Orthrus: Memory-Efficient Parallel Token Generation via Dual-View Diffusion

Chien Van Nguyen[†]
University of Oregon

Chaitra Hegde
Google DeepMind

Van Cuong Pham
University of Oregon

Ryan A. Rossi
Adobe Research

Franck Dernoncourt
Adobe Research

Thien Huu Nguyen
University of Oregon

Abstract

We introduce **Orthrus**, a simple and efficient dual-architecture framework that unifies the exact generation fidelity of autoregressive Large Language Models (LLMs) with the high-speed parallel token generation of diffusion models. The sequential nature of autoregressive decoding represents a fundamental bottleneck for high-throughput inference. While diffusion language models attempt to overcome this limitation through parallel generation, they often suffer from significant performance degradation, substantial training costs, and a lack of exact distributional guarantees. Orthrus resolves this dichotomy natively. Designed to seamlessly integrate into existing Transformers, the framework augments a frozen LLM with a lightweight, trainable module to create a parallel diffusion view alongside the standard autoregressive view. In this unified system, both views attend to the same high-fidelity Key-Value (KV) cache; the autoregressive head executes context pre-filling to construct accurate KV representations, while the diffusion head executes parallel generation. By employing an exact consensus mechanism between the two views, Orthrus guarantees lossless inference, delivering up to a $7.8\times$ speedup with only an $O(1)$ memory cache overhead and minimal parameter additions. We release the code at https://github.com/chiennv2000/orthrus.

1 Introduction

Autoregressive (AR) Large Language Models (LLMs) are currently the predominant architecture in natural language processing, demonstrating robust performance across a diverse set of complex reasoning and generation tasks (Radford et al., 2019; Brown et al., 2020; Radford et al., 2018; Touvron et al., 2023; Achiam et al., 2023; Guo et al., 2025). However, AR models suffer from a fundamental inefficiency during the decoding phase. While the pre-filling stage processes prompt tokens in parallel by leveraging self-attention, the generation phase computes tokens strictly sequentially. This one-by-one generation creates a memory-bandwidth bottleneck, leading to hardware underutilization and high inference latency.

Diffusion Language Models (DLMs) (Nie et al., 2025; Arriola et al., 2025; Zhu et al., 2025; Ye et al., 2025a) natively bypass this bottleneck by generating blocks of tokens in parallel. Despite providing significant inference speedups, DLMs consistently underperform AR models of a similar scale and require massive training datasets to achieve baseline coherence. Recent approaches attempt to adapt pre-trained AR models into diffusion models to bridge this quality gap (Hu et al., 2024; Wu et al., 2025; Liu et al., 2026; Fu et al., 2026). However, these adaptations remain computationally expensive, often requiring continued pre-training on hundreds of billions to trillions of tokens, and still fail to match the exact predictive distribution of the original AR models due to architectural divergence.

[†]Email: chienn@uoregon.edu

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

To overcome this dichotomy, we propose resolving the trade-off at the fundamental architectural level by *unifying the strengths of both paradigms within a single Transformer*. We introduce Orthrus, a novel dual-architecture framework designed to natively support parallel generation without sacrificing the exact predictive distribution of the base autoregressive model. The core architectural insight of Orthrus is that the AR bottleneck is strictly confined to the generation phase; its self-attention mechanism remains optimal for building context representations. Consequently, Orthrus freezes the pre-trained AR model and utilizes its standard forward pass exclusively during the pre-filling stage to compute a high-fidelity Key-Value (KV) cache. To enable high-speed parallel generation, we augment the network by integrating a lightweight, trainable diffusion module directly alongside the AR attention heads.

This structural unification allows both views to operate over the exact same context, resulting in zero redundant cache overhead. During generation, the diffusion head conditions directly on the high-quality KV cache constructed by the AR head to generate multiple future tokens in parallel. To guarantee lossless inference, Orthrus incorporates an intra-model consensus mechanism that validates the proposed tokens against the frozen AR model, ensuring that the final output exactly matches its predictive distribution. Beyond inference acceleration, Orthrus is designed to provide robust generation efficiency across diverse tasks and varying context lengths, addressing an important limitation of traditional speculative decoding methods whose acceptance lengths can vary substantially across different generation settings. By decoupling the parallel generation mechanism from the sequential constraints of the base model, Orthrus achieves lossless inference acceleration while maintaining robust performance across diverse tasks and varying context lengths, surpassing existing diffusion language models in generation fidelity and state-of-the-art speculative decoding methods in inference efficiency.

In summary, our main contributions are:

- **A Novel Dual-Architecture Framework:** We introduce Orthrus, a unified architecture that integrates parallel diffusion generation into a standard AR Transformer. By sharing a single KV cache and leveraging intra-model consensus, Orthrus preserves the exact predictive distribution of the base LLM while eliminating redundant historical KV cache storage.
- **Significant Inference Acceleration:** Orthrus enables high-speed parallel token generation without sacrificing generation quality, achieving up to a $7.8\times$ speedup over standard AR decoding across diverse reasoning and code generation tasks.
- **Robustness Across Tasks and Context Lengths:** Orthrus maintains strong and consistent generation efficiency across diverse tasks and varying context lengths, outperforming traditional speculative decoding methods in average acceptance length.
- **Parameter- and Training-Efficient Adaptation:** Orthrus introduces parallel generation capabilities by fine-tuning only 16% of the model parameters using fewer than 1B training tokens, requiring under 24 hours on a single 8xH200 GPU node.

2 Preliminaries

To contextualize the architectural design of our proposed framework, we formalize the distinct probability modeling paradigms of Autoregressive (AR) and Masked Diffusion Language Models (MDMs). This formulation isolates the mathematical trade-off between generation quality and inference speed, establishing the foundation for our structural unification.

2.1 Autoregressive and Diffusion Paradigms

Autoregressive Language Modeling. AR models learn the true data distribution by factorizing the joint probability of a sequence $\mathbf{x} = (x_1, x_2, \ldots, x_N)$ using the exact chain rule of probability $p_{\text{AR}}(\mathbf{x}) = \prod_{i=1}^{N} p_\theta(x_i \mid \mathbf{x}_{<i})$. The model parameters θ are typically optimized via the negative log-likelihood over the data distribution $\mathcal{D}$:

$$\mathcal{L}_{\text{AR}}(\theta) = -\mathbb{E}_{\mathbf{x}\sim\mathcal{D}}\left[\sum_{i=1}^{N} \log p_\theta(x_i \mid \mathbf{x}_{<i})\right] \tag{1}$$

By imposing no conditional independence assumptions, this formulation ensures each token x_i is strictly conditioned on the entire preceding trajectory. While this causal dependency achieves state-of-the-art fidelity, it mandates sequential sampling. During inference, generating N tokens requires N distinct forward passes, repeatedly loading the Key-Value (KV) cache creating a fundamental, memory-bandwidth-bound bottleneck (Leviathan et al., 2022; Adnan et al., 2024; Ho et al., 2024).

