Reinforcement Learning in the Wild with Maximum Likelihood-based Model Transfer

Hannes Eriksson[*], **Tommy Tram**
Zenseact
Chalmers University of Technology
Gothenburg, Sweden

Debabrota Basu
Univ. Lille, Inria, CNRS
Centrale Lille, UMR 9189 – CRIStAL
Lille, France

Mina Alibeigi
Zenseact
Gothenburg, Sweden

Christos Dimitrakakis
University of Oslo, University of Neuchatel
Neuchatel, Switzerland

Abstract

In this paper, we study the problem of transferring the available Markov Decision Process (MDP) models to learn and plan efficiently in an unknown but similar MDP. We refer to it as *Model Transfer Reinforcement Learning (MTRL)* problem. First, we formulate MTRL for discrete MDPs and Linear Quadratic Regulators (LQRs) with continuous state actions. Then, we propose a generic two-stage algorithm, MLEMTRL, to address the MTRL problem in discrete and continuous settings. In the first stage, MLEMTRL uses a *constrained Maximum Likelihood Estimation (MLE)*-based approach to estimate the target MDP model using a set of known MDP models. In the second stage, using the estimated target MDP model, MLEMTRL deploys a model-based planning algorithm appropriate for the MDP class. Theoretically, we prove worst-case regret bounds for MLEMTRL both in realisable and non-realisable settings. We empirically demonstrate that MLEMTRL allows faster learning in new MDPs than learning from scratch and achieves near-optimal performance depending on the similarity of the available MDPs and the target MDP.

1 Introduction

Deploying autonomous agents in the real world poses a wide variety of challenges. As in [DALM$^+$21], we are often required to learn the real-world model with limited data, and use it to plan to achieve satisfactory performance in the real world. There might also be safety and reproducibility constraints, which require us to track a model of the real-world environment [SBL21]. In light of these challenges, we attempt to construct a framework that can aptly deal with optimal decision making for a novel task, by leveraging external knowledge. As the novel task is unknown, we adopt the Reinforcement Learning (RL) [SB18] framework to guide an agent's learning process and to achieve near-optimal decisions.

An RL agent interacts directly with the environment to improve its performance. Specifically, in model-based RL, the agent tries to learn a model of the environment and then use it to improve performance [MBP$^+$23]. In many applications, the depreciation in performance due to sub-optimal model learning can be paramount. For example, if the agent interacts with living things or expensive equipment, decision-making with an imprecise model might incur significant cost [PN17]. In such instances, boosting the model learning by leveraging external knowledge from the existing models,

[*]Contact at `hannes.eriksson@zenseact.com`.

Workshop on Aligning Reinforcement Learning Experimentalists and Theorists (ARLET 2024).

such as simulators [PAZA18], physics-driven engines, etc., can be of great value [TJS08]. A model trained on simulated data may perform reasonably well when deployed in a new environment, given the novel environment is *similar enough* to the simulated model. Also, RL algorithms running on different environments yield data and models that can be used to plan in another similar enough real-life environment. In this work, we study the problem where we have access to multiple source models built using simulators or data from other environments, and we want to transfer the source models to perform efficient model-based RL in a different real-life environment.

Example 1. *Let us consider that a company is designing autonomous driving agents for different countries in the world. The company has designed two RL agents that have learned to drive well in USA and UK. Now, the company wants to deploy a new RL agent in India. Though all the RL agents are concerned with the same task, i.e. driving, the models encompassing driver behaviors, traffic rules, signs, etc., can differ for each. For example, UK and India have left-handed traffic, while the USA has right-handed traffic. However, learning a new controller* specifically *for every new geographic location is computationally expensive and time-consuming, as both data collection and learning take time. Thus, the company might use the models learned for UK and USA, to estimate the model for India, and use it further to build a new autonomous driving agent (RL agent). Hence, being able to transfer the source models to the target environment allows the company to use existing knowledge to build an efficient agent faster and resource efficiently.*

We address this problem of model transfer from source models to a target environment to plan efficiently. We observe that this problem falls at the juncture of *transfer learning* and *reinforcement learning* [TS09, Laz12, LB17]. [Laz12] enlists three approaches to transfer knowledge from the *source tasks* to a *target task*. (i) *Instance transfer:* data from the source tasks is used to guide decision-making in the novel task [TJS08]. (ii) *Representation transfer:* a representation of the task, such as learned neural network features, are transferred to perform the new task [ZSP18]. (iii) *Parameter transfer:* the parameters of the RL algorithm or *policy* are transferred [RCG$^+$15]. In our paper, the source tasks are equivalent to the source models, and the target task is the target environment. Moreover, we adopt the **model transfer reinforcement learning** approach (MTRL), which encompasses both (i) and (ii) (Section 3).

[Lan06] describes three possible benefits of transfer learning. The first is **learning speed improvement**, i.e. decreasing the amount of data required to learn the solution. Secondly, **asymptotic improvement**, where the solution results in better asymptotic performance. Lastly, **jumpstart improvement**, where the initial proxy model serves as a better starting solution than that of one learning the true model from scratch. In this work, we propose a new algorithm to transfer RL that achieves both learning speed improvement and jumpstart improvement (Section 5). However, we might not find an asymptotic improvement in performance if compared with the best and unbiased algorithm in the true setting. Rather, we aim to achieve a model estimate that allows us to plan accurately in the target MDP (Section A).

Contributions. We aim to answer two central questions:
1. *How can we accurately construct a model using a set of source models for an RL agent deployed in the wild?*
2. *Does the constructed model allow efficient planning and yield improvements over learning from scratch?*
In this paper, we address these questions as follows:
1. *A Taxonomy of MTRL:* First, we formulate the problem with the Markov Decision Processes (MDPs). We further provide a taxonomy of the problem depending on a discrete or continuous set of source models, and whether the target model is realisable by the source models (Section 3).

2. *Algorithm Design with MLE:* Following that, we design a two-stage algorithm MLEMTRL to plan in an unknown target MDP (Section 4). In the first stage, MLEMTRL uses a Maximum Likelihood Estimation (MLE) approach to estimate the target MDP using the source MDPs. In the second stage, MLEMTRL uses the estimated model to perform model-based planning. We instantiate MLEMTRL for discrete state-action (tabular) MDPs and Linear Quadratic Regulators (LQRs). We also derive a generic bound on the goodness of the policy computed using MLEMTRL (Section A). We further provide a meta-algorithm, called Meta-MLEMTRL, to control the adaptation to the non-realisable setting (Section B).

3. *Performance Analysis:* In Section 5, we empirically verify whether MLEMTRL improves the performance for unknown tabular MDPs and LQRs than learning from scratch. MLEMTRL exhibits learning speed improvement for tabular MDPs and LQRs. For LQRs, it incurs learning speed

improvement and asymptotic improvement. We also observe that the more similar the target and source models are, the better the performance of MLEMTRL, as indicated by the theoretical analysis. An ablation study of Meta-MLEMTRL under realisable and non-realisable settings further shows provable improvements yielded in the asymptotic and non-realisable regimes.

Before elaborating on the contributions, we posit this work in the existing literature (Section 1) and discuss the background knowledge of MDPs and MLEs (Section 2).

Related Works. Our work on Model Transfer Reinforcement Learning is situated in the field of Transfer RL (TRL) and also is closely related to the multi-task RL and Bayesian multi-task RL literature. In this section, we elaborate on these connections.

TRL is widely studied in Deep Reinforcement Learning. [ZLJZ23] introduces different ways of transferring knowledge, such as *policy transfer*, where the set of source MDPs $\mathcal{M}_s$ has a set of expert policies associated with them. The expert policies are used together with a new policy for the novel task by transferring knowledge from each policy. [RCG$^+$15] uses this approach, where a student learner is combined with a set of teacher networks to guide learning in multi-task RL. [PBS15] develops an actor-critic structure to learn ways to transfer its knowledge to new domains. [AKS19] invokes generalisation across Q-functions by learning a master policy. Here, *we focus on model transfer instead of policy*.

Another seminal work in TRL, by [TS09] distinguishes between *multi-task learning* and *transfer learning*. Multi-task learning deals with problems where the agent aims to learn from a distribution over scenarios, whereas transfer learning makes no specific assumptions about the source and target tasks. Thus, in transfer learning, the tasks could involve different state and action spaces and different transition dynamics. Specifically, we focus on **model-transfer** [AS97] approach to TRL, where the state-action spaces and also dynamics can be different. [AS97] performs model transfer for a target task with an identical transition model. Thus, the main consideration is to transfer knowledge to tasks with the same dynamics but varying rewards. [LB17] assumes a context similar to that of [AS97], where the model dynamics are identical across environments. In our work, we rather assume that the reward function is the same, but the transition models are different. We believe this is an interesting question as the harder part of learning an MDP is learning the transition model. These works explicate a deep connection between the fields of *multi-task learning* and *TRL*. In general, TRL can be viewed as an extension of multi-task RL, where multiple tasks can either be learned simultaneously or have been learned *a priori*. This flexibility allows us to learn even in settings where the state-actions and transition dynamics are different among tasks. [RBGM17] describes a multi-task Maximum Likelihood Estimation procedure for optimal control of an aircraft. They identify a mixture of Gaussians, where the mixture is over each of the tasks. Here, we adopt an MLE approach to TRL to optimise performance for the target MDP (or a target task) rather than restricting to a mixture of Gaussians.

