A Joint Imitation-Reinforcement Learning Framework for Reduced Baseline Regret

Sheelabhadra Dey\textsuperscript{1}, Sumedh Pendurkar\textsuperscript{1}, Gunii Sharon\textsuperscript{1} and Josiah P. Hanna\textsuperscript{2,3}

Abstract—In various control task domains, existing controllers provide a baseline level of performance that—though possibly suboptimal—should be maintained. Reinforcement learning (RL) algorithms that rely on extensive exploration of the state and action space can be used to optimize a control policy. However, fully exploratory RL algorithms may decrease performance below a baseline level during training. In this paper, we address the issue of online optimization of a control policy while minimizing regret with respect to a baseline policy performance. We present a joint imitation-reinforcement learning framework, denoted JIRL. The learning process in JIRL assumes the availability of a baseline policy and is designed with two objectives in mind (a) training while leveraging demonstrations from the baseline policy to minimize regret with respect to the baseline policy, and (b) eventually surpassing the baseline performance. JIRL addresses these objectives by initially imitating the baseline policy and gradually shifting control from the baseline to an RL agent. Experimental results show that JIRL effectively accomplishes the aforementioned objectives in several, continuous action-space domains. The results demonstrate that JIRL is comparable to a state-of-the-art algorithm in its final performance while incurring significantly lower baseline regret during training. Moreover, the results show a reduction factor of up to 21 in baseline regret over a trust-region-based approach that guarantees monotonic policy improvement.

I. INTRODUCTION

Deep reinforcement learning (RL) can produce policies that perform at, and even surpass, human-level control in various domains \cite{1}. As such, one might wonder why is deep RL not ubiquitously used to automate everyday tasks such as driving, traffic management, or medical procedures? For such domains, it is necessary that the performance of any control policy is at least as good as the policy currently under operation and, ideally, improves upon it. Current RL algorithms, however, cannot provide such guarantees for the general case (unless assuming specific domain knowledge \cite{2}). As they possess no conceptual model of the world to begin with, such algorithms must perform extensive exploration, i.e., sampling different actions in various situations (world states). During exploration, the outcomes of different actions in different states are learned and the control function (denoted as ‘policy’) is updated accordingly. For example, consider an inefficient traffic signal controller at an intersection. Improving the controller’s efficiency (e.g., \textit{w.r.t} vehicle throughput) is desired however we should never allow the controller to perform significantly worse compared to the currently deployed controller as this might result in an abnormal cascading affect that can jam an entire city.

In RL, performance degradation over some baseline controller is commonly measured through the reduction in the accumulated reward and is denoted as \textit{baseline regret} \cite{3}, \cite{2}. Our main contribution is in proposing an approach for optimizing a control policy while minimizing baseline regret \textit{w.r.t} a given baseline controller.

We propose the joint imitation-reinforcement learning (JIRL) framework for baseline regret minimization which utilizes a suboptimal policy that could be of any nature (deterministic/stochastic/rule-based/human-operated) while learning and applying an improved policy. Under this framework, a baseline policy and an RL policy jointly select actions. If the RL agent’s action substantially differs from the baseline action then the baseline action is applied, otherwise the RL agent’s action is applied. The RL policy is then updated with an off-policy learning algorithm while divergence from the baseline is penalized. Gradually, as the RL agent’s policy improves, it is allowed to take actions that further diverge from the baseline. This procedure allows the RL controller to gradually find an optimal policy while discouraging highly-exploratory actuation. Note, however, that JIRL does not provide any guarantees regarding the resulting baseline regret. This is to be expected as providing such a guarantees is challenging without making specific assumptions about the environment. Nonetheless, experimental results show a clear trend where the JIRL framework can fully train a real-world RC car in 45 minutes leading to a 30% improvement in performance \textit{w.r.t} a baseline policy while minimizing baseline regret.

