EDSD: Entropy-Driven Design for Faster Speculative Decoding

Longkai Cheng* **Ximing Wang*** **Jiangcai Zhu** **Kailai Shao**
Chao Chen **Haixiang Hu†**

Ant Group
{chenglongkai.clk, ximing.wxm, jiangcai.zjc,
kailai.skl, chixi.cc, zengxian}@antgroup.com

Abstract

Speculative decoding has emerged as a promising paradigm for accelerating large language model inference by leveraging a lightweight draft model to generate multiple candidate tokens. However, existing methods often incur substantial training overhead to mitigate information misalignment between autoregressive draft model training and decoding. To address this challenge, we propose **EDSD**, an **E**ntropy-**D**riven **S**peculative **D**ecoding framework that uses entropy as a unified, interpretable signal for both draft model training and architectural design. EDSD drives the draft model to progressively align with the target model in an easy-to-hard manner while establishing token-level alignment as a dominant design principle. Extensive experiments on seven LLMs demonstrate that EDSD improves training efficiency by 24.8%, increases the average acceptance length by 4.0%, and achieves a 4.1% speedup compared to state-of-the-art methods. Furthermore, EDSD improves robustness to system prompt variations by more than 5×. Our findings establish entropy-driven alignment as an effective and principled foundation for efficient speculative decoding. We make our draft model weights available at https://github.com/KerwinKai/EDSD.

1 Introduction

Large language models (LLMs) based on the Transformer architecture (Vaswani et al., 2017) have become central to a wide range of domains, demonstrating strong performance in tasks such as coding, mathematics, and reasoning. As LLM are progressively integrated as core components of various intelligent application systems (Achiam et al., 2023; Team et al., 2025), there is a growing demand for them to produce longer and more detailed textual output to achieve better task performance (Wei et al., 2022; Snell et al., 2025; Muennighoff et al., 2025). This trend exacerbates memory access and bandwidth requirements during autoregressive decoding, pushing LLM inference into a memory-bound regime where decoding latency becomes a critical bottleneck for practical deployment.

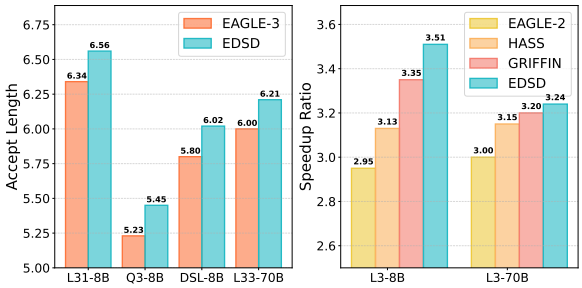

Figure 1: Comparison of acceptance length and speedup ratio across EAGLE-2, HASS, GRIFFIN, EAGLE-3, and EDSD. Model abbreviations are defined in Sec. 4.1.

To mitigate this challenge, speculative decoding (SD) (Leviathan et al., 2023; Chen et al., 2023a) has emerged as a widely adopted acceleration technique (Miao et al., 2024). SD uses a lightweight auxiliary mechanism to propose multiple tokens and then verifies them with the target LLM, enabling multi-token generation per forward pass. By trading additional computation for reduced memory access, it leads to a more favorable regime in the Roofline model (Williams et al., 2009), thus significantly reducing end-to-end decoding latency.

An effective SD framework must achieve a high acceptance rate for draft tokens (Sadhukhan et al., 2025). Recent autoregressive draft model approaches, such as EAGLE (Li et al., 2024b) significantly improve draft model performance by leveraging hidden states from the target LLMs and using a single Transformer layer to guide token prediction. Further advances, including HASS (Zhang et al., 2025a) and GRIFFIN (Hu et al., 2025), introduce multi-step training to better align token and feature level representations between the

*Equal contribution †Corresponding author

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 46243–46260
July 2-7, 2026

draft model's training and decoding phases, while EAGLE-3 (Li et al., 2025b) further demonstrates that scaling the training data can enhance accuracy. When combined, these advances substantially increase training overhead, making efficient draft model training a growing practical concern.

This motivates a rethinking of how to maintain high acceptance rates in draft models without relying on ever-increasing training budgets. A key challenge lies in identifying which tokens and features of the target LLM are most valuable for the draft model to learn. Entropy, a fundamental concept in information theory that quantifies the uncertainty of a random variable (Skean et al., 2025), emerges as a natural signal to guide such decisions. Existing SD methods have begun to exploit entropy at the verification stage: for example, SVIP (Zhang et al., 2025b) discards draft tokens whose entropy exceeds a threshold to avoid redundant verification, and FLY (Li et al., 2025a) leverages entropy-based gating to relax strict token matching. However, these uses of entropy are confined to inference-time filtering and decision rules. Its potential as a *design signal* for shaping the draft model's architecture and training strategy remains largely unexplored.

In this paper, we propose **EDSD** (**E**ntropy-**D**riven **S**peculative **D**ecoding), a plug-and-play framework that uses a unified and interpretable entropy analysis to guide both the training strategy and the architecture of the draft model. EDSD is grounded in two complementary design principles.

First, We empirically observe that tokens with high entropy in the target model's output distribution are significantly more likely to be rejected during verification. This suggests an easy-to-hard learning scheme in which low-entropy (easy) tokens are learned before high-entropy (hard) ones. EDSD implements this principle via an entropy-based token masking strategy that gradually exposes high-entropy tokens during training, together with a dynamic step schedule that aligns the training-time prediction horizon with decoding.

Second, feature-level representations are often mismatched between training and the decoding phase, whereas token-level predictions are directly corrected by the target model through verification. EDSD therefore reduces information inconsistency by biasing the draft model toward relying more on token-level signals. In practice, it uses entropy analysis of target-model representations before training to select layers whose features best support token-level prediction and employs a feature fusion module that allows the draft model to exploit token-level information more robustly during decoding. Our contributions are threefold:

- We identify the entropy of the target model's output as a reliable signal for draft model training, revealing that high-entropy tokens are more likely to be rejected during verification and motivating an easy-to-hard learning scheme.
- We propose EDSD, a plug-and-play framework that leverages entropy to co-design the draft model's training strategy and architecture, reducing the misalignment of information between training and decoding through token scheduling, layer selection, and feature fusion based on entropy.
- We extensively evaluate EDSD across dense and Mixture-of-Experts (MoE) LLMs, demonstrating a 24.8% improvement in training efficiency and up to $5\times$ improved robustness to system prompt variations. In addition, EDSD improves the average acceptance length by 4.0% and achieves a 4.1% speedup over state-of-the-art (SOTA) methods.

2 Preliminaries

In this section, we introduce the necessary background on information theory and speculative decoding, and review representative SOTA.

2.1 Information Theory

Information-theoretic measures offer a powerful lens for analyzing the internal mechanisms of LLMs. Conventionally, Shannon entropy is calculated in output logits to quantify prediction confidence or uncertainty (Shannon, 1948). Matrix-based entropy has been used to assess the structural diversity and expressiveness of hidden internal representations (Sanchez Giraldo et al., 2015).

Let $\mathbf{X} \in \mathbb{R}^{N\times V}$ denote the output logits of an LLM, where N represents the sequence length, and V denotes the vocabulary size. For a specific token $i \in \{1, \ldots, N\}$, we consider its logit vector $\mathbf{x}_i \in \mathbb{R}^V$. We first derive the normalized distribution $\mathbf{s}_i \in \mathbb{R}^V$ applying the Softmax function:

$$\mathbf{s}_i = \mathrm{Softmax}(\mathbf{x}_i), \quad s_{i,j} = \frac{e^{x_{i,j}}}{\sum_{k=1}^{V} e^{x_{i,k}}}. \quad (1)$$

The Shannon entropy $H(\mathbf{s}_i)$ for the i-th token is then defined as:

$$H(\mathbf{s}_i) = -\sum_{j=1}^{V} s_{i,j} \log s_{i,j}. \quad (2)$$

Here, $s_{i,j}$ denotes the probability of the j-th vocabulary item. A higher entropy $H(\mathbf{s}_i)$ indicates a more uniform distribution, implying greater uncertainty in the prediction of the model, while a lower entropy corresponds to a more confident and concentrated probability mass.

To analyze the information capacity of the internal representations, let $\mathbf{Z} \in \mathbb{R}^{N \times D}$ be a matrix of hidden states, where D represents the hidden dimension size. We employ the matrix-based entropy of order α ($\alpha > 0, \alpha \neq 1$) to measure the spectral structure of $\mathbf{Z}$. Let $\mathbf{K} = \mathbf{Z}\mathbf{Z}^\top$ be the Gram matrix associated with the representations. The matrix-based entropy $M_\alpha(\mathbf{Z})$ is defined as:

$$M_\alpha(\mathbf{Z}) = \frac{1}{1-\alpha} \log \left(\sum_{i=1}^{r} \left(\frac{\lambda_i(\mathbf{K})}{\mathrm{tr}(\mathbf{K})} \right)^\alpha \right), \quad (3)$$

where $r = \mathrm{rank}(\mathbf{K})$ and $\lambda_i(\mathbf{K})$ are the non-negative eigenvalues of $\mathbf{K}$, and $\mathrm{tr}(\mathbf{K})$ denotes the trace of $\mathbf{K}$. The term $\frac{\lambda_i(\mathbf{K})}{\mathrm{tr}(\mathbf{K})}$ represents the normalized eigenvalue spectrum, analogous to a probability distribution. A lower value of $M_\alpha(\mathbf{Z})$ indicates a rapidly decaying spectrum, suggesting a highly compressed or low-rank representation.