The Bayesian approach to multi-task RL [WFRT07, LG10] tackles the problem of learning jointly how to act in multiple environments. [LG10] handles the *open-world assumption*, i.e. the number of tasks is unknown. This allows them to transfer knowledge from existing tasks to a novel task, using value function transfer. However, this is significantly different from our setting, as we are considering model-based transfer. Further, *we adopt an MLE-based framework in lieu of the full Bayesian procedure described in their work*. In Bayesian RL, [TSZ22] also investigates a learning technique to generalise over multiple problem instances. By sampling a large number of instances, the method is expected to learn how to generalise from the existing tasks to a novel task. We do not assume access to such prior or posterior distributions to sample from.

There is another related line of work, namely multi-agent transfer RL [DSC19]. For example, [LLC$^+$23] develops a TRL framework for autonomous driving using federated learning. They accomplish this by aggregating knowledge for independent agents. This setting is different from general transfer learning but could be incorporated if the source tasks are learned simultaneously with the target task. This requires cooperation among agents, which is out of the scope.

2 Background

Here, we introduce the important concepts on which this work is based upon. Firstly, we introduce the way we model the dynamics of the tasks. Secondly, we describe the Maximum Likelihood Estimation framework used in this work.

Markov Decision Process (MDP). We study sequential decision-making problems that can be represented as MDPs [Put14]. An MDP $\mu = (\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}, \gamma)$ consists of a discrete or continuous state space denoted by $\mathcal{S}$, a discrete or continuous action-space $\mathcal{A}$, a reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ which determines the quality of taking action a in state s, and a transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ inducing a probability distribution over the successor states s' given a current state s and action a. Finally, in the infinite-horizon formulation, a discount factor $\gamma \in [0, 1)$ is assigned. The overarching objective for the agent is to compute a decision-making policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ that maximises the expected sum of future discounted rewards up until the horizon T: $V_\mu^\pi(s) = \mathbb{E}\Big[\sum_{t=0}^{T} \gamma^t \mathcal{R}(s_t, a_t)\Big]$. $V_\mu^\pi(s)$ is called the value function of policy π for MDP μ. Let $V_\mu^* = V_\mu^{\pi^*}$ denote the optimal value function. The technique used to obtain the the optimal policy $\pi^* = \sup_\pi V_\mu^\pi$ depends on the MDP class. The MDPs with discrete state-action spaces are referred to as tabular MDPs. In this paper, we also study a class of MDPs with continuous state-action spaces, namely Linear Quadratic Regulators (LQRs) [Kal60]. In tabular MDPs, we employ VALUEITERATION [Put14] for model-based planning, whereas in the LQR setting, we use RICCATIITERATION [Wil71].

The standard metric used to measure the performance of a policy π [Bel82] for an MDP μ is *regret* $R(\mu, \pi)$. Regret is the difference between the optimal value function and the value function of π. In this work, we extend the definition of regret for MTRL, where the optimality is taken for a policy class in the target MDP.

Maximum Likelihood Estimation (MLE). One of the most popular methods of constructing point estimators is the *Maximum Likelihood Estimation* [CB21] framework. Given a density function $f(x \mid \theta_1, \ldots, \theta_n)$ and associated i.i.d. data $X_1, \ldots, X_t$, the goal of the MLE scheme is to maximise, $\ell(\theta \mid x) \triangleq \ell(\theta_1, \ldots, \theta_n \mid x_1, \ldots, x_t) \triangleq \log \prod_{i=1}^{t} f(x_i \mid \theta_1, \ldots, \theta_n)$. $\ell(\cdot)$ is called the log-likelihood function. The set of parameters θ maximising $\ell(\theta \mid x)$ is called the *maximum likelihood estimator* of θ given the data $X_1, \ldots, X_t$. MLE has many desirable properties that we leverage in this work. For example, the MLE satisfies *consistency*, i.e. under certain conditions, it achieves optimality even for *constrained* MLE. An estimator being consistent means that if the data $X_1, \ldots, X_t$ is generated by $f(\cdot \mid \theta)$ and as $t \to \infty$, the estimate almost surely converges to the true parameter θ. [KW56] shows that MLE admits the consistency property given the following assumptions hold. The model is *identifiable*, i.e. the densities at two parameter values must be different unless the two parameter values are identical. Further, the parameter space is *compact* and *continuous*. Finally, if the log-density is *dominated*, one can establish that MLE converges to the true parameter almost surely [NP87]. For problems where the likelihood is unbounded, flat, or otherwise unstable, one may introduce a penalty term in the objective function. This approach is called *penalised maximum likelihood estimation* [CRI03, OBM23]. As we in our work are mixing over known parameters, we do not need to add regularisation to our objective to guarantee convergence.

In this work, we iteratively collect data and compute new point estimates of the parameters and use them in our decision-making procedure. To carry out MLE, a likelihood function has to be chosen. In this work, we investigate two such likelihood functions in Section 4, one for each model class.

3 A Taxonomy of Model Transfer RL

Now, we formally define the Model Transfer RL problem and derive a taxonomy of settings encountered in MTRL.

MTRL: Problem Formulation. Let us assume that we have access to a set of source MDPs $\mathcal{M}_s \triangleq \{\mu_i\}_{i=1}^m$. The individual MDPs can belong to a finite or infinite but compact set depending on the setting. For example, for tabular MDPs with finite state-actions, this is always a finite set. Whereas for MDPs with continuous state-actions, the transitions can be parameterised by real-valued vectors/matrices, corresponding to an infinite but compact set. Given access to $\mathcal{M}_s$, we want to find an optimal policy for an unknown target MDP μ^* that we encounter while deploying RL in the wild. At each step t, we use $\mathcal{M}_s$ and the data observed from the target MDP $D_{t-1} \triangleq \{s_0, a_0, s_1, \ldots, s_{t-1}, a_{t-1}, s_t\}$ to construct an estimate of μ^*, say $\hat{\mu}^t$. Now, we use $\hat{\mu}^t$ to run a model-based planner, such as VALUEITERATION or RICCATIITERATION, that leads to a policy π^t. After completing this planning step, we interact with the target MDP using π_t that yields an action a_t, and leads to observing s_{t+1}, r_{t+1}. We update the dataset with these observations: $D_t \triangleq D_{t-1} \cup \{a_t, s_t\}$. Here, we assume that all the source and target MDPs share the same reward function $\mathcal{R}$. We do not restrict the state-action space of target and source MDPs.

Our goal is to compute a policy π^t that performs as close as possible with respect to the optimal policy π^* for the target MDP as the number of interactions with the target MDP $t \to \infty$. This allows us to define a notion of regret for MTRL: $R(\mu^*, \pi_t) \triangleq V^*_{\mu^*} - V^{\pi_t}_{\mu^*}$. Here, π_t is a function of the source models $\mathcal{M}_s$, the data collected from target MDP D_t, and the underlying MTRL algorithm. The goal of an MTRL algorithm is to minimise $R(\mu^*, \pi_t)$. For the parametric policies π_θ with $\theta \in \Theta \subset \mathbb{R}^d$, we can specialise the regret further for this parametric family: $R(\mu^*, \pi_{\theta_t}) = V^{\pi_{\theta^*}}_{\mu^*} - V^{\pi_{\theta_t}}_{\mu^*}$. For example, for LQRs, we by default work with linear policies. We use this notion of regret in our theoretical and experimental analysis.

Three Classes of MTRL Problems. We begin by illustrating MTRL using Figure 1. In the figure, the source MDPs $\mathcal{M}_s$ are depicted in red. This green area is the convex hull spanned by the source models $\mathcal{C}(\mathcal{M}_s)$. The target MDP μ^*, the best representative within the convex hull of the source models μ, and the estimated MDP $\hat{\mu}$ are shown in blue, yellow, and purple, respectively. If the target model is *inside* the convex hull, we call it a **realisable** setting. whereas If the target model is outside (as in Figure 1), then we have a **non-realisable** setting.

Figure 1 also shows that the total deviation of the estimated model from the target model depends on two sources of errors: (i) realisability, i.e. how far is the target MDP μ^* from the convex hull of the source models $\mathcal{C}(\mathcal{M}_s)$ available to us, and (ii) estimation, i.e. how close is the estimated MDP $\hat{\mu}$ to the best possible representation μ of the target MDP. In the realisable case, the realisability gap can be reduced to zero, but not otherwise. This approach allows us to decouple the effect of the expressibility of the source models and the goodness of the estimator.

Now, we further elaborate on these three classes and the corresponding implications of performing MLE.

I. Finite and Realisable Plausible Models. If the true model μ^* is one of the target models, i.e. $\hat{\mu} \in \mathcal{M}_s$, we have to identify the target MDP from a finite set of plausible MDPs. Thus, the corresponding MLE involves a finite set of parameters, i.e. the parameters of the source MDPs $\mathcal{M}_s$. We compute the MLE $\hat{\mu}$ by solving the optimisation problem:

$$\hat{\mu} \in \arg\max_{\mu' \in \mathcal{M}_s} \log \mathbb{P}(D_t \,|\, \mu'), D_t \sim \mu^*. \tag{1}$$

This method may serve as a reasonable heuristic for the TRL problem, where the target MDP is the same as or reasonably close to one of the source MDPs. However, this method will potentially be sub-optimal if the target MDP is too different from the source MDPs. Even if μ^* lies within the convex hull of the source MDPs (the green area in Figure 1), this setting restricts the selection of a model to one of the red boxes. Thus, this setting fails to leverage the expressiveness of the source models as MLE allows us to accurately estimate models which are also in $\mathcal{C}(\mathcal{M}_s)$. Thus, we focus on the two settings described below.

