RELAY: MOMENTUM DRAFTING FROM HIDDEN TRANSITIONS FOR SPECULATIVE DECODING

Anonymous authors
Paper under double-blind review

ABSTRACT

Speculative decoding accelerates large language model inference by using a lightweight drafter to propose tokens for parallel verification. Recent feature-guided drafters improve draft quality by reusing hidden features from the target forward pass, but these features are typically treated as static endpoints, exposing what the target model has computed without explicitly modeling how its representations evolve toward prediction. This leaves the lightweight drafter to infer the missing transformation, making farther-ahead predictions harder to align with the target model. We instead observe that transitions between target-model stages provide a direct drafting signal, as they capture representation changes rather than merely their resulting states. Based on this, we propose RELAY, a lightweight drafter that performs momentum drafting from hidden transitions. RELAY decomposes the target forward pass into low-, mid-, and top-stage features, converts their stagewise differences into early and late momentum states, and recurrently refines these momenta with an Attention-MLP architecture during draft-time generation. By propagating hidden-transition momentum across draft steps, RELAY uses target-side computation as evolving guidance. Experiments across multiple backbones, benchmarks, and serving settings show consistent efficiency gains, achieving up to 31.3% higher speedup and 30.0% longer average acceptance length over EAGLE-3. Code is available at `https://anonymous.4open.science/r/RELAY-8421/`.

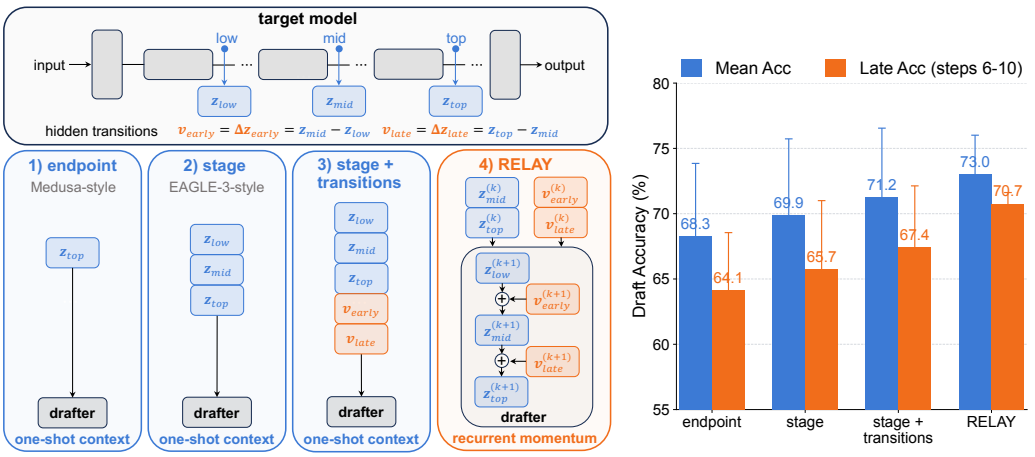
1 INTRODUCTION

Autoregressive generation is inefficient for large language models (Brown et al., 2020; Touvron et al., 2023) because each decoding step advances the sequence by only one token while repeatedly invoking the entire target model. Speculative decoding (Leviathan et al., 2023; Chen et al., 2023; Xia et al., 2024) alleviates this sequential bottleneck by letting a lightweight drafter generate a short draft continuation, which the target model then verifies in parallel (Chen et al., 2024) without changing the target model distribution (Leviathan et al., 2023; Chen et al., 2023). This makes the drafter a central component of speculative decoding, as a useful drafter should generate draft tokens that are well aligned with the target model and likely to be accepted (Zhou et al., 2024; Zhang et al., 2025).

An intuitive drafter design is to use a smaller standalone language model, but this introduces a quality-cost trade-off (Leviathan et al., 2023; Chen et al., 2023). A prominent line of lightweight drafters instead reuses features produced during the target forward pass, including auxiliary draft heads (Cai et al., 2024; Ankner et al., 2024) and one-layer Transformer drafters (Li et al., 2024a;b; 2025). Compared with token histories alone, such target-side features expose part of the model's internal computation to the drafter, allowing lightweight drafters to obtain richer guidance without substantially increasing their size.

Although effective, these feature-guided drafters raise a deeper design question: *how should target-side features be used to better align the drafter with the target model's predictions*? Existing methods typically provide these features as static input context for draft prediction (Cai et al., 2024; Li et al., 2025; Zhang et al., 2025). This tells the drafter what the target model has computed, but not how its representations progress toward prediction. The drafter must therefore infer this missing transformation on its own when generating future tokens, which can be challenging for a lightweight module and can accumulate errors over longer draft rollouts.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.



(a) Using target-side features for drafting

(b) Diagnostic preview on ShareGPT dataset

Figure 1: Overview and diagnostic preview of **RELAY**. (a) Different ways of using target-side features for drafting. Endpoint conditioning uses only the top feature, stage conditioning uses coarse depth-stage features, and stage+transitions further exposes hidden transitions. RELAY carries these hidden transitions as recurrent momentum states and composes them through residual updates during draft-time generation. (b) Parameter-matched diagnostic results on the ShareGPT dataset. Mean Acc is averaged over all 10 draft positions, while Late Acc is averaged over positions 6-10. Error bars denote the standard deviation over the draft positions used to compute each metric. The progression from endpoint to stage, stage+transitions, and RELAY highlights the benefit of increasingly dynamic use of target-side computation.

To seek more direct guidance, we rethink how these features emerge inside the target model. In Transformers, hidden features are formed through repeated updates to a shared residual stream (Elhage et al., 2021), and shallow and deep layers tend to encode progressively different information, from local patterns to semantic representations (Tenney et al., 2019; Jawahar et al., 2019; Geva et al., 2021). Thus, target-side features used for drafting should not be treated as isolated endpoints. They are outcomes of hidden transformations across coarse depth stages, and the transitions between these stages indicate how representations are updated toward prediction.

Based on this insight, we propose **RELAY**, a drafter that *performs momentum drafting from hidden transitions for speculative decoding* by relaying target-derived transition momentum from stage to stage and step to step. As illustrated in Figure 1(a), RELAY coarsely decomposes the target forward pass into early and late stages, using the resulting stage features as anchors and their differences as hidden transitions. These transitions provide initial directions of representation change and initialize two recurrent states, namely early and late momentum. At each draft step, the early branch uses attention to predict a context-sensitive early transition increment, preserving the target-derived transition direction while efficiently learning its residual correction. The predicted correction is accumulated into the previous momentum, and the resulting early momentum is added to the current draft state through a residual update to form a mid-level draft state. Starting from this mid-level draft state, the late branch uses an MLP for local refinement, analogously updating the late momentum and composing it into the top-level draft state. These updated draft and momentum states are carried to the next draft step, allowing the drafter to inherit not only where the target representation arrives, but also how it moves toward prediction.

