PLANNED DIFFUSION

Daniel Israel[1*] **Tian Jin**[2*] **Ellie Cheng**[2] **Guy Van den Broeck**[1]
Aditya Grover[1] **Suvinay Subramanian**[3] **Michael Carbin**[2]

[1]University of California, Los Angeles [2]MIT CSAIL [3]Google

ABSTRACT

Most existing large language models are autoregressive: they generate text one token at a time, and cannot decode any new tokens until they have decoded every token before it. Discrete diffusion language models offer a promising alternative by generating multiple tokens in parallel, but sampling from them requires a *denoising order*, the strategy for deciding which tokens to decode at each step. Determining the right denoising order is difficult, and existing approaches use heuristics that create a steep trade-off between quality and latency. We propose *planned diffusion*, a system that trains the model to determine its own denoising order. Planned diffusion uses a single model that transitions between autoregressive and diffusion-based generation: first, the model autoregressively generates a plan that partitions the response into semantically independent chunks, defining a denoising order that parallelizes sampling across chunks; second, the model executes this plan via diffusion denoising. On AlpacaEval, a suite of 805 instruction-following prompts, planned diffusion achieves Pareto-optimal trade-off between quality and latency, achieving 1.27x to 1.81x speedup over autoregressive generation with only 0.87% to 5.4% drop in win rate. Our empirical results show that planned diffusion exhibits superior performance scaling on downstream tasks compared to autoregressive baselines while offering the runtime flexibility to precisely navigate the quality-latency trade-off.

1 INTRODUCTION

Large language models achieve strong performance on diverse tasks such as open-ended dialogue (Ouyang et al., 2022), code generation (Chen et al., 2021), and mathematical reasoning (Lewkowycz et al., 2022). Most existing large language models are autoregressive (Brown et al., 2020; Chowdhery et al., 2023; Touvron et al., 2023): they generate text one token at a time, and cannot decode any token until they have decoded every token before it. This sequential dependence between decoding steps within a response creates a fundamental latency bottleneck. Discrete diffusion language models (Sahoo et al., 2024) take a different approach: they decode multiple tokens per step. However, sampling from discrete diffusion models requires a *denoising order*: the strategy for deciding which tokens to decode at each step. Determining the right denoising order is difficult, and existing approaches use heuristics such as random order or confidence-based thresholds, creating a steep trade-off between quality and latency (Wu et al., 2025).

We propose *planned diffusion*, a system that trains the model to determine its own denoising order. Typical language model responses contain semantically independent chunks that do not need to be denoised sequentially. For example, in a response with a bulleted list, the model can denoise each bullet point concurrently (Ning et al., 2024). Planned diffusion exploits this by having the model identify independent chunks itself and denoise tokens in different chunks in parallel.

[*]Equal contribution.

Figure 1: A real example of planned diffusion. (1) The model first generates a sequential plan using control tags to define the structure and length of independent text chunks. (2) The runtime then translates this plan into a scaffold, initializing each chunk with a corresponding number of mask tokens. (3) The model denoises all chunks in parallel with diffusion, generating the text for each section simultaneously to produce the complete response.

Planned diffusion uses a single model that transitions between autoregressive and diffusion-based generation. First, in an autoregressive planning stage, the model sequentially generates a generation plan composed of structural control tags. This plan partitions the response into a set of semantically independent chunks, defining a denoising order so that tokens in different chunks can denoise in parallel. Second, in a parallel diffusion stage, the model executes this denoising order, filling in all planned chunks simultaneously via diffusion denoising. This single-model, hybrid approach offers a distinct architectural advantage over other acceleration techniques, such as speculative decoding (Leviathan et al., 2023), which requires a separate draft model. To the best of our knowledge, this is the first text-only model trained with both discrete diffusion and autoregressive objectives. Figure 1 presents a sample generation produced by our planned diffusion model.

We make the following contributions:

1. We introduce *planned diffusion*, a new parallel generation technique that decomposes text generation into a sequential planning stage and a parallel diffusion stage.
2. We design the control tag language, model training methodology, and inference algorithm that enable a single model to perform this hybrid generation process.
3. We demonstrate that planned diffusion achieves a Pareto-optimal trade-off between quality and latency on AlpacaEval, a suite of 805 instruction-following prompts; it achieves a 1.27x to 1.81x speedup over autoregressive generation while incurring only a 0.87% to 5.4% drop in win rate, respectively.
4. We present sensitivity analysis that validates that the planning mechanism of our model is minimal and reliable, and show that simple runtime knobs offer tunable control over the quality–latency trade-off.

2 RELATED WORK

Our work builds upon recent developments in non-autoregressive and parallel decoding strategies. We position our contributions in the context of two primary research areas: diffusion-based language models and methods for achieving semantic parallelism.

Diffusion Language Models. Diffusion models have recently emerged as a new paradigm for generative language tasks (Austin et al., 2021; Sahoo et al., 2024). A significant body of research is focused on accelerating the inference process, which traditionally involves many iterative denoising steps (Liu et al., 2025a). These acceleration techniques include KV caching for diffusion models (Ma et al., 2025; Liu et al., 2025d), the use of autoregressive verification (Hu et al., 2025; Israel et al., 2025a), and the development of fast sampling strategies that reduce the number of required steps (Wu et al., 2025; Li et al., 2025; Hong et al., 2025). Our proposed planned diffusion framework is orthogonal and complementary to these methods; any diffusion sampling strategy can be integrated into the diffusion component of our algorithm to further increase performance. Other related work include block diffusion (Arriola et al., 2025) which enforces an autoregressive structure over blocks and planned denoising (Liu et al., 2025c) which learns an adaptive denoising schedule. While relevant, neither targets semantic parallelism, which is the purpose of planned diffusion.

Semantic Parallelism. While diffusion models achieve token-wise parallelism, they are not trained to achieve parallelism across larger portions of text at the semantic level. We define *semantic parallelism* as a broad class of techniques that produce models capable of parallelizing over semantically independent chunks of tokens. Many recent works explore semantic parallelism (Ning et al., 2024; Liu et al., 2024; Jin et al., 2025; Rodionov et al., 2025; Pan et al., 2025; Wen et al., 2025; Yang et al., 2025). Existing works operate within a purely autoregressive framework, while to the best of our knowledge, our work is the first to propose a hybrid autoregressive-diffusion model for text (see Appendix E for a more thorough discussion). By combining an autoregressive planning phase with a parallel diffusion phase, our method utilizes diffusion parallelism in a structured manner corresponding to semantic parallelism, presenting a novel approach for efficient text generation.

Other Parallel Generation Techniques. There is a rich line of work on parallel generation. Insertion-based models parallelize by inserting tokens into partial positions, reducing decoding depth via simultaneous insertion updates (Stern et al., 2019). Speculative decoding accelerates autoregressive models by drafting multiple tokens and verifying them in parallel (Leviathan et al., 2023; Chen et al., 2023; Zhang et al., 2024). Planned diffusion differs conceptually as we use a single hybrid model for both autoregressive and parallel generation.

3 PRELIMINARIES

Generative language models learn a probability distribution over sequences of discrete tokens. In this work, our focus is on two main paradigms: autoregressive and discrete diffusion.

Autoregression. Autoregressive models are the standard for sequential text generation. They factorize the joint probability of a token sequence $x = (x_1, x_2, \cdots, x_L)$. The probability distribution under this model, which we denote as p_{AR}, is given by:

$$p_{\text{AR}}(x) = \prod_{i=1}^{|x|} p_\theta(x_i | x_{<i}) \tag{1}$$

where p_θ is a parameterized conditional distribution over tokens. In autoregression, each token is sequentially sampled conditioned on all previously generated tokens.

