UNIFYING AUTOREGRESSIVE AND DISCRETE DIFFUSION LANGUAGE MODELING VIA CROSS-REGRESSIVE DECODING

Dmitry Abulkhanov[1,*] **Daniil Strizhakov**[2,3,*] **Maxim Panov**[1]

ABSTRACT

Inference acceleration can unintentionally change model behavior, complicating alignment-sensitive deployments where post-training (like RLHF) should be preserved. We introduce **Cross-Regression**, a decoding-time method that accelerates generation while providing an explicit mechanism to preserve or relax distributional fidelity. Cross-Regression augments a pretrained autoregressive transformer with a dual-stream design: a frozen control stream computes exact next-token probabilities, and a predictive stream proposes multi-token drafts in parallel. An energy-based acceptance test, derived from the per-token log probability ratio between control and predictive streams, determines how many proposed tokens can be safely committed. The method provides an explicit control between *lossless sampling* and a faster *lossy regime* with controllable deviation. Across models from 1.5B to 70B parameters, we observe strong scaling of acceptance length and realize 3–6$\times$ speedups with near-complete quality retention across reasoning, code, and dialogue benchmarks, and we demonstrate modality transfer by accelerating Whisper decoding.

1 INTRODUCTION

The sequential, autoregressive nature of Large Language Models (LLMs) inherently limits their inference speed posing a significant challenge for real-time applications. To address this, techniques such as speculative decoding Chen et al. (2023); Leviathan et al. (2023a); Xia et al. (2024) and diffusion-based parallel generation Ho et al. (2020); Austin et al. (2021); Li et al. (2023); Christopher et al. (2025) have emerged, offering substantial speedups by leveraging parallelism.

Discrete diffusion models, particularly Masked Diffusion Models (MDMs), offer a promises to break this sequential bottleneck through parallel token generation (Austin et al., 2021; Hoogeboom et al., 2021). Rather than generating left-to-right, MDMs frame generation as iterative denoising: starting from a fully masked sequence, the model progressively unmasks multiple tokens per step until a complete sequence emerges. Recent large-scale MDMs including LLaDA (Nie et al., 2025b), Dream (Wu et al., 2025), and Mercury (Labs et al., 2025) have demonstrated competitive performance when trained at sufficient scale. However, MDMs rely on a mean-field approximation that factorizes the denoising distribution across positions: $p_{\hat{\theta}}(\mathbf{x}^0 \mid \mathbf{x}^t) \approx \prod_{i=1}^{L} p_{\hat{\theta}}(x_i^0 \mid \mathbf{x}^t)$. This assumes tokens can be predicted independently within each step, ignoring the sequential dependencies essential for coherent text. When configured for speedups comparable to speculative decoding (3–5$\times$), diffusion approaches retain only 37–62% of baseline task performance—a catastrophic collapse that renders the speedup practically useless.

Hybrid architectures that combine autoregressive fidelity with parallel generation have shown conceptual promise (Arriola et al., 2025; Huang & Tang, 2025), but require training dedicated models from scratch or extensive fine-tuning. As shown in Table 17, training competitive diffusion language models requires $\approx 35,000 - 150,000$ GPU hours, while even fine-tuning approaches like SBD (Arriola et al., 2025) require thousands of hours. This prohibitive cost limits accessibility and prevents

[*]Equal contribution. [1]Mohamed Bin Zayed University of AI, Abu Dhabi, UAE. [2]Moscow State Institute of Steel and Alloy, Moscow, Russia. [3]T-Tech. Correspondence: dmitry.abulkhanov@gmail.com

Table 1: Training cost comparison for 7B–8B parameter LM acceleration methods (H100 GPU hours). Cross-Regression requires three orders of magnitude less compute than the nearest competitor.

Method	**#Tokens**	**TFLOPs**	**GPU Hours**
LLaDA (Nie et al., 2025b)	2.3T	1.1×10^{11}	151,000
Dream (Wu et al., 2025)	600B	2.5×10^{10}	34,500
TiDAR (Liu et al., 2025)	150B	7.2×10^{9}	9,900
SBD (Gat et al., 2025)	70B	3.4×10^{9}	4,600
Cross-Regression *(Ours)*	0.3B	4.1×10^{6}	**20**

practitioners from applying acceleration to their specific fine-tuned models without massive retraining investment.

Yet, these existing hybrid approaches typically encounter a difficult trade-off. They either compromise output fidelity by forgoing rigorous, token-by-token verification against the target model, risking uncontrolled quality degradation–or they reintroduce computational overhead by requiring an additional, separate forward pass of the target model for such verification Leviathan et al. (2023a); Cai et al. (2024); Xia et al. (2024).

These paradigms exhibit complementary strengths and limitations. Autoregressive models provide exact sequential probabilities and efficient KV caching but generate only one token per forward pass. Diffusion models enable parallel generation and iterative refinement but suffer from mean-field approximation errors that degrade output quality. Hybrid architectures attempt to combine both paradigms but require prohibitive training costs. Speculative decoding (Stern et al., 2018; Leviathan et al., 2023b; Chen et al., 2023; Xia et al., 2024; Christopher et al., 2025) sidesteps training entirely by using a small draft model whose candidates a large target model verifies in parallel, preserving the exact target distribution through rejection sampling—yet it wastes computation when verification fails, discarding all subsequent draft tokens regardless of their quality with no mechanism for refinement or recovery.

This paper introduces *Cross-Regression* algorithm, which achieves multi-token acceptance through a dual-stream architecture where a single transformer forward pass produces both exact autoregressive probabilities (via control stream P) and parallel future-token proposals (via predictive stream Q). Unlike existing parallel decoding frameworks (see Table 2 for a comparison) which decouple proposal from verification or rely on non-causal approximations, Cross-Regression computes proposal and verification scores jointly, directly targeting the memory-bandwidth bottleneck that dominates large-model inference.

Cross-Regressive decoding as framework offer the following advantages:

Flexibility: the lightweight training procedure allows to apply this method to arbitrary fine-tuned models, with considerable compute bugdet without the prohibitive costs of $\sim 10,000 - 100,000$ GPU-hours required by discrete diffusion or hybrid approaches, while combines their non-autoregressive traits. We emperically show scalling of this method to the model sizes up to 70B parameters.

Exactness/Speedup Knob: It offers two regimes with an explicit quality-speed tradeoff: a lossless mode ($\beta = 0$) preserving the exact distribution via residual correction (Theorem 4.2), and a lossy mode ($\beta > 0$) using EMA-smoothed energy to offset transient errors for guaranteed acceptance gains (Theorem 4.4).

Efficiency: Cross-Regression achieves 3–6$\times$ speedups while maintaining over 99% quality retention in lossy mode, compared to 37–62% for diffusion methods at comparable speeds.

2 Related work

Approaches such as multi-token prediction (Gloeckle et al., 2024), Medusa (Cai et al., 2024), and Eagle (Li et al., 2024), add output heads to an existing language model to predict consecutive future tokens. This multi-head prediction enables self-speculative decoding without a separate draft. Unlike our approach, these methods require adding architectural constructions which introduce a tradeoff challenge and a large to explore hyperparameter space. Any-order autoregressive models (Kong et al.,

2023) discard the left-to-right order but maintain causal attention and exact key-value caches, and have similar draft variants (Pannatier et al., 2024; Guo & Ermon, 2025). Concurrently, Fathi et al. (2025) and Esoteric Language Models (Sahoo et al., 2025) extend Block Diffusion (Arriola et al., 2025) using hybrid noising paths and specialized attention for KV-caching.

	Cross-Regression	**Speculative Decoding**	**Diffusion**
Core mechanism	Adaptive iterative denoising	Rejection sampling	Rigid multi-step denoising schedule
Acceptance latency	✓ Early acceptance	✓ Early acceptance	× Locked by schedule
Causality	Hybrid	Fully causal	Fully non-causal
KV caching	✓ Yes	✓ Yes	× No
Information persistence	✓ Retains accepted; corrects from mismatch	× Discards suffix after mismatch	× Recomputes entire block each step
Distribution guarantee	✓ Exact (lossless mode)	✓ Exact	× No guarantee
Architecture	Base model + LoRA	Two separate models	Dedicated retrained model

Table 2: Essential differences of non-autoregressive / parallel decoding frameworks.

3 BACKGROUND

Discrete Diffusion and the Forward Masking Process Masked Diffusion Models model sequences via a forward corruption process that progressively masks tokens. The forward transition kernel $q_{t|0}(\mathbf{x}^t \mid \mathbf{x}^0)$ operates independently per position according to

$$q_{t|0}(x_i^t \mid x_i^0) = (1-\alpha_t) \cdot \mathbf{1}_{x_i^t = 0} + \alpha_t \cdot \mathbf{1}_{x_i^t = x_i^0},$$

where $\alpha_t : [0,1] \to [0,1]$ is a monotonically decreasing schedule with boundary conditions $\alpha_0 = 1$ (no masking at $t = 0$) and $\alpha_1 = 0$ (fully masked at $t = 1$). Common schedules include the linear schedule $\alpha_t = 1 - t$ providing uniform masking rate, the cosine schedule $\alpha_t = \cos^2(\pi t/2)$ which concentrates masking transitions near $t \approx 0.5$ for smoother learning, and the log-linear schedule $\alpha_t = \exp(-\lambda t)$ for decay rate $\lambda > 0$ common in continuous-time formulations.

The joint forward distribution for the entire sequence factorizes as $q_{t|0}(\mathbf{x}^t \mid \mathbf{x}^0) = \prod_{i=1}^{L} q_{t|0}(x_i^t \mid x_i^0)$, which follows from the independence assumption in the masking noise. Denoting by $M_t = \{i : x_i^t = 0\}$ the set of masked positions at time t, the expected number of masked tokens is $\mathbb{E}[|M_t|] = L(1-\alpha_t)$.

Masked Diffusion Models are trained to minimize the variational Evidence Lower Bound, which takes a specific form for discrete diffusion with absorbing states. The continuous-time ELBO is

$$\mathcal{L}(\hat{\theta}) = \int_0^1 \frac{\alpha_t'}{1-\alpha_t} \, \mathbb{E}_{q(\mathbf{x}^0, \mathbf{x}^t)} \left[\sum_{i : x_i^t = 0} -\log p_{\hat{\theta}}(x_i^0 \mid \mathbf{x}^t) \right] dt.$$

The time weighting $\alpha_t'/(1-\alpha_t)$ is positive since $\alpha_t' < 0$ for a decreasing schedule, and it emphasizes times where masking is actively occurring. The expectation averages over clean data $\mathbf{x}^0 \sim p_{\text{data}}$ and noisy states $\mathbf{x}^t \sim q_{t|0}(\cdot \mid \mathbf{x}^0)$. Only positions currently masked contribute to the loss, and the cross-entropy loss is standard classification loss for predicting the original token.

Mean-Field Approximation The exact reverse posterior required for denoising can be derived via Bayes' rule. For a single token, this takes the form $q_{s|t}(x_i^s \mid x_i^t, x_i^0) = \frac{q_{t|s}(x_i^t | x_i^s) \cdot q_{s|0}(x_i^s | x_i^0)}{q_{t|0}(x_i^t | x_i^0)}$. Due to the coordinate-wise independence of the forward process, the joint reverse posterior factorizes exactly as $q_{s|t}(\mathbf{x}^s \mid \mathbf{x}^t, \mathbf{x}^0) = \prod_{i=1}^{L} q_{s|t}(x_i^s \mid x_i^t, x_i^0)$.

