SELF-GENERATIVE ADVERSARIAL FINE-TUNING FOR LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Post-training is central to making large language models (LLMs) useful, but it relies on high-quality examples and feedback that are scarce and costly to collect. Iterative self-improvement makes better use of the available data by generating new training examples, provided the model can assess what it generates. Existing methods accept model-generated data without a grounded correction signal or rely on self-evaluation scores that may bias toward the model's own style. We introduce Self-Generative Adversarial fine-tuning for LLM (SGALM), which learns that assessment from real examples. At each round, one LLM generates complete examples from few-shot prompts and learns to distinguish its generations from the real data. The generation is rewarded to be realistic while the discrimination is trained by the truth, instantiating a generative adversarial game without a external feedback or new human labels. Across tasks from different domains, SGALM significantly improves over SFT and existing self-improvement method on different backbones. SGALM also serves as a synthetic-data engine and a realness-based filter, whose generated data continues to improve downstream training. We also diagnose discriminator score and generation diversity which provide evidence against self-evaluation bias and mode collapse.

1 INTRODUCTION

Pretraining gives language models broad capabilities; post-training determines how reliably they follow instructions, solve tasks, and respond to human preferences. Supervised fine-tuning (SFT) teaches from examples, while methods such as reinforcement learning (RL) from human feedback and direct preference optimization (DPO) learn from judgments about model responses Ouyang et al. (2022); Bai et al. (2022); Rafailov et al. (2023). Each stage needs high-quality demonstrations or feedback. Such data are scarce because writing, checking, and comparing responses takes human effort, especially across many tasks and behaviors. This limits how broadly and how often models can be updated.

To address this bottleneck, the field has increasingly turned towards iterative self-improvement, by utilizing LLMs to generate their own training signals. This includes generating synthetic data for SFT Wang et al. (2023); Honovich et al. (2023) and more sophisticated self-play mechanisms Gulcehre et al. (2023); Xiong et al. (2023); Chen et al. (2024); Yuan et al. (2024); Pang et al. (2024), where the model evolves as both the student and the teacher in a closed feedback loop. This can expand training without requiring a new human annotation for every generated sample. The difficulty is deciding what to learn from those samples. Training indiscriminately on generated data can reinforce mistakes, while a model judging its own outputs may favor patterns it already produces

A key challenge of iterative self-improvement is distinguishing useful synthetic data from hallucinations. SPIN Chen et al. (2024) optimizes a preference objective that treats real responses as preferred to responses from the previous model, whereas Self-Rewarding Yuan et al. (2024) uses the LLM's own zero-shot judgments as a training signal. We ask whether the real examples already available for fine-tuning can supply feedback for the distinguishing process, resulting in SGALM. It uses one LLM to generate complete new examples and to distinguish them from the real dataset. The model learns the distinction from fresh real and generated samples, then uses the discriminator's score to update its generation policy. Thus, its feedback is learned from iterative real/generated contrasts rather than a fixed real-over-generated preference or an ungrounded self-judge.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

This interaction follows the generative adversarial network (GAN) formulation Goodfellow et al. (2014): a generator tries to produce samples that a discriminator cannot distinguish from real ones, while the discriminator learns from each new contrast. SGALM the challenge of applying the GAN framework to discrete text generation and discrimination within an LLM by utilizing in-context learning (ICL) for generation and the output distribution for optimizable discrimination. Thanks to the human-like intelligence and the powerful, consistent input-output spaces (natural language) of contemporary LLMs, a single LLM can serve as both the generator and discriminator in a GAN architecture.

Figure 1 summarizes the iterative loop of self-generative adversarial fine-tuning LLM (SGALM). Not only the process can serve as a way to **fine-tune the model** itself, but also the outcome model can serve as a **synthetic data engine** which can generate and filter examples for training another model. We summarize our contributions as follows:

- We introduce an iterative self-improvement method which formulates LLM fine-tuning as a self-contained adversarial game where a LLM jointly evolves its generation and discrimination capabilities without an external reward model.
- SGALM realize the GAN framework to discrete text generation by ICL-based generation, and a reinforce-style generator update. We provide a theoretical analysis giving explicit sufficient conditions under which the classical population GAN equilibrium for the few-shot generator entails zero-shot target alignment.
- Experiments covering math reasoning, instruction following, and coding show statistically significant improvements over existing methods across different benchmarks and backbones. Experiments also show SGALM trained model is superior in synthetic data generation and filtering for downstream training.

2 PRELIMINARIES

A generative adversarial network (GAN) compares samples from a target distribution p_T with those from a generator G, whose output distribution is p_G. A discriminator D assigns a score to each sample. The two models optimize the following objective:

$$\min_G \max_D J \equiv \mathbb{E}_{z' \sim p_G}[\log(1 - D(z'))] + \mathbb{E}_{z \sim p_T}[\log D(z)]. \tag{1}$$

The discriminator D is trained to distinguish between real samples z drawn from the data distribution p_T and fake samples z' produced by the generator G. Simultaneously, the generator G aims to synthesize realistic samples that deceive the discriminator D, minimizing the D's ability to differentiate them from real data.

The minimax game is played by iteratively updating D and p_G: in every iteration, z' and z are drawn from p_G and p_T respectively; first D is updated towards $\max_D J$; then G is updated towards $\min_G J$. It theoretically drives the generator p_G to recover the true data distribution p_T when the discriminator can no longer distinguish generated samples from real ones. Due to the complexity of the bi-level optimization problem (1), the training of GANs typically makes approximation by decompose (1) into a discrimination objective $\max_D J(D)$, and a generation objective $\min_G J(G)$, and optimizes them iteratively.

In computer vision, the adversarial game has been successfully deployed for diverse applications such as domain adaptation Ganin & Lempitsky (2015), style transfer Zhu et al. (2017), and data augmentation Antoniou et al. (2017). In natural language processing, however, prior attempts Yu et al. (2017); Che et al. (2017); Guo et al. (2018) primarily focused on employing Reinforcement Learning to bypass the non-differentiability of discrete tokens and were limited to training small models for text generation. SGALM revisits this adversarial formulation in the setting of fine-tuning LLMs, using one model for both roles and a policy-gradient estimator for discrete generation. Detail related works are discussed in Appendix A.

3 OUR METHOD

Let z be a text sequence, with length $|z|$ and token z_t at position t. Its distribution factorizes autoregressively as $p(z) = \prod_{t=1}^{|z|} p(z_t \mid z_{<t})$. An LLM with parameters θ maps each prefix $z_{<t}$

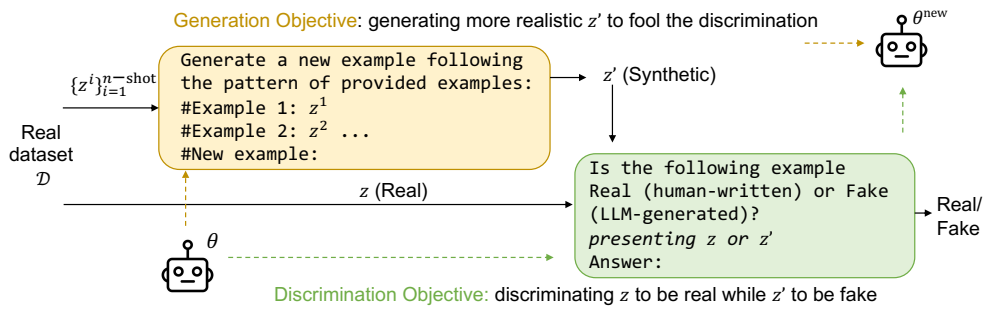


Figure 1: Illustration of the proposed self-generative adversarial fine-tuning framework.

to a token distribution $p_\theta(\cdot \mid z_{<t})$. We are given a dataset $\mathcal{D}$ of real samples drawn from a target distribution p_T. SGALM uses this dataset to train the model and to guide the generation of new samples.

The first noticeable difference between SGALM and existing methods is that z need not to be an input-out pair (e.g. (x, y)), but can be a document, an instruction–response pair, or another text sequence. Rather than generating new sample by answering to specific input, SGALM generates by few-shot ICL. The underlying insight is that if an LLM is capable of generating realistic, diverse, and correct samples within the target distribution, then it is expertise enough, which already implies answering to a x. This distinction makes SGALM can be applied to broader datasets beyond input-out pair, and the outcome model as a synthetic data engine.

Given a dataset $\mathcal{D}$ and a generative model p_θ, the method has three steps: **generation** produces synthetic samples z'; **discrimination** assigns a scalar score $D(z)$ to real and generated samples; and **updating** trains both roles. For an LLM, this requires varied generated examples, a trainable discrimination score, and a generator update that works with discrete tokens while both roles share parameters.

3.1 Generation

The GAN framework requires the generation of z' to be diverse enough, as drawing from a continuous distribution. Typical GANs implement G using a model that takes noise drawn from a continuous distribution as input, but LLMs lacks such mechanism. We vary the few-shot context and sample from the model's token distribution. This uses the in-context learning (ICL) capability of modern LLMs to generate examples related to the real dataset Brown et al. (2020); Wang et al. (2023).

