ICML
International Conference
On Machine Learning

ICML 2026 Academic Presentation

cMoLLM at Scale:

Horizontal Scaling Laws for Convolutionally-Gated Mixture-of-LLMs

Xin Yang · Yemin Wang · Mingda Liu · Letian Li · Shuaishuai Cao · Zhengxiao He · Ryan Dong

Seoul, South Korea · 2026

Dynamic convolution view of MoE

Pipeline-level scaling

Stable soft routing

Talk Outline

ICML International Conference On Machine Learning

01

1. Motivation

Dense scaling and limits of prior MoE-style methods

02

2. Key Insight

MoE layers as dynamic 1×1 convolutions

03

3. cMoLLM Method

Soft kernel mixing over end-to-end streams

04

4. Results & Scaling

Ablations, scaling laws, and future directions

Motivation: Capacity Is Still Tied to Compute

Dense Transformers activate every parameter for every token, making scaling increasingly expensive.

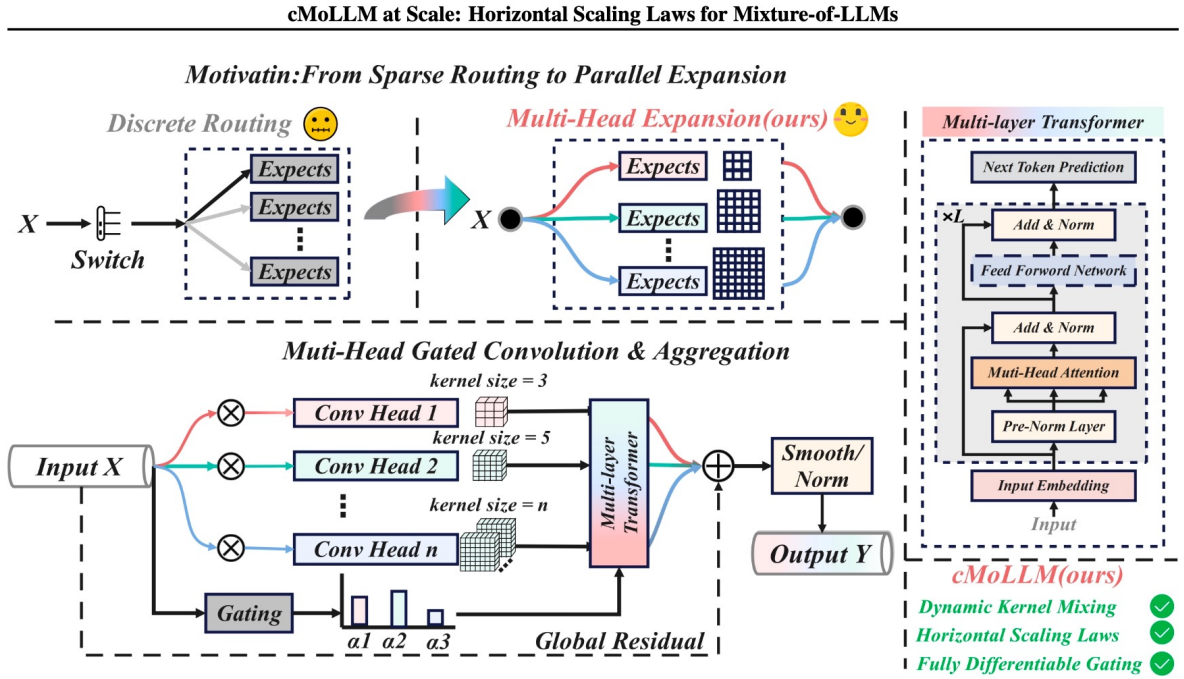


Figure 1. **Teaser of cMoLLM.** cMoLLM scales capacity at the pipeline level via convolutionally-gated mixture over end-to-end streams, yielding better perplexity and downstream accuracy than dense baselines under matched compute, without virtual tokens or auxiliary heads.