II. PROBLEM DEFINITION

We assume a Markov Decision Process \cite{4} with state space, \(S\), action space, \(A\), transition probabilities, \(P\), reward function, \(R: S \times A \mapsto \mathbb{R}\), and discount factor, \(\gamma\). An RL agent is assumed to start from state \(s_0\) and select action \(a_0\) according to a policy, \(\pi: S \mapsto A\). \(\pi\) might be stochastic, i.e., mapping states to a distribution over actions. \(\pi\) can be defined by a function approximator with a parameter set \(\theta\) and is denoted \(\pi_\theta\) in such cases. Based on the chosen action, the agent receives a reward \(r_\theta\) from the environment and reaches the next state, \(s_1\), according to the transition probability, \(P(s_1|s_0, a_0)\). The process repeats and generates a trajectory, \(\tau := (s_0, a_0, r_\theta, s_1, a_1, r_\theta, s_2, \ldots)\).
In addition to the standard MDP formulation, we assume an available baseline policy.

**Assumption 1 (Baseline Policy).** There exists a baseline policy, $\pi_0$, and at every time-step we can observe the action that $\pi_0$ would take in the current state.

The baseline policy is not assumed to be optimal or exploratory (i.e., stochastic). That is, $\pi_0$ can be a deterministic (e.g., rule-based) suboptimal controller. An example of such a controller is traffic signal controllers in modern intersections.

**Objective:** Learn a parameterized policy that maximizes the sum of discounted rewards in an expected trajectory. That is, minimize baseline regret, neither approach is sufficient as common RL algorithms perform extensive exploration and behavior cloning might learn from suboptimal demonstrations, failing to optimize $J(\pi_0)$. Our proposed JIRL framework attempts to close this gap and merge the two approaches in a way that combines the best of each.

**A. Bounding the Baseline Regret**

Our JIRL framework does not provide any bounds on the accumulated baseline regret. This is, however, to be expected. In the general case i.e., with no additional assumptions regarding the provided MDP or baseline policy, the baseline regret cannot be bounded by a scalar. This claim follows from the fact that no regret bounds (including baseline regret) can be provided for the multi-armed bandit problem in the general case (see Bubeck et al. [7] for proof), and any multi-armed bandit instance (P1) can be reduced to an MDP (P2).

The relevant reduction mapping constructs an MDP with a single state, $s_0$, in P2 where every bandit from P1, $b_i$, becomes an action at P2, $a_i$. The transition probabilities are $\forall a \ P(s_0|s_0, a) = 1$ and the reward, $R(s_0, a_i)$, for any action, $a_i$, in P2 follows the utility distribution from playing $b_i$ in P1. For such a construction, it is easy to see that the baseline regret from following any policy in P1 over any baseline policy equals the baseline regret for equivalent policies in P2.

**III. RELATED WORK**

A line of previous work did provide some guarantees regarding baseline regret. These, however, do not stand in contradiction to our previous claim regarding the infeasibility of regret bounds as these works all rely on some simplifying assumptions. Approaches assuming extensive baseline exploration relied on a stochastic baseline policy to sample high-variance trajectories. These trajectories were used to create a batch of data and then employed offline-RL or batch-RL [8] algorithms for policy improvement [9], [10]. However, policy improvement is guaranteed only if the baseline policy executes an optimized trajectory with non-zero probability (baseline policy coverage) and the data generated is stationary. Ghavamzadeh et al. [2] additionally learned a model of the environment but assumed access to the error in the model estimation. JIRL, by contrast, can learn from a deterministic baseline policy and avoids learning a model due to the inaccuracies associated with model estimation without extensive exploration.

Adding safety constraints can prevent an agent from diverging from a baseline controller during and after training [11], [12], [13]. However, ensuring such safety requires that for any achievable state, at least one safe action can be taken or the availability of a model that can accurately designate unavoidable safety violations following a given state-action pair. Algorithms that assume the availability of a safe action commonly rely on shielding [14], action correction [15], [16], [17], [18], or ergodic MDPs [19]. These approaches generally require domain knowledge about which actions will lead to constraint violations. Methods relying on a model assume that the model is available beforehand [20], [21] while others learn the model online [15], [17], [22], [23]. JIRL, by contrast, doesn’t assume access to explicit safety constraints or the model of the environment.

