TEST-TIME META-ADAPTATION WITH SELF-SYNTHESIS

Zeyneb N. Kaya, Nick Rui
Stanford University
{zeynebnk, nickrui}@stanford.edu

ABSTRACT

As strong general reasoners, large language models (LLMs) encounter diverse domains and tasks, where the ability to adapt and self-improve at test time is valuable. We introduce MASS, a meta-learning framework that enables LLMs to self-adapt by generating problem-specific synthetic training data and performing targeted self-updates optimized for downstream performance at inference time. We train this behavior end-to-end via bilevel optimization: an inner loop adapts on self-generated examples while an outer loop meta-learns data-attribution signals and rewards post-update task performance. The synthetic data is optimized with scalable meta-gradients, backpropagating the downstream loss through the inner updates to reward useful generations. Experiments on mathematical reasoning show that MASS learns to synthesize per-instance curricula that yield effective, data-efficient test-time adaptation.

1 INTRODUCTION

Large language models (LLMs) have demonstrated great general-purpose capabilities, yet they are typically deployed as static artifacts. In real-world applications, however, models must continuously adapt to evolving tasks, new information, and shifting distributions encountered during deployment (Zheng et al., 2024; Wu et al., 2025).

The ability of models to *learn how to learn* at test time represents a critical frontier in AI development (Zweiger et al., 2025; Sun et al., 2020; Finn et al., 2017; Hu et al., 2025). Rather than relying on fixed pretrained capabilities, we explore whether models can learn how to best adapt themselves at test time for a new task or domain. Our approach frames this challenge as a bilevel optimization problem where models generate and curate their own problem-specific synthetic data to achieve the best self-updates at inference time. This enables models to utilize test-time compute to adapt to each unique problem they encounter and become data-efficient where high-quality task-specific supervision is sparse (Wang et al., 2023; Zelikman et al., 2022; Dong & Ma, 2025).

We introduce MASS (**M**ETA-**A**DAPTATION WITH **S**ELF-**S**YNTHESIS), a meta-learning framework for models to self-adapt at test time. We demonstrate the effectiveness of our method in mathematical reasoning across diverse fields; by dynamically constructing a synthetic curriculum for self-improvement at test time, MASS addresses problem-specific knowledge gaps while offering a scalable alternative to massive offline pretraining.

2 METHODS

We formulate test-time adaptation as a bilevel meta-learning problem. Each input is treated as a distinct task, and we learn *what* small synthetic self-generated dataset enables the best downstream performance for each task. Concretely, MASS has a *generator* which generates a corpus of synthetic auxiliary problem-solution pairs given a target task and a *scorer* which assigns scores to these examples based on their relevance to the target task (Calian et al., 2025). During training, MASS then performs temporary parameter updates with this weighted dataset prior to solution generation. Optimization of the model on this downstream performance allows the scorer to learn what examples are most important and the generator to generate such examples. The overall pipeline is illustrated in Figure 1.

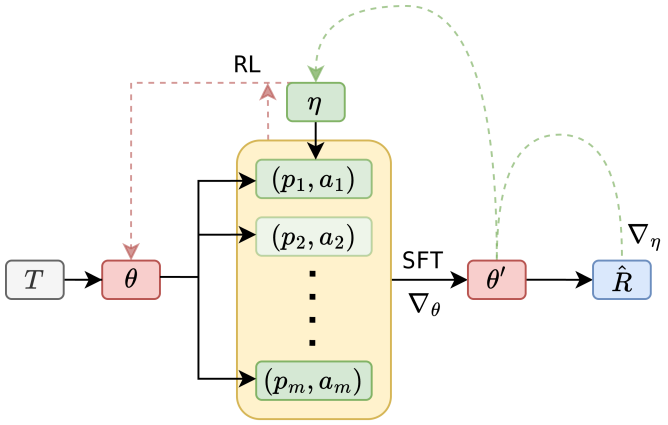

Figure 1: **MASS pipeline.** The generator parameterized by θ generates m auxiliary training examples $(p_1, a_1), \ldots, (p_m, a_m)$ given the target task T. The examples are assigned scores using the scorer parameterized by η. Then, θ performs a weighted SFT self-update on the synthetic data to produce an adapted model θ' before producing resposne $\hat{R}$ in attempting the target problem. Meta-gradients computed by backpropagating target performance through the inner updates are used to update η to identify examples that help the downstream loss and reward θ to generate better examples.

