MoSE: Mixture of Slimmable Experts for Efficient and Adaptive Language Models

Nurbek Tastan [1] **Stefanos Laskaridis** [†2] **Karthik Nandakumar** [1 3] **Samuel Horváth** [1]

Abstract

Mixture-of-Experts (MoE) models scale large language models efficiently by sparsely activating experts, but once an expert is selected, it is executed fully. Hence, the trade-off between accuracy and computation in an MoE model typically exhibits large discontinuities. We propose Mixture of Slimmable Experts (MoSE), an MoE architecture in which each expert has a nested, slimmable structure that can be executed at variable widths. This enables conditional computation not only over which experts are activated but also over how much of each expert is utilized. Consequently, a single pretrained MoSE model can support a more continuous spectrum of accuracy-compute trade-offs at inference time. We present a simple and stable training recipe for slimmable experts under sparse routing, combining multi-width training with standard MoE objectives. During inference, we explore strategies for runtime width determination, including a lightweight test-time training mechanism that learns how to map router confidence/probabilities to expert widths under a fixed budget. Experiments on GPT-style models, various routing regimes, zero-shot downstream reasoning benchmarks, and continual pre-training adaptation of DeepSeek model show that MoSE matches or improves standard MoE at full width and consistently shifts the compute-quality frontier toward lower inference FLOPs. The code can be found at: https://github.com/tnurbek/mose.

1. Introduction

Large language models (LLMs) continue to benefit from scale, but the associated training and inference costs (compute, energy, and latency) increasingly constrain their deployment and accessibility (Radford et al., 2019; Brown et al., 2020; Grattafiori et al., 2024). Mixture-of-Experts (MoE) architectures address this challenge by sparsely activating experts, allowing parameter counts to grow without proportional increases in per-token computation. This conditional computation principle dates back to sparsely-gated MoE layers (Jordan & Jacobs, 1994; Shazeer et al., 2017) and has been refined in large-scale systems such as switch transformers and subsequent MoE LLMs (Fedus et al., 2022; DeepSeek-AI et al., 2024a; Jiang et al., 2024; DeepSeek-AI et al., 2024b; Agarwal et al., 2025; Yang et al., 2025). As a result, MoE models have become a dominant paradigm for scaling LLMs efficiently (Du et al., 2022; Jiang et al., 2024; Agarwal et al., 2025).

However, despite sparse expert selection, once an expert is activated, it is almost always executed at full capacity. This design choice leaves a substantial dimension of conditional computation unexplored: the ability to adapt the amount of computation allocated within each activated expert. In practice, different tokens, contexts, and experts vary widely in their computational needs, suggesting that expert capacity itself should be elastic rather than fixed.

We introduce **Mixture of Slimmable Experts (MoSE)**, an MoE architecture that equips each expert with slimmable widths via nested subnetworks. MoSE generalizes conditional computation in MoE models by decoupling expert **selection** from expert **capacity**: the router determines which experts are active, while execution width controls how much of each expert is used. This enables a single pretrained model to flexibly trade off accuracy and compute at inference time, without retraining or modifying expert parameters. For example, one can:

(i) run all activated experts at a uniform width to obtain a smooth quality-compute curve, or
(ii) adapt widths per token based on router outputs.

Beyond uniform width control, MoSE enables adaptive width allocation conditioned on router confidence. We further propose a lightweight **test-time training** procedure that learns a low-dimensional mapping from router outputs to expert widths under a fixed compute budget, while keep-

[†]Work done independently of Amazon. [1]Mohamed bin Zayed University of Artificial Intelligence (MBZUAI), UAE [2]Amazon Science, UK [3]Michigan State University (MSU), USA. Correspondence to: Nurbek Tastan <nurbek.tastan@mbzuai.ac.ae>.

AdaptFM: Resource-Adaptive Foundation Model Inference Workshop at ICML 2026, Seoul, South Korea. 2026.

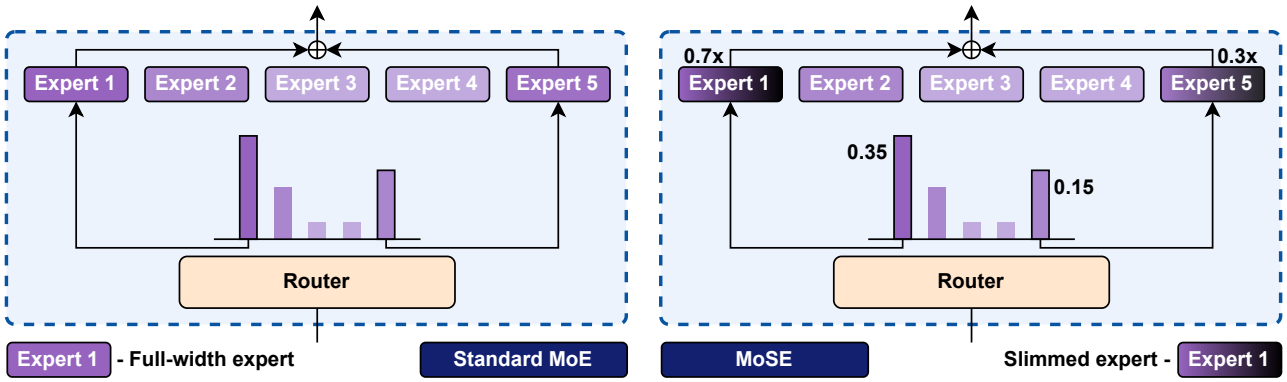

Figure 1. Comparison between standard MoE and MoSE. In the standard case, the router selects a fixed number of full-width experts and activates all their parameters. In the proposed method, the router not only selects multiple experts but also adjusts their widths, allowing more experts to contribute under the same parameter budget. This increases expert diversity without increasing total compute cost, potentially improving model accuracy at the same efficiency.

ing all model parameters frozen. This allows MoSE to automatically allocate computation where it is most beneficial at inference time, yielding consistent improvements in compute-quality trade-offs with negligible adaptation cost.

MoSE is especially natural for MoE models because routing probabilities already provide a token-specific signal of expert importance. Experts with lower routing probabilities contribute less to the final output and can therefore be executed at reduced widths with a limited effect on quality. At the same time, expert FFNs dominate MoE-layer computation due to their large hidden dimension expansion. Width adaptation, therefore, targets the main compute bottleneck while leveraging information that is already present in the routing distribution. We make the following contributions:

- We propose **MoSE**, an MoE architecture with slimmable experts that enables conditional computation over both expert selection and expert capacity.
- We present a simple and stable training recipe for slimmable experts under sparse routing, requiring no changes to the standard MoE pipeline.
- We introduce inference-time width allocation strategies, including a lightweight **test-time training** method that learns compute-aware width assignment under fixed budgets.
- Through extensive experiments on GPT-style and modern pretrained MoE models, including fine-grained routing regimes, zero-shot downstream reasoning tasks, and continual-pretraining adaptation, we show that MoSE consistently improves the compute-quality frontier at comparable or lower inference FLOPs.

2. Methodology

2.1. Mixture-of-Experts

We consider a decoder-only transformer in which the feed-forward network (FFN) in each block is replaced by an MoE layer. The MoE layer consists of n experts $(\mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_n)$ and a router/gating network $\mathcal{G}$. The experts are neural networks, each with their own parameters. For a token representation (input) $\boldsymbol{x} \in \mathbb{R}^d$, a gating network $\mathcal{G}$ produces scores $\mathcal{G}(\boldsymbol{x})$ over n experts and selects a sparse subset of experts, thereby saving computations through the sparsity of $\mathcal{G}(\boldsymbol{x})$ (typically using top-k). Let us denote $\mathcal{E}_i(\boldsymbol{x})$ as the output of the i-th expert network for the given input $\boldsymbol{x}$. The output $\boldsymbol{y}$ of the MoE module can be written as follows:

$$\boldsymbol{y} = \sum_{i=1}^{n} \mathcal{G}(\boldsymbol{x})_i \mathcal{E}_i(\boldsymbol{x}). \tag{1}$$

We save computation whenever $\mathcal{G}(\boldsymbol{x})_i = 0$, since we do not need to compute $\mathcal{E}_i(\boldsymbol{x})$.

2.2. MoSE: Mixture of Slimmable Experts

MoSE extends the standard MoE formulation by equipping each expert with **slimmable** (ordered) widths. Instead of executing a selected expert at full capacity, MoSE allows for controlling how much of each expert is used by adjusting its internal width. This introduces an additional axis of conditional computation: beyond selecting which experts are active, the model can adapt the **capacity** of each active expert.

Each expert $\mathcal{E}_i$ in MoSE is implemented as a transformer FFN with an expansion ratio of 4 (most language models follow this setting (Radford et al., 2019; Fedus et al., 2022; Touvron et al., 2023)), consisting of two fully connected layers. For $\boldsymbol{x} \in \mathbb{R}^d$, a full-width expert computes:

$$\mathcal{E}_i(\boldsymbol{x}) = \phi\left(\boldsymbol{x} W_{\text{up}}^{(i)} + b_{\text{up}}^{(i)}\right) W_{\text{down}}^{(i)} + b_{\text{down}}^{(i)}, \tag{2}$$

where $W_{\text{up}}^{(i)} \in \mathbb{R}^{d \times 4d}$ and $W_{\text{down}}^{(i)} \in \mathbb{R}^{4d \times d}$, $\phi(\cdot)$ denotes a nonlinear activation function (e.g., GELU (Hendrycks & Gimpel, 2016)). The intermediate hidden dimension dominates the computational cost of the expert, making it the natural target for slimmability.

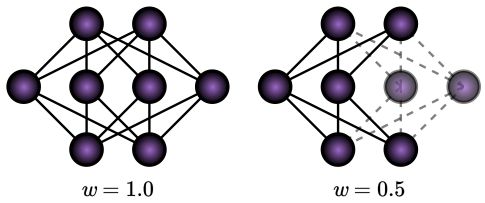

Figure 2. Slimmable expert in MoSE. Example with $w = 0.5$, where only half of the intermediate units of the expert FFN are activated by slicing the hidden dimension.

To enable slimmability, we define a discrete set of width multipliers (this can also be a continuous set):

$$\mathcal{A} = \{w_1, \ldots, w_m\}, \quad 0 < w_1 < \cdots < w_r = 1.0, \quad (3)$$

and let $m(w) = \lceil w \cdot 4d \rceil$ denote the active hidden size. The width-w version of expert $\mathcal{E}_i$, denoted as $\mathcal{E}_i^w$, is obtained by slicing the intermediate dimension:

$$\begin{aligned} W_{\text{up}}^{(i)}(w) &= W_{\text{up}}^{(i)}[:, : m(w)], \\ W_{\text{down}}^{(i)}(w) &= W_{\text{down}}^{(i)}[: m(w), :]. \end{aligned} \quad (4)$$

Figure 2 illustrates this mechanism for $w = 0.5$. All widths share parameters through this nested structure.

MoSE forward computation. Given $\boldsymbol{x}$, the gating network $\mathcal{G}$ selects a sparse set of experts as in standard MoE (top-k). For each selected expert $\mathcal{E}_i$, MoSE additionally assigns a width $w \in \mathcal{A}$. The MoSE output is then

$$\boldsymbol{y} = \sum_{i=1}^{n} \mathcal{G}(\boldsymbol{x})_i \mathcal{E}_i^{w(\boldsymbol{x})}(\boldsymbol{x}), \quad (5)$$

where computation is skipped whenever $\mathcal{G}(\boldsymbol{x})_i = 0$, and the cost of each active expert scales with $m(w(\boldsymbol{x}))$. When all $w(\boldsymbol{x}) = 1.0$, MoSE reduces to standard MoE.

2.3. MoSE Pre-Training

Pre-training must ensure that experts perform well across multiple widths while preserving stable MoE routing. We optimize the standard language modeling objective, augmented with MoE-specific regularization and stochastic multi-width training.

