Scaling the Horizon of Contrastive Reinforcement Learning

David Yan[1], Brandon Cho[1], Andrew Tu[1], Vedant Badoni[1], Zechang Yang[1], Raj Ghugare[1], Benjamin Eysenbach[1]

[1]**Princeton University**

Accepted at the Workshop on Reinforcement Learning Beyond Rewards (RLBrew), RLC 2026.

Abstract

Contrastive reinforcement learning (CRL) is a popular self-supervised RL algorithm that learns to reach goals without using any external rewards. However, existing results on BuilderBench (Ghugare et al., 2025) tasks show that CRL struggles when scaled to open-ended environments with increasing degrees of freedom. We first propose modified BuilderBench environments that run entirely on the GPU, speeding up training by up to $16\times$. We then perform a large-scale study of CRL failure modes and design choices on BuilderBench. We find that horizon reduction methods are effective at improving performance for both PPO and CRL, and that horizon reduction implicitly regularizes CRL's contrastive objective. Using these insights, we introduce StableCRL, a tuned implementation of CRL that incorporates regularization techniques such as in-trajectory negative sampling. On both short- and long-horizon tasks, StableCRL substantially improves over baseline CRL and achieves success rates comparable to a dense-reward PPO reference on several settings.

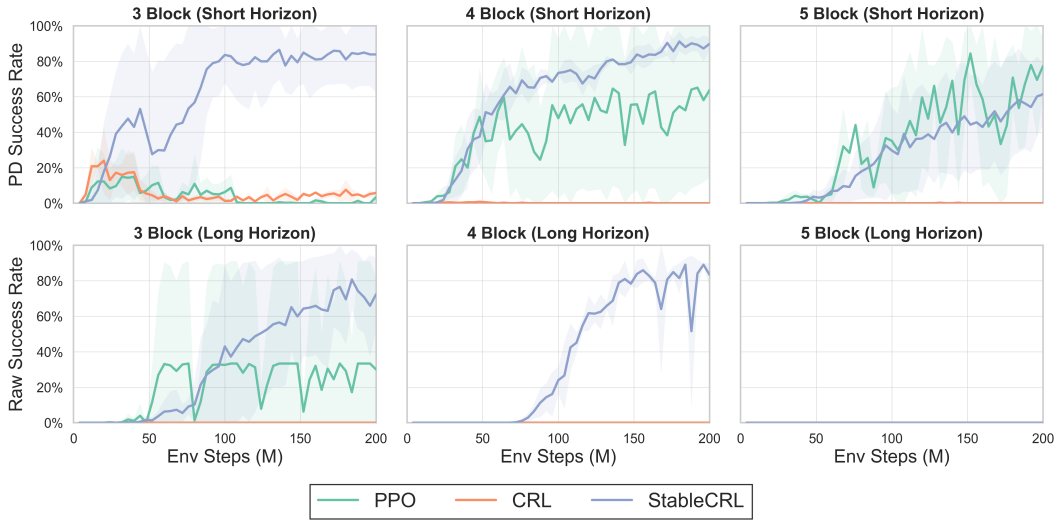

Figure 1: **StableCRL solves long-horizon, open-ended tasks**. We plot the success rate for PPO, CRL, and StableCRL on both short- and long-horizon versions of block-stacking tasks. Short horizon and long horizon refers to PD-5 and the raw action space, respectively. StableCRL is significantly stronger than CRL and solves long-horizon tasks without explicit horizon reduction techniques.

1 Introduction

Contrastive reinforcement learning (CRL) (Eysenbach et al., 2022) is a promising alternative to reward-based RL. By reformulating value estimation as a contrastive learning problem, CRL circumvents the fundamental problem of dense reward specification. Empirically, CRL has been demonstrated to exhibit highly desirable properties: it implicitly performs exploration (Liu et al., 2025), scales well to large, deep network architectures (Wang et al., 2025), and is effective for combinatorial reasoning problems (Ziarko et al., 2025). These properties position CRL as a strong candidate for general-purpose goal-conditioned RL.

However, recent results from the BuilderBench benchmark (Ghugare et al., 2025) suggest that CRL struggles to scale to environments with increasing degrees of freedom. On simple tasks such as block stacking, CRL lags significantly behind reward-based methods such as PPO (Schulman et al., 2017). As the number of cubes in an environment increases, CRL's performance falls dramatically, failing to solve any tasks requiring 3 or more cubes. The goal of this work is simple: *can we push CRL to solve difficult, open-ended tasks from BuilderBench*?

