IEEE Robotics and Automation Letters (RAL) paper presented at
2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Adaptive Curriculum Learning with Successor Features for Imbalanced Compositional Reward Functions

Laszlo Szoke[1,2,+], Shahaf S. Shperberg[3], Jarrett Holtz[4,5], Alessandro Allievi[4,5]

***Abstract*—This work addresses the challenge of reinforcement learning with reward functions that feature highly imbalanced components in terms of importance and scale. Reinforcement learning algorithms generally struggle to handle such imbalanced reward functions effectively. Consequently, they often converge to suboptimal policies that favor only the dominant reward component. For example, agents might adopt passive strategies, avoiding any action to evade potentially unsafe outcomes entirely. To mitigate the adverse effects of imbalanced reward functions, we introduce a curriculum learning approach based on the successor features representation. This novel approach enables our learning system to acquire policies that take into account all reward components, allowing for a more balanced and versatile decision-making process.**

***Index Terms*—Reinforcement Learning, Continual Learning**

I. Introduction

In reinforcement learning (RL), agents interact with an environment to learn a policy that maximizes their cumulative rewards. In certain domains, the environment's reward function comprises multiple components collectively aggregated as a weighted sum. For instance, in autonomous driving, the reward may encompass factors such as distance traveled, adherence to traffic regulations, collision avoidance, fuel efficiency, passenger comfort, and other considerations.

A significant challenge emerges when certain components within a reward function carry substantially greater weight than others. For example, scenarios where the negative impact of a collision is orders of magnitude higher than the positive reward for making progress. Typically, RL algorithms tasked with handling such imbalances tend to converge toward policies that heavily favor the dominant components of the reward, often at the expense of completely disregarding other crucial factors.

Manuscript received: November, 15, 2023; March, 28, 2024; April, 2, 2024.
This paper was recommended for publication by Editor Jens Kober upon evaluation of the Associate Editor and Reviewers' comments.

+ corresponding author: szoke.laszlo95@gmail.com,
[1]Budapest University of Technology and Economics, [2]Robert Bosch Kft, Hungary, [3]Ben-Gurion University, Israel, [4]Robert Bosch LLC. [5]University of Texas at Austin, USA

*Laszlo Sz. is a beneficiary of Project C1484516. Project no. C1484516 has been implemented with the support provided by the Ministry of Culture and Innovation of Hungary from the National Research, Development, and Innovation Fund, financed under the KDP-2021 funding scheme. The work at Ben-Gurion University was supported by ISF grant #909/23 and by MOST grant #1001706842, awarded to Shahaf Shperberg. Special thanks to Brad Knox for providing the original idea of a successor feature based reward curriculum and to both Brad Knox and Peter Stone for their invaluable collaboration, recommendations, and support throughout this work.
Digital Object Identifier (DOI): see top of this page.

In the context of autonomous driving, consider algorithms that directly optimize a reward function that endeavors to strike a balance between distance-driven and collision penalties. For instance, algorithms that impose a collision penalty equivalent to the cumulative reward of 1,000,000 kilometers driven [1]. These algorithms often converge toward a degenerate policy where the vehicle remains stationary to evade the substantial negative penalties associated with collisions. Another example is a video game, where progress serves as the primary reward, but smaller rewards are also gained from acquiring collectibles. Traditional RL algorithms may face challenges in prioritizing the collection of these items, especially if they are challenging to obtain and are sparse, as progressing through the game will dominate the reward.

To tackle the issue of learning from imbalanced rewards, we leverage successor features (SFs), a value function representation that decouples environmental dynamics from rewards. Within the SF framework, the weights assigned to different reward components are embodied as a preference vector. This allows adapting preference values to reflect varying importance levels among components without necessitating a complete relearning of the environment's dynamics.

We introduce *Preference-Vector Curriculum* (`PrefVeC`), an approach that dynamically evolves the preferences throughout the training process. `PrefVeC` starts by assigning weights that prioritize the least influential reward component. Through iterative adjustments, it progressively increases the weights of dominant components, steering them towards gradual alignment with the actual, imbalanced reward function during training. Consequently, `PrefVeC` enhances the performance of RL agents when confronted with imbalanced reward functions.

The SFs framework facilitates curriculum learning through three key mechanisms:

1) Knowledge Transfer. SFs provide a natural avenue for transferring knowledge between policies corresponding to different reward weights.
2) Adaptive Weight Adjustment. SFs provide an effective means of determining the next preference value within the curriculum. This can be achieved by calculating a set of weights that would yield a specified percentage of changes in actions, as derived from the current policy and computed based on the experienced trajectories recorded in the buffer.
3) Policy Combination. SFs enable the integration of all policies learned throughout the curriculum into a comprehensive policy that outperforms each individual policy,

as opposed to merely selecting the last trained policy. Policies learned using SFs can be evaluated with weights that differ from those used during training. Consequently, the agent can assess all policies at each state using the true set of weights and select the one with the best expected value.

We evaluate the effectiveness of our approach across three domains: a basic GridWorld and two driving scenarios. Our evaluations demonstrate that `PrefVeC` learns policies that better avoid the pitfall of optimizing for only one dominant reward component compared to baseline methods that directly optimize imbalanced reward functions.

II. Related Work

A. Reward Decomposition and Successor Features

Reward decomposition has garnered attention in the reinforcement learning (RL) community, with the aims of enhancing sample efficiency [2], [3], fostering explainability [4], and promoting cooperation in multi-agent systems [5]. These approaches involve learning distinct Q-values for each reward component and subsequently aggregating these values to determine the optimal action for each state. For example, [6], [7] decompose the reward function into positive and negative reward components. The method involves learning Q-values separately for each reward component, Q^+, and Q^-, respectively. During deployment, the agent considers a Q-value that is a linear combination of the positive and negative Q-values with respect to a parameter w: $Q^r(s,a) = w \cdot Q^+(s,a) + (1-w) \cdot Q^-(s,a)$. This approach has shown superior performance compared to other reward decomposition and standard Q-value-based RL methods. However, it addresses neither more than two reward components nor imbalanced rewards.

A prominent line of research in reward decomposition centers around successor features (SFs) [8]. SFs are an extension of the successor representation [9] that separates the environment dynamics from the rewards. This separation enables knowledge transfer between tasks that share the same environment but optimize different reward functions. Recent advancements in this area include the introduction of FastRL by Barreto et al. [10], a method that generalizes policy improvement and policy evaluation for SFs to enhance sample efficiency when learning new tasks. Szoke et al. [11] further enhanced FastRL by introducing stabilizing target networks and fostering information sharing among policies. Other notable research directions involve employing mutual information methods to learn reward decomposition [12], extending SFs to handle reward functions that cannot be linearly decomposed [13] [14], bounding the performance of SFs-based policies during task transfers [15], and leveraging SFs for knowledge transfer across environments with varying dynamics [16]–[19]. Furthermore, SFs have found application in multi-agent systems to enhance collaboration and coordination among agents [20]–[22]. SFs decomposition handles multiple reward components but lacks a solution for learning with imbalanced rewards.

