Simpler and Faster Language Models with Multi-Token Prediction via Continual Pretraining

Anonymous ACL submission

Abstract

Multi-token prediction (MTP) improves representation learning and enables speculative decoding, but existing approaches typically require pretraining from scratch or adding auxiliary prediction modules. We propose SLIM (Scalar-gated Layer Integration for Multi-token prediction), a lightweight method that introduces MTP capability to pretrained next-token language models during continual pretraining. For each additional future-token target, SLIM learns a scalar-gated convex combination of layer-wise hidden states and decodes the resulting branch representation with the model's existing unembedding matrix. This design adds only $N \times k$ scalar parameters for a model with N layers and k additional future-token targets. Across five pretrained dense and mixture-of-experts models, SLIM consistently increases acceptance length in self-speculative decoding, while largely preserving standard language-modeling benchmark performance under the same continual-pretraining budget as an NTP baseline. Analysis of the learned gates shows offset-specific reliance on intermediate layers, suggesting that pretrained next-token models already contain exploitable future-token information. SLIM offers a simple way to elicit MTP capability from existing checkpoints with negligible architectural overhead.

1 Introduction

Multi-token prediction (MTP) (Gloeckle et al., 2024) extends the standard next-token prediction (NTP) objective by training a language model to predict multiple future tokens from a single context. This objective has been shown to improve both inference efficiency through self-speculative decoding and representation quality, and has been adopted in production-scale systems such as DeepSeek-V3 (DeepSeek-AI et al., 2024) and GLM-5 (GLM-5 Team, 2026). However, existing MTP approaches are typically integrated during pretraining from scratch, which is poorly aligned with the common practice of building on strong pretrained checkpoints. Post-hoc acceleration methods such as Medusa (Cai et al., 2024), EAGLE (Li et al., 2024b), and LayerSkip (Elhoushi et al., 2024) address this gap, but introduce auxiliary prediction modules, millions of additional parameters, or dedicated training recipes that increase deployment complexity.

This raises a practical question: can MTP capability be introduced into a pretrained language model without repeating pretraining from scratch or substantially modifying the architecture? Recent findings suggest that this is possible. Although NTP-trained models are optimized for next-token prediction, prior analyses indicate that they already encode future-token information in their intermediate representations. Future Lens (Pal et al., 2023) demonstrates that intermediate Transformer layers linearly encode signals about tokens several steps ahead, while Samragh et al. (2025) extract future-token predictions from NTP-trained checkpoints. Relatedly, Aynetdinov and Akbik (2025) show that, even within from-scratch pretraining, MTP benefits from prior NTP exposure: a forward curriculum that begins with NTP and gradually transitions to MTP outperforms training with the MTP objective from the very first step, particularly for smaller language models. This perspective reframes MTP not as a capability that must be built into a model from the beginning, but as one that can be exposed and refined after pretraining.

We propose SLIM (**S**calar-gated **L**ayer **I**ntegration for **M**ulti-token prediction), a lightweight method for eliciting MTP capability during CPT. As illustrated in Figure 1(a), SLIM constructs each additional future-token branch by learning a scalar-gated convex combination of the hidden states from all N Transformer layers. The resulting branch representations are decoded through the model's existing unembedding matrix,

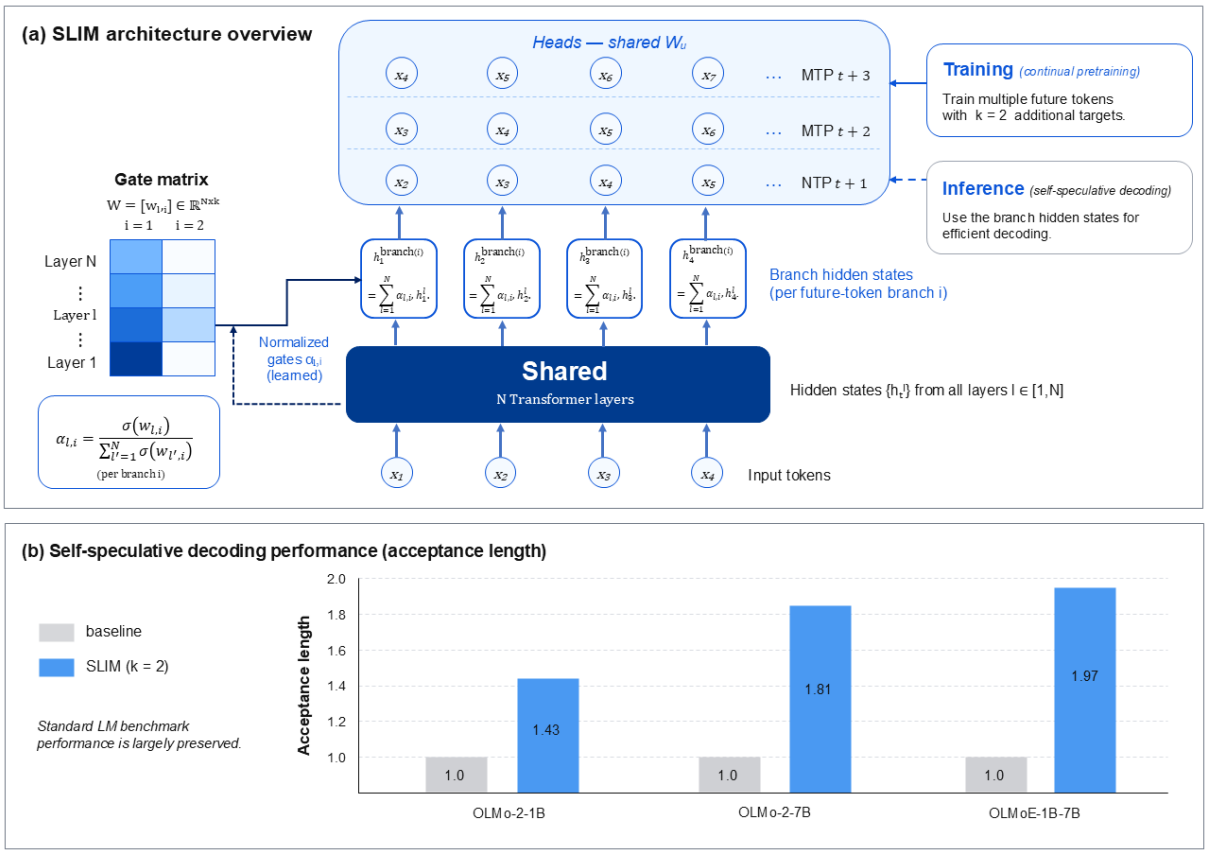


Figure 1: Overview of SLIM. (a) During continual pretraining, SLIM learns scalar-gated layer combinations to predict multiple future tokens while reusing the model's existing unembedding matrix. (b) SLIM accelerates inference via self-speculative decoding, increasing mean acceptance length across representative backbones while largely preserving standard language-modeling benchmark performance.

introducing no additional output projection or auxiliary LM head. Thus, for a model with N layers and k additional future-token targets, SLIM adds only $N \times k$ scalar parameters. This design reflects the hypothesis that future-token information is distributed across intermediate layers rather than being fully exposed in the final hidden state. It also lets different future-token offsets select different levels of abstraction, instead of forcing all predictions to rely on a single final-layer representation.

We evaluate SLIM on five pretrained models spanning dense and mixture-of-experts architectures: OLMo-2-1B, OLMo-2-7B (OLMo Team et al., 2025), OLMoE-1B-7B (Muennighoff et al., 2024), SmolLM2-1.7B (Allal et al., 2025), and TinyLlama-1.1B (Zhang et al., 2024). Under the same CPT token budget as an NTP baseline, SLIM consistently increases acceptance length under self-speculative decoding, indicating that the adapted models commit more tokens per verification step. Figure 1(b) summarizes this effect on representative backbones, where SLIM improves acceptance length from 1.00 to 1.43 on OLMo-2-1B, from 1.00 to 1.81 on OLMo-2-7B, and from 1.00 to 1.97 on OLMoE-1B-7B. These gains are obtained without adding draft modules or new output heads, making SLIM a lightweight path toward self-speculative decoding for pretrained checkpoints. Standard language-modeling benchmark performance is largely preserved, and learned gate analyses further show that different future-token offsets rely on distinct intermediate layers rather than collapsing onto the final hidden state.