However, during inference $\mathbf{x}^0$ is unknown—it is precisely what we wish to sample. The model must therefore learn an approximation $p_{\hat{\theta}}(\mathbf{x}^0 \mid \mathbf{x}^t)$. Standard masked diffusion architectures using

bidirectional Transformers parameterize this as

$$p_{\hat{\theta}}(\mathbf{x}^0 \mid \mathbf{x}^t) \approx \prod_{i=1}^{L} p_{\hat{\theta}}(x_i^0 \mid \mathbf{x}^t),$$

where each $p_{\hat{\theta}}(x_i^0 \mid \mathbf{x}^t)$ is a softmax over the vocabulary at position i, computed via a single forward pass through the bidirectional encoder. This constitutes the mean-field approximation: it assumes tokens can be predicted independently given the noisy observation $\mathbf{x}^t$.

3.1 Energy-Based Models for Sequence Generation

Energy-Based Models provide an alternative formulation for sequence distributions that avoids explicit factorization assumptions. An EBM defines a distribution over sequences through an energy function $E : \mathcal{V}^L \to \mathbb{R}$ via the Boltzmann distribution $p_{\text{EBM}}(\mathbf{x}) = \frac{1}{\mathcal{Z}} \exp\big(-E(\mathbf{x})\big)$, where $\mathcal{Z} = \sum_{\mathbf{x} \in \mathcal{V}^L} \exp(-E(\mathbf{x}))$ is the partition function ensuring normalization. The energy function assigns lower values to more probable sequences, with the exponential transformation converting energies to (unnormalized) probabilities.

The fundamental challenge with EBMs is the intractability of the partition function $\mathcal{Z}$. Despite this computational challenge, EBMs offer expressive advantages over factorized models. The energy function $E(\mathbf{x})$ can capture arbitrary dependencies between positions without imposing conditional independence assumptions.

4 Method

Energy Decomposition The key insight motivating Cross-Regression emerges from examining how different model classes relate through energy decomposition. Consider the true data distribution $p_{\text{data}}(\mathbf{x})$ with corresponding energy $E_{\text{data}}(\mathbf{x}) = -\log p_{\text{data}}(\mathbf{x}) + \text{const}$. We can decompose this energy into components that different model classes capture with varying fidelity.

The mean-field approximation $p_{\text{MF}}(\mathbf{x}) = \prod_i p_{\text{MF}}(x_i)$ captures only the marginal energies through $E_{\text{MF}}(\mathbf{x}) = \sum_{i=1}^{L} E_i^{\text{marginal}}(x_i)$, where $E_i^{\text{marginal}}(x_i) = -\log p_{\text{MF}}(x_i)$. The residual energy $\Delta E_{\text{MF}}(\mathbf{x}) = E_{\text{data}}(\mathbf{x}) - E_{\text{MF}}(\mathbf{x})$ contains all inter-token dependencies that the mean-field model fails to capture. The autoregressive model $p_{\text{AR}}(\mathbf{x}) = \prod_i p(x_i \mid \mathbf{x}_{<i})$ captures causal dependencies through $E_{\text{AR}}(\mathbf{x}) = \sum_{i=1}^{L} E_i^{\text{causal}}(x_i \mid \mathbf{x}_{<i})$, where $E_i^{\text{causal}}(x_i \mid \mathbf{x}_{<i}) = -\log p(x_i \mid \mathbf{x}_{<i})$. Under the assumption that the autoregressive model is perfectly trained on the data distribution, we have $p_{\text{AR}} = p_{\text{data}}$ and thus $E_{\text{AR}} = E_{\text{data}}$ up to an additive constant.

Theorem 4.1 (Acceptance Length Gain of Lossy Regime)**.** *Under assumption 4.3, let L_0 denote the expected acceptance length with $\beta = 0$, and L_β with EMA parameter $\beta \in (0, 1)$. Then*

$$\mathbb{E}[L_\beta] \geq \mathbb{E}[L_0] \cdot \left(1 + c \cdot \frac{\sigma}{|\mu|} \cdot \beta + O(\beta^2)\right) \tag{1}$$

for a universal constant $c > 0$. In the high-variance regime $\sigma/|\mu| \gg 1$, EMA smoothing yields substantial gains in expected acceptance length.

This observation suggests a natural factorization strategy. Given a mean-field proposal distribution $Q(\mathbf{x}) = \prod_i Q_i(x_i)$ and the target autoregressive distribution $P(\mathbf{x}) = p_{\text{AR}}(\mathbf{x})$, the importance weight relating them is

$$w(\mathbf{x}) = \frac{P(\mathbf{x})}{Q(\mathbf{x})} = \frac{\prod_i P(x_i \mid \mathbf{x}_{<i})}{\prod_i Q_i(x_i)} = \prod_{i=1}^{L} \frac{P(x_i \mid \mathbf{x}_{<i})}{Q_i(x_i)}.$$

Taking logarithms yields the cumulative energy correction

$$\log w(\mathbf{x}) = \sum_{i=1}^{L} \left[\log P(x_i \mid \mathbf{x}_{<i}) - \log Q_i(x_i)\right] = \sum_{i=1}^{L} \Delta E_i,$$

where $\Delta E_i = \log P_i - \log Q_i$ is the per-token energy residual measuring the discrepancy between the causal target and the mean-field proposal at position i.

Algorithm 1 Cross-Regression Decoding

Notation: $P_k \equiv P(\cdot \mid \mathbf{x}, \acute{\mathbf{x}}_{1:k-1})$ *and* $Q_k \equiv Q(\cdot \mid \mathbf{x}, \acute{\mathbf{x}}_{1:k-1})$
Require: Dual-stream model $\mathcal{M}$ (both P and Q), prompt $\mathbf{x}_0$, horizon γ, EMA β, threshold τ
Ensure: Verified–draft pair $(\mathbf{x}, \acute{\mathbf{x}})$
$(\mathbf{x}, \acute{\mathbf{x}}, m) \leftarrow \big(\mathbf{x}_0,\ \mathrm{RandomTokens}(\gamma),\ 0\big)$
while not terminated **do**
▷ *Parallel dual-stream forward pass*
$\{(P_k, Q_k)\}_{k=1}^{\gamma} \leftarrow \mathcal{M}(\mathbf{x}, \acute{\mathbf{x}})$ ▷ Shared forward
▷ *Fill beyond valid prefix and evaluate*
Sample $\acute{x}_k \sim Q_k$ for $k = m+1, \ldots, \gamma$
▷ *Cache* $\{(p_k, q_k)\}_{k=1}^{\gamma}$
$(p_k, q_k) \leftarrow \big(P_k(\acute{x}_k), Q_k(\acute{x}_k)\big)$
$n \leftarrow$ EnergyAccept$\big(\{(p_k, q_k)\}, \beta, \tau\big)$ ▷ Alg. 2
$\mathbf{x} \leftarrow \mathbf{x} \,\|\, \acute{\mathbf{x}}_{1:n}$ ▷ Accept verified tokens
if $n = \gamma$ **then**
$(\acute{\mathbf{x}}, m) \leftarrow \big(\mathrm{RandomTokens}(\gamma),\ 0\big)$
else
$(P, Q) \leftarrow \mathcal{M}(\mathbf{x}, 1)$ ▷ Correction sampling
Sample $x_{\text{corr}} \sim p_{\text{res}}$ where $p_{\text{res}}(v) \propto \max\big(0, P(v) - Q(v)\big)$
$\mathbf{x} \leftarrow \mathbf{x} \,\|\, x_{\text{corr}}$
▷ *Shift draft buffer*
▷ *TODO:* $\acute{x}$ *resampling*
$(\acute{\mathbf{x}}_{1:\gamma-n-1}, m) \leftarrow \big(\acute{\mathbf{x}}_{n+2:\gamma},\ \gamma - n - 1\big)$
end if
end while
return $(\mathbf{x}, \acute{\mathbf{x}})$

Algorithm 2 Energy-Based Acceptance

Require: Cached probabilities $\{(p_k, q_k)\}_{k=1}^{\gamma}$, EMA coefficient β, threshold $\tau \le 0$, horizon γ
Ensure: Number of accepted tokens n
$(\rho, n) \leftarrow (0, 0)$
for $k = 1$ to γ **do**
$E_k \leftarrow \log p_k - \log q_k$ ▷ Energy at position k
$\rho \leftarrow \beta\rho + (1-\beta)E_k$
$\hat{\rho}_k \leftarrow \rho \,/\, (1 - \beta^k)$ ▷ Bias-corrected EMA
Sample $u_k \sim \mathrm{LogUniform}(\tau, 0)$
$\eta_k \leftarrow \max(\tau \cdot \sqrt{k/\gamma},\ u_k)$ ▷ Acceptance bound
if $\hat{\rho}_k < \eta_k$ **then return** n
end if
$n \leftarrow k$
end for
return n ▷ Full acceptance

4.1 Dual-Stream Architecture

Cross-Regression operates on an extended state space that includes both committed and provisional tokens at various stages of verification. The accepted sequence $\mathbf{x} = (x_1, \ldots, x_n) \in \mathcal{V}^n$ represents the currently verified suffix—these are tokens that have passed the Metropolis-Hastings acceptance criterion and been permanently committed to the output. Each scalar token $x_i \in \mathcal{V}$ within this sequence satisfies the property that the cumulative energy remained below threshold at the moment of acceptance. Once a token enters $\mathbf{x}$, it remains fixed; the accepted sequence grows monotonically as generation proceeds.

The predictive stream candidates $\acute{\mathbf{x}} = (\acute{x}_1, \ldots, \acute{x}_\gamma) \in \mathcal{V}^\gamma$ contain γ candidate tokens generated in parallel by the predictive stream. The acute notation distinguishes these unverified proposals from accepted tokens: each $\acute{x}_j \sim Q(\cdot \mid \mathbf{x}, \tilde{\mathbf{x}})$ awaits sequential verification against the target distribution p_{AR}, the predictive stream state $\tilde{\mathbf{x}} = (\chi_1, \ldots, \chi_m)$ stores previously proposed but unaccepted tokens after current round of acceptance. Some prefix of $\acute{\mathbf{x}}$ will be accepted and appended to $\mathbf{x}$, while the remaining tokens—those that were proposed but not accepted—persist in the predictive stream state for subsequent iterations.

4.2 Cross-Regressive Decoding

Cross-Regression samples from a joint Energy-Based Model structured as

$$\pi(\mathbf{x}, \acute{\mathbf{x}} \mid \mathbf{x}_0, \tilde{\mathbf{x}}) \propto p_{\text{AR}}(\mathbf{x} \mid \mathbf{x}_0) \cdot Q(\acute{\mathbf{x}} \mid \mathbf{x}, \tilde{\mathbf{x}}) \cdot \exp\left(-E_{\text{couple}}(\mathbf{x}, \acute{\mathbf{x}})\right), \tag{2}$$

where the coupling energy enforces consistency between accepted and proposed tokens.

