Variance-Seeking Meta-Exploration to Handle Out-of-Distribution Tasks

Anonymous authors

Abstract

Meta-Reinforcement Learning (meta-RL) yields the potential to improve the sample efficiency of reinforcement learning algorithms. Through training an agent on multiple meta-RL tasks, the agent is able to learn a policy based on past experience, and leverage this to solve new, unseen tasks. Accordingly, meta-RL promises to solve real-world problems, such as real-time heating, ventilation and air-conditioning (HVAC) control without accurate simulators of the target building. In this paper, we propose a meta-RL method which trains an agent on first order models to efficiently learn and adapt to the internal dynamics of a real-world building. We recognise that meta-agents trained on first order simulator models do not perform well on second order models, owing to the meta-RL assumption that the test tasks should be from within the same distribution as the training tasks. In response, we propose a novel exploration method called variance seeking meta-exploration which enables a meta-RL agent to perform well on complex tasks outside of its training distribution. Our method programs the agent to prefer exploring task dependent state-action pairs, and in turn, allows it to adapt efficiently to challenging second order models which bear greater semblance to real-world problems.

1 Introduction

Reinforcement learning (RL) algorithms aim to learn a policy that maximises the reward in one specific environment. The coalescence of deep learning with RL has significantly improved RL's ability to solve diverse and challenging problems [21, 14, 2]. However, in the learning phase, an RL agent still requires extensive exploratory interactions to learn a good policy. These exploratory actions can be dangerous and can have high associated costs in the real-world, which means it is currently impractical to train RL agents directly in a real-world environment.

Meta-RL methods aim to improve the sample efficiency of RL by training with an objective to quickly perform well on many related environments [24, 6, 19]. During the training phase, the agent is trained on a number of meta-training tasks sampled from a training task distribution. At meta-test time, the agent's learning performance is measured on an unseen task. Critically, meta-learning methods assume that the training and testing tasks are always sampled from the *same* distribution. Therefore, meta-RL may fail to generalise to tasks that lie far outside the training distribution [12, 13]. As the real-world lies outside the training distribution by construction, this assumption means Meta-RL agents may fail to adapt to real-world problems.

We show that this problem appears in practice, when we try to apply Meta-RL to price-adaptive control of heating, ventilation and cooling (HVAC) systems. Agents trained on simple first-order temperature dynamics fail to generalise to more complex and out-of-distribution (OOD) second-order simulation models. We hypothesize that one of the reasons that meta-RL fails in OOD tasks is because the exploration strategies learnt by the agent overfit on the training task distribution. In response to this problem, we propose an exploration method called *variance-seeking meta-exploration*. This method learns to identify states and actions with high variance in their returns across tasks, and then

Preprint. Under review.

focuses on exploring these states to reduce uncertainty when faced with a new task. We demonstrate empirically that this exploration method improves performance of Meta-RL on OOD tasks.

2 Preliminaries

We define a probability distribution $p(M)$ over a range of possible tasks modeled as Markov Decision Processes [18, MDPs] $M_i = \langle S, A, R, T_i \rangle$. Each MDP within this distribution has the same state space $s \in S$ and action set $a \in A$, as well as the same reward function $R(s_t, a_t) \to \mathbb{R}$. The state transition function $T_i(s_t, a_t, s_{t+1}) \to \Pr(s_{t+1} \mid s_t, a_t)$ varies across MDPs according to a latent variable ψ identifying the task, however, ψ_i the MDP identifier for M_i is kept hidden from the agent at all times. We use the terms MDP and task interchangeably throughout the paper.

In standard RL, an MDP-specific policy $\pi_{\psi_i}(s_t) \to a_t$ is learned which maximises the discounted reward objective for a given discount factor γ and episode length H,

$$\max_{\pi_{\psi_i}} \mathbb{E}\left[\sum_{t=0}^{H} \gamma^t R(s_t, a_t) \,\middle|\, s_{t+1} \sim T_i(s_t, a_t), a_t = \pi_{\psi_i}(s_t)\right] \text{ for fixed } M_i \sim p(M). \tag{1}$$

Policy π_{ψ_i} induces $Q_{\pi_{\psi_i}}(s_t, a_t)$, the expected discounted sum of rewards an agent receives in state s_t by taking action a_t and then following policy $\pi_{\psi_i} = \arg\max_a Q_{\pi_{\psi_i}}(s_{t+1}, a)$. One way to learn an optimal control policy is Q-learning [26], which incrementally updates the current value estimate Q_ω with step size α to minimise the magnitude of the 1-step temporal difference (TD) error δ [23],

$$\begin{aligned} \delta_\omega(s_t, a_t) &= \Big(r_t + \gamma \max_a Q_\omega(s_{t+1}, a)\Big) - Q_\omega(s_t, a_t), \\ Q'_\omega(s_t, a_t) &= Q_\omega(s_t, a_t) + \alpha \delta_\omega(s_t, a_t). \end{aligned} \tag{2}$$

In practice, state space S is often too large to track values of Q_ω exactly. Deep Q-Networks [14, DQN] solves this problem by using a deep neural network to approximate values of Q_ω, minimising the magnitude of the TD error in eq. (2) using stochastic gradient descent.

