Meta-GC-TTT: Training Offline Goal-Conditioned Policies for Test-Time Adaptation

Anonymous Author(s)
Affiliation
Address
email

Abstract

In offline goal-conditioned reinforcement learning, Test-Time Training (TTT) can specialize a pre-trained policy to the current state and goal at deployment. This turns a broad goal-conditioned policy into a query-specific expert. Yet, standard offline pre-training optimizes the policy *before* TTT, not *after* it. As a result, the policy is not trained for the gradient dynamics it will face at test time. We introduce META-GC-TTT, a framework for learning test-time-trainable goal-conditioned policies. META-GC-TTT samples state-goal tasks from the offline dataset, adapts the policy with TTT, and updates the base policy for post-TTT performance. Our evaluation on the OGBench loco-navigation suite demonstrates that meta-learned initializations significantly improve zero-shot performance and achieve a $(3-5\times)$ increase in adaptation efficiency. Overall, offline goal-conditioned policies should not only be trained not only to act, but also to adapt.

1 Introduction

The standard paradigm of modern machine learning has shifted from training task-specific models from scratch to a two-phase process: *large-scale pre-training* followed by *specialized adaptation*. This evolution, exemplified by Foundation Models in language and vision [Radford et al., 2018, Touvron et al., 2023, Ouyang et al., 2022, Yang et al., 2025, Team et al., 2025, Radford et al., 2021, Dosovitskiy et al., 2021, Beyer et al., 2024], leverages vast unlabeled datasets and self-supervised objectives to learn underlying data structures.

However, this paradigm has only recently begun to transition to Reinforcement Learning (RL) for robotics. Emerging research proposed Vision-Language-Action (VLA) models [Zitkovich et al., 2023, Kim et al., 2024, Black et al., 2026, Intelligence et al., 2025] and *offline goal-conditioned RL (GCRL)* [Liu et al., 2022, Eysenbach et al., 2022b] as promising candidates for robotic foundation models. In particular, GCRL offers two pivotal advantages: *(i) offline scalability*, which leverages massive, diverse datasets while bypassing the safety and cost constraints of online interaction; and *(ii) self-supervised objectives*, which provide a reward-free training signal analogous to next-token prediction in LLMs. By treating reached states as goals, this objective enables the extraction of general-purpose skills directly from unlabeled trajectories.

Yet, scaling alone may be insufficient for high-performance agents. While pre-training provides a broad "common sense," complex tasks require a final specialization phase. This paradigm has recently been pushed to its logical extreme with *Test-Time Training (TTT)* [Sun et al., 2020, Hardt and Sun, 2024], where a model is adapted on-the-fly for each specific test instance. By performing gradient descent on data relevant to the current query at the moment of inference, TTT transforms a broad generalist model into a localized expert tailored to the immediate task.

At its core, this work connects goal-conditioned pre-training with test-time adaptation through *meta-learning*. If a model is intended to be adaptive at test-time, the training phase should explicitly

Submitted to 40th Conference on Neural Information Processing Systems (NeurIPS 2026). Do not distribute.

optimize for this capability. Meta-learning algorithms [Finn et al., 2017, Nichol et al., 2018] provide the framework to ensure a model may truly *learn to learn*, optimizing the initialization specifically for its ability to specialize via local gradients.

The main contributions of this paper are as follows:

- We introduce META-GC-TTT, a unified post-training framework that leverages meta-learning to optimize policies for test-time adaptation, instantiating three distinct algorithms: CONTINUAL-TTT, TT-FOMAML and TT-REPTILE.
- We propose a novel meta-update mechanism using an early-stopping approach based on goal-conditioned test loss. This dynamically determines the inner-loop horizon to prevent overfitting and stabilize adaptation.
- We provide an extensive evaluation across the loco-navigation suite of OGBench. Our results demonstrate that meta-learned initializations significantly improve zero-shot performance and achieve up to a $5\times$ increase in adaptation efficiency compared to standard TTT.

2 Related Work

Offline Reinforcement Learning Offline RL focuses on learning optimal policies from static datasets without environment interaction [Ernst et al., 2005, Levine et al., 2020], and represents a promising paradigm for training robotics foundation models [Amin et al., 2025]. The primary challenge is the distributional shift between the behavior policy π_β and the learned policy π_θ, which causes catastrophic value overestimation on out-of-distribution (OOD) actions. Conservative methods like CQL [Kumar et al., 2020] and in-sample methods like IQL [Kostrikov et al., 2021] have been developed to mitigate these issues. Such techniques are in practice deployed for the underlying policy through this work.

Goal-Conditioned RL GCRL enables agents to reach diverse targets by training goal-conditioned policies [Liu et al., 2022, Eysenbach et al., 2022b, Ma et al., 2022], and can be applied naturally to offline settings [Agarwal et al., 2023], as most objectives are self-supervised [Andrychowicz et al., 2017, Tian et al., 2021], and do not require reward labels. Despite the success of self-supervised objectives, evidence suggests that effective reasoning horizon remains a primary bottleneck for scaling these models to complex tasks [Park et al., 2025b]. We utilize the OGBench suite [Park et al., 2025a] to evaluate how META-GC-TTT reduces this effective horizon through localized adaptation.

Test-Time Training (TTT) The TTT paradigm involves adapting model parameters on-the-fly for each specific test instance [Sun et al., 2020]. In decision-making, TTT transforms a generalist model into a localized expert [Park et al., 2024, Bagatella et al., 2025]. Our work extends the GC-TTT framework [Bagatella et al., 2025] by aligning the initialization specifically for inference-time gradient dynamics.

Optimization-based Meta-Learning Meta-learning aims to learn representations that facilitate rapid adaptation [Finn et al., 2017]. Algorithms like MAML optimize a model's initialization specifically for post-adaptation performance, while Reptile merges expert parameters with a base model via interpolation [Nichol et al., 2018]. Multiple variations of these methods have been proposed [Rajeswaran et al., 2019, Antoniou et al., 2018, Finn et al., 2018, Liu et al., 2019]. Our framework, META-GC-TTT, differentiates itself by integrating meta-learning with an efficient TTT inner loop.

3 Background

3.1 Goal-Conditioned Offline Reinforcement Learning

Goal-Conditioned Reinforcement Learning (GCRL) may be formalized through a goal-augmented Markov decision process (G-MDP) defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{G}, P, R, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{G}$ is the goal space (often $\mathcal{G} \subseteq \mathcal{S}$), $P(s'|s,a)$ represents goal-independent transition dynamics, $R(s,g)$ is the goal-conditioned reward function (typically sparse: $R(s,g) = \mathbf{1}_{d(s,g)\leq\epsilon}$), and $\gamma \in [0,1)$ is the discount factor. The objective is to learn a policy $\pi(a|s,g)$ maximizing

an expectation over the *value function*:

$$V^{\pi}(s|g) = \mathbb{E}_{P,\pi}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, g) \mid s_0 = s\right]. \tag{1}$$

An state-action value function may be defined similarly: $Q^{\pi}(s, a|g) = \mathbb{E}_{P,\pi}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, g) \mid s_0 = s, a_0 = a\right]$.

In the offline setting, π needs to be estimated from a static dataset $\mathcal{D} = \{(s_i, a_i, r_i, s'_i)\}_{i=1}^{n}$ collected by an unknown behavior policy π_β. A multitude of algorithms has been proposed for this setting [Park et al., 2023, Zeng et al., 2023, Zheng et al., 2023, Eysenbach et al., 2022a, Zhou and Kao, 2026], among which we will consider two representative instances. Among supervised approaches, Goal-Conditioned Behavioral Cloning (GCBC) [Yang et al., 2022] trains a policy parameterized by θ by maximizing the likelihood of actions taken in the dataset:

$$\mathcal{L}_{BC}(\theta) = \mathbb{E}_{(s,a,g)\sim\mathcal{D}_g}\left[-\log \pi_\theta(a|s, g)\right] \tag{2}$$

While stable, GCBC is limited by the quality of π_β. Reinforcement learning algorithms instead rely on the estimation of V^{π} for improving the policy. Among those, Goal-Conditioned Implicit Q-Learning (GCIQL) [Kostrikov et al., 2021, Zeng et al., 2023] trains value estimates through expectile regression and TD learning, and updates the policy $\pi_\theta(a|s, g)$ via either Advantage Weighted Regression (AWR) [Peng et al., 1910]:

$$\mathcal{L}_{AWR}(\theta) = \mathbb{E}_{(s,a,g)\sim\mathcal{D}_g}\left[-\exp(\beta A(s, a, g)) \log \pi_\theta(a|s, g)\right] \tag{3}$$

or DDPG-BC [Lillicrap et al., 2019, Fujimoto and Gu, 2021], which regularizes a deterministic policy gradient with a behavioral cloning term:

$$\mathcal{L}_{DDPG-BC}(\theta) = \mathbb{E}_{(s,a,g)\sim\mathcal{D}_g}\left[-Q(s, \hat{a}, g) + \alpha\|\hat{a} - a\|^2\right] \text{ where } \hat{a} \sim \pi_\theta(\cdot|s, g). \tag{4}$$

In either case, the policy is in practice a neural network updated through first-order methods, making it suitable for gradient-based test-time adaptation.

