TURNING THE TIDE: CROSS-ARCHITECTURE DISTILLATION FOR DIFFUSION LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion large language models (dLLMs) support parallel decoding and bidirectional context, but competitive dLLMs have 8–16B parameters, which limits deployment. Knowledge distillation is a natural remedy, yet existing dLLM methods only reduce inference steps: the distilled model is faster but no smaller. Distillation for *model compression*, routine for autoregressive (AR) LLMs, has not been studied for dLLMs. Because teacher and student differ in backbone, attention, and tokenizer, compressing a dLLM is a cross-architecture problem with three obstacles: timestep-dependent teacher reliability, context scarcity from heavy masking, and tokenizer/vocabulary mismatch. We present TIDE, a *regime-adaptive* framework whose components are matched to the teacher–student gap: TIDAL (timestep-aware scheduling) and COMPDEMO (context enrichment) for shared-tokenizer pairs, and Reverse CALM (noise-filtering chunk-level alignment) for cross-tokenizer pairs. TIDE distills 8B-dense and 16B-MoE teachers, one per run, into a 0.6B student that needs 11–22$\times$ less memory than the teachers. Naive distillation does not help: it causes *negative transfer* for the shared-tokenizer pair and gives no gain for the cross-tokenizer pair. TIDE improves on it in both pipelines ($+3.00$ and $+1.95$ on the eight-benchmark average) and ends above the non-distilled student ($+0.88$ and $+1.53$), most clearly on GSM8K ($+6.68$ in the cross-tokenizer pipeline) and less clearly on code. The reported scores of a same-size AR model remain higher (40.9 vs. 34.20 average). To our knowledge, this is the first evidence that a large dLLM can be compressed across heterogeneous architectures. Code is available at https://anonymous.4open.science/r/tide-5E36/.

1 INTRODUCTION

Autoregressive language models dominate natural language generation (Vaswani et al., 2017; Yang et al., 2024; 2025), but diffusion language models (dLLMs) are gaining ground as an alternative. Early work such as D3PM (Austin et al., 2021a) and MDLM (Sahoo et al., 2024) explored the basic training frameworks. More recent models such as LLaDA (Nie et al., 2025) and Dream (Ye et al., 2025) reach the scale of large language models, and LLaDA 2.0 (Bie et al., 2025) scales to 100B parameters. Unlike autoregressive models, dLLMs iteratively denoise a fully masked sequence, so they can decode in parallel and use bidirectional context. However, competitive dLLMs have 8B–16B parameters or more, with a matching computation cost (Nie et al., 2025; Liu et al., 2025; Bie et al., 2025; Ye et al., 2025), which makes them hard to deploy (Figure 1).

One simple way to obtain stronger small models is knowledge distillation (Hinton et al., 2015), which compresses a large model into a smaller one. For autoregressive LMs, methods such as MiniLLM (Gu et al., 2024), DistiLLM (Ko et al., 2024), GKD (Agarwal et al., 2024), and TAID (Shing et al., 2025) are well established; their designs include new Kullback–Leibler divergence losses and the use of teacher feedback. For dLLMs, existing methods (CDLM (Kim et al., 2025), DDD (Hayakawa et al., 2024), LSD (Fu et al., 2025), and SDTT (Deschenaux & Gulcehre, 2024)) instead distill a model into a faster version of itself: they reduce the number of denoising steps and keep the architecture and parameter count unchanged (Appendix Table 2). To our knowledge, distillation as *model compression*, which transfers a large teacher into a much smaller student as is common for autoregressive LLMs, has not been studied for dLLMs.

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

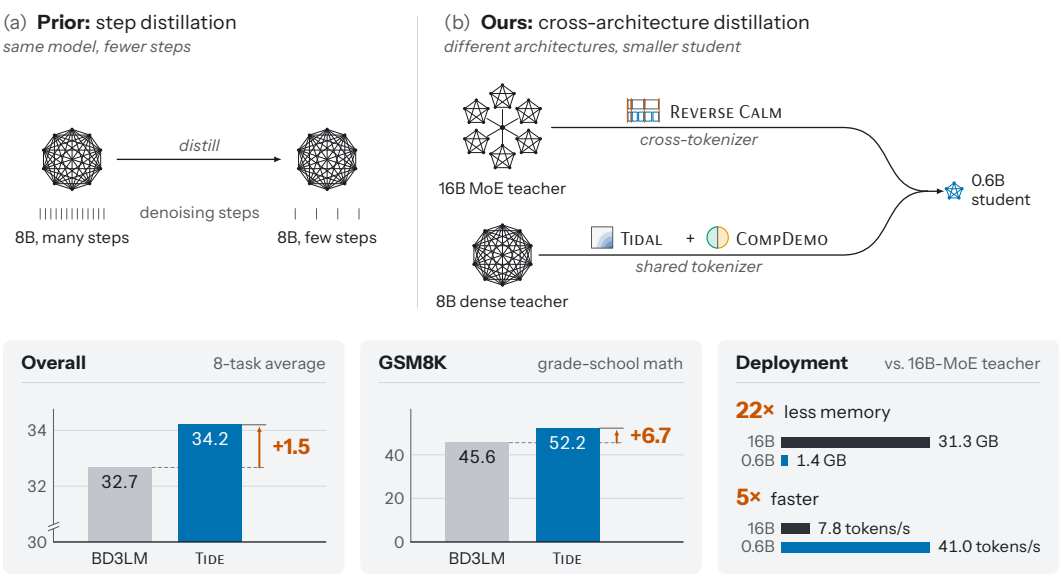

Figure 1: Cross-architecture distillation for dLLMs. Step distillation keeps the model size fixed; TIDE compresses heterogeneous 16B-MoE and 8B-dense teachers into a 0.6B student, one teacher per run. The bars compare the best (cross-tokenizer) student with the non-distilled BD3LM: $+1.5$ on the eight-task average (the axis starts at 30) and $+6.7$ on GSM8K. "$22\times$ less memory" and "$5\times$ faster" are relative to the 16B-MoE teacher; against the 8B-dense teacher the student needs $11\times$ less memory and is only $\sim 1.09\times$ faster, and a same-size AR model is faster still (Appendix Table 7).

Compressing a dLLM is a cross-architecture problem, because a capable teacher dLLM and a deployable small student usually differ in size, backbone, attention mechanism, and tokenizer. This raises three challenges. First, *timestep-dependent teacher reliability*: timesteps are sampled at random during training, and the teacher's reliability varies strongly across them. Second, *context scarcity from heavy masking*: at high noise levels most tokens are masked, so the teacher's raw predictions are too uninformative to transfer contextual representations. Third, *tokenizer/vocabulary mismatch*: when the vocabularies differ, standard token-level likelihood objectives are undefined.

The three challenges concern different parts of distillation: when the student should trust the teacher, how much context the teacher sees, and how the outputs of the two models are compared. TIDE (Figure 2) therefore works at three levels of the training pipeline and contributes one component at each:

- **Scheduling level (TIDAL).** The teacher is reliable when little is masked and unreliable when most of the sequence is masked. TIDAL scales the distillation weight along two axes (the diffusion timestep and the training progress) so that the student follows the teacher mainly when the teacher's predictions can be trusted.
- **Contextual level (COMPDEMO).** Under heavy masking the teacher has little context to condition on. COMPDEMO splits the masked positions into two halves and runs the teacher twice, each time revealing one half as context, so every masked position is predicted with more context.
- **Output level (Reverse CALM).** When teacher and student use different tokenizers, token-level losses are undefined. Reverse CALM compares probabilities over shared text spans (chunks), and reversing the usual loss direction filters out chunks the teacher is unsure about.

TIDE is *regime-adaptive*: a shared-tokenizer pair uses TIDAL and COMPDEMO, and a cross-tokenizer pair uses Reverse CALM, which we apply without the TIDAL curriculum (section 2.3).

We evaluate TIDE on two pipelines that distill into the same 0.6B block-diffusion student (BD3LM (Arriola et al., 2025)), which needs 11–22$\times$ less memory than its teachers: cross-tokenizer distillation from a 16B-MoE teacher (LLaDA2 (Bie et al., 2025)) and shared-tokenizer distillation from an 8B-dense teacher (WeDLM (Liu et al., 2025)). Naive distillation leaves the student at or below its non-distilled version, clearly below in the shared-tokenizer pipeline. The matched TIDE configu-

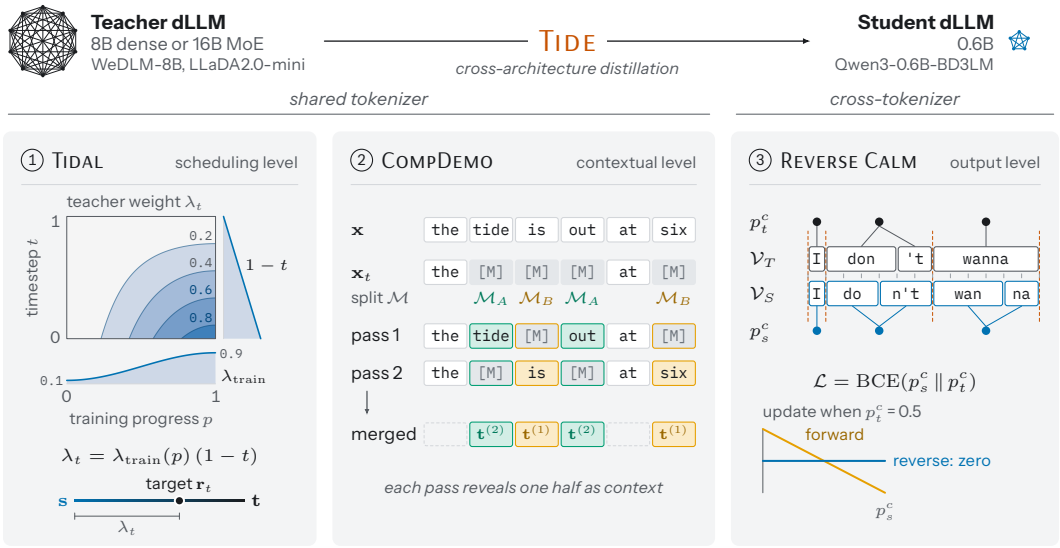

Figure 2: Overview of TIDE. Knowledge is transferred from a large teacher to a 0.6B student through three *regime-adaptive* components, applied according to the teacher–student gap: (1) TIDAL for dual-axis interpolation, (2) COMPDEMO for complementary teacher demonstration, and (3) Reverse CALM for cross-tokenizer alignment.

ration improves over its naive baseline (+3.00 shared-tokenizer, +1.95 cross-tokenizer) and over the non-distilled student on the eight-benchmark average (+0.88 and +1.53; best 34.20 vs. 32.67). The same pattern in two pipelines that differ in teacher, tokenizer, and baseline objective makes a single-run artifact less likely. The gains concentrate on grade-school math word problems (GSM8K, where only the cross-tokenizer gain carries over to its variants) and code, while knowledge and commonsense tasks change little; most of the absolute code score is inherited from the BD3LM initialization (section 3.2). The shared-tokenizer regime prefers TIDAL+COMPDEMO by 2.76, while the two cross-tokenizer configurations differ by only 0.37. A same-size AR model (Qwen3-0.6B) remains faster in our measurements and more accurate by its reported scores (40.9 vs. 34.20 average).

