CHIP: ONE χ^2 BALL FOR BOTH ERRORS OF REINFORCEMENT LEARNING IN LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning updates a large language model with data written by the model's previous version, and the error of one such update splits into exactly two terms: a *bias*, because the objective we can compute scores the new policy on the states the old policy visited, and a *variance*, because that objective is estimated from a finite batch. We find that the χ^2 divergence between the new and old policies controls both errors. For ideal importance weights, χ^2 equals their variance, which governs finite-batch estimation noise; the same χ^2 budget also bounds the total-variation distance in the bias bound. We enforce this single budget at every sampled position through a projection layer that replaces the actor's proposed next-token distribution with the closest distribution inside the χ^2 ball around the old policy. The solution has a normalized weighted harmonic form governed by one scalar, and implicit differentiation gives its gradients in closed form. The layer adds less than 1% to each training step and replaces clipping at the same point in standard training recipes. We call this algorithm **CHIP** (CHi-squared Projections). Replacing clipping with CHIP raises validation accuracy in all five estimator families we tested, by $+0.7$ to $+10.7$ points. In a six-scale Qwen3 ladder, CHIP leads in seventeen of eighteen cells and achieves an average validation gain of $+6.9$ points over the last sixty steps. On multi-turn agentic coding tasks, CHIP achieves 45.8% pass@1 on SWE-bench Verified at training step 130, outperforming CLIP by 6.8 percentage points.

1 INTRODUCTION

Reinforcement learning (RL) is now a standard final stage for training large language models, using either human preferences (Ziegler et al., 2019; Stiennon et al., 2020; Ouyang et al., 2022) or automatically checkable answers (Shao et al., 2024; Yu et al., 2025). A typical training cycle has three steps. *(1) Rollout:* freeze the current model as the *behavior policy* π_{old} and use it to generate a batch of responses. *(2) Scoring:* assign each generated token an *advantage* $\hat{A}_t$, which measures whether choosing that token led to a better-than-average outcome. *(3) Update:* optimize the trainable model, or *actor*, π_θ, to make positive-advantage tokens more likely and negative-advantage tokens less likely (Sutton et al., 1999). Recent work has proposed many ways to compute the advantages (Schulman et al., 2016; Shao et al., 2024; Liu et al., 2025; Hu, 2025); this paper studies how the resulting batch is used in step 3.

The two errors in one update. One update separates what we want from what the optimizer sees through two replacements. First, generating new rollouts after every gradient step is too expensive, so the update is *off-policy*: π_θ is optimized using a batch produced by π_{old}. The *true improvement* is how much better the new policy actually performs. Since it depends on states visited by the new policy, old-policy data can only provide an *exact objective*: the full expectation of an importance-weighted score on states visited by π_{old}. Their difference is **layer 1**, a systematic bias from reusing old-policy data. Second, the optimizer cannot compute this expectation and instead uses its *batch estimate* on one minibatch. Their difference is **layer 2**, the sampling error of a finite batch. The total

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

error is exactly the sum of the two:

$$\underbrace{\text{true improvement} - \text{batch estimate}}_{\text{total error of the update}} = \underbrace{\text{true improvement} - \text{exact objective}}_{\textbf{layer 1: bias}} + \underbrace{\text{exact objective} - \text{batch estimate}}_{\textbf{layer 2: variance}}.$$

Layer 1 remains even with unlimited data; layer 2 is finite-batch noise, because each update uses only one estimate of the exact objective.

Most existing policy-optimization algorithms address only layer 1 (Kakade and Langford, 2002; Schulman et al., 2015; 2017). They keep the exact objective close to the true improvement by limiting how far the new policy moves. But layer 1 is only one part of the end-to-end error of an RL update: the optimizer uses a finite-batch estimate of that objective, leaving layer 2 uncontrolled.

This paper. We find that the χ^2 divergence between the new and old policies controls both error layers of an RL update. This leads to a simple principle: keep the χ^2 divergence within its budget at every update, thereby controlling both sources of error. To enforce it, at every sampled position the network proposes a next-token distribution as usual; before the proposal enters the loss, we replace it with the closest distribution that satisfies the constraint. The constraint defines a χ^2 ball around the old policy. Finding the point in this ball closest to the proposal is mathematically called a *projection*. Because this projection is built into the network, the constraint holds by construction, just as softmax makes a valid probability distribution part of the architecture.

The projection problem has a strictly convex objective over a convex feasible set; its KKT conditions therefore characterize the unique global optimum. We prove that this optimum has a closed-form expression: a normalized weighted harmonic mean of the proposal and the old policy, governed by one scalar. By implicitly differentiating the scalar equation that determines this parameter, we also obtain the gradients of the projected solution in closed form. Both the forward and backward passes cost $O(k)$ on a small kept set; in practice, the layer adds less than 1% to each training step. We call this χ^2 projection method **CHIP** (CHi-squared Projections). Compared with existing methods for controlling policy updates, CHIP controls both error layers with one constraint. Unlike clipping, it constrains the full distribution without zeroing token gradients; unlike TRPO, it remains first order and requires no second-order optimization. Because it occupies the same place as clipping, CHIP is a drop-in replacement for standard training recipes.

Contributions. *(1)* We decompose the error of one RL update into a bias layer and a variance layer, and show that a single quantity, the χ^2 divergence between the new and old policies, controls both at once. *(2)* We formulate a convex projection problem whose solution keeps every update inside the χ^2 budget by construction. It admits a closed-form solution and closed-form gradients, and adds less than 1% to each training step. *(3)* Across five policy-optimization algorithms and six model scales, replacing clipping with CHIP raises validation accuracy and reduces the late-training degradation of clipped baselines. On a multi-turn agentic software-engineering task, CHIP remains stable while the clipped control declines. Proofs are in the appendix.

2 The two error layers of one update

2.1 What we want and what we optimize

Let $\eta(\pi)$ be the true expected return of policy π and $A_\pi(s,a)$ its advantage function. What an update should increase is the performance difference between the new policy and the old one, and the *performance-difference lemma* (Kakade and Langford, 2002) states it exactly:

$$\eta(\pi_{\text{new}}) - \eta(\pi_{\text{old}}) = \mathbb{E}_{s\sim d_{\pi_{\text{new}}},\, a\sim\pi_{\text{new}}}\big[A_{\pi_{\text{old}}}(s,a)\big], \tag{1}$$

d_π being how often π finds itself in each state: *the improvement of a new policy equals the old policy's advantage, averaged over the states and actions the new policy visits.* Exact, but not usable: it averages over states visited by the *new* policy, which we cannot sample before committing to it. Replacing $d_{\pi_{\text{new}}}$ by $d_{\pi_{\text{old}}}$ and correcting the action distribution by *importance weighting* gives the objective we can actually write down,

$$L_{\pi_{\text{old}}}(\pi) = \mathbb{E}_{(s,a)\sim\pi_{\text{old}}}\left[r(s,a)\, A_{\pi_{\text{old}}}(s,a)\right], \qquad r(s,a) = \frac{\pi(a \mid s)}{\pi_{\text{old}}(a \mid s)}. \tag{2}$$

The ratio $r(s, a)$ tells us how much more or less likely the new policy is to choose action a at state s than the old policy. Every expectation in (2) runs over the old policy, so $L_{\pi_{\text{old}}}$ can be estimated from the data already in the rollout buffer; but it is still an expectation, and what enters training is a sample average over one minibatch,

$$\hat{L} = \frac{1}{N} \sum_{i=1}^{N} r_i \, \hat{A}_i, \qquad (s_i, a_i) \sim \pi_{\text{old}}, \tag{3}$$

Two substitutions therefore separate what we want from what we optimize: the states were swapped ($d_\pi \to d_{\pi_{\text{old}}}$), and the expectation was swapped for one batch ($L \to \hat{L}$); each leaves an error, and the next subsection takes them in turn.

2.2 The two error layers

Layer 1 comes from the first substitution. Equation (2) is exact at $\pi = \pi_{\text{old}}$ and drifts from (1) as π moves away, because it keeps scoring the new policy on the old one's states. That drift is layer 1, and it is what the classical analysis bounds.

Theorem 1 (Schulman et al., 2015). *With* $\alpha = \max_s D_{\text{TV}}\big(\pi(\cdot \mid s), \pi_{\text{old}}(\cdot \mid s)\big)$ *and* $\epsilon_{\max} = \max_{s,a} |A_{\pi_{\text{old}}}(s, a)|$,

$$\eta(\pi) - \eta(\pi_{\text{old}}) \;\geq\; L_{\pi_{\text{old}}}(\pi) - \frac{4\epsilon_{\max}\gamma}{(1-\gamma)^2}\,\alpha^2.$$

The theorem says when an update is guaranteed to improve true performance: the exact objective $L_{\pi_{\text{old}}}(\pi)$ must exceed the penalty caused by moving away from the old policy. This penalty depends on α, the maximum total-variation distance between the new and old policies. However, the optimizer uses only the finite-batch estimate $\hat{L}$, not the exact objective itself. The theorem therefore controls layer 1, but not layer 2.

Layer 2 comes from replacing the expectation with one minibatch. Importance weighting makes the batch estimate correct on average, $\mathbb{E}[\hat{L}] = L_{\pi_{\text{old}}}(\pi)$, but any particular minibatch can over- or underestimate the exact objective. The optimizer updates the policy using this particular estimate, so this sampling error affects the update. With N samples, its variance is $\text{Var}(\hat{L}) = \text{Var}(r\hat{A})/N$. Because each sample is multiplied by the importance weight r, highly variable weights can make the batch estimate noisier. Theorem 1 does not control this finite-batch error. This is layer 2.

3 One quantity bounds both layers

Section 2 left two error terms of different kinds: one is controlled by how far the policy moved, the other by how noisy a finite batch is. This section shows that a single quantity measures both.

3.1 Layer 2 is a divergence

Lemma 1 (Weight identities). *Let* $\pi, \pi_{\text{old}} \in \Delta(\mathcal{V})$ *with* $\pi_{\text{old}} > 0$ *and* $r(x) = \pi(x)/\pi_{\text{old}}(x)$. *Then*

$$\mathbb{E}_{\pi_{\text{old}}}[r] = 1, \qquad \text{Var}_{\pi_{\text{old}}}(r) = \mathbb{E}_{\pi_{\text{old}}}[r^2] - 1 = \sum_x \frac{\big(\pi(x) - \pi_{\text{old}}(x)\big)^2}{\pi_{\text{old}}(x)} = \chi^2(\pi \,\|\, \pi_{\text{old}}). \tag{4}$$

For an i.i.d. minibatch of size N *with* $|\hat{A}| \leq \epsilon_A$, *the estimate in* (3) *satisfies*

$$\text{Var}(\hat{L}) = \frac{1}{N}\text{Var}_{\pi_{\text{old}}}(r\hat{A}) \leq \frac{\epsilon_A^2}{N}\big(1 + \chi^2(\pi \| \pi_{\text{old}})\big).$$

The lemma gives two simple facts. First, the importance weights average to 1. When x is sampled from π_{old}, $\pi_{\text{old}}(x)$ cancels the denominator of $r(x)$, leaving $\pi(x)$, whose probabilities sum to 1. Second, for the ideal weights defined above, their variance is exactly the χ^2 divergence between the new and old policies. The last line connects this identity directly to training: because $\hat{L}$ averages N weighted samples, a χ^2 budget bounds $\text{Var}(\hat{L})$. This is how χ^2 controls layer 2. Appendix C gives the proof.

3.2 THE SAME BALL BOUNDS LAYER 1

Proposition 1 (A χ^2 budget is a total-variation budget)**.** *For all $p, q \in \Delta(\mathcal{V})$ with $q > 0$,*

$$\chi^2(p \,\|\, q) \;\geq\; D_{\mathrm{TV}}(p, q)^2. \tag{5}$$

Hence at every state where $\chi^2(\bar{\pi}_\theta \| \pi_{\mathrm{old}}) \leq \varepsilon$ holds, $D_{\mathrm{TV}}(\bar{\pi}_\theta, \pi_{\mathrm{old}})^2 \leq \varepsilon$. If it holds at every *state for the policy being evaluated, then $\alpha^2 \leq \varepsilon$ and Theorem 1 applies.*

Theorem 1 measures layer 1 through α^2, the squared total-variation distance between the new and old policies. Equation (5) shows that a χ^2 budget gives $\alpha^2 \leq \varepsilon$ whenever the budget holds at every state (Appendix D gives the proof). Lemma 1 shows that the same budget bounds $\mathrm{Var}(\hat{L})$ in layer 2. **One constraint, both layers**: this is why we call the χ^2 ball a *trust region* in this paper.[1]

3.3 WHAT EACH CONSTRAINED POSITION IS GUARANTEED

Per-token movement. A χ^2 budget directly limits how much each token probability can change. If $\chi^2(p \| \pi_{\mathrm{old}}) \leq \varepsilon$, then every token x satisfies $\left|p(x) - \pi_{\mathrm{old}}(x)\right| \leq \sqrt{\varepsilon \pi_{\mathrm{old}}(x)}$, because the contribution of any one token cannot exceed the total χ^2 budget. The bound applies in both directions and to every token at once. Since π_{old} is refreshed after every rollout, a large change can still occur, but it must be spread over several updates. RL-trained models are known to lose entropy (Cui et al., 2025a) through sharp one-step decreases in token probabilities; this bound therefore predicts that CHIP should preserve higher entropy than clipping, as confirmed in Section 6.1.

