# ECHOSPEC: EFFICIENT TARGET-INTEGRATED SPECULATIVE DECODING VIA OUTPUT-PROJECTION ADAPTATION AND POST-TARGET REFINEMENT

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Autoregressive decoding is a major bottleneck in large language model inference due to its sequential and memory-bandwidth-bound nature. Speculative decoding alleviates this bottleneck by drafting multiple future tokens for parallel verification. While recent methods improve draft quality using separate auxiliary transformers, these models introduce target-specific parameters and significant training costs. Furthermore, the auxiliary weights and KV caches increase serving memory, particularly at high concurrency. In this paper, we introduce ECHOSPEC, a target-integrated speculative decoding framework that reuses the frozen target model for drafting with minimal adaptation. At each layer, we apply a learned bias exclusively to the attention output projection at draft positions. This approach is highly parameter-efficient compared to broadly adapting the target model. Nonetheless, drafting in parallel with verification prevents the draft positions from accessing the bonus token. To resolve the resulting information gap, we introduce a lightweight post-target refinement block in ECHOSPEC rather than scaling up target-side adaptation. As a result, ECHOSPEC requires only 2.2–4.5K target-generated training samples, features 27.9–32.1$\times$ fewer trainable parameters than DFlash, and eliminates the need for a separate draft KV cache. Under long-context, high-concurrency workloads, it reduces the active memory footprint and delivers higher output throughput than separate transformer drafters such as DFlash and EAGLE-3.

## 1 INTRODUCTION

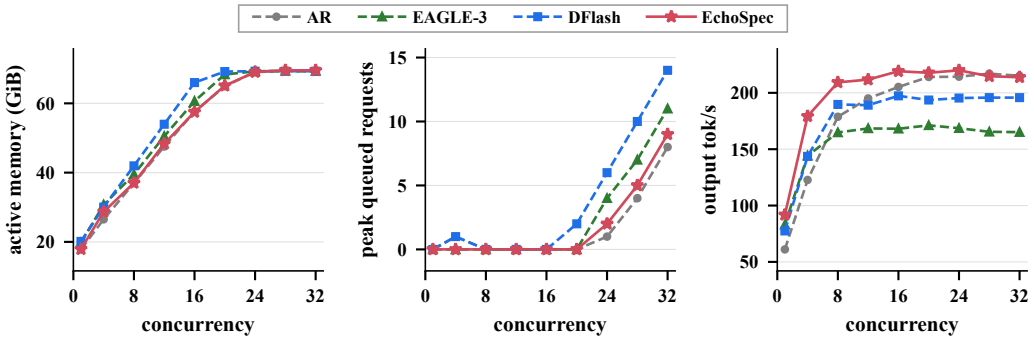
As large language models (LLMs) increasingly generate long reasoning traces (Guo et al., 2025; OpenAI et al., 2026), sequential autoregressive (AR) decoding has become a primary bottleneck in LLM inference. This is because producing each token requires reading all model weights, making each decoding step memory-bandwidth-bound (Shazeer, 2019; Pope et al., 2023). Speculative decoding (SD) mitigates this bottleneck by letting a drafter propose multiple tokens and then having the target verify them in parallel while preserving the target's output distribution (Leviathan et al., 2023; Chen et al., 2023a). Its speedup, however, depends on the drafting cost and the number of tokens accepted per target forward pass. To achieve high acceptance, recent methods such as EAGLE-3 and DFlash employ separate transformer drafters, but this design introduces additional training and serving costs (Li et al., 2025; Chen et al., 2026).

These drafters improve agreement with the target by using the target's hidden states. To learn how to interpret these hidden states for future-token prediction, each drafter requires a substantial number of parameters and a large response training dataset generated by the target. EAGLE-3 uses roughly 532K examples, while DFlash uses 800K for training. Both the hidden-state input and the target-generated data make each drafter strictly tied to its specific target. Therefore, adapting such drafters to a new target requires an additional stage of large-scale data generation and training. These costs recur as the LLM ecosystem evolves and target models are updated frequently (Chen et al., 2023b; Zhao et al., 2026).

In addition, these transformer drafters incur high serving overhead. Production systems generally batch many requests to improve GPU utilization (Kwon et al., 2023; Agrawal et al., 2024). In



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) Active memory (↓) (b) Peak queued requests (↓) (c) Output throughput (↑)

Figure 1: Serving behavior of a separate drafter against the number of concurrent requests on GovReport inputs longer than 8K tokens with Qwen3-8B.

batched serving, a separate transformer drafter introduces its own parameters and KV cache on top of the target's. These added states increase active memory relative to AR as concurrency (the number of batched requests) grows (Figure 1a). This leaves less room for the target KV cache, especially for long-context requests, increasing queued requests (Figure 1b). As queue lengths grow at high concurrency, the drafter's throughput saturates and can eventually fall below standard AR (Figure 1c).

The training and serving costs of separate, target-specific drafters motivate us to reuse the target model itself for drafting. Integrating drafting directly into the target eliminates the need for a separate drafter and its associated KV cache during serving. Furthermore, because the target's existing knowledge is leveraged directly, this approach minimizes the new parameters that must be learned specifically for drafting. To realize this target-integrated design, we append a small number of mask tokens in addition to the standard input tokens. The target model processes these masks as draft positions to predict multiple future tokens. However, this integrated approach introduces a critical architectural challenge: how to adapt the frozen target model to guide these mask positions toward accurate future-token prediction without negating the parameter and memory benefits of the design.

To address this architectural challenge, we introduce ECHOSPEC, a lightweight, target-integrated speculative decoding framework that reuses the frozen target model for drafting with minimal adaptation. Its contributions are summarized as follows:

- **Minimal target adaptation.** To minimize the trainable parameters required to adapt the target's computation, we introduce a parameter-efficient bias adapter called O-bias. Through systematic evaluation, we find that applying this bias exclusively to the attention output projection ($W_O$) maximizes draft acceptance with minimal computational overhead.
- **Post-target refinement.** Because mask positions are processed during the target's forward pass, O-bias cannot access the bonus token, which is only generated after verification. This creates an information gap. To resolve this issue, we propose a lightweight post-target refinement block. By incorporating the newly available bonus token, this block refines the draft representations along the accepted path to further increase draft acceptance.
- **Efficient target-specific adaptation and serving.** ECHOSPEC requires only 2.2–4.5K target-generated training samples, uses 27.9–32.1× fewer trainable parameters than DFlash, and requires no separate draft KV cache. It delivers 7.7% higher average output throughput (tokens/s) than DFlash under high concurrency (16–32 concurrent requests). Furthermore, ECHOSPEC outperforms recent target-integrated baselines such as Apple MTP (Samragh et al., 2025) and PaSS (Monea et al., 2023) under the same parameter and training budget.

## 2 RELATED WORK

**Speculative decoding.** SD accelerates LLM generation by proposing multiple future tokens with a drafter and verifying them in parallel with the target model (Leviathan et al., 2023). Specifically, verification uses rejection sampling to determine the accepted draft prefix and sample an additional token at its boundary, referred to as the *bonus token*. To balance accepted prefix length against

Table 1: Comparison with representative speculative decoding methods. 'Seen' denotes the number of training samples processed, which is computed as training steps times global batch size.

| Method | Train data (K seq.) | Seen (K seq.) | Own KV cache |
|---|---|---|---|
| DFlash (Chen et al., 2026) | 800 | 4800 | O |
| EAGLE-3 (Li et al., 2025) | 532 | 21280 | O |
| Apple MTP (Samragh et al., 2025) | 1000 | 400 | **X** |
| BiTA (Lin et al., 2025) | 150 | 600 | **X** |
| PaSS (Monea et al., 2023) | – | 80 | **X** |
| **ECHOSPEC** | **4.5** | **9** | **X** |

inference overhead, existing methods differ primarily in how they construct the drafter. These approaches include separate auxiliary transformers conditioned on target representations, as well as methods that integrate drafting directly into the target's forward pass. We summarize representative methods in Table 1.

**Target-representation-conditioned transformer drafters.** These methods train a separate auxiliary transformer to predict future tokens using the target's hidden states. For example, EAGLE-3 (Li et al., 2025) combines features from multiple target layers and generates draft tokens autoregressively. In contrast, DFlash (Chen et al., 2026) conditions a block diffusion transformer on target features to predict multiple masked tokens in parallel within a single draft forward pass. However, these separate transformers require their own parameters and associated KV caches, and they incur substantial training costs, as shown in Table 1.