Energy-Based Soft Coupling Rather than enforcing hard constraints between streams, we employ a soft energy-based coupling that measures and accumulates the discrepancy between control and predictive distributions. The per-position coupling energy is defined as the log probability ratio:

$$E_k = \log P(\acute{x}_k \mid \mathbf{x}, \acute{\mathbf{x}}_{<k}) - \log Q(\acute{x}_k \mid \mathbf{x}, \tilde{\mathbf{x}}),$$

where positive values indicate alignment (control stream assigns higher probability) and negative values indicate divergence (predictive stream overestimates).

EMA Smoothing for Variance Reduction This energy is accumulated through exponential moving average smoothing: $\rho_k = \beta\rho_{k-1} + (1-\beta)E_k, \quad \hat{\rho}_k = \frac{\rho_k}{1-\beta^k}$, where $\hat{\rho}_k$ is the bias-corrected estimate. The coupling mechanism accepts position k if and only if the smoothed energy exceeds an adaptive threshold: $\hat{\rho}_k \geq \eta_k, \quad$ where $\eta_k = \max\left(\tau\sqrt{k/\gamma},\, u_k\right)$ for base threshold $\tau \leq 0$, horizon γ, and stochastic perturbation $u_k \sim \text{LogUniform}(\tau, 0)$.

The joint distribution over accepted sequences and predictive stream proposals thus takes the form:

$$\pi(\mathbf{x}, \acute{\mathbf{x}} \mid \mathbf{x}_0, \tilde{\mathbf{x}}) \propto p_{\text{AR}}(\mathbf{x} \mid \mathbf{x}_0) \cdot Q(\acute{\mathbf{x}} \mid \mathbf{x}, \tilde{\mathbf{x}}) \cdot \prod_{k=1}^{n} \mathbf{1}\{\hat{\rho}_k \geq \eta_k\},$$

where $n = \max\{k : \hat{\rho}_j \geq \eta_j \text{ for all } j \leq k\}$ is the acceptance length

4.3 Lossless and Lossy Regimes

Cross-Regression provides explicit control over the speed-quality trade-off through two distinct operating regimes, governed by the EMA coefficient β and the base threshold τ. The interplay between β and τ defines a continuous family of operating points along the Pareto frontier of speed versus quality. At one extreme ($\beta = 0$, $\tau = 0$), the method is provably lossless with acceptance behavior similar to speculative decoding with reiteration of not accepted tokens. At the other extreme ($\beta \to 1$, $\tau \to -\infty$), the method would accept all proposals regardless of quality, maximizing speed but abandoning distributional guarantees.

Lossless Regime ($\beta = 0$) When the EMA coefficient is set to $\beta = 0$, Cross-Regression *exactly preserves* the target autoregressive distribution p_{AR}. In this configuration, the smoothed energy reduces to the instantaneous energy at each position:

$$\hat{\rho}_k = E_k = \log P_k - \log Q_k,$$

eliminating all temporal averaging. The acceptance criterion becomes a position-wise rejection sampling test: token $\acute{x}_k$ is accepted if and only if $\log P_k - \log Q_k \geq u_k$, where $u_k \sim \text{LogUniform}(\tau, 0)$.

When $\tau = 0$, this further simplifies to $P_k \geq Q_k \cdot e^{u_k}$ with $u_k \leq 0$, which similar to the standard speculative decoding acceptance rule (Leviathan et al., 2023b). The key property of the lossless regime is that rejected tokens trigger resampling from the residual distribution $p_{\text{res}}(v) \propto \max(0, P(v) - Q(v))$, ensuring that the marginal distribution of accepted tokens matches p_{AR} exactly. Theorem 4.2 formalizes this guarantee: tokens sampled via Cross-Regression with $\beta = 0$ are distributed identically to tokens sampled directly from the target model.

Theorem 4.2 (Distributional Equivalence of Lossless Regime)**.** *Tokens sampled via cross-regressive decoding from $p(x)$ and $q(x)$ with $\beta = 0$ are distributed identically to tokens sampled directly from $p(x)$.*

The proof appears in Section D.1.

Lossy Regime ($\beta > 0$ or $\tau < 0$) Relaxing either parameter introduces controlled deviation from the exact distribution in exchange for increased acceptance lengths and reduced variance. Two complementary mechanisms enable this trade-off:

EMA Smoothing ($\beta > 0$): When $\beta > 0$, the exponential moving average accumulates energy across positions, allowing transient negative energies (where the predictive stream Q locally overestimates

the target P) to be offset by subsequent positive energies (where Q underestimates). Concretely, a token $\acute{x}_k$ with $E_k < 0$ may still be accepted if the smoothed estimate $\hat{\rho}_k$ remains above threshold due to favorable energies at earlier positions.

Threshold Relaxation ($\tau < 0$): Setting the base threshold $\tau < 0$ introduces explicit tolerance for cumulative negative energy. The adaptive threshold $\eta_k = \max(\tau\sqrt{k/\gamma}, u_k)$ allows sequences where the predictive stream systematically overestimates certain tokens to still be accepted, provided the discrepancy remains bounded.

Assumption 4.3 (Stationary Energy)**.** The energy sequence $\{E_k\}$ satisfies $\mathbb{E}[E_k] = \mu < 0$ and $\mathrm{Var}(E_k) = \sigma^2 < \infty$ with uncorrelated increments.

Theorem 4.4 (Acceptance Length Gain of Lossy Regime)**.** *Under assumption 4.3, let L_0 denote the expected acceptance length with $\beta = 0$, and L_β with EMA parameter $\beta \in (0, 1)$. Then*

$$\mathbb{E}[L_\beta] \geq \mathbb{E}[L_0] \cdot \left(1 + c \cdot \frac{\sigma}{|\mu|} \cdot \beta + O(\beta^2)\right) \tag{3}$$

for a universal constant $c > 0$. In the high-variance regime $\sigma/|\mu| \gg 1$, EMA smoothing yields substantial gains in expected acceptance length.

The proof appears in Section D.2.

Residual Energy Guidance The per-position energy E_k serves not merely as a passive discrepancy measure but as an active *guidance signal* that steers the decoding process. We term this mechanism *residual energy guidance* to emphasize its dual role in both acceptance control and error correction.

For correction, the residual energy directly defines the correction distribution. Upon rejection at position $n + 1$, the *residual distribution*

$$p_{\text{res}}(v) \propto \max\big(0, P(v) - Q(v)\big)$$

captures precisely the probability mass that the predictive stream failed to account for. Sampling $x_{\text{corr}} \sim p_{\text{res}}$ (Line 12 of Algorithm 1) thus corrects the specific deficiency rather than discarding the predictive stream's contribution entirely. This targeted correction, guided by the residual energy, ensures distributional exactness in lossless mode while enabling the predictive stream cache mechanism to recycle subsequent tokens $\acute{\mathbf{x}}_{n+2:\gamma}$ for the next iteration.

5 Experiments and Analysis

We fine-tune a lightweight, parameter-efficient LoRA adapter that equips the *predictive stream* $\mathcal{M}_q$ with the ability to forecast *multi-token* futures, while the *control stream* $\mathcal{M}_p$ remains frozen. The adapter increases the total parameter count by well under one percent, so memory footprint at inference time is effectively unchanged. The complete training procedure is detailed in Algorithm 3 in Appendix B.

Whisper Experiments To demonstrate the architectural universality of Cross-Regression, we extend our evaluation beyond decoder-only language models to the Encoder-Decoder architecture of Whisper (Radford et al., 2023) for speech recognition. We fine-tuned a predictive stream LoRA adapter on the Whisper decoder layers using the Librispeech (Panayotov et al., 2015) `train-clean` split for a single epoch. This minimal training regime tests the method's sample efficiency in learning to draft future tokens conditioned on audio encoder features.

Results in Table 4 reveal a consistent scaling trend: speedups increase significantly with model size, rising from $1.5\times$ for Whisper Tiny to $2.7\times$ for Whisper Medium. This aligns with our findings in Section A.1, suggesting that larger models possess richer latent representations that the predictive stream can leverage for more accurate parallel drafting. Furthermore, the method maintains robust acceleration even under high-entropy sampling ($T = 1$), confirming its efficacy for diverse generation settings. These results establish Cross-Regression as a viable acceleration strategy for conditional sequence generation tasks beyond standard text completion.

Table 3 demonstrate that Cross-Regression establishes the speed-quality Pareto frontier.

[1]LLaDA 8B and LLaMA3 8B share identical architectures. Diffusion comparisons are contextual as they benchmarked against LLaMA3 in original works (Nie et al., 2025b; Wu et al., 2025).

Base	Framework	Method	GSM8K			HumanEval			Mean	
			Acc@1	*Rel. Acc*	*Speedup*	*Pass@1*	*Rel. Pass*	*Speedup*	*Quality*	*Speedup*
LLaMA3 8B	Diffusion	LLaDA (T=1024)	0.83	100.0%	0.33	0.19	100.0%	0.52	100.0%	0.43
		LLaDA (T=256)	0.77	92.5%	1.34	0.06	32.3%	2.30	62.4%	1.82
		LLaDA (T=128)	0.46	55.8%	4.18	0.04	19.4%	4.80	37.6%	4.49
	AutoReg.	Baseline	0.67	100.0%	1.00	0.40	100.0%	1.00	100.0%	1.00
	Spec. Decoding	PLD Saxena (2023)	0.67	100.0%	1.44	0.40	100.0%	1.56	100.0%	1.50
		Lookahead Fu et al. (2024)	0.67	100.0%	1.71	0.40	100.0%	1.72	100.0%	1.72
		Medusa Cai et al. (2024)	0.67	100.0%	2.42	0.40	100.0%	2.32	100.0%	2.37
		Eagle Li et al. (2024)	0.67	100.0%	3.61	0.40	100.0%	3.76	100.0%	3.69
	Cross-Regression	Lossless	0.67	100.0%	3.95	0.40	100.0%	2.98	100.0%	3.46
		Lossy ($\varepsilon \leq 14$)	0.66	**98.0%**	**5.03**	0.40	**101.5%**	**3.59**	**99.8%**	**4.31**
Qwen2.5 7B	Diffusion	Dream (T=1024)	0.48	100.0%	0.09	0.30	100.0%	0.05	100.0%	0.07
		Dream (T=64)	0.32	66.3%	1.14	0.07	22.4%	0.53	44.3%	0.83
		Dream (T=16)	0.10	20.1%	2.44	0.10	32.5%	2.44	26.3%	2.44
	AutoReg.	Baseline	0.84	100.0%	1.00	0.22	100.0%	1.00	100.0%	1.00
	Cross-Regression	Lossless	0.84	100.0%	4.48	0.22	100.0%	2.86	100.0%	3.67
		Lossy ($\varepsilon \leq 8$)	0.74	88.6%	5.46	0.29	132.5%	3.13	110.6%	4.29
		Lossy ($\varepsilon \leq 12$)	0.69	**83.0%**	**5.82**	0.23	**104.9%**	**3.23**	**94.0%**	**4.52**

Table 3: **Speed-quality trade-off analysis across decoding frameworks.** Cross-Regression achieves Pareto-optimal performance, with leading results in **bold**. *Speedup* quantifies tokens accepted per forward pass; for diffusion methods, it equals completion length divided by denoising steps. Metrics shown as absolute values (gray) and relative to underlined baselines. Low-temperature sampling setting.[1]

Table 4: Cross-Regression speedup factors on Whisper Radford et al. (2023) models for speech-to-text generation on Librispeech test clean Panayotov et al. (2015). Speedup is measured as the average number of tokens accepted per forward pass.

