



Learning From Its Own Answers

27 fine-tunes of small open models, and why the data's source mattered more than the adapter

Paper: PT-R-2026-039 v1.0

Author: PureTensor Ltd

Date: 25 October 2026

Status: Published

Abstract

We fine-tuned small open models 27 times, to learn which choices move a model and which do not. Every run used one task, GSM8K grade-school mathematics, and one evaluation: the full GSM8K test set, plus three measures of what the training cost elsewhere. Those were zero-shot log-likelihood accuracy on ARC-Challenge and ARC-Easy, which needs no answer parsing, and compliance with an instruction to answer ARC questions with a letter. Twenty runs share one starting point, Qwen2.5-0.5B-Instruct, so the comparisons among them are paired on the same test items. Nothing here is new: these are standard methods on a standard task at small scale, and the paper is a careful record of what they did.

Supervised fine-tuning on GSM8K's reference solutions lowered the instruct model's accuracy from 46.9% to between 22.2% and 39.5%, every run significantly. Adapter rank from 2 to 64, DoRA, the target modules, a tenfold lower learning rate, a second epoch, NEFTune and full fine-tuning all landed inside that band. Apart from one overscaled configuration, the same runs lost between 0.9 and 2.6 points of ARC-Challenge log-likelihood accuracy and most of the model's compliance with the letter instruction (70.2% for the untuned model, 3.1–25.1% after LoRA).

Training on the model's own correct samples (rejection-sampling fine-tuning, RFT) was the only method that did not lose task accuracy: 48.4%, a difference from the untuned model that is not significant ($p = 0.26$). It also left knowledge and compliance unchanged. DPO on self-generated pairs and a short GRPO run likewise kept knowledge and compliance close to the untuned model's, but fell to 41.2% and 42.1% on GSM8K. GRPO, rewarded for a GSM8K answer format, raised letter compliance on ARC to 99.8%, and in its first 30 steps collapsed onto that format reward before recovering without intervention.

Three measurement problems would each have changed a conclusion, and we report them: a generation cap that truncated 12.6% of the untuned model's answers and, while it was in place, made RFT look like a significant gain; a first-character answer parser that counts a reply beginning "A planet ..." as option A; and evaluation nondeterminism of up to 2.9 points on identical weights.

1. Scope



A team that fine-tunes models has to choose an adapter type, a rank, target modules, a learning rate, a data source and a training objective, and most published guidance studies one of these at a time on one model. We wanted one grid, on one task and one harness, that shows the size of each choice relative to the others, together with what each costs on knowledge and behaviour the task does not touch.

We kept the models small (135M to 1.7B parameters) so that each run took minutes, and the task standard so that the numbers can be checked against others' work. Every method is from the literature (Section 12). The contribution is a controlled comparison and an honest account of its measurement.

2. Models, Data and Training

Models. Qwen2.5-0.5B-Instruct is the reference model, and 20 runs start from it. Other runs start from the pretrained Qwen2.5-0.5B, from Qwen2.5-1.5B-Instruct, and from SmolLM2 at 135M, 360M and 1.7B (instruct variants).

Training data. The GSM8K training split: 7,473 problems with worked solutions. For supervised fine-tuning the target is the reference solution with its calculator annotations (such as `<<48/2=24>>`) removed, ending in the dataset's `#### <answer>` line. Every prompt is the model's chat template around one user turn, "Solve the problem. Reason step by step, then give the final answer on its own line as '#### <number>'.", followed by the problem. The loss covers the completion only.

Reference recipe. LoRA rank 16, alpha 32, dropout 0.05, on all linear projections; learning rate $2e-4$ with a cosine schedule and 3% warmup; batch 16; one epoch (468 steps); maximum length 1,024 tokens; bf16; seed 42. Every other run changes one thing, as listed in Table 1.