Figure 1(b) provides a diagnostic preview of this design, with the setup detailed in Section 3. This controlled comparison provides three observations. i) Stage conditioning improves over endpoint conditioning, suggesting that coarse depth-stage features expose useful target-side computation beyond the final hidden state. ii) Explicitly providing hidden transitions improves draft accuracy, suggesting that exposing stagewise representation changes helps the lightweight drafter more effectively exploit target-side features. iii) RELAY achieves the best performance, especially on later draft positions, suggesting that recurrent momentum provides a more effective way to exploit hidden transitions than one-shot conditioning.

In summary, our contributions are threefold:

- To the best of our knowledge, we are the first to formulate feature-guided speculative decoding around hidden-transition momentum. This shifts drafter design from using target-side features as static context to modeling how these features evolve across target-model stages, opening a complementary direction for exploiting target internal computation in speculative drafting.
- We propose RELAY, a lightweight drafter that converts stagewise hidden transitions into recurrent momentum states for draft-time generation. Initialized from early and late transitions, these states are carried across draft steps via an asymmetric Attention-MLP architecture, enabling the drafter to use target-side features as evolving guidance rather than one-shot context.
- Comprehensive experiments across multiple backbones, benchmarks, and serving settings demonstrate the effectiveness of RELAY. Compared with EAGLE-3, RELAY achieves up to 31.3% higher speedup and 30.0% longer average acceptance length.

2 Related Work

Speculative Decoding Speculative decoding accelerates autoregressive generation by decoupling token proposal from target-model verification. Pioneering works (Leviathan et al., 2023; Chen et al., 2023) use a smaller standalone drafter to propose candidates, which are verified by the target model in parallel while preserving the target distribution. Follow-up work improves this draft-and-verify paradigm along two broad directions. One direction focuses on verification-side efficiency, where tree-based (Miao et al., 2024; Li et al., 2024b) and serving-aware methods (Chen et al., 2024) organize speculative continuations into efficient verification structures. Optimal-transport formulations (Sun et al., 2023) further study speculative acceptance. The other direction improves the drafter itself (Zhou et al., 2024; Zhang et al., 2025), often by exploiting target-side features. Medusa (Cai et al., 2024) and Hydra (Ankner et al., 2024) attach auxiliary draft heads to target hidden states. EAGLE (Li et al., 2024a) predicts second-to-top target features with a lightweight Transformer drafter, while EAGLE-3 (Li et al., 2025) extends this with multi-layer target feature fusion. More recently, block-diffusion drafters (Chen et al., 2026; Zhang et al., 2026) mark a shift from sequential autoregressive drafting toward parallel block generation. DFlash (Chen et al., 2026) generates an entire draft block in a single forward pass while conditioning on target-side features, and DSpark (Cheng et al., 2026) builds on parallel drafting with lightweight sequential dependency modeling and confidence-scheduled verification to improve long-range acceptance and serving efficiency. Despite these advances, target-side features are still mainly used as static context for draft prediction. RELAY instead models stagewise hidden transitions and propagates them as recurrent momentum states, providing evolving guidance across draft steps.

Depth-wise Representation Evolution Probing and interpretability work show that Transformer representations are not homogeneous across depth. In BERT-style encoders, linguistic information often progresses from surface-level patterns to syntactic structure and then to more semantic or task-oriented information (Tenney et al., 2019; Jawahar et al., 2019; Rogers et al., 2020). Attention and head-level analyses (Clark et al., 2019; Voita et al., 2019) suggest that linguistic relations and contextual dependencies are unevenly distributed across layers and heads. Mechanistic and representation-geometry studies (Elhage et al., 2021; Geva et al., 2021; Ethayarajh, 2019) further show that Transformer states are updated through the residual stream, with feed-forward layers acting as key-value memories and upper-layer embeddings becoming more contextual and anisotropic. Together, these findings motivate viewing target-side features as a depth-wise representation trajectory, where changes between coarse stages provide useful drafting signals.

Dynamical Views of Neural Networks Neural networks have often been interpreted as dynamical systems in which hidden representations evolve through iterative transformations (Weinan et al., 2017). Residual networks motivate discrete state-evolution views and stable deep architectures (Haber & Ruthotto, 2018), while Neural ODEs (Chen et al., 2018) and Neural CDEs (Kidger et al., 2020) extend this view to continuous-depth and input-controlled dynamics. Momentum-based variants further augment this evolution with momentum states, as in Momentum ResNets (Sander et al., 2021) and Heavy Ball Neural ODEs (Xia et al., 2021), which have been shown to effectively stabilize forward propagation and straighten representation trajectories. Inspired by this perspective, RELAY views drafting as an iterative state-evolution process in which hidden transitions serve as momentum states that guide representation updates across draft steps.

3 PRELIMINARIES

Feature-Guided Speculative Decoding For clarity, we formalize the autoregressive feature-guided setting adopted by RELAY. Let p_θ denote the target language model and q_ϕ denote a lightweight drafter. Given a prefix $x_{\le t}$, the drafter proposes a length-K continuation

$$q_\phi(\hat{x}_{t+1:t+K} \mid x_{\le t}) = \prod_{k=1}^{K} q_\phi(\hat{x}_{t+k} \mid x_{\le t}, \hat{x}_{t+1:t+k-1}), \tag{1}$$

where K is the draft length. The target model evaluates these draft tokens in parallel and verifies them left to right. Let $c_{t+k-1} = (x_{\le t}, \hat{x}_{t+1:t+k-1})$ be the verification context. The draft token $\hat{x}_{t+k}$ is accepted with probability

$$\alpha_{t+k} = \min\left(1, \frac{p_\theta(\hat{x}_{t+k} \mid c_{t+k-1})}{q_\phi(\hat{x}_{t+k} \mid c_{t+k-1})}\right). \tag{2}$$

Upon rejection, the remaining draft tokens are discarded and a replacement token is sampled from

$$p_{\text{corr}}(\cdot \mid c_{t+k-1}) = \text{norm}\left([p_\theta(\cdot \mid c_{t+k-1}) - q_\phi(\cdot \mid c_{t+k-1})]_+\right), \tag{3}$$

where $[a]_+ = \max(a, 0)$ and $\text{norm}(\cdot)$ normalizes the positive part. If all K draft tokens are accepted, the next token is sampled from $p_\theta(\cdot \mid c_{t+K})$. This verification rule preserves the target-model distribution (Leviathan et al., 2023; Chen et al., 2023). The accepted draft length $A \in \{0, \dots, K\}$ is the length of the accepted draft prefix, and a larger $\mathbb{E}[A]$ generally improves decoding efficiency for a fixed draft length and verification overhead.