Our contributions are as follows:

- **A minimal-overhead method for CPT-based MTP.** SLIM introduces MTP capability into pretrained checkpoints using only $N \times k$ learnable scalar gates and the model's existing unembedding matrix.
- **Evidence that MTP can be elicited from pretrained NTP checkpoints.** Across five dense and MoE models, SLIM improves self-

speculative decoding acceptance length under the same CPT budget as an NTP baseline while largely preserving standard benchmark performance.

- **An interpretable account of future-token information across layers.** Learned gates show that different future-token offsets rely on distinct intermediate layers rather than collapsing onto the final hidden state.

2 Related Work

Multi-Token Prediction. Predicting multiple future tokens has a long history, from blockwise parallel decoding (Stern et al., 2018) to ProphetNet's n-step-ahead objective (Qi et al., 2020). At pretraining scale, Gloeckle et al. (2024) show that attaching multiple MTP heads to a shared language-model trunk improves coding performance and enables self-speculative decoding, and production-scale systems such as DeepSeek-V3 (DeepSeek-AI et al., 2024) and GLM-5 (GLM-5 Team, 2026) incorporate MTP into their training pipelines. Subsequent work studies curriculum schedules from NTP to MTP (Aynetdinov and Akbik, 2025), leap-based future prediction (Liu et al., 2025), and efficient-inference formulations (Mehra et al., 2025). These approaches demonstrate the value of MTP, but they typically assume that MTP is introduced during pretraining or implemented with additional prediction heads.

A complementary line of work suggests that future-token information may already be present in NTP-pretrained models. Intermediate Transformer representations encode information not fully exposed at the final layer (Belrose et al., 2023), and Future Lens (Pal et al., 2023) shows that intermediate layers linearly encode signals about tokens several steps ahead. Samragh et al. (2025) further extract future-token predictions from NTP-trained checkpoints, although Mehra et al. (2025) find that simply attaching MTP heads to a frozen NTP backbone yields limited gains because upper layers are strongly specialized for immediate next-token prediction. SLIM builds on this evidence, but takes a lighter route: it elicits MTP capability during continual pretraining using only $N \times k$ scalar gates over layer-wise hidden states and the model's existing unembedding matrix, without LoRA adapters, auxiliary samplers, or additional LM heads.

Self-Speculative Decoding. Speculative decoding (Leviathan et al., 2023; Chen et al., 2023) drafts candidate continuations and verifies them in parallel; self-speculative variants reuse the target model itself as the drafter. Medusa (Cai et al., 2024) attaches multiple prediction heads to the final hidden state, EAGLE (Li et al., 2024b) trains a feature-level draft module, and later variants improve drafting through dynamic draft trees (Li et al., 2024a) or sequentially dependent heads (Ankner et al., 2024). EAGLE-3 (Li et al., 2025) is closest in spirit to SLIM in that it uses information from multiple layers, but it still relies on a separate draft module with substantial additional parameters. LayerSkip (Elhoushi et al., 2024) avoids auxiliary draft modules through early exits, but requires a dedicated training recipe with layer dropout and early-exit losses. In contrast, SLIM uses standard CPT with an MTP objective and adds only scalar layer-integration gates, making it directly applicable to pretrained checkpoints with negligible architectural overhead.

3 Preliminaries

3.1 Next-Token Prediction

Given a token sequence $\mathbf{x} = (x_1, \ldots, x_T)$, a standard autoregressive language model with parameters θ is trained to predict the next token x_{t+1} from the prefix $x_{\leq t}$. The next-token prediction (NTP) objective minimizes the negative log-likelihood:

$$\mathcal{L}_{\text{NTP}} = -\sum_{t=1}^{T-1} \log P_\theta(x_{t+1} \mid x_{\leq t}). \quad (1)$$

At inference time, the model generates autoregressively by repeatedly sampling or selecting from $P_\theta(x_{t+1} \mid x_{\leq t})$.

3.2 Multi-Token Prediction

Multi-token prediction (MTP) augments NTP by training the model to predict additional future tokens from the same prefix. In this work, k denotes the number of *additional* future-token targets and does not include the standard NTP target x_{t+1}.[1] Given $x_{\leq t}$, MTP predicts $(x_{t+2}, \ldots, x_{t+k+1})$ in addition to x_{t+1}. The future-token loss is

$$\mathcal{L}_{\text{MTP}} = -\sum_{t=1}^{T-k-1} \sum_{i=2}^{k+1} \log P_\theta(x_{t+i} \mid x_{\leq t}). \quad (2)$$

[1] This follows the convention of DeepSeek-AI et al. (2024). Under the convention of Gloeckle et al. (2024), where n counts all predicted positions including x_{t+1}, our $k = 2$ corresponds to $n = 3$.

Conventional MTP implementations often compute these probabilities using multiple prediction heads on top of a shared backbone (Gloeckle et al., 2024; DeepSeek-AI et al., 2024). The resulting future-token predictions can be used as draft tokens for self-speculative decoding, where multiple proposed tokens are verified by the target model in parallel (Stern et al., 2018; Leviathan et al., 2023). MTP can also act as an auxiliary training signal that encourages the model to represent dependencies beyond the immediate next token (Gloeckle et al., 2024).

In contrast to from-scratch MTP pretraining, we introduce the MTP objective during continual pretraining of an existing NTP-pretrained checkpoint. The next section describes how SLIM implements this objective without adding separate LM heads or auxiliary draft modules.

4 Method

4.1 Design Rationale

SLIM introduces lightweight MTP branches into an NTP-pretrained checkpoint during continual pretraining (CPT), without adding separate LM heads or auxiliary draft modules, where each branch predicts one additional future token beyond the backbone's next-token target. The design is motivated by two observations. MTP training benefits from prior NTP exposure, especially at smaller scales (Aynetdinov and Akbik, 2025). Second, NTP-trained models already encode future-token information in their internal representations (Pal et al., 2023; Samragh et al., 2025). These findings suggest that MTP capability may be elicited from an existing checkpoint rather than learned entirely from scratch.

The key design question is where to extract this future-token information. A natural baseline is to attach all MTP branches to the final hidden state. However, the final layer of an autoregressive language model is highly specialized for immediate next-token prediction and may not expose future-token signals distributed across intermediate layers. In a preliminary experiment, attaching MTP branches to the final hidden state yielded acceptance length close to 1, confirming that the final layer alone provides little usable future-token signal. SLIM therefore lets each branch learn its own mixture of layer representations through scalar-gated layer integration.

4.2 Scalar-Gated Layer Integration

Let the pretrained backbone have N Transformer layers, and let $h_t^l \in \mathbb{R}^d$ denote the hidden state at token position t from layer $l \in \{1, \ldots, N\}$. For each additional future-token offset $i \in \{2, \ldots, k+1\}$, SLIM forms a branch hidden state by aggregating all layer representations:

$$h_t^{\text{branch}(i)} = \sum_{l=1}^{N} \alpha_{l,i} h_t^l. \tag{3}$$

The layer weights $\alpha_{l,i}$ are computed from learnable scalar parameters $w_{l,i} \in \mathbb{R}$ using a normalized sigmoid parameterization:

$$\alpha_{l,i} = \frac{\sigma(w_{l,i})}{\sum_{l'=1}^{N} \sigma(w_{l',i})}. \tag{4}$$

The scalars form a gate matrix $W \in \mathbb{R}^{N \times k}$, where each column corresponds to one additional future-token offset. We initialize $W \sim \mathcal{N}(0, 0.02)$, yielding near-uniform layer weights at the beginning of CPT.