Improving CRL first requires understanding its limits. Currently, there exists no extensive empirical investigation of failure modes and design choices for CRL on open-ended environments such as BuilderBench. To perform such a study, we first re-implement simplified BuilderBench environments with a GPU-accelerated backend, yielding training speedups of up to $16\times$ (Figure 2). Our new environments enable quicker iteration, longer training schedules, and larger model sizes.

Our resulting analysis of BuilderBench and CRL yields a number of important findings. We identify the curse of horizon as a major difficulty of BuilderBench, and find that reducing horizon length is an effective way to increase performance of both PPO and CRL (Figure 3). Applying scaling techniques from Wang et al. (2025) does not improve performance (Figure 5), suggesting that increasing model capacity alone cannot solve long-horizon tasks. Instead, we observe that horizon reduction *implicitly* regularizes the CRL critic's contrastive objective by making it more difficult (Figure 4).

We then use these findings to develop StableCRL, a tuned implementation of CRL that combines a number of existing modifications to CRL. We incorporate in-trajectory negative sampling (Ziarko et al., 2025) and tune the entropy cost to achieve a similar regularization effect to horizon reduction. Using just these two changes, StableCRL substantially outperforms the baseline implementation of CRL and is able to rival on-policy methods such as PPO, even without explicit horizon reduction techniques (Figure 1). Additionally, StableCRL is more scalable than CRL. On long-horizon 5-block stacking, scaling StableCRL enables partial success while no other method makes progress (Figure 8). Our main contributions are the following:

1. We introduce simplified, GPU-accelerated BuilderBench environments to enable efficient experimentation and training.
2. We find that horizon reduction methods improve performance, and observe that horizon reduction acts as implicit regularization for the CRL critic. Finally, we also demonstrate that increasing CRL network size alone is insufficient for scaling to longer-horizon tasks.
3. We introduce StableCRL, a regularized implementation of CRL on BuilderBench that achieves significantly better performance than the original implementation. StableCRL can solve long-horizon 4-block stacking, while PPO and CRL make no progress.

2 Related Work

RL Benchmarks. Standard benchmarks in RL typically task agents to maximize a single reward function (Mnih et al., 2013; Brockman et al., 2016; Tunyasuvunakool et al., 2020). Many recent benchmarks increase the complexity of potential behaviors by enabling compositional and open-ended interaction (Matthews et al., 2024; Küttler et al., 2020). In robotics, benchmarks such as (Park et al., 2025a; Bortkiewicz et al., 2025; Yu et al., 2021) study increasingly diverse behaviors by en-

abling diverse goal-conditioned tasks with large state spaces. Prior work has argued that these benchmarks are still limited in the diversity of potential behaviors and independent degrees of freedom (Ghugare et al., 2025). To tackle this problem, BuilderBench (Ghugare et al., 2025) proposed a benchmark that tasks agents with building various structures using building blocks. Thus, both the task horizon and the number of degrees of freedom increase with the number of blocks in the environment. The initial BuilderBench simulator was not end-to-end hardware-accelerated, limiting its scalability to longer-horizon tasks with more cubes. Our work builds directly on BuilderBench by introducing a simplified but end-to-end, hardware-accelerated variation that significantly boosts scalability (see Fig. 2).

Goal-conditioned RL. Goal-conditioned reinforcement learning (GCRL) studies agents that learn multi-task policies conditioned on goal states (Kaelbling, 1993). The primary challenge in GCRL is exploration under sparse rewards. To get dense feedback from all trajectories, Andrychowicz et al. (2018) uses off-policy trajectories to train on goals that were actually reached. More recently, self-supervised approaches (Eysenbach et al., 2022; Ghosh et al., 2020; Eysenbach et al., 2023) have removed an explicit dependence on a reward function by using purely supervised learning losses like maximum likelihood (Ghosh et al., 2020; Oh et al., 2018) or contrastive learning (Eysenbach et al., 2022).

The curse of horizon in RL refers to the problem of assigning credit to actions that cause a delayed reward (Sutton & Barto, 2018). Delayed rewards increase the variance of estimated returns in Monte Carlo estimation (Hung et al., 2018; Arjona-Medina et al., 2019) and increase the bias in TD learning (Schulman et al., 2018; Sutton, 1988). Park et al. (2025b) observe that offline reinforcement learning struggles to scale to long-horizon tasks, even with unlimited data. We similarly find that horizon reduction methods are effective and improve results on BuilderBench, suggesting that the curse of horizon remains a major challenge even for online, off-policy learning.

