CG-TTRL: Context-Guided Test-Time Reinforcement Learning for On-Device Large Language Models

Peyman Hosseini[1,2] * **Ondrej Bohdal**[1] **Taha Ceritli**[1] **Ignacio Castro**[2]
Matthew Purver[2] **Mete Ozay**[1] **Umberto Michieli**[1]

[1]Samsung R&D Institute UK [2]Queen Mary University of London
{s.hosseini, i.castro, m.purver}@qmul.ac.uk
{o.bohdal.1, t.ceritli, m.ozay, u.michieli}@samsung.com

Abstract

Test-Time Reinforcement Learning (TTRL) has shown promise in adapting foundation models for complex tasks at test time, resulting in large performance improvements. TTRL leverages an elegant two-phase sampling strategy: first, multi-sampling derives a pseudo-label via majority voting, while subsequent downsampling and reward-based fine-tuning encourages the model to explore and learn diverse valid solutions, with the pseudo-label modulating the reward signal. Meanwhile, In-Context Learning has been widely explored at inference time to enhance model performance without weight updates. However, TTRL's two-phase sampling strategy under-utilizes contextual guidance, which can potentially improve pseudo-label accuracy in the initial exploitation phase while regulating exploration in the second. To address this, we propose Context-Guided TTRL (CG-TTRL), integrating context dynamically into both sampling phases and propose a method for efficient context selection for on-device applications. Our evaluations on mathematical and scientific QA benchmarks show CG-TTRL outperforms TTRL (e.g. additional 7% relative accuracy improvement over TTRL), while boosting efficiency by obtaining strong performance after only a few steps of Test-Time Training (e.g. 8% relative improvement rather than 1% over TTRL after 3 steps).

1 Introduction

Personalizing foundation models to distribution shifts and domain-specific tasks (e.g., mathematical reasoning) with minimal computational overhead has become a popular area in Artificial Intelligence (AI) research. Traditional approaches to training language models often rely on static datasets and pre-defined objectives, limiting adaptability to new or complex tasks (Berglund et al., 2024). Test-Time Adaptation (TTA) addresses this challenge by enabling models to dynamically adjust to new tasks at inference (Sun et al., 2020; Niu et al., 2022; Zhou et al., 2025). In recent years, the integration of reinforcement learning (RL) with language models has gained significant attention for its potential to enhance reasoning and decision-making capabilities at test time (Ouyang et al., 2022; Zuo et al., 2025).

Lying within the aforementioned paradigms, Test-Time Reinforcement Learning (TTRL) (Zuo et al., 2025), aims to dynamically adapt to new tasks at test-time without any external supervision. It refines model outputs through iterative interactions with feedback mechanisms. However, TTRL faces challenges such as reliance on noisy reward signals and limited generalization, a common problem in RL-based algorithms (Jiang et al., 2023), hindering its adaptability in practice. Provision of context both in traditional RL setups such as in robotics (Benjamins et al., 2023) and in new RL applications in AI agents (Lee et al., 2023) has shown to significantly improve the performance of these approaches. The definition of context is broad in such settings. It can vary from additional information about the environment such as arm stiffness or gravity in robotics (Benjamins et al., 2023) or similar question-answer pairs in novel applications of RL where foundation models act as AI agents (Lee et al., 2023). The latter is widely explored by the Natural Language Processing (NLP) community under the name of In-Context Learning (ICL).

In-Context Learning (Brown et al., 2020) has been increasingly adopted to improve the performance of Large Language Models (LLMs) in complex tasks such as Mathematical Reasoning (Zhou et al., 2022; Agarwal et al., 2024), Code Generation (Patel et al., 2024), and Medical Di-

* Work done during internship at Samsung R&D Institute UK.

Transactions of the Association for Computational Linguistics, vol. 14, pp. 1883–1898, 2026. https://doi.org/10.1162/TACL.a.783
Action Editor: Xavier Carreras. Submission batch: 11/2025; Revision batch: 2/2026; Published 8/2026.

agnostics (Ferber et al., 2024) without altering model weights at inference time. Additionally, the context window of foundation models has been continuously expanding, with recent open-source models having context windows surpassing tens of thousands of tokens (Yang et al., 2025b; Abdin et al., 2024) or even hundreds of thousands of tokens (Liu et al., 2024; Gemma Team et al., 2025; Meta, 2025). These advancements enable inclusion of a large number of context examples or task demonstrations in the query. At the same time, previous research has shown unrepresentative or misleading examples can even cause performance degradation (Zhao et al., 2021; Min et al., 2022; Ye and Durrett, 2022; Hosseini et al., 2025). Thus, increasing the number of context examples alone does not usually lead to better performance.

Recent research has focused on advancing context selection algorithms (Dong et al., 2024; Mao et al., 2025), employing techniques such as contrastive learning (Gao and Das, 2024; Zhang et al., 2024) or iterative filtering (Li and Qiu, 2023) to identify high-impact examples. However, these methods often rely on auxiliary models or computationally intensive metrics, limiting their utility in resource-constrained settings. For Test-Time Adaptation (TTA), which requires dynamic context selection to guide learning, and for on-device LLMs, such overhead is impractical. Lightweight, storage-efficient algorithms are thus essential. Efficiency is critical for personalization tasks, where context must dynamically adapt to user patterns or domain shifts. Current techniques prioritize accuracy over efficiency, neglecting on-device deployment trade-offs. Our work bridges this gap with lightweight context selection mechanisms tailored for TTRL, ensuring scalable adaptation.

Contribution: Our paper's contributions can be summarized as follows:

- CG-TTRL framework (Figure 1): To our knowledge, we propose the first framework to integrate In-Context Learning (ICL) into an unsupervised fine-tuning pipeline, specifically enhancing TTRL's two-phase strategy. This approach dynamically leverages context to refine pseudo-labels during the exploitation phase (i.e., majority voting after multi-sampling) and regulate exploration phase (i.e., fine-tuning on diverse explored solutions leading to the pseudo-label), addressing a gap in self-supervised adaptation.

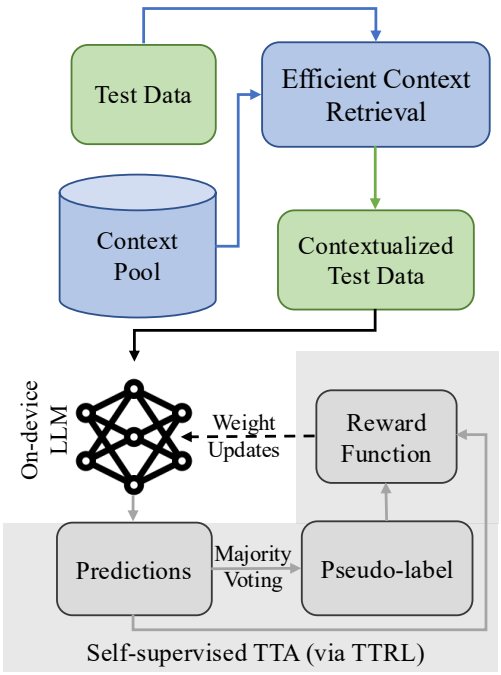

Figure 1: Our CG-TTRL approach improves TTRL by efficiently selecting context.

- Lightweight context selection: We design efficient context selection methods tailored for on-device deployment, prioritizing storage and computational efficiency without compromising adaptation quality.

- Empirical validation: Through evaluations on mathematical and scientific QA benchmarks, we demonstrate CG-TTRL's superior accuracy and efficiency compared to TTRL, even under low-resource constraints.

- On-device scalability: Our work establishes a paradigm for continuous, context-aware self-improvement, addressing the trade-off between efficiency and personalization..

2 Related Work

Test-Time Adaptation: Test-Time Adaptation (TTA) enables models to adjust to distribution shifts during inference. Self-supervised methods like entropy minimization (Wang et al., 2021; Gao et al., 2024) and pseudo-labeling (Goyal et al., 2022; Yu et al., 2024) refine predictions without labeled data. Test-Time Reinforcement Learning (TTRL) (Zuo et al., 2025) uses a two-phase sampling strategy to balance exploitation and exploration, computing rewards that are used to fine-tune the model. However, TTRL's reliance on noisy pseudo-labels and limited contextual guid-

ance restricts its robustness to various types of questions. Integrating contextual cues, inspired by robotics (Benjamins et al., 2023) and AI agents (Lee et al., 2023), has shown promise in stabilizing model adaptation. Our CG-TTRL framework addresses these gaps by dynamically incorporating context to refine pseudo-labels and regulate exploration, enhancing both accuracy and performance improvement speed for on-device applications.

