min_{u_i} f_i(u_i) + \frac{\rho}{2}\|u_i - v_i^{(k)} + w_i^{(k)}\|_2^2 \tag{4.9a}$$

$$v_i^{(k+1)} = \arg\min_{v_i} g(\sum_i^N v_i) + \frac{N\rho}{2}\|u_i^{(k+1)} - v_i + w_i{}^{(k)}\|_2^2 \tag{4.9b}$$

$$w_i^{(k+1)} = w_i{}^{(k)} + u_i^{(k+1)} - v_i^{(k+1)} \tag{4.9c}$$

A shortcoming of the updates rules given in Eq. 4.9 is that it requires a copy of the decision variable for each agent, i.e., v_i. Intuitively, the global objective only depends on the aggregate behavior of the population, and thus a more efficient algorithm (Eq. 4.10) is possible using the mean of the variables, denoted as $\bar{u}$, $\bar{v}$, and $\bar{w}$ respectively.

$$u_i^{(k+1)} = \arg\min_{u_i} f_i(u_i) + \frac{\rho}{2}\|u_i - u_i^{(k)} + \bar{u}^{(k)} - \bar{v}^{(k)} + \bar{w}^{(k)}\|_2^2 \tag{4.10a}$$

$$\bar{v}^{(k+1)} = \arg\min_{\bar{v}} g(N\bar{v}) + \frac{N\rho}{2}\|\bar{u}^{(k+1)} - \bar{v} + \bar{w}^{(k)}\|_2^2 \tag{4.10b}$$

$$\bar{w}^{(k+1)} = \bar{w}^{(k)} + \bar{u}^{(k+1)} - \bar{v}^{(k+1)} \tag{4.10c}$$

An equality such as $v_i^{(k)} = \bar{v}^{(k)} + (u_i^{(k)} + w_i^{(k)}) - (\bar{u}^{(k)} + \bar{w}^{(k)})$, to show Eq. 4.9 and Eq. 4.10 are equivalent, can be derived from the stationarity condition [31]. For more details on the algorithm, we refer interested readers to [29].

4.4. Approach

We first formulate the problem and elaborate on the optimization procedure, with focus on the coordination between the load aggregator and the TCLs (Section 4.4.1). We then describe a PWM-based strategy for TCL-level control (Section 4.4.2).

4.4.1. Problem Formulation and Optimization

The problem we address can be formulated as Eq. 4.11: we want to simultaneously optimize a grid-level objective $g(\cdot)$, which is a function of aggregate power consumption, and make sure the actions of each TCL are admissible based on operational constraints and end-use requirements, characterized by the set $\mathcal{P}_i$. By applying convex relaxation to discrete actions, and characterizing the flexibility as a convex set $\mathcal{P}_i$, our approach is computationally viable for tasks with long planning horizons. Recall the definition in Section 4.3.1, $\mathbf{u_i} \in \mathbb{R}^T$ is the power consumption of a TCL over a planning horizon. The subscript i denotes the i$^{\text{th}}$ TCL. While we did not include any TCL-level objective

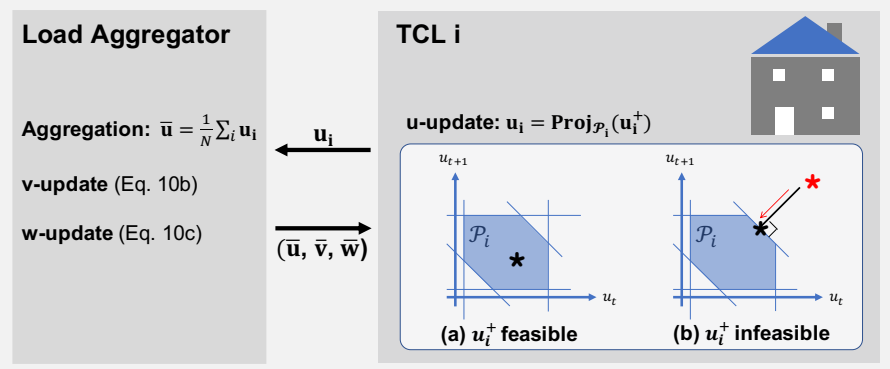


Figure 4.2: Coordination. The *u-update* step is distributed to and computed in parallel at each TCL as a projection operation. The load aggregator calculates $\bar{\mathbf{u}}$, and then updates $\bar{\mathbf{v}}$ and $\bar{\mathbf{w}}$. Finally, the load aggregator broadcasts the $\bar{\mathbf{u}}$, $\bar{\mathbf{v}}$, and $\bar{\mathbf{w}}$ to the population. This procedure repeats until convergence.

other than satisfying its constraints, it is possible to incorporate such objectives.

$$\begin{aligned} \min_{\mathbf{u_i}} \quad & g\Big(\sum_i^N \mathbf{u_i}\Big) \\ \text{s.t.} \quad & \mathbf{u_i} \in \mathcal{P}_i, \ \forall i \end{aligned} \tag{4.11}$$

The problem in Eq. 4.11 can be written in a ADMM-compatible form (Eq. 4.12) by (i) introducing a copy of the variable $\mathbf{u_i}$ as $\mathbf{v_i}$, and (ii) representing the constraints set $\mathcal{P}_i$ with the indicator function $\mathbb{I}_{\mathcal{P}_i}$. By definition, $\mathbb{I}_{\mathcal{P}_i}(\mathbf{u_i}) = 0$, if $\mathbf{u_i} \in \mathcal{P}_i$, else $\mathbb{I}_{\mathcal{P}_i}(\mathbf{u_i}) = \infty$ [202]. The large penalty for an inadmissible $\mathbf{u_i}$ forces the solver to find $\mathbf{u_i}$ that satisfies the constraints.

$$\begin{aligned} \min_{\mathbf{u_i}, \mathbf{v_i}} \quad & \sum_i \mathbb{I}_{\mathcal{P}^i}(\mathbf{u_i}) + g\Big(\sum_i \mathbf{v_i}\Big) \\ \text{s.t.} \quad & \mathbf{u_i} = \mathbf{v_i} \end{aligned} \tag{4.12}$$

Note that Eq. 4.12 now has the same form as Eq. 4.8, and thus may be solved with the update rules in Eq. 4.10. Figure 4.2 summarizes the coordination procedure between the load aggregator and each TCL. Firstly, each TCL updates its action, $\mathbf{u_i}$, locally. Given that $f_i = \mathbb{I}_{\mathcal{P}_i}$ and $\mathcal{P}_i$ is a polytope for a TCL, the *u-update* step (Eq. 4.10a) may be implemented efficiently as a projection operation [202], also illustrated in Figure 4.2. To simplify notation, we denote $\mathbf{u_i}^+ = \mathbf{v_i}^{(k)} - \mathbf{w_i}^{(k)} = \mathbf{u_i}^{(k)} - \bar{\mathbf{u}}^{(k)} + \bar{\mathbf{v}}^{(k)} - \bar{\mathbf{w}}^{(k)}$; $\mathbf{u_i}^+$ can be interpreted as the desired power profile for TCL i at the end of the k$^{\text{th}}$ iteration, and the projection $\mathbf{Proj}_{\mathcal{P}_i}(\mathbf{u_i}^+)$ ensures that the coordinated power profile is admissible for the TCL. Secondly, the load aggregator collects the actions from all agents to find the mean, $\bar{\mathbf{u}}$, and sequentially updates $\bar{\mathbf{v}}$ following Eq. 4.10b, and $\bar{\mathbf{w}}$ following Eq. 4.10c. Finally, the load aggregator broadcasts $\bar{\mathbf{u}}$, $\bar{\mathbf{v}}$, and $\bar{\mathbf{w}}$ to all TCLs, such that they can

update $\mathbf{u_i}$ locally. This procedure repeats until convergence.

While we use ADMM in this work, we note that alternative distributed optimization algorithms exist in the literature, such as consensus-based approaches [**?**] that are especially relevant for distributed energy management. However, for the application in this paper, ADMM is suitable given the communication topology[14] and provides scalability given the ease of solving each TCL's local problem. There are three key advantages to the ADMM-based approach for this application. Firstly, the ADMM naturally decomposes the grid-level problem into subproblems. Thus, each TCL can ensure its local objective and constraints are satisfied, without sharing them with the load aggregator, thereby preserving privacy. Secondly, the *u-update* at each TCL is computed in parallel. Thus, the approach is highly scalable to large population. Finally, ADMM is guaranteed to converge to the grid-level optimum given a convex objective [29]. As will become clear through our demonstrations, a variety of grid objectives can be formulated as convex problems.

4.4.2. TCL-level Control

Recall that we applied convex relaxation to TCL dynamics, i.e., $m_t P_m \in \{0, P_m\}$ (Eq. 4.1a) to $u_t \in [0, P_m]$ (Eq. 4.1b), such that the grid-level objective can be optimized efficiently over long time horizons. In this section, we describe how to translate the continuous power trajectory back to on/off actuation with PWM, a method for generating quasi-continuous output from an on/off actuator. In [32], it was demonstrated that a TCL could be treated as a variable power unit via low-frequency PWM. The action normalized by rated power, $u_t/P_m \in [0, 1]$, can be interpreted as duty cycle ratio, i.e., the portion of time the TCL is *on* within the control time-step. Specifically, we implemented PWM with Sigma-Delta (Σ-Δ) modulation [170], where TCL switches between *on* and *off* when the cumulative error, $\epsilon_t = \sum_{k=0}^{t}(u_k/P_m - m_k)\Delta T$, exceeds the limits. Figure 4.3 illustrates its use in tracking a sine wave. It is clear from Figure 4.3 that fewer switchings are needed when a signal is close to either 0 or 1. We also observe that the solution tends to be close to a feasible initialization. The intuition may be that an individual TCL does not need to drastically change its default behavior to collaboratively achieve a grid-level objective. Given these observations, we initialize each $\mathbf{u_i}$ with a sequence of interlaced 0s and P_ms to encourage sparsity in the solution. Furthermore, by placing 0s and P_ms with care in the initialization, short-cycling can be reduced.

4.5. Experiment 1: Simulation Study

In this section, we evaluated COHORT in simulation as an initial proof of concept. Following [130, 103, 204, 230, 150, 31], we validated our approach on a population of TCLs simulated with 1^{st}-order linear thermal models (Eq. 4.1a). We simulated 1000 TCLs, using parameters sampled from uniform distributions around nominal values [130, 103, 204, 230]. Specifically, we followed the same parameter distributions and values

[14] i.e. there being a load aggregator that centrally collects and broadcasts information. In comparison, consensus-based methods would be more suitable for peer-to-peer communication topology.

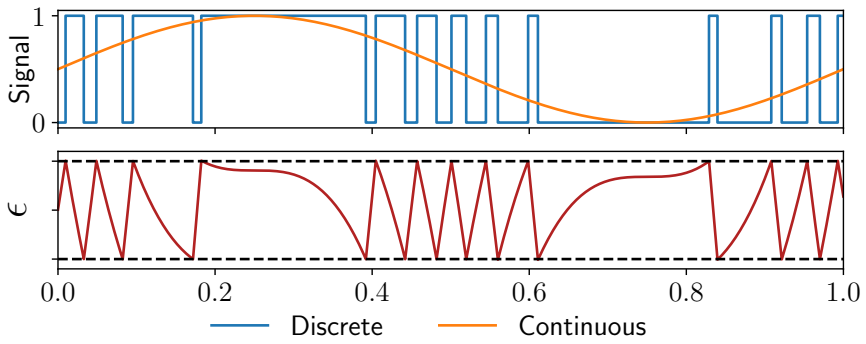


Figure 4.3: Sigma-Delta modulation converts a continuous signal to a discrete signal by switching states when the cumulative error, ϵ, exceeds the limits (dashed black lines).

for temperature setpoint and exogenous variable as [103]. We used a deadband of Δ= 1°C throughout this work. While these assumptions may not reflect realistic system dynamics and population heterogeneity, we adopted them such that the performance of our approach is directly comparable to those reported in the literature. We lifted these assumption and validated COHORT in a real-world testbed in Section 5.6.

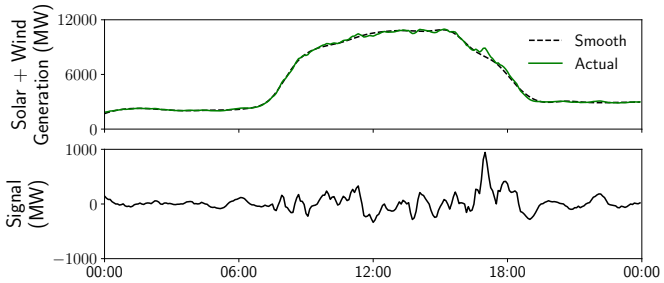
We applied COHORT to address challenges arising from increasing penetration of renewable generation. Firstly, we used the inherent flexibility in the TCL population to absorb the variations in renewable generation in a generation following use case (Section 4.5.1). Secondly, we shifted the TCL load to alleviate the need to quickly ramp up / down energy generation in areas of high renewable penetration (Section 4.5.2). The load curves used for both use cases (Figure 4.4) are from CAISO [115] on 31 March 2020.

4.5.1. Use Case 1: Generation Following

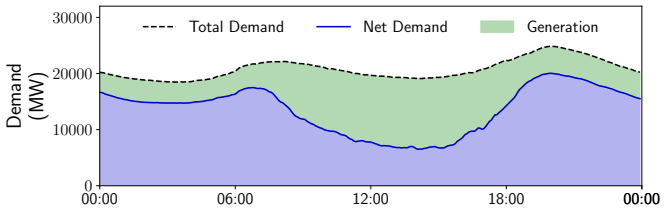
The generation following signal was produced following the same procedure in [31], as shown in Figure 4.4a. The TCL population tracked a scaled version of the generation following signal around its baseline power consumption. The objective function is the mean squared error (MSE) between the reference signal, denoted by $\tilde{\mathbf{u}}$, and actual aggregate energy consumption (Eq. 4.13). Other tasks such as DR events [225, 204] and frequency regulation [103, 230] boil down to tracking a given reference signal, and thus may be addressed with the same objective function.

All the optimization problems in this work[15] were solved using `CVXPY` [**?**] with hyperparameter $\rho = 10$. In this use case, we used a 5-min control time-step, and planned for the next time-step, i.e., T = 1. For tracking with PWM, it is necessary to use a smaller time-step. Throughout this work, we used a tracking time-step that is 1/15 of the control time-step. The error limit used for Σ-Δ modulation is 0.1kWh in the simulation study.

[15]The code is available at https://github.com/INFERLab/COHORT.



(a) Use Case 1: The 5-min generation following signal is produced by taking the difference between the actual renewable generation and its smooth spline fit, following [31].



(b) Use Case 2: The duck curve (i.e. the net demand) exemplifies the need for generators to quickly ramp up energy production when the sun sets in areas of high renewable penetration [118].

Figure 4.4: Load Profiles from CAISO, 2020/03/31

$$g_{\text{track}}\Big(\sum_i \mathbf{u_i}\Big) = \frac{1}{T}||\tilde{\mathbf{u}} - \sum_i \mathbf{u_i}||_2^2 \tag{4.13}$$

The behavior of the TCL population is shown in Figure 4.5. COHORT tracked the reference signal with a small error, while maintaining the temperature of the population within the deadband (dashed green line). Note that the discrepancy between the reference signal and actual power consumption came solely from discretization error. The performance of our approach with comparison to seminal works on TCL control is summarized in Table 4.1. Our tracking performance is comparable to that in [130], reported in normalized[16] root MSE (RMSE), and is not as good as [103], reported in mean absolute percentage error (MAPE). While [103] may be more suited for reference tracking tasks, it is not applicable to any planning-based task, e.g. load shifting. Compared to the baseline scenario where the TCL population only maintains temperature, our approach increased the switching frequency by 158.7%, which is similar to [103, 130]. By adjusting the error limits in Σ-Δ modulation, one can trade-off tracking performance and switching frequency.

Figure 4.6 show actions of individual TCLs. We initialized each action with either 0 or P_m based on its previous action, and switched if the temperature was close to the edge of the deadband. Given this initialization scheme, the majority of continuous actions

[16] by average aggregate power

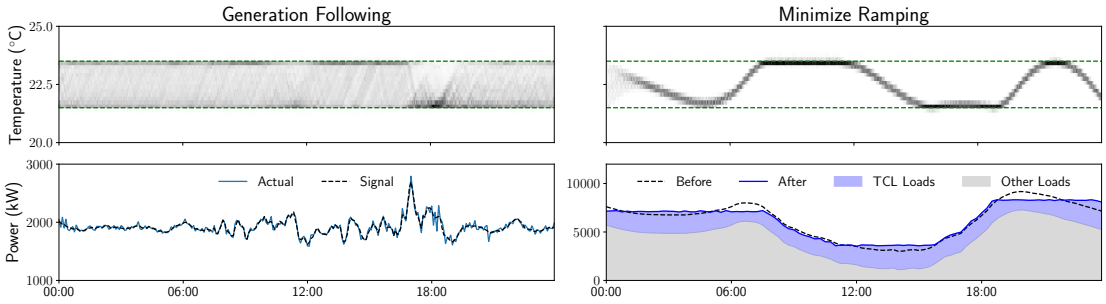


Figure 4.5: Simulation Study. (Top) the temperature distribution and (Bottom) the aggregate power of the population

u_t are close to either 0 or P_m, making conversion to on/off actuation possible with a reasonable number of switchings.

In this use case, our approach took an average of 5.4 iterations to reach consensus. Interestingly, the number of iterations till convergence is almost independent of the population size, which is also observed in [88]. Since the *u-update* step is distributed to and computed in parallel at each TCL, the overall computation time and the computation cost at the load aggregator scale very well with the population size.

Table 4.1: Performance Comparison for Reference Tracking

	Tracking		**Switching**
	Norm. RMSE	**MAPE**	**Increase**
	(%)	(%)	(%)
[130]	0.8-2.27	-	170-300
[103]	-	< 1	116.7
COHORT	2.04	1.53	158.7

4.5.2. Use Case 2: Minimize Ramping

While we first evaluated our approach in a reference tracking use case, a major advantage of COHORT compared with existing methods is its ability to coordinate TCLs over long time horizons. Thus, we applied our approach to flatten the *duck curve* by shifting load over a day. Specifically, we used a 15-min control time-step, and plan for an entire day ahead, i.e., T= 96. An example of the *duck curve*, named after its resemblance to a duck [118], is given in Figure 4.4b. We scaled down the load curve such that the TCL demand accounts for 20% of the total demand [103]. We formulate the objective as minimizing total ramping, i.e., the difference in net demand between consecutive time-steps (Eq. 4.14). $\mathbf{P}_{\text{total}}$, $\mathbf{P}_{\text{net}}$, and $\mathbf{P}_{\text{gen}}$ are the total demand, net demand, and renewable generation, respectively. This problem is also known as total variation minimization [57]. While this objective is trickier to optimize, convergence to global optimum is still

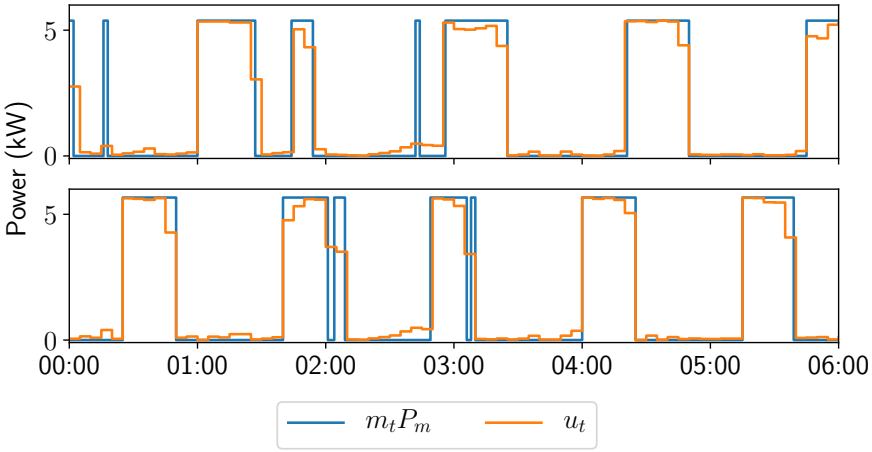


Figure 4.6: Actions of Individual TCLs

guaranteed, because g_{tv} is a convex function.

$$\begin{aligned} g_{\text{tv}}\big(\sum_i \mathbf{u_i}\big) &= \sum_{k=t+1}^{t+T} |P_{\text{net,k}} - P_{\text{net,k-1}}| \\ \text{where,} \quad \mathbf{P}_{\text{net}} &= \mathbf{P}_{\text{total}} - \mathbf{P}_{\text{gen}} \\ \mathbf{P}_{\text{total}} &= \mathbf{P}_{\text{non-shiftable}} + \sum_i \mathbf{u_i} \end{aligned} \tag{4.14}$$

The behavior of the TCL population for this use case is also shown in Figure 3. The TCLs systematically shifted their load and reduced ramping by 23.1% compared to the baseline scenario, where the TCL population was operated by on-off control. The temperature of the TCLs shifted accordingly within the deadband. Since the TCLs are operating in cooling mode, reduced energy consumption results in higher temperature, and vice versa. Note that the 23.1% reduction in ramping is based on the assumption that 20% of total demand is flexible. Further reduction is possible by extending our approach to other flexible loads.

4.6. Experiment 2: Hardware-in-the-loop Simulation

In this section, we validated COHORT in a HIL simulation, with primary focus on its performance on a real-world testbed. Specifically, we controlled a residential AC via a smart thermostat. More details on the real-world testbed is presented in Section 4.6.1. We augmented the testbed with simulated instances of residential ACs, modeled after real-world data traces (Section 4.6.2). We integrated the real-world system and the simulated instances as a HIL simulation, and showcased it in a peak load curtailment use case, based on load profiles from PJM [175], as elaborated in Section 4.6.3. The HIL simulation was executed from 11-25 July 2020, a 15-day period, and the results are

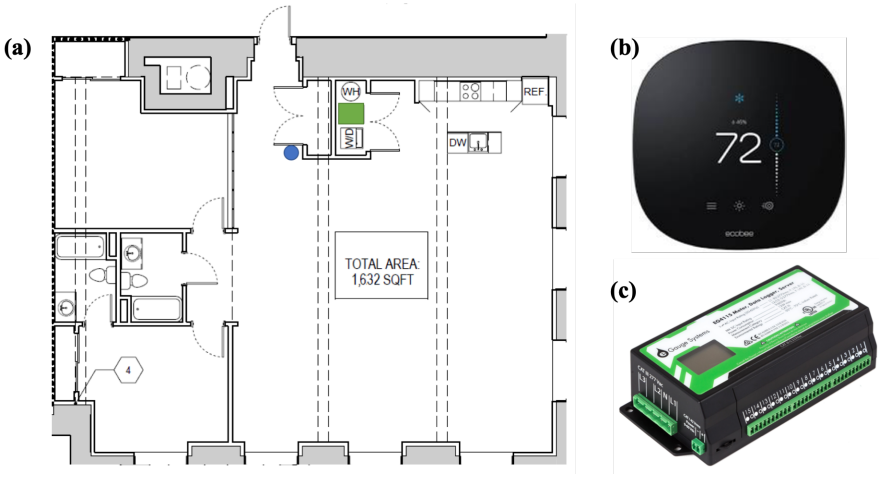


Figure 4.7: The floor plan of the apartment is shown in (a), with the circle and the rectangle marking the location of the smart thermostat and the indoor AC unit. The apartment is instrumented with (b) an ecobee smart thermostat and (c) an eGauge energy metering unit.

summarized in Section 4.6.4.

4.6.1. Real-World Testbed

The testbed is first author's apartment located in Pittsburgh, PA, USA (Figure 4.7). The 1632 ft^2 apartment has three regular occupants and was occupied most of the time during the experimental period. The AC unit is controlled via an `ecobee` smart thermostat, installed at a location shaded from direct solar radiation, and energy consumption of the AC is monitored for verification only via `eGauge` energy metering [74], with sampling rate up to 1Hz. Power measurements are not needed for control.

We monitored zone temperate and sent commands to the smart thermostat via ecobee API [73]. To get the effect of on/off commands, we sent a low temperature setpoint (70 °F) when we want the AC to be *on* and a high temperature setpoint (80 °F) when we want the AC to be *off*. This simple strategy worked surprisingly well. Figure 4.8 shows a comparison of the commands vs. the power draw on the AC circuit (at 1Hz) for an on/off event. The response of the AC to *on* is almost instantaneous, and the response to *off* is delayed by a few seconds, but negligible compared to the control time-step. Such response was observed throughout the experiment. The ease of integration with a smart thermostat implies COHORT could be scaled to 11% of households in the US already equipped with smart thermostats [127], with minimal effort and no retrofit.

4.6.2. Modeling of Residential ACs

To have a population of simulated ACs with realistic thermodynamics and population heterogeneity, we took advantage of ecobee's *Donate Your Data* dataset [72]. For a

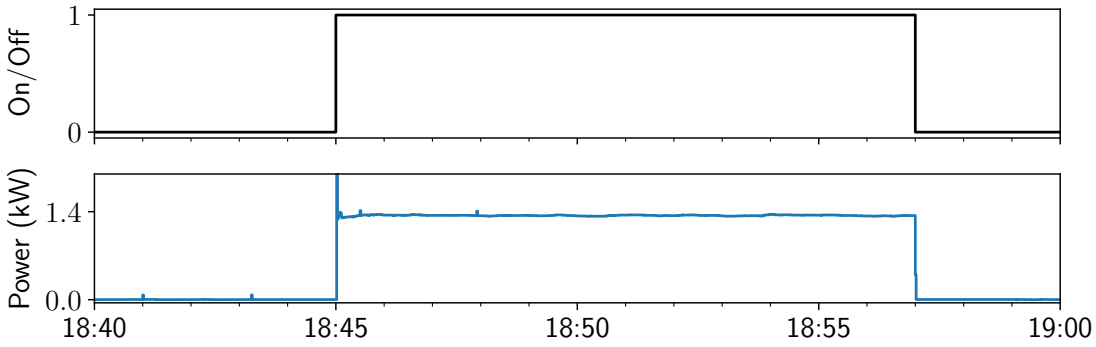


Figure 4.8: Comparison of the on/off commands vs. the actual power draw by the AC measured at 1Hz

comprehensive description of the dataset, we refer interested readers to [111]. We selected households in the same municipal area as the real-world testbed using data from 2019. We only used households with single-stage cooling, no less than 60 cooling days[17], and less than 10% missing data. These criteria left us with 106 households. While the simulated population size is relatively small, the primary focus of the HIL simulation is on the real-world testbed.

The raw data came in 5-min intervals and we down-sampled it to 15-min based on the control time-step. We used the same model form as [123] (Eq. 4.15), where T_t is the control temperature, u_t is the duty cycle ratio, $T_{a,t}$ is the outdoor ambient temperature, and $d_{o,t}$ and $d_{s,t}$ are binary flags for occupancy sensor activation and scheduled sleep time. These variables would be explained shortly. Both the model orders, i.e., p and q, and the disturbance terms were selected based on the Akaike Information Criterion (AIC). The median of selected model orders are $p = 5$ and $q = 2$.

$$T_{t+1} = \sum_{i=0}^{p-1} a_i T_{t-i} + \sum_{i=0}^{q-1} b_{u,i} u_{t-i} + b_a T_{a,t} + b_o d_{o,t} + b_s d_{s,t} \tag{4.15}$$

The state variable, the *control temperature*, is what an ecobee uses for operating the AC with respect to the setpoint. It is a weighted average of the temperature measurements at the main thermostat and remote sensors [111]. Note that the control temperature in the dataset came in 1F resolution. The control action, *duty cycle ratio*, is the equipment run-time normalized between 0 and 1. Similar to [111], we assumed that the house was occupied if any of the motion sensors were triggered or during scheduled sleep time. Since the raw data from occupancy sensors were sharp spikes, we passed the data through a low-pass filter. We used a separate variable for scheduled sleep time. Despite having different form from Eq. 4.1b, Eq. 4.15 can also be written in the form of Eq. 4.2b as a linear system, and thus the same formulation of flexibility (Eq. 4.5) still applies.

