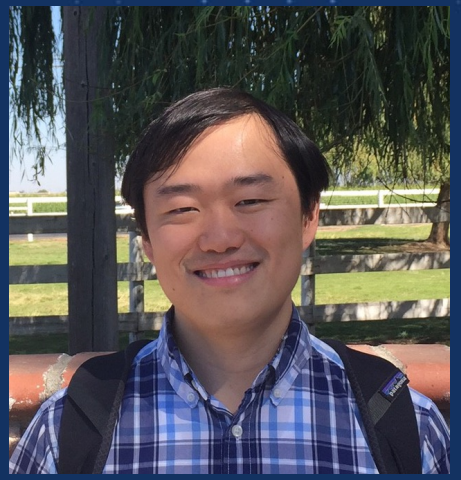
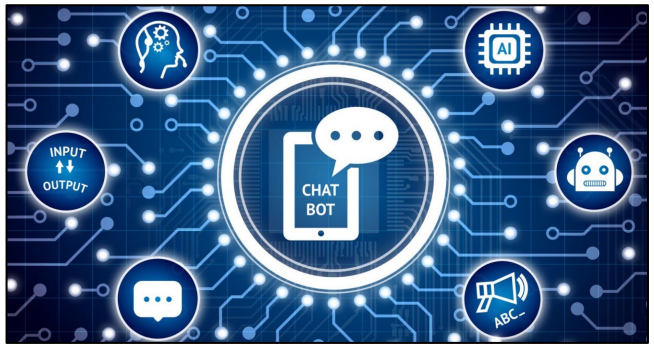
Towards Inside-out Interpretability: Analyzing Training Dynamics in Multi-layer Transformer

Yuandong Tian
Research Scientist

Meta AI (FAIR)

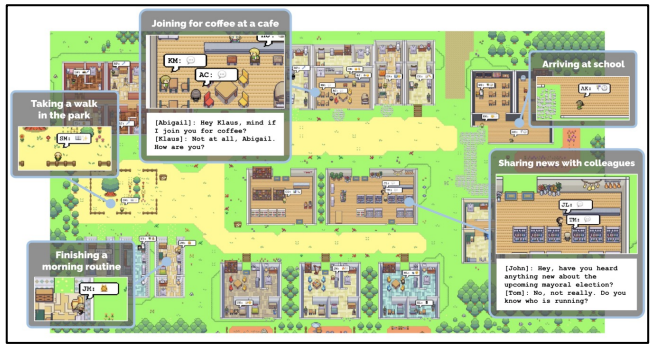


Large Language Models (LLMs)

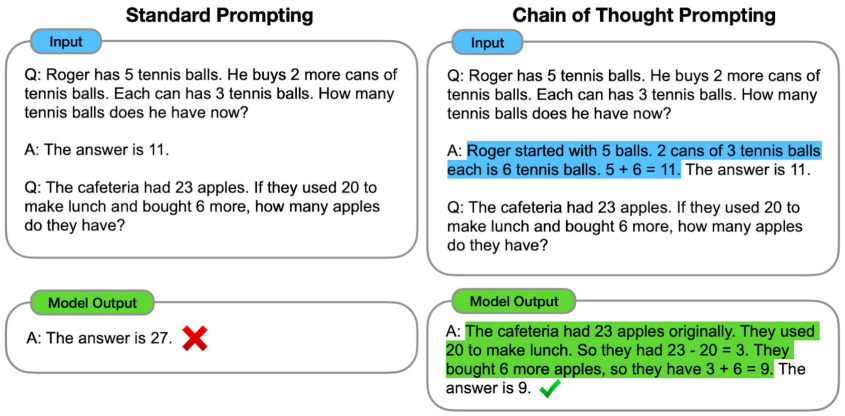


Conversational AI

Content Generation



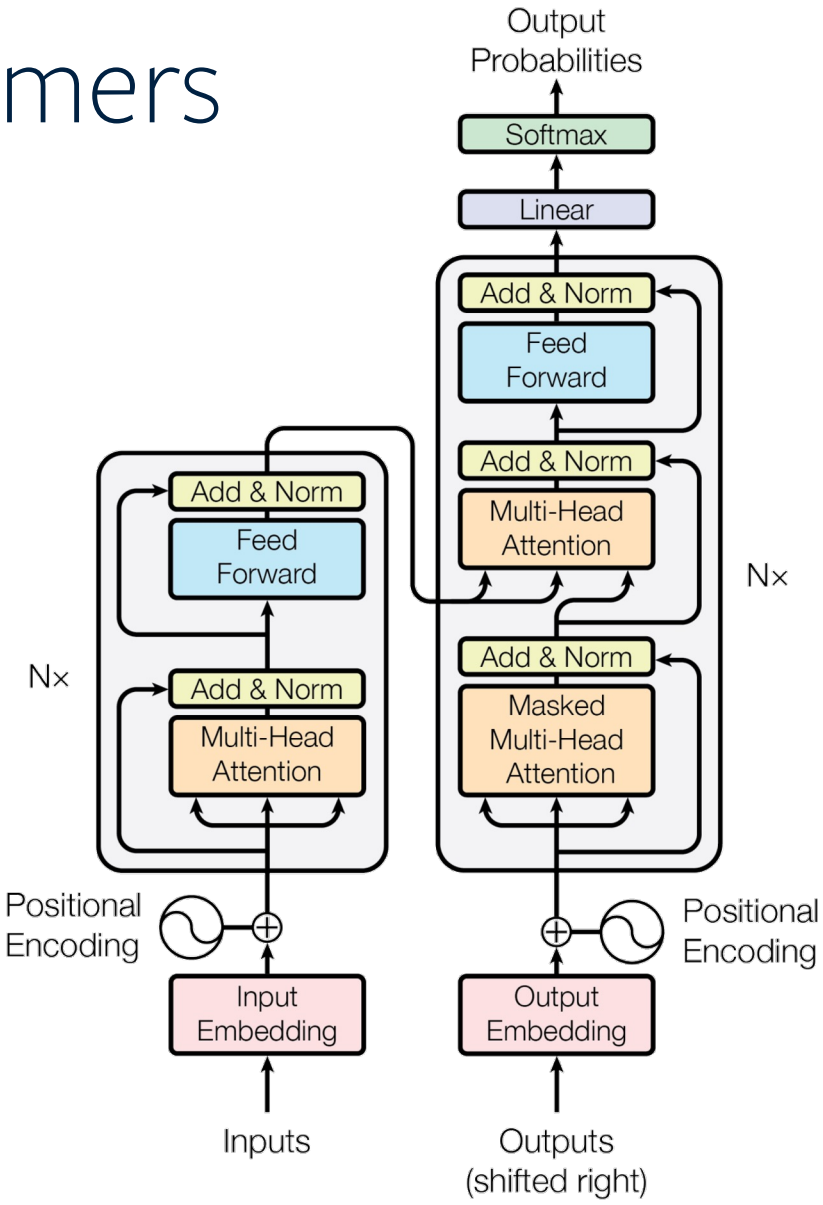
AI Agents

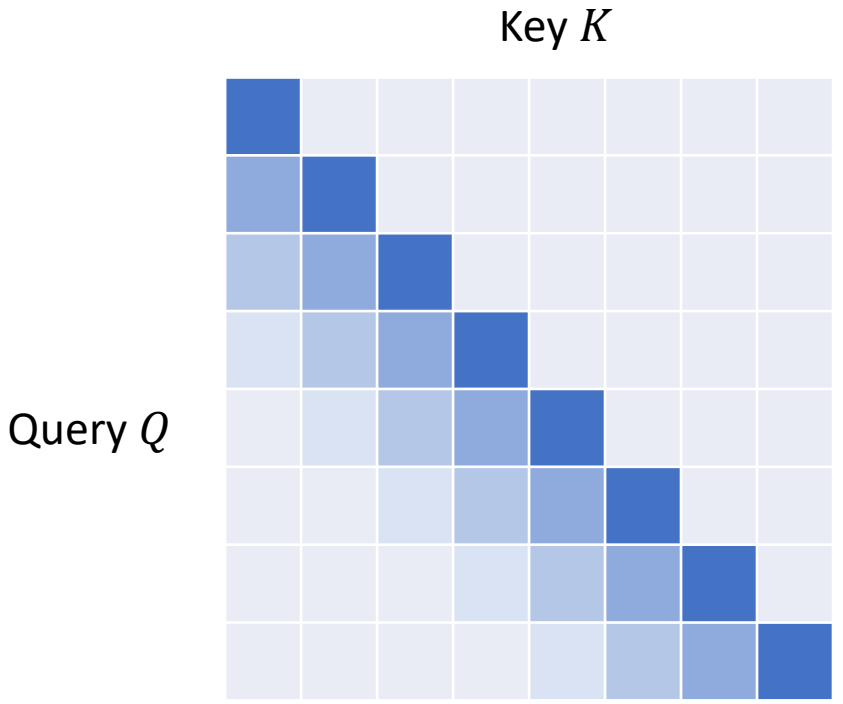


Reasoning

Planning

Transformers





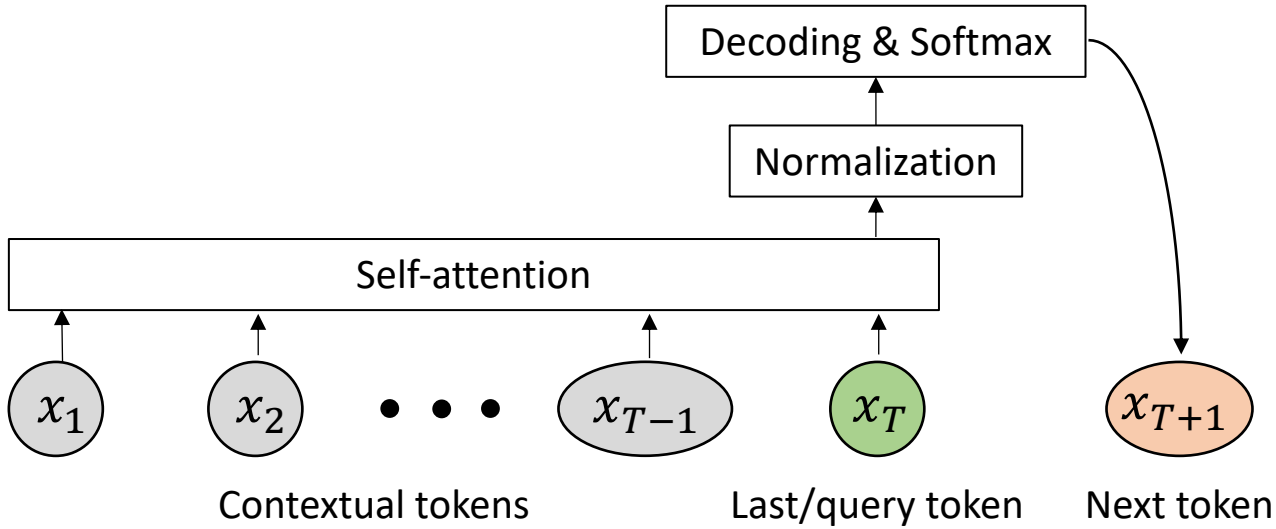
Attention mechanism

[A. Vaswani et al, Attention is all you need, NeurIPS'17]

Part I
Understanding Attention Mechanism

Understanding Attention in 1-layer Setting

$U = [\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots \boldsymbol{u}_M]^T$: token embedding matrix



Self-attention

$$\hat{\boldsymbol{u}}_T = \sum_{t=1}^{T-1} b_{tT} \boldsymbol{u}_{x_t} = U^T X^T \boldsymbol{b}_T$$

$$b_{tT} := \frac{\exp(\boldsymbol{u}_{x_T}^\top W_Q W_K^\top \boldsymbol{u}_{x_t} / \sqrt{d})}{\sum_{t=1}^{T-1} \exp(\boldsymbol{u}_{x_T}^\top W_Q W_K^\top \boldsymbol{u}_{x_t} / \sqrt{d})}$$

Normalized version $\tilde{\boldsymbol{u}}_T = U^T \mathrm{LN}(X^T \boldsymbol{b}_T)$

Objective:

$$\max_{W_K, W_Q, W_V, U} J = \mathbb{E}_D \left[\boldsymbol{u}_{x_{T+1}}^T W_V \tilde{\boldsymbol{u}}_T - \log \sum_l \exp(\boldsymbol{u}_l^T W_V \tilde{\boldsymbol{u}}_T) \right]$$

[*Y. Tian et al*, Scan and Snap: Understanding Training Dynamics and Token Composition in 1-layer Transformer, *NeurIPS'23*]

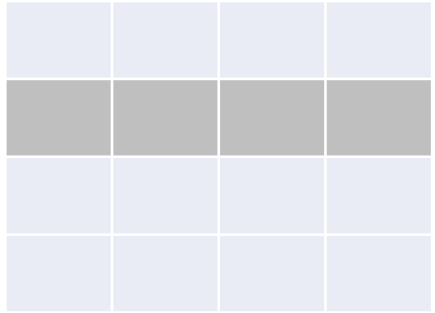
Reparameterization

- Parameters W_K, W_Q, W_V, U makes the dynamics complicated.

- Reparameterize the problem with independent variable Y and Z
  - $Y = UW_V^T U^T$
  - $Z = UW_Q W_K^T U^T$ (pairwise logits of self-attention matrix)

- Then the dynamics becomes easier to analyze

Training dynamics of Y and Z

$Z =$  $\boldsymbol{z}_m$

Training Dynamics:

$$\dot{Y} = \eta_Y \mathrm{LN}(X^T \boldsymbol{b}_T)(\boldsymbol{x}_{T+1} - \boldsymbol{\alpha})^T$$

$$\dot{Z} = \eta_Z \boldsymbol{x}_T (\boldsymbol{x}_{T+1} - \boldsymbol{\alpha})^T Y^T \frac{P^{\perp}_{X^T \boldsymbol{b}_T}}{\|X^T \boldsymbol{b}_T\|_2} X^T \mathrm{diag}(\boldsymbol{b}_T) X$$

$\boldsymbol{z}_m$: All logits of the contextual tokens when attending to last token $x_T = m$

Here $Z = [\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_M]^T$, each $\boldsymbol{z}_m \in \mathbb{R}^M$ is the attention score for query/last token m:

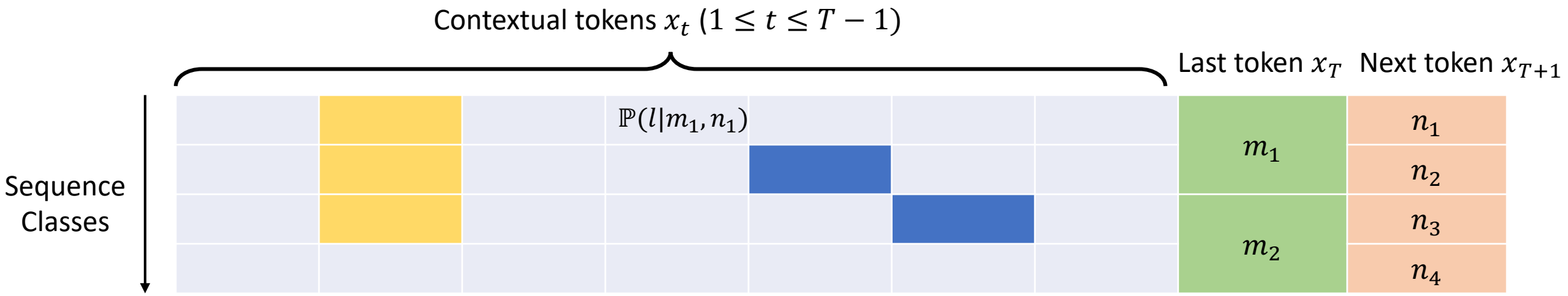
$$\dot{\boldsymbol{z}}_m = \eta_Z X^\top[i] \mathrm{diag}(\boldsymbol{b}_T[i]) X[i] \frac{P^{\perp}_{X^\top[i] \boldsymbol{b}_T[i]}}{\|X^\top[i] \boldsymbol{b}_T[i]\|_2} Y(\boldsymbol{x}_{T+1}[i] - \boldsymbol{\alpha}[i])$$

Major Assumptions

- No positional encoding
- Sequence length $T \to +\infty$
- Learning rate of decoder Y larger than self-attention layer Z ($\eta_Y \gg \eta_Z$)
- Other technical assumptions

Data Distribution

$x_t \in [M]$ for $1 \le t \le T$
$x_{T+1} \in [K]$
$K \ll M$



Distinct tokens: There exists unique n so that $\mathbb{P}(l|n) > 0$

Common tokens: There exists multiple n so that $\mathbb{P}(l|n) > 0$

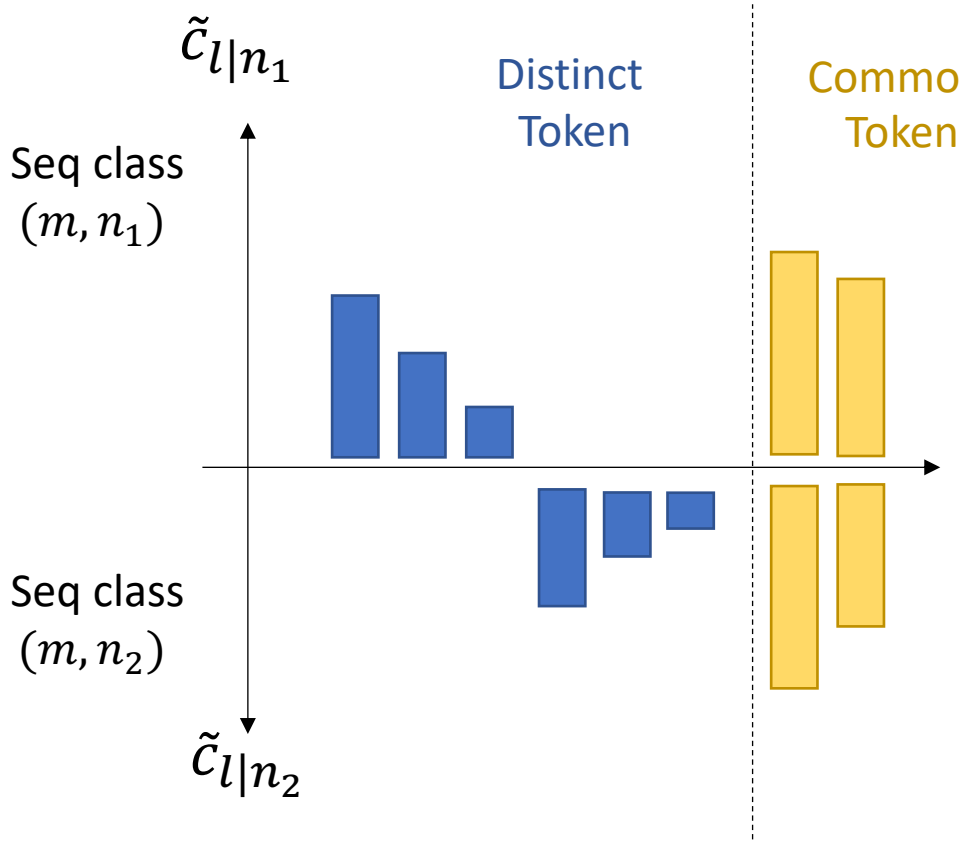
$\mathbb{P}(l|m,n) = \mathbb{P}(l|n)$ is the conditional probability of token l given last token $x_T = m$ and $x_{T+1} = n$

Assumption: $m = \psi(n)$, i.e., no next token shared among different last tokens

Question: Given the data distribution, how does the self-attention layer behave?

