Calibrated Progressive Distillation: Co-Designing Curriculum and Target Mixing for Knowledge Distillation of Large Language Models

Mengxiang Zhang
The University of Hong Kong
mxzhang6@connect.hku.hk

Lingyuan Liu
Independent Researcher
ly.liu@my.cityu.edu.hk

Abstract

Knowledge distillation (KD) is a key technique for compressing large language models (LLMs), yet it faces challenges stemming from the teacher–student capacity gap. While existing KD methods address these challenges either by mixing teacher and student distributions in the distillation target or by using curriculum learning to sequence training from easy to hard examples, they typically design these two strategies independently, missing the opportunity for synergistic co-design. To bridge this gap, we propose **Calibrated Progressive Distillation (CPD)**, a white-box KD framework that co-designs curriculum scheduling and target mixing through a unified difficulty-aware principle. CPD uses a difficulty profile to select epoch-specific subsets that ensure a uniform increase in average difficulty, adapting to the dataset's intrinsic hardness structure. Simultaneously, the mixing coefficient in the distillation target and the distillation temperature are synchronized with this progression, gradually shifting supervision from teacher-dominated to student-informed signals as training advances. Theoretically, CPD ensures bounded gradients and induces an implicit attention shift from easy to hard samples. Empirically, CPD consistently outperforms advanced KD methods across diverse tasks, while reducing training runtime by over 10%. Our work demonstrates that aligning data scheduling with distillation signal design is crucial for effective and efficient LLM distillation.

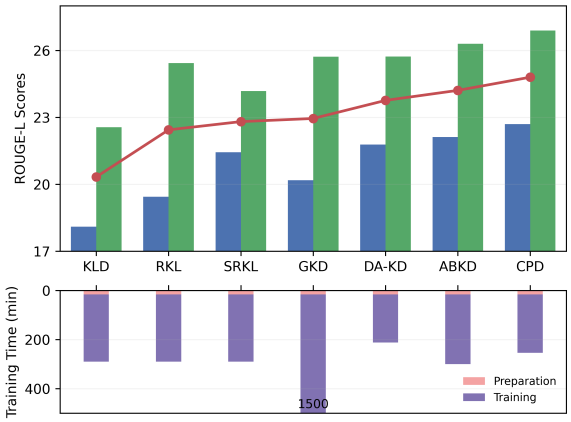

Figure 1: Effectiveness and efficiency of CPD versus strong KD baselines on five instruction-following benchmarks. CPD is compared against KLD (Hinton et al., 2015), RKL (Gu et al., 2023), on-policy GKD (Agarwal et al., 2024), SRKL (Ko et al., 2024), DA-KD (He et al., 2025), and ABKD (Wang et al., 2025) under two compression settings: Qwen2.5 (1.5B→0.5B) and OpenLLaMA2 (7B→3B). CPD consistently outperforms all methods while reducing training time by over 10%.

1 Introduction

Large language models (LLMs) require massive computational resources, hindering their deployment on edge devices (Aryan et al., 2023). Knowledge distillation (KD) offers a promising solution by transferring knowledge from a large teacher to a compact student model (Hinton et al., 2015). In the white-box setting, where the full teacher output distribution is accessible, KD leverages rich soft targets for more effective learning, particularly when utilizing open-source LLMs such as DeepSeek-v3 (Liu et al., 2024) and Qwen2.5 (Yang et al., 2024a).

However, two fundamental and interconnected challenges persist. First, the capacity gap between teacher and student often leads to **mode averaging** or **mode collapse**, where the student either oversmooths the teacher's nuanced predictions or fixates on dominant modes (Gu et al., 2023; Agarwal et al., 2024). Recent work addresses this via target mixing, which blends student and teacher distributions. Theoretical analysis (Shing et al., 2025) and empirical studies (Ko et al., 2024; He et al., 2025; Wang et al., 2025) have demonstrated that such mixing directly mitigates these mode collapse and averaging problems by reducing the distributional discrepancy attributable to the capacity gap. Yet, the mixing ratio is typically set heuristically, decoupled from the data's intrinsic difficulty, which ne-

Findings of the Association for Computational Linguistics: ACL 2026, pages 6757–6776
July 2-7, 2026

cessitates costly hyperparameter tuning and yields a suboptimal, one-size-fits-all supervisory signal.

Second, and critically related, standard KD suffers from a **training–inference mismatch** due to its use of a static dataset. While curriculum learning (CL) sequences samples from easy to hard to dynamically align training with learning progress (Bengio et al., 2009), existing KD integrations (Liu and Zhang, 2025; Ko et al., 2025; He et al., 2025) schedule data volume uniformly, ignoring the underlying difficulty distribution. This leads to a non-uniform difficulty trajectory (Tee and Zhang, 2023), stalling robust learning. Crucially, this curriculum design problem is inherently linked to the first challenge: an uncalibrated curriculum fails to inform how the teacher's guidance (i.e., the mixing ratio) should evolve as the student progresses. Treating CL and target mixing as independent modules misses the synergistic opportunity to co-design what the student learns (data) with how it learns (supervision).

To address these limitations, we propose **Calibrated Progressive Distillation (CPD)**, a white-box KD framework that *co-designs curriculum scheduling and target mixing* through a shared difficulty-aware principle. CPD first constructs a **Difficulty Profile**, representing the cumulative average difficulty as samples are sorted from easy to hard, and utilizes it to determine epoch-specific training subsets. This ensures a *uniform increase in average difficulty* regardless of the underlying difficulty distribution of the dataset. Concurrently, the mixing coefficient in the distillation target and the distillation temperature are *synchronized* with this progression: Early epochs emphasize teacher-consistent, sharp targets for stable learning on easy data, while later epochs gradually incorporate student predictions into the distillation target—enabling refined alignment with nuanced teacher knowledge on harder examples.

Theoretically, we show that CPD yields bounded gradients and induces an implicit attention shift from easy to hard samples. Empirically, CPD consistently improves student performance across instruction following, mathematical reasoning, and code generation. These gains are observed across diverse model families, including Qwen2.5 (Yang et al., 2024a), GPT-2 (Radford et al., 2019) and OpenLLaMA2 (Touvron et al., 2023), while simultaneously reducing training data usage and distillation runtime (see Fig. 1).

Our contributions are threefold:

Methodologically, we introduce CPD, the new framework to co-design curriculum scheduling and target mixing via a global difficulty profile;

Theoretically, we analyze gradient dynamics to show how CPD stabilizes optimization and aligns learning emphasis with curriculum progression;

Empirically, we demonstrate CPD's generality, efficiency, and superiority over strong baselines across tasks, architectures, and distillation settings.

2 Background and Preliminary Study

2.1 Current Framework for White-Box KD

We define the white-box KD framework for autoregressive LMs. The student model minimizes:

$$L_s = \alpha \cdot L_{ce} + (1 - \alpha) \cdot L_{kd}, \tag{1}$$

where L_{ce} is the cross-entropy loss against ground truth y, and L_{kd} measures divergence between teacher $p(y|x)$ and student $q_\theta(y|x)$ distributions scaled by temperature τ. The detailed description of white-box KD is shown in appendix A.1.

2.2 Limitations of Conventional Curriculum Learning in Knowledge Distillation

Curriculum learning is theoretically well-motivated in KD: easy samples, where the untrained student's output q_θ aligns with the teacher's p, yield stable gradients, while hard samples induce high-variance updates (Ko et al., 2025). A curriculum that sequences samples by increasing difficulty therefore promotes robust early learning (Wang et al., 2021). This staged alignment implements a core theoretical principle shared with on-policy KD (Agarwal et al., 2024): both progressively adapt the training distribution to align with the student's evolving competence, thereby reducing the distributional shift between training and the student's own generative states (Ko et al., 2024). CL achieves this via a pre-ordered difficulty gradient, whereas on-policy KD uses dynamic student outputs, offering a computationally efficient alternative to this adaptive learning paradigm. However, this ideal assumes the difficulty of selected samples increases smoothly.

Empirically, existing CL methods in KD schedule data by *uniformly increasing the number of samples per epoch* (Fig. 2a), ignoring the dataset's intrinsic difficulty distribution. As Fig. 2b shows, this results in a *non-uniform progression in average subset difficulty*, a direct consequence of the skewed structure shown in the difficulty profiles (Figures 2c–d). Such erratic difficulty trajectories

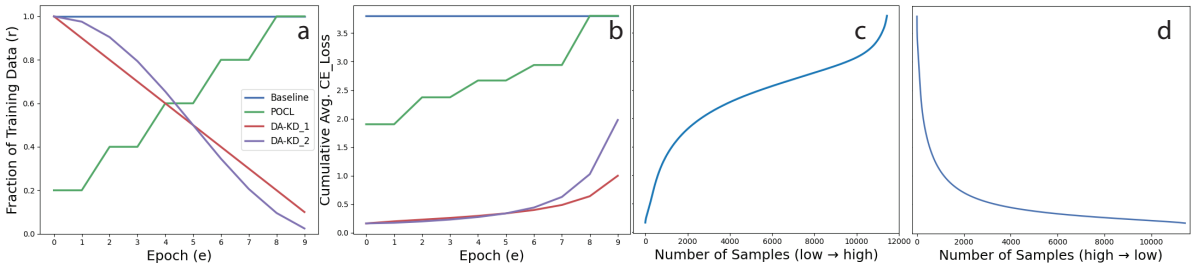

Figure 2: (a) Ratio of training data used per epoch under different CL frameworks; (b) Cumulative average cross-entropy loss of the active subset over epochs; (c) *Incremental difficulty profile*: cumulative average difficulty vs. sample count under ascending difficulty ordering; (d) *Diminishing difficulty profile*: same under descending ordering. POCL (Liu and Zhang, 2025) and DA-KD (He et al., 2025) use uniform sample inclusion or removal, leading to non-uniform difficulty progression.

can destabilize training and hinder performance (Tee and Zhang, 2023). This limitation is not merely a curriculum design flaw; it fundamentally disconnects the data schedule from the distillation signal. A curriculum with unpredictable difficulty jumps cannot reliably inform how strongly the teacher should guide the student at each stage, creating a misalignment between learning material and supervision. Our first core design principle is thus to directly control the *average difficulty trajectory* via Difficulty-Calibrated Curriculum Scheduling, ensuring a smooth and predictable learning pathway that can be synergistically coupled with target mixing.

2.3 Limitations of Static and Decoupled Mixing Strategies in KD

Target mixing, using a blend like $t \cdot q_\theta(y|x) + (1-t) \cdot p(y|x)$, is a popular strategy to mitigate mode collapse and training-inference mismatch (Hinton et al., 2015; Gu et al., 2023; Ko et al., 2024). While beneficial, current implementations suffer from a key oversight: *the mixing ratio t is decoupled from the curriculum's difficulty progression.*

First, t is typically scheduled independently of data difficulty—being fixed (Ko et al., 2024; He et al., 2025), linearly annealed (Ko et al., 2025), or adapted via loss (Shing et al., 2025). This ignores a fundamental pedagogical analogy: as a student advances from easy to hard material, the optimal blend of guidance (teacher) and self-practice (student) should evolve (Zhang and Liu, 2025). A static or arbitrarily scheduled t fails to calibrate the supervisory signal to the student's current competency level, encapsulated by the difficulty of the active curriculum subset.

Second, and as a direct consequence, existing methods treat CL and target mixing as independent modules (He et al., 2025; Ko et al., 2025). Their decoupling creates a suboptimal regime where the content of learning (curriculum) is not informed by, nor does it inform, the nature of the supervision (mixing ratio). This forces costly joint hyperparameter tuning to realign these components heuristically. We argue that effective distillation requires synergistic co-design: the mixing ratio should be explicitly modulated by the calibrated difficulty of the training subset. Our CPD framework operationalizes this by deriving t directly from the same difficulty metric that drives the curriculum, ensuring the teacher's guidance weakens precisely as the student masters increasingly challenging concepts.

