SEAR: Sample Efficient Action Chunking Reinforcement Learning

C F Maximilian Nagy[1,2] **Onur Celik**[1] **Emiliyan Gospodinov**[1] **Florian Seligmann**[1]

Weiran Liao[1] **Aryan Kaushik**[1]

Gerhard Neumann[1,2]

[1]Autonomous Learning Robots, Karlsruhe Institute of Technology
[2]FZI Forschungszentrum Informatik, Karlsruhe
nagy@fzi.de

Abstract

Action chunking improves exploration and accelerates value propagation in long-horizon reinforcement learning, but naively applying off-policy methods to the temporally extended action space at reduced decision frequency offsets these gains, leading to poor sample efficiency. Existing online action chunking methods address these issues through computationally expensive critic-only approaches or by relying on offline data. We introduce SEAR, a sample-efficient off-policy algorithm that enables online reinforcement learning with action chunks by addressing both challenges. To handle high-dimensional action sequences, SEAR employs a causal transformer critic trained with multi-horizon targets that provide a training signal for every prefix of an action chunk, effectively increasing the useful gradients per sample. To restore decision frequency, SEAR operates with a receding horizon and random replanning, ensuring uniform state coverage while combining the fast value propagation of large chunks with the reactivity of small ones. SEAR outperforms state-of-the-art online reinforcement learning methods including SimbaV2 on challenging Metaworld manipulation tasks. Beyond online RL, applying SEAR to the offline-to-online method QC improves its performance on the demanding OGBench cube-triple tasks.

1 Introduction

Action chunking has seen remarkable success in imitation learning (IL) [6, 36, 29, 4, 28, 19, 37, 25, 14]. There, a policy is trained to predict a sequence of future actions instead of a single action, which are then executed in an open-loop manner, enabling the policy to capture coherent multi-step behaviors.

Building on these results, a growing line of work explores action chunking reinforcement learning (RL) [16, 18, 17, 33, 26, 30]. For reinforcement learning, action chunking enables faster and more stable reward propagation through unbiased N step returns computed over executed chunks, which is particularly beneficial in long horizon and sparse reward settings [18, 17]. By operating over temporally extended action sequences, action chunking facilitates the discovery of meaningful skills and improves exploration by sampling coherent action trajectories rather than relying on single time step random walk noise. This structured exploration leads to more consistent state space coverage and can substantially reduce exploration burden. Additionally, action chunking provides a natural

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

interface for incorporating offline data when available, allowing learned temporal patterns to be reused effectively during online learning [18].

However, action chunking significantly increases the action space of the policy, requiring the critic to evaluate chunks rather than individual actions. In this enlarged action space, critic approximation errors can lead to poor actor updates [30], requiring careful algorithm design. In particular, naively applying action chunking to off-the-shelf single-action RL algorithms significantly reduces performance, as shown in Figure 4a.

Moreover, executing N actions in an open-loop manner reduces the policy's decision frequency, as the policy is only queried every N steps. This has two consequences: Firstly, it requires the predicted action sequence to be highly accurate, as errors cannot be corrected during execution [17]. Secondly, this reduces the policy's "state-action coverage", especially if the initial state distribution is narrow (Figure 2).

A common way to restore policy reactivity is receding horizon control, where the policy predicts an action chunk of length N but executes only a prefix of k actions before replanning. While this strategy is widely used in imitation learning to balance long horizon planning with closed-loop control, it has so far not been explored in the context of reinforcement learning. However, applying receding horizon control in reinforcement learning is nontrivial, as executing shorter prefixes at evaluation time introduces a distribution shift from training on fixed length action chunks.

We introduce *Sample Efficient Action Chunking Reinforcement Learning (SEAR)*, an off-policy RL algorithm that learns action chunking policies. SEAR uses a *multi-horizon* critic target that captures the causal relation between a chunk's actions while providing per-action training targets. Combined with a causal transformer critic, this enables stable learning even for large chunk sizes and even when no prior data is available. Furthermore, SEAR improves the state-action coverage of the actor using *random replanning*, where only a random prefix of each chunk is executed during data collection. Together, random replanning and multi-horizon targets enable SEAR to effectively use receding horizon control to decouple the long action chunks useful during training with shorter evaluation action chunk sizes. In the extreme case, SEAR can also use action chunking only during training, leading to better single-step success rates than state-of-the-art non-chunked methods.

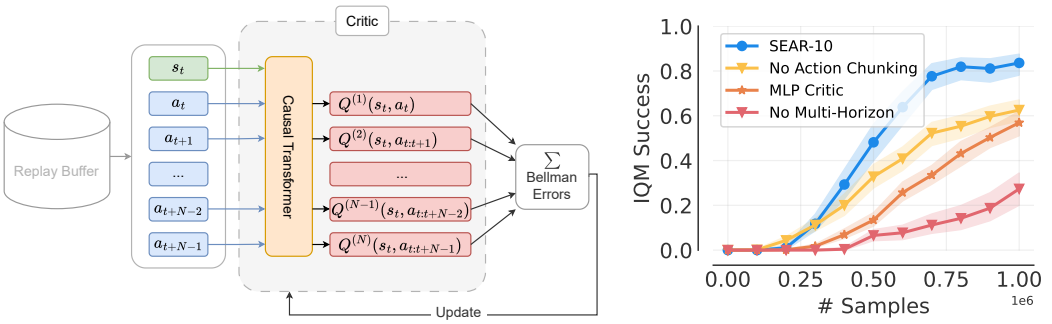
We evaluate SEAR on the challenging Metaworld [34] online RL benchmark and the OGBench [27] offline-to-online RL benchmark. On Metaworld, SEAR achieves a substantial improvement of over 20% IQM success rate [2] compared to previous methods among the 20 hardest environments, as shown in Figure 4. Through extensive ablation studies, we highlight the impact of each design decision of our approach, such as using a causal transformer critic with and without multi-horizon targets, and random replanning (Figure 1, Figure 2). We further evaluate the impact of different chunk sizes (Figure 3). In particular, we demonstrate that receding horizon control, which is used for the first time for RL in our work, is only effective if the main components of our algorithms are used, i.e., multi-horizon targets, causal transformer critics, and random replanning during training.

We summarize our contributions as follows:

- We introduce SEAR, an off-policy RL method for action chunking grounded in the maximum-entropy RL framework.
- We incorporate a causal transformer critic and multi-horizon N-step returns, leading to improved sample efficiency.
- We increase the state-action coverage by executing only a random prefix of each action chunk before replanning, complementing a receding horizon execution mode for fast reactivity.
- Through extensive experiments, we demonstrate that SEAR consistently improves the performance for online and offline-to-online RL, with an over 20% higher IQM success rate on the 20 hardest Metaworld environments. In ablations, we show the importance and performance impact of all design decisions of our action chunking approach.

2 Related Work

In robotic learning, action chunking agents generate multiple consecutive actions at once. This was first introduced in the context of imitation learning by Zhao et al. [36] and was widely adopted by other imitation learning approaches [6, 36, 29, 4, 28, 19, 37, 25, 14].



(a) Critic Update

(b) Critic Ablations (Metaworld-Hard)

Figure 1: **Overview of SEAR's (a) critic design and (b) how it affects performance.** The critic function expects the state s_t and an action chunk $a_{t:t+N}$ of chunk size N as argument. During the critic update **(a)**, these training points are sampled from the replay buffer and the causal transformer critic predicts Q-values $Q^{(1)}(s_t, a_t), Q^{(2)}(s_t, a_{t:t+1}), \ldots, Q^{(N)}(s_t, a_{t+N-1})$ for all action chunk prefixes , thereby generating *multi-horizon* predictions that increase the amount of training data for the subsequent critic update. Only fitting the critic on the whole-chunk target $Q^{(N)}$, as is common in offline RL [18, 35]. does not work well in online RL **(b)**. Furthermore, bootstrapping an action chunking critic with a single-step policy, comparable to T-SAC ("No Action Chunking") lowers success, as does replacing the transformer backbone with an MLP.

N-step Returns in Reinforcement Learning In reinforcement learning, N-step returns have a long history as a compromise between unbiased, high-variance Monte-Carlo returns and high-bias, low-variance TD returns [31]. A fundamental problem of applying N-step returns to off-policy algorithms is that the action-, and therefore also the reward-distribution, of the replay buffer differs from that of the current policy. The standard way to mitigate this issue is importance sampling, where care must be taken to reduce variance [23].

A principled way to avoid this bias is to use an action-chunk-conditioned critic where the N-step return is computed over the given chunk. This was first proposed in the TOP-ERL [16] algorithm for the episodic reinforcement learning setting and has since been adapted to traditional single-action policies [32] and action chunking policies [18, 17, 33, 26, 30].

Multi-Target Updates for Action Chunking Critics A fundamental aspect of learning Q-functions with action chunking is the increased dimensionality of the resulting action space, rendering Q-function approximation more difficult. Most action chunking methods use critics that map from this extended action space to a single Q-target [18, 17, 33, 26]. For these methods, increasing the chunk size N increases the action space dimension without introducing new training targets. Experimental evidence suggests that these critic fitting approaches are unsuitable for online RL (cf Section 5.3, [30]). However, in the offline and offline-to-online RL setting, where demonstration data is available as a prior for learning, this approach can be used effectively [18, 17, 33, 26].

When the critic instead learns a Q-target for each action of a chunk, the ratio between action space dimension and number of training targets becomes independent of the chunk size N. This kind of multi-horizon critic was first proposed in the episodic RL setting by TOP-ERL [16]. TOP-ERL uses a trajectory generator to generate actions for the whole episode at once. Segments of these trajectories are then used for off-policy updates using the reparametrization trick [10]. For each prefix of these segments, an N-step Q-target is calculated and used for critic updates. In contrast to the action chunking setting, the segment length in TOP-ERL can be varied without affecting the action space dimension. T-SAC [32] adapts TOP-ERL's critic for standard policies without action chunking.

CQN-AS [30] is the only previous action-chunking method that uses a multi-target critic. Its critic is trained to predict the returns of each individual action instead of the N-step return of the combined actions. SEAR is the first method that fully exploits the temporal structure of action chunks by adapting TOP-ERL's critic structure into the action chunking setting. This allows SEAR to be used for online RL, in contrast to methods that treat each chunk as a single data point. These methods are

typically confined to the offline-to-online RL case and do not work well for online RL. In contrast to CQN-AS, SEAR explicitly models the causal dependencies between actions.

Exploration in RL Balancing exploration and exploitation is a fundamental challenge in continuous reinforcement learning. Some algorithms employ state-independent, fixed exploration noise on top of an otherwise deterministic policy [20, 7, 33, 30]. A common way to introduce state-dependent exploration are Maximum-Entropy approaches, that learn when and how to explore [10, 3, 5]. Another approach to state-dependent exploration is to use an expressive imitation learning policy to guide exploration [18]. SEAR is the first online RL algorithm that learns an action chunking policy in the maximum entropy setting.

