Ensembling Diverse Policies Improves Generalization of Deep Reinforcement Learning Algorithms to Environmental Changes in Continuous Control Tasks

by

Abilmansur Zhumabekov

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science
University of Alberta

Abstract

Deep Reinforcement Learning (DRL) algorithms have shown great success in solving continuous control tasks. However, they often struggle to generalize to changes in the environment. Although retraining may help policies adapt to changes, it may be quite costly in some environments. Ensemble methods, which are widely used in machine learning to boost generalization, have not been commonly adopted in DRL for continuous control applications. In this work, we introduce a simple ensembling technique for DRL policies with continuous action spaces. It aggregates actions by performing weighted averaging based on the uncertainty levels of the policies. We investigate its zero-shot generalization properties in a complex continuous control domain — the optimal control of home batteries in the CityLearn environment, the subject of a 2022 international AI competition. Our results indicate that the proposed ensemble has better generalization capacity than a single policy. Further, we show that promoting diversity among policies during training can reliably improve the zero-shot performance of the ensemble in the test phase. Finally, we examine the merits of the uncertainty-based weighted averaging in an ensemble by comparing it to two alternative approaches: unweighted averaging and selecting the action of the least uncertain policy.

Preface

This thesis is an original work by Abilmansur Zhumabekov. We submitted a paper based on this work to the Adaptive and Learning Agents Workshop at AAMAS 2023. A part of the reinforcement learning algorithm used as a baseline has been implemented in collaboration with Zhihu Yang (Research engineer at Zouyebang Education Technology Co. Ltd.) and Daniel May (Ph.D. student at the dept. of Electrical and Computer Engineering, University of Alberta) when participating in the NeurIPS 2022: CityLearn Challenge [40].

Acknowledgements

First, I would like to thank my supervisor, professor Matthew Taylor, for his wise guidance throughout my journey at the University of Alberta. Because of the recent pandemic and travel restrictions, I could not arrive in Edmonton for almost an entire academic year, but Matt's priceless support reached me far and well, regardless of my geographic location.

I also want to acknowledge professor Omid Ardakanian, whose precise and actionable advice, as well as pointers to papers and datasets, greatly pushed my research forward. Further, I want to thank Daniel May, who helped me understand numerous concepts from the domain of electric grid management, reviewed my code, bounced off ideas, shared his academic experience, and much more.

Next, I am grateful to the organizers of the 'NeurIPS 2022: CityLearn challenge,' the competition that greatly stimulated my development as an engineer and helped me form ideas for the research in this work. I am especially thankful to professor Zoltan Nagy for his quick responses to my questions about battery control, as well as his effort in making the real-world dataset from the challenge public. While participating in the competition, I had the pleasure of collaborating with Zhihu Yang, a research engineer at Zuoyebang Education Technology, from whom I learned a lot about coding and who became a great friend of mine.

In the 2022 winter term, I had an amazing experience working with Osmar Zaiane and Sadaf Ahmed, instructors of the CMPUT 175 course. By being great leaders and entrusting me with the lead TA role, they made a big contribution to my professional development.

Last but not least, I want to express immense gratitude to my family in Kazakhstan, who support me with endless love and care throughout my life.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

This chapter presents the motivation for this work, lists our contributions, and provides the layout of the thesis.

1.1 Motivation

Deep reinforcement learning (DRL) algorithms have attained remarkable performance in a variety of challenging continuous control tasks such as locomotion and manipulation [14, 19, 20]. However, DRL agents have limited generalization capabilities, tending to be overly specialized to their environment and failing to perform optimally when faced with perturbations [38, 44]. This is especially relevant for DRL agents trained in a simulator or digital twin for deployment in a real-world setting. The differences between the training and deployment (test) environment include state space, transition dynamics, observation function, etc. [44].

Closing this generalization gap is the focus of a broad body of research. For example, recent works have shown that generalization techniques from supervised learning, such as L2 regularization, dropout, data augmentation, and batch normalization, prove useful in DRL as well [24, 36].

Another generally accepted approach to boosting the generalization properties of machine learning (ML) models is to build ensembles of diverse models [8, 39]. Despite the prevalence of ensembling in the context of general ML, there remains a scarcity of

research exploring the use of (diverse) ensemble methods for continuous control tasks in DRL. In particular, their use for improving generalization to perturbations in the environment has been limited to date.

1.2 Our Contributions

This study introduces the 'Diverse σ-weighted ensemble' for continuous action spaces in DRL (see Chapter 3). We examine its zero-shot generalization properties on the data and task of the 2022 CityLearn Challenge [40] — household battery control for demand response, which is a challenging, partially observable continuous control task. Our key contribution is training *diverse* DRL policies and combining them according to their uncertainty in the given task. The main insights of this work can be summarized as follows:

1. Compared to using only a single policy, the proposed ensembling method performs significantly better in the test phase and resists overfitting for much longer. Here, overfitting denotes the phenomenon when the training cost is converging but the test cost is diverging as the training progresses. Overfitting can be quantified as the difference between training and test performances.

2. Promoting policy diversity in ensembles can significantly improve their zero-shot test performance, albeit the extent of improvement varies across different ensembling approaches.

3. The effectiveness of the proposed ensembling method comes from its ability to leverage diversity not only in the actions but also in the uncertainty levels of its members.

1.3 Thesis Outline

We start with the background chapter which introduces the battery control problem and the CityLearn simulator. Chapter 2 continues with a brief overview of the Reinforcement Learning framework used to formalize the task. We then review ensembles in ML broadly and in DRL more thoroughly, highlighting important differences from our work. Finally, the background concludes with an overview of the policy diversity method employed in this work to promote diversity in ensembles.

In Chapter 3, we propose the 'Diverse σ-weighted ensemble,' which combines the actions of multiple RL policies based on their uncertainty levels. The ensemble is called 'Diverse' because we promote diversity in policy outputs during training (see Chapter 2.5). Chapter 3 also details simple and interpretable metrics we employ to gauge policy diversity.

Chapter 4 follows by explaining the dataset and the cross-validation procedure used in all of our experiments. It then describes the training procedure and reward design for the DRL algorithm used in our work.

Next, the Results and Discussion chapter compares the zero-shot generalization properties of the 'Diverse σ-weighted' ensemble with the single policy approach. Crucially, we study the role of policy diversity on the ensemble's capacity to generalize. Finally, we study the benefits of using the σ-weighted (i.e., uncertainty-based) averaging method by comparing it to two alternatives: unweighted averaging and choosing the action of the least uncertain policy.

The thesis concludes with Chapter 6 that gives a summary of our work, lists its limitations, and proposes directions for future research.

Chapter 2

Background

In this chapter, we describe the demand response task for electric grids with renewable distributed energy sources. We then define the battery control problem in the CityLearn environment and briefly explain the reinforcement learning framework employed in this work. After reviewing the use of ensembles in ML and RL, we close the chapter by discussing policy diversity in RL.

2.1 Demand Response for Electric Grids

The adoption of distributed energy resources (DERs), such as solar panels and electric energy storage systems, can offset, shift, or reduce electricity and emission costs for the entire grid and individual customers. However, the intermittent nature of DER usage and generation patterns poses a significant challenge to the stability of the traditional grid [25]. One prominent approach to tackling this challenge is to employ *demand response* (DR). The US Department of Energy defines DR as "... changes in electric usage by end-use customers from their normal consumption patterns in response to changes in the price of electricity over time, or to incentive payments ..." [5]. DR approaches are broadly classified into direct DR (direct, external control of end-user's assets) or price-based DR, which uses real-time fluctuation of a monetary incentive signal to nudge end-user behavior.

Intelligent algorithms are needed to perform DR effectively. Given the success of

RL in other continuous control tasks, a body of research investigating the application of RL to DR has started to develop [29]. Here, we focus on price-based DR, in which homeowners aim to optimize their energy use and battery operation based on a given price signal which is an exogenous variable.

2.2 The Battery Control Task in CityLearn

To facilitate research on RL for DR, Vazquez et al. published the CityLearn environment [28] on the basis of OpenAI Gym [12]. Within CityLearn, we focus on the price-based DR task defined in the 2022 CityLearn Challenge [40]: controlling charging and discharging of a household battery, given the time-series input about the building's energy demand and solar generation, electricity pricing, carbon emission rate, as well as various weather signals (details provided in Chapter 2.3).

While CityLearn supports both building-level (single-agent) and district-level (multi-agent) objectives, we focus on the single-agent metrics in this work. Hence, the objective of each house is twofold: to minimize the electricity cost, as well as the carbon emission cost. Notice that minimizing the electricity cost is not equivalent to minimizing emissions, because electricity prices in today's electricity markets do not solely reflect the carbon intensity of power plants. The costs are defined as follows:

$$C_{price} = \sum_{t=1}^{T} C_{price}(t) = \sum_{t=1}^{T} p_t * (d_t - g_t + b_t)^+$$

$$C_{emission} = \sum_{t=1}^{T} C_{emission}(t) = \sum_{t=1}^{T} c_t * (d_t - g_t + b_t)^+$$

Where $C_{price}(t)$ is the electricity cost and $C_{emission}(t)$ is the emission cost. t is the time-step with the duration of one hour and T is the duration of the control task in hours. p_t and c_t are respectively the electricity pricing and carbon emission rates per unit of net energy demand (the expression inside brackets). d_t is the non-shiftable electricity demand of a household, g_t is the energy generated by its solar panels, and b_t

is the amount of energy charged into the battery (negative values imply discharging). The + superscript indicates that negative values are clipped to 0.

We adopt the normalized scoring employed in the 2022 CityLearn challenge [40]:

$$\begin{aligned} \hat{C}_{price} &= \frac{C_{price}}{C_{price}^{noop}}, \\ \hat{C}_{emission} &= \frac{C_{emission}}{C_{emission}^{noop}}, \\ \hat{C} &= \frac{1}{2}(\hat{C}_{price} + \hat{C}_{emission}), \end{aligned} \tag{2.1}$$

where C_{price}^{noop} and $C_{emission}^{noop}$ are respectively C_{price} and $C_{emission}$ with b_t set to 0, i.e., costs with no-battery or no control. $\hat{C}_{price}$ and $\hat{C}_{emission}$ are respectively the normalized electricity cost and the normalized emission cost. Finally, $\hat{C}$ is the 'building cost,' which we aim to minimize.