To train a single MoSE model that is usable at many widths while keeping training overhead minimal, we sample two widths per iteration: (i) the full width $w_{\max}$ and (ii) one random width $w \sim \text{Uniform}(w_{\min}, w_{\max})$. Concretely, for each mini-batch, we run the model twice and backpropagate through the sum (or average) of the two losses. This strategy is inspired by random width training (Tastan et al., 2025; Horváth et al., 2021) in slimmable networks and provides a simple pre-training mechanism with low overhead compared to other sampling strategies that involve more than two widths at a time. Let $\mathcal{L}_{\text{LM}}(w)$ denote the standard autoregressive next-token loss evaluated when all MoSE experts run at width w. Our primary objective is

$$\begin{aligned} \mathcal{L}_{\text{LM}}^{\text{MoSE}} &= \frac{1}{2}\big(\mathcal{L}_{\text{LM}}(w_{\max}) + \mathcal{L}_{\text{LM}}(w)\big), \\ w &\sim \text{Uniform}(w_{\min}, w_{\max}). \end{aligned} \quad (6)$$

As in standard sparsely-gated MoE training (Shazeer et al., 2017), we incorporate auxiliary losses to stabilize routing and prevent expert collapse. Specifically, the auxiliary objective includes a load balancing loss, which encourages an even distribution of tokens across experts within a batch, and a router z-loss (Zoph et al., 2022), which penalizes large router logits and improves numerical stability.

Notably, although the training uses uniform widths, results show that the learned representations generalize well to non-uniform, router-conditioned allocations at inference time.

Practical regimes. Each MoSE training step evaluates one full-width configuration and one sampled width configuration, so the per-step overhead depends on the sampled width and is lower than that of two full-width passes. We consider two practical usage regimes: (i) **full pretraining**, which yields the strongest compute-quality frontier, and (ii) **continual pretraining (CPT)** adaptation on top of an existing pretrained MoE checkpoint, which provides a cheaper path to enable slimmability post hoc.

2.4. MoSE Inference

After pre-training, MoSE supports multiple inference-time execution modes that differ in how widths are assigned to activated experts. Throughout, let $\mathcal{S}(\boldsymbol{x})$ denote the sparse set of selected experts (i.e., the experts with non-zero routing weight, $\mathcal{G}(\boldsymbol{x}) \neq 0$). We assign widths only to the experts in this sparse set $\mathcal{S}(\boldsymbol{x})$.

(1) Uniform-width execution. The simplest mode uses a single global width $w \in [w_{\min}, w_{\max}]$ (or $w \in \mathcal{A}$ if discretized) for all activated experts ($w(x) = w$ in Equation 5). This provides a stable and easily controlled compute-quality trade-off and matches the width control used during pre-training.

(2) Router-conditioned widths (normalized probability mode). We next assign widths using the router probabilities. Let $p_i(\boldsymbol{x})$ denote router probabilities normalized over the selected experts:

$$p_i(\boldsymbol{x}) = \frac{\mathcal{G}(\boldsymbol{x})_i}{\sum_{j \in \mathcal{S}(\boldsymbol{x})} \mathcal{G}(\boldsymbol{x})_j}, \quad i \in \mathcal{S}(\boldsymbol{x}). \quad (7)$$

Given a sharpness parameter $\gamma > 0$, we define un-normalized width scores:

$$s_i(\boldsymbol{x}; \gamma) = p_i(\boldsymbol{x})^{\gamma}, \quad i \in \mathcal{S}(\boldsymbol{x}), \quad (8)$$

and normalize them into allocation weights:

$$q_i(\boldsymbol{x};\gamma) = \frac{s_i(\boldsymbol{x};\gamma)}{\sum_{j\in\mathcal{S}(\boldsymbol{x})} s_j(\boldsymbol{x};\gamma)}. \tag{9}$$

Intuitively, γ controls how concentrated the width allocation is: $\gamma \to 0$ approaches uniform allocation across the selected experts, while larger γ increasingly concentrates width on higher-probability experts. In this normalized-probability mode, we fix $\gamma = 1.0$, yielding a direct proportional mapping from router probabilities to expert width. In contrast, the test-time training mode learns γ from data, allowing the probability-to-width mapping to adapt to the compute budget.

Budgeted width assignment. Given a per-token width budget $\Gamma \in (0, |\mathcal{S}(\boldsymbol{x})|)$ measured in "full-expert widths" (e.g., $\Gamma = |\mathcal{S}(\boldsymbol{x})|$ matches standard MoE with full width experts), we assign widths:

$$\tilde{w}_i(\boldsymbol{x};\gamma) = \Gamma \cdot q_i(\boldsymbol{x};\gamma), \quad i \in \mathcal{S}(\boldsymbol{x}). \tag{10}$$

We then map $\tilde{w}_i$ into a valid execution width by enforcing bounds and (optionally) discretizing (Equation 3):

$$w_i(\boldsymbol{x}) = \mathrm{Proj}_{[w_{\min}, w_{\max}]}(\tilde{w}_i(\boldsymbol{x};\gamma)), \tag{11}$$

where Proj denotes a simple projection/clipping.

Intuitively, Γ controls how much total compute is available for a token, while γ controls how that compute is distributed across the selected experts.

(3) Test-time training for width identification. Finally, we propose a lightweight test-time training (TTT) mechanism that learns how to map router probabilities to widths by optimizing the single scalar γ (shared across layers) or a small set $\{\gamma_\ell\}$ (layer-specific), while keeping all model and expert weights fixed. Concretely, for a short adaptation stream $\mathcal{D}_{\text{calib}}$, we minimize the language modeling loss under the router-conditioned width policy:

$$\gamma^\star = \arg\min_{\gamma>0} \mathbb{E}_{\boldsymbol{x}\in\mathcal{D}_{\text{calib}}} \left[\mathcal{L}_{\text{LM}}(\gamma;\Gamma)\right] \tag{12}$$

where $\mathcal{L}_{\text{LM}}(\gamma;\Gamma)$ is computed by running MoSE with widths set via Equations 7-10 under a chosen budget Γ. Since γ is extremely low-dimensional, this adaptation is lightweight and is performed without modifying the pretrained model parameters. In deployment, we then fix $\gamma = \gamma^\star$ and use it to assign widths for subsequent inputs.

3. Experiments

3.1. Experimental Setup

All experiments are conducted using a sparsely-gated mixture-of-experts (MoE) training framework (Shazeer et al., 2017). Models are pre-trained on the OpenWebText corpus (Gokaslan et al., 2019) using top-k routing, and all MoSE models are trained using uniform-width execution.

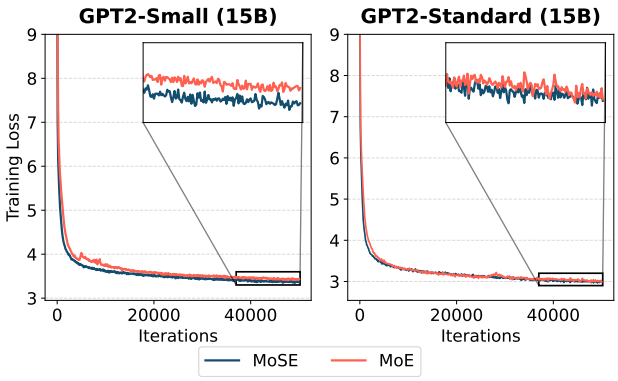

Figure 3. Pre-training dynamics of MoE and MoSE on OpenWebText dataset using GPT2-Small model.

We evaluate MoSE across a range of model scales and routing granularities, spanning GPT2-Small (55M), GPT2-Standard (322M), GPT2-Medium (1B), and GPT2-Large (2.5B) (Radford et al., 2019), together with finer-grained expert settings (E64A8 and E128A8) and continual-pretraining adaptation on top of pretrained MoE checkpoints, including DeepSeek-V2-Lite (16B) (DeepSeek-AI et al., 2024a). Unless otherwise specified, the main experiments use routing configurations E8A2, E8A4, and E16A4 with training budgets ranging from 3B to 15B tokens. Exact architectural parameters, routing hyperparameters, and per-experiment training schedules are provided in Appendix B.

After pre-training, models are evaluated under several inference-time execution modes (detailed in Section 2.4), including (1) uniform-width execution, (2) router-conditioned widths with fixed γ, and (3) test-time training for widths, where the sharpness parameter γ (shared or layer-wise) is learned using a short calibration stream while keeping all pretrained weights fixed. Inference-time compute is reported in MFLOPs per token.

We report language modeling perplexity on OpenWebText (Gokaslan et al., 2019) and WikiText-103 (Merity et al., 2017), as well as zero-shot performance on LAMBADA (accuracy and perplexity) (Paperno et al., 2016), the Winograd Schema Challenge (WSC) (Levesque et al., 2012), HellaSwag (Zellers et al., 2019), PIQA (Bisk et al., 2020), and Social IQA (Sap et al., 2019). All downstream evaluations are performed without task-specific fine-tuning. Further experimental details are deferred to Appendix B.

3.2. MoSE Pre-Training Results

In this section, we study the pre-training behavior of MoSE on the OpenWebText dataset, focusing on optimization stability and convergence relative to the MoE baseline.

Training dynamics and stability. Figure 3 shows the evo-

Table 1. Pre-training and zero-shot evaluation results of MoE and MoSE across model sizes and training budgets. Results are reported on OpenWebText and WikiText-103 for language modeling perplexity, and on LAMBADA and Winograd Schema Challenge (WSC) for zero-shot evaluation. * denotes lower FLOPs at comparable or better performance (Section 3.3). The best results are highlighted in bold.

Model	Tokens	Method	OpenWebText [ppl]	WikiText-103 [ppl]	LAMBADA [acc]	LAMBADA [ppl]	WSC [acc]
GPT2-Small (55M)	3B	MoE	41.82	160.13	0.152	123.527	0.4982
		MoSE (w=1.0)	39.02	154.85	0.145	113.031	0.5092
		MoSE* (TTT)	**38.48**	**153.24**	**0.164**	**110.129**	**0.5201**
	15B	MoE	31.03	120.48	0.187	65.413	0.5165
		MoSE (w=1.0)	31.26	121.50	0.203	**50.437**	0.5421
		MoSE* (TTT)	**30.81**	**120.20**	**0.214**	51.085	**0.5458**
GPT2-Standard (322M)	3B	MoE	27.75	**89.22**	0.206	39.808	**0.5201**
		MoSE (w=1.0)	27.66	89.56	0.217	39.075	0.5128
		MoSE* (TTT)	**27.32**	89.41	**0.224**	**39.060**	0.5165
	15B	MoE	20.81	77.22	0.294	18.280	0.5311
		MoSE (w=1.0)	20.65	**76.93**	0.336	16.516	0.5495
		MoSE* (TTT)	**20.38**	76.99	**0.338**	**16.430**	**0.5531**

lution of training loss as a function of iterations for MoE and MoSE under 15B token budget. Across both settings, MoSE closely tracks the convergence trajectory of MoE throughout training. This behavior indicates that introducing width-adaptive execution does not hinder optimization or destabilize pre-training. These results show that MoSE can be trained end-to-end within standard MoE pre-training pipelines without additional modifications.

Last-iterate pre-training performance. Table 1 summarizes the outcomes of MoE and MoSE across model sizes and training token budgets, evaluated on OpenWebText (test set) as well as multiple downstream benchmarks. For MoSE, we report two variants: (i) uniform-width execution ($w = 1.0$), which assigns full width to all activated experts, and (ii) MoSE with test-time training for width identification (with a maximum budget $\Gamma = |\mathcal{S}(\boldsymbol{x})|$). For the latter, we report the layer-wise γ variant throughout.

Across all settings, MoSE with uniform execution already matches or improves upon the MoE baseline, indicating that the architectural flexibility introduced by MoSE does not degrade representational capacity. More notably, when coupled with test-time training, MoSE consistently achieves the best performance (despite using lower FLOPs; discussed in later sections), yielding lower perplexity on OpenWebText and WikiText-103, as well as improved zero-shot performance on LAMBADA and WSC. These gains hold across both 3B and 15B token regimes and scale with model size, showing that the learned width identification mode complements pre-training rather than interfering with it.

