MERL: Multi-Head Reinforcement Learning

Yannis Flet-Berliac and **Philippe Preux**

SequeL Inria, University of Lille
CRIStAL, CNRS

Abstract

A common challenge in reinforcement learning is how to convert the agent's interactions with an environment into fast and robust learning. For instance, earlier work makes use of domain knowledge to improve existing reinforcement learning algorithms in complex tasks. While promising, previously acquired knowledge is often costly and challenging to scale up. Instead, we decide to consider problem knowledge with signals from quantities relevant to solve any task, e.g., self-performance assessment and accurate expectations. $\mathcal{V}^{ex}$ is such a quantity. It is the fraction of variance explained by the value function V and measures the discrepancy between V and the returns. Taking advantage of $\mathcal{V}^{ex}$, we propose MERL, a general framework for structuring reinforcement learning by injecting problem knowledge into policy gradient updates. As a result, the agent is not only optimized for a reward but learns using problem-focused quantities provided by MERL, applicable out-of-the-box to any task. In this paper: (a) We introduce and define MERL, the multi-head reinforcement learning framework we use throughout this work. (b) We conduct experiments across a variety of standard benchmark environments, including 9 continuous control tasks, where results show improved performance. (c) We demonstrate that MERL also improves transfer learning on a set of challenging pixel-based tasks. (d) We ponder how MERL tackles the problem of reward sparsity and better conditions the feature space of reinforcement learning agents.

1 Introduction

The problem of learning how to act optimally in an unknown dynamic environment has been a source of many research efforts for decades (Nguyen & Widrow, 1990; Werbos, 1989; Schmidhuber & Huber, 1991) and is still at the forefront of recent work in deep Reinforcement Learning (RL) (Burda et al., 2019; Ha & Schmidhuber, 2018; Silver et al., 2016; Espeholt et al., 2018). Nevertheless, current algorithms tend to be fragile and opaque (Iyer et al., 2018): they require a large amount of training data collected from an agent interacting with a simulated environment where the reward signal is often critically sparse. Collecting signals that will make the agent more efficient is, therefore, at the core of the algorithms designers' concerns.

Previous work in RL uses prior knowledge (Lin, 1992; Clouse & Utgoff, 1992; Ribeiro, 1998; Moreno et al., 2004) to reduce sample inefficiency. While promising and unquestionably necessary, the integration of such priors into current methods is likely costly to implement, it may cause undesired constraints and can hinder scaling up. Therefore, we propose a framework to directly integrate *non-limiting constraints* in current RL algorithms while being applicable to any task. In addition to an increased efficiency, the agent should learn from all interactions, not just the rewards. Indeed, if the probability of receiving a reward by chance is arbitrarily low, then the time required to learn from it will be arbitrarily long (Whitehead, 1991). This barrier to learning will prevent agents from significantly reducing learning time. One way to overcome this barrier is to learn

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

complementary and task-agnostic signals of self-performance assessment and accurate expectations from different sources (Schmidhuber, 1991; Oudeyer & Kaplan, 2007), whatever the task to master.

From the above considerations and building on existing auxiliary task methods, we design a framework that integrates problem knowledge quantities into the learning process. In addition to providing a method technically applicable to any policy gradient method or environment, the central idea of MERL is to incorporate a measure of the discrepancy between the estimated state value and the observed returns as an auxiliary task. This discrepancy is formalized with the notion of the fraction of variance explained $\mathcal{V}^{ex}$ (Kvålseth, 1985). One intuition the reader can have is that MERL transforms a reward-focused task into a task regularized with problem knowledge signals: self-performance assessment and accurate expectations.

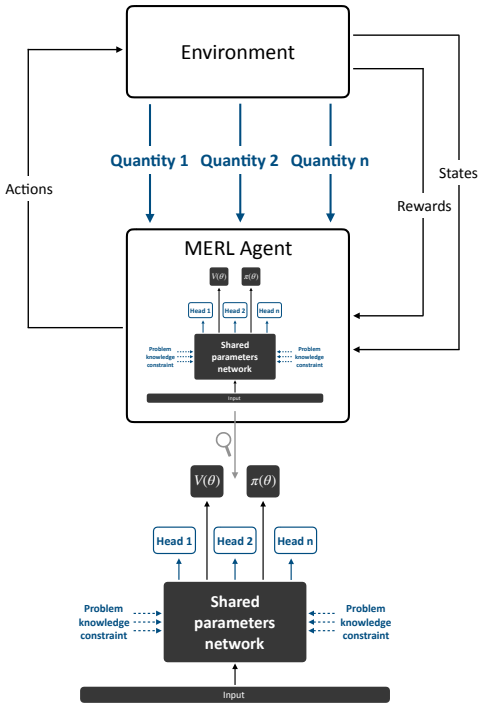

Figure 1: High-level overview of the Multi-hEad Reinforcement Learning (MERL) framework.

