MERL: Multi-Head Reinforcement Learning

Yannis Flet-Berliac, Philippe Preux

MERL: Multi-Head Reinforcement Learning

Yannis Flet-Berliac and **Philippe Preux**

Univ. Lille, SequeL Inria,
UMR 9189 - CRIStAL, CNRS

Abstract

A common challenge in reinforcement learning is how to efficiently sample an environment to convert the agent's interactions into fast and robust learning, leading to high performance in complex tasks. For instance, earlier work makes use of domain/prior knowledge to improve existing reinforcement learning algorithms. While promising, previously acquired knowledge is often costly and challenging to scale up. Instead, we decide to consider the use of problem knowledge, which constitutes signals from any relevant quantity useful to solve many tasks, e.g., self-performance assessment and accurate expectations. We propose MERL, a general framework for structuring reinforcement learning by injecting problem knowledge into policy gradient updates. Unlike other auxiliary tasks methods, MERL is generally applicable to any task. As a result, policy and value functions are no longer only optimized for a reward but are learned using task-agnostic quantities. In this paper: (a) We introduce and define MERL, our new multi-head reinforcement learning framework. (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 tasks. (d) We investigate how our approach tackles the problem of reward sparsity and better condition the feature space in the context of deep 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 [20, 36, 26, 25] and is still at the forefront of recent works in deep reinforcement learning (DRL) [4, 9, 31, 6]. Nevertheless, current algorithms tend to be fragile and opaque [11]: 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. We consider two lines of thought.

First, previous work in reinforcement learning uses prior knowledge [15, 5, 18, 24] as a resource for reducing 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 integrate non-limiting constraints directly in current RL methods and that is directly applicable to most tasks. Second, if the probability of receiving a reward by chance is arbitrarily low, the time required to learn from it will be arbitrarily long [37]. This barrier to learning will prevent agents from significantly reducing learning time. One way to overcome this barrier is to learn by getting alternative and complementary signals from different sources [21, 27], whatever the task to master.

5th Deep Reinforcement Learning Workshop.
33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

Building upon existing auxiliary tasks methods, we design MERL, a framework that integrates problem-oriented quantities into the learning process. In this framework, in addition to learn to maximize the returns while sampling an environment, the agent also learns to correctly evaluate its performance on the control problem by minimizing MERL objective functions augmented with several problem-focused quantities. One intuition the reader can have is that MERL transforms a reward-focused task into a task regularized with problem knowledge signals. In contrast to existing auxiliary tasks methods, our framework is directly applicable to most environments. In turns, the agents can learn a richer representation of the environment and consequently better address the task at hand. Fig. 1 provides a high-level overview of this approach, providing a preliminary understanding of MERL properties: task-agnostic auxiliary quantities sampled from the environment and an enhanced actor-critic architecture with a lightly modified learning algorithm.

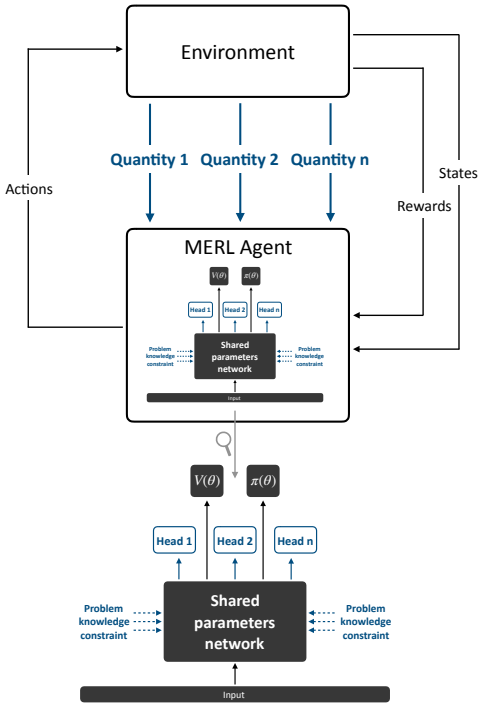

Figure 1: High-level overview of MERL.

