Quiz 4 - Stability

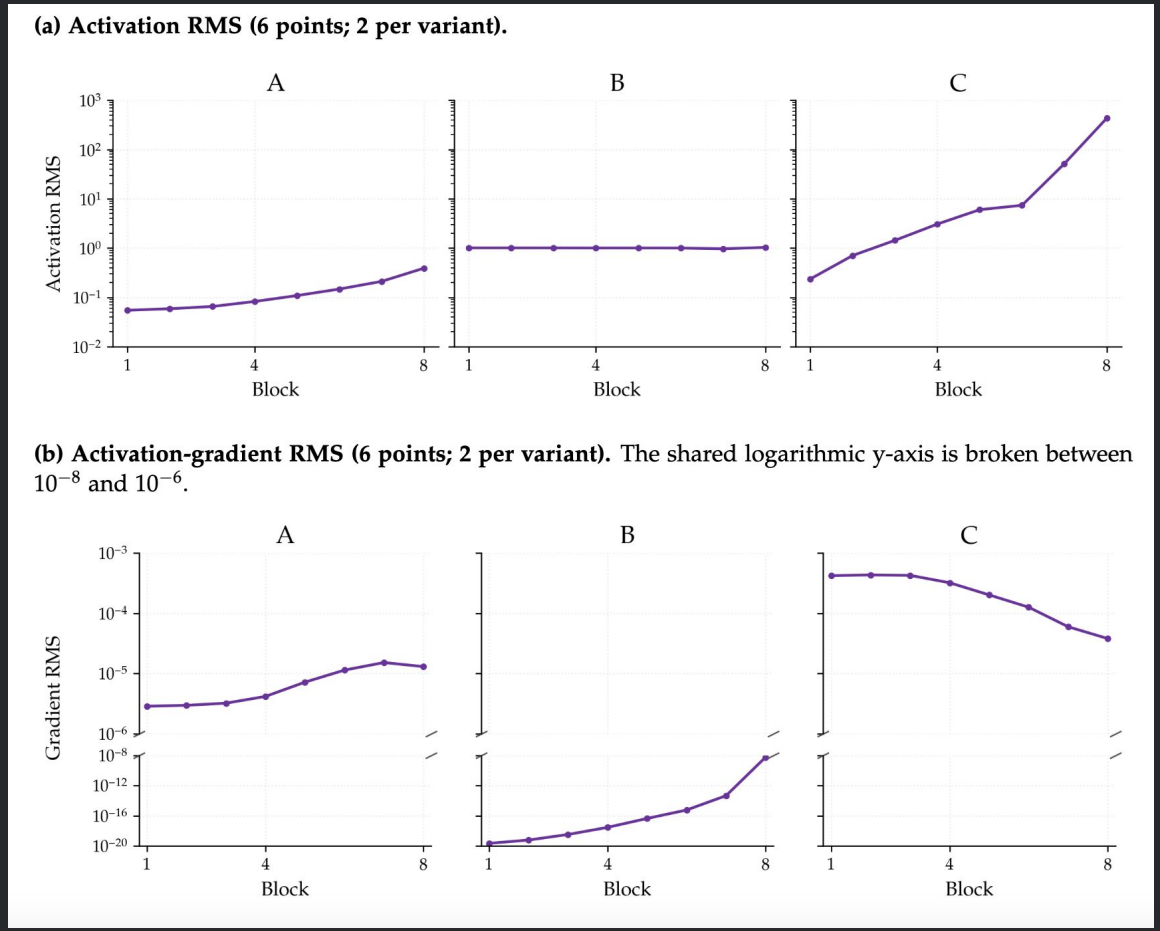
10/29

Problem 1: Norms

Match the methods to the correct norms

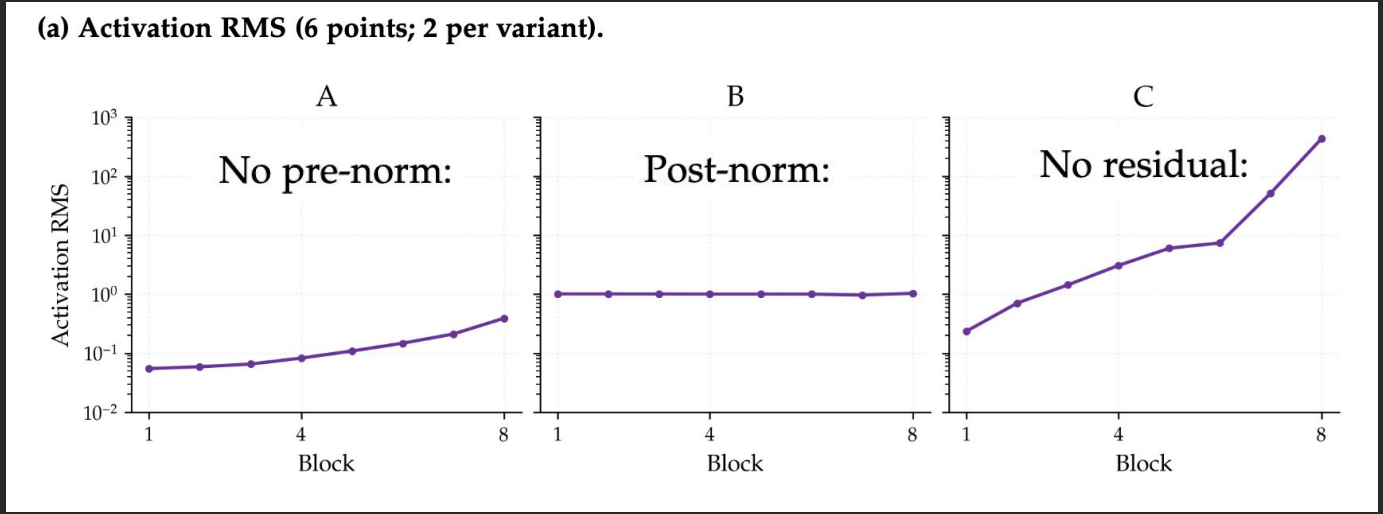
(a) activations (b) gradients

No pre-norm: $x' = x + F(x)$,
Post-norm: $x' = N(x + F(x))$,
No residual: $x' = F(N(x))$.



Problem 1: Norms

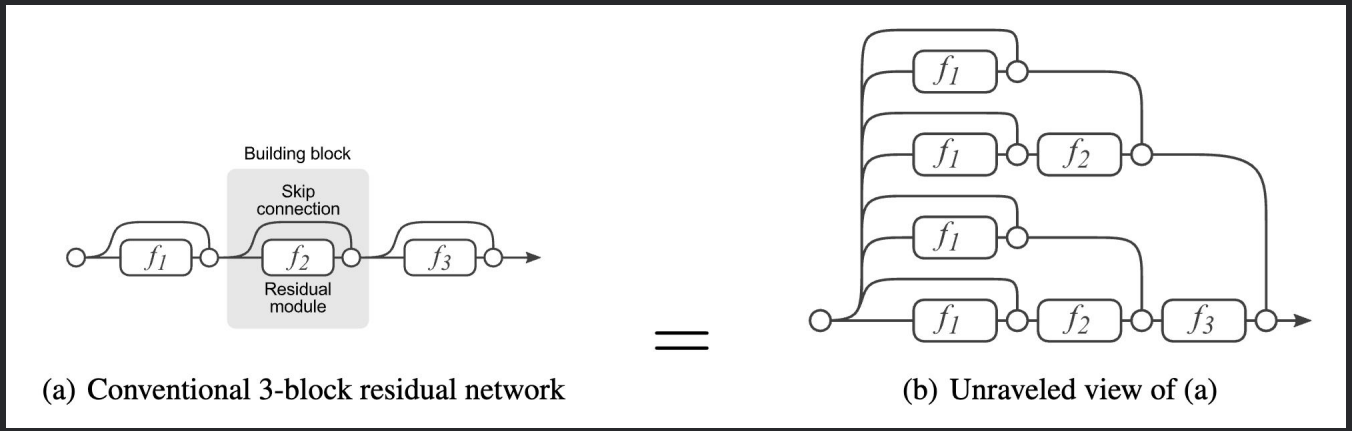
Answer to (a): No pre-norm has norm growing over the layers, post-norm keeps the norms flat, no residual has much faster growth



Problem 1: Norms

Reasoning: Post-norm will definitionally keep norm flat (RMS 1). Without residuals, the norm growth is a lot more uncontrolled.

One intuition for pre-norm's larger activations is that later layers naturally have exponentially less contribution. Therefore, later layers need larger magnitude to compensate and contribute.

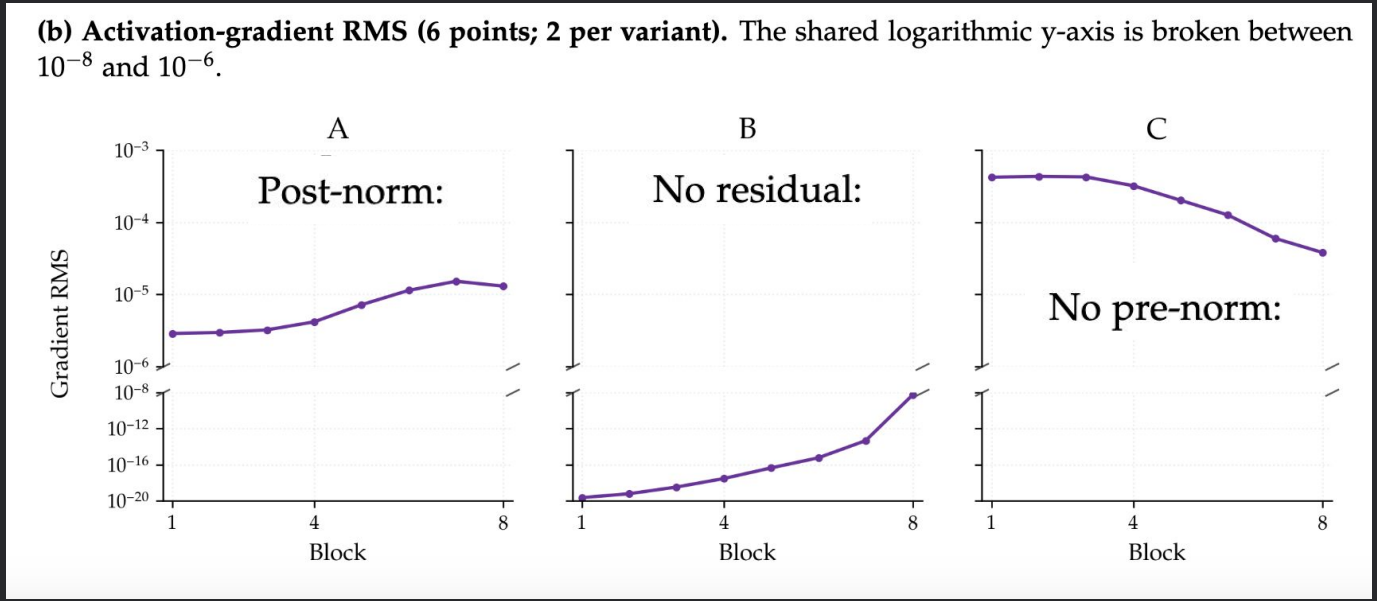


(a) Conventional 3-block residual network

(b) Unraveled view of (a)

Problem 1: Norms

Answer to (b) Post-norm has smaller gradients for earlier layers, no residuals has extra small gradients, and no pre-norm has larger gradients for earlier layers



Problem 1: Norms

Reasoning: Under the unraveled view, its natural that earlier layers would have larger gradients, explaining no pre-norm. No residuals has extremely poor vanishing gradients.