2.1 Memory-based Meta-RL

In meta-RL the objective is to learn a global policy π which is able to maximise the reward for *all* MDPs sampled from $p(M)$, under the same objective (1) for each sampled M_i,

$$\max_{\pi} \mathbb{E}\left[\mathbb{E}\Big[\sum_{t=0}^{H} \gamma^t R(s_t, a_t) \;\Big|\; s_{t+1} \sim T_i(s_t, a_t), a_t = \pi(s_t)\Big] \,\middle|\, M_i \sim p(M)\right]. \tag{3}$$

Maximising (3) forces the agent to learn a policy π which can adapt within H time steps to maximise the reward. In doing so, the agent must learn the underlying task structure for MDPs in $p(M)$, to efficiently handle new tasks sampled from $p(M)$. In other words, the agent learns how to learn.

Memory based meta-RL [24, 3] uses a recurrent neural network (RNN) based policy to maximise eq. (3) The reward and action at the previous step (r_{t-1}) and (a_{t-1}) respectively are appended to the state space of the agent. The RNN policy selects action a_t, conditioned on the hidden state h_t of the RNN, in addition to the state s_t. Therefore, the policy becomes $\pi(a_t \mid h_t, s_t)$. Intuitively, the recurrent update of the the hidden state can be understood as an approximation of the hidden MDP identifier ψ_i. Instead of explicitly inferring ψ_i, the agent learns to infer an internal representation of ψ_i as h_t, given agent's experience up till $t-1$ in the current MDP. In addition to identifying the MDP, the agent also learns a policy $\pi(a_t \mid h_t, s_t)$ that acts optimally according to it's belief (h_t) about the current MDP at time t. Both the MDP inference procedure and the policy are learnt simultaneously by optimising the RNN during meta-training. We use the terms *RL*^{2} and memory based meta-RL interchangeably in this paper.

2.2 Meta-RL for HVAC control

Price-adaptive control of heating, ventilation and cooling (HVAC) systems aims to reduce the energy consumption of HVAC systems during periods of peak demand. By doing so, the reliance on non-renewable sources, which is maximum during peak periods, can be reduced. We cast this HVAC

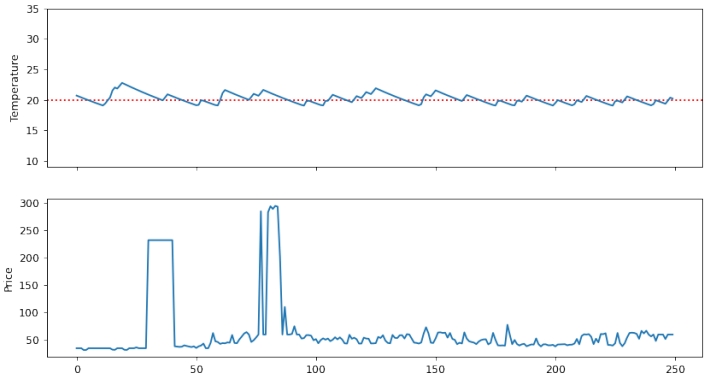


Figure 1: *HVAC Control Problem* An agent is tasked with optimally balancing thermal comfort with energy costs. Comfort is the temperature proximity to the setpoint (top) and costs are electricity price (bottom). By setting appropriate temperature set-points, the agent learns to avoid both price spikes by pre-heating while maintaining temperature close to the target comfort level.

control problem as an RL problem, where the agent controls a central cooling and heating system of different sized buildings via thermostat setpoints (Figure 1). As part of this task, the agent must pay for its energy usage according to real-time energy prices. Because of the high costs of sub-optimal actions, training an RL agent directly in the real world is infeasible. One solution to this problem is to train the agent on a carefully designed accurate simulator of the target building so as to avoid real-world exploration [1, 27, 10]. This, however, is highly time-consuming and expensive, making it infeasible to deploy on a wider scale. In response, we propose a meta-RL solution which can efficiently and cost-effectively train an agent to adapt to the dynamics of any real-world building. More specifically, we use meta-RL to train an agent on a number of easy to develop first order dynamics models [15]. As a result, the agent learns to adapt to the new and unseen dynamics of a building. In doing so, our proposed method dispenses with the need to construct accurate simulators of the target building. We test the agent's ability to adapt to real world buildings by measuring its performance on much more realistic and expensive second order dynamics models [7].