Effective sample size. The same budget also quantifies how much information remains in a reweighted batch. When importance weights are unequal, N samples carry less information than N equally weighted samples; this is measured by the *effective sample size*, $\mathrm{ESS} = N(\mathbb{E}[r])^2/\mathbb{E}[r^2]$ (Kong, 1992; Owen, 2013). For the ideal importance weights, Lemma 1 gives $\mathrm{ESS} = N/(1 + \chi^2(\pi \| \pi_{\mathrm{old}})) \geq N/(1+\varepsilon)$. A smaller χ^2 budget therefore guarantees a larger effective sample size: the tighter the χ^2 ball, the more of the batch's nominal information is retained.

3.4 WHY THE QUANTITY HAS TO BE χ^2 AND NOT KL

KL is a common divergence between probability distributions (Becker et al., 2026). One might therefore hope that a sufficiently tight KL budget would control both error layers. It cannot. The failure comes from layer 2: KL divergence does not bound the finite-batch sampling error, as shown in the following proposition.

Proposition 2 (KL blind spot)**.** *For every $\varepsilon > 0$ and every $M > 0$, there exist policies π and π_{old} on an alphabet of at least two tokens such that $\mathrm{KL}(\pi \| \pi_{\mathrm{old}}) \leq \varepsilon$ while $\mathrm{Var}_{\pi_{\mathrm{old}}}(r) = \chi^2(\pi \| \pi_{\mathrm{old}}) \geq M$.*

Put simply, no matter how small the KL budget is, the layer-2 sampling variance can still be arbitrarily large. A χ^2 budget gives the missing control. For the ideal importance weights, Lemma 1 gives $\mathrm{Var}_{\pi_{\mathrm{old}}}(r) = \chi^2(\pi \| \pi_{\mathrm{old}})$, so $\chi^2(\pi \| \pi_{\mathrm{old}}) \leq \varepsilon$ directly implies $\mathrm{Var}_{\pi_{\mathrm{old}}}(r) \leq \varepsilon$. Thus χ^2 provides a direct upper bound on layer-2 variance, while KL does not.

Appendix C gives the general construction. For example, suppose the old policy assigns a token $x^\star$ probability 10^{-6}, while the new policy assigns it probability 10^{-3}, giving an importance ratio of 10^3. Its contribution to KL is $10^{-3} \ln 10^3 \approx 0.0069$. If $\varepsilon = 0.05$, this uses only 14% of the KL budget, so the KL constraint allows this change. However, the same token contributes $10^{-6} \times 10^6 = 1$ to $\mathbb{E}_{\pi_{\mathrm{old}}}[r^2]$, whose value is 1 when the two policies agree. This single token therefore roughly doubles the second moment. In a batch of 10^5 tokens, it appears only once every ten batches on average: most batches miss it, while an occasional batch contains it with weight 10^3. The resulting batch estimates can therefore vary sharply even though the KL divergence is small. For the same example with $\varepsilon = 0.05$, the token alone contributes approximately $10^{-6}(10^3 - 1)^2 \approx 1$, about twenty times the entire χ^2 budget, so the χ^2 constraint rejects the change.

[1] In TRPO, "trust region" refers to controlling layer 1, the surrogate error. We use the term for the two-layer error studied here: the χ^2 ball bounds the total-variation quantity in layer 1 and the finite-batch variance in layer 2. The global layer-1 guarantee of Theorem 1 requires the constraint at every state; CHIP enforces it at sampled positions.

4 CHIP: ENFORCING THE BUDGET BY CONSTRUCTION

We enforce $\chi^2(\cdot\|\pi_{\text{old}}) \leq \varepsilon$ at every sampled position *by construction*: the actor's raw distribution passes through a projection layer, and the projected distribution is what enters the loss.

4.1 THE PROJECTION AND ITS CLOSED FORM

At one position we have two distributions: the actor output π_θ, which is where the optimizer *wants* to go, and the stored behavior distribution π_{old}, against which the constraint is measured. The projection solves

$$\bar{\pi}_\theta = \arg\min_{p \in \Delta(\mathcal{V})} \chi^2\big(p \,\|\, \pi_\theta\big) \text{ s.t. } \chi^2\big(p \,\|\, \pi_{\text{old}}\big) \leq \varepsilon. \tag{6}$$

In words, the constraint defines the allowed distributions around π_{old}, and the objective selects from them the distribution closest to the actor output π_θ. The constraint provides the guarantees in Section 3. Using χ^2 for both the objective and the constraint simplifies the optimization and reduces it to one-dimensional root-finding.

Theorem 2 (Closed-form harmonic projection)**.** *Problem* (6) *has a unique solution. If* $\chi^2(\pi_\theta\|\pi_{\text{old}}) \leq \varepsilon$ *then* $\bar{\pi}_\theta = \pi_\theta$*. Otherwise*

$$\bar{\pi}_\theta(x) = p_{\eta^\star}(x), \qquad p_\eta(x) = \frac{1}{Z(\eta)} \frac{\pi_\theta(x)\,\pi_{\text{old}}(x)}{\pi_{\text{old}}(x) + \eta\,\pi_\theta(x)}, \qquad Z(\eta) = \sum_x \frac{\pi_\theta(x)\,\pi_{\text{old}}(x)}{\pi_{\text{old}}(x) + \eta\,\pi_\theta(x)}, \tag{7}$$

where $\eta^\star > 0$ *is the unique root of the strictly decreasing scalar function* $h(\eta) = \chi^2(p_\eta\|\pi_{\text{old}}) - \varepsilon$*.*

The scalar η is a dial: $\eta = 0$ returns the actor untouched, $\eta \to \infty$ returns π_{old}, and $\eta^\star$ is the single point between them where the budget is met exactly; finding it is one-dimensional root-finding on a monotone function.

Why does such a clean formula exist? Because $\chi^2(p\|q) = \sum_x p(x)^2/q(x) - 1$ is used for both the objective and the constraint, the Lagrangian separates into independent one-variable quadratics with no cross terms. Each coordinate's optimality condition is therefore linear in $p(x)$, while normalization is absorbed into $Z(\eta)$. This reduces the projection to one-dimensional root-finding and yields the closed-form solution above (Appendix F).

Taking reciprocals gives $1/\bar{\pi}_\theta(x) \propto 1/\pi_\theta(x) + \eta^\star/\pi_{\text{old}}(x)$: the output is a weighted *harmonic* mean of the actor output and the old policy. Dividing by π_{old} gives the ratio transform $\bar{r}(x) = \bar{\pi}_\theta(x)/\pi_{\text{old}}(x) = r(x)/\big(Z(\eta^\star)(1 + \eta^\star r(x))\big)$. This transform is smooth, strictly increasing, and bounded. Because it is strictly increasing, the projection preserves the ordering of the importance ratios. Small ratios pass nearly untouched; large ratios are damped, more so the larger they are; but the map never flattens, so no token's gradient ever dies.

4.2 SOLVING, TRAINING, DIFFERENTIATING, AND SCALING

The scalar solve. Computing the projected distribution reduces to finding the scalar $\eta^\star$ in Theorem 2 and substituting it into the closed-form solution. We first check whether the actor output is already inside the χ^2 ball. If it is, the projection returns the actor output unchanged. Otherwise, $h(0) > 0$, h is strictly decreasing (Lemma 3), and $h(\eta) \to -\varepsilon$ as $\eta \to \infty$, so h has exactly one root. We find this root by increasing an upper bound until h becomes negative and then applying bisection. Substituting the resulting $\eta^\star$ into Equation (7) gives the projected distribution (Algorithm 1, Appendix A).

Objective. The objective has two terms. The first performs the RL update using the projected distribution $\bar{\pi}_\theta$, rather than the raw actor distribution π_θ, in the importance ratio. Since $\bar{\pi}_\theta$ lies inside the χ^2 ball, the distribution used by the update always satisfies the constraint. The second term trains the raw actor to match the projected distribution (Otto et al., 2021):

$$\mathcal{J}_{\text{CHIP}}(\theta) = \mathbb{E}\left[\frac{1}{|o|}\sum_t \left(\frac{\bar{\pi}_\theta(o_t \mid s_t)}{\pi_{\text{old}}(o_t \mid s_t)}\hat{A}_t - \alpha\,\mathrm{KL}\Big(\pi_\theta(\cdot \mid s_t) \,\Big\|\, \lfloor \bar{\pi}_\theta(\cdot \mid s_t) \rfloor\Big)\right)\right], \tag{8}$$

Here, $\lfloor \cdot \rfloor$ denotes stop-gradient, so the KL term updates the raw actor but treats the projected distribution as a fixed target. This term is needed because the projection requires π_{old}, which is available during training but not during rollout or deployment. At those times, the model uses the raw actor, so training it to match the projected distribution carries the effect of the constraint into the model itself. If the raw actor is already inside the χ^2 ball, the projection leaves it unchanged and the KL term is zero. The objective does not depend on how $\hat{A}_t$ is computed, so CHIP can replace clipping without changing the advantage estimator.

Gradients. On feasible positions the layer is the identity and gradients pass through. On projected positions $\bar{\pi}_\theta$ depends on π_θ twice: explicitly through (7), and implicitly because moving π_θ moves the root $\eta^\star$. Since $\eta^\star$ solves a single scalar equation, the implicit function theorem gives its derivative as a ratio of two ordinary partial derivatives. The following proposition gives the gradient formula that backpropagation needs (the vector–Jacobian product) in closed form.

Proposition 3 (Closed-form backward pass)**.** *At a projected position with $\eta^\star > 0$, hold π_{old} fixed and let $Z = Z(\eta^\star)$, $d_x = \pi_{\text{old}}(x) + \eta^\star \pi_\theta(x)$, $b_x = \pi_\theta(x)^2 \pi_{\text{old}}(x)/d_x^2$, $B = \sum_x b_x$, $c_x = \pi_\theta(x)^3 \pi_{\text{old}}(x)/d_x^3$, and $C = \sum_x c_x$. Given an upstream gradient $u = \partial \mathcal{L}/\partial \bar{\pi}_\theta$, the gradient through the projection layer with respect to its actor input is*

$$\frac{\partial \mathcal{L}}{\partial \pi_\theta(x)} = \big(u(x) - \langle u, \bar{\pi}_\theta \rangle\big) \frac{\pi_{\text{old}}(x)^2}{d_x^2\, Z} \;-\; \frac{s}{\partial h/\partial \eta} \left(\frac{2\pi_\theta(x)\pi_{\text{old}}(x)^2}{d_x^3 Z^2} - \frac{2B\,\pi_{\text{old}}(x)^2}{d_x^2 Z^3} \right), \quad (9)$$

with $s = \big(\langle u, \bar{\pi}_\theta \rangle B - \langle u, b \rangle\big)/Z$ and $\partial h/\partial \eta = -2C/Z^2 + 2B^2/Z^3 < 0$ (Lemma 3), evaluated at $\eta^\star$. All quantities are sums and elementwise operations over the represented support; the backward pass costs $O(k)$ per position and materializes no matrices.

The first term in (9) differentiates the projected distribution while holding $\eta^\star$ fixed; the second accounts for how $\eta^\star$ changes to keep the constraint satisfied. The proof is in Appendix G. Generic differentiable-optimization layers instead differentiate a full KKT system (Amos and Kolter, 2017).

Scaling to LLM vocabularies. Storing a full next-token distribution at every sampled position is prohibitive for vocabularies above 10^5 tokens. We therefore sparsify the current actor π_θ and the old policy π_{old} separately. For each distribution, we take its own top-K tokens, retain the shortest probability-ranked prefix whose cumulative mass reaches $1 - \delta$, and always include the sampled token so that its gradient is available. All remaining tokens share a small default probability $p_d > 0$, and the retained probabilities are rescaled so that each distribution sums to one. Since the two models may retain different tokens, the projection uses the union of their retained sets, allowing both the forward and backward passes to operate on only $O(k)$ entries. Theorem 3 (Appendix H) shows that this sparse computation differs from the full-vocabulary χ^2 divergence only by a bounded tail term.

5 Related work

TRPO. TRPO (Schulman et al., 2015) controls the approximation error in a policy update through constrained optimization. It maximizes the exact objective while constraining the KL divergence between the new and old policies, so that the exact objective remains close to the true improvement (Kakade and Langford, 2002). However, solving this constrained problem requires second-order information, which is expensive at the scale of LLM training.

PPO. PPO (Schulman et al., 2017) avoids this cost by replacing the constrained optimization with a clipped importance-ratio objective. Clipping is cheaper, but it has two limitations. First, it discards information discontinuously: when a sampled token's ratio crosses the clipping boundary in the direction favored by its advantage, its gradient becomes zero. Second, clipping acts only on the sampled token's ratio rather than the full next-token distribution, so the probabilities of other tokens can still change substantially (Wang et al., 2020).

Recent variants. Later methods adjust the clipping interval (Wang et al., 2019; Xi et al., 2025), apply clipping to a sequence-level ratio (Zheng et al., 2025), or add stabilizers around the clipped objective (Yu et al., 2025). Because these methods still rely on clipping, they retain its discontinuous removal of gradient information and do not directly constrain the full next-token distribution. More

fundamentally, TRPO, PPO, and these newer variants all control only layer 1—the gap between the true improvement and the exact objective—and none controls layer 2, the finite-batch sampling error.

6 EXPERIMENTS

Setup. We evaluate CHIP across model scales and policy-optimization algorithms on mathematical reasoning, and on a multi-turn agentic software-engineering task. Every comparison is a matched two-arm A/B test: within each pair, the two runs use the same model, data, estimator, optimizer, learning rate, batch construction, and training schedule, and differ only in whether the policy update uses clipping or CHIP. Experiment-specific model, training, and evaluation settings are given in the corresponding subsection.

