# FINE-TUNING DISCRETE DIFFUSION LLMS BY DISTILLING FROM AUTOREGRESSIVE TEACHERS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Discrete diffusion language models (dLLMs) have recently emerged as an alternative paradigm for text generation, with the potential to achieve more efficient inference than autoregressive (AR) models. However, AR models still continue to achieve stronger generation quality than dLLMs. The AR models provide strong token-level predictive distributions and benefit from more established training methods. We explore how to leverage AR models during training while preserving the parallel generation capability of dLLMs at inference time. However, directly distilling AR knowledge into dLLMs is challenging because AR models and dLLMs condition on different contexts, leading to mismatched predictive distributions. To address this challenge, we propose Weight-annEaled Autoregressive distillatioN for dLLMs (WEAN), the first AR-to-dLLM distillation method designed for downstream fine-tuning. WEAN controls where and when AR supervision is applied: it selectively transfers AR knowledge to reliable token positions and dynamically adjusts the distillation strength according to denoising steps and training stages. Specifically, WEAN focuses distillation on structural tokens, where AR and dLLM predictions are more compatible, and gradually reduces AR guidance as the dLLM adapts to the target distribution. Experimental results show that WEAN consistently improves both generation quality and downstream utility compared with the strongest baselines by 17.0% and 2.03% on studied text datasets. WEAN generalizes to code and mathematics domains, showing its broad applicability. We release the replication package on the anonymous link.[1]

## 1 INTRODUCTION

Diffusion-based large language models (dLLMs) have recently emerged as an alternative approach to text generation (Sahoo et al., 2024; Nie et al., 2026; Ye et al., 2025). Unlike autoregressive (AR) models, which generate tokens one at a time, dLLMs start from a fully masked sequence and gradually recover text through multiple denoising steps, predicting many tokens in parallel (Austin et al., 2021; Sahoo et al., 2024; Liu et al., 2026b). This parallel generation process provides potential inference speed advantages over AR models (Khanna et al., 2025; Song et al., 2025). Recent commercial LLMs have also explored the dLLM paradigm, like Mercury (Khanna et al., 2025) and Google Gemini Diffusion (Google DeepMind, 2025). Fine-tuning is the main approach for adapting pretrained dLLMs to downstream applications. Since training dLLMs from scratch requires large-scale data and computation (Gong et al., 2025; Bie et al., 2025), most users instead start from pretrained models such as LLaDA (Nie et al., 2026) and adapt them with limited task-specific data.

Current dLLM fine-tuning methods mainly improve training by reweighting tokens or diffusion steps (Xu et al., 2025; Parashar et al., 2026), but they still rely only on information available in the task data. Distillation provides an alternative by introducing additional supervision from a stronger teacher model (Hinton et al., 2015). In particular, AR models continue to achieve stronger generation quality than dLLMs. For example, 'Qwen3-8B' substantially outperforms 'LLaDA-8B' on generation benchmarks (Bertolani et al., 2026). Their training and fine-tuning techniques are also more mature (Nie et al., 2026; Ye et al., 2025). Therefore, it is natural to ask whether AR models can provide useful guidance for improving dLLM fine-tuning. The AR model is only used during training, while the final model remains a dLLM and preserves its parallel generation ability.

[1] https://anonymous.4open.science/r/DLLM-Distillation-3396



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.

However, applying AR distillation to dLLM fine-tuning is non-trivial. The AR teacher and dLLM student condition on different contexts. The AR model predicts each token from a left-to-right prefix, whereas the dLLM predicts masked tokens conditioned on an arbitrary set of visible tokens. Their predictive distributions may differ due to different conditioning contexts, making direct distribution matching unreliable (Zhang et al., 2025; Su et al., 2026). Existing AR-to-dLLM distillation mainly focuses on building dLLMs, converting AR models into dLLMs, or accelerating sampling (Gong et al., 2025; Zhang et al., 2025; 2026a; Su et al., 2026). They treat the AR model as the primary target and encourage the dLLM to reproduce its behavior. This objective is different from downstream fine-tuning, where the goal is to improve an existing dLLM on a target dataset rather than make it imitate an AR model. This paper studies two questions: (1) *whether AR models can provide useful supervision during dLLM fine-tuning*, and (2) *how to use AR knowledge while avoiding the mismatch between AR and dLLM predictions*. We propose Weight-annEaled Autoregressive distillatioN for dLLMs (WEAN), the first AR-to-dLLM distillation method designed for downstream fine-tuning.

For the first question, both our analysis and experiments give a positive answer. The reason is that AR distillation transfers local token distributions rather than generated sequences. Compared with standard dLLM training, which supervises masked positions only with ground-truth tokens (Sahoo et al., 2024), AR models provide soft distributions over alternative tokens and capture additional information about plausible continuations (Touvron et al., 2023). However, this supervision is only beneficial when the AR and dLLM conditioning contexts lead to compatible predictions. When the visible context of the dLLM differs substantially from the AR prefix, the AR distribution may become inconsistent with the student condition, introducing unreliable guidance. Our findings indicate that AR supervision is effective when used as a selective and temporary signal during dLLM fine-tuning rather than as a persistent imitation target.

For the second question, based on this observation, WEAN controls the AR signal across token types, denoising steps, and training stages. First, we apply distillation only to structural tokens, such as punctuation marks and delimiters. These tokens mainly determine text organization and are more predictable from local context, reducing the mismatch between AR and dLLM predictions. Second, we anneal the distillation strength during both denoising and training. During denoising, the AR model receives less weight when most tokens remain masked, and the dLLM has limited context, while its influence increases as more tokens are revealed. During training, the distillation weight gradually decreases to zero, allowing the dLLM to first benefit from additional supervision and later adapt to the target data distribution without excessive teacher bias.

Our experiments in Section 4.3 validate these design choices and show the importance of controlling the strength of AR supervision. Unlike prior methods that progressively increase the teacher's influence over the training (Zhang et al., 2026b), WEAN gradually reduces the AR signal as the dLLM becomes better aligned with the target distribution. We evaluate WEAN by fine-tuning 'LLaDA-8B-Instruct' (Nie et al., 2026) and 'Dream-7B' (Ye et al., 2025) on three text-generation tasks spanning conditional and unconditional generation. Across tasks, WEAN improves both utility and generation quality over standard fine-tuning by 17.0% and 2.03%. We extend WEAN to code and mathematics datasets, where it consistently improves performance.

Our contributions are summarized as follows. ❶ We propose WEAN, the first method that uses AR distillation to improve downstream dLLM fine-tuning. ❷ We provide an analysis of the compatibility between AR and dLLM objectives, identifying when and where AR supervision is beneficial. ❸ Experiments across three generation tasks and two dLLMs show that WEAN improves the utility and quality of generated data compared with existing fine-tuning methods. ❹ We show that WEAN generalizes to additional domains, including code and mathematics, showing its broad applicability.

## 2 PRELIMINARIES

### 2.1 DIFFUSION LARGE LANGUAGE MODELS

DLLMs formulate text generation as an iterative denoising process over discrete token sequences (Sahoo et al., 2024; Nie et al., 2026). Given a clean token sequence $\mathbf{x}_0 = (x_1, \ldots, x_n)$, a forward diffusion process progressively corrupts the sequence by randomly masking tokens according to a predefined noise schedule, producing a corrupted sequence $\mathbf{x}_t$ at diffusion timestep $t$. The reverse process learns to recover the original sequence by predicting the masked tokens conditioned on the corrupted sequence, $p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t, t)$, where $\theta$ denotes the model parameters and $t$ denotes the

Table 1: Comparison with existing dLLMs distillation method. 'Shared vocab.' is that the teacher must use the student's tokenizer. 'Self' is that the teacher is the student's initialization checkpoint.

| Method | Teacher | Distillation Usage | Cross-Tokenizer | Cached Logits | Target |
|---|---|---|---|---|---|
| TCSM (Zhang et al., 2025) | AR (same vocab.) | Pretraining | × | × | Teacher |
| Gumbel (Zhang et al., 2026a) | AR (same vocab.) | Pretraining | × | × | Teacher |
| OPDLM (Su et al., 2026) | AR (self) | AR→dLLM conversion | × | × | Teacher |
| TIDE (Zhang et al., 2026b) | dLLM | Compression | ✓ | × | Teacher |
| WEAN (Ours) | Any AR | Fine-tuning | ✓ | ✓ | Data |

diffusion timestep. At each training step, a diffusion timestep is randomly sampled, and the model minimizes the reconstruction loss of the original sequence, $\mathcal{L}_{\text{diff}} = -\mathbb{E}_{(\mathbf{x}_0,t)}\left[\log p_\theta(\mathbf{x}_0 \mid \mathbf{x}_t, t)\right]$, where $\mathbf{x}_0$ is the clean sequence and $\mathbf{x}_t$ is the corresponding corrupted sequence at timestep $t$.

