

# Dual-Stream Decoding for Accelerated Large Language Models

Dmitry Abulkhanov<sup>1,\*</sup> Daniil Strizhakov<sup>2,\*</sup> Maxim Panov<sup>1</sup>

<sup>1</sup>Mohamed Bin Zayed University of AI, Abu Dhabi, UAE

<sup>2</sup>NITU MISiS

\*Equal contribution Correspondence: dmitry.abulkhanov@gmail.com

## Abstract

Autoregressive language models are trained to generate text one token at a time, causing inference latency and cost to scale linearly with output length. However, modern large language models often exhibit semi-autoregressive predictive capabilities, frequently aided by speculative decoding or other multi-token prediction methods. In contrast, discrete diffusion models promise parallel text generation but fundamentally struggle to model sequential correlations due to their reliance on mean-field approximations, effectively ignoring the causality inherent in natural language.

We introduce **Cross-Regression**, an approach aimed at achieving true hybridization of autoregressive and discrete diffusion sequence modeling. Cross-Regression is a decoding algorithm that exploits latent information about future tokens embedded in pretrained transformer hidden states via a dual-stream construction: a *control stream* that preserves exact autoregressive probabilities, and a *predictive stream* that proposes parallel candidates. Cross-Regression uses residual energy guidance to refine predictions without discarding intermediate computation, while enabling early acceptance analogous to speculative decoding. The method can operate in a lossless accelerated mode that preserves the original model distribution, or in a faster lossy mode with controllable deviation. Experiments demonstrate substantial speedups in text generation, reducing the number of sequential forward passes by up to  $5\times$  on models with up to 70B parameters, while remaining compatible with KV caching and without requiring prohibitively expensive training.

## 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. \(2023\)](#); [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 promise to break this sequential bottleneck through parallel token generation ([Austin et al., 2021](#); [Hooeboom et al., 2021](#)). Rather than generating left-to-right, MDMs frame generation as iterative denoising: starting from a fully masked sequence, the model progressively unmask multiple tokens per step until a complete sequence emerges. Recent large-scale MDMs including LLaDA ([Nie et al., 2025](#)), Dream ([Wu et al., 2026](#)), and Mercury ([Khanna 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:  $p_{\hat{\theta}}(\mathbf{x}_0 | \mathbf{x}^t) \approx \prod_{i=1}^L p_{\hat{\theta}}(x_{0,i} | \mathbf{x}^t)$ . This assumes tokens can be predicted independently within each step, ignoring the sequential dependencies essential for coherent

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

Table 1: Essential differences of non-autoregressive / parallel decoding frameworks.

text. At speedups comparable to speculative decoding ( $3\text{--}5\times$ ), diffusion approaches retain only 37–62% of baseline task performance, making 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. 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<sup>1</sup>. This prohibitive cost limits accessibility and prevents 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. (2023); 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., 2023; 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, but it wastes computation when verification fails: all subsequent draft tokens are discarded, with no 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 1 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 budget 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 empirically show scaling of this method to the model sizes up to 70B parameters.

<sup>1</sup>See Table 3 in the appendix for a detailed training cost comparison.

**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 4.1), and a lossy mode ( $\beta > 0$ ) using EMA-smoothed energy to offset transient errors for guaranteed acceptance gains (Theorem 4.3).

**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 (Shih et al., 2022) 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.

## 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  $q_{t|0}(\mathbf{x}^t | \mathbf{x}_0)$  operates independently per position according to

$$q_{t|0}(x_i^t | x_{0,i}) = (1 - \alpha_t) \cdot \mathbf{1}_{x_i^t = [\text{MASK}]} + \alpha_t \cdot \mathbf{1}_{x_i^t = x_{0,i}},$$

where  $\alpha_t : [0, 1] \rightarrow [0, 1]$  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  $\alpha_t = 1 - t$ , the cosine schedule  $\alpha_t = \cos^2(\pi t/2)$ , and the log-linear schedule  $\alpha_t = \exp(-\kappa t)$  for decay rate  $\kappa > 0$ .

The joint forward distribution for the entire sequence factorizes as  $q_{t|0}(\mathbf{x}^t | \mathbf{x}_0) = \prod_{i=1}^L q_{t|0}(x_i^t | x_{0,i})$ , which follows from the independence assumption in the masking noise. Denoting by  $M_t = \{i : x_i^t = [\text{MASK}]\}$  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

$$\mathcal{L}(\hat{\theta}) = \int_0^1 -\frac{\alpha'_t}{1 - \alpha_t} \mathbb{E}_{q(\mathbf{x}_0, \mathbf{x}^t)} \left[ \sum_{i: x_i^t = [\text{MASK}]} -\log p_{\hat{\theta}}(x_{0,i} | \mathbf{x}^t) \right] dt.$$

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  $\mathbf{x}^t \sim q_{t|0}(\cdot | \mathbf{x}_0)$ . 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  $q_{s|t}(x_i^s | x_i^t, x_{0,i}) = \frac{q_{t|s}(x_i^t | x_i^s) \cdot q_{s|0}(x_i^s | x_{0,i})}{q_{t|0}(x_i^t | x_{0,i})}$ . Due to the coordinate-wise independence of the forward process, the joint reverse posterior factorizes exactly as  $q_{s|t}(\mathbf{x}^s | \mathbf{x}^t, \mathbf{x}_0) = \prod_{i=1}^L q_{s|t}(x_i^s | x_i^t, x_{0,i})$ .

However, during inference  $\mathbf{x}_0$  is unknown; it is precisely what we wish to sample. The model must therefore learn an approximation  $p_{\hat{\theta}}(\mathbf{x}_0 | \mathbf{x}^t)$ . Standard masked diffusion

**Algorithm 1** Energy-Based Acceptance

---

**Require:** Cached probabilities  $\{(p_k, q_k)\}_{k=1}^\gamma$ , EMA coefficient  $\beta$ , threshold  $\lambda \leq 0$ , horizon  $\gamma$   
**Ensure:** Number of accepted tokens  $n$

```

1:  $(\rho, n) \leftarrow (0, 0)$ 
2: for  $k = 1$  to  $\gamma$  do
3:    $E_k \leftarrow \log q_k - \log p_k$  ▷ Energy at position  $k$ 
4:    $\rho \leftarrow \beta\rho + (1 - \beta)E_k$ 
5:    $\hat{\rho}_k \leftarrow \rho / (1 - \beta^k)$  ▷ Bias-corrected EMA
6:   Sample  $u_k \sim \text{LogUniform}(\lambda, 0)$ 
7:    $\eta_k \leftarrow \max(\lambda \cdot \sqrt{k/\gamma}, u_k)$  ▷ Acceptance bound
8:   if  $\hat{\rho}_k > \eta_k$  then return  $n$ 
9:   end if
10:   $n \leftarrow k$ 
11: end for
12: return  $n$  ▷ Full acceptance