We evaluated the modeling performance with mean absolute error (MAE) over 1 to 6

[17]Cooling days are defined as days the AC is operating exclusively in cooling mode, following [111].

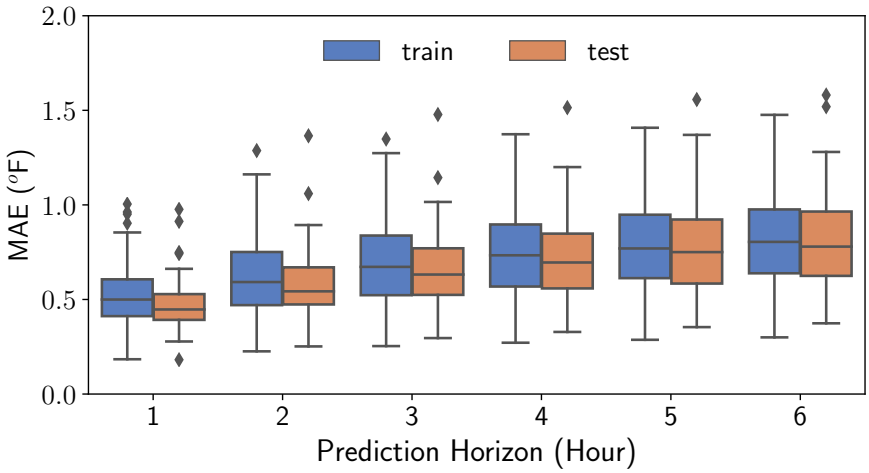


Figure 4.9: Distribution of model prediction error for 1 to 6 hour prediction horizons over 106 households

hour prediction horizons. Figure 4.9 shows the distribution of MAE over 106 households. The train set and the test set were the first 2/3 and last 1/3 of cooling days, respectively. The weather was checked to have a similar distribution over the train-test split. The majority of prediction error is less than 1F even at a 6-hour prediction horizon. This result is comparable to that of [123]. Bear in mind, in interpreting the results, that the control temperature came in a low resolution of 1F.

4.6.3. Experimental Setup

Growing peak demand decreases the average utilization of generators [75] and increases the need to build and operate high marginal cost peaking generation [35]. Thus, we showcased our approach on a peak load curtailment use case, the objective of which is given in Eq. 4.16. As an infinity norm[18] minimization problem, the solver minimizes the maximum total load within the planning horizon. We used a 15-min control time-step and found a 16-hour planning horizon to be sufficient. We re-planned at each hour to avoid compounding modeling error. Similar to the experiment in Section 4.5.2, we scaled down the PJM load profile with the assumption that TCL loads account for about 20% of the total load.

$$\begin{aligned} g_{\text{peak}}\big(\sum_i \mathbf{u_i}\big) &= ||\mathbf{P}_{\text{total}}||_\infty \\ \text{where,} \qquad \mathbf{P}_{\text{total}} &= \mathbf{P}_{\text{non-shiftable}} + \sum_i \mathbf{u_i} \end{aligned} \tag{4.16}$$

We integrated the real-world testbed with simulated instances of residential ACs to form an HIL simulation. The HIL simulation is also integrated with day-ahead weather

[18]Infinity norm is defined as $||\mathbf{x}||_\infty := \max(|x_i|)$.

forecast via Dark Sky API [193] and total load profile from PJM Data Miner 2 [175].

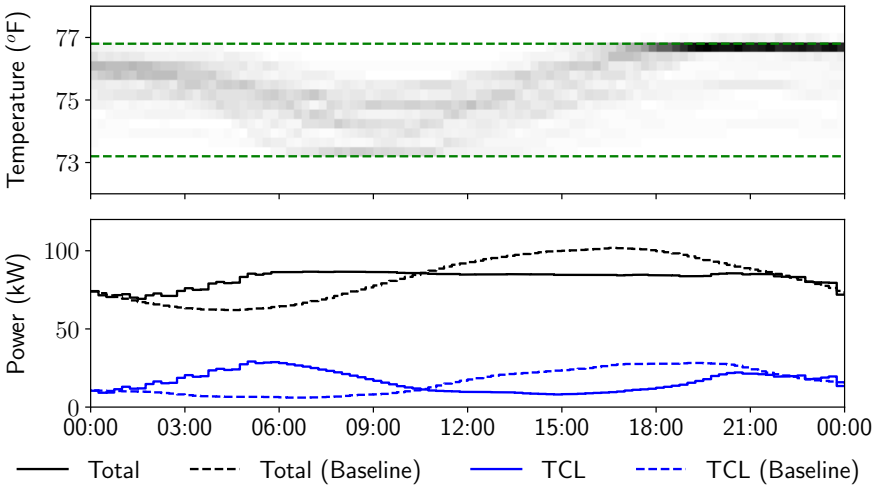
We modeled the real-world testbed using the same procedures as described in Section 4.6.2. Note that the temperature time series pulled from ecobee API comes in 0.1F resolution, and thus allows for more accurate modeling. To identify the system accurately, the testbed was excited by square wave signals, one-hour on followed by one-hour off, on 10 July 2020. During the experiment, the model was updated regularly based on new data. We used P_m=1.4kW for the real-world system based on actual power measurements. As discussed in Section 4.4.2, we initialize $\mathbf{u_i}$ with sequences of interlaced 0s and P_ms to encourage sparsity in the solution. For the AC unit, we initialized the solution for each hour with [P_m, 0, 0, 0] and used an error limit of 0.075kWh.

For the simulated instances, we selected models that performed no worse than 75$^{\text{th}}$ percentile on any of the prediction horizon, which left us with 72 households. The dataset does not contain information on the rated power of the AC units, and thus we sized the AC based on the floor area, following Energy Star recommendations [**?**]. We used the same occupancy data from the same time last year. The temperature setpoint for the entire population, including the real-world testbed, was assigned to be 75F throughout the experiment.

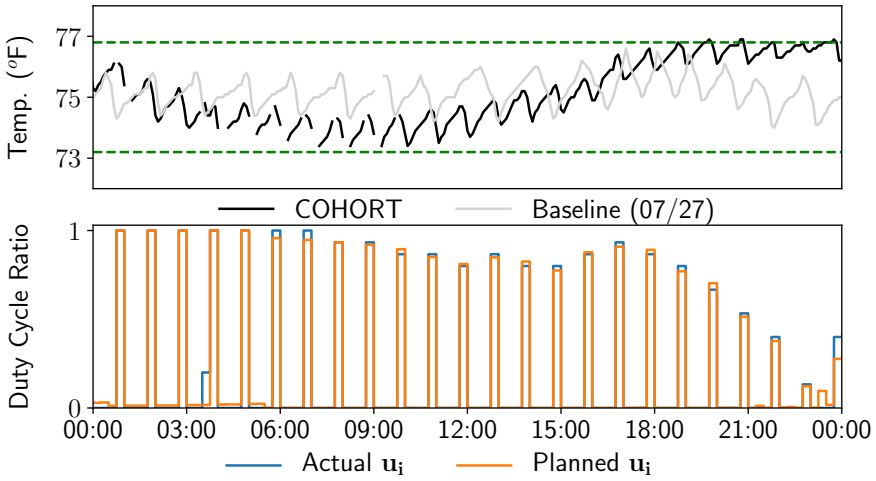
4.6.4. Results

The performance of COHORT compared with the baseline scenario, where the TCL population was operated by on-off control, is exemplified in Figure 4.10, using time series from 24 July 2020. The baseline TCL load was simulated using a model of the real-world testbed, along with the rest of the population, based on the same weather data. The baseline total load was a scaled down version of the actual PJM load profile, under the assumption that TCLs account for about 20% of the total load. At the aggregate level (Figure 4.10a), the TCL peak approximately coincides with the utility peak in the baseline scenario (dashed lines). COHORT systematically shifted the TCL load away from peak hours to early morning, thereby reducing the utility peak by 15%. The temperature of the population shifted accordingly. By pre-cooling the households when the demand was low, the TCLs could reduce energy consumption during the peak hours and let temperature slowly float up, without violating comfort constraints. The same behavior is observed in the real-world testbed (Figure 4.10b).

Also in the real-world testbed, the planned vs. actual duty cycle ratio at each 15-min control time-step shows good overall agreement. By initializing the actions at each hour with [P_m, 0, 0, 0], the *on* events were spaced apart, thereby reducing the risk of short cycling. The temperature, as logged by ecobee at 5-min intervals, was maintained within the deadband. To compare with the baseline scenario, we also include in the same plot the temperature time-series in the real-world testbed on 27 July 2020, when the ecobee maintained temperature at 75F setpoint using its default strategy. The slightly wider temperature range and the gradual temperature shift were barely perceptible by the occupants. All occupants were happy with their thermal comfort during the experiment. In single blind tests, occupants not involved in the experiment were not able to tell if the AC was operated by COHORT or on-off control.



(a) (Top) temperature distribution and (Bottom) aggregate power



(b) (Top) temperature and (Bottom) duty cycle ratio

Figure 4.10: HIL Simulation. Behaviour of (a) the population and (b) the real-world testbed on 2020/07/24

Over the 15-day experimental period, COHORT reduced daily peak loads by an average of 12.5% (with a 2.9% standard deviation), when the TCL accounts for about 20% of the total load. In comparison, the peak load was reduced by 8.8% in the best-performing case in [56], when the TCL peak account for 74% of the utility peak in the baseline scenario. While the experimental setups are different, the difference in performance likely comes from the fact that a mere 1-hour planning horizon was used in [56]. Instead, we planned ahead for 16-hour and managed to level the total load from about 6:00-21:00 in the case shown in Figure 4.10a. This affirms the advantage of being able to plan over long time horizons.

4.7. Conclusions

We proposed COHORT, a novel distributed control solution for coordinating TCLs to jointly optimize a grid-level objective, while satisfying each TCL's end-use requirements and operational constraints. Our approach decomposes the grid-level problem into subproblems and coordinates their solutions to find the grid-level optimum. To be computationally viable over long planning horizons, we apply convex relaxation to the discrete action space and characterize each TCL's flexibility as a convex set. After coordination, each TCL tracks the agreed-upon power trajectory locally with a PWM-based strategy, which translates the continuous power back to on/off actuation. Since each TCL is responsible for its own control, it can incorporate detailed and system-specific dynamics and constraints, which is difficult to accomplish in a centralized architecture. Furthermore, the coordination process is independent of each TCL's dynamics and control scheme, making COHORT extensible to other flexible loads.

We validated COHORT in simulation studies and a HIL simulation to address challenges arising from growing peak demand and increasing penetration of renewable generation. In the generation following use case, our approach showed comparable performance to prior work. A major advantage of our approach compared with existing work is its ability to coordinate TCLs over long planning horizons. Thus, we applied it to load shifting use cases, assuming TCLs account for 20% of the total load [103]. Firstly, we used it to smooth out the duck curve, based on actual load profile from CAISO, and reduced ramping by 23.1% compared to the baseline scenario. Secondly, we applied it to curtail peak load based on load profile from PJM. The experiment was conducted on a HIL simulation, including a real-world testbed. Over the 15-day experiment period, COHORT reduced daily peak loads by an average of 12.5%. Furthermore, the occupants living in the real-world testbed, i.e., the first author's apartment, reported no discomfort and could not distinguish whether the AC was operated by on-off control or COHORT. Finally, COHORT is easily integrated with a commercial smart thermostat, which provides readily-available control and communication infrastructure. Thus, our approach is scalable to households already equipped with smart thermostats with minimal effort and no retrofit.

In summary, COHORT is shown to be a practical, scalable, and versatile solution for coordinating TCLs to provide grid services. Currently, quite a few iterations are required to reach consensus for problems with long planning horizons, which may raise

concern about the communication cost. This was discussed in [88], with the conclusion that the network latency and message size between the aggregator and the end users during coordination are not bottlenecks for modern networks[19]. Regardless, the number of iterations could be reduced by initializing COHORT with an approximate solution from imitation learning, to be incorporated as future work. Another research direction would be to extend the current work to other flexible loads, with tracking strategies tailored to those loads.

[19] More concretely, the message size is less than 1KB per agent per iteration in our problem setup and the total message size grows linearly with the number of agents. Overall, the bandwidth requirement is low. As a reference, the network latency for 4G network ranges from 30 to 160ms. Even for 100 iterations, the latency is small compared to a 1-hour re-planning time-step sufficient for load shifting.

5. PROF: PROject Feasibility for energy optimization

This section summarize the work, published in [40], to address to Research Question 3.1 (Section 1.4.3). [138] is a related attempt in embedding the power flow constraints within the learning algorithm.

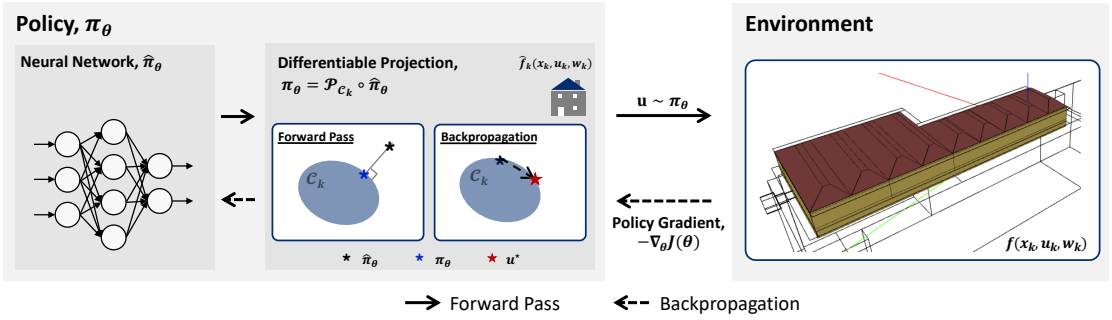


Figure 5.1: The PROF framework. Our policy consists of a neural network followed by a differentiable projection onto a convexified set of operational constraints, $\hat{\mathcal{C}}_k$ (which is constructed via an approximate model, $\hat{f}_k$, of the environment). The differentiable projection layer enforces the constraints in the forward pass, and induces policy gradients that make the neural network cognizant of the constraints in its learning.

5.1. Introduction

There has been increasing interest in using learning-based methods such as reinforcement learning (RL) for applications in energy systems control. However, a fundamental challenge with many of these methods is that they do not respect the physical constraints or functional requirements associated with the systems in which they operate. Therefore, there have been many calls for embedding safety guarantees into learning-based methods in the context of energy systems applications [228, 90, 66].

One common proposal to address this challenge is to provide machine learning methods with "soft penalties" to encourage them to learn feasible solutions. For instance, the authors of [226, 38] incentivize their RL-based building HVAC controller to satisfy thermal comfort constraints by adding a constraint violation penalty to the reward function. While such approaches often involve tuning some weight on the penalty term, recent work has proposed more theoretically-grounded approaches to choosing these weights; for instance, in the setting of approximating AC optimal power flow, the authors of [79, 37] interpret the weight on their constraint violation penalty as a dual variable, and learn it via primal-dual updates. [100] adopt a similar approach in an inverter control problem. However, a challenge with these types of "soft penalty" methods in general is that while they *incentivize* feasibility, they do not strictly *enforce* it, which is potentially untenable in safety-critical applications.

Given this limitation, a second class of approaches has aimed to strictly enforce operational constraints. For instance, in some cases, the outputs of a machine learning algorithm can be clipped post-hoc in order to make them feasible. However, a challenge is that such post-hoc corrections are not taken into account during the learning process, potentially negatively impacting overall performance. More recent approaches based in deep learning have therefore aimed to enforce simple classes of constraints in a way that *can* be taken into account during learning; for instance, [222] train a neural network to approximate AC optimal power flow (OPF), and enforce box constraints on certain variables via sigmoid activations in the last layer of the neural network. In general, however, existing approaches have only been able to accommodate *simple* sets of constraints, prompting a need for methods that can incorporate broader classes of constraints.

In this work, we propose a method to enforce general convex constraints into RL-based controllers in a way that can be taken into account during the learning process. In particular, we construct a neural network-based policy that culminates in a *projection* onto a set of constraints characterized by the underlying system. While the "true" constraints associated with the system may be somewhat complex, we observe that simple, approximate physical models are often available for many systems of interest, allowing us to specify convex approximations to the relevant constraints. The projections onto these (approximate) sets can thus be characterized as convex optimization problems, allowing us to leverage recent developments in differentiable convex optimization [13, 6] to train our neural network and projection *end-to-end* using standard RL methods. The result is a powerful neural policy that can flexibly optimize performance on the true underlying dynamics, while still satisfying the specified constraints.

We demonstrate our PROjected Feasibility approach, PROF, on two settings of interest. Specifically, we explore a building operation setting in which the goal is to reduce energy consumption during the heating season, while ensuring the satisfaction of thermal comfort constraints. We additionally explore an inverter control setting where the goal is to mitigate curtailment, while satisfying inverter operational constraints and nodal voltage bounds. In both settings, we find that our controller achieves good performance with respect to the control objective, while ensuring that relevant operational constraints are satisfied.

To summarize, our key contributions are as follows:

- **A framework for incorporating convex constraints.** We propose a projection-based method to flexibly enforce convex constraints within neural policies (as summarized in Figure 5.1). By examining the gradient fields of the differentiable projection layer, we recommend the incorporation of an auxiliary loss for more robust results. We also show in an ablation study (Section 5.5.3) that propagating gradients through the differentiable projection layer is indeed conducive to policy learning.

- **Demonstration on building control.** In the building control setting, we show that PROF further improves energy efficiency by 10% and 4%, respectively, compared to the best-performing RL agents in [226] and [38]. By using a locally-linear assumption to approximate the building thermodynamics and thereby formulating

the constraints as a polytope [230, 42], we largely maintain the temperature within the deadband, except when the control is saturated.

- **Demonstration on inverter control.** In the inverter control setting, PROF satisfies the voltage constraints 100% of the time over more than half a million time steps (1 week at one second per time step), with a randomly initialized neural network, compared to 22% over-voltage violations incurred by a Volt/Var control strategy. With respect to the objective of minimizing renewable generation curtailment, PROF performs as well as possible within its conservative safety set after learning safely for a day.

5.2. Related Work

Our approach relies on recent developments in implicit neural network layers, and is thematically similar to several recent works in safe RL. We briefly discuss these topics, and refer interested readers to [66, 228, 90, 183, 70] for comprehensive reviews of relevant work in power and energy systems application domains.

Implicit layers. A neural network can be viewed as a composition of functions, or *layers*, with parameters that can be adjusted to improve performance on some task. While many of the layers commonly used within neural networks (e.g., convolutions or sigmoid functions) represent *explicit* functions that provide a direct mapping between inputs and outputs, there has recently been a great deal of interest in expanding the set of commonly-used layers to include those representing *implicit* functions [131]. This has included the creation of layers capturing optimization problems [13, 65, 205, 211, 6, 94], physical equations [62, 48, 95], sequence modeling processes [17], and games [143]. In this work, we leverage advances in differentiable optimization in particular, namely by incorporating a differentiable convex optimization layer into our neural policy in order to project proposed control actions onto the feasible set of constraints.

Safe reinforcement learning. While (deep) RL methods in general lack safety or stability guarantees, there has been recent interest in learning RL-based controllers that attempt to maintain some notion of safety during training and/or inference – e.g., to satisfy physical constraints or avoid particularly negative outcomes [87]. These include methods that aim to determine "safe" regions of the state space by making smoothness assumptions on the underlying dynamics [210, 25, 206, 7], methods that combine concepts from RL and control theory [157, 148, 174, 36, 102, 224, 67], approaches based on formal verification logics [112, 104, 84], and methods that aim to bound some (discounted) cost function characterizing violations of safety constraints [220, 200, 3, 10]. While the particular notion of "safety" considered varies between settings, relevantly to the present work, several of these prior works employ some form of differentiable projection within the loop of deep RL. For instance, within the context of constrained Markov decision processes (C-MDPs), [220] project neural network-based policies onto a linearly-constrained set of policies with bounded cumulative discounted cost. In the

context of asymptotic stability, [67] project the actions output by their controller onto a convex set of actions satisfying stability specifications obtained via robust control. In the setting of robotic motion planning, [173] project actions onto a linear set of robotic operational constraints, and apply separate updates to the neural network based on both pre-projection and post-projection actions. Similarly to this prior work, our approach employs differentiable projections within a neural network policy to enforce operational constraints over some planning horizon.

5.3. Preliminaries

We now present background on technical concepts used by PROF.

5.3.1. Differentiable Projection Layers

As previously described, a neural network is a composition of parameterized functions (*layers*) whose parameters are adjusted during training via *backpropagation* (a class of gradient-based methods). Any function can be incorporated into a neural network as a layer provided that it satisfies two main conditions. The first condition is that it must have a *forward procedure* to map from inputs to outputs (i.e., do *inference*). The second is that it must have a *backwards procedure* to compute gradients of the outputs with respect to the inputs and function parameters, in order to enable backpropagation.

With that in mind, consider the L_2-norm projection $\mathcal{P}_{\mathcal{C}} : \mathbb{R}^n \to \mathcal{C}$ that maps from some point in $\hat{u} \in \mathbb{R}^n$ to its closest point in some constraint set $\mathcal{C} \subseteq \mathbb{R}^n$ as follows:

$$\mathcal{P}_{\mathcal{C}}(\hat{u}) = \arg\min_{u \in \mathcal{C}} \frac{1}{2}\|u - \hat{u}\|_2^2. \tag{5.1}$$

In cases where $\mathcal{C}$ is convex, Equation 5.1 is a convex optimization problem. The forward procedure of this operation can then be implemented by simply solving the optimization problem, e.g., using standard convex optimization solvers. Perhaps less evidently, it is also possible to construct a backwards procedure for this problem by using the implicit function theorem [134], as described in previous work (e.g., [13, 6]).

As an example, consider the case where $\mathcal{C}$ characterizes linear constraints, i.e., $\mathcal{C} \equiv \{u : Au = b, Gu \leq h\}$ for some $A \in \mathbb{R}^{n_{\text{eq}} \times n}$, $b \in \mathbb{R}^{n_{\text{eq}}}$, $G \in \mathbb{R}^{n_{\text{ineq}} \times n}$, and $h \in \mathbb{R}^{n_{\text{ineq}}}$. It is then possible to efficiently compute gradients through Equation 5.1 by implicitly differentiating through its KKT conditions, i.e., conditions that are necessary and sufficient to describe its optimal solutions. In particular, as described in [13], the KKT conditions for stationarity, primal feasibility, and complementary slackness for this case are given by

$$\begin{aligned} u^\star - \hat{u} + A^T\nu^\star + G^T\lambda^\star &= 0 \\ Au^\star - b &= 0 \\ \text{diag}(\lambda^\star)(Gu^\star - h) &= 0, \end{aligned} \tag{5.2}$$

where $u^\star$, $\lambda^\star$, and $\nu^\star$ are the optimal primal and dual solutions. By the implicit function

theorem, we can then take derivatives through these conditions at the optimum in order to obtain relevant gradients. Specifically, the total differentials of these KKT conditions are given by

$$\begin{aligned} \mathsf{d}u - \mathsf{d}\hat{u} + \mathsf{d}A^T\nu^\star + A^T\mathsf{d}\nu + \mathsf{d}G^T\lambda^\star + G^T\mathsf{d}\lambda = 0 \\ \mathsf{d}Au^\star + A\mathsf{d}u - \mathsf{d}b = 0 \\ \mathrm{diag}(Gu^\star - h)\mathsf{d}\lambda + \mathrm{diag}(\lambda^\star)(\mathsf{d}Gu^\star + G\mathsf{d}z - \mathsf{d}h) = 0. \end{aligned} \tag{5.3}$$

As described in [13], these equations can then be rearranged to solve for the Jacobians of any of the solution variables $u^\star, \lambda^\star, \nu^\star$ with respect to any of the problem parameters $\hat{u}, A, b, G, h$ (or, in practice, to solve directly for these Jacobians' left matrix-vector product with some backward pass vector, in order to reduce space complexity).

While the above example is for the case of a linearly-constrained projection operation, these kinds of gradients can be computed for convex projection problems in general. For instance, [67] compute gradients through a projection onto a second order cone by differentiating through the fixed point equations of their solver, and [6] provide a method and library for differentiable disciplined convex programs. A key benefit of using these kinds of projection layers for constraint enforcement is that they allow gradients through the enforcement procedure to flow back to the neural network, thereby informing the parameter updates of this network during training.

5.4. Approach

We now describe PROF, which incorporates differentiable projections onto convex(ified) sets of operational constraints within a neural policy.

5.4.1. Problem Formulation

Consider a discrete-time dynamical system

$$x_{k+1} = f(x_k, u_k, w_k), \tag{5.4}$$

where $x_k \in \mathbb{R}^s$ is the state at time k, $u_k \in \mathbb{R}^a$ is the control input, $w_k \in \mathbb{R}^d$ is an uncontrollable disturbance (which we assume to be observable), and $f : \mathbb{R}^s \times \mathbb{R}^a \times \mathbb{R}^d \to \mathbb{R}^s$ denotes the system dynamics. Letting $\mathcal{X}_k$ and $\mathcal{U}_k$ denote the allowable state and action space, respectively, we can define the set of all feasible actions over the planning horizon T as $\mathcal{C}_k$, where

$$\mathcal{C}_k = \left\{ u_{k:k+T-1} \;\middle|\; \begin{array}{l} x_{i+1} = f(x_i, u_i, w_i), \\ x_i \in \mathcal{X}_i,\ u_i \in \mathcal{U}_i \end{array} \quad \forall i \in \{k, ..., k+T-1\} \right\}. \tag{5.5}$$

Our goal is then to learn a policy that optimizes the control objective, J, while enforcing the operational constraints. To simplify notation, we denote $\mathbf{u} = u_{k:k+T-1}$. In the case

of a deterministic policy, i.e., $\mathbf{u} = \pi_\theta$, the learning problem is simply

$$\min_\theta \quad J(\theta) \quad \text{s.t.} \;\; \pi_\theta \in \mathcal{C}_k. \tag{5.6}$$

In the case of a stochastic policy, e.g. $\mathbf{u} \sim \mathcal{N}(\boldsymbol{\mu}, \text{diag}(\boldsymbol{\sigma}^2))$, $[\boldsymbol{\mu}, \boldsymbol{\sigma}] = \pi_\theta(x_k)$, we can write the problem as

$$\min_\theta \quad J(\theta) \quad \text{s.t.} \;\; \mathbf{u}, \boldsymbol{\mu} \in \mathcal{C}_k. \tag{5.7}$$

In this case, it is necessary to sample actions around $\boldsymbol{\mu}$ in order to estimate policy gradients. At the same time the actions sampled from π_θ might fall outside of $\mathcal{C}_k$. Thus, we enforce that both $\boldsymbol{\mu}$ and the sample action $\mathbf{u}$ satisfy the constraints.

5.4.2. Approximate Convex Constraints

In practice, there are two key challenges inherent in solving Equations 5.6–5.7 as written. The first is that the disturbances w_i are not known ahead of time, meaning that the optimization problem must be solved under uncertainty. One approach to addressing this, from the field of robust control [231], involves constructing an uncertainty set over the disturbance, and then optimizing for worst-case or expected cost under this uncertainty set. Here, we simply assume a predictive model of the disturbances is available. (By re-planning frequently, we observe that the prediction errors have limited empirical impact on performance in the two applications we study.) We will use the notation $\hat{w}_k$ to denote our forecast of the disturbance if k is a future time step, and the true value of the disturbance if k is the present or a prior time step.