Fig. 1 provides a preliminary understanding of MERL assets: an enriched actor-critic architecture with a lightly modified learning algorithm places the agent amidst task-agnostic auxiliary quantities directly sampled from the environment. In the sequel of this paper, we use two problem knowledge quantities to demonstrate the performance of MERL: $\mathcal{V}^{ex}$, a compelling measure of self-performance, and future states prediction, commonly used in auxiliary task methods. The reader is further encouraged to introduce many other relevant signals. We demonstrate that while being able to predict the quantities from the different MERL heads correctly, the agent outperforms the on-policy baseline that does not use the MERL framework on various continuous control tasks. We also show that, interestingly, our framework allows to better transfer the learning, from one task to another, on several *Atari 2600* games.

2 Preliminaries

We consider a Markov Decision Process (MDP) with states $s \in \mathcal{S}$, actions $a \in \mathcal{A}$, transition distribution $s_{t+1} \sim P(s_t, a_t)$ and reward function $r(s, a)$. Let $\pi = \{\pi(a|s), s \in \mathcal{S}, a \in \mathcal{A}\}$ denote a stochastic policy and let the objective function be the traditional expected discounted reward:

$$J(\pi) \triangleq \mathop{\mathbb{E}}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^t r\left(s_t, a_t\right)\right], \tag{1}$$

where $\gamma \in [0, 1)$ is a discount factor (Puterman, 1994) and $\tau = (s_0, a_0, s_1, \dots)$ is a trajectory sampled from the environment. Policy gradient methods aim at modelling and optimizing the policy directly (Williams, 1992). The policy π is generally implemented with a function parameterized by θ. In the sequel, we will use θ to denote the parameters as well as the policy. In deep RL, the policy is represented in a neural network called the policy network and is assumed to be continuously differentiable with respect to its parameters θ.

2.1 Fraction of Variance Explained: $\mathcal{V}^{ex}$

The fraction of variance that the value function explains about the returns. It corresponds to the proportion of the variance in the dependent variable V that is predictable from the independent

variable s_t. We define $\mathcal{V}_\tau^{ex}$ as the *fraction of variance explained* for a trajectory τ:

$$\mathcal{V}_\tau^{ex} \triangleq 1 - \frac{\sum_{t \in \tau} \left(\hat{R}_t - V(s_t)\right)^2}{\sum_{t \in \tau} \left(\hat{R}_t - \overline{R}\right)^2}, \tag{2}$$

where $\hat{R}_t$ and $V(s_t)$ are respectively the return and the expected return from state $s_t \in \tau$, and $\overline{R}$ is the mean of all returns in trajectory τ. In statistics, this quantity is also known as the coefficient of determination R^2 and it should be noted that this criterion may be negative for non-linear models (Kvålseth, 1985), indicating a severe lack of fit of the corresponding function:

- $\mathcal{V}_\tau^{ex} = 1$: V perfectly explains the returns - V and the returns are *correlated*;
- $\mathcal{V}_\tau^{ex} = 0$ corresponds to a simple average prediction - V and the returns are *not correlated*;
- $\mathcal{V}_\tau^{ex} < 0$: V provides a worse fit to the outcomes than the mean of the returns.

One can have the intuition that $\mathcal{V}_\tau^{ex}$ close to 1 implies that the trajectory τ provides valuable signals because they correspond to transitions sampled from an exercised behavior. On the other hand, $\mathcal{V}_\tau^{ex}$ close to 0 indicates that the value function is not correlated with the returns, therefore, the corresponding samples are not expected to provide as valuable information as before. Finally, $\mathcal{V}_\tau^{ex} < 0$ corresponds to a high mean-squared error for the value function, which means for the related trajectory that the agent still has to learn to perform better. In Flet-Berliac & Preux (2019), policy gradient methods are improved by using $\mathcal{V}^{ex}$ as a criterion to dropout transitions before each policy update. We will show that $\mathcal{V}^{ex}$ is also a relevant indicator for assessing self-performance in the context of MERL agents.