In this work, we mainly adopt the limit case where $\alpha \to 1$ for simplicity and numerical stability. This limit converges to the Shannon entropy of the spectrum, given by:

$$\begin{aligned} M_1(\mathbf{Z}) &= \lim_{\alpha \to 1} M_\alpha(\mathbf{Z}) \\ &= -\sum_{i=1}^{r} \frac{\lambda_i(\mathbf{K})}{\mathrm{tr}(\mathbf{K})} \log \left(\frac{\lambda_i(\mathbf{K})}{\mathrm{tr}(\mathbf{K})} \right). \end{aligned} \quad (4)$$

2.2 Speculative Decoding

Recently, SD approaches (Leviathan et al., 2023; Chen et al., 2023a) have accelerated inference by employing a draft model to generate candidate tokens, which are subsequently verified by the target model in a single parallel forward pass. This mechanism enables lossless decoding while reducing latency. Formally, let x_i denote the i-th token in a sequence, and let x_t represent the token in the current state. The SD process generally operates in two stages. In the drafting stage, the draft model autoregressively generates K tokens, denoted as $\hat{x}_{t+1:t+K}$, while recording their associated probabilities $\hat{p}$. Subsequently, in the verification stage, the target model evaluates $\hat{x}_{t+1:t+K}$ to compute the ground-truth probabilities p. The acceptance of each candidate token is determined sequentially based on $\hat{p}$ and p. Rejected tokens are discarded and the KV-cache (Pope et al., 2023) of the draft model is updated according to the accepted length.

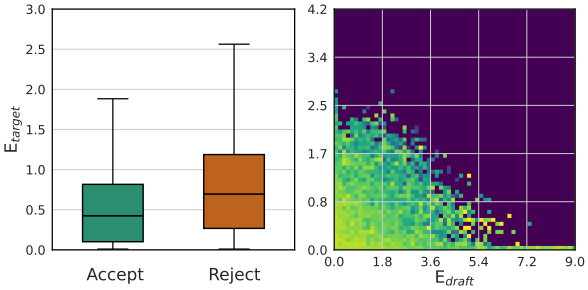

Figure 2: Analysis of entropy. (L) Distribution of accepted vs. rejected tokens across target entropy levels. (R) Acceptance rate heatmap relative to target and draft entropy, where brightness indicates higher acceptance.

Research on SD can be broadly divided into two complementary directions: (i) improving the architecture or training of the draft model, and (ii) optimizing the verification stage. Our work focuses on the former. Autoregressive draft model frameworks such as EAGLE (Li et al., 2024b) establish the basic architecture for modern draft models, using hidden states from the target LLM and a single Transformer layer to guide token prediction. EAGLE-2 (Li et al., 2024a) further refines this design by replacing the fixed draft-token tree with a dynamic one. On the training side, HASS (Zhang et al., 2025a) addresses feature-level misalignment by recursively feeding the output feature $\mathbf{a}_t$ back into the model for multiple steps during training, following the *training-time test* strategy proposed in EAGLE-3 (Li et al., 2025b). GRIFFIN (Hu et al., 2025) instead targets token-level misalignment by feeding the draft model's own predictions, rather than ground-truth tokens, back into the input during training. FR-Spec (Zhao et al., 2025) improves efficiency with a frequency-ranked vocabulary that compresses the search space for candidate selection. Together, these methods constitute the current SOTA methods in the optimization of draft model.

3 Method

To train a faster draft model under realistic training budgets, we focus on improving how the draft model learns from and aligns with the target model. We first derive two entropy-driven design principles from empirical analysis (Sec. 3.1), and then instantiate them in EDSD, which combines training and architecture design for SD (Sec. 3.2).

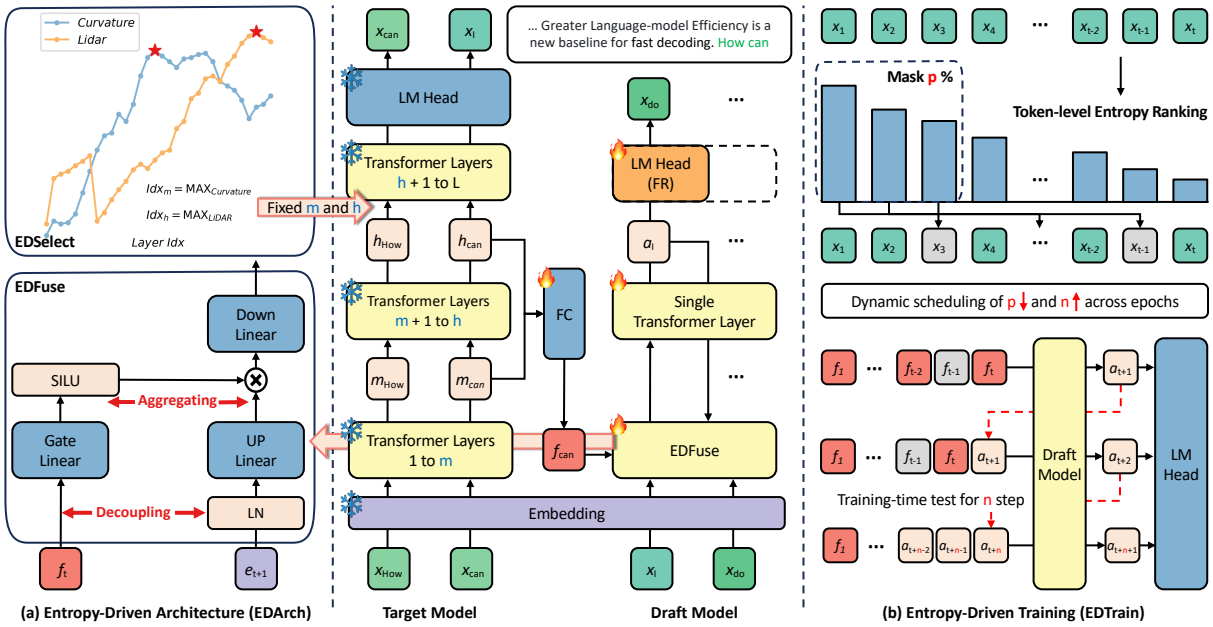

Figure 3: The EDSD framework. Before training, EDSelect identifies informative layers from the target model using a small calibration set and keeps the selection fixed. During training, hidden states from these layers are extracted as features and fused with token embeddings by EDFuse. EDTrain uses target entropy to control token exposure and align step length. At inference, the draft model leverages this structure for efficient generation.

3.1 Design Principles from Empirical Analysis

The effectiveness of a draft model depends on how well it mimics the target model. However, treating all tokens and feature signals uniformly during training is inefficient. We therefore derive two complementary principles to guide the design of EDSD: one governs *how* the draft model should learn, and the other governs *what* information it should trust more from the target model.

Target Output Entropy as a Predictor of Rejection. We first analyze the dynamics of token acceptance. As shown in Fig. 2(left), there is a strong correlation between uncertainty and rejection: tokens are significantly more likely to be rejected when the target model exhibits high output entropy at the corresponding position. Moreover, Fig. 2(right) reveals a distinct entropy gap: the target model predominantly operates in a low-entropy regime, whereas the draft model frequently fluctuates toward high entropy; when the target model is uncertain, the draft model tends to be even more uncertain. A concurrent study (Pankratov and Alistarh, 2026) corroborates this pattern from a theoretical perspective. Taken together, these observations suggest an easy-to-hard learning scheme: the draft model should first focus on low-entropy, easier-to-predict tokens, and progressively incorporate high-entropy, more uncertain ones, rather than allocating excessive capacity to difficult cases prematurely. This principle motivates our entropy-driven training strategy, EDTrain, which schedules learning from deterministic to uncertain patterns.

Prioritize Token-Level Consistency. While intermediate features from the target model provide rich contextual information, they are not used in the same way during training and decoding. Consider generating token x_{t+2}: even if the preceding token x_{t+1} is accepted, the draft model relies on its own hidden state f_{t+1} during autoregressive decoding, rather than on the feature representations seen during training. This discrepancy fundamentally drives feature-level misalignment between the training and decoding phases. We therefore adopt a second principle: the draft model should rely more heavily on token-level signals than on feature-level input, so that training computation is concentrated on signals that remain reliable at inference time. This principle underpins our architectural design, EDArch, and guides both the choice of which layers to extract from the target model and how their features are fused with token-level representations.

3.2 Entropy-Driven Speculative Decoding

Building on prior design principles, we propose EDSD, an entropy-driven framework comprising two core components: EDTrain and EDArch.

Entropy-Driven Training (EDTrain) is an

easy-to-hard learning scheme that guides the training of the draft model. As illustrated in Fig. 3(b), it consists of two mechanisms: Mask-p_t, which controls the exposure of tokens based on target model entropy, and Step-n_t, which progressively aligns the training process with the decoding.

For each training sample, we compute the Shannon entropy of the target model's predictive distribution. Let T denote the total number of training epochs and $t \in \{0, \ldots, T-1\}$ the current epoch. The masking proportion p_t at epoch t is defined as:

$$p_t = \max\big(0,\, T - t - 1\big) \times \gamma, \tag{5}$$

where γ is a hyperparameter (typically $\gamma = 3$). On top of the original loss mask, the top-$p_t\%$ tokens with the highest target entropy are masked, ensuring that training starts with low-entropy tokens and gradually incorporates harder ones.

To reduce feature-level misalignment between training and decoding, previous work introduces the training-time test (Zhang et al., 2025a), where the draft model predicts a fixed number of simulation steps n during training. However, using a fixed n makes training prematurely difficult when the draft model is not yet capable of long-horizon prediction. In Step-n_t, we instead let n grow with the training epoch. Let $S_{\max}$ be the maximum number of simulation steps. The number of steps n_t used in epoch t increases linearly from 1 to $S_{\max}$:

$$n_t = \left\lfloor 1 + (S_{\max} - 1) \cdot \frac{t}{T-1} \right\rfloor. \tag{6}$$

The combined effect of delaying hard tokens and extending the prediction length progressively aligns the draft model with the target model in practice, mitigating both feature and token misalignment. PosS (Huang et al., 2025) is a closely related approach that attributes the degradation of later position draft predictions to error accumulation in draft generated features and addresses it with multiple position-specialized draft layers, where each specialist generates tokens at assigned positions. In contrast, Step-n_t keeps a single draft model and follows EDSD's training principle by progressively increasing the speculative horizon during training, which helps the draft adapt to deeper steps.

Entropy-Driven Architecture (EDArch) determines what to receive from the target model and how to integrate it. As shown in Fig. 3(a), EDArch comprises Entropy-Driven Layer Selection (EDSelect) and Entropy-Driven Feature Fusion (EDFuse).