Discrete Diffusion. Discrete diffusion models learn to reverse a fixed data corruption process that gradually introduces noise into a clean sequence. For text, this involves incrementally replacing tokens with a special *mask* token (Austin et al., 2021). Let x be a clean sequence of tokens. The forward corruption process q produces a noisy version $\tilde{x}$ at a noise level $t \in [0, 1]$, masking each position independently.

$$q_{t|0}(\tilde{x}_i \mid x_i) = \begin{cases} t, & \text{if } \tilde{x}_i = \text{[MASK]} \\ 1 - t, & \text{if } \tilde{x}_i = x_i \\ 0 & \text{otherwise} \end{cases} \qquad q_{t|0}(\tilde{x} \mid x) = \prod_i q_{t|0}(\tilde{x}_i \mid x_i). \tag{2}$$

Because the true posterior over the forward noising process is intractable (Lou et al., 2024), diffusion models learn an approximation by maximizing a variational lower bound on the log-likelihood (Sahoo et al., 2024).

$$\log p_{\text{D}}(x) \geq \mathbb{E}_{t \sim U(0,1),\, \tilde{x} \sim q_{t|0}(\tilde{x}|x)} \frac{1}{t} \sum_i \mathbb{1}(\tilde{x}_i = \text{[MASK]}) \log p_\theta(x_i \mid \tilde{x}). \tag{3}$$

A key property of a trained diffusion model is that for any disjoint sets of positions $\mathcal{O}, \mathcal{Q} \subseteq [L]$, we can obtain a conditional distribution p_{D} over queried tokens at positions $\mathcal{Q}$ given observed tokens at positions $\mathcal{O}$. This conditional distribution factorizes as follows

$$p_{\text{D}}(x_{\mathcal{Q}} \mid x_{\mathcal{O}}) = \prod_{i \in \mathcal{Q}} p_{\text{D}}(x_i \mid x_{\mathcal{O}}). \tag{4}$$

In planned diffusion, we will assume p_{AR} and p_{D} can be conditioned on prior context tokens $c = (c_1, c_2, \cdots)$. Thus, we shall utilize the distributions $p_{\text{AR}}(\cdot \mid c)$ and $p_{\text{D}}(\cdot \mid c)$ respectively.

Denoising Order. Sampling from a discrete diffusion model requires selecting a *denoising order* (Kim et al., 2025; Turok et al., 2026). We first define an *ordered partition* of positions $[L] = \{1, \ldots, L\}$ as a tuple $\sigma = (\sigma_1, \ldots, \sigma_T)$ of non-empty disjoint subsets with $\bigcup_{t=1}^{T} \sigma_t = [L]$. We write $\sigma_{<t} = \sigma_1 \cup \cdots \cup \sigma_{t-1}$ for positions revealed before step t. Positions within the same subset decode in *parallel*; positions in different subsets decode *sequentially*. Thus, a denoising order is a procedure that maps available inputs to an ordered partition of $[L]$. Any ordered partition σ and prefix tokens c imply a joint distribution

$$p_{\text{D}}(x \mid c;\, \sigma) = \prod_{t=1}^{T} p_{\text{D}}(x_{\sigma_t} \mid x_{\sigma_{<t}}, c)\,. \tag{5}$$

obtained by substituting $\mathcal{O} = \sigma_{<t}$ and $\mathcal{Q} = \sigma_t$ into Equation 4. We shall define *uniform* and *entropy* based ordering. For the sake of simplicity, we will only define the orderings where $|\sigma_t| = 1$ for all t. Let π be a permutation of $[L]$ drawn uniformly at random. The uniform ordered partition is

$$\sigma_t^{\text{Unif}} := \{\pi(t)\} \qquad t = 1, \ldots, L \tag{6}$$

Uniform random unmasking is the denoising order implied by the training objective in Equation 3, but in practice it does not achieve the best results compared to entropy ordered unmasking (Israel et al., 2025a). The entropy ordered partition is constructed such that at each step t, it selects the lowest predicted token entropy

$$\sigma_t^{\text{Ent}} := \operatorname*{arg\,min}_{i \notin \sigma_{<t}} H\big(p_{\text{D}}(x_i \mid x_{\sigma_{<t}})\big) \qquad t = 1, \ldots, L \tag{7}$$

where H denotes Shannon entropy. Unlike σ^{Unif}, the partition is not fixed in advance but depends on the tokens revealed at each step. This is the default inference strategy of Ye et al. (2025).

4 PLANNED DIFFUSION

Planned diffusion combines an autoregressively sampled plan with discrete diffusion. The plan segments the output into semantically independent chunks and conditions the content of each; the diffusion model then fills in the chunks in parallel, each of which with any denoising order. Unlike a denoising order, which can govern only *when* and *where* each position is sampled, the plan can also determine *what* is sampled in parallel.

4.1 FORMAL DESCRIPTION

We formalize a single iteration of planned diffusion. Beginning from prior tokens c, the model autoregressively generates a *plan* z, which specifies K chunks with lengths $l_1, \ldots, l_K$. This induces a total sequence length $L = \sum_{k=1}^{K} l_k$ and a partition of $[L]$ into contiguous chunks $s_1, \ldots, s_K$, where $s_k = \{l_{k-1} + 1 \,..\, l_k\}$ for $k = 1, \ldots, K$. We can define a denoising order for each chunk by letting $\sigma^{(k)}$ be any ordered partition of s_k into exactly l_k steps. Thus, planned diffusion admits a joint distribution over plan z and tokens x generated in order $\sigma(z)$ conditioned on previous tokens c.

$$\sigma_t^{\mathrm{Plan}}(z) = \bigcup_{\{k\,:\,t \le l_k\}} \sigma_t^{(k)}, \qquad t = 1, \ldots, \max_k l_k \tag{8}$$

which reveals one position from every chunk S_k with $l_k \geq t$ simultaneously at step t. The diffusion phase generates content $x \in \mathcal{V}^L$, and the joint distribution over plan and content is:

$$p_{\mathrm{PD}}(z, x \mid c;\, \sigma_t^{\mathrm{Plan}}(z)) = \underbrace{p_{\mathrm{AR}}(z \mid c)}_{\text{Planning}} \cdot \underbrace{p_{\mathrm{D}}\big(x \mid z,\, c;\, \sigma_t^{\mathrm{Plan}}(z)\big)}_{\text{Diffusion}} \tag{9}$$

Because p_{D} accepts any valid ordered partition of $[L]$, the choice of $\sigma^{(k)}$ is fully decoupled from the model. Setting each $\sigma^{(k)} = \sigma^{\mathrm{Ent}}$ restricted to S_k recovers the entropy strategy applied independently within each chunk, which is the configuration used in our experiments. The full planned diffusion sampling algorithm, which can perform multiple iterations of planning and diffusion, can be seen in Algorithm 1.

Algorithm 1 Planned Diffusion

function PLANNED_DIFFUSION(c)
  loop
    Sample plan $z \sim p_{\mathrm{AR}}(\cdot \mid c)$ ▷ Autoregressively generate plan
    Parse z to get K chunks $s_1, \ldots, s_K$ with lengths $l_1, \ldots, l_K$
    for $k = 1, \ldots, K$ **do in parallel**
      Sample $x_{s_k} \sim p_{\mathrm{D}}\Big(\cdot \mid z,\, c;\, \sigma^{(k)}\Big)$ ▷ Diffuse each chunk in parallel
    end for
    $x \leftarrow (x_{s_1}, \ldots, x_{s_K})$ ▷ Concatenate chunks together
    $c \leftarrow (c,\, z,\, x)$ ▷ Append plan and output to prior tokens
    if z[end] = <eos/> **then**
      break ▷ Final planning token triggers end of generation
    end if
  end loop
  $c \leftarrow$ REMOVECONTROLTAGS(c) ▷ Strip planning tokens from final output
  return c
end function

4.2 IMPLEMENTATION

Control Tags. Chunks denoting groups of semantically related tokens are first defined during a sequential planning stage. This is done using a paired `<topic>...</topic>` tag structure. Within this structure, the model generates a concise description of the chunk's content (e.g., "definition" in Figure 1) and its predicted length (e.g., "30" in Figure 1). During the parallel diffusion stage, the model generates the tokens for each chunk within a corresponding `<async>...</async>` tag pair. Finally, the `<sync/>` tag conveys generation dependency: tokens that follow `<sync/>` may require details produced inside the preceding `<async>` chunks, so the sampling algorithm continues sequential planning only after those chunks are filled and available. We add all control tags to the model's vocabulary for training and inference and strip them during post-processing of final outputs.

