WINDOW POOL: VERIFY-ALIGNED ROUTING FOR MOE SPECULATIVE DECODING

Anonymous authors
Paper under double-blind review

ABSTRACT

Speculative decoding (SD) verifies $\gamma{+}1$ draft tokens in a single target forward pass. For a Mixture-of-Experts (MoE) target this parallelism undercuts the intended saving: the verification block activates the *union* of the per-token top-k expert sets, so expert-weight traffic grows even though the token count is amortized. Existing remedies cap experts at serve time or reshape the draft tree, but leave the pretrained router untouched. We instead attack the union inflation at its source: the pretrained routing prior. We first show that document-level expert-pool pretraining already induces a prior favorable to speculation—lower verification inflation, and graceful rather than catastrophic degradation under forced window sharing—and then build on it. We propose **Window Pool**, a continued-pretraining recipe that aligns the expert-sharing group with the SD verification window, with randomized window length and pool size, layer selection, and interleaved unconstrained steps. *Our main result is on the deployment grid:* across a $W \times d_w$ sweep of prefill-only window-sharing constraints, Window Pool beats the document-pool base in every one of the twelve constrained cells (typical gap $+0.5$–1.0 mean Avg), while preserving unconstrained quality. When the same routers are placed under a hard per-verify-round expert budget B (uniform MoE-Spec substitution on all MoE layers), the trained prior—not the serve-time cap—is what makes the budget survivable: at $B{=}8$ the standard router collapses (MMLU $50.3 \rightarrow 25.5$, LAMBADA $72.8 \rightarrow 0.5$) while Window Pool retains MMLU 40.6 and LAMBADA 63.6. On HumanEval-164 end-to-end SD, Window Pool does not regress under budgeting ($\tau 3.60$, $16/164$ at traffic 128) where the standard router falls to 3.15 / $6/164$; against the document-pool base the SD gap is within task-count noise, so the contribution is constrained-sharing robustness rather than SD speedup. Window Pool is a training-time routing prior rather than a serve-time heuristic: by internalizing the verification constraint during continued pretraining, it renders aggressive expert-budgeting viable for MoE speculative decoding.

1 INTRODUCTION

Autoregressive decoding of large language models is memory-bound: every generated token requires streaming model weights from memory to compute (Leviathan et al., 2023). Speculative decoding (Leviathan et al., 2023; Chen et al., 2023) attacks this by having a cheap drafter propose several tokens that the target model verifies in one batched forward pass, converting a sequence of memory-bound steps into a single wider one. Modern drafters such as EAGLE-3 and DFlash (Li et al., 2025; Chen et al., 2026) make this practical, delivering multi-fold speedups on dense targets.

Mixture-of-Experts (MoE) models (Shazeer et al., 2017; Fedus et al., 2022) appear to be an ideal target: they activate only k of n experts per token, so per-token weight traffic is already small. In practice the two techniques interact badly. During verification the target processes a block of $\gamma{+}1$ draft tokens simultaneously, and each token routes independently. The block therefore touches the *union* of the per-token expert sets, whose cardinality can substantially exceed k (Saxena et al., 2025; McDanel et al., 2026). Speculation buys parallelism in tokens but pays for it in expert-weight traffic, and under memory offloading the payment can exceed the gain.

Recent work mitigates this mismatch at inference time by capping verifier-side expert usage or reshaping draft-tree expansion, without changing the pretrained router (Section 2). A complementary

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.

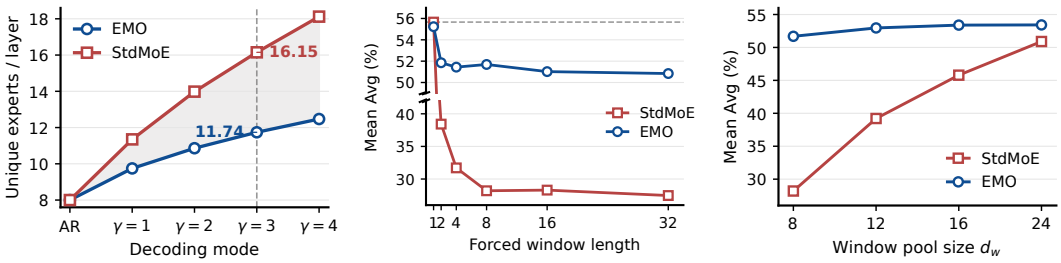


Figure 1: The routing prior controls verification inflation and tolerance to expert sharing (StdMoE vs. EMO). **(a)** Unique experts per layer under EAGLE-3 verification on HumanEval-16, against autoregressive decoding (AR, $|U|{=}k{=}8$). **(b)** 11-task mean Avg with pool size fixed at $d_w{=}8$ (forced broadcast, since $d_w{=}k$; prefill only, all layers) while window length W varies. **(c)** The same mean, also prefill only and all layers, with window length fixed at $W{=}8$ while pool size $d_w \in \{8, 12, 16, 24\}$ varies ($d_w{=}8$ recovers the broadcast in **(b)**). All three panels constrain *all* layers at prefill; the deployment grid of Section 6.2 instead restricts to layers 2–15, so absolute numbers are not directly comparable across the two.

line reshapes routing during training, but existing objectives either smooth adjacent-token switches without bounding block-level unions, or pool experts at document granularity—much longer than a verification window—so they do not directly target SD verification cost (Section 2).

The gap is the training-time sharing group: it should match the unit inference pays for, the SD verification window. A hard expert-union cap at verification is enforced by standard MoE-Spec, not by a new selector. Section 4 establishes two facts on a 1B-active / 14B-total MoE (16 layers, 127 routed experts plus one shared expert, top-$k{=}8$): document-pool pretraining already cuts verification inflation (Figure 1**(a)**) and makes zero-inflation window sharing survivable rather than catastrophic (Figure 1**(b)**–**(c)**). The obstacle is therefore not that expert sharing is fundamentally lossy, but that standard routers are trained where no sharing is ever required.

We close that gap by training the prior (Figure 2). **Window Pool** (Section 5) continues pretraining so that, in addition to the document pool, tokens inside a window of length W share an expert pool S_w of size d_w. The naive version—a fixed, always-on window constraint—is worse than not training at all. Three ingredients are needed: (i) randomizing W and d_w per step so one checkpoint serves many deployment budgets, (ii) exempting the lowest layers, where routing is least clusterable, and (iii) interleaving unconstrained "native" steps so the model does not forget how to route freely. At serve time, a hard per-verify-round expert budget is enforced by standard MoE-Spec (McDanel et al., 2026): aggregate router mass over the draft tree, keep a top-B shortlist S_B, and substitute each token's top-k into S_B, capping per-layer unique experts at B. Window pools S_w and shortlists S_B are different objects—one is a training/deployment sharing group, the other a per-round union cap—but both target the same cost: $|U(S)|$ during verify. The contribution of this paper is the trained prior; the serve-time cap is an off-the-shelf budget enforcer that the prior makes survivable.

