Quiz 5 - Capacity

11/10

Problem 1: Depth vs Width

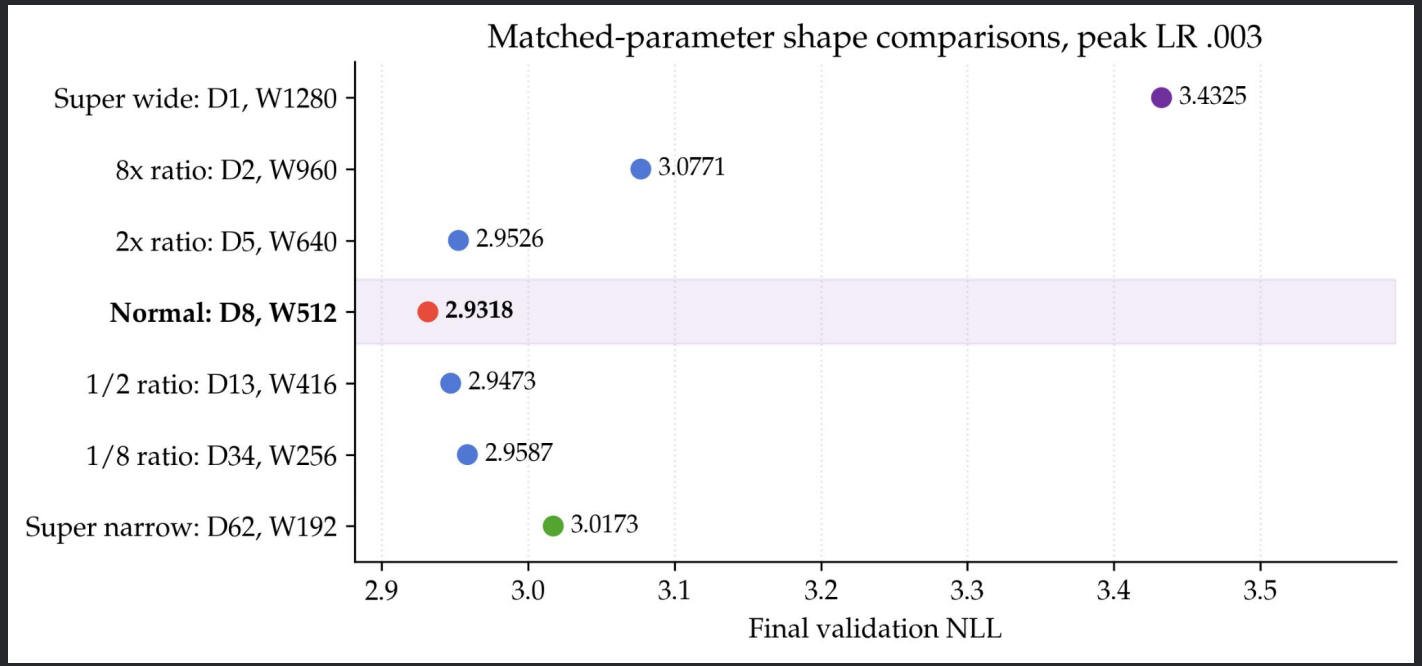
Which models do better or worse?

Model	D	W	W/D	Parameters (M)
Super wide	1	1280	1280.0	34.247
$8\times$ normal ratio	2	960	480.0	34.596
$2\times$ normal ratio	5	640	128.0	34.947
Normal	**8**	**512**	**64.0**	**34.613**
$\frac{1}{2}\times$ normal ratio	13	416	32.0	36.041
$\frac{1}{8}\times$ normal ratio	34	256	7.5	34.428
Super narrow	62	192	3.1	34.746

	Comparison
(a)	Super narrow versus super wide
(b)	$\frac{1}{2}\times$ normal aspect ratio versus $2\times$ normal aspect ratio
(c)	$\frac{1}{8}\times$ normal aspect ratio versus $8\times$ normal aspect ratio
(d)	Super narrow versus normal

Problem 1: Depth vs Width

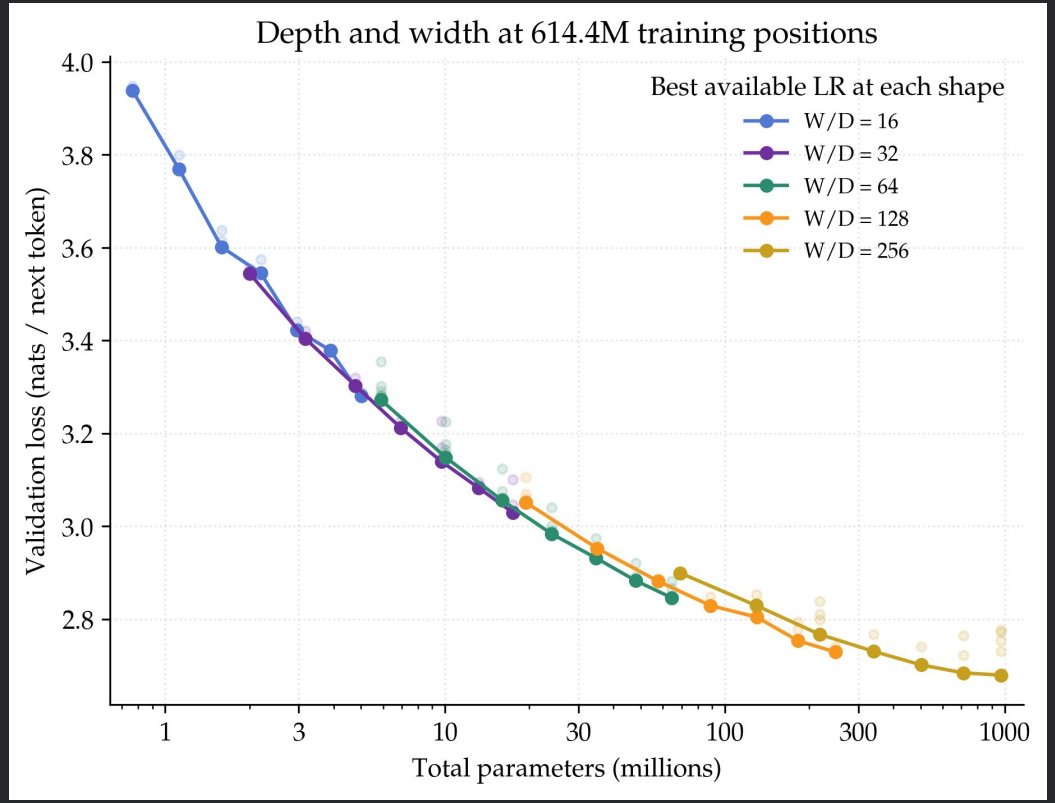
Answer: narrow < fat, 2x = 1/2x, 8x > 1/8x, narrow > normal



Problem 1: Depth vs Width

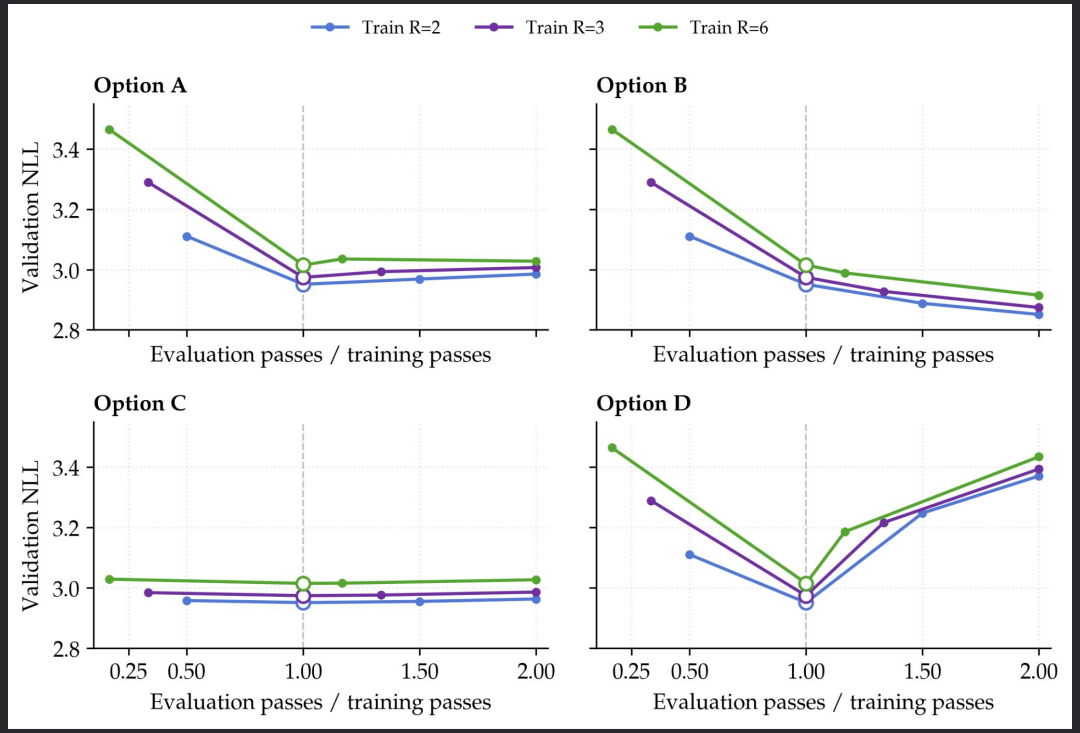
Reasoning: From assignment, larger regret from going too wide after a certain point