2.1 Data Generation and Inner-loop adaptation

We have a generator model π_θ and a scorer s_η. For a target task T, we sample m auxiliary examples $(p_i, a_i) \sim \pi_\theta(\cdot \mid T)$ for $i \in [m]$, forming an auxiliary dataset $\mathcal{D}(T) = \{(p_i, a_i)\}_{i=1}^m$. A scorer s_η assigns each example a relevance weight $s_i = s_\eta(T, p_i, a_i)$.

Initializing from the current model parameters θ, we perform a short update on the weighted auxiliary data according to

$$\mathcal{L}_{\text{inner}}(\theta, \eta; T) = \sum_{i=1}^{m} s_\eta(T, p_i, a_i)\, \ell(p_i, a_i; \theta),$$

where $\ell(\cdot)$ is a supervised loss. The adapted model θ' is then used to attempt the target task T.

2.2 Outer objective

We optimize θ and η so that the adapted parameters θ' achieve strong performance on T:

$$\min_{\theta,\eta} \; \mathcal{L}_{\text{outer}}(\theta'(\theta, \eta; T);\, T)\,.$$

In our experiments, we study two instantiations of $\mathcal{L}_{\text{outer}}$ depending on the assumptions of the setting: if a gold solution $R^\star$ is provided for T during training, we use standard cross-entropy; if no gold solution is available but we can verify candidate responses, we sample k attempts $\{\hat{R}_j\}_{j=1}^k$ from $\pi_{\theta'}(\cdot \mid T)$, and treat verified responses as the target.

2.3 Optimization

Backpropagating through the inner update with higher-order differentiation yields the meta-gradient for the scorer parameters η:

$$\frac{\partial \mathcal{L}_{\text{outer}}}{\partial \eta} = \frac{\partial \mathcal{L}_{\text{outer}}}{\partial \theta'} \frac{\partial \theta'}{\partial \eta}.$$

Updating according to this encourages the scorer s_η to upweight auxiliary examples whose induced adaptation step improves outer performance on T. We can derive the sensitivity to each example-level score from this computation as $\frac{\partial \mathcal{L}_{\text{outer}}}{\partial s_i} = \left\langle \nabla_{\theta'} \mathcal{L}_{\text{outer}}(\theta'; T),\; \frac{\partial \theta'}{\partial s_i} \right\rangle$, which directly measures whether increasing s_i would decrease the outer loss.

We use the meta-signal $-\frac{\partial \mathcal{L}_{\text{outer}}}{\partial s_i}$ as a reward shaping term for the generator π_θ to produce auxiliary examples that are useful for adaptation: auxiliary samples that would *reduce* the outer loss receive positive reinforcement.

Let x_i denote the generator input (here, $x_i \equiv T$ plus any formatting/context) and y_i denote the generator output representing the auxiliary pair (p_i, a_i). We compute GRPO-style advantages $\hat{A}_i$ computed from the rewards and normalized across the m samples. We then optimize a clipped policy-gradient surrogate:

$$\mathcal{L}_{\text{aux}}(\theta) = -\mathbb{E}_{\{y_i\}\sim\pi_{\theta_{\text{old}}}(\cdot|x)}\left[\frac{1}{m}\sum_{i=1}^{m}\min\Big(\frac{\pi_\theta(y_i \mid x_i)}{\pi_{\theta_{\text{old}}}(y_i \mid x_i)}\,\hat{A}_i,\ \text{clip}(\frac{\pi_\theta(y_i \mid x_i)}{\pi_{\theta_{\text{old}}}(y_i \mid x_i)}, 1-\epsilon, 1+\epsilon)\,\hat{A}_i\Big)\right].$$

To keep solving in-distribution, we include an additional term that reinforces solving performance to yield $\mathcal{L}_{\text{generator}}(\theta) = \mathcal{L}_{\text{aux}}(\theta) + \gamma\,\mathcal{L}_{\text{solve}}(\theta)$, with hyperparameter γ. When gold solutions are available, $\mathcal{L}_{\text{solve}}$ is an SFT cross-entropy loss on the gold solution. When only verification is available, $\mathcal{L}_{\text{solve}}$ is a similar GRPO-style loss over the k attempts on T using binary verifier outcomes as rewards (and $\pi_{\theta_{\text{old}}} := \pi_{\theta'}$ for that solve-policy update).