```

---

architectures using bidirectional Transformers parameterize this as

$$p_{\hat{\theta}}(\mathbf{x}_0 \mid \mathbf{x}^t) \approx \prod_{i=1}^L p_{\hat{\theta}}(x_{0,i} \mid \mathbf{x}^t),$$

where each  $p_{\hat{\theta}}(x_{0,i} \mid \mathbf{x}^t)$  is a softmax over the vocabulary at position  $i$ , computed via a single forward pass through the bidirectional encoder. This constitutes 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  $E : \mathcal{V}^L \rightarrow \mathbb{R}$  via the Boltzmann distribution  $p_{\text{EBM}}(\mathbf{x}) = \frac{1}{\mathcal{Z}} \exp(-E(\mathbf{x}))$ , 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.

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

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

Taking logarithms yields the cumulative energy correction

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

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

This factorization reveals that the mean-field gap can be corrected through multiplicative importance weights, which in log-space become additive energy corrections. The challenge is that directly computing and applying these weights requires evaluating the full autoregressive model, negating the computational benefits of the parallel mean-field proposal. Cross-Regression resolves this tension through incremental correction within a Markov chain framework, accepting partial sequences when cumulative corrections remain small and rejecting when they become large, thereby amortizing the cost of causal evaluation over multiple accepted tokens.

#### 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  $\mathbf{x} = (x_1, \dots, x_n) \in \mathcal{V}^n$  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  $\hat{\mathbf{x}} = (\hat{x}_1, \dots, \hat{x}_\gamma) \in \mathcal{V}^\gamma$  contain  $\gamma$  candidate tokens generated in parallel by the predictive stream. The acute notation distinguishes these unverified proposals from accepted tokens: each  $\hat{x}_j \sim Q(\cdot | \mathbf{x}, \tilde{\mathbf{x}})$  awaits sequential verification against the target distribution  $p_{\text{AR}}$ , the predictive stream state  $\tilde{\mathbf{x}} = (\tilde{x}_1, \dots, \tilde{x}_m)$  stores previously proposed but unaccepted tokens after the current round of acceptance. Some prefix of  $\hat{\mathbf{x}}$  will be accepted and appended to  $\mathbf{x}$ , while the remaining tokens, 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

$$\pi(\mathbf{x}, \hat{\mathbf{x}} | \mathbf{x}_0, \tilde{\mathbf{x}}) \propto p_{\text{AR}}(\mathbf{x} | \mathbf{x}_0) \cdot Q(\hat{\mathbf{x}} | \mathbf{x}, \tilde{\mathbf{x}}) \cdot \exp(-E_{\text{couple}}(\mathbf{x}, \hat{\mathbf{x}})), \quad (1)$$

where the coupling energy enforces consistency between accepted and proposed tokens. The complete decoding procedure is detailed in Algorithm 2 in Appendix A.1.

**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:

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

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

**EMA smoothing for variance reduction** This energy is accumulated through exponential moving average smoothing:  $\rho_k = \beta \rho_{k-1} + (1 - \beta) E_k$ ,  $\hat{\rho}_k = \frac{\rho_k}{1 - \beta^k}$ , where  $\hat{\rho}_k$  is the bias-corrected estimate. The coupling mechanism accepts position  $k$  if and only if the smoothed

energy stays below an adaptive threshold:  $\hat{\rho}_k \leq \eta_k$ , where  $\eta_k = \max(\lambda\sqrt{k/\gamma}, u_k)$  for base threshold  $\lambda \leq 0$ , horizon  $\gamma$ , and stochastic perturbation  $u_k \sim \text{LogUniform}(\lambda, 0)$ .

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

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

where  $n = \max\{k : \hat{\rho}_j \leq \eta_j \text{ for all } j \leq k\}$  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  $\beta$  and the base threshold  $\lambda$ . The interplay between  $\beta$  and  $\lambda$  defines a continuous family of operating points along the Pareto frontier of speed versus quality. At one extreme ( $\beta = 0, \lambda = 0$ ), the method is provably lossless with acceptance behavior similar to speculative decoding with reiteration of not accepted tokens. At the other extreme ( $\beta \rightarrow 1, \lambda \rightarrow -\infty$ ), 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  $p_{\text{AR}}$ . In this configuration, the smoothed energy reduces to the instantaneous energy at each position:

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

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

When  $\lambda = 0$ , this further simplifies to  $P_k \geq Q_k \cdot e^{-u_k}$  with  $u_k \leq 0$ , which is equivalent to the standard speculative decoding acceptance rule (Leviathan et al., 2023). 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  $p_{\text{AR}}$  exactly. Theorem 4.1 formalizes this guarantee: tokens sampled via Cross-Regression with  $\beta = 0$  are distributed identically to tokens sampled directly from the target model.

The lossless regime is appropriate when distributional exactness is paramount: it allows to apply this acceleration method to any pretrained/fine-tuned LLM without altering its behaviour with strict adherence to a original RLHF alignment. The trade-off is that acceptance lengths are determined solely by the instantaneous alignment between  $P$  and  $Q$ , leading to higher variance and potentially shorter acceptance runs when the predictive stream makes occasional errors.

**Theorem 4.1** (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 B.1.

**Lossy regime ( $\beta > 0$  or  $\lambda < 0$ )** 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 positive energies (where the predictive stream  $Q$  locally overestimates the target  $P$ ) to be offset by subsequent negative energies (where  $Q$  underestimates). Concretely, a token  $\hat{x}_k$  with  $E_k > 0$  may still be accepted if the smoothed estimate  $\hat{\rho}_k$  remains below threshold due to favorable energies at earlier positions.