Our contributions are:

- We present the first study of distillation-based model compression for dLLMs and identify three obstacles that make it a cross-architecture problem: timestep-dependent teacher reliability, context scarcity from heavy masking, and tokenizer/vocabulary mismatch.
- We propose TIDE, built on *regime-adaptive matching*: TIDAL (scheduling) and COMPDEMO (context) for shared tokenizers, and Reverse CALM (alignment) for different ones.
- On eight benchmarks and two teacher–student pipelines, we give the first evidence that cross-architecture dLLM distillation is feasible but not automatic: naive objectives do not help, while TIDE improves on them and on the non-distilled student. We also compare the student with its teachers, analyze why the gains depend on the task, and give a gradient analysis that motivates Reverse CALM.

2 METHOD

TIDE (figure 2) distills a large teacher dLLM f_T (parameters θ_T) into a smaller student dLLM f_S (parameters θ_S), where teacher and student may differ in architecture, attention mechanism, and tokenizer vocabulary. Let $\mathbf{x} = (x_1, \ldots, x_L)$ denote a clean token sequence and $\mathbf{x}_t$ its noised version at a diffusion timestep $t \in [\epsilon, 1)$, where the positions in the mask set $\mathcal{M}$ are replaced with a special `[MASK]` token. The student predicts the clean tokens at masked positions from both the ground-truth labels and the teacher's predicted categorical token distribution.

A regime-adaptive design. TIDE has one component per obstacle and applies the subset that fits the teacher–student pair. When teacher and student share a tokenizer, their token distributions are

comparable, so we schedule the teacher signal over the diffusion timestep (TIDAL) and give the teacher more context (COMPDEMO). When the tokenizers differ, token-level objectives are undefined, so we align chunk-level probabilities with an objective that filters alignment noise (Reverse CALM). TIDAL is used only with forward losses, whose target it interpolates (Appendix D).

2.1 DUAL-AXIS LAMBDA MODULATION (TIDAL)

Unlike in autoregressive (AR) distillation, where the teacher always sees the full left context, the teacher's reliability in dLLM distillation varies with the diffusion timestep. When little is masked ($t \approx 0$), the teacher sees almost the entire sequence and predicts reliably; when most of it is masked ($t \approx 1$), the teacher has little context and its predictions become unreliable. The student's capacity to absorb the teacher also changes over training. These two effects motivate a *dual-axis* schedule that modulates distillation strength along both the diffusion timestep and the training progress.

Axis 1: Diffusion timestep. The coefficient λ_t is the weight the target places on the teacher; we shrink it as the timestep t grows and scale it by a training-progress base $\lambda_{\text{train}} \in [\lambda_{\text{init}}, \lambda_{\max}]$ (Axis 2, Eq. 2), so the student trusts the teacher mainly when the teacher is reliable:

$$\lambda_t = \lambda_{\text{train}} \times (1 - t), \tag{1}$$

At high noise levels ($t \approx 1$), $\lambda_t \approx 0$ and the student dominates the target, so unreliable teacher predictions are not distilled; at low noise ($t \approx 0$), $\lambda_t \approx \lambda_{\text{train}}$ and the student relies on the teacher.

Axis 2: Training progress. Because the student is weak early on, we raise the teacher's weight gradually; the base coefficient λ_{train} follows a cosine schedule over normalized training progress $p \in [0, 1]$:

$$\lambda_{\text{train}} = \lambda_{\text{init}} + (\lambda_{\max} - \lambda_{\text{init}}) \times \frac{1}{2}\left(1 - \cos(\pi \cdot p)\right), \tag{2}$$

where λ_{init} and $\lambda_{\max}$ are hyperparameters (defaults 0.1 and 0.9). Early in training, $\lambda_{\text{train}} \approx \lambda_{\text{init}}$ keeps the target student-dominated to prevent representation collapse; later, $\lambda_{\text{train}} \to \lambda_{\max}$ shifts supervision toward the teacher.

TAID (Shing et al., 2025) proposes a similar interpolation for AR models, but only along the training-progress axis; TIDAL adds the diffusion-timestep axis, which is specific to dLLMs (section 3.3 compares the two schedules).

Interpolated target and loss. The target blends the student's own prediction with the teacher's, so the student imitates itself early and leans on the teacher as λ_t grows. Given the student logits $\mathbf{s}$ and teacher logits $\mathbf{t}$ at the masked positions, we form the interpolated target:

$$\mathbf{r}_t = \text{softmax}\left(\frac{(1 - \lambda_t) \cdot \mathbf{s} + \lambda_t \cdot \mathbf{t}}{T}\right), \tag{3}$$

where T is the temperature and $\mathbf{r}_t$ is detached so no gradient flows through the mixed target. The loss then pulls the student's tempered distribution toward this target:

$$\mathcal{L}_{\text{TIDAL}} = D_{\text{KL}}\left(\mathbf{r}_t \,\middle\|\, \text{softmax}\left(\frac{\mathbf{s}}{T}\right)\right) \times T^2. \tag{4}$$

The factor T^2 keeps the distillation gradient's magnitude comparable across temperatures, offsetting the $\approx 1/T^2$ scaling that temperature softening introduces (Hinton et al., 2015). This loss is computed only at masked positions for memory efficiency. The main runs use a midrange timestep weighting $w(t) = \exp\left(-\frac{(t-0.5)^2}{2\sigma^2}\right)$ with $\sigma = 0.15$ to emphasize intermediate timesteps.

2.2 COMPLEMENTARY DEMONSTRATION

In standard dLLM distillation, the teacher receives the same masked input $\mathbf{x}_t$ as the student. At high masking ratios, the limited context makes the teacher's predictions noisy, which degrades the distillation signal. We propose *Complementary Demonstration* (COMPDEMO), which uses the mask structure to give the teacher more context. The mechanism relies on the mask structure of dLLMs, which AR distillation lacks.

Motivation. Shenfeld et al. (2026) show that demonstration-conditioned teachers give better training signals in the AR setting. Discrete diffusion has a direct analog: revealing a subset of masked tokens

moves the teacher to a lower effective timestep and improves its predictions. We partition the masked positions into two complementary subsets, each serving as additional context for one of two teacher forward passes, so every masked position is predicted with enriched context.

Mask splitting. Given the set of masked positions $\mathcal{M}$, we randomly partition this set into two complementary subsets, $\mathcal{M}_A$ and $\mathcal{M}_B$, such that:

$$\mathcal{M}_A \cup \mathcal{M}_B = \mathcal{M}, \quad \mathcal{M}_A \cap \mathcal{M}_B = \emptyset, \quad |\mathcal{M}_A|/|\mathcal{M}| \approx \rho, \tag{5}$$

where $\rho = 0.5$ is the demonstration ratio.

Two-pass teacher inference. We perform two forward passes through the frozen teacher model:

$$\begin{aligned} \text{Pass 1: } \mathbf{t}^{(1)} &= f_T(\text{reveal } \mathcal{M}_A, \text{mask } \mathcal{M}_B), \\ \text{Pass 2: } \mathbf{t}^{(2)} &= f_T(\text{reveal } \mathcal{M}_B, \text{mask } \mathcal{M}_A), \end{aligned} \tag{6}$$

In Pass 1, positions in $\mathcal{M}_A$ retain clean tokens as demonstration context while $\mathcal{M}_B$ stays masked, yielding teacher logits at $\mathcal{M}_B$; Pass 2 is symmetric. The merged logits are $\mathbf{t}_{\text{final}}[\mathcal{M}_B] \leftarrow \mathbf{t}^{(1)}[\mathcal{M}_B]$ and $\mathbf{t}_{\text{final}}[\mathcal{M}_A] \leftarrow \mathbf{t}^{(2)}[\mathcal{M}_A]$. CompDemo doubles the teacher's forward passes, but since the teacher is frozen (no gradient computation), overall training time increases by approximately 50%.

2.3 Distillation Objectives

TIDE uses one of two distillation pipelines, depending on whether the teacher and student share a tokenizer. Each pipeline has an objective matched to its alignment granularity.

Shared-tokenizer objective (WeDLM → BD3LM). When teacher and student share a tokenizer family (Liu et al., 2025), their token-level distributions are directly comparable, so the student learns from the ground-truth tokens and, where the schedule trusts the teacher, from the teacher's token distribution through the TIDAL KL loss (section 2.1), with optional CompDemo (section 2.2):

$$\mathcal{L}_B = \mathcal{L}_{\text{CE}} + w_{\text{tidal}} \cdot \mathcal{L}_{\text{TIDAL}}. \tag{7}$$

Cross-tokenizer objective (LLaDA2 → BD3LM). When teacher and student use different tokenizers ($\mathcal{V}_T \neq \mathcal{V}_S$), token-level KL is undefined because the vocabularies do not align. We therefore introduce *Chunk-level Approximate Likelihood Matching* (CALM), which adapts ALM (Minixhofer et al., 2025) to the dLLM setting.

Chunk alignment. Using tokenkit (Minixhofer et al., 2024; 2025), we align the two token sequences at the byte level to identify *chunks*: the minimal text spans that contain one or more complete tokens from each vocabulary. For example, if the student splits the word `hello` into `hel` and `lo` while the teacher keeps it as the single token `hello`, both sides form one shared chunk for `hello`, and the per-token log-probabilities on each side are summed into that chunk's score. Let C denote the total number of aligned chunks. We construct binary alignment matrices $\boldsymbol{A}_S \in \{0,1\}^{L_S \times C}$ and $\boldsymbol{A}_T \in \{0,1\}^{L_T \times C}$, where $[\boldsymbol{A}_S]_{i,c} = 1$ if and only if student token i is assigned to chunk c. Because teacher and student use different chat templates with incompatible special tokens, we align only *content* tokens and exclude template markup, which would otherwise create spurious cross-tokenizer chunks.

Chunk-level log-probabilities. For each token x_i, we compute the log-probability as $\log P(x_i) = \text{logits}_{x_i} - \text{logsumexp}(\text{logits})$, which avoids materializing the full $[b, L, V]$ softmax matrix. The chunk-level log-probabilities are then a matrix product:

$$\text{LP}_S = \mathbf{lp}_S \cdot \boldsymbol{A}_S \in \mathbb{R}^{b \times C}, \qquad \text{LP}_T = \mathbf{lp}_T \cdot \boldsymbol{A}_T \in \mathbb{R}^{b \times C}, \tag{8}$$

where $\mathbf{lp}_S$ and $\mathbf{lp}_T$ are the per-token log-probability vectors. With temperature scaling, the chunk probabilities are $p_s^c = \exp(\text{LP}_S^c/T)$ and $p_t^c = \exp(\text{LP}_T^c/T)$.

Forward CALM. A natural baseline pushes the student's chunk probability toward the teacher's, treating each chunk as an independent Bernoulli outcome; this *forward*, or *mode-covering*, direction spreads student mass to cover every chunk the teacher marks present. The binary cross-entropy at masked positions is:

$$\mathcal{L}_{\text{Fwd-CALM}} = -\left[p_t^c \log p_s^c + (1 - p_t^c) \log(1 - p_s^c)\right]. \tag{9}$$

Because CALM operates on scalar chunk probabilities $p^c \in [0, 1]$, BCE is the appropriate loss, and the forward/reverse analysis differs from that of the KL divergence used in token-level distillation.