In this case, the distribution of all possible first order model corresponds to the meta-training task distribution $p(M)$. The latent parameter ψ that generates MDPs in $p(M)$ corresponds to the parameters: time to cool and time to heat of the first order model. At test time, the test MDP is not sampled from $p(M)$ and is instead sampled from an unseen distribution of more complex second order models of real world buildings. This allows us to test the agent's ability to adapt to out-of-distribution tasks.

The state-space of the agent consists of: the room temperature, the outside temperature, the current energy price, the energy price forecast for next five time periods, and the action and reward for previous step (for meta-RL). At each step, the agent decides the target temperature setpoint for the heating system. So the discrete action-space consists of natural numbers between 0 and 40. For the price data, we use publicly available 15-minute dispatch prices from the Australian electricity market operator [16].

3 Exploration using task-dependent state-action pairs

To address the transfer problem identified in the previous section, we propose a novel exploration objective term to bias exploration. The key idea behind our exploration objective is that to identify the MDP; the agent should focus on exploring those state-action pairs whose expected reward depends upon MDP identifier ψ_i First, we motivate our objective using a simple grid navigation task. Then, we show how to use our exploration objective to improve meta-RL exploration.

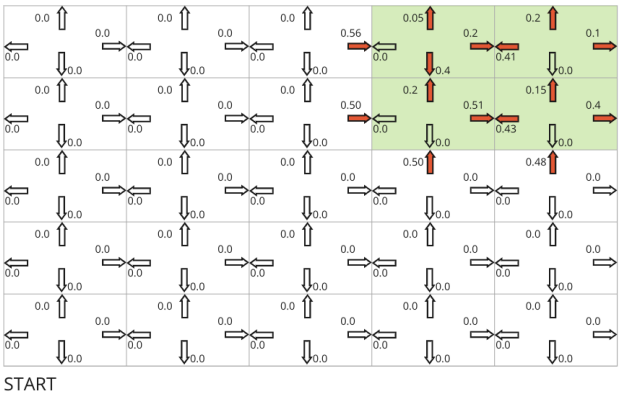


Figure 2: *Grid navigation problem* The agent is tasked with finding the goal location in the top right corner within 15 steps. The agent gets a reward of $+1$ in the goal location and a reward of -0.2 on all other cells. The agent can stay stationary, or move in one of the 4 cardinal directions. The variance of TD-error across tasks is shown next to the move action in each state, with high variance actions coloured red. Only state-action pairs close to the possible goal locations have high TD-error variance.

3.1 Motivation behind exploration objective

In theory, Q-learning learns a policy for a single MDP. However, in this case, we train the agent using data from many randomly sampled MDPs from $p(M)$. Because for the same state and action, $Q(s_t, a_t)$ can have different value across MDPs in $p(M)$, the agent cannot learn a fixed $Q(s_t, a_t)$ that holds true for all MDPs. However, we show via an example that after sufficient training, the variance of TD error differs across the state-action pairs.

We use a simple grid navigation task to demonstrate how the variance of TD error across MDPs differs in state-action pairs. In this task, the agent's goal is to navigate to a goal in a 5×5 grid within 15-time steps. However, the goal is randomly assigned to one of the four possible goal locations in the top right corner. This task is a modification of the navigation task proposed by [28]. $p(M)$ in this example includes only four MDPs where each possible goal location gives a new MDP. The MDP identifier ψ_i here represents the goal location.

We train tabular Q-learning on MDPs sampled according to $p(M)$. ψ_i remains hidden for the agent at all times. Upon convergence, we measure the variance of TD error across all tasks. The results are shown in figure 2, TD-error variance of the state-action pairs near the possible goal locations is highest. This experiment demonstrates that the variance of TD error differs across state-action pairs.

As demonstrated above in our grid navigation example, the TD error varies only for certain state-action pairs. Because the $Q(s_t, a_t)$ is learnt using the TD error in Eq (3), the $Q(s_t, a_t)$ of only a few state-action pairs will be updated when in a new MDP. This means that the $Q(s_t, a_t)$ of most state-action pairs will stay constant across MDPs. We call the state-action pairs whose $Q(s_t, a_t)$ are independent of MDP identifier ψ_i, *task invariant* state-action pairs. The state-action pairs whose $Q(s_t, a_t)$ do depend on ψ_i and therefore have a high variance of TD errors across tasks are called *task dependent* state-action pairs.

During the exploration steps, when a meta-RL agent infers the underlying MDP, the task dependent state-action pairs can provide useful data. On the other hand, task invariant state-action pairs tell the agent little about the underlying MDP. For example, in the grid navigation task, the high TD error variance state-actions are close to the possible goal location. The agent must visit these high variance state-action pairs near the possible goal locations to infer the underlying MDP. On the other hand, any state-actions far off from the goal locations cannot be used to infer the MDP. This shows that the high TD error variance of a state-action pair is a good signal for MDP inference.