Feature-guided speculative decoding conditions the drafter on target-side features produced during the target forward pass. Let $\mathcal{Z}_{\le t} = \{\mathbf{z}_{i,\ell}\}_{i \le t, \ell \in \mathcal{L}}$ denote hidden features from selected target-model layers over the prefix. Feature-guided drafters can be abstracted as

$$q_\phi\left(\hat{x}_{t+k} \mid c_{t+k-1}, \mathcal{Z}_{\le t}\right), \qquad k = 1, \dots, K. \tag{4}$$

This covers auxiliary-head methods (Cai et al., 2024; Ankner et al., 2024) and lightweight Transformer drafters (Li et al., 2024a; 2025). In these methods, $\mathcal{Z}_{\le t}$ serves as target-side input context that exposes part of the target model's internal computation to the drafter.

Hidden Transitions as Drafting Signals Following EAGLE-3's multi-layer feature usage (Li et al., 2025), we select three low-middle-top stage features from the target forward pass, denoted by $\mathbf{z}_{i,\text{low}}$, $\mathbf{z}_{i,\text{mid}}$, and $\mathbf{z}_{i,\text{top}}$ for each prefix position $i \le t$. Their stagewise differences define the early and late hidden transitions:

$$\mathbf{v}_{i,\text{early}} = \Delta\mathbf{z}_{i,\text{early}} = \mathbf{z}_{i,\text{mid}} - \mathbf{z}_{i,\text{low}}, \qquad \mathbf{v}_{i,\text{late}} = \Delta\mathbf{z}_{i,\text{late}} = \mathbf{z}_{i,\text{top}} - \mathbf{z}_{i,\text{mid}} \quad (i \le t). \tag{5}$$

Figure 1(b) compares parameter-matched diagnostic variants under a controlled setup, with trainable budgets matched by adjusting the drafter MLP width. The target model is LLaMA-3.1-8B-Instruct, and the ShareGPT data[1] are randomly split into 95% for training and 5% for testing. All drafters are trained for 15 epochs, with the training-time test length set to 7. The endpoint, stage, and stage+transitions variants use the same EAGLE-3-style drafter backbone and token-embedding input, differing only in target-side conditioning: $\{\mathbf{z}_{i,\text{top}}\}_{i \le t}$ for endpoint, $\{\mathbf{z}_{i,\text{low}}, \mathbf{z}_{i,\text{mid}}, \mathbf{z}_{i,\text{top}}\}_{i \le t}$ for stage, and $\{\mathbf{z}_{i,\text{low}}, \mathbf{z}_{i,\text{mid}}, \mathbf{z}_{i,\text{top}}, \mathbf{v}_{i,\text{early}}, \mathbf{v}_{i,\text{late}}\}_{i \le t}$ for stage+transitions. RELAY further turns these transitions into recurrent momentum states, as detailed in Section 4. Token accuracy is evaluated over $K = 10$ draft positions, with Mean Acc over all positions, Late Acc over positions 6–10, and error bars denoting the corresponding position-wise standard deviation. This comparison highlights the added value of hidden transitions beyond stage features and the further benefit of propagating them as recurrent momentum.

4 RELAY

Figure 2 illustrates the overall workflow of RELAY. We first extract stage features from the frozen target model and initialize momentum states from their transitions, then perform dual-branch momentum drafting to generate draft tokens, and finally describe the prediction and training procedures.

[1] https://huggingface.co/datasets/anon8231489123/ShareGPT_Vicuna_unfiltered

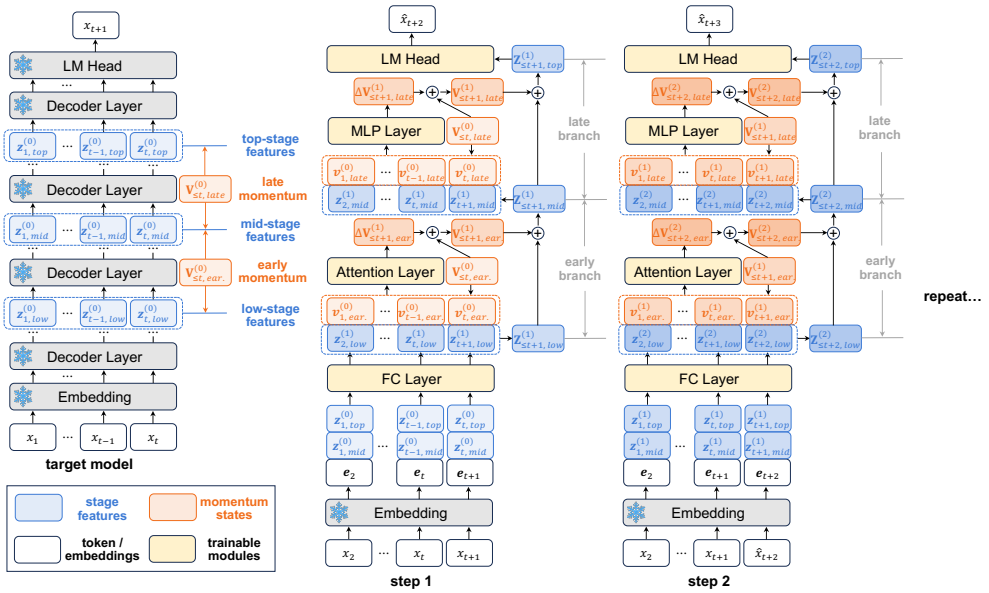


Figure 2: Framework of RELAY. The frozen target model provides low-/mid-/top-stage features, whose differences initialize early and late momentum states for the drafter. RELAY then unrolls an Attention-MLP draft process, where the early and late branches recurrently update momentum states and compose them into stage features for next-token prediction across draft steps.