Overall Picture of the Training Dynamics

At initialization



Co-occurrence probability

$$\tilde{c}_{l|n_1} := \mathbb{P}(l|m, n_1) \exp(z_{ml})$$

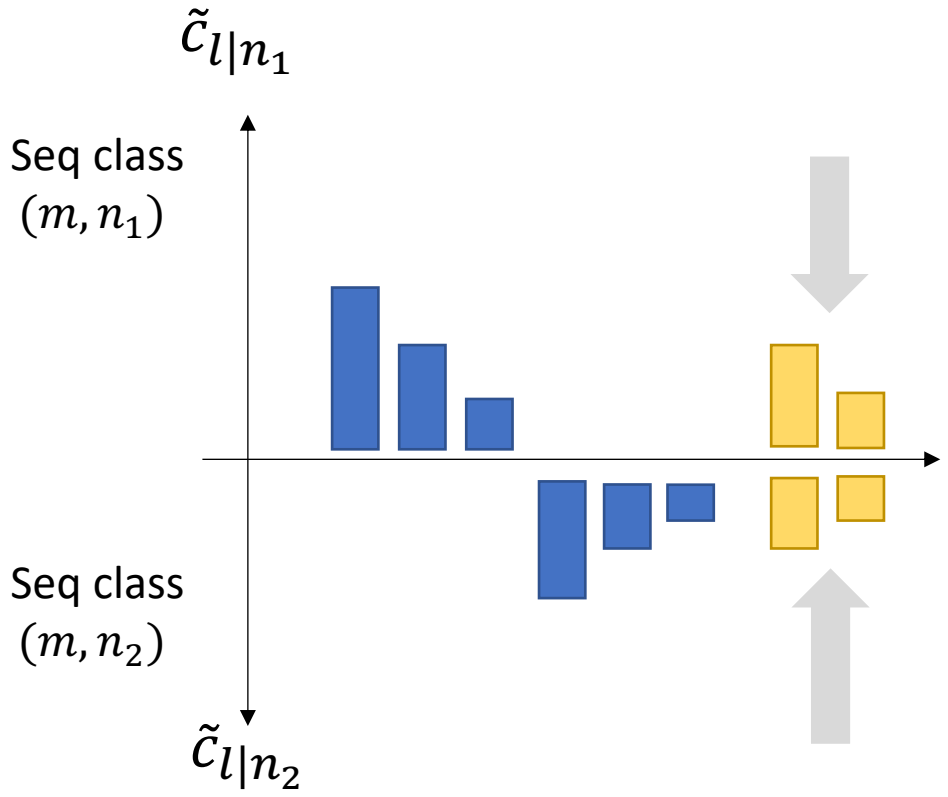
Initial condition: $z_{ml}(0) = 0$

$$Z = \begin{bmatrix} & & & \\ \boldsymbol{z}_m & & & \\ & & & \\ & & & \end{bmatrix}$$

$\boldsymbol{z}_m$: All logits of the contextual tokens when attending to last token $x_T = m$

Overall Picture of the Training Dynamics

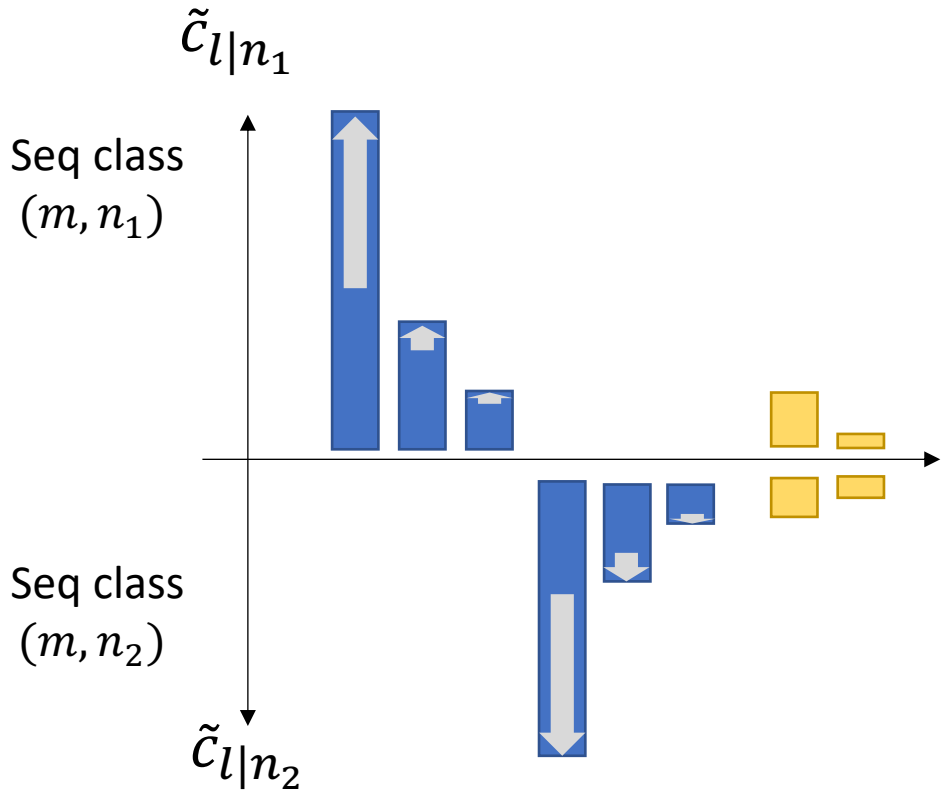
Common Token Suppression



(a) $\dot{z_{ml}} < 0$, for common token l

Overall Picture of the Training Dynamics

Winners-emergence



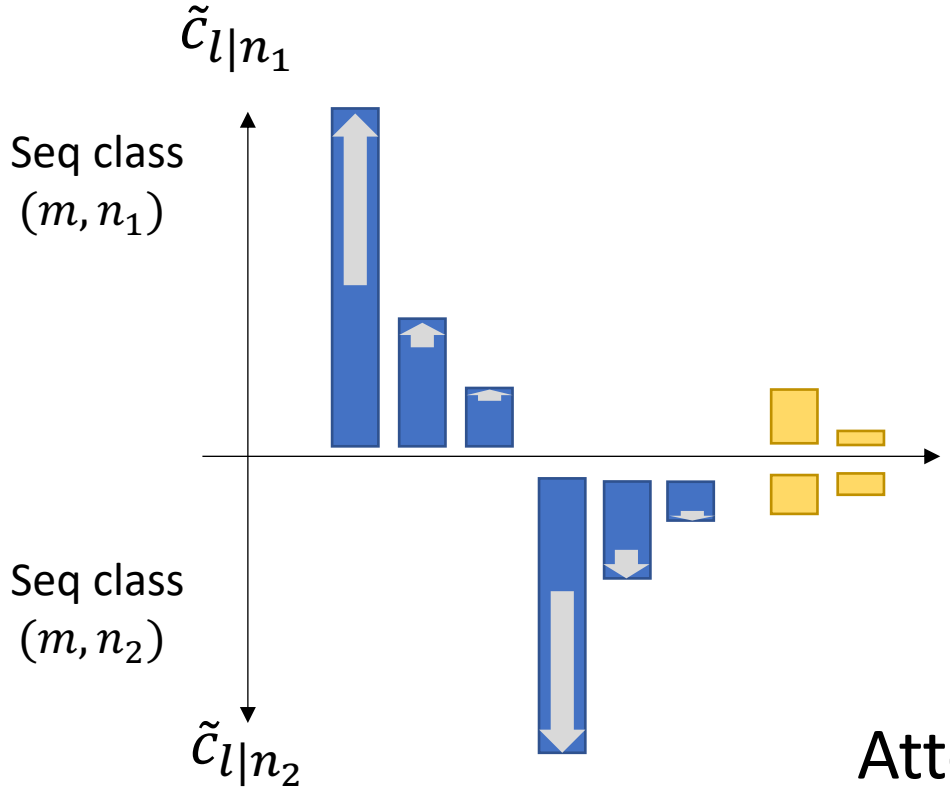
(a) $\dot{z_{ml}} < 0$, for common token l

(b) $\dot{z_{ml}} > 0$, for distinct token l

Learnable TF-IDF (Term Frequency, Inverse Document Frequency)

Overall Picture of the Training Dynamics

Winners-emergence



(a) $\dot{z_{ml}} < 0$, for common token l

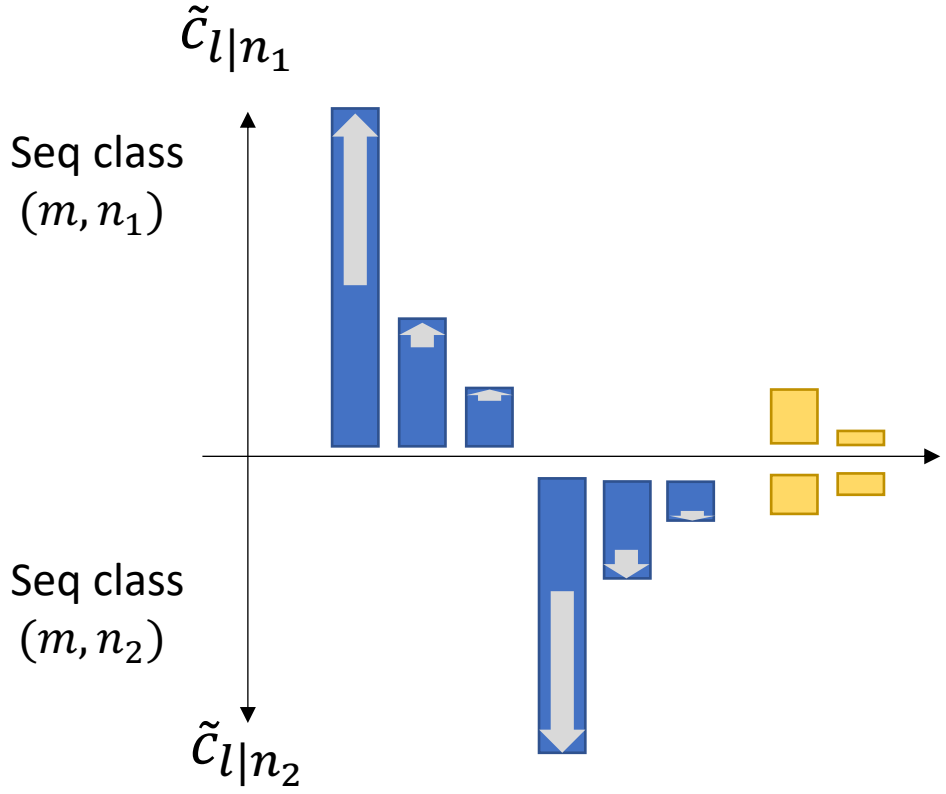
(b) $\dot{z_{ml}} > 0$, for distinct token l

(c) $z_{ml}(t)$ grows faster with larger $\mathbb{P}(l|m, n)$

Attention looks for **discriminative** tokens that **frequently co-occur** with the query.

Overall Picture of the Training Dynamics

Winners-emergence



(c) $z_{ml}(t)$ grows faster with larger $\mathbb{P}(l|m,n)$

Theorem 3 Relative gain $r_{l/l'|n}(t) := \frac{\tilde{c}^2_{l|n}(t)}{\tilde{c}^2_{l'|n}(t)} - 1$ has a close form:

$$r_{l/l'|n}(t) = r_{l/l'|n}(0)\chi_l(t)$$

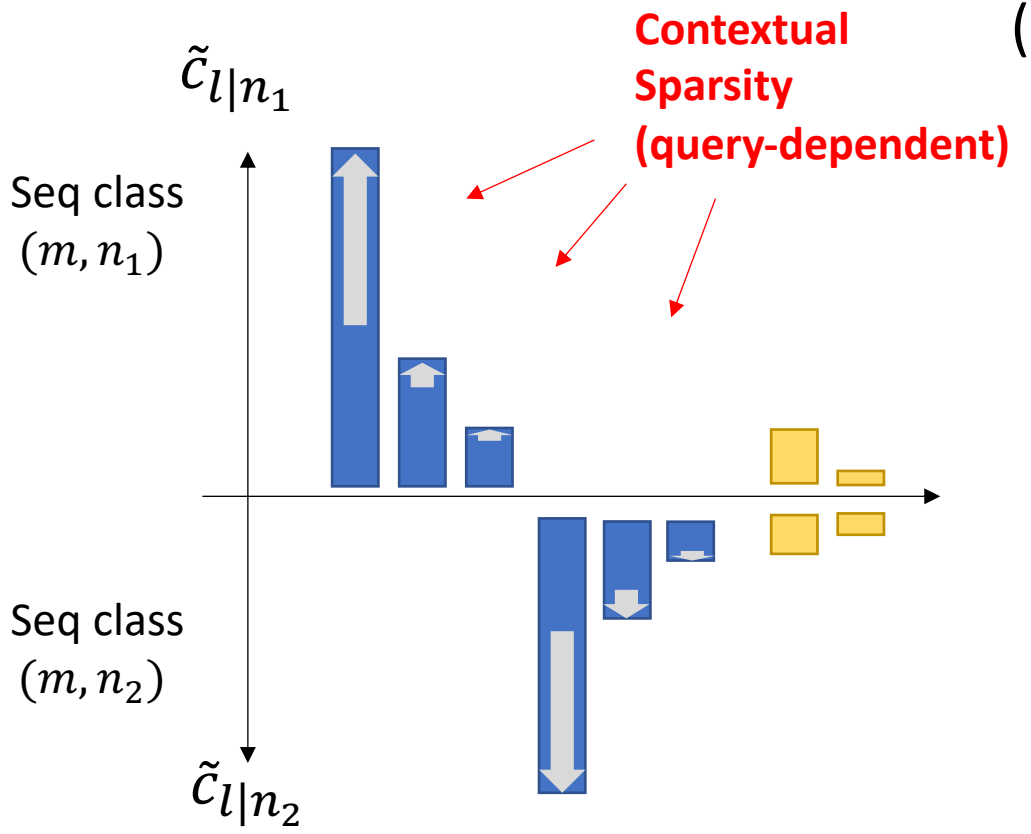
If l_0 is the dominant token: $r_{l_0/l|n}(0) > 0$ for all $l \neq l_0$ then

$$e^{2f^2_{nl_0}(0)B_n(t)} \leq \chi_{l_0}(t) \leq e^{2B_n(t)}$$

where $B_n(t) \geq 0$ monotonously increases, $B_n(0) = 0$

Overall Picture of the Training Dynamics

Winners-emergence



(c) $z_{ml}(t)$ grows faster with larger $\mathbb{P}(l|m,n)$

Theorem 3 Relative gain $r_{l/l'|n}(t) := \frac{\tilde{c}^2_{l|n}(t)}{\tilde{c}^2_{l'|n}(t)} - 1$ has a close form:

$$r_{l/l'|n}(t) = r_{l/l'|n}(0)\chi_l(t)$$

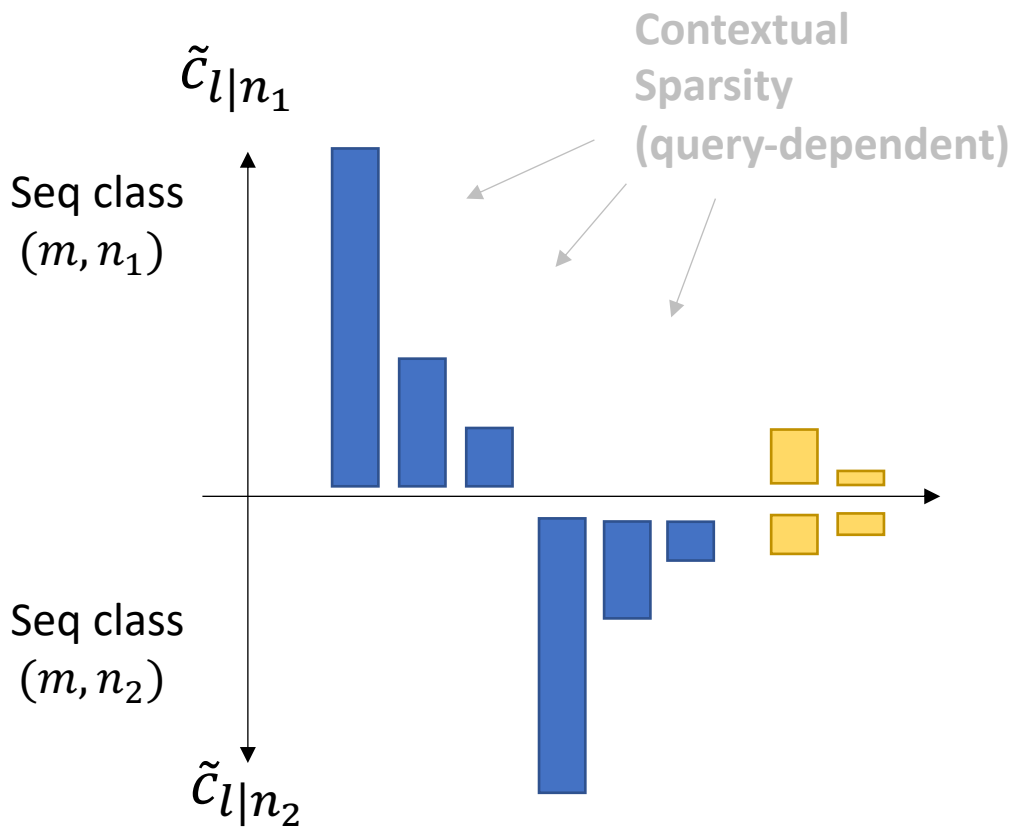
If l_0 is the dominant token: $r_{l_0/l|n}(0) > 0$ for all $l \neq l_0$ then

$$e^{2f^2_{nl_0}(0)B_n(t)} \leq \chi_{l_0}(t) \leq e^{2B_n(t)}$$

where $B_n(t) \geq 0$ monotonously increases, $B_n(0) = 0$

Overall Picture of the Training Dynamics

Attention frozen



Theorem 4 When $t \to +\infty$,

$$B_n(t) \sim \ln\left(C_0 + 2K\frac{\eta_z}{\eta_Y}\ln^2\left(\frac{M\eta_Y t}{K}\right)\right)$$