Contrastive Representations for Temporal Reasoning (CRTR). Ziarko et al. (2025) introduce in-trajectory negative sampling as a technique to regularize CRL when each trajectory has unique "context" in the state that makes it trivial to classify states from different trajectories. However, Ziarko et al. primarily study the offline setting where the representations are learned from pre-existing trajectories. Moreover, CRTR primarily considers discrete state spaces, so the critic is not conditioned on an action. In contrast, we apply their regularization techniques to online CRL for continuous manipulation tasks, which requires learning both an actor and a critic.

Scaling contrastive reinforcement learning. Wang et al. (2025) suggests that depth can unlock new capabilities in self-supervised RL, especially when paired with dense contrastive supervision, residual architectures, and layer normalization. Our results suggest that scaling network capacity alone is not enough to overcome the curse of horizon (Figure 5). However, once the learning objective is regularized, scaling network size enables progress on difficult, long-horizon tasks.

3 A fast and open-ended BuilderBench environment

Performing a large-scale study of failure modes and design choices requires an efficient environment. However, the existing BuilderBench environments scale poorly as scene complexity increases, with a 6-block environment running at just 8000 steps per second (SPS), as demonstrated in Figure 2. As such, we first re-implement the BuilderBench environment backend using MJX (Google DeepMind, 2026) and MJWarp (Google DeepMind and NVIDIA, 2026). The original BuilderBench implementation uses JAX (Bradbury et al., 2018) to accelerate model training, but performs environment steps on CPU using native MuJoCo (Todorov et al., 2012). This separation creates communication overhead between the CPU and GPU on every simulation step. In contrast, our MJX implementation enables both simulation and model training entirely on GPU, enabling significantly faster training. On simple environments such as 3-block stacking, we find up to a $\sim 16\times$ (265k vs 16.5k SPS) speedup on PPO training. Even on larger environments such as 8-block stacking, our implementation is $\sim 8.7\times$ (55k vs 6.3k SPS) faster when training CRL. We run all experiments on a single NVIDIA RTX 3090 GPU.

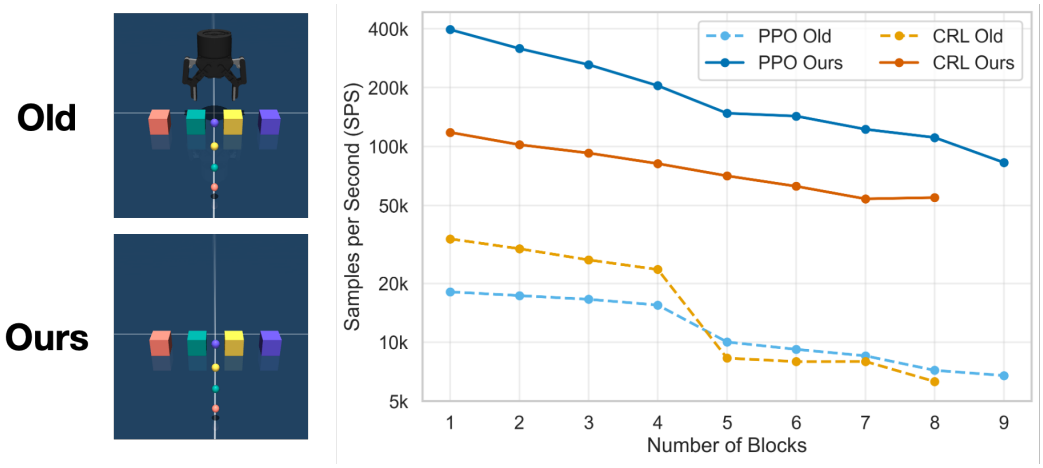

Figure 2: **Environment Comparisons**. Left: Our environment removes the robot arm, leading to more effective degrees of freedom as the number of blocks increases. Right: We train PPO and CRL across environments of varying complexity and measure steps-per-second (SPS) during training. Our simplified, GPU-accelerated environment is significantly faster than the default BuilderBench environment. CRL runs out of GPU memory on both 9-block environments.