In the sequel of this paper, we first present the on-policy gradient method used to demonstrate the performance of MERL. We choose on-policy learning primarily for its unbiasedness and stability compared to off-policy methods [19]. In addition, on-policy is empirically known as being less sample efficient than off-policy learning; hence, this issue emerged as an interesting research topic. Nevertheless, our method can be applied to off-policy methods as well, and we leave this investigation open for future work.

Then, we introduce the framework and detail on how it can be applied to most DRL methods and any environment. We propose two of the multi-head/problem knowledge quantities mentioned above, which are enough to improve the performance, but the reader is further encouraged to introduce 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. Finally we show that 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) defined on a state space $\mathcal{S}$, an action space $\mathcal{A}$ and a reward function $r(s, a)$ where $s \in \mathcal{S}$, $a \in \mathcal{A}$. 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 [33] 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 [34]. The policy π_θ is generally modeled with a function parameterized by θ; in the rest of the paper, π_θ and θ denote the same policy. In DRL, the policy is represented in a neural network called the policy network. Proximal Policy Optimization (PPO) [30] is among the most commonly used and state-of-the-art policy gradient methods.

2.1 Policy Gradient Method with Clipped Surrogate Objective

PPO-Clip [30] is an on-policy gradient-based algorithm. In previous work, 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 [29], 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}:

$$\theta_{new} \leftarrow \underset{\theta}{\operatorname{argmax}} \underset{s,a\sim\pi_{\theta_{old}}}{\mathbb{E}} \left[L^{\mathrm{PPO}}\left(s, a, \theta_{old}, \theta\right) \right]. \tag{2}$$

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

$$L^{\mathrm{PPO}}(s, a, \theta_{old}, \theta) = \min\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)} A^{\pi_{\theta_{old}}}(s, a),\ g(\epsilon, A^{\pi_{\theta_{old}}}(s, a)) \right), \tag{3}$$

where

$$g(\epsilon, A) = \begin{cases} (1+\epsilon)A & A \geq 0 \\ (1-\epsilon)A & A < 0. \end{cases} \tag{4}$$

By taking the minimum of the two terms in Eq. (3), the ratio $\frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)}$ is constrained to stay within a small interval around 1. The expected advantage function $A^{\pi_{\theta_{old}}}$ for the new policy is estimated by an old policy and then re-calibrated using the probability ratio between the new and the old policy.

2.2 Related Work

MERL incorporates three key ideas: (a) the use of auxiliary quantities as a means to tackle the problem of reward sparsity, (b) the introduction of quantities of self-performance assessment and accurate expectations which constitute task-agnostic signals, widely applicable unlike environment-specific prior knowledge, (c) policy and value function approximation with neural networks augmented with a multi-head layer.

Auxiliary tasks have been used to facilitate representation learning for decades [32], along with intrinsic motivation [28, 22] and artificial curiosity [21, 27]. In the context of imitation learning of sequences provided by experts, [14] introduces a supervised loss for fitting a recurrent model on the hidden representations to predict the next observed state. Other works on auxiliary tasks [12, 17, 3] allow the agents to maximize other pseudo-reward functions simultaneously. Our method is much distinctive from those previous approaches. First, neither the introduction of additional neural networks (*e.g.* for memory) nor the introduction of a replay buffer is needed. Second, the quantities we introduce are compatible with *any* task: previous methods only work on pixel-based environments. Third, MERL neither introduces additional iterations nor modifies the reward function of the policy gradient algorithms it is applied to. For all those reasons, MERL is generally applicable and can be added out-of-the-box to most policy gradient algorithms with a negligible computational cost overhead.

From a different perspective, [8] 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 [16], 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 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.

Finally, many previous studies have focused on the use of imitation learning to address a task directly. There are two main approaches: behavioral cloning [23], which attempts to learn a task policy through supervised learning, and inverse RL [1], which attempts to learn a reward function from a set of demonstrations. Although, these successful approaches often push the agent only to learn how to perform a task from expert demonstrations with a relatively modest understanding of its own behavior. We propose a method that gives the agent relevant quantities that brings its learning closer to *how to learn to accomplish* the task at hand, in addition to being optimized to only solve it.