Run	Change from the reference recipe	Trainable params	Steps
LoRA r16	none (seeds 42, 1 and 2)	8.8M	468
LoRA r2; r64	rank 2 or 64, alpha = $2r$	1.1M; 35.2M	468
DoRA r16	weight-decomposed LoRA	9.1M	468
q/v only; MLP only	LoRA on the q and v projections; or on gate, up and down	1.1M; 6.6M	468
lr $2e-5$	learning rate ten times lower	8.8M	468
2 epochs	two passes over the data	8.8M	936
500 examples	500 random problems, three epochs	8.8M	96
NEFTune	noise on input embeddings, alpha 5	8.8M	468
Replay	3,000 general chat examples mixed in	8.8M	655
PiSSA r16	adapter initialised on principal singular vectors	8.8M	468
rsLoRA r64	rank-stabilised scaling, alpha 128 or 16	35.2M	468
Full fine-tune	all weights, lr $1e-5$, gradient checkpointing	494.0M	468
DPO	324 self-generated preference pairs, beta 0.1, lr $5e-5$, batch 8	8.8M	41
RFT	2,293 of the model's own correct samples	8.8M	144
GRPO	group-relative policy optimisation, lr $5e-5$	8.8M	200

Table 1. The 20 runs on Qwen2.5-0.5B-Instruct (17 rows; three seeds and two rsLoRA settings share rows). All use LoRA r16 unless stated.



On-policy data. For RFT and DPO we sampled the untuned model four times (temperature 0.8, top-p 0.95) on 3,000 randomly chosen training problems. Of the 12,000 samples, 51.1% were correct; 76.4% of problems had at least one correct sample and 25.7% had four. RFT kept one correct sample for each of the 2,293 problems that had one, and appended a ##### <answer> line where the sample lacked it, since the untuned model used the requested format in only 10.6% of samples. DPO paired a correct sample that used the format with an incorrect sample; because the format was rare, only 324 problems offered such a pair. The replay data are the first 3,000 single-turn examples under 3,000 characters from the smol-magpie-ultra subset of SmolTalk.

GRPO. The reward was 1.0 for a correct final number plus 0.2 for including #####, with no KL penalty. Each step drew 8 completions (temperature 0.9, at most 320 tokens) for each of 4 prompts, so 200 steps covered 800 problems, about 11% of the training set. Other settings were the library's defaults.

Other models. On Qwen2.5-1.5B-Instruct we trained the reference LoRA and a QLoRA counterpart (4-bit NF4 base with double quantisation, bf16 compute, gradient checkpointing). The pretrained Qwen2.5-0.5B and the SmoLLM2 models received the reference LoRA, and SmoLLM2-360M also a full fine-tune at lr 2e-5. SmoLLM2-1.7B used batch 8 with two accumulation steps.

Compute and software. Training ran on two single-GPU cloud instances, one NVIDIA A100 (40 GB) and one NVIDIA A10 (24 GB), so wall times are not comparable across runs; Table 1 gives steps instead. Each run took under 10 minutes. The second evaluation pass (Section 3) ran on a third instance, also an A10. Training used PyTorch 2.8, Transformers 4.57, PEFT 0.21 and TRL 1.16, except GRPO, which used TRL 0.24 because the later trainer needs a newer vLLM than the instances' CUDA 12.8 driver supports. Generation used vLLM 0.11.

3. Evaluation

Task accuracy. The full GSM8K test set (1,319 problems), greedy decoding, the training prompt, scored by exact match on the number after #####, or on the last number in the reply when ##### is absent. Adapters were merged into the base weights before serving. Our first pass capped generation at 400 tokens. That truncated 166 of the untuned model's answers, against 12 to 46 for the supervised runs, which handicapped the baseline. GSM8K figures for the Qwen2.5-0.5B-Instruct runs therefore come from a second pass at 1,024 tokens, where the untuned model truncated 12 answers. The other models received only the first pass (Section 8).

Knowledge retained. Zero-shot multiple-choice log-likelihood on the ARC-Challenge (1,172 questions) and ARC-Easy (2,376) test sets, following the evaluation-harness recipe: context "Question: {question}\nAnswer:", each option as a continuation, no chat template, and the option with the highest log-probability per byte chosen (acc_norm). It involves no generation and no parsing, so it measures what the weights prefer, not how the model chooses to reply.