Masked Diffusion Language Models. Diffusion Language Models (DLMs) bypass the sequential bottleneck by framing generation as a parallel denoising process. Given a historical context $\mathbf{c} = \mathbf{x}_{\leq t}$ and a corrupted block of future tokens $\mathbf{y}^t$, the reverse process trains a network parameterized by ϕ to predict the original tokens $\mathbf{y}^0$ simultaneously:

$$\mathcal{L}_{\text{MDM}}(\phi) = -\mathbb{E}_{\mathbf{x}\sim\mathcal{D},t,\mathbf{y}^t}\left[\sum_{k\in\mathcal{M}} \log p_\phi(y_k^0 \mid \mathbf{c}, \mathbf{y}^t)\right] \tag{2}$$

where $\mathcal{M}$ is the set of masked indices. For highly accelerated inference (where denoising steps $T \ll |\mathcal{M}|$), the model relies on a strong conditional independence assumption:

$$p_{\text{DLM}}(\mathbf{y}^0 \mid \mathbf{c}) \approx \prod_{k\in\mathcal{M}} p_\phi(y_k^0 \mid \mathbf{c}, \mathbf{y}^t) \tag{3}$$

While this formulation heavily amortizes memory-bandwidth costs by computing the entire block in a single forward pass, it inherently violates the strict causal dependency of the autoregressive model. Because the prediction of token y_k does not condition on the exact, realized token y_{k-1}, the joint probability distribution modeled by the DLM drifts from the true AR target distribution (Ma et al., 2025; Chen et al., 2025; Liu et al., 2026; Wu et al., 2025).

2.2 The Limits of Adaptation and Structural Unification

To mitigate the high computational costs of training DLMs from scratch, recent works explore adapting pre-trained AR models into diffusion frameworks (Tian et al., 2025; Gat et al., 2025; Wu et al., 2025; Cheng et al.; Liu et al., 2026; Zhou et al., 2026). These approaches repurpose the robust representations of AR baselines by fine-tuning them on block-wise masked diffusion objectives (Equation 2). While these methods transition the model from sequential to parallel generation, adaptation fundamentally alters the base model, introducing severe performance trade-offs. This distributional drift is particularly catastrophic for reasoning-heavy tasks: during long-horizon generation, conditional errors compound rapidly, causing performance degradation. For instance, state-of-the-art adaptations like Fast-dLLM-v2 (Wu et al., 2025) suffers an 11-point accuracy drop on MATH-500 (Hendrycks et al., 2020) relative to its AR baseline. Furthermore, because these adapted models typically rely on multiple iterative filtering steps during inference to recover coherence, they often negate the theoretical speed advantages of parallel decoding, resulting in marginal latency improvements. By modifying the base weights and discarding the strict sequential forward pass, adapted models lose the ability to recover the exact predictive distribution of the original models.

The mathematical dichotomy establishes that exact causal conditioning ensures high fidelity but forces sequential computation, while conditional independence (Eq. 3) enables parallelism at the cost of distributional drift. We resolve this tension by structurally unifying both paradigms at the attention level. Rather than permanently converting the base model, Orthrus decouples parallel generation from sequential constraints by grounding it within the frozen, high-fidelity representations of the AR baseline. We detail this dual-architecture design in Section 3.

3 Methodology: The Orthrus Architecture

The design of Orthrus is rooted in a fundamental architectural trade-off: standard autoregressive (AR) models produce high-fidelity representations due to their strict causal conditioning, yet are bottlenecked by sequential generation. Conversely, parallel diffusion generation offers rapid decoding but often suffer from conditional drift and lower representation quality. To reconcile this trade-off, Orthrus introduces a unified dual-view architecture. By injecting a lightweight diffusion head into a pre-trained AR model, we preserve its exact representation space while enforcing a strict functional decoupling: the frozen AR head is dedicated exclusively to constructing high-fidelity context representations, and the trainable diffusion head is specialized for high-speed parallel generation.

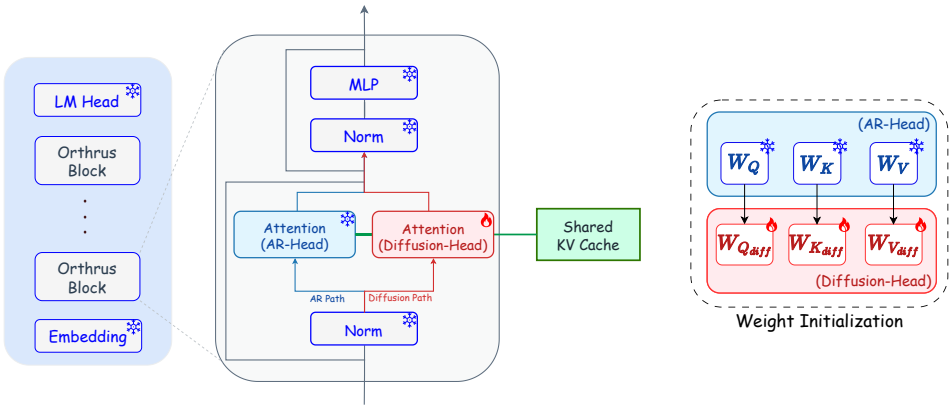

Figure 1: **The Orthrus dual-view architecture.** Each Orthrus block features two distinct, parallel attention paths: a frozen AR head (blue) and a trainable diffusion head (red). The frozen AR head is used to encode context into KV representations, while the diffusion head enables parallel token generation. Both paths seamlessly attend over this single shared cache.

3.1 Unified Dual-View Attention Mechanism

Consider a prompt sequence $\mathbf{x}_{1:t} = (x_1, \ldots, x_t)$. During prefilling, the frozen AR backbone $\mathcal{M}_{\text{AR}}$ processes the full context in **a single forward pass**, producing causal Key-Value representations $(\mathbf{K}_{\text{AR}}, \mathbf{V}_{\text{AR}})$. At generation time, however, producing K continuation tokens requires K **sequential forward passes**, each conditioned on all prior KV states, a fundamental memory-bandwidth bottleneck that our architecture is designed to eliminate.

Parallel Diffusion View. We augment each transformer layer with a trainable diffusion attention module, parameterized by projection matrices $(\mathbf{W}_{Q_{\text{diff}}}, \mathbf{W}_{K_{\text{diff}}}, \mathbf{W}_{V_{\text{diff}}})$ initialized from their frozen AR counterparts, as illustrated in Figure 1. To generate K tokens in a single forward pass, we construct an extended sequence by concatenating the first token decoded by the AR view with $K-1$ `<mask>` embeddings, forming a parallel block of K positions. These positions are processed simultaneously through the diffusion view, whose queries attend jointly over the frozen AR cache and the bidirectional self-representations of the mask block:

$$\mathbf{O}_{\text{diff}} = \text{Softmax}\left(\frac{\mathbf{Q}_{\text{diff}}\,[\mathbf{K}_{\text{AR}} \,\|\, \mathbf{K}_{\text{diff}}]^\top}{\sqrt{d_{\text{head}}}}\right)[\mathbf{V}_{\text{AR}} \,\|\, \mathbf{V}_{\text{diff}}], \tag{4}$$

where $[\cdot\|\cdot]$ denotes concatenation along the sequence axis and $\mathbf{O}_{\text{diff}} \in \mathbb{R}^{K \times d_{\text{head}}}$ contains the hidden states for all K parallel positions. Two structural properties follow directly. Because $(\mathbf{K}_{\text{AR}}, \mathbf{V}_{\text{AR}})$ are reused in-place from the prefill pass, so the diffusion view introduces **zero additional historical KV cache memory**. Since only $(\mathbf{W}_{Q_{\text{diff}}}, \mathbf{W}_{K_{\text{diff}}}, \mathbf{W}_{V_{\text{diff}}})$ are updated during training, the total number of trainable parameters is approximately 16% of the full model.