For a fixed parameter count, preferable to go deeper than wider



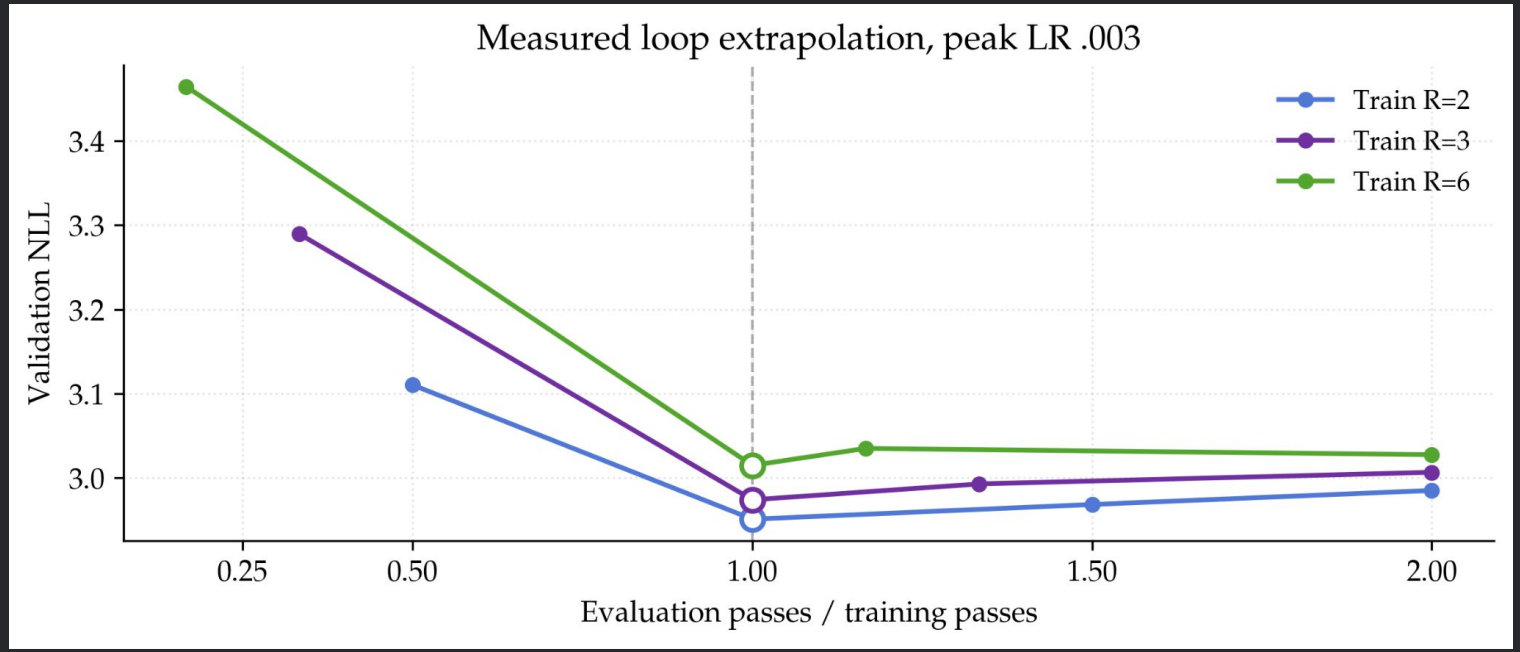
Problem 2: Looping

Part (a): Which plot best summarizes looping under extra inference?



Problem 2: Looping

Answer: Option A



Problem 2: Looping

Reasoning: Without explicit training, models don’t generalize well past the loop count they were trained with.

This is a common phenomenon in deep learning: transformers often don’t length generalize past their training context, reasoning models often don’t generalize to unseen effort levels, etc.

Problem 2: Looping

Part (b): Can you grow a looped model into an unlooped model?

Use the *plain* six-pass model (no extra boundary RMS or recall). All three plans use peak LR .003, and the reference token budget:

- **Loop:** keep the shared block throughout.
- **Grow:** halfway through training, clone the shared block into six independently trainable copies and continue.
- **Dense:** train standard d8 with independent blocks throughout.

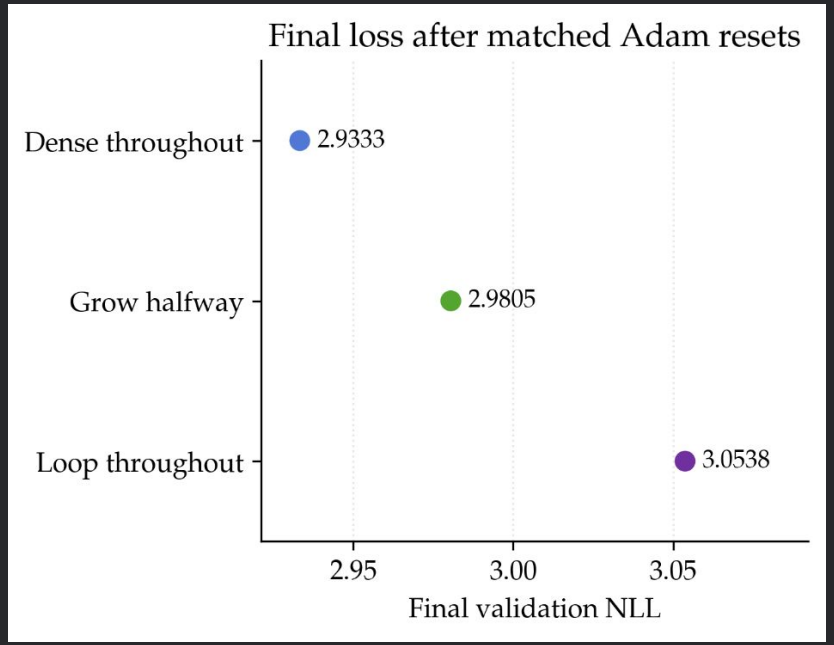
All reset Adam halfway and keep the data and LR clocks. Growth leaves the other weights unchanged. All execute eight layers throughout. Given $L_{\text{dense}} < L_{\text{loop}}$ and $M = (L_{\text{dense}} + L_{\text{loop}})/2$, where does L_{grow} fall?

- **A.** $L_{\text{grow}} \geq L_{\text{loop}}$: as bad as or worse than keeping the loop.
- **B.** $M \leq L_{\text{grow}} < L_{\text{loop}}$: between the baselines, closer to Loop (or exactly midway).
- **C.** $L_{\text{dense}} \leq L_{\text{grow}} < M$: between the baselines, closer to Dense.
- **D.** $L_{\text{grow}} < L_{\text{dense}}$: better than training dense throughout.

Record your choice, assignment evidence and rationale on the separate answer sheet.

Problem 2: Looping

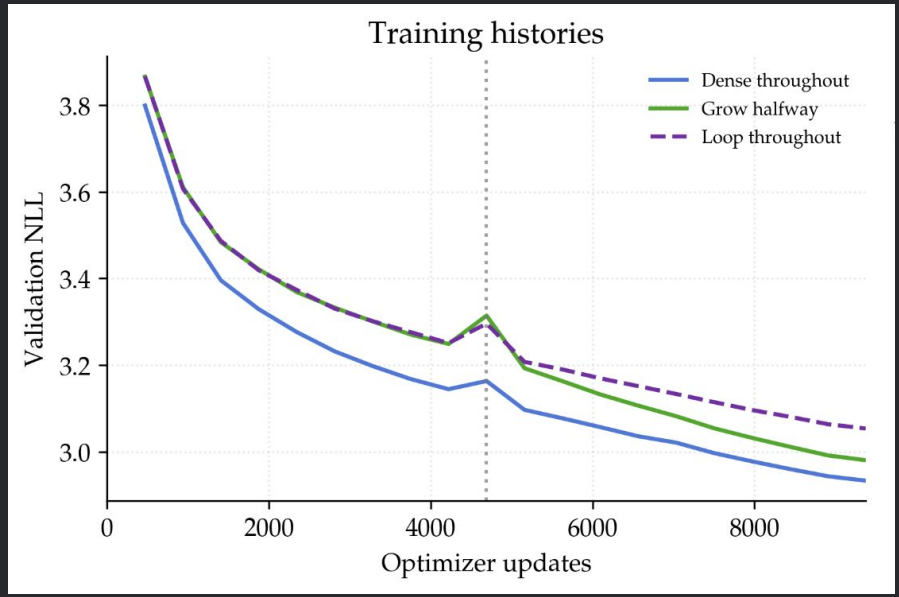
Answer: Loss is closer to dense than looped (option C)



Problem 2: Looping

Reasoning: Looped model starts off much worse, but dense allows it to recover most of the performance lost by looping

Looping can mostly be constrained by representation, overcome with sufficient dense training



Problem 3: Rank bottleneck

Part (a): Under the wide models, which interventions help performance?

Compare two versions of Q1's **super wide** $D = 1$, $W = 1280$ model:

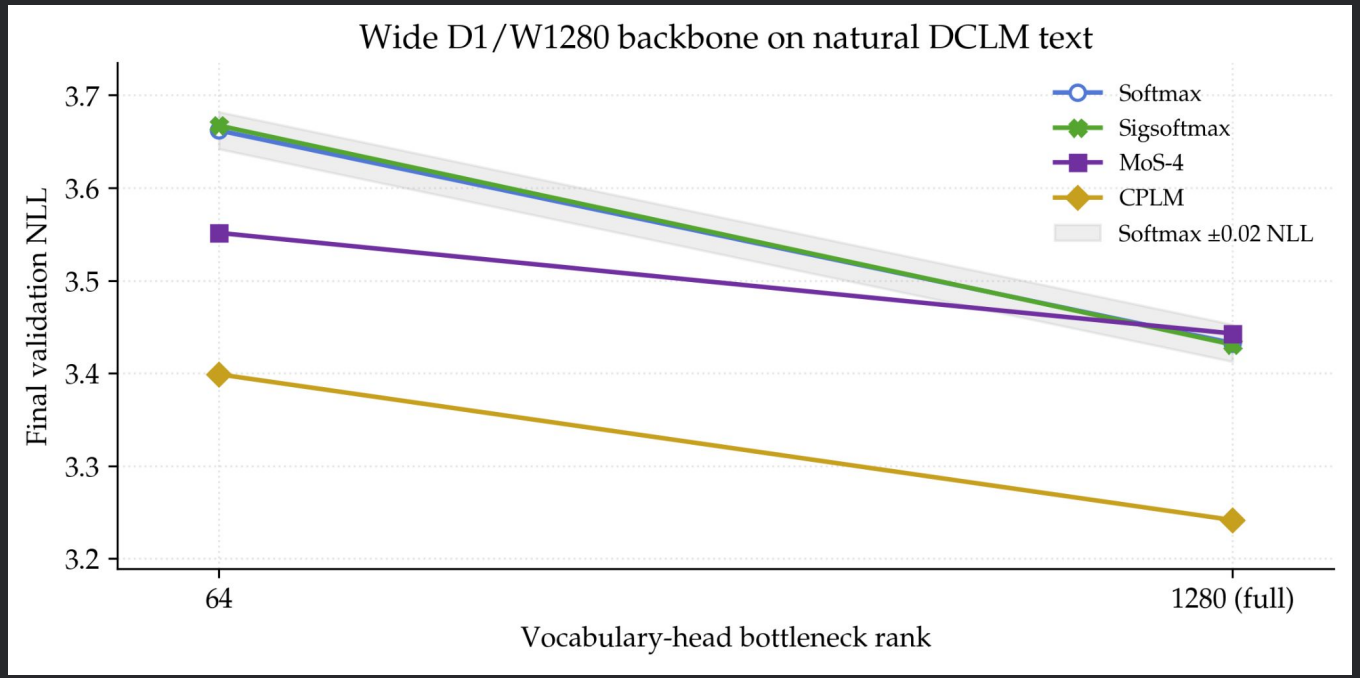
- **Wide:** the full-rank vocabulary head ($r = 1280$).
- **Wide + rank-64 bottleneck:** the same backbone with an induced rank-64 vocabulary-head bottleneck.