2.2 Policy Gradient Method: PPO with Clipped Surrogate Objective

In this paper, we consider on-policy learning primarily for its unbiasedness and stability compared to off-policy methods (Nachum et al., 2017). On-policy is also empirically known as being less sample efficient than off-policy learning hence this issue emerged as an interesting research topic. However, our method can be applied to off-policy methods as well, and we leave this investigation open for future work.

PPO (Schulman et al., 2017) is among the most commonly used and state-of-the-art on-policy policy gradient methods. Indeed, PPO has been tested on a set of benchmark tasks and has proven to produce impressive results in many cases despite a relatively simple implementation. For instance, instead of imposing a hard constraint like TRPO (Schulman et al., 2015), PPO formalizes the constraint as a penalty in the objective function. In PPO, at each iteration, the new policy θ_{new} is obtained from the old policy θ_{old}:

$$\theta_{new} \leftarrow \underset{\theta}{\operatorname{argmax}} \underset{s_t, a_t \sim \pi_{\theta_{old}}}{\mathbb{E}} \left[L^{\text{PPO}}\left(s_t, a_t, \theta_{old}, \theta\right) \right]. \tag{3}$$

We use the clipped version of PPO whose objective function is:

$$L^{\text{PPO}}(s_t, a_t, \theta_{old}, \theta) = \min \left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} A^{\pi_{\theta_{old}}}(s_t, a_t),\ g(\epsilon, A^{\pi_{\theta_{old}}}(s_t, a_t)) \right), \tag{4}$$

where

$$g(\epsilon, A) = \begin{cases} (1+\epsilon)A, A \geq 0 \\ (1-\epsilon)A, A < 0. \end{cases} \tag{5}$$

A is the advantage function, $A(s,a) \triangleq Q(s,a) - V(s)$. The expected advantage function $A^{\pi_{\theta_{old}}}$ is estimated by an old policy and then re-calibrated using the probability ratio between the new and the old policy. In Eq. 4, this ratio is constrained to stay within a small interval around 1, making the training updates more stable.

2.3 Related Work

Auxiliary tasks have been adopted to facilitate representation learning for decades (Suddarth & Kergosien, 1990; Klyubin et al., 2005), along with intrinsic motivation (Schmidhuber, 2010; Pathak

et al., 2017) and artificial curiosity (Schmidhuber, 1991; Oudeyer & Kaplan, 2007). The use of auxiliary tasks to allow the agents to maximize other pseudo-reward functions simultaneously has been studied in a number of previous works (Shelhamer et al., 2016; Dosovitskiy & Koltun, 2016; Burda et al., 2018; Du et al., 2018; Riedmiller et al., 2018; Kartal et al., 2019), including incorporating unsupervised control tasks and reward predictions in the UNREAL framework (Jaderberg et al., 2016), applying auxiliary tasks to navigation problems (Mirowski et al., 2016), or for utilizing representation learning (Lesort et al., 2018) in the context of model-based RL. Lastly, in imitation learning of sequences provided by experts, Li et al. (2016) introduces a supervised loss for fitting a recurrent model on the hidden representations to predict the next observed state.

Our method incorporates two key contributions: a multi-head layer with auxiliary task signals both environment-agnostic and technically applicable to any policy gradient method, and the use of $\mathcal{V}_\tau^{ex}$ as an auxiliary task to measure the discrepancy between the value function and the returns in order to allow for better self-performance assessment and more efficient learning. In addition, MERL differs from previous approaches in that its framework addresses all the advantages mentioned hereafter: (a) neither the introduction of new neural networks (e.g., for memory) nor the introduction of a replay buffer or an off-policy setting is needed, (b) all relevant quantities are compatible with any task and is not limited to pixel-based environments, (c) no additional iterations are required, and no modification to the reward function of the policy gradient algorithms it is applied to is necessitated. The above reasons make MERL generally applicable and technically suitable out-of-the-box to most policy gradient algorithms with a negligible computational cost overhead.

From a different perspective, Garcıa & Fernández (2015) gives a detailed overview of previous work that has changed the optimality criterion as a safety factor. But most methods use a hard constraint rather than a penalty; one reason is that it is difficult to choose a single coefficient for this penalty that works well for different problems. We are successfully addressing this problem with MERL. In Lipton et al. (2016), catastrophic actions are avoided by training an intrinsic fear model to predict whether a disaster will occur and using it to shape rewards. Compared to both methods, MERL is more scalable and lightweight while it successfully incorporates quantities of self-performance assessments (e.g., variance explained of the value function) and accurate expectations (e.g., next state prediction) leading to an improved performance.

