Diverging Emerging Field of Multi-Task Reinforcement Learning

Mudit Verma
Arizona State University
muditverma@asu.edu

ABSTRACT

Multi-Task Reinforcement Learning or MTRL is an emerging field that is gaining tremendous attention from researches since the use of Deep Reinforcement Learning. In this report, we will motivate the need for Multi-Task Reinforcement Learning & see how modern researches are different in many aspects. We will formalize the notion of a task in an attempt to unify formal definitions of different works & provide many domains that have been used in several notable ones. We will answer some questions about the MTRL field, like knowledge about the task to the agent, works using model-free and model-based settings, online and offline model updates & how reuse of experience has been adapted for MTRL. We will then discuss three notable works that differ in approach & the definition of the task and alongside visit some other relevant literature in brief. We will cover Meta-Learning, Hierarchical RL & Adoption of special normalization function with a distributed RL setting as our representative selection of works to review. We will then highlight some of the important challenges faced by MTRL algorithms like Task Interference, Distraction Dilemma and Scalability & how various works have addressed them. We will conclude our report & note some possible future directions.

KEYWORDS

Multi-Task Reinforcement Learning, Markov Decision Processes, Meta-Learning, Hierarchical Reinforcement Learning

1 INTRODUCTION

Reinforcement Learning has enjoyed a lot of success in recent years, surpassing human limits in a variety of domains, especially strategy games like Chess and Go. However, humans are still the better multi-task agents, performing hundreds of tasks each day which may be as similar as achieving the same goal again and again (like opening several doors in a day) or completely different (like cooking & driving a car) or even perform tasks which have conflicting goals (like opening a door & closing it). Another area at which the humans are better is sample efficiency. It is well known that humans can learn manipulation tasks relatively easily than their robot counterparts. These are some of the many motivations recent researches have used to contribute in the field of Multi-Task Reinforcement Learning or MTRL.

[33] is an in-depth survey of Multi-Task Learning (& not specific to RL). It does talk about how multi-task learning can behave or be improved when it is used with other paradigms, like RL, which is our concern. It has a short section on MTRL and lists out various works which we skip for brevity. [26] is a more specific survey over the intersection of Multi-Task RL and Transfer Learning and presents many ideas that are still used but with a flavor of Deep Learning. However, this survey does shed light on some of the important issues that one can come across in MTRL, a prominent one being “negative transfer" or “task interference" in some other literature. These and some other issues are discussed in 7.

Although [26] is an excellent resource for a variety of reasons, it does not cover the MTRL works in the era of Deep Learning. Since the beginning of the Deep Reinforcement Learning, many previous works have been modified to work with MTRL, whereas, many new methods have been developed too. Deep RL has been extensively used in the past years for image-based environments for end-to-end agent learning, and hence, it becomes an interesting topic to study. The goal of this report is to explore a variety of Deep Learning (or gradient-based) based Reinforcement Learning methods maximizing a Multi-Task objective from different lenses.

We will begin by answering one of the first questions - Why can’t we use multiple single-task agents? A simple answer is, we can, but we can also do better, usually by exploiting these task structures. We will then revisit some background to Reinforcement Learning, mostly to have a consistent notation across the report. The next obvious question, is what kinds of tasks do we keep referring to in the MTRL? We answer this in Section 4. Since the literature on this topic is vast - covering many directions - hence the term “diverging", we will focus ourselves along certain lines of thoughts in Section 5 and have a deeper look at a few representative recent works like Meta-Learning, Hierarchical Reinforcement Learning & even a method which generates a single policy for multiple tasks in 6. Now, although MTRL brings a lot of benefits like generality & lesser learning parameters, to name a few, we will try to cover some of the more important challenges to MTRL in Section 7. Finally, we will end this report with some future directions to MTRL & conclude in Section 8.

2 WHY NOT MULTIPLE SINGLE-TASK AGENTS?

Some readers might reject the importance of Multi-Task learning by assuming multiple single-task specific agents. Reinforcement Learning (RL), for a long time, has focused on improving single task performance & so this question is valid in the sense that we hope RL algorithms are fast, sample-efficient & general, but by no surprise conventional single task methods struggle with all of these. Of course, the use of single task agents, one for perfecting on each task, is a possible solution, but not a viable one—first reason being the speed of improving the agent's policy for each task. The original AlphaGo agent [21] is considered to be one of the famous milestones in modern-day Reinforcement Learning, but it is also popular for the time it took to train the agent. Not only the time but also the number of examples (close to 30 million), which brings us to the topic of sample efficiency, something which RL algorithms strive to improve to date. Finally, generality is another issue & a change in the environment reward function can make the agent fail miserably. Not only the reward, but change in the state transition function can also negatively affect the performance. These were some issues in training an agent for one task; the ultimate goal, however, was to train several agents for their respective tasks. A thing to note at this point is, AlphaGo [21] was a high rated Go Player, which means it could essentially adapt in some sense when it played against players using different strategies. This gives some hint towards the agent being general to some extent or rather, solving for different tasks. This is what motivates much of the ongoing research in the field of Multi-Task RL; the similarity between tasks is something that can be exploited.