In a different line of work, a number of approaches have aimed to ensure safe policy updates to improve upon a baseline policy on tabular problems [24], [25]. By contrast, JIRL is evaluated on continuous action tasks where the value function and policy are parametrized via a function approximator (neural network). Trust region policy optimization (TRPO) [26] and proximal policy optimization (PPO) [27] update stochastic policies by taking the largest step possible to improve performance, while satisfying a constraint on the KL-Divergence between the new and old policies. Since, TRPO and PPO approximate a monotonic improvement in performance, initializing TRPO (or PPO) with a baseline policy is a suitable candidate for comparison against JIRL in terms of the baseline regret. In the TRPO/PPO framework, provided the domain allows stochastic policies, the initial stochastic baseline policy can be learned through observations (e.g., using imitation learning). Such an approach, however, requires a – potentially expensive – initial imitation learning phase.
IV. THE JOINT IMITATION-REINFORCEMENT LEARNING FRAMEWORK

In this section, we introduce our main contribution, the joint imitation-reinforcement learning framework (JIRL). JIRL extends over previous imitation and reinforcement learning algorithms by (1) generalizing the notion of penalized rewards [28], [29] to apply to continuous action spaces, (2) enabling learning from a baseline policy that can be queried during the training phase, and (3) defining a criterion for determining whether the RL agent or baseline policy should be given control at a particular time-step.

The JIRL framework is detailed in Algorithm 1. At each timestep, the baseline policy is assumed to provide a suggested action to take, \( a^b_t \), that is derived from its internal policy (Line 3). Next, the RL agent determines if it should follow its own (stochastic) policy at the current state or defer to the baseline action (Line 4). If the RL policy is followed, \( a^\ell_t \) is sampled from \( \pi_\theta \) and applied to the environment; otherwise, the baseline action, \( a^b_t \), is applied. Regardless of the action applied to the environment, the RL agent is trained on the observed outcome (given as a full transition \( (s_t, a_t, r_t, s_{t+1}) \), Line 10). Note that training the RL agent on transitions originating from the baseline policy requires an off-policy RL learning procedure. In cases where the baseline action \( a^b_t \) was applied, we train the RL agent on a fabricated, counterfactual transition in which the RL agent took \( a^\ell_t \) and ended up in the same state that resulted from \( a^b_t \) (Line 15). The reward for this fabricated transition — obtained through the function \( \text{PenalizeReward()} \) — is set to be lower than the reward affiliated with the baseline policy action. Doing so ensures that an RL agent (aiming to maximize return) will update its (stochastic) policy to shift probability towards \( a^b_t \) from \( a^\ell_t \), i.e., towards imitating the baseline and reducing future baseline regret.

JIRL is, thus, a general framework that can work on top of any off-policy RL algorithm with a stochastic parameterized policy that implements \( \text{TrainRL()} \). For example, an actor-critic algorithm [30] can implement \( \text{TrainRL}(\text{transition}) \) as: store \( \text{transition in a replay buffer, periodically train both the actor and critic using stochastic gradient descent.} \) JIRL, nonetheless, requires a specific implementation for \( \text{PenalizeReward()} \) in Line 15 and \( \text{SafeForRL()} \) in Line 4. These two functions determine the interplay between the imitation and reinforcement learning within JIRL and are discussed next.

A. Penalized reward for continuous actions

Following Hester et al. [28], the reward for the fabricated transition (Line 15 in Algorithm 1) is penalized such that the RL agent will be trained towards imitating the baseline policy. We do so by introducing the fabricated transition, \( (s_t, a^\ell_t, r^\ell_t, s_{t+1}) \), where \( r^\ell_t < r_t \). When considering continuous action spaces, setting a constant penalty, \( \ell \), such that \( R^\ell = r_t - \ell \) [28] will result in a non-smooth reward function as \( \lim_{a \rightarrow a^b} R^\ell(s_t, a) \neq R^\ell(s_t, a^b) \) where \( R^\ell \) is the penalized reward function. As a result, an imitation learning process that uses fabricated transitions might fail to converge on the baseline policy. In order to address this issue, we present a continuous penalized reward function that is based on a Gaussian function. The penalized reward computation is presented in Algorithm 2. In this case, the penalized reward tends toward zero as \( a^\ell_t \) tends towards \( a^b_t \). The hyperparameter representing the penalty function variance is chosen, for a given domain, based on the required action precision (smaller variance = more precise imitation).

Note that for domains with a discrete action space, the penalty in Algorithm 2 should follow Piot et al. [31], i.e., implemented as a margin function that is 0 when \( a^\ell_t = a^b_t \) and positive otherwise.