2.3 Reinforcement Learning

To apply RL techniques to the battery control problem, the task can be formulated as a partially observable Markov decision process (POMDP), which is a tuple $\langle S, \Omega, O, A, T, R \rangle$ [9].

S is the set of states s, where each state contains all the information necessary for choosing an optimal action a from the continuous set of actions A. The states are not directly accessible by the agent and have to be inferred from the observations o coming from the continuous set of observations Ω. The mapping from states to observations, sometimes conditional on the actions, is done by the observation function $O : S \times A \times \Omega \to [0; \infty)$. The transition function $T : S \times A \times S \to [0; \infty)$ represents the probability density of the next state $s_{t+1} \in S$ given the current state $s_t \in S$ and action $a_t \in A$. Finally, $R : S \times A \times S \to \mathbb{R}$ is the reward function.

An RL agent aims to maximize the expected return, the discounted sum of future rewards, by learning a policy π [22]. The policy π is a probabilistic mapping of observations $o \in \Omega$ to actions $a \in A$. Therefore the RL objective can be formulated

as follows:

$$J(\pi) = \mathbb{E}_{\pi}[\sum_{t=0}^{H} \gamma^t r_t(o_t, a_t)] \tag{2.2}$$

Where r_t is the reward for taking action a_t when observing o_t, $\gamma \in (0; 1]$ is the discount factor, and H is the duration of an episode.

If the RL agent uses a neural network to map observations to actions, it is called a deep reinforcement learning (DRL) agent. There exists a variety of algorithms to train DRL agents [17]. In this work, we employ Soft Actor-Critic (SAC) [20], an established DRL algorithm known for its relative robustness and sample efficiency.

In the CityLearn environment, observation $o \in \Omega$ is a vector with information about the system in the past hour: month, day of the month, the hour of the day, household electricity demand, solar generation, battery state of charge (SoC), net demand (electricity demand - solar generation + charging), and weather. Weather information consists of outdoor temperature, humidity, diffuse and direct solar irradiance, as well as their forecasted values for 6, 12, and 24 hours ahead. Detailed information on all observation features is given in Appendix A.1. Provided with an observation o, the DRL agent must choose an action $a \in [-1; 1]$ that determines the battery (dis)charging rate in the upcoming hour and that maximizes the RL objective in Equation 2.2. Rewards are carefully designed in Chapter 4.2 so that maximizing the RL objective corresponds to minimizing the costs that we care about (Equation 2.1).

We note that this is a partially observable task as o does not contain all the information necessary for picking an optimal action. For example, the agent does not see the physical properties of the battery and photovoltaic equipment, as well as indoor temperature and humidity. Moreover, the agent does not know the house occupants' plans for the future, which is essential for forecasting electricity consumption and choosing an optimal (dis)charging rate.

2.4 Ensembles

In this section, we discuss the use of ensembling techniques in machine learning and then review prior work on their applications in DRL.

2.4.1 Ensembles in Machine Learning

Ensembles are commonly used in machine learning (ML) to improve the generalization ability of models in tasks such as regression, classification, clustering, and more [21, 39]. Some of the popular ensembling approaches are Bagging and Stacking [10]. In Bagging, multiple learners are trained on different subsets of the training dataset, and their outputs are combined by averaging or voting at test time [2]. In Stacking, several models with different architectures are trained on the same data, and a meta-learner is used to combine their outputs in the test phase [3]. In addition, the meta-learner can be trained to identify which model is better suited for different parts of the feature space [1].

Ensembling methods are highly effective largely because they leverage some form of diversity [10]. Diversity may come from an auxiliary penalty term imposed on outputs, or from variations in training data, input representations, learning algorithms, etc. [27].

To develop intuition, as an example, let us consider a theoretical justification for the use of ensembles in the regression task and why diversity is important. In regression, we are given a training dataset $D = \{(x_1, y_1), ..., (x_n, y_n)\}$, where x_i are input vectors sampled from a distribution $P(X)$ and $y_i \in \mathbb{R}$ are labels. The labels are obtained from the target function f as follows: $y = f(x) + \epsilon$. Typically, ϵ is white noise, but for simplicity, we set it to 0 because it does not affect the conclusions that interest us [6, 10]. Further, h is the learner model trained on the inputs from D with the goal of minimizing the expected generalization error over the inputs x drawn from the entire distribution $P(X)$ [23]. The generalization error is often measured with the mean

squared error [8], and for an arbitrary input $x \sim P(X)$ can be written as:

$$GErr(h(x)) = \mathbb{E}_D[(h(x) - f(x))^2]$$

It can be shown [8, 10] that

$$GErr(h) = (\mathbb{E}[h] - f)^2 + \mathbb{E}[(h - \mathbb{E}[h])^2] \triangleq bias(h)^2 + variance(h) \tag{2.3}$$

where we drop the subscript D and the argument x for clarity. The bias is the expected deviation of $h(x)$ from $f(x)$, and the variance measures the variability of $h(x)$ depending on the training dataset D.

Now, let's consider an ensemble H of N learners $h_1, h_2, ..., h_N$, whose outputs are combined, without loss of generality, with unweighted averaging. Then the bias-variance decomposition in Equation 2.3 can be extended to this ensembling scenario as the bias-variance-covariance decomposition [4, 10]:

$$GErr(H) = \overline{bias}(H)^2 + \frac{1}{N}\overline{variance}(H) + (1 - \frac{1}{N})\overline{covariance}(H) \tag{2.4}$$

Where $\overline{bias}$ is the average bias of learners $h_1, h_2, ..., h_N$:

$$\overline{bias}(H) = \frac{1}{N}\sum_{i=1}^{N}(\mathbb{E}[h_i] - f)$$

For the average bias to be small, the individual learners should have small biases. Additionally, different learners can reduce the ensemble's generalization error by having biases of different signs (i.e., positive and negative) that (partially) cancel each other out.

Next, $\overline{variance}$ is the average variance of the learners:

$$\overline{variance}(H) = \frac{1}{N}\sum_{i=1}^{N}\mathbb{E}(h_i - \mathbb{E}[h_i])^2$$

It is better if all learners have low variances, but their importance decreases as the ensemble size N increases.

Finally, $\overline{covariance}$ is the average covariance of different learners:

$$\overline{covariance}(H) = \frac{1}{2N(N-1)}\sum_{i=1}^{N}\sum_{j=i+1}^{N}\mathbb{E}[h_i - \mathbb{E}[h_i]]\mathbb{E}[h_j - \mathbb{E}[h_j]]$$

Like the average bias, this term is indicative of the role of diversity. When the learners are similar, they make correlated errors which make the $\overline{covariance}$ positive. Conversely, when the learners make uncorrelated or negatively correlated errors, then $\overline{covariance}$ is 0 or negative. Therefore, encouraging diversity among learners $h_1, h_2, ..., h_N$ can improve generalization by reducing the $\overline{covariance}$ term in Equation 2.4 [10].

While the benefits of using diverse ensembles have been established for many ML tasks [21, 39], their application to RL is underexplored.

2.4.2 Ensembles in Deep Reinforcement Learning

In this section, we summarize existing works investigating ensembling techniques for DRL algorithms.

An et al. [32] use an ensemble of Q-networks in an offline RL setting. They estimate the Q value of a state-action pair by choosing the minimal value outputted by the set of Q-networks. It leads to the penalization of out-of-distribution actions for which there is high uncertainty in Q-value estimates. The authors show that this ensembling approach outperforms existing offline RL methods. Further, they show that promoting diversity in gradient updates of Q-networks significantly reduces the required ensemble size.

Ensembling both critics and actors proves useful in stabilizing learning and improving exploration during training, according to Lee et al. [34], where the mean and standard deviation of Q-value estimates are used to reweight Bellman backups and to perform UCB exploration.

Yang et al. [31] use three different DRL algorithms in an ensemble: PPO [18], A2C [16], and DDPG [15] to trade stock shares. In each quarter, only one of the algorithms is used to trade, but all three can be evaluated in the background. The algorithm with the best evaluation score is selected to trade in the next quarter. According to the authors, different models are sensitive to different trends, so ensembles should work

better than any of their members alone. Although their experiments demonstrate the effectiveness of their ensembling strategy, it has a limiting assumption that evaluation scores can be computed for the algorithms that did not participate in trading.

Finally, Ghosh et al. [33] show that ensembles can improve the generalization performance of RL agents. Their method of combining actions is shown to work for discrete action spaces, but transferring it to continuous action spaces is a non-trivial task.

2.5 Policy Diversity

Established ensembling methods in ML are highly effective largely because they leverage some form of diversity, which may come from an auxiliary penalty term imposed on outputs or from variations in training data, input representations, learning algorithms, etc. [10, 21, 27]. For this reason, one of the goals of this paper is to investigate the effect of policy diversity on the DRL ensemble's generalization capacity.

In RL, diversity can stem from variations in the environment or the agent behaviors (policies) [41]. In this paper, we focus on policy diversity, which can be quantified by measuring the difference between trajectories (state-action or observation-action sequences) traversed by the policies [26, 35] or by evaluating the disparity in policy actions when provided with the same states/observations [30, 41, 43].

In our study, we employ the Diversity via Determinants (DvD) method proposed by Parker-Holder et al. [30]. It adds an auxiliary diversity term to the objective, which encourages policies to output diverse actions when provided with the same observations:

$$J(\phi_1, \phi_2, ..., \phi_N) = \sum_{i=1}^{N} \mathbb{E}_{\tau \sim \pi_{\phi_i}}[R(\tau)] + \lambda Div(\phi_1, \phi_2, ..., \phi_N), \tag{2.5}$$

where $\phi_1, ..., \phi_N$ are parameters of N policies, τ is the trajectory traversed by a policy in an episode, and R is the return (discounted sum of rewards). Importantly, the diversity term $Div(\cdot)$ captures the volume spanned by policies in a behavioral

manifold. In other words, it measures the degree to which outputs of different policies are different from one another when faced with the same observations. For more details, we refer the reader to [30].

One of the benefits of DvD is that it is task-agnostic, meaning it does not require hand-crafting policy representations for a specific domain. Moreover, it allows tuning the degree of diversity by controlling λ — the importance coefficient of the diversity objective. Last but not least, it is easy to implement thanks to a reference implementation [37].