Context Selection: In-Context Learning (ICL) relies on selecting representative examples to guide model predictions. Early approaches used random selection (Brown et al., 2020), while later methods employed semantic similarity via embeddings (Liu et al., 2022) or retrieval-augmented generation (Karpukhin et al., 2020). Recent techniques leverage contrastive learning (Gao and Das, 2024) and iterative filtering (Li and Qiu, 2023) to identify high-impact examples, but often require auxiliary models or intensive computations. For on-device deployment, efficiency is critical. Sparse attention mechanisms (Child et al., 2019) and locally-sensitive hashing (Kitaev et al., 2020) reduce computational overhead, while storage-efficient algorithms such as coreset selection (Coleman et al., 2019) optimize context retention. Our context selection prioritizes efficiency and feasibility using minimal compute and storage resources to dynamically curate relevant examples for continuous adaptation at test time.

On-device LLMs: Deploying LLMs on-device necessitates addressing computational and memory constraints (Hosseini et al., 2024), particularly for dynamic adaptation scenarios like Test-Time Adaptation (TTA). While traditional LLMs require high-end hardware for inference (Borzunov et al., 2023), on-device LLMs such as Llama 3.2 1B (Dubey et al., 2024), Gemma 3 1B (Gemma Team et al., 2025), and Qwen2.5 1.5B (Yang et al., 2025a) offer a balance between performance and efficiency, enabling localized processing of sensitive data (Dhar et al., 2021). These models leverage compression techniques to reduce footprint, yet face trade-offs in adaptability to shifting data distributions. As a standard practice, such foundation models are adapted for downstream tasks by adapters deployed on the device (Gunter et al., 2024; Dong et al., 2024; Hosseini et al., 2026). Adapter merging is common to support multi-tasking and save storage (Bohdal et al., 2025; Ceritli et al., 2025; Shenaj et al., 2025).

3 Methodology

3.1 Setup

We first summarize Test-Time Reinforcement Learning, which we extend and improve in our work. As part of TTRL, the user provides a query q, which often corresponds to a challenging reasoning problem, for which TTA is helpful. TTRL generates M predictions $\{\hat{y}_i\}_{i=1}^{M}$ to identify a pseudo-label y via majority voting. The next step is to compute a reward R that can be used to fine-tune the model with parameters θ. To compute the reward, $N < M$ already generated samples $\{\hat{y}'_i\}_{i=1}^{N}$ are selected and compared with the pseudo-label y. If the generated sample $\hat{y}'_i$ is the same as the pseudo-label y, the reward $R(\hat{y}'_i, y)$ becomes 1, otherwise 0. The objective is to maximize the expected reward by optimizing the parameters θ via gradient ascent on the computed reward.

3.2 Context-Guided TTRL

Our primary goal is to improve the performance of TTRL, while ensuring suitability for on-device use cases. Such scenarios require solutions that are efficient in terms of both storage and runtime. We design a simple technique that is computationally lightweight, does not require storing additional models and most importantly leads to significant improvements over the performance of TTRL. Algorithm 1 summarizes the full CG-TTRL procedure, which we describe in detail below.

We propose to extend TTRL via efficiently-selected context to improve its performance, and as it turns out, also the ability to obtain strong performance improvement quickly. Figure 2 provides a high-level comparison between our Context-Guided TTRL (CG-TTRL) and vanilla TTRL. We include C context examples $\{q_i^c, s_i^c, y_i^c\}_{i=1}^{C}$, where q_i^c denotes the example queries, with detailed step-by-step solutions s_i^c and final responses y_i^c. We use these examples to extend the query q as $q' = \{q_i^c, s_i^c, y_i^c\}_{i=1}^{C} \cup q$.

There may be a high number of examples relevant to a query. Thus, we develop an efficient method to identify the most promising ones. Our key idea is to utilize TF-IDF features to efficiently identify the most useful examples. TF-IDF score (Sparck Jones, 1988) consists of two components, the term frequency $\mathrm{TF}(t, e)$ and inverse document

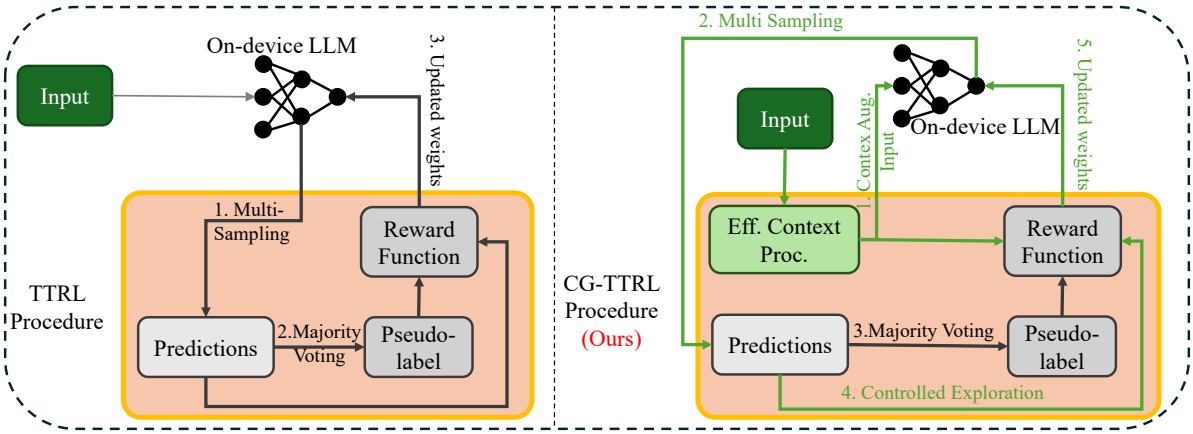

Figure 2: Comparison of vanilla TTRL (left) and our CG-TTRL that efficiently adds context (right).

frequency IDF(t) for term t and example document $e \in E_{\text{all}}$, where E_{all} denotes full context pool of available examples. TF(t, e) is computed as the number of times term t appears in example e out of the total number of terms in example e. IDF(t) is computed as the logarithm of the total number of examples in E_{all} over the number of examples in E_{all} containing term t. The TF-IDF score for term t and example e is then simply computed as TF$(t, e) \times$ IDF(t). We only extract the TF-IDF features for the query components and not for the responses or step-by-step solutions.

We assume available examples are stored as pairs of text (query, detailed solution and final response) and TF-IDF features for the query part of the text. When the user gives a new query q, we perform the following steps: 1) extract the TF-IDF features for query q: $f_{TF-IDF}(q)$, 2) find the most similar examples in the context pool in terms of TF-IDF features, 3) select top C examples in terms of similarity to include as context in the prompt.

We compute the similarity between query q and all K available queries $\{q_i^p\}_{i=1}^K$ in the context pool using cosine similarity as:

$$\text{sim}(q, q_i^p) = \cos(f_{TF-IDF}(q), f_{TF-IDF}(q_i^p)).$$

We then order the available queries in terms of the similarity and select the C queries with the largest similarity to query q. These queries from the overall context pool and their associated step-by-step solutions as well as final responses will form the context set $\{q_i^c, s_i^c, y_i^c\}_{i=1}^C$ that is combined with query q.

Figure 3 presents an overview of the described context selection mechanism and how it is used within the overall CG-TTRL approach.