Finetuning Dataset. We annotate the SlimOrca instruction–finetuning dataset (Lian et al., 2023) for parallel generation, following Jin et al. (2025). We prompt a GEMINI model with the syntax and semantics of the control tags. See Appendix A for more details. GEMINI inserts the control tags into the response from each instruction-response pair in the dataset. The opening `<async>` carries two attributes: `topic` (a concise label, $\leq$ 3 words) and `tokens` (a coarse length estimate, e.g., multiples of 10 tokens). We impose that every non-control-tag token lies inside exactly one `<async>...</async>` chunk (no nesting, no overlap). During preprocessing for training, we insert 0–10 tokens of stochastic padding inside each `<async>` chunk, allowing the diffusion model to generate with variable length shorter than the masked input. We validate well-formedness (balanced tags, coverage, non-overlap, attribute types/ranges) and discard malformed cases. Figure 5 contains a concrete example of the tagging language used by planned diffusion.

Training Loss. The goal of training is to maximize the joint probability over plan tokens and their content. Given an annotated dataset $\mathcal{D}$ in which a single clean example is given by $Y \in \mathcal{D}$, we decompose Y into sets of planning tokens Z and content tokens X, such that $Y = Z \cup X$. X^t is a noised sequence of tokens under noise distribution $q_{t|0}$ as previously defined. Thus, X^t contains masked tokens with probability t. Let f_θ be the function to instantiate planned diffusion, parameterized by θ. We use the notation $f_\theta(x, i)$ to signify that the model takes input x and makes a prediction at index i of the sequence. Finally, let $M_i(X)$ be an attention masking function that takes as input a sequence X and outputs a subset $M_i(X) \subseteq X$ of the sequence that f_θ has access to at a particular index i. We describe the attention masking in more detail in the following paragraph. With CE as the cross-entropy loss, our overall training objective is given by

$$\mathcal{L}(\theta) = \mathop{\mathbb{E}}_{\substack{Y\sim\mathcal{D}\\ t\sim U(0,1)}} \frac{1}{|Y|} \sum_{y_i \in Y} \underbrace{\mathbb{1}(y_i \in Z)\mathrm{CE}(f_\theta(y_{<i}, i),\, y_i)}_{\text{Autoregressive}} + \underbrace{\frac{1}{t}\mathbb{1}(y_i \in X)\mathrm{CE}\big(f_\theta(M_i(X^t \cup Z), i),\, y_i\big)}_{\text{Diffusion}} \quad (10)$$

Note that in the training objective, the same noise parameterized by t is applied to chunks across multiple iterations of planning and diffusion, where future diffusion chunks will be conditioned on previously sampled diffusion chunks. In this decision, we are applying the interpretation of a diffusion model as an any-order autoregressive model capable of supporting arbitrary conditional queries at inference time (Shi et al., 2024). The same technique is used by Llada 7B (Nie et al., 2025), which is trained with a diffusion objective, but during inference used for semi-autoregressive block sampling.

Attention Mask. We implement the following rules via the attention mask M_i. First, planning tokens, which are composed of control tags and their attributes, are given causal attention (Plan 1 and 2 in Figure 5). Second, diffused tokens use bidirectional attention, as required for diffusion-based parallel denoising (Diffusion 1 and 2 in Figure 5). In Section 5, we also show that restricting bidirectional dense attention within each chunk is another way to train planned diffusion. We refer to this variant as *planned diffusion sparse attention* (PSDA), which enforces full independence between chunks. The attention mask for PDSA can be seen in Appendix H.

Variable Length Denoising. Typically, diffusion models are configured to generate given a fixed number of denoising steps. Generation quality increases and speed decreases with the number of steps. Unlike

vanilla diffusion, planned diffusion does not generate a predetermined number of tokens, so the number of denoising steps cannot be fixed. We define a parameter r called the *steps ratio*. Given a generation length $|x|$, the number of denoising steps will be $s = r * |x|$. For a plan z and multiple diffusion chunks lengths $l_1, ..., l_K$, the step ratio is defined as $s = r * \max_k l_k$. The number of denoising steps depends only on the length of the longest chunk, because tokens in each chunk are denoised in parallel to other chunks. Higher step ratio corresponds to higher quality and slower generation, while lower step ratio leads to faster generation but worse quality.

5 Experimental Evaluation

We experimentally assess the performance of planned diffusion, focusing on its trade-off between generation quality and latency. Our results show that planned diffusion expands the latency-quality Pareto frontier for text generation when compared to autoregressive and other diffusion-based approaches. Furthermore, we demonstrate that our method scales better with additional compute; planned diffusion continues to improve with more training, whereas the performance of the autoregressive baseline plateaus.

Training setup. We fine-tune Dream-7B-Base (Ye et al., 2025; Qwen et al., 2025); the base model is first pre-trained autoregressively and then further pre-trained with a diffusion objective. We train with AdamW (Kingma & Ba, 2017; Loshchilov & Hutter, 2019), peak learning rate 5×10^{-5} with linear decay, and bfloat16 precision. We use per-GPU batch size 1 and global batch size 4. Because autoregressive and diffusion language models have different optimal epoch counts (Prabhudesai et al., 2025), we sweep epochs over $\{2, 4, 8, 16\}$. We fine-tune on Gemini-annotated SlimOrca instruction-following data (see Section 4.2); we keep the control tags for planned diffusion and strip them for autoregressive and diffusion baselines. Training runs on $4\times$H200 (141 GB) with PyTorch (Imambi et al., 2021) and HuggingFace (Wolf et al., 2020).

Decoding strategies. We compare seven decoding strategies. (1) *Autoregressive* samples tokens sequentially from the autoregressive model. (2) *Diffusion* samples masked tokens in parallel from the diffusion model; we configure the number of denoising steps to equal the number of new tokens as this produces the highest quality generation (Lou et al., 2024; Shi et al., 2025; Sahoo et al., 2024). (3) *Fast-dLLM* (Wu et al., 2025) samples from the same diffusion model but with an inference-time only optimization. We configure denoising steps to be half the number of new tokens and use a confidence threshold of 0.9, which Wu et al., 2025 use as the default value. (4) *Pasta-SFT (Jin et al., 2025)* utilizes control tags to implement semantic parallelism in a purely autoregressive setting. We compare to the SFT-trained version of Pasta, as opposed to RL-trained, for fair comparison with planned diffusion, which is trained only with SFT. (5) *Skeleton-of-Thought* (Ning et al., 2024) autoregressively samples a bullet-point outline, then applies regular-expression-based syntactic pattern matching for extracting points for parallel expansion. (6) *Planned diffusion* (ours) samples a denoising plan autoregressively from the planned diffusion model, then samples masked tokens within each chunk in parallel from the same model; for each chunk, we configure denoising steps to equal its predicted token count. (7) *Planned diffusion with sparse attention* (ours) is a variant of planned diffusion. PDSA treat concurrently generated chunks as fully independent conditioned on the plan, and as such, denoises the chunks using block-sparse attention (c.f., Figure 6). By using a sparse attention mask, PDSA improves compute efficiency with sparse attention implementations but reduces model expressivity and GPU utilization from vanilla planned diffusion.

Inference setup. We sample with temperature 0.2 and top-p 0.95 following (Ye et al., 2025), and cap the total sequence length at 2048 tokens. Inference runs on the same H200 hardware configuration.

Benchmark and metrics. We evaluate on AlpacaEval (805 instruction-following prompts) (Li et al., 2023; Dubois et al., 2024). For each method we report: (i) *average latency*—the mean wall-clock time per response; and (ii) *quality*—length-controlled win rate (LCWR) with an LLM-as-judge. We use the recom-

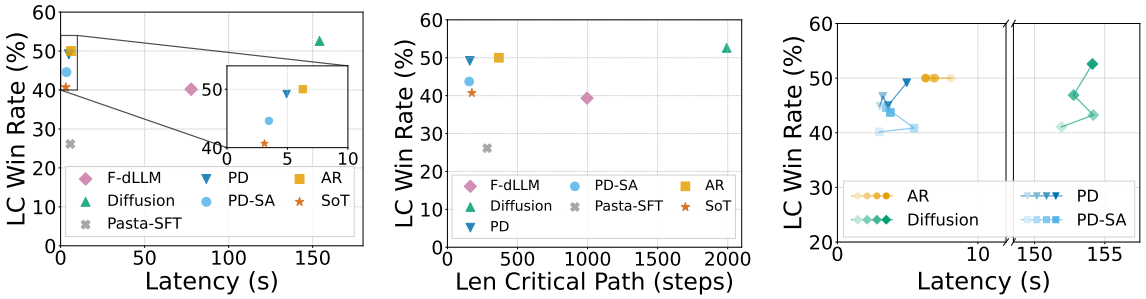

Figure 2: Evaluation of planned diffusion on the AlpacaEval benchmark. **Left:** A comparison of latency versus length-controlled win rate shows planned diffusion establishing a new Pareto frontier, offering a better trade-off between speed and quality. **Middle:** An analysis of the average critical path length reveals that planned diffusion requires substantially fewer sequential forward passes than autoregressive generation. **Right:** A scaling analysis shows that planned diffusion's win rate continues to improve with more training, while the autoregressive baseline's performance flatlines. Within each method, color brightness from lightest to darkest encodes 2, 4, 8, and 16 traing epochs. **AR**: Autoregressive; **F-dLLM**: Fast-dLLM; **SoT**: Skeleton-of-Thought; **PD**: Planned Diffusion; **PDSA**: Planned Diffusion Sparse Attention.

mended default configuration from (Dubois et al., 2024) due to its high correlation with human preference. We fix the LCWR reference to the best autoregressive baseline. We identify this reference by evaluating the autoregressive variants (2, 4, 8, and 16 epochs) against the 2-epoch variant and choosing the model with the highest quality [1]. The 16-epoch model wins and serves as the fixed reference for all LCWR scores.

