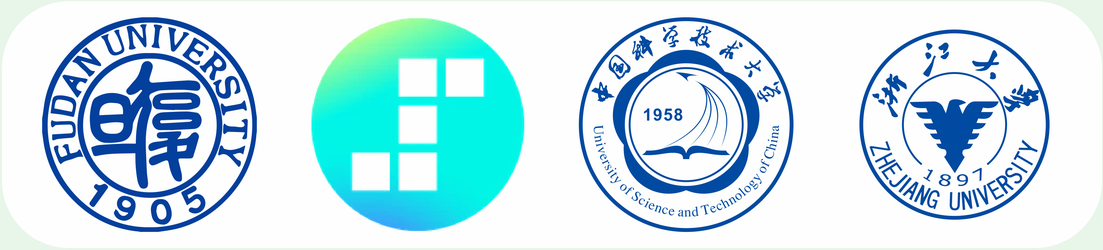
ICLR

ICLR 2026 · ORAL PRESENTATION

Mixture-of-Experts Can Surpass Dense LLMs Under Strictly Equal Resource

Houyi Li[1,2] **Ka Man Lo**[2] **Shijie Xuyang**[1] **Ziqi Wang**[3] **Wenzhen Zheng**[2]
Haocheng Zhang[1] **Zhao Li**[4] **Shuigeng Zhou**[1*] **Xiangyu Zhang**[2] **Daxin Jiang**[2]
[1]Fudan University [2]StepFun [3]University of Science and Technology of China [4]Zhejiang University



🤗 huggingface.co/kamanphoebe/moe_surpass_dense

1. Motivation: The Parameter-Subsidy Blind Spot

How do prior works compare MoE vs. Dense? They leave **Total Parameters (*N*)** uncontrolled:

Perspective A — Data-Centric (Fix *D*)

MoE matches Dense with **less compute**, but $N_{\text{MoE}} \gg N_{\text{Dense}}$.

e.g., DeepSeekMoE [1] 16B vs. 7B Dense

Perspective B — Compute-Centric (Fix *C* & N_a)

Sparser MoE → better perf, but *N* swells to **~100× dense baseline**.

e.g., Kimi K2 [2] 1T total, 32B active

Why must we control *N*?

***N* = Model Capacity**

= HBM Footprint = Deployment Cost

Without controlling *N*, we **cannot distinguish**:

Does MoE match Dense because *sparse architecture has no inherent capacity loss*?

Or simply because *extra parameters compensate for it*?

When prior work says "MoE matches Dense," it really means: ***MoE got a parameter subsidy.***

[1] Dai et al., DeepSeekMoE, 2024 [2] Kimi Team, Kimi K2, 2025

2. Our Core Research Question

Can MoE surpass Dense LLMs under equal *N*, *C*, and *D*?

Can we train a **Sparse LLM** comparable to a Dense model of the **same total size** (but with **lower inference cost**), using **equal or less training computation** and the **same unique tokens**?

This question is critical for any team building SOTA models.
We believe posing a truly valuable question is itself a core contribution.

To rigorously answer it, we paid an extreme experimental cost:

250+ LLM	50 Trillion	Thousands of GPU
pretrained from scratch	tokens consumed in total	GPU cluster for training