The forward objective can also follow the TIDAL curriculum, pulling the student toward the teacher gradually by interpolating the target in chunk-probability space:

$$p_{\text{mix}} = (1 - \lambda_t) \cdot p_s^c + \lambda_t \cdot p_t^c,$$
$$\mathcal{L}_{\text{CALM-TIDAL}} = -\big[p_{\text{mix}} \log p_s^c + (1 - p_{\text{mix}}) \log(1 - p_s^c)\big]. \tag{10}$$

Limitations of forward CALM. Forward CALM regresses p_s^c onto p_t^c; for a one-token chunk with $T = 1$ its update is exactly $p_t^c - p_s^c$ (Appendix D). Under imperfect cross-tokenizer alignment, the student therefore also fits the teacher's probabilities on misaligned chunks, noise included.

Reverse CALM. We instead reverse the direction of the BCE loss and score the fixed teacher probability under the student:

$$\mathcal{L}_{\text{Rev-CALM}} = -\big[p_s^c \log p_t^c + (1 - p_s^c) \log(1 - p_t^c)\big]. \tag{11}$$

Because $\mathcal{L}_{\text{Rev-CALM}}$ is linear in p_s^c, its gradient is $\partial p_s^c / \partial \theta$ times the coefficient $\log \frac{p_t^c}{1-p_t^c}$, which depends only on the fixed teacher (Figure 6, Appendix D). This gives dual-end noise filtering that forward CALM lacks: chunks the teacher is unsure about ($p_t^c \approx 0.5$), as poorly aligned chunks tend to be, zero the coefficient, and a low p_s^c suppresses the update through a small $\partial p_s^c / \partial \theta$. The resulting signal is *mode-seeking*: it drives the student toward the teacher's dominant outcome for each chunk instead of matching the teacher's probability. The loss is related to a Bernoulli KL but differs from it by the student's entropy (Appendix D). We do not combine TIDAL with Reverse CALM; Appendix D explains why we expect TIDAL to weaken its gradient.

Training objectives. As in the shared-tokenizer case, the student learns from the ground-truth tokens and a chunk-level distillation term:

$$\mathcal{L}_A = \mathcal{L}_{\text{CE}} + w_{\text{calm}} \cdot \mathcal{L}_{\text{dist}}, \tag{12}$$

where $\mathcal{L}_{\text{dist}} \in \{\mathcal{L}_{\text{CALM-TIDAL}},\ \mathcal{L}_{\text{Rev-CALM}}\}$. Both losses are computed at masked positions. When COMPDEMO is enabled, teacher logits are replaced by the merged two-pass logits (section 2.2).

3 EXPERIMENTS

3.1 EXPERIMENTAL SETUP

Models. The student model is Qwen3-0.6B-BD3LM, a 0.6B-parameter block diffusion language model initialized from Qwen3-0.6B-Base (Yang et al., 2025) and obtained from Zhou et al. (2026). Following the BD3LM (Arriola et al., 2025) framework, the model takes the concatenation of $[\mathbf{x}_t, \mathbf{x}_0]$ with a specialized attention mask during training and uses block diffusion with bidirectional attention for inference. We distill from two heterogeneous teachers: **(A)** LLaDA2.0-mini (Bie et al., 2025), an MoE dLLM with an independent tokenizer derived from the Ling series (Team et al., 2025); and **(B)** WeDLM-8B-Instruct (Liu et al., 2025), an 8B dense causal dLLM initialized from Qwen3-8B-Base (Yang et al., 2025).

Training. All main experiments use a learning rate of 5e-5, 10 training epochs, and bfloat16 precision. Following the training recipe of Zhou et al. (2026), we combine four SFT datasets: Tulu-3 SFT Mixture (Lambert et al., 2025), SmolTalk (Allal et al., 2025), and OpenCoder OPC-SFT Stage 1 and Stage 2 (Huang et al., 2025). The student uses a sequence length of 512 tokens and a block size of 32. Each 10-epoch run takes 32–67 hours on 32 H100-80GB GPUs, and Appendix B lists the other hyperparameters.

Evaluation. We evaluate on eight benchmarks covering reasoning (GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021b), BBH (Suzgun et al., 2022)), knowledge (MMLU-Pro (Wang et al., 2024), MMLU (Hendrycks et al., 2021a)), commonsense (HellaSwag (Zellers et al., 2019)), and code generation (HumanEval (Chen et al., 2021), MBPP (Austin et al., 2021b)). Inference and evaluation hyperparameters follow Zhou et al. (2026); Appendix B gives the task-specific configurations.

Baselines. We compare against two references from Zhou et al. (2026). The primary one is the undistilled BD3LM (Arriola et al., 2025), the released checkpoint that initializes our students: the

Table 1: Main results across eight benchmarks. All distillation methods include a cross-entropy loss term. AR: reported scores collected by Zhou et al. (2026); all dLLM columns come from our harness, and Appendix C.4 gives paired significance tests. **Bold**: best among dLLM models; underline: second best.

	Qwen3-0.6B		*Shared-Tokenizer*			*Cross-Tokenizer*		
Benchmark	AR	BD3LM	KL	TIDE-Rev	TIDE-Interp	CALM	TIDE-Interp	TIDE-Rev
GSM8K	59.6	45.56	43.97	45.03	48.98	48.60	49.89	**52.24**
MATH	32.4	13.08	9.40	9.76	11.16	13.14	12.98	**13.20**
BBH	41.5	26.32	25.79	26.00	26.79	24.21	26.85	**27.37**
MMLU-Pro	24.7	13.80	13.19	12.88	14.48	13.47	14.02	**14.52**
HellaSwag	47.4	39.28	39.78	39.50	**40.50**	40.42	39.57	39.88
MMLU	52.8	39.15	39.57	39.09	**39.92**	39.42	39.54	39.59
HumanEval	32.3	46.34	41.46	42.68	48.78	43.90	**49.39**	48.17
MBPP	36.6	37.80	31.20	31.40	37.80	34.80	38.40	**38.60**
Avg	40.9	32.67	30.55	30.79	33.55	32.25	33.83	**34.20**

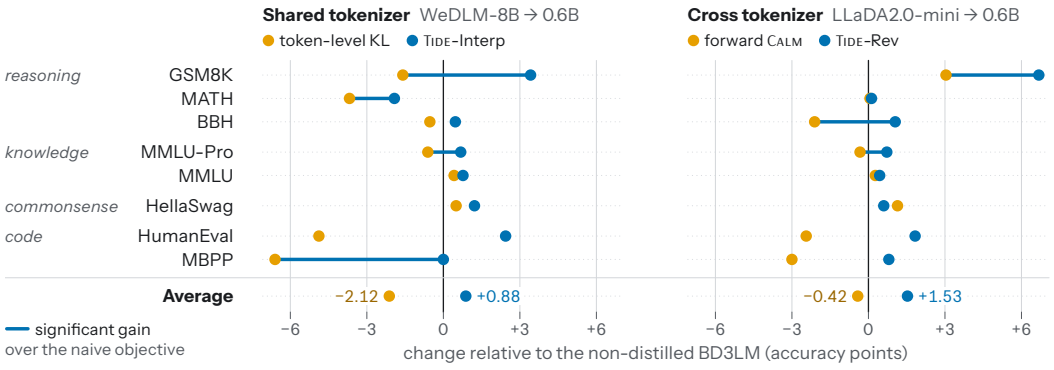

Figure 3: Per-benchmark change relative to the non-distilled BD3LM, computed from Table 1. Orange: the naive objective (token-level KL, forward CALM); blue: the matched TIDE configuration (TIDE-Interp, TIDE-Rev). A bar joins the two where the gain over the naive objective is significant after Bonferroni correction (Appendix C.4); the averages are not tested.

same 0.6B model before distillation, fine-tuned on the same data with the cross-entropy loss only and evaluated with the same `lm-evaluation-harness` fork and decoding as our students (Appendix B). Our students train for 10 further epochs from it, so only the comparison with naive distillation, which receives the same training, holds training length fixed. We also list the AR model Qwen3-0.6B-Base, which has the same parameter budget and tokenizer but decodes autoregressively. Its scores are the reported numbers collected by Zhou et al. (2026), not outputs of our harness, so it is only an indicative cross-paradigm reference, and we do not claim to surpass it at this scale. Appendix C.2 evaluates both teachers under our protocol.

3.2 MAIN RESULTS

Table 1 reports two pipelines with the same Qwen3-0.6B-BD3LM student. The *shared-tokenizer* pipeline distills WeDLM-8B-Instruct, with token-level KL as the naive baseline; the *cross-tokenizer* pipeline distills LLaDA2.0-mini, with forward CALM as the naive baseline. TIDE-Interp applies TIDAL and COMPDEMO (progressive scheduling and enriched teacher logits), and TIDE-Rev uses the mode-seeking Reverse CALM (Reverse KL when the tokenizers match). We also run each configuration in its non-native pipeline (Appendix Table 4).

Naive distillation does not help, but TIDE does. Both naive objectives end at or below the non-distilled BD3LM (32.67). Token-level KL drops to 30.55 (-2.12), a clear negative transfer in the shared-tokenizer pipeline, and forward CALM, which adapts the cross-tokenizer ALM ob-

jective (Minixhofer et al., 2025) to dLLMs, reaches 32.25 (-0.42). Adding the matched TIDE components lifts both pipelines above their naive baselines (shared $30.55 \rightarrow 33.55$, $+3.00$; cross $32.25 \rightarrow 34.20$, $+1.95$) and above the non-distilled reference (33.55 and 34.20 vs. 32.67, i.e. $+0.88$ and $+1.53$). Paired McNemar tests on per-item outputs (Appendix C.4) find the gain over the naive baseline significant after Bonferroni correction on four of eight tasks in the shared-tokenizer pipeline (GSM8K, MATH, MMLU-Pro, MBPP) and on three in the cross-tokenizer pipeline (GSM8K, BBH, MMLU-Pro), with no significant regression. On GSM8K, the TIDE student alone solves 189 items versus 131 for token-level KL, and 150 versus 101 against forward CALM. Since the naive losses alone do not help, we attribute the gain to the TIDE components. The improvement holds for two architecturally distinct teacher–student pairs and two different naive objectives, which argues against a lucky run or an under-tuned baseline, although each configuration was trained only once (section 4).

Choosing a configuration per regime. In the shared-tokenizer pipeline, where token-level alignment is exact, TIDE-Interp outperforms TIDE-Rev by 2.76 on average. In the cross-tokenizer pipeline, both TIDE configurations beat forward CALM (32.25) by a wide margin (33.83 and 34.20) and differ by only 0.37; we adopt Reverse CALM for its dual-end filtering of alignment noise (section 2.3).