Chapter 3

Diverse σ-weighted Ensembling Technique

When training a DRL agent on a given environment, only one policy π_ϕ is typically learned. With SAC, as with most actor-critic models, the actor's policy is defined as $\pi_\phi = \langle \mu_\phi, \sigma_\phi \rangle$, meaning the actor is modeling a Normal distribution $\mathcal{N}$ with characterizing parameters mean μ_ϕ and standard deviation σ_ϕ. During training, the agent samples this distribution stochastically so that

$$\hat{a}_{train}(o_t) \sim \mathcal{N}(\mu_\phi(o_t), \sigma_\phi(o_t))$$

while during evaluation (test), actions are deterministically selected:

$$\hat{a}_{eval}(o_t) = \mu_\phi(o_t)$$

In order to constrain actions, SAC further applies *tanh* function as well as scaling [20]:

$$a_{train} = c_a tanh(\hat{a}_{train})$$

$$a_{eval} = c_a tanh(\hat{a}_{eval})$$

The action scaling coefficient c_a is a hyperparameter (for details on all hyperparameters, see Appendix A.2). Together with the scaling coefficient, *tanh* function puts the actions into $[-c_a; c_a]$ range. We refer to this procedure as the 'Single Policy' approach and depict its training and evaluation pipeline in Figure 3.1.

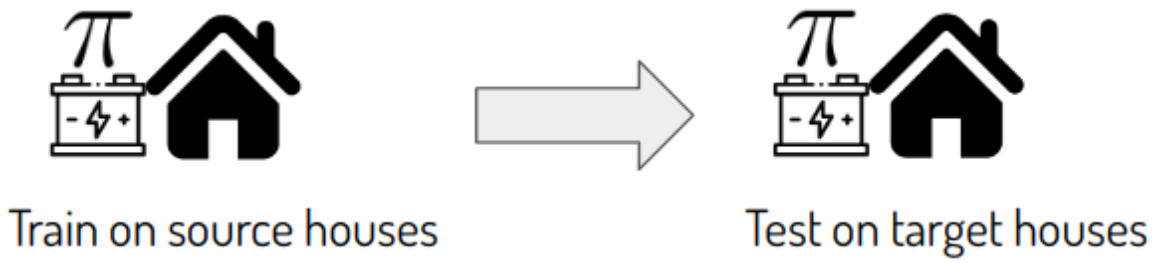


Figure 3.1: Single Policy Training and Evaluation Process.

To improve zero-shot generalization on a perturbed environment, we propose an ensembling method for continuous action spaces. We train multiple actors (with one shared critic) in separate (but identical) environments and aggregate them in an ensemble during the test phase. The ensemble's output is a weighted average of its individual members' actions, where each weight is inversely proportional to the degree of uncertainty of the policy. We use $\sigma_\phi(o)$ as a proxy for this uncertainty.

More concretely, we train N policies that could be represented as follows:

$$\pi_{\phi_i} = \langle \mu_{\phi_i}, \sigma_{\phi_i} \rangle, \text{ for } i = 1, 2, ..., N$$

During training, each policy's action is sampled stochastically and executed in a separate training environment:

$$\hat{a}^i_{train}(o_t) \sim \mathcal{N}(\mu_{\phi_i}(o_t), \sigma_{\phi_i}(o_t)),$$
$$a^i_{train} = c_a tanh(\hat{a}^i_{train})$$

All N actors are trained in parallel, and their loss is augmented with the diversity term [30] discussed in Chapter 2.5.

During evaluation (test), we combine the outputs of these policies into one action that is executed in the test environment:

$$a^\sigma_{eval}(o_t) = \frac{\sum_{i=1}^N w_i a^i_{eval}(o_t)}{\sum_{i=1}^N w_i}, \tag{3.1}$$

where $a^i_{eval}(o_t) = c_a tanh(\mu_{\phi_i}(o_t))$ and $w_i = \frac{1}{\sigma_{\phi_i}(o_t)}$. This approach is illustrated in Figure 3.2, and further referred to as '(Diverse) σ-weighted ensemble.' The motiva-

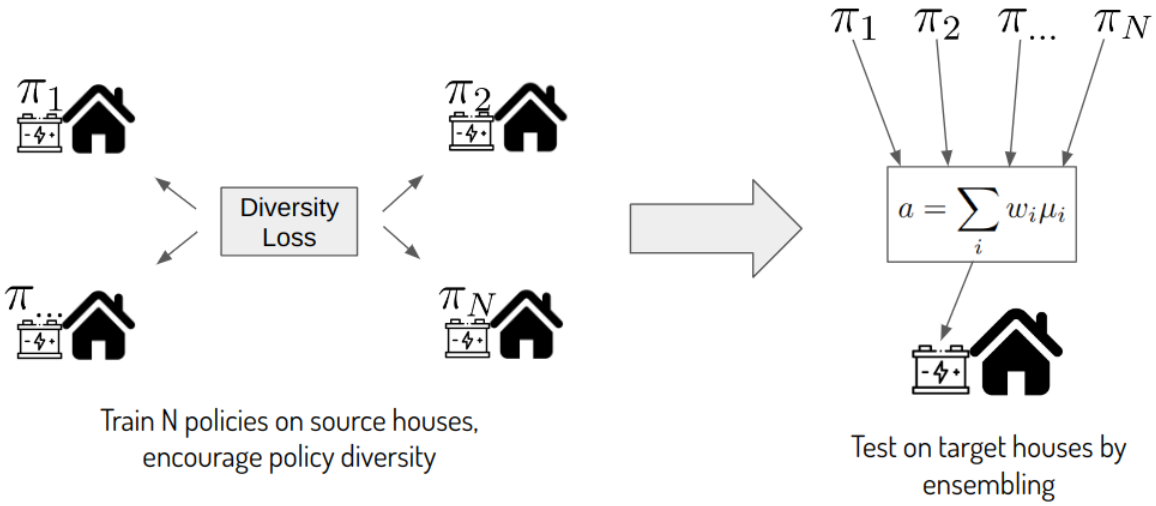


Figure 3.2: Ensemble Policy Training and Evaluation Process.

tion behind this weighting of actions is that standard deviations $\sigma_{\phi_i}(o_t)$ measure the uncertainties of their corresponding policies π_{ϕ_i}. Distinct policies go through different experiences and updates during training, so they might have varying degrees of certainty in their actions when faced with an observation o_t in the test environment. This disparity can increase further when policy diversity is promoted during training. Thus, by using standard deviations, we are taking into account the confidence levels of different policies, which, as experiments reveal, leads to better performance at test time.

To confirm that policies in the 'Diverse' ensemble generate more diverse actions, compared to the 'Non-diverse' ensemble, we calculate their standard deviation:

$$\mathcal{D}^a(o_t) = \sqrt{\frac{\sum_{i=1}^{N}(a^i_{eval}(o_t) - \bar{a}_{eval}(o_t))^2}{N}}, \tag{3.2}$$

where $\bar{a}_{eval}(o_t)$ is the mean of the actions chosen by the policies in an ensemble:

$$\bar{a}_{eval}(o_t) = \frac{1}{N}\sum_{i=1}^{N} a^i_{eval}(o_t). \tag{3.3}$$

Furthermore, policy diversity can also manifest in the diversity of uncertainties among policies. To measure it, we calculate the coefficient of variation of σ_{ϕ_i} values:

$$\mathcal{D}^\sigma(o_t) = \frac{1}{\bar{\sigma}(o_t)}\sqrt{\frac{\sum_{i=1}^{N}(\sigma_{\phi_i}(o_t) - \bar{\sigma}(o_t))^2}{N}}, \tag{3.4}$$

where $\bar{\sigma}(o_t) = \frac{1}{N}\sum_{i=1}^{N}\sigma_{\phi_i}(o_t)$ is the average of uncertainties.

The coefficient of variation is the standard deviation of σ_{ϕ_i} values divided by their mean. Therefore, this metric is independent of the average uncertainty level (scale of σ_{ϕ_i} values). We cannot use the coefficient of variation for measuring action diversity $\mathcal{D}^a$ because a^i_{eval} can be negative, but using the standard deviation metric is acceptable since the action values are tightly bounded: $a \in \{-c_a; c_a\}$.

In the Results and Discussion (Chapter 5) we report values $\mathcal{D}^a$ and $\mathcal{D}^\sigma$ that are averages of $\mathcal{D}^a(o_t)$ and $\mathcal{D}^\sigma(o_t)$ across all target (test) buildings and over the entire test episode (see Chapter 4.1). We choose these metrics because they are easy to interpret and implement, and they give insights into the benefits of our proposed ensembling method (see Chapter 5.3).

To the best of our knowledge, the 'Diverse σ-weighted ensemble' is novel. While Lee et al. [34] use their ensemble to improve training stability and exploration, we use ours to improve zero-shot generalization to the test environment that will be different from the training environment. Unlike An et al. [32], our focus is on the case when the agent is allowed to learn online during training (i.e., influence the environment and receive feedback). In contrast to Yang et al. [31], our 'Diverse σ-weighted' ensembling approach does not require evaluating all policies and picking one of them. Instead, it simply combines all of their actions with weighted averaging. Finally, Ghosh et al. [33] show that DRL ensembles can improve generalization in tasks with discrete action spaces, whereas our work focuses on continuous action spaces and examines the effect of policy diversity on the ensemble's generalizability.

Chapter 4

Methodology and Experimental Procedure

In this chapter, we describe our experimental setup by providing details on the dataset and its train-test split in Chapter 4.1. We then explain the reward design and training procedure in Chapter 4.2.

4.1 Dataset and Cross-Validation

We use the dataset from CityLearn 2022 challenge [40], which contains 1-hour resolution data for a period of 1 year obtained from a neighborhood of 17 single-family houses in Fontana, California [45]. After examining the hourly power consumption profiles of each building [42] and discussing them with the dataset's publishers, we decided to omit 2 buildings (numbered 12 and 15) with highly abnormal consumption profiles. These abnormalities could have resulted from malfunctioning measurement equipment. Next, to perform cross-validation, the remaining 15 buildings were partitioned into 3 groups of 5 buildings each: the first group (buildings 1 through 5), the second group (buildings 6 through 10), and the third group (buildings 11, 13, 14, 16, 17).

In all experiments, to attain statistically significant results, we perform 3-fold cross-validation with 5 independent trials in each. For every fold, we train an algorithm on one group (5 source buildings) and test on the remaining two groups (10 target

buildings). We perform statistical comparisons using the Mann-Whitney U test (also called the Wilcoxon rank-sum test) [7].