**Parallel prediction inside the target.** These methods draft within the frozen target's forward pass and differ in which components they train. PaSS (Monea et al., 2023) appends learned lookahead embeddings to the token sequence fed into the target model for draft generation. BiTA (Lin et al., 2025) combines learned prefix key-value vectors and mask embeddings with a tree attention mask for parallel drafting. Both methods keep the target's weights frozen and learn additional input vectors for drafting. Recently, Apple MTP (Samragh et al., 2025) extends adaptation to the target's internal computation by applying gated low-rank updates to the target's linear layers at draft positions, together with a two-layer output module conditioned on the previously sampled token.

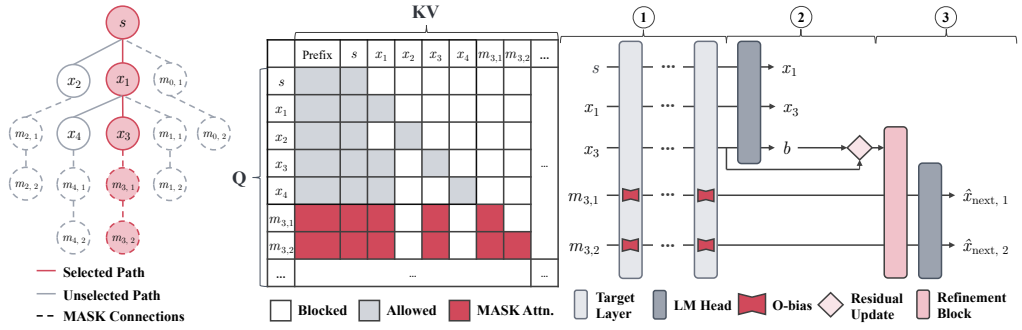
These approaches demonstrate that a frozen target can be guided toward future-token prediction through lightweight learned components. In contrast, our work pursues minimal adaptation of the target's internal computation while retaining competitive SD performance. Under this design principle, we use a bias vector as the adaptation component and further restrict its application to $W_O$, which we identify as the most effective attention site. We additionally employ a lightweight post-target refinement block to compensate for the missing bonus-token information during parallel drafting. Furthermore, we demonstrate that ECHOSPEC is particularly effective at high concurrency with long contexts, a regime not examined by prior target-integrated methods.

# 3 ECHOSPEC

## 3.1 OVERVIEW

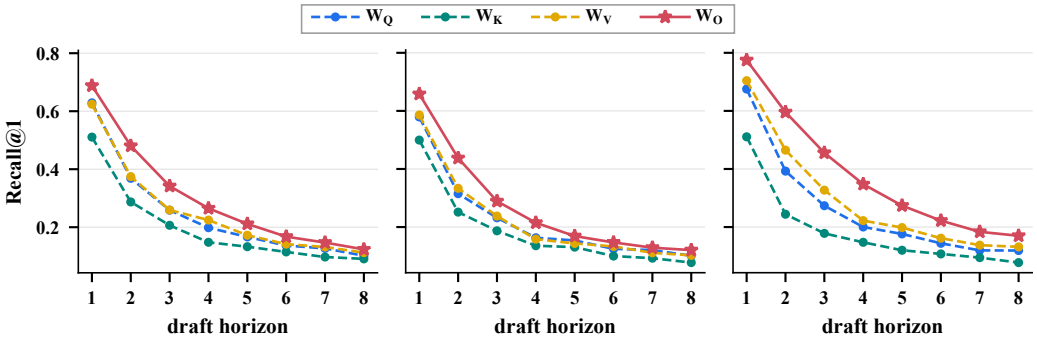
ECHOSPEC uses O-bias to enable drafting within the frozen target and a post-target refinement block to improve the resulting draft representations. Each decoding cycle begins with a draft tree constructed by combining top-ranked candidates across draft slots from the previous cycle (Appendix A). We denote the last token of the verified prefix by $s$, which is also set to $x_0$, and each candidate node by $x_n$ for $n = 1, \ldots, N$, where $N$ is the number of candidate nodes. To generate draft tokens in parallel with token verification, we attach a chain of $M$ masks to each node $x_n$ in the tree, where these mask nodes are denoted as $m_{n,1}, \ldots, m_{n,M}$ (Figure 2a). To preserve the causal dependencies within the tree structure, we use a tree-attention mask (Figure 2b). Each token or mask can attend only to the verified prefix, its ancestors, and itself. In this way, a single target forward pass both verifies the current draft tokens and prepares draft representations for the next cycle.

We carry out each cycle in three steps (Figure 2c). First, the target processes the entire draft tree and its attached masks in a single forward pass, with O-bias applied only at mask positions (Section 3.2).




(a) Draft tree (b) Tree attention matrix (c) One decoding cycle

Figure 2: **ECHOSPEC overview.**




(a) Qwen3-8B (b) Qwen3-4B (c) Llama-3.1-8B-Instruct

Figure 3: Recall@1 at each draft horizon for the four single-site adapters of the same sweep.

Second, verification selects the accepted path from this tree and produces the bonus token $b$. Third, we pass the mask hidden states from the accepted path to the refinement block and the LM head conditioned on $b$ to generate the next draft tokens (Section 3.3).

### 3.2 O-BIAS: MINIMAL TARGET ADAPTATION

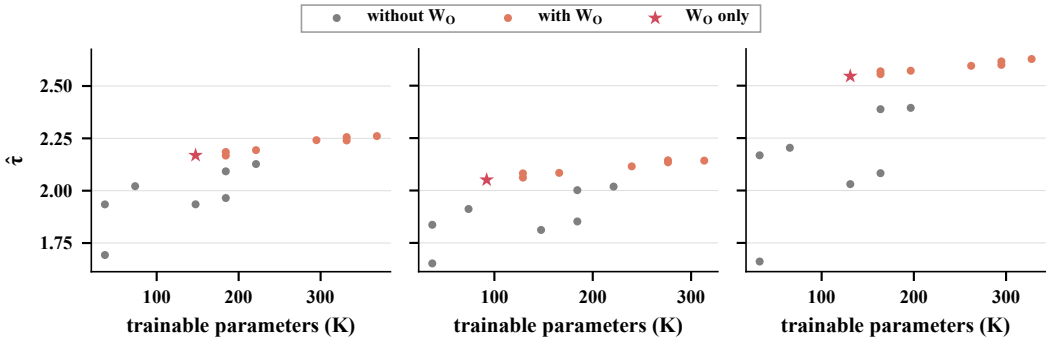
**The bias adapter.** Self-attention directly determines how the target retrieves and uses information from the preceding context. To predict multiple future tokens, we add a trainable component to the attention computation at mask positions. To achieve minimal adaptation, we restrict this trainable component to a simple bias vector. We apply this bias to the weight projections in self-attention, which consist of $W_Q$, $W_K$, $W_V$, and $W_O$. While each layer has its own projection weights, we assign a layer-specific bias to a selected projection. For a chosen projection site $W^{(\ell)} \in \{W_Q^{(\ell)}, W_K^{(\ell)}, W_V^{(\ell)}, W_O^{(\ell)}\}$ in layer $\ell$, the biased output $\widetilde{W}^{(\ell)}(z_r)$ is given by

$$\widetilde{W}^{(\ell)}(z_r) = \begin{cases} W^{(\ell)} z_r + \mathbf{b}^{(\ell)}, & r \in \mathcal{M}, \\ W^{(\ell)} z_r, & \text{otherwise}, \end{cases} \tag{1}$$

where $z_r$ denotes the input to the projection at position $r$, and $\mathcal{M}$ denotes the set of all mask positions. To preserve the target's original verification behavior, we apply this bias exclusively at mask positions ($r \in \mathcal{M}$), ensuring it does not modify the frozen target's standard representations.

**Offline draft quality measurement.** To evaluate different design choices offline, we measure draft quality on a validation set (detailed in Section 3.4). We define $\mathrm{Recall@}k(j)$ as the frequency with which the correct target token appears among the top-$k$ predictions at draft horizon $j$. Assuming independence across draft horizons, we define the following proxy acceptance length to aggregate the top-1 predictions:

$$\hat{\tau} = 1 + \sum_{j=1}^{M} \prod_{j'=1}^{j} \mathrm{Recall@1}(j'). \tag{2}$$




(a) Qwen3-8B (b) Qwen3-4B (c) Llama-3.1-8B-Instruct

Figure 4: Proxy acceptance length $\hat{\tau}$ versus trainable parameter count for every non-empty subset of $\{W_Q, W_K, W_V, W_O\}$ adapted with mask-gated biases.

**$W_O$ is the effective site.** To identify which adaptation site is most effective for future-token prediction, we apply the bias independently to $W_Q$, $W_K$, $W_V$, and $W_O$. To measure how well each site predicts an acceptable token at each draft horizon, we compare Recall@1. As shown in Figure 3, adapting $W_O$ consistently yields a higher Recall@1 than adapting the other projections. This suggests that $W_O$ is the most effective site for inducing multi-token prediction.