3.3. MoSE Inference Results

In this section, we evaluate MoSE in the inference-time regime, where expert routing is fixed by a pre-trained router and the model dynamically assigns execution widths to the activated experts. Unless otherwise specified, all experiments report compute-quality trade-offs measured in MFLOPs per token versus validation perplexity, and Pareto frontiers are constructed by varying the width budget while keeping model parameters fixed. We compare MoSE (TTT) against uniform-width execution and vanilla MoE.

Figure 4 compares MoSE across GPT2-Small, GPT2-Standard, and GPT2-Medium models under a fixed routing setting (E8A2; $n = 8, k = 2$). Across all model sizes, MoSE (TTT) consistently shifts the Pareto frontier downward relative to uniform-width execution, achieving lower perplexity at comparable compute. This demonstrates that the proposed width identification mechanism scales effectively with model size and does not rely on model-specific tuning. Notably, the relative improvement remains stable as the base model grows, indicating that test-time training learns a compute-aware width allocation strategy that works across scales.

3.4. Extended Evaluation: Scale, Adaptation

We next extend the empirical scope of MoSE along three axes that are important for practical deployment: (i) larger backbones and finer-grained MoE routing regimes (see Appendix D.7), (ii) post-pretraining slimmability adaptation on top of existing pretrained MoE checkpoints, and (iii) transfer beyond perplexity to zero-shot downstream reasoning benchmarks (see Appendix D.8).

To evaluate post-pretraining deployment, we additionally study continual-pretraining (CPT) slimmability adaptation starting from pretrained MoE checkpoints. Figure 5 shows that MoSE remains effective on DeepSeek-V2-Lite (16B), improving the inference-time Pareto frontier by 38.1% at comparable perplexity, corresponding to approximately 920 MFLOPs/token absolute savings. Detailed CPT results, including validation perplexity across widths and additional post-pretraining comparisons, are deferred to Appendix E.

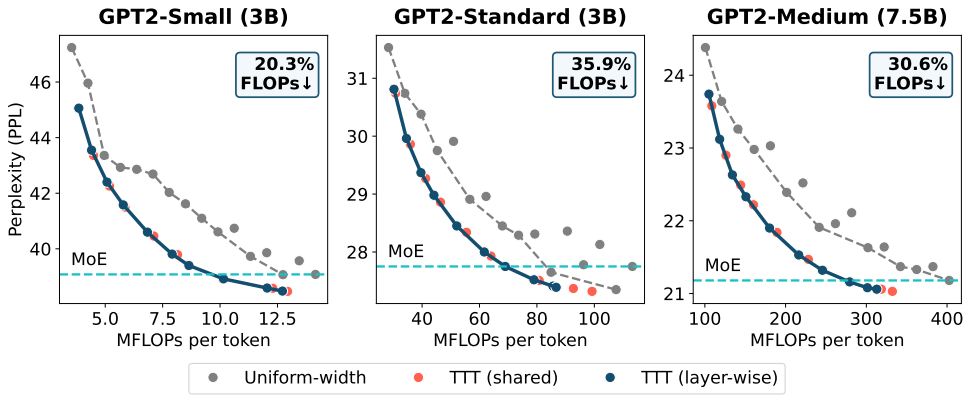

Figure 4. Compute-quality trade-offs across GPT2-Small, Standard, and Medium models under **E8A2** setting. MoSE with test-time training learns compute-aware width identification that shift the Pareto frontier, achieving lower perplexity than **uniform-width** mode

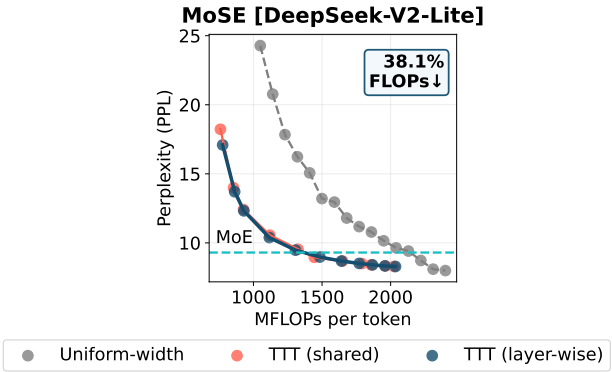

Figure 5. Pareto frontier on DeepSeek-V2-Lite (16B) after continual-pretraining slimmability adaptation. MoSE achieves 38.1% lower FLOPs at matched perplexity, corresponding to approximately 920 MFLOPs/token absolute savings.

4. Related Work

Mixture-of-Experts Transformers. Mixture-of-Experts (MoE) models date back to classical conditional computation via expert specialization and learned gating (Jacobs et al., 1991; Jordan & Jacobs, 1994). Modern deep learning MoE revived this idea at scale by introducing sparsely-gated expert layers, enabling large parameter counts without proportional increases in per-token compute (Shazeer et al., 2017; Fedus et al., 2022). Early work on sparsely gated MoE layers demonstrated the feasibility of conditional computation, and subsequent works such as switch transformers (Fedus et al., 2022), GLaM (Du et al., 2022), and V-MoE (Riquelme et al., 2021) established MoE as a practical approach for large-scale transformer pretraining. Across these works, the dominant design choice is to replace transformer FFN layers with expert layers and select a fixed number of experts (e.g., top-k) per token. A common limitation of this paradigm is that once an expert is selected, it is executed at full capacity, even though the routing itself is sparse.

Slimmable Networks. Slimmable networks train a single set of parameters that can be executed at multiple widths, enabling runtime accuracy-efficiency trade-offs without training separate models (Rippel et al., 2014; Yu et al., 2019; Yu & Huang, 2019b; Horváth et al., 2021; Kusupati et al., 2022; Devvrit et al., 2023; Tastan et al., 2025). Related "train-once, deploy-many" approaches include once-for-all networks, which optimize a super-network that can be specialized into multiple subnetworks post hoc (Cai et al., 2020; 2025). These ideas have inspired a broad range of follow-up work, mostly focused on resource efficiency (Horváth et al., 2021; Mei et al., 2022), communication and computational efficiency (Wang et al., 2022), collaborative fairness (Tastan et al., 2025), and neural architecture search (Yu & Huang, 2019a).

Elastic MoEs. Recent work on elastic Mixture-of-Experts (MoE) tackles the long-standing rigidity of Top-k routing, where models trained with a fixed number of active experts fail to generalize when the inference budget changes. RoE (Roster of Experts) (Zibakhsh et al., 2025) approaches elasticity without retraining by injecting stochasticity into the router and aggregating multiple diverse forward passes, effectively turning a single MoE into a test-time ensemble; while this improves robustness and quality under larger compute budgets, it does not change the underlying expert representations and incurs additional inference costs proportional to the number of passes. In contrast, Elastic MoE (EMoE) (Gu et al., 2025) modifies training to expose experts to diverse co-activation patterns via stochastic sampling and hierarchical router regularization, enabling the model to tolerate larger inference-time k than seen during training; however, its gains depend on carefully designed routing losses and still assume a flat expert set without explicitly structuring expert importance.

MoSE bridges these lines of work by equipping MoE layers with slimmable widths, enabling conditional computation not only over which experts are activated but also over the internal capacity used by each activated expert.

Impact Statement

This paper presents work aimed at advancing the field of Machine Learning. MoSE enables adaptive inference-time trade-offs between compute and model capacity in mixture-of-experts language models, which may improve accessibility and efficiency across deployment settings. At the same time, the slimmable nature of the model introduces an additional axis of conditional behavior, and future safety-alignment and robustness evaluations should account not only for the base model behavior but also for behavior across different execution widths and compute regimes. Beyond this consideration, we are not aware of any additional societal consequences that require specific discussion.

References

Agarwal, S., Ahmad, L., Ai, J., Altman, S., Applebaum, A., Arbus, E., Arora, R. K., Bai, Y., Baker, B., Bao, H., Barak, B., Bennett, A., Bertao, T., Brett, N., Brevdo, E., Brockman, G., Bubeck, S., Chang, C., Chen, K., Chen, M., Cheung, E., Clark, A., Cook, D., Dukhan, M., Dvorak, C., Fives, K., Fomenko, V., Garipov, T., Georgiev, K., Glaese, M., Gogineni, T., Goucher, A., Gross, L., Guzman, K. G., Hallman, J., Hehir, J., Heidecke, J., Helyar, A., Hu, H., Huet, R., Huh, J., Jain, S., Johnson, Z., Koch, C., Kofman, I., Kundel, D., Kwon, J., Kyrylov, V., Le, E. Y., Leclerc, G., Lennon, J. P., Lessans, S., Lezcano-Casado, M., Li, Y., Li, Z., Lin, J., Liss, J., Lily, Liu, Liu, J., Lu, K., Lu, C., Martinovic, Z., McCallum, L., McGrath, J., McKinney, S., McLaughlin, A., Mei, S., Mostovoy, S., Mu, T., Myles, G., Neitz, A., Nichol, A., Pachocki, J., Paino, A., Palmie, D., Pantuliano, A., Parascandolo, G., Park, J., Pathak, L., Paz, C., Peran, L., Pimenov, D., Pokrass, M., Proehl, E., Qiu, H., Raila, G., Raso, F., Ren, H., Richardson, K., Robinson, D., Rotsted, B., Salman, H., Sanjeev, S., Schwarzer, M., Sculley, D., Sikchi, H., Simon, K., Singhal, K., Song, Y., Stuckey, D., Sun, Z., Tillet, P., Toizer, S., Tsimpourlas, F., Vyas, N., Wallace, E., Wang, X., Wang, M., Watkins, O., Weil, K., Wendling, A., Whinnery, K., Whitney, C., Wong, H., Yang, L., Yang, Y., Yasunaga, M., Ying, K., Zaremba, W., Zhan, W., Zhang, C., Zhang, B., Zhang, E., and Zhao, S. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*, 2025.

Bisk, Y., Zellers, R., Le bras, R., Gao, J., and Choi, Y. PIQA: Reasoning about Physical Commonsense in Natural Language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 7432–7439, Apr. 2020. doi: 10.1609/aaai.v34i05.6239. URL https://ojs.aaai.org/index.php/AAAI/article/view/6239.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901. Curran Associates, Inc., 2020.

Cai, H., Gan, C., Wang, T., Zhang, Z., and Han, S. Once-for-all: Train one network and specialize it for efficient deployment. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=HylxE1HKwS.

Cai, R., Muralidharan, S., Yin, H., Wang, Z., Kautz, J., and Molchanov, P. LLamaflex: Many-in-one LLMs via generalized pruning and weight sharing. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=AyC4uxx2HW.

Cai, T., Li, Y., Geng, Z., Peng, H., Lee, J. D., Chen, D., and Dao, T. Medusa: Simple LLM inference acceleration framework with multiple decoding heads. In Salakhutdinov, R., Kolter, Z., Heller, K., Weller, A., Oliver, N., Scarlett, J., and Berkenkamp, F. (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 5209–5235. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/cai24b.html.