3 Preliminaries

Notation. We aim to train a reinforcement learning policy $\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^+$ in the continuous state-action space $\mathcal{S} \times \mathcal{A}$ using the Markov decision process (MDP) formalization. The MDP is defined by the tuple $(\mathcal{S}, \mathcal{A}, P, r, \gamma, \rho_0)$, where $\gamma \in [0, 1)$ is the discount factor, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}^+$ is the transition probability function returning the likelihood for transitioning to state $s_{t+1} \in \mathcal{S}$ when executing action $a_t \in \mathcal{A}$ in state $s_t \in \mathcal{S}$ and $r : \mathcal{S} \times \mathcal{A} \to [r_{\min}, r_{\max}]$ is the bounded reward function. The initial state distribution $\rho_0 : \mathcal{S} \to \mathbb{R}^+$ denotes the probability of starting at state $s_0 \in \mathcal{S}$. Similar to Haarnoja et al. [10], we denote ρ^π as the state or state-action distribution under the current policy π and denote the time step of a (random) variable with a subscript. This subscript applies to the reward in the sense that $r_t \triangleq r(s_t, a_t)$, which allows for a more compact notation.

We extend the single-step setup to a multi-step action chunking setup, where we overload the subscript notation when referring to a chunksize of N by writing $a_{t:t+N}$, which denotes the sequence of N actions $(a_t, a_{t+1}, ..., a_{t+N-1})$. Additionally, we refer to ρ as the state-action distribution induced by the action chunking policy $\pi(a_{t:t+N}|s_t)$ and which provides the density of a sequence of state-action chunks. Naturally, the dimensionality of the chunk-extended action space grows linearly with chunk size N following $\dim(a_{t:t+N}) = N \dim(a_t)$.

Maximum Entropy Reinforcement Learning. We consider the maximum entropy reinforcement learning (MaxEnt-RL) setup that augments the reward function with the entropy of the current policy, leading to the objective

$$J(\pi) = \sum_{t=0}^{\infty} \gamma^t \mathbb{E}_{\rho^\pi} \left[r_t + \alpha \mathcal{H}(\pi(a_t|s_t)) \right]. \tag{1}$$

Here, $\alpha \in \mathbb{R}^+$ is a scaling factor that trade-offs maximizing the entropy $\mathcal{H}(\pi(a|s)) = -\int \pi(a|s) \log \pi(a|s)$ against maximizing the task return and thereby plays a crucial role in exploration control. This objective also leads to an entropy augmented definition of the Q-function under the current policy π

$$Q^\pi(s_t, a_t) = r_t + \sum_{i=1}^{\infty} \gamma^i \mathbb{E}_{\rho^\pi} \left[r_{t+i} + \alpha \mathcal{H}(\pi(a_{t+i}|s_{t+i})) \right], \tag{2}$$

with $Q^\pi : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$. The MaxEnt-RL objective [39, 38, 9] has several benefits over the standard RL objective [31], among which improved exploration is the key reason we base our method on it.

4 Sample Efficient Action Chunking RL

Naively applying action chunking to a policy does not yield satisfying results (see Figure 4 and Seo & Abbeel [30]). This section introduces SEAR's elementary algorithmic and design choices that are required to obtain action-chunking policies that are sample efficient and achieve high success rates. We start by formalizing the MaxEnt RL objective for action chunking policies, which we use to define the action-chunk-conditioned Q-function. Based on this definition, we introduce multi-horizon Q-function targets that effectively increase the number of target points for training the respective Q-function. This Q-function allows for straightforward policy updates, which we present in the subsequent section. Finally, we propose random replanning that plays a significant role in state coverage and receding horizon replanning. Figure 1 summarizes the update procedure for the critic.

4.1 Action Chunking Maximum Entropy RL

We aim to optimize an action chunking policy $\pi(a_{t:t+N} \mid s_t)$ that predicts an action sequence $a_{t:t+N}$ of length N given the current state s_t. For this, we start by formalizing the infinite horizon MaxEnt-RL objective with action chunk size N by introducing the objective

$$J(\pi) = \mathbb{E}_{s_0 \sim \rho_0} V^\pi(s_0), \tag{3}$$

where $V^\pi(s_t)$ is defined as

$$V^\pi(s_t) := \sum_{k=0}^{\infty} \mathbb{E}_{\rho^\pi} \gamma^k \big[r_{t+k} - \alpha \log \pi(a_{t+k} \mid s_{t+N\lfloor k/N \rfloor}) \big]. \tag{4}$$

For a given state s_t, the whole action chunk $a_{t:t+N}$ is sampled from the policy $\pi(a_{t:t+N}|s_t)$. Thus, starting from time step t, action a_{t+k} only depends on the last step divisible by N, ie. $s_{t+N\lfloor k/N \rfloor}$. The sampled action chunk $a_{t:t+N}$ is then executed in an open-loop fashion in the environment. All intermediate states and values are saved in the replay buffer and used for the critic and actor updates (see Figure 1). The objective in Equation 3 generalizes the known MaxEnt-RL objective in Equation 1 to the action chunking case. The entropy bonus is calculated and discounted for every action separately. Therefore, the optimal policy is independent of the chunk size, which would not be the case if the entropy bonus was calculated using the joint likelihood of a chunk's actions. The generalization can be easily verified by setting the chunk size to $N = 1$, which recovers Objective 1.

Action Chunk Extended Q-function. Objective 3 naturally leads to an action chunk extended Q-function

$$Q^\pi(s_t, a_{t:t+N}) = r_t + \mathbb{E}_p \left[\sum_{n=1}^{N} \gamma^n r_{t+n} + \gamma^N V^\pi(s_{t+N}) \right] \tag{5}$$

As opposed to the common definition of the Q-function, here, the action chunk $a_{t:t+N}$ is passed as an argument to the Q-function instead of a single action a_t [18, 32]. These actions are fixed and executed in the environment in an open-loop manner, leading to the state sequence $s_{t+1:t+1+N}$ and the corresponding reward values $r_{t+1:t+1+N}$. We indicate this open-loop execution of the action chunk using the first expectation under the state transition probability p. Please note that this distribution is opposed to the expectation over ρ^π in $V^\pi(s_t)$ (Equation 4) that explicitly follows from executing the stochastic policy π rather than a fixed action sequence.

Intuitively, the extended Q-function measures the expected return when executing the action chunk $a_{t:t+N}$ in state s_t and following the action chunking policy π thereafter. This aligns with the general definition of the Q-function, but extends it to the action chunking case, which can be easily verified by setting the action chunk length $N = 1$ and thereby recovering the regular Q-function of MaxEnt-RL (cf. Equation 2).

4.2 Multi-Horizon Q-Function Targets

A significant advantage of the Q-function definition in Equation 5 is that the temporal structure of the action sequence can be leveraged for off-policy updates without requiring any importance weights [23] to account for the bias induced by using off-policy data in the Q-value targets.

More concretely, we define the Q-value target $G^{(N)}$ as

$$G^{(N)}(s_t, a_{t:t+N}) = \underbrace{\sum_{i=0}^{N-1} \gamma^i \, r_{t+i}}_{\text{N-step return}} + \gamma^N \hat{Q}^{(N)}_{\bar{\phi}}(s_{\tilde{t}}, a_{\tilde{t}:\tilde{t}+N}), \tag{6}$$

where $\tilde{t} = t + N$ and the target value are computed using target parameters $\bar{\phi}$ and sampled next-state action chunk $a_{\tilde{t}:\tilde{t}+N} \sim \pi_\theta\big(\cdot \mid s_{\tilde{t}}\big)$ as

$$\hat{Q}^{(N)}_{\bar{\phi}}(s_{\tilde{t}}, a_{\tilde{t}:\tilde{t}+N}) = \min_{j \in \{1,2\}} \left[Q^{(N)}_{\bar{\phi}_j}(s_{\tilde{t}}, a_{\tilde{t}:\tilde{t}+N}) - \alpha \sum_{i=0}^{N-1} \gamma^i \log \pi_\theta\big(a_{\tilde{t}+i} \mid s_{\tilde{t}}\big) \right]. \tag{7}$$

However, simply using $G^{(N)}$ to train the Q-function does not exploit the available data effectively because information is lost while adding up the chunk's N rewards in Equation 5. Therefore, we

train the critic on all target values $G^{(1)}, G^{(2)}, ..., G^{(N)}$ (calculated from the chunk's prefixes) in parallel when updating the network parameters ϕ. As seen in Figure 1, each target value is calculated using Equation 6, but with varying horizons N, such that *multi-horizon targets* are calculated. These multi-horizon targets were also recently proposed by Li et al. [16], but in the episode-based RL framework with finite time horizon tasks. Causal models such as transformers suit this target well, as the k-step target $G^{(k)}$ only depends on the first k actions of a chunk. We therefore follow Li et al. [16] and use a causal transformer critic to effectively handle multi-horizon targets. The final objective for updating the parameters ϕ is then given by

$$J(\phi) = \frac{1}{2N} \sum_{k=0}^{N-1} \left(Q_\phi^{(k)}(s_t, a_{t:t+k}) - G^{(k)}(s_t, a_{t:t+k}) \right)^2, \tag{8}$$

where the data is sampled from the replay buffer $\mathcal{D}$. Figure 1 visualizes the critic update.

4.3 Optimizing the Action Chunking Policy

With the Q-function Q_ϕ from Section 4.2 we can easily update the chunking policy $\pi_\theta(a_{t:t+N}|s_t)$ following recent MaxEnt-RL approaches [9, 10] by using the reverse KL divergence

$$J(\pi) = \mathbb{E}_{\mathcal{D}} \Big[\mathrm{KL} \left(\pi_\theta(\cdot \mid s_t) \;\Big\|\; \frac{\exp\big(Q_\phi^{(N)}(s_t, a_{t:t+N})/\alpha\big)}{Z(s_t)} \right) \Big], \tag{9}$$

where the states s_t are sampled from the replay buffer $\mathcal{D}$.

4.4 State Coverage in Action Chunking RL

Like most action chunking RL methods, we split the data of the replay buffer into new chunks for the policy and actor update. The policy and critic are thus updated on chunks that may start in arbitrary time steps. However, an action chunking policy with fixed chunk size N is only evaluated at every N-th state of the underlying MDP. Particularly in environments with deterministic or narrowly distributed initial states and large chunk sizes, it can happen that the policy is rarely evaluated on large parts of the state space. This effect is illustrated in the top panel of Fig. 2 for $N = 3$, where rectangles denote replanning states and circles indicate intermediate states visited only during open-loop execution. As shown in Section 5, we identify this as a key limitation of fixed-size action chunking. To address it, we propose *random replanning*: During data collection, we only execute a random prefix of each chunk. This increases the actor's state coverage, leading to more reliable updates for both the critic and the actor.

