Test-time Offline Reinforcement Learning on Goal-related Experience

Marco Bagatella [*1,2] **Mert Albaba** [*1,2] **Jonas Hübotter** [1] **Georg Martius** [2,3] **Andreas Krause** [1]

Abstract

Foundation models compress a large amount of information in a single, large neural network, which can then be queried for individual tasks. There are strong parallels between this widespread framework and offline goal-conditioned reinforcement learning algorithms: a universal value function is trained on a large number of goals, and the policy is evaluated on a single goal in each test episode. Extensive research in foundation models has shown that performance can be substantially improved through test-time training, specializing the model to the current goal. We find similarly that test-time offline reinforcement learning on experience related to the test goal can lead to substantially better policies at minimal compute costs. We propose a novel self-supervised data selection criterion, which selects transitions from an offline dataset according to their relevance to the current state and quality with respect to the evaluation goal. We demonstrate across a wide range of high-dimensional loco-navigation and manipulation tasks that fine-tuning a policy on the selected data for a few gradient steps leads to significant performance gains over standard offline pre-training.

1 Introduction

Machine learning models are largely static: after a computationally expensive training phase, inference traditionally involves a single forward pass (or multiple, in the case of autoregressive models), without any further parameter updates. This framework is widely adopted across modalities and domains, from early works on image classification (LeCun et al., 1998; He et al., 2016) to many modern vision/language models (Brown et al., 2020; Rombach et al., 2022). However, perfectly imitating training data with a neural network is challenging, and predictions of neural networks are often noisy and imprecise. As a consequence, base models are often specialized to down-stream tasks through fine-tuning (Hu et al., 2022; Kim et al., 2022; Black et al., 2024). More recently, across (self-)supervised vision and language tasks, several works improved performance by specializing the model to an individual task, either through in-context learning or test-time training (e.g., Brown et al., 2020; Sun et al., 2020; Hardt & Sun, 2024; Hübotter et al., 2025). In contrast, in offline reinforcement learning, these approaches have not yet been much explored. While the dynamic conditioning of a learned policy at test-time has been explored in hierarchical methods (Nachum et al., 2018; Eysenbach et al., 2019; Park et al., 2023), the weights of the policy itself remain generally frozen during evaluation.

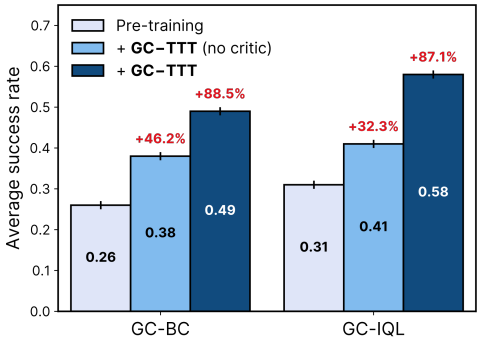

Figure 1. We introduce test-time training in the context of offline goal-conditioned reinforcement learning. The same data used for pre-training is filtered and leveraged to improve the policy locally during evaluation. This results in significant performance gains in standard benchmarks (left) when combined with common offline RL backbones, GC-BC and GC-IQL.

Our work focuses on the test-time training of goal-conditioned policies. The standard pipeline of offline goal-conditioned reinforcement learning involves (1) a (pre-)training phase, in which a policy learns to reach *arbitrary* goals, often through relabeling or self-supervision, and (2) an inference phase, in which the policy is queried to achieve *one* specific goal. We show that **specializing the policy to an individual goal at test-time significantly improves its performance**, without leveraging any information beyond the pre-training dataset and the pre-trained agent.

[*]Equal contribution [1]ETH Zürich, Switzerland [2]Max Planck Institute for Intelligent Systems, Germany [3]University of Tübingen, Germany. Correspondence to: Marco Bagatella <marco.bagatella@inf.ethz.ch>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

We propose ***g**oal-**c**onditioned **t**est-**t**ime **t**raining* (**GC-TTT**), which fine-tunes the base policy at test-time on goal-related experience from the pre-training dataset.[1] GC-TTT selects experience according to a natural notion of relevance and optimality, ensuring that it is (1) related to the agent's current state, and (2) optimal with respect to a bootstrapped value function estimate (i.e., a *critic*). Based on this goal-related experience, GC-TTT efficiently updates the actor through few gradient steps according to standard policy learning objectives. We repeat this process in a receding-horizon fashion to periodically and dynamically adapt the policy to the current trajectory.

We demonstrate how GC-TTT improves performance in standard offline goal-conditioned benchmarks, suggesting that existing methods that learn to achieve arbitrary goals are systematically underfitting with respect to individual goals. We show that GC-TTT can learn from both expert and play-like data, and additionally derive a variant, which does not require a learned critic and retains good performance on expert data. Both variants are agnostic to the backbone RL algorithm. Within these settings, we ablate the frequency of test-time training, and further investigate the compute allocation at test-time, comparing the cost of test-time training against increased model sizes.

We thus make the following contributions:

- We propose a test-time training framework for goal-conditioned policies.
- We develop GC-TTT, a practical algorithm for dynamically training on goal-related experience during evaluation.
- We demonstrate significant performance gains on standard benchmarks when applying goal-conditioned test-time training on top of existing algorithms.
- We demonstrate that GC-TTT significantly outperforms existing algorithms even when inference FLOPs are matched by scaling the network sizes of baselines.

We discuss related work in Appendix A and provide background on offline RL in Appendix B.

2 Goal-conditioned Test-time Training

We propose to fine-tune the policy dynamically during evaluation, leveraging data from the pre-training dataset $\mathcal{D}$, which is "close" to the agent's current state $s \in \mathcal{S}$ and "optimal" for reaching the agent's current goal $g^\star \in \mathcal{G}$. We denote this carefully selected set of relevant and optimal sub-trajectories as $\mathcal{D}(s, g^\star)$. During evaluation, we then dynamically adapt the policy to the current state-goal pair $(s, g^\star)$ by fine-tuning it on uniform samples from $\mathcal{D}(s, g^\star)$ for a few gradient steps, using the following objective:

$$J_{\text{TTT}}(\theta) = -\mathbb{E}_{s' \sim \mathcal{D}(s, g^\star)} \mathcal{L}(s', g^\star; \theta), \tag{1}$$

where we overload $\mathcal{D}$ to represent a uniform distribution over states in the dataset. Here, $\mathcal{L}$ is any standard policy learning loss, such as behavior cloning or off-policy reinforcement learning.[2] While test-time training might use a different loss than pre-training, for simplicity, we use the same loss for TTT as for pre-training throughout. We set the goal for policy fine-tuning deterministically to the evaluation goal $g^\star$, as the policy will only be queried with this goal.

