Building a Miniature AI Factory:
Post-Training, Inference Engines, and Provider Systems

Tanvir Ahmed
https://github.com/mailtotanvir/Nano-RL-Inference

September 2026

Abstract

Modern AI systems are often described as a model plus an API. That shorthand hides the engineering that determines whether a training result can become a reliable product: frozen data boundaries, reward contracts, reference policies, rollout collection, inference engines, model routing, authentication, rate limits, usage accounting, and artifact custody. This paper documents a constrained, end-to-end replication of that chain using Qwen/Qwen2.5-0.5B-Instruct, a deterministic arithmetic environment, LoRA supervised fine-tuning (SFT), and four reinforcement-learning families: REINFORCE, PPO, RLOO, and GRPO. The study selected a 0.5B model and a single NVIDIA L4 specifically to make the full loop affordable, inspectable, and disposable rather than to claim frontier capability. Training used 512 addition/subtraction examples; evaluation withheld an entire division-template family of 128 examples. A three-seed, 21-cell campaign found the same held-out result, 4/128, for every saved adapter. This is not evidence that the methods are equivalent. It is evidence that reward-bearing optimization did not transfer under this strict contract. We also run the merged adapter through vLLM and SGLang, and expose local and live backends through an OpenAI-compatible FastAPI provider with routing, authentication, rate limits, streaming, metrics, and usage accounting. The contribution is an explainer backed by executable artifacts: what each RL family is buying, where serving changes the problem, why a favorable internal reward is not a generalization claim, and how to retain evidence after a short-lived GPU has been deleted.

1 Introduction

Modern AI systems are often described as a model plus an API. That shorthand is useful until it hides the engineering that determines whether a training result can become a reliable product: frozen data boundaries, reward contracts, reference policies, rollout collection, inference engines, model routing, authentication, rate limits, usage accounting, and artifact custody.

This paper documents a constrained, end-to-end reconstruction of that chain. It is not a claim that a 0.5B model is competitive with frontier reasoning systems, nor a claim that five RL families are interchangeable. It is a practical explainer: we build small but real versions of the post-training and serving components, then retain enough evidence to distinguish an internal optimization signal from transferable behavior and a functioning engine from a usable provider.

The organizing question is therefore practical: *what does it take to build, inspect, and operate a miniature version of the post-training and inference stack used by modern AI products?* The remainder follows the construction order. Section 2 establishes the experimental and architectural choices. Section 3 explains why evidence custody is part of the system. The subsequent sections unpack the learning loops, results, serving tradeoffs, provider ecosystem, and lessons.

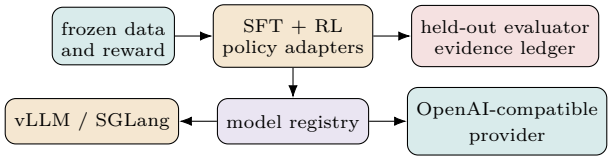

Table 1: Method selection is a tradeoff among feedback shape, variance, stability, and operational cost. These are the implemented miniature forms, not a claim of exhaustive or production-scale implementations.

Method	What it changes	Why build it here	Main tradeoff to watch
SFT	Fits verified answer demonstrations	Establishes a common policy and formatting prior	Cannot improve beyond the coverage of demonstrations
REINFORCE	Raises probability of rewarded sampled actions	Simplest direct reward-to-gradient baseline	Sparse reward makes the estimate high variance
PPO	Clips policy-ratio movement and adds KL pressure	Shows how a useful policy can be protected from destructive updates	Requires old-policy bookkeeping, advantages, and a reference policy
RLOO	Uses sibling rollouts as a leave-one-out baseline	Reduces variance without fitting a separate learned critic	Needs grouped samples and careful episode construction
GRPO	Normalizes reward within a response group	Replaces a large critic with relative group evidence for verifiable tasks	Rollout count, group composition, KL, and generation cost become first-order settings

2 Experiment design and architectural choices

2.1 One small model, one L4, and LoRA adapters

The base is `Qwen/Qwen2.5-0.5B-Instruct`. A 0.5B instruct model is large enough to exercise realistic tokenization, chat templates, adapter loading, generation, reference-policy comparisons, and server integration, while remaining feasible on one L4. The purpose is iteration economics: if the entire train, evaluate, serve, collect, checksum, and teardown loop cannot be completed under a small budget, it is difficult to inspect its failure modes honestly.