Use peak LR .003 for all methods; extra head parameters are allowed. For each model, compare Sigsoftmax, MoS-4 and CPLM to **its own Softmax baseline**. Write $<$, $=$ or $>$ in each box to compare the method's validation loss (left) with its same-rank Softmax loss (right). Gaps below .02 NLL count as equal (six comparisons, 1 point each).

Method	Rank 64			Rank 1280		
Sigsoftmax	L_{method}	☐	L_{Softmax}	L_{method}	☐	L_{Softmax}
MoS-4	L_{method}	☐	L_{Softmax}	L_{method}	☐	L_{Softmax}
CPLM	L_{method}	☐	L_{Softmax}	L_{method}	☐	L_{Softmax}

Problem 3: Rank bottleneck

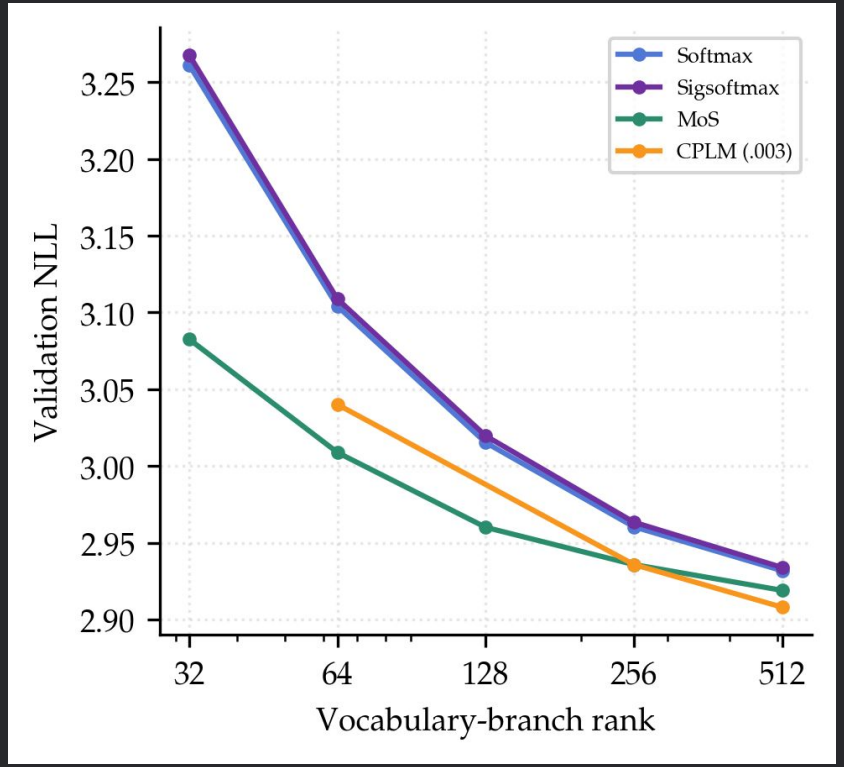
Answer: MoS-4 helps only at rank 64. CPLM helps at both ranks. Otherwise equal



Problem 3: Rank bottleneck

Reasoning: Assignment plots show that:

1) 1280 shouldn’t have an issue with rank bottleneck

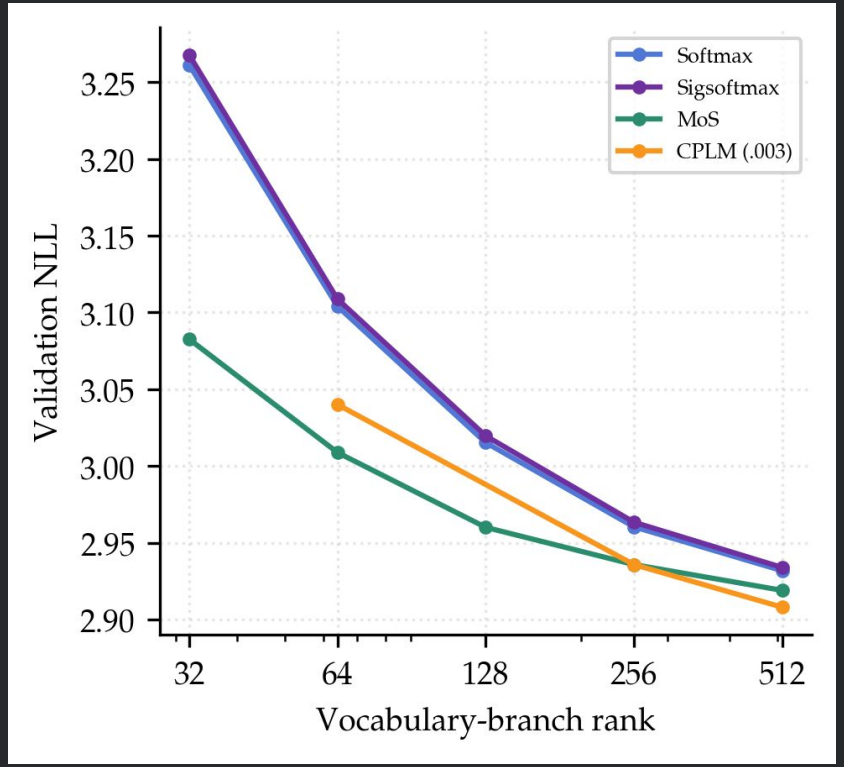


Problem 3: Rank bottleneck

Reasoning: Assignment plots show that:

2) sigsoftmax has no effect

adding a nonlinearity doesn't remove the actual softmax bottleneck from the model

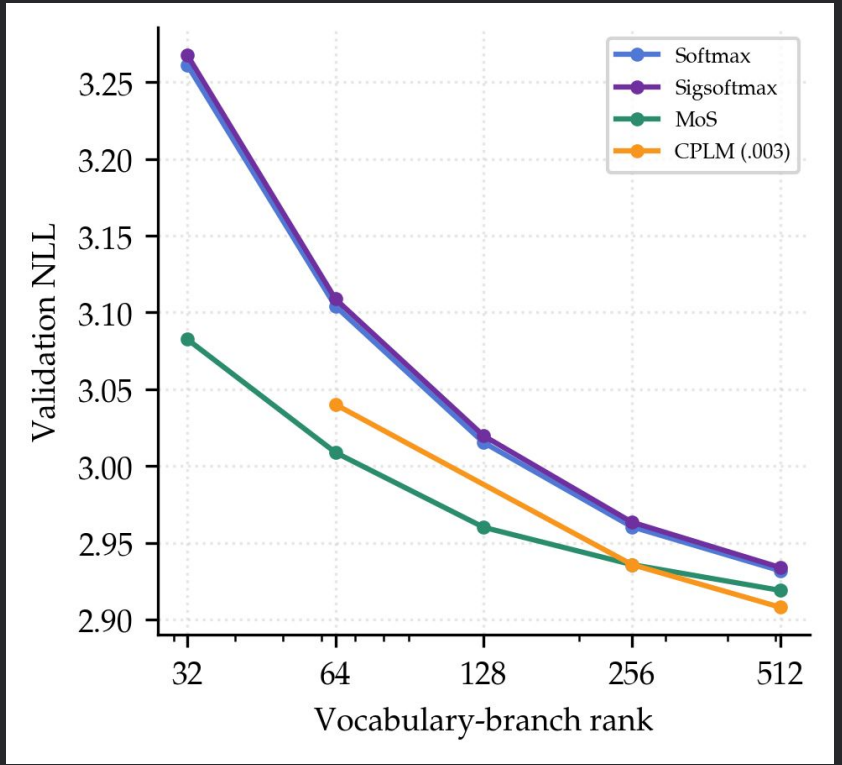


Problem 3: Rank bottleneck

Reasoning: Assignment plots show that:

3) MoS-4 helps most at small ranks

MoS-4 can be thought of as an effective rank multiplier, so unimportant when there isn't a strong rank bottleneck

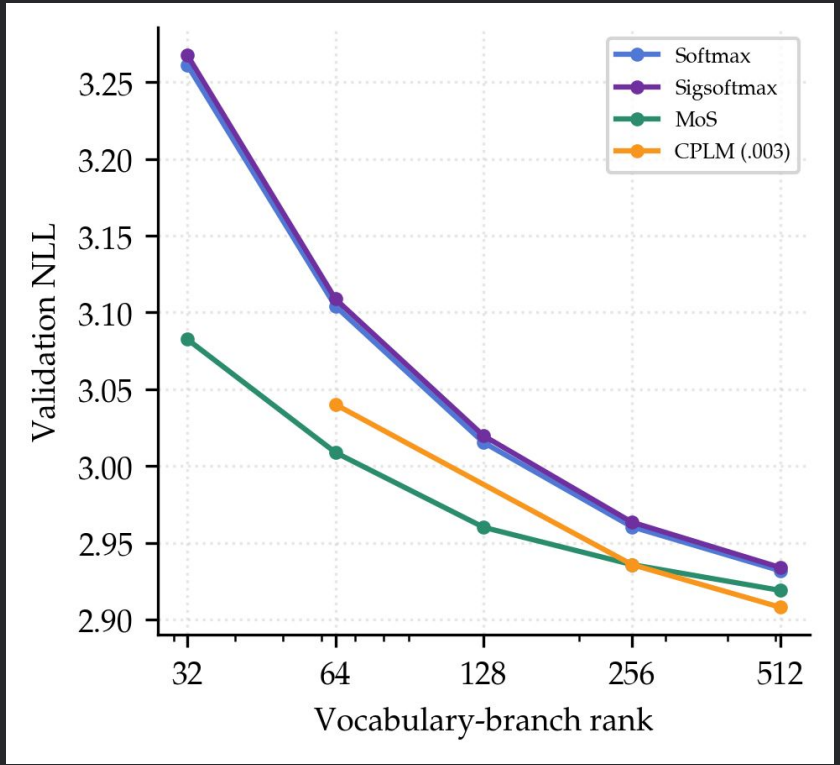


Problem 3: Rank bottleneck

Reasoning: Assignment plots show that:

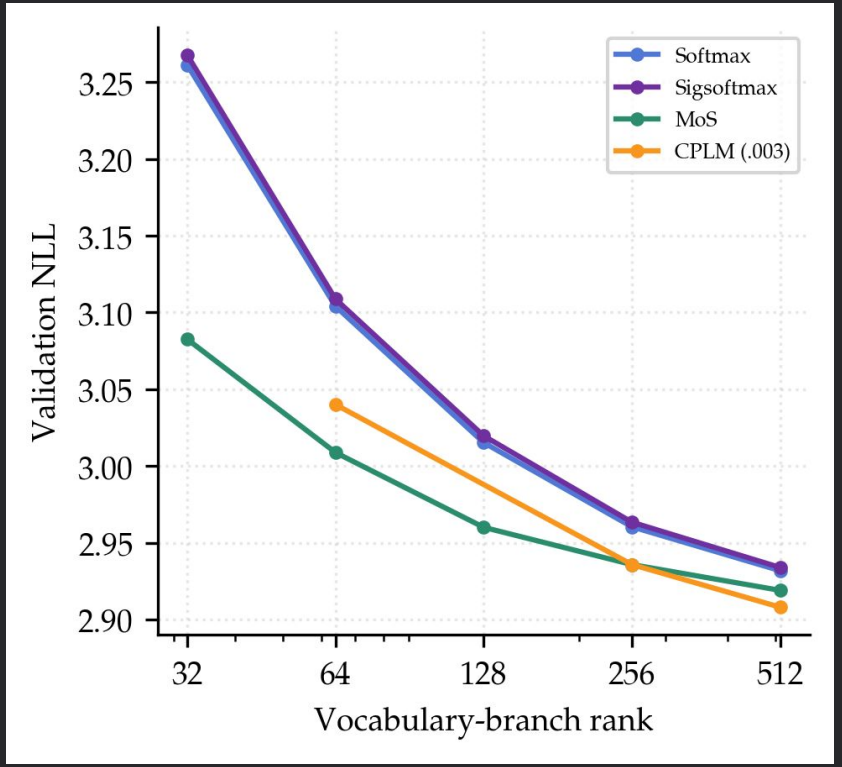
4) CPLM helps at all ranks

Though CPLM alleviates rank bottleneck, it seems to have benefits beyond this



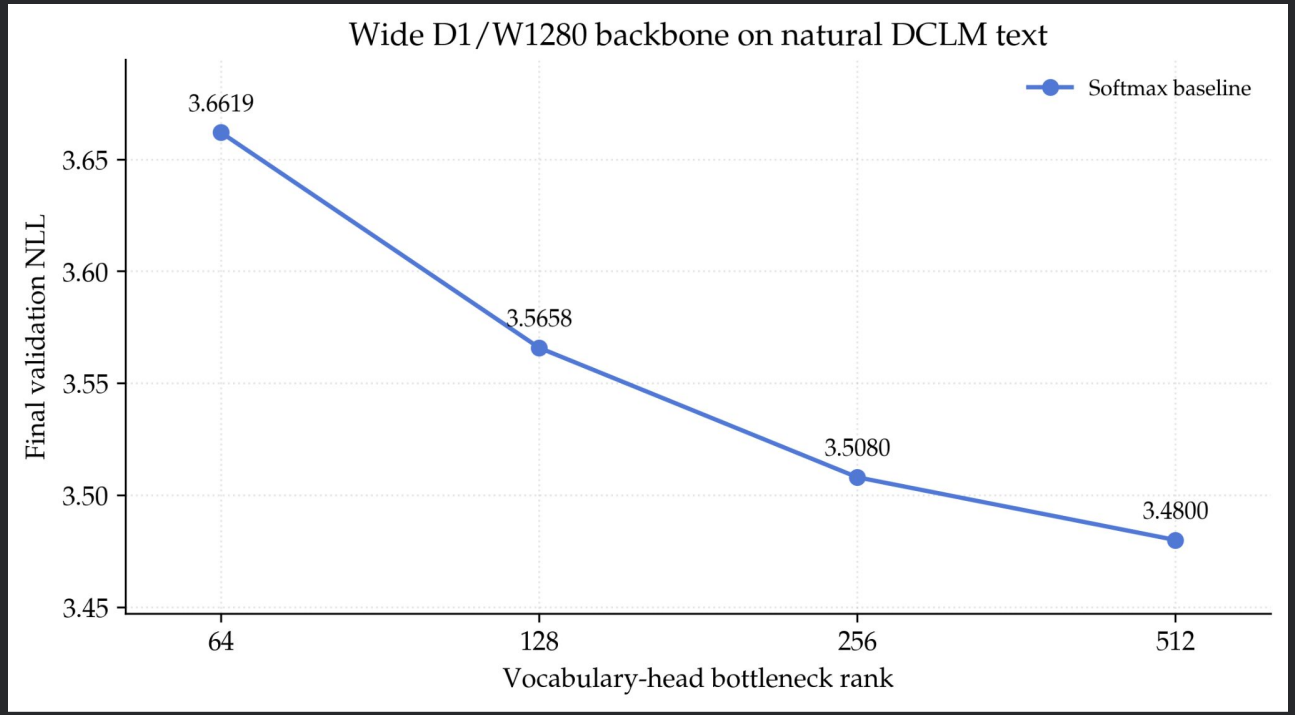
Problem 3: Rank bottleneck

General strategy: Can utilize when algorithms help to understand what issues the algorithms is alleviating



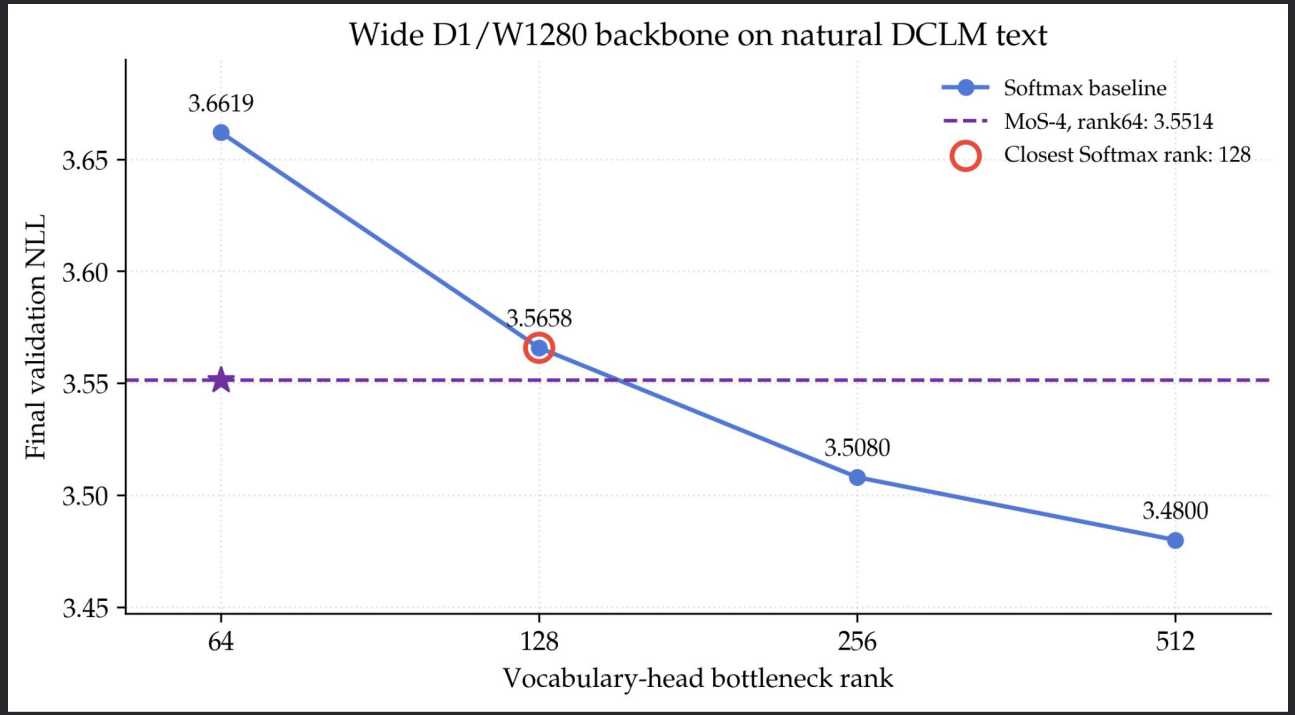
Problem 3: Rank bottleneck

Part (b): Which rank is MoS-4 closest to?



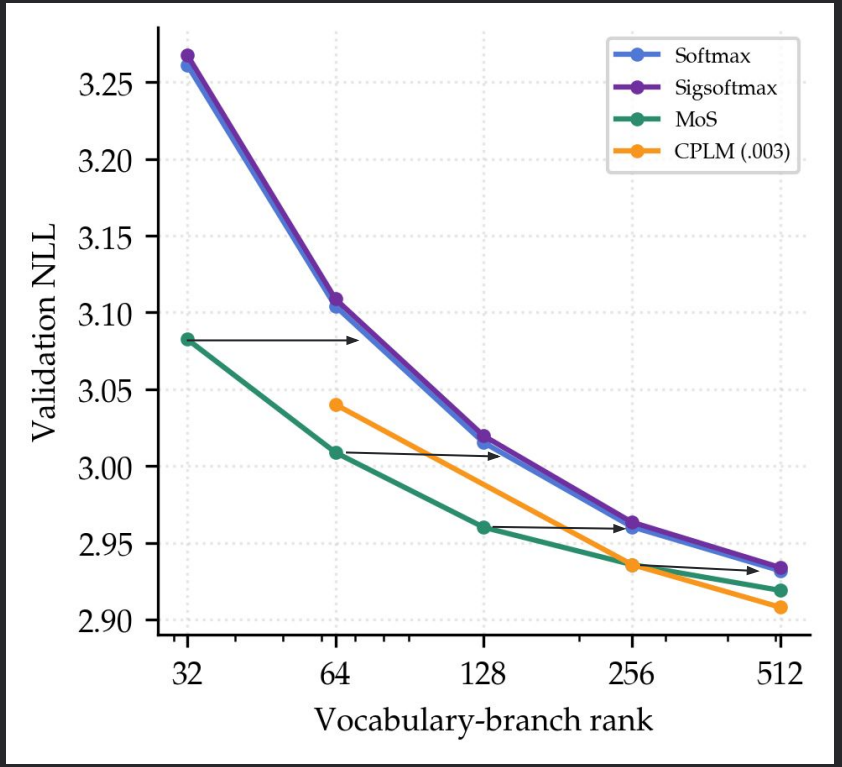
Problem 3: Rank bottleneck

Answer: Closest to 2x rank, or rank 128



Problem 3: Rank bottleneck

Reasoning: Even though the effective dimension is 4x larger, can see that 4 heads leads to a 2x rank multiplier across ranks