Our contributions are:

- We show that document-pool MoE induces a routing prior favorable to speculative decoding: lower verification inflation, and graceful rather than catastrophic degradation under forced window sharing.
- **Window Pool**, a continued-pretraining recipe that aligns the expert-sharing group with the SD verification window, with randomized budgets, layer selection, and native-step interleaving. On a $W \times d_w$ deployment grid of prefill-only window-sharing constraints, Window Pool beats the document-pool base in every one of the twelve constrained cells while preserving unconstrained quality.
- A budget-robustness study under a hard MoE-Spec B on the same routers: the trained prior, not the serve-time cap, is what makes the budget survivable—at $B{=}8$ the standard router collapses (MMLU 50.3 $\rightarrow$ 25.5 vs. 49.9 $\rightarrow$ 40.6 for Window Pool). On HumanEval-164 SD, Window Pool does not regress under budgeting (τ3.60, 16/164 at traffic 128);

against the document-pool base the SD gap is within task-count noise, so the contribution is constrained-sharing robustness, not SD speedup.

Scope. We report routing-level inflation, quality under fixed (W, d_w) budgets, and end-to-end speculative-decoding metrics (acceptance length, pass@1, profiled traffic, and matched-B MoE-Spec on StdMoE / EMO / Window Pool) on a 1B/14B MoE. Wall-clock speedup under an offloading stack is deferred to Section 7.

2 Related Work

Speculative decoding. Speculative decoding (Leviathan et al., 2023; Chen et al., 2023) accelerates generation by verifying draft tokens in parallel while preserving the target distribution. Later drafters improved acceptance and drafting cost in two waves: feature-based heads such as Medusa (Cai et al., 2024) and EAGLE-3 (Li et al., 2025), followed by increasingly parallel draft generation via multi-token prediction (DeepSeek-AI, 2024), DFlash (Chen et al., 2026), and DSpark (DeepSeek-AI, 2026). We do not modify the drafter or the verification acceptance rule. We change how the *target* routes during training; at verification we apply standard MoE-Spec.

MoE-aware speculative decoding. Saxena et al. (2025) showed that MoE targets can forfeit the benefit of speculation because a verification block activates the *union* of per-token expert sets, not just k experts per token. Training-free remedies since then split along two axes. *Verifier-side* methods cap how many experts a verification round may touch: MoE-Spec (McDanel et al., 2026) ranks experts from aggregated router mass on the draft tree and enforces a fixed per-layer budget via substitution; AcceptMoE (Liang et al., 2026) self-sizes the eligible set from commitment probabilities and cache residency. *Draft-side* methods reduce the union before verification: EcoSpec (Xie et al., 2026) biases tree expansion toward prefixes that reuse already-active experts, while EVICT (Pan et al., 2026) truncates the tree by draft-path benefit and verification-latency cost (it does not predict expert reuse). All of these take the pretrained router as fixed. **Window Pool** continues pretraining so window-level sharing is cheap on the target router itself, rather than only clipping experts after the fact. At serve time we apply the same broad verifier-side pattern as MoE-Spec—aggregate tree router mass, keep a top-B shortlist, substitute—as an off-the-shelf budget enforcer on a Window Pool target, with window pools S_w and verify shortlists S_B kept as distinct objects (Figure 2). We do not claim a new ranking criterion; the contribution is the verify-aligned training prior and the observation that it makes a fixed serve-time B survivable where a standard router collapses.

Inference-time reuse-aware gating. S2-MoE (Huang et al., 2026) is the closest inference-time counterpart: it aggregates speculative-token routing logits and applies a soft bias toward reusable experts at serve time, without retraining. Window Pool pursues the same goal—expert reuse across speculative tokens—but moves the intervention into training: the target router and experts are jointly adapted to a shared window pool, so reuse becomes a property of the representations rather than a serve-time patch. We therefore do not claim Window Pool as the first method to promote speculative-token expert sharing; the novel claim is the training-time, verify-window-aligned routing/representation adaptation. A direct experimental comparison against S2-MoE is left to future work; here we position it as the closest conceptual counterpart rather than as an experimental baseline.

Training-time routing locality. Instruction-following pruning (Hou et al., 2025) makes sparsity depend on the user instruction rather than a fixed pattern. EMO (Wang et al., 2026) constrains routing to document-level pools for modularity and subset deployment. StickyMoE (Kayyam, 2026) penalizes expert switches between adjacent tokens to improve cache hit rates on edge devices. Window Pool is closest in spirit to this line, but constrains a different object: not pairwise smoothness or document-scale pooling, but the union $|U(S)|$ over a verification-width window. We retain EMO's document pool and add a second, shorter-horizon level aligned with that window. A serve-time MoE-Spec cap does not replace this training signal: it enforces an explicit B when soft window routing alone still yields a large tree union.

Edge deployment. CATS (Han et al., 2026) targets speculative decoding under device memory constraints, where expert-weight traffic dominates. This is the regime in which reducing $|U(S)|$—

whether by Window Pool or by a serve-time MoE-Spec cap—translates most directly into wall-clock gains, and is our intended deployment setting.

3 Background and Problem Setup

3.1 MoE routing

Let a MoE layer hold n routed experts and (optionally) a shared expert that is always active. For token t with hidden state x_t, the router produces $p_t = \mathrm{softmax}(r(x_t)) \in \mathbb{R}^n$ and the layer activates the top-k entries of p_t. In our target model, n=127 routed experts plus one shared expert, and k=8.

Document-level expert pools. EMO (Wang et al., 2026) adds a second level of selection. For a document D it forms the mean router distribution $\bar{p}_D = \frac{1}{|D|}\sum_{t\in D} p_t$, takes its top-$d_D$ entries as a document pool $\mathcal{D}_D$, and renormalizes each token's distribution onto that pool:

$$\hat{p}_t(i) = \begin{cases} p_t(i) / \sum_{j\in\mathcal{D}_D} p_t(j) & i \in \mathcal{D}_D, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

During pretraining d_D is sampled per document (here from $[8, 128]$), so the model learns to operate under many pool budgets. The stated motivation is modularity: at deployment one can ship a domain-specific expert subset.

3.2 Verification cost in speculative decoding

In SD with an independent drafter, each round proposes γ tokens; together with the current token the target verifies a block S of width $|S| = \gamma{+}1$ in a single forward pass. Every token in the block routes independently, so the set of expert weights that must be resident for the layer is

$$U(S) \;=\; \bigcup_{u\in S} \text{Top-}k(p_u), \qquad k \;\le\; |U(S)| \;\le\; \min(n,\, k\,|S|). \tag{2}$$

Autoregressive decoding corresponds to $|S|$=1 and $|U(S)| = k$. The quantity we care about is the *inflation* $|U(S)|/k$, which directly scales the expert-weight traffic of a verification round. We report $\mathbb{E}[|U(S)|]$ averaged over layers and rounds, following the metric used by prior MoE-SD work (McDanel et al., 2026; Liang et al., 2026).

Two objects that target the same cost. Window-level training acts on a sharing group S_w (a pool of size d_w over a length-W window). Serve-time budgeting acts on a per-round shortlist S_B (a top-B set over the draft-tree tokens being verified). Both aim to keep $|U(S)|$ small during verify, but they are not the same object: S_w reshapes the router's prior; S_B enforces a hard union cap on one verification forward.