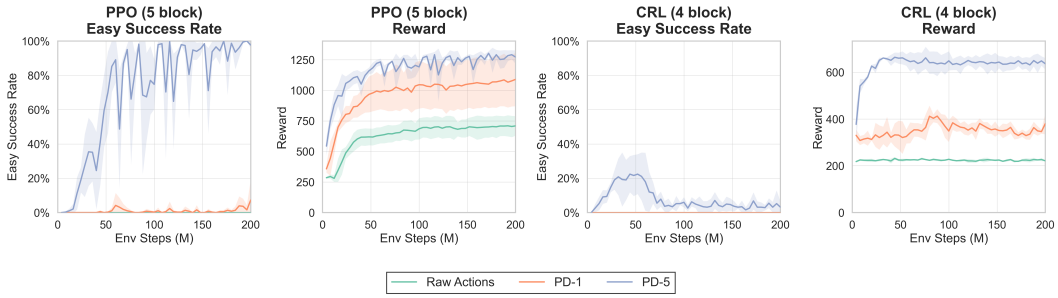

Figure 3: **BuilderBench is a horizon problem**. Easy success rate and reward for PPO and CRL under the PD-1 action space transformation and additional PD-5 horizon reduction described in Section 4.1. Reducing the effective horizon increases performance for both PPO and CRL.

We also introduce a simplified but more open-ended version of the BuilderBench block stacking task. Rather than controlling a robot arm, the agent must vertically stack n cubes by directly controlling the blocks (Figure 2). In this environment, the effective decision space scales with the number of blocks because the agent can move any block at any timestep. In contrast, the robot gripper in the original environment imposes a local constraint on the actions of the agent and limits the effective decision space. Our environment remains difficult even without the robot arm; baseline CRL cannot solve any stacking tasks with ≥ 3 blocks, while PPO achieves no success on long-horizon versions of tasks with ≥ 4 blocks (Figure 1).

4 Experiments

To study the effect of horizon length, we modify the action space of the environment. In the default action space, which we denote Raw Actions, the model directly outputs a raw force vector (x, y, z, ψ, ω) to move the block, where ψ is a yaw control and ω is a continuous variable that is discretized to select the block to be controlled.

In our modified action space, which we denote as PD-n, the model outputs a target block pose (x, y, z, ψ, ω), where ω is similarly used for block selection. A proportional-derivative controller

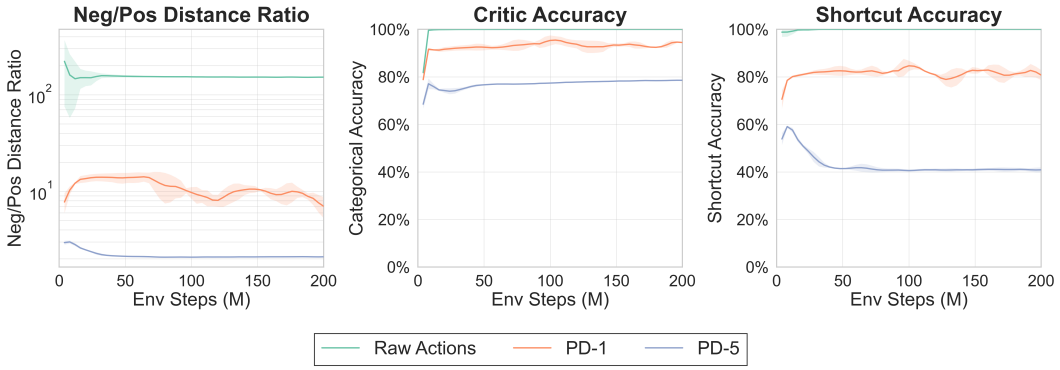

Figure 4: **Horizon reduction implicitly regularizes CRL**. In the long-horizon setting (raw actions), the contrastive problem is trivial to solve without action knowledge, as evidenced by a high shortcut accuracy and distance ratio. Re-parameterizing the action space (PD-1) and reducing the horizon (PD-5) lead to increased reward (Figure 3) but decreased critic accuracy. Horizon reduction therefore has a regularizing effect on the critic.

outputs low-level actions towards the target for n environment steps. Increasing n can be seen as a form of *policy horizon reduction* through a hierarchical policy, as described in Park et al. (2025b).

In order to obtain nonzero success rates, we perform all experiments using PD-5, unless otherwise specified. Similar to Ghugare et al. (2025), we compute a dense reward metric using a distance computed via Hungarian matching. PPO is provided this reward during training, while CRL receives only the goal state. We therefore include PPO as a task-performance reference rather than as a supervision-matched algorithmic comparison. A rollout is considered successful if all cubes are within 0.02 m of their goal state. We also consider an "easy" success criterion with a larger threshold of 0.05 m to obtain a more informative signal in difficult environments. We train all models for 2×10^8 steps. During each evaluation, we report the mean dense episode reward, the success rate, and the easy success rate over 128 evaluation episodes. For each experimental setting, we plot the mean and ± 1 std over three seeds, unless otherwise specified.