We further examine whether this single-site trend holds when interactions among projection sites are considered. For this, we evaluate the proxy acceptance length $\hat{\tau}$ for every non-empty subset of $\{W_Q, W_K, W_V, W_O\}$. Figure 4 shows that adapting $W_O$ alone is the most parameter-efficient choice. While adding other projection sites on top of $W_O$ yields little additional gain in $\hat{\tau}$ despite introducing more trainable parameters, subsets without $W_O$ strictly underperform those containing it. In particular, adapting $W_O$ alone achieves a higher $\hat{\tau}$ than adapting $W_Q$, $W_K$, and $W_V$ combined. Therefore, to keep target adaptation at a minimal level, we apply the bias only to $W_O$ (termed O-bias) in the remainder of our method.

### 3.3 Post-target refinement block

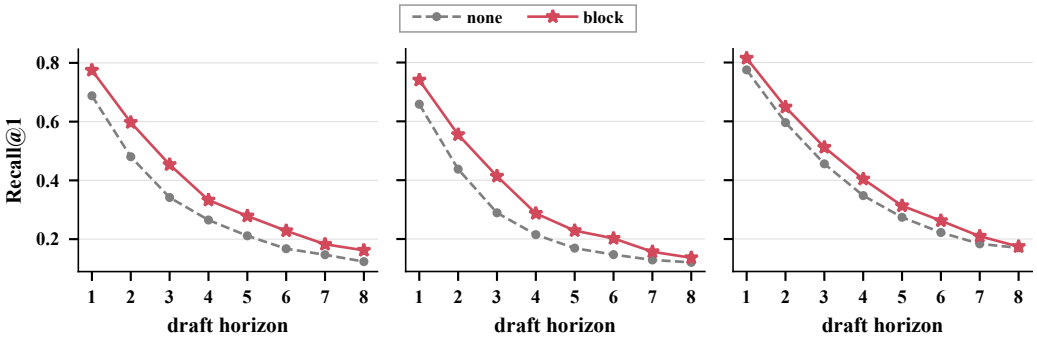
**Parallel drafting leaves a bonus token information gap.** While O-bias substantially improves draft quality, the mask positions may still lack the information needed to distinguish the correct token from other likely candidates. Unlike separate auxiliary drafters that condition their predictions on the bonus token, our approach generates drafts at mask positions in parallel during the target's forward pass. This creates an information gap, because O-bias must compute drafts before the bonus token is known. However, once verification determines the accepted path, the bonus token becomes available. Therefore, we refine the mask states after verification, allowing the draft representations to incorporate this newly revealed token.

**Refinement block on the accepted path.** To implement this post-target refinement, we introduce a single transformer block—the *refinement block*—applied exclusively to the hidden states along the accepted path. Specifically, to condition this block on the newly available bonus token, we project the bonus token embedding $b$ with low-rank projection matrices $A$ and $B$. We define the input sequence to the refinement block as:

$$\tilde{H} = \left[h_{x_i} + BAb,\ h_{m_{i,1}},\ h_{m_{i,2}},\ \ldots,\ h_{m_{i,M}}\right], \tag{3}$$

where $x_i$ denotes the last accepted node, and $h_v$ denotes the final-layer hidden state of the target at node $v$.

Because the refinement block receives only $M+1$ input slots, in contrast to the full context processed by the standard target layer, we can adopt the target layer's architecture while significantly reducing its attention head and MLP inner dimensions. This ensures the block remains lightweight. After the forward pass through the refinement block, the $j$-th output (for $j = 2, \ldots, M+1$) integrates information from slots $1, \ldots, j$ via causal attention. Therefore, we pass the outputs at slots $2, \ldots, M+1$ through the target LM head to obtain the logits for the next cycle's draft tree.




(a) Qwen3-8B (b) Qwen3-4B (c) Llama-3.1-8B-Instruct

Figure 5: Recall@1 at each draft horizon of the refinement block on the bias-only adapter.

We empirically confirm that this refinement block improves draft quality over using O-bias alone (Figure 5). For models like Qwen3-8B and Qwen3-4B, where O-bias alone yields a lower Recall@1 baseline, the refinement block provides consistent gains across all draft horizons. In contrast, Llama-3.1-8B-Instruct achieves a high Recall@1 with O-bias alone, making the additional performance gains from the refinement block more modest.

### 3.4 Training

The target model remains frozen, and the mask embedding is fixed at the average of the embedding vectors in the training set. Thus, the only trainable components are the O-bias and the refinement block. Each mask position is trained to predict the token at its corresponding horizon. We optimize these components using the mean cross-entropy across all training horizons, given by

$$\mathcal{L} = \frac{1}{M_{\text{train}}} \sum_{j=1}^{M_{\text{train}}} \text{CE}\big(p_j, q_j\big), \tag{4}$$

where $p_j$ and $q_j$ denote the token distribution of the frozen target and the final predicted token distribution from the mask at horizon $j$, respectively. We set $M_{\text{train}}{=}8$ during training and choose $M_{\text{infer}}$ independently at serving time. For the refinement block, we set the attention head dimension to $0.5\times$ that of the target and the MLP intermediate dimension to $0.0625\times$ that of the target, while keeping the residual hidden dimension unchanged. We set the bonus token projection rank to 256.

We train on target-generated responses to prompts from the NVIDIA Nemotron Post-Training Dataset V2 (Nathawani et al., 2025) and CodeAlpaca (Chaudhary, 2023). We use 2.2K sequences for Qwen3-4B and Llama-3.1-8B-Instruct, and 4.5K for Qwen3-8B, while additionally holding out 64 samples from each dataset for validation. To reach approximately 9K training steps, we train Qwen3-4B and Llama-3.1-8B-Instruct for 4 epochs and Qwen3-8B for 2 epochs. We use the AdamW optimizer with a learning rate of $1 \times 10^{-3}$, a gradient clipping threshold of 1.0, and a cosine scheduler with a warmup ratio of 0.04.

## 4 Experiments

### 4.1 Setup

**Targets and workload.** We evaluate three frozen target models: Qwen3-8B, Qwen3-4B, and Llama-3.1-8B-Instruct. We use GovReport and MultiNews from LongBench-E (Bai et al., 2024) as workloads, splitting each dataset by input length into a 4K–8K band and an 8K+ band. Each run serves every prompt in the split once at a fixed concurrency and caps each response at 512 output tokens. We sweep concurrency (the number of batched requests) from 1 to 32.

**Baselines.** We compare EchoSpec against AR decoding and four SD methods: EAGLE-3 (Li et al., 2025), DFlash (Chen et al., 2026), Apple MTP (Samragh et al., 2025), and PaSS (Monea et al., 2023). For EAGLE-3, we use the official checkpoints on Llama-3.1-8B-Instruct and the checkpoints released by AngelSlim (Tencent, 2025) on both Qwen3 models. We train Apple MTP

Table 2: Parameter comparison of ECHOSPEC and Apple MTP model

| Configuration | Method | Parameters, M (%) | | Total (M) |
|---|---|---|---|---|
| | | Target adaptation | Post-target refinement | |
| Qwen3-8B | Apple MTP (LoRA rank 8) | 21.82 (66.8) | 10.84 (33.2) | 32.66 |
| | ECHOSPEC | 0.15 (0.5) | 32.51 (99.5) | 32.66 |
| Qwen3-4B | Apple MTP (LoRA rank 8) | 16.52 (86.1) | 2.67 (13.9) | 19.19 |
| | ECHOSPEC | 0.09 (0.5) | 19.09 (99.5) | 19.18 |
| Llama-3.1-8B-Instruct | Apple MTP (LoRA rank 6) | 15.73 (46.0) | 18.48 (54.0) | 34.21 |
| | ECHOSPEC | 0.13 (0.4) | 34.09 (99.6) | 34.22 |

Table 3: Token capacity of KV cache in each target and method.

| Method | Qwen3-8B | | Qwen3-4B | | Llama-3.1-8B-Instruct | |
|---|---|---|---|---|---|---|
| | Tokens | % of AR | Tokens | % of AR | Tokens | % of AR |
| AR | 395,146 | 100.0 | 160,483 | 100.0 | 446,652 | 100.0 |
| DFlash | 333,970 | 84.5 | 134,143 | 83.6 | 371,445 | 83.2 |
| EAGLE-3 | 370,975 | 93.9 | 146,902 | 91.5 | 418,548 | 93.7 |
| ECHOSPEC | 394,563 | 99.9 | 160,127 | 99.8 | 446,076 | 99.9 |

and PaSS using the same training policy as ECHOSPEC, described in Section 3.4. Specifically, for Apple MTP, we configure its trainable parameter count to be comparable to that of ECHOSPEC, as shown in Table 2. We also evaluate learning rates of $1 \times 10^{-3}$ and $1 \times 10^{-4}$, selecting the one that yields the longer proxy acceptance length $\hat{\tau}$.