3 BACKGROUND

Reinforcement learning has long been used for sequential decision making under uncertainty. For notational convenience & reconciliation among different works, let us define a few terms. A learning *agent* is required to learn how to interact with an *environment* $\boldsymbol{E}$ in discrete time steps $t \in \{0, 1, 2..\}$. Some works try to handle infinite timesteps, but we will restrict ourselves to finite possible values of t. A Reinforcement Learning (RL) problem is often formalized as a Markov Decision Process [5].

We can define an MDP as a tuple (S, A, p, R, γ), in some works as $(S, A, p, R, \gamma, d_o)$ where S is the set of valid states & A is the set of all possible actions. p is the transition function $SxA \longrightarrow S$, or the dynamics of the environment. d_o is the initial state distribution. When action $a_t \in A$ is taken in state $s_t \in S$ the state transitions to state $s_{t+1} \in S$ and a reward r_{t+1} is obtained. Thought there are subtle variations in expressing how the reward is modelled. The agent's goal, in a single-task setting is to follow a policy π, which is a mapping $S \longrightarrow A$ can be deterministic or a distribution over action, to maximize the expected return $G = \Sigma_{t=t_1}^{t=t_n} \gamma^{t-t_1} r_t$. We can further define, *action values* and *state values* as $q^{\pi}(s, a) = \mathbb{E}_{\pi}[G_t | S_t = s, A_t = a]$ and $v^{\pi} = \mathbb{E}_{\pi}[G_t | S_t = s]$.

As a general definition, works treat each task as a separate MDP tuple, which may have different dynamics & tend to exploit similarities in the substructure of the MDP. Hence, we can define tasks as the set $T = \{D_i = (S_i, A_i, p_i, R_i, \gamma_i)_{i=1}^{N}\}$ for N tasks. However, many works sample a task from the distribution of tasks rather than explicitly stating a set of finite tasks. Even in those cases, the tasks are MDPs.

4 TASK & DOMAINS

[25] is one of the first works on the topic in concern & motivates the problem with a simple real-world example, which we would refer here too. The consider the case of a cleaning robot in a hotel. At first glance, this may seem like using a single task agent strategy should be enough; however, in a real scenario, it is unlikely that any two rooms would be left in the same state after the guests leave. This type of problem is also considered by the more recent work on Meta-Learning [12], where they want the agent to reach a goal location in different maze configurations. However, this is definitely not a standard testbed for the MTRL problem.

Section 3 defines a task as an MDP which can share specific parts of the MDP with other tasks. To manifest this, we will consider a few works and list out the tasks and domains they have evaluated on.

The general trend of defining tasks has been to fix a domain & change the goal (which is the reward metrics). [12] uses self-created **2D point navigation**, in which the point agent must navigate to reach a goal location. To stress that their method is well suited for more complex tasks, they also report results on **Locomotion Task** where the task is to move a quadrupedal robot in several directions. [1] also uses a quadrupedal robot called **Cliff Environment** same as one used by [19]. They also use a **Maze Domain** [16] for showing results on different domains. [29] also choose a **colored Maze Environment** for their results; however, they created this by themselves. Similar to the point 2D navigation of [12], but more complex and realistic is **Meta World Domain** [32] used by [30]. The Meta world task is like ImageNet of robotics; it has 10 & 50 Manipulation task challenges. However, the authors in [30] extend the goals like "reaching out" of a robotic arm by generating different locations (hence different tasks). [31] uses a **Pioneer 3AT** robot model simulated on Gazebo; however, the "Task" is for the agent to be able to perform 12 different movement-based control tasks. [20] creates a more realistic domain of **Blockworld** with motion-based tasks for their experiments. [7] report their results on a simple **Gridworld domain** where a task is different locations. Finally, PopArt [14], unlike other works that have predominantly assumed tasks within the shared domain, performs their experiments with different games as their tasks. The agent essentially has to figure out which game it is playing and then solve it. They use the **Atari-57 Games** [4] for their first experiment. They also report results on **DmLab-30** [3] which is created by DeepMind for testing reinforcement learning algorithms. DmLab-30 is similar to Labyrinth & maze environments.

5 COMPARING WORKS

Is the task known?