3 MERL: Multi-Head Framework for Reinforcement Learning

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. In the context of DRL, we introduce two of the quantities predicted by these heads and show how to incorporate them in PPO-Clip.

3.1 Value Function and Policy

In DRL, 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 introduce two examples of these quantities.

3.2 Variance Explained

Let us define the first quantity we use in MERL: the *fraction of variance explained* $\mathcal{V}^{ex}$. We compute the fraction of variance that the value function V_ϕ explains about the returns $\hat{R}$. In other terms, it indicates the ability of the value function to model the samples. It corresponds to the proportion of the variance in the dependent variable that is predictable from the independent variables. We compute $\mathcal{V}^{ex}$ at each policy gradient update with the samples used for the gradient computation. In statistics, this quantity is also known as the coefficient of determination R^2 [13]. For the sake of clarity, we will not use this notation for the coefficient of determination, but we will refer to this criterion as:

$$\mathcal{V}^{ex} = 1 - \frac{\sum_{t=0}^{T} \left(\hat{R}_t - V_\phi(s_t)\right)^2}{\sum_{t=0}^{T} \left(\hat{R}_t - \overline{R}\right)^2}. \tag{5}$$

It should be noted that this criterion may be negative for non-linear models, indicating a severe lack of fit [13] of the corresponding function:

- $\mathcal{V}^{ex} = 1$ if the fitted value function V_ϕ perfectly explains the returns;
- $\mathcal{V}^{ex} = 0$ corresponds to a simple average prediction;
- $\mathcal{V}^{ex} < 0$ if the fitted value function provides a worse fit to the outcomes than the mean of the discounted rewards.

We denote $\mathrm{MERL}^{\mathrm{VE}}$ as the corresponding MERL head, with parameters ϕ^{VE} and its objective function is defined by:

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

Interpretation. $\mathcal{V}^{ex}$ measures the ability of the value function to fit the returns. $\mathcal{V}^{ex} = 0.43$ implies that 43% of the variability of the dependent variable $\hat{R}$ has been accounted for, and the remaining 57% of the variability is still unaccounted for. For instance in [7], $\mathcal{V}^{ex}$ is used to filter the samples that will be used to update the policy. By its definition, this quantity is a highly relevant indicator for assessing self-performance in reinforcement learning.

3.3 Future States

At each timestep, one of the agent's MERL heads tries to predict 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:

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

3.4 Problem-Constrained Policy Update

Once the 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 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

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

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

4 Experiments

4.1 Methodology

We evaluate MERL in multiple high-dimensional environments, ranging from *MuJoCo* [35] to the *Atari 2600* games [2]. 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 tasks methods [12, 17, 3] cannot be applied to challenging continuous control tasks like *MuJoCo*. Thus, we naturally compare the performance of our method with PPO [30]. The experiments on the *Atari 2600* games allow us to study the transfer learning abilities of MERL 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, $\mathrm{MERL}^{\mathrm{VE}}$ is a scalar whereas $\mathrm{MERL}^{\mathrm{FS}}$ 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 [10]. 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 *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 better performance.

Task	PPO	Ours
Ant	1728 ± 64	$\mathbf{2157 \pm 212}$
HalfCheetah	1557 ± 21	$\mathbf{2117 \pm 370}$
Hopper	$\mathbf{2263 \pm 125}$	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	$\mathbf{-7.78 \pm 0.8}$
Swimmer	93.2 ± 8.7	$\mathbf{124.6 \pm 5.6}$
Walker2d	2309 ± 332	$\mathbf{2347 \pm 353}$

We see from the results 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.