For downstream adaptation, pretrained dLLMs are fine-tuned with supervised objectives on task-specific data (Zhou et al., 2026). Given a training example $\mathbf{x}$, a subset of tokens is masked according to the diffusion schedule, producing a masked sequence $\mathbf{x}_t$. The model is then trained to recover the original sequence conditioned on the masked input. Specifically, the standard dLLM fine-tuning objective can be written as, $\mathcal{L}_{\text{SFT}} = -\mathbb{E}_{\mathbf{x},t}\left[\sum_{i\in\mathcal{M}_t} \log p_\theta\left(x_i \mid \mathbf{x}_t, t\right)\right]$, where $\mathcal{M}_t$ denotes the set of masked positions at timestep $t$, and $\mathbf{x}_t$ represents the partially masked sequence.

### 2.2 Autoregressive Models

AR models generate text sequentially, predicting each token based on the preceding tokens (Brown et al., 2020; Touvron et al., 2023), $p(\mathbf{x}) = \prod_{i=1}^{n} p(x_i|x_{<i})$, where $x_i$ is the token at position $i$, and $x_{<i} = (x_1, \ldots, x_{i-1})$ denotes all tokens before position $i$. Thus, the probability of the full sequence is the product of the conditional probability of each token given its preceding tokens. AR models are trained using maximum likelihood. At each position, the model is given the ground-truth preceding tokens and learns to predict the next token by minimizing the negative log-likelihood, $\mathcal{L}_{\text{AR}} = -\sum_{i=1}^{n} \log p_\theta(x_i|x_{<i})$. where $\theta$ denotes the model parameters, $x_i$ is the ground-truth token at position $i$, and $x_{<i}$ denotes all tokens before position $i$. Minimizing this loss encourages the model to assign a higher probability to the correct next token (Touvron et al., 2023).

During inference, an AR model generates tokens sequentially from left to right, with each new token conditioned on the previously generated tokens. Thus, AR decoding requires sequential model calls, which can make generation slow for long sequences (Leviathan et al., 2023; Liu et al., 2025; 2026a).

### 2.3 Distribution Gap between dLLM and AR Model

Although both AR and dLLMs model the same text distribution, they represent different conditional distributions. An AR model predicts each token based on a complete left-to-right prefix, $p_{\text{AR}}(x_i|x_{<i})$, whereas a dLLM predicts masked tokens conditioned on an arbitrary subset of visible tokens, $p_{\text{dLLM}}(x_i|\mathbf{x}_t, t)$. This difference leads to a context mismatch between AR teachers and dLLM students. When the masking level is low, $\mathbf{x}_t$ contains most tokens, and the dLLM prediction can be close to the AR prediction. However, when the masking level is high, the dLLM must predict from substantially less context, while the AR model still conditions on the complete preceding sequence. Therefore, directly matching the two distributions may introduce unreliable supervision.

In addition, dLLMs predict multiple masked tokens in parallel, $p_\theta(\mathbf{x}_{\mathcal{M}_t}|\mathbf{x}_t, t) \approx \prod_{i\in\mathcal{M}_t} p_\theta(x_i|\mathbf{x}_t, t)$, which enables efficient parallel decoding but differs from the sequential dependency modeling of AR models. These differences motivate our selective AR distillation strategy, which controls the AR signal based on when and where its predictions are reliable.

### 2.4 dLLMs Distillation

Knowledge distillation transfers knowledge from a teacher model to a student model through its predictions (Hinton et al., 2015). For dLLMs, existing methods mainly use distillation to reduce sampling steps, construct a dLLM from an AR model, or compress a large dLLM into a smaller one. In these settings, the student is generally trained to reproduce behavior provided by the teacher. For sampling acceleration, CDLM (Kim et al., 2025) and learnable sampler distillation (Fu et al., 2026) reduce the number of denoising steps required at inference time. TCSM, Gumbel Distillation,

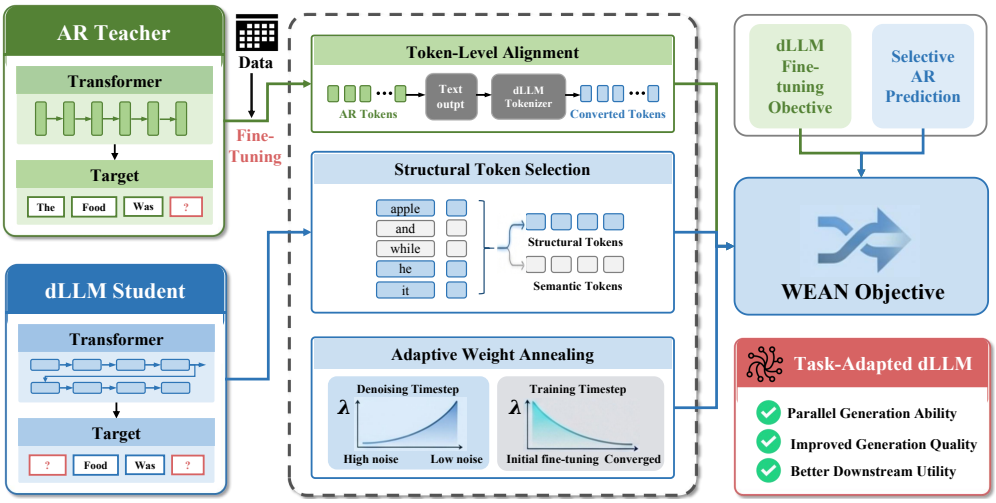



Figure 1: Overview of WEAN. An AR teacher is first fine-tuned on downstream data and aligned with the dLLM vocabulary space. WEAN selectively transfers AR knowledge through structural token selection and adaptive weight annealing, controlling where and when AR supervision is applied. The dLLM preserves parallel generation ability while improving generation quality and utility.

and OPDLM instead transfer knowledge from an AR model while *constructing or converting* a dLLM (Zhang et al., 2025; 2026a; Su et al., 2026). These methods require the teacher and student to share a tokenizer or originate from the same checkpoint. TIDE studies model compression, distilling a large dLLM into a smaller one while supporting different tokenizers (Zhang et al., 2026b).

Table 1 shows that WEAN differs in both distillation stage and objective. We use an AR to improve fine-tuning of an already pretrained dLLM, while the task data remains the target distribution. Because AR and dLLM may use different tokenizers, their token-level distributions cannot be directly compared. We address this problem to allow cross-family and larger AR models to serve as teachers. Moreover, in WEAN, the AR teacher provides its supervision before fine-tuning and does not need to run alongside the student during training. This reduces the memory and computation required for using large AR teachers. More related work for this paper is provided in Appendix A.

# 3 Methodology

## 3.1 Why AR Models Can Guide dLLMs

Although AR models and dLLMs use different generation procedures, they approximate the same underlying data distribution. An AR model factorizes, $p_\phi(\mathbf{x}) = \prod_i p_\phi(x_i|x_{<i})$, while a dLLM learns, $p_\theta(x_i|\mathbf{x}_t, t)$. Therefore, AR predictions can serve as auxiliary supervision by providing an alternative conditional view of the same data distribution. Standard dLLM training optimizes, $-\log p_\theta(x_i|\mathbf{x}_t, t)$, while AR distillation minimizes, $D_{\mathrm{KL}}(p_\phi(x_i|x_{<i})\|p_\theta(x_i|\mathbf{x}_t, t))$.

However, the effectiveness of this supervision depends on the compatibility between the conditional distributions associated with the two prediction tasks. At a masked position $i \in \mathcal{M}_t$, the AR teacher and dLLM student condition on different observations: $c_i^{\mathrm{AR}} = x_{<i}$, $c_i^{\mathrm{dLLM}} = \mathbf{x}_t$. To isolate the effect of conditioning from model approximation errors, let $p_*$ denote the target data distribution. Because the two contexts provide different information about $x_i$, these conditional distributions may differ, $p_*(x_i \mid c_i^{\mathrm{AR}}) \neq p_*(x_i \mid c_i^{\mathrm{dLLM}}, t)$. We define this intrinsic conditional mismatch as,

$$\Delta(t) = D_{\mathrm{KL}}\left(p_*(x_i \mid x_{<i}) \,\|\, p_*(x_i \mid \mathbf{x}_t, t)\right), \tag{1}$$

where $p_*(x_i \mid x_{<i})$ and $p_*(x_i \mid \mathbf{x}_t, t)$ denote the ground-truth conditional distributions of the token at position $i$ given the complete prefix and the masked sequence at timestep $t$. We use $\Delta(t)$ as shorthand for the mismatch. When the AR and dLLM conditioning contexts become more compatible, $\Delta(t)$ becomes smaller, and AR supervision is more reliable. Conversely, when the conditioning context of the teacher differs substantially from that of the student, the teacher prediction becomes

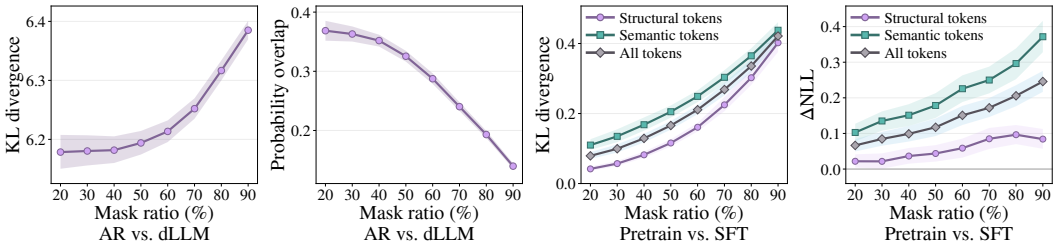



Figure 2: The left panels compare AR and dLLM predictions on masked structural tokens at different mask ratios. The right panels show how SFT changes AR and dLLM predictions by token type, measured by pretrained-to-SFT KL divergence and the reduction in NNL of tokens in the real text ($\Delta$NLL). Shaded bands show 95% confidence intervals. We provide details of our motivation experiments in Appendix C.1.

less aligned with the student's prediction objective, reducing the effectiveness of direct distillation. Thus, AR predictions provide a meaningful source of auxiliary supervision for dLLM fine-tuning.