3.2 Training: Dual-Pass Block Masking

Because the AR backbone is strictly frozen, training reduces to aligning the diffusion view's parallel predictions with the AR model's exact target distribution. Given a sequence $\mathbf{x} = (x_1, \ldots, x_L)$, we sample B random anchor positions $\{a_b\}_{b=1}^{B}$ and extract contiguous blocks of length K, forming clean blocks $\mathbf{y}_b = (x_{a_b}, \ldots, x_{a_b+K-1})$. Each block is corrupted by retaining the first token as a visible anchor and replacing the remaining $K-1$ positions with `<mask>` tokens:

$$\tilde{y}_{b,k} = \begin{cases} x_{a_b} & k = 1 \\ \texttt{<mask>} & k = 2, \ldots, K \end{cases} \tag{5}$$

The B corrupted blocks are concatenated and processed against the frozen AR KV cache $(\mathbf{K}_{\text{AR}}, \mathbf{V}_{\text{AR}})$ computed over the full sequence.

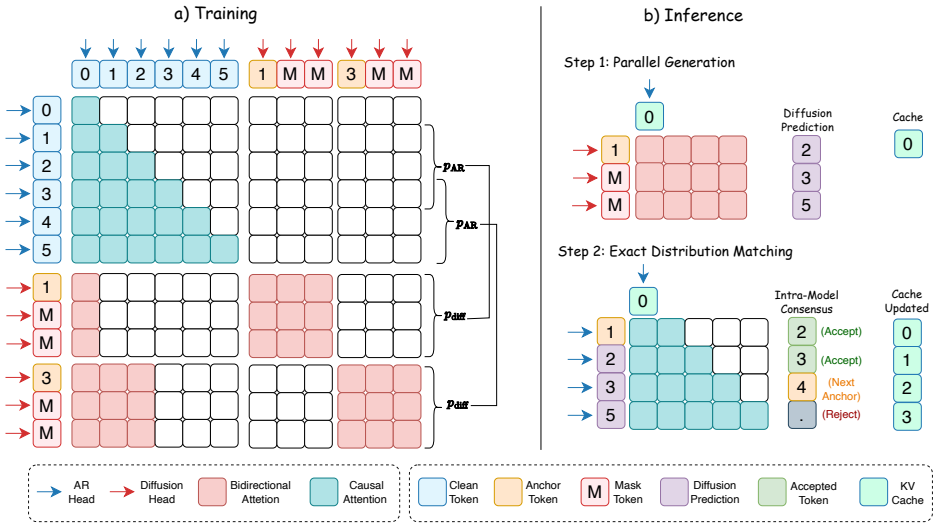

Figure 2: **The Orthrus dual-view attention mechanism. (a) Training:** The AR path (blue arrows) processes the **clean context** using standard causal masking to establish the exact target distribution. The diffusion path (red arrows) processes **corrupted parallel blocks** (an anchor plus `<mask>` tokens). The diffusion head attends directly to the KV representations constructed by the AR path, and its parallel predictions (p_{diff}) are distilled to match the exact corresponding AR rows (p_{AR}). **(b) Inference:** The diffusion head projects K candidate tokens in parallel (Step 1), which the AR head validates in a single pass (Step 2). Accepted tokens' KV states are seamlessly appended to the shared cache.

Dual-pass attention mask for the diffusion view. While the frozen AR path processes the clean historical context utilizing standard causal masking (top rows of Figure 2(a), denoted by blue arrows), the trainable diffusion head processes the **corrupted parallel blocks** and requires a specialized routing mechanism to prevent data leakage. To enforce this correct information flow during training, we construct a structured block mask $\mathbf{M}_{\text{diff}}$ for the diffusion view (represented by the bottom rows and red arrows) implemented using FlexAttention (Dong et al., 2024). For a diffusion query at position q and a key at position k, attention is permitted if and only if:

$$\mathbf{M}_{\text{diff}}[q,k] = \underbrace{\mathbf{1}[k < L] \cdot \mathbf{1}[k \leq a_b - 1]}_{\text{causal AR context}} \;\mid\; \underbrace{\mathbf{1}[k \geq L] \cdot \mathbf{1}[\lfloor q/K \rfloor = \lfloor (k-L)/K \rfloor]}_{\text{bidirectional within block}}, \tag{6}$$

This specialized mask enforces two disjoint viewing rules: (i) each position within the corrupted block attends causally to the clean AR context preceding its block anchor, preventing future leakage; and (ii) all positions within the same block attend bidirectionally to one another, enabling parallel context aggregation across the mask span. By explicitly mapping $\mathbf{M}_{\text{diff}}$ to the bottom rows of the attention matrix, this structural isolation ensures that the corrupted context, comprising the anchor token and subsequent `<mask>` tokens processed via the diffusion path (red arrows) can jointly predict the future trajectory without attending to other parallel blocks.

Training objective. During training, the diffusion view utilizes the `<mask>` tokens to predict the subsequent tokens within the block, minimizing the forward KL divergence against the full predictive distribution of the frozen AR model over all masked positions:

$$\mathcal{L}_{\text{Orthrus}} = \mathbb{E}_{\mathbf{x},\{a_b\}} \left[\sum_{b=1}^{B} \sum_{k=1}^{K} D_{\text{KL}}(p_{\text{AR}}(\cdot \mid \mathbf{x}_{\leq a_b+k-1}) \parallel p_{\text{diff}}(\cdot \mid \mathbf{x}_{<a_b}, \tilde{\mathbf{y}}_b)) \right], \tag{7}$$

where $p_{\text{AR}}(\cdot \mid \mathbf{x}_{\leq a_b+k-1})$ is the full token distribution predicted by the frozen AR head at sequence position $a_b + k - 1$, and $p_{\text{diff}}(\cdot \mid \mathbf{x}_{<a_b}, \tilde{\mathbf{y}}_b)$ is the parallel prediction of the diffusion view at the corresponding masked position. This soft distillation objective transfers the full predictive distribution of the AR model into the diffusion view. Gradients flow exclusively through diffusion module and the AR backbone remains strictly frozen throughout.

