Palette Inpainting Diffusion Curriculum Reinforcement Learning (PIDCRL)

Faruk Oruç[1], Erdi Sayar[2], Giovanni Iacca[3], Alois Knoll[1], Erdal Kayacan[2]

***Abstract*—Curriculum reinforcement learning (CRL) aims to speed up agent learning by organizing tasks in a progressively harder sequence. However, many existing CRL methods struggle to guide agents toward meaningful goals, especially when domain knowledge is limited or unavailable. To overcome this limitation, we introduce PIDCRL, a diffusion-based CRL framework that leverages a mask-conditional, image-to-image pretrained diffusion model to automatically generate curriculum goals from agent trajectory heatmaps. Given a heatmap trajectory image as input, the model produces candidate curriculum goals, which we then filter and select using several strategies, including averaging, Q-value scoring, and a trainable reward-based mechanism. These strategies identify curriculum goals that are both achievable and appropriately challenging, enabling effective curricula without expert-designed heuristics. Across three maze environments, PIDCRL matches or outperforms ten state-of-the-art CRL baselines.**

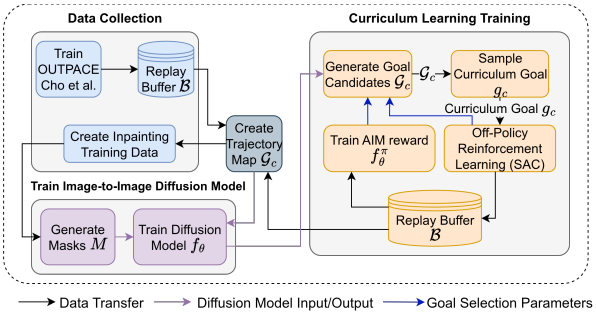


Fig. 1: Overview of PIDCRL. The pipeline comprises three phases: (I) data collection and conversion of trajectories to heatmaps; (II) Palette diffusion model training; (III) inference, where the trained model generates and filters curriculum goals to guide the RL agent.

I. Introduction

Reinforcement learning (RL) enables agents to learn optimal actions through trial-and-error interactions with the environment, guided by a reward signal to adjust their strategies to maximize cumulative rewards. Deep RL [1], [2], which integrates deep neural networks with RL, provides a scalable solution for high-dimensional decision-making tasks, such as mastering complex video games [3], autonomous driving [4], and robot manipulation tasks [5], [6]. A key challenge in RL is efficiently exploring large state spaces, which becomes increasingly costly as their size grows [7]. Curriculum reinforcement learning (CRL) provides the agent with instructive sequences that progressively lead it toward the target objective. Several strategies exist for curriculum goal generation. Uncertainty-based methods leverage exploration guidance but can be less effective in large goal spaces [8], [9]. Other approaches involve interpolating between source and target task distributions [10], [11] or minimizing the distance (e.g., Euclidean) between the distribution of achieved states and a desired goal distribution, though distance metrics may be suboptimal in certain environments, such as Maze [12]. Generative adversarial networks (GANs) have also been employed for CRL, although they usually depend on arbitrary thresholds [13]. The present work starts from the hypothesis that existing CRL methods may provide a valuable source of knowledge, currently untapped, that could be used to generate curriculum goals more effectively. To this aim, we propose an innovative approach to curriculum goal generation by repurposing Palette [14]–a diffusion model originally developed for image colorization, inpainting, uncropping, and JPEG restoration–for curriculum generation. Specifically, we represent both the agent's trajectory and the curriculum goal as heatmap images, with a marker indicating the goal location. We use a state-of-the-art CRL algorithm, OUTPACE [9], for building a dataset of such trajectories. By learning from trajectory-goal pairs, the diffusion model implicitly captures underlying environment-specific features and generates goals that are both achievable from the current trajectory and progressively challenging, without relying on explicit domain knowledge or handcrafted heuristics [15]. The main contributions of our paper are:

1) Adaptation of diffusion models for CRL: We adapt the image-to-image translation capabilities of the Palette [14] diffusion model to the domain of curriculum learning. By processing agent-trajectory heatmaps, we introduce a visual generative approach to defining learning goals.
2) Context-aware training and generative inference: We construct a robust dataset using OUTPACE [9] (the best-performing method among our baselines (Section IV-A)). During training, a binary mask highlights the region to be reconstructed (Section IV-B), enabling the diffusion model to inpaint that area conditioned on the visible context. At inference (Section IV-C), the model receives a masked agent trajectory image as input and generates a set of curriculum goal candidates.
3) Strategic goal selection: We implement and evaluate distinct selection strategies (Section IV-D) to filter the generated candidates and identify a goal that is achievable yet sufficiently challenging.

[1]Technical University of Munich, {faruk.oruc,k}@tum.de
[2]Paderborn University, {erdi.sayar, erdal.kayacan}@uni-paderborn.de,
[3]University of Trento, giovanni.iacca@unitn.it

II. Related Work

Palette [14] is a versatile framework for image-to-image translation using conditional diffusion models. The framework is evaluated on four key tasks: colorization, where it adds color to grayscale images; inpainting, where it fills in missing regions; uncropping, where it extends image boundaries; and JPEG restoration, where it recovers quality from compressed images and outperforms GAN and regression baselines without task-specific tuning.