3. Equal *N* and *C*: The Fundamental Trade-off

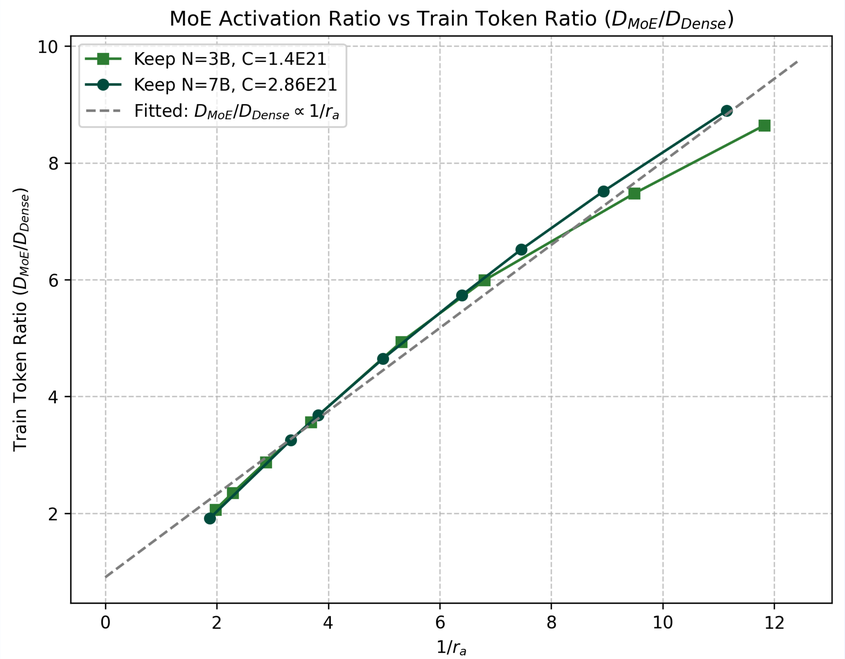


Figure 1a: Train Token Ratio vs. 1/r_a across 3B & 7B

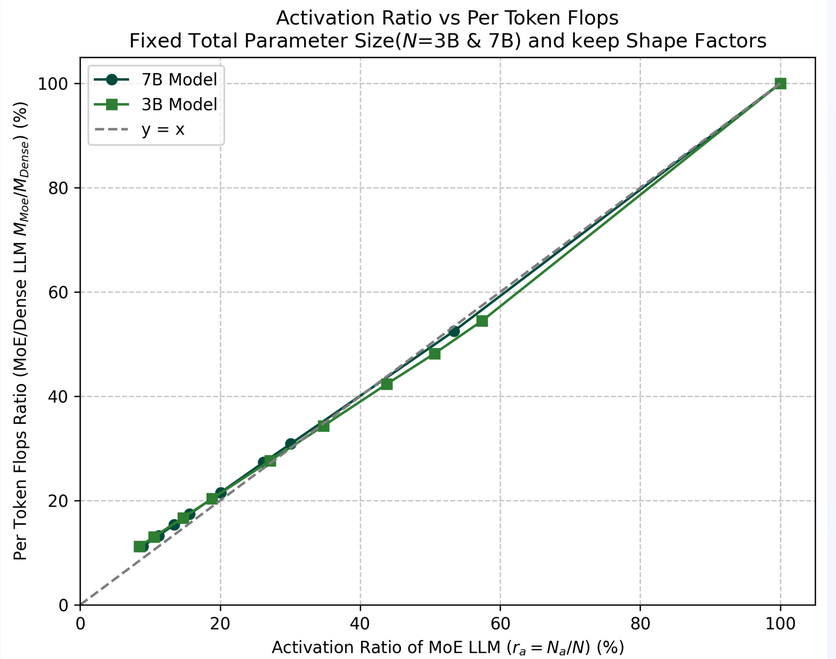


Figure 1b: Per-Token FLOPs Ratio vs. r_a across 3B & 7B

$$\frac{D_{\text{MoE}}}{D_{\text{dense}}} \approx r_a^{-1} \frac{\kappa_{\text{Dense}}}{\kappa_{\text{MoE}}} \propto \frac{1}{r_a}$$

$$\frac{M_{\text{MoE}}}{M_{\text{dense}}} \approx r_a \cdot \frac{\kappa_{\text{MoE}}}{\kappa_{\text{Dense}}} \propto r_a$$

Derivation: Per-token forward FLOPs:

$M_{\text{dense}} \approx 2N \cdot \kappa_{\text{Dense}}, \quad M_{\text{MoE}} \approx 2r_a N \cdot \kappa_{\text{MoE}}$

where $r_a = N_a/N, \quad \kappa = 1 + \frac{2\gamma}{4+3\alpha}, \quad \gamma = S/D_m, \alpha = D_{\text{ffn}}/D_m, \beta = D_{\text{act}}/D_m$

Equal *C* and *N*: $C = M_{\text{dense}} D_{\text{dense}} = M_{\text{MoE}} D_{\text{MoE}}$

$\Rightarrow \kappa_{\text{Dense}} D_{\text{dense}} = r_a \, \kappa_{\text{MoE}} D_{\text{MoE}}$

Key Insight: Two sides of one coin —

→ MoE's per-token FLOPs are **proportional to** r_a compared to Dense.

→ Its training token demand is **proportional to** $1/r_a$.

→ The sparser the MoE, the more data it demands under equal *N* & *C*.

Derivation omits minor terms (attention head shape, dense sublayers in MoE). See paper for full form.

4. Methodology: Three-Step Pipeline

🏗️ Step 1: Optimal MoE Architecture Search

1.1 Layer ratio (L_e vs L_d) + Shared Expert

1.2 FFN param allocation (Top-K)

1.3 Shape hyperparameters (ζ, β)

→ Lock structural choices (e.g., 1-dense, Top-K) that do not affect κ_{MoE}, and fix κ_{MoE}

📊 Step 2: Activation Rate

With optimal architecture from Step 1, sweep r_a at **fixed N and C**.

Identify optimal r_a^* region.

→ (κ_{MoE} is now a constant)

🔄 Step 3: Data Reuse

Fix **unique tokens** to match Dense.

Multi-epoch training offsets MoE's token hunger.

→ Strict equal N, C, D comparison

Each step builds upon conclusions from the previous → **Greedy optimization** of MoE design space

Controlled-variable discipline: Dense baselines are *also* tuned (aspect ratio & FFN expansion from Li et al., 2025). Hyperparameters via **StepLaw**, applied **uniformly** to both Dense and MoE — no "heavily-tuned MoE vs. neglected Dense."

Step1: Optimal MoE Architecture Search

📝 Architecture Search Conclusion Table

Component	Optimal Setting
Layer Arrangement	`1dense+SE` (One initial dense layer + MoE with Shared Expert)
Gate Normalization	Normalized Top-K gating works for $K > 1$
Top-K Routing	Intermediate K (avoid K=1 or overly large K)
Shape Ratios	$\zeta \approx 88$ (Hidden/Layers) and $\mu \approx 22$ (MoE FFN ratio)

🔬 Experiments Results

Table 1: Layer Arrangement
(Optimal: 1 dense layer + SE)

N	N_a	M	H	D_h	L	E	K	D_e	D_{se}	Scheme	$\mathcal{L}$	Conclusion
2.02B	346M	8.77e8	22	64	16	35	2	800	1600	full+SE	1.6813	`interleave` > `full`
2.02B	346M	8.77e8	22	64	16	68	2	800	1600	interleave+SE	1.6766	
2.02B	346M	8.77e8	22	64	16	70	4	800	0	interleave	1.6697	
2.15B	366M	6.63e9	11	128	16	85	5	352	1760	1dense+SE	1.8700	`1dense+SE` is best
2.15B	366M	6.63e9	22	64	16	85	5	352	1760	1dense+SE	1.8557	
2.15B	367M	6.63e9	11	128	16	70	4	800	0	interleave	1.8737	
2.15B	367M	6.63e9	22	64	16	70	4	800	0	interleave	1.8620	
2.15B	368M	9.31e8	22	64	17	37	4	800	0	1dense	1.6752	D_{se} ratio impacts little
2.15B	368M	9.31e8	22	64	17	36	3	800	800	1dense+SE	1.6712	
2.15B	368M	9.31e8	22	64	17	35	2	800	1600	1dense+SE	1.6726	