---

**Algorithm 1:** Joint Imitation-Reinforcement Learning

**Input:** baseline policy, \( \pi_\theta \), maximum number of training steps, \( L \), off-policy RL algorithm, \( \text{TrainRL()} \)

**Output:** optimized policy

**Initialize:** RL policy parameters, \( \theta \)

1. \( s_0 \leftarrow \) Reset environment;
2. for \( i = 0 \) to \( L \) do
3. \( a^\ell_t \leftarrow \pi_\theta(s_t); \# \) Baseline’s action
4. if \( \text{SafeForRL}(s_t, a^\ell_t, \pi_\theta, t) \) then
5. \( a_t \sim \pi_\theta(\cdot|s_t); \# \) RL action
6. else
7. \( a_t \leftarrow a^\ell_t; \)
8. end
9. \( s_{t+1} \sim p(s_{t+1}|s_t, a_t); \)
10. \( \text{TrainRL}(s_t, a_t, r_t, s_{t+1}); \)
11. if \( a_t = a^b_t \) then
12. \( \# \) Fabricated (penalized) transition for \( a^\ell_t \)
13. \( a^\ell_t \leftarrow \pi_\theta(\cdot|s_t); \)
14. \( r^p = \text{PenalizeReward}(r_t, a^\ell_t, a^\ell_t); \)
15. \( \text{TrainRL}(s_t, a^\ell_t, r^p, s_{t+1}); \)
16. end
17. if \( s_{t+1} \) is terminal then
18. \( s_{t+1} \leftarrow \) Reset environment;
19. end
20. end
21. return \( \pi_\theta \)

---

**Algorithm 2:** Penalized reward for fabricated transitions

**Hyperparameters:** penalty variance, \( \sigma^2 \)

**Input:** reward, \( r_t \), baseline policy’s action, \( a^b_t \), RL action, \( a^\ell_t \)

**Output:** penalized reward value

1. Function \( \text{PenalizeReward}(r, a^b_t, a^\ell_t); \)
2. \( \tilde{r} \leftarrow |r_t| \left( 1 - \exp \left( -\|a^b_t - a^\ell_t\| / \sigma^2 \right) \right); \)
3. \( r^p \leftarrow r - \tilde{r}; \)
4. return \( r^p \)
B. RL control criteria

There is a balance to strike between imitating the baseline policy and taking exploratory actions so as to eventually outperform the same (presumably suboptimal) baseline. We address this balance through the function \texttt{SafeForRL()} through which the RL agent determines whether to act and explore or follow the baseline action and train towards imitating the baseline policy. The intuition behind the selection criterion in \texttt{SafeForRL()} is that when the RL agent is uncertain about its actions (measured as high entropy policy), it should be trained towards imitating the baseline policy. As the RL agent becomes more certain regarding its actions, it is allowed to drift further away from the baseline policy and explore other promising actions. We define the divergence value (denoted $\rho$) as the summation of the RL policy entropy term with the norm distance between the expected RL action and the baseline policy action.\footnote{For discrete action spaces, the norm distance between two actions, $a^b, a^b$, can be defined as some constant if $a^b \neq a^b$, else zero.}

Our proposed approach for determining which policy to follow is inspired by the control criteria that was presented in Menda et al. \cite{Menda2021}. Our suggested criteria is summarized in Algorithm 3. It extends the criteria from Menda et al. \cite{Menda2021} by considering divergence over consecutive time steps. Factoring the divergence values over several timesteps is important as small divergences can accumulate over time and result in a significant divergence. We observed this phenomenon when applying JIRL to an autonomous driving domain where the RL policy can slowly, yet steadily, steer the vehicle off the road. As a result, the divergence condition in Line 6 is factored over the minimum between the number of steps since the last RL-baseline control switch and a hyper-parameter $K$. An RL-baseline control switch occurs at time step $t$ if $(a_t = a^b_t$ and $a_{t-1} = a^d_{t-1})$ or $(a_t = a^d_t$ and $a_{t-1} = a^b_{t-1})$. $K$ is chosen appropriately for each domain. For domains where it is relatively easy to recover back to states the baseline would visit, e.g., Inverted Pendulum (see Section V-A), $K$ should be set lower. For domains that allow a chain of subtle divergences to lead to states that the baseline would not visit, e.g., lane following in autonomous driving, $K$ should be set higher.