This normalized convex combination has two advantages. First, it prevents the MTP branches from arbitrarily amplifying hidden-state magnitudes, which helps preserve the pretrained NTP behavior. Second, the learned gates are directly interpretable as per-layer contributions to each future-token branch.

Each branch hidden state is decoded through the model's existing unembedding matrix $W_U \in \mathbb{R}^{|\mathcal{V}| \times d}$:

$$P_{\text{MTP}}(x_{t+i} \mid x_{\leq t}) = \text{softmax}\left(W_U h_t^{\text{branch}(i)}\right), \tag{5}$$

where $\mathcal{V}$ is the vocabulary. SLIM therefore introduces no new output projection, LM head, or draft module. The only additional parameters are the $N \times k$ scalar gates in W.

4.3 Continual Pretraining Objective

During CPT, SLIM optimizes a joint objective that combines the standard NTP loss with the MTP loss induced by the k branches:

$$\mathcal{L} = \mathcal{L}_{\text{NTP}} + \lambda \mathcal{L}_{\text{MTP}}, \tag{6}$$

where λ controls the strength of future-token supervision. Gradients update both the pretrained backbone parameters and the gate matrix W. Thus, SLIM is not a frozen probing method: it lightly

adapts the checkpoint so that future-token information becomes more accessible while the NTP objective anchors the original next-token behavior.

Because the MTP branches reuse the same backbone hidden states and the same unembedding matrix as NTP, the additional supervision remains tied to the pretrained language-modeling space. In practice, the loss weight λ controls the trade-off between stronger self-speculative decoding behavior and preservation of downstream reasoning performance.

4.4 Self-Speculative Decoding

After CPT, the learned MTP branches enable self-speculative decoding without an auxiliary draft model. Given a prefix $x_{\leq t}$, the NTP path predicts x_{t+1} while the MTP branches propose draft tokens for $x_{t+2}, \ldots, x_{t+k+1}$, which the same target model verifies in parallel. Accepted tokens are committed, and decoding resumes from the last accepted position; if no draft token is accepted, decoding falls back to the standard autoregressive step.

We evaluate this behavior using acceptance length, the average number of committed tokens per verification step. This metric does not directly measure end-to-end wall-clock speedup, but it captures whether the learned branches allow the model to advance by more than one token per verification step.

5 Experiments

5.1 Experimental Setup

Training. We evaluate SLIM on five pretrained backbones spanning dense and mixture-of-experts architectures: OLMo-2-1B and OLMo-2-7B (OLMo Team et al., 2025), OLMoE-1B-7B (Muennighoff et al., 2024), SmolLM2-1.7B (Allal et al., 2025), and TinyLlama-1.1B (Zhang et al., 2024). For each model, we run 10B tokens of continual pretraining (CPT) using the data mixture released with the original checkpoint whenever available. We compare SLIM against two reference points: the initial pretrained checkpoint and an NTP baseline trained on the same data for the same number of tokens. The initial checkpoint measures the effect of additional CPT itself, while the same-budget NTP baseline isolates the effect of replacing standard CPT with the SLIM objective. Unless otherwise stated, we use $k = 2$ additional future-token targets and $\lambda = 0.1$ for the MTP loss. To accelerate gate learning while preserving the pretrained backbone, we train the scalar gate matrix with a learning rate $100\times$ larger than that of the backbone. Full training details are provided in Appendix A.

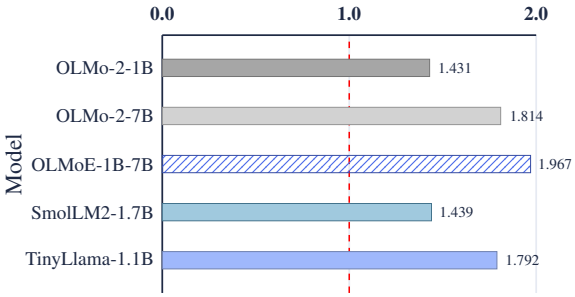


Figure 2: Overall acceptance length at k=2 averaged across 11 SPEED-Bench domains. The red dashed line marks the autoregressive baseline.

Evaluation. We evaluate SLIM along two complementary axes: standard language-modeling benchmarks and self-speculative decoding. For language-modeling benchmarks, we use `lm-evaluation-harness` (Gao et al., 2023) on ARC-Challenge, HellaSwag, MMLU, NQ-Open, OpenBookQA, PIQA, WinoGrande, GSM8K, and HumanEval.[2] GSM8K and HumanEval are omitted for SMolLM2-1.7B and TinyLlama-1.1B because their initial scores on both tasks are nearly zero.

For self-speculative decoding, we report acceptance length (Abramovich et al., 2026), defined as

$$\text{Acceptance length} = G/V, \tag{7}$$

where G is the total number of generated tokens and V is the number of verification steps. Standard autoregressive decoding has acceptance length 1, while the maximum value with k additional MTP branches is $k+1$. We measure acceptance length on the SPEED-Bench qualitative split, which contains 880 prompts across 11 semantic categories, using a batch size of 1. We use acceptance length as a decoding-efficiency proxy rather than as a direct measurement of end-to-end wall-clock speedup.

5.2 Self-Speculative Decoding

Figure 2 shows that SLIM obtains acceptance length greater than 1 for every model, indicating

[2] ARC-Challenge, HellaSwag, OpenBookQA, PIQA, and WinoGrande (5-shot, `acc_norm`); MMLU (5-shot, `acc`); NQ-Open (5-shot, exact match); GSM8K (8-shot, exact match); HumanEval (0-shot, pass@1).

that scalar-gated MTP branches successfully provide self-speculative decoding capability after CPT. The effect is consistent across dense and MoE architectures. Within the OLMo-2 family, scaling from 1B to 7B increases acceptance length from 1.431 to 1.814, suggesting that larger pretrained representations contain more exploitable future-token structure. OLMoE-1B-7B achieves the highest acceptance length overall (1.967), suggesting that sparse expert routing may be beneficial for multi-token drafting, although we leave a detailed analysis of this interaction to future work.

Figure 3 provides a per-domain breakdown. SLIM performs strongly on coding, math, reasoning, and roleplay, where local structure and repeated token patterns are common. Multilingual prompts show weaker acceptance length, likely reflecting the English-dominant composition of the CPT data. These domain-level differences suggest that SLIM is most effective when future tokens are locally constrained by syntactic, structural, or discourse-level regularities, while broader multilingual coverage may require more diverse CPT data.

5.3 Language Modeling Benchmarks

To assess whether SLIM preserves the base model's general language capabilities, we evaluate all models on standard downstream benchmarks. Table 1 reports the full results. Across most metrics, SLIM largely preserves the NTP baseline, showing that the MTP objective does not broadly disrupt the pretrained model's language-modeling capabilities. HumanEval scores remain identical to the NTP baseline for the models where the comparison is meaningful, suggesting that coding-relevant representations are preserved. The main consistent trade-off appears on GSM8K, where SLIM trails the NTP baseline by 4.7, 2.3, and 1.8 points on OLMo-2-1B, OLMo-2-7B, and OLMoE-1B-7B, respectively. This suggests that SLIM improves token-level drafting behavior while incurring a localized penalty on multi-step mathematical reasoning. We next show that this trade-off can be controlled by adjusting the MTP-loss weight λ.

5.4 Loss-Weight Sensitivity

The benchmark results above show that the main trade-off of SLIM appears on multi-step mathematical reasoning, especially GSM8K. To test whether this degradation is an inherent limitation of scalar-gated MTP or a controllable consequence of the MTP supervision strength, we conduct an additional sensitivity experiment on OLMo-2-1B with a smaller MTP-loss weight, $\lambda = 0.01$, while keeping all other hyperparameters fixed.

