LEARNING FROM WHAT THE TEACHER TRUSTS: CONFIDENCE-ALIGNED ON-POLICY DISTILLATION

Anonymous authors
Paper under double-blind review

ABSTRACT

On-policy distillation is a promising paradigm for aligning Large Language Models. However, standard objectives compel the student to indiscriminately mimic the teacher, forcing the model to learn from low-agreement or noisy trajectories generated during exploration. While recent approaches attempt to mitigate this via external reward models or step-wise teacher intervention, they introduce significant computational and communication overhead. In this work, we propose Teacher-Confidence Aligned On-Policy Distillation (TCAD), a scalable, reward-free paradigm that leverages the teacher's intrinsic sequence-level confidence as a relative ranking signal among rollouts for the same prompt. By dynamically reweighting trajectories based on model-relative teacher certainty, TCAD softly down-weights low-agreement rollouts without requiring auxiliary reward models. To address the approximation error caused by vocabulary truncation in large-scale distillation, we introduce support-symmetric candidate construction and probability-proportional Monte Carlo optimization over a compact event space. These mechanisms reduce the distribution mismatch between teacher and student supports while optimizing a compact support-plus-tail divergence with a controllable Monte Carlo sample-count trade-off. Extensive experiments across diverse benchmarks demonstrate that our proposed method consistently improves over standard on-policy distillation baselines, achieving strong alignment efficiency without external supervision.

1 INTRODUCTION

Knowledge distillation Hinton et al. (2015); Yang et al. (2025); Team (2025) has established itself as a cornerstone technique for transferring capabilities from powerful, large-scale teacher models to smaller, efficient student models. In the modern Large Language Model (LLM) OpenAI (2023); Yang et al. (2025); Touvron et al. (2023) landscape, On-Policy Distillation (OPD) Gu et al. (2024); Agarwal et al. (2024) has emerged as a critical approach for alignment. Unlike offline approaches Taori et al. (2023) that rely on static datasets, OPD learns from trajectories sampled from the student's current policy. This dynamic interaction mitigates the exposure bias Gu et al. (2024); Arora et al. (2022) inherent in static supervision, and it often improves robustness to errors made by the student policy. However, standard OPD suffers from indiscriminate imitation Xiao et al. (2026); Xu et al. (2025). In the standard setup Gu et al. (2024); Agarwal et al. (2024), the distillation objective minimizes the divergence between teacher and student distributions at every step of a generated sequence, regardless of how well the sampled trajectory agrees with the teacher. This leads to a "blind imitation" dilemma: when the student generates a low-agreement or logically flawed trajectory, the objective still forces the student to mimic the teacher's token predictions conditioned on that erroneous context. Matching the teacher on erroneous prefixes can provide useful corrective supervision; the concern is not that such contexts are inherently harmful, but that uniform weighting does not distinguish their training utility from that of higher-agreement trajectories.

To address this problem, recent paradigms have sought to integrate quality control into the training loop. Some approaches introduce external quality signals, such as outcome reward or preference models, to evaluate and filter student rollouts Ouyang et al. (2022); Xiao et al. (2026), which necessitates expensive preference data and increases system complexity. Alternatively, teacher-intervention approaches mitigate indiscriminate imitation by increasing teacher involvement during sampling Xu et al. (2025). While this can improve trajectory quality, it may substantially increase

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.

sampling-time cost due to frequent teacher calls during decoding, amplifying both computation and communication overhead.

In this work, we argue that a capable teacher provides not only token-level probability targets but also intrinsic sequence-level confidence that can serve as a relative ranking signal for distributional consistency. Specifically, we use the teacher's length-normalized log-likelihood of a trajectory as a model-relative confidence signal to reweight on-policy rollouts. This metric is not intended to estimate absolute correctness. Rather, among multiple rollouts generated for the same prompt, it provides a model-relative ordering of trajectories that can be used to modulate their training contributions. Importantly, we do not claim that teacher confidence is a calibrated measure of ground-truth correctness, as it may be affected by domain shift or stylistic mismatch. We therefore treat it as a pragmatic signal for stabilizing distillation and mitigating the blind imitation of misaligned samples.

Based on this insight, we propose **Teacher-Confidence Aligned On-Policy Distillation** (TCAD), which transforms confidence scores into normalized rollout weights and modulates the distillation loss accordingly. This yields a lightweight form of agreement-aware training without relying on external reward models or step-wise teacher intervention. To ensure computational efficiency at scale, TCAD further employs truncated vocabulary supervision. However, standard truncation can induce a support mismatch between the teacher and the student and can distort the full-vocabulary divergence by renormalizing only selected tokens. Consequently, we introduce support-symmetric candidate construction to cover high-probability tokens from both distributions. Once this compact support-plus-tail event space is constructed, its divergence can be evaluated exactly; we additionally use probability-proportional Monte Carlo sampling as a stochastic optimization scheme whose value estimate is unbiased for the compact divergence. The support budget controls which token-level distinctions are retained, whereas the Monte Carlo sample count controls the quality–cost trade-off of stochastic optimization within that fixed representation.

Our contributions are summarized as follows:

- We introduce TCAD, a scalable, reward-free on-policy distillation paradigm that uses teacher sequence-level confidence as a weighting signal. This mitigates blind imitation on low-agreement trajectories without external reward models or step-wise teacher intervention.
- We propose support-symmetric candidate construction and probability-proportional Monte Carlo optimization for truncated distillation. The compact event space preserves residual tail mass, while Monte Carlo sampling provides an unbiased value estimator for the compact divergence; the resulting objective remains a compact approximation to the full-vocabulary divergence.
- We evaluate TCAD with EvalScope Team (2024) and OpenCompass Contributors (2023) on 15 benchmarks across four competencies. Under OPD-FKL (OPD with forward KL), TCAD improves macro-average by 6.09%$\sim$7.33% over student initialization and by 1.60%$\sim$1.69% over the corresponding OPD baseline, with representative gains of 10.20%$\sim$21.61% on GSM8K and 2.50%$\sim$30.16% on HumanEval.

2 Related Work

On-Policy Knowledge Distillation. Knowledge distillation (KD) Hinton et al. (2015) is a widely used paradigm in LLM post-training. Compared to offline distillation on fixed datasets, on-policy distillation learns from trajectories sampled from the student's current policy, which can reduce distribution shift induced by the student's evolving generation behavior Ross et al. (2011); Agarwal et al. (2024); Gu et al. (2024). A practical challenge in this setting is that rollouts vary substantially in agreement with the teacher: standard token-level objectives typically weight trajectories uniformly, so low-agreement rollouts may still contribute large gradients and can amplify errors. Prior work mitigates this issue by introducing external quality signals, such as reward or preference models in alignment pipelines Ouyang et al. (2022); Xiao et al. (2026), or by increasing teacher involvement during sampling Xu et al. (2025). In contrast, we study a lightweight alternative that uses sequence-level teacher confidence as a model-relative weighting signal to reweight on-policy rollouts, without relying on external verifiers. Unlike hard rejection, soft weighting retains all sampled responses; these weights are not importance-sampling corrections between teacher and student policies.

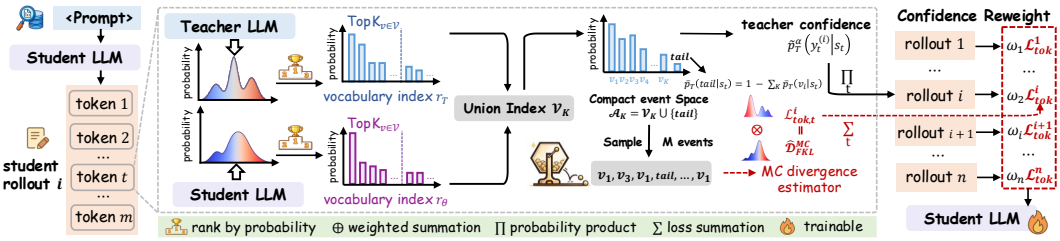

Figure 1: **Overview framework of TCAD.** The training operates at two granularities. **(1) Token-level (center):** At each generation step, we construct a support-symmetric candidate set $\mathcal{V}_K$ by combining high-probability tokens from both teacher and student distributions. The student is optimized with a Monte Carlo estimate of the chosen divergence over sampled events from the selected support plus an aggregate tail event. **(2) Sequence-level (right):** Multiple on-policy rollouts are reweighted using teacher confidence scores, producing normalized weights w_i that modulate each rollout's contribution to the aggregated objective.

Efficient Vocabulary-Level Distillation. From a scalability perspective, on-policy distillation also faces a practical bottleneck: computing token-level distillation losses over the full vocabulary is expensive, and restricted-support objectives are commonly used to approximate the full softmax Jean et al. (2015); Team (2025). However, deterministic truncation can introduce artifacts when the selected support undercovers high-probability tokens under either the teacher or the student distribution. It can also overemphasize the relative probabilities within a renormalized support while ignoring how the unselected vocabulary contributes to the original full-vocabulary divergence. Our method separates candidate selection, tail aggregation, and stochastic optimization. Support-symmetric selection retains high-ranked tokens from either distribution; tail aggregation preserves the total probability mass of omitted tokens. Monte Carlo sampling then estimates the resulting compact divergence using a finite number of draws from a divergence-specific proposal.

