Zero-Knowledge Learning

David Koplow[1] and Sam Layton[2]

[1]MIT
[2]Stanford
dkoplow@mit.edu

Abstract

High computational costs and strict privacy constraints currently bottleneck the post-training of Large Language Models. We introduce **Zero-Knowledge Learning (ZKL)**, an online protocol based on Evolution Strategies (ES) that enables continuous learning from private inference streams. Each inference request contributes at most a single bounded scalar reward associated with a random perturbation seed. Prompts, completions, and token-level traces are never transmitted. In the absence of auxiliary information, semantic reconstruction of individual prompts from these scalar observations is information-theoretically infeasible. Residual privacy risks arise only through aggregate influence on the model parameters, which is inherently sourceless and dominated by population-level effects rather than individual records. We also optimize the protocol to be implemented with virtually no speed degradation. Experiments demonstrate that ZKL can outperform GRPO by up to 15× in sample efficiency across some reasoning and alignment tasks.

Keywords post-training, alignment, optimization, language models, safety

1 Introduction

Post-training has recently emerged as a central focus in the development of Large Language Models (LLMs). These techniques have become indispensable, unlocking advanced reasoning capabilities through Reinforcement Learning with Verifiable Rewards (RLVR) [13] while simultaneously serving as the primary mechanism for safety and alignment [21]. Driven by the demand for enhanced capability and robust alignment, the computational budget allocated to post-training has surged; once a marginal expense relative to pre-training, post-training costs are projected to rival or even surpass pre-training expenditures [21]. Conspicuously, this rapid advancement has predominantly relied on gradient-based RL paradigms through dominant methods such as Proximal Policy Optimization (PPO), Direct Preference Optimization (DPO), or Group Relative Policy Optimization (GRPO) [32, 27, 33]. While these methods have proven effective, this work challenges the ubiquity of this paradigm by demonstrating the emerging efficacy of zero-order methods.

Despite their dominance, gradient-based RL methods are beset by multiple structural liabilities. These algorithms are notoriously sample inefficient [39], requiring prohibitive amounts of interaction data to achieve convergence. This inefficiency is compounded by high gradient variance [30], which introduces significant stochasticity and destabilizes the optimization landscape. Moreover, precise temporal credit assignment remains an unsolved challenge, often necessitating complex and biased proxy rewards [46]. Beyond these stability issues, RL performance is exceedingly brittle, exhibiting

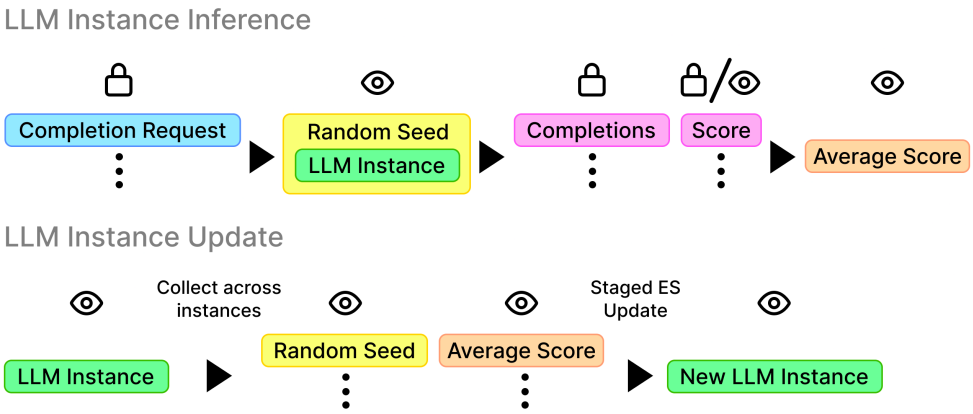

Figure 1: This figure illustrates the Zero-Knowledge Learning protocol and its separation between private inference and parameter updates.

extreme sensitivity to hyperparameters and the specific cognitive priors of the initial model [6]. Finally, the paradigm is threatened by a tightening data bottleneck: high-quality, task-specific training data is difficult to procure [40], a crisis exacerbated by increasingly stringent privacy regulations that restrict the availability of public data for training [20].

In light of these structural limitations, Zero-Order (ZO) optimization has staged a formidable resurgence as a viable alternative to gradient-based learning. Malladi et al. [15] first demonstrated the efficacy of ZO methods for fine-tuning LLMs on downstream tasks, a finding subsequently expanded by a suite of advanced techniques including ZO-Bench [44], and DPZero [45] which established ZO's utility for both classification and privacy-preserving adaptation. However, these earlier iterations largely failed to compete directly with state-of-the-art Reinforcement Learning (RL) baselines until the recent breakthrough by Qiu et al. [24]. Their work empirically demonstrated that ZO optimization could consistently outperform GRPO on the Countdown reasoning task, achieving superior sample efficiency and a more favorable KL-divergence tradeoff [22]. Furthermore, the applicability of this paradigm was significantly broadened by Sarkar et al. [31], who established that ZO methods can be effectively applied to Low-Rank Adaptation (LoRA) updates.

Although these early results are encouraging, Zero-Order evolutionary approaches have not yet seen broad adoption over well-established gradient-based techniques. To address this shortfall, we present **Zero-Knowledge Learning (ZKL)** as a competitive framework for post-training contemporary LLMs. Our experiments show that ZKL offers superior sample efficiency and speed, exhibits robustness across a wide range of model scales and task types, and maintains structural privacy, thereby enabling the use of previously inaccessible datasets for post-training. [20].

Our precise contributions are summarized as follows:

- **Efficient Implementation:** We derive an optimal minibatching strategy for Online ES, achieving up to 3× greater sample efficiency than Qiu et al. [24] and 15× over GRPO, provided alongside a highly efficient open-source implementation.
- **The ZKL Protocol:** We formalize a privacy-preserving training framework that communicates only perturbation seeds and scalars, enabling the use of private, online inference as training data.

- **Experimental Validation:** We show the efficacy of ZKL as a post-training procedure for multiple model sizes and both reasoning and RLHF tasks.
- **Inference-Time Training:** We demonstrate the practicality of ZKL for inference time optimization using multi-objective loss functions.
- **Theory:** We prove that ZKL functions as an unbiased estimator of the optimal update direction, ensuring consistent convergence properties, and discuss theoretical privacy analysis.

2 Background and Related Work

Although the Zero-Knowledge Learning (ZKL) protocol offers a generalized framework for derivative-free optimization, we position this work squarely within the domain of post-training pre-trained Transformers. This regime is currently dominated by gradient-based paradigms, ranging from on-policy reinforcement learning methods like PPO [32] and GRPO [33] to preference-based objectives such as DPO [27] and IPO [8]. While significant research has focused on mitigating the sample inefficiency of these methods [12, 28], ZKL diverges from this trajectory by abandoning the gradient requirement entirely.

In the following sections, we review successful applications of zero-order (ZO) optimization, tracing the evolution from finite-difference methods to recent hyperscale Evolution Strategies (ES), and identifying the specific privacy and scalability gaps ZKL is designed to close.

2.1 Zero-Order Optimization for LLM Fine-Tuning

Zero-Order (ZO) optimization refers to derivative-free algorithms that optimize an objective relying solely on function evaluations [36, 19]. In deep learning, this is predominantly realized through gradient approximation via finite differences, a technique rooted in Simultaneous Perturbation Stochastic Approximation (SPSA) [35]. Malladi et al. [15] successfully adapted this paradigm for language models (MeZO), using random weight perturbations to reduce memory overhead, with subsequent work improving convergence via momentum and differential privacy [45]. However, while ZO methods can match first-order performance on classification tasks [44], finite-difference approximations have historically failed to compete with gradient-based RL on complex reasoning and generation tasks [24].