3.2 Goal-Conditioned Test-time Training (GC-TTT)

Test-Time Training (TTT) is a model adaptation paradigm where a base model is fine-tuned for each specific test instance τ during deployment [Sun et al., 2020, Hardt and Sun, 2024]. Given base model parameters θ_0 and a task-specific support set $\mathcal{D}_\tau$, TTT utilizes a transformation operator U_τ^K to produce specialized *expert* parameters (or "fast weights") $\theta^{TTT} = U_\tau^K(\theta_0)$. This operator typically consists of K steps of stochastic gradient descent on some task-specific objective $\mathcal{L}_\tau$:

$$\theta_k \leftarrow \theta_{k-1} - \alpha\nabla_{\theta_{k-1}}\mathcal{L}_\tau(\theta_{k-1}, \mathcal{D}_\tau) \tag{5}$$

where α is a learning rate. In the context of GCRL, Bagatella et al. [2025] recently demonstrated that TTT can specialize goal-conditioned policies to specific test instances (in this case, state-goal pairs $(s, g) = \tau$). Given a base policy π_{θ_0} and a state-goal pair (s, g), GC-TTT (i) describes a self-supervised procedure $\texttt{filter}(\mathcal{D}, s, g)$ to extract an instance-specific dataset $\mathcal{D}_{(s,g)} = \mathcal{D}_\tau$ from an unlabeled pre-training dataset $\mathcal{D}$ and (ii) simply applies the transformation operator $U_{(s,g)}^K$ on top of some underlying GCRL objective $\mathcal{L}_{(s,g)}$. We refer to Appendix A for more details.

3.3 Optimization-Based Meta-Learning

Optimization-based meta-learning seeks to find an initialization θ that minimizes the "learning distance" to future tasks τ, ensuring the model is easy to fine-tune. Model-Agnostic Meta-Learning (MAML) [Finn et al., 2017] explicitly optimizes for the performance of the model *after* adaptation. The meta-objective finds an initialization $\theta^\diamond$ such that the parameters resulting from the inner-loop adaptation $U_g^K(\theta)$ perform optimally on a held-out query set $\mathcal{D}_\tau^{test}$:

$$\theta^\diamond = \arg\min_{\theta\in\Theta} \mathbb{E}_\tau\left[\mathcal{L}_\tau\left(U_\tau^K(\theta), \mathcal{D}_\tau^{test}\right)\right] \tag{6}$$

This induces a bilevel optimization where the outer loop updates θ by back-propagating through the gradient descent path of the inner loop. To avoid the computational cost of the second-order

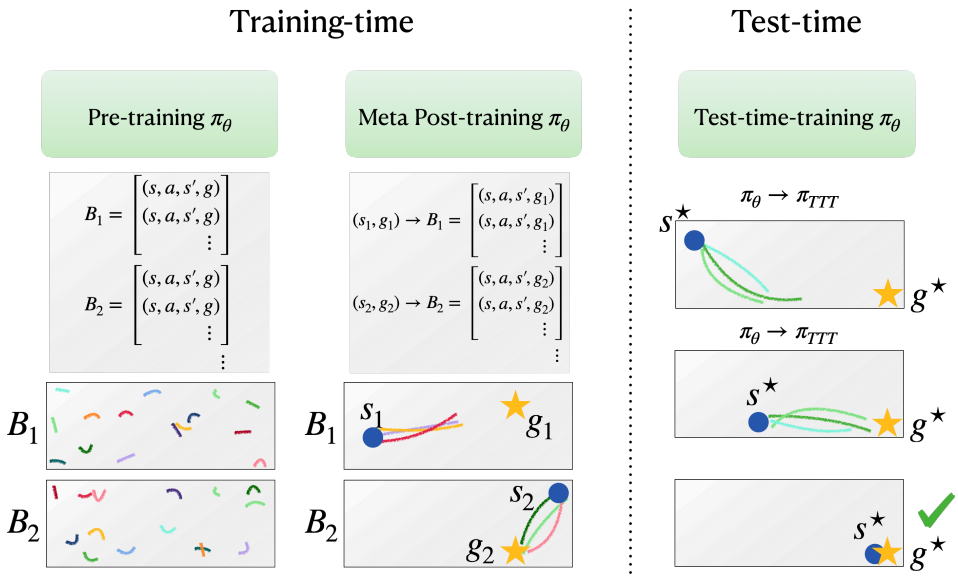

Figure 1: META-GC-TTT is an offline post-training scheme, based on meta-learning, to adapt a pre-trained policy and prepare it for test-time training. During post-training, we perform a meta-optimization loop by sampling starting states and goals as task instances, fine-tuning local copies of the policy on relevant batches of data and using a meta-optimization rule for back-propagating the knowledge of fine-tuned models into the base policy.

derivatives required by MAML, First-Order MAML (FOMAML) [Finn et al., 2017] approximates the meta-gradient by treating the inner-loop update as a constant:

$$\theta \leftarrow \theta - \beta \cdot \nabla_{U_\tau^K(\theta)} \mathcal{L}_\tau(U_\tau^K(\theta)), \tag{7}$$

where β is a meta-learning rate. This assumes a locally linear loss landscape, making it memory-efficient while remaining effective for small inner-loop learning rates. Reptile [Nichol et al., 2018] is a first-order algorithm that simplifies the meta-update rule by performing model merging with the adapted weights:

$$\theta \leftarrow \theta + \epsilon(U_{\mathcal{D}_\tau}^K(\theta) - \theta) \tag{8}$$

where ϵ is a step size. Unlike MAML, it does not require an explicit query/support split for the meta-gradient and is claimed to find an initialization that is geometrically close to the manifolds of optimal solutions for the task distribution.

4 META-GC-TTT

We propose META-GC-TTT, a post-training framework designed to align pre-trained offline GCRL policies for efficient test-time adaptation, schematized in Figure 1. Through meta-learning, we transform the policy into an *adaptive process*. Rather than only maximizing zero-shot performance with standard pre-training objectives while remaining agnostic of test-time processes, we further optimize the policy parameters θ to be "ready for TT gradients," ensuring that a small number of steps K on a local goal-specific dataset $\mathcal{D}_{(s,g)}$ can shift the behavior from a general prior to a specialized expert.

Given a goal-conditioned policy π_{θ_0} pre-trained through an underlying RL algorithm (e.g. GCBC or GCIQL) and an offline dataset $\mathcal{D}$, we can instantiate META-GC-TTT as a bilevel optimization routine (Algorithm 1). In the inner loop, we leverage GC-TTT [Bagatella et al., 2025]: given a test-time instance as a state-goal tuple (s, g), GC-TTT filters the dataset $\mathcal{D}$ to extract task-specific batches $\mathcal{D}_{(s,g)}$, and quickly adapts the policy through a few gradient descent steps. As the adaptation procedure is gradient-based, we *amortize* it through an outer meta-learning loop, as described in Algorithm 1.

Algorithm 1 involves several routines, namely:

1. *Task sampling:* A meta-batch of B tasks, each composed of a state-goal pair (s_i, g_i), is sampled uniformly from the dataset $\mathcal{D}$.