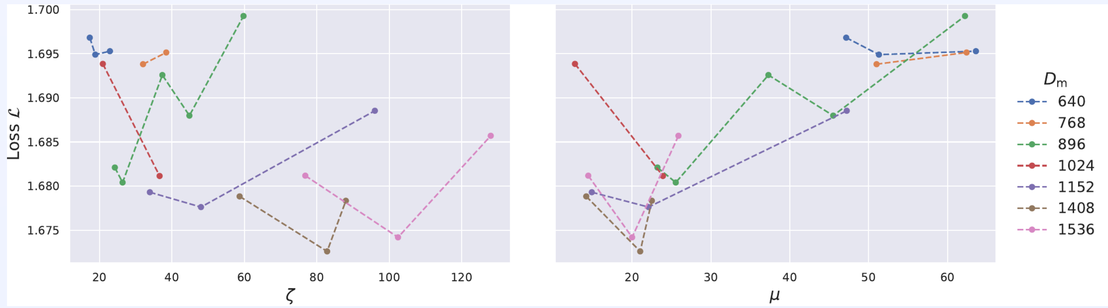
Table 2: Gate Normalization
(Norm before Top-K stabilizes training)

N	N_a	r_a (%)	M	E	K	D_e	D_{se}	Norm	$\mathcal{L}$	$\bar{\mathcal{L}}_{balance}$
2.15B	368M	17.08	9.31e8	35	2	800	1600	Y	1.6726	**1.355**
2.15B	368M	17.08	9.31e8	35	2	800	1600	N	1.6712	1.452
2.15B	368M	17.08	9.31e8	37	4	800	0	Y	1.6752	**1.409**
2.15B	368M	17.08	9.31e8	37	4	800	0	N	1.6750	1.440

Table 3: Top-K Routing
(Intermediate K=2/3 yields best BPC)

N	N_a	r_a (%)	M	E	K	D_e	D_{se}	$\mathcal{L}$
2.15B	591M	27.47	8.00e9	8	1	3528	3528	2.0470
2.15B	591M	27.40	8.00e9	88	11	320	3520	**2.0338**
2.15B	949M	44.00	1.01e10	8	2	3176	6352	**1.9996**
2.15B	948M	44.05	1.01e10	88	22	288	6336	2.0266
2.15B	1.24B	57.57	1.19e10	8	3	2888	8664	**2.0156**
2.11B	1.22B	57.68	1.18e10	88	33	256	8448	2.0235

Figure 2: Shape Ratios
(Optimal $\zeta \approx 88$, $\mu \approx 22$)