*Threshold relaxation ( $\lambda < 0$ ):* Setting the base threshold  $\lambda < 0$  introduces explicit tolerance for cumulative positive energy. The adaptive threshold  $\eta_k = \max(\lambda\sqrt{k/\gamma}, u_k)$  allows

| Base       | Framework        | Method                       | GSM8K       |              |             | HumanEval   |               |             | Mean         |             |
|------------|------------------|------------------------------|-------------|--------------|-------------|-------------|---------------|-------------|--------------|-------------|
|            |                  |                              | Acc@1       | Rel. Acc     | Speedup     | Pass@1      | Rel. Pass     | Speedup     | Quality      | Speedup     |
| LLaMA3 8B  | Diffusion        | LLaDA (T=1024)               | <u>0.83</u> | 100.0%       | 0.33        | <u>0.19</u> | 100.0%        | 0.52        | 100.0%       | 0.43        |
|            |                  | LLaDA (T=256)                | <u>0.77</u> | 92.5%        | 1.34        | 0.06        | 32.3%         | 2.30        | 62.4%        | 1.82        |
|            |                  | LLaDA (T=128)                | <u>0.46</u> | 55.8%        | 4.18        | <u>0.04</u> | 19.4%         | 4.80        | 37.6%        | 4.49        |
|            | AutoReg.         | Baseline                     | <u>0.67</u> | 100.0%       | 1.00        | <u>0.40</u> | 100.0%        | 1.00        | 100.0%       | 1.00        |
|            | Spec.            | 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        |
|            | Decoding         | 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 ( $\epsilon \leq 14$ ) | 0.66        | <b>98.0%</b> | <b>5.03</b> | 0.40        | <b>101.5%</b> | <b>3.59</b> | <b>99.8%</b> | <b>4.31</b> |
| Qwen2.5 7B | Diffusion        | Dream (T=1024)               | <u>0.48</u> | 100.0%       | 0.09        | <u>0.30</u> | 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                     | <u>0.84</u> | 100.0%       | 1.00        | <u>0.22</u> | 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 ( $\epsilon \leq 8$ )  | 0.74        | 88.6%        | 5.46        | 0.29        | 132.5%        | 3.13        | 110.6%       | 4.29        |
|            |                  | Lossy ( $\epsilon \leq 12$ ) | 0.69        | <b>83.0%</b> | <b>5.82</b> | 0.23        | <b>104.9%</b> | <b>3.23</b> | <b>94.0%</b> | <b>4.52</b> |

Table 2: **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.<sup>2</sup>

sequences where the predictive stream systematically overestimates certain tokens to still be accepted, provided the discrepancy remains bounded.

**Assumption 4.2** (Stationary energy). The energy sequence  $\{E_k\}$  satisfies  $\mathbb{E}[E_k] = \mu > 0$  and  $\text{Var}(E_k) = \sigma^2 < \infty$  with uncorrelated increments.

**Theorem 4.3** (Acceptance length gain of lossy regime). *Under Assumption 4.2, let  $\tau_0$  denote the expected acceptance length with  $\beta = 0$ , and  $\tau_\beta$  with EMA parameter  $\beta \in (0, 1)$ . Then*

$$\mathbb{E}[\tau_\beta] \geq \mathbb{E}[\tau_0] \cdot \left(1 + c \cdot \frac{\sigma}{|\mu|} \cdot \beta + O(\beta^2)\right) \quad (2)$$

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 B.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 acceptance, the smoothed cumulative energy  $\hat{\rho}_k$  provides a running assessment of proposal quality. Unlike hard rejection at the first misaligned token, the EMA accumulation allows transient positive energies (where  $Q$  locally overestimates) to be offset by subsequent negative energies (where  $Q$  underestimates), rejecting only when systematic divergence accumulates. This corresponds to Lines 3–8 of Algorithm 1.

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

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

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 2) 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  $\hat{x}_{n+2:\gamma}$  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. 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 objective that encourages the predictive stream to assign high probability to the target continuation tokens generated by the frozen base model. Let  $\hat{y}_{i+t}$  denote the target token at step  $t$ , and let  $q_t(\cdot)$  be the predictive-stream distribution at that step. With  $s_t$  as a weight derived from the matching suffix length, the loss is:

$$\mathcal{L} = - \frac{\sum_{t=1}^w (s_t + 1) \cdot \log q_t(\hat{y}_{i+t})}{\sum_{t=1}^w (s_t + 1)}$$

The complete training procedure is detailed in Algorithm 3 in Appendix D.

Table 2 demonstrates that Cross-Regression establishes the speed-quality Pareto frontier.

### 5.1 Scaling laws and task-dependent acceleration

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.

Table 2 demonstrate that Cross-Regression establishes the speed-quality Pareto frontier.

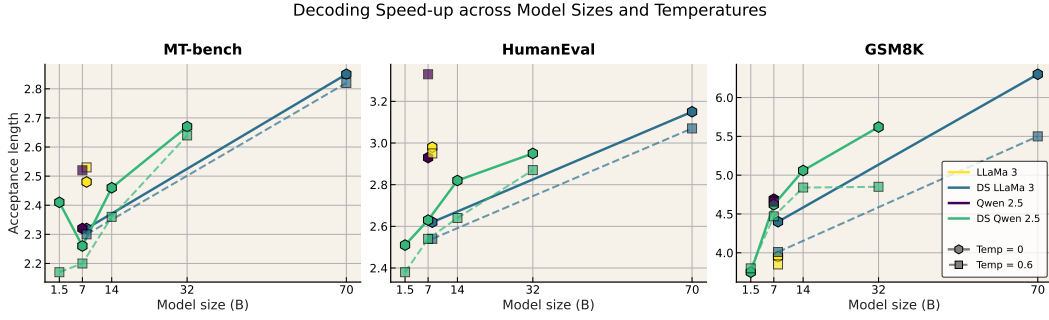


Figure 1: Scaling law of speed-up as a function of model parameter count. Results are aggregated over GSM8K, HumanEval, and MT-bench datasets with temperatures  $T \in \{0, 0.6\}$ .

**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.

**Different domains and base models** Figure 1 reveals that the mean acceptance length ( $\tau$ ) is principally driven by model scale, as exemplified by the DS Qwen 2.5 series where  $\tau$  consistently increases with parameter count, from 2.89 (1.5B) to 3.75 (32B) at  $T=0$ . This scaling benefit is most pronounced on structured tasks like GSM8K, which consistently yield the highest acceptance lengths (e.g., 5.62), surpassing more open-ended benchmarks. Conversely, performance is moderated by sampling temperature, with the introduction of stochasticity ( $T=0.6$ ) generally causing a slight reduction in  $\tau$  across most models due to increased predictive difficulty. Finally, we observe that downstream fine-tuning does not guarantee improved acceleration; the base LLaMa 3 and Qwen 2.5 models achieve comparable or superior acceptance lengths to their DeepSeek (DS) tuned counterparts, indicating that optimization for task performance is distinct from achieving the predictive alignment necessary for efficient acceleration.