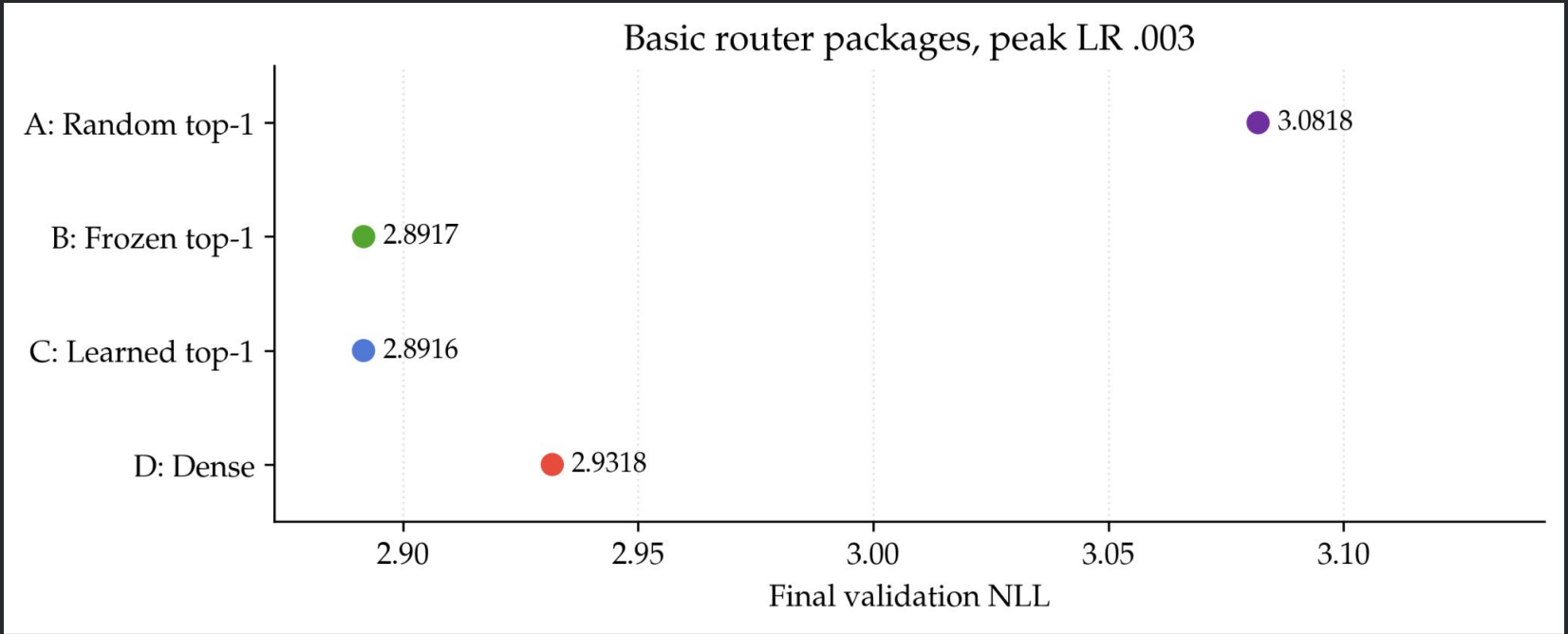
Problem 4: MoE loss comparisons

Part (a): How do these router losses compare?

Package	Intervention
A	Random top-1: independent uniform routing per token/application; unit combine weight.
B	Frozen top-1: hold a random router matrix fixed; train the backbone and experts.
C	Learned top-1: train the router, without a balancing loss.
D	Dense: standard d8.

Problem 4: MoE loss comparisons

Answer: B = C < D < A



Problem 4: MoE loss comparisons

Reasoning: Random routing should do poorly because the model can not specialize the experts to different inputs, so each expert gets a fourth of the total gradient. Learned routing should do better than dense

Main surprise is learned and frozen doing the same. This has also been seen in Fan et al, 2024

Dataset	Routing Level	Routing Strategy	Validation Perplexity *Sequence*	*Token*
Multilingual	Global	Language	11.716	-
	Global	Learned	11.605	-
	Layer-wise	Frozen	**10.601**	**9.810**
	Layer-wise	Random	11.685	11.752
	Layer-wise	Learned	**10.666**	**9.907**

Problem 4: MoE loss comparisons

Frozen routing is conceptually similar to Hash layers [Roller 2021], though still data dependent

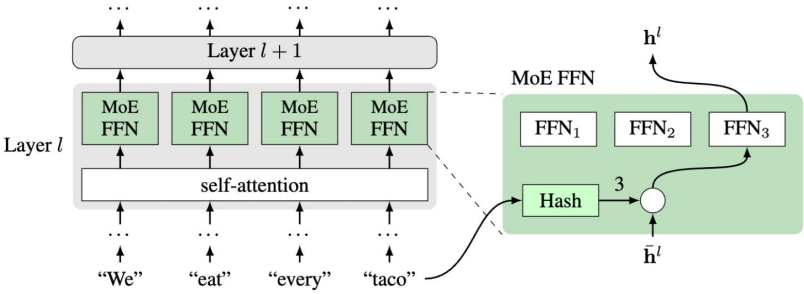


Figure 1: **Overview of the Hash Layer.** Tokens are routed to fixed expert modules based on their hash.

Intuition: Even if router is frozen, model can adapt embeddings to match router

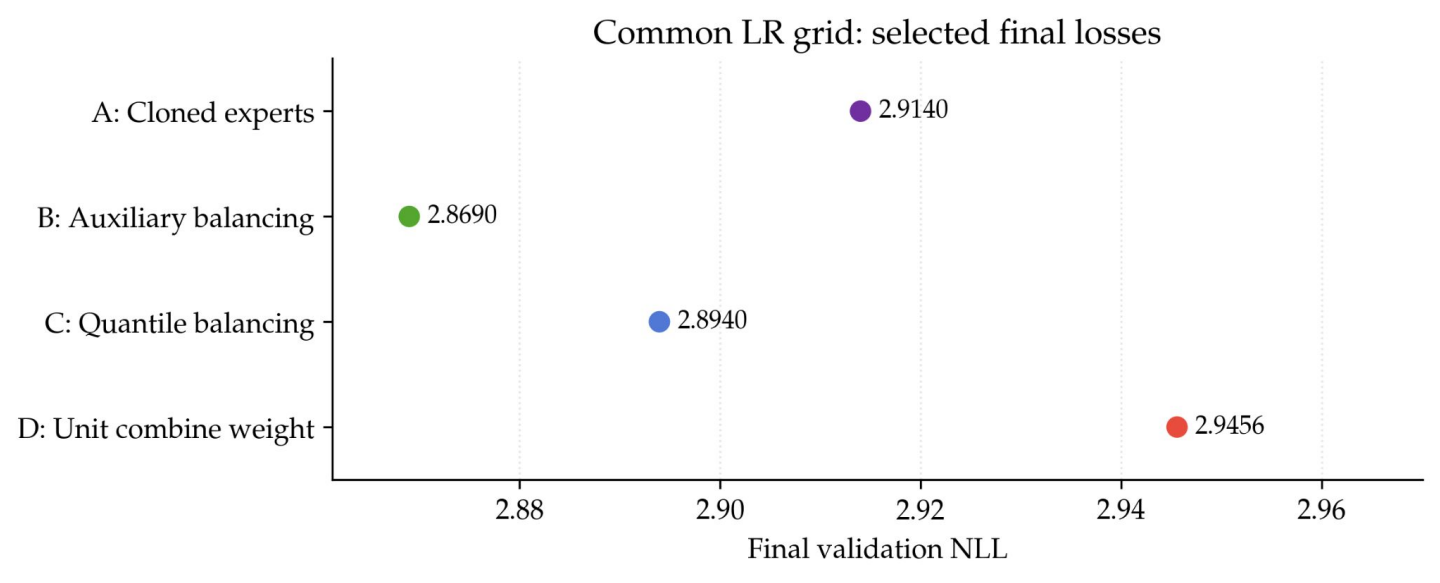
Problem 4: MoE loss comparisons

Part (b): Pick the best router

Package	Intervention
A	Cloned expert initialization
B	Auxiliary load balancing ($\lambda = .01$)
C	Quantile load balancing
D	Remove probability combine weight (use weight 1)

