A New Language Modeling Paradigm: Diffusion Large Language Models

Fuliang Liu

Nanjing University

School of Computer Science

fuliangliu@smail.nju.edu.cn

Abstract: Autoregressive (AR) modeling dominates modern large language models (LLMs), but its left-to-right decoding is inherently sequential and restricts parallelism. Diffusion-based generative modeling provides an alternative: it generates by iteratively denoising from noise, offering bidirectional context and parallel token updates. This report surveys diffusion large language models (DLLMs) with an emphasis on the *core insights* that make diffusion training scalable and discrete diffusion feasible. We first revisit score-based diffusion theory and explain why diffusion learns *noise-conditional scores* rather than the raw data score, which turns score learning into a simple supervised regression. We then focus on discrete masked diffusion for language, highlighting SEDD and its key obstacle: timestep-dependent ratio estimation. We explain how RADD reveals an analytic timestep factorization that removes the need for a time-conditioned Transformer, simplifying the objective into an AR-like denoising cross-entropy. Next, we summarize SMDM's scaling-law perspective on masked diffusion LMs. Finally, we discuss systems and post-training conversion: why full-attention DLLMs cannot directly use KV cache, how Fast-dLLM enables KV cache and parallel decoding, and how recent *AR→Block-Diffusion* conversion (Block Diffusion, SDAR, Fast-dLLM v2) reuses pretrained AR capabilities while enabling blockwise parallel generation with efficient attention masks and AR-aligned prediction pathways.

CONTENTS

1 INTRODUCTION

Autoregressive (AR) language models factorize the joint probability of a token sequence into a product of next-token conditionals. This yields exact likelihood training and strong performance, but inference remains fundamentally sequential: tokens must be generated one-by-one, limiting decoding throughput.

Diffusion models represent a different paradigm. Instead of predicting tokens strictly left-to-right, diffusion models define a forward corruption process that gradually destroys structure, and learn a reverse process that iteratively denoises. When adapted to language, this yields diffusion large language models (DLLMs), which update many positions in parallel and can exploit bidirectional context during generation.

This report is organized around a central question: *what are the key ideas that make diffusion language modeling both theoretically principled and practically scalable*? We therefore emphasize (i) the "noise-conditioning" insight that simplifies score learning in continuous diffusion, (ii) the discrete analogue of score learning via ratio/conditional estimation (SEDD), (iii) the major simplification brought by RADD (analytic timestep separation, removing time-conditioning), (iv) the scaling-law view of masked diffusion LMs (SMDM), and (v) system / post-training conversion techniques that make diffusion-style decoding deployable (Fast-dLLM, Block-Diffusion conversion such as SDAR and Fast-dLLM v2).

2 DIFFUSION MODELS AND DEEP GENERATIVE MODELING

This section presents the formula-centric backbone of diffusion as score-based modeling and explains a key insight often missed in "formula-only" expositions: **we do not directly train a network to predict the raw data score $\nabla_x \log p_{\textbf{data}}(x)$, but instead predict a *noise-conditional* score.** This design is what makes diffusion training scalable.

2.1 SCORE-BASED MODELING AND THE DATA SCORE

Let $p_{\text{data}}(\mathbf{x})$ be the data distribution on $\mathbf{x} \in \mathbb{R}^d$. Score-based generative modeling aims to learn the score

$$\mathbf{s}^*(\mathbf{x}) \triangleq \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x}). \tag{1}$$

If $\mathbf{s}^*$ were known, one could sample from p_{data} using Langevin dynamics / SDE-based sampling Song & Ermon (2019); Song et al. (2021).

Why not learn $\nabla_x \log p_{\textbf{data}}(x)$ directly? In high dimensions, p_{data} often concentrates near a complicated low-dimensional manifold. The raw score can be ill-behaved (high curvature, unstable gradients) and, crucially, it is not paired with a clean supervised "target" for training. This is where diffusion introduces the key trick: **smooth the data distribution with noise, and learn the score of the smoothed distribution instead.**

2.2 NOISE-CONDITIONAL SCORE: THE SCALABILITY INSIGHT

Define a Gaussian corruption (a.k.a. diffusion "forward noising"):

$$\mathbf{x}_\sigma = \mathbf{x}_0 + \sigma\boldsymbol{\epsilon}, \qquad \mathbf{x}_0 \sim p_{\text{data}},\ \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{2}$$