We further adopt the most difficult deployment setting from Nweye et al. [42], restricting training to the first 5 months of data and performing testing on the remaining 7 months. This setup mimics a to-scale deployment scenario from an accurately simulated training environment with 'few' data streams to a real environment with many data streams.

For each experiment, we report the zero-shot performance on the 7 months of the target building data in terms of metrics established in Chapter 2.2, averaged across all folds and trials (15 samples).

4.2 Training Procedure and Reward Design

First, it is important to clarify how training episodes are counted. For a single policy, one training episode is equivalent to one pass through the first 5-month of data for 5 source buildings. For ensembles of size N (e.g., N=4), when each ensemble member goes through the same data once, we count it as N training episodes completed by the ensemble. While counting training episodes may seem involved, one test episode simply corresponds to one pass through the 7-month data for 10 target buildings.

On a related note, our SAC algorithm performs random exploration at the beginning of training, when it samples actions randomly from a uniform distribution and saves resulting transitions to the replay buffer. The duration of that period must be standardized for ensembles of different sizes. In our experiments, we consider ensembles of sizes N=1, 2, 4, 8 — where N=1 corresponds to the single policy. When the biggest ensemble of size N=8 goes through the training data once — its members gather 8 episodes of cumulative random experience. To ensure a fair comparison, each ensemble must collect the same amount of cumulative random exploration experience. This is achieved by fixing the number of exploration episodes at 8.

Finally, we describe our reward function that encourages minimization of the cost

in Equation 2.1. It consists of price and emission components:

$$r_t^{price} = C_{price}^{noop}(t) - C_{price}(t),$$
$$r_t^{emission} = C_{emission}^{noop}(t) - C_{emission}(t)$$

At the end of the random exploration period, we calculate the means and standard deviations of observations and rewards (separately for each component) and use these to normalize them. Normalization is finalized by scaling the reward up by a factor of c_r (see Appendix A.2).

Normalized reward terms $\hat{r}_t^{price}$ and $\hat{r}_t^{emission}$ are then combined into the building reward:

$$r_t = \frac{1}{2}(\hat{r}_t^{price} + \hat{r}_t^{emission})$$

We put this reward into the RL agent's objective (Equation 2.2). Thus, the agent's goal is to maximize the expected sum of future building rewards, which, by design, encourages the minimization of the building cost (Equation 2.1).

To sum up, this chapter provided details about the cross-validation method and the dataset, as well as explained the training procedure and reward design. These methods are employed throughout our experiments presented in the next chapter.

Chapter 5

Results and Discussion

We now present and discuss the results of our experiments. We start with Chapter 5.1, which examines the generalization capability of the σ-weighted ensembling method (See Chapter 3), in both diverse and non-diverse settings, comparing it to the canonical 'Single Policy' approach (Figure 3.1). For the sake of comparison, we also use two rule-based controllers (RBCs) as baselines:

- RBC_{ToU} - The Time-of-Use Peak Reduction strategy that has been deployed in real life on the majority of houses from the dataset [42]. It charges the battery from 9 am to 12 pm and discharges from 6 pm to 9 am. Both charging and discharging rates are 2kW/h (31.25% of battery capacity). Discharging is only allowed when the battery is at least 25% full.

- RBC_{HC} - The Hand-Crafted controller of our design. We used its slightly modified version as a part of our solution when participating in the CityLearn 2022 challenge [40]. Its implementation details are given in Appendix A.3.

We then study the benefits of performing σ-weighted averaging of policy actions in Chapter 5.3 by comparing it with two alternative action selection mechanisms:

- Simple-averaging — combining actions using the unweighted average from Equation 3.3.

- Min-σ — selecting only one action $\mu_{\phi_i}(o_t)$ with the smallest $\sigma_{\phi_i}(o_t)$ and ignoring the rest.

All experiments are conducted with an ensemble size of $N = 4$. Refer to Chapter 5.2 for more details on the choice of N.

5.1 Diverse Ensembles of DRL policies

In this experiment, using the Single Policy approach (Figure 3.1) as a baseline, we examine zero-shot generalization capabilities of the 'Diverse σ-weighted' ensemble proposed in Chapter 3. We also compare it to its non-diverse ablation, labeled 'Non-diverse σ-weighted' ensemble, to study the role of policy diversity.

Figure 5.1 shows zero-shot costs (lower is better) on target buildings plotted against the number of training episodes completed on source buildings for each approach. The shaded areas span standard error over 15 trials from the validation procedure described in Chapter 4.1, while the lines denote the averages. Since we evaluate test scores after every pass through the 5-month training data, ensembles of size N have values only for every Nth training episode completed (see Chapter 4.2). For reader's reference, the cost achieved by an optimal controller with complete knowledge of future is 0.70.

Table 5.1 shows the zero-shot costs averaged across target buildings and cross-validation folds. We evaluate the test costs at different stages of learning — after the 40th, 80th, and 120th episodes of training on source buildings. From Figure 5.1 we notice that the test costs of all methods are unstable at the initial stage of training. They are remarkably low after the first N training episodes that follow 8 random-exploration episodes, so we include the test costs obtained after the 8+Nth training episode as well. We mark in bold the results for which ensembles outperform the Single Policy approach with $p \leq 0.05$ when comparing with Single Policy's every column. We also underline the cases when one method outperforms the others in

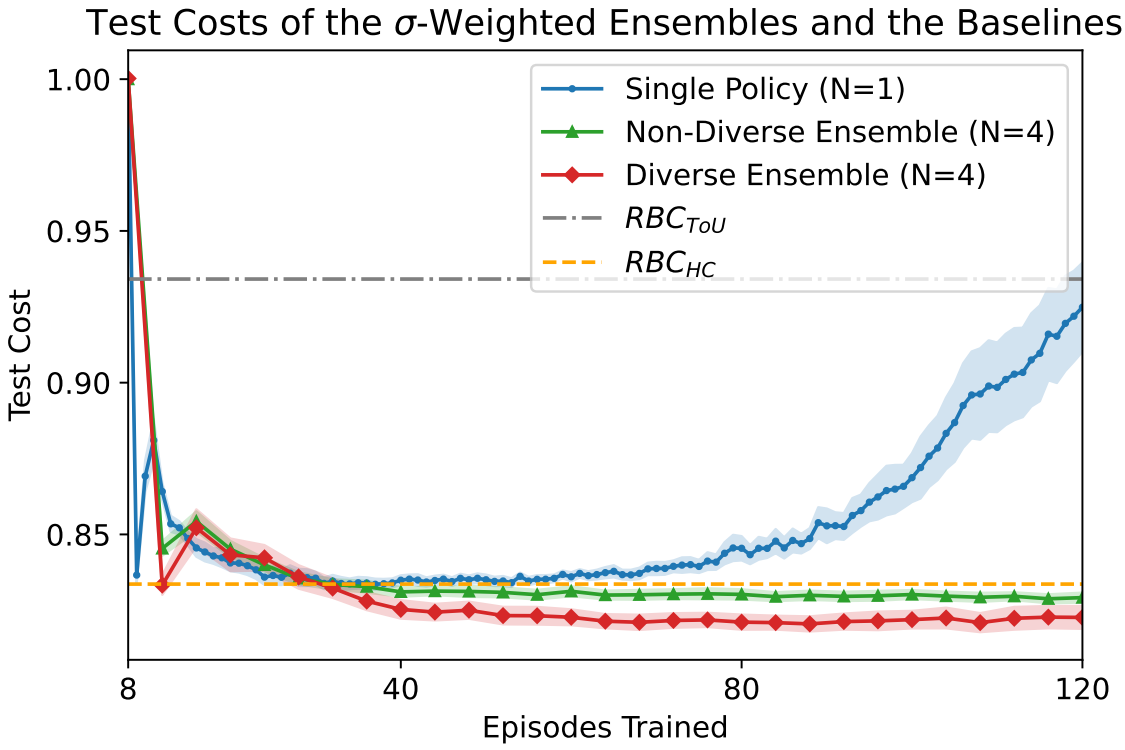


Figure 5.1: Zero-shot costs averaged across target buildings, comparing Diverse and Non-diverse σ-weighted ensembles to the baselines. Shaded regions denote the standard error. All policies train on the first 5 months of the source buildings data and are tested on the remaining 7 months of the target buildings data. The ensembles achieve lower test costs compared to the Single Policy and resist overfitting for longer. The Diverse Ensemble outperforms its Non-Diverse counterpart.

Table 5.1: For the σ-weighted ensembles and the Single Policy: zero-shot test costs (in %) on target buildings obtained after 8+Nth, 40th, 80th, 120th episodes of training on source buildings. In bold: an ensemble outperforms the Single Policy with $p \leq 0.05$ within any column. Underlined: the diverse ensemble outperforms the other methods with $p \leq 0.05$ within any column.

Method	Episodes Trained			
	8+N	40	80	120
Single Policy (N=1)	83.66	83.49	84.54	92.49
Non-diverse ensemble (N=4)	84.54	**83.11**	**83.02**	**82.92**
Diverse ensemble (N=4)	83.33	**82.53**	**82.11**	**82.26**

every column (e.g., the diverse ensemble evaluated after 40 episodes outperforms other methods evaluated after 8+N, 40, 80, and 120 episodes).

We note that the Non-diverse σ-weighted ensemble converges to lower costs compared to the Single Policy and resists overfitting to training data for much longer.

Further, the diverse ensemble outperforms its non-diverse counterpart regardless of the duration of the training with a statistical significance of $p \leq 0.05$, demonstrating that policy diversity further improves the zero-shot generalization ability of the σ-weighted ensemble.

To support the claims above, in Figure 5.2, we plot the training costs achieved throughout the training process. During the initial random exploration phase (see Chapter 4.2 for details), the training costs are very high, so we omit them in the plot for a better comparison. We note that, as expected, the training costs decrease monotonically for all methods. This stands in contrast to the Single Policy's test cost, which noticeably diverges after 40 episodes, while both σ-weighted ensembles maintain low test cost values even after 120 episodes of training (Figure 5.1). These observations confirm that the ensembles exhibit higher resistance against overfitting to training data compared to the Single Policy approach.