Core Problem

- Dense LLMs couple capacity and activated computation.
- Classical MoE usually improves capacity only inside FFN blocks.
- Sparse Top-K routing can introduce expert collapse and brittle utilization.
- The paper asks: can we scale capacity horizontally without unstable routing?

Desired direction: pipeline-level mixture with stable, hardware-friendly routing.

Why Prior Pipeline-Level Scaling Is Not Enough

ParaScale and AltUp are useful baselines, but each brings extra structural assumptions or optimization risks.

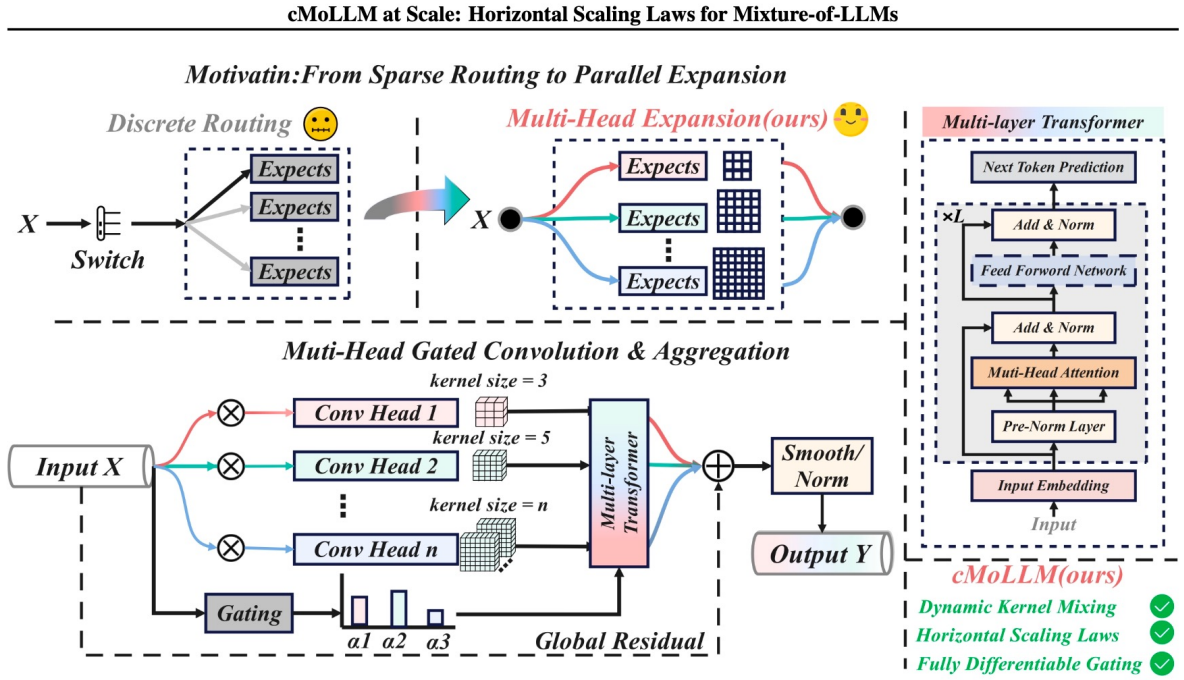


Figure 1. **Teaser of cMoLLM.** cMoLLM scales capacity at the pipeline level via convolutionally-gated mixture over end-to-end streams, yielding better perplexity and downstream accuracy than dense baselines under matched compute, without virtual tokens or auxiliary heads.

Observed Gap

The missing design point

- No virtual tokens: avoid sequence-length overhead.
- No auxiliary branch: keep the pipeline simple and adaptive.
- No sparse Top-K: reduce routing discontinuity and collapse risk.
- Keep computation close to dense Transformer cost for small N.

cMoLLM targets this gap with dynamic convolutional routing.

Key Insight: MoE as Dynamic 1×1 Convolution

Experts become kernels; routing becomes input-conditioned kernel aggregation.