Algorithm 1 Context-Guided TTRL (CG-TTRL)

Require: query q, context pool E_{all}, policy/model π_θ with parameters θ, context size C, samples M, sub-samples N

1: **Retrieve context:** compute $f_{TF\text{-}IDF}(q)$ and select top-C examples $\{q_i^c, s_i^c, y_i^c\}_{i=1}^C$ from E_{all} by cosine similarity
2: **Augment query:** form $q' = \{q_i^c, s_i^c, y_i^c\}_{i=1}^C \cup q$
3: **Multi-sample with context:** generate M predictions $\{\hat{y}_i\}_{i=1}^M$ from $\pi_\theta(\cdot \mid q')$
4: **Majority vote:** derive pseudo-label $y \leftarrow \text{mode}(\{\hat{y}_i\}_{i=1}^M)$
5: **Reward computation:** sample $N < M$ predictions $\{\hat{y}_i'\}_{i=1}^N$; set $R(\hat{y}_i', y) = 1$ if $\hat{y}_i' = y$, else 0
6: **Fine-tune:** update θ via gradient ascent on $\mathbb{E}[R(\hat{y}_i', y)]$ using context-augmented samples
7: **return** fine-tuned parameters θ

TF-IDF is a lightweight feature extraction approach as it only consists of a small number of counting operations without needing to store any additional models on the device. However, as our experimental evaluation shows, its performance is competitive with more complex and significantly less efficient variations. Overall, our approach leads to an efficient yet well-performing way to improve TTRL in on-device applications.

See Figures 9 and 10 for a concrete end-to-end illustration of how TTRL and CG-TTRL process an individual query across all stages, showing a case where TTRL's majority vote fails, while CG-TTRL's context-enriched prompt succeeds.

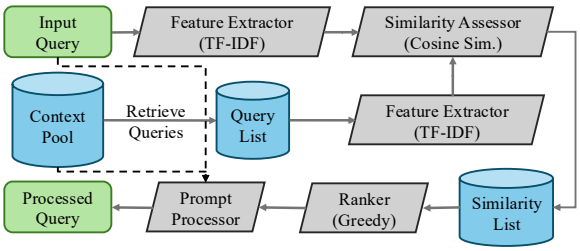

Figure 3: Our CG-TTRL approach efficiently finds the most similar queries in the context pool and uses them to modify the prompt.

4 Evaluation

4.1 Evaluation Details

Datasets: We have tested and analysed our solution on a number of challenging reasoning benchmarks, in particular MATH-500 (Hendrycks et al.), GSM-100 (Cobbe et al., 2021), AMC (AIMO, 2024), AIME 2024 (Maxwell-Jia, 2024) and GPQA (Rein et al., 2024), following the paper introducing TTRL (Zuo et al., 2025). MATH-500, GSM-100, AMC, and AIME are mathematical reasoning benchmarks, while GPQA is a challenging Q&A benchmark that includes questions from other domains. We use MATH-500 and AMC datasets for fine-tuning the model (only one dataset is used for fine-tuning in one experiment) and also consider out-of-distribution evaluation on the remaining datasets. Evaluation on GPQA corresponds to a case of a large domain shift, while the other mathematical reasoning benchmarks correspond to a milder domain shift, as the fine-tuning is done on a mathematical reasoning benchmark. Specifically, *MATH-500* is considered a *difficult* benchmark as it presents high-level mathematical reasoning and problem-solving proficiency, spanning Algebra, Combinatorics, Geometry, Number Theory, and Precalculus. *AIME*, is an *extremely difficult* benchmark at olympiad-level; it required deep mathematical ingenuity and multi-step reasoning. *AMC* is a high school competitive match benchmark. While it is a *difficult* benchmark, it is commonly considered easier than AIME since it consists of multiple-choice questions, allowing for potential elimination strategies as opposed to AIME. These three datasets differ in their difficulty level but are of quite a similar domain. *GSM-100* (a subset of GSM8K) is a benchmark of elementary and middle school-level word problems. It focuses on multi-step arithmetic, basic algebra, ratios, and percentages, and is considered to be of *moderate* difficulty. *GPQA* is an *extremely difficult* benchmark consisting of 198 expert-level (PhD level and above) scientific (not only mathematics) questions, significantly differing from other benchmarks domain. As evidenced by the embedding visualisation in Figure 4, from a quantitative perspective, our benchmark suite covers a range of domain shifts from mild (AMC, AIME, GSM-100, which share the mathematical reasoning domain with MATH-500) to substantial (GPQA, which represents graduate-level multi-domain Q&A with negligible vocabulary overlap with MATH-500). The performance is evaluated in terms of accuracy on the test samples. Context examples for GSM-100 and MATH-500 come from within these datasets. GSM-100 includes a set of 15 examples with detailed solutions, so we use these as the pool for selecting in-context examples. MATH-500 includes step-by-step solutions for all 500 examples. We consider all examples other than the current test one as the context pool. For the other three datasets (i.e., AMC, AIME, and GPQA), the datasets only come with the question and ground-truth answer and do not include a step-by-step solution. To address this, we consider the MATH-500 dataset as the context pool as it is the largest and most diverse (containing questions of different domains and difficulty levels in mathematics). Hence, we search for similar context examples from the MATH-500 dataset.

Models: We focus on models suitable for on-device deployment, in particular models with 1.5B parameters. For our main experiments we use Qwen2.5-Math-1.5B-Instruct (Yang et al., 2024) as an example of model specialized for mathematical reasoning, and Qwen2.5-1.5B-Instruct (Yang et al., 2025a; Qwen Team, 2024) and DeepSeek-R1-Distill (1.5B) (DeepSeek-AI, 2025) models as examples of more general-purpose models. For additional analyses we also consider Qwen2.5-Math-7B-Instruct (Yang et al., 2024) to study scaling to larger models, and Qwen2.5-Math-1.5B (Yang et al., 2024) to study the impact of using models that are not instruction-tuned.

Baselines: We compare with a vanilla zero-shot approach and with TTRL, which we extend as part of our CG-TTRL solution.

Hyperparameters: We select hyperparameters and other details for CG-TTRL training and evaluation based on performance on the validation set. In particular, based on the validation performance,

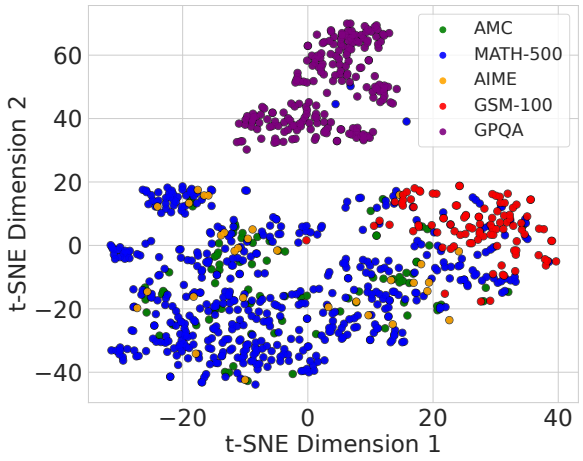

Figure 4: 2D visualization of dataset embeddings produced by Qwen3-Embedding-8B and projected with t-SNE. GPQA (purple) forms a clearly separated cluster from mathematical reasoning benchmarks, indicating a substantial domain shift. Other evaluation datasets (AMC, AIME, GSM-100) lie largely within or near the MATH-500 distribution, suggesting a comparatively milder shift. This pattern supports the interpretation that GPQA performance gains are unlikely to be explained by close feature-space overlap with MATH-500.

we select $C = 3$ as the number of context set examples when using MATH-500 dataset and $C = 5$ when using AMC dataset. Fine-tuning is done for 40 epochs (120 steps) on MATH-500 dataset, and 50 epochs (100 steps) for AMC, with Adam optimizer and learning rate of 5×10^{-7} for the actor model and 9×10^{-6} for the critic model, consistent with the original TTRL paper. The maximum input and generation length is set to 2048 tokens, taking into account limitations of on-device language models. Consistent with TTRL project, we use temperature $\tau = 1$ to balance between exploitation and exploration characteristics of the algorithm. We set $\tau = 0$ for the final stage, where we evaluate the performance of the models after fine-tuning. We sample $M = 32$ (64 for the AMC dataset) outputs for the majority-voting stage and downsample $N = 16$ for the reward calculation and fine-tuning stage. We used 2×NVIDIA A100 80GB GPUs for each experiment.