Section 3 introduces the notion of Task in an MTRL setting. Further, Section 4 describes what these tasks are in terms of the domains & goals used by various works. However, it is not necessary that which task to achieve is known to the agent. [14] is one such work that does not require a task index information for the agent policy at the test time. The tasks are 57 Atari Games; hence they have different goals & different state appearances. It is interesting as, in essence, the task index information is embedded in the state space the agent is in. This, however, does not make sense for other works like [1, 12] etc. since, for them, a task is within a domain & not across domains. Hence, although specific regions of a state-space can still encode task-specific information, one can imagine that it would be a hard problem. In general, therefore, works defining a task within a domain, conditions their policy on a task id. The other set of works that might not require such a conditioning, are model-based scenarios where the environment has been modified to supply only task-specific rewards - then these rewards are indicators of which task the agent is supposed to solve.

Negative Interference

Negative Interference (for Negative Transfer & Task Interference) is, in fact, one of the major issues for MTRL. Section 7 gives a detailed discussion on which methods are robust against this.

Human Inputs

Human Supervision can prove to be an important part in being able to learn a policy for a multi task RL problem. This is clearly shown by [1] & discussed in Section 6. However, not many works have utilized this supervision technique in their methods. [1] lists out some related works like program-interpreters that require human supervision and generates discrete computational structures like sketches. Most other methods in the field do not require such supervision, apart from domain specification, which is implicit. Moreover, none of the methods deal with the problem of interpretability for human in the loop, since they have not assumed a human for any part of the method, as per the surveyed works in this report.

Model Free vs. Model-Based?

One of the primary advocates of model-based learning in the MTRL setting is meta-learning. Since, when a model is present, works like [12] or other single-task learners can be deployed to solve for the [7] is model-based and figures out, by process of elimination, which MDP is the underlying one for the given task out of a set of MDP models. MTRL problem. However, in general & especially for image-based domains, learning a model is a difficult task. Recent attempts like [15] have tried learning a model for Atari games, but even they report that the learned models are not stable.

Moving on, there have been many attempts at using model-free methods for the MTRL problem. Most of the works discussed here attempt to learn either a policy π_i for some task T_i (maybe by learning the whole family of policies $\mathbf{\Pi}$) or they learn a base policy and improve that when w.r.t. task at hand [12]. Model-Free gives an advantage of obtaining the policy directly by using the popular deep neural networks & since interpretability is not considered as an issue by these works; model-free strategy gives decent results.

Online vs. Offline?

As we will see, not many of the methods here are on-policy methods & use policy gradients. Also, many of the algorithms have an (advantage) actor-critic architecture, which is also on-policy. This restricts these methods to reuse any interaction experience. [13] is one such offline technique, whereas others like [1, 12], to name a few, are on-policy & uses policy gradients (although an off-policy derivation is possible with Importance Sampling). All of these methods will be discussed in Section 6.

Reusing Experience

For settings which are off policy (that is the loss gradients are computed from some fixed policy or some other policy) usually, store "experiences" and would sample these experiences as i.i.d later on called as "experience replay". The problem with existing experience replay is that some tasks may have more "experience" in the dataset than others (which can cause the Distraction Dilemma, more in Section 7). [2, 11] propose a possible solution, named, Hindsight Relabelling. The idea is while sampling for state transitions when exploring for say a particular task T_1; then it may so happen that some of the experienced states, which may not be relevant for T_1 are actually relevant for another task T_2. For example a soccer-agent is trying to improve the task of kicking the ball into the goal-post. However, while exploring, many of the transitions were rewarding for the other task of passing the ball. Under usual "experience-replay transitions would have been thrown away since they are not good at the original task, however, in hindsight relabelling, these transitions would be "relabelled" as transitions for the task they are good at, hence balancing & generating data for other tasks as well.

6 NOTABLE METHODS

The previous section refers to various works with respect to a certain property like whether they consider the task label as an input or not, or are is there a way for humans to input some general knowledge about sub-tasks for a goal task. In this section we will look at some popular works which are driving the Multi-Task Learning research in recent times. As a general trend, many works have tried learning a model of the environment to use it for various tasks, which is a hard task. Some other works have tried obtaining an agent which, when trained on a few examples of a novel task, can readily adapt. Other works have tried a different approach altogether. They more explicitly exploit the structure of tasks presented in a hierarchical scheme of sub-tasks or sub-modules. These sub-tasks may be inferred too from experience, or maybe assumed to be given. A third, interesting, line of work addresses two aspects of MTRL, training parallelization & balancing tasks. It is considered by several works that some tasks require less training than others, and therefore this imbalance may cause performance issues. We revisit this issue in Section 7.

Now we will look at these three directions in MTRL.

Meta RL

[12] defines the goal of "meta-learning" as being able to train a model on a variety of tasks, such that it can solve new learning tasks using only a small number of training samples. In effect, theirs is an intelligent way of initializing model parameters, such that it takes only a few updates to get a sufficiently well-performing model. It should be noted that meta-learning is model-agnostic as long as the model parameters can be optimized via gradient-based updates. Although "meta-learning" is a more general class that can be used with several paradigms such as regression models or classification models, they show how this can be used in a Reinforcement Learning setting.