4.1 State and Momentum Initialization

Let i denote the sequence position and (k) denote the draft rollout step. Given a target-verified prefix $x_{\le t}$, the frozen target model extracts hidden features from the selected low-, mid-, and top-stage layers over the prefix. These features initialize the stage-feature state:

$$\mathbf{Z}^{(0)}_{\le t,\text{low}} = \{\mathbf{z}^{(0)}_{i,\text{low}}\}_{i\le t}, \quad \mathbf{Z}^{(0)}_{\le t,\text{mid}} = \{\mathbf{z}^{(0)}_{i,\text{mid}}\}_{i\le t}, \quad \mathbf{Z}^{(0)}_{\le t,\text{top}} = \{\mathbf{z}^{(0)}_{i,\text{top}}\}_{i\le t}.$$

The stagewise differences initialize the early and late momentum states:

$$\mathbf{V}^{(0)}_{\le t,\text{early}} = \{\mathbf{v}^{(0)}_{i,\text{early}} = \mathbf{z}^{(0)}_{i,\text{mid}} - \mathbf{z}^{(0)}_{i,\text{low}}\}_{i\le t}, \quad \mathbf{V}^{(0)}_{\le t,\text{late}} = \{\mathbf{v}^{(0)}_{i,\text{late}} = \mathbf{z}^{(0)}_{i,\text{top}} - \mathbf{z}^{(0)}_{i,\text{mid}}\}_{i\le t}.$$

Together, the stage-feature states $\mathbf{Z}^{(0)}_{\le t} = \{\mathbf{Z}^{(0)}_{\le t,\text{low}}, \mathbf{Z}^{(0)}_{\le t,\text{mid}}, \mathbf{Z}^{(0)}_{\le t,\text{top}}\}$ and the momentum states $\mathbf{V}^{(0)}_{\le t} = \{\mathbf{V}^{(0)}_{\le t,\text{early}}, \mathbf{V}^{(0)}_{\le t,\text{late}}\}$ serve as the initial recurrent state for draft-time generation.

4.2 Dual-Branch Momentum Drafting

The initialized momentum states encode target-derived transition directions over the verified prefix. During draft rollout, each newly generated token changes the context, so these directions should be adapted rather than reused unchanged. RELAY therefore predicts transition increments. Each increment serves as a residual correction to the previous momentum, reducing the burden of re-estimating a full transition while preserving target-derived directional guidance across draft steps.

At draft step k, RELAY takes tokens up to position $t+k$ and predicts the next draft token $\hat{x}_{t+k+1}$. Let $\mathbf{e}_i$ denote the token embedding produced by the frozen embedding layer of the target model. For each current position $i = 2, \ldots, t+k$, RELAY first injects the current token embedding with the previous mid- and top-stage features:

$$\mathbf{z}^{(k)}_{i,\text{low}} = \mathbf{W}_{\text{inj}} \left[\mathbf{e}_i, \mathbf{z}^{(k-1)}_{i-1,\text{mid}}, \mathbf{z}^{(k-1)}_{i-1,\text{top}}\right]. \tag{6}$$

This shifted projection anchors the current low-stage state to the current token embedding, while using the previous mid- and top-stage features as historical bias.

Early Branch The early branch models the transition from low-stage to mid-stage features. This transition depends on prefix-level interactions, so RELAY uses an attention layer to estimate a context-sensitive early transition increment:

$$\Delta \mathbf{v}_{i,\text{early}}^{(k)} = \left[\text{Attn}\left(\{\left[\text{RMSNorm}\left(\mathbf{z}_{j,\text{low}}^{(k)}\right), \mathbf{v}_{j-1,\text{early}}^{(k-1)}\right]\}_{j=2}^{t+k}\right)\right]_i. \tag{7}$$

The increment predicts how the previous early momentum should be corrected under the current draft context. RELAY then updates the early momentum:

$$\mathbf{v}_{i,\text{early}}^{(k)} = \mathbf{v}_{i-1,\text{early}}^{(k-1)} + \Delta \mathbf{v}_{i,\text{early}}^{(k)}. \tag{8}$$

The mid-stage feature is composed by adding the updated early momentum to the low-stage feature:

$$\mathbf{z}_{i,\text{mid}}^{(k)} = \mathbf{z}_{i,\text{low}}^{(k)} + \mathbf{v}_{i,\text{early}}^{(k)}. \tag{9}$$

Late Branch The late branch models the transition from mid-stage to top-stage features. After the early branch forms context-aware mid-stage features, the late transition mainly performs lightweight nonlinear refinement toward the prediction space. RELAY therefore uses an MLP:

$$\Delta \mathbf{v}_{i,\text{late}}^{(k)} = \text{MLP}\left(\left[\text{RMSNorm}\left(\mathbf{z}_{i,\text{mid}}^{(k)}\right), \mathbf{v}_{i-1,\text{late}}^{(k-1)}\right]\right). \tag{10}$$

The late momentum is updated analogously:

$$\mathbf{v}_{i,\text{late}}^{(k)} = \mathbf{v}_{i-1,\text{late}}^{(k-1)} + \Delta \mathbf{v}_{i,\text{late}}^{(k)}. \tag{11}$$

Finally, RELAY composes the top-stage feature as

$$\mathbf{z}_{i,\text{top}}^{(k)} = \mathbf{z}_{i,\text{mid}}^{(k)} + \mathbf{v}_{i,\text{late}}^{(k)}. \tag{12}$$

The resulting top-stage state is carried to the next draft step and used for next-token prediction.

4.3 Training and Inference

Following EAGLE-3 (Li et al., 2025), RELAY is trained with training-time test, where the drafter is unrolled under the same autoregressive condition as inference. The target model remains frozen. Given a training sequence, we first run the target model to extract the selected low-, mid-, and top-stage features, which initialize the stage-feature and momentum states. RELAY is then unrolled for $K_{\text{train}} = 7$ draft steps. At each step, the predicted token is fed back as the next input, so later draft steps are trained on self-generated rather than ground-truth contexts. This exposes momentum states to rollout-induced context shifts, encouraging self-correction against representation drift.

Let $q_\phi^{(k)}(\cdot)$ denote the token distribution predicted by RELAY at draft step k, and let $p_\theta(\cdot \mid c_{t+k})$ denote the target distribution for the same rollout context. We minimize the multi-step distillation loss using the target distribution as soft labels:

$$\mathcal{L}_{\text{TTT}} = -\sum_{k=1}^{K_{\text{train}}} \lambda_k \sum_{x \in \mathcal{V}} p_\theta(x \mid c_{t+k}) \log q_\phi^{(k)}(x), \tag{13}$$

where $\mathcal{V}$ is the vocabulary and λ_k is the loss weight for draft step k.

During inference, the target model first runs on the verified prefix $x_{\le t}$, produces the next token x_{t+1}, and provides the selected stage features for initialization. RELAY initializes $(\mathbf{Z}_{\le t}^{(0)}, \mathbf{V}_{\le t}^{(0)})$ from these target-side features and starts the draft rollout from the extended context $x_{\le t+1}$. At draft step k, the LM head maps the top-stage feature to the draft-token distribution:

$$q_\phi^{(k)}(\cdot) = \text{Softmax}\left(\text{LMHead}\left(\text{RMSNorm}\left(\mathbf{z}_{t+k,\text{top}}^{(k)}\right)\right)\right). \tag{14}$$

RELAY recurrently generates $\hat{x}_{t+2}, \ldots, \hat{x}_{t+K+1}$, which are then verified by the target model using the standard speculative decoding rule.