CRL frameworks systematically design sequences of learning experiences to enhance agent performance and training efficiency. These methods construct intermediate objectives that guide agents toward target goals, demonstrating particular utility in robotics manipulation tasks [6]. Hindsight experience replay (HER) [16] can be considered as an implicit curriculum approach. However, its effectiveness diminishes when target goals are distant from initial states. Building upon HER, hindsight goal generation (HGG) [12] improves this limitation by combining value function maximization with Wasserstein distance minimization.

Other methods, such as CURROT [17] and GRADIENT [18], employ optimal transport principles for curriculum design. CURROT reformulates CRL as a constrained optimization problem using Wasserstein distance to quantify distributional divergence, while GRADIENT introduces task-specific contextual metrics and accommodates non-parametric distributions in diverse settings. Self-paced reinforcement learning (SPRL) [10] adapts principles from self-paced learning to RL.

GoalGAN [13], instead, utilizes GANs [19] to generate intermediate goals without explicit target distribution alignment. A discriminator evaluates goal difficulty relative to the current policy, while a generator proposes regions via indicator reward functions. Policies are conditioned on both state and goal, akin to universal value function approximators [20].

PLR [8] prioritizes experiences with high temporal-difference error-based learning potential, creating a self-paced curriculum. VSD [21] selects goals by estimating epistemic uncertainty in value function predictions, favoring goals with balanced confidence. ACL [22] maximizes learning progress through metrics like prediction accuracy improvements and network complexity growth.

ALP-GMM [11] fits Gaussian mixtures using absolute learning progress (ALP) scores, derived from reward differences across episodes. OUTPACE [9] employs adversarial intrinsic motivation (AIM) [23] to minimize the Wasserstein distance. It generates curricula via conditional normalized maximum likelihood to classify goal-associated states and prioritizes uncertain, temporally distant objectives using meta-learning and Wasserstein-based temporal approximations.

III. Background

We now introduce the background concepts on multi-goal RL, soft actor-critic (SAC), Wasserstein distance, AIM, and diffusion models, which are the main elements of our approach.

A. Multi-Goal Reinforcement Learning

We formulate the multi-goal RL problem as a goal-conditioned Markov decision process (MDP), defined by the tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{G}, \mathcal{T}, R, \rho_0, \gamma_r \rangle$. Here, $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, and $\mathcal{G}$ is the space of possible goals. The transition dynamics $\mathcal{T}(s'|s,a)$ gives the probability of transitioning to state s' after taking action a in state s. The initial state s_0 and the desired goal g for an episode are sampled from a joint distribution $\rho_0(s_0, g)$. The reward function $R(s, a, g, s')$ provides feedback, and $\gamma_r \in [0, 1)$ is a discount factor. The agent's objective is to learn a goal-conditioned policy $\pi(a|s, g)$ that maximizes the expected discounted cumulative reward: $J(\pi) = \mathbb{E}_{s_0,g\sim\rho_0,a_t\sim\pi(\cdot|s_t,g),s_{t+1}\sim\mathcal{T}(\cdot|s_t,a_t)} \left[\sum_{t=0}^{\infty} \gamma_r^t R(s_t, a_t, g, s_{t+1})\right]$. In this work, we utilize the AIM [23] reward function r_φ, detailed in Section III-B, to provide denser learning signals. Building upon multi-goal RL, we further enhance the agent's learning efficiency by incorporating curriculum goals. CRL structures the training process by presenting tasks in a sequence of increasing difficulty, allowing the agent to master simpler tasks before progressing to more complex ones.

B. Wasserstein Distance and Adversarial Intrinsic Motivation Reward Function

The AIM reward function can be learned, as suggested by Durugkar et al. [23], through the minimization of the Wasserstein distance between the state visitation distribution ρ_π and the target goal distribution $\mathcal{G}$. The minimization of the Wasserstein distance $W_1(\rho_\pi, \mathcal{G})$, also referred to as the Kantorovich–Rubinstein distance, enables the formulation of a reward function that assesses the required effort to transition the state visitation distribution ρ_π to the desired goal distribution $\mathcal{G}$. It is expressed as follows: $W_1(\rho_\pi, \mathcal{G}) = \sup_{\|f\|_L \leq 1} [\mathbb{E}_{g\sim\mathcal{G}}[f(g)] - \mathbb{E}_{s\sim\rho_\pi}[f(s)]]$. Here, $f(s)$ is a potential function and monotonically increases along trajectories and reaches its peak value at $f(g)$. A reward function r_φ^π can be approximated by a neural network whose output increases as the states approach the desired goal $g \in \mathcal{G}$ and can be trained using the data collected by the policy π. Leveraging the estimation of the Wasserstein distance $W_1(\rho_\pi, \mathcal{G})$, the loss function for training the parameterized reward function r_φ^π is defined as follows:

$$\begin{aligned}\mathcal{L}_\varphi = &\mathbb{E}_{(s,g)\sim\mathcal{B}}\left[f_\varphi^\pi(s) - f_\varphi^\pi(g)\right] + \\ &\lambda \cdot \mathbb{E}_{(s,s',g)\sim\mathcal{B}}\left[\max\left(\left|f_\varphi^\pi(s) - f_\varphi^\pi(s')\right| - 1, 0\right)^2\right]\end{aligned} \tag{1}$$

where $\mathcal{B}$ is the replay buffer and the second component of the sum is a penalty term, and the coefficient λ is necessary to ensure smoothness [23]. The reward, which is the negative of the Wasserstein distance $-W_1(\rho_\pi, \mathcal{G})$, can then be calculated as follows[1]: $r_\varphi(s, g) = f_\varphi^\pi(s) - f_\varphi^\pi(g)$.