Post-norm has smaller gradients for earlier layers since every time we apply the norm, we rescale all gradients for earlier layers. This is unlike pre-norm, where there exist pathways to the earlier gradient that are unscaled.

Problem 2: RNNs vs Transformers, depth

Part (a): Does the stream grow more for RNN, Transformer, or neither?

(a) Stream growth (4 points). Use the released default LRs: .003 for the transformer and .001 for the RNN, without tuning. Let x_ℓ be the stream after complete block ℓ, including its MLP and depth skips. On the same four held-out sequences, measure

$$R = \frac{\mathrm{RMS}(x_4)}{\mathrm{RMS}(x_1)} \qquad \text{for both models.}$$

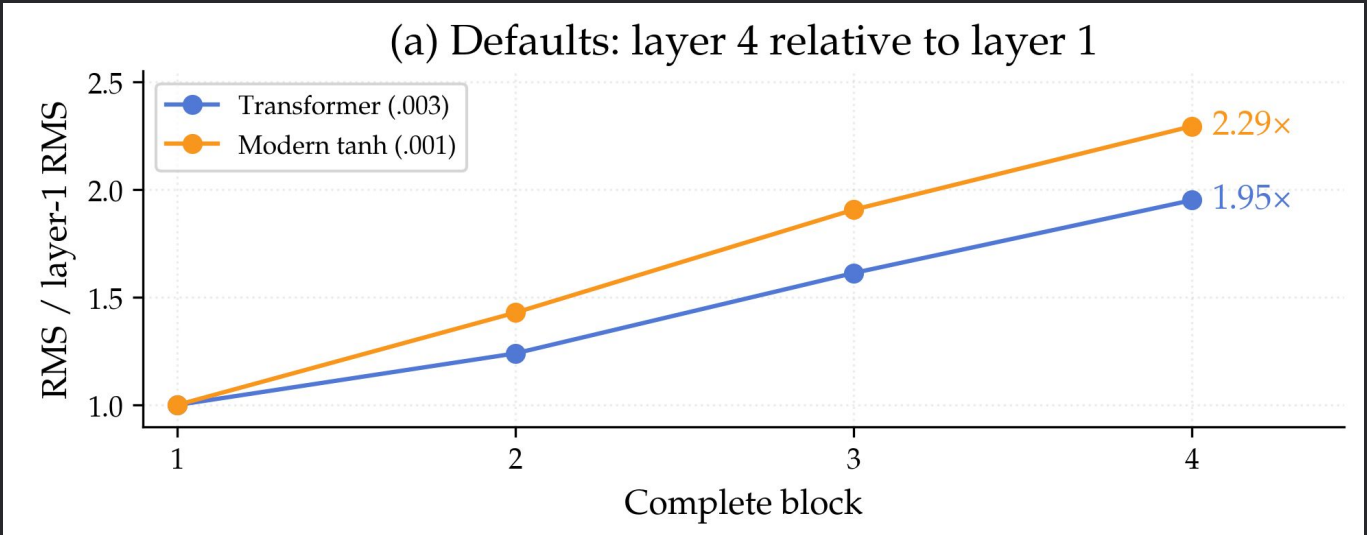
RMS averages over batch, time, and features. Which model has the larger R from layer 1 to layer 4? Choose **Within 50%** if $\max(R_\mathrm{T}, R_\mathrm{R}) / \min(R_\mathrm{T}, R_\mathrm{R}) \leq 1.50$; otherwise choose the larger ratio.

○ Transformer ○ RNN ○ Within 50%

Problem 2: RNNs vs Transformers, depth

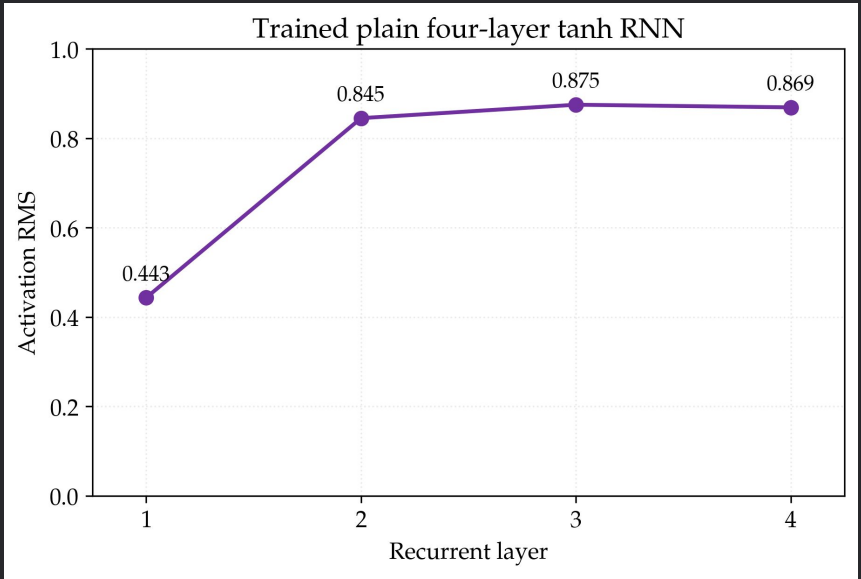
Answer to (a): The stream norm growth is comparable for both

Implication: the phenomena of activations growing over depth is not specific to attention, maybe later layers simply want to contribute more



Problem 2: RNNs vs Transformers, depth

One interesting ablation is measuring the same quantity for a plain RNN (no MLP's, residuals, prenorm). Interestingly, after layer 2, there is no norm growth. This may impact why our plain 4 layer RNN's were not training well.



Problem 2: RNNs vs Transformers, depth

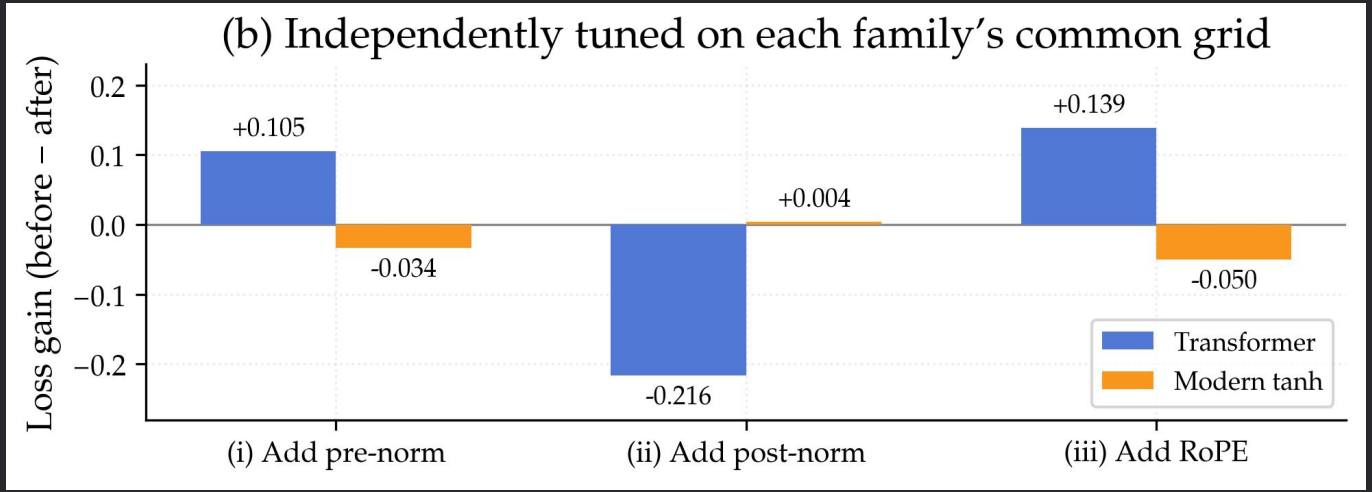
Part (b): Does the intervention helps transformers more, RNNs more, or neither?

	Before	After
(i)	No pre-norm: $x + F(x)$	Pre-norm: $x + F(N(x))$
(ii)	Pre-norm: $x + F(N(x))$	Post-norm: $N(x + F(x))$
(iii)	No RoPE	RoPE

For each family, compute $I = L^\star_{\text{before}} - L^\star_{\text{after}}$ for $L^\star$ as LR-tuned final validation loss. Choose **Transformer** if $I_{\text{T}} - I_{\text{R}} \geq .02$ (the transformer benefits more from the intervention), **RNN** if $I_{\text{R}} - I_{\text{T}} \geq .02$ (the RNN benefits more from the intervention), and **Similar** if $|I_{\text{T}} - I_{\text{R}}| < .02$ (both have similar benefits).