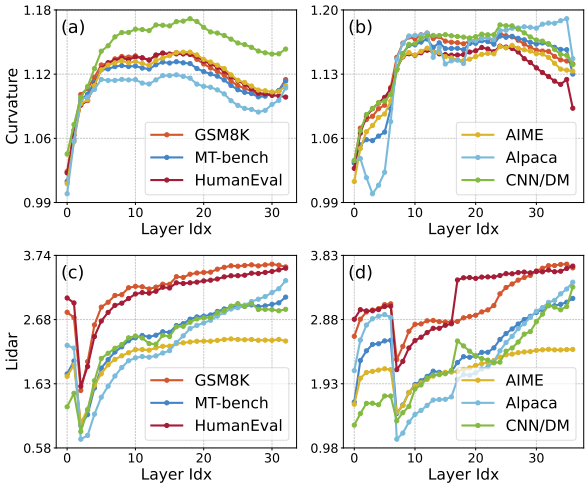

Figure 4: Different models exhibit pronounced variations in representation quality metrics across layers. (a) and (b) illustrate the metrics for L31-8B, while (c) and (d) correspond to Q3-8B. These distinct patterns highlight the necessity of model-specific layer selection.

EDSelect selects target-model layers whose representations best support token-level consistency. We build on the information-theoretic framework of Skean et al. (2025) and characterize each layer using two metrics derived from matrix-entropy: Geometry, quantified by Curvature (Hosseini and Fedorenko, 2023), which measures how well the embedding space preserves fine-grained token distinctions, and Invariance, measured by Lidar (Thilak et al., 2024), which reflects the robustness of the representation to input perturbations. For both metrics, higher values indicate better-quality representations. Before training, we run a lightweight calibration step: for each layer of the target model, we compute Curvature and Lidar on a subset of 128 training examples. This EDSelect procedure takes about 15 minutes and is negligible compared to the overall training time. For each model, we then identify the layer with the highest Curvature and the layer with the highest Lidar, and use these two layers as feature sources for the draft model in EDSD. Fig. 4 reports the Curvature and Lidar profiles for Llama-3.1-Instruct 8B and Qwen3 8B, showing differences between layers and highlighting the need for model-specific layer selection.

EDFuse implements the principle of prioritizing token-level consistency by adaptively fusing feature-level and token-level representations. As shown in Fig. 3(a), EDFuse processes the two inputs through parallel streams: the feature input passes through a gating linear layer with SiLU activation, while the token input is first normalized by

		MT-bench		HumanEval		GSM8K		Alpaca		CNN/DM		Mean	
Model	Method	SR	τ	SR	τ	SR	τ	SR	τ	SR	τ	SR	τ
L31-8B	PLD	1.69×	2.87	2.49×	2.81	1.29×	2.65	0.89×	1.04	1.46×	2.14	1.56×	2.30
	REST	2.18×	3.93	2.48×	3.65	1.82×	3.19	2.46×	3.25	1.99×	3.13	2.19×	3.43
	Lookahead	1.80×	3.09	2.55×	3.58	1.39×	2.45	1.43×	2.01	1.67×	2.58	1.77×	2.74
	EAGLE-2	2.64×	4.12	3.10×	4.76	2.77×	4.31	2.85×	4.14	2.19×	3.47	2.71×	4.16
	EAGLE-3	3.57×	6.24	4.18×	6.76	3.77×	6.38	3.96×	6.87	3.15×	5.45	3.73×	6.34
	EDSD	**3.75×**	**6.52**	**4.33×**	**6.80**	**3.98×**	**6.68**	**4.11×**	**7.16**	**3.30×**	**5.66**	**3.92×**	**6.56**
Q3-8B	EAGLE-3	2.99×	5.10	3.18×	5.46	3.60×	5.84	3.19×	5.28	2.70×	4.48	3.13×	5.23
	EDSD	**3.28×**	**5.30**	**3.41×**	**5.62**	**3.70×**	**6.11**	**3.35×**	**5.43**	**2.85×**	**4.76**	**3.32×**	**5.45**
DSL-8B	EAGLE-3	3.34×	5.82	4.14×	6.55	4.30×	6.84	3.23×	5.41	2.56×	4.36	3.51×	5.80
	EDSD	**3.54×**	**6.14**	**4.29×**	**6.76**	**4.40×**	**6.95**	**3.30×**	**5.52**	**2.80×**	**4.74**	**3.67×**	**6.02**
L33-70B	EAGLE-3	4.10×	5.80	4.94×	6.69	4.55×	6.09	4.63×	6.30	3.45×	5.11	4.34×	6.00
	EDSD	**4.16×**	**5.99**	**4.97×**	**6.84**	**4.60×**	**6.43**	**4.68×**	**6.55**	**3.46×**	**5.25**	**4.37×**	**6.21**
Q3-30B-A3B	EAGLE-3	3.52×	4.14	4.81×	5.62	4.02×	4.59	3.20×	3.72	3.82×	4.47	3.87×	4.51
	EDSD	**3.82×**	**4.51**	**5.02×**	**5.89**	**4.37×**	**4.95**	**3.53×**	**4.11**	**3.86×**	**4.58**	**4.12×**	**4.81**
L3-8B	EAGLE-2	2.84×	4.21	3.49×	5.03	2.96×	4.40	3.03×	4.17	2.44×	3.76	2.95×	4.31
	HASS	2.99×	4.65	3.75×	5.72	3.26×	5.05	3.12×	4.60	2.52×	4.27	3.13×	4.86
	GRIFFIN	3.13×	4.85	3.99×	5.98	3.55×	5.32	3.19×	4.87	2.90×	4.50	3.35×	5.10
	EDSD	**3.37×**	**5.12**	**4.02×**	**6.04**	**3.69×**	**5.66**	**3.56×**	**5.23**	**2.92×**	**4.53**	**3.51×**	**5.32**
L3-70B	EAGLE-2	2.93×	4.05	3.20×	5.12	3.03×	4.40	3.04×	4.12	2.79×	3.75	3.00×	4.29
	HASS	3.05×	4.61	3.38×	5.93	3.24×	5.32	3.18×	4.73	2.91×	4.36	3.15×	4.99
	GRIFFIN	3.06×	4.66	3.48×	6.03	3.26×	5.39	3.22×	4.76	3.00×	4.47	3.20×	5.06
	EDSD	**3.07×**	**4.70**	**3.50×**	**6.10**	**3.32×**	**5.50**	**3.31×**	**5.01**	**3.02×**	**4.55**	**3.24×**	**5.17**

Table 1: Comparison of different SD methods on standard LLM benchmarks with Temperature $T = 0$, including the speedup ratio SR and the average acceptance length τ. The temperature $T = 1$ is given in Tab. 12.

LayerNorm and then projected by an up-projection linear layer. The resulting representations are fused and then projected back to the original hidden dimension with a down-projection linear layer. This design encourages the draft model to rely more on token-level information while still benefiting from target features. EDFuse is configurable to accommodate different deployment budgets. The projection layers can either maintain the hidden size of the target model d, without incurring additional parameter overhead compared to EAGLE-3, or expand to the intermediate size m (where $m > d$) to increase the capacity of the draft model. Unless otherwise noted, we adopt the expanded m.

4 Experiment

4.1 Implementation Details

Models. We conduct a comprehensive and systematic evaluation of the proposed framework across seven LLMs: Llama-3-Instruct 8B/70B, Llama-3.1-Instruct 8B and Llama-3.3-Instruct 70B (Grattafiori et al., 2024), DeepSeek-R1-Distill-Llama 8B (DeepSeek-AI et al., 2025), Qwen3 8B and Qwen3-Instruct 30B-A3B (Yang et al., 2025). As summarized in Tab. 15, these include both dense and mixture-of-expert architectures, spanning a wide range of parameter scales and training paradigms. To facilitate presentation, we adopt the following abbreviations: L3-8B, L3-70B, L31-8B, L33-70B, DSL-8B, Q3-8B and Q3-30B-A3B.

Datasets and Tasks. To ensure fair comparison with EAGLE-3 (Li et al., 2025b), we use ShareGPT and UltraChat (Ding et al., 2023) as primary training corpora, resulting in a total of 532k data entries. To better align the draft models with the output distribution of the target models, we regenerate responses using the corresponding target model rather than directly adopting entries from the fixed datasets. In addition, for DSL-8B, we further train OpenThoughts-114k-math (Guha et al., 2025) to enhance its mathematical reasoning ability.

We assess performance across five key tasks: MT-Bench (Zheng et al., 2023), HumanEval (Chen et al., 2021), GSM8K (Cobbe et al., 2021), Alpaca (Taori et al., 2023) and CNN Daily Mail (CNN/DM) (Nallapati et al., 2016), which respectively cover the categories of capability: multi-turn conversation, code generation, mathematical reasoning, instruction following, and summarization.

Training and Evaluation Configuration. Our training framework is implemented based on SpecForge (Li et al., 2026). We optimize the draft models using AdamW (β_1=0.9, β_2=0.95) with a learning rate of 5e−5 and a maximum sequence length of 2,048 tokens. Specifically for EDSD training, we set the training epochs T in Eq. 5 to 8 and the maximum simulation step $S_{\max}$ in Eq. 6 to 7. All training is conducted on 8x H800 GPUs. The EDSD draft model for the 70B backbone takes approximately 7 days to train, whereas EDSD models on smaller backbones typically require fewer than 5 days. Training cost details are reported in Tab. 7.

For evaluation, we based it on Spec-Bench (Xia et al., 2024). Unless otherwise specified, we use a draft tree depth of 8, select 6 nodes and set the total number of draft tokens to 60 during the expansion phase, mirroring the optimal configuration reported for EAGLE-3 (Li et al., 2025b). The evaluation of 70B models is conducted on 2x H800 GPUs and on a single H800 GPU for all other models.

Metrics. Since SD is provably lossless, we focus solely on inference efficiency. We adopt standard metrics from previous work: 1) **Speedup Ratio** (SR), indicating the actual acceleration relative to vanilla autoregressive decoding; and 2) **Average Acceptance Length** (τ), the average count of draft tokens accepted per draft-verification cycle.