The second challenge pertains to the form of the set $\mathcal{C}_k$, which may be poorly structured or otherwise difficult to optimize over. In particular, our framework relies on obtaining convex approximations to the constraints in order to enable differentiable projections (see Section 5.3.1). Fortunately, for many energy systems applications, some approximate model $\hat{f}_k$ is often available based on domain knowledge that allows $\mathcal{C}_k$ to be approximated as a convex set, despite the complex nature of the true dynamical system.

Thus, letting $\hat{f}_i$ denote our approximations of the dynamics and $\hat{w}_i$ denote the (forecast or known) disturbance at each $i = k, \ldots, k+T-1$, we define our approximate convex constraint set as

$$\hat{\mathcal{C}}_k = \left\{ u_{k:k+T-1} \;\middle|\; \begin{array}{ll} x_{i+1} = \hat{f}_i(x_i, u_i, \hat{w}_i), & \forall i \in \{k, ..., k+T-1\} \\ x_i \in \mathcal{X}_i, \; u_i \in \mathcal{U}_i & \end{array} \right\}. \tag{5.8}$$

We note that f and w are approximated *solely* for the purposes of constructing approximate constraint sets, and are not used otherwise during training and inference (i.e., our neural policy interacts with the *true* dynamics and disturbances during training and inference).

Algorithm 3: PROF

procedure `main(`*env, J*`):`
  // input: environment, control objective
  initialize neural network $\hat{\pi}_\theta$, replay memory $\mathcal{M}$
  specify RL algorithm $\mathcal{A}$, batch size M, update interval K
  specify planning horizon T
  // online execution
  for $k = 1, \ldots$ **do**
    observe state x_k
    predict future disturbances $\hat{w}_{k:k+T-1}$
    construct constraint set $\hat{\mathcal{C}}_k$, policy $\pi_\theta = \mathcal{P}_{\hat{\mathcal{C}}_k} \circ \hat{\pi}_\theta$
    compute $u_k =$ `inference(`π_θ, x_k, T`)`
    execute action `env.step(`u_k`)`
    save `memory.append(`x_k, u_k, $\hat{w}_{k:k+T-1}$`)`
    // update policy every K time steps
    if $\mathrm{mod}(k, K) = 0$ **then**
      $\hat{\pi}_\theta =$ `train(`$\hat{\pi}_\theta$, J, $\mathcal{M}$, $\mathcal{A}$`)`
    end
  end
end

procedure `inference(`π_θ, x_k, T`):`
  // input: neural policy, current state, planning horizon
  select action $u_{k:k+T-1} \sim \pi_\theta$
  // only return the current action; replan at each time step
  return u_k
end

procedure `train(`$\hat{\pi}_\theta$, J, $\mathcal{M}$, $\mathcal{A}$`):`
  // input: neural policy, objective, replay memory, RL algorithm
  initialize $\mathcal{L}(\theta) = 0$
  for $i = 1, \ldots, M$ **do**
    sample $x, u, w \sim \mathcal{M}$
    construct constraint set $\hat{\mathcal{C}}_k$, policy $\pi_\theta = \mathcal{P}_{\hat{\mathcal{C}}_k} \circ \hat{\pi}_\theta$
    compute training loss

$$\mathcal{L}(\theta) \mathrel{+}= J(\theta) + \lambda \|\pi_\theta(x) - \hat{\pi}_\theta(x)\|_2^2$$

  end
  train $\hat{\pi}_\theta$ via $\mathcal{A}$ to minimize $\mathcal{L}$
  return $\hat{\pi}_\theta$
end

5.4.3. Policy Optimization

Let $\hat{\pi}_\theta$ be any (e.g., fully-connected or recurrent) neural network parameterized by θ. Our policy entails passing the output from the neural network to the differentiable projection layer $\mathcal{P}_{\hat{\mathcal{C}}_k}$ characterized by the approximate constraints, which enforces that the resultant action is feasible with respect to these constraints. The overall (differentiable) neural policy is then given by

$$\pi_\theta(x_k) = \mathcal{P}_{\hat{\mathcal{C}}_k} \circ \hat{\pi}_\theta(x_k).^{20} \tag{5.9}$$

The key benefit of embedding a differentiable projection into our policy is that it enforces constraints in a way that is visible to the neural network during learning. In this work, we implement the differentiable projection using the `cvxpylayers` library [6].

We construct the following loss function, which is a weighted sum of the control objective J and an auxiliary loss term to be explained shortly in this section. $\lambda > 0$ is a hyperparameter.

$$\mathcal{L}(\theta, x_k) = J(\theta) + \lambda \|\pi_\theta(x_k) - \hat{\pi}_\theta(x_k)\|_2^2. \tag{5.10}$$

We then train our policy (Equation 5.9) to minimize this cost using standard approaches in deep reinforcement learning. The full algorithm is presented in Algorithm 3.

Visualization of gradient fields. To provide more intuition on the differentiable projection layer and our cost function, we visualize the gradient fields in a hypothetical example with a deterministic policy and a planning horizon of $T = 1$. Specifically, for the purposes of illustration, let $u^\bullet$ and $u^\star$ denote unique optimal actions minimizing some convex control cost J in the unconstrained and constrained settings, respectively:

$$\begin{aligned} u^\bullet \sim \pi_{\theta^\bullet}; \quad \theta^\bullet &= \arg\min_\theta \quad J(\theta) \\ u^\star \sim \pi_{\theta^\star}; \quad \theta^\star &= \arg\min_\theta \quad J(\theta) \quad \text{s.t. } u \in \mathcal{C}_k. \end{aligned}$$

In Figure 5.2, we then plot the gradient fields in two cases: (a) $u^\bullet \notin \mathcal{C}_k$, and (b) $u^\bullet \in \mathcal{C}_k$. Note that $u^\bullet$ and $u^\star$ are assumed to be known here for illustrative purposes only, and are not known during training.

In particular, we plot the gradients (black arrows) of $\|u^\bullet - \mathcal{P}_{\mathcal{C}_k} \circ \hat{\pi}\|_2^2$ with respect to the output of the neural network $\hat{\pi}$. These indicate the direction in which the neural network would be incentivized to update in order to minimize the system cost. If no differentiable projection were embedded within the policy, all the gradients would point towards $u^\bullet$ without regard for the constraints. Instead, in the case of $u^\bullet \notin \mathcal{C}_k$ (Figure 5.2a), the gradients through the differentiable projection layer point towards $u^\star$ instead of $u^\bullet$. More specifically, if $\hat{\pi}_\theta(x_k) \in \mathcal{C}_k$, then the projection layer is simply the identity, and the gradients point directly towards $u^\star$; otherwise, the gradients point along the boundary of $\mathcal{C}_k$ in the direction of $u^\star$.

This case is of particular interest, as in many practical applications some operational constraint will be binding. As a concrete example, the ultimate energy-saving strategy

[20] We use the notation $f \circ g(x) \coloneqq f(g(x))$ to denote function composition.

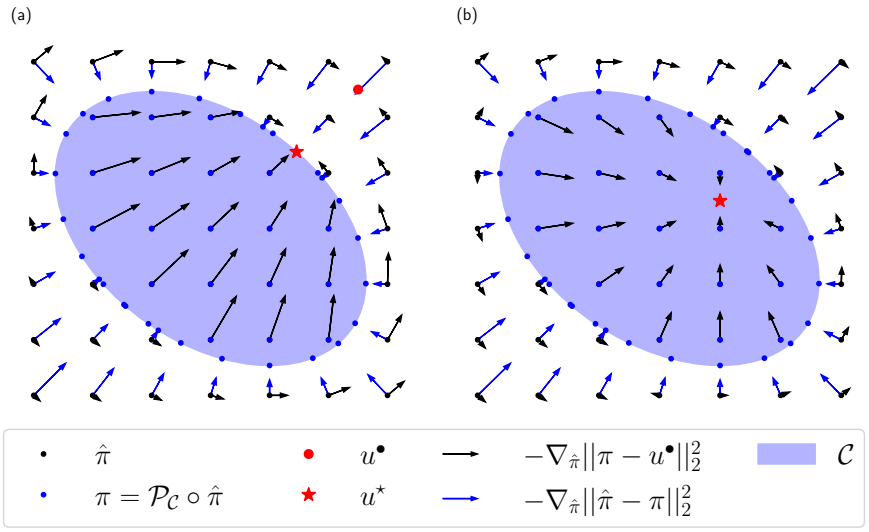


Figure 5.2: Illustrative example of gradients from the differentiable projection layer. $u^{\bullet}$ and $u^{\star}$ denote unique optimal actions minimizing some convex control objective J in the unconstrained and constrained settings, respectively; $\nabla_{\hat{\pi}} \|\pi - u^{\bullet}\|_2^2$ is thus a proxy for $\nabla_{\hat{\pi}} J$. (a) $u^{\bullet} \notin \mathcal{C}$. The gradients $\nabla_{\hat{\pi}} J$ point towards $u^{\star}$ as desired, such that $\pi = \mathcal{P}_{\hat{\mathcal{C}}} \circ \hat{\pi}$ will reach this optimal point. (b) $u^{\bullet} = u^{\star}$ on the interior of $\mathcal{C}$. The gradients $\nabla_{\hat{\pi}} J$ do not cause $\hat{\pi}$ (or its projection) to update towards the interior. Adding a weighted auxiliary loss term, e.g., $\|\pi - \hat{\pi}\|$, can help direct updates towards the interior.

for building operations is to keep all mechanical systems off (i.e., $u^\bullet = 0$), which obviously violates occupants' comfort requirements and is outside the set of allowable actions (i.e., $u^\bullet \notin \mathcal{C}_k$). Thus, the problem is to find a policy that uses the mechanical system as little as possible without violating comfort requirements. Given the common case where the control objective is convex, this then lies on the boundary of the constraint set (i.e., $u^\star = \mathcal{P}_{\mathcal{C}_k} \circ u^\bullet$).

We also depict the case where the solution of the unconstrained problem already satisfies the constraints, i.e., $u^\bullet = u^\star \in C_k$ (Figure 5.2b). If this is generally the case for a particular application, we note that a constraint enforcement approach (ours or otherwise) is likely not needed, and indeed utilizing gradients through the projection layer may actually degrade performance. Specifically, if $\hat{\pi}_\theta(x_k) \notin \mathcal{C}_k$, the gradients do not point towards the interior of the constraint set, meaning that $\pi_\theta(x_k) = \mathcal{P}_{\mathcal{C}_k} \circ \hat{\pi}_\theta(x_k)$ will lie on the boundary of the constraints despite the optimal solution being in the interior. This can be amended by augmenting the loss function with a (weighted) auxiliary term such as $\|\pi_\theta(x_k) - \hat{\pi}_\theta(x_k)\|_2^2$ whose gradients (blue arrows) point towards the interior.

It may not be known a priori whether or not $u^\bullet$ is in the constraint set in general or at any given time, except when domain experts are fully clear on the structure of the solutions for specific applications. In particular, $\mathcal{C}_k$ is time-varying, making it difficult to know for sure whether or not the constraints will indeed be binding at any given time. For robustness, we therefore recommend incorporating the auxiliary loss $\|\pi_\theta(x_k) - \hat{\pi}_\theta(x_k)\|_2^2$ within the RL training cost, unless it is known from domain knowledge that the constraints will certainly be active. As such, we formulate the training cost function as previously given in Equation 5.10.

5.5. Experiment 1: Energy-efficient Building Operation

There is significant potential to save energy through more efficient building operation. Buildings account for about 40% of the total energy consumption in the United States, and it is estimated that up to 30% of that energy usage may be reduced through advanced sensing and control strategies [78]. However, this potential is largely untapped, as the heterogeneous nature of building environments limits the ability of control strategies developed for one building to scale to others [38]. RL can address this challenge by adapting to individual buildings by directly interacting with the environment.

The most important constraint in building operation is to maintain a satisfactory level of comfort for occupants, while minimizing energy consumption. It is common in the RL-based building control literature to penalize thermal comfort violations [226, 38], which incentivizes but does not guarantee the satisfaction of these comfort requirements. In comparison, our proposed neural policy can largely maintain temperature within the specified comfortable range, except when the control is saturated.

We evaluate our policy in the same simulation testbed as [226, 38], following the same experimental setup as [38]. Specifically, we first pre-train the neural policy by imitating a proportional-controller (P-controller). We then evaluate and further train our agent in the simulation environment, using a different sequence of weather data.

5.5.1. Problem Description

Simulation testbed. We utilize an EnergyPlus (E+) model of a 600m^2 multi-functional space (Figure 5.3a), based on the Intelligent Workplace (IW) on Carnegie Mellon University (CMU) campus, located in Pittsburgh, PA, USA. The system of interest is the water-based radiant heating system, of which a schematic is provided in Figure 5.3b. In this experiment, we control the *supply water temperature* so as to maintain the state variable, i.e., the *zone temperature*, within a comfortable range during the heating season. In the existing control, the supply water (SW) is maintained at a constant flow rate, and its temperature is managed by a P-controller. For more information on the simulation testbed, refer to [226].

Approximate system model. We approximate the environment as a linear system as follows:

$$x_{k+1} \approx \hat{f}(x_k, u_k, w_k) = Ax_k + B_u u_k + B_d w_k, \tag{5.11}$$

where x_k represents the *zone temperature* and u_k represents the *supply water temperature*. w_k includes distributions from weather and occupancy. While building thermodynamics are fundamentally nonlinear, the locally-linear assumption works well for many control inputs [177]. We identify the approximate model parameters A, B_u, and B_d with prediction error minimization [177] on the same data used to pre-train the RL agent (see Section 5.5.2). The root mean squared error (RMSE) of this model on a unseen test set is 0.14°C.

Objective. Since our goal is to minimize energy consumption, we define the control cost at each time step as the agent's control action, i.e. *supply water temperature*, which is linearly proportional to the heating demand, i.e., $c_k = u_k$.

In contrast to the objectives in [226, 38], which are defined as weighted sum of energy cost and some penalty on thermal comfort violations, we consider the thermal comfort requirement as hard constraints, in the form of Equation 5.7.

Constraints. To maintain a satisfactory comfort level, we require the zone temperature to be within a deadband $\mathcal{X} = \{x \mid 21.9^oC \leq x \leq 25.5^oC\}$ when the building is occupied, based on the building code requirement of 10% Predicted Percentage of Dissatisfied (PPD) [77]. We allow for a wider temperature range during unoccupied hours. For the action, the allowable range of supply water temperature for the physical system is $\mathcal{U} = \{u \mid 20^oC \leq u \leq 65^oC\}$.

While it may appear from this description that we have only simple box constraints on both the state and action, we highlight the fact that actions are coupled over time through the building thermodynamics [230]. More concretely, a future state depends on all past actions. Thus, a box constraint on x_{k+l+1} is in fact a constraint on $u_{k:k+l}$. In this case, assuming $\hat{f}$ to be a linear system, $\hat{\mathcal{C}}_k$ is then a set of linear inequalities, which can be geometrically interpreted as a polytope.[21] We refer interested readers to [42, 230]

[21] A polytope can be characterized as a set $\mathcal{S} = \{x \in \mathbb{R}^n | Ax \leq b\}$.

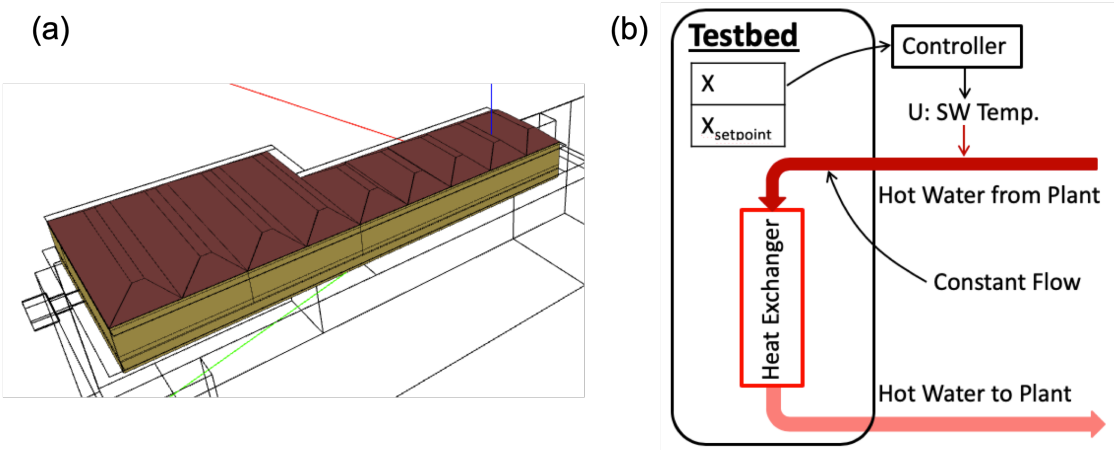


Figure 5.3: Building simulation testbed (reproduced from [38]).

for more details on this formulation. In fact, it was experimentally demonstrated in [42] that projecting actions onto the polytope constructed with an approximate linear model was sufficient to maintain temperature within the deadband in a real-world residential household (though [42] did not then differentiate through this projection).

Control time step. The EnergyPlus model has a 5-minute simulation time step. Following [226, 38], we use a 15-min control time step (i.e., each action is repeated 3 times) and a planning horizon of $T = 12$ (i.e., a 3 hour look-ahead).

5.5.2. Implementation Details

Offline pre-training. We pre-train a long short-term memory (LSTM) recurrent policy (without a subsequent projection) by imitating a P-controller operating under the Typical Meteorological Year 3 (TMY3) [217] weather sequence, from Jan. 1 to Mar. 31. We min-max normalize all of the state, action, and disturbance, and use a learning rate of 10^{-3}. Specifically, we use the pre-trained weights after training on the expert demonstrations for 20 epochs following the same procedures as [44]. We refer readers to [44] for more details on the neural network architecture, training procedures, loss, and performance evaluation.

Online policy learning. We optimize the policy with PPO [190] over the weather sequence in 2017 from Jan. 1 to Mar. 31. We use $\lambda = 10$ (see Equation 5.10), a learning rate of 5×10^{-4}, and RMSprop [203] as the optimizer[22]. We update the policy every four days, by iterating over those samples for 8 epochs with a batch size of 32. For hyperparameters, we use a temporal discount rate of $\gamma = 0.9$, $\epsilon = 0.2$ (see Equation 3.3), and a Gaussian policy (see Equation 5.7) with σ linearly decreased from 0.1 to 0.01.

[22]The code is available at https://github.com/INFERLab/PROF.

Table 5.1: Performance comparison. Our method saves energy while incurring minimal comfort violations.

	Heating Demand (kW)	**PPD Mean (%)**	**PPD SD (%)**
Existing P-Controller [226]	43709	9.45	5.59
Agent #6 [226]	37131	11.71	3.76
Baseline P-Controller [38]	35792	9.71	6.87
Gnu-RL [38]	34687	9.56	6.39
LSTM & Clip + No Update	37938	**8.55**	3.39
LSTM & Clip	36068 ± 2187	9.18 ± 0.67	3.49
PROF (ours)	**33271** ± 1862	9.68 ± 0.48	3.66

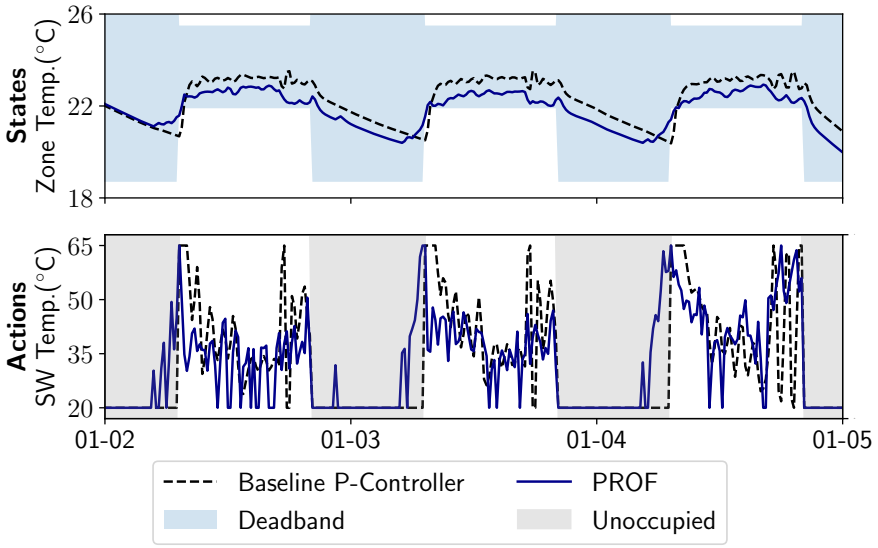
5.5.3. Results

After pre-training on expert demonstrations from the baseline P-controller, our agent directly operated the simulation testbed based on actual weather sequences in Pittsburgh from Jan. 1 to Mar. 31 in 2017. Figure 5.4a shows the behavior of our agent at the onset of deployment over a 3-day period. The baseline P-controller reactively turns on heating when the environment switches from unoccupied to occupied, which results in thermal comfort violations in the mornings. In comparison, PROF preheats the environment such that the environment is already at a comfortable temperature when occupants arrive in the morning. Notably, the differentiable projection layer manages to enforce this preheating behavior despite this behavior not being present in the expert demonstrations.

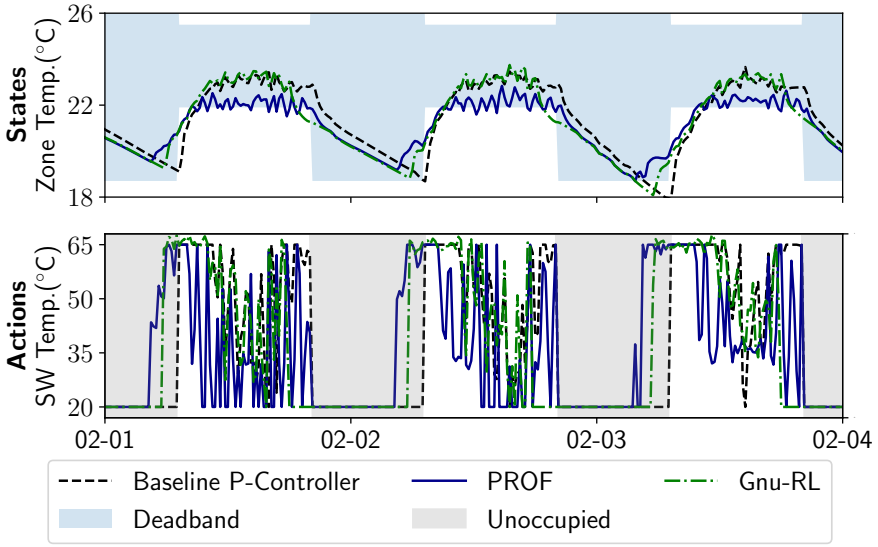
Figure 5.4b shows the behavior of our agent in comparison with Gnu-RL [38], having interacted with and trained on the environment for a month. Gnu-RL is updated via PPO, similarly to the current work, and incorporates domain knowledge on system dynamics. In comparison to Gnu-RL [38], which ends up trying to maintain temperature at the setpoint, PROF learns an energy-saving behavior by maintaining the temperature at the lower end of the deadband. This explains the further energy savings compared with Gnu-RL [38]. However, we also notice that the temperature requirement may be violated on cold mornings. This happens when the control action is saturated, i.e., full heating over the 3-hour planning horizon is not sufficient to bring temperature back to the comfortable range. (In principle, even these constraint violations could be mitigated by increasing the length of the planning horizon.)

Table 5.1 summarizes the performance of our agent with comparison to the RL agents in [226, 38]. Our proposed agent (averaged over 5 random seeds) saves 10% and 4% energy compared to the best-performing agents in [226] and [38], respectively.

We also compare our method to two ablations: (1) `LSTM & Clip + No Update`, which uses the same pre-trained weights and the projection layer to enforce feasible actions, but does not update the policy, and (2) `LSTM & Clip`, which uses the same pre-trained weights and the projection layer to enforce feasible actions during inference, but does not



(a) The differentiable projection layer enforces preheating behavior to ensure deadband constraints are never violated, even though this behavior is not present in the expert demonstrations.



(b) The agent has found a more energy-efficient control strategy by maintaining temperature at the lower end of the deadband.

Figure 5.4: Behavior of our proposed agent (a) at the onset of deployment, with pretrained weights based on expert demonstrations and (b) after a month of interacting with and training on the environment.

propagate gradients through the differentiable projection layer in the policy updates. We find that `LSTM & Clip` slightly improves upon `LSTM & Clip + No Update`, but is less performant compared to PROF. This affirms our hypothesis that the gradients through the differentiable projection layer are cognizant of the constraints and are thus conducive to policy learning.

5.6. Experiment 2: Inverter Control

Distributed energy resources (DERs), e.g., solar photovoltaic (PV) panels and energy storage, are becoming increasingly prevalent in an effort to curb carbon dioxide emissions and combat climate change. However, DERs interfacing with the power grid via power electronics, such as inverters, also introduce unintended challenges for grid operators. For instance, over-voltages have become a common occurrence in areas with high renewable penetration [196], and power electronics-interfaced generation has low-inertia and requires active control at much faster timescales compared to traditional synchronous machines [154].

To alleviate these issues, IEEE standard 1547.8-2018 [22] recommends a Volt/Var control strategy in which the reactive power contribution of an inverter is based on local voltage measurements. As will be clear in our empirical evaluation, this network-agnostic heuristic based on local information alleviates, but does not avoid, over-voltage issues. Given that the optimal solution needs to be obtained at the system-level and that the problem needs to be solved at very short timescales, a common paradigm is to address the problem in a quasi-static fashion [116] adopted in works such as [18, 116, 100], where one chooses a policy over the next time period, e.g., 15 minutes-1 hour, and uses the policy without update for fast inference. In this work, we adopt the same paradigm and consider real-time control on a 1-second timescale of both active (P) and reactive (Q) power setpoints at each inverter.

We envision that a neural policy can learn from its prior experiences, in contrast to the traditional *fit-and-forget* approach [66], and is capable of making decisions faster compared to solving optimization problems. Our primary contribution compared to existing work is the ability to enforce physical constraints within the neural network. In fact, we successfully enforce voltage constraints 100% of the time with a randomly initialized neural network, over more than half a million time-steps (i.e., 1 week with a one-second time step). The assumed control and communication scheme is consistent with the new definitions for smart inverter capabilities under IEEE standard 1547.1-2020 [114].

5.6.1. Problem Description

The problem we are considering here is to control active and reactive power setpoints at each inverter in order to maximize utilization (i.e., minimize curtailment) of renewable generation, while satisfying the maximum and minimum grid voltage requirements. Here, we first define the considered test case and input data, and describe the model of the network. We refer readers to [18] for more details on the problem set-up.

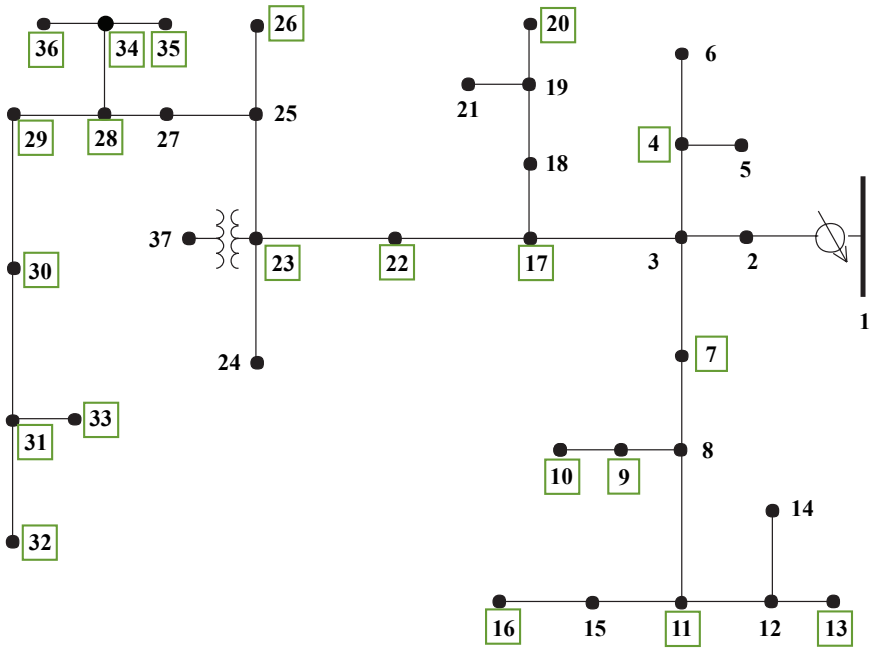


Figure 5.5: IEEE 37-bus feeder system, where the solar PV systems are indicated by green rectangles.