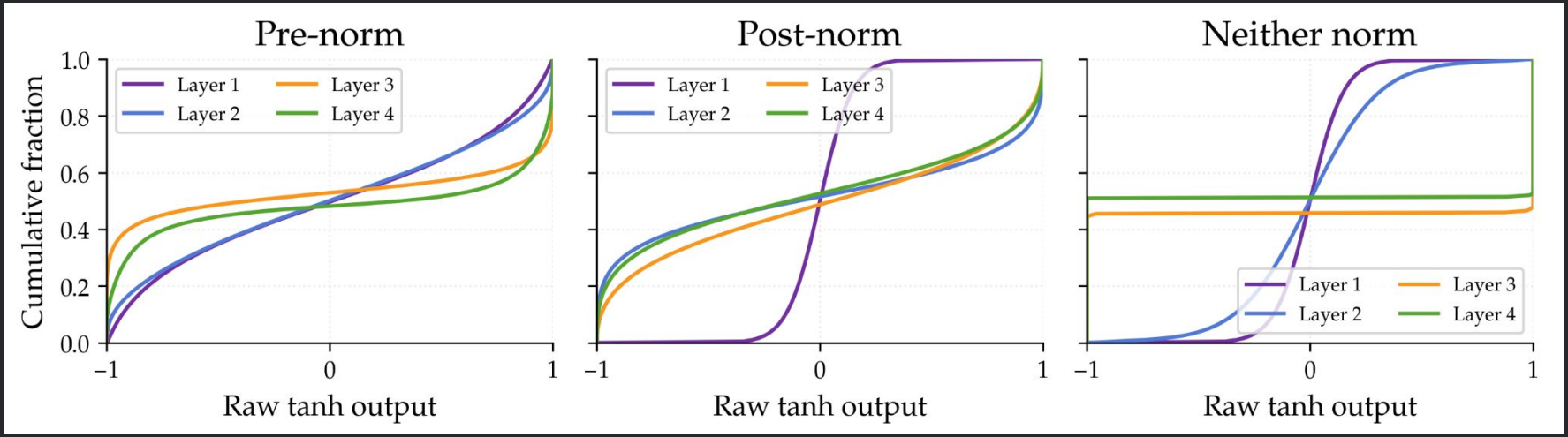
Problem 2: RNNs vs Transformers, depth

Answer to (b): Pre-norm helps transformer more, post-norm hurts transformer more (so RNN is better), RoPE helps transformer more



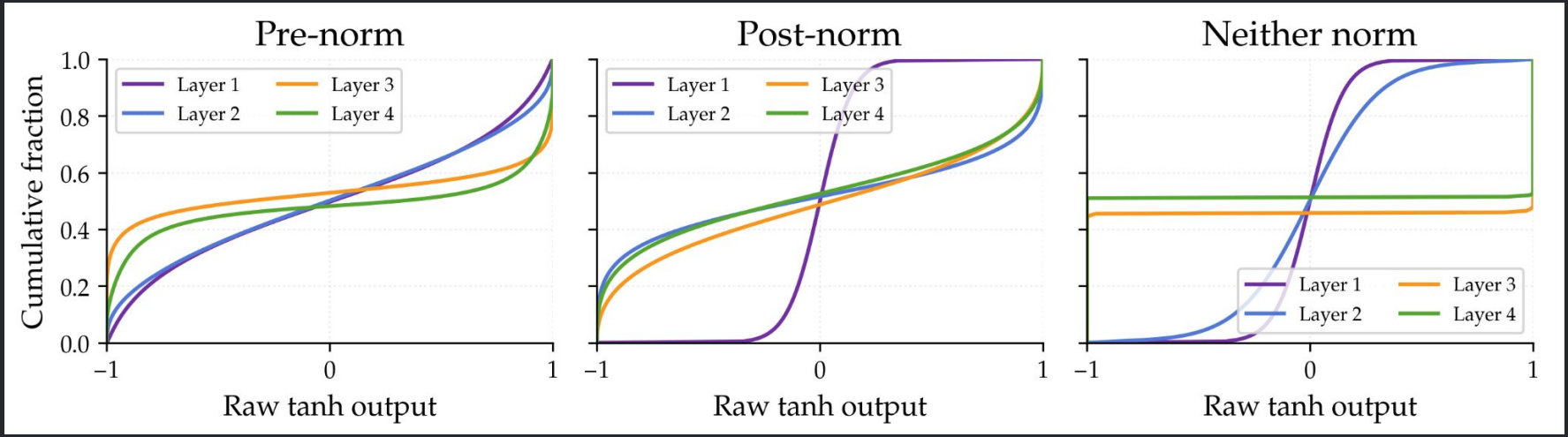
Problem 2: RNNs vs Transformers, depth

Reasoning for (i) and (ii): The RNN has natural norm control due to its tanh components (which clamp output between -1, 1). Visualized below as the empirical quantiles of the tanh components. Therefore, norms have little effect on training stability or loss. Nonetheless, either norm reduces tanh saturation



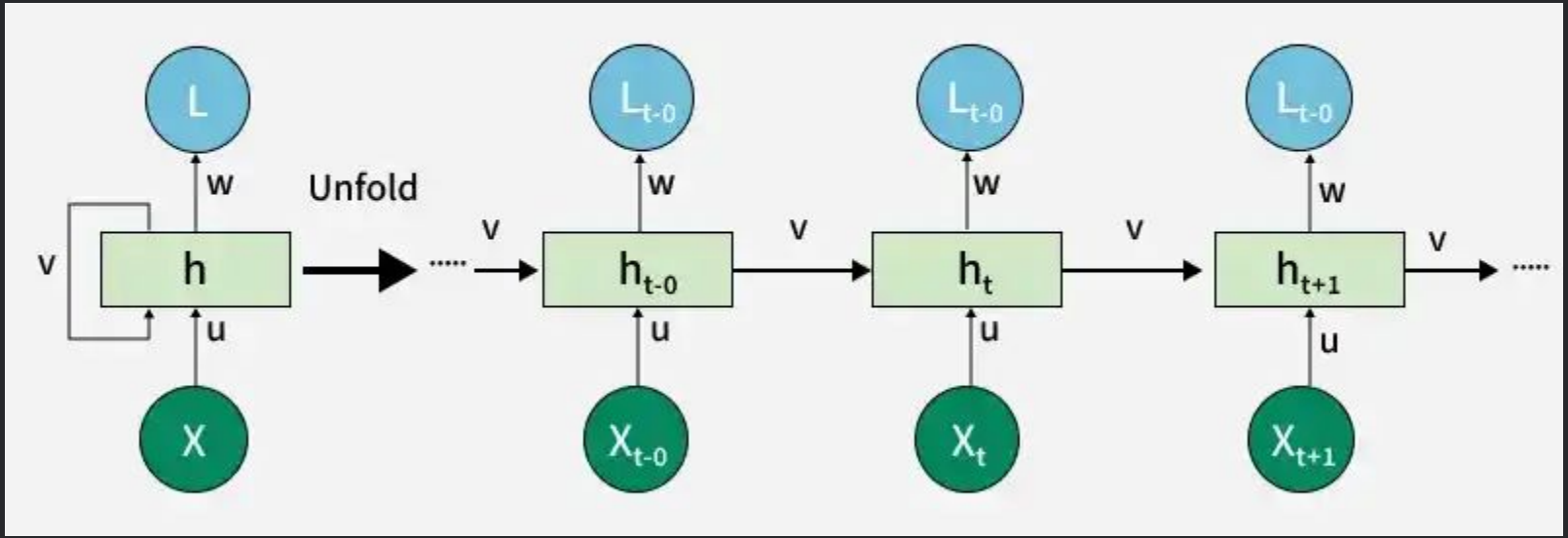
Problem 2: RNNs vs Transformers, depth

Reasoning for (i) and (ii): From the assignment, you know that pre-norm helps transformers and post-norm hurts transformers. This yields the final answer



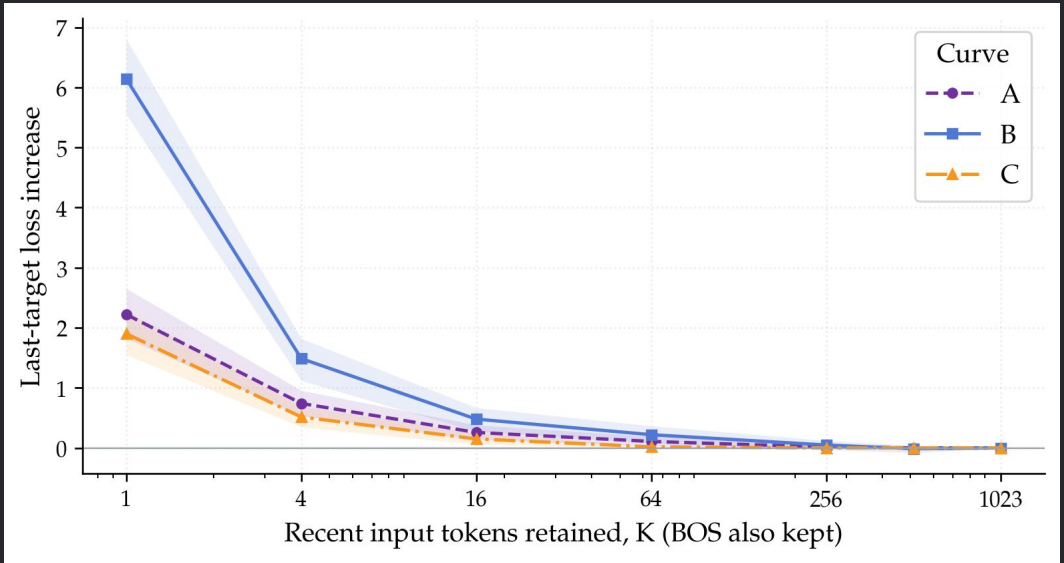
Problem 2: RNNs vs Transformers, depth

Reasoning for (iii): RNN's already have temporal structure so they do not need positional encodings. These are for more important for a transformer.



Problem 3: RNNs vs Transformers, time

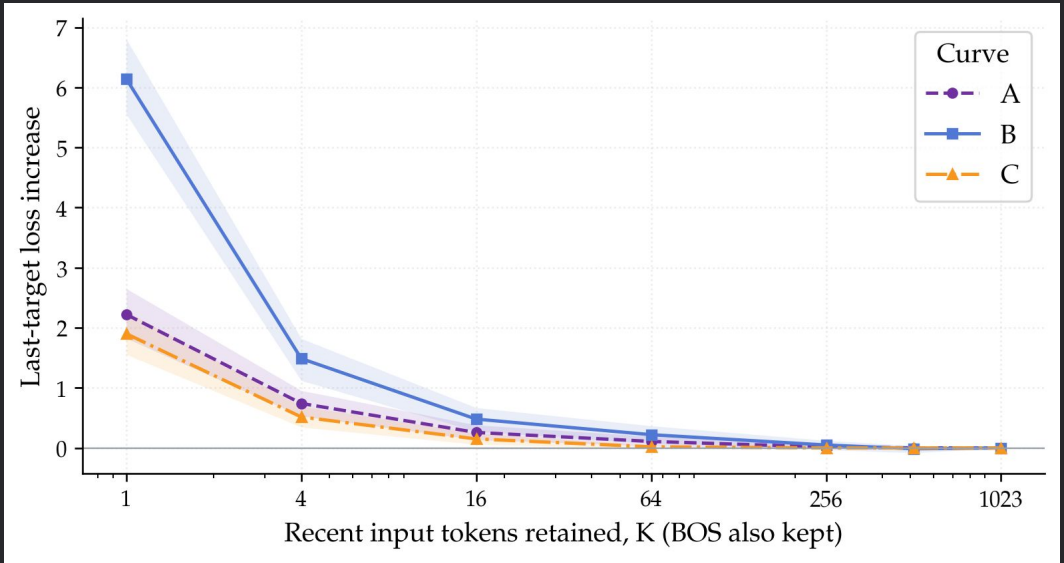
Part (a): Which algorithm corresponds to each loss profile after context removal?