Design goals. We target three properties simultaneously:

- **Native quality.** Unconstrained inference quality must not regress meaningfully relative to the base checkpoint.
- **Constrained sharing quality.** Quality under a small $|U(S)|$ must improve relative to the base prior. We probe two training-free proxies: forced window broadcast (d_w=k; Figure 1**(b)**) and a hard window mask with intermediate pool sizes $d_w \ge k$ (Figure 1**(c)**).
- **Serve-time budget.** At verification, an explicit cap $|U(S)| \le B$ must be enforceable without discarding the trained prior.

4 Motivation: Two Measurements

Models. We use two architecturally identical 1B-active / 14B-total MoE checkpoints trained on the same corpus and token budget: **StdMoE**, a standard top-k router, and **EMO**, trained with document-level pools (Eq. 1). Both have 16 layers, d_{model}=2048, 127 routed experts, one shared expert, and k=8.

Protocol. Verification statistics use the first 16 HumanEval (Chen et al., 2021) prompts with an EAGLE-3 (Li et al., 2025) chain drafter, `max_tokens`=128, and $\gamma \in \{1, 2, 3, 4\}$; we log $|U(S)|$ per layer per round. Quality uses the lm-evaluation-harness (Gao et al., 2023) with fixed seed 42; we report an 11-task mean (AVG) over MMLU (Hendrycks et al., 2021), CEval (Huang et al., 2023), ARC-Easy/Challenge (Clark et al., 2018), GSM8K (Cobbe et al., 2021), LAMBADA (Paperno et al., 2016), HellaSwag (Zellers et al., 2019), BBH (Suzgun et al., 2022), WinoGrande (Sakaguchi et al., 2021), PIQA (Bisk et al., 2020), and BoolQ (Clark et al., 2019). Unless noted otherwise, sharing constraints for quality are applied at *prefill only* (the prompt); autoregressive decode remains unconstrained. This is a proxy for sharing tolerance under a small $|U(S)|$, not a claim that decode-time tree verification has already been capped. We report two training-free probes (Figure 1): forced window broadcast (d_w=8=k) versus window length W, and a W=8 hard window mask versus pool size d_w. Both probes constrain all layers at prefill. They are not the deployment grid of Section 6.2, which constrains only layers 2–15. Full per-task companions appear in Appendix B.

4.1 MEASUREMENT 1: THE ROUTING PRIOR CONTROLS VERIFICATION INFLATION

Figure 1**(a)** shows that both models activate exactly k=8 experts per layer in autoregressive decoding, as expected, but diverge sharply under speculation. At γ=3 (verification width 4) StdMoE reaches 16.15 unique experts per layer—a 2.02× inflation that erases much of the benefit of sparse activation—whereas EMO reaches 11.74 (1.47×). The gap widens monotonically with γ (full values in Table 7). This confirms that verification inflation is not an immutable property of sparse routing but a consequence of *how the router was trained*. Document-level pooling, although defined over a unit far longer than a verification window, already concentrates routing enough to cut inflation by roughly a third.

4.2 MEASUREMENT 2: SHARING IS SURVIVABLE ONLY UNDER A POOLED PRIOR

Measurement 1 suggests pushing further. We consider operating points in which $|U(S)|$ is bounded *by construction* at inference time, with no retraining, to isolate what the pretrained routing prior can tolerate.

Forced window broadcast (d_w=k). Partition the sequence into windows of length W, and force every token in a window to use the same top-k expert set, chosen as the top-k of the window-mean router distribution $\bar{p}_S$. Under this intervention $|U(S)| = k$ exactly for any verification block contained in a window, so inflation is 1.0× regardless of γ. Figure 1**(b)** plots mean Avg quality against W. StdMoE drops sharply as soon as sharing is required and stays near chance as W grows; EMO degrades only slowly. Per-task numbers (Table 8) make the same split concrete: at W=32, StdMoE collapses to chance on MMLU (50.3 → 25.9) and near-chance on WinoGrande and HellaSwag, whereas EMO retains most of its MMLU (50.1 → 43.7) and is nearly unchanged on HellaSwag and PIQA. Degradation of EMO from W=8 to W=32 is small, indicating that the binding constraint is *sharing at all* rather than the specific window length.

Hard window mask (W=8, $d_w \geq k$). Forced broadcast is the extreme d_w=k. A milder probe keeps a window pool of size $d_w \geq k$ and lets tokens pick distinct experts inside it, so $|U(S)| \leq d_w$. Figure 1**(c)** fixes W=8 and varies $d_w \in \{8, 12, 16, 24\}$ (d_w=8 recovers broadcast). StdMoE recovers only as the pool grows (mean Avg 28.2 → 50.9), remaining well below its unconstrained baseline until d_w is large. EMO is nearly insensitive to d_w (51.7–53.4): once sharing is required, extra pool slots barely help. Full 11-task numbers are in Table 9.

Interpretation. Document-level pool training exposes the router, throughout pretraining, to a regime in which a token must be served by a restricted expert subset chosen from aggregate rather than per-token statistics. This appears to produce representations in which nearby tokens are genuinely servable by a common expert set—hence both the survival of zero-inflation broadcast (Figure 1**(b)**) and the flat d_w curve (Figure 1**(c)**). A standard router never experiences such a regime and learns routes that are individually near-optimal but collectively incompatible, so it needs a large pool before sharing is tolerable.

These measurements motivate a two-part response. First, if the prior determines tolerance to sharing—both at the zero-inflation extreme d_w=k (Figure 1**(b)**) and at intermediate pool sizes

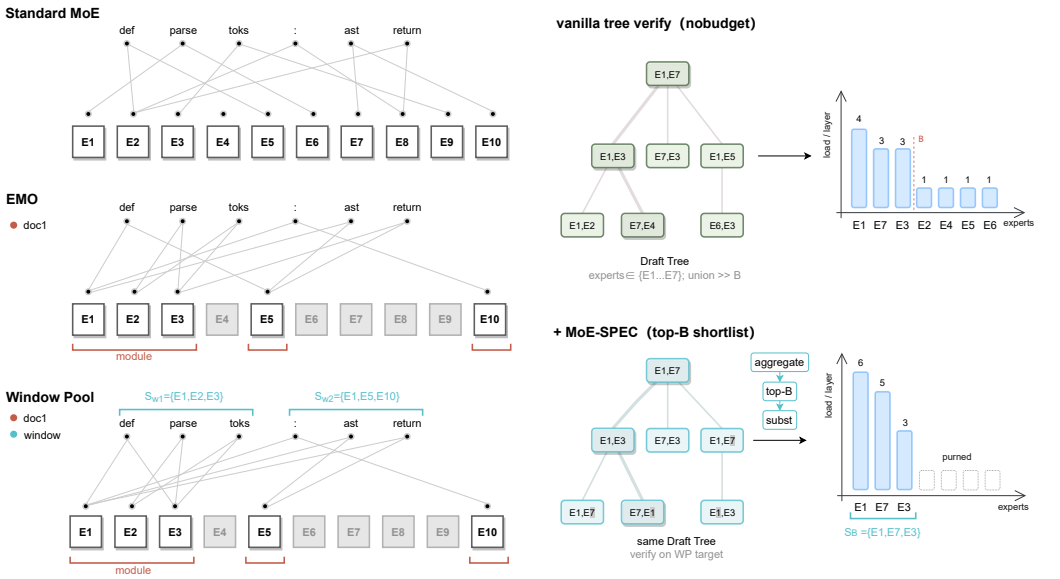


Figure 2: Method overview. **Left:** expert routing under Standard MoE, document-level modules (EMO), and Window Pool. Consecutive tokens share a window pool S_w of size d_w (e.g. $S_{w_1}=\{E_1, E_2, E_3\}$, $S_{w_2}=\{E_1, E_5, E_{10}\}$), nested inside the document module set. **Right:** tree verification on a Window Pool target. Without a serve-time budget the draft-tree tokens activate a large expert union (top). A standard MoE-Spec cap aggregates router mass over the tree (sum), keeps a top-B shortlist S_B (here $S_B=\{E_1, E_7, E_3\}$), and substitutes each token's top-k into S_B so the per-layer unique-expert count is capped at B (bottom). S_w is a training/deployment window pool; S_B is a per-verify-round shortlist and is not the same object.