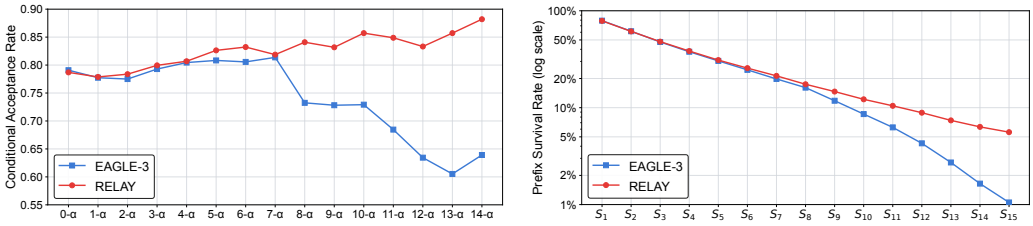
Table 1: Speedup ratios and average acceptance lengths τ of different methods. V represents Vicuna, L31 represents LLaMA-Instruct 3.1, L33 represents LLaMA-Instruct 3.3, and DSL represents DeepSeek-R1-Distill-LLaMA. SpS denotes standard speculative sampling. For non-greedy decoding, we report only methods that preserve lossless speculative verification under standard acceptance-rejection rules.

		MT-bench		**HumanEval**		**GSM8K**		**Alpaca**		**CNN/DM**		**Mean**	
Model	**Method**	*Speedup*	τ	*Speedup*	τ	*Speedup*	τ	*Speedup*	τ	*Speedup*	τ	*Speedup*	τ
Temperature $= 0$													
V 13B	SpS	1.93×	2.27	2.23×	2.57	1.77×	2.01	1.76×	2.03	1.93×	2.33	1.92×	2.24
	PLD	1.58×	1.63	1.85×	1.93	1.68×	1.73	1.16×	1.19	2.42×	2.50	1.74×	1.80
	Medusa	2.07×	2.59	2.50×	2.78	2.23×	2.64	2.08×	2.45	1.71×	2.09	2.12×	2.51
	Lookahead	1.65×	1.69	1.71×	1.75	1.81×	1.90	1.46×	1.51	1.46×	1.50	1.62×	1.67
	Hydra	2.88×	3.65	3.28×	3.87	2.93×	3.66	2.86×	3.53	2.05×	2.81	2.80×	3.50
	EAGLE	3.07×	3.98	3.58×	4.39	3.08×	3.97	3.03×	3.95	2.49×	3.52	3.05×	3.96
	EAGLE-2	4.26×	4.83	4.96×	5.41	4.22×	4.79	4.25×	4.89	3.40×	4.21	4.22×	4.83
	EAGLE-3	4.53×	6.68	5.45×	7.79	4.48×	6.55	4.37×	6.61	4.04×	6.63	4.57×	6.85
	RELAY	**4.86**×	**7.65**	**6.16**×	**9.97**	**4.63**×	**6.99**	**4.61**×	**7.34**	**4.35**×	**7.68**	**4.92**×	**7.92**
L31 8B	EAGLE-2	3.16×	4.05	3.66×	4.71	3.39×	4.24	3.28×	4.12	2.65×	3.45	3.23×	4.11
	DFlash	2.60×	4.11	2.91×	4.25	3.70×	5.36	2.56×	3.78	2.43×	3.53	2.84×	4.21
	EAGLE-3	4.02×	6.20	4.52×	6.82	4.10×	6.34	4.26×	6.83	3.38×	5.39	4.06×	6.32
	RELAY	**4.43**×	**7.13**	**5.03**×	**8.51**	**4.49**×	**6.99**	**4.58**×	**7.78**	**3.75**×	**5.95**	**4.45**×	**7.27**
L33 70B	EAGLE-2	2.83×	3.67	3.12×	4.09	2.83×	3.69	3.03×	3.92	2.44×	3.55	2.85×	3.78
	EAGLE-3	4.10×	5.82	4.50×	6.75	4.71×	6.38	4.66×	6.68	3.37×	5.14	4.36×	6.15
	RELAY	**4.40**×	**6.74**	**5.91**×	**8.46**	**5.05**×	**7.20**	**4.96**×	**7.53**	**3.49**×	**5.40**	**4.77**×	**7.07**
DSL 8B	EAGLE-2	2.92×	3.80	3.42×	4.29	3.40×	4.40	3.01×	3.80	3.53×	3.33	3.26×	3.92
	EAGLE-3	3.98×	5.89	4.41×	6.57	4.77×	7.10	3.84×	5.71	3.34×	4.96	4.07×	6.04
	RELAY	**4.34**×	**6.74**	**4.74**×	**7.33**	**5.59**×	**9.23**	**4.10**×	**6.14**	**3.50**×	**5.32**	**4.45**×	**6.95**
Temperature $= 1$													
V 13B	SpS	1.62×	1.84	1.72×	1.97	1.46×	1.73	1.52×	1.78	1.66×	1.89	1.60×	1.84
	EAGLE	2.32×	3.20	2.65×	3.63	2.57×	3.60	2.45×	3.57	2.23×	3.26	2.44×	3.45
	EAGLE-2	3.80×	4.40	4.22×	4.89	3.77×	4.41	3.78×	4.37	3.25×	3.97	3.76×	4.41
	EAGLE-3	3.87×	5.74	4.55×	6.65	3.81×	5.79	3.72×	5.67	3.56×	5.83	3.90×	5.93
	RELAY	**4.04**×	**6.15**	**4.83**×	**8.05**	**3.98**×	**6.47**	**3.87**×	**6.03**	**3.77**×	**6.42**	**4.10**×	**6.62**
L31 8B	EAGLE-2	2.44×	3.16	3.39×	4.39	2.86×	3.74	2.83×	3.65	2.44×	3.14	2.80×	3.62
	DFlash	2.06×	3.10	3.04×	4.45	2.32×	3.42	2.08×	3.08	**2.94**×	2.01	2.49×	3.21
	EAGLE-3	2.87×	4.33	3.90×	6.13	3.32×	5.01	3.40×	5.68	2.75×	4.50	3.25×	5.13
	RELAY	**3.05**×	**5.17**	**4.34**×	**7.33**	**3.72**×	**6.28**	**3.61**×	**6.05**	2.92×	**4.83**	**3.53**×	**5.93**
L33 70B	EAGLE-2	2.73×	3.51	2.89×	3.81	2.52×	3.36	2.77×	3.73	2.32×	3.27	2.65×	3.54
	EAGLE-3	3.93×	5.62	4.72×	6.43	4.48×	6.14	4.47×	6.45	3.25×	5.02	4.17×	5.93
	RELAY	**4.18**×	**6.27**	**5.41**×	**7.83**	**4.79**×	**6.80**	**4.79**×	**7.26**	**3.40**×	**5.29**	**4.52**×	**6.70**
DSL 8B	EAGLE-2	2.69×	3.41	3.01×	3.82	3.16×	4.05	2.64×	3.29	2.35×	3.13	2.77×	3.54
	EAGLE-3	3.21×	4.93	3.46×	5.28	4.19×	6.53	2.92×	4.52	2.73×	4.27	3.30×	5.11
	RELAY	**3.46**×	**5.52**	**3.64**×	**5.73**	**4.94**×	**8.40**	**3.17**×	**5.03**	**2.91**×	**4.63**	**3.62**×	**5.86**