Quality and Latency. We plot the latency–quality trade-off in the left figure of Figure 2. Planned diffusion (PD) achieves a 22.4× speedup over Fast-dLLM (F-dLLM) and higher quality (44.6% vs. 40.2% length-controlled win rate). Relative to autoregressive decoding (AR), planned diffusion achieves a 1.27× speedup while achieving a 49.2% length-controlled win rate against the autoregressive reference (50.0%). Planned diffusion with sparse attention (PDSA) provides another point on the speed-quality Pareto frontier by further improving speedup to 1.81× relative to autoregressive decoding at 44.6% win rate. Diffusion attains the highest quality (52.6%) but requires substantially more inference time, requiring 25× the latency of autoregressive decoding. Compared to other semantic parallelism methods, planned diffusion is firmly on the Pareto frontier. Skeleton-of-thought offers marginally faster generations but incurs a significant decrease in accuracy. Pasta-SFT fails to properly learn the control tag semantics without additional RL training and produces worse quality and latency.

Speedup Analysis. We attribute much of planned diffusion's speedup over autoregressive decoding to its shorter *critical path* of generation. We define critical path length as the number of forward passes required to produce the final response. In a dLLM, the critical path will be given by the number of denoising steps s, so the total complexity is the cost of attention multiplied by the number iterations: $O(n^2 * s)$, where n is the sequence length. Planned diffusion reduces the number of steps s by a factor of l_{max}/n, where l_{max} is the length of the longest chunk within a plan. The middle panel of Figure 2 shows that, on AlpacaEval, the average critical path of autoregressive decoding is 2.3× as long as that of planned diffusion (PD, 367.3 vs. 160.0 steps) and 2.8× as long as the sparse attention variant (PDSA, 367.3 vs. 155.2 steps). This is expected, as planned diffusion enables multiple chunks to denoise simultaneously. The realized speedup (1.85×) is smaller than the critical path reduction (2.8× for PD, 2.3× for PD-DA) because each planned diffusion step does more work: KV-cache reuse is lower and per-step compute is heavier than an autoregressive token step.We also observe a difference in the total number of tokens generated (this includes control tokens and

[1] They tie on length controlled win rate, so we break ties using raw win rates.

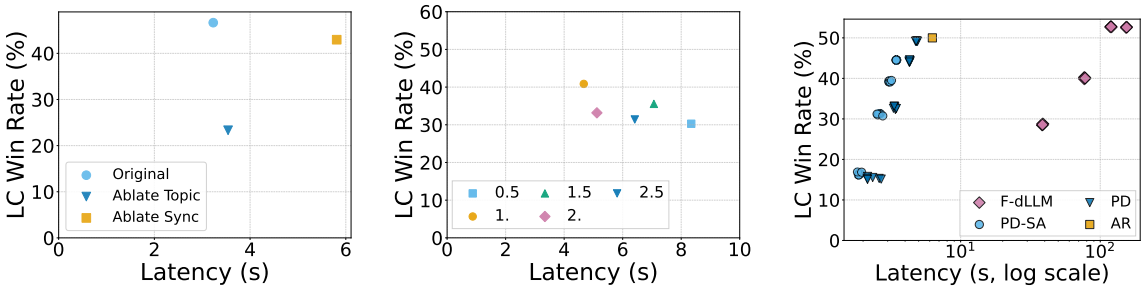

Figure 3: Additional analysis. **Left (Plan ablation):**Removing `topic` from autoregressive plan harms quality yet removing `<sync/>` tokens boosts speedup without significantly hurting quality. **Middle (chunk lengths):** LCWR peaks at a length-scaling factor of 1.0; length prediction in autoregressive plan does not exhibit systemic error. **Right (Quality versus Latency Sweep):** Varying the step ratio ($r = \{0.25, 0.5, 0.75, 1\}$) and the confidence threshold ($\tau = \{0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$) hyperparameters produces a smooth quality-latency trade-off.

pad tokens) between planned diffusion and autoregressive decoding. Planned diffusion and planned diffusion with sparse attention produce approximately 9.1%[2] and 3.4% fewer tokens, respectively.

Scaling. The right panel of Figure 2 shows how the latency-quality tradeoff evolves with training epochs for the four training configurations we examine. Autoregressive training shows no benefit from additional training compute: the length-controlled win rate remains flat at 50.0% across 2, 4, 8, and 16 epochs. Both variants of planned diffusion benefit moderately from additional training compute. Planned diffusion improves from 44.9% (2 epochs) to 49.2% (16 epochs), a gain of 4.3 percentage points, while planned diffusion with sparse attention improves from 40.2% (2 epochs) to 43.7% (16 epochs), a gain of 3.5 percentage points. Diffusion benefits significantly from additional training compute, rising from 41.1% (2 epochs) to 52.6% (16 epochs), a gain of 11.5 percentage points that ultimately surpasses the autoregressive baseline.

Takeaway. Planned diffusion sets a new latency–quality Pareto frontier and continuously improves with more training, while the autoregressive baseline plateaus.

6 Additional Analysis

We present additional analysis on several design decisions key to planned diffusion.

Denoising Plan Ablation. We remove two components of the autoregressive plan and evaluate their impact on inference quality and latency. We train all model variants for 4 epochs in this study. The original 4-epoch planned diffusion achieves an LC win rate of 46.65% at 3.23 s latency. Removing the `topic` attribute from the training data and re-training the planned diffusion model significantly harms inference quality. Specifically, it reduces the length controlled win rate from 46.65% to 23.33% and the latency from 3.23 seconds to 3.54 seconds. We conclude that `topic` attributes are critical for maintaining inference quality.

Recall that `<sync/>` marks a synchronization barrier: decoding beyond it begins only after all prior content is finalized. Deleting all `<sync/>` tokens from the training data and re-training the planned diffusion model modestly reduces quality. Specifically, the LC win rate falls from 46.65% to 42.96% while the latency

[2]The gap in output length is in part due to difference in optimal training epochs for planned diffusion (8ep) and autoregressive (16ep). The output length difference between 16ep planned diffusion and autoregressive reduces to 5.6%.

increases from 3.23 s to 5.81 s. We conclude that omitting <sync/> slightly reduces quality but does not offer a latency benefit in this setting.

chunk Lengths. We test whether the model predicts chunk lengths accurately. Accurate length prediction is key to achieving good generation quality for planned diffusion, as the diffusion denoising phase of generation cannot alter the chunk length. Systematic over-prediction wastes time by adding masks and denoising steps, while systematic under-prediction harms quality by forcing content truncation. To test for potential systematic deviation from the optimal generation length, we multiply the model's predicted chunk length by a length scaling factor to set the number of masks, sweeping the factor over $\{0.5, 1.0, 1.5, 2.0, 2.5\}$. We then measure length-controlled win rate (LCWR) and latency under identical inference settings. Latency rises with factors above 1.0 as expected because larger chunks require more mask tokens and denoising steps. Quality peaks at the model's originally predicted length (i.e scaling factor of 1.0). Deviating by $\pm 50\%$ of the predicted length reduces LCWR. Interestingly, we do not observe additional denoising steps leading to accuracy improvements. The model's length predictions are accurate; we do not observe systematic over/under-prediction.