IEEE 37-bus test case. We evaluate our method on the IEEE 37-bus distribution test system [113], with 21 solar PV systems indicated by green rectangles in Figure 5.5. We utilize a balanced, single-phase equivalent of the system, and simulate the nonlinear AC power flows using PYPOWER [180]. For the simulation, the solar generation and loads are based on 1-second solar irradiance and load data collected from a neighborhood in Rancho Cordova, CA [19] over a period of one week (604800 samples).

Approximate system model. Denote the number of buses, excluding the slack bus (e.g., the distribution substation), as N, the net active and the reactive power as $\mathbf{p} \in \mathbb{R}^N$ and $\mathbf{q} \in \mathbb{R}^N$, and the voltage at all buses as $\mathbf{v} \in \mathbb{R}^N$. We linearize the AC power flow equations around the flat voltage solution, i.e. $\bar{\mathbf{v}} = 1$, using the method in [28]. The reference active and reactive power corresponding to $\bar{\mathbf{v}} = 1$ is denoted as $\bar{\mathbf{p}}$ and $\bar{\mathbf{q}}$. The linearized grid model, $\hat{f}$, is given by Equation 5.12, where $\mathbf{R}$, $\mathbf{B} \in \mathbb{R}^{N \times N}$ represent system-dependent network parameters that can be either estimated from linearization (e.g., [28]) or data-driven methods:

$$\begin{aligned} \mathbf{v} \approx \hat{f}(\mathbf{p}, \mathbf{q}) &= \bar{\mathbf{v}} + \mathbf{R}(\mathbf{p} - \bar{\mathbf{p}}) + \mathbf{B}(\mathbf{q} - \bar{\mathbf{q}}) \\ &= \bar{\mathbf{v}} + \underbrace{[\mathbf{R}, \mathbf{B}]}_{\mathbf{H}} \underbrace{\begin{bmatrix} \mathbf{p} - \bar{\mathbf{p}}, \\ \mathbf{q} - \bar{\mathbf{q}} \end{bmatrix}}_{\mathbf{u}}. \end{aligned} \tag{5.12}$$

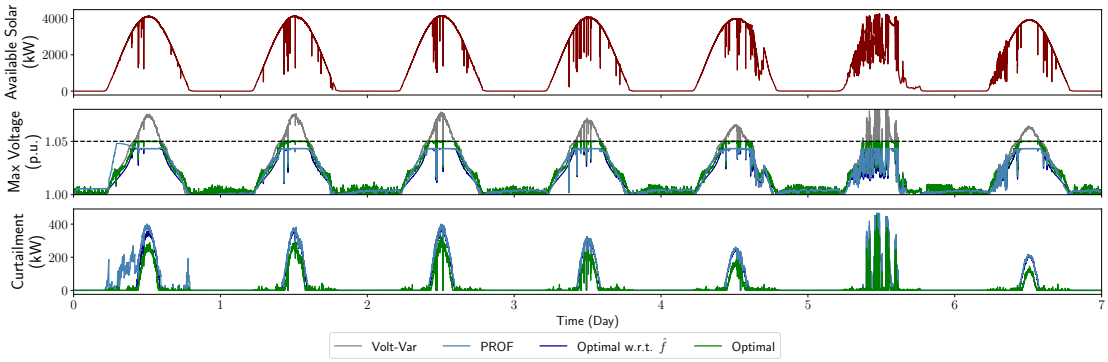


Figure 5.6: PROF satisfies voltage constraints throughout the experiment, and learns to minimize curtailment as well as possible within its conservative safety set, $\hat{C}_k$, after learning safely for a day.

A notable advantage of the method in [28] is that the resulting model has bounded error with respect to the true dynamics. By incorporating the error bound when constructing the safety set, the safety set is guaranteed to be a conservative under-approximation of the true safety set, and thus allow us to satisfy voltage constraints 100% of the time.

Policy. Our policy takes as input the voltage from the previous time-step, load, and generation at all the buses, and outputs active and reactive power setpoints at each inverter. (This is a deterministic policy; see Equation 5.6.) Note that while the grid model (Equation 5.12) contains all N buses, only those with inverters are controllable.

Our neural architecture is similar to the one used in [100], which consists of a utility-level network, and inverter-level networks for individual inverters. The utility-level network collects information from all nodes, and broadcasts an intermediate representation to all inverter-level networks. Using this information along with its local observations, each inverter makes its local control decisions, which are then projected onto the constraints (discussed below).

Objective. The objective is to minimize the curtailment of solar generation, or equivalently to maximize the utilization of the available solar power, p_{av}. Specifically, letting $\mathcal{I}$ denote the set of buses with inverters, the objective is

$$J(\theta) = \min_{\mathbf{p}_\mathcal{I}, \mathbf{q}_\mathcal{I}} \sum_{i \in \mathcal{I}} [p_{av,i} - p_i]_+, \quad \text{where } \begin{bmatrix} \mathbf{p}_\mathcal{I} & \mathbf{q}_\mathcal{I} \end{bmatrix} = \pi_\theta \tag{5.13}$$

Constraints. For an individual inverter, i, with rated power s_i and an available power (from available solar generation) $p_{av,i}$, the feasible action space is

$$\mathcal{U}_i(k) = \{(p_i, q_i) : 0 \le p_i \le p_{av,i}(k), p_i^2 + q_i^2 \le s_i^2\}$$

$$\mathcal{U}(k) := \mathcal{U}_1(k) \times \cdots \times \mathcal{U}_{|\mathcal{I}|}(k).$$

At the same time, the voltage at each bus should remain between 0.95-1.05 $p.u.$ The primary challenge of satisfying voltage constraints is that the voltage at each bus depends on actions of neighboring nodes, i.e.

$$\mathcal{X} = \{v \mid 0.95 \times 1 \leq \mathbf{v} \approx \bar{\mathbf{v}} + \mathbf{H}\mathbf{u} \leq 1.05 \times 1\},$$

where the sparsity pattern of H is characterized by the admittance matrix. We jointly project actions from all inverters at each time step k onto the constraints $\mathcal{U}(k) \cap \mathcal{X}$.

5.6.2. Implementation Details

We evaluate PROF by executing it over the 1-week dataset (at 1 second) once. Similarly to other quasi-static approaches, we update the policy every 15-minutes. Similarly to [100], we optimize the neural policy with stochastic samples by directly differentiating through the objective (Equation 5.13) and the linearized grid model (Equation 5.12). However our method differs in that [100] characterized the constraints as a regularization term, and learned the policy via primal-dual updates. We incorporate the constraints directly via the differentiable projection layer and thus guarantee constraint satisfaction.

We use λ=10 (see Equation 5.10), a learning rate of 10^{-3}, and RMSprop [203] as the optimizer. At every 15 minutes, we sample 16 batches of data with size of 64 from the replay memory. We keep a replay memory size of 86400, i.e., samples from the previous day. For the both the utility-level network and the inverter-level network, we use fully-connected layers with ReLU activations. The utility-level network has hidden layer sizes (256, 128, 64), and each inverter-level network has hidden layer sizes (16, 4) and outputs active and reactive power. On top of the neural network, we implement the differentiable projection layer, following the constraints described in Section 5.6.1.

We compare our methods to three baselines, (1) a Volt/Var strategy following IEEE 1547.8 [22], (2) the optimal solution with respect to the linearized grid model, and (3) the optimal solution with respect to the true AC power flow equations.

5.6.3. Results

The performance of PROF in comparison to the three baselines is summarized in Figure 5.6. For clarity, we only show the maximum voltage over all buses; under-voltage is not a concern for this particular test case.

We see that the Volt/Var strategy violates voltage constraints 22.3% of time, mostly around noon when the solar generation is high and there is a surplus of energy. Since the Volt/Var baseline does not adjust active power, there is no curtailment.

In comparison, PROF satisfies the voltage constraints throughout the experiment, even with a randomly initialized neural policy. While PROF performs poorly on the first morning, it quickly improves its policy. In fact, the behavior of PROF is barely distinguishable from the optimal solution with respect to the linearized grid model, after learning safely for a day. This implies that PROF learned to control inverters as well as possible given its approximate model, which constructs a conservative under-approximation of the true safety set.

The optimal baseline with respect to the true AC power flow equations unsurprisingly achieves the best performance with respect to minimizing curtailment, as it can push the maximum voltage to the allowable limit in order to maximally reduce the amount of curtailed energy. However, inverter control is a task that requires near real-time inputs, and we find that running this baseline can be prohibitively slow. Specifically, we evaluate the computation time of different operations by averaging over 1000 randomly sampled problems from our dataset on a personal laptop. For PROF, on average, a forward pass in the neural network (excluding the projection layer) took 4.5 ms and the differentiable projection operation took 8.6 ms. The computation cost of the differentiable projection could be further reduced by using customized projection solvers such as the ones in [13, 67] that avoid the "canonicalization" costs introduced by general-purpose solvers such as the one we use [6]. In comparison, solving the optimization baseline with respect to the true AC power flow equations took 1.02s on the same machine, which is even longer than the 1s control time-step.

5.7. Discussion and Conclusions

In this work, we have presented a method, PROF, for integrating convex operational constraints into neural network policies for energy systems applications. In particular, we propose a policy that entails passing the output of a neural network to a differentiable projection layer, which enforces a convex approximation of the operational constraints. These convex constraint sets are obtained using approximate models of the system dynamics, which can be fit using system data and/or constructed using domain knowledge. We can then train the resultant neural policy via standard RL algorithms, using an augmented cost function designed to effect desirable policy gradients. The result is that our neural policy is cognizant of relevant operational constraints during learning, enhancing overall performance.

We find in both the building energy optimization and inverter control settings that PROF successfully enforces relevant constraints while improving performance on the control objective. In particular, in the building thermal control setting, we find that our approach achieves a 4% energy savings over the state of the art while largely maintaining the temperature within the deadband. In the inverter control setting, our method perfectly satisfies the voltage constraints over more than half a million time steps, while learning to minimize curtailment as much as possible within the safety set.

While these results demonstrate the promise of our method, a key limitation is in its computational cost. In particular, computing a projection during every forward pass of training and inference is decidedly more expensive than running a "standard" neural network. A fruitful area for future work – both in the context of our method, and in the context of research in differentiable optimization layers as a whole – may be to improve the speed of such differentiable projection layers. For instance, this might entail developing special-purpose differentiable solvers [13, 67] for optimization problems commonly encountered in energy systems applications, developing approximate solvers that do not rely on obtaining optimal solutions in order to compute reasonable gradients, or employing cheaper projection schemes such as α-projection [191] where possible.

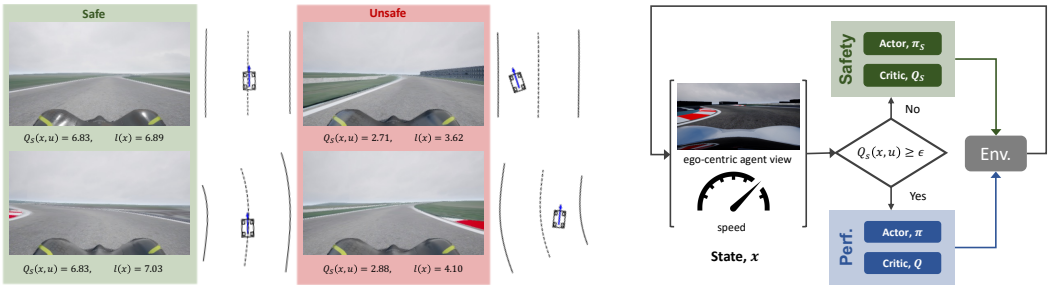
Additionally, the success of our method (and many other constraint enforcement methods) depends fundamentally on the quality of the approximate model used to characterize the constraint sets. In particular, this determines the extent to which the resultant approximate constraint sets are a good representation of the true operational constraints. While we were able to employ reasonably high-quality approximation schemes in the context of this work, future work on safely updating the models or the constraint sets directly [81] may greatly improve the quality of the solutions.

More generally, while our work highlights one approach to enforcing physical constraints within learning-based methods, we believe this is only the start of a broader conversation on closely integrating domain knowledge and control constraints into learning-based methods. In particular, strictly enforcing physical constraints will be paramount to the real-world success of these methods in energy systems contexts, and we hope that our paper will serve to spark further inquiry into this important line of work.

6. SAGE: Safe Autonomous racinG on Ego-vision

This section addresses Research Question 3.2 (Section 1.4.3) in a challenging task of autonomous racing. Analogous to racing being traditionally used as the proving ground for automotive technology, we also envision autonomous racing to be a particularly challenging proving ground for autonomous agents. As autonomous technology approaches maturity, it is of paramount importance for autonomous vehicles to adheres to safety specifications, whether in urban driving or high-speed racing. Racing demands each vehicle to drive at its physical limits with little safety margin, when any infraction could lead to catastrophic failures. Given this inherent tension, autonomous racing serves as a particularly challenging proving ground for safe learning algorithms.

[108] is a precursor of this work, where we developed and released an open-source, high-fidelity, and multi-modal environment, `Learn-to-Race` (`L2R`) for autonomous racing.



(a) Examples of Safe vs. Unsafe States[23] (b) SAGE Architecture

Figure 6.1: SAGE Overview. (a) The safety critic, Q_S, verifies the safety of a state-action pair by checking if $Q_S(x, u) \geq \epsilon$. Some examples of safe vs. unsafe states are provided, using safety margin $\epsilon = 3$, $u = \mathbf{0}$, and speed = 10m/s. (While vehicle pose is NOT available to the safety critic, we illustrate them here for reference.) (b) SAGE consists of two policies, which are in charge of safety and performance, independently. The safety controller intervenes when the current state-action pair is deemed unsafe by the safety critic.

6.1. Introduction

Racing requires each vehicle to make sub-second decision in a fast changing environment and operate at its physical limits [144], when any safety infraction could lead to catastrophic failure. Thus, autonomous racing is a particularly challenging proving ground for autonomous agents to optimize performance, while adhering to safety constraints. In the reinforcement learning (RL) literature, it is common to define safety as satisfying safety specifications [181] under the constrained Markov decision process (CMDP) framework [10], which extends the Markov decision process (MDP) by incorporating constraints on expected cumulative costs.

Due to the low sensor cost and high information content, camera-based perception is gaining increasing popularity in autonomous vehicles [197]. While end-to-end au-

tonomous driving on visual input is an extensively-researched topic for urban driving, largely thanks to the release of the CARLA simulator [68], it is less so for high-speed racing, which may be partly attributed to the lack of open-source, high-fidelity simulation environments. The recent release of `Learn-to-Race` (`L2R`) [107] changes that and lowers the barrier of entry for autonomous racing research.

In this work, we study the problem of constrained RL for autonomous racing, using the vehicle's ego-camera view and speed as input. Due to the nature of the task, the autonomous agent needs to be able to 1) identify and avoid unsafe scenarios and 2) make fast safety verification given the perception data. In Figure 6.1a, we show examples of ego-camera views and the corresponding safety value $Q_S(x, u)$, estimated by our proposed safety critic, and the distance to road boundary $l(x)$. While it is straightforward to determine whether a state is safe based on the vehicle pose, which is illustrated for reference, the distinction from ego-camera views is much more subtle. Also evident from the examples is that safety does not necessarily corresponds to distance to road boundary. Regarding the requirement for fast decision-making, in our experiments, `L2R` operates under the setting where the simulator executes the agent's command upon receiving it, and does not wait for the agent to complete its computation. Thus, high latency can adversely impact agent performance, where, as discussed in prior art [197], perception stacks in autonomous race-cars account for nearly 60% of total latency.

Given these considerations, we propose to incorporate Hamilton-Jacobi (HJ) reachability theory, a safety verification method for general non-linear systems, into the CMDP framework. HJ reachability not only provides a control-theoretical approach to learn about safety, but also enables low-latency safety verification. As a reachable set takes into consideration all possible trajectories over a specified time horizon, safety verification via HJ reachability only requires evaluating the safety value of the current state. Furthermore, safety verification under HJ Reachability theory does not depend on the performance policy. Thus, we can bypass the challenges involved with solving a constrained optimization problem with a neural policy and learn two policies that independently manage safety and performance (Figure 6.1b): the performance policy focuses exclusively on optimizing performance, while the safety critic verifies if the current state is safe and intervenes when necessary. We refer to our approach as **S**afe **A**utonomous racin**G** on **E**go-vision (SAGE).

Secondly, we compare the HJ Bellman update rule [81] to alternatives for learning a safety critic [194, 26] on two classical control benchmarks, where safe states are known, analytically. In comparison to expected cumulative costs, the safety value defined by HJ reachability characterizes the worst case outcome, which is more amenable to safety analysis. Given the same off-policy samples, the HJ Bellman update rule is more accurate and sample efficient.

Finally, we evaluate our methods on `Safety Gym` [181] and `Learn-to-Race` (`L2R`) [107], a recently-released, high-fidelity autonomous racing environment, which challenges the agent to make safety-critical decisions in a complex and fast-changing environment. While SAGE is by no means free from failure, it has significantly fewer constraint vi-

[23] Unsafe here refers to $Q_S(x, u) < \epsilon$.

olations compared to other constrained RL baselines in `Safety Gym`. We also report new state-of-the-art results on the `L2R` benchmark task, and show that incorporating a learnable safety critic grounded in control theory boosts performance especially during the initial learning phase.

6.2. Related Work

Autonomous racing. One approach for autonomous racing is via model predictive control [144, 184, 119], which solves an optimization problem with a model of the system dynamics. Aside from the challenges in modeling the complex dynamics, a significant drawback of such approach is the dependence on extensive sensor installation for localization and state estimation [34]. Another approach is to use a modular pipeline [119, 197], starting from perception on raw sensory inputs, to localization and object-detection, and finally to planning and control. While this approach is most commonly used in practice, disadvantages of the approach include over-complexity and error propagation [221, 83]. Recently, there is a lot of interest in using RL-based approaches for autonomous racing. In [82, 50], RL agents were trained using low-dimensional features as inputs. In [45, 69], intermediate features were extracted from perception pipelines to determine control actions. In [34, 216], RL agents were trained end-to-end on visual inputs by imitating expert demonstration; in [34], a data-driven model of the environment was further utilized to train the agent by unrolling future trajectories.

In comparison to racing, there is significantly more literature on end-to-end autonomous driving for urban scenarios [54, 167, 55, 46, 223, 176, 227, 223]. It is beyond our scope to cover this large research field, and we refer interested readers to survey papers, such as [221, 96], for more information. While we focus on high-speed racing and its unique challenges, we believe the discussion here for safety analysis on ego-vision is also relevant to urban driving.

Constrained reinforcement learning. There is growing interest in enforcing some notion of safety in RL algorithms, e.g., satisfying safety constraints, avoiding worst-case outcomes, or being robust to environmental stochasticity [87]. We focus on the notion of safety as satisfying constraints. CMDP [10] is a widely-used framework for studying RL under constraints, where the agent maximizes cumulative rewards, subject to limits on cumulative costs characterizing constraint violations. Solving a CMDP problem is challenging, because the policy needs to be optimized over the set of feasible states; this requires off-policy evaluation of the constraint functions, to determine whether a policy is feasible [4]. As a result, safety grows with experience, but requires diverse state-action pairs, including unsafe ones [194]. Furthermore, one needs to solve a constrained optimization problem with a non-convex neural policy. This may be implemented with techniques inspired by convex optimization, such as primal-dual updates [26] and projection [220], or by upper bounding the expected cost at each policy iteration [4]. Most relevant to our work is [26, 194, 201], which also uses a safety critic to verify if a state is safe; we compare our control-theoretic learning rule with theirs in Section 6.5.1.

Guaranteed safe control. Guaranteeing the safety of general continuous nonlinear systems is challenging, but there are several approaches that have been successful. These

methods typically rely on knowledge of the environment dynamics. Control barrier functions (CBFs) provide a measure of safety with gradients that inform the acceptable safe actions [11]. For specific forms of dynamics, e.g., control-affine [49], and unlimited actuation bounds, this approach can be scalable to higher-dimensional systems and can be paired with an efficient online quadratic program for computing the instantaneous control [49]. Unfortunately, finding a valid control barrier function for a general system is a nontrivial task. Lyapunov-based methods [51, 52] suffer from the same limitation of requiring hand-crafted functions.

HJ reachability is a technique that uses continuous-time dynamic programming to directly compute a value function that captures the optimal safe control for a general nonlinear system [20, 80]. This method can provide hard safety guarantees for systems, subject to bounded uncertainties and disturbances. There are two major drawbacks to HJ reachability. The first is that the technique suffers from the curse of dimensionality and scales exponentially with number of states in the system. Because of this, the technique can only be used directly on systems of up to 4-5 dimensions [20]. When using specific dynamics formulations and/or restricted controllers, this upper limit can be extended [47, 133]. Second, because of this computational cost, the value function is typically computed offline based on assumed system dynamics and bounds on uncertainties. This can lead the safety analysis to be invalid or overly conservative.

There are many attempts in injecting some form of control theory into RL algorithms. In comparison to works that assume specific problem structure [49, 63] or existence of a nominal model [49, 23], our proposed approach is applicable to general nonlinear systems and does not require a model. But, we do assume access to a distance metric defined on the state space. Our primary inspiration is recent work by [81] that connects HJ reachability with RL and introduces a HJ Bellman update, which can be applied to deep Q-learning for safety analysis. This method loses hard safety guarantees due to the neural approximation, but enables scalable learning of safety value function. However, an agent trained using the method in [81] will focus exclusively on safety. Thus, we extend the method by formulating it within the CMDP framework, thereby enabling performance-driven learning.

6.3. Preliminaries

Constrained MDPs. The problem of RL with safety constraints is often formulated as a CMDP. On top of the MDP tuple $(\mathcal{X}, \mathcal{U}, R, \mathcal{F})$, where $\mathcal{X}$ is the state space, $\mathcal{U}$ is the action space, $\mathcal{F} : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$ characterizes the system dynamics, and $R : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ is the reward function, CMDP includes an additional set of cost functions, $\{C_1, \ldots, C_m\}$, where each $C_i : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ maps state-action transitions to costs characterizing constraint violations.

The objective of RL is to find a policy $\pi : \mathcal{X} \to \mathcal{P}(\mathcal{U})$ that maximizes the expected cumulative rewards, $V_R^\pi(x) = \mathbb{E}_{x_k, u_k \sim \pi}\left[\sum_{k=0}^{\infty} \gamma^k R(x_k, u_k) | x_0 = x\right]$, where $\gamma \in [0, 1)$ is a temporal discount factor. Similarly, the expected cumulative costs are defined as $V_{C_i}^\pi(x) = \mathbb{E}_{x_k, u_k \sim \pi}\left[\sum_{k=0}^{\infty} \gamma^k C_i(x_k, u_k) | x_0 = x\right]$; CMDP requires the policy to be feasible by imposing a limit for the costs, i.e., $V_{C_i}(\pi) \leq \chi_i, \forall i$. Putting everything together, the

RL problem in a CMDP is:

$$\pi^* = \arg\max_{\pi} \; V_R^{\pi}(x) \quad \text{s.t.} \quad V_{C_i}^{\pi}(x) \leq \chi_i \;\; \forall i. \tag{6.1}$$

HJ Reachability. To generate the safety constraint, one can apply HJ reachability to a general nonlinear system model, denoted as $\dot{x} = f(x, u)$. Here $x \in \mathbb{R}^n$ is the state, u is the control contained within a compact set $\mathcal{U}$. f is assumed uniformly continuous and bounded, and Lipschitz in x for all u. For discrete-time approximations, the time step $\Delta t > 0$ is used. We denote all allowable states as $\mathcal{K}$, for which there exists a terminal reward $l(x)$, such that $x \in \mathcal{K} \iff l(x) \geq 0$. An $l(x)$ that satisfies this condition is the signed distance to the boundary of $\mathcal{K}$. Taking autonomous driving as an example, $\mathcal{K}$ is the drivable area and $l(x)$ is the distance to road boundary or obstacle. This set $\mathcal{K}$ is the complement of the failure set that must be avoided. The goal of this HJ reachability problem is to compute a safety value function that maps a state to its safety value, with respect to $l(x)$, over time. This is done by capturing the minimum reward achieved over time by the system applying an optimal control policy:

$$V_S(x, T) = \sup_{u(\cdot)} \min_{t \in [0,T]} l(\xi_{x,T}^{u}(t)), \tag{6.2}$$

where ξ is the state trajectory, $T < 0$ is the initial time, and 0 is the final time. To solve for this safety value function, a form of continuous dynamic programming is applied backwards in time, from $t = 0$ to $t = T$, using the Hamilton-Jacobi-Isaacs Variational Inequality (HJI-VI):

$$\begin{aligned} \min\left\{ \frac{\partial V_S}{\partial t} + \max_{u \in \mathcal{U}} \langle f(x, u), \nabla V_S(x) \rangle, l(x) - V_S(x, t) \right\} = 0, \\ V_S(x, 0) = l(x). \end{aligned} \tag{6.3}$$

The super-zero level set of this function is called the reachable tube, and describes all states from which the system can remain outside of the failure set for the time horizon. For the infinite-time, if the limit exists, we define the converged value function as $V_S(x) = \lim_{T \to -\infty} V_S(x, T)$. While the HJI-VI is difficult to solve, once solved, safety verification only requires evaluating the safety value of the current state.

Once the safety value function is computed, the optimal safe control can be found online by solving the Hamiltonian: $\pi_S^*(x) = \arg\max_{u \in \mathcal{U}} \langle f(x, u), \nabla V_S(x) \rangle$. This safe control is typically applied in a least-restrictive way, wherein the safety controller becomes active only when the system approaches the boundary of the reachable tube, i.e., $u \sim \pi$ if $V_S(x, T) \geq 0$ and π_S^* otherwise.

The newly introduced discounted safety Bellman equation [81] modifies the HJI-VI in (6.3) in a time-discounted formulation for discrete time:

$$\begin{aligned} V_S(x) &= (1 - \gamma) l(x) + \gamma \min\left\{ l(x), \max_{u \in \mathcal{U}} V_S(x + f(x, u)\Delta t) \right\}, \\ V_S(x, 0) &= l(x). \end{aligned} \tag{6.4}$$

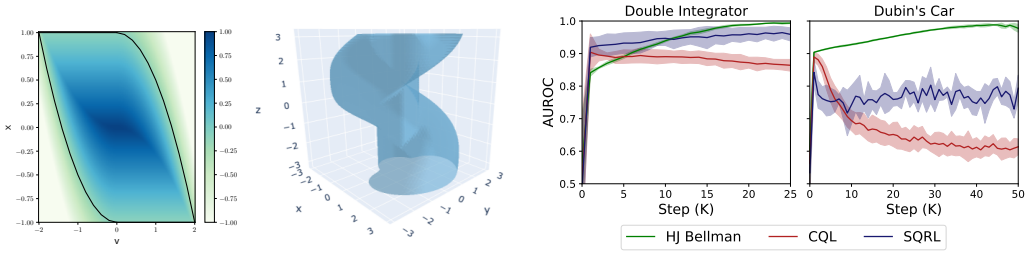
Algorithm 4: SAGE-Environment Interaction

```
Initialize: Performance actor π and critic Q;
Initialize: Safety actor π_S and critic Q_S;
for i = 0, ..., # Episodes do
    x = env.reset()
    while not terminal do
        u ~ π(x);
        // The safe actor intervenes when the current state-action is deemed
          unsafe by the safety critic.
        if Q_S(x, u) < ϵ then
            u ~ π_S(x)
        end
        x', r = env.step(u)
        // Update performance actor-critic and safety actor-critic. See
          Appendix A.2 for details.
    end
end
```

This formulation induces a contraction mapping, which enables convergence of the value function when applied to dynamic programming schemes, commonly used in RL.