4.1 Characterizing Failure Modes on BuilderBench

BuilderBench is a horizon problem. We hypothesize that a major difficulty of BuilderBench is the long effective horizon of its tasks. We evaluate the following three settings: raw actions (long horizon), PD-1, and PD-5, and compare their easy success rate and mean return (Figure 3). Switching from raw actions to the PD-1 action space improves attained reward for both PPO and CRL, suggesting that simply reformatting the action space is helpful. Similarly, reducing the effective horizon by changing the controller from PD-1 to PD-5 results in a significant increase in success rate for both PPO and CRL. Our results suggest that horizon length is a major difficulty for BuilderBench. While existing work has proposed novel learning algorithms structured around horizon reduction (Sutton et al., 1999; Park et al., 2025b), we demonstrate that CRL can be modified to handle long-horizon tasks without explicit horizon reduction in Section 4.3.

Scaling alone is ineffective. Why does CRL fail on long horizons? One hypothesis is that the default CRL implementation in BuilderBench simply has insufficient capacity. For instance, Wang et al. (2025) finds significant performance boosts by increasing the depth of networks; in contrast, the default CRL implementation uses actor and critic networks with depth 4. We re-implement the recurrent block structure described in Wang et al. and train networks of varying size on a 4-block stacking task using a PD-5 controller (Figure 5). However, we find limited benefits from scaling. We first experiment with trading off width for depth, while maintaining an equal parameter count. Increasing the depth to 16 and decreasing the width to 256 slightly decreases the achieved reward. We also try increasing the depth to 32 while maintaining a width of 1024, which similarly does not

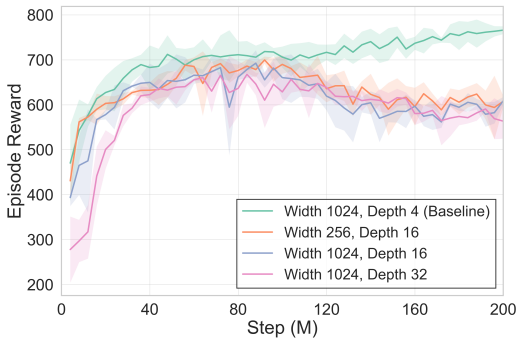

Figure 5: **Naive scaling of CRL fails.** We attempt to scale up the depth of CRL networks. We try both trading off width for depth and scaling up depth while maintaining width. Neither improves the resulting performance of the policy.

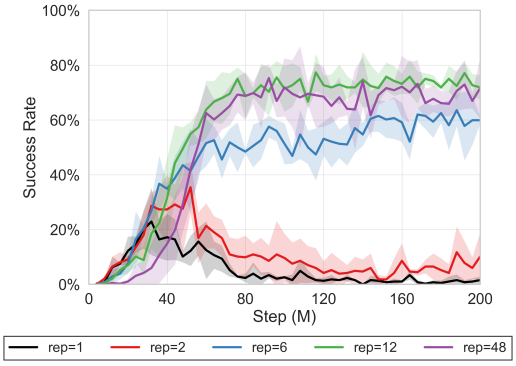

Figure 6: **Repetition factor ablation**. Easy success rate on 4-cube PD-5 stacking for repetition factors $r \in \{1, 2, 6, 12, 48\}$, with PPO (dashed) for reference. Performance improvements saturate at a repetition factor of 12.

improve reward. These negative results suggest that low network capacity alone is an insufficient explanation for CRL's poor performance on BuilderBench.

Horizon reduction as regularization. Our negative results on scaling suggest that increasing horizon length may impact the optimization objective. To examine this hypothesis, we examine several metrics during training, such as distance ratio, critic accuracy, and shortcut accuracy. The distance ratio is the mean negative example distance divided by the mean positive example distance, so larger values indicate that positives are geometrically closer than negatives. Critic accuracy measures the percentage of rows where the critic correctly predicts the positive goal as the most likely state. Finally, shortcut accuracy measures the percent of rows where a trivial Euclidean distance baseline correctly predicts the positive goal as the most likely state.

Both re-parameterizing the action space and horizon reduction improve performance by implicitly regularizing the CRL critic. In the long-horizon setting, the critic overfits to a trivial classification problem, as indicated by an extremely large distance ratio and nearly 100% shortcut accuracy (Figure 4). One possible explanation is that longer horizons cause the state space to change relatively slowly, which means many positive goal samples will be highly similar to the initial anchor state. Re-parameterizing the action space reduces the final distance ratio from ~ 150 to ~ 9, and decreasing the effective horizon reduces it to ~ 2. We observe a similar decrease in shortcut accuracy. By making the contrastive problem more difficult, the critic is forced to learn a more useful representation for control beyond a simple distance metric.