Quality versus Latency Sweep. By default, to maximize generation quality, we set the number of denoising steps for a chunk equal to that chunk's length, and we unmask a position only when the model is highly confidence of its prediction. In this analysis, we measure the quality–latency trade-off by varying the number of denoising steps as well as confidence threshold for decoding.

Planned diffusion employs a hyperparameter called step ratio (see Section 4.2), which determines the number of denoising steps relative to the chunk length. The step ratio r scales steps relative to chunk length. When r equals 1 the number of steps equals the number of tokens; smaller r uses fewer steps per token. The confidence threshold τ used by Wu et al. (2025) selects which positions to unmask at each denoising step. For each masked position, if the model's top-token probability at that position is at least τ, we decode that token and unmask the position; otherwise we keep it masked for decoding at a later time step.

Sweeping r over $\{0.25, 0.5, 0.75, 1\}$ and τ over $\{0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$ yields a smooth trade-off between generation quality and inference latency; Across the step ratio and the confidence threshold, planned diffusion and its sparse attention variant yield higher quality at equal or lower latency for most operating points.

Takeaway. Our analysis validates that our planning mechanism is minimal and reliable. At inference time, the step ratio and confidence threshold provide tunable control of the speed–quality trade-off.

7 Conclusion

In this work, we introduce planned diffusion, a novel hybrid architecture combining autoregressive planning with parallel diffusion-based execution to improve the trade-off between latency and quality in text generation. Our model exploits opportunities for parallelism within the semantic structure of text. Experimental evaluation shows that planned diffusion expands the latency–quality Pareto frontier, achieving a significant speedup over pure autoregressive and diffusion baselines with a minimal drop in quality. This efficiency gain is driven by a shorter critical path and our approach's scalability with respect to training. We also show planned diffusion offers fine-grained control over the speed-quality trade-off at inference time. Planned diffusion provides a promising framework for developing faster and more efficient large language models.

ACKNOWLEDGEMENT

The authors acknowledge the MIT Office of Research Computing and Data for providing high performance computing resources that have contributed to the research results reported within this paper. This work was supported in part by the MIT-Google Program for Computing Innovation and by SRC JUMP 2.0 (CoCoSys). This material is based upon work supported in part by the National Science Foundation Graduate Research Fellowship under Grant No. 2141064. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. This work was funded in part by the DARPA ANSR, CODORD, and SAFRON programs under awards FA8750-23-2-0004, HR00112590089, and HR00112530141, NSF grant IIS1943641, and gifts from Adobe Research, Cisco Research, Qualcomm, and Amazon. This work was also supported by NSF CAREER Grant #2341040, Schmidt Sciences Early Career Fellowship, Okawa Foundation Research Award. Approved for public release; distribution is unlimited.

REPRODUCIBILITY STATEMENT

We specify the starting checkpoint, fine-tuning hyperparameters, inference settings, and hardware/software setup in Section 5. Section A includes a shortened version of the data-annotation prompt; we omit in-context examples for brevity.

LLM USE DISCLOSURE

We used LLMs to improve sentence-level writing and to search for related work.

REFERENCES

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

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

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.

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. In *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901, 2020.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating large language model decoding with speculative sampling, 2023. URL `https://arxiv.org/abs/2302.01318`.

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

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. Vicuna: An open-source

chatbot impressing gpt-4 with 90%* chatgpt quality, March 2023. URL https://lmsys.org/blog/2023-03-30-vicuna/.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023.

Yann Dubois, Balázs Galambosi, Percy Liang, and Tatsunori B Hashimoto. Length-controlled alpacaeval: A simple way to debias automatic evaluators. *arXiv preprint arXiv:2404.04475*, 2024.

Xinyang Geng, Arnav Gudibande, Hao Liu, Eric Wallace, Pieter Abbeel, Sergey Levine, and Dawn Song. Koala: A dialogue model for academic research. Blog post, April 2023. URL https://bair.berkeley.edu/blog/2023/04/03/koala/.

Feng Hong, Geng Yu, Yushi Ye, Haicheng Huang, Huangjie Zheng, Ya Zhang, Yanfeng Wang, and Jiangchao Yao. Wide-in, narrow-out: Revokable decoding for efficient and effective dllms. *arXiv preprint arXiv:2507.18578*, 2025.

Emiel Hoogeboom, Alexey A. Gritsenko, Jasmijn Bastings, Ben Poole, Rianne van den Berg, and Tim Salimans. Autoregressive diffusion models, 2022. URL https://arxiv.org/abs/2110.02037.

Zhanqiu Hu, Jian Meng, Yash Akhauri, Mohamed S Abdelfattah, Jae-sun Seo, Zhiru Zhang, and Udit Gupta. Accelerating diffusion language model inference via efficient kv caching and guided diffusion. *arXiv preprint arXiv:2505.21467*, 2025.

Sagar Imambi, Kolla Bhanu Prakash, and GR Kanagachidambaresan. Pytorch. In *Programming with TensorFlow: solution for edge computing applications*, pp. 87–104. Springer, 2021.

Daniel Israel, Guy Van den Broeck, and Aditya Grover. Accelerating diffusion llms via adaptive parallel decoding. *arXiv preprint arXiv:2506.00413*, 2025a.

Daniel Israel, Aditya Grover, and Guy Van den Broeck. Enabling autoregressive models to fill in masked tokens. *arXiv preprint arXiv:2502.06901*, 2025b.

Tian Jin, Ellie Y Cheng, Zachary Ankner, Nikunj Saunshi, Blake M Elias, Amir Yazdanbakhsh, Jonathan Ragan-Kelley, Suvinay Subramanian, and Michael Carbin. Learning to keep a promise: Scaling language model decoding parallelism with learned asynchronous decoding. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=ZfX43ZZRZR.

Jaeyeon Kim, Kulin Shah, Vasilis Kontonis, Sham Kakade, and Sitan Chen. Train for the worst, plan for the best: Understanding token ordering in masked diffusions, 2025. URL https://arxiv.org/abs/2502.06768.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. URL https://arxiv.org/abs/1412.6980.

Andreas Köpf, Yannic Kilcher, Dimitri Von Rütte, Sotiris Anagnostidis, Zhi Rui Tam, Keith Stevens, Abdullah Barhoum, Duc Nguyen, Oliver Stanley, Richárd Nagyfi, et al. Openassistant conversations-democratizing large language model alignment. *Advances in neural information processing systems*, 36: 47669–47681, 2023.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding, 2023. URL https://arxiv.org/abs/2211.17192.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. Solving quantitative reasoning problems with language models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 3843–3857, 2022.

Pengxiang Li, Yefan Zhou, Dilxat Muhtar, Lu Yin, Shilin Yan, Li Shen, Yi Liang, Soroush Vosoughi, and Shiwei Liu. Diffusion language models know the answer before decoding. *arXiv preprint arXiv:2508.19982*, 2025.

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Alpacaeval: An automatic evaluator of instruction-following models. https://github.com/tatsu-lab/alpaca_eval, 5 2023.

Wing Lian, Guan Wang, Bleys Goodson, Eugene Pentland, Austin Cook, Chanvichet Vong, and "Teknium". Slimorca: An open dataset of gpt-4 augmented flan reasoning traces, with verification, 2023. URL https://https://huggingface.co/Open-Orca/SlimOrca.

Anji Liu, Oliver Broadrick, Mathias Niepert, and Guy Van den Broeck. Discrete copula diffusion, 2025a. URL https://arxiv.org/abs/2410.01949.

Jiaming Liu, Hao Chen, Pengju An, Zhuoyang Liu, Renrui Zhang, Chenyang Gu, Xiaoqi Li, Ziyu Guo, Sixiang Chen, Mengzhen Liu, et al. Hybridvla: Collaborative diffusion and autoregression in a unified vision-language-action model. *arXiv preprint arXiv:2503.10631*, 2025b.

Mingdao Liu, Aohan Zeng, Bowen Wang, Peng Zhang, Jie Tang, and Yuxiao Dong. Apar: Llms can do auto-parallel auto-regressive decoding, 2024. URL https://arxiv.org/abs/2401.06761.

Sulin Liu, Juno Nam, Andrew Campbell, Hannes Stärk, Yilun Xu, Tommi Jaakkola, and Rafael Gómez-Bombarelli. Think while you generate: Discrete diffusion with planned denoising, 2025c. URL https://arxiv.org/abs/2410.06264.