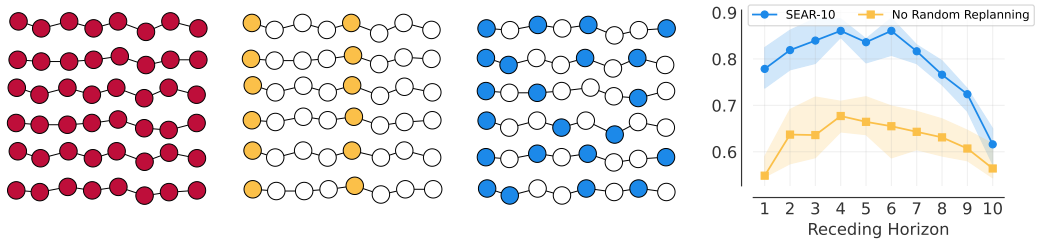
5 Experiments

We evaluate SEAR in two settings: purely online RL on Metaworld and offline-to-online RL on OGBench. We compare against state-of-the-art baselines in both settings and conduct ablation studies to understand the contribution of each design choice.

5.1 Experiment Setup

We evaluate SEAR in the online setting on the Metaworld ML1 Benchmark [34], which features a diverse set of 50 manipulation tasks. We follow Agarwal et al. [1] and report the IQM success and bootstrap 95% confidence intervals. Following prior work on Metaworld [32, 16, 15, 11], we only consider episodes to be successful if the success condition holds in the episode's last time step. This increases the task complexity and forces longer-term and stable behavior. Even with these stricter success criteria, many Metaworld tasks remain trivially solvable. We therefore focus on the 20 hardest tasks in Metaworld, as measured by SimbaV2's performance. For transparency, the IQM results for all 50 tasks are included in Appendix G. On Metaworld, all SEAR and SimbaV2 experiments were conducted with 10 seeds, while CQN-AS was evaluated on 5 seeds.

In the offline-to-online setting, we use the first four tasks of the OGBench [27] cube-triple-play domain, as they are the hardest manipulation tasks that QC and QC-FQL can solve with a dataset of 1M offline transitions [18]. All OGBench results were conducted with 8 seeds and include 1



(a) No Action Chunking | (b) Uniform Action Chunking | (c) SEAR's Random Replanning | (d) Metaworld-Hard IQM Success

Figure 2: **Uniform Action Chunking Limits Actor State Coverage.** In methods without action chunking like T-SAC and SimbaV2, the actor is evaluated in each state of the trajectory **(a)**. Prior action chunking methods evaluate the actor in every N-th state **(b)**. This creates areas of the state space where the actor is rarely evaluated during training (white dots). When evaluating such a policy with receding horizons, the actor is queried in these out-of-distribution states and the resulting performance drop offsets most gains from the higher decision frequency. SEAR introduces *random replanning* which breaks this pattern, restoring uniform state coverage **(c)**. As a result, SEAR benefits from receding horizon control to a much higher degree than methods without random replanning **(d)**.

million offline steps before transitioning to the online phase. We apply SEAR's multi-horizon critic targets and random replanning to the QC framework while keeping all other hyperparameters and network architectures fixed. Because QC queries the critic extensively, especially when combined with multi-horizon targets, we use an MLP critic rather than a transformer to keep wall-clock training times feasible.

We test SEAR with chunk sizes of 5, 10, and 20, and include chunk size 1 as a single-action baseline. As online RL baselines we use CQN-AS [30] with chunk sizes 5 and 10 and SimbaV2 [13], representing the state of the art with and without action chunking respectively. Unless otherwise noted, we report success after 1 million online environment steps.

All action chunking policies were evaluated using a receding horizon length of 5. For chunk size 5 policies, this has no effect, easing comparison to prior methods without receding horizons. Implementation details and hyperparameters are provided in Appendix D.

5.2 Comparison to State-of-the-Art

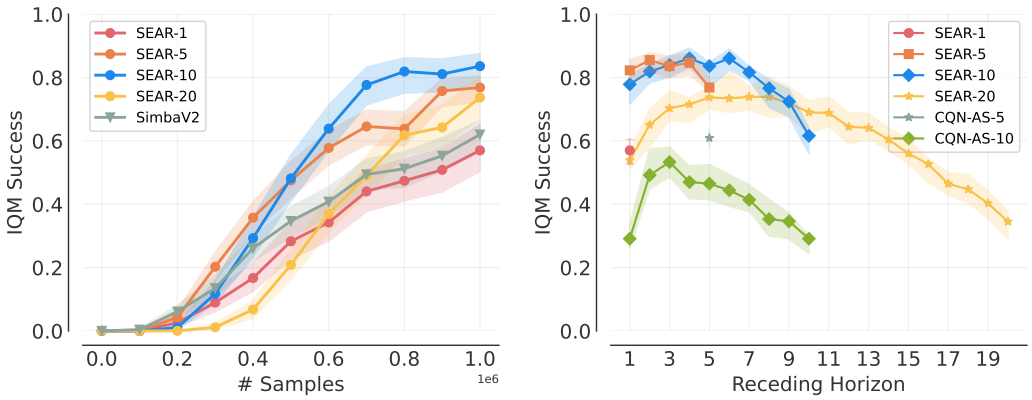
Figure 4a shows that SEAR achieves over 80% IQM success on Metaworld-Hard, substantially outperforming SimbaV2 and CQN-AS, the state of the art in online RL without and with action chunking respectively, which both plateau around 60%. A naive chunked extension of SimbaV2 achieves only around 20% IQM success, demonstrating that the action chunking techniques common in offline and offline-to-online RL do not transfer to the online setting without further algorithmic work. T-SAC, which chunks the critic but retains a single-step actor, performs similarly poorly.

In the offline-to-online setting (Figure 4b), applying SEAR's components to the QC framework outperforms both QC and QC-FQL across the OGBench cube-triple-play tasks, with the advantage most pronounced early in online training.

5.3 Why Does SEAR Outperform Prior Action Chunking Methods?

We attribute SEAR's gains to two design choices that address distinct failure modes of prior action chunking methods.

Actor state coverage. Figure 2d demonstrates that random replanning is necessary for effective use of receding horizon control. Without it, SEAR itself does not benefit from shorter receding horizons. Figure 3b shows the same failure mode in prior methods: CQN-AS-10 with a receding horizon does not match CQN-AS-5 without one.



(a) Sample Efficiency

(b) Effect of receding horizons

Figure 3: **SEAR works well across Chunk Sizes and benefits consistently from Receding Horizons.** We compare IQM Success on Metaworld-Hard for different chunk sizes. **(a)** All SEAR policies with action chunking achieve higher performance than SimbaV2, with only the single-action version SEAR-1 having marginally lower performance. The best performance is achieved by SEAR-10, indicating a favourable trade-off between value propagation and decision frequency. **(b)** For all tested chunk sizes, including one (i.e., no action chunking), training a policy on a larger chunk size and using it with a receding horizon yields higher performance than using a policy with a smaller chunk size and no receding horizon. All tested policies degrade in performance when increasing the receding horizon towards the chunk size, highlighting the challenge of reduced decision frequencies. Similarly, all action chunking policies drop in performance when reducing the receding horizon towards one. Across all tested chunks sizes (and including CQN-AS), the optimal receding horizon length lies between three and five.

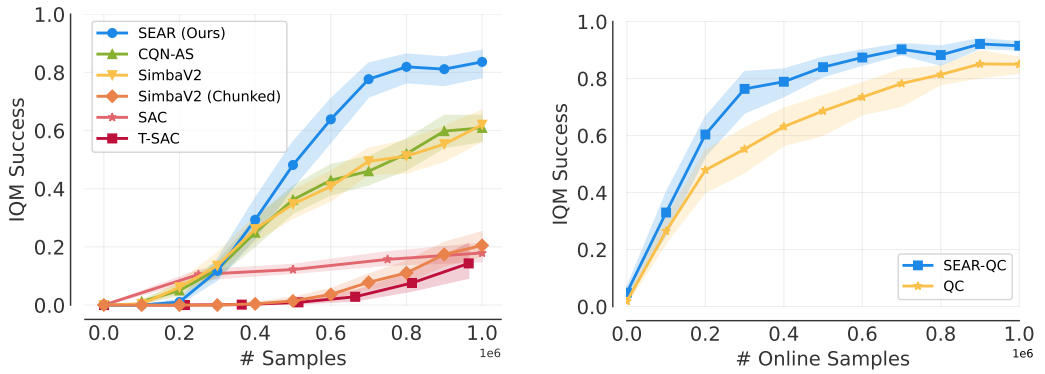
Critic design. The ablation in Figure 1 shows that multi-horizon targets are the most important component of SEAR's critic, as removing them and fitting only the full-chunk target $Q^{(N)}$ reduces performance to around 20% IQM success, comparable to the weakest baselines. Bootstrapping the transformer critic with action chunks rather than single-step actions also matters, as it allows the multi-horizon targets to improve value propagation. Replacing the transformer backbone with an MLP reduces performance, but SEAR with an MLP remains competitive, confirming that the transformer alone does not account for the gains.

Receding horizon analysis. Figure 3a shows that all SEAR policies with action chunking outperform SimbaV2, with SEAR-10 achieving the best final performance. SEAR-1 (no action chunking) performs slightly below SimbaV2, indicating that the use of a transformer critic is not the source of SEAR's performance gains. Figure 3b shows that a medium-size receding horizon is optimal. Horizons near the full chunk size reduce decision frequency and harm reactivity. For horizons near one, we frequently observed pausing behavior of the policy, leading to low success rates. SEAR policies achieve maximum performance at a receding horizon of three to five steps across all tested chunk sizes. Notably, SEAR policies trained with action chunking and deployed at a receding horizon of one outperform both SEAR-1 and SimbaV2, making action chunking with SEAR an effective way to obtain a high-performing single-action policy.

6 Conclusion

This work introduced SEAR, an action chunking algorithm for the previously underexplored online RL setting, achieving over 80% IQM success rate on Metaworld-Hard and substantially outperforming both action chunking and non-action chunking baselines.

The core of SEAR's design is that multi-horizon critic targets provide a proper critic fit over the enlarged action space of chunks. Given a well-fitted critic, random replanning restores the state coverage that open-loop chunk execution degrades, making receding horizon control effective. Together, these two components allow SEAR to benefit from the fast value propagation and structured



(a) Metaworld-Hard

(b) OGBench Cube-Triple-Play 1M

Figure 4: **Comparison to State-of-the-Art Baselines.** **(a)** In the online setting, SEAR achieves over 80% IQM success, significantly above the 60% success of CQN-AS and SimbaV2, which we consider to be the state-of-the art in online RL with and without action chunking respectively. As further baselines we include T-SAC which shares a similar critic with SEAR, SAC and a naive action chunking extention of SimbaV2. They achieve around 20% IQM success. **(b)** In the offline-to-online setting, we achieve higher IQM success rates than QC, especially early in training.

exploration of long chunks during training, while maintaining the reactivity of a short decision horizon at evaluation time. Through extensive ablation studies, we demonstrated that both components are necessary.

We further demonstrated that SEAR's multi-horizon targets and random replanning transfer to the offline-to-online setting. Applying them to the QC framework, while keeping all other components fixed, yields consistent improvements on the OGBench cube-triple-play tasks. This suggests that the core ideas behind SEAR are not specific to the online setting and can benefit action chunking methods more broadly.