Inference vs. Training Phase Clarification. We emphasise that all results reported in Table 1 and throughout our evaluation reflect *inference-phase* performance of the fine-tuned model, evaluated at temperature $\tau = 0$ (greedy decoding) after the Test-Time Training (TTT) phase, which in our setup is performed *offline* on each test query. Steps 1–3 in the TTRL pipeline (based on Figure 2) constitute the adaptation phase, while Step 4 is a standard inference call to the fine-tuned model, which is the stage where evaluation is done. Table 3 additionally quantifies the computational cost of the full TTT + inference pipeline, including time per iteration and total convergence time, confirming the practical suitability of our method for on-device deployment.

4.2 Experimental Results

Main Results: We report the main results in Table 1 (bold results highlight the best performance for each model). We use either MATH-500 or AMC datasets for Test-Time Training (TTT) and evaluate on queries from all considered datasets. Hence we include evaluation on both in-domain and out-domain examples, with the out-of-domain examples coming either from the other mathematical reasoning benchmarks or GPQA that represents a more substantial domain shift to non-mathematical reasoning. The results indicate strong improvements compared to both TTRL and the zero-shot approach where we use the model directly without any TTA. For example, DeepSeek-R1-Distill has on average performance of 37.2% without TTA, TTRL boosts the perfomance to 56.8% (relative improvements of 52.7%) and our CG-TTRL improves the performance further to 59.4% (relative improvement of 59.7%).

Domain Shift Performance: The results in Table 1 allow us to compare in-domain and out-of-domain performance. We observe that CG-TTRL is consistently beneficial across this full spectrum, with strong improvements even under large domain shift (GPQA: +40.7% relative over base model for Qwen2.5-Math-1.5B-Instruct, TTT on MATH-500). The clear separation of GPQA in the embedding space confirms that these gains cannot be attributed to feature overlap or data contamination. This confirms the model genuinely generalises through contextual guidance. In addition, from a benchmark difficulty perspective, we observe that we get the highest relative performance gain on AIME (+133.3%) and GPQA (+40.7%) for the Qwen2.5-MATH-1.5B-Instruct model. These two benchmarks are the ones that are generally considered as extremely challenging benchmarks compared to the MATH dataset,

	Name	MATH-500	GSM-100	AMC	AIME 2024	GPQA	Avg.
TTT on MATH-500	Qwen2.5-Math-1.5B-Instruct	72.8%	86.1%	48.2%	10.0%	27.3%	48.9%
	w/ TTRL	78.6%	91.7%	54.2%	20.0%	37.4%	56.4%
	Rel. $\Delta_{\text{TTRL/Base}}$	+8.0%	+6.5%	+12.4%	+100%	+37.0%	+15.3%
	w/ CG-TTRL	**79.2%**	**94.4%**	**56.6%**	**23.3%**	**38.4%**	**58.4%**
	Rel. $\Delta_{\text{CG-TTRL/Base}}$	+8.8%	+9.6%	+17.4%	+133.3%	+40.7%	+19.4%
	Rel. $\Delta_{\text{CG-TTRL/TTRL}}$	**+0.8%**	**+3.0%**	**+4.4%**	**+16.7%**	**+2.7%**	**+3.5%**
	Qwen2.5-1.5B-Instruct	54.6%	72.2%	21.7%	3.3%	28.3%	36.0%
	w/ TTRL	62.6%	83.3%	32.5%	**13.3%**	34.3%	45.2%
	Rel. $\Delta_{\text{TTRL/Base}}$	+14.6%	+15.4%	+49.8%	+303.0%	+21.2%	+25.6%
	w/ CG-TTRL	**64.0%**	**86.1%**	**36.1%**	10.0%	**34.8%**	**46.2%**
	Rel. $\Delta_{\text{CG-TTRL/Base}}$	+17.2%	+19.3%	+66.4%	+203.0%	23.0%	+28.3%
	Rel. $\Delta_{\text{CG-TTRL/TTRL}}$	**+2.2%**	**+3.4%**	**+11.1%**	**-24.8%**	**+1.5%**	**+2.2%**
	DeepSeek-R1-Distill	51.0%	77.8%	22.9%	3.3%	30.8%	37.2%
	w/ TTRL	75.6%	88.9%	57.8%	**23.3%**	38.4%	56.8%
	Rel. $\Delta_{\text{TTRL/Base}}$	+48.2%	+14.3%	+152.4%	+606.0%	+24.7%	+52.7%
	w/ CG-TTRL	**78.6%**	**97.2%**	**59.0%**	**23.3%**	**38.9%**	**59.4%**
	Rel. $\Delta_{\text{CG-TTRL/Base}}$	+54.1%	+24.9%	+157.6%	+606.0%	+26.3%	+59.7%
	Rel. $\Delta_{\text{CG-TTRL/TTRL}}$	**+4.0%**	**+9.3%**	**+2.1%**	**+0.0%**	**+1.3%**	**+4.6%**
TTT on AMC	Qwen2.5-Math-1.5B-Instruct	72.8%	86.1%	48.2%	10.0%	27.3%	48.9%
	w/ TTRL	75.4%	86.1%	**54.2%**	16.7%	33.8%	53.2%
	Rel. $\Delta_{\text{TTRL/Base}}$	+3.6%	+0.0%	+12.4%	+67.0%	+23.8%	+8.8%
	w/ CG-TTRL	**76.0%**	**91.7%**	**54.2%**	**20.0%**	**35.4%**	**55.5%**
	Rel. $\Delta_{\text{CG-TTRL/Base}}$	+4.4%	+6.5%	+12.4%	+100.0%	+29.7%	+13.5%
	Rel. $\Delta_{\text{CG-TTRL/TTRL}}$	**+0.8%**	**+6.5%**	**+0.0%**	**+19.8%**	**+4.7%**	**+4.3%**
	Qwen2.5-1.5B-Instruct	54.6%	72.2%	21.7%	3.3%	28.3%	36.0%
	w/ TTRL	56.4%	77.8%	30.1%	**10.0%**	30.8%	41.0%
	Rel. $\Delta_{\text{TTRL/Base}}$	+3.3%	+7.8%	+38.7%	+203.0%	+8.8%	+13.9%
	w/ CG-TTRL	**56.8%**	**83.3%**	**35.0%**	**10.0%**	**32.3%**	**43.5%**
	Rel. $\Delta_{\text{CG-TTRL/Base}}$	+4.0%	+15.4%	+61.3%	+203.0%	+14.1%	+20.8%
	Rel. $\Delta_{\text{CG-TTRL/TTRL}}$	**+0.7%**	**+7.1%**	**+16.3%**	**+0.0%**	**+4.9%**	**+6.1%**
	DeepSeek-R1-Distill	51.0%	77.8%	22.9%	3.3%	30.8%	37.2%
	w/ TTRL	67.4%	**87.7%**	42.2%	16.7%	33.3%	49.5%
	Rel. $\Delta_{\text{TTRL/Base}}$	+32.2%	+12.7%	+84.3%	+406.0%	+8.1%	+33.1%
	w/ CG-TTRL	**71.1%**	**87.7%**	**47.0%**	**20.0%**	**34.9%**	**52.1%**
	Rel. $\Delta_{\text{CG-TTRL/Base}}$	+39.4%	+12.7%	+105.2%	+506.1%	+13.3%	+40.1%
	Rel. $\Delta_{\text{CG-TTRL/TTRL}}$	**+5.5%**	**+0.0%**	**+11.4%**	**+19.8%**	**+4.8%**	**+5.3%**

Table 1: Main results: comparison of zero-shot, TTRL and our Context-Guided TTRL (CG-TTRL) on each task. Top block: Test-Time Training (TTT) on MATH-500. Bottom block: TTT on AMC.

which shows that after training on MATH dataset, we see a significant performance boost on datasets from more challenging data distributions.

Comparison with In-Context Learning (ICL): We study the impact of ICL on the base model without TTRL in the top section of Table 2. For consistency with our CG-TTRL, we select examples in the same way, using TF-IDF features. We see that while ICL improves the performance of the base model, its improvement is not as large as that of TTRL. Beyond quantitative results, the qualitative example in Figure 10 illustrate this advantage concretely. ICL provides relevant context at inference time, but it cannot benefit from the reward-driven adaptation of the TTT stage. CG-TTRL, however, uses context to improve both the quality of majority voting (producing more stable, accurate pseudo-labels) and the final inference step, resulting in higher overall performance.