Computing $\nabla_\eta \mathcal{L}_{\text{outer}}$ requires differentiating through the inner update(s), which naïvely involves expensive second-order terms and large activation storage when backpropagating through an unrolled inner loop. We build on recent work on scalable bilevel differentiation Kemaev et al. (2025); Calian et al. (2025), computing meta-gradients in a memory-efficient mixed-mode form (forward-over-reverse vs. standard reverse-over-reverse unroll), as well as block-level rematerialization with gradient checkpointing.

Each meta-training iteration then proceeds as: (1) generate data for T, (2) compute scores via η, (3) adapt model parameters on weighted data, (4) compute outer loss on T, (5) update both η and θ using the meta-gradients.

3 Experiments

3.1 Experimental Setup

We work with the task of mathematical reasoning, evaluating performance on the MATH-500 benchmark (Lightman et al., 2023), which contains mathematical reasoning problems from various different fields of math, measuring our methods' ability to adapt for different domains. We train on 1,000 examples sampled from the training split of the MATH dataset (Hendrycks et al., 2021), disjoint from the evaluation tasks. We note that we minimally optimize for actual math generation, and that our approach primarily leverages training data as signal to *meta-learn*.

We work with Llama 3.1-8B-Instruct (Grattafiori et al., 2024) as our base model and utilize parameter-efficient LoRA training (Hu et al., 2021) for temporary inner loop updates. The scorer model follows the base model's architecture but is lightweight in size and has a sigmoid-constrained numerical scoring head. We train for about 100 steps with 20 warmup steps where the generator is frozen and the scorer updated. We perform 2 inner steps and unroll through both.

For each task, we generate 12 candidate synthetic training examples. For the verifier-only setting, the outer loss is computed by sampling 6 solution attempts on the target task. At test time we generate 6 synthetic training examples then perform an unweighted SFT LoRA update before attempting the target theorem.

3.2 Results

We compare multiple baselines. Our results are shown in Table 1.

- **Base:** The base model (Llama-3.1-8B-Instruct).
- **Base Test-Time Self-Synthesis (TT-SS):** The base model generates k synthetic training examples, performs a LoRA update, then attempts the target task.
- **Base Test-Time Training (TTT):** k examples are sampled from the MATH training data; the base model performs a LoRA update, then attempts the target task.

- **Solver GRPO:** The base model trained with vanilla GRPO directly for math solving with binary correctness rewards.
- **MASS:** MASS generates k synthetic training examples, performs a LoRA update, then attempts the target task (this represents the setting with no gold solution, only verification).
- **MASS$_{gold}$:** MASS generates k synthetic training examples, performs a LoRA update, then attempts the target task (this represents the setting using the gold solution in the outer loss).

Table 1: Accuracy of methods on MATH-500.

Method	**MATH-500**
Base	43.6%
Base TTT	41.2%
Base TT-SS	46.6%
Solver GRPO	49.1%
MASS	**59.0%**
MASS$_{gold}$	54.1%

In the evaluations in Table 1, MASS achieves the strongest performance, substantially outperforming all baselines and improving on the Base model by 15.4pp (x1.35). Without meta-learning, the model shows limited capabilities in generating targeted beneficial synthetic data for self-adaptation (Base-TT-SS), but is able to improve from Base by 3.0pp. Naive TTT slightly hurts performance, suggesting that generic test-time updates without problem-specific supervision can introduce drift. MASS performs strongly in both settings where it updates according to golden solutions and verified self-generated solutions.