Theorem 4.1

For linear experts:

$$\mathbf{MoE(x) = \Sigma_k\, g_k(x)\, E_k(x)}$$

If $E_k(x) = x\, W_k^\top$, then

$$\mathbf{MoE(x) = Conv_{1\times 1}(x;\, \tilde{K}(x))}$$

$$\mathbf{\tilde{K}(x) = \Sigma_k\, g_k(x)\, W_k}$$

Interpretation: dynamic convolution provides a continuous, differentiable lens for conditional computation.

$$\mathrm{MoE}(\mathbf{x}) = \sum_{k=1}^{N} g_k(\mathbf{x})\, E_k(\mathbf{x}), \tag{2}$$

where $g(\mathbf{x}) \in \mathbb{R}^N$ denotes the routing weights produced by the gate. In practice, only the top-K entries of $g(\mathbf{x})$ are nonzero, yielding sparse computation. Each expert E_k is typically a two-layer FFN with its own parameters.

3.4. Pointwise Convolution as a Linear Transform

A pointwise (1×1) convolution applies the same linear map across token positions. Given an input sequence $\mathbf{X} \in \mathbb{R}^{L\times d_{\text{in}}}$ and a kernel $\mathbf{K} \in \mathbb{R}^{d_{\text{out}}\times d_{\text{in}}}$, the operation is given by

$$\mathrm{Conv}_{1\times1}(\mathbf{X};\mathbf{K}) = \mathbf{X}\mathbf{K}^\top, \tag{3}$$

which corresponds to applying an identical token-wise linear projection from $\mathbb{R}^{d_{\text{in}}}$ to $\mathbb{R}^{d_{\text{out}}}$. This observation serves as a key building block for our subsequent analysis.

4. Method: cMoLLM

We now present our main theoretical result and the cMoLLM architecture.

4.1. MoE as Dynamic Convolution: A Formal Equivalence

$$\sum_{k=1} g_k(\mathbf{x}) \cdot (\mathbf{x}\mathbf{W}_k^{\cdot}) = \mathbf{x}\Big(\textstyle\sum_{k=1}^{N} g_k(\mathbf{x})\mathbf{W}_k\Big) = \mathrm{Conv}_{1\times1}\Big(\mathbf{x};\, \textstyle\sum_{k=1}^{N} g_k(\mathbf{x})\mathbf{W}_k\Big). \tag{6}$$

The weighted sum of expert weight matrices forms a dynamic kernel that varies with $\mathbf{x}$. (Classical MoE uses Top-K routing so only K terms are nonzero; we use soft mixing over all streams.) See Section A for extension to nonlinear experts.

Corollary 4.2 (Nonlinear Two-Layer Experts)**.** *Under Theorem 3.2, consider two-layer experts of the form* $E_k(\mathbf{x}) = \sigma(\mathbf{x}\mathbf{W}_{k,1}^\top)\mathbf{W}_{k,2}^\top$ *with routing weights* $\{g_k(\mathbf{x})\}_{k=1}^N$ *satisfying* $\sum_k g_k(\mathbf{x}) = 1$*. Then the MoE layer can still be written as a dynamic* 1×1 *convolution with an input-dependent effective kernel that incorporates a data-dependent mask induced by* σ*; see Section A for details.*