6.4. Safe Autonomous racinG on Ego-vision

In this section, we describe our framework for safety-aware autonomous racing. We are inspired by guaranteed-safe methods, such as HJ reachability, which provides a systematic way to verify safety. Thus, we formulate our problem as a combination of constrained RL and HJ reachability theory, adopting a control-theoretic approach to learn safety. Building upon prior work on neural approximation of HJ Reachability [81], we demonstrate that it is possible to directly update the safety value function on high-dimensional sensory input, thereby expanding the scope of applications to problems previously inaccessible. We highlight the notable aspects of our framework:
i) HJ reachability provides a control-theoretic and low-latency way to verify safety. By incorporating HJ Reachability theory in the CMDP framework, we have a control-theoretic update rule to learn about safety and can verify safety by evaluating the safety value of the current state. Another positive outcome of the formulation is that the original constrained problem is decomposed into two unconstrained optimization problems, making our formulation more amenable to gradient-based learning.
ii) Scales to high-dimensional visual context. Compared to standard HJ Reachability methods, whose computational complexity scales exponentially with the state dimension, we updated the safety value directly on vision embedding, with neural approximation. This is the highest-dimensional problem studied studied via HJ reachability to-date.
Problem formulation. We inject HJ reachability theory into the CMDP framework. Starting with Eqn. 6.1, we can interpret the negative of a cost as a reward for safety



(a) Double Integrator (b) Dubins' Car (c) Performance comparison of learning rules (averaged over 5 random seeds)

Figure 6.2: We use two classical control benchmarks, *double integrator* and *Dubins' car*, to evaluate the performance of different learning rules for safety analysis. (a) shows the safety value function of the double integrator and the black line delineates $V_S(x) = 0$, within which the particle can remain within the allowable range of $x \in [-1, 1]$. (b) shows the iso-surface of the safety value function at 0, i.e., $V_S(x) = 0$, for Dubins' Car, within which the car can reach a unit circle at the origin. The performance comparison is summarized in (c).

and, without loss of generality, reverse the direction of the inequality constraint. Recall that the super-zero level set of the safety value function, i.e., $\{x|V_s(x) \geq 0\}$, designates all states from which the system can remain within the set of allowable states, $\mathcal{K}$, over infinite time horizon. Thus, the safety value function derived from HJ Reachability can be plugged into CMDP (Eqn. 6.5):

$$\pi^* = \arg\max_{\pi} \; V_R^{\pi}(x), \quad \text{s.t.} \quad V_S(x) \geq \epsilon, \tag{6.5}$$

where $\epsilon \geq 0$ is introduced as a safety margin. A key difference from the original CMDP formulation (Eqn. 6.1) is that constraint satisfaction, $V_S(x) \geq \epsilon$, no longer depends on the policy, π. Thus, we can bypass the challenges of solving CMDPs (Section 6.2) and decompose learning under safety constraints into optimizing for performance and updating safety value estimation. While a number of works have similar dual-policy architecture [49, 23, 201], ours design is informed by HJ Reachability theory. Another difference is that HJ Reachability considers safety as absolute, and there is no mechanism to allow for some level of safety infraction, and thus χ_i in Eqn. 6.1 is not longer present.

Update of Safety Critic. We apply HJ Bellman update, in place of standard Bellman backup, to learn the safety value function. The learning rule proposed by [81] is defined on discrete action space, which we modify for continuous action space (Eqn. 6.6). While the safety actor is sub-optimal during learning, the resulting HJ Bellman target is a conservative estimation of the safety value, as $Q_S(x', u') \leq \max_{u' \in \mathcal{U}} Q_S(x', u')$, which is desirable for safety analysis. $Q_S(x, u)$ is updated model-free using state-action transitions, and only additionally requires $l(x)$. We assume $l(x)$ can be acquired from the vehicle's sensing capability [27] or estimated from perception [45].

$$\begin{aligned} Q_S(x, u) &= (1 - \gamma) l(x) + \gamma \min\{l(x), Q_S(x', u')\}, \\ u' &\sim \pi_S(x'). \end{aligned} \tag{6.6}$$

SAGE. We propose SAGE, which consists of a performance policy and a safety policy. The safety backup controller is applied in a least restrictive way, only intervening when the RL agent is about to enter into an unsafe state, i.e., $u \sim \pi$, if $Q_S(x,u) \geq \epsilon$ and $u \sim \pi_S$ otherwise. The performance policy may be implemented with any RL algorithm. Since we expect the majority of samples to be from the performance policy, it is more appropriate to update the safety actor critic with an off-policy algorithm. In this work, we base our implementation of the safety actor critic on soft-actor critic (SAC) haarnoja2018soft. The safety critic is updated with Eqn. 6.6, and the safety actor π_S parametrized by θ_θ is updated via Eqn. 6.7, where η is the learning rate.

$$\theta_S \leftarrow \theta_S + \eta \mathbb{E}_{x,u\sim\mathcal{D}} \nabla_u Q_S(x,u) \nabla_\theta \pi_{\theta_S}(x). \tag{6.7}$$

Algorithm 4 provides an overview for `SAGE` and a detailed version is presented in Appendix A.2.

6.5. Experiments

We evaluate `SAGE` on three sets of benchmarks, of increasing difficulty. While the our intended application is autonomous racing, the first two set of benchmarks can be considered as some abstraction of vehicles with the objective of avoiding obstacles and/or moving towards goals. Firstly, we evaluate on two classical control tasks, where the safe vs. unsafe states are known analytically, and we compare the HJ Bellman update used in `SAGE` to alternatives for learning safety critics in the literature. Secondly, we compare `SAGE` to constrained RL baselines in Safety Gym. Finally, we challenge `SAGE` in `Learn-to-Race` and conduct ablation to better understand how different components of `SAGE` contribute to its performance.

6.5.1. Experiment: Classical Control Benchmarks

As mentioned earlier, safety critics have been trained in other works [26, 194] with different learning rules. The objective here is to compare the HJ Bellman update with alternatives. Thus, we focus on safety analysis with off-policy samples, and evaluate on two classical control benchmarks *Double Integrator* [81] and *Dubins' Car* [20], where the safe / lively[24] states (Figure 6.2a and 6.2b) and the optimal safety controller are known analytically. Double Integrator characterizes a particle moving on the x-axis, with velocity v. By controlling the acceleration, the objective is to keep the particle on a bounded range on x-axis. Dubins' Car is a simplified car model, where the car moves at a constant speed. By controlling the turning rate, the goal is to reach a unit circle regardless of the heading. More information on the two tasks are provided in Appendix A.1.1.

In this experiment, we generate state-action pairs with a random policy, and evaluate the safety value function with respect to the optimal safety controller, π_S^*. In both *Safety Q-functions for RL* (SQRL) [194] and the *Conservative Safety Critic* (CSC) [26],

[24]Liveness refers the ability to reach the specified goal [110].

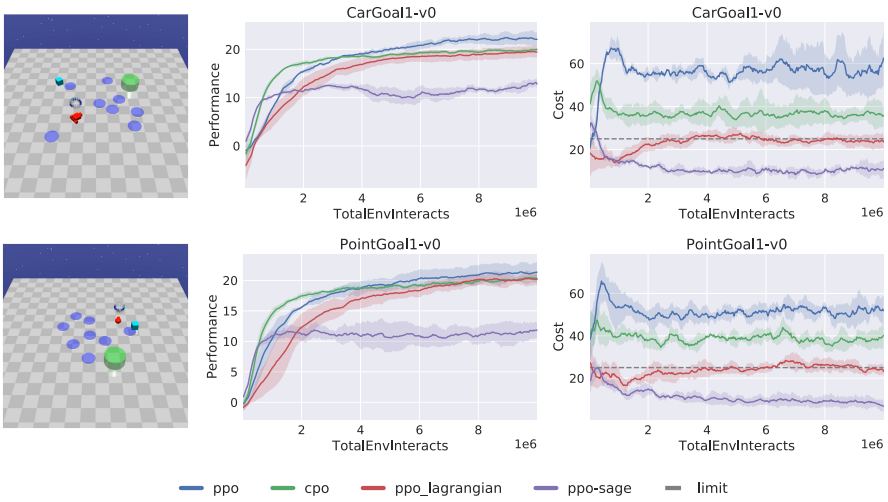


Figure 6.3: Performance of SAGE with comparison to baselines in the CarGoal1-v0 (top row) and PointGoal1-v0 (bottom row) benchmarks (averaged over 5 random seeds). In Goal tasks, agents must navigate to observed goal locations (indicated by the green regions), while avoiding obstacles (e.g., vases in cyan, and hazards in blue).

the safety value function is defined as the expected cumulative cost, i.e., $Q_C^\pi(x,u) \coloneqq \mathbb{E}_{x_k,u_k\sim\pi}\left[\sum_{k=0}^{\infty}\gamma^k C(x_k)|x_0=x,u_0=u\right]$, where $C(x_k)=1$ if a failure occurs at x_k and 0 otherwise. In this case, both the environment and optimal safety policy are deterministic. Thus, by definition, $Q_C^{\pi_S^*}(x,\pi_S^*(x))$ should be 0 if x is a safe state. SQRL uses the standard Bellman backup to propagate the failure signal. On top of that, CSC uses conservative Q-learning (CQL) [137] to correct for difference between the behavior policy, i.e., the random policy, and the evaluation policy, i.e., the optimal safety policy, and overestimates Q_C to err on the side of caution.

Since the safe vs. unsafe states are known for these benchmark tasks, we can directly compare the performance of these safety critics learned with different learning rules (Figure 6.2c). While the theoretical cut-off for safe vs. unsafe states is 0, the performance of SQRL is sensitive to the choice of the cut-off; thus, we report AUROC instead. For both CQL and SQRL, we do a grid search around the hyperparameters used in the original papers and report the best results. The implementation details and additional results are included in Appendix A.1.2. Directly applying Bellman update for safety analysis, as in SQRL, performs reasonably well on *Double Integrator*, but not on the more challenging *Dubins' Car*. In our experiment, CQL consistently under-performs SQRL. In comparison, HJ Bellman update has AUROC close to 1 on both tasks and has very small variance over different runs. It is worth-noting that the result with the HJ Bellman update is achieved without explicitly addressing the distribution mismatch [208], which typically challenges off-policy evaluation problems. This experiment only compares the efficacy of the different learning rules for safety critic given the same off-

(a) Aerial (b) Third-person (c) Ego-view

Figure 6.4: We use the `Learn-to-Race` (`L2R`) framework [107] for evaluation; this environment provides simulated racing tracks that are modeled after real-world counterparts, such as the famed Thruxton Circuit in the UK (`Track01:Thruxton`, (a)). Here, learning-based agents can be trained and evaluated according to challenging metrics and realistic vehicle and environmental dynamics, making `L2R` a compelling target for safe reinforcement learning. Each track features challenging components for autonomous agents, such as sharp turns (shown in (b)), where SAGE only uses ego-camera views (shown in (c)) and speed.

policy samples, and does not intend to compare other aspects of SQRL and CSC.

One caveat is that SQRL and CQL uses a binary signal for failures, while HJ Bellman update has access to the distance, $l(x)$. On one hand, HJ Bellman update does assume more information. On the other hand, it may be more practical to learn safety from distance measurements than experiencing failures. Applied to autonomous driving, this translates to learning to avoid obstacles from distance measurements that are becoming prevalent on cars with assisted driving capabilities [27], in comparison to experiencing collisions.

6.5.2. Experiment: `Safety Gym`

We additionally evaluate our proposed approach, SAGE, in Safety Gym [181]. Specifically, we evaluate on the standard CarGoal1-v0 and PointGoal1-v0 benchmarks, where the agent navigates to a goal while avoiding hazards. We compare SAGE against baselines including: Constrained Policy optimization (CPO) achiam2017constrained, an unconstrained RL algorithm (Proximal Policy optimization (PPO) schulman2017proximal), and its Lagrangian variant (PPO-Lagrangian). By default, distance measurements from LiDAR are available to all baselines in these benchmarks, and thus SAGE has direct access to $l(x)$. Episodic Performance and Cost curves are shown in Figure 6.3 and implementation details are included in Appendix A.3.

PPO-SAGE has significantly fewer constraint violations, compared to other baselines, and the number of violations decreases over time. While CPO and PPO-Lagrangian take into account that a certain number of violations are permissible, there is no such mechanism in SAGE, as HJ Reachability theory defines safety in an absolute sense. While the inability to allow for some level of safety infractions, unfortunately, compromises performance, SAGE learns mature obstacle-avoidance behaviors, compared to some policies, which may ignore traps in favor of fast navigation to goal locations. Violations that do

occur in SAGE result from neural approximation error, and the number of violations decreases over time as the safety actor-critic gains experience, despite the randomized and constantly-changing episodic layouts.

6.5.3. Experiment: `Learn-to-Race`

Task Overview. In this paper, we evaluate our approach using the Arrival Autonomous Racing Simulator, through the newly-introduced and OpenAI-gym compliant `Learn-to-Race` (`L2R`) task and evaluation framework herman2021learn. `L2R` provides multiple simulated racing tracks, modeled after real-world counterparts, such as Thruxton Circuit in the UK (`Track01:Thruxton`; see Figure 6.4). `L2R` can provide access to RGB images from any specified location, semantic segmentation, and vehicle states (e.g., pose, velocity). In each episode, an agent is spawned on the selected track. At each time-step, it uses its observations to determine normalized steering angle and acceleration. All learning-based agents receive the reward specified by `L2R`, which is formulated as a weighted sum of reward for driving fast and penalty for leaving the drivable area; the main objective is to complete laps in as little time as possible. Additional metrics are defined to evaluate driving quality.

Implementation Details. To characterize the performance of our approach, we report results on the Average Speed and the Episode Completion Percentage (*ECP*) metrics herman2021learn as proxies for agent performance and safety, respectively. We report other metrics defined by `L2R` in Appendix A.6.

We use `Track01:Thruxton` in `L2R` (Fig. 6.4) for all stages of agent interaction with the environment. During training, the agent is spawned at random locations along the race track and uses a stochastic policy. During evaluation, the agent is spawned at a fixed location and uses a deterministic policy. The episode terminates when the agent successfully finishes a lap, leaves the drivable area, collides with obstacles, or does not progress for a number of time-steps. For each agent, we report averaged results across 5 random seeds, evaluated every 5000 steps over an episode (one lap). We use SAC as the performance policy, and all agents only have access to ego-camera view (Figure 6.4c) and speed, unless specified otherwise. The implementation, including network architecture and hyperparameters, are detailed in Appendix A.5.

Ablation Study. To demonstrate the benefit of utilizing domain knowledge in the form of a nominal model, we use a kinematic bike model [132] to calculate the safety value and derive the corresponding safety controller, detailed in Appendix in A.4. We refer to this as the *static safety actor-critic*. In all our experiments, only the static safety actor-critic has access to vehicle pose, i.e., location and heading. We evaluate the performance of this static safety actor-critic by coupling a random agent with it (`SafeRandom`). We test `SafeRandom` on a series of safety margins to account for unmodelled dynamics; the performance averaged over 10 random seeds is summarized in Figure A.9. For instance, $\epsilon \geq 4.2$ achieves 80+% ECP, in comparison to 0.5% ECP by `Random` agent.

We examine the effect of having a safety controller, by comparing `SAC` with an instance of itself that is coupled with a static safety actor-critic (`SafeSAC`). We set the safety margin ϵ to be 4.2, based on empirical results from `SafeRandom`. We also compare the

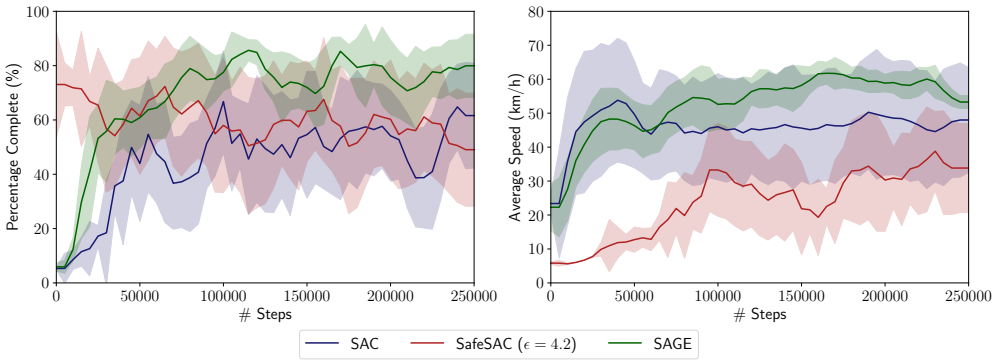


Figure 6.5: **Left:** Episode percent completion and **Right:** speed evaluated every 5000 steps over an episode (a single lap) and averaged over 5 random seeds. Results reported based on `Track01:Thruxton` in `L2R`.

performance of using the static safety actor-critic (`SafeSAC`) and a dynamically-updating one (`SAGE`). Since the `SAGE` agent is expected to have a better characterization of the safety value, the agent no longer depends on a large safety margin to remain safe and thus `SAGE` uses a safety margin of 3.0m, which accounts for the vehicle dimensions[25]. We also report results of `SafeSAC` with the same safety margin in in Appendix A.6.

Results. The performance comparison between different agents is summarized in Figure 6.5. For reference, a single lap in `Track01:Thruxton` is 3.8km, whereas CARLA, the *de facto* environment for urban driving research, has in total 4.3km drivable roads in the original benchmark [55]. Thus, successfully completing an episode, i.e., a lap, is quite challenging. *The static safety actor-critic significantly boosts initial safety performance.* With the help of the static safety actor-critic, the `SafeSAC` can complete close to 80% of a lap, in comparison to slightly more than 5% with `SAC`. This, again, showcases the benefit of injecting domain knowledge in the form of a nominal model. However, there are two notable limitations with the static safety controller. Firstly, it is extremely conservative, hard-braking whenever the vehicle is less safe. As a result, the `SafeSAC` agent has an initial speed of less than 10km/h. Secondly, as the `SAC` learns to avoid activating the safety controller and drive faster, the static safety controller is no longer able to recover the vehicle from marginally safe states. In fact, by applying the 'optimal' safety action from Eqn. A.9, i.e., maximum brake and steer towards centerline, the vehicle will lose traction and spin out of control. As a result, the ECP actually decreases over time for `SafeSAC`.

SAGE learns safety directly from vision context and can recover from marginally safe states more smoothly. Having a safety actor-critic that is dedicated to learning about safety significantly boosted the initial safety performance of `SAGE`, in comparison to the `SAC` agent, even though both the performance and the safety actor-critics are ran-

[25]The HJ reachable tube is computed with respect to the back axle of the vehicle and does not account for the physical dimension of the vehicle. Using the car length as the safety margin is a rough engineering estimate.

domly initialized. Moreover, this shows that the safety value function can be learned from scratch on vision embedding. In practice, we envision the safety actor-critic to be warm-started with the nominal model or observational data, and fine-tuned by interactions with the environment. Furthermore, the learnable safety actor-critic can recover from marginally safe states smoothly, avoiding the two undesirable behaviors from the static safety actor-critic. A qualitative comparison of such behaviors is available at the anonymized paper website. While `SAGE` outperforms other baselines, there is still significant performance gap with human, as the speed record at Thruxton Circuit is 237 km/h (average speed).

6.6. Conclusion

In this paper, we propose `SAGE` for end-to-end autonomous racing, which can learn to identify unsafe states from ego-camera views and recover from unsafe states, despite the complex dynamics with unstable regimes. We demonstrate on two classical control benchmarks that the HJ Bellman update is more effective than alternatives for learning the safety critic. Compared to constrained RL baselines in the Safety Gym, we show that `SAGE` has significantly fewer constraint violations. We report the new state-of-the-art result on `Learn-to-Race`, and we demonstrate that the safety value can be learned directly on visual context, thereby expanding HJ reachability to broader applications.

Throughout our experiments, we find it is highly effective to inject domain knowledge, in the form of nominal model or control-theoretic learning rule. In our experiments, the safety actor-critics were randomly initialized. But, in practice, we expect it to be pre-trained with a nominal model and/or observational data, prior to interacting with the environment. While neural approximation enables us to scale HJ reachability to high-dimensional visual inputs, we unfortunately lose the hard guarantees on safety. An important next-step is to characterize neural approximation error and find ways to retain the safety guarantees with function approximators.

7. Conclusions

In this section, we first recap the contributions and practical implications of individual chapters, echoing the research vision in Section 1.2. Then, we point out some directions for future work.

7.1. Contributions and Practical Implications

Gnu-RL: Buildings account for about 40% of the total energy in the United States [141], and it is estimated that up to 30% of that energy usage may be reduced through advanced sensing and control strategies [92, 78]. However, as discussed in Section 1.4.1, MPC is not scalable due to the heterogeneity of building environments, and RL is not practical for real-world building systems due to its sample complexity. To address *Research Question 1*, we combine the strength of MPC and RL, and propose a control strategy that is both practical and scalable. Gnu-RL can be deployed on real-world building systems with satisfactory initial performance, using no prior knowledge other than historical data. In our experiments, both in simulation and on a real-world testbed, we show Gnu-RL improve energy efficiency over time, while maintaining a satisfactory level of thermal comfort.

COHORT: COHORT is a practical, scalable, and versatile solution to coordinate a population of TCLs, which account for 20% of the electricity consumption in the United States, to jointly provide grid services. COHORT can incorporate detailed, system-specific models and constraints from individual TCLs, and at the same time is scalable to large population sizes. Furthermore, COHORT is computationally-scalable to long planning horizons, which unlocks the potential to shift TCLs over extended period of time, which is identified as the most promising use case for these loads by [92, 135, 186]. As a result, COHORT is not only applicable to *reference tracking*, but also load shifting use cases, such as *ramping minimization* and *peak load curtailment*. Finally, we validated COHORT is practical for real-world systems via a hardware-in-the-loop simulation.

PPOF: PROF is a method to flexibly enforce convex operational constraints within neural policies, in such a way that the constraints are taken into account during the learning process. Over-voltage has already become a common occurrence in areas with high renewable penetration. In the experiment on a distribution network, PROF, controlling inverter setpoints, satisfies the voltage constraints 100% of the time, in comparison to 22% over-voltage violations incurred by a Volt/Var control baseline. Voltage support from inverters increase the hosting capacity of the existing networks and reduce the curtailment of renewable generation. Furthermore, as the renewable energy resources gradually replace fossil fuel ones over the course of coming decades, a learning-based control strategy enables the inverter to adapt to the transitioning power grid autonomously. At the same time, embedding safety guarantees in a learning-based solution instill confidence in industry practitioners.

SAGE: We propose SAGE by incorporating HJ reachability theory, a safety verification method for non-linear systems, into the CMDP framework, as an approach to enforce safety constraints in performance-driven learning. Though HJ reachability is traditionally not scalable to high-dimensional systems, we demonstrate that with neural approximation, the HJ safety value can be learned directly on vision context—the highest-dimensional problem studied via the method, to-date. We evaluate our method on several benchmark tasks, including `Safety Gym` and `Learn-to-Race` (`L2R`), a recently-released high-fidelity autonomous racing environment. Our approach has significantly fewer constraint violations in comparison to other constrained RL baselines in `Safety Gym`, and achieves the new state-of-the-art results on the `L2R` benchmark task.

7.2. Future Work

Here, we also summarize some possible directions for future work, on top of the more specific research directions at the end of each chapter.

Recent developments in offline RL present new ways to utilize historical data. Existing buildings can be expected to have a large amount of historical data stored in the BAS. How to best learn from the historical data to improve the current operation is a pivotal question. In Gnu-RL, one of the key ideas is to apply behavior cloning on historical data to warm-start the agent. Behavior cloning is a classical idea that formulates the optimal control task as a supervised learning task, assuming the historical data is generated by an expert. Recent developments in offline RL [140] present a new paradigm for utilizing the same historical data. In the case that the behavior policy is sub-optimal, as in the case of hysteresis or PID control, offline RL should have superior performance compared to behavior cloning on the same data based on the analysis in [136]. On the other hand, behavior cloning is easy to implement, and potentially more attractive from an engineering perspective. Thus, further research is needed to understand what technique is most appropriate for utilizing historical building data to improve current operation.

Being able to enforce safety constraints will be paramount to the real-world success of learning-based methods in energy systems. While we examined approaches to enforce device-level and network-level constraints, arising from human needs or physical necessity in this thesis, we believe this is only the start of a broader conversation on closely integrating domain knowledge, and control theory into learning-based methods. We hope that this thesis will serve to spark further inquiry into this important line of work. There is growing urgency to curb carbon emissions by improving the operation of existing energy systems, along with other approaches to jointly combat climate change. Embedding safety guarantees in learning-based solutions is essential to instill confidence in industry practitioners and enable widespread adoption.

How to retain hard safety guarantees from HJ reachability theory, while scaling to high-dimensional systems with function approximator? While HJ reachability theory

is a powerful tool for safety analysis that provides hard safety guarantees for general non-linear systems, subject to bounded disturbances [20, 80], a major drawback of the approach is that its computational complexity scales exponentially with the number of states in the system. Thus, there is strong incentive in using function approximation to scale to high-dimensional autonomous systems. Numerous attempts [8, 21, 187, 80, 110, 41] have been made in developing methods to learn the value function with neural networks. However, these methods, unfortunately, lose the safety guarantees, except under restricted settings.

The outstanding question here is whether it is possible to retain some form of safety guarantees while using function approximation, e.g., neural networks, which is necessary to scale to high-dimensional systems.

References

[1] California ISO - Managing oversupply. Accessed: 2020-03-21.

[2] Achiam, J. Spinning Up in Deep Reinforcement Learning.

[3] Achiam, J., Held, D., Tamar, A., and Abbeel, P. Constrained Policy Optimization. In *Proceedings of the 34th International Conference on Machine Learning* (2017).

[4] Achiam, J., Held, D., Tamar, A., and Abbeel, P. Constrained policy optimization. In *International Conference on Machine Learning* (2017), PMLR, pp. 22–31.

[5] Agbi, C., Song, Z., and Krogh, B. Parameter identifiability for multi-zone building models. In *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)* (2012), IEEE, pp. 6951–6956.

[6] Agrawal, A., Amos, B., Barratt, S., Boyd, S., Diamond, S., and Kolter, J. Z. Differentiable Convex Optimization Layers. In *Advances in Neural Information Processing Systems* (2019), pp. 9558–9570.

[7] Akametalu, A. K., Kaynama, S., Fisac, J. F., Zeilinger, M. N., Gillula, J. H., and Tomlin, C. J. Reachability-based safe learning with Gaussian processes. In *53rd IEEE Conference on Decision and Control, CDC 2014* (2014).

[8] Akametalu, A. K., and Tomlin, C. J. Temporal-difference learning for online reachability analysis. In *2015 European Control Conference (ECC)* (2015), IEEE, pp. 2508–2513.

[9] Akorede, M. F., Hizam, H., and Pouresmaeil, E. Distributed energy resources and benefits to the environment. *Renewable and sustainable energy reviews 14*, 2 (2010), 724–734.

[10] Altman, E. *Constrained Markov Decision Processes*, vol. 7. CRC Press, 1999.

[11] Ames, A. D., Coogan, S., Egerstedt, M., Notomista, G., Sreenath, K., and Tabuada, P. Control barrier functions: Theory and applications. In *2019 18th European Control Conference (ECC)* (2019), IEEE, pp. 3420–3431.

[12] Amos, B., Jimenez, I., Sacks, J., Boots, B., and Kolter, J. Z. Differentiable MPC for End-to-end Planning and Control.