Imbalanced reward functions often arise when agent safety takes precedence, as prioritizing safety is typically deemed vastly more critical than other objectives. Gimelfarb et al. [23] introduced risk-aware successor features (SFs) to transfer safety-related insights between tasks. In contrast to our approach, this method encodes safety as the variance of rewards rather than integrating it directly into the reward function. It's worth noting that this approach is designed explicitly for safety-related tasks by encouraging policies that result in low variance. Consequently, it is not readily applicable to general cases of imbalanced reward functions.

B. Curriculum Learning in RL

In Curriculum Learning (CL) [24] [25], the objective is to craft curricula tailored to a specific target task or distribution of tasks for the agent's training. The strategic use of curricula can significantly enhance the agent's performance compared to direct training on the target task or distribution.

Within the Reinforcement Learning (RL) community, CL offers versatile methods for modifying the learning tasks and their presentation order to the agent [26]. Noteworthy approaches include decomposing challenging tasks into simpler missions [27], prioritizing transitions for policy updates [28], controlling initial or goal states [29] [30], and altering reward functions or transition dynamics [31].

To the best of our knowledge, this work is the first endeavor to address the challenge of learning with imbalanced reward functions. Moreover, it marks the first utilization of reward decomposition and the successor features (SF) representation to facilitate curriculum learning.

III. Background on Successor Features

A Markov Decision Process (MDP) is a framework for modeling decision-making often employed to formulate RL problems. An MDP is a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, T, D_0, \gamma, R)$ in which $\mathcal{S}$ is the state space, D_0 is the start state distribution and γ is the discount factor. In addition, T is the transition function, such that $T(\cdot|s,a)$ is the next-state distribution after taking action a from state s, and $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the reward function. The agent's objective is to learn a policy $\pi : \mathcal{S} \to \mathcal{A}$ that optimizes the discounted cumulative reward:

$$J(\pi) = \mathop{\mathbb{E}}_{\substack{s_0 \sim D_0,\ a_t \sim \pi(s_t),\\ s_{t+1} \sim T(\cdot|s_t,a_t)}} \Big[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1}) \Big]. \tag{1}$$

The action-value function Q associated with a policy π can be derived from the optimization target in Eq. (1)

$$Q_\pi(s,a) = \mathop{\mathbb{E}}_{\substack{a_t \sim \pi(s_t),\\ s_{t+1} \sim T(\cdot|s_t,a_t)}} \Big[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1}) | s_0 = s, a_0 = a \Big] \tag{2}$$

The successor features (SFs) representation, introduced by Barreto et al. [8], involves mapping actions and states to a set of features through the function $\phi : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto \mathbb{R}^d$. In this formulation, reward functions are defined by a preference vector $\mathbf{w} \in \mathbb{R}^d$, as expressed in Eq. (3):

$$R_{\mathbf{w}}(s,a,s') = \phi(s,a,s')^\top \mathbf{w}. \tag{3}$$

We can rewrite Eq. (2) w.r.t. a preference vector $\mathbf{w}$ as follows:

$$
\begin{aligned}
Q_\pi^{\mathbf{w}}(s,a) &= \mathop{\mathbb{E}}_{\substack{a_t\sim\pi(s_t)\\ s_{t+1}\sim T(\cdot|s_t,a_t)}} \Big[\sum_{t=0}^{\infty}\gamma^t R(s_t,a_t,s_{t+1})|s_0=s,a_0=a\Big] \\
&= \mathop{\mathbb{E}}_{\substack{a_t\sim\pi(s_t)\\ s_{t+1}\sim T(\cdot|s_t,a_t)}} \Big[\sum_{t=0}^{\infty}\gamma^t \phi(s_t,a_t,s_{t+1})^\top \mathbf{w}|s_0=s,a_0=a\Big] \\
&= \mathop{\mathbb{E}}_{\substack{a_t\sim\pi(s_t)\\ s_{t+1}\sim T(\cdot|s_t,a_t)}} \Big[\sum_{t=0}^{\infty}\gamma^t \phi(s_t,a_t,s_{t+1})^\top |s_0=s,a_0=a\Big]\mathbf{w} \\
&= \psi^\pi(s,a)^\top \mathbf{w} \qquad (4)
\end{aligned}
$$

where,

$$
\psi^\pi(s,a) \equiv \mathop{\mathbb{E}}_{\substack{a_t\sim\pi(s_t)\\ s_{t+1}\sim T(\cdot|s_t,a_t)}} \Big[\sum_{t=0}^{\infty}\gamma^t \phi\left(s_t,a_t,s_{t+1}\right)|s_0=s,a_0=a\Big]. \qquad (5)
$$

Here, $\psi^\pi(s,a)$ is referred to as the successor features vector, the expected value of the discounted feature functions along a trajectory when acting according to a given policy π. Under the SFs representation, $Q_\pi^{\mathbf{w}}(s,a)$ (Eq. (4)) facilitates the evaluation of policies on multiple tasks, where each task is associated with a different preference vector $\mathbf{w}$. This evaluation is known as General Policy Evaluation (GPE) [10]. Furthermore, this formulation enables the enhancement of individual base policies, defined with specific preference vectors w, through their combination. Given a set of base policies Π, a new policy $\pi' \notin \Pi$ can be defined as follows:

$$
\pi'(s) \in \mathop{\text{argmax}}_{a\in\mathcal{A}}(\max_{\pi\in\Pi} Q_\pi^{\mathbf{w}}(s,a)). \qquad (6)
$$

This operation also results in a policy π' that is as good or better than all policies in Π. This paradigm is known as General Policy Improvement (GPI). FastRL [10] is an algorithmic approach that employs GPE and GPI to tackle new tasks defined by different preference vectors efficiently.

IV. Problem Statement

In this work, we consider an MDP, $\mathcal{M}_{\mathbf{w}} = (\mathcal{S},\mathcal{A},T,D_0,\gamma,R_{\mathbf{w}})$ in which the reward function $R_{\mathbf{w}}$ is expressed as a linear function of a predefined set of features ϕ and a weight vector called its preference vector $\mathbf{w}$, as defined in Eq. (3). Additionally, we assume that the reward function exhibits an *imbalance*, where some indices i and j satisfy $w^i >> w^j$, where w^i and w^j are respectively the i-th and j-th components in $\mathbf{w}$. Note that this assumption also implies that there is more than one component in the reward function. To tackle this imbalance, we leverage the SFs formulation to construct a *curriculum* over preference vectors. This curriculum allows the agent to adapt to the imbalanced reward function progressively.