Where the gains come from. For the best cross-tokenizer model, GSM8K ($+6.68$) and code move most (Figure 3) and BBH gains 1.05, while knowledge and commonsense tasks (MMLU, HellaSwag) change by less than a point and MATH is flat. The cross-tokenizer GSM8K gain is significant ($p = 2 \times 10^{-6}$) and holds, with a smaller margin, on GSM8K-Platinum (Vendrow et al., 2025) ($+2.15$) and GSM-Plus (Li et al., 2024) ($+1.41$), whereas the shared-tokenizer student falls below the non-distilled student on both (-3.97 and -1.92; Appendix C.3); we therefore claim the GSM8K gain only for the cross-tokenizer pipeline. On code, the matched TIDE students (HumanEval 48.78 and 48.17) score above the reported AR number (32.3), but most of this comes from the BD3LM initialization (46.34): distillation adds 1.8–2.4 points on HumanEval (3–4 of 164 problems) and at most 0.8 on MBPP, too few to be individually significant. On MATH, the shared-tokenizer student drops by 1.92 ($p = 6.8 \times 10^{-3}$, uncorrected); since the AR model it was converted from reports 32.4, the diffusion conversion and initialization, more than parameter count, appear to limit MATH.

Why the gains depend on the task. We see two likely reasons. First, the bidirectional context of dLLMs may suit the structured outputs of code and multi-step arithmetic, where the gains are largest (Appendix E); distillation also nearly halves the teacher–student KL on GSM8K (Figure 7). Second, broad world knowledge is likely bounded by the student's capacity: under our protocol, both teachers score 76–83 on HellaSwag against about 40 for the student with or without distillation, whereas on GSM8K the cross-tokenizer student closes about 59% of the gap to its teacher (Appendix C.2).

3.3 ABLATION STUDIES

Component-level ablation (suggestive). We ablate TIDE-Interp in the shared-tokenizer pipeline by removing one component at a time (3 epochs; Appendix Table 5; definitions in Appendix B). The full method has the best average (33.14), but the differences are small. Removing the timestep axis leaves a TAID-style schedule, so Full vs. w/o Tstep compares TIDAL with that schedule: the timestep axis adds 0.26 on average and 3.05 on HumanEval, in line with TIDAL's premise that teacher reliability varies with the masking ratio. Adding the training-progress axis to the timestep-only schedule of prior work (Baseline) gains only 0.08 on average (0.83 on GSM8K), and removing COMPDEMO costs 0.17 (2.44 on HumanEval). Full is also slightly worse than Baseline on MBPP (37.00 vs. 38.60), MMLU-Pro (13.76 vs. 14.12), and MATH (11.68 vs. 11.74). The 0.08 gain is at the scale of evaluation-side variation (re-evaluating a fixed checkpoint moves its average by up to 0.07; Appendix C.1), the HumanEval effects amount to 5 and 4 of 164 problems, and without training seeds we cannot tell whether the 0.17–0.26 differences exceed run-to-run variation. Given also the reduced budget and the missing cross-tokenizer ablation, Table 5 is suggestive evidence for the dual-axis design, whose main support remains the premise above; the case for Reverse CALM rests on Table 1 and its noise filtering (section 2.3).

3.4 INFERENCE EFFICIENCY

We benchmark inference on one H100-80GB GPU in bfloat16, in a *controlled* setting that generates 256 tokens from a fixed prompt (Appendix Table 7) and during the benchmark runs (Appendix Table 8).

Memory is the main benefit. The 0.6B student needs only 1.4 GB of peak memory (Appendix Table 7), 22× less than LLaDA2 (31.3 GB) and 11× less than WeDLM (15.5 GB), which puts a capable dLLM within reach of commodity hardware. In the controlled single-stream benchmark it is also about 5.3× faster than the 16B LLaDA2 teacher (41.0 vs. 7.8 tokens/s), and about as fast as the non-distilled student (42.1 tokens/s). Against the 8B WeDLM teacher (37.7 tokens/s) the speedup is small, likely because single-stream decoding underutilizes the GPU; batched serving and long outputs may show the student's efficiency advantage more clearly. During the benchmark runs, the student stays at 40.5–41.7 tokens/s on every task, while LLaDA2 ranges from 5.6 to 10.3 (Appendix Table 8).

4 LIMITATIONS

Scale and model coverage. We evaluate a single 0.6B student initialized from one base checkpoint (Qwen3-0.6B converted to BD3LM), so we do not know how the benefits of TIDE change with student capacity (e.g., 1.3B or 3B) or initialization. To our knowledge, this is the only publicly released block-diffusion student at a deployable scale; a larger one would first need an AR-to-diffusion conversion, and each added scale repeats a training matrix dominated by the frozen teacher (42–90 H100 GPU-days per configuration; Appendix B). The student uses block diffusion with staircase attention; continuous-state or encoder-style dLLM students would need adapted loss formulations and are not studied.

Teachers and training regime. We use two teachers, one dense (WeDLM-8B) and one MoE (LLaDA2.0-mini, 16B); a third family such as Dream (Ye et al., 2025) or DiffuLLaMA (Gong et al., 2025) is untested, and we do not isolate how teacher scale, family, or pretraining data affect transfer. TIDE distills on a fixed offline corpus with a 512-token context; on-policy or interactive distillation and longer sequences are not studied. We evaluate the teachers under the student's protocol, which understates them where their long chain-of-thought outputs are truncated (Appendix C.2).

Statistical scope of the gains. The gains over the non-distilled student concentrate on GSM8K and code, while MMLU, MMLU-Pro, HellaSwag, and MATH move by less than a point for the best student. Greedy decoding makes our evaluation deterministic for a fixed harness configuration, and re-evaluation with different batching moves the eight-task averages by at most 0.07 (Appendix C.1), but we do not estimate variance across training seeds, which neither this re-evaluation nor our paired tests cover. The gain over the non-distilled student may also partly reflect the 10 further training epochs, which the comparison with naive distillation controls for. We therefore count sub-point changes as no change, limit our positive claims to GSM8K in the cross-tokenizer pipeline and to code, and report the eight-benchmark average only as a summary. We also do not isolate the extra compute of COMPDEMO (about 50% more training time) or report results for Reverse CALM with TIDAL (Appendix D).

Deployment positioning. At the 0.6B scale, a same-size AR model is faster than our distilled dLLM (51.3 vs. 41.0 tokens/s, single stream) and, by its reported scores, more accurate (40.9 vs. 34.20 average). The practical benefit is the 11–22× memory reduction relative to the teachers, so we see dLLM compression as suited to memory-constrained or batched settings, not to single-stream, latency-bound use; a batched-throughput comparison against AR is left to future work. Our accuracy evaluation also uses as many sampling steps as new tokens (Appendix Table 6), so it does not test decoding several tokens per step.

5 CONCLUSION

We present TIDE, to our knowledge the first framework for distillation-based compression of a dLLM into a much smaller student with a different architecture. Across two pipelines and eight benchmarks, TIDE consistently improves over naive distillation: where the naive objectives leave the student at or below the non-distilled reference, TIDE raises the eight-benchmark average over that reference by $+0.88$ and $+1.53$, with the gains concentrated on cross-tokenizer GSM8K and code, and the student needs 11–22× less memory than its teachers. These results show that dLLM compression is feasible with the open dLLMs available today. Larger students and a wider range of teachers remain to be studied.

AI USE STATEMENT

We used generative AI tools to aid and polish the writing of this paper. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work focuses on improving the efficiency of diffusion language models through knowledge distillation. All datasets (Tulu-3 SFT Mixture (Lambert et al., 2025), SmolTalk (Allal et al., 2025), and OpenCoder OPC-SFT (Huang et al., 2025)) and pretrained models (Qwen3 (Yang et al., 2025), WeDLM (Liu et al., 2025), and LLaDA2.0-mini (Bie et al., 2025)) used in this work are publicly released for research and are used in accordance with their respective licenses and intended use; our use is limited to non-commercial research and does not redistribute the underlying data. Because we collect or annotate no new data and use only these established, publicly released datasets, we rely on the data curation and safety filtering performed by their original creators, and we neither extract nor release any personally identifying information. Training is compute-intensive: the six distilled configurations used about 353 H100 GPU-days, mostly for teacher forward passes (Appendix B), whereas the resulting 0.6B student needs only 1.4 GB of memory at inference. Because distillation copies a teacher's behavior into a smaller, cheaper-to-deploy model, it can also propagate the teacher's biases and errors into more widely deployed students; we recommend applying to the student the same bias and safety evaluations one would apply to the teacher before any downstream use.

REPRODUCIBILITY STATEMENT

Code is available at the anonymous repository `https://anonymous.4open.science/r/tide-5E36/`. Section 3.1 specifies the student and teacher checkpoints and the training data, and Appendix B lists the main training hyperparameters of both pipelines (Table 3), the ablation configurations, the per-task evaluation configuration (Table 6), the answer-extraction and code-execution protocol, the random-seed and determinism settings of our `lm-evaluation-harness`-based evaluation, and the inference-efficiency protocol. All datasets and pretrained models we use are publicly available (see the ethics statement). The gradient analysis motivating Reverse CALM is derived in Appendix D.

REFERENCES

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

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Agustín Piqueres Lajarín, Vaibhav Srivastav, Joshua Lochner, Caleb Fahlgren, Xuan-Son Nguyen, Clémentine Fourrier, Ben Burtenshaw, Hugo Larcher, Haojun Zhao, Cyril Zakka, Mathieu Morlon, Colin Raffel, Leandro von Werra, and Thomas Wolf. Smollm2: When smol goes big – data-centric training of a small language model, 2025. URL https://arxiv.org/abs/2502.02737.

Marianne Arriola, Aaron Gokaslan, Justin T Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Block diffusion: Interpolating between autoregressive and diffusion language models. *arXiv preprint arXiv:2503.09573*, 2025.

Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. Structured denoising diffusion models in discrete state-spaces. *Advances in neural information processing systems*, 34:17981–17993, 2021a.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models, 2021b. URL https://arxiv.org/abs/2108.07732.

Tiwei Bie, Maosong Cao, Kun Chen, Lun Du, Mingliang Gong, Zhuochen Gong, Yanmei Gu, Jiaqi Hu, Zenan Huang, Zhenzhong Lan, Chengxi Li, Chongxuan Li, Jianguo Li, Zehuan Li, Huabin Liu, Lin Liu, Guoshan Lu, Xiaocheng Lu, Yuxin Ma, Jianfeng Tan, Lanning Wei, Ji-Rong Wen, Yipeng Xing, Xiaolu Zhang, Junbo Zhao, Da Zheng, Jun Zhou, Junlin Zhou, Zhanchao Zhou, Liwang Zhu, and Yihong Zhuang. Llada2.0: Scaling up diffusion language models to 100b, 2025. URL https://arxiv.org/abs/2512.15745.

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

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

Justin Deschenaux and Caglar Gulcehre. Beyond autoregression: Fast llms via self-distillation through time. *arXiv preprint arXiv:2410.21035*, 2024.

Feiyang Fu, Tongxian Guo, and Zhaoqiang Liu. Learnable sampler distillation for discrete diffusion models. *arXiv preprint arXiv:2509.19962*, 2025.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024. URL https://zenodo.org/records/12608602.

Shansan Gong, Shivam Agarwal, Yizhe Zhang, Jiacheng Ye, Lin Zheng, Mukai Li, Chenxin An, Peilin Zhao, Wei Bi, Jiawei Han, Hao Peng, and Lingpeng Kong. Scaling diffusion language models via adaptation from autoregressive models, 2025. URL https://arxiv.org/abs/2410.17891.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. MiniLLM: Knowledge distillation of large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=5h0qf7IBZZ.