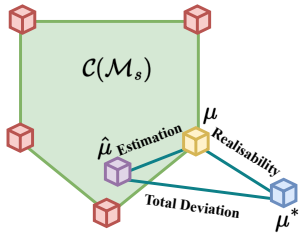

Figure 1: An illustration of the MTRL setting. The source models $\mathcal{M}_s$ are the red boxes. The green area is the convex hull $\mathcal{C}(\mathcal{M}_s)$ spanned by the source models. The target MDP μ^* is displayed in blue, and the best proxy model is contained in the convex hull μ in yellow. Finally, the estimator of the best proxy model $\hat{\mu}$ is shown in purple.

II. Infinite and Realisable Plausible Models. In this setting, the target MDP μ^* is in the convex hull $\mu^* \in \mathcal{C}(\mathcal{M}_s)$ of the source MDPs. Thus, for Class I, we extend the parameter space considered in MLE to an infinite but compact parameter set.

Let us define the convex hull as $\mathcal{C}(\mathcal{M}_s) \triangleq \{\mu_1 w_1 + \ldots + \mu_m w_m \,|\, \mu_i \in \mathcal{M}_s, w_i \geq 0, i = 1, \ldots, m, \sum_{i=1}^m w_i = 1\}$. Then, the corresponding MLE problem with the corresponding likelihood function is

$$\hat{\mu} \in \arg\max_{\mu' \in \mathcal{C}(\mathcal{M}_s)} \log \mathbb{P}(D_t \,|\, \mu'), D_t \sim \mu^*. \tag{2}$$

Since $\mathcal{C}(\mathcal{M}_s)$ induces a compact subset of model parameters $\mathcal{M}' \subset \mathcal{M}$, Equation (2) leads to a *constrained maximum likelihood estimation problem* [AS58]. It implies that if the parameter corresponding to the target MDP is in $\mathcal{M}'$, it can be correctly identified. In the case where the optimum lies inside, we can use constrained MLE to accurately identify the true parameters given enough experience from μ^*. This approach allows us to leverage the expressibility of the source models completely. However, μ^* might lie outside or on the boundary. Either of them may pose problems for the optimiser.

Algorithm 1 Maximum Likelihood Estimation for Model-based Transfer RL (MLEMTRL)

Input: weights $\boldsymbol{w}^0$, m source MDPs $\mathcal{M}_s$, data D_0, discount factor γ, iterations T.
for $t = 0, \dots, T$ **do**
  // STAGE 1: MODEL ESTIMATION //
  $\boldsymbol{w}^{t+1} \leftarrow \textsc{Optimiser}(\log \mathbb{P}(D_t \mid \Sigma_{i=1}^m w_i \mu_i), \boldsymbol{w}^t)$
  Estimate the MDP: $\mu^{t+1} = \sum_{i=1}^m w_i \mu_i$
  // STAGE 2: MODEL-BASED PLANNING //
  Compute the policy: $\pi^{t+1} \in \arg\max_\pi V^\pi_{\mu^{t+1}}$
  // CONTROL //
  Observe $s_{t+1}, r_{t+1} \sim \mu^*(s_t, a_t), a_t \sim \pi^{t+1}(s_t)$
  Update the dataset $D_{t+1} = D_t \cup \{s_t, a_t, s_{t+1}, r_{t+1}\}$
end for
return An estimated MDP model μ^T and a policy π^T

III. Infinite and Non-realisable Plausible Models. This class is similar to Class II with the important difference that the true parameter μ^* is outside the convex hull of source MDPs $\mathcal{C}(\mathcal{M}_s)$, and thus, the corresponding parameter is not in the induced parameter subset $\mathcal{M}'$. This key difference means the true parameters cannot be correctly identified. Instead, the objective is to identify the best proxy model $\mu \in \mathcal{M}'$. The performance loss for using μ instead of μ^* is intimately related to the model dissimilarity $||\mu^* - \mu||_1$. This allows us to describe the limitation of expressivity of the source models by defining the *realisability gap*: $\epsilon_{\text{Realise}} \triangleq \min_{\mu \in \mathcal{C}(\mathcal{M}_s)} \|\mu^* - \mu\|_1$. The realisability gap becomes important while dealing with continuous state-action MDPs with parameterised dynamics, such as LQRs.

4 MLEMTRL: MTRL with Maximum Likelihood Model Transfer

Now, we present the proposed algorithm, MLEMTRL. The algorithm consists of two stages, a *model estimation* stage, and a *planning* stage. After having obtained a plan, then the agent will carry out its decision-making in the environment to acquire new experiences. We sketch an overview of MLEMTRL in Algorithm 1. For completeness, we also provide an extension to MLEMTRL called Meta-MLEMLTRL. This extension combines the MLEMTRL estimated model with the empirical model of the target task. This allows us to identify the true model even in the non-realisable setting. The details of this algorithm are available in Section B.

4.1 Stage 1: Model Estimation

The first stage of the proposed algorithm is *model estimation*. During this procedure, the likelihood of the data needs to be computed for the appropriate MDP class. In the tabular setting, we use a product of multinomial likelihoods, where the data likelihood is over the distribution of successor states s' for a given state-action pair (s, a). In the LQR setting, we use a linear-Gaussian likelihood, which is also expressed as a product over data observed from the target MDP.

Likelihood for Tabular MDPs. The log-likelihood that we attempt to maximise in tabular MDPs is a product over $|\mathcal{S}| \times |\mathcal{A}|$ of pairs of multinomials, where p_i is the probability of event i, $n^{s,a}$ is the number of times the state-action pairs (s, a) appear in the data D_t, and $x_i^{s,a}$ is the number of times the state-action pair (s, a, s_i) occurs in the data. That is, $\sum_{i=1}^{|\mathcal{S}|} x_i^{s,a} = n^{s,a}$. Specifically,

$$\log \mathbb{P}(D_t \mid \boldsymbol{p}) = \log \left(\prod_{s,a} n^{s,a}! \prod_{i=1}^{|\mathcal{S}|} \frac{p_i^{x_i^{s,a}}}{x_i^{s,a}!} \right) \tag{3}$$

Likelihood for Linear-Gaussian MDPs. For continuous state-action MDPs, we use a linear-Gaussian likelihood. In this context, let d_s be the dimensionality of the state-space, $\boldsymbol{s} \in \mathbb{R}^{d_s}$ and d_a be the dimensionality of the action-space. Then, the mean function $\mathbf{M}$ is a $\mathbb{R}^{d_s} \times \mathbb{R}^{d_a + d_s}$ matrix. The mean visitation count to the successor state $\boldsymbol{s}'_t$ when an action $\boldsymbol{a}_t$ is taken at state $\boldsymbol{s}_t$ is given by $\mathbf{M}(\boldsymbol{a}_t, \boldsymbol{s}_t)$. We denote the corresponding covariance matrix of size $\mathbb{R}^{d_s} \times \mathbb{R}^{d_s}$ by $\mathbf{S}$. Thus, we express the log-likelihood by $\log \mathbb{P}(D_t \mid \mathbf{M}, \mathbf{S}) = \log \prod_{i=1}^t \frac{\exp\left(-\frac{1}{2} \boldsymbol{v}^\top \mathbf{S}^{-1} \boldsymbol{v}\right)}{(2\pi)^{d_s/2} |\mathbf{S}|^{1/2}}$, where $\boldsymbol{s}'_i - \mathbf{M}(\boldsymbol{a}_i, \boldsymbol{s}_i) = \boldsymbol{v}$.

Model Estimation as a Mixture of Models. As the optimisation problem involves weighing multiple source models together, we add a weight vector $\boldsymbol{w} \in [0,1]^m$ with the usual property that $\boldsymbol{w}$ sum to 1. This addition results in another outer product over the likelihoods shown above. Henceforth, μ will refer to either the parameters associated with the product-Multinomial likelihood or the linear-Gaussian likelihood, depending on the model class.

$$\min_{\boldsymbol{w}} \quad \log \mathbb{P}(D_t \mid \Sigma_{i=1}^m w_i \mu_i), D_t \sim \mu^*, \mu_i \in \mathcal{M}_s, \text{s.t.} \quad \sum_{i=1}^m w_i = 1, w_i \geq 0. \tag{4}$$

Because of the constraint on $\boldsymbol{w}$, this is a constrained nonlinear optimisation problem. We can use any optimiser algorithm, denoted by OPTIMISER, for this purpose.

OPTIMISER. In our implementations, we use Sequential Least-Squares Quadratic Programming (SLSQP) [Kra88] as the OPTIMISER. SLSQP is a quasi-Newton method solving a quadratic programming subproblem for the Lagrangian of the objective function and the constraints.

Specifically, in Line 4 of Algorithm 1, we compute the next weight vector $\boldsymbol{w}^{t+1}$ by solving the optimisation problem in Eq. (4). Let $f(\boldsymbol{w}) = \log \mathbb{P}(D_t \mid \Sigma_{i=1}^m w_i \mu_i)$. Further, let $\lambda = \{\lambda_1, \ldots, \lambda_m\}$ and κ be Lagrange multipliers. We then define the Lagrangian $\mathcal{L}$, $\mathcal{L}(\boldsymbol{w}, \lambda, \kappa) = f(\boldsymbol{w}) - \lambda^\top \boldsymbol{w} - \kappa(1 - \mathbf{1}^\top \boldsymbol{w})$. Here, $\boldsymbol{w}^k$ is the k-th iterate. Finally, taking the local approximation of Eq. (4), we define the optimisation problem as $\min_{\boldsymbol{d}} \frac{1}{2}\boldsymbol{d}^\top \nabla^2 \mathcal{L}(\boldsymbol{w}, \lambda, \kappa)\boldsymbol{d} + \nabla f(\boldsymbol{w}^k)\boldsymbol{d} + f(\boldsymbol{w}^k)$ s.t. $\boldsymbol{d} + \boldsymbol{w}^k \geq 0, \mathbf{1}^\top \boldsymbol{w}^k = 1$.This minimisation problem yields the search direction $\boldsymbol{d}_k$ for the k-th iteration. Applying this iteratively and using the construction above ensures that the constraints posed in Eq. (4) are adhered to at every step of MLEMTRL. At convergence, the k-th iterate, $\boldsymbol{w}^k$ is considered as the next $\boldsymbol{w}^{t+1}$ in Line 1 of Algorithm 1.