3.3 Inference: Exact Distribution Matching via Intra-Model Consensus

At inference time, the structural unification of Orthrus enables a continuous, high-throughput generation loop executed entirely over a singular KV cache. Let $\mathbf{x}_{\leq t}$ denote the currently generated sequence prefix, and $(\mathbf{K}_{\text{AR}}^{<t}, \mathbf{V}_{\text{AR}}^{<t})$ its corresponding high-fidelity cache computed natively by the AR backbone. The Orthrus inference loop proceeds through a continuous cycle of projection and structural synchronization:

Parallel Block Projection. To bypass the sequential bottleneck, the diffusion view utilizes the shared KV cache to project a continuous trajectory of future tokens. To initiate parallel generation, we construct a block $\tilde{\mathbf{y}}_t$ of size K by taking the current anchor token x_t and concatenating it with $K-1$ `<mask>` tokens. The diffusion head processes this entire extended block in **a single parallel forward pass**. Unlike other DLMs that rely on multi-step iterative denoising, we empirically find that this single-step projection is substantially more efficient, achieving a strictly higher token-per-forward-pass ratio. By conditioning directly on the high-fidelity KV cache $(\mathbf{K}_{\text{AR}}^{<t}, \mathbf{V}_{\text{AR}}^{<t})$ natively constructed by the AR view, this pass yields a full, simultaneous projection of K candidate tokens $\hat{\mathbf{y}} = (\hat{y}_1, \ldots, \hat{y}_K) \sim p_{\text{diff}}(\cdot \mid \mathbf{x}_{<t}, \tilde{\mathbf{y}}_t)$ (Figure 2(b), Step 1).

Intra-Model Distribution Matching. To guarantee that the parallel projection strictly recovers the target distribution without conditional drift, the trajectory $\hat{\mathbf{y}}$ must be mathematically aligned with the exact causal distribution of the base model. The architecture routes the fully materialized block $\hat{\mathbf{y}}$ through the frozen AR head. Because these K positions are fully populated in the input sequence, the AR head computes the exact target probabilities $p_{\text{AR}}(v \mid \mathbf{x}_{\leq t}, \hat{\mathbf{y}}_{1:k-1})$ for all $k \in \{1, \ldots, K\}$ simultaneously in a **single forward pass**.

Architectural Consensus Mechanism. With both the parallel prior distribution p_{diff} and the exact target distribution p_{AR} computed within the same representational space, the architecture dynamically synchronizes the projected tokens via a strict left-to-right evaluation. The consensus mechanism enforces strict structural identity with the causal AR path. A projected token $\hat{y}_k$ is retained if and only if it matches the greedy AR prediction exactly:

$$\hat{y}_k = \arg\max_{v \in \mathcal{V}} p_{\text{AR}}(v \mid \mathbf{x}_{\leq t}, \hat{\mathbf{y}}_{1:k-1}) \tag{8}$$

For diverse generation (with temperature $T > 0$), the architecture leverages an exact rejection sampling to align the parallel projection with the target distribution, guaranteeing strictly lossless sampling (Leviathan et al., 2022). If structural divergence occurs at index $j \leq K$, verification halts. The architecture commits the synchronized prefix $\hat{\mathbf{y}}_{1:j-1}$ alongside the exact causal correction token y_j drawn directly from p_{AR}, and truncates the shared KV cache to step $t + j$ (Figure 2(b), Step 2). This synchronization preserves the exact predictive distribution of the base model, delivering lossless inference acceleration.

4 Experiments

4.1 Experimental Setup

Baselines and Model Scalability. To demonstrate the scalability and generalizability of our dual-view architecture, we select the state-of-the-art Qwen3 model family (Yang et al., 2025) as our foundation baselines. Specifically, we evaluate the 1.7B, 4B, and 8B parameter variants to observe how Orthrus scales from small to standard large language models. The original AR backbone of each model remains frozen, with only the injected diffusion attention module being optimized.

Evaluation Benchmarks. To rigorously test the capacity of the diffusion head to mirror exact causal distributions without conditional drift, we evaluate Orthrus across a diverse and highly complex suite of zero-shot reasoning and algorithmic tasks. For mathematical reasoning, we benchmark performance on GSM8K (Cobbe et al., 2021), MATH-500 (Hendrycks et al., 2020), and recent AIME challenges (AIME24, AIME25) (Art of Problem Solving, 2026). For structural and programmatic generation, we utilize HumanEval (Chen et al., 2021), MBPP (Austin et al., 2021), Pseudo2code (Ye et al., 2025b), and LiveCodeBench-v5 (Jain et al., 2024). This comprehensive task selection ensures that our empirical claims are validated across long-horizon generative trajectories that strictly penalize distributional divergence.

Implementation Details. During training, we configure the parallel projection block size to $K = 32$ across all model scales. To maximize throughput, we adopt a single-step prediction strategy for the masked block, which we find sufficient to produce high-quality for the diffusion prediction. The models are trained for two epochs on a dataset of 600K examples (detailed in Appendix A). For each training instance, we construct a clean text context with a maximum length of 2048 tokens and generate a corresponding corrupted sequence containing 256 masked blocks placed at random anchor positions. The autoregressive backbone remains strictly frozen, only the newly injected diffusion heads are updated. Training is conducted on a single 8×H200 GPU node, utilizing FlexAttention (Dong et al., 2025) with the FlashAttention-4 backend (Zadouri et al., 2026) to implement the customized training masks.

4.2 Efficiency Benchmarking

Efficiency Metrics. We isolate algorithmic efficiency using Effective Tokens Per Forward Pass:

$$\text{TPF} = \frac{\text{Total Generated Tokens}}{\text{Total Forward Passes}} \tag{9}$$

This hardware-agnostic metric quantifies the average token throughput per inference step. Relative speedups are benchmarked against AR baselines, which are bounded to a maximum TPF of 1. For Orthrus, each continuous generation cycle inherently requires exactly two forward passes. By guaranteeing at least one token per cycle, this establishes a strict theoretical lower bound of 0.5 TPF (1 token per 2 passes). However, by leveraging the parallel diffusion view to project token blocks in a single initial forward pass, Orthrus bypasses the sequential bottleneck of standard AR inference.

Furthermore, our architecture conceptually advances the goals of traditional speculative decoding. Unlike standard speculative paradigms that rely on external draft models, incurring significant memory overhead to maintain isolated KV caches, our intra-model approach achieves parallel acceleration natively over a single shared KV cache, making it highly optimal for high-throughput production. A discussion comparing our architecture against speculative drafting systems is detailed in Section 4.4.

Table 1: **Efficiency Benchmarking.** We report Tokens Per Forward Pass (TPF) and relative speedup against the sequential AR baseline (which operates at **TPF** = 1.0 and 1.0× speedup) for both greedy decoding (temperature $T = 0$) and diverse sampling (temperature $T = 1$).

Task	Qwen3-1.7B				Qwen3-4B				Qwen3-8B			
	$T = 0$		$T = 1$		$T = 0$		$T = 1$		$T = 0$		$T = 1$	
	TPF	**Speedup**	**TPF**	**Speedup**	**TPF**	**Speedup**	**TPF**	**Speedup**	**TPF**	**Speedup**	**TPF**	**Speedup**
GSM8K	4.20	4.37×	3.88	4.04×	4.91	4.48×	4.97	4.53×	5.30	5.22×	4.69	4.62×
MATH-500	4.71	4.74×	4.47	4.50×	6.02	5.76×	5.62	5.38×	6.35	5.95×	5.80	5.43×
AIME24	4.33	5.65×	3.84	5.01×	5.00	6.06×	4.70	5.70×	5.63	6.81×	4.7	5.68x
AIME25	3.89	4.80×	3.55	4.38×	5.17	5.86×	4.37	4.95×	5.25	6.03×	4.40	5.05×
HumanEval	2.75	3.07×	2.60	2.90×	3.57	3.67×	3.37	3.46×	3.94	3.68×	3.37	3.15×
MBPP	2.76	3.07×	2.88	3.20×	3.74	4.43×	3.50	4.15×	3.95	3.90×	3.75	3.70×
Pseudo2code	4.60	4.90×	4.37	4.65×	7.40	7.38×	6.95	6.93×	7.51	7.83×	7.29	7.60×
LiveCodeBench-v5	3.86	5.87×	3.58	5.44×	4.50	6.28×	4.28	5.97×	5.17	6.68×	4.34	5.61×
Average	**3.89**	**4.25×**	**3.65**	**4.27×**	**5.04**	**5.20×**	**4.72**	**5.13×**	**5.39**	**5.36×**	**4.43**	**5.02×**