## 6 Conclusion

We introduced Cross-Regression, a decoding framework that unifies autoregressive and discrete diffusion language modeling through a dual-stream architecture. By combining a control stream that preserves exact autoregressive probabilities with a predictive stream that proposes parallel candidates, our method achieves  $3\text{--}6\times$  speedups across models up to 70B parameters while retaining over 99% output quality. Cross-Regression enables practitioners to accelerate arbitrary fine-tuned models without prohibitive retraining costs. Future work includes adaptive horizon scheduling and extension to multimodal generation.

## Reproducibility statement

We provide complete algorithmic details in Algorithms 2 and 1, with full proofs of theoretical results in Appendices B.1 and B.2. Training hyperparameters, model configurations, and evaluation protocols are detailed in Appendix D. 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 of 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, 2025. URL <http://arxiv.org/abs/2503.09573>.
- 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 *International Conference on Machine Learning, ICML’24*. JMLR.org, 2024. doi: 10.48550/arXiv.2401.10774.
- 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, B. Kailkhura, and Ferdinando Fioretto. Speculative Diffusion Decoding: Accelerating Language Generation through Diffusion, 2025. URL <http://arxiv.org/abs/2408.05636>.
- K. Cobbe, Vineet Kosaraju, Mo 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.org*, 2021.
- Nima Fathi, Torsten Scholak, and Pierre-André Noël. Unifying Autoregressive and Diffusion-Based Sequence Generation. *arXiv.org*, 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. *International Conference on Machine Learning*, 2024. doi: 10.48550/arXiv.2402.02057.
- 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.org*, 2025. doi: 10.48550/arXiv.2509.04185.
- Fabian Gloeckle, Badr Youbi Idrissi, Baptiste Rozière, David Lopez-Paz, and Gabriel Synnaeve. Better & faster large language models via multi-token prediction. In *International Conference on Machine Learning, ICML’24*. JMLR.org, 2024. doi: 10.48550/arXiv.2404.19737.

- Gabe Guo and Stefano Ermon. Reviving Any-Subset Autoregressive Models with Principled Parallel Sampling and Speculative Decoding, 2025. URL <http://arxiv.org/abs/2504.20456>.
- 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. *Advances in Neural Information Processing Systems 35*, pp. 30016–30030, 2022. doi: 10.52202/068431-2176.
- 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.org*, 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.org*, 2020.
- Samar Khanna, Siddhant Kharbanda, Shufan Li, Harshit Varma, Eric Wang, Sawyer Birnbaum, Ziyang Luo, Yanis Miraoui, Akash Palrecha, S. Ermon, et al. Mercury: Ultra-fast language models based on diffusion. *arXiv.org*, 2025. doi: 10.48550/arXiv.2506.17298.
- 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. doi: 10.48550/arXiv.2403.00835.
- Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pp. 19274–19286. PMLR, 2023. doi: 10.48550/arXiv.2211.17192.
- David Li, Nikita Gushchin, Dmitry Abulkhanov, Eric Moulines, Ivan Oseledets, Maxim Panov, and Alexander Korotin. IDLM: Inverse-distilled diffusion language models. In *Forty-third International Conference on Machine Learning*, 2026. URL <https://arxiv.org/abs/2602.19066>.
- Yifan Li, Kun Zhou, Wayne Xin Zhao, and Ji-Rong Wen. Diffusion Models for Non-autoregressive Text Generation: A Survey, 2023. URL <http://arxiv.org/abs/2303.06574>.
- Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE: Speculative Sampling Requires Rethinking Feature Uncertainty, 2024. URL <http://arxiv.org/abs/2401.15077>.
- Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle-3: Scaling up inference acceleration of large language models via training-time test. *Advances in Neural Information Processing Systems 38*, pp. 151568–151587, 2025. doi: 10.52202/085713-4562.
- 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.org*, 2025. doi: 10.48550/arXiv.2511.08923.
- Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models. *Advances in Neural Information Processing Systems 38*, pp. 56354–56392, 2025. doi: 10.52202/085713-1689.
- Arnaud Pannatier, Evann Courdier, and François Fleuret.  $\{\sigma\}$ -GPTs: A New Approach to Autoregressive Models, 2024. URL <http://arxiv.org/abs/2404.09562>.

- Gregory Polyakov, Ilseyar Alimova, Dmitry Abulkhanov, Ivan Sedykh, Andrey Bout, Sergey Nikolenko, and Irina Piontkovskaya. ToolReflection: Improving large language models for real-world API calls with self-generated data. In Ehsan Kamalloo, Nicolas Gontier, Xing Han Lu, Nouha Dziri, Shikhar Murty, and Alexandre Lacoste (eds.), *Proceedings of the 1st Workshop for Research on Agent Language Models (REALM 2025)*, pp. 184–199, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-264-0. doi: 10.18653/v1/2025.realm-1.14. URL <https://aclanthology.org/2025.realm-1.14/>.
- Max Ryabinin, Tim Dettmers, Michael Diskin, and Alexander Borzunov. SWARM parallelism: Training large models can be surprisingly communication-efficient. 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. 29416–29440. PMLR, 23–29 Jul 2023. URL <https://proceedings.mlr.press/v202/ryabinin23a.html>.
- 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, 2025. URL <http://arxiv.org/abs/2506.01928>.
- Israfel Salazar, Manuel Fernández Burda, Shayekh Islam, Arshia Soltani Moakhar, Shivalika Singh, Fabian Farestam, Angelika Romanou, Danylo Boiko, Dipika Khullar, Mike Zhang, Dominik Krzemiński, Jekaterina Novikova, Luisa Shimabucoro, Joseph Marvin Imperial, Rishabh Maheshwary, Sharad Duwal, Alfonso Amayuelas, Swati Rajwal, Jebish Purbey, Ahmed Ruby, Nicholas Popovič, Marek Suppa, Azmine Tushik Wasi, Ram Mohan Rao Kadiyala, Olga Tsymboi, Maksim Kostritsya, Bardia moakhar, Gabriel da Costa Merlin, Otávio Coletti, Maral Jabbarishiviari, MOHAMMADAMIN FARAHANIFARD, Silvia Fernandez, María Grandury, Dmitry Abulkhanov, Drishti Sharma, Andre Guarnier De Mitri, Leticia Marchezi, Setayesh Heydari, Johan S Obando Ceron, Nazar Kohut, Beyza Ermis, Desmond Elliott, Enzo Ferrante, Sara Hooker, and Marzieh Fadaee. Kaleidoscope: In-language exams for massively multilingual vision evaluation. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 81112–81164, 2026. URL [https://proceedings.iclr.cc/paper\\_files/paper/2026/file/8365049fc3fe46cb5ebd648d58ba0933-Paper-Conference.pdf](https://proceedings.iclr.cc/paper_files/paper/2026/file/8365049fc3fe46cb5ebd648d58ba0933-Paper-Conference.pdf).
- Apoorv Saxena. Prompt lookup decoding, November 2023. URL <https://github.com/apoorvumang/prompt-lookup-decoding/>.
- Andy Shih, Dorsa Sadigh, and Stefano Ermon. Training and Inference on Any-Order Autoregressive Models the Right Way, October 2022. URL <http://arxiv.org/abs/2205.13554>. arXiv:2205.13554 [cs.LG].
- 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.org*, 2019.
- Nikita Sorokin, Tikhonov Anton, Dmitry Abulkhanov, Ivan Sedykh, Irina Piontkovskaya, and Valentin Malykh. CCT-code: Cross-consistency training for multilingual clone detection and code search. In Abteen Ebrahimi, Samar Haider, Emmy Liu, Sammar Haider, Maria Leonor Pacheco, and Shira Wein (eds.), *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 4: Student Research Workshop)*, pp. 178–185, Albuquerque, USA, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-192-6. doi: 10.18653/v1/2025.naacl-srw.17. URL <https://aclanthology.org/2025.naacl-srw.17/>.
- Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. Blockwise parallel decoding for deep autoregressive models. *Neural Information Processing Systems*, abs/1811.03115, 2018. URL <http://arxiv.org/abs/1811.03115>.
- Adam Suma and Sam Dauncey. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv.org*, 2025. doi: 10.48550/arXiv.2501.12948.