Algorithm 1 META-GC-TTT Post-training

Require: Pre-trained policy π_{θ_0}, Dataset $\mathcal{D}$, rates α, β, outer batch size B, outer steps N and inner steps K

$\theta \leftarrow \theta_0$
for $n = 1$ **to** N **do** ▷ Outer loop
  Sample meta-batch of tasks $(s_i, g_i)_{i=1}^{B}$ ▷ Task sampling
  for $i = 1$ **to** B **do**
    $\mathcal{D}_{(s_i,g_i)} \leftarrow \texttt{filter}(\mathcal{D}, s_i, g_i)$ ▷ Data filtering
    $\mathcal{D}^{train}_{(s_i,g_i)}, \mathcal{D}^{test}_{(s_i,g_i)} \leftarrow \text{split}(\mathcal{D}_{(s_i,g_i)})$
    $\theta'_i \leftarrow \theta$
    for $k = 1$ **to** K **do** ▷ Inner Loop: Adaptation with $U^K_{(s_i,g_i)}$
      $\theta'_i \leftarrow \theta'_i - \alpha \nabla \mathcal{L}_{(s_i,g_i)}(\theta'_i, \text{sample}(\mathcal{D}^{train}_{(s_i,g_i)}))$
    end for
  end for
  $\theta \leftarrow \texttt{meta\_update}(\theta, \{\theta'_i\}, \{\mathcal{D}^{test}_i\}, \beta)$ ▷ Meta-update
end for

2. *Data filtering*: We follow the same procedure as GC-TTT (see Appendix A), and obtain a task-specific dataset $\mathcal{D}_{(s_i,g_i)}$.
3. The *inner loop* consists of K gradient steps of an offline GCRL loss on batches sampled from the training split, using SGD or other optimizers [Kingma and Ba, 2017], resulting in a specialized set of parameters θ'_i.
4. The *meta-update* combines the specialized parameters θ'_i with the base model θ. We instantiate three algorithms based on the meta-update rule:
   - **CONTINUAL-TTT:** $\theta \leftarrow \frac{1}{B}\sum_{i=1}^{B} \theta'_i$, updating the base parameters to the average of specialized parameters.
   - **TT-FOMAML:** $\theta \leftarrow \theta - \beta \frac{1}{B}\sum_{i=1}^{B} \nabla_{\theta'_i} \mathcal{L}(\theta'_i, \mathcal{D}^{test}_i)$, using first-order gradients of the test loss [Finn et al., 2017].
   - **TT-REPTILE:** $\theta \leftarrow \theta + \beta(\frac{1}{B}\sum_{i=1}^{B}(\theta'_i - \theta))$, merging specialized parameters with base parameters via interpolation [Nichol et al., 2018].

Test-Aware Meta-Update (Early-Stopping) Standard meta-update procedures assume that K gradient steps consistently yield an expert model superior to the base initialization. In practice, however, fixed-length adaptation risks inner-loop overfitting, vanishing gradients, or the accumulation of approximation errors [Antoniou et al., 2018]. To mitigate this, we additionally introduce a *test-aware* meta-update strategy that utilizes the test set $\mathcal{D}^{test}_{(s_i,g_i)}$ as an inner-loop validation mechanism. We monitor the test loss at each step $k \in [0, K]$ and perform the meta-update using the optimal checkpoint:

$$k^* = \arg\min_{k \in \{0,\dots,K\}} \mathcal{L}_{\mathcal{D}^{test}_{(s,g)}}(U^k_{(s,g)}(\theta)). \tag{9}$$

This approach provides two primary advantages. First, it *serves as a diagnostic tool* for the meta-learning process: $k^* = 0$ prevents updates from poor specialization by rejecting the step, $k^* = 1$ recovers standard stochastic gradient descent (SGD) for TT-REPTILE and CONTINUAL-TTT, and $k^* > 1$ confirms that the initialization has acquired a structure capable of capturing complex dependencies through extended optimization. Second, it *enhances hyperparameter robustness* by eliminating the need to tune K as a fixed parameter. By setting K to a large upper bound, the mechanism dynamically determines the optimal adaptation length per task, facilitating autonomous deployment. While test loss remains a proxy for final performance, we evaluate the effectiveness of early stopping in Section 5.2.

5 Experiments

Having described a framework for post-training goal-conditioned policies, and aligning them to test-time adaptation, we now turn to a broad empirical evaluation. Following [Bagatella et al., 2025], we will rely on a suite of goal-conditioned loco-navigation tasks from OGBench [Park et al., 2025a], each of which defines five goal-conditioned tasks as shown in Figure 2. We will cover

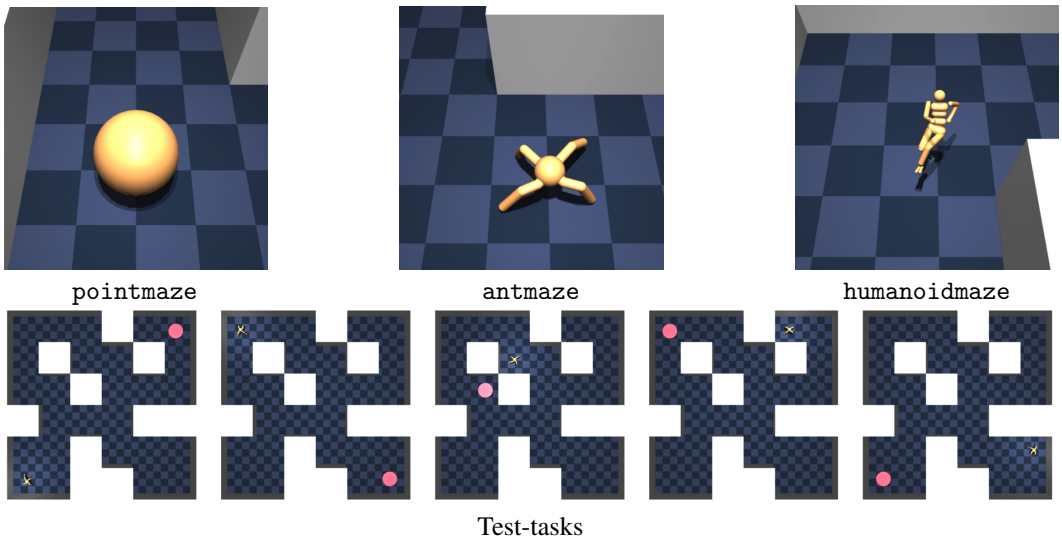

Figure 2: (Top) Maze environments from OGBench [Park et al., 2025a] representing increasing locomotion difficulty. (Bottom) The specific test-tasks used for evaluation.

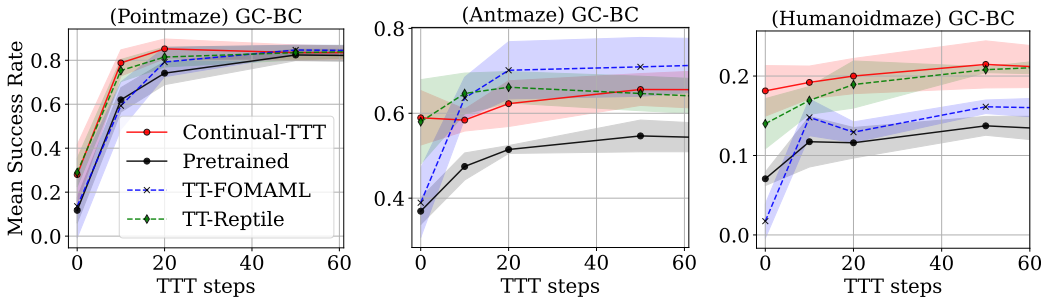

Figure 3: Meta-post-training improves zero-shot initialization and adaptation on high-quality datasets. Meta-learning variants consistently outperform standard GC-TTT baselines. TT-REPTILE and CONTINUAL-TTT provide superior starting success while TT-FOMAML optimizes for steeper recovery and higher peak performance despite initial zero-shot "dips."

three embodiments (`pointmaze`, `antmaze` and `humanoidmaze`) and two data sources: long, expert demonstrations (`navigate`) as well as short, disjoint trajectory segments (`stitch`).

In each environment, we apply META-GC-TTT on top of two representative offline GCRL algorithms, namely GCBC [Yang et al., 2022] (which is applied to expert `navigate` data) and GCIQL [Kostrikov et al., 2021], which we evaluate on `stitch` data due to its off-policy nature. We pre-train each algorithm for 400k steps (i.e., until convergence) to provide stable initializations for a subsequent 50k-step meta-post-training phase using TT-REPTILE, TT-FOMAML, or CONTINUAL-TTT. After post-training, we evaluate each policy in a zero-shot evaluation, as well as in combination with GC-TTT [Bagatella et al., 2025] across a range of test-time compute budgets. We report success rates averaged across the five standard evaluation tasks and three seeds. A detailed account of our checkpoint selection criteria and hyperparameter configurations is provided in Section 5.2 and Appendix F.