If action parameterization and horizon reduction improve performance by regularizing the contrastive objective, one promising direction is to examine preexisting regularization techniques for CRL, which we explore in Section 4.2. Notably, the observed overfitting behavior closely resembles the problem of per-trajectory context that CRTR (Ziarko et al., 2025) is designed to solve. Although we have no fixed context between trajectories, slowly evolving states in the long-horizon setting might constitute "implicit" context, similar to the Rubik's Cube environment studied in CRTR.

4.2 Stabilizing Contrastive RL

In-trajectory negatives are effective. We apply the in-trajectory negative sampling technique described in Ziarko et al. (2025). For a given repetition factor r, we sample r state-goal pairs from each trajectory. We sweep across $r \in \{1, 2, 6, 12, 48\}$ on 4-cube stacking with PD-5 controls (Figure 6). At low repetition factors ($r = 1, 2$), training remains unstable and the model obtains a final success rate of under 20%. Starting at $r = 6$, performance improves substantially and increasing r to 12

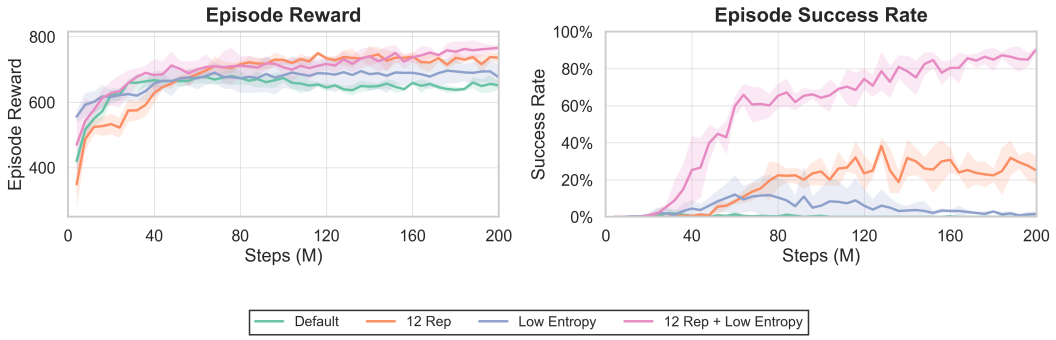

Figure 7: **StableCRL ablations**. We examine the importance of each change in StableCRL on PD-5 4-block stacking. Alone, increasing the repetition factor is the most important change, and combining it with a lower entropy penalty is especially effective.

further improves performance. We observe diminishing returns for $r > 12$, with little improvement between $r = 12$ and $r = 48$. As such, we adopt $r = 12$ as our default setting.

Reducing entropy cost. In CRL, the actor loss contains an entropy bonus $\alpha_t \log \pi_\theta(a_i \mid s_i, g_i)$, where α_t is a hyperparameter controlling the strength of the bonus. Empirically, we observe that BuilderBench's default CRL entropy bonus of 0.1 on the actor keeps the actor's sampled negative log probability high throughout training. We hypothesize that having more random actions harms training because it results in slower state changes. We also note that Eysenbach et al. (2022) and Wang et al. (2025) set the entropy coefficient to zero for state-based environments. Motivated by this precedent, we set the entropy coefficient to 0.01. Figure 7 shows that while decreasing the entropy weight alone is not particularly effective , combining it with in-trajectory negative sampling results in high performance.

4.3 StableCRL enables novel problem solving

StableCRL is competitive with PPO on short- and long-horizon tasks. We denote the combination of in-trajectory negatives and reduced entropy coefficient as StableCRL. StableCRL achieves much higher success rates than PPO on 3-block stacking, higher success on 4-block stacking, and comparable success on the 5-block stacking tasks in the short-horizon setting. The default CRL algorithm, by contrast, has success rates near zero for the short-horizon 4- and 5-block tasks (see Figure 1). These results demonstrate that simply regularizing the contrastive objective can yield strong performance on BuilderBench, even without any explicit horizon reduction methods.

StableCRL is scalable to difficult tasks. Unlike base CRL, which failed to benefit from scaling (Section 4.1), StableCRL scales more effectively when using techniques from Wang et al. (2025). We scale the actor and critic depth to 32 layers with a width of 1024 and denote this setting StableCRL-Scaled. Scaling improves performance on some of the most difficult tasks. In Figure 8, StableCRL-Scaled is the only variant to make non-zero progress on the long-horizon 5-block environment with raw actions, and significantly outperforms base StableCRL on the horizon-reduced PD-5 6-block task. However, scaling network size also leads to instabilities during training, resulting in higher variance and worse average performance than StableCRL on both 4-block stacking tasks and the 5-block short-horizon task. These results support the findings of Wang et al. (2025) that scaling depth and network capacity unlock new capabilities, but suggest that tackling longer-horizon tasks still requires careful regularization of the underlying CRL objective.