Duan et al. [3] show that memory based meta-RL correctly learns which states to explore for MDP inference in a maze navigation problem similar to our grid example. However, in real-world tasks such as the HVAC problem, the state-space can be large and continuous, meaning it is impossible to say which state-action pairs the meta-RL agent will learn to use for MDP inference. Because meta-RL agents learn to maximise the discounted reward over MDPs in $p(M)$, they learn to use the

state-action pairs that allow MDP inference only for the tasks **within** $p(M)$. The state-actions that the agent learns to use for MDP inference may not necessarily be task dependent. Moreover, as we show in our experiments, the learnt state-actions struggle to help in MDP inference in OOD tasks.

To avoid learning state-actions that only allow MDP inference for tasks within $p(M)$, we propose biasing the agent's inference steps towards the task-dependent state-action pairs. In doing so, we hope that the agent will use task-dependent state-action pairs for MDP inference instead of learning the state-actions that only allow MDP inference for within distribution tasks.

The simple grid navigation example above has only a small number of state-action pairs, and the variance of each state-action pair can be explicitly calculated and stored in a table. However, in most problems like our HVAC problem, the state space is continuous, and the variance of each (s_t, a_t) just like the $Q(s_t, a_t)$ needs to be approximated. To approximate the TD error variance for continuous states, we use a separate a function approximator f_θ, which learns to approximate $f_\theta(s_t, a_t) \approx \delta(s_t, a_t)$, where $\delta(s_t, a_t)$ is the TD error of a fixed DQN for state-action pair (s_t, a_t). $f_\theta(s_t, a_t)$ minimises the squared error of its predictions. The objective it minimises is

$$\mathcal{L}(\theta) = \mathbb{E}_{p(m)}[\delta(s_t, a_t) - f_\theta(s_t, a_t)]^2 \tag{4}$$

If the the variance of TD error across MDPs is high for (s_t, a_t), approximator $f_\theta(s_t, a_t)$ will struggle to learn a stable mapping and therefore have a high error for (s_t, a_t). On the other hand, if TD error variance is low for (s_u, a_u), $f_\theta(s_u, a_u)$ can learn to predict the fixed TD error correctly (given enough data) and therefore minimise its error. We use the mean squared error of $f_\theta(s_t, a_t)$ to approximate the TD error variance for continuous states.

Algorithm 1 Variance-seeking Meta-exploration

Input: Training distribution of MDPs $p(m)$
Initialise: DQN $d_\omega(s_t, a_t)$, Two replay buffers β_d and β_f, TD error approximating network $f_\theta(s_t, a_t)$, meta-RL policy $\pi_\phi(a_t|h_t, s_t)$

while *not done* **do** ▷ Training the DQN
  Sample a random MDP M_i from $p(M)$
  Rollout the DQN policy $d_\omega(s_t, a_t)$ in M_i
  Store the resulting trajectory in replay buffer β_d
  Sample a random trajectory τ_i from β_d
  Update DQN weights ω using τ_i
end while
return ω ▷ The DQN weights after training
while *not done* **do** ▷ Training the meta-RL policy
  Sample a random MDP M_i from $p(M)$
  for each step t in $0, 1, ..., H-1$ **do**
    Select action: $a_t = \pi_\phi(a_t|h_t, s_t)$
    Advance MDP: $s_{t+1}, r_{t+1} = M_i(a_t)$
    Compute TD error of DQN: $\delta_\omega(s_t, a_t)$ using Eq (4)
    Compute the variance of TD error: r^{error} using Eq (8)
    if $t <= n$ **then**:
      Augment reward: $\tilde{r}_{t+1} = \kappa r_{t+1} + \nu r^{error}$ using Eq (9)
    else
      Set reward to environment reward $\tilde{r}_{t+1} = r_{t+1}$
    end if
    Store (s_t,a_t, $\delta_\omega(s_t, a_t)$) in β_f
  end for
  Update meta-RL policy weights ϕ to maximise $\tilde{r}$ using Eq (1)
  Update $f_\theta(s_t, a_t)$ weights θ using entire β_f by minimising $\mathcal{L}(\theta)$ in Eq (7)
end while
return ϕ ▷ The meta-RL policy weights after training

3.2 Learning to explore via task-dependent state-action pairs

Following the work of [22], we assume that memory based meta-RL or RL^2 performs MDP inference in its first n steps in a MDP. Therefore, we formulate a training objective that encourages the agent to collect data from task dependent state-action pairs in it's initial interactions with the MDP. The individual components involved are summarised as follows:

- A DQN $d_\omega(s_t, a_t)$, parameterised by ω;
- A function approximator $f_\theta(s_t, a_t)$, predicting the TD error of $d_\omega(s_t, a_t)$. f_θ is implemented as a deep neural network; and
- A memory-based meta-RL policy $\pi_\phi(a_t \mid h_t, s_t)$.