A smaller $\Delta(t)$ indicates closer conditional distributions associated with the AR and dLLM prediction tasks. Conversely, a larger $\Delta(t)$ indicates a greater mismatch, so directly matching the teacher may introduce conflicting supervision even when the teacher accurately learns its own conditional distribution. In Figure 2, we observe that there is a smaller gap between the AR teacher and dLLM at lower mask ratios, as measured by lower forward KL divergence and higher probability overlap. This observation motivates assigning greater weight to distillation at lower mask ratios.

The last two panels of Figure 2 further show that SFT changes the prediction distributions for structural tokens less than those for semantic tokens, as indicated by lower pretrained-to-SFT KL divergence. SFT also achieves smaller improvements in predicting structural tokens, measured by the reduction in the negative log-likelihood (NLL) of the corresponding tokens in the real text. This suggests that structural predictions are less sensitive to task-specific adaptation. Thus, we prioritize structural tokens as distillation targets. Together, these observations motivate assigning greater weight to structural-token distillation at lower mask ratios, where the AR–dLLM mismatch is smaller. We evaluate this token-selection choice in Section 4.3.

### 3.2 Overview

WEAN aims to improve dLLM fine-tuning by incorporating knowledge from an AR teacher. Specifically, given a pretrained AR model and a pretrained dLLM, WEAN first adapts the AR model to the target dataset and obtains token-level teacher distributions. Since the AR teacher and dLLM student use different token spaces and conditioning contexts, WEAN aligns the teacher predictions and selectively applies AR supervision during dLLM fine-tuning.

Figure 1 presents the overall workflow of WEAN. Specifically, WEAN is built upon three components. First, we align the AR teacher distribution with the dLLM prediction space to enable token-level distillation. Second, we identify reliable locations for knowledge transfer by restricting distillation to tokens where AR predictions are more compatible with dLLM predictions. Third, we dynamically adjust the distillation strength according to the denoising process and training progress. The final objective combines the original dLLM fine-tuning loss with the adaptive AR distillation loss. In Section 3.1, we analyze why AR models can serve as effective teachers for dLLMs. Then, Section 3.3 presents the technical details of WEAN in the following section.

### 3.3 Technical Details

**AR Teacher and Token-Level Alignment.** We first adapt the pretrained AR model to the downstream dataset. Specifically, given a dataset $\mathcal{D}$, the AR teacher is fine-tuned using SFT (Touvron et al., 2023) with the standard AR objective, $\phi^* = \arg\min_\phi -\mathbb{E}_{\mathbf{x}\sim\mathcal{D}} \sum_{i\in\mathcal{R}(\mathbf{x})} \log p_\phi(x_i \mid x_{<i})$, where $\mathcal{R}(\mathbf{x})$ is the set of token positions included in the SFT loss, and $\phi^*$ denotes the parameters of the adapted AR teacher. This step reduces the distribution gap between the pretrained AR model and the target data, allowing the teacher to provide more relevant token-level supervision during dLLM fine-tuning. For each sequence in the training dataset and each prediction position $i$, the AR teacher

outputs a distribution over all tokens in its vocabulary, $p_{\phi^*}(v|x_{<i}),\ v \in \mathcal{V}_{\text{AR}}$, where $\mathcal{V}_{\text{AR}}$ is the AR vocabulary. However, the AR teacher and dLLM student may use different tokenizers, resulting in different vocabulary spaces, $\mathcal{V}_{AR} \neq \mathcal{V}_d$, where $\mathcal{V}_d$ is the dLLM vocabulary. Therefore, the teacher distribution cannot be directly compared with the dLLM prediction distribution.

Thus, we align the AR teacher distribution to the dLLM vocabulary space before distillation. For an AR token $a \in \mathcal{V}_{\text{AR}}$, we decode it into text and tokenize it using the dLLM tokenizer. If an AR token corresponds to a single dLLM token, we define a mapping $m(a) = v$, where $v \in \mathcal{V}_d$. Since multiple AR tokens may correspond to the same dLLM token (Rust et al., 2021), their probabilities are aggregated. For example, if the dLLM tokenizer lowercases text, distinct AR tokens decoding to `A` and `a` could both map to the dLLM token `a`. The vocabulary-aligned teacher $\tilde{p}_{\phi^*}(v|x_{<i}) \propto \sum_{a:m(a)=v} p_{\phi^*}(a|x_{<i})$, where $a : m(a) = v$ denotes all AR tokens whose mapped dLLM token is $v$. The aggregated probabilities are normalized over the dLLM vocabulary, producing a teacher distribution $\tilde{p}_{\phi^*}$ that can be directly compared with the dLLM prediction distribution. We provide more details on tokenizer alignment in Appendix C.2.

**Where to Distill: Structural Token Selection.** After aligning the teacher distribution to the dLLM vocabulary space, a natural question is where AR supervision should be applied. Although AR models provide informative soft predictions, the quality of these predictions depends on the type of token being predicted. For semantic tokens, such as content-bearing nouns and adjectives, predictions often depend on long-range semantic information that may not be available in the dLLM context $\mathbf{x}_t$. Prior work shows that neural language models are highly sensitive to word order in nearby context, whereas distant context is used primarily as a coarse semantic field or topic (Khandelwal et al., 2018). For example, predicting a noun such as "apple" may depend on the topic and entities mentioned elsewhere in the context. Therefore, the AR prediction $p_{\phi^*}(x_i|x_{<i})$ can differ substantially from the dLLM prediction $p_\theta(x_i|\mathbf{x}_t, t)$. In contrast, structural tokens, such as punctuation marks, are mainly determined by local syntactic patterns. Recent work shows that LLMs rely on such explicit structural markers, particularly punctuation, to guide how contextual information is integrated (Liu & Ding, 2025). Empirical studies presented in Section 3.1 show that predictions at structural-token positions are less sensitive to missing semantic context.

Referring to previous works (Liu et al., 2026b; Yang et al., 2023), we use part-of-speech (POS) tags and token categories to identify structural tokens. We treat punctuation marks, delimiters, formatting symbols, and closed-class function words as structural tokens, while content words, such as nouns, verbs, adjectives, and adverbs, are treated as semantic tokens. Let $S$ denote the set of token positions identified as structural by the above rule. WEAN applies distillation only to masked positions belonging to $\mathcal{S}$, $\mathcal{I}_t = \mathcal{M}_t \cap \mathcal{S}$. The AR distillation loss is, $\mathcal{L}_{\text{KD}} = -\mathbb{E}_{t,\mathbf{x}_t}\left[\sum_{i\in\mathcal{I}_t} D_{\text{KL}}(\tilde{p}_{\phi^*}(x_i|x_{<i})\|p_\theta(x_i|\mathbf{x}_t, t))\right]$.

**When to Distill: Adaptive Distillation Scheduling.** Even for structural tokens, the reliability of AR supervision varies across denoising timesteps. At a large timestep $t$, fewer tokens are visible in $\mathbf{x}_t$, and the dLLM prediction condition differs more from the AR prefix. Meanwhile, the AR teacher continues to condition on the complete prefix $x_{<i}$. This creates a larger mismatch between the teacher and student distributions.

As denoising, the masking level decreases and more tokens are revealed, providing the dLLM with richer contextual information. Therefore, AR supervision becomes more reliable at later denoising stages. Based on this observation, since $t$ represents the masking probability, larger $t$ corresponds to earlier denoising stages with less available context. We reduce AR supervision when $t$ is large and increase it as denoising proceeds, $\lambda(t) = \alpha(1-t)$, where $\alpha$ controls the distillation strength.

During fine-tuning, the pretrained dLLM may initially benefit from additional token-level guidance from the AR teacher. As the student adapts to the target distribution, however, continued teacher supervision may constrain its learning of task-specific patterns. We therefore gradually reduce the teacher's influence over training. Let $s$ and $S$ denote the current and total training steps, respectively, and let $u = s/S$ and $u_0 = \min(0.2, \rho)$. We define,

$$\lambda_s(s) = 1 \text{ if } u < u_0; \quad \frac{\rho - u}{\rho - u_0} \text{ if } u_0 \le u < \rho; \quad \lambda_s(s) = 0 \text{ otherwise.} \tag{2}$$

For $\rho > 0$, the weight decreases from one to zero during the first $\rho S$ training steps and remains zero thereafter, allowing the student to continue adapting through the original denoising objective alone. Setting $\rho = 0$ disables distillation, while $\rho = 1$ anneals the weight over the entire training run.