Our main experimental results are organized around three questions, which define the following subsections.

5.1 Does META-GC-TTT improve adaptation efficiency and performance?

High-quality Data and Supervised Learning Figure 3 reports success rates as a function of the number of test-time training (TTT) steps $K \in \{0 \text{ (zero-shot)}, 10, 20, 50\}$ when using GCBC as the underlying algorithm, and training on `navigate` data. Across the environments, we observe that

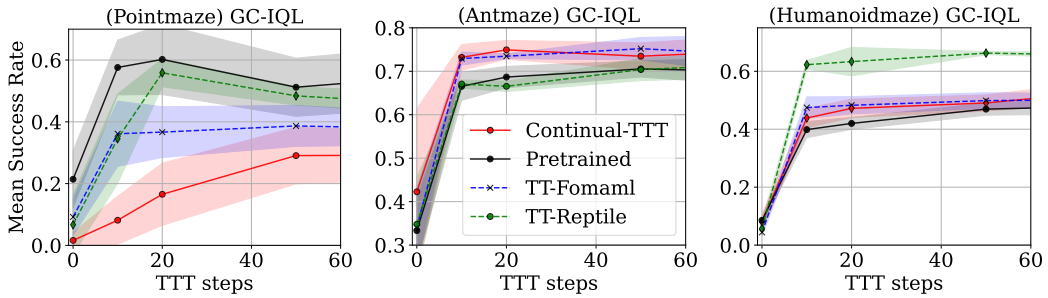

Figure 4: On mixed-quality datasets, meta-post-training positively impacts test-time training on two environments, while leading to instabilities on `pointmaze`.

meta-learned policies significantly improve both zero-shot performance and overall efficiency of the adaptation process.

Regarding zero-shot performance, we observe substantial improvements, particularly with TT-REPTILE and CONTINUAL-TTT. These meta-update rules implicitly optimize the base model as well as the adapted model for performance on training tasks; assuming the task distribution is not disjoint, this optimization generalizes effectively to similar test-time goals.

In contrast, TT-FOMAML specifically optimizes for adaptation potential rather than zero-shot success, typically providing the steepest relative increase in performance within the first few adaptation steps. While this objective can occasionally degrade zero-shot performance—as seen in `humanoidmaze`, where the initialization likely shifts toward a high-sensitivity region of the loss landscape - it facilitates recovery. On `antmaze`, TT-FOMAML achieves the largest absolute gains among all evaluated methods, reaching a peak success rate of 0.72 compared to the 0.55 baseline.

All three meta-update results translate into significant adaptation efficiency gains during deployment. In `pointmaze`, meta-post-trained models achieve an 80% success rate in as few as 10 to 20 steps, whereas the baseline requires 50 steps. This represents an approximate 3–$5\times$ gain in computational efficiency, demonstrating that meta-learning effectively aligns pre-trained policies for rapid online specialization.

Mixed Data and Off-policy Learning While results on GCBC demonstrate the potential of META-GC-TTT with expert data, standard offline RL assumes a more challenging regime: learning from suboptimal, mixed-quality datasets (e.g. `stitch`), which in turn demand algorithms capable of policy evaluation and improvement. To investigate this setting, we apply META-GC-TTT while using GCIQL [Kostrikov et al., 2021] as the underlying algorithm. Unlike GCBC, GCIQL requires goal-conditioned critics to model the optimality of actions; these estimators are trained with expectile regression or TD learning, introducing significant optimization instabilities. In Figure 4 we observe that meta-updates may fail to converge when applied to the highly filtered, off-policy batches encountered during the meta-training phase, as `pointmaze` shows. We further study this phenomenon in Appendix D, which shows that policy-only updates may circumvent these issues, and successfully meta-learn.

Beyond `pointmaze`, we observe that META-GC-TTT performs well out-of-the-box. On `antmaze`, CONTINUAL-TTT and TT-FOMAML consistently outperform the baseline across nearly all adaptation steps. In the high-dimensional `humanoidmaze`, TT-REPTILE emerges as the most effective method, significantly exceeding the performance of its counterparts. These results suggest that no single meta-learning algorithm is globally superior; rather, their efficacy depends on the specific dynamics and state-space complexity of the environment.

Notably, when trained on `stitch` datasets, meta-learned policies do not exhibit the same significant zero-shot improvements observed in the BC expert setting. We hypothesize that this may be related to the additional complexity introduced by learning a goal-conditioned critic from fragmented data.

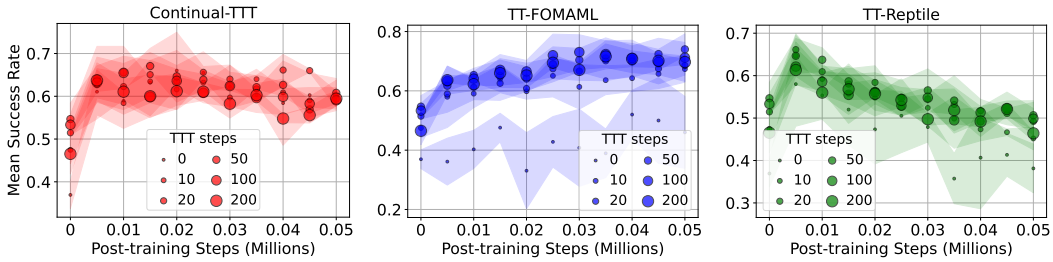

Figure 5: The test-time success rate evolution over post-training steps for different meta-update rules on `antmaze`. Performance curves are heavily dependent on the meta-algorithm used.

5.2 How much compute should be allocated for post-training?

While we have shown that meta-learning is beneficial for test-time performance and practical because it can be performed entirely offline, it remains computationally expensive. Therefore, this section studies trends in downstream performance as more compute is allocated to post-training. Figure 5 plots success rates for an increasing number of post-training iterations with a representative environment-algorithm pair (`antmaze`, GCBC) and each of the three meta-updates. In all cases, we observe that as little as 5k outer loops are sufficient to significantly boost zero-shot and TTT performance. At the same time, we also report three distinct behavioral patterns:

- CONTINUAL-TTT quickly drives zero-shot and TTT performance to a similar range, and generally maintains performance through post-training;
- TT-FOMAML exhibits gradual but slower improvement, potentially due to an ill-conditioned loss landscape;
- TT-REPTILE shows performance degradation over meta-learning steps, maintaining a gap between zero-shot and TTT success rates.

These results indicate that the relationship between optimization objectives and actual task success is non-trivial. The lack of a direct correlation suggests that minimizing surrogate loss does not consistently yield improvements in loco-navigation performance.

5.3 Does early-stopping ease the burden of hyperparameter tuning?

While the previous sections demonstrate the effectiveness of META-GC-TTT, the framework introduces several critical hyperparameters that govern the stability of meta-post-training and the efficiency of test-time adaptation. Appendix F provides a comprehensive report of these parameters, including inner/outer batch sizes and learning rates.

The number of inner-loop steps (K) is particularly crucial from a practical standpoint. A significant limitation in the applicability of standard optimization-based meta-learning is that K is typically fixed a priori rather than tuned per task. Our findings indicate that the choice of K during meta-post-training significantly impacts the Area Under the Success Curve (AUC, Appendix E).

Table 1 illustrates this trade-off: simply increasing K does not guarantee superior performance. Notably, TT-FOMAML exhibits a "collapse" as K increases, with success rates dropping to zero. This is likely because the first-order approximation becomes increasingly inaccurate as error accumulates over more inner steps, making the objective function overly complex on long horizons.

To lift the burden of choosing K, we test our proposed Early-Stopping strategy (ref. Section 4), which selects the optimal checkpoint within a 200-step window based on test loss. As reported in Table 1, this strategy consistently recovers performance lost to overfitting for both CONTINUAL-TTT and TT-REPTILE. Interestingly, TT-FOMAML benefits less from early stopping. In this case, the zero-shot model begins with a higher initial loss; while inner-loop optimization successfully reduces this loss, the resulting meta-update remains suboptimal. This suggests that the performance bottleneck for TT-FOMAML resides within the inherent limitations of the MAML meta-update approximation rule. Ultimately, these results indicate that the ideal TTT horizon is highly task-dependent: while

Table 1: Inner-loop steps comparison. All numbers reported are Area Under Success Curve for TTT scaling curves. For TT-REPTILE and CONTINUAL-TTT, early-stopping retains performance by avoiding overfitting and keeping expert models that minimize the test loss.