Instruction compliance. The ARC-Challenge questions again, generated through the chat template with the instruction "Answer the multiple-choice question with the letter of the correct option only." (greedy, up to 200 tokens). A reply complies if it opens with an option label followed by punctuation, a line break or nothing: "B", "B.", "(B)". Section 10 explains why we rejected two simpler parsers.

Statistics. Intervals are Wilson 95% intervals. Differences from a run's own starting model are tested with the exact two-sided McNemar test on per-item outcomes, which uses the pairing of items. We report more than 100 such tests, so isolated p-values between 0.01 and 0.05 are weak evidence, and we say so where a reading depends on one.

Noise. Three seeds of the reference recipe scored 35.9, 36.9, 36.9% on GSM8K in the first pass (standard deviation 0.57 points) and 36.8, 37.0 and 37.6% in the second. Scoring identical



weights twice through vLLM changed GSM8K accuracy by 0.2 to 2.9 points across seven models (median 0.4), from batch-order nondeterminism in greedy decoding. Batched and unbatched log-likelihood scoring of the same model agreed within 0.3 points. Differences of a point or two between single runs are within this noise.

4. Imitating Reference Solutions Made the Model Worse

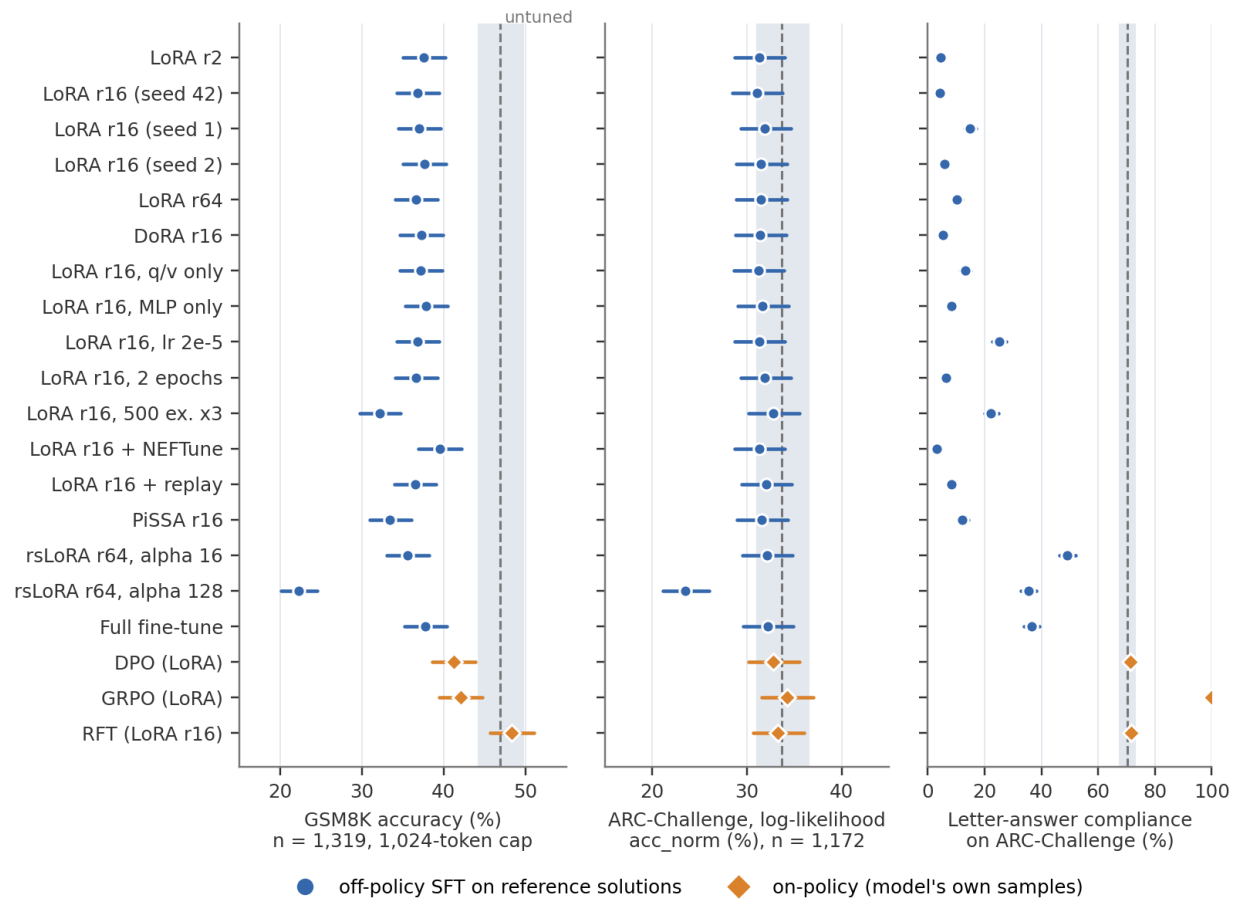


Figure 1. Every run on Qwen2.5-0.5B-Instruct, with Wilson 95% intervals; the dashed line and grey band mark the untuned model. Left: GSM8K accuracy at the 1,024-token cap. Centre: ARC-Challenge log-likelihood accuracy. Right: compliance with the instruction to answer ARC with a letter. Values are in Table 2.

Run	GSM8K	Change	p	ARC-C	ARC-E	Letter compliance
Untuned	46.9			33.7	60.2	70.2
LoRA r2	37.5	−9.4	<0.0001	31.3	55.9	4.4
LoRA r16, seed 42	36.8	−10.2	<0.0001	31.1	55.1	4.3
LoRA r16, seed 1	37.0	−9.9	<0.0001	31.9	55.9	14.8
LoRA r16, seed 2	37.6	−9.3	<0.0001	31.5	57.2	5.9
LoRA r64	36.6	−10.3	<0.0001	31.5	56.2	10.2
DoRA r16	37.2	−9.7	<0.0001	31.4	55.5	5.3