SFT uses PEFT LoRA with rank 8, alpha 16, 0.05 dropout, and targets `q_proj` and `v_proj`. One epoch, per-device batch size four, gradient accumulation four, and learning rate 2×10^{-4} form the cold-start policy. These are not claimed as optimal settings. They were chosen to create a stable, reproducible adapter that each RL method could inherit, rather than letting every method begin by rediscovering answer format.

2.2 A task where correctness is cheap, but transfer is not

The environment has 512 generated train examples from addition and subtraction templates. The held-out set contains 128 division-template examples. The reward uses conservative normalization and exact numeric matching. That makes the reward auditable: a response is either the expected integer or reduced fraction, or it is not. It also creates a harsh distinction between fitting a prompt family and transferring across one.

This choice is important. A high-quality learned reward model could obscure whether a policy was rewarded because of its behavior or because of a judge's preference. Here, the reward is not the hard part. The hard part is whether a policy optimized on one family carries useful structure into another family.

3 Evidence discipline and campaign custody

The project treats evidence as a system boundary, not a publication task postponed until the end. Every training cell receives an explicit seed, output directory, SFT-adapter dependency, and metric path. A successful adapter is evaluated immediately after training, so the record binds the training invocation to the model artifact and its held-out score rather than leaving evaluation as a later manual step.

The three seeds, 13, 42, and 97, crossed seven method labels, producing 21 lifecycle cells and 21 saved-adapter evaluation files. The campaign ledger records planned, completed, and failed transitions; the final metric-complete campaign recorded 21 completed cells. This discipline also preserves negative evidence. A result that does not improve is not discarded because it does not make a favorable chart.

GPU work used a short-lived on-demand `g2-standard-8` with one NVIDIA L4 in GCP. The rate captured before campaign work was $0.853624312 per hour. Artifacts were copied to the laptop and an existing OCI mirror, and their SHA-256 values matched before the VM was deleted. The metric-complete archive hash is maintained in the repository evidence ledger. No idle GPU VM was retained.

The split itself is another evidence control. Training uses 512 generated addition and subtraction examples, while held-out scoring uses 128 division-template examples. Exact-match normalization is conservative. This does not make the evaluation universal, but it prevents a training-family prompt form from silently becoming the definition of success.

4 Five learning loops, not one algorithm

4.1 SFT: the cold start that gives RL a language

SFT is the control condition and the practical starting point. It teaches the model the chat-format contract and the terse-answer behavior expected by the evaluator. In a product setting, SFT is often the cheapest route when trusted traces already exist. Its limitation is equally important: it can fit the demonstrated surface without acquiring the capability that a withheld task family demands.

4.2 REINFORCE: the direct but noisy path

For REINFORCE, a completion is sampled at temperature 0.8 with a 16-token cap. Exact-match reward multiplies the completion log probability:

$$\mathcal{L}_{\text{REINFORCE}} = -r(y)\frac{1}{|y|}\sum_{t}\log \pi_\theta(y_t \mid x, y_{<t}).$$

The attraction is transparency. A reward-bearing answer directly receives more probability mass. The cost is variance: if correct answers are rare, most samples contribute no learning signal. This implementation is therefore an educational baseline for inspecting the reward-to-policy connection, not a claim that naïve REINFORCE is the preferred production post-training method.

4.3 PPO: conservative movement has its own machinery

PPO begins at the SFT adapter and keeps a frozen copy as a reference. It computes a sampled-completion probability ratio, clips it to $[0.8, 1.2]$, and adds KL pressure with coefficient 0.05:

$$\begin{aligned}\mathcal{L}_{\text{PPO}} = & -\min\left(r_t A_t, \text{clip}(r_t, 0.8, 1.2)A_t\right) \\ & + 0.05\,\text{KL}(\pi_\theta \| \pi_{\text{ref}}).\end{aligned}$$

The practical reason to build it is not the equation itself. It is to learn what changes when a policy is good enough to damage. PPO buys a trust region-like update boundary, but demands old-policy state, reference behavior, advantages, and careful rollout accounting. The small implementation uses reward as the advantage signal; it intentionally does not claim the full robustness of a production PPO stack.

4.4 RLOO: a critic replacement with a sampling contract

RLOO uses four responses per prompt. Each response is judged against the other three, so its advantage is relative to sibling outcomes rather than a learned value function. The pinned TRL 0.15 configuration uses 128 total episodes, four samples per prompt, response length 16, one PPO epoch, learning rate 10^{-5}, and KL coefficient 0.05. This is useful where adding a critic is undesirable, but it makes batch grouping part of algorithm correctness. A wrong global batch or missing sibling sample is not a harmless implementation detail.