	Inner-loop Steps						
Method	**1**	**10**	**20**	**50**	**100**	**200**	**200 Early-stopping**
`antmaze`							
TT-REPTILE	0.707	0.690	0.683	0.651	0.659	0.652	0.685
TT-FOMAML	0.602	0.560	0.449	0.417	0.358	0.000	0.404
CONTINUAL-TTT	0.732	0.688	0.658	0.636	0.529	0.423	0.607
`humanoidmaze`							
TT-REPTILE	0.623	0.630	0.634	0.641	0.636	0.533	0.624
TT-FOMAML	0.305	0.186	0.066	0.005	0.045	0.000	0.000
CONTINUAL-TTT	0.466	0.483	0.510	0.511	0.408	0.248	0.518

complex goals require sustained refinement, simpler tasks risk performance degradation through behavioral drift if optimized for too many steps.

6 Conclusion

This work investigates the synergy between meta-learning and test-time training (TTT) in the context of offline goal-conditioned reinforcement learning to bridge the gap between large-scale pre-training and task-specific specialization. Our empirical evaluation provides strong evidence that meta-learning-based post-training, formalized through the META-GC-TTT framework, serves as a powerful alignment tool that prepares pre-trained models for rapid adaptation and enhances overall adaptation efficiency. Additionally, we proposed an early-stopping mechanism that maintains robustness by preventing the model from overfitting to suboptimal data batches, and eliminates the need to tune the number of inner steps as a fixed hyperparameter.

Limitations and Future Work Several promising avenues remain for the development of adaptive agents. A deeper investigation into the optimization dynamics of specific meta-algorithms could yield more robust hyperparameter selection strategies, particularly regarding the trade-off between inner-loop learning rates and outer-loop stability. Furthermore, combining META-GC-TTT with hierarchical algorithms (e.g., SAW [Zhou and Kao, 2026]) could mitigate scaling bottlenecks associated with long horizons [Park et al., 2025b] by adapting high-level planners alongside low-level controllers.

References

Siddhant Agarwal, Ishan Durugkar, Peter Stone, and Amy Zhang. f-policy gradients: A general framework for goal-conditioned rl using f-divergences. In *NeurIPS*, 2023.

Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Kevin Black, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Jared DiCarlo, Danny Driess, et al. π* 0.6: a vla that learns from experience. *arXiv preprint arXiv:2511.14759*, 1, 2025.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *NeurIPS*, 2017.

Antreas Antoniou, Harrison Edwards, and Amos Storkey. How to train your maml. *arXiv preprint arXiv:1810.09502*, 2018.

Marco Bagatella, Mert Albaba, Jonas Hübotter, Georg Martius, and Andreas Krause. Test-time offline reinforcement learning on goal-related experience, 2025. URL `https://arxiv.org/abs/2507.18809`.

Lucas Beyer, Andreas Steiner, André Susano Pinto, Alexander Kolesnikov, Xiao Wang, Daniel Salz, Maxim Neumann, Ibrahim Alabdulmohsin, Michael Tschannen, Emanuele Bugliarello, Thomas

Unterthiner, Daniel Keysers, Skanda Koppula, Fangyu Liu, Adam Grycner, Alexey Gritsenko, Neil Houlsby, Manoj Kumar, Keran Rong, Julian Eisenschlos, Rishabh Kabra, Matthias Bauer, Matko Bošnjak, Xi Chen, Matthias Minderer, Paul Voigtlaender, Ioana Bica, Ivana Balazevic, Joan Puigcerver, Pinelopi Papalampidi, Olivier Henaff, Xi Xiong, Radu Soricut, Jeremiah Harmsen, and Xiaohua Zhai. Paligemma: A versatile 3b vlm for transfer, 2024. URL https://arxiv.org/abs/2407.07726.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control, 2026. URL https://arxiv.org/abs/2410.24164.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021. URL https://arxiv.org/abs/2010.11929.

Damien Ernst, Pierre Geurts, and Louis Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6, 2005.

Benjamin Eysenbach, Tianjun Zhang, Sergey Levine, and Russ R Salakhutdinov. Contrastive learning as goal-conditioned reinforcement learning. *Advances in Neural Information Processing Systems*, 35:35603–35620, 2022a.

Benjamin Eysenbach, Tianjun Zhang, Ruslan Salakhutdinov, and Sergey Levine. Contrastive learning as goal-conditioned reinforcement learning. In *NeurIPS*, 2022b.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.

Chelsea Finn, Kelvin Xu, and Sergey Levine. Probabilistic model-agnostic meta-learning. *Advances in neural information processing systems*, 31, 2018.

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems*, 34:20132–20145, 2021.

Moritz Hardt and Yu Sun. Test-time training on nearest neighbors for large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=CNL2bku4ra.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization, 2025. URL https://arxiv.org/abs/2504.16054.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. Openvla: An open-source vision-language-action model, 2024. URL https://arxiv.org/abs/2406.09246.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. URL https://arxiv.org/abs/1412.6980.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning, 2021. URL https://arxiv.org/abs/2110.06169.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning, 2020. URL https://arxiv.org/abs/2006.04779.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning, 2019. URL https://arxiv.org/abs/1509.02971.

Hao Liu, Richard Socher, and Caiming Xiong. Taming maml: Efficient unbiased meta-reinforcement learning. In *International conference on machine learning*, pages 4061–4071. PMLR, 2019.

Minghuan Liu, Menghui Zhu, and Weinan Zhang. Goal-conditioned reinforcement learning: Problems and solutions, 2022. URL https://arxiv.org/abs/2201.08299.

Yecheng Jason Ma, Jason Yan, Dinesh Jayaraman, and Osbert Bastani. Offline goal-conditioned reinforcement learning via f-advantage regression. In *NeurIPS*, 2022.

Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

Seohong Park, Dibya Ghosh, Benjamin Eysenbach, and Sergey Levine. Hiql: Offline goal-conditioned rl with latent states as actions. *Advances in Neural Information Processing Systems*, 36:34866–34891, 2023.

Seohong Park, Kevin Frans, Sergey Levine, and Aviral Kumar. Is value learning really the main bottleneck in offline rl? *Advances in Neural Information Processing Systems*, 37:79029–79056, 2024.

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. *arXiv preprint arXiv:2506.04168*, 2025b.

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning, 2019. *URL https://arxiv. org/abs*, 1910.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021.

Aravind Rajeswaran, Chelsea Finn, Sham M Kakade, and Sergey Levine. Meta-learning with implicit gradients. *Advances in neural information processing systems*, 32, 2019.

Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *International conference on machine learning*, pages 9229–9248. PMLR, 2020.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, Gaël Liu, Francesco Visin, Kathleen Kenealy, Lucas Beyer, Xiaohai Zhai, Anton Tsitsulin, Robert Busa-Fekete, Alex Feng, Noveen Sachdeva, Benjamin Coleman, Yi Gao, Basil Mustafa, Iain Barr, Emilio Parisotto, David Tian, Matan Eyal, Colin Cherry, Jan-Thorsten Peter, Danila Sinopalnikov, Surya Bhupatiraju, Rishabh Agarwal, Mehran Kazemi, Dan Malkin, Ravin Kumar, David Vilar, Idan Brusilovsky, Jiaming Luo, Andreas Steiner, Abe