3 Methodology

Fig. 3 shows the overview of our CPD framework. Given a teacher and a student LLM, CPD first estimates instance difficulty using the untrained student's cross-entropy loss and constructs a Difficulty Profile by sorting samples from easy to hard. This profile drives Difficulty-Calibrated Curriculum Scheduling, which selects epoch-specific subsets to enforce a uniform increase in average difficulty, adapting subset size to the dataset's intrinsic hardness distribution. Concurrently, CPD employs a Progressively Mixed Target Distribution in which the distillation target is a convex combination of teacher and student predictions. Both the mixing coefficient and temperature increase linearly across epochs, thereby synchronizing the supervision signal with the curriculum progression and the evolving student capacity. Full algorithmic details are in Algorithm 1 in Appendix A.2. We detail each component below.

3.1 Construction of Difficulty Profile

Given a dataset $D = \{(x_i, y_i)\}_{i=1}^N$, we first quantify the intrinsic difficulty of each instance using a pre-initialization signal from the student model.

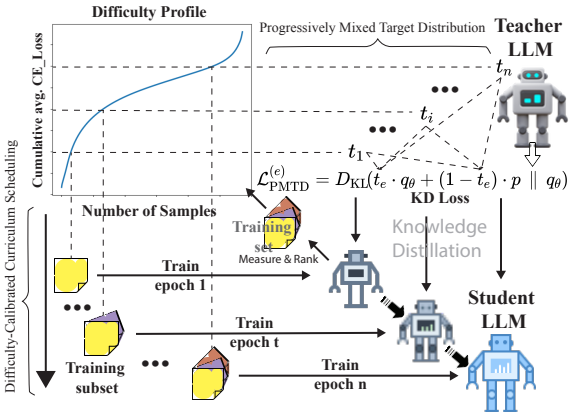

Figure 3: Overview of Calibrated Progressive Distillation (CPD). (1) A **Difficulty Profile** is built by sorting samples from easy to hard (by untrained student's CE loss) and computing cumulative average difficulty. (2) **Difficulty-Calibrated Curriculum Scheduling** selects epoch-specific subsets to ensure a uniform increase in average difficulty, adapting to the dataset's hardness structure. (3) **Progressively Mixed Target Distribution** shifts the distillation target from teacher-only to a student-aware blend as training progresses.

Specifically, without any training, we compute the cross-entropy loss of the untrained student model q_θ on each example: $\mathcal{L}_i = -\log q_\theta(y_i|x_i)$. This loss serves as a proxy for instance hardness: lower values indicate higher compatibility with the model's inductive bias and thus are deemed *easier*, while higher losses correspond to *harder* instances. This approach aligns with recent data-centric paradigms that leverage pre-training or zero-shot signals to characterize data properties without external supervision (Li et al., 2024).

We then sort all instances in ascending order of $\mathcal{L}_i$ to obtain a difficulty-sorted dataset: $D_{\text{ds}} = \{s_1, s_2, \ldots, s_N\}$, where s_i denotes the i-th easiest sample. Based on this ordering, we construct the *Difficulty Profile*, which is a function that captures the evolution of a prefix subset's average difficulty as additional samples are included. Formally, for any $k \in \{1, \ldots, N\}$, the cumulative average difficulty of the first k samples is defined as: $\bar{\mathcal{L}}(k) = \frac{1}{k}\sum_{j=1}^{k} \mathcal{L}_{s_j}$.

The Difficulty Profile is the sequence $\{\bar{\mathcal{L}}(k)\}_{k=1}^{N}$, which reveals the underlying structure of hardness distribution across the dataset. Crucially, this profile is often non-linear because it reflects the clustering of easy or hard examples. This phenomenon invalidates the use of curricula based on uniform sample inclusion.

3.2 Difficulty-Calibrated Curriculum Scheduling

Building upon the Difficulty Profile $\{\bar{\mathcal{L}}(k)\}_{k=1}^{N}$ constructed in the previous step, we design a curriculum that ensures the average difficulty of the training subset increases *uniformly* across epochs. Let n denote the total number of training epochs. Our goal is to partition the training process such that at epoch $e \in \{1, \ldots, n\}$, the active subset $D^{(e)} \subseteq D$ satisfies: $\frac{1}{|D^{(e)}|}\sum_{(x_i,y_i)\in D^{(e)}} \mathcal{L}_i = \frac{e}{n} \cdot \bar{\mathcal{L}}(N)$, where $\bar{\mathcal{L}}(N) = \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}_i$ is the average difficulty of the full dataset. This formulation enforces a linear progression in average difficulty from $\bar{\mathcal{L}}(N)/n$ in the first epoch to $\bar{\mathcal{L}}(N)$ in the final epoch.

To realize this, we leverage the monotonicity of the Difficulty Profile. Since $\bar{\mathcal{L}}(k)$ is non-decreasing in k (as samples are sorted from easiest to hardest), for each target average difficulty $\tau_e = \frac{e}{n} \cdot \bar{\mathcal{L}}(N)$, there exists a unique smallest index $k_e \in \{1, \ldots, N\}$ such that: $\bar{\mathcal{L}}(k_e) \geq \tau_e$.

In practice, we compute k_e via inverse lookup on the precomputed profile: $k_e = \min\{k \in \{1, \ldots, N\} \mid \bar{\mathcal{L}}(k) \geq \frac{e}{n} \cdot \bar{\mathcal{L}}(N)\}$. The training subset for epoch e is then defined as the prefix of the difficulty-sorted dataset up to k_e: $D^{(e)} = \{s_1, s_2, \ldots, s_{k_e}\}$.

By construction, $k_1 < k_2 < \cdots < k_n = N$, ensuring both the number of samples and the average difficulty increase monotonically with epoch. Crucially, unlike conventional "baby-step" curricula that increment sample count uniformly (Wang et al., 2021), our scheduling adapts the subset size to the *intrinsic difficulty distribution* of the data, guaranteeing a uniform progression in learning challenge. This calibrated progression forms the foundation for synchronizing the distillation target mixing ratio in our full CPD framework.

3.3 Progressively Mixed Target Distribution

To complement our Difficulty-Calibrated Curriculum Scheduling, we introduce a *Progressively Mixed Target Distribution (PMTD)* that dynamically adjusts the supervision signal throughout training. Standard knowledge distillation minimizes the KLD between the student's output distribution $q_\theta(y|x;\tau)$ and the teacher's softened distribution $p(y|x;\tau)$ (Hinton et al., 2015). In contrast, our approach constructs a convex combination of the teacher and student distributions as the distillation target: $p_t^{(e)}(y|x;\tau_e) = t_e \cdot q_\theta(y|x;\tau_e) + (1 -$

$t_e) \cdot p(y|x;\tau_e)$, where τ_e denotes the distillation temperature at epoch e, and minimizes the asymmetric KLD:

$$\mathcal{L}^{(e)}_{\text{PMTD}} = D_{\text{KD}}\left(p_t^{(e)}(y|x;\tau_e) \,\middle\|\, q_\theta(y|x;\tau_e)\right). \quad (2)$$

This formulation encourages the student to gradually shift from full alignment with the teacher toward increasingly self-consistent predictions as training progresses. The mixing coefficient $t_e \in [0,1]$ is scheduled to increase linearly with epoch $e \in \{1,\ldots,n\}$: $t_e = \frac{e}{n}$. This design is deliberately synchronized with our curriculum: as the average difficulty of the training subset rises uniformly (Section 3.2), the student is increasingly held accountable to the teacher's output, reflecting growing capacity to absorb complex knowledge.

Furthermore, to harmonize the softness of supervision with the student's evolving proficiency, we employ a linearly increasing distillation temperature: $\tau_e = \tau_1 + (\tau_n - \tau_1) \cdot \frac{e-1}{n-1}$, where τ_1 and τ_n are initial and final temperatures. Early in training, a lower τ_e yields sharper teacher distributions for stable learning on easy examples; In later stages, a higher τ_e reveals fine-grained logit relationships and enables nuanced knowledge transfer on harder instances, which ensures that the distillation signal remains calibrated to both student capacity and curriculum difficulty.

4 Theoretical Analysis

To understand why CPD yields stable and effective optimization, we analyze the gradient behavior of our PMTD loss with respect to the student parameters θ. The gradient of the loss (See Eq. 2) with respect to θ is:

$$\nabla_\theta \mathcal{L}^{(e)}_{\text{PMTD}} = \sum_y \underbrace{\left[t_e \log \frac{p_t^{(e)}(y)}{q_\theta(y)} + t_e - \frac{p_t^{(e)}(y)}{q_\theta(y)}\right]}_{K^{(e)}(y)} \nabla_\theta q_\theta(y), \quad (3)$$

where the per-label coefficient $K^{(e)}(y)$ governs the direction and magnitude of the update. Full derivation is provided in Appendix A.3. We now examine $K^{(e)}(y)$ in two training phases, aligned with our curriculum design.

Early Training (Easy Samples, Small t_e). In the initial epochs, CPD uses only the easiest samples (Section 3.2). For such samples, empirical and theoretical studies (Ko et al., 2025) show that the untrained student's prediction is already close to the teacher's: $q_\theta(y) \approx p(y)$. Consequently, $p_t^{(e)}(y) \approx q_\theta(y)$, and substituting into $K^{(e)}(y)$ yields: $K^{(e)}(y) \approx t_e \log 1 + t_e - 1 = t_e - 1$. Since $t_e = e/n$ is small (e.g., $t_1 = 1/n$), $K^{(e)}(y) \approx -1$, a bounded constant independent of the instantaneous values of $q_\theta(y)$ or $p(y)$. This ensures that gradients do not vanish or explode due to extreme probability ratios, which is a common issue in standard KL-based distillation when $q_\theta(y) \to 0$. Moreover, the magnitude $|K^{(e)}(y)| \approx 1 - t_e$ is relatively large when t_e is small, providing strong learning signals on easy examples during early training.

Late Training (Hard Samples, Large t_e). In later epochs, harder samples are introduced, and $t_e \to 1$. For hard examples, the student's initial predictions often deviate significantly from the teacher's, and it is common that $q_\theta(y) \gg p(y)$ for low-probability labels (He et al., 2025). In this regime, $\frac{p(y)}{q_\theta(y)} \to 0$, so: $p_t^{(e)}(y) = t_e \cdot q_\theta(y) + (1 - t_e) \cdot p(y) \approx t_e \cdot q_\theta(y)$, and thus: $K^{(e)}(y) \approx t_e \log t_e + t_e - t_e = t_e \log t_e$. Since $t_e \in (0,1]$, $t_e \log t_e$ is finite, again yielding bounded gradients regardless of the divergence between q_θ and p.

Notably, for easy samples that were dominant early on, $q_\theta(y) \approx p(y)$ still holds, so $K^{(e)}(y) \approx t_e - 1 \to 0$ as $t_e \to 1$. This implies that CPD *automatically downweights easy samples* in later stages and shifts the optimization focus toward hard examples, a result that aligns precisely with the intent of our Difficulty-Calibrated Curriculum Scheduling.

Remark 1 (Gradient Boundedness)**.** *The per-label gradient coefficient $K^{(e)}(y)$ remains bounded for all epochs and outputs, ensuring stable training without explicit regularization such as gradient clipping or label smoothing.*

Remark 2 (Curriculum–Distillation Alignment)**.** *The evolution of $K^{(e)}(y)$ induces an* implicit attention shift*: early updates emphasize easy samples via large-magnitude gradients, while late updates suppress them and prioritize hard samples through non-vanishing, bounded signals.*

5 Empirical Analysis

We evaluate CPD on three representative LLM distillation tasks: general instruction following, mathematical reasoning, and code generation. Key hyperparameters are set as follows: initial and final distillation temperatures $\tau_1 = 1$ and $\tau_n = 2$; supervised fine-tuning (SFT) weight $\alpha = 0.3$ for off-policy KD and $\alpha = 0$ for on-policy KD. We compare CPD against four categories of white-box KD baselines: (1) **Standard KD losses without curriculum and target mixing**: KLD (Hinton et al., 2015), RKL

(Gu et al., 2023), and TVD (Wen et al., 2023); (2) **KD losses with mixed target distributions but no curriculum**: GKD (on-policy) (Agarwal et al., 2024), JSD, SKL/SRKL (Ko et al., 2024), ABKD (Wang et al., 2025), and TAID (Shing et al., 2025); (3) **Curriculum-based methods without target mixing**: KLD+POCL (Liu and Zhang, 2025); (4) **Joint curriculum and target mixing approaches**: DA-KD (He et al., 2025). Full experimental details are in Appendix A.2.