Context Selection Strategies: We compare with

Name	MATH-500	GSM-100	AMC	AIME 2024	GPQA	Avg.
Base Model	72.8%	86.1%	48.2%	10.0%	27.3%	48.9%
w/ In-context Learning	72.3%	91.7%	50.6%	13.3%	28.3%	51.2%
Rel. $\Delta_{\text{ICL/Base}}$	-0.7%	+6.6%	+5.0%	+33.3%	+3.7%	+4.7%
w/ TTRL	78.6%	91.7%	54.2%	20.0%	37.4%	56.4%
Rel. $\Delta_{\text{TTRL/Base}}$	+8.0%	+6.5%	+12.4%	+100%	+37.0%	+15.3%
w/ CG-TTRL$_{\text{Random}}$	78.0%	**94.4**%	55.4%	20.0%	34.8%	56.5%
Rel. $\Delta_{\text{Random/Base}}$	+7.1%	+9.6%	+14.9%	+100.0%	+27.5%	+15.5%
Rel. $\Delta_{\text{Random/TTRL}}$	-0.8%	+2.9%	+2.2%	+0.0%	-6.9%	+0.2%
w/ CG-TTRL$_{\text{SBERT}}$	78.0%	**94.4**%	54.2%	20.0%	35.4%	56.4%
Rel. $\Delta_{\text{SBERT/Base}}$	+7.1%	+9.6%	+12.4%	+100.0%	+29.7%	+15.3%
Rel. $\Delta_{\text{SBERT/TTRL}}$	-0.8%	+2.9%	+0.0%	+0.0%	-5.3%	+0.0%
w/ CG-TTRL$_{\text{Hybrid}}$	77.6%	**94.4**%	56.6%	**23.3**%	36.9%	57.8%
Rel. $\Delta_{\text{Hybrid/Base}}$	+6.6%	+9.6%	+17.4%	+133.3%	+35.2%	+18.2%
Rel. $\Delta_{\text{Hybrid/TTRL}}$	-1.3%	+2.9%	+4.4%	+16.7%	-1.3%	+2.5%
w/ CG-TTRL$_{\text{TF-IDF + MMR}}$	79.0%	**94.4**%	**57.8**%	**23.3**%	37.4%	**58.4**%
Rel. $\Delta_{\text{TF-IDF + MMR/Base}}$	+8.5%	+9.6%	+19.9%	+133.3%	+37.0%	+19.4%
Rel. $\Delta_{\text{TF-IDF + MMR/TTRL}}$	+0.5%	+2.9%	+6.6%	+16.7%	+0.0%	+3.5%
w/ CG-TTRL$_{\text{TF-IDF}}$	**79.2**%	**94.4**%	56.6%	**23.3**%	**38.4**%	**58.4**%
Rel. $\Delta_{\text{TF-IDF/Base}}$	+8.8%	+9.6%	+17.4%	+133.3%	+40.7%	+19.4%
Rel. $\Delta_{\text{TF-IDF/TTRL}}$	+0.8%	+2.9%	+4.4%	+16.7%	+2.7%	+3.5%

Table 2: Comparison with In-Context Learning (ICL) and various context-selection strategies: Random, Sentence Bert (SBERT), hybrid, TF-IDF + MMR, and TF-IDF that we normally use as part of CG-TTRL. TTT on MATH-500 with Qwen2.5-Math-1.5B-Instruct model.

random selection of the examples and a number of more advanced strategies to select the context. These methods are less efficient than our solution based on TF-IDF and they include the following options. 1) Using Sentence BERT model to extract embeddings and then using these to find the most similar examples. 2) Hybrid solution that combines TF-IDF with Sentence BERT embeddings (TF-IDF provides weights for the Sentence BERT embeddings). 3) Combination of TF-IDF with Maximal Marginal Relevance (MMR) weights (Juseon-Do et al., 2025; Kapuriya et al., 2025), where we iteratively add the most relevant examples into the context set. We report the results for these other strategies for context selection in Table 2. The results indicate that Random selection and Sentence BERT embeddings lead to similar performance as vanilla TTRL, with hybrid solution leading to improvements but not as large as vanilla TF-IDF. Combination of TF-IDF features with MMR obtains the same performance as vanilla TF-IDF, suggesting that the advanced iterative selection may not be needed.

Model Scaling: While we focus mainly on on-device settings, we also tested the benefits of our method for larger models. In particular, we evaluate scaling of our method from Qwen2.5-Math-1.5B-Instruct model to its 7B variant. For these experiments we perform TTT on MATH-500 and evaluate across the different mathematical reasoning datasets. The results in Figure 5 show we can successfully use our CG-TTRL method also for larger models. TTRL leads only to relatively minor improvements over the larger base model, but with our solution the improvements are noticeable.

Number of Context Examples: We study how the number of in-context examples influences CG-TTRL performance in Figure 6, on a scenario with TTT on MATH-500, with evaluation across all datasets and using Qwen2.5-Math-1.5B-Instruct model. Results show a higher number of examples in general helps improve the performance, but after some point, the gain plateaus.

Efficiency Analysis: We analyse the efficiency of

	TTRL	CG-TTRL$_{\text{TF-IDF}}$
# Generated tokens	760.3	706.3
# Total tokens	850.0	1,687.0
Time per iter.	0.53h	0.58h
Convergence iters.	73.7	67.8
Convergence time	39.28h	39.22h
TTFT infer.	0.32s	0.39s
E2E infer.	6.43s	6.56s

Table 3: Efficiency analysis of CG-TTRL vs TTRL in terms of the number of generated tokens, total number of processed tokens, time per iteration, convergence iterations (iterations until near top performance), total time until near top performance. Time To First Token (TTFT) and End-to-End (E2E) inference assess the inference speed after TTT phase and mirror real-world deployment performance. Average across three models is reported. CG-TTRL converges faster than TTRL.

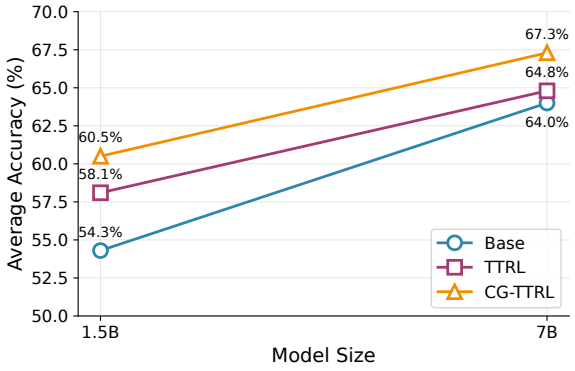

Figure 5: Model scaling analysis: our CG-TTRL method is useful also for larger models, e.g. ones with 7B parameters.

our solution compared to vanilla TTRL in Table 3. We consider a scenario where we use MATH-500 and report the average across the three models that we used for main evaluation. The results show that CG-TTRL leads to shorter generated outputs, even if the total number of processed tokens is larger due to the added context. The time per iteration is slightly longer for CG-TTRL, but it requires fewer iterations to reach near top performance. Overall, it takes slightly less time for CG-TTRL to reach near top performance than for TTRL. To further demonstrate the inference speed in a deployment setting, the last two rows in Table 3 report the average Time-To-First-Token and End-to-End inference time where Test-Time Training is done offline and at deployment time, the model is only used to do inference without any TTT.

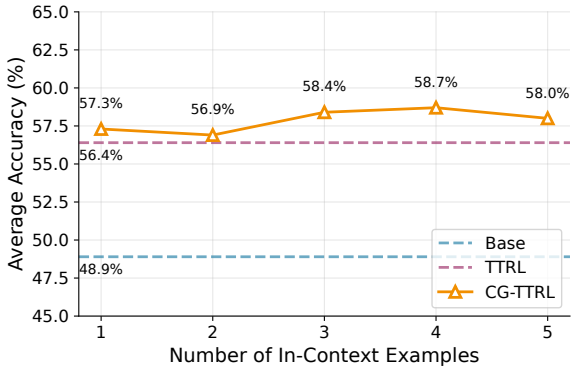

Figure 6: Number of context examples: using more examples is helpful in general, with consistent improvements of our CG-TTRL over both TTRL and the base model.