[1]Note that $r_\varphi(s, g)$ is used both as the reward signal (maximized by the policy) and, when the initial state s_0 is fixed, as a measure to evaluate goal proximity during curriculum selection (minimized to find accessible goals)

C. Diffusion Models

Diffusion models [24] express a probability distribution $p(x_0)$ through latent variables in the form $p_\theta(\boldsymbol{x}_0) := \int p_\theta(\boldsymbol{x}_{0:N})\, d\boldsymbol{x}_{1:N}$, where $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ are latent variables of the same dimensionality as the data $\boldsymbol{x}_0 \sim p(\boldsymbol{x}_0)$. These models are characterized by a forward and a reverse diffusion process. The forward diffusion process approximates the posterior $q(\boldsymbol{x}_{1:N} \mid \boldsymbol{x}_0)$ using a Markov chain that perturbs the input data by gradually adding Gaussian noise $\boldsymbol{x}_0 \sim q(\boldsymbol{x}_0)$ in N steps with a predefined variance schedule $\beta_1, \ldots, \beta_N$. This process is defined as:

$$q(\boldsymbol{x}_{1:N} \mid \boldsymbol{x}_0) := \prod_{k=1}^{N} q(\boldsymbol{x}_k \mid \boldsymbol{x}_{k-1}) \tag{2}$$
$$q(\boldsymbol{x}_k \mid \boldsymbol{x}_{k-1}) := \mathcal{N}\left(\boldsymbol{x}_k; \sqrt{1-\beta_k}\boldsymbol{x}_{k-1}, \beta_k \boldsymbol{I}\right).$$

Here, $\mathcal{N}(\boldsymbol{x}_k;\, \sqrt{1-\beta_k}\, \boldsymbol{x}_{k-1},\, \beta_k \boldsymbol{I})$ denotes a multivariate Gaussian (normal) distribution with mean $\boldsymbol{\mu} = \sqrt{1-\beta_k}\, \boldsymbol{x}_{k-1}$ and covariance matrix $\boldsymbol{\Sigma} = \beta_k \boldsymbol{I}$, where $\boldsymbol{I}$ is the identity matrix. The reverse diffusion process aims to recover the original input data from the noisy (diffused) data. It learns to progressively reverse the diffusion process step by step and approximates the joint distribution $p_\theta(\boldsymbol{x}_{0:N})$. This process is defined as:

$$p_\theta(\boldsymbol{x}_{0:N}) := p(\boldsymbol{x}_N) \prod_{k=1}^{N} p_\theta(\boldsymbol{x}_{k-1} \mid \boldsymbol{x}_k) \tag{3}$$
$$p_\theta(\boldsymbol{x}_{k-1} \mid \boldsymbol{x}_k) := \mathcal{N}(\mathbf{x}_{k-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_k, k), \boldsymbol{\Sigma}_\theta(\mathbf{x}_k, k))$$

where $p(\boldsymbol{x}_N) = \mathcal{N}(\boldsymbol{x}_N; \mathbf{0}, \boldsymbol{I})$. The optimization of the reverse diffusion process is achieved by maximizing the evidence lower bound (ELBO) $\mathbb{E}_q\left[\ln \frac{p_\theta(\boldsymbol{x}_{0:N})}{q(\boldsymbol{x}_{1:N}|\boldsymbol{x}_0)}\right]$. Once trained, sampling data from Gaussian noise $\boldsymbol{x}_N \sim p(\boldsymbol{x}_N)$ and running through the reverse diffusion process from $k = N$ to $k = 0$ yields an approximation of the original data distribution.

IV. Methodology

In multi-goal RL, the agent is tasked with achieving multiple goals, each sampled from a goal distribution at the start of every episode. By integrating a curriculum design into multi-goal RL, we restructure the task to begin with simpler goals and progressively increase the task difficulty. To generate these curriculum goals, we utilize Palette [14], an image-to-image diffusion model, originally developed for image inpainting, which we repurpose to create curriculum goals for CRL.

As RL and diffusion processes operate on distinct temporal values, we denote the diffusion iteration index as $k \in \{0, \ldots, N\}$ and the RL trajectory index as $t \in \{0, \ldots, T\}$.

The curriculum goal generation process begins with training a conditional diffusion model. This training involves preparing a dataset of images where each image represents an RL episode. These images encode the agent's trajectory as a heatmap where pixel intensity reflects the frequency or recency of state visits– and the goal as a distinct black box marking its location. During inference (i.e., during the reverse-diffusion process), the trained diffusion model generates curriculum goals for the RL agent during its roll-out phase. Specifically, during RL training, we construct the agent's trajectory heatmap image $\mathcal{I}$ for one episode and apply a mask to occlude the trajectory part of the image and a random subset of pixels. The masked region is initialized with Gaussian noise, while unmasked pixels retain their original values (Section IV-B). The diffusion model then iteratively denoises the masked region, conditioned on the masked pixels of the original image. The final output is an image combining the curriculum goal (in the denoised region) with the agent's trajectory heatmap.