2.2 Evolution Strategies for Post-Training

Distinct from finite-difference approximations, Evolution Strategies (ES) optimize a policy directly in parameter space without explicitly approximating a gradient [29]. Canonical implementations [30, 41] operate by instantiating a population of K agents perturbed by Gaussian noise to compute a weighted update direction.

While long considered intractable for LLMs due to the "curse of dimensionality" [11], Qiu et al. [24] recently challenged this assumption, demonstrating that ES can outperform RL baselines on reasoning tasks using extremely small populations ($K \approx 32$). However, in order to reduce variance, Qiu averaged fitness over the *entire* prompt dataset, multiplying effective rollouts by the size of their dataset.

To address this, Sarkar et al. [31] proposed applying ES to Low-Rank Adaptation (LoRA) matrices (EGGROLL). While demonstrating impressive results, their work assumes that ES for LLMs is most effective with massive populations ($K > 1000$). However, our results would argue that this assumption is flawed. Furthermore, if K is kept small (as validated by Qiu et al.), traditional full-parameter ES is no longer memory-constrained and can run fully instantiated models in parallel at peak performance without the need for LoRA.

2.3 Privacy, Federated Learning, and Zero-Knowledge Proofs

Parallel to these algorithmic developments, we must distinguish *Zero-Knowledge Learning* (ZKL) from the cryptographic primitive of Zero-Knowledge Proofs (ZKPs) [9]. While ZKPs rely on cryptographic hardness, ZKL is a training protocol designed to optimize agents with "zero knowledge" of intermediate tokens. This ensures sensitive content remain siloed on the deployment side, aligning closely with communication-efficient Federated Learning (FL) [43, 23]. Unlike traditional FL which aggregates gradient vectors [42], ZKL operates in the regime of only integer perturbation seeds and scalar rewards.

We note that ZKL provides *structural* privacy (hiding the training process), not inherent *mathematical* differential privacy regarding the output model [37]. While techniques like DP-Zero [45] could theoretically be integrated, ZKL focuses on enabling training on private inference streams.

2.4 Position of ZKL

Our work positions ZKL at the intersection of these paradigms, synthesizing the parameter efficiency of small-population ES with the data scalability required for industrial application. We adopt the low-population hypothesis ($K \ll d$) of Qiu et al. [24] to eliminate memory overhead, but reject their prohibitively expensive full-dataset update rule. Instead, we introduce a stochastic minibatching formulation re-engineered for asynchronous, online inference, allowing ZKL to operate on real-time deployment streams rather than static datasets. By combining these refinements, we achieve a up to 3× improvement in sample efficiency over Qiu et al.'s baselines.

3 Zero-Knowledge Learning

We introduce Zero-Knowledge Learning as an efficient algorithmic variant of standard ES methods, and enables a privacy preserving protocol during training.

3.1 Weight Updates

ZKL builds on traditional Evolution Strategies (ES). As in ES and other descent algorithms, it iteratively updates its parameters in some direction u with a learning rate α. Traditional ES algorithms estimate this update step u by forming a weighted sum of Gaussian noise $\delta_j \sim \mathcal{N}(0, \sigma I)$, over the parameter space $\delta_j \in \mathbb{R}^{|\theta|}$. These small noise perturbations δ_j paired with their given weights w_j are then summed to get the update step $u = \sum w_j \delta_j$.

When using ES to post-train LLMs, models are updated for T iterations, with each step sampling a fixed K perturbations δ_j. For each mutated model with perturbed parameters $\tilde{\theta}_{tj} = \theta_t + \delta_j$, ES then generates weights w_j by doing a rollout over the entire prompt dataset, and then averaging the corresponding rewards rij from every rollout, producing average reward scores $\bar{r}_j$ for each model

pertrubation. With K rewards for each model perturbation, ES then generates weights w_j by normalizing $\bar{r}_j$.

In ZKL (Algorithm 1), we assign each seed a unique minibatch of the data instead of reusing the same prompt over multiple model perturbations. This introduces a new hyperparameter B for the minibatch size. Thus, each step yields $B \times K$ unique prompts, agent responses, and individual rewards. We present a proof in Appendix **??** that this modification still serves as an unbiased gradient estimator.

This variant is **more powerful than standard ES since it mirrors inference time dynamics**; every prompt is only accessed once, and the procedure can scale to online distributed training on arbitrarily large datasets. However, we still need to contend with the performance constraints that are caused by hosting and running inference on many perturbations of a model. For ES to work at all, every model performing inference must be different by exactly the noise constructed by its respective seed. Because floating-point arithmetic is not reversible, restoring weights after perturbation usually requires storing a full copy to avoid cumulative round-off. Storing this clone in GPU or CPU RAM or disk is prohibitively slow, inefficient, and expensive for modern LLMs, which can have over a trillion parameters.

However, this standard limitation can be overcome through an implementation trick; we can **operate in higher precision when doing parameter transformations** since `fp16` and `bf16` additions or subtractions can be recovered exactly when carried out in `fp32`. We generate noise and apply each perturbation in high precision, then downcast for execution whenever the seed changes. This lets us add or remove random perturbations without extra GPU memory or compute during inference, because the model parameters themselves are updated in place. Changing the seed does introduce an $O(\#\text{params})$ cost, but memory overhead can be kept negligible by offsetting updates and tiling parameter operations.

ZKL runs vLLM for inference and Ray for data-parallel and distributed scaling [10, 18]. We introduce a custom vLLM handler that wraps the base LLM worker and controls both random parameter perturbations and generation. The implementation is fully compatible with PyTorch's automatic differentiation system, allowing experimentation with hybrid integrations that combine post-training and optimization techniques.

Our code is available on GitHub: [link redacted]

3.2 The Protocol

This section introduces the ZKL protocol (Figure 1) and explains its appeal. At inference time, each request is served by a model instance defined by a perturbation seed, which produces outputs scored locally by a verifier or reward function. Only scalar rewards, aggregated across examples and paired with their seeds, are returned for training. Each prompt participates in training through at most one bounded scalar reward, observed without access to prompt content, completion text, or persistent user identity. The model provider regenerates perturbations from the seeds, aggregates rewards across instances, and applies an Evolution Strategies update to produce a new checkpoint. Prompts, contexts, and token-level outputs never leave deployment, so post-training proceeds without exposure to private inference data. This design enables capabilities unavailable to standard post-training methods.

Better Data. Frontier labs develop models and earn revenue from inference, but realistic applications are expensive to emulate and too risky to train on directly. Live production traffic includes

Algorithm 1 Training Procedure for ZKL

Require: Steps T, batch size B, pop K, noise σ, learning rate α
Ensure: Trained parameters θ_T
  Initialize parameters θ_0, dataset $\mathcal{D}$, reward function $R(x, y)$
  for $t = 1$ **to** T **do** ▷ Iterate over T batches
    for $j = 1$ **to** K **do** ▷ Create K perturbed instances
      Sample perturbation $\delta_j \sim \mathcal{N}(0, \sigma^2 I)$
      $\bar{r}_j \leftarrow 0$
      for $i = 1$ **to** B **do** ▷ Rollouts for perturbation j
        Sample $x_{ij} \sim \mathcal{D}$ (without replacement)
        $y_{ij} \leftarrow \pi(x_{ij} \mid \theta_t + \delta_j)$
        $\bar{r}_j \leftarrow \bar{r}_j + \frac{1}{B} R(x_{ij}, y_{ij})$
      end for
    end for
    $\bar{r} \leftarrow \frac{1}{K} \sum_{j=1}^{K} \bar{r}_j$ ▷ Population reward mean
    $s \leftarrow \mathrm{StdDev}(\{\bar{r}_j\}_{j=1}^{K})$ ▷ Population reward std
    $w_j \leftarrow (\bar{r}_j - \bar{r})/(s + \epsilon)$ ▷ ϵ stabilizes normalization
    $\theta_{t+1} \leftarrow \theta_t + \alpha \sum_{j=1}^{K} w_j \delta_j$ ▷ ES update
  end for
  return θ_T

private user prompts, proprietary corporate information, and sensitive government tasks, which makes using it for training both legally and ethically fraught. In contrast to RL approaches like GRPO, ZKL relies on a single scalar reward per seed and never inspects token-level data. Given verifiable rewards or simple behavioral indicators, it can learn without seeing either the inputs or the outputs. Since the provider only receives an average reward and a random seed, reconstructing the semantic content of individual prompts is information-theoretically underdetermined. Rewards can be derived from LLM-based evaluators or from proxy signals that do not depend on raw content, such as whether a user switches to or retries with another model. The distributed architecture supports incremental updates while keeping the service running smoothly.