**Configuration.** For EAGLE-3, we use 7 draft steps with a top-10 configuration and 10 draft tokens, as used in Chen et al. (2026). DFlash uses the block size set by the authors, i.e., 16 on the Qwen3 models and 10 on Llama-3.1-8B-Instruct. We configure Apple MTP and PaSS to perform drafting and verification in parallel, identical to ECHOSPEC. To use a verification window size comparable to EAGLE-3 and DFlash, we set $M_{\text{infer}}$=2 and use a dynamic tree with 5 draft nodes (including the root) for ECHOSPEC, Apple MTP, and PaSS. This yields 15 positions per verification window.

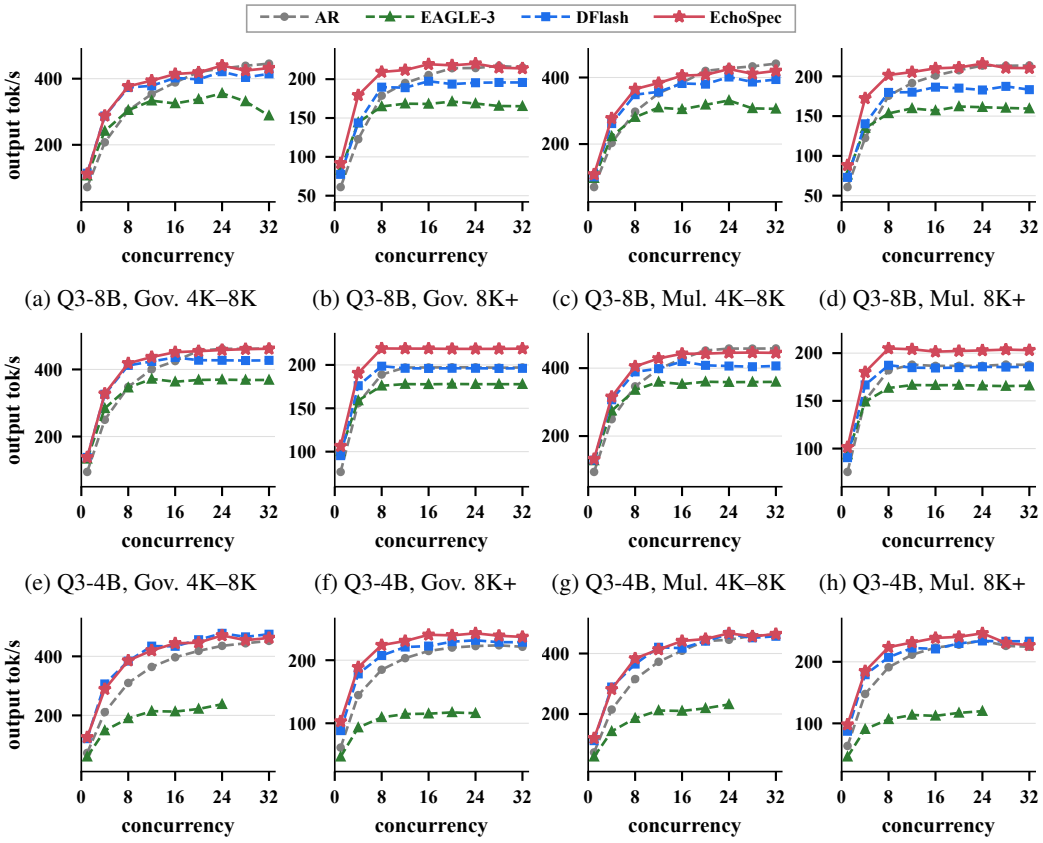
**Serving.** All measurements come from SGLang (Zheng et al., 2024) with no overlap scheduling. We evaluate DFlash and EAGLE-3 using their official SGLang implementations, while we implement the remaining methods, including ECHOSPEC, in SGLang. The two 8B targets run on an A100 (80GB) GPU, and the Qwen3-4B target runs on an A100 (40GB) GPU. We leave 10 GB of device memory free for CUDA graphs and give each method the same memory budget across targets and GPUs. CUDA graph capture exceeds this memory budget for a small number of method-specific high-concurrency settings. We exclude those cases and report them in Appendix B.

**Metrics.** We report aggregate output throughput as the total number of generated output tokens divided by the wall-clock time to serve the full split. We also report the KV-cache capacity in tokens and the average acceptance length $\tau$ in each cycle of SD, including the bonus token.

### 4.2 EXPERIMENTAL RESULTS

**Main evaluation.** We compare the output throughput of ECHOSPEC against DFlash and EAGLE-3 in Figure 6, with additional results at temperature 1 reported in Appendix C. ECHOSPEC delivers higher overall output throughput than these SD baseline methods in the high-concurrency regime. Specifically, ECHOSPEC achieves 6.8% higher output throughput on average than DFlash at temperature 0 for concurrencies 16–32. As a result, its throughput advantage over AR decoding declines more slowly with increasing concurrency compared to DFlash and EAGLE-3. For example, ECHOSPEC maintains higher output throughput than AR up to concurrency 24, while DFlash and EAGLE-3 do so only up to concurrencies 8 and 4, respectively (Figure 6b).

Beyond raw throughput, ECHOSPEC demonstrates significant advantages in resource efficiency. Table 3 reports the KV-cache token capacity for ECHOSPEC, DFlash, and EAGLE-3. ECHOSPEC retains nearly the same token capacity as AR, whereas the baseline methods reduce this capacity due to the additional state required by their auxiliary transformers. Furthermore, ECHOSPEC achieves




(a) Q3-8B, Gov. 4K–8K (b) Q3-8B, Gov. 8K+ (c) Q3-8B, Mul. 4K–8K (d) Q3-8B, Mul. 8K+

(e) Q3-4B, Gov. 4K–8K (f) Q3-4B, Gov. 8K+ (g) Q3-4B, Mul. 4K–8K (h) Q3-4B, Mul. 8K+

(i) L3.1-8B, Gov. 4K–8K (j) L3.1-8B, Gov. 8K+ (k) L3.1-8B, Mul. 4K–8K (l) L3.1-8B, Mul. 8K+

Figure 6: Output throughput against concurrency at temperature 0. Abbreviations adopted: Q3 for Qwen3; L3.1-8B for Llama-3.1-8B-Instruct; Gov. for GovReport; Mul. for MultiNews.

Table 4: Trainable parameter comparison across SD methods.

| Target | EAGLE-3 | DFlash | ECHOSPEC |
|---|---|---|---|
| Qwen3-8B | 399.7M | 1,048.6M | 32.7M |
| Qwen3-4B | 218.2M | 537.4M | 19.2M |
| Llama-3.1-8B-Instruct | 424.7M | 1,048.6M | 34.2M |

these results using substantially fewer trainable parameters. As detailed in Table 4, ECHOSPEC requires 27.9–32.1× fewer trainable parameters than DFlash and 11.4–12.4× fewer than EAGLE-3.

**Comparison against other target-integrated drafters.** Table 5 reports the draft quality of ECHOSPEC, Apple MTP, and PaSS under matched training settings. ECHOSPEC achieves a higher acceptance length $\tau$ than Apple MTP and PaSS by 11–21% and 5–12%, respectively. These gains are consistently observed across the evaluated targets and datasets, showing that ECHOSPEC produces higher-quality drafts than the other target-integrated drafters under the same training budget.

We further evaluate whether this improved draft quality translates into higher serving performance under matched serving settings. For all evaluated targets and datasets across concurrencies 16–32 and at temperature 0, ECHOSPEC improves output throughput over Apple MTP and PaSS by 13.2–22.5% and 1.5–7.1%, respectively. These results confirm that the higher acceptance length of ECHOSPEC consistently translates into higher output throughput in the evaluated high-concurrency regime.