Satoshi Hayakawa, Yuhta Takida, Masaaki Imaizumi, Hiromi Wakaki, and Yuki Mitsufuji. Distillation of discrete diffusion through dimensional correlations. *arXiv preprint arXiv:2410.08709*, 2024.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021a. URL https://openreview.net/forum?id=d7KBjmI3GmQ.

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

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

Siming Huang, Tianhao Cheng, J. K. Liu, Jiaran Hao, Liuyihan Song, Yang Xu, J. Yang, Jiaheng Liu, Chenchen Zhang, Linzheng Chai, Ruifeng Yuan, Zhaoxiang Zhang, Jie Fu, Qian Liu, Ge Zhang, Zili Wang, Yuan Qi, Yinghui Xu, and Wei Chu. Opencoder: The open cookbook for top-tier code large language models, 2025. URL https://arxiv.org/abs/2411.04905.

Minseo Kim, Chenfeng Xu, Coleman Hooper, Harman Singh, Ben Athiwaratkun, Ce Zhang, Kurt Keutzer, and Amir Gholami. Cdlm: Consistency diffusion language models for faster sampling. *arXiv preprint arXiv:2511.19269*, 2025.

Jongwoo Ko, Sungnyun Kim, Tianyi Chen, and Se-Young Yun. Distillm: Towards streamlined distillation for large language models, 2024. URL https://arxiv.org/abs/2402.03898.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tulu 3: Pushing frontiers in open language model post-training, 2025. URL https://arxiv.org/abs/2411.15124.

Qintong Li, Leyang Cui, Xueliang Zhao, Lingpeng Kong, and Wei Bi. GSM-Plus: A comprehensive benchmark for evaluating the robustness of LLMs as mathematical problem solvers. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2961–2984, 2024.

Aiwei Liu, Minghua He, Shaoxun Zeng, Sijun Zhang, Linhao Zhang, Chuhan Wu, Wei Jia, Yuan Liu, Xiao Zhou, and Jie Zhou. Wedlm: Reconciling diffusion language models with standard causal attention for fast inference. *arXiv preprint arXiv:2512.22737*, 2025.

Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. *arXiv preprint arXiv:2310.16834*, 2023.

Benjamin Minixhofer, Edoardo M Ponti, and Ivan Vulić. Zero-shot tokenizer transfer. *Advances in Neural Information Processing Systems*, 37:46791–46818, 2024.

Benjamin Minixhofer, Ivan Vulić, and Edoardo Maria Ponti. Universal cross-tokenizer distillation via approximate likelihood matching. *arXiv preprint arXiv:2503.20083*, 2025.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models. *arXiv preprint arXiv:2502.09992*, 2025.

Khouloud Saadi and Di Wang. What should feature distillation transfer in llms? a task-tangent geometry view, 2026. URL https://arxiv.org/abs/2507.10155.

Subham S Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. *Advances in Neural Information Processing Systems*, 37:130136–130184, 2024.

Idan Shenfeld, Mehul Damani, Jonas Hübotter, and Pulkit Agrawal. Self-distillation enables continual learning. *arXiv preprint arXiv:2601.19897*, 2026.

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

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed H. Chi, Denny Zhou, and Jason Wei. Challenging big-bench tasks and whether chain-of-thought can solve them, 2022. URL https://arxiv.org/abs/2210.09261.

Ling Team, Ang Li, Ben Liu, Binbin Hu, Bing Li, Bingwei Zeng, Borui Ye, Caizhi Tang, Changxin Tian, Chao Huang, Chao Zhang, Chen Qian, Chenchen Ju, Chenchen Li, Chengfu Tang, Chilin Fu, Chunshao Ren, Chunwei Wu, Cong Zhang, Cunyin Peng, Dafeng Xu, Daixin Wang, Dalong Zhang, Dingnan Jin, Dingyuan Zhu, Dongke Hu, Fangzheng Zhao, Feifan Wu, Feng Zhu, Gangshan Wang, Haitao Zhang, Hailin Zhao, Hanxiao Zhang, Hanzi Wang, Hao Qian, Haoyi Yu, Heng Zhang, Hongliang Zhang, Hongzhi Luan, Huirong Dong, Huizhong Li, Jia Li, Jia Liu, Jialong Zhu, Jian Sha, Jianping Wei, Jiaolong Yang, Jieyue Ma, Jiewei Wu, Jinjing Huang, Jingyun Tian, Jingyuan Zhang, Jinquan Sun, Juanhui Tu, Jun Liu, Jun Xu, Jun Zhou, Junjie Ou, Junpeng Fang, Kaihong

Zhang, Kaiqin Hu, Ke Shi, Kun Tang, Kunlong Chen, Lanyin Mei, Lei Liang, Lei Xu, Libo Zhang, Lin Ju, Lin Yuan, Ling Zhong, Lintao Ma, Lu Liu, Lu Yu, Lun Cai, Meiqi Zhu, Mengying Li, Min Chen, Minghao Xue, Minghong Cai, Mingming Yin, Peijie Jiang, Peilong Zhao, Pingping Liu, Qian Zhao, Qing Cui, Qingxiang Huang, Qingyuan Yang, Quankun Yu, Shaowei Wei, Shijie Lian, Shoujian Zheng, Shun Song, Shungen Zhang, Shuo Zhang, Siyuan Li, Song Liu, Ting Guo, Tong Zhao, Wanli Gu, Weichang Wu, Weiguang Han, Wenjing Fang, Wubin Wang, Xiang Shu, Xiao Shi, Xiaoshun Lan, Xiaolu Zhang, Xiaqing Sun, Xin Zhao, Xingyu Lu, Xiong Xu, Xudong Wang, Xudong Wang, Xuemin Yang, Yajie Yang, Yang Xiang, Yanzhe Li, Yi Zhang, Yilong Wang, Yingxue Li, Yongzhen Guo, Yuzhuo Fu, Yuanyuan Wang, Yue Yang, Yue Yu, Yufeng Deng, Yun Zhang, Yunfei Yu, Yuqi Zhang, Yuxiao He, Zengke Gui, Zhaoxin Huan, Zhaoyang Wang, Zhibo Zhu, Zhihao Wang, Zhiqiang Zhang, Zhoufei Wang, Zihang Zeng, Ziqi Liu, Zitao Xuan, and Zuoli Tang. Every activation boosted: Scaling general reasoner to 1 trillion open language foundation, 2025. URL https://arxiv.org/abs/2510.22115.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Joshua Vendrow, Edward Vendrow, Sara Beery, and Aleksander Madry. Do large language model benchmarks test reliability? *arXiv preprint arXiv:2502.03461*, 2025.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark, 2024. URL https://arxiv.org/abs/2406.01574.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

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

Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Dream 7b: Diffusion large language models. *arXiv preprint arXiv:2508.15487*, 2025.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp. 4791–4800, 2019.

Songming Zhang, Xue Zhang, Zengkui Sun, Yufeng Chen, and Jinan Xu. Dual-space knowledge distillation for large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 18164–18181, 2024.

Zhanhui Zhou, Lingjie Chen, Hanghang Tong, and Dawn Song. dllm: Simple diffusion language modeling, 2026. URL https://arxiv.org/abs/2602.22661.

Fengqi Zhu, Zebin You, Yipeng Xing, Zenan Huang, Lin Liu, Yihong Zhuang, Guoshan Lu, Kangyu Wang, Xudong Wang, Lanning Wei, Hongrui Guo, Jiaqi Hu, Wentao Ye, Tieyuan Chen, Chenchen Li, Chengfu Tang, Haibo Feng, Jun Hu, Jun Zhou, Xiaolu Zhang, Zhenzhong Lan, Junbo Zhao, Da Zheng, Chongxuan Li, Jianguo Li, and Ji-Rong Wen. Llada-moe: A sparse moe diffusion language model, 2025. URL https://arxiv.org/abs/2509.24389.

A Related Work

Diffusion language models. D3PM (Austin et al., 2021a) introduced discrete diffusion for text generation. MDLM (Sahoo et al., 2024) and SEDD (Lou et al., 2023) strengthened its theoretical basis with simplified masked diffusion and score estimation, respectively. Practical dLLMs built on this work differ in architecture: LLaDA (Nie et al., 2025) uses full bidirectional attention, BD3LM (Arriola et al., 2025) introduces block diffusion with staircase attention, Dream (Ye et al., 2025) extends masked diffusion with rectified estimation, and WeDLM (Liu et al., 2025) proposes a causal diffusion architecture that combines sliding-window and global attention. LLaDA-MoE (Zhu et al., 2025) is a sparse mixture-of-experts diffusion LM that activates only a small subset of experts per token, which scales capacity without a proportional increase in inference cost. DiffuLLaMA (Gong et al., 2025) converts pre-trained AR models into diffusion LMs. These models span encoder, decoder-block, and causal variants with dense or sparse mixture-of-experts backbones, which motivates a cross-architecture distillation framework.

Knowledge distillation of large language models. Knowledge distillation (Hinton et al., 2015) transfers knowledge from a large teacher to a smaller student through soft targets. For AR models, representative methods include reverse KL minimization (Gu et al., 2024), skewed KL divergence (Ko et al., 2024), on-policy distillation (Agarwal et al., 2024), dual-space transfer (Zhang et al., 2024), feature-level distillation (Saadi & Wang, 2026), and time-varying interpolation (Shing et al., 2025). These methods target AR models. TIDE builds on this line of work, particularly TAID's interpolation principle, and adapts it to dLLMs, where teacher reliability varies with the diffusion timestep and all masked tokens are predicted simultaneously.

Distillation for diffusion language models. Existing dLLM distillation methods (CDLM (Kim et al., 2025), DDD (Hayakawa et al., 2024), LSD (Fu et al., 2025), and SDTT (Deschenaux & Gulcehre, 2024)) focus only on *step compression*: the student keeps the teacher's architecture, tokenizer, and parameter count, and the goal is to reduce the number of denoising steps at inference. Because these methods hold the model size fixed, distillation as *model compression*, which transfers a large teacher into a much smaller student, has to our knowledge not been studied for dLLMs. Table 2 contrasts the two directions. A step-distillation method cannot serve as a baseline for model compression because it keeps the model size fixed. The two directions are complementary: a student compressed with TIDE could also be step-distilled. TIDE's setting is *cross-architecture*, because a capable teacher and a deployable student differ in architecture, attention mechanism, and possibly tokenizer. For the cross-tokenizer case, we build on ALM (Minixhofer et al., 2025) and ZeTT (Minixhofer et al., 2024) and adapt chunk-level approximate likelihood matching from the AR setting to dLLMs as CALM (section 2.3).

Table 2: Step distillation and model compression for dLLMs.

	Step distillation (CDLM, DDD, LSD, SDTT)	**Model compression** (TIDE)
What is reduced	denoising steps	parameters (8–16B $\rightarrow$ 0.6B)
Student vs. teacher	same architecture, tokenizer, size	different backbone and attention; tokenizer may differ
Compared against	the same model with more steps	naive distillation; the non-distilled student

B Training, Inference, and Evaluation Details

Training configurations.

Table 3 lists the main training hyperparameters for both pipelines. Training cost is dominated by the frozen teacher, which runs a forward pass at every step (two with COMPDEMO). One 10-epoch configuration takes 32–67 hours on 32 H100-80GB GPUs (42–90 H100 GPU-days), and the six distilled configurations in Table 1 used about 353 H100 GPU-days in total. Table 4 lists these configurations.