With respect to training costs, the Diverse and Non-diverse ensembles perform

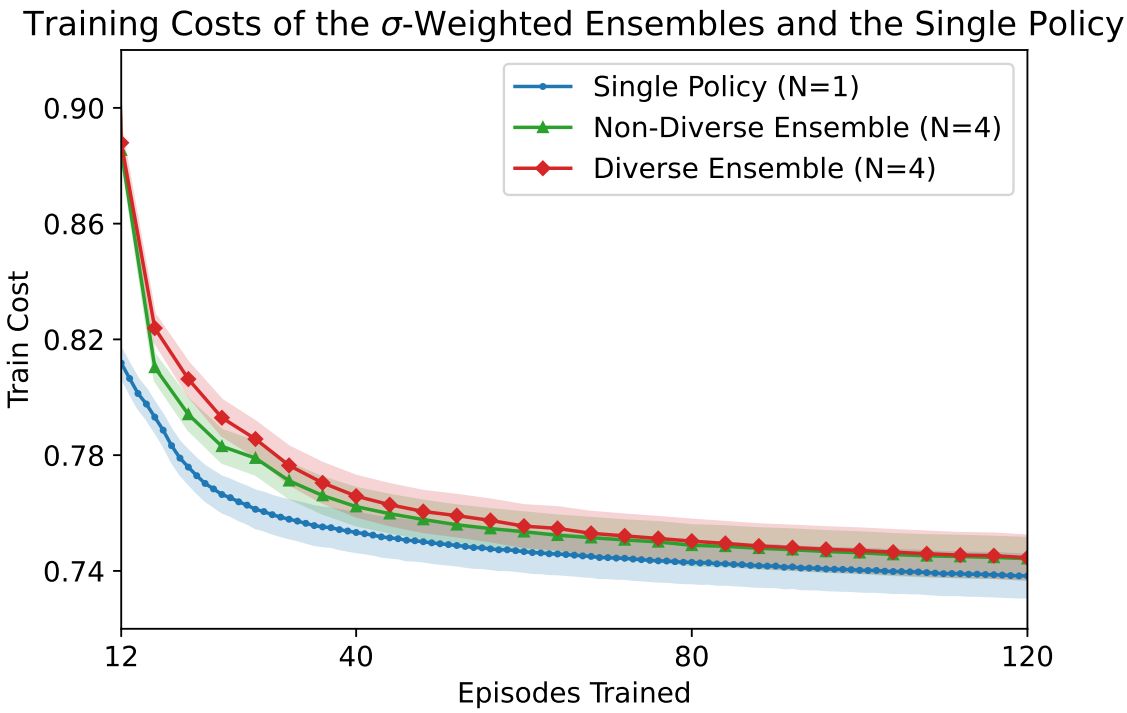


Figure 5.2: Training costs averaged across source buildings, comparing Single Policy vs. Diverse and Non-diverse σ-weighted ensembles. Shaded regions denote the standard error. All agents train on the first 5 months of source building data and, for this plot, are evaluated on the same data. The ensembles do not outperform the Single Policy on training data.

Table 5.2: Comparison of diverse ensembles of different sizes. Zero-shot test costs (in %) on target buildings obtained after 8+Nth, 40th, 80th, and 120th episodes of training on source buildings. In bold: an ensemble outperforms the Single Policy with $p \leq 0.05$ when compared to any column.

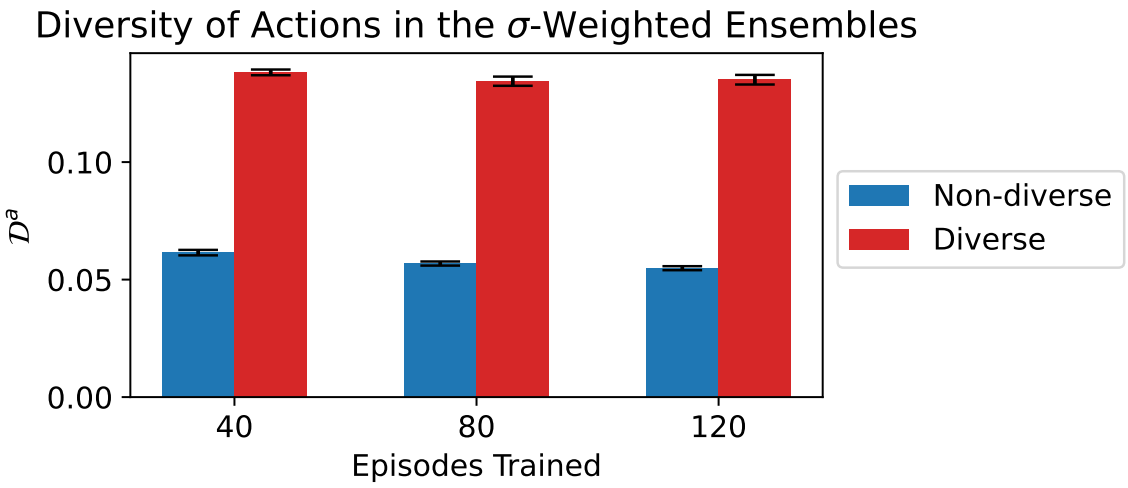
Method	Episodes Trained			
	8+N	40	80	120
single policy (N=1)	83.66	83.49	84.54	92.49
diverse ensemble (N=2)	83.19	**82.71**	**82.46**	84.76
diverse ensemble (N=4)	83.33	**82.53**	**82.11**	**82.26**
diverse ensemble (N=8)	**82.74**	**82.86**	**81.95**	**81.76**

equally, and both do worse than the Single Policy approach (Figure 5.2). Comparing that to Figure 5.1 further affirms that the differences in test costs do not come from the differences in training costs.

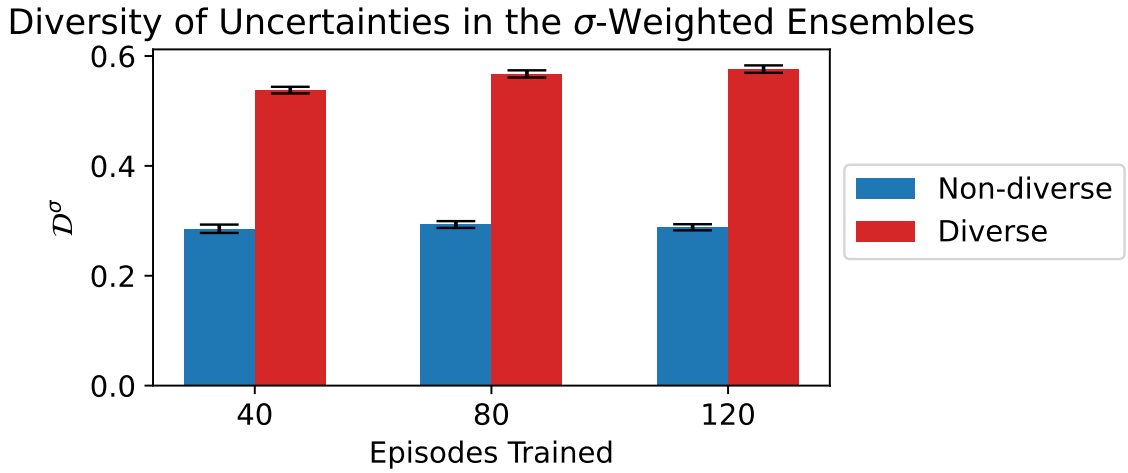
To confirm that policies in the diverse ensemble indeed output more diverse actions, we plot the diversity metric for actions $\mathcal{D}^a$ (Equation 3.2) in Figure 5.3a for both 'Diverse' and 'Non-diverse' σ-weighted ensembles of size N=4. From this plotting, it can be seen that policies in the diverse ensemble differ in their decisions much more than policies in the non-diverse ensemble. Similarly, in Figure 5.3b we illustrate the diversity in uncertainty levels $\mathcal{D}^\sigma$ (Equation 3.4) in diverse and non-diverse ensembles. We notice that policies in the diverse ensemble have greater variation in their uncertainties as well.

5.2 Effect of Ensemble Size

Figure 5.4 and Table 5.2 show the effect of varying the ensemble size. As we can see, the diverse ensembles retain their merits regardless of their size, although the smallest ensemble of size N=2 starts diverging earlier. Further, statistical comparison between ensembles of different sizes confirms that the ensemble of size N=2 underperforms bigger ensembles after the 120th training episode ($p \leq 0.05$). In contrast,



(a) Diversity in Policy Actions



(b) Diversity in Policy Uncertainties

Figure 5.3: Comparison for (a) diversity in actions a^i_{eval} and (b) diversity in uncertainties σ_{ϕ_i}, averaged over 15 trials, for policies in 'Non-diverse' and 'Diverse' σ-weighted ensembles. The error bars denote the standard errors. The 'Diverse' ensemble exhibits higher diversity in both actions and uncertainties.

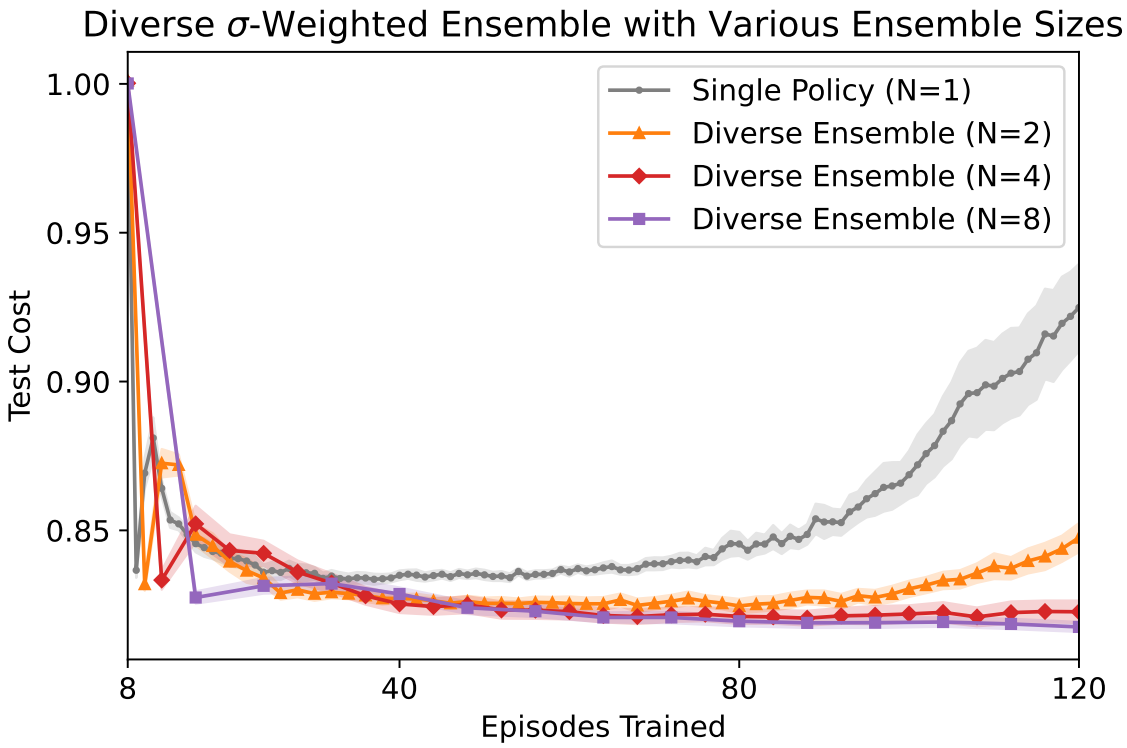


Figure 5.4: Zero-shot costs averaged across target buildings, achieved by single policy vs. diverse ensembles of various sizes. Shaded regions denote the standard error. The Diverse σ-weighted ensembles outperform the Single Policy regardless of the ensemble size, but N=2 starts diverging earlier than N=4 and N=8.