In some domains, it may be unreasonable to assume a baseline policy that provides an action at every time-step. In such domains, JIRL can still be applied provided that the baseline policy can intervene when safety is compromised. Autonomous vehicles with an expert safety driver are an example of such a domain. Such cases require minimal change to Algorithm 1 where if the baseline policy does not intervene (i.e., no action is provided) then $d^b_t \leftarrow \text{Null}$ in Line 3, and Algorithm 3 is simply implemented as return $d^b_t$ equals Null.

C. Discussion

It is important to note that, unless making specific assumptions regarding the implementation of \texttt{SafeForRL()}, the JIRL framework provides no bounds on the amount of baseline regret incurred.

Instead, JIRL provides a trade-off between baseline regret and allowable exploration towards finding an optimized final policy. A lower value for the penalty variance (in Algorithm 2) encourages imitating the baseline – reducing baseline regret but discouraging finding new, more optimal behaviors. Similarly, a low scaling factor, \( C \), will allow the RL policy to take actions more frequently – promoting exploration at the expense of possible higher baseline regret.

The underlying RL algorithm, implementing \texttt{TrainRL()}, is constantly being fed with a mixture of transitions generated from either the baseline, RL agent, or fabricated, penalized transitions. There is an important distinction to make between these transitions. RL transitions can be utilized for on-policy learning towards the optimal policy. Baseline induced transitions can only be utilized for off-policy learning and the optimal policy can only be learned if coverage \([5, \text{Chapter 5}]\) is assumed. Fabricated transitions can be utilized for on-policy learning as they are composed of actions sampled from the RL policy. However, since the affiliated reward is fabricated, such transitions may bias learning away from the optimal policy. As a result, convergence to the optimal policy can be guaranteed if (1) the underlying RL algorithm provides such guarantees, and (2) only RL induced transitions are considered eventually. As training progresses, the accumulated safe divergence value is expected to decrease since the RL policy entropy usually decreases (depending on the underlying RL algorithm). Consequently, RL control becomes more common and baseline/fabricated transitions are encountered less. As a result, learning the optimal policy requires an RL algorithm that “forgets” older transitions. Such a forgetting attribute is common in RL algorithms that use a bounded replay buffer \([30] \) which are, thus, particularly suitable for JIRL.

| Algorithm 3: Should we follow the RL policy? |
|---------------------------------------------|
| **Hyperparameters:** scaling factor, \( C \), number of steps to consider, \( K \) |
| **Input:** state, \( s \), baseline’s action, \( d^b \), RL policy, \( \pi_\theta \), time step, \( t \) |
| **Output:** True iff the RL policy should be followed at the current state |
| 1 Function \texttt{SafeForRL}(s, d^b, \pi_\theta, t) : |
| 2 \( h \leftarrow \mathcal{H}(\pi_\theta(|s|)) \); \# Entropy of the RL policy at the current state |
| 3 \( d \leftarrow \|d^b - \mathbb{E}_{\pi_\theta}[a]\| \); \# Distance between the RL expected action and the baseline’s action |
| 4 \( \rho_t \leftarrow h + d; \) |
| 5 \( k \leftarrow \text{minimum between } K \text{ and number of steps since last control switch}; \# \text{For } k > 1 \text{ we assume that the input, } s, \text{ includes the last } k \text{ states} \) |
| 6 \( \text{followRL} \leftarrow \prod_{i=1-k} \pi_\theta(a^b_i | s_i) > C \prod_{i=1-k} \rho_i; \) |
| 7 return followRL |
V. EXPERIMENTS

The goal of our experiments is to evaluate the effectiveness of JIRL in continuous control tasks with respect to the following objectives: (a) leveraging the baseline’s online demonstrations to reduce the regret w.r.t the baseline policy during training, and (b) eventually surpassing the baseline performance. Specifically, we aim to show that (1) applying JIRL on top of a state-of-the-art RL algorithm results in significant reduction of the baseline regret while not degrading the final performance, and (2) JIRL outperforms a straightforward approach for eliminating baseline regret, which is, applying TRPO and PPO over the baseline policy (assuming a stochastic version is available or can be learned).