3 Method

In this section, we present TCAD, which addresses noisy on-policy rollouts and the cost of full-vocabulary token supervision. The method combines model-relative rollout weighting, support-symmetric candidate construction, and stochastic optimization over a compact event space.

3.1 Reward-Free Sequence Alignment

The student may generate rollouts with different degrees of agreement with the teacher. Rather than introducing an auxiliary reward or verifier, TCAD uses model-relative teacher confidence to weight these rollouts.

Let $\{y^{(i)}\}_{i=1}^{n} \sim \pi_\theta(\cdot \mid x)$ denote n responses sampled from the student for prompt x. We score each rollout using the teacher's length-normalized log-likelihood as a model-relative confidence measure:

$$\bar{\ell}_T(x, y^{(i)}) = \frac{1}{|y^{(i)}|} \sum_{t=1}^{|y^{(i)}|} \log p_T\big(y_t^{(i)} \mid s_t\big). \tag{1}$$

We convert these scores into normalized weights with a sharpened softmax:

$$w_i = \frac{\exp\big(\alpha\, \bar{\ell}_T(x, y^{(i)})\big)}{\sum_{j=1}^{n} \exp\big(\alpha\, \bar{\ell}_T(x, y^{(j)})\big)}, \qquad i = 1, \ldots, n. \tag{2}$$

Here, $\alpha > 0$ controls weight concentration, with $\alpha > 1$ sharpening relative to $\alpha = 1$. Since $\exp(\bar{\ell}_T) = \pi_T(y \mid x)^{1/|y|}$, the transformation raises geometric-mean token likelihoods, not sequence probabilities, to the power α (Appendix A.3). We use the score as a relative teacher-preference signal, not as a calibrated correctness estimator. We examine a weaker, testable criterion: for correct–incorrect rollout pairs (y^+, y^-) drawn from the same prompt, the score should rank the correct rollout higher more often than chance,

$$\Pr\big[\bar{\ell}_T(x, y^+) > \bar{\ell}_T(x, y^-)\big] > \frac{1}{2}. \tag{3}$$

This criterion is evaluated in Sec. 4.3; it does not by itself guarantee a benefit from the resulting weight magnitudes. The weights are treated as stop-gradient coefficients: low-confidence rollouts are attenuated but not removed. The weights are shift-invariant and become uniform as $\alpha \to 0$; with a unique maximum, $\alpha \to \infty$ selects that rollout. Length normalization does not eliminate all length or style bias.

3.2 Support-Symmetric Candidate Construction

Teacher-only truncation may miss tokens that are highly probable under the student, creating support mismatch when $K \ll |\mathcal{V}|$. We therefore construct a fixed-budget support that accounts for both distributions. Let $r_T(v \mid s_t)$ and $r_\theta(v \mid s_t)$ denote the rank of token v in the teacher and student distributions, respectively. We define a unified priority metric $\mathcal{M}(v \mid s_t)$ to select the top-K tokens:

$$\mathcal{V}_K(s_t) = \mathrm{TopK}_{v \in \mathcal{V}} \big(\mathcal{M}(v \mid s_t),\, K\big). \tag{4}$$

We instantiate this metric using the Union principle:

$$\mathcal{M}(v \mid s_t) = -\min \big(r_T(v \mid s_t),\, r_\theta(v \mid s_t)\big). \tag{5}$$

We refer to this fixed-budget selection rule as *Dual-Union*. The min-rank priority gives each token credit for being highly ranked by either distribution while keeping the support size fixed at K. The priority is symmetric under exchanging teacher and student ranks; this does not guarantee equal candidate counts from each model or retain the full union of two top-K sets.

3.3 Stochastic Optimization of the Compact Divergence

Dense full-vocabulary targets are expensive to store and supervise. We therefore keep selected candidates explicit and aggregate the remaining vocabulary into one tail event, yielding a compact approximation to the full-vocabulary objective. For each state s_t, define

$$\mathcal{A}_K(s_t) = \mathcal{V}_K(s_t) \cup \{\texttt{tail}\}, \qquad \bar{p}_d(\texttt{tail} \mid s_t) = 1 - \sum_{v \in \mathcal{V}_K(s_t)} p_d(v \mid s_t), \tag{6}$$

where $d \in \{T, \theta\}$ and $\bar{p}_d(v \mid s_t) = p_d(v \mid s_t)$ for $v \in \mathcal{V}_K(s_t)$. This preserves residual mass but discards token-level structure inside the tail. Let $R = \mathcal{V} \setminus \mathcal{V}_K(s_t)$ and $\tau_d = p_d(R \mid s_t)$. For positive tail masses, the KL chain rule gives, suppressing the state,

$$D_{\mathrm{KL}}(p_T \| p_\theta) - D_{\mathrm{KL}}(\bar{p}_T \| \bar{p}_\theta) = \tau_T D_{\mathrm{KL}}(p_T(\cdot \mid R) \| p_\theta(\cdot \mid R)) \geq 0. \tag{7}$$

Here $p_d(v \mid R) = p_d(v)/\tau_d$. The gap vanishes when the tail conditional distributions agree; small tail mass alone does not guarantee a small gap.

Once $\mathcal{A}_K(s_t)$ is constructed, its divergence contains only $K + 1$ events and can be evaluated exactly. We use Monte Carlo sampling instead to change how events participate in each update.

Using forward KL as an example, the compact divergence can be written equivalently as a weighted sum and as an expectation:

$$D_{\mathrm{FKL}}(\bar{p}_T \| \bar{p}_\theta) = \sum_{a \in \mathcal{A}_K(s_t)} \bar{p}_T(a \mid s_t) \log \frac{\bar{p}_T(a \mid s_t)}{\bar{p}_\theta(a \mid s_t)} = \mathbb{E}_{a \sim \bar{p}_T(\cdot \mid s_t)} \left[\log \frac{\bar{p}_T(a \mid s_t)}{\bar{p}_\theta(a \mid s_t)} \right]. \tag{8}$$

Drawing M events with replacement, $a_1, \ldots, a_M \overset{\text{i.i.d.}}{\sim} \bar{p}_T(\cdot \mid s_t)$, gives

$$\widehat{D}^{\mathrm{MC}}_{\mathrm{FKL}}(s_t) = \frac{1}{M} \sum_{m=1}^{M} \log \frac{\bar{p}_T(a_m \mid s_t)}{\bar{p}_\theta(a_m \mid s_t)}. \tag{9}$$

We use $M = K = 256$ by default and study downstream sensitivity to M in Sec. 4.3. At fixed parameters and event space, $\mathbb{E}[\widehat{D}^{\mathrm{MC}}_{\mathrm{FKL}}] = D_{\mathrm{FKL}}(\bar{p}_T \| \bar{p}_\theta)$. This is value-level unbiasedness, not a guarantee of gradient unbiasedness for student-dependent proposals or lower variance than exact enumeration. Divergence-specific proposals and differentiation are discussed in Appendix A.2. We denote the selected estimator for a divergence family ϕ as $\widehat{D}^{\mathrm{MC}}_{\phi}(s_t; \theta)$.

3.4 Overall Training Objective

The per-state loss is the selected Monte Carlo divergence estimate:

$$\mathcal{L}_{\text{tok}}(s_t;\theta) = \widehat{D}_{\phi}^{\text{MC}}(s_t;\theta). \tag{10}$$

Aggregating over on-policy rollouts with the sequence-level weights from Sec. 3.1, the total objective for a prompt x is:

$$\mathcal{L}(x;\theta) = \sum_{i=1}^{n} w_i \cdot \frac{1}{|y^{(i)}|} \sum_{t=1}^{|y^{(i)}|} \mathcal{L}_{\text{tok}}\big(s_t^{(i)};\theta\big). \tag{11}$$

Each rollout is averaged over its response tokens before weighting; minibatch loss is the mean of these prompt-level objectives. This differs from pooling all weighted tokens under a single length-dependent denominator. Algorithm 1 follows this normalization; divergence-specific differentiation is discussed in Appendix A.2.

4 Experiments

All experiments are implemented in the VeRL framework Sheng et al. (2024) under matched training budgets and consistent environments (Appendix B.1). We evaluate three aspects: generality across dense and MoE teacher–student pairs, compatibility with standard token-level divergence objectives, and ablations that isolate component contributions under truncated-support supervision.

4.1 Experimental Setup

Training Datasets. We build a composite instruction-tuning corpus of approximately 100k samples with a 7:3 mixture of general-purpose and STEM-oriented data (mathematics and code). The general-purpose portion (about 70%) is sourced from open instruction collections such as Tulu-3 SFT Mixture Lambert et al. (2024) and MegaScience Fan et al. (2025), which provide diverse coverage for linguistic fluency and broad world knowledge. The remaining 30% targets mathematical problem solving and program synthesis, filtered from mixed open datasets, including OmniThought Cai et al. (2025) and DeepSeek-R1-Distill Liu et al. (2025). The mixture balances broad capability with reasoning-intensive competencies under a fixed budget; all training data are decontaminated against the evaluation benchmarks to reduce direct overlap.