Attention **scanning**:

When training starts, $B_n(t) = O(\ln t)$

Attention **snapping**:

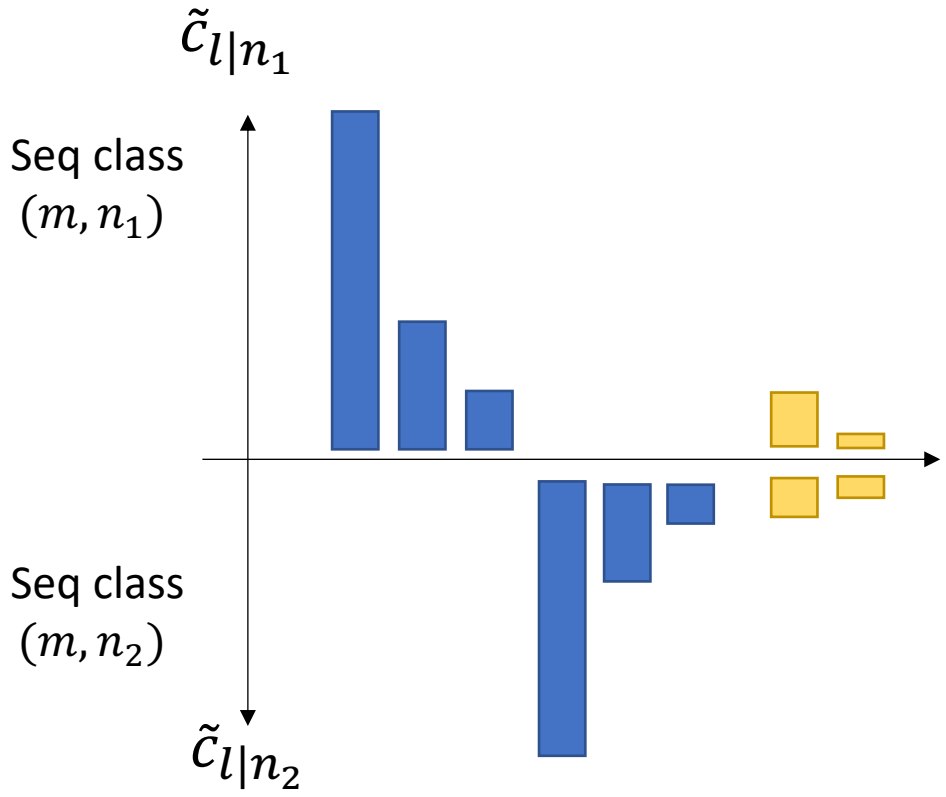
When $t \geq t_0 = O\left(\frac{2K\ln M}{\eta_Y}\right)$, $B_n(t) = O(\ln\ln t)$

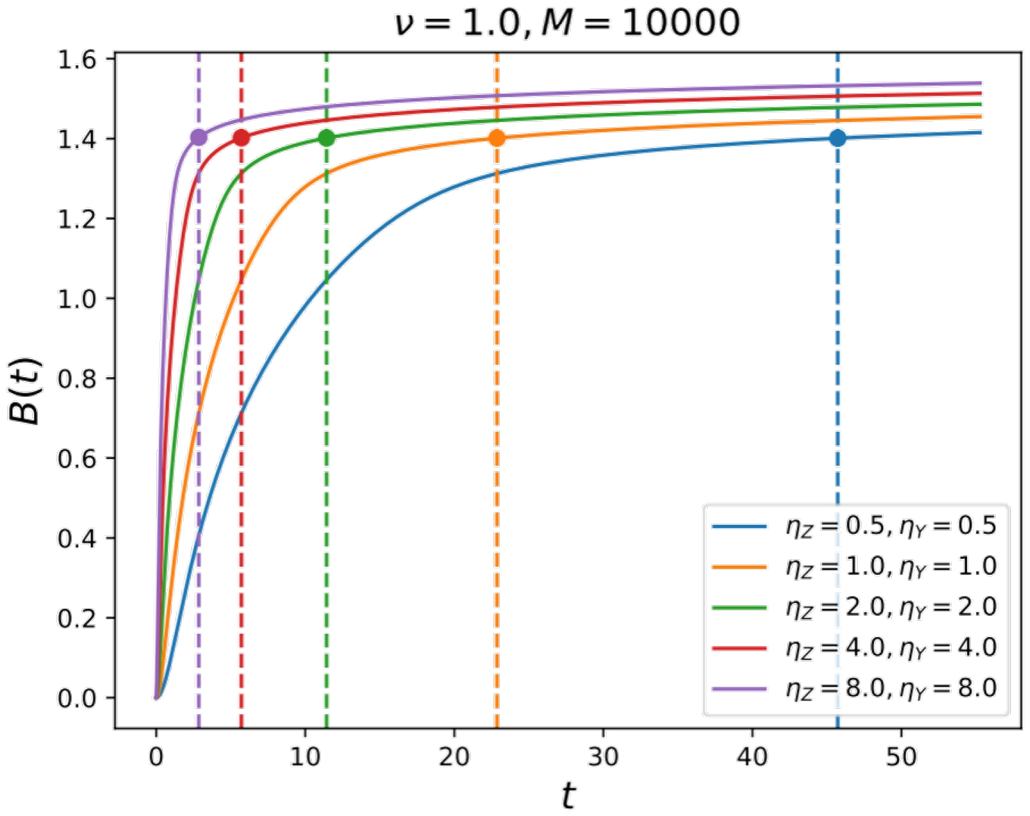
(1) η_z and η_Y are large, $B_n(t)$ is large and attention is sparse

(2) Fixing η_z, large η_Y leads to slightly small $B_n(t)$ and denser attention

Overall Picture of the Training Dynamics

Attention frozen



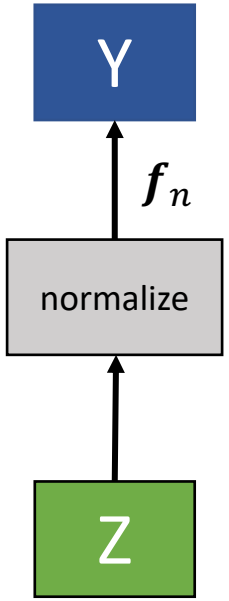


Larger learning rate η_z leads to faster phase transition

$$B_n(t) \sim \ln\left(C_0 + 2K\frac{\eta_z}{\eta_Y}\ln^2\left(\frac{M\eta_Y t}{K}\right)\right)$$

Overall strategy of the theoretical analysis

- The power of infinite sequence length $T \to +\infty$



Lemma 2. *Given the event* $\{x_T = m, x_{T+1} = n\}$, *when* $T \to +\infty$, *we have*

$$X^\top \boldsymbol{b}_T \to \boldsymbol{c}_{m,n}, \qquad X^\top \mathrm{diag}(\boldsymbol{b}_T) X \to \mathrm{diag}(\boldsymbol{c}_{m,n})$$

where $\boldsymbol{c}_{m,n} = [c_{1|m,n}, c_{2|m,n}, \ldots, c_{M|m,n}]^\top \in \mathbb{R}^M$. *Note that* $\boldsymbol{c}_{m,n}^\top \mathbf{1} = 1$.

Here $c_{l|m,n} := \dfrac{T\mathbb{P}(l|m,n)\exp(z_{ml})}{\sum_{l'} T\mathbb{P}(l'|m,n)\exp(z_{ml'})} = \dfrac{\mathbb{P}(l|m,n)\exp(z_{ml})}{\sum_{l'} \mathbb{P}(l'|m,n)\exp(z_{ml'})} =: \dfrac{\tilde{c}_{l|m,n}}{\sum_{l'} \tilde{c}_{l'|m,n}}$

Define $\boldsymbol{f}_n := \boldsymbol{f}_{m,n} := \boldsymbol{c}_{m,n} / \left\| \boldsymbol{c}_{m,n} \right\|_2$ a ℓ_2-normalized version of $\boldsymbol{c}_{m,n}$.

Overall strategy of the theoretical analysis

- Since $\eta_Y \gg \eta_Z$, we analyze the dynamics of decoder Y first, treating the output of Z as constant.

$$\dot{Y} = \eta_Y \boldsymbol{f}_n (\boldsymbol{e}_n - \boldsymbol{\alpha}_n)^\top, \quad \boldsymbol{\alpha}_n = \frac{\exp(Y^\top \boldsymbol{f}_n)}{\mathbf{1}^\top \exp(Y^\top \boldsymbol{f}_n)}$$

- The analysis gives backpropagated gradient:

Theorem 1. *If Assumption 2 holds, the initial condition* $Y(0) = 0$, $M \gg 100$, η_Y *satisfies* $M^{-0.99} \ll \eta_Y < 1$, *and each sequence class appears uniformly during training, then after* $t \gg K^2$ *steps of batch size* 1 *update, given event* $x_{T+1}[i] = n$, *the backpropagated gradient* $\boldsymbol{g}[i] := Y(\boldsymbol{x}_{T+1}[i] - \boldsymbol{\alpha}[i])$ *takes the following form:*

$$\boldsymbol{g}[i] = \gamma \left(\iota_n \boldsymbol{f}_n - \sum_{n' \neq n} \beta_{nn'} \boldsymbol{f}_{n'} \right) \tag{9}$$

Overall strategy of the theoretical analysis

- Given the backpropagated gradient, we can analyze the behavior of the self-attention layer.

Theorem 2 (Fates of contextual tokens). *Let G_{CT} be the set of common tokens (CT), and $G_{DT}(n)$ be the set of distinct tokens (DT) that belong to next token n. Then if Assumption 2 holds, under the self-attention dynamics (Eqn. 10), we have:*

- ***(a)*** *for any distinct token $l \in G_{DT}(n)$, $\dot{z}_{ml} > 0$ where $m = \psi(n)$;*
- ***(b)*** *if $|G_{CT}| = 1$ and at least one next token $n \in \psi^{-1}(m)$ has at least one distinct token, then for the single common token $l \in G_{CT}$, $\dot{z}_{ml} < 0$.*

Simple Real-world Experiments

WikiText2 (original parameterization)

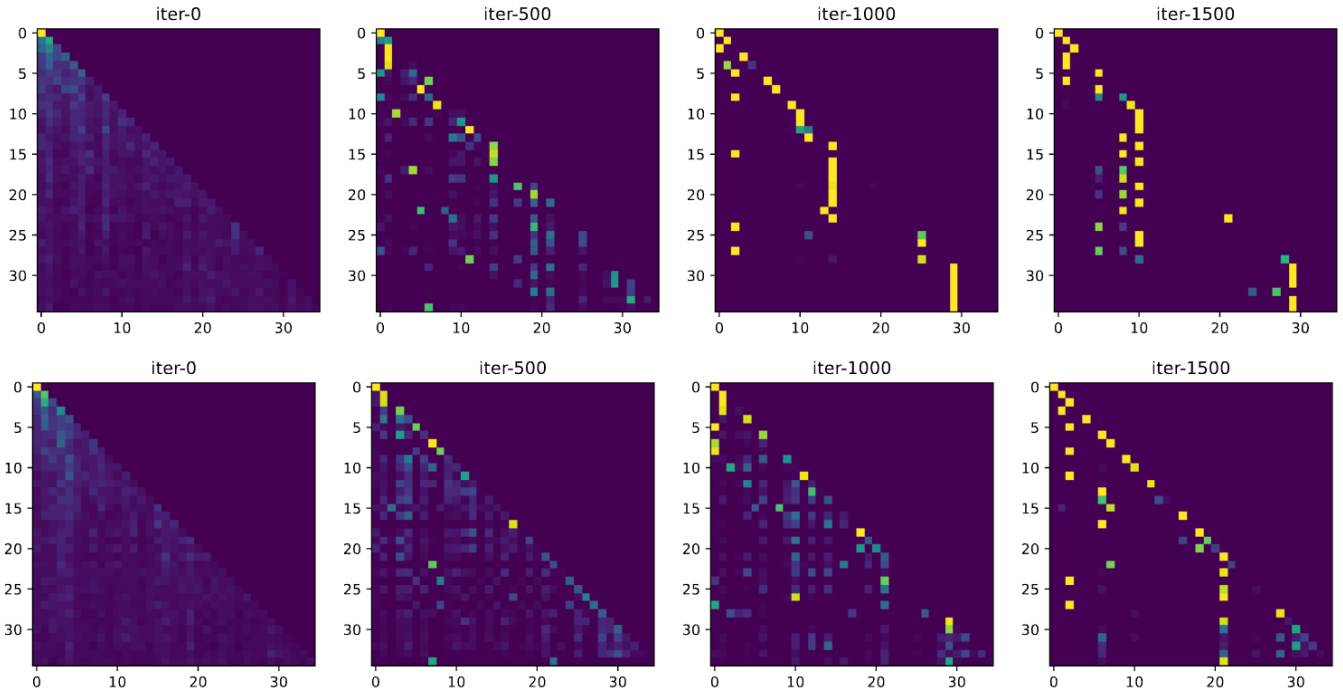


Figure 7: Attention patterns in the lowest self-attention layer for 1-layer (top) and 3-layer (bottom) Transformer trained on WikiText2 using SGD (learning rate is 5). Attention becomes sparse over training.

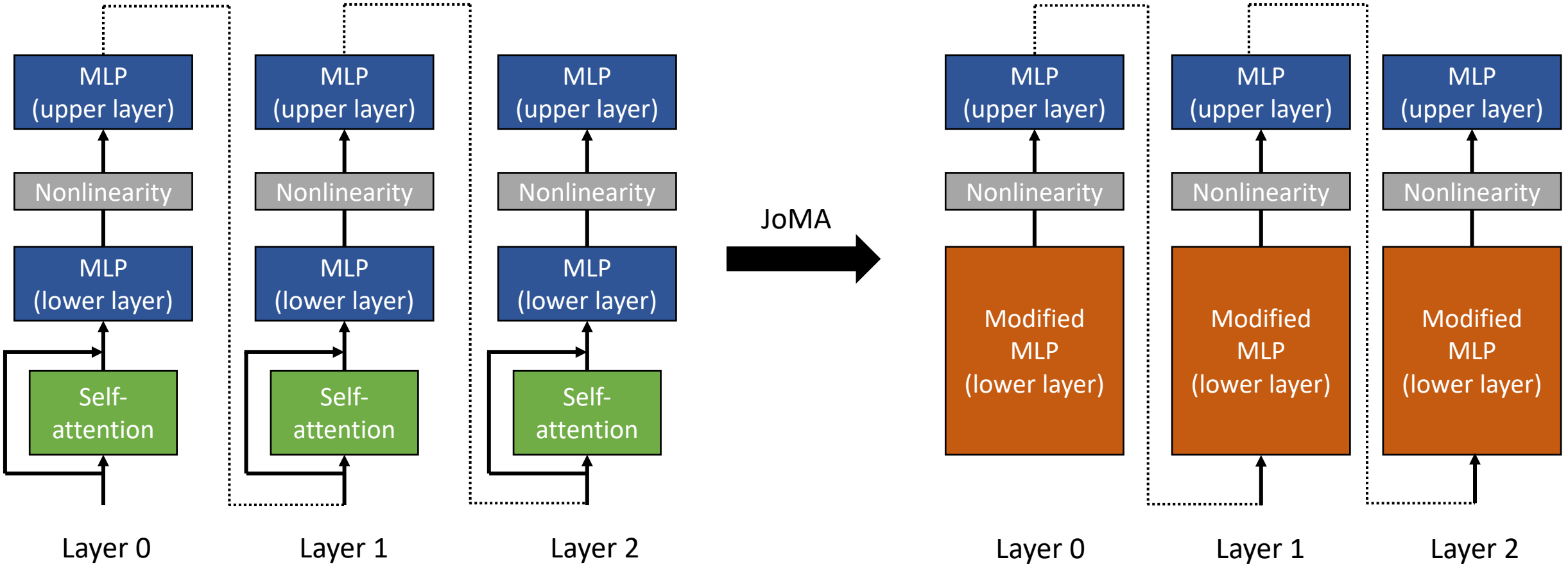
How to get rid of the assumptions?

- A few annoying assumptions in the analysis
  - No residual connections
  - No embedding vectors
  - The decoder needs to learn faster than the self-attention ($\eta_Y \gg \eta_Z$).
  - Single layer analysis

- How to get rid of them?