4.2 Stage 2: Model-based Planning

When an appropriate model μ^t has been identified at time step t, the next stage of the algorithm involves model-based planning in the estimated MDP. We describe two model-based planning techniques, VALUEITERATION and RICCATIITERATION for tabular MDPs and LQRs, respectively.

VALUEITERATION. Given the model, μ^t and the associated reward function $\mathcal{R}$, the optimal value function of μ^t can be computed iteratively as [SB18]:

$$V^*_{\mu^t}(s) = \max_a \sum_{s'} \mathcal{T}^a_{s,s'} \Big(\mathcal{R}(s,a) + \gamma V^*_{\mu^t}(s') \Big). \tag{5}$$

The fixed-point solution to Eq. (5) is the optimal value function. When the optimal value function has been obtained, one can simply select the action maximising the action-value function. Let π^{t+1} be the policy selecting the maximising action for every state, then π^{t+1} is the policy the model-based planner will use at time step $t+1$.

RICCATIITERATION. A LQR-based control system, and thus, the corresponding MDP, is defined by four system matrices [Kal60]: $\mathbf{A}, \mathbf{B}, \mathbf{Q}, \mathbf{R}$. The matrices $\mathbf{A}, \mathbf{B}$ are associated with the transition model $\boldsymbol{s}_{t+1} - \boldsymbol{s}_t = \mathbf{A}\boldsymbol{s}_t + \mathbf{B}\boldsymbol{a}_t$. The matrices $\mathbf{Q}, \mathbf{R}$ dictate the quadratic cost (or reward) of a policy π under an MDP μ is $V^\pi_\mu = \sum_{t=0}^T \boldsymbol{s}_t^\top \mathbf{Q} \boldsymbol{s}_t + \boldsymbol{a}_t^\top \mathbf{R} \boldsymbol{a}_t$. Optimal policy is identified following [Wil71], i.e. $\boldsymbol{a}_t = -\mathbf{K}\boldsymbol{s}_t$ at time t, where $\mathbf{K}$ is computed using $\mathbf{A}, \mathbf{B}, \mathbf{Q}, \mathbf{R}$. We refer to App. D for details.

5 Experimental Analysis

To benchmark the performance of MLEMTRL, we compare ourselves to a posterior sampling method (**PSRL**) [ORVR13], equipped with a combination of product-Dirichlet and product-Normal Inverse Gamma priors for the tabular setting, and Bayesian Multivariate Regression prior [Min00] for the continuous setting. In PSRL, at every round, a new model is sampled from the prior, and it learns in the target MDP from scratch. Finally, for model-based planning, we use RICCATIITERATIONS to obtain the optimal linear controller for the sampled model. In the continuous action setting, we compare the performance to the baseline algorithm multi-task soft-actor critic (**MT-SAC**) [HZAL18, YQH$^+$20] and a modified **MT-SAC-TRL** using data from the novel task during learning. In the tabular MDP setting, we compare against multi-task proximal policy optimisation (**MT-PPO**) [SWD$^+$17, YQH$^+$20] and similarly **MT-PPO-TRL**.

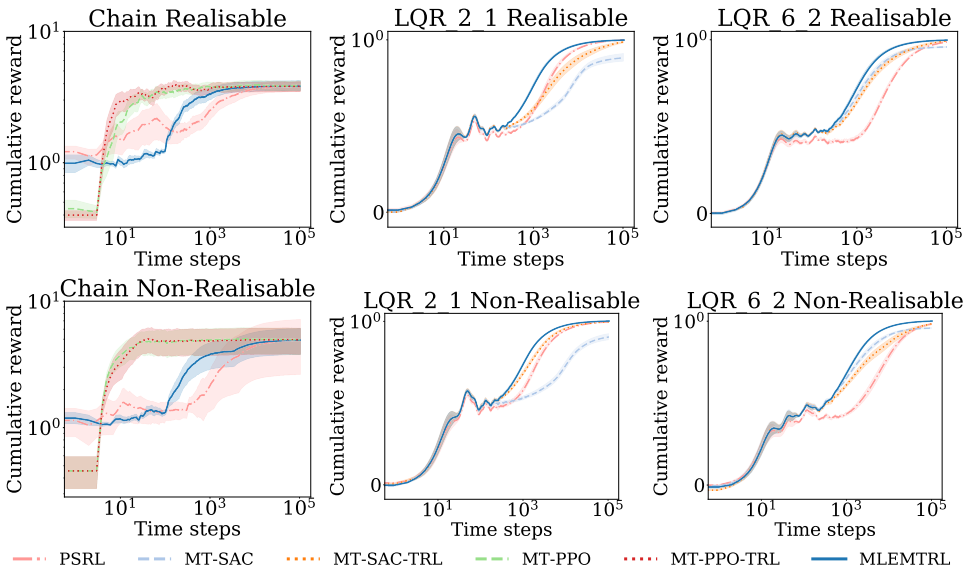

Figure 2: Depicted is the average cumulative reward at every time step computed over 10 novel tasks in the realisable/non-realisable setting. The shaded regions represent the standard error of the average cumulative reward at the time step.

The objectives of our empirical study are three-fold:
1. How does MLEMTRL impact performance in terms of **learning speed**, **jumpstart improvement** and **asymptotic convergence** compared to our baseline?
2. What is the performance loss of MLEMTRL in the **non-realisable setting**?
3. How does Meta-MLEMTRL perform in the **non-realisable setting**?

We conduct two kinds of experiments to verify our hypotheses. Firstly, in the upper row of Figure 2, we consider the realisable setting, where the novel task μ^* is part of the convex hull $\mathcal{C}(\mathcal{M}_s)$. In this case, we are looking to identify an improvement in some or all of the aforementioned qualities compared to the baselines. Further, in the bottom row of Figure 2, we investigate whether the algorithm can generalise to the case beyond what is supported by the theory in Section 3. We begin by recalling the goals of the transfer learning problem [Lan06].

1. *Learning Speed Improvement:* A learning speed improvement would be indicated by the algorithm reaching its asymptotic convergence with less data.
2. *Asymptotic Improvement:* An asymptotic improvement would mean the algorithm converges asymptotically to a superior solution to that one of the baseline.
3. *Jumpstart Improvement:* A jumpstart improvement can be verified by the behaviour of the algorithm during the early learning process. In particular, if the algorithm starts at a better solution than the baseline, or has a simpler optimisation surface, it may more rapidly approach better solutions with much less data.

RL Environments. We test the algorithms in a tabular MDP, i.e. Chain [DFR98], CartPole [BSA83], and two LQR tasks in *Deepmind Control Suite* [TDM$^+$18]: *dm_LQR_2_1* and *dm_LQR_6_2*. Further details on experimental setups are deferred to Appendix C.1.

(1) Impacts of Model Transfer with MLEMTRL. We begin by evaluating the proposed algorithm in the Chain environment. The results of the said experiment are available in the leftmost column of Figure 2. In it, we evaluate the performance of MLEMTRL against PSRL, MT-PPO, MT-PPO-TRL. The experiments are done by varying the slippage parameter $p \in [0.00, 0.50]$ and the results are computed for each different setup of Chain from scratch. In this experiment, we can see the baseline algorithms MT-PPO and MT-PPO-TRL perform very well. This could partially be explained by PSRL and MLEMTRL not only having to learn the transition distribution but also the reward function. The value function transfer in the PPO-based baselines implicitly transfers not only the empirical transition model but also the reward function. We can see that MLEMTRL has improved learning speed compared to PSRL in both realisable and non-realisable settings. An additional experiment with a known reward function across tasks is shown in Figure 7 in Appendix.

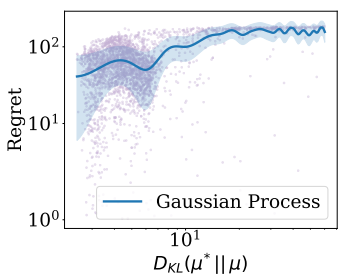

Figure 3: A log-log plot of regret vs. KL-divergence between the true MDP and the best proxy model in CartPole. Thick blue line is a Gaussian Process regression fitted on observed data (in purple).

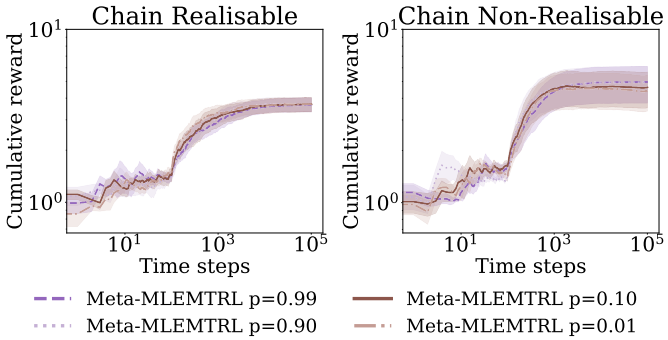

Figure 4: Figure depicting an ablation study of the prior parameter p in the Meta-MLEMTRL algorithm. The y-axis is the average cumulative reward at each time step computed over 10 novel tasks and the shaded region represents the standard error. When $p = 1$, the algorithm reduces to MLEMTRL. When $p = 0$, it reduces to standard maximum likelihood model estimation.