Sampling	Whisper Tiny	Whisper Small	Whisper Medium
Temperature 0	1.5×	2.4×	2.7
Temperature 1	1.3×	2.1×	2.3

Comparison with Speculative Decoding. On LLaMA3 8B, Cross-Regression's **lossless mode** ($\tau = 3.46$) is competitive with the state-of-the-art EAGLE ($\tau = 3.69$). Crucially, the **lossy mode** achieves superior acceleration ($\tau = 4.31$) with negligible quality loss (99.8% retention), establishing a new Pareto frontier that surpasses existing lossless methods.

Advantage over Discrete Diffusion. The comparison with the diffusion framework is even more telling, as it reveals a fundamental weakness in existing parallel decoding strategies. Diffusion-based methods exhibit a severe speed-quality trade-off. For instance, to achieve a high acceptance length ($\tau = 4.49$), the LLaDA model's relative quality plummets to just 37.6%, with similar catastrophic degradation observed across other diffusion models and settings. In stark contrast, Cross-Regression achieves a comparable acceleration ($\tau = 4.31$) while preserving virtually all of the original model's performance. This highlights a critical advantage of our hybrid, iterative refinement approach: Cross-Regression's ability to aggressively accelerate inference without the uncontrolled and precipitous quality degradation inherent in rigid, multi-step parallel decoding schedules.

REPRODUCIBILITY STATEMENT

We provide complete algorithmic details in Algorithms 1 and 2, with full proofs of theoretical results in Appendices D.1 and D.2. Training hyperparameters, model configurations, and evaluation protocols are detailed in Appendix B. Code and trained adapters will be released upon publication.

ETHICS STATEMENT

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here. Cross-Regression accelerates inference without modifying the underlying model distribution in lossless mode, thus inheriting the safety properties (or lack thereof) of the base model.

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, May 2025. URL http://arxiv.org/abs/2503.09573. arXiv:2503.09573 [cs].

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.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D. Lee, Deming Chen, and Tri Dao. Medusa: Simple llm inference acceleration framework with multiple decoding heads. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

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.

Jacob K. Christopher, Brian R. Bartoldson, Tal Ben-Nun, Michael Cardei, Bhavya Kailkhura, and Ferdinando Fioretto. Speculative Diffusion Decoding: Accelerating Language Generation through Diffusion, February 2025. URL http://arxiv.org/abs/2408.05636. arXiv:2408.05636 [cs].

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.

Nima Fathi, Torsten Scholak, and Pierre-André Noël. Unifying Autoregressive and Diffusion-Based Sequence Generation. *arXiv preprint arXiv:2504.06416*, 2025. doi: 10.48550/ARXIV.2504.06416. URL https://arxiv.org/abs/2504.06416. Publisher: arXiv Version Number: 1.

Yichao Fu, Peter Bailis, Ion Stoica, and Hao Zhang. Break the sequential dependency of llm inference using lookahead decoding. *arXiv preprint arXiv:2402.02057*, 2024.

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.

Fabian Gloeckle, Badr Youbi Idrissi, Baptiste Rozière, David Lopez-Paz, and Gabriel Synnaeve. Better & faster large language models via multi-token prediction. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Gabe Guo and Stefano Ermon. Reviving Any-Subset Autoregressive Models with Principled Parallel Sampling and Speculative Decoding, April 2025. URL http://arxiv.org/abs/2504.20456. arXiv:2504.20456 [cs].

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. Argmax flows and multinomial diffusion: Learning categorical distributions. *Advances in Neural Information Processing Systems*, 34:12454–12465, 2021.

Chihan Huang and Hao Tang. CtrlDiff: Boosting Large Diffusion Language Models with Dynamic Block Prediction and Controllable Generation. *arXiv preprint arXiv:2505.14455*, 2025. doi: 10.48550/ARXIV.2505.14455. URL https://arxiv.org/abs/2505.14455. Publisher: arXiv Version Number: 1.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Lingkai Kong, Jiaming Cui, Haotian Sun, Yuchen Zhuang, B. Aditya Prakash, and Chao Zhang. Autoregressive diffusion model for graph generation. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 17391–17408. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/kong23b.html.

Siqi Kou, Lanxiang Hu, Zhezhi He, Zhijie Deng, and Hao Zhang. Cllms: Consistency large language models. In *Forty-first International Conference on Machine Learning*, 2024.

Inception Labs, Samar Khanna, Siddhant Kharbanda, Shufan Li, Harshit Varma, Eric Wang, Sawyer Birnbaum, Ziyang Luo, Yanis Miraoui, Akash Palrecha, et al. Mercury: Ultra-fast language models based on diffusion. *arXiv preprint arXiv:2506.17298*, 2025.

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

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast Inference from Transformers via Speculative Decoding, May 2023b. URL http://arxiv.org/abs/2211.17192. arXiv:2211.17192 [cs].

Yifan Li, Kun Zhou, Wayne Xin Zhao, and Ji-Rong Wen. Diffusion Models for Non-autoregressive Text Generation: A Survey, May 2023. URL http://arxiv.org/abs/2303.06574. arXiv:2303.06574 [cs].

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE: Speculative Sampling Requires Rethinking Feature Uncertainty, February 2024. URL http://arxiv.org/abs/2401.15077. arXiv:2401.15077 [cs].

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, Jan Kautz, Ce Zhang, and Pavlo Molchanov. Tidar: Think in diffusion, talk in autoregression. *arXiv preprint arXiv:2511.08923*, 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*, 2025a.

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*, 2025b.

Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: an asr corpus based on public domain audio books. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pp. 5206–5210. IEEE, 2015.

Arnaud Pannatier, Evann Courdier, and François Fleuret. {\sigma}-GPTs: A New Approach to Autoregressive Models, April 2024. URL http://arxiv.org/abs/2404.09562. arXiv:2404.09562 [cs].

Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision. In *International conference on machine learning*, pp. 28492–28518. PMLR, 2023.

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, June 2025. URL http://arxiv.org/abs/2506.01928. arXiv:2506.01928 [cs].

Apoorv Saxena. Prompt lookup decoding, November 2023. URL https://github.com/apoorvumang/prompt-lookup-decoding/.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*, 2019.

Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. Blockwise parallel decoding for deep autoregressive models. *CoRR*, abs/1811.03115, 2018. URL http://arxiv.org/abs/1811.03115.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv: 2307.09288*, 2023.

Zirui Wu, Zirui Liu, Jiacheng Xie, Jialui Jiao, Yansong Gao, Zhenguo Feng, W. Victoria, Guorui Zhou, and Lingpeng Kong. Dreamon: Diffusion language models for code infilling beyond fixed-size canvas. https://hku-nlp.github.io/blog/2025/dreamon, 2025.

Heming Xia, Zhe Yang, Qingxiu Dong, Peiyi Wang, Yongqi Li, Tao Ge, Tianyu Liu, Wenjie Li, and Zhifang Sui. Unlocking Efficiency in Large Language Model Inference: A Comprehensive Survey of Speculative Decoding, June 2024. URL http://arxiv.org/abs/2401.07851. arXiv:2401.07851 [cs].

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*, 2025.

A APPENDIX

A.1 SCALING LAWS AND TASK-DEPENDENT ACCELERATION

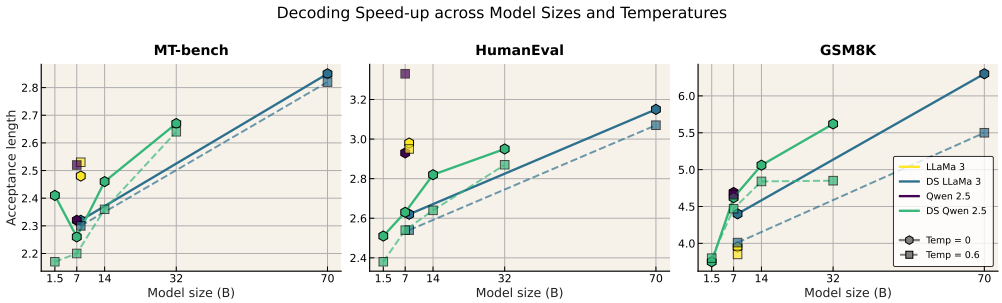

Figure 1: Scaling law of speed-up as a function of model parameter count. Results are aggregated over GSM8K, HumanEval, and MT-bench datasets with temperatures $T \in \{0, 0.6\}$.

We investigate the scalability of Cross-Regression by evaluating inference acceleration across a wide range of model sizes, from 1.5B to 70B parameters. As illustrated in Figure 1, we observe a strong positive correlation between model scale and decoding speed-up across all benchmarks. Larger models consistently exhibit higher acceleration factors; for instance, on the GSM8K benchmark, the speed-up factor rises steeply from $\approx 3.8\times$ for the 1.5B model to a peak of $6.3\times$ for the 70B model. This trend suggests that the richer latent representations in larger transformers enable the predictive stream to draft future tokens with significantly higher fidelity, thereby minimizing interventions by the control stream.

The magnitude of this acceleration is heavily task-dependent. Structured reasoning tasks like GSM8K and code generation (HumanEval) yield the highest gains ($3.0\times$–$6.3\times$), as their logical sequential dependencies are more amenable to parallel prediction. Conversely, open-ended conversational tasks (MT-bench) exhibit more modest speed-ups ($2.0\times$–$2.8\times$), reflecting the higher entropy and lower predictability of unconstrained dialogue. Furthermore, the method demonstrates robustness to stochasticity: while greedy decoding ($T = 0$) yields the highest throughput, increasing the temperature to $T = 0.6$ results in only a marginal reduction in speed-up, confirming that Cross-Regression effectively models the target distribution even in non-deterministic settings.

B EXTENDED TRAINING DETAILS

We fine-tune a lightweight, parameter-efficient LoRA adapter that equips the *predictive stream* $\mathcal{M}_q$ with the ability to forecast *multi-token* futures, while the *control stream* $\mathcal{M}_p$ remains frozen. The adapter increases the total parameter count by well under one percent, so memory footprint at inference time is effectively unchanged. Because only the adapter weights are updated, the optimisation process requires only a fraction of the compute budget needed for full-model fine-tuning, making the training cost negligible in practice. Freezing $\mathcal{M}_p$ also prevents catastrophic forgetting, ensuring that the original distribution of the base model is preserved and can be recovered exactly in lossless decoding mode.

Data Preparation For every prefix $x_{1:i}$ extracted from the raw corpus, the frozen base model $\mathcal{M}_p$ is asked to generate a continuation of exactly $w = 16$ tokens. The pair $(x_{1:i}, \hat{y}_{i+1:i+w})$ is then treated as a supervised example for the adapter, following the look-ahead distillation paradigm. Ablation studies confirmed that $w = 16$ offers the best speed–quality trade-off during training.

Objective Function We employ a weighted distillation objective that encourages the predictive stream to match the target distribution of the future tokens. The loss function , with s_t is a weight derived from the matching suffix length, is defined as:

$$\mathcal{L} = -\frac{\sum_{t=1}^{T}(s_t + 1) \cdot p_t^\top \log q_t}{\sum_{t=1}^{T}(s_t + 1)}$$

Throughout our experiments we use checkpoints from three backbone families: LLaMA-2 (Touvron et al., 2023), Qwen 2.5 (Yang et al., 2025), and the distilled DeepSeek-AI models (Guo et al., 2025). This diversity demonstrates that the proposed training recipe is agnostic to pre-training corpus and architectural refinements.