V. Preference-Vector Curriculum (PrefVeC)

We introduce a novel approach that harnesses the successor features (SFs) representation to craft a series of n task variations. These tasks are defined by a set of preference vectors, denoted as $W = \mathbf{w}_1,\ldots,\mathbf{w}_n$, which gradually transition from an initial preference vector $\mathbf{w_1}$—accounting for only one reward component—to the final preference vector $\mathbf{w_n}$, aligning with the true imbalanced reward function. We term this approach the Preference-Vector Curriculum, or simply `PrefVeC`. Formally, given a target task $\mathcal{M}_t = (\mathcal{S},\mathcal{A},T,D_0,\gamma,R_{\mathbf{w}_t})$, `PrefVeC` operates on a set of source tasks $\mathbb{M} = \mathcal{M}_1,\ldots,\mathcal{M}_n$, corresponding to the set of preference vectors W. `PrefVeC` learns a policy for each task and utilizes General Policy Evaluation (GPE) (as in Eq. (4)) to integrate them into a unified policy for the target task.

We investigate two variants of `PrefVeC`: $\texttt{PrefVeC}_\texttt{S}$ and $\texttt{PrefVeC}_\texttt{D}$. $\texttt{PrefVeC}_\texttt{S}$ employs a predefined *static* curriculum, while $\texttt{PrefVeC}_\texttt{D}$ *dynamically* constructs the set of preference vectors W based on information gathered from the previous task in the sequence.

A. Static Curriculum

In the context of $\texttt{PrefVeC}_\texttt{S}$, the entire set of preference vectors W is generated prior to the commencement of training, a process facilitated by a predetermined *Curriculum*. Let $\mathbf{w}_T = [w_T^1,\ldots,w_T^d]$ denote the preference vector for target task T. For clarity, assume that $w_T^1,\ldots,w_T^d$ are ordered in ascending order based on their magnitude (absolute value).

Within the $\texttt{PrefVeC}_\texttt{S}$ framework, users are granted the flexibility to hand-design preference vectors for individual tasks, tailoring them to meet specific task-related requirements. The determination of the preference vectors that form the curriculum is guided by practical engineering considerations, accounting for scale differences and the desired granularity of curriculum progression.

While any curriculum is theoretically valid, in our experiments, we have chosen to explore curricula in which the weight vectors are systematically incremented one component at a time in ascending order of magnitude until they ultimately converge to the target preference vectors. This specific type of curriculum is termed as *Incremental Magnitude Curriculum* (IMC). In the IMC approach, the initial task, $\mathcal{M}_1$ is associated with a preference vector in which the first component possesses the lowest magnitude and the subsequent components are set to zero: $\mathbf{w}_1 = [w_i^1,0,\ldots,0]$. Subsequently, each reward weight is incrementally scaled towards its true value over the span of multiple tasks until it converges to the target preference vector. Within $\texttt{PrefVeC}_\texttt{S}$, the rate at which each reward component converges to its target value is predetermined by the user. For instance, if $\mathbf{w}_T = [1,-20,50]$, a valid IMC progression could be represented by the following sequence of tasks: $[[1,0,0],[1,-5,0],[1,-15,0],[1,-20,0],[1,-20,10],[1,-20,30],[1,-20,50]]$, where each task receives an equal portion of the training budget. It is worth noting that when two or more components, denoted as w^j, share the same magnitude, they are adjusted simultaneously in this IMC approach.

B. Dynamic Curriculum

Similar to $\texttt{PrefVeC}_\texttt{S}$, the dynamic curriculum $\texttt{PrefVeC}_\texttt{D}$ also produces an IMC by gradually adding the different reward components according to their magnitudes. However, unlike $\texttt{PrefVeC}_\texttt{S}$, the rate at which the weight is increased for

each component is not predefined. The SFs representation facilitates the selection of the next preference vector (task) by considering the discrepancy in the optimal actions for each state. The motivation is to choose a minimally different subsequent task such that the optimal actions would differ from those of the previous task. This fosters substantial progress towards the target task while maintaining a stable curriculum. Formally, let π_i be the policy corresponding to the i^{th} task (with preference vector $\mathbf{w}_i$). Given a discrepancy threshold ρ, our objective is to identify the subsequent preference vector $\mathbf{w}_{i+1}$ such that it satisfies the following discrepancy measure:

$$Pr_{s\sim D_{\pi_i}(\mathcal{M}_i)}(\operatorname*{argmax}_{a\in\mathcal{A}} Q_{\pi_i}^{\mathbf{w}_{i+1}}(s,a) \neq \operatorname*{argmax}_{a\in\mathcal{A}} Q_{\pi_i}^{\mathbf{w}_i}(s,a)) = \rho \quad (7)$$

where the probability pertains to the distribution of states, contingent on the initial state distribution D_0 and the transition function T in $\mathcal{M}_i$, as well as the policy π_i. Using Eq. (4), we can rewrite Eq. (7) as follows,

$$Pr_{s\sim D_{\pi_i}(\mathcal{M}_i)}(\operatorname*{argmax}_{a\in\mathcal{A}} \psi^{\pi_i}(s,a)^\top \mathbf{w}_{i+1} \neq \operatorname*{argmax}_{a\in\mathcal{A}} \psi^{\pi_i}(s,a)^\top \mathbf{w}_i) = \rho \quad (8)$$

Assuming only the preference of the j-th reward components differs between consecutive tasks, with the j-th reward components' magnitude increased by some value δ, we can establish a relationship between $\psi^{\pi_i}(s,a)^\top \mathbf{w}_{i+1}$ and $\psi^{\pi_i}(s,a)^\top \mathbf{w}_i$ as follows:

$$\begin{aligned}\psi^{\pi_i}(s,a)^\top \mathbf{w}_{i+1} &= \sum_{k=1}^{d} \psi^{\pi_i}(s,a)^k w_{i+1}^k \\ &= \sum_{k=1}^{d} \psi^{\pi_i}(s,a)^k w_i^k + \psi^{\pi_i}(s,a)^j \delta \\ &= \psi^{\pi_i}(s,a)^\top \mathbf{w}_i + \psi^{\pi_i}(s,a)^j \delta \end{aligned} \quad (9)$$