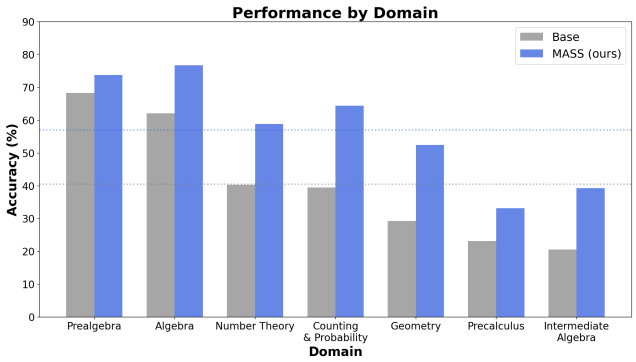

Figure 2: Performance gains by domain.

MASS especially shows its potential in improving performance across domains. In Figure 2, it provides the greatest gains where initial performance is weakest (Intermediate Algebra, 1.92x) and overall improves consistency of performance across domains. MASS demonstrates that it can effectively utilize test-time compute to generate problem-specific synthetic data to adapt itself to the domain in a data-efficient manner, better leveraging its knowledge and resources for its environment.

4 Conclusion

Test-time MASS (Meta-Adaptation with Self-Synthesis) demonstrates that models can meta-learn to self-adapt at inference time by generating problem-specific synthetic curricula. MASS learns *what* self-generated updates are useful for optimal performance gains on the current input, enabling generalizable per-instance self-improvement. We consistently improve over strong and compute-matched baselines in mathematical reasoning. Our work emphasizes synthetic data generation for self-improvement and efficient meta-learning for data attribution, and opens up the method as a general mechanism for models to robustly adapt in any setting.

REFERENCES

Dan A. Calian, Gregory Farquhar, Iurii Kemaev, Luisa M. Zintgraf, Matteo Hessel, Jeremy Shar, Junhyuk Oh, András György, Tom Schaul, Jeffrey Dean, Hado van Hasselt, and David Silver. Datarater: Meta-learned dataset curation, 2025. URL https://arxiv.org/abs/2505.17895.

Kefan Dong and Tengyu Ma. Stp: Self-play llm theorem provers with iterative conjecturing and proving. *arXiv preprint arXiv:2502.00212*, 2025. URL https://arxiv.org/abs/2502.00212.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *CoRR*, abs/1703.03400, 2017. URL http://arxiv.org/abs/1703.03400.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, and Abhinav Pandey et al. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

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. *NeurIPS*, 2021.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021. URL https://arxiv.org/abs/2106.09685.

Jinwu Hu, Zhitian Zhang, Guohao Chen, Xutao Wen, Chao Shuai, Wei Luo, Bin Xiao, Yuanqing Li, and Mingkui Tan. Test-time learning for large language models, 2025. URL https://arxiv.org/abs/2505.20633.

Iurii Kemaev, Dan A. Calian, Luisa M. Zintgraf, Gregory Farquhar, and Hado van Hasselt. Scalable meta-learning via mixed-mode differentiation. *arXiv preprint arXiv:2505.00793*, 2025. URL https://arxiv.org/abs/2505.00793.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.

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 Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 9229–9248. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/sun20b.html.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. SELF-INSTRUCT: Aligning language models with self-generated instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 13484–13508, 2023. URL https://aclanthology.org/2023.acl-long.754/.

Tongtong Wu, Linhao Luo, Yuan-Fang Li, Shirui Pan, Thuy-Trang Vu, and Gholamreza Haffari. Continual learning of large language models: A comprehensive survey. *ACM Computing Surveys*, 2025.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. STar: Bootstrapping reasoning with reasoning. *arXiv preprint arXiv:2203.14465*, 2022. URL https://arxiv.org/abs/2203.14465.

Junhao Zheng, Chengming Zheng, Qianli Xue, Yujun Guo, Siyi Shi, Kaiwen Zhou, Xiao Xu, and Tao He. Towards lifelong learning of large language models: A survey. *arXiv preprint arXiv:2406.06391*, 2024.

Adam Zweiger, Jyothish Pari, Han Guo, Ekin Akyürek, Yoon Kim, and Pulkit Agrawal. Self-adapting language models. *arXiv preprint arXiv:2506.10943*, 2025. URL https://arxiv.org/abs/2506.10943.