**Ablation study of the refinement block.** We examine the contribution of the refinement block to draft quality in Table 5. For the Qwen3 targets and datasets, the block increases the acceptance

Table 5: Output throughput at 4K–8K and temperature 0, in tokens per second, with comparison of AR, Apple MTP, PaSS, ECHOSPEC with no block and ECHOSPEC. Additional results for 8K+ contexts and temperature 1 are reported in Appendix C.

| Configuration | Method | concurrency | | | | | | | | | $\tau$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 4 | 8 | 12 | 16 | 20 | 24 | 28 | 32 | |
| Qwen3-8B GovReport | AR | 72 | 208 | 304 | 354 | 389 | **420** | 431 | **439** | **445** | – |
| | PaSS | 105 | 274 | 359 | 370 | 394 | 393 | 414 | 397 | 404 | 1.90 |
| | Apple MTP | 79 | 218 | 302 | 330 | 354 | 361 | 379 | 368 | 378 | 1.87 |
| | ECHOSPEC, no block | **113** | 286 | 372 | 388 | 410 | 409 | 431 | 413 | 421 | 2.07 |
| | ECHOSPEC | 112 | **288** | **377** | **394** | **414** | 419 | **440** | 425 | 432 | **2.12** |
| Qwen3-8B MultiNews | AR | 71 | 203 | 297 | 353 | 384 | **420** | **428** | **433** | **442** | – |
| | PaSS | 103 | 263 | 347 | 360 | 380 | 383 | 402 | 387 | 395 | 1.81 |
| | Apple MTP | 75 | 207 | 288 | 316 | 338 | 348 | 364 | 356 | 361 | 1.74 |
| | ECHOSPEC, no block | 107 | 275 | 363 | 378 | 401 | 400 | 419 | 406 | 413 | 1.96 |
| | ECHOSPEC | **109** | **277** | **365** | **383** | **406** | 408 | 425 | 411 | 420 | **2.01** |
| Qwen3-4B GovReport | AR | 94 | 250 | 348 | 401 | 425 | **456** | **462** | **461** | **463** | – |
| | PaSS | 132 | 311 | 399 | 414 | 427 | 424 | 432 | 432 | 433 | 1.82 |
| | Apple MTP | 100 | 256 | 346 | 375 | 394 | 401 | 404 | 403 | 404 | 1.83 |
| | ECHOSPEC, no block | **139** | 325 | 417 | 432 | 444 | 446 | 450 | 455 | 448 | 1.98 |
| | ECHOSPEC | 138 | **328** | **418** | **437** | **451** | 454 | 458 | **461** | 462 | **2.03** |
| Qwen3-4B MultiNews | AR | 94 | 250 | 346 | 397 | 428 | **452** | **458** | **457** | **458** | – |
| | PaSS | 128 | 305 | 390 | 406 | 419 | 418 | 422 | 422 | 422 | 1.75 |
| | Apple MTP | 95 | 245 | 335 | 361 | 384 | 391 | 388 | 388 | 389 | 1.73 |
| | ECHOSPEC, no block | **134** | **317** | 401 | 420 | 437 | 439 | 440 | 441 | 436 | 1.90 |
| | ECHOSPEC | 132 | 316 | **405** | **428** | **442** | 443 | 445 | 446 | 445 | **1.93** |
| Llama-3.1-8B-Instruct GovReport | AR | 73 | 211 | 310 | 364 | 397 | 419 | 436 | 444 | 452 | – |
| | PaSS | 125 | 288 | 386 | 407 | 433 | 440 | 461 | 446 | 454 | 2.21 |
| | Apple MTP | 91 | 222 | 314 | 348 | 373 | 381 | 405 | 392 | 400 | 1.98 |
| | ECHOSPEC, no block | **129** | **296** | **394** | 419 | **449** | **447** | **473** | **456** | **464** | 2.32 |
| | ECHOSPEC | 126 | 289 | 386 | **420** | 444 | **447** | 471 | 455 | 462 | **2.35** |
| Llama-3.1-8B-Instruct MultiNews | AR | 73 | 215 | 315 | 373 | 409 | 441 | 445 | **459** | 463 | – |
| | PaSS | 120 | 282 | 381 | 407 | 434 | 441 | 460 | 447 | 455 | 2.07 |
| | Apple MTP | 84 | 210 | 302 | 338 | 360 | 373 | 390 | 383 | 390 | 1.80 |
| | ECHOSPEC, no block | **123** | **287** | **387** | **414** | **445** | 445 | 466 | 453 | 463 | 2.16 |
| | ECHOSPEC | 120 | 282 | 384 | 413 | 441 | **448** | **467** | 457 | **464** | **2.18** |

length $\tau$ by 1.6–2.6% over using O-bias alone. In contrast, the gain is more modest on Llama-3.1-8B-Instruct, where $\tau$ increases by 0.9–1.3%.

The serving results further show how these acceptance length gains affect actual output throughput. The refinement block improves output throughput on the Qwen3 targets by 1.3–2.2% across concurrencies of 16–32, demonstrating that the gain in acceptance length outweighs the computational overhead of the block. However, on Llama-3.1-8B-Instruct, the smaller gain in $\tau$ is insufficient to consistently offset this cost, resulting in a neutral throughput impact relative to O-bias alone.

## 5 CONCLUSION

In this paper, we presented ECHOSPEC, a target-integrated SD method designed to minimize drafting overhead while achieving competitive draft quality. ECHOSPEC enables future-token drafting through a lightweight O-bias adaptation and further improves draft quality with a post-target refinement block. ECHOSPEC requires significantly fewer trainable parameters and incurs lower training costs than auxiliary drafters. Furthermore, it delivers high output throughput with a smaller memory footprint in high-concurrency, long-context serving regimes. These results demonstrate that effective speculative drafting does not require a separate draft model, and that minimally adapting the target provides a highly promising direction for efficient LLM serving.

### AI USE STATEMENT

In this work, we used generative AI tools to aid and polish writing, retrieve and discover relevant works, and support ideation and execution. We have not used generative AI tools for generating draft sections of the paper, generating synthetic datasets and proving mathematical claims, and the rest of the required disclosure tasks are not applicable to this work. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

### REPRODUCIBILITY STATEMENT

We describe the details of ECHOSPEC and the training of its modules throughout Section 3 and provide the tree generation policy in Appendix A. In addition, we explain the evaluation setup including GPU types, model checkpoints, prompt datasets, and additional details in Section 4.1.

## REFERENCES

Amey Agrawal, Nitin Kedia, Ashish Panwar, Jayashree Mohan, Nipun Kwatra, Bhargav S. Gulavani, Alexey Tumanov, and Ramachandran Ramjee. Taming throughput-latency tradeoff in llm inference with sarathi-serve. In *Proceedings of the 18th USENIX Conference on Operating Systems Design and Implementation*, OSDI'24, USA, 2024. USENIX Association. ISBN 978-1-939133-40-3.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, et al. Longbench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 3119–3137, 2024.

Sahil Chaudhary. Code alpaca: An instruction-following llama model for code generation, 2023. URL https://github.com/sahil280114/codealpaca.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating large language model decoding with speculative sampling, 2023a. URL https://arxiv.org/abs/2302.01318.

Jian Chen, Yesheng Liang, and Zhijian Liu. DFlash: Block diffusion for flash speculative decoding. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=Oz335dV48X.

Lingjiao Chen, Matei Zaharia, and James Zou. How is chatgpt's behavior changing over time?, 2023b. URL https://arxiv.org/abs/2307.09009.

Zhuoming Chen, Avner May, Ruslan Svirschevski, Yuhsun Huang, Max Ryabinin, Zhihao Jia, and Beidi Chen. Sequoia: Scalable and robust speculative decoding. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 129531–129563. Curran Associates, Inc., 2024. doi: 10.52202/079017-4116. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/ea1f5f0878d43ff4fb8bf64ef4a2326c-Paper-Conference.pdf.

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

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, SOSP '23, pp. 611–626, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400702297. doi: 10.1145/3600006.3613165. URL https://doi.org/10.1145/3600006.3613165.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 19274–19286. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/leviathan23a.html.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE-2: Faster inference of language models with dynamic draft trees. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 7421–7432, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.422. URL https://aclanthology.org/2024.emnlp-main.422/.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle-3: Scaling up inference acceleration of large language models via training-time test. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 136737–136756. Curran Associates, Inc., 2025. doi: 10.52202/085713-4562. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/c7b5a35ea98b62512a869c19ea7b03cb-Paper-Conference.pdf.

Feng Lin, Hanling Yi, Yifan Yang, Hongbin Li, Xiaotian Yu, Guangming Lu, and Rong Xiao. Bita: Bi-directional tuning for lossless acceleration in large language models. *Expert Systems with Applications*, 279:127305, 2025. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2025.127305. URL https://www.sciencedirect.com/science/article/pii/S0957417425009273.

Giovanni Monea, Armand Joulin, and Edouard Grave. Pass: Parallel speculative sampling, 2023. URL https://arxiv.org/abs/2311.13581.