Zhiyuan Liu, Yicun Yang, Yaojie Zhang, Junjie Chen, Chang Zou, Qingyuan Wei, Shaobo Wang, and Linfeng Zhang. dllm-cache: Accelerating diffusion large language models with adaptive caching. *arXiv preprint arXiv:2506.06295*, 2025d.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. URL https://arxiv.org/abs/1711.05101.

Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution, 2024. URL https://arxiv.org/abs/2310.16834.

Xinyin Ma, Runpeng Yu, Gongfan Fang, and Xinchao Wang. dkv-cache: The cache for diffusion language models. *arXiv preprint arXiv:2505.15781*, 2025.

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

Xuefei Ning, Zinan Lin, Zixuan Zhou, Zifu Wang, Huazhong Yang, and Yu Wang. Skeleton-of-thought: Prompting llms for efficient parallel generation, 2024. URL https://arxiv.org/abs/2307.15337.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pp. 27730–27744, 2022.

Jiayi Pan, Xiuyu Li, Long Lian, Charlie Snell, Yifei Zhou, Adam Yala, Trevor Darrell, Kurt Keutzer, and Alane Suhr. Learning adaptive parallel reasoning with language models, 2025. URL https://arxiv.org/abs/2504.15466.

Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. Efficiently scaling transformer inference. *Proceedings of machine learning and systems*, 5:606–624, 2023.

Mihir Prabhudesai, Mengning Wu, Amir Zadeh, Katerina Fragkiadaki, and Deepak Pathak. Diffusion beats autoregressive in data-constrained settings, 2025. URL https://arxiv.org/abs/2507.15857.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

Gleb Rodionov, Roman Garipov, Alina Shutova, George Yakushev, Erik Schultheis, Vage Egiazarian, Anton Sinitsin, Denis Kuznedelev, and Dan Alistarh. Hogwild! inference: Parallel llm generation via concurrent attention, 2025. URL https://arxiv.org/abs/2504.06261.

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

Subham Sekhar Sahoo, Zhihan Yang, Yash Akhauri, Johnna Liu, Deepansha Singh, Zhoujun Cheng, Zhengzhong Liu, Eric Xing, John Thickstun, and Arash Vahdat. Esoteric language models. *arXiv preprint arXiv:2506.01928*, 2025.

Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis Titsias. Simplified and generalized masked diffusion for discrete data. *Advances in neural information processing systems*, 37:103131–103167, 2024.

Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis K. Titsias. Simplified and generalized masked diffusion for discrete data, 2025. URL https://arxiv.org/abs/2406.04329.

Mitchell Stern, William Chan, Jamie Kiros, and Jakob Uszkoreit. Insertion transformer: Flexible sequence generation via insertion operations, 2019. URL https://arxiv.org/abs/1902.03249.

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.

Gilad Turok, Chris De Sa, and Volodymyr Kuleshov. Duel: Exact likelihood for masked diffusion via deterministic unmasking. *arXiv preprint arXiv:2603.01367*, 2026.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: long papers)*, pp. 13484–13508, 2023.

Hao Wen, Yifan Su, Feifei Zhang, Yunxin Liu, Yunhao Liu, Ya-Qin Zhang, and Yuanchun Li. Parathinker: Native parallel thinking as a new paradigm to scale llm test-time compute, 2025. URL https://arxiv.org/abs/2509.04475.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Huggingface's transformers: State-of-the-art natural language processing, 2020. URL https://arxiv.org/abs/1910.03771.

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*, 2025.

Xinyu Yang, Yuwei An, Hongyi Liu, Tianqi Chen, and Beidi Chen. Multiverse: Your language models secretly decide how to parallelize and merge generation, 2025. URL https://arxiv.org/abs/2506.09991.

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.

Jun Zhang, Jue Wang, Huan Li, Lidan Shou, Ke Chen, Gang Chen, and Sharad Mehrotra. Draft&; verify: Lossless large language model acceleration via self-speculative decoding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11263–11282. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.607. URL http://dx.doi.org/10.18653/v1/2024.acl-long.607.

Chuyang Zhao, Yuxing Song, Wenhao Wang, Haocheng Feng, Errui Ding, Yifan Sun, Xinyan Xiao, and Jingdong Wang. Monoformer: One transformer for both diffusion and autoregression. *arXiv preprint arXiv:2409.16280*, 2024.

A Data Annotation Prompt

We present a shortened version of our data-annotation prompt below. We use it to instruct Gemini Flash 2.0 (temperature = 1.0, top-p = 0.95) to annotate our training data.

```
You will first identify whether the given chatbot response may be generated in
parallel. You are to then annotate the chatbot response using specific tags
that highlight segments suitable for parallel generation.

Use <async> tags to denote segments of text that may be generated asynchronously
in parallel with respect to the text that follows. Thus apply <async> tags only
to sentences that do not serve as necessary context for subsequent sentences.
Sentences that are crucial for understanding or generating following text are
not suitable for parallel asynchronous generation. For each <async> tag, include
a very concise topic description of the text surrounded within the <async> tags.
The topic description will be accessible to text generation after the closing
async tag to ensure continuity and coherence.

Use the singleton <sync/> tag for synchronization. All content generated before
<sync/>, including text marked by <async> is accessible to subsequent text
generation after the <sync/> tag, ensuring continuity and coherence.

Detailed Instructions:
- Tagging Rules:
 - Use <async> tag in pairs.
 - Ensure all text content is encapsulated within <async> tags.
 - Ensure that each <async> tag encompasses at least five words.
 - Refrain from altering the content of the response during annotation.
 - Use a maximum of 3 words in the topic description.
 - Use <sync/> sparingly as it introduces significant slowdown.
```

B Table Summary of Evaluations

For completeness, we provide a table summary of the results in Figure 2.

Table 1: Comparison of methods based on latency and length-controlled win rate.

Method	**Latency (s)**	**LC Win Rate (%)**
Diffusion	154.124	52.59%
AR	6.272	50.00%
Fast-dLLM Wu et al. (2025)	77.723	40.20%
Pasta-SFT (Jin et al., 2025)	5.740	26.13%
Skeleton-of-thought (SoT) (Ning et al., 2024)	3.101	40.72%
Planned Diffusion (PD)	3.471	44.59%
Planned Diffusion Dense Attention (PD-DA)[DI:TODO FIX]	4.934	49.17%

C Full Speedup Distribution

This section details the performance variance observed across the evaluation dataset. Figure 4 illustrates the distribution of speedup ratios obtained by comparing planned diffusion against the autoregressive baseline. The dataset consists of $N = 803$ test samples. The analysis yields a mean speedup ratio of 3.08 and the distribution exhibits a standard deviation of 4.10. This significant variance relative to the mean suggests a right-skewed distribution, indicating that while the average performance gain is substantial, there exists a subset of cases where the proposed method achieves exceptionally high speedups.

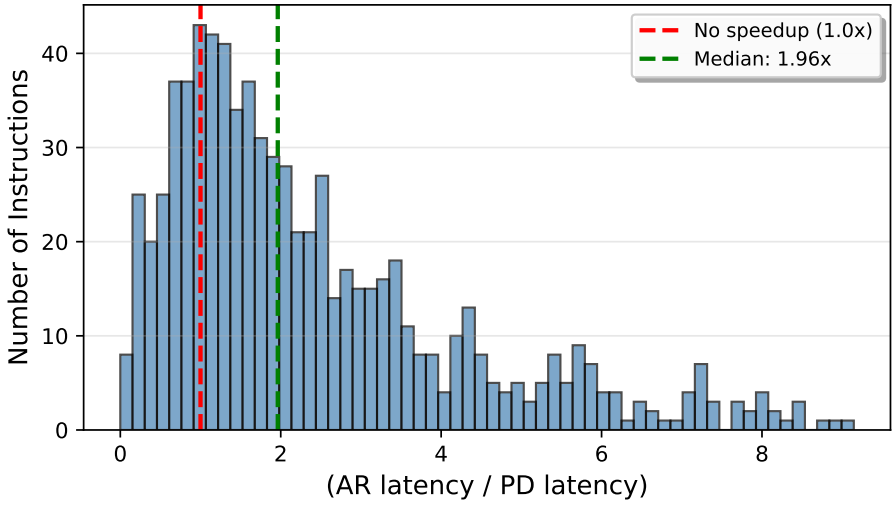

Figure 4: Distribution of Speedup Ratios.