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 and is trained 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

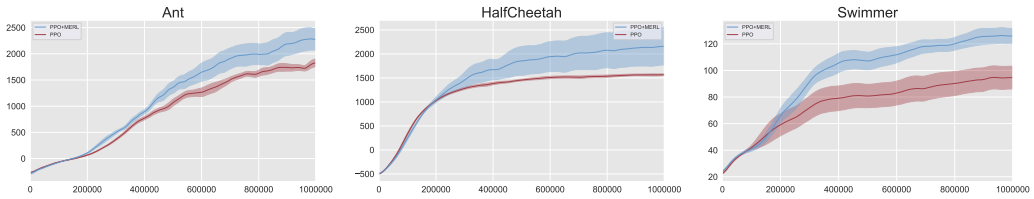

Figure 2: Experiments on 3 MuJoCo environments (10^6 timesteps, 7 seeds) with PPO+MERL. Red is the baseline, blue is with our method. The line is the average performance, while the shaded area represents its standard deviation.

its high-dimensional video input (size 210 x 160) and the discrepancy of tasks between games. To investigate the advantages of using MERL for transfer learning we choose a set of 6 different Atari games with an action space of 9, which is the average size of the action space in the Atari domain. This choice has two benefits: first, the neural network shared between the policy, the value function and MERL heads do not need to be further modified when performing transfer learning and second, the 6 games provide a diverse range of game-play while sticking to the same size of action space.

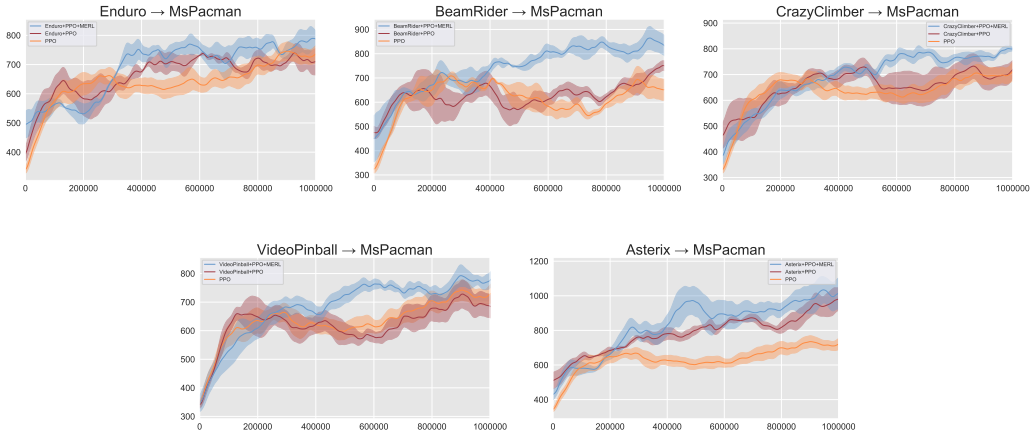

Figure 3: Transfer learning tasks from 5 Atari games to Ms. Pacman (2×10^6 timesteps, 4 seeds). Performance on the second task. Orange is PPO solely trained on Ms. Pacman, red and blue are respectively PPO and our method transferring the learning. The line is the average performance, while the shaded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

The results from Fig. 3 demonstrate that our method can reasonably adapt to a different task if we compare to the same method where MERL heads are not used. This result can be interpreted with the intuition 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 to the objective function through problem knowledge signals, those auxiliary quantities make the neural network optimize for task-agnostic 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 HalfCheetah, using only the $\mathrm{MERL}^{\mathrm{VE}}$ head degrades the performance, but when combining it 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.

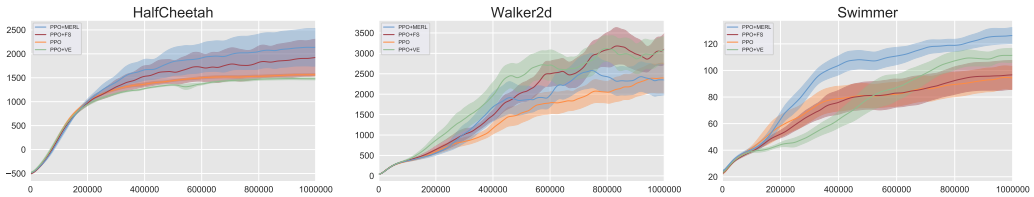

Figure 4: Ablation experiments with only one MERL head (FS or VE) (10^6 timesteps, 4 seeds). Blue is MERL with the two heads, red with the FS head, green with the VE head and orange with no MERL head. The line is the average performance, while the shaded area represents its standard deviation.