5.1 General Instruction-Following

Setup. We distill Qwen2.5 (0.5B student from 1.5B teacher) (Yang et al., 2024a), GPT-2 (0.1B from 1.5B) (Radford et al., 2019) and OpenLLaMA2 (3B from 7B) (Geng and Liu, 2023) on the `Databricks-dolly-15k` dataset (Conover et al., 2023) (11.5K train, 1K validation, 0.5K test). Models are evaluated on five instruction-following benchmarks: DollyEval, SelfInst (Wang et al., 2022a), VicunaEval (Chiang et al., 2023), S-NI (Wang et al., 2022b), and UnNI (Honovich et al., 2022). We report (1) average ROUGE-L scores across five random seeds, and (2) winning rates (WR) via LLM-as-a-Judge (Zheng et al., 2023) using DeepSeek-V3-0324 (Liu et al., 2024).

Results. As shown in Tab. 1 and 6, CPD consistently outperforms all baseline KD methods across both model scales and all five evaluation benchmarks under both evaluation metrics and across model scales, demonstrating its effectiveness and scalability. Moreover, CPD achieves these gains with only 82.3% of the training iterations required by standard baselines, confirming its computational efficiency. All reported improvements are statistically significant according to Wilcoxon signed-rank tests ($p < 0.05$). The main experiment includes **three backbone families**: Qwen2.5 (1.5B→0.5B), GPT-2 (1.5B→0.1B), and OpenLLaMA2 (7B→3B), with GPT-2 and OpenLLaMA2 results deferred to the appendix due to space constraints. All three show consistent trends. For the extensive analyses in Section 6, we use GPT-2 as a computationally efficient proxy, given our resource constraints and the consistent performance across architectures in the main experiments.

5.2 Mathematical Reasoning and Code Generation

Setup. For mathematical reasoning, we distill Qwen2.5-Math-1.5B from Qwen2.5-Math-7B-Inst (Yang et al., 2024b) using 20K training and 2K validation samples from MetaMathQA (Yu et al., 2023), with chain-of-thought prompting (Wei et al., 2022). Evaluation is conducted on GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021) using *pass@1*.

For code generation, we distill Qwen2.5-Coder-1.5B from Qwen2.5-Coder-7B-Inst (Hui et al., 2024) using 20K/2K samples from `Evol-Instruct-Code-80k-v1` (Luo et al., 2023), a dataset enhanced via Evol-Instruct (Xu et al., 2024). Models are evaluated on HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021) under the EvalPlus framework (Liu et al., 2023), also using *pass@1*.

Results. Tab. 2 demonstrates that CPD maintains superior performance across mathematical reasoning and code generation domains, consistently outperforming competing KD methods on both mathematical and code benchmarks. These results confirm that the benefits of CPD extend beyond general instruction following to structured, symbolic tasks, highlighting its generality and robustness across diverse LLM application domains.

6 Analysis and Discussion

6.1 Evaluation on Diverse KD Scenarios

To evaluate the generalizability of CPD, we conduct experiments under two critical scenarios: (1) varying teacher-student capacity gaps, and (2) low-resource settings with limited training data.

Varying Capacity Gaps. We distill the GPT-2-1.5B teacher to three differently sized students (0.1B, 0.3B, and 0.8B). Results in Tab. 7 and Fig. 4(a) demonstrate CPD's robust performance. Notably, as the student capacity increases (from 0.1B to 0.8B), CPD's relative advantage becomes more pronounced. For the largest 0.8B student, CPD achieves the highest average score (26.17), outperforming strong baselines like ABKD (25.98) and KLD+POCL (25.53). This trend indicates that CPD's co-designed curriculum and target mixing effectively leverages increased student capacity, mitigating the mode collapse risk inherent in large gaps. Compared to DA-KD, which jointly schedules curriculum and target mixing in a decoupled manner, CPD consistently outperforms across all capacity gaps (Tab. 7), confirming the advantage of **co-designing** curriculum and target mixing through a unified difficulty-aware principle.

Low-Resource Settings. We simulate data scarcity by randomly sampling 10%, 30%, 50%,

		Qwen2.5-1.5B (M_t) → Qwen2.5-0.5B (M_s)											
		DollyEval		**SelfInst**		**VicunaEval**		**S-NI**		**UnNI**		**Avg.**	
Categories	KD Methods	*R-L*	*WR*	*R-L*	*WR*	*R-L*	*WR*	*R-L*	*WR*	*R-L*	*WR*	*R-L*	*WR*
	M_t	27.48	53.65	18.60	53.86	18.27	54.53	34.94	53.63	34.86	54.23	26.83	53.98
	SFT	24.86	47.64	14.42	45.75	15.77	48.18	22.80	41.42	24.59	42.86	20.49	45.17
(1)	KLD	26.71	51.32	15.75	49.35	15.08	51.88	26.81	48.25	27.62	48.19	22.40	49.80
	RKL	27.13	51.04	16.90	49.97	16.10	51.62	30.08	49.51	28.95	48.98	23.83	50.23
	TVD	26.76	50.94	15.71	49.35	16.20	51.25	30.32	50.60	29.95	49.67	23.79	50.36
(2)	GKD (on-policy)	27.43	51.56	16.06	49.90	16.63	52.77	30.55	51.21	30.68	50.07	24.27	51.10
	JSD	26.84	51.46	15.85	48.28	16.16	51.11	26.92	48.48	27.61	46.48	22.68	49.16
	SKL	27.02	51.63	17.11	51.12	16.73	52.92	30.43	50.65	28.87	48.64	24.03	50.99
	SRKL	26.71	50.78	16.64	49.31	16.23	49.92	30.36	48.65	29.50	48.58	23.89	49.45
	ABKD	27.69	53.75	18.27	52.70	17.13	53.62	**34.68**	**53.54**	33.60	53.67	26.27	53.45
	TAID	27.43	52.81	18.41	51.85	18.11	52.95	34.19	53.00	33.25	52.80	26.28	52.68
(3)	KLD+POCL	27.31	51.37	15.39	48.05	16.63	53.84	29.95	50.86	31.22	51.07	24.10	51.04
(4)	DA-KD	27.43	51.62	17.62	51.90	17.79	53.28	34.08	52.20	34.16	52.99	26.22	52.40
	CPD	**27.82**	**53.89**	**18.58**	**53.20**	**18.43**	**54.76**	34.63	53.30	**34.35**	**54.08**	**26.76**	**53.84**

Table 1: Instruction-following evaluation on Qwen2.5 (1.5B→0.5B). M_t and M_s denote teacher and student models, respectively. Reported metrics are ROUGE-L (R-L) and winning rate (WR) on DollyEval, SelfInst, VicunaEval, S-NI, and UnNI; Avg. denotes mean across benchmarks. red indicates the student outperforms the teacher. Best student results are in **bold**. GPT-2 and OpenLLaMA2 results are in Tab. 6

		Qwen2.5-Math-7B-Inst (M_t) → Qwen2.5-Math-1.5B (M_s)			Qwen2.5-Coder-7B-Inst (M_t) → Qwen2.5-Coder-1.5B (M_s)		
		GSM8K	**MATH**	**Avg.**	**HumanEval**	**MBPP**	**Avg.**
Categories	KD Methods	*pass@1*	*pass@1*	*pass@1*	*pass@1*	*pass@1*	*pass@1*
	M_t	90.3	80.2	85.3	61.6	76.9	69.3
	M_s	75.9	48.7	62.3	42.4	60.4	51.4
(1)	KLD	77.9	55.2	66.6	45.1	61.3	53.2
	RKL	78.4	56.4	67.4	45.4	61.4	53.4
	TVD	78.2	56.9	67.6	46.2	61.7	54.0
(2)	GKD (on-policy)	80.2	58.3	69.3	46.2	62.6	54.4
	JSD	78.9	56.8	67.9	45.1	61.0	53.1
	SKL	79.4	57.9	68.7	46.4	61.8	54.1
	SRKL	79.5	58.4	69.0	46.8	62.0	54.4
	ABKD	80.7	59.8	70.3	48.4	63.8	56.1
	TAID	80.1	59.6	69.9	47.5	63.5	55.5
(3)	KLD+POCL	79.4	57.8	68.6	46.1	61.8	53.9
(4)	DA-KD	79.9	59.3	69.6	47.3	62.3	54.8
	CPD	**81.4**	**60.1**	**70.8**	**48.5**	**64.4**	**56.5**

Table 2: Results on mathematical reasoning (GSM8K, MATH) and code generation (HumanEval, MBPP) using Qwen2.5-Math and Qwen2.5-Coder (7B→1.5B). Performance is measured by *pass@1*; best scores are in **bold**.

and 80% of the `Databricks-dolly-15k` training set. Fig. 4(b) demonstrates that CPD consistently outperforms the baseline methods (SFT, KLD, and DA-KD) at every corresponding level of data availability (10% to 100%). Notably, its performance gains are most pronounced under significant data constraints. This consistent superiority across all resource levels confirms the robustness and data efficiency of CPD's co-designed curriculum in low-resource distillation scenarios.

6.2 Evolution of Difficulty Rankings During Training

CPD relies on a static difficulty profile computed from the untrained student. To assess whether this profile becomes stale as the student learns, we measure the Spearman Rank Correlation (SRC) between the initial static ranking and rankings derived from checkpoints at different epochs, along with the overlap of the resulting training subsets.

As shown in Tab. 3, while rank correlation declines over training, the training subsets themselves overlap substantially (>80%). This is because subset selection depends on positional thresholds in the difficulty-ordered list, and local rank permutations have minimal impact on the actual training data per epoch. We further compare performance using a dynamically updated profile:

Epoch	SRC	Subset Intersection (%)
5	0.731	90.31
10	0.543	86.28
15	0.415	80.19
20	0.372	100.00

Table 3: Rank correlation and subset overlap between static and dynamic difficulty profiles (GPT-2 1.5B→0.1B).

Method	Avg. ROUGE-L	Time (mins)
CPD (static)	22.70	254
CPD (dynamic)	22.84	542

Table 4: Performance and computational cost of static vs. dynamic profiling (GPT-2 1.5B→0.1B).

Dynamic profiling (Tab. 4) yields a negligible gain (+0.14 ROUGE-L, +0.62%) at more than double the computational cost (+113% training time), confirming that static profiling offers the best trade-off for CPD.