Figure 3 in the appendix illustrates how GC-TTT fine-tunes the pre-trained policy at test-time. In the following, we discuss the two key components of GC-TTT: (1) selecting relevant and optimal experience from the dataset, and (2) fine-tuning the policy dynamically during evaluation.

2.1 *What to train on?* Selecting Relevant and Optimal Experience

The first step of GC-TTT is to select trajectories which are relevant to the current state of the agent and optimal for achieving the target goal. To determine the relevance of sub-trajectories in $\mathcal{D}$ to the agent's current state $s \in \mathcal{S}$, we leverage a notion of temporal distance. In practice, this can be estimated by the learned quasimetric $-V(s, g)$ of a value function estimate (Wang et al., 2023) or by the locally correct distance function d conventionally exposed by the goal-conditioned reward function (Andrychowicz et al., 2017). We consider a sub-trajectory $(s_1, \dots) \in \mathcal{D}$ as related to the current state s if $d(s, s_1) < \epsilon$ for some $\epsilon > 0$, normally also provided by the environment. This results in a filtered set of sub-trajectories of diverse lengths:

$$\textbf{Relevance:} \quad \mathcal{D}_{\text{rel}}(s) = \{(s_1, \dots s_H) \in \mathcal{D} \mid d(s, s_1) < \epsilon\}. \tag{2}$$

The threshold ϵ may be selected adaptively such that $\mathcal{D}_{\text{rel}}(s)$ is of a desired size; however, in our evaluated environments, fixing ϵ to a constant was sufficient. We note that the distance function does not need to be globally accurate, but only locally.

While this operation selects sub-trajectories that are relevant to the agent's current state, not all of them might be useful for reaching the agent's target goal $g^\star \in \mathcal{G}$. We thus further filter the data to include only those sub-trajectories which are most likely to eventually reach $g^\star$. To measure this, we estimate the returns of the sub-trajectories if they were to be extended using the agent's current policy. We adopt an H-step return estimate (Sutton & Barto, 2018) which considers both the rewards along the sub-trajectory and the

[1] While we propose using the pre-training dataset, leveraging privileged or auxiliary data is also possible.

[2] For completeness, we include an overview in Appendix F.

estimated value of its final state:

$$\widehat{V}((s_1, \dots, s_H) \mid g^\star) = \sum_{i=1}^{H-1} \gamma^{i-1} R(s_i, g^\star) + \gamma^{H-1} V(s_H \mid g^\star). \quad (3)$$

In practice, the resulting estimate combines an evaluation of the behavioral policy inducing $(s_1, \dots, s_H)$ with a value estimate of the current policy π_θ. We find that H-step return estimates effectively trade off bias and variance, providing a reliable signal for data selection. These return estimates rely on a value estimate (i.e., a critic), which is a core component across most offline GCRL algorithms. When a critic is not available, we can use simple trajectory returns according to the behavioral policy and the reward function R, as we demonstrate in Section 3. Given the return estimates $\widehat{V}(\tau \mid g^\star)$, we set the scalar C to their q-th percentile among all relevant sub-trajectories in $\mathcal{D}_{\text{rel}}(s)$, and select the most optimal ones:

$$\textbf{Optimality:} \quad \mathcal{D}(s, g^\star) = \{\tau \in \mathcal{D}_{\text{rel}}(s) \mid \widehat{V}(\tau \mid g^\star) \geq C\}. \quad (4)$$

2.2 *When to train?* Receding Horizon Training

It remains to decide when to fine-tune the policy based on the TTT objective from Equation (1). In principle, we can update the policy whenever either the agent's state or goal change. Since we expect neighboring states to lead to similar fine-tuned policies, we opt for a natural receding horizon approach (Morari & Lee, 1999). We describe the full GC-TTT algorithm in Algorithm 1 in the appendix.

Every K steps, we re-initialize the policy to its pre-training weights. Considering the current state s and goal $g^\star$, we then fine-tune the pre-trained policy on relevant and optimal data. We then roll out the fine-tuned policy for K steps, before its weights are once again reset, and the entire process is repeated. Intuitively, each fine-tuning allows the agent to focus on actions to be taken in its immediate future. Crucially, this allows the policy to only focus on parts of its task, instead of trying to solve it all-at-once. Furthermore, this framework allows dynamic trajectory corrections during each rollout: if the agent strays away from the optimal trajectory, GC-TTT can select helpful data to correct the direction towards the final goal. From this perspective, there are clear parallels between this high-level routine, and model predictive control (MPC, Rawlings et al., 2017), though importantly, our approach does not require a model. We remark that the update rule of GC-TTT may also be applied in different ways than we present here. For instance, it is possible to just fine-tune the policy once, e.g., at the start of the episode or when an error is detected.

3 Experiments

We provide an empirical validation of our contributions spanning four environments and two algorithmic backbones, and identify five main insights in the following.

Environments We rely on a suite of goal-conditioned tasks from OGBench (Park et al., 2025). Namely, we evaluate three loco-navigation tasks of increasing complexity (`pointmaze`, `antmaze` and `humanoidmaze`), spanning from 2 to 21 degrees of freedom.

We evaluate all environments in their `medium` instance, across two datasets of different qualities, namely `navigate` and `stitch`. The former includes full demonstrations for any evaluation state-goal pair, while the latter may only be solved by "stitching" different trajectories together. For ease of interpretation, we refer to them as `expert` and `play`, respectively. We additionally consider one manipulation task, in which a robotic arm is tasked with relocating a cube (`cubesingle`).

Backbones In principle, GC-TTT is applicable across the broad class of value-based offline goal-conditioned algorithms. We select GC-BC (Yang et al., 2022) and GC-IQL (Kostrikov et al., 2022) for evaluation, which form a representative set of common offline RL algorithms.[3]

Insight 1: GC-TTT substantially improves the policy across diverse environments and learning algorithms. To begin with, we evaluate the performance of GC-TTT across the described array of environments and algorithms. We train the backbone algorithm until convergence and report the average performance at 300k, 350k and 400k gradient steps, as in the protocol described by (Park et al., 2025). Performances are computed as the average success rate across four fixed goals in each environment; we report mean and standard error across 3 seeds. We report our results in Table 1 and Figure 6 in the appendix. We observe that GC-TTT improves the performance of the backbone for the majority of algorithm-environment combinations, and does not impact it negatively in the remaining ones. Interestingly, test-time training is capable of reliably solving `pointmaze` with simple techniques (i.e., GC-BC). This suggests that standard approaches for offline goal-conditioned RL might systematically underfit with respect to each specific goal, as a few gradient steps are sufficient to significantly improve their policies. This sheds some light on one of the open problems discussed in Park et al. (2025). Furthermore, as the environment complexity increases (e.g., `antmaze` or `humanoidmaze`), the improvements induced by GC-TTT remain significant; and `cubesingle` confirms that this trend holds in settings with fundamentally different dynamics.