Revisiting Sections 3 and 4, [12] defines a task T_i as an MDP given by tuple $(S_i, A_i, p(T_i), R_i, \gamma, d_o(T_i))$. The aim is to adapt the model to $\mathbb{Z}(T)$ distribution over tasks such that T_i is drawn from $\mathbb{Z}(T)$. Although, they consider Cost instead of the more general Reward structure for the MDP. At each time step, let, f_θ be the policy function that maps states s_t to control (or actions) a_t at each time-step $t \in \{0, 1, 2..\}$.

The original model is given by function f_θ, however, when adapting to a specific task T_i, θ becomes $\theta_i^{'}$. Their meta-objective then is defined as:

$$\min_\theta \Sigma_{T_i \sim \mathbb{Z}(T)} L_{T_i}(f_{\theta_i^{'}})$$

A single gradient update towards making this model f_θ adapt to the new task T_i, we perform one gradient step as :

$$\theta_i^{'} = \theta - \alpha \nabla_\theta L_{T_i}(f_\theta)$$

An easier way to understand the updates is, the meta-optimization (including various tasks) is over θ, whereas the objective is calculated using the updated model parameters $\theta_i^{'}$.

The meta RL scheme thus requires a model to be such that the parameters can be updated via gradient based methods. However, for Reinforcement Learning tasks, the L_{T_i}) function which is a feedback or cost for the MDP, is given by Reward or cost functions, which are usually non-differentiable. Work on Policy Gradients, therefore, come in handy to adapt meta-RL to use expected rewards as follows :

$$L_{T_i}(f_\theta) = -\mathbb{E}_{s_t, a_t \sim f_\theta, p(T_i)}[\Sigma_{t=t_1}^{t=t_N} R_i(s_t, a_t)]$$

Since the work uses policy gradients, they are a on-policy algorithm & require new samples before for every gradient update step. They use multiple multi-step rollouts to sample trajectories using the current policy.

Stemming from this initial work on meta-learning, they also try to leverage the knowledge of what components of MDP is changing across tasks. A simple idea, they mention [1] is to include task identifiers z_i which maybe one-hot encoded versions of task id i in task T_i. Then, they condition the policy and Q values with these task ids as well.

Hierarchical RL

A lot of work has employed Hierarchical Reinforcement Learning for Multi-Task learning problems. We will discuss [1] as the representative work in this field; however, there are other works such as [13] which leverages the options framework [24] or [20] which defines hierarchical policies and selects whether to use a learned policy or to learn a new skill (like exploitation/exploration). We will first discuss [1] and then, for brevity & completeness, briefly visit the idea of [13].

[1] is an interesting work since they employ weak human supervision for MTRL. This is the only work among those reviewed, which used some form of human input for the MTRL setting. The main intuition behind their work is whether informing the learner about the abstract structure of policies, without specifying high-level behaviors use the primitive actions can be useful. Policy Sketches, a term they

[1]https://cs330.stanford.edu/slides/cs330_mtrl.pdf

invent, to name a sequence of these un-grounded high-level behaviors. These symbols can be shared across tasks, like the high-level action of “get wood" that can be reused in both “make plank" and “make stick" tasks. The main benefit of such sketches is that now the shared latent structure similarity between tasks is understood better. As one would expect, their work outperforms other unsupervised methods that do not provide any task-specific guidance. They also compare the sketches with options framework; however, the difference is that unlike options, these sketches do have a formal definition attached to them beforehand. Their method learns the respective sub-policy for each of these symbols in a sketch & execute those, similar to how options work.

Getting into the mechanics, they also define Tasks as MDPs; however, they limit tasks to only differ in reward & initial state distribution. Hence tasks are performed in a shared environment defined by the MDP tuple (S, A, P, R, γ) are as defined before (Section 3). However, each task $T_i \sim T$ is a pair (d_o^i, R_i) where $R_i : S \longrightarrow \mathbb{R}$ is a task-specific reward function & d_o^i is the task-specific initial state distribution. Further they assume that each task T_i has a corresponding policy sketch K_i provided beforehand, which is a sequence $(b_{i1}, b_{i2}, b_{i3}...)$ of high level symbols drawn from vocabulary $\mathcal{B}$. They use a decoupled actor-critic algorithm (decoupling is of the policies from value functions) to solve the MTRL problem.

Now, let $\mathbf{\Pi}$ be family of all task specific policies Π_i such that $\mathbf{\Pi} := \bigcup_i \{\Pi_i\}$. If π represents the base policy, a function on environment state, parameterized by some weight θ, then the Multi Task objective to maximise is,

$$J(\mathbf{\Pi}) = \Sigma_i J(\Pi_i) = \Sigma_i \mathbb{E}_{s_i \sim \Pi_i}[\Sigma_k \gamma^k R_i(s_k)]$$

across tasks $i \in T$.