(Figure 1(**c**))—we should *train* that prior on the sharing unit speculative decoding pays for—a verification-width window rather than a whole document (Section 5.1). Second, even a well-aligned prior need not keep the draft-tree union small on every verify round: training pools S_w and per-round shortlists S_B remain distinct, so a serve-time MoE-Spec cap is still useful when soft window routing leaves a large tree union (Section 5.4).

5 Method: Window Pool

5.1 Window-level expert pools

Let the sequence be partitioned into non-overlapping windows of length W, respecting document boundaries (a window never spans two documents). For a window S, define the window-mean router distribution and the window pool

$$\bar{p}_S = \frac{1}{|S|}\sum_{u\in S} p_u, \qquad \mathcal{D}_S = \text{Top-}d_w(\bar{p}_S), \qquad d_w \geq k. \tag{3}$$

The document pool of Eq. 1 is retained and applied first, so $\mathcal{D}_S \subseteq \mathcal{D}_D$; we never discard the modularity property that motivated EMO. We consider three ways to impose the window pool.

(a) Soft outside-mass penalty. Penalize probability mass that window tokens place outside the window pool:

$$\mathcal{L}_{\text{out}} = \frac{1}{|S|}\sum_{u\in S}\Big(1 - \sum_{i\in\mathcal{D}_S} p_u(i)\Big), \tag{4}$$

added to the training objective with weight λ_{win}. The pool $\mathcal{D}_S$ is computed from $\text{stopgrad}(\bar{p}_S)$ to prevent all tokens in a window from collapsing toward each other.

(b) Hard window mask. Mask and renormalize each token's distribution onto $\mathcal{D}_S$ before top-k selection, mirroring Eq. 1 at the window level. Tokens may still pick different experts within the pool, so $|U(S)| \leq d_w$.

(c) Forced window broadcast. Assign every token in the window the top-k of $\bar{p}_S$, i.e. $d_w = k$ with identical selection. This makes $|U(S)| = k$ exactly and is the training-time counterpart of the inference-time intervention studied in Section 4.2.

The full objective is

$$\mathcal{L} = \mathcal{L}_{\text{CE}} + \mu\, \mathcal{L}_{\text{LB}} + \lambda_{\text{win}}\, \mathcal{L}_{\text{out}}, \tag{5}$$

where $\mathcal{L}_{\text{LB}}$ is the load-balancing loss inherited from the base recipe. In our best configurations λ_{win}=0 and the constraint is imposed structurally by (b) or (c); Section 6.5 shows that adding the soft penalty on top is harmful.

5.2 Three ingredients that make it work

Naively training with a single fixed window configuration applied to all layers at every step degrades the model below the training-free baseline (Section 6.5). We identify three necessary modifications.

Budget randomization. Sample $W \sim \{8, 12, 16, 24, 32\}$ and $d_w \sim \{8, 12, 16, 24, 32\}$ independently per step (subject to $d_w \leq d_D$). A deployment does not know in advance the speculation depth or the memory budget it will face, and a model trained at a single (W, d_w) overfits to it. Randomization produces one checkpoint that can be operated across the budget range, analogous to how EMO randomizes d_D.

Layer selection. Apply the window constraint only to layers $2 \ldots 15$, leaving layers 0–1 unconstrained. Early layers carry token-identity information whose routing is least amenable to clustering across a window; constraining them costs quality without materially reducing traffic.

Native-step interleaving. On a fraction of steps (20–25%) disable the window constraint entirely while keeping the document pool active. This preserves the model's ability to route freely, so that native quality is not sacrificed for gains under forced sharing. It also serves as an implicit regularizer against the collapse mode in which the router learns to make $\bar{p}_S$ trivially peaked.

5.3 Training setup

We continue pretraining from the released EMO 1B/14B checkpoint on the OLMoE-mix-0924 corpus (Muennighoff et al., 2024) on a 5B-token schedule, with sequence length 2048, global batch 256 sequences, and learning rate 3×10^{-6} on 4×8 Ascend NPUs. Document boundaries are preserved so that window construction never merges documents. The document pool remains active with $d_D \sim [8, 128]$, matching the base recipe. Full hyperparameters are in Appendix A.

All Window Pool variants in Section 6 branch from a common ancestor: the CT-only control is trained first, and the window-constrained runs are initialized from it. The comparison in Table 5 is therefore against the exact checkpoint each variant started from, which isolates the effect of the window constraint from that of the additional data.

5.4 Serve-time budget: MoE-Spec on a Window Pool target

Window Pool reshapes the router's prior so that tokens in a length-W window are more likely to be servable by a shared pool S_w. It does not, by itself, guarantee a hard bound on $|U(S)|$ when the verifier consumes an entire draft tree in one forward: tree tokens need not lie in one training window, and even within a window soft or hard-masked routing can still activate up to d_w distinct experts. The hard per-round bound is enforced at serve time by standard MoE-Spec (McDanel et al., 2026) (Figure 2, right).

Procedure. For each MoE layer and each verification round over draft-tree positions $\mathcal{T}$:

Table 1: 11-task mean Avg (%) under Native and under prefill-only window constraints on layers 2–15. d_w=8 is forced broadcast; d_w>8 is a hard window mask. Bold marks the best among StdMoE / EMO / Window Pool in each constrained column (Native is unconstrained).

	Native	W=8				W=16				W=32			
Model	—	d_w=8	12	16	24	d_w=8	12	16	24	d_w=8	12	16	24
StdMoE	55.7	33.4	43.6	47.8	51.8	31.8	40.2	44.9	49.9	30.1	38.0	42.4	48.4
EMO	55.2	52.6	54.4	54.3	54.4	52.1	54.0	54.5	54.5	51.8	53.6	54.1	54.8
Window Pool	55.3	**53.5**	**54.8**	**55.0**	**55.2**	**52.8**	**54.5**	**55.0**	**54.9**	**52.6**	**54.1**	**54.8**	**55.1**

1. **Aggregate.** Sum router probabilities over $t \in \mathcal{T}$ to obtain a tree-level score vector $m = \sum_{t \in \mathcal{T}} p_t$.
2. **Shortlist.** Form $S_B = \text{Top-}B(m)$ with budget $B \geq k$.
3. **Substitute.** For each $t \in \mathcal{T}$, replace the unrestricted top-k of p_t by the top-k of p_t restricted to S_B (renormalize on S_B if needed). The layer then activates at most B unique routed experts on that round, i.e. $|U(\mathcal{T})| \leq B$.