Friesen, Abhanshu Sharma, Abheesht Sharma, Adi Mayrav Gilady, Adrian Goedeckemeyer, Alaa Saade, Alex Feng, Alexander Kolesnikov, Alexei Bendebury, Alvin Abdagic, Amit Vadi, András György, André Susano Pinto, Anil Das, Ankur Bapna, Antoine Miech, Antoine Yang, Antonia Paterson, Ashish Shenoy, Ayan Chakrabarti, Bilal Piot, Bo Wu, Bobak Shahriari, Bryce Petrini, Charlie Chen, Charline Le Lan, Christopher A. Choquette-Choo, CJ Carey, Cormac Brick, Daniel Deutsch, Danielle Eisenbud, Dee Cattle, Derek Cheng, Dimitris Paparas, Divyashree Shivakumar Sreepathihalli, Doug Reid, Dustin Tran, Dustin Zelle, Eric Noland, Erwin Huizenga, Eugene Kharitonov, Frederick Liu, Gagik Amirkhanyan, Glenn Cameron, Hadi Hashemi, Hanna Klimczak-Plucińska, Harman Singh, Harsh Mehta, Harshal Tushar Lehri, Hussein Hazimeh, Ian Ballantyne, Idan Szpektor, Ivan Nardini, Jean Pouget-Abadie, Jetha Chan, Joe Stanton, John Wieting, Jonathan Lai, Jordi Orbay, Joseph Fernandez, Josh Newlan, Ju yeong Ji, Jyotinder Singh, Kat Black, Kathy Yu, Kevin Hui, Kiran Vodrahalli, Klaus Greff, Linhai Qiu, Marcella Valentine, Marina Coelho, Marvin Ritter, Matt Hoffman, Matthew Watson, Mayank Chaturvedi, Michael Moynihan, Min Ma, Nabila Babar, Natasha Noy, Nathan Byrd, Nick Roy, Nikola Momchev, Nilay Chauhan, Noveen Sachdeva, Oskar Bunyan, Pankil Botarda, Paul Caron, Paul Kishan Rubenstein, Phil Culliton, Philipp Schmid, Pier Giuseppe Sessa, Pingmei Xu, Piotr Stanczyk, Pouya Tafti, Rakesh Shivanna, Renjie Wu, Renke Pan, Reza Rokni, Rob Willoughby, Rohith Vallu, Ryan Mullins, Sammy Jerome, Sara Smoot, Sertan Girgin, Shariq Iqbal, Shashir Reddy, Shruti Sheth, Siim Põder, Sijal Bhatnagar, Sindhu Raghuram Panyam, Sivan Eiger, Susan Zhang, Tianqi Liu, Trevor Yacovone, Tyler Liechty, Uday Kalra, Utku Evci, Vedant Misra, Vincent Roseberry, Vlad Feinberg, Vlad Kolesnikov, Woohyun Han, Woosuk Kwon, Xi Chen, Yinlam Chow, Yuvein Zhu, Zichuan Wei, Zoltan Egyed, Victor Cotruta, Minh Giang, Phoebe Kirk, Anand Rao, Kat Black, Nabila Babar, Jessica Lo, Erica Moreira, Luiz Gustavo Martins, Omar Sanseviero, Lucas Gonzalez, Zach Gleicher, Tris Warkentin, Vahab Mirrokni, Evan Senter, Eli Collins, Joelle Barral, Zoubin Ghahramani, Raia Hadsell, Yossi Matias, D. Sculley, Slav Petrov, Noah Fiedel, Noam Shazeer, Oriol Vinyals, Jeff Dean, Demis Hassabis, Koray Kavukcuoglu, Clement Farabet, Elena Buchatskaya, Jean-Baptiste Alayrac, Rohan Anil, Dmitry, Lepikhin, Sebastian Borgeaud, Olivier Bachem, Armand Joulin, Alek Andreev, Cassidy Hardin, Robert Dadashi, and Léonard Hussenot. Gemma 3 technical report, 2025. URL `https://arxiv.org/abs/2503.19786`.

Stephen Tian, Suraj Nair, Frederik Ebert, Sudeep Dasari, Benjamin Eysenbach, Chelsea Finn, and Sergey Levine. Model-based visual planning with self-supervised functional distances. In *ICLR*, 2021.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL `https://arxiv.org/abs/2505.09388`.

Rui Yang, Yiming Lu, Wenzhe Li, Hao Sun, Meng Fang, Yali Du, Xiu Li, Lei Han, and Chongjie Zhang. Rethinking goal-conditioned supervised learning and its connection to offline rl. In *10th International Conference on Learning Representations, ICLR 2022*, 2022.

Zilai Zeng, Ce Zhang, Shijie Wang, and Chen Sun. Goal-conditioned predictive coding for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 36:25528–25548, 2023.

Chongyi Zheng, Ruslan Salakhutdinov, and Benjamin Eysenbach. Contrastive difference predictive coding. *arXiv preprint arXiv:2310.20141*, 2023.

John L. Zhou and Jonathan C. Kao. Flattening hierarchies with policy bootstrapping, 2026. URL `https://arxiv.org/abs/2505.14975`.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, Quan Vuong, Vincent Vanhoucke, Huong Tran, Radu Soricut, Anikait Singh, Jaspiar Singh, Pierre Sermanet, Pannag R Sanketi, Grecia Salazar, Michael S Ryoo, Krista Reymann, Kanishka Rao, Karl Pertsch, Igor Mordatch, Henryk Michalewski, Yao Lu, Sergey Levine, Lisa Lee, Tsang-Wei Edward Lee, Isabel Leal, Yuheng Kuang, Dmitry Kalashnikov, Ryan Julian, Nikhil J Joshi, Alex Irpan, brian ichter, Jasmine Hsu, Alexander Herzog, Karol Hausman, Keerthana Gopalakrishnan, Chuyuan Fu, Pete Florence, Chelsea Finn, Kumar Avinava Dubey, Danny Driess, Tianli Ding, Krzysztof Marcin Choromanski, Xi Chen, Yevgen Chebotar, Justice Carbajal, Noah Brown, Anthony Brohan, Montserrat Gonzalez Arenas, and Kehang Han. RT-2: Vision-language-action models transfer web knowledge to robotic control. In *7th Annual Conference on Robot Learning*, 2023. URL `https://openreview.net/forum?id=XMQgwiJ7KSX`.

A Goal-Conditioned Test-Time Training (GC-TTT)

This section provides a brief introduction to Goal-Conditioned Test-time Training (GC-TTT) [Bagatella et al., 2025]. GC-TTT involves two distinct operations: dataset filtering and policy updates.

The filtering procedure actively condition the data distribution of an offline dataset $\mathcal{D}$ on both the target goal g and the current state of the agent s, constructing $\mathcal{D}_{(s,g)}$. Trajectories τ in $\mathcal{D}$ are first filtered based on their proximity to the current agent state s, retaining only trajectories containing at least one state that is ϵ-close to s:

$$\texttt{is\_relevant}(\tau, s, \epsilon) \iff \min_{s' \in \tau} d(s, s') \leq \epsilon \tag{10}$$

Trajectories that pass the relevance filter are subsequently ranked according to an optimality score, selecting the *top-k* (or *top-$p\%$*) trajectories. In the absence of a learned value function, trajectories that do not reach the goal state (i.e. they have no states ϵ-close to g) are simply filtered out, and remaining trajectories are ranked based on a proxy to the goal distance (e.g. L2):

$$\texttt{critic\_free\_score}(\tau, g) = \max_{s' \in \tau} -d(s', g) \tag{11}$$

While computationally simple, this method requires the dataset to contain trajectories that successfully reach the goal. It is not applicable when optimal paths must be composed from many sub-optimal, fragmented segments. When a learned goal-conditioned value function $V(\cdot|g)\colon \mathcal{S} \to \mathbb{R}$ is available, every state in the relevant trajectories can be evaluated, and the trajectory score can be defined instead by the maximum value attained:

$$\texttt{critic\_based\_score}(\tau, g) = \max_{s' \in \tau} V(s'|g) \tag{12}$$

Among all trajectories surviving the relevance filter, we select the top-$p\%$ based on their optimality ranking. Crucially, we extract *sub-trajectories* from these samples, beginning at the state closest to s and ending at the state closest to g (or the state with the highest V-value).

Once $\mathcal{D}_{s,g}$ is constructed, it can be used to optimize near-arbitrary offline RL objectives [Yang et al., 2022, Kostrikov et al., 2021, Zhou and Kao, 2026].

B Test-time Loss Visualization

To understand the dynamics of TTT, we analyze how the loss varies w.r.t. test-time gradient steps of the actor and critic components of the Goal-Conditioned Implicit Q-Learning (GCIQL) training. Figure 6 illustrates these trends for a representative `antmaze stitch` task.

Across all evaluated learning rates, we observe a distinct discrepancy in optimization stability. The actor loss typically exhibits a smooth decrease, suggesting that the policy effectively refines its local behavior using the retrieved transitions. In contrast, the critic loss (and by extension, the total loss) is significantly more erratic. As the learning rate increases (e.g., 3×10^{-4}), the actor loss begins to lose its smoothness, yet it remains significantly more stable than the critic. This pattern suggests that while the agent can reliably improve its action selection at test time, the underlying value function is prone to noise that can potentially mislead the adaptation process.