Recall that the neural network f_θ is trained to predict ϵ given a noisy image $\mathcal{I}_N$ and $\mathcal{I}_k$. Consequently, once $\mathcal{I}_k$ is known, we estimate $\mathcal{I}_0$ by rearranging the terms in Eq. (3), yielding:

$$p_\theta(\mathcal{I}_{0:N}|\mathcal{I}_N) = p(\mathcal{I}_N) \prod_{k=1}^{N} p_\theta(\mathcal{I}_{k-1}|\mathcal{I}_k, \mathcal{I}_N) \tag{4}$$

where:

$$p(\mathcal{I}_N) = \prod_{i,j} \begin{cases} \delta(\mathcal{I}_{N,i,j} - \mathcal{I}_{0,i,j}) & \text{if } \mathcal{M}_{i,j} = 0 \\ \mathcal{N}(\mathcal{I}_{N,i,j}; 0, \sigma^2) & \text{if } \mathcal{M}_{i,j} = 1. \end{cases} \tag{5}$$

$\mathcal{I}_N$ is obtained by applying a binary mask to the image, so as to control which part of the image should undergo the diffusion model's noising and denoising process:

$$\mathcal{I}_N = \mathcal{I}_0 \odot (1 - \mathcal{M}) + \eta \odot \mathcal{M} \tag{6}$$

where $\mathcal{M}$ is the binary mask, $\mathcal{I}_N$ is the noisy image in the masked region, $\eta \sim \mathcal{N}(0, \sigma^2)$ is Gaussian noise, and $\odot$ is element-wise multiplication. The reverse diffusion process, parameterized by θ, is then applied, starting from the last step N and proceeding backward to step 1:

$$\mu_\theta(\mathcal{I}_k, \mathcal{I}_N, \bar{\alpha}_k) = \frac{1}{\sqrt{\alpha_k}} \left(\mathcal{I}_k - \frac{1-\alpha_k}{\sqrt{1-\bar{\alpha}_k}} f_\theta(\mathcal{I}_k, \mathcal{I}_N, \bar{\alpha}_k) \right) \tag{7}$$

$$\mathcal{I}_{k-1}|\mathcal{I}_k = \frac{\mathcal{I}_k}{\sqrt{\alpha_k}} - \frac{1-\alpha_k}{\sqrt{\alpha_k(1-\bar{\alpha}_k)}} f_\theta(\mathcal{I}_k, \mathcal{I}_N, \bar{\alpha}_k) + \sqrt{1-\alpha_k}\epsilon \tag{8}$$

where $\epsilon \sim \mathcal{N}(0, I)$. We adopt the following simplified loss function to train the conditional model f_θ, which approximates the noise term ϵ:

$$\mathcal{L} = \mathbb{E}_{\mathcal{I}_t, \mathcal{I}_N, \epsilon \sim \mathcal{N}(0,I)} \left\| \epsilon - f_\theta(\mathcal{I}_k, \mathcal{I}_N, \bar{\alpha}_k) \right\|^2. \tag{9}$$

A. Data Collection

The data collection for the inpainting diffusion model involves a series of systematic steps to record, process, and visualize agent trajectories. Initially, we generated a dataset by training the OUTPACE method [9], independently for each maze environment, and recorded, for each episode, the position of the generated curriculum goal along with the agent's state sequence $\mathcal{S} = s_0, s_1, \ldots, s_T$, where s_t represents the agent's position at timestep t and T denotes the final timestep of the episode.

To visualize the agent's behavior, we scaled and discretized its continuous positions onto an $N \times M$ pixel grid, generating a heatmap for each episode. In the heatmap (see the input

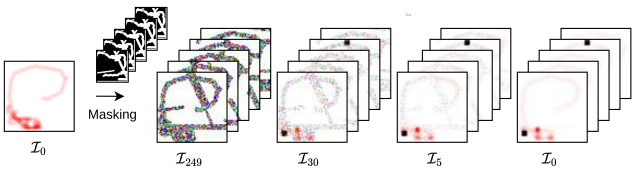


Fig. 2: A masked heatmap batch is denoised using the Palette diffusion model to reconstruct the agent's trajectory and curriculum goals.

image $\mathcal{I}_0$ on the left in Fig. 2 for an example), darker red shades indicate regions that were frequently visited by the agent during that episode, while lighter shades denote less-visited areas; unvisited regions appear white. The curriculum goal for the episode is highlighted as a black square.

B. Mask Configuration

Masks play a critical role in guiding the inpainting process by specifying which image regions should be reconstructed by the diffusion model. In our framework, masked regions are initialized with pure Gaussian noise while preserving original pixel values in unmasked areas. The Palette diffusion model fills the masked regions with Gaussian noise. During the reverse process, the diffusion model denoises the masked regions, conditioned on the masked pixels of the input image. For our experiments, we employ a path-based mask, illustrated above the arrow in Figure 2. This mask was designed to occlude regions corresponding to the agent's trajectory, the goal location, and a random subset of other pixels. By incorporating both trajectory context and an element of randomness, it encourages the diffusion model to generate curriculum goals that are contextually relevant to the agent's recent path, thereby producing meaningful exploration targets.

C. Diffusion Model for Curriculum Goal Generation

We employ a diffusion model to generate curriculum goals. This process involves two phases: training the model and then using it for inference during RL training. By training a diffusion model on a dataset containing agent trajectory heatmaps and their corresponding curriculum goals (generated using OUTPACE [9] in our experiments) for each episode i, we aim to capture the relationship between the agent's trajectory and the curriculum goals. In other words, the generated dataset guides the agent through curriculum goals, and the diffusion model is trained to learn this guidance based on the provided trajectory heatmaps. Note that we do not train OUTPACE jointly with the diffusion model; OUTPACE is used only to generate the dataset for this work. In principle, the same type of data could be collected by other means, without relying on OUTPACE. To generate curriculum goals, we use the trained diffusion model in our RL training setup (Section IV-D).