To maximize this objective they make use of policy gradients, gradient of the objective thus is,

$$\nabla_\theta J(\mathbf{\Pi}) = \Sigma_i \nabla_\theta J(\Pi_i)$$

or

$$\nabla_\theta J(\mathbf{\Pi}) = \Sigma_i \Sigma_k (\nabla_\theta log\pi(a_{ik|s_{ik}}))(q_k - c_i(s_{ik}))$$

Note, k is time-step of policy execution of a task i & function c_i is the advantage. Since c_i depends upon the task identity & since the value function is itself unknown (a usual proxy for advantage term), this c_i is approximated with data. They use another parameterized function approximator and minimize the mean squared error with gradient descent. As can be noted, like the meta-learning in previous Section 6, this is also an on-policy approach and the family of policies $\mathbf{\Pi}$ is used to perform rollouts and save the transitions as tuples (states, primitive-actions, sketch-symbols, rewards, task-id). These stored transitions are then used to compute the actor and critic gradients.

As a sideline improvement, for performing better over complex tasks, they present a notion of curriculum training where they try to focus the learner to learn “easy tasks" that give high rewards from rollouts in the hope to learn optimal sub-policy behaviors & then bootstrap these to solve for more complex tasks which can have sparse rewards.

Their results show the work perform much better than a baseline option-critic. An impressive result was the method’s ability to perform well on a zero-shot task (performance on a novel task, given its sketch, without any interaction) & adaptation-task (performance on a novel task, with no sketch, with few interactions). Finally, they also perform an ablation, testing how their model behaves without task identity information and highlight the importance of informing MTRL models of which the models are solving for.

For completeness, we will briefly cover [13], which leverages the reusability of temporally extended actions (or in a form, options) to improve upon the MTRL baselines. They formulate a set of options, initially comprising of only primitive actions and incrementally introduce one option at a time. If the new option improves its objective, then it is kept, and the algorithm continues; otherwise, the algorithm returns the found options. The objective depends upon three sets of parameters, θ for base policy, ϕ for option policy & ψ for option stop function. The algorithm, therefore, initializes an option based on these, optimizes these parameters through gradient updates for some number of iterations over the objective & continues the main loop of option discovery until their objective keeps on improving. Among many improvements reported, a notable disadvantage of the work is that the policy obtained is very sensitive to the set hyperparameters.

PopArt & Impala

PopArt [14] and Impala [10] are works which, unlike the rest of the discussed works, do not contribute much in the form of RL setting or does not give a new way of learning but rather does something very interesting. IMPALA [10] is an agent by DeepMind whose strength is preforming actor-critic updates in a distributed setting. To give context, in an actor-critic setting, the actor learns a policy function from states to distribution over action & the critic learns the state value function - both of these perform an interplay while calculating the loss. For more details, readers can refer to [10], which gives a brief over A2C. The usual A2C run is on-policy, that is, say there are multiple actors and one learner, each of the actors will copy the learner policy model & rollout to gain their set of experiences. These experiences, after all actors have performed their rollouts, will be used to update the learner model. However, in a distributed setting as say one of the actors A run its rollout, it is possible that the learner has been updated because some other actor B finished its task, now the rollout that this actor A is performing will give stale information because it is based on an old policy model that was copied when the rollout began. IMPALA tackles exactly this problem via its v-trace correction. The details are out of the scope of the global context here, so we will skip to the part & discuss how it is used. (However, it was necessary to know that the actors are working independently & stale information can be corrected).

PopArt [14] uses the IMPALA architecture, and for a multi-task setting, assumes that each actor has been assigned a task T_i to perform rollouts. Hence they have a scalable, distributed multi-task agent. However, it's not this simple. Using IMPALA has its disadvantages - for a multi-task setting - the updates are dependent upon the rewards in multiple ways. In their setting, they assume that the transition dynamics and action space are shared among tasks, and each task has its own reward function (in RL settings goals are usually inferred from reward setup, unlike in planning where goals can be explicit.). The state-space of MDP is defined as $S = \{\{(s_j, i)\}_{s_j \in S_i}\}_{i=1}^{N}$, where i is the task index as per our notation throughout the report. It should be noted here that the task index is supplied at the training time; however, at the test time, PopArt is not conditioned on a task index. The algorithm returns one policy $\pi(A|S)$ over this state-space & since the task has to be inferred at the test time, this becomes a much more difficult task than others. Section 5 covered few thoughts on this.

PopArt leverages the popart normalization function introduced in [28] (which was for value functions) to modify the value function term in the v-trace equation of IMPALA. They claim that the outputs of the normalization (the first and second-order moments) are non-stationary & hence perform another trick to handle this. Finally, since the global problem is to solve the multi-task problem & these were for single tasks, they rewrite the equations to handle vector versions of the value function & weights, where each index i is for the corresponding task T_i. They note that in the updates, only the value function and not the policy function is conditioned on task index i. Now, since these values are only needed for obtaining a better policy function in the actor-critic arrangement, these can be thrown away later on & only the policy function can be used without task index.