Step2: Optimal AR Search at 2B

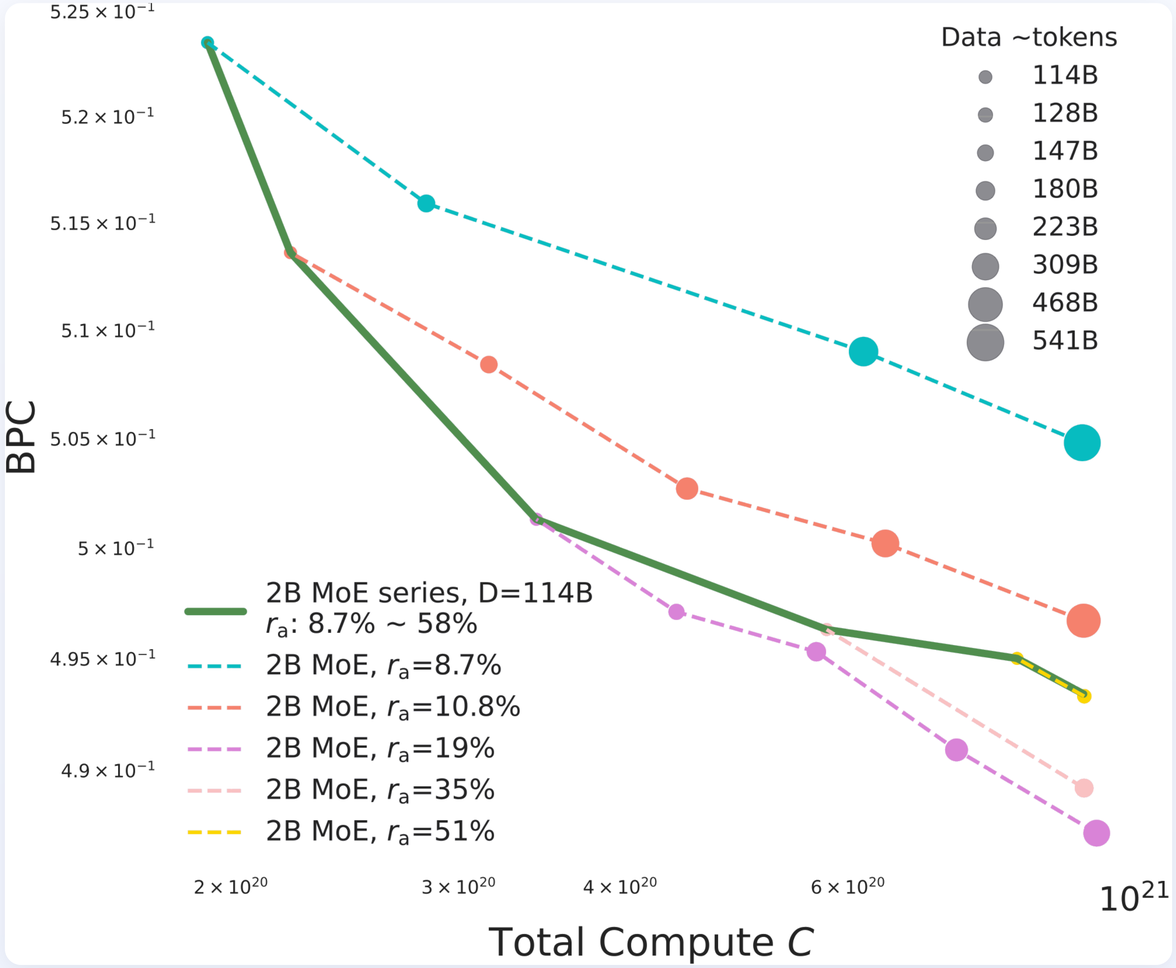


Figure 3(a): Fixed r_a, add D → log-log linear. Fixed D, raise r_a → NOT log-log linear. First finding — loss is not log-log-linear in compute in general.

- **Fixed r_a, vary D**: Loss and Computation relationship is almost log-log linear. Compute only scales with Data tokens.
- **Fixed D, vary r_a**: Non-linear relationship. Compute only scales with r_a.

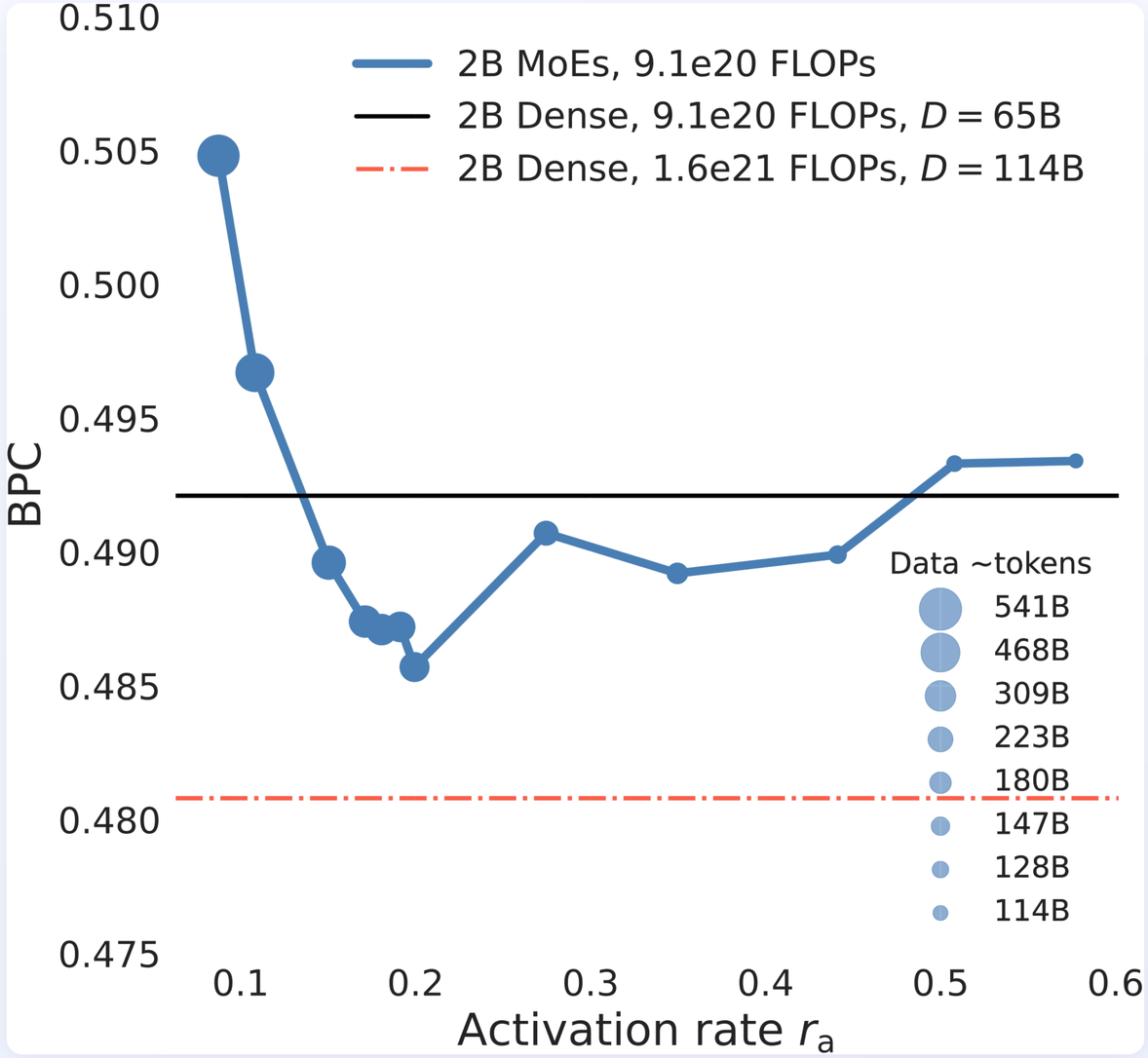


Figure 3(b): Fixed N, C, sweep r_a ⇒ optimal region around 20%. Second finding — is not "More sparse is better".