Let $p_\sigma(\mathbf{x})$ be the marginal distribution of $\mathbf{x}_\sigma$. Diffusion trains a network $\mathbf{s}_\theta(\mathbf{x}, \sigma)$ to approximate the *noise-conditional score*

$$\mathbf{s}_\sigma(\mathbf{x}) \triangleq \nabla_{\mathbf{x}} \log p_\sigma(\mathbf{x}). \tag{3}$$

Key benefit: supervised target becomes available. Under Gaussian corruption, the score matching objective admits a tractable form known as *denoising score matching (DSM)* Hyvärinen (2005); Song & Ermon (2019):

$$\mathcal{L}_{\text{DSM}}(\theta) = \mathbb{E}_{\mathbf{x}_0, \boldsymbol{\epsilon}} \left[\left\| \mathbf{s}_\theta(\mathbf{x}_0 + \sigma\boldsymbol{\epsilon}, \sigma) + \frac{\boldsymbol{\epsilon}}{\sigma} \right\|_2^2 \right]. \tag{4}$$

This reveals the core "insight":

- we **do not need** $\nabla_x \log p_{\text{data}}(x)$ as a label;
- after adding noise, the optimal score estimator has a simple regression target $-\epsilon/\sigma$;
- training reduces to standard supervised learning on synthetic noise.

This is precisely why diffusion training scales: the network learns a family of denoisers/scores across noise levels using easy-to-sample corruption.

Connection to "predicting noise" in DDPMs. DDPMs Ho et al. (2020) further parameterize the reverse process by predicting ϵ (or an equivalent reparameterization), leading to the well-known objective

$$\mathcal{L}_{\text{DDPM}}(\theta) = \mathbb{E}_{t,\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\|_2^2\right], \tag{5}$$

which is exactly the "noise regression" view implied by Eq. (4). In short: **noise-conditioning transforms an intractable score-learning problem into a scalable denoising regression problem.**

2.3 Sampling: From High Noise to Data

Once $\mathbf{s}_\theta$ (or ϵ_θ) is trained, sampling proceeds from noise to data using annealed Langevin dynamics / reverse diffusion Song & Ermon (2019); Song et al. (2021); Ho et al. (2020). This continuous backbone motivates discrete DLLMs: replace Gaussian noise with token masking, and replace continuous scores with discrete analogues (ratios / conditional token distributions).

3 Diffusion for Discrete Language Modeling

For language, tokens are categorical so gradients w.r.t. tokens do not exist. Discrete diffusion therefore replaces "add Gaussian noise" with a token-space corruption process (typically masking), and replaces score estimation with learning discrete objects such as probability ratios (SEDD) or conditional clean-token distributions (RADD/SMDM/LLaDA).

Autoregressive models factorize

$$p_{\text{AR}}(x_{1:L}) = \prod_{i=1}^{L} p(x_i \mid x_{<i}), \tag{6}$$

while masked diffusion LMs model generation as iterative reconstruction from corrupted sequences, enabling parallel updates and bidirectional context Lou et al. (2024); Ou et al. (2025); Nie et al. (2025b;a).

4 SEDD: Score-Entropy Discrete Diffusion (Corrected)

SEDD Lou et al. (2024) builds discrete diffusion around an *absorbing* forward process: tokens are replaced by a special mask symbol $[\mathrm{M}]$ and remain masked thereafter. Let $x_0 \in \mathcal{V}^L$ be a clean sequence. At time t, each position is independently masked with probability $1 - \alpha_t$:

$$q(x_t \mid x_0) = \prod_{i=1}^{L}\Big[\alpha_t\,\mathbf{1}(x_t^i = x_0^i) + (1-\alpha_t)\,\mathbf{1}(x_t^i = [\mathrm{M}])\Big]. \tag{7}$$

4.1 What is the "score" in absorbing discrete diffusion? (Ratio view)

The main obstacle in discrete diffusion is that $\nabla_x \log p(x)$ is undefined for categorical x. SEDD replaces "gradient score" with a **ratio-based object** called the *concrete score*: compare the probability of two *neighboring transitive states* that differ by changing one masked position into a concrete token. Concretely, if $x_t^i = [\mathrm{M}]$, define $\hat{x}_t^{(i\leftarrow v)}$ as the same sequence except position i is set to $v \in \mathcal{V}$. The concrete score is a ratio of marginals at time t:

$$r_t(i, v \mid x_t) \;\propto\; \frac{p_t(\hat{x}_t^{(i\leftarrow v)})}{p_t(x_t)}. \tag{8}$$

Intuitively, $r_t(i, v \mid x_t)$ measures how much more (or less) likely the sequence would be if the masked slot were concretized as v. This ratio plays the role of a discrete "direction" for denoising, analogous to how continuous scores guide movement from noise to data.