Few-epoch Analysis: In on-device scenarios it may be more practical to adapt the model only using a small number of epochs (or steps), so we analyse the behaviour of both CG-TTRL and TTRL when using between 1 and 5 epochs for TTT. We use the MATH-500 dataset and Qwen2.5-Math-1.5B-Instruct model for this analysis, reporting the average result across all datasets. In this case one epoch corresponds to doing three update steps. Figure 7 shows CG-TTRL leads to a large increase even when using only one epoch for TTT (+8% relative to zero-shot), unlike TTRL that improves more slowly (+1% relative after one epoch). This observation highlights the usefulness of CG-TTRL for on-device use cases.

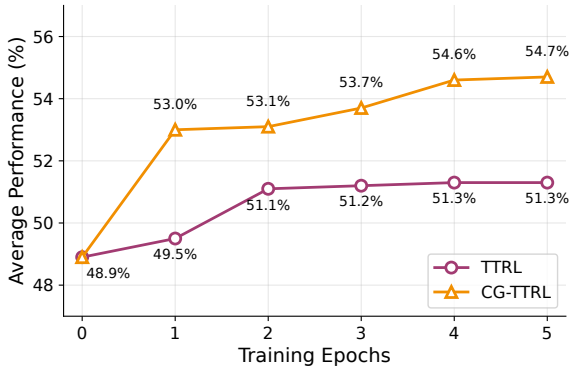

Figure 7: Few-epoch analysis: CG-TTRL significantly gains more than TTRL in this setup.

Training Dynamics: We analyse the training dynamics in Figure 8, using MATH-500 dataset and Qwen2.5-Math-1.5B-Instruct model. The analy-

Name	MATH-500	GSM-100	AMC	AIME 2024	GPQA	Avg.
Qwen2.5-Math-1.5B	34.6%	72.2%	33.7%	13.3%	26.3%	36.0%
w/ TTRL	74.2%	94.4%	51.8%	23.3%	35.4%	55.8%
Rel. $\Delta_{\text{TTRL/Base}}$	+114.5%	+30.7%	+53.7%	+75.2%	34.6%	+55.0%
w/ CG-TTRL	77.4%	94.4%	51.8%	23.3%	34.8%	56.3%
Rel. $\Delta_{\text{CG-TTRL/Base}}$	+123.7%	+30.7%	+53.7%	+75.2%	+32.3%	+56.4%
Rel. $\Delta_{\text{CG-TTRL/TTRL}}$	+4.3%	+0.0%	+0.0%	+0.0%	-1.7%	+0.9%
Qwen2.5-Math-1.5B-Instruct	72.8%	86.1%	48.2%	10.0%	27.3%	48.9%
w/ TTRL	78.6%	91.7%	54.2%	20.0%	37.4%	56.4%
Rel. $\Delta_{\text{TTRL/BaseTTRL}}$	+8.0%	+6.5%	+12.4%	+100%	+37.0%	+15.3%
w/ CG-TTRL	79.2%	94.4%	56.6%	23.3%	38.6%	58.4%
Rel. $\Delta_{\text{CG-TTRL/Base}}$	+8.8%	+9.6%	+17.4%	+133.3%	+41.4%	+19.4%
Rel. $\Delta_{\text{CG-TTRL/TTRL}}$	+0.8%	+3.0%	+4.4%	+16.7%	+3.2%	+3.5%

Table 4: Impact of CG-TTRL and TTRL on models trained with and without instruction tuning. As evident by the results, CG-TTRL works more effectively with instruction-tuned agents as these models can further utilize the provided context.

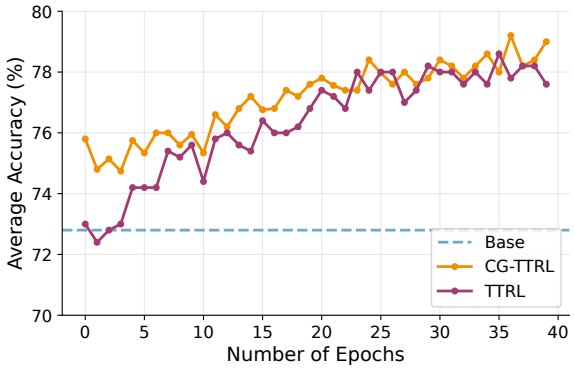

Figure 8: Training dynamics analysis: our CG-TTRL method leads to strong performance faster, but after significant amount of training, the performance of TTRL becomes closer.

sis shows that CG-TTRL is able to obtain significantly stronger performance than TTRL when only a small number of epochs is used for training. After larger number of epochs the performance gap becomes smaller, with CG-TTRL still outperforming TTRL by a certain margin.

Instruction Tuning: We compare the impact of CG-TTRL on models trained with and without instruction tuning in Table 4. CG-TTRL is useful for both, but emjoys more gains with instruction-tuned models that also perform better in general.

5 Limitations

In our paper we have focused on on-device scenarios and studied models that have suitable sizes for such use cases. As shown on the analysis with a 7B model, our method can benefit also larger models and it is likely the benefits would extend to even larger models. However, TTRL is computationally expensive, so such analysis would be challenging to conduct with larger models. A limitation of TTRL more broadly is that it typically uses a larger number of rollouts, resulting in significant time required for the adaptation. In fact, the experiments take substantial time and so running TTRL until convergence is not practical in on-device scenarios. In such cases it is realistic to only perform e.g. one epoch of training.

6 Conclusion

We have presented CG-TTRL technique that improves TTRL via efficiently selected context. Our solution has been designed to be especially suitable for on-device deployments, where efficiency and strong performance are crucial. Evaluation across a number of reasoning benchmarks has shown that CG-TTRL leads to noticeable improvements over TTRL's performance, while also obtaining strong performance improvement significantly faster. Relative improvements have been particularly large in the case of few-epoch adaptation where only a very small number of steps are used for the adaptation, a scenario especially suitable for on-device use cases.

References

Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J Hewett, Mojan Javaheripi, Piero Kauffmann, et al. 2024. Phi-4 technical report. *arXiv preprint arXiv:2412.08905*.

Rishabh Agarwal, Avi Singh, Lei Zhang, Bernd Bohnet, Luis Rosias, Stephanie Chan, Biao Zhang, Ankesh Anand, Zaheer Abbas, Azade Nova, et al. 2024. Many-shot in-context learning. *Advances in Neural Information Processing Systems*, 37:76930–76966.

AI-MO. 2024. AIMO validation AMC. Hugging Face Datasets. Apache-2.0 license.

Carolin Benjamins, Theresa Eimer, Frederik Schubert, Aditya Mohan, Sebastian Döhler, André Biedenkapp, Bodo Rosenhahn, Frank Hutter, and Marius Lindauer. 2023. Contextualize me–The case for context in reinforcement learning. *Transactions on Machine Learning Research*.

Lukas Berglund, Meg Tong, Maximilian Kaufmann, Mikita Balesni, Asa Cooper Stickland, Tomasz Korbak, and Owain Evans. 2024. The reversal curse: LLMs trained on "A is B" fail to learn "B is A". In *The Twelfth International Conference on Learning Representations*.

Ondrej Bohdal, Konstantinos Theodosiadis, Asterios Mpatziakas, Dimitris Filippidis, Iro Spyrou, Christos Zonios, Anastasios Drosou, Dimosthenis Ioannidis, Kyeng-Hun Lee, Jijoong Moon, et al. 2025. On-device system of compositional multi-tasking in large language models. *arXiv preprint arXiv:2510.13848*.

Alexander Borzunov, Max Ryabinin, Artem Chumachenko, Dmitry Baranchuk, Tim Dettmers, Younes Belkada, Pavel Samygin, and Colin A Raffel. 2023. Distributed inference and fine-tuning of large language models over the internet. *Advances in Neural Information Processing Systems*, 36:12312–12331.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901.