DeepSeek-AI, Liu, A., Feng, B., Wang, B., Wang, B., Liu, B., Zhao, C., Dengr, C., Ruan, C., Dai, D., Guo, D., Yang, D., Chen, D., Ji, D., Li, E., Lin, F., Luo, F., Hao, G., Chen, G., Li, G., Zhang, H., Xu, H., Yang, H., Zhang, H., Ding, H., Xin, H., Gao, H., Li, H., Qu, H., Cai, J. L., Liang, J., Guo, J., Ni, J., Li, J., Chen, J., Yuan, J., Qiu, J., Song, J., Dong, K., Gao, K., Guan, K., Wang, L., Zhang, L., Xu, L., Xia, L., Zhao, L., Zhang, L., Li, M., Wang, M., Zhang, M., Zhang, M., Tang, M., Li, M., Tian, N., Huang, P., Wang, P., Zhang, P., Zhu, Q., Chen, Q., Du, Q., Chen, R. J., Jin, R. L., Ge, R., Pan, R., Xu, R., Chen, R., Li, S. S., Lu, S., Zhou, S., Chen, S., Wu, S., Ye, S., Ma, S., Wang, S., Zhou, S., Yu, S., Zhou, S., Zheng, S., Wang, T., Pei, T., Yuan, T., Sun, T., Xiao, W. L., Zeng, W., An, W., Liu, W., Liang, W., Gao, W., Zhang, W., Li, X. Q., Jin, X., Wang, X., Bi, X., Liu, X., Wang, X., Shen, X., Chen, X., Chen, X., Nie, X., Sun, X., Wang, X., Liu, X., Xie, X., Yu, X., Song, X., Zhou, X., Yang, X., Lu, X., Su, X., Wu, Y., Li, Y. K., Wei, Y. X., Zhu, Y. X., Xu, Y., Huang, Y., Li, Y., Zhao, Y., Sun, Y., Li, Y., Wang, Y., Zheng, Y., Zhang, Y., Xiong, Y., Zhao, Y., He,

Y., Tang, Y., Piao, Y., Dong, Y., Tan, Y., Liu, Y., Wang, Y., Guo, Y., Zhu, Y., Wang, Y., Zou, Y., Zha, Y., Ma, Y., Yan, Y., You, Y., Liu, Y., Ren, Z. Z., Ren, Z., Sha, Z., Fu, Z., Huang, Z., Zhang, Z., Xie, Z., Hao, Z., Shao, Z., Wen, Z., Xu, Z., Zhang, Z., Li, Z., Wang, Z., Gu, Z., Li, Z., and Xie, Z. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*, 2024a.

DeepSeek-AI, Liu, A., Feng, B., Xue, B., Wang, B., Wu, B., Lu, C., Zhao, C., Deng, C., Zhang, C., Ruan, C., Dai, D., Guo, D., Yang, D., Chen, D., Ji, D., Li, E., Lin, F., Dai, F., Luo, F., Hao, G., Chen, G., Li, G., Zhang, H., Bao, H., Xu, H., Wang, H., Zhang, H., Ding, H., Xin, H., Gao, H., Li, H., Qu, H., Cai, J. L., Liang, J., Guo, J., Ni, J., Li, J., Wang, J., Chen, J., Chen, J., Yuan, J., Qiu, J., Li, J., Song, J., Dong, K., Hu, K., Gao, K., Guan, K., Huang, K., Yu, K., Wang, L., Zhang, L., Xu, L., Xia, L., Zhao, L., Wang, L., Zhang, L., Li, M., Wang, M., Zhang, M., Zhang, M., Tang, M., Li, M., Tian, N., Huang, P., Wang, P., Zhang, P., Wang, Q., Zhu, Q., Chen, Q., Du, Q., Chen, R. J., Jin, R. L., Ge, R., Zhang, R., Pan, R., Wang, R., Xu, R., Zhang, R., Chen, R., Li, S. S., Lu, S., Zhou, S., Chen, S., Wu, S., Ye, S., Ye, S., Ma, S., Wang, S., Zhou, S., Yu, S., Zhou, S., Pan, S., Wang, T., Yun, T., Pei, T., Sun, T., Xiao, W. L., Zeng, W., Zhao, W., An, W., Liu, W., Liang, W., Gao, W., Yu, W., Zhang, W., Li, X. Q., Jin, X., Wang, X., Bi, X., Liu, X., Wang, X., Shen, X., Chen, X., Zhang, X., Chen, X., Nie, X., Sun, X., Wang, X., Cheng, X., Liu, X., Xie, X., Liu, X., Yu, X., Song, X., Shan, X., Zhou, X., Yang, X., Li, X., Su, X., Lin, X., Li, Y. K., Wang, Y. Q., Wei, Y. X., Zhu, Y. X., Zhang, Y., Xu, Y., Xu, Y., Huang, Y., Li, Y., Zhao, Y., Sun, Y., Li, Y., Wang, Y., Yu, Y., Zheng, Y., Zhang, Y., Shi, Y., Xiong, Y., He, Y., Tang, Y., Piao, Y., Wang, Y., Tan, Y., Ma, Y., Liu, Y., Guo, Y., Wu, Y., Ou, Y., Zhu, Y., Wang, Y., Gong, Y., Zou, Y., He, Y., Zha, Y., Xiong, Y., Ma, Y., Yan, Y., Luo, Y., You, Y., Liu, Y., Zhou, Y., Wu, Z. F., Ren, Z. Z., Ren, Z., Sha, Z., Fu, Z., Xu, Z., Huang, Z., Zhang, Z., Xie, Z., Zhang, Z., Hao, Z., Gou, Z., Ma, Z., Yan, Z., Shao, Z., Xu, Z., Wu, Z., Zhang, Z., Li, Z., Gu, Z., Zhu, Z., Liu, Z., Li, Z., Xie, Z., Song, Z., Gao, Z., and Pan, Z. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024b.

Devvrit, F., Kudugunta, S., Kusupati, A., Dettmers, T., Chen, K., Dhillon, I. S., Tsvetkov, Y., Hajishirzi, H., Kakade, S. M., Farhadi, A., and Jain, P. Matformer: Nested transformer for elastic inference. In *Workshop on Advancing Neural Network Training: Computational Efficiency, Scalability, and Resource Optimization (WANT@ NeurIPS 2023)*, 2023.

Du, N., Huang, Y., Dai, A. M., Tong, S., Lepikhin, D., Xu, Y., Krikun, M., Zhou, Y., Yu, A. W., Firat, O., Zoph, B., Fedus, L., Bosma, M. P., Zhou, Z., Wang, T., Wang, E., Webster, K., Pellat, M., Robinson, K., Meier-Hellstern, K., Duke, T., Dixon, L., Zhang, K., Le, Q., Wu, Y., Chen, Z., and Cui, C. GLaM: Efficient Scaling of Language Models with Mixture-of-Experts. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S. (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 5547–5569. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/du22c.html.

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

Gokaslan, A., Cohen, V., Pavlick, E., and Tellex, S. Openwebtext corpus. http://Skylion007.github.io/OpenWebTextCorpus, 2019.

Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Vaughan, A., et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Gu, N., Zhang, Z., Feng, Y., Chen, Y., Fu, P., Lin, Z., Wang, S., Sun, Y., Wu, H., Wang, W., et al. Elastic moe: Unlocking the inference-time scalability of mixture-of-experts. *arXiv preprint arXiv:2509.21892*, 2025.

Hendrycks, D. and Gimpel, K. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

Horváth, S., Laskaridis, S., Almeida, M., Leontiadis, I., Venieris, S., and Lane, N. Fjord: Fair and accurate federated learning under heterogeneous targets with ordered dropout. *Advances in Neural Information Processing Systems*, 34:12876–12889, 2021.

Jacobs, R. A., Jordan, M. I., Nowlan, S. J., and Hinton, G. E. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991.

Jiang, A. Q., Sablayrolles, A., Roux, A., Mensch, A., Savary, B., Bamford, C., Chaplot, D. S., de las Casas, D., Hanna, E. B., Bressand, F., Lengyel, G., Bour, G., Lample, G., Lavaud, L. R., Saulnier, L., Lachaux, M.-A., Stock, P., Subramanian, S., Yang, S., Antoniak, S., Scao, T. L., Gervet, T., Lavril, T., Wang, T., Lacroix, T., and Sayed, W. E. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

Jordan, M. I. and Jacobs, R. A. Hierarchical mixtures of experts and the em algorithm. *Neural computation*, 6(2): 181–214, 1994.

Kusupati, A., Bhatt, G., Rege, A., Wallingford, M., Sinha, A., Ramanujan, V., Howard-Snyder, W., Chen, K., Kakade, S., Jain, P., et al. Matryoshka representation learning. *Advances in Neural Information Processing Systems*, 35:30233–30249, 2022.

Levesque, H. J., Davis, E., and Morgenstern, L. The winograd schema challenge. In *Proceedings of the Thirteenth International Conference on Principles of Knowledge Representation and Reasoning*, KR'12, pp. 552–561. AAAI Press, 2012. ISBN 9781577355601.

Leviathan, Y., Kalman, M., and Matias, Y. Fast Inference from Transformers via Speculative Decoding. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J. (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 19274–19286. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/leviathan23a.html.

Mei, Y., Guo, P., Zhou, M., and Patel, V. Resource-adaptive federated learning with all-in-one neural composition. *Advances in Neural Information Processing Systems*, 35: 4270–4284, 2022.

Merity, S., Xiong, C., Bradbury, J., and Socher, R. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Byj72udxe.

Miao, X., Oliaro, G., Zhang, Z., Cheng, X., Wang, Z., Zhang, Z., Wong, R. Y. Y., Zhu, A., Yang, L., Shi, X., Shi, C., Chen, Z., Arfeen, D., Abhyankar, R., and Jia, Z. Specinfer: Accelerating large language model serving with tree-based speculative inference and verification. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, ASPLOS '24, pp. 932–949, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400703867. doi: 10.1145/3620666.3651335. URL https://doi.org/10.1145/3620666.3651335.

Paperno, D., Kruszewski, G., Lazaridou, A., Pham, N. Q., Bernardi, R., Pezzelle, S., Baroni, M., Boleda, G., and Fernández, R. The LAMBADA dataset: Word prediction requiring a broad discourse context. In Erk, K. and Smith, N. A. (eds.), *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1525–1534, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1144. URL https://aclanthology.org/P16-1144/.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language models are unsupervised multitask learners. *OpenAI*, 2019. URL https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

Rippel, O., Gelbart, M., and Adams, R. Learning ordered representations with nested dropout. In *International Conference on Machine Learning*, pp. 1746–1754. PMLR, 2014.

Riquelme, C., Puigcerver, J., Mustafa, B., Neumann, M., Jenatton, R., Susano Pinto, A., Keysers, D., and Houlsby, N. Scaling vision with sparse mixture of experts. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 8583–8595. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/48237d9f2dea8c74c2a72126cf63d933-Paper.pdf.

Sap, M., Rashkin, H., Chen, D., Le Bras, R., and Choi, Y. Social IQa: Commonsense reasoning about social interactions. In Inui, K., Jiang, J., Ng, V., and Wan, X. (eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 4463–4473, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1454. URL https://aclanthology.org/D19-1454/.

Schaeffer, R., Miranda, B., and Koyejo, S. Are emergent abilities of large language models a mirage? In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=ITw9edRDlD.

Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q., Hinton, G., and Dean, J. Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=B1ckMDqlg.

Tastan, N., Horváth, S., and Nandakumar, K. Aequa: Fair Model Rewards in Collaborative Learning via Slimmable Networks. In Singh, A., Fazel, M., Hsu, D., Lacoste-Julien, S., Berkenkamp, F., Maharaj, T., Wagstaff, K., and Zhu, J. (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 59210–59236. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/tastan25a.html.

Tastan, N., Horváth, S., and Nandakumar, K. Stochastic self-organization in multi-agent systems. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=rS3Jb9AAej.

Tiwari, R., Xi, H., Tomar, A., Hooper, C. R. C., Kim, S., Horton, M., Najibi, M., Mahoney, M. W., Keutzer, K., and Gholami, A. Quantspec: Self-speculative decoding with hierarchical quantized KV cache. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=7SHbJENgHX.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., and Lample, G. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Wang, H.-P., Stich, S., He, Y., and Fritz, M. Progfed: Effective, communication, and computation efficient federated learning by progressive training. In *International Conference on Machine Learning*, pp. 23034–23054. PMLR, 2022.

Wu, Q., Bansal, G., Zhang, J., Wu, Y., Li, B., Zhu, E., Jiang, L., Zhang, X., Zhang, S., Liu, J., Awadallah, A. H., White, R. W., Burger, D., and Wang, C. Autogen: Enabling next-gen LLM applications via multi-agent conversations. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=BAakY1hNKS.

Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., Zheng, C., Liu, D., Zhou, F., Huang, F., Hu, F., Ge, H., Wei, H., Lin, H., Tang, J., Yang, J., Tu, J., Zhang, J., Yang, J., Yang, J., Zhou, J., Zhou, J., Lin, J., Dang, K., Bao, K., Yang, K., Yu, L., Deng, L., Li, M., Xue, M., Li, M., Zhang, P., Wang, P., Zhu, Q., Men, R., Gao, R., Liu, S., Luo, S., Li, T., Tang, T., Yin, W., Ren, X., Wang, X., Zhang, X., Ren, X., Fan, Y., Su, Y., Zhang, Y., Zhang, Y., Wan, Y., Liu, Y., Wang, Z., Cui, Z., Zhang, Z., Zhou, Z., and Qiu, Z. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Yu, J. and Huang, T. Autoslim: Towards one-shot architecture search for channel numbers. *arXiv preprint arXiv:1903.11728*, 2019a.

Yu, J. and Huang, T. S. Universally Slimmable Networks and Improved Training Techniques. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019b.

Yu, J., Yang, L., Xu, N., Yang, J., and Huang, T. Slimmable Neural Networks. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=H1gMCsAqY7.

Zellers, R., Holtzman, A., Bisk, Y., Farhadi, A., and Choi, Y. HellaSwag: Can a machine really finish your sentence? In Korhonen, A., Traum, D., and Màrquez, L. (eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4791–4800, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1472. URL https://aclanthology.org/P19-1472/.

Zibakhsh, S., Samragh, M., Nishu, K., Hannah, L., Kundu, A., and Cho, M. Moes are stronger than you think: Hyper-parallel inference scaling with roe. *arXiv preprint arXiv:2509.17238*, 2025.

Zoph, B., Bello, I., Kumar, S., Du, N., Huang, Y., Dean, J., Shazeer, N., and Fedus, W. St-moe: Designing stable and transferable sparse expert models. *arXiv preprint arXiv:2202.08906*, 2022.

A. Discussion & Limitations

MoSE provides an additional axis of conditional computation in MoE architectures, which allows for an additional degree of freedom in dedicating representational capacity at deployment time, paving the way for novel optimizations for LLMs. For example, MoSE could be leveraged as a mechanism of self-speculation (Leviathan et al., 2023; Miao et al., 2024; Cai et al., 2024; Tiwari et al., 2025). A lightweight "draft" model would be implicitly realized by executing the same model at reduced widths, thus accelerating inference without the need to retrain for explicit alignment. Additionally, in agentic settings (Wu et al., 2024; Tastan et al., 2026), an agent could select the model width depending on perceived task difficulty or uncertainty. Therefore, smaller widths could be used for easy steps, while additional computation could be leveraged under high-uncertainty and complex tasks. This equips MoE models with an adaptive dimension that can save on latency without sacrificing performance.

The current empirical validation covers GPT-style models trained from scratch, a larger 2.5B/50B setting, finer-grained routing regimes, zero-shot downstream reasoning benchmarks, and continual pre-training adaptation of DeepSeek-V2-Lite (16B). A remaining limitation is that we do not yet report full from-scratch training at frontier MoE scale.

B. Experimental Details

B.1. Datasets and Evaluation Tasks

This section details the datasets used for pre-training and evaluation, the evaluation protocol, and the metrics we report.

Pre-training. All models are pre-trained on the OpenWebText corpus (Gokaslan et al., 2019), a large-scale corpus of English web text commonly used for training GPT-style language models. We use OpenWebText exclusively for pre-training in all experiments to ensure that differences in performance arise from model design and inference-time execution rather than changes in training data. We pre-train models for multiple token budgets depending on model scale and routing configuration (see Table 2). For continual pre-training (CPT) experiments, we initialize from a pretrained non-slimmable MoE checkpoint and continue on the same OpenWebText corpus with the same optimization and training recipe.

Evaluation. We evaluate both language modeling performance and zero-shot transfer on a set of standard benchmarks. All evaluations are conducted **zero-shot** (i.e., without any task-specific fine-tuning). We use the following datasets:

- **OpenWebText**: We report perplexity on a held-out split of OpenWebText to measure in-domain pre-training quality.
- **WikiText-103**: A Wikipedia-based language modeling benchmark used to assess out-of-domain generalization. We report perplexity on the standard evaluation split.
- **LAMBADA**: A dataset designed to test long-range context understanding via next-word prediction. We report both zero-shot **accuracy** (exact match next-word prediction) and **perplexity**. The validation split of this dataset consists of 5153 entries.
- **Winograd Schema Challenge (WSC)**: A pronoun resolution benchmark designed to test commonsense reasoning. We evaluate in the zero-shot setting and report accuracy. The validation split of this dataset consists of 273 entries.
- **HellaSwag:** A commonsense completion benchmark with four candidate endings. We evaluate in the zero-shot multiple-choice setting and report accuracy.
- **PIQA:** A physical commonsense reasoning benchmark with two candidate solutions. We evaluate in the zero-shot multiple-choice setting and report accuracy.
- **Social IQA (SIQA):** A social commonsense reasoning benchmark with three candidate answers. We evaluate in the zero-shot multiple-choice setting and report accuracy.

For all evaluations, we keep the pretrained model parameters fixed. We do not perform fine-tuning on any downstream task. For MoSE with test-time training, the only adapted parameters are the *sharpness* parameters γ (either shared across transformer blocks or layer-wise), while all model weights, expert weights, and router parameters remain unchanged.

For MoSE with test-time training, we learn γ by minimizing the language modeling loss over a short calibration stream $\mathcal{D}_{\text{calib}}$ (drawn from any available dataset, e.g., OpenWebText in our settings). The calibration optimizes only γ under a given budget Γ, and then reuses the learned $\gamma^\star$ to evaluate the model under that budget. We emphasize that this procedure is lightweight because γ is extremely low-dimensional (one scalar per layer in the layer-wise variant, or a single scalar in the shared variant). In our experiments, we limit the calibration dataset to only 50 batches, with a batch size of 6.

Metrics. We report the following metrics across datasets:

- **Perplexity (PPL):** For language modeling datasets (OpenWebText, WikiText-103, and LAMBADA), we report perplexity computed from the average negative log-likelihood. Lower is better.
- **Accuracy (Acc):** For LAMBADA and WSC, we report accuracy, corresponding to exact-match prediction.

Compute reporting. Alongside task metrics, we report inference-time compute in **MFLOPs per token**. Compute is measured by counting the floating-point operations incurred by the transformer forward pass, including expert computation under the given routing and width policy. We use MFLOPs per token to compare compute-quality trade-offs across inference modes and to trace Pareto frontiers by varying the budget Γ.

B.2. Hyperparameters

This section describes the hyperparameters shared across all experiments. A summary of model architectures, routing settings, pretraining token budgets, and training schedules is provided in Table 2. Unless otherwise stated, the hyperparameters described below are held fixed across model sizes and routing configurations.

Model architectures and routing. We evaluate three GPT-style model scales: GPT2-Small, GPT2-Standard, GPT2-Medium, and GPT2-Large under multiple routing configurations, denoted by $\mathrm{E}n\mathrm{A}k$, where n is the number of experts and k is the number of activated experts per token. We additionally evaluate finer grained expert settings and continual pre-training adaptation on top of pretrained MoE checkpoints, including DeepSeek-V2-Lite (16B parameters). The specific combinations of model scale, routing setting, parameter count, and training token budget used in our experiments are summarized in Table 2. The sequence length is fixed to 1024 tokens for all experiments. These configurations are chosen to systematically vary routing capacity and model scale while maintaining comparable training settings.

Pretraining optimization. We optimize all models using the AdamW optimizer with a learning rate 6×10^{-4}, weight decay 10^{-1}, and momentum parameters $(\beta_1, \beta_2) = (0.9, 0.95)$. Gradients are clipped to a maximum norm of 1.0. We use a linear learning rate schedule with warmup, followed by linear decay to a minimum learning rate of 6×10^{-5}. The warmup lasts for 200 iterations.

All runs use top-k expert routing with auxiliary regularization; we apply a load-balancing loss with a coefficient 0.01 and a router z-loss with a coefficient 0.001. Training and evaluation capacities are set to 1.25 and 2.0, respectively, across runs with $k = 2$; when $k = 4$, we increase the capacities to 1.75 and 3.0 respectively.

Slimmable training. For MoSE, the hidden dimension of MoE experts is made slimmable. We set $w_{\min} = 0.25$ and $w_{\max} = 1.0$. For practical reasons, we discretize this range $\mathcal{A} \in [w_{\min}, w_{\max}]$ (see Equation 3) using a step size 0.05, resulting in $\mathcal{A} = \{0.25, 0.30, 0.35, \ldots, 1.00\}$. This strategy allows a single pretrained model to support a continuum of inference-time widths with minimal training overhead.

Test-time training. Test-time width identification is applied only after pre-training. The parameter γ is optimized using a short calibration stream while keeping all pretrained model parameters fixed. The learned γ is then reused for all subsequent inference-time evaluations. We fix the calibration set to OpenWebText everywhere, and we limit the calibration dataset to only 50 batches, with a batch size of 6. We use the SGD optimizer with a learning rate of 0.01 for this training.

Compute. The GPT2-Small, GPT2-Standard, and GPT2-Medium experiments reported in this paper were run using NVIDIA A100-SXM4-40GB GPUs. The larger GPT-Large and DeepSeek-V2-Lite continual pre-training experiments were run using NVIDIA RTX PRO 6000 Blackwell GPUs. Training is performed using DDP spanning 4 GPUs.

Table 2. Model architectures, routing settings, and training configurations used in the from-scratch full pre-training experiments.

MODEL	ROUTING	PARAMS	LAYERS	HEADS	HIDDEN	TOKENS	BATCH	ITERS
GPT2-SMALL	E8A2	55M	6	6	384	3B	12×6×4	10K
	E8A2	55M	6	6	384	15B	12×6×4	50K
	E8A4	55M	6	6	384	4.5B	8×9×4	15K
	E16A4	83M	6	6	384	7.5B	8×9×4	25K
	E64A8	253M	6	6	384	3B	6×8×4	15K
	E128A8	480M	6	6	384	3B	6×8×4	15K
GPT2-STANDARD	E8A2	322M	12	12	768	3B	6×12×4	10K
	E8A2	322M	12	12	768	15B	6×12×4	50K
	E8A4	322M	12	12	768	9B	6×12×4	30K
GPT2-MEDIUM	E8A2	1B	24	16	1024	7.5B	2×12×4	75K
GPT2-LARGE	E8A2	2.5B	36	20	1280	50B	4×6×4	500K

C. Additional Analyses

C.1. What do γ values look like?

Figure 6 illustrates how the learned sharpness parameter γ adapts to the computing budget (in terms of MoE FLOPs) and model scale. Across all models, γ is optimized for the respective budget: low budgets favor larger γ values, corresponding to more concentrated width allocation, while higher budgets drive γ toward or below 1.0, yielding more uniform allocation across active experts.

When using a single shared γ, test-time training consistently finds a well-calibrated value for each budget that already delivers strong performance, aligning with the Pareto-optimal trends observed in earlier plots. The layer-wise variant is more expressive, assigning different sharpness levels to different transformer blocks, revealing non-uniform importance across layers, particularly at low budgets. This additional flexibility explains the improved compute-quality trade-offs observed with layer-wise width identification while maintaining stable behavior as the budget increases.