4.2 Score Entropy: why SEDD can learn ratios without modeling p_t

SEDD introduces **score entropy**, a loss that is the discrete analogue of score matching: instead of matching $\nabla_x \log p_\sigma(x)$, it matches the ratio structure needed by the reverse process Lou et al. (2024). Operationally, SEDD trains a Transformer to output logits over vocabulary for each masked position, and uses these logits to parameterize the ratio estimator needed by the reverse kernel.

A practical implication (and also a pain point emphasized by later work) is:

- SEDD's ratio estimation is inherently **timestep-dependent**: the optimal ratio differs across t because the marginal p_t changes with masking severity.
- Therefore, SEDD-style implementations typically require a **time-conditioned** network (explicit t embeddings) to represent $r_t(\cdot)$ well.

This timestep dependence is exactly what RADD will simplify.

4.3 SEDD training objective: Score Entropy and Denoising Score Entropy

SEDD's central contribution is to replace gradient-based score matching (undefined for categorical tokens) with a **ratio-based** objective called *score entropy* Lou et al. (2024). Let p be a discrete distribution on states (e.g., a corrupted token sequence state in a diffusion chain). SEDD introduces a *score network* $s_\theta(x)_y > 0$ that assigns a positive "score" to transitioning from state x to a different state y, and defines the **score entropy loss**:

$$\mathcal{L}_{\mathrm{SE}} = \mathbb{E}_{x\sim p}\left[\sum_{y\neq x} w_{xy}\left(s_\theta(x)_y - \frac{p(y)}{p(x)}\log s_\theta(x)_y + K\left(\frac{p(y)}{p(x)}\right)\right)\right], \tag{9}$$

where $w_{xy} \geq 0$ are user-chosen weights and $K(a) = a(\log a - 1)$ is a normalizing constant ensuring $\mathcal{L}_{\mathrm{SE}} \geq 0$ Lou et al. (2024). Intuitively, $\frac{p(y)}{p(x)}$ is exactly the **ratio signal** that we want the model to capture, and the optimal solution satisfies $s_{\theta^*}(x)_y \propto \frac{p(y)}{p(x)}$ (up to the weighting scheme).

Why Eq. (9) is not directly scalable. The expression contains the unknown ratio $\frac{p(y)}{p(x)}$, where p is the (time-dependent) diffusion marginal. In large state spaces (language), directly estimating or summing over such ratios is intractable, which motivates a denoising-style reformulation that only requires sampling from a tractable corruption kernel.

Denoising Score Entropy (scalable form). SEDD considers a perturbation view where the diffusion marginal p is generated by perturbing a base density p_0 through a transition kernel $p(\cdot \mid x_0)$:

$$p(x) = \sum_{x_0} p(x \mid x_0)\, p_0(x_0). \tag{10}$$

Then SEDD proves that $\mathcal{L}_{\mathrm{SE}}$ is equivalent (up to a θ-independent constant) to the **denoising score entropy** objective Lou et al. (2024):

$$\mathcal{L}_{\mathrm{DSE}} = \mathbb{E}_{x_0\sim p_0}\mathbb{E}_{x\sim p(\cdot|x_0)}\left[\sum_{y\neq x} w_{xy}\left(s_\theta(x)_y - \frac{p(y \mid x_0)}{p(x \mid x_0)}\log s_\theta(x)_y\right)\right]. \tag{11}$$

The key scalability insight. Eq. (11) replaces the intractable ratio $\frac{p(y)}{p(x)}$ with a *tractable conditional ratio* $\frac{p(y|x_0)}{p(x|x_0)}$ under the known perturbation kernel. In diffusion language models, $p(\cdot \mid x_0)$ is exactly the forward corruption process (e.g., masking-based), so these conditional probabilities can be computed analytically. As a result, Monte Carlo training only needs: (i) sample $x_0 \sim p_0$ (a clean data sample), (ii) sample x from the forward corruption, and (iii) evaluate the network once to obtain all $s_\theta(x)_y$.

Time-dependent score network. Since the diffusion marginal changes with timestep, SEDD typically parameterizes a time-conditioned score network $s_\theta(x_t, t)$ and applies the denoising score entropy at each diffusion time using the corresponding forward kernel $p_{t|0}(\cdot \mid x_0)$ Lou et al. (2024). This timestep dependence is precisely what RADD will later simplify by analytically separating the t-dependence from the learned conditional prediction.