Ablation configurations. For the component-level ablations in Section 3.3 (Table 5), all configurations are trained for 3 epochs instead of the 10 used for the main-table models, so we read those

Table 3: Training hyperparameters.

Parameter	Cross-Tok.	Shared-Tok.
Teacher	LLaDA2.0-mini (16B MoE)	WeDLM-8B-Instruct (8B)
Student init	Qwen3-0.6B-BD3LM	Qwen3-0.6B-BD3LM
Methods	Reverse CALM	TIDAL + COMPDEMO
Learning rate	5e-5	5e-5
Epochs	10	10
Sequence length (student)	512	512
Teacher max length	1024	768
Block size	32	32
Precision	bfloat16	bfloat16
Teacher precision	bfloat16	bfloat16
Attention impl.	flex_attention	flex_attention
Strategy	DDP	DDP
Dataset	Tulu-3 SFT + SmolTalk + OpenCoder-SFT (Python)	
TIDAL $\lambda_{\text{init}} \to \lambda_{\max}$	-	$0.1 \to 0.9$
TIDAL schedule	-	cosine
TIDAL timestep weight	-	midrange
COMPDEMO demo_ratio	-	0.5
Temperature T	-	2.0

Table 4: The six distilled configurations of Table 1. Each also trains with the cross-entropy loss on the ground-truth tokens.

Pipeline (teacher)	Column in Table 1	Distillation term
Shared-tokenizer (WeDLM-8B)	KL (naive baseline)	token-level KL
	TIDE-Interp (matched)	TIDAL loss (Eq. 4) with COMPDEMO
	TIDE-Rev (non-native)	token-level reverse KL
Cross-tokenizer (LLaDA2.0-mini)	CALM (naive baseline)	forward CALM (Eq. 9)
	TIDE-Rev (matched)	Reverse CALM (Eq. 11)
	TIDE-Interp (non-native)	CALM with TIDAL (Eq. 10) and COMPDEMO

results as suggestive rather than as precise per-component values. The three ablation conditions are defined as follows:

- **w/o Tstep (removing the timestep axis):** $\lambda_t = \lambda_{\text{train}}$, i.e. lambda depends only on the training progress, a TAID-style schedule.
- **Baseline (timestep-only schedule):** $\lambda_t = \text{const} \times (1 - t)$, the schedule from prior dLLM work, in place of the dual-axis schedule.
- **w/o COMPDEMO:** a single teacher forward pass with the dual-axis TIDAL, without complementary demonstration.

Evaluation protocol. All evaluations use diffusion sampling with a block size of 32 and a classifier-free guidance scale of 0.0. The number of sampling steps depends on the task and ranges from 3 to 256; Table 6 gives the configuration for each task.

Code execution and answer extraction. HumanEval and MBPP are scored as pass@1 by executing each generated completion against the official unit tests in the standard `lm-evaluation-harness` code-execution sandbox, with the harness's default stop-sequence truncation and code extraction applied to every model and no model-specific filtering. The multiple-choice tasks (MMLU, HellaSwag) use `max_new_tokens` $= 3$ because their targets are single tokens, and the open-ended tasks use 256. All task-level settings (prompts, few-shot count, `max_new_tokens`, answer extraction, and code execution) are identical for the distilled students and the non-distilled BD3LM, and the teachers in Appendix C.2 use the same task-level settings (Table 9 describes their decoding). The AR reference in Table 1 is not part of this protocol: its scores are the reported numbers collected by Zhou et al. (2026).

Random seeds and determinism. We evaluate with the standard `lm-evaluation-harness` (Gao et al., 2024) flow, wrapping the BD3LM diffusion sampler as a

Table 5: Component-level ablation on the shared-tokenizer pipeline (WeDLM → Qwen3-BD3LM, 3 epochs); each column removes one component from Full. Full vs. w/o Tstep (TAID-style) measures the timestep axis that TIDAL adds to TAID; Full vs. Baseline (timestep only) measures the training-progress axis. **Bold**: best per row.

Benchmark	Baseline (timestep only)	w/o Tstep (TAID-style)	w/o COMPDEMO	Full (TIDAL + COMPDEMO)
GSM8K	48.07	48.82	**48.90**	**48.90**
MATH	11.74	**11.96**	11.84	11.68
BBH	26.37	26.51	**26.77**	26.66
MMLU-Pro	14.12	**14.42**	13.76	13.76
HellaSwag	40.03	**40.35**	40.16	40.27
MMLU	39.81	39.84	39.58	**39.92**
HumanEval	45.73	43.90	44.51	**46.95**
MBPP	**38.60**	37.20	38.20	37.00
Avg	33.06	32.88	32.97	**33.14**

Table 6: Per-task evaluation configuration.

Task	max_new_tokens	steps	few-shot	batch_size
MMLU	3	3	0	128
MMLU-Pro	256	256	0	64
HellaSwag	3	3	0	128
GSM8K	256	256	0	128
BBH	256	256	3	32
MATH	256	256	0	32
HumanEval	256	256	0	128
MBPP	256	256	0	32

harness model. Following the protocol of Zhou et al. (2026), we do not pass an explicit evaluation seed and therefore inherit the harness defaults (`random` $= 0$, `numpy` $= 1234$, `torch` $= 1234$, few-shot $= 1234$). Our default decoding is deterministic and does not depend on the `torch` seed: at temperature 0 the sampler reduces to an $\arg\max$ over logits (no Gumbel noise is added), and `low_confidence` remasking selects tokens by softmax confidence, not by random draws. Few-shot selection is deterministic as well. Only BBH uses in-context examples (3-shot; the other tasks are zero-shot), and the harness's `first_n` sampler picks a fixed exemplar set instead of drawing it with the seed, so no part of our evaluation depends on the random seed. The harness fork, sampling configuration, and seeds are the same as those that produced the baseline numbers of Zhou et al. (2026), so the distilled student and its non-distilled reference are evaluated under identical conditions, and we report single-run results. Batch composition can still change individual greedy outputs slightly; Appendix C.1 measures this effect.

Inference efficiency protocol. All inference efficiency measurements in Section 3.4 use a single NVIDIA H100-80GB GPU with bfloat16 precision. In the controlled setting, we generate 256 tokens and report the best of five independent runs. In the evaluation setting, we run 50 randomly sampled examples from each benchmark and report the average.

C ADDITIONAL EVALUATION

C.1 EVALUATION STABILITY

Greedy decoding is deterministic for a fixed harness configuration, but individual outputs can change with batch composition. To measure this, we re-evaluated all seven dLLM checkpoints of Table 1 on all eight tasks on a single GPU with per-rank batching, instead of the 8-way data-parallel runs behind Table 1. The eight-task averages reproduce to within 0.07 points for every configuration (e.g., non-distilled 32.66 vs. 32.67, naive KL 30.58 vs. 30.55, best cross-tokenizer 34.25 vs. 34.20). Over the 56 configuration–task pairs, single-task deviations are typically 0.0–0.6 points and at most 1.2. The paired tests (Appendix C.4) and the student columns of the teacher comparison (Appendix C.2)

Table 7: Inference efficiency (controlled setting): 256 tokens from a fixed prompt on one H100-80GB GPU in bfloat16. Mem: peak memory; Lat.: latency.

Model	**Params** (B)	**Mem** (GB)	**Lat.** (s)	**Tokens/s**
BD3LM (distilled)	0.60	1.4	6.25	41.0
BD3LM (no distill)	0.60	1.4	6.08	42.1
Qwen3 (AR)	0.60	1.2	4.99	51.3
WeDLM (teacher)	8.19	15.5	6.79	37.7
LLaDA2 (teacher)	16.26	31.3	32.55	7.8

Table 8: Per-benchmark inference speed (tokens/s) during evaluation runs. No Dist.: undistilled BD3LM; Dist.: distilled. The student's speed is nearly constant because of its fixed diffusion schedule.

	Teachers		*AR*	*Student*	
Benchmark	LLaDA2	WeDLM	Qwen3	No Dist.	Dist.
GSM8K	8.3	38.6	51.4	41.9	40.7
MATH	10.3	38.4	51.1	41.8	40.6
BBH	9.8	38.5	51.1	42.0	40.9
MMLU-Pro	5.6	38.3	51.0	42.2	40.5
HellaSwag	6.8	37.2	51.5	42.4	40.6
MMLU	6.2	37.1	51.4	41.9	41.1
HumanEval	9.8	36.8	51.1	42.5	41.7
MBPP	9.1	37.0	51.8	42.3	41.6
Avg	8.2	37.7	51.3	42.1	40.9

use this re-evaluation. This measures evaluation-side variation only and does not bound variation across training seeds.

C.2 Teacher–student gap

Table 9 evaluates both teachers on all eight benchmarks with the harness, prompts, and token budgets used for the students; its caption lists how the teacher runs differ from the student runs, and Figure 4 plots the same numbers. It shows the gap under the student's deployment protocol, which understates the teachers where their long chain-of-thought outputs are truncated or miss the answer-extraction patterns. On GSM8K, the cross-tokenizer student closes about 59% of the gap to LLaDA2 (45.94 $\rightarrow$ 51.71 vs. 55.80) with 3.7% of its parameters and 22$\times$ less memory; measured against the teachers' reported scores, the closed fraction is smaller. On knowledge and commonsense tasks (MMLU, HellaSwag) the gap stays wide with or without distillation, consistent with section 3.2.

C.3 GSM8K variants

To check whether the GSM8K gains hold beyond the original test set, we evaluate our checkpoints under the same protocol on GSM8K-Platinum (Vendrow et al., 2025), which corrects mislabeled answers and removes flawed questions, and on GSM-Plus (Li et al., 2024), which perturbs the test questions (Table 10 and Figure 5). The cross-tokenizer student keeps its lead over the non-distilled student on both, with a smaller margin than on GSM8K. The shared-tokenizer student falls below the non-distilled student on both, consistent with its MATH drop (Appendix C.4), so we do not claim a GSM8K gain for the shared-tokenizer pipeline.

C.4 Paired significance tests

For each task we compare two models on the same items with an exact McNemar test, i.e., a binomial test on the items that exactly one of the two models solves, and apply a Bonferroni correction over the eight tasks ($\alpha = 0.05/8 = 0.00625$). The tests use the per-item outputs of the re-evaluation in Appendix C.1; "gained/lost" counts the items that only the first or only the second model solves.

Table 9: Teachers and students under the student's evaluation protocol. The student columns (the non-distilled BD3LM and the best cross-tokenizer student, TIDE-Rev) are from the single-GPU re-evaluation of Appendix C.1; their eight-task averages (32.66 and 34.25) match Table 1 to within 0.07. The teacher columns use the same software environment, harness, prompts, and token budgets as the students (Table 6; BBH 3-shot, all others 0-shot) but were not part of that run: WeDLM-8B was evaluated on one GPU in the same session with standard Hugging Face `generate` (autoregressive) decoding, and LLaDA2.0-mini afterwards with 8-way data parallelism (5-way for BBH and MMLU-Pro) at batch size 1. †The protocol understates the teacher because long chain-of-thought outputs are truncated or miss the extraction patterns: the WeDLM report gives 92.27 on GSM8K (3-shot) and 64.80 on MATH (4-shot) (Liu et al., 2025), and LLaDA2's MMLU and MMLU-Pro scores are extraction and truncation effects. ‡LLaDA2 subsets (first-N examples): HellaSwag $n = 2{,}000$ of 10,042; BBH and MMLU-Pro 20 per subtask ($n = 540$ and 280); MATH with the algebra subset capped at 1,000 ($n = 4{,}813$ of 5,000). All other entries use the full test sets.