Inference-Time Learning. ZKL also aligns with the economic realities of LLM development. Instead of depending on an expensive, distinct training stage, it transforms paid inference into the driving force behind post-training. Because prompts are never reused, ongoing inference continuously produces new training signals.

3.3 Experiment and Results

We present a series of experiments fine-tuning Qwen2.5 and 3 on the Countdown and HH datasets to evaluate this new learning paradigm and show that ZKL is highly sample efficient, fast, and exhibits convergence behavior consistent with established post-training methods [26, 25, 22, 2]. Our results reveal three key findings that position ZKL as a strong candidate for post-training: (i) We identify a favorable K–B tradeoff that yields substantial gains in sample efficiency which, when combined with speed improvements, makes ZKL exceptionally efficient in practice; (ii) ZKL proves robust across a wide range of model sizes; (iii) ZKL operates effectively in a fully distributed online learning regime with no prompt repetition, matching the standard online setting and enabling cross-training.

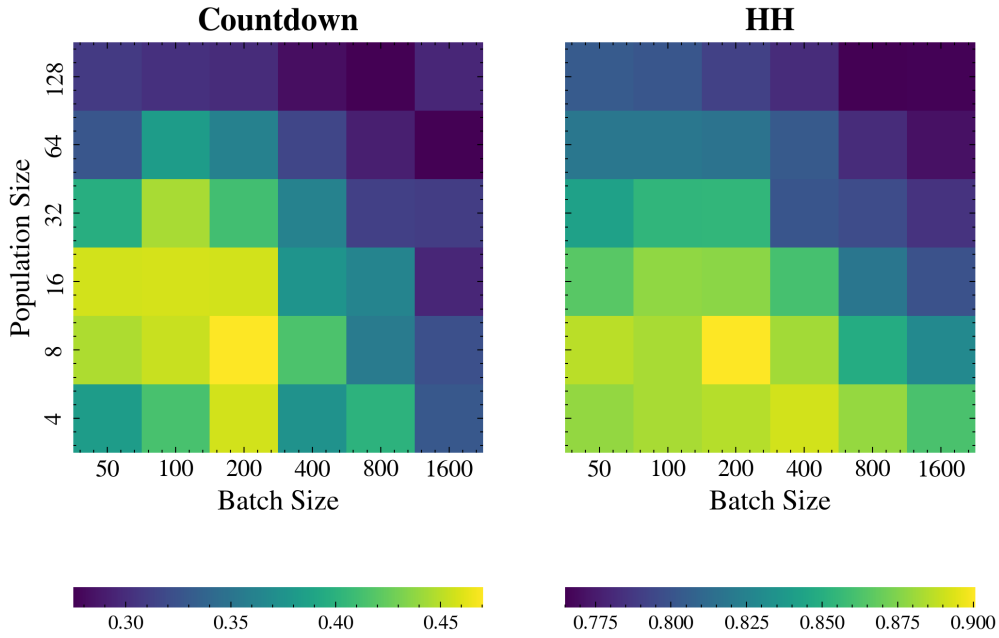

Figure 2: Compute-optimal tradeoff between ES population size K and per-perturbation batch size B under a fixed two-hour budget on four A100 GPUs for Qwen2.5-7B-Instruct trained on Countdown and HH datasets. Color indicates achieved validation performance. While the exact loss landscape and optimal hyperparameters do change with problem, learning rate, and model size, small populations and batch sizes are generally best under constrained compute. Increasing population without increasing batch size degrades performance due to higher reward noise. We found $K = 8$ and $B \in 100, 200$ to generally perform best across different problems and model sizes. A batch size of 100 is often preferable to a batch size of 200 in our setup, even if there is a small performance penalty because it requires less data and converges twice as fast.

3.4 Population and Sample Efficiency

3.4.1 Population and sample size tradeoffs

A core advantage of ES-based post-training is that it avoids token-level credit assignment and can produce stable updates from only a few rollouts. ZKL improves on prior ES formulations by minibatching: each perturbation is evaluated on a distinct batch of prompts, and the update uses only the aggregated rewards. This reduces redundant computation relative to full-dataset evaluation, and it matches the deployment regime where prompts are effectively non-repeating.

Under a fixed compute budget, the dominant design choice is how to allocate rollouts between the mutation population K and the per-perturbation batch size B. Larger K explores more directions in parameter space but increases the variance of the normalized weights unless B is also increased; larger B reduces reward noise per direction but reduces the number of directions explored. Figure 2 shows a clear compute-optimal tradeoff between K and B on Countdown: performance is best in the small-population regime, and high-quality updates are achievable with surprisingly small K when B is chosen to control reward variance.

Across most experiments, $K = 8$ and $B = 100$ is a strong default. This configuration yields

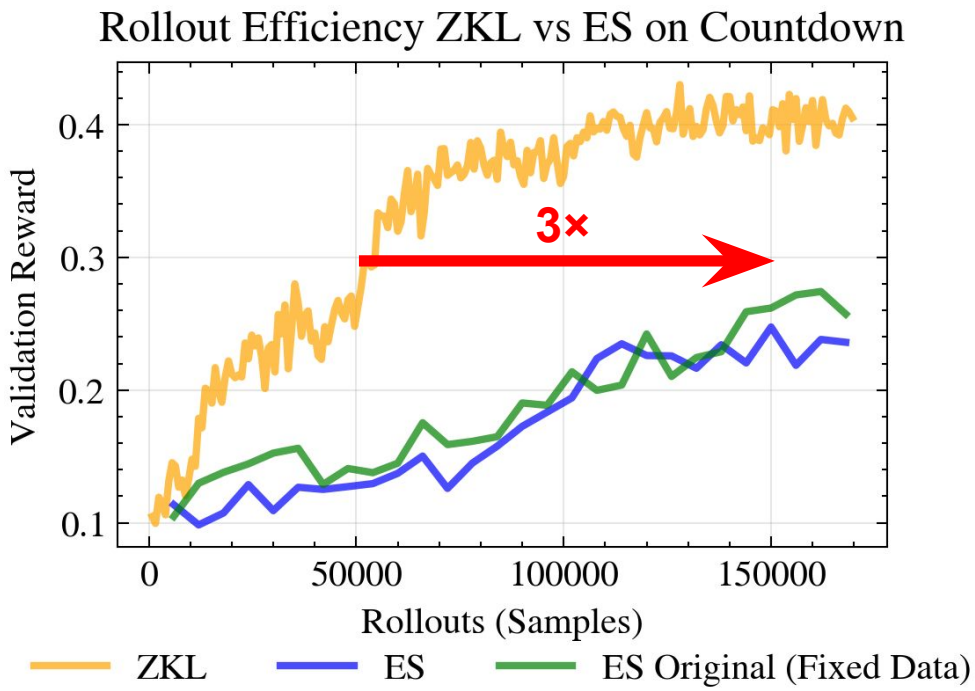

Figure 3: Sample efficiency comparison measured by total rollouts of a Qwen2.5-3B-Instruct model trained on Countdown. ZKL, using $K = 8$ and $B = 100$ with no prompt reuse, achieves higher validation performance with substantially fewer rollouts than ES baselines that rely on larger populations and repeated evaluation on fixed datasets. When training is halted after ZKL completes one epoch over the data, ZKL has already surpassed the baselines, demonstrating superior rollout efficiency.

substantially fewer inference rollouts per effective update than GRPO for a given target performance, while also producing smoother training curves. While the exact optimum shifts with model size and reward noise, the broader conclusion is robust: ZKL operates best with small populations and moderate mini-batches, which is precisely the regime compatible with production inference constraints.