5 RADD: Reparameterizing Absorbing Discrete Diffusion (The "Simplification" Insight)

RADD Ou et al. (2025) provides the key theoretical clarification: **in absorbing diffusion, the concrete score can be factorized into (i) a time-independent conditional probability of clean data and (ii) an analytic, time-dependent scalar.**

5.1 Theorem-level message: isolate t analytically

RADD shows (informally summarized) that for masked positions, the ratio object needed by the reverse process can be expressed as

$$r_t(i, v \mid x_t) = c(t) \cdot p(x_0^i = v \mid x_t), \tag{12}$$

where $c(t)$ is an analytic scalar determined by the forward schedule, and the only hard part is the **time-independent** conditional distribution $p(x_0^i = v \mid x_t)$ Ou et al. (2025).

Why this matters. Eq. (12) means we no longer need a Transformer that takes t as an input just to represent the diffusion timestep effect. Instead:

- train a single (time-independent) denoiser $\pi_\theta(v \mid x_t)$;
- incorporate timestep effects via a known scalar $c(t)$ outside the network.

This is the core **simplification**: *RADD makes absorbing discrete diffusion look much closer to AR/MLM-style training.*

5.2 Loss simplification: from time-conditioned ratio learning to AR-like denoising CE

With timestep separated, the objective becomes essentially a masked denoising cross-entropy, potentially with an analytic weight induced by the schedule:

$$\mathcal{L}_{\text{RADD}}(\theta) = \mathbb{E}_t \, \mathbb{E}_{x_0, x_t \sim q(\cdot|x_0,t)} \left[- \sum_{i: x_t^i = [\text{M}]} w(t) \, \log \pi_\theta(x_0^i \mid x_t) \right], \tag{13}$$

where $w(t)$ is known (and can be derived from the forward process) Ou et al. (2025). Conceptually:

- **SEDD:** learn timestep-dependent ratios via a t-conditioned Transformer.
- **RADD:** learn time-independent clean-conditionals; treat timestep as an analytic scalar.

This also explains RADD's practical advantages:

- simpler model interface (no t embedding required for the core denoiser),
- easier optimization and better cacheability during sampling (outputs can be reused when the noisy sample does not change),
- clearer connection to any-order autoregressive models (AO-ARMs) Ou et al. (2025).

6 SMDM: SCALING UP MASKED DIFFUSION MODELS ON TEXT

SMDM Nie et al. (2025b) addresses a gap in earlier discrete diffusion work: many papers validate only at limited scale, making it unclear whether masked diffusion LMs can scale like AR models. SMDM contributes two key insights: **(i) scaling-law evidence** and **(ii) simple mechanisms that significantly improve downstream performance.**

6.1 SCALING LAWS: DO MASKED DIFFUSION LMS SCALE LIKE AR?

SMDM establishes one of the first scaling-law studies for masked diffusion models (MDMs) on text, showing that their loss scaling trend with compute/model/data can be comparable to autoregressive models, with a relatively small compute gap under controlled settings Nie et al. (2025b). This answers a fundamental concern: diffusion LMs are not inherently "unscalable"; the main challenge is often **inference cost** (many denoising steps), not necessarily pretraining scalability.

6.2 WHY INFERENCE IS STILL HARD: NFE AND STEP BUDGET

Even if training scales, generation typically requires multiple denoising iterations. Let S be the number of denoising steps (number of function evaluations, NFE). Then latency roughly scales with S forward passes. This makes system-level acceleration and/or reducing S without quality collapse a first-class problem, motivating Fast-dLLM and block-diffusion conversion.

6.3 PRACTICAL IMPROVEMENT: SIMPLE GUIDANCE FOR MASKED DIFFUSION

SMDM further highlights that diffusion-style generation quality can often be improved using simple guidance mechanisms, analogous in spirit to classifier-free guidance in continuous diffusion Nie et al. (2025b). At a high level, if the model can produce an "unconditional" prediction and a "conditional" prediction, one can interpolate logits to trade off diversity and fidelity:

$$\text{logits} \leftarrow (1+\gamma)\,\text{logits}_{\text{cond}} - \gamma\,\text{logits}_{\text{uncond}}, \tag{14}$$

where γ controls guidance strength Nie et al. (2025b). This matters for language because masked diffusion often struggles with global coherence if steps are too few; guidance helps compensate.

7 FAST-DLLM: WHY KV CACHE BREAKS IN FULL-ATTENTION DLLMS, AND HOW TO FIX IT

A critical deployment bottleneck for diffusion LMs is inference inefficiency. Compared to AR decoding, diffusion-style decoding often cannot directly benefit from standard KV cache, and also suffers when attempting to decode many tokens in parallel.