- **Optimal Region**: At Fixed N & C, MoE beats Dense around $r_a^* \approx 20\%$, utilizing ~4.8× unique tokens.

7. Step3A: Optimal AR Search and Data-Reuse at 3B

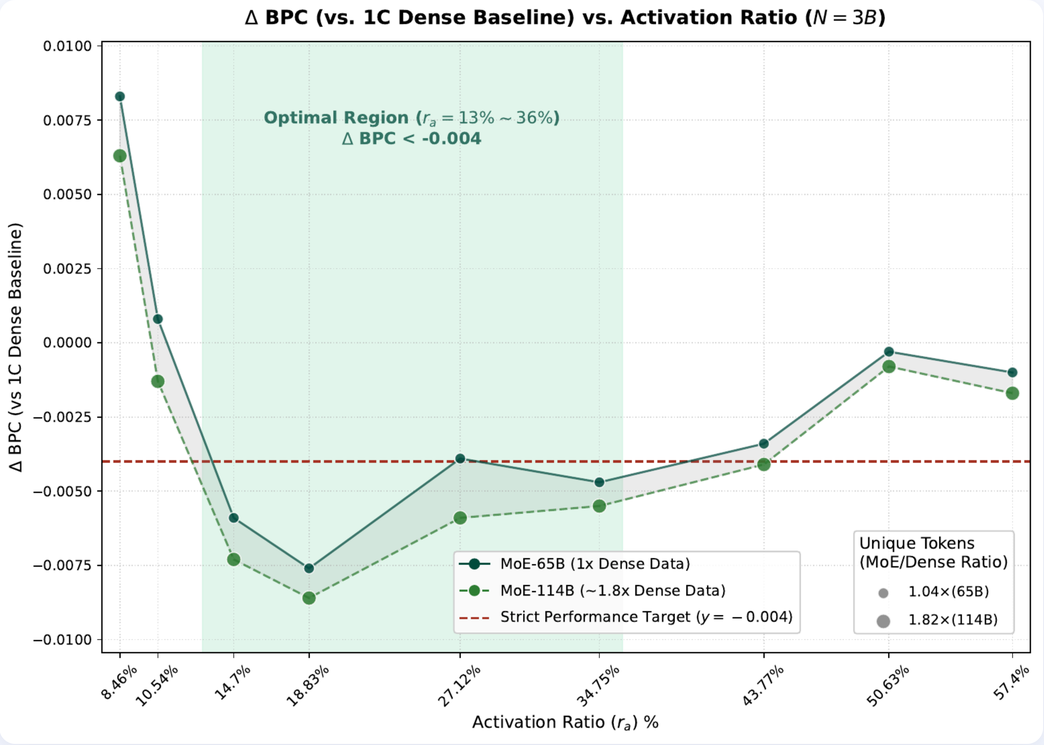


Figure 4: Data Reuse at 3B Scale

Finding: Optimal region is also around 20%. **Data Reuse works.** MoE is comparable with Dense LLM at same *N*, same *C*, and even same unique tokens.

Blue Line: MoE Unique Data

Green Dashed: Strict Data Reuse

Red Dashed: Dense Baseline

Table 4: 3B Data Reuse and AR Search Results.

Name	Computation	Act Ratio	FLOPs/Tok	Train Tok	Unique Tok	Reuse	ΔBPC
Baseline1	1.00×	100%	1.00×	1.00×	1.00×	1.00	**0.0000**
MoE-65B-Exp3	1.00×	14.7%	16.7%	5.99×	1.04×	5.77	**-0.0059**
MoE-65B-Exp4	1.01×	18.8%	20.4%	4.94×	1.04×	4.75	**-0.0076**
MoE-65B-Exp5	0.99×	27.1%	27.6%	3.56×	1.04×	3.43	**-0.0039**
MoE-65B-Exp6	0.99×	34.8%	34.3%	2.88×	1.04×	2.77	**-0.0047**

Key Takeaways:

1. At **fixed *N* and *C***, MoE outperforms Dense LLM.
2. These MoE configurations use the **exact same amount of Unique Tokens** (1.04× ≈ 1×) as the baseline.
3. Operating at **16.70% ~ 34.33% Per-token FLOPs** yields significant inference speedup!