To encourage the agent to visit task dependent state-action pairs in its first n steps, we transform the agent's reward during first n steps. We reward the agent for visiting those state-actions where the squared error of $f_\theta(s_t, a_t)$ is maximum, as shown below

$$\begin{aligned} r^{error}(s_t, a_t) &= (f_\theta(s_t, a_t) - \delta(s_t, a_t))^2 \\ \widetilde{R}(s_t, a_t) &= \kappa R(s_t, a_t) + \nu r^{error}(s_t, a_t) \end{aligned} \tag{5}$$

where R is the environment reward and κ and ν are tunable hyperparameters.

To incentivize exploration in task dependent state-action pairs, the agent is trained to maximise the augmented reward $\widetilde{R}$ in the first n steps. The agent then maximises environment reward R for the rest of the episode. Next we describe the steps involved in training the agent.

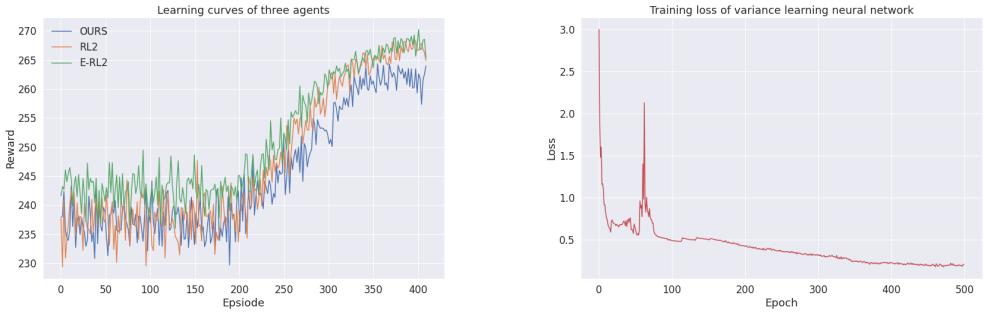
Algorithm 1 shows the complete pseudo-code for training. Firstly, the DQN $d_\omega(s_t, a_t)$ is trained using data from MDPs sampled from $p(M)$. Once the training loss of the DQN stabilises and stops decreasing, the weights of the DQN are frozen for all the next steps. The purpose of DQN is only to identify the high TD error variance states.

In the next phase, we train the function approximator $f_\theta(s_t, a_t)$ simultaneously with the meta-RL policy $\pi_\phi(a_t \mid h_t, s_t)$. During early training steps, the error of $f_\theta(s_t, a_t)$ is high because of small training data. Because the policy $\pi_\phi(a_t \mid h_t, s_t)$ gets rewarded for visiting the states where the error of $f_\theta(s_t, a_t)$ is maximum, it will maximise the data collected from these state-actions. As more data becomes available for training $f_\theta(s_t, a_t)$, it's error begins to reduce. The error continues to decrease until only the high TD-error variance state-actions are left to be learnt. Because the learning target for $f_\theta(s_t, a_t)$ has a high variance, the error continues to stay high even when $\pi_\phi(a_t \mid h_t, s_t)$ focuses on collecting data only from these state-action pairs. Towards the end of training we expect an equilibrium, where $f_\theta(s_t, a_t)$ cannot reduce it's error further because of high variance in it's target values (figure 3b). The meta-RL policy on the other hand cannot get a higher reward because the error of $f_\theta(s_t, a_t)$ is lower for all state-action pairs except those with a high TD error variance. This concurrent training regime pits $f_\theta(s_t, a_t)$ and $\pi_\phi(a_t \mid h_t, s_t)$ against each other. This is similar to a minimax two player game.

4 Experiments

To evaluate our method, we compare our exploration strategy against state of the art Meta-RL algorithms on its ability to generalize to out-of-distribution tasks in the HVAC domain.

Algorithms We compare our method against the two memory based meta-RL methods in literature. The first method is the original memory-based meta-RL method proposed by [24, 3]. By maximising the RL reward in Eq (1), this method learns a Bayes optimal trade off for exploration and exploitation [17]. If we do not augment the environment reward in the first n steps, our method reduces to this original method. The second method we compare against is *E-RL*^{2}[22]. This method has shown strong improvements over RL^2 for tasks requiring complex exploration. By setting the environment reward in the first n steps to 0 during the gradient update, the gradient computation of the RL objective in Eq (1) *explicitly* assigns credit for the first n exploratory steps. Our method becomes equivalent to *E-RL*^{2} if we augment the reward to 0 for the first n steps. Similar to *E-RL*^{2}, our method continues to receive the original environment reward at each step including the the first n steps of the policy roll



(a) Learning curves of three agents

(b) Training loss variance learning network

Figure 3: **(a)** The learning curves for all agents. The reward for each episode was averaged across the 30 random seeds. **(b)** The training loss of variance learning network. To show the loss trajectory more clearly, the first two epochs, where the loss was very high were omitted. To show the convergence towards the end, we ran the training for 100 extra epochs towards the end.

out. We also compare the three agents against an Oracle, which is a RL agent trained specifically on the test MDP until convergence. The oracle was trained using the IMPALA algorithm [4] because of its strong demonstrated performance in challenging tasks using a single machine.