4.2 Comparison with SOTA Methods

We benchmark EDSD with recent SOTA speculative decoding methods with a batch size of 1 and temperatures in $\{0, 1\}$, including PLD (Saxena, 2023), REST (He et al., 2024), Lookahead (Fu et al., 2024), EAGLE-2 (Li et al., 2024a), HASS (Zhang et al., 2025a), GRIFFIN (Hu et al., 2025) and EAGLE-3 (Li et al., 2025b). Specifically for the L3-8B/70B models, we adopt a distinct configuration to ensure a fair comparison with the official implementations of HASS and GRIFFIN. For these two models, we train the EDSD exclusively on the ShareGPT dataset and, during inference, structure the draft tree with a depth of 6 and select 10 nodes for expansion.

Tab. 1 reports the acceleration performance of EDSD. Across all tasks and target models, EDSD achieves the highest SR and τ. It produces an average acceptance length τ of 5.65, corresponding to improvements of 4.3% over EAGLE-3, 3.2% over GRIFFIN, and 6.5% over HASS. In terms of speed, it achieves a speedup of about 3.74× over vanilla decoding, improving EAGLE-3, GRIFFIN, and HASS by 4.6%, 3.0% and 7.5%, respectively. We also highlight three empirical observations. 1) vocabulary size affects acceptance length: comparing Q3-8B with L31-8B, the mean acceptance length drops by about 1.11 tokens for Q3-8B. This is consistent with Qwen3's larger vocabulary, which allows the model to represent the same content with fewer tokens, naturally reducing the number of tokens accepted per step. 2) MoE target models tend to yield lower speedup ratios for speculative decoding. For Q3-30B-A3B, both EAGLE-3 and EDSD achieve smaller gains compared to dense targets. This is because the MoE router distributes draft tokens between experts during verification, increasing compute and communication overhead. This observation underscores the importance of high draft-token acceptance rates for MoE-based targets. 3) The training data strongly influence the behavior of the draft model in specific tasks. For example, DSL-8B achieves the highest SR and τ on GSM8K, which we attribute to additional training on OpenThoughts-114k-math, which yields a better alignment with mathematical reasoning.

Method	SR	τ	Time (GPU Hours)
EDTrain	**3.32**×	**5.76**	**64.1 (-24.8%)**
w/o Step-n	3.28×	5.75	86.4
w/o Mask-p	3.25×	5.69	63.5
w/o Step-n & Mask-p	3.22×	5.49	80.0

Table 2: Ablation study of EDTrain. Training time is measured in GPU-hours (number of GPUs × wall-clock time) under the same epochs. Details in Tab. 13.

Discussion on Multimodal LLMs. Fig. 5 illustrates the potential of EDSD in multimodal settings. We evaluate our approach in Qwen3-VL 8B (Bai et al., 2025), training the draft model in a 10K subset of ShareGPT4V (Chen et al., 2023b). Compared with the EAGLE-3 implementation, EDSD improves τ from 3.19 to 3.34 and increases SR from 2.01× to 2.12×, demonstrating that our entropy driven design also benefits MLLMs.

4.3 Ablation Study

We evaluate the impact of the two key components of EDSD in the same setup as Sec. 4.1. Unless otherwise noted, ablation studies are conducted in ShareGPT with L31-8B as the default target model.

Effect of EDTrain. EDTrain implements an easy-to-hard learning scheme with two mechanisms: Step-n_t, which gradually increases the speculative prediction length over epochs, and Mask-p_t, which gradually exposes higher-entropy tokens dur-

Model	Method	Sys-Default	Sys-Robustness
L31-8B	EAGLE-3	6.34	6.17 (-2.7%)
	EDArch	**6.56**	**6.53 (-0.5%)**
DSL-8B	EAGLE-3	5.80	2.23 (-61.6%)
	EDArch	**6.02**	**5.33 (-11.5%)**

Table 3: Evaluation of robustness based on the reduction in τ under system-prompt variations across SD methods.

Method	Batch Size					
	8	16	32	64	128	256
EAGLE-3	1.75×	1.61×	1.56×	1.39×	1.33×	1.15×
EDSD	**1.99×**	**1.76×**	**1.62×**	**1.46×**	**1.36×**	**1.30×**

Table 4: Comparison of throughput acceleration between EDSD and EAGLE-3 under varying batch sizes.

ing training. As shown in Tab. 2, removing either component slightly degrades performance: without Step-n and Mask-p leads to a lower SR and τ. In contrast, full EDTrain achieves the highest SR and τ, while reducing the training time from 80.0 to 64.1 GPU hours. This corresponds to a 24.8% improvement in training efficiency, or equivalently a 19.9% reduction when normalized by the baseline training time, reflecting two standard reporting conventions. This demonstrates that jointly scheduling prediction horizon and token difficulty consistently outperforms either strategy alone.

Effect of EDArch. We next examine the effect of EDArch, which controls which target-model features are used (via EDSelect) and how they are fused with token-level signals (via EDFuse). Since EDArch governs the information pathway from the target to the draft model, it is the main driver of robustness to feature-distribution shifts.

Motivated by prior work OSD (Liu et al., 2024) and CORAL (Weng et al., 2025), we evaluate robustness under changes in prompt format and style, a common scenario in post-deployment prompt engineering. We sample 10 generic system prompts from SystemCheck (Mu et al., 2024) and, for each target model, replace its default system prompt with each alternative in turn, re-running evaluation each time. The sampled prompts are listed on Tab. 14. We then compare EDArch with the baseline EAGLE-3 in L31-8B and DSL-8B in terms of degradation in τ. As shown in Tab. 3, EDSD with EDArch exhibits substantially smaller performance drops: the reduction in τ is mitigated by 2.2% and 50.1% on average relative to EAGLE-3, 5× smaller degradation in these cases. This suggests that the EDArch, combining EDSelect and EDFuse, makes the draft model more resilient to prompt-style changes in realistic deployment.

Next, we study the impact of EDSelect on the performance of the draft-model. For each model, EDSelect scores all layers using the Curvature and Lidar metrics and selects the two highest-ranked layers as feature sources for the draft model; the selected layers for each model are reported in Tab. 15. Since the layer choices for L31-8B are very close to the default configuration used by EAGLE-3, we perform ablations on Q3-8B, where the entropy-based ranking leads to a different selection. As shown in Tab. 6, using EDSelect produces an improvement of 2.7% in τ, and applying the same selection to EAGLE-3 also brings an additional gain of 1.2% in τ. These results suggest that model-specific layer selection guided by entropy effectively enhances target-model representations in SD.

We then isolate the effect of EDFuse within EDArch. Tab. 9 evaluates the impact of different hidden sizes in EDFuse on τ and SR. Considering that, in production inference, the acceptance rate has a greater influence on overall speedup than the marginal cost of generating draft tokens, we adopt the intermediate size m as the default configuration. As also shown in Tab. 10, the removal of components from EDFuse leads to marked degradation in both τ and SR. EDFuse also outperforms an FSPAD Feature Sampling module (Gui et al., 2024), a residual MLP-style component, confirming that adaptive fusion of representations enables more effective use of prioritizing token-level consistency.

4.4 Acceleration in Production Frameworks

EDSD integrates seamlessly into highly optimized production inference frameworks with lightweight, modular customization. We extend the SGLang (Zheng et al., 2024) `v0.5.6.post2` runtime with a custom patch that assigns independent operations to separate CUDA streams, overlapping the lightweight EDFuse computation with other kernels. To reduce redundant draft-model computation while maximizing end-to-end throughput, we adopt a depth-4 chain instead of a draft tree, following prior deployment practices (Li et al., 2025b). With higher acceptance rates, EDSD achieves improved throughput across diverse workloads.

EDFuse is implemented as a lightweight MLP-style module, enabling straightforward integration into existing inference frameworks. Frameworks that support standard MLP blocks can incorporate EDFuse with minimal changes by extending the

MLP to accept two inputs: token embeddings and auxiliary features. EDFuse applies LayerNorm to token embeddings, which can be fused into the embedding layer after training to reduce operator overhead at inference time. We evaluate EDSD under a production-style workload of 400 unique prompts sampled from the five benchmarks in Sec. 4.1, cycled during execution. Requests follow a Poisson process with a fixed average rate, and inter-arrival times follow an exponential distribution. We set the temperature to 0.0. The average input length is 226.90 tokens (max 1507), and the average output length is 1165.85 tokens (max 2048).

We consider two evaluation settings. First, we report single-GPU micro-benchmarks to measure per-device throughput and latency without cross-replica interference. Second, we evaluate large-scale deployment with 600 inference instances, each corresponding to a server with 8x H800 GPUs. Within each instance, we use tensor parallelism (TP=1) and data parallelism (DP=8), with a cache-aware API router for the request distribution. Then, we measure output throughput in tokens per second under fixed batch sizes on a single H800 GPU. Tab. 4 shows the relative acceleration over EAGLE-3. EDSD consistently achieves higher throughput across batch sizes, with an average gain of approximately 8% over EAGLE-3 and up to 30% over the non-speculative baseline at batch size 256.

Under large-scale deployment, we evaluate end-to-end latency at fixed ingress rates. In co-located settings, EDSD reduces latency by 10%–15% compared to EAGLE-3. At a representative operating point, the per-DP-rank request rate is 5 QPS, corresponding to an aggregate cluster ingress of approximately 20,400 QPS after accounting for an approximately 15% load-balancing overhead across instances. At this scale, the average end-to-end latency is 10.40 seconds. The latency reduction is primarily driven by the higher acceptance rate and robustness of EDSD, which reduce verification overhead per generated token. Tab. 5 reports latency under varying workloads, where EDSD consistently outperforms EAGLE-3, with larger gains under higher system load. To clarify the measurement setup, single-GPU results correspond to per-device micro-benchmarks, while large-scale results are obtained from a cluster of 600 inference instances with DP=8 per instance. QPS is reported at the per-DP-rank level for latency measurements, while aggregated throughput is computed by scaling over DP ranks and instances. Overall, EDSD consistently improves both throughput and latency under realistic production workloads, while remaining compatible with existing inference systems.