Given an agent trajectory heatmap image for an episode i, a path-based mask $\mathcal{M}$ is applied to this heatmap, occluding parts of the trajectory and random pixels, as described in Section IV-B. Unlike the diffusion model training, in this case, we do not have any goal information in the input image, and we expect the trained diffusion model to generate it for us. In this case, by only using the inference (reverse) process of the diffusion model, we feed the masked image to the model and obtain the generated agent's trajectory and curriculum goal for the next episode $i+1$. The masked region is initialized with Gaussian noise, while unmasked pixels retain their original values. The diffusion model then iteratively denoises the masked region, conditioned on the masked pixels of the input image. The final output is an image combining the curriculum goal (in the denoised region) with the agent's trajectory heatmap. Examples of the diffusion process are shown in Fig. 2. As shown in Fig. 2, the input image is given to the diffusion model. It is to be noted that the input image has no goal information. The mask is applied to the input image, and the diffusion model is used to gradually denoise the noise in the mask region. In the final output, a goal is generated and used as the curriculum goal for the next episode. The curriculum goal coordinate g_c, required by the RL agent, is extracted from the denoised image $\mathcal{I}_0$ via the `PixelToCoord` procedure. First, the image is converted to grayscale. Then, the black square marker is isolated by thresholding on low-intensity pixels, and its geometric centroid is computed in pixel coordinates. Finally, these pixel coordinates are mapped back to the continuous environment space by dividing by an image scale factor (see Appendix), which is the exact inverse of the `CoordToPixel` function used to generate the training heatmaps. This yields the curriculum goal $g_c = (x, y)$ for the next episode.

D. Goal Selection Strategies

Instead of using a single masked image as input to the diffusion model, we propose generating a series of path-based masked images. Given a heatmap, we apply k distinct binary masks to produce a batch of k partially masked inputs, where the number of masks, k, is a hyperparameter. Each masked heatmap serves as an independent input channel to the diffusion process. This multi-channel mechanism enables the model to independently reconstruct the masked regions for each of the k inputs, yielding a batch of k denoised heatmaps containing the goal information. This process is illustrated in Fig. 2. This batched masking strategy allows us to generate multiple candidate curriculum goals for the same trajectory by applying different sets of masks. Additionally, it enhances the robustness of our algorithm by mitigating failure cases where a denoised image may contain no goal information or multiple goals. A key challenge, however, is selecting the optimal curriculum goal from the generated candidates. To address this, we propose four strategies for curriculum goal selection, namely:

1) Averaging: Calculates the arithmetic mean of the coordinates of all candidate goals $\mathcal{G}_c$ and uses this average as the curriculum goal g_c for the next episode. The arithmetic mean is chosen because it is the simplest and most computationally efficient aggregation method.
2) AIM reward: We utilize the trainable AIM reward (Section III-B), to select a curriculum of goal candidates that maximizes the AIM reward. The AIM reward is a trainable

reward function designed to estimate the agent's proximity to the desired goal. Specifically, it quantifies the reward associated with the agent's current position relative to the desired goal.

3) Q-function: This strategy utilizes the agent's learned state-action value function, Q, and guides the agent by selecting a curriculum goal from the generated candidates $\mathcal{G}_c$. The Q-function provides an estimate of the expected cumulative reward. For each candidate goal $g_c \in \mathcal{G}_c$, we compute a score derived from the Q-function, potentially evaluated over the agent's recent trajectory $(s_{0:T}, a_{0:T})$, denoted as $Q(s_{0:T}, a_{0:T}, g_c)$. These scores are normalized into a probability distribution using the softmax function, and a curriculum goal g_c is then sampled according to these probabilities. This method favors goals that yield a higher Q-value, thereby selecting objectives aligned with the agent's learned policy and value estimations, which promotes efficient learning.
4) AIM reward & Q-function: Both the AIM reward and the Q-function can be used to select the curriculum goal for the next episode. Since the AIM reward estimates the immediate reward while the Q-function predicts the cumulative reward, they can be combined with different weighting terms to effectively select the curriculum goal.

The overall training pipeline is illustrated in Fig. 1. Initially, a trajectory dataset, comprising state sequences and corresponding curriculum goals, is generated by training a baseline agent with the OUTPACE method [9] independently for each maze environment. This data is then transformed into an image dataset; each sample is a trajectory heatmap visualizing the agent's path and marked with the episode's curriculum goal. A masking mechanism (Section IV-B) is applied to each heatmap, typically occluding portions of the trajectory and the goal location, thereby framing the task as an image inpainting problem. A diffusion model is subsequently trained on this masked dataset to reconstruct the occluded regions, learning the relationship between trajectory heatmaps and corresponding curriculum goals. Once trained, this diffusion model is employed during the PIDCRL training phase (Section IV-C). In each training episode, the agent's trajectory is converted into a heatmap image, which excludes explicit goal information. Given a heatmap, k distinct binary masks are applied to generate a batch of k partially masked inputs. These masked heatmaps serve as input to the trained diffusion model, which generates candidate curriculum goals for the next episode based on the observed trajectory pattern. Finally, an optimal curriculum goal is selected from the candidates using the mechanism described in Section IV-D and utilized in the PIDCRL training process.