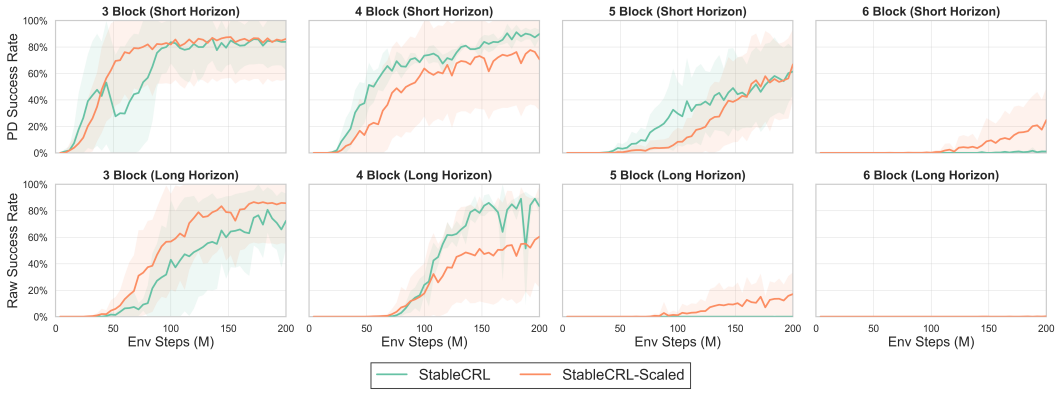

Figure 8: **Scaling StableCRL**. Scaling the critic and actor depth from 4 to 32 layers (StableCRL-Scaled) improves performance on the more difficult long-horizon tasks. However, these gains come at the cost of increased training instability, yielding lower average performance on certain easier tasks. Short horizon and long horizon refer to PD-5 and raw action space, respectively. StableCRL-Scaled runs are averaged over 10 seeds, due to higher observed variance.

4.4 Future Work and Limitations

Improving network scaling for StableCRL. While we have shown that scaling the actor and critic network depth in StableCRL yields positive success rates for more complex tasks, such as the PD-controller 5-block and 6-block tasks presented in Figure 8, we observe high variance in performance across seeds. Future work could investigate methods for reducing training instability for these scaled networks.

Algorithmic improvements for CRL. In-trajectory negative sampling and entropy tuning are simple, hand-crafted methods for increasing the difficulty of the CRL objective. One avenue for future work is developing algorithmic modifications to automatically adjust the difficulty of the objective.

Understanding failure modes for 6 or more blocks. Although StableCRL and its scaled variant perform significantly better on vertical stacking tasks involving 3–6 blocks than baseline CRL, these methods still struggle on the longer-horizon tasks. A promising direction for future work is to understand why this remains a hard limit for StableCRL.

Expanding to other tasks. BuilderBench supports a wide range of structures beyond vertical stacks. Future research can examine whether our regularization recipe can generalize to other structures, which may exhibit new failure modes.

References

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay, 2018. URL https://arxiv.org/abs/1707.01495.

Jose A. Arjona-Medina, Michael Gillhofer, Michael Widrich, Thomas Unterthiner, Johannes Brandstetter, and Sepp Hochreiter. Rudder: Return decomposition for delayed rewards, 2019. URL https://arxiv.org/abs/1806.07857.

Michał Bortkiewicz, Władysław Pałucki, Vivek Myers, Tadeusz Dziarmaga, Tomasz Arczewski, Łukasz Kuciński, and Benjamin Eysenbach. Accelerating goal-conditioned rl algorithms and research, 2025. URL https://arxiv.org/abs/2408.11052.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Yash Katariya, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne,

and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/jax-ml/jax.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016. URL https://arxiv.org/abs/1606.01540.

Benjamin Eysenbach, Tianjun Zhang, Sergey Levine, and Russ R Salakhutdinov. Contrastive learning as goal-conditioned reinforcement learning. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 35603–35620. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/e7663e974c4ee7a2b475a4775201ce1f-Paper-Conference.pdf.

Benjamin Eysenbach, Soumith Udatha, Sergey Levine, and Ruslan Salakhutdinov. Imitating past successes can be very suboptimal, 2023. URL https://arxiv.org/abs/2206.03378.