Problem 3: RNNs vs Transformers, time

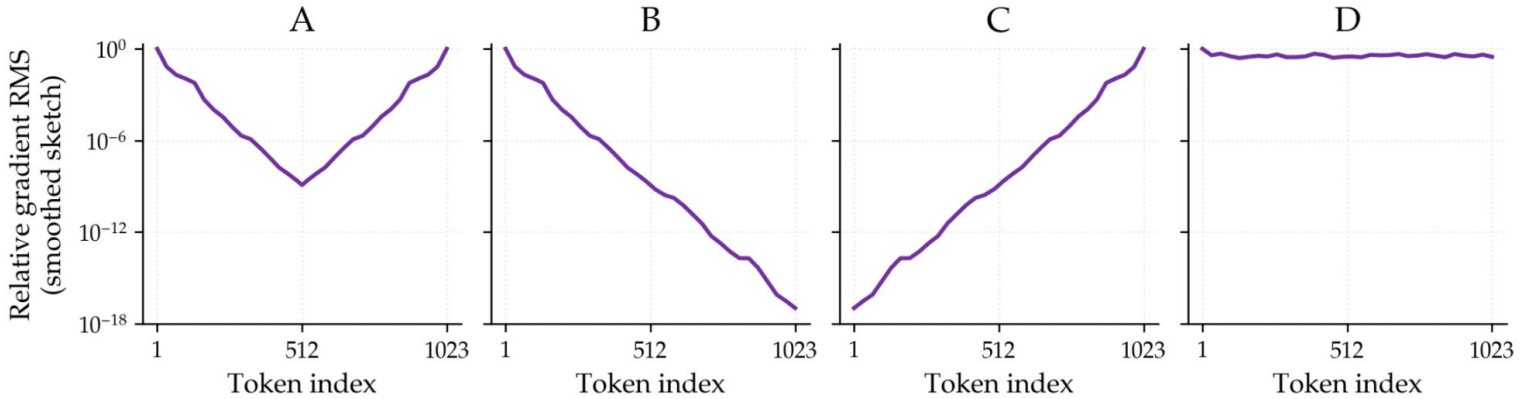
Answer to (a): B is transformer, A is GRU, C is RNN

Reasoning: Transformers actually leverage their context so they would be hurt the most by removing tokens. RNN's are horrible at leveraging context, GRU slightly



Problem 3: RNNs vs Transformers, time

Part (b): Which plot is the gradient of the last token loss with respect to the ith hidden state? What about full sequence loss?

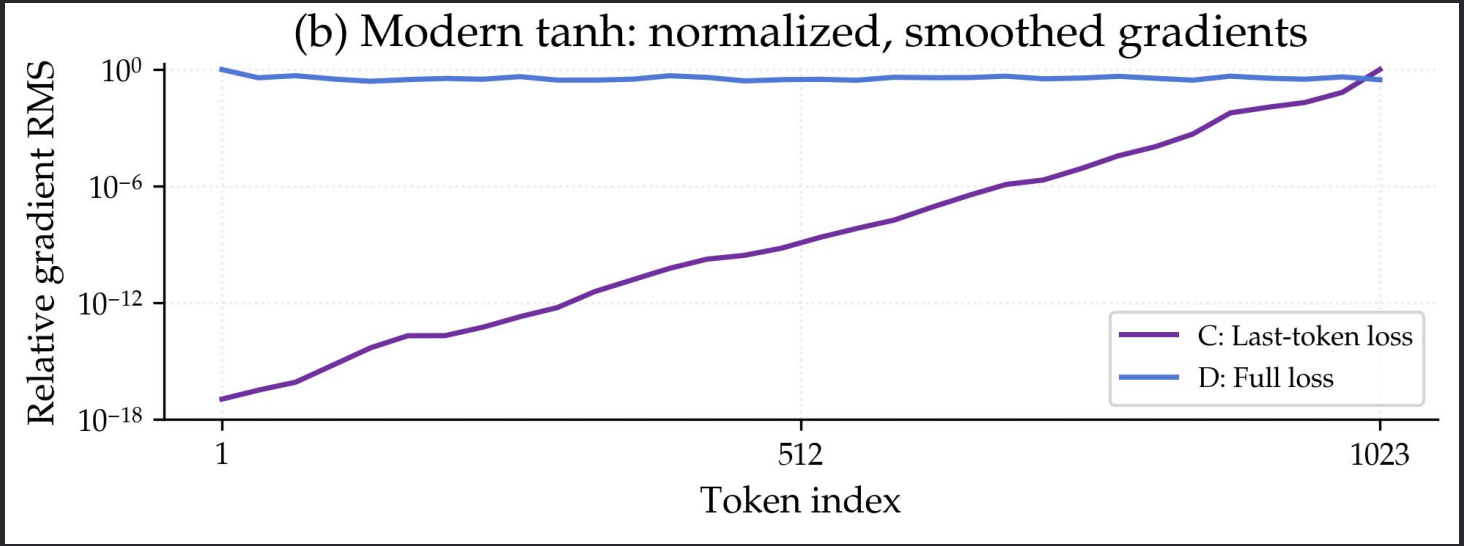


Last-token loss gradient: $L = L_T$ ○ A ○ B ○ C ○ D

Full-loss gradient: $L = T^{-1} \sum_{s=1}^{T} L_s$ ○ A ○ B ○ C ○ D

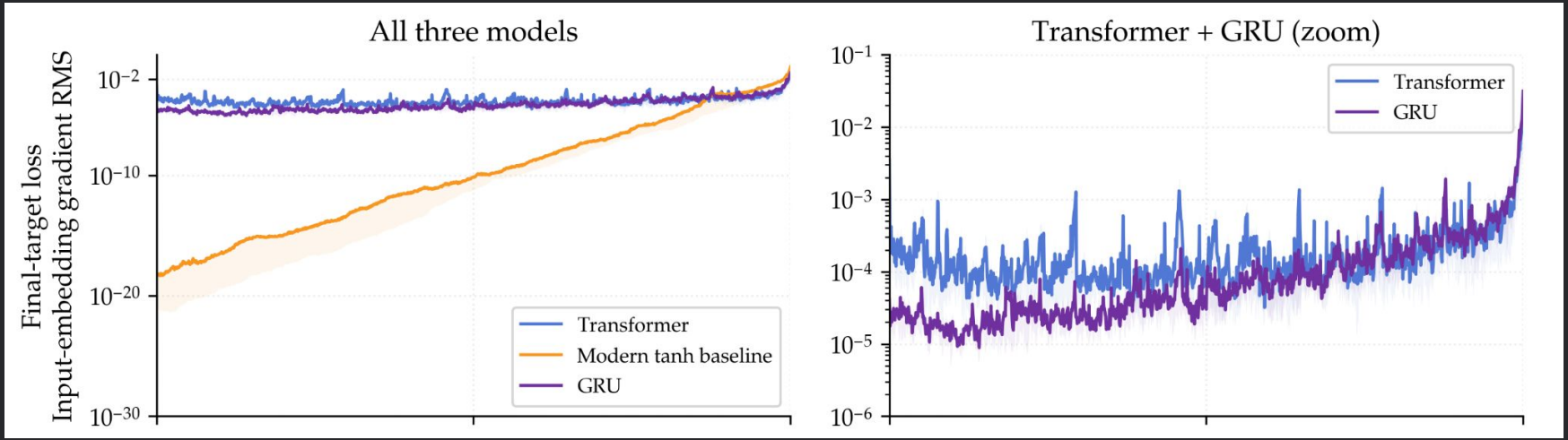
Problem 3: RNNs vs Transformers, time

Answer to (b): The last token loss gradient is exponentially decreasing in distance from last token (C) and the full sequence loss gradient is flat (D).



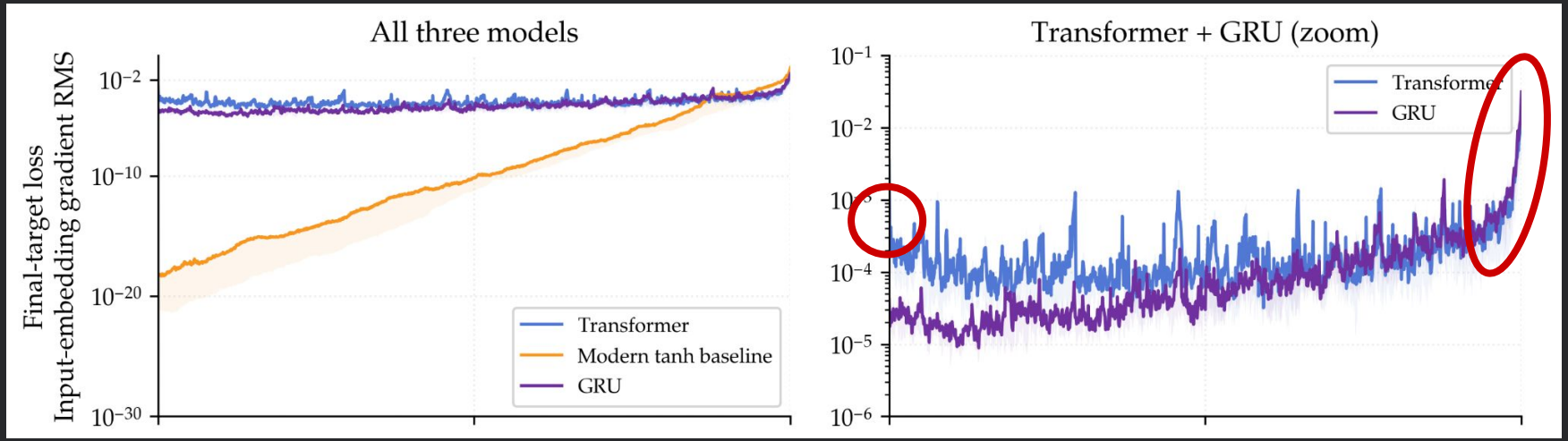
Problem 3: RNNs vs Transformers, time

Reasoning for final token: RNN's have horrible dependence across time and earlier tokens have little influence on later tokens. Therefore, the gradient is quite poor. This is much better for GRU, but it's still far less even than transformer