We sample a few-shot subset $\{z^i\}_{i=1}^{n\text{-shot}} \subset \mathcal{D}$ to construct a context prompt Z_{ctx} containing the n real examples. The model generates a "fake" sample z' via:

$$z' \sim p_\theta^G(z') \equiv p_\theta(z' \mid G_{\text{Prompt}}, Z_{\text{ctx}}), \tag{2}$$

where G_{Prompt} is the generative instruction in the prompt, such as "*generate a new example following the pattern of provided examples*". Since LLMs possess ICL capacity to identify patterns in context, it is expected that the LLM can learn p_T from the provided Z_{ctx}. Meanwhile, as the number of combinations and permutations of examples $\{z^i\}$ from $\mathcal{D}$ to form Z_{ctx} is vast, and the inherent randomness in the generation process $p_\theta(\cdot \mid x)$ is controllable, the generation of z' maintains sufficient diversity. Because a new Z_{ctx} is sampled across generations, the fake distribution presented to the discriminator is the context-marginal distribution, rather than the conditional distribution for one fixed context; this marginal is formalized in Eq. (9).

3.2 Discrimination

The discrimination mechanism is expected to output a scalar $D(z)$ for each sample z (or z'), to evaluate the objective (1). Typical GANs implement the discriminator model with a classifier with demanded output. For using LLM as discriminator, there are implementations of using additional module to map the final hidden states Yu et al. (2023), or directly asking the LLM to output a scalar score in text Yuan et al. (2024). However, they are not feasible in SGALM, as the former one breaks

the behavior of LLM as general intelligence, while the latter one requires first fine-tuning on scoring dataset.

In SGALM, we ask the LLM if the input is "Real" or "Fake", and obtain the scalar score based on its output distribution. Specifically, given an input z, we have a distribution $p_\theta(\cdot|D_{\text{Prompt}}, z)$, where D_{Prompt} is the discriminative instruction in prompt, like "*Is the example Real (human-written) or Fake (LLM-generated)?*". Then we have the output probability $p_\theta(\text{Real}|D_{\text{Prompt}}, z)$ and $p_\theta(\text{Fake}|D_{\text{Prompt}}, z)$. Define the normalized binary distribution on {"Real", "Fake"$|z$} as

$$p_\theta^{\text{real}}(z) \equiv \frac{p_\theta(\text{Real}|D_{\text{Prompt}}, z)}{p_\theta(\text{Real}|D_{\text{Prompt}}, z) + p_\theta(\text{Fake}|D_{\text{Prompt}}, z)}, \tag{3}$$

and $p_\theta^{\text{fake}}(z) \equiv 1 - p_\theta^{\text{real}}(z)$. These continuous values indicate how real or fake the LLM considers sample z. We set $D_\theta(z) = p_\theta^{\text{real}}(z)$ so that the binary cross-entropy objective below has the usual probabilistic interpretation. A monotone transform may be used as a heuristic ranking reward, but the density-ratio result in Section 4.1 applies specifically to this probability-valued discriminator.

3.3 Updating

In SGALM, the discrimination objective is straight-forward, maximizing the probability of telling "Real" for real samples z while telling "Fake" for fake samples z'. With $D_\theta(z) = p_\theta^{\text{real}}(z)$, the discrimination objective is

$$\begin{aligned} &\max_D \mathbb{E}_{z'\sim p_G}[\log(1 - D(z'))] + \mathbb{E}_{z\sim p_T}[\log D(z)] = \\ &\min_\theta -\mathbb{E}_{z'\sim p_G}[\log(1 - D_\theta(z'))] - \mathbb{E}_{z\sim p_T}[\log D_\theta(z)]. \end{aligned} \tag{4}$$

Note that specifically in SGALM, G and D are implemented with the same model θ, i.e., $p_G(z') = p_\theta^G(z')$ and $D(z) = D_\theta(z)$. However, the objective here is to discriminate real/fake samples better, rather than generating worse. So denoting a detached copy of θ as $\theta^\dagger$, (4)

$$= \min_\theta -\mathbb{E}_{z'\sim p_{\theta^\dagger}^G}[\log(1-D_\theta(z'))] - \mathbb{E}_{z\sim p_T}[\log D_\theta(z)].$$

The discriminator parameters are updated by taking gradient steps with respect to this supervised binary cross-entropy objective:

$$\nabla_\theta J(D) = -\mathbb{E}_{z'\sim p_{\theta^\dagger}^G}[\nabla_\theta \log(1-D_\theta(z'))] - \mathbb{E}_{z\sim p_T}[\nabla_\theta \log D_\theta(z)]. \tag{5}$$

The expectations are approximated via Monte Carlo sampling using fake samples generated from p_θ^G and real samples from the dataset $\mathcal{D}$.

The generator is updated to produce samples that receive higher scores from a detached discriminator:

$$\min_G \mathbb{E}_{z'\sim p_G}[\log(1-D_{\theta^\dagger}(z'))] = \min_\theta \mathbb{E}_{z'\sim p_\theta^G}[\log(1-D_{\theta^\dagger}(z')]. \tag{6}$$

Specifically, let the detached raw reward be $r(z') = -\log(1 - D_{\theta^\dagger}(z'))$. For the unnormalized objective in (6), the score-function identity gives

$$\nabla_\theta L_G = -\mathbb{E}_{z'\sim p_\theta^G}\left[(r(z') - b)\nabla_\theta \log p_\theta^G(z')\right], \tag{7}$$

where b is any action-independent baseline. Equation (7) is a reinforce-style policy-gradient update. The discriminator is detached while computing r, so the generation update step cannot optimize (6) directly through blinding the discriminator, although the shared-parameter update can change future discriminator outputs. The complete algorithm is provided in Algorithm 1.

4 Theoretical Analysis

We separate the standard population-level GAN equilibrium (Sections 4.1) from the SGALM-specific question of parameter sharing (Sections 4.2). and the connection between few-shot generation and zero-shot behavior. Sections 4.1 and 4.2 are supportive results for understanding the method, while Section 4.3 gives the connection between few-shot generation and zero-shot behavior, explaining why SGALM process itself works for alignment between p_T and not only p_θ^G (with few-shot demonstrations), but also p_θ (zero-shot).

Algorithm 1 SGALM synchronous generator–discriminator optimization.

Input: Real dataset $\mathcal{D}$, LLM θ, iterations T.
for $i = 0, \ldots, T-1$ **do**
  Generate $\mathcal{D}'_i \sim p^G_{\theta_i}$ by (2).
  Discriminate real and generated samples: compute $D_{\theta_i}(z)$ for $z \in \mathcal{D}'_i \cup \mathcal{D}$
  Update θ_i with (5) and (7), yielding θ_{i+1}.
end for

4.1 What the Discriminator Score Represents

The population analysis needed here is the classical GAN result of Goodfellow et al. (2014), recapitulated for SGALM in Appendix B.1–B.2. Under the usual unrestricted-functional-capacity and attained-global-optimum assumptions, given generator p_G, the discriminator has optimum

$$D^*(z) = \frac{p_T(z)}{p_T(z) + p_G(z)}, \qquad \frac{D^*(z)}{1 - D^*(z)} = \frac{p_T(z)}{p_G(z)}, \tag{8}$$

and the global adversarial optimum is $p_G = p_T$. These are standard population-level GAN properties rather than new theoretical contributions. Thus the optimal discriminator is a density ratio statistic between target and the current generator. Section 5.5 directly diagnoses its empirical relationship with task correctness and finds that the learned realness score is strongly associated with verified correctness.

4.2 Parameter Sharing and Self-Evaluation Bias

A natural concern is that one LLM acting as both generator and discriminator may reward its own generation style. At the functional level, Eq. (8) still describes an unrestricted optimal discriminator for the induced generator distribution. Parameter sharing, however, restricts which generator-discriminator pairs can be represented simultaneously, and an update to the shared θ can change both induced functions even when the inactive branch is detached. Consequently, the classical result should be viewed as an idealized characterization of the adversarial objective rather than a claim that practical SGALM reaches the population GAN equilibrium. Our empirical question is instead whether the resulting adversarial signal provides an effective fine-tuning objective under finite-step, shared-parameter optimization

Finite shared capacity can nevertheless couple the two roles and create a bottleneck. Empirically, the non-shared variant is competitive and is better on harder tasks for the 3B backbones, whereas shared is consistently better for larger model (Appendix C.3). We therefore treat parameter sharing as an efficiency-oriented design whose benefit depends on model capacity, not as a universally superior choice.

4.3 From Few-Shot Generation to Zero-Shot Understanding

A natural question when interpreting SGALM is whether matching samples generated with few-shot contexts says anything about behavior without demonstrations. Here we state explicit sufficient conditions to achieve the "zero-shot target alignment" in the sense of matching the target distribution without target demonstrations.

Let $C = (z^1, \ldots, z^n) \sim p_T^n$ be a random n-shot context, with the fixed generation instruction suppressed from notation, and define the distribution observed by the discriminator as

$$q_\theta(z) \equiv \mathbb{E}_{C \sim p_T^n}\left[p_\theta^G(z \mid C)\right]. \tag{9}$$

Thus the classical population equilibrium establishes $q_{\theta^*} = p_T$.

The following assumption is an idealized latent-domain model of ICL, motivated by Bayesian accounts of ICL in stylized settings Akyürek et al. (2022); Ahn et al. (2023); Bai et al. (2023); Wu et al. (2023); Li et al. (2023); Wu et al. (2025).

Assumption 4.1 (Shared latent-domain ICL model)**.** Let $\mathcal{H}$ be a finite family of domains containing the target T, with a fixed sequence distribution p_h for each $h \in \mathcal{H}$. At any analyzed parameter

value θ, the model has a full-support domain prior $\pi_\theta \in \Delta(\mathcal{H})$, so $\pi_\theta(h) > 0$ for every h. For a context $C = (z^1, \ldots, z^n)$, it selects $h_\theta(C) \in \arg\max_{h\in\mathcal{H}} \pi_\theta(h) \prod_{i=1}^n p_h(z^i)$, and generates according to $p_\theta^G(\cdot \mid C) = p_{h_\theta(C)}(\cdot)$. The same prior governs the no-demonstration mode: with the empty-product convention, $p_\theta^0(\cdot) = p_{h_\theta(\emptyset)}(\cdot)$, where $h_\theta(\emptyset) \in \arg\max_h \pi_\theta(h)$, using the same deterministic tie-breaking rule. SGALM is assumed to remain in this model class during the analyzed fine-tuning trajectory.