Dibya Ghosh, Abhishek Gupta, Ashwin Reddy, Justin Fu, Coline Devin, Benjamin Eysenbach, and Sergey Levine. Learning to reach goals via iterated supervised learning, 2020. URL https://arxiv.org/abs/1912.06088.

Raj Ghugare, Roger Creus Castanyer, Catherine Ji, Kathryn Wantlin, Jin Schofield, Karthik Narasimhan, and Benjamin Eysenbach. BuilderBench: The building blocks of intelligent agents, 2025. https://doi.org/10.48550/arXiv.2510.06288.

Google DeepMind. MuJoCo XLA (MJX). https://mujoco.readthedocs.io/en/latest/mjx.html, 2026.

Google DeepMind and NVIDIA. MuJoCo Warp (MJWarp). https://github.com/google-deepmind/mujoco_warp, 2026.

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, 2018. URL https://arxiv.org/abs/1810.06721.

Leslie Pack Kaelbling. Learning to achieve goals. In *International Joint Conference on Artificial Intelligence*, 1993. URL https://api.semanticscholar.org/CorpusID:5538688.

Heinrich Küttler, Nantas Nardelli, Alexander H. Miller, Roberta Raileanu, Marco Selvatici, Edward Grefenstette, and Tim Rocktäschel. The nethack learning environment, 2020. URL https://arxiv.org/abs/2006.13760.

Grace Liu, Michael Tang, and Benjamin Eysenbach. A single goal is all you need: Skills and exploration emerge from contrastive RL without rewards, demonstrations, or subgoals. In *International Conference on Learning Representations*, 2025. https://doi.org/10.48550/arXiv.2408.05804.

Michael Matthews, Michael Beukman, Benjamin Ellis, Mikayel Samvelyan, Matthew Jackson, Samuel Coward, and Jakob Foerster. Craftax: A lightning-fast benchmark for open-ended reinforcement learning, 2024. URL https://arxiv.org/abs/2402.16801.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning, 2013. URL https://arxiv.org/abs/1312.5602.

Junhyuk Oh, Yijie Guo, Satinder Singh, and Honglak Lee. Self-imitation learning, 2018. URL https://arxiv.org/abs/1806.05635.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. Ogbench: Benchmarking offline goal-conditioned rl, 2025a. URL https://arxiv.org/abs/2410.20092.

Seohong Park, Kevin Frans, Deepinder Mann, Benjamin Eysenbach, Aviral Kumar, and Sergey Levine. Horizon reduction makes rl scalable, 2025b. URL https://arxiv.org/abs/2506.04168.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation, 2018. URL https://arxiv.org/abs/1506.02438.

Richard S. Sutton. Learning to predict by the methods of temporal differences. *Mach. Learn.*, 3 (1):9–44, August 1988. ISSN 0885-6125. DOI: 10.1023/A:1022633531479. URL https://doi.org/10.1023/A:1022633531479.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018. URL http://incompleteideas.net/book/the-book-2nd.html.

Richard S. Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: a framework for temporal abstraction in reinforcement learning. *Artif. Intell.*, 112(1–2):181–211, August 1999. ISSN 0004-3702. DOI: 10.1016/S0004-3702(99)00052-1. URL https://doi.org/10.1016/S0004-3702(99)00052-1.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE, 2012. DOI: 10.1109/IROS.2012.6386109.

Saran Tunyasuvunakool, Alistair Muldal, Yotam Doron, Siqi Liu, Steven Bohez, Josh Merel, Tom Erez, Timothy Lillicrap, Nicolas Heess, and Yuval Tassa. dm_control: Software and tasks for continuous control. *Software Impacts*, 6:100022, November 2020. ISSN 2665-9638. DOI: 10.1016/j.simpa.2020.100022. URL http://dx.doi.org/10.1016/j.simpa.2020.100022.

Kevin Wang, Ishaan Javali, Michał Bortkiewicz, Tomasz Trzciński, and Benjamin Eysenbach. 1000 layer networks for self-supervised RL: Scaling depth can enable new goal-reaching capabilities. In *Advances in Neural Information Processing Systems*, 2025. https://doi.org/10.48550/arXiv.2503.14858.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Avnish Narayan, Hayden Shively, Adithya Bellathur, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning, 2021. URL https://arxiv.org/abs/1910.10897.

Alicja Ziarko, Michal Bortkiewicz, Michal Zawalski, Benjamin Eysenbach, and Piotr Milos. Contrastive representations for temporal reasoning, 2025. URL https://arxiv.org/abs/2508.13113.