6.1 ACROSS MODEL SCALES

We train Qwen3 models (Yang et al., 2025) at 0.6B, 1.7B, 4B, 8B, 14B, and 32B for 300 steps on DAPO-Math-17k (Yu et al., 2025). Each step contains 256 prompts with eight responses per prompt, divided into eight minibatches and trained for one epoch with a constant learning rate of 10^{-6} (Table 4). The GRPO control uses a clip range of 0.2 and a KL penalty to the reference model with coefficient 10^{-3}. CHIP uses $\varepsilon{=}0.05$ from 0.6B to 8B and $\varepsilon{=}0.25$ at 14B and 32B. We evaluate every ten steps using greedy decoding on the DAPO validation set and seven held-out benchmarks.

Table 1 compares CHIP and GRPO across six Qwen3 model sizes on three measures: training success on DAPO-Math-17k, average greedy accuracy on eight validation benchmarks, and greedy accuracy on held-out AIME 2025. At the last checkpoint reached by both runs, **CHIP scores higher in seventeen of the eighteen model-size comparisons** and ties the remaining one, AIME 2025 at 1.7B. On the eight-benchmark validation average, CHIP is higher at $27/30$, $30/30$, $27/27$, $30/30$, $28/30$, and $30/30$ synchronized evaluations from 0.6B to 32B, respectively. Averaged across the six model sizes, CHIP gains 6.9 validation points over the final sixty steps.

Figure 1 shows that the clearest difference appears late in training. From 0.6B to 14B, GRPO reaches its highest validation score between steps 60 and 250, then ends 3.5–27.2 points below that peak. At 1.7B, 4B, 8B, and 14B, CHIP instead finishes at its peak or within 2.7 points of it. The exception is 0.6B: CHIP and GRPO fall by similar amounts from their respective peaks (3.8 versus 3.7 points), but CHIP remains higher at 27 of the 30 synchronized evaluations and at the end. At 32B, GRPO remains stable and CHIP has a smaller average lead of 1.2 points over the final sixty steps.

Appendix I.1 examines why the two methods behave differently. In the GRPO runs, clipping activates for only 0.16–0.29% of sampled tokens, whereas CHIP's χ^2 projection activates at 1.9–20.3% of sampled positions. These measurements indicate that the full next-token distribution often moves outside the χ^2 constraint even when the probability ratio of the sampled token remains inside the clipping interval. Clipping misses these changes because it checks only the sampled token, whereas CHIP checks the entire represented distribution at that position. The appendix also reports the χ^2 divergence before and after projection and policy entropy.

6.2 ACROSS POLICY-OPTIMIZATION ALGORITHMS

We compare GRPO, Dr. GRPO, PPO, GSPO, and DAPO on Qwen3-8B trained on DAPO-Math-17k, using the evaluation suite in Section 6.1. For each algorithm, we keep its original training settings and replace only clipping with CHIP. We use $\varepsilon{=}0.05$ for all five algorithms and additionally test $\varepsilon{=}0.25$ for GSPO and DAPO, giving seven paired comparisons. Table 2 reports results at training step 150.

CHIP improves DAPO validation accuracy in all seven comparisons, with gains of 0.7–10.7 percentage points. The largest gains are for GRPO ($+10.7$) and PPO ($+7.5$). Training success is also higher in all seven comparisons. On held-out AIME, CHIP improves six of seven; the exception is GSPO with $\varepsilon{=}0.25$, where CHIP scores 39.7%, compared with 40.4% for the clipped baseline.

		0.6B	1.7B	4B	8B	14B	32B
DAPO Train	GRPO	0.283	0.419	0.323	0.625	0.664	0.714
	CHIP	0.344 (+6.0±2.9)	0.535 (+11.6±3.0)	0.675 (+35.2±10.2)	0.716 (+9.1±3.1)	0.750 (+8.6±1.2)	0.738 (+2.4±2.9)
DAPO Eval	GRPO	0.268	0.380	0.269	0.518	0.559	0.597
	CHIP	0.284 (+1.6±1.8)	0.451 (+7.1±1.2)	0.570 (+30.1±7.7)	0.582 (+6.3±1.0)	0.624 (+6.5±0.8)	0.602 (+0.5±0.7)
AIME'25 Eval	GRPO	0.067	0.200	0.067	0.294	0.304	0.296
	CHIP	0.100 (+3.3±6.0)	0.200 (+0.0±6.2)	0.325 (+25.8±4.9)	0.372 (+7.8±7.7)	0.400 (+9.6±5.2)	0.358 (+6.2±2.2)

Table 1: CHIP versus GRPO across six Qwen3 model sizes. Results are measured at the last checkpoint reached by both runs. DAPO Train reports the median training success rate over the last twenty training steps; DAPO Eval is greedy accuracy averaged over eight validation benchmarks; and AIME'25 Eval is greedy accuracy on its thirty problems. Each CHIP cell reports its score, followed in parentheses by its gain over GRPO and the standard deviation of that gain across checkpoints over the final sixty steps. Shading marks the higher score. CHIP uses ε=0.05 at 0.6B–8B and ε=0.25 at 14B–32B.

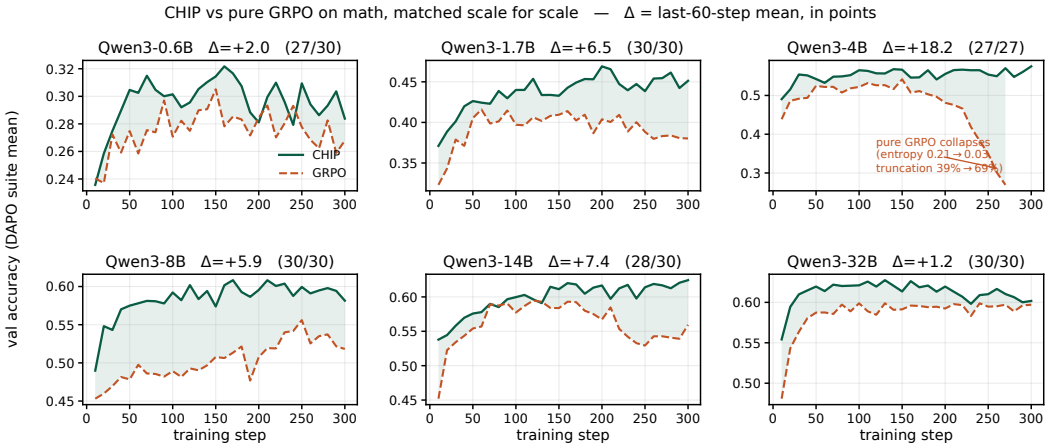

Figure 1: CHIP versus GRPO across six Qwen3 model sizes on the DAPO validation-suite mean. Each panel title reports the average CHIP–GRPO difference over the final sixty steps and the number of synchronized evaluations won by CHIP; the shaded area shows the difference between the two curves.

		GRPO	Dr.GRPO	PPO	GSPO$_{.05}$	GSPO$_{.25}$	DAPO$_{.05}$	DAPO$_{.25}$
DAPO Train	Baseline	0.623	0.680	0.630	0.683	0.683	0.677	0.699
	CHIP	0.718 (+9.5)	0.686 (+0.6)	0.718 (+8.8)	0.709 (+2.6)	0.707 (+2.4)	0.743 (+6.6)	0.747 (+4.8)
DAPO Eval	Baseline	0.585	0.645	0.623	0.674	0.674	0.653	0.653
	CHIP	0.692 (+10.7)	0.659 (+1.4)	0.698 (+7.5)	0.681 (+0.7)	0.683 (+0.9)	0.691 (+3.8)	0.720 (+6.7)
AIME Eval	Baseline	0.289	0.397	0.334	0.404	0.404	0.325	0.325
	CHIP	0.398 (+10.9)	0.456 (+5.8)	0.476 (+14.2)	0.421 (+1.7)	0.397 (-0.7)	0.376 (+5.1)	0.441 (+11.6)

Table 2: CHIP versus clipped baselines for five policy-optimization algorithms on Qwen3-8B. All results are measured at training step 150. Subscripts indicate CHIP's χ^2 budget. DAPO Train reports training success rate; DAPO Eval reports validation accuracy; AIME Eval reports accuracy on AIME 2024 and 2025, using 32 responses per problem. Parentheses show CHIP's gain over the baseline in percentage points, and shading marks the higher score.

6.3 Multi-turn agentic software engineering

We evaluate CHIP on multi-turn agentic software-engineering tasks using Qwen3-32B trained on R2E-Gym (Jain et al., 2025). CLIP is an asynchronous clipped RL implementation adapted from the DeepSWE recipe, while the published DeepSWE recipe uses synchronous RL (Luo et al., 2025).

Both methods use the same training setup and rollout schedule, and neither trains a critic. CHIP replaces CLIP's sampled-token clipping objective with its projection-based objective using $\varepsilon = 0.25$. We evaluate pass@1 on all 500 tasks in SWE-bench Verified at temperature 1, reporting formal checkpoints every ten training steps from step 0 to step 130.

Figure 2 shows that CHIP and CLIP start at the same 20.8% score. CHIP is higher at 10 of the 14 evaluated checkpoints and ties CLIP at steps 0 and 30. From step 40 onward, CHIP remains ahead at every checkpoint. At step 130, CHIP achieves 45.8%, compared with 39.0% for CLIP, a gain of 6.8 percentage points. The dotted line marks the published 42.2% on-policy result reported for DeepSWE (Luo et al., 2025). At step 130, CHIP exceeds this reference by 3.6 percentage points, while CLIP remains below it. Detailed settings and the evaluation protocol are provided in Appendix I.3.

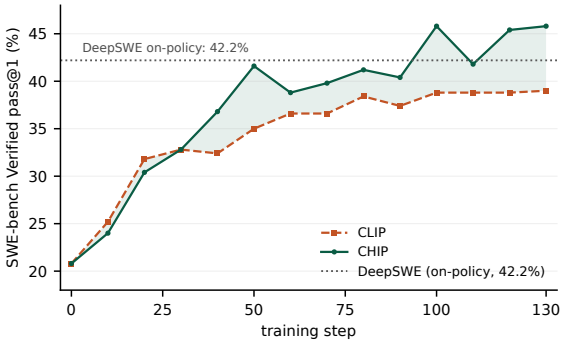

Figure 2: Pass@1 on 500 SWE-bench Verified tasks using Qwen3-32B at temperature 1. Points show evaluations at steps 0, 10, . . ., 130. Shading shows the difference between the two curves. The solid line is CHIP ($\varepsilon = 0.25$), the dashed line is CLIP, and the dotted line marks the published 42.2% on-policy result for DeepSWE (Luo et al., 2025).

6.4 Radius ablation

We compare CHIP with $\varepsilon \in \{0.05, 0.1, 0.25\}$ on Qwen3-8B, using the DAPO settings in Section 6.2. All other settings are fixed. Table 3 and Figure 3 show that $\varepsilon = 0.25$ achieves the highest validation accuracy at all six evaluations from steps 150 to 200, exceeding the clipped DAPO baseline by 2.9–6.6 percentage points.

Step	DAPO	CHIP radius ε		
		0.05	0.1	0.25
150	0.653	0.691	0.661	**0.720**
160	0.660	0.706	0.671	**0.715**
170	0.670	0.689	0.678	**0.705**
180	0.673	0.712	0.698	**0.713**
190	0.674	0.689	0.690	**0.703**
200	0.664	0.684	0.698	**0.725**

Table 3: DAPO validation accuracy on Qwen3-8B at steps 150–200. Bold marks the highest score at each step.

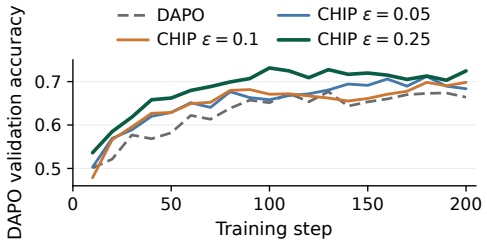

Figure 3: Validation curves from steps 10 to 200, without smoothing. The dashed line is the clipped DAPO baseline.

Additional diagnostics and limitations are reported in Appendices I.1 and J.

7 Conclusion

An RL update has two errors: off-policy bias from scoring the new policy on old states, and mini-batch variance. The χ^2 divergence between successive policies controls both; CHIP enforces this budget with a closed-form projection that replaces clipping. Across the tested algorithms, model sizes, and multi-turn coding task, CHIP improves validation performance and reduces late-training degradation.

AI USE STATEMENT

We used generative AI tools only for coding assistance and language revision. The authors wrote the manuscript and used AI to improve its wording and clarity. The authors take full responsibility for the code and the final paper.

ETHICS STATEMENT

This work studies reinforcement learning for mathematical reasoning and coding tasks. Improved coding capabilities can support software development but may also be misused. Deployment should include safeguards for code execution and access to external systems.

REPRODUCIBILITY STATEMENT

We implement CHIP in the open-source verl framework (Sheng et al., 2025). Appendix A provides the training algorithm, and Appendices B–H provide the proofs. Section 6 and Appendix I describe the datasets, training settings, and evaluation procedures. All experimental results are computed from training logs.

REFERENCES

Sergios Agapiou, Omiros Papaspiliopoulos, Daniel Sanz-Alonso, and Andrew M. Stuart. Importance sampling: Intrinsic dimension and computational cost. *Statistical Science*, 32(3):405–431, 2017. doi: 10.1214/17-STS611.

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting REINFORCE-style optimization for learning from human feedback in LLMs. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 12248–12267, 2024. doi: 10.18653/v1/2024.acl-long.662.

Brandon Amos and J. Zico Kolter. OptNet: Differentiable optimization as a layer in neural networks. In *International Conference on Machine Learning (ICML)*, 2017.

Philipp Becker, Niklas Freymuth, Serge Thilges, Fabian Otto, and Gerhard Neumann. TROLL: Trust regions improve reinforcement learning for large language models. In *International Conference on Learning Representations (ICLR)*, 2026.