3.4.2 Increased sample efficiency

In the production setting, the relevant metric is reward improvement per unit inference compute. On that metric, ZKL is favorable because it avoids repeated evaluation of identical prompts across perturbations and concentrates compute on rollouts that directly contribute to the update.

Figure 3 shows the rollout–performance tradeoff across methods. For an equivalent number of rollouts, ZKL reaches higher validation performance earlier and converges with substantially fewer total evaluations than prior ES variants by up to 3×. This implies an up to 15× improvement in data efficiency over GPRO from the baseline presented in Figure 6 of Qiu et al. [24]. This reflects both improved sample efficiency and better utilization of inference compute: ZKL allocates nearly all rollouts to unique, update-relevant prompts, whereas full-dataset ES repeatedly expends compute on redundant evaluations that do not change the update direction. As a result, ZKL delivers greater reward improvement per unit inference compute, which is the dominant constraint in deployed systems. This property makes ZKL particularly well suited to production environments, where

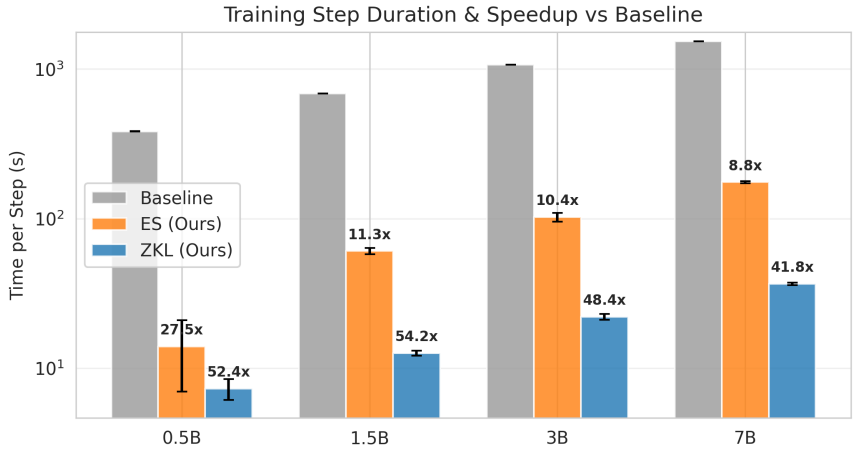

Figure 4: End-to-end training step time across implementations and model sizes on Countdown. Bars show wall-clock duration per step for a baseline implementation, a prior ES setup with large populations and full-batch evaluation, and ZKL with compute-optimal mini-batching. Error bars denote step-to-step variability. ZKL reduces step time by an order of magnitude or more across model sizes, indicating that perturbation overhead is negligible relative to rollout computation. All runs were conducted with DP=4, TP=1 on A100s.

inference throughput, not offline optimization time, defines the effective post-training budget.

Inference Speed. In practice, rollout computation dominates per-step time in the small-population regime. Figure 4 (timing) demonstrates that replacing full-dataset ES evaluation with ZKL's mini-batched online evaluation produces substantial wall-clock speed gains fro their original implementation by about $10\times$, and that choosing the compute-optimal (K, B) setting further increases throughput to up to $50\times$.

3.5 Performance gains

Data and compute scaling. Because ZKL is online and does not reuse prompts within an update step, it scales naturally with large inference streams. The effective dataset can grow without requiring prompt replay or centralized storage. Empirically, increasing the number of rollouts improves validation reward fashion until saturation, after which we obverse a small degradation from the optimal though above the original. It seems that the highest accuracy achieved by a model through ZKL depends on the task and the model capacity which is standard post-training behavior (Figure 5).

3.6 Performance is equivalent when using online regimes

The viability of deploying ZKL in production depends not only on its peak performance, but also on how predictable and safe its optimization dynamics are. By tracking off-target degradation, we provide crucial evidence that the model can be selectively fine-tuned without catastrophically losing its broader foundational skills or drifting into unintended behaviors. In parallel, the hybrid reward function framework acts as a stand-in for real-world conditions, where models must simultaneously optimize for multiple objectives across heterogeneous data distributions. Taken together, these two

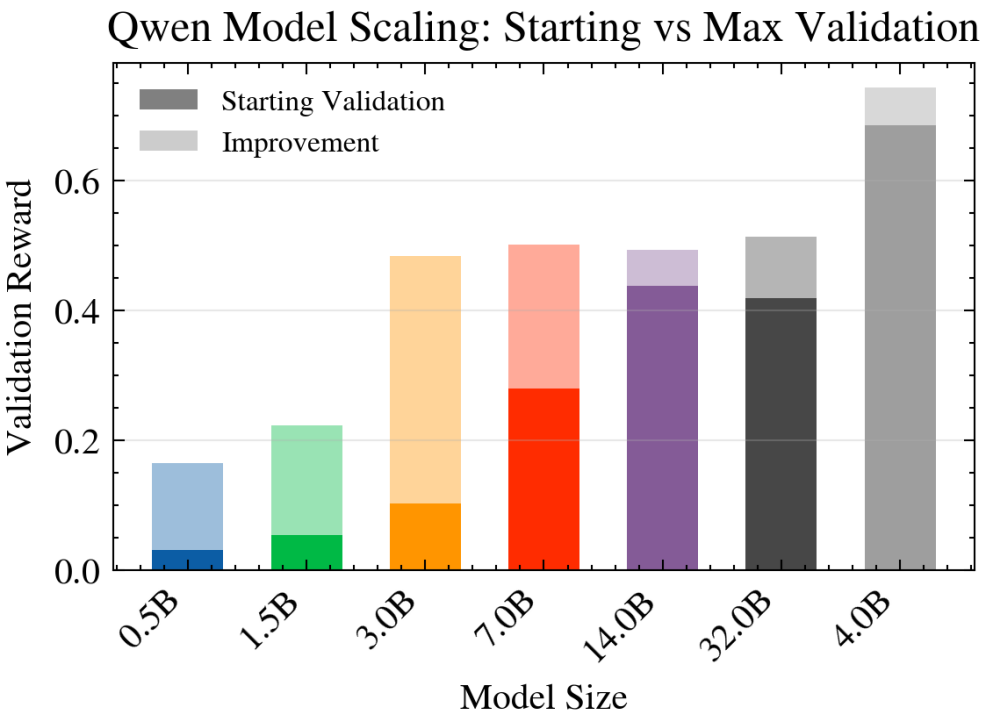

Figure 5: The final performance after ZKL training on Countdown seems decided by intial model capacity; notably Qwen3-4B-Instruct-2507 outperforms much larger models and still improves. Bars represent Qwen models trained with ES ($p = 8$, batch 100/seed, $\sigma = 0.001$, SGD $lr = 5 \times 10^{-4}$). A deeper exploration of what about model capacity makes ZKL effective is left for future research. Solid regions denote initial validation reward; translucent extensions show improvement to peak. While performance generally scales with capacity, relative gains diminish, and the 4B model (Qwen3) notably outperforms the 14B and 32B variants. Models $\leq$14B converged (peaked then degraded), whereas the 32B model continued improving until termination at $\approx$100 GPU-hours. Runs utilized 4$\times$A100 GPUs with memory-optimized parallelism: DP=4/TP=1 for $\leq$7B; DP=4/TP=2 for 14B and 32B.

metrics demonstrate that the ZKL framework captures the core capabilities needed to underpin safe production deployment, beyond the engineering considerations outlined in Sections 3.1 and 3.2.