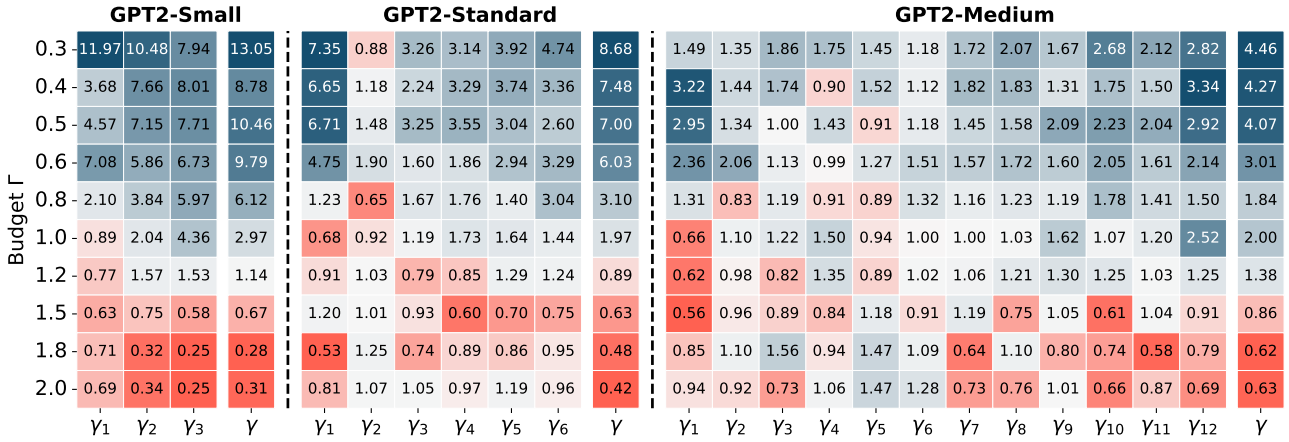

Figure 6. Learned γ parameters across budgets and model scales under E8A2 setting. Visualization of the learned γ parameters obtained via test-time training for MoSE under different routing budgets Γ. Each row corresponds to a budget and columns show either layer-wise γ_ℓ values (left of each pair) or a shared γ (right of each pair).

C.2. FLOPs Breakdown

Figure 7 illustrates the breakdown of inference-time floating-point operations (FLOPs) across model scales for GPT2-Small, GPT2-Standard, and GPT2-Medium. For each model, we decompose the total MFLOPs per token into contributions from MoE computation, attention, and other operations (linear layers, embeddings, etc.). The total MFLOPs per token are shown at the center of each donut chart.

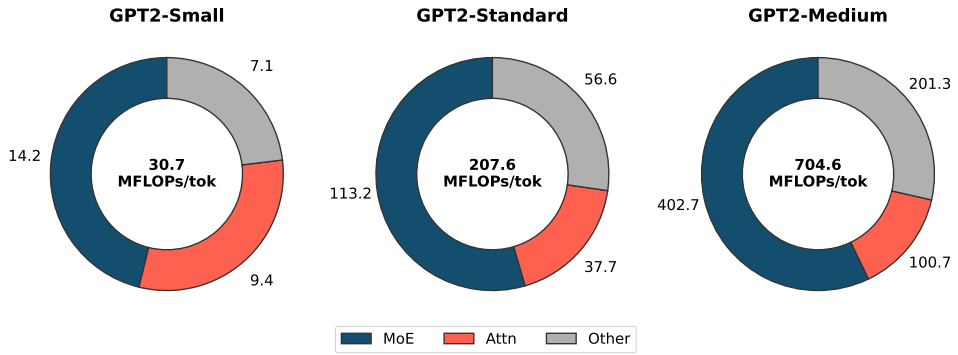

Figure 7. Inference-time FLOPs breakdown across model scales. Donut charts show the absolute MFLOPs per token attributed to MoE experts, attention, and other operations for GPT2-Small, GPT2-Standard, and GPT2-Medium. Total MFLOPs per token are shown at the center of each subplot.

Across all model sizes, MoE computation accounts for the largest fraction of the FLOPs, and this dominance becomes more pronounced as the model scale increases. While attention and other components grow with model size, their relative contribution remains substantially smaller than that of the MoE experts. For example, in GPT2-Small, expert computation already constitutes a significant portion of total compute, and by GPT2-Medium, MoE experts account for the majority of inference-time FLOPs.

This breakdown highlights why width adaptation is particularly effective for MoE models at scale. Since expert computation dominates the overall compute budget, reducing expert width directly targets the primary cost driver while leaving attention and routing mechanisms unchanged. As a result, MoSE is able to achieve substantial inference-time compute savings without modifying the underlying model architecture or routing behavior.

C.3. Width-wise Pre-training Dynamics

Figure 8 demonstrates the validation perplexity dynamics of MoSE across different execution widths during pre-training. Across all model scales and training budgets, larger widths consistently converge faster and achieve lower perplexity, while smaller widths follow the same overall training trajectory with higher steady-state perplexity. Importantly, all widths exhibit stable and monotonic convergence, indicating that slimmable training does not introduce optimization instability even at reduced widths.

The relative ordering of widths is preserved throughout training: wider configurations dominate narrower ones at all stages, and the gap between widths narrows as training progresses. This behavior is consistent across both shorter (3B-token) and longer (15B-token) training regimes, as well as across model scales.

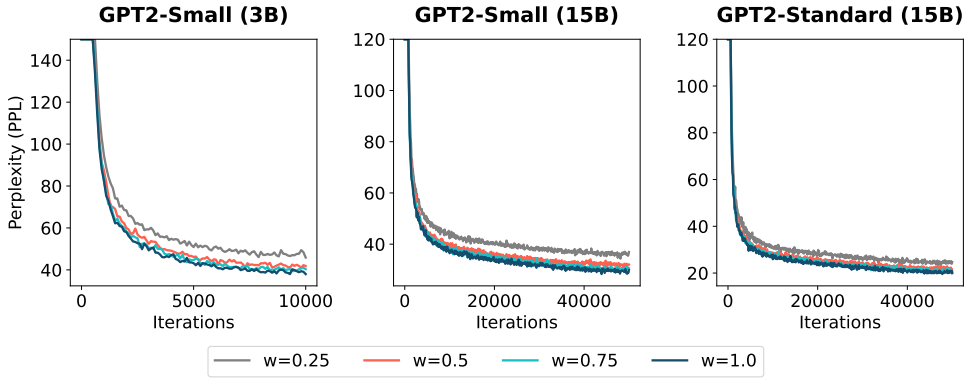

Figure 8. Validation perplexity dynamics across widths during pre-training. Validation perplexity as a function of training iterations for different execution widths $w \in \{0.25, 0.5, 0.75, 1.0\}$ during MoSE pre-training on OpenWebText. Results are shown for GPT2-Small trained for 3B (left) and 15B (middle) tokens, and GPT2-Standard trained with 15B tokens (right). For clarity, we clip early iteration values.

C.4. Runtime Diagnostics

The main paper uses FLOPs/token as the primary hardware-agnostic measure of inference cost. This section justifies that choice empirically. In our implementation, latency and throughput track FLOPs closely across the evaluated operating points, so FLOPs is an informative proxy for end-to-end runtime in the regimes considered here.

These measurements are also important for interpreting the practicality of width adaptation. The inference-time width policy itself is lightweight: the sharpness parameter γ is calibrated once, and the serving-time overhead only requires a simple transformation of the top-k router probabilities before projection to executable widths. The runtime plots therefore test whether the algorithmic savings predicted by FLOPs are reflected in actual latency and throughput measurements.

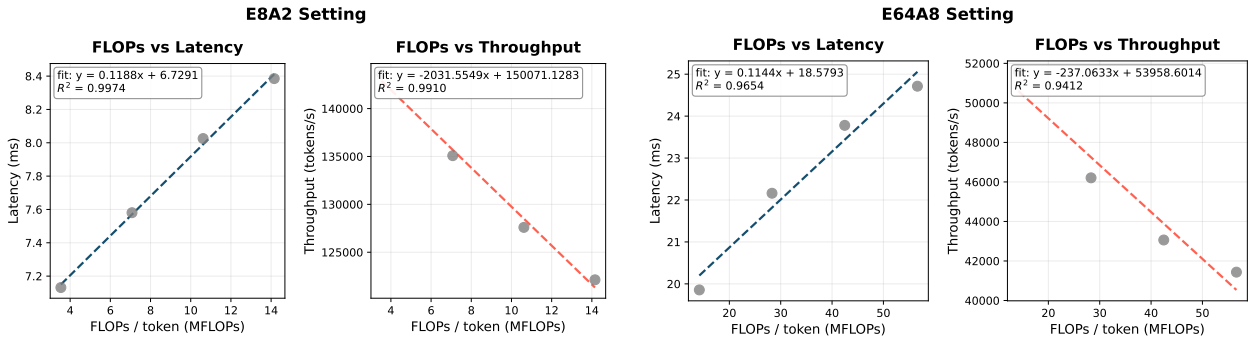

Figure 9. **Runtime analysis.** Latency and throughput track FLOPs closely in both E8A2 and E64A8 settings, supporting the use of FLOPs as the primary hardware-agnostic compute axis in the main paper.

We emphasize that these results justify the use of FLOPs for the current experimental setup; they do not claim that runtime must remain linear in FLOPs under all hardware or deployment regimes. In larger or more memory-bound systems, the realized wall-clock gains may differ, which is why we continue to present FLOPs as the primary architecture-level metric in the main body.

C.5. Routing Diagnostics

A natural concern with the MoSE training objective is whether exposing the same model to both full-width and sampled-width executions could distort router behavior, introduce expert imbalance, or destabilize optimization. Our expectation is that this should not happen because widths are *not* learned by the router during training: the router still learns standard expert selection under the usual MoE stabilization terms, while width allocation itself is only introduced at inference/calibration time.

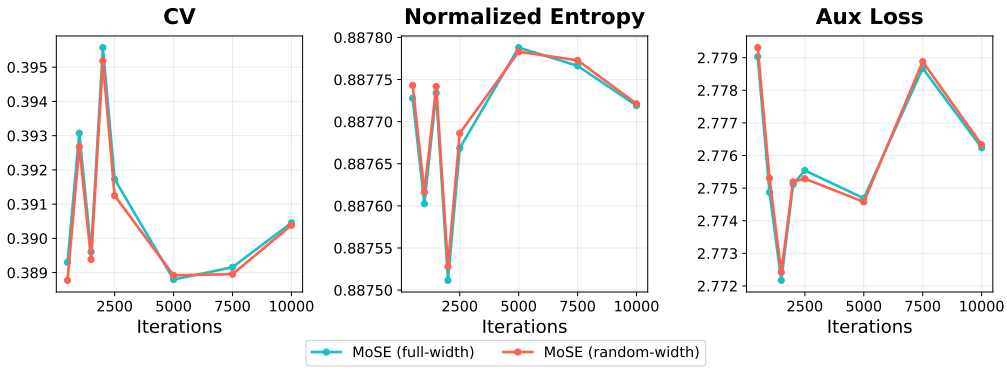

Figure 10. **Routing-stability diagnostics** for the full-width and sampled-width branches during training. We report coefficient of variation (CV) of expert loads, normalized routing entropy, and auxiliary/load-balancing loss. The close match between the two branches indicates that the multi-width objective does not induce routing instability.

The diagnostics in Figure 10 test this claim directly. We compare the full-width and sampled-width branches using three complementary statistics:

(i) the coefficient of variation (CV) of expert loads, which measures imbalance across experts;
(ii) normalized routing entropy, which measures whether routing collapses toward a small subset of experts; and
(iii) the auxiliary/load-balancing loss, which directly tracks the regularization used to avoid expert collapse.

Similar trajectories across the two branches indicate that the multi-width objective does not induce a new routing pathology.

We additionally visualize expert-utilization heatmaps (Figure 11) and compare global expert-selection histograms across widths (Figure 12). These analyses are meant to address a stronger question than simple training stability: not only does MoSE train stably, but the sampled-width branch preserves nearly the same routing structure as the full-width branch, supporting the interpretation that Equation 6 adds width robustness without perturbing the learned routing policy.