Evaluation Benchmarks. We employ the EvalScope Team (2024) and OpenCompass Contributors (2023) frameworks to evaluate student performance across 15 benchmarks. To ensure a comprehensive assessment of student capabilities, we categorize these metrics into four core competencies: (1) *Code Generation*, evaluated via LiveCodeBench (Jain et al., 2025), MBPP (Austin et al., 2021), and OpenAI HumanEval (Chen et al., 2021); (2) *Mathematics*, covering diverse difficulty levels including GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021b), and the challenging AIME 2024/2025 (Balunovic et al., 2025); (3) *General Knowledge*, assessed through MMLU (standard and Redux) (Hendrycks et al., 2021a), C-Eval (Huang et al., 2023), CMMLU (Li et al., 2024), and the multilingual MMMLU-ZH (Xuan et al., 2025); and (4) *Complex Reasoning*, utilizing GPQA (standard and super) (Rein et al., 2023) and DROP (Dua et al., 2019) to test deep logical inference beyond simple fact retrieval. We follow the default evaluation protocols provided by EvalScope/OpenCompass and report the standard metrics for each benchmark.

Experimental Framework. To assess generalization across architectures and objectives, we evaluate TCAD across diverse model topologies and divergence measures. *Architectural Paradigms.* We employ two representative teacher-student pairs that cover the primary configurations in modern LLM deployment:

1. *Dense-to-Dense (Qwen3 Series Yang et al. (2025)):* Distilling Qwen3-32B-Instruct into Qwen3-4B-Base represents the standard homogeneous compression setting for dense Transformers.
2. *MoE-to-MoE (DeepSeek Series DeepSeek-AI (2024)):* Distilling the massive DeepSeek-V2.5 (236B total parameters, 21B active) into the compact DeepSeek-V2-Lite-Chat (16B total, 2.4B

Table 1: **Main Results (Macro-Average).** Comparison of distillation performance across two architectural paradigms. We report the average accuracy (%). SFT denotes supervised fine-tuning on teacher demonstrations, RSO denotes rejection sampling optimization, and OPD-FKL/RKL/JSD denote on-policy distillation with forward KL, reverse KL, and Jensen–Shannon objectives. Values in parentheses denote the gap relative to the *Student initialization* for all methods. Bold indicates the best performance within each objective group. Benchmark-level paired t-tests show that improvements are significant across all model-objective settings ($p < 0.01$).

Method	Mathematics	Code Generation	General Knowledge	Complex Reasoning	Average
1. Dense-to-Dense: Teacher (Qwen3-32B-Instruct) → Student (Qwen3-4B-Base)					
Student	27.80	41.43	73.45	48.64	47.83
SFT	36.18 (+8.38)	44.72 (+3.29)	73.98 (+0.53)	48.76 (+0.12)	50.91 (+3.08)
RSO	41.30 (+13.50)	46.60 (+5.17)	73.60 (+0.15)	49.22 (+0.58)	52.68 (+4.85)
OPD-FKL (Base)	40.07 (+12.27)	45.03 (+3.60)	74.59 (+1.14)	49.23 (+0.59)	52.23 (+4.40)
OPD-FKL + **Ours**	**41.68 (+13.88)**	**47.00 (+5.57)**	**76.31 (+2.86)**	**50.69 (+2.05)**	**53.92 (+6.09)**
OPD-RKL (Base)	36.79 (+8.99)	31.59 (-9.84)	72.86 (-0.59)	44.09 (-4.55)	46.33 (-1.50)
OPD-RKL + **Ours**	**41.02 (+13.22)**	**34.08** (-7.35)	**75.18 (+1.73)**	**46.76** (-1.88)	**49.26 (+1.43)**
OPD-JSD (Base)	38.59 (+10.79)	43.87 (+2.44)	75.04 (+1.59)	50.63 (+1.99)	52.03 (+4.20)
OPD-JSD + **Ours**	**40.55 (+12.75)**	**46.25 (+4.82)**	**76.39 (+2.94)**	**51.82 (+3.18)**	**53.75 (+5.92)**
Reference: Teacher	35.75	48.15	84.33	62.98	57.80
2. MoE-to-MoE: Teacher (DeepSeek-V2.5) → Student (DeepSeek-V2-Lite-Chat)					
Student	20.20	25.32	48.19	31.68	31.35
SFT	22.80 (+2.60)	31.10 (+5.78)	51.80 (+3.61)	33.20 (+1.52)	34.73 (+3.38)
RSO	23.60 (+3.40)	35.80 (+10.48)	54.85 (+6.66)	35.25 (+3.57)	37.38 (+6.03)
OPD-FKL (Base)	23.33 (+3.13)	35.42 (+10.10)	54.58 (+6.39)	35.00 (+3.32)	37.08 (+5.73)
OPD-FKL + **Ours**	**24.60 (+4.40)**	**37.95 (+12.63)**	**56.34 (+8.15)**	**35.83 (+4.15)**	**38.68 (+7.33)**
OPD-RKL (Base)	23.57 (+3.37)	36.92 (+11.60)	55.32 (+7.13)	34.32 (+2.64)	37.53 (+6.18)
OPD-RKL + **Ours**	**24.71 (+4.51)**	**38.78 (+13.46)**	**56.46 (+8.27)**	**36.48 (+4.80)**	**39.11 (+7.76)**
OPD-JSD (Base)	23.77 (+3.57)	35.54 (+10.22)	54.20 (+6.01)	34.58 (+2.90)	37.02 (+5.67)
OPD-JSD + **Ours**	**24.78 (+4.58)**	**38.13 (+12.81)**	**55.84 (+7.65)**	**35.62 (+3.94)**	**38.59 (+7.24)**
Reference: Teacher	51.21	69.06	78.54	56.35	63.79

active) evaluates robustness within the sparse mixture-of-experts paradigm, where knowledge is distributed across conditionally activated expert networks.

Complementing these architectural configurations, we benchmark our sequence-level alignment strategy against both supervised and OPD baselines. We include *SFT*, which fine-tunes the student on teacher-generated demonstrations using standard supervised cross-entropy, and *RSO* (rejection sampling optimization), which samples multiple candidate responses and trains on teacher-preferred high-confidence outputs after rejection sampling. Initializing directly from the *Student* checkpoint, we further evaluate performance gains across three standard on-policy distillation objectives: OPD-FKL, OPD-RKL, and OPD-JSD, corresponding to forward KL, reverse KL, and Jensen–Shannon divergence token-level losses. The corresponding *Base* rows denote standard OPD without our sequence-level confidence weighting or support-symmetric Monte Carlo estimator. This controlled setup helps attribute observed differences primarily to the proposed mechanism. Unless otherwise stated, we keep training budgets and hyperparameters consistent across compared methods, and report full implementation details in Appendix B.1.

Evaluation Protocol. For benchmark evaluation, each method is trained with three independent random seeds, producing three independently trained checkpoints. We evaluate each checkpoint with deterministic greedy decoding ($T = 0$, Top-$K = 1$) and a standardized 8,192-token generation context, then average the resulting scores across seeds. The empirical standard deviation of the macro-average across these three checkpoints is below 0.5 percentage points for every evaluated model–objective setting. For comparisons between an OPD baseline and its TCAD variant, we conduct paired tests over benchmark-level scores using the same benchmark instances and evaluation configuration; detailed sample sizes and inference settings are provided in Appendix C.

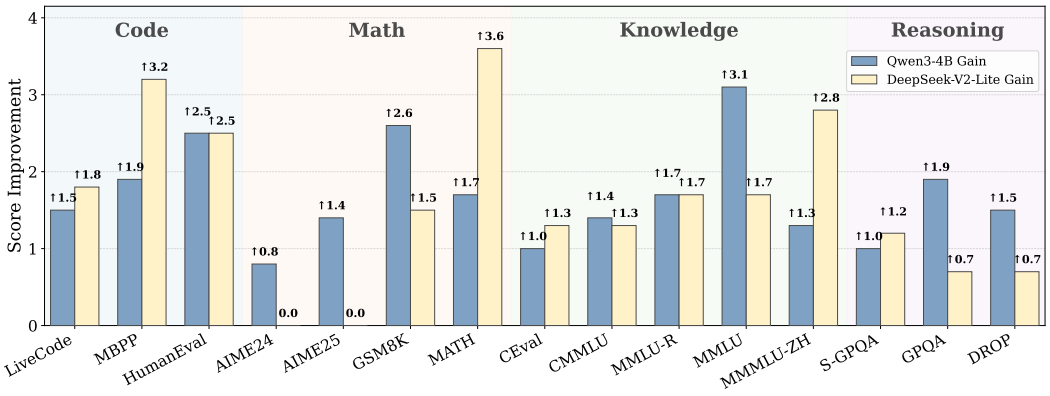

Figure 2: **Performance Improvement Analysis.** Breakdown of domain-level gains for Qwen3-4B-Base and DeepSeek-V2-Lite-Chat. TCAD improves performance across benchmarks, with notable gains on Mathematics and Code Generation.