3.6.1 Scaling

Across tasks, the strongest determinant of final performance is base-model capacity (Figure 5). Larger models both improve faster and reach higher plateaus under the same budgets, and some tasks only become learnable once the model exceeds a capacity threshold. This is most evident in our multi-task results, where reasoning tasks with sparse verifiable rewards require larger base models to show consistent improvement. The implication is that ZKL does not eliminate the need for capable pretraining priors, but it does provide an efficient and stable mechanism to exploit them under privacy constraints.

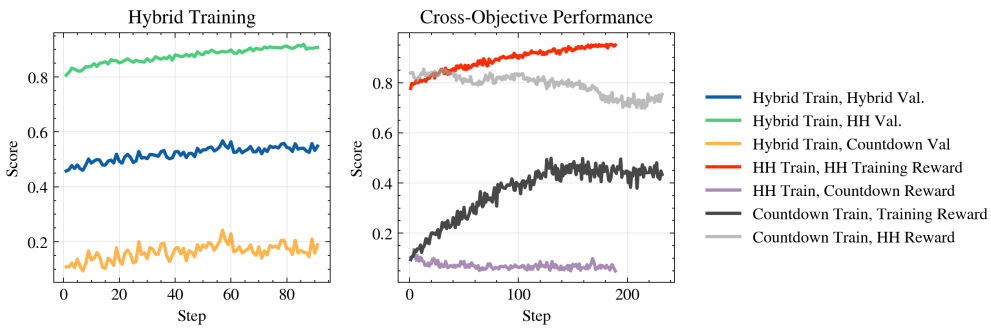

Figure 6: Simultaneous and cross-objective optimization training Qwen2.5-3B-Instruct with ZKL on the HH and Countdown datasets. **Left:** Mixed-stream training on Countdown and HH-RLHF with summed rewards improves both objectives smoothly. **Right:** Cross-objective validation shows no systematic pre-convergence degradation on held-out objectives, suggesting a lower risk of misalignment when training on other objectives.

3.6.2 Cross training works

Optimizing with private or partially observed feedback introduces failure modes uncommon in standard post-training. Hidden prompts, tokens, or outputs make errors harder to diagnose and regressions easier to miss. We highlight three risks and how ZKL mitigates them.

Hidden misalignment. There is a danger of reward hacking and subtle misalignment when the training procedure cannot be closely inspected. However, since this system would be deployed at scale, the model would have to be capable of doing many tasks and would likely have a reward function that integrates many goals which would prevent overfitting to a specific domain. As shown in the left of Figure 6, ZKL supports training on hybrid objectives that have different relative weights or magnitudes. We see when trained on a dataset containing half HH and half Countdown data, the validation on both subsets of the data increases during training, despite their dramatically different reward score. Notably, neither task reaches the same final performance they reach when trained independently, though a deeper analysis as to why this happens is outside the scope of this work.

Off-target degradation. Private reward optimization can improve a narrow objective while silently harming unmeasured tasks. As shown on the right of Figure 6 supports these claims, we do not see a significant degradation in the Countdown task when trained in HH or in the HH task when trained in countdown until convergence. As mentioned in Section 3.6.1, we notice general degradation in model performance in both the trained and untrained tasks after convergence; this might be caused by the model leaving the locally linear active parameter space that allows ES learning to function. Additionally, because ZKL supports multi-objective optimization, reward functions that incentives harmlessness can be optimized at the same time to further reduce the risk of misalignment as shown in the left side of Figure 6.

4 Privacy Guarantees of ZKL

We analyze ZKL under an *honest-but-curious* provider. The provider follows the protocol but logs everything it can observe. It knows the base model, the optimizer, the perturbation distribution, and

all public hyperparameters. The only data-dependent values it receives are, at each step, perturbation seeds and one bounded scalar reward summary per seed. Prompts, contexts, completions, token traces, gradients, activations, and user identifiers stay on the deployment side.

The privacy properties of ZKL follow from *channel reduction*: learning depends on a low-dimensional transcript.

4.1 Structural privacy by construction

ZKL offers *structural privacy*. The provider never has access to raw text, token-wise losses, or any other high-dimensional training signals. Each private interaction can influence training only through a single bounded scalar associated with a seed. This design removes the main leakage pathways found in gradient-based RL and federated learning, where per-example vectors can be aligned, averaged, and exploited. Structural privacy is a property of the system's architecture, not a cryptographic guaranty. It constrains what information can cross the boundary during training, but it does not, on its own, stop the resulting model from potentially encoding that information.

4.2 What the scalar transcript can reveal

A single bounded scalar contains only a small amount of semantic content. Absent strong auxiliary side information, recovering a specific prompt or completion from seed-tagged scalars is underdetermined: many different interactions can produce the same reward for a given perturbed model. Although the provider can reconstruct each update $\Delta\theta_t$ from the seeds and the model weights, those weights are themselves functions only of these scalars. There is no token-level trace connecting information across steps, and no per-example gradient available to invert. This assertion is limited in scope. ZKL does not claim the worst-case indistinguishability. As with any non-DP training approach, membership or property inference can still be performed on the final parameters or via correlations in the observed rewards [34, 16, 5]. The contribution of ZKL is different: it eliminates direct, high-bandwidth leakage during training, forcing any inference to rely instead on indirect, aggregate population-level effects.

4.3 Formal guarantees

Structural privacy constrains what the provider *observes*, not what the final model *reveals*. If formal protection against membership and property inference is required, differential privacy must be enforced. ZKL makes this clean because the provider-visible, data-dependent objects are scalars. If each scalar reward is privatized on the client side before transmission, then the entire transcript the provider sees, including all iterates and checkpoints, inherits (ε, δ)-DP by post-processing and composition [4, 1, 17]. We give a concrete mechanism and accountant in Appendix B, leveraging recent DP results for zeroth-order learning [38, 45, 14, 3].

5 Discussion and Limitations

Zero-Knowledge Learning is a new promising paradigm for post-training large language models. It demonstrates that stable, efficient improvement is possible without access to prompts, tokens, or gradients, and that inference-time feedback can be converted into reliable learning signals under strict privacy constraints. This addresses a practical gap in existing post-training methods, which either depend on centralized data access or impose expensive compute costs.

Our findings suggest that ZKL is primarily constrained by task design and the capacity of the underlying model, rather than by optimization instabilities or the effects of noise. Although private optimization can introduce worries about opaque failure modes and reward hacking, ZKL supports the joint optimization of performance and safety goals, providing a practical way to reduce these risks. More broadly, ZKL offers a route to turn post-training into an ongoing, privacy-preserving procedure that continually learns from deployment while remaining compliant with legal and operational requirements.

The central aim of this work is to present the ZKL protocol as a new algorithmic framework. The empirical findings are intended as a proof of concept, not as a comprehensive benchmark of all LLM abilities. Our experiments were confined to two datasets in order to illustrate the algorithm's robustness in both verifiable and model-based reward settings. As a result, ZKL's behavior on more intricate tasks, such as extended creative writing, advanced software development, or multilingual translation, has not yet been evaluated. Although the initial results are encouraging, the range of tasks we considered was constrained by the computational cost of running high-population ES experiments.