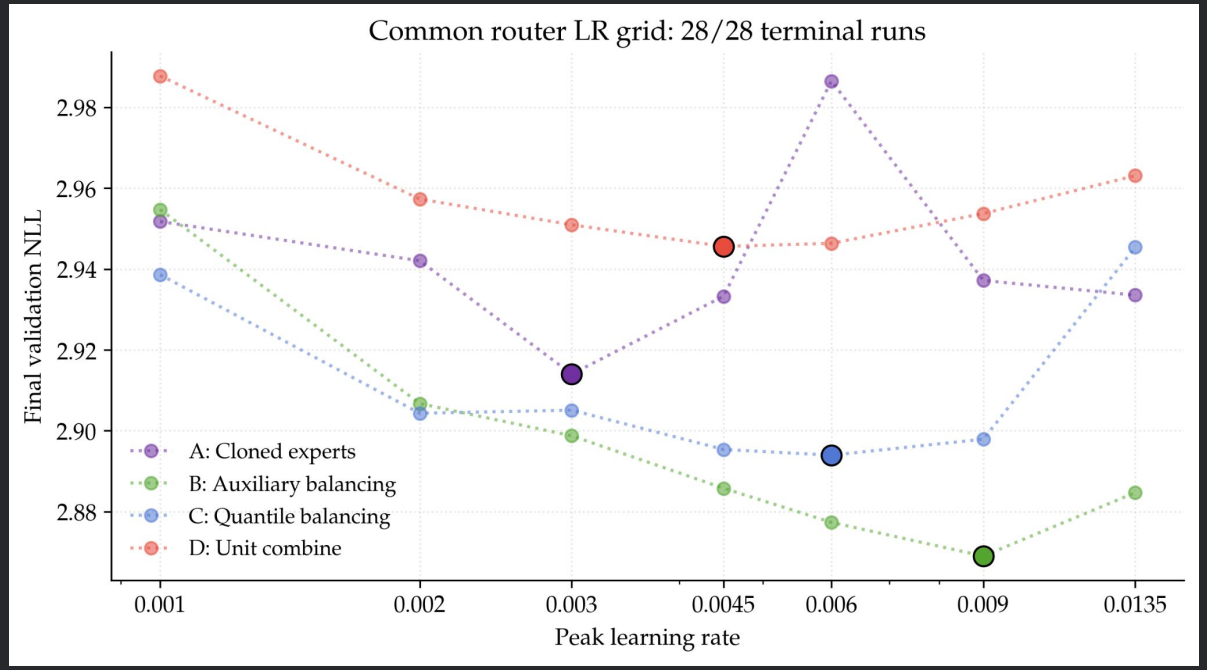
Problem 4: MoE loss comparisons

Answer: Auxiliary balancing does best



Problem 4: MoE loss comparisons

Answer: Visualizing learning rate sweep



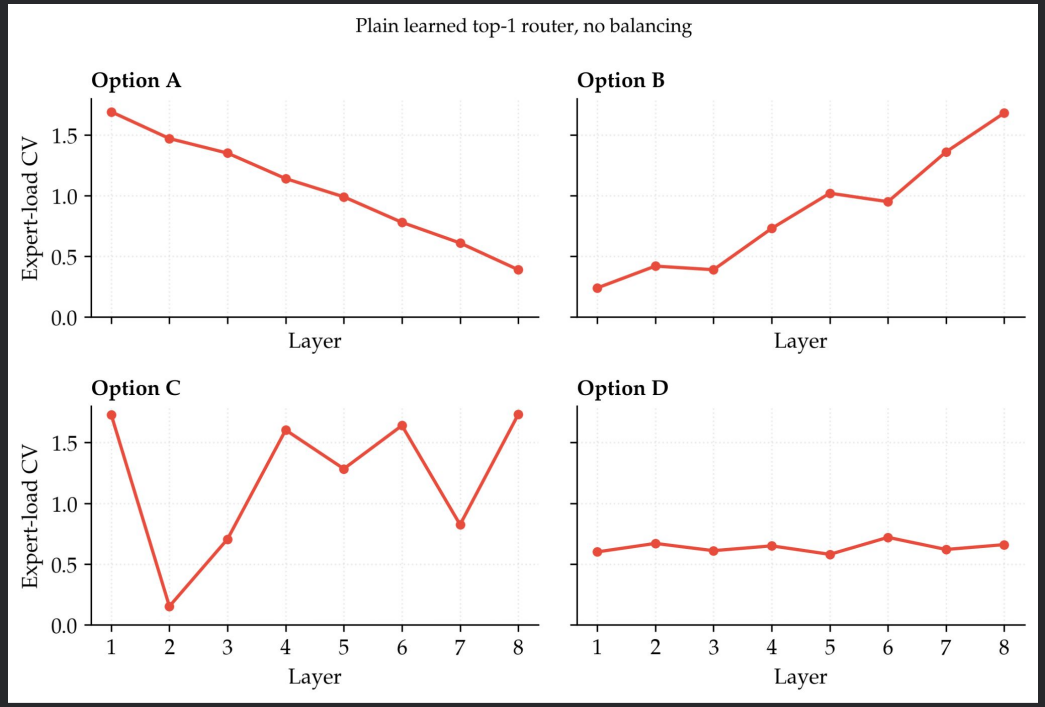
Problem 4: MoE loss comparisons

Reasoning: Quantile balancing more aggressively enforces load balancing relative to aux loss with the benefit of being more stable.

Since aux loss is sufficiently stable at our scale for balancing, we prefer it over quantile balancing. On the other hand, larger scales may opt for the safer choice a the loss detriment (for example, Marin uses QB)

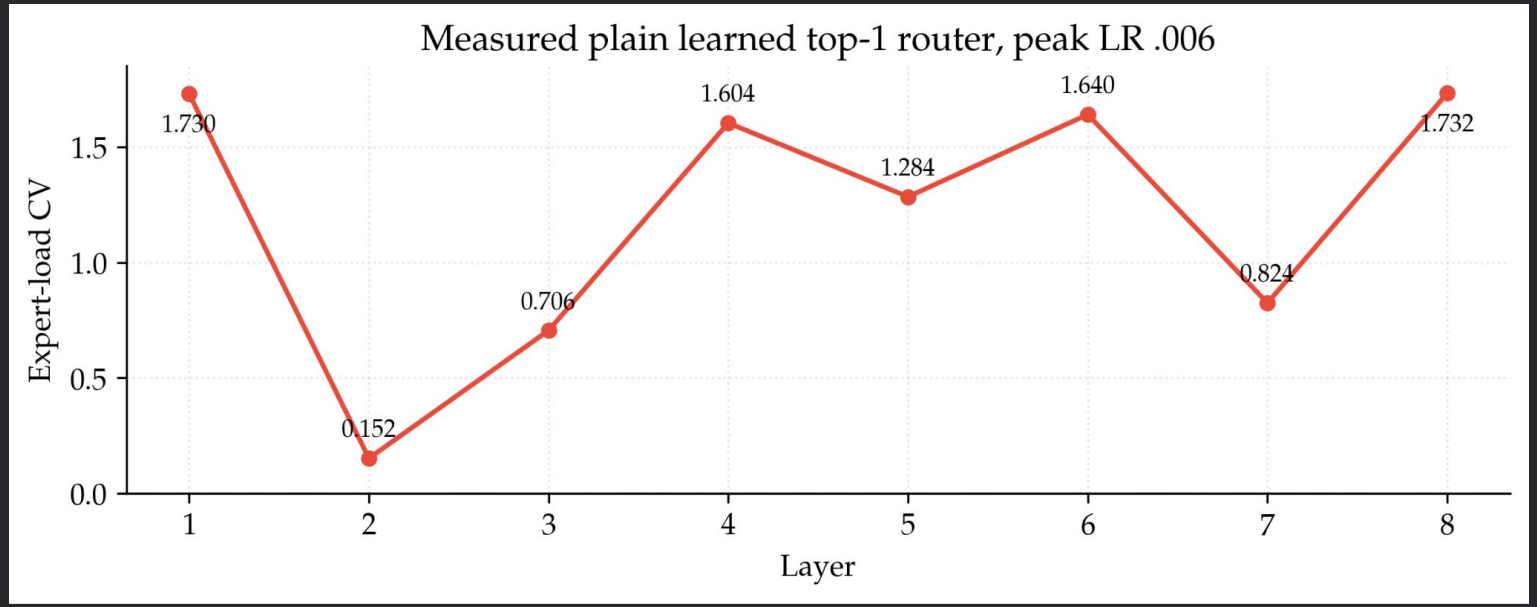
Problem 5: MoE router balance

Part (a): What best matches the plain router's load balance?



Problem 5: MoE router balance

Answer: Option C



Problem 5: MoE router balance

Reasoning: When tracking metrics on the assignment, you might have found that plain learned has quite worse mean and max CV across the layers, especially relative to Aux and Quantile (rules out Option D)

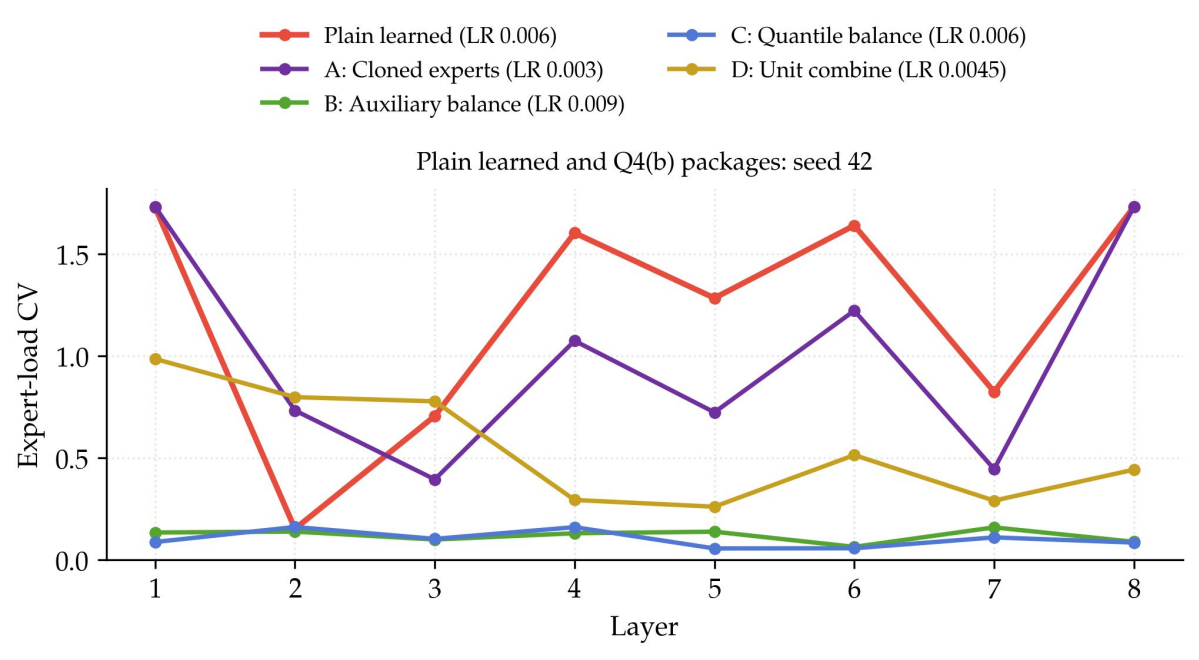
Assignment router metrics: seed 42

Recipe	LR	Val NLL	Mean CV	Max CV	$H/\log 4$	Mean $p_{\max}$
Plain learned	0.006	2.9235	1.209	1.732	0.534	0.717
A: Cloned experts	0.003	2.9140	1.007	1.732	0.645	0.631
B: Auxiliary balance	0.009	2.8690	0.119	0.159	0.602	0.656
C: Quantile balance	0.006	2.8940	0.103	0.161	0.668	0.623
D: Unit combine	0.0045	2.9456	0.545	0.986	0.999	0.265

CV / entropy / confidence: 8 sequences × 128 positions. NLL: full 1000-sequence validation set.