Table 2: Results on text-generation fine-tuning tasks. 'Cate.' and 'C2ST-S.' are abbreviations of 'Category' and 'C2ST-Sim'. The best results are shown in **bold**.

| Method | Yelp | | | | OpenReview | | | | PubMed | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Cate.↑ | Rating↑ | C2ST-S.↑ | MAUVE↑ | Area↑ | Rating↑ | C2ST-S.↑ | MAUVE↑ | C2ST-S.↑ | MAUVE↑ |
| *LLaDA-8B-Instruct* | | | | | | | | | | |
| Pre-DLLM | 57.8 | 41.5 | 0.013 | 0.045 | 26.1 | 36.3 | 0.092 | 0.111 | 0.077 | 0.035 |
| SFT-DLLM | 71.8 | 67.9 | 0.190 | 0.120 | 45.6 | 40.2 | 0.256 | 0.342 | 0.091 | 0.036 |
| GIFT | 71.1 | 69.6 | 0.194 | 0.125 | 46.9 | 35.5 | 0.256 | 0.298 | 0.089 | 0.027 |
| LIFT | 68.5 | 68.6 | 0.208 | 0.176 | 43.9 | 38.0 | 0.334 | 0.473 | 0.108 | 0.039 |
| NaRA | 73.1 | 67.4 | 0.169 | 0.095 | 43.6 | 37.5 | 0.224 | 0.234 | 0.073 | 0.032 |
| WEAN (ours) | **73.8** | **70.1** | **0.238** | **0.284** | **48.4** | **41.9** | **0.388** | **0.531** | **0.125** | **0.042** |
| *Dream-v0-Instruct-7B* | | | | | | | | | | |
| Pre-DLLM | 70.7 | 61.4 | 0.038 | 0.025 | 40.8 | 41.5 | 0.042 | 0.028 | 0.048 | 0.015 |
| SFT-DLLM | 69.7 | 67.1 | 0.044 | 0.020 | 44.9 | 39.7 | 0.036 | 0.040 | 0.029 | 0.011 |
| GIFT | 74.3 | 66.0 | 0.055 | 0.051 | 44.8 | 38.3 | 0.055 | 0.032 | 0.040 | 0.012 |
| LIFT | 74.0 | 67.4 | 0.051 | 0.047 | 43.8 | 36.3 | 0.041 | 0.026 | 0.030 | 0.012 |
| NaRA | 68.5 | 67.1 | 0.046 | 0.030 | 47.9 | 41.0 | 0.069 | **0.091** | 0.028 | 0.013 |
| WEAN (ours) | **75.3** | **69.2** | **0.068** | **0.059** | **48.6** | **42.2** | **0.075** | 0.069 | **0.054** | **0.021** |

**Training Objective.** The final training objective of WEAN combines the standard dLLM fine-tuning objective and the selectively applied AR distillation loss,

$$\mathcal{L}_{\text{WEAN}} = \mathcal{L}_{\text{SFT}} + \lambda(s,t)\mathcal{L}_{\text{KD}}, \text{ where, } \lambda(s,t) = \lambda_s(s)\lambda_t(t). \tag{3}$$

Thus, $\lambda(s,t)$ controls the AR influence according to both the training and the denoising timestep. Appendix C.3 shows the pseudocode of WEAN.

## 4 Experiment Analysis

This section verifies the effectiveness of WEAN by answering four Research Questions (RQs) as follows. (**RQ1**) Does WEAN outperform the baseline methods? (**RQ2**) Why does WEAN effectively improve dLLM fine-tuning? (**RQ3**) How do the hyperparameters of WEAN affect the fine-tuning performance? (**RQ4**) Can WEAN be extended to other types of data fine-tuning?

### 4.1 Experimental Setup

**Models and Benchmarks.** We use 'LLaDA-8B-Instruct' (Nie et al., 2026) and 'Dream-v0-Instruct-7B' (Ye et al., 2025) as the pretrained dLLM generator backbones, and 'Qwen3-32B' (Yang et al., 2025) as the AR teacher. We evaluate on six benchmarks spanning text, code, and mathematics: `Yelp` (Yang et al., 2025), `OpenReview` (Xie et al., 2024), `PubMed` (Yu et al., 2023), `CodeNet` (Puri et al., 2021), `PrimeVul` (Ding et al., 2024), and `DM-Math` (Saxton et al., 2019). `Yelp` and `OpenReview` are conditional text-generation tasks with two downstream labels, whereas `PubMed` is used for unconditional generation. `CodeNet`, `PrimeVul`, and `DM-Math` evaluate the utility of generated code and mathematical question–answer pairs. The studied datasets elaborate on Section B.1. More implementations of WEAN are provided in Section 3.3.

**Baselines.** We compare WEAN with the following baselines. ❶ Pre-dLLM, which generates data using the pretrained dLLM without task-specific fine-tuning; ❷ SFT-dLLM, which applies standard dLLM fine-tuning without distillation; ❸ GIFT (Xu et al., 2025), ❹ LIFT (Parashar et al., 2026), and ❺ NaRA (Wang et al., 2026), which improve dLLM fine-tuning through token weighting, timestep-aware supervision, and noise-aware adaptation. We provide more details in Section B.2.

**Evaluation.** Each method produces the same number of synthetic examples. We train a downstream classifier or solver only on the generated data and evaluate it on the corresponding real test set. We report the classification accuracy for `Yelp` (Category and Rating) and `OpenReview` (Area and Rating), problem-classification accuracy for `CodeNet`, vulnerability-detection F1 for `PrimeVul`, and answer exact match (EM) for `DM-Math`. For the text benchmarks, we report C2ST-Sim (Lopez-Paz & Oquab, 2017) and MAUVE (Pillutla et al., 2022). We provide evaluation details in Appendix B.3.

Table 3: Ablation study results on Yelp, OpenReview, and PubMed. Higher is better for all metrics, and bold indicates the best result.

| Method | Yelp | | | | OpenReview | | | | PubMed | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Cate.↑ | Rating↑ | C2ST-S.↑ | MAUVE↑ | Area↑ | Rating↑ | C2ST-S.↑ | MAUVE↑ | C2ST-S.↑ | MAUVE↑ |
| Pre-AR-Dist | 70.4 | 62.5 | 0.112 | 0.034 | 31.1 | 36.8 | 0.146 | 0.186 | 0.059 | 0.023 |
| Pre-DLLM-Dist | 58.5 | 52.8 | 0.064 | 0.026 | 24.1 | 30.7 | 0.098 | 0.122 | 0.038 | 0.009 |
| SFT-DLLM-Dist | 65.7 | 57.9 | 0.081 | 0.029 | 28.0 | 36.3 | 0.124 | 0.159 | 0.051 | 0.017 |
| WEAN-S | 64.9 | 66.4 | 0.131 | 0.201 | 42.1 | 36.3 | 0.215 | 0.488 | 0.084 | 0.032 |
| WEAN-A | 70.7 | 68.0 | 0.196 | 0.235 | 46.7 | 39.8 | 0.289 | 0.504 | 0.114 | 0.039 |
| WEAN (ours) | **73.8** | **70.1** | **0.238** | **0.284** | **48.4** | **41.9** | **0.388** | **0.531** | **0.125** | **0.042** |

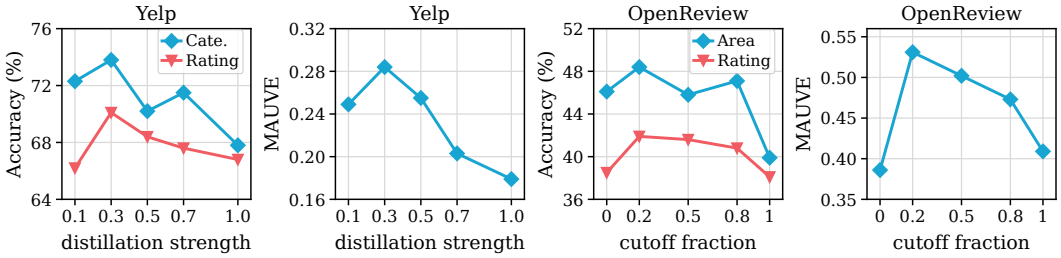



Figure 3: Hyperparameter analysis of WEAN on Yelp and OpenReview datasets with distillation strength $\alpha$ and cutoff fraction $\rho$.

### 4.2 Performance of WEAN (RQ1)

As shown in Table 2, WEAN achieves the best result on 19 of the 20 model and metric combinations across three datasets and two dLLM backbones. On 'LLaDA-8B-Instruct', WEAN outperforms all baselines. The gains are particularly clear on distributional similarity. Compared with the strongest baseline, WEAN improves MAUVE on Yelp from 0.176 to 0.284. On OpenReview, it improves C2ST-Sim from 0.334 to 0.388 and MAUVE from 0.473 to 0.531. The results remain strong on 'Dream-v0-Instruct-7B', where WEAN obtains the best result on nine of ten metrics. For example, compared with the strongest baseline for each metric, WEAN improves Category on Yelp from 74.3 to 75.3 and Rating from 67.4 to 69.2. On PubMed, it improves C2ST-Sim from 0.048 to 0.054.