Algorithm 1 details the PIDCRL steps. The method initializes the off-policy algorithm, replay buffer, networks, and sets the initial curriculum goal equal to the desired goal (line 3). In each episode, the agent interacts with the environment using the current curriculum goal (lines 7-9). The agent's trajectory is then converted to a heatmap image and passed

Algorithm 1 PIDCRL process

1: **Input:** No. of episodes E, no. of steps T
2: Select an off-policy algorithm $\mathbb{A}$ ▷ In our case, $\mathbb{A}$ is SAC
3: Initialize replay buffer $\mathcal{B} \leftarrow \emptyset$, $g_c \leftarrow \{g_d\}$ and networks Q_ϕ, π_ψ, r_φ
4: **for** episode $= 0 \dots E$ **do**
5: Sample initial state s_0
6: **for** $t = 0 \dots T$ **do**
7: $a_t = \pi(s_t, g_c)$
8: Execute a_t, obtain next state s_{t+1}
9: Store transition $(s_t, a_t, r_t, s_{t+1}, g_c)$ in $\mathcal{B}$
10: $\mathcal{I}_0 = \texttt{CoordtoPixel}(s_{0:T})$ ▷ Convert the trajectory to a heatmap image
11: $\mathcal{G}_c \leftarrow \texttt{PaletteCurriculumGenerator}(\mathcal{I}_0)$
12: Find $g_c = \texttt{GoalSelector}(\mathcal{G}_c)$
13: Sample a minibatch b from replay buffer $\mathcal{B}$
14: Update Q and π with b to minimize $\mathcal{L}_Q$ and $\mathcal{L}_\pi$
15: Update the AIM reward function r_φ
16: $success \leftarrow 0$ ▷ Success rate
17: Sample a desired goal $g_d \sim \mathcal{G}$
18: **for** i $= 1 \dots n_{testrollout}$ **do**
19: $a_t = \pi(s_t, g_d)$
20: Execute a_t, obtain next state s_{t+1} and reward r_t
21: **if** $|\phi(s_{t+1}) - g_d| \leq \kappa$ **then**
22: $success =\leftarrow success + 1/n_{testrollout}$
23: **return** $success$

to `PaletteCurriculumGenerator` (line 11) to produce candidate curriculum goals $\mathcal{G}_c$. From these, one goal g_c is selected using the `GoalSelector` (line 12) for the next episode. Networks are updated using minibatches from the replay buffer (lines 14-15). Success rate is measured by test rollouts that verify if the agent reaches the desired goal within threshold κ (lines 16-22).

V. Experiments

To validate the effectiveness of our approach, PIDCRL, we performed evaluations in three challenging maze tasks, namely PointUMaze, PointNMaze, and PointSpiralMaze, simulated using MuJoCo [25]. We benchmarked PIDCRL against a comprehensive set of ten established CRL baselines: ACL [22], GoalGAN [13], HGG [12], ALP-GMM [11], VDS [21], SPRL [10], PLR [8], CURROT [17], GRADIENT [18], and OUTPACE [9]. All results are reported over five random seeds. The implementation code is publicly available at https://github.com/farukoruc/DiffusionOutpace. This repository also includes full-resolution plots, extended quantitative results, and additional analyses (located in the `supplementary/` folder).

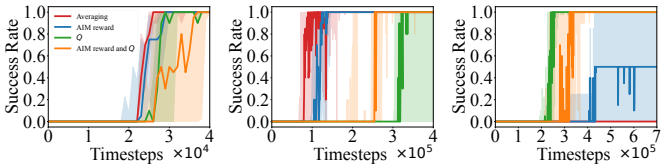
A. Goal Selection Mechanism

We first conduct an ablation study to investigate the four goal selection mechanisms discussed in Section IV-D. Results are reported in Fig. 3. The performance differences observed among the four goal selection mechanisms mainly stem from the interaction between each mechanism's underlying strategy and

TABLE I: No. of timesteps (rounded) to reach success rate 1.0 (average $\pm$ std. dev. across 5 runs). Only methods reaching a success rate 1.0 in all runs within $1e6$ timesteps per environment are included; methods failing in any run (seed) for a given environment are excluded.

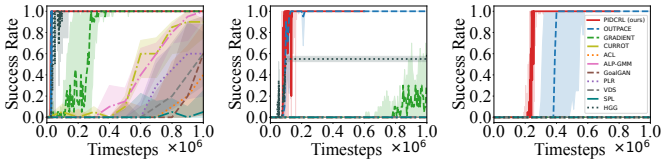
Algorithm	PointUMaze	PointNMaze	PointSpiralMaze
PIDCRL (Ours)	25400 $\pm$ 1020	87625 $\pm$ 21702	234125 $\pm$ 23089
OUTPACE	29800 $\pm$ 4166	113333 $\pm$ 24267	396875 $\pm$ 111451
HGG	48750 $\pm$ 24314	-	-
GRADIENT	263431 $\pm$ 114795	-	-

the specific complexities of the maze environments. Examples of curriculum goals generated by AIM reward & Q-function strategy across the different maze environments are visualized in Fig.5.



(a) PointUMaze **(b)** PointNMaze **(c)** PointSpiralMaze

Fig. 3: Performance of different goal selection strategies for PIDCRL in maze tasks.