Table 1 details these efficiency gains across our evaluation suite. Orthrus delivers substantial inference acceleration on all reasoning and algorithmic tasks, achieving an average TPF of 5.39 at the 8B parameter scale. Crucially, unlike existing DLMs that inherently trade generation quality for inference speed, Orthrus *mathematically guarantees exact distributional parity with the AR baseline*, ensuring strictly lossless acceleration. Beyond single-request inference, we further evaluate Orthrus under continuous batching; detailed results are provided in Appendix B.

4.3 Comparison with State-of-the-Art Diffusion Models

While diffusion language models offer a novel path to parallel decoding, they often suffer from significant conditional drift. Achieving baseline coherence in these models demands massive computational resources. For instance, adaptation approaches like SDAR (Cheng et al.) require continuous pre-training on 50B tokens, while models such as Dream (Ye et al., 2025a) are trained on upwards of 580B tokens. Despite these training costs, these models still exhibit performance degradation and struggle to math the high-fidelity reasoning capabilities inherent to autoregressive models.

Table 2 contrasts Orthrus with state-of-the-art diffusion paradigms on complex mathematical and structural reasoning benchmarks. The results demonstrate a clear performance gap: existing diffusion-based models, including Dream (Ye et al., 2025a), Fast-dLLM-v2 (Wu et al., 2025), LLaDA-1.5 (Zhu et al., 2025), SDAR (Cheng et al.), Mercury Coder (Khanna et al., 2025) and Diffusion Gemini (Google DeepMind, 2025), consistently lag behind in accuracy. Crucially, even though SDAR is initialized from the same Qwen3 foundation model as our architecture, suffers from degraded performance. In contrast, Orthrus is computationally efficient. By decoupling the diffusion head from the frozen AR backbone, we avoid the destructive interference caused by full-model fine-tuning. We successfully inject parallel generation capabilities by fine-tuning only 16% of the total model parameters on less than 1B tokens, a lightweight process requiring under 24 hours on a single 8xH200 node.

Table 2: Performance comparison with SOTA Diffusion Models.

Model	Params	GSM8K	MATH-500	AIME-24	AIME-25	HumanEval	MBPP
Dream-7B	7B	79.3	39.6	-	-	57.9	58.8
LLaDA-1.5	7B	82.4	57.4	-	-	52.4	42.8
SDAR-Qwen3-8B	8B	91.7	78.6	10.0	10.0	78.7	72.0
SDAR-Qwen3-30B-A3B	30B	91.4	77.8	16.7	10.8	71.6	87.2
Fast-dLLM-v2	7B	83.7	61.5	-	-	43.3	28.2
Mercury Coder Small	-	-	-	-	-	90.0	76.6
Gemini Diffusion	-	-	-	-	20.0	89.6	76.0
Orthrus-Qwen3-8B	**8.7B**	**96.0**	**86.2**	**28.3**	**23.3**	**95.1**	**93.4**

Most importantly, because Orthrus relies on intra-model consensus rather than altering the base weights, its reasoning performance is directly inherited from, and upper-bounded by the selected frozen AR baseline. In our experiments, Orthrus achieves the exact zero-shot accuracy of the base Qwen3-8B model, establishing a new state-of-the-art for parallel generation fidelity. This structural property makes Orthrus a highly scalable, plug-and-play framework: *it can be seamlessly adapted to any high-quality existing open-source AR model* to unlock parallel throughput without sacrificing elite reasoning capabilities.

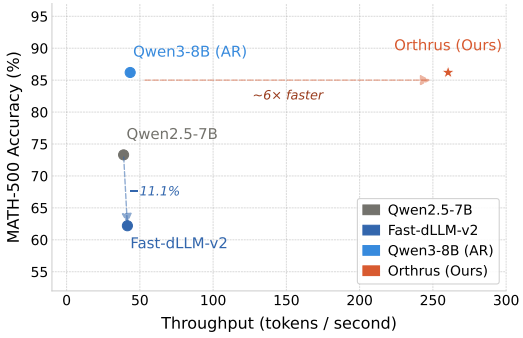

Figure 3: **Throughput vs. Accuracy on MATH-500.** Orthrus delivers a 6× speedup over the Qwen3-8B baseline with strictly lossless performance, whereas Fast-dLLM-v2 suffers severe accuracy degradation.

To further illustrate this structural advantage, Figure 3 visualizes the performance-efficiency trade-off in terms of absolute wall-clock throughput (tokens per second). Adaptation methods like Fast-dLLM-v2 incur a severe 11.1 point degradation on the MATH-500 benchmark relative to their AR baselines. Furthermore, their theoretical acceleration is often negated in practice, yielding negligible speedups due to the multiple iterative refinement steps required to recover output coherence. In contrast, Orthrus bypasses these inefficiencies entirely, delivering up to a 6× speedup with strictly lossless generation.

4.4 Comparison with Speculative Decoding

We contextualize Orthrus against state-of-the-art speculative decoding methods, specifically EAGLE-3 (Li et al., 2025) and DFlash (Chen et al., 2026), evaluated on the Qwen3-8B foundation model. Conventional speculative decoding frameworks typically employ a separate draft model to propose candidate tokens, which are subsequently verified by the target AR model. This approach introduces additional memory overhead, particularly when the drafter and verifier maintain separate KV caches. In contrast, Orthrus adopts a structurally unified architecture in which the diffusion head directly reuses the KV cache of the frozen AR backbone, eliminating redundant historical KV cache storage. A detailed empirical analysis of memory consumption is provided in Appendix B.

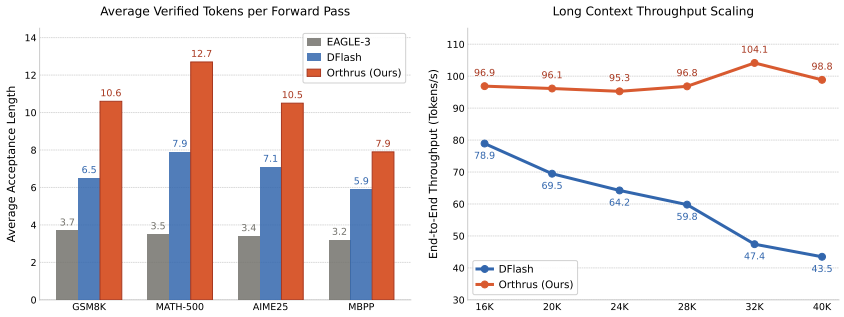

Figure 4: **Performance comparison across tasks and context lengths. Left:** Average acceptance length of Orthrus, EAGLE-3, and DFlash across reasoning and coding benchmarks. **Right:** End-to-end generation throughput across context lengths from 16K to 40K tokens.