Step3B: Scaling to 7B — AR Analysis & DataReuse

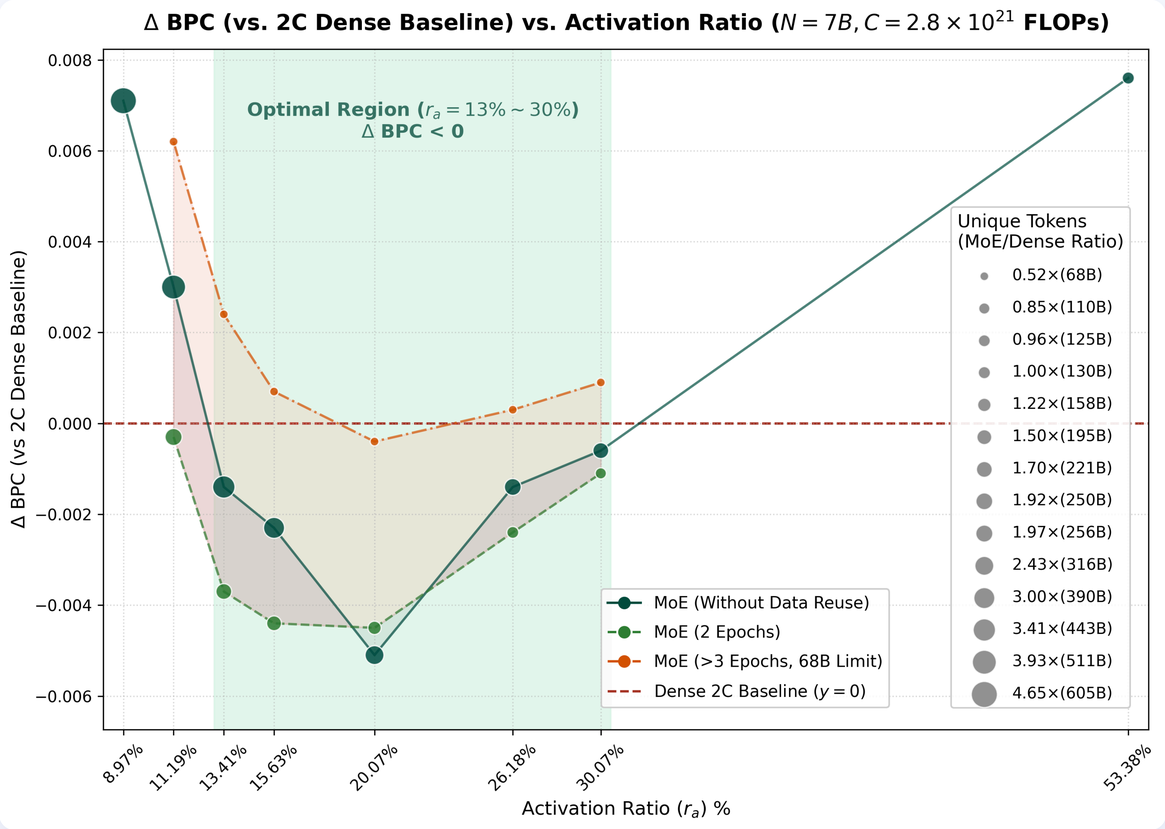


Figure 5: 7B Scale: ΔBPC vs Dense-2C Baseline under various Data Reuse configurations.

Red Dashed: Dense-2C Baseline

Blue Line: MoE Unique Data

Green Dashed: 2-Epoch Reuse

Orange Dashed: Strict Reuse (68B)

Optimal AR Range (around 20%) is consistent with 2B/3B

Note: Since all MoE models easily surpass the Dense-1C baseline under equal N and C, we omit it here and plot against the stronger **Dense-2C** baseline.

- 🔴 **Red dashed:** Dense-2C baseline (2× Computation, 130B tokens)
- 🔵 **Blue solid (MoE-Unique):** No reuse; $r_a \in [13\%, 30\%]$ → surpasses Dense-2C
- 🟢 **Green dashed (MoE-2Ep):** 2-epoch reuse (½ unique tokens); $r_a \in [11\%, 30\%]$ → still outperforms
- 🟠 **Orange dashed (MoE-68B):** Strict reuse (only 68B unique); comparable to Dense-2C **only at $r_a \approx 20\%$**

Takeaways:

1. **Optimal AR Range is scale invariance also around 20% (13%~30%).**
2. Even with **strictly equal unique tokens** (68B), MoE at optimal r_a matches Dense trained with **2× compute and 2× data**.