Taha Ceritli, Ondrej Bohdal, Mete Ozay, Jijoong Moon, Kyeng-Hun Lee, Hyeonmok Ko, and Umberto Michieli. 2025. HydraOpt: Navigating the efficiency-performance trade-off of adapter merging. *arXiv preprint arXiv:2507.17706*.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Cody Coleman, Christopher Yeh, Stephen Mussmann, Baharan Mirzasoleiman, Peter Bailis, Percy Liang, Jure Leskovec, and Matei Zaharia. 2019. Selection via proxy: Efficient data selection for deep learning. *arXiv preprint arXiv:1906.11829*.

DeepSeek-AI. 2025. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning.

Sauptik Dhar, Junyao Guo, Jiayi Liu, Samarth Tripathi, Unmesh Kurup, and Mohak Shah. 2021. A survey of on-device machine learning: An algorithms and learning theory perspective. *ACM Transactions on Internet of Things*, 2(3):1–49.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Baobao Chang, et al. 2024. A survey on in-context learning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 1107–1128.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The Llama 3 herd of models. *arXiv e-prints*, pages arXiv–2407.

Dyke Ferber, Georg Wölflein, Isabella C Wiest, Marta Ligero, Srividhya Sainath, Narmin Ghaffari Laleh, Omar SM El Nahhas, Gustav Müller-Franzes, Dirk Jäger, Daniel Truhn, et al. 2024. In-context learning enables multimodal large language models to classify cancer pathology images. *Nature Communications*, 15(1):10104.

Xiang Gao and Kamalika Das. 2024. Customizing language model responses with contrastive in-context learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18039–18046.

Zhengqing Gao, Xu-Yao Zhang, and Cheng-Lin Liu. 2024. Unified entropy optimization for open-set test-time adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 23975–23984.

Google Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al. 2025. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*.

Sachin Goyal, Mingjie Sun, Aditi Raghunathan, and J Zico Kolter. 2022. Test time adaptation via conjugate pseudo-labels. *Advances in Neural Information Processing Systems*, 35:6204–6218.

Tom Gunter, Zirui Wang, Chong Wang, Ruoming Pang, Andy Narayanan, Aonan Zhang, Bowen Zhang, Chen Chen, Chung-Cheng Chiu, David Qiu, et al. 2024. Apple intelligence foundation language models. *arXiv preprint arXiv:2407.21075*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. *Sort*, 2(4):0–6.

Peyman Hosseini, Ondrej Bohdal, Ahmed Alajrami, Andrea Maracani, Ignacio Castro, Matthew Purver, Mete Ozay, Savas Ozkan, and Taha Ceritli. 2026. DuoMem: Towards capable on-device memory agents via dual-space distillation. *arXiv preprint arXiv:2606.29961*.

Peyman Hosseini, Ignacio Castro, Iacopo Ghinassi, and Matthew Purver. 2025. Efficient solutions for an intriguing failure of LLMs: Long context window does not mean LLMs can analyze long sequences flawlessly. In *Proceedings of the 31st international conference on computational linguistics*, pages 1880–1891.

Peyman Hosseini, Mehran Hosseini, Ignacio Castro, and Matthew Purver. 2024. Cost-effective attention mechanisms for low resource settings: Necessity & sufficiency of linear transformations. *arXiv preprint arXiv:2403.01643*.

Yiding Jiang, J Zico Kolter, and Roberta Raileanu. 2023. On the importance of exploration for generalization in reinforcement learning. *Advances in Neural Information Processing Systems*, 36:12951–12986.

Juseon-Do Juseon-Do, Jaesung Hwang, Jingun Kwon, Hidetaka Kamigaito, and Manabu Okumura. 2025. Considering length diversity in retrieval-augmented summarization. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 2489–2500.

Janak Kapuriya, Manit Kaushik, Debasis Ganguly, and Sumit Bhatia. 2025. Exploring the role of diversity in example selection for in-context learning. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2962–2966.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick SH Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *EMNLP*, pages 6769–6781.

Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. In *International Conference on Learning Representations*.

Jonathan Lee, Annie Xie, Aldo Pacchiano, Yash Chandak, Chelsea Finn, Ofir Nachum, and Emma Brunskill. 2023. Supervised pretraining can learn in-context reinforcement learning. *Advances in Neural Information Processing Systems*, 36:43057–43083.

Xiaonan Li and Xipeng Qiu. 2023. Finding support examples for in-context learning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 6219–6235.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. 2024. DeepSeek-V3 technical report. *arXiv preprint arXiv:2412.19437*.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2022. What makes good in-context examples for GPT-3? *DeeLIO 2022*, pages 100–114.

Haitao Mao, Guangliang Liu, Yao Ma, Rongrong Wang, Kristen Johnson, and Jiliang Tang. 2025. A survey to recent progress towards understanding in-context learning. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 7302–7323.

Maxwell-Jia. 2024. AIME 2024. Hugging Face Datasets.

AI Meta. 2025. The Llama 4 herd: The beginning of a new era of natively multimodal intelligence. *Meta AI Blog*.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the role of demonstrations: What makes in-context learning work? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11048–11064.

Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Yaofo Chen, Shijian Zheng, Peilin Zhao, and Mingkui Tan. 2022. Efficient test-time model adaptation without forgetting. In *International Conference on Machine Learning*, pages 16888–16905. PMLR.

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

Arkil Patel, Siva Reddy, Dzmitry Bahdanau, and Pradeep Dasigi. 2024. Evaluating in-context learning of libraries for code generation. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 2908–2926.

Qwen Team. 2024. Qwen2.5: A party of foundation models.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. 2024. GPQA: A graduate-level Google-Proof Q&A benchmark. In *First Conference on Language Modeling*.

Donald Shenaj, Ondrej Bohdal, Taha Ceritli, Mete Ozay, Pietro Zanuttigh, and Umberto Michieli. 2025. K-Merge: Online continual merging of adapters for on-device large language models. *arXiv preprint arXiv:2510.13537*.

Karen Sparck Jones. 1988. *A statistical interpretation of term specificity and its application in retrieval*. Taylor Graham Publishing, GBR.

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

Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. 2021. Tent: Fully test-time adaptation by entropy minimization. In *International Conference on Learning Representations*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. 2025a. Qwen2.5 technical report. *arXiv preprint arXiv:2505.09388*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. 2025b. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu,

Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. 2024. Qwen2.5-Math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*.

Xi Ye and Greg Durrett. 2022. The unreliability of explanations in few-shot prompting for textual reasoning. *Advances in Neural Information Processing Systems*, 35:30378–30392.

Yeonguk Yu, Sungho Shin, Seunghyeok Back, Mihwan Ko, Sangjun Noh, and Kyoobin Lee. 2024. Domain-specific block selection and paired-view pseudo-labeling for online test-time adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22723–22732.

Jing Zhang, Hui Gao, Peng Zhang, Boda Feng, Wenmin Deng, and Yuexian Hou. 2024. LA-UCL: LLM-Augmented Unsupervised Contrastive Learning framework for few-shot text classification. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 10198–10207.

Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. In *International Conference on Machine Learning*, pages 12697–12706. PMLR.

Hattie Zhou, Azade Nova, Hugo Larochelle, Aaron Courville, Behnam Neyshabur, and Hanie Sedghi. 2022. Teaching algorithmic reasoning via in-context learning. *arXiv preprint arXiv:2211.09066*.

Xingzhi Zhou, Zhiliang Tian, Boyang Zhang, Yibo Zhang, Ka Chun Cheung, Simon See, Hao Yang, Yun Zhou, and Nevin L Zhang. 2025. Test-time adaptation on noisy data via model-pruning-based filtering and flatness-aware entropy minimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 10852–10860.

Yuxin Zuo, Kaiyan Zhang, Li Sheng, Shang Qu, Ganqu Cui, Xuekai Zhu, Haozhan Li, Yuchen Zhang, Xinwei Long, Ermo Hua, et al. 2025. TTRL: Test-Time Reinforcement Learning. *Advances in Neural Information Processing Systems*, 38.