[13] Amos, B., and Kolter, J. Z. Optnet: Differentiable optimization as a layer in neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70* (2017), JMLR. org, pp. 136–145.

[14] ASHRAE, A. Ashrae 62.1-2010 ventilation for acceptable indoor air quality. *Atlanta, GA* (2010).

[15] Aswani, A., Master, N., Taneja, J., Culler, D., and Tomlin, C. Reducing transient and steady state electricity consumption in hvac using learning-based model-predictive control. *Proceedings of the IEEE 100*, 1 (Jan 2012), 240–253.

[16] Atam, E., and Helsen, L. Control-oriented thermal modeling of multizone buildings: methods and issues: intelligent control of a building system. *IEEE Control Systems Magazine 36*, 3 (2016), 86–111.

[17] Bai, S., Kolter, J. Z., and Koltun, V. Deep equilibrium models. *arXiv preprint arXiv:1909.01377* (2019).

[18] Baker, K., Bernstein, A., Dall'Anese, E., and Zhao, C. Network-cognizant voltage droop control for distribution grids. *IEEE Transactions on Power Systems 33*, 2 (2017), 2098–2108.

[19] Bank, J., and Hambrick, J. Development of a high resolution, real time, distribution-level metering system and associated visualization modeling, and data analysis functions. National Renewable Energy Laboratory, Tech. Rep. NREL/TP-5500-56610.

[20] Bansal, S., Chen, M., Herbert, S., and Tomlin, C. J. Hamilton-jacobi reachability: A brief overview and recent advances. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)* (2017), IEEE, pp. 2242–2253.

[21] Bansal, S., and Tomlin, C. J. Deepreach: A deep learning approach to high-dimensional reachability. In *2021 IEEE International Conference on Robotics and Automation (ICRA)* (2021), IEEE, pp. 1817–1824.

[22] Basso, T. In *IEEE 1547 and 2030 Standards for Distributed Energy Resources Interconnection and Interoperability with the Electricity Grid* (National Renewable Energy Laboratory, 2014).

[23] Bastani, O. Safe reinforcement learning with nonlinear dynamics via model predictive shielding. In *2021 American Control Conference (ACC)* (2021), IEEE, pp. 3488–3494.

[24] Bengea, S., Kelman, A., Borrelli, F., Taylor, R., and Narayanan, S. Model predictive control for mid-size commercial building hvac: Implementation, results and energy savings. In *Second international conference on building energy and environment* (2012), pp. 979–986.

[25] Berkenkamp, F., Turchetta, M., Schoellig, A. P., and Krause, A. Safe Model-based Reinforcement Learning with Stability Guarantees. In *Advances in Neural Information Processing Systems* (2017).

[26] Bharadhwaj, H., Kumar, A., Rhinehart, N., Levine, S., Shkurti, F., and Garg, A. Conservative safety critics for exploration. *arXiv preprint arXiv:2010.14497* (2020).

[27] BMW. Automotive sensors – the sense organs of driver assistance systems, Sep 2021.

[28] Bolognani, S., and Dörfler, F. Fast power system analysis via implicit linearization of the power flow manifold. In *2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)* (2015), IEEE, pp. 402–409.

[29] Boyd, S., Parikh, N., and Chu, E. *Distributed optimization and statistical learning via the alternating direction method of multipliers*. Now Publishers Inc, 2011.

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

[31] Burger, E. M., and Moura, S. J. Generation following with thermostatically controlled loads via alternating direction method of multipliers sharing algorithm. *Electric Power Systems Research 146* (2017), 141–160.

[32] Burke, W., and Auslander, D. Low-frequency pulse width modulation design for hvac compressors. In *ASME 2009 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference* (2009), American Society of Mechanical Engineers Digital Collection, pp. 291–297.

[33] Cai, J., Kim, D., Braun, J. E., and Hu, J. Optimizing zone temperature setpoint excitation to minimize training data for data-driven dynamic building models. In *2016 American Control Conference (ACC)* (2016), IEEE, pp. 1478–1484.

[34] Cai, P., Wang, H., Huang, H., Liu, Y., and Liu, M. Vision-based autonomous car racing using deep imitative reinforcement learning. *IEEE Robotics and Automation Letters 6*, 4 (2021), 7262–7269.

[35] Callaway, D. S., and Hiskens, I. A. Achieving controllability of electric loads. *Proceedings of the IEEE 99*, 1 (2010), 184–199.

[36] Chang, Y.-C., Roohi, N., and Gao, S. Neural Lyapunov Control. In *Advances in Neural Information Processing Systems* (2019), pp. 3245–3254.

[37] Chatzos, M., Fioretto, F., Mak, T. W., and Van Hentenryck, P. High-fidelity machine learning approximations of large-scale optimal power flow. *arXiv preprint arXiv:2006.16356* (2020).

[38] Chen, B., Cai, Z., and Bergés, M. Gnu-RL: A precocial reinforcement learning solution for building HVAC control using a differentiable mpc policy. In *Proceedings of the 6th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation* (2019), pp. 316–325.

[39] Chen, B., Cai, Z., and Bergés, M. Gnu-rl: A practical and scalable reinforcement learning solution for building hvac control using a differentiable mpc policy. *Frontiers in Built Environment* (2020), 174.

[40] Chen, B., Donti, P. L., Baker, K., Kolter, J. Z., and Bergés, M. Enforcing policy feasibility constraints through differentiable projection for energy optimization. In *Proceedings of the Twelfth ACM International Conference on Future Energy Systems* (2021), pp. 199–210.

[41] Chen, B., Francis, J., Oh, J., Nyberg, E., and Herbert, S. L. Safe autonomous racing via approximate reachability on ego-vision, 2021.

[42] Chen, B., Francis, J., Pritoni, M., Kar, S., and Berg'es, M. Cohort: Coordination of heterogeneous thermostatically controlled loads for demand flexibility. In *Proceedings of the 7th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation* (2020), pp. 31–40.

[43] Chen, B., Jin, M., Wang, Z., Hong, T., and Bergés, M. Towards off-policy evaluation as a prerequisite for real-world reinforcement learning in building control. In *Proceedings of the 1st International Workshop on Reinforcement Learning for Energy Management in Buildings amp; Cities* (New York, NY, USA, 2020), RLEM'20, Association for Computing Machinery, p. 52–56.

[44] Chen, B., Jin, M., Wang, Z., Hong, T., and Bergés, M. Towards off-policy evaluation as a prerequisite for real-world reinforcement learning in building control. In *Proceedings of the 1st International Workshop on Reinforcement Learning for Energy Management in Buildings & Cities* (2020), pp. 52–56.

[45] Chen, C., Seff, A., Kornhauser, A., and Xiao, J. Deepdriving: Learning affordance for direct perception in autonomous driving. In *Proceedings of the IEEE international conference on computer vision* (2015), pp. 2722–2730.

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

[47] Chen, M., Herbert, S. L., Vashishtha, M. S., Bansal, S., and Tomlin, C. J. Decomposition of reachable sets and tubes for a class of nonlinear systems. *IEEE Transactions on Automatic Control 63*, 11 (2018), 3675–3688.

[48] Chen, R. T., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. Neural ordinary differential equations. In *Advances in neural information processing systems* (2018), pp. 6571–6583.

[49] Cheng, R., Orosz, G., Murray, R. M., and Burdick, J. W. End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks. In *Proceedings of the AAAI Conference on Artificial Intelligence* (2019), vol. 33, pp. 3387–3395.

[50] Chisari, E., Liniger, A., Rupenyan, A., Van Gool, L., and Lygeros, J. Learning from simulation, racing in reality. In *2021 IEEE International Conference on Robotics and Automation (ICRA)* (2021), IEEE, pp. 8046–8052.

[51] Chow, Y., Nachum, O., Duenez-Guzman, E., and Ghavamzadeh, M. A lyapunov-based approach to safe reinforcement learning. *arXiv preprint arXiv:1805.07708* (2018).

[52] Chow, Y., Nachum, O., Faust, A., Duenez-Guzman, E., and Ghavamzadeh, M. Lyapunov-based safe policy optimization for continuous control. *arXiv preprint arXiv:1901.10031* (2019).

[53] Chua, K., Calandra, R., McAllister, R., and Levine, S. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds. Curran Associates, Inc., Red Hook, NY, 2018, pp. 4754–4765.

[54] Codevilla, F., Müller, M., López, A., Koltun, V., and Dosovitskiy, A. End-to-end driving via conditional imitation learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)* (2018), IEEE, pp. 4693–4700.

[55] Codevilla, F., Santana, E., López, A. M., and Gaidon, A. Exploring the limitations of behavior cloning for autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2019), pp. 9329–9338.

[56] Cole, W. J., Rhodes, J. D., Gorman, W., Perez, K. X., Webber, M. E., and Edgar, T. F. Community-scale residential air conditioning control for effective grid management. *Applied Energy 130* (2014), 428–436.

[57] Condat, L. A direct algorithm for 1-d total variation denoising. *IEEE Signal Processing Letters 20*, 11 (2013), 1054–1057.

[58] Costanzo, G. T., Gehrke, O., Bondy, D. E. M., Sossan, F., Bindner, H., Parvizi, J., and Madsen, H. A coordination scheme for distributed model predictive control: Integration of flexible ders. In *IEEE PES ISGT Europe 2013* (2013), IEEE, pp. 1–5.

[59] Costanzo, G. T., Iacovella, S., Ruelens, F., Leurs, T., and Claessens, B. J. Experimental analysis of data-driven control for a building heating system. *Sustainable Energy, Grids and Networks 6* (2016), 81–90.

[60] Creutzig, F., Agoston, P., Goldschmidt, J. C., Luderer, G., Nemet, G., and Pietzcker, R. C. The underestimated potential of solar energy to mitigate climate change. *Nature Energy 2*, 9 (2017), 1–9.

[61] Dalamagkidis, K., Kolokotsa, D., Kalaitzakis, K., and Stavrakakis, G. S. Reinforcement learning for energy conservation and comfort in buildings. *Building and environment 42*, 7 (2007), 2686–2698.

[62] de Avila Belbute-Peres, F., Smith, K., Allen, K., Tenenbaum, J., and Kolter, J. Z. End-to-end differentiable physics for learning and control. In *Advances in Neural Information Processing Systems* (2018), pp. 7178–7189.

[63] Dean, S., Tu, S., Matni, N., and Recht, B. Safely learning to control the constrained linear quadratic regulator. In *2019 American Control Conference (ACC)* (2019), IEEE, pp. 5582–5588.

[64] Deru, M., Field, K., Studer, D., Benne, K., Griffith, B., Torcellini, P., Liu, B., Halverson, M., Winiarski, D., Rosenberg, M., et al. US Department of Energy commercial reference building models of the national building stock.

[65] Djolonga, J., and Krause, A. Differentiable Learning of Submodular Models. In *Advances in Neural Information Processing Systems* (2017), pp. 1013–1023.

[66] Dobbe, R., Hidalgo-Gonzalez, P., Karagiannopoulos, S., Henriquez-Auba, R., Hug, G., Callaway, D. S., and Tomlin, C. J. Learning to control in power systems: Design and analysis guidelines for concrete safety problems. *Electric Power Systems Research 189* (2020), 106615.

[67] Donti, P. L., Roderick, M., Fazlyab, M., and Kolter, J. Z. Enforcing robust control guarantees within neural network policies. In *International Conference on Learning Representations* (2021).

[68] Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., and Koltun, V. Carla: An open urban driving simulator. In *Conference on robot learning* (2017), PMLR, pp. 1–16.

[69] Drews, P., Williams, G., Goldfain, B., Theodorou, E. A., and Rehg, J. M. Aggressive deep driving: Combining convolutional neural networks and model predictive control. In *Conference on Robot Learning* (2017), PMLR, pp. 133–142.

[70] Drgoňa, J., Arroyo, J., Figueroa, I. C., Blum, D., Arendt, K., Kim, D., Ollé, E. P., Oravec, J., Wetter, M., Vrabie, D. L., et al. All you need to know about model predictive control for buildings. *Annual Reviews in Control* (2020).

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

[72] ecobee. Donate your data. Accessed: 2020-06-30.

[73] ecobee. Getting started with the ecobee API. Accessed: 2020-06-30.

[74] eGauge. eGauge Core - Residential. Accessed: 2020-05-22.

[75] EIA, U. Peak-to-average electricity demand ratio rising in new england and many other us regions, 2014. Accessed: 2020-07-30.

[76] Evans, R., and Gao, J. Deepmind ai reduces google data centre cooling bill by 40%. https://deepmind.com/blog/deepmind-ai-reduces-google-data-centre-cooling-bill-40/, 2016. Accessed: 2019-06-19.

[77] Fanger, P. Thermal environment—human requirements. *Environmentalist 6*, 4 (1986), 275–278.

[78] Fernandez, N. E., Katipamula, S., Wang, W., Xie, Y., Zhao, M., and Corbin, C. D. Impacts of commercial building controls on energy savings and peak load reduction. Tech. rep., Pacific Northwest National Lab.(PNNL), Richland, WA (United States), 2017.

[79] Fioretto, F., Mak, T. W., and Van Hentenryck, P. Predicting ac optimal power flows: Combining deep learning and lagrangian dual methods. In *Proceedings of the AAAI Conference on Artificial Intelligence* (2020), vol. 34, pp. 630–637.

[80] Fisac, J. F., Akametalu, A. K., Zeilinger, M. N., Kaynama, S., Gillula, J., and Tomlin, C. J. A general safety framework for learning-based control in uncertain robotic systems. *IEEE Transactions on Automatic Control 64*, 7 (2018), 2737–2752.

[81] Fisac, J. F., Lugovoy, N. F., Rubies-Royo, V., Ghosh, S., and Tomlin, C. J. Bridging hamilton-jacobi safety analysis and reinforcement learning. In *2019 International Conference on Robotics and Automation (ICRA)* (2019), IEEE, pp. 8550–8556.

[82] Florian, F., Yunlong, S., Kaufmann, E., Scaramuzza, D., and Duerr, P. Super-human performance in gran turismo sport using deep reinforcement learning, 2020.

[83] Francis, J., Kitamura, N., Labelle, F., Lu, X., Navarro, I., and Oh, J. Core challenges in embodied vision-language planning. *arXiv preprint arXiv:2106.13948* (2021).

[84] Fulton, N., and Platzer, A. Verifiably safe off-model reinforcement learning. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems* (2019), Springer, pp. 413–430.

[85] Fux, S. F., Ashouri, A., Benz, M. J., and Guzzella, L. Ekf based self-adaptive thermal model for a passive house. *Energy and Buildings 68* (2014), 811–817.

[86] Gao, G., Li, J., and Wen, Y. Energy-efficient thermal comfort control in smart buildings via deep reinforcement learning. *arXiv preprint arXiv:1901.04693* (2019).

[87] Garcia, J., and Fernández, F. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research 16*, 1 (2015), 1437–1480.

[88] Gebbran, D., Hardjawana, W., Mhanna, S., Vucetic, B., Chapman, A. C., and Verbic, G. Practical considerations of der coordination with distributed optimal power flow. In *IEEE International Conference on Smart Grids and Energy Systems* (2020).

[89] Giovanis, G., Lu, M., and Chen, M. Optimizing dynamic programming-based algorithms. https://github.com/SFU-MARS/optimized_dp, 2021.

[90] Glavic, M. (deep) reinforcement learning for electric power system control and related problems: A short review and perspectives. *Annual Reviews in Control 48* (2019), 22–35.

[91] Goetzler, B., Guernsey, M., Kassuga, T., Young, J., Savidge, T., Bouza, A., Neukomm, M., and Sawyer, K. Grid-interactive efficient buildings technical report series: heating, ventilation, and air conditioning (hvac); water heating; appliances; and refrigeration. Tech. rep., National Renewable Energy Lab.(NREL), Golden, CO (United States), 2019.

[92] Goetzler, W., Shandross, R., Young, J., Petritchenko, O., Ringo, D., and McClive, S. Energy savings potential and rd&d opportunities for commercial building hvac systems. Tech. rep., Navigant Consulting, Burlington, MA (United States), 2017.

[93] Gorecki, T. T., Qureshi, F. A., and Jones, C. N. Openbuild: An integrated simulation environment for building control. In *2015 IEEE Conference on Control Applications (CCA)* (2015), IEEE, pp. 1522–1527.

[94] Gould, S., Hartley, R., and Campbell, D. Deep declarative networks: A new hope. *arXiv preprint arXiv:1909.04866* (2019).

[95] Greydanus, S., Dzamba, M., and Yosinski, J. Hamiltonian neural networks. In *Advances in Neural Information Processing Systems* (2019), pp. 15379–15389.

[96] Grigorescu, S., Trasnea, B., Cocias, T., and Macesanu, G. A survey of deep learning techniques for autonomous driving. *Journal of Field Robotics 37*, 3 (2020), 362–386.

[97] Gu, B., Ergan, S., and Akinci, B. Generating as-is building information models for facility management by leveraging heterogeneous existing information sources: A case study. In *Construction Research Congress 2014: Construction in a Global Network* (2014), pp. 1911–1920.

[98] Guerrero, J., Gebbran, D., Mhanna, S., Chapman, A. C., and Verbič, G. Towards a transactive energy system for integration of distributed energy

resources: Home energy management, distributed optimal power flow, and peer-to-peer energy trading. *Renewable and Sustainable Energy Reviews 132* (2020), 110000.

[99] Guglielmetti, R., Macumber, D., and Long, N. Openstudio: an open source integrated analysis platform. Tech. rep., National Renewable Energy Lab.(NREL), Golden, CO (United States), 2011.

[100] Gupta, S., Kekatos, V., and Jin, M. Deep learning for reactive power control of smart inverters under communication constraints. In *2020 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)* (2020), IEEE, pp. 1–6.

[101] Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning* (2018), PMLR, pp. 1861–1870.

[102] Han, M., Tian, Y., Zhang, L., Wang, J., and Pan, W. H_∞ Model-free Reinforcement Learning with Robust Stability Guarantee. *CoRR* (2019).

[103] Hao, H., Sanandaji, B. M., Poolla, K., and Vincent, T. L. Aggregate flexibility of thermostatically controlled loads. *IEEE Transactions on Power Systems 30*, 1 (2014), 189–198.

[104] Hasanbeig, M., Kroening, D., and Abate, A. Deep reinforcement learning with temporal logics. In *International Conference on Formal Modeling and Analysis of Timed Systems* (2020), Springer, pp. 1–22.

[105] Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., and Meger, D. Deep reinforcement learning that matters. In *Thirty-Second AAAI Conference on Artificial Intelligence* (2018).

[106] Herbert, S., Choi, J. J., Qazi, S., Gibson, M., Sreenath, K., and Tomlin, C. J. Scalable learning of safety guarantees for autonomous systems using hamilton-jacobi reachability. *arXiv preprint arXiv:2101.05916* (2021).

[107] Herman, J., Francis, J., Ganju, S., Chen, B., Koul, A., Gupta, A., Skabelkin, A., Zhukov, I., Kumskoy, M., and Nyberg, E. Learn-to-race: A multimodal control environment for autonomous racing. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2021), pp. 9793–9802.

[108] Herman, J., Francis, J., Ganju, S., Chen, B., Koul, A., Gupta, A., Skabelkin, A., Zhukov, I., Kumskoy, M., and Nyberg, E. Learn-to-race: A multimodal control environment for autonomous racing. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)* (October 2021), pp. 9793–9802.

[109] Hoke, A., Giraldez, J., Palmintier, B., Ifuku, E., Asano, M., Ueda, R., and Symko-Davies, M. Setting the smart solar standard: Collaborations between hawaiian electric and the national renewable energy laboratory. *IEEE Power and Energy Magazine 16*, 6 (2018), 18–29.

[110] Hsu, K.-C., Rubies-Royo, V., Tomlin, C. J., and Fisac, J. F. Safety and liveness guarantees through reach-avoid reinforcement learning. In *Robotics: Science and Systems* (2021).

[111] Huchuk, B., O'Brien, W., and Sanner, S. A longitudinal study of thermostat behaviors based on climate, seasonal, and energy price considerations using connected thermostat data. *Building and Environment 139* (2018), 199–210.

[112] Hunt, N., Fulton, N., Magliacane, S., Hoang, N., Das, S., and Solar-Lezama, A. Verifiably safe exploration for end-to-end reinforcement learning. *arXiv preprint arXiv:2007.01223* (2020).

[113] IEEE. 37 node distribution test feeder. https://ewh.ieee.org/soc/pes/dsacom/testfeeders/. Online.

[114] IEEE. Ieee standard conformance test procedures for equipment interconnecting distributed energy resources with electric power systems and associated interfaces. *IEEE Std 1547.1-2020* (2020), 1–282.

[115] ISO, C. Today's outlook. Accessed: 2020-06-30.

[116] Jalali, M., Kekatos, V., Gatsis, N., and Deka, D. Designing reactive power control rules for smart inverters using support vector machines. *IEEE Transactions on Smart Grid 11*, 2 (2019), 1759–1770.

[117] Jia, R., Jin, M., Sun, K., Hong, T., and Spanos, C. Advanced building control via deep reinforcement learning. *Energy Procedia 158* (2019), 6158–6163.

[118] Jones-Albertus, B. Confronting the duck curve: How to address over-generation of solar energy, 2017. Accessed: 2020-05-15.

[119] Kabzan, J., Hewing, L., Liniger, A., and Zeilinger, M. N. Learning-based model predictive control for autonomous racing. *IEEE Robotics and Automation Letters 4*, 4 (2019), 3363–3370.

[120] Kakade, S. M. A natural policy gradient. In *Advances in neural information processing systems* (2002), pp. 1531–1538.

[121] Kakade, S. M., et al. *On the sample complexity of reinforcement learning.* PhD thesis, 2003.

[122] Kamthe, S., and Deisenroth, M. P. Data-efficient reinforcement learning with probabilistic model predictive control. *arXiv preprint arXiv:1706.06491* (2017).

[123] Kara, E. C., Tabone, M. D., MacDonald, J. S., Callaway, D. S., and Kiliccote, S. Quantifying flexibility of residential thermostatically controlled loads for demand response: a data-driven approach. In *Proceedings of the 1st ACM conference on embedded systems for energy-efficient buildings* (2014), pp. 140–147.

[124] Karaguzel, O. T., and Lam, K. P. Development of whole-building energy performance models as benchmarks for retrofit projects. In *Proceedings of the 2011 Winter Simulation Conference (WSC)* (2011), IEEE, pp. 838–849.

[125] Kazmi, H., Mehmood, F., Lodeweyckx, S., and Driesen, J. Gigawatt-hour scale savings on a budget of zero: Deep reinforcement learning based optimal control of hot water systems. *Energy 144* (2018), 159–168.

[126] Killian, M., and Kozek, M. Ten questions concerning model predictive control for energy efficient buildings. *Building and Environment 105* (2016), 403–412.

[127] King, J. Energy impacts of smart home technologies. *Report A1801* (2018).

[128] Kingma, D. P., and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[129] Kingma, D. P., and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).

[130] Koch, S., Mathieu, J. L., and Callaway, D. S. Modeling and control of aggregated heterogeneous thermostatically controlled loads for ancillary services. In *Proc. PSCC* (2011), Citeseer, pp. 1–7.

[131] Kolter, Z., Duvenaud, D., and Johnson, M. Tutorial: Deep implicit layers - neural odes, deep equilibirum models, and beyond. http://implicit-layers-tutorial.org/, 12 2020.

[132] Kong, J., Pfeiffer, M., Schildbach, G., and Borrelli, F. Kinematic and dynamic vehicle models for autonomous driving control design. In *2015 IEEE Intelligent Vehicles Symposium (IV)* (2015), IEEE, pp. 1094–1099.

[133] Kousik, S., Vaskov, S., Bu, F., Johnson-Roberson, M., and Vasudevan, R. Bridging the gap between safety and real-time performance in receding-horizon trajectory design for mobile robots. *The International Journal of Robotics Research 39*, 12 (2020), 1419–1469.

[134] Krantz, S. G., and Parks, H. R. *The implicit function theorem: history, theory, and applications*. Springer Science & Business Media, 2012.

[135] Kroposki, B., Johnson, B., Zhang, Y., Gevorgian, V., Denholm, P., Hodge, B.-M., and Hannegan, B. Achieving a 100% renewable grid: Operating electric power systems with extremely high levels of variable renewable energy. *IEEE Power and Energy Magazine 15*, 2 (2017), 61–73.

[136] Kumar, A., Hong, J., Singh, A., and Levine, S. Should i run offline reinforcement learning or behavioral cloning? In *Deep RL Workshop NeurIPS 2021* (2021).

[137] Kumar, A., Zhou, A., Tucker, G., and Levine, S. Conservative q-learning for offline reinforcement learning. *arXiv preprint arXiv:2006.04779* (2020).

[138] Lange, H., Chen, B., Berges, M., and Kar, S. Learning to solve ac optimal power flow by differentiating through holomorphic embeddings. *arXiv preprint arXiv:2012.09622* (2020).

[139] Levine, S. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909* (2018).

[140] Levine, S., Kumar, A., Tucker, G., and Fu, J. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643* (2020).

[141] Li, X., and Wen, J. Review of building energy modeling for control and operation. *Renewable and Sustainable Energy Reviews 37* (2014), 517–537.

[142] Li, Y., Wen, Y., Guan, K., and Tao, D. Transforming cooling optimization for green data center via deep reinforcement learning. *arXiv preprint arXiv:1709.05077* (2017).

[143] Ling, C. K., Fang, F., and Kolter, J. Z. What game are we playing? end-to-end learning in normal and extensive form games. *arXiv preprint arXiv:1805.02777* (2018).

[144] Liniger, A., Domahidi, A., and Morari, M. Optimization-based autonomous racing of 1: 43 scale rc cars. *Optimal Control Applications and Methods 36*, 5 (2015), 628–647.

[145] Liu, S., and Henze, G. P. Experimental analysis of simulated reinforcement learning control for active and passive building thermal storage inventory: Part 2: Results and analysis. *Energy and buildings 38*, 2 (2006), 148–161.

[146] Ljung, L. System identification. *Wiley Encyclopedia of Electrical and Electronics Engineering* (1999), 1–19.

[147] Lü, X., Lu, T., Kibert, C. J., and Viljanen, M. Modeling and forecasting energy consumption for heterogeneous buildings using a physical–statistical approach. *Applied Energy 144* (2015), 261–275.

[148] Luo, B., Wu, H.-N., and Huang, T. Off-Policy Reinforcement Learning for H_∞ Control Design. *IEEE Transactions on Cybernetics 45*, 1 (2014), 65–76.

[149] Maasoumy, M., Razmara, M., Shahbakhti, M., and Vincentelli, A. S. Handling model uncertainty in model predictive control for energy efficient buildings. *Energy and Buildings 77* (2014), 377–392.

[150] Mahdavi, N., Braslavsky, J. H., Seron, M. M., and West, S. R. Model predictive control of distributed air-conditioning loads to compensate fluctuations in solar power. *IEEE Transactions on Smart Grid 8*, 6 (2017), 3055–3065.

[151] Mai, T., Hand, M. M., Baldwin, S. F., Wiser, R. H., Brinkman, G. L., Denholm, P., Arent, D. J., Porro, G., Sandor, D., Hostick, D. J., et al. Renewable electricity futures for the united states. *IEEE Transactions on Sustainable Energy 5*, 2 (2013), 372–378.

[152] Mather, B., and Yuan, G. Onward and upward: Distributed energy resource integration [guest editorial]. *IEEE Power and Energy Magazine 18*, 6 (2020), 16–19.

[153] Mathworks. Tune PID controllers - MATLAB. https://www.mathworks.com/help/control/ref/pidtuner-app.html, 2020. Accessed: 2020-04-19.

[154] Milano, F., Dörfler, F., Hug, G., Hill, D. J., and Verbič, G. Foundations and challenges of low-inertia systems. In *2018 Power Systems Computation Conference (PSCC)* (2018), IEEE, pp. 1–25.