- Hugo Touvron, Louis Martin, Kevin R. Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bash-lykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv.org*, 2023.
- Zirui Wu, Lin Zheng, Zhihui Xie, Jiacheng Ye, Jiahui Gao, Shansan Gong, Yansong Feng, Zhenguo Li, Wei Bi, Guorui Zhou, et al. Dreamon: Diffusion language models for code infilling beyond fixed-size canvas. <https://hku-nlp.github.io/blog/2025/dreamon>, 2026.
- 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, 2024. URL <http://arxiv.org/abs/2401.07851>.
- 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*, 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.org*, 2025. doi: 10.48550/arXiv.2508.15487.

## A Appendix

### A.1 The Full Decoding Algorithm

The Cross-Regression decoding algorithm is described in Algorithm 2. It uses a dual-stream architecture to jointly compute proposal and verification scores.

---

**Algorithm 2** Cross-Regression Decoding

---

*Notation:*  $P_k \equiv P(\cdot \mid \mathbf{x}, \hat{\mathbf{x}}_{1:k-1})$  and  $Q_k \equiv Q(\cdot \mid \mathbf{x}, \hat{\mathbf{x}}_{1:k-1})$

**Require:** Dual-stream model  $\mathcal{M}$  (both  $P$  and  $Q$ ), prompt  $\mathbf{x}_0$ , horizon  $\gamma$ , EMA  $\beta$ , threshold  $\lambda$

**Ensure:** Verified-draft pair  $(\mathbf{x}, \hat{\mathbf{x}})$

- 1:  $(\mathbf{x}, \hat{\mathbf{x}}, m) \leftarrow (\mathbf{x}_0, \text{RandomTokens}(\gamma), 0)$
- 2: **while** not terminated **do**
  - ▷ *Parallel dual-stream forward pass*
  - 3:  $\{(P_k, Q_k)\}_{k=1}^\gamma \leftarrow \mathcal{M}(\mathbf{x}, \hat{\mathbf{x}})$  ▷ Shared forward
  - ▷ *Fill beyond valid prefix and evaluate*
  - 4: Sample  $\hat{x}_k \sim Q_k$  for  $k = m + 1, \dots, \gamma$
  - ▷ *Cache  $\{(p_k, q_k)\}_{k=1}^\gamma$*
  - 5:  $(p_k, q_k) \leftarrow (P_k(\hat{x}_k), Q_k(\hat{x}_k))$
  - 6:  $n \leftarrow \text{ENERGYACCEPT}(\{(p_k, q_k)\}, \beta, \lambda)$  ▷ Alg. 1
  - 7:  $\mathbf{x} \leftarrow \mathbf{x} \parallel \hat{\mathbf{x}}_{1:n}$  ▷ Accept verified tokens
  - 8: **if**  $n = \gamma$  **then**
  - 9:  $(\hat{\mathbf{x}}, m) \leftarrow (\text{RandomTokens}(\gamma), 0)$
  - 10: **else**
  - 11:  $(P, Q) \leftarrow \mathcal{M}(\mathbf{x}, 1)$  ▷ Correction sampling
  - 12: Sample  $x_{\text{corr}} \sim p_{\text{res}}$  where  $p_{\text{res}}(v) \propto \max(0, P(v) - Q(v))$
  - 13:  $\mathbf{x} \leftarrow \mathbf{x} \parallel x_{\text{corr}}$
  - ▷ *Shift draft buffer*
  - 14:  $(\hat{\mathbf{x}}_{1:\gamma-n-1}, m) \leftarrow (\hat{\mathbf{x}}_{n+2:\gamma}, \gamma - n - 1)$
  - 15: **end if**
  - 16: **end while**
  - 17: **return**  $(\mathbf{x}, \hat{\mathbf{x}})$

---

## B Proofs of Theoretical Results

Let  $P(x)$  denote the target autoregressive distribution (control stream) and  $Q(x)$  the proposal distribution (predictive stream). 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 B.1** (Energy). *For a proposed token  $\hat{x}$ , the energy is*

$$E = \log Q(\hat{x}) - \log P(\hat{x}).$$

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

$$p_{\text{res}}(v) := \frac{\max(0, P(v) - Q(v))}{\sum_{v'} \max(0, P(v') - Q(v'))}. \quad (3)$$

### B.1 Proof of Theorem 4.1

*Proof.* We proceed in three steps: (1) show that  $\beta = 0$  reduces cross-regressive decoding to rejection sampling with residual correction, (2) verify single-token correctness, and (3) extend to full sequences by induction.

**Step 1: Reduction when  $\beta = 0$ .**

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

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

Hence acceptance is determined by instantaneous token-level energy, i.e., the usual token-wise acceptance ratio between target  $P$  and proposal  $Q$ .