The remainder of this section details the data pipeline, objective function, and optimisation hyper-parameters.

Algorithm 3 LoRA Denoising Distillation

Require: Dual-head model $\mathcal{M}$: frozen target head P_θ, trainable draft head $Q_{\theta+\Delta\phi}$
Require: Dataset $\mathcal{D}$, replay buffer $\mathcal{B}$
Ensure: Trained adapter $\Delta\phi$

for batch $(\mathbf{x}_{1:T}, \mathrm{id}) \sim \mathcal{D}$ **do**
  ▷ *Sample from implicit prior*
  $\acute{\mathbf{x}}_{1:T} \leftarrow \mathcal{B}[\mathrm{id}]$ **if exists else** $\mathbf{x}_{1:T}$
  $\acute{\mathbf{x}}_{1:T} \leftarrow RandomNoise(\acute{\mathbf{x}}_{1:T})$
  ▷ *Parallel forward with divergent contexts*
  $\{p_t\}_{t=1}^{T} \leftarrow P_\theta(\mathbf{x}_{1:T})$ ▷ Target on ground truth
  $\{q_t\}_{t=1}^{T} \leftarrow Q_{\theta+\Delta\phi}(\tilde{\mathbf{x}}_{1:T})$ ▷ Draft on prior samples
  ▷ *Matching suffix lengths via backward recurrence*
  $s_T \leftarrow 0, \quad s_t \leftarrow (s_{t+1}+1) \cdot \mathbf{1}[\arg\max q_{t+1} = x_{t+1}] \quad$ for $t = T-1, \ldots, 1$
  ▷ *Weighted distillation*
  $\mathcal{L} \leftarrow -\sum_{t=1}^{T}(s_t+1) \cdot p_t^\top \log q_t \Big/ \sum_{t=1}^{T}(s_t+1)$
  Update $\Delta\phi$ via $\nabla_{\Delta\phi}\mathcal{L}$
  ▷ *Evolve implicit prior*
  $\mathcal{B}[\mathrm{id}] \leftarrow [\arg\max q_t]_{t=1}^{T}$ ▷ Update replay buffer
end for
return $\Delta\phi$

B.1 Data Preparation

Self-generated targets. For every prefix $x_{1:i}$ extracted from the raw corpus, the frozen base model $\mathcal{M}_p$ is asked to generate a continuation of exactly $w = 16$ tokens. The pair $(x_{1:i}, \hat{y}_{i+1:i+w})$ is then treated as a supervised example for the adapter, following the look-ahead distillation paradigm. Ablation studies confirmed that $w = 16$ offers the best speed–quality trade-off during training.

Corpora. We train two separate adapters, each on a distinct 7k-task dataset. The first dataset consists solely of GSM8K arithmetic-reasoning problems; the second is a balanced mixture of UltraChat, ShareGPT, and BaGel instructions, giving a multi-domain "Mixed-7k" corpus. All texts are tokenised with the backbone's SentencePiece vocabulary and clipped to 2048 tokens so the 16-token future always fits in context.

Temperature settings. We experiment with two temperature regimes that are applied consistently during both training and validation. In the first regime, generation targets are produced with greedy decoding ($T = 0.0$) and the same setting is used at validation time, yielding a deterministic baseline. In the second regime, targets are sampled with temperature $T = 0.6$, again matched by validation at $T = 0.6$, which improves the model's ability to handle open-ended sampling. Results for both regimes are reported in the ablation tables.

Benchmarks. The resulting adapters are evaluated on four test suites: GSM8K for mathematical reasoning, HumanEval for code generation, MBPP for programming tasks, and ShareGPT for open-domain instruction following. This spectrum gauges generalisation from purely self-distilled supervision to structurally diverse evaluation domains.

Parameter	Value
LoRA rank	32
LoRA target	all-linear
Training window w	16
Partial-correct threshold	0.05
Optimizer	AdamW
Learning rate	5×10^{-4} (cosine, 3% warm-up)
Weight-decay	0
Batch size	8 seq/H100 (grad-accum. 1)
Precision	bf16
Gradient checkpointing	True
Gradient clip	1.0
Epochs	5 (search) / 20 (final)
Early-stopping	patience 5, threshold 0
Model max length	2048
Attention implementation	SDPA

Table 5: Unified training-script parameters.

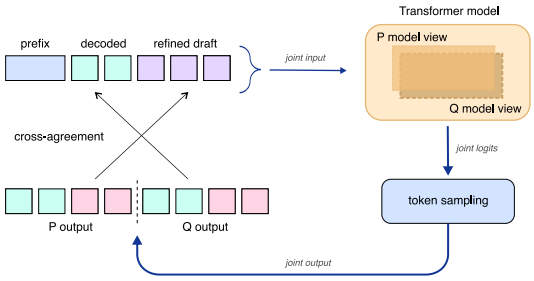

Figure 2: A simplified schematic of the Cross-Regression decoding process. It illustrates the dual-stream architecture where a non-causal predictive stream drafts future tokens that are validated by a causal control stream via a cross-agreement procedure, enabling simultaneous drafting and acceptance.

B.2 Optimisation Hyper-parameters

Training leverages LoRA with rank 32 and targets all linear layers. We use the AdamW optimiser (weight-decay $= 0$) with an initial learning rate of 5×10^{-4} scheduled by cosine decay after a 3% warm-up phase, batch size 16 per H100 GPU, and gradient accumulation set to 1. Checkpointing is enabled to fit the full 2048-token context. Runs default to bf16 precision on H100; fp16 is used only on earlier hardware. The pilot grid search employed five epochs with early-stopping (patience $= 5$, threshold $= 0$) to find the best learning-rate region. Final headline results are reported after a fixed 20-epoch training pass under the same hyper-parameters.

B.3 Inference Windows

The predictive horizon equals the training window ($\gamma = w = 16$) by default, but can be increased to 32 tokens at inference without destabilising the model.

C Technical Details

Correctness of Carried-Over Draft Probabilities in Predictive Stream

A subtle but critical detail distinguishes our implementation from the naïve formulation. When draft tokens are carried over from a previous iteration (positions $1, \ldots, m$), they were sampled from the predictive distribution conditioned on a *different* prefix—specifically, the context before the correction token x_{corr} was appended. If we were to re-evaluate these tokens under the current Q_k, we would be computing $Q(\acute{x}_k \mid \mathbf{x}, x_{\text{corr}}, \acute{\mathbf{x}}_{1:k-1})$, which differs from the distribution $Q(\acute{x}_k \mid \mathbf{x}^{(\text{prev})}, \acute{\mathbf{x}}_{1:k-1})$

Model	Size	Acceptance Rate	X-regression Accuracy
Temperature 0 (Greedy)			
DeepSeek-Qwen	7B	4.62	60.0
LLaMA-3	8B	3.95	60.0
Vicuna	33B	3.02	28.3
Qwen2.5	7B	4.69	82.0
Temperature 0.6			
DeepSeek-Qwen	7B	4.47	60.0
LLaMA-2	7B	3.15	24.2
LLaMA-3	8B	3.85	61.0
Qwen2.5	7B	4.66	82.0
Vicuna	33B	2.86	21.0

Table 6: Acceptance rate on GSM8K with GSM8K-trained adapter. Test window is 32, train window is 16, without look-ahead.

Model	Size	Acceptance Rate
Temperature 0 (Greedy)		
DeepSeek-LLaMA	8B	2.32
DeepSeek-Qwen	1.5B	2.41
DeepSeek-Qwen	7B	2.26
DeepSeek-Qwen	14B	2.46
DeepSeek-Qwen	32B	2.67
LLaMA-2	7B	2.39
LLaMA-3	8B	2.48
Vicuna	33B	2.19
Qwen2.5	7B	2.32
Temperature 0.6		
DeepSeek-LLaMA	8B	2.30
DeepSeek-Qwen	1.5B	2.17
DeepSeek-Qwen	7B	2.20
DeepSeek-Qwen	14B	2.36
DeepSeek-Qwen	32B	2.64
LLaMA-2	7B	2.48
LLaMA-3	8B	2.53
Vicuna	33B	2.17
Qwen2.5	7B	2.52

Table 7: Acceptance rate on MT-Bench with mixed-trained adapter. Test window is 32, train window is 16, without look-ahead.

from which they were actually drawn. This mismatch violates the importance sampling assumption underlying the acceptance criterion: the energy $E_k = \log p_k - \log q_k$ is only a valid importance weight when q_k reflects the true proposal density. To preserve correctness, we cache each $\acute{q}_k = Q_k(\acute{x}_k)$ at the moment of sampling and carry these cached probabilities alongside the draft tokens themselves. During verification, only the target probabilities p_k are re-evaluated against the updated context, while the proposal probabilities $\acute{q}_k$ remain fixed to their original values. This ensures that the energy-based acceptance test correctly measures divergence between where the token came from and where the control stream currently places probability mass.

Implementation Note: Fixed-Length Draft Buffer. The practical implementation differs from a purely conceptual presentation in how the draft buffer is managed. To enable efficient batched forward passes through the dual-steam model, the draft buffer $\acute{\mathbf{x}}$ maintains a fixed length of γ tokens throughout execution rather than dynamically resizing. Modern transformer inference exploits parallelism via fixed-shape tensor operations; a variable-length buffer would require costly dynamic reallocation or padding overhead on each iteration. We therefore initialize $\acute{\mathbf{x}}$ with γ random placeholder tokens before decoding begins and track the count of valid carried-over tokens m as a separate variable. The

Model	Size	Acceptance Rate
Temperature 0 (Greedy)		
DeepSeek-LLaMA	8B	2.62
DeepSeek-Qwen	1.5B	2.51
DeepSeek-Qwen	7B	2.63
DeepSeek-Qwen	14B	2.82
DeepSeek-Qwen	32B	2.95
LLaMA-2	7B	2.62
LLaMA-3	8B	2.98
Vicuna	33B	2.42
Qwen2.5	7B	2.93
Temperature 0.6		
DeepSeek-LLaMA	8B	2.54
DeepSeek-Qwen	1.5B	2.38
DeepSeek-Qwen	7B	2.54
DeepSeek-Qwen	14B	2.64
DeepSeek-Qwen	32B	2.87
LLaMA-2	7B	2.71
LLaMA-3	8B	2.95
Vicuna	33B	3.13
Qwen2.5	7B	3.33

Table 8: Acceptance rate on HumanEval with mixed-trained adapter. Test window is 32, train window is 16, without bideirectional attention mask.