[3] See Appendix E for a more extensive discussion.

	pointmaze		antmaze		humanoidmaze		cubesingle	**avg.**
	expert	play	expert	play	expert	play		
GC-BC	0.09 (0.01)	0.51 (0.02)	0.32 (0.00)	**0.52** (0.03)	0.08 (0.00)	0.28 (0.05)	0.03 (0.00)	0.26 (0.01)
GC-BC + TTT (no critic)	0.70 (0.01)	–	**0.48** (0.03)	–	0.13 (0.01)	–	–	0.38 (0.01)
GC-BC + TTT	**0.86** (0.00)	**0.79** (0.00)	**0.44** (0.01)	**0.51** (0.03)	**0.18** (0.03)	**0.53** (0.01)	**0.10** (0.01)	**0.49** (0.01)
GC-IQL	0.16 (0.03)	0.31 (0.07)	0.64 (0.01)	0.36 (0.04)	0.08 (0.02)	0.07 (0.02)	0.53 (0.01)	0.31 (0.01)
GC-IQL + TTT (no critic)	0.73 (0.01)	–	**0.73** (0.01)	–	**0.13** (0.02)	–	–	0.41 (0.01)
GC-IQL + TTT	**0.84** (0.01)	**0.84** (0.03)	0.67 (0.02)	**0.72** (0.03)	**0.15** (0.01)	**0.28** (0.04)	**0.59** (0.02)	**0.58** (0.01)

Table 1. Success rates of GC-TTT and its critic-free variant across loco-navigation and manipulation, on top of GC-BC and GC-IQL. Numbers in parentheses are standard errors across 3 seeds. **Bold** numbers denote results that are within the standard error of the best. Underlined numbers denote whether TTT outperforms pre-training.

Insight 2: GC-TTT can be applied without value estimates if expert data is available. We now turn our attention to a critic-free variant of GC-TTT. This algorithm replaces the H-step return estimate (cf. Equation (3)) with the trajectory returns (i.e., a discounted sum of rewards along the trajectory). As such, this variant does not require additionally training a critic network (and thus combines seamlessly with, e.g., BC). However, this critic-free variant cannot infer optimality from trajectories that do not reach the target goal, and is therefore limited to expert data. As shown in Table 1, on such tasks with expert data, the critic-free variant retains much of the effectiveness of GC-TTT. In contrast, in play tasks, all relevant sub-trajectories are likely to achieve the same trajectory return of 0.

Insight 3: Selecting both relevant and optimal data is necessary. A core component of GC-TTT is the selection of *relevant* and *optimal* data from the offline dataset (cf. Section 2.1). We ablate this design choice in Figure 2 (left) in the appendix, where we report the average success rates with GC-IQL as backbone in the `pointmaze/antmaze` play environments. We observe that selecting random data from the dataset is not effective, as the global objective of the backbone algorithm has already converged. Selecting relevant but suboptimal data does not improve performance. Selecting optimal data that may be irrelevant to the agent's current state yields a slight increase in success rate. We attribute this to the relatively small size of the environments, which means that by chance some selected trajectories might also be relevant. Remarkably, GC-TTT leads to a substantial performance gain by combining both relevance to the agent's current state and optimality for the agent's goal. We additionally plot data selected by GC-TTT over the course of an evaluation episode in Figure 4 in the appendix.

Insight 4: The frequency of test-time training should adapt depending on the difficulty of the environment. We discuss the compute cost of GC-TTT in Appendix C. The cost of GC-TTT scales linearly in the frequency of test-time training. Hence, from this perspective, updating the policy less frequently seems desirable. At the same time, frequent updates allow the agent to focus on local information and to quickly correct when diverging from the optimal path to the goal. We demonstrate this in Figure 2 (middle) in the appendix, where we evaluate GC-TTT with GC-IQL in `antmaze play`. We find that the value estimates used for data selection are not accurate over long horizons (> 200 steps in `antmaze play`), leading to poor performance if the policy is updated too infrequently. However, as the frequency of TTT increases, we observe that GC-TTT leads to significant performance gains until improvement eventually saturates when fine-tuning every 100 steps. We repeat the same experiment on `pointmaze play`, which is an arguably simpler environment. We observe that performance already saturates at a lower frequency (i.e., $1/200$), suggesting that test-time training should be applied at shorter intervals in more complex environments.

Insight 5: GC-TTT scales better than model size. Having shown that GC-TTT predictably improves when allocating more compute, we analyze another option to scale test-time compute, namely by training larger policies, which are more expensive to evaluate. For this, we compare the performance of GC-TTT with a given frequency $1/K$ to the performance of larger policies that are not trained at test-time, but which have matched inference FLOPs to GC-TTT. To match the inference FLOPs of GC-TTT scaling and model scaling, we assume that compute requirements scale linearly with TTT frequency, but quadratically in the width of the policy. In Figure 2 (right) in the appendix, we find that GC-TTT consistently outperforms model scaling across a broad range of inference FLOPs.

4 Conclusion

This work introduces a framework of test-time training for offline goal-conditioned RL. We propose a self-supervised data selection scheme which chooses relevant and optimal data for the agent's current state and goal from an offline

dataset of trajectories. Our proposed method, GC-TTT, periodically fine-tunes the pre-trained policy on this data during evaluation. We find that GC-TTT consistently leads to significant improvements across several environments and underlying RL algorithms.

Our work opens up several exciting directions for future research, which we discuss in Appendix D.

References

Agarwal, S., Durugkar, I., Stone, P., and Zhang, A. f-policy gradients: A general framework for goal-conditioned rl using f-divergences. In *NeurIPS*, 2023.

Akyürek, E., Damani, M., Zweiger, A., Qiu, L., Guo, H., Pari, J., Kim, Y., and Andreas, J. The surprising effectiveness of test-time training for few-shot learning. *arXiv preprint arXiv:2411.07279*, 2024.

Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Abbeel, P., and Zaremba, W. Hindsight experience replay. In *NeurIPS*, 2017.

Atkeson, C. G., Moore, A. W., and Schaal, S. Locally weighted learning. *Lazy learning*, 1997.

Bagatella, M., Hübotter, J., Martius, G., and Krause, A. Active fine-tuning of multi-task policies. In *ICML*, 2025.

Bertolissi, R., Hübotter, J., Hakimi, I., and Krause, A. Local mixtures of experts: Essentially free test-time training via model merging. *arXiv preprint arXiv:2505.14136*, 2025.