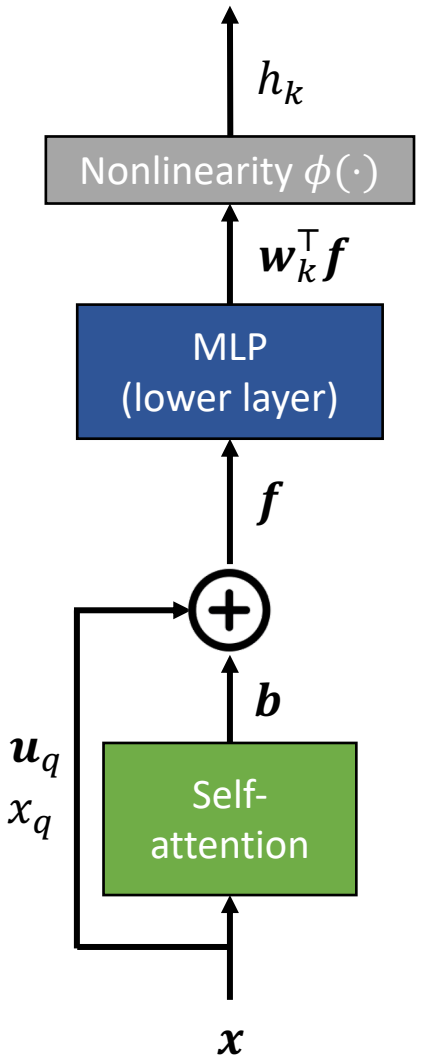
- New research work: **JoMA**

JoMA: JOint Dynamics of MLP/Attention layers



[Y. Tian et al, *JoMA: Demystifying Multilayer Transformers via JOint Dynamics of MLP and Attention,* ICLR'24]

JoMA Settings



"This is an apple"

$$h_k = \phi(\boldsymbol{w}_k^\top \boldsymbol{f})$$

$$\boldsymbol{f} = U_C \boldsymbol{b} + \boldsymbol{u}_q$$

U_C and $\boldsymbol{u}_q$ are embeddings

$$\boldsymbol{b} = \sigma(\boldsymbol{z}_q) \circ \boldsymbol{x}/A$$

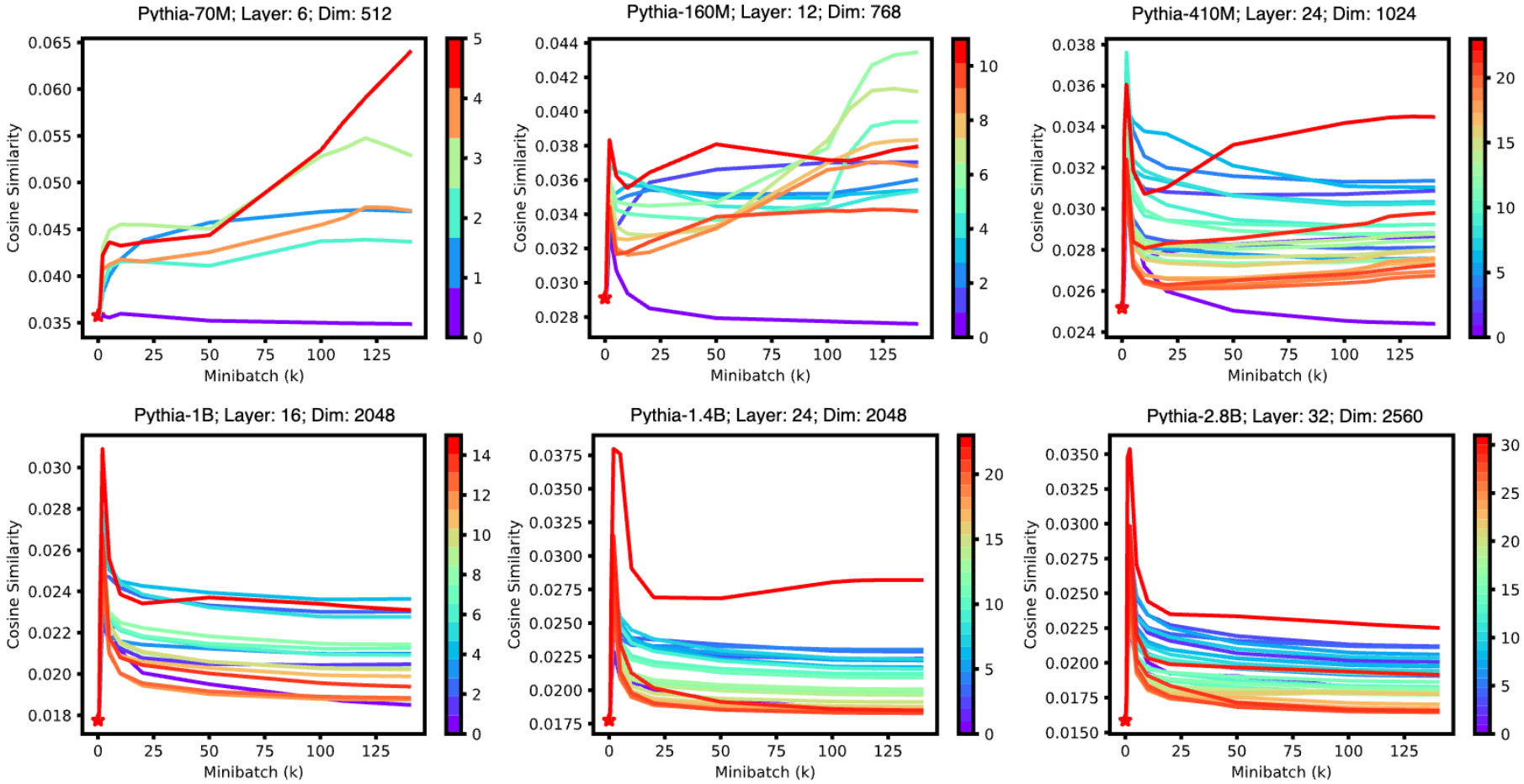
SoftmaxAttn: $b_l = \frac{x_l e^{z_{ql}}}{\sum_l x_l e^{z_{ql}}}$

ExpAttn: $b_l = x_l e^{z_{ql}}$

LinearAttn: $b_l = x_l z_{ql}$

Assumption (Orthogonal Embeddings $[U_c, u_q]$)

Cosine similarity between embedding vectors at different layers.



JoMA Dynamics

$$\dot{\boldsymbol{v}} = \boldsymbol{\Delta}_m \circ \exp(\boldsymbol{z}_m) = \boldsymbol{\Delta}_m \circ \exp\left(\frac{\boldsymbol{v}^2}{2} + \boldsymbol{c}\right)$$

Modified MLP (lower layer)

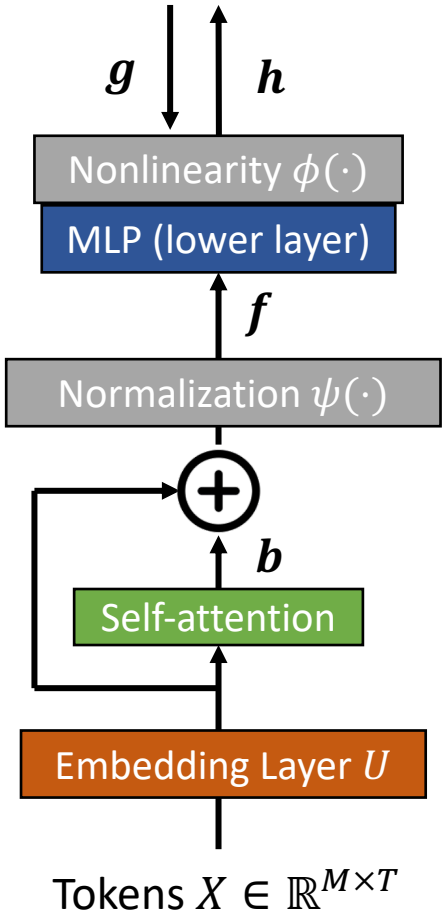
Theorem 1 (JoMA). *Let* $\boldsymbol{v}_k := U_C^\top \boldsymbol{w}_k$, *then the dynamics of Eqn. 3 satisfies the invariants:*

- *Linear attention.* *The dynamics satisfies* $\boldsymbol{z}_m^2(t) = \sum_k \boldsymbol{v}_k^2(t) + \boldsymbol{c}$.
- *Exp attention.* *The dynamics satisfies* $\boldsymbol{z}_m(t) = \frac{1}{2}\sum_k \boldsymbol{v}_k^2(t) + \boldsymbol{c}$.
- *Softmax attention.* *If* $\bar{\boldsymbol{b}}_m := \mathbb{E}_{q=m}[\boldsymbol{b}]$ *is a constant over time and* $\mathbb{E}_{q=m}\left[\sum_k g_{h_k} h'_k \boldsymbol{b}\boldsymbol{b}^\top\right] = \bar{\boldsymbol{b}}_m \mathbb{E}_{q=m}\left[\sum_k g_{h_k} h'_k \boldsymbol{b}\right]$, *then the dynamics satisfies* $\boldsymbol{z}_m(t) = \frac{1}{2}\sum_k \boldsymbol{v}_k^2(t) - \|\boldsymbol{v}_k(t)\|_2^2 \bar{\boldsymbol{b}}_m + \boldsymbol{c}$.

Under zero-initialization ($\boldsymbol{w}_k(0) = 0$, $\boldsymbol{z}_m(0) = 0$), *then the time-independent constant* $\boldsymbol{c} = 0$.

There is residual connection.
Joint dynamics works for any learning rates between self-attention and MLP layer.
No assumption on the data distribution.

Comparison between Scan&Snap and JoMA



A General Formulation:

$$\boldsymbol{h} = \phi(W^\top \boldsymbol{f}) \qquad \boldsymbol{f} = \psi(U(X\boldsymbol{b} + \boldsymbol{x}_T)) \qquad \boldsymbol{b} = \sigma(X^\top Z \boldsymbol{x}_T) \in \mathbb{R}^T$$

Notations

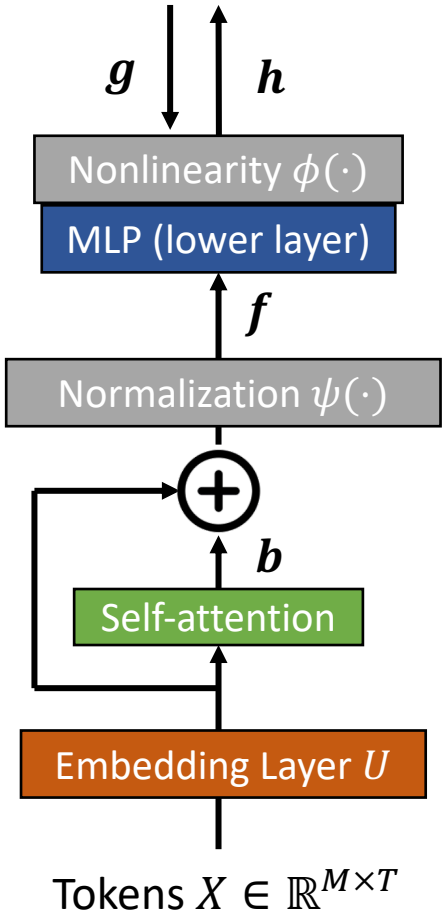
1. Input tokens $X \in \mathbb{R}^{M\times T}$
2. Pairwise attention logits $Z \in \mathbb{R}^{M\times M}$

Per-layer objective $J = \boldsymbol{g}^\top \boldsymbol{h}$
We can get gradient dynamics via computing differential $\mathrm{d}J$

Common assumptions

1. W and Z are trainable parameters
2. U is fixed and column orthogonal
3. $X = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T] \in \mathbb{R}^{M\times T}$ contains one-hot column vectors

Comparison between Scan&Snap and JoMA



Scan&Snap

1. $\psi = \ell_2$ normalization
2. $\phi =$ linear
3. $\boldsymbol{b} =$ softmax
4. $\boldsymbol{g}$ from cross-entropy
5. W learns much faster than Z

JoMA

1. $\psi =$ id (no normalization),
2. $\phi =$ linear or nonlinear (homogenous, e.g., ReLU)
3. $\boldsymbol{b} = \exp(\mathrm{X}^\top Z x_T)/A$
4. $\boldsymbol{g}$ constant

JoMA derivation in a few lines

$$\psi = \mathrm{id}, \boldsymbol{g}' \coloneqq \phi' \boldsymbol{g}, V \coloneqq U^\top W$$

$$\dot{V} = (X\boldsymbol{b} + \boldsymbol{x}_T)\boldsymbol{g}'^\top$$
$$\dot{Z} = X\mathrm{diag}(b)X^\top V \boldsymbol{g}' \boldsymbol{x}_T^\top$$

X's columns are one-hot vectors

$$\left(\dot{V} \circ V\right)1\boldsymbol{x}_T^\top = \dot{Z} + \mathrm{diag}(\boldsymbol{x}_T)V\boldsymbol{g}'\boldsymbol{x}_T^\top$$

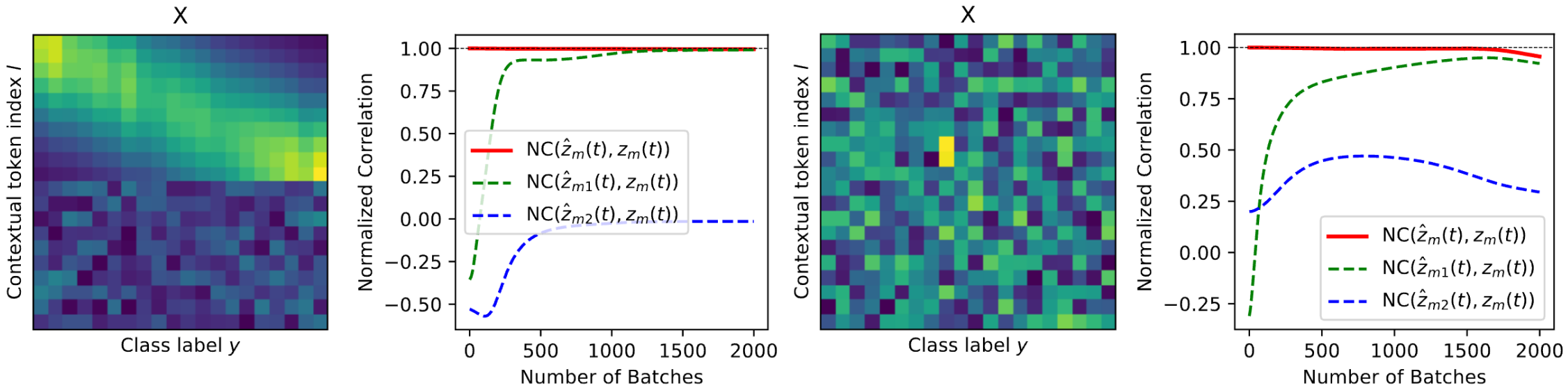
$x_T = m$

$$\sum_k \dot{\boldsymbol{v}}_k \circ \boldsymbol{v}_k = \dot{\boldsymbol{z}}_m + [0, 0, V[m, :]\boldsymbol{g}', 0]^\top$$

$z_{mm} = 0$
$V(0)_m = 0$

$$\sum_k \dot{\boldsymbol{v}}_k \circ \boldsymbol{v}_k = \dot{\boldsymbol{z}}_m \text{ (JoMA Thm. 1)}$$

Verification of JoMA dynamics



$\boldsymbol{z}_m(t)$: Real attention logits

$\hat{\boldsymbol{z}}_m(t)$: Estimated attention logits by JoMA

$$\hat{\boldsymbol{z}}_m(t) = \underbrace{\frac{1}{2}\sum_k \boldsymbol{v}_k^2(t)}_{\hat{\boldsymbol{z}}_{m1}(t)} \underbrace{- \|\boldsymbol{v}_k(t)\|_2^2 \overline{\boldsymbol{b}}_m}_{\hat{\boldsymbol{z}}_{m2}(t)} + \boldsymbol{c}$$

JoMA for Linear Activation

	Linear
$\dot{\boldsymbol{v}} = \boldsymbol{\Delta}_m \circ \exp\left(\frac{\boldsymbol{v}^2}{2}\right)$	Modified MLP (lower layer)

$$\Delta_{lm} = \mathbb{E}[g|l,m]\mathbb{P}[l|m]$$ Discriminancy × CoOccurrence

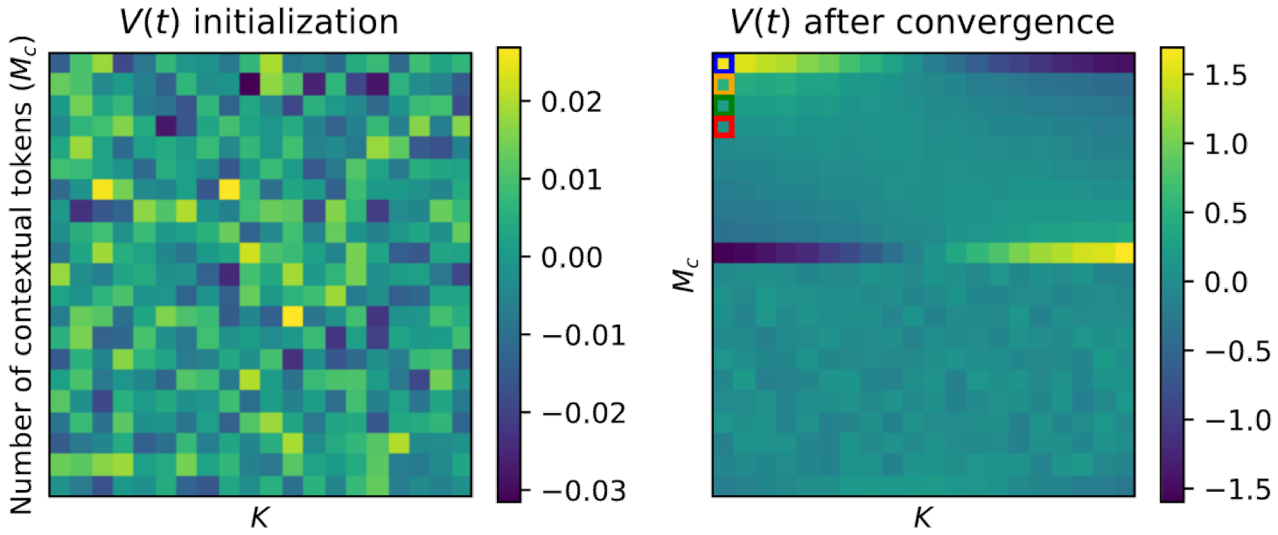
We can prove $\dfrac{\mathrm{erf}(v_l(t)/2)}{\Delta_{lm}} = \dfrac{\mathrm{erf}(v_{l'}(t)/2)}{\Delta_{l'm}}$

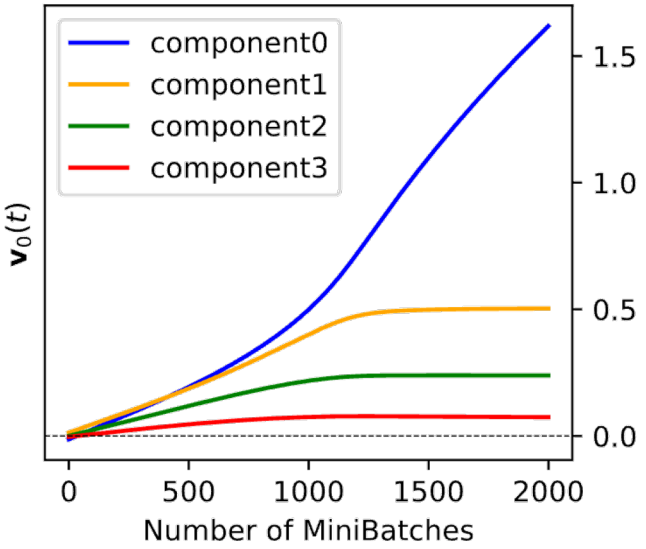
$$\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}\mathrm{d}t \in [-1,1]$$

Only one component $l^* = \mathrm{argmax}\,|\Delta_{lm}|$ of $\boldsymbol{v}$ goes to $+\infty$, other components stay finite.