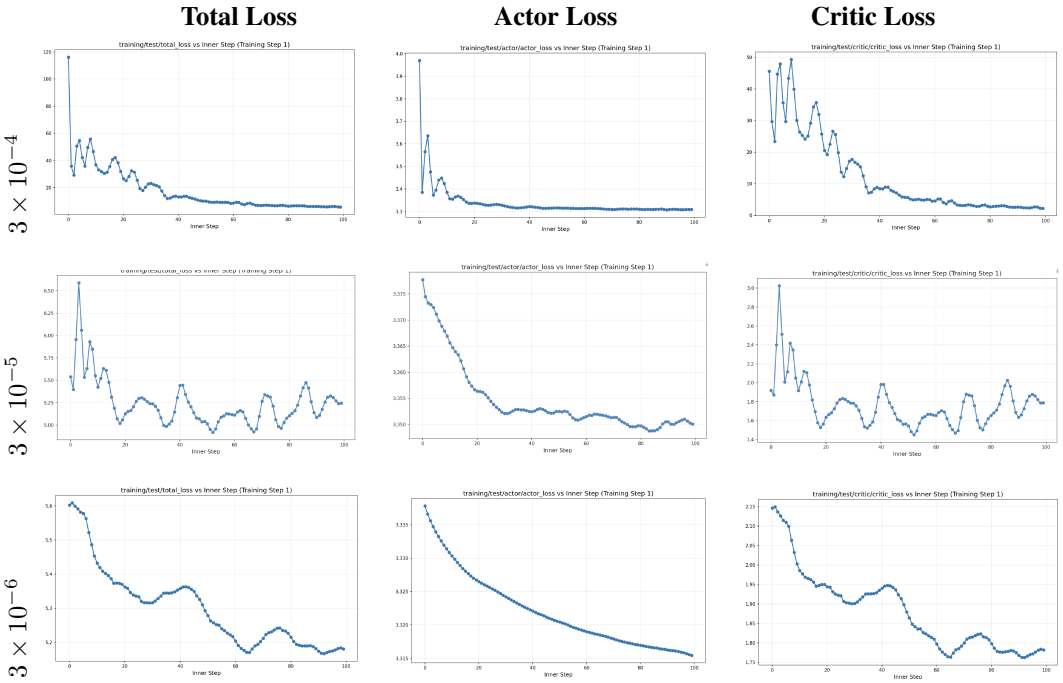

Figure 6: Comparison of total loss and its components, namely actor and critic losses, across multiple learning rates on a representative `antmaze stitch` task. The actor loss is typically smoother than the critic loss, which causes instabilities during META-GC-TTT.

C Actor-Only Test-time Training

The observed instability of the critic during TTT (Appendix B) suggests that updating the critic at test-time may be unnecessary or even detrimental. To test this hypothesis, we perform an ablation study comparing full adaptation (policy + critic) against policy-only updates. As shown in Table 2, updating the actor only retains the majority of performance gains, particularly in complex environments like `antmaze` and `humanoidmaze`.

Table 2: Area Under Success Curve (AUC) comparison: Full TTT (policy + critic) vs. actor-only TTT. Values represent the mean across 3 independent seeds.

Setting / Learning Rate	3×10^{-4}	5×10^{-4}	1×10^{-3}	3×10^{-3}	5×10^{-3}
`Pointmaze`					
TTT (Full Update)	0.21	0.28	0.33	0.41	0.53
TTT (Actor-Only)	0.22	0.24	0.27	0.32	0.35
`Antmaze`					
TTT (Full Update)	0.54	0.62	0.67	0.70	0.66
TTT (Actor-Only)	0.55	0.61	0.65	0.68	0.65
`Humanoidmaze`					
TTT (Full Update)	0.42	0.39	0.35	0.03	0.00
TTT (Actor-Only)	0.38	0.38	0.34	0.02	0.00

This indicates that the effectiveness of GC-TTT is largely driven by policy refinement and, as Park et al. [2024] claim, the policy extraction is the bottleneck in offline RL. This finding is of significant practical importance for the design of meta-learning pipelines: *by meta-post-training only the actor, we may circumvent convergence issues.*

D Addressing Pathologies in Off-Policy META-GC-TTT

An unexpected result in our initial evaluation was the failure of meta-learning algorithms to maintain baseline performance in `pointmaze stitch` (see Figure 4).

In accordance to our findings in Sections B and C, we investigate the significance of this degradation through a slight change of the inner-loop objective, specifically replacing the DDPG-BC loss with standard BC. As illustrated in Figure 7, META-GC-TTT shines in this setting, with TT-REPTILE and TT-FOMAML achieving near-ceiling success rates ($> 95\%$) within a handful of TTT steps, and maintaining this performance up to 200 steps. This further supports our original conjecture: the initial failures observed in Figure 4 likely stem from off-policy learning in specific environment-hyperparameter configurations, and on-policy corrections are likely to mitigate these issues.

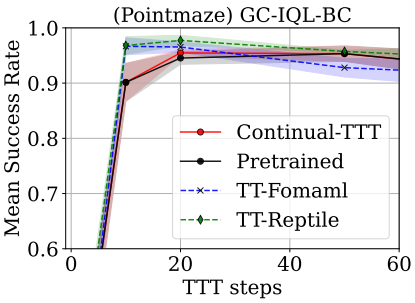

Figure 7: Switching the TTT objective to Behavioral Cloning (BC) on `pointmaze stitch` restores the effectiveness of meta-learning on top of a GCIQL backbone.

More broadly, these substantial performance gains align with the assertions of Park et al. [2024], suggesting that the primary bottleneck in offline RL is often policy extraction rather than value learning. META-GC-TTT serves as a robust mechanism for this extraction, leveraging pre-trained representations and goal-conditioned value functions to maximize task success.

E Success Metrics

To rigorously evaluate the adaptation performance of our models, we employ two primary statistical tools: standard error estimation for visualizing uncertainty and the normalized area under the success curve for summarizing adaptation efficiency.

Uncertainty Estimation and Error Bars To provide a robust estimate of performance across different runs, we aggregate results over multiple random seeds at each test-time training (TTT) budget. For a given configuration with n valid seeds, let s_i denote the success rate for seed i. The mean success $\hat{\mu}$ is calculated as:

$$\hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} s_i.$$

The uncertainty displayed in our plots represents the seed-wise dispersion, estimated via the standard error (SE):

$$\text{SE} = \frac{\hat{\sigma}}{\sqrt{n}}, \quad \text{where} \quad \hat{\sigma} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (s_i - \hat{\mu})^2}.$$

To visualize the 95% confidence interval, we report the approximate confidence half-width as $1.96 \times \text{SE}$. This interval provides a measure of the reliability of the mean success rate, ensuring that the observed improvements are statistically consistent across independent experimental runs.

Area Under the Success Curve (AUC) While success rates at specific TTT budgets are informative, they do not fully capture the trade-off between computational cost and agent efficacy. We therefore utilize the *normalized area under the success curve* (AUC) to provide a single-scalar summary of adaptation performance. Given a set of points (x_k, y_k), where x_k represents the number of `TTT_steps` and y_k represents the corresponding `mean_success`, the AUC is computed using the trapezoidal rule:

$$\text{AUC} = \frac{1}{x_{\max}} \sum_{k=1}^{K-1} \frac{y_k + y_{k+1}}{2} (x_{k+1} - x_k),$$

where $x_{\max}$ is the maximum number of adaptation steps. This normalization yields a unitless score bounded between 0 and 1. A higher AUC reflects a dual benefit: it indicates both how quickly the

model specializes (adaptation efficiency) and the peak performance level achieved within the given budget.