TTRL End-to-End Example

Step 1: Prompt Formulation

System Prompt: You are an expert mathematical/scientific problem solver. When presented with a problem, follow these guidelines: (1) Think step-by-step, (2) Explain each step of your reasoning, (3) Use proper scientific notation and terminology, (4) adapt to context if example problems are provided before the main question, and (5) format your final answer inside boxed tags clearly.
Problem: Every morning Aya goes for a 9-kilometer-long walk and stops at a coffee shop afterwards. When she walks at a constant speed of s kilometers per hour, the walk takes her 4 hours, including t minutes spent in the coffee shop. When she walks s+2 kilometers per hour, the walk takes her 2 hours and 24 minutes, including t minutes spent in the coffee shop. Suppose Aya walks at s+1/2 kilometers per hour. Find the number of minutes the walk takes her, including the t minutes spent in the coffee shop.
Final Prompt / Model Input: {System Prompt} + {Problem}

Step 2: Multi Sampling for Voting Phase
Sampling Multiple Responses (e.g., 8) in High Temperature:
Response 1: "We have: 9/s + t/60 = 4 and 9/(s+2) + t/60 = 2.4. Subtracting: 9/s - 9/(s+2) = 1.6. 9(s+2-s)/[s(s+2)] = 1.6. 18/[s(s+2)] = 1.6. s(s+2) = 11.25. s^2 + 2s - 11.25 = 0. s = 2.5 km/h. t = 60(4 - 9/2.5) = 24 min. At 3 km/h: time = 9/3 = 3 hours = 180 min. Total = 180 + 24 = 204 minutes. Answer: \boxed{204}" [Correct]
. . .
Response 8: "Setting up the problem: walking distance = 9 km. Case 1: speed = s km/h, total time = 4 hours including t min coffee. Case 2: speed = s+2 km/h, total time = 2.4 hours including t min coffee. Walking time equations: 9/s + t/60 = 4 and 9/(s+2) + t/60 = 2.4. Subtract: 9/s - 9/(s+2) = 1.6. 9(s+2-s)/[s(s+2)] = 1.6. 18/[s(s+2)] = 1.6. s(s+2) = 11.25. s^2 + 2s - 11.25 = 0. s = 2.5 km/h. t = 60(4 - 9/2.5) = 24 min. At 3 km/h: walking time = 3 hours = 180 min. Total = 180 min. Answer: \boxed{180}" [Incorrect - forgot to add coffee time]
Majority Voting on Answers: [204, 240, 180, 210, 180, 205, 204, 180] => Majority vote = 180
Rewards Based on Matching the Majority Vote: Rewards = [0, 0, 1, 0, 1, 0, 0, 1].

Step 3: Sub-sampling for Test-Time Training Phase After the rewards are assigned based on matching with the majority vote in the previous step, in the third step, the TTRL phase randomly subsamples a subset of the responses generated in the previous step randomly (e.g., 4 samples). E.g.: [(Response 1, Rewards[1]), (Response 3, Rewards[1]), (Response 4, Rewards[4]), (Response 7, Rewards[7])]. At this fine-tuning stage, some RL algorithm such as PPO is used to encourage the agent to take actions that lead to the majority vote outcome and discourage the agent from taking actions that end up in other results.

Step 4: Updated Inference Stage After Test-Time Training Stage
Finally, after step 3, we prompt the model with the same {Model Input} created at the end of Step 1 and ask it to solve the task at a lower temperature (e.g. $\tilde{0}.3$). Expectedly, the model becomes more likely to generate the output leading to the majority vote at this stage.
Fine-tuned Model Output: "This is a rate problem with two scenarios. Let w = walking time in hours. Scenario 1: w + t/60 = 4 where w = 9/s. Scenario 2: w' + t/60 = 2.4 where w' = 9/(s+2). Subtracting: 9/s - 9/(s+2) = 1.6. 9(s+2-s)/[s(s+2)] = 1.6. 18/[s(s+2)] = 1.6. s(s+2) = 11.25. s^2 + 2s - 11.25 = 0. s = 2.5 km/h. t = 60(4 - 9/2.5) = 24 min. At 3 km/h: time = 9/3 = 3 hours = 180 min. Total = 180 min. Answer: \boxed{180}" [Incorrect - forgot to add coffee time]

Figure 9: An end-to-end example of how an input is processed at each step of TTRL. This example illustrates a failure case in which the majority vote converges to an incorrect answer (180), causing reward signals to reinforce incorrect reasoning, a key limitation that motivates our CG-TTRL approach.

CG-TTRL End-to-End Example

Step 1: Prompt Formulation

System Prompt: Same as "System Prompt" in Figure 9.
Problem: Same as "Problem" in Figure 9
Retrieving/Finding Most Relevant Context Examples: Different algorithms/models (TF-IDF, TF-IDF with MMR Relevance, SBERT or hybrid solutions) can be utilised to retrieve the most similar context examples (e.g., 5 examples) and append them as context to the input:
Context Ex. 1: "Question: Suzanne walks four miles every third day. What is the fewest number of miles she can walk in February? Solution: February has 28 days with one extra day during leap years. We want the fewest number of miles, so we go with 28 days in February. The fewest number of days she can walk is floor(28/3) =9. So the fewest number of miles she can walk is 9x4=\boxed{36} miles. · · ·
Context Ex. 5: · · ·
Final Prompt / Model Input: {System Prompt} + {Context Example 1; Context Example 2; · · · ; Context Example 5} + {Now using the knowledge gained from these context examples, try solving this problem} + {Problem}

Step 2: Multi Sampling for Voting Phase

Sampling Multiple Responses (e.g., 8) in High Temperature:
Response 1: "Let's set up equations. Walking time + coffee time = total time. At speed s: 9/s + t/60 = 4. At speed s+2: 9/(s+2) + t/60 = 2.4. Solving: 9/s - 9/(s+2) = 1.6. 9(s+2-s)/[s(s+2)] = 1.6. 18/[s(s+2)] = 1.6. s(s+2) = 11.25. s^2 + 2s - 11.25 = 0. s = 2.5 km/h. t/60 = 4 - 9/2.5 = 0.4, so t = 24 minutes. At speed s+1/2 = 3 km/h: walking time = 9/3 = 3 hours = 180 minutes. Total time = 180 + 24 = 204 minutes. Answer: \boxed{204}" [Correct] · · ·
Response 8: · · ·
Majority Voting on Answers: [204, 204, 204, 180, 240, 205, 204, 204] => Majority vote = 204
Rewards Based on Matching the Majority Vote: Rewards = [1, 1, 1, 0, 0, 0, 1, 1].

Step 3: Sub-sampling for Test-Time Training Phase

This step is similar to Step 3 for the TTRL process (in Figure 9) without any difference.

Step 4: Updated Inference Stage After Test-Time Training Stage

Finally, after step 3, we prompt the model with the same {Model Input} created at the end of Step 1 and ask it to solve the task at a lower temperature (e.g. $\tilde{0}.3$). Expectedly, the model becomes more likely to generate the output leading to the majority vote at this stage. Please note that this step also uses the same context examples shown in Step 1. Thus, in Context-Guided Test-Time Reinforcement Learning (CG-TTRL), context helps boost performance in multiple components of the system.
Fine-tuned Model Output: "Let walking time be w hours. Then w + t/60 = total time. For first case: 9/s + t/60 = 4. For second case: 9/(s+2) + t/60 = 2.4. Subtracting: 9/s - 9/(s+2) = 1.6. 18/[s(s+2)] = 1.6. s^2 + 2s = 11.25. s = 2.5 km/h. t = 60×(4 - 9/2.5) = 24 min. At 3 km/h: time = 9/3 = 3 hours = 180 min. Total = 204 min. Answer: \boxed{204}" [Correct]

Note: In-Context Learning (ICL) is different from CG-TTRL in that it only consists of Steps 1 and 4. In ICL, you just find the relevant context examples and use them to have the model infer the answer for a target problem without including any multi-sampling stages for either majority voting or TTT. In CG-TTRL, the context helps create better and more stable majority votes. This improves reward quality for the TTT stage and ultimately the product-ready inference phase.

Figure 10: An end-to-end example of how an input is processed at each step of the Context-Guided Test-Time Reinforcement Learning (CG-TTRL) process and how it differs from ICL.