Limitations Action chunking RL is best suited to sufficiently long-horizon tasks that do not require high reactivity, such as robotic manipulation, where action chunking aids exploration and task horizons are long enough to benefit from improved value propagation. Current action chunking methods do not work in high-reactivity domains like locomotion, as action chunking, even when only used during training, degrades data quality.

As such, there are few benchmarks available for that action chunking is a good fit. In this work, we focus on Metaworld [34] for online RL, which offers the currently broadest set of diverse manipulation tasks for state-based online RL. Other benchmarks such as ManiSkill [8] operate at very short episode horizons. In the offline-to-online setting, widely used benchmarks such as Robomimic [22] and most OGBench domains [27] are saturated, limiting insightful evaluation to a narrow set of tasks.

SEAR incurs a higher computation cost during training than single-step methods due to the multi-horizon targets and the transformer critic. However, compared to the state-of-the-art action chunking RL method CQN-AS [30], policy evaluation is fast, as action chunks are generated with a single neural network forward pass.

Outlook To make action chunking methods more broadly useful, including in domains requiring high reactivity such as locomotion, dynamic replanning algorithms would enable training and evaluation with an action chunk size that adapts to the current observation.

In addition, the computational overhead of the transformer critic could be reduced with more efficient sequence models such as state space models that keep the causal structure at a computational cost that is linear in the number of tokens.

Acknowledgments and Disclosure of Funding

The research presented in this paper was funded by the Horizon Europe Project XSCAVE under Grant 101189836. This work was further supported by the European Research Council (ERC) under the European Union's Horizon Europe programme through the project SMARTI^3 (Grant Agreement No. 101171393). The authors acknowledge support by the state of Baden-Württemberg through bwHPC, and the HoreKa supercomputer. This work has been supported by the German Federal Ministry of Research, Technology, and Space (BMFTR) under the Robotics Institute Germany (RIG). The authors gratefully acknowledge the Gauss Centre for Supercomputing e.V. (www.gauss-centre.eu) for funding this project by providing computing time on the GCS Supercomputer JUWELS at Jülich Supercomputing Centre (JSC).

References

[1] Agarwal, R., Schwarzer, M., Castro, P. S., Courville, A., and Bellemare, M. G. Deep reinforcement learning at the edge of the statistical precipice. In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=uqv8-U4lKBe.

[2] Agarwal, R., Schwarzer, M., Castro, P. S., Courville, A., and Bellemare, M. G. Deep reinforcement learning at the edge of the statistical precipice. *Advances in Neural Information Processing Systems*, 2021.

[3] Ball, P. J., Smith, L., Kostrikov, I., and Levine, S. Efficient online reinforcement learning with offline data. In *International Conference on Machine Learning*, pp. 1577–1594. PMLR, 2023.

[4] Black, K., Brown, N., Driess, D., Esmail, A., Equi, M., Finn, C., Fusai, N., Groom, L., Hausman, K., Ichter, B., et al. $\pi 0$: A vision-language-action flow model for general robot control. corr, abs/2410.24164, 2024. doi: 10.48550. *arXiv preprint ARXIV.2410.24164*, 2024.

[5] Celik, O., Li, Z., Blessing, D., Li, G., Palenicek, D., Peters, J., Chalvatzaki, G., and Neumann, G. DIME: Diffusion-based maximum entropy reinforcement learning. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=Aw6dBR7Vxj.

[6] Chi, C., Feng, S., Du, Y., Xu, Z., Cousineau, E., Burchfiel, B. C., and Song, S. Diffusion Policy: Visuomotor Policy Learning via Action Diffusion. In *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, July 2024. doi: 10.15607/RSS.2023.XIX.026.

[7] Fujimoto, S., Hoof, H., and Meger, D. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pp. 1587–1596. PMLR, 2018.

[8] Gu, J., Xiang, F., Li, X., Ling, Z., Liu, X., Mu, T., Tang, Y., Tao, S., Wei, X., Yao, Y., Yuan, X., Xie, P., Huang, Z., Chen, R., and Su, H. Maniskill2: A unified benchmark for generalizable manipulation skills. In *International Conference on Learning Representations*, 2023.

[9] Haarnoja, T., Tang, H., Abbeel, P., and Levine, S. Reinforcement learning with deep energy-based policies. In *International conference on machine learning*, pp. 1352–1361. PMLR, 2017.

[10] Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. Pmlr, 2018.

[11] Joshi, V., Xu, Z., Liu, B., Stone, P., and Zhang, A. Benchmarking massively parallelized multi-task reinforcement learning for robotics tasks. *ArXiv*, abs/2507.23172, 2025. URL https://api.semanticscholar.org/CorpusID:280401281.

[12] Lahoti, A., Li, K., Chen, B., Wang, C., Bick, A., Kolter, J. Z., Dao, T., and Gu, A. Mamba-3: Improved sequence modeling using state space principles. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=HwCvaJOiCj.

[13] Lee, H., Lee, Y., Seno, T., Kim, D., Stone, P., and Choo, J. Hyperspherical normalization for scalable deep reinforcement learning. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=kfYxyvCYQ4.

[14] Lee, J., Duan, J., Fang, H., Deng, Y., Liu, S., Li, B., Fang, B., Zhang, J., Wang, Y. R., Lee, S., Han, W., Pumacay, W., Wu, A., Hendrix, R., Farley, K., VanderBilt, E., Farhadi, A., Fox, D., and Krishna, R. Molmoact: Action reasoning models that can reason in space, 2025. URL https://arxiv.org/abs/2508.07917.

[15] Li, G., Zhou, H., Roth, D., Thilges, S., Otto, F., Lioutikov, R., and Neumann, G. Latent space exploration and trajectory space update in temporally-correlated episodic reinforcement learning. In *ICRA 2024 Workshop—Back to the Future: Robot Learning Going Probabilistic*, 2024. URL https://openreview.net/forum?id=e8dcuniLcA.

[16] Li, G., Tian, D., Zhou, H., Jiang, X., Lioutikov, R., and Neumann, G. TOP-ERL: Transformer-based off-policy episodic reinforcement learning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=N4NhVN30ph.

[17] Li, Q., Park, S., and Levine, S. Decoupled q-chunking, 2025. URL https://arxiv.org/abs/2512.10926.

[18] Li, Q., Zhou, Z., and Levine, S. Reinforcement learning with action chunking. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=XUks1Y96NR.

[19] Li, Y., Ma, X., Xu, J., Cui, Y., Cui, Z., Han, Z., Huang, L., Kong, T., Liu, Y., Niu, H., Peng, W., Qiao, J., Ren, Z., Shi, H., Su, Z., Tian, J., Xiao, Y., Zhang, S., Zheng, L., Li, H., and Wu, Y. Gr-rl: Going dexterous and precise for long-horizon robotic manipulation, 2025. URL https://arxiv.org/abs/2512.01801.

[20] Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[21] Luo, J., Xu, C., Wu, J., and Levine, S. Precise and dexterous robotic manipulation via human-in-the-loop reinforcement learning. *Science Robotics*, 10(105), August 2025. doi: 10.1126/scirobotics.ads5033. URL https://www.science.org/doi/10.1126/scirobotics.ads5033.

[22] Mandlekar, A., Xu, D., Wong, J., Nasiriany, S., Wang, C., Kulkarni, R., Fei-Fei, L., Savarese, S., Zhu, Y., and Martín-Martín, R. What matters in learning from offline human demonstrations for robot manipulation. In *arXiv preprint arXiv:2108.03298*, 2021.

[23] Munos, R., Stepleton, T., Harutyunyan, A., and Bellemare, M. Safe and efficient off-policy reinforcement learning. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper_files/paper/2016/file/c3992e9a68c5ae12bd18488bc579b30d-Paper.pdf.

[24] Nair, S., Rajeswaran, A., Kumar, V., Finn, C., and Gupta, A. R3m: A universal visual representation for robot manipulation. In *6th Annual Conference on Robot Learning*, 2022. URL https://openreview.net/forum?id=tGbpgz6yOrI.

[25] Nikolov, N., Albanese, G., Dey, S., Yanev, A., Gool, L. V., Zaech, J.-N., and Paudel, D. P. Spear-1: Scaling beyond robot demonstrations via 3d understanding, 2025. URL https://arxiv.org/abs/2511.17411.

[26] Park, K., Park, S., Lee, Y., and Levine, S. Scalable offline model-based rl with action chunks, 2025. URL https://arxiv.org/abs/2512.08108.

[27] Park, S., Frans, K., Eysenbach, B., and Levine, S. OGBench: Benchmarking offline goal-conditioned RL. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=M992mjgKzI.

[28] Physical Intelligence, Amin, A., Aniceto, R., Balakrishna, A., Black, K., Conley, K., Connors, G., Darpinian, J., Dhabalia, K., DiCarlo, J., Driess, D., Equi, M., Esmail, A., Fang, Y., Finn, C., Glossop, C., Godden, T., Goryachev, I., Groom, L., Hancock, H., Hausman, K., Hussein, G., Ichter, B., Jakubczak, S., Jen, R., Jones, T., Katz, B., Ke, L., Kuchi, C., Lamb, M., LeBlanc, D., Levine, S., Li-Bell, A., Lu, Y., Mano, V., Mothukuri, M., Nair, S., Pertsch, K., Ren, A. Z., Sharma, C., Shi, L. X., Smith, L., Springenberg, J. T., Stachowicz, K., Stoeckle, W., Swerdlow, A., Tanner, J., Torne, M., Vuong, Q., Walling, A., Wang, H., Williams, B., Yoo, S., Yu, L., Zhilinsky, U., and Zhou, Z. $\pi^*_{0.6}$: a vla that learns from experience, 2025. URL https://arxiv.org/abs/2511.14759.

[29] Reuss, M., Pari, J., Agrawal, P., and Lioutikov, R. Efficient diffusion transformer policies with mixture of expert denoisers for multitask learning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=nDmwloEl3N.

[30] Seo, Y. and Abbeel, P. Coarse-to-fine q-network with action sequence for data-efficient reinforcement learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=VoFXUNc9Zh.

[31] Sutton, R. S., Barto, A. G., et al. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[32] Tian, D., Li, G., Zhou, H., Celik, O., and Neumann, G. Chunking the critic: A transformer-based soft actor-critic with n-step returns, 2025. URL https://arxiv.org/abs/2503.03660.

[33] Yang, J., Zhu, B., Chen, J., and Jiang, Y.-G. Actor-critic for continuous action chunks: A reinforcement learning framework for long-horizon robotic manipulation with sparse reward. *arXiv preprint arXiv:2508.11143*, 2025.

[34] Yu, T., Quillen, D., He, Z., Julian, R., Narayan, A., Shively, H., Bellathur, A., Hausman, K., Finn, C., and Levine, S. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning, 2021. URL https://arxiv.org/abs/1910.10897.

[35] Zhang, Y., Yu, S., Zhang, T., Guang, M., Hui, H., Long, K., Wang, Y., Yu, C., and Ding, W. Sac flow: Sample-efficient reinforcement learning of flow-based policies via velocity-reparameterized sequential modeling, 2026. URL https://arxiv.org/abs/2509.25756.