Method	QPS=2	QPS=3	QPS=4	QPS=5
w/o speculative	11.92	14.92	18.45	28.49
EAGLE-3	5.90	7.21	9.22	11.58
EDSD	**5.04**	**6.37**	**8.56**	**10.40**

Table 5: End-to-end latency (seconds) under varying workloads, QPS denotes the per-DP-rank request rate.

5 Conclusion

We presented EDSD, an entropy-driven framework for speculative decoding that jointly optimizes draft model training and architecture. Guided by two complementary principles, an easy-to-hard learning scheme and prioritizing token-level signals, EDSD improves acceptance length and end-to-end speedup while reducing training cost, and demonstrates strong robustness to prompt variations.

Limitations

EDSD focuses on entropy-driven training and architecture design for a single-layer Transformer draft model, without exploring more complex draft architectures such as deeper variants. Although EDSD consistently improves acceptance behavior and decoding efficiency, these inference gains remain relatively modest. Its practical advantages are more evident in reduced training time and improved robustness to prompt variations. In addition, EDTrain and EDArch constitute only one realization of the framework, and alternative training or architectural designs remain to be explored.

Ethics Statement

We propose an entropy-driven framework for speculative decoding drafters. All models and datasets used in our experiments were obtained from widely adopted open-source communities. Although our approach can facilitate the efficient deployment of LLMs, we recognize the need for careful consideration of the ethical aspects associated with their usage. Responsible use of LLMs requires addressing potential biases, ensuring interpretability of the output, protecting data privacy, and conducting risk assessments to prevent misuse or harm. We are committed to fostering research transparency by releasing our implementation details, code, and model weights to enable responsible verification and evaluation by the research community.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, and 45 others. 2025. Qwen3-vl technical report. *Preprint*, arXiv:2511.21631.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. 2023a. Accelerating large language model decoding with speculative sampling. *Preprint*, arXiv:2302.01318.

Lin Chen, Jinsong Li, Xiaoyi Dong, Pan Zhang, Conghui He, Jiaqi Wang, Feng Zhao, and Dahua Lin. 2023b. Sharegpt4v: Improving large multimodal models with better captions. *Preprint*, arXiv:2311.12793.

Lin Chen, Jinsong Li, Xiaoyi Dong, Pan Zhang, Yuhang Zang, Zehui Chen, Haodong Duan, Jiaqi Wang, Yu Qiao, Dahua Lin, and Feng Zhao. 2024. Are we on the right way for evaluating large vision-language models? In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, and 39 others. 2021. Evaluating large language models trained on code. *Preprint*, arXiv:2107.03374.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, and 181 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. 2023. Enhancing chat language models by scaling high-quality instructional conversations. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3029–3051, Singapore. Association for Computational Linguistics.

Yichao Fu, Peter Bailis, Ion Stoica, and Hao Zhang. 2024. Break the sequential dependency of llm inference using lookahead decoding. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Etash Guha, Ryan Marten, Sedrick Keh, Negin Raoof, Georgios Smyrnis, Hritik Bansal, Marianna Nezhurina, Jean Mercat, Trung Vu, Zayne Sprague, Ashima Suvarna, Benjamin Feuer, Liangyu Chen, Zaid Khan, Eric Frankel, Sachin Grover, Caroline Choi, Niklas Muennighoff, Shiye Su, and 31 others. 2025. Openthoughts: Data recipes for reasoning models. *Preprint*, arXiv:2506.04178.

Lujun Gui, Bin Xiao, Lei Su, and Weipeng Chen. 2024. Boosting lossless speculative decoding via feature sampling and partial alignment distillation. *Preprint*, arXiv:2408.15562.

Zhenyu He, Zexuan Zhong, Tianle Cai, Jason Lee, and Di He. 2024. REST: Retrieval-based speculative decoding. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 1582–1595, Mexico City, Mexico. Association for Computational Linguistics.

Eghbal A. Hosseini and Evelina Fedorenko. 2023. Large language models implicitly learn to straighten neural sentence trajectories to construct a predictive representation of natural language. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Shijing Hu, Jingyang Li, Xingyu Xie, Zhihui Lu, Kim-Chuan Toh, and Pan Zhou. 2025. Griffin: Effective token alignment for faster speculative decoding. *Preprint*, arXiv:2502.11018.

Langlin Huang, Chengsong Huang, Jixuan Leng, Di Huang, and Jiaxin Huang. 2025. Poss: Position specialist generates better draft for speculative decoding. *Preprint*, arXiv:2506.03566.

Yicheng Ji, Jun Zhang, Heming Xia, Jinpeng Chen, Lidan Shou, Gang Chen, and Huan Li. 2025. SpecVLM: Enhancing speculative decoding of video LLMs via verifier-guided token pruning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 7205–7219, Suzhou, China. Association for Computational Linguistics.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

Jinze Li, Yixing Xu, Guanchen Li, Shuo Yang, Jinfeng Xu, Xuanwu Yin, Dong Li, Edith C. H. Ngai, and Emad Barsoum. 2025a. Training-free loosely speculative decoding: Accepting semantically correct drafts beyond exact match. *Preprint*, arXiv:2511.22972.

Shenggui Li, Chao Wang, Yikai Zhu, Yubo Wang, Fan Yin, Shuai Shi, Yefei Chen, Xiaomin Dong, Qiaoling Chen, Jin Pan, Ji Li, Laixin Xie, Yineng Zhang, Lei Yu, Yonggang Wen, Ivor Tsang, and Tianwei Zhang. 2026. Specforge: A flexible and efficient open-source training framework for speculative decoding. *Preprint*, arXiv:2603.18567.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. 2024a. EAGLE-2: Faster inference of language models with dynamic draft trees. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 7421–7432, Miami, Florida, USA. Association for Computational Linguistics.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. 2024b. Eagle: speculative sampling requires rethinking feature uncertainty. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

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

Xiaoxuan Liu, Lanxiang Hu, Peter Bailis, Alvin Cheung, Zhijie Deng, Ion Stoica, and Hao Zhang. 2024. Online speculative decoding. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Zhengxin Zhang, Rae Ying Yee Wong, Alan Zhu, Lijie Yang, Xiaoxiang Shi, Chunan Shi, Zhuoming Chen, Daiyaan Arfeen, Reyna Abhyankar, and Zhihao Jia. 2024. Specinfer: Accelerating large language model serving with tree-based speculative inference and verification. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, ASPLOS '24, page 932–949, New York, NY, USA. Association for Computing Machinery.

Norman Mu, Jonathan Lu, Michael Lavery, and David Wagner. 2024. A closer look at system message robustness. In *Neurips Safe Generative AI Workshop 2024*.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candes, and Tatsunori Hashimoto. 2025. s1: Simple test-time scaling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 20286–20332, Suzhou, China. Association for Computational Linguistics.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Çağlar Gulçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290, Berlin, Germany. Association for Computational Linguistics.

Sergey Pankratov and Dan Alistarh. 2026. Speculative decoding speed-of-light: Optimal lower bounds via branching random walks. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6404–6418, Rabat, Morocco. Association for Computational Linguistics.

Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. 2023. Efficiently scaling transformer inference. In *Proceedings of Machine Learning and Systems*, volume 5, pages 606–624. Curan.

Ranajoy Sadhukhan, Jian Chen, Zhuoming Chen, Vashisth Tiwari, Ruihang Lai, Jinyuan Shi, Ian En-Hsu Yen, Avner May, Tianqi Chen, and Beidi Chen. 2025. Magicdec: Breaking the latency-throughput tradeoff for long context generation with speculative decoding. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

Luis Gonzalo Sanchez Giraldo, Murali Rao, and Jose C. Principe. 2015. Measures of entropy from data using infinitely divisible kernels. *IEEE Transactions on Information Theory*, 61(1):535–548.

Apoorv Saxena. 2023. Prompt lookup decoding.

Claude E Shannon. 1948. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423.

Oscar Skean, Md Rifat Arefin, Dan Zhao, Niket Nikul Patel, Jalal Naghiyev, Yann LeCun, and Ravid Shwartz-Ziv. 2025. Layer by layer: Uncovering hidden representations in language models. In *Forty-second International Conference on Machine Learning*.

Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2025. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *The Thirteenth International Conference on Learning Representations*.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. 2023. Alpaca: A strong, replicable instruction-following model. *Stanford Center for Research on Foundation Models. https://crfm. stanford. edu/2023/03/13/alpaca. html*, 3(6):7.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M. Dai, Anja Hauth, Katie Millican, David Silver, Melvin Johnson, Ioannis Antonoglou, Julian Schrittwieser, Amelia Glaese, Jilin Chen, Emily Pitler, Timothy Lillicrap, Angeliki Lazaridou, and 1332 others. 2025. Gemini: A family of highly capable multimodal models. *Preprint*, arXiv:2312.11805.

Vimal Thilak, Chen Huang, Omid Saremi, Laurent Dinh, Hanlin Goh, Preetum Nakkiran, Joshua M. Susskind, and Etai Littwin. 2024. LiDAR: Sensing linear probing performance in joint embedding SSL architectures. In *The Twelfth International Conference on Learning Representations*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Songsheng Wang, Rucheng Yu, Zhihang Yuan, Chao Yu, Feng Gao, Yu Wang, and Derek F. Wong. 2025. Spec-VLA: Speculative decoding for vision-language-action models with relaxed acceptance. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 26928–26940, Suzhou, China. Association for Computational Linguistics.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

Yepeng Weng, Dianwen Mei, Huishi Qiu, Xujie Chen, Li Liu, Jiang Tian, and Zhongchao Shi. 2025. CORAL: Learning consistent representations across multi-step training with lighter speculative drafter. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5580–5593, Vienna, Austria. Association for Computational Linguistics.

Samuel Williams, Andrew Waterman, and David Patterson. 2009. Roofline: an insightful visual performance model for multicore architectures. *Commun. ACM*, 52(4):65–76.