We first evaluate generation efficiency across diverse reasoning and coding tasks using average acceptance length, defined as the mean number of tokens accepted per verification step. As shown in Figure 4 (Left), Orthrus consistently outperforms EAGLE-3 and DFlash across all evaluated benchmarks. Notably, on MATH-500, Orthrus achieves an average acceptance length of 11.7, substantially exceeding DFlash (7.9) and EAGLE-3 (3.5). These results demonstrate the effectiveness of our unified architecture in producing high-quality parallel token predictions across different generation tasks. Beyond task-level robustness, Orthrus demonstrates remarkable stability as context length increases. As illustrated in Figure 4 (Right), Orthrus maintains nearly constant end-to-end throughput across context lengths ranging from 16K to 40K tokens. In contrast, DFlash experiences a substantial throughput decline. At 40K tokens, Orthrus achieves a 2.27× higher throughput than DFlash. These findings highlight the robustness of Orthrus across varying context lengths, demonstrating its ability to sustain efficient parallel generation in long-context scenarios.

5 Ablation Study and Analysis

Effect on Parallel Block Size (K). We evaluate throughput and latency sensitivity to the parallel block size (K) on MATH-500 using Orthrus-Qwen3-8B. By processing the extended block simultaneously against a pre-computed KV cache, the diffusion view maintains a constant forward-pass latency across all evaluated sizes (Figure 5). Scaling to $K = 32$ increases the TPF to 6.35, yielding a 3.6× throughput multiplier over $K = 4$ with zero latency penalty. We select $K = 32$ as the optimal configuration to maximize parallel acceleration.

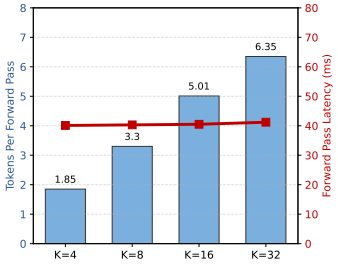

Figure 5: Throughput vs. Latency.

Ablation on Multi-Step Denoising. To validate our single-step projection strategy, we evaluate a multi-step iterative denoising variant adapted from Fast-dLLM-v2 (Wu et al., 2025). During training, rather than masking all future tokens, we randomly mask 50% of the block positions and apply a complementary masking strategy across dual views to ensure comprehensive supervision. During inference, it requires two sequential forward passes to fully materialize the block. As detailed in Table 3, while iterative refinement is standard in diffusion literature, the additional computational pass degrades throughput. The 2-step prediction strategy slashes the TPF by 1.8×, confirming that single-step projection is optimal for our approach.

Table 3: Impact of multi-step denoising.

Strategy	**Passes**	**TPF**
Multi-Step	2	3.53
Orthrus (Single-Step)	**1**	**6.35**

Ablation on Trainable Components. To investigate the effectiveness of selectively training sequence-mixing components, we compare three configurations: (1) updating only attention projections, (2) jointly updating attention projections and feed-forward networks (FFNs), and (3) updating

only FFNs while freezing attention. All variants are evaluated on Qwen3-1.7B under identical training conditions.

Table 4: **Ablation on trainable components.** All configurations use Qwen3-1.7B under identical training conditions.

Trainable Components	Params (%)	Acceptance	TPF
Attention Only	**16**	**6.94**	**3.9**
Attention + FFN	77	6.65	3.8
FFN Only	61	2.53	1.7

As shown in Table 4, training only attention projections achieves the best performance, with an average acceptance length of 6.94 and a TPF of 3.9, while updating only 16% of model parameters. Jointly training attention and FFN layers increases the trainable parameter count to 77% without improving performance. In contrast, training only FFN layers substantially reduces the average acceptance length. These results suggest that the primary distinction between the AR and diffusion views lies in how information is mixed across sequence positions rather than in the capacity of position-wise transformations. Consequently, adapting only the sequence-mixing components provides an effective and parameter-efficient mechanism for parallel generation.

Generalization Across Model Architectures To evaluate the generalizability of Orthrus beyond the Qwen3 model family, we extend our framework to two additional architectures: Llama-3.2-3B-Instruct[3], a standard Transformer with grouped-query attention (GQA), and Qwen3.5-4B[4], a hybrid architecture combining GQA with Gated DeltaNet layers. The underlying principle remains unchanged: the parallel generation view reuses the native historical state of the frozen AR backbone. For standard attention layers, this corresponds to the KV cache, whereas Gated DeltaNet layers additionally maintain recurrent summary and short-convolution states.

Table 5: **Generalization across model architectures.** Orthrus extends to both dense and hybrid architectures.

Backbone	Architecture	Trainable (%)	Speedup
Llama-3.2-3B-Instruct	GQA	13%	4.2×
Qwen3.5-4B	GQA + Gated DeltaNet	20%	5.3×

As shown in Table 5, Orthrus achieves $4.24\times$ and $5.3\times$ wall-clock speedups on Llama-3.2-3B-Instruct and Qwen3.5-4B, respectively, while updating only 13% and 20% of their model parameters. These results demonstrate that the proposed framework generalizes beyond the Qwen3 family and conventional attention-based Transformers, highlighting the broader applicability of shared-state parallel generation.

6 Conclusion

In this work, we introduced **Orthrus**, a novel dual-architecture framework that fundamentally reconciles the trade-off between autoregressive generation fidelity and diffusion-based parallelism. By embedding a lightweight, trainable diffusion module within a frozen, pre-trained AR backbone, we established a unified system capable of parallel token generation that natively utilizes a shared high-fidelity KV cache. Our empirical results demonstrate that Orthrus effectively breaks the sequential generation bottleneck, delivering up to a $7.8\times$ speedup across diverse mathematical and structural benchmarks while incurring zero redundant memory overhead. By leveraging intra-model consensus, our approach guarantees lossless inference parity, offering a highly efficient and scalable solution for high-throughput deployment.

[3] https://huggingface.co/meta-llama/Llama-3.2-3B-Instruct
[4] https://huggingface.co/Qwen/Qwen3.5-4B

7 Acknowledgments

This research was partially supported by NSF Grant #2239570. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the U.S. Government.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Muhammad Adnan, Akhil Arunkumar, Gaurav Jain, Prashant J Nair, Ilya Soloveychik, and Purushotham Kamath. Keyformer: Kv cache reduction through key tokens selection for efficient generative inference. *Proceedings of Machine Learning and Systems*, 6:114–127, 2024.

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.

Art of Problem Solving. Aime problems and solutions. https://artofproblemsolving.com/wiki/index.php/AIME_Problems_and_Solutions, 2026. Accessed: 2026-04-22.

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

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

Jian Chen, Yesheng Liang, and Zhijian Liu. Dflash: Block diffusion for flash speculative decoding. *arXiv preprint arXiv:2602.06036*, 2026.

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.

Zigeng Chen, Gongfan Fang, Xinyin Ma, Ruonan Yu, and Xinchao Wang. dparallel: Learnable parallel decoding for dllms. *arXiv preprint arXiv:2509.26488*, 2025.