The method shows that they are able to outperform single task DQN expert agents while using similar frame consumption. Again, to emphasize their results, they do not take into account the task indices. Here each task is a different Atari Game from Atari-57 [4] or DmLab30 [3].

7 CHALLENGES TO MTRL

Multi-Task RL has many benefits, as we have seen in the previous sections. However, it's no free lunch, and so it has some issues of its own. The idea of optimizing a policy over multiple tasks together, obtaining a model, or the other methods we looked at, all exploited the structure between tasks. However, if the similarity between tasks is a benefit, its duel, the differences between tasks can pose robustness issues. Further, its possible that even similar tasks but with different importance can harm each other, this can be thought of like survival of the fittest (tasks compete to get better upon the shared resource of trainable parameters) or simply a case of mode collapse, where more dominant tasks would thrive, and other minor ones will suffer. These are the two major issues with MTRL; however, we will also look at another issue of Scalability & data efficiency that different works have addressed.

Task Interference

Task Interference, as hinted before, is the problem of having conflicting tasks, or part of these tasks. [26] also notes this problem and calls it as "negative transfer" as they are interested in transfer learning methods for MTRL. [27] notes the problem as jointly optimizing for several tasks can negatively impact the single-task performance for many algorithms that use shared parameters or transfer learning. They curb this by their Distral Algorithm, which basically tries to have agents specific to each task, and a shared policy for all the agents. Then the goal for agents is to become experts in their assigned tasks but at the same time attempt to be as close to the "distilled" shared policy.

A more recent work [30] uses a subtle intuition that one should not naively use all the shared weights for all the tasks, and rather they create a routing module & then route the weights of the base policy network according to the task in consideration & perform a soft combination over all these routes to obtain the final task-specific policy. This work is of interest as they visualize the t-SNE of probabilities for each task from the routing network, which comes out to be like a neat Voronoi diagram indicating the effectiveness of their routing module. Moreover, they note that tasks that are opposite in nature like "drawer open" and "drawer close" lie at the opposite ends of the t-SNE, whereas unrelated tasks lie far away. A similar idea, w.r.t . the routing module is by [6], which uses an attention mechanism to group tasks into sub-networks on a state level & avoids negative transfer wherever possible.

[7] essentially guarantees that no negative transfer occurs, with other claims about improving the sample efficiency. The idea here is similar to the meta-learning idea of [12], in that they also strive to improve for one task & then accelerate towards other tasks; however, the technique and approach is completely different. They define a "exploration-exploitation" trade-off in optimizing over different MDPs and show that their method is similar to the E^3 algorithm, which does not exhibit negative transfer.

[8] is one work that tries to address this issue. They still define tasks as MDPs but also assume that these tasks can be represented by a shared linear approximation of some subset of features. First, they point that negative transfer is an issue for a Group-LASSO fitted-Q Algorithm, especially when these tasks cannot be represented as a linear approximation. Then they define a straight-forward “trace-norm" or the feature matrix & use this as a regularizer in their loss function and note that negative transfer is mitigated, but not removed. This is one of the few works that present excellent numerical analysis.

There are other works that note this issue & attempt to solve it; however, we will list a few more for completeness but refrain from expanding onto those for brevity. [29] claim that their method of modeling the distribution of task MDPs as a hierarchical Bayesian infinite mixture model and then using this model to get a prior for a new MDP (new task) limits negative transfer. [17] takes the problem of MTRL a notch further by tackling the problem of multi-task, multi-agent RL under partial observability. They do so by introducing a form of a decentralized distillation of single-task policies into a shared policy.

Distraction Dilemma

Distraction Dilemma, although the term has not been used by any work as per our knowledge of yet, [18] uses it in the context of MTRL in reference to works we covered in Section 6. The idea is simple & trivial, that when the goal is to optimize a general policy function over multiple tasks, some of the tasks will tend to dominate others, which is, that these “dominant" tasks affect the learning procedure much more than others leading to the policy being better at this single-dominant-task performance & not on the rest. We can also view the issue is similar to the situation of mode collapse in Generative Adversarial Networks [9, 23], but of course, GANs are different from MTRL.

[14] were the first to use the term “distraction" to refer to the issue in concern. In fact, while other methods like [7, 27] have mentioned mitigating the issue as a bi-product of their work, in [14], the contribution is to handle the distraction dilemma. They use a PopArt normalization over a previous work IMPALA to tackle its issues & thereby improving upon the problem of some dominant tasks causing the policy to get distracted. A task can be a dominant one for a variety of reasons, one of which being that the stored experience is biased towards that task. [7] does not directly imply solving this dilemma, but does say that their method performs an implicit weighting of tasks. This, we think, should have a positive effect for the Distraction Dilemma, however, there are no results to support this claim as of yet.