In the centre and rightmost columns of Figure 2, we can see the results of running the algorithms in the LQR settings with the baseline algorithms PSRL, MT-SAC and MT-SAC-TRL. The variation over tasks is given by the randomness over the stiffness of the joints in the problem. In these experiments, we can see a clear advantage of MLEMTRL compared to all baselines in terms of learning speed improvements, and in some cases, asymptotic performance.

In Figure 2, the performance metric is the average cumulative reward at every time step, for 10^5 time steps and the shaded region represents the standard deviation, where the statistics are computed over 10 independent tasks.

(2) Impact of Realisability Gap on Regret. Now, we further illustrate the observed relation between model dissimilarity and degradation in performance. Figure 3 depicts the regret against the KL-divergence of the target model to the best proxy model in the convex set. We observe that model dissimilarity influences the performance gap in MLEMTRL. This is also justified in Section A where the bounds have an explicit dependency on the model difference. In this figure, only the non-zero regret experiments are shown. This is to have an idea of which models result in poor performance. As its shown, it is those models that are very dissimilar. Additional results in Figure 5 in Appendix further illustrate the dependency on model similarity.

(3) Performance of Meta-MLEMTRL under Non-realisability. To validate the performance of the proposed meta algorithm Meta-MLEMTRL, we perform an ablation study over the prior hyperparameter p. In Figure 4, we illustrate the results of running the Meta-MLEMTRL in the Chain environment for both the realisable and non-realisable settings. The choice of p determines how much the algorithm should be biased towards the model estimated by MLEMTRL. When $p = 1$, Meta-MLEMTRL reduces to MLEMTRL. Similarly, if $p = 0$ then the algorithm will forego the MLEMTRL-estimate for the empirical estimate. Figure 4 shows that the performance of Meta-MLEMTRL is stable in long-run for different values of p. However, we identify that higher p values yield positive improvements in the cumulative reward over 10^5 steps, especially in the non-realisable setting. This indicates that the MLEMTRL-estimated model acts a good representation, while combined with the asymptotically converging empirical estimate obtained by Meta-MLEMTRL.

Summary of Results. In the experiments, we sought to identify whether the proposed algorithm shows superiority in terms of the transfer learning goals given by [Lan06]. In the LQR-based environments, we can see a clear superiority of MLEMTRL in terms of learning speed compared to all baselines and in some cases, an asymptotic improvement. In the Chain environment, the proposed algorithm, MLEMTRL, outperforms PSRL in terms of learning speed. Also, we perform an ablation study of Meta-MLEMTRL under realisable and non-realisable settings, demonstrating provable improvements in the asymptotic and non-realisable regimes.

Algorithms and Results for Non-relisable Settings. In appendix, we motivate the use of our framework in settings where an agent is to be deployed in a new domain, which is not realisable by existing, known domains. We provide a meta algorithm extending MLEMTRL for this setting (Appendix B). Finally, we prove worst-case performance bounds of the algorithm in both the realisable and non-realisable settings (Appendix A).

References

[AJO08] Peter Auer, Thomas Jaksch, and Ronald Ortner. Near-optimal regret bounds for reinforcement learning. *Advances in neural information processing systems*, 21, 2008.

[AKS19] Isac Arnekvist, Danica Kragic, and Johannes A Stork. Vpe: Variational policy embedding for transfer reinforcement learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 36–42. IEEE, 2019.

[AS58] John Aitchison and SD Silvey. Maximum-likelihood estimation of parameters subject to restraints. *The annals of mathematical Statistics*, 29(3):813–828, 1958.

[AS97] Christopher G Atkeson and Juan Carlos Santamaria. A comparison of direct and model-based reinforcement learning. In *Proceedings of international conference on robotics and automation*, volume 4, pages 3557–3564. IEEE, 1997.

[Bel82] David E Bell. Regret in decision making under uncertainty. *Operations research*, 30(5):961–981, 1982.

[BSA83] Andrew G Barto, Richard S Sutton, and Charles W Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE transactions on systems, man, and cybernetics*, (5):834–846, 1983.

[CB21] George Casella and Roger L Berger. *Statistical inference*. Cengage Learning, 2021.

[CRI03] Gabriela Ciuperca, Andrea Ridolfi, and Jérôme Idier. Penalized maximum likelihood estimator for normal mixtures. *Scandinavian Journal of Statistics*, 30(1):45–59, 2003.

[DALM+21] Gabriel Dulac-Arnold, Nir Levine, Daniel J Mankowitz, Jerry Li, Cosmin Paduraru, Sven Gowal, and Todd Hester. Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. *Machine Learning*, 110(9):2419–2468, 2021.

[DFR98] Richard Dearden, Nir Friedman, and Stuart Russell. Bayesian q-learning. *Aaai/iaai*, 1998:761–768, 1998.

[DSC19] Felipe Leno Da Silva and Anna Helena Reali Costa. A survey on transfer learning for multiagent reinforcement learning systems. *Journal of Artificial Intelligence Research*, 64:645–703, 2019.

[EDM03] Eyal Even-Dar and Yishay Mansour. Approximate equivalence of markov decision processes. In *Learning Theory and Kernel Machines*, pages 581–594. Springer, 2003.

[HZAL18] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

[Kal60] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. 1960.

[Kra88] Dieter Kraft. A software package for sequential quadratic programming. *Forschungsbericht- Deutsche Forschungs- und Versuchsanstalt fur Luft- und Raumfahrt*, 1988.

[KW56] Jack Kiefer and Jacob Wolfowitz. Consistency of the maximum likelihood estimator in the presence of infinitely many incidental parameters. *The Annals of Mathematical Statistics*, pages 887–906, 1956.

[Lan06] Pat Langley. Transfer of knowledge in cognitive systems. In *Talk, workshop on Structural Knowledge Transfer for Machine Learning at the Twenty-Third International Conference on Machine Learning*, 2006.

[Laz12] Alessandro Lazaric. Transfer in reinforcement learning: a framework and a survey. In *Reinforcement Learning*, pages 143–173. Springer, 2012.

[LB17] Romain Laroche and Merwan Barlier. Transfer reinforcement learning with shared dynamics. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[LG10] Alessandro Lazaric and Mohammad Ghavamzadeh. Bayesian multi-task reinforcement learning. In *ICML-27th International Conference on Machine Learning*, pages 599–606. Omnipress, 2010.

[LLC+23] Xinle Liang, Yang Liu, Tianjian Chen, Ming Liu, and Qiang Yang. Federated transfer reinforcement learning for autonomous driving. In *Federated and Transfer Learning*, pages 357–371. Springer, 2023.

[MBP+23] Thomas M Moerland, Joost Broekens, Aske Plaat, Catholijn M Jonker, et al. Model-based reinforcement learning: A survey. *Foundations and Trends® in Machine Learning*, 16(1):1–118, 2023.

[Min00] Thomas Minka. Bayesian linear regression. Technical report, Citeseer, 2000.

[NP87] Whitney K Newey and James L Powell. Asymmetric least squares estimation and testing. *Econometrica: Journal of the Econometric Society*, pages 819–847, 1987.

[OBM23] Reda Ouhamma, Debabrota Basu, and Odalric Maillard. Bilinear exponential family of mdps: frequentist regret bound with tractable exploration & planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 9336–9344, 2023.

[ORVR13] I.. Osband, D. Russo, and B. Van Roy. (more) efficient reinforcement learning via posterior sampling. In *Advances in Neural Information Processing Systems*, pages 3003–3011, 2013.

[PAZA18] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 3803–3810. IEEE, 2018.

[PBS15] Emilio Parisotto, Jimmy Lei Ba, and Ruslan Salakhutdinov. Actor-mimic: Deep multitask and transfer reinforcement learning. *arXiv preprint arXiv:1511.06342*, 2015.

[PN17] Athanasios S Polydoros and Lazaros Nalpantidis. Survey of model-based reinforcement learning: Applications on robotics. *Journal of Intelligent & Robotic Systems*, 86(2):153–173, 2017.

[Put14] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[QFPL20] Jian Qian, Ronan Fruit, Matteo Pirotta, and Alessandro Lazaric. Concentration inequalities for multinoulli random variables. *arXiv preprint arXiv:2001.11595*, 2020.

[RBGM17] Cédric Rommel, Joseph Frédéric Bonnans, Baptiste Gregorutti, and Pierre Martinon. Aircraft dynamics identification for optimal control. In *7th European Conference on Aeronautics and Space Sciences (EUCASS 2017)*, 2017.

[RCG+15] Andrei A Rusu, Sergio Gomez Colmenarejo, Caglar Gulcehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy distillation. *arXiv preprint arXiv:1511.06295*, 2015.

[RHG+21] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *The Journal of Machine Learning Research*, 22(1):12348–12355, 2021.

[SB18] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[SBL21] Julian Skirzyński, Frederic Becker, and Falk Lieder. Automatic discovery of interpretable planning strategies. *Machine Learning*, 110(9):2641–2683, 2021.

[SL05] Alexander L Strehl and Michael L Littman. A theoretical analysis of model-based interval estimation. In *Proceedings of the 22nd international conference on Machine learning*, pages 856–863, 2005.

[SWD+17] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[TDM+18] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

[TJS08] Matthew E Taylor, Nicholas K Jong, and Peter Stone. Transferring instances for model-based reinforcement learning. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 488–505. Springer, 2008.

[TS09] Matthew E Taylor and Peter Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(7), 2009.