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, 2024. *arXiv preprint arXiv:2410.24164*, 2024.

Bottou, L. and Vapnik, V. Local learning algorithms. *Neural computation*, 4(6), 1992.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. In *NeurIPS*, 2020.

Cleveland, W. S. Robust locally weighted regression and smoothing scatterplots. *Journal of the American statistical association*, 74(368), 1979.

Cleveland, W. S. and Devlin, S. J. Locally weighted regression: an approach to regression analysis by local fitting. *Journal of the American statistical association*, 83(403), 1988.

Dalal, K., Koceja, D., Hussein, G., Xu, J., Zhao, Y., Song, Y., Han, S., Cheung, K. C., Kautz, J., Guestrin, C., et al. One-minute video generation with test-time training. *arXiv preprint arXiv:2504.05298*, 2025.

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Eysenbach, B., Salakhutdinov, R. R., and Levine, S. Search on the replay buffer: Bridging planning and reinforcement learning. In *NeurIPS*, 2019.

Eysenbach, B., Zhang, T., Salakhutdinov, R., and Levine, S. Contrastive learning as goal-conditioned reinforcement learning. In *NeurIPS*, 2022.

Ghosh, D., Bhateja, C. A., and Levine, S. Reinforcement learning from passive data via latent intentions. In *ICML*, 2023.

Hansen, N., Jangir, R., Sun, Y., Alenyà, G., Abbeel, P., Efros, A. A., Pinto, L., and Wang, X. Self-supervised policy adaptation during deployment. In *ICLR*, 2021.

Hardt, M. and Sun, Y. Test-time training on nearest neighbors for large language models. In *ICLR*, 2024.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *CVPR*, 2016.

Heess, N., Wayne, G., Silver, D., Lillicrap, T., Erez, T., and Tassa, Y. Learning continuous control policies by stochastic value gradients. In *NeurIPS*, 2015.

Horgan, D., Quan, J., Budden, D., Barth-Maron, G., Hessel, M., van Hasselt, H., and Silver, D. Distributed prioritized experience replay. In *ICLR*, 2018.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W., et al. Lora: Low-rank adaptation of large language models. In *ICLR*, 2022.

Hübotter, J., Sukhija, B., Treven, L., As, Y., and Krause, A. Transductive active learning: Theory and applications. In *NeurIPS*, 2024.

Hübotter, J., Bongni, S., Hakimi, I., and Krause, A. Efficiently learning at test-time: Active fine-tuning of llms. In *ICLR*, 2025.

Kim, M. J., Pertsch, K., Karamcheti, S., Xiao, T., Balakrishna, A., Nair, S., Rafailov, R., Foster, E. P., Sanketi, P. R., Vuong, Q., et al. Openvla: An open-source vision-language-action model. In *CoRL*, 2022.

Kostrikov, I., Nair, A., and Levine, S. Offline reinforcement learning with implicit q-learning. In *ICLR*, 2022.

Krause, B., Kahembwe, E., Murray, I., and Renals, S. Dynamic evaluation of neural sequence models. In *ICML*, 2018.

Krause, B., Kahembwe, E., Murray, I., and Renals, S. Dynamic evaluation of transformer language models. *arXiv preprint arXiv:1904.08378*, 2019.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 1998.

Ma, Y. J., Yan, J., Jayaraman, D., and Bastani, O. Offline goal-conditioned reinforcement learning via f-advantage regression. In *NeurIPS*, 2022.

MacKay, D. J. Information-based objective functions for active data selection. *Neural computation*, 4(4), 1992.

Morari, M. and Lee, J. H. Model predictive control: past, present and future. *Computers & chemical engineering*, 1999.

Nachum, O., Gu, S. S., Lee, H., and Levine, S. Data-efficient hierarchical reinforcement learning. In *NeurIPS*, 2018.

Park, S., Ghosh, D., Eysenbach, B., and Levine, S. Hiql: Offline goal-conditioned rl with latent states as actions. In *NeurIPS*, 2023.

Park, S., Frans, K., Eysenbach, B., and Levine, S. Ogbench: Benchmarking offline goal-conditioned rl. In *ICLR*, 2025.

Peng, X. B., Kumar, A., Zhang, G., and Levine, S. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

Pinneri, C., Sawant, S., Blaes, S., Achterhold, J., Stueckler, J., Rolinek, M., and Martius, G. Sample-efficient cross-entropy method for real-time planning. In *CoRL*, 2020.

Rawlings, J. B., Mayne, D. Q., Diehl, M., et al. *Model predictive control: theory, computation, and design*, volume 2. Nob Hill Publishing Madison, WI, 2017.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022.

Ross, S., Gordon, G., and Bagnell, D. A reduction of imitation learning and structured prediction to no-regret online learning. In *AISTATS*, 2011.

Schaul, T., Horgan, D., Gregor, K., and Silver, D. Universal value function approximators. In *ICML*, 2015.

Silver, D., Singh, S., Precup, D., and Sutton, R. S. Reward is enough. *Artificial Intelligence*, 299, 2021.

Simonds, T. and Yoshiyama, A. Ladder: Self-improving llms through recursive problem decomposition. *arXiv preprint arXiv:2503.00735*, 2025.

Sun, Y., Wang, X., Liu, Z., Miller, J., Efros, A., and Hardt, M. Test-time training with self-supervision for generalization under distribution shifts. In *ICML*, 2020.

Sun, Y., Li, X., Dalal, K., Xu, J., Vikram, A., Zhang, G., Dubois, Y., Chen, X., Wang, X., Koyejo, S., et al. Learning to (learn at test time): Rnns with expressive hidden states. *arXiv preprint arXiv:2407.04620*, 2024.

Sutton, R. S. and Barto, A. G. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018.

Tian, S., Nair, S., Ebert, F., Dasari, S., Eysenbach, B., Finn, C., and Levine, S. Model-based visual planning with self-supervised functional distances. In *ICLR*, 2021.

Wang, T., Torralba, A., Isola, P., and Zhang, A. Optimal goal-reaching reinforcement learning via quasimetric learning. In *ICML*, 2023.

Yang, R., Lu, Y., Li, W., Sun, H., Fang, M., Du, Y., Li, X., Han, L., and Zhang, C. Rethinking goal-conditioned supervised learning and its connection to offline RL. In *ICLR*, 2022.

Zheng, C., Salakhutdinov, R., and Eysenbach, B. Contrastive difference predictive coding. In *ICLR*, 2024.

Zuo, Y., Zhang, K., Qu, S., Sheng, L., Zhu, X., Qi, B., Sun, Y., Cui, G., Ding, N., and Zhou, B. Ttrl: Test-time reinforcement learning. *arXiv preprint arXiv:2504.16084*, 2025.