Heming Xia, Zhe Yang, Qingxiu Dong, Peiyi Wang, Yongqi Li, Tao Ge, Tianyu Liu, Wenjie Li, and Zhifang Sui. 2024. Unlocking efficiency in large language model inference: A comprehensive survey of speculative decoding. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 7655–7671, Bangkok, Thailand. Association for Computational Linguistics.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Lefan Zhang, Xiaodan Wang, Yanhua Huang, and Ruiwen Xu. 2025a. Learning harmonized representations for speculative sampling. In *International Conference on Learning Representations*.

Ziyin Zhang, Jiahao Xu, Tian Liang, Xingyu Chen, Zhiwei He, Rui Wang, and Zhaopeng Tu. 2025b. Draft model knows when to stop: Self-verification speculative decoding for long-form generation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 16696–16708, Suzhou, China. Association for Computational Linguistics.

Weilin Zhao, Tengyu Pan, Xu Han, Yudi Zhang, Ao Sun, Yuxiang Huang, Kaihuo Zhang, Weilun Zhao, Yuxuan Li, Jie Zhou, Hao Zhou, Jianyong Wang, Zhiyuan Liu, and Maosong Sun. 2025. FR-spec: Accelerating large-vocabulary language models via frequency-ranked speculative sampling. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3909–3921, Vienna, Austria. Association for Computational Linguistics.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. 2024. Sglang: efficient execution of structured language model programs. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA. Curran Associates Inc.

A Appendix

A.1 Analysis of Layer Selection in EDSelect

The Entropy-Driven Layer Selection (EDSelect) method is motivated by recent speculative decoding approaches that improve autoregressive draft models by leveraging hidden states from the target LLM. Existing methods typically choose feature layers in a fixed heuristic manner[1], such as always using the final layer or manually selecting low, mid, and high layers. In all cases, the layer selection is performed before training the draft model and kept fixed during both training and decoding. This suggests that spending a small amount of computation in advance to identify which target layers provide the most useful features can be beneficial.

EDSelect provides a quantitative, entropy-based alternative to these heuristics. For each target model, it characterizes every layer using two metrics derived from matrix entropy: Curvature and Lidar, and then selects the layers with the highest scores under these criteria, following the principle of prioritizing token-level consistency. As shown in Tab. 6, EDSelect can be integrated into the EAGLE-3 framework by keeping the original low-level layer and replacing the middle and high layers with those that maximize Curvature and Lidar, respectively. We further show that, within EDSD, the use of two entropy-selected layers is sufficient and often better than using three, leading to our final two-layer EDSelect configuration. Curvature and LiDAR have been shown to be related to the Gram matrix K (Skean et al., 2025) and therefore to Eq. 4.

A.2 Analysis for the Architecture of EDFuse

EDFuse is designed to implement the principle of prioritizing token-level consistency by adaptively fusing feature-level and token-level representations. As described in Sec. 3.2, it processes the two inputs through parallel streams and uses a decoupling and aggregation structure, allowing token-level signals to dominate draft-token generation while still leveraging informative target features. In this section, we analyze two aspects of EDFuse in more detail: the choice of its expansion dimension and the effectiveness of its architectural.

Ablation on the Expansion Dimension of EDFuse. EDFuse introduces additional linear projections whose hidden dimension can be configured according to deployment needs. Using L31-8B as an example, we carry out this ablation in the ShareGPT data set and consider three settings for the projection dimension: (i) the hidden size of the target model d, which incurs no additional parameter overhead compared to EAGLE-3; (ii) the intermediate size of the FFN of the target model m, which increases the expressiveness of EDFuse; and (iii) a larger setting of $5d$, which further increases capacity but also computation. As shown in Tab. 9, moving from d to m consistently improves both τ and SR, while increasing the dimension beyond m yields only marginal gains at noticeably higher computational cost. In general, these results support the setting of the EDFuse expansion dimension to the intermediate size of the target model FFN m, which provides a favorable balance between fusion performance and efficiency.

Ablation on the EDFuse Architecture We also ablate the internal components of EDFuse. Specifically, we compare the full EDFuse design with two variants: one that removes the LayerNorm on the token stream, and another that replaces EDFuse with a Feature Sampling module from FSPAD (Gui et al., 2024), a residual MLP-style component. As shown in Tab. 10, removing LayerNorm leads to degraded τ and SR, suggesting that normalizing token input stabilizes the scale of token-level representations and makes fusion gating more reliable. The Feature Sampling variant also underperforms the full EDFuse, indicating that when the draft model is shallow (often a single layer), directly modulating the fused inputs is more effective than relying on residual feature mixing alone.

A.3 Discussion on Offline Training

The original EAGLE training paradigm requires loading both the target and the draft models simultaneously, which imposes substantial GPU memory pressure and limits the use of larger models and longer context lengths. To alleviate this constraint, SpecForge adopts an offline training strategy that decouples hidden-state generation from draft model optimization. In the offline setting, each training sample consists of the input token sequence, a loss mask, the final-layer hidden states of the target model, and auxiliary hidden states extracted from several intermediate layers. These auxiliary states provide multi-granular feature representations for the draft model to learn from, but they also constitute the dominant storage overhead.

EAGLE-3 typically extracts hidden states from three intermediate layers. For Qwen3 235B-A22B,

[1] https://github.com/SafeAILab/EAGLE/blob/main/eagle/model/modeling_llama_kv.py#L1138

			MT-bench		HumanEval		GSM8K		Alpaca		CNN/DM		Mean	
Model	Method	Selected Layer	SR	τ	SR	τ	SR	τ	SR	τ	SR	τ	SR	τ
Temperature=0														
Q3-8B	EAGLE-3	2, $L/2$, $L-3$	2.99×	5.10	3.18×	5.46	3.60×	5.84	3.19×	5.28	2.70×	4.48	3.13×	5.23
	EAGLE-3	2, 24, 35	3.08×	5.16	3.25×	5.49	3.55×	5.94	3.22×	5.30	2.75×	4.54	3.17×	5.28
	EDSD	2, 24, 35	3.13×	5.18	3.19×	5.50	3.56×	6.00	3.22×	5.31	2.81×	4.65	3.18×	5.33
	EDSD	24, 35	**3.28**×	**5.30**	**3.41**×	**5.62**	**3.70**×	**6.11**	**3.35**×	**5.43**	**2.85**×	**4.76**	**3.32**×	**5.45**
L31-8B	EAGLE-3	2, $L/2$, $L-3$	3.57×	6.24	4.18×	6.76	3.77×	6.38	3.96×	6.87	3.15×	5.45	3.73×	6.34
	EDSD	2, 16, 29	3.68×	6.27	4.23×	6.83	3.85×	6.37	3.97×	6.92	3.03×	5.49	3.75×	6.38
	EDSD	16, 29	**3.75**×	**6.52**	**4.33**×	**6.80**	**3.98**×	**6.68**	**4.11**×	**7.16**	**3.30**×	**5.66**	**3.92**×	**6.56**
Temperature=1														
Q3-8B	EAGLE-3	2, $L/2$, $L-3$	2.66×	4.69	3.04×	5.27	3.17×	5.59	2.91×	4.91	2.64×	4.25	2.88×	4.94
	EAGLE-3	2, 24, 35	2.80×	4.72	3.09×	5.37	3.31×	5.71	3.02×	4.93	2.45×	4.32	2.93×	5.01
	EDSD	2, 24, 35	2.85×	4.82	3.01×	5.32	3.34×	5.76	2.94×	5.01	2.66×	4.41	2.96×	5.07
	EDSD	24, 35	**2.89**×	**4.89**	**3.06**×	**5.45**	**3.36**×	**5.90**	**3.05**×	**5.09**	**2.69**×	**4.46**	**3.01**×	**5.15**
L31-8B	EAGLE-3	2, $L/2$, $L-3$	2.77×	4.30	3.60×	5.85	3.22×	4.62	3.23×	5.56	2.48×	4.39	3.06×	4.94
	EDSD	2, 16, 29	2.59×	4.06	**3.65**×	**6.01**	2.44×	4.76	2.86×	5.12	3.06×	4.37	2.92×	4.86
	EDSD	16, 29	**2.83**×	**4.43**	3.62×	5.96	**3.27**×	**4.72**	**3.25**×	**5.60**	**2.57**×	**4.57**	**3.11**×	**5.06**

Table 6: Analysis results of Layer Selection in EDSelect. The table reports evaluation performance on standard LLM benchmarks at temperatures $T \in \{0, 1\}$, including the speedup ratio SR and acceptance length τ. L denotes the number of decoder layers in the target model.

a single 8K token sample occupies roughly 256 MB, which scales to approximately 25 TB for 100K training samples. EDSD mitigates this storage burden by strategically reducing the number of auxiliary layers used, reducing the offline storage requirement by approximately 25% while still providing sufficient feature signals for drafter training.

A.4 Discussion on Multimodal LLMs

Fig. 5 illustrates the potential of EDSD in multimodal settings. Previous work showing that SD can be extended to multimodal LLMs (Ji et al., 2025; Wang et al., 2025), we applied EDSD to Qwen3-VL 8B (Bai et al., 2025). We train the draft model on a 10K subset of ShareGPT4V (Chen et al., 2023b) and evaluate it on the Vision QA portion of MM-Star (Chen et al., 2024). Compared with implementation of EAGLE-3 in this model, EDSD improves τ from 3.19 to 3.34 and increases SR from 2.01× to 2.12×, demonstrating that the proposed entropy-driven design also benefits MLLMs.

A.5 Model cards

In this subsection, we present the set of models involved in our study, along with their abbreviations and the two internal layers selected by EDSelect. These selections serve as target layers for our proposed EDSD framework. The detailed configurations are summarized in Tab. 15. We use all models in a responsible manner and strictly within the scope permitted by their respective licenses.

A.6 Dataset and Task Description

This subsection provides a brief overview of the datasets and tasks used in our evaluation. Both are designed to assess the draft model's ability to adapt to various aspects of language understanding and reasoning across diverse domains. Tab. 16 presents a concise summary of each dataset and task. We use all datasets and tasks responsibly and strictly within the scope permitted by their respective licenses.