[TSZ22] Aviv Tamar, Daniel Soudry, and Ev Zisselman. Regularization guarantees generalization in bayesian reinforcement learning through algorithmic stability. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 8423–8431, 2022.

[VGO+20] Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature methods*, 17(3):261–272, 2020.

[WFRT07] Aaron Wilson, Alan Fern, Soumya Ray, and Prasad Tadepalli. Multi-task reinforcement learning: a hierarchical bayesian approach. In *Proceedings of the 24th international conference on Machine learning*, pages 1015–1022, 2007.

[Wil71] Jan Willems. Least squares stationary optimal control and the algebraic riccati equation. *IEEE Transactions on automatic control*, 16(6):621–634, 1971.

[WOS+03] Tsachy Weissman, Erik Ordentlich, Gadiel Seroussi, Sergio Verdu, and Marcelo J Weinberger. Inequalities for the l1 deviation of the empirical distribution. *Hewlett-Packard Labs, Tech. Rep*, 2003.

[YQH+20] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pages 1094–1100. PMLR, 2020.

[ZLJZ23] Zhuangdi Zhu, Kaixiang Lin, Anil K Jain, and Jiayu Zhou. Transfer learning in deep reinforcement learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[ZSKP20] Amy Zhang, Shagun Sodhani, Khimya Khetarpal, and Joelle Pineau. Learning robust state abstractions for hidden-parameter block mdps. In *International Conference on Learning Representations*, 2020.

[ZSP18] Amy Zhang, Harsh Satija, and Joelle Pineau. Decoupling dynamics and reward for transfer learning. *arXiv preprint arXiv:1804.10689*, 2018.

Appendix

A Theoretical Analysis of MLEMTRL

In this section, we further justify the use of our framework by deriving worst-case performance degradation bounds relative to the optimal controller. The performance loss is shown to be related to the realisability of μ^* under $\mathcal{C}(\mathcal{M}_s)$. In Figure 1, we visualise the model dissimilarities, where $||\mu - \hat{\mu}||_1$ is the model estimation error, $||\mu^* - \mu||_1$ is the realisability gap and $||\mu^* - \hat{\mu}||_1$ the total deviation of the estimated model. Note that by the norm on MDP, we always refer to the L_1 norm over transition matrices.

Theorem A.1 (Performance Gap for Non-Realisable Models)**.** *Let $\mu^* = (\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}^*, \gamma)$ be the true underlying MDP. Further, let $\mu = (\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}, \gamma)$ be the maximum log-likelihood*

$$\mu \in \arg\max_{\mu' \in \mathcal{C}(\mathcal{M}_s)} \; -\log \mathbb{P}(D_\infty \mid \mu'), D_\infty \sim \mu^*$$

and $\hat{\mu} = (\mathcal{S}, \mathcal{A}, \mathcal{R}, \hat{\mathcal{T}}, \gamma)$ be a maximum log-likelihood estimator of μ. In addition, let $\pi^*, \pi, \hat{\pi}$ be the optimal policies for the respective MDPs. Then, if $\mathcal{R}$ is a bounded reward function $\forall_{(s,a)}\, r(s,a) \in [0,1]$ and with ϵ_{Estim} being the estimation error and

$$\epsilon_{\text{Realise}} \triangleq \min_{\mu \in \mathcal{C}(\mathcal{M}_s)} \|\mu^* - \mu\|_1$$

the realisability gap. Then, the performance gap is given by,

$$||V^*_{\mu^*} - V^{\hat{\pi}}_{\mu^*}||_\infty \leq \frac{3(\epsilon_{\text{Estim}} + \epsilon_{\text{Realise}})}{(1-\gamma)^2}. \tag{6}$$

For the full proof, see Appendix A.1. This result is comparable to recent results such as [ZSKP20] but here with an explicit decomposition into model estimation error and realisability gap terms.

Remark 1 (Bound on L_1 Norm Difference in the Realisable Setting)**.** *It is known [SL05, AJO08, QFPL20] that in the realisable setting, it is possible to bound the model estimation error term ϵ_{Estim} via the following argument. Let μ^* be the true underlying MDP, and $\hat{\mu}$ be an MLE estimate of μ^*, as defined in Theorem A.1. If $\mathcal{R}$ is a bounded reward function, i.e. $r(s,a) \in [0,1], \forall (s,a)$, and ϵ_{Estim} is upper bound on the L_1 norm between $\mathcal{T}^*$ and $\hat{\mathcal{T}}$. If $n^{s,a}$ be the number of times (s,a) occur together, then with probability $1 - SA\delta$,*

$$||\mathcal{T}^* - \hat{\mathcal{T}}||_1 \leq \epsilon_{\text{Estim}} \leq \sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} \sqrt{\frac{2 \log\left((2^S - 2)/\delta\right)}{n^{s,a}}}.$$

From this, it can be said that the total L_1 norm then scales on the order of $\mathcal{O}(SA\sqrt{S + \log(1/\delta)}/\sqrt{T})$.

This result is specific to tabular MDPs. In tabular MDPs, the maximum likelihood estimate coincides with the empirical mean model. Further details are in Appendix A.2.

Remark 2 (Performance Gap in the Realisable Setting)**.** *A trivial worst-case bound for the realisable case (Section 3) can be obtained by setting $\epsilon_{\text{Realise}} = 0$ because by definition of the realisable case $\mu^* \in \mathcal{C}(\mathcal{M}_s)$.*

B A Meta-Algorithm for MLEMTRL under Non-realisability

To guarantee good performance even in the non-realisable setting, we can proceed in two steps. First, we can add the target task to the set of source tasks. Second, we can construct a meta-algorithm, combining the model estimated by MLEMTRL and the empirical estimation of the target task. In this section, we propose a meta-algorithm based on the latter. We illustrate it in Algorithm 2.

The main change in Algorithm 2 compared to Algorithm 1 is internally keeping track of the empirical model, and in Line 8, computing a posterior probability distribution over the respective models by weighting the two likelihoods together with their respective priors. The resulting algorithm then

Algorithm 2 Meta-MLEMTRL

Input: prior p, weights $\boldsymbol{w}^0$, m source MDPs $\mathcal{M}_s$, data D_0, discount factor γ, iterations T.
for $t = 0, \ldots, T$ **do**
  // STAGE 1: OBTAIN MODEL WEIGHTS //
  $\boldsymbol{w}^{t+1} \leftarrow \text{MLEMTRL}(\boldsymbol{w}^t, \mathcal{M}_s, \mathcal{D}_t, \gamma, 1)$
  Estimate the MDP: $\mu^{t+1} = \sum_{i=1}^{m} w_i \mu_i$
  Compute log-likelihood $\ell^{t+1}_{\text{MLEM}} = \log \mathbb{P}(\mathcal{D}_t \,|\, \mu^{t+1})$
  Compute log-likelihood of empirical model $\ell^{t+1}_{\text{Empirical}} = \log \mathbb{P}(\mathcal{D}_t \,|\, \hat{\mu}^{t+1})$
  Sample $\tilde{\mu}^{t+1}$ as μ^{t+1} w.p. $\propto p \exp\left(\ell^{t+1}_{\text{MLEM}}\right)$ and $\hat{\mu}^{t+1}$ w.p. $\propto (1-p) \exp\left(\ell^{t+1}_{\text{Empirical}}\right)$.
  // STAGE 2: MODEL-BASED PLANNING //
  Compute the policy: $\pi^{t+1} \in \arg\max_{\pi} V^{\pi}_{\tilde{\mu}^{t+1}}$
  // CONTROL //
  Observe $s_{t+1}, r_{t+1} \sim \mu^*(s_t, a_t), a_t \sim \pi^{t+1}(s_t)$
  Update the dataset $D_{t+1} = D_t \cup \{s_t, a_t, s_{t+1}, r_{t+1}\}$
end for
return An estimated MDP model $\tilde{\mu}^T$ and a policy π^T

trades off its bias to the prior based on the choice of prior hyperparameter p. Since asymptotically the likelihood of the data under the empirical model is greater than the likelihood of the data under the MLEM-estimated model, as more and more data is collected the Meta-MLEMTRL algorithm performs similarly to the optimal planning using the empirical estimate. This is intended behaviour and allows for the algorithm to asymptotically plan optimally even in the non-realisable setting.

We experimentally study the behaviour of Meta-MLEMTRL and its dependence on the prior parameter p in the next section.

C Detailed Proofs

C.1 Proof of Theorem A.1

Proof of Theorem A.1. We begin by introducing the appropriate definitions and lemmas.

Definition 1 (ϵ-homogeneity)**.** *Given two MDPs $\mu_1 = (\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}_1, \gamma)$ and $\mu_2 = (\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}_2, \gamma)$ we say μ_2 is a ϵ-homogenous partition of μ_1 with respect to L_k norm if*

$$\forall a \in \mathcal{A} \quad \Big(\sum_{s' \in \mathcal{S}} \big(\sum_{s \in \mathcal{S}} \mathcal{T}_1(s, a, s') - \mathcal{T}_2(s, a, s') \big)^k \Big)^{\frac{1}{k}} \leq \epsilon. \tag{7}$$

Note that this definition of ϵ-homogeneity is a special case of Definition 3 [EDM03], where the reward functions and state spaces are taken to be identical. We will only study partitions with respect to the L_1 norm between transition probabilities.