4.2 Main Results

Table 1 compares TCAD with supervised, rejection-sampling, and standard OPD baselines on the Qwen (Dense) and DeepSeek (MoE) settings.

Consistent Improvement across Diverse Regimes. TCAD yields consistent gains across the evaluated architectures, baselines, and optimization objectives. By prioritizing higher-agreement rollouts and stabilizing truncated-support supervision, our approach improves both dense and MoE distillation settings under the same experimental protocol. Under OPD-FKL, TCAD reaches 53.92% macro-average in the Dense-to-Dense (Qwen) setting, corresponding to a +6.09% gain over the student initialization and a +1.69% gain over the OPD-FKL baseline, with representative gains of +13.88% in Mathematics and +5.57% in Code Generation over the student. In the MoE-to-MoE (DeepSeek) setting, TCAD reaches 38.68% macro-average, improving over the student by +7.33% and over the OPD-FKL baseline by +1.60%, with particularly large gains in Code Generation (+12.63%) and Mathematics (+4.40%). The same trend holds against additional baselines: TCAD improves over RSO by +1.24% on Qwen and +1.30% on DeepSeek in macro-average, while also outperforming SFT by a clear margin in both settings.

Beyond OPD-FKL, the gains persist across other token-level alignment losses, suggesting that TCAD is compatible with multiple OPD objectives. For example, under OPD-JSD, TCAD reaches 46.25% in Qwen Code Generation (+4.82% over the student) and 76.39% in General Knowledge (+2.94%). This supports the compatibility of support construction and stochastic compact optimization with truncated supervision when full-vocabulary target materialization is computationally expensive. OPD-RKL can be more sensitive to initialization and on-policy dynamics: the Qwen OPD-RKL baseline degrades, whereas the DeepSeek setting remains stable. In such cases, TCAD mitigates degradation relative to the OPD-RKL baseline, e.g., by +2.49% in Qwen Code Generation, indicating improved robustness without claiming that reverse KL is uniformly preferable.

4.3 Analysis and Ablation Studies

We study component ablations, support strategies, within-query confidence ranking, MC sample-count sensitivity, and hyperparameter/efficiency trade-offs; an illustrative qualitative case study is deferred to the appendix. Unless otherwise stated, all analyses use the *Qwen3-4B-Base* student with the *OPD-FKL* objective.

Component Necessity. First, we isolate the gains attributable to sequence-level confidence weighting and Monte Carlo sampling while keeping the same support construction. We denote the complete framework as *Ours* and contrast it with two variants: *w/o TC.*, which excludes sequence-level confidence reweighting, and *w/o MC Sampling*, which replaces stochastic event sampling with the deterministic compact divergence on the selected support. Table 2 (Top Block) shows that teacher-confidence weighting improves the macro-average from 52.47% to 53.92%, while Monte Carlo

sampling improves it from 52.35% to 53.92%. Because each variant removes a different component from the same recipe, these numbers are component-level evidence rather than additive causal effects; Sec. 4.3 further examines the MC sample count.

Table 2: **Ablation Analysis on Qwen3-4B (Averaged Accuracy, %).** Top: effects of teacher-confidence weighting and Monte Carlo sampling. Bottom: impact of support construction strategies. Parentheses denote the drop relative to *Our Method*.

Setting	Code Generation	Mathematics	Knowledge	Reasoning	Average
Sequence-Level Weighting and MC Sampling					
Our Method	**47.00**	**41.68**	**76.31**	**50.69**	**53.92**
w/o Teacher Conf.	45.30 (-1.70)	40.29 (-1.39)	74.88 (-1.43)	49.41 (-1.28)	52.47 (-1.45)
w/o MC Sampling	45.15 (-1.85)	40.17 (-1.51)	74.69 (-1.62)	49.39 (-1.30)	52.35 (-1.57)
Support Construction Strategy					
Dual-Union (Ours)	**47.00**	**41.68**	**76.31**	**50.69**	**53.92**
Teacher-Only	43.68 (-3.32)	39.79 (-1.89)	75.05 (-1.26)	50.00 (-0.69)	52.13 (-1.79)
Student-Only	45.23 (-1.77)	39.91 (-1.77)	75.00 (-1.31)	49.82 (-0.87)	52.49 (-1.43)

Impact of Support Strategy. A key hypothesis of our work is that standard *Teacher-Only* truncation can introduce a mismatch in on-policy distillation by under-covering tokens that are highly probable under the student distribution. To examine this, we compare our *Dual-Union* strategy against two unidirectional variants: *Teacher-Only* (standard Top-K) and *Student-Only*, while keeping the confidence weighting and Monte Carlo estimator fixed. Results in Table 2 (Bottom Block) are consistent with this hypothesis. Notably, the *Teacher-Only* support underperforms in Code Generation (43.68% vs. 47.00%). *Student-Only* support improves over Teacher-Only in Code (45.23%), suggesting that accounting for the student's high-probability region can be beneficial under truncation. Overall, *Dual-Union* achieves the best performance across domains, supporting the use of a symmetric support that covers both teacher and student preferences for on-policy alignment.

Table 3: **Teacher confidence as a within-prompt ranking signal.** We sample 100 prompts and 8 student rollouts per prompt (800 rollouts total), retain prompts containing both correct and incorrect responses, and compute all statistics within each prompt. Metric symbols are defined in the second column.

Metric	Interpretation	Result
r	Global point-biserial correlation between teacher confidence and correctness.	0.5573
ρ	Mean within-query Spearman correlation between confidence and correctness.	0.5431
P	Mixed groups with positive within-query correlation.	88.56%
A	Within-query pairwise accuracy: a correct rollout receives a higher score than an incorrect rollout.	81.77%

Table 4: **MC sample-count sensitivity.** At fixed $K = 256$, we report downstream macro-average scores (%) and the mean state-wise cosine similarity between MC and exact compact FKL gradients over 100 matched states at the common initialization (Appendix D.3).

Optimizer	K	M	Score	Grad. cosine
Exact compact	256	–	52.35	1.00
MC	256	64	52.94	0.72
MC	256	128	53.26	0.81
MC	256	256	**53.92**	0.90
MC	256	512	53.69	0.96

Teacher-confidence probe. Because TCAD uses teacher likelihood to rank rollouts from the same query rather than as a calibrated correctness estimator, we conduct a within-query probe with 100 prompts and 8 student rollouts per prompt. The teacher score shows positive association with correctness ($r = 0.5573$; mean within-query $\rho = 0.5431$), with positive correlation in $P = 88.56\%$ of mixed groups and pairwise ranking accuracy $A = 81.77\%$. These results support teacher confidence as a useful relative ranking signal, while the imperfect ranking accuracy motivates soft rather than binary weighting. The full protocol and statistics are reported in Appendix D.1.

MC Sample-Count Sensitivity. For a fixed compact event space, exact enumeration provides a deterministic reference, while Monte Carlo optimization evaluates the same compact objective from a finite number of probability-proportional samples. We compare the two choices and sweep the number of samples M while keeping $K = 256$ fixed. Table 4 shows that the score increases from 52.94 at $M = 64$ to 53.92 at $M = 256$, compared with 52.35 for exact enumeration. Doubling M to 512 yields 53.69, with no additional downstream gain in this sweep. The initialization-time mean state-wise gradient cosine nevertheless increases from 0.72 to 0.96 as M grows, indicating that larger sample counts more closely reproduce the exact compact gradient direction under matched states. Thus, $M = 256$ provides the best observed downstream score, while larger M improves gradient fidelity without improving the final score in this setting. We use $M = 256$ as a sufficient default, without claiming it is universally optimal.

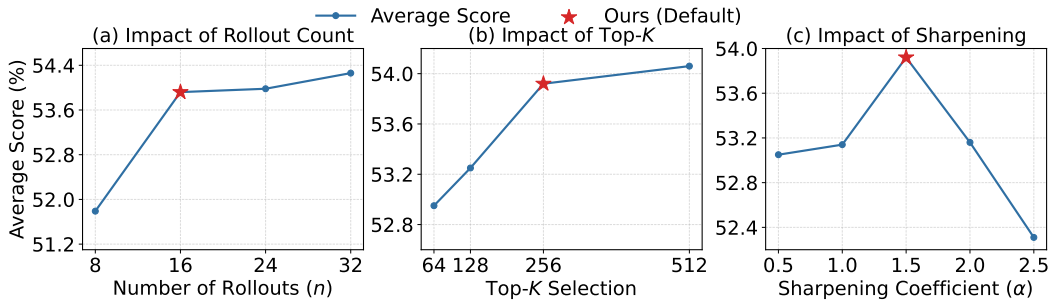

Figure 3: **Hyperparameter sensitivity analysis.** We report the impact of rollout count n, truncation size K, and sharpening coefficient α on average performance. Red stars indicate default configuration.

Hyperparameter Sensitivity and Efficiency. Figure 3 summarizes sensitivity to rollout count n, support size K, and sharpening coefficient α. In the evaluated range, larger n and K show diminishing returns, while $\alpha = 1.5$ gives the highest score. We balance these gains against latency (Appendix Table 5): $(n = 16, K = 256)$ requires 115s per step, compared with 265s (2.3×) for $n = 32$ and 170s (48% overhead) for $K = 512$. We therefore use $(n = 16, K = 256, \alpha = 1.5)$ as the default configuration.