In plain words, the same domain prior controls both few-shot and zero-shot behavior, while demonstrations modify domain selection through their likelihood.

Assumption 4.2 (Target identifiability)**.** The target is mixture-identifiable in $\{p_h : h \in \mathcal{H}\}$: for every $w \in \Delta(\mathcal{H})$, $\sum_{h\in\mathcal{H}} w_h p_h = p_T \implies w_T = 1$. Moreover, for every $h \neq T$, there is a positive-probability set of target contexts on which h has strictly larger context likelihood than T: $\Pr_{C\sim p_T^n}\left[\prod_{i=1}^n p_h(z^i) > \prod_{i=1}^n p_T(z^i)\right] > 0$.

The first condition rules out the counterexample in which non-target context-specific generators mix to exactly p_T. The second makes the prior ordering identifiable from few-shot decisions: demonstrations cannot always overwhelm an incorrectly ordered prior. For distinct probability mass functions sharing the same support, the second condition follows because each likelihood ratio $p_h(z)/p_T(z)$ must exceed one on a set of positive p_T probability.

Theorem 4.3 (Few-shot equilibrium implies zero-shot target alignment)**.** *Suppose Assumptions 4.1–4.2 hold. If the idealized population adversarial objective attains an equilibrium θ^* with $q_{\theta^*} = p_T$, then T is the unique maximizer of π_{θ^*} and $p_{\theta^*}^0(\cdot) = p_T(\cdot)$.*

As a consequence, since language models factorize the joint distribution autoregressively as $p(z) = \prod_{t=1}^{|z|} p(z_t \mid z_{<t})$, zero-shot joint matching implies matched continuation distributions for target-supported prefixes, i.e., for every $z_{<t}$ with $p_T(z_{<t}) > 0$, $p_{\theta^*}^0(z_{\geq t} \mid z_{<t}) = p_T(z_{\geq t} \mid z_{<t})$. The proof is in Appendix B.3.

5 Empirical Results

We evaluate SGALM as both a fine-tuning method to align p_θ to p_T, and a synthetic-data engine for training another model.

5.1 Experimental Setup

Models and datasets. Our main experiments use Qwen2.5-3B-Instruct Yang et al. (2024) on GSM8K Cobbe et al. (2021), ARC-Challenge Clark et al. (2018), and MBPP Austin et al. (2021). Their training sets contain 7.47k, 1.12k, and 0.12k examples, respectively, allowing us to study three domains and substantially different data regimes. Appendix C.1 reports a matched six-benchmark comparison with SPIN; Appendices C.3 and C.4 report Llama up to 7B and open-ended instruction-following results.

Baselines. We compare against Base, SFT, Self-Instruct Wang et al. (2023), Self-Rewarding Yuan et al. (2024), SPIN Chen et al. (2024), Iterative-RS, and Iterative-RPO Pang et al. (2024). Iterative-RS is a reject-sampling version of iterative self-play adapted from ReST Gulcehre et al. (2023) and Iterative-DPO Xiong et al. (2023) without external reward models.

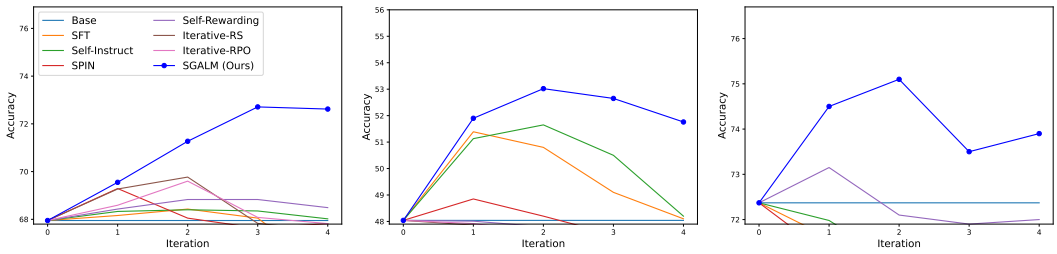
Controlled synthetic-data budget. For all iterative methods, each iteration uses the same backbone, the same dataset split, the same evaluation protocol, and the same number of generated samples. For ICL-based generation, all methods use the same 4-shot setting.

Uncertainty and paired comparisons. Each $\pm$ value in the main results is the standard deviation across five independent training–evaluation runs with different seeds. Because all methods are evaluated on the same test items, we additionally report paired bootstrap confidence intervals for method differences, resampling runs and test items as described in Appendix C.2.

Implementation and reproducibility. Unless otherwise specified, the synthetic-data scale per iteration equals the size of the real training set. Appendix C.8 reports the iteration schedule, optimization settings, and 4-shot ICL generation protocol; Appendix C.9 reports the training cost; and Appendix C.11.2 provides the concrete prompt template and generation/discrimination examples. Code is available at `https://anonymous.4open.science/r/SGALM_Code-4868/`.

Table 1: Performance on Qwen2.5-3B-Instruct. N.A. means the baseline is not applicable under no-external-reward setting.

Method	GSM8K	ARC	MBPP
Base	67.95±1.24	48.04±1.46	72.37±1.56
SFT	68.61±1.26	51.39±1.43	71.60±1.95
Self-Instruct	68.40±1.23	51.65±1.48	71.98±1.74
Self-Rewarding	68.83±1.24	48.01±1.48	73.15±1.95
SPIN	69.29±1.24	48.85±1.42	71.21±1.10
Iterative-RS	69.77±1.26	47.86±1.43	N.A.
Iterative-RPO	69.60±1.26	47.92±1.45	N.A.
SGALM (D-only)	69.83±1.23	49.24±1.46	73.15±1.56
SGALM (G-only)	69.98±1.25	50.58±1.41	73.54±1.95
SGALM (w/o S)	71.65±1.24	51.65±1.45	**76.26±1.10**
SGALM	**72.71±1.23**	**53.02±1.45**	75.10±1.95



(a) GSM8K (Math, 7.5k) (b) ARC (Science, 1.1k) (c) MBPP (Coding, 0.1k)

Figure 2: Test accuracy across training iterations for SGALM, its variants, and baseline methods.

5.2 Main Results

Table 1 reports the main results. Relative to SFT, SGALM improves by 4.10, 1.63, and 3.50 percentage points on GSM8K, ARC-Challenge, and MBPP; relative to SPIN, the gains are 3.42, 4.17, and 3.89 points. All six paired differences have 95% confidence intervals excluding zero (Appendix C.2). The D-only and G-only variants also improve over several baselines. Both updates can help, though the full model does not beat every ablation on every task. N.A. denotes settings where a baseline requires an unavailable external verifier or reward.

5.3 Breadth across Benchmarks and Backbones

Under the UltraChat200k setting matched to SPIN, SGALM obtains a 62.7 six-benchmark average versus 61.6 for SPIN, leading on four of six tasks (Appendix C.1). Cross-backbone ablations on Llama-3.2-3B and Llama-3.1-8B further show that the complete method is best on five of six model–task combinations, with the non-shared 3B model performing best on MBPP (Appendix C.3). Instruction following dataset IFEval results are reported in Appendix C.4.

5.4 Ablations and Training Dynamics

We implement three variants: D-only, which keeps only the discrimination update; G-only, which keeps only the generation update; and w/o S, which uses separate generator and discriminator models. Figure 2 shows test accuracy across iterations. The curves make the data-scarcity issue visible: even on GSM8K, the largest training set in our experiments, several iterative baselines begin to overfit before four iterations, and methods that generate only a new response y for a fixed question x often drop below Base. In contrast, methods that generate new $z = (x, y)$ samples maintain more diversity, and SGALM obtains the strongest overall performance because it combines ICL-based sample diversity with grounded adversarial feedback.

Contribution of discrimination and generation. D-only improves performance even without an explicit generation update because the shared parameters affect the model's generation distribution

Table 2: Direct diagnostics of the learned discriminator. Correctness is measured on 100 generations per dataset (four responses for each of 25 held-out prompts). Perturbations start from held-out real samples and either change task semantics or preserve correctness while changing surface form.

	Generated responses				Controlled perturbations		
Dataset	Correct	$\mathbb{E}[D \mid C_T{=}1]$	$\mathbb{E}[D \mid C_T{=}0]$	AUROC	Original	Semantic	Surface
GSM8K	73/100	0.41	0.02	0.86	0.93	0.01	0.79
MBPP	76/100	0.56	0.15	0.79	0.70	0.11	0.64

while learning a better real/fake distinction. This effect is measurable: the cumulative KL shift of the generation distribution is 5.89 for D-only, compared with 11.92 for full SGALM and 12.60 for G-only (Appendix C.10.1). G-only is also strong because it directly optimizes generated samples against the real-data-grounded discriminator score. Unlike Self-Rewarding, which asks the model to rank multiple responses for the same question, SGALM provides a continuous real/fake score from the output distribution in Eq. (3); this makes the signal optimizable and grounded by contrast with the real dataset. Full SGALM combines both effects. The w/o S variant tests whether sharing parameters causes a capacity bottleneck: it is comparable overall and better on MBPP, but requires roughly two-model training cost. SGALM is better on GSM8K and ARC but slightly worse than w/o S on MBPP, suggesting that sharing is efficient in general but can become capacity-limited when the task is harder or the base model is smaller.