Lemma C.1 (Lemma 3 [EDM03])**.** *Let μ_2 be an ϵ-homogenous partition of μ_1, then, with respect to L_1 norm, an arbitrary policy π in μ_2 induces an $\frac{\epsilon}{(1-\gamma)^2}-$optimal policy in μ_1.*

$$||V^{\pi}_{\mu_1} - V^{\pi}_{\mu_2}||_{\infty} \leq \frac{\epsilon}{(1-\gamma)^2}. \tag{8}$$

Lemma C.2 (Lemma 4 [EDM03])**.** *Let μ_2 be an ϵ-homogenous partition of μ_1, then, with respect to L_1 norm, the optimal policy in μ_2 induces an $\frac{2\epsilon}{(1-\gamma)^2}-$optimal policy in μ_1.*

$$||V^{*}_{\mu_1} - V^{*}_{\mu_2}||_{\infty} \leq \frac{2\epsilon}{(1-\gamma)^2}. \tag{9}$$

Given the assumptions in Theorem A.1 hold true. Then, using the $\epsilon-$homogeneity definition in Definition 1, let $\hat{\mu}$ be an $\epsilon_{\text{Estim}}-$homogenous partition of μ and μ be an $\epsilon_{\text{Realise}}-$homogenous

partition of μ^*. Under the L_1 norm then, we have

$$\forall a \in \mathcal{A} \quad \Big(\sum_{s' \in \mathcal{S}} \sum_{s \in \mathcal{S}} \mathcal{T}(s,a,s') - \hat{\mathcal{T}}(s,a,s') \Big) \leq \epsilon_{\text{Estim}}, \tag{10}$$

$$\Big(\sum_{s' \in \mathcal{S}} \sum_{s \in \mathcal{S}} \mathcal{T}^*(s,a,s') - \mathcal{T}(s,a,s') \Big) \leq \epsilon_{\text{Realise}}. \tag{11}$$

Using triangle inequalities we can then bound the L_1 norm between the true underlying MDP and the maximum likelihood estimator,

$$\forall a \in \mathcal{A} \quad \Big(\sum_{s' \in \mathcal{S}} \sum_{s \in \mathcal{S}} \mathcal{T}^*(s,a,s') - \hat{\mathcal{T}}(s,a,s') \Big) \tag{12}$$

$$= \Big(\sum_{s' \in \mathcal{S}} \sum_{s \in \mathcal{S}} \mathcal{T}^*(s,a,s') - \mathcal{T}(s,a,s') + \mathcal{T}(s,a,s') - \hat{\mathcal{T}}(s,a,s') \Big) \tag{13}$$

$$\leq \Big(\sum_{s' \in \mathcal{S}} \sum_{s \in \mathcal{S}} \mathcal{T}^*(s,a,s') - \mathcal{T}(s,a,s') \Big) + \Big(\sum_{s' \in \mathcal{S}} \sum_{s \in \mathcal{S}} \mathcal{T}(s,a,s') - \hat{\mathcal{T}}(s,a,s') \Big) \tag{14}$$

$$\leq \epsilon_{\text{Estim}} + \epsilon_{\text{Realise}}. \tag{15}$$

Thus, $\hat{\mu}$ is a $(\epsilon_{\text{Estim}} + \epsilon_{\text{Realise}})-$homogenous partition of μ^*. The next steps involves creating a bound on the performance gap between the value functions and policies in $||V^*_{\mu^*} - V^{\hat{\pi}}_{\mu^*}||_\infty$. Using triangle inequalities the performance gap can be expanded further,

$$\begin{aligned} ||V^*_{\mu^*} - V^{\hat{\pi}}_{\mu^*}||_\infty &= ||V^*_{\mu^*} - V^{\hat{\pi}}_{\hat{\mu}} + V^{\hat{\pi}}_{\hat{\mu}} - V^{\hat{\pi}}_{\mu^*}||_\infty \\ &\leq ||V^*_{\mu^*} - V^{\hat{\pi}}_{\hat{\mu}}||_\infty + ||V^{\hat{\pi}}_{\hat{\mu}} - V^{\hat{\pi}}_{\mu^*}||_\infty. \end{aligned} \tag{16}$$

The first term on the right side of the inequality in Eq. 16 can be bounded using Lemma C.2 since $\hat{\pi}$ is the optimal policy in $\hat{\mu}$,

$$||V^*_{\mu^*} - V^{\hat{\pi}}_{\hat{\mu}}||_\infty \leq \frac{2(\epsilon_{\text{Estim}} + \epsilon_{\text{Realise}})}{(1-\gamma)^2}. \tag{17}$$

Likewise, the second term in the inequality can be bounded using Lemma C.1,

$$||V^{\hat{\pi}}_{\hat{\mu}} - V^{\hat{\pi}}_{\mu^*}||_\infty \leq \frac{\epsilon_{\text{Estim}} + \epsilon_{\text{Realise}}}{(1-\gamma)^2}. \tag{18}$$

Combining these two terms yields us $||V^*_{\mu^*} - V^{\hat{\pi}}_{\mu^*}||_\infty \leq \frac{3(\epsilon_{\text{Estim}} + \epsilon_{\text{Realise}})}{(1-\gamma)^2}$. □

C.2 Proof of Remark 1

Proof of Remark 1. The analysis of the concentration of ϵ_{Estim} follows the works of [AJO08, QFPL20]. Let $\Delta^{\mathcal{S}}$ be the $(\mathcal{S}-1)-$dimensional simplex and $\mathcal{T}^*_{s,a} \in \Delta^{\mathcal{S}}$ be the transition kernel for a state-action pair of the true underlying MDP μ^* and $\hat{\mathcal{T}}_{s,a} \in \Delta^{\mathcal{S}}$ a random vector. If $\hat{\mathcal{T}}_{s,a}$ is taken to be the empirical estimate of $\mathcal{T}^*_{s,a}$ then the following lemma can be invoked.

Proposition 1 ([WOS+03])**.** *Let $\mathcal{T}^*_{s,a} \in \Delta^{\mathcal{S}}$ and $\hat{\mathcal{T}}_{s,a} \sim \frac{1}{n^{s,a}}\text{Multinomial}(n^{s,a}, \mathcal{T}^*_{s,a})$. Then, for $S \geq 2, \delta \in [0,1]$ and $n^{s,a} \geq 1$,*

$$\begin{aligned} &\mathbb{P}\left(||\mathcal{T}^*_{s,a} - \hat{\mathcal{T}}_{s,a}||_1 \geq \sqrt{\frac{2S\log(2/\delta)}{n^{s,a}}} \right) \\ \leq &\mathbb{P}\left(||\mathcal{T}^*_{s,a} - \hat{\mathcal{T}}_{s,a}||_1 \geq \sqrt{\frac{2\log\left((2^S-2)/\delta)\right)}{n^{s,a}}} \right) \leq \delta. \end{aligned} \tag{19}$$

The invocation of Proposition 1, with $T = \sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}} n^{s,a}$ yields us a L_1 norm bound for the difference of transition kernels associated with a particular state-action pair s, a. Next, union bounding over all possible state and action combinations yields us a bound on the total L_1 norm.

$$\mathbb{P}\left(\bigcup_{s\in\mathcal{S}}\bigcup_{a\in\mathcal{A}}\left(||\mathcal{T}^*_{s,a} - \hat{\mathcal{T}}_{s,a}||_1 \geq \sqrt{\frac{2\log\left((2^S-2)/\delta)\right)}{n^{s,a}}}\right)\right) \tag{20}$$

$$\leq \sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}}\mathbb{P}\left(||\mathcal{T}^*_{s,a} - \hat{\mathcal{T}}_{s,a}||_1 \geq \sqrt{\frac{2\log\left((2^S-2)/\delta)\right)}{n^{s,a}}}\right) \tag{21}$$

$$\leq SA\delta. \tag{22}$$

From this, we have that, with probability $1 - SA\delta$,

$$||\mathcal{T}^* - \hat{\mathcal{T}}||_1 \leq \epsilon_{\text{Estim}} \leq \sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}}\sqrt{\frac{2\log\left((2^S-2)/\delta)\right)}{n^{s,a}}}. \tag{23}$$

The total L_1 norm then scales on the order of $\mathcal{O}(SA\sqrt{S-\log(\delta)}/\sqrt{T})$, which is the final result. □

D Details of Planning: RICCATIITERATION

An LQR-based control system is defined by its system matrices [Kal60]. Let d_s be the state dimensionality and d_a be the action dimensionality. Then, $\mathbf{A} \in \mathbb{R}^{d_s} \times \mathbb{R}^{d_s}$ is a matrix describing state associated state transitions. $\mathbf{B} \in \mathbb{R}^{d_s} \times \mathbb{R}^{d_a}$ is a matrix describing control associated state transitions. The final two system matrices are cost related with $\mathbf{Q} \in \mathbb{R}^{d_s} \times \mathbb{R}^{d_s}$ being a positive definite cost matrix of states and $\mathbf{R} \in \mathbb{R}^{d_a} \times \mathbb{R}^{d_a}$ a positive definite cost matrix of control inputs. The transition model described under this model is given by,

$$\boldsymbol{s}_{t+1} - \boldsymbol{s}_t = \mathbf{A}\boldsymbol{s}_t + \mathbf{B}\boldsymbol{a}_t. \tag{24}$$

When an MDP is mentioned in the context of an LQR system in this work, the MDP is the set of system matrices. Further, the cost (or reward) of a policy π under an MDP μ is

$$V^\pi_\mu = \sum_{t=0}^{T} \boldsymbol{s}_t^\top \mathbf{Q}\boldsymbol{s}_t + \boldsymbol{a}_t^\top \mathbf{R}\boldsymbol{a}_t. \tag{25}$$

Optimal policy identification can be accomplished using [Wil71]. It begins by solving for the cost-to-go matrix $\mathbf{P}$ by,

$$\underset{\mathbf{P}}{\text{solve}} \quad \mathbf{A}^\top\mathbf{P}\mathbf{A} - \mathbf{P} + \mathbf{Q} - (\mathbf{A}^\top\mathbf{P}\mathbf{B})(\mathbf{R} + \mathbf{B}^\top\mathbf{P}\mathbf{B})^{-1}(\mathbf{B}^\top\mathbf{P}\mathbf{A}) = 0.$$

Then, using $\mathbf{P}$ the control input $\boldsymbol{a}$ for a particular state $\boldsymbol{s}$ is

$$\boldsymbol{a} = -(\mathbf{R} + \mathbf{B}^\top\mathbf{P}\mathbf{B})^{-1}(\mathbf{B}^\top\mathbf{P}\mathbf{A})\boldsymbol{s}. \tag{26}$$

With some abuse of notation and for compactness, we allow ourselves to write $\boldsymbol{a}_t = -(\mathbf{R} + \mathbf{B}^\top\mathbf{P}\mathbf{B})^{-1}(\mathbf{B}^\top\mathbf{P}\mathbf{A})\boldsymbol{s}_t$ for $\boldsymbol{a}_t \sim \pi(\boldsymbol{s}_t)$.