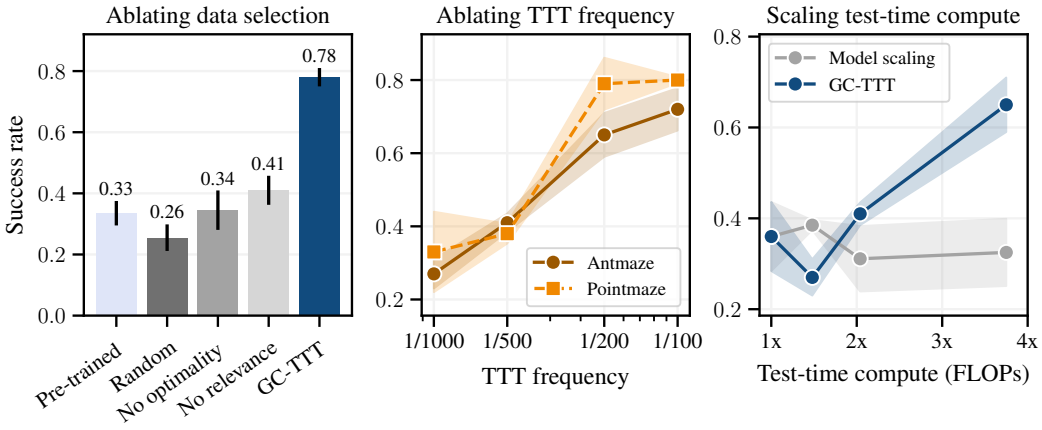

Figure 2. **Left:** Ablation of the data selection criteria. Both relevance and optimality have to be considered to filter the dataset for test-time training. **Middle:** Allocating more compute by increasing the frequency of TTT improves performance, and saturates slightly earlier in simpler environments. **Right:** We compare scaling test-time compute of GC-TTT (by increasing TTT frequency) to scaling the policy networks such that inference FLOPs are matched. We find that GC-TTT scales well with increased test-time compute, while scaling model size does not yield significant improvements. The initial drop of GC-TTT is because value estimates are not accurate over long-horizons (cf. Insight 4).

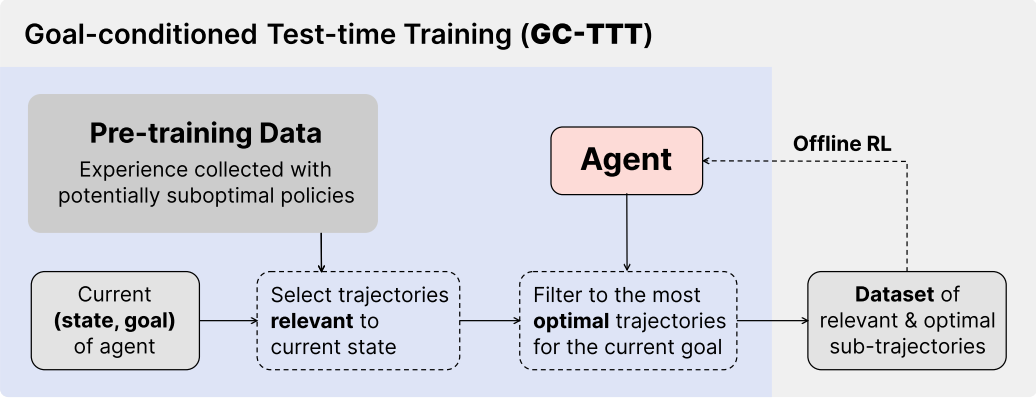

Figure 3. **GC-TTT** specializes the agent to the next steps for achieving its target goal.

A Related Work

Goal-conditioned reinforcement learning Reinforcement learning (RL) research primarily builds upon the framework of Markov decision processes (MPDs), which define their objective based on a scalar function of states and action, referred to as a reward function (Sutton & Barto, 2018). While reward functions may be very expressive (Silver et al., 2021), a conditional reward is more flexible and can model a *family* of behaviors. One such approach is goal-conditioned reinforcement learning (GCRL). Here, the agent's objective is to achieve some specified goal which is modeled by a sparse reward, indicating whether the goal is achieved (Andrychowicz et al., 2017; Eysenbach et al., 2022; Ma et al., 2022; Agarwal et al., 2023). The GCRL framework has been remarkably successful when coupled with neural function approximation (Schaul et al., 2015), which is capable of amortizing the enlarged input space of the policy, compared to individual-task RL. A prominent example of GCRL is the RL-training of large language models (e.g., DeepSeek-AI, 2025) where the language model learns to achieve a broad family of goals such as solving math problems. As the reward function is often known, several methods for relabeling (Andrychowicz et al., 2017) and self-supervision (Tian et al., 2021) have been proposed to allow off-policy learning for all possible goals from arbitrary experience. Due to the particular structure of the reward function, goal-conditioned RL allows for specific algorithms beyond TD-learning, including contrastive (Eysenbach et al., 2022; Zheng et al., 2024) and quasimetric (Wang et al., 2023) formulations. Furthermore, goal-conditioned algorithms can be easily adapted to the offline setting considered in our work (Ma et al., 2022; Park et al., 2023; 2025). In both offline and online settings, the goal-conditioned policy is evaluated by commanding a target goal or a subgoal selected by a high-level component (Nachum et al.,

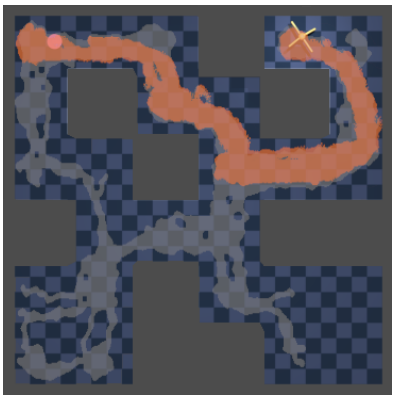

Figure 4. Visualization of data selection by GC-TTT in `antmaze play` during one evaluation episode (in orange). A random subset of trajectories from the dataset is shown in gray.