[27] views this problem as the other extreme from the first issue of Task Interference, we discussed in Section 7. Section 7 also discusses this work in brief, & the authors claim that distillation helps mitigate this problem too.

Scalability

Scalability is an important issue for RL in general, and hence it trickles it way down to MTRL as well. Here scalability is the ability to scale well to many tasks for MTRL. These tasks may, however, be similar or “opposite" in nature. To bein with, [12] proves their technique is scalable by showing results of their work on complex RL problems, the adaptation on a high-dimensional locomotion task in the MuJoCo simulator & achieves good results. [14] claims as part of possible extensions that learning by on-policy methods may enable their work to scale to many tasks. They build their work on IMPALA [10] which uses the term “scale" with respect to scalability in compute, however, this gives them an advantage in learning the policy as well & they were able to report good results on large-scale tasks. [6] shows that their approach scales sub-linearly with an increasing number of tasks as their attention scheme automatically learns to group related tasks in the same of their architecture. They are among the few works, which also talks about being efficient even when action spaces of tasks are not aligned. Finally, [7] states that their work is itself motivated by sample complexity analysis & they prove that their work significantly reduced per task sample complexity when performing MTRL. Although they do not specifically comment about scaling to multiple tasks, but one can hope that alleviating per-task load will affect scaling to multiple tasks positively.

8 CONCLUSION

Multi-Task Reinforcement Learning (MTRL), majorly from the perspective of deep learning, was discussed in this report. MTRL is seen as an emerging field, with lots of relevant literature available today utilizing modern research works like deep neural networks. We motivated the use of MTRL and how the idea stems from exploiting similarities among structures of different tasks. We answered an important question of why multiple single task agents are not the optimal solution to the MTRL problem & that multi-task learning has positive impacts upon speed, parameter size, generalization, to name a few. Then, we presented the basic mechanics of the RL setting, the formal definition of a task & did it in a fashion to reconcile formalisms of various works discussed. To manifest these formal definitions, we example many examples of tasks and their respective domains for a variety of notable works. Once we had the basic machinery & understanding of the problem in place, we noted a few lines of questioning like whether the task information is known to the agent, what works have dealt with an online & offline setting, brief on how does model-free & model-based learning in MTRL algorithms behave, etc. We presented to the readers, several notable works in diverging directions of Meta-RL, Hierarchical RL & how distributed RL was leveraged for the MTRL problem. Alongside this, various related works were mentioned to get a better perspective on the problem. Last but not least, we discuss three of the most noted problem in the MTRL setting, namely Task Interference, Distraction Dilemma & scalability. We showed how different works dealt with these issues.

Since this is an active area of research, especially in contemporary times, we will end the report with some possible future directions. PopArt [14] showed that it could be used to rescale several reward functions (or tasks) and hence combining it with a policy distillation like Distral [27] can be effective. [1] is the only work that has the human input of policy sketches. These sketches can also serve as possible ways of involving interpretability into the RL mechanics. Explanations and Interpretability are an active area of research & symbol based explanations have majorly been used in AI planning for human in the loop [22]. The report highlights the need for a standard dataset to be followed by MTRL researches. There may be several environments for different types of task requirements, but unification is a must as it becomes difficult to compare results across these diverging works. Parallelization & Scalability is another area that future researches must take note of. Future works should include more extensive studies on Scalability and parallelization ability of their work & try to provide more formal guarantees like IMPALA [10].

REFERENCES

[1] Jacob Andreas, Dan Klein, and Sergey Levine. 2017. Modular multitask reinforcement learning with policy sketches. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 166–175.

[2] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. 2017. Hindsight Experience Replay. In *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.). Curran Associates, Inc., 5048–5058. http://papers.nips.cc/paper/7090-hindsight-experience-replay.pdf

[3] Charles Beattie, Joel Z Leibo, Denis Teplyashin, Tom Ward, Marcus Wainwright, Heinrich Küttler, Andrew Lefrancq, Simon Green, Víctor Valdés, Amir Sadik, et al. 2016. Deepmind lab. *arXiv preprint arXiv:1612.03801* (2016).

[4] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. 2013. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research* 47 (2013), 253–279.

[5] Richard Bellman. 1957. A Markovian decision process. *Journal of mathematics and mechanics* (1957), 679–684.

[6] Timo Bram, Gino Brunner, Oliver Richter, and Roger Wattenhofer. 2019. Attentive Multi-Task Deep Reinforcement Learning. *arXiv preprint arXiv:1907.02874* (2019).

[7] Emma Brunskill and Lihong Li. 2013. Sample complexity of multi-task reinforcement learning. *arXiv preprint arXiv:1309.6821* (2013).

[8] Daniele Calandriello, Alessandro Lazaric, and Marcello Restelli. 2014. Sparse multi-task reinforcement learning. In *Advances in Neural Information Processing Systems*. 819–827.