Shuang Cheng, Yihan Bian, Dawei Liu, Linfeng Zhang, Qian Yao, Zhongbo Tian, Wenhai Wang, Qipeng Guo, Kai Chen, Biqing Qi, et al. Sdar: A synergistic diffusion-autoregression paradigm for scalable sequence generation, 2025. *URL https://arxiv. org/abs/2510.06303*, 1(3).

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

Juechu Dong, Boyuan Feng, Driss Guessous, Yanbo Liang, and Horace He. Flex attention: A programming model for generating optimized attention kernels. *arXiv preprint arXiv:2412.05496*, 2(3):4, 2024.

Juechu Dong, Boyuan Feng, Driss Guessous, Yanbo Liang, and Horace He. Flexattention: A programming model for generating fused attention variants. *Proceedings of Machine Learning and Systems*, 7, 2025.

Yonggan Fu, Lexington Whalen, Abhinav Garg, Chengyue Wu, Maksim Khadkevich, Nicolai Oswald, Enze Xie, Daniel Egert, Sharath Turuvekere Sreenivas, Shizhe Diao, et al. Nemotron-labs-diffusion: A tri-mode language model unifying autoregressive, diffusion, and self-speculation decoding. In *Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing*, pages 54868–54887, 2026.

Itai Gat, Heli Ben-Hamu, Marton Havasi, Daniel Haziza, Jeremy Reizenstein, Gabriel Synnaeve, David Lopez-Paz, Brian Karrer, and Yaron Lipman. Set block decoding is a language model inference accelerator. *arXiv preprint arXiv:2509.04185*, 2025.

Google DeepMind. Gemini diffusion is our new experimental research model. https://blog.google/innovation-and-ai/models-and-research/google-deepmind/gemini-diffusion/, May 2025.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, 2025.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

Namgyu Ho, Sangmin Bae, Taehyeon Kim, Hyunjik Jo, Yireun Kim, Tal Schuster, Adam Fisch, James Thorne, and Se-Young Yun. Block transformer: Global-to-local language modeling for fast inference. *Advances in Neural Information Processing Systems*, 37:48740–48783, 2024.

Jinyi Hu, Shengding Hu, Yuxuan Song, Yufei Huang, Mingxuan Wang, Hao Zhou, Zhiyuan Liu, Wei-Ying Ma, and Maosong Sun. Acdit: Interpolating autoregressive conditional modeling and diffusion transformer. *arXiv preprint arXiv:2412.07720*, 2024.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*, 2024.

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

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding, 2023. *URL https://arxiv. org/abs/2211.17192*, 1(2), 2022.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle-3: Scaling up inference acceleration of large language models via training-time test. *arXiv preprint arXiv:2503.01840*, 2025.

Jingyu Liu, Xin Dong, Zhifan Ye, Rishabh Mehta, Yonggan Fu, Vartika Singh, Ce Zhang, and Pavlo Molchanov. Tidar: Think in diffusion, talk in autoregression. *Proceedings of Machine Learning and Systems*, 8:748–762, 2026.

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

Dhruv Nathawani, Shuoyang Ding, Vitaly Lavrukhin, Igor Gitman, Somshubra Majumdar, Evelina Bakhturina, Boris Ginsburg, and Jane Polak Scowcroft. Nemotron-Post-Training-Dataset-v2, August 2025. URL https://huggingface.co/datasets/nvidia/Nemotron-Post-Training-Dataset-v2.

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.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

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

Yuchuan Tian, Yuchen Liang, Shuo Zhang, Yingte Shu, Guangwen Yang, Wei He, Sibo Fang, Tianyu Guo, Kai Han, Chao Xu, et al. From next-token to next-block: A principled adaptation path for diffusion llms. *arXiv preprint arXiv:2512.06776*, 2025.

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.

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*, 2025.

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

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*, 2025a.

Xi Ye, Fangcong Yin, Yinghui He, Joie Zhang, Howard Yen, Tianyu Gao, Greg Durrett, and Danqi Chen. Longproc: Benchmarking long-context language models on long procedural generation. *arXiv preprint arXiv:2501.05414*, 2025b.

Ted Zadouri, Markus Hoehnerbach, Jay Shah, Timmy Liu, Vijay Thakkar, and Tri Dao. Flashattention-4: Algorithm and kernel pipelining co-design for asymmetric hardware scaling. *arXiv preprint arXiv:2603.05451*, 2026.

Zhanhui Zhou, Lingjie Chen, Hanghang Tong, and Dawn Song. dllm: Simple diffusion language modeling. *arXiv preprint arXiv:2602.22661*, 2026.

Fengqi Zhu, Rongzhen Wang, Shen Nie, Xiaolu Zhang, Chunwei Wu, Jun Hu, Jun Zhou, Jianfei Chen, Yankai Lin, Ji-Rong Wen, et al. Llada 1.5: Variance-reduced preference optimization for large language diffusion models. *arXiv preprint arXiv:2505.19223*, 2025.

A Training Details

To train the Orthrus dual-view architecture, we employ a optimized distillation pipeline that isolates the diffusion head while keeping the autoregressive (AR) backbone strictly frozen. Below, we detail the dataset composition, hardware configuration, and hyperparameters utilized to train the models.

Table 6: **Training Hyperparameters.**

Hyperparameter	**Value**
Max Sequence Length (L)	2048
Anchor Blocks per Sequence	256
Parallel Block Size (K)	32
Training Epochs	2
Peak Learning Rate	2×10^{-4}
Learning Rate Scheduler	Cosine
Warmup Ratio	0.05
Gradient Clipping	1.0
Micro Batch Size	1
Gradient Accumulation Steps	16
Global Batch Size	128
Precision	`bfloat16`
Data Domain Split	1:1:1 (Chat:Math:Code)

Datasets. To ensure robust performance across diverse domains, the training corpus is constructed by sampling from the open-source Nemotron-Post-Training-Dataset-v2 (Nathawani et al., 2025). To guarantee exact distributional alignment, target outputs are generated directly by the frozen AR head and used as the distillation signal to train the diffusion head. The sampled data is strictly balanced across three domains: **Mathematical Reasoning**, **Code Generation**, and **General Chat & Instruction Tuning**. During data loading, we enforce a uniform sampling strategy across these

three categories. To maximize hardware utilization, we employ sequence packing, concatenating supervised examples up to a strict maximum sequence length, denoted as $L = 2048$ tokens. This packing strategy yields $471,952$ training instances, equivalent to 0.96B total tokens. For each packed sequence, we uniformly sample exactly 256 anchor blocks. The diffusion view utilizes the `<mask>` tokens within these blocks to predict the corresponding future token trajectory by minimizing the forward KL divergence against the frozen AR teacher's exact predictive distribution. Only the parameters of the newly injected diffusion attention heads are updated.

Hyperparameters. Training was executed on a single computational node equipped with 8 GPUs (e.g., 8×H200). We utilize PyTorch FSDP-2 with a micro-batch size of 1 per device and 16 gradient accumulation steps, yielding an effective global batch size of 128 sequences per optimization step. The diffusion parameters are optimized in `bfloat16` precision to reduce memory overhead and accelerate computation. The model is trained for 2 epochs using a cosine learning rate scheduler with a peak learning rate of 2×10^{-4} and a 5% warmup ratio. To ensure stability during training, we apply gradient clipping with a maximum norm of 1.0. A summary of the training hyperparameters is provided in Table 6.