In addition, because this paper concentrates on the architectural design and mathematical specification of the ZKL update rule, many questions about its long-term scaling properties remain unanswered. We did not investigate ZKL's performance on models larger than 32B parameters, nor did we examine how it behaves under the large, heterogeneous prompt distributions characteristic of commercial production deployments. We invite researchers with access to more extensive computing infrastructure to apply the ZKL protocol to a broader suite of benchmarks and to larger model families, with the goal of more fully characterizing the scaling laws of zero-order post-training. Such studies are crucial for assessing whether the sample-efficiency improvements observed in our limited set of tasks extend to the broad competencies expected of frontier models.

Impact Statement

Zero-Knowledge Learning (ZKL) reshapes how Large Language Models are trained by decoupling the optimization process from direct access to training data. Using a zero-order protocol, ZKL enables a model to learn from private inference sessions through a minimal interface: an exchange of scalar rewards and random seeds. This approach overcomes the current data bottleneck in AI, where the most informative training data are locked behind Zero-Data Retention (ZDR) policies or strict privacy constraints. The core result of this work is to show that a model can be stably and efficiently aligned without the training provider ever viewing a prompt, a completion, or a gradient.

ZKL's architecture offers privacy at a structural level. Because the training provider only receives bounded scalar signals, they cannot reconstruct the semantic content of a user's private interactions. This inherently reduces opportunities for data abuse and large-scale surveillance. In addition, ZKL supports multi-objective optimization, allowing practitioners to simultaneously target performance and safety characteristics, such as safety and helpfulness. Our empirical results indicate that ZKL improves reasoning ability without eroding safety mechanisms or causing off-target degradation, provided that tasks remain within the model's competence.

However, ZKL introduces its own set of concerns. Although it conceals the details of the training interactions, it does not guaranty Differential Privacy (DP) for the resulting model parameters. Without extra protections, an adversary could still perform membership inference attacks to test whether a particular record affected training (although we outline an approach to implement this in the appendix B). There is also a risk of covert misalignment: since the provider cannot see tokens during private inference, they may miss instances of reward hacking or subtle forms of bias. In

practice, deploying ZKL safely requires pairing this protocol with local, client-side oversight and periodic model evaluations to ensure that "zero-knowledge" does not lead to zero scrutiny.

References

[1] Martín Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 308–318, 2016. doi: 10.1145/2976749.2978318.

[2] Anthropic. hh-rlhf: Human preference data for helpfulness and harmlessness. `https://huggingface.co/datasets/Anthropic/hh-rlhf`, 2022. Accessed 2026-01-29.

[3] Eli Chien, Wei-Ning Chen, and Pan Li. Privacy amplification in differentially private zeroth-order optimization with hidden states, 2025. URL `https://arxiv.org/abs/2506.00158`.

[4] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3–4):211–407, 2014. doi: 10.1561/0400000042.

[5] Wenjie Fu, Huandong Wang, Chen Gao, Guanghua Liu, Yong Li, and Tao Jiang. Membership inference attacks against fine-tuned large language models via self-prompt calibration. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. URL `https://openreview.net/forum?id=PAWQvrForJ`.

[6] Kanishk Gandhi, Ayush Chakravarthy, Anikait Singh, Nathan Lile, and Noah D. Goodman. Cognitive behaviors that enable self-improving reasoners, or, four habits of highly effective stars, 2025. URL `https://arxiv.org/abs/2503.01307`.

[7] John Garrett, Echedey Luis, H.-H. Peng, Tim Cera, gobinathj, Josh Borrow, Mehmet Keçeci, Splines, Suraj Iyer, Yuming Liu, cjw, and Mikhail Gasanov. garrettj403/scienceplots: 2.1.1, November 2023. URL `https://zenodo.org/records/10206719`.

[8] Mohammad Gheshlaghi Azar, Zhaohan Daniel Guo, Bilal Piot, Remi Munos, Mark Rowland, Michal Valko, and Daniele Calandriello. A general theoretical paradigm to understand learning from human preferences. In *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 238 of *Proceedings of Machine Learning Research*, pages 4447–4455. PMLR, 2024. URL `https://proceedings.mlr.press/v238/gheshlaghi-azar24a.html`.

[9] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989. doi: 10.1137/0218012.

[10] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles (SOSP '23)*, 2023. doi: 10.1145/3600006.3613165.

[11] Joel Lehman, Jay Chen, Jeff Clune, and Kenneth O. Stanley. ES is more than just a traditional finite difference approximator. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pages 450–457, 2018. doi: 10.1145/3205455.3205523.

[12] Ziniu Li, Tian Xu, Yushun Zhang, Yang Yu, Ruoyu Sun, and Zhi-Quan Luo. Remax: A simple, effective, and efficient reinforcement learning method for alignment with human feedback. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235 of *Proceedings of Machine Learning Research*, pages 29128–29163. PMLR, 2024. URL https://proceedings.mlr.press/v235/li24bf.html.

[13] Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023. doi: 10.48550/arXiv.2305.20050. URL https://arxiv.org/abs/2305.20050.

[14] Zhihao Liu, Jinyu Lou, Weixin Bao, Yiming Hu, Bohan Li, Zeyu Qin, and Kui Ren. Differentially private zeroth-order methods for scalable large language model finetuning, 2024. URL https://arxiv.org/abs/2402.07818.

[15] Sadhika Malladi, Tianyu Gao, Aditya Nichani, Aditya Damian, Jason D. Lee, Xuandong Han, Danqi Chen, and Sanjeev Arora. Fine-tuning language models with just forward passes. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. URL https://arxiv.org/abs/2305.17333.

[16] Matthieu Meeus, Shubham Jain, Marek Rei, and Yves-Alexandre de Montjoye. Did the neurons read your book? document-level membership inference for large language models. In *33rd USENIX Security Symposium (USENIX Security 24)*. USENIX Association, 2024. URL https://www.usenix.org/conference/usenixsecurity24/presentation/meeus.

[17] Ilya Mironov. Rényi differential privacy, 2017. URL https://arxiv.org/abs/1702.07476.

[18] Philipp Moritz, Robert Nishihara, Stephanie Wang, Alexey Tumanov, Richard Liaw, Eric Liang, Melih Elibol, Zongheng Yang, William Paul, Michael I. Jordan, and Ion Stoica. Ray: A distributed framework for emerging AI applications, 2017. URL https://arxiv.org/abs/1712.05889.

[19] Yurii Nesterov and Vladimir Spokoiny. Random gradient-free minimization of convex functions. *Foundations of Computational Mathematics*, 17:527–566, 2017. doi: 10.1007/s10208-016-9330-3.

[20] Henrik Nolte, Michèle Finck, and Kristof Meding. Machine learners should acknowledge the legal implications of large language models as personal data, 2025. URL https://arxiv.org/abs/2503.01630.

[21] 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. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022. URL https://papers.nips.cc/paper_files/paper/2022/hash/b1efde53be364a73914f58805a001731-Abstract.html.

[22] Jiayi Pan, Xiuyu Li, Long Lian, Charlie Snell, Yifei Zhou, Adam Yala, Trevor Darrell, Kurt Keutzer, and Alane Suhr. Learning adaptive parallel reasoning with language models, 2025. URL https://arxiv.org/abs/2504.15466.

[23] Zeyu Qin et al. Federated full-parameter tuning of billion-sized language models with communication cost under 18 kilobytes, 2023. URL https://arxiv.org/abs/2312.06353.

[24] X. Qiu, C. Gan, et al. Evolution strategies at scale: LLM fine-tuning beyond reinforcement learning, 2025. URL https://arxiv.org/abs/2509.24372.

[25] Qwen Team. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

[26] Qwen Team, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, et al. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115. arXiv:2412.15115v2.

[27] 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. *Advances in Neural Information Processing Systems*, 36, 2023. URL https://papers.nips.cc/paper_files/paper/2023/hash/3b9d8b5f1e7b1a5a6b0b7a8c4b6f8d3b-Abstract.html.