Fast-dLLM Wu et al. (2025b) is important because it explicitly targets the **root systems issue**: enabling KV cache and parallel decoding for diffusion LLMs.

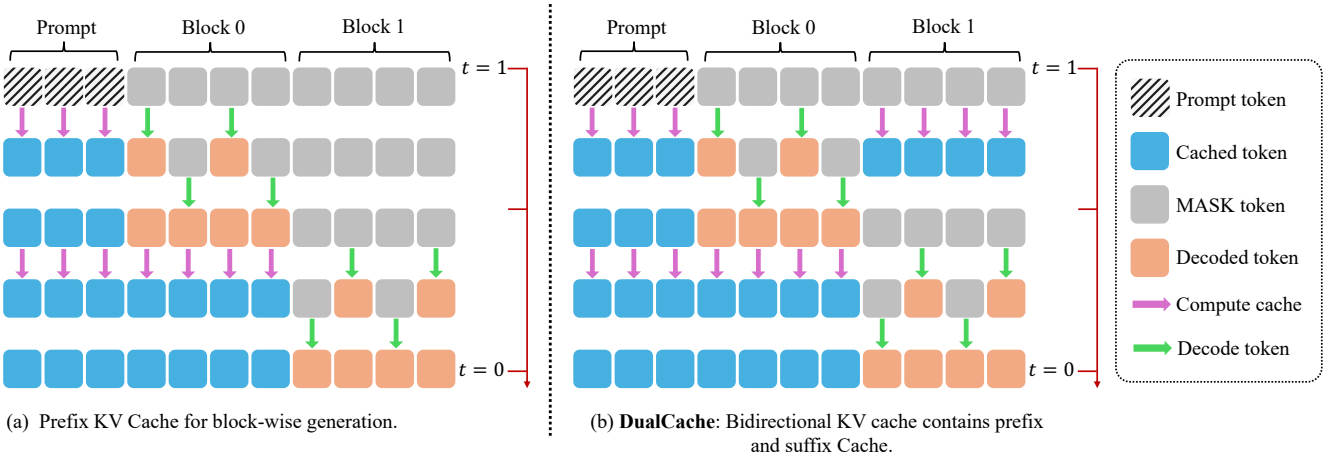
7.1 WHY "STANDARD KV CACHE" DOES NOT WORK IN FULL-ATTENTION DIFFUSION

In autoregressive decoding, KV cache works because:

- attention is causal (future tokens are never attended),
- once a token is generated, its key/value contribution to later tokens is fixed,
- we can reuse previously computed K/V for the prefix.

In many diffusion LMs (masked denoisers), attention is **bidirectional** within the whole sequence canvas. During sampling, the sequence content changes repeatedly due to masking/unmasking (and remasking). This breaks KV cache reuse for two reasons:

1. **Bidirectional dependence:** each position's representation can depend on *all* other positions, including future slots. When any slot changes, it can affect many attention outputs.



(a) Prefix KV Cache for block-wise generation.

(b) **DualCache**: Bidirectional KV cache contains prefix and suffix Cache.

Figure 1: **Illustration of our Key-Value Cache for Block-Wise Decoding.** (a) During prefix-only caching, the KV cache is computed once for the prompt and reused across multiple decoding steps within each block. The cache is updated after completing a block to maintain consistency, with negligible overhead. (b) DualCache extends this approach by caching both prefix and masked suffix tokens, further accelerating decoding. The high similarity of KV activations across steps allows effective reuse with minimal approximation error.

2. **Remasking changes the "input tokens" repeatedly:** the set of masked vs. unmasked tokens changes across steps, so the hidden states (and thus K/V) are not stable across iterations.

Therefore, naive reuse of K/V computed at an earlier denoising step is invalid and can cause quality collapse.

7.2 Fast-dLLM's answer: block-wise approximate cache + confidence-guided parallel decoding

Fast-dLLM proposes a **block-wise approximate KV cache** tailored to diffusion LMs, enabling reuse with negligible quality drop Wu et al. (2025b), as shown in 1. It also analyzes why naive parallel decoding degrades quality: unmasking many tokens assumes conditional independence and disrupts token dependencies. Fast-dLLM mitigates this using confidence-aware selection so that only sufficiently reliable predictions are accepted in parallel Wu et al. (2025b).

This section sets up the natural next step: instead of fighting full-sequence bidirectional attention, **design the model so that caching becomes structurally valid**—which is exactly what block-diffusion conversion (SDAR / Fast-dLLM v2 / Block Diffusion) does.