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

Ganqu Cui, Yuchen Zhang, Jiacheng Chen, Lifan Yuan, Zhi Wang, Yuxin Zuo, Haozhan Li, Yuchen Fan, Huayu Chen, Weize Chen, Zhiyuan Liu, Hao Peng, Lei Bai, Wanli Ouyang, Yu Cheng, Bowen Zhou, and Ning Ding. The entropy mechanism of reinforcement learning for reasoning language models. *arXiv preprint arXiv:2505.22617*, 2025a.

Ganqu Cui et al. Process reinforcement through implicit rewards. *arXiv preprint arXiv:2502.01456*, 2025b.

Asen L. Dontchev and R. Tyrrell Rockafellar. *Implicit Functions and Solution Mappings: A View from Variational Analysis*. Springer, 2009. doi: 10.1007/978-0-387-87821-8.

Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. Implementation matters in deep RL: A case study on PPO and TRPO. In *International Conference on Learning Representations (ICLR)*, 2020.

Bofei Gao et al. Omni-MATH: A universal olympiad level mathematic benchmark for large language models. In *International Conference on Learning Representations (ICLR)*, 2025.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. OlympiadBench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 3828–3850, 2024. doi: 10.18653/v1/2024.acl-long.211.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *NeurIPS Datasets and Benchmarks*, 2021.

Jian Hu. REINFORCE++: A simple and efficient approach for aligning large language models. *arXiv preprint arXiv:2501.03262v1*, 2025.

Audrey Huang, Wenhao Zhan, Tengyang Xie, Jason D. Lee, Wen Sun, Akshay Krishnamurthy, and Dylan J. Foster. Correcting the mythos of KL-regularization: Direct alignment without overoptimization via chi-squared preference optimization. *arXiv preprint arXiv:2407.13399*, 2024.

Naman Jain, Jaskirat Singh, Manish Shetty, Liang Zheng, Koushik Sen, and Ion Stoica. R2E-Gym: Procedural environments and hybrid verifiers for scaling open-weights SWE agents. *arXiv preprint arXiv:2504.07164*, 2025.

Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2002.

Augustine Kong. A note on importance sampling using standardized weights. Technical Report 348, Department of Statistics, University of Chicago, 1992. URL https://d3qi0qp55mx5f5.cloudfront.net/stat/docs/tech-rpts/tr348.pdf.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving quantitative reasoning problems with language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

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. In *International Conference on Learning Representations (ICLR)*, 2024.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding R1-Zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025.

Michael Luo, Naman Jain, Jaskirat Singh, Sijun Tan, Ameen Patel, Qingyang Wu, Alpay Ariyak, Colin Cai, Tarun Venkat, Shang Zhu, Ben Athiwaratkun, Manan Roongta, Ce Zhang, Li Erran Li, Raluca Ada Popa, Koushik Sen, and Ion Stoica. DeepSWE: Training a fully open-sourced, state-of-the-art coding agent by scaling RL. Together AI and Agentica blog post, July 2, 2025. URL https://www.together.ai/blog/deepswe.

Alberto Maria Metelli, Matteo Papini, Francesco Faccio, and Marcello Restelli. Policy optimization via importance sampling. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Fabian Otto, Philipp Becker, Ngo Anh Vien, Hanna Carolin Ziesche, and Gerhard Neumann. Differentiable trust region layers for deep reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2021.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Art B. Owen. *Monte Carlo Theory, Methods and Examples*. Online book, 2013. URL https://artowen.su.domains/mc/.

Igal Sason and Sergio Verdú. f-divergence inequalities. *IEEE Transactions on Information Theory*, 62(11):5973–6006, 2016. doi: 10.1109/TIT.2016.2603151.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning (ICML)*, 2015.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *International Conference on Learning Representations (ICLR)*, 2016.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. HybridFlow: A flexible and efficient RLHF framework. In *European Conference on Computer Systems (EuroSys)*, 2025. doi: 10.1145/3689031.3696075.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F. Christiano. Learning to summarize with human feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Richard S. Sutton, David A. McAllester, Satinder P. Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems (NIPS)*, volume 12, 1999.

Alexandre B. Tsybakov. *Introduction to Nonparametric Estimation*. Springer, 2009. doi: 10.1007/b13794.

Yuhui Wang, Hao He, Xiaoyang Tan, and Yaozhong Gan. Trust region-guided proximal policy optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Yuhui Wang, Hao He, and Xiaoyang Tan. Truly proximal policy optimization. In *Uncertainty in Artificial Intelligence (UAI)*, 2020.

Zhiheng Xi, Xin Guo, Yang Nan, Enyu Zhou, Junrui Shen, Wenxiang Chen, Jiaqi Liu, Jixuan Huang, Zhihao Zhang, Honglin Guo, Xun Deng, Zhikai Lei, Miao Zheng, Guoteng Wang, Shuo Zhang, Peng Sun, Rui Zheng, Hang Yan, Tao Gui, Qi Zhang, and Xuanjing Huang. BAPO: Stabilizing off-policy reinforcement learning for LLMs via balanced policy optimization with adaptive clipping. *arXiv preprint arXiv:2510.18927*, 2025.

An Yang et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Weinan Dai, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. DAPO: An open-source LLM reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476v1*, 2025.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, Jingren Zhou, and Junyang Lin. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*, 2025.

Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B. Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.

APPENDIX

A ALGORITHM

Algorithm 1 describes one rollout-and-update cycle. We first generate a batch with the old policy and keep that policy fixed while updating the actor. At each sampled position, CHIP checks the actor's next-token distribution. If it exceeds the χ^2 budget, CHIP projects it before computing the objective. Otherwise, the distribution is unchanged. Appendix F derives the projection, and Appendix G derives its gradient. The sparse representation used for large vocabularies is described in Appendix H. The projection takes positive, normalized distributions over the represented categories. Any omitted tokens with a shared default probability can be included as one aggregate tail category, so their probability mass is not discarded. The radius is $\varepsilon \geq 0$.

Algorithm 1 One CHIP training iteration

Copy the actor to $\pi_{\text{old}} \leftarrow \pi_\theta$; keep π_{old} fixed for this batch.
Sample prompts $q \sim \mathcal{D}$ and generate responses $o \sim \pi_{\text{old}}$.
Store the sparse old-policy distribution at each sampled position s_t.
Compute advantages $\hat{A}_t$ using the baseline's estimator and its original grouping.
for each minibatch **do**
  Compute the actor distributions $\pi_\theta(\cdot \mid s_t)$.
  Form the union of tokens retained separately by the current and old policies; include the sampled token and account for the remaining tail mass.
  for all positions t **in parallel do**
    if $\varepsilon = 0$ **then**
      Set $\bar{\pi}_\theta \leftarrow \pi_{\text{old}}$.
    else if $\chi^2(\pi_\theta \| \pi_{\text{old}}) \leq \varepsilon$ **then**
      Set $\bar{\pi}_\theta \leftarrow \pi_\theta$; no projection is needed.
    else
      Increase a positive upper bound until h is negative, then bisect to the chosen tolerance.
      Use the feasible endpoint of the final bracket as the numerical value of $\eta^\star$.
      Compute $\bar{\pi}_\theta$ from $\eta^\star$ using (7).
    end if
  end for
  Compute (8), treating π_{old} and $\hat{A}_t$ as constants.
  Differentiate the ratio term through $\bar{\pi}_\theta$ using Proposition 3; stop the gradient through $\bar{\pi}_\theta$ only in the KL target.
  Update the actor parameters to increase the objective.
end for

The following sections give the derivations used in the main text. Each section states the result to be shown, introduces the required notation, and explains the steps of the proof.

B PERFORMANCE-DIFFERENCE LEMMA

The performance-difference lemma expresses the difference between two policies' expected returns using the old policy's advantage function.

Definitions. Consider a discounted Markov decision process with discount factor $\gamma \in (0, 1)$, bounded rewards, and a common initial-state distribution for both policies. A trajectory $\tau =$

$(s_0, a_0, r_0, s_1, a_1, r_1, \dots)$ records the states, actions, and rewards generated by policy π. Define

$$\eta(\pi) = \mathbb{E}_{\tau\sim\pi}\Big[\sum_{t\geq 0}\gamma^t r_t\Big] \qquad \text{the expected discounted return,}$$

$$V_\pi(s) = \mathbb{E}_{\tau\sim\pi}\Big[\sum_{t\geq 0}\gamma^t r_t \,\Big|\, s_0 = s\Big] \qquad \text{the value of state } s,$$

$$Q_\pi(s,a) = \mathbb{E}\big[r_0 + \gamma V_\pi(s_1) \,\big|\, s_0 = s, a_0 = a\big] \qquad \text{the value of action } a \text{ followed by policy } \pi.$$

The advantage $A_\pi(s,a) = Q_\pi(s,a) - V_\pi(s)$ measures how taking action a compares with sampling an action from π at the same state. Its mean under π is zero: $\mathbb{E}_{a\sim\pi(\cdot|s)}[A_\pi(s,a)] = 0$.

Lemma 2. *For any two policies π, π_{old} under these assumptions,*

$$\eta(\pi) - \eta(\pi_{\text{old}}) = \mathbb{E}_{\tau\sim\pi}\Big[\sum_{t\geq 0}\gamma^t A_{\pi_{\text{old}}}(s_t, a_t)\Big].$$

The trajectories in this expectation come from the new policy π, but each action is evaluated using $A_{\pi_{\text{old}}}$. The expected discounted sum of these advantages equals the return difference exactly.

Proof. Step 1: rewrite the advantage. By the definition of Q above, for any state–action pair (s_t, a_t),

$$A_{\pi_{\text{old}}}(s_t,a_t) = Q_{\pi_{\text{old}}}(s_t,a_t) - V_{\pi_{\text{old}}}(s_t) = \mathbb{E}\big[r_t + \gamma V_{\pi_{\text{old}}}(s_{t+1}) - V_{\pi_{\text{old}}}(s_t) \,\big|\, s_t, a_t\big].$$

Thus the advantage is the expected reward plus the discounted value of the next state, minus the value of the current state. Both values use π_{old}, regardless of which policy selected a_t.

Step 2: sum the discounted advantages. Multiply by γ^t and sum over t. Taking expectations over trajectories from π and using conditional expectation gives the equality below. Bounded rewards allow us to exchange the sum and expectation:

$$\mathbb{E}_{\tau\sim\pi}\left[\sum_{t\geq 0}\gamma^t A_{\pi_{\text{old}}}(s_t,a_t)\right] = \mathbb{E}_{\tau\sim\pi}\left[\sum_{t\geq 0}\gamma^t r_t\right] + \mathbb{E}_{\tau\sim\pi}\left[\sum_{t\geq 0}\big(\gamma^{t+1}V_{\pi_{\text{old}}}(s_{t+1}) - \gamma^t V_{\pi_{\text{old}}}(s_t)\big)\right].$$

Step 3: cancel the value terms. The second sum begins with

$$\big(\gamma V(s_1) - V(s_0)\big) + \big(\gamma^2 V(s_2) - \gamma V(s_1)\big) + \big(\gamma^3 V(s_3) - \gamma^2 V(s_2)\big) + \cdots$$

(where $V = V_{\pi_{\text{old}}}$). The intermediate value terms cancel. Summing through $t = T-1$ leaves $\gamma^T V(s_T) - V(s_0)$. Since V is bounded and $\gamma < 1$, the first term tends to zero as $T \to \infty$. The remaining value term is $-V_{\pi_{\text{old}}}(s_0)$. Thus

$$\mathbb{E}_{\tau\sim\pi}\left[\sum_{t\geq 0}\gamma^t A_{\pi_{\text{old}}}(s_t,a_t)\right] = \mathbb{E}_{\tau\sim\pi}\left[\sum_{t\geq 0}\gamma^t r_t - V_{\pi_{\text{old}}}(s_0)\right].$$

Step 4: identify the two expected returns. Under trajectories from π, the expected reward sum is $\mathbb{E}_{\tau\sim\pi}[\sum_t \gamma^t r_t] = \eta(\pi)$. Both policies have the same initial-state distribution, so $\mathbb{E}[V_{\pi_{\text{old}}}(s_0)] = \eta(\pi_{\text{old}})$. Substituting these two identities gives

$$\mathbb{E}_{\tau\sim\pi}\Big[\sum_{t\geq 0}\gamma^t A_{\pi_{\text{old}}}(s_t,a_t)\Big] = \eta(\pi) - \eta(\pi_{\text{old}}). \qquad \square$$

State visitation and normalization. Define the discounted visitation weights $\rho_\pi(s) = \sum_{t\geq 0}\gamma^t \Pr_\pi(s_t = s)$. They sum to $1/(1-\gamma)$, not one. The normalized distribution is $d_\pi(s) = (1-\gamma)\rho_\pi(s)$. Grouping the trajectory sum by state gives

$$\eta(\pi) - \eta(\pi_{\text{old}}) = \sum_s \rho_\pi(s)\sum_a \pi(a \mid s) A_{\pi_{\text{old}}}(s,a) = \frac{1}{1-\gamma}\mathbb{E}_{s\sim d_\pi,\, a\sim\pi}[A_{\pi_{\text{old}}}(s,a)].$$

Thus the state-visitation form of the lemma requires either an unnormalized sum or a normalized expectation multiplied by $1/(1-\gamma)$. The factor must be kept consistent with the definition of return.