JoMA for Linear Activation

	Linear
$\dot{\boldsymbol{v}} = \boldsymbol{\Delta}_m \circ \exp\left(\frac{\boldsymbol{v}^2}{2}\right)$	Modified MLP (lower layer)





Attention becomes sparser (Consistent with Scan&Snap)

JoMA for nonlinear activation

	Nonlinear
$\dot{\boldsymbol{v}} = (\boldsymbol{\mu} - \boldsymbol{v}) \circ \exp\left(\frac{\boldsymbol{v}^2}{2}\right)$	Modified MLP (lower layer)

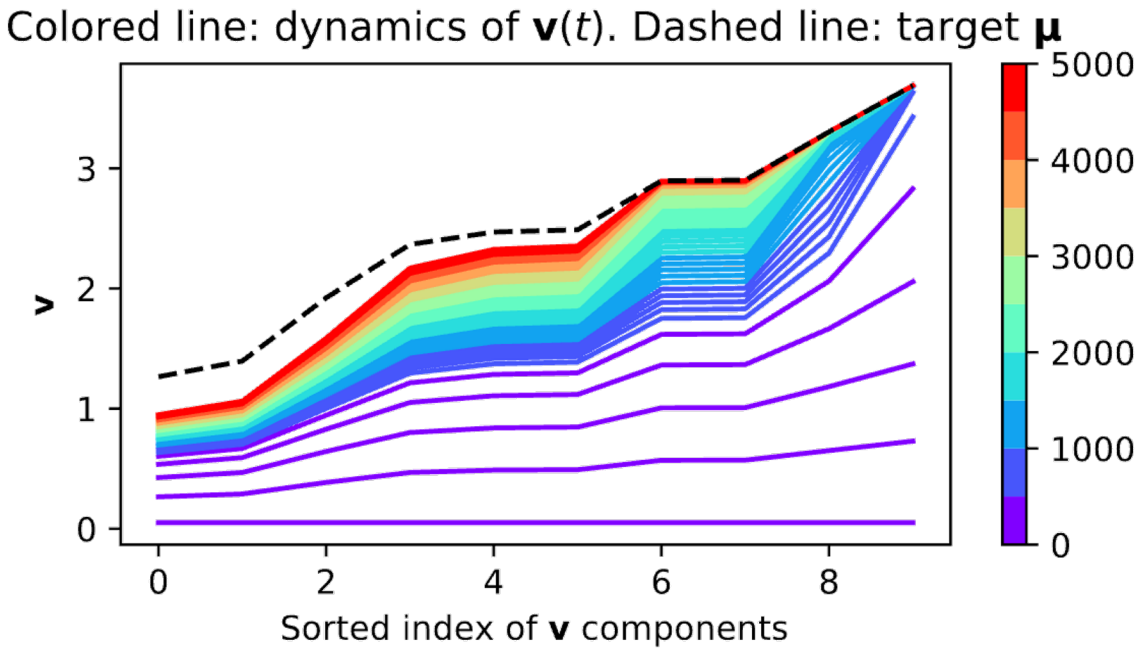
$\boldsymbol{\mu} \sim \boldsymbol{\Delta}_m$: Critical points that emerges from the presence of nonlinearity

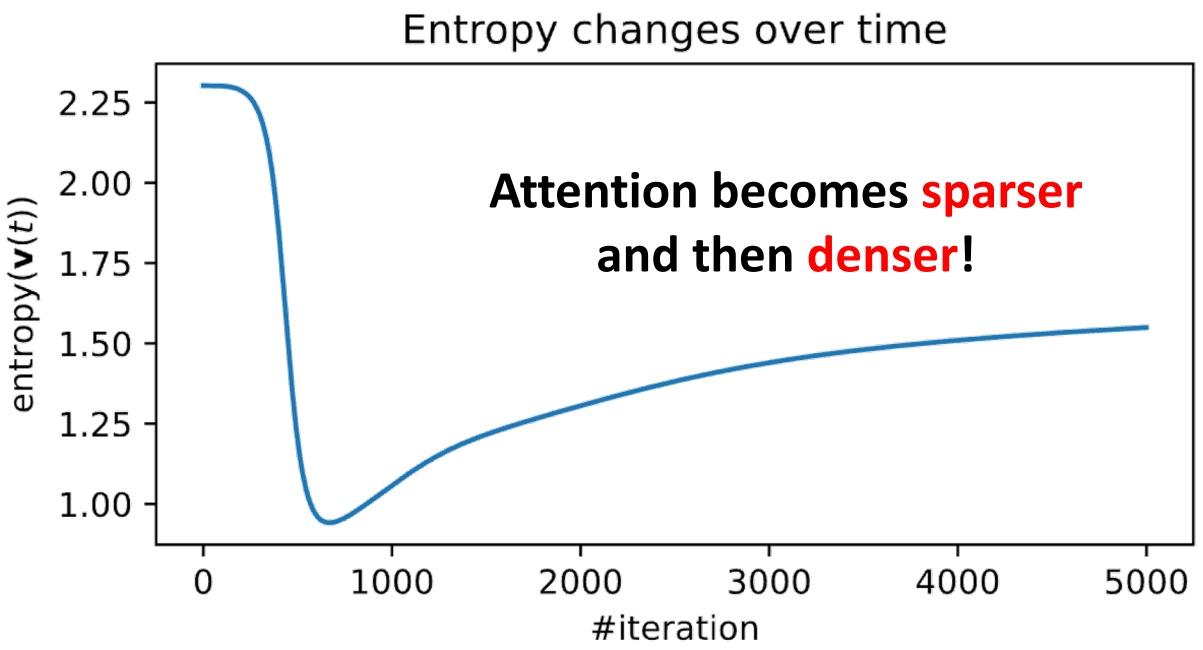
What does the dynamics look like?

JoMA for nonlinear activation

	Nonlinear
$\dot{\boldsymbol{v}} = (\boldsymbol{\mu} - \boldsymbol{v}) \circ \exp\left(\frac{\boldsymbol{v}^2}{2}\right)$	Modified MLP (lower layer)

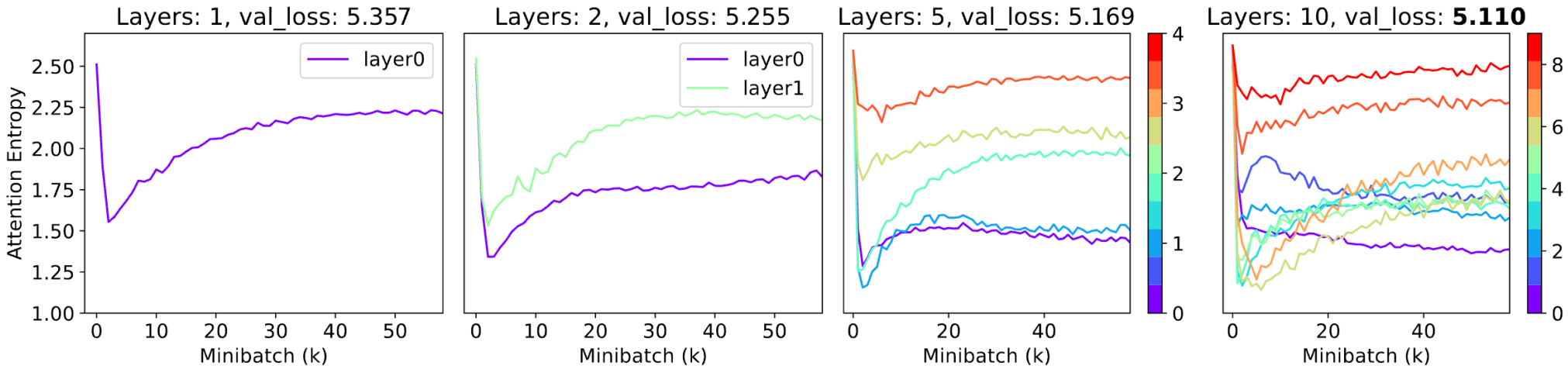
$\boldsymbol{\mu} \sim \boldsymbol{\Delta}_m$: Critical points that emerges from the presence of nonlinearity



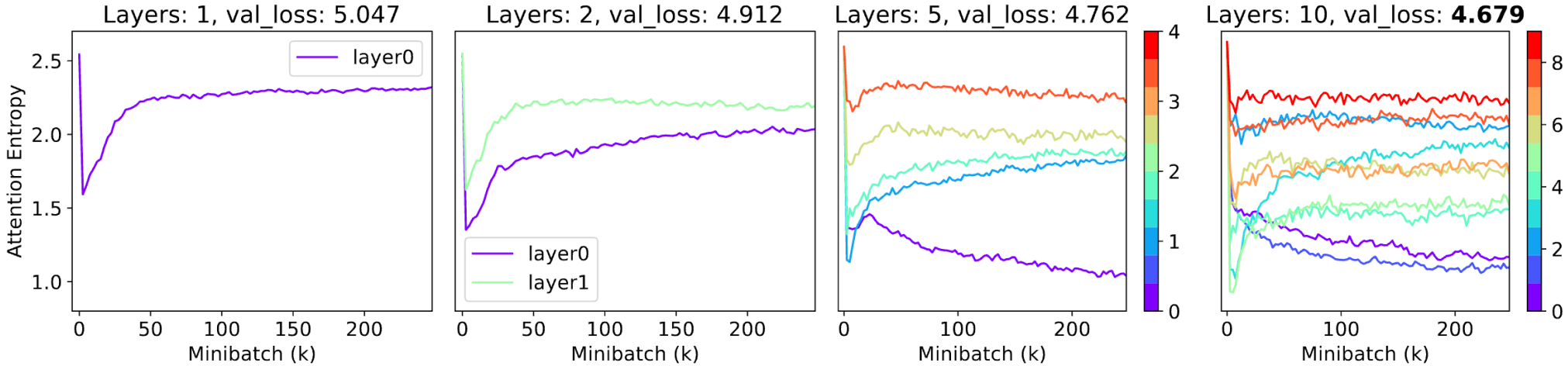


Real-world Experiments

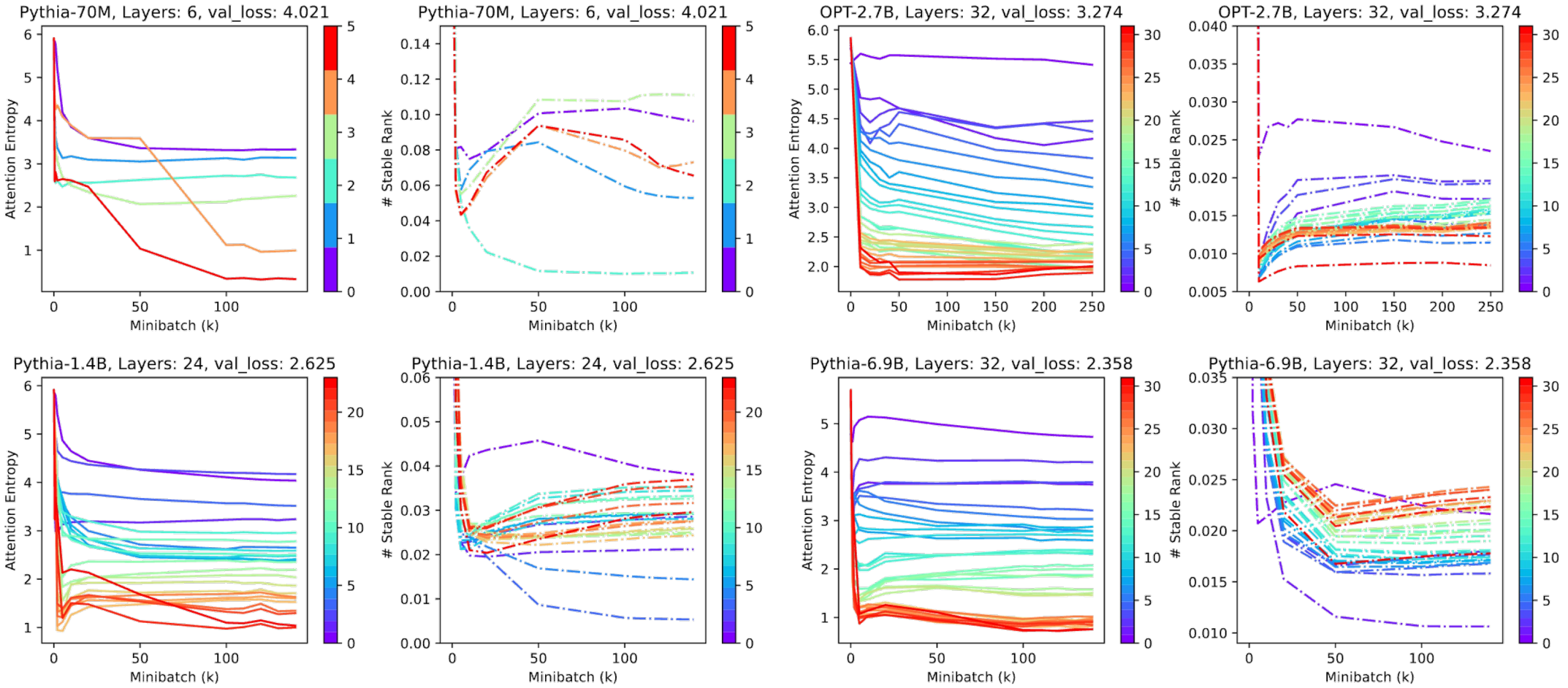
Wikitext2



Wikitext103



Real-world Experiments



Stable Rank of the lower layer of MLP shows the “bouncing back” effects as well.

JoMA for nonlinear activation

	Nonlinear
$\dot{\boldsymbol{v}} = (\boldsymbol{\mu} - \boldsymbol{v}) \circ \exp\left(\frac{\boldsymbol{v}^2}{2}\right)$	Modified MLP (lower layer)

Why? Convergence speed of salient/non-salient components are very different.