4.5 Discussion

From the experiments, we see 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 something useful while improving in this task. At the same time, it also addresses the problem of the sparsity of rewards.

5 Conclusion

In this paper, we introduced MERL, a generally applicable deep reinforcement learning framework for problem-focused representations in contrast with many current reward-centric algorithms. The virtual agent is able to predict problem-solving quantities in a multi-head layer to better address reinforcement learning problems. Our framework improves the performance of PPO for continuous control *MuJoCo* tasks and *Atari 2600* games in transfer learning tasks.

With MERL, we inject environment-agnostic problem knowledge directly in the policy gradient optimization. The advantage of this framework is threefold. First, the agent learns a better representation for single-task learning, and that is generalizable to other tasks. The multi-head layer provides a more problem-focused representation to the function approximations, which is therefore not only reward-centric. Moreover, continuous problem-solving signals help to address the problem of reward sparsity. Second, MERL can be seen as being a hybrid model-free and model-based framework with a small and lightweight component for self-performance assessment and accurate expectations. MERL heads seem to introduce regularization to the function approximation. In addition, this results in better performance and improved transfer learning. Third, MERL is directly applicable to most policy gradient algorithms and environments; it does not need to be redesigned for different problems and can be extended with many other relevant problem-solving quantities.

Even if 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 his environment. We also believe that predicting complementary quantities related to the objective of a task is a worthwhile idea to explore more in supervised learning. Finally, the identification of additional MERL quantities (*e.g.* prediction of immediate reward, 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 works.

References

[1] Abbeel, P., Ng, A.Y.: Apprenticeship learning via inverse reinforcement learning. In: Proceedings of the twenty-first international conference on Machine learning. p. 1. ACM (2004)

[2] Bellemare, M.G., Naddaf, Y., Veness, J., Bowling, M.: The arcade learning environment: An evaluation platform for general agents. Journal of Artificial Intelligence Research **47**, 253–279 (2013)

[3] Burda, Y., Edwards, H., Pathak, D., Storkey, A., Darrell, T., Efros, A.A.: Large-scale study of curiosity-driven learning. arXiv preprint arXiv:1808.04355 (2018)

[4] Burda, Y., Edwards, H., Storkey, A., Klimov, O.: Exploration by random network distillation. In: International Conference on Learning Representations (2019)

[5] Clouse, J.A., Utgoff, P.E.: A teaching method for reinforcement learning. In: Machine Learning, pp. 92–101. Elsevier (1992)

[6] Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., Doron, Y., Firoiu, V., Harley, T., Dunning, I., et al.: Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In: International Conference on Machine Learning. pp. 1406–1415 (2018)

[7] Flet-Berliac, Y., Preux, P.: Samples are not all useful: Denoising policy gradient updates using variance. arXiv preprint arXiv:1904.04025 (2019)

[8] Garcıa, J., Fernández, F.: A comprehensive survey on safe reinforcement learning. Journal of Machine Learning Research **16**(1), 1437–1480 (2015)

[9] Ha, D., Schmidhuber, J.: Recurrent world models facilitate policy evolution (2018)

[10] Islam, R., Henderson, P., Gomrokchi, M., Precup, D.: Reproducibility of benchmarked deep reinforcement learning tasks for continuous control. arXiv preprint arXiv:1708.04133 (2017)

[11] Iyer, R., Li, Y., Li, H., Lewis, M., Sundar, R., Sycara, K.P.: Transparency and explanation in deep reinforcement learning neural networks. In: Proceedings of AAAI/ACM Conference on Artificial Intelligence, Ethics, and Society. AAAI/ACM (2018)