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

OpenAI, :, Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, Alex Iftimie, Alex Karpenko, Alex Tachard Passos, Alexander Neitz, Alexander Prokofiev, Alexander Wei, Allison Tam, Ally Bennett, Ananya Kumar, Andre Saraiva, Andrea Vallone, Andrew Duberstein, Andrew Kondrich, Andrey Mishchenko, Andy Applebaum, Angela Jiang, Ashvin Nair, Barret Zoph, Behrooz Ghorbani, Bohan Zhang, Ben Rossen, Benjamin Sokolowsky, Boaz Barak, Bob McGrew, Borys Minaiev, Botao Hao, Bowen Baker, Brandon Houghton, Brandon McKinzie, Brydon Eastman, Camillo Lugaresi, Cary Bassin, Cary Hudson, Chak Ming Li, Charles de Bourcy, Chelsea Voss, Chen Shen, Chong Zhang, Chris Koch, Chris Orsinger, Christopher Hesse, Claudia Fischer, Clive Chan, Dan Roberts, Daniel Kappler, Daniel Levy, Daniel Selsam, David Dohan, David Farhi, David Mely, David Robinson, Dimitris Tsipras, Doug Li, Dragos Oprica, Eben Freeman, Eddie Zhang, Edmund Wong, Elizabeth Proehl, Enoch Cheung, Eric Mitchell, Eric Wallace, Erik Ritter, Evan Mays, Fan Wang, Felipe Petroski Such, Filippo Raso, Florencia Leoni, Foivos Tsimpourlas, Francis Song, Fred von Lohmann, Freddie Sulit, Geoff Salmon, Giambattista Parascandolo, Gildas Chabot, Grace Zhao, Greg Brockman, Guillaume Leclerc, Hadi Salman, Haiming Bao, Hao Sheng, Hart Andrin, Hessam Bagherinezhad, Hongyu Ren, Hunter Lightman, Hyung Won Chung, Ian Kivlichan, Ian O'Connell, Ian Osband, Ignasi Clavera Gilaberte, Ilge Akkaya, Ilya Kostrikov, Ilya Sutskever, Irina Kofman, Jakub Pachocki, James Lennon, Jason Wei, Jean Harb, Jerry Twore, Jiacheng Feng, Jiahui Yu, Jiayi Weng, Jie Tang, Jieqi Yu, Joaquin Quiñonero Candela, Joe Palermo, Joel Parish, Johannes Heidecke, John Hallman, John Rizzo, Jonathan Gordon, Jonathan Uesato, Jonathan Ward, Joost Huizinga, Julie Wang, Kai Chen, Kai Xiao, Karan Singhal, Karina Nguyen, Karl Cobbe, Katy Shi, Kayla Wood, Kendra Rimbach, Keren Gu-Lemberg, Kevin Liu, Kevin Lu, Kevin Stone, Kevin Yu, Lama Ahmad, Lauren Yang, Leo Liu, Leon Maksin, Leyton Ho, Liam Fedus, Lilian Weng, Linden Li, Lindsay McCallum, Lindsey

Held, Lorenz Kuhn, Lukas Kondraciuk, Lukasz Kaiser, Luke Metz, Madelaine Boyd, Maja Trebacz, Manas Joglekar, Mark Chen, Marko Tintor, Mason Meyer, Matt Jones, Matt Kaufer, Max Schwarzer, Meghan Shah, Mehmet Yatbaz, Melody Y. Guan, Mengyuan Xu, Mengyuan Yan, Mia Glaese, Mianna Chen, Michael Lampe, Michael Malek, Michele Wang, Michelle Fradin, Mike McClay, Mikhail Pavlov, Miles Wang, Mingxuan Wang, Mira Murati, Mo Bavarian, Mostafa Rohaninejad, Nat McAleese, Neil Chowdhury, Neil Chowdhury, Nick Ryder, Nikolas Tezak, Noam Brown, Ofir Nachum, Oleg Boiko, Oleg Murk, Olivia Watkins, Patrick Chao, Paul Ashbourne, Pavel Izmailov, Peter Zhokhov, Rachel Dias, Rahul Arora, Randall Lin, Rapha Gontijo Lopes, Raz Gaon, Reah Miyara, Reimar Leike, Renny Hwang, Rhythm Garg, Robin Brown, Roshan James, Rui Shu, Ryan Cheu, Ryan Greene, Saachi Jain, Sam Altman, Sam Toizer, Sam Toyer, Samuel Miserendino, Sandhini Agarwal, Santiago Hernandez, Sasha Baker, Scott McKinney, Scottie Yan, Shengjia Zhao, Shengli Hu, Shibani Santurkar, Shraman Ray Chaudhuri, Shuyuan Zhang, Siyuan Fu, Spencer Papay, Steph Lin, Suchir Balaji, Suvansh Sanjeev, Szymon Sidor, Tal Broda, Aidan Clark, Tao Wang, Taylor Gordon, Ted Sanders, Tejal Patwardhan, Thibault Sottiaux, Thomas Degry, Thomas Dimson, Tianhao Zheng, Timur Garipov, Tom Stasi, Trapit Bansal, Trevor Creech, Troy Peterson, Tyna Eloundou, Valerie Qi, Vineet Kosaraju, Vinnie Monaco, Vitchyr Pong, Vlad Fomenko, Weiyi Zheng, Wenda Zhou, Wenting Zhan, Wes McCabe, Wojciech Zaremba, Yann Dubois, Yinghai Lu, Yining Chen, Young Cha, Yu Bai, Yuchen He, Yuchen Zhang, Yunyun Wang, Zheng Shao, and Zhuohan Li. Openai o1 system card, 2026. URL https://arxiv.org/abs/2412.16720.

Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. Efficiently scaling transformer inference. In D. Song, M. Carbin, and T. Chen (eds.), *Proceedings of Machine Learning and Systems*, volume 5, pp. 606–624. Curan, 2023. URL https://proceedings.mlsys.org/paper_files/paper/2023/file/c4be71ab8d24cdfb45e3d06dbfca2780-Paper-mlsys2023.pdf.

Mohammad Samragh, Arnav Kundu, David Harrison, Kumari Nishu, Devang Naik, Minsik Cho, and Mehrdad Farajtabar. Your llm knows the future: Uncovering its multi-token prediction potential, 2025. URL https://arxiv.org/abs/2507.11851.

Noam Shazeer. Fast transformer decoding: One write-head is all you need, 2019. URL https://arxiv.org/abs/1911.02150.

Tencent. AngelSlim, 6 2025. URL https://github.com/Tencent/AngelSlim.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Zican Dong, Yupeng Hou, Beichen Zhang, Yingqian Min, Junjie Zhang, Peiyu Liu, et al. A survey of large language models. *Frontiers of Computer Science*, 20(12):2012627, 2026.

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. Sglang: Efficient execution of structured language model programs. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 62557–62583. Curran Associates, Inc., 2024. doi: 10.52202/079017-2000. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/724be4472168f31ba1c9ac630f15dec8-Paper-Conference.pdf.

## A How the draft tree is built

ECHOSPEC constructs the draft tree using draft token probabilities, following the tree construction of Chen et al. (2024); Li et al. (2024). Since ECHOSPEC generates candidates for multiple draft positions in parallel, multiple candidate sequences can be constructed by combining candidates across these positions. We denote the probability of the $r_j$-th ranked candidate at draft position $j$ by $c_j^{r_j}$. For a candidate sequence $\mathcal{S}$, the cumulative probability is given by

$$P(\mathcal{S}) = \prod_{j=1}^{d} c_j^{r_j}, \tag{5}$$

where $d$ denotes the sequence length and $\mathcal{S} = (r_1, \ldots, r_d)$ denotes the sequence of candidate ranks selected in the first $d$ draft positions.