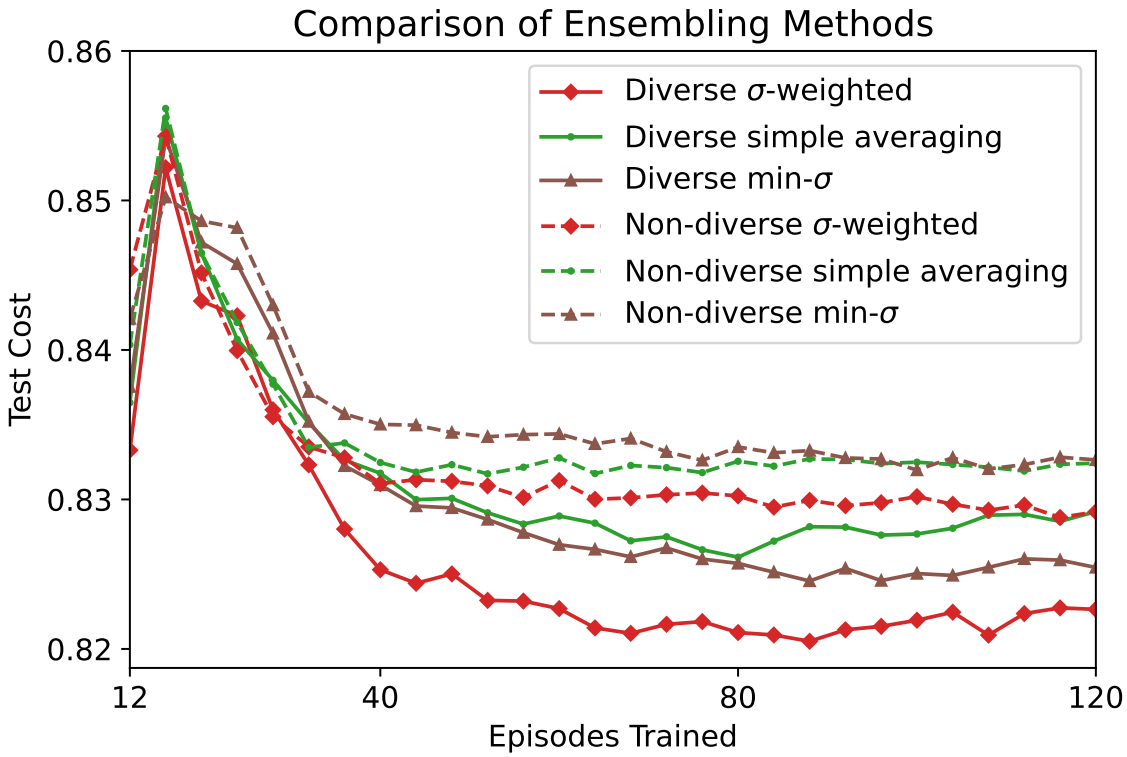


Figure 5.5: σ-weighted averaging vs. Simple-averaging vs. Min-σ action selection. The plot suggests that the σ-weighted ensemble achieves the lowest test costs in both 'Diverse' and 'Non-diverse' settings, and that diversity is helpful to all ensembling methods.

the difference between ensembles of size N=4 and N=8 is not statistically significant ($p > 0.05$) for all episodes considered in Table 5.2. Since N=8 does not provide significant benefits over N=4, we perform all other experiments with the ensemble size N=4.

5.3 Comparison of Ensembling Methods

In this section, we investigate the effect of the ensembling method choice and its role in leveraging policy diversity. To do so, we compare the σ-weighted ensembling technique to two baselines introduced at the beginning of Chapter 5, the Simple-averaging ensemble and the Min-σ ensemble, in the diverse and non-diverse setting.

Figure 5.5 shows the test costs of these approaches averaged over 15 trials. We do

Table 5.3: Zero-shot test costs of different ensembling methods when trained with and without diversity, obtained after 8+Nth, 40th, 80th, and 120th episodes of training. In bold: a diverse ensemble outperforms its non-diverse version with $p \leq 0.05$ within the same column.

Method	Episodes Trained			
	8+N	40	80	120
Non-diverse Simple-averaging	84.03	83.25	83.26	83.24
Non-diverse Min-σ	84.21	83.50	83.35	83.26
Non-diverse σ-weighted	84.54	83.11	83.02	82.92
Diverse Simple-averaging	83.65	83.18	**82.61**	**82.92**
Diverse Min-σ	83.75	**83.10**	**82.57**	**82.54**
Diverse σ-weighted	**83.33**	**82.53**	**82.11**	**82.26**

not shade the standard errors to avoid clutter. To focus on the differences between each approach, we skip plotting the test cost evaluated after 8 random exploration episodes (where all methods get a cost of about 1). The plot suggests that all methods benefit from enhanced policy diversity and that σ-weighted ensembles achieve lower zero-shot test costs compared to the alternatives in both diverse and non-diverse training scenarios.

Table 5.3 compares zero shot costs of the tested ensembling methods. We boldface the cases where a diverse ensemble outperforms its non-diverse version with $p \leq 0.05$. From both the table and Figure 5.5 it is clear that Min-σ and σ-weighted ensembles are better at leveraging diversity than the Simple-averaging method.

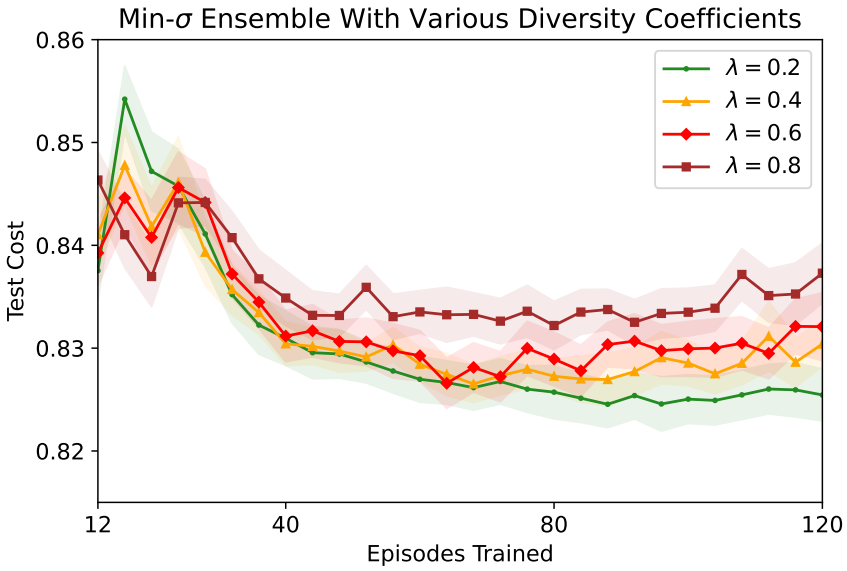
Further statistical analysis shows that the diverse σ-weighted ensemble significantly outperforms ($p \leq 0.05$) the diverse Simple-averaging method when tested after 40, 80, and 120 training episodes. The only difference between these approaches is that σ-weighted averaging leverages the diversity in uncertainty levels σ_{ϕ_i} among ensemble members π_{ϕ_i}, while Simple-averaging does not. Therefore, it is reasonable to conclude that leveraging the diversity of uncertainties in an ensemble improves the

generalization performance.

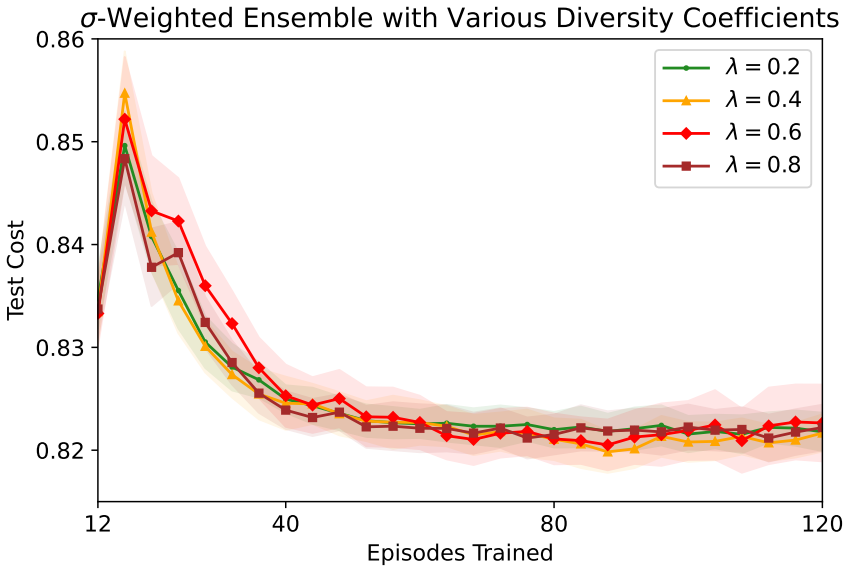
Moreover, Table 5.3 suggests that diverse Simple-averaging successfully outperforms its non-diverse counterpart when tested after long training but not so after shorter periods of training. It seems that exploiting the diversity in actions alone, without accounting for uncertainties, has a positive but limited effect on generalization. In contrast, the diverse σ-weighted method, which leverages diversity in both actions and uncertainties, outperforms its non-diverse counterpart more consistently. These results indicate that leveraging the variations in both actions and uncertainties (Figure 5.3) is important and that diverse σ-weighted averaging gains boosts in zero-shot test performance from both.