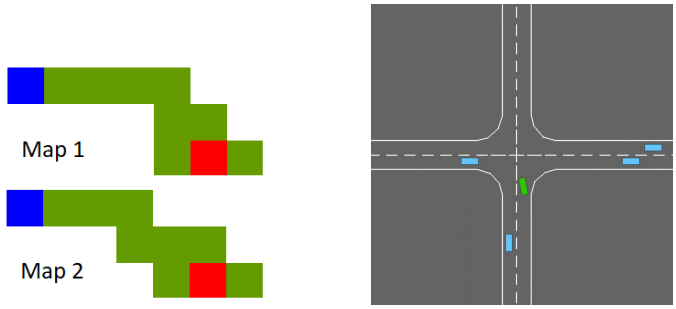
Given a state s, let $a_o = \operatorname{argmax} \psi^{\pi_i}(s,a)^\top \mathbf{w}_i$ denote the optimal action w.r.t. ψ^{π_i} and $\mathbf{w}_i$. A discrepancy in the optimal action w.r.t. $\mathbf{w}_i$ and $\mathbf{w}_{i+1}$ would occur if, for some action $a' \neq a_o$, $\psi^{\pi_i}(s,a')^\top \mathbf{w}_{i+1} > \psi^{\pi_i}(s,a_o)^\top \mathbf{w}_{i+1}$. Based on Eq. (9), this happens when:

$$\psi^{\pi_i}(s,a')^\top \mathbf{w}_i + \psi^{\pi_i}(s,a')^j \delta > \psi^{\pi_i}(s,a_o)^\top \mathbf{w}_i + \psi^{\pi_i}(s,a_o)^j \delta$$

Consequently, by isolating δ, we can compute the minimal amount, denoted as δ_s^*, by which the preference of the j-th reward component needs to be incremented so that some action a' becomes better than a_o:

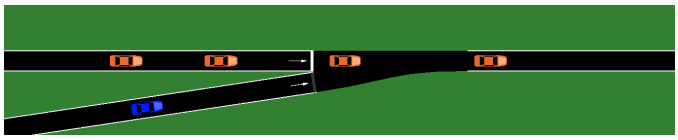
$$\delta_s^* > \min_{a'\neq a_o} \frac{\psi^{\pi_i}(s,a_o)^\top \mathbf{w}_i - \psi^{\pi_i}(s,a')^\top \mathbf{w}_i}{\psi^{\pi_i}(s,a')^j - \psi^{\pi_i}(s,a_o)^j} \quad (10)$$

Eq. (10) bounds the minimal adjustment to the preference of the j-th reward component necessary to change the optimal policy w.r.t. a specific state s. That is, choosing $\delta \geq \delta_s^*$ ensures that $\operatorname*{argmax}_{a\in\mathcal{A}} \psi^{\pi_i}(s,a)^\top \mathbf{w}_{i+1} \neq \operatorname*{argmax}_{a\in\mathcal{A}} \psi^{\pi_i}(s,a)^\top \mathbf{w}_i$. Consequently, in order to satisfy Eq. (8), we need to choose a value δ so that the optimal action would change with probability ρ when states are sampled from $D_\pi(\mathcal{M}_i)$. Formally, we require that $Pr_{s\sim D_{\pi_i}(\mathcal{M}_i)}(\delta \geq \delta_s^*) = \rho$. Since this equation cannot be directly computed w.r.t. all states and actions in the state and action spaces, we aim to approximate δ based on the trajectories experienced by the agent. To achieve that, we sample



(a) Gridworld scenes

(b) Intersection scene



(c) Merge scene

Fig. 1: Example scenarios from each environment. *a:* The maps, with the red agent navigating to the blue box; the path is painted green. *b:* The agent controls the green car, aiming to turn left. *c:* The agent is in control of the blue car, with the objective of merging smoothly into heavy traffic.

N states from the replay buffer and compute the right-hand side of Eq. (10), denoted as $\text{Min}\delta_s^*$, for each sampled state s (excluding cases where the denominator is zero). Then, we sort the sampled states in ascending order based on their $\text{Min}\delta_s^*$ values, resulting in a list $L = [\text{Min}\delta_{s_1}^*, \ldots, \text{Min}\delta_{s_N}^*]$. Finally, we choose $\delta = L[\lceil \rho \cdot N \rceil] + \varepsilon$. $L[\lceil \rho \cdot N \rceil]$ is the expected value of the right-hand side of Eq. (10) at the ρ percentile of the samples, and epsilon is a small value (we used $1e-3$) serves to ensure that the inequality is upheld . If $w_{i+1}^j + \delta > w_T^j$, i.e., the weight is pushed beyond the desired target weight, we set $w_{i+1}^j = w_T^j$, and move to the next reward component, $j+1$.

VI. Implementing PrefVeC

The PrefVeC approach applies to various RL algorithms. In our implementation (available on GitHub), we extend the well-known Deep Q-Networks (DQN) algorithm [32]. DQN employs a deep neural network to approximate the Q-value for each possible action in a given state. In contrast, the models used by the PrefVeC variants approximate the successor-feature values (ψ) for each state-action pair. Similar to DQN, we use a two-network structure: a main network with trainable parameters θ and a target network ζ. Training involves minimizing the temporal difference error (TD-error) between these networks and θ values are copied to ζ periodically. Both PrefVeC_S and PrefVeC_D employ separate identical models for each task in $\mathbb{M}$. Training takes place sequentially on the tasks, with each task receiving a fraction of the training budget equal to $\frac{1}{n}$. The trajectories gathered during training are stored in a modified version of the Prioritized Experience Replay buffer (PER) [28]. PER is employed for non-uniform sampling of trajectories to train the model. It prioritizes transitions based on the magnitude of their temporal difference (TD) error, which quantifies the difference between the predicted and

target Q values. Given a stored transition $\tau = (s_t, a_t, s_{t+1}, r)$, the TD error yields:

$$\text{TD\_error}(\tau) = |(r + \max_{a'} Q(s_{t+1}, a'; \zeta)^j) - Q(s_t, a_t; \theta)^j)| \quad (11)$$

where $Q(\cdot;\theta)$, $Q(\cdot;\zeta)$ are the main and target Q-values.