4.5 GRPO: the group becomes the baseline

GRPO also uses four generations per prompt, but normalizes reward inside the group. The implementation uses one epoch, batch size four, gradient accumulation four, 16-token completions, and learning rate 10^{-5}. The appeal is clear for verifiable tasks: a group can provide relative evidence without a large learned critic. The operational lesson is equally clear: generation count, response length, reference behavior, batch divisibility, and KL drift are part of the actual algorithm budget, not afterthoughts.

Table 2: Held-out division-template exact matches. Every cell is 4/128; mean accuracy is 0.03125.

Method	S13	S42	S97	Mean
SFT	4/128	4/128	4/128	.03125
REINFORCE	4/128	4/128	4/128	.03125
PPO	4/128	4/128	4/128	.03125
RLOO	4/128	4/128	4/128	.03125
GRPO	4/128	4/128	4/128	.03125
GRPO k1 label	4/128	4/128	4/128	.03125
GRPO k3 label	4/128	4/128	4/128	.03125

Table 3: Inference tradeoffs on the same merged adapter and one L4. SGLang led several single-request and medium cells; vLLM retained a higher short concurrency-four aggregate token rate. Output lengths and scheduler behavior differ, so no universal engine ranking follows.

Workload	vLLM p50 ms	SGLang p50 ms	Faster p50	vLLM tok/s	SGLang tok/s	Throughput reading
Short, single	136.48	104.27	SGLang	64.90	152.04	SGLang in this cell
Short, concurrency 4	171.75	163.92	SGLang	53.59	37.63	vLLM aggregate tok/s
Medium, single	941.76	338.78	SGLang	67.70	188.16	SGLang in this cell
Medium, concurrency 4	1047.66	405.20	SGLang	61.16	160.77	SGLang in this cell
Long, single	430.87	656.49	vLLM	67.26	194.84	output length matters

4.6 KL estimators: a configuration label is not an ablation

On 32 development prompts, the SFT policy was compared to the frozen base at the next-token distribution. Mean k1 was 0.06868447 and mean k3 was 0.06909129. These estimates are close here, but k1 is a forward-KL expectation while k3 is a non-negative control-variate form. The campaign includes GRPO k1 and k3 labels for operationally matched execution. The current trainer does not alter its loss formula by that label, so they cannot support a causal estimator claim. This is a general lesson: naming a setting is not the same as measuring its intervention.

5 Results: why reward did not become transfer

The campaign completed all 21 cells. Every saved adapter scored 4/128 on every seed in the template-disjoint evaluation (Table 2). One GRPO cell reported mean rollout reward 0.79167 and KL 0.37643, then scored 4/128 held out. This does not mean reward was wrong. It means the reward answered a narrower question: whether a sampled response matched training-family answers. The held-out set asked whether that learning crossed a template boundary.

The most defensible interpretation is narrow. With this small model, data family, reward, output budget, and training exposure, the listed update rules did not change template-disjoint behavior. It does *not* establish that the methods are generally equivalent, that arithmetic reasoning is impossible for the model, or that more data, a different curriculum, a process reward, or a different evaluation would fail. The result is useful precisely because it prevents an attractive but unsupported story: a rising rollout reward cannot be substituted for generalization.

6 From adapter to inference service

A model artifact is not an inference product. The SFT adapter was merged for engine compatibility, then served through vLLM 0.8.3 and SGLang 0.4.4.post1 on the same L4. Both engines were exercised through short, medium, and long request shapes with single-request and concurrency-four cells. Ten requests were measured per cell after warmup. This is a workload probe, not a universal benchmark.

Figure 1 and the table answer a more useful question than "which engine wins?" A serving choice depends on request length, concurrency, queueing, stopping behavior, cache policy, and whether the business requirement values first-token latency, total latency, or aggregate throughput. The long cell makes the point: lower latency and higher reported token throughput can point in different directions when completion behavior differs. A benchmark