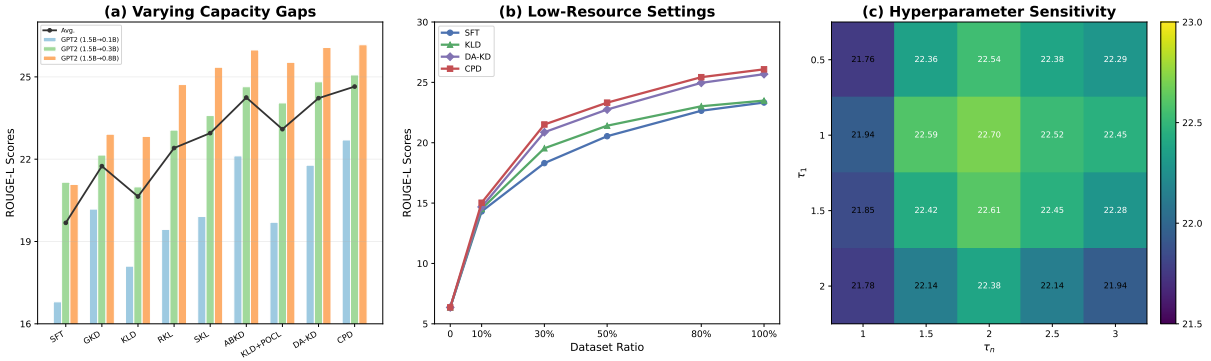

Figure 4: (a) Average ROUGE-L scores for distilling GPT-2-1.5B to students of different sizes (0.1B, 0.3B, 0.8B), including DA-KD comparison. (b) Performance under low-resource settings: ROUGE-L scores for distilling to GPT-2-0.1B using progressively larger subsets (10%-100%) of the training data, including DA-KD comparison. (c) Hyperparameter sensitivity heatmap: Average ROUGE-L score across five benchmarks for CPD with GPT-2, evaluated over a 2D grid of initial (τ_1) and final (τ_n) temperature parameters.

6.3 Ablation Analysis

Impact of Difficulty Metrics. We evaluate four difficulty metrics for constructing the Difficulty Profile: cross-entropy loss of the untrained student (CE-LOSS, our default), ROUGE-L between student generations and ground truth (Liu and Zhang, 2025), distillation difficulty score (DDS) (He et al., 2025), and sentence-level entropy (SE) (Zhu et al., 2021). As shown in Tab. 5 (a) and 8, CE-LOSS achieves the best overall performance across both instruction-following and math reasoning tasks, matching or exceeding the more complex DDS while being computationally more efficient (requiring only a forward pass of the untrained student). ROUGE-L underperforms on math tasks, likely due to insensitivity to arithmetic errors, and SE yields the lowest scores, indicating that entropy alone is insufficient. These results confirm CE-LOSS as a simple, effective, and task-agnostic difficulty estimator.

KD Methods	GPT-2-1.5B → GPT-2-0.1B Avg. ROUGE-L	Qwen2.5-Math-7B-Inst → Qwen2.5-Math-1.5B Avg. pass@1	TI
a. *Impact of Difficulty Metrics.*			
CPD (CE-LOSS)	22.70	56.5	-
CPD (ROUGE-L)	22.48	54.2	-
CPD (DDS)	22.61	56.4	-
CPD (SE)	20.87	54.7	-
b. *Impact of Curriculum Scheduling.*			
CPD (DCS)	22.70	56.5	23517
CPD (BSS)	21.89	55.7	28588
CPD (LS)	21.11	54.4	15008
CPD (CS)	21.65	56.0	24299
c. *Impact of Mixture Ratio Dynamics.*			
CPD (linear)	22.70	56.5	-
CPD (cosine)	22.14	55.1	-
CPD (fixed)	21.62	54.2	-
CPD (adaptive)	22.68	56.6	-

Table 5: Ablation on a) difficulty metrics for Difficulty Profile construction, b) curriculum scheduling strategies, and c) mixture ratio scheduling. Avg. denotes mean performance across five instruction-following benchmarks (ROUGE-L) and two math reasoning benchmarks (pass@1). Total iterations (TI) are measured on `Databricks-dolly-15k` over 20 epochs with batch size 8.

Impact of Curriculum Scheduling. We compare CPD's Difficulty-Calibrated Scheduling (DCS) against three alternatives: baby-step scheduling (BSS) (Liu and Zhang, 2025), linear growth (LS), and cosine growth (CS) (He et al., 2025). As shown in Tab. 5 (b) and 9, DCS consistently achieves the highest performance on both task types, validating that a uniform increase in average difficulty, adapted to the dataset's intrinsic distribution, leads to more effective knowledge transfer. Moreover, DCS attains superior results with fewer iterations than BSS and CS, demonstrating training efficiency.

Impact of Mixture Ratio Dynamics. We evaluate four strategies for scheduling the mixture ratio: linear growth (our default), cosine growth, fixed ratio (Ko et al., 2024), and adaptive growth (Shing et al., 2025). Tab. 5 (c) and 10 shows that both linear and adaptive scheduling significantly outperform the fixed and cosine strategies, confirming the importance of dynamic adjustment. Linear growth achieves performance nearly on par with adaptive scheduling despite its simplicity and zero overhead, due to its synchronization with the difficulty-calibrated curriculum. The cosine schedule underperforms linear, likely because its non-linear trajectory misaligns with the linear difficulty progression enforced by the curriculum. This confirms that synchronization with the curriculum—not the specific functional form—is the key factor.

6.4 Hyperparameter Sensitivity Analysis

We analyze the sensitivity of CPD to its core hyperparameters—the initial (τ_1) and final (τ_n) temperatures governing the curriculum's difficulty trajectory. The results, visualized as a performance heatmap in Fig. 4(c), reveal two key findings. First, CPD exhibits a stable optimal region centered around $(\tau_1, \tau_n) = (1.0, 2.0)$. Performance remains high across a broad neighborhood of these values, indicating robustness to precise tuning. Second, the performance decline outside this region is gradual, not abrupt, confirming there is no sharp performance cliff.

6.5 Computational Cost Analysis

CPD introduces a one-time preparation phase to compute the Difficulty Profile, which involves a forward pass of the full dataset through the untrained student to obtain cross-entropy losses (e.g., taking 16 minutes for `Dolly` on one A100 GPUs). Critically, this cost is incurred offline before training begins. The subsequent training phase itself incurs no additional overhead and, due to the optimized curriculum, requires 17.7% fewer iterations than the baseline (See Fig. 1). The total time, including preparation and training, is 254 minutes, representing 87.5% of the baseline's training-only time (290 mins). This demonstrates that the modest upfront investment is decisively offset by significantly improved training efficiency.

6.6 Discussion

Why CPD Excels. A natural question is *why* co-designing curriculum and target mixing yields consistent improvements. Our theoretical analysis (Section 4) provides a mechanistic explanation: the per-label gradient coefficient $K^{(e)}(y)$ remains bounded for all epochs and outputs, ensuring stable training without explicit regularization. Moreover, the evolution of $K^{(e)}(y)$ induces an *implicit attention shift*—early updates emphasize easy samples via large-magnitude gradients, while late updates suppress them and prioritize hard samples through non-vanishing, bounded signals (Remarks 1–2). This alignment between curriculum progression and gradient dynamics means that CPD actively stabilizes the optimization trajectory, rather than merely sequencing data.

Robustness in Low-Resource Settings. This gradient stabilization mechanism is especially beneficial when training data is limited (Section 6.1). In low-resource regimes, data distribution fluctuations are more pronounced, increasing the risk of unstable gradients. CPD's co-designed curriculum and target mixing smooth the optimization landscape by ensuring that the difficulty of training samples increases uniformly and synchronizes with the distillation target. This explains CPD's pronounced advantage at low data ratios (Fig. 4b), where standard methods suffer from erratic gradient updates on a small, potentially unrepresentative data pool.

7 Related Works

We discuss related work and defer a concentrated account to Appendix A.6.

8 Conclusion

We propose **Calibrated Progressive Distillation (CPD)**, a white-box KD framework that co-designs curriculum scheduling and target mixing via a shared, difficulty-aware principle. CPD constructs a Difficulty Profile from the untrained student's cross-entropy losses to enforce a uniform increase in average training difficulty. This curriculum is synchronized with a progressively mixed distillation target, where both the mixing coefficient and temperature evolve linearly in tandem. This integrated co-design aligns the training data with the supervisory signal. Theoretically, CPD ensures bounded gradients and induces an implicit attention shift from easy to hard samples. Empirically, CPD demonstrates robust generalizability: it outperforms advanced KD baselines across diverse tasks for a range of model families and student model sizes; it consistently outperforms baselines under varying teacher-student capacity gaps and exhibits exceptional data efficiency in severely low-resource settings. Furthermore, CPD reduces total training runtime by over 10% through its efficient curriculum. Collectively, our findings position CPD as a general, efficient, and theoretically grounded method for LLM compression, advancing their practical accessibility. Future work may explore adaptive difficulty metrics, scalable difficulty estimation (e.g., via sampling or dynamic batching) for extending CPD to larger-scale or pretraining scenarios, and extensions to multimodal tasks.

9 Limitations

CPD depends on accurate per-sample difficulty estimation to co-design curriculum scheduling and tar-

get mixing. Currently, CPD uses token-level cross-entropy loss for this purpose, which works well for instruction-following tasks but yields limited gains in mathematical reasoning and code generation. In these domains, functional correctness is highly sensitive to small discrete errors, such as arithmetic mistakes or syntax bugs. Since token-level likelihoods often fail to capture these nuances, they can lead to misleading difficulty estimates. This suggests a need for task-aware difficulty metrics. Future work should incorporate execution-based feedback (e.g., test case outcomes or answer verification) into the distillation pipeline to better align difficulty assessment with actual correctness.

Besides, a limitation of CPD is its use of a static Difficulty Profile, computed once using the untrained student. While this provides a stable and computationally efficient curriculum, it does not dynamically adapt to the student's evolving capacity during training. Consequently, the relative difficulty of a sample is not updated as the student learns, which may limit the optimality of the curriculum sequence in later stages. However, our analysis (Section 6.2) shows that despite declining rank correlation, the training subsets themselves overlap substantially (>80%), and dynamic profiling yields negligible gains at more than double the computational cost. This design choice represents a deliberate trade-off: a fully dynamic curriculum, akin to on-policy KD which regenerates training data at multiple stages, would incur significant computational overhead (e.g., up to 80% of total training time (Ko et al., 2024)). Our static profile avoids this cost while still providing a robust, data-aware learning progression, as evidenced by CPD's strong empirical performance. Future work could explore semi-dynamic variants that periodically update the difficulty profile every K epochs to better capture student progress while managing computational cost.

References

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

Abi Aryan, Aakash Kumar Nain, Andrew McMahon, Lucas Augusto Meyer, and Harpreet Singh Sahota. 2023. The costly dilemma: generalization, evaluation and cost-optimal deployment of large language models. *arXiv preprint arXiv:2308.08061*.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Wei-Lin Chiang, Zhuohan Li, Ziqing Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E Gonzalez, et al. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. *See https://vicuna. lmsys. org (accessed 14 April 2023)*, 2(3):6.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Together Computer. 2023. Redpajama-data: An open source recipe to reproduce llama training dataset.

Mike Conover, Matt Hayes, Ankit Mathur, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, and Reynold Xin. 2023. Free dolly: Introducing the world's first truly open instruction-tuned llm.

Shen Gao, Zhengliang Shi, Minghang Zhu, Bowen Fang, Xin Xin, Pengjie Ren, Zhumin Chen, Jun Ma, and Zhaochun Ren. 2024. Confucius: Iterative tool learning from introspection feedback by easy-to-difficult curriculum. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18030–18038.

Xinyang Geng and Hao Liu. 2023. Openllama: An open reproduction of llama.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. 2023. Minillm: Knowledge distillation of large language models. *arXiv preprint arXiv:2306.08543*.

Changyi He, Yifu Ding, Jinyang Guo, Ruihao Gong, Haotong Qin, and Xianglong Liu. 2025. Da-kd: Difficulty-aware knowledge distillation for efficient large language models. In *Forty-second International Conference on Machine Learning*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Or Honovich, Thomas Scialom, Omer Levy, and Timo Schick. 2022. Unnatural instructions: Tuning language models with (almost) no human labor. *arXiv preprint arXiv:2212.09689*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2022. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, et al. 2024. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*.

Jongwoo Ko, Tianyi Chen, Sungnyun Kim, Tianyu Ding, Luming Liang, Ilya Zharkov, and Se-Young Yun. 2025. Distillm-2: A contrastive approach boosts the distillation of llms. *arXiv preprint arXiv:2503.07067*.