9. The Sweet Spot & Data Reuse Impact

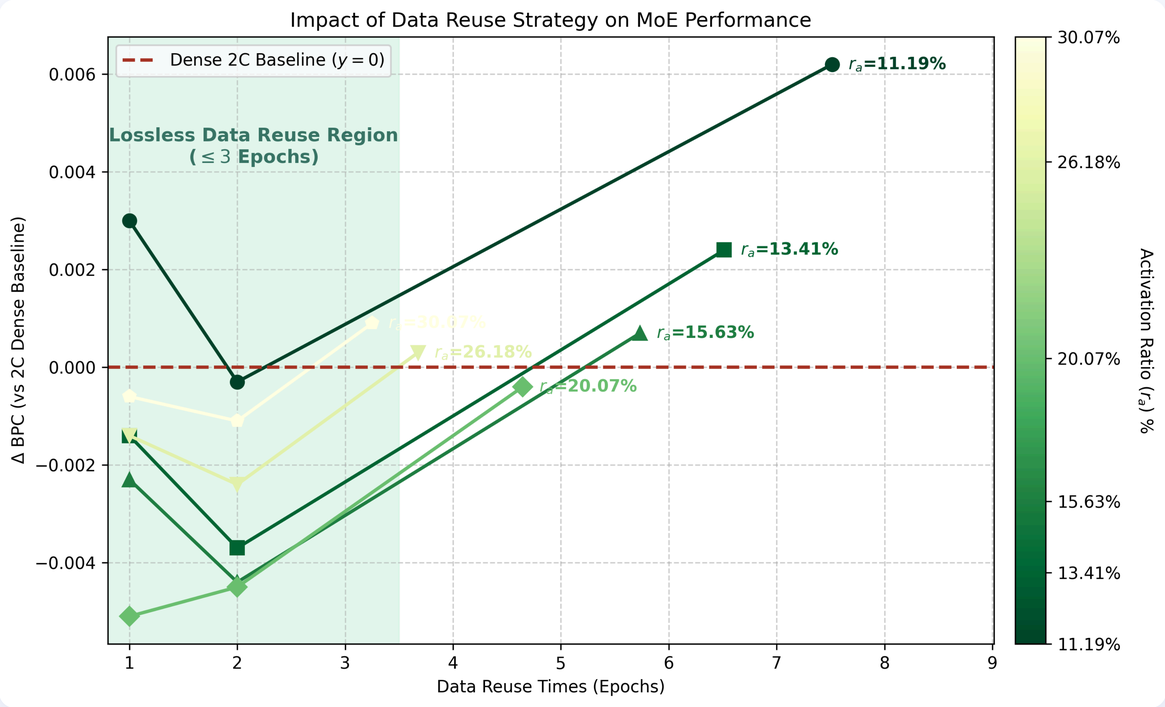


Figure 6: Impact of Data Reuse Strategy on MoE Performance

📊 **Lines Explanation:** Each colored curve represents an MoE model with a specific **Activation Rate (r_a)**. The X-axis indicates the number of Data Reuse Epochs.

- ✅ **Moderate Data Reuse Boosts MoE:** 2 epochs boost BPC at $r_a \in \{13\%, 16\%, 20\%\}$
- ✅ **≤3 epochs:** MoE advantage is preserved
- ⚠️ **>3 Epochs:** Performance degrades sharply

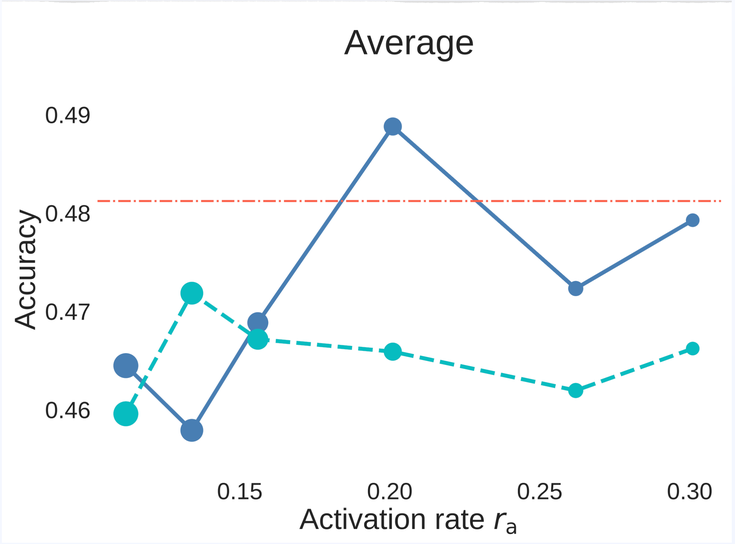
🏆 The Sweet Spot — MoE-68B-Exp4 at r_a = 20%

1/2 compute of Dense-2C, 1/2 unique tokens, same N, BPC matches Dense-2C1, at 21.5% per-token FLOPs ⇒ ~ 5× inference speedup at equal memory.

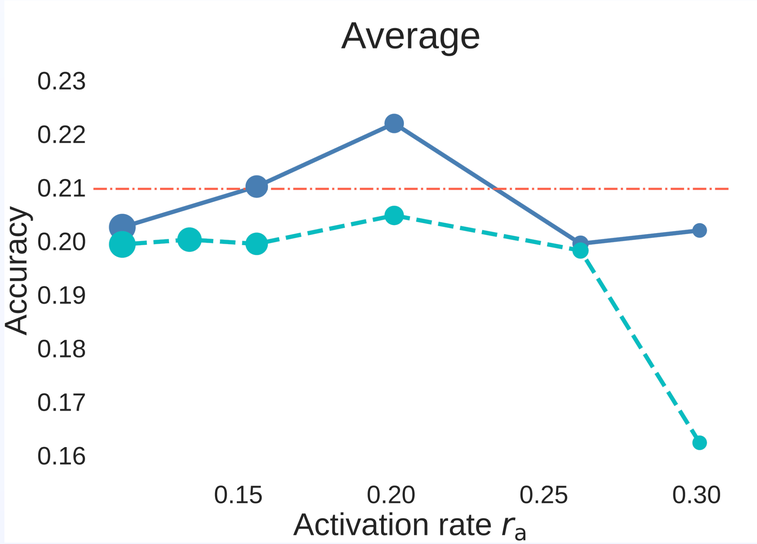
Table 5: 7B Resource-Ratio comparison under different training strategies.

Model / Strategy	r_a (%)	Unique (D)	Epochs	FLOPs/Tok	BPC	Δ vs 2C
Dense 1C	100%	68 B	1.00	100%	0.4736	+0.0142
Dense 2C (Red line)	100%	130 B	1.00	100%	0.4594	**0.0000**
MoE-Unique	11.19%	511 B	1.00	13.3%	0.4624	+0.0030
MoE-Unique	13.41%	443 B	1.00	15.3%	0.4580	-0.0014
MoE-Unique	15.63%	390 B	1.00	17.4%	0.4571	-0.0023
MoE-Unique (Blue)	**20.07%**	**316 B**	**1.00**	**21.5%**	**0.4543**	**-0.0051**
MoE-Unique	26.18%	250 B	1.00	27.2%	0.4580	-0.0014
MoE-Loose	11.19%	256 B	2.00	13.3%	0.4591	-0.0003
MoE-Loose	13.41%	221 B	2.00	15.3%	0.4557	-0.0037
MoE-Loose	15.63%	195 B	2.00	17.4%	0.4550	-0.0044
MoE-Loose (Green)	**20.07%**	**158 B**	**2.00**	**21.5%**	**0.4549**	**-0.0045**
MoE-Loose	26.18%	125 B	2.00	27.2%	0.4570	-0.0024
MoE-68B	11.19%	68 B	7.52	13.3%	0.4656	+0.0062
MoE-68B	13.41%	68 B	6.51	15.3%	0.4618	+0.0024
MoE-68B	15.63%	68 B	5.74	17.4%	0.4601	+0.0007
MoE-68B (Orange)	**20.07%**	**68 B**	**4.65**	**21.5%**	**0.4590**	**-0.0004**
MoE-68B	26.18%	68 B	3.67	27.2%	0.4597	+0.0003