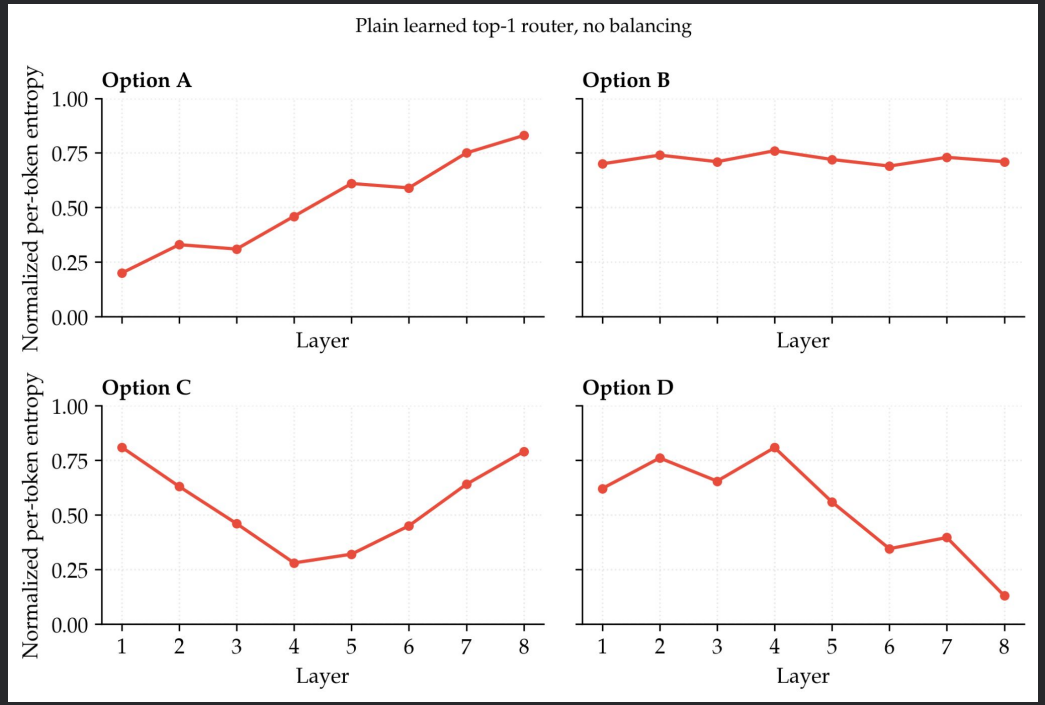
Problem 5: MoE router balance

The balance doesn't have any specific pattern, it's mostly a failure of not optimizing for router loss appropriately



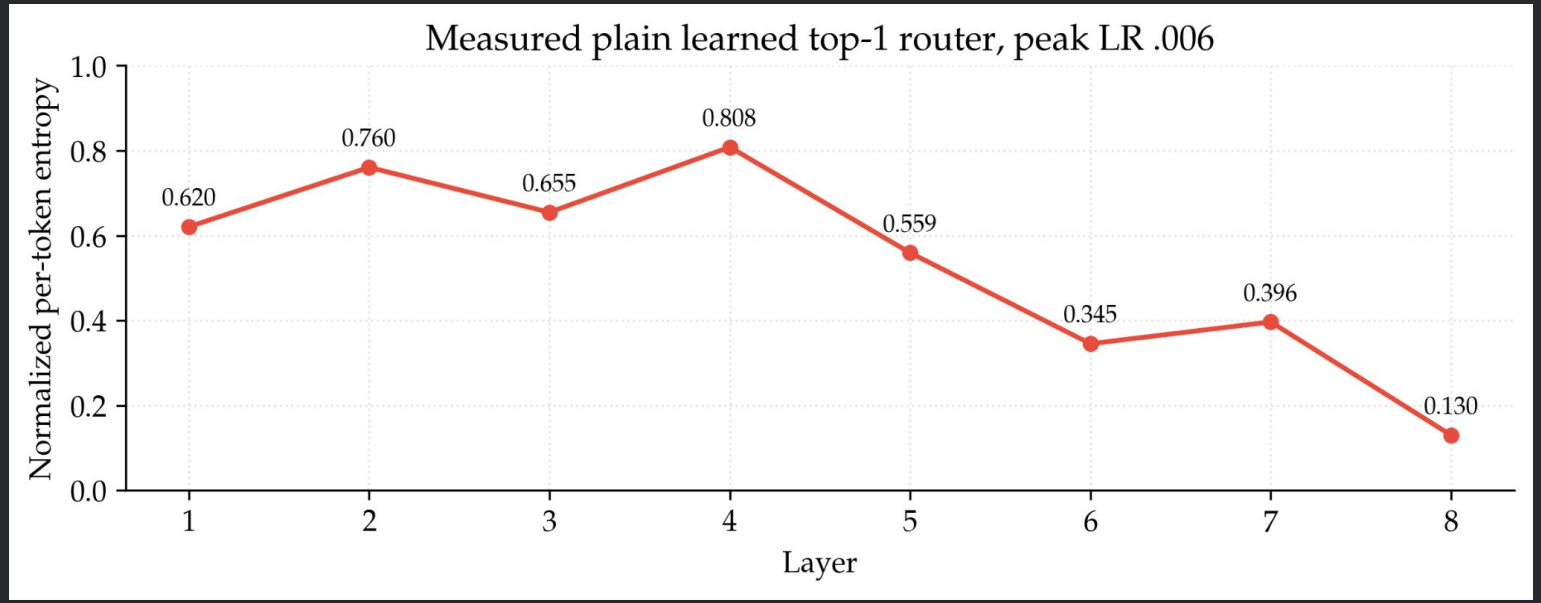
Problem 5: MoE router balance

Part (b): What best matches the plain router's entropy (negative confidence)?



Problem 5: MoE router balance

Answer: Option D

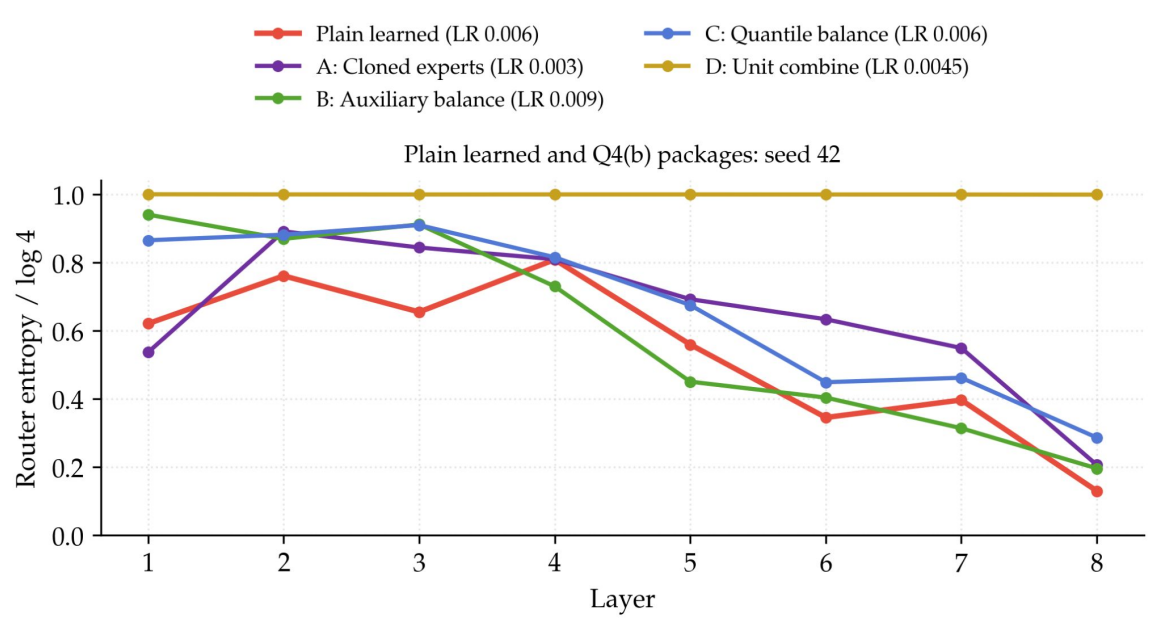


Problem 5: MoE router balance

Reasoning: Later layers have more available information from better embeddings, so they can make more confident predictions about routing. In general, earlier layers have a harder time specializing the experts for this reason.

Problem 5: MoE router balance

The later layers having lower entropy is relatively robust across a couple of different architectures.



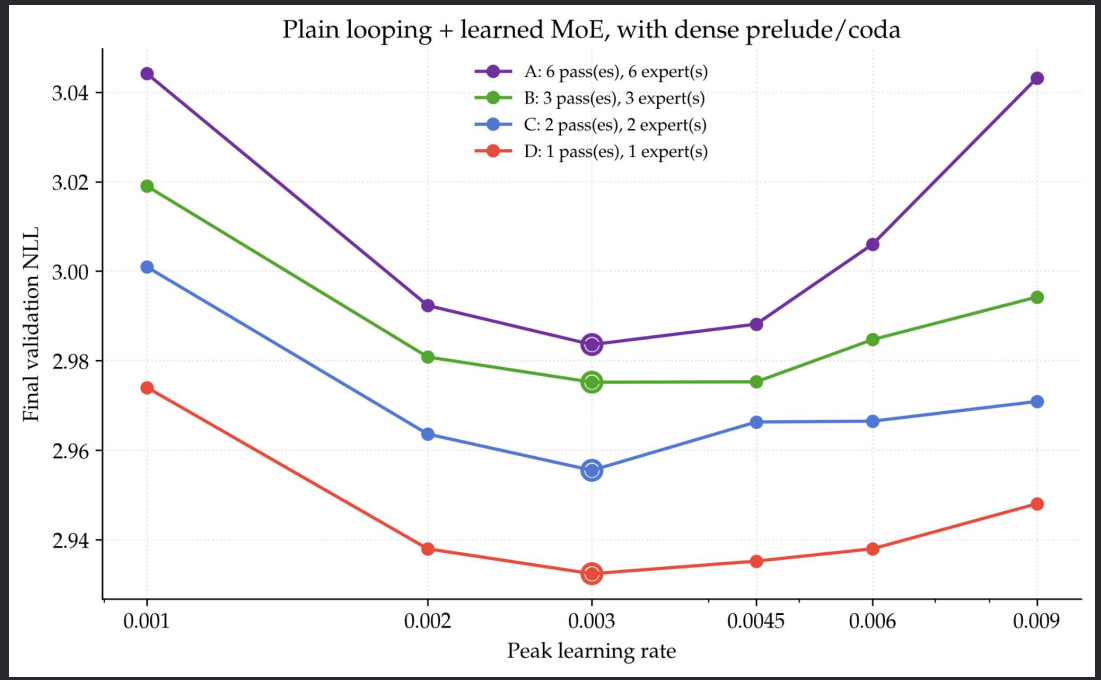
Problem 6: MoE + Looping

Under an inference and unique parameter constraint, should we use dense or MoE + looping?