5.5 Discriminator Reliability and Semantic Correctness

Figure 3 shows that seen and held-out real samples have similar score distributions, while generated samples remain separated, showing that the shared discriminator does not simply reward its own outputs or memorize the training set. Table 2 then controls the prompt and task style by sampling multiple responses to the same held-out prompts and assigning correctness with task-specific verifiers. Correct generations receive substantially higher scores than incorrect ones, with AUROC 0.86 on GSM8K and 0.79 on MBPP. In the perturbation diagnostic, changing only the final answer or a code line reduces the score by 0.92 and 0.59, whereas correctness-preserving surface changes reduce it by only 0.14 and 0.06. These results show that learned D score is much more sensitive to semantic correctness than to the tested formatting artifacts. Construction details are provided in Appendix C.5.

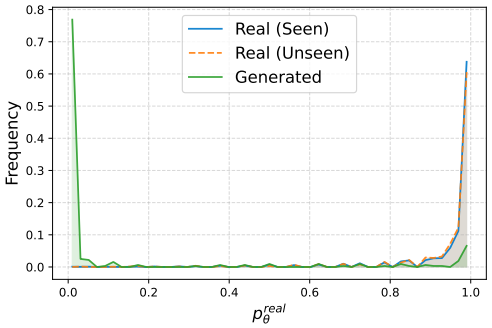


Figure 3: Discrimination score distributions for seen real, held-out real, and generated samples.

After the first iteration, discriminator scores for real and generated samples separate sharply. Within-batch reward standardization is consequently important for the generator update; sensitivity results are reported in Appendix C.6.

5.6 Evidence against Mode Collapse

We do not observe identical-sample collapse, although later iterations increasingly share short prefixes before diverging into distinct suffixes. On GSM8K, 28% of iteration-4 generations begin with "`Q: A bakery`", 10% with "`Q: A bakery makes`", and only 1.25% with "`Q: A bakery makes cupcakes`", showing rapid diversification as the prefix grows. After the first iteration, the average log realness scores remain in narrow ranges for generated samples (approximately -1.9 to -1.7) and real samples (-0.04 to -0.06). Because task accuracy peaks within four iterations, we do not interpret the observed trajectory as evidence of convergence to, or indefinite stability around, the population GAN equilibrium. Instead, these results show that the adversarial signal can be useful for fine-tuning within a finite training regime. We therefore use a fixed iteration budget selected on validation performance. Full trajectories and examples are provided in Appendices C.10.1 and C.12.

Table 3: SFT Qwen2.5-1.5B-Instruct on GSM8K with synthetic data generated by different methods.

Base	50.72 ± 1.38		
SFT with real	57.12 ± 0.59		
SFT with real +	Syn. ($1\times$)	Syn. ($2\times$)	Syn. ($4\times$)
Self-Instruct	57.81 ± 1.03	57.08 ± 1.15	54.33 ± 1.20
RFT	58.05 ± 0.79	58.12 ± 0.82	57.68 ± 0.94
Self-Instruct+RFT	57.97 ± 0.88	58.42 ± 0.76	58.06 ± 0.73
SGALM	58.32 ± 0.70	59.45 ± 0.65	$\mathbf{59.89 \pm 0.66}$

5.7 SGALM AS A SYNTHETIC-DATA ENGINE

After fine-tuning, SGALM can generate synthetic samples using Eq. (2) and filter them by $p_\theta^{\text{real}}(z')$. We compare against Self-Instruct, RFT, and Self-Instruct+RFT. To keep generation cost similar, SGALM generates $8\times$ as many candidates as needed and keeps the top $1/8$ by realness score. A smaller Qwen2.5-1.5B-Instruct model is then trained on the real GSM8K training set plus $1\times$, $2\times$, or $4\times$ synthetic samples; the prompt template and representative generation/discrimination cases are given in Appendix C.11.2.

Table 3 shows that mean accuracy rises from 58.32% to 59.89% as retained SGALM data increase from $1\times$ to $4\times$. In contrast, Self-Instruct degrades sharply from 57.81% to 54.33% at $4\times$, falling below the real-data SFT baseline; RFT and Self-Instruct+RFT also stagnate or decline as synthetic data increases. Moreover, SGALM-generated data outperforms both real-data-only SFT (57.12%) and all other synthetic-data baselines at every synthetic scale. This supports the central claim that synthetic-data quantity is not sufficient: without a grounded filter, low-quality or hallucinated samples accumulate. A filtering ablation in Appendix C.7 confirms that the discrimination score is effective for selecting synthetic data: with the same number of SGALM candidates retained, choosing the top $1/8$ by discriminator score outperforms choosing a random $1/8$ significantly.

6 CONCLUSION

We presented SGALM, a self-contained adversarial fine-tuning framework in which one LLM alternates between supervised source discrimination and a discriminator-rewarded policy-gradient update. Under functional-capacity assumptions, the idealized population objective recovers the classical GAN density ratio; under additional explicit latent-domain identifiability conditions, few-shot population alignment is sufficient for zero-shot target alignment. These statements characterize an idealized population optimum rather than convergence of the practical shared-parameter dynamics. Empirically, SGALM improves downstream performance without requiring us to establish that this equilibrium is reached, suggesting that the adversarial interaction is useful as a finite-step fine-tuning mechanism in its own right. Controlled diagnostics show that the learned score tracks verified correctness more strongly than the tested surface variations. SGALM improves over SFT and SPIN on the three main tasks, with broader results and synthetic-data studies supporting its applicability beyond a single benchmark. The results also expose boundaries: parameter sharing is capacity dependent, and synthetic-data gains saturate without careful filtering.

7 AI USE STATEMENT

In this work, we used generative AI tools for assisting in the writing of proofs and implementing methods, providing feedback on research methodology and experiments, and proposing or refining hypotheses. We have not used generative AI tools for helping develop theoretical models or conceptual frameworks, formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting with translation, supporting qualitative and thematic data analysis, interpreting results. And generating synthetic data sets, cleaning and reformatting dataset are not applicable to this work. We have reviewed all AI-assisted work. We checked LLM-generated proofs for potential mathematical error. LLM-generated code was verified and tested for correctness by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

8 Reproducibility Statement

Code is provided at `https://anonymous.4open.science/r/SGALM_Code-4868/`.

References

Kwangjun Ahn, Xiang Cheng, Hadi Daneshmand, and Suvrit Sra. Transformers learn to implement preconditioned gradient descent for in-context learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 45614–45650, 2023.

Ekin Akyürek, Dale Schuurmans, Jacob Andreas, Tengyu Ma, and Denny Zhou. What learning algorithm is in-context learning? investigations with linear models. *arXiv preprint arXiv:2211.15661*, 2022.

Antreas Antoniou, Amos Storkey, and Harrison Edwards. Data augmentation generative adversarial networks. *arXiv preprint arXiv:1711.04340*, 2017.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Yu Bai, Fan Chen, Huan Wang, Caiming Xiong, and Song Mei. Transformers as statisticians: Provable in-context learning with in-context algorithm selection. In *Advances in Neural Information Processing Systems*, volume 36, pp. 57125–57211, 2023.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.

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

Tong Che, Yanran Li, Ruixiang Zhang, R Devon Hjelm, Wenjie Li, Yangqiu Song, and Yoshua Bengio. Maximum-likelihood augmented discrete generative adversarial networks. *arXiv preprint arXiv:1702.07983*, 2017.

Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. Self-play fine-tuning converts weak language models to strong language models. *arXiv preprint arXiv:2401.01335*, 2024.

Wei-Lin Chiang, Zhuohan Li, Ziqing Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E Gonzalez, et al. Vicuna: An open-source chatbot impressing GPT-4 with 90%* ChatGPT quality. *See https://vicuna.lmsys.org*, 2023.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

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. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Yann Dubois, Chen Xuechen Li, Rohan Taori, Tianyi Zhang, Ishaan Gulrajani, Jimmy Ba, Carlos Guestrin, Percy S Liang, and Tatsunori B Hashimoto. AlpacaFarm: A simulation framework for methods that learn from human feedback. In *Advances in Neural Information Processing Systems*, volume 36, pp. 30039–30069, 2023.

Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *International Conference on Machine Learning*, pp. 1180–1189, 2015.

Dongyoung Go, Tomasz Korbak, Germán Kruszewski, Jos Rozen, Nahyeon Ryu, and Marc Dymetman. Aligning language models with preferences through f-divergence minimization. In *International Conference on Machine Learning*, pp. 11546–11583, 2023.

Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, volume 27, 2014.

Caglar Gulcehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek Sharma, Siddhant Jayakumar, Nal Kalchbrenner, Orhan Firat, and Nando de Freitas. Reinforced self-training (ReST) for language modeling. *arXiv preprint arXiv:2308.08998*, 2023.

Jiaxian Guo, Sidi Lu, Han Cai, Weinan Zhang, Yong Yu, and Jun Wang. Long text generation via adversarial training with leaked information. In *AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Or Honovich, Thomas Scialom, Omer Levy, and Timo Schick. Unnatural instructions: Tuning language models with (almost) no human labor. In *Annual Meeting of the Association for Computational Linguistics*, pp. 14409–14428, 2023.

Harrison Lee, Samrat Phatale, Hassan Mansoor, Kellie Ren Lu, Thomas Mesnard, Johan Ferret, Colton Bishop, Ethan Hall, Victor Carbune, and Abhinav Rastogi. RLAIF: Scaling reinforcement learning from human feedback with AI feedback. *arXiv preprint arXiv:2309.00267*, 2023.