Jongwoo Ko, Sungnyun Kim, Tianyi Chen, and Se-Young Yun. 2024. Distillm: Towards streamlined distillation for large language models. *arXiv preprint arXiv:2402.03898*.

Ming Li, Yong Zhang, Shwai He, Zhitao Li, Hongyu Zhao, Jianzong Wang, Ning Cheng, and Tianyi Zhou. 2024. Superfiltering: Weak-to-strong data filtering for fast instruction-tuning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14255–14273.

Zheng Li, Xiang Li, Lingfeng Yang, Borui Zhao, Renjie Song, Lei Luo, Jun Li, and Jian Yang. 2023. Curriculum temperature for knowledge distillation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 1504–1512.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. 2023. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. *Advances in Neural Information Processing Systems*, 36:21558–21572.

Lingyuan Liu and Mengxiang Zhang. 2025. Being strong progressively! enhancing knowledge distillation of large language models through a curriculum learning framework. *arXiv preprint arXiv:2506.05695*.

Ziyang Luo, Can Xu, Pu Zhao, Qingfeng Sun, Xiubo Geng, Wenxiang Hu, Chongyang Tao, Jing Ma, Qingwei Lin, and Daxin Jiang. 2023. Wizardcoder: Empowering code large language models with evol-instruct. *arXiv preprint arXiv:2306.08568*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Ayan Sengupta, Shantanu Dixit, Md Shad Akhtar, and Tanmoy Chakraborty. 2023. A good learner can teach better: Teacher-student collaborative knowledge distillation. In *Proceedings of the The Twelfth International Conference on Learning Representations, Virtual Event*, pages 25–29.

Makoto Shing, Kou Misaki, Han Bao, Sho Yokoi, and Takuya Akiba. 2025. Taid: Temporally adaptive interpolated distillation for efficient knowledge transfer in language models. *arXiv preprint arXiv:2501.16937*.

Qwen Team. 2024. Qwen2. 5 technical report. *arXiv preprint arXiv:2407.10671*.

Ren Jie Tee and Mengmi Zhang. 2023. Integrating curricula with replays: Its effects on continual learning. In *Proceedings of the AAAI Symposium Series*, volume 1, pages 109–116.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Guanghui Wang, Zhiyong Yang, Zitai Wang, Shi Wang, Qianqian Xu, and Qingming Huang. 2025. Abkd: Pursuing a proper allocation of the probability mass in knowledge distillation via $alpha$-$beta$-divergence. *arXiv preprint arXiv:2505.04560*.

Xin Wang, Yudong Chen, and Wenwu Zhu. 2021. A survey on curriculum learning. *IEEE transactions on pattern analysis and machine intelligence*, 44(9):4555–4576.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2022a. Self-instruct: Aligning language models with self-generated instructions. *arXiv preprint arXiv:2212.10560*.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Anjana Arunkumar, Arjun Ashok, Arut Selvan Dhanasekaran, Atharva Naik, David Stap, et al. 2022b. Benchmarking generalization via in-context instructions on 1,600+ language tasks. *arXiv preprint arXiv:2204.07705*, 2:2.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou,

et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Yuqiao Wen, Zichao Li, Wenyu Du, and Lili Mou. 2023. F-divergence minimization for sequence-level knowledge distillation. *arXiv preprint arXiv:2307.15190*.

Liuyu Xiang, Guiguang Ding, and Jungong Han. 2020. Learning from multiple experts: Self-paced knowledge distillation for long-tailed classification. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part V 16*, pages 247–263. Springer.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, Qingwei Lin, and Daxin Jiang. 2024. Wizardlm: Empowering large pre-trained language models to follow complex instructions. In *The Twelfth International Conference on Learning Representations*.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, et al. 2024a. Qwen2. 5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, et al. 2024b. Qwen2. 5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*.

Chuanpeng Yang, Yao Zhu, Wang Lu, Yidong Wang, Qian Chen, Chenlong Gao, Bingjie Yan, and Yiqiang Chen. 2024c. Survey on knowledge distillation for large language models: methods, evaluation, and application. *ACM Transactions on Intelligent Systems and Technology*.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2023. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*.

Mengxiang Zhang and Lingyuan Liu. 2025. Staged knowledge distillation through least-to-most prompting: Optimizing teacher guidance via difficulty-aware training. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 8489–8501.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623.

Qingqing Zhu, Xiuying Chen, Pengfei Wu, JunFei Liu, and Dongyan Zhao. 2021. Combining curriculum learning and knowledge distillation for dialogue generation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 1284–1295.

A Technical Appendices

A.1 Preliminary Formulation of White-box KD

We provide background and a formal overview of white-box knowledge distillation (KD) for autoregressive student language models trained with a teacher large language model. Given a dataset of source–target sequence pairs (x, y), the student model is optimized using two objectives: (1) standard supervised fine-tuning (SFT) via cross-entropy loss between the ground-truth target y and the student's predicted distribution $q_\theta(y|x)$, and (2) a KD loss that aligns the student's token-level output distribution with that of the teacher at each position i.

The cross-entropy loss is defined as:

$$L_{ce} = -\sum_{i=1}^{|y|} \log q_\theta(y_i|x, y_{<i}), \tag{4}$$

where $q_\theta(y_i|x, y_{<i})$ denotes the student's predicted probability for token y_i, conditioned on input x and preceding tokens $y_{<i}$. This probability is obtained via softmax over the student's logits:

$$q_\theta(y_i|x, y_{<i}) = \frac{\exp(z^s_{y_i})}{\sum_{j\in V}\exp(z^s_j)}, \tag{5}$$

with $z^s_{y_i}$ the logit for token y_i and V the vocabulary.

The KD loss is formulated as:

$$L_{kd} = -\tau^2 \cdot \sum_{i=1}^{|y|} D\left(p(y_i|x, y_{<i}; \tau) \parallel q_\theta(y_i|x, y_{<i}; \tau)\right), \tag{6}$$

where $D(\cdot \parallel \cdot)$ denotes a divergence measure—commonly KL divergence, Jensen–Shannon divergence (JSD), or their symmetric variants—and $\tau > 0$ is the distillation temperature that smooths the output distributions. The temperature-scaled probabilities are:

$$q_\theta(y_i|x, y_{<i}; \tau) = \frac{\exp(z^s_{y_i}/\tau)}{\sum_{j\in V}\exp(z^s_j/\tau)}, \tag{7}$$

$$p(y_i|x, y_{<i}; \tau) = \frac{\exp(z^t_{y_i}/\tau)}{\sum_{j\in V}\exp(z^t_j/\tau)}, \tag{8}$$

with $z^t_{y_j}$ denoting the teacher's logits. Following Hinton et al. (2015), the τ^2 scaling ensures numerical stability and balances the magnitude of L_{kd} relative to L_{ce}.

The total training objective combines both losses:

$$L_s = \alpha \cdot L_{ce} + (1-\alpha) \cdot L_{kd}, \tag{9}$$

where $\alpha \in [0,1]$ controls the trade-off between SFT and distillation signals.

Recent work in white-box KD has explored alternative divergence functions $D(\cdot \parallel \cdot)$ in Eq. (6), including standard KL divergence (Hinton et al., 2015), reverse KL (RKL) (Gu et al., 2023), JSD (Agarwal et al., 2024), and symmetric variants such as SKL and SRKL (Ko et al., 2024). Full formulations of these divergences are provided in Appendix A.4.5.

A.2 Calibrated Progressive Distillation Training Algorithm

Algorithm 1 outlines the complete training pipeline of Calibrated Progressive Distillation, encompassing three core components: (1) the construction of a difficulty profile that captures how average hardness evolves when samples are sorted from easy to hard, (2) a Difficulty-Calibrated Curriculum Scheduling, which selects epoch-specific subsets to ensure a uniform increase in average difficulty—adapting subset size to the dataset's intrinsic hardness distribution, and (3) a Progressively Mixed Target Distribution for distillation, where the target is a convex combination of teacher and student predictions, with the mixing coefficient and distillation temperature both increasing linearly across epochs.

A.3 Details for Gradient Analysis

We provide a complete derivation of the gradient of the Progressively Mixed Target Distribution (PMTD) loss to support the stability claims in Section 4. Consider a single input–output pair (x, y); for notational simplicity, we denote the student's predicted probability for the true label as $q_\theta = q_\theta(y \mid x)$ and the teacher's as $p = p(y \mid x)$. Both are strictly positive and differentiable with respect to the student parameters θ. The mixed target at epoch e is defined as $p_t^{(e)} = t_e \cdot q_\theta + (1 - t_e) \cdot p$, where $t_e \in [0,1]$ is a fixed scalar (independent of θ), and p is treated as constant during backpropagation.

The PMTD loss is the asymmetric Kullback–Leibler divergence: $\mathcal{L}_{\text{PMTD}}^{(e)} = D_{\text{KL}}\left(p_t^{(e)} \,\middle\|\, q_\theta\right) = p_t^{(e)} \log \frac{p_t^{(e)}}{q_\theta}$.

We compute its gradient with respect to θ:

Algorithm 1 Calibrated Progressive Distillation (CPD)

Input: $D = \{(x_i, y_i)\}_{i=1}^N$: training dataset; q_θ: untrained student LLM; p: teacher LLM; n: total epochs; τ_0, τ_n: initial and final distillation temperatures.
Output: q_θ^*: distilled student LLM.
for $i = 1$ to N **do**
  $\mathcal{L}_i \leftarrow -\log q_\theta(y_i \mid x_i)$
end for
Sort D by $\mathcal{L}_i$ in ascending order $\rightarrow D_{\text{ds}} = \{s_1, \ldots, s_N\}$
for $k = 1$ to N **do**
  $\bar{\mathcal{L}}(k) \leftarrow \frac{1}{k} \sum_{j=1}^{k} \mathcal{L}_{s_j}$
end for
$\bar{\mathcal{L}}_{\text{full}} \leftarrow \bar{\mathcal{L}}(N)$
for $e = 1$ to n **do**
  $k_e \leftarrow \min\left\{k \,\middle|\, \bar{\mathcal{L}}(k) \geq \frac{e}{n} \cdot \bar{\mathcal{L}}_{\text{full}}\right\}$
  $D^{(e)} \leftarrow \{s_1, \ldots, s_{k_e}\}$
  $t_e \leftarrow \frac{e}{n}$
  $\tau_e \leftarrow \tau_0 + (\tau_n - \tau_0) \cdot \frac{e-1}{n-1}$
end for
for $e = 1$ to n **do**
  $p_t^{(e)}(y|x;\tau_e) \leftarrow t_e \cdot q_\theta(y|x;\tau_e) + (1-t_e) \cdot p(y|x;\tau_e)$
  $\mathcal{L}_{\text{KD}}^{(e)} \leftarrow D_{\text{PMTD}}\left(p_t^{(e)}(\cdot|x;\tau_e) \,\middle\|\, q_\theta(\cdot|x;\tau_e)\right)$
  while not converged for fixed number of steps **do**
    Update q_θ by minimizing $\mathcal{L}_{\text{KD}}^{(e)}$ on $D^{(e)}$
  end while
end for
return $q_\theta^* \leftarrow q_\theta$

$$\begin{aligned} &\nabla_\theta \mathcal{L}_{\text{PMTD}}^{(e)} \\ &= \nabla_\theta \left[p_t^{(e)} \log p_t^{(e)} - p_t^{(e)} \log q_\theta \right]. \end{aligned} \tag{10}$$

We differentiate each term separately. For the first term:

$$\begin{aligned} &\nabla_\theta \left[p_t^{(e)} \log p_t^{(e)} \right] = \left(\log p_t^{(e)} + 1 \right) \nabla_\theta p_t^{(e)} \\ &= t_e \left(\log p_t^{(e)} + 1 \right) \nabla_\theta q_\theta, \end{aligned} \tag{11}$$

since $\nabla_\theta p_t^{(e)} = t_e \nabla_\theta q_\theta$.