3 Multi-Head Framework for Reinforcement Learning using $\mathcal{V}^{ex}$

Our multi-head architecture and its associated learning algorithm are directly applicable to most state-of-the-art policy gradient methods. Let h be the index of each MERL head: MERL^h. We propose two of the quantities predicted by these heads and show how to incorporate them into PPO.

3.1 Policy and Value Function Representation

In deep RL, the policy is generally represented in a neural network called the policy network, with parameters θ, and the value function is parameterized by the value network, with parameters ϕ. Each MERL head MERL^h takes as input the last embedding layer from the value network and is constituted of only one layer of fully-connected neurons, with parameters ϕ_h. The output size of each head corresponds to the size of the predicted MERL quantity. Below, we elaborate on two.

3.2 $\mathcal{V}^{ex}$ Estimation

In order to have an estimate of the fraction of variance explained, we write $\mathrm{MERL}^{\mathrm{VE}}$ as the corresponding MERL head with parameters ϕ^{VE}. Its objective function is defined by:

$$L^{\mathrm{MERL}^{\mathrm{VE}}}(\tau, \phi, \phi^{\mathrm{VE}}) = \|\mathrm{MERL}^{\mathrm{VE}}(\tau) - \mathcal{V}_\tau^{ex}\|_2^2. \tag{6}$$

3.3 Future States Estimation

Auxiliary task methods based on next state prediction are, to the best of our knowledge, the most commonly used in the RL literature. We include such auxiliary task into MERL, in order to assimilate our contribution to the previous work and to provide a enriched evaluation of the proposed framework. At each timestep, one of the agent's MERL heads predicts a future state s' from s. While a typical

MERL quantity can be fit by regression on mean-squared error, we observed that predictions of future states are better fitted with a cosine-distance error. We denote $\mathrm{MERL}^{\mathrm{FS}}$ the corresponding head, with parameters ϕ^{FS}, and S the observation space size (size of vector s). We define its objective function as:

$$L^{\mathrm{MERL}^{\mathrm{FS}}}(s, \phi, \phi^{\mathrm{FS}}) = 1 - \frac{\sum_{i=1}^{S} \mathrm{MERL}_i^{\mathrm{FS}}(s) \cdot s'_i}{\sqrt{\sum_{i=1}^{S} (\mathrm{MERL}_i^{\mathrm{FS}}(s))^2} \sqrt{\sum_{i=1}^{S} (s'_i)^2}}. \tag{7}$$

3.4 Problem-Constrained Policy Update

Once a set of MERL heads MERL^h and their associated objective functions L^{MERL^h} have been defined, we modify the gradient update step of the policy gradient algorithms. The objective function incorporates all L^{MERL^h}. Of course, each MERL objective is associated with its coefficient c_h. It is worthy to note that we used the exact same MERL coefficients for all our experiments, which demonstrate the framework's ease of applicability. Algorithm 1 illustrates how the learning is achieved. In Eq. (9), only the (boxed) MERL objectives parameterized by ϕ are added to the value update and modify the learning algorithm.

Algorithm 1 PPO+MERL update.

Initialise policy parameters θ_0
Initialise value function and MERL^h functions parameters ϕ_0

for k = 0,1,2,... **do**
Collect *set of trajectories* $\mathcal{D}_k = \{\tau_i\}$ with horizon T by running policy π_{θ_k} in the environment
Compute MERL^h *estimates* at timestep t from sampling the environment
Compute *advantage estimates* A_t at timestep t based on the current value function V_{ϕ_k}
Compute *future rewards* $\hat{R}_t$ from timestep t

Gradient Update

$$\theta_{k+1} = \underset{\theta}{\operatorname{argmax}} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \min \left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} A^{\pi_{\theta_k}}(s_t, a_t), g(\epsilon, A^{\pi_{\theta_k}}(s_t, a_t)) \right) \tag{8}$$

$$\phi_{k+1} = \underset{\phi}{\operatorname{argmin}} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \left(V_{\phi_k}(s_t) - \hat{R}_t \right)^2 + \boxed{\sum_{h=0}^{H} c_h L^{\mathrm{MERL}^h}} \tag{9}$$

4 MERL applied to Continuous Control Tasks and the Atari domain

4.1 Methodology

We evaluate MERL in multiple high-dimensional environments, ranging from *MuJoCo* (Todorov et al., 2012) to the *Atari 2600* games (Bellemare et al., 2013). The experiments in *MuJoCo* allow us to evaluate the performance of MERL on a large number of different continuous control problems. It is worthy to note that the universal characteristics of the auxiliary quantities we design ensure that MERL is directly applicable to any task. Other popular auxiliary task methods (Jaderberg et al., 2016; Mirowski et al., 2016; Burda et al., 2018) are not out-of-the-box applicable to continuous control tasks like *MuJoCo*. Thus, we naturally compare the performance of our method with PPO (Schulman et al., 2017) where MERL heads are not used. Later, we also experiment with MERL on the *Atari 2600* games to study the transfer learning abilities of our method on a set of diverse tasks.