In our context, the TD error is defined w.r.t. SFs rather than Q-values. A natural adaptation of Eq. (11) to SFs is to represent the rewards and Q-values in terms of $\phi, \mathbf{w}$, and ψ, in accordance to Eq. (3) and Eq. (4). Given a stored transition $\tau = (s_t, a_t, s_{t+1}, \phi, i)$, collected on the i-th task in the curriculum, the adapted TD-error is expressed as:

$$\text{TD\_error}(\tau) = \sum_j \left(\left| ((\phi^j + \max_{a'} \psi(s_{t+1}, a'; \zeta_i)^j) - \psi(s_t, a_t; \theta_i)^j) \cdot w_i^j \right| \right) \quad (12)$$

Here, $\psi(s_t, a_t; \theta_i)$ is the prediction of the i-th task main model on the state s_t and the action a_t and $\psi(s_{t+1}, a'; \zeta_i)$ is the prediction of the i-th task target model on the state s_{t+1} and the action a'. This adaptation ensures the preservation of the significance of each component, as portrayed by the preference vector $\mathbf{w}$. To illustrate, consider the j-th reward component pertaining to collision occurrence as carrying a substantial weight in $\mathbf{w}$. In such a scenario, any discrepancies in $\psi(\cdot;\theta_i)^j$ would be given higher priority compared to similar, or even more pronounced, errors in less critical reward components. Another possibility is to ignore the weight when assigning preferences to transitions, and only considering the error in the prediction:

$$\text{TD\_error}(\tau) = \sum_j \left(\left| ((\phi^j + \max_{a'} \psi(s_{t+1}, a'; \zeta_i)^j) - \psi(s_t, a_t; \theta_i)^j) \right| \right) \quad (13)$$

We use Eq. (12) to adjust the sample priorities within PER. This variant is denoted as *w*PER. In the subsequent section, we conduct an ablation study comparing *w*PER with the second PER variant, derived from Eq. (13), as outlined.

When transitioning to the next task in the sequence, we copy the network weights to initialize the model, reset epsilon decay, and clear the replay buffer.

VII. Empirical Evaluation

We experimented on three environments to assess `PrefVeC`'s performance in settings with highly imbalanced reward functions. Our goal was to test whether a curriculum-based approach can improve policy learning in the presence of significant reward imbalances. We compare the two `PrefVeC` variants with DQN across progressively complex environments. Additionally, our approach is benchmarked against two contemporary RL algorithms, PPO [33] and SAC [34]. The implementation of these algorithms is sourced from the RLLib framework [35]. Furthermore, we extend the comparison to include two well-established safe RL algorithms, CPO [36] and P3O [37], leveraging their implementations from the OmniSafe framework [38].

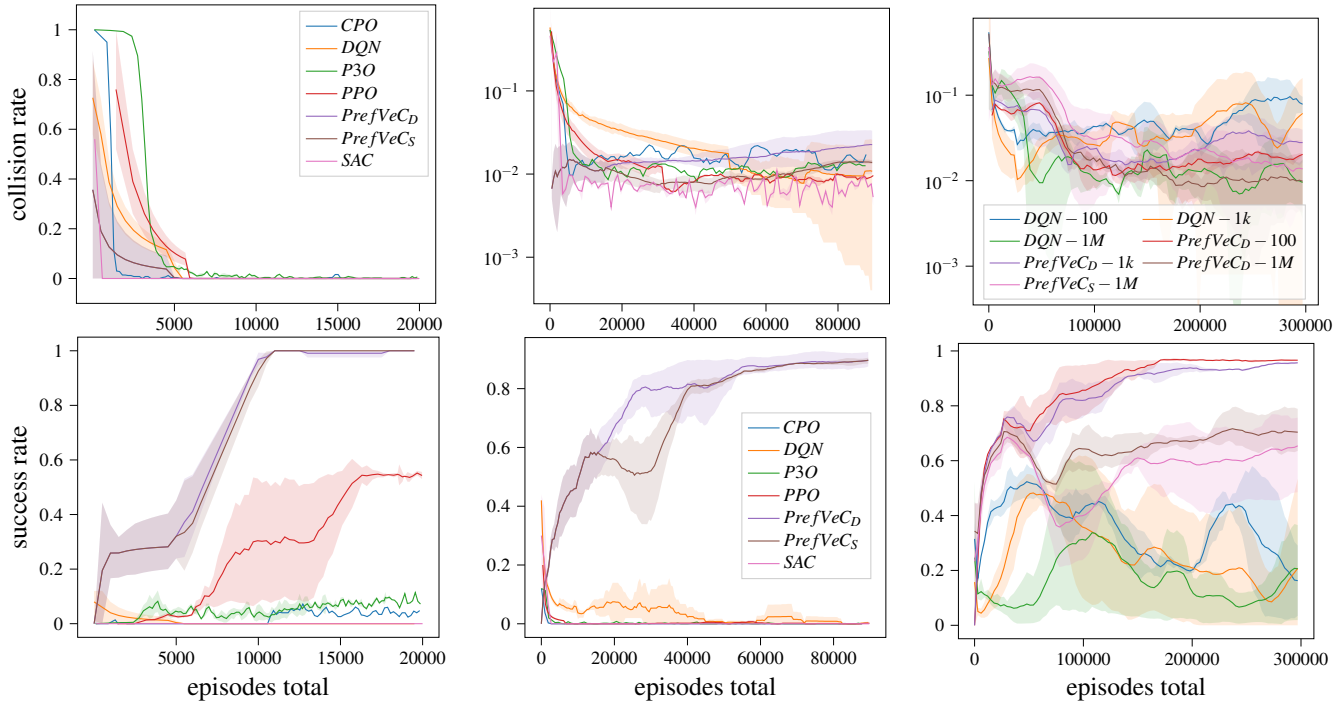
A. Experimental Setting

a) Environments: We evaluated `PrefVeC` in the following three simulated environments, repeating each experiment three times with varying seeds.

Gridworld: A 10x10 grid with a fixed goal position, depicted in Fig.1a, where each grid cell is either safe (green) or unsafe (white). In each instance, the agent (red) is spawned at a random safe location and is tasked with reaching the goal (blue) without ever entering the unsafe region. This relatively simple environment serves as a foundational test-bed for our hypothesis, intentionally avoiding complexities inherent in more intricate environments. In every episode, the agent randomly faces one of the two maps depicted in Fig. 1a. The agent observes the map as a 10x10x3 image, where the three channels correspond to the locations of the agent, the path, and the goal. Feature functions are defined as $\phi^1 = 1$ for collisions, $\phi^2 = 1$ for goal attainment, and 0 otherwise, and $\mathbf{w}_T = [-1000, 1]$. For $\texttt{PrefVeC}_\texttt{S}$, we used the following predefined curriculum: $w_i^1 \in \{0, -2, -50, -100, -500, -1000\}$, where $w_i^2 = 1$ for all tasks.