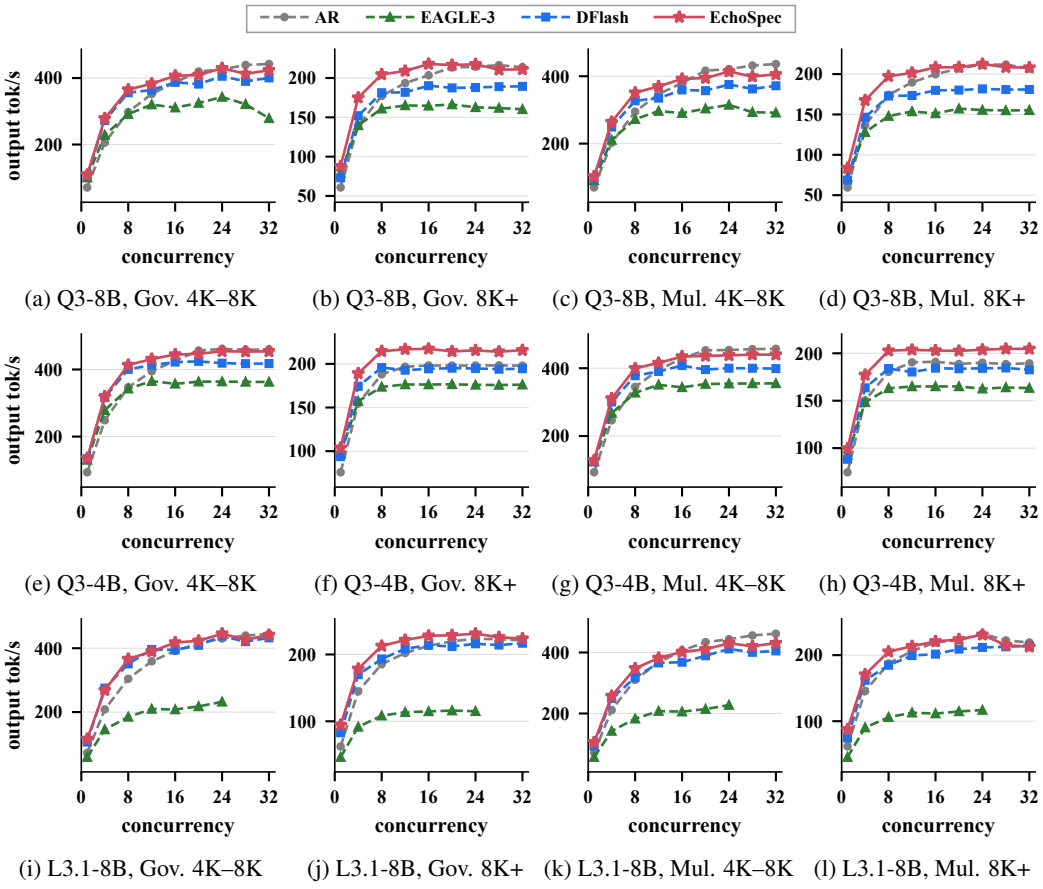
We construct the tree by ranking candidate sequences according to their cumulative probabilities and selecting higher-probability sequences first until the predefined number of candidate nodes is reached. Candidate sequences that share the same prefix also share the corresponding nodes in the tree. In this way, the tree allocates more candidate nodes to draft sequences with higher probability while maintaining a fixed verification size.

## B Settings where CUDA Graph runs out of memory

CUDA Graph capture runs out of memory for EAGLE-3 on Llama-3.1-8B-Instruct in the 8K+ band at concurrency 28 and 32, across all datasets and at temperatures 0 and 1.

## C EVALUATION ABLATIONS

Figure 7 provides output throughput against concurrency at temperature 1 across both context-length bands. Tables 6 and 7 provide the remaining detailed serving results for 4K–8K at temperature 1 and 8K+ at temperatures 0 and 1.




(a) Q3-8B, Gov. 4K–8K (b) Q3-8B, Gov. 8K+ (c) Q3-8B, Mul. 4K–8K (d) Q3-8B, Mul. 8K+

(e) Q3-4B, Gov. 4K–8K (f) Q3-4B, Gov. 8K+ (g) Q3-4B, Mul. 4K–8K (h) Q3-4B, Mul. 8K+

(i) L3.1-8B, Gov. 4K–8K (j) L3.1-8B, Gov. 8K+ (k) L3.1-8B, Mul. 4K–8K (l) L3.1-8B, Mul. 8K+

Figure 7: Output throughput against concurrency at temperature 1, in the layout of Figure 6.

We report that ECHOSPEC delivers higher output throughput overall than these SD baseline methods in the high-concurrency regime, even at temperature 1 in Figure 7. Specifically, ECHOSPEC achieves 8.6% higher output throughput on average than DFlash at temperature 1 for concurrencies 16–32. As a result, even at temperature 1, ECHOSPEC still shows a similar trend that its throughput advantage over AR decoding declines more slowly with increasing concurrency compared to DFlash and EAGLE-3.

The trends of comparison between ECHOSPEC, Apple MTP and PaSS are observed in Table 5 generally also hold at temperature 1 and with 8K+ contexts as shown in Tables 6 and 7. We show that the higher acceptance length of ECHOSPEC translates into higher output throughput in most cases even at temperature 1 or with 8K+ contexts. ECHOSPEC achieves a higher acceptance length $\tau$ than Apple MTP and PaSS by 11%–20% and 4%–12%, respectively. For all evaluated targets and datasets across concurrencies 16–32, ECHOSPEC generally changes output throughput over Apple MTP and PaSS by 8.6–19.3% and −0.7–7.8%, respectively.

We also examine that refinement block can gain higher draft quality even at temperature 1 or at 8K+ contexts in Tables 6 and 7. For the Qwen3 targets and datasets, the block increases the acceptance length $\tau$ by 2.0–3.6% over using O-bias alone. In contrast, the gain is more modest on Llama-3.1-8B-Instruct, where $\tau$ increases by 0.5–1.1%. Furthermore, the refinement block changes output throughput on the Qwen3 targets by −0.9–3.9% across concurrencies of 16–32, demonstrating that the gain in acceptance length usually outweighs the computational overhead of the block. However,

Table 6: Output throughput at 4K–8K and temperature 1, in the layout of Table 5.

| Configuration | Method | concurrency 1 | 4 | 8 | 12 | 16 | 20 | 24 | 28 | 32 | $\tau$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Qwen3-8B GovReport | AR | 71 | 206 | 298 | 352 | 387 | **420** | 426 | **439** | **442** | – |
| | PaSS | 103 | 267 | 348 | 360 | 382 | 383 | 404 | 388 | 397 | 1.82 |
| | Apple MTP | 77 | 213 | 295 | 324 | 345 | 352 | 373 | 361 | 368 | 1.80 |
| | EchoSpec, no block | 107 | 273 | 360 | 374 | 395 | 396 | 420 | 402 | 411 | 1.97 |
| | EchoSpec | **109** | **279** | **365** | **383** | **407** | 409 | **430** | 413 | 424 | **2.02** |
| Qwen3-8B MultiNews | AR | 70 | 203 | 295 | 347 | 380 | **417** | **421** | **432** | **436** | – |
| | PaSS | 98 | 255 | 334 | 349 | 371 | 371 | 388 | 375 | 381 | 1.73 |
| | Apple MTP | 75 | 205 | 284 | 310 | 330 | 339 | 357 | 345 | 352 | 1.67 |
| | EchoSpec, no block | 101 | 261 | 343 | 357 | 380 | 380 | 400 | 386 | 395 | 1.84 |
| | EchoSpec | **102** | **264** | **351** | **370** | **392** | 395 | 414 | 399 | 406 | **1.89** |
| Qwen3-4B GovReport | AR | 94 | 249 | 347 | 395 | 426 | **456** | **462** | **458** | **460** | – |
| | PaSS | 129 | 310 | 394 | 409 | 424 | 423 | 431 | 427 | 426 | 1.79 |
| | Apple MTP | 98 | 252 | 338 | 367 | 389 | 399 | 399 | 400 | 396 | 1.80 |
| | EchoSpec, no block | **137** | **321** | 410 | 425 | 441 | 442 | 445 | 449 | 442 | 1.95 |
| | EchoSpec | 136 | 319 | **413** | **431** | **445** | 447 | 454 | 453 | 454 | **1.99** |
| Qwen3-4B MultiNews | AR | 93 | 246 | 345 | 395 | 429 | **453** | **454** | **456** | **457** | – |
| | PaSS | 124 | 300 | 382 | 398 | 416 | 414 | 418 | 416 | 410 | 1.72 |
| | Apple MTP | 93 | 243 | 330 | 358 | 378 | 383 | 391 | 380 | 384 | 1.70 |
| | EchoSpec, no block | **130** | 310 | 397 | 413 | 431 | 432 | 436 | 438 | 431 | 1.86 |
| | EchoSpec | 128 | **311** | **400** | **416** | **434** | 436 | 438 | 440 | 440 | **1.90** |
| Llama-3.1-8B-Instruct GovReport | AR | 72 | 208 | 304 | 359 | 392 | 415 | 430 | **440** | 445 | – |
| | PaSS | 116 | **272** | 364 | 381 | 409 | 410 | 436 | 419 | 433 | 2.04 |
| | Apple MTP | 84 | 207 | 294 | 329 | 355 | 356 | 377 | 363 | 379 | 1.83 |
| | EchoSpec, no block | **119** | **272** | **371** | **395** | **422** | 422 | 445 | 430 | **446** | 2.12 |
| | EchoSpec | 116 | 269 | 366 | 390 | 418 | **423** | **446** | 426 | 441 | **2.14** |
| Llama-3.1-8B-Instruct MultiNews | AR | 72 | 212 | 311 | 367 | **406** | **434** | **443** | **456** | **462** | – |
| | PaSS | 107 | **260** | **355** | 377 | 405 | 400 | 423 | 410 | 416 | 1.85 |
| | Apple MTP | 76 | 193 | 280 | 310 | 337 | 344 | 363 | 357 | 362 | 1.64 |
| | EchoSpec, no block | **108** | 256 | 352 | 379 | 402 | 406 | 430 | 405 | 429 | 1.90 |
| | EchoSpec | 105 | 257 | 348 | **382** | 402 | 410 | 430 | 420 | 431 | **1.92** |

on Llama-3.1-8B-Instruct, the refinement block results in a neutral throughput impact relative to O-bias alone.