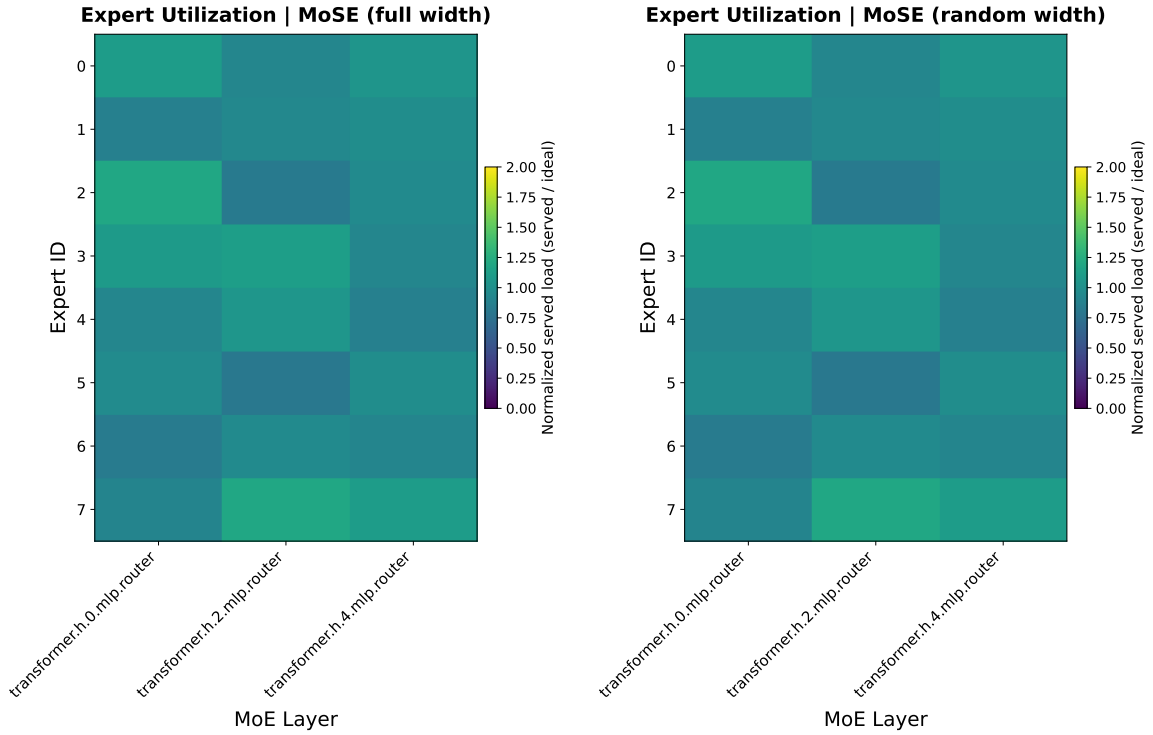

Figure 11. Expert-utilization heatmaps for the full-width and sampled-width branches. Expert usage is nearly identical across branches, with no dead experts or severe overload, indicating that width sampling does not alter the qualitative routing pattern learned by the model.

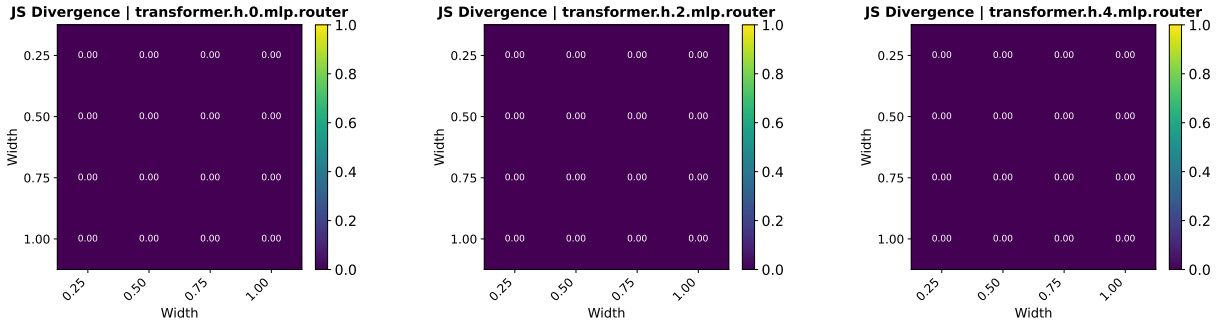

Figure 12. JS divergence between global expert-selection histograms across width pairs. We aggregate routed expert assignments over a fixed validation shard and compare the resulting histograms across widths. Near-zero divergence indicates that global routing distributions remain unchanged across widths.

D. Additional Results

D.1. Scaling Pre-Training Tokens at Fixed Routing

Figure 13 studies the effect of increasing the amount of pre-training data while keeping the routing configuration fixed (E8A2). We increase the number of pre-training tokens from 3B to 15B for GPT2-Small and GPT2-Standard and evaluate MoSE using the same inference-time width allocation procedure. As shown, increasing the amount of pre-training data consistently shifts the Pareto frontier downward, improving perplexity at all compute budgets. Importantly, the advantage of test-time training over alternative inference modes remains stable under increased data scale, suggesting that the learned width identification captures structural properties of the routing-compute trade-off rather than overfitting to a particular data regime.

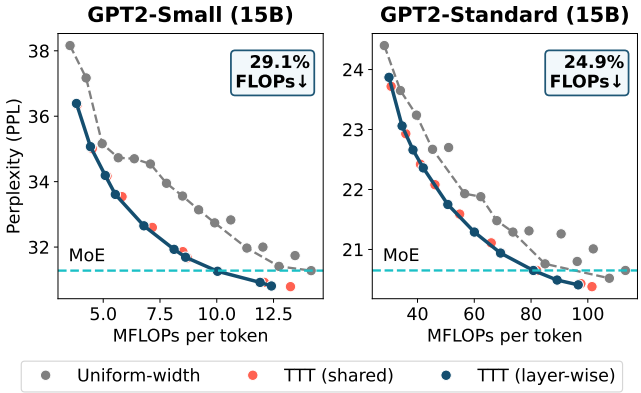

Figure 13. Scaling pre-training tokens at E8A2 setting. We scale the number of pre-training tokens from 3B (Figure 4) to 15B for GPT2-Small and GPT2-Standard, while keeping the same routing setup and compute budget. Our test-time training for MoSE width identification continues to dominate across the Pareto frontier, and increasing the amount of pre-training data consistently shifts the quality-compute trade-off downward. The relative advantage of test-time training (with both layer-wise and shared parameters) remains stable under increased data scale.

D.2. Effect of Routing Setting and Expert Count

Figure 14 analyzes the effect of the routing setting on MoSE using GPT2-Small, comparing E8A2, E8A4, and E16A4 configurations. Increasing both the width budget and the number of activated experts consistently improves the compute-quality trade-off, shifting the Pareto frontier downward. Across all settings, MoSE with test-time width identification maintains strong performance, indicating that the method adapts naturally to different routing granularities. This result highlights that width identification complements routing capacity: larger budgets provide more flexibility, which MoSE effectively exploits at inference time.

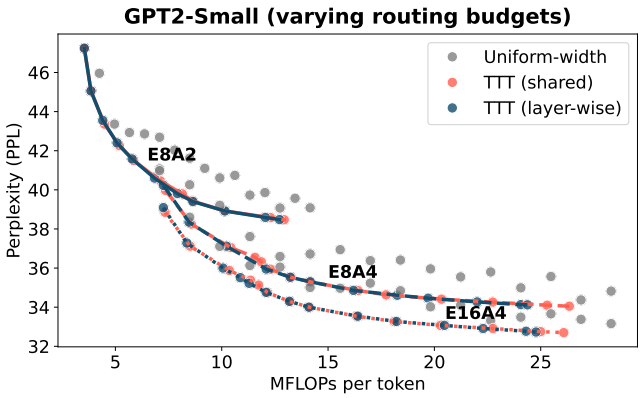

Figure 14. Effect of routing setting on MoSE with test-time width identification (GPT2-Small). We evaluate MoSE under different routing settings (E8A2, E8A4, E16A4) using GPT2-Small model. Points denote individual configurations, while solid curves trace the Pareto frontiers of respective modes. Increasing the budget and the expert count consistently shifts the Pareto frontier downward, achieving lower perplexity at comparable FLOP consumption.

D.3. Increased Expert Count and Pre-Training Tokens

The main-body inference results already show that MoSE improves the compute-quality frontier across model scales and routing settings. This section isolates a complementary regime in which both the active expert count and the pre-training token budget are increased.

Specifically, Figure 15 focuses on E8A4 setting and compares the GPT2-Small and GPT2-Standard models trained with an increased amount of pretraining data (4.5B and 9B pretraining tokens, respectively). Under this setting, MoSE with test-time training continues to outperform across both model sizes. This confirms our findings from Figures 4, 13, and 14 and supports the robustness of MoSE across routing settings, model sizes, and data scales.

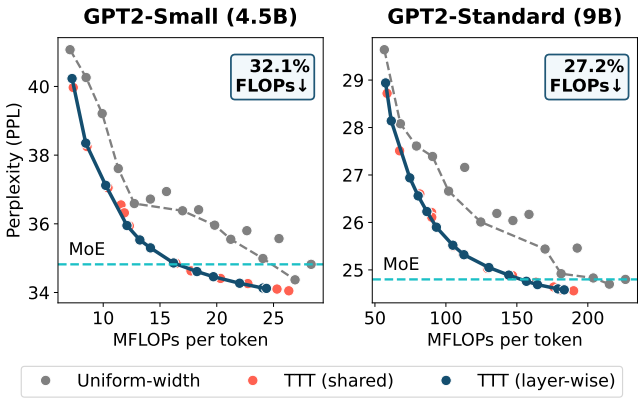

Figure 15. MoSE under E8A4 setting and increased pre-training tokens. We compare inference-time execution modes for the GPT2-Small (4.5B tokens) and GPT2-Standard (9B tokens) models. MoSE (TTT) maintains a clear advantage over the uniform-width mode across both model sizes.

D.4. Inference-Time Routing Transfer

Figure 16 examines the robustness of MoSE to inference-time routing changes when using a fixed pretrained checkpoint. We train a single MoSE model under the E16A4 routing configuration and evaluate it at inference time with fewer activated experts (E16A2, E16A3, and E16A4) without any retraining. The first three panels report Pareto frontiers for each evaluation setting independently, while the rightmost panel overlays the resulting frontiers to enable direct comparison.

As shown in Figure 16, MoSE with test-time training degrades gracefully as the number of active experts is reduced, maintaining strong perplexity-compute trade-offs even under more restrictive routing. Notably, E16A3 achieves similar performance to E16A4 at lower compute, indicating that activating more experts can be redundant once a moderate routing capacity is reached. This suggests diminishing returns from increasing the number of activated experts beyond this point for

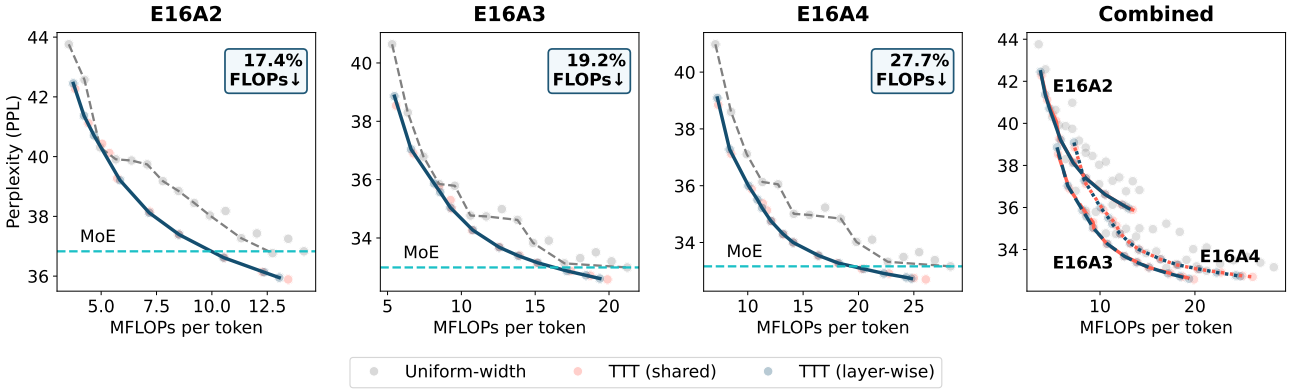

Figure 16. Inference-time routing transfer with a fixed checkpoint (GPT2-Small under E16A4 setting). This experiment evaluates a single MoSE checkpoint trained with a setting **E16A4** under different numbers of activated experts. The first three panels report results for each evaluation setting independently, while the fourth plot overlays Pareto frontiers of MoSE (TTT). Notably, E16A3 achieves essentially the same performance as E16A4 at reduced compute, indicating that E16A3 provides a more compute-efficient operating point for evaluation.

this experiment.