LoRA, q/v only	37.1	−9.8	<0.0001	31.2	55.2	13.1
LoRA, MLP only	37.8	−9.1	<0.0001	31.7	56.6	8.2



Run	GSM8K	Change	p	ARC-C	ARC-E	Letter compliance
LoRA, lr 2e-5	36.8	-10.2	<0.0001	31.3	56.6	25.1
LoRA, 2 epochs	36.6	-10.3	<0.0001	31.9	56.0	6.3
LoRA, 500 examples	32.1	-14.8	<0.0001	32.8	57.9	22.3
LoRA + NEFTune	39.5	-7.4	<0.0001	31.3	57.0	3.1
LoRA + replay	36.5	-10.5	<0.0001	32.0	56.7	8.2
PiSSA r16	33.4	-13.5	<0.0001	31.6	59.0	12.2
rsLoRA r64, alpha 16	35.6	-11.4	<0.0001	32.1	58.1	49.1
rsLoRA r64, alpha 128	22.2	-24.7	<0.0001	23.5	41.4	35.5
Full fine-tune	37.8	-9.2	<0.0001	32.2	57.6	36.5
DPO	41.2	-5.7	0.0002	32.8	60.6	71.2
GRPO	42.1	-4.9	0.0008	34.2	63.0	99.8
RFT	48.4	+1.4	0.263	33.3	61.0	71.6

Table 2. Runs on Qwen2.5-0.5B-Instruct. GSM8K accuracy at the 1,024-token cap; change and p (exact McNemar) against the untuned model. ARC-C and ARC-E are log-likelihood acc_norm. Letter compliance is the share of generated ARC-Challenge replies that open with an option letter. All in percent.

Every supervised run on the reference solutions scored below the untuned model, the closest of them by 7.4 points with $p = 1e-6$. The spread among the standard configurations is small next to that gap: rank 2 (1.1M trainable parameters) and rank 64 (35.2M) are 0.9 points apart, DoRA matches LoRA, and neither the target modules, the learning rate, a second epoch, replay nor full fine-tuning escapes the band of 36.5 to 37.8%. NEFTune scored highest of the supervised runs (39.5%), 2.4 points above the mean of the reference recipe's three seeds; on one seed that is suggestive at most.

The likely explanation is the targets. GSM8K's reference solutions are terse: the supervised models answer in about 109 tokens, where the untuned model reasons for about 297. Training on them replaced reasoning the model already had with a shorter style that it executes less reliably. The small-data run, 500 problems for three epochs, adopted the style as fully as the others and lost more accuracy, which fits the reading that the data mainly taught form.