From the lemma to the surrogate. Replace ρ_π by $\rho_{\pi_{\text{old}}}$, then use the ratio $r(s,a) = \pi(a \mid s)/\pi_{\text{old}}(a \mid s)$ to express the action sum under the old policy. Assuming the old policy assigns positive probability to every action used by π, the resulting surrogate for the ordinary discounted return is

$$\sum_s \rho_{\pi_{\text{old}}}(s) \sum_a \pi(a \mid s) A_{\pi_{\text{old}}}(s,a) = \frac{1}{1-\gamma} \mathbb{E}_{s\sim d_{\pi_{\text{old}}},\, a\sim\pi_{\text{old}}}[r(s,a) A_{\pi_{\text{old}}}(s,a)].$$

The TRPO bound controls the error from replacing the state-visitation weights. A sample mean drawn from normalized $d_{\pi_{\text{old}}}$ estimates the expectation on the right; multiplying by $1/(1-\gamma)$ gives the surrogate on the ordinary-return scale. This distinction is needed when relating (1), (2), and (3).

C Weight identities and the KL blind spot

We first show that the variance of the importance weight equals the χ^2 divergence, and then use this identity to bound the variance of the sampled objective. These are standard importance-sampling results (Agapiou et al., 2017; Metelli et al., 2018). We then construct distributions with bounded KL divergence but arbitrarily large weight variance.

The moment calculations below hold for fixed policies at one state, with $\pi_{\text{old}}(x) > 0$. The minibatch variance formula additionally assumes independent, identically distributed samples and bounded advantages.

Proof of Lemma 1. First moment. By definition of expectation under π_{old},

$$\mathbb{E}_{\pi_{\text{old}}}[r] = \sum_x \pi_{\text{old}}(x)\, r(x) = \sum_x \pi_{\text{old}}(x)\, \frac{\pi(x)}{\pi_{\text{old}}(x)} = \sum_x \pi(x) = 1.$$

The sampling probability $\pi_{\text{old}}(x)$ cancels the denominator of the weight. The remaining probabilities sum to one, so the mean weight is always 1.

Second moment. Repeat with r^2:

$$\mathbb{E}_{\pi_{\text{old}}}[r^2] = \sum_x \pi_{\text{old}}(x)\, \frac{\pi(x)^2}{\pi_{\text{old}}(x)^2} = \sum_x \frac{\pi(x)^2}{\pi_{\text{old}}(x)}.$$

Each term in the second moment is divided by $\pi_{\text{old}}(x)$. A token can therefore contribute strongly if its old probability is small relative to its new probability. Using $\text{Var}(X) = \mathbb{E}[X^2] - (\mathbb{E}[X])^2$ and $\mathbb{E}_{\pi_{\text{old}}}[r] = 1$,

$$\text{Var}_{\pi_{\text{old}}}(r) = \mathbb{E}_{\pi_{\text{old}}}[r^2] - 1 = \sum_x \frac{\pi(x)^2}{\pi_{\text{old}}(x)} - 1.$$

Relation to χ^2. Expand the square in the definition $\chi^2(\pi\|\pi_{\text{old}}) = \sum_x (\pi(x) - \pi_{\text{old}}(x))^2/\pi_{\text{old}}(x)$:

$$\begin{aligned}\sum_x \frac{(\pi(x) - \pi_{\text{old}}(x))^2}{\pi_{\text{old}}(x)} &= \sum_x \frac{\pi(x)^2 - 2\pi(x)\pi_{\text{old}}(x) + \pi_{\text{old}}(x)^2}{\pi_{\text{old}}(x)} \\ &= \sum_x \frac{\pi(x)^2}{\pi_{\text{old}}(x)} - 2\sum_x \pi(x) + \sum_x \pi_{\text{old}}(x).\end{aligned}$$

Both distributions sum to one. The final two terms therefore add to -1, giving $\sum_x \pi(x)^2/\pi_{\text{old}}(x) - 1$. This is exactly the weight variance derived above.

Variance of the weighted advantage. Since $\text{Var}(X) = \mathbb{E}[X^2] - (\mathbb{E}X)^2 \le \mathbb{E}[X^2]$, we can bound the variance using the second moment. Apply this to $X = r\hat{A}$ with $|\hat{A}| \le \epsilon_A$:

$$\text{Var}_{\pi_{\text{old}}}(r\hat{A}) \le \mathbb{E}_{\pi_{\text{old}}}[r^2 \hat{A}^2] \le \epsilon_A^2\, \mathbb{E}_{\pi_{\text{old}}}[r^2] = \epsilon_A^2 \big(1 + \chi^2(\pi\|\pi_{\text{old}})\big).$$

For an i.i.d. minibatch, independence gives

$$\text{Var}(\hat{L}) = \text{Var}\left(\frac{1}{N}\sum_{i=1}^N r_i \hat{A}_i\right) = \frac{1}{N^2}\sum_{i=1}^N \text{Var}(r_i \hat{A}_i) = \frac{1}{N}\text{Var}_{\pi_{\text{old}}}(r\hat{A}).$$

Combining this identity with the preceding bound gives the claimed result. □

Scope of the minibatch bound. If states are also sampled from a fixed distribution μ, the corresponding identity is

$$\mathbb{E}_{s\sim\mu,\, a\sim\pi_{\mathrm{old}}}[r(s,a)^2] = 1 + \mathbb{E}_{s\sim\mu}\big[\chi^2\big(\pi(\cdot \mid s)\|\pi_{\mathrm{old}}(\cdot \mid s)\big)\big]\,.$$

The variance bound then uses this state-averaged divergence. A budget at every state in the support of μ bounds the average by ε. Without independence, $\mathrm{Var}(\hat{L})$ also contains covariance terms $N^{-2}\sum_{i\neq j}\mathrm{Cov}(r_i\hat{A}_i, r_j\hat{A}_j)$. Tokens from one response and advantages computed from a shared group need not be independent. The fixed-policy calculation also does not directly apply to a policy selected using the same batch. Finally, unbiased estimation of an advantage-based surrogate requires exact advantages or $\mathbb{E}[\hat{A} \mid s,a] = A_{\pi_{\mathrm{old}}}(s,a)$; importance weighting alone does not ensure this condition.

Proof of Proposition 2. For any KL budget $\varepsilon > 0$ and target variance $M > 0$, we construct two distributions whose KL divergence is at most ε but whose χ^2 divergence is at least M. We increase the probability of one rare token by a large ratio. Its KL contribution stays small, but its χ^2 contribution grows without bound.

Step 1: define the distributions. Let $R \geq 2$ be a ratio that we will increase as needed. For a rare token $x^\star$, set

$$m = \frac{\varepsilon}{\ln R}, \qquad \pi_{\mathrm{old}}(x^\star) = \frac{m}{R}, \qquad \pi(x^\star) = m.$$

The probability of $x^\star$ increases by a factor of R. To keep the total probability equal to one, subtract the added probability $\Delta = m - \frac{m}{R} = m\big(1-\frac{1}{R}\big)$ from another token x_0, where $\pi_{\mathrm{old}}(x_0) = c \geq \frac{1}{2}$. Thus $\pi(x_0) = \pi_{\mathrm{old}}(x_0) - \Delta$, and all other probabilities stay unchanged. A two-token example takes $c = 1 - m/R$, so that $\pi(x_0) = 1 - m$. For sufficiently large R, $m = \varepsilon/\ln R \leq \frac{1}{2}$, so $\Delta \leq m \leq \frac{1}{2} \leq c$. Both distributions are therefore valid.

Step 2: bound the KL divergence. Only $x^\star$ and x_0 contribute, because all other probabilities are unchanged:

$$\mathrm{KL}(\pi\|\pi_{\mathrm{old}}) = \underbrace{\pi(x^\star)\ln\frac{\pi(x^\star)}{\pi_{\mathrm{old}}(x^\star)}}_{=\,m\ln R} + \underbrace{(c-\Delta)\ln\frac{c-\Delta}{c}}_{\leq\,0} \;\leq\; m\ln R \;=\; \frac{\varepsilon}{\ln R}\cdot\ln R \;=\; \varepsilon.$$

The second term is nonpositive because $(c-\Delta)/c \leq 1$. The rare token contributes ε, so the total KL divergence is at most ε.

Step 3: lower-bound the χ^2 divergence. Each term is nonnegative, so the total is at least the contribution of $x^\star$:

$$\chi^2(\pi\|\pi_{\mathrm{old}}) \;\geq\; \frac{\big(\pi(x^\star)-\pi_{\mathrm{old}}(x^\star)\big)^2}{\pi_{\mathrm{old}}(x^\star)} = \frac{\big(m-\frac{m}{R}\big)^2}{m/R} = mR\Big(1-\tfrac{1}{R}\Big)^2 \;\geq\; \frac{mR}{4} = \frac{\varepsilon\, R}{4\ln R},$$

Here $(1-\frac{1}{R})^2 \geq \frac{1}{4}$ because $R \geq 2$. The lower bound $\varepsilon R/(4\ln R)$ tends to infinity with R. We can therefore choose R so that $\varepsilon R/(4\ln R) \geq M$, while the KL divergence remains at most ε.

The χ^2 bound. In contrast, if $\chi^2(\pi\|\pi_{\mathrm{old}}) \leq \varepsilon$, then $\mathrm{Var}_{\pi_{\mathrm{old}}}(r) \leq \varepsilon$ by the identity (4). □

The same construction also gives arbitrarily large variance for a bounded weighted score, not just for the weight itself. Take $f(x) = \mathbf{1}\{x = x^\star\}$. Then $\mathbb{E}_{\pi_{\mathrm{old}}}[rf] = m$ and $\mathbb{E}_{\pi_{\mathrm{old}}}[(rf)^2] = mR$, so $\mathrm{Var}_{\pi_{\mathrm{old}}}(rf) = mR - m^2 \to \infty$ as $R \to \infty$.

Numerical example. The example in the main text uses $\pi_{\mathrm{old}}(x^\star) = 10^{-6}$ and $\pi(x^\star) = 10^{-3}$, giving $R = 10^3$:

- KL contribution: $\pi(x^\star)\ln R = 10^{-3}\ln 10^3 \approx 6.9\times10^{-3}$, which is 14% of $\varepsilon = 0.05$.
- χ^2 contribution: $(10^{-3}-10^{-6})^2/10^{-6} \approx (10^{-3})^2/10^{-6} = 1$, about twenty times the $\varepsilon = 0.05$ budget.

- Weight variance: the token contributes $\pi_{\text{old}}(x^\star)\, r(x^\star)^2 = 10^{-6} \times (10^3)^2 = 1$ to the second moment. With identical policies, the total is $\mathbb{E}[r^2] = 1$. The update approximately doubles this total and increases $\mathrm{Var}(r)$ from 0 to ≈ 1; the other tokens' contributions change by only $O(10^{-3})$.
- Sampling frequency: in a batch of 10^5 old-policy samples, the expected number of appearances of $x^\star$ is $10^5 \times 10^{-6} = 0.1$. Most batches therefore contain no occurrence of this token. When it does appear, its weight is 10^3, equal to the combined weight of one thousand samples whose weights are 1.

Separately, χPO uses χ^2 regularization for offline preference alignment (Huang et al., 2024), rather than a per-update projection between successive policies.

D From χ^2 to total variation

This section shows that bounding χ^2 also bounds total variation. We then distinguish a bound at sampled states from a bound at all states.

Proof of Proposition 1. Write total variation as a sum, multiplying and dividing each term by $\sqrt{q(x)}$:

$$2\, D_{\text{TV}}(p, q) = \sum_x \big|p(x) - q(x)\big| = \sum_x \frac{\big|p(x) - q(x)\big|}{\sqrt{q(x)}} \cdot \sqrt{q(x)}.$$

Apply the Cauchy–Schwarz inequality, $\sum_x a_x b_x \le \sqrt{\sum_x a_x^2}\,\sqrt{\sum_x b_x^2}$, using $a_x = |p(x) - q(x)|/\sqrt{q(x)}$ and $b_x = \sqrt{q(x)}$:

$$2\, D_{\text{TV}}(p, q) \;\le\; \sqrt{\sum_x \frac{\big(p(x) - q(x)\big)^2}{q(x)}} \;\cdot\; \sqrt{\sum_x q(x)} \;=\; \sqrt{\chi^2(p \,\|\, q)} \cdot 1,$$

The first sum is $\chi^2(p\|q)$, and the second is 1 because q is a probability distribution. Squaring both sides gives $4\, D_{\text{TV}}(p, q)^2 \le \chi^2(p\|q)$ (Tsybakov, 2009; Sason and Verdú, 2016). This is a stronger bound than (5).

What the bound guarantees. At each state where the projection enforces $\chi^2(\bar{\pi}_\theta\|\pi_{\text{old}}) \le \varepsilon$, it also ensures $D_{\text{TV}}^2 \le \varepsilon/4$. If this holds at every state, the maximum over states satisfies $\alpha^2 \le \varepsilon/4$, which is stronger than the condition used in the main text. Enforcing the constraint only at sampled states gives the same bound at those states, but does not bound the all-state maximum α. It therefore does not by itself satisfy the theorem's all-state assumption. □

E Effective sample size

Effective sample size (ESS) summarizes how evenly the importance weights are distributed. For N i.i.d. samples with weights $w_i = r_i \ge 0$, define the population quantity as $\text{ESS} = N(\mathbb{E}[w])^2/\mathbb{E}[w^2]$. Its empirical counterpart, $(\sum_i w_i)^2 / \sum_i w_i^2$, is the standard weight-based diagnostic of Kong (1992) (see Owen, 2013). The empirical value is defined when $\sum_i w_i > 0$. It is N when all weights are equal and approaches 1 when one weight dominates.

For importance ratios, Lemma 1 gives $\mathbb{E}[w] = 1$ and $\mathbb{E}[w^2] = 1 + \chi^2$. Substituting into the population definition gives the expression used in Section 3.3:

$$\text{ESS} = \frac{N \cdot 1^2}{1 + \chi^2} = \frac{N}{1 + \chi^2(\pi\|\pi_{\text{old}})}.$$

Thus $\chi^2 \le \varepsilon$ guarantees $\text{ESS}/N \ge 1/(1+\varepsilon)$ at each constrained token position. This is a bound on the weight-based ESS, not an exact precision guarantee for every estimated quantity: the estimator's variance also depends on the quantity being estimated. It is also a population bound, not a lower bound on the empirical ESS of every realized batch.