**Step 2: Single-token correctness.**

For one position, draw  $\hat{x} \sim Q$  and accept with probability

$$a(\hat{x}) = \min\left(1, \frac{P(\hat{x})}{Q(\hat{x})}\right).$$

Then

$$\Pr(\text{accept}) = \sum_{v \in \mathcal{V}} Q(v) \min\left(1, \frac{P(v)}{Q(v)}\right) = \sum_{v \in \mathcal{V}} \min(P(v), Q(v)) := Z. \quad (4)$$

Conditioned on acceptance, the accepted token has distribution

$$\Pr(x = v \mid \text{accept}) = \frac{Q(v) \min(1, P(v)/Q(v))}{Z} = \frac{\min(P(v), Q(v))}{Z}.$$

On rejection (probability  $1 - Z$ ), correction sampling uses

$$p_{\text{res}}(v) = \frac{(P(v) - Q(v))_+}{1 - Z},$$

since  $\sum_v (P(v) - Q(v))_+ = 1 - Z$ .

Therefore the marginal output distribution is

$$\Pr(x = v) = Z \cdot \frac{\min(P(v), Q(v))}{Z} + (1 - Z) \cdot \frac{(P(v) - Q(v))_+}{1 - Z} \quad (5)$$

$$= \min(P(v), Q(v)) + (P(v) - Q(v))_+ \quad (6)$$

$$= P(v), \quad (7)$$

using  $\min(a, b) + (a - b)_+ = a$  for  $a, b \geq 0$ .

**Step 3: Extension to full sequences.**

We prove by induction on sequence length that the generated sequence follows the target autoregressive law  $p_{\text{AR}}$ .

*Base case.* Step 2 gives  $x_1 \sim p_{\text{AR}}(\cdot \mid \mathbf{x}_0)$ .

*Inductive step.* Assume  $\mathbf{x}_{1:n} \sim p_{\text{AR}}(\cdot \mid \mathbf{x}_0)$ . At position  $n + 1$ , proposal  $\hat{x}_{n+1} \sim Q(\cdot \mid \mathbf{x}_{1:n}, \bar{\mathbf{x}})$  is accepted/corrected by the same single-token mechanism with target  $p_{\text{AR}}(\cdot \mid \mathbf{x}_0, \mathbf{x}_{1:n})$ , yielding

$$x_{n+1} \sim p_{\text{AR}}(\cdot \mid \mathbf{x}_0, \mathbf{x}_{1:n}).$$

Hence

$$\Pr(\mathbf{x}_{1:n+1}) = \Pr(\mathbf{x}_{1:n}) p_{\text{AR}}(x_{n+1} \mid \mathbf{x}_0, \mathbf{x}_{1:n}) = p_{\text{AR}}(\mathbf{x}_{1:n+1} \mid \mathbf{x}_0).$$

By induction, the full generated sequence has law  $p_{\text{AR}}$ .  $\square$

## B.2 Proof of Theorem 4.3

*Proof.* We establish the expected acceptance-length gain under EMA smoothing.

**Step 1: Variance reduction from EMA.**

The EMA process is

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

so

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

Under Theorem 4.2,  $\mathbb{E}[E_k] = \mu > 0$ ,  $\text{Var}(E_k) = \sigma^2$ , and increments are uncorrelated. Then

$$\text{Var}(\rho_k) = (1 - \beta)^2 \sum_{j=0}^{k-1} \beta^{2j} \sigma^2 = (1 - \beta)^2 \sigma^2 \frac{1 - \beta^{2k}}{1 - \beta^2}. \quad (8)$$

As  $k \rightarrow \infty$ ,

$$\text{Var}(\rho_k) \rightarrow \frac{(1 - \beta)\sigma^2}{1 + \beta}.$$

For  $\hat{\rho}_k = \rho_k / (1 - \beta^k)$ , asymptotically:

$$\text{Var}(\hat{\rho}_k) \rightarrow \sigma_{\hat{\rho}}^2 := \frac{(1 - \beta)\sigma^2}{1 + \beta}.$$

Thus EMA reduces effective variance by factor  $(1 - \beta)/(1 + \beta)$ .

### Step 2: Survival probability under positive drift.

Let  $S_k$  denote the cumulative smoothed energy proxy. Under normal approximation:

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

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

Acceptance up to position  $k$  requires staying below the acceptance boundary (denote a realized boundary level by  $\eta$ ):

$$\Pr(\text{survive to } k) \approx \Pr(S_k \leq \eta) = \Phi\left(\frac{\eta - k\mu}{\sqrt{k}\sigma_{\text{eff}}}\right).$$

Because  $\mu > 0$ , the mean  $k\mu$  drifts upward toward rejection; reducing  $\sigma_{\text{eff}}$  suppresses transient excursions and improves survival in the high-variance regime.

### Step 3: Expected Acceptance Length Calculation.

The expected acceptance length is:

$$\mathbb{E}[L] = \sum_{k=1}^{\gamma} \mathbb{P}(S_j \leq \tau \text{ for all } j \leq k) \approx \sum_{k=1}^{\gamma} \mathbb{P}(S_k \leq \tau). \quad (9)$$

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

$$\mathbb{P}(S_{k^*} \leq \tau) \approx \Phi\left(\frac{\tau + 1}{\sigma_{\text{eff}} / \sqrt{|\mu|}}\right). \quad (10)$$

The survival probability increases when  $\sigma_{\text{eff}}$  decreases. The derivative is:

$$\frac{\partial}{\partial \sigma_{\text{eff}}} \Phi\left(\frac{\tau + 1}{\sigma_{\text{eff}} / \sqrt{|\mu|}}\right) = -\phi(z) \cdot \frac{(\tau + 1)\sqrt{|\mu|}}{\sigma_{\text{eff}}^2} < 0, \quad (11)$$

where  $\phi$  is the standard normal PDF and  $z$  is the argument of  $\Phi$ .

### Step 4: Derivation of the Gain Formula.

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

$$\Delta\sigma_{\text{eff}} = \sigma \left( \sqrt{\frac{1-\beta}{1+\beta}} - 1 \right) \approx -\frac{\sigma\beta}{2} + O(\beta^2). \quad (12)$$

The corresponding change in survival probability at position  $k$  is:

$$\Delta\mathbb{P}(S_k \leq \tau) \approx \phi(z_k) \cdot \frac{(\tau + k|\mu|)\sqrt{|\mu|}}{\sigma^2} \cdot \frac{\sigma\beta}{2}, \quad (13)$$

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

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

$$\mathbb{E}[L_\beta] - \mathbb{E}[L_0] \approx \sum_{k=1}^{\gamma} \int_0^\infty e^{-\tau} \cdot \phi(z_k) \cdot \frac{(\tau + k|\mu|)\sqrt{|\mu|}}{\sigma^2} \cdot \frac{\sigma\beta}{2} d\tau \quad (14)$$

$$= \frac{\beta\sqrt{|\mu|}}{2\sigma} \sum_{k=1}^{\gamma} \int_0^\infty e^{-\tau} \phi(z_k) (\tau + k|\mu|) d\tau. \quad (15)$$

Let  $L_\beta$  be acceptance length under EMA coefficient  $\beta$ . Then

$$\mathbb{E}[L_\beta] = \sum_{k=1}^{\gamma} \Pr(L_\beta \geq k) \approx \sum_{k=1}^{\gamma} \Pr(S_k \leq \eta),$$

with expectation over random boundary perturbations implicit.