A. Domain Description

Figure 1 shows snapshots from the domains used in our experiments. The goal in the Inverted pendulum task from the OpenAI gym [33] is to train an agent to swing up a pendulum and keep it at an upright position. In the Lunar lander task from the OpenAI gym, the objective is to train a space probe to land on a landing pad without crashing. The goal in the Lane following (LF) task is to train an autonomous vehicle to drive around a custom track following a lane. Using images from a front-facing camera as input, we adopt the training set-up used in Raffin et al. [34]. This domain is evaluated both in the CARLA simulator [35] and on a Waveshare JetRacer which is an autonomous scaled car that uses NVIDIA’s Jetson Nano as the main control platform. In the Walker-2D task from the PyBullet environment [36], a 2-legged robot learns to stay upright and walk. Note that, for this domain, baseline regret can be significantly reduced when setting a higher $K$ value (as discussed in Section IV-B). However, doing so also results in considerably slower learning. We observed that values of $K$ between $5 - 10$ and $1 - 3$ achieve an acceptable trade-off between reducing the baseline regret and training time for the Lane following domain and the Walker-2D domain respectively. For the Inverted pendulum and Lunar lander domains, all values of $K$ between $1 - 5$ gave similar results in terms of reducing the baseline regret. $\sigma^2$ values from the range $[0.01 - 0.1]$ resulted in good performance (similar to those reported) in all domains.

We observed that assigning a fixed penalty of $-1$ to transitions that resulted in the baseline given control (when the RL agent is deemed unsafe to act), yields slightly faster learning. Such events represent an unsafe divergence of the RL policy from the baseline and are, thus, penalized. Such a penalty was used for obtaining the reported results.

B. Baseline Policies

For the Inverted pendulum, Lunar lander, Lane following (JetRacer) and Walker-2D tasks, suboptimal deterministic baseline policies were defined in order to demonstrate that JIRL can learn from and outperform such a policy. The baseline policies for all the domains except the Lane following tasks were learned by training an agent from scratch using soft actor-critic (SAC) [30] with the help of early stopping and reward shaping to induce a suboptimal behavior. For the Lane following task, the baseline policies were obtained using supervised learning on image-action pairs that were collected from an expert human demonstrator. A detailed description of our experimental set-up and hyperparameters is available in a technical report that is available online at https://pi-star-lab.github.io/JIRL. The codebase for all the experiments is available at https://github.com/Pi-Star-Lab/JIRL.

C. Results

In all the following experiments, soft actor-critic (SAC) [30] was used as the underlying RL algorithm within JIRL. For each domain, we trained 5 instances of JIRL(SAC) and vanilla SAC with the same set of hyper-parameters for SAC (as specified in the technical report) and different random seeds per instance. We used the implementation of SAC provided in Stable Baselines [37]. Figure 2 shows the training curves for both approaches along with the TRPO/PPO baselines (when applicable).

In all four domains, JIRL(SAC) resulted in a final policy that is at or above the vanilla SAC algorithm. Moreover, JIRL is shown to clearly reduce the baseline regret over vanilla SAC (the baseline performance can be seen on the left side of the JIRL curve where RL control is at a minimum). These results support the claim that applying JIRL on top of a state-of-the-art RL algorithm results in significant reduction of the baseline regret while not degrading the final performance. In all but the Lane following (CARLA) domain, JIRL(SAC) resulted in a final policy that is superior to the baseline policy. The discrepancy in the Lane following (CARLA) domain is due to the use of a highly optimized baseline policy. SAC was unable to outperform the baseline performance when trained using a reward function similar to the one presented in Kendall et al. [18]. When adjusting the reward function to reward greater speeds, it is possible to surpass the baseline performance (human demonstrator) by driving faster as demonstrated in the Lane following (JetRacer) domain (see Figure 2e). The JetRacer clocked a lap-time of 13.5 seconds using the baseline
Fig. 2: (a)-(e) Reward and percentage of RL control using the JIRL framework on top of SAC, vanilla SAC, and the best between TRPO+IL or PPO+IL when applicable. For the TRPO+IL and PPO+IL curves the required initial IL phase is omitted. In all the subfigures, the x-axis is the number of environment steps, the y-axis on the left is the smoothed reward, and the y-axis on the right is the percentage of RL control in the JIRL framework. The shaded region represents the 95% confidence interval.