D KV Caching in Planned Diffusion

KV caching plays a substantial role in the efficiency of planned diffusion. The KV cache of planned diffusion is derived from the model's hybrid attention mask. The general principle is that a token can be cached if its key and value embeddings are unaffected by future tokens. This is determined by whether a token attends to future positions in the sequence. The autoregressive planning stage uses causal attention and a conventional application of KV caching (Pope et al., 2023). In contrast, the diffusion stage employs a bidirectional mask, in which tokens inside an <async> chunk attend to each other. Because bidirectional attention does not support KV caching (Israel et al., 2025b), tokens inside an <async> chunk cannot be cached until their respective denoising process is complete. However, subsequent tokens, such as those in a new planning stage following a <sync/> tag, can efficiently attend to the KV cache of the preceding planning and diffusion stages. This caching mechanism is essential for combining the speed of diffusion with the computational savings of autoregressive KV caching.

E Further Discussion of Hybrid Models

When referring to hybrid models between autoregression and discrete diffusion, we define a hybrid model as a single architecture, capable of pure left-to-right autoregression and pure discrete diffusion. This definition

precludes existing works that interpolate between the autoregressive and discrete diffusion objective, such as Block Diffusion (Arriola et al., 2025), Eso-LM (Sahoo et al., 2025), or ARDMs Hoogeboom et al. (2022). Another necessary clarification is that discrete diffusion should not be confused with vanilla continuous diffusion. For continuous diffusion, multimodal hybrids over autoregrssion and diffusion exist, such as HybridVLA (Liu et al., 2025b) and Monoformer Zhao et al. (2024).

F Dataset Performance Breakdown

The AlpacaEval dataset Li et al. (2023) is composed of 5 datasets: self-instruct (Wang et al., 2023), open-assistant (Köpf et al., 2023), vicuna (Chiang et al., 2023), koala (Geng et al., 2023), and hh-rlhf (Bai et al., 2022). For planned diffusion, we report the length-controlled win rate for each and each standard error respectively. We observe that win rate is not especially sensitive to the dataset source of the instruction.

Table 2: Planned diffusion LC Win Rates across different datasets

Dataset	**Number of Samples**	**LC Win Rate (% $\pm$ SE)**
self-instruct	249	45.7 ± 0.65
open-assistant	187	44.2 ± 0.91
vicuna	80	44.2 ± 1.66
koala	156	48.0 ± 0.57
hh-rlhf	129	46.3 ± 1.04

G Qualitative Analysis

To better understand the behavior of Planned Diffusion, we analyze specific examples of generated outputs, focusing on the interaction between the autoregressive planning stage and the parallel diffusion stage. The examples below display the instruction and the raw output, which reveals the internal control tags generated by the planner.

In the raw output:

- `<promise>-<topic>... length</topic>` indicates the autoregressive plan for a specific span.
- `<async>...</async>` indicates the text generated in parallel via discrete diffusion.
- `[PAD]` tokens indicate the unused capacity of the allocated span length.

G.1 Successful Example

The following example demonstrates a successful decomposition of a list-based query. The instruction asks for popular careers in development studies. The autoregressive planner successfully identifies distinct semantic entities before generation begins.

The planner generates specific, discriminatory topics for each span: `UN orgs`, `WB orgs`, `NGOs`, `media orgs`, etc. Because the planning stage explicitly disambiguates the content of each span, the subsequent diffusion execution stage, which generates these spans in parallel with independent attention masks, does not suffer from mode collapse. Each diffusion process is conditioned on a unique semantic target, resulting in a comprehensive, non-repetitive list that correctly follows the instruction.

Instruction

Please answer the following question based on the information provided in the article.

Development studies is an interdisciplinary branch of social science. Development studies is offered as a specialized master's degree in a number of reputed universities around the world. It has grown in popularity as a subject of study since the early 1990s, and has been most widely taught and researched in developing countries and countries with a colonial history, such as the UK, where the discipline originated. Students of development studies often choose careers in international organisations such as the United Nations, World Bank, non-governmental organisations (NGOs), media and journalism houses, private sector development consultancy firms, corporate social responsibility (CSR) bodies and research centers.

What are some popular careers for students of development studies

Raw Output

<promise>-<topic>introduction 2</topic>
<async>Some popular careers for students of development studies include:[PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD]</async><sync/>
<promise>-<topic>UN orgs 4</topic>
<promise>-<topic>WB orgs 4</topic>
<promise>-<topic>NGOs 4</topic>
<promise>-<topic>media orgs 4</topic>
<promise>-<topic>development consultancy 4</topic>
<promise>-<topic>CSR bodies 4</topic>
<promise>-<topic>research centers 4</topic>
<async>1. International Organizations: Students often work for organizations such as the United Nations (UN) to address global issues, including poverty, education, healthcare, and education.[PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD]</async>
<async>2. World Bank (WB): Similar to the UN, the World Bank offers opportunities to work on projects related to development, finance, and infrastructure in various countries.[PAD][PAD][PAD][PAD][PAD][PAD][PAD]</async>
<async>3. Non-governmental Organizations (NGOs): Many students of development studies pursue careers in NGOs, as these organizations are often involved in development projects and initiatives.[PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD]</async>
<async>4. Media and Journalism Houses: Students may become reporters and journalists, covering topics related to global issues, development, and the work of various organizations.[PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD]</async>
<async>5. Private sector development consultancy firms: Companies that provide advice and support to local businesses, helping them improve their operations, strategies, and adapt to the market.[PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD]</async>
<async>6. Corporate Social Responsibility (CSR) bodies: Students can work in these bodies to help businesses and organizations understand and meet their social responsibilities and promote sustainable practices.[PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD]</async>
<async>7. Research Centers: Students may also work in research centers, where they can contribute to the advancement of development studies research, analysis, and presentation.[PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD]</async><sync/>
<promise>-<topic>summary 4</topic>
<async>These are just a few examples of the many careers available to students of development studies. The field is diverse and offers opportunities to contribute to the global advancement of development and social change.[PAD][PAD][PAD][PAD]</async>

Clean Output

Some popular careers for students of development studies include:
1. International Organizations: Students often work for organizations such as the United Nations (UN) to address global issues, including poverty, education, healthcare, and education.
2. World Bank (WB): Similar to the UN, the World Bank offers opportunities to work on projects related to development, finance, and infrastructure in various countries.
3. Non-governmental Organizations (NGOs): Many students of development studies pursue careers in NGOs, as these organizations are often involved in development projects and initiatives.
4. Media and Journalism Houses: Students may become reporters and journalists, covering topics related to global issues, development, and the work of various organizations.
5. Private sector development consultancy firms: Companies that provide advice and support to local businesses, helping them improve their operations, strategies, and adapt to the market.
6. Corporate Social Responsibility (CSR) bodies: Students can work in these bodies to help businesses and organizations understand and meet their social responsibilities and promote sustainable practices.
7. Research Centers: Students may also work in research centers, where they can contribute to the advancement of development studies research, analysis, and presentation. These are just a few examples of the many careers available to students of development studies. The field is diverse and offers opportunities to contribute to the global advancement of development and social change.

G.2 Failed Example

The following example illustrates a failure mode that can result from poor planning. In this case, the autoregressive planner fails to decompose the semantic content effectively. Instead of generating specific locations (e.g., "NYC", "Denver", "LA"), the planner generates generic structural tags: `list part 1`, `list part 2`, `list part 3`.

Instruction

Make a list of the top places in the U.S. to visit at the given time.