Table 2 shows that lowering λ from 0.1 to 0.01 recovers GSM8K from 29.58 to 32.60 and HumanEval from 6.09 to 6.70, matching the initial checkpoint on HumanEval and substantially reducing the GSM8K regression. The remaining benchmarks are preserved or slightly improved, as reported in Appendix B. This result indicates that the GSM8K drop is not an unavoidable consequence of SLIM, but rather a controllable trade-off induced by the strength of future-token supervision.

The recovery in reasoning performance comes at the cost of reduced self-speculative decoding behavior. The average acceptance length on SPEED-Bench decreases from 1.44 to 1.12 when λ is reduced from 0.1 to 0.01. Thus, λ acts as a simple knob that interpolates between stronger speculative drafting and stronger preservation of reasoning-sensitive capabilities. This controllability is practically useful: applications prioritizing decoding efficiency may prefer a larger λ, whereas applications requiring stronger multi-step reasoning may choose a smaller value.

5.5 Computational Overhead

Parameters. SLIM adds only the scalar gates $W \in \mathbb{R}^{N \times k}$—no new output heads, draft models, or auxiliary modules. As shown in Table 3, this overhead is negligible relative to the backbone.

FLOPs. Following Kaplan et al. (2020), we exclude embedding and unembedding operations from FLOPs estimates. SLIM adds $2 \times N \times k \times d$ operations per token for gate-weighted layer aggregation. The MTP loss further requires k forward passes through the shared unembedding matrix at training time, comparable to other multi-token training objectives.

As shown in Table 4, the FLOPs introduced by SLIM's gate-weighted layer aggregation are on the order of 10^{-5} relative to backbone computation.

5.6 Training Dynamics

To examine how MTP ability emerges during CPT, we evaluate intermediate checkpoints every 2B tokens along two axes: aggregate language-modeling performance and acceptance length. Figure 4 shows that acceptance length increases steadily with training tokens, whereas the language-

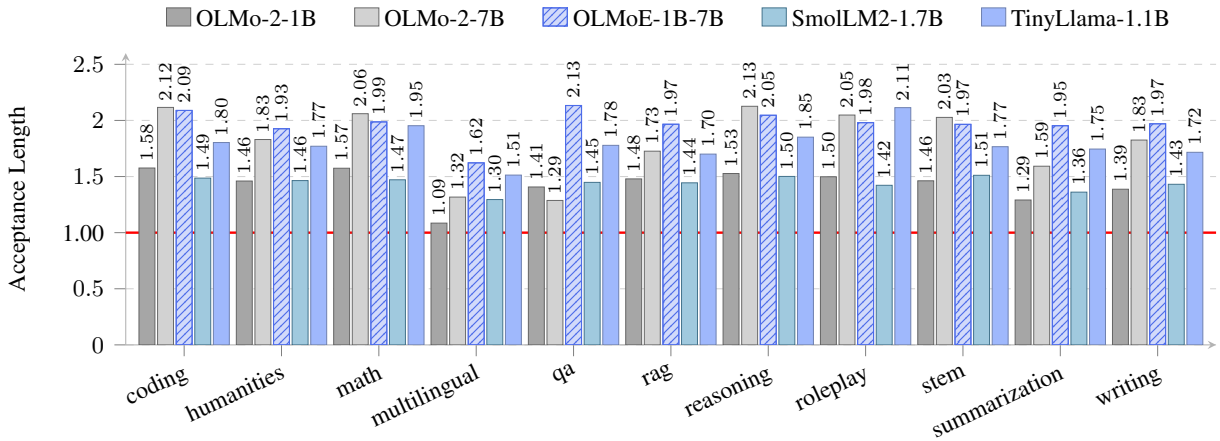


Figure 3: Acceptance length comparison across models.

Model	Variant	ARC-C	HellaSwag	MMLU	NQ-Open	OBQA	PIQA	WinoG.	GSM8K	HumanEval
OLMo-2-1B	Initial	45.1	67.7	42.7	14.0	41.6	77.1	66.7	33.2	6.7
	NTP	45.6 (0.5)	67.6 (0.1)	41.5 (1.2)	14.0	40.4 (1.2)	76.3 (0.8)	66.4 (0.3)	34.3 (1.1)	6.1 (0.6)
	SLIM	45.2 (0.1)	66.0 (1.7)	40.8 (1.9)	13.1 (0.9)	40.2 (1.4)	76.4 (0.7)	65.5 (1.2)	29.6 (3.6)	6.1 (0.6)
OLMo-2-7B	Initial	62.3	79.5	62.6	28.3	47.4	82.3	75.7	66.9	11.3
	NTP	60.8 (1.5)	79.4 (0.1)	62.0 (0.6)	29.0 (0.7)	48.4 (1.0)	82.3	75.8 (0.1)	63.5 (3.4)	14.0 (2.7)
	SLIM	61.4 (0.9)	78.7 (0.8)	61.1 (1.5)	27.8 (0.5)	48.6 (1.2)	81.9 (0.4)	73.9 (1.8)	61.2 (5.7)	14.0 (2.7)
OLMoE-1B-7B	Initial	53.3	76.8	52.6	20.2	46.0	82.1	71.7	9.0	13.4
	NTP	52.0 (1.3)	77.1 (0.3)	51.9 (0.7)	19.0 (1.2)	44.8 (1.2)	81.9 (0.2)	70.4 (1.3)	8.0 (1.0)	13.4
	SLIM	53.0 (0.3)	76.3 (0.5)	50.8 (1.8)	18.8 (1.4)	45.0 (1.0)	81.7 (0.4)	70.3 (1.4)	6.2 (2.8)	13.4
SmolLM2-1.7B	Initial	51.0	71.1	50.3	16.2	44.8	77.9	68.3	—	—
	NTP	51.8 (0.8)	71.1	50.3	16.3 (0.1)	45.2 (0.4)	78.4 (0.5)	69.1 (0.8)	—	—
	SLIM	51.9 (0.9)	71.0 (0.1)	50.4 (0.1)	16.2	45.6 (0.8)	78.9 (1.0)	68.0 (0.3)	—	—
TinyLlama-1.1B	Initial	35.5	61.8	27.1	11.7	39.0	74.7	64.0	—	—
	NTP	37.0 (1.5)	61.7 (0.1)	25.9 (1.2)	12.3 (0.6)	39.4 (0.4)	74.7	62.7 (1.3)	—	—
	SLIM	36.4 (0.9)	60.3 (1.5)	25.4 (1.7)	11.5 (0.2)	37.0 (2.0)	74.8 (0.1)	61.6 (2.4)	—	—

Table 1: Benchmark results (k=2). All scores reported in %. Values in parentheses denote the change relative to the Initial pretrained checkpoint: green indicates improvement, red indicates degradation.

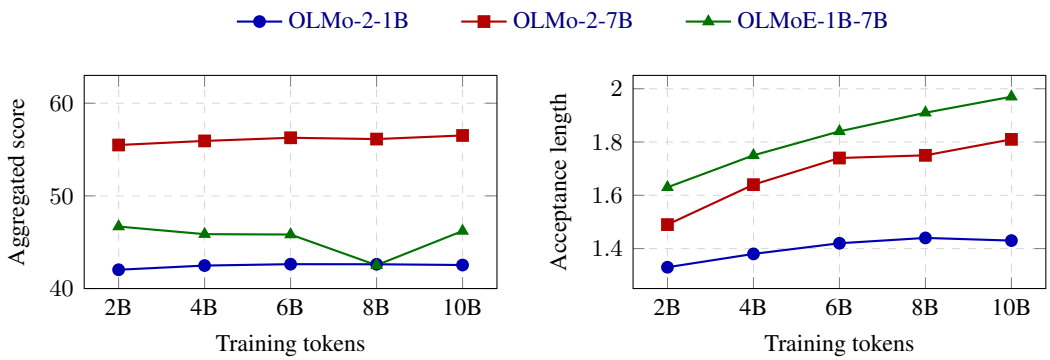


Figure 4: Training dynamics of SLIM across continual pretraining, reported on a common 2B-token axis. (a) Average accuracy over language-modeling benchmarks. (b) Mean acceptance length under self-speculative decoding with $k = 2$.

modeling average remains approximately flat. This supports the claim that SLIM elicits self-speculative decoding behavior without degrading benchmark performance.