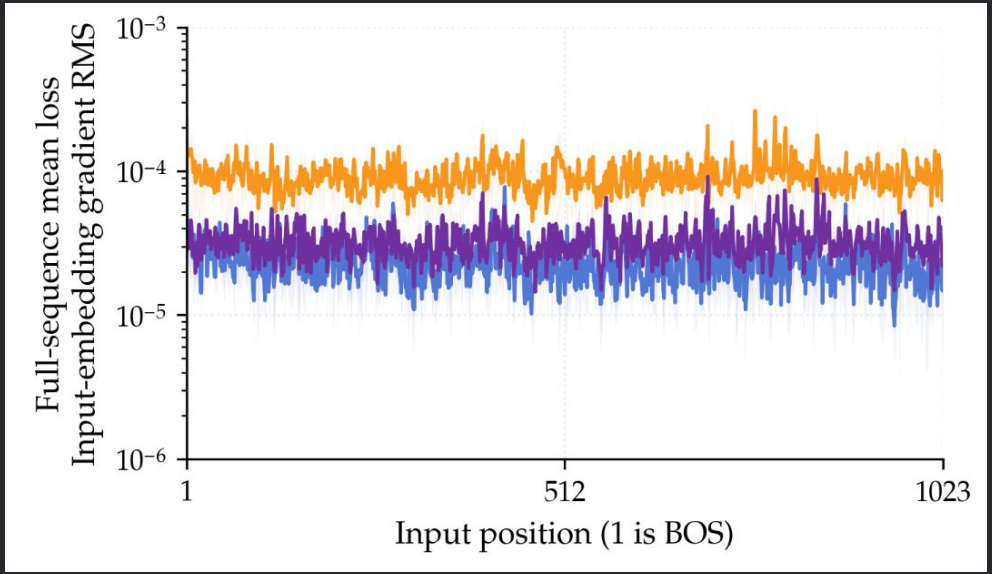
Problem 3: RNNs vs Transformers, time

Aside: This diagram also reveals that transformers have extra sensitivity to the most recent tokens. It also reveals attention sinks by squinting at the uptick at index 0 (not clearly visualized, but ~10x bigger gradient for first token)



Problem 3: RNNs vs Transformers, time

Reasoning for sequence: Since we now get to average over all the tokens, the final gradient will depend equally on all of the tokens. This is straightforward if the loss only depended on the previous token, though it's true more generally



Problem 4: Pinning residual stream norms

Can we pin the norms of the residual stream after training for a bit and be okay?

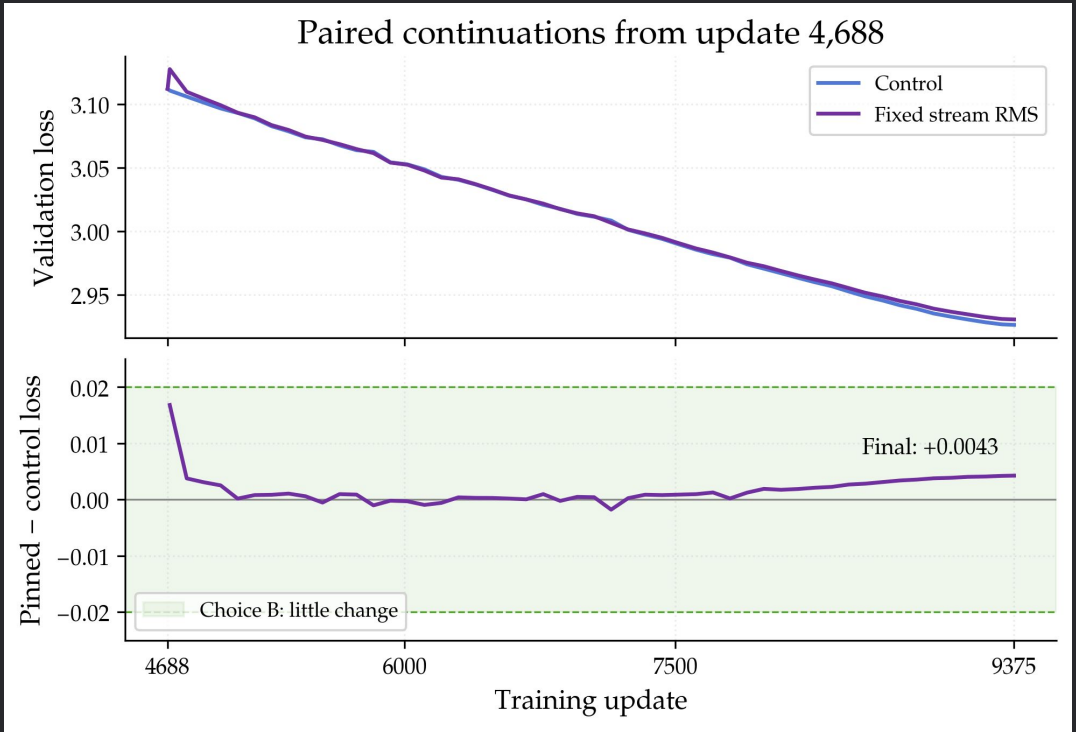
Pin stream norms halfway through training. At update 4,688 of the LR .003 baseline, fork two continuations. One continues unchanged. In the other, after each attention/MLP residual addition y, apply

$$y \leftarrow s_j \, y / \operatorname{RMS}(y).$$

where s_j is the mean tokenwise RMS on 16 validation sequences at the fork. Compare this constraint with the unchanged continuation at update 9,375. How much does the loss change after this modification? Choose the final loss-regret interval $\Delta = L_{\text{changed}} - L_{\text{control}}$.

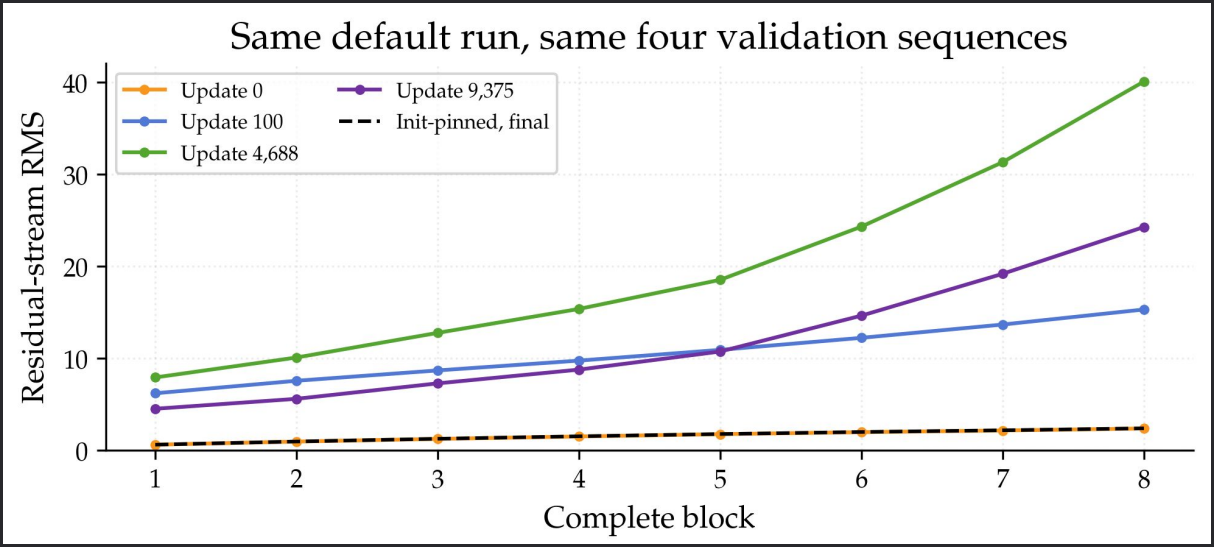
Problem 4: Pinning residual stream norms

Answer: The loss gap is 0.0043, so effectively no impact from pinning (choice B)



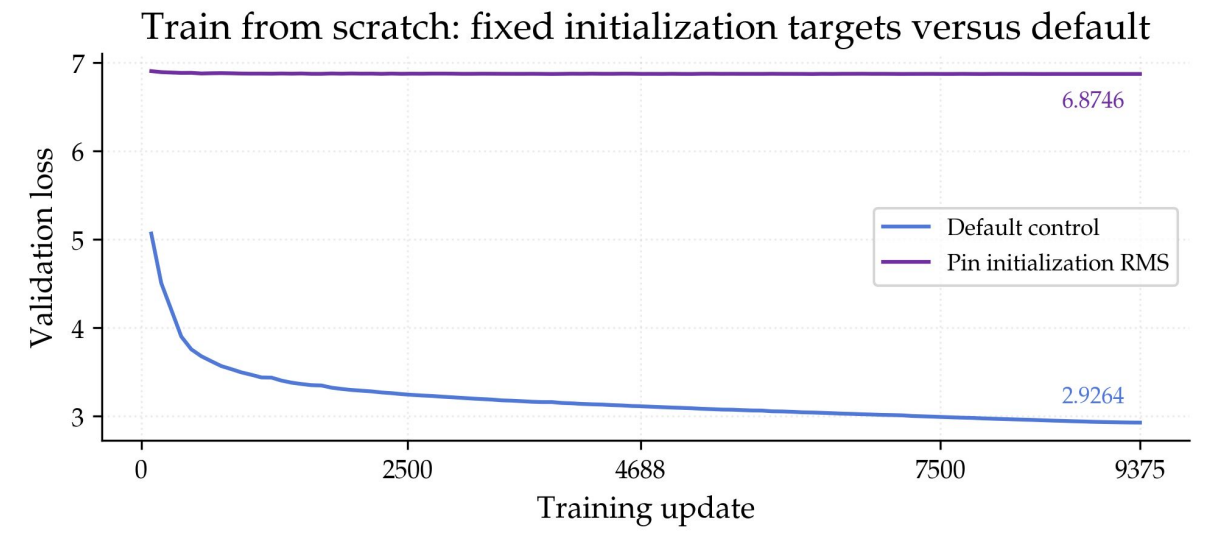
Problem 4: Pinning residual stream norms

Reasoning: After a certain amount into training, the norms are still changing, but the shape is somewhat stable. Perhaps it's fine to pin the norms because the model retains the same expressivity? Earlier plot had larger regret for more time



Problem 4: Pinning residual stream norms

Aside: If you had instead pinned the activation norms to the values at initialization, the model does not train at all (it gets the same loss as no residuals). Therefore, important to do this part way through training



Problem 5: Residual design

Part (a): How much does loss change if we only have residuals for a block rather than residuals per module?