Intersection: The intersection environment in our study is based on the publicly available highway-env, with environment parameters adopted from a previous research study [39]. In this autonomous driving scenario depicted in Fig. 1b, the EGO vehicle's objective is to execute a left turn at an intersection while navigating alongside 3 to 10 additional agents within the environment. We employ a discrete action space, allowing the EGO vehicle to choose between slowing down, speeding up, or maintaining its current speed. The agent observes the 15 closest vehicles, including the ego vehicle. For each vehicle, the provided information includes the *x*-*y* position, velocities, and heading (sine and cosine). The feature functions for this environment are as follows: $\phi^1 = 1$ if a collision occurs (and zero otherwise), ϕ^2 is scaled between 0 and 1 based on the vehicle speed, and $\phi^3 = 1$ if the EGO vehicle successfully arrives at its destination, and zero otherwise. Moreover, the preference vector is $\mathbf{w}_T = [-1000, 1, 1]$. $\texttt{PrefVeC}_\texttt{S}$ follows this preference vector curriculum: w_i^1 sequentially takes on values of 0, -2, -50, -100, -500, and -1000, while w_i^2 and w_i^3 are set to 1 throughout the curriculum.

Merge: In this environment (illustrated in Fig. 1c), we simulate a scenario resembling a highway on-ramp merge using the SUMO simulator [40]. In this setting, the EGO agent is randomly spawned on a ramp and is assigned to smoothly merge onto a highway through a single-lane on-ramp. This on-ramp may already have uncooperative agents present on the highway. As in the intersection environment, agents have access to a set of discrete actions, allowing them to choose between slowing down, speeding up, or maintaining their current speed. The agent observes the relative *x*-*y* position and speed of the nearest vehicle on each side. Additionally, the agent knows the ego vehicle's speed, the speed limit, and its progress (expressed as a percentage of the road length). The initial speed of the EGO vehicle is randomly set within a range that allows the vehicle to decelerate and come to a complete stop before reaching the merge point. The feature functions in this environment are defined as follows: $\phi^1 = 1$ if a collision occurs (and zero otherwise), and ϕ^2 is determined by the distance traveled (scaled by $0.001/m$). The preference vector of the target is $\mathbf{w}_T = [-1000000, 1]$ (as motivated in [1]). For the curriculum, $\texttt{PrefVeC}_\texttt{S}$ considers the following tasks: w_i^1 takes on sequential values of 0, -2, -50, -1000, -5000,



(a) Grid (b) Intersection (c) Merge

Fig. 2: Training plots on the different environments. Top: collision rates. Bottom: success rates.

and -1000000, and $w_i^2 = 1$ for all tasks.

b) Hyperparameters and Networks Architecture: All algorithms use relatively simple 3-layer MLP models with hidden layers of size 64-32-n_a (where n_a is the number of actions in the environment). DQN utilize a *Linear* activation function for the final layer, while PrefVeC$_S$ and PrefVeC$_D$ use *sigmoid* for each $\psi_j^{\pi_i}$.

In the intersection environment, we used the DQN implementation from [39], and adapted the models used by PrefVeC$_S$ and PrefVeC$_D$ models to align with the architecture employed in the DQN model. We employed the Adam optimizer with a learning rate of 0.001 and harnessed PER (*w*PER for PrefVeC$_S$ and PrefVeC$_D$) with a buffer size of 500,000 samples and a γ of 0.99. The total training budgets for each environment were 20,000 episodes for GridWorld, 90,000 for Intersection, and 300,000 for Merge. In addition, we used a batch size of 128 and conducted 10 weight updates after each episode. The target network weights were updated every 10 network updates to facilitate stable training. Lastly, PrefVeC$_D$ used $\rho = 0.1$ in all of the experiments.

B. Results

The experiment results are presented through plots for each environment. In these plots, the x-axis represents the number of training episodes, while the y-axis captures two key aspects: the agent's collision rate (first row) and the percentage of task completion (second row). Solid lines depict the average performance across the three runs, while the shaded area represents the standard deviation.

a) Gridworld: The results for the Gridworld environment are presented in Fig. 2a. The results show that all baselines struggle to cope with the imbalanced reward, ultimately learning a degenerate policy where the agent almost always remains idle. The exception is PPO, in which the agent has a roughly 50% successful completion rate after training. In stark contrast, the PrefVeC variants successfully learn to always solve the task thanks to the utilization of a curriculum. PrefVeC$_S$ and PrefVeC$_D$ have the exact same collision rate (overlapping in Fig. 2a) and a similar success rate. Notably, all agents manage to learn to— almost—completely avoid collisions after ∼6000 episodes.

b) Intersection: The results for the Intersection environment are reported in Fig. 2b. In this scenario, we observe trends akin to those in the GridWorld environment. With the baselines only rarely completing the task, as they mostly learn to remain stationary or avoid entering the intersection. Conversely, both iterations of the PrefVeC framework demonstrate great performance, achieving success rates close to 100% while keeping collision rates relatively low and comparable to the other algorithms. Among the PrefVeC variants, PrefVeC$_D$ outperforms PrefVeC$_S$ in success rate, whereas PrefVeC$_S$ displays a slightly lower collision rate. Furthermore, it's worth noting that the dynamic variant, PrefVeC$_D$, demonstrates a higher degree of stability in its performance improvement course throughout the training process when compared to PrefVeC$_S$.

C. Ablation Study and Analysis

a) Different Reward Weight Vectors: In Fig. 2c, we performed an ablation study on the Merge environment, focusing on assessing the impact of different weight vectors. To narrow down the scope of the evaluation to a manageable number of algorithm configurations, we omitted baselines with

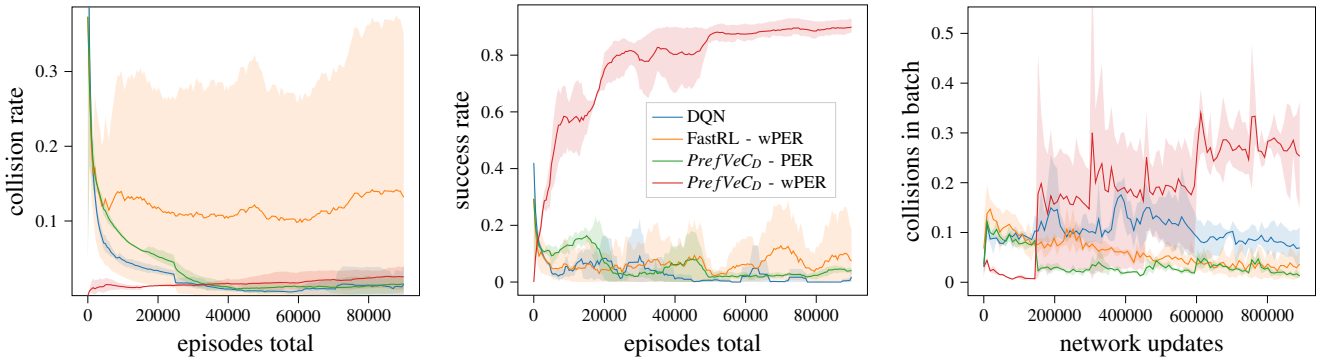


Fig. 3: Training plots on the intersection environment for the ablation studies.