8 AR → Block-Diffusion Conversion: Efficient Training Masks and AR-Aligned Prediction

A major recent direction is *post-training conversion*: start from a strong pretrained AR LLM and adapt it into a **block-diffusion** generator, preserving AR capabilities while enabling parallel decoding Arriola et al. (2025); Cheng et al. (2025); Wu et al. (2025a). Compared to training diffusion LMs from scratch (e.g., large-scale MDM pretraining), conversion is far more data- and compute-efficient.

8.1 Block diffusion: semi-autoregressive factorization

Partition a length-L sequence into K contiguous blocks of size D:

$$x = (b_1, \ldots, b_K), \qquad b_k \in \mathcal{V}^D, \qquad KD = L. \tag{15}$$

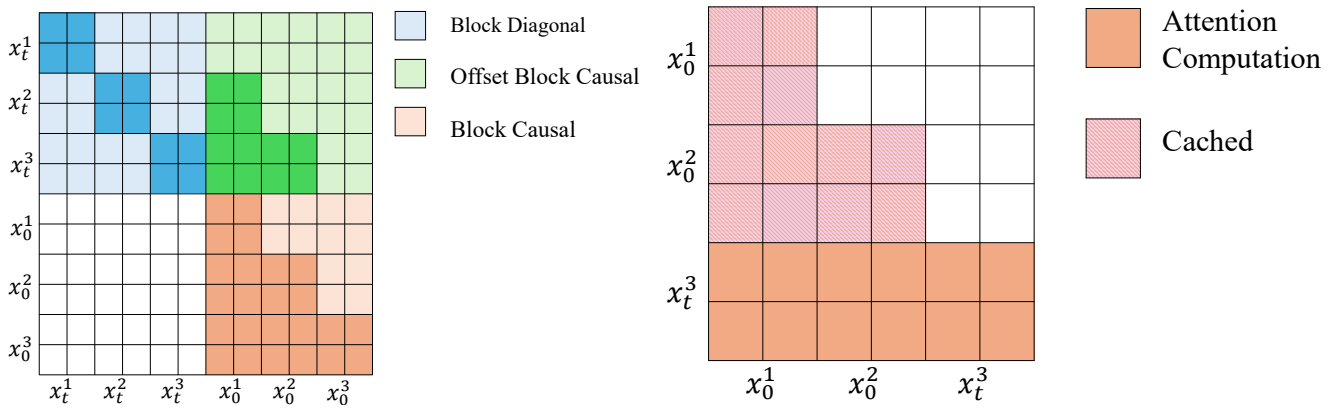
Block diffusion keeps AR structure across blocks:

$$P_\theta(x) = \prod_{k=1}^{K} P_\theta(b_k \mid b_{<k}), \tag{16}$$

but implements each conditional $P_\theta(b_k \mid b_{<k})$ using **intra-block masked diffusion / iterative denoising**, allowing parallel token updates within a block Arriola et al. (2025); Cheng et al. (2025).

8.2 Attention Mask Design for Block-wise Diffusion Training

8.3 Attention Mask Design



(a) Training-time attention mask.

(b) Inference-time attention mask.

Figure 2: Specialized attention mask design for diffusion language modeling. **(a)** During training, each input consists of a corrupted sequence x_t and corresponding targets x_0, concatenated and processed in a single forward pass. The attention mask combines intra-block bidirectional attention (Block Diagonal), cross-block causal dependency from clean tokens to noised ones (Offset Block Causal), and traditional left-to-right causality among clean tokens (Block Causal). **(b)** During inference, previously decoded blocks of x_0 are reused via caching. Only the current noised block x_t is computed in each decoding step, which attends to cached prefixes (shaded) and updates its own block in a self-contained fashion.

A core implementation trick in both Fast-dLLM v2 and SDAR is a **specialized** $2L \times 2L$ **attention mask** that enables *block-wise diffusion training on top of a pretrained AR LLM* in a *single forward pass* Wu et al. (2025a). The key is to concatenate the **noised sequence** x_t and the corresponding **clean sequence** x_0 into a length-$2L$ input:

$$\tilde{x} = [x_t;\ x_0] \in \mathcal{V}^{2L}. \tag{17}$$

Here we slightly abuse notation as in the appendix: x^b denotes the set of tokens in block b (block size D), and the training target is the block-conditional distribution

$$p_\theta(x^b \mid x_t^b,\ x^{<b}), \tag{18}$$

where x_t^b is the noised version of the current block and $x^{<b}$ are all *clean* tokens in previous blocks.