5 EXPERIMENTS

Models and Benchmarks Experiments are conducted on representative open-source chat and reasoning models, including Vicuna 13B (Chiang et al., 2023), LLaMA-Instruct 3.1 8B, LLaMA-Instruct 3.3 70B (Grattafiori et al., 2024), and DeepSeek-R1-Distill-LLaMA-8B (Guo et al., 2025). Following EAGLE-3 (Li et al., 2025) and Spec-Bench (Xia et al., 2024), each draft model is evaluated without task-specific fine-tuning. The evaluation covers five generation scenarios: MT-bench (Zheng et al., 2023) for multi-turn conversation, HumanEval (Chen et al., 2021) for code generation, GSM8K (Cobbe et al., 2021) for mathematical reasoning, Alpaca (Taori et al., 2023) for instruction following, and CNN/Daily Mail (Nallapati et al., 2016) for summarization.

Metrics Since RELAY keeps the target model frozen and follows the standard speculative decoding verification rule, it preserves the target-model distribution, so generation quality is not separately evaluated. Following EAGLE-3 (Li et al., 2025), acceleration is evaluated using three metrics: speedup ratio over vanilla autoregressive decoding, average acceptance length τ, and acceptance rate n-α. Here, τ denotes the average number of tokens produced per draft-and-verify cycle, while n-α



(a) Conditional Acceptance Rate (b) Prefix Survival Rate

Figure 3: Long-horizon drafting reliability of EAGLE-3 and RELAY on MT-bench using LLaMA-Instruct 3.1 8B as the target model. (a) Conditional acceptance rate across draft positions. (b) Prefix survival rate across accepted prefix lengths, shown on a logarithmic scale.

measures the acceptance rate when the drafter input contains n self-generated states, reflecting the drafter's robustness over multi-step rollouts.

Baselines Vanilla autoregressive decoding is used as the 1.00× speedup baseline. For the main comparison, RELAY is compared with representative speculative methods, including standard speculative sampling (Leviathan et al., 2023; Chen et al., 2023), PLD (Saxena, 2023), Medusa (Cai et al., 2024), Lookahead (Fu et al., 2024), Hydra (Ankner et al., 2024), EAGLE (Li et al., 2024a), EAGLE-2 (Li et al., 2024b), EAGLE-3 (Li et al., 2025), and recent DFlash (Chen et al., 2026).

Implementation Details RELAY is trained with training-time test following the data protocol of EAGLE-3 (Li et al., 2025). ShareGPT and UltraChat-200K (Ding et al., 2023) are used for chat models, and OpenThoughts-114k-Math is additionally used for DeepSeek-R1-Distill-LLaMA 8B. The target model remains frozen throughout training. RELAY is unrolled for $K_{\text{train}} = 7$ draft steps and optimized with AdamW using $(\beta_1, \beta_2) = (0.9, 0.999)$, zero weight decay, and gradient clipping at 0.5. During inference, RELAY uses a confidence-based dynamic draft tree. The tree construction uses only drafter-side probabilities and keeps the speculative verification rule unchanged. Detailed tree-construction and baseline settings are provided in Appendix A.

5.1 Main Results

Table 1 summarizes the end-to-end speedup and average acceptance length across target models, benchmarks, and decoding settings. RELAY consistently outperforms EAGLE-3 across all evaluated configurations, achieving up to 31.3% higher speedup and 30.0% longer average acceptance length. The improvements span diverse generation scenarios, indicating that momentum drafting generalizes beyond a particular task or model family. On LLaMA-Instruct 3.1 8B, RELAY also achieves higher mean performance than DFlash under both decoding settings, providing a comparison with a recent block-diffusion drafting approach.

The gains remain substantial across model scales and task types. On LLaMA-Instruct 3.3 70B, RELAY improves mean speedup and average acceptance length over EAGLE-3 by 9.4% and 15.0%, respectively, under greedy decoding, with the speedup gain reaching 31.3% on HumanEval. This larger end-to-end gain is consistent with more expensive target-model verification, where additional accepted tokens provide greater amortized benefit. On GSM8K with DeepSeek-R1-Distill-LLaMA, RELAY improves speedup by 17.2% and acceptance length by 30.0%, representing one of the strongest gains in our evaluation. This reasoning-intensive setting places greater demands on farther-ahead draft alignment, consistent with the benefits of recurrent hidden-transition guidance over longer rollout horizons. Across backbones, gains are often larger under greedy decoding, while RELAY remains consistently ahead of EAGLE-3 at temperature 1. Although sampling introduces additional uncertainty into draft trajectories and weakens future-token predictability, RELAY still maintains robust acceleration under non-greedy decoding.

5.2 Long-Horizon Drafting Reliability

To examine how drafter reliability evolves over longer rollouts, we compare RELAY and EAGLE-3 on MT-bench using LLaMA-Instruct 3.1 8B as the target model. As shown in Figure 3(a), the

two methods exhibit similar conditional acceptance rates at early draft positions, while RELAY develops a clear advantage as the rollout extends. The gap becomes pronounced beyond the eighth position, with an absolute gain of up to 25.2%, indicating stronger robustness under increasingly self-generated contexts. We further measure prefix survival rate S_n, defined as the fraction of draft prefixes whose first n tokens remain consecutively accepted by the target model. Figure 3(b) shows the advantage accumulates over longer prefixes. At a prefix length of 15, RELAY retains 5.6% of draft prefixes, compared with only 1.1% for EAGLE-3, corresponding to a 5.3× higher survival rate. Together, these results show recurrent hidden-transition guidance becomes increasingly valuable at longer horizons, enabling RELAY to sustain drafter-target alignment as rollout depth grows.