[36] Zhao, T. Z., Kumar, V., Levine, S., and Finn, C. Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware. In *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, July 2023. doi: 10.15607/RSS.2023.XIX.016.

[37] Zheng, J., Li, J., Wang, Z., Liu, D., Kang, X., Feng, Y., Zheng, Y., Zou, J., Chen, Y., Zeng, J., Zhang, Y.-Q., Pang, J., Liu, J., Wang, T., and Zhan, X. X-vla: Soft-prompted transformer as scalable cross-embodiment vision-language-action model, 2025. URL https://arxiv.org/abs/2510.10274.

[38] Ziebart, B. D. Modeling purposeful adaptive behavior with the principle of maximum causal entropy. 2010. URL https://api.semanticscholar.org/CorpusID:11919065.

[39] Ziebart, B. D., Maas, A. L., Bagnell, J. A., Dey, A. K., et al. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pp. 1433–1438. Chicago, IL, USA, 2008.

A Algorithm

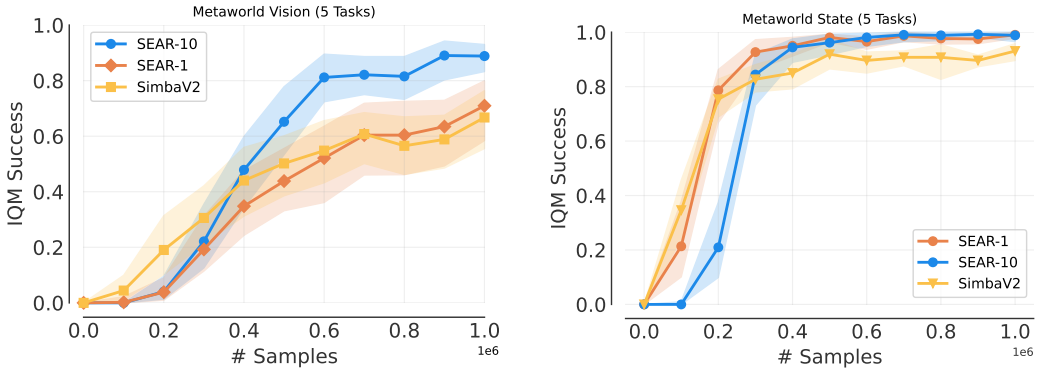
A.1 Pseudocode

B Vision Experiments

To test how SEAR handles vision observations, we trained SEAR-10 and SEAR-1 on six Metaworld tasks with visual observations: hammer-v3, handle-pull-side-v3, peg-unplug-side-v3, handle-pull-v3, plate-slide-side-v3 and door-open-v3.

Algorithm 1 Sample Efficient Action Chunking RL (SEAR)

Initialize actor parameters θ, critic parameters ϕ_1, ϕ_2, target critic parameters $\bar{\phi}_1 \leftarrow \phi_1, \bar{\phi}_2 \leftarrow \phi_2$
for step $= 1, 2, \ldots$ **do**
  # Data collection with random prefix execution
  if $|\mathcal{D}| <$ step **then**
    Sample action chunk $a_{t:t+N} \sim \pi_\theta(\cdot \mid s_t)$
    Sample random prefix length $k \sim \text{Uniform}(1, N)$
    for $i = 0$ to $k - 1$ **do**
      Execute a_{t+i} in environment, observe r_{t+i}, s_{t+i+1}
      Add transition $(s_{t+i}, a_{t+i}, r_{t+i}, s_{t+i+1})$ to $\mathcal{D}$
    end for
  end if
  # Update Networks
  Sample batch of chunks $(s_t, a_{t:t+N}, r_{t:t+N}, s_{t+N})$ from $\mathcal{D}$
  Update critics by minimizing $J(\phi_{1,2})$ (Equation 8)
  Update actor by minimizing $J(\theta)$ (Equation 9)
  update α with $\mathcal{H}_{\text{target}} = N * \dim \mathcal{A}$ [10]
  # Update target networks
  $\bar{\phi}_j \leftarrow \tau\phi_j + (1 - \tau)\bar{\phi}_j$ for $j \in \{1, 2\}$
end for



(a) Metaworld-Vision

(b) State-based Results on the same Tasks.

Figure 5: **SEAR-10 works well with visual observations. (a)** Across five visual manipulation tasks, SEAR-10 outperforms both its non-action chunking variant SEAR-1, as well as the non-chunking baseline SimbaV2. **(b)** With state observations, SEAR-10 does not outperform SEAR-1 on the tasks chosen for visual manipulation. This makes it unlikely that the vision results are explained by the specific environments tested. The per-task training curves are provided in Figure 10.

We chose these tasks because both SEAR-10 and the non-chunking baselines (SEAR-1 and SimbaV2) solved them within 500k steps in the state-based setting (Appendix Figure 9), which let us obtain a meaningful, non-floor/non-ceiling comparison. To isolate the effects of SEAR's action chunking, took care not to choose tasks where SEAR-10 already compares favorably to SEAR-1 with state observations. Following recent work on real-world, vision-based RL [21], we encoded the observations using the pretrained ResNet-50 encoder presented in [24].

As seen in Figure 5, SEAR-10 has a distinct advantage over SEAR-1 in both sample efficiency and final performance when using visual observations. At 1M online steps, SEAR-10 has an IQM success of 93% vs SEAR-1 with 80%. Compared to both SEAR variants, SimbaV2 has the advantage until around 400k environment steps, but the final performance at 1M steps is comparable to SEAR-1 at 76% ICQ success. This suggests that SEAR's action chunking is effective at visual manipulation. The per-task training curves are provided in Figure 10.

C Implementation Details

An implementation of our algorithm is provided as part of the Supplementary Material.

D Experiment Details

D.1 Compute Demands

All experiments were performed on NVIDIA A100 GPUs.

OGBench. We evaluate three methods across 4 tasks with 8 seeds each: QC-SEAR ($11\,\text{h} \times 8 \times 4 = 352$ GPU hours), QC ($4\,\text{h} \times 8 \times 4 = 128$ GPU hours), and QC-FQL ($3\,\text{h} \times 8 \times 4 = 96$ GPU hours). Subtotal: 576 GPU hours.

MetaWorld. SEAR main results (4 chunk sizes, 50 tasks, 10 seeds): $3\,\text{h} \times 10 \times 4 \times 50 = 6{,}000$ GPU hours. SEAR ablations (3 ablations, 20 tasks, 10 seeds): $3\,\text{h} \times 10 \times 3 \times 20 = 1{,}800$ GPU hours. CQN-AS (2 chunk sizes, 20 tasks, 5 seeds): $16\,\text{h} \times 5 \times 2 \times 20 = 3{,}200$ GPU hours. SimbaV2 (50 tasks, 10 seeds): $2\,\text{h} \times 10 \times 50 = 1{,}000$ GPU hours. Subtotal: 12,000 GPU hours.

Total: 12,576 GPU hours.

For SEAR, four seeds are run in parallel on a single GPU, so the 3h per seed above reflects $12\,\text{h}/4$ wall-clock time rather than the runtime of a single training run. We run multiple seeds in parallel as the extensive evaluation, as well as our current replay buffer implementation limit GPU utilization. These numbers represent a conservative estimate of the required compute, as we used the runtime of SEAR-10 for all SEAR variants, while SEAR-1 and SEAR-5 train faster. Additionally, the run times of SEAR and CQN-AS include evaluation across all possible receding horizons, which takes significant time. The above does not include compute spent developing the method and debugging its implementation.

D.2 SEAR

SEAR's main hyperparameter besides the chunk size is the embedding dimension of the transformer ("hidden dim" in the talbe below). We tried 128, 256 and 512 and chose 512 as the smaller ones had insufficient capacity. With regard to loss functions we tried l2 and C51 and chose the latter because it stabilized training. All other hyper parameters were chosen with common values from prior methods and not specifically optimized.

D.3 SimbaV2

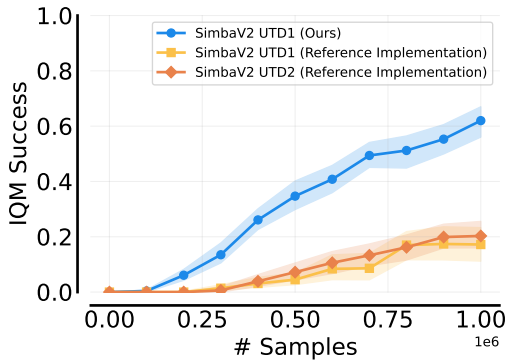


Figure 6: Comparison between our SimbaV2 implementation and the reference implementation.

For SimbaV2, we use our reimplementation within the same training pipeline as SEAR and disable the reference reward scaling, since Meta-World rewards are already on a consistent scale across tasks,

while keeping all other components and techniques described in Section 4 of SimbaV2 [13]. For transparency, we also report results with the original implementation, shown in Fig. 6.

D.4 Hyperparameter values

Table 1 gives an overview of the hyperparameters that we use for SEAR and baselines. Below, we provide further details on the specific hyperparameter choices. The full hyperparameters and algorithm-specific options can be found as part of the Supplementary Material.

Table 1: Hyperparameters used in our experiments. CQN-AS has no actor component. SIMBAV2-MLP denotes the MLP head used in SimbaV2, including its spherical normalization. $\dim(A)$ denotes the dimensionality of a single action, and N is the chunk size.

Component	**SEAR**	**CQN-AS**	**SimbaV2**
Actor			
architecture	SimbaV2-MLP	N/A	SimbaV2-MLP
hidden dimension	512	N/A	512
number of blocks	1	N/A	2
Critic			
architecture	Transformer	MLP + GRU	SimbaV2-MLP
number of bins	101	101	101
value range	$[0, 1000]$	$[0, 1000]$	$[0, 1000]$
hidden dimension	512	1024	512
number of heads	16	N/A	N/A
number of blocks	2	1	2
Algorithm			
total timesteps	1,000,000	1,000,000	1,000,000
seed timesteps	0	4,000	0
update-to-data ratio (UTD)	1	1	1
chunk size	1 / 5 / 10 / 20	5 / 10	1
batch size	256	256	256
discount γ	0.99	0.99	0.99
target critic update ratio	$5 \cdot 10^{-2}$	1.0	$5 \cdot 10^{-2}$
target entropy	$-\dim(A^N)$	N/A	$-\dim(A)$
Optimization			
critic optimizer	AdamW	AdamW	AdamW
learning rate	$3 \cdot 10^{-4}$	$5 \cdot 10^{-5}$	$3 \cdot 10^{-4}$
weight decay	$1 \cdot 10^{-4}$	0.1	$1 \cdot 10^{-4}$
β_1	0.9	0.9	0.9
β_2	0.999	0.999	0.999

CQN-AS As the authors of [30] neither evaluate on nor provide hyperparameters for Metaworld [34], we tuned the hyperparameters of CQN-AS specifically for the 20 hardest MetaWorld task. Note that, since we operate in a purely online RL setting, we do not use the behavior cloning component of CQN-AS. We use an exploration noise schedule that linearly decreases from 0.1 to 0.01 until step $500,000$, and is then kept constant for the remaining training time. We also adopt the dueling Q-networks and distributional critic that [30] use only for vision-based environments, as we found it to improve performance. To align CQN-AS with SEAR, we fix the update-to-data at one. We further tuned the learning rate and batch size, although both had only a minor impact on the final success rate in our experiments. All other hyperparameters were adopted from the state-based tasks of [30]. We found that a larger critic or using more bins or levels for the coarse-to-fine action selection algorithm does not improve the success rate on Metaworld. See Table 1 for a summary of the hyperparameters.