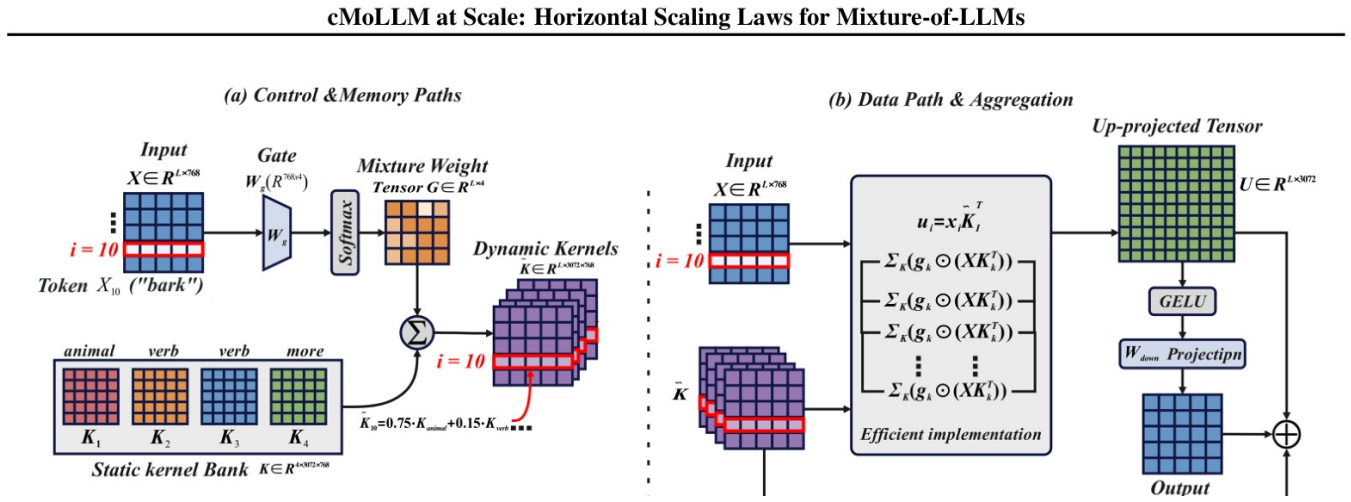
Assumptions 3.1 and 3.2 are not needed for Theorem 4.1 itself, but they will be used in our later complexity, stability, and toy-model analyses that build on this equivalence.

Remark 4.3 (Key Differences from Standard Convolution). While Theorem 4.1 reveals structural similarity, standard convolutional layers differ from explicit MoE in three aspects:

1. **Static vs. Learnable Gating.** In CNN, activation func-

cMoLLM Architecture

A convolutionally-gated mixture over end-to-end LLM streams.



Design Recipe

- Keep the Transformer backbone.
- Replace FFN stack with stream-wise kernels.
- Use a lightweight gate to produce soft mixture weights.
- Apply dynamic 1×1 convolution through grouped / pointwise primitives.

No virtual tokens | **No auxiliary heads**

No Top-K | **No low-rank**

Forward Pass and Training Strategy

A single cMoLLM block uses soft weights to mix per-stream kernels before GELU and down-projection.

Algorithm 1 Forward pass of a Transformer layer with cMoLLM

Input: $\mathbf{X} \in \mathbb{R}^{L \times d}$, number of streams N, kernels $\{\mathbf{K}_k\}_{k=1}^{N}$, $\mathbf{W}_{\text{down}}$
Self-Attention: $\mathbf{H} \leftarrow \text{SelfAttention}(\mathbf{X})$
Gating logits: $\mathbf{G} \leftarrow \mathbf{H}\mathbf{W}_g + \mathbf{b}_g \in \mathbb{R}^{L \times N}$
Soft mixture weights: $g_{i,k} \leftarrow \text{softmax}(\mathbf{G}_{i,:})_k$ for each token i, stream k
Mixed kernel: $\widetilde{\mathbf{K}}_i \leftarrow \sum_{k=1}^{N} g_{i,k}\,\mathbf{K}_k$
Dynamic 1×1 conv: $\mathbf{U}_i \leftarrow \mathbf{H}_i\,\widetilde{\mathbf{K}}_i^{\top}$
Nonlinearity & down-proj: $\mathbf{Y}_i \leftarrow \mathbf{W}_{\text{down}}\,\sigma(\mathbf{U}_i)$
Output: $\mathbf{Y}$ (residual + norm as usual)

Forward Pass

- Self-attention produces hidden states H.
- A softmax gate estimates token-wise stream weights $g_{i,k}$.
- The mixed kernel $\tilde{K}_i = \Sigma_k\, g_{i,k}\, K_k$ is applied as dynamic 1×1 convolution.