5.3 Deployment in Serving Frameworks

To evaluate RELAY under realistic serving workloads, we integrate it into SGLang (Zheng et al., 2024) and deploy LLaMA-Instruct 3.3 70B as the target model, where the high cost of autoregressive decoding makes speculative acceleration particularly valuable. We measure throughput across increasing concurrency levels and compare RELAY with EAGLE-3 under the same draft-token budget. As shown in Table 2, speculative speedup decreases for both methods as concurrency grows, since batched autoregressive decoding becomes increasingly efficient and leaves less headroom for speculation. Nevertheless, RELAY consistently maintains higher speedup across all three benchmarks. At concurrency 16, EAGLE-3 falls below the autoregressive baseline on HumanEval, GSM8K, and Alpaca, whereas RELAY remains at or above baseline with 1.2×, 1.0×, and 1.2× speedup, respectively.

Table 2: Serving performance across concurrency levels with LLaMA-Instruct 3.3 70B as the target model. Baseline reports autoregressive throughput in tokens per second (TPS), while speculative decoding methods report speedup over baseline. Parentheses denote the draft-token budget, and τ denotes average acceptance length.

	Concurrency					*Avg.* τ
Method	1	2	4	8	16	
HumanEval						
Baseline (TPS)	36	69	131	245	415	–
EAGLE-3 (60)	3.8×	3.1×	2.2×	1.4×	0.9×	5.79
RELAY (60)	**4.6**×	**3.9**×	**2.8**×	**1.8**×	**1.2**×	**7.78**
GSM8K						
Baseline (TPS)	35	68	129	233	361	–
EAGLE-3 (60)	3.2×	2.5×	1.8×	1.1×	0.8×	5.06
RELAY (60)	**3.7**×	**2.9**×	**2.1**×	**1.4**×	**1.0**×	**6.08**
Alpaca						
Baseline (TPS)	36	69	132	235	382	–
EAGLE-3 (60)	3.3×	2.6×	1.9×	1.2×	0.8×	5.10
RELAY (60)	**4.2**×	**3.3**×	**2.4**×	**1.7**×	**1.2**×	**7.04**

The serving gains are consistent with the higher average acceptance length achieved by RELAY. RELAY increases τ from 5.79 to 7.78 on HumanEval, from 5.06 to 6.08 on GSM8K, and from 5.10 to 7.04 on Alpaca, allowing more tokens to be amortized over each target-model verification. This advantage becomes particularly important at higher concurrency, where speculative overhead is harder to offset and insufficient acceptance can eliminate acceleration altogether. Overall, RELAY preserves a larger fraction of its single-request speedup under batched serving, indicating that improved draft acceptance translates into more robust serving efficiency.

6 Conclusion and Discussion

In this paper, we present RELAY, a feature-guided drafter that revisits target-side computation dynamically. Rather than using hidden features as static conditioning signals, RELAY models transitions between target-model stages and propagates them as momentum states during draft-time generation. This shifts speculative drafter design from static feature reuse toward transition-aware state evolution, offering a new way to exploit the target model's internal computation.

Limitations and Future Work RELAY improves the drafter side of speculative decoding, especially by maintaining stronger alignment over longer draft rollouts. However, the benefit of such long-horizon drafting can still be limited by how speculative tokens are organized and verified. Current verification structures may not fully exploit a drafter reliable at later positions. A promising direction is to combine RELAY with more advanced tree construction and adaptive acceptance mechanisms, so that longer reliable drafts can be converted into larger end-to-end speedups.

Reproducibility Statement

We provide a complete description of the RELAY architecture and training procedure in Section 4. Appendix A further documents the training hyperparameters, inference settings, and baseline configurations used in our experiments. The anonymous implementation is available at `https://anonymous.4open.science/r/RELAY-8421/`.

AI Use Statement

Generative AI tools, including ChatGPT and Claude Opus, were used to support specific aspects of this work. They provided feedback on experimental design and assisted with the interpretation of experimental observations. Claude Opus was also used for code implementation and editing, with all resulting code reviewed and tested by the authors. During manuscript preparation, generative AI was used for literature retrieval and summarization as well as language editing. All technical decisions, experimental results, claims, and conclusions were independently reviewed and verified by the authors, who take full responsibility for the final content of this work.

References

Zachary Ankner, Rishab Parthasarathy, Aniruddha Nrusimha, Christopher Rinard, Jonathan Ragan-Kelley, and William Brandon. Hydra: Sequentially-dependent draft heads for medusa decoding. *arXiv preprint arXiv:2402.05109*, 2024.

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

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 *ICML*, pp. 5209–5235, 2024.

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

Jian Chen, Yesheng Liang, and Zhijian Liu. DFlash: Block diffusion for flash speculative decoding. In *ICML*, 2026.

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

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *NeurIPS*, 31, 2018.

Zhuoming Chen, Avner May, Ruslan Svirschevski, Yuhsun Huang, Max Ryabinin, Zhihao Jia, and Beidi Chen. Sequoia: Scalable, robust, and hardware-aware speculative decoding. *arXiv preprint arXiv:2402.12374*, 2024.

Xin Cheng, Xingkai Yu, Chenze Shao, Jiashi Li, Yunfan Xiong, Yi Qian, Jiaqi Zhu, Shirong Ma, Xiaokang Zhang, Jiasheng Ye, et al. Dspark: Confidence-scheduled speculative decoding with semi-autoregressive generation. *arXiv preprint arXiv:2607.05147*, 2026.

Wei-Lin Chiang, Zhuohan Li, Ziqing Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E Gonzalez, et al. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. *See https://vicuna. lmsys. org*, 2(3):6, 2023.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D Manning. What does bert look at? an analysis of bert's attention. In *ACL workshop BlackboxNLP*, pp. 276–286, 2019.

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

Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. Enhancing chat language models by scaling high-quality instructional conversations. In *EMNLP*, pp. 3029–3051, 2023.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, et al. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 1(1):12, 2021.

Kawin Ethayarajh. How contextual are contextualized word representations? comparing the geometry of bert, elmo, and gpt-2 embeddings. In *EMNLP*, pp. 55–65, 2019.

Yichao Fu, Peter Bailis, Ion Stoica, and Hao Zhang. Break the sequential dependency of llm inference using lookahead decoding. In *ICML*, pp. 14060–14079, 2024.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. In *EMNLP*, pp. 5484–5495, 2021.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

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

Eldad Haber and Lars Ruthotto. Stable architectures for deep neural networks. *Inverse Problems*, 34(1):014004, 2018.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. What does bert learn about the structure of language? In *ACL*, pp. 3651–3657, 2019.