Yingcong Li, Muhammed Emrullah Ildiz, Dimitris Papailiopoulos, and Samet Oymak. Transformers as algorithms: Generalization and stability in in-context learning. In *International Conference on Machine Learning*, pp. 19565–19594, 2023.

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. In *Advances in Neural Information Processing Systems*, volume 35, pp. 27730–27744, 2022.

Richard Yuanzhe Pang, Weizhe Yuan, He He, Kyunghyun Cho, Sainbayar Sukhbaatar, and Jason Weston. Iterative reasoning preference optimization. In *Advances in Neural Information Processing Systems*, volume 37, pp. 116617–116637, 2024.

Ethan Perez, Saffron Huang, Francis Song, Trevor Cai, Roman Ring, John Aslanides, Amelia Glaese, Nat McAleese, and Geoffrey Irving Martin. Red teaming language models with language models. *arXiv preprint arXiv:2202.03286*, 2022.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, volume 36, pp. 53728–53741, 2023.

Ilia Shumailov, Zakhar Shumaylov, Yiren Zhao, Nicolas Papernot, Ross Anderson, and Yarin Gal. AI models collapse when trained on recursively generated data. *Nature*, 631(8022):755–759, 2024.

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. *arXiv preprint arXiv:2212.10560*, 2023.

Jingfeng Wu, Difan Zou, Zixiang Chen, Vladimir Braverman, Quanquan Gu, and Peter L Bartlett. How many pretraining tasks are needed for in-context learning of linear regression? *arXiv preprint arXiv:2310.08391*, 2023.

Shiguang Wu, Yaqing Wang, and Quanming Yao. Why in-context learning models are good few-shot learners? In *International Conference on Learning Representations*, 2025.

Wei Xiong, Hanze Dong, Chenlu Ye, Ziqi Wang, Han Zhong, Heng Ji, Nan Jiang, and Tong Zhang. Iterative preference learning from human feedback: Bridging theory and practice for rlhf under kl-constraint. *arXiv preprint arXiv:2312.11456*, 2023.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. WizardLM: Empowering large language models to follow complex instructions. *arXiv preprint arXiv:2304.12244*, 2024.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. SeqGAN: Sequence generative adversarial nets with policy gradient. In *AAAI Conference on Artificial Intelligence*, volume 31, 2017.

Zze Yu, Jia Jaw Lau, Hui Zhang, and Bryan Kian Hsiang Low. Fine-tuning language models with generative adversarial reward modelling. *arXiv preprint arXiv:2305.06176*, 2023.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason E Weston. Self-rewarding language models. In *International Conference on Machine Learning*, 2024.

Yizhe Zhang, Zhe Gan, Kai Fan, Zhi Chen, Ricardo Henao, Dinghan Shen, and Lawrence Carin. Adversarial feature matching for text generation. In *International Conference on Machine Learning*, pp. 4006–4015, 2017.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *IEEE International Conference on Computer Vision*, pp. 2223–2232, 2017.

Table 4: Technical comparison with representative self-improvement and synthetic-data methods. "New z" means that both the input and output may be generated.

Method	External signal	Generation scope	Training signal	Generator update	Main limitation
Self-Instruct Wang et al. (2023)	None	New z	Heuristic filtering	SFT	Error accumulation
ReST Gulcehre et al. (2023)	Reward model	Given x, new y	Reward filtering	SFT/RL	Reward availability
Iterative-DPO Xiong et al. (2023)	Reward model	Given x, new y	Preference pairs	DPO	Reward availability
Iterative-RPO Pang et al. (2024)	Ground-truth answer	Given x, new y	Verified preference	Preference optimization	Verifiable tasks only
Self-Rewarding Yuan et al. (2024)	Self-judge	Given x, new y	LLM-as-a-judge	Preference optimization	Self-evaluation drift
SPIN Chen et al. (2024)	Real responses	Given x, new y	Real $>$ generated	DPO-like	Fixed pairwise ordering
SGALM	**Real dataset**	**New z**	**Learned real/fake score**	**REINFORCE-style**	**Capacity/stability**

A Related Works

We review methods that use LLM-generated training signals and position SGALM relative to self-play, reward-guided iteration, and text GANs. Table 4 summarizes the technical distinctions that are most relevant to our setting.

A.1 Synthetic Data Generation

The paradigm began with Self-Instruct Wang et al. (2023); Honovich et al. (2023), which demonstrated that a weak model could be fine-tuned on instruction-output pairs generated by itself. Subsequent works like Alpaca Dubois et al. (2023) and Vicuna Chiang et al. (2023) popularized this distillation approach. More recent methods focus on increasing data complexity and diversity; for example, WizardLM Xu et al. (2024) employs "Evol-Instruct" to progressively rewrite instructions into more complex forms. However, these methods primarily rely on static distillation from a fixed teacher, lacking a dynamic feedback loop to correct errors during training, and the error would accumulate to collapse Shumailov et al. (2024).

A.2 Self-Play and Iterative Refinement

To enable models to improve themselves without a stronger teacher, Self-Play mechanisms have been introduced. Reinforcement Learning from AI Feedback (RLAIF) Lee et al. (2023); Bai et al. (2022) replaces human preference labeling with model-generated preferences, scaling up the RLHF pipeline. Self-Rewarding Language Models Yuan et al. (2024) integrate the reward modeling capability directly into the policy model, allowing the LLM to act as its own judge during training. Similarly, SPIN Chen et al. (2024) proposes a method where the model plays against its previous iteration. By treating the previous epoch's model as a generator of "negative" samples and the ground truth as "positive," SPIN optimizes a DPO-like objective to widen the gap between the two. Though also inspired by GAN, SPIN makes discrimination by assuming a strict dichotomy where model generations are always inferior to data, potentially penalizing valid but diverse responses. Our work (SGALM) addresses this by introducing a learnable discriminator that assesses quality dynamically rather than relying on a static assumption.

Related to self-play, iterative training methods like ReST Gulcehre et al. (2023), Iterative DPO Xiong et al. (2023) and Iterative RPO Pang et al. (2024) involve repeated cycles of sampling, ranking/filtering, and fine-tuning. While these methods improve performance, they typically rely on fixed reward models or ground-truth verification (e.g., in math problems), which limits their applicability in open-ended domains compared to our adversarial approach.

A.3 Adversarial Training in LLMs

With the rise of LLMs, adversarial concepts have resurfaced with a focus on robustness and alignment rather than generation from scratch. Red Teaming Perez et al. (2022) employs an adversarial LLM to generate test cases that provoke harmful outputs from a target LLM, serving as a distinct "attacker." In the context of alignment, f-GAN formulations have been theoretically connected to divergence minimization in language models Go et al. (2023). Most closely related to our work is the concept of using a discriminator to guide decoding. Methods like Discriminator-Guided Generation train a classifier to rerank or guide the beam search of an LLM. However, these discriminators are usually small, separate models (e.g., BERT-based) or fixed reward models. SGALM differs by unifying the generator and discriminator into a single LLM that iteratively updates both roles, leveraging the

high-level reasoning capabilities of modern LLMs to perform nuanced discrimination beyond simple binary classification.

A.4 TEXT GAN

GANs have achieved immense success in continuous domains like computer vision but have historically struggled with discrete text generation. Early Text GANs. The discrete nature of text disrupts the gradient flow from the discriminator to the generator. Early solutions like SeqGAN Yu et al. (2017), MaliGAN Che et al. (2017), and LeakGAN Guo et al. (2018) employed Reinforcement Learning (specifically Policy Gradient) to bypass the non-differentiable token generation. However, these models famously suffered from mode collapse and training instability Zhang et al. (2017), and generally underperformed compared to maximum likelihood estimation (MLE) in the pre-LLM era.

B THEORETICAL ANALYSIS

For completeness, we adapt the standard population-level GAN analysis of Goodfellow et al. (2014) to SGALM notation. Propositions B.1 and B.2 are classical results and are not claimed as new contributions; they provide intermediate statements for the subsequent ICL theorem.

We rely on the following standard idealizations for this population analysis. Functional capacity: The feasible generator and discriminator functions include p_T and the pointwise optimal discriminator, respectively, and the two functions are jointly realizable when parameters are shared. Discrete generation: Text distributions are probability mass functions on a countable sequence space; where policy gradients are discussed, $p_\theta^G(z)$ is differentiable with respect to θ on its support. Optimization scope: We characterize an attained global minimax optimum.

B.1 OPTIMAL DISCRIMINATION BY POSTERIOR PROBABILITY

We first analyze the behavior of the discrimination update, i.e., updating (4) with (5). This process is mathematically equivalent to binary classification with cross-entropy. Under equal class weights, pointwise optimization gives the posterior probability of the real class.