Load Balancing

$$L = LLM + Lbal \qquad Lbal = \alpha \cdot N \cdot \Sigma_k\, p_k^{\,2}$$

- Discourages collapse onto a few streams.
- Keeps routing fully differentiable.
- No kernel regularization, progressive sparsification, or low-rank terms.

Gating Variants

The paper evaluates four convolutional gates; multi-head gives the strongest overall trade-off.

Simple

Local token-wise gate: feature conv × sigmoid gate conv.

Evaluated

Context-aware

Local gate plus global sequence context from pooled statistics.

Evaluated

Multi-head

Independent gate heads specialize across channel groups and temporal patterns.

Best overall

Adaptive

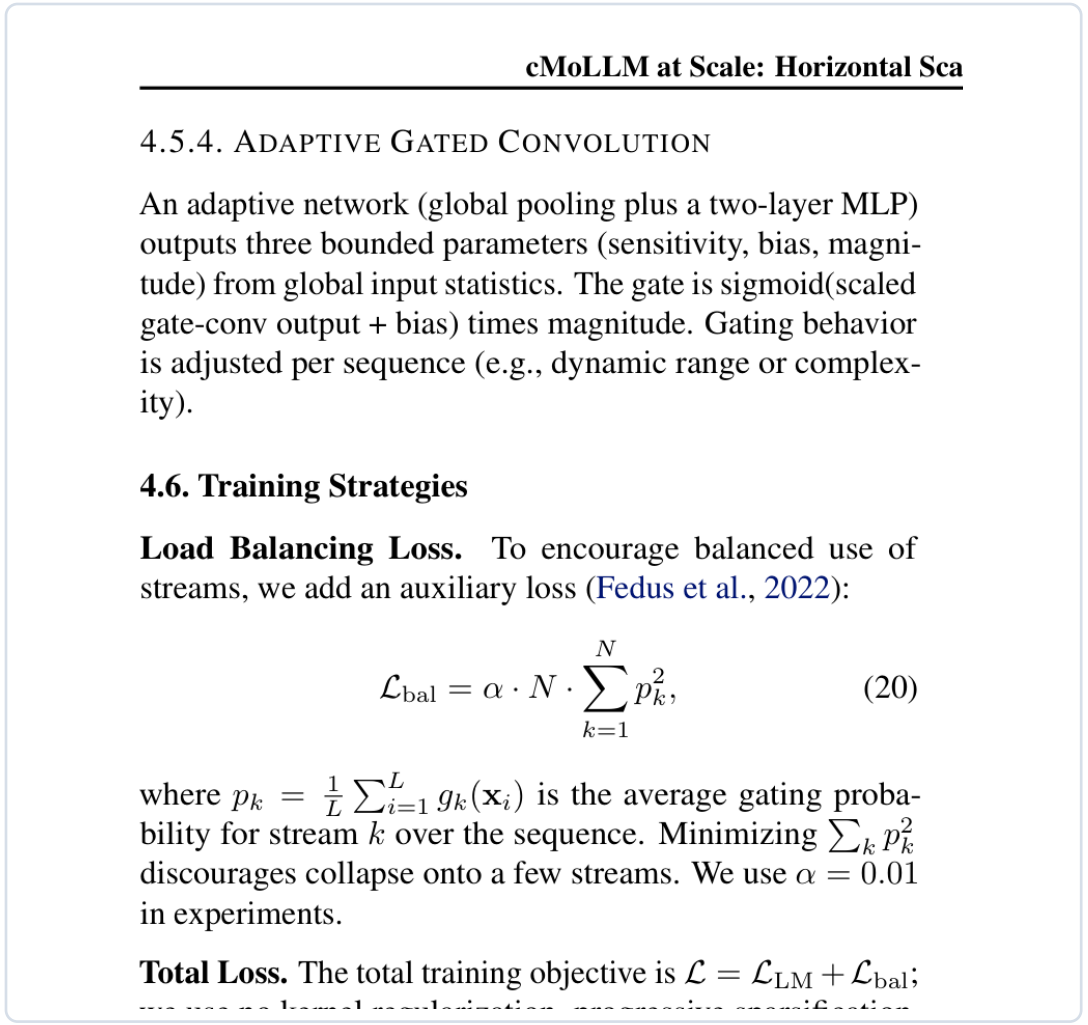
Per-sequence sensitivity, bias, and magnitude adapt gate behavior.