	Teachers		*Student (0.6B)*	
Benchmark	LLaDA2.0-mini (16B)	WeDLM-8B	BD3LM (no distill)	TIDE-Rev (cross)
GSM8K	55.80	61.71†	45.94	51.71
MATH	20.49†‡	19.16†	13.04	13.20
BBH	29.81†‡	39.52	26.32	27.17
MMLU-Pro	1.79†‡	15.10†	13.75	14.71
HellaSwag	76.45‡	83.05	39.38	39.93
MMLU	35.10†	57.99	39.14	39.50
HumanEval	85.98	81.10	45.73	48.78
MBPP	54.40	61.60	38.00	39.00

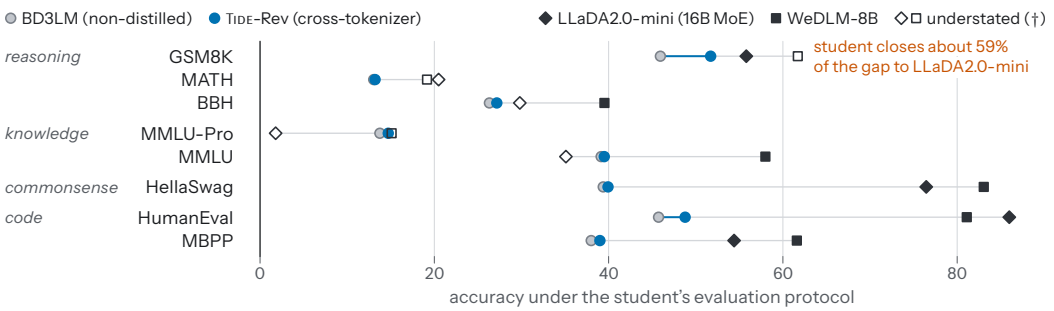

Figure 4: Teachers and students on each benchmark under the student's evaluation protocol (Table 9; student scores from the re-evaluation of Appendix C.1). Open teacher marks are the entries marked with † in Table 9, which the protocol understates. On GSM8K the distilled student closes about 59% of the gap to LLaDA2.0-mini; on HellaSwag both teachers score 76–83 against about 40 for the student, with or without distillation.

Table 10: Accuracy on two GSM8K variants under the protocol of Table 1.

Model	**GSM8K-Platinum**	**GSM-Plus**
BD3LM (no distill)	45.49	27.12
TIDE-Interp (shared-tokenizer)	41.52	25.20
TIDE-Rev (cross-tokenizer)	47.64	28.53

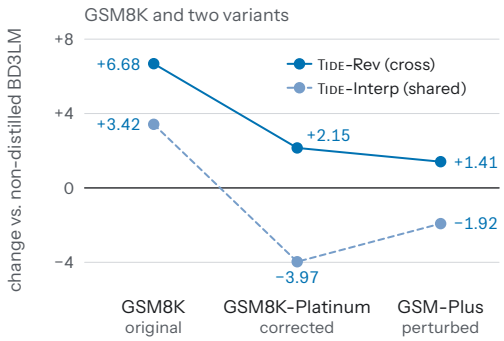

Figure 5: Change relative to the non-distilled BD3LM on GSM8K (Table 1) and on its two variants (Table 10). The cross-tokenizer student stays ahead on both variants with a smaller margin; the shared-tokenizer student falls below the non-distilled student on both.

- **TIDE vs. naive, shared-tokenizer** (TIDE-Interp vs. token-level KL). Significant on seven of eight tasks without correction and on four with it: GSM8K (189 gained/131 lost, $p = 1.4 \times 10^{-3}$), MATH (332/240, $p = 1.4 \times 10^{-4}$), MMLU-Pro (1020/837, $p = 2.4 \times 10^{-5}$), and MBPP (54/20, $p = 9.6 \times 10^{-5}$). BBH ($p = 0.015$), HumanEval ($p = 0.021$), and HellaSwag ($p = 0.023$) do not survive the correction.
- **TIDE vs. naive, cross-tokenizer** (TIDE-Rev vs. forward CALM). GSM8K (150/101, $p = 2.4 \times 10^{-3}$), BBH (643/441, $p < 10^{-6}$), and MMLU-Pro ($p = 1.1 \times 10^{-3}$) survive the correction. The other five tasks are not individually significant, and none shows a significant regression (e.g., HellaSwag $p = 0.16$, MATH $p = 0.57$).
- **Best TIDE per pipeline vs. non-distilled BD3LM.** The cross-tokenizer GSM8K gain is significant (163/87, $p = 2 \times 10^{-6}$; +5.77 in this run vs. +6.68 in Table 1). Most other per-task differences, including the code tasks with their small test sets (164 and 500 problems), are not individually significant, whereas the shared-tokenizer student's MATH drop is significant without correction ($p = 6.8 \times 10^{-3}$).

Figure 3 shows the per-benchmark changes of Table 1 and marks the gains over the naive objective that survive the correction. For a single model, the binomial 95% confidence interval is about ± 2.7 points on GSM8K (1,319 items); the paired tests are sharper because they cancel item difficulty shared by the two models. None of these tests covers variation across training seeds.

D GRADIENT ANALYSIS

This appendix derives the gradients behind the choice of Reverse CALM in Section 2.3. Since $p_s^c = \exp(\mathrm{LP}_S^c/T)$, the chain rule gives $\partial p_s^c/\partial\theta = (p_s^c/T)\,\partial \mathrm{LP}_S^c/\partial\theta$, where LP_S^c is the student's log-probability of chunk c; both gradients below carry this factor.

Forward CALM gradient. Differentiating the forward loss with respect to p_s^c produces a ratio p_t^c/p_s^c, but the factor p_s^c from the chain rule cancels it:

$$\frac{\partial \mathcal{L}_{\text{fwd}}}{\partial\theta} = -\Big(\frac{p_t^c}{p_s^c} - \frac{1-p_t^c}{1-p_s^c}\Big)\frac{\partial p_s^c}{\partial\theta} = -\frac{1}{T}\Big(p_t^c - \frac{(1-p_t^c)\,p_s^c}{1-p_s^c}\Big)\frac{\partial \mathrm{LP}_S^c}{\partial\theta}. \tag{13}$$

The gradient therefore stays finite as $p_s^c \to 0$, and also as $p_s^c \to 1$, where $\partial \mathrm{LP}_S^c/\partial\theta$ vanishes at the same rate as $1 - p_s^c$. For a one-token chunk with $T = 1$, the update on the chunk's logit is exactly $p_t^c - p_s^c$: forward CALM regresses p_s^c onto p_t^c and stops only at $p_s^c = p_t^c$, wherever p_t^c lies, including on chunks where it reflects alignment noise.

Reverse CALM gradient. The reverse loss is linear in p_s^c, so its gradient is

$$\frac{\partial \mathcal{L}_{\text{rev}}}{\partial\theta} = -\sum_c \frac{\partial p_s^c}{\partial\theta}\cdot \log\frac{p_t^c}{1-p_t^c}. \tag{14}$$

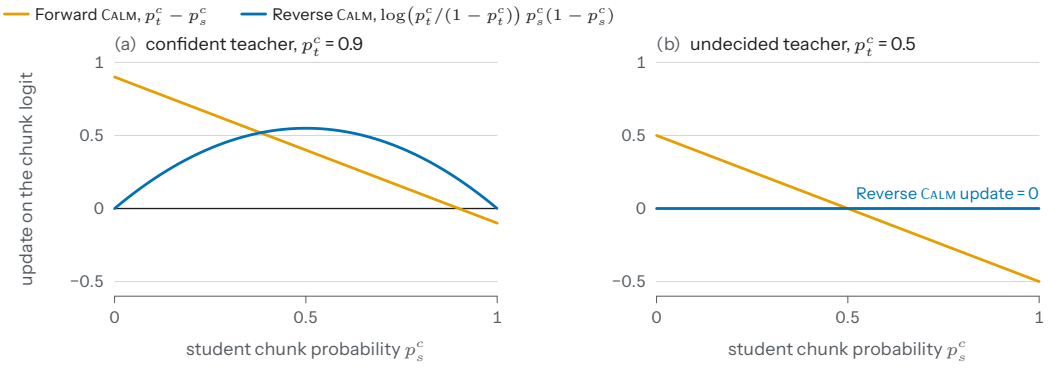

Figure 6: Update on the logit of a one-token chunk (T=1) as a function of the student probability p_s^c, for a confident teacher (p_t^c=0.9, left) and an undecided one (p_t^c=0.5, right). Forward CALM, $p_t^c - p_s^c$, pulls p_s^c toward p_t^c from any starting point. Reverse CALM, $\log \frac{p_t^c}{1-p_t^c}\, p_s^c(1-p_s^c)$, vanishes at both ends of p_s^c and is zero when the teacher is undecided, as it tends to be on poorly aligned chunks. Both stay bounded. This controlled, training-free illustration shows the noise filtering that motivates Reverse CALM.

The coefficient $\log \frac{p_t^c}{1-p_t^c}$ depends only on the fixed teacher probabilities, its sign is set by whether p_t^c exceeds 0.5, and it is bounded: $|\log \frac{p_t^c}{1-p_t^c}| \leq |\log \frac{1-\epsilon}{\epsilon}|$ for $p_t^c \in [\epsilon, 1-\epsilon]$. The student concentrates its updates on the teacher's high-probability modes. For a one-token chunk with $T = 1$, the update on the chunk's logit is $\log \frac{p_t^c}{1-p_t^c} \cdot p_s^c(1-p_s^c)$.

Dual-end noise filtering. The reverse update is filtered on both ends: (1) chunks the teacher is unsure about ($p_t^c \approx 0.5$), as poorly aligned chunks tend to be, zero the teacher-end coefficient; (2) chunks with low p_s^c contribute a small $\frac{\partial p_s^c}{\partial \theta}$, suppressing the student-end signal. Forward CALM has neither filter: at $p_t^c = 0.5$ it still pulls p_s^c toward 0.5, and for $p_t^c > 0.5$ its update is largest on the chunks the student finds least likely (Figure 6). Both gradients stay bounded; the two directions differ in which chunks drive the update.

Relation to Bernoulli KL and what Reverse CALM optimizes. Writing the Bernoulli KL

$$\mathrm{KL}_{\mathrm{Bern}}(p_s^c \| p_t^c) = p_s^c \log \frac{p_s^c}{p_t^c} + (1-p_s^c) \log \frac{1-p_s^c}{1-p_t^c}, \tag{15}$$

the reverse loss satisfies the exact identity $\mathcal{L}_{\mathrm{rev}} = \mathrm{KL}_{\mathrm{Bern}}(p_s^c \| p_t^c) + H(p_s^c)$, where $H(p_s^c) = -p_s^c \log p_s^c - (1-p_s^c)\log(1-p_s^c)$ is the student's Bernoulli entropy. In the forward case, the cross-entropy equals the KL up to the teacher's entropy, which is constant; here the entropy term belongs to the student and depends on θ. Hence $\mathcal{L}_{\mathrm{rev}}$ is not a Bernoulli KL up to a constant, and its minimizer is not $p_s^c = p_t^c$. Because $\mathcal{L}_{\mathrm{rev}}$ is linear in p_s^c with bounded slope $-\log \frac{p_t^c}{1-p_t^c}$, it drives $p_s^c \to 1$ when $p_t^c > 0.5$ and $p_s^c \to 0$ otherwise. We therefore interpret Reverse CALM as a *bounded, teacher-thresholded mode-seeking* surrogate: it transfers the teacher's dominant per-chunk decision and ignores chunks on which the teacher is undecided, which suits noisy cross-tokenizer alignment.