Proposition B.1. *For a fixed generator distribution $p_G(z)$ and equal real/fake class weights, the optimal discriminator trained via* (4) *is, for every z with $p_T(z) + p_G(z) > 0$,*

$$D^*(z) = \frac{p_T(z)}{p_T(z) + p_G(z)} \tag{10}$$

Proof. The discriminator $D(z)$ is derived from the probability of the token "Real". Training the model to distinguish between real samples $z \sim p_T$ and fake samples $z' \sim p_\theta$ corresponds to maximizing the standard binary log-likelihood objective $J(D)$:

$$J(D) = \mathbb{E}_{z\sim p_T}[\log D(z)] + \mathbb{E}_{z'\sim p_G}[\log(1 - D(z'))]$$

We can rewrite this expectation as a sum over the countable sequence space $\mathcal{Z}$:

$$J(D) = \sum_{z\in\mathcal{Z}} \left(p_T(z) \log D(z) + p_G(z) \log(1 - D(z))\right).$$

To find the optimal $D(z)$ for any point z, we differentiate the integrand with respect to $D(z)$ and set the derivative to zero:

$$\frac{p_T(z)}{D(z)} - \frac{p_G(z)}{1 - D(z)} = 0$$

Rearranging the terms yields:

$$D^*(z) = \frac{p_T(z)}{p_T(z) + p_G(z)}$$

□

This result identifies the ideal discriminator with the real-class posterior induced by the real and generated distributions.

B.2 Optimal Generation from True Distribution

Next, we characterize the global optimum of the population minimax envelope in which the discriminator is re-optimized for each candidate generator.

Proposition B.2. *Let $C(p_G) = \max_D J(D, p_G)$ denote the population minimax envelope. Its global minimum is achieved if and only if the generated distribution matches the real distribution, i.e., $p_{G^*}(z) = p_T(z)$.*

Proof. Substituting the re-optimized discriminator $D^*(p_G)$ into the value function gives

$$C(p_G) = \mathbb{E}_{z\sim p_T}\left[\log \frac{p_T(z)}{p_T(z) + p_G(z)}\right] + \mathbb{E}_{z\sim p_G}\left[\log \frac{p_G(z)}{p_T(z) + p_G(z)}\right]$$

This expression relates to the Jensen-Shannon Divergence (JSD). By factoring out $-\log 4$, we obtain:

$$C(p_G) = -\log 4 + 2 \cdot \text{JSD}(p_T \| p_G)$$

Since the Jensen-Shannon Divergence is non-negative ($\text{JSD}(P\|Q) \geq 0$) and equals zero if and only if $P = Q$, the global minimum of the generator objective is achieved exactly when:

$$p_{G^*}(z) = p_T(z)$$

□

Thus $p_G = p_T$ uniquely characterizes the global optimum of the idealized minimax envelope.

B.3 From Few-Shot Generation to Zero-Shot Understanding

Theorem B.3 (Few-shot equilibrium implies zero-shot target alignment)**.** *Suppose Assumptions 4.1–4.2 hold. If the idealized population adversarial objective attains an equilibrium θ^* with $q_{\theta^*} = p_T$, then T is the unique maximizer of π_{θ^*} and $p^0_{\theta^*}(\cdot) = p_T(\cdot)$.*

Proof. For $h \in \mathcal{H}$, define the probability that the MAP selector chooses domain h under a random target context,

$$w_h \equiv \Pr_{C\sim p_T^n}[h_{\theta^*}(C) = h].$$

The deterministic tie-breaking rule makes $w = (w_h)_{h\in\mathcal{H}}$ a probability vector. By Assumption 4.1 and the law of total probability, the context-marginal generator is

$$q_{\theta^*} = \mathbb{E}_{C\sim p_T^n}[p_{h_{\theta^*}(C)}] = \sum_{h\in\mathcal{H}} w_h p_h.$$

At the stipulated population equilibrium $q_{\theta^*} = p_T$. Mixture identifiability in (4.2) therefore gives $w_T = 1$. Hence

$$h_{\theta^*}(C) = T \quad \text{for } p_T^n\text{-almost every } C. \tag{11}$$

It remains to identify the no-demonstration prior mode. Fix any $h \neq T$ and let

$$B_h = \left\{ C : \prod_{i=1}^n p_h(z^i) > \prod_{i=1}^n p_T(z^i) \right\}.$$

By Assumption 4.2, $p_T^n(B_h) > 0$. Intersecting B_h with the probability-one event in Eq. (11) still has positive probability. Because the sequence space is countable, choose a context C in this intersection with $p_T^n(C) > 0$. The MAP selector chooses T on this context, so

$$\begin{aligned}\pi_{\theta^*}(T)\prod_{i=1}^n p_T(z^i) &\geq \pi_{\theta^*}(h)\prod_{i=1}^n p_h(z^i) \\ &> \pi_{\theta^*}(h)\prod_{i=1}^n p_T(z^i).\end{aligned}$$

The choice $p_T^n(C) > 0$ means that the target likelihood product is positive; dividing it out yields $\pi_{\theta^*}(T) > \pi_{\theta^*}(h)$. Because $h \neq T$ was arbitrary and $\mathcal{H}$ is finite, T is the unique maximizer of π_{θ^*}.

Finally, applying the empty-context clause of Assumption 4.1 gives

$$p^0_{\theta^*}(\cdot) = p_{h_{\theta^*}(\emptyset)}(\cdot) = p_T(\cdot),$$

which proves the claim. □

C Experiments

C.1 Matched Six-Benchmark Evaluation

We fine-tune Qwen2.5-3B-Instruct on UltraChat200k and evaluate Base, SPIN, and SGALM on the six benchmarks used by SPIN. All three models use the same evaluation configuration.

Table 5: Matched evaluation on the six-benchmark suite used by SPIN. SGALM improves the average and leads on four of six tasks.

Method	ARC-C	TruthfulQA	WinoGrande	GSM8K	HellaSwag	MMLU	Avg.
Base	48.0	48.9	62.2	67.9	65.5	65.1	59.6
SPIN	51.0	53.2	**62.5**	69.6	67.7	**65.9**	61.6
SGALM	**52.9**	**55.0**	62.1	**71.9**	**68.6**	65.5	**62.7**

C.2 Paired Statistical Comparisons

For each main-task comparison, we report the percentage-point difference and a 95% paired bootstrap confidence interval. Bootstrap replicates resample training runs and then paired test items; two-sided p-values are computed from the resulting distribution of paired differences. This paired analysis is distinct from the marginal standard deviations in Table 1.

Table 6: Paired differences between SGALM and baselines/variants. Brackets contain 95% confidence intervals.

Task	SGALM–SFT	SGALM–SPIN	SGALM–G-only	SGALM–non-shared
GSM8K	$+4.10\ [2.40, 5.80]$	$+3.42\ [1.67, 5.17]$	$+2.73\ [0.73, 4.72]$	$+1.06\ [-0.54, 1.96]$
ARC-C	$+1.63\ [0.07, 3.63]$	$+4.17\ [2.02, 6.32]$	$+2.44\ [0.59, 4.29]$	$+1.37\ [-0.68, 3.42]$
MBPP	$+3.50\ [1.00, 6.00]$	$+3.89\ [1.39, 6.39]$	$+1.56\ [-0.39, 3.51]$	$-1.16\ [-3.36, 1.04]$

C.3 Results on Llama

These experiments provide cross-backbone and cross-scale component checks. Both update types transfer across backbones, but the benefit of parameter sharing depends on capacity: the non-shared 3B variant is strongest on MBPP, whereas shared SGALM is strongest on all three tasks for the 8B model.

Table 7: Cross-backbone and cross-scale results on Llama models.

Model	Method	GSM8K	ARC	MBPP
Llama-3.2-3B-Instruct	Base	62.8	55.9	67.3
	SFT	64.2	57.8	66.5
	SPIN	65.1	56.4	68.1
	SGALM (D-only)	65.0	57.0	68.4
	SGALM (G-only)	66.5	58.2	69.6
	SGALM (non-shared)	68.2	58.7	**72.0**
	SGALM	**68.5**	**59.1**	71.2
Llama-3.1-8B-Instruct	Base	73.3	68.8	52.4
	SFT	73.5	68.8	57.6
	SPIN	74.1	69.1	56.8
	SGALM (D-only)	74.2	69.0	57.9
	SGALM (G-only)	75.4	69.3	59.5
	SGALM (non-shared)	75.8	69.3	60.1
	SGALM	**77.0**	**69.8**	**61.2**

C.4 Results on Open-Ended Instruction Following Tasks

Table 8: IFEval instruction-following results on Qwen2.5-3B-Instruct.

Method	Inst. loose	Inst. strict	Prompt loose	Prompt strict
Base	65.3	59.6	54.2	46.0
SFT	66.5	60.8	56.2	47.2
SPIN	66.7	60.6	56.8	47.6
SGALM	**68.6**	**62.0**	**58.0**	**49.4**

We train on 5.51k examples from HuggingFaceH4/ifeval-like-data and evaluate on IFEval. SGALM scores highest on all four instruction-following metrics in Table 8, including the strict prompt-level measure, where it scores 49.4 versus 47.6 for SPIN.

C.5 Discriminator Diagnostics

For the correctness diagnostic in Table 2, we select 25 held-out prompts per dataset, sample four responses to each prompt, and evaluate correctness with the task-specific oracle. Holding the prompt fixed controls its topic and surface style. For the perturbation diagnostic, we start from held-out real samples: semantic corruptions substitute the final GSM8K number or mutate an MBPP code line, while surface controls alter formatting without changing the verified answer or program behavior. These diagnostics contain 100 generated responses per dataset and should be interpreted as targeted tests rather than exhaustive calibration studies.

At initialization, the average real/generated scores are $0.9923/0.9854$. After one SGALM iteration, the real-sample mean remains above 0.9 while the generated-sample mean falls below 0.2. Seen and held-out real samples have similar score distributions in Figure 3. Table 2 then tests whether the score also changes with answer correctness and formatting.

C.6 Hyperparameter Sensitivity

Table 9 changes one factor at a time for Qwen2.5-3B-Instruct on GSM8K. Removing reward normalization lowers accuracy by 2.32 points. Accuracy changes by less than a point across the three tested temperatures, while a 1:2 discriminator-to-generator update ratio gives the highest accuracy among the tested ratios.