Evaluated

Result pattern: increasing stream count generally improves perplexity; multi-head remains stable at N = 8.

Complexity: Horizontal Scaling Law

For bounded N, capacity grows through streams while per-token compute remains a constant-factor expansion.



cMoLLM at Scale: Horizontal Sca

4.5.4. Adaptive Gated Convolution

An adaptive network (global pooling plus a two-layer MLP) outputs three bounded parameters (sensitivity, bias, magnitude) from global input statistics. The gate is sigmoid(scaled gate-conv output + bias) times magnitude. Gating behavior is adjusted per sequence (e.g., dynamic range or complexity).

4.6. Training Strategies

Load Balancing Loss. To encourage balanced use of streams, we add an auxiliary loss (Fedus et al., 2022):

$$\mathcal{L}_{\text{bal}} = \alpha \cdot N \cdot \sum_{k=1}^{N} p_k^2, \qquad (20)$$

where $p_k = \frac{1}{L} \sum_{i=1}^{L} g_k(\mathbf{x}_i)$ is the average gating probability for stream k over the sequence. Minimizing $\sum_k p_k^2$ discourages collapse onto a few streams. We use $\alpha = 0.01$ in experiments.

Total Loss. The total training objective is $\mathcal{L} = \mathcal{L}_{\text{LM}} + \mathcal{L}_{\text{bal}}$;

FLOPs_cMoLLM ≈ (N + 1) d d_ff + FLOPs_gate

Horizontal scaling: effective capacity grows with the number of streams while per-token cost stays close to the dense baseline for small N.

Experimental Setup

GPT-2-style language modeling and downstream evaluation under matched training settings.

Base architecture	GPT-2-style Transformer
Training data	FineWeb large-scale curated web corpus
Model scales	85M, 350M, 760M parameters
Sequence length	4096
FFN dimension	3072
Learning rate	6 × 10^-5
Metrics	validation loss, PPL, GLUE, SQuAD v2
Reporting	3 random seeds; mean ± std

Ablation: gating variants × stream counts N ∈ {1, 2, 4, 8}

Scaling: 85M → 350M → 760M

Main Ablation Results: 85M Scale

Multi-head gating with N = 8 achieves the best loss, PPL, and SQuAD v2 in Table 4.

cMoLLM at Scale: Horizontal Scaling Laws for Mixture-of-LLMs

Table 4. Full experimental results (3 seeds; mean ± std): validation loss, perplexity (PPL), GLUE (%), SQuAD v2 (%). "conv" = gating variant; n = streams. **Bold** = best (SOTA) in column.

conv	n	Loss ↓	PPL ↓	GLUE (%) ↑	SQuAD v2 (%) ↑
base	–	3.23±0.03	25.40±0.65	42.45±0.99	50.07±0.65
simple	1	3.04±0.03	20.94±0.22	44.16±0.56	51.13±0.54
simple	2	3.01±0.02	20.39±0.56	44.62±1.02	52.08±0.95
simple	4	2.97±0.04	19.53±0.49	44.92±0.87	55.71±1.30
simple	8	3.00±0.03	20.24±0.12	44.68±0.33	55.95±0.51
context_aware	1	3.04±0.03	20.94±0.41	44.16±0.42	51.12±1.28
context_aware	2	2.98±0.03	19.83±0.55	46.79±1.21	52.13±1.13
context_aware	4	2.98±0.02	19.78±0.12	47.12±0.25	55.64±0.87
context_aware	8	2.96±0.01	19.31±0.68	**48.98±0.64**	56.11±0.73
multi_head	1	3.04±0.01	20.93±0.22	44.16±1.18	51.18±0.66
multi_head	2	2.99±0.05	19.90±0.28	47.78±0.45	52.98±1.35
multi_head	4	2.96±0.04	19.44±0.41	48.02±0.79	56.37±1.56
multi_head	8	**2.93±0.02**	**18.82±0.37**	48.82±1.01	**57.06±0.52**
adaptive	1	3.04±0.05	20.94±0.46	44.16±0.98	51.28±0.88
adaptive	2	3.00±0.02	20.14±0.26	46.64±0.43	52.43±1.45
adaptive	4	2.97±0.03	19.68±0.71	47.03±0.36	55.89±1.39
adaptive	8	2.97±0.03	19.58±0.65	47.81±0.75	56.74±0.72