[9] Tong Che, Yanran Li, Athul Paul Jacob, Yoshua Bengio, and Wenjie Li. 2016. Mode regularized generative adversarial networks. *arXiv preprint arXiv:1612.02136* (2016).

[10] Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymir Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. 2018. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. *arXiv preprint arXiv:1802.01561* (2018).

[11] Benjamin Eysenbach, Xinyang Geng, Sergey Levine, and Ruslan Salakhutdinov. 2020. Rewriting History with Inverse RL: Hindsight Inference for Policy Improvement. *arXiv preprint arXiv:2002.11089* (2020).

[12] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 1126–1135.

[13] Francisco M Garcia, Chris Nota, and Philip S Thomas. 2020. Learning Reusable Options for Multi-Task Reinforcement Learning. *arXiv preprint arXiv:2001.01577* (2020).

[14] Matteo Hessel, Hubert Soyer, Lasse Espeholt, Wojciech Czarnecki, Simon Schmitt, and Hado van Hasselt. 2019. Multi-task deep reinforcement learning with popart. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 3796–3803.

[15] Lukasz Kaiser, Mohammad Babaeizadeh, Piotr Milos, Blazej Osinski, Roy H Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, et al. 2019. Model-based reinforcement learning for atari. *arXiv preprint arXiv:1903.00374* (2019).

[16] George Konidaris and Andrew G Barto. 2007. Building Portable Options: Skill Transfer in Reinforcement Learning.. In *IJCAI*, Vol. 7. 895–900.

[17] Shayegan Omidshafiei, Jason Pazis, Christopher Amato, Jonathan P How, and John Vian. 2017. Deep decentralized multi-task multi-agent reinforcement learning under partial observability. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR.org, 2681–2690.

[18] Jesus Rodriguez. 2019. Teaching Multi-Task Reinforcement Learning Agents to Not Get Distracted. https://towardsdatascience.com/teaching-multi-task-reinforcement-learning-agents-to-not-get-distracted-8d4b8e23deeb

[19] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. 2015. Trust region policy optimization. In *International conference on machine learning*. 1889–1897.

[20] Tianmin Shu, Caiming Xiong, and Richard Socher. 2017. Hierarchical and interpretable skill acquisition in multi-task reinforcement learning. *arXiv preprint arXiv:1712.07294* (2017).

[21] David Silver and Demis Hassabis. 2016. Alphago: Mastering the ancient game of go with machine learning. *Research Blog* 9 (2016).

[22] Sarath Sreedharan, Utkash Soni, Mudit Verma, Siddharth Srivastava, and Subbarao Kambhampati. 2020. Bridging the Gap: Providing Post-Hoc Symbolic Explanations for Sequential Decision-Making Problems with Black Box Simulators. *arXiv preprint arXiv:2002.01080* (2020).

[23] Akash Srivastava, Lazar Valkov, Chris Russell, Michael U Gutmann, and Charles Sutton. 2017. Veegan: Reducing mode collapse in gans using implicit variational learning. In *Advances in Neural Information Processing Systems*. 3308–3318.

[24] Richard S Sutton, Doina Precup, and Satinder Singh. 1999. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence* 112, 1-2 (1999), 181–211.

[25] Fumihide Tanaka and Masayuki Yamamura. 2003. Multitask reinforcement learning on the distribution of MDPs. In *Proceedings 2003 IEEE International Symposium on Computational Intelligence in Robotics and Automation. Computational Intelligence in Robotics and Automation for the New Millennium (Cat. No. 03EX694)*, Vol. 3. IEEE, 1108–1113.

[26] Matthew E Taylor and Peter Stone. 2009. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research* 10, Jul (2009), 1633–1685.

[27] Yee Teh, Victor Bapst, Wojciech M Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. 2017. Distral: Robust multitask reinforcement learning. In *Advances in Neural Information Processing Systems*. 4496–4506.

[28] Hado P van Hasselt, Arthur Guez, Matteo Hessel, Volodymyr Mnih, and David Silver. 2016. Learning values across many orders of magnitude. In *Advances in Neural Information Processing Systems*. 4287–4295.

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

[30] Ruihan Yang, Huazhe Xu, Yi Wu, and Xiaolong Wang. 2020. Multi-Task Reinforcement Learning with Soft Modularization. *arXiv preprint arXiv:2003.13661* (2020).

[31] Zhaoyang Yang, Kathryn E Merrick, Hussein A Abbass, and Lianwen Jin. 2017. Multi-Task Deep Reinforcement Learning for Continuous Action Control.. In *IJCAI*. 3301–3307.

[32] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. 2019. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. *arXiv preprint arXiv:1910.10897* (2019).

[33] Yu Zhang and Qiang Yang. 2017. A survey on multi-task learning. *arXiv preprint arXiv:1707.08114* (2017).