Larger models yield larger gains. Over 10B CPT tokens, OLMo-2-7B improves from 1.49 to 1.81, and OLMoE-1B-7B improves from 1.63 to 1.97, whereas OLMo-2-1B improves from 1.33 to 1.43 and begins to plateau near the end of training. These dynamics suggest that larger or more expressive pretrained backbones provide more usable future-token structure for SLIM to elicit.

Setting	GSM8K	HumanEval	Acc. Len.
Initial	33.22	6.70	1.00
NTP	34.34	6.10	1.33
SLIM ($\lambda = 0.1$)	29.58	6.09	1.44
SLIM ($\lambda = 0.01$)	32.60	6.70	1.12

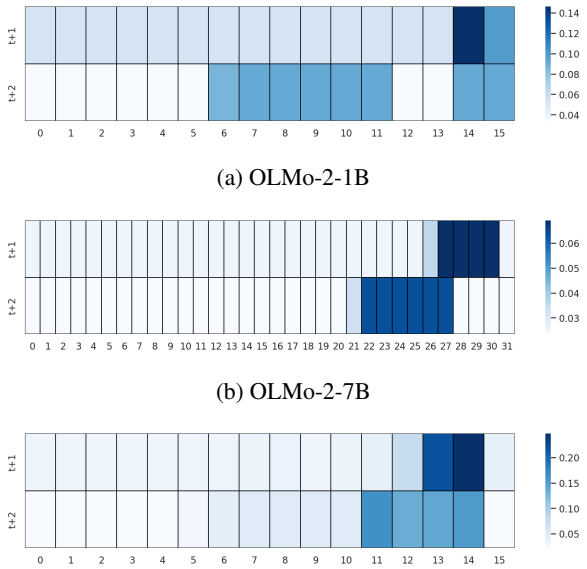
Table 2: Effect of the MTP-loss weight on OLMo-2-1B. Lowering λ recovers reasoning-sensitive benchmarks such as GSM8K and HumanEval, at the cost of lower acceptance length.

Model	Model Size	SLIM	Ratio
OLMo-2-1B	1.3B	32	2.4×10^{-8}
OLMo-2-7B	6.8B	64	9.4×10^{-9}
OLMoE-1B-7B	7.0B	32	4.6×10^{-9}
SmolLM2-1.7B	1.7B	48	2.8×10^{-8}
TinyLlama-1.1B	1.0B	44	4.0×10^{-8}

Table 3: Additional parameters introduced by SLIM ($k = 2$). The total overhead is bounded by $N \times k$ scalars regardless of model size. The OLMoE-1B-7B ratio is computed against total parameters.

Model	Backbone	SLIM	Ratio
OLMo-2-1B	7.8×10^{9}	131,072	1.7×10^{-5}
OLMo-2-7B	4.1×10^{10}	524,288	1.3×10^{-5}
OLMoE-1B-7B	6.6×10^{9}	131,072	2.0×10^{-5}
SmolLM2-1.7B	1.0×10^{10}	196,608	2.0×10^{-5}
TinyLlama-1.1B	6.0×10^{9}	180,224	3.0×10^{-5}

Table 4: SLIM introduces negligible computational overhead relative to backbone FLOPs per token ($k = 2$).



(a) OLMo-2-1B

(b) OLMo-2-7B

(c) OLMoE-1B-7B

Figure 5: Learned gate heatmaps after continual pretraining (k=2). Vertical axis: future-token offset (t+1, t+2); horizontal axis: layer index. Each cell shows the scalar gate weight assigned to that layer for predicting the token at the corresponding offset.

5.7 Analysis of Learned Gates

Figure 5 visualizes the learned gates after CPT. Across all three models, the gates exhibit consistent offset-dependent layer specialization: the t+1 branch concentrates on the penultimate layers rather than the final layer, while the t+2 branch shifts toward upper-middle depths. SLIM thus does not collapse onto the final hidden state but selects different layers depending on the prediction offset. The under-weighting of the final layer is consistent with prior interpretability findings that the final hidden state of a pretrained next-token language model is heavily specialized for x_{t+1} prediction and exposes less of the residual stream's broader content (Belrose et al., 2023; Pal et al., 2023). Appendix Figures 6 further show that the initially near-uniform gates progressively concentrate onto a small band of layers, mirroring the acceptance-length rise in Section 5.6. Overall, the gate analyses support SLIM's central mechanism: MTP capability is not injected by a new predictive module but progressively elicited by learning how each branch routes information from the pretrained layer stack.

6 Conclusion

We introduced SLIM, a lightweight method for eliciting multi-token prediction capability from pretrained next-token language models via continual pretraining. SLIM learns scalar-gated combinations of layer-wise hidden states and decodes the resulting branch representations with the model's existing unembedding matrix. This design adds only $N \times k$ scalar parameters while allowing different future-token offsets to draw on different intermediate layers. Across five pretrained dense and mixture-of-experts models, SLIM consistently improves self-speculative decoding acceptance length, while largely preserving standard language-modeling benchmark performance. The learned gate patterns further show that future-token information is not confined to the final hidden state, but can be accessed from distinct intermediate layers depending on the prediction offset. These findings suggest MTP can be elicited from existing checkpoints with negligible architectural overhead, rather than requiring from-scratch pretraining.

Limitations

This work has several limitations. First, we evaluate decoding efficiency primarily through acceptance length rather than end-to-end wall-clock latency. Acceptance length directly measures how many tokens are committed per verification step, but actual speedup depends on implementation details such as verification overhead, KV-cache management, batching, and hardware. Future work should benchmark optimized SLIM decoding kernels under realistic serving conditions.

Second, SLIM introduces a trade-off between stronger self-speculative decoding behavior and some downstream reasoning performance. In particular, we observe regressions on GSM8K, suggesting that future-token supervision can interfere with multi-step mathematical reasoning. Our λ sensitivity analysis shows that this trade-off can be mitigated by reducing the MTP-loss weight, but a more principled adaptive weighting strategy remains an important direction.

Third, our experiments focus on a moderate continual-pretraining budget and on $k{=}2$ additional future-token targets. As shown in Appendix C.1, extending to $k{=}4$ leads to noticeably weaker downstream performance and a *lower* mean acceptance length than $k{=}2$ on OLMo-2-1B, alongside less stable gate patterns. This suggests that longer-range MTP under SLIM may require more CPT tokens, offset-specific learning-rate or loss-weight schedules, or more adaptive gating mechanisms, which we leave to future work. Finally, although we evaluate both dense and MoE backbones, we do not fully analyze how expert routing interacts with scalar-gated MTP branches. Understanding this interaction may help explain why MoE models show strong acceptance-length gains.

References

Talor Abramovich, Maor Ashkenazi, Carl Putterman, Benjamin Chislett, Tiyasa Mitra, Bita Darvish Rouhani, Ran Zilberstein, and Yonatan Geifman. 2026. SPEED-Bench: A unified and diverse benchmark for speculative decoding. *arXiv preprint arXiv:2604.09557*.