For small  $\beta$ , using

$$\sigma_{\text{eff}}(\beta) = \sigma \sqrt{\frac{1-\beta}{1+\beta}} = \sigma \left( 1 - \frac{\beta}{2} + O(\beta^2) \right),$$

a first-order expansion of the survival terms yields a relative gain proportional to  $(\sigma/\mu)\beta$ . Therefore

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

for some universal constant  $c > 0$ , proving equation 2.  $\square$

## C Training Cost Analysis

Table 3 compares the training requirements of 7B–8B language model acceleration methods under a common accounting framework. It reports the inputs and resulting costs once; the remainder of this section states the shared methodology and the assumptions that differ across training paradigms.

### C.1 Estimation Methodology

For full-model training, we use the standard transformer approximation [Kaplan et al. \(2020\)](#); [Hoffmann et al. \(2022\)](#)

$$C_{\text{full}} = 6PD, \quad (16)$$

where  $P$  is the number of trainable parameters and  $D$  is the number of training tokens. The factor of six combines a two-FLOP forward pass and a four-FLOP backward pass per parameter and token. When the base model is frozen, the corresponding estimate is

$$C_{\text{aux}} = 2P_{\text{base}}D + 6P_{\text{aux}}D, \quad (17)$$

which includes feature extraction through the base model and forward–backward computation through the trainable module.

Table 3: Training cost comparison for 7B–8B parameter language model acceleration methods. GPU-hour estimates assume 203 TFLOPS sustained utilization on H100 GPUs unless marked as empirical. Paradigms are training from scratch (FS), continual pre-training (CP), fine-tuning (FT), and auxiliary module training (AUX).

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

<sup>†</sup>Base model (7–8B) is frozen; only the auxiliary module (draft head or LoRA adapter) is trained.

<sup>‡</sup>Includes the frozen base-model forward pass:  $C = 2P_{\text{base}}D + 6P_{\text{aux}}D$ .

<sup>§</sup>Empirical estimate; utilization is lower than 203 TFLOPS because of memory-bandwidth constraints.

Given global batch size  $B$  in tokens and sustained utilization  $U$ , the remaining columns follow from

$$S = \frac{D}{B}, \quad (18)$$

$$H = \frac{C}{U \times 3600}, \quad U = 203 \times 10^{12} \text{ FLOPS}. \quad (19)$$

The utilization is the mixed-precision Megatron-LM baseline used to normalize analytical estimates. Auxiliary-module training is more sensitive to memory bandwidth, so empirical GPU hours can exceed the FLOP-based value.

## C.2 Method-Specific Assumptions

We analyze seven acceleration methods spanning four training paradigms: training from scratch, continual pre-training, fine-tuning, and auxiliary module training. In Table 3, model size, token budget, and batch size are reported inputs; steps, TFLOPs, and H100 hours are estimates derived from Equations 16–19.

**Training from scratch.** 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. LLaDA [Nie et al. \(2025\)](#) is an 8B masked diffusion language model trained from scratch on 2.3 trillion tokens with a global batch size of 5.2M tokens. Its reported 0.13M H800 GPU hours correspond to an effective utilization of approximately 235 TFLOPS, whereas the table applies the common H100 normalization.

**Continual pre-training.** 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. TiDAR [Liu et al. \(2025\)](#) adapts a pre-trained 8B AR model into a hybrid diffusion-drafting and AR-verification architecture, using 150 billion tokens and a global batch size of 2M tokens.

**Fine-tuning.** 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.

**From assumptions to estimates.** For the full-model rows, we substitute each method’s reported  $P$ ,  $D$ , and  $B$  into Equations 16–19. For example, Dream gives  $C = 6(7 \times 10^9)(600 \times 10^9) = 2.52 \times 10^{22}$  FLOPs, or  $2.52 \times 10^{10}$  TFLOPs;  $S = (600 \times 10^9)/(4 \times 10^6) = 150,000$  steps; and  $H = C/(203 \times 10^{12} \times 3600) \approx 34,500$  H100 hours. The same row-wise calculation produces the baseline, LLaDA, TiDAR, and SBD estimates in the right half of the table.

**Frozen-base auxiliary modules.** EAGLE Li et al. (2024) and EAGLE-3 Li et al. (2025) freeze the 7B–8B base model and train a draft head. EAGLE uses approximately 70M tokens from 68K ShareGPT dialogues and reports 1–2 days on four A100 GPUs. EAGLE-3 expands the corpus to approximately 600M tokens from ShareGPT and UltraChat and trains for ten epochs, yielding roughly 800K steps. For their TFLOP estimates,  $P_{\text{base}}$ ,  $P_{\text{aux}}$ , and  $D$  enter the frozen-base formula above; their GPU-hour entries remain empirical because base-model feature extraction dominates wall-clock time.

**Cross-Regression.** Our configuration follows Appendix D: rank-32 LoRA on all linear layers updates approximately 80M parameters while the 7B base model remains frozen. Twenty epochs over 7K examples of at most 2,048 tokens yield 286M training tokens; a batch of eight sequences corresponds to 16K tokens and 17,500 optimization steps. These values enter the same frozen-base formula as EAGLE. Offline target generation and adapter training both invoke the base model, making nominal FLOPs an imperfect proxy for the reported wall-clock cost.

## D 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 (Suma & Dauncey, 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.

### D.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

**Algorithm 3** LoRA Denoising Distillation

---

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

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

---

$(x_{1:i}, \hat{y}_{i+1:i+w})$  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.

## D.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  $5 \times 10^{-4}$  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.

| Parameter                 | Value                                   |
|---------------------------|-----------------------------------------|
| LoRA rank                 | 32                                      |
| LoRA target               | all-linear                              |
| Training window $w$       | 16                                      |
| Partial-correct threshold | 0.05                                    |
| Optimizer                 | AdamW                                   |
| Learning rate             | $5 \times 10^{-4}$ (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 4: Unified training-script parameters.