For the second term, applying the product rule:

$$\begin{aligned} &\nabla_\theta \left[p_t^{(e)} \log q_\theta \right] \\ &= \left(\nabla_\theta p_t^{(e)} \right) \log q_\theta + p_t^{(e)} \cdot \tfrac{1}{q_\theta} \nabla_\theta q_\theta \\ &= t_e \log q_\theta \cdot \nabla_\theta q_\theta + \tfrac{p_t^{(e)}}{q_\theta} \nabla_\theta q_\theta. \end{aligned} \tag{12}$$

Subtracting (12) from (11), we obtain:

$$\begin{aligned} &\nabla_\theta \mathcal{L}_{\text{PMTD}}^{(e)} \\ &= t_e \left(\log p_t^{(e)} + 1 \right) \nabla_\theta q_\theta - t_e \log q_\theta \cdot \nabla_\theta q_\theta - \tfrac{p_t^{(e)}}{q_\theta} \nabla_\theta q_\theta \\ &= \left[t_e \log \tfrac{p_t^{(e)}}{q_\theta} + t_e - \tfrac{p_t^{(e)}}{q_\theta} \right] \nabla_\theta q_\theta. \end{aligned}$$

Thus, the per-example gradient takes the form:

$$\begin{aligned} &\nabla_\theta \mathcal{L}_{\text{PMTD}}^{(e)} = K^{(e)} \cdot \nabla_\theta q_\theta, \\ &\text{where} \quad K^{(e)} = t_e \log \tfrac{p_t^{(e)}}{q_\theta} + t_e - \tfrac{p_t^{(e)}}{q_\theta}. \end{aligned} \tag{13}$$

This expression underpins the boundedness and adaptive emphasis properties analyzed in Section 4.

Crucially, because $p_t^{(e)}$ is a convex combination of q_θ and p, the ratio $p_t^{(e)}/q_\theta$ is always finite and positive, preventing the unbounded gradients that can arise in standard KL distillation when $q_\theta \to 0$ while p remains fixed. The mixing coefficient t_e further regularizes the magnitude of $K^{(e)}$, ensuring stable optimization throughout training.

A.4 Detailed Experimental Setup

This section outlines the complete experimental configuration, including model architectures, datasets, training protocols, evaluation procedures, and baseline methods used in our empirical analysis.

A.4.1 Base Models Description

Our study encompasses three prominent families of large language models—GPT-2, OpenLLaMA2, and Qwen2.5—selected to represent a broad spectrum of architectural designs, training data sources, and target applications. This diversity enables us to evaluate the robustness and generalizability of our method across both general-purpose and domain-specialized scenarios. Specifically, GPT-2 and OpenLLaMA2 are employed for instruction-following tasks, while Qwen2.5 variants are used for mathematical reasoning and code generation. These models were chosen for their public availability, architectural heterogeneity, and widespread adoption in standard benchmarks, ensuring reproducibility and meaningful comparative evaluation.

- **Qwen2.5** (Team, 2024): A suite of domain-specialized LLMs from the Qwen team, fine-tuned for high-fidelity performance in technical reasoning tasks. For instruction following, we use Qwen2.5-1.5B as the teacher and Qwen2.5-0.5B as the student; For mathematical reasoning, we use Qwen2.5-Math-7B-Inst as the teacher and Qwen2.5-Math-1.5B as the student; for code generation, we employ Qwen2.5-Coder-7B-Inst and Qwen2.5-Coder-1.5B, respectively. These purpose-built pairs enable a precise assessment of distillation effectiveness in architectures explicitly optimized for vertical domains.

- **GPT-2** (Radford et al., 2019): A decoder-only Transformer originally introduced by OpenAI. In our distillation framework, we use GPT-2 (1.5B parameters) as the teacher and GPT-2 (0.1B parameters) as the student, both adapted via instruction tuning. This intra-architecture setup—featuring a significant capacity disparity—provides a controlled setting to study knowledge transfer within a consistent modeling paradigm.

- **OpenLLaMA2** (Geng and Liu, 2023): An open-source, permissively licensed reimplementation of Meta's LLaMA, trained solely on the publicly available RedPajama corpus (Computer, 2023), in contrast to the proprietary data used for the original LLaMA (Touvron et al., 2023). We adopt OpenLLaMA2-7B as the teacher and OpenLLaMA2-3B as the student for instruction-following tasks. This configuration allows us to examine distillation dynamics in models developed under transparent, community-driven training regimes.

All models are initialized from official public checkpoints, and identical optimization settings and training procedures are applied across all distillation experiments to ensure a fair and controlled comparison.

A.4.2 Dataset Description

We evaluate the CPD framework across three core NLP domains: instruction-following, mathematical reasoning, and code generation. Below, we describe the datasets used for training and evaluation in each category.

- `databricks-dolly-15k` (instruction-following; (Conover et al., 2023)): A human-authored dataset of 15,000 instruction–response pairs spanning seven task types, including brainstorming, closed QA, and summarization. Its naturally phrased, high-quality prompts make it ideal for assessing general instruction-following ability.

- `self-instruct-eval` (instruction-following; (Wang et al., 2022a)): A synthetically generated dataset created via self-instruct bootstrapping. We use its training split (52K instructions yielding 82K input–output pairs) and evaluation set (252 expert-defined tasks with 50K public examples) to test generalization to unseen task formulations.

- `vicuna-eval` (instruction-following; (Chiang et al., 2023)): A benchmark of 80 complex,

open-ended prompts requiring multi-step reasoning and contextual understanding. Widely used for conversational model evaluation, it provides a rigorous test of instruction adherence and reasoning depth.

- `super_natural-instructions` (instruction-following; (Wang et al., 2022b)): A large-scale collection of 1,616 expert-written NLP tasks across 76 categories. Its test set includes 9,000 samples from 119 held-out tasks, enabling robust zero-shot generalization evaluation.

- `unnatural-instructions-core` (instruction-following; (Honovich et al., 2022)): A dataset of 66,000 instruction–response pairs generated by automatically perturbing natural instructions. The core subset is used to examine model behavior under predominantly synthetic training data.

- `MetaMathQA` (mathematical reasoning; (Yu et al., 2023)): A dataset of 39,500 math problems produced through iterative forward–backward reasoning transformations. It emphasizes diverse solution strategies and is designed to improve generalization in mathematical reasoning.

- `GSM8K` (mathematical reasoning; (Cobbe et al., 2021)): A set of 8,500 grade-school math word problems requiring multi-step arithmetic reasoning. Each problem demands explicit chain-of-thought decomposition, making it a standard benchmark for evaluating reasoning fidelity in distilled models.

- `MATH` (mathematical reasoning; (Hendrycks et al., 2021)): A collection of competition-level math problems covering algebra, geometry, number theory, and more, with varying difficulty. It targets deep mathematical understanding and symbolic reasoning capabilities.

- `Evol-Instruct-Code-80k-v1` (code generation; (Luo et al., 2023)): Built using the Evol-Instruct framework, this dataset starts from Code Alpaca (20K samples) and applies iterative instruction evolution techniques—including constraint injection, reasoning depth amplification, misleading code insertion, and complexity escalation—to yield 78K progressively challenging code instructions. It has demonstrated strong empirical gains in code fine-tuning.

- `MBPP` (code generation; (Austin et al., 2021)): A crowd-sourced dataset of 974 programming problems, each with a natural language description (docstring), function signature, and three test cases. A verified subset ensures high solution correctness, supporting reliable evaluation.

- `HumanEval` (code generation; (Chen et al., 2021)): A set of 164 handcrafted coding tasks, each providing a function signature and docstring. Designed to be absent from common pretraining corpora, it offers an unbiased assessment of code synthesis performance.

A.4.3 Training Details

All experiments are conducted on a consistent hardware platform consisting of four NVIDIA A100 80GB GPUs and an Intel(R) Xeon(R) Platinum 8350C CPU to ensure reproducibility and comparability across training runs.

Hyperparameter Settings. We employ a unified set of hyperparameters for the CPD framework across all tasks. The distillation temperature τ is linearly increased from an initial value of $\tau_1 = 1$ to a final value of $\tau_n = 2$. For off-policy distillation, the supervised fine-tuning (SFT) loss weight is initialized at $\alpha_0 = 0.3$. In contrast, on-policy distillation excludes the SFT term entirely ($\alpha = 0$ throughout), as the student learns exclusively from its own generations refined via teacher feedback. Following Agarwal et al. (2024), we adopt a balanced mixing strategy that combines 50% student-generated outputs with 50% ground-truth responses to stabilize training and enhance generalization.

For baseline methods, we adopt recommended settings from prior work: the SKL and SRKL mix ratios (α_{SKL}, α_{SRKL}) are set to 0.5 (Ko et al., 2024); ABKD uses $\alpha_{\text{abkd}} = 0.2$ and $\beta_{\text{abkd}} = 0.7$ (Wang et al., 2025); TAID applies a linear schedule that increases its mix ratio from 0 to 1 (Shing et al., 2025); and DA-KD uses $\lambda_{\text{dakd}} = 0.9$ (He et al., 2025).

Model checkpoint selection is task-specific: for instruction-following, we select the best checkpoint based on Rouge-L scores on the validation set, which correlate well with human judgments in instruction and summarization tasks (Agarwal

		GPT-2-1.5B (M_t) → GPT-2-0.1B (M_s)											
		DollyEval		**SelfInst**		**VicunaEval**		**S-NI**		**UnNI**		**Avg.**	
Categories	KD Methods	*R-L*	*WR*	*R-L*	*WR*	*R-L*	*WR*	*R-L*	*WR*	*R-L*	*WR*	*R-L*	*WR*
	M_t	27.43	52.79	14.12	52.15	16.37	52.85	27.68	53.44	31.92	52.95	23.50	52.84
	SFT	23.33	47.04	10.01	42.28	14.72	48.31	16.38	36.32	19.57	39.37	16.80	42.67
(1)	KLD	23.49	47.30	10.33	42.78	14.96	49.11	19.71	40.80	22.01	42.04	18.10	44.41
	RKL	23.79	47.87	12.13	46.96	14.94	48.67	23.81	45.38	22.52	42.63	19.44	46.30
	TVD	24.32	48.25	11.09	45.01	15.51	49.70	25.93	47.66	26.55	46.96	20.68	47.52
(2)	GKD (on-policy)	24.67	48.57	11.48	45.51	15.66	49.94	23.81	45.60	25.26	45.42	20.18	47.01
	JSD	24.07	48.66	11.38	45.59	15.87	50.79	22.84	44.20	23.06	43.30	19.44	46.51
	SKL	24.24	48.64	12.27	47.23	15.71	50.15	23.33	45.06	24.02	44.25	19.91	47.07
	SRKL	25.22	49.13	12.86	48.21	15.18	49.18	25.51	47.18	28.43	48.59	21.44	48.46
	ABKD	25.63	50.46	**13.35**	**49.56**	16.05	51.82	26.44	47.99	29.12	49.01	22.12	49.77
	TAID	25.42	49.89	13.18	47.93	15.85	50.72	26.31	47.78	28.87	48.87	21.93	49.04
(3)	KLD+POCL	24.87	49.85	11.56	46.19	16.13	52.01	21.59	42.51	24.34	44.66	19.70	47.04
(4)	DA-KD	25.68	50.65	12.96	48.87	15.83	49.72	25.89	46.39	28.54	48.67	21.78	48.86
	CPD	**26.08**	**51.14**	13.34	49.41	**16.75**	**52.22**	**27.32**	**48.94**	**30.02**	**50.05**	**22.70**	**50.35**
		OpenLLaMA2-7B (M_t) → OpenLLaMA2-3B (M_s)											
		DollyEval		**SelfInst**		**VicunaEval**		**S-NI**		**UnNI**		**Avg.**	
Categories	KD Methods	*R-L*	*WR*	*R-L*	*WR*	*R-L*	*WR*	*R-L*	*WR*	*R-L*	*WR*	*R-L*	*WR*
	M_t	26.26	52.03	19.15	51.12	17.31	49.39	31.20	51.34	31.84	51.05	25.15	50.98
	SFT	24.87	49.20	18.78	50.29	16.50	48.99	28.65	47.87	28.73	47.61	23.51	48.79
(1)	KLD	23.76	49.68	17.48	48.05	15.68	47.97	28.42	48.10	27.48	46.75	22.56	48.11
	RKL	26.67	50.78	18.78	49.71	**18.43**	**53.67**	31.27	50.42	32.01	52.18	25.43	51.35
	TVD	24.62	48.57	18.32	48.90	15.85	48.49	29.01	48.54	28.08	47.54	23.18	48.41
(2)	GKD (on-policy)	26.30	50.50	18.91	50.06	17.53	50.92	34.21	52.38	31.67	50.46	25.72	50.87
	JSD	25.64	50.34	17.18	47.54	15.56	47.58	27.93	49.29	27.39	46.48	22.74	48.25
	SKL	24.99	49.08	18.77	49.54	16.69	49.92	29.71	50.86	28.49	48.02	23.73	49.48
	SRKL	25.99	52.06	19.11	50.16	16.93	50.36	30.07	49.94	28.81	48.38	24.18	50.18
	ABKD	**27.18**	52.20	19.35	51.17	17.83	53.70	34.51	55.69	32.65	53.50	26.30	53.25
	TAID	27.05	**52.32**	19.18	51.25	16.73	50.48	33.96	55.36	32.13	52.06	25.81	52.29
(3)	KLD+POCL	26.24	51.19	17.40	45.99	16.98	50.82	34.00	54.84	33.43	53.07	25.61	51.18
(4)	DA-KD	26.85	51.77	18.95	50.39	17.54	52.08	32.81	53.42	32.51	52.59	25.73	52.05
	CPD	26.92	52.15	**19.57**	**52.24**	17.67	52.48	**35.76**	**57.75**	**34.53**	**54.32**	**26.89**	**53.79**