Standard SFT and existing dLLM fine-tuning methods do not yield consistent improvements across settings. On 'Dream-v0-Instruct-7B', for example, SFT-DLLM underperforms Pre-dLLM on several metrics. Pre-AR-Dist also achieves weak distributional similarity and often performs worse than SFT-DLLM, despite using an AR teacher. These results indicate that directly introducing AR supervision is insufficient. In contrast, WEAN consistently improves both task-specific performance and distributional similarity, suggesting that its selective weighting and annealing mechanisms enable more effective AR-to-dLLM knowledge transfer.

### 4.3 Understanding the Effectiveness of WEAN (RQ2)

We examine how teacher choice and distillation targets affect performance. 'Pre-AR-Dist', 'Pre-DLLM-Dist', and 'SFT-DLLM-Dist' use pretrained AR, pretrained dLLM, and task-fine-tuned dLLM teachers, respectively. For dLLM-to-dLLM distillation, we use diffusiongemma-26B-A4B-it (Team et al., 2026) as the teacher. 'WEAN-S' and 'WEAN-A' replace the structural-token targets in WEAN with semantic tokens and all tokens. Table 3 shows that WEAN achieves the highest scores across all ten reported metrics.

SFT-DLLM-Dist consistently outperforms Pre-DLLM-Dist, showing that task adaptation improves the effectiveness of the dLLM teacher. For example, category accuracy on Yelp increases from 58.5% to 65.7%. However, Pre-AR-Dist outperforms both dLLM-teacher baselines on every metric. Sharing the diffusion formulation therefore does not automatically make a dLLM a better teacher for a diffusion student. Meanwhile, the gap between Pre-AR-Dist and WEAN shows that using an AR teacher alone is insufficient to achieve the performance of our full method.

WEAN-A outperforms WEAN-S across all metrics, indicating that including structural tokens improves over semantic-only distillation. However, restricting distillation to structural tokens, as in

Table 4: Utility of synthetic data generated by WEAN and baselines on code (`CodeNet` and `PrimeVul`) and mathematics (`DM-Math`) tasks. Bold indicates the best result in each column.

| Method | *LLaDA-8B-Instruct* | | | *Dream-v0-Instruct-7B* | | |
|---|---|---|---|---|---|---|
| | `CodeNet` | `PrimeVul` | `DM-Math` | `CodeNet` | `PrimeVul` | `DM-Math` |
| | Acc.↑ | F1↑ | EM↑ | Acc.↑ | F1↑ | EM↑ |
| Pre-DLLM | 90.7 | 6.76 | 1.48 | 90.8 | 4.52 | 1.42 |
| SFT-DLLM | 96.5 | 5.21 | 7.47 | 93.9 | 10.14 | 1.22 |
| GIFT | 93.2 | 5.12 | 7.34 | 10.0 | 10.17 | 1.22 |
| LIFT | 91.9 | 5.04 | 4.80 | 19.3 | 10.03 | 1.21 |
| NaRA | 93.9 | 5.08 | 7.67 | 68.5 | 10.35 | 1.25 |
| WEAN (ours) | **98.7** | **7.93** | **9.51** | **96.5** | **12.77** | **1.72** |

WEAN, further improves every metric over WEAN-A. Thus, distilling more tokens does not necessarily improve performance. Together with the smaller SFT-induced changes for structural tokens in Figure 2, these results support applying teacher supervision selectively to structural predictions while allowing semantic predictions to adapt through the original denoising objective.

### 4.4 Hyperparameter Analysis (RQ3)

Figure 3 studies two hyperparameters: the distillation strength $\alpha$, which scales teacher supervision, and the cutoff fraction $\rho$, which determines when distillation ends. For $\rho > 0.2$, the training-progress weight remains at one for the first $20\%$ of training and then decreases linearly to zero at step $\rho S$. For $0 < \rho \leq 0.2$, it remains at one until the cutoff and then switches directly to zero; $\rho = 0$ disables distillation. On `Yelp`, $\alpha = 0.3$ achieves the highest category accuracy, rating accuracy, and MAUVE among the tested settings. Both smaller and larger values yield lower scores, suggesting that moderate teacher supervision is more effective than either weak or strong distillation. On `OpenReview`, all three metrics peak at $\rho = 0.2$, which retains distillation for the first $20\%$ of training and then removes it entirely. Disabling distillation from the start or extending it further into training yields lower scores, supporting the use of moderate AR supervision early in training.

### 4.5 Extend WEAN to Other Types of Datasets (RQ4)

Table 4 shows that WEAN achieves the highest downstream utility across all three datasets and both dLLM backbones. On `CodeNet`, it improves problem-classification accuracy over the strongest baseline, SFT-DLLM, from 96.5% to 98.7% for 'LLaDA-8B-Instruct' and from 93.9% to 96.5% for 'Dream-v0-Instruct-7B'. On `PrimeVul`, WEAN achieves F1 scores of 7.93 and 12.77, exceeding the best baseline scores of 6.76 and 10.35. The improvements also extend to `DM-Math`, where solver EM increases from the best baseline scores of 7.67% and 1.42% to 9.51% and 1.72%.

With 'LLaDA-8B-Instruct' as the generator, all baselines achieve lower F1 scores on `PrimeVul` than Pre-DLLM. With 'Dream-v0-Instruct-7B', these baselines also achieve lower EM accuracy on `DM-Math`. Thus, fine-tuning or distillation alone does not consistently improve the downstream utility of synthetic data. In contrast, WEAN outperforms Pre-DLLM across all three datasets and both backbones, including the settings where the other methods degrade performance. These results show that the benefits of WEAN extend beyond natural-language tasks to synthetic data for code classification, vulnerability detection, and mathematical problem solving.

We compare the computational requirements of WEAN and baselines in Appendix D, and analyze the limitations of this work in Appendix E.

## 5 Conclusion

This paper presents WEAN, the first AR-to-dLLM distillation method for downstream fine-tuning. WEAN studies the mismatch between AR teachers and dLLM students by selectively transferring supervision at reliable token positions and gradually adjusting its strength across denoising steps and training. Our evaluation across text, code, and mathematics tasks shows that controlled AR guidance can improve the utility and generation quality of fine-tuned dLLMs while preserving their parallel generation capability. These results suggest that AR models can serve as useful auxiliary teachers for adapting pretrained dLLMs and improving fine-tuning performance.

## 6 AI USE STATEMENT

Generative AI tools are used to assist with language editing and limited code development and debugging. All research ideas, method design, implementation, experiments, analysis, and conclusions are developed and verified by the authors. All AI-assisted text and code are reviewed, tested, and revised by the authors. The AI models studied and evaluated in this work are part of the research methodology and are described in the corresponding sections.

## REFERENCES

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 17981–17993, 2021.

Thomas Bertolani, Davide Bucciarelli, Leonardo Zini, Marcella Cornia, and Lorenzo Baraldi. Diffusion language models: An experimental analysis. *arXiv preprint arXiv:2606.19475*, 2026.

Tiwei Bie, Maosong Cao, Kun Chen, Lun Du, Mingliang Gong, Zhuochen Gong, Yanmei Gu, Jiaqi Hu, Zenan Huang, Zhenzhong Lan, et al. Llada2. 0: Scaling up diffusion language models to 100b. *arXiv preprint arXiv:2512.15745*, 2025.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Shuang Cheng, Yihan Bian, Dawei Liu, Yuhua Jiang, Yihao Liu, Linfeng Zhang, Qian Yao, Zhongbo Tian, Wenhai Wang, Qipeng Guo, et al. Sdar: A synergistic diffusion-autoregression paradigm for scalable sequence generation. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 22058–22075, 2026.

Justin Deschenaux and Caglar Gulcehre. Beyond autoregression: Fast llms via self-distillation through time. In *International Conference on Learning Representations*, volume 2025, pp. 5007–5045, 2025.

Yangruibo Ding, Yanjun Fu, Omniyyah Ibrahim, Chawin Sitawarin, Xinyun Chen, Basel Alomair, David Wagner, Baishakhi Ray, and Yizheng Chen. Vulnerability detection with code language models: How far are we? *arXiv preprint arXiv:2403.18624*, 2024.

Zhangyin Feng, Daya Guo, Duyu Tang, Nan Duan, Xiaocheng Feng, Ming Gong, Linjun Shou, Bing Qin, Ting Liu, Daxin Jiang, et al. Codebert: A pre-trained model for programming and natural languages. In *Findings of the association for computational linguistics: EMNLP 2020*, pp. 1536–1547, 2020.

Feiyang Fu, Tongxian Guo, and Zhaoqiang Liu. Learnable sampler distillation for discrete diffusion models. *Advances in Neural Information Processing Systems*, 38:70161–70190, 2026.