Block-structured $2L \times 2L$ **mask.** Fast-dLLM v2 constructs a full mask $\mathcal{M}_{\text{full}} \in \{0,1\}^{2L\times 2L}$ with a **2-by-2 block form**, which is shown as 2(a):

$$\mathcal{M}_{\text{full}} = \begin{bmatrix} \mathcal{M}_{\text{BD}} & \mathcal{M}_{\text{OBC}} \\ 0 & \mathcal{M}_{\text{BC}} \end{bmatrix}. \tag{19}$$

This design explicitly separates (i) **within-block bidirectional refinement** in the noised part, and (ii) **block-level causal semantics** carried by the clean part, while preventing information leakage from x_t back into x_0.

(1) $\mathcal{M}_{\mathrm{BD}}$: Block-diagonal bidirectional attention (inside x_t). Within the noised half x_t, tokens only attend bidirectionally *within the same block*, enabling parallel refinement:

$$[\mathcal{M}_{\mathrm{BD}}]_{ij} = \begin{cases} 1, & \text{if } i, j \text{ belong to the same block,} \\ 0, & \text{otherwise.} \end{cases} \tag{20}$$

So the model performs diffusion-style denoising *locally within each block* rather than globally across the whole sequence.

(2) $\mathcal{M}_{\mathrm{OBC}}$: Offset block-causal attention (from x_t to x_0 prefixes). Each noised token is allowed to attend to *clean tokens in previous blocks* from the x_0 half:

$$[\mathcal{M}_{\mathrm{OBC}}]_{ij} = \begin{cases} 1, & \text{if } j \text{ is in a block before } i, \\ 0, & \text{otherwise.} \end{cases} \tag{21}$$

This is the crucial "**conditioning channel**" that realizes $p_\theta(x^b \mid x_t^b, x^{<b})$: the current noised block can use the *clean prefix blocks* as fixed causal context, matching AR semantics across blocks.

(3) 0: No leakage from x_t back to x_0. The lower-left block is set to zero so tokens in the clean half x_0 *never attend to* the noised half x_t. This avoids contaminating the clean-stream representations with time-dependent noise patterns, which is important for stability and for preserving AR-like behavior.

(4) $\mathcal{M}_{\mathrm{BC}}$: Block-causal attention (inside x_0). Inside the clean half x_0, Fast-dLLM v2 uses a **block-causal** mask: a token can attend to all tokens in the same block and all previous blocks:

$$[\mathcal{M}_{\mathrm{BC}}]_{ij} = \begin{cases} 1, & \text{if } j \text{ is in the same or an earlier block as } i, \\ 0, & \text{otherwise.} \end{cases} \tag{22}$$

This enforces left-to-right generation *at the block level*, while allowing full interaction *within a block* (which matches the block-diffusion decoding style used at inference).

Why the $2L \times 2L$ mask matters (the correct takeaway). This mask is not merely "two views"; it is a **structural factorization** that makes block-wise diffusion training efficient and AR-consistent:

- It trains the desired conditional $p_\theta(x^b \mid x_t^b, x^{<b})$ by letting x_t attend to clean prefix blocks (via $\mathcal{M}_{\mathrm{OBC}}$), while keeping refinement local within blocks (via $\mathcal{M}_{\mathrm{BD}}$).
- It preserves AR inductive bias by ensuring the clean stream is block-causal and never sees the noised stream (bottom-left zero block).
- It sets up inference-time caching naturally: previously decoded clean blocks behave as a frozen prefix context, and only the current noised block needs iterative refinement Wu et al. (2025a).

8.4 SDAR: LIGHTWEIGHT AR-TO-BLOCK-DIFFUSION CONVERSION

SDAR Cheng et al. (2025) performs a paradigm conversion:

1. start from pretrained AR θ_{AR},
2. continue training with a conditional block-denoising objective,
3. decode autoregressively across blocks, but denoise in parallel within each block.

The key advantage is conceptual and practical: AR provides strong global coherence and training efficiency; diffusion provides intra-block parallelism and bidirectional refinement Cheng et al. (2025).

8.5 FAST-DLLM V2: SDAR-LIKE BLOCK DIFFUSION, PLUS TWO CRUCIAL "ENGINEERING" DIFFERENCES

Fast-dLLM v2 Wu et al. (2025a) is extremely close in paradigm to SDAR (both are AR$\rightarrow$block diffusion conversion), but it emphasizes two differences you highlighted:

(i) Shifted prediction (AR-aligned prediction pathway). Masked-token prediction normally uses the same position's hidden state. But AR models are trained to predict x_i from position $i-1$ hidden state (NTP geometry). Fast-dLLM v2 therefore uses **shifted prediction**: predict the token at i using the hidden state at $i-1$ when x_i is masked. This preserves AR representations and stabilizes conversion Wu et al. (2025a).