(a) Keep only a whole-block skip (10 points). The default block has an attention skip and an MLP skip:

$$u = x + A(N_1(x)), \qquad y = u + M(N_2(u)),$$

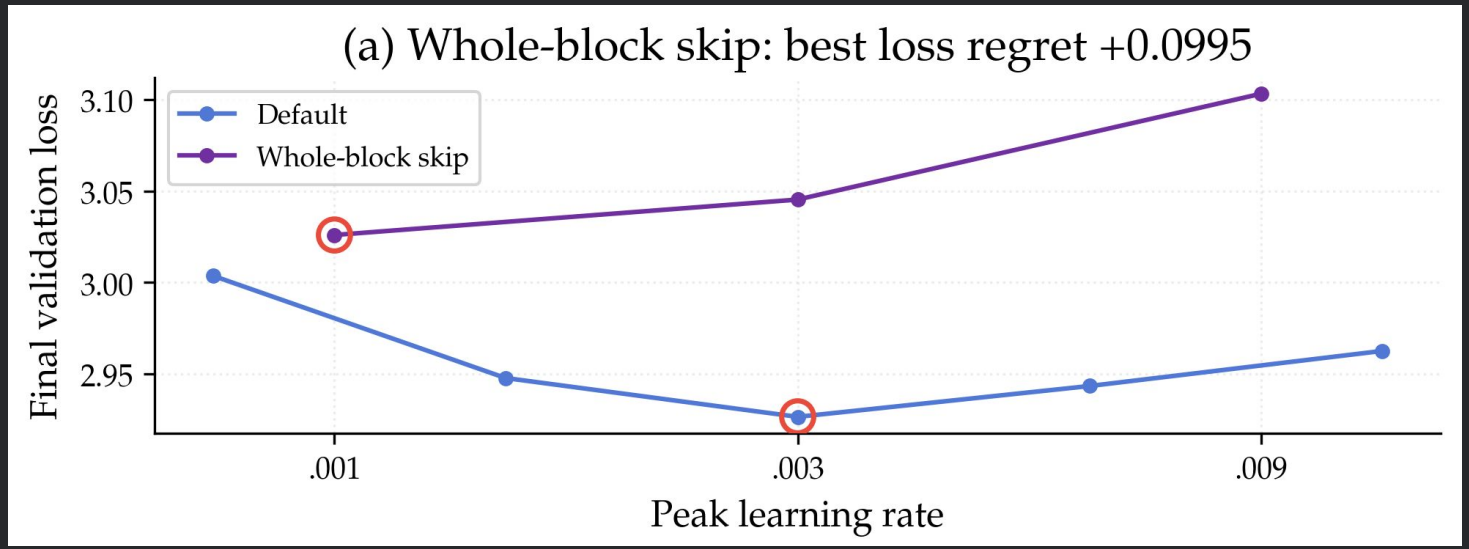
where A is attention, M is the MLP, and N_1, N_2 are RMSNorms. Replace this rule by

$$y = x + M(N_2(A(N_1(x)))),$$

Problem 5: Residual design

Answer to (a): We incur ~0.1 loss regret (Choice D)

Reasoning: It is interesting this trains at all. The blocks are only two layers, so maybe it is not catastrophic to train each two layer subnetwork without residuals.



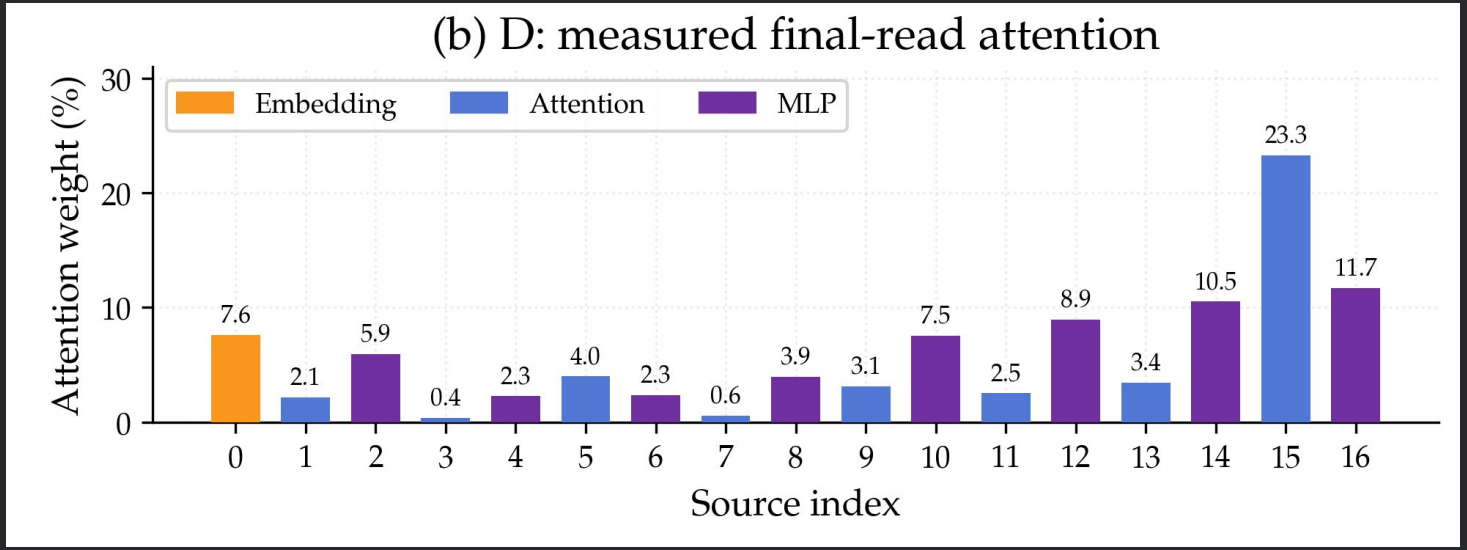
Problem 5: Residual design

Part (b): Which best matches the attention residuals attention pattern for the final layer attending to all previous layers?



Problem 5: Residual design

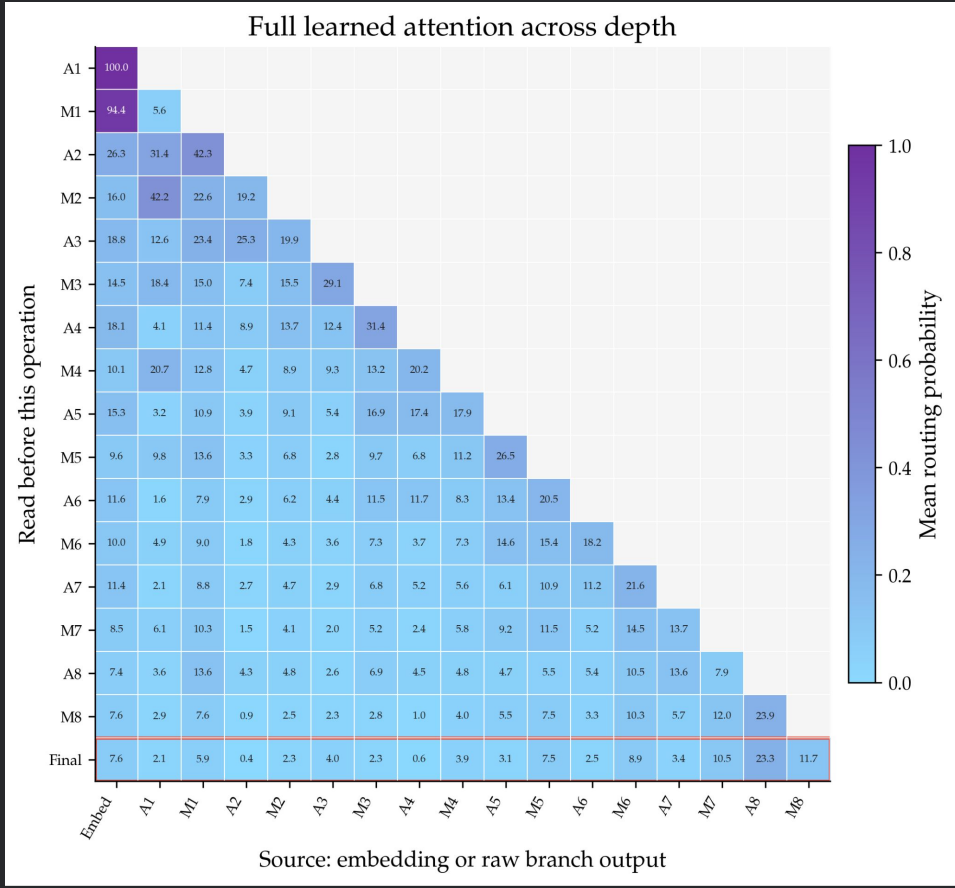
Answer (b): The correct choice was D, which pays slightly more attention to the later layers



Problem 5: Residual design

Reasoning: in the assignment, you might have built a heatmap that shows you this relationship across all of the layers

For all the later layers, there is more attention to the latest layer instead of the earliest layer. One intuition: later layers already contain earlier layers



Problem 6: Changing the recurrent update

Part (a): Rank the following recurrent updates

(a) Rank four designs (12 points). Let $c_t = \tanh(Wx_t + Uh_{t-1} + b)$.