D.5 Sample Efficiency Comparison of SEAR, SimbaV2, and CQN-AS

Here, we provide further details on the experimental setup for the comparison of the sample efficiency of SEAR and baseline methods in Figure 4. The naively chunked SimbaV2 baseline uses action chunking for the actor and critic, but without multi-horizon returns and random replanning. SEAR and the naively chunked SimbaV2 baseline use a chunk size of 10 with a receding horizon of 5 during evaluation, which performs best according to the ablations in Section 5.3. CQN-AS uses a chunk size of 5, which performed better than 10 in our experiments, and no receding horizon control.

E Evalution

E.1 Metaworld

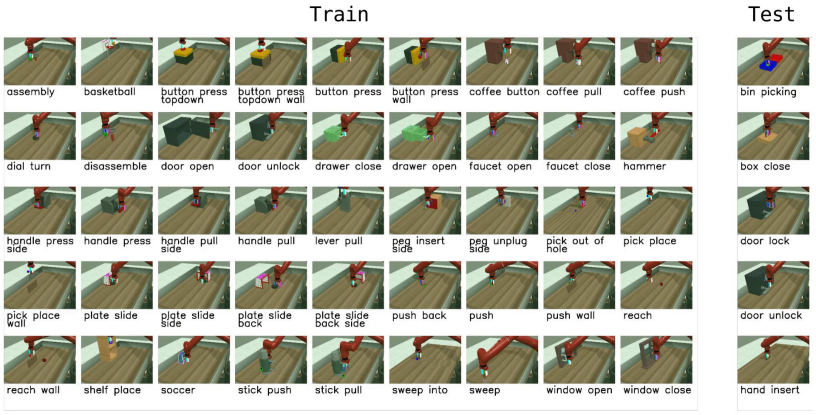


Figure 7: **All tasks in the Metaworld benchmark [34]**

Metaworld [34] is an open-source benchmark for meta-reinforcement learning and multi-task robotic manipulation. It contains 50 diverse manipulation tasks that span skills such as grasping, pushing, insertion, and object placement. Unlike benchmarks focused on a narrow task distribution, Metaworld provides a broad and challenging testbed for algorithms that aim to generalize across behaviors. Figure 7 illustrates the full MT50 suite, highlighting the variety and complexity of the tasks.

F Computational Efficiency

As implementation details of the replay buffer and the environment obscure the computational requirements of the methods, we compare the time for inference (i.e., sampling a chunk) and policy updates (i.e., updating the critic and policy). This ensures fairness, as CQN-AS's replay buffer implementation is optimized for image observations and thus less efficient for state observations. All measurements were obtained with UTD=1 on an RTX 4500 Blackwell GPU:

Method	Inference Time [ms]	Time per Update [ms]	Update Time per Environment Step [ms]
CQN-AS-1	5.0	14.6	14.6
CQN-AS-5	5.5	18.6	18.6
CQN-AS-10	5.8	29.8	29.8
CQN-AS-20	6.3	60.4	60.4
SEAR-1	1.7	9.7	9.7
SEAR-5	1.7	22.6	9.0
SEAR-10	1.7	50.7	10.1
SEAR-20	1.7	126.6	12.7
SEAR-1 (MLP)	1.7	5.0	5.0
SEAR-5 (MLP)	1.7	7.0	2.8
SEAR-10 (MLP)	1.7	7.9	1.6
SEAR-20 (MLP)	1.7	9.0	0.9
SimbaV2	1.7	5.0	5.0

First, we observe that SEAR(-MLP) and SimbaV2 (which share the same actor) are faster at inference than the critic-only method CQN-AS.

Second, regarding update time, we observe that all action-chunking methods exhibit increasing update times with higher chunk sizes. As a reference point, SimbaV2 spends only 5ms per environment step on updates. As SEAR does only one update per chunk (which is on average executed halfway, due to random replanning), SEAR spends 9-13ms per environment step on updates, roughly 1.8-2.5x the time of SimbaV2.

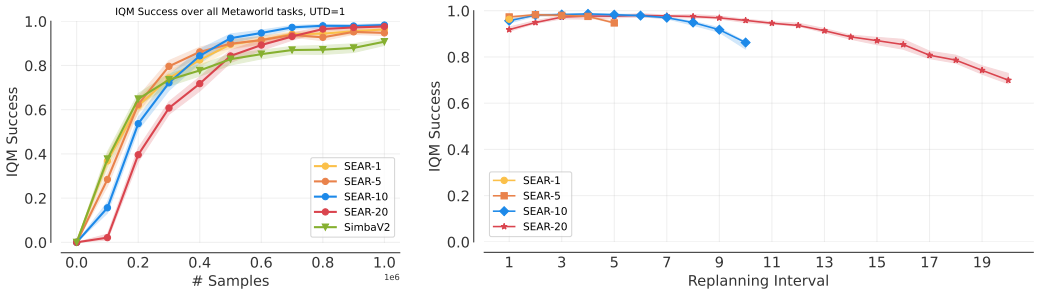
CQN-AS does one update per environment step, which takes between 14ms and 60ms, depending on chunk size.

SEAR-10's ~2x update cost per environment step means running SEAR-10 for 500k samples costs about as much update wall-clock time as running SimbaV2 for 1M samples. On Metaworld-Hard, SimbaV2 has an IQM success of 62% at 1M steps, which is better than SEAR-10's 48% success at 500k steps. Under a comparable compute budget, SimbaV2 thus has an advantage. How relevant this is when including time spent on environment interaction, evaluation and buffer operations will be application dependent.

Future work could improve the runtime of SEAR in multiple ways. The most straightforward way would be to exchange the transformer for a more compute-efficient backbone like Mamba 3 [12]. Furthermore, the critic update could be made more compute-efficient by using a value function for the multi-horizon targets, like TOP-ERL.

G Extended Results

As Metaworld contains a number of very simple tasks, we focus our analysis in the main text on a set of the 20 hardest tasks as measured by SimbaV2's performance. In this section we provide the per-environment performance of the methods discussed in the main text. Table 4 provides a list of the 20 hard tasks used in the main text and the per-task performance of SEAR-10, CQN-AS and SimbaV2. Table 5 contains the tasks that were not used for the main text along with the per-task performance of SEAR-1,5,10,20 and SimbaV2. To demonstrate that the key results of the main text, namely SEAR's superior sample efficiency and its effectiveness when using receding horizons are



(a) Sample Efficiency

(b) Effect of receding horizons

Figure 8: The experiments of 3, but over all 50 Metaworld environments, not just the 20 hardest that were used in the main text. SEAR still reaches a higher success rate than SimbaV2. The effect of a receding horizon is very similar to the result of the main text: $k = 1$ is suboptimal, as is $k = N$. The optimal value is around $k = 4$ independent of chunk size.

not dependent on the choice of tasks, we reproduced Figure 3 of the main text over all 50 tasks in Figure 8. Finally, we include learning curves for all environments in Figure 9.

	SEAR-1	SEAR-5	SEAR-10	SEAR-20
Average	0.57 $^{+0.07}_{-0.08}$	0.84 $^{+0.03}_{-0.04}$	0.84 $^{+0.04}_{-0.05}$	0.70 $^{+0.06}_{-0.07}$
assembly-v3	0.20 $^{+0.43}_{-0.20}$	0.00 $^{+0.00}_{-0.00}$	0.00 $^{+0.00}_{-0.00}$	0.00 $^{+0.17}_{-0.00}$
basketball-v3	0.01 $^{+0.15}_{-0.01}$	0.69 $^{+0.17}_{-0.20}$	0.97 $^{+0.03}_{-0.06}$	0.97 $^{+0.03}_{-0.11}$
button-press-v3	1.00 $^{+0.00}_{-0.05}$	0.96 $^{+0.03}_{-0.18}$	1.00 $^{+0.00}_{-0.00}$	1.00 $^{+0.00}_{-0.01}$
coffee-pull-v3	0.92 $^{+0.06}_{-0.40}$	0.94 $^{+0.04}_{-0.08}$	0.97 $^{+0.03}_{-0.07}$	0.81 $^{+0.08}_{-0.08}$
coffee-push-v3	0.67 $^{+0.18}_{-0.33}$	0.70 $^{+0.09}_{-0.10}$	0.80 $^{+0.08}_{-0.05}$	0.76 $^{+0.12}_{-0.09}$
disassemble-v3	0.14 $^{+0.38}_{-0.14}$	0.44 $^{+0.40}_{-0.40}$	0.00 $^{+0.01}_{-0.00}$	0.00 $^{+0.17}_{-0.00}$
hammer-v3	1.00 $^{+0.00}_{-0.17}$	0.99 $^{+0.01}_{-0.16}$	1.00 $^{+0.00}_{-0.30}$	1.00 $^{+0.00}_{-0.17}$
hand-insert-v3	0.71 $^{+0.09}_{-0.10}$	0.75 $^{+0.18}_{-0.20}$	0.93 $^{+0.07}_{-0.25}$	0.66 $^{+0.17}_{-0.12}$
handle-pull-v3	0.94 $^{+0.02}_{-0.11}$	0.78 $^{+0.14}_{-0.25}$	0.95 $^{+0.05}_{-0.30}$	0.54 $^{+0.33}_{-0.42}$
lever-pull-v3	0.72 $^{+0.12}_{-0.17}$	0.76 $^{+0.17}_{-0.16}$	0.60 $^{+0.17}_{-0.23}$	0.53 $^{+0.23}_{-0.15}$
peg-unplug-side-v3	0.97 $^{+0.03}_{-0.06}$	0.88 $^{+0.06}_{-0.07}$	0.99 $^{+0.01}_{-0.05}$	0.98 $^{+0.02}_{-0.08}$
pick-out-of-hole-v3	0.99 $^{+0.01}_{-0.03}$	0.96 $^{+0.03}_{-0.08}$	1.00 $^{+0.00}_{-0.03}$	1.00 $^{+0.00}_{-0.19}$
pick-place-v3	0.27 $^{+0.28}_{-0.21}$	0.63 $^{+0.16}_{-0.12}$	0.90 $^{+0.07}_{-0.10}$	0.74 $^{+0.18}_{-0.37}$
pick-place-wall-v3	0.00 $^{+0.00}_{-0.00}$	0.00 $^{+0.02}_{-0.00}$	0.00 $^{+0.01}_{-0.00}$	0.00 $^{+0.00}_{-0.00}$
push-back-v3	0.82 $^{+0.18}_{-0.47}$	0.97 $^{+0.03}_{-0.05}$	0.97 $^{+0.03}_{-0.20}$	0.99 $^{+0.01}_{-0.17}$
push-v3	0.62 $^{+0.26}_{-0.33}$	0.82 $^{+0.12}_{-0.15}$	0.96 $^{+0.04}_{-0.09}$	0.99 $^{+0.01}_{-0.04}$
shelf-place-v3	0.00 $^{+0.17}_{-0.00}$	0.38 $^{+0.45}_{-0.38}$	0.22 $^{+0.45}_{-0.22}$	0.00 $^{+0.12}_{-0.00}$
soccer-v3	0.42 $^{+0.17}_{-0.19}$	0.61 $^{+0.08}_{-0.07}$	0.72 $^{+0.13}_{-0.20}$	0.80 $^{+0.08}_{-0.08}$
stick-pull-v3	0.08 $^{+0.36}_{-0.08}$	0.53 $^{+0.31}_{-0.40}$	0.79 $^{+0.08}_{-0.40}$	0.55 $^{+0.27}_{-0.35}$
stick-push-v3	0.09 $^{+0.46}_{-0.04}$	0.73 $^{+0.17}_{-0.28}$	0.47 $^{+0.29}_{-0.26}$	0.12 $^{+0.48}_{-0.12}$