Model	Temp	Accuracy	Accept Rate
Qwen2.5 7B	0.0	82.0	4.69
Qwen2.5 7B	0.6	82.0	4.67
Qwen2.5 7B	1.0	84.0	4.37
LLaMA-3 8B	0.0	60.0	3.95
LLaMA-3 8B	0.6	61.0	3.83
LLaMA-3 8B	1.0	60.6	3.52
DeepSeek-Qwen 7B	0.0	60.0	4.62
DeepSeek-Qwen 7B	0.6	60.0	4.47
DeepSeek-Qwen 7B	1.0	55.0	3.85
Vicuna 33B	0.0	28.3	3.02
Vicuna 33B	0.6	21.0	2.93
Vicuna 33B	1.0	25.0	2.68
LLaMA-2 7B	0.0	7.1	3.21
LLaMA-2 7B	0.6	24.2	3.02
LLaMA-2 7B	1.0	22.2	2.77

Table 9: GSM8K to GSM8K evaluation under standard configuration with window 32 and threshold 0.

random initial tokens carry no semantic weight—they are immediately overwritten during the first sampling phase since fresh sampling across all γ positions. On subsequent iterations, positions 1 through m retain valid carried-over candidates from the previous draft while positions $m+1$ through γ are resampled. When rejection occurs, unverified tokens are shifted to the buffer's front (rather than creating a new shorter sequence), and m is updated accordingly. This fixed-buffer strategy preserves algorithmic correctness while enabling static memory allocation and consistent batch shapes for GPU-efficient inference.

Model	MATH500	GSM8K	HumanEval	MBPP
DeepSeek-R1-Distill-Qwen-32B	94.3%	82.7%*	86.0%*	–
DeepSeek-R1-Distill-Qwen-14B	93.9%	87.0%*	78.9%*	–
DeepSeek-R1-Distill-Llama-8B	89.1%	62.8%*	49.9%*	–
DeepSeek-R1-Distill-Qwen-7B	92.8%	78.6%*	40.8%*	–
DeepSeek-R1-Distill-Qwen-1.5B	83.9%	70.0%*	37.9%*	–
Vicuna 33B	–	38.7%*	–	–
LLaMA-2 7B	–	30.8%*	14.0%	26.1%
LLaMA-3 8B	51.9%	80.6%	72.6%	72.8%
Qwen2.5 7B	49.8%	85.4%	57.9%	74.9%

Table 10: Original model metrics on benchmarks. Values marked with * are from third-party sources; unmarked values are official.

Model	Threshold	Smoothing	Accept Rate	Accuracy
LLaMA-3 8B	0	0.05	4.02	67.5
	4	0.05	4.34	67.0
	4	0.2	4.32	67.0
	4	0.5	4.16	66.8
	8	0.05	4.64	65.9
	8	0.2	4.63	66.4
	8	0.5	4.78	66.5
	12	0.05	4.92	66.3
Qwen2.5 7B	0	0.05	4.82	6.0
	4	0.05	5.20	2.4
	4	0.2	5.16	2.3
	4	0.5	4.97	61.1
	8	0.05	5.46	13.4
	8	0.2	5.46	74.1
	8	0.5	5.62	62.9
	12	0.05	5.73	68.0
ABEL-7B-001	0	0.05	4.69	56.5
	4	0.05	4.77	57.0
	4	0.2	4.75	57.0
	4	0.5	4.72	56.9
	8	0.05	4.83	57.8
	8	0.2	4.83	57.6
	8	0.5	4.87	55.7
	12	0.05	4.89	56.9
LLaMA-2 7B	0	0.05	3.13	6.9
	4	0.05	3.34	3.3
	4	0.5	3.23	1.7
	8	0.2	3.53	3.6
	8	0.5	3.55	1.2
	12	0.05	3.65	8.7
	12	0.2	3.68	3.2
	16	0.05	3.77	2.5

Table 11: Acceptance rate on GSM8K with GSM8K-trained adapter. Test window is 32, train window is 16. Trained and evaluated at temperature 0.1.

D MISSING PROOFS

Let $p(x)$ denote a target autoregressive distribution and $q(x)$ a proposal distribution. Cross-regressive decoding generates tokens by proposing from q and accepting based on an energy criterion controlled by smoothing parameter $\beta \in [0, 1)$.

Definition D.1 (Energy)**.** *For a proposed token $\acute{x}$, the energy is $E = \log p(\acute{x}) - \log q(\acute{x})$.*

Model	Threshold	Smoothing	Accept Rate	Pass Rate
LLaMA-3 8B	0	0.05	3.01	39.6
	0	0.2	3.01	39.6
	0	0.5	3.01	39.6
	4	0.05	3.10	40.2
	4	0.2	3.08	39.6
	4	0.5	2.99	40.9
	8	0.05	3.24	40.2
	8	0.2	3.24	40.2
Qwen2.5 7B	0	0.05	2.86	22.1
	0	0.2	2.86	22.1
	0	0.5	2.86	22.1
	4	0.05	2.99	28.7
	4	0.2	2.96	28.0
	4	0.5	2.90	27.4
	8	0.05	3.13	29.3
	8	0.2	3.11	27.6
LLaMA-2 7B	0	0.05	2.50	1.2
	0	0.2	2.50	1.2
	0	0.5	2.50	1.2
	4	0.05	2.69	1.2
	4	0.5	2.65	0.0
	8	0.05	2.80	1.8
	8	0.5	2.77	0.0
	12	0.05	2.88	1.2

Table 12: Acceptance rate on HumanEval with mixed-trained adapter. Test window is 32, train window is 16. Trained and evaluated at temperature 0.1.

Model / Method	MT-bench τ	HumanEval τ	GSM8K τ
LLaMA 2 7B			
PLD (Saxena, 2023)	1.38	1.52	1.32
Lookahead (Fu et al., 2024)	1.61	1.72	1.58
CLLM (Kou et al., 2024)	<u>2.40</u>	<u>3.00</u>	1.10
Medusa (Cai et al., 2024)	**2.70**	2.88	<u>1.60</u>
Cross-Regression	2.67	**3.50**	**3.47**
LLaMA 2 13B			
PLD (Saxena, 2023)	1.42	1.63	1.41
Lookahead (Fu et al., 2024)	1.58	1.80	1.65
Cross-Regression	**2.87**	**4.40**	**3.90**

Table 13: Average acceptance lengths τ of different speculative decoding methods for standard language models as reported in the relevant papers. **Bold** indicates the best result for each benchmark; <u>underline</u> indicates the second best.

Definition D.2 (Residual Distribution)**.** *The residual distribution for correction sampling is*

$$p_{\text{res}}(v) := \frac{\max(0, p(v) - q(v))}{\sum_{v'} \max(0, p(v') - q(v'))}. \tag{4}$$

D.1 Proof of Theorem 4.2

Proof. We proceed in three steps: (1) show that $\beta = 0$ reduces cross-regressive decoding to standard speculative decoding, (2) verify the acceptance criterion produces the correct mixture, and (3) extend to full sequences by induction.

Step 1: Reduction to Speculative Decoding.

	Sequential Draft and Validation		**Joint Draft and Validation**
Benchmark	Acceptance Length	Tokens/Fwds	Acceptance Length
Llama2-7B-Chat			
GSM8K	5.07	2.54	3.47
HumanEval	4.08	2.04	3.50
MT-Bench	3.91	1.95	2.67
Llama2-13B-Chat			
GSM8K	5.75	2.88	3.90
HumanEval	4.95	2.48	4.40
MT-Bench	4.06	2.03	2.87

Table 14: Comparison between sequential drafting and validation passes versus joint processing in a single pass layout.

Layer Range	**#Layers**	**Occupancy**	$\tau_{\textbf{Seq}}$	$\tau_{\textbf{Joint}}$
Layers 0–3	4		3.4	2.3
Layers 4–7	4		3.7	2.4
Layers 8–11	4		3.9	2.6
Layers 12–15	4		3.9	2.6
Layers 16–19	4		3.8	2.5
Layers 20–23	4		3.5	2.3
Layers 24–27	4		3.3	2.2
Layers 28–31	4		3.1	2.1
Layers 0–7	8		3.9	2.6
Layers 8–15	8		4.2	2.8
Layers 16–23	8		4.0	2.8
Layers 24–31	8		3.5	2.4
Layers 0–15	16		4.3	2.9
Layers 8–23	16		**4.6**	**3.2**
Layers 16–31	16		4.2	2.9
Layers 0–23	24		4.6	3.2
Layers 8–31	24		**4.8**	**3.3**
Full (baseline)	32		4.7	3.3

Table 15: Acceptance length τ with LoRA applied to different layer ranges in Llama-2-7B-chat, evaluated on GSM8K. The *Occupancy pattern* column visually represents which portions of the 32-layer architecture have LoRA applied (black sections).

With $\beta = 0$, the EMA recursion simplifies to

$$\rho_k = (1 - 0) \cdot E_k + 0 \cdot \rho_{k-1} = E_k. \tag{5}$$

The bias-correction factor is $1 - \beta^k = 1 - 0^k = 1$ for all $k \geq 1$. Thus the smoothed estimate equals the instantaneous energy:

$$\hat{\rho}_k = \frac{\rho_k}{1 - \beta^k} = E_k = \log p_k - \log q_k. \tag{6}$$

The cumulative energy test reduces to checking whether

$$S_n = \sum_{k=1}^{n} E_k = \sum_{k=1}^{n} \log \frac{p_k}{q_k} \leq \tau. \tag{7}$$

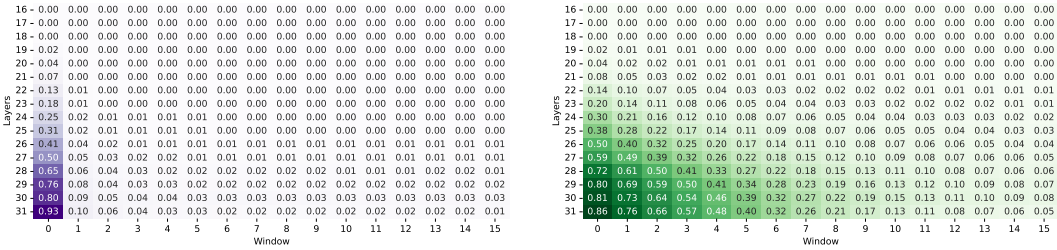

Figure 3: Comparison of per-layer output from vanilla Llama-2-7B-chat (left) and LoRA-tuned (right). Averaged over text generations based on 50 prompts from GSM8K dataset (Cobbe et al., 2021).

Model Series	**Size**	**MT-b**	**HE**	**GSM8K**	**Mean**
		τ	τ	τ	τ
Temperature=0					
LLaMa 3	8B	2.48	2.98	3.95	3.14
DS LLaMa 3	8B	2.32	2.62	4.40	3.11
Qwen 2.5	7B	2.32	2.93	4.69	3.31
DS Qwen 2.5	1.5B	2.41	2.51	3.75	2.89
	7B	2.26	2.63	4.62	3.17
	14B	2.46	2.82	5.06	3.45
	32B	2.67	2.95	5.62	3.75
Temperature=0.6					
LLaMa 3	8B	2.53	2.95	3.85	3.11
DS LLaMa 3	8B	2.30	2.54	4.01	2.95
Qwen 2.5	7B	2.52	3.33	4.66	3.50
DS Qwen 2.5	1.5B	2.17	2.38	3.80	2.78
	7B	2.20	2.54	4.47	3.07
	14B	2.36	2.64	4.84	3.28
	32B	2.64	2.87	4.85	3.45

Table 16: Cross-Regression acceptance length (τ) across different model series, sizes and sampling temperatures. Performance is evaluated on MT-bench (MT-b), HumanEval (HE), and GSM8K. The DS prefix denotes models finetuned by DeepSeek.