[155] Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning* (2016), pp. 1928–1937.

[156] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602* (2013).

[157] Morimoto, J., and Doya, K. Robust Reinforcement Learning. *Neural Computation 17*, 2 (2005), 335–359.

[158] Moyer, C. How google's alphago beat a go world champion. *The Atlantic 28* (2016).

[159] Mozer, M. C. The neural network house: An environment hat adapts to its inhabitants. In *Proc. AAAI Spring Symp. Intelligent Environments* (1998), vol. 58.

[160] Mozer, M. C. An intelligent environment must be adaptive. *IEEE Intelligent Systems and their applications 14*, 2 (1999), 11–13.

[161] Mozer, M. C., Vidmar, L., and Dodier, R. H. The neurothermostat: Predictive optimal control of residential heating systems. In *Advances in Neural Information Processing Systems* (1997), pp. 953–959.

[162] Müller, F. L., and Jansen, B. Large-scale demonstration of precise demand response provided by residential heat pumps. *Applied Energy 239* (2019), 836–845.

[163] Munir, S., Arora, R. S., Hesling, C., Li, J., Francis, J., Shelton, C., Martin, C., Rowe, A., and Berges, M. Real-time fine grained occupancy estimation using depth sensors on arm embedded platforms. In *2017 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)* (2017), IEEE, pp. 295–306.

[164] Nagy, A., Kazmi, H., Cheaib, F., and Driesen, J. Deep reinforcement learning for optimal control of space heating. *arXiv preprint arXiv:1805.03777* (2018).

[165] Neukomm, M., Nubbe, V., and Fares, R. Grid-interactive efficient buildings. Tech. rep., US Department of Energy, Navigant, 2019.

[166] Nghiem, T. X., and Jones, C. N. Data-driven demand response modeling and control of buildings with gaussian processes. In *2017 American Control Conference (ACC)* (2017), IEEE, pp. 2919–2924.

[167] Ohn-Bar, E., Prakash, A., Behl, A., Chitta, K., and Geiger, A. Learning situational driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), pp. 11296–11305.

[168] Owusu, P. A., and Asumadu-Sarkodie, S. A review of renewable energy sources, sustainability issues and climate change mitigation. *Cogent Engineering 3*, 1 (2016), 1167990.

[169] Park, J. Y., Dougherty, T., Fritz, H., and Nagy, Z. Lightlearn: An adaptive and occupant centered controller for lighting based on reinforcement learning. *Building and Environment 147* (2019), 397–414.

[170] Park, S. Principles of sigma-delta modulation for analog-to-digital converters.

[171] Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in pytorch.

[172] Peng, K. S., and Morrison, C. T. Model predictive prior reinforcement learning for a heat pump thermostat. In *IEEE International Conference on Automatic Computing: Feedback Computing* (2016), vol. 16.

[173] Pham, T.-H., De Magistris, G., and Tachibana, R. Optlayer-practical constrained optimization for deep reinforcement learning in the real world. In *2018 IEEE International Conference on Robotics and Automation (ICRA)* (2018), IEEE, pp. 6236–6243.

[174] Pinto, L., Davidson, J., Sukthankar, R., and Gupta, A. Robust Adversarial Reinforcement Learning. In *Proceedings of the 34th International Conference on Machine Learning* (2017), JMLR. org, pp. 2817–2826.

[175] PJM. Data miner 2 seven-day load forecast. Accessed: 2020-07-09.

[176] Prakash, A., Chitta, K., and Geiger, A. Multi-modal fusion transformer for end-to-end autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2021), pp. 7077–7087.

[177] Privara, S., Cigler, J., Váňa, Z., Oldewurtel, F., Sagerschnig, C., and Žáčeková, E. Building modeling as a crucial part for building predictive control. *Energy and Buildings 56* (2013), 8–22.

[178] Prívara, S., Váňa, Z., Gyalistras, D., Cigler, J., Sagerschnig, C., Morari, M., and Ferkl, L. Modeling and identification of a large multi-zone office building. In *2011 IEEE International Conference on Control Applications (CCA)* (2011), IEEE, pp. 55–60.

[179] Quaschning, V. V. *Renewable energy and climate change*. John Wiley & Sons, 2019.

[180] R. D. Zimmerman, C. E. Murillo-Sanchez, and R. J. Thomas. MATPOWER: Steady-State Operations, Planning and Analysis Tools for Power Systems Research and Education. *IEEE Transactions on Power Systems 26*, 1 (2011), 12–19.

[181] Ray, A., Achiam, J., and Amodei, D. Benchmarking safe exploration in deep reinforcement learning. *arXiv preprint arXiv:1910.01708 7* (2019).

[182] Rockett, P., and Hathway, E. A. Model-predictive control for non-domestic buildings: a critical review and prospects. *Building Research & Information 45*, 5 (2017), 556–571.

[183] Rolnick, D., Donti, P. L., Kaack, L. H., Kochanski, K., Lacoste, A., Sankaran, K., Ross, A. S., Milojevic-Dupont, N., Jaques, N., Waldman-Brown, A., et al. Tackling climate change with machine learning. *arXiv preprint arXiv:1906.05433* (2019).

[184] Rosolia, U., Carvalho, A., and Borrelli, F. Autonomous racing using learning model predictive control. In *2017 American Control Conference (ACC)* (2017), IEEE, pp. 5115–5120.

[185] Ross, S. C., Vuylsteke, G., and Mathieu, J. L. Effects of load-based frequency regulation on distribution network operation. *IEEE Transactions on Power Systems 34*, 2 (2018), 1569–1578.

[186] Roth, A., and Reyna, J. Grid-interactive efficient buildings technical report series: whole-building controls, sensors, modeling, and analytics. Tech. rep., National Renewable Energy Lab.(NREL), Golden, CO (United States), 2019.

[187] Rubies-Royo, V., Fridovich-Keil, D., Herbert, S., and Tomlin, C. J. A classification-based approach for approximate reachability. In *2019 International Conference on Robotics and Automation (ICRA)* (2019), IEEE, pp. 7697–7704.

[188] Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. Trust region policy optimization. In *International Conference on Machine Learning* (2015), pp. 1889–1897.

[189] Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438* (2015).

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

[191] Shah, S., Arunesh, S., Pradeep, V., Andrew, P., and Milind, T. Solving online threat screening games using constrained action space reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence* (2020), vol. 34, pp. 2226–2235.

[192] Silver, D., Bagnell, J. A., and Stentz, A. Learning from demonstration for autonomous navigation in complex unstructured terrain. *The International Journal of Robotics Research 29*, 12 (2010), 1565–1592.

[193] Sky, D. Dark sky api. https://darksky.net/dev. Accessed: 2019-06-19.

[194] Srinivasan, K., Eysenbach, B., Ha, S., Tan, J., and Finn, C. Learning to be safe: Deep RL with a safety critic. *arXiv preprint arXiv:2010.14603* (2020).

[195] Star, E. Portfolio manager technical reference: Climate and weather. https://portfoliomanager.energystar.gov/pdf/reference/Climate%20and%20Weather.pdf, 2018. Accessed: 2019-06-19.

[196] Stringer, N., Bruce, A., MacGill, I., Haghdadi, N., Kilby, P., Mills, J., Veijalainen, T., Armitage, M., and Wilmot, N. Consumer-led transition: Australia's world-leading distributed energy resource integration efforts. *IEEE Power and Energy Magazine 18*, 6 (2020), 20–36.

[197] Strobel, K., Zhu, S., Chang, R., and Koppula, S. Accurate, low-latency visual perception for autonomous racing: Challenges, mechanisms, and practical solutions. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2020), IEEE, pp. 1969–1975.

[198] Sun, Y., Wachche, S. V., Mills, A., and Ma, O. 2018 renewable energy grid integration data book. Tech. rep., National Renewable Energy Lab.(NREL), Golden, CO (United States), 2020.

[199] Sutton, R. S., and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

[200] Taleghan, M. A., and Dietterich, T. G. Efficient Exploration for Constrained MDPs. In *2018 AAAI Spring Symposia* (2018).

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

[202] Tibshirani, R. Proximal gradient descent. http://www.stat.cmu.edu/ ryantibs/convexopt/lectures/prox-grad.pdf. Accessed: 2019-06-19.

[203] Tieleman, T., and Hinton, G. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning 4*, 2 (2012), 26–31.

[204] Tindemans, S. H., Trovato, V., and Strbac, G. Decentralized control of thermostatic loads for flexible demand response. *IEEE Transactions on Control Systems Technology 23*, 5 (2015), 1685–1700.

[205] Tschiatschek, S., Sahin, A., and Krause, A. Differentiable Submodular Maximization. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence* (2018), pp. 2731–2738.

[206] Turchetta, M., Berkenkamp, F., and Krause, A. Safe Exploration in Finite Markov Decision Processes with Gaussian Processes. In *Advances in Neural Information Processing Systems* (2016).

[207] Vinyals, O., Ewalds, T., Bartunov, S., Georgiev, P., Vezhnevets, A. S., Yeo, M., Makhzani, A., Küttler, H., Agapiou, J., Schrittwieser, J., et al. Starcraft ii: A new challenge for reinforcement learning. *arXiv preprint arXiv:1708.04782* (2017).

[208] Voloshin, C., Le, H. M., Jiang, N., and Yue, Y. Empirical study of off-policy policy evaluation for reinforcement learning. *arXiv preprint arXiv:1911.06854* (2019).

[209] Vrettos, E., Oldewurtel, F., and Andersson, G. Robust energy-constrained frequency reserves from aggregations of commercial buildings. *IEEE Transactions on Power Systems 31*, 6 (2016), 4272–4285.

[210] Wachi, A., Sui, Y., Yue, Y., and Ono, M. Safe Exploration and Optimization of Constrained MDPs Using Gaussian Processes. In *Proceedings of the AAAI Conference on Artificial Intelligence* (2018), vol. 32.

[211] Wang, P.-W., Donti, P., Wilder, B., and Kolter, Z. SATNet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver. In *Proceedings of the 36th International Conference on Machine Learning* (2019), pp. 6545–6554.

[212] Wang, Y., Velswamy, K., and Huang, B. A long-short term memory recurrent neural network based reinforcement learning controller for office heating ventilation and air conditioning systems. *Processes 5*, 3 (2017), 46.

[213] Wang, Z., Chen, B., Li, H., and Hong, T. Alphabuilding rescommunity: A multi-agent virtual testbed for community-level load coordination. *Advances in Applied Energy 4* (2021), 100061.

[214] Watter, M., Springenberg, J., Boedecker, J., and Riedmiller, M. Embed to control: A locally linear latent dynamics model for control from raw images. In *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds. Curran Associates, Inc., 2015, pp. 2746–2754.

[215] Wei, T., Wang, Y., and Zhu, Q. Deep reinforcement learning for building hvac control. In *Proceedings of the 54th Annual Design Automation Conference 2017* (2017), ACM, p. 22.

[216] Weiss, T., and Behl, M. Deepracing: a framework for autonomous racing. In *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)* (2020), IEEE, pp. 1163–1168.

[217] Wilcox, S., and Marion, W. *Users manual for TMY3 data sets.* National Renewable Energy Laboratory Golden, CO, 2008.

[218] Xu, Z., Diao, R., Lu, S., Lian, J., and Zhang, Y. Modeling of electric water heaters for demand response: A baseline pde model. *IEEE Transactions on Smart Grid 5*, 5 (2014), 2203–2210.

[219] Yang, L., Nagy, Z., Goffin, P., and Schlueter, A. Reinforcement learning for optimal control of low exergy buildings. *Applied Energy 156* (2015), 577–586.

[220] Yang, T.-Y., Rosca, J., Narasimhan, K., and Ramadge, P. J. Projection-based constrained policy optimization. *arXiv preprint arXiv:2010.03152* (2020).

[221] Yurtsever, E., Lambert, J., Carballo, A., and Takeda, K. A survey of autonomous driving: Common practices and emerging technologies. *IEEE access 8* (2020), 58443–58469.

[222] Zamzam, A. S., and Baker, K. Learning optimal solutions for extremely fast ac optimal power flow. In *2020 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)* (2020), IEEE, pp. 1–6.

[223] Zhang, J., and Ohn-Bar, E. Learning by watching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2021), pp. 12711–12721.

[224] Zhang, K., Hu, B., and Basar, T. Policy Optimization for $\mathcal{H}_2$ Linear Control with $\mathcal{H}_\infty$ Robustness Guarantee: Implicit Regularization and Global Convergence. In *Learning for Dynamics and Control* (2020), PMLR, pp. 179–190.

[225] Zhang, W., Lian, J., Chang, C.-Y., and Kalsi, K. Aggregated modeling and control of air conditioning loads for demand response. *IEEE transactions on power systems 28*, 4 (2013), 4655–4664.

[226] Zhang, Z., and Lam, K. P. Practical implementation and evaluation of deep reinforcement learning control for a radiant heating system. In *Proceedings of the 5th Conference on Systems for Built Environments* (New York, NY, USA, 2018), BuildSys '18, ACM, pp. 148–157.

[227] Zhang, Z., Liniger, A., Dai, D., Yu, F., and Van Gool, L. End-to-end urban driving by imitating a reinforcement learning coach. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2021), pp. 15222–15232.

[228] Zhang, Z., Zhang, D., and Qiu, R. C. Deep reinforcement learning for power system applications: An overview. *CSEE Journal of Power and Energy Systems 6*, 1 (2019), 213–225.

[229] Zhao, J., Lam, K. P., and Ydstie, B. E. Energyplus model-based predictive control (epmpc) by using matlab/simulink and mle+.

[230] Zhao, L., Zhang, W., Hao, H., and Kalsi, K. A geometric approach to aggregate flexibility modeling of thermostatically controlled loads. *IEEE Transactions on Power Systems 32*, 6 (2017), 4721–4731.

[231] Zhou, K., and Doyle, J. C. *Essentials of Robust Control*, vol. 104. Prentice hall Upper Saddle River, NJ, 1998.

A. SAGE - Supplementary Materials

A.1. Classical Control Benchmarks

The objective of this section is to compare the learning rule proposed by [81], i.e., $Q(x,u) = (1-\gamma)l(x) + \gamma \min\{l(x), \max_{u' \in \mathcal{U}} Q(x',u'\}$ with alternatives for learning safety value function. We evaluate it on two classical control benchmarks, *Double Integrator* and *Dubins' Car*, as described in Section A.1.1, where the analytical solution to safe states and optimal safe actions are known. Thus, we implement the learning rule here as Eqn. A.1. This is slightly different from the general case of Eqn. 6.6, where the optimal safety policy is unknown.

$$\begin{aligned} &Q(x,u) = (1-\gamma)l(x) + \gamma \min\{l(x), Q(x',u'\}, \\ &\text{where } u' = \pi^*_S. \end{aligned} \tag{A.1}$$

A.1.1. Model Dynamics

Double Integrator. The double integrator models a particle moving along the x-axis at velocity v. The control input is the acceleration a. The goal in this case is keep the particle within a fixed boundary, in this case $x \in [-1,1]$, subject to $a \in [-1,1]$.

$$\begin{cases} \dot{x} = v \\ \dot{v} = a \end{cases} \tag{A.2}$$

By solving the Hamiltonian, i.e., $\pi^*_S(x) = \arg\max_{u \in \mathcal{U}} \langle f(x,u), \nabla V_S(x) \rangle$, we can get the optimal safe control as:

$$a^* = \begin{cases} \overline{a} & \text{if } \partial V_S / \partial v \geq 0 \\ \underline{a} & \text{otherwise} \end{cases} \tag{A.3}$$

Dubins' Car. The Dubins' car models a vehicle moving at constant speed, in this case $v = 1$. Similar to the kinematic vehicle model, x, y, ϕ describes the position and heading of the vehicle, and control input is the turning rate $u \in [-1,1]$. The goal is to reach a unit circle centred at the origin.

$$\begin{cases} \dot{x} = v\cos(\phi) \\ \dot{y} = v\sin(\phi) \\ \dot{\phi} = u \end{cases} \tag{A.4}$$

Note that Dubins' Car is a reach task, i.e. reaching a specified goal, instead of an avoid task, i.e. avoiding specified obstacles. The reach task can be simply implemented by setting $\pi^*_S(x) = \arg\min_{u \in \mathcal{U}} \langle f(x,u), \nabla V_S(x) \rangle$ bansal2017hamilton. In other words, the optimal safe action for a given state is the one that minimises the distance to the goal.

The corresponding optimal safe control is

$$u^* = \begin{cases} \underline{u} & \text{if } \partial V_S / \partial \theta \geq 0 \\ \overline{u} & \text{otherwise} \end{cases} \tag{A.5}$$

The ground truth safety value function for these two benchmarks are shown in Figure 6.2a and 6.2b.

Implementation & Evaluation. We use a neural network with hidden layers of size [16, 16] for the double integrator and [64, 64, 32] for Dubins' car. We use ADAM [128] as the optimiser with a learning rate of 0.001, batch size of 64. We update the safety value function over 25K steps for Double Integrator and 50K steps for Dubin's Car, and report classification accuracy every 1000 steps averaged over 5 random seeds. While the safety value is defined over continuous state space, we evaluate the performance over a discrete mesh on the state space. By definition, the safety value at a given state x is $Q(x, u^*)$, where $u^* = \pi^*_S(x)$.

Qualitative Results. Qualitative comparison between the ground truth value and that learned via HJ Bellman update is shown in Figure A.1 and A.2. As we can see, the neural approximation largely recovers the ground truth value, except for minute difference.

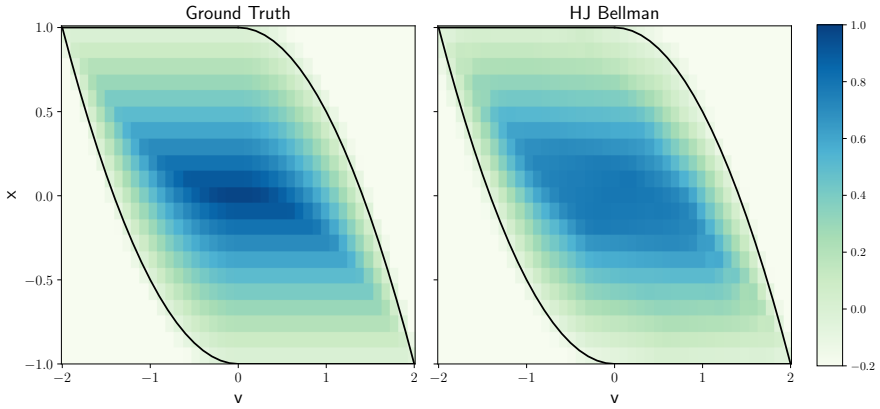
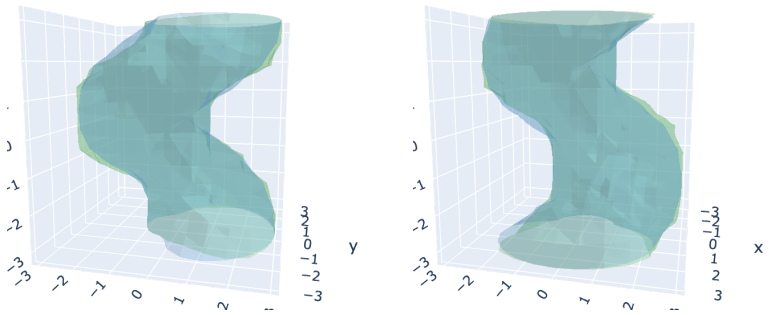


Figure A.1: A comparison between the ground truth safety value and that learned via HJ Bellman update for double integrator; The black line delineates $V_S(x) = 0$.

While we do not need to learn the safety actor in this case, we further demonstrate that $\nabla_u Q_S(x, u)$ can indeed by used to update the safety actor. In Figure A.3, we compare the ground truth $\partial V / \partial v$, with which one can determine the optimal safe action with Eqn. A.3, and the gradient through the safety critic, i.e., $\nabla_u Q_S(x, u)$. We can see that $\nabla_u Q_S(x, u)$, consistently point towards the correct optimal safe action within the safe set, i.e., the area delineated by the black line. The safety value outside the safe set is not learned, as the region is outside the support of data when episodes terminate upon failures.



(a) (b)

Figure A.2: A comparison between isosurface of the ground truth safety value (blue) and that learned via HJ Bellman update (green) for Dubins' car

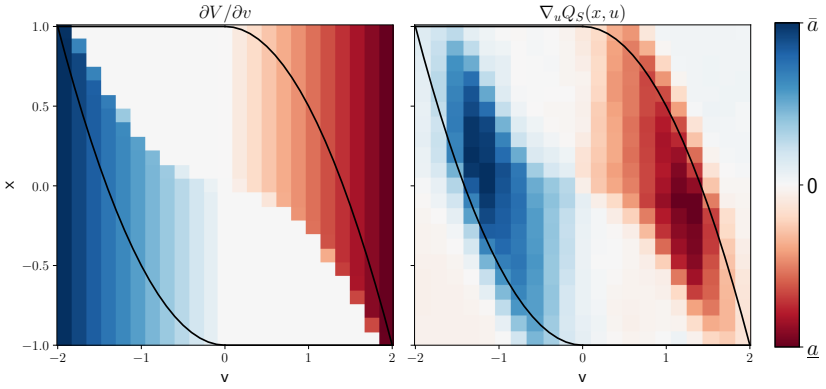


Figure A.3: The gradients through the safety critic, i.e., $\nabla_u Q_S(x,u)$, consistently point towards the correct optimal safe action, as indicated by $\partial V/\partial v$ (Eqn. A.3), within the safe set (the area delineated by the black line) for double integrator.

A.1.2. Learning Rule Comparison for Safety Critic

Firstly, we describes the approaches pertaining to learning the safety critic in [194, 26]. In both Safety Q-functions for RL (SQRL) srinivasan2020learning and Conservative Safety Critic (CSC) bharadhwaj2020conservative, the safety critic is defined as the expected cumulative cost, i.e. $Q_C^\pi(x,u) \coloneqq \mathbb{E}_{x_k,u_k\sim\pi}\left[\sum_{k=0}^{\infty}\gamma^k C(x_k)|x_0=x,u_0=u\right]$, where $C(x_k)=1$ if a failure occurs at x_k and 0 otherwise. Both papers endowed the safety critic, $Q_C^\pi(x,u)$ with a probabilistic interpretation, i.e. the expected probability of failure.

SQRL. The safety critic is trained by propagating the failure signal using the standard Bellman backup, as in Eqn. A.6, where $\mathcal{D}$ denotes the replay memory, γ_S is a time-

discount parameter, and $\bar{Q}_C^\pi$ is the delayed target network. This approach for learning the safety critic is also adopted in [201].

$$Q_C^\pi \leftarrow E_{x,u,x' \sim \mathcal{D}} \left[C(x) + \gamma_S (1 - C(x)) \mathbb{E}_{u' \sim \pi} \bar{Q}_C^\pi (x', u') \right] \tag{A.6}$$

CSC. On top of using Bellman backup to propagate the failure signals, CSC uses conservative Q-learning (CQL) kumar2020conservative to correct for the distribution mismatch between the behaviour policy and the evaluation policy, and overestimate Q_C to err on the side of caution. The resulting objective is given in Eqn. A.7, where $\mathcal{B}^\pi Q(x,u) = C(x) + \gamma \mathbb{E}_{x' \sim P(x'|x,u), u' \sim \pi(x')} \bar{Q}(x', u')$ is the Bellman operator and α is a hyperparamter that controls the extent of conservativeness. If $\alpha = 0$, the objective is the same as that of SQRL.

$$\begin{aligned} \mathcal{L} = & \frac{1}{2} \mathbb{E}_{s,a \sim \mathcal{D}} \left[Q_C(x,u) - \mathcal{B}^\pi \bar{Q}_C(x,u)) \right] \\ & - \alpha \left[\mathbb{E}_{x \sim \mathcal{D}, u \sim \pi(x)} Q_C(x,u) - \mathbb{E}_{x,u \sim \mathcal{D}} Q_C(x,u) \right] \end{aligned} \tag{A.7}$$

Note that CSC reversed the sign in front of α compared to the original implementation in CQL so as to over-estimate Q_C. This learning objective does not guarantee point-wise conservativeness, but conservativeness in expectation, i.e. $\mathbb{E}_\pi \hat{Q}_c^\pi(x,u) \geq \mathbb{E}_\pi Q_c^\pi(x,u)$. τ is the rate for polyak averaging of the target network, i.e. $Q' \leftarrow \tau Q + (1-\tau) Q'$.

Implementation Details. In [194], the authors used a learning rate of 3×10^{-4}, $\gamma_S = 0.7$. Using the same learning rate, we did grid search over $\gamma_S = [0.7, 0.9]$ and $\tau = [0.1, 0.01]$. We observed that $\gamma_S = 0.9$ had better performance, and thus selected $\gamma_S = 0.9$ and $\tau = 0.1$.

In [26], the authors used a learning rate of 2×10^{-4}, $\gamma_S = 0.99$, and selected $\alpha = 0.5$ from from 0.05, 0.5, and 5. Using the same learning rate and γ_S, we did grid search over $\alpha = [0.01, 0.05, 0.5, 5]$ and $\tau = [0.1, 0.01]$. We selected $\alpha = 0.01$ and $\tau = 0.1$.

Results. The main results are summarized in Figure 6.2c. In Figure A.4, we show a qualitative comparison between the ground truth safety value and that learned via different learning rules. In interpreting the results, note that both the environment and optimal safety policy are deterministic. Thus, $Q_C^{\pi_S^*}(x, \pi_S^*(x))$ should be 0 if x is a safe state, following the definition. Due to the difference in definition, i.e., $Q_S(x) \geq 0$ is safe for HJ safety value and $Q_C(x) \leq 0$ is safe in SQRL and CQL, we plot $1 - Q_C$ such that in Figure A.4 the larger value consistently indicates safety and the cut-off for safe vs. unsafe is 0.

SQRL largely captures the correct safe states, though the classification performance is highly dependent on picking an appropriate threshold. CQL does underestimate the level of safety (and overestimate Q_C) as intended, and the pattern of underestimation appear to corresponds to $\nabla_x Q_S$ (refer to Figure A.3).

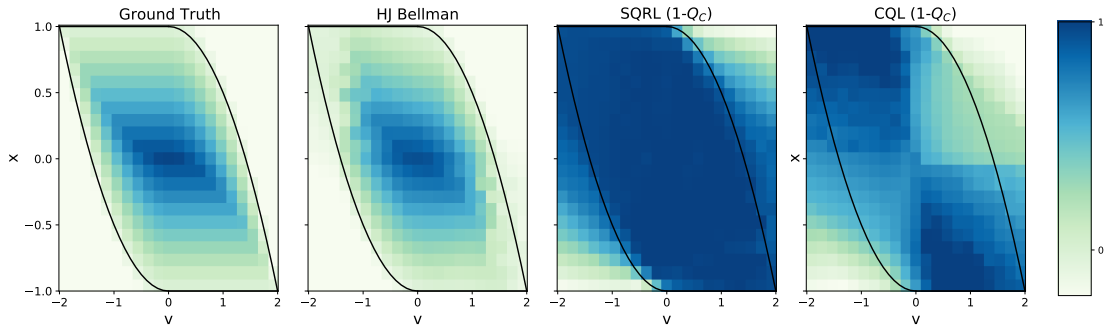


Figure A.4: Comparison between the group truth safety value and the safety critics from different learning rules for double integrator

A.2. Algorithm

relies on a dual policy structure, the rationale of which is explained in Section 6.4. This pairing of a safety policy and a performance policy is important, as we are able to decompose the problem of learning under safety constraints into optimizing for performance and updating the safety value function, separately.