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Agustín Piqueres Lajarín, Vaibhav Srivastav, Joshua Lochner, Caleb Fahlgren, Xuan-Son Nguyen, Clémentine Fourrier, Ben Burtenshaw, Hugo Larcher, Haojun Zhao, Cyril Zakka, Mathieu Morlon, and 3 others. 2025. SmolLM2: When smol goes big – data-centric training of a small language model. *arXiv preprint arXiv:2502.02737*.

Zachary Ankner, Rishab Parthasarathy, Aniruddha Nrusimha, Christopher Rinard, Jonathan Ragan-Kelley, and William Brandon. 2024. Hydra: Sequentially-dependent draft heads for Medusa decoding. In *First Conference on Language Modeling (COLM)*.

Ansar Aynetdinov and Alan Akbik. 2025. Pre-training curriculum for multi-token prediction in language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 25573–25588, Vienna, Austria. Association for Computational Linguistics.

Nora Belrose, Zach Furman, Logan Smith, Danny Halawi, Igor Ostrovsky, Lev McKinney, Stella Biderman, and Jacob Steinhardt. 2023. Eliciting latent predictions from transformers with the tuned lens. *arXiv preprint arXiv:2303.08112*.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D. Lee, Deming Chen, and Tri Dao. 2024. Medusa: Simple LLM inference acceleration framework with multiple decoding heads. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*.

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

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, and 1 others. 2024. DeepSeek-V3 technical report. *arXiv preprint arXiv:2412.19437*.

Mostafa Elhoushi, Akshat Shrivastava, Diana Liskovich, Basil Hosmer, Bram Wasti, Liangzhen Lai, Anas Mahmoud, Bilge Acun, Saurabh Agarwal, Ahmed Roman, Ahmed A. Aly, Beidi Chen, and Carole-Jean Wu. 2024. LayerSkip: Enabling early exit inference and self-speculative decoding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12622–12642. Association for Computational Linguistics.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, and 5 others. 2023. A framework for few-shot language model evaluation.

GLM-5 Team. 2026. GLM-5: from vibe coding to agentic engineering. *arXiv preprint arXiv:2602.15763*.

Fabian Gloeckle, Badr Youbi Idrissi, Baptiste Rozière, David Lopez-Paz, and Gabriel Synnaeve. 2024. Better & faster large language models via multi-token prediction. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235, pages 15706–15734. PMLR.

Kshitij Gupta, Benjamin Thérien, Adam Ibrahim, Mats Leon Richter, Quentin Gregory Anthony, Eugene Belilovsky, Irina Rish, and Timothée Lesort. 2023. Continual pre-training of large language models: How to re-warm your model? In *Workshop on Efficient Systems for Foundation Models @ ICML2023*.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *Preprint*, arXiv:2001.08361.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, pages 19274–19286.

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

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. 2024b. EAGLE: Speculative sampling requires rethinking feature uncertainty. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, pages 28935–28948.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. 2025. EAGLE-3: Scaling up inference acceleration of large language models via training-time test. In *Advances in Neural Information Processing Systems*.

Xiaohao Liu, Xiaobo Xia, Weixiang Zhao, Manyi Zhang, Xianzhi Yu, Xiu Su, Shuo Yang, See-Kiong Ng, and Tat-Seng Chua. 2025. L-MTP: Leap multi-token prediction beyond adjacent context for large language models. In *Advances in Neural Information Processing Systems*.

Somesh Mehra, Javier Alonso Garcia, and Lukas Mauch. 2025. On multi-token prediction for efficient LLM inference. *arXiv preprint arXiv:2502.09419*.

Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Jacob Morrison, Sewon Min, Weijia Shi, Pete Walsh, Oyvind Tafjord, Nathan Lambert, Yuling Gu, Shane Arora, Akshita Bhagia, Dustin Schwenk, David Wadden, Alexander Wettig, Binyuan Hui, Tim Dettmers, Douwe Kiela, and 5 others. 2024. OLMoE: Open mixture-of-experts language models. *arXiv preprint arXiv:2409.02060*.

OLMo Team, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, Nathan Lambert, Dustin Schwenk, Oyvind Tafjord, Taira Anderson, David Atkinson, Faeze Brahman, Christopher Clark, Pradeep Dasigi, Nouha Dziri, and 21 others. 2025. 2 OLMo 2 Furious. *arXiv preprint arXiv:2501.00656*.

Koyena Pal, Jiuding Sun, Andrew C. Yuan, Byron C. Wallace, and David Bau. 2023. Future lens: Anticipating subsequent tokens from a single hidden state. In *Proceedings of the 27th Conference on Computational Natural Language Learning (CoNLL)*, pages 548–560, Singapore. Association for Computational Linguistics.

Weizhen Qi, Yu Yan, Yeyun Gong, Dayiheng Liu, Nan Duan, Jiusheng Chen, Ruofei Zhang, and Ming Zhou. 2020. ProphetNet: Predicting future n-gram for sequence-to-Sequence pre-training. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2401–2410. Association for Computational Linguistics.

Mohammad Samragh, Arnav Kundu, David Harrison, Kumari Nishu, Devang Naik, Minsik Cho, and Mehrdad Farajtabar. 2025. Your LLM knows the future: Uncovering its multi-token prediction potential. *arXiv preprint arXiv:2507.11851*.

Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. 2018. Blockwise parallel decoding for deep autoregressive models. In *Advances in Neural Information Processing Systems*, volume 31.

Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. 2024. TinyLlama: An open-source small language model. *arXiv preprint arXiv:2401.02385*.

A Experimental Details

This appendix provides additional implementation details for the continual-pretraining experiments. The goal is to make the comparison between the NTP baseline and SLIM reproducible and controlled. For each backbone, the NTP and SLIM variants use the same pretrained checkpoint, data source, optimizer, learning-rate schedule, sequence length, global batch size, and token budget. The only substantive difference is that SLIM adds scalar-gated MTP branches and optimizes the auxiliary MTP objective.

A.1 Base Models

We conduct continual-pretraining experiments on five pretrained language models spanning dense and mixture-of-experts architectures. All base checkpoints are obtained from HuggingFace: OLMo-2-1B,[3] OLMo-2-7B,[4] OLMoE-1B-7B,[5] SmolLM2-1.7B,[6] and TinyLlama-1.1B.[7]

This selection is intended to test whether SLIM is tied to a particular backbone or can be applied broadly to existing NTP-pretrained checkpoints. The OLMo-2 pair allows us to examine scaling within the same model family, OLMoE-1B-7B tests whether scalar-gated MTP can be elicited from a sparse expert-routed backbone, and SmolLM2-1.7B and TinyLlama-1.1B provide additional dense checkpoints trained with different data mixtures and design choices.

Family	Params	Seq. Len	GBS	Tok/Step	Steps
OLMo-2-1B	1B	4096	512	2.1M	5000
OLMo-2-7B	7B	4096	1024	4.2M	2500
OLMoE-1B-7B	1B×7B	4096	1024	4.2M	2500
SmolLM2-1.7B	1.7B	2048	1024	2.1M	5000
TinyLlama-1.1B	1.1B	2048	4096	8.4M	1250

Table 5: Training hyperparameters by model family. GBS denotes global batch size, and Tok/Step denotes the number of training tokens processed per optimization step.

[3] https://huggingface.co/allenai/OLMo-2-0425-1B

[4] https://huggingface.co/allenai/OLMo-2-1124-7B

[5] https://huggingface.co/allenai/OLMoE-1B-7B-0125

[6] https://huggingface.co/HuggingFaceTB/SmolLM2-1.7B

[7] https://huggingface.co/TinyLlama/TinyLlama_v1.1

A.2 Training Setup

All experiments use AdamW with β_1=0.9, β_2=0.95, ε=10^{-8}, gradient clipping at 1.0, and a constant learning rate with no warm-up. We use the constant learning-rate setup following Gupta et al. (2023). The learning rate is 4×10^{-5} for OLMo-2-1B, OLMo-2-7B, and TinyLlama-1.1B, and 5×10^{-5} for OLMoE-1B-7B and SmolLM2-1.7B. Weight decay is 0.01 for OLMo-2-1B, OLMo-2-7B, OLMoE-1B-7B, and SmolLM2-1.7B, and 0.1 for TinyLlama-1.1B.