Writing $\tau = -\log u$ for $u \sim \mathrm{Uniform}(0, 1)$, the condition $S_n \leq \tau$ becomes

$$\exp(S_n) = \prod_{k=1}^{n} \frac{p_k}{q_k} \geq u. \tag{8}$$

This is precisely the standard speculative decoding acceptance criterion.

Step 2: Single-Token Correctness.

For a single position, let $\acute{x} \sim q$ and $u \sim \mathrm{Uniform}(0, 1)$ be independent. The acceptance probability is

$$\mathbb{P}\left(\frac{p(\acute{x})}{q(\acute{x})} \geq u\right) = \mathbb{E}_{\acute{x}\sim q}\left[\min\left(1, \frac{p(\acute{x})}{q(\acute{x})}\right)\right] = \sum_{v\in\mathcal{V}} q(v) \cdot \min\left(1, \frac{p(v)}{q(v)}\right) = \sum_{v\in\mathcal{V}} \min(p(v), q(v)). \tag{9}$$

Let $Z \coloneqq \sum_v \min(p(v), q(v))$. The accepted samples follow distribution

$$\mathbb{P}(x = v \mid \text{accept}) = \frac{q(v) \cdot \min(1, p(v)/q(v))}{Z} = \frac{\min(p(v), q(v))}{Z}. \tag{10}$$

Upon rejection (probability $1 - Z$), correction sampling draws from

$$p_{\text{res}}(v) = \frac{\max(0, p(v) - q(v))}{\sum_{v'} \max(0, p(v') - q(v'))} = \frac{(p(v) - q(v))_+}{1 - Z}, \tag{11}$$

where we used $\sum_v (p(v) - q(v))_+ = 1 - Z$ (since $\sum_v p(v) = \sum_v q(v) = 1$).

The overall distribution is the mixture:

$$\begin{aligned}
\mathbb{P}(x = v) &= Z \cdot \frac{\min(p(v), q(v))}{Z} + (1 - Z) \cdot \frac{(p(v) - q(v))_+}{1 - Z} && (12)\\
&= \min(p(v), q(v)) + (p(v) - q(v))_+ && (13)\\
&= p(v), && (14)
\end{aligned}$$

where the last equality holds because $\min(a, b) + (a - b)_+ = a$ for all $a, b \geq 0$.

Step 3: Extension to Full Sequences.

We prove by induction on sequence length that the joint distribution over accepted sequences equals p_{AR}.

Base case: For the first token, Step 2 shows $x_1 \sim p(\cdot \mid \mathbf{x}_0)$.

Inductive step: Suppose $\mathbf{x}_{1:n} \sim p_{\text{AR}}(\cdot \mid \mathbf{x}_0)$. For position $n+1$, the proposal $\acute{x}_{n+1} \sim q(\cdot \mid \mathbf{x}_{1:n}, \tilde{\mathbf{x}})$ is accepted or corrected using the same mechanism as Step 2, but with target distribution $p(\cdot \mid \mathbf{x}_0, \mathbf{x}_{1:n})$. By Step 2, this yields $x_{n+1} \sim p(\cdot \mid \mathbf{x}_0, \mathbf{x}_{1:n})$.

The KV cache ensures that conditioning on $\mathbf{x}_{1:n}$ is exact, not approximate. Therefore,

$$\mathbb{P}(\mathbf{x}_{1:n+1}) = \mathbb{P}(\mathbf{x}_{1:n}) \cdot p(x_{n+1} \mid \mathbf{x}_0, \mathbf{x}_{1:n}) = p_{\text{AR}}(\mathbf{x}_{1:n+1} \mid \mathbf{x}_0). \tag{15}$$

By induction, $\pi(\mathbf{x}_{1:T}) = p_{\text{AR}}(\mathbf{x}_{1:T} \mid \mathbf{x}_0)$ for all T, completing the proof. □

D.2 Proof of Theorem 4.4

Proof. We establish the expected acceptance length gain through analysis of the EMA-smoothed energy process.

Step 1: Variance Reduction from EMA.

The EMA recursion defines the filtered process:

$$\rho_k = \beta \rho_{k-1} + (1 - \beta) E_k, \quad \rho_0 = 0. \tag{16}$$

Unrolling the recursion:

$$\rho_k = (1 - \beta) \sum_{j=0}^{k-1} \beta^j E_{k-j}. \tag{17}$$

Under Theorem 4.3 (uncorrelated increments), the variance is:

$$\begin{aligned}
\text{Var}(\rho_k) &= (1 - \beta)^2 \sum_{j=0}^{k-1} \beta^{2j} \text{Var}(E_{k-j}) && (18)\\
&= (1 - \beta)^2 \sigma^2 \sum_{j=0}^{k-1} \beta^{2j} && (19)\\
&= (1 - \beta)^2 \sigma^2 \cdot \frac{1 - \beta^{2k}}{1 - \beta^2}. && (20)
\end{aligned}$$

As $k \to \infty$:

$$\text{Var}(\rho_k) \to \frac{(1 - \beta)^2 \sigma^2}{1 - \beta^2} = \frac{(1 - \beta)^2 \sigma^2}{(1 - \beta)(1 + \beta)} = \frac{(1 - \beta)\sigma^2}{1 + \beta}. \tag{21}$$

The bias-corrected estimate $\hat{\rho}_k = \rho_k/(1-\beta^k)$ has variance:

$$\mathrm{Var}(\hat{\rho}_k) = \frac{\mathrm{Var}(\rho_k)}{(1-\beta^k)^2} \to \frac{(1-\beta)\sigma^2}{1+\beta} \quad \text{as } k \to \infty. \tag{22}$$

Define $\sigma_\beta^2 \coloneqq (1-\beta)\sigma^2/(1+\beta)$ as the effective variance under EMA smoothing.

Step 2: Impact on Survival Probability.

The key insight is that EMA smoothing reduces the probability of early rejection due to transient negative fluctuations. Without smoothing, a single large negative E_k can cause immediate rejection even if subsequent tokens would recover. With EMA, such spikes are averaged out.

Under Gaussian approximation, the cumulative energy at position k is approximately:

$$S_k \overset{\text{approx}}{\sim} \mathcal{N}(k\mu, k\sigma_{\text{eff}}^2), \tag{23}$$

where $\sigma_{\text{eff}} = \sigma$ for $\beta = 0$ and $\sigma_{\text{eff}} = \sigma\sqrt{(1-\beta)/(1+\beta)}$ for $\beta > 0$.

The probability of surviving to position k (i.e., $S_k \le \tau$) is:

$$\mathbb{P}(S_k \le \tau) = \mathbb{E}_\tau\left[\Phi\left(\frac{\tau - k\mu}{\sqrt{k}\sigma_{\text{eff}}}\right)\right], \tag{24}$$

where Φ is the standard normal CDF.

Since $\mu < 0$, we have $-k\mu = k|\mu| > 0$, so:

$$\frac{\tau - k\mu}{\sqrt{k}\sigma_{\text{eff}}} = \frac{\tau + k|\mu|}{\sqrt{k}\sigma_{\text{eff}}}. \tag{25}$$

For fixed k and τ, decreasing σ_{eff} (by increasing β) increases the argument of Φ, hence increases the survival probability.

Step 3: Expected Acceptance Length Calculation.

The expected acceptance length is:

$$\mathbb{E}[L] = \sum_{k=1}^{\gamma} \mathbb{P}(S_j \le \tau \text{ for all } j \le k) \approx \sum_{k=1}^{\gamma} \mathbb{P}(S_k \le \tau). \tag{26}$$

For the dominant contribution near $k \approx \mathbb{E}[L_0]$, we Taylor expand. Let $k^* = |\mu|^{-1}$ be the characteristic acceptance length. Near k^*:

$$\mathbb{P}(S_{k^*} \le \tau) \approx \Phi\left(\frac{\tau + 1}{\sigma_{\text{eff}}/\sqrt{|\mu|}}\right). \tag{27}$$

The survival probability increases when σ_{eff} decreases. The derivative is:

$$\frac{\partial}{\partial \sigma_{\text{eff}}}\Phi\left(\frac{\tau+1}{\sigma_{\text{eff}}/\sqrt{|\mu|}}\right) = -\phi(z)\cdot\frac{(\tau+1)\sqrt{|\mu|}}{\sigma_{\text{eff}}^2} < 0, \tag{28}$$

where ϕ is the standard normal PDF and z is the argument of Φ.

Step 4: Derivation of the Gain Formula.

The change in σ_{eff} from $\beta = 0$ to $\beta > 0$ is:

$$\Delta\sigma_{\text{eff}} = \sigma\left(\sqrt{\frac{1-\beta}{1+\beta}} - 1\right) \approx -\frac{\sigma\beta}{2} + O(\beta^2). \tag{29}$$

Table 17: Training cost comparison for 7B–8B parameter language model acceleration methods. All GPU hour estimates assume 203 TFLOPS sustained utilization on H100 GPUs. Methods are grouped by training paradigm: from scratch (FS), continual pre-training (CP), fine-tuning (FT), and auxiliary module training (AUX).

Method	Paradigm	Params	Tokens	Batch Size	Steps	TFLOPs	H100 Hours
Baseline (AR)	FS	7B	2.0T	–	–	8.4×10^{10}	115,000
LLaDA	FS	8B	2.3T	5.2M	442,308	1.1×10^{11}	151,000
Dream	CP	7B	600B	4.0M	150,000	2.5×10^{10}	34,500
TiDAR	CP	8B	150B	2.0M	75,000	7.2×10^{9}	9,900
SBD	FT	8B	70B	2.0M	35,000	3.4×10^{9}	4,600
EAGLE-3	AUX	0.24B†	600M	–	~800,000	$9.3 \times 10^{6\ddagger}$	~900§
EAGLE	AUX	0.24B†	70M	–	~100,000	$1.1 \times 10^{6\ddagger}$	~150§
Ours	AUX (LoRA)	80M†	286M	16K	17,500	$4.1 \times 10^{6\ddagger}$	**18**

†Base model (7–8B) is frozen; only the auxiliary module (draft head or LoRA adapter) is trained.
‡Includes frozen base model forward pass: $C = 2P_{\text{base}}D + 6P_{\text{aux}}D$.
§Empirical estimates; actual utilization lower than 203 TFLOPS due to memory bandwidth constraints.

The corresponding change in survival probability at position k is:

$$\Delta\mathbb{P}(S_k \le \tau) \approx \phi(z_k) \cdot \frac{(\tau + k|\mu|)\sqrt{|\mu|}}{\sigma^2} \cdot \frac{\sigma\beta}{2}, \tag{30}$$

where $z_k = (\tau + k|\mu|)/(\sqrt{k}\sigma)$.

Summing over positions and integrating over $\tau \sim \text{Exp}(1)$:

$$\mathbb{E}[L_\beta] - \mathbb{E}[L_0] \approx \sum_{k=1}^{\gamma} \int_0^\infty e^{-\tau} \cdot \phi(z_k) \cdot \frac{(\tau + k|\mu|)\sqrt{|\mu|}}{\sigma^2} \cdot \frac{\sigma\beta}{2}\, d\tau \tag{31}$$

$$= \frac{\beta\sqrt{|\mu|}}{2\sigma} \sum_{k=1}^{\gamma} \int_0^\infty e^{-\tau}\phi(z_k)(\tau + k|\mu|)\, d\tau. \tag{32}$$

The dominant contribution comes from $k \approx k^* = |\mu|^{-1}$. Evaluating the integral:

$$\int_0^\infty e^{-\tau}\phi\left(\Phi^{-1}(1 - e^{-\tau})\right)\, d\tau = c_0 > 0, \tag{33}$$

which is a universal positive constant.