Next, statistical comparison of diverse σ-weighted and diverse Min-σ approaches does not report a significant difference in their performance. However, we note that these results are given for the best λ (importance coefficient of the DvD diversity term, as described in Chapter 2.5) for each ensemble type, found from the search space $\lambda \in \{0.2, 0.4, 0.6, 0.8\}$. Details on λ values for each ensemble are given in Appendix A.2. Figure 5.6 compares test costs of Min-σ and σ-weighted ensembles under different values of λ. From the plots, it is evident that the σ-weighted ensemble is more robust to changes in the λ hyperparameter. This outcome suggests the importance of considering the outputs of all policies, not just the most confident one.

To sum up, this subsection shows that the σ-weighted ensemble reliably outperforms the alternatives by leveraging the diversity in both actions and uncertainties of all of its members. Crucially, the disparity found in zero-shot generalization properties of these few ensembling approaches prompts further research into a more extensive set of ensembling techniques.



(a) Min-σ ensemble



(b) σ-weighted ensemble

Figure 5.6: Comparison of Min-σ and σ-weighted ensembles under different diversity importance coefficients λ. Shaded regions denote the standard error. The plots suggest that the σ-weighted ensembling method is more robust to the choice of the λ hyperparameter.

Chapter 6

Conclusions and Future Work

In this work, we proposed the 'Diverse σ-weighted ensemble' of DRL policies for continuous control tasks, which weighs the actions of its members based on their degrees of uncertainty. We then performed experiments on a realistic battery control task in CityLearn. First, we showed that the proposed ensemble can improve zero-shot generalization to environmental changes in continuous control tasks. Next, we demonstrated that promoting policy diversity in ensembles significantly and reliably improves test performance further. Lastly, we found that the effectiveness of the Diverse σ-weighted ensemble stems from its ability to leverage diversity in both actions and uncertainties of all of its members.

Future work will focus on extending our experiments to other continuous control benchmarks with various types of environmental changes. In addition, it is important to compare the σ-weighted ensembling method with a bigger set of ensembling techniques and deeper explore the role of DRL algorithm choice, critic centralization, and ensemble member diversity.

Bibliography

[1] D. H. Wolpert, "Stacked generalization," *Neural networks*, vol. 5, no. 2, pp. 241–259, 1992.

[2] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, pp. 123–140, 1996.

[3] L. Breiman, "Stacked regressions," *Machine learning*, vol. 24, pp. 49–64, 1996.

[4] N. Ueda and R. Nakano, "Generalization error of ensemble estimators," in *Proceedings of International Conference on Neural Networks (ICNN'96)*, IEEE, vol. 1, 1996, pp. 90–95.

[5] U. D. o. Energy, "Benefits of demand response in electricity markets and recommendations for achieving them," Lawrence Berkeley National Laboratory, Berkeley, CA, USA, Tech. Rep., 2006.

[6] V. Sethu, *The bias-variance tradeoff*, https://www.inf.ed.ac.uk/teaching/courses/mlsc/Notes/Lecture4/BiasVariance.pdf, Accessed: 2023-02-19, 2007.

[7] N. Nachar, "The Mann-Whitney U: A test for assessing whether two independent samples come from the same distribution," *Tutorials in quantitative Methods for Psychology*, vol. 4, no. 1, pp. 13–20, 2008.

[8] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning: Data mining, inference, and prediction* (Springer Series in Statistics), 2nd. New York, NY, USA: Springer Publishing, 2009.

[9] M. T. Spaan, "Partially observable Markov decision processes," in *Reinforcement Learning: State-of-the-art*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, ch. 12, pp. 387–414.

[10] Z.-H. Zhou, *Ensemble methods: Foundations and algorithms* (Machine learning and pattern recognition), 1st. Boca Raton, FL, USA: CRC Press, 2012.

[11] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the 3rd International Conference on Learning Representations*, ser. ICLR 2015, Online: arXiv.org, 2015.

[12] G. Brockman *et al.*, "OpenAI Gym," *arXiv preprint arXiv:1606.01540*, vol. preprint, 2016.

[13] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16, New York, NY, USA: ACM, 2016, pp. 785–794.

[14] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel, "Benchmarking deep reinforcement learning for continuous control," in *Proceedings of the 33rd International Conference on Machine Learning*, ser. ICML 2016, vol. 48, Online: PMLR, 2016, pp. 1329–1338.

[15] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," in *Proceedings of the 4th International Conference on Learning Representations*, ser. ICLR 2016, Online: OpenReview.net, 2016.

[16] V. Mnih *et al.*, "Asynchronous methods for deep reinforcement learning," in *Proceedings of the 33rd International Conference on Machine Learning*, ser. ICML 2016, vol. 48, Online: PMLR, 2016, pp. 1928–1937.

[17] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.

[18] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," vol. preprint, 2017. arXiv: 1707.06347.

[19] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proceedings of the 35th International Conference on Machine Learning*, ser. ICML 2018, vol. 80, Online: PMLR, 2018, pp. 1582–1591.

[20] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft Actor-Critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proceedings of the 35th International Conference on Machine Learning*, ser. ICML 2018, vol. 80, Online: PMLR, 2018, pp. 1329–1338.

[21] O. Sagi and L. Rokach, "Ensemble learning: A survey," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 8, no. 4, e1249, 2018.

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

[23] K. Weinberger, *Lecture 12: Bias-variance tradeoff*, https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote12.html, Accessed: 2023-02-19, 2018.

[24] K. Cobbe, O. Klimov, C. Hesse, T. Kim, and J. Schulman, "Quantifying generalization in reinforcement learning," in *Proceedings of the 36th International Conference on Machine Learning*, ser. ICML 2019, vol. 97, Online: PMLR, 2019, pp. 1282–1289.

[25] A. R. Jordehi, "Optimisation of demand response in electric power systems, a review," *Renewable and Sustainable Energy Reviews*, vol. 103, pp. 308–319, 2019.

[26] M. A. Masood and F. Doshi-Velez, "Diversity-inducing policy gradient: Using maximum mean discrepancy to find a set of diverse policies," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence AI for Improving Human Well-being*, ser. IJCAI 2019, Online: International Joint Conferences on Artificial Intelligence Organization, 2019, pp. 5923–5929.

[27] L. Rokach, *Ensemble learning: Pattern classification using ensemble methods* (Machine perception and artificial intelligence), 2nd. Singapore: World Scientific Publishing Co. Pte. Ltd., 2019, vol. 85.

[28] J. R. Vázquez-Canteli, J. Kämpf, G. Henze, and Z. Nagy, "CityLearn v1.0: An OpenAI Gym environment for demand response with deep reinforcement learning," in *Proceedings of the 6th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, ser. BuildSys '19, New York, NY, USA: ACM, 2019, 356–357.

[29] J. R. Vázquez-Canteli and Z. Nagy, "Reinforcement learning for demand response: A review of algorithms and modeling techniques," *Applied Energy*, vol. 235, pp. 1072–1089, 2019.

[30] J. Parker-Holder, A. Pacchiano, K. M. Choromanski, and S. J. Roberts, "Effective diversity in population based reinforcement learning," in *Advances in Neural Information Processing Systems*, ser. NeurIPS 2020, vol. 33, Red Hook, NY, USA: Curran Associates, Inc., 2020, pp. 18 050–18 062.

[31] H. Yang, X.-Y. Liu, S. Zhong, and A. Walid, "Deep reinforcement learning for automated stock trading: An ensemble strategy," in *Proceedings of the First ACM International Conference on AI in Finance*, ser. ICAIF '20, New York, NY, USA: ACM, 2020.

[32] G. An, S. Moon, J.-H. Kim, and H. O. Song, "Uncertainty-based offline reinforcement learning with diversified Q-ensemble," in *Advances in neural information processing systems*, ser. NeurIPS 2021, vol. 34, Red Hook, NY, USA: Curran Associates, Inc., 2021, pp. 7436–7447.

[33] D. Ghosh, J. Rahme, A. Kumar, A. Zhang, R. P. Adams, and S. Levine, "Why generalization in RL is difficult: Epistemic POMDPs and implicit partial observability," in *Advances in Neural Information Processing Systems*, ser. NeurIPS 2021, vol. 34, Red Hook, NY, USA: Curran Associates, Inc., 2021, pp. 25 502–25 515.

[34] K. Lee, M. Laskin, A. Srinivas, and P. Abbeel, "SUNRISE: A simple unified framework for ensemble learning in deep reinforcement learning," in *Proceedings of the 38th International Conference on Machine Learning*, ser. ICML 2021, vol. 139, Online: PMLR, 2021, pp. 6131–6141.

[35] C. Li, T. Wang, C. Wu, Q. Zhao, J. Yang, and C. Zhang, "Celebrating diversity in shared multi-agent reinforcement learning," in *Advances in Neural Information Processing Systems*, ser. NeurIPS 2021, vol. 34, Red Hook, NY, USA: Curran Associates, Inc., 2021, pp. 3991–4002.

[36] Z. Liu, X. Li, B. Kang, and T. Darrell, "Regularization matters in policy optimization–an empirical study on continuous control," in *Proceedings of the 9th International Conference on Learning Representations*, ser. ICLR 2021, Online: OpenReview.net, 2021.

[37] M. Samsikova, *Effective diversity in population based reinforcement learning: DvD-TD3 Pytorch implementation*, https://github.com/holounic/DvD-TD3/, Accessed: 2023-01-30, 2021.

[38] S. Witty *et al.*, "Measuring and characterizing generalization in deep reinforcement learning," *Applied AI Letters*, vol. 2, no. 4, e45, 2021.

[39] M. Ganaie, M. Hu, A. Malik, M. Tanveer, and P. Suganthan, "Ensemble deep learning: A review," *Engineering Applications of Artificial Intelligence*, vol. 115, p. 105 151, 2022.

[40] I. E. Lab and AIcrowd, *NeurIPS 2022 CityLearn challenge: Using AI for building's energy management*, https://www.aicrowd.com/challenges/neurips-2022-citylearn-challenge, Accessed: 2023-01-30, 2022.

[41] K. R. McKee, J. Z. Leibo, C. Beattie, and R. Everett, "Quantifying the effects of environment and population diversity in multi-agent reinforcement learning," *Autonomous Agents and Multi-Agent Systems*, vol. 36, no. 21, 2022.