B Further Analysis

Training Objective. In our standard configuration, the diffusion view is trained via forward KL divergence (Equation 7) to distill the full predictive distribution of the AR teacher. To validate this design, we ablate our soft distillation objective by training a variant of Orthrus-Qwen3-8B using standard Cross-Entropy (CE) against the hard ground-truth tokens of the dataset.

Table 7: Impact of the Training Objective.

Objective	Accuracy	TPF
Cross-Entropy (Hard)	86.2%	5.86
KL Divergence (Soft)	**86.2%**	**6.35**

As shown in Table 7, while the exact intra-model consensus mechanism guarantees that both models achieve the identical baseline accuracy on MATH-500, their inference speeds diverge dramatically. Training with hard labels causes the diffusion head to overfit to the dataset's surface syntax rather than internalizing the specific causal trajectory preferred by the AR base model. During inference, this structural misalignment triggers high rejection rates in the consensus validation phase, slashing the Effective Tokens Per Forward Pass (TPF) from 6.35 down to 5.86.

Memory Footprint Scaling. A key advantage of the Orthrus architecture is its extreme memory efficiency during inference, particularly regarding peak GPU memory and dynamic Key-Value (KV) cache. By integrating the diffusion view alongside the standard autoregressive attention mechanism, Orthrus maintains a lean memory profile. Across varying sequence lengths, the peak GPU memory penalty is negligible (~ 100 MiB). Furthermore, the architecture eliminates the redundant KV cache overhead typical of multi-model speculative decoding. Because the diffusion head conditions directly on the AR head's causal cache, the only additional memory required is the transient state for the fixed-size parallel projection block ($K = 32$). Consequently, Orthrus exhibits a strictly constant $O(1)$ KV cache overhead. As demonstrated in Figure 6, this manifests as a fixed $\Delta \approx 4.5$ MiB increase regardless of the total sequence length L, allowing the framework to scale to massive context windows without compounding memory degradation.

Continuous Batching and Serving Efficiency To evaluate the practical serving efficiency of Orthrus, we extend our experiments beyond single-request inference to continuous batching with multiple concurrent requests. We evaluate Orthrus-Qwen3-8B against its autoregressive (AR) baseline on MATH-500 using a single NVIDIA H100 GPU. All configurations use the same backbone and generation settings, and we report aggregate generation throughput in tokens per second.

As shown in Table 8, Orthrus consistently achieves higher aggregate throughput than standard AR decoding across all evaluated concurrency levels. At concurrency 1, Orthrus achieves a 5.11× speedup, while at concurrency 16, it sustains a 2.90× speedup, processing 1001.2 tokens/s compared to 345.4 tokens/s for the AR baseline.

The relative speedup decreases as concurrency increases, since batching improves GPU utilization and arithmetic intensity for the AR baseline, reducing the relative benefits of parallel token generation.

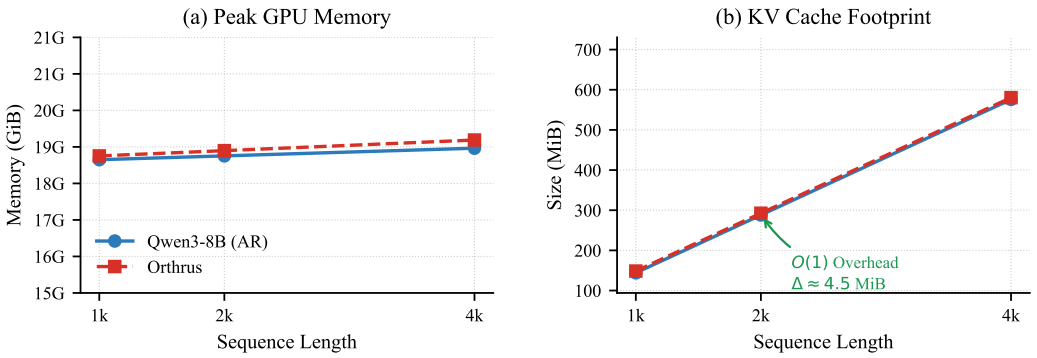

Figure 6: **Memory footprint scaling of Orthrus versus the Qwen3-8B baseline. (a)** The peak GPU memory overhead is practically negligible ($< 1\%$), demonstrating that the dual-view architecture minimizes VRAM penalties. **(b)** The KV cache footprint exhibits a strictly constant $O(1)$ overhead (≈ 4.5 MiB) across all sequence lengths. By completely sharing the historical AR cache, Orthrus natively bypasses the linear cache redundancy typical of standard speculative decoding.

Table 8: **Continuous batching efficiency.** Aggregate generation throughput under varying numbers of concurrent requests. Orthrus consistently outperforms the AR baseline, maintaining a 2.90× speedup at concurrency 16.

Concurrency	Orthrus (tok/s)	AR (tok/s)	Speedup
1	210.6	41.2	5.11×
4	674.3	137.6	4.90×
8	763.1	206.6	3.69×
16	1001.2	345.4	2.90×

Nevertheless, Orthrus maintains substantial acceleration under higher concurrency, demonstrating its effectiveness beyond single-request inference and its potential for practical LLM serving workloads.

C Limitations

Because the Orthrus architecture strictly freezes the autoregressive backbone to guarantee exact inference parity, its generative capabilities are strictly upper-bounded by the foundation model. The diffusion head is distilled exclusively to mirror the AR teacher's exact predictive distribution. Consequently, the framework acts solely as an inference accelerator and inherently inherits any biases, knowledge gaps, or hallucination tendencies present in the underlying base model.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims regarding exact distributional parity, speedups, and memory overhead are directly supported by the experimental results in Section 4 and the scaling analysis in the Appendix.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We explicitly discuss the limitations of our approach in Appendix C.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: The paper proposes an empirical architectural framework and does not rely on formal theoretical proofs.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Full details regarding hyperparameters, dataset construction, training settings, and inference evaluation metrics are provided in Section 4.1 and Appendix A.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We open-source all code, model weights, and scripts.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: We detail the training data composition, configurations, hyperparameters in Section 4.1 and Appendix A.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: As detailed in Section 4.1, all reported generative metrics rely strictly on greedy decoding ($T = 0$) for deterministic evaluation to verify exact distributional alignment, making variance or error bars inapplicable.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Hardware specifications (a single 8xH200 GPU node) and expected execution times (under 24 hours) are explicitly disclosed in Sections 1, 4.1, and Appendix A.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: The research conforms entirely to the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: This paper introduces foundational architectural improvements for inference efficiency and does not present direct, foreseeable negative societal impacts beyond those already associated with standard LLMs.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.

- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper proposes an architectural framework and evaluates on established benchmarks, it does not release new high-risk datasets or standalone foundational models requiring explicit safeguards.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We properly cite and credit the foundation models (Qwen3 family) and open-source datasets (Nemotron) utilized in our experiments.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The code and model weights will be fully documented and released alongside the paper upon acceptance.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The research does not involve human subjects or crowdsourcing.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The research does not involve human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: While LLMs are the subject of the optimization research, they are not utilized as a non-standard component or core method to conduct the research itself.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.