5. Learning From Its Own Answers

RFT, trained on 2,293 of the model's own correct samples, was the only run that did not lose task accuracy: 48.4% against 46.9% for the untuned model (95% interval 45.7-51.1, $p = 0.26$). The difference is not significant. RFT kept the model's own answer length (304 tokens), its ARC log-likelihood accuracy and its letter compliance (71.6%). In short, it learned the requested answer format without unlearning anything we measured. Two confounds remain: it trained on fewer examples than the supervised runs, from a subset of the problems, and the model's sampling accuracy on training problems (51.1%) was well above its test accuracy, which may reflect exposure to GSM8K's training split during pretraining.

DPO on 324 pairs and a 200-step GRPO run sat in between. Both lost task accuracy (41.2% and 42.1%, $p = 0.0002$ and 0.0008), but less than any supervised run, and both kept ARC knowledge within a point of the untuned model's and letter compliance at or above it. Both were small, single runs; they say little about DPO or GRPO at scale. What the three on-policy runs share is that none of them caused the collateral drift that imitation did.



6. GRPO: Reward Hacking, Then Recovery

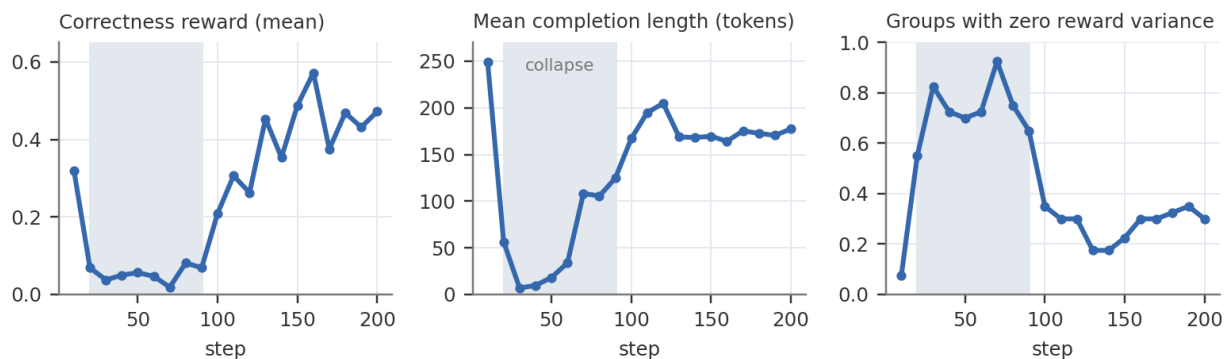


Figure 2. GRPO training on Qwen2.5-0.5B-Instruct, logged every 10 steps; each point averages 10 steps of 32 completions. The shaded span marks the collapse onto the format reward.

In the first logging window, GRPO's completions averaged 249 tokens and 32% were correct. By step 30 they averaged 7 tokens, typically a bare ##### N, and 4% were correct: the policy had found that the format bonus could be collected without the reasoning. In 82% of prompt groups every completion then earned the same reward, which in a group-relative method means no learning signal. The policy recovered without intervention: from step 70 completions lengthened, and by step 160 the correctness reward reached 57%. One run cannot isolate the cause of the collapse or of the recovery. A format bonus that pays independently of correctness invites this behaviour, and the fraction of zero-variance groups is an early warning worth logging.

The run's clearest effect was outside the task. Rewarded for following GSM8K's answer format, the model followed the ARC instruction to answer with a letter in 99.8% of questions, against 70.2% before training, and its ARC-Easy log-likelihood accuracy rose from 60.2% to 63.0% ($p = 1e-8$). We did not expect the second effect and have one run of it.

7. What Fine-Tuning Cost Elsewhere

Supervised fine-tuning cost measurable knowledge. Against the untuned model's 33.7% on ARC-Challenge and 60.2% on ARC-Easy, the standard LoRA runs lost about 2 points and 3 to 5 points respectively (for the reference recipe, -2.6 and -5.1 , $p = 0.001$ and $2e-15$). Full fine-tuning at $1e-5$ and the 500-example run lost least among them, under 3 points on ARC-Easy. None of the supervised runs approached the loss of the overscaled rsLoRA run (-10.2 and -18.8).