5 Conclusion

In this paper, we proposed TCAD to address two practical challenges in on-policy distillation: indiscriminate imitation from noisy on-policy rollouts and scalable approximation of token-level alignment under large vocabularies. TCAD combines (1) a reward-free, sequence-level reweighting scheme based on relative teacher confidence, which down-weights low-agreement trajectories without auxiliary verifiers or step-wise teacher intervention, and (2) support-symmetric candidate construction with stochastic optimization of a compact support-plus-tail divergence, which mitigates teacher–student support mismatch while preserving residual probability mass under vocabulary truncation. Across dense and MoE teacher–student settings and multiple divergence objectives, TCAD improves distillation performance under practical post-training constraints. We substantiate these improvements through component ablations, confidence-ranking probes, MC sample-count and gradient-alignment diagnostics, efficiency and sensitivity studies, and qualitative case studies. In particular, the confidence probe supports teacher confidence as a within-query ranking signal, while the MC analysis identifies $M = 256$ as the best observed downstream setting and shows that larger sample counts improve initialization-time gradient fidelity without improving the final score in this setting.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

Kushal Arora, Layla El Asri, Hareesh Bahuleyan, and Jackie Chi Kit Cheung. Why exposure bias matters: An imitation learning perspective of error accumulation in language generation. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Findings of the Association for Computational Linguistics: ACL 2022, Dublin, Ireland, May 22-27, 2022*, pp. 700–710. Association for Computational Linguistics, 2022. doi: 10.18653/V1/2022.FINDINGS-ACL.58.

Jacob Austin, Augustus Odena, Maxwell I. Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie J. Cai, Michael Terry, Quoc V. Le, and Charles Sutton. Program synthesis with large language models. *CoRR*, abs/2108.07732, 2021.

Mislav Balunovic, Jasper Dekoninck, Ivo Petrov, Nikola Jovanovic, and Martin T. Vechev. Matharena: Evaluating llms on uncontaminated math competitions. *CoRR*, abs/2505.23281, 2025. doi: 10.48550/ARXIV.2505.23281.

Wenrui Cai, Chengyu Wang, Junbing Yan, Jun Huang, and Xiangzhong Fang. Reasoning with omnithought: A large cot dataset with verbosity and cognitive difficulty annotations. *CoRR*, abs/2505.10937, 2025. doi: 10.48550/ARXIV.2505.10937.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Joshua Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *CoRR*, abs/2107.03374, 2021.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168, 2021.

OpenCompass Contributors. Opencompass: A universal evaluation platform for foundation models. `https://github.com/open-compass/opencompass`, 2023.

DeepSeek-AI. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *CoRR*, abs/2405.04434, 2024. doi: 10.48550/ARXIV.2405.04434.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pp. 2368–2378. Association for Computational Linguistics, 2019. doi: 10.18653/V1/N19-1246.

Run-Ze Fan, Zengzhi Wang, and Pengfei Liu. Megascience: Pushing the frontiers of post-training datasets for science reasoning. *CoRR*, abs/2507.16812, 2025. doi: 10.48550/ARXIV.2507.16812.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. Minillm: Knowledge distillation of large language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021a.

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 Joaquin Vanschoren and Sai-Kit Yeung (eds.), *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*, 2021b.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531, 2015.

Yuzhen Huang, Yuzhuo Bai, Zhihao Zhu, Junlei Zhang, Jinghan Zhang, Tangjun Su, Junteng Liu, Chuancheng Lv, Yikai Zhang, Jiayi Lei, Yao Fu, Maosong Sun, and Junxian He. C-eval: A multi-level multi-discipline chinese evaluation suite for foundation models. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/c6ec1844bec96d6d32ae95ae694e23d8-Abstract-Datasets_and_Benchmarks.html.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025.

Sébastien Jean, KyungHyun Cho, Roland Memisevic, and Yoshua Bengio. On using very large target vocabulary for neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pp. 1–10. The Association for Computer Linguistics, 2015. doi: 10.3115/V1/P15-1001.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tülu 3: Pushing frontiers in open language model post-training. 2024.

Haonan Li, Yixuan Zhang, Fajri Koto, Yifei Yang, Hai Zhao, Yeyun Gong, Nan Duan, and Timothy Baldwin. CMMLU: measuring massive multitask language understanding in chinese. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, volume ACL 2024 of *Findings of ACL*, pp. 11260–11285. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.FINDINGS-ACL.671.

Cong Liu, Zhong Wang, ShengYu Shen, Jialiang Peng, Xiaoli Zhang, ZhenDong Du, and YaFang Wang. The chinese dataset distilled from deepseek-r1-671b. https://huggingface.co/datasets/Congliu/Chinese-DeepSeek-R1-Distill-data-110k, 2025.

OpenAI. GPT-4 technical report. *CoRR*, abs/2303.08774, 2023. doi: 10.48550/ARXIV.2303.08774.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. 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 F. Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof q&a benchmark. *CoRR*, abs/2311.12022, 2023. doi: 10.48550/ARXIV.2311.12022.

Stéphane Ross, Geoffrey J. Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In Geoffrey J. Gordon, David B. Dunson, and Miroslav Dudík (eds.), *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011*, volume 15 of *JMLR Proceedings*, pp. 627–635. JMLR.org, 2011.

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. *arXiv preprint arXiv: 2409.19256*, 2024.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. Stanford alpaca: An instruction-following llama model, 2023.

Gemma Team. Gemma 3 technical report. *CoRR*, abs/2503.19786, 2025. doi: 10.48550/ARXIV.2503.19786.

ModelScope Team. EvalScope: Evaluation framework for large models, 2024. URL `https://github.com/modelscope/evalscope`.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models. *CoRR*, abs/2302.13971, 2023. doi: 10.48550/ARXIV.2302.13971.

Bangjun Xiao, Bingquan Xia, Bo Yang, Bofei Gao, Bowen Shen, Chen Zhang, Chenhong He, Chiheng Lou, Fuli Luo, Gang Wang, et al. Mimo-v2-flash technical report. *arXiv preprint arXiv:2601.02780*, 2026.

Wenda Xu, Rujun Han, Zifeng Wang, Long T. Le, Dhruv Madeka, Lei Li, William Yang Wang, Rishabh Agarwal, Chen-Yu Lee, and Tomas Pfister. Speculative knowledge distillation: Bridging the teacher-student gap through interleaved sampling. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025.

Weihao Xuan, Rui Yang, Heli Qi, Qingcheng Zeng, Yunze Xiao, Yun Xing, Junjue Wang, Huitao Li, Xin Li, Kunyu Yu, Nan Liu, Qingyu Chen, Douglas Teodoro, Edison Marrese-Taylor, Shijian Lu, Yusuke Iwasawa, Yutaka Matsuo, and Irene Li. Mmlu-prox: A multilingual benchmark for advanced large language model evaluation. *CoRR*, abs/2503.10497, 2025. doi: 10.48550/ARXIV.2503.10497.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jian Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *CoRR*, abs/2505.09388, 2025. doi: 10.48550/ARXIV.2505.09388.

A Additional Preliminaries and MC Estimation Details

In this section, we introduce the notation and formalize (i) on-policy token-level distillation for autoregressive LMs, and (ii) truncated-support objectives that are commonly used for scalability. We further review a generic sequence-level mechanism based on power transformations, which will be instantiated in our method to prioritize high-confidence trajectories.

A.1 On-Policy Distillation

We consider on-policy distillation for autoregressive language models. Let π_T denote a fixed teacher policy and π_θ a student policy parameterized by θ. Given a prompt $x \sim \mathcal{D}$, the student samples a response $y = (y_1, \ldots, y_{|y|}) \sim \pi_\theta(\cdot \mid x)$, which induces a sequence of generation states $s_t = (x, y_{<t})$. The standard on-policy distillation objective seeks to minimize a divergence between the teacher and student distributions over the vocabulary $\mathcal{V}$ at every generated state.

Computational Infeasibility at Scale. However, directly matching the full-vocabulary distributions can be computationally infeasible in large-scale post-training, particularly under on-policy settings where supervision must be computed for dynamically generated prefixes and often across multiple rollouts. For a state s_t, let $p_T(\cdot \mid s_t)$ and $p_\theta(\cdot \mid s_t)$ denote the teacher and student token distributions over the vocabulary $\mathcal{V}$. The full forward KL objective is

$$D_{\mathrm{FKL}}(p_T \| p_\theta) = \mathbb{E}_{v \sim p_T(\cdot \mid s_t)} \left[\log \frac{p_T(v \mid s_t)}{p_\theta(v \mid s_t)} \right], \tag{12}$$

with analogous expectation forms for reverse KL and Jensen–Shannon divergence. This expectation view suggests a natural Monte Carlo estimator, but full-vocabulary supervision remains costly. We therefore construct a compact event space from a selected token set while preserving the total mass of unselected tokens; Monte Carlo sampling is then applied after this compact space has been formed. Specifically, for any distribution p, define

$$\bar{p}(a \mid s_t) = \begin{cases} p(a \mid s_t), & a \in \mathcal{V}_K(s_t), \\ 1 - \sum_{v \in \mathcal{V}_K(s_t)} p(v \mid s_t), & a = \texttt{tail}, \end{cases} \tag{13}$$

over the compact event space $\mathcal{A}_K(s_t) = \mathcal{V}_K(s_t) \cup \{\texttt{tail}\}$. The `tail` event is only a bookkeeping device for the aggregate probability of the unselected vocabulary; it avoids discarding out-of-support mass without expanding every low-probability token individually.