(a) PointUMaze **(b)** PointNMaze **(c)** SpiralMaze

Fig. 4: Comparison of test success rates for PIDCRL and baseline CRL algorithms across three maze environments. Curves represent the mean success rate over five random seeds, with shaded areas indicating the standard deviation.

Averaging: This simple strategy computes the geometric mean of the coordinates of all candidate goals $\mathcal{G}_c$ generated by the diffusion model. While computationally inexpensive, it lacks awareness of the environment's geometry and constraints. As a result, averaging can yield curriculum goals located within impassable regions (e.g., walls). In simpler environments like PointUMaze and PointNMaze, this limitation is less detrimental. The maze structures are less convoluted, and the agent's interaction with the environment (e.g., collisions) can often compensate for suboptimal goal placement. An averaged goal might still provide a useful directional cue, guiding exploration towards the target region. However, in the complex PointSpiralMaze (Fig. 3c), which demands precise navigation, Averaging performs poorly because infeasible or poorly placed goals can easily get the agent stuck in unproductive loops.

AIM reward: This strategy selects the candidate goal $g \in \mathcal{G}_c$ that minimizes the learned function $r^{\pi}_{\varphi}(s_0, g)$, derived from the AIM framework (Section III-B). This function estimates the distance to reach a goal g from the initial state s_0, without explicitly encoding the maze geometry. In simpler

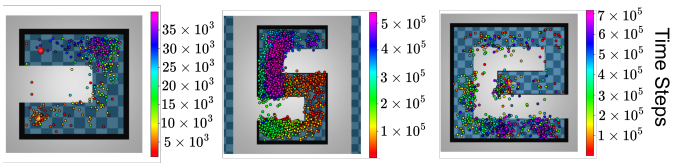


Fig. 5: Curriculum goals generated by PIDCRL over timesteps.

mazes (PointUMaze, PointNMaze, Fig. 3a-b), this approach performs well, rapidly guiding the agent towards the target region by selecting goals considered most accessible. Minimizing this learned proximity serves as a reasonable heuristic for progressive goal setting in less structured environments. However, its performance degrades significantly in the complex PointSpiralMaze (Fig. 3c). Because the AIM proximity function lacks awareness of the intricate maze structure and long-term path feasibility, selecting goals solely based on minimizing it can lead the agent towards infeasible regions.

Q-function: This strategy leverages the agent's learned state-action value function Q_ϕ to score candidate goals, selecting goals predicted to yield high cumulative future rewards. The effectiveness of this strategy hinges on the accuracy of the Q-function, which improves as the agent accumulates environmental experience during training. This explains the initially slower convergence observed for the Q-function selection mechanism in PointUMaze and PointNMaze (Fig. 3a-b). However, as the value estimates become more reliable, this method excels at identifying goals that are both feasible and strategically advantageous for long-term reward maximization. This makes it particularly effective in complex environments like PointSpiralMaze (Fig. 3c), where understanding the environmental structure and long-term consequences is crucial.

AIM reward & Q-function: This combined strategy seeks to balance the strengths of the AIM reward and Q-function. It selects goals based on a weighted sum of the negative AIM proximity (favoring accessible goals) and the Q-value (favoring strategically valuable goals). By incorporating both, this approach aims for robustness across different levels of complexity of the environment. As shown in Fig. 3, this combined strategy often achieves a performance between AIM reward and Q-function. It represents a practical compromise, leveraging both learned proximity and value estimation for curriculum goal selection.

B. Comparison with State-of-the-art methods

The results, illustrated in Fig. 4 and detailed in Table I, highlight PIDCRL's effectiveness. For comparison with baseline methods, we utilize for PIDCRL the best-performing goal selection strategy for each respective environment: namely, Averaging for PointUMaze and PointNMaze, and the Q-function for PointSpiralMaze.

Figure 4 shows that PIDCRL consistently achieves higher success rates more rapidly than the ten baseline methods across all three maze environments. Table I further quantifies this, showing that PIDCRL reaches a 1.0 success rate significantly faster, an advantage that is more pronounced in

complex environments. Notably, strong baselines like HGG and GRADIENT, while competitive in PointUMaze, often fail to achieve complete success in the more challenging mazes within the given timeframe, underscoring PIDCRL's robustness and efficiency.

VI. Conclusion and Limitations

PIDCRL formulates curriculum-goal generation as an image inpainting problem by training a Palette diffusion model on OUTPACE-generated trajectory–goal pairs. During RL, it uses the model to complete masked trajectory heatmaps, generate candidate goals, and select an appropriate goal to guide the agent. The experimental results show that PIDCRL consistently generates challenging yet achievable goals, matching or outperforming state-of-the-art CRL methods in three complex maze environments. Despite its effectiveness, PIDCRL has several limitations. It relies on an initially generated dataset for diffusion model training, and diffusion-based inference introduces additional computational overhead during RL. Moreover, the current trajectory representation is restricted to two-dimensional environments, and extending it to higher-dimensional state spaces may require larger datasets and increased computational resources. Future work will explore data generation without synthetic supervision and more efficient diffusion architectures. It will also investigate extensions to three-dimensional and more complex environments.

Acknowledgment

The authors gratefully acknowledge the funding of this project by computing time provided by the Paderborn Center for Parallel Computing (PC^2). This work was partially supported by the Horizon Europe Grant Agreement No. 101136056.

References

[1] François-Lavet, Vincent and Henderson, Peter and Islam, Riashat and Bellemare, Marc G and Pineau, Joelle and others. An introduction to deep reinforcement learning. *Foundations and Trends® in Machine Learning*, 11(3-4):219–354, 2018.

[2] Efe Camci, Domenico Campolo, and Erdal Kayacan. Deep reinforcement learning for motion planning of quadrotors using raw depth images. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7, 2020.

[3] Mnih, Volodymyr and Kavukcuoglu, Koray and Silver, David and Rusu, Andrei A and Veness, Joel and Bellemare, Marc G and Graves, Alex and Riedmiller, Martin and Fidjeland, Andreas K and Ostrovski, Georg and others. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015.

[4] Feng, Shuo and Sun, Haowei and Yan, Xintao and Zhu, Haojie and Zou, Zhengxia and Shen, Shengyin and Liu, Henry X. Dense reinforcement learning for safety validation of autonomous vehicles. *Nature*, 615(7953):620–627, 2023.

[5] Erdi Sayar, Zhenshan Bing, Carlo D'Eramo, Ozgur S Oguz, and Alois Knoll. Contact Energy Based Hindsight Experience Prioritization. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5434–5440. IEEE, 2024.

[6] Sayar, Erdi and Iacca, Giovanni and Knoll, Alois. Multi-Objective Evolutionary Hindsight Experience Replay for Robot Manipulation Tasks. In *Genetic and Evolutionary Computation Conference*, page 403–411, New York, NY, USA, 2024. Association for Computing Machinery.

[7] Wong, Esther and Leung, Kin and Field, Tony. State-space decomposition for reinforcement learning. Technical report, Department of Computing, Imperial College London, London, UK, 2021.

[8] Jiang, Minqi and Grefenstette, Edward and Rocktäschel, Tim. Prioritized Level Replay. In *International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 4940–4950. PMLR, 2021.

[9] Daesol Cho and Seungjae Lee and H. Jin Kim. Outcome-directed Reinforcement Learning by Uncertainty & Temporal Distance-Aware Curriculum Goal Generation. In *International Conference on Learning Representations*, 2023.

[10] Pascal Klink and Hany Abdulsamad and Boris Belousov and Carlo D'Eramo and Jan Peters and Joni Pajarinen. A Probabilistic Interpretation of Self-Paced Learning with Applications to Reinforcement Learning. *Journal of Machine Learning Research*, 22(182):1–52, 2021.

[11] Portelas, Rémy and Colas, Cédric and Hofmann, Katja and Oudeyer, Pierre-Yves. Teacher algorithms for curriculum learning of Deep RL in continuously parameterized environments. In *Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pages 835–853. PMLR, 2020.

[12] Ren, Zhizhou and Dong, Kefan and Zhou, Yuan and Liu, Qiang and Peng, Jian. Exploration via Hindsight Goal Generation. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[13] Florensa, Carlos and Held, David and Geng, Xinyang and Abbeel, Pieter. Automatic goal generation for reinforcement learning agents. In *International Conference on Machine Learning*, pages 1515–1528. PMLR, 2018.

[14] Saharia, Chitwan and Chan, William and Chang, Huiwen and Lee, Chris and Ho, Jonathan and Salimans, Tim and Fleet, David and Norouzi, Mohammad. Palette: Image-to-image diffusion models. In *ACM SIGGRAPH 2022 Conference Proceedings*, pages 1–10, 2022.

[15] Sayar, Erdi and Iacca, Giovanni and Oguz, Ozgur S and Knoll, Alois. Diffusion-based Curriculum Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 37, pages 97587–97617, 2024.

[16] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

[17] Klink, Pascal and Yang, Haoyi and D'Eramo, Carlo and Peters, Jan and Pajarinen, Joni. Curriculum Reinforcement Learning via Constrained Optimal Transport. In *International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 11341–11358. PMLR, 2022.

[18] Huang, Peide and Xu, Mengdi and Zhu, Jiacheng and Shi, Laixi and Fang, Fei and ZHAO, DING. Curriculum Reinforcement Learning using Optimal Transport via Gradual Domain Adaptation. In *Advances in Neural Information Processing Systems*, volume 35, pages 10656–10670. Curran Associates, Inc., 2022.

[19] Goodfellow, Ian and Pouget-Abadie, Jean and Mirza, Mehdi and Xu, Bing and Warde-Farley, David and Ozair, Sherjil and Courville, Aaron and Bengio, Yoshua. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[20] Schaul, Tom and Horgan, Daniel and Gregor, Karol and Silver, David. Universal Value Function Approximators. In *International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1312–1320. PMLR, 2015.

[21] Zhang, Yunzhi and Abbeel, Pieter and Pinto, Lerrel. Automatic Curriculum Learning through Value Disagreement. In *Advances in Neural Information Processing Systems*, volume 33, pages 7648–7659. Curran Associates, Inc., 2020.

[22] Alex Graves and Marc G. Bellemare and Jacob Menick and Rémi Munos and Koray Kavukcuoglu. Automated Curriculum Learning for Neural Networks. In *International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1311–1320. PMLR, 2017.

[23] Durugkar, Ishan and Tec, Mauricio and Niekum, Scott and Stone, Peter. Adversarial Intrinsic Motivation for Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 34, pages 8622–8636. Curran Associates, Inc., 2021.

[24] Ho, Jonathan and Jain, Ajay and Abbeel, Pieter. Denoising Diffusion Probabilistic Models. In *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851. Curran Associates, Inc., 2020.

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