Algorithm 1 Goal-conditioned Test-time Training

Require: Pre-trained policy parameters θ, dataset $\mathcal{D}$, horizon K, number of gradient steps N, learning rate α, distance d, goal-conditioned value estimate $\widehat{V}$, locality threshold ϵ, percentile q.

for each evaluation episode **do**
  $s \sim \mu_0, g^\star \sim \mu_g$ ▷ sample initial state and evaluation goal
  $\bar{\theta} \leftarrow \theta$ ▷ store policy parameters
  while not done **do**
    $\mathcal{D}_{\text{rel}}(s) \leftarrow \{(s_1, \dots) \in \mathcal{D} \mid d(s, s_1) < \epsilon\}$ ▷ select **relevant** sub-trajectories (Eq. 2)
    $C \leftarrow q$-th percentile of $\{\widehat{V}(\tau \,|\, g^\star) \mid \tau \in \mathcal{D}_{\text{rel}}(s)\}$
    $\mathcal{D}(s, g^\star) \leftarrow \{\tau \in \mathcal{D}_{\text{rel}}(s) \mid \widehat{V}(\tau \,|\, g^\star) \geq C\}$ ▷ filter to **optimal** sub-trajectories (Eq. 4)
    for $i \in [1, \dots, N]$ **do**
      $\theta \leftarrow \theta - \alpha \nabla_\theta \mathbb{E}_{s' \sim \mathcal{D}(s, g^\star)} \mathcal{L}(s', g^\star; \theta)$ ▷ fine-tune policy locally
    end for
    for $i \in [1, \dots, K]$ **do**
      $a \sim \pi_\theta(s \mid g)$ ▷ sample action
      $s \sim P(\cdot \mid s, a)$ ▷ execute action
    end for
    $\theta \leftarrow \bar{\theta}$ ▷ reset policy
  end while
end for

Figure 5. The four envs considered from OGBench (Park et al., 2025): from top left in clockwise order, `humanoidmaze`, `cubesingle`, `antmaze`, `pointmaze`.

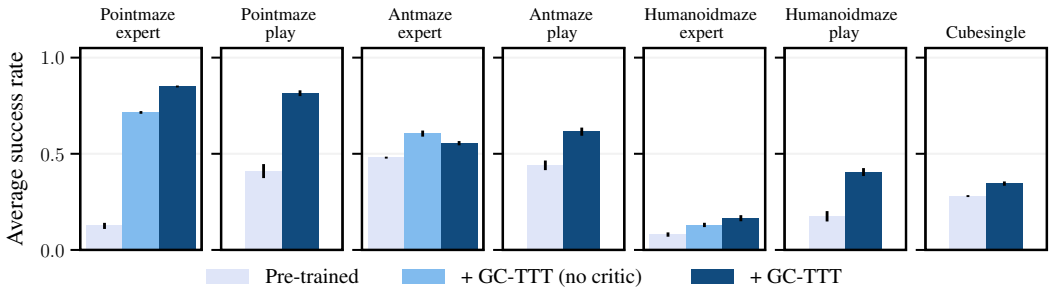

Figure 6. Success rates of GC-TTT within each environment, averaged across RL backbones.

2018; Park et al., 2023). The policy parameters then remain unchanged throughout evaluation. Our work investigates efficient training of the policy weights at test-time, and can be combined with any of the abovementioned value-based algorithms.

Test-time training In machine learning, models are traditionally trained on a fixed training set and then kept frozen during evaluation. While this has been the standard practice in machine learning for decades, early work has also discussed specializing the model at test-time to each prediction task. First examples of this so-called transductive approach are local learning (Cleveland, 1979; Cleveland & Devlin, 1988; Atkeson et al., 1997) and local fine-tuning (Bottou & Vapnik, 1992). More recently, the idea of test-time training (TTT) (Sun et al., 2020) has regained attention in the context of fine-tuning large foundation models during evaluation (e.g., Krause et al., 2018; 2019; Hardt & Sun, 2024; Sun et al., 2024). TTT on (self-)supervised signals for few gradient steps has since shown success in domains such as control (Hansen et al., 2021), language modeling (Hardt & Sun, 2024; Hübotter et al., 2025; Sun et al., 2024; Bertolissi et al., 2025), abstract reasoning (Akyürek et al., 2024), and video generation (Dalal et al., 2025). Many standard TTT methods train on carefully selected data from the pre-training dataset (i.e., do not add any new priviledged information; Hardt & Sun, 2024; Hübotter et al., 2025), and several works studied how to select data for imitation optimally (e.g., MacKay, 1992; Hübotter et al., 2024; Bagatella et al., 2025).

Test-time reinforcement learning In this work, we study test-time offline RL (TTORL), where the offline dataset contains trajectories from different policies conditioned on different goals. Therefore, unlike in previous work on TTT, this data should not be imitated directly. Despite this challenge, we show that GC-TTT can substantially improve performance the performance of standard offline RL algorithms. Our work is closely related to concurrent work, which studies a form of test-time online RL (abbreviated TTRL) with language models (Zuo et al., 2025; Simonds & Yoshiyama, 2025). Unlike their work, we propose to dynamically train during evaluation of a single goal, which we identify as crucial for achieving maximum performance. Intuitively, our work on TTRL combines the pre-training paradigm commonly pursued in GCRL and the standard RL paradigm of continuously training on experience collected for a single task. In GC-TTT, the pre-trained model is specialized to each individual task during evaluation.

B Background

We model the dynamical system as a reward-free Markov decision process $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, \gamma, \mu_0)$ (Eysenbach et al., 2022), where $\mathcal{S}$ and $\mathcal{A}$ are potentially continuous state and action spaces, $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is a stochastic transition function, γ is a discount factor and $\mu_0 \in \Delta(\mathcal{S})$ is an initial state distribution. We introduce a goal space $\mathcal{G}$ and identify it with the state space $\mathcal{G} = \mathcal{S}$ for simplicity, although goal abstraction remains possible. As standard in goal-conditioned settings, we assume the existence of a distance function $d : \mathcal{S} \times \mathcal{G} \to \mathbb{R}$ to determine *goal achievement*, and define a conditional reward function as

$$R(s, g) = \begin{cases} -1 & \text{if } d(s, g) \geq \epsilon \\ 0 & \text{otherwise,} \end{cases} \tag{5}$$

for some small fixed threshold ϵ. In turn, the reward function induces a conditional value function for each policy $\pi : \mathcal{S} \times \mathcal{G} \to \Delta(\mathcal{A})$:

$$V^\pi(s_0 \mid g) = \mathop{\mathbb{E}}_{P,\pi}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, g)\right] \quad \text{where} \quad s_{t+1} \sim P(s_t, a_t),\ a_t \sim \pi(s_t \mid g). \tag{6}$$

Intuitively, the value function computes the negative, expected, discounted number of steps required to reach the goal under a given policy. The optimal policy for some goal distribution $\mu_\mathcal{G}$ can then be defined as $\pi^\star = \arg\max_\pi \mathbb{E}_{g\sim\mu_\mathcal{G}, s_0\sim\mu_0} V^\pi(s_0; g)$, and induces a quasi-metric structure in its value function (Wang et al., 2023). Most practical algorithms optimize over a broad and dense goal distribution $\mu_\mathcal{G}$ (see, e.g., Andrychowicz et al., 2017), but are only deployed to achieve one specific goal during each episode at inference.