Implementation. For the continuous control *MuJoCo* tasks, the agents have learned using separated policy and value networks. In this case, we build upon the value network to incorporate our framework's heads. On the contrary, when playing *Atari 2600* games from pixels, the agents were

given a CNN network shared between the policy and the value function. In that case, MERL^h are naturally attached to the last embedding layer of the shared network. In both configurations, the outputs of MERL^h heads are the same size as the quantity they predict: for instance, MERL^{VE} is a scalar whereas MERL^{FS} is a state.

Hyper-parameters Setting. We used the same hyper-parameters as in the main text of the corresponding paper. We made this choice within a clear and objective protocol of demonstrating the benefits of using MERL. Hence, its reported performance is not necessarily the best that can be obtained, but it still exceeds the baseline. Using MERL adds as many hyper-parameters as there are heads in the multi-head layer and it is worth noting that MERL hyper-parameters are the same for all tasks. We report all hyper-parameters in Tables 1 and 2.

Table 1: Hyper-parameters used in PPO+MERL

Hyper-parameter	Value
Horizon (T)	2048 (MuJoCo), 128 (Atari)
Adam stepsize	$3 \cdot 10^{-4}$ (MuJoCo), $2.5 \cdot 10^{-4}$ (Atari)
Nb. epochs	10 (MuJoCo), 3 (Atari)
Minibatch size	64 (MuJoCo), 32 (Atari)
Number of actors	1 (MuJoCo), 4 (Atari)
Discount (γ)	0.99
GAE parameter (λ)	0.95
Clipping parameter (ϵ)	0.2 (MuJoCo), 0.1 (Atari)
Value function coef	0.5

Table 2: MERL hyper-parameters

Hyper-parameter	Value
MERL^{VE} coef c_{VE}	0.5
MERL^{FS} coef c_{FS}	0.01

Performance Measures. We examine the performance across a large number of trials (with different seeds for each task). Standard deviation of returns, and average return are generally considered to be the most stable measures used to compare the performance of the algorithms being studied (Islam et al., 2017). Thereby, in the rest of this work, we use those metrics to establish the performance of our framework quantitatively.

4.2 Single-Task Learning: Continuous Control

We apply MERL to PPO in several continuous control tasks, where using auxiliary tasks has not been explored in detail in the literature. Specifically, we use 9 *MuJoCo* environments. Due to space constraints, only 3 graphs from varied tasks are shown in Fig. 2. The complete set of 9 tasks is reported in Table 3.

Table 3: Average total reward of the last 100 episodes over 7 runs on the 9 MuJoCo environments. **Boldface** $mean \pm std$ indicate statistically better performance.

Task	PPO	Ours
Ant	1728 ± 64	$\mathbf{2157 \pm 212}$
HalfCheetah	1557 ± 21	$\mathbf{2117 \pm 370}$
Hopper	2263 ± 125	2105 ± 200
Humanoid	577 ± 10	$\mathbf{603 \pm 8}$
InvertedDoublePendulum	5965 ± 108	$\mathbf{6604 \pm 130}$
InvertedPendulum	474 ± 14	$\mathbf{497 \pm 12}$
Reacher	-7.84 ± 0.7	-7.78 ± 0.8
Swimmer	93.2 ± 8.7	$\mathbf{124.6 \pm 5.6}$
Walker2d	2309 ± 332	2347 ± 353

The results demonstrate that using MERL leads to better performance on a variety of continuous control tasks. Moreover, learning seems to be faster for some tasks, suggesting that MERL takes advantage of its heads to learn relevant quantities from the beginning of learning, when the reward signals may be sparse. Interestingly, by looking at the performance across all 9 tasks, we observed better results by predicting only the next state and not the subsequent ones.