Package	Passes	Core blocks	Experts per core block	Unique params (M)
A	6	1	6	29.37
B	3	2	3	30.42
C	2	3	2	31.47
D	1	6	1	34.61

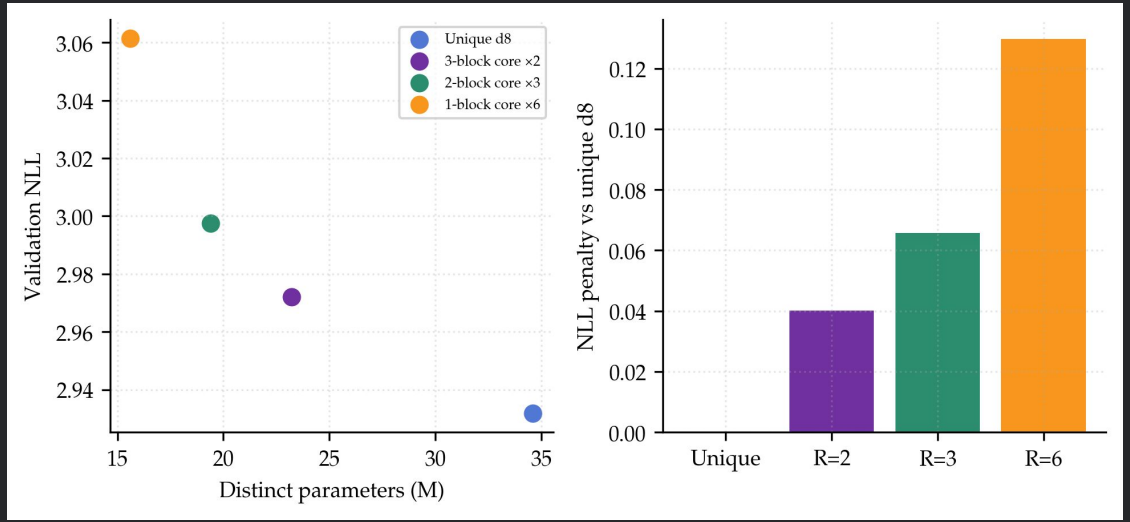
Problem 6: MoE + Looping

Answer: Dense ends up doing best



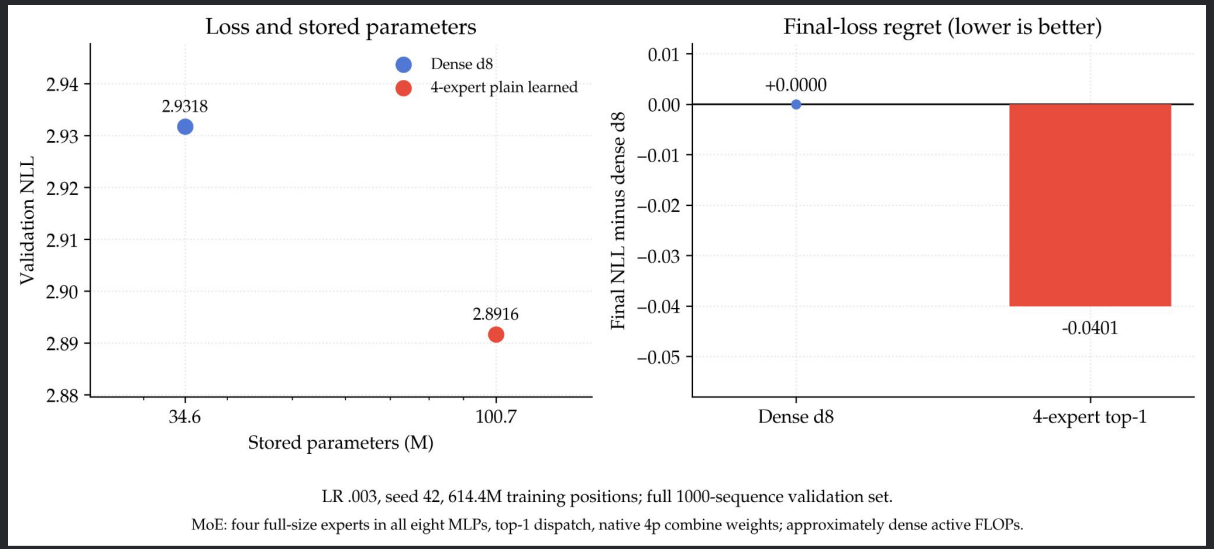
Problem 6: MoE + Looping

Reasoning: We can think about the benefit of MoE and the regret of looping separately and then compose. For looping, we can see that there is roughly 0.02 regret per loop



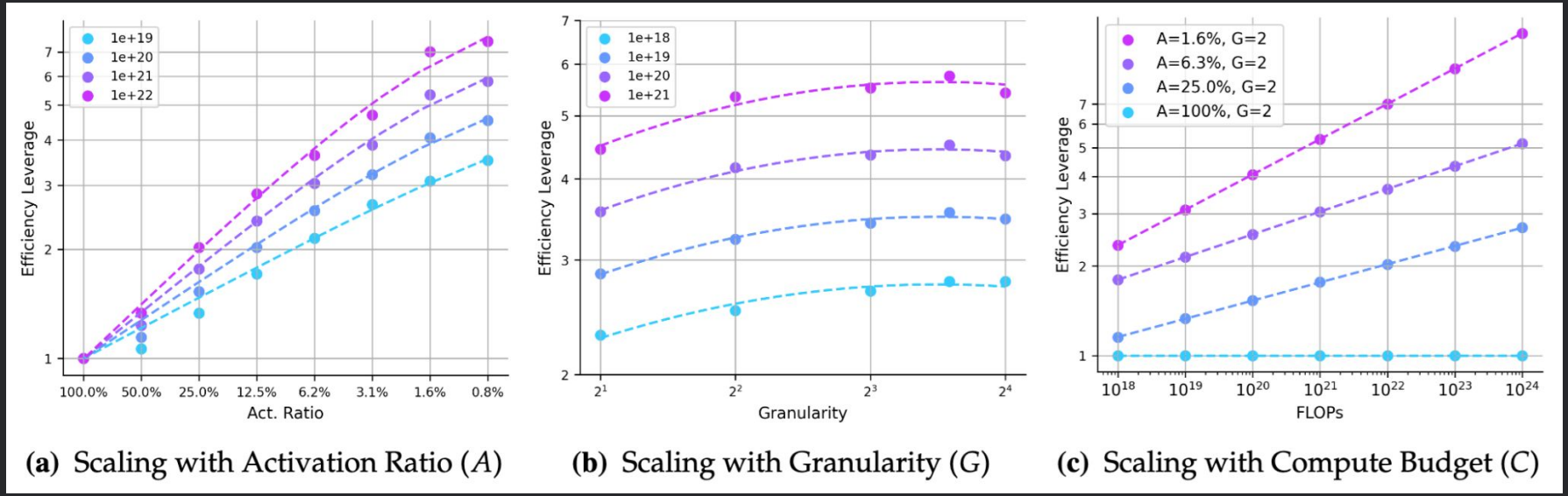
Problem 6: MoE + Looping

Reasoning: From MoE's, we know that plain learned with 4 experts offers a 0.04 benefit on top of dense model. We can roughly estimate every factor of 2 experts helps loss by 0.02



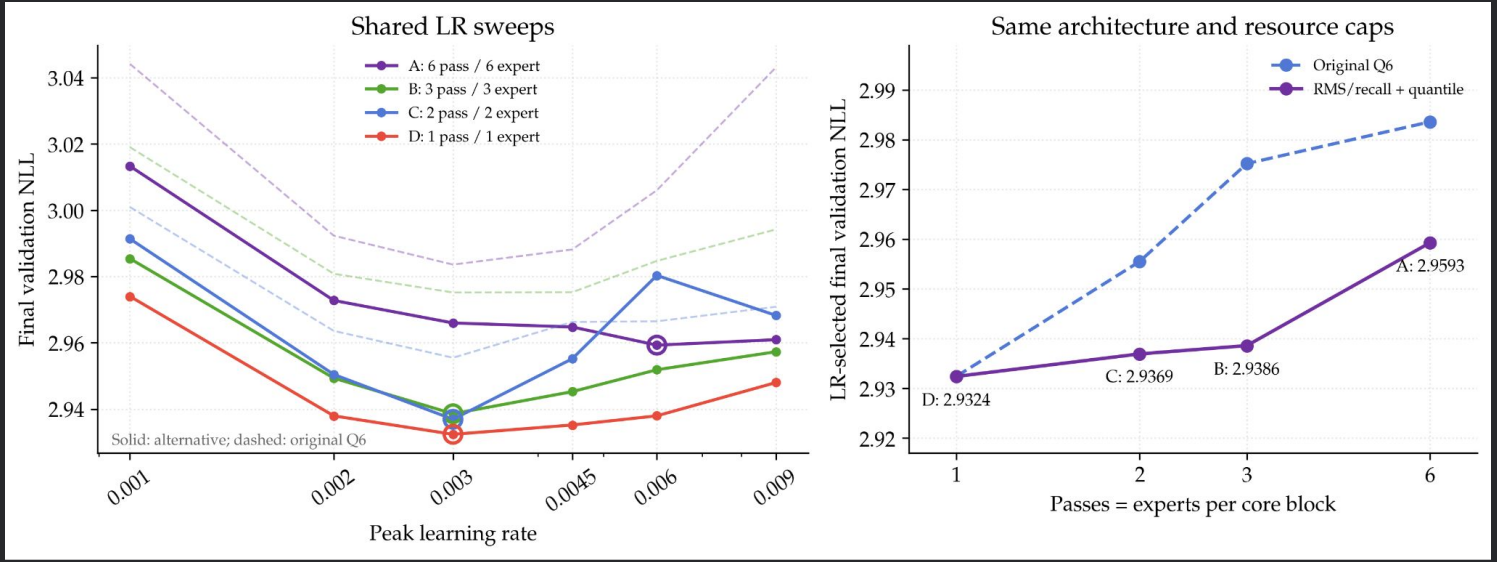
Problem 6: MoE + Looping

Reasoning: This is more rigorously established in [Tian et al, 2025]



Problem 6: MoE + Looping

Reasoning: Composing these facts gives that loops hurt more than MoE's help. Part of this is a consequence of our recipes, if we use better looping and better MoE recipes, the gap exists but is smaller. Better recipes may flip the trend



Announcements

Tonight, we will release Generalization assignment (💜), due on our quiz next Thursday (11/19)

After this, we will have our (final) Exotic assignment, which is currently planned to be RL