[12] Jaderberg, M., Mnih, V., Czarnecki, W.M., Schaul, T., Leibo, J.Z., Silver, D., Kavukcuoglu, K.: Reinforcement learning with unsupervised auxiliary tasks. arXiv preprint arXiv:1611.05397 (2016)

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

[14] Li, X., Li, L., Gao, J., He, X., Chen, J., Deng, L., He, J.: Recurrent reinforcement learning: a hybrid approach. In: International Conference on Learning Representations (2016)

[15] Lin, L.J.: Self-improving reactive agents based on reinforcement learning, planning and teaching. Machine learning **8**(3-4), 293–321 (1992)

[16] Lipton, Z.C., Azizzadenesheli, K., Kumar, A., Li, L., Gao, J., Deng, L.: Combating reinforcement learning's sisyphean curse with intrinsic fear. arXiv preprint arXiv:1611.01211 (2016)

[17] Mirowski, P., Pascanu, R., Viola, F., Soyer, H., Ballard, A.J., Banino, A., Denil, M., Goroshin, R., Sifre, L., Kavukcuoglu, K., et al.: Learning to navigate in complex environments. arXiv preprint arXiv:1611.03673 (2016)

[18] Moreno, D.L., Regueiro, C.V., Iglesias, R., Barro, S.: Using prior knowledge to improve reinforcement learning in mobile robotics. In: Proceedings Towards Autonomous Robotics Systems (2004)

[19] Nachum, O., Norouzi, M., Xu, K., Schuurmans, D.: Bridging the gap between value and policy based reinforcement learning. In: Advances in Neural Information Processing Systems. pp. 2775–2785 (2017)

[20] Nguyen, D., Widrow, B.: The truck backer-upper: An example of self-learning in neural networks. In: Advanced Neural Computers. pp. 11–19 (1990)

[21] Oudeyer, P.Y., Kaplan, F.: What is intrinsic motivation? a typology of computational approaches. Frontiers in Neurorobotics **1**, 6 (2007)

[22] Pathak, D., Agrawal, P., Efros, A.A., Darrell, T.: Curiosity-driven exploration by self-supervised prediction. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. pp. 16–17 (2017)

[23] Pomerleau, D.A.: Alvinn: An autonomous land vehicle in a neural network. In: Advances in neural information processing systems. pp. 305–313 (1989)

[24] Ribeiro, C.H.: Embedding a priori knowledge in reinforcement learning. Journal of Intelligent and Robotic Systems **21**(1), 51–71 (1998)

[25] Robinson, T., Fallside, F.: Dynamic reinforcement driven error propagation networks with application to game playing. In: Conference of the Cognitive Science Society. pp. 836–843 (1989)

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

[27] Schmidhuber, J.: Curious model-building control systems. In: IEEE International Joint Conference on Neural Networks. pp. 1458–1463. IEEE (1991)

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

[29] Schulman, J., Levine, S., Abbeel, P., Jordan, M.I., Moritz, P.: Trust region policy optimization. In: International Conference on Machine Learning. pp. 1928–1937 (2015)

[30] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347 (2017)

[31] Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al.: Mastering the game of Go with deep neural networks and tree search. Nature **529**, 484 (2016)

[32] Suddarth, S., Kergosien, Y.: Rule-injection hints as a means of improving network performance and learning time. Neural Networks pp. 120–129 (1990)

[33] Sutton, R.S., Barto, A.G.: Introduction to reinforcement learning. Cambridge: MIT Press (1998)

[34] Sutton, R.S., McAllester, D.A., Singh, S.P., Mansour, Y.: Policy gradient methods for reinforcement learning with function approximation. In: Advances in Neural Information Processing Systems. pp. 1057–1063 (2000)

[35] Todorov, E., Erez, T., Tassa, Y.: Mujoco: A physics engine for model-based control. In: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 5026–5033. IEEE (2012)

[36] Werbos, P.J.: Neural networks for control and system identification. In: IEEE Conference on Decision and Control,. pp. 260–265 (1989)

[37] Whitehead, S.: Complexity and cooperation in q-learning. In: Eighth International Workshop on Machine Learning. pp. 363–367 (1991)