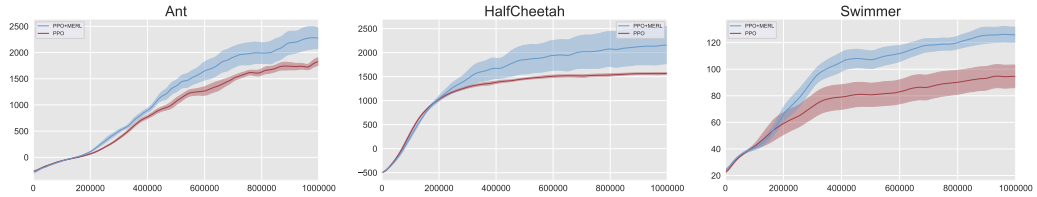

Figure 2: Experiments on 3 MuJoCo environments (10^6 timesteps, 7 seeds) with PPO+MERL. Red is the baseline, blue is with our method. The line is the average performance, while the shaded area represents its standard deviation.

4.3 Transfer Learning: Atari Domain

Because of training time constraints, we consider a transfer learning setting where, after the first 10^6 training steps, the agent switches to a new task for another 10^6 steps. The agent is not aware of the task switch. *Atari 2600* has been a challenging testbed for many years due to its high-dimensional video input (210 x 160) and the discrepancy of tasks between games. To investigate the advantages of MERL in transfer learning, we choose a set of 6 Atari games with an action space of 9, which is the average size of the action space in the Atari domain. This experimental choice is beneficial in that the 6 games provide a diverse range of game-play while sticking to the same size of action space.

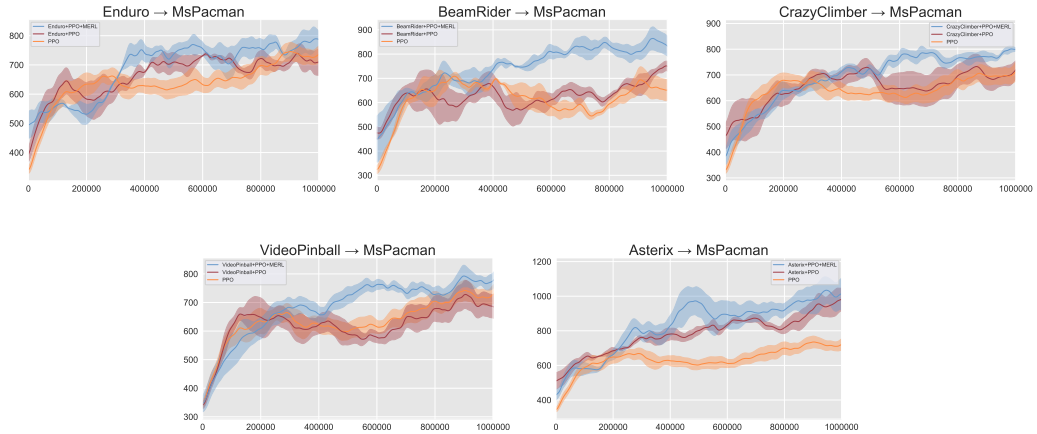

Figure 3: Transfer learning tasks from 5 Atari games to Ms. Pacman (2×10^6 timesteps, 4 seeds). Performance on the second task. Orange is PPO solely trained on Ms. Pacman, red and blue are respectively PPO and our method transferring the learning. The line is the average performance, while the shaded area represents its standard deviation.

Fig. 3 demonstrates that our method can better adapt to different tasks. This can suggest that MERL heads learn and help represent information that is more generally relevant for other tasks, such as self-performance assessment or accurate expectations. In addition to adding a regularization term to the objective function with problem knowledge signals, those auxiliary quantities make the neural network optimize for task-agnostic sub-objectives.

4.4 Ablation Study

We conduct an ablation study to evaluate the separate and combined contributions of the two heads. Fig. 4 shows the comparative results in HalfCheetah, Walker2d, and Swimmer. Interestingly, with

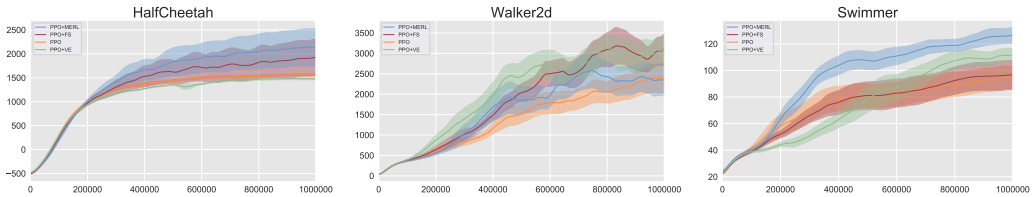

Figure 4: Ablation experiments with only one MERL head (FS or VE) (10^6 timesteps, 4 seeds). Blue is MERL with the two heads, red with the FS head, green with the VE head and orange with no MERL head. The line is the average performance, the shaded area represents its standard deviation.