In the on-policy setting, the token-level divergence is evaluated on states sampled from the student itself. The resulting training objective can be written as

$$\mathcal{J}_{\mathrm{KD}}(\theta) = \mathbb{E}_{x \sim \mathcal{D}} \Big[\mathbb{E}_{y \sim \pi_\theta(\cdot \mid x)} \big[\tfrac{1}{|y|} \sum_{t=1}^{|y|} \widehat{D}_\phi^{\mathrm{MC}}(s_t; \theta) \big] \Big]. \tag{14}$$

Here $\widehat{D}_\phi^{\mathrm{MC}}$ denotes a Monte Carlo estimate of the chosen divergence family on $\mathcal{A}_K(s_t)$. For a fixed compact event space, the sample average estimates the corresponding event-space divergence; relative to the original full-vocabulary divergence, it remains a compact approximation that becomes tighter as $\mathcal{V}_K(s_t)$ expands. This motivates a support construction that accounts for both teacher and student preferences and a probability-proportional optimizer whose sample-count trade-off can be studied separately from the approximation induced by support truncation.

A.2 Monte Carlo Objectives for Different Divergences

For completeness, we detail how the same compact event space supports the divergences used in our experiments. For forward KL, samples are drawn from the teacher-side compact distribution $a_m \sim \bar{p}_T(\cdot \mid s_t)$:

$$\widehat{D}_{\mathrm{FKL}}^{\mathrm{MC}}(s_t) = \frac{1}{M} \sum_{m=1}^{M} \log \frac{\bar{p}_T(a_m \mid s_t)}{\bar{p}_\theta(a_m \mid s_t)}. \tag{15}$$

For reverse KL, samples are drawn from the student-side compact distribution $a_m \sim \bar{p}_\theta(\cdot \mid s_t)$:

$$\widehat{D}_{\mathrm{RKL}}^{\mathrm{MC}}(s_t) = \frac{1}{M} \sum_{m=1}^{M} \log \frac{\bar{p}_\theta(a_m \mid s_t)}{\bar{p}_T(a_m \mid s_t)}. \tag{16}$$

For Jensen–Shannon divergence, let $\bar{m} = \frac{1}{2}(\bar{p}_T + \bar{p}_\theta)$ and draw $a_m \sim \bar{p}_T$, $b_m \sim \bar{p}_\theta$:

$$\widehat{D}^{\mathrm{MC}}_{\mathrm{JSD}}(s_t) = \frac{1}{2M}\sum_{m=1}^{M} \log \frac{\bar{p}_T(a_m \mid s_t)}{\bar{m}(a_m \mid s_t)} + \frac{1}{2M}\sum_{m=1}^{M} \log \frac{\bar{p}_\theta(b_m \mid s_t)}{\bar{m}(b_m \mid s_t)}. \tag{17}$$

Each estimator is the sample mean of the integrand under its corresponding proposal distribution; therefore, conditioned on a fixed $\mathcal{A}_K(s_t)$, its expectation equals the compact divergence on that event space.

Proposition A.1 (Estimator property on the compact event space)**.** *For a fixed state s_t and compact event space $\mathcal{A}_K(s_t)$, the Monte Carlo estimators in Eqs. equation 15–equation 17 are unbiased estimators of their corresponding compact divergences over $\bar{p}_T$ and $\bar{p}_\theta$. They are not unbiased estimators of the original full-vocabulary divergences; the approximation gap depends on the probability mass and distributional structure of tokens omitted from $\mathcal{V}_K(s_t)$, as well as the information lost by aggregating that omitted region into a single tail event.*

Proof. Each estimator is an empirical mean of i.i.d. samples from the proposal distribution used in the corresponding expectation form, so linearity of expectation gives the compact divergence exactly. The second claim follows because replacing the omitted vocabulary by `tail` is a many-to-one projection that preserves total residual mass but discards token-level ratios inside the omitted region. □

Value estimates and parameter gradients. The proposition is a statement about divergence values at fixed parameters and event space, not about differentiation through sampled indices or support selection. For a fixed support, a student-dependent proposal obeys

$$\nabla_\theta \mathbb{E}_{a\sim q_\theta}[f_\theta(a)] = \mathbb{E}_{a\sim q_\theta}\left[\nabla_\theta f_\theta(a) + f_\theta(a)\nabla_\theta \log q_\theta(a)\right]. \tag{18}$$

Differentiating only the sampled integrand generally omits the second term. For FKL with fixed support, the teacher proposal is independent of θ and this term vanishes. RKL and the student-side term of JSD require accounting for proposal dependence, for example via a score-function estimator or an equivalent surrogate, to obtain the gradient of the compact divergence. The value estimators above alone do not specify such a gradient implementation. These conditional statements also do not include derivatives through discrete support selection or the on-policy rollout distribution.

A.3 Power Transformations of Sequence Distributions

To formalize sequence-level sharpening without an additional verifier, we review *power transformations*.

For a sharpening coefficient $\alpha \geq 1$, the power distribution is defined by the proportionality:

$$p^{(\alpha)}(y \mid x) \propto p(y \mid x)^\alpha. \tag{19}$$

For $\alpha > 1$ this transformation increases probability ratios between higher- and lower-probability candidates; for $\alpha = 1$ it is unchanged. Specifically, for any two sequences y, y' where the model prefers y (i.e., $p(y \mid x) > p(y' \mid x)$), the likelihood ratio under the power distribution satisfies:

$$\frac{p^{(\alpha)}(y \mid x)}{p^{(\alpha)}(y' \mid x)} = \left(\frac{p(y \mid x)}{p(y' \mid x)}\right)^\alpha \geq \frac{p(y \mid x)}{p(y' \mid x)}. \tag{20}$$

Equality holds at $\alpha = 1$, and the inequality is strict for $\alpha > 1$. In TCAD, define $g_i = \exp(\bar{\ell}_T(x, y^{(i)})) = \pi_T(y^{(i)} \mid x)^{1/|y^{(i)}|}$. Then $w_i = g_i^\alpha / \sum_j g_j^\alpha$ and $w_i/w_j = \exp(\alpha(\bar{\ell}_i - \bar{\ell}_j))$. This is an exact power transformation of the geometric-mean likelihoods on the sampled candidate set, not of sequence probabilities when response lengths differ. Any $\alpha > 0$ preserves the score ordering; values below one soften the weights and values above one sharpen them.

B Training Algorithm and Implementation Details

This appendix gives the complete training procedure and implementation details for TCAD. The algorithm first computes within-prompt teacher-confidence weights for sampled rollouts and then optimizes the compact token-level divergence with probability-proportional Monte Carlo sampling.

Algorithm 1 On-Policy Distillation with Support-Symmetric Top-K and Stochastic Compact Optimization

Input: dataset $\mathcal{D}$; teacher π_T; student π_θ
Input: top-K size K; rollouts n; MC samples M (default $M = K$)
Input: divergence family D_ϕ (FKL/RKL/JSD); sharpening coefficient α
Output: updated student parameters θ

for each training iteration **do**
  Sample a minibatch $\mathcal{B}$ from $\mathcal{D}$; $\mathcal{L}_\mathcal{B} \leftarrow 0$
  for each prompt x **do**
    Sample n responses $\{y^{(i)}\}_{i=1}^{n} \sim \pi_\theta(\cdot \mid x)$
  // Compute within-prompt relative confidence weights
    for $i = 1$ to n **do**
      $\bar{\ell}_i \leftarrow \frac{1}{|y^{(i)}|} \sum_{t=1}^{|y^{(i)}|} \log p_T\left(y_t^{(i)} \mid x, y_{1:t-1}^{(i)}\right)$
    end for
    $w_i \leftarrow \frac{\exp(\alpha\, \bar{\ell}_i)}{\sum_{j=1}^{n} \exp(\alpha\, \bar{\ell}_j)}$ for $i = 1, \ldots, n$
    Treat w_i as stop-gradient coefficients
  // Average each rollout, then apply its confidence weight
    $\mathcal{L} \leftarrow 0$
    for $i = 1$ to n **do**
      $\mathcal{L}_i \leftarrow 0$
      for each token position t in $y^{(i)}$ **do**
        $s \leftarrow (x, y_{1:t-1}^{(i)})$ ▷ generation state
        $\mathcal{V}_K(s) \leftarrow \mathrm{TopK}_{v \in \mathcal{V}}\left(\mathcal{M}(v|s), K\right)$
        Build $\mathcal{A}_K(s) = \mathcal{V}_K(s) \cup \{\texttt{tail}\}$ with compact distributions $\bar{p}_T, \bar{p}_\theta$
        Draw M events per proposal with replacement (Appendix A.2)
        $\widehat{D}_\phi^{\mathrm{MC}} \leftarrow \mathrm{MCEstimate}(\bar{p}_T, \bar{p}_\theta, \mathcal{A}_K(s), M)$
        $\mathcal{L}_i \leftarrow \mathcal{L}_i + \widehat{D}_\phi^{\mathrm{MC}}$
      end for
      $\mathcal{L} \leftarrow \mathcal{L} + w_i \cdot \mathcal{L}_i / |y^{(i)}|$
    end for
    $\mathcal{L}_\mathcal{B} \leftarrow \mathcal{L}_\mathcal{B} + \mathcal{L}/|\mathcal{B}|$
  end for
  Update θ using the minibatch objective $\mathcal{L}_\mathcal{B}$
end for