We closely followed the training method and hyperparameters of [25] and trained all our agents using IMPALA. Similar to [24], all our agents used LSTM [8] instead of a RNN.Importantly, in our method, because we want to emphasise on the error signal from $f_\theta(s_t, a_t)$ we set κ in in Eq (7) to 0.05 (close to 0). While ν in set to 0.55, which we found to be the good in terms of final reward.

Tasks To test the ability to generalise to out of distribution tasks, we compare the performance of the three methods on the HVAC problem. The three methods were trained on the MDPs sampled from a distribution of 1st order models. The first order models in the training distribution were generated by randomly varying the time to heat (α_{heat}) and time to cool (α_{cool}) parameter of the model. At the start of each episode, a new MDP was randomly sampled from this 1st order training distribution. All three methods converged around the same number of steps. To allow a fair comparison between methods, we trained all agents for exactly 410 episodes. We chose 410 episodes, because beyond this, the three agents showed no improvement in average reward. The number of exploration steps in each episode was set to n = 40, which is around 3.5 hours of exploration in the real-world. All hyper-parameters were fixed across the three agents.

Evaluation Currently the meta-RL literature lacks a consistent evaluation procedure [5]. A number of studies report the meta-training performance [19, 5, 22]. However, to report the performance on out-of distribution tasks we followed the evaluation procedure of [25]. At test time, the weights of all agents were fixed and the reward after n steps in each MDP was recorded. This was necessary because both *E-RL*^{2} and our method are trained to maximise the reward after n steps. We report the mean and median rewards of all agents across 30 random seeds in Table 1, where each random seed was evaluated for 100 episodes in a fixed MDP. Both the 1st order and the 2nd order evaluation tasks were held fixed for all random seeds.

4.1 Results and discussion

On the first order (within distribution) MDPs, *E-RL*^{2} achieves the highest reward, slightly higher than RL^2 (Table 1). We hypothesise that the strong performance of *E-RL*^{2} on within distribution tasks can be attributed to a better exploration strategy learnt in comparison to RL^2. Our method performs worse than both *E-RL*^{2} and RL^2. This is expected because our method unlike RL^2 does not learn a Bayes optimal exploration strategy. Instead our augmented reward biases the agent to visit high TD error variance states, which may not be the best for MDP inference within this specific task distribution. On the other hand RL^2 and *E-RL*^{2} optimise only the RL reward. Therefore they are capable of learning state-actions that allow the best MDP inference for the specific task distribution.

Table 1: Performance on 1st order (training distribution) and 2nd order (outside of training distribution) MDPs. The mean and median are calculated using 30 random seeds for each agent.

	First-order model		Second-order model	
Agent	mean	median	mean	median
RL^2	266.7	**270.75**	246.70	252.33
E-RL^2	**268.2**	269.83	239.76	249.94
Ours	263.87	268.36	**253.46**	**255.14**
Oracle	271.01	271.64	255.82	256.24

However, despite showing strong results on within distribution tasks, both RL^2 and *E-RL*^{2} fail to replicate their successes in OOD second order tasks. We hypothesise that this is because the state-actions that they rely upon to make MDP inference are learnt specifically for the training task distribution. Therefore, when faced with an OOD task, these learnt state-actions may fail to aid the MDP inference process. On the other hand our exploration method does not directly learn the state-action pairs for MDP inference. Instead, our augmented reward biases the agent to use high TD error variance state-actions to make MDP inference, allowing for better MDP inference (close to the Oracle) in the OOD tasks than the state-actions learnt directly by meta-RL.

5 Related work

Hochreiter et al. [9] use RNN's for meta-learning in supervised learning problems. Building on the work of [9], [24, 3] use RNNs for meta-learning in reinforcement learning problems. Ortega et al. [17] cast memory based meta-RL in a Bayesian framework and show that the meta-RL strategies learnt are approximately Bayes optimal. To maintain an explicit belief over the possible MDPs, Varibad [28] learns a variational auto-encoder (VAE) to infer the MDP. The policy is conditioned on the VAEs output and learns an approximate Bayes optimal behaviour. Training the VAE makes this method significantly slower in training than memory-based methods [19], while the performance at test time is similar. Therefore we used memory-based meta-RL in our work for faster training time. Moreover, in comparison to other meta-RL methods such as MAML [6] and PEARL [19], we chose memory based meta-RL for this work because of the low computational requirements when deployed in the real-world.