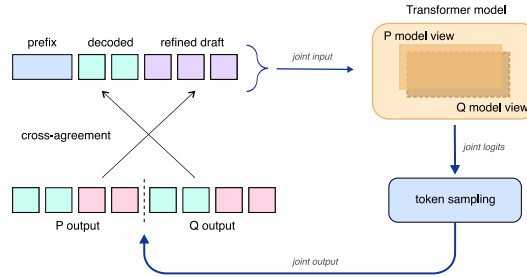


Figure 2: A simplified schematic of the Cross-Regression decoding process. It illustrates the dual-stream architecture where a non-causal predictive stream drafts future tokens that are validated by a causal control stream via a cross-agreement procedure, enabling simultaneous drafting and acceptance.

### D.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.

## E 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  $1, \dots, m$ ), they were sampled from the predictive distribution conditioned on a *different* prefix—specifically, the context before the correction token  $x_{\text{corr}}$  was appended. If we were to re-evaluate these tokens under the current  $Q_k$ , we would be computing  $Q(\hat{x}_k \mid \mathbf{x}, x_{\text{corr}}, \hat{\mathbf{x}}_{1:k-1})$ , which differs from the distribution  $Q(\hat{x}_k \mid \mathbf{x}^{(\text{prev})}, \hat{\mathbf{x}}_{1:k-1})$  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  $\hat{q}_k = Q_k(\hat{x}_k)$  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  $\hat{q}_k$  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 head currently places probability mass.

| Model                         | Size | Acceptance Rate | X-regression Accuracy |
|-------------------------------|------|-----------------|-----------------------|
| <i>Temperature 0 (Greedy)</i> |      |                 |                       |
| 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                  |
| <i>Temperature 0.6</i>        |      |                 |                       |
| 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 5: Acceptance rate on GSM8K with GSM8K-trained adapter. Test window is 32, train window is 16, without look-ahead.

| Model                         | Size | Acceptance Rate |
|-------------------------------|------|-----------------|
| <i>Temperature 0 (Greedy)</i> |      |                 |
| 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            |
| <i>Temperature 0.6</i>        |      |                 |
| 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 6: Acceptance rate on MT-Bench with mixed-trained adapter. Test window is 32, train window is 16, without look-ahead.

**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-head model, the draft buffer  $\hat{x}$  maintains a fixed length of  $\gamma$  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  $\hat{x}$  with  $\gamma$  random placeholder tokens before decoding begins and track the count of valid carried-over tokens  $m$  as a separate variable. The random initial tokens carry no semantic weight—they are immediately overwritten during the first sampling phase since  $m = 0$  triggers fresh sampling across all  $\gamma$  positions. On subsequent iterations, positions 1 through  $m$  retain valid carried-over candidates from the previous draft while positions  $m + 1$  through  $\gamma$  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                         | Size | Acceptance Rate |
|-------------------------------|------|-----------------|
| <i>Temperature 0 (Greedy)</i> |      |                 |
| 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            |
| <i>Temperature 0.6</i>        |      |                 |
| 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 7: Acceptance rate on HumanEval with mixed-trained adapter. Test window is 32, train window is 16, without bidirectional 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 8: GSM8K to GSM8K evaluation under standard configuration with window 32 and threshold 0.

| 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 9: 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 10: Acceptance rate on GSM8K with GSM8K-trained adapter. Test window is 32, train window is 16. Trained and evaluated at temperature 0.1.

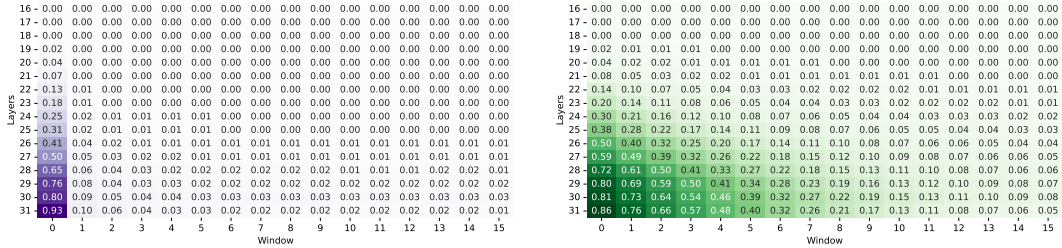


Figure 3: Comparison of per-layer output from vanilla Llama-2-7B-chat (left) and LoRA-tuned (right). Averaged over text generations based on 50 prompts from GSM8K dataset (Cobbe et al., 2021).

| Model      | 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 11: 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<br>$\tau$ | HumanEval<br>$\tau$ | GSM8K<br>$\tau$ |
|-----------------------------|--------------------|---------------------|-----------------|
| <i>LLaMA 2 7B</i>           |                    |                     |                 |
| 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)   | <b>2.70</b>        | 2.88                | <u>1.60</u>     |
| Cross-Regression            | 2.67               | <b>3.50</b>         | <b>3.47</b>     |
| <i>LLaMA 2 13B</i>          |                    |                     |                 |
| PLD (Saxena, 2023)          | 1.42               | 1.63                | 1.41            |
| Lookahead (Fu et al., 2024) | 1.58               | 1.80                | 1.65            |
| Cross-Regression            | <b>2.87</b>        | <b>4.40</b>         | <b>3.90</b>     |

Table 12: Average acceptance lengths  $\tau$  of different speculative decoding methods for standard language models as reported in the relevant papers. **Bold** indicates the best result for each benchmark; underline indicates the second best.

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

Table 13: Comparison between sequential drafting and validation passes versus joint processing in a single pass layout.

| <b>Layer Range</b> | <b>#Layers</b> | <b>Occupancy</b> | $\tau_{\text{Seq}}$ | $\tau_{\text{Joint}}$ |
|--------------------|----------------|------------------|---------------------|-----------------------|
| 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             |                  | <b>4.6</b>          | <b>3.2</b>            |
| Layers 16–31       | 16             |                  | 4.2                 | 2.9                   |
| Layers 0–23        | 24             |                  | 4.6                 | 3.2                   |
| Layers 8–31        | 24             |                  | <b>4.8</b>          | <b>3.3</b>            |
| Full (baseline)    | 32             |                  | 4.7                 | 3.3                   |

Table 14: Acceptance length  $\tau$  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).

| Model Series    | Size | MT-b   | HE     | GSM8K  | Mean   |
|-----------------|------|--------|--------|--------|--------|
|                 |      | $\tau$ | $\tau$ | $\tau$ | $\tau$ |
| 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   |
|                 | 1.5B | 2.41   | 2.51   | 3.75   | 2.89   |
| DS Qwen 2.5     | 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   |
|                 | 1.5B | 2.17   | 2.38   | 3.80   | 2.78   |
| DS Qwen 2.5     | 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 15: Cross-Regression acceptance length ( $\tau$ ) 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.