Table 3: IQM success with 95% bootstrapped confidence interval across the 20 hardest Metaworld tasks, comparing SEAR with various chunk sizes. The best result is highlighted, results that are non-significantly worse that the best are lightly highlighted. All policies except for SEAR-1 use a fixed receding horizon of five actions during evaluation. On average, SEAR-5 and SEAR-10 perform best, with SEAR-10 outperforming SEAR-5 on individual environments.

	SEAR-10	CQN-AS-5	CQN-AS-10	SimbaV2
Average	0.84 $^{+0.04}_{-0.05}$	0.61 $^{+0.04}_{-0.05}$	0.47 $^{+0.06}_{-0.06}$	0.62 $^{+0.05}_{-0.06}$
assembly-v3	0.00 $^{+0.00}_{-0.00}$	0.88 $^{+0.05}_{-0.03}$	0.60 $^{+0.20}_{-0.35}$	0.92 $^{+0.08}_{-0.08}$
basketball-v3	0.97 $^{+0.03}_{-0.06}$	0.25 $^{+0.03}_{-0.10}$	0.15 $^{+0.03}_{-0.07}$	0.27 $^{+0.36}_{-0.17}$
button-press-v3	1.00 $^{+0.00}_{-0.00}$	0.43 $^{+0.42}_{-0.30}$	0.15 $^{+0.20}_{-0.15}$	0.95 $^{+0.03}_{-0.07}$
coffee-pull-v3	0.97 $^{+0.03}_{-0.08}$	0.60 $^{+0.15}_{-0.13}$	0.55 $^{+0.03}_{-0.00}$	0.88 $^{+0.10}_{-0.20}$
coffee-push-v3	0.80 $^{+0.08}_{-0.05}$	0.43 $^{+0.15}_{-0.08}$	0.37 $^{+0.18}_{-0.13}$	0.38 $^{+0.37}_{-0.22}$
disassemble-v3	0.00 $^{+0.01}_{-0.00}$	0.92 $^{+0.07}_{-0.62}$	0.35 $^{+0.58}_{-0.35}$	0.62 $^{+0.08}_{-0.42}$
hammer-v3	1.00 $^{+0.00}_{-0.30}$	0.98 $^{+0.02}_{-0.07}$	1.00 $^{+0.00}_{-0.03}$	0.97 $^{+0.03}_{-0.05}$
hand-insert-v3	0.93 $^{+0.07}_{-0.24}$	0.73 $^{+0.10}_{-0.15}$	0.72 $^{+0.17}_{-0.55}$	0.53 $^{+0.32}_{-0.22}$
handle-pull-v3	0.95 $^{+0.05}_{-0.31}$	0.98 $^{+0.02}_{-0.05}$	0.98 $^{+0.02}_{-0.03}$	0.80 $^{+0.08}_{-0.27}$
lever-pull-v3	0.60 $^{+0.14}_{-0.20}$	0.42 $^{+0.08}_{-0.18}$	0.12 $^{+0.23}_{-0.07}$	0.45 $^{+0.17}_{-0.12}$
peg-unplug-side-v3	0.99 $^{+0.01}_{-0.04}$	0.85 $^{+0.07}_{-0.08}$	0.88 $^{+0.12}_{-0.13}$	0.72 $^{+0.05}_{-0.22}$
pick-out-of-hole-v3	1.00 $^{+0.00}_{-0.02}$	0.57 $^{+0.22}_{-0.12}$	0.62 $^{+0.05}_{-0.32}$	0.68 $^{+0.08}_{-0.17}$
pick-place-v3	0.90 $^{+0.07}_{-0.10}$	0.58 $^{+0.18}_{-0.17}$	0.48 $^{+0.22}_{-0.40}$	0.00 $^{+0.17}_{-0.00}$
pick-place-wall-v3	0.00 $^{+0.01}_{-0.00}$	0.12 $^{+0.30}_{-0.12}$	0.18 $^{+0.24}_{-0.18}$	0.00 $^{+0.33}_{-0.00}$
push-back-v3	0.97 $^{+0.03}_{-0.19}$	0.90 $^{+0.08}_{-0.07}$	0.97 $^{+0.03}_{-0.65}$	0.90 $^{+0.08}_{-0.30}$
push-v3	0.96 $^{+0.04}_{-0.09}$	0.70 $^{+0.05}_{-0.30}$	0.57 $^{+0.03}_{-0.05}$	0.65 $^{+0.16}_{-0.12}$
shelf-place-v3	0.22 $^{+0.43}_{-0.22}$	0.43 $^{+0.40}_{-0.33}$	0.00 $^{+0.00}_{-0.00}$	0.30 $^{+0.37}_{-0.30}$
soccer-v3	0.72 $^{+0.13}_{-0.19}$	0.48 $^{+0.08}_{-0.22}$	0.32 $^{+0.12}_{-0.13}$	0.52 $^{+0.16}_{-0.02}$
stick-pull-v3	0.79 $^{+0.08}_{-0.39}$	0.37 $^{+0.05}_{-0.22}$	0.32 $^{+0.20}_{-0.29}$	0.50 $^{+0.25}_{-0.32}$
stick-push-v3	0.47 $^{+0.29}_{-0.27}$	0.45 $^{+0.12}_{-0.25}$	0.62 $^{+0.18}_{-0.33}$	0.25 $^{+0.47}_{-0.17}$

Table 4: IQM success rate with 95% bootstrapped confidence interval across the 20 hardest Metaworld tasks, comparing SEAR-10, the best performing SEAR variant, with baselines. The best result is highlighted, results that are non-significantly worse that the best are lightly highlighted. All policies except for SimbaV2 use a fixed receding horizon of five actions during evaluation. CQN-AS performs better with chunk size five (CQN-AS-5) than with chunk size ten (CQN-AS-10). Overall, SEAR-10 performs best.