Shansan Gong, Shivam Agarwal, Yizhe Zhang, Jiacheng Ye, Lin Zheng, Mukai Li, Chenxin An, Peilin Zhao, Wei Bi, Jiawei Han, et al. Scaling diffusion language models via adaptation from autoregressive models. In *International Conference on Learning Representations*, volume 2025, pp. 5046–5073, 2025.

Google DeepMind. Gemini diffusion. 2025. Accessed 2026-07-25.

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

Urvashi Khandelwal, He He, Peng Qi, and Dan Jurafsky. Sharp nearby, fuzzy far away: How neural language models use context. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 284–294, 2018.

Samar Khanna, Siddhant Kharbanda, Shufan Li, Harshit Varma, Eric Wang, Sawyer Birnbaum, Ziyang Luo, Yanis Miraoui, Akash Palrecha, Stefano Ermon, et al. Mercury: Ultra-fast language models based on diffusion. *arXiv e-prints*, pp. arXiv–2506, 2025.

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.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pp. 19274–19286. PMLR, 2023.

David Li, Nikita Gushchin, Dmitry Abulkhanov, Eric Moulines, Ivan Oseledets, Maxim Panov, and Alexander Korotin. Idlm: Inverse-distilled diffusion language models. In *Forty-third International Conference on Machine Learning*, 2026.

Enshu Liu, Xuefei Ning, Yu Wang, and Zinan Lin. Distilled decoding 1: One-step sampling of image auto-regressive models with flow matching. In *International Conference on Learning Representations*, volume 2025, pp. 42150–42173, 2025.

Enshu Liu, Qian Chen, Xuefei Ning, Shengen Yan, Guohao Dai, Zinan Lin, and Yu Wang. Distilled decoding 2: One-step sampling of image auto-regressive models with conditional score distillation. *Advances in Neural Information Processing Systems*, 38:5448–5476, 2026a.

Enshu Liu, Xuefei Ning, Yu Wang, and Zinan Lin. Ni sampling: Accelerating discrete diffusion sampling by token order optimization. *arXiv preprint arXiv:2604.18471*, 2026b.

Wei Liu and Nai Ding. Information integration in large language models is gated by linguistic structural markers. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, 2025.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

David Lopez-Paz and Maxime Oquab. Revisiting classifier two-sample tests. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=SJkXfE5xx.

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. *Advances in Neural Information Processing Systems*, 38:50608–50646, 2026.

Shubham Parashar, Atharv Chagi, Jacob Helwig, Lakshmi Jotsna, Sushil Vemuri, James Caverlee, Dileep Kalathil, and Shuiwang Ji. Learnability-informed fine-tuning of diffusion language models. *arXiv preprint arXiv:2605.22939*, 2026.

Krishna Pillutla, Swabha Swayamdipta, Rowan Zellers, John Thickstun, Sean Welleck, Zaid Harchaoui, et al. Mauve: Measuring the gap between neural text and human text using divergence frontiers. *Advances in neural information processing systems*, 2022.

Ruchir Puri, David S Kung, Geert Janssen, Wei Zhang, Giacomo Domeniconi, Vladimir Zolotov, Julian Dolby, Jie Chen, Mihir Choudhury, Lindsey Decker, et al. Codenet: A large-scale ai for code dataset for learning a diversity of coding tasks. *arXiv preprint arXiv:2105.12655*, 2021.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

Phillip Rust, Jonas Pfeiffer, Ivan Vulic, Sebastian Ruder, and Iryna Gurevych. How good is your tokenizer? on the monolingual performance of multilingual language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021*, pp. 3118–3135, 2021.

Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Mariano Marroquin, Justin T Chiu, Alexander M Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

David Saxton, Edward Grefenstette, Felix Hill, and Pushmeet Kohli. Analysing mathematical reasoning abilities of neural models. *arXiv preprint arXiv:1904.01557*, 2019.

Yuxuan Song, Zheng Zhang, Cheng Luo, Pengyang Gao, Fan Xia, Hao Luo, Zheng Li, Yuehang Yang, Hongli Yu, Xingwei Qu, et al. Seed diffusion: A large-scale diffusion language model with high-speed inference. *arXiv preprint arXiv:2508.02193*, 2025.

Xingyu Su, Jacob Helwig, Shubham Parashar, Atharv Chagi, Lakshmi Jotsna, Degui Zhi, James Caverlee, Dileep Kalathil, and Shuiwang Ji. Data-efficient autoregressive-to-diffusion language models via on-policy distillation. *arXiv preprint arXiv:2606.06712*, 2026.

Jaesung Tae, Hamish Ivison, Sachin Kumar, and Arman Cohan. Tess 2: A large-scale generalist diffusion language model. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 21171–21188, 2025.

DiffusionGemma Team, Adrien Ali Taïga, James Assiene, Daniele Calandriello, Rahma Chaabouni, João Gante, Tamara von Glehn, Nate Keating, Chris Knutsen, Martin Kukla, et al. Diffusiongemma technical report. *arXiv preprint arXiv:2608.00146*, 2026.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Shuaidi Wang, Zhan Zhuang, Ruping Huang, and Yu Zhang. Nara: Noise-aware lora for parameter-efficient fine-tuning of diffusion llms. *arXiv preprint arXiv:2605.29716*, 2026.

Chulin Xie, Zinan Lin, Arturs Backurs, Sivakanth Gopi, Da Yu, Huseyin A Inan, Harsha Nori, Haotian Jiang, Huishuai Zhang, Yin Tat Lee, et al. Differentially private synthetic data via foundation model apis 2: Text. *arXiv preprint arXiv:2403.01749*, 2024.

Guowei Xu, Wenxin Xu, Jiawang Zhao, and Kaisheng Ma. Gift: Guided importance-aware fine-tuning for diffusion language models. *arXiv preprint arXiv:2509.20863*, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Dongjie Yang, Zhuosheng Zhang, and Hai Zhao. Learning better masking for better language model pre-training. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pp. 7255–7267, 2023.

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.

Da Yu, Arturs Backurs, Sivakanth Gopi, Huseyin Inan, Janardhan Kulkarni, Zinan Lin, Chulin Xie, Huishuai Zhang, and Wanrong Zhang. Training private and efficient language models with synthetic data from llms. In *Socially Responsible Language Modelling Research*, 2023.

Yifan Yu, Yuqing Jian, Junxiong Wang, Zhongzhu Zhou, Donglin Zhuang, Xinyu Fang, Sri Yanamandra, Xiaoxia Wu, Qingyang Wu, Shuaiwen Leon Song, et al. Introspective diffusion language models. *arXiv preprint arXiv:2604.11035*, 2026.

Chi Zhang, Xixi Hu, Bo Liu, and Qiang Liu. Gumbel distillation for parallel text generation. *arXiv preprint arXiv:2603.22216*, 2026a.

Gongbo Zhang, Wen Wang, Ye Tian, and Li Yuan. Turning the tide: Cross-architecture distillation for diffusion large language models. *arXiv preprint arXiv:2604.26951*, 2026b.

Ruixiang Zhang, Shuangfei Zhai, Yizhe Zhang, James Thornton, Zijing Ou, Joshua M Susskind, and Navdeep Jaitly. Target concrete score matching: A holistic framework for discrete diffusion. In *International Conference on Machine Learning*, pp. 76716–76753. PMLR, 2025.

Zhanhui Zhou, Lingjie Chen, Hanghang Tong, and Dawn Song. dllm: Simple diffusion language modeling. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pp. 78–88, 2026.

# A Related Works

**Fine-Tuning dLLMs.** Diffusion-based large language models (dLLMs) have recently emerged as a new paradigm for text generation (Austin et al., 2021; Sahoo et al., 2024; Nie et al., 2026; Ye et al., 2025). Recent work improves dLLM fine-tuning by modifying its training objective. GIFT gives more weight to tokens that are more important for learning (Xu et al., 2025), LIFT adjusts which tokens receive more attention at different diffusion timesteps (Parashar et al., 2026), and NaRA adapts LoRA to the denoising structure of dLLMs (Wang et al., 2026). Our work introduces distillation to bring this AR advantage into dLLM fine-tuning.

**Adapting AR Models into dLLMs** Rather than pretraining from scratch, a growing line converts AR checkpoints into dLLMs (Gong et al., 2025; Ye et al., 2025; Tae et al., 2025). Gong et al. (2025) introduced a method that starts from an existing AR model and gradually trains it to recover masked tokens from context, rather than only predicting the next token. Several dLLMs are built on this approach. For example, 'Dream 7B' is initialized from 'Qwen2.5' (Ye et al., 2025), and 'TESS 2' is adapted from 'Mistral-7B' (Tae et al., 2025). More recent work converts pretrained AR models into dLLMs at scale (Bie et al., 2025; Cheng et al., 2026; Yu et al., 2026). The LLaDA family also used this approach when scaling to 100B parameters (Bie et al., 2025). These results suggest that starting from an AR model is an effective way to build large dLLMs. Previous methods use knowledge from AR models when building or adapting the dLLM. In contrast, we keep the dLLM unchanged and use a frozen AR model to guide its fine-tuning.