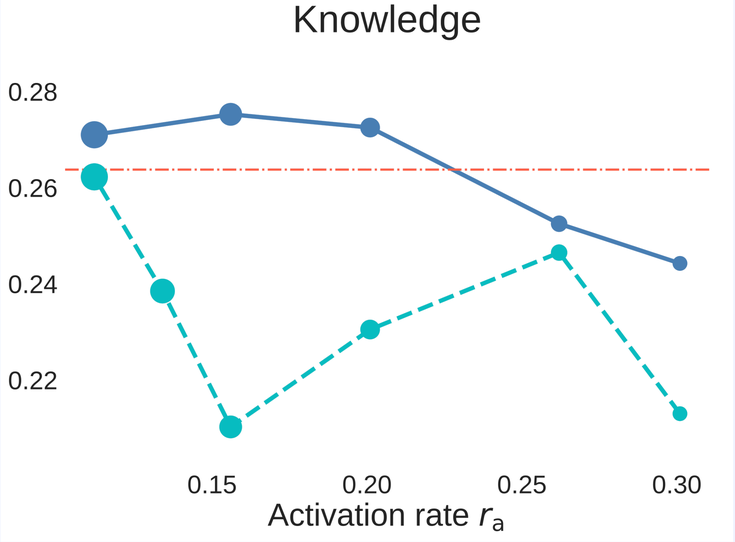
Step3C: Downstream Task Evaluation (7B)



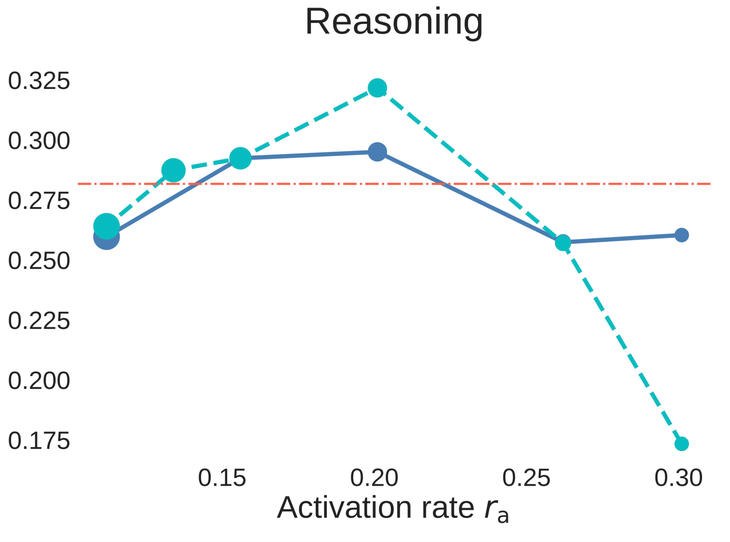
(a) Pretrain Average



(b) SFT Average



(c) SFT Knowledge



(d) SFT Reasoning

Figure 7: *Downstream benchmark performance of 7B aligned models.*

🔍 Key Findings

- **Unique Tokens > Dense:** At $r_a \in [16\%, 20\%]$, the Average Benchmark score is **significantly higher** than the Dense LLM.
- **Unique Tokens = ½ Dense:** At the optimal r_a, the Average Benchmark score **matches** the Dense-2C model.
- Trade-off: **Knowledge** recall degrades with fewer unique tokens; **Reasoning** is largely **preserved** and even surpasses Dense at optimal r_a with strict reuse.

📊 Chart Legend

Blue Line: MoE Unique Data

Green Dashed: Strict Data Reuse

Red Dashed: 2C-Dense Baseline

Guidance for Building SOTA MoE LLM

1. Equal-Ground Superiority
MoE can match/surpass Dense under strictly **equal** N, C, and unique data.

2. Scale-Invariant Optimal Sparsity
"Sparser is better" is a misconception. **Optimal AR Region (10%~30%)** is consistent across scales.

3. The Fundamental Trade-off
MoE trades higher token consumption for massive per-token FLOPs reduction (~5× inference speedup).

4. Data Reuse Works
Multi-epoch training (≤3 Ep) preserves MoE advantages. Reasoning improves; knowledge slightly degrades.

💡 Guidance for Building SOTA Models

Choose a safe sparsity and take a more aggressive data scaling strategy.

Example Scenario:
To train a **1T Dense Model** following Chinchilla, you need 20× data = **20T Tokens**.

But if you train a **1T MoE model with 100B active parameters** (10% AR), to match the 1T Dense performance, you need **200T Tokens**!

(You can use 3~5 epochs of data reuse if you don't have enough unique tokens).

Thank You!

Q & A

~200 LLMs at 2B · 50+ at 7B · 50T tokens processed · All checkpoints released

🤗 **huggingface.co/kamanphoebe/moe_surpass_dense**

For more detailed questions, contact:

hyli22@m.fudan.edu.cn

Houyi Li
LLM Researcher