B.1 Computing Infrastructure and Framework

All experiments were conducted on a high-performance computing cluster equipped with NVIDIA H100 Tensor Core GPUs. Our implementation utilizes VeRL, a scalable library for Large Language Model post-training optimized with Ray. To handle the computational demand of on-policy generation, we adopted a decoupled actor-learner architecture. The student policy was deployed on a single node with 8 $\times$ H100 GPUs using Fully Sharded Data Parallelism, while the teacher policy was distributed across 2 nodes (16 $\times$ H100 GPUs) using Tensor Parallelism for rollout generation ($n = 16$) and likelihood scoring.

B.2 Model Specifications

We employed two teacher–student pairs covering dense and sparse architectural paradigms. For the Dense-to-Dense setting, we distilled `Qwen3-32B-Instruct` into `Qwen3-4B-Base`. For

Table 5: **Training Efficiency Profile.** Latency under different rollout counts and truncation sizes. The default setting ($n = 16, K = 256$) balances cost and performance.

Rollout (n)	**Top-**K	**Latency (s/step)**	**Cost Factor**
8	128	77	0.67×
16	64	96	0.83×
16 (Ours)	256	115	1.00×
16	512	170	1.48×
24	256	170	1.48×
32	256	265	2.30×

the MoE-to-MoE setting, `DeepSeek-V2.5` (236B total parameters, 21B active) served as teacher and `DeepSeek-V2-Lite-Chat` (16B total, 2.4B active) as student. Teachers were frozen, and students were initialized from their respective pretrained checkpoints.

B.3 Training Configuration

We maintained consistent hyperparameters across distillation comparisons. Students were optimized with AdamW and a cosine learning-rate schedule. The maximum learning rate was 1×10^{-6}, with global batch size 32 and maximum sequence length 4096 tokens (2048 for prompts and 2048 for responses). The training data contains approximately 100k samples with a 70%/30% mixture of general-purpose and mathematics/code data; all training data are decontaminated against the evaluation benchmarks. We use the checkpoint saved at 500 steps as a matched-compute operating point for all methods.

For TCAD, we used rollout count $n = 16$, sampling temperature $T = 1.0$, Top-$p = 0.95$, truncation size $K = 256$, and sharpening coefficient $\alpha = 1.5$. Monte Carlo samples were drawn with replacement from the compact event space, using $M = K = 256$. Under the same rollout and sequence-length configuration, dense full-vocabulary targets exceeded available memory; the compact support-plus-tail representation therefore provides a memory-feasibility and target-storage benefit rather than eliminating teacher forward inference or top-K extraction.

C Evaluation Protocol and Detailed Results

We conducted all evaluations with **EvalScope** and **OpenCompass**, using **SGLang** as the inference backend.

C.1 Inference Setup

For all evaluation tasks, we used deterministic greedy decoding ($T = 0$, Top-$K = 1$), a maximum generation context of 8,192 tokens, and evaluation batch size 128. Each method was trained with three independent random seeds, and each seed produced a separately trained checkpoint that was evaluated under this deterministic decoding protocol. Main-text metrics are averages across these three training seeds. For significance testing, we use paired t-tests over benchmark-level scores, pairing each OPD base objective with its TCAD variant. Explicit thinking or internal chain-of-thought features were disabled during inference.

C.2 Benchmark Statistics

We evaluated 15 benchmarks across code generation, mathematics, general knowledge, and complex reasoning. The dataset-level sample sizes are listed in Table 6.

Table 6: **Benchmark Statistics.** The number of evaluation samples used for each dataset.

Domain	Dataset	Sample Size
Code Generation	OpenAI HumanEval	1,773
	LiveCodeBench	1,000
	MBPP	500
Mathematics	MATH	5,000
	GSM8K	1,320
	AIME 2024	480
	AIME 2025	480
General Knowledge	MMLU	3,000
	MMMLU (ZH-CN)	3,000
	MMLU-Redux	1,000
	C-Eval	1,000
	CMMLU	1,000
Complex Reasoning	DROP	3,000
	GPQA	1,192
	Super GPQA	1,000

C.3 Detailed Evaluation Results

Table 7 provides the complete benchmark-level results for the Dense-to-Dense (Qwen) and MoE-to-MoE (DeepSeek) settings, complementing the macro-average results in the main text.

Table 7: **Full Breakdown of Results on Qwen and DeepSeek Benchmarks.** Detailed comparison of Qwen3-4B-Base (top) and DeepSeek-V2-Lite-Chat (bottom) students trained via OPD-FKL/RKL/JSD objectives. We report accuracy (%). **Bold** indicates the best student performance. Benchmark-level paired t-tests show that improvements are significant across all model-objective settings ($p < 0.01$).

Domain	Benchmark	Student	OPD-FKL		OPD-RKL		OPD-JSD		Teacher
			Base	+ Ours	Base	+ Ours	Base	+ Ours	(Reference)
Part I: Qwen Results (Dense-to-Dense) — Teacher: Qwen3-32B-Instruct									
Code	LiveCodeBench	38.43	44.10	**45.61**	39.10	41.35	42.87	45.30	46.47
	MBPP	3.33	8.47	**10.37**	5.67	6.56	5.00	7.23	4.27
	HumanEval	82.52	82.52	85.02	50.00	54.33	83.74	**86.22**	93.70
Domain Average: Code		*41.43*	*45.03*	***47.00***	*31.59*	*34.08*	*43.87*	*46.25*	*48.15*
Math	AIME 2024	22.99	29.79	30.55	29.17	31.58	31.67	**33.01**	32.46
	AIME 2025	14.37	23.20	24.59	22.92	**27.61**	20.83	23.94	22.85
	GSM8K	33.23	52.26	**54.84**	39.35	47.23	48.02	49.63	44.40
	MATH	40.60	55.03	56.74	55.72	**57.66**	53.83	55.62	43.28
Domain Average: Math		*27.80*	*40.07*	***41.68***	*36.79*	*41.02*	*38.59*	*40.55*	*35.75*
Knowl.	C-Eval	70.93	73.70	**74.75**	72.40	74.33	73.30	74.32	86.13
	CMMLU	72.83	73.07	74.50	71.23	73.71	73.53	**75.21**	85.97
	MMLU-Redux	79.37	80.73	**82.42**	76.33	79.24	79.53	80.67	88.20
	MMLU	77.39	77.06	80.19	77.33	79.54	79.34	**80.62**	88.74
	MMMLU-ZH	66.73	68.37	69.69	67.03	69.08	69.50	**71.13**	72.60
Domain Average: Knowl.		*73.45*	*74.59*	*76.31*	*72.86*	*75.18*	*75.04*	***76.39***	*84.33*
Reas.	Super GPQA	30.50	30.60	31.62	23.90	28.54	30.97	**32.13**	45.87
	GPQA	30.11	31.26	33.15	23.56	25.61	34.15	**35.56**	52.35
	DROP	85.30	85.83	87.30	84.80	86.13	86.78	**87.77**	90.71
Domain Average: Reas.		*48.64*	*49.23*	*50.69*	*44.09*	*46.76*	*50.63*	***51.82***	*62.98*
Part II: DeepSeek Results (MoE-to-MoE) — Teacher: DeepSeek-V2.5									
Code	LiveCodeBench	6.92	8.33	**10.15**	9.07	10.12	8.87	9.69	41.60
	MBPP	45.84	47.13	**50.34**	47.02	48.80	46.53	49.51	79.00
	HumanEval	23.20	50.81	53.36	54.67	**57.42**	51.22	55.19	86.59
Domain Average: Code		*25.32*	*35.42*	*37.95*	*36.92*	***38.78***	*35.54*	*38.13*	*69.06*
Math	AIME 2024	0.00	0.00	0.00	0.00	0.00	0.00	0.00	23.75
	AIME 2025	0.00	0.00	0.00	0.00	0.00	0.00	0.00	20.21
	GSM8K	59.95	68.62	70.15	68.98	70.35	69.80	**70.86**	89.76
	MATH	20.86	24.70	28.25	25.30	**28.49**	25.27	28.26	71.10
Domain Average: Math		*20.20*	*23.33*	*24.60*	*23.57*	*24.71*	*23.77*	***24.78***	*51.21*
Knowl.	C-Eval	53.02	54.73	56.01	57.10	**57.10**	54.00	56.45	78.60
	CMMLU	58.64	59.53	**60.80**	60.13	60.38	58.10	60.26	79.80
	MMLU-Redux	36.40	52.07	**53.79**	50.03	52.75	50.67	52.18	80.40
	MMLU	47.45	55.49	**57.20**	54.64	56.76	55.88	56.68	81.32
	MMMLU-ZH	45.44	51.10	53.90	54.70	**55.31**	52.33	53.63	72.60
Domain Average: Knowl.		*48.19*	*54.58*	*56.34*	*55.32*	***56.46***	*54.20*	*55.84*	*78.54*
Reas.	Super GPQA	10.40	14.27	15.43	14.47	**15.79**	14.63	15.65	35.10
	GPQA	23.46	24.41	25.08	20.91	**27.52**	23.58	24.81	45.93
	DROP	61.17	66.33	66.98	**67.57**	66.13	65.53	66.40	88.03
Domain Average: Reas.		*31.68*	*35.00*	*35.83*	*34.32*	***36.48***	*34.58*	*35.62*	*56.35*