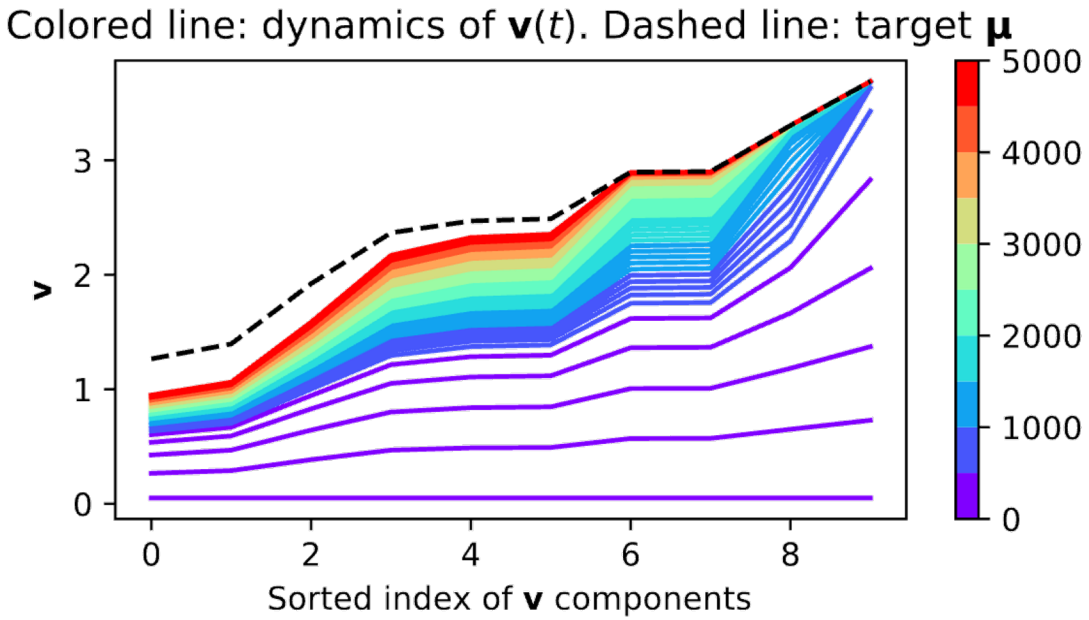
$$\frac{\text{ConvergenceRate}(j)}{\text{ConvergenceRate}(k)} \sim \frac{\exp\left(\mu_j^2/2\right)}{\exp(\mu_k^2/2)}$$

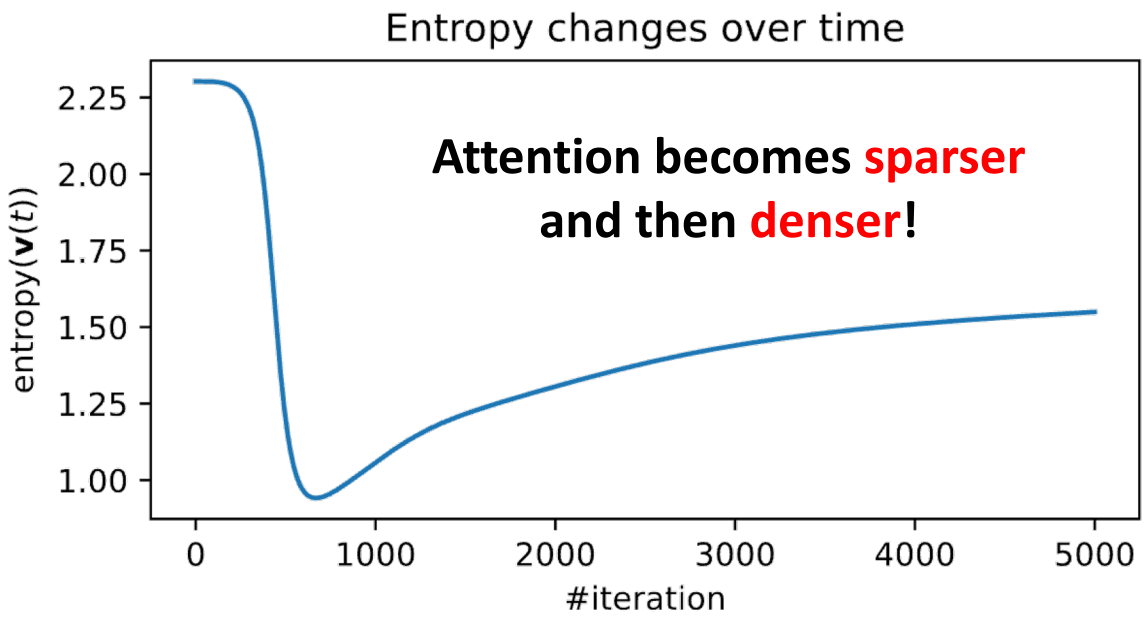
$$\text{ConvergenceRate}(j) \coloneqq \ln 1/\delta_j(t)$$
$$\delta_j(t) \coloneqq 1 - v_j(t)/\mu_j$$

JoMA for nonlinear activation

	Nonlinear
$\dot{\boldsymbol{v}} = (\boldsymbol{\mu} - \boldsymbol{v}) \circ \exp\left(\frac{\boldsymbol{v}^2}{2}\right)$	Modified MLP (lower layer)

Why? Convergence speed of salient/non-salient components are very different.



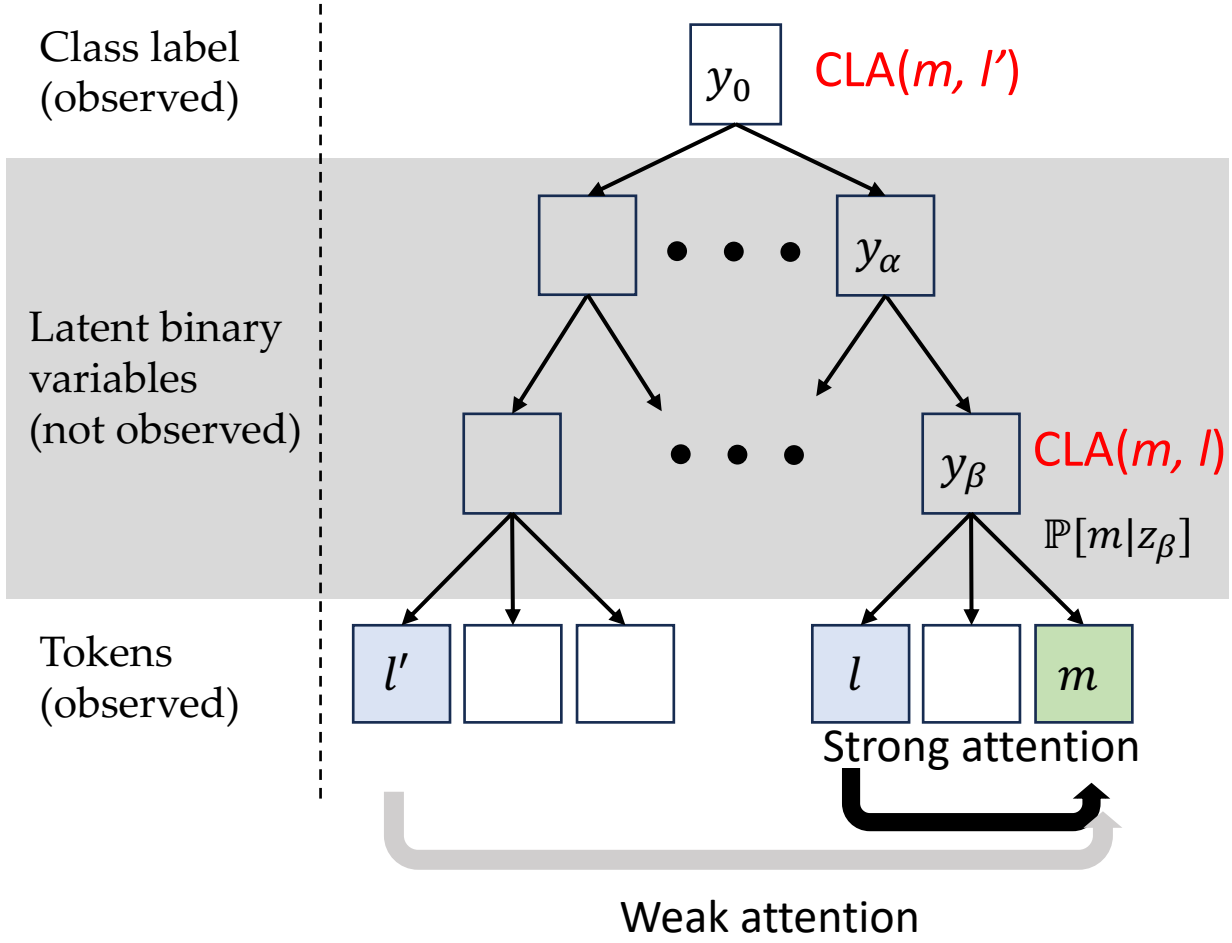


Why is this “bouncing back” property useful?

It seems that it only slows down the training??

Not useful in 1-layer, but useful in multiple Transformer layers!

Multilayer Transformer



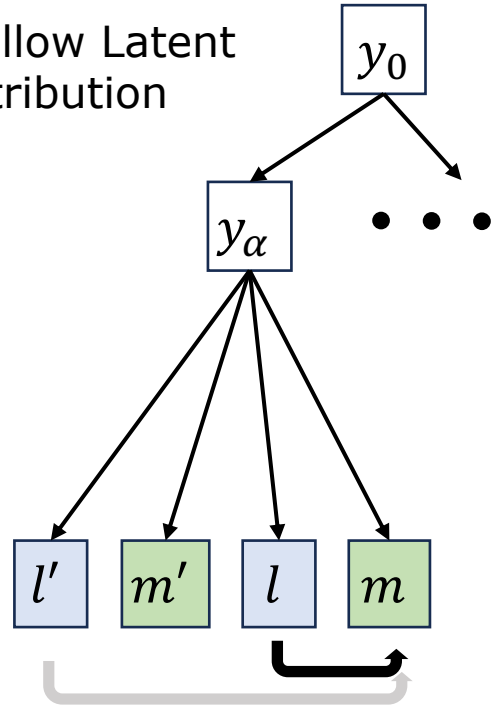
$$\mathbb{P}[l|m] \approx 1 - \frac{H}{L}$$

H: height of the common latent ancestor (CLA) of l & m

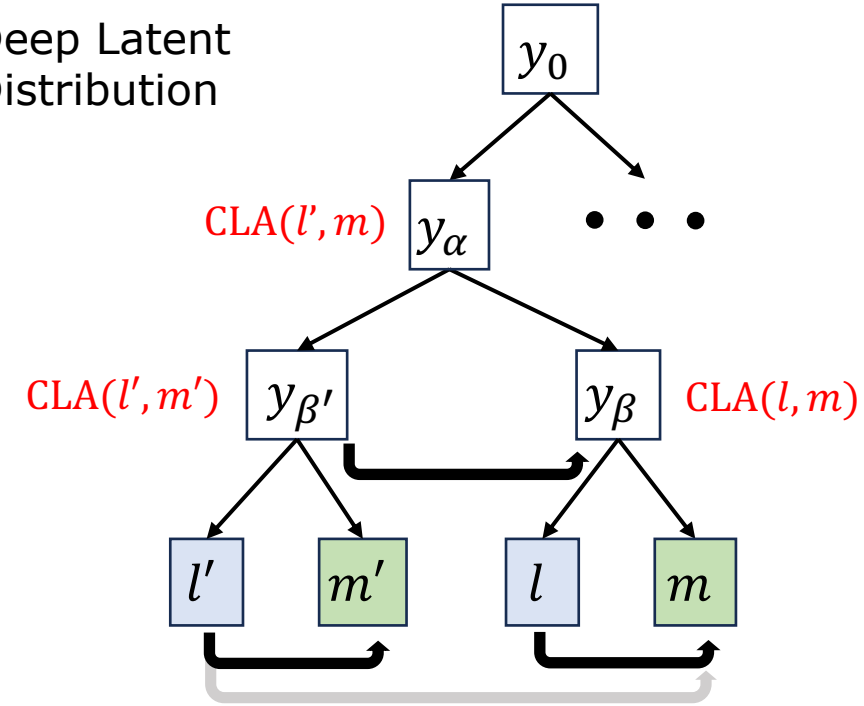
L: total height of the hierarchy

Multilayer Transformer

Shallow Latent Distribution



Deep Latent Distribution



Learning the current hierarchical structure by
slowing down the association of tokens that are not directly correlated

Future Work

- Embedding vectors
- Positional Encoding
- Formulate the dynamics of Multi-layer Transformers
  - How intermediate latent concept gets learned during training?
  - Why we need over-parameterization?

Pattern Superposition

The same neuron in MLP hidden layers can be activated by multiple irrelevant combinations of tokens

Pythia-160M

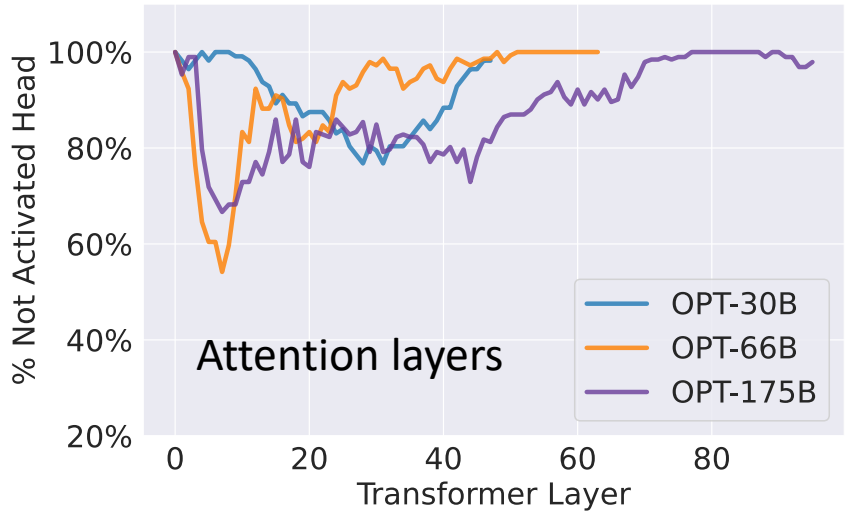
A	**Every morning**, as the city slowly awakens with the distant hum of traffic and the chirping of sparrows, John takes a moment to savor the peaceful ambiance before he walks his dog, Max, around the block, greeting familiar faces and enjoying the fresh air.
B	**In the realm of physics**, when water is subjected to a temperature of 100°C at one atmosphere of pressure, it undergoes a phase transition from liquid to gas, producing steam that has long been harnessed for various technological and culinary applications.
A	**The Sahara Desert**, stretching across North Africa, is the third largest desert in the world and is renowned for its vast sand dunes and scorching temperatures. Despite its harsh conditions, it\'s home to various unique species that have adapted to its extreme environment.
B	**Novels**, beyond their entertainment value, serve as mirrors to society, often reflecting cultural, social, and political nuances of their time. Authors like George Orwell and Jane Austen used their works to critique and provide insights into the world they lived in.

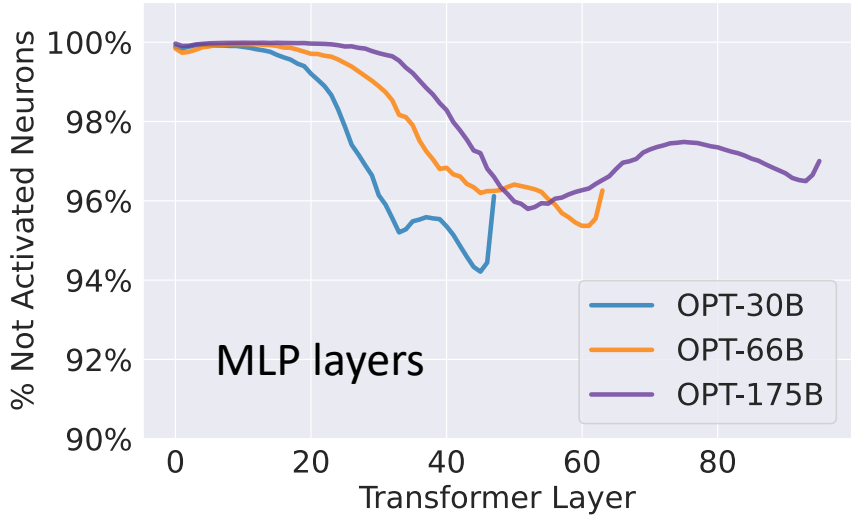
Pythia-70M

A	Cats **are known for their independent nature**. Many people appreciate them for their low-maintenance lifestyle, often content with just a comfortable spot to nap and an occasional playtime.
B	Rainforests **are vital for the Earth's ecosystem**. They provide a habitat for countless species, many of which are not found anywhere else. Additionally, they play a crucial role in regulating global climate and producing oxygen.
A	**The** Eiffel **Tower, an iconic landmark in** Paris, was originally constructed as a temporary exhibit for the 1889 World's Fair. Over the years, it has become a symbol of the city's romance and architectural prowess, attracting millions of tourists annually.
B	The human digestive system is a complex network of organs working together to break down food into essential nutrients. Beginning with the mouth and ending at the small intestine, each part plays **a crucial role in** ensuring our bodies receive the energy and vitamins needed for daily function.