Patrick Kidger, James Morrill, James Foster, and Terry Lyons. Neural controlled differential equations for irregular time series. *NeurIPS*, 33:6696–6707, 2020.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *ICML*, pp. 19274–19286. PMLR, 2023.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle: speculative sampling requires rethinking feature uncertainty. In *ICML*, pp. 28935–28948, 2024a.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle-2: Faster inference of language models with dynamic draft trees. In *EMNLP*, pp. 7421–7432, 2024b.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle-3: Scaling up inference acceleration of large language models via training-time test. *NeurIPS*, 38:136737–136756, 2025.

Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Zhengxin Zhang, Rae Ying Yee Wong, Alan Zhu, Lijie Yang, Xiaoxiang Shi, et al. Specinfer: Accelerating large language model serving with tree-based speculative inference and verification. In *ASPLOS*, pp. 932–949, 2024.

Ramesh Nallapati, Bowen Zhou, Cicero Dos Santos, Çağlar Gulçehre, and Bing Xiang. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *SIGNLL*, pp. 280–290, 2016.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. A primer in bertology: What we know about how bert works. *Transactions of the Association for Computational Linguistics*, 8:842–866, 2020.

Michael E Sander, Pierre Ablin, Mathieu Blondel, and Gabriel Peyré. Momentum residual neural networks. In *ICML*, pp. 9276–9287. PMLR, 2021.

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

Ziteng Sun, Ananda Theertha Suresh, Jae Hun Ro, Ahmad Beirami, Himanshu Jain, and Felix Yu. Spectr: Fast speculative decoding via optimal transport. *NeurIPS*, 36:30222–30242, 2023.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. 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, 2023.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. Bert rediscovers the classical nlp pipeline. In *ACL*, pp. 4593–4601, 2019.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *ACL*, pp. 5797–5808, 2019.

E Weinan, Yali Duan, Linghua Kong, and Min Guo. A proposal on machine learning via dynamical systems. *Links*, 2024:08–27, 2017.

Hedi Xia, Vai Suliafu, Hangjie Ji, Tan Nguyen, Andrea Bertozzi, Stanley Osher, and Bao Wang. Heavy ball neural ordinary differential equations. *NeurIPS*, 34:18646–18659, 2021.

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. *Findings of ACL*, pp. 7655–7671, 2024.

Jiebin Zhang, Zhenghan Yu, Song Liu, Eugene J Yu, Zheng Li, Dawei Zhu, Jiangshan Duo, Weimin Xiong, Yifan Song, Guanghua Yu, et al. Dflare: Scaling up draft capacity for block diffusion speculative decoding. *arXiv preprint arXiv:2606.02091*, 2026.

Lefan Zhang, Xiaodan Wang, Yanhua Huang, and Ruiwen Xu. Learning harmonized representations for speculative sampling. In *ICLR*, volume 2025, pp. 35367–35388, 2025.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *NeurIPS*, 36:46595–46623, 2023.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody H Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E Gonzalez, et al. Sglang: Efficient execution of structured language model programs. In *NeurIPS*, volume 37, pp. 62557–62583, 2024.

Yongchao Zhou, Kaifeng Lyu, Ankit Singh Rawat, Aditya Krishna Menon, Afshin Rostamizadeh, Sanjiv Kumar, Jean-François Kagy, and Rishabh Agarwal. Distillspec: Improving speculative decoding via knowledge distillation. In *ICLR*, volume 2024, pp. 32011–32050, 2024.

Table 3: Dynamic draft tree settings for different target backbones and decoding temperatures. The beam width is fixed to 10 for all configurations. D_1 and D_2 denote the depths of the first and second stages, respectively, and ρ denotes the refinement threshold.

	Temperature $= 0$			**Temperature** $= 1$		
Target Backbone	D_1	D_2	ρ	D_1	D_2	ρ
Vicuna 13B	8	6	0.4	7	4	0.5
LLaMA-3.1-8B-Instruct	7	5	0.4	7	5	0.4
LLaMA-3.3-70B-Instruct	12	–	–	12	–	–
DeepSeek-R1-Distill-LLaMA-8B	7	6	0.3	7	6	0.4

A Implementation Details

Training Hyperparameters For all target backbones, we set the RMSNorm epsilon to 10^{-5} and use a batch size of 4. RELAY is optimized with AdamW using $(\beta_1, \beta_2) = (0.9, 0.999)$, zero weight decay, and gradient clipping at 0.5. The learning rate is linearly warmed up to 1.5×10^{-4} over the first 1.5% of training steps and then decayed with a cosine schedule to 1.5×10^{-5}. The maximum sequence length is 2048, and the training-time test length is $K_{\text{train}} = 7$. Losses across draft steps are geometrically weighted as $\lambda_k = 0.8^{k-1}$. The same hyperparameter configuration is used across all target backbones. The environment where we run experiments is:

- Operating system: Linux 5.15.0-94-generic
- CPU information: Intel Xeon Platinum 8378A @ 3.00GHz
- GPU information: NVIDIA A800-SXM4-80GB $\times$ 8

Dynamic Draft Tree Settings During inference, RELAY uses a dynamic draft tree with an optional second-stage refinement constructed from drafter-side probabilities. The beam width is fixed to 10 for all configurations. The first stage performs beam expansion to depth D_1, while the second stage further expands branches whose cumulative probability exceeds a refinement threshold ρ to depth D_2. The tree settings are tuned for each target backbone and decoding temperature, as summarized in Table 3. Tree construction affects only draft organization and leaves the standard speculative verification rule unchanged.

Baseline Implementations Results for standard speculative sampling (Leviathan et al., 2023; Chen et al., 2023), PLD (Saxena, 2023), Medusa (Cai et al., 2024), Lookahead (Fu et al., 2024), Hydra (Ankner et al., 2024), EAGLE (Li et al., 2024a), and EAGLE-2 (Li et al., 2024b) are taken from EAGLE-3 (Li et al., 2025) under the corresponding target-model, benchmark, and decoding settings. To enable hardware-matched comparisons with the strongest recent baselines, we reproduce EAGLE-3 (Li et al., 2025)[2] and DFlash (Chen et al., 2026)[3] on our hardware using their official checkpoints and implementations. For both methods, we use the inference and draft configurations recommended by their original implementations without modification. We compare DFlash on LLaMA-3.1-8B-Instruct, for which its official checkpoint is specifically trained on UltraChat-200K and ShareGPT to align with EAGLE-3 training data.

[2] https://github.com/SafeAILab/EAGLE

[3] https://github.com/z-lab/dflash