Best Row

2.93
Loss ↓

18.82
PPL ↓

48.82
GLUE ↑

57.06
SQuAD v2 ↑

Best cMoLLM variant: multi-head, N = 8

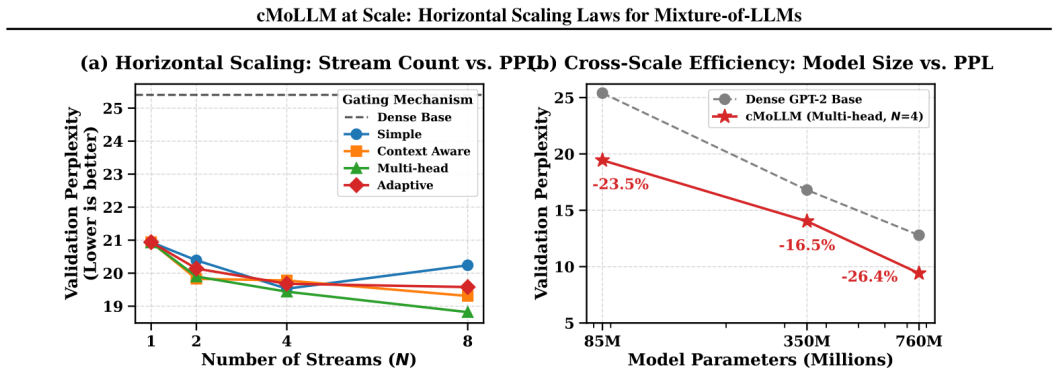
Key message: soft stream routing gives consistent gains over the dense base under the same evaluation protocol.

Cross-Scale Results

The advantage of multi-head cMoLLM is preserved from 85M to 760M parameters.

Table 5. Scaling experiments (3 seeds; mean ± std) across model sizes (Table 2). `mh` = multi-head cMoLLM; `base` = dense baseline. **Bold** = SOTA in column.

type	Loss ↓	PPL ↓	GLUE ↑	SQuAD v2 ↑
base-85M	3.23±0.03	25.40±0.65	42.45±0.99	50.07±0.65
base-350M	2.82±0.04	16.80±0.47	50.68±1.08	59.24±1.24
base-760M	2.55±0.05	12.79±0.28	55.74±0.46	62.91±0.74
mh-85M	2.96±0.03	19.44±0.56	48.02±0.84	56.37±0.54
mh-350M	2.64±0.03	14.03±0.37	53.82±1.03	61.08±1.26
mh-760M	**2.24±0.02**	**9.41±0.45**	**58.45±0.77**	**64.56±1.32**