Part II
Applications based on Attention Properties

Contextual Sparsity





Key Observation

Keeping only **high activation (contextual!)** in attention/MLP

- results in 85% structured sparsity
  - 80% attention, 95% MLP
- leads to 7× potential parameter reduction for each input
- maintains same accuracy

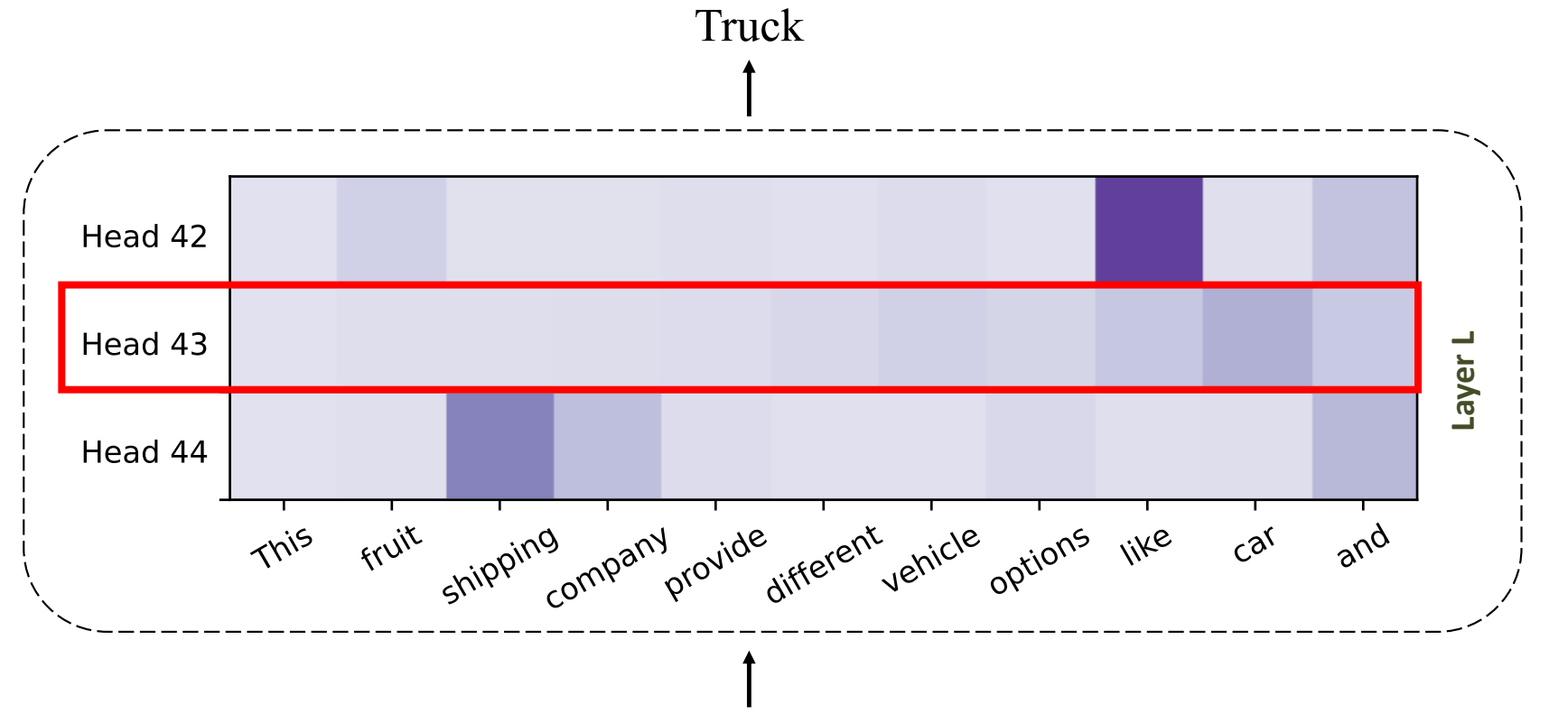
Contextual sparsity widely exists in pre-trained models,

e.g., OPT /LLaMA /Bloom/GPT

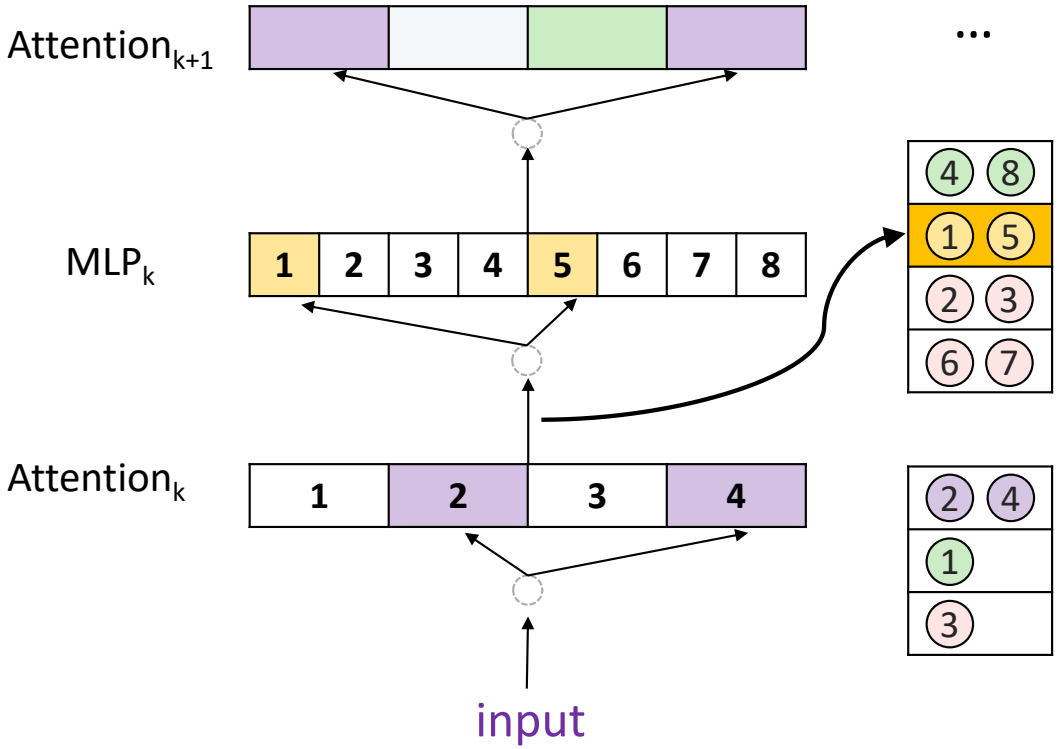
[Z. Liu et al, Deja vu: Contextual sparsity for efficient LLMs at inference time, *ICML'23 (oral)]*

Example: Contextual Sparsity in Attention

Head 43 outputs (almost) uniform attention score.



Proposed idea: **Predicting** Contextual Sparsity



Challenge:

How to predict high activation on-the-fly without computing the full attention or MLP?

Benefits:

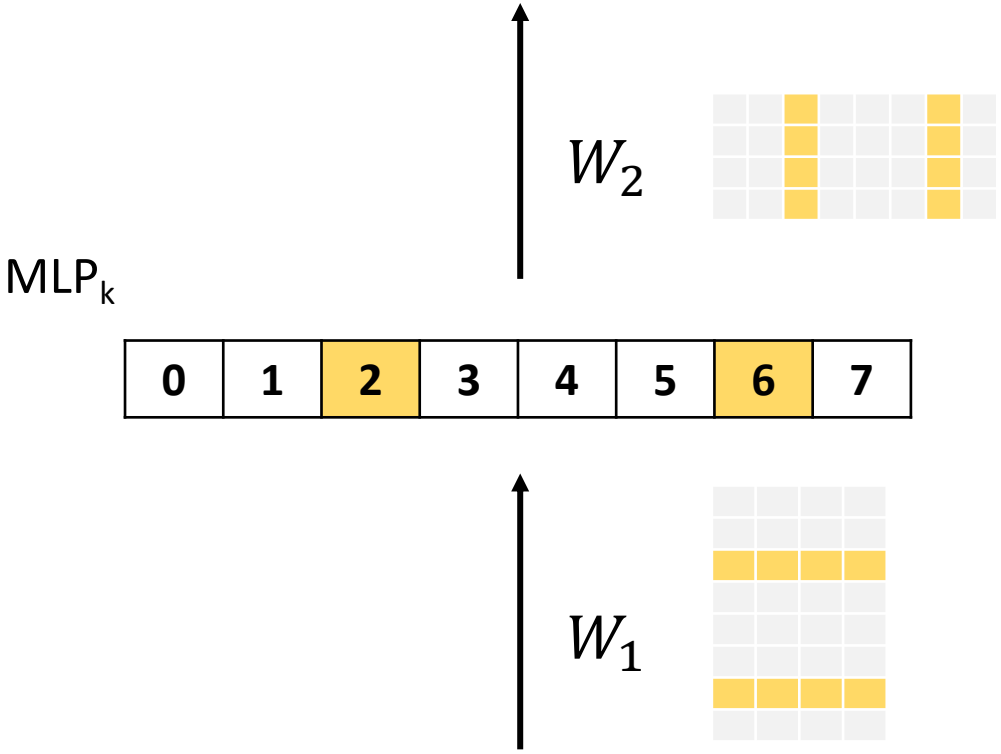
Only load the desired set of parameters

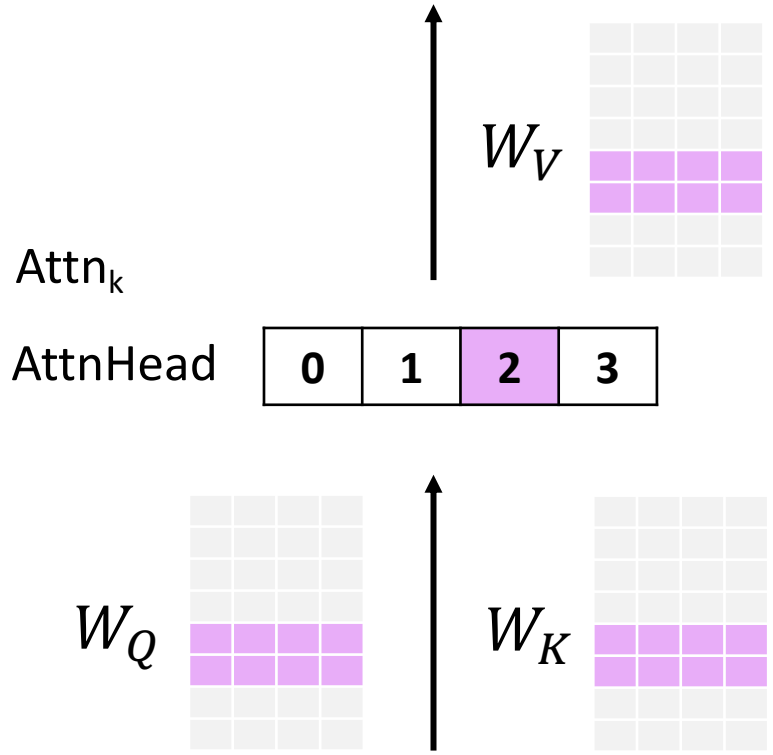
(save cache memory!)

[Z. Liu et al, Deja vu: Contextual sparsity for efficient LLMs at inference time, *ICML'23 (oral)]*

Proposed idea: **Predicting** Contextual Sparsity

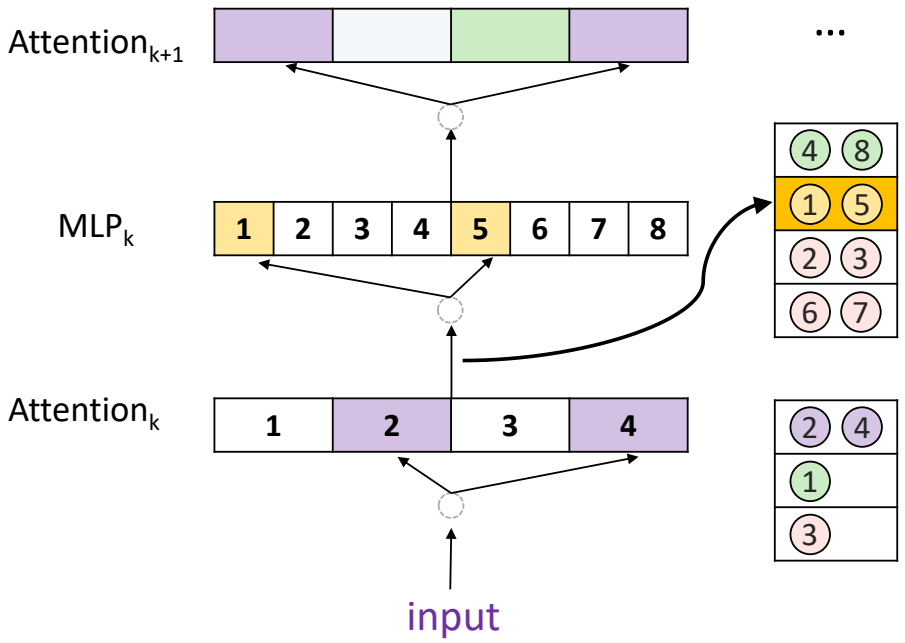
Given the input, predicting which MLP hidden nodes are activated / which attention heads will be used.





Just need to load $W_1[[2,6],:]$ and $W_2[:,[2,6]]$

Proposed idea: **Predicting** Contextual Sparsity

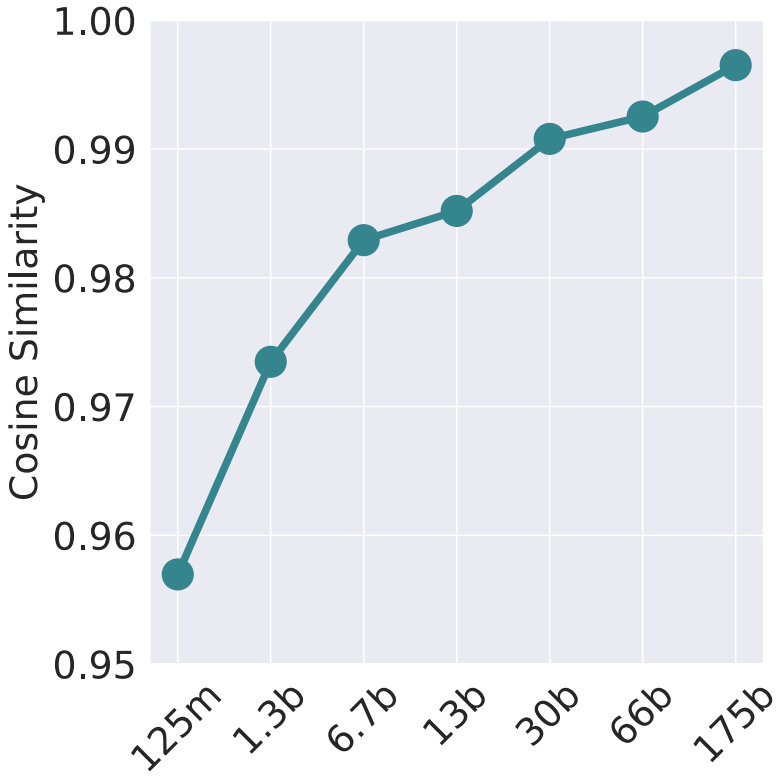


Key idea: design a "similarity"-based prediction

Formulate the prediction problem as nearest-neighbor search (NNS).

NNS algorithms can make prediction based on the similarity between input & parameters.

Slowly Changing Embeddings across Layers



Challenge: how to reduce prediction overhead?

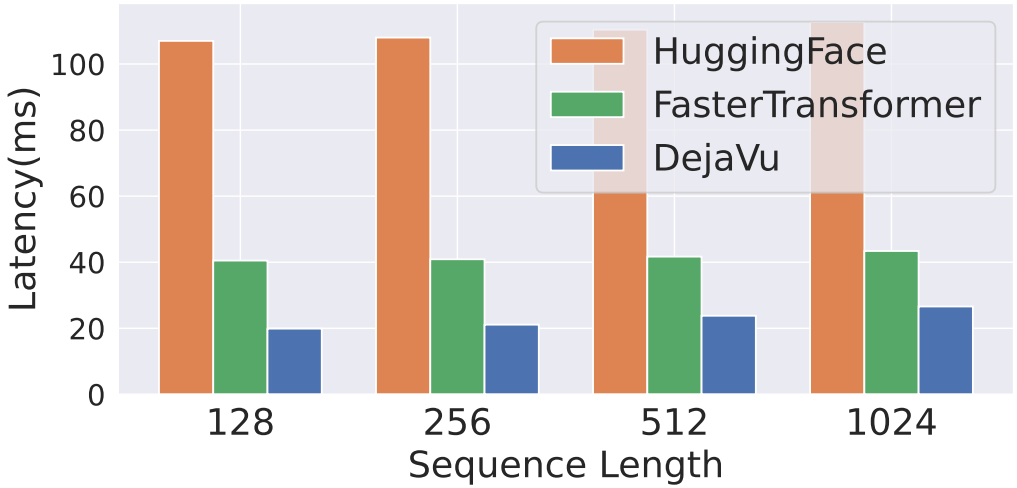
Key insight: cosine similarity between embeddings at consecutive layers is very **high**.

Substantial Engineering Efforts:

- async prediction
- low-cost small trainable MLP as predictors
- system optimization

Use the embeddings from previous layer(s) to asynchronously make the prediction.

Deja Vu: 2X FasterTransformer and 6X HuggingFace



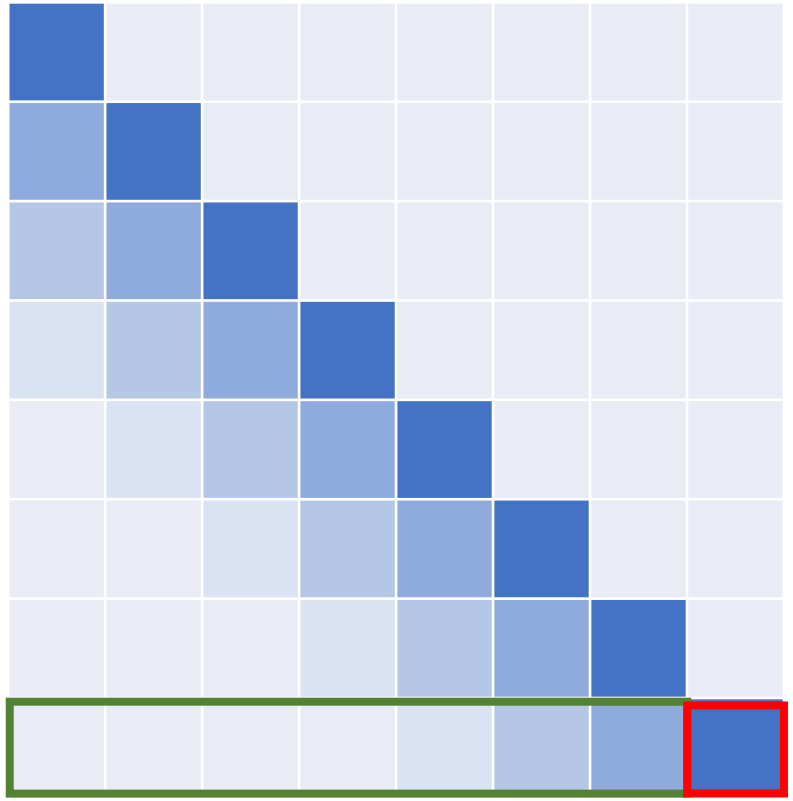
	COPA	OpenBookQA	Winogrande	Lambada
OPT-175B	0.86	0.446	0.726	0.758
Deja Vu-OPT-175B	0.85	0.45	0.726	0.753
OPT-175B + W4A16	0.85	0.44	0.714	0.757
Deja Vu-OPT-175B + W4A16	**0.86**	**0.452**	**0.726**	**0.754**

- Demonstrates best performance with batch size=1, ReLU, 175B model
- Maintains accuracy even combined with quantization.
- Achieves speed up with larger batch size, more activation functions, and smaller models.