	SEAR-1	SEAR-5	SEAR-10	SEAR-20	SimbaV2
Average	1.00 $^{+0.00}_{-0.00}$	0.99 $^{+0.00}_{-0.00}$	1.00 $^{+0.00}_{-0.00}$	1.00 $^{+0.00}_{-0.00}$	0.99 $^{+0.00}_{-0.01}$
bin-picking-v3	0.96 $^{+0.04}_{-0.47}$	0.89 $^{+0.10}_{-0.27}$	0.75 $^{+0.20}_{-0.46}$	0.53 $^{+0.38}_{-0.40}$	0.80 $^{+0.18}_{-0.30}$
box-close-v3	1.00 $^{+0.00}_{-0.03}$	0.83 $^{+0.05}_{-0.06}$	0.95 $^{+0.04}_{-0.10}$	0.97 $^{+0.03}_{-0.17}$	0.92 $^{+0.06}_{-0.08}$
button-press-topdown-v3	1.00 $^{+0.00}_{-0.01}$	1.00 $^{+0.00}_{-0.03}$	1.00 $^{+0.00}_{-0.00}$	1.00 $^{+0.00}_{-0.00}$	0.98 $^{+0.02}_{-0.03}$
button-press-topdown-wall-v3	1.00 $^{+0.00}_{-0.00}$	1.00 $^{+0.00}_{-0.01}$	1.00 $^{+0.00}_{-0.00}$	1.00 $^{+0.00}_{-0.00}$	1.00 $^{+0.00}_{-0.00}$
button-press-wall-v3	1.00 $^{+0.00}_{-0.00}$	1.00 $^{+0.00}_{-0.00}$	1.00 $^{+0.00}_{-0.18}$	1.00 $^{+0.00}_{-0.01}$	0.98 $^{+0.02}_{-0.08}$
coffee-button-v3	1.00 $^{+0.00}_{-0.03}$	1.00 $^{+0.00}_{-0.03}$	1.00 $^{+0.00}_{-0.01}$	1.00 $^{+0.00}_{-0.00}$	1.00 $^{+0.00}_{-0.00}$
dial-turn-v3	0.93 $^{+0.05}_{-0.15}$	0.81 $^{+0.13}_{-0.21}$	0.88 $^{+0.10}_{-0.16}$	0.88 $^{+0.06}_{-0.18}$	0.88 $^{+0.10}_{-0.13}$
door-close-v3	1.00 $^{+0.00}_{-0.00}$	1.00 $^{+0.00}_{-0.00}$	1.00 $^{+0.00}_{-0.01}$	1.00 $^{+0.00}_{-0.00}$	1.00 $^{+0.00}_{-0.00}$
door-lock-v3	1.00 $^{+0.00}_{-0.00}$	1.00 $^{+0.00}_{-0.00}$	1.00 $^{+0.00}_{-0.02}$	1.00 $^{+0.00}_{-0.00}$	1.00 $^{+0.00}_{-0.00}$
door-open-v3	1.00 $^{+0.00}_{-0.00}$	0.98 $^{+0.02}_{-0.09}$	1.00 $^{+0.00}_{-0.01}$	1.00 $^{+0.00}_{-0.01}$	1.00 $^{+0.00}_{-0.07}$
door-unlock-v3	1.00 $^{+0.00}_{-0.00}$	1.00 $^{+0.00}_{-0.00}$	1.00 $^{+0.00}_{-0.01}$	1.00 $^{+0.00}_{-0.00}$	1.00 $^{+0.00}_{-0.03}$
drawer-close-v3	1.00 $^{+0.00}_{-0.00}$	1.00 $^{+0.00}_{-0.00}$	1.00 $^{+0.00}_{-0.02}$	1.00 $^{+0.00}_{-0.01}$	1.00 $^{+0.00}_{-0.00}$
drawer-open-v3	0.22 $^{+0.50}_{-0.22}$	0.83 $^{+0.17}_{-0.44}$	0.27 $^{+0.45}_{-0.27}$	0.94 $^{+0.06}_{-0.46}$	1.00 $^{+0.00}_{-0.00}$
faucet-close-v3	1.00 $^{+0.00}_{-0.03}$	1.00 $^{+0.00}_{-0.04}$	1.00 $^{+0.00}_{-0.06}$	1.00 $^{+0.00}_{-0.00}$	1.00 $^{+0.00}_{-0.00}$
faucet-open-v3	1.00 $^{+0.00}_{-0.03}$	0.98 $^{+0.02}_{-0.08}$	1.00 $^{+0.00}_{-0.03}$	1.00 $^{+0.00}_{-0.13}$	1.00 $^{+0.00}_{-0.07}$
handle-press-side-v3	1.00 $^{+0.00}_{-0.00}$	1.00 $^{+0.00}_{-0.06}$	1.00 $^{+0.00}_{-0.00}$	1.00 $^{+0.00}_{-0.00}$	1.00 $^{+0.00}_{-0.00}$
handle-press-v3	1.00 $^{+0.00}_{-0.03}$	1.00 $^{+0.00}_{-0.00}$	1.00 $^{+0.00}_{-0.00}$	1.00 $^{+0.00}_{-0.01}$	1.00 $^{+0.00}_{-0.03}$
handle-pull-side-v3	1.00 $^{+0.00}_{-0.03}$	0.92 $^{+0.07}_{-0.13}$	1.00 $^{+0.00}_{-0.17}$	1.00 $^{+0.00}_{-0.03}$	0.98 $^{+0.02}_{-0.03}$
peg-insert-side-v3	0.99 $^{+0.01}_{-0.06}$	0.97 $^{+0.03}_{-0.04}$	1.00 $^{+0.00}_{-0.01}$	0.98 $^{+0.02}_{-0.05}$	0.90 $^{+0.10}_{-0.52}$
plate-slide-back-side-v3	1.00 $^{+0.00}_{-0.00}$	1.00 $^{+0.00}_{-0.00}$	1.00 $^{+0.00}_{-0.00}$	1.00 $^{+0.00}_{-0.00}$	1.00 $^{+0.00}_{-0.03}$
plate-slide-back-v3	1.00 $^{+0.00}_{-0.00}$	0.97 $^{+0.03}_{-0.27}$	1.00 $^{+0.00}_{-0.00}$	1.00 $^{+0.00}_{-0.00}$	1.00 $^{+0.00}_{-0.00}$
plate-slide-side-v3	0.90 $^{+0.06}_{-0.07}$	0.82 $^{+0.08}_{-0.07}$	0.82 $^{+0.07}_{-0.08}$	0.95 $^{+0.05}_{-0.12}$	0.82 $^{+0.10}_{-0.17}$
plate-slide-v3	1.00 $^{+0.00}_{-0.00}$	1.00 $^{+0.00}_{-0.01}$	1.00 $^{+0.00}_{-0.00}$	1.00 $^{+0.00}_{-0.00}$	1.00 $^{+0.00}_{-0.00}$
push-wall-v3	0.98 $^{+0.02}_{-0.04}$	0.95 $^{+0.04}_{-0.12}$	1.00 $^{+0.00}_{-0.01}$	0.96 $^{+0.04}_{-0.24}$	0.78 $^{+0.05}_{-0.10}$
reach-v3	1.00 $^{+0.00}_{-0.01}$	1.00 $^{+0.00}_{-0.02}$	1.00 $^{+0.00}_{-0.00}$	1.00 $^{+0.00}_{-0.16}$	0.95 $^{+0.05}_{-0.05}$
reach-wall-v3	0.98 $^{+0.02}_{-0.03}$	0.98 $^{+0.02}_{-0.04}$	0.98 $^{+0.02}_{-0.08}$	0.96 $^{+0.03}_{-0.20}$	0.97 $^{+0.03}_{-0.05}$
sweep-into-v3	0.96 $^{+0.04}_{-0.09}$	0.96 $^{+0.04}_{-0.06}$	1.00 $^{+0.00}_{-0.05}$	0.99 $^{+0.01}_{-0.09}$	0.75 $^{+0.15}_{-0.17}$
sweep-v3	0.97 $^{+0.03}_{-0.10}$	1.00 $^{+0.00}_{-0.05}$	1.00 $^{+0.00}_{-0.01}$	1.00 $^{+0.00}_{-0.00}$	0.88 $^{+0.08}_{-0.15}$
window-close-v3	1.00 $^{+0.00}_{-0.00}$	1.00 $^{+0.00}_{-0.02}$	1.00 $^{+0.00}_{-0.00}$	1.00 $^{+0.00}_{-0.00}$	1.00 $^{+0.00}_{-0.00}$
window-open-v3	1.00 $^{+0.00}_{-0.01}$	0.99 $^{+0.01}_{-0.03}$	1.00 $^{+0.00}_{-0.01}$	1.00 $^{+0.00}_{-0.00}$	1.00 $^{+0.00}_{-0.00}$

Table 5: IQM success rate with 95% bootstrapped confidence interval across the 30 easiest Metaworld tasks, comparing SEAR variants with a non-chunking SimbaV2 baseline. On nearly all tasks, the policies achieve close to a perfect success rate. We therefore exclude these environments from our other experiments.

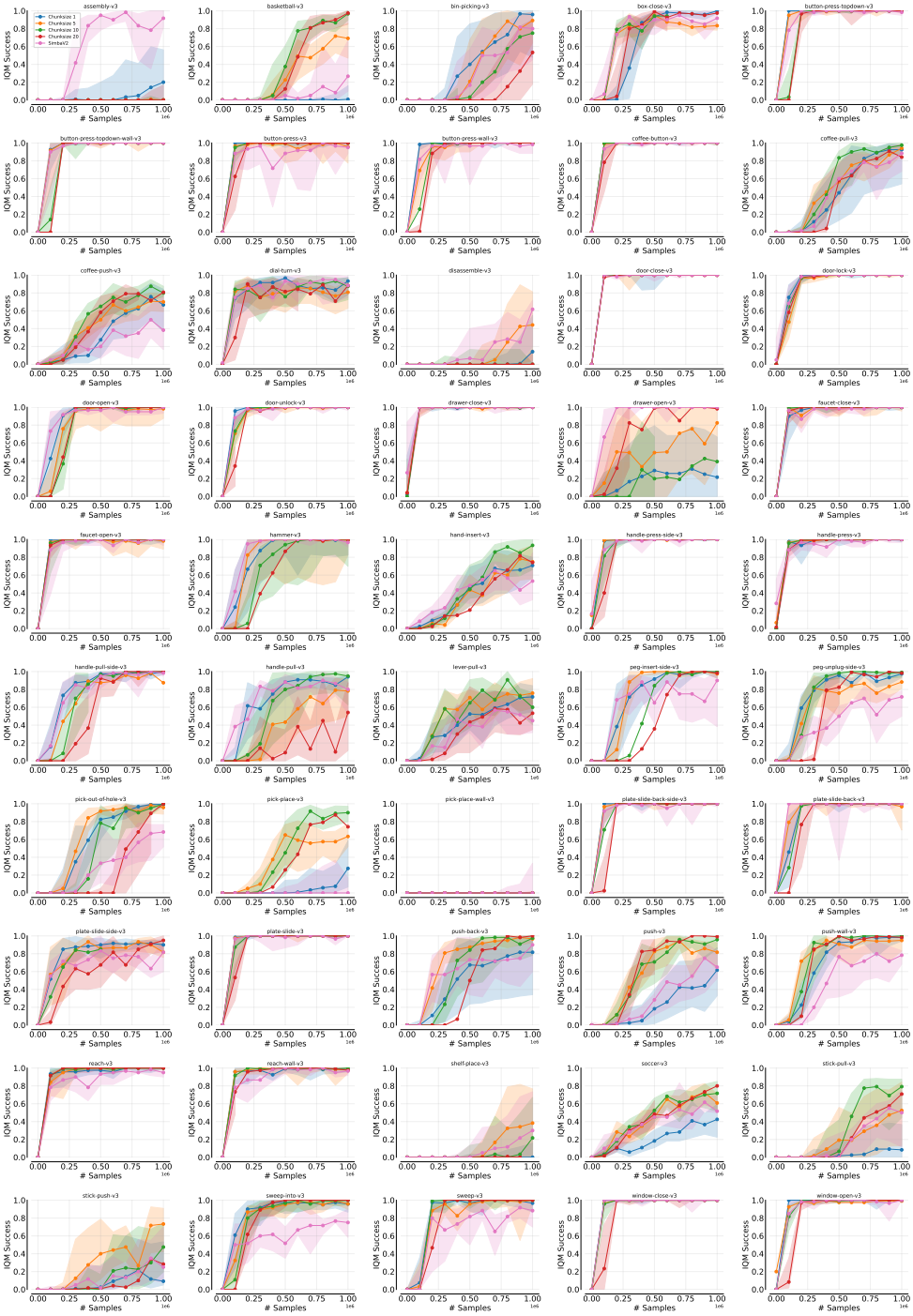


Figure 9: Learning curves for all 50 Metaworld Environments.

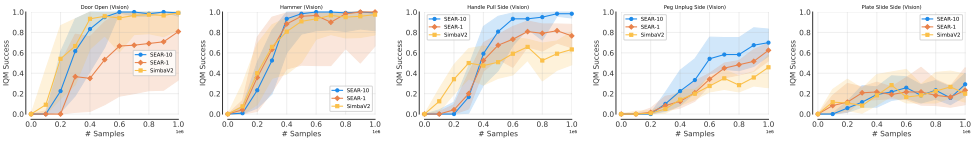


Figure 10: Learning curves for the 5 Metaworld Environments used in the visual RL experiments.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims made in the abstract and introduction are supported by the experimental results in in Section 5. While actor state coverage is hard to quantify and changes throughout training, we quanitfy its effect on receding horizon effectiveness in Figure 2d.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The limitations of the method are discussed in Section 6.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: The paper does not present theoretical results.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Implementations of the presented methods including source code, dependency versions and configuration files are provided. All software used, especially the benchmarks are freely accessible.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Implementations of the presented methods including source code, dependency versions and configuration files are provided. All software used, especially the benchmarks are freely accessible.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The experimental setup is described in Section 5, with hyperparameters and further details provided in Appendix D. Furthermore, the configuration files and implementations needed to reproduce our results are part of the supplementary material.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We follow best practices and report IQM [1] success with 95% bootstrap confidence intervals computed over 500 samples.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We discuss the compute resources needed to reproduce our results in Appendix D.1.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We use neither human subjects nor do we publish any datasets. SEAR has no societal impacts that sets it apart from other RL methods, e.g. possible replacement of jobs.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: Our research concerns itself with algorithmic improvements for RL which have no immediate impact on society.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: We release no models or data that have a risk of misuse.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All assets used (benchmarks, algorithms, implementations) are properly cited.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The code released contains suitable documentation.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: We neither use crowdsourcing nor human subjects in this work.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: We neither use crowdsourcing nor human subjects in this work.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: Our method does not involve LLMs.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.