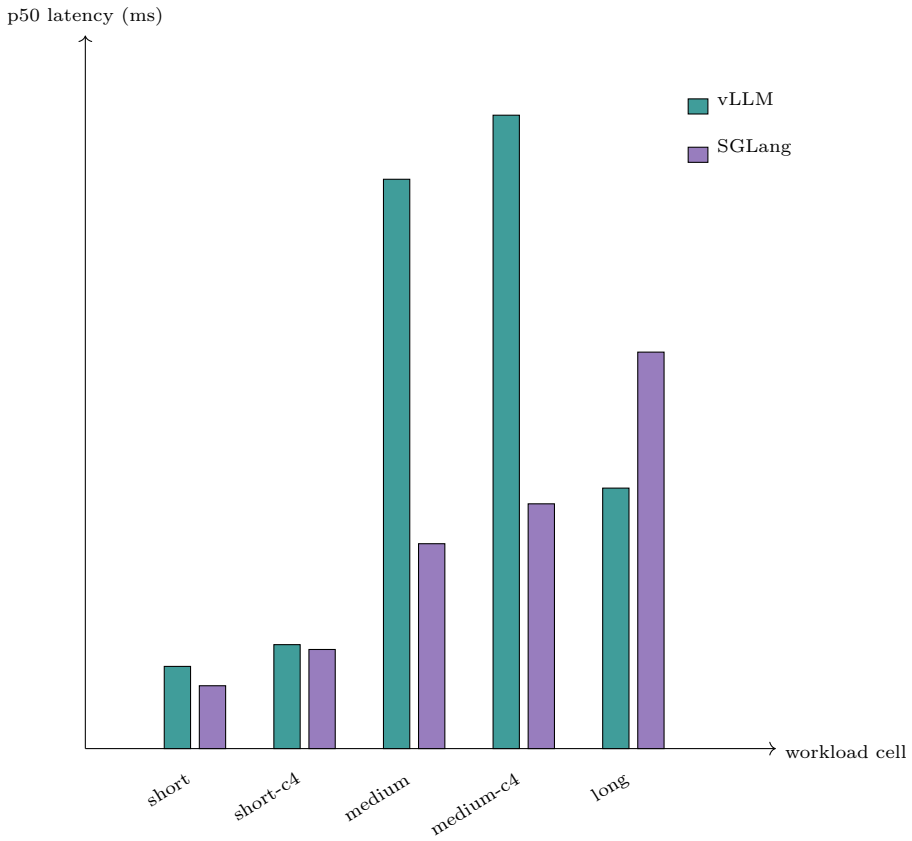

Figure 1: Measured p50 latency across five workload cells. Lower bars are faster; this view intentionally does not collapse latency and output throughput into one winner.

is a protocol, not a decorative number.

7 The provider boundary is where engines become an ecosystem

The FastAPI provider exposes `/health`, `/v1/models`, `/v1/completions`, `/v1/chat/completions`, `/v1/responses`, `/usage`, and `/metrics`. It validates bearer tokens by hash, resolves a public model identifier through deterministic routing, uses a per-identity token bucket, supports streaming completion events, accounts for requests and token estimates, and exports request metrics.

The design deliberately keeps vLLM, SGLang, and generic remote OpenAI-compatible inference behind a common backend protocol. A client should depend on a stable OpenAI-shaped contract, not on a local server's port, a vendor's model naming convention, or an engine-specific streaming behavior. A live remote-backend acceptance artifact returned `42`, counted 10 prompt and 1 completion token, and recorded 1578.84 ms latency. This is evidence of a real provider path, not a claim of a public, production-hardened deployment.

8 Lessons and limitations

For ML engineers, the transferable pattern is to choose the learning loop from the feedback available. Verified answers invite SFT and group-relative rollouts; expensive or ambiguous rewards demand stronger variance and evaluation controls. For platform teams, the transferable pattern is to place model identity, authentication, routing, quotas, usage, and observability outside the serving engine. For leaders, the cost lesson is that an L4 bill is only affordable learning when the campaign starts ready, produces evidence, and is torn down without losing the result.

The limitations are material. This is one model family, small deterministic data, a strict family-disjoint test,

compact educational REINFORCE and PPO implementations, brief engine samples, and no integrated public GCP provider deployment. GRPO k1 and k3 labels are not a loss-formula ablation. This work should be read as a reproducible engineering explainer and evidence record, not a certification, a broad algorithmic comparison, or a leaderboard claim.

9 Conclusion

The central result is not that five RL paths were tried or that two engines returned latency numbers. It is that a miniature AI factory made their dependencies inspectable. Training can improve a local reward while failing to transfer. A model can serve through two engines without either engine being universally superior. An API can look compatible while still needing governance around identity, cost, and evidence. Building the complete chain made those distinctions concrete. The model is one component; the accountable system around it is the artifact.

Reproducibility. Code, frozen data manifest, campaign ledger, per-seed metrics, benchmark JSON, and claim-evidence matrix are in https://github.com/mailtotanvir/Nano-RL-Inference. The selected adapter is at https://huggingface.co/mailtotanvir/nano-rl-inference-arithmetic-sft-adapter. Cite the Zenodo concept DOI https://doi.org/10.5281/zenodo.22886076; version 0.1.1 is pinned by https://doi.org/10.5281/zenodo.22886166.