TABLE I: Comparison between the accumulated per step baseline regret in the JIRL framework and best of TRPO/PPO w.r.t the baseline performance. Results are averaged over 5 runs. The advantage of JIRL over TRPO/PPO is statistically significant in all the domains based on a paired t-test.

| Domain              | JIRL 0-50% | JIRL 50-100% | JIRL Full RL | JIRL Total | TRPO/PPO Total |
|---------------------|------------|--------------|--------------|------------|----------------|
| Inverted pendulum   | 3071       | 396          | 0            | 3467       | 37290          |
| Lunar lander        | 2044       | 819          | 0            | 2863       | 60744          |
| LF (CARLA)          | 13.6       | 178.4        | 0            | 192        | NA             |
| LF (JetRacer)       | 0.8        | 2.7          | 0            | 3.5        | NA             |
| Walker-2D           | 10         | 188          | 16616        | 16814      | 33039          |

VI. CONCLUSION

We introduce a joint imitation-reinforcement learning framework (JIRL) and demonstrate its ability to optimize a control policy while reducing regret with respect to an available baseline controller. Assuming a baseline controller that is available during the training process, JIRL periodically switches between the baseline policy’s actions and exploratory
actions from the RL agent. We define a control switching criterion that is based on the RL policy’s entropy and its divergence from the baseline policy’s actions accumulated over a series of timesteps. Moreover, we present a Gaussian penalty function for penalizing uncertain RL divergences from the baseline controller where the certainty levels are measured through the policy entropy. We also add another constant penalty for RL induced actions that lead to control switches. Doing so, decreases the number of actions that lead to lesser reward than the baseline would accrue. JIRL is shown to perform on par with the baseline during the learning process and eventually surpass a suboptimal baseline in all examined domains. Moreover, JIRL was shown to converge to a policy of similar quality (sum of discounted rewards) as a vanilla implementation of the underlying state-of-the-art RL algorithm (SAC) while reducing the accumulated baseline regret by up to $\times 21$.

REFERENCES

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al., “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, p. 529, 2015.

[2] M. G. Bellemare, M. Everitt, and A. Cheば, “Safe policy improvement by minimizing robust baseline regret,” in *Advances in Neural Information Processing Systems*, 2016, pp. 2298–2306.

[3] F. Wu, S. Zilerstein, and X. Chen, “Multi-agent planning with baseline regret minimization,” in *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*, 2017, pp. 444–450.

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

[5] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[6] S. Ross, G. Gordon, and D. Bagnell, “A reduction of imitation learning and structured prediction to no-regret online learning,” in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011, pp. 627–635.

[7] S. Bubeck, V. Perchet, and P. Rigollet, “Bound regret in stochastic multi-armed bands,” in *Conference on Learning Theory*, 2013, pp. 122–134.

[8] S. Lange, T. Gabel, and M. Riedmiller, “Batch reinforcement learning,” in *Reinforcement learning*. Springer, 2012, pp. 45–73.

[9] P. Thomas, G. Theocharous, and M. Ghavamzadeh, “High confidence policy improvement,” in *International Conference on Machine Learning*, 2015, pp. 2380–2388.

[10] R. Laroche, P. Trichclair, and R. T. Des Combes, “Safe policy improvement with baseline bootstrapping,” in *International Conference on Machine Learning*, PMLR, 2019, pp. 3652–3661.

[11] A. Geramifard, J. Redding, N. Roy, and J. P. How, “Uav cooperative control with stochastic risk models,” in *Proceedings of the 2011 American Control Conference*. IEEE, 2011, pp. 3393–3398.

[12] A. Geramifard, “Practical reinforcement learning using representation learning and safe exploration for large scale markov decision processes,” Ph.D. dissertation, Massachusetts Institute of Technology, 2012.

[13] J. Achiam, D. Held, A. Tamar, and P. Abbeel, “Constrained policy optimization,” 2017.

[14] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu, “Safe reinforcement learning via shielding,” in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[15] J. Zhang and K. Cho, “Query-efficient imitation learning for end-to-end autonomous driving,” *arXiv preprint arXiv:1805.06450*, 2016.

[16] W. Saunders, G. Sastry, A. Stuhlmüller, and O. Evans, “Trial without error: Towards safe reinforcement learning via human intervention,” *arXiv preprint arXiv:1707.05173*, 2017.

[17] G. Dalal, K. Dvijotham, M. Vecerik, T. Hester, C. Paduraru, and Y. Tassa, “Safe exploration in continuous action spaces,” *arXiv preprint arXiv:1801.08757*, 2018.