**Distillation for dLLMs.** Distillation trains a student model on the output distributions of a teacher model (Hinton et al., 2015). TCSM (Zhang et al., 2025) uses the predictions of a pretrained AR model as training targets to train a new dLLM from scratch. Gumbel Distillation (Zhang et al., 2026a) trains dLLM students to imitate the outputs of an AR teacher while learning the relationships among tokens. OPDLM (Su et al., 2026) follows a similar idea, using a frozen AR model to guide the diffusion model throughout its denoising process, helping retain the knowledge learned by the AR model. None of these methods uses an independent AR teacher to adapt an already pretrained dLLM. A separate line distills dLLMs into faster or smaller one (Deschenaux & Gulcehre, 2025; Li et al., 2026). TIDE distills dLLM teachers into a small one, and adjusts the distillation weight with both training progress and diffusion timestep (Zhang et al., 2026b). We use supervision from a frozen AR teacher as an auxiliary objective for adapting a dLLM, helping the student acquire task-specific knowledge more effectively. As the teacher guidance is gradually removed, the resulting model remains a task-adapted dLLM that can perform downstream generation.

# B Experimental Setup Details

## B.1 LLM Base Models and Benchmarks.

We use 'LLaDA-8B-Instruct' (Nie et al., 2026) and 'Dream-v0-Instruct-7B' (Ye et al., 2025) as the pretrained dLLM generator backbones. For methods involving AR supervision, we use 'Qwen3-32B' (Yang et al., 2025) as the teacher model. We evaluate on six benchmarks spanning text, code, and mathematics. For text generation, we use three benchmarks commonly adopted in synthetic-text evaluation (Yang et al., 2025; Xie et al., 2024; Yu et al., 2023). The `Yelp` dataset (Yang et al., 2025) contains 20,000 reviews. Each review is associated with a 10-class business-category label (Category) and a 5-class rating label (Rating). The `OpenReview` dataset (Xie et al., 2024) contains ICLR paper reviews, each annotated with a 12-class paper-area label (Area) and a 5-class recommendation label (Rating). The `PubMed` dataset (Yu et al., 2023) contains 5,000 training abstracts without downstream labels and is therefore used for unconditional text generation.

For code generation, we use `CodeNet` (Puri et al., 2021) and `PrimeVul` (Ding et al., 2024). For `CodeNet`, the generator is conditioned on a programming problem description to produce solution code. Each generated solution inherits the identifier of its conditioning problem. We train a code classifier only on the generated solutions and evaluate its problem-classification accuracy on held-out real solutions. For `PrimeVul`, we construct a conditional generation task using the vulnerable and non-vulnerable labels. We train a fixed binary vulnerability detector only on the generated functions and evaluate it on the real chronological test split using F1. For mathematical generation, we use the DeepMind Mathematics Dataset (`DM-Math`) (Saxton et al., 2019), which programmatically generates mathematical question–answer pairs across arithmetic, algebra, calculus, probability, and related categories. Each dLLM generates synthetic question–answer pairs from the target training distribution. A solver is then trained only on these generated pairs and evaluated using exact match on an independently generated reference test set.

### B.2 Baselines

We compare WEAN with the following baselines. ❶ Pre-dLLM: use the pretrained dLLM directly, without task-specific fine-tuning, to generate downstream training data. ❷ SFT-dLLM: fine-tune the dLLM on the original downstream-task corpus using the standard dLLM objective, without distillation. ❸ GIFT (Xu et al., 2025): fine-tune the dLLM using entropy-based token-importance weighting, without teacher supervision. ❹ LIFT (Parashar et al., 2026): select which masked tokens to supervise according to token learnability and the diffusion timestep. ❺ NaRA (Wang et al., 2026): replace standard noise-agnostic LoRA with noise-aware low-rank adaptation whose updates are conditioned on the diffusion timestep.

### B.3 Synthetic-Data Evaluation.

For every generator-based method, including Pre-dLLM, we generate the same number of synthetic examples using identical conditioning inputs, label allocations, and decoding settings. Unless otherwise specified, each method generates 5,000 examples per benchmark.

For `Yelp` and `OpenReview`, we train a RoBERTa-base classifier (Liu et al., 2019) on the generated data for 10 epochs and evaluate it on the real test set. We report classification accuracy for both L1 and L2. The classifier input is truncated to 512 tokens. For `CodeNet` and `PrimeVul`, we train a fixed CodeBERT-base classifier (Feng et al., 2020) on the generated code, using the same model architecture and optimization settings across all generation methods. We report problem-classification accuracy on `CodeNet` and positive-class F1 on `PrimeVul`. For `DM-Math`, we train a fixed T5-base solver (Raffel et al., 2020) only on the generated question–answer pairs and report normalized answer exact match on the independently generated reference test set.

For the three text-generation benchmarks, we additionally evaluate intrinsic generation quality using: ❶ *C2ST-Sim* (Lopez-Paz & Oquab, 2017), which measures how difficult it is for a classifier to distinguish generated text from real text; higher scores indicate greater similarity between the two distributions; and ❷ *MAUVE* (Pillutla et al., 2022), computed between 1,000 generated samples and 1,000 real test samples using GPT-2-large features. We do not apply these text-based quality metrics to the code and mathematics benchmarks.

## C Implementation Details

This section introduces the motivation, experiment details, details of tokenizer alignment, and an overview of our method WEAN.

### C.1 Motivation Experiment Details

**AR and dLLM comparison.** For the left panels of Figure 2, we sample 256 held-out `Yelp` responses with a maximum sequence length of 512 tokens. We compare a task-fine-tuned `Qwen3-8B` AR reference with a vanilla-SFT `LLaDA-8B-Instruct` dLLM. The main experiments use `Qwen3-32B` as the teacher instead. We keep the prompt and the first 32 response tokens visible and vary the mask ratio over the remaining response tokens from 20% to 90%. Positions masked

at a lower ratio remain masked at higher ratios. At each ratio, we compare the two models on all currently masked structural tokens. We project the AR predictions into the dLLM vocabulary and retain a normalized top-32 distribution. We measure probability overlap at temperature 1 and forward KL divergence at temperature 4. The dLLM distribution remains normalized over its full vocabulary.

**Changes after SFT.** For the right panels, we compare pretrained `LLaDA-8B-Instruct` with its vanilla-SFT counterpart using the same responses and masking procedure. We evaluate both models on identical inputs and keep the evaluated token positions fixed to those selected at 20% masking. We report results separately for structural tokens, semantic tokens, and all tokens. We measure pretrained-to-SFT KL divergence over the full vocabulary at temperature 1. We also measure the reduction in negative log-likelihood of tokens in the real text as $\Delta\mathrm{NLL} = \mathrm{NLL}_{\mathrm{pre}} - \mathrm{NLL}_{\mathrm{SFT}}$. A positive value indicates improved prediction of these tokens after SFT.

**Averaging and confidence intervals.** We first average token-level measurements within each response and then average across responses with equal weight. Shaded bands show 95% confidence intervals obtained by bootstrap resampling of responses.

### C.2 Tokenizer Alignment

Because the AR teacher and dLLM student may use different tokenizers, the same response can be segmented into different token sequences, and their predictions are defined over different vocabularies. We align both vocabulary entries and response-token positions before distillation. Vocabulary alignment determines which dLLM token receives the probability assigned to an AR token, while position alignment determines which teacher prediction supervises a given student position.

For vocabulary alignment, let $d_{\mathrm{AR}}(a)$ denote the text decoded from an AR token $a \in \mathcal{V}_{\mathrm{AR}}$, and let $\tau_d(\cdot)$ denote the dLLM tokenizer. We decode each AR token and re-tokenize its text using $\tau_d$, defining the partial mapping,

$$m(a) = v \quad \text{if} \quad \tau_d\big(d_{\mathrm{AR}}(a)\big) = [v], \tag{4}$$

where $v \in \mathcal{V}_d$. This mapping retains only AR tokens that correspond to exactly one dLLM token. If re-tokenization produces zero or multiple tokens, $m(a)$ is undefined, and the corresponding AR candidate is excluded from the projection. This restriction allows each retained teacher candidate to supervise a single student-token prediction. Multiple AR tokens may map to the same dLLM token, in which case their probabilities are added together.

For position alignment, we use the character offsets of tokens in the original response rather than their token indices, since the two tokenizers may produce sequences of different lengths. Let $\pi(i)$ denote the AR position aligned with dLLM position $i$. The teacher prediction used at student position $i$ is then conditioned on the AR-tokenized prefix $x^{\mathrm{AR}}_{<\pi(i)}$.

At each aligned position, let $\mathcal{C}_i$ contain the top-$M$ candidates from the AR teacher distribution. We apply $m$ to these candidates, discard those without a valid mapping, and aggregate the remaining probability mass by dLLM token. When the retained mass is positive, the projected distribution is,

$$\tilde{p}_{\phi^*}(v \mid x_{<i}) = \frac{\sum_{a \in \mathcal{C}_i:\, m(a)=v} p_{\phi^*}(a \mid x^{\mathrm{AR}}_{<\pi(i)})}{\sum_{a \in \mathcal{C}_i:\, m(a)\ \text{defined}} p_{\phi^*}(a \mid x^{\mathrm{AR}}_{<\pi(i)})}. \tag{5}$$

The numerator collects the probability mass assigned to student token $v$, and the denominator normalizes over all retained candidates. Student tokens that receive no mapped probability mass have zero probability in this distribution. Thus, $\tilde{p}_{\phi^*}$ provides a normalized approximation to the teacher prediction in the dLLM vocabulary space.