To identify the most suitable k, evaluating routing settings under full-width execution is sufficient, as the resulting Pareto frontier closely follows the best configuration.

D.5. Ablation Study of MoSE Inference modes

Figure 17 isolates the effect of our proposed inference-time width assignment strategy in MoSE by comparing uniform-width identification, router-conditioned widths based on normalized probabilities (constant $\gamma = 1.0$), and test-time training (TTT) for width identification. Across model scales and expert settings, the normalized-probability baseline consistently improves over uniform widths, confirming that router confidence contains useful information for allocating compute. However, its Pareto frontier remains uniformly dominated by the learned TTT variants. Learning the sharpness parameter γ at test time, either as a single shared value or in a layer-wise form, yields systematically better perplexity-compute trade-offs, demonstrating that effective width allocation requires adapting the probability-to-width mapping (given the budget) rather than relying on fixed normalization alone. The gap is most pronounced at lower compute budgets, where the accurate concentration of width on a subset of experts is critical, while shared γ already captures much of the gain, and the layer-wise variant provides additional flexibility and is more expressive.

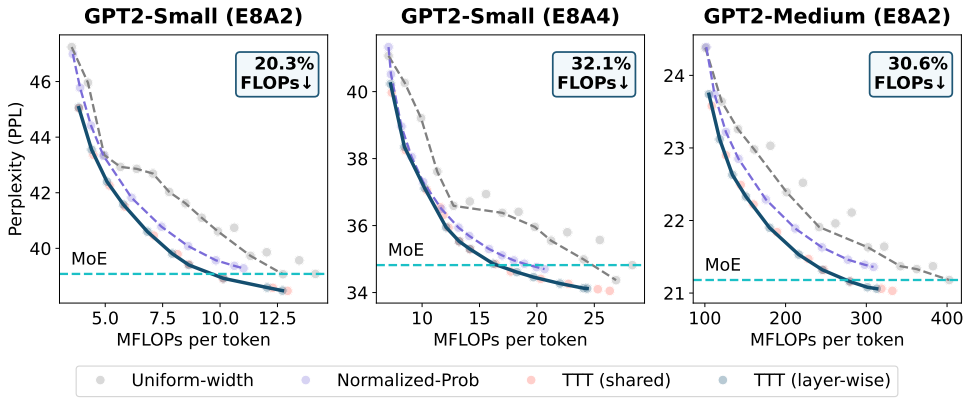

Figure 17. Ablation of MoSE inference modes. This figure compares the three inference-time execution modes supported by MoSE. Results are shown across different expert settings and model scales. Points denote individual configurations, while curves trace Pareto frontiers. The normalized-probability mode improves over uniform widths but remains consistently inferior to the learned TTT variant, indicating that performance gains are not solely due to probability-based allocation but require learning the probability-width mapping.

D.6. Transferability of γ

Figure 18 reports zero-shot transfer performance on LAMBADA, using sharpness parameters γ calibrated on OpenWebText and reused without modification. We also report our results in terms of accuracy and perplexity. As shown in the left panel, both learned variants consistently outperform the uniform-width MoSE across compute budgets, with MoSE (layerwise

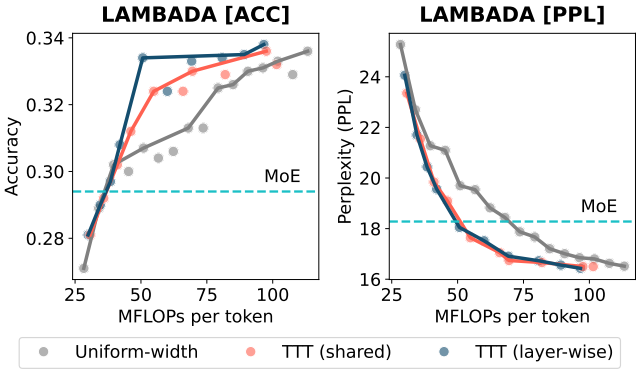

Figure 18. Transferability of the learned sharpness parameter γ on LAMBADA (GPT2-Standard, 15B tokens).

γ) achieving the highest accuracy at moderate MFLOPs. The accuracy curves exhibit mild non-monotonicity and local fluctuations, which are expected given that accuracy is a discrete metric and is sensitive to sampling noise and prediction discretization (Schaeffer et al., 2023).

In contrast, the right panel (perplexity) provides a smoother and more stable signal. Both test-time training modes yield consistently lower perplexity than the non-adapted variant across the full compute range, with layerwise γ forming the dominant Pareto frontier. The steady downward trend in perplexity confirms that the transferred γ values preserve their intended effect on width allocation, even under dataset shift. Together, these results indicate that the learned sharpness parameter generalizes effectively across datasets, improving zero-shot performance without test-time training, while perplexity serves as the more reliable indicator of this transfer behavior.

D.7. Large-Scale Experiments

Figures 19 and 20 extend the main inference study to larger and finer-grained settings. On GPT2-Large (2.5B parameters, 50B tokens), MoSE continues to improve the inference-time Pareto frontier and achieves 32.8% lower FLOPs at matched perplexity. In finer-grained routing regimes, MoSE remains effective and yields 32.0% and 40.8% FLOPs reductions for E64A8 and E128A8, respectively.

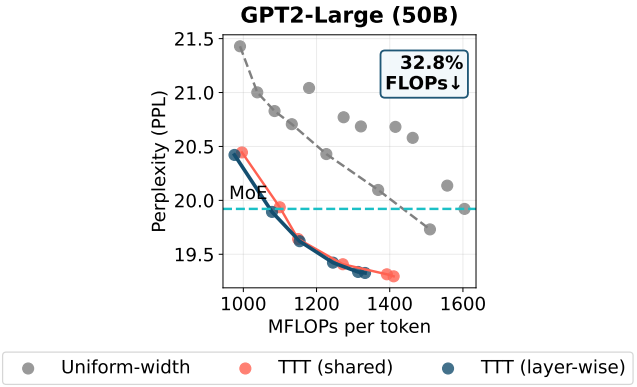

Figure 19. Pareto frontier on GPT2-Large (2.5B parameters, 50B tokens), where MoSE achieves 32.8% lower FLOPs at matched perplexity.

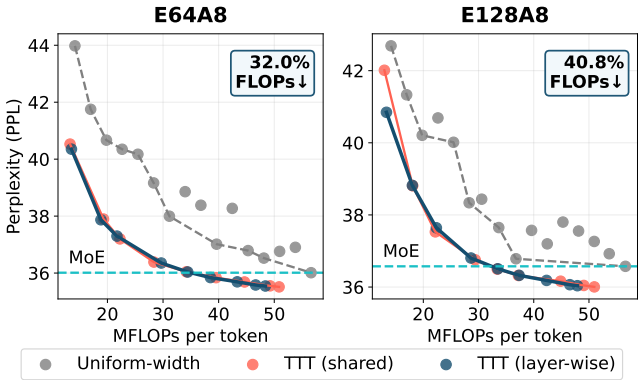

Figure 20. Pareto frontiers in finer-grained routing regimes. MoSE achieves 32.0% and 40.8% FLOPs reductions for E64A8 and E128A8, respectively, at comparable perplexity.

D.8. Further Downstream Task Results

Table 3 reports zero-shot downstream reasoning results on HellaSwag, PIQA, and SIQA. Across all three benchmarks, MoSE improves accuracy while reducing MFLOPs/token, indicating that the gains of width-adaptive execution are not limited to perplexity-based evaluation but transfer to downstream reasoning tasks as well. Additional runtime measurements and routing-stability diagnostics are provided in Appendix C.4 and Appendix C.5.

Table 3. Zero-shot downstream reasoning results on the GPT2-Medium setting. MoSE improves accuracy while reducing inference compute on HellaSwag, PIQA, and SIQA.

TASK	METHOD	ACCURACY	MFLOPS/TOKEN
HELLASWAG	MOE	0.335	402.7
HELLASWAG	MOSE (TTT)	**0.385**	**272.5**
PIQA	MOE	0.675	402.7
PIQA	MOSE (TTT)	**0.710**	**319.1**
SIQA	MOE	0.290	402.7
SIQA	MOSE (TTT)	**0.305**	**319.1**

E. Continual Pre-Training Results

This section provides additional results for continual pre-training (CPT) slimmability adaptation, where we start from a pretrained MoE checkpoint and continue training under the MoSE objective. These experiments complement the main-body DeepSeek result in Figure 5 and isolate the practical regime where retraining from scratch is too expensive, but a pretrained MoE checkpoint already exists.

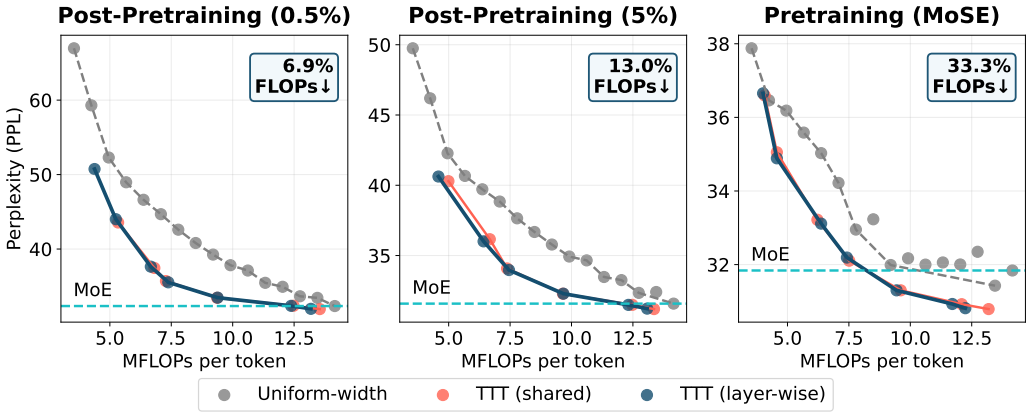

Figure 21. Post-pretraining slimmability adaptation compared with full MoSE pretraining. Left: 0.5% continued pretraining. Middle: 5% continued pretraining. Right: full MoSE pretraining. Even short CPT phases recover meaningful inference-time savings, while full pretraining yields the strongest compute-quality frontier.

We view full multi-width pre-training and CPT as complementary regimes rather than substitutes. Full pre-training yields the strongest compute-quality frontier because width robustness is learned from the start. CPT, by contrast, asks how much of that benefit can be recovered post hoc with a short continued-training phase, and whether slim-width subnetworks can be enabled without degrading the original full-width checkpoint.

Figure 21 addresses the first question. Even a 0.5% continued pre-training budget already recovers measurable inference-time savings, while 5% continued pre-training yields a 13.0% FLOPs reduction and substantially narrows the gap to full MoSE pre-training. Figure 22 addresses the second question: the full-width validation curve remains stable throughout adaptation, while slimmed-width subnetworks improve rapidly and become usable early in training. Together, these results show that slimmability can be induced post hoc under the same training recipe, providing a lightweight deployment path when full pretraining is impractical.

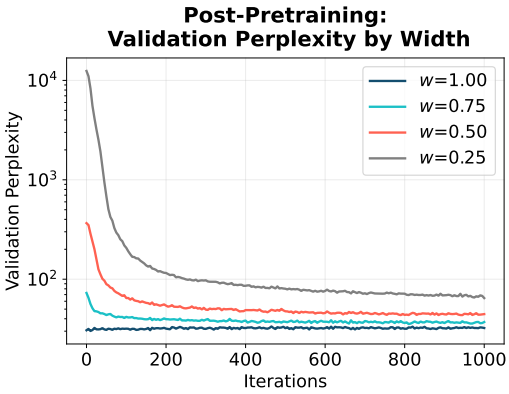

Figure 22. Validation perplexity across widths during continual pre-training adaptation. The full-width model remains stable, while reduced-width subnetworks improve rapidly and become usable early in adaptation.