We optimize the performance policy using SAC, but it may be switched for any other comparable RL algorithms. The safety policy is used least-restrictively, that is only intervene when the RL agent is about to enter into an unsafe state and thus allowing the performance policy maximum freedom in exploring safely. Instead of using the optimal safe policy from solving Hamiltonian, the safe policy is updated via gradients through the safety critic, same as other actor-critic algorithms.

A.3. Implementation Details: Safety Gym Experiment

Following the default CarGoal1-v0 and PointGoal1-v0 benchmarks in Safety Gym, all agents were given LiDARs observation with respect to hazard, goal, and vase, with avoiding hazards as the safety constraints. Both environments were initialised with a total of 8 hazards and 1 vase. Agent's are endowed with accelerometer, velocimeter, gyro, and magnetometer sensors; their LiDAR configurations included 16 bins, with max distance of 3.

The baselines we considered, i.e., CPO, PPO and PPO-Lagrangian follows the default implementation that comes with Safety Gym. PPO- wraps the proposed safety actor critic around the PPO base agent. Despite PPO being an on-policy algorithm, the safety critic was implemented with off-policy updates, using prioritised memory replay based on the TD-error of predicting safety value. Since $l(x)$ is small in this environment, we scaled cost by a factor of 100. For the safety actor-critic, We used γ_S annealing from 0.85 to 1 following [81], $\tau = 0.005$, critic learning rate of 0.001, actor learning rate of 0.0003, and $\alpha = 0.2$ (regularisation on policy entropy). We used a safety margin $\epsilon = 0.25$, mainly to account for the dimension of the hazards (radius = 0.2).

For each model, on each Safety Gym benchmark, results were reported as the average

Algorithm 5: : Safe Autonomous Racing via Approximate Reachability on Ego-vision

Initialize: performance critic Q_ϕ and actor π_θ;
Initialize: safety critic Q_{ϕ_S}, and actor π_{θ_S}; target networks $\phi'_S \leftarrow \phi_S$;
Initialize: replay buffer $\mathcal{D}$;
for *i = 0, . . . , # Episodes* **do**
  x = env.reset()
  while *not terminal* **do**
    u $\sim \pi_\theta(x)$;
    // The safe actor intervenes when the current state-action is deemed unsafe by the safety critic.
    if $Q_{\phi_S}(x, u) < \epsilon$ **then**
      $u \sim \pi_{\theta_S}(x)$
    end
    x', r = env.step(u)
    $\mathcal{D}$.store(x, a, x', r)
    $x = x'$
    Update performance critic Q_ϕ and actor π_θ with preferred RL algorithm;
    // Update the safety critic:
    Sample N transitions (x, u, x') from $\mathcal{D}$;
    // Calculate the target value with the discounted Bellman safety update

$$y = (1-\gamma)l(x) + \gamma \min\{l(x), Q_{\phi'_S}(x', u')\},$$

$$\text{where } u' \sim \pi_{\theta_S}$$

$$\mathcal{L}_{\phi_S} = N^{-1} \sum (Q_{\phi_S}(x, u) - y)^2$$

$$\phi_S \leftarrow \phi_S - \alpha \nabla_{\phi_S} \mathcal{L}_{\phi_S}$$

    // Update the safety actor with deterministic policy gradient:

$$\theta_S \leftarrow \theta_S + \alpha N^{-1} \sum \nabla_u Q(x, u) \nabla_{\theta_S} \pi_{\theta_S}(x)$$

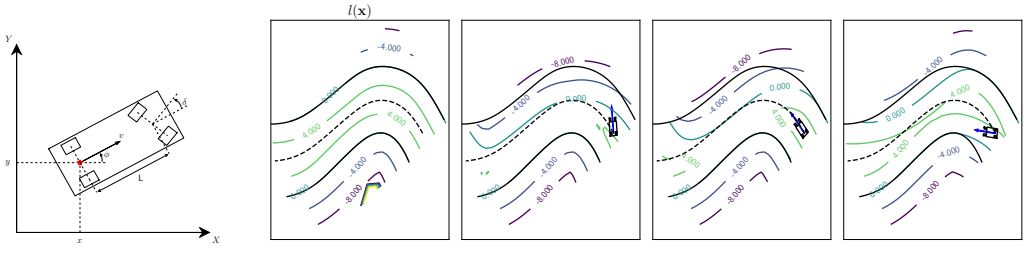
    // Update the target networks:

$$\phi'_S \leftarrow \tau \phi_S + (1-\tau)\phi'_S$$

  end
end

across 5 instances. All experiments in Safety Gym were run on an Intel(R) Core(TM) i9-9920X CPU @ 3.50GHz – with 1 CPU, 12 physical cores per CPU, and a total of 24 logical CPU units.

A.4. Static Safety Actor-Critic Derived from Kinematic Bike Model



(a) nominal model (b) $V_s(x, y, v, \phi)$ computed via the nominal model, where v=12m/s

Figure A.5: (a) We compute the safety value function, via a kinematic vehicle model. (b) We illustrate different views of the 4D state space, given fixed velocity and three different yaw angles, indicated by the blue arrows.

To demonstrate the benefit of utilizing domain knowledge in the form of a nominal model and to compare with the learnable safety actor-critic in , we use the kinematic vehicle model [132] (see Figure A.5a), which is a significant simplification of a realistic race car model [119], to compute the safety value and corresponding 'optimal'[26] safety controller. The dynamics and 'optimal' safety control is given in Eqn. A.8 and A.9, where the state is $\mathbf{x} = [x, y, v, \phi]$, and the action is $\mathbf{u} = [a, \delta]$. x, y, v, ϕ are the vehicle's location, speed, and yaw angle. a is the acceleration, and δ is the steering angle. The actions are bounded, i.e., $a \in [\underline{a}, \overline{a}]$ and $\delta \in [\underline{\delta}, \overline{\delta}]$. $L = 3$m is the car length.

$$f(\mathbf{x}, \mathbf{u}) = \begin{cases} \dot{x} = v\cos(\phi) \\ \dot{y} = v\sin(\phi) \\ \dot{v} = a \\ \dot{\phi} = v\tan\delta/L \end{cases} \tag{A.8}$$

$$\begin{aligned} a^* &= \begin{cases} \underline{a} & \text{if } \partial V_S/\partial v \leq 0 \\ \overline{a} & \text{else} \end{cases}, \\ \delta^* &= \begin{cases} \overline{\delta} & \text{if } \partial V_S/\partial \phi \geq 0 \\ \underline{\delta} & \text{else} \end{cases} \end{aligned} \tag{A.9}$$

Intuitively, the 'optimal' safety policy brakes and steers towards the center of the track as much as possible. We also derive the 'optimal' safety policy here. The optimal safety control is derived by solving the Hamiltonian as given in Eqn. A.10a. By definition,

[26] only with respect to the nominal model

$\nabla V_S(\mathbf{x}) = [\partial V_S/\partial x, \partial V_S/\partial y, \partial V_S/\partial v, \partial V_S/\partial \phi]$.

$$\pi^*_S(\mathbf{x}) = \arg\max_{\mathbf{u}\in\mathcal{U}} \langle f(\mathbf{x},\mathbf{u}), \nabla V_S(\mathbf{x})\rangle \tag{A.10a}$$

$$= \arg\max_{[a,\delta]\in\mathcal{U}} [v\cos(\phi)\frac{\partial V_S}{\partial x} + v\sin(\phi)\frac{\partial V_S}{\partial y} + a\frac{\partial V_S}{\partial v} + v\tan\delta/L\frac{\partial V_S}{\partial \phi}] \tag{A.10b}$$

$$= \arg\max_{[a,\delta]\in\mathcal{U}} [a\frac{\partial V_S}{\partial v} + v\tan\delta/L\frac{\partial V_S}{\partial \phi}] \tag{A.10c}$$

From Eqn A.10c, it is clear that the actions given by Eqn. A.9 maximize the Hamiltonian.

We parametrized the racetrack as a cubic spline and computed $l(\mathbf{x})$ by projection onto the spline. Setting $V_S(\mathbf{x}, 0) = l(\mathbf{x})$, we calculated the backward reachable tube using the code from [89]. For efficient computation, we divided the racetrack into overlapping segments and computed the safety value segment-wise. Fig. A.5b illustrates resulting safety value function at slices of state space, as the agent enters into a sharp turn. It is clear that the safety value at each location can be quite different from the initialization, $l(\mathbf{x})$.

A.5. Implementation Details: Learn-to-Race Experiment

Vision Encoder. We condition the optimization of the performance policy as well as the safety value updates on pretrained embedding of vehicle's visual scene context. The perception module maps ego-images from the on-board RGB camera to feature embedding of reduced dimension. To learn this mapping, we use a standard variational autoencoding (VAE) [129] paradigm, with a convolutional encoder.

We use an image reconstruction objective with binary cross-entropy loss, Adam optimizer [128], and a latent vector dimension of 32. We train the VAE encoder to reconstruct ego-images, sampled from the vehicle's front camera during random agent execution; examples are provided in Figure A.6. We further refine the encoder by training the VAE module to reconstruct projected road boundaries, illustrated in Figure A.7, with inputs in the left column and the reconstructed outputs in the right column.

Neural Architecture. As illustrated in Figure A.8, the vision encoder takes an image as input to produce a latent vector, which is concatenated with speed and action embedding and passed to the performance and safety actor-critics. The specific implementation of layers are summarized in Table A.1.

Specifically, we use a squashed Gaussian policy (Eqn. A.11) for both performance and safety actors, following [101].

$$u = \tanh(\mu(x) + \sigma(x)\odot\xi), \quad \xi \sim \mathcal{N}(0, \mathbf{I}) \tag{A.11}$$

Agent training details. During training, the agent is spawned at random locations along the race track and uses a stochastic policy. During evaluation, the agent is spawned at a fixed location and uses a deterministic policy. The episode terminates when the

Figure A.6: VAE image reconstruction, with real images in the left column and reconstructed images in the right column.

agent successfully finishes a lap, leaves the drivable area, collides with obstacles, or does not progress for a number of steps. For each agent, we report averaged results across 5 random seeds, evaluated every 5000 steps over an episode (one lap). In total, we train each agent over 250,000 steps, and evaluate it over 50 episodes.

During its interaction with the environment, the agent receives a 192×144 ego-camera view and its speed at each time-step. The agent encodes the RGB image frame and its speed to a 40-dimensional feature representation, subsequently used as input to both actor-critic networks. We initialise the replay buffer with 2000 random transitions, following [2]. After 2000 steps, we perform a policy update at each time step. For the SafeSAC agent, we only save state-action transitions from the performance actor to the replay buffer. For the agent, we save all state-action transitions.

Implementation Details. For all experiments, we implemented the models using the `PyTorch` 1.8.0. We optimised both the performance and safety actor-critic with Adam [128], with a learning rate of 0.003. We used $\gamma = 0.99$ for the performance critic, and annealed γ_S from 0.85 to 1 for the safety critic following [81]. We used $\tau = 0.005$ for the performance critic, and $\tau = 0.05$ for the safety critic. For both the performance and safety actor, we include the policy entropy term with $\alpha = 0.2$. We used a batch size of 256, and a replay buffer size of 250,000.

Computing hardware. For rendering the simulator and performing local agent verification and analysis, we used a single GPU machine, with the following CPU specifications: Intel(R) Core(TM) i5-4690K CPU @ 3.50GHz; 1 CPU, 4 physical cores per CPU, total of 4 logical CPU units. The machine includes a single GeoForce GTX TITAN X GPU, with 12.2GB GPU memory. For generating multi-instance experimental results, we used a cluster of three multi-GPU machines with the following CPU specifications:

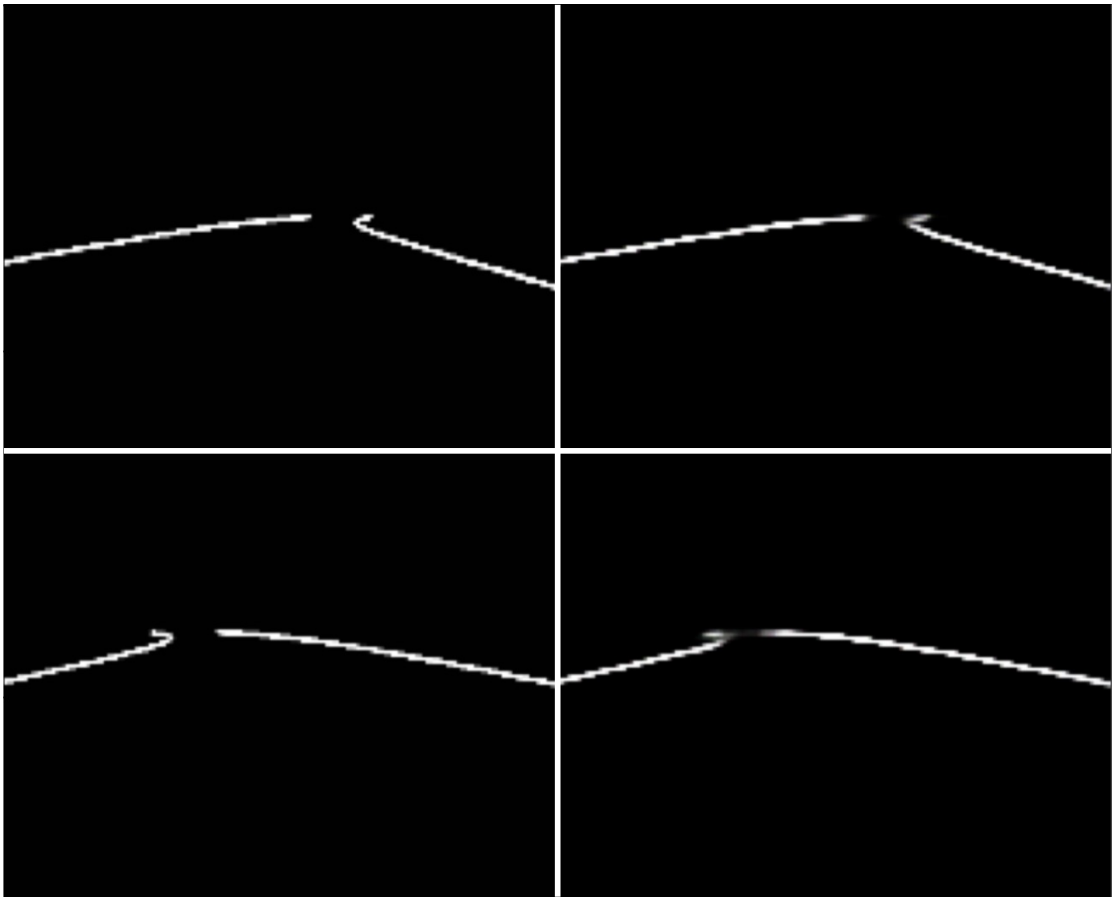


Figure A.7: VAE reconstruction of projected road boundary images, with real images in the left column and reconstructed images in the right column.

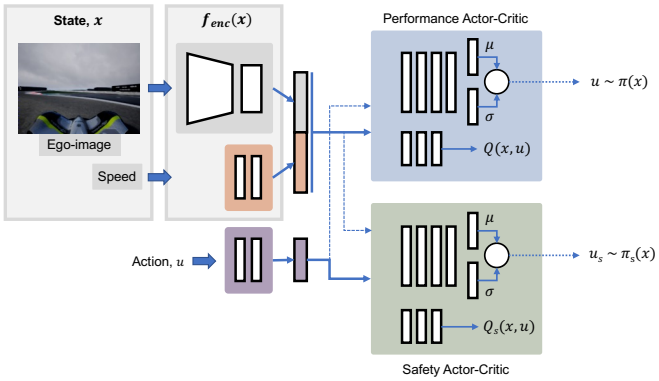


Figure A.8: SAGE neural architecture overview.

2x Intel(R) Xeon(R) Gold 5218R CPU @ 2.10GHz; 80 total CPU cores using a Cascade

Table A.1: Network Architecture

Operation	Input (dim.)	Output (dim.)	Parameters
	Visual Encoder		
Conv2d	(N, chan, 42, 144), chan : 3→32	`conv1`	k:=(4,4), s:=2, p:=1, activation:=ReLU
Conv2d	`conv1`, chan : 32→64	`conv2`	k:=(4,4), s:=2, p:=1, activation:=ReLU
Conv2d	`conv2`, chan : 64→128	`conv3`	k:=(4,4), s:=2, p:=1, activation:=ReLU
Conv2d	`conv3`, chan : 128→256	`conv4`	k:=(4,4), s:=2, p:=1, activation:=ReLU
Flatten	—	—	—
	Visual Encoder Bottleneck Representation		
Linear (mu)	$N \times$ h_dim	$N \times 32$	—
Linear (sigma)	$N \times$ h_dim	$N \times 32$	—
	Visual Decoder (only for pre-training Visual Encoder)		
Unflatten	—	—	—
ConvTranspose2d	`encoder.conv4`: encoder.conv4.chan: 256 →128	`convtranspose1`	k:=(4,4), s:=2, p:=1, activation:=ReLU
ConvTranspose2d	`convtranspose1`, chan : 128 →64	`convtranspose2`	k:=(4,4), s:=2, p:=1, activation:=ReLU
ConvTranspose2d	`convtranspose2`, chan : 64 →32	`convtranspose3`	k:=(4,4), s:=2, p:=1, activation:=ReLU
ConvTranspose2d	`convtranspose3`, chan : 32 →3	`convtranspose4`	k:=(4,4), s:=2, p:=1, activation:=Sigmoid
	Safety Actor-Critic		
`actor_network`	—	—	—
`q_function1`	—	—	—
`q_function2`	—	—	—
	Performance Actor-Critic		
`actor_network`	—	—	—
`q_function1`	—	—	—
`q_function2`	—	—	—
	Actor Network (Policy): SquashedGaussianMLPActor		
Linear	$N \times 32$	$N \times 64$	activation:=ReLU
Linear	$N \times 64$	$N \times 64$	activation:=ReLU
Linear	$N \times 64$	$N \times 32$	activation:=ReLU
Linear (projection: mu_layer)	$N \times 32$	$N \times 3$	—
Linear (projection: log_std_layer)	$N \times 32$	$N \times 3$	—
	Q function		
`speed_encoder`	—	—	—
`regressor`	—	—	—
	Speed Encoder		
Linear	$N \times 1$	$N \times 8$	activation:=ReLU
Linear	$N \times 8$	$N \times 8$	activation:=Identity
	Regressor		
Linear	$N \times 42$	$N \times 32$	activation:=ReLU
Linear	$N \times 32$	$N \times 64$	activation:=ReLU
Linear	$N \times 64$	$N \times 64$	activation:=ReLU
Linear	$N \times 64$	$N \times 32$	activation:=ReLU
Linear	$N \times 32$	$N \times 32$	activation:=ReLU
Linear	$N \times 32$	$N \times 1$	activation:=Identity

Table A.2: `Learn-to-Race` task [107] results on `Track01` (Thruxton Circuit), for learning-*free* agents, with respect to the task metrics: Episode Completion Percentage (**ECP**), Episode Duration (**ED**), Average Adjusted Track Speed (**AATS**), Average Displacement Error (**ADE**), Trajectory Admissibility (**TrA**), Trajectory Efficiency (**TrE**), and Movement Smoothness (**MS**). Arrows (↑↓) indicate directions of better performance, across agents. **Bold** results in tables A.2 and A.3 are generally best, however, asterisks (*) indicate metrics which may be misleading, for incomplete racing episodes.

Agent	**ECP** (↑)	**ED*** (↓)	**AATS** (↑)	**ADE** (↓)	**TrA** (↑)	**TrE** (↑)	**MS** (↑)
`HUMAN`	**100.0** $\pm$ 0.0	78.6 $\pm$ 5.2	**79.29** $\pm$ 4.7	2.4 $\pm$ 0.1	0.93 $\pm$ 0.01	**1.00** $\pm$ 0.02	**11.7** $\pm$ 0.1
`Random`	0.50 $\pm$ 0.30	**4.67** $\pm$ 3.2	11.90 $\pm$ 3.80	1.5 $\pm$ 0.60	0.81 $\pm$ 0.04	0.33 $\pm$ 0.38*	6.7 $\pm$ 1.1
`MPC`	100.0 $\pm$ 0.0	301.40 $\pm$ 10.10	45.10 $\pm$ 0.0	**0.90** $\pm$ 0.10	**0.98** $\pm$ 0.01	0.85 $\pm$ 0.03	10.4 $\pm$ 0.60

Lake architecture; memory of 512 GiB DDR4 3200 MHz, 16x32 GiB DIMMs. Each

Table A.3: `Learn-to-Race` task [107] results on `Track01` (Thruxton Circuit), for learning-*based* agents.

Agent	ECP (↑)	ED* (↓)	AATS (↑)	ADE (↓)	TrA (↑)	TrE (↑)	MS (↑)
`SAC`	61.61 ± 38.57	272.75 ± 256.51	47.99 ± 30.9	1.54 ± 1.07	$\mathbf{0.94} \pm 0.02$	$\mathbf{0.28} \pm 0.12$	11.84 ± 2.12
`SafeRandom` (ours), $\delta = 3.0$	36.46 ± 23.71	654.37 ± 447.05	8.44 ± 1.37	3.93 ± 0.21	0.81 ± 0.10	0.00 ± 0.00	13.21 ± 1.88
`SafeRandom` (ours), $\delta = 4.2$	63.63 ± 39.46	761.80 ± 494.65	11.68 ± 1.07	2.74 ± 0.16	0.90 ± 0.07	0.02 ± 0.01	$\mathbf{13.63} \pm 2.01$
`SafeSAC` (ours), $\delta = 3.0$	25.70 ± 11.31	66.90 ± 23.22	49.67 ± 3.34	1.35 ± 0.05	0.86 ± 0.06	0.14 ± 0.05	8.46 ± 2.35
`SafeSAC` (ours), $\delta = 4.2$	49.05 ± 41.66	617.52 ± 842.49	33.83 ± 26.21	1.80 ± 0.63	0.91 ± 0.12	0.07 ± 0.11	10.03 ± 2.75
(ours)	$\mathbf{79.94} \pm 23.20$	$\mathbf{59.19} \pm 29.99$	$\mathbf{53.28} \pm 3.76$	$\mathbf{0.99} \pm 0.17$	0.91 ± 0.03	0.22 ± 0.03	9.27 ± 1.68

machine includes 8x NVIDIA GeForce RTX 2080 Ti GPUs, each with 11GB GDDR6 of GPU memory. Experiments were orchestrated on the these machines using Kubernetes, an open-source container deployment and management system.

All experiments were conducted using version 0.7.0.182276 of the Arrival Racing Simulator. The simulator and `Learn-to-Race` framework [107] are available for academic-use, here: https://learn-to-race.org.

A.6. Additional Results

Performance of the SafeRandom agent. Recall that the `SafeRandom` agent takes random actions and uses the safety value function precomputed from the nominal model. The optimal safety controller intervene whenever the safety value of the current state falls belong the safety margin. The safety margin is necessary because 1) the nominal model is a significant over-simplification of vehicle dynamics, and 2) the HJ Reachability computation does not take into consideration of the physical dimension of the vehicle.

The performance of the `SafeRandom` agent at different safety margin is summarised in Figure A.9. For safety margin $\epsilon \geq 4.2$, the `SafeRandom` agent can finish 80+% of the lap, and thus we use $\epsilon = 4.2$ as the safety margin for the `SafeSAC` agent. On the other hand, the performance decrease drastically when the safety margin is reduced to 3.

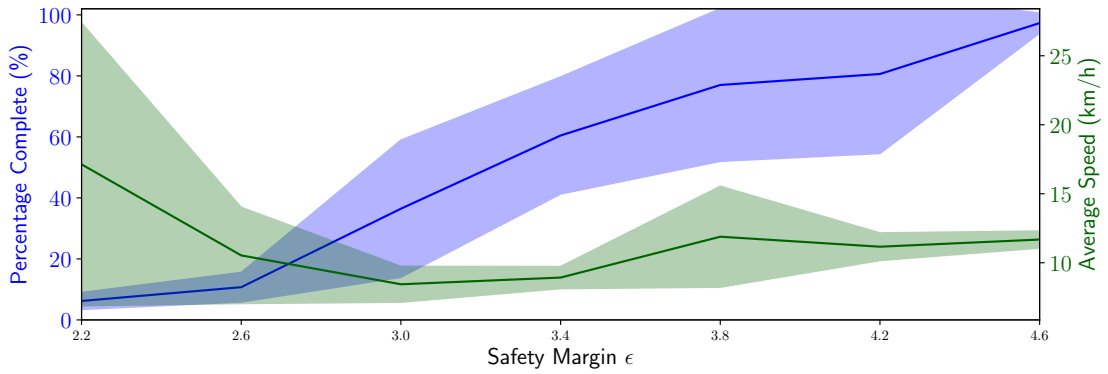


Figure A.9: Performance of the `SafeRandom` agent at different safety margin (averaged over 10 random seeds)

SafeSAC & performance with same safe margin. While we choose the safety

margin ϵ based on performance of the `SafeRandom` agent over a range of margins and our best engineering judgement, some may wonder if the superior performance of over SafeSAC may be attributed to the use of different safety margins. Thus, we also show here the performance of a SafeSAC agent with the same safety margin as in Figure A.10. Given the smaller safety margin, the ECP is low initially, which is inline with the observation from `SafeRandom`. Furthermore, the ECP barely improves over time. As the performance agent learns to drive faster, it is increasingly difficulty for the static actor-critic to catch the vehicle in marginally safe states.

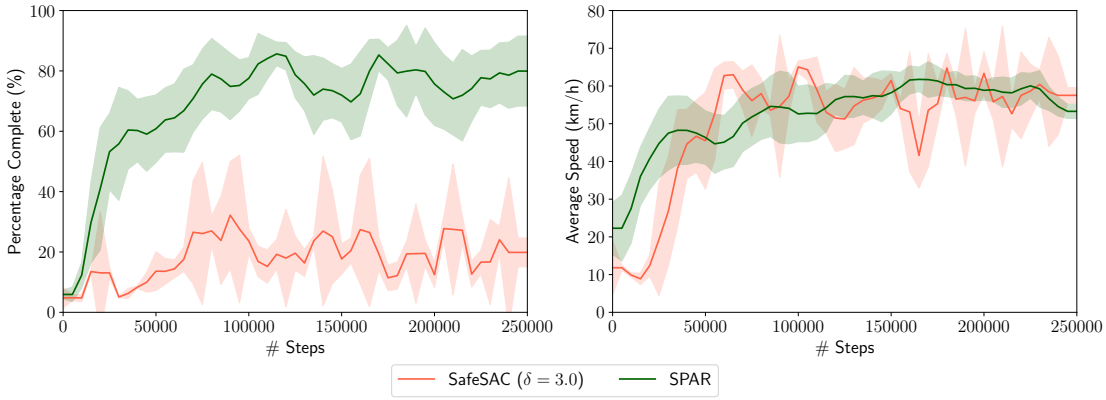


Figure A.10: Performance of `SafeSAC` ($\epsilon = 3$) with comparison to

Learn-to-Race benchmark results. In tables A.2 and A.3, we follow [107] in reporting on all of their driving quality metrics, for the `Learn-to-Race` benchmark: Episode Completion Percentage (ECP), Episode Duration (ED), Average Adjusted Track Speed (AATS), Average Displacement Error (ADE), Trajectory Admissibility (TrA), Trajectory Efficiency (TrE), and Movement Smoothness (MS).

We highlight the fact that such metrics as TrA, TrE, and MS are most meaningful for agents that *also* have high ECP results. Taking TrA, for example, safe policies score higher ECP values but may spend more time in inadmissible positions (as defined by the task, i.e., with at least one wheel touching the edge of the drivable area), compared to policies without a safety backup controller that may quickly terminate episodes by driving out-of-bounds (thus spending less time in the inadmissible positions). On the other hand, policies that have low completion percentages also have low ED scores, due to more frequent failures and subsequent environment resets.

We observe new state-of-the-art performance received by our approach, across the driving quality metrics, in the `Learn-to-Race` benchmark.