Table 9: One-factor sensitivity study on GSM8K.

Factor	Setting	Accuracy
Reward normalization	Within-batch (default)	72.71
	None	70.39
Sampling temperature	0.6	**73.05**
	0.7 (default)	72.71
	0.8	72.41
D:G update ratio	1:2	**73.26**
	1:1 (default)	72.71
	2:1	71.90

C.7 Synthetic-Data Scaling and Filtering

We train Qwen2.5-1.5B-Instruct on the real GSM8K training set plus synthetic data. Each $\pm$ entry in Table 10 is the mean and standard deviation over three independent runs of generation and training.

Table 10: Downstream GSM8K accuracy as the retained synthetic-data scale increases.

SFT training data	$1\times$	$2\times$	$4\times$
Real only		57.12 ± 0.59	
Real + Self-Instruct+RFT	57.97 ± 0.88	58.42 ± 0.76	58.06 ± 0.73
Real + SGALM (top $1/8$)	58.32 ± 0.70	59.45 ± 0.65	$\mathbf{59.89 \pm 0.66}$

Table 11 compares selecting the top $1/8$ of each candidate pool by discriminator score with selecting a random $1/8$. Both rows retain the same number of examples for downstream SFT. Score selection yields 58.32%, 59.45%, and 59.89% accuracy, versus 56.82%, 56.17%, and 55.03% for random selection. The difference grows from 1.50 to 4.86 points across the three pool sizes. Using all candidates gives lower accuracy, with a larger downstream training set.

Table 11: Filtering ablation on GSM8K (accuracy in %). Score and random filters each retain $1/8$ of the generated candidates; the unfiltered row uses all candidates.

Total candidates	$8\times$	$16\times$	$32\times$
Top-$1/8$ retained	58.32	59.45	59.89
Random-$1/8$ retained	56.82	56.17	55.03
All candidates	54.95	52.03	48.48

C.8 Implementation Details

Here we provide implementation details and important hyperparameter settings. For each iterative method—SGALM, Self-Rewarding, SPIN, Iterative-RS, and Iterative-RPO—each iteration generates as many synthetic samples as the real training set (7.47/1.12/0.12k for GSM8K/ARC/MBPP) and performs one training epoch. We use batch size 64 and select the initial learning rate from $\{1 \times 10^{-6}, 5 \times 10^{-7}, 3 \times 10^{-7}\}$. SFT and Self-Instruct use AdamW; policy- or preference-based updates use RMSProp. SGALM standardizes detached discriminator rewards within each generator batch.

For SGALM and all ICL-based baselines, we independently sample four training examples to construct each Z_{ctx}.

C.9 Training Cost

All timing comparisons use 4 $\times$ NVIDIA A100-80GB GPUs. On GSM8K, SPIN takes 19 minutes for generation and 63 minutes for training per iteration, while SGALM takes 23 and 68 minutes, respectively. The totals are 82 versus 91 minutes, so SGALM incurs approximately 11% additional

wall-clock time relative to SPIN. Absolute SGALM times per iteration are approximately 0.1 hours for generation and 0.3 hours for training on both ARC and MBPP.

C.10 More Results

C.10.1 Generation Distribution

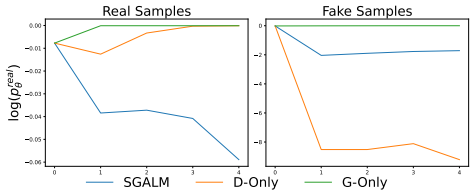


Figure 4: $\log(p_\theta^{\text{real}})$ across iterations for real and generated samples. The separation after the first iteration shows that the discriminator learns a nontrivial real/fake distinction, while generated samples still have usable relative scores for standardized generation rewards.

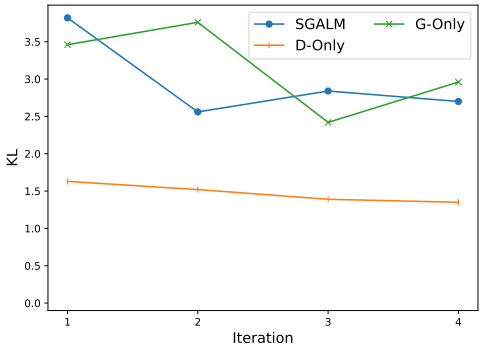


Figure 5: $\mathrm{KL}(p^G_{\theta^i} || p^G_{\theta^{i-1}})$ for each iteration i.

Figure 5 shows the KL divergence between the generation distributions at iterations i and $i-1$, $\mathrm{KL}(p^G_{\theta^i} || p^G_{\theta^{i-1}})$. The KL-divergence is estimated by $\frac{1}{N}\sum_{z\sim p^G_{\theta^{i-1}}}^{N} \frac{p^G_{\theta^i}(z)}{p^G_{\theta^{i-1}}(z)} \log \frac{p^G_{\theta^i}(z)}{p^G_{\theta^{i-1}}(z)} - \frac{p^G_{\theta^i}(z)}{p^G_{\theta^{i-1}}(z)} + 1$. D-only changes the generation distribution despite having no explicit generator objective: the discriminator update changes the parameters shared by both roles. Figure 4 separately shows how real and generated samples receive different discriminator scores after training begins.

C.11 Generation and Discrimination Cases

Here we provide some cases about the generation and discrimination.

C.11.1 Prompt Template

First,we provide the exact prompt we use in SGALM (specifically for Qwen2.5-3B-Instruct). The generation prompt is provided as follows, where blue means official chat template provided by Qwen2.5-3B-Instruct, brown means the prompt we set for SGALM shared across all data, and the others (black) are specific data samples.

```
<|im_start|>system
You are a helpful assistant. Follow the user's
examples to complete the task.<|im_end|>



<|im_start|>user
```

```
Here are 4 examples. Please follow the pattern and
provide a new example.

### Example 1:
Q: The bakery has 8 indoor tables and 12 outdoor
tables. Each indoor table has 3 chairs and each
outdoor table has 3 chairs. How many chairs are
there in total?
A: Let's think step by step. There are 8 x 3 =
<<8*3=24>>24 indoor chairs.
There are 12 x 3 = <<12*3=36>>36 outdoor chairs.
In total there are 24 + 36 = <<24+36=60>>60 chairs
#### 60

### Example 2:
Q: Ken had fifty pencils, and he wanted to share
some of them with his two friends, Manny and
Nilo. Ken gave ten pencils to Manny and ten more
pencils to Nilo than he gave to Manny. He kept
the rest of the pencils. How many pencils did Ken
keep?
A: Let's think step by step. Nilo received 10 +
10 = <<10+10=20>>20.
Thus, Ken gave a total of 10 + 20 =
<<10+20=30>>30.
Therefore, Ken kept 50 - 30 = <<50-30=20>>20
pencils.
#### 20

### Example 3:
Q: A small store made 50 posters to advertise
their products. Two-fifths of them are small
posters while half of them are medium posters.
The rest are large posters. How many large
posters are there?
A: Let's think step by step. 50 x 2/5 =
<<50*2/5=20>>20 posters are small.
And 50/2 = <<50/2=25>>25 posters are medium.
So 20 + 25 = <<20+25=45>>45 posters are not large.
Therefore, 50 - 45 = <<50-45=5>>5 posters are
large.
#### 5

### Example 4:
Q: Earl started delivering newspapers on the
first floor of a condominium building. He then
went up 5 floors then went down 2 floors. He
again went up 7 floors and found that he is 9
floors away from the top of the building. How
many floors does the building have?
```

```
A: Let's think step by step. Earl was on the 1 +
5 = <<1+5=6>>6th floor after going up 5 floors.
When he went down 2 floors, he was on the 6 - 2 =
<<6-2=4>>4th floor.
Since he went up 7 floors, he was then on the 4 +
7 = <<4+7=11>>11th floor.
Since he is 9 floors away from the top of the
building, therefore the building has 11 + 9 =
<<11+9=20>>20 floors.
#### 20

### New Example:
<|im_end|>



<|im_start|>assistant
```

And the discrimination prompt is as follows.

```
<|im_start|>system
You are a content detector. Follow the user's
instruction to analyze the given text to determine
if it is Real (human-written) or Fake
(LLM-generated). Answer with exactly one word:
'Real' (if human-written) or 'Fake' (if
LLM-generated).<|im_end|>



<|im_start|>user
Q: Francie saves up her allowance for several
weeks. She receives an allowance of $5 a week for
8 weeks. Then her dad raises her allowance, and
she receives $6 a week for 6 weeks. Francie uses
half of the money to buy new clothes. With the
remaining money, she buys a video game that costs
$35. How much money does Francie have remaining
after buying the video game?
A: Let's think step by step. When her allowance
is $5 a week, Francie gets a total of $5 * 8 =
$<<5*8=40>>40
When her allowance is $6 a week, Francie gets a
total of $6 * 6 = $<<6*6=36>>36
The total amount of money Francie gets is $40 +
$36 = $<<40+36=76>>76
After purchasing new clothes, she has $76 / 2 =
$<<76/2=38>>38 remaining
After buying the video game, Francie has $38 -
$35 = $<<38-35=3>>3 remaining
#### 3
```

```
Is above text Real (human-written) or Fake
(LLM-generated)? Answer with exactly one word:
'Real' (if human-written) or 'Fake' (if
LLM-generated).
<|im_end|>


<|im_start|>assistant
```

C.11.2 Generation and Discrimination Cases

We show case study on GSM8K. We use iteration 0 model (original Qwen2.5-3B-Instruct) and iteration 4 model to generate and discriminate 320 cases respectively. We also provide discrimination cases on the real samples (seen in train set).