[28] Rajkumar Ramamurthy, Prithviraj Ammanabrolu, Kianté Brantley, Jack Hessel, Rafet Sifa, Christian Bau, Hannaneh Hajishirzi, and Yejin Choi. Is reinforcement learning (not) for natural language processing: Benchmarks, baselines, and building blocks for natural language policy optimization. In *International Conference on Learning Representations (ICLR)*, 2023. URL https://openreview.net/forum?id=8uA8Gxqj0u.

[29] Ingo Rechenberg. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Frommann-Holzboog, 1973.

[30] Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning, 2017. URL https://arxiv.org/abs/1703.03864.

[31] Bidipta Sarkar, Mattie Fellows, Juan Agustin Duque, Alistair Letcher, Antonio León Villares, Anya Sims, Dylan Cope, Jarek Liesen, Lukas Seier, Theo Wolf, Uljad Berdica, Alexander David Goldie, Aaron Courville, Karin Sevegnani, Shimon Whiteson, and Jakob Nicolaus Foerster. Evolution strategies at the hyperscale, 2025. URL https://arxiv.org/abs/2511.16652.

[32] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

[33] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Xiao, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

[34] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 3–18. IEEE, 2017. doi: 10.1109/SP.2017.41.

[35] James C. Spall. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE Transactions on Automatic Control*, 37(3):332–341, 1992. doi: 10.1109/9.119632.

[36] James C. Spall. *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control*. John Wiley & Sons, 2003. ISBN 978-0-471-33052-3.

[37] Robin Staab, Mark Vero, Mislav Balunović, and Martin Vechev. Beyond memorization: Violating privacy via inference with large language models. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024. doi: 10.48550/arXiv.2310.07298. URL https://openreview.net/forum?id=kmn0BhQk7p.

[38] Xinyu Tang, Ashwinee Panda, Milad Nasr, Saeed Mahloujifar, and Prateek Mittal. Private fine-tuning of large language models with zeroth-order optimization, 2024. URL https://arxiv.org/abs/2401.04343.

[39] Anirudh Vemula, Wen Sun, and J. Andrew Bagnell. Contrasting exploration in reinforcement learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 2801–2810, 2019. URL http://proceedings.mlr.press/v89/vemula19a.html.

[40] Pablo Villalobos, Jaime Sevilla, Lennart Heim, Tamay Besiroglu, Marius Hobbhahn, and Anson Ho. Will we run out of data? an analysis of the limits of scaling datasets in machine learning, 2022. URL https://arxiv.org/abs/2211.04325.

[41] Daan Wierstra, Tom Schaul, Tobias Glasmachers, Yi Sun, Jan Peters, and Jürgen Schmidhuber. Natural evolution strategies. *Journal of Machine Learning Research*, 15(1):949–980, 2014. URL http://jmlr.org/papers/v15/wierstra14a.html.

[42] Yuhang Yao, Jianyi Zhang, Junda Wu, Chengkai Huang, Yu Xia, Tong Yu, Ruiyi Zhang, Sungchul Kim, Ryan Rossi, Ang Li, Lina Yao, Julian McAuley, Yiran Chen, and Carlee Joe-Wong. Federated large language models: Current progress and future directions, 2024. URL https://arxiv.org/abs/2409.15723.

[43] Sixing Yu, W. Qian, and A. Jannesari. Resource-aware federated learning using knowledge extraction and multi-model fusion. In *IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)*, 2022. doi: 10.1109/TPS-ISA56419.2022.00028.

[44] Hao Zhang, Fang Li, Samyak Rawlekar, and Narendra Ahuja. Revisiting zeroth-order optimization for memory-efficient LLM fine-tuning: A benchmark. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235 of *Proceedings of Machine Learning Research*, pages 59191–59209. PMLR, 2024. URL https://proceedings.mlr.press/v235/zhang24ad.html.

[45] Liang Zhang, Bingcong Li, Kiran Koshy Thekumparampil, Sewoong Oh, and Niao He. DPZero: Private fine-tuning of language models without backpropagation. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235 of *Proceedings of Machine Learning Research*, pages 59210–59246. PMLR, 2024. URL https://proceedings.mlr.press/v235/zhang24af.html.

[46] Zhen Zhang et al. The lessons of developing process reward models in mathematical reasoning, 2025. URL https://arxiv.org/abs/2501.07301.

A Proof of Unbiased Estimator

Formally, we define prompt-response pairs (x_i, y_i) where $y_i = f(x_i|\theta)$ with f as our model and $\theta \in \mathbb{R}^p$ as our model parameters. Additionally, models are fine tuned with arbitrary reward functions $R(x_i, y_i) = R(x_i, f(x_i|\theta))$ where response pairs are scored $(x_i, y_i) \mapsto [0, 1]$. In our setup, ZKL adds small perturbations $\delta_j \in \mathbb{R}^p$ to our model parameters θ and generates rewards $r_{ij} = R(x_i, f(x_i|\theta+\delta_j))$ for each prompt x_i. Now, taking the expected value of these rewards, we define our parameter-reward function $\mu_\theta(\delta_j) := \mathbb{E}[R(x_i, f(x_i|\theta + \delta_j))]$ where $\mu_r : \mathbb{R}^p \to \mathbb{R}$.

For our two assumptions, we first assume that μ_θ is locally linear in δ_j. That is, there exists a linear functional $R^T \in \mathbb{R}^p$ such that for sufficiently small perturbations δ_j, we have $\|\mu_\theta(\delta_j) - R^T\delta_j\| < \varepsilon$ for some small ε. Secondly, we assume rewards take the form of $r_{ij} = R^T\delta_j + \varepsilon_{ij}$ where $\mathbb{E}[\varepsilon_{ij}|\delta_j] = 0$ and ε_i is defined as a random variable collectively representing all randomness from prompt selection, response generation, and reward model scoring.

We now want to show that for iid. $\delta_j \sim \mathcal{N}(0, \sigma I)$, our estimator $\hat{R}^T = \sum_{j=1}^K w_j\delta_j^T$ is unbiased in the direction of R^T, that is there exists an α such that $\mathbb{E}[\alpha R^T - \hat{R}^T] = 0$. Expanding our estimator, we see that:

$$\hat{R}^T = \sum_{j=1}^K w_j\delta_j^T = \sum_{j=1}^K \frac{1}{s}\left(\frac{1}{N}\sum_{i=1}^N r_{ij} - \frac{1}{NK}\sum_{i=1}^N\sum_{j=1}^K r_{ij}\right)\delta_j^T$$

Substituting in $r_{ij} = R^T\delta_j + \varepsilon_{ij}$, we get:

$$\begin{aligned}
\hat{R}^T &= \sum_{j=1}^K \frac{1}{s}\left(\frac{1}{N}\sum_{i=1}^N (R^T\delta_j + \varepsilon_{ij}) - \frac{1}{NK}\sum_{i=1}^N\sum_{j=1}^K (R^T\delta_j + \varepsilon_{ij})\right)\delta_j^T \\
&= \frac{1}{s}\sum_{j=1}^K \left(R^T\delta_j + \frac{1}{N}\sum_{i=1}^N \varepsilon_{ij} - \frac{1}{K}\sum_{j=1}^K \left(R^T\delta_j + \frac{1}{N}\sum_{i=1}^N \varepsilon_{ij}\right)\right)\delta_j^T \\
&= \frac{1}{s}\sum_{j=1}^K \left(R^T\delta_j + \bar{\varepsilon}_j - R^T\bar{\delta} - \bar{\varepsilon}\right)\delta_j^T \\
&\propto R^T\sum_{j=1}^K (\delta_j - \bar{\delta})\delta_j^T + \sum_{j=1}^K (\bar{\varepsilon}_j - \bar{\varepsilon})\delta_j^T
\end{aligned}$$