Table 6: Instruction-following evaluation on GPT-2 (1.5B→0.1B) and OpenLLaMA2 (7B→3B). Reported metrics are ROUGE-L (R-L) and winning rate (WR) on DollyEval, SelfInst, VicunaEval, S-NI, and UnNI; Avg. denotes mean across benchmarks. red indicates the student outperforms the teacher. Results are averaged over five random seeds. Best student results are in **bold**.

et al., 2024); for mathematical reasoning and code generation, we use *pass@1* on the validation set as the selection criterion.

Instruction-Following Experiments. We use the `databricks-dolly-15k` dataset (Conover et al., 2023), randomly partitioned into 12.5K training, 1K validation, and 0.5K test samples. Instances exceeding the model's maximum context length are filtered prior to training. Both teacher models—Qwen2.5 (1.5B), GPT-2 (1.5B) and OpenLLaMA2 (7B)—are first fine-tuned on this dataset before distillation.

Hyperparameter tuning is performed via grid search over learning rates {5e-4, 1e-4, 5e-5} and batch sizes {8, 16}, guided by validation performance. All non-curriculum baselines—including GKD (on-policy), KLD, RKL, JSD, TVD, SKL, SRKL, ABKD, and TAID—are trained for 20 epochs.

For curriculum-based methods, KLD+POCL (Liu and Zhang, 2025) follows a staged schedule: 25%, 50%, 75%, and 100% of the training data are used for 8 epochs each, totaling 32 epochs. However, the total number of training iterations is matched to that of the 20-epoch baselines by adjusting batch composition. DA-KD (He et al., 2025) is trained for 20 epochs using a linear decay schedule for data selection, resulting in 52.5% of the total iterations of the standard baselines. Our method, CPD, also trains for 20 epochs using the proposed difficulty-calibrated curriculum. To ensure fair comparison, each epoch's training subset is duplicated (via simple repetition) to increase its effective size, yielding a total iteration count equal to 82.3% of the baseline runs.

For OpenLLaMA2, we apply Low-Rank Adaptation (LoRA) (Hu et al., 2022), a parameter-efficient fine-tuning approach. All linear layers in the Transformer's self-attention and MLP modules are configured as LoRA target modules.

Mathematical Reasoning Experiments. We sample 20K training and 2K validation examples from the `MetaMathQA` dataset (Yu et al., 2023). To promote structured, step-by-step reasoning, each prompt is prefixed with the instruction: *"Please reason step by step, and put your final answer within* `\boxed{}`*."* following the chain-of-thought prompting paradigm (Wei et al., 2022).

All non-curriculum baselines are trained for 4 epochs with a fixed learning rate of 5×10^{-5}. KLD+POCL (Liu and Zhang, 2025) uses 8 epochs under the same learning rate, with a staged curriculum: 25%, 50%, 75%, and 100% of the training data are each trained for 2 epochs. DA-KD (He et al., 2025) is trained for 4 epochs with a fixed learning rate of 5×10^{-5}, where each epoch follows a linear decay schedule for data selection; to ensure fair comparison, the epoch-specific train-

GPT-2-1.5B (M_t) → GPT-2-0.1B (M_s)						
KD Methods	**DollyEval**	**SelfInst**	**VicunaEval**	**S-NI**	**UnNI**	**Avg.**
SFT	23.33	10.01	14.72	16.38	19.57	16.80
GKD	24.67	11.48	15.66	23.81	25.26	20.18
KLD	23.49	10.33	14.96	19.71	22.01	18.10
RKL	23.79	12.13	14.94	23.81	22.52	19.44
SKL	24.24	12.27	15.71	23.33	24.02	19.91
ABKD	25.63	13.35	16.05	26.44	29.12	22.12
KLD+POCL	24.87	11.56	16.13	21.59	24.34	19.70
DA-KD	25.68	12.96	15.83	25.89	28.54	21.78
CPD	26.08	13.34	16.75	27.32	30.02	22.70
GPT-2-1.5B (M_t) → GPT-2-0.3B (M_s)						
SFT	25.25	13.34	16.17	23.77	27.27	21.16
GKD	24.51	14.26	16.86	26.05	29.05	22.15
KLD	24.75	12.84	16.14	24.00	27.20	20.99
RKL	25.49	14.30	18.09	26.74	30.68	23.06
SKL	26.11	14.35	18.12	27.48	31.87	23.59
ABKD	27.02	15.14	19.21	29.24	32.57	24.64
KLD+POCL	26.13	14.75	18.45	28.74	32.19	24.05
DA-KD	26.44	15.03	18.67	29.08	32.86	24.42
CPD	27.32	15.87	19.53	29.59	33.04	25.07
GPT-2-1.5B (M_t) → GPT-2-0.8B (M_s)						
SFT	25.19	13.40	16.10	23.89	26.82	21.08
GKD	25.52	14.25	16.82	27.50	30.48	22.91
KLD	26.51	13.99	17.07	26.48	30.12	22.83
RKL	26.45	15.23	18.29	29.81	33.80	24.72
SKL	27.61	15.36	18.28	30.91	34.59	25.35
ABKD	28.72	15.95	18.35	31.53	35.34	25.98
KLD+POCL	28.03	15.48	18.15	31.14	34.87	25.53
DA-KD	28.61	15.78	18.42	31.38	35.14	25.87
CPD	29.03	16.14	18.58	31.67	35.41	26.17

Table 7: Performance (ROUGE-L) of GPT-2 distillation across five instruction-following evaluation datasets under varying teacher-student capacity gaps. The teacher is GPT-2-1.5B, distilled to students of sizes 0.1B, 0.3B, and 0.8B. DA-KD results are included for comparison. Avg. shows the average score across all five datasets.

ing subset is duplicated (via simple repetition) to double its effective size. Similarly, CPD is trained for 4 epochs with the same learning rate, using the proposed difficulty-calibrated curriculum schedule, and also doubles each epoch's subset via repetition to align iteration counts with baselines.

Across all configurations, we maximize the per-GPU batch size within memory limits and employ gradient accumulation to maintain an effective batch size of 128. LoRA (Hu et al., 2022) is applied with identical target modules as in the instruction-following experiments (i.e., all linear layers in self-attention and MLP subnetworks).

Code Generation Experiments. We select 20K training and 2K validation samples from the `Evol-Instruct-Code-80k-v1` dataset (Luo et al., 2023). The training protocol mirrors that of the mathematical reasoning experiments: non-curriculum baselines train for 4 epochs at a fixed learning rate of 5×10^{-5}; KLD+POCL uses 8 epochs with staged data exposure (2 epochs per curriculum stage); DA-KD trains for 4 epochs with linear decay-based data scheduling and subset duplication; and CPD trains for 4 epochs using its difficulty-calibrated curriculum with the same duplication strategy for fair iteration matching.

Gradient accumulation is again used to achieve an effective batch size of 128 across four A100 GPUs. LoRA (Hu et al., 2022) is consistently applied to all linear layers in the self-attention and MLP components to ensure parameter efficiency and training stability.

A.4.4 Evaluation Details

All evaluations are performed on a single NVIDIA A100 80GB GPU, following established protocols from prior literature: we adopt the evaluation setup of Gu et al. (2023) for instruction-following, that of Yang et al. (2024b) for mathematical reasoning, and the methodology of Hui et al. (2024) for code generation.

Instruction-Following Experiments. To ensure consistent and unbiased inference, we apply a fixed prompt template (Fig. 5) uniformly across all models. Responses are generated with a temperature of 1.0 and a maximum output length of 512 tokens. To reduce stochastic variability and improve result reliability, we generate five independent responses per input using distinct random seeds ({10, 20, 30, 40, 50}) and report metrics averaged over these replicates.

For LLM-as-a-Judge evaluation (Zheng et al., 2023), we employ the pairwise comparison prompt shown in Fig. 6, configured with a temperature of 0.7. Within the GPT-2 family, student outputs are compared against those of the vanilla GPT-2 (1.5B) teacher; for the OpenLLaMA2 family, comparisons

KD Methods	GPT-2-1.5B (M_t) → GPT-2-0.1B (M_s)					Qwen2.5-Math-7B-Inst (M_t) → Qwen2.5-Math-1.5B (M_s)	
	DollyEval	**SelfInst**	**VicunaEval**	**S-NI**	**UnNI**	**GSM8K**	**MATH**
	Rouge-L	*Rouge-L*	*Rouge-L*	*Rouge-L*	*Rouge-L*	*pass@1*	*pass@1*
CPD (CE-LOSS)	26.08	13.34	16.75	27.32	30.02	48.5	64.4
CPD (ROUGE-L)	26.04	13.15	16.64	26.83	29.75	46.2	62.1
CPD (DDS)	26.14	13.31	16.46	27.18	29.96	48.1	64.6
CPD (SE)	25.31	11.85	14.79	25.42	26.96	46.5	62.9

Table 8: Ablation on difficulty metrics for Difficulty Profile construction. CE-LOSS = cross-entropy loss; DDS = distillation difficulty score; SE = sentence-level entropy. Avg. denotes mean performance across five instruction-following benchmarks (ROUGE-L) and two math reasoning benchmarks (pass@1).

```
Below is an insruction that describes a task.
Write a response that appropriately completes the request.

### Instruction:
{instruction}

### Input:
{input}

### Response:
```

Figure 5: Prompt template used in instruction-following experiments, adapted from (Gu et al., 2023)

are made relative to the vanilla OpenLLaMA2 (7B) teacher.

```
### System:
Please act as an impartial judge and evaluate the quality of
the responses provided by two AI assistants to the user
question displayed below. You should choose the assistant
that follows the user's instructions and answers the user's
question better. Your evaluation should consider factors such
as the helpfulness, relevance, accuracy, depth, creativity,
and level of detail of their responses. Begin your evaluation
by comparing the two responses and provide a short
explanation. Avoid any position biases and ensure that the
order in which the responses were presented does not
influence your decision. Do not allow the length of the
responses to influence your evaluation. Do not favor certain
names of the assistants. Be as objective as possible. After
providing your explanation, output your final verdict by
strictly following this format: "[[A]]" if assistant A is
better, "[[B]]" if assistant B is better, and "[[C]]" for
a tie.

### User Question:
{question}

[The Start of Assistant A's Answer]
{answer a}
[The End of Assistant A's Answer]

[The Start of Assistant B's Answer]
{answer b}
[The End of Assistant B's Answer]
```