[42] K. Nweye, S. Sankaranarayanan, and Z. Nagy, "MERLIN: Multi-agent offline and transfer learning for occupant-centric energy flexible operation of grid-interactive communities using smart meter data and CityLearn," vol. preprint, 2022. arXiv: 2301.01148.

[43] T. Zhang, A. K. G.S., M. Afshari, P. Musilek, M. E. Taylor, and O. Ardakanian, "Diversity for transfer in learning-based control of buildings," in *Proceedings of the 13th ACM International Conference on Future Energy Systems*, ser. e-Energy '22, New York, NY, USA: ACM, 2022, pp. 556–564.

[44] R. Kirk, A. Zhang, E. Grefenstette, and T. Rocktäschel, "A survey of zero-shot generalisation in deep reinforcement learning," *Journal of Artificial Intelligence Research*, vol. 76, pp. 201–264, 2023.

[45] K. Nweye, S. Siva, and G. Z. Nagy, *The CityLearn challenge 2022*, version V1, 2023. DOI: 10.18738/T8/0YLJ6Q. [Online]. Available: https://doi.org/10.18738/T8/0YLJ6Q.

Appendix A: Implementation Details

The appendix provides details necessary for replicating our work. The codebase can be found at https://github.com/IRLL/diverse-ensemble-citylearn

A.1 DRL agent's observation space

Table A.1 lists raw observation features outputted by the CityLearn environment. The day of the month is not included by default, but it can be easily added. We do not use the day type feature (day of the week) because it is not correlated with any important variable in the environment except electricity pricing (which is included in observations). "Periodic transformation" in the comments means that we convert a feature into sine and cosine components, for example:

$$\text{Month}_{sin} = sin(2\pi\frac{\text{Month}}{12}), \quad \text{Month}_{cos} = cos(2\pi\frac{\text{Month}}{12})$$

We do so because there is a natural continuity between the 12th and 1st months, the last day of a month and the first day of the next month, the 24th hour of one day and the 1st hour of the next day. In each time-step, the agent takes in the information about the previous hour and makes a (dis)charging decision that takes effect in the upcoming hour. For example, when the 'hour' feature is 5, it means that the 5th hour of the day just passed and a (dis)charging action needs to be selected for the next, 6th, hour. The raw features (Table A.1) describing the previous hour take up 31 entries in the observation vector.

Table A.1: Raw observation features for DRL agent, and their value ranges

Feature	value range	comments
month	1, 2, ..., 12	periodic transformation
day of month	1, 2, ..., 31	periodic transformation
hour	1, 2, ..., 24	periodic transformation
outdoor temperature	[5.6; 32.2]	+ 6h, 12h, 24h predictions
outdoor relative humidity	[10.0; 100.0]	+ 6h, 12h, 24h predictions
diffuse solar irradiance	[0.0; 1017.0]	+ 6h, 12h, 24h predictions
direct solar irradiance	[0.0; 953.0]	+ 6h, 12h, 24h predictions
carbon intensity	[0.0704; 0.2818]	
non-shiftable load	[0.0; 8.85]	
solar generation	[0.0; 4.78]	upper limit depends on the nominal power of solar panels
battery state of charge	[0.0; 1.0]	
net electricity consumption	[-9.78; 13.85]	non-shiftable load - solar generation + battery charging
electricity pricing	[0.21; 0.54]	+ 6h, 12h, 24h predictions

Further, we add extra features that capture relevant differences between households. To that end, we keep electricity consumption and solar generation histories for each house, separately for each hour of the day. Using the histories, we calculate the average net electricity consumption observed in the past 14 days during the upcoming hour of the day. We do the same with solar generation. This gives us 2 features that represent power consumption and generation of a household in the past 2 weeks, but only for the upcoming hour of the day. We repeat the same calculation for three more hours: the 2nd upcoming hour, the 6th upcoming hour, and the 12th upcoming hour. For example, if the previous hour ('hour' feature in the observation) is 15, then we compute past-14-day averages for the following hours: 16, 17, 21, and 3. Together, we obtain 8 values that describe the household's power consumption and generation patterns. Lastly, we include 7 features from the previous time-step: carbon intensity, electricity consumption, solar generation, and electricity pricing (as well as its +6, +12, and +24 hour predictions).

These additional variables improve the training performance of our DRL agent, compared to having only raw features. Overall, the total size of the observation vector is 46. We are confident that better observation features can be designed, but it is not the focus of our work.

A.2 Hyperparameters

Table A.2 lists the hyperparameters we used for training SAC policies. For each parameter, we performed a grid search while keeping others fixed. Comparisons were based on training costs achieved when training on buildings 1 through 5 [45].

The auxiliary diversity term DvD requires two hyperparameters: diversity importance coefficient λ (Equation 2.5) and the number of observations used for embedding each policy [30]. The latter parameter is fixed at 20, as in a reference implementation [37]. The λ coefficient was grid-searched from the set $\{0.2, 0.4, 0.6, 0.8\}$. The best value was 0.4 for the Simple-averaging ensemble, 0.2 for the Min-σ ensemble, and

Table A.2: SAC hyperparameters

Parameter	Value	Search space
optimizer	Adam [11]	—
learning rate	3×10^{-4}	$\{0.3, 1, 3, 10, 30\} \times 10^{-4}$
discount factor (γ)	0.986	{0.98, 0.986, 0.99}
replay buffer size	2^{18}	$\{2^{17}, 2^{18}, 2^{19}, 2^{20}\}$
number of hidden layers	3	{2, 3, 4}
size of each hidden layer	512	{64, 128, 256, 512, 1024}
number of samples per minibatch	256	{64, 128, 256, 512}
entropy coefficient	0.2	{0.0, 0.1, 0.2, 0.3, 0.4, 0.5}
nonlinearity (actor)	ReLU	{ReLU, Tanh}
nonlinearity (critic)	ReLU, Batchnorm	{ReLU, Tanh}, {Batchnorm, no Batchnorm}
target smoothing coefficient (τ)	0.005	—
target update interval	1	—
gradient steps	1	—
action scaling coefficient (c_a)	0.5	{0.25, 0.5, 0.75, 1.0}
reward scaling coefficient (c_r)	10	{1, 5, 10, 20}

there was no significant difference between the values for the σ-weighted ensemble (we used 0.6).

A.3 Hand-Crafted Policy Design

The hand-crafted (HC) controller used as one of the baselines in the first experiment (Section 5.1) consists of 2 modules: a predictor of net demand in the next hour and a decision tree which relies on the predictions to make a charging/discharging decision.

Net demand, ignoring the battery usage, is equal to non-shiftable load minus solar generation. We used the XGBoost algorithm [13] to predict the non-shiftable load of a household, based on the following input features:

- periodically normalized (sin-cos transformed) hour of the day
- past 14 days average of non-shiftable load for the next hour (e.g., if the next hour is 5, we take the average of 14 values corresponding to the non-shiftable load at hour 5 that was observed in the previous 14 days)
- 1 day history of non-shiftable load (i.e., 24 most recent values of non-shiftable load)

As for predicting solar generation, we trained a linear regression model with the following features:

- solar generation in the past 2 hours
- average solar generation in the district in the past 2 hours

Both models are trained only on the training data available in each cross-validation fold. For example, when DRL policies are trained on the first 5 months of data of the first 5 buildings, the predictive models are also trained on that same data.

Figure A.1 shows the two-level decision structure of the HC controller. We let δ be the difference between non-shiftable load and solar generation values predicted

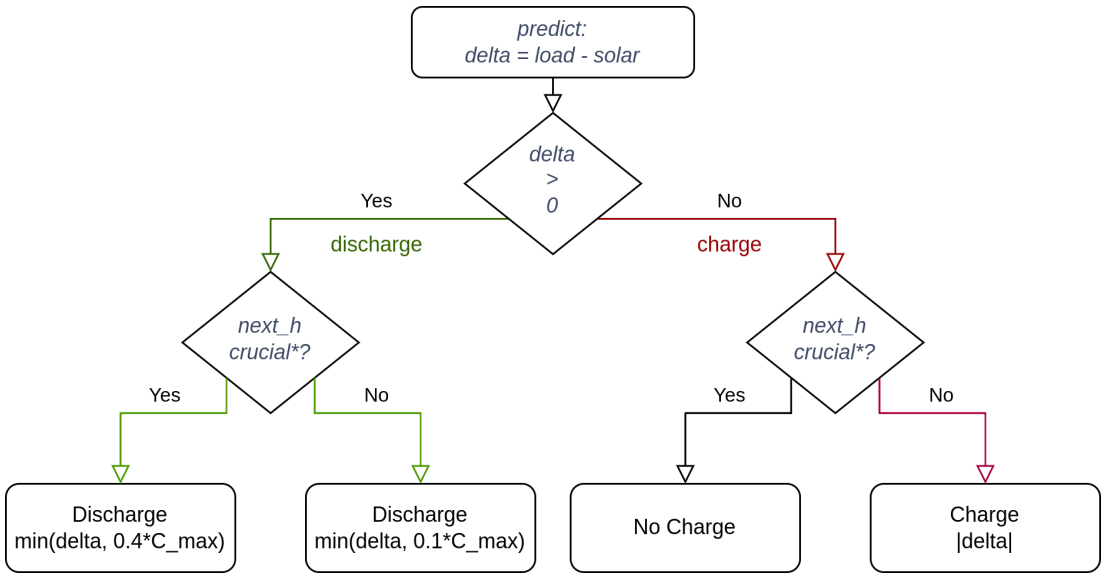


Figure A.1: Hand-crafted policy's decision tree

for the next hour. If $\delta > 0$, demand is predicted to be higher than generation, then HC attempts to charge by the amount δ. Otherwise, HC tries to discharge by $|\delta|$. However, in both cases, there is a limit to (dis)charging. Since the electricity pricing is much higher during hours 16 through 20 (15:00 – 19:00 or 3 pm – 7 pm), we call them 'crucial' hours. Up to 40% of the battery capacity is allowed to be discharged during crucial hours, but only 10% during other hours. Charging is not limited during non-crucial hours other than by $|\delta|$ and the battery's remaining capacity. On the other hand, when the next hour is crucial, even if surplus generation is predicted — charging is not allowed because predictions are imperfect, and charging by mistake is very costly during those hours. The limiting parameters were tuned based on the algorithm's performance on buildings 1 through 5.