November

Raw Output

<promise>-<topic>introduction 2</topic>
<promise>-<topic>list prompt 1</topic>
<async>Sure, here's a list of the top places in the U.S. to visit in November:</async>
<async>Here are the top places to visit:[PAD][PAD]</async><sync/>
<promise>-<topic>list part 1 4</topic>
<promise>-<topic>list part 2 3</topic>
<promise>-<topic>list part 3 3</topic>
<promise>-<topic>list part 4 3</topic>
<promise>-<topic>list part 5 3</topic>
<promise>-<topic>list part 6 3</topic>
<promise>-<topic>list part 7 3</topic>
<promise>-<topic>list part 8 3</topic>
<promise>-<topic>list part 9 3</topic>
<promise>-<topic>list part 10 3</topic>
<async>1. New York City[PAD][PAD][PAD][PAD][PAD]</async>
<async>2. New York City - Lots of shopping, museums, and entertainment.[PAD][PAD][PAD][PAD][PAD]</async>
<async>2. New York City - It's a big city with lots of fun things, shops, and places to eat.[PAD][PAD][PAD][PAD][PAD][PAD]</async><async>3. New York City - New York City is a big city with lots of things to see, like the Statue of Liberty, the park, and the buildings.[PAD][PAD][PAD][PAD][PAD][PAD][PAD]</async>
<async>4. Denver, Colorado - A great place for football, hiking, and skiing. It has the big city of Denver, the state of Colorado, and the beautiful mountains around it. It's also famous for the Denver Zoo.[PAD][PAD][PAD][PAD]</async>
<async>5. Los Angeles, California - Known for its beautiful coastline, Hollywood, and shopping, Los Angeles is a top place to visit in November. Visit places like Santa Monica, Santa Barbara, and Santa Cruz, and enjoy the local food, beaches, and culture.[PAD][PAD][PAD][PAD][PAD][PAD][PAD]</async>
<async>8. Sacramento, California - Known for its rich history and beautiful coastline, Sacramento is a major city in the state of California. It is the capital city of the United States and is home to many major events and festivals, such as the Sacramento Street Festival, the Sacramento Zoo, and the Sacramento Riverwalk.[PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD]</async>
<async>10. Denver, Colorado
- Denver is known for its beautiful landscapes, history, and art scene. It's one of the most popular cities in the U.S. and has a lot to do and see. Visit the U.S. Space Center, the Denver Zoo, and the University of Colorado Boulder to get a taste of the city.[PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD]</async>
<async>10. Denver, Colorado
- Denver is a major city in the U.S. with diverse history, culture, and landscapes. It is home to numerous attractions, such as the Denver Zoo, Science Center, andUSS, as well as the University of Colorado Boulder and the Denver Zoo. The city is also known for its mountains, forests, and parks, making it a great place for outdoor activities.[PAD][PAD][PAD][PAD][PAD][PAD][PAD]</async>
<async>10. Denver, Colorado
- Denver is a major city in the U.S. and offers a mix of history, culture, and beauty. Some of the top places to visit in Denver include the U.S. Open Center, Central Park, the Denver Zoo, and the Denver Nature Center. The University of Colorado, also located in Denver, is a great place to explore, and the Denver Art Museum offers a unique glimpse into the city.[PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD]</async>

Clean Output

Sure, here's a list of the top places in the U.S. to visit in November: Here are the top places to visit:
1. New York City
2. New York City - Lots of shopping, museums, and entertainment.
2. New York City - It's a big city with lots of fun things, shops, and places to eat.
3. New York City - New York City is a big city with lots of things to see, like the Statue of Liberty, the park, and the buildings.
4. Denver, Colorado - A great place for football, hiking, and skiing. It has the big city of Denver, the state of Colorado, and the beautiful mountains around it. It's also famous for the Denver Zoo.
5. Los Angeles, California - Known for its beautiful coastline, Hollywood, and shopping, Los Angeles is a top place to visit in November. Visit places like Santa Monica, Santa Barbara, and Santa Cruz, and enjoy the local food, beaches, and culture.
8. Sacramento, California - Known for its rich history and beautiful coastline, Sacramento is a major city in the state of California. It is the capital city of the United States and is home to many major events and festivals, such as the Sacramento Street Festival, the Sacramento Zoo, and the Sacramento Riverwalk.
10. Denver, Colorado
- Denver is known for its beautiful landscapes, history, and art scene. It's one of the most popular cities in the U.S. and has a lot to do and see. Visit the U.S. Space Center, the Denver Zoo, and the University of Colorado Boulder to get a taste of the city.
10. Denver, Colorado
- Denver is a major city in the U.S. with diverse history, culture, and landscapes. It is home to numerous attractions, such as the Denver Zoo, Science Center, andUSS, as well as the University of Colorado Boulder and the Denver Zoo. The city is also known for its mountains, forests, and parks, making it a great place for outdoor activities.
10. Denver, Colorado
- Denver is a major city in the U.S. and offers a mix of history, culture, and beauty. Some of the top places to visit in Denver include the U.S. Open Center, Central Park, the Denver Zoo, and the Denver Nature Center. The University of Colorado, also located in Denver, is a great place to explore, and the Denver Art Museum offers a unique glimpse into the city.

H PLANNED DIFFUSION ATTENTION MASKS

The attention mask for planned diffusion combines causal and bidirectional attention. Causal attention is used for sequential planning stages. Bidirectional attention is used within <async> chunks for parallel denoising. We enable concurrent chunks to cross-attend, unlike the sparse attention variant. After a <sync/> token, subsequent tokens can to attend to all prior tokens. For illustrative purposes, this example shortens the diffusion chunks.

In planned diffusion sparse attention (SPDA), the asynchronous blocks cannot attend to on another. Sparsity enables more compute efficient training and inference.

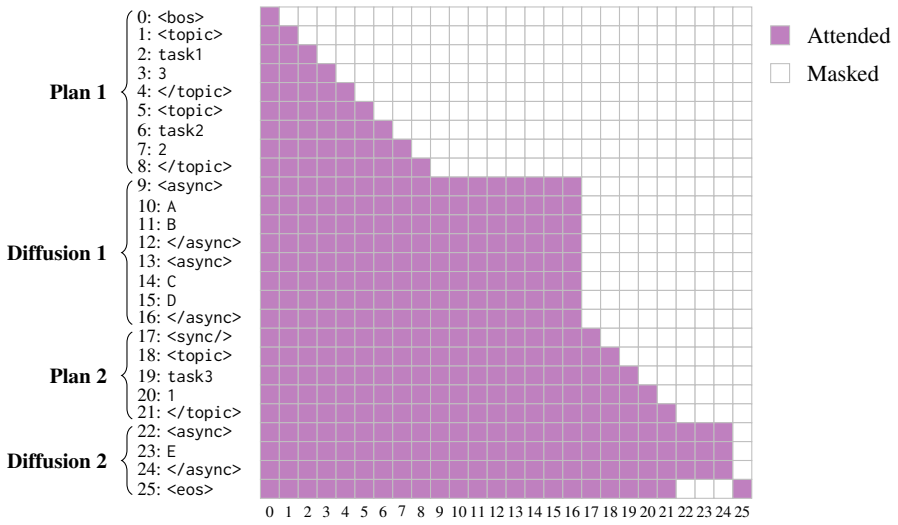

Figure 5: Attention mask for planned diffusion.

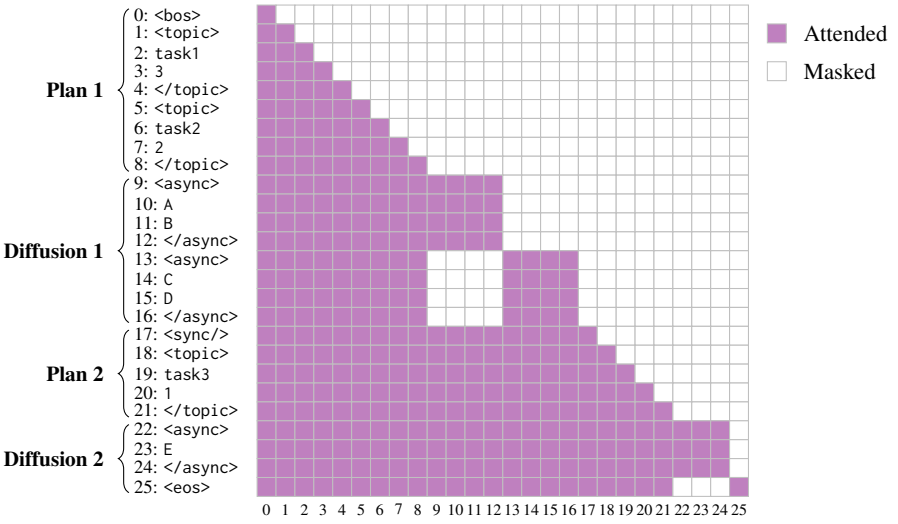

Figure 6: Attention mask for planned diffusion with sparse attention (PDSA).