poor performance on prior tasks. Our focus was on DQN as a natural baseline and PrefVeC$_D$, the most successful variant based on previous experiments. For this experiment, we used three distinct collision penalties: -100, $-1k$, and $-1M$. The results for both algorithms showed relatively similar success rates and collision outcomes for the -100 and -1000 weights, while the $-1M$ weights led to lower success rates and reduced collision rates, aligning with expectations. Notably, PrefVeC$_D$ consistently achieved significantly higher success rates with either comparable or lower collision rates in both cases. It's worth mentioning that DQN achieved non-zero rewards due to the initial speed of the EGO vehicle. In an auxiliary experiment (omitted for brevity) where we fixed the initial speed of the EGO to zero, DQN failed to complete the task, while PrefVeC$_D$ maintained similar results. The results also feature PrefVeC$_S$ with a collision penalty of $-1M$. PrefVeC$_S$ exhibits inferior outcomes compared to PrefVeC$_D$ with the same preference vector, displaying a higher collision rate alongside a lower success rate.

b) PER Modification: Fig. 3 showcases an ablation study conducted within the interaction domain. This study compares *w*PER, derived from Eq. (12), with PER that disregards the weighting of errors based on $\mathbf{w}$, as derived from Eq. (13). The findings underscore the importance of using $\mathbf{w}$ for the scaling of transitions. Notably, PrefVeC$_D$ employing unweighted PER yields results akin to DQN concerning collision and success rates. This outcome stems from the oversight of error weighting, in which relatively minor errors related to collisions are deprioritized, thus impeding their inclusion in training samples, despite their significance. Consequently, the models are updated during training without observing sufficient collisions, hampering the agent's ability to learn collision handling. This conjecture finds support in the right plot of Fig. 3, which illustrates the percentage of collisions sampled in each training batch. While PrefVeC$_D$ employing PER scarcely encounters collisions throughout training, its counterpart utilizing *w*PER observes a growing number of collisions, mirroring the escalating importance of collisions within the preference vector for each task. Notably, the PER employed by DQN, akin to *w*PER, inherently considers the importance of collisions when assigning preferences through the utilization of Q-values.

c) The Importance of Curriculum: This study compares PrefVeC$_D$ (with *w*PER) to other reward decomposition approaches that do not use a curriculum. In particular, we compare PrefVeC$_D$ to FastRL [10] equipped with *w*PER. [1] The results demonstrate a significant performance advantage of PrefVeC$_D$ over FastRL, achieving markedly lower collision rates and substantially higher success rates. Importantly, FastRL closely resembles PrefVeC when no curriculum is applied (i.e., training solely on the target task). Therefore, the substantial enhancement in results can be attributed to the integration of curricula, introduced in our approach.

d) Resulting Curricula: The curricula automatically generated by PrefVeC$_D$ exhibit variation across domains and even within different runs of the same domain. Notably, the first task in each curriculum, by definition, ignores collisions, while the last task utilizes the target reward vector, representing the full penalty for collisions. In Gridworld, the resulting curriculum consistently comprised three tasks for all runs, with the collision penalty of the second task ranging from -397 to -480. In the intersection domain, the curricula contained between two and five tasks, with intermediate tasks focusing on low-collision penalties (ranging from -0.5 to -60). Similarly, in the merge environment, curricula comprised between two to five tasks, with intermediate tasks emphasizing low-collision penalties (up to -2617 for the scenario with a target reward function of $-1M$). Although we have not explicitly experimented with different ρ values, the performance difference between PrefVeC$_D$ and PrefVeC$_S$ suggests that the approach is relatively insensitive to varied ρ values. Both PrefVeC$_D$ and PrefVeC$_S$ yield distinct curricula but maintain a significant performance advantage over the baselines.

VIII. Conclusion, Limitations, and Discussion

In this work, we addressed the challenge of imbalanced reward environments by introducing PrefVeC, a novel approach that integrates successor feature representation with curriculum learning. We first introduced a straightforward version of PrefVeC, denoted as PrefVeC$_S$, and later developed PrefVeC$_D$, an enhanced variant capable of dynamically constructing a curriculum. Our implementation of PrefVeC builds upon the DQN algorithm.

[1] In this setting, MaxPain [7] can be conceptualized as FastRL with slight variations. This is achieved by utilizing a single policy with two SFs, one corresponding to positive rewards and the other to negative rewards.

We conducted extensive empirical evaluations across three diverse environments, revealing that the standard DQN struggles to adapt to imbalanced rewards, often leading to policies where the agent fails to make meaningful progress. In contrast, our `PrefVeC` variants demonstrated substantial enhancements in handling reward imbalances, both with and without an expert-provided curriculum. Notably, the `PrefVeC` approach, while showing significant promise, does have limitations. `PrefVeC` tackles reward imbalance by gradually considering reward components. Since safety concerns are often associated with high penalties, they are often ignored by `PrefVeC` at the beginning of the training process. This approach is appropriate when training is done using a simulator, but less suitable for training in a physical environment. Consequently, synergizing `PrefVeC` with Sim2Real approaches could enhance the transferability of policies trained in simulators to physical robots. Another limitation arises in scenarios where taking no action at all represents the optimal strategy, due to substantial reward imbalances. In these cases, `PrefVeC` still biases learning towards non-degenerate policies, prioritizing the acquisition of autonomous behaviors even at the potential expense of lower cumulative rewards. Additionally, there may be cases in which learning an intermediate reward component interferes with learning a more significant one. For example, this might happen when both components involve the termination of the episode. Addressing and mitigating these limitations remains a subject for future research and investigation.

REFERENCES

[1] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "On a formal model of safe and scalable self-driving cars," *arXiv preprint arXiv:1708.06374*, 2017.

[2] S. Russell and A. Zimdars, "Q-decomposition for reinforcement learning agents," in *ICML*. AAAI Press, 2003, pp. 656–663.

[3] H. van Seijen, M. Fatemi, R. Laroche, J. Romoff, T. Barnes, and J. Tsang, "Hybrid reward architecture for reinforcement learning," in *NIPS*, 2017, pp. 5392–5402.

[4] Z. Juozapaitis, A. Koul, A. Fern, M. Erwig, and F. Doshi-Velez, "Explainable reinforcement learning via reward decomposition," in *IJCAI/ECAI Workshop on explainable artificial intelligence*, 2019.