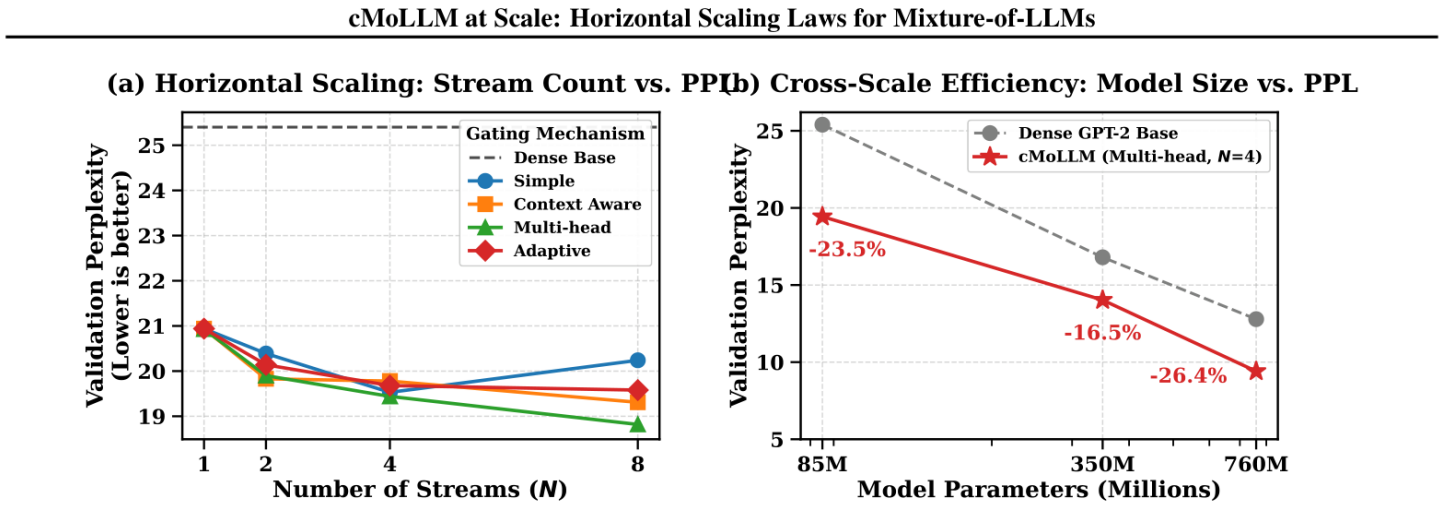
Cross-scale takeaway

- At every reported scale, cMoLLM improves loss and perplexity over the dense baseline.
- The 760M cMoLLM model reaches PPL 9.41 versus 12.79 for the dense base.
- Downstream GLUE and SQuAD v2 accuracy also improve at all three model scales.

Scaling claim: cMoLLM scales favorably with both stream count and parameter count.

Scaling Behavior: Stream Count and Model Size

Horizontal scaling improves perplexity as N increases, while cross-scale gains remain stable.



Interpretation

- N = 1 → 4/8 reduces validation PPL across most gating variants.
- Multi-head remains stable at high stream count.
- The cMoLLM curve stays below the dense baseline at all tested model sizes.
- Empirical evidence matches the proposed horizontal scaling law.

Summary: What cMoLLM Buys

A compact view of the paper's theoretical, architectural, and empirical contributions.

Theoretical Lens

MoE-style routing is equivalent to input-conditioned dynamic 1×1 convolution.

Architectural Design

End-to-end streams are softly mixed via convolutional kernels rather than sparse Top-K selection.

Empirical Behavior

Better PPL, GLUE, and SQuAD with stable utilization and favorable scaling.

cMoLLM: more effective capacity under matched compute, with fully differentiable routing and no virtual tokens / auxiliary heads / Top-K.

Limitations, Future Work, and Q&A

The method is promising, but larger-scale validation remains the next key step.

Limitations

- Current experiments are at GPT-2 scale up to approximately 760M parameters.
- 7B+ validation and distributed training need to be tested.
- The equivalence is explored mainly for mixture layers; attention-side extensions remain open.

Future Work

- Scale to larger models and hardware clusters.
- Extend MoE-convolution equivalence to attention.
- Combine with retrieval or other conditional computation mechanisms.

Thank you

Questions?

cMoLLM at Scale
Horizontal Scaling Laws for Convolutionally-Gated Mixture-of-LLMs