Where $\bar{\delta}_j := \frac{1}{K}\sum_{j=1}^K \delta_j$, $\bar{\varepsilon}_j := \frac{1}{N}\sum_{i=1}^N \varepsilon_{ij}$, and $\bar{\varepsilon} := \frac{1}{NK}\sum_{j=1}^K\sum_{i=1}^N \varepsilon_{ij}$. Now, given that p is large enough that we may treat the $\delta_j \in \mathbb{R}^p$ as approximately orthogonal and each $\delta_j \sim \mathcal{N}(0, \sigma^2 I)$, we take expectations of $\hat{R}^T$. Since $\mathbb{E}[\varepsilon_{ij}|\delta_j] = 0$ and ε_{ij} is independent of δ_j, we have:

$$\mathbb{E}\Big[\sum_{j=1}^K (\bar{\varepsilon}_j - \bar{\varepsilon})\delta_j^T\Big] = 0.$$

Thus,

$$\mathbb{E}[\hat{R}^T] \propto R^T\,\mathbb{E}\left[\sum_{j=1}^K (\delta_j - \bar{\delta})\delta_j^T\right]$$

Write

$$M := \sum_{j=1}^{K} (\delta_j - \bar{\delta})\delta_j^T = \sum_{j=1}^{K} \delta_j \delta_j^T - K\,\bar{\delta}\bar{\delta}^T.$$

Using $\delta_j \sim \mathcal{N}(0, \sigma^2 I)$ independently,

$$\mathbb{E}[\delta_j \delta_j^T] = \sigma^2 I, \quad \Longrightarrow \quad \mathbb{E}\Big[\sum_{j=1}^{K} \delta_j \delta_j^T\Big] = K\sigma^2 I,$$

and since $\bar{\delta} \sim \mathcal{N}(0, \frac{\sigma^2}{K} I)$,

$$\mathbb{E}[K\,\bar{\delta}\bar{\delta}^T] = \sigma^2 I.$$

Therefore,

$$\mathbb{E}[M] = (K-1)\sigma^2 I.$$

Substituting back,

$$\mathbb{E}[\hat{R}^T] \propto R^T (K-1)\sigma^2 I = \alpha R^T,$$

for $\alpha = \frac{(K-1)\sigma^2}{s}$. Hence there exists an α such that

$$\mathbb{E}[\alpha R^T - \hat{R}^T] = 0,$$

so $\hat{R}^T$ is unbiased in the direction of R^T.

B Even Stronger Privacy Variant Derivation

We assume any noise used to privatize scalar rewards is added by the client or a trusted execution environment prior to transmission; the provider is not trusted to perform privatization.

To obtain a theorem-grade guarantee against the provider, the cleanest route is to add DP noise to what the provider sees (the scalars), then invoke post-processing.

Definition B.1 (Neighboring datasets)**.** Let a *record* be one private interaction (prompt, context, completion, and its locally computed reward). Two datasets D, D' are neighbors if they differ in one record.

Definition B.2 ((ε, δ)-DP [4])**.** A randomized mechanism $\mathcal{M}$ is (ε, δ)-DP if for all neighboring D, D' and all measurable events E:

$$\Pr[\mathcal{M}(D) \in E] \leq e^{\varepsilon} \Pr[\mathcal{M}(D') \in E] + \delta.$$

Mechanism to privatize ZKL. For each perturbation j at step t, privatize the mean reward by adding Gaussian noise with an unknown seed:

$$\tilde{r}_{t,j} = \bar{r}_{t,j} + \mathcal{N}(0, \tau^2). \tag{1}$$

Then compute weights $w_{t,j}$ and the update θ_{t+1} using only $\tilde{r}_{t,1:K}$ and seeds.

Lemma B.3 (Sensitivity of mean reward)**.** *Assume per-record rewards are clipped to* $[0, 1]$*. Replacing one record in a minibatch changes* $\bar{r}_{t,j}$ *by at most* $1/B$*. Hence the* ℓ_2 *sensitivity of* $\bar{r}_{t,j}$ *is* $\Delta = 1/B$*.*

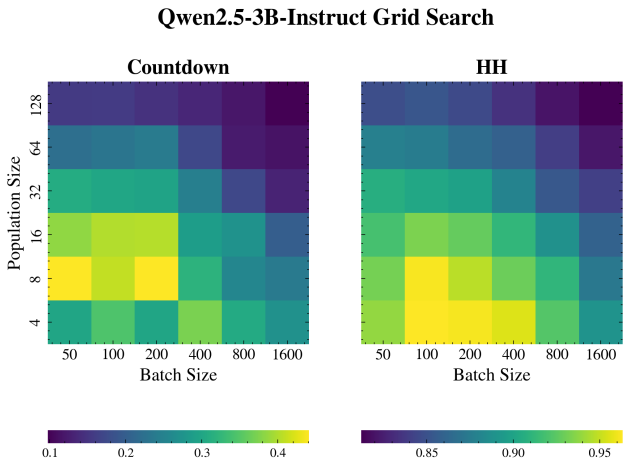

Figure 7: Compute-optimal tradeoff between ES population size K and per-perturbation batch size B under a fixed two-hour budget on four A100 GPUs for Qwen2.5-7B-Instruct trained on Countdown and HH datasets. Color indicates achieved validation performance. While the exact loss landscape and optimal hyperparameters do change with problem, learning rate, and model size, small populations and batch sizes are generally best under constrained compute. Increasing population without increasing batch size degrades performance due to higher reward noise. We found $K = 8$ and $B \in 100, 200$ to generally perform best across different problems and model sizes. A batch size of 100 is often preferable to a batch size of 200 in our setup, even if there is a small performance penality because it requires less data and converges twice as fast.

Theorem B.4 (Per-perturbation DP via Gaussian mechanism [4])**.** *Under clipping to* $[0, 1]$*, releasing* $\tilde{r}_{t,j} = \bar{r}_{t,j} + \mathcal{N}(0, \tau^2)$ *is* (ε, δ)*-DP for one record in* $S_{t,j}$ *provided*

$$\tau \;\geq\; \frac{\Delta\sqrt{2\ln(1.25/\delta)}}{\varepsilon} \quad \textit{with } \Delta = 1/B.$$

Theorem B.5 (Step-level DP for the provider-visible transcript)**.** *At step* t*, release* $\tilde{\mathbf{r}}_t = (\tilde{r}_{t,1}, \ldots, \tilde{r}_{t,K})$*. If each coordinate uses the same* (ε, δ) *Gaussian mechanism and the* K *minibatches are disjoint (as in ZKL within-step non-reuse), then* $\tilde{\mathbf{r}}_t$ *is* (ε, δ)*-DP* with respect to any single record *at that step, up to standard group/parallel composition bookkeeping [4].*

Theorem B.6 (Transcript DP across many steps (composition))**.** *Let* $\mathcal{T}$ *be the full provider transcript across* T *steps: seeds plus privatized scalars plus all iterates* (θ_t)*. If each step is* $(\varepsilon_t, \delta_t)$*-DP with respect to record-level adjacency, then the full transcript is* $\left(\sum_{t=1}^{T} \varepsilon_t, \sum_{t=1}^{T} \delta_t\right)$*-DP by basic composition [4]. Tighter bounds follow via Rényi DP / moments accountant [17, 1].*

Remark B.7 (Post-processing closes the loop). All downstream quantities the provider computes from the privatized scalars, including $w_{t,j}$, $\Delta\theta_t$, checkpoints, and even a released final model, are post-processing of the DP transcript. DP is therefore preserved automatically [4].

C Additional Gridsearches