[5] J. R. Kok and N. Vlassis, "Sparse cooperative q-learning," in *ICML*, vol. 69. ACM, 2004.

[6] S. Elfwing and B. Seymour, "Parallel reward and punishment control in humans and robots: Safe reinforcement learning using the maxpain algorithm," in *ICDL-EPIROB*. IEEE, 2017, pp. 140–147.

[7] J. Wang, S. Elfwing, and E. Uchibe, "Deep reinforcement learning by parallelizing reward and punishment using the maxpain architecture," in *ICDL-EPIROB*. IEEE, 2018, pp. 175–180.

[8] A. Barreto, W. Dabney, R. Munos, J. J. Hunt, and T. Schaul, "Successor features for transfer in reinforcement learning," *NeurIPS*, vol. 30, 2017.

[9] P. Dayan, "Improving generalization for temporal difference learning: The successor representation," *Neural Comput.*, vol. 5, no. 4, pp. 613–624, 1993.

[10] A. Barreto, S. Hou, D. Borsa, D. Silver, and D. Precup, "Fast reinforcement learning with generalized policy updates," *Proc. Natl. Acad. Sci. USA*, vol. 117, no. 48, pp. 30 079–30 087, 2020.

[11] L. Szoke, S. Aradi, T. Bécsi, and P. Gáspár, "Skills to drive: Successor features for autonomous highway pilot," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–12, 2022.

[12] S. Hansen, W. Dabney, A. Barreto, T. Van de Wiele, D. Warde-Farley, and V. Mnih, "Fast task inference with variational intrinsic successor features," *arXiv preprint arXiv:1906.05030*, 2019.

[13] C. Reinke and X. Alameda-Pineda, "Successor feature representations," 2023.

[14] K. Brantley, S. Mehri, and G. J. Gordon, "Successor feature sets: Generalizing successor representations across policies," in *AAAI*. AAAI Press, 2021, pp. 11 774–11 781.

[15] M. W. Nemecek and R. Parr, "Policy caches with successor features," in *ICML*, ser. Proceedings of Machine Learning Research, vol. 139. PMLR, 2021, pp. 8025–8033.

[16] L. Lehnert and M. L. Littman, "Successor features combine elements of model-free and model-based reinforcement learning," *J. Mach. Learn. Res.*, vol. 21, pp. 196:1–196:53, 2020.

[17] M. Janner, I. Mordatch, and S. Levine, "gamma-models: Generative temporal difference learning for infinite-horizon prediction," *NeurIPS*, vol. 33, pp. 1724–1735, 2020.

[18] L. Lehnert, S. Tellex, and M. L. Littman, "Advantages and limitations of using successor features for transfer in reinforcement learning," *arXiv preprint arXiv:1708.00102*, 2017.

[19] M. Abdolshah, H. Le, T. K. George, S. Gupta, S. Rana, and S. Venkatesh, "A new representation of successor features for transfer across dissimilar environments," in *International Conference on Machine Learning*. PMLR, 2021, pp. 1–9.

[20] T. Gupta, A. Kumar, and P. Paruchuri, "Successor features based multi-agent RL for event-based decentralized mdps," pp. 6054–6061, 2019.

[21] S. H. Kim, N. Van Stralen, G. Chowdhary, and H. T. Tran, "Disentangling successor features for coordination in multi-agent reinforcement learning," pp. 751–760, 2022.

[22] A. Filos, C. Lyle, Y. Gal, S. Levine, N. Jaques, and G. Farquhar, "Psiphi-learning: Reinforcement learning with demonstrations using successor features and inverse temporal difference learning," in *ICML*. PMLR, 2021, pp. 3305–3317.

[23] M. Gimelfarb, A. Barreto, S. Sanner, and C.-G. Lee, "Risk-aware transfer in reinforcement learning using successor features," *NeurIPS*, vol. 34, pp. 17 298–17 310, 2021.

[24] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proceedings of the 26th annual international conference on machine learning*, 2009, pp. 41–48.

[25] P. Soviany, R. T. Ionescu, P. Rota, and N. Sebe, "Curriculum learning: A survey," *Int. J. Comput. Vis.*, vol. 130, no. 6, pp. 1526–1565, 2022.

[26] S. Narvekar, B. Peng, M. Leonetti, J. Sinapov, M. E. Taylor, and P. Stone, "Curriculum learning for reinforcement learning domains," *J. Mach. Learn. Res.*, vol. 21, pp. 181:1–181:50, 2020.

[27] M. Asada, S. Noda, S. Tawaratsumida, and K. Hosoda, "Purposive behavior acquisition for a real robot by vision-based reinforcement learning," *Machine learning*, vol. 23, no. 2, pp. 279–303, 1996.

[28] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *arXiv preprint arXiv:1511.05952*, 2015.

[29] S. Racanière, A. K. Lampinen, A. Santoro, D. P. Reichert, V. Firoiu, and T. P. Lillicrap, "Automated curriculum generation through setter-solver interactions," in *ICLR*, 2020.

[30] C. Florensa, D. Held, X. Geng, and P. Abbeel, "Automatic goal generation for reinforcement learning agents," in *ICML*, 2018.

[31] Y. Wu and Y. Tian, "Training agent for first-person shooter game with actor-critic curriculum learning," in *ICLR*. OpenReview.net, 2017.

[32] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[33] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[34] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *ICML*. PMLR, 2018, pp. 1861–1870.

[35] E. Liang, R. Liaw, R. Nishihara, P. Moritz, R. Fox, K. Goldberg, J. Gonzalez, M. Jordan, and I. Stoica, "Rllib: Abstractions for distributed reinforcement learning," in *International conference on machine learning*. PMLR, 2018, pp. 3053–3062.

[36] J. Achiam, D. Held, A. Tamar, and P. Abbeel, "Constrained policy optimization," in *ICML*. PMLR, 2017, pp. 22–31.

[37] L. Zhang, L. Shen, L. Yang, S. Chen, X. Wang, B. Yuan, and D. Tao, "Penalized proximal policy optimization for safe reinforcement learning," pp. 3744–3750, 2022.

[38] J. Ji, J. Zhou, B. Zhang, J. Dai, X. Pan, R. Sun, W. Huang, Y. Geng, M. Liu, and Y. Yang, "Omnisafe: An infrastructure for accelerating safe reinforcement learning research," *CoRR*, vol. abs/2305.09304, 2023.

[39] E. Leurent and J. Mercat, "Social attention for autonomous decision-making in dense traffic," *arXiv preprint arXiv:1911.12250*, 2019.

[40] P. Á. López, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. WieBner, "Microscopic traffic simulation using SUMO," in *ITSC*. IEEE, 2018, pp. 2575–2582.