TIDAL and reverse direction. TIDAL interpolates the target of a forward objective, shifting the emphasis from the student to the teacher as λ_t grows (Eq. 10). Substituting the mixed probability $(1-\lambda_t)\, p_s^c + \lambda_t\, p_t^c$ for p_s^c in the reverse loss would only scale its gradient by $(1-\lambda_t)$, which falls to 0.1 at low noise late in training (Eq. 1), exactly where the teacher is most reliable. We therefore use TIDAL only with forward-direction objectives and report no results for its combination with Reverse CALM.

E CASE STUDY

To see what distillation transfers beyond aggregate benchmark gains, we measure dark knowledge transfer and analyze errors qualitatively.

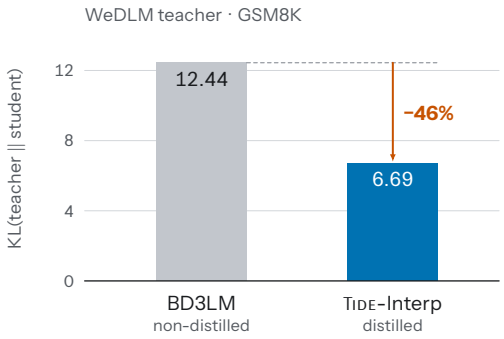

Figure 7: KL divergence to the WeDLM teacher on GSM8K. The distilled student's KL divergence is 46% lower than that of the non-distilled baseline (6.69 vs. 12.44).

Dark knowledge transfer. In the shared-tokenizer pipeline, we measure the KL divergence between the student's and the teacher's predictions at intermediate denoising timesteps. TIDE-Interp reduces the KL divergence to the WeDLM teacher on GSM8K by 46% (6.69 vs. 12.44; Figure 7), which indicates that the distilled student's predictions move closer to the teacher's. We omit the cross-tokenizer comparison (LLaDA2 to the student) because the vocabularies do not align.

Qualitative error analysis. We select four problems on which the non-distilled baseline fails and exactly one distilled variant succeeds (Table 12); Table 11 shows two of them in detail. On GSM8K #39, a multi-step distance problem, the baseline confuses time allocation with speed; TIDE-Rev (distilled from LLaDA2) reaches the correct total, but only because two errors cancel, so this case does not show better reasoning. On HumanEval #50, the baseline calls `encode_shift` again instead of reversing the shift, and only TIDE-Interp (distilled from WeDLM) correctly subtracts 5. These examples suggest that the two teachers can impart different knowledge: in these cases, LLaDA2 improves type-level awareness in code generation (HumanEval #11), while WeDLM fixes a proportional-reasoning error (GSM8K #77) and helps with algorithmic inversion (HumanEval #50).

VERBATIM MODEL OUTPUTS

Below are the unedited model outputs for the four cases in Table 12; only the long explanation that TIDE-Rev appends on HumanEval #50 is left out (see the note there).

GSM8K #39, Baseline (No Distill): Incorrect.

> Dana can skip at a rate of speed 3 miles per hour, so she can run at a rate of speed $3 \times 2 = 6$ miles per hour. If she spends one-third of the time running, she will run $6/3 = 2$ miles per hour. If she spends two-thirds of the time walking, she will walk $6/3 = 2$ miles per hour. The total distance Dana can travel in 6 hours is $2 \times 2 + 4 \times 2 = 12 + 4 = 16$ miles. #### 16. The answer is: 16

GSM8K #39, TIDE-Rev (LLaDA2): Correct.

> Dana can skip at a rate of 3 miles per hour, so she can run at a rate of $3 \times 2 = 6$ miles per hour. If she spends one-third of the time running, she will run for $6/3 = 2$ hours. If she spends two-thirds of the time walking, she will walk for $6/3 = 2$ hours. In 2 hours, she will run for $2 \times 6 = 12$ miles. In 2 hours, she will walk for $2 \times 3 = 6$ miles. Therefore, Dana can travel a total of $12 + 6 = 18$ miles in six hours. #### 18. The answer is: 18

GSM8K #39, TIDE-Interp (WeDLM): Incorrect.

> Dana can skip at a rate of 3 miles per hour, so she can run at a rate of $3 \times 2 = 6$ miles per hour. If she spends one-third of the time running, she will spend $6/3 = 2$ hours running. If she spends two-thirds of the time walking, she will spend $6/3 = 2$ hours walking. In total, she will spend $2 + 2 = 4$ hours running and walking. If she can run at a rate of 6 miles per hour, she will travel $6 \times 6 = 36$ miles in 6 hours running. If she can skip at a rate of 3 miles

Table 11: Case studies in detail: reasoning chains and generated code of the baseline and the distilled variants.

GSM8K #39: Multi-Step Distance Calculation

Prompt: Dana can run at a rate of speed four times faster than she can walk, but she can skip at a rate of speed that is half as fast as she can run. If she can skip at 3 miles per hour, how many miles can she travel in six hours if she spends one-third of the time running and two-thirds of the time walking? (*Ground Truth:* 18)

Model	Output	Analysis
Baseline (No Distill)	Dana can skip at a rate of speed 3 miles per hour, so she can run at a rate of speed $3 * 2 = 6$ miles per hour. If she spends one-third of the time running, she will run $6/3 = 2$ miles per hour. If she spends two-thirds of the time walking, she will walk $6/3 = 2$ miles per hour. The total distance Dana can travel in 6 hours is $2 * 2 + 4 * 2 = 12 + 4 = 16$ miles. Answer: **16**	Confuses time allocation ($6/3 = 2$ hours) with speed ("2 miles per hour").
TIDE-Rev (LLaDA2)	Dana can skip at a rate of 3 miles per hour, so she can run at a rate of $3 * 2 = 6$ miles per hour. If she spends one-third of the time running, she will run for $6/3 = 2$ hours. If she spends two-thirds of the time walking, she will walk for $6/3 = 2$ hours. In 2 hours, she will run for $2 * 6 = 12$ miles. In 2 hours, she will walk for $2 * 3 = 6$ miles. Therefore, Dana can travel a total of $12 + 6 = 18$ miles in six hours. Answer: **18**	Right total from two errors that cancel: 2 hours of walking (should be 4) at the 3 mph skipping speed (walking is 1.5 mph).
TIDE-Interp (WeDLM)	Dana can skip at a rate of 3 miles per hour, so she can run at a rate of $3 * 2 = 6$ miles per hour. If she spends one-third of the time running, she will spend $6/3 = 2$ hours running. If she spends two-thirds of the time walking, she will spend $6/3 = 2$ hours walking. In total, she will spend $2 + 2 = 4$ hours running and walking. If she can run at a rate of 6 miles per hour, she will travel $6 * 6 = 36$ miles in 6 hours running. If she can skip at a rate of 3 miles per hour, she will travel $3 * 6 = 18$ miles in 6 hours skipping. Therefore, she will travel a total of $36 + 18 = 54$ miles. Answer: **54**	Gives walking 2 hours (should be 4), then runs and skips for the full 6 hours each.

HumanEval #50: Algorithmic Inversion

Prompt: The function `encode_shift` shifts each character by +5 in the alphabet. Implement the function `decode_shift`: reverse the encoding, and return the original string.

Model	Output	Analysis
Baseline (No Distill)	`return encode_shift(s)`	Calls `encode_shift` again (+5) instead of reversing (-5).
TIDE-Rev (LLaDA2)	`return encode_shift(s)` + 500-char explanation that incorrectly justifies this as valid decoding.	Same erroneous code as Baseline, but with a longer (incorrect) explanation appended.
TIDE-Interp (WeDLM)	`return "".join( [chr(((ord(ch) - 5 - ord("a")) % 26) + ord("a")) for ch in s])`	Correctly subtracts 5 to reverse the shift.

Table 12: Summary of the case studies. The non-distilled baseline fails on all four problems, and the two pipelines fix different ones. ✓ marks a correct final answer; on #39, TIDE-Rev reaches it only through two errors that cancel (Table 11).

Case & Error Type	Base	TIDE-Rev	TIDE-Interp
Grade-school math (GSM8K)			
#39: rate-distance confusion in multi-step problem	×	✓	×
#77: proportional relationships reversed	×	×	✓
Code generation (HumanEval)			
#11: bitwise XOR on string characters	×	✓	×
#50: decode calls encode (no reversal)	×	×	✓

per hour, she will travel $3 \times 6 = 18$ miles in 6 hours skipping. Therefore, she will travel a total of $36 + 18 = 54$ miles in 6 hours. #### 54. The answer is: 54

GSM8K #77, Baseline (No Distill): Incorrect.

Sarah does 400 pounds of laundry. Raymond does $4 \times 400 = 1600$ pounds of laundry. Raymond does $1600/2 = 800$ pounds of laundry. Raymond does $800 - 400 = 400$ pounds of laundry than David.

GSM8K #77, TIDE-Rev (LLaDA2): Incorrect.

$400/4 = 100$ pounds of laundry. Raymond does $100/2 = 50$ pounds of laundry. The difference is $100 - 50 = 50$ pounds.

GSM8K #77, TIDE-Interp (WeDLM): Correct.

Sarah does 400 pounds of laundry. Sarah does 4 times as much laundry as David, so David does $400/4 = 100$ pounds of laundry. Raymond does half as much laundry as Sarah, so Raymond does $400/2 = 200$ pounds of laundry. The difference in the amount of laundry Raymond and David do is $200 - 100 = 100$ pounds.

HumanEval #11 (`string_xor`), Baseline (No Distill): Failed.

```
result = ""
for a, b in zip(a, b):
    result += str(a ^ b)
return result
```

HumanEval #11 (`string_xor`), TIDE-Rev (LLaDA2): Passed.

```
result = ''
for a, b in zip(a, b):
    if a != b:
        result += '1'
    else:
        result += '0'
return result
```

HumanEval #11 (`string_xor`), TIDE-Interp (WeDLM): Failed.

```
result = ""
for a, b in zip(a, b):
    result += str(a ^ b))
return result[:-1]
```

HumanEval #50 (`decode_shift`), Baseline (No Distill): Failed.

```
return encode_shift(s)
```

HumanEval #50 (`decode_shift`), TIDE-Rev (LLaDA2): Failed.

```
    return encode_shift(s)
```

Note: The generated code is identical to the Baseline. TIDE-*Rev also produces a long explanation that incorrectly describes this as a valid decoding strategy, which indicates that the model does not understand that decoding requires reversing the shift.*

HumanEval #50 (`decode_shift`), TIDE-Interp (WeDLM): Passed.

```
    return "".join(
        [chr(((ord(ch) - 5 - ord("a"))
              % 26) + ord("a"))
         for ch in s])
```