The relative gain is therefore:

$$\frac{\mathbb{E}[L_\beta] - \mathbb{E}[L_0]}{\mathbb{E}[L_0]} \ge c \cdot \frac{\sigma}{|\mu|} \cdot \beta + O(\beta^2), \tag{34}$$

where $c > 0$ absorbs the constants from the integration. Rearranging:

$$\mathbb{E}[L_\beta] \ge \mathbb{E}[L_0] \cdot \left(1 + c \cdot \frac{\sigma}{|\mu|} \cdot \beta + O(\beta^2)\right). \tag{35}$$

This completes the proof of equation 3. □

E Training Cost Analysis

This section presents a comprehensive analysis of the computational requirements for training various language model acceleration methods. We establish a unified framework for comparing training costs across different paradigms, ranging from full pre-training to lightweight auxiliary module training.

E.1 METHODOLOGY

We estimate training costs using the standard approximation for transformer training FLOPs Kaplan et al. (2020); Hoffmann et al. (2022):

$$C = 6PD \tag{36}$$

where C denotes the total floating-point operations, P represents the number of trainable model parameters, and D is the total number of training tokens. The factor of 6 accounts for the forward pass (2 FLOPs per parameter per token) and the backward pass (4 FLOPs per parameter per token).

To convert FLOPs to GPU hours, we use:

$$H = \frac{C}{U \times 3600} \tag{37}$$

where H is the total GPU hours and U is the sustained utilization in FLOPS. We standardize our estimates using $U = 203 \times 10^{12}$ FLOPS (203 TFLOPS) per H100 GPU, derived from the Megatron-LM baseline under mixed-precision training conditions.

The number of training steps S is computed as:

$$S = \frac{D}{B} \tag{38}$$

where B denotes the global batch size in tokens.

E.2 METHOD DESCRIPTIONS AND CALCULATIONS

We analyze seven acceleration methods spanning four training paradigms: training from scratch, continual pre-training, fine-tuning, and auxiliary module training.

E.2.1 BASELINE: AUTOREGRESSIVE PRE-TRAINING

As a reference point, we consider standard autoregressive (AR) pre-training of a 7B parameter model on 2 trillion tokens, consistent with LLaMA-2 Touvron et al. (2023) and Megatron-LM Shoeybi et al. (2019) configurations.

$$C_{\text{baseline}} = 6 \times (7 \times 10^9) \times (2 \times 10^{12}) = 8.4 \times 10^{22} \text{ FLOPs} \tag{39}$$

$$H_{\text{baseline}} = \frac{8.4 \times 10^{22}}{203 \times 10^{12} \times 3600} = 114{,}890 \approx 115{,}000 \text{ H100 hours} \tag{40}$$

E.2.2 LLADA: LARGE LANGUAGE DIFFUSION WITH MASKING

LLaDA Nie et al. (2025a) is a masked diffusion language model trained from scratch without AR initialization. The 8B parameter model is trained on 2.3 trillion tokens with a global batch size of 5.2M tokens (batch size 1280 $\times$ sequence length 4096).

$$C_{\text{LLaDA}} = 6 \times (8 \times 10^9) \times (2.3 \times 10^{12}) = 1.104 \times 10^{23} \text{ FLOPs} \tag{41}$$

$$S_{\text{LLaDA}} = \frac{2.3 \times 10^{12}}{5.2 \times 10^6} = 442{,}308 \text{ steps} \tag{42}$$

$$H_{\text{LLaDA}} = \frac{1.104 \times 10^{23}}{203 \times 10^{12} \times 3600} = 151{,}067 \approx 151{,}000 \text{ H100 hours} \tag{43}$$

The original paper reports 0.13M H800 GPU hours, implying a higher effective utilization of approximately 235 TFLOPS.

E.2.3 DREAM

Dream Ye et al. (2025) initializes a 7B diffusion model from pre-trained AR weights (e.g., Qwen-2.5 or LLaMA-3), enabling convergence with significantly less data. The model undergoes continual pre-training on 600 billion tokens with an estimated global batch size of 4M tokens.

$$C_{\text{Dream}} = 6 \times (7 \times 10^9) \times (6 \times 10^{11}) = 2.52 \times 10^{22} \text{ FLOPs} \tag{44}$$

$$S_{\text{Dream}} = \frac{6 \times 10^{11}}{4 \times 10^6} = 150{,}000 \text{ steps} \tag{45}$$

$$H_{\text{Dream}} = \frac{2.52 \times 10^{22}}{203 \times 10^{12} \times 3600} = 34{,}484 \approx 34{,}500 \text{ H100 hours} \tag{46}$$

E.2.4 TIDAR: THINK IN DIFFUSION, TALK IN AUTOREGRESSION

TiDAR Liu et al. (2025) adapts a pre-trained 8B AR model (e.g., Qwen) into a hybrid architecture capable of both diffusion-based drafting and AR verification. Continual pre-training is performed on 150 billion tokens with a global batch size of 2M tokens.

$$C_{\text{TiDAR}} = 6 \times (8 \times 10^9) \times (1.5 \times 10^{11}) = 7.2 \times 10^{21} \text{ FLOPs} \tag{47}$$

$$S_{\text{TiDAR}} = \frac{1.5 \times 10^{11}}{2 \times 10^6} = 75{,}000 \text{ steps} \tag{48}$$

$$H_{\text{TiDAR}} = \frac{7.2 \times 10^{21}}{203 \times 10^{12} \times 3600} = 9{,}852 \approx 9{,}900 \text{ H100 hours} \tag{49}$$

E.2.5 SBD: SET BLOCK DECODING

SBD Gat et al. (2025) fine-tunes a pre-trained 8B next-token prediction model (e.g., LLaMA-3.1) to enable block decoding capabilities. Training uses 70 billion tokens from a mixture of reasoning and instruction data, with a global batch size of 2M tokens.

$$C_{\text{SBD}} = 6 \times (8 \times 10^9) \times (7 \times 10^{10}) = 3.36 \times 10^{21} \text{ FLOPs} \tag{50}$$

$$S_{\text{SBD}} = \frac{7 \times 10^{10}}{2 \times 10^6} = 35{,}000 \text{ steps} \tag{51}$$

$$H_{\text{SBD}} = \frac{3.36 \times 10^{21}}{203 \times 10^{12} \times 3600} = 4{,}598 \approx 4{,}600 \text{ H100 hours} \tag{52}$$

E.2.6 EAGLE AND EAGLE-3

EAGLE Li et al. (2024) and EAGLE-3 Li et al. (2025) employ a fundamentally different paradigm: the base model (7–8B parameters) remains frozen, and only a lightweight draft head ($\sim$0.24–0.99B parameters depending on base model size) is trained. For these methods, Equation 36 applies only to the trainable draft head; however, practical training time is dominated by the forward pass through the frozen base model for feature extraction, making pure FLOP calculations underestimates.

EAGLE. The original EAGLE model is trained on the ShareGPT dataset comprising 68,000 dialogue samples, corresponding to approximately 70M tokens. Training employs the AdamW optimizer with learning rate 3×10^{-5}, $(\beta_1, \beta_2) = (0.9, 0.95)$, and gradient clipping at 0.5. The trainable draft head parameters for the 7B base model total 0.24B. Training completes in 1–2 days on 4$\times$ A100 (40GB) GPUs, or equivalently on a single node of 8$\times$ RTX 3090 GPUs.

$$D_{\text{EAGLE}} = 70 \times 10^6 \text{ tokens} \tag{53}$$

$$H_{\text{EAGLE}} \approx 100\text{–}150 \text{ H100 hours (empirical)} \tag{54}$$

EAGLE-3. EAGLE-3 scales up training data by combining ShareGPT (68K samples) with UltraChat-200K (464K samples), yielding approximately 532K training samples ($\sim$600M tokens). The model is trained for 10 epochs over this combined dataset, totaling approximately 800,000 training steps. This represents roughly 8$\times$ more training data than EAGLE, contributing to EAGLE-3's approximately 1.4$\times$ latency improvement over EAGLE-2 at batch size 1.

$$D_{\text{EAGLE-3}} = 600 \times 10^6 \text{ tokens} \tag{55}$$

$$S_{\text{EAGLE-3}} \approx 800{,}000 \text{ steps} \tag{56}$$

$$H_{\text{EAGLE-3}} \approx 800\text{–}1{,}000 \text{ H100 hours} \tag{57}$$

E.2.7 Cross-Regression Adapter

Our method employs a parameter-efficient LoRA adapter to enable multi-token prediction while keeping the base model entirely frozen. The adapter uses rank 32 and targets all linear layers, yielding approximately 80M trainable parameters for a 7B base model—well under 1% of the total parameter count.

Dataset and Training Configuration. Training uses a compact 7K-sample dataset (either GSM8K for arithmetic reasoning or a balanced mixture of UltraChat, ShareGPT, and BaGel for multi-domain coverage), with sequences clipped to 2048 tokens. Self-distillation targets are generated by the frozen base model producing 16-token continuations for each prefix. Training proceeds for 20 epochs with batch size 8 per H100 GPU.

Trainable Parameters. For a 7B LLaMA-style architecture with 32 transformer layers, hidden dimension 4096, and MLP intermediate dimension 11008, LoRA with rank 32 targeting all linear layers contributes:

$$P_{\text{attn/layer}} = 4 \times 2 \times d_{\text{model}} \times r = 4 \times 2 \times 4096 \times 32 = 1.05 \times 10^6 \tag{58}$$

$$P_{\text{MLP/layer}} = 3 \times 2 \times (d_{\text{model}} + d_{\text{ff}}) \times r/2 \approx 1.45 \times 10^6 \tag{59}$$

$$P_{\text{LoRA}} = 32 \times (1.05 + 1.45) \times 10^6 \approx 80 \times 10^6 \text{ parameters} \tag{60}$$

Training Tokens and Steps. With 7,000 examples at maximum sequence length 2048 tokens:

$$D_{\text{epoch}} = 7{,}000 \times 2{,}048 = 14.3 \times 10^6 \text{ tokens} \tag{61}$$

$$D_{\text{total}} = 20 \times 14.3 \times 10^6 = 286 \times 10^6 \text{ tokens} \tag{62}$$

$$S_{\text{total}} = \frac{7{,}000}{8} \times 20 = 17{,}500 \text{ steps} \tag{63}$$

Computational Cost. Training comprises two phases: (1) offline target generation via the frozen base model, and (2) LoRA adapter optimisation. The computational cost is dominated by the forward pass through the frozen 7B model during each training step, rather than the lightweight LoRA gradient computation. As with EAGLE-family methods, FLOP-based estimates significantly underestimate wall-clock time due to memory bandwidth limitations during inference through the frozen model.

E.3 Summary

Table 17 presents the complete comparison of training costs across all methods. The relative cost is computed with respect to LLaDA as the most expensive method representing full diffusion model training.