F THE PROJECTION: PRIMAL, DUAL, AND PROPERTIES

This appendix proves Theorem 2. We first derive the projected distribution for a fixed multiplier, then show how to find that multiplier. At one position, assume $\pi_\theta, \pi_{\text{old}} > 0$ on the finite represented support and $\varepsilon > 0$. We solve

$$\min_p \ \chi^2(p \,\|\, \pi_\theta) \quad \text{s.t.} \quad \chi^2(p \,\|\, \pi_{\text{old}}) \le \varepsilon, \qquad \textstyle\sum_x p(x) = 1.$$

We first solve without the constraints $p(x) \ge 0$, then verify that the solution is positive. It therefore also solves the original problem.

Step 1: include the constraints in the objective. Introduce a multiplier $\eta \ge 0$ for the divergence constraint and a multiplier $\lambda \in \mathbb{R}$ for normalization. The resulting function is called the Lagrangian. We will choose the multipliers so that its minimizer sums to one and satisfies the divergence constraint. If $\eta > 0$, the divergence must equal ε; if the constraint is inactive, $\eta = 0$. This is the complementary-slackness condition. These conditions are sufficient for optimality because the problem is convex. Strong duality also holds: $p = \pi_{\text{old}}$ satisfies the equality constraint and has divergence $0 < \varepsilon$, so Slater's condition is met (Boyd and Vandenberghe, 2004).

Step 2: solve for the distribution at a fixed multiplier. Appendix C shows that

$$\chi^2(p \,\|\, q) = \sum_x \frac{p(x)^2}{q(x)} - 1,$$

for any p summing to one. We use these quadratic expressions as an equivalent objective and constraint, while keeping normalization as an explicit equality. The Lagrangian of this equivalent problem is

$$\mathcal{L}(p, \eta, \lambda) = \underbrace{\sum_x \frac{p(x)^2}{\pi_\theta(x)} - 1}_{\chi^2(p\|\pi_\theta)} + \eta\Big(\underbrace{\sum_x \frac{p(x)^2}{\pi_{\text{old}}(x)} - 1}_{\chi^2(p\|\pi_{\text{old}})} - \varepsilon\Big) + \lambda\Big(\sum_x p(x) - 1\Big),$$

which regroups as

$$\mathcal{L}(p, \eta, \lambda) = \sum_x p(x)^2 \Big(\frac{1}{\pi_\theta(x)} + \frac{\eta}{\pi_{\text{old}}(x)}\Big) - (1 + \eta) - \eta\varepsilon + \lambda\Big(\sum_x p(x) - 1\Big).$$

For fixed multipliers, each coordinate appears in its own quadratic term. We can therefore minimize over each $p(x)$ separately by setting its derivative to zero:

$$\frac{\partial \mathcal{L}}{\partial p(x)} = 2p(x)\Big(\frac{1}{\pi_\theta(x)} + \frac{\eta}{\pi_{\text{old}}(x)}\Big) + \lambda = 0$$

$$\implies \quad p(x) = -\frac{\lambda}{2} \cdot \frac{1}{\frac{1}{\pi_\theta(x)} + \frac{\eta}{\pi_{\text{old}}(x)}} = -\frac{\lambda}{2} \cdot \frac{\pi_\theta(x)\,\pi_{\text{old}}(x)}{\pi_{\text{old}}(x) + \eta\,\pi_\theta(x)},$$

The last step multiplies numerator and denominator by $\pi_\theta(x)\pi_{\text{old}}(x)$. Every coordinate has the same factor $-\lambda/2$. The condition $\sum_x p(x) = 1$ gives $-\lambda/2 = 1/Z(\eta)$ with $Z(\eta) = \sum_x \pi_\theta(x)\pi_{\text{old}}(x)/(\pi_{\text{old}}(x) + \eta\pi_\theta(x))$, exactly the formula (7). All factors are positive, so $p_\eta(x) > 0$ and the nonnegativity constraints are satisfied. The objective has positive-definite Hessian $2\,\text{diag}(1/\pi_\theta)$, which makes it strictly convex. The constrained minimizer is therefore unique.

Step 3: reduce the remaining optimization to one scalar. For fixed η, let $D(\eta)$ be the minimum of the objective plus the divergence-constraint term over distributions that sum to one. The minimizer is p_η from Step 2. Maximizing this dual function over $\eta \ge 0$ gives the required multiplier. To compute it, write $d_x = \pi_{\text{old}}(x) + \eta\pi_\theta(x)$ so that $p_\eta(x) = \frac{1}{Z} \cdot \frac{\pi_\theta(x)\pi_{\text{old}}(x)}{d_x}$ and $\frac{1}{\pi_\theta(x)} + \frac{\eta}{\pi_{\text{old}}(x)} = \frac{d_x}{\pi_\theta(x)\pi_{\text{old}}(x)}$,

$$\sum_x p_\eta(x)^2 \Big(\frac{1}{\pi_\theta(x)} + \frac{\eta}{\pi_{\text{old}}(x)}\Big) = \frac{1}{Z^2} \sum_x \frac{\pi_\theta(x)^2 \pi_{\text{old}}(x)^2}{d_x^2} \cdot \frac{d_x}{\pi_\theta(x)\pi_{\text{old}}(x)}$$

$$= \frac{1}{Z^2} \sum_x \frac{\pi_\theta(x)\pi_{\text{old}}(x)}{d_x} = \frac{Z}{Z^2} = \frac{1}{Z(\eta)}.$$

Since p_η sums to one, its normalization term is zero, giving

$$D(\eta) = \mathcal{L}(p_\eta, \eta, \lambda(\eta)) = \frac{1}{Z(\eta)} - 1 - \eta(1+\varepsilon).$$

For each fixed normalized p, the expression minimized to obtain D is affine in η. The minimum over these expressions is concave, so D is concave. To compute its derivative, first differentiate Z:

$$Z'(\eta) = \sum_x \frac{\partial}{\partial \eta} \frac{\pi_\theta(x)\pi_{\text{old}}(x)}{\pi_{\text{old}}(x) + \eta\pi_\theta(x)} = -\sum_x \frac{\pi_\theta(x)^2 \pi_{\text{old}}(x)}{d_x^2} =: -B(\eta),$$

Here the derivative of $1/d_x$ with respect to η is $-\pi_\theta(x)/d_x^2$. Hence

$$D'(\eta) = -\frac{Z'(\eta)}{Z(\eta)^2} - (1+\varepsilon) = \frac{B(\eta)}{Z(\eta)^2} - 1 - \varepsilon.$$

The divergence of p_η from the old policy is

$$\begin{aligned}\chi^2(p_\eta \,\|\, \pi_{\text{old}}) &= \sum_x \frac{p_\eta(x)^2}{\pi_{\text{old}}(x)} - 1 = \frac{1}{Z^2}\sum_x \frac{\pi_\theta(x)^2 \pi_{\text{old}}(x)^2}{d_x^2\, \pi_{\text{old}}(x)} - 1 \\ &= \frac{1}{Z^2}\sum_x \frac{\pi_\theta(x)^2 \pi_{\text{old}}(x)}{d_x^2} - 1 = \frac{B}{Z^2} - 1.\end{aligned}$$

Thus the derivative of the dual is the divergence minus its budget:

$$D'(\eta) = \chi^2(p_\eta \,\|\, \pi_{\text{old}}) - \varepsilon = h(\eta).$$

When the actor distribution is outside the constraint, the optimal multiplier is positive and satisfies $h(\eta) = 0$. We therefore need to solve only one scalar equation. If the actor is already feasible, the optimal multiplier is zero, as shown below.

Step 4: show that h is strictly decreasing. Concavity implies that $h = D'$ is nonincreasing. The following lemma shows that it is strictly decreasing whenever the two input distributions differ. This will establish that an active projection has only one root.

Lemma 3 (Strict monotonicity of the dual derivative)**.** *Define $C(\eta) = \sum_x \pi_\theta(x)^3 \pi_{\text{old}}(x)/d_x^3$. Then $h'(\eta) = -2C/Z^2 + 2B^2/Z^3 = 2\big(B^2 - ZC\big)/Z^3$, and $h'(\eta) < 0$ for all $\eta \geq 0$ whenever $\pi_\theta \neq \pi_{\text{old}}$.*

Proof. Compute the derivative. Differentiate $h = B/Z^2 - 1 - \varepsilon$ with the quotient rule, using $Z' = -B$ (computed above) and

$$B'(\eta) = \sum_x \frac{\partial}{\partial \eta} \frac{\pi_\theta(x)^2 \pi_{\text{old}}(x)}{d_x^2} = \sum_x \pi_\theta(x)^2 \pi_{\text{old}}(x) \cdot \Big(-\frac{2\pi_\theta(x)}{d_x^3} \Big) = -2C,$$

giving

$$h' = \frac{B'}{Z^2} - \frac{2B\,Z'}{Z^3} = -\frac{2C}{Z^2} + \frac{2B^2}{Z^3} = \frac{2\big(B^2 - ZC\big)}{Z^3}.$$

Show that the derivative is negative. We need $B^2 < ZC$. The Cauchy–Schwarz inequality: for nonnegative sequences, $\big(\sum_x \sqrt{a_x c_x}\big)^2 \leq \big(\sum_x a_x\big)\big(\sum_x c_x\big)$, with equality if and only if the two sequences are proportional. Set

$$a_x = \frac{\pi_\theta(x)\pi_{\text{old}}(x)}{d_x} \quad (\text{so } Z = \textstyle\sum_x a_x), \qquad c_x = \frac{\pi_\theta(x)^3 \pi_{\text{old}}(x)}{d_x^3} \quad (\text{so } C = \textstyle\sum_x c_x),$$

Then

$$\sqrt{a_x\, c_x} = \sqrt{\frac{\pi_\theta(x)^4 \pi_{\text{old}}(x)^2}{d_x^4}} = \frac{\pi_\theta(x)^2 \pi_{\text{old}}(x)}{d_x^2} = b_x \quad (\text{so } B = \textstyle\sum_x \sqrt{a_x c_x}).$$

Cauchy–Schwarz gives $B^2 \leq ZC$. Equality requires $c_x/a_x = \pi_\theta(x)^2/d_x^2$ to be constant across x. This would give $\pi_\theta(x) = k(\pi_{\text{old}}(x) + \eta\pi_\theta(x))$, or $\pi_\theta(x)(1 - k\eta) = k\pi_{\text{old}}(x)$, for one constant $k > 0$. The distributions would then be proportional. Since both sum to one, they would be equal. Therefore $\pi_\theta \neq \pi_{\text{old}}$ implies $B^2 < ZC$ and $h' < 0$. □

Step 5: show that the required solution exists. At $\eta = 0$, $Z(0) = \sum_x \pi_\theta(x) = 1$ and $p_0 = \pi_\theta$, so $h(0) = \chi^2(\pi_\theta \| \pi_{\text{old}}) - \varepsilon$, which is positive exactly when the actor is outside the constraint. As $\eta \to \infty$, the denominator $d_x \approx \eta \pi_\theta(x)$, so $p_\eta(x) \to \pi_{\text{old}}(x) / \sum_y \pi_{\text{old}}(y) = \pi_{\text{old}}(x)$. Hence $\chi^2(p_\eta \| \pi_{\text{old}}) \to 0$ and $h \to -\varepsilon < 0$. A continuous, strictly decreasing function that starts positive and ends negative has exactly one root, $\eta^\star$. At this root the constraint holds with equality ($\chi^2(p_{\eta^\star} \| \pi_{\text{old}}) = \varepsilon$), which is complementary slackness with $\eta^\star > 0$. When instead $h(0) \le 0$, the unconstrained minimizer $p = \pi_\theta$ is already feasible. Its objective value is zero, the smallest possible value, so the projection returns π_θ unchanged. This also covers $\pi_\theta = \pi_{\text{old}}$. If $\varepsilon = 0$, the only feasible distribution is π_{old}; return it directly rather than searching for a finite root.

How projection changes probability ratios. For an active projection, divide (7) by $\pi_{\text{old}}(x)$ and write $r(x) = \pi_\theta(x)/\pi_{\text{old}}(x)$,

$$\bar{r}(x) = \frac{p_{\eta^\star}(x)}{\pi_{\text{old}}(x)} = \frac{1}{Z} \cdot \frac{\pi_\theta(x)}{d_x} = \frac{1}{Z} \cdot \frac{\pi_\theta(x)/\pi_{\text{old}}(x)}{1 + \eta^\star \, \pi_\theta(x)/\pi_{\text{old}}(x)} = \frac{1}{Z} \cdot \frac{r(x)}{1 + \eta^\star r(x)} = \varphi\big(r(x)\big).$$

The map $\varphi(u) = Z^{-1} u/(1 + \eta^\star u)$ has derivative $\varphi'(u) = Z^{-1}(1 + \eta^\star u)^{-2} > 0$: it is strictly increasing, so it preserves the ordering of ratios. It also satisfies $\varphi(u) < 1/(Z\eta^\star)$. The relative factor $\varphi(u)/u = 1/[Z(1 + \eta^\star u)]$ decreases with u, so larger ratios are reduced more relative to smaller ones. These are properties of the ratio transformation for fixed Z and $\eta^\star$; they do not imply that the full projection has a nonzero derivative in every input direction. Appendix G accounts for the dependence of Z and $\eta^\star$ on the input.