This is the aggregate→top-B→substitute pattern of verifier-side budgets such as MoE-Spec (McDanel et al., 2026). We do not claim a new ranking criterion here; the contribution of this paper is the trained prior (Section 5.1), and MoE-Spec is the off-the-shelf budget enforcer that the prior makes survivable. The distinct roles of the training pool S_w and the per-round shortlist S_B are defined in Section 3; we keep them intentionally separate.

Why the prior matters under a fixed B. When the trained prior already concentrates mass on a few experts, the top-B shortlist tends to overlap the experts the router would have chosen, so substitution costs less accuracy for a given B. When the tree union is still large, MoE-Spec enforces the traffic budget that Window Pool only encourages. The two therefore compose: training improves the operating point under sharing; MoE-Spec makes the budget hard at verify time. Tables 4 and 3 evaluate this composition at B=13 and B=8.

6 Experiments

6.1 Evaluation protocol

Each checkpoint is evaluated under unconstrained routing and under training-free sharing constraints applied at *prefill only* on layers 2–15 (matching the Window Pool training layer range); autoregressive decode remains unconstrained. We report an 11-task mean (Avg) over MMLU, CEval, ARC-Easy/Challenge, GSM8K, LAMBADA, HellaSwag, BBH, WinoGrande, PIQA, and BoolQ.

Native document pool only (no window constraint).

Forced (d_w=k) window broadcast of length W: every token in the window shares one top-k set (zero inflation by construction).

Hard mask (d_w>k) tokens may pick distinct experts inside a pool of size d_w, so $|U(S)| \leq d_w$.

Unless noted, Window Pool denotes the randomized continued-pretraining checkpoint of Section 5 (randomized W/d_w, layers 2–15, 20% native steps).

6.2 Main results: quality under fixed (W, d_w)

Table 1 sweeps the deployment grid $W \in \{8, 16, 32\} \times d_w \in \{8, 12, 16, 24\}$. Table 2 expands Native and two deploy points to all 11 tasks (plus Avg). Both tables share the same L2–15 prefill protocol.

We highlight three results.

Table 2: Full 11-task breakdown (%) at Native and two deploy points (same protocol as Table 1): W=32, d_w=8 (tight long-window broadcast) and W=8, d_w=12 (mild short-window mask). Bold: best Avg among the three models within each setting. Abbreviations: CE=CEval, AE/AC=ARC-Easy/Challenge, GSM=GSM8K, LAM=LAMBADA, HS=HellaSwag, WG=WinoGrande.

Model	Setting	Avg	MMLU	CE	AE	AC	GSM	LAM	HS	BBH	WG	PIQA	BoolQ
StdMoE	Native	55.7	50.3	31.5	81.5	52.4	10.4	72.8	58.9	30.8	72.7	79.4	71.7
EMO	Native	55.2	50.1	30.8	82.0	53.6	9.1	71.7	57.2	30.3	69.7	79.3	73.6
Window Pool	Native	55.3	49.9	32.9	82.3	53.8	9.4	71.6	57.1	28.5	70.1	79.4	73.6
StdMoE	W=32, d_w=8	30.1	25.5	24.2	41.8	20.9	1.1	4.0	31.5	20.6	52.5	60.9	48.1
EMO	W=32, d_w=8	51.8	45.1	29.1	80.3	51.0	7.2	62.9	56.0	28.2	62.7	77.8	69.0
Window Pool	W=32, d_w=8	**52.6**	45.1	29.7	80.5	50.8	6.8	66.6	55.4	29.0	65.4	78.9	70.2
StdMoE	W=8, d_w=12	43.6	36.0	24.9	70.9	38.5	3.9	38.1	49.0	26.8	56.6	72.9	62.0
EMO	W=8, d_w=12	54.4	48.3	31.7	82.7	53.8	8.8	66.1	58.3	30.1	66.8	78.7	72.8
Window Pool	W=8, d_w=12	**54.8**	49.5	31.9	83.0	53.4	7.6	69.8	57.5	30.4	68.4	79.2	71.8

Table 3: Quality (%) under Native and under MoE-Spec B=8 on all MoE layers. Five of the 11 tasks. Native is the same as Table 2. B=8 is 1 shared + 7 routed, uniform aggregation, applied to every prefill and decode forward.

	MMLU		GSM8K		LAMBADA		WinoGrande		BoolQ	
Model	Nat.	B=8	Nat.	B=8	Nat.	B=8	Nat.	B=8	Nat.	B=8
StdMoE	50.3	25.5	10.4	1.1	72.8	0.5	72.7	50.9	71.7	45.8
EMO	50.1	39.8	9.1	6.8	71.7	59.0	69.7	56.7	73.6	60.6
Window Pool	49.9	40.6	9.4	5.9	71.6	63.6	70.1	56.0	73.6	61.9

Native quality is essentially preserved. Window Pool matches EMO and StdMoE on the unconstrained 11-task Avg (55.3 vs. 55.2 / 55.7), with MMLU within 0.4 points of the EMO base.

Under every fixed budget, Window Pool beats EMO. Across all twelve constrained cells in Table 1, Window Pool improves mean Avg over EMO (typical gap +0.5–1.0; largest at tight broadcast, e.g. +0.9 at W=8, d_w=8). StdMoE remains far below both whenever sharing is required.

Gains concentrate where the prior is most stressed. At zero-inflation broadcast (d_w=8), Table 2 shows Window Pool recovering Avg and, at W=32, a clear WinoGrande lift over EMO (62.7 → 65.4) with matched MMLU (45.1), plus LAMBADA (62.9 → 66.6) and BoolQ (69.0 → 70.2). The milder W=8, d_w=12 cell keeps the same ranking with a smaller Avg gap.

6.3 Quality under a hard verify budget

Table 3 places the same three routers under a hard MoE-Spec B=8 budget on all MoE layers and evaluates five of the 11 tasks (MMLU, GSM8K, LAMBADA, WinoGrande, BoolQ). The Native column repeats the unconstrained scores from Table 2. The B=8 column is the same budget as the B=8 rows of the end-to-end SD table (Table 4, Section 6.4), applied here with the lm-evaluation-harness rather than the SD tree.

The standard router collapses; the pooled routers do not. Under B=8, StdMoE falls to near-chance on MMLU (25.5) and LAMBADA (0.5). EMO and Window Pool both retain most of their quality, and Window Pool is above EMO on MMLU (40.6 vs. 39.8), LAMBADA (63.6 vs. 59.0), and BoolQ (61.9 vs. 60.6), with EMO slightly ahead on GSM8K and WinoGrande. This is the same ranking as the deployment grid (Table 1): the trained prior, not the serve-time cap, is what makes the budget survivable.