A number of studies have applied RL to control HVAC systems. Ruelens et al. [20] use offline-RL on the predicted energy prices and demand for residential buildings. However, the policy learnt offline could be sub-optimal due to any errors in the forecasting model. This is in contrast to a meta-RL solution where the agent learns to adapt its policy to deal with any errors in the forecast. To minimise expensive exploration in the real-world, [10] first learn a simple linear dynamics model of a data centre from the data collected during exploration, and then use this learnt model to train the agent. This model based RL approach can be quite expensive in particular if the underlying dynamics model is non-linear. In comparison our model-free meta-RL approach makes no assumptions about the underlying dynamics model of the target building. Other studies [1, 27, 11] use RL for HVAC control in accurate simulation models of the target building. However, developing accurate simulators is expensive and time-consuming. Our method instead uses simple and easy to develop first order models to learn RL policies that can generalise to more complex environments including the real-world.

6 Conclusions and future work

In this paper, we propose training a meta-RL agent on cheaper first order models to overcome the significant problems associated with developing expensive simulators of target buildings for HVAC control. Our experiments show, however, that meta-RL fails to deliver strong performances in more complex second order models. This is problematic because the real-world is much more complex than the first order models. To allow meta-RL to accomplish more complex tasks than those within its training distribution, we propose a novel exploration method called variance seeking meta-exploration. This method biases the agent to explore task dependent state-action pairs. We show in our experiments that the agent trained using our method can generalise to second order tasks much better as compared

to existing methods. This shows the benefit of adding an inductive bias towards certain state-action pairs during the exploration phase for OOD tasks. Our method also sheds light on the importance of TD error variance for meta-RL exploration. Further research into the properties of states with high TD error variance across environments, and the resulting behaviour of exploring in such states, is necessary to provide a sound theoretical underpinning to our method. In addition to this, we are also working to test our meta-RL solution to control large real-world buildings to minimise their carbon footprint.

References

[1] Donald Azuatalam, Wee-Lih Lee, Frits de Nijs, and Ariel Liebman. Reinforcement learning for whole-building hvac control and demand response. *Energy and AI*, 2:100020, 2020. ISSN 2666-5468. doi: https://doi.org/10.1016/j.egyai.2020.100020. URL `https://www.sciencedirect.com/science/article/pii/S2666546820300203`.

[2] Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1329–1338, New York, New York, USA, 20–22 Jun 2016. PMLR. URL `http://proceedings.mlr.press/v48/duan16.html`.

[3] Yan Duan, John Schulman, Xi Chen, Peter L. Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl^2: Fast reinforcement learning via slow reinforcement learning, 2016. URL `https://arxiv.org/abs/2008.02790`.

[4] Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Vlad Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. IMPALA: Scalable distributed deep-RL with importance weighted actor-learner architectures. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1407–1416. PMLR, 10–15 Jul 2018. URL `http://proceedings.mlr.press/v80/espeholt18a.html`.

[5] Rasool Fakoor, Pratik Chaudhari, Stefano Soatto, and Alexander J. Smola. Meta-q-learning. In *International Conference on Learning Representations*, 2020.

[6] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135. PMLR, 06–11 Aug 2017. URL `http://proceedings.mlr.press/v70/finn17a.html`.

[7] R.T. Guttromson, D.P. Chassin, and S.E. Widergren. Residential energy resource models for distribution feeder simulation. In *2003 IEEE Power Engineering Society General Meeting (IEEE Cat. No.03CH37491)*, volume 1, pages 108–113 Vol. 1, July 2003. doi: 10.1109/PES.2003.1267145.

[8] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9 (8):1735–1780, 1997.

[9] Sepp Hochreiter, A. Steven Younger, and Peter R. Conwell. Learning to learn using gradient descent. In Georg Dorffner, Horst Bischof, and Kurt Hornik, editors, *Artificial Neural Networks — ICANN 2001*, pages 87–94, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg. ISBN 978-3-540-44668-2.

[10] Nevena Lazic, Craig Boutilier, Tyler Lu, Eehern Wong, Binz Roy, MK Ryu, and Greg Imwalle. Data center cooling using model-predictive control. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL `https://proceedings.neurips.cc/paper/2018/file/059fdcd96baeb75112f09fa1dcc740cc-Paper.pdf`.

[11] Hepeng Li, Zhiqiang Wan, and Haibo He. Real-time residential demand response. *IEEE Transactions on Smart Grid*, 11(5):4144–4154, 2020. doi: 10.1109/TSG.2020.2978061.