(ii) Complementary masking (stronger and more balanced supervision). Fast-dLLM v2 uses complementary masks so every token is trained both as context and as target (in expectation), improving data efficiency Wu et al. (2025a).

(iii) Hierarchical caching for fast inference. Beyond the conversion objective, Fast-dLLM v2 designs block-level and sub-block caches to support high-throughput inference, achieving strong speedups while maintaining accuracy Wu et al. (2025a).

8.6 Unifying view: what SDAR and Fast-dLLM v2 are really doing

Both SDAR and Fast-dLLM v2 can be seen as optimizing a conditional denoising objective compatible with a pretrained AR backbone:

$$\max_{\theta} \; \mathbb{E}_{k,t} \, \mathbb{E}_{\text{corruption}} \left[\log P_\theta(b_k \mid \text{corrupt}_t(b_k),\, b_{<k}) \right], \tag{23}$$

while enforcing **AR-consistent structure across blocks** and **parallel refinement within blocks**. Their primary difference is how aggressively they preserve AR geometry (shifted prediction) and how they structure training/inference engineering (complementary masks, hierarchical cache) Cheng et al. (2025); Wu et al. (2025a).

9 Conclusion

Diffusion language modeling becomes practical only when we understand (and exploit) the right abstractions. In continuous diffusion, the essential scalability insight is to learn *noise-conditional* scores rather than the raw data score, turning score learning into supervised noise regression. In discrete language diffusion, SEDD shows how to replace gradients with ratio-based "scores", while RADD provides a key simplification: the timestep dependence can be separated analytically, removing the need for a time-conditioned Transformer and reducing training to an AR-like denoising cross-entropy. At scale, SMDM provides evidence that masked diffusion LMs can follow competitive scaling trends, shifting the main challenge to inference efficiency. Finally, systems and conversion techniques close the deployment gap: Fast-dLLM explains why full-attention DLLMs cannot directly use KV cache and proposes cache- and confidence-aware acceleration, while block-diffusion conversion (Block Diffusion, SDAR, Fast-dLLM v2) offers a compelling "best of both worlds": preserve pretrained AR capabilities and coherence across blocks, but enable parallel denoising within blocks through efficient attention masks and AR-aligned prediction pathways.

References

M. Arriola et al. Block diffusion: Interpolating between autoregressive and diffusion language models. In *International Conference on Learning Representations (ICLR)*, 2025.

Shuang Cheng, Yihan Bian, Dawei Liu, Yuhua Jiang, Yihao Liu, Linfeng Zhang, Wenhai Wang, Qipeng Guo, Kai Chen, Biqing Qi, and Bowen Zhou. Sdar: A synergistic diffusion-autoregression paradigm for scalable sequence generation. *arXiv preprint arXiv:2510.06303*, 2025.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Aapo Hyvärinen. Estimation of non-normalized statistical models by score matching. In *Journal of Machine Learning Research*, 2005.

Aaron Lou et al. Discrete diffusion modeling by estimating the ratios of the data distribution. In *International Conference on Machine Learning (ICML)*, 2024.

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*, 2025a.

Shen Nie et al. Scaling up masked diffusion models on text. In *International Conference on Learning Representations (ICLR)*, 2025b.

Jingyang Ou, Shen Nie, Kaiwen Xue, Fengqi Zhu, Jiacheng Sun, Zhenguo Li, and Chongxuan Li. Your absorbing discrete diffusion secretly models the conditional distributions of clean data. In *International Conference on Learning Representations (ICLR)*, 2025.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations (ICLR)*, 2021.

Chengyue Wu, Hao Zhang, Shuchen Xue, Shizhe Diao, Yonggan Fu, Zhijian Liu, Pavlo Molchanov, Ping Luo, Song Han, and Enze Xie. Fast-dllm v2: Efficient block-diffusion llm. *arXiv preprint arXiv:2509.26328*, 2025a.

Chengyue Wu, Hao Zhang, Shuchen Xue, Zhijian Liu, Shizhe Diao, Ligeng Zhu, Ping Luo, Song Han, and Enze Xie. Fast-dllm: Training-free acceleration of diffusion llm by enabling kv cache and parallel decoding. *arXiv preprint arXiv:2505.22618*, 2025b.