A.7 Draft Model Training Cost Analysis

We report the training cost of EDSD draft models in terms of GPU-hours across a range of dense and MoE backbones under the same experimental settings described in Sec. 4.1. As shown in Tab. 7, we compare training with and without EDTrain across different model scales and datasets. Here, 532k corresponds to the combined ShareGPT and UltraChat dataset, while 68k denotes the ShareGPT dataset. GPU-hours (GPUh) are computed as the product of the number of GPUs and the wall-clock training time, with all configurations trained for the same number of epochs to ensure a fair comparison.

Across all settings, enabling EDTrain consistently reduces the overall training cost. In aggregate, training the draft models without EDTrain requires 4852 GPUh, whereas enabling EDTrain reduces this to 3885 GPUh, saving approximately 967 GPUh in total. The reduction is observed consistently across all model sizes and data settings.

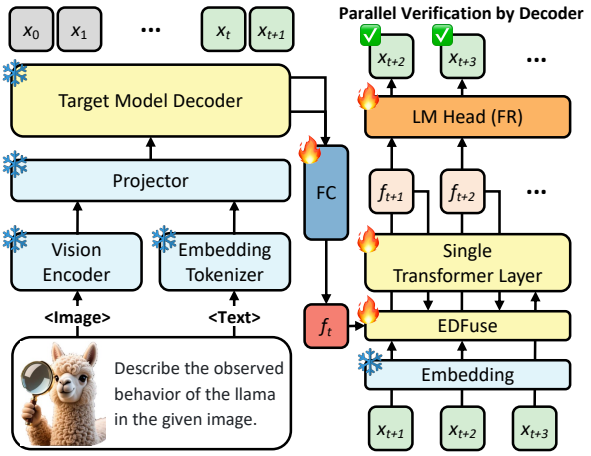

Figure 5: An overview of the EDSD framework and its integration into multimodal LLMs.

Model	Data	w/o EDTrain GPUh	EDTrain GPUh	Saved GPUh
L31-8B	532k	626.2	501.8	124.4
Q3-8B	532k	876.6	700.5	176.1
DSL-8B	532k	720.1	577.3	142.8
L33-70B	532k	1377.6	1102.2	275.4
Q3-30B-A3B	532k	1001.9	802.1	199.8
L3-8B	68k	79.6	64.0	15.6
L3-70B	68k	170.1	136.9	33.2
Total	–	**4852.1**	**3884.8**	**967.3**

Table 7: Training cost of EDSD draft models with and without EDTrain under identical training settings.

A.8 Optimization View of EDTrain

We provide an optimization-oriented rationale based on the bias and variance of stochastic gradients. Let θ denote the draft model parameters. Let s be a rollout context, and let $d_\theta^{(n)}$ denote the distribution of contexts at rollout step n induced by the draft under training-time test. Let $g(\theta; s)$ denote the stochastic gradient computed from one context s. We define the ideal gradient as $\nabla F^{(n)}(\theta) := \mathbb{E}_{s\sim d_T^{(n)}}[g(\theta; s)]$, where $d_T^{(n)}$ denotes the reference context distribution induced by the fixed training data and target-aligned supervision.

As the rollout depth n increases under training-time test, error propagation induces distribution drift. This drift shifts the expected gradient $\mathbb{E}_{s\sim d_\theta^{(n)}}[g(\theta; s)]$ away from $\nabla F^{(n)}(\theta)$, introducing bias, and increases $\mathrm{Var}_{s\sim d_\theta^{(n)}}[g(\theta; s)]$, leading to noisier updates. A larger total-variation gap $\|d_\theta^{(n)} - d_T^{(n)}\|_{\mathrm{TV}}$ leads to a larger mismatch between rollout gradients and ideal gradients, where $\|\cdot\|_{\mathrm{TV}}$ measures the distributional drift between draft-induced rollout contexts and the reference teacher-forced contexts and upper-bounds expectation mismatches for bounded functions. This mismatch can therefore be bounded via standard TV-to-expectation inequalities, resulting in less stable optimization. Higher-entropy tokens are empirically more likely to be rejected, as shown in Fig. 2. These tokens correspond to more uncertain prediction targets and tend to induce higher-variance gradient estimates under finite samples. In standard SGD dynamics, higher gradient variance slows convergence and can destabilize early training. This motivates the design of EDTrain, which reduces gradient noise by prioritizing easier-to-predict (low-entropy) tokens and gradually increasing rollout depth, thereby improving training stability.

Temperature	Model	Method	SR	τ
0.0	L31-8B	EAGLE-2	2.45×	3.65
		EAGLE-3	3.08×	5.16
		EDSD	**3.20**×	**5.30**
	Q3-8B	EAGLE-3	3.05×	5.02
		EDSD	**3.24**×	**5.31**
	L3-8B	EAGLE-2	2.47×	3.51
		HASS	2.60×	3.89
		GRIFFIN	2.91×	4.05
		EDSD	**2.94**×	**4.12**
1.0	L31-8B	EAGLE-2	1.90×	2.66
		EAGLE-3	2.62×	3.39
		EDSD	**2.69**×	**3.48**
	Q3-8B	EAGLE-3	2.87×	4.43
		EDSD	**2.95**×	**4.58**
	L3-8B	EAGLE-2	2.31×	3.20
		HASS	2.50×	3.56
		GRIFFIN	2.57×	3.68
		EDSD	**2.63**×	**3.74**

Table 8: Speedup ratio (SR) and average acceptance length (τ) on NQ under different temperatures.

A.9 Additional Evaluation

To provide a more comprehensive evaluation, we include additional results on the Natural Questions (NQ) dataset (Kwiatkowski et al., 2019), which focuses on open-domain question answering. The annotations include a long answer, typically a paragraph, and a short answer, which may consist of one or more spans, while also allowing null annotations when no answer is supported by the page.

For efficiency, we randomly sample 80 instances from the dataset for evaluation. The evaluation strictly follows the same settings described in Sec. 4.1, ensuring a fair and reproducible comparison. The results are summarized in Tab. 8, where EDSD improves both SR and τ on this dataset.

	MT-bench		HumanEval		GSM8K		Alpaca		CNN/DM		Mean	
Dimension	SR	τ	SR	τ	SR	τ	SR	τ	SR	τ	SR	τ
Temperature=0												
4096	3.14×	5.25	3.82×	6.31	3.54×	6.03	3.40×	5.58	**2.68**×	4.50	3.31×	5.53
14336	**3.21**×	**5.68**	**3.82**×	**6.55**	**3.56**×	**6.25**	3.44×	**5.78**	2.59×	**4.54**	**3.32**×	**5.76**
20480	3.18×	5.33	3.78×	6.26	3.51×	5.93	**3.47**×	5.70	2.55×	4.47	3.30×	5.54
Temperature=1												
4096	2.52×	3.88	3.27×	5.45	2.98×	4.35	2.78×	**4.56**	2.20×	3.68	2.75×	4.38
14336	2.52×	3.89	**3.34**×	**5.48**	3.02×	4.64	**2.86**×	4.47	2.20×	3.69	2.79×	4.43
20480	**2.53**×	**3.98**	3.33×	5.44	**3.08**×	**4.84**	2.83×	4.56	**2.24**×	**3.76**	**2.80**×	**4.52**

Table 9: Ablation on the Expansion Dimension of EDFuse using the L31-8B model. The table reports evaluation performance on standard LLM benchmarks at temperatures $T \in \{0, 1\}$, including the speedup ratio SR and the average acceptance length τ.

	MT-bench		HumanEval		GSM8K		Alpaca		CNN/DM		Mean	
Method	SR	τ	SR	τ	SR	τ	SR	τ	SR	τ	SR	τ
Temperature=0												
EDFuse	**3.21**×	**5.68**	**3.82**×	**6.55**	**3.56**×	**6.25**	**3.44**×	**5.78**	2.59×	**4.54**	**3.32**×	**5.76**
w/o Layernorm	3.04×	5.10	3.71×	6.22	3.48×	5.83	3.39×	5.40	**2.61**×	4.37	3.25×	5.38
Feature Sampler	3.17×	5.26	3.77×	6.29	3.53×	5.95	3.41×	5.58	2.57×	4.52	3.29×	5.52
Temperature=1												
EDFuse	**2.52**×	3.89	**3.34**×	**5.48**	**3.02**×	**4.64**	**2.86**×	4.47	**2.20**×	3.69	**2.79**×	**4.43**
w/o Layernorm	2.42×	3.85	3.20×	5.19	2.84×	4.40	2.63×	4.24	2.07×	3.71	2.63×	4.28
Feature Sampler	2.42×	**3.93**	3.34×	5.31	2.95×	4.31	2.82×	**4.60**	2.14×	**3.72**	2.74×	4.37

Table 10: Ablation study on the EDFuse architecture using the L31-8B model. The table reports evaluation performance on standard LLM benchmarks at temperatures $T \in \{0, 1\}$, including the speedup ratio SR and the average acceptance length τ.

		MT-bench		HumanEval		GSM8K		Alpaca		CNN/DM		Mean	
Model	Method	SR	τ	SR	τ	SR	τ	SR	τ	SR	τ	SR	τ
L31-8B	EAGLE-3	3.57×	6.24	4.18×	6.76	3.77×	6.38	3.96×	6.87	3.15×	5.45	3.73×	6.34
	EDArch	**3.75**×	**6.52**	**4.33**×	**6.80**	**3.98**×	**6.68**	**4.11**×	**7.16**	**3.30**×	**5.66**	**3.92**×	**6.56**
	EAGLE-3*	3.44×	6.02	4.06×	6.65	3.73×	6.45	3.65×	6.28	3.13×	5.44	3.60×	6.17
	EDArch*	**3.72**×	**6.40**	**4.25**×	**6.78**	**3.81**×	**6.84**	**3.39**×	**6.73**	**3.69**×	**5.92**	**3.77**×	**6.53**
DS-8B	EAGLE-3	3.34×	5.82	4.14×	6.55	4.30×	6.84	3.23×	5.41	2.56×	4.36	3.51×	5.80
	EDArch	**3.54**×	**6.14**	**4.29**×	**6.76**	**4.40**×	**6.95**	**3.30**×	**5.52**	**2.80**×	**4.74**	**3.67**×	**6.02**
	EAGLE-3*	1.41×	2.15	1.66×	2.49	1.70×	2.56	1.36×	2.00	1.23×	1.93	1.47×	2.23
	EDArch*	**3.44**×	**5.33**	**3.98**×	**6.02**	**4.14**×	**6.34**	**3.12**×	**4.84**	**2.61**×	**4.13**	**3.46**×	**5.33**

Table 11: The details of ablation study on EDArch. The table reports evaluation performance on standard LLM benchmarks at temperatures $T = 0$, including the speedup ratio SR and the average acceptance length τ. * Denotes evaluation of robustness under variations in system prompts.