Computing the root. For an active projection with $\varepsilon > 0$, start with a positive upper bound, for example $\eta_{\text{hi}} = 1$, and double it ($\eta_{\text{hi}} \leftarrow 2\eta_{\text{hi}}$) until $h(\eta_{\text{hi}}) < 0$, then bisect on $[0, \eta_{\text{hi}}]$. Each evaluation of h is an $O(k)$ sum over the support. Bisection halves the bracket width at each step. Optional Newton steps can use the derivative from Lemma 3, but they should remain inside the bracket; otherwise, take a bisection step.

G CLOSED-FORM GRADIENTS

This appendix assumes $\varepsilon > 0$ and strictly positive input distributions on a fixed finite support. It proves Proposition 3 and derives the closed-form gradient in (9). We are given the loss gradient with respect to the projected probabilities, $u(y) = \partial \mathcal{L}/\partial \bar{\pi}_\theta(y)$, and need the gradient with respect to the input probabilities, $\partial \mathcal{L}/\partial \pi_\theta(x)$. The old policy π_{old} stays fixed. By the chain rule,

$$\frac{\partial \mathcal{L}}{\partial \pi_\theta(x)} = \sum_y u(y) \, \frac{\partial \bar{\pi}_\theta(y)}{\partial \pi_\theta(x)},$$

Changing π_θ affects the projected distribution in two ways: it changes the probabilities in (7) directly, and it changes the root $\eta^\star$. We compute these two contributions separately and then add them.

Notation. Consider one position where projection is needed. Define

$$d_x = \pi_{\text{old}}(x) + \eta \pi_\theta(x), \quad a_x = \frac{\pi_\theta(x)\pi_{\text{old}}(x)}{d_x}, \quad b_x = \frac{\pi_\theta(x)^2 \pi_{\text{old}}(x)}{d_x^2}, \quad c_x = \frac{\pi_\theta(x)^3 \pi_{\text{old}}(x)}{d_x^3},$$

and let $Z = \sum_x a_x$, $B = \sum_x b_x$, $C = \sum_x c_x$. Then $p(x) = a_x/Z$ is the projected distribution when $\eta = \eta^\star$. Appendix F gives $h(\eta, \pi_\theta) = B/Z^2 - 1 - \varepsilon$. Differentiating the expressions above with respect to η gives $\partial a_x/\partial \eta = -b_x$ and $\partial b_x/\partial \eta = -2c_x$; summing gives $Z' = -B$ and $B' = -2C$.

Step 1: differentiate with respect to the input. First hold η fixed. Since $\partial d_x/\partial \pi_\theta(x) = \eta$, the quotient rule gives

$$\frac{\partial a_x}{\partial \pi_\theta(x)} = \frac{\pi_{\text{old}}(x)\, d_x - \pi_\theta(x)\pi_{\text{old}}(x) \cdot \eta}{d_x^2} = \frac{\pi_{\text{old}}(x)\big(d_x - \eta \pi_\theta(x)\big)}{d_x^2} = \frac{\pi_{\text{old}}(x)^2}{d_x^2},$$

since $d_x - \eta\pi_\theta(x) = \pi_{\text{old}}(x)$ by definition of d_x. Similarly,

$$\frac{\partial b_x}{\partial \pi_\theta(x)} = \frac{2\pi_\theta(x)\pi_{\text{old}}(x)\, d_x^2 - \pi_\theta(x)^2 \pi_{\text{old}}(x) \cdot 2 d_x \eta}{d_x^4}$$
$$= \frac{2\pi_\theta(x)\pi_{\text{old}}(x)\big(d_x - \eta\pi_\theta(x)\big)}{d_x^3} = \frac{2\pi_\theta(x)\pi_{\text{old}}(x)^2}{d_x^3}.$$

For $y \neq x$, neither a_y nor b_y depends on $\pi_\theta(x)$, so their partial derivatives are zero. Thus $\pi_\theta(x)$ changes Z only through a_x, and B only through b_x. Substituting these derivatives into $h = B/Z^2 - 1 - \varepsilon$ gives

$$\frac{\partial h}{\partial \pi_\theta(x)} = \frac{1}{Z^2}\frac{\partial b_x}{\partial \pi_\theta(x)} - \frac{2B}{Z^3}\frac{\partial a_x}{\partial \pi_\theta(x)} = \frac{2\pi_\theta(x)\pi_{\text{old}}(x)^2}{d_x^3 Z^2} - \frac{2B\,\pi_{\text{old}}(x)^2}{d_x^2 Z^3},$$
$$\frac{\partial h}{\partial \eta} = -\frac{2C}{Z^2} + \frac{2B^2}{Z^3} < 0,$$

where Lemma 3 establishes the strict inequality for the derivative with respect to η.

Step 2: differentiate the projected probabilities. Holding η fixed, differentiate $p(y) = a_y/Z$ with respect to $\pi_\theta(x)$:

$$\frac{\partial p(y)}{\partial \pi_\theta(x)}\Big|_\eta = \frac{\delta_{xy}}{Z}\frac{\partial a_x}{\partial \pi_\theta(x)} - \frac{a_y}{Z^2}\frac{\partial Z}{\partial \pi_\theta(x)} = \delta_{xy}\,\frac{\pi_{\text{old}}(x)^2}{d_x^2 Z} - p(y)\,\frac{\pi_{\text{old}}(x)^2}{d_x^2 Z},$$

Here δ_{xy} is 1 if $x = y$ and 0 otherwise, and $\partial Z/\partial\pi_\theta(x) = \partial a_x/\partial\pi_\theta(x)$. Next, hold π_θ fixed and differentiate with respect to η:

$$\frac{\partial p(y)}{\partial \eta} = \frac{1}{Z}\frac{\partial a_y}{\partial \eta} - \frac{a_y}{Z^2}\,Z' = \frac{-b_y}{Z} + \frac{a_y B}{Z^2} = \frac{p(y)\,B - b_y}{Z}.$$

Step 3: differentiate the root. The root $\eta^\star(\pi_\theta)$ satisfies $h(\eta^\star, \pi_\theta) = 0$. When $\pi_\theta(x)$ changes, $\eta^\star$ must also change to keep this equation satisfied. Differentiating the equation gives the implicit-function derivative (Dontchev and Rockafellar, 2009):

$$\frac{\partial h}{\partial \pi_\theta(x)} + \frac{\partial h}{\partial \eta}\cdot\frac{\partial \eta^\star}{\partial \pi_\theta(x)} = 0 \quad\Longrightarrow\quad \frac{\partial \eta^\star}{\partial \pi_\theta(x)} = -\,\frac{\partial h/\partial \pi_\theta(x)}{\partial h/\partial \eta}.$$

The denominator is nonzero because $\partial h/\partial\eta < 0$ (Lemma 3). We therefore obtain the derivative by dividing two scalars. Unlike differentiation through a full KKT system, this step does not require solving a linear system.

Step 4: compute the loss gradient. Apply the chain rule to combine the direct change in p with the change through $\eta^\star$, then multiply by the given loss gradient u and sum:

$$\frac{\partial \mathcal{L}}{\partial \pi_\theta(x)} = \sum_y u(y)\left[\frac{\partial p(y)}{\partial \pi_\theta(x)}\Big|_\eta + \frac{\partial p(y)}{\partial \eta}\cdot\frac{\partial \eta^\star}{\partial \pi_\theta(x)}\right].$$

Using Step 2, the direct contribution is

$$\sum_y u(y)\,\frac{\partial p(y)}{\partial \pi_\theta(x)}\Big|_\eta = \big(u(x) - \langle u, p\rangle\big)\frac{\pi_{\text{old}}(x)^2}{d_x^2 Z}, \qquad \langle u, p\rangle = \sum_y u(y)p(y),$$

For the contribution through $\eta^\star$, define the scalar

$$s = \sum_y u(y)\,\frac{\partial p(y)}{\partial \eta} = \frac{\langle u, p\rangle B - \langle u, b\rangle}{Z},$$

Then this contribution is $s\cdot\partial\eta^\star/\partial\pi_\theta(x) = -\big(s/(\partial h/\partial\eta)\big)\cdot\partial h/\partial\pi_\theta(x)$. Substituting the expression for $\partial h/\partial\pi_\theta(x)$ from Step 1 yields exactly (9).

Cost and differentiability. Computing d, a, b, c requires only elementwise operations. Computing $Z, B, C, \langle u, p\rangle, \langle u, b\rangle, s$ requires sums over the k represented tokens. The backward pass therefore costs $O(k)$ per position, without constructing a full Jacobian or solving a linear system. At strictly feasible positions ($\chi^2(\pi_\theta \| \pi_{\text{old}}) < \varepsilon$), the projection returns the input unchanged, so the loss gradient is u itself. Multiplying by the softmax Jacobian $\partial \pi_\theta / \partial \ell = \operatorname{diag}(\pi_\theta) - \pi_\theta \pi_\theta^\top$ gives the gradient with respect to the logits ℓ. The projection is smooth where the constraint is strictly active or strictly inactive. At the feasibility boundary, it is continuous but is generally not differentiable. Returning the identity-branch gradient at equality is an implementation convention, not a derivative of the full map at that boundary. If $\varepsilon = 0$, the output is the fixed distribution π_{old} and the input gradient is zero.

This derivation gives only the gradient through the projection. In (8), the KL term treats $\bar{\pi}_\theta$ as a fixed target and contributes its own direct gradient with respect to π_θ. The old policy and the advantages are held fixed throughout the actor update.

H SPARSIFICATION: SCHEME AND ERROR BOUND

This appendix describes the sparse distributions used by CHIP and bounds the difference between their χ^2 divergence and that of the original distributions. The current and old policies may retain different token sets.

Sparse distributions. Let p and q be the current and old next-token distributions over a vocabulary of size V. Each policy selects its own tokens using the top-K and cumulative-mass rule, and the sampled token is always included. Let the resulting sets be S_p and S_q.

We use the actual retained probability mass for normalization. It need not equal $1-\delta$: the cumulative sum may exceed the threshold, or the storage limit may prevent reaching it. Define

$$\delta_p = 1 - \sum_{x \in S_p} p(x), \qquad \tau_p = (V - |S_p|) p_d,$$

where $0 < p_d < 1/V$ is the default probability assigned to each discarded token. The sparse distribution is

$$p'(x) = \begin{cases} \gamma_p p(x), & x \in S_p, \\ p_d, & x \notin S_p, \end{cases} \qquad \gamma_p = \frac{1 - \tau_p}{1 - \delta_p}.$$

The old-policy distribution q' is constructed in the same way using its own set S_q, discarded mass δ_q, and default mass τ_q. Thus both sparse distributions sum to one. Their retained sets and normalization factors need not be equal.

Computing the projection. We represent both sparse distributions on the union $U = S_p \cup S_q$. A token retained by only one policy has its retained value on that side and the default probability on the other.

The remaining $n = V - |U|$ tokens have probability p_d under both sparse distributions. When $n > 0$, we represent these tokens by one aggregate category of probability np_d.

This aggregation is exact for the projection because all these tokens have identical input probabilities and therefore receive identical projected probabilities. Their contributions to the sums in Appendices F and G are

$$Z_{\text{tail}} = \frac{np_d}{1+\eta}, \qquad B_{\text{tail}} = \frac{np_d}{(1+\eta)^2}, \qquad C_{\text{tail}} = \frac{np_d}{(1+\eta)^3}.$$

Including these terms gives the same result as explicitly representing every discarded token. In the CHIP objective, the policy-gradient term uses only the sampled token, which lies in U, and the KL term treats the projection as a fixed target. Thus each scalar-function evaluation and the backward pass through the projection require $O(|U|)$ work.

Approximation error. The next result compares the original and sparse divergences without requiring $S_p = S_q$.

Theorem 3 (Sparsification error for independently selected sets)**.** *Let p, q be strictly positive probability distributions, and let p', q' be their sparse distributions defined above. Define*

$$M_p = \max(\delta_p, \tau_p), \qquad M_q = \max(\delta_q, \tau_q), \qquad R = \max_x \left\{ \frac{p(x)}{q(x)}, \frac{p'(x)}{q'(x)} \right\}.$$

Then

$$\left|\chi^2(p\|q) - \chi^2(p'\|q')\right| \le 2RM_p + R^2 M_q. \tag{10}$$

Proof. First, bound how much sparsification changes each distribution. On the retained set, normalization changes the total mass by $|\delta_p - \tau_p|$. On the discarded set, the original and replacement masses are δ_p and τ_p. Therefore,

$$\begin{aligned}\|p - p'\|_1 &= |\delta_p - \tau_p| + \sum_{x \notin S_p} |p(x) - p_d| \\ &\le |\delta_p - \tau_p| + \delta_p + \tau_p = 2M_p.\end{aligned}$$

Similarly, $\|q - q'\|_1 \le 2M_q$.

Next, connect the original and sparse distributions by

$$p_t = (1-t)p + tp', \qquad q_t = (1-t)q + tq', \qquad 0 \le t \le 1.$$

At each token, $r_t(x) = p_t(x)/q_t(x)$ is a weighted average of the two endpoint ratios. Hence $0 < r_t(x) \le R$.

Let $F(t) = \chi^2(p_t\|q_t)$, $\Delta p = p' - p$, and $\Delta q = q' - q$. Differentiating F, and using $\sum_x \Delta p(x) = \sum_x \Delta q(x) = 0$, gives

$$F'(t) = \sum_x (2r_t(x) - R)\Delta p(x) - \sum_x \left(r_t(x)^2 - \frac{R^2}{2} \right) \Delta q(x).$$

The two coefficients have absolute values at most R and $R^2/2$, respectively. Thus

$$|F'(t)| \le R\|\Delta p\|_1 + \frac{R^2}{2}\|\Delta q\|_1.$$

Integrating from 0 to 1 and substituting the preceding bounds yields

$$|F(1) - F(0)| \le 2RM_p + R^2 M_q,$$

which proves the claim. □

What the bound guarantees. The bound accounts for replacing discarded probabilities and renormalizing the retained probabilities. It applies when the two policies select different token sets.