The larger cost was behavioural. The untuned model answered ARC with a letter, as instructed, in 70.2% of questions. After supervised fine-tuning it mostly answered in the prose style of GSM8K solutions, reasoning in full sentences before, or instead of, giving a letter: compliance fell to 3.1-25.1% for the LoRA runs and 36.5% for full fine-tuning. Replay of 3,000 general chat examples did not prevent it (8.2%). The on-policy runs kept or raised compliance (71.6% for RFT, 71.2% for DPO, 99.8% for GRPO). For a model that serves more than one task, a loss of instruction-following of this size matters more than a two-point loss on a knowledge benchmark, and only a generative measure shows it.

8. Other Models



Model and run	GSM8K	Change	p	ARC-C	Change	Letter compliance
Qwen2.5-1.5B-Instruct, untuned	68.2			46.8		98.3
1.5B, LoRA r16 (bf16)	61.0	-7.3	<0.0001	45.3	-1.5	64.5
1.5B, QLoRA r16 (4-bit)	60.1	-8.1	<0.0001	45.2	-1.6	48.2
Qwen2.5-0.5B pretrained, untuned	35.0			32.3		0.0
0.5B pretrained, LoRA r16	35.6	+0.5	0.764	31.4	-0.9	11.9
SmolLM2-1.7B, untuned	40.8			43.9		100.0
1.7B, LoRA r16	41.0	+0.2	0.918	42.2	-1.7	37.8
SmolLM2-360M, untuned	9.0			34.0		100.0
360M, LoRA r16	11.4	+2.4	0.023	33.7	-0.3	78.8
360M, full fine-tune	9.4	+0.4	0.753	34.0	-0.1	100.0
SmolLM2-135M, untuned	1.7			27.8		99.1
135M, LoRA r16	3.5	+1.8	0.003	26.8	-1.0	85.8

Table 3. Runs on other models, against each model's own untuned starting point. GSM8K is from the first pass, at the 400-token cap; ARC-C is log-likelihood acc_norm. All in percent.

On Qwen2.5-1.5B-Instruct the reference LoRA and QLoRA gave the same task accuracy (61.0% and 60.1%), both below the untuned model's 68.2%, and the same knowledge loss, which was not significant. These figures are from the 400-token pass, which truncated 179 of the untuned model's answers and at most 8 of the tuned models', so the true loss is if anything larger. The 4-bit run lost more letter compliance (48.2% against 64.5%). It peaked at 5.0 GB of GPU memory against 31.8 GB, but it also used gradient checkpointing and the bf16 run did not, so most of that difference is activation memory, not quantisation.

On the pretrained Qwen2.5-0.5B, the same supervised data neither helped nor hurt task accuracy (35.6% against 35.0%). It did install the answer format (97.4% of answers used it, against 43.3% before), and it left knowledge unchanged. The tuned model rarely stopped after its answer (1286 of 1,319 generations ran to the cap), consistent with a mismatch between the pretrained checkpoint's stop token and the chat template; its answer was read from the first #####.

The SmolLM2 models show the floor of the method. The 135M and 360M models could not do the task before or after training (under 12%), and SmolLM2-1.7B did not improve (41.0% against 40.8%) while its letter compliance fell from 100.0% to 37.8%. All four SmolLM2 runs gained ARC-Easy log-likelihood accuracy after training on mathematics, by 2.4 to 8.4 points; we have no explanation for this and note that per-byte normalised log-likelihood can shift with changes in the model's overall calibration.

9. Settings That Need Retuning

Two methods failed at the reference learning rate, and both failures concern the defaults, not the methods. Rank-stabilised LoRA scales the adapter by alpha divided by the square root of the rank instead of by alpha divided by the rank. With alpha 128 at rank 64 that is a factor of 16, eight times the reference recipe's factor of 2 at the same learning rate. That run fell to 22.2% on GSM8K and 23.5% on ARC-Challenge. With alpha 16 (a factor of 2) rsLoRA behaved like ordinary LoRA (35.6%). PiSSA initialises the adapter on the principal singular directions of each weight matrix, so training moves the directions the model relies on most. At 2e-4 it lost more task accuracy than LoRA (33.4%), though not more knowledge (31.6% on ARC-Challenge).