HalfCheetah, using only the $\mathrm{MERL}^{\mathrm{VE}}$ head degrades the performance, but when it is combined with the $\mathrm{MERL}^{\mathrm{FS}}$ head, it outperforms PPO+FS. Results of the complete ablation analysis demonstrate that each head is potentially valuable for enhancing learning and that their combination can produce remarkable results. In addition, it may be intuited that finding a variety of complementary MERL heads to cover the scope of the problem in a holistic perspective can significantly improve learning.

4.5 Discussion

The experiments suggest that MERL successfully optimizes the policy according to complementary quantities seeking for good performance and safe realization of tasks, i.e. it does not only maximize a reward but instead ensures the control problem is appropriately addressed. Moreover, we show that MERL is directly applicable to policy gradient methods while adding a negligible computation cost. Indeed, for the *MuJoCo* and *Atari* tasks, the computational cost overhead is respectively 5% and 7% with our training infrastructure. All of these factors result in a generally applicable algorithm that more robustly solves difficult problems in a variety of environments with continuous action spaces or by using only raw pixels for observations. Thanks to a consistent choice of complementary quantities injected in the optimization process, MERL is able to better align an agent's objectives with higher-level insights into how to solve a control problem. Besides, since many current methods involve that successful learning depends on the agent's chance to reach the goal by chance in the first place, correctly predicting MERL heads gives the agent an opportunity to learn from useful signals while improving in a given task.

5 Conclusion

In this paper, we introduced $\mathcal{V}^{ex}$, a new auxiliary task, to measure the discrepancy between the value function and the returns, which successfully assesses the agent's performance and helps learn more efficiently. We also proposed MERL, a generally applicable deep RL framework for learning problem-focused representations, which we demonstrated the effectiveness with a combination of two auxiliary tasks. We established that injecting problem knowledge signals directly in the policy gradient optimization allows for a better state representation that is generalizable to many tasks. $\mathcal{V}^{ex}$ provides a more problem-focused state representation to the agent, which is, therefore, not only reward-centric. MERL can be labeled as being a hybrid model-free and model-based framework, formed with lightweight embedded models of self-performance assessment and accurate expectations. MERL heads introduce a regularization term to the function approximation while addressing the problem of reward sparsity through auxiliary task learning. Those features nourish a framework technically applicable to any policy gradient algorithm or environment; it does not need to be redesigned for different problems and can be extended with other relevant problem-solving quantities, comparable to $\mathcal{V}^{ex}$.

Although the relevance and higher performance of MERL have only been shown empirically, we think it would be interesting to study the theoretical contribution of this framework from the perspective of an implicit regularization of the agent's representation on the environment. We also believe that the identification of additional MERL quantities (e.g., prediction of time until the end of a trajectory) and the effect of their combination is also a research topic that we find most relevant for future work.

References

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

Yuri Burda, Harri Edwards, Deepak Pathak, Amos Storkey, Trevor Darrell, and Alexei A Efros. Large-scale study of curiosity-driven learning. *arXiv preprint arXiv:1808.04355*, 2018.

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. In *International Conference on Learning Representations*, 2019.

Jeffery A Clouse and Paul E Utgoff. A teaching method for reinforcement learning. In *Machine Learning*, pp. 92–101. Elsevier, 1992.

Alexey Dosovitskiy and Vladlen Koltun. Learning to act by predicting the future. In *International Conference on Learning Representations*, 2016.

Yunshu Du, Wojciech M Czarnecki, Siddhant M Jayakumar, Razvan Pascanu, and Balaji Lakshminarayanan. Adapting auxiliary losses using gradient similarity. *arXiv preprint arXiv:1812.02224*, 2018.

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymir Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International Conference on Machine Learning*, pp. 1406–1415, 2018.

Yannis Flet-Berliac and Philippe Preux. Samples are useful? not always: denoising policy gradient updates using variance explained. *arXiv preprint arXiv:1904.04025*, 2019.

Javier Garcıa and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.

David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution, 2018.

Riashat Islam, Peter Henderson, Maziar Gomrokchi, and Doina Precup. Reproducibility of benchmarked deep reinforcement learning tasks for continuous control. *arXiv preprint arXiv:1708.04133*, 2017.