E Additional Experimental Analysis

E.1 Experimental Setup

The experiments are deployed in PYTHON 3.7, with support from SCIPY [VGO+20], STABLE-BASELINES3 [RHG+21] and ran on a i5-4690k CPU and a GTX-960 GPU. The parameters for the variations of SAC and PPO are kept to be the default ones.

RL Environments: Chain. A common testbed for RL algorithms in tabular settings is the Chain [DFR98] environment. In it, there is a chain of states where the agent can either walk forward or backward. At the end of the chain, there is a state yielding the highest rewards. At every step, there is a chance of the effect of the opposite action occurring. This is denoted as the slipping probability. The slippage parameter is also what is used to create the source models, in this case, those parameters are $\{0.01, 0.20, 0.50\}$. For PSRL and MLEMTRL we use a product-NormalGamma prior over the reward functions. For PSRL, we use product-Dirichlet priors over the transition matrix.

RL Environments: LQR Tasks. We investigate two LQR tasks in the *Deepmind Control Suite* [TDM+18], namely *dm_LQR_2_1* and *dm_LQR_6_2*. These environments are continuous state and actions whereby the task is to control a two joint one actuator and six joint two actuators towards the center of the platform for the two tasks, respectively. They consist of unbounded control inputs and rewards with the state spaces $s \in \mathbb{R}^4$ and $s \in \mathbb{R}^{12}$, respectively. In the Deepmind Control suite every task is made to be different by varying the seed at creation. The seed determines the stiffness of the joints.

RL Environments: CartPole. We also conduct some experiments on the CartPole [BSA83] environment. In this case, we use a continuous control version of it and formulate it as a LQR problem. The environment has a single continuous action and a state space $s \in \mathbb{R}^4$. To create different tasks we vary the environmental parameters of the problem, namely the gravity, mass of cart, mass of pole the length of the pole.

E.2 Impacts of Realisability

In the experiment depicted in Figure 5, we investigate the convergence rate and the jumpstart improvement of the MLEMTRL algorithm on 100 independent target MDP realisations at six different levels of divergence. The divergence is measured from the centroid of the convex hull to the target MDP. Further, in the topmost row, all of the target MDPs belong to the convex hull of source models.

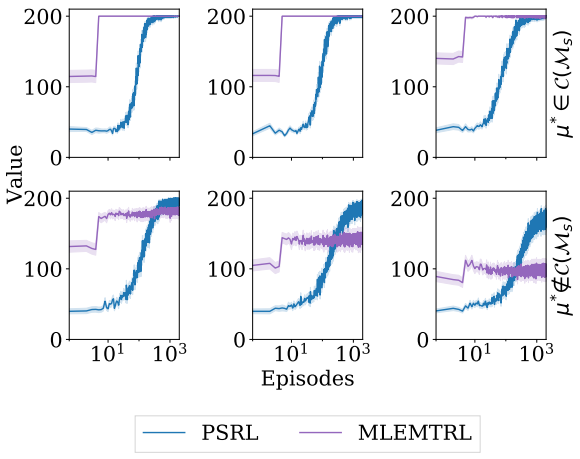

Figure 5: We compare MLEMTRL against PSRL in terms of convergence speed and early performance. The value functions of the algorithms are plotted against the log of the number of episodes. The shaded area is the 68% confidence of the mean over multiple MDPs with the same model dissimilarity. In the topmost row, all of the true MDPs are within the convex hull $\mathcal{C}(\mathcal{M}_s)$. In the bottom row, the MDPs are outside. As you go from top-left to bottom-right, the divergence from the true model to the average model in the convex hull increases. For utility, higher values are better.

As we can see, in this setting, identification of the true model occurs rapidly. One reason for this is because of the near-determinism of the environment. Compared to the agent learning from scratch, we observe zero-regret with faster convergence. As we go from top-left to bottom-right, the divergence increases. For the bottom-most row, we can again observe a faster learning rate. In this case, the degradation in performance increases with the divergence, resulting in poor performance in the final case. The experiment demonstrates that under the TRL framework, we require that the source models are not too dissimilar from the target model.

E.3 Impacts of Multi-Task Learning as a Baseline

In Figure 6 we investigate the performance increase of using multi-task RL and transfer RL compared to regular RL. The baseline algorithm is Soft Actor-Critic and its associated multi-task and transfer learning formulations. As we can see, **MT-SAC-TRL** appears to have overall strongest performance, with **MT-SAC** a close second. Because of the nature of the problem (unbounded negative rewards), it is also possible for the algorithms to diverge during learning, which further strengthens the argument for using multi-task or transfer learning for robustness.

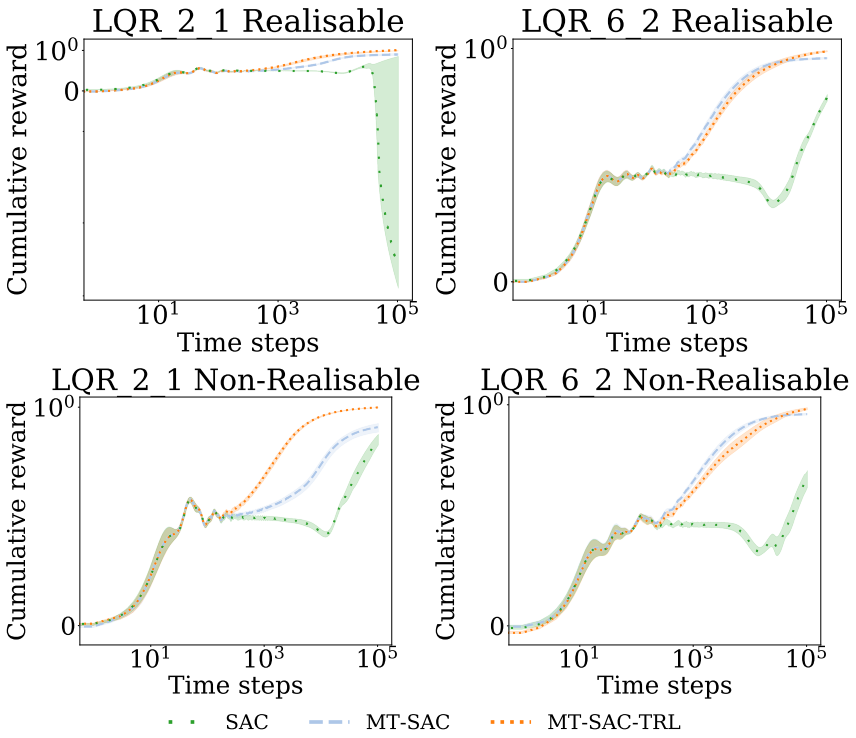

Figure 6: Figure depicting the performance boost of using multi-task and transfer reinforcement learning compared to standard reinforcement learning. The y-axis represents average cumulative reward at every time step and the shaded region is the standard error.

E.4 Model-based Transfer Reinforcement Learning with Known Reward Function

In Figure 7, we aim to contrast the difference from the figure in the main paper where now the reward function is known a priori to **MLEMTRL** and **PSRL**.

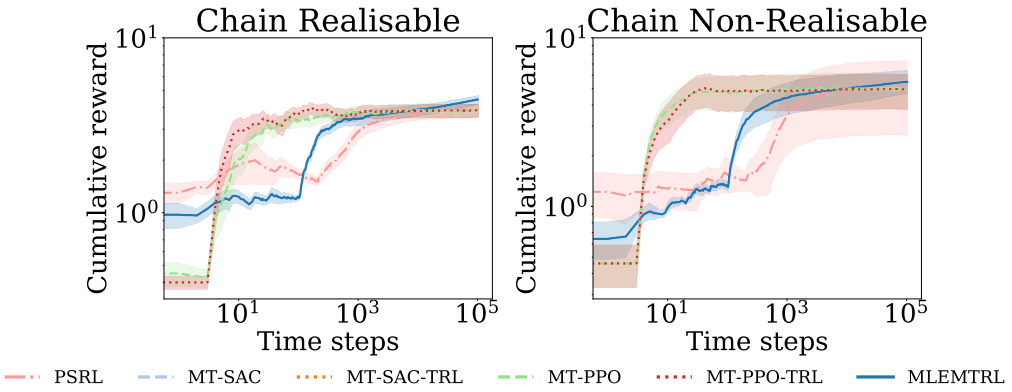

Figure 7: Performance of MLEMTRL, the meta algorithms, and the baselines for the case with known reward function in Chain. The y-axis is the average cumulative reward at every time step and the shaded region is the standard error.