PiSSA is normally run at a lower learning rate, which we did not try.

The full fine-tune needs a similar caveat. Its weights and optimiser states were held in bf16 without a 32-bit master copy, so updates smaller than bf16's resolution at a given weight were lost. Its result is a lower bound on what full fine-tuning at this learning rate can do.

10. Measurement Lessons

Three first-pass measurements would have supported wrong conclusions. Each is easy to repeat.

The generation cap. At 400 tokens the untuned model's answers were truncated about 6 times as often as the supervised models'. Raising the cap to 1,024 moved the untuned model from 44.3% to 46.9%, and it changed two readings: RFT went from a significant gain (+2.9 points, $p = 0.02$) to no detectable difference, and GRPO from no detectable difference (-1.7 , $p = 0.25$) to a loss. Any comparison between models that answer at different lengths needs a cap well above the longer length.

The answer parser. Our first ARC parser took the first character of the reply, so it counted "A planet's rotation ..." as option A and "Chlorophyll ..." as option C. A more lenient parser, which searched the reply for a standalone letter, had the same flaw through the article "A": after supervised fine-tuning it chose A for 341 of the 1,172 questions, against 236 for the untuned model. On that parser PiSSA appeared to lose two thirds of its ARC accuracy (46.0% to 14.9%), a loss the log-likelihood measure does not show. We replaced both parsers: log-likelihood for knowledge and a strict pattern for compliance.

Evaluation nondeterminism. Greedy decoding through a batched inference server is not deterministic. The same weights scored up to 2.9 points apart on GSM8K in two passes. A single-pass difference of a point or two between runs carries no information.

11. What These Numbers Will Not Carry

These are small models on one task. GSM8K rewards a narrow skill, and the results say nothing direct about larger models, other tasks or longer training. Most runs used one seed; the seed spread in Section 3 applies to the reference recipe, not to every method. Every method used the reference recipe's hyperparameters, which is fair as a comparison of defaults and unfair to methods designed for other settings, PiSSA and rsLoRA in particular. The on-policy runs differ from the supervised ones in data size as well as in data source. The DPO and GRPO runs were short and single. The pretrained models may have seen GSM8K's training split. The runs used two GPU types and, for GRPO, a different library version, and every evaluation used greedy decoding.

What the evidence supports is narrower. On this task and at this scale, the source of the training data mattered more than any adapter setting we varied. Imitating external reference solutions cost task accuracy, about two points of measured knowledge and most of the model's compliance with an unrelated instruction, under every configuration we tried. Training on the model's own filtered samples cost none of these, and gained nothing we could detect on the task.

12. Methods and Artefacts

Methods: LoRA (Hu et al., 2021); DoRA (Liu et al., 2024); QLoRA (Dettmers et al., 2023); rsLoRA (Kalajdziewski, 2023); PiSSA (Meng et al., 2024); NEFTune (Jain et al., 2023); DPO (Rafailov et al., 2023); GRPO (Shao et al., 2024); rejection-sampling fine-tuning (Yuan et al., 2023), related



to STaR (Zelikman et al., 2022). Data and models: GSM8K (Cobbe et al., 2021); ARC (Clark et al., 2018); Qwen2.5 (Qwen Team, 2024); SmoLM2 and SmoLTalk (Allal et al., 2025). Log-likelihood scoring follows the EleutherAI Language Model Evaluation Harness. Software: PyTorch, Hugging Face Transformers, PEFT and TRL, and vLLM.

Every adapter and checkpoint, training log, generated answer and per-item score is retained, with the analysis code that produced every number in this paper.

PureTensor operates a sovereign AI infrastructure fleet (on-premises GPU compute, storage and serving) run day-to-day by autonomous agents under human direction. This paper is PT-R-2026-039. Training and the first evaluation pass ran on 24 October 2026; the log-likelihood and 1,024-token evaluations ran on 25 October 2026.