Table 4: End-to-end speculative decoding on HumanEval-164. Protocol: AcceptMoE-style tree, `ignore_eos`=$True$. τ: mean acceptance length over verify rounds. pass@1: 164 problems. Traffic: $\sum_{\ell=0}^{15} |U_\ell|$. Each model at three budgets: Native (no B), MoE-Spec B=13, MoE-Spec B=8, all on layers 0–15 (uniform). B includes the shared expert.

System	Budget	τ	tok1	pass@1 (164)	Traffic
StdMoE + EAGLE-3	Native	3.95	0.81	17/164	655
StdMoE + MoE-Spec	B=13	3.42	0.76	12/164	208
StdMoE + MoE-Spec	B=8	3.15	0.72	6/164	128
EMO + EAGLE-3	Native	3.74	0.81	17/164	366
EMO + MoE-Spec	B=13	3.64	0.80	18/164	205
EMO + MoE-Spec	B=8	3.51	0.78	14/164	128
Window Pool + EAGLE-3	Native	3.84	0.81	23/164	506
Window Pool + MoE-Spec	B=13	3.58	0.79	18/164	207
Window Pool + MoE-Spec	B=8	3.60	0.81	16/164	128

6.4 End-to-end SD: robustness under a hard budget

The deployment grid and the B=8 quality table above establish that the trained prior is what makes a hard expert budget survivable. We now check that this robustness carries to end-to-end speculative decoding rather than only to single-forward quality probes. Table 4 reports SD metrics on HumanEval-164 under a shared AcceptMoE-style tree protocol (64 draft tokens, depth 5, top-k=10), `max_new_tokens`=128, `ignore_eos`=$True$. Each target uses its own EAGLE-3 tree drafter. Each model is evaluated at three budgets: unconstrained Native, matched-B=13 MoE-Spec, and B=8 MoE-Spec (uniform router aggregation + substitution on all MoE layers 0–15; B includes the shared expert, so B=8 is 1 shared + 7 routed). τ and tok1 are tree acceptance statistics averaged over verify rounds; pass@1 is on the 164 HumanEval problems; Traffic is the profiled $\sum_{\ell=0}^{15} |U_\ell|$ under the same protocol. Window Pool uses the H1 drafter; StdMoE and EMO use their own drafters.

Under a hard budget, Window Pool does not regress where StdMoE collapses. At B=8 all three rows have identical profiled traffic 128, so the acceptance and pass@1 differences reflect what the trained prior can withstand, not a looser budget. StdMoE falls from τ3.95 / 17/164 to 3.15 / 6/164; Window Pool stays at 3.60 / 16/164. The same pattern appears in the quality table above (MMLU 50.3 → 25.5 for StdMoE vs. 49.9 → 40.6 for Window Pool).

Against the document-pool base, Window Pool does not win SD. At B=13, EMO (τ3.64, 18/164) is at or above Window Pool (3.58, 18/164). At B=8, Window Pool is +0.09 τ and +2/164 over EMO, within the binomial noise on 164 problems. Uncapped, Window Pool has the highest pass@1 (23/164 vs. 17/164) but also the highest traffic (506 vs. 366); we do not claim Window Pool routes to fewer experts naturally. The contribution of Window Pool is therefore constrained-sharing robustness, not SD speedup over the document-pool base.

Traffic is the speed proxy. The harness speedup on this research harness does not offload experts and therefore tracks τ rather than unique-expert traffic, so we report Traffic as the speed proxy. Converting Traffic into wall-clock under an offloading stack is deferred to Section 7.

6.5 Ablations

The deployment-grid and end-to-end results above use a single randomized Window Pool checkpoint. Table 5 compares earlier training variants on a four-task diagnostic suite (HellaSwag, PIQA, WinoGrande, MMLU); step counts are *not* matched across rows, and none of the rows is that randomized checkpoint.

Table 5: Earlier training variants on a four-task diagnostic suite (Native vs. forced $W{=}32$ at prefill). These rows are not the randomized checkpoint in Tables 4 and 1–2, and they are not at a matched token budget. HellaSwag and PIQA are `acc_norm`, not the scores in the 11-task tables.

	Native				Forced $W{=}32$			
Model	HS	PIQA	WG	MMLU	HS	PIQA	WG	MMLU
EMO (base)	77.23	80.25	69.69	50.15	74.53	78.07	61.40	43.74
CT-only (control)	77.33	79.87	70.40	49.94	74.59	78.40	62.35	43.44
WP-hard (all layers)	76.77	79.16	69.46	49.84	73.38	78.02	63.77	44.99
WP-hard, L2–15	77.25	79.76	69.85	49.87	73.77	79.38	63.61	**45.13**
WP-hard, 25% native	**77.40**	79.71	**69.85**	**50.08**	72.74	78.18	62.27	44.36
WP-bcast, L2–15, 20% native	76.94	79.65	69.61	49.27	72.87	**80.03**	**65.19**	44.23

Continued training alone does not help. The CT-only control in Table 5 matches the base checkpoint in both modes (MMLU 43.44 vs. 43.74 forced). Gains in the Window Pool rows are therefore attributable to the window constraint rather than to additional tokens or optimizer state.

Naive always-on constraint is harmful. In a run not shown in Table 5, training with forced broadcast at a fixed $W{=}32$ on all layers for the full 5B tokens, with no randomization and no native steps, yields forced-mode MMLU of 40.59 and HellaSwag of 70.50—substantially *worse* than applying the same constraint to the untrained base checkpoint (43.74 / 74.53). Simply training on the deployment constraint is counterproductive; the model appears to solve the training objective by degrading its router rather than by learning clusterable representations. This is the negative result that motivates the three ingredients of Section 5.2.

Adding the soft outside-mass penalty hurts. Setting $\lambda_{\text{win}}{=}0.02$ on top of the hard mask degrades both modes relative to $\lambda_{\text{win}}{=}0$ (native MMLU 49.70 vs. 49.84; forced MMLU 44.67 vs. 44.99, and forced WinoGrande 61.88 vs. 63.77). Once the constraint is imposed structurally, an additional soft penalty on outside mass is redundant and over-regularizes.

Soft-only training does not reach the target. In an earlier phase we trained with the soft penalty alone (Eq. 4, $\lambda_{\text{win}} \in \{0.01, 0.025, 0.05\}$, no mask) and measured unique experts directly. Relative to a matched $\lambda{=}0$ control at 15.46 unique experts per layer at $\gamma{=}3$, the soft penalty reduced inflation substantially (to 10.37–12.29 depending on λ and step), confirming the loss is doing something. However, the outside-mass term plateaued around 0.7–0.8 unscaled, indicating that the router never actually concentrated inside the pool, and only the shortest runs improved on the original base checkpoint's 11.74. Structural constraints therefore dominate; we omit the soft-only runs as a full table.

7 Limitations

Harness speedup is not offloading wall-clock. Table 4 reports tree/AR tok/s on a research harness and matched-B MoE-Spec traffic on StdMoE, EMO, and Window Pool, not deployment latency under expert offloading. Whether those traffic cuts convert into wall-clock gains depends on the serving stack and is not measured here.