For iteration 0 model, the generated/real cases has scores (p_θ^{real}) with mean $0.9570/0.9816$, median $1.0/1.0$, std $0.1764/0.1060$, minimum $0.0052/0.0000$, maximum $1.0/1.0$. One generated case with highest score (1.0) is

```
Q: Sarah earns $15 for each hour she works at her
job. She also gets a $20 weekly bonus. If Sarah
works 40 hours in a week, how much money will she
earn including her bonus?\n\nLet's think step by
step. Sarah earns $15 per hour and works 40
hours, so her earnings from hourly work would be
40 * 15 = $<<40*15=600>>600.\nAdditionally, she
receives a $20 weekly bonus.\nTherefore, the
total amount Sarah earns is 600 + 20 =
$<<600+20=620>>620.\n#### 620
```

The generated case with lowest score (0.0052) is

```
Q: A jar contains 50 red marbles and 30 blue
marbles. If a marble is randomly selected from
the jar, the probability that it is blue is twice
the probability that it is red. How many
additional blue marbles need to be added to the
jar so that the probability of selecting a blue
marble becomes equal to the probability of
selecting a red marble?\n\nA: Let's think step by
step. Initially, there are 50 red marbles and 30
blue marbles, making a total of 50 + 30 = 80
marbles.\n\nLet \\( x \\) be the number of
additional blue marbles to be added.\n\nAfter
adding \\( x \\) blue marbles, the total number
of marbles becomes \\( 80 + x \\).\n\nThe number
of blue marbles will be \\( 30 + x \\).\n\nGiven
that the probability of selecting a blue marble
is twice the probability of selecting a red
marble, we can set up the following
equation:\n\n\\[\n\\frac{30 + x}{80 + x} = 2
\\times \\frac{50}{80 + x}\n\\]\n\nSimplify the
right-hand side:\n\n\\[\n\\frac{30 + x}{80 + x} =
\\frac{100}{80 + x}\n\\]\n\nCross-multiply to
solve for \\( x \\):\n\n\\[\n(30 + x)(80 + x) =
100(80 + x)\n\\]\n\nExpand both
sides:\n\n\\[\n2400 + 30x + 80x + x^2 = 8000 +
100x\n\\]\n\nCombine like terms:\n\n\\[\nx^2 +
110x + 2400 = 8000 + 100x\n\\]\n\nSubtract \\(
8000 + 100x \\) from both sides:\n\n\\[\nx^2 +
110x + 2400 - 8000 - 100x =
0\n\\]\n\nSimplify:\n\n\\[\nx^2 + 10x - 5600 =
0\n\\]\n\nSolve this quadratic equation using the
quadratic formula \\( x = \\frac{-b \\pm
\\sqrt{b^2 - 4ac}}{2a} \\), where \\( a =
```

A real case with very low score 1.583×10^{-4} is

```
Q: By the time Anne is two times as old as Emile,
Emile will be six times as old as Maude. If Maude
will be 8 years old, how old will Anne be?\nA:
Let's think step by step. If Maude's age is 8 by
the time Anne's age is four times Emile's age,
Emile will be six times as old as Maude, which
totals 6*8 = 48 years.\nIf Emile's age is 48
years old by the time Anne's age is twice her
number, Anne will be 2*48 = <<48*2=96>>96
years.\n#### 96
```

Another real case with very low score 1.230×10^{-4} is

```
Q: Jenny leaves her house at 8:00 with some
cupcakes for her daughter's birthday party at
school. She jogs to the school at 15 miles per
hour. Half an hour later, her wife Anna realizes
that Jenny used peanut butter frosting because
she forgot one of the kids in the class had a
severe peanut allergy. Jenny doesn't have her
cell phone, so Anna leaves the house driving at
45 miles her hour to catch Jenny and warn her.
How long does Anna spend traveling in
minutes?\nA: Let's think step by step. We know
that the time that Anna spends traveling is 20
minutes less than Jenny's travel time: a = j -
20\nWe also know that the distance both people
travel is equal, so their speeds times their
travel times must be equal: 15j =
45a\nSubstituting the first equation into the
second equation, we get 15j = 45(j -
20)\nMultiplying through the parentheses we get
15 j = 45j - 900\nSubtracting 45 j from both
sides, we get -30j = -900\nDividing both sides by
-30, we get j = 30\nSince Anna travels 20 minutes
less than Jenny, she traveled 30 minutes - 20
minutes = <<20-10=10>>10 minutes\n#### 10
```

For iteration 4 model, the generated/real cases has scores (p_θ^{real}) with mean $0.0699/0.9434$, median $0.0179/0.9922$, std $0.1245/0.1282$, minimum $0.0000/0.0015$, maximum $0.9062/1.0$. The generated case with highest score (0.9062) is

```
Q: A school garden club plants flowers in rows.
They planted 5 rows of tulips with 6 tulips in
each row last week. This week, they decided to
increase the number of rows by 2 but kept the
same number of tulips per row. How many tulips
will they plant this week?\n\nLet's think step by
step. Last week, the club planted 5 rows of
tulips with 6 tulips in each row, so they planted
5 * 6 = <<5*6=30>>30 tulips.\nThis week, they
increased the number of rows by 2, so they will
have 5 + 2 = <<5+2=7>>7 rows.\nThey still plan to
plant the same number of tulips per row, which is
6 tulips.\nTherefore, this week they will plant 7
* 6 = <<7*6=42>>42 tulips.\n#### 42
```

The generated case with lowest score (4.7981×10^{-6}) is

```
Q: A factory produces 3 types of widgets: Type A,
Type B, and Type C. The production ratio of Type
A to Type B to Type C is 5:3:2. If the factory
produced a total of 360 widgets today, how many
more Type C widgets must be added to reach a
production ratio of 5:3:4?\n\nLet's think step by
step. First, we determine the total parts of the
original ratio: 5 + 3 + 2 = 10 parts.\nGiven that
the total production is 360 widgets, each part
corresponds to 360 / 10 = <<360/10=36>>36
widgets.\nFor the original ratio of 5:3:2, the
number of Type A, Type B, and Type C widgets
produced are:\nType A: 5 * 36 =
<<5*36=180>>180\nType B: 3 * 36 =
<<3*36=108>>108\nType C: 2 * 36 =
<<2*36=72>>72\n\nTo achieve a new ratio of 5:3:4,
let's denote the new number of Type C widgets as
\\(x\\). This means the number of Type A and Type
B widgets will remain the same (180 and 108
respectively), and the new ratio equation will
be:\n\\[ \\frac{180}{108} = \\frac{180}{x} =
\\frac{108}{x} \\]\n\nWe know that the new ratio
should be 5:3:4, so:\n\\[ \\frac{180}{72+x} =
\\frac{180}{x} = \\frac{108}{x} \\]\n\nSolving
for \\(x\\):\n\\[ \\frac{180}{72+x} =
\\frac{180}{x} \\]\n\\[ 180x = 180(72 + x)
\\]\n\\[ 180x = 180 \\times 72 + 180x \\]\n\\[
180x = 12960 + 180x \\]\n\\[ 0 = 12960 \\]\n\nIt
seems there was a m
```

A real case with very low score 0.0015 is

```
Q: There are twice as many cows in Devonshire as
there are hearts on a standard deck of 52 playing
cards. If there are 4 hearts on a card, calculate
the total cost of the cows when they are sold at
$200 each.\nA: Let's think step by step. If there
are 4 hearts on a card, a standard deck of 52
playing cards will have 208 hearts.\nThere are
twice as many cows in Devonshire as there are
hearts on a standard deck of 52 playing cards,
meaning there are 2*208 = <<2*208=416>>416 cows
in Devonshire.\nThe total cost of the cows when
they are sold at $200 each is 416*$200
=$<<416*200=83200>>83200\n#### 83200
```

Another real case with very low score 0.0019 is

```
Q: Porter is a painter who creates beautiful
paintings of wild animals.  Most recently, he
painted a mural of a pack of wolves standing on a
snowy-white mountainside underneath a full moon.
He put the painting up for auction and it sold
for $1000 less than five times more than he had
made on his previous painting.  If he received
$44,000 for the sale of his most recent painting,
how much, in dollars, did he make for selling his
previous painting?\nA: Let's think step by step.
If $44,000 is $1000 less than five times more
than he had made on his previous painting, then
$44,000+$1000 = $45,000 is five times what he
made on his previous painting.\nIf $45,000 is
five times what he made for his previous
painting, then he made $45,000/5 =
$<<45000/5=9000>>9,000 for selling his previous
painting.\n#### 9,000
```

C.12 No Mode Collapse Phenomenon Observed

At iteration 4, no two generated samples in the inspected set are identical. One phenomenon to note is that the generated sample is likely to share a few common prefixes. For example, 28% of generated samples (iteration 4 model) start with "`Q: A bakery`", but as the context grow longer, they gradually become distinguishable: 10% of generated samples start with "`Q: A bakery makes`" and 1.25% of generated samples start with "`Q: A bakery makes cupcakes`". This might be explained as the consequence of the common pattern of the few-shot examples from the same training set, and memorized knowledge in the model. Because the iteration 0 model also shows similar phenomenon.

The samples share short openings but differ as the generated text grows. We do not observe identical full sequences in this set. This might be interpreted as being consistent with context-dependent domain selection in Assumption 4.1: the prompt induces shared target structure while the sampled continuations remain diverse.