Table 7: Output throughput at 8K+ in the layout of Table 5.

| Configuration | Method | concurrency 1 | 4 | 8 | 12 | 16 | 20 | 24 | 28 | 32 | $\tau$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | Temperature = 0 | | | | | | |
| Qwen3-8B GovReport | AR | 61 | 123 | 179 | 195 | 205 | 214 | 214 | **217** | **215** | – |
| | PaSS | 87 | 173 | 202 | 203 | 209 | 208 | 210 | 204 | 204 | 1.88 |
| | Apple MTP | 68 | 147 | 183 | 189 | 198 | 199 | 200 | 197 | 197 | 1.85 |
| | EchoSpec, no block | 91 | 177 | 207 | 209 | 216 | 215 | 215 | 213 | 213 | 2.05 |
| | EchoSpec | **92** | **179** | **209** | **212** | **219** | **218** | **220** | 215 | 214 | **2.11** |
| Qwen3-8B MultiNews | AR | 61 | 122 | 176 | 192 | 201 | 208 | 214 | **213** | **214** | – |
| | PaSS | 84 | 166 | 194 | 196 | 202 | 202 | 207 | 202 | 202 | 1.81 |
| | Apple MTP | 65 | 142 | 175 | 183 | 189 | 192 | 197 | 191 | 191 | 1.74 |
| | EchoSpec, no block | 87 | 170 | 199 | 202 | 207 | 207 | 214 | 208 | 209 | 1.95 |
| | EchoSpec | **88** | **172** | **202** | **205** | **210** | **211** | **216** | 211 | 210 | **2.00** |
| Qwen3-4B GovReport | AR | 77 | 156 | 189 | 197 | 197 | 197 | 197 | 197 | 197 | – |
| | PaSS | 102 | 185 | 209 | 209 | 209 | 209 | 209 | 208 | 209 | 1.80 |
| | Apple MTP | 81 | 162 | 194 | 197 | 196 | 197 | 197 | 196 | 196 | 1.79 |
| | EchoSpec, no block | **107** | **191** | 217 | 217 | 217 | 218 | 217 | 216 | 217 | 1.96 |
| | EchoSpec | 106 | **191** | **219** | **219** | **219** | **219** | **218** | **219** | **219** | **2.00** |
| Qwen3-4B MultiNews | AR | 76 | 150 | 182 | 188 | 187 | 187 | 187 | 188 | 188 | – |
| | PaSS | 98 | 175 | 198 | 196 | 196 | 196 | 195 | 195 | 196 | 1.75 |
| | Apple MTP | 78 | 155 | 183 | 182 | 183 | 182 | 182 | 182 | 182 | 1.73 |
| | EchoSpec, no block | **103** | **180** | **206** | 203 | **202** | **203** | **203** | 203 | 200 | 1.89 |
| | EchoSpec | 101 | **180** | 205 | **204** | **202** | 202 | **203** | **204** | **203** | **1.93** |
| Llama-3.1-8B-Instruct GovReport | AR | 61 | 145 | 185 | 203 | 215 | 220 | 222 | 224 | 221 | – |
| | PaSS | 102 | 189 | 225 | 228 | 237 | 236 | 240 | 234 | 232 | 2.23 |
| | Apple MTP | 78 | 156 | 195 | 206 | 214 | 215 | 217 | 215 | 213 | 1.97 |
| | EchoSpec, no block | **104** | **191** | **228** | **233** | **242** | **240** | **244** | 238 | **238** | 2.33 |
| | EchoSpec | 103 | 189 | 224 | 230 | 240 | 239 | 243 | **239** | 237 | **2.35** |
| Llama-3.1-8B-Instruct MultiNews | AR | 63 | 148 | 191 | 212 | 223 | 228 | 235 | 226 | 224 | – |
| | PaSS | 99 | 185 | 223 | 228 | 236 | 237 | 243 | 227 | 223 | 2.08 |
| | Apple MTP | 73 | 149 | 190 | 202 | 210 | 213 | 219 | 206 | 201 | 1.82 |
| | EchoSpec, no block | **100** | **188** | **226** | **234** | **240** | **242** | **248** | **232** | **227** | 2.17 |
| | EchoSpec | 98 | 185 | 224 | 231 | 239 | 241 | 246 | 231 | **227** | **2.18** |
| | | | | | Temperature = 1 | | | | | | |
| Qwen3-8B GovReport | AR | 61 | 139 | 177 | 194 | 204 | 213 | 214 | **216** | **214** | – |
| | PaSS | 84 | 168 | 197 | 200 | 206 | 204 | 207 | 201 | 199 | 1.80 |
| | Apple MTP | 66 | 145 | 180 | 186 | 196 | 196 | 197 | 194 | 193 | 1.77 |
| | EchoSpec, no block | 87 | 171 | 202 | 206 | 211 | 210 | 211 | 209 | 208 | 1.93 |
| | EchoSpec | **88** | **175** | **205** | **209** | **218** | **216** | **218** | 211 | 211 | **2.00** |
| Qwen3-8B MultiNews | AR | 60 | 137 | 174 | 190 | 200 | 207 | **212** | **212** | 207 | – |
| | PaSS | 81 | 163 | 191 | 192 | 198 | 199 | 205 | 194 | 193 | 1.72 |
| | Apple MTP | 65 | 140 | 173 | 180 | 186 | 190 | 194 | 184 | 189 | 1.67 |
| | EchoSpec, no block | 83 | 165 | 195 | 196 | 202 | 204 | 207 | 205 | 205 | 1.83 |
| | EchoSpec | **84** | **168** | **197** | **202** | **208** | **209** | **212** | 208 | **208** | **1.88** |
| Qwen3-4B GovReport | AR | 76 | 155 | 188 | 197 | 198 | 198 | 198 | 198 | 198 | – |
| | PaSS | 100 | 183 | 206 | 206 | 206 | 207 | 205 | 205 | 207 | 1.77 |
| | Apple MTP | 80 | 160 | 194 | 195 | 194 | 192 | 195 | 191 | 193 | 1.77 |
| | EchoSpec, no block | **104** | **189** | 213 | 213 | 214 | **215** | 214 | **216** | 214 | 1.92 |
| | EchoSpec | **104** | **189** | **215** | **217** | **217** | 214 | **215** | 214 | **216** | **1.96** |
| Qwen3-4B MultiNews | AR | 75 | 150 | 180 | 190 | 191 | 189 | 190 | 189 | 189 | – |
| | PaSS | 97 | 173 | 198 | 196 | 193 | 194 | 197 | 196 | 195 | 1.71 |
| | Apple MTP | 77 | 153 | 181 | 181 | 183 | 183 | 182 | 182 | 183 | 1.70 |
| | EchoSpec, no block | **100** | **178** | 201 | 202 | 199 | 202 | 202 | 201 | 203 | 1.85 |
| | EchoSpec | 99 | 177 | **203** | **204** | **203** | **203** | **204** | **205** | **205** | **1.89** |
| Llama-3.1-8B-Instruct GovReport | AR | 62 | 145 | 185 | 202 | 213 | 220 | 224 | 223 | 221 | – |
| | PaSS | 95 | 176 | 213 | 219 | 228 | 223 | 226 | 219 | 219 | 2.03 |
| | Apple MTP | 71 | 148 | 183 | 195 | 204 | 204 | 208 | 202 | 201 | 1.82 |
| | EchoSpec, no block | **96** | **180** | **214** | 219 | **231** | 226 | **232** | **226** | **226** | 2.11 |
| | EchoSpec | 94 | 179 | 213 | **222** | 228 | **229** | 231 | **226** | 224 | **2.12** |
| Llama-3.1-8B-Instruct MultiNews | AR | 62 | 146 | 187 | 206 | 219 | 223 | **233** | **222** | **219** | – |
| | PaSS | **89** | 170 | 208 | **215** | **221** | 221 | 229 | 211 | 207 | 1.84 |
| | Apple MTP | 66 | 138 | 177 | 191 | 194 | 197 | 202 | 189 | 187 | 1.64 |
| | EchoSpec, no block | **89** | **172** | **210** | **215** | **221** | **224** | 230 | 214 | 212 | 1.89 |
| | EchoSpec | 88 | 170 | 205 | 214 | **221** | **224** | 231 | 215 | 213 | **1.91** |