[12] Bhairav Mehta, Tristan Deleu, Sharath Chandra Raparthy, Chris J. Pal, and Liam Paull. Curriculum in gradient-based meta-reinforcement learning, 2020.

[13] Russell Mendonca, Xinyang Geng, Chelsea Finn, and Sergey Levine. Meta-reinforcement learning robust to distributional shift via model identification and experience relabeling, 2020.

[14] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015. ISSN 1476-4687. doi: 10.1038/nature14236. URL `https://doi.org/10.1038/nature14236`.

[15] R.E. Mortensen and K.P. Haggerty. A stochastic computer model for heating and cooling loads. *IEEE Transactions on Power Systems*, 3(3):1213–1219, Aug 1988. ISSN 1558-0679. doi: 10.1109/59.14584.

[16] Australian Energy Market Operator. National electricity market data dashboard. `https://aemo.com.au/en/energy-systems/electricity/national-electricity-market-nem/data-nem/data-dashboard-nem`. Accessed: 2021-09-29.

[17] Pedro A. Ortega, Jane X. Wang, Mark Rowland, Tim Genewein, Zeb Kurth-Nelson, Razvan Pascanu, Nicolas Heess, Joel Veness, Alex Pritzel, Pablo Sprechmann, Siddhant M. Jayakumar, Tom McGrath, Kevin Miller, Mohammad Azar, Ian Osband, Neil Rabinowitz, András György, Silvia Chiappa, Simon Osindero, Yee Whye Teh, Hado van Hasselt, Nando de Freitas, Matthew Botvinick, and Shane Legg. Meta-learning of sequential strategies, 2019. URL `https://arxiv.org/abs/2008.02790`.

[18] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley Series in Probability and Statistics. John Wiley & Sons, Inc., 1994.

[19] Kate Rakelly, Aurick Zhou, Chelsea Finn, Sergey Levine, and Deirdre Quillen. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5331–5340. PMLR, 09–15 Jun 2019. URL `http://proceedings.mlr.press/v97/rakelly19a.html`.

[20] Frederik Ruelens, Bert J. Claessens, Stijn Vandael, Bart De Schutter, Robert Babuška, and Ronnie Belmans. Residential demand response of thermostatically controlled loads using batch reinforcement learning. *IEEE Transactions on Smart Grid*, 8(5):2149–2159, Sep. 2017. ISSN 1949-3061. doi: 10.1109/TSG.2016.2517211.

[21] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016. ISSN 1476-4687. doi: 10.1038/nature16961. URL `https://doi.org/10.1038/nature16961`.

[22] Bradly Stadie, Ge Yang, Rein Houthooft, Peter Chen, Yan Duan, Yuhuai Wu, Pieter Abbeel, and Ilya Sutskever. The importance of sampling inmeta-reinforcement learning. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL `https://proceedings.neurips.cc/paper/2018/file/d0f5722f11a0cc839fa2ca6ea49d8585-Paper.pdf`.

[23] Richard S. Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3(1):9–44, 1988.

[24] Jane X. Wang, Zeb Kurth-Nelson, Hubert Soyer, Joel Z. Leibo, Dhruva Tirumala, Rémi Munos, Charles Blundell, Dharshan Kumaran, and Matt M. Botvinick. Learning to reinforcement learn. In *CogSci*, 2017. URL https://mindmodeling.org/cogsci2017/papers/0252/index.html.

[25] Jane X. Wang, Michael King, Nicolas Porcel, Zeb Kurth-Nelson, Tina Zhu, Charlie Deck, Peter Choy, Mary Cassin, Malcolm Reynolds, Francis Song, Gavin Buttimore, David P. Reichert, Neil Rabinowitz, Loic Matthey, Demis Hassabis, Alexander Lerchner, and Matthew Botvinick. Alchemy: A structured task distribution for meta-reinforcement learning, 2021. URL https://arxiv.org/abs/2102.02926.

[26] Christopher J. C. H. Watkins. *Learning from delayed rewards*. Phd thesis, King's College, Cambridge, UK, 1989.

[27] Zhiang Zhang, Adrian Chong, Yuqi Pan, Chenlu Zhang, and Khee Poh Lam. Whole building energy model for hvac optimal control: A practical framework based on deep reinforcement learning. *Energy and Buildings*, 199:472–490, 2019. ISSN 0378-7788. doi: https://doi.org/10.1016/j.enbuild.2019.07.029. URL https://www.sciencedirect.com/science/article/pii/S0378778818330858.

[28] Luisa Zintgraf, Kyriacos Shiarlis, Maximilian Igl, Sebastian Schulze, Yarin Gal, Katja Hofmann, and Shimon Whiteson. Varibad: A very good method for bayes-adaptive deep rl via meta-learning. In *International Conference on Learning Representations*, 2020.