Window Pool does not improve end-to-end SD over the document-pool base. At matched budget, EMO matches or exceeds Window Pool on HumanEval-164 SD (τ3.64 vs. 3.58 at $B{=}13$; within binomial noise at $B{=}8$), and Window Pool's uncapped traffic is higher, not lower (506 vs. 366). The contribution of Window Pool is therefore constrained-sharing robustness and the deployment-grid win over EMO (Table 1), not SD speedup over the document-pool base. A serve-time cap on a Window Pool target does not by itself form a better end-to-end Pareto frontier than the same cap on the document-pool base.

Single model scale and family. All results use one 1B-active / 14B-total MoE. Whether the mechanism transfers to larger models, to different expert counts, and—critically—whether a standard MoE can be converted into a window-tolerant one by continued training alone (rather than requiring document-pool pretraining from scratch) remains open.

Ablation variants are not at a matched token budget. The variants in Table 5 branch from a shared ancestor but are reported at different numbers of subsequent steps: the all-layer hard-mask run is substantially further along its schedule than the layer-restricted and native-interleaved runs, which are reported early. Rankings among variants should therefore be treated as provisional.

8 Conclusion

Speculative decoding and Mixture-of-Experts interact badly because parallel verification activates the union of per-token expert sets. We showed that this interaction is governed by the routing prior rather than by sparse routing per se: an MoE pretrained with document-level expert pools not only inflates less under speculation, but survives a training-free intervention that forces an entire 32-token window through a single expert set—an intervention that reduces an otherwise identical standard MoE to chance accuracy. Building on this, Window Pool aligns the training-time sharing group with the verification window, and with budget randomization, layer selection, and native-step interleaving improves constrained 11-task Avg on a fixed-(W, d_w) deployment grid while preserving unconstrained quality. Under a hard per-verify-round MoE-Spec budget, the trained prior is what makes the budget survivable: at B=8, StdMoE collapses on quality (MMLU $50.3 \rightarrow 25.5$, LAMBADA $72.8 \rightarrow 0.5$) and on HumanEval-164 speculative decoding ($\tau 3.95 \rightarrow 3.15$, $17 \rightarrow 6/164$), while Window Pool retains MMLU 40.6, LAMBADA 63.6, and $\tau 3.60$ / $16/164$, all at the same profiled traffic 128. Against the document-pool base, Window Pool does not win speculative decoding outright—at B=13 EMO matches or exceeds it, and at B=8 the gap is within task-count noise—but it preserves the document-pool base's budget robustness while keeping unconstrained quality. Wall-clock gains under expert offloading remain an important follow-up.

AI use statement

In this work, we used generative AI tools primarily for language polishing, literature discovery, consistency checking, and adversarial review of the manuscript. The research methodology, mathematical formulation, experimental design, implementation, analysis, and conclusions were developed and verified by the authors. All AI-assisted text was carefully reviewed and revised by the authors to ensure technical correctness, consistency with the implementation and recorded experimental results, and faithful representation of our contributions. Generative AI was not used to create, modify, or fabricate experimental measurements. We take full responsibility for the final content of this work, including all claims, citations, figures, tables, and other artifacts produced with the assistance of generative AI.

Ethics statement

This work studies routing priors for Mixture-of-Experts speculative decoding, with the aim of reducing expert-weight traffic during verification. Our experiments use publicly available pretrained models, the OLMoE-mix-0924 text corpus, and established evaluation benchmarks, and do not involve human subjects, user studies, or the collection of private or sensitive personal information. The proposed method is intended to reduce the computational and memory costs of deploying MoE language models under speculative decoding. Improved inference efficiency may lower deployment barriers, but it may also make the existing capabilities and associated risks of the underlying models more accessible. Moreover, window-sharing and expert-budget constraints may affect different tasks or capabilities unevenly, and therefore budgeted models should be evaluated under the safety requirements appropriate to their intended applications. Our method does not introduce new training data or capabilities beyond continued pretraining of the original models on a public corpus. The limitations, licenses, biases, and potential misuse risks of the underlying models and datasets remain applicable. We take responsibility for the ethical considerations associated with the methodology, experiments, and reported results.

Reproducibility statement

We provide the model configuration, window-pool construction, randomization ranges for W and d_w, layer range, native-step fraction, optimization hyperparameters, and training-data composition in Section 5 and Appendix A. Section 4 specifies the expert-union measurement (drafter, prompt set, γ range, and token budget) and the quality-benchmark protocol (harness, tasks, and seed). Sections 6.2, 6.3, and 6.4 specify the deployment grid, the hard-budget quality evaluation, and the end-to-end speculative-decoding protocol, including the MoE-Spec budget and the tree settings. The pretrained models, training corpus, and evaluation benchmarks used in our experiments are publicly available and properly cited. Baseline methods are reproduced from their published implementations and configurations whenever available; differences in continued pretraining or evaluation settings are documented and evaluated under matched constraints. Code, configurations, and training and evaluation commands will be released to help reproduction of the reported results.

References

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: Reasoning about physical commonsense in natural language. In *AAAI Conference on Artificial Intelligence*, 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 *International Conference on Machine Learning (ICML)*, 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. *arXiv preprint arXiv:2602.06036*, 2026.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2019.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

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

DeepSeek-AI. Deepseek-V3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

DeepSeek-AI. DSpark: Confidence-scheduled speculative decoding with semi-autoregressive generation. *arXiv preprint arXiv:2607.05147*, 2026.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, et al. A framework for few-shot language model evaluation, 2023. EleutherAI lm-evaluation-harness.

Yuning Han, Yangchenchen Jin, Dylan Zhao, and Jingwei Sun. CATS: Cascaded adaptive tree speculation for memory-limited LLM inference acceleration. *arXiv preprint arXiv:2605.11186*, 2026.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations (ICLR)*, 2021.

Bairu Hou, Qibin Chen, Jianyu Wang, Guoli Yin, Chong Wang, Nan Du, Ruoming Pang, Shiyu Chang, and Tao Lei. Instruction-following pruning for large language models. *arXiv preprint arXiv:2501.02086*, 2025.

Haochen Huang, Shengxuan Qiu, and Meng Li. S2-MoE: Enabling efficient self-speculative decoding for mixture-of-experts on edge devices. *arXiv preprint arXiv:2608.15018*, 2026. URL https://arxiv.org/abs/2608.15018.

Yuzhen Huang, Yuzhuo Bai, Zhiheng Zhu, Junhao Zhang, Jinghan Zhang, Junteng Su, Junteng Liu, Chuancheng Lv, YiKun Zhang, Jiayi Lei, Yao Fu, Maosong Sun, and Junxian He. C-Eval: A multi-level multi-discipline chinese evaluation suite for foundation models. *arXiv preprint arXiv:2305.08322*, 2023.

Ali Kayyam. Sticky routing: Training MoE models for memory-efficient inference. *arXiv preprint arXiv:2607.08780*, 2026.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning (ICML)*, 2023.

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

Shuang Liang, Hao Chen, Zhiwen Mo, Qianzhou Wang, Guoyu Li, Lingxiao Ma, and Wayne Luk. AcceptMoE: Commitment-weighted self-sizing verifier expert sets for efficient MoE speculative decoding. *arXiv preprint arXiv:2608.02989*, 2026.

Bradley McDanel, Steven Li, Sruthikesh Surineni, and Harshit Khaitan. MoE-Spec: Expert budgeting for efficient speculative decoding. *arXiv preprint arXiv:2602.16052*, 2026.

Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Jacob Morrison, et al. OLMoE: Open mixture-of-experts language models. *arXiv preprint arXiv:2409.02060*, 2024.

Lehan Pan, Ziyang Tao, Ruoyu Pang, Xiao Wang, Jianjun Zhao, and Yanyong Zhang. Making every verified token count: Adaptive verification for MoE speculative decoding. *arXiv preprint arXiv:2605.00342*, 2026.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quang Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Raffaella Bernardi, João Ferreira, et al. The LAMBADA dataset: Word prediction requiring a broad discourse context. *arXiv preprint arXiv:1606.06031*, 2016.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9), 2021.

Anish Saxena, Po-An Tsai, Hritvik Taneja, Aamer Jaleel, and Moinuddin Qureshi. Utility-driven speculative decoding for mixture-of-experts. *arXiv preprint arXiv:2506.20675*, 2025.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations (ICLR)*, 2017.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, and Jason Wei. Challenging BIG-Bench tasks and how chain-of-thought reasoning can solve them. *arXiv preprint arXiv:2210.09261*, 2022.

Ryan Wang, Akshita Bhagia, and Sewon Min. EMO: Pretraining mixture of experts for emergent modularity. *arXiv preprint arXiv:2605.06663*, 2026.

Jincheng Xie, Runheng Liu, Heyan Huang, Yawen Ling, Hanbin Dai, Yu Zheng, and Wen Hu. Less experts, faster decoding: Cost-aware speculative decoding for mixture-of-experts. *arXiv preprint arXiv:2607.12696*, 2026.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2019.

A HYPERPARAMETERS

Table 6: Continued pretraining configuration.

Setting	Value
Base checkpoint	EMO 1B-active / 14B-total
Layers / d_{model}	16 / 2048
Routed experts / shared	127 / 1
Top-k	8
Corpus	OLMoE-mix-0924
Tokens	5×10^9
Sequence length	2048
Global batch	256 sequences (5.24×10^5 tokens/step)
Learning rate	3×10^{-6}, warmup 0, linear decay
Optimizer	AdamW, $\beta = (0.9, 0.95)$, wd 0.1, clip 1.0
Precision / sharding	bf16 params, fp32 reduce, FSDP full
Hardware	4×8 Ascend NPU
Document pool d_D	$\sim [8, 128]$ (unchanged from base recipe)
Window length W	$\sim \{8, 12, 16, 24, 32\}$ per step
Window pool d_w	$\sim \{8, 12, 16, 24, 32\}$ per step, $d_w \leq d_D$
Constrained layers	2–15 (layers 0–1 native)
Native-step fraction	0.20–0.25
λ_{win}	0 (structural constraint only)

Windows are non-overlapping and never span a document boundary; document boundaries are taken from the corpus EOS positions. The window pool is computed from the document-pool-renormalized distribution, so $\mathcal{D}_S \subseteq \mathcal{D}_D$ always holds. The shared expert is exempt from the window constraint and is always active.

B MOTIVATION TABLES

Table 7 accompanies Figure 1**(a)**. Tables 8 and 9 give the full 11-task numbers behind Figure 1**(b)**–**(c)** (prefill-only; all layers). Avg is the same mean used in the figure y-axes. d_w=8 in Table 9 reuses the W=8 forced-broadcast rows from Table 8.

Table 7: Mean unique experts per layer during EAGLE-3 verification (HumanEval-16, `max_tokens`=128). Both models have k=8 and 16 layers. AR denotes autoregressive decoding (verification width 1). Inflation is relative to AR. Companion to Figure 1**(a)**.

Target	AR	γ=1	γ=2	γ=3	γ=4	Inflation @ γ=3
StdMoE	8.00	11.35	13.98	16.15	18.12	2.02×
EMO	8.00	**9.75**	**10.86**	**11.74**	**12.47**	**1.47×**

Table 8: Companion to Figure 1**(b)**: training-free forced window broadcast (d_w=k) at prefill, all layers. Full 11-task scores (%) plus Avg. Chance is 25.0 for MMLU and 50.0 for WinoGrande. Abbreviations as in Table 2.

Model	W	Avg	MMLU	CE	AE	AC	GSM	LAM	HS	BBH	WG	PIQA	BoolQ
StdMoE	Native	55.7	50.3	31.5	81.5	52.4	10.4	72.8	58.9	30.8	72.7	79.4	71.7
	2	38.4	26.5	24.1	62.0	32.2	2.3	27.3	41.5	22.8	54.1	74.1	56.0
	4	31.7	24.4	23.1	45.8	21.7	1.8	10.0	29.3	21.3	52.6	66.9	51.9
	8	28.2	24.7	23.5	31.1	21.4	0.5	3.5	25.5	21.1	48.6	62.7	47.6
	16	28.3	25.2	25.9	30.6	21.8	0.8	1.7	25.9	21.0	51.6	59.1	47.9
	32	27.5	25.9	24.4	32.9	20.7	1.0	1.0	26.1	20.1	50.2	55.9	44.0
EMO	Native	55.2	50.1	30.8	82.0	53.6	9.1	71.7	57.2	30.3	69.7	79.3	73.6
	2	51.8	44.9	31.2	81.6	52.2	6.8	62.2	56.3	27.7	64.4	73.6	69.5
	4	51.4	45.5	30.7	81.7	51.8	5.2	59.5	56.3	28.2	64.4	75.5	67.0
	8	51.7	45.0	31.7	81.4	50.9	5.8	60.6	56.8	27.1	64.7	76.9	67.8
	16	51.0	44.6	30.7	81.1	51.0	4.7	59.8	56.2	26.1	62.7	77.8	66.6
	32	50.8	43.7	29.2	80.0	50.7	5.5	60.6	55.7	26.3	61.4	78.5	67.7

Table 9: Companion to Figure 1**(c)**: W=8 hard window mask vs. pool size d_w at prefill, all layers. d_w=8 is forced broadcast (same rows as W=8 in Table 8). Full 11-task scores (%) plus Avg.

Model	d_w	Avg	MMLU	CE	AE	AC	GSM	LAM	HS	BBH	WG	PIQA	BoolQ
StdMoE	8	28.2	24.7	23.5	31.1	21.4	0.5	3.5	25.5	21.1	48.6	62.7	47.6
	12	39.2	27.6	23.3	66.0	32.8	2.5	29.7	40.6	23.3	53.2	74.0	58.3
	16	45.8	34.9	23.9	74.0	42.9	5.8	46.9	50.6	28.3	59.2	75.5	61.7
	24	50.9	44.1	27.9	78.6	47.7	6.9	59.9	56.5	30.1	64.4	78.7	65.2
EMO	8	51.7	45.0	31.7	81.4	50.9	5.8	60.6	56.8	27.1	64.7	76.9	67.8
	12	53.0	47.4	31.1	82.5	52.9	5.7	62.3	57.6	29.3	66.5	77.0	70.3
	16	53.4	48.0	30.1	82.6	53.7	6.4	62.2	57.4	30.2	68.2	77.1	71.5
	24	53.4	48.2	30.9	82.4	53.3	6.9	63.0	57.0	29.0	68.4	76.0	72.4