Figure 6: Pairwise comparison prompt used in LLM-as-a-Judge evaluation, adapted from (Zheng et al., 2023)

Mathematical Reasoning & Code Generation Experiments. We use task-specific prompt templates during inference: Fig. 7 for mathematical reasoning and Fig. 8 for code generation. Following Yang et al. (2024b), we employ an 8-shot prompt for GSM8K and a 4-shot prompt for MATH to elicit chain-of-thought reasoning, with all in-context examples drawn from their work. Generation is performed using greedy decoding with a maximum output length of 1024 tokens to accommodate extended reasoning traces. For code generation, we evaluate model outputs using the EvalPlus framework (Liu et al., 2023), which extends standard test suites with additional hidden test cases to enable more robust correctness assessment.

```
<8 examples> # for GSM8K
<4 examples> # for MATH

### Problem:
{instruction}
Please reason step by step, and put your final answer within
\boxed{}.

### Solution:
```

Figure 7: Prompt template used in mathematical reasoning experiments, adapted from (Yang et al., 2024b)

```
Please provide a self-contained Python script that solves the
following problem in a markdown code block:

### Problem:
{instruction}

### Response:
Below is a Python script with a self-contained function that
solves the problem and passes corresponding tests:
```

Figure 8: Prompt template used in code generation experiments, adapted from (Hui et al., 2024)

A.4.5 Baseline Description

This section details the knowledge distillation (KD) loss functions included in our comparative study. Each baseline quantifies the discrepancy between the teacher's output distribution p and the student's distribution q_θ using a distinct statistical divergence. To ensure a comprehensive and representative evaluation, we include both classical and recently proposed divergence-based objectives from the distillation literature.

The Kullback–Leibler Divergence (KLD) (Hinton et al., 2015), a standard choice in KD, is defined as:

$$D_{\mathrm{KLD}}(p \parallel q_\theta) = \mathbb{E}_{y\sim p}\left[\log \frac{p(y)}{q_\theta(y)}\right], \tag{14}$$

where the expectation is taken over tokens sampled from the teacher distribution p. This objective encourages the student to assign high probability to outputs deemed likely by the teacher.

The Reverse KL Divergence (RKL) swaps the

	GPT-2-1.5B (M_t) → GPT-2-0.1B (M_s)					Qwen2.5-Math-7B-Inst (M_t) → Qwen2.5-Math-1.5B (M_s)	
	DollyEval	**SelfInst**	**VicunaEval**	**S-NI**	**UnNI**	**GSM8K**	**MATH**
KD Methods	*Rouge-L*	*Rouge-L*	*Rouge-L*	*Rouge-L*	*Rouge-L*	*pass@1*	*pass@1*
CPD (DCS)	26.08	13.34	16.75	27.32	30.02	48.5	64.4
CPD (BSS)	25.75	12.89	15.91	26.50	28.40	47.9	63.5
CPD (LS)	24.35	12.15	14.98	26.18	27.88	46.6	62.1
CPD (CS)	25.26	12.37	15.67	26.58	28.36	48.2	63.8

Table 9: Ablation on curriculum scheduling strategies. DCS = Difficulty-Calibrated Scheduling (proposed); BSS = baby-step scheduling (Liu and Zhang, 2025); LS = linear growth; CS = cosine growth (He et al., 2025). Avg. denotes mean performance across five instruction-following benchmarks (ROUGE-L) and two math reasoning benchmarks (pass@1).

roles of the distributions:

$$D_{\mathrm{RKL}}(q_\theta \parallel p) = \mathbb{E}_{y\sim p}\left[\log \frac{q_\theta(y)}{p(y)}\right]. \tag{15}$$

Unlike KLD, RKL places greater emphasis on aligning the student's high-probability regions with those of the teacher and has been shown to improve training stability in certain distillation regimes.

The Jensen–Shannon Divergence (JSD) (Agarwal et al., 2024) offers a symmetric and smoothed alternative:

$$D_{\mathrm{JSD}}(p, q_\theta) = \frac{1}{2}\mathbb{E}_{y\sim p}\left[\log \frac{p(y)}{m(y)}\right] + \frac{1}{2}\mathbb{E}_{y\sim q_\theta}\left[\log \frac{q_\theta(y)}{m(y)}\right], \tag{16}$$

where $m(y) = \frac{1}{2}p(y) + \frac{1}{2}q_\theta(y)$ is the average (midpoint) distribution. JSD promotes bidirectional alignment and is less sensitive to distributional mismatches than asymmetric divergences.

The Total Variation Distance (TVD) (Wen et al., 2023) measures the ℓ_1 difference between the two distributions:

$$D_{\mathrm{TVD}}(p, q_\theta) = \frac{1}{2}\sum_y |p(y) - q_\theta(y)|. \tag{17}$$

Bounded in $[0, 1]$, TVD provides intuitive interpretability and robustness to outliers, though it yields zero gradients in regions where the supports of p and q_θ do not overlap.

Skewed KL (SKL) and Skewed Reverse KL (SRKL) (Ko et al., 2024) introduce interpolation in the reference distribution to mitigate extreme gradients:

$$D_{\mathrm{SKL}}(p \parallel q_\theta) = D_{\mathrm{KLD}}\left(p \parallel \beta p + (1-\beta)q_\theta\right), \tag{18}$$

$$D_{\mathrm{SRKL}}(q_\theta \parallel p) = D_{\mathrm{KLD}}\left(q_\theta \parallel (1-\beta)p + \beta q_\theta\right), \tag{19}$$

where $\beta \in [0, 1]$ controls the mixing weight. These variants enhance training stability by smoothing the target distribution.

Adaptive Beta Knowledge Distillation (ABKD) (Wang et al., 2025) defines a generalized divergence based on power means:

$$D_{\mathrm{ABKD}}(p \parallel q_\theta) = -\frac{1}{\alpha\beta}\sum_k \left[p(k)^\alpha q_\theta(k)^\beta - \frac{\alpha}{\alpha+\beta}p(k)^{\alpha+\beta} - \frac{\beta}{\alpha+\beta}q_\theta(k)^{\alpha+\beta}\right], \tag{20}$$

where $p = [p(k)]_{k=1}^C$ and $q_\theta = [q_\theta(k)]_{k=1}^C$ are discrete distributions over C classes, and $\alpha, \beta > 0$ are tunable parameters.

Time-Adaptive Interpolated Distillation (TAID) (Shing et al., 2025) employs a dynamic interpolation between teacher and student logits:

$$D_{\mathrm{TAID}}(p \parallel q_\theta) = D_{\mathrm{KLD}}\left(p_t \parallel q_\theta\right), \tag{21}$$

where $p_t = \mathrm{softmax}\big((1-t)\cdot z^s + t\cdot z^t\big)$, with z^s and z^t denoting student and teacher logits, respectively, and $t \in [0, 1]$ is a time-dependent mixing coefficient that evolves during training.

Finally, the Bidirectional Discrepancy Loss (BDL) from DA-KD (He et al., 2025) is formulated as:

$$D_{\mathrm{BDL}}(p \parallel q_\theta) = D_{\mathrm{KLD}}\left((1-\lambda)p + \lambda q_\theta \parallel \lambda p + (1-\lambda)q_\theta\right), \tag{22}$$

where $\lambda \in [0, 1]$ balances the contribution of the teacher and student distributions in the mixed reference and target. This symmetric construction encourages mutual consistency between the two models.

A.5 Additional Experimental Results

This section presents supplementary empirical results that further substantiate the effectiveness and robustness of the CPD framework.

Tab. 6 reports performance metrics for GPT-2 and OpenLLaMA2 as the student model, including ROUGE-L scores and pairwise winning rates averaged over five random seeds. These results confirm that CPD consistently outperforms baseline distillation methods and generalizes well across different large language model architectures.

Tab. 7 reports performance (ROUGE-L) of GPT-2 distillation across five instruction-following evaluation datasets under varying teacher-student capacity gaps.

Tab. 8 presents an ablation study analyzing the influence of alternative difficulty metrics—used to

	GPT-2-1.5B (M_t) → GPT-2-0.1B (M_s)					Qwen2.5-Math-7B-Inst (M_t) → Qwen2.5-Math-1.5B (M_s)	
	DollyEval	**SelfInst**	**VicunaEval**	**S-NI**	**UnNI**	**GSM8K**	**MATH**
KD Methods	*Rouge-L*	*Rouge-L*	*Rouge-L*	*Rouge-L*	*Rouge-L*	*pass@1*	*pass@1*
CPD (linear)	26.08	13.34	16.75	27.32	30.02	48.5	64.4
CPD (fix)	25.31	12.73	15.70	26.52	27.85	46.1	62.3
CPD (adaptive)	26.11	13.32	16.82	27.34	29.82	48.6	64.5

Table 10: Ablation on mixture ratio scheduling: linear (proposed), fixed (Ko et al., 2024), and adaptive (Shing et al., 2025). Avg. denotes mean performance across five instruction-following benchmarks (ROUGE-L) and two math reasoning benchmarks (pass@1).

construct the instance-level difficulty profile—on CPD's downstream performance. This highlights the sensitivity of the framework to the choice of difficulty estimator and motivates our design decisions.

Tab. 9 investigates the impact of different calibrated curriculum scheduling strategies—governing how the training subset evolves across epochs—on distillation outcomes, demonstrating the importance of aligning curriculum progression with empirical difficulty.

Tab. 10 examines the effect of various dynamic mixing strategies for the target distribution on CPD's performance, underscoring the benefit of co-designing mixing dynamics with the curriculum schedule.

A.6 Related Work

White-Box Knowledge Distillation for LLMs. White-box knowledge distillation for large language models leverages access to the teacher's internal logits or soft probability distributions, enabling richer supervision than black-box alternatives (Yang et al., 2024c). A major line of research in this setting focuses on designing effective divergence measures for the distillation loss. While Kullback–Leibler divergence (KLD) remains widely used (Hinton et al., 2015), its asymmetry can lead to mode-averaging effects that compromise distillation fidelity (Gu et al., 2023). To mitigate this, recent works have explored alternatives such as reverse KLD (RKL) (Gu et al., 2023), Jensen–Shannon divergence (JSD) (Agarwal et al., 2024), skewed variants (SKL and SRKL) (Ko et al., 2024), α-β-divergence (ABKD) (Wang et al., 2025), and task-aware interpolated divergence (TAID) (Shing et al., 2025). Despite promising results in specific scenarios, these loss-centric approaches often exhibit inconsistent performance across tasks and datasets (Agarwal et al., 2024; Ko et al., 2024), highlighting the need for complementary strategies that go beyond the choice of divergence.

Curriculum Learning in Knowledge Distillation. Curriculum learning (CL) has been investigated to structure the sequencing of training samples, though its application has been predominantly explored in computer vision (Xiang et al., 2020; Li et al., 2023) with limited adoption in NLP (Zhu et al., 2021). Recent KD-specific CL approaches include MPDistil (Sengupta et al., 2023), which uses reinforcement learning for curriculum scheduling; Confucius (Gao et al., 2024), tailored for tool-augmented learning via black-box KD; POCL (Liu and Zhang, 2025), a plug-and-play framework inspired by progressive overload; and DA-KD (He et al., 2025), which dynamically adjusts the training set based on sample difficulty. Nevertheless, these methods typically treat curriculum design in isolation and fail to jointly model the interplay among instance difficulty, target fidelity, and the student's evolving capacity—a gap that CPD explicitly addresses in the white-box KD setting.