Finally, we retain the top-$K$ entries of the projected distribution and renormalize their probabilities to sum to one, assigning zero probability to all other tokens. We cache the retained token identifiers and normalized probabilities for use as teacher targets in the KL distillation loss. Here, $M$ controls the number of AR candidates considered before projection, while $K$ controls the number of dLLM tokens retained after projection and aggregation. These cached targets are reused throughout student fine-tuning without additional teacher evaluations.

**Algorithm 1** The workflow of WEAN.

**Input:** Training set $\mathcal{D}$; pretrained AR teacher $p_{\phi_0}$ and dLLM student $p_{\theta_0}$; distillation strength $\alpha$; cutoff fraction $\rho \in [0, 1]$; total updates $S$; candidate and cache sizes $M, K$.
**Output:** Fine-tuned dLLM $p_{\theta_S}$.

`Stage 1: Prepare the teacher and cache supervision`

Fine-tune $p_{\phi_0}$ on $\mathcal{D}$ to obtain $p_{\phi^\star}$, then freeze its parameters
**for** $x \in \mathcal{D}$ **do**
  Identify structural-token positions $\mathcal{S}(x)$ in the response
  **for** $i \in \mathcal{S}(x)$ **do**
    Align student position $i$ with AR position $\pi(i)$ using character offsets
    Obtain the top-$M$ candidates from $p_{\phi^\star}(\cdot \mid x^{\mathrm{AR}}_{<\pi(i)})$
    Map candidates into the dLLM vocabulary, retaining only single-token mappings
    Sum probabilities mapped to the same dLLM token and normalize
    Retain the top-$K$ entries, renormalize, and cache as $q_{x,i}$
  **end for**
**end for**

`Stage 2: Fine-tune the student with cached teacher targets`

**for** $s = 0, \ldots, S-1$ **do**
  Sample a training sequence $x \sim \mathcal{D}$ and timestep $t \sim p(t)$
  Mask response positions $\mathcal{M}_t$ to obtain $x_t$, keeping the prompt visible
  Compute $\mathcal{L}_{\mathrm{SFT}}$ on $\mathcal{M}_t$ using the original tokens as targets
  $\mathcal{I}_t \leftarrow \mathcal{M}_t \cap \mathcal{S}(x)$ ▷ Masked structural positions
  Compute the distillation loss from cached teacher targets:
  $$\mathcal{L}_{\mathrm{KD}} \leftarrow \sum_{i \in \mathcal{I}_t} D_{\mathrm{KL}}(q_{x,i} \,\|\, p_{\theta_s}(x_i = \cdot \mid x_t, t))$$
  Compute the training-progress weight $\lambda_s(s)$ using Equation (2)
  $\lambda(s,t) \leftarrow \alpha(1-t)\lambda_s(s)$ ▷ Adaptive teacher weight
  Combine the objectives: $\mathcal{L} \leftarrow \mathcal{L}_{\mathrm{SFT}} + \lambda(s,t)\mathcal{L}_{\mathrm{KD}}$
  Take one optimizer step on $\mathcal{L}$ to update $\theta_s$ to $\theta_{s+1}$
**end for**
**return** Fine-tuned dLLM $p_{\theta_S}$

Table 5: Computational cost of WEAN and the baselines. 'GPU memory' means the peak GPU memory. '$--$' indicates that generation throughput is not applicable to training stages.

| Method | Stage | GPU Memory | Time | Tokens/s |
|---|---|---|---|---|
| Pre-dLLM | dLLM generation | 24.0GB | 7.0h | 40.0 |
| SFT-dLLM | dLLM fine-tuning | 43.0GB | 5.2h | – |
| GIFT | dLLM fine-tuning | 43.8GB | 5.3h | – |
| LIFT | dLLM fine-tuning | 44.1GB | 5.2h | – |
| NaRA | dLLM fine-tuning | 44.5GB | 4.0h | – |
| SFT/GIFT/LIFT/NaRA | dLLM generation | 24.5GB | 7.0h | 40.0 |
| WEAN | AR-teacher fine-tuning | 77.5GB | 18.0h | – |
| | Teacher-cache construction | 69.0GB | 3.5h | – |
| | dLLM fine-tuning | 45.0GB | 5.8h | – |
| | dLLM generation | 24.5GB | 7.0h | 40.0 |
| Qwen3-32B | AR generation | 69.0GB | 14.7h | 19.0 |

### C.3 Overflow of WEAN

Algorithm 1 summarizes the WEAN. During offline teacher preparation, we fine-tune the pretrained AR teacher on the target dataset $\mathcal{D}$ to obtain $p_{\phi^\star}$ and then freeze its parameters. For each training sequence $x$, we identify the structural-token positions $\mathcal{S}(x)$ within the response using the categories DET, AUX, ADP, PUNCT, CCONJ, SCONJ, and PART. We align each student position $i \in \mathcal{S}(x)$ with an AR prediction position $\pi(i)$ using character offsets in the response text. At each aligned position, we obtain the top-$M$ teacher candidates and map them into the dLLM vocabulary, retaining only candidates that correspond to a single dLLM token. Probabilities mapped to the same student

token are summed and normalized. We then retain the top-$K$ entries, renormalize their probabilities to sum to one, and cache the resulting distribution as $q_{x,i}$, with zero probability assigned to all other tokens. These cached targets eliminate the need to run the teacher during student fine-tuning.

During student fine-tuning, each update $s$ samples a training sequence $x$ and a diffusion timestep $t$. We mask response positions $\mathcal{M}_t$ to obtain $x_t$ while keeping the prompt visible. The standard supervised denoising loss $\mathcal{L}_{\mathrm{SFT}}$ uses the original tokens in $x$ as targets at all masked response positions. Teacher supervision applies only to masked structural positions, $\mathcal{I}_t = \mathcal{M}_t \cap \mathcal{S}(x)$. The distillation loss $\mathcal{L}_{\mathrm{KD}}$ sums the forward KL divergence from each cached teacher distribution $q_{x,i}$ to the corresponding student prediction; it is zero when $\mathcal{I}_t$ is empty. We compute the training-progress weight $\lambda_s(s)$ using Equation (2), where the cutoff fraction $\rho$ determines when distillation ends. The overall weight $\lambda(s,t) = \alpha(1-t)\lambda_s(s)$ combines the distillation strength, mask-dependent weighting, and training-progress schedule. Finally, we take one optimizer step on $\mathcal{L}_{\mathrm{SFT}} + \lambda(s,t)\mathcal{L}_{\mathrm{KD}}$ to update $\theta_s$ to $\theta_{s+1}$. Only the student parameters are updated, and the procedure returns $p_{\theta_S}$ after $S$ updates.

## D Computational Resource Requirements

Table 5 summarizes the representative computational cost of WEAN and the baselines on `OpenReview`. We report peak memory per GPU and wall-clock time for a representative `Yelp` experiment with 'LLaDA-8B-Instruct', using four NVIDIA A100 80GB GPUs, 20 training epochs, an effective batch size of 4,096, and a maximum sequence length of 512. Generation time is estimated for synthesizing 5,000 examples. The reported generation throughput is the aggregate number of non-padding response tokens produced per second across the four GPUs. Model loading and file I/O are excluded. The values are representative estimates because the exact cost varies with response length, denoising steps, and system utilization.

The additional cost of WEAN is confined to training. The AR teacher is fine-tuned and queried offline, after which only sparse projected distributions are stored. Consequently, the teacher and student never need to reside in GPU memory simultaneously. The peak memory of the dLLM fine-tuning stage is therefore close to that of the fine-tuning baselines, although WEAN incurs the one-time costs of adapting the teacher and constructing the cache.

At response generation time, WEAN discards both the AR teacher and the cached distributions. Its model architecture, memory footprint, and decoding schedule are identical to those of the corresponding fine-tuned dLLM. Thus, all dLLM fine-tuning methods have comparable inference cost. In the representative setting, the dLLM produces approximately 40.0 tokens/s, compared with 19.0 tokens/s for the autoregressive 'Qwen3-32B' teacher, corresponding to an approximately $2.1\times$ throughput improvement. This advantage follows from predicting multiple tokens in parallel at each denoising step, whereas the AR teacher must decode one token at a time.

## E Limitation

This section concludes the limitations of WEAN. First, the effectiveness of AR distillation depends on the compatibility between the teacher and student models. When their representations, tokenization schemes, or conditioning contexts differ substantially, transferring AR knowledge may become less reliable. Second, although dLLMs can offer faster generation, incorporating teacher guidance introduces additional computational and storage costs during training, particularly for generating and storing supervision from large teacher models. Future work should investigate the applicability of AR distillation to broader model families, domains, and more complex generation scenarios.