For each backbone, the NTP baseline and SLIM are trained for approximately 10B tokens on the same data mixture. The token budget is matched by adjusting global batch size and training steps, as shown in Table 5. Thus, the comparison isolates the effect of replacing standard CPT with scalar-gated MTP, rather than the effect of additional training compute.

For SLIM, we use k=2 additional future-token targets and λ=0.1 unless otherwise stated. To accelerate gate learning while keeping the pretrained backbone anchored, the scalar gate parameters are trained with a learning rate 100× larger than the backbone learning rate. All other optimizer and scheduler settings are shared with the NTP baseline.

All experiments are conducted on a single node with 8 × NVIDIA H100 80GB GPUs.

A.3 Training Data

OLMo-2-1B and OLMo-2-7B are trained on the OLMo-2 data mixture, a blend of DCLM (47.2%), math (20.8%), FLAN (16.6%), Wikipedia (7.1%), Semantic Scholar (5.9%), and StackExchange (2.5%). OLMoE-1B-7B uses the OLMoE data mixture, dominated by DCLM (95.1%) with small additions from StarCoder, PeS2o, and math corpora.

SmolLM2-1.7B is trained on the SmolLM2-1.7B Stage 4 data mixture released by the original authors, a blend of DCLM (35%), Stack-Edu (24%), FineWeb-Edu (23%), InfiWebMath (9.54%), FineMath (4.46%), and Cosmopedia-v2 (4%). Two minor components of the original mixture, OWM and AugGSM8K, together accounting for approximately ∼0.1%, are omitted. Within Stack-Edu and across the two math corpora, the natural distribution of the source data is preserved. TinyLlama-1.1B is trained on SlimPajama as a single-source CPT dataset.

We intentionally use the original or released

training mixture whenever possible. The purpose of this work is not to propose a new data recipe, but to test whether MTP capability can be elicited from existing NTP-pretrained checkpoints through a lightweight objective change.

A.4 Hyperparameters

Table 5 summarizes the model-family-specific hyperparameters used for continual pretraining. We adjust global batch size and the number of optimization steps so that each model family processes approximately the same total token budget of 10B tokens. This design is important because our goal is to compare the effect of the training objective—standard NTP versus SLIM—rather than the effect of additional training compute.

The sequence length follows the setting of each model family and its corresponding training setup. OLMo-2 and OLMoE are trained with a sequence length of 4096, while SmolLM2 and TinyLlama are trained with a sequence length of 2048. The resulting number of tokens per optimization step is computed as sequence length multiplied by global batch size. We then choose the number of steps so that the total number of training tokens remains approximately matched across models.

For each backbone, the same hyperparameter configuration is used for the NTP baseline and the corresponding SLIM run. Therefore, any difference between NTP and SLIM cannot be attributed to differences in batch size, sequence length, training steps, optimizer settings, or token budget. The only objective-level difference is the addition of scalar-gated future-token branches and the MTP loss in SLIM.

This controlled setup is especially important for interpreting the results in the main paper. The NTP baseline measures the effect of continuing the pretrained checkpoint on the same data, whereas SLIM measures whether replacing standard CPT with scalar-gated MTP can elicit useful future-token behavior under the same training budget.

Model	ARC-C	HellaSwag	MMLU	NQ-Open	OBQA	PIQA	WinoG.	GSM8K	HumanEval
OLMo-2-1B (initial)	45.05	67.66	42.66	13.96	41.60	77.09	66.69	33.22	6.70
OLMo-2-1B NTP	45.56	67.62	41.53	13.99	40.40	76.33	66.38	34.34	6.10
SLIM (λ=0.1)	45.22	65.97	40.76	13.10	40.20	76.44	65.51	29.58	6.09
SLIM (λ=0.01)	45.99	67.48	41.28	14.38	42.22	76.55	65.75	32.60	6.70

Table 6: Benchmark performance of OLMo-2-1B under different training configurations (k=2). Lowering λ from 0.1 to 0.01 recovers GSM8K and HumanEval to the level of the initial pretrained checkpoint, while leaving the remaining benchmarks essentially unchanged.

SPEED-Bench Domain	λ=0.1	λ=0.01
Coding	1.576	1.150
Humanities	1.460	1.120
Math	1.574	1.200
Multilingual	1.085	1.030
QA	1.407	1.070
RAG	1.479	1.080
Reasoning	1.527	1.140
Roleplay	1.498	1.090
STEM	1.462	1.140
Summarization	1.291	1.080
Writing	1.387	1.110
Average	**1.435**	**1.115**

Table 7: Per-domain acceptance length on SPEED-Bench for OLMo-2-1B SLIM under λ=0.1 and λ=0.01 (k=2).

B Sensitivity of the MTP loss weight

As discussed in Section 5.4, SLIM with the default λ=0.1 exhibits a regression on GSM8K relative to the NTP baseline. To test whether this regression is an inherent property of SLIM or a controllable consequence of the MTP supervision strength, we re-run continual pretraining on OLMo-2-1B with λ=0.01 while holding all other hyperparameters fixed.

This experiment tests whether λ acts as a practical trade-off parameter. If the GSM8K regression were an unavoidable consequence of scalar-gated MTP, reducing λ would not be expected to recover reasoning-sensitive benchmarks. Conversely, if the regression is caused by overly strong future-token supervision, a smaller λ should better preserve the original NTP behavior while reducing speculative-decoding gains.

Table 6 shows that lowering λ from 0.1 to 0.01 recovers GSM8K from 29.58 to 32.60 and HumanEval from 6.09 to 6.70. HumanEval returns to the initial-checkpoint level, and the GSM8K regression is substantially reduced. The remaining seven benchmarks are preserved or slightly improved, indicating that the regression is not an inherent limitation of SLIM but a controllable consequence of MTP supervision strength.

This recovery comes at a consistent cost in acceptance length. As shown in Table 7, the average acceptance length decreases from 1.435 under λ=0.1 to 1.115 under λ=0.01. The largest reductions occur in domains where SLIM originally performs strongly, including coding, math, reasoning, and roleplay.

Together, these results indicate that λ acts as a simple efficiency–preservation knob. A larger λ encourages stronger speculative drafting, while a smaller λ keeps the adapted checkpoint closer to the original NTP solution and better preserves reasoning-sensitive capabilities. Whether finer-grained mechanisms, such as layer-wise weighting, offset-specific λ schedules, or selective gradient detachment from shallow layers, can improve this trade-off remains an open question.

C Additional Analyses

This appendix provides two additional analyses that complement the main results. First, we examine whether increasing the future-token horizon from k=2 to k=4 improves downstream quality or self-speculative decoding. Second, we visualize how the learned scalar gates evolve during continual pretraining, showing how SLIM gradually routes future-token information from the pretrained layer stack.

C.1 Extended Horizon: k=4 Results on OLMo-2-1B

To probe how SLIM behaves under a longer prediction horizon, we additionally train OLMo-2-1B with k=4 future-token targets under the same CPT budget, learning rate, and loss weight (λ=0.1) as the k=2 configuration. This experiment asks whether simply exposing more draft positions improves downstream quality or self-speculative decoding.