Rahul Iyer, Yuezhang Li, Huao Li, Michael Lewis, R Sundar, and Katia P Sycara. Transparency and explanation in deep reinforcement learning neural networks. In *Proceedings of AAAI/ACM Conference on Artificial Intelligence, Ethics, and Society*. AAAI/ACM, 2018.

Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397*, 2016.

Bilal Kartal, Pablo Hernandez-Leal, and Matthew E Taylor. Terminal prediction as an auxiliary task for deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 15, pp. 38–44, 2019.

AS Klyubin, D Polani, and CL Nehaniv. Empowerment: a universal agent-centric measure of control. In *Proceedings of the 2005 IEEE Congress on Evolutionary Computation 1 pp. 128-*, 2005.

Tarald O Kvålseth. Cautionary Note about R^2. *The American Statistician*, 39(4):279–285, 1985.

Timothée Lesort, Natalia Díaz-Rodríguez, Jean-Franois Goudou, and David Filliat. State representation learning for control: An overview. *Neural Networks*, 108:379–392, 2018.

Xiujun Li, Lihong Li, Jianfeng Gao, Xiaodong He, Jianshu Chen, Li Deng, and Ji He. Recurrent reinforcement learning: a hybrid approach. In *International Conference on Learning Representations*, 2016.

Long-Ji Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8(3-4):293–321, 1992.

Zachary C Lipton, Kamyar Azizzadenesheli, Abhishek Kumar, Lihong Li, Jianfeng Gao, and Li Deng. Combating reinforcement learning's sisyphean curse with intrinsic fear. *arXiv preprint arXiv:1611.01211*, 2016.

Piotr Mirowski, Razvan Pascanu, Fabio Viola, Hubert Soyer, Andrew J Ballard, Andrea Banino, Misha Denil, Ross Goroshin, Laurent Sifre, Koray Kavukcuoglu, et al. Learning to navigate in complex environments. *arXiv preprint arXiv:1611.03673*, 2016.

David L Moreno, Carlos V Regueiro, Roberto Iglesias, and Senén Barro. Using prior knowledge to improve reinforcement learning in mobile robotics. In *Proceedings Towards Autonomous Robotics Systems*, 2004.

Ofir Nachum, Mohammad Norouzi, Kelvin Xu, and Dale Schuurmans. Bridging the gap between value and policy based reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 2775–2785, 2017.

Derrick Nguyen and Bernard Widrow. The truck backer-upper: An example of self-learning in neural networks. In *Advanced Neural Computers*, pp. 11–19, 1990.

Pierre-Yves Oudeyer and Frederic Kaplan. What is intrinsic motivation? a typology of computational approaches. *Frontiers in Neurorobotics*, 1:6, 2007.

Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 16–17, 2017.

Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1994. ISBN 0471619779.

Carlos HC Ribeiro. Embedding a priori knowledge in reinforcement learning. *Journal of Intelligent and Robotic Systems*, 21(1):51–71, 1998.

Martin Riedmiller, Roland Hafner, Thomas Lampe, Michael Neunert, Jonas Degrave, Tom Wiele, Vlad Mnih, Nicolas Heess, and Jost Tobias Springenberg. Learning by playing solving sparse reward tasks from scratch. In *International Conference on Machine Learning*, pp. 4341–4350, 2018.

J Schmidhuber and R Huber. Learning to generate artificial fovea trajectories for target detection. *International Journal of Neural Systems*, 2(1/2):135–141, 1991.

Jürgen Schmidhuber. Curious model-building control systems. In *IEEE International Joint Conference on Neural Networks*, pp. 1458–1463. IEEE, 1991.

Jürgen Schmidhuber. Formal theory of creativity, fun, and intrinsic motivation (1990–2010). *IEEE Transactions on Autonomous Mental Development*, 2(3):230–247, 2010.

John Schulman, Sergey Levine, Pieter Abbeel, Michael I Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, pp. 1928–1937, 2015.

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

Evan Shelhamer, Parsa Mahmoudieh, Max Argus, and Trevor Darrell. Loss is its own reward: Self-supervision for reinforcement learning. In *International Conference on Learning Representations*, 2016.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529:484, 2016.

S Suddarth and Y Kergosien. Rule-injection hints as a means of improving network performance and learning time. *Neural Networks*, pp. 120–129, 1990.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE, 2012.

Paul J Werbos. Neural networks for control and system identification. In *IEEE Conference on Decision and Control,*, pp. 260–265, 1989.

S.D. Whitehead. Complexity and cooperation in q-learning. In *Eighth International Workshop on Machine Learning*, pp. 363–367, 1991.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.