D Additional Mechanism Diagnostics

D.1 Teacher-confidence Probe

To test the within-query ranking assumption in Eq. equation 3, we sample 100 prompts and generate 8 student rollouts per prompt, yielding 800 rollouts in total. We retain mixed groups containing both correct and incorrect responses, determine correctness using task-specific reward-derived supervision, and rescore each fixed rollout with the same length-normalized teacher log-likelihood used by TCAD. All statistics are computed within each query rather than globally.

The complete results are reported in Table 3 in the main text. This appendix section records the evaluation protocol and clarifies that all statistics are computed within each query rather than across queries.

D.2 Training Dynamics

We report training-loss trajectories for the Qwen3-4B-Base and DeepSeek-V2-Lite-Chat students in Figure 4. The solid lines represent TCAD and the dashed lines represent the OPD-FKL baseline. These curves are descriptive because the two methods use different sequence-level weighting objectives.

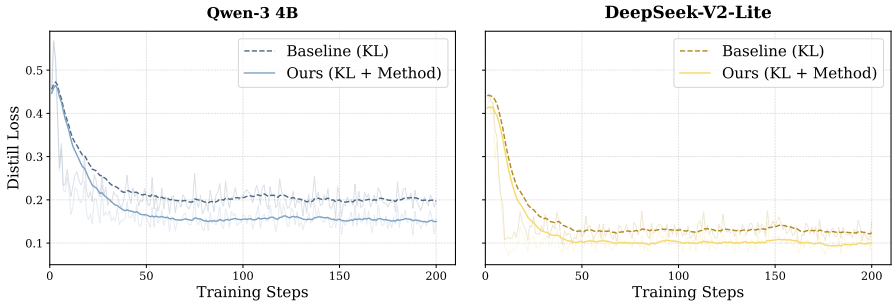

Figure 4: **Training loss trajectories.** Training-loss curves for Qwen3-4B-Base (left) and DeepSeek-V2-Lite-Chat (right). Solid lines represent TCAD and dashed lines represent the OPD-FKL baseline.

D.3 Gradient-Alignment Diagnostic

To quantify how the Monte Carlo sample count M affects gradient fidelity, we compare the MC gradient with exact compact enumeration at the common student initialization, before any distillation updates. This evaluation keeps all sampling budgets at identical parameters and therefore avoids confounding gradient fidelity with different training trajectories.

We randomly sample 100 token-level training states s_t. For each state, we construct the compact event space $\mathcal{A}_K(s_t)$ once with $K = 256$ and reuse the same event space for exact enumeration and all MC variants. Under the OPD-FKL objective, we compute the full-parameter student gradient of the exact compact divergence,

$$g_{\text{exact}}(s_t) = \nabla_\theta D_{\text{FKL}}(\bar{p}_T(\cdot \mid s_t) \| \bar{p}_\theta(\cdot \mid s_t)), \tag{21}$$

and, for each $M \in \{64, 128, 256, 512\}$, the gradient of the corresponding MC estimate,

$$g_{\text{MC}}^{(M)}(s_t) = \nabla_\theta \widehat{D}_{\text{FKL}}^{\text{MC},(M)}(s_t), \tag{22}$$

where the M events are sampled with replacement from the compact teacher distribution $\bar{p}_T(\cdot \mid s_t)$.

For each state, we flatten the gradients over all trainable student parameters and compute

$$C_M(s_t) = \frac{\left\langle g_{\text{MC}}^{(M)}(s_t), g_{\text{exact}}(s_t) \right\rangle}{\left\| g_{\text{MC}}^{(M)}(s_t) \right\|_2 \, \| g_{\text{exact}}(s_t) \|_2}. \tag{23}$$

We report the mean state-wise cosine similarity over the 100 matched states,

$$\mathrm{GradCos}(M) = \frac{1}{100} \sum_{t=1}^{100} C_M(s_t). \tag{24}$$

The resulting values are 0.72, 0.81, 0.90, and 0.96 for $M = 64, 128, 256,$ and 512, respectively. The monotonic increase indicates that larger MC sample counts more closely reproduce the exact compact FKL gradient direction at initialization. This diagnostic is separate from the downstream scores reported in the main text: it measures local gradient fidelity under matched states, not performance throughout training.

D.4 Qualitative Case Study

The following case study illustrates how confidence weighting can distinguish between two functionally correct rollouts with different structural redundancy.

E Limitations and Scope

The confidence probe tests a relative-ranking property within each prompt, not global calibration or correctness verification. Consequently, TCAD may be less effective when teacher likelihood is poorly aligned with quality for a particular domain, or when sampled rollouts do not contain meaningful quality variation. Monte Carlo sampling is not claimed to improve over exact compact enumeration for every sample count; its observed benefit depends on the sample count, proposal distribution, and training regime. Finally, the compact support-plus-tail construction remains an approximation to the full-vocabulary divergence because token-specific structure inside the tail is discarded.

Table 8: **Case study: teacher-confidence reweighting on two rollouts for the same prompt.** Both rollouts implement the same algorithm and produce the same output. The low-weight rollout contains repetitive, low-information steps (highlighted in red), which increase sequence length without adding task-relevant content and can reduce teacher log-likelihood under length normalization.

Prompt

```
Create a JavaScript program which takes an array of dates and returns the average date.
[2018-01-01, 2018-01-05, 2018-01-10]
```

Teacher confidence: High-weight

Student Response:

To compute the **average date** from an array of dates in JavaScript, you can follow these steps:

1. **Convert each date to a timestamp (number of milliseconds since the Unix epoch).**
2. **Sum all the timestamps.**
3. **Divide the total by the number of dates to get the average timestamp.**
4. **Convert the average timestamp back to a date object.**

Here's a complete JavaScript program that does this:

```
// Input array of dates in string format
const dates = ['2018-01-01', '2018-01-05', '2018-01-10'];

// Function to convert an array of date strings to an average date
function getAverageDate(dates) {
// Convert each date string to a Date object
const dateObjects = dates.map(dateStr => new Date(dateStr));

......
// Call the function and log the result
const averageDate = getAverageDate(dates);
console.log("Average Date:", averageDate.toISOString());
```

Example Output:

```
Average Date:  2018-01-05T00:00:00.000Z
```

```
> The function is generally accurate if you're dealing with dates that don't
span across decades or involve daylight saving changes in different time zones.
For datetime calculations across different time zones or durations, UTC-aware
computations become more nuanced.
```

Teacher confidence: Low-weight

Student Response:

To compute the **average date** from an array of dates in JavaScript, you can follow these steps:

1. **Convert each input date into a numeric timestamp (milliseconds since the Unix epoch).**
2. **Sum all timestamp values.**
3. **Divide by the number of dates to get the average timestamp, then convert it back to a Date object.**
4. **Convert the dates into millisecond timestamps again so the averaging is done in numbers.**
5. **Average those millisecond values again and convert the averaged timestamp back into a Date again.**

```
// Input array of dates in string format
const dates = ['2018-01-01', '2018-01-05', '2018-01-10'];

// Function to convert an array of date strings to an average date
function getAverageDate(dates) {
// Convert each date string to a Date object
const dateObjects = dates.map(dateStr => new Date(dateStr));

......
// Call the function and log the result
const averageDate = getAverageDate(dates);
console.log("Average Date:", averageDate.toISOString());
```

Example Output:

```
Average Date:  2018-01-05T00:00:00.000Z
> The function is generally accurate for simple date ranges, but results can
be sensitive to time zones or daylight-saving effects.  In practice, this
is generally accurate for simple date ranges.  For cross-time-zone datetime
arithmetic, it is safer to reason in UTC or use UTC-aware handling.  So, for
time zones, it is safer to stay in UTC and use UTC-aware handling.
```

AI USE STATEMENT

Generative AI tools were used only for language polishing, prose revision, and the formatting of tables and references. The methodology, theoretical development, proofs, and experimental design were developed and conducted by the authors without substantive AI assistance. All AI-assisted content was reviewed and verified by the authors, who take full responsibility for the paper's text, technical claims, proofs, and experimental results.