Offline policy (pre-)training The standard offline goal-conditioned reinforcement learning pipeline pre-trains a policy π on an offline dataset $\mathcal{D}$ of trajectories $(s_0, a_0, s_1, a_1, \dots)$. Most practical methods parameterize the policy as a neural network π_θ, and use stochastic optimization to find

$$\theta^\star_{\text{pre}} = \operatorname*{argmax}_\theta J_{\text{pre}}(\theta), \tag{7}$$

for a given pre-training objective J_{pre} (e.g., stochastic value gradients (Heess et al., 2015) or behavior cloning (Ross et al., 2011)). This objective is normally specified as an expectation over the state-goal distribution from the pre-training dataset:

$$J_{\text{pre}}(\theta) = -\mathbb{E}_{s\sim p_s(\cdot\mid\mathcal{D}),\ g\sim p_g(\cdot\mid s,\mathcal{D})}\, \mathcal{L}(s, g, \theta), \tag{8}$$

where p_s and p_g are state and goal distributions, respectively. Normally, the loss function $\mathcal{L}$ will also depend on actions sampled from $\mathcal{D}$; however, these actions are naturally those paired with selected states (e.g., when $\mathcal{L}$ is a behavior cloning loss). Except for prioritized sampling schemes (Horgan et al., 2018), p_s is generally uniform; p_g is instead conditioned on s, and may sample future goals from the same trajectories, or random ones (Ghosh et al., 2023). $\mathcal{L}$ represents an arbitrary loss function; within offline reinforcement learning, it normally lies on a spectrum between supervised learning (behavior cloning) and fully off-policy reinforcement learning. At its core, the objective in Equation 8 aims to find a policy that is optimal *on average* (w.r.t. the goal distribution p_g), which may lead to a locally suboptimal solution for *specific goals*, especially in noisy settings or with limited model capacity.

After this training phase, the policy is evaluated on a *single goal per episode*. We study the problem of fine-tuning the pre-trained model during test-time using offline RL to specialize the policy *locally*. We call this setting *test-time offline reinforcement learning* (TTORL). Our method, GC-TTT, specializes the policy to the agent's current state and goal at test-time.

C Computational Efficiency

While GC-TTT leads to substantial performance gains, it incurs additional computational costs at test-time. This cost scales with several design choices; in particular, it scales linearly with the TTT frequency $1/K$ and with the number of gradient steps N for each iteration. Each gradient update can be as efficient as two forward passes, of which one is required at each time step for standard evaluation. Moreover, there is an overhead at each fine-tuning iteration due to data selection: if parallelization is possible (e.g., on graphics accelerators), this can be near-constant in practice, otherwise the overhead increases linearly with the number of samples $|\mathcal{D}|$. Finally, this cost is not distributed evenly through the evaluation, but rather at regular intervals, which can result in a non-constant control frequency.

In practice, we find that on modern accelerators GC-TTT completes a single evaluation episode (1000 steps) in ~ 85 seconds, for an average control frequency > 10 Hz. While performance can be further improved by efficient implementations and more performant hardware, this number is comparable to the inference speed of methods relying on efficient model-based planning (Pinneri et al., 2020), or VLAs with diffusion heads (Black et al., 2024). For context, a critic-free version of the algorithm and the pre-trained policy reach a control frequency of > 75 and ~ 190 Hz, respectively. For an empirical study of the trade-off between performance and compute requirements, we refer to Section 3.

D Limitations and Future Work

The main practical limitation of this work arguably lies in its compute requirements, which we discuss in Appendix C. While our measured average control frequency of GC-TTT is compatible with some robotic applications, high-frequency control would require development of a lazy variant of GC-TTT. Further, GC-TTT relies on reasonable value estimates and on available data related to the agent's current state and goal.

By showing that test-time training can effectively improve policies from off-policy experiential data, our work opens up several exciting directions for further research. On a practical level, our findings suggest that current offline GCRL algorithms are unable to accurately fit each of the tasks they are trained on. The reason for this should be investigated, and might suggest directions for improving offline RL pre-training. Moreover, GC-TTT does not leverage the data that is freshly collected at test-time, beyond the current state. We believe that leveraging this new experience with a test-time online RL algorithm is an exciting direction. Finally, the framework proposed in this work can be readily extended beyond goal-reaching tasks to more general decision-making settings, including other domains such as reasoning in natural language. We expect that progressively shifting computational resources to test-time training can substantially improve performance in areas ranging from robotic control to reasoning agents.

E Experiment Details

Backbones GC-TTT is applicable across the broad class of value-based offline goal-conditioned algorithms. We select a representative subset of algorithms, and focus our evaluation on GC-BC (Yang et al., 2022) and GC-IQL (Kostrikov et al., 2022). GC-BC (behavior cloning) is a supervised algorithm for goal-conditional imitation, which directly matches the policy's output to the actions present in the offline dataset. GC-IQL is an implicit method for offline RL, which bypasses evaluation on out-of-distribution actions through expectile regression. We adopt the variant using advantage-weighted regression (AWR, Peng et al., 2019) for policy extraction. We select BC and IQL due to their widespread adoption, and

their representativeness of on-policy and off-policy learning in offline settings, respectively.

F Discussion of Offline RL Algorithms

The empirical validation of this work builds upon two widespread algorithms fro extracting policies from offline data. This section provides a concise introduction to them

F.1 Behavior Cloning

Behavior Cloning (Ross et al., 2011) is a standard approach for policy learning, which reduces a control problem to supervised reconstruction. Given a distribution μ over state-action pairs, a policy π_θ may be trained by minimizing

$$J_{\text{BC}}(\theta) = -\mathop{\mathbb{E}}_{(s,a)\sim\mu} \log \pi_\theta(a|s). \tag{9}$$

The resulting policy will maximize the likelihood of actions in the dataset, and thus converge to the behavioral policy, if there is one.