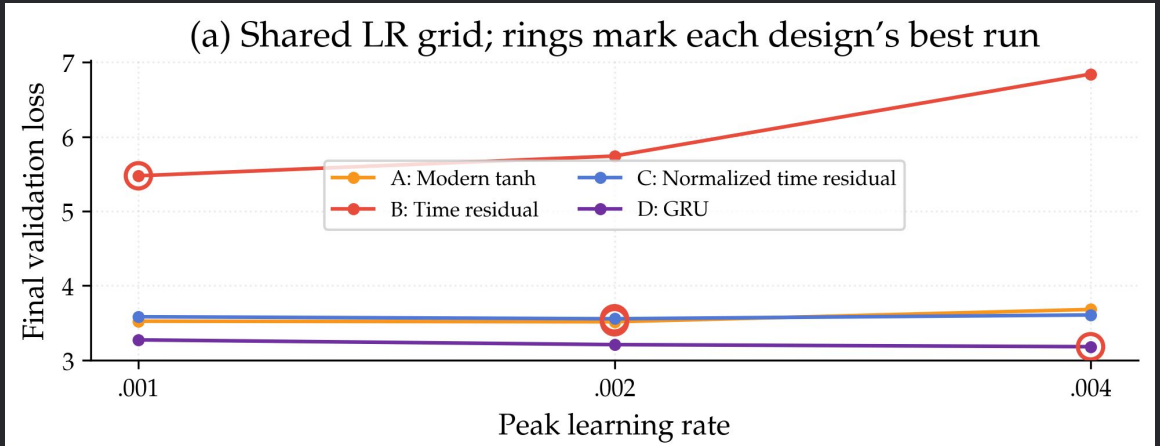
A:	Modern tanh baseline	$h_t = c_t,$	
B:	Time residual	$h_t = h_{t-1} + c_t,$	
C:	Normalized time residual	$s_t = h_{t-1} + c_t,$	$h_t = \dfrac{s_t}{\sqrt{\frac{1}{512}\sum_{j=1}^{512} s_{t,j}^2 + 10^{-5}}},$
D:	Full GRU	update below.	

The full GRU update is

$$
\begin{aligned}
r_t &= \sigma(W_r x_t + U_r h_{t-1} + b_r),\\
z_t &= \sigma(W_z x_t + U_z h_{t-1} + b_z),\\
\tilde{h}_t &= \tanh(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h),\\
h_t &= (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t.
\end{aligned}
$$

Problem 6: Changing the recurrent update

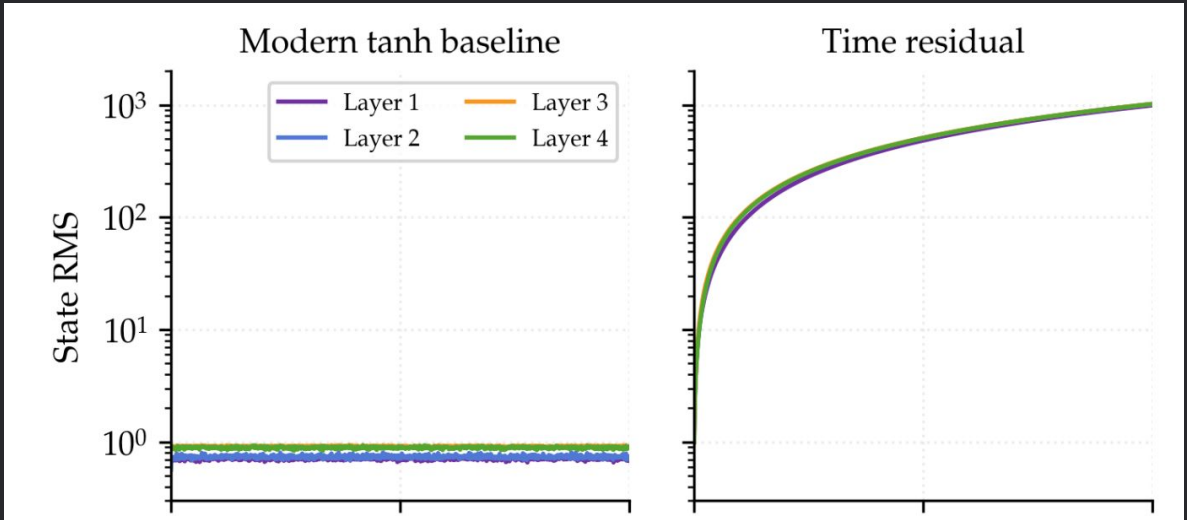
Answer to part (a): GRU < RNN < normed time residual < time residual



Design	Best LR	Best final loss
A: Modern tanh baseline	0.002	3.5160
B: Time residual	0.001	5.4757
C: Normalized time residual	0.002	3.5565
D: GRU	0.004	3.1813

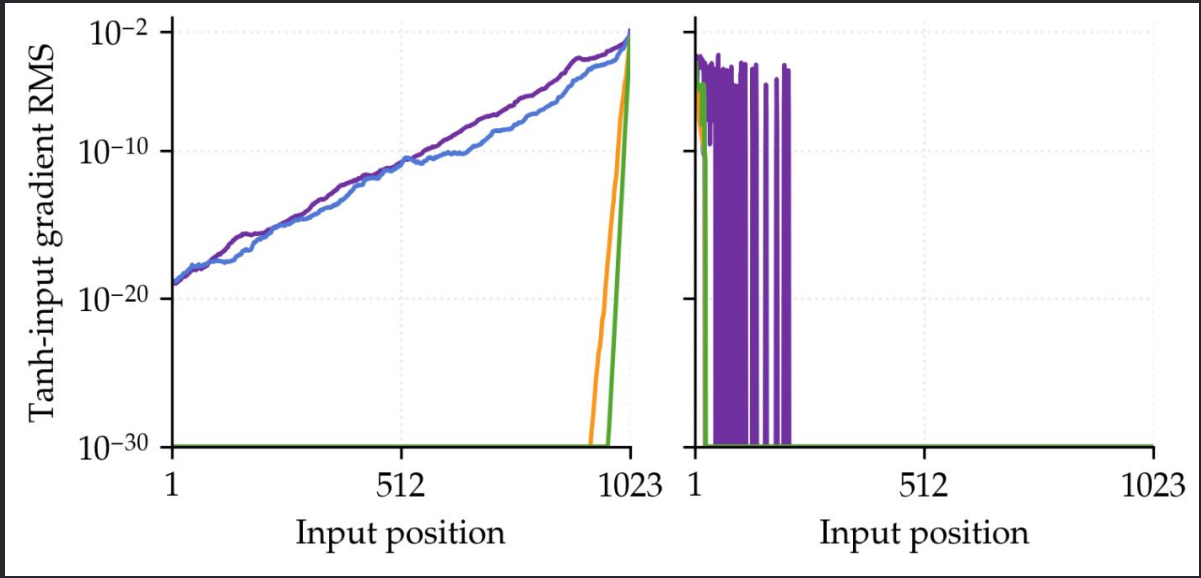
Problem 6: Changing the recurrent update

Reasoning: Adding residuals would lead to accumulation in hidden state norm over time



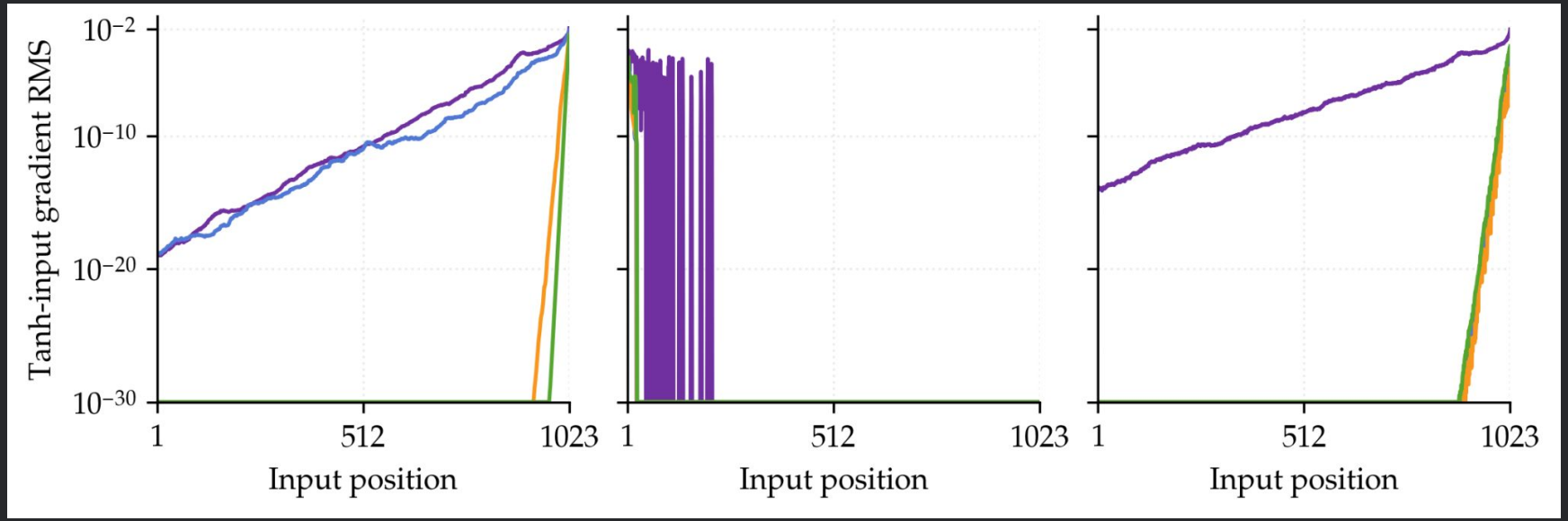
Problem 6: Changing the recurrent update

Reasoning: However, as we discussed for transformers, this would result in larger gradients for earlier time steps. This can result in crazy unstable gradients for this architecture (vanishing for default, exploding for time residuals)



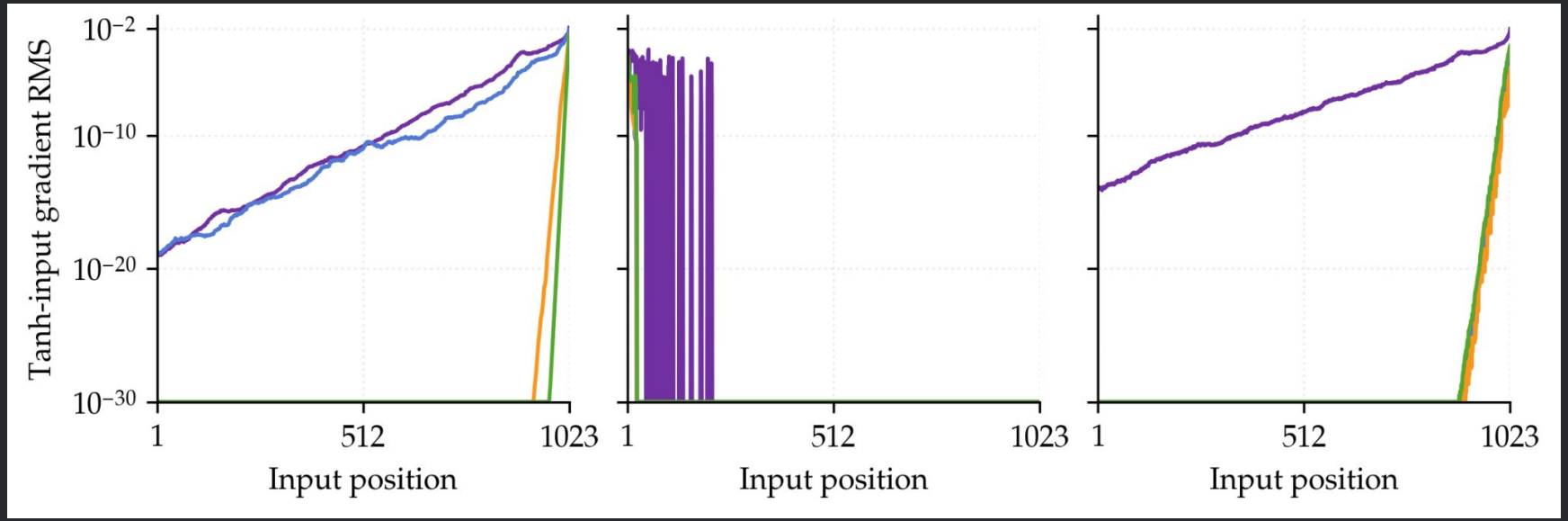
Problem 6: Changing the recurrent update

Reasoning: Normalization would fix the gradient stability.