It bounds the magnitude of the error, not its sign: sparse divergence may be larger or smaller than the original divergence. The bound is small when the discarded and default-assigned masses are small and both the original and sparse probability ratios are controlled. Small discarded mass alone is insufficient, because assigning a small default probability on only one side can produce a large sparse ratio.

A usable numerical bound therefore requires information about the original ratios as well as the sparse ratios. The projection is exact for the sparse distributions; approximation relative to the original distributions is a separate issue quantified by the bound above.

I Additional experimental results

Table 4 lists the common math and code training settings. In the measured runs, projection adds less than 1% to training-step time. Appendix I.1 gives diagnostics and additional controls for the model-scale experiment; Appendix J summarizes limitations. Appendix I.2 adds the Eurus-Code comparison, and Appendix I.3 provides training details and checkpoint results for the multi-turn coding experiment in Section 6.3.

Table 4: Common training settings for the math and Eurus-Code experiments. Within each paired comparison, these settings are the same for the clipped baseline and CHIP. Algorithm-specific settings and evaluation checkpoints are given in Section 6 and Appendix I.2. The multi-turn coding experiment uses different settings, listed in Appendix I.3.

	Value	Note
Rollout batch size	256 prompts	per training step
Group size G	8	responses per prompt; the GRPO baseline
Trajectories per step	2048	256×8
Minibatch size	32 prompts	8 optimizer steps per training step
Epochs per batch	1	each rollout batch is consumed once
Learning rate	10^{-6}	constant, no warmup or decay
Max prompt / response	2048 / 4096	tokens
Training schedule	300 steps	comparison checkpoints specified per experiment
Math training set	16,893	DAPO-Math-17k
Code training set	24,576	Eurus-Code, six shards of 4096
Math validation suite	6,431	eight benchmarks, scored every 10 steps

Sources and comparison protocol. The math evaluation suite includes MATH-500, a subset of MATH (Hendrycks et al., 2021; Lightman et al., 2024), Minerva (Lewkowycz et al., 2022), OlympiadBench (He et al., 2024), and Omni-MATH (Gao et al., 2025). Implementation choices can substantially affect PPO and TRPO comparisons (Engstrom et al., 2020). Within each comparison, we keep the training recipe fixed and change the policy-update loss, as described in Section 6.

I.1 Clipping, projection, and additional controls

Figure 4 reports training diagnostics for the six model sizes in Section 6.1. Taking the median activation rate over the logged training steps, GRPO clipping activates on 0.16–0.29% of sampled tokens, while CHIP projects the represented next-token distribution at 1.9–20.3% of sampled positions. The high projection rate means that the proposed distribution exceeds the χ^2 budget at many positions and requires correction. Clipping checks only the sampled token's probability ratio and advantage. That ratio can stay within the clipping interval while other tokens' probabilities change substantially, so clipping can miss changes in the rest of the distribution. CHIP addresses this blind spot by checking the whole represented next-token distribution.

For each CHIP run, we also take the median of the logged batch-mean χ^2 values. Across model sizes, these medians range from 8.5 to 1.2×10^3 before projection and from 0.0027 to 0.0122 after projection. This reduction shows that the projection actively corrects large distributional changes before they enter the policy objective.

CHIP preserves policy entropy better during math training, while the clipped GRPO baseline shows a larger overall decline. CHIP ends with higher entropy at every model size: final values range from 0.083 to 0.249 for CHIP and from 0.021 to 0.109 for GRPO. Thus CHIP retains more token-level diversity, while the baseline's predictions become more concentrated.



Figure 4: Training diagnostics for the six model sizes in Figure 1. **(a)** Median clipping and projection activation rates over training, on a logarithmic scale. GRPO clipping activates on 0.16–0.29% of sampled tokens; CHIP projection activates at 1.9–20.3% of sampled positions. **(b)** Batch-mean χ^2 between the current and old distributions in the CHIP runs, before and after projection. **(c)** Policy entropy for both methods. Curves use nine-step running means.

The 4B run. CHIP achieves higher late-training accuracy, leading GRPO by an average of 18.2 percentage points over the final sixty shared training steps. GRPO's accuracy falls from 54.2% to 26.9% after step 150, accompanied by a sharp entropy decline. CHIP retains more policy entropy, combining better validation performance with less concentrated predictions.

The 14B run. CHIP outperforms GRPO at 28 of the 30 evaluated checkpoints and reaches its highest validation accuracy, 62.4%, at the final step—6.5 percentage points above GRPO. While GRPO loses 3.5 points from its peak, CHIP finishes at its best performance. CHIP also retains higher policy entropy, 0.103 versus 0.034, showing that its accuracy advantage is accompanied by better preservation of token-level diversity.

Both clipping and CHIP are error-control mechanisms. Their practical role is to act as a brake on policy updates, limiting how far the distribution moves from the old policy at each update. Figure 5 shows their different effects in the 14B runs. GRPO's sampled-token KL estimate rises to 7.1 times its early value, while clipping activates on only 0.1–0.2% of tokens. CHIP's post-projection χ^2 instead falls to 29% of its early value, with projection active on 1.5–6.7% of positions. CHIP therefore provides the intended braking effect on its projected distribution: policy movement is smaller later in training. Clipping does not produce this late-training slowdown in the GRPO run, whose update measure instead increases. CHIP also retains higher policy entropy.

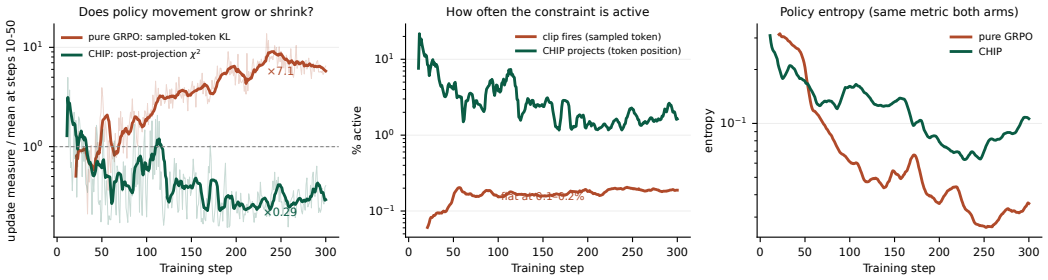

Figure 5: Update diagnostics for the 14B runs. Left: GRPO's sampled-token KL estimate and CHIP's post-projection χ^2 on the represented distribution, each normalized by its own mean over steps 10–50. The curves show relative changes in their respective metrics. Middle: fractions of sampled tokens affected by clipping and positions requiring projection. Right: token entropy, measured in the same way for both methods.

Removing the reference-policy KL penalty. GRPO's KL loss penalizes deviation from a reference policy, providing an additional form of error control. To remove its influence, we compare CHIP with a clip-only GRPO baseline that omits this KL term. In Table 5, CHIP achieves higher training and evaluation scores at all four model sizes, with DAPO validation gains of 8.5–21.6 percentage points. CHIP's advantage therefore persists without the reference-policy KL penalty in the baseline.

		0.6B	1.7B	4B	8B
DAPO Train	Clip only	0.138	0.324	0.453	0.594
	CHIP	0.344 (+20.6)	0.535 (+21.1)	0.667 (+21.4)	0.716 (+12.2)
DAPO Eval	Clip only	0.130	0.310	0.359	0.497
	CHIP	0.284 (+15.4)	0.451 (+14.1)	0.575 (+21.6)	0.582 (+8.5)
AIME'25 Eval	Clip only	0.000	0.067	0.117	0.233
	CHIP	0.100 (+10.0)	0.200 (+13.3)	0.432 (+31.6)	0.372 (+13.9)

Table 5: CHIP versus a clipped baseline without a reference-policy KL penalty, at four model sizes. Both methods are evaluated at step 300, with clip range 0.2 for the baseline and $\varepsilon = 0.05$ for CHIP. Training scores are medians over the last twenty steps; DAPO Eval is the mean over the same eight-benchmark suite as Table 1. Parentheses give CHIP minus baseline in percentage points.

I.2 Executable code generation on Eurus-Code

This additional experiment uses Eurus-Code (Cui et al., 2025b), where generated programs are scored by executing tests in a sandbox. Training success rates are 4–14%. We report the 8B comparison, in which both methods train for 300 steps and CHIP uses $\varepsilon = 0.1$.

Training and evaluation. Both methods use GRPO advantages without a reference-policy KL penalty; CLIP uses a clipping range of 0.2. Prompt and response lengths are limited to 2048 and 4096 tokens. We evaluate every ten training steps using greedy decoding. Each program is scored by the fraction of test cases passed, using at most the first ten cases with a 10-second timeout per case. We report the mean score over APPS, CodeContests, Codeforces, and TACO.

Results. Figure 6 and Table 6 report validation scores averaged over four executable-code benchmarks. CHIP leads at all 21 retained shared checkpoints from step 60 through step 300, by 0.4–15.0 percentage points. Its average score over the last sixty steps is 0.545, compared with 0.425 for the clipped baseline, a 12.0-point difference. At step 300, the scores are 0.534 and 0.407, respectively, a 12.7-point difference. The baseline leads by 15.9 points at step 20 and 10.1 points at step 40. The baseline reaches its highest score, 0.462, at step 210, when CHIP leads by 9.2 points. CHIP's later scores remain around 0.53–0.55. CHIP's advantage thus persists across the retained later checkpoints: it reaches a higher performance plateau and maintains it through the end of training.

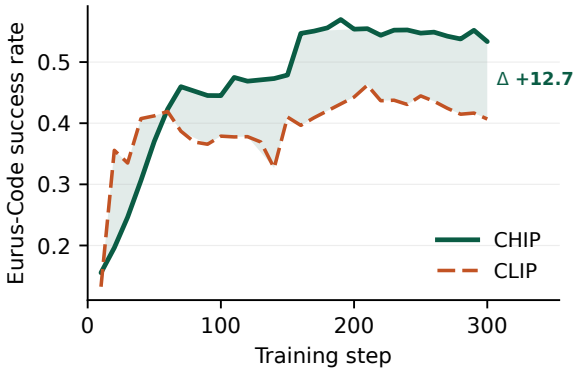

Figure 6: Eurus-Code validation scores. Solid: CHIP; dashed: CLIP; shading: score difference. We retain checkpoints with all four benchmark scores above 0.001, then iteratively remove checkpoint means below 60% of both nearest retained neighbors. Excluded steps: CHIP 80, 130; CLIP 180, 190.

Step	CLIP	CHIP	Δ
60	0.419	**0.423**	+0.4
100	0.379	**0.445**	+6.6
140	0.327	**0.473**	+14.6
160	0.397	**0.547**	+15.0
200	0.443	**0.554**	+11.1
240	0.431	**0.553**	+12.2
270	0.425	**0.542**	+11.8
300	0.407	**0.534**	+12.7

Table 6: Selected shared checkpoints. Scores are means over four executable-code benchmarks; Δ is CHIP minus CLIP in percentage points.

I.3 Multi-turn agentic coding: experimental settings

This subsection provides the training settings and evaluation protocol for the experiment in Section 6.3.

Training setup. We train Qwen3-32B on R2E-Gym (Jain et al., 2025) with a recipe adapted from DeepSWE (Luo et al., 2025). Each rollout batch contains 16 problems and eight trajectories per problem, giving 128 trajectories. The batch is split into four minibatches of 32 trajectories, so each rollout produces four optimizer updates. Both methods use the same data order, model initialization, optimizer, rollout settings, and update schedule.

Advantages and optimization. Both methods use leave-one-out advantages (Ahmadian et al., 2024) (`loop`): for group size $G = 8$, $\hat{A}_i = R_i - \frac{1}{G-1}\sum_{j \neq i} R_j$, where R_i is the reward of trajectory i for the same problem. Neither method normalizes this value by the group reward standard deviation or trains a critic. We use AdamW with learning rate 10^{-6}, no warmup, and weight decay 0.01. Neither method uses a reference-policy KL penalty or an entropy bonus.

Generation and resources. Each method uses four nodes with eight GPUs per node (32 GPUs). Sampling temperature is 1, with at most 50 agent steps per trajectory, a response-token limit of $32{,}768$, and a trajectory timeout of $5400\,\mathrm{s}$. Trajectories that hit a limit remain in the batch, but their response-loss masks are zero.

CLIP and CHIP objectives. CLIP uses the sampled-token clipped surrogate with the asymmetric ratio interval $[0.8, 1.28]$ and no dual-clip term. CHIP replaces this surrogate with the projection-based objective in Section 4.2, using $\varepsilon = 0.25$ and regression coefficient $\alpha = 1$. Its sparse representation uses a top-K cap of 64 and tail-mass threshold $\delta = 10^{-5}$, and always retains the sampled token. The regression term matches the actor to its projection; it is separate from a reference-policy KL penalty.

Evaluation. We measure pass@1 on the fixed 500-task SWE-bench Verified set. The evaluation uses the SGLang 0.4.6.post5 server with the rllm SWEAgent, temperature 1, and the fixed task order obtained with seed 42. We evaluate five segments of 100 tasks and require 500 nonempty, unique task results in canonical order. A task counts as solved when its generated patch passes the repository tests within the evaluation limits. The formal comparison reports checkpoints at steps 0, 10, . . ., 130 for both methods.

J LIMITATIONS

We evaluate CHIP on mathematical reasoning, executable code generation, and multi-turn coding tasks. Future work will examine variation across training seeds, generalization to held-out coding problems, and performance with additional models and training settings.