F Hyperparameter Choice and Ablations

Hyperparameter Choice For all hyperparameters not related to meta-learning, we use the default settings as reported by [Park et al., 2025a] and [Bagatella et al., 2025]. META-GC-TTT introduces several hyperparameters:

- **Learning Rates (α, β):** The inner learning rate (α) is the primary driver of test-time improvement. We find that the optimal α for meta-learning might differ from the one used at test-time as it will affect post-training dynamics (see Figure 8). The meta-learning rate (β) is algorithm-specific: CONTINUAL-TTT requires none, TT-FOMAML uses an meta-gradient learning rate (swept in $\{3 \times 10^{-4}, 3 \times 10^{-5}\}$), and TT-REPTILE uses $\beta \in \{0.01, 0.1, 0.2, 0.5\}$ as a weight interpolation coefficient.
- **Batch Sizes (B_{inner}, B):** To ensure a fair comparison and minimize the search space, we fix the inner batch size to $B_{inner} = 1024$, following Park et al. [2025a]. To optimize memory overhead we fix the meta-batch size $B = 1$. This is crucial for scaling as avoids storing multiple copies of the model during the outer loop.
- **Number of Inner Steps (K):** Determining the optimal K is challenging: a high K risks overfitting on simple tasks, while a low K may be ineffective on complex navigation goals. We conduct an extensive sweep $K \in \{1, 10, 20, 50, 100, 200\}$ and evaluate our **Early-Stopping** strategy fixing $K = 100$ for expert datasets and $K = 200$ for stitch datasets.
- **Meta-Steps (N):** We fix the post-training duration to $N = 50k$ steps, which we empirically found to be effective to improve performance with the tested datasets.

Table 3 reports all settings-specific hyper-parameters.

F.1 Inner Learning Rate Ablation

Our results indicate that the inner learning rate (α) used during both test-time training and meta-post-training is the most critical factor for stability. While a high α is beneficial to increase success rate at test time (see Table 4), it frequently induces instability during the meta-learning process, ultimately resulting in a degraded initialization.

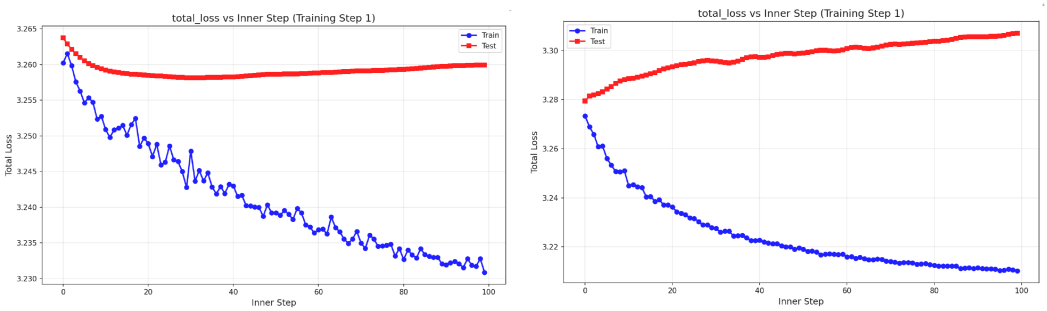

Figure 8: Visualization of the train and test loss of a task during post-training with various learning rates: bigger on the left (3×10^{-4}), smaller on the right (3×10^{-5})

.

As illustrated in Figure 8, high α causes the model to overfit rapidly to the train batches encountered. While this over-specialization is beneficial test-time performance, it is detrimental for meta-updating the model. We empirically observe that models trained with higher learning rates exhibit lower training losses but higher test losses throughout the post-training phase. Even when utilizing an early-stopping strategy to mitigate this overfitting, the resulting updates are often rejected, thereby stalling meta-learning progress.

Algorithm	Actor loss	Inner LR	Meta LR	Inner steps (K)	TTT LR	Checkpoint	Early stop
Dataset: `pointmaze_expert`							
CONTINUAL-TTT	BC	3×10^{-5}	3×10^{-4}	100	3×10^{-4}	30k	Yes
TT-FOMAML	BC	3×10^{-5}	3×10^{-4}	100	3×10^{-4}	35k	Yes
TT-REPTILE	BC	3×10^{-5}	0.01	100	3×10^{-4}	45k	Yes
Dataset: `antmaze_expert`							
CONTINUAL-TTT	BC	3×10^{-5}	3×10^{-4}	100	3×10^{-4}	10k	Yes
TT-FOMAML	BC	3×10^{-5}	3×10^{-4}	100	3×10^{-4}	35k	Yes
TT-REPTILE	BC	3×10^{-4}	0.5	100	3×10^{-4}	5k	Yes
Dataset: `humanoidmaze_expert`							
CONTINUAL-TTT	BC	3×10^{-5}	3×10^{-4}	100	3×10^{-5}	25k	Yes
TT-FOMAML	BC	3×10^{-5}	3×10^{-4}	100	3×10^{-4}	40k	Yes
TT-REPTILE	BC	3×10^{-5}	0.5	100	3×10^{-5}	50k	Yes
Dataset: `pointmaze_stitch`							
CONTINUAL-TTT	BC	3×10^{-3}	3×10^{-4}	200	3×10^{-3}	30k	Yes
TT-FOMAML	BC	3×10^{-3}	3×10^{-4}	1	3×10^{-3}	5k	No
TT-REPTILE	BC	3×10^{-3}	1×10^{-4}	10	3×10^{-3}	40k	No
CONTINUAL-TTT	DDPG-BC	5×10^{-3}	3×10^{-4}	50	5×10^{-3}	35k	No
TT-FOMAML	DDPG-BC	5×10^{-3}	3×10^{-4}	10	5×10^{-3}	5k	No
TT-REPTILE	DDPG-BC	5×10^{-3}	0.1	1	5×10^{-3}	20k	No
Dataset: `antmaze_stitch`							
CONTINUAL-TTT	DDPG-BC	3×10^{-5}	3×10^{-4}	100	1×10^{-3}	25k	Yes
TT-FOMAML	DDPG-BC	3×10^{-5}	3×10^{-4}	100	1×10^{-3}	15k	Yes
TT-REPTILE	DDPG-BC	3×10^{-5}	0.1	100	1×10^{-3}	5k	Yes
Dataset: `humanoidmaze_stitch`							
CONTINUAL-TTT	DDPG-BC	3×10^{-4}	3×10^{-4}	20	3×10^{-4}	10k	No
TT-FOMAML	DDPG-BC	3×10^{-4}	3×10^{-4}	1	3×10^{-4}	50k	No
TT-REPTILE	DDPG-BC	3×10^{-4}	0.1	200	3×10^{-4}	10k	Yes

Table 3: Best Hyper-parameters for each dataset.

Table 4: A higher learning rate α is generally beneficial for GC-TTT. We report Area Under the Success Curve (AUC) over 3 seeds.

	Inner Learning Rate (α)				
Environment	$\mathbf{3 \times 10^{-5}}$	$\mathbf{1 \times 10^{-4}}$	$\mathbf{3 \times 10^{-4}}$	$\mathbf{1 \times 10^{-3}}$	$\mathbf{3 \times 10^{-3}}$
`pointmaze/stitch`	0.21	0.28	0.33	0.41	**0.53**
`antmaze/stitch`	0.54	0.62	0.67	**0.70**	0.66
`humanoidmaze/stitch`	0.44	0.48	**0.48**	0.40	0.03

G Pre-training Success Curves

Departing from the meta-learning analysis, Figure 9 illustrates test-time success rates across varying pre-training horizons. We observe diverse, environment-specific scaling behaviors that provide broader insights into offline GCRL performance. Notably, extending the pre-training budget beyond the 400k-step baseline used for our meta-learning initializations does not yield further gains.

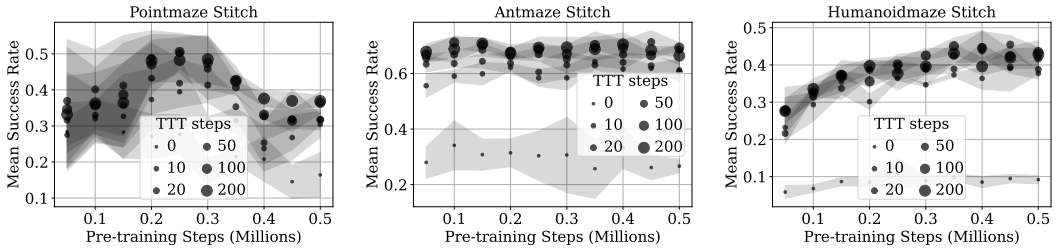

Figure 9: The test-time success rate evolution over pre-training steps is environment-dependent. On `pointmaze stitch` it starts degrading after 250k steps, on `antmaze stitch` it plateaus after 100k steps, while on `humanoidmaze stitch` converges at 350k steps. In all cases, 400k steps are sufficient for convergence.