The motivation for this experiment is straightforward. If the main limitation of SLIM were only the

Model	Variant	ARC-C	HellaSwag	MMLU	NQ-Open	OBQA	PIQA	WinoG.	GSM8K	HumanEval
OLMo-2-1B	Initial	45.1	67.7	42.7	14.0	41.6	77.1	66.7	33.2	6.7
	NTP	45.6 (0.5)	67.6 (0.1)	41.5 (1.2)	14.0	40.4 (1.2)	76.3 (0.8)	66.4 (0.3)	34.3 (1.1)	6.1 (0.6)
	SLIM (k=2)	45.2 (0.1)	66.0 (1.7)	40.8 (1.9)	13.1 (0.9)	40.2 (1.4)	76.4 (0.7)	65.5 (1.2)	29.6 (3.6)	6.1 (0.6)
	SLIM (k=4)	45.2 (0.1)	66.0 (1.7)	40.6 (2.1)	13.4 (0.6)	40.6 (1.0)	76.7 (0.4)	65.7 (1.0)	29.5 (3.7)	5.5 (1.2)

Table 8: Benchmark results on OLMo-2-1B comparing SLIM at k=2 and k=4. All scores are reported in %. Values in parentheses denote the change relative to the Initial pretrained checkpoint: green indicates improvement, red indicates degradation.

Variant	Mean Acceptance Length
SLIM (k=2)	1.435
SLIM (k=4)	1.287

Table 9: Mean acceptance length of OLMo-2-1B under self-speculative decoding with SLIM at k=2 vs. k=4. Despite having twice as many draft positions available, the k=4 variant yields a shorter effective acceptance length.

number of available draft positions, then increasing k from 2 to 4 should allow the model to commit more tokens per verification step. However, longer-horizon branches are also harder to learn because they must predict tokens farther away from the current prefix. Thus, this experiment tests whether the benefits of additional draft capacity outweigh the difficulty of learning reliable farther-offset predictions under the same CPT budget.

Downstream quality. As shown in Table 8, increasing the horizon from k=2 to k=4 does not improve downstream benchmark performance on OLMo-2-1B. On most tasks, the two settings are within noise of each other. ARC-C and HellaSwag are essentially unchanged, while NQ-Open, OBQA, PIQA, and WinoGrande show only small fluctuations. GSM8K remains nearly identical, moving from 29.6 to 29.5. The clearest degradation appears on HumanEval, where k=4 drops to 5.5.

Acceptance length. The more striking effect appears in self-speculative decoding. Although k=4 exposes more draft positions per step, Table 9 shows that the mean acceptance length decreases from 1.435 at k=2 to 1.287 at k=4. This indicates that simply adding more future-token branches is not sufficient. Later draft tokens are useful only if earlier draft tokens are accepted, so unreliable farther-offset predictions can reduce the effective benefit of longer draft chains.

Implications. These observations are consistent with our central hypothesis that SLIM elicits MTP capability progressively from the pretrained layer stack rather than learning it from scratch. When the prediction horizon exceeds what can be cleanly read out from intermediate representations under a moderate CPT budget, the additional branches may fail to specialize reliably. Scaling SLIM to longer horizons may therefore require more CPT tokens, offset-specific loss weighting, curriculum schedules that introduce farther offsets gradually, or more expressive gating mechanisms. We leave a systematic study of these directions to future work.

C.2 Gate Dynamics During Continual Pretraining

Figures 6 shows how the learned k=2 gates evolve during continual pretraining. These visualizations complement the final gate heatmaps in the main paper by showing that specialization is not present from the beginning, but emerges gradually as CPT proceeds.

Across all three backbones, the gates begin from a relatively diffuse distribution and progressively concentrate onto a small band of layers. This supports the interpretation that SLIM does not add an independent prediction module. Instead, it learns how to route future-token information already present in the pretrained layer stack. The dynamics are also offset-specific: the t+1 branch tends to rely on later layers, whereas the t+2 branch shifts toward shallower or upper-middle layers.

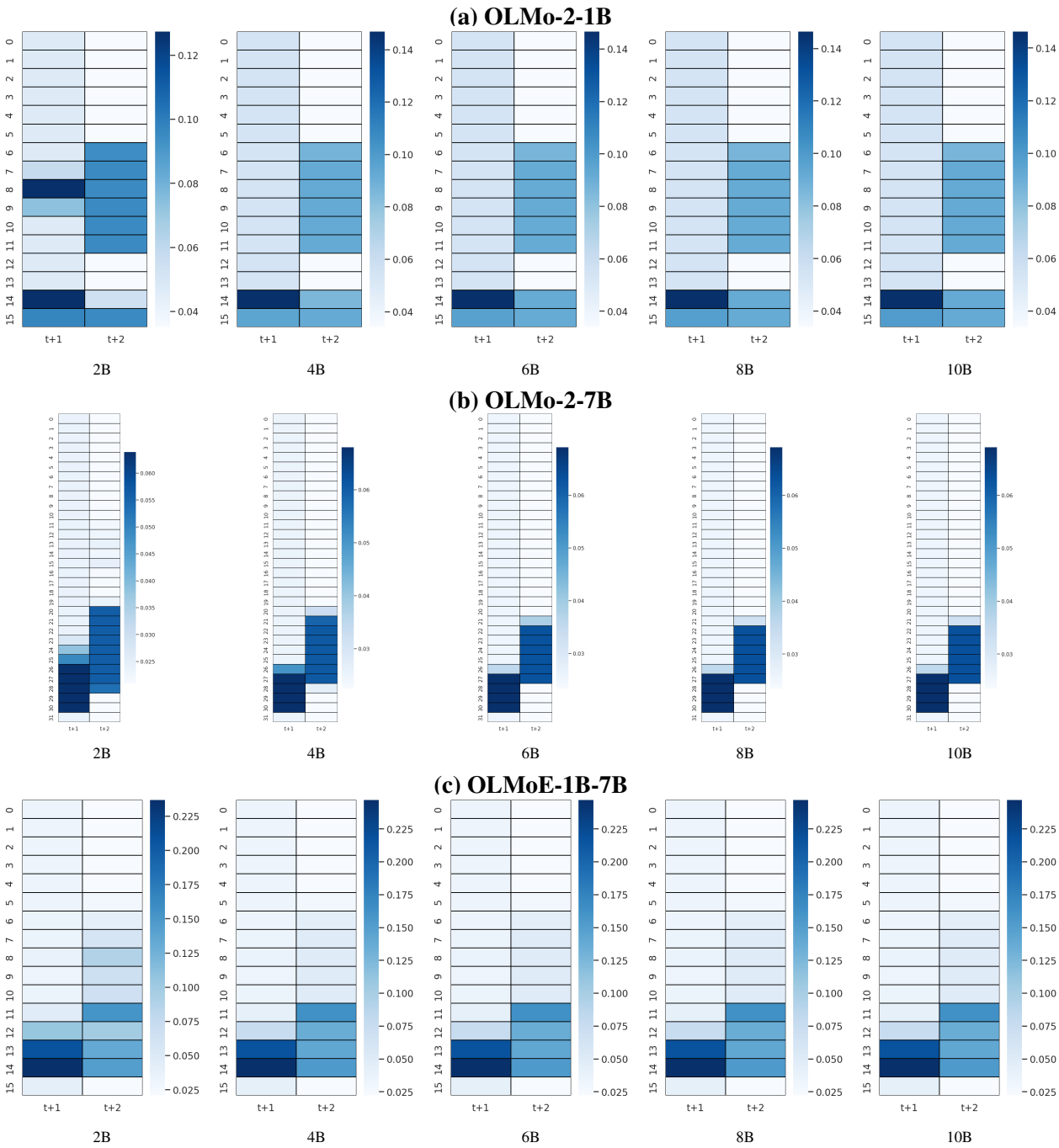


Figure 6: Evolution of learned k=2 gate values across continual pretraining for **(a) OLMo-2-1B**, **(b) OLMo-2-7B**, and **(c) OLMoE-1B-7B**. Snapshots are taken every 2B training tokens. Across model families, the gates progressively concentrate from an initially diffuse distribution into offset-specific layer bands, indicating that SLIM learns to route future-token information from different regions of the pretrained layer stack.