Problem 6: Changing the recurrent update

Reasoning: Normalization would fix the gradient stability. It is harder to predict this beforehand, but normalization still hurts performance. One intuition is that normalization can hurt because it prevents input-dependent norm changes



Problem 6: Changing the recurrent update

Part (b): What happens if we took the GRU and tuned a fixed gate rather than learn an input-dependent gate?

(b) Fixed versus learned write gate (8 points). Keep the GRU's learned reset gate and candidate. Replace its write gate by a constant z, shared across all channels, layers, and time steps:

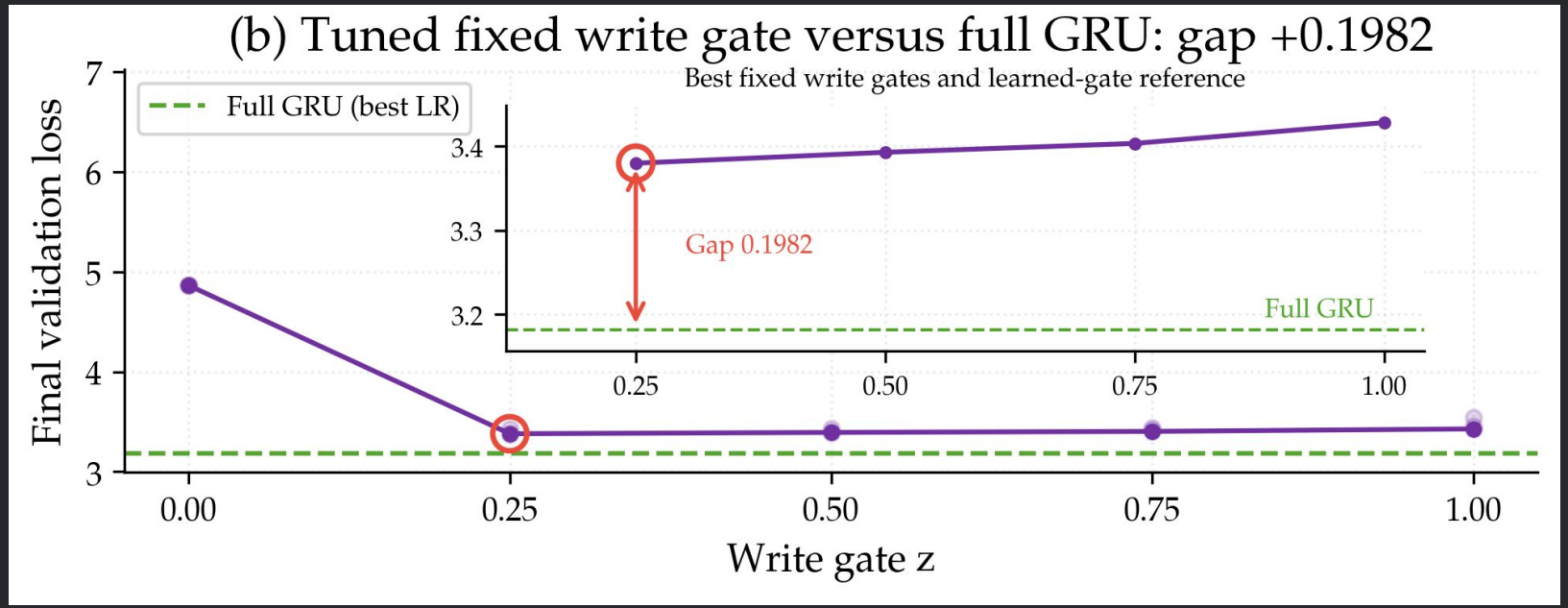
$$h_t = (1 - z)h_{t-1} + z\tilde{h}_t, \qquad h_0 = 0.$$

Train from scratch, tuning $z \in \{0.0, 0.25, 0.5, 0.75, 1.0\}$ and LR $\eta \in \{.001, .002, .004\}$. Compare the best fixed-z model with the full GRU from (a), tuned over the same LR grid. Which interval contains the final validation loss gap

$$\Delta = \min_{z,\eta} L_{\text{fixed}}(z, \eta) - \min_{\eta} L_{\text{GRU}}(\eta)?$$

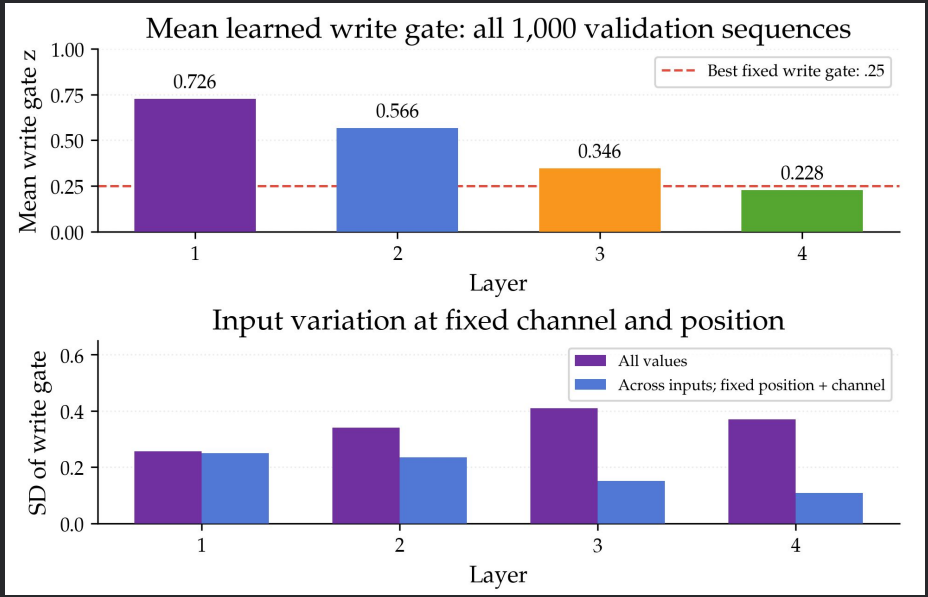
Problem 6: Changing the recurrent update

Answer to (b): The choice of write gate between 0.25, 0.5, and 0.75 had little impact. However, this significantly underperformed a data-dependent gate



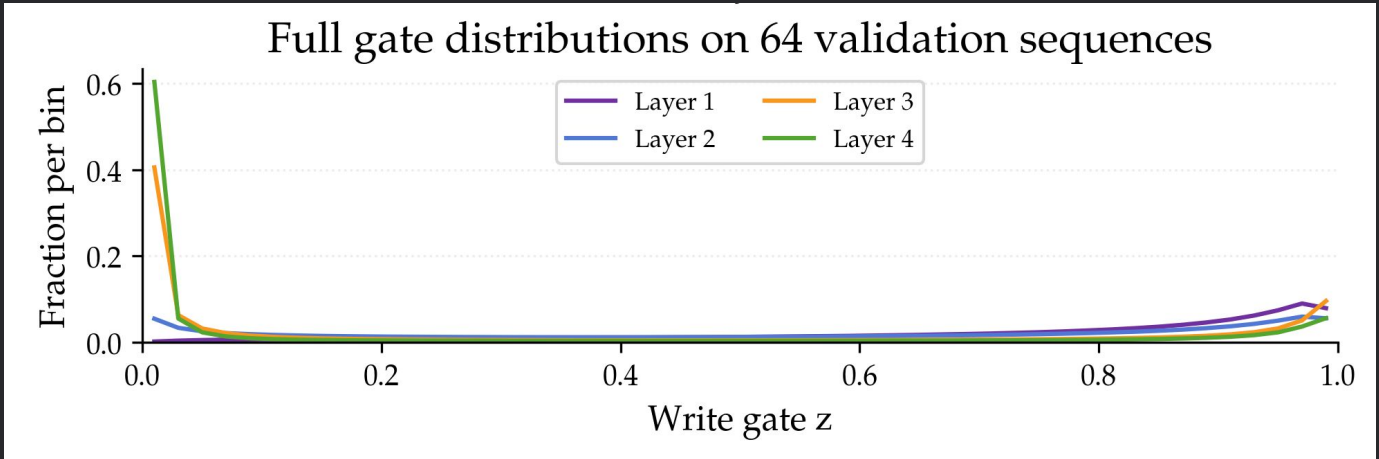
Problem 6: Changing the recurrent update

Reasoning: If you track the statistics of the gates, you'll notice that the model leverages the flexibility quite a bit. Different layers have different mean values and different inputs have different choices



Problem 6: Changing the recurrent update

Reasoning: Importantly, the write gate is often 0 or 1, ~40% outside 0.1-0.9). Therefore, its ability to turn on/off is important, not just its average value.



Next unit: Capacity

So far, we've reasoned about how to maintain stable training and signal propagation in various architectures.

Next unit, we will reason about capacity: how expressive is my model, how much capacity can we get for fixed compute/parameters, and how can we overcome bottlenecks

We will release the assignment hopefully by tonight.

Reminder: No class on Tuesday!