F.2 Implicit Q-Learning

Implicit Q-Learning (Kostrikov et al., 2022) is an offline RL algorithm which avoids querying the critic on out-of-distribution actions, and directly estimates a value function through expectile regression. Given a distribution μ of state-action-next state transitions labeled with a reward, IQL defines the following losses:

$$\mathcal{L}_Q(\phi) = \mathop{\mathbb{E}}_{(s,a,r,s')\sim\mu} (r + \gamma V_\psi(s') - Q_\phi(s,a))^2, \tag{10}$$

and

$$\mathcal{L}_V(\psi) = \mathop{\mathbb{E}}_{(s,a,r)\sim\mu} L^\alpha(Q_\phi(s,a) - V_\psi) \quad \text{with } L^\alpha(x) = |\alpha - \mathop{1}_{x<0}| x^2. \tag{11}$$

As the expectile α approaches one, V approximates the maximum of Q. Thus, IQL is capable of off-policy learning, and can estimate the value function of the optimal policy (Kostrikov et al., 2022). An optimal policy may then be extracted through advantage weighted regression (Peng et al., 2019):

$$\mathcal{L}_\pi(\theta) = \mathop{\mathbb{E}}_{(s,a,r)\sim\mu} \exp\Big(\beta\big(Q_\phi(s,a) - V_\psi(s)\big)\Big) \log \pi_\theta(a|s), \tag{12}$$

where β interpolates between extracting the behavior policy, or the greedy one.

G Additional Experiments

G.1 Ablation on the finetuning learning rate and the number of gradient steps

Figure 7 presents the success rate of GC-TTT with GC-IQL on `antmaze` play as the number of test-time training gradient steps N changes. We observe that increasing the number of gradient steps helps initially, as the policy can better fit the local data. However, an excessive number of gradient steps may decrease performance, as the policy is trained on a small dataset, and offline issues such as value overestimation may arise. Regarding the learning rates, the higher learning rate facilitates quicker adaptation and shows a slight advantage in peak performance. While there are differences, both learning rates yield comparable results as gradient steps increases.

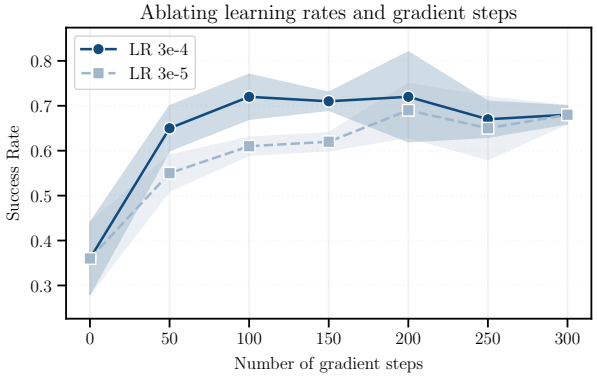

Figure 7. GC-TTT results for different gradient steps.

G.2 Value-based relevance criterion

The relevance criterion defined in Equation 2 relies on the reward criterion normally exposed in goal-conditioned settings.

	Reward-based	Value-based (C=-14)	Value-based (C=-18)	Value-based (C=-22)
`antmaze` play	0.73 ± 0.01	0.68 ± 0.04	0.73 ± 0.01	0.67 ± 0.03

Table 2. Success rates of GC-TTT with the original relevance criterion and a value-based version, on top of GC-IQL. Numbers in parentheses are standard errors across 3 seeds.

When this is not available, however, the criterion may be replaced by a proxy based on a value estimate:

$$\textbf{Value-based relevance:} \qquad \mathcal{D}_{\text{rel}}(s) = \{(s_1, \dots s_H) \in \mathcal{D} \mid V(s, s_1) > C\}. \tag{13}$$

This time, C is a constant hyperparameter, which, similarly to ϵ, can control the maximum temporal distance between the current state s and selected trajectories.

We find that, empirically, this modification does not affect performance significantly: we report performance of GC-TTT with the original and the value-based relvance criterion in `antmaze` in Table 2.

G.3 Parameter scaling ablation

Figure 2 (right) studies the extent to which performance may be improved by scaling the parameter count of the policy. In order to ensure that the absence of improvement does not stem from hyperparameter choices, we additionally report results for different learning rates in Figure 8.

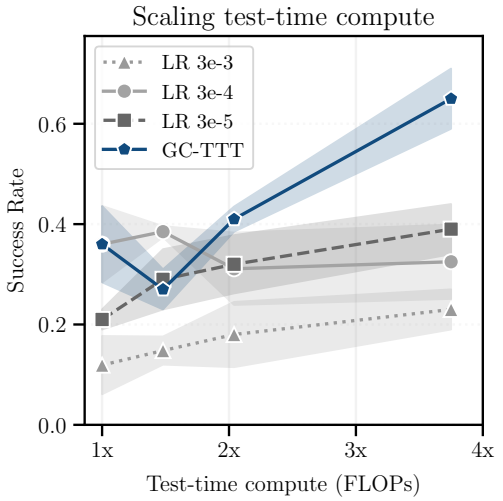

Figure 8. Model scaling results for different learning rates.

H Implementation Details

For environments and backbone algorithms, we adopt the default hyperparameters presented in OGBench (Park et al., 2025), with the exception of GC-IQL, which we evaluate in its AWR variant. We set the BC regularization coefficient α to values of 0.003, 0.3, 0.1 and 1.0 for `pointmaze`, `antmaze`, `humanoidmaze` and `cubesingle`, respectively.

GC-TTT introduces some additional hyperparameters: the horizon $K = 100$, the number of gradient steps $N = 100$, and the percentile $q = 0.2$.

For further details, we refer to the code released on our anonymous website.

H.1 Estimating FLOPs

Figure 2 (right) presents estimates of test-time compute (FLOPs) in its x-axis. In order to compute these estimates, we make the following simplifying assumptions:

- The input and output size of the policy is negligible with respect to its witdh w; hence, the number of sum/multiply operations for one forward pass is $C \approx 2nw^2 = 4w^2$, as the policy is an MLP with $n = 2$ hidden layers.
- The cost of a forward pass does not depend on the batch size.
- A backward pass requires twice the compute as a forward pass.

Following from these assumptions, the cost for a single evaluation episode with 1000 steps is $C_{\text{no-TTT}} \approx 1000C = 4000w^2$. Considering the test-time training frequency f and the number of gradient steps $m = 100$, the cost of the same operation with GC-TTT is $C_{\text{TTT}} = 1000f(1 + 6Cm) + 1000C$. The first term includes the cost of data selection (1 for the single forward pass required for computing values used in 4) and fine-tuning ($6Cm$, where we assume that the critic is the same size of the

policy, and we need to compute gradients of the policy with respect to the critic's output). The cost of other operations not involving the neural network are not considered. Given the default width $w = 512$ we may then compute the compute cost without GC-TTT ($\approx 10^9$ FLOPs), and for test-time training frequencies $[1/1000, 1/500, 1/200]$ ($\approx 1.6 \cdot 10^9, 2.2 \cdot 10^9$ and $4 \cdot 10^9$ FLOPs, respectively). Given these increased compute budgets, we can finally solve for the values of w necessary for meeting this compute cost without GC-TTT ($\approx 624, 732, 992$), which were used to obtain the grey curve in Figure 2 (right).