[Z. Liu et al, Deja vu: Contextual sparsity for efficient LLMs at inference time, *ICML'23 (oral)]*

H2O: Leverage attention sparsity for fast inference

Inference in Vanilla Transformers is $O(n^2)$



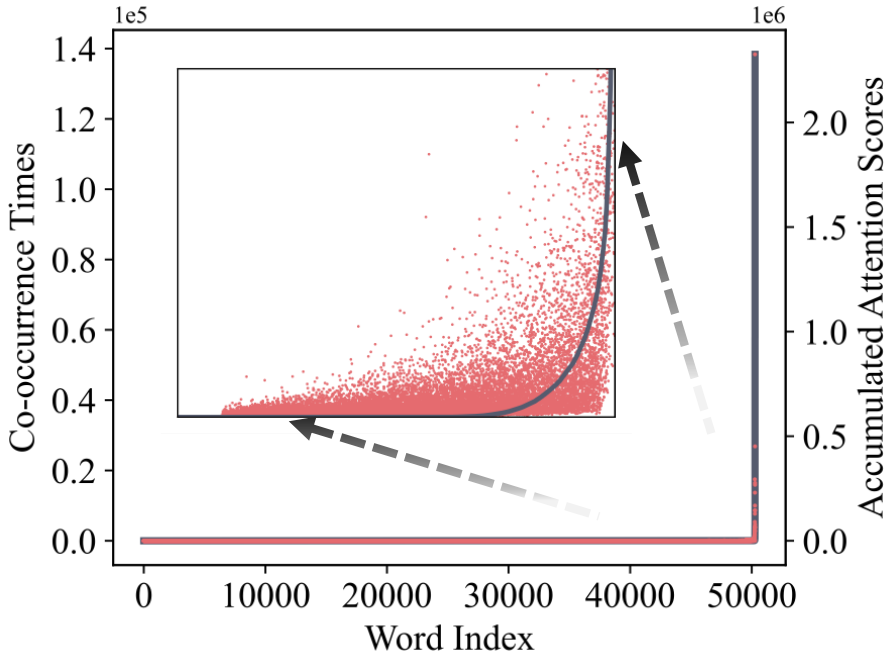
In order to generate this token,
do we really need to store all previous tokens?

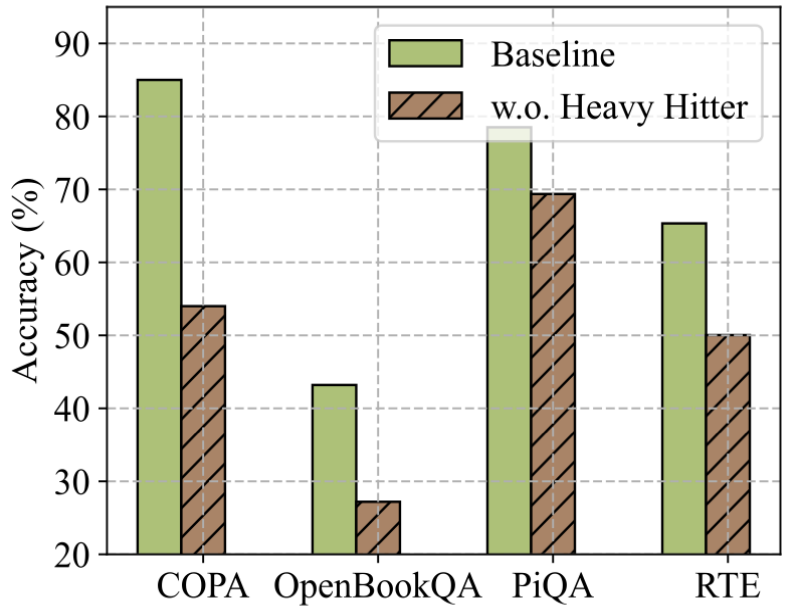
The answer is **No**

[*Z. Zhang et al*, H2O: Heavy-Hitter Oracle for Efficient Generative Inference of Large Language Models, *NeurIPS'23*]

Finding “Heavy-Hitters” in Attention

Challenge: how to evict tokens? Once evicted, future tokens can no longer attend to it

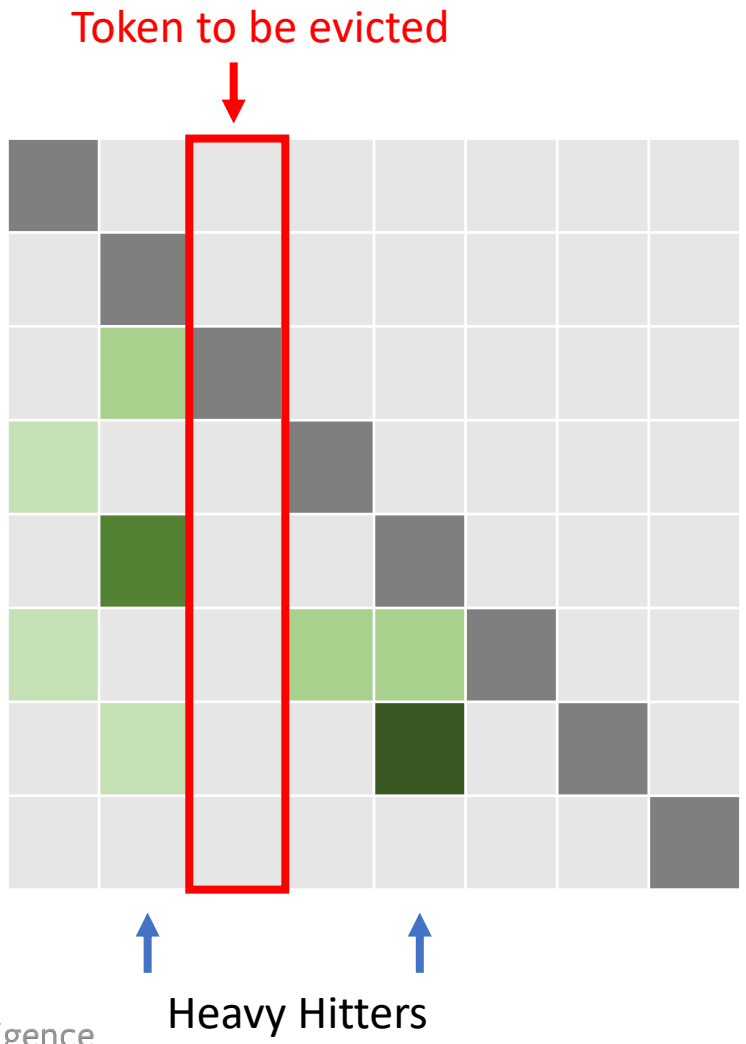




Key Observation: a small set of tokens are important along the generation

- accumulated attention scores of all the tokens follow a power-law distribution
- masking heavy-hitter tokens degrades model quality

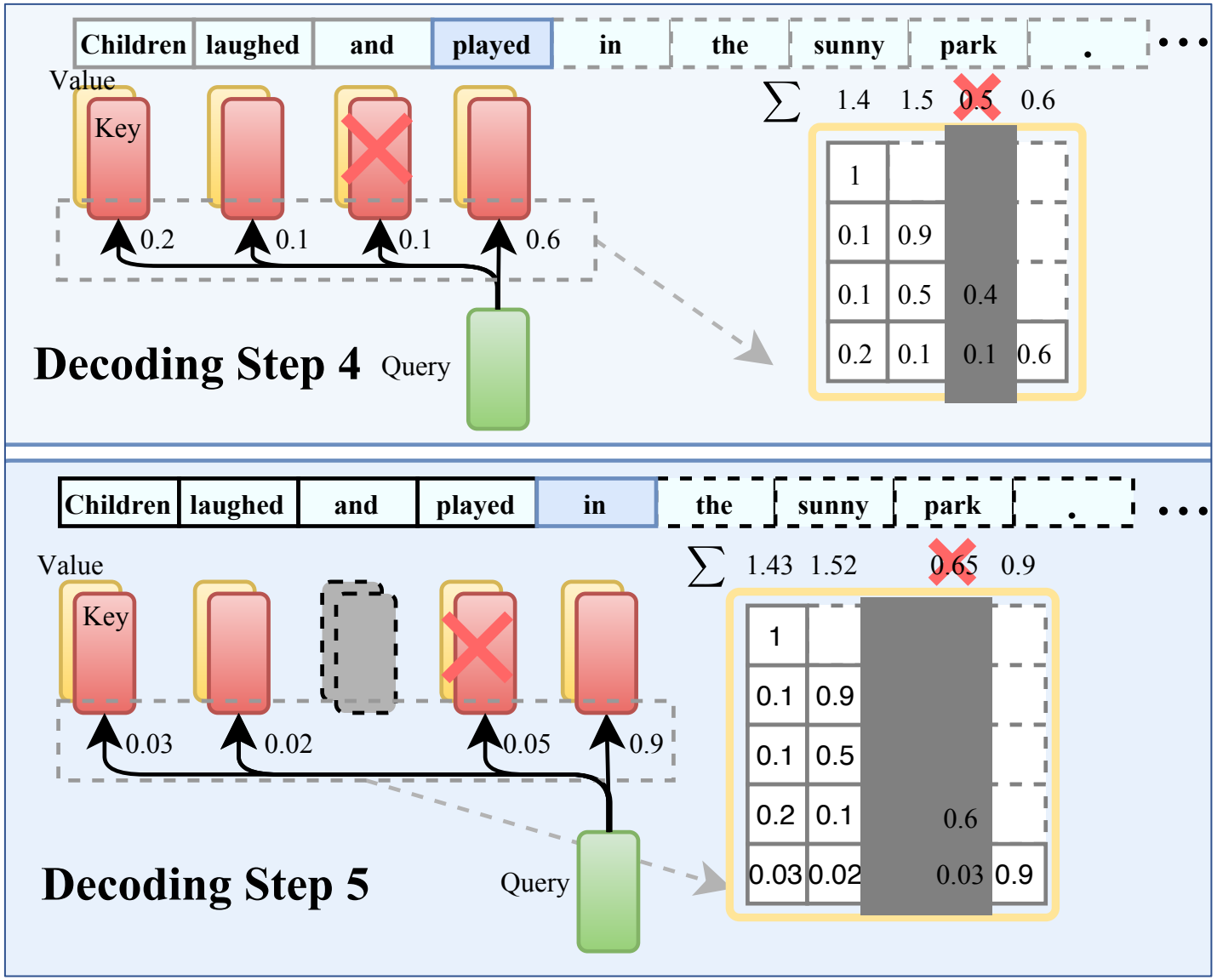
Greedy Algorithm to Pick Heavy Hitters



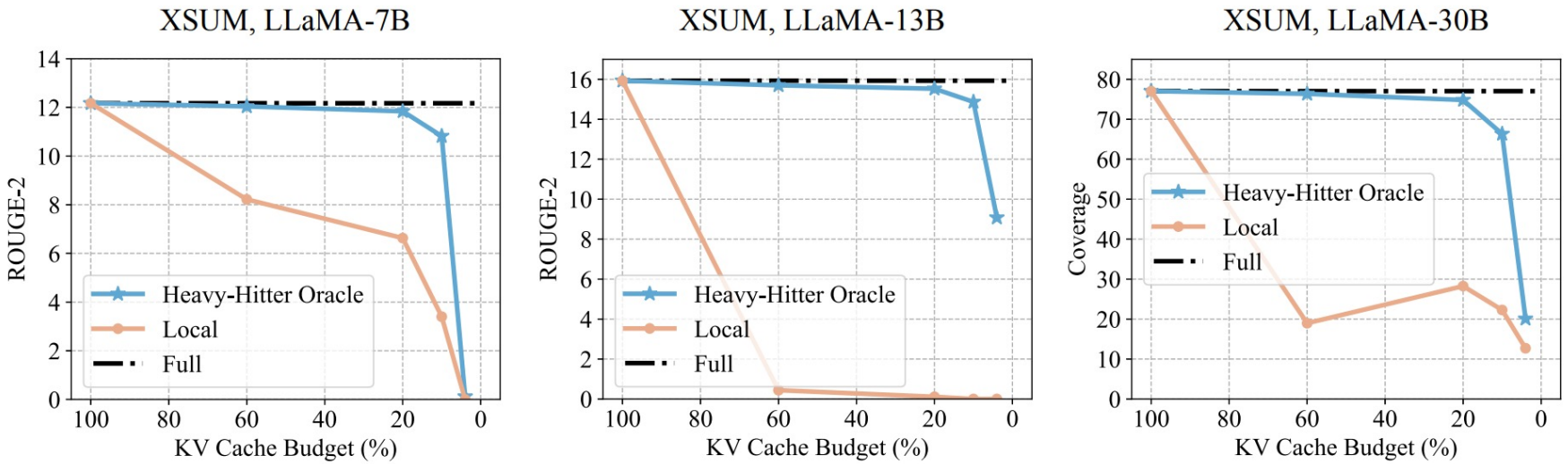
Local greedy algorithm

- sum up the attention scores of the previous tokens every decoding step
- Add local / recent tokens

H2O: Heavy Hitter Oracle



H2O: 3-29X Throughput and 1.9X Latency



A100 GPU	FlexGen	H_2O
Throughput (token/s)	494	918 (1.9X)
Latency (s)	99	53 (1.9X)

T4 GPU	Hugging Face	Deep Speed	FlexGen	H_2O
Throughput token/s	0.6	0.6	8.5	18.83 (3-29X)

- compatible with quantization
- generate sentences with less repetition + more creativity

Model Input

In a small, bustling cafe nestled in the heart of a vibrant city, a serendipitous event unfolded, leaving a lasting impression on all who witnessed it. As the patrons sat sipping their coffees and engaging in animated conversations, a talented street musician entered the cafe, carrying a weathered guitar and radiating an aura of creativity.

LLaMA-7B Full Cache Output

He began to play, and the patrons were captivated. The musician's performance was so moving that the patrons began to applaud, and the musician was so moved that he began to cry. The patrons were so moved that they began to cry, and the musician was so

LLaMA-7B Local 20% Cache Output

He (((((((((((((, [)), 1999, 2000, 2001, 1, and, and, and, and, and, and, and, and, and, and, and, and,

LLaMA-7B H_2O 20% Cache Output

He began to play, and the room was filled with the sound of his music. The patrons of the cafe were enthralled by the music, and the atmosphere was electric. The cafe was packed with people, all of whom were enjoying the music. The musician was a young

less repetition + more creativity

Model Input

In a small, bustling cafe nestled in the heart of a vibrant city, a serendipitous event unfolded, leaving a lasting impression on all who witnessed it. As the patrons sat sipping their coffees and engaging in animated conversations, a talented street musician entered the cafe, carrying a weathered guitar and radiating an aura of creativity.

LLaMA-7B Full Cache Output

He began to play, and the patrons were captivated. The musician's performance was so moving that the patrons began to applaud, and the musician was so moved that he began to cry. The patrons were so moved that they began to cry, and the musician was so

What are these heavy hitters?

Where are they?

LLaMA-7B Local 20% Cache Output

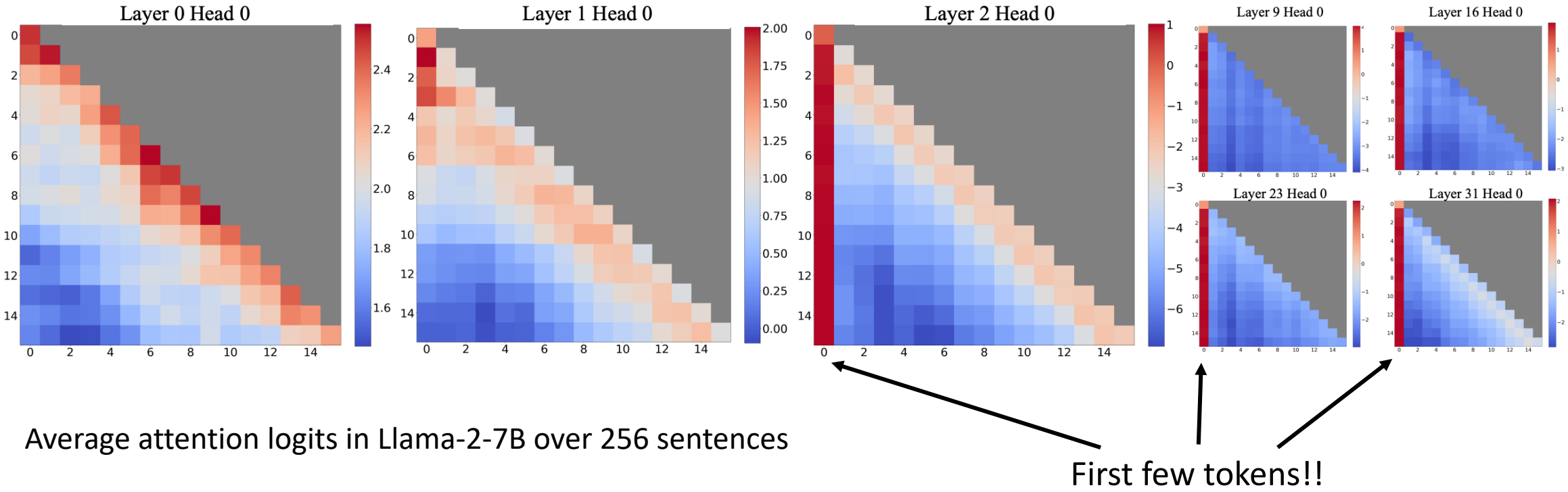
He (((((((((((((, [)), 99, 100, 2019, and, and, and, and, and, and, and, and, and, and, and,

LLaMA-7B H_2O 20% Cache Output

He began to play, and the room was filled with the sound of his music. The patrons of the cafe were enthralled by the music, and the atmosphere was electric. The cafe was packed with people, all of whom were enjoying the music. The musician was a young

less repetition + more creativity

Attention Sinks: Initial tokens draw a lot of attentions



Average attention logits in Llama-2-7B over 256 sentences

- Observation: Initial tokens have large attention scores, even if they're not semantically significant.
- Attention Sink: Tokens that disproportionately attract attention irrespective of their relevance.

[G. Xiao et al, *Efficient Streaming Language Models with Attention Sinks,* ICLR'24]

Understanding Attention Sinks

- **Why?** Attention scores **have to** sum up to **1** for all contextual tokens. