Generative Proto-Sequence: Sequence-Level Decision Making for Long-Horizon Reinforcement Learning

Anonymous authors
Paper under double-blind review

Abstract

Deep reinforcement learning (DRL) methods often face challenges in environments characterized by large state spaces, long action horizons, and sparse rewards, where effective exploration and credit assignment are critical. We introduce Generative Proto-Sequence (GPS), a novel generative DRL approach that produces variable-length discrete action sequences. By generating entire action sequences in a single decision rather than selecting individual actions at each timestep, GPS reduces the temporal decision bottleneck that impedes learning in long-horizon tasks. This sequence-level abstraction provides three key advantages: (1) it facilitates more effective credit assignment by directly connecting state observations with the outcomes of complete behavioral patterns; (2) by committing to coherent multi-step strategies, our approach facilitates better exploration of the state space; and (3) it promotes better generalization by learning macro-behaviors that transfer across similar situations rather than memorizing state-specific responses. Extensive evaluations on mazes of varying sizes and complexities demonstrate that GPS consistently outperforms leading action repetition and temporal methods, where it converges faster and achieves higher success rates across all environments.

1 Introduction

Deep reinforcement learning (DRL) has demonstrated impressive performance across diverse applications (Mnih et al., 2015; Silver et al., 2016; Levine et al., 2016). However, significant challenges remain when dealing with environments characterized by large state spaces, long-range tasks, and sparse rewards. In such contexts, traditional DRL methods that select actions sequentially often suffer from inefficient exploration and poor credit assignment (Mesnard et al., 2020; Raileanu & Rocktäschel, 2020; Ecoffet et al., 2021), leading to difficulties in learning effective policies for tasks that require coordinated, multi-step strategies. These challenges are further exacerbated by sparse reward signals, whose limited feedback hinders the agent's ability to discover and reinforce successful behaviors (Arjona-Medina et al., 2019; Hung et al., 2019).

Recent research efforts have attempted to address these challenges using diverse techniques such as hierarchical DRL (Kulkarni et al., 2016; Xu et al., 2022), temporal abstraction (Biedenkapp et al., 2021; Zhang et al., 2022b; Saanum et al., 2023; Patel & Siegelmann, 2024), sequence modeling (Chen et al., 2021; Janner et al., 2021; Giladi & Katz, 2023), and action repetition strategies (Srinivas et al., 2017; Sharma et al., 2017; Dabney et al., 2020). By creating sub-tasks or capturing higher-level behavioral patterns (Rosete-Beas et al., 2023; Vezzani et al., 2022; Wang et al., 2023), these techniques aim to reduce decision frequency and enhance learning efficiency in long-horizon tasks. Although these methods offer promising results, they often require careful sub-task design (Ajay et al., 2023), reward shaping (Liu et al., 2022), or complex training procedures (Seo & Abbeel, 2024b; Rosete-Beas et al., 2023). While there are temporal abstraction methods that generate multi-step action sequences, they often depend on iterative rollouts, autoregressive decoding, or model-based simulations. These solutions introduce computational overhead and restrict the ability to efficiently generate diverse action sequences. To our knowledge, no method supports the generation of *coherent, variable-length action sequences directly from state observations in a single decision step.*

In this study, we propose Generative Proto-Sequence (GPS), a novel actor-critic architecture capable of producing variable-length action sequences. Instead of actions, our Actor generates a *proto-sequence* embedding, which is then decoded into a discrete action sequence using a Decoder component. The Critic evaluates the state and the entire generated sequence jointly, with gradients flowing from the critic through the Decoder to the Actor, facilitating end-to-end learning of strategic, multi-step action sequences. This design enables the agent to generate and execute complex exploratory behaviors in a single decision, enhancing both generalization and long-horizon credit assignment.

We evaluated GPS on a large set of challenging maze environments with varying sizes and configurations, including rooms, corridors, and randomly generated obstacles. Our results demonstrate that GPS consistently learns more efficiently, generalizes better to novel maze layouts, and significantly outperforms leading baselines in terms of success rate and convergence speed, particularly in large and complex mazes. Our contributions are as follows:

- We introduce a novel architecture that enables end-to-end generation and evaluation of variable-length discrete action sequences, facilitating improved credit assignment and exploration.
- We demonstrate that producing multi-step action sequences in a single shot leads to superior generalization and faster convergence, particularly in large and complex environments.
- We provide extensive empirical results on challenging maze benchmarks, showing significant improvements over top-performing action repetition and temporal methods baselines in metrics such as convergence speed and success rate.

2 Related Work

2.1 Temporal Abstraction Through Action Repetition

Early works in temporal abstraction explored repeating single primitive actions to extend decision horizons. Recent research in DRL has produced various approaches for performing multiple actions as a single block. Earlier works (Srinivas et al., 2017; Sharma et al., 2017) introduced frameworks for dynamic action selection and repetition, though their repetition policies operated independently from chosen actions, limiting strategic development.

DAR (Srinivas et al., 2017) augments discrete action spaces by duplicating each base action with multiple repetition rates. While this expansion can improve learning in environments benefiting from temporal abstraction, it produces an inefficient representation—duplicated actions are treated as unrelated, preventing the agent from exploiting their shared underlying behavior and leading to slower learning and imbalanced trade-offs between coarse and fine control. FiGAR (Sharma et al., 2017) addresses this by decoupling behavior and repetition into two jointly trained policies; however, the repetition policy operates independently from the chosen action, limiting the development of nuanced, action-specific repetition strategies.

The authors of Dabney et al. (2020) proposed an exploration strategy repeating actions for random durations to reduce inefficient dithering. Temporl (Biedenkapp et al., 2021) advanced this by enabling agents to determine both action and repetition duration, improving learning efficiency. However, its hierarchical structure artificially decouples action selection from duration determination. Despite showing promise, these studies share a limitation: *temporal abstraction is achieved solely through simple repetition of primitive actions*, without generating coherent, variable-length action sequences.

2.2 Multi-Step Action Sequence Generation

Beyond single-action repetition, several methods focus on generating and partially committing to multi-step action sequences. The authors of Zhang et al. (2022a) introduced a generative planning method (GPM) that produces multi-step plans. Since GPM is trained by maximizing value, the plans generated from it can be regarded as intentional action sequences to reach high-value states and improve sample efficiency. PrAC (Coad et al., 2022) enables agents to generate n-step plans and commit to them while being predictable, balancing adaptability and control stability. The work of Saanum et al. (2023) incentivizes compressible

action sequences by integrating sequence priors, while Patel & Siegelmann (2024) introduced a model-based sequence RL framework (SRL) reducing decision frequency through action chunking.

Despite recent progress, most existing methods for generating multi-step action sequences still face major limitations. Many rely on heavy processes such as iterative rollouts, autoregressive decoding, or model-based simulation, which can be slow and inflexible (Li et al., 2024; Li, 2023; Zhang et al., 2025). For example, methods like PrAC and SRL use learned environment models for both planning and training, adding extra model-based complexity (Kumar et al., 2024; Luo et al., 2024). To stay adaptable, some approaches also use external switching mechanisms or mid-sequence re-planning, as seen in GPM and PrAC. This treats long-term planning as an add-on to a step-by-step framework rather than as a core design principle. As a result, sequence generation and evaluation are often optimized separately, which can lead to poor credit assignment (Dai et al., 2018). One case is the use of handcrafted regularization, such as rewarding shorter or more "compressible" sequences (Saanum et al., 2023). However, when objectives are split in this way, it becomes unclear whether failures come from a bad plan or from breaking the secondary constraint, making end-to-end training harder and reducing stability during execution.

2.3 Temporal Abstraction Using Hierarchies and LLMs

Hierarchical methods have advanced multi-action decision-making through skill discovery and sequencing. TACO-RL (Rosete-Beas et al., 2023) learns latent skills from unstructured data for long-horizon tasks. ASPiRe (Xu et al., 2022) accelerated RL by combining specialized skill priors. The work of Vezzani et al. (2022) introduced a skill scheduler sequencing pretrained skills, while SHRL (Wang et al., 2023) combined high-level policies with low-level skills for visual navigation. These approaches improve temporal abstraction by leveraging reusable skills rather than primitive actions.

Recent works have leveraged large language models and value-based reinforcement learning methods with action discretization for action sequence generation. CQN-AS (Seo & Abbeel, 2024a) proposed a value-based algorithm learning precise value functions from noisy action sequences. AlphaMaze (Dao & Vu, 2025) improved LLMs' spatial reasoning by combining supervised fine-tuning with policy optimization.

Our work draws inspiration from Dulac-Arnold et al. (2015), who generated embedding representations of proto-sequences mapped to discrete actions. In GPS, we propose key improvements: our approach is fully differentiable and trainable end-to-end, unlike Dulac-Arnold et al. (2015) whose k-nearest neighbors mapping broke the computation graph. Additionally, by using a VAE-based decoder instead of clustering, we automatically create sequence representations without manual embedding design. This enables efficient generation of coherent, variable-length action sequences that extend beyond simple repetition or skill sequencing.

3 Method

Overview. Our proposed approach is presented in Figure 1. GPS consists of three components: *Actor*, *Proto-Sequence Decoder (PSD)*, and *Critic*. The Actor receives the current state as input, and produces a *proto-sequence* – an embedding-based representation of a sequence of actions. The PSD receives the proto-sequence as input, and translates it into a discrete set of actions (e.g., $a_t, a_{t+1}, ..., a_{t+L}$), which are then executed sequentially by the agent. Finally, our Critic receives the sequence and predicts the expected cumulative reward obtained from its execution.

GPS differs from previous studies in several important aspects. First, unlike previous studies (Dulac-Arnold et al., 2015), it is end-to-end differentiable and does not require training workarounds. Secondly, our VAE decoder produces more diverse and flexible action sequences than autoregressive or model-based approaches, and also does so in an efficient, one-shot manner. Thirdly, sequence generation and evaluation are learned jointly, without regularization or switching mechanisms, thus improving credit assignment. Finally, by committing to the entire sequence (unlike the frequent re-evaluation of Zhang et al. (2022a)) we reduce execution overhead and increase behavioral predictability by forcing GPS to learn robust policies.

Another important aspect of our proposed approach is its ability to generate action sequences that differ from those on which it was trained. By creating novel sequences, GPS does not simply "memorize" a fixed

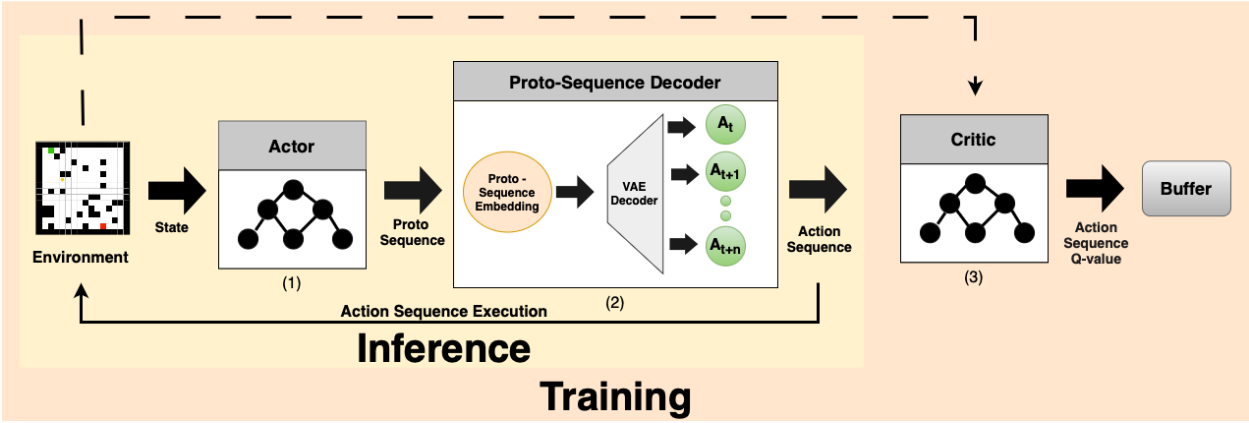

Figure 1: The three components of our proposed approach: (1) The Actor encodes the current state to produce a proto-sequence embedding. (2) The Decoder translates this latent embedding into a variable-length discrete action sequence. (3) The Critic evaluates the state-action-sequence pair and assigns it a Q-value representing the expected cumulative reward (3). During inference, only the actor and decoder components are used.

set of actions, but is able to generalize to larger action spaces. We elaborate on GPS's capacity to produce novel sequences in Section 5.1. Despite using a frozen decoder pre-trained on a limited set of sequences, GPS can generate action sequences that differ from those in the decoder's training data. The ability to generalize is important to our method, since sequence memorization is not applicable to domains with large action spaces. We elaborate on GPS's capacity to produce novel sequences in Section 5.1.

3.1 The Actor

The Actor serves as GPS's policy network. Given a state s_t, the Actor analyzes the input and outputs a proto-sequence embedding $k = \pi_{\theta^\pi}(s_t)$, where θ^π and π are the parameters of the Actor's neural network and the current policy, respectively. The proto-sequence k is a latent embedding of a sequence of actions, represented as a vector in the embedding space $k \in \mathbb{R}^d$, where d is the dimensionality of the embedding space. This representation provides our Actor with significant flexibility, as it can create action sequences of varying length using a fixed-size representation.

The proto-sequence is next used by the PSD to produce a discrete sequence of actions, and this sequence is evaluated by the Critic (Section 3.3). The parameters θ^π of the Actor are then updated using an actor-critic approach analogous to the Deep Deterministic Policy Gradient (DDPG) algorithm (Lillicrap et al., 2016), leveraging the learning signal provided by the Critic. Specifically, the actor's parameters θ^π are adjusted to produce proto-sequence embeddings k that maximize the expected cumulative reward estimated by the critic, $Q_{\theta^Q}(s_t, \mathbf{a})$. This optimization is achieved by updating θ^π to minimize the negative Q-value provided by the critic $-Q_{\theta^Q}(s_t, g_{\theta^\omega}(\pi_{\theta^\pi}(s_t)))$, using backpropagated gradients from the output of the critic network Q_{θ^Q}. These gradients pass through the decoder network g_{θ^ω} and subsequently through the actor network π_{θ^π}, enabling the update of the latter's parameters θ^π.

3.2 The Proto-Sequence Decoder

The goal of the PSD is to translate the latent proto-sequence k generated by the Actor into a sequence of executable actions in the original action space $\{a_t, a_{t+1}, ..., a_{t+L}\} \in A$. We define the PSD as a function $g_{\theta^\omega} : K \rightarrow A'^{L_{max}}$, parameterized by θ^ω, where A' extends A with an EOS token to handle variable-length sequences within a fixed-length format, padding shorter sequences as needed. This function maps from the latent proto-sequence space K to sequences of fixed length L_{max}.

We use a Variational Autoencoder (VAE) (Kingma & Welling, 2013) as our PSD. We train the architecture on a diverse set of synthetic action sequences of varying lengths. For detailed information on the generation process of these sequences, see Appendix L.1. After training, we discard the VAE's encoder and retain only the learned decoder network g_{θ^ω}. The decoder is integrated into our agent architecture, transforming the Actor's latent proto-sequence embeddings into sequences of discrete actions. GPS will then execute the full sequence, without changes or early stopping. We chose VAE for its efficiency, ability to generate complete sequences in a single step, and its structured latent space that enables smooth interpolation and principled probabilistic modeling.

We pre-train the PSD and keep its parameters fixed while jointly training the actor and critic. We had several reasons for this separation: *a)* simplicity and robustness – separately training the Decoder reduces the number of moving parts in our architecture, enabling faster training. Separate training also prevents a 'moving target' scenario, where multiple components adapt at the same time, compromising training stability Sutton et al. (1999); *b)* Diversity and prevention of mode collapse: by training the Decoder separately, we have full control of its training set. We use a diverse training set that ensures that the Decoder can generate action sequences of all types. In addition, by freezing the Decoder after its training, we prevent a scenario where this component "forgets" how to generate specific actions during the other components' training; *c)* Transferability – by ensuring our Decoder can generate action sequences of all types, we can train it once and then use it in all tasks with this particular action space.

3.3 The Critic

The goal of our Critic is similar to the role of the critic in an actor-critic architecture. The Critic receives the current state s_t and the one-hot encoded discrete actions sequence $\mathbf{A} = (a_t, a_{t+1}, \ldots, a_{t+L})$ produced by the PSD. It then attempts to predict $Q_{\theta^Q}(s_t, \mathbf{A})$, which represents the cumulative discounted reward obtained by executing $\mathbf{A}$ and following the policy after the end of the sequence:

$$Q_{\theta^Q}(s_t, \mathbf{A}) \approx \mathbb{E}_{\pi,P}\left[\sum_{k=0}^{L-1} \gamma^k r_{t+k} + \gamma^L V^\pi(s_{t+L})\right]$$

where V^π is the value function under policy π, θ^Q are the Critic's parameters, and $L = \mathit{eff_len}(\mathbf{A})$ denotes the effective length of the action sequence $\mathbf{A}$.

The Critic's parameters θ^Q are updated by minimizing the Mean Squared Error (MSE) loss against a Temporal Difference (TD) target y_t:

$$L(\theta^Q) = \mathbb{E}_{(s_t, \mathbf{A}, \text{rewards}, s_{\text{next}})}\left[(Q(s_t, \mathbf{A}; \theta^Q) - y_t)^2\right]$$

The target y_t is constructed from the sum of discounted rewards $R_t(\mathbf{A})$ obtained by executing sequence $\mathbf{A}$, and the discounted value of the subsequent state $s_{t+\mathrm{L}}$, estimated using target Actor ($\text{Actor}_{\text{target}}$) and target Critic ($Q_{\text{target}}$) networks:

$$y_t = R_t(\mathbf{A}) + \gamma^L Q_{\text{target}}(s_{t+L}, \text{PSD}(\text{Actor}_{\text{target}}(s_{t+L})); \theta^{Q-})$$

This update mechanism, which relies on TD errors and target networks, is characteristic of many actor-critic algorithms, and shares similarities with methods such as DDPG (Lillicrap et al., 2016). While the Critic learns to accurately predict $Q(s_t, \mathbf{A}; \theta^Q)$, the Actor is trained to produce proto-sequences that, when decoded by the PSD, maximize this predicted Q-value.

3.4 Training Set Augmentation Using Sequence Subsets and Inference

To enhance learning efficiency and improve credit assignment, our training procedure leverages reward information from subsequences of each executed action sequence. For each sequence $\mathbf{A} = (a_t, \ldots, a_{t+L})$ of length L, we extract transitions corresponding to multiple contiguous subsequences $(a_i, \ldots, a_j)$ where $t \leq i < j < t + L$. For each such subsequence starting from an intermediate state s_i, we calculate the accumulated discounted reward obtained during its execution. This process effectively generates multiple learning samples of varying temporal lengths from a single interaction sequence, enriching the training data.

Table 1: The setup and properties of the mazes used in the evaluation.

Environment	Dist. from start to goal	Train Set Size	Train Optimal Avg. Path	Val Set Size	Val Optimal Avg. Path	Test Set Size	Test Optimal Avg. Path
8x8	[1 - 14]	100	5.14	100	5.49	1000	5.31
16x16	[16 - 26]	100	18.04	100	18.0	1000	17.98
16x16_obstacles_15%	[20 - 30]	100	21.02	100	21.35	210	21.31
16x16_obstacles_25%	[20 - 30]	100	21.63	100	21.34	400	21.54
16x16_rooms	[20 - 30]	100	20.93	100	21.01	585	21.02
16x16_corridors	[10 - 30]	100	12.84	100	12.76	545	13.14
24x24	[20 - 30]	100	23.39	100	23.26	1000	23.56
24x24_obstacles_15%	[10 - 20]	100	15.04	100	14.58	1000	14.73
24x24_obstacles_25%	[10 - 20]	100	15.8	100	15.05	1000	15.11

The subsequence extraction strategies for these state-subsequence-reward tuples, which we add to the replay buffer, include two primary approaches: (1) prefix extraction, which fixes the starting state while varying the end point, and (2) suffix extraction, which fixes the goal state while varying the starting point. This bidirectional approach diversifies the replay buffer with different time scales and enables the Critic to learn value estimates $Q_{\theta^Q}(s_i, (a_i, \ldots, a_j))$ for sequences of different lengths concurrently. As shown in our analysis in Section 5.2, these extraction strategies significantly accelerate learning and improve overall performance. It is important to note that during inference (test time), our architecture does not utilize the Critic component, since no training takes place. Instead, the Actor and PSD produce the action sequence, and the latter is executed in full.

4 Experiments and Results

4.1 Evaluation Environment

Mazes are a foundational benchmark in DRL research, commonly used to evaluate an agent's ability to perform complex sequential decision-making and navigation tasks. Their structured yet variable environments provide a controlled setting for evaluating generalization, exploration, and memory, which are central to DRL performance (Pašukonis et al., 2023). We use four types of mazes in our evaluation:

- **Empty.** These mazes have no walls or obstacles, except for their boundaries.
- **Sparse Obstacles.** This setup has randomly placed obstacles in K% of the cells of each maze (e.g., 15%).
- **Rooms.** This setup consists of four large rooms with small doors between them. We also add randomly placed obstacles in 5% of open cells.
- **Corridors.** These mazes have only narrow corridors for the agent to navigate.

Our evaluation uses fully observable MDPs where the agent has complete visibility of the entire maze grid, including its position, goal location, and all obstacles.

Similarly to Dao & Vu (2025), we use an LLM to produce the code used in our maze generation. Our code, as well as the mazes generated for our evaluation, are available in the appendix. All information on our generated mazes is presented in Table 1. For each maze size and type, the table presents: *a)* the sizes of our training, validation, and test sets, *b)* the range for the distance between the start and goal positions, and *c)* the average length of the optimal path.

4.2 Baselines & Evaluated Methods

We evaluate two versions of GPS and three discrete-action baselines: *DQN*, *TempoRL*, and *DAR*. Full implementation details of our approach are included in the Appendix.

GPS: Our primary approach generates action sequences using a VAE-based decoder with Gumbel-Softmax sampling. This stochastic mechanism applies a temperature-controlled softmax to produce action distributions that maintain differentiability while approximating discrete samples. The Gumbel-Softmax technique

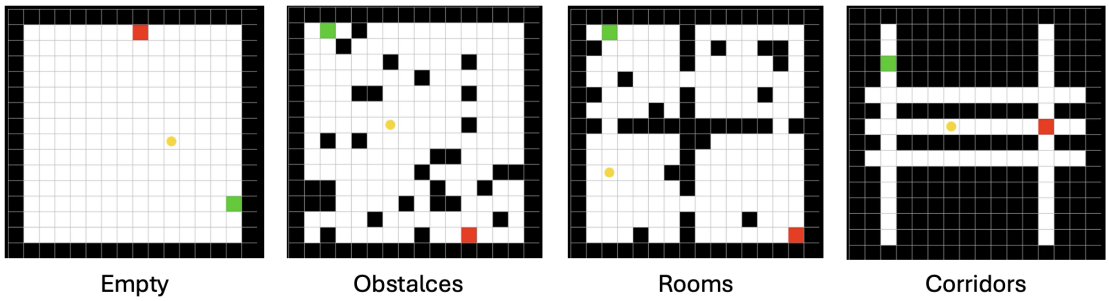

Figure 2: Examples of our generated mazes (16×16). We use four maze environments (left to right): **EMPTY** – open space; **15% Obstacles** – random obstacle placement; **ROOMS** – structured rooms with doorways; **CORRIDORS** – narrow paths requiring precise navigation. Red squares mark start positions, green squares mark goals, and yellow circles show the current agent location.

creates a relaxation of categorical distributions that preserves gradients for backpropagation, facilitating end-to-end training of our actor-critic architecture.

GPS-D: A deterministic variant of our approach that uses argmax operations with a straight-through estimator in the decoder instead of Gumbel-Softmax sampling. This version produces consistent, deterministic action sequences for each proto-sequence embedding.

DQN: Deep Q-Network (Mnih et al., 2013) is a foundational model-free DRL algorithm that learns state-action values. DQN utilizes experience replay and a target network to stabilize its learning.

DAR: Dynamic Action Repetition (Srinivas et al., 2017) extends discrete action spaces by repeating original actions at varying rates. DAR enables the agent to select different levels of temporal control, allowing for some action abstraction.

TempoRL: Temporal Reinforcement Learning (Biedenkapp et al., 2021) introduces a proactive approach, where the agent selects both an action and its duration. TempoRL employs a hierarchical structure with a behavior policy for action selection and a skip policy for duration, enabling more fine-grained temporal abstraction and efficient exploration.

4.3 Experimental Setup

State and action representations. We represent the state using a tensor of shape $(N, M, 3)$, where N and M are the height and width of the maze grid. The channels use a similar encoding to that of MiniGridLibrary (Chevalier-Boisvert et al., 2023): *a) Object type*: identifies all environmental elements including walls, empty spaces, agent position, goal location, and starting position; *b) Object color*: provides distinguishing colors for the start position, goal location, and current agent position; *c) Placeholder channel*: consistently set to 0, maintaining compatibility with the MiniGrid format. Our discrete environment supports the four basic actions – up, down, left, right – represented as a four-entry one-hot vector

Reward function. We define the rewards function as follows:

$$R = r_{goal} - \frac{1}{l_{max}} \times n_{valid} - \frac{3}{l_{max}} \times n_{invalid}$$

We define this reward function identically across all methods, where r_{goal} is 1 if the agent reached the goal (0 otherwise), l_{max} is the maximal start-goal distance (see Table 1) acting as a regularizer, n_{valid} is the number of valid actions taken, and $n_{invalid}$ is the number of invalid actions (e.g., bumping into a wall).

The underlying step-level reward computation is fully Markovian: each step returns $+1.0$ for reaching the goal, $-1.0/l_{max}$ for valid moves, or $-3.0/l_{max}$ for invalid actions (wall collisions), depending only on the current state and action. When GPS executes a sequence of length L, we aggregate these Markovian step

rewards: $R_{total} = \sum(r_t)$ for $t = 1..L$. This reward structure encourages goal achievement while penalizing excessive steps and invalid actions, ensuring fair comparison across all methods.

Evaluation metrics. We use three evaluation metrics:

- **Average success rate (ASR):** the percentage of episodes evaluated where the agent navigates successfully from start to goal position within a predefined number of steps.
- **Path efficiency ratio (PER):** for successfully completed episodes, we calculate the ratio between the episode length and the optimal (minimal) length:

$$PER = \frac{l_{opt}}{l_{episode}}$$

- **Sequence Generation Frequency (*SGF*):** This metric reflects how often the agent generates a new action sequence. It is calculated as the average number of times the Actor is invoked per evaluation episode. Lower values suggest the agent relies on longer-term proto-sequences before needing to generate a new sequence. This metric is relevant to GPS, DAR, and TempoRL baselines.

These metrics are complementary, as they allow us to evaluate the policy's *effectiveness*, *efficiency*, and *decision frequency*, under identical reward optimization across all approaches.

Neural architecture setup. All models use a shared CNN feature extractor followed by method-specific linear layers. DQN outputs Q-values for cardinal directions, DAR expands this for multiple repetition rates, and TempoRL implements a branching architecture for action and skip duration. In GPS, actor and critic networks use separate but identical CNN architectures. The actor produces a 16-dimensional proto-sequence embedding, which the decoder converts into action sequences through a multi-layer network with normalization. Full details are in the appendix.

Hyperparameters & Hardware. Unless otherwise noted for specific ablation studies, experiments were conducted using a common set of key hyperparameters, summarized in Tables 12–17 and 20 in the appendix. We selected the values based on preliminary experiments and common practices. All experiments were conducted on a system running Red Hat 5.14 with x86_64 architecture. We used an NVIDIA RTX 2080 GPU with 8GB of VRAM.

Training Protocol & Model Selection. We used different training setups based on maze size and type. Detailed step counts are in Table 2. Model selection for final testing used the checkpoint from each run yielding the highest average success rate on a held-out set of validation environments during training. Exploration employed an ϵ-greedy strategy, with random sequences being sampled from the same pool of 400 synthetic sequences used to train the PSD (see Appendix L.1 for generation details). This ensures exploratory sequences follow similar structural patterns to those the decoder was trained to generate. While GPS incurs computational overhead per decision, our analysis shows this is often offset by reduced decision frequency and faster convergence (see Appendix F for detailed trade-offs).

4.4 Evaluation Results

4.4.1 Evaluating the Average Success Rate (ASR).

The results of our evaluation are presented in Table 2. GPS consistently outperforms the baselines in most cases, with several key observations:

Ability to learn, converge quickly, and generalize. GPS demonstrates high sample efficiency and rapid convergence. It achieved an ASR of 0.70 ± 0.14 on the 16×16 empty maze with only 100K training steps (compared to DQN's 0.18 ± 0.03), and substantially outperformed baselines with an ASR of 0.82 ± 0.31 for 24×24 empty maze at 500k steps, while the top baseline TempoRL only reached 0.29 ± 0.02.

Table 2: ASR Performance at Diff erent Training Steps. Values represent mean ± standard deviation over three seeds.

Environment	100k Steps					500k Steps					1M Steps					1.5M Steps				
	DQN	GPS	GPS-D	TempoRL	DAR	DQN	GPS	GPS-D	TempoRL	DAR	DQN	GPS	GPS-D	TempoRL	DAR	DQN	GPS	GPS-D	TempoRL	DAR
8x8	0.77 ± 0.08	0.99 ± 0.01	0.93 ± 0.02	0.84 ± 0.07	0.57 ± 0.05	0.86 ± 0.08	1.00 ± 0.0	1.00 ± 0.01	0.95 ± 0.04	0.74 ± 0.03	0.86 ± 0.08	1.00 ± 0.0	1.00 ± 0.0	0.95 ± 0.04	0.75 ± 0.04	-	-	-	-	-
16x16	0.18 ± 0.03	0.70 ± 0.14	0.50 ± 0.07	0.31 ± 0.06	0.30 ± 0.04	0.72 ± 0.10	1.00 ± 0.01	0.95 ± 0.01	0.83 ± 0.06	0.62 ± 0.04	0.75 ± 0.05	1.00 ± 0.0	0.97 ± 0.01	0.86 ± 0.05	0.67 ± 0.02	0.76 ± 0.06	1.00 ± 0.0	0.99 ± 0.01	0.87 ± 0.06	0.67 ± 0.02
16x16_obst_15%	0.10 ± 0.05	0.38 ± 0.05	0.31 ± 0.02	0.07 ± 0.02	0.31 ± 0.02	0.76 ± 0.01	0.90 ± 0.02	0.66 ± 0.07	0.69 ± 0.04	0.66 ± 0.06	0.82 ± 0.02	0.95 ± 0.03	0.80 ± 0.09	0.77 ± 0.03	0.80 ± 0.09	0.85 ± 0.03	0.95 ± 0.03	0.91	0.80 ± 0.01	0.84 ± 0.06
16x16_obst_25%	0.03 ± 0.01	0.08 ± 0.02	0.04 ± 0.02	0.03 ± 0.01	0.03 ± 0.01	0.54 ± 0.10	0.44 ± 0.15	0.27 ± 0.13	0.13 ± 0.06	0.08 ± 0.06	0.70 ± 0.07	0.59 ± 0.12	0.48 ± 0.08	0.62 ± 0.07	0.11 ± 0.07	0.73 ± 0.08	0.64 ± 0.17	0.51 ± 0.14	0.74 ± 0.09	0.15 ± 0.04
16x16_rooms	0.04 ± 0.03	0.13 ± 0.07	0.06 ± 0.04	0.03 ± 0.02	0.02 ± 0.01	0.55 ± 0.02	0.56 ± 0.15	0.39 ± 0.13	0.42 ± 0.10	0.04 ± 0.02	0.66 ± 0.01	0.86 ± 0.06	0.67 ± 0.06	0.60 ± 0.04	0.09 ± 0.02	0.66 ± 0.01	0.89 ± 0.07	0.72 ± 0.06	0.68 ± 0.06	0.14 ± 0.05
16x16_corr	0.48 ± 0.09	0.96 ± 0.03	0.84 ± 0.07	0.60 ± 0.07	0.24 ± 0.14	0.81 ± 0.02	0.99 ± 0.01	0.92 ± 0.01	0.90 ± 0.02	0.46 ± 0.04	0.81 ± 0.02	0.99 ± 0.01	0.92 ± 0.01	0.91 ± 0.03	0.60 ± 0.02	-	-	-	-	-
24x24	0.04 ± 0.01	0.25 ± 0.08	0.17 ± 0.07	0.02 ± 0.01	0.05 ± 0.01	0.11 ± 0.02	0.82 ± 0.31	0.73 ± 0.41	0.29 ± 0.02	0.20 ± 0.04	0.25 ± 0.03	0.90 ± 0.17	0.81 ± 0.28	0.51 ± 0.04	0.26 ± 0.01	0.26 ± 0.03	0.97 ± 0.06	0.87 ± 0.21	0.54 ± 0.06	0.26 ± 0.01
24x24_obst_15%	0.02 ± 0.01	0.20 ± 0.04	0.09 ± 0.01	-	0.04 ± 0.01	0.10 ± 0.03	0.67 ± 0.07	0.36 ± 0.06	-	0.06 ± 0.01	0.12 ± 0.01	0.77 ± 0.06	0.42 ± 0.05	-	0.10 ± 0.01	0.14 ± 0.01	0.82 ± 0.01	0.45 ± 0.03	-	0.12 ± 0.02
24x24_obst_25%	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Note: The ASR for each algorithm at specific training step intervals. A gray background indicates the highest ASR achieved for that environment across all steps and algorithms. A yellow background indicates the highest ASR within that specific step interval (excluding any cell already marked gray). '-' indicates unavailable data.

This supports our hypothesis that modeling action sequences rather than individual actions enables more strategic exploration. GPS primarily learns to *generate sequences that move the agent in the correct general direction toward goals*, allowing progress in unseen environments even without perfectly optimized paths. The deterministic variant, GPS-D, also shows strong performance, supporting the robustness of the proto-sequence concept.

Scalability and superior ability to solve complex environments. The performance gap widens in larger environments. In the empty 24×24 maze, GPS achieves almost perfect performance of 0.97 ± 0.06 at 1.5M steps, whereas DQN and TempoRL only reach 0.26 ± 0.03 and 0.54 ± 0.06 respectively after 1.5M steps. In complex 24×24 environments with 15% obstacles, our approach achieves an ASR=0.82 after 1.5M steps, almost eight times its closest competitor. Even in the most difficult environments (24×24 with 25% obstacles), GPS maintains a significant relative advantage (ASR=0.36 vs. 0.14 for TempoRL).

Performance on medium-sized and structured environments. In structured "rooms" environments, GPS achieves an ASR of 0.89 ± 0.07 at 1.5M steps, significantly outperforming DQN (0.66 ± 0.01) and TempoRL (0.68 ± 0.06), while DAR struggles considerably (0.14 ± 0.05). This demonstrates GPS's effectiveness in navigating complex structured layouts that require coordinated multi-step strategies. In the challenging "16x16_obstacles_15%" setup for 16×16 mazes, GPS achieves strong performance with an ASR of 0.95 ± 0.03, showing moderate improvements over DQN (0.85 ± 0.03), TempoRL (0.8 ± 0.01), and DAR (0.84 ± 0.06). In the most challenging "16x16_obstacles_25%" setup, GPS achieves an ASR of 0.64 ± 0.17, which is comparable with TempoRL (0.74 ± 0.09) and DQN (0.73 ± 0.08), showing that extremely dense obstacle distributions can limit the advantages of sequence-level decision making when precise navigation is required. The performance gap becomes most pronounced in larger 24 ×24 environments, where GPS achieves excellent performance (ASR of 0.97 ± 0.06 for empty 24x24 and 0.82 ± 0.01 for 24x24_obstacles_15%) compared to much lower success rates for all baselines.

The baseline methods demonstrate varying strengths across different environments. DQN achieves competitive performance in several cases, particularly in medium-complexity environments like 16x16 "obstacles_25%" (ASR 0.73) and structured rooms (ASR 0.66). TempoRL shows strong performance in specific configurations, excelling in corridors (ASR 0.90) and achieving solid results in obstacle environments (ASR 0.74-0.80). DAR exhibits more variable performance, performing well in some obstacle configurations (ASR 0.84 in 16x16 "obstacles_15%") but struggling significantly in structured environments like rooms (ASR 0.14). For both TempoRL and DAR, which might require more extensive training to converge optimally, we observed improved performance with larger training datasets (see Table 4 in the appendix), though computational constraints limited further exploration. While DAR also showed improvements under these conditions, it still faced significant challenges in complex structured environments, suggesting that simple action repetition may have inherent limitations for certain maze types. GPS's strong performance stems from operating in the space of action sequences rather than individual actions, enabling more strategic exploration and the discovery of long-horizon rewards that would be difficult to find using single actions or simple repetition methods.

Table 3: Comparative Performance Analysis: Convergence Speed and Efficiency Metrics. Values represent mean ± standard deviation over three seeds.

Environment	ASR Converge>0.9 Step					PER					SGF				
	DQN	GPS	GPS-D	TempoRL	DAR	DQN	GPS	GPS-D	TempoRL	DAR	DQN	GPS	GPS-D	TempoRL	DAR
8x8	>1M	100k	100k	200k	>1M	0.98 ± 0.03	0.89 ± 0.01	**0.99 ± 0.01**	0.80 ± 0.01	0.61 ± 0.02	-	3.36 ± 0.05	**2.87 ± 0.07**	5.28 ± 0.26	3.99 ± 0.20
16x16	>1M	200k	400k	>1.5M	>1.5M	0.91 ± 0.15	0.84 ± 0.01	**0.99 ± 0.01**	0.94 ± 0.01	0.79 ± 0.04	-	8.79 ± 0.66	**7.69 ± 0.86**	12.50 ± 0.97	8.10 ± 0.36
16x16_obstacles_15%	>1.5M	500k	>1.5M	>1.5M	>1.5M	**0.96 ± 0.07**	0.74 ± 0.03	**0.96 ± 0.01**	**0.96 ± 0.01**	0.65 ± 0.03	-	14.25 ± 1.08	**11.04 ± 0.43**	15.56 ± 1.35	11.23 ± 0.79
16x16_obstacles_25%	>1.5M	>1.5M	>1.5M	>1.5M	>1.5M	0.95 ± 0.09	0.67 ± 0.01	0.94 ± 0.02	**0.98 ± 0.02**	N/R	-	17.63 ± 0.75	**12.36 ± 0.91**	20.71 ± 1.54	N/R
16x16_rooms	>1.5M	900k	>1.5M	>1.5M	>1.5M	0.92 ± 0.14	0.68 ± 0.01	0.93 ± 0.01	**0.96 ± 0.02**	N/R	-	15.82 ± 0.81	**13.18 ± 2.8**	17.56 ± 4.35	N/R
16x16_corridors	>1M	100k	200k	500k	>1M	0.95 ± 0.09	0.80 ± 0.01	**0.97 ± 0.01**	0.90 ± 0.01	0.71 ± 0.01	-	8.54 ± 0.50	7.16 ± 0.05	9.40 ± 0.17	**5.71 ± 2.09**
24x24	>1.5M	500k	600k	>1.5M	>1.5M	N/R	0.80 ± 0.07	**0.99 ± 0.01**	0.96 ± 0.01	N/R	-	**12.40 ± 1.80**	12.78 ± 5.98	17.10 ± 0.41	N/R
24x24_obstacles_15%	>1.5M	>1.5M	>1.5M	-	>1.5M	N/R	0.58 ± 0.01	N/R	-	N/R	-	**14.55 ± 0.34**	N/R	-	N/R
24x24_obstacles_25%	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Note: We present four key performance metrics. The first column shows training steps required to achieve a 90% success rate (lower is better), with highlighted values indicating the fastest convergence. Path Efficiency Ratio (PER) measures trajectory optimality (higher is better, max=1.0), with **bold values** showing best performance. Sequence Generation Frequency (SGF) indicates the average number of decision points needed per episode (lower generally indicates better temporal abstraction). '-' indicates N/A and 'N/R' for ASR < 0.5, indicating insufficient success rate for meaningful PER/SGF evaluation.

4.4.2 Evaluating the Path Efficiency Ratio (PER).

The results of our evaluation are presented in Table 3. PER is calculated at the final training checkpoint using the total time steps per environment detailed in Table 1. We report PER for GPS, GPS-D, and two baselines introduced in Section 4.2, allowing for direct comparison across methods. A key GPS characteristic is **Self-Correction Through Sequential Decision Points**. GPS can adjust its course at subsequent decision points without requiring an initially perfect action sequence. This sequence-level closed-loop control enables course corrections while retaining the benefits of temporal abstraction (Empirical validation of this self-correction capability is provided in Appendix E, demonstrating GPS's ability to generate correction sequences with high success rates (62.4-99.6% immediate correction) across diverse maze environments). Leveraging this capability, GPS adopts a strategy of **Trading Path Efficiency for Robust Navigation**, prioritizing directional correctness over strict path optimality. This approach develops more transferable navigation skills—particularly evident in larger or more obstacle-dense mazes—explaining cases where PER is lower despite higher ASR and faster convergence (see Tables 2 and 3).

GPS-D consistently yields higher PER than GPS in all environments. For example, in the 16×16 empty maze, GPS-D's PER is 0.99 versus GPS's 0.84; in the 24×24 empty maze, PER is 0.99 for GPS-D and 0.8 for GPS. GPS's Gumbel-Softmax sampling introduces stochasticity that enables broader exploration but can cause path deviations. GPS-D's deterministic $argmax$ decoder produces more consistent trajectories, trading exploration advantages for improved exploitation. Among the baselines, *DQN* and *TempoRL* often show high PER, frequently achieving near-optimal paths. For instance, in 16×16 "obstacle_15%" maze, *DQN* reached PER 0.96 and *TempoRL* 0.96. However, GPS often surpasses their ASR in complex environments. *DAR* generally shows a lower PER.

findings reveal a conceptual trade-off in our approach: GPS's stochasticity boosts exploration and rapid convergence to high ASR, while GPS-D's determinism excels in path efficiency once a good policy is learned. This sequence-level exploration-exploitation trade-off offers practitioners a choice between prioritizing solution discovery (GPS) or execution efficiency (GPS-D) based on their specific requirements.

4.4.3 Evaluating Sequence Generation Frequency (SGF).

The results are presented in Table 3, where lower values generally indicate superior temporal abstraction due to fewer policy invocations per episode. For methods reporting SGF, our approaches GPS and GPS-D demonstrate competitive performance across environments. In 16×16 empty maze, GPS-D achieves an SGF of 7.69, outperforming *DAR*'s 8.10 and *TempoRL*'s 12.5, while GPS records 8.79. However, in the 16×16 corridors environment, *DAR* (SGF 5.71) outperform GPS-D (7.16) and GPS (8.54). Despite this environment-dependent variation, our methods often operate with limited interventions-such as GPS-D's 12.78 SGF in 24×24 empty mazes versus *TempoRL*'s 17.10-demonstrating effective generation of extended proto-sequences.

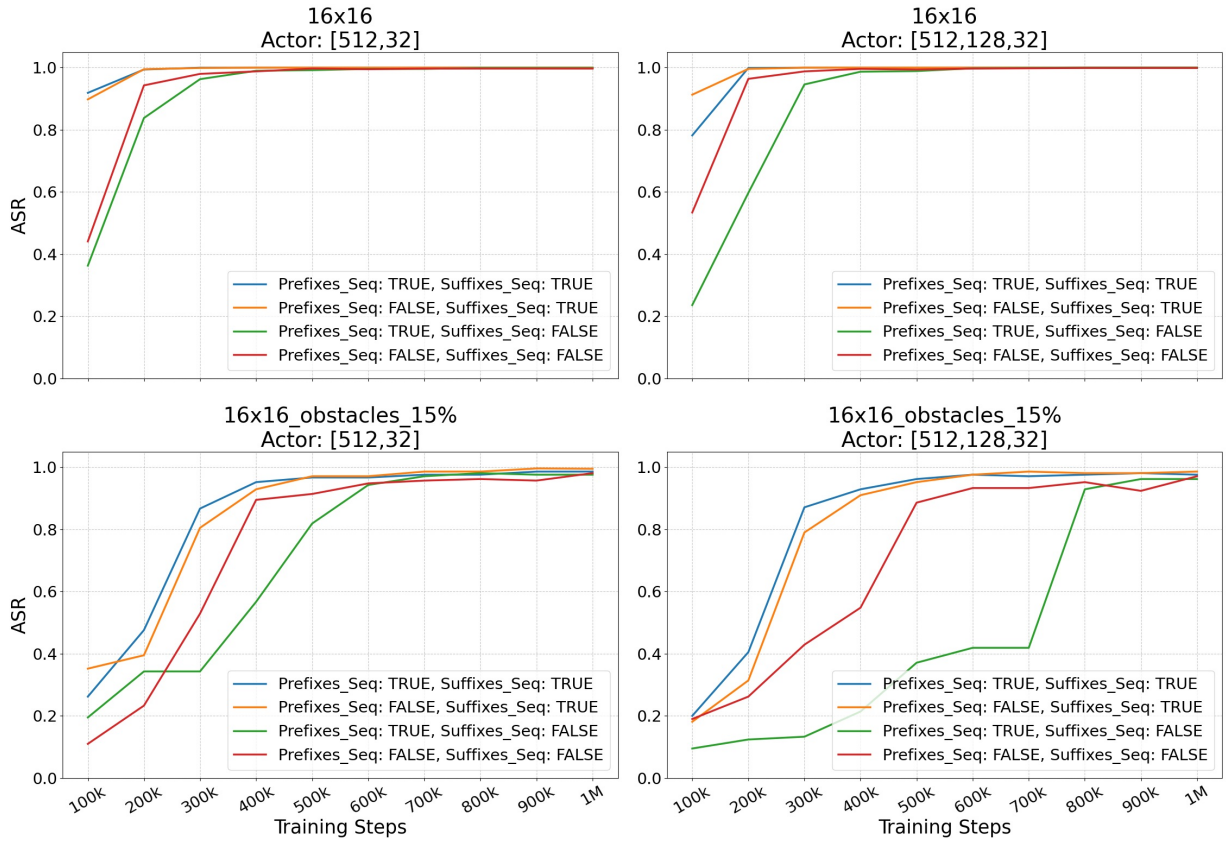

Figure 3: Impact of Subsequence Buffering Strategy on Average Success Rate.

GPS-D's generally low SGF combined with its high PER indicates capability for efficient, strategic trajectory generation through robust behavioral patterns, making it ideal for scenarios requiring predictable execution or constrained resources. GPS offers a compelling trade-off with competitive PER and favorable SGF compared to *TempoRL* (e.g., 8.79 vs. 12.5 in 16×16 empty; 15.82 vs. 17.56 in 16×16 rooms), alongside faster Average Success Rate convergence as discussed in Section 4.4.1. It balances path efficiency, sequence compactness, and learning speed effectively.

In conclusion, SGF analysis confirms our sequence-generation paradigm's effectiveness for temporal abstraction. GPS-D provides efficient, long-term utility with fewer, optimal decisions, while GPS balances competitive SGF, good PER, and rapid ASR. The choice between them depends on application priorities: efficiency vs. predictability or adaptation vs. broader performance.

5 Analysis and Discussion

5.1 Analyzing GPS's Ability to Generate Novel Sequences

While we provided our Decoder with a diverse training set, the latter did not include all possible trajectories. Our reasons were twofold. First, while including all possible action combinations was feasible in our (relatively small) action space, doing the same for larger, more complex action spaces would be infeasible or very costly. secondly, we wanted to evaluate GPS's ability to generalize and produce trajectories that were not in the training set. We consider the ability to generalize important, because the lack of it may limit the usefulness of our approach in large action spaces.

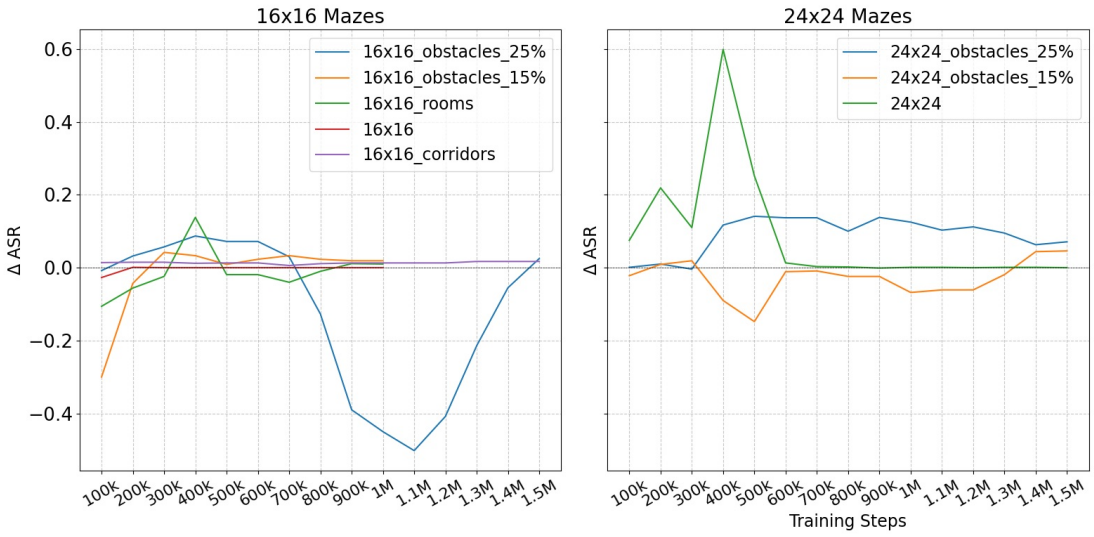

Figure 4: Impact of Actor Network Scaling on Average Success Rate (Δ ASR) in Mazes.

As described in Appendix L.1, the PSD was pre-trained on a set of 400 synthetic sequences generated according to simple, common-sense heuristics for navigation tasks: *a)* each sequence contained at most two distinct action types. *b)* Actions of the same type appeared in contiguous blocks (e.g., "up, up, left" allowed; "up, left, up" disallowed). *c)* No immediately contrasting actions were allowed (e.g., "up, down" prohibited). *d)* Maximum sequence length was capped at $L_{\max}$ (shorter sequences permitted). *e)* Avoidance of loops. After full training, we gathered 15 action sequences by sampling states from the GPS replay buffer and generating the corresponding action sequences through the actor and PSD. Eleven of these did not appear in the PSD's training set and were not fully aligned with at least one of the navigation patterns described above. Using the encoding up→0, down→1, left→2, right→3, the novel sequences were:

$$
\begin{aligned}
[\,&[1,2,1,1],\ [1,1,0],\ [3,1,0,1],\ [0,0,3,1],\\
&[3,3,2],\ [3,0,3],\ [1,1,2,3],\ [2,0,1],\\
&[0,0,3,3,0,3],\ [3,3,1,2],\ [2,1,1,3]]
\end{aligned}
$$

These results show that GPS can create new action sequences not seen during training because it works in a structured embedding space. In this space, sequences with similar structures are grouped together, making it possible to blend known patterns and generate new ones, as shown in Figure 6 in the appendix.

5.2 Sequence Subsets Augmentation

We investigate our subsequence buffering approach (Section 3.4), implemented through *prefixes* (fixed start, varying end point) and *suffixes* (fixed goal state, varying starting point). Figure 3 shows that the baseline without subsequence buffering (red) consistently learns most slowly and often converges sub-optimally, while all subsequence buffering variants substantially improve learning efficiency. Using prefixes and suffixes simultaneously (blue) generally produces the most rapid learning, though the suffix-only configuration (orange) performs nearly as well, suggesting backward sampling provides particularly valuable learning signals. The prefix-only approach (green) typically shows slower convergence than other subsequence methods. These performance patterns remain consistent across different maze structures and actor networks.

5.3 Performance Under Stochastic Dynamics

Up to this point, we conducted our evaluation in a deterministic setting. We now evaluate GPS's robustness in a stochastic setting, which is more challenging to our approach because of its "commitment" to fully implementing its generated trajectories. We used the well established "sticky actions" mechanism Dabney et al. (2020), with 25% probability of repeating the previous action instead of executing the planned one. This introduces temporal correlations and execution uncertainty that challenge sequence-based methods, as errors can potentially cascade throughout multi-step sequences.

Results show that GPS maintains strong performance under these conditions, achieves near-perfect or perfect success rates across all tested environments while preserving competitive convergence times and reasonable path efficiency. The robust performance under stochastic conditions provides evidence that GPS's sequence generation approach may confer resilience beyond deterministic settings. Operating at the sequence level appears to offer some natural buffering against action execution uncertainties, though comprehensive evaluation across diverse stochastic environments would be needed to fully establish this robustness. Detailed experimental details and quantitative results supporting this analysis are presented in Appendix C.

5.4 Performance Under Partial Observability

Beyond stochastic dynamics, we evaluated GPS's robustness under partial observability constraints, where agents only perceive a 7×7 local view window centered on their current position rather than the full maze structure. This setting is particularly challenging for sequence-based methods as it requires generating multi-step action sequences without complete environmental information, testing whether our approach can maintain effective navigation strategies when operating with limited sensory input.

Results reveal environment-dependent performance patterns under partial observability. In empty and corridor environments, GPS demonstrates strong performance, maintaining clear advantages over DQN in empty mazes throughout training (0.54 vs 0.15 at 100K steps, 0.71 vs 0.65 at convergence) and achieving competitive results in corridors (0.78 vs 0.78 ASR at 1M steps), as action sequences help maintain consistent exploration patterns despite limited visibility. However, the moderate path efficiency ratios (PER 0.50) in these environments indicate suboptimal navigation compared to DQN's perfect efficiency, suggesting that while GPS reaches goals successfully, the generated sequences include unnecessary detours when operating with incomplete information.

In contrast, GPS faces increased challenges in obstacle-dense and rooms environments. While maintaining early advantages in obstacle mazes (0.30 vs 0.05 at 100K steps), DQN eventually surpasses GPS (0.78 vs 0.34 at 1.5M steps), and rooms environments show reduced performance (0.48 vs 0.65 ASR). We hypothesize these difficulties stem from GPS generating multi-step sequences without seeing upcoming barriers or doorways, while DQN can immediately incorporate newly revealed environmental features. The increased Sequence Generation Frequency in complex environments (16.63 in obstacles, 11.71 in empty mazes) reflects less effective navigation with incomplete spatial knowledge. No configuration reaches 0.9 ASR within 1.5M steps, indicating substantial sample complexity increases under partial observability compared to full observability settings where convergence typically occurs within 500K-1M steps. Complete experimental analysis is provided in Appendix D.

5.5 Impact of Actor Network Scaling

We examined actor network size impact (small: two-layer (512, 32); large: three-layer (512, 128, 32)) on maze navigation performance, measured by ΔASR (Large - Small) (see Figure 4). In simpler 16x16 mazes (empty or corridor), both architectures performed similarly. With 15% obstacles, the smaller network initially outperformed (ΔASR ≈ -0.3 at 100K steps) before convergence at 300K steps. In denser 25% obstacles, the smaller network significantly outperformed from 800K steps, peaking at ΔASR ≈ -0.5 at 1.1M steps. In larger 24x24 mazes with 15% obstacles, the small network generally led, despite the large network's brief advantage (500K steps). However, in the most complex 25% obstacles maze, the large network consistently outperformed, maintaining ΔASR between 0.1-0.15. The 24x24 empty maze showed fluctuating performance with occasional spikes for the larger network around 200K and 500K steps.

These results suggest a trade-off: smaller networks suffice or excel in smaller or moderately complex environments (possibly due to better regularization or more stable sequence generation learning), while larger networks demonstrate clear benefits in more complex environments.

5.6 Computational Complexity and Runtime

GPS introduces additional runtime overhead compared to baselines, as its Actor-Decoder pipeline makes each decision 1.6–3.3x slower in wall-clock time. However, because GPS operates at a sequence level and makes decisions less frequently, the effective runtime per episode is comparable to baselines in moderately complex settings (e.g., 16x16 mazes). In larger, more difficult environments GPS shows higher normalized costs, but this is partly because it continues to solve tasks where baselines fail. Despite the per-decision slowdown, GPS converges to high success rates much faster overall (90% in 3.3 hours vs. DQN's 5.83 hours), demonstrating that its superior sample efficiency compensates for runtime overhead in practice. This makes GPS more useful in challenging tasks where training effectiveness outweighs raw inference speed. A comprehensive analysis is presented in Section F in the Appendix.

6 Conclusions, Limitations, and Future Work

GPS is a novel actor-critic method that generates variable-length action sequences in a single step. GPS maps state observations to proto-sequences, which are decoded into discrete action sequences. This approach enhances credit assignment and exploration in long-horizon tasks by moving beyond sequential single-action selection. Our evaluation shows GPS consistently surpasses leading action repetition and temporal methods in complex maze environments, achieving higher success rates and faster convergence.

Although our approach shows benefits, particularly in complex environments, several limitations should be acknowledged. First, a new PSD needs to be trained for each unique action space, which adds to the complexity of our approach. GPS has not been evaluated on large action spaces, so adaptations to the decoder component may be needed. Furthermore, our analysis suggests that a partial information setting (e.g., limited visibility) sometimes reduces the effectiveness or our proposed approach. Finally, we have not yet adapted GPS to continuous action spaces.

Another important aspect of our approach is its suitability for domains with varying requirements. While GPS significantly outperforms the baselines in large, long horizon setups and is able to generate solutions rapidly, the resulting solutions are not always strictly optimal. This trade-off between solution quality and computational efficiency is a well-documented phenomenon in deep reinforcement learning, where methods that prioritize rapid inference may forgo perfect optimality in favor of operational practicality and responsiveness. Similar observations have been reported in recent DRL research Sohaib et al. (2025); Wu et al. (2023), highlighting how faster-converging methods frequently produce solutions that, while feasible and effective in real-world applications, may not match the optimality of simpler, slower approaches such as DQN.

Future work will focus on extending to more complex domains, such as those with larger action spaces or continuous control. Furthermore, we will aim to adapt GPS to function more effectively in partial information and stochastic settings. To this end, we are considering the creation of a re-planning component, and mechanism that will adapt the maximal length of the generated sequences based on the available information. Finally, we will explore modifications to our approach that will enable us to define desired trade-offs between path optimality and success. By doing so, our aim is to automatically adapt GPS's strategy to match the requirements of various domains.

References

Anurag Ajay, Seungwook Han, Yilun Du, Shuang Li, Abhi Gupta, Tommi Jaakkola, Josh Tenenbaum, Leslie Kaelbling, Akash Srivastava, and Pulkit Agrawal. Compositional foundation models for hierarchical planning. *Advances in Neural Information Processing Systems*, 36:22304–22325, 2023.

Jose A Arjona-Medina, Michael Gillhofer, Michael Widrich, Thomas Unterthiner, Johannes Brandstetter, and Sepp Hochreiter. Rudder: Return decomposition for delayed rewards. *Advances in Neural Information Processing Systems*, 32, 2019.

André Biedenkapp, Raghu Rajan, Frank Hutter, and Marius Lindauer. Temporl: Learning when to act. In *International Conference on Machine Learning*, pp. 914–924. PMLR, 2021.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.

Maxime Chevalier-Boisvert, Bolun Dai, Mark Towers, Rodrigo de Lazcano, Lucas Willems, Salem Lahlou, Suman Pal, Pablo Samuel Castro, and Jordan Terry. Minigrid & miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks. *CoRR*, abs/2306.13831, 2023.

Josiah D Coad, James Ault, Jeff Hykin, and Guni Sharon. A framework for predictable actor-critic control. In *Deep Reinforcement Learning Workshop NeurIPS 2022*, 2022.

Will Dabney, Georg Ostrovski, and André Barreto. Temporally-extended {\epsilon}-greedy exploration. *arXiv preprint arXiv:2006.01782*, 2020.

Zihang Dai, Qizhe Xie, and Eduard Hovy. From credit assignment to entropy regularization: Two new algorithms for neural sequence prediction. *arXiv preprint arXiv:1804.10974*, 2018.

Alan Dao and Dinh Bach Vu. Alphamaze: Enhancing large language models' spatial intelligence via grpo. *arXiv preprint arXiv:2502.14669*, 2025.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Gabriel Dulac-Arnold, Richard Evans, Hado van Hasselt, Peter Sunehag, Timothy Lillicrap, Jonathan Hunt, Timothy Mann, Theophane Weber, Thomas Degris, and Ben Coppin. Deep reinforcement learning in large discrete action spaces. *arXiv preprint arXiv:1512.07679*, 2015.

Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O Stanley, and Jeff Clune. First return, then explore. *Nature*, 590(7847):580–586, 2021.

Liad Giladi and Gilad Katz. Feedback decision transformer: Offline reinforcement learning with feedback. In *IEEE International Conference on Data Mining (ICDM)*. IEEE, 2023.

Chia-Chun Hung, Timothy Lillicrap, Josh Abramson, Yan Wu, Mehdi Mirza, Federico Carnevale, Arun Ahuja, and Greg Wayne. Optimizing agent behavior over long time scales by transporting value. *Nature communications*, 10(1):5223, 2019.

Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. *Advances in neural information processing systems*, 34:1273–1286, 2021.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. *Advances in neural information processing systems*, 29, 2016.

Nishanth Kumar, Tom Silver, Willie McClinton, Linfeng Zhao, Stephen Proulx, Tomás Lozano-Pérez, Leslie Pack Kaelbling, and Jennifer Barry. Practice makes perfect: Planning to learn skill parameter policies. *arXiv preprint arXiv:2402.15025*, 2024.

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40, 2016.

Jinghan Li, Zhicheng Sun, and Yadong Mu. Closed-loop long-horizon robotic planning via equilibrium sequence modeling. *arXiv preprint arXiv:2410.01440*, 2024.

Wenhao Li. Efficient planning with latent diffusion. *arXiv preprint arXiv:2310.00311*, 2023.

Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *4th International Conference on Learning Representations (ICLR)*, 2016.

Jinxin Liu, Donglin Wang, Qiangxing Tian, and Zhengyu Chen. Learn goal-conditioned policy with intrinsic motivation for deep reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pp. 7558–7566, 2022.

Fan-Ming Luo, Tian Xu, Hang Lai, Xiong-Hui Chen, Weinan Zhang, and Yang Yu. A survey on model-based reinforcement learning. *Science China Information Sciences*, 67(2):121101, 2024.

Marlos C Machado, Marc G Bellemare, Erik Talvitie, Joel Veness, Matthew Hausknecht, and Michael Bowling. Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *Journal of Artificial Intelligence Research*, 61:523–562, 2018.

Thomas Mesnard, Théophane Weber, Fabio Viola, Shantanu Thakoor, Alaa Saade, Anna Harutyunyan, Will Dabney, Tom Stepleton, Nicolas Heess, Arthur Guez, et al. Counterfactual credit assignment in model-free reinforcement learning. *arXiv preprint arXiv:2011.09464*, 2020.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, and Georg Ostrovski. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

Jurgis Pašukonis, Timothy P Lillicrap, and Danijar Hafner. Evaluating long-term memory in 3d mazes. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=yHLvIlE9RGN`.

Devdhar Patel and Hava Siegelmann. Overcoming slow decision frequencies in continuous control: Model-based sequence reinforcement learning for model-free control. *arXiv preprint arXiv:2410.08979*, 2024.

Roberta Raileanu and Tim Rocktäschel. Ride: Rewarding impact-driven exploration for procedurally-generated environments. *arXiv preprint arXiv:2002.12292*, 2020.

Erick Rosete-Beas, Oier Mees, Gabriel Kalweit, Joschka Boedecker, and Wolfram Burgard. Latent plans for task-agnostic offline reinforcement learning. In *Conference on Robot Learning*, pp. 1838–1849. PMLR, 2023.

Tankred Saanum, Noémi Éltető, Peter Dayan, Marcel Binz, and Eric Schulz. Reinforcement learning with simple sequence priors. *Advances in Neural Information Processing Systems*, 36:61985–62005, 2023.

Younggyo Seo and Pieter Abbeel. Coarse-to-fine q-network with action sequence for data-efficient robot learning. *arXiv preprint arXiv:2411.12155*, 2024a.

Younggyo Seo and Pieter Abbeel. Coarse-to-fine q-network with action sequence for data-efficient robot learning. *arXiv preprint arXiv:2411.12155*, 2024b.

Sahil Sharma, Aravind Srinivas, and Balaraman Ravindran. Learning to repeat: Fine grained action repetition for deep reinforcement learning. *arXiv preprint arXiv:1702.06054*, 2017.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

Rana Muhammad Sohaib, Syed Tariq Shah, Muhammad Ali Jamshed, Oluwakayode Onireti, and Poonam Yadav. Optimizing urllc in open ran: A deep reinforcement learning-based trade-off analysis. *IEEE Communications Standards Magazine*, 2025.

Aravind Srinivas, Sahil Sharma, and Balaraman Ravindran. Dynamic action repetition for deep reinforcement learning. In *Proc. AAAI*, 2017.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

Giulia Vezzani, Dhruva Tirumala, Markus Wulfmeier, Dushyant Rao, Abbas Abdolmaleki, Ben Moran, Tuomas Haarnoja, Jan Humplik, Roland Hafner, Michael Neunert, et al. Skills: Adaptive skill sequencing for efficient temporally-extended exploration. *arXiv preprint arXiv:2211.13743*, 2022.

Shuo Wang, Zhihao Wu, Xiaobo Hu, Youfang Lin, and Kai Lv. Skill-based hierarchical reinforcement learning for target visual navigation. *IEEE Transactions on Multimedia*, 25:8920–8932, 2023.

Pengfei Wu, Chen Chen, Dexiang Lai, and Jian Zhong. A safe drl method for fast solution of real-time optimal power flow. *arXiv preprint arXiv:2308.03420*, 2023.

Mengda Xu, Manuela Veloso, and Shuran Song. Aspire: Adaptive skill priors for reinforcement learning. *Advances in Neural Information Processing Systems*, 35:38600–38613, 2022.

Dongkun Zhang, Jiaming Liang, Ke Guo, Sha Lu, Qi Wang, Rong Xiong, Zhenwei Miao, and Yue Wang. Carplanner: Consistent auto-regressive trajectory planning for large-scale reinforcement learning in autonomous driving. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 17239–17248, 2025.

Haichao Zhang, Wei Xu, and Haonan Yu. Generative planning for temporally coordinated exploration in reinforcement learning. In *International Conference on Learning Representations*, 2022a.

Haichao Zhang, Wei Xu, and Haonan Yu. Generative planning for temporally coordinated exploration in reinforcement learning. *arXiv preprint arXiv:2201.09765*, 2022b.