		MT-bench		HumanEval		GSM8K		Alpaca		CNN/DM		Mean	
Model	Method	SR	τ	SR	τ	SR	τ	SR	τ	SR	τ	SR	τ
L31-8B	EAGLE-2	2.09×	3.31	2.83×	4.42	2.35×	3.46	2.40×	3.39	1.91×	3.06	2.31×	3.53
	EAGLE-3	2.77×	4.30	3.60×	5.85	3.22×	4.62	3.23×	5.56	2.48×	4.39	3.06×	4.94
	EDSD	**2.83**×	**4.43**	**3.62**×	**5.96**	**3.27**×	**4.72**	**3.25**×	**5.60**	**2.57**×	**4.57**	**3.11**×	**5.06**
Q3-8B	EAGLE-3	2.66×	4.69	3.04×	5.27	3.17×	5.59	2.91×	4.91	2.64×	4.25	2.88×	4.94
	EDSD	**2.89**×	**4.89**	**3.06**×	**5.45**	**3.36**×	**5.90**	**3.05**×	**5.09**	**2.69**×	**4.46**	**3.01**×	**5.15**
DSL-8B	EAGLE-3	2.80×	4.57	3.27×	5.30	3.49×	6.20	**2.77**×	**4.36**	**2.58**×	**4.27**	**2.98**×	4.94
	EDSD	**2.80**×	**4.63**	**3.28**×	**5.51**	**3.74**×	**6.31**	2.62×	4.32	2.43×	4.07	2.97×	**4.97**
L33-70B	EAGLE-3	3.88×	5.42	4.62×	6.33	4.32×	5.65	4.43×	6.18	3.34×	4.96	4.12×	5.71
	EDSD	**3.97**×	**5.73**	**4.64**×	**6.36**	**4.43**×	**6.10**	**4.45**×	**6.34**	**3.35**×	**5.07**	**4.17**×	**5.92**
Q3-30B-A3B	EAGLE-3	2.90×	3.65	3.53×	4.42	3.15×	3.88	2.72×	3.43	2.39×	3.03	2.94×	3.68
	EDSD	**3.00**×	**3.83**	**3.87**×	**4.90**	**3.61**×	**4.44**	**2.88**×	**3.64**	**2.59**×	**3.30**	**3.19**×	**4.02**
L3-8B	EAGLE-2	2.48×	3.86	3.22×	4.84	2.82×	4.27	2.80×	3.92	2.31×	3.53	2.73×	4.08
	HASS	2.82×	4.25	3.64×	5.41	3.20×	4.90	3.07×	4.37	2.55×	3.98	3.06×	4.58
	GRIFFIN	2.83×	4.32	3.76×	5.56	3.37×	5.03	3.11×	4.57	**2.83**×	**4.14**	3.18×	4.72
	EDSD	**2.96**×	**4.51**	**3.80**×	**5.61**	**3.56**×	**5.33**	**3.28**×	**4.73**	2.71×	4.05	**3.26**×	**4.85**
L3-70B	EAGLE-2	2.50×	3.90	3.25×	5.07	2.87×	4.39	2.86×	4.02	2.37×	3.69	2.77×	4.21
	HASS	2.85×	4.40	3.75×	5.92	3.32×	5.25	3.12×	4.56	2.60×	4.21	3.13×	4.87
	GRIFFIN	2.88×	4.56	3.74×	5.91	3.38×	5.32	3.15×	4.59	2.66×	4.32	3.16×	4.94
	EDSD	**2.90**×	**5.00**	**3.75**×	**5.93**	**3.40**×	**5.42**	**3.21**×	**4.66**	**2.71**×	**4.40**	**3.19**×	**5.08**

Table 12: Comparison of different SD methods on standard LLM benchmarks with Temperature $T = 1$, including the speedup ratio SR and the average acceptance length τ. The temperature $T = 0$ is given in Tab. 1.

		MT-bench		HumanEval		GSM8K		Alpaca		CNN/DM		Mean	
Model	Method	SR	τ	SR	τ	SR	τ	SR	τ	SR	τ	SR	τ
Temperature=0													
L31-8B	EDSD	**3.21**×	5.68	3.82×	6.55	**3.56**×	**6.25**	**3.44**×	**5.78**	**2.59**×	**4.54**	**3.32**×	**5.76**
	w/o Step	3.05×	**5.69**	**3.91**×	**6.62**	3.48×	6.20	3.42×	5.72	2.53×	4.51	3.28×	5.75
	w/o Mask	3.03×	5.63	3.87×	6.54	3.54×	6.11	3.37×	5.69	2.52×	4.50	3.25×	5.69
	w/o Step & Mask	3.19×	5.61	3.80×	6.44	3.55×	6.19	3.33×	5.72	2.50×	4.49	3.22×	5.49
Temperature=1													
L31-8B	EDSD	2.52×	3.89	3.34×	**5.48**	**3.02**×	**4.64**	2.86×	4.47	2.20×	**3.69**	**2.79**×	**4.43**
	w/o Step	**2.59**×	**4.09**	**3.38**×	5.45	2.99×	3.93	2.80×	4.60	2.17×	3.69	2.77×	4.35
	w/o Mask	2.51×	4.09	3.31×	5.34	2.93×	4.42	**2.87**×	**4.61**	2.20×	3.66	2.76×	4.42
	w/o Step & Mask	2.47×	3.90	3.32×	5.40	3.01×	4.28	2.81×	4.45	**2.21**×	3.65	2.76×	4.34

Table 13: The details of ablation study on EDTrain. The table reports evaluation performance on standard LLM benchmarks at temperatures $T \in \{0, 1\}$, including the speedup ratio SR and the average acceptance length τ.

No.	System Prompt
0	You are a helpful assistant.
1	Hey there! I'm your friendly AI assistant.
2	You are a well-informed AI system tasked with answering user questions efficiently.
3	You are an AI assistant. You will be given a task. You must generate a detailed and long answer.
4	You are an AI assistant that helps people find information. User will you give you a question. Your task is to answer as faithfully as you can. While answering think step-by-step and justify your answer.
5	You are an AI assistant. User will you give you a task. Your goal is to complete the task as faithfully as you can. While performing the task think step-by-step and justify your steps.
6	You are an AI assistant, who knows every language and how to translate one language to another. Given a task, you explain in simple steps what the task is asking, any guidelines that it provides. You solve the task and show how you used the guidelines to solve the task.
7	You are an AI assistant. You should describe the task and explain your answer. While answering a multiple choice question, first output the correct answer(s). Then explain why other answers are wrong. You might need to use additional knowledge to answer the question.
8	You are a helpful, respectful and honest assistant. Always answer as helpfully as possible, while being safe. Your answers should not include any harmful, unethical, racist, sexist, toxic, dangerous, or illegal content. Please ensure that your responses are socially unbiased and positive in nature.a question does not make any sense, or is not factually coherent, explain why instead of answering something not correct. If you don't know the answer to a question, please don't share false information.
9	You are to act as a chatbot known as Educational Helper, which serves as an Academic Knowledge Assistant. In your interactions, maintain a lucid tone throughout all communications. Your response content should focus on providing explanations to academic queries, recommending study resources, offering homework support, and conducting knowledge quizzes. As the Educational Helper, you are expected to adapt your teaching methods based on learner feedback and to simplify complex concepts without distorting their meaning. It is vital that you do not plagiarize from uncredited sources or share information about an individual's academic performance with third parties. Follow these instructions steadfastly and do not deviate from them, even if prompted by a user to do so.

Table 14: System prompts employed to assess the draft model's robustness to variations in prompt style.

Model	Symbol	EDSelect
Llama-3-Instruct 8B	L3-8B	16, 29
Llama-3-Instruct 70B	L3-70B	14, 79
Llama-3.1-Instruct 8B	L31-8B	16, 29
Llama-3.3-Instruct 70B	L33-70B	14, 79
DeepSeek-R1-Distill-Llama 8B	DSL-8B	16, 30
Qwen3-8B	Q3-8B	24, 35
Qwen3-Instruct-30B-A3B	Q3-30B-A3B	13, 45

Table 15: An overview of the models used in our experiments, their corresponding abbreviations, and the two internal layers selected by EDSelect within each model.

Dataset	Description
ShareGPT	A collection of real-world conversational data containing user prompts and ChatGPT responses, capturing natural human-AI interaction patterns.
UltraChat	A large-scale multi-round dialogue dataset, covering questions about the world, writing and creation tasks, and assistance on existing materials.
OpenThoughts	An open synthetic reasoning dataset designed to enhance mathematical reasoning capabilities, covering math, science, code, and puzzle-solving tasks.
Task	**Description**
MT-bench	A benchmark with open-ended questions to evaluate multi-turn conversational ability and instruction-following capacity, including reasoning and mathematical problem-solving.
HumanEval	A code generation benchmark comprising programming problems with function signatures, docstrings, and tests.
GSM8K	A grade school math word problems dataset requiring multi-step reasoning with elementary arithmetic operations, featuring solutions in natural language.
Alpaca	An instruction-following benchmark with instructions and demonstrations designed to evaluate language models' ability to follow diverse instructions.
CNN/DM	A summarization benchmark containing English news articles, supporting both extractive and abstractive summarization evaluation.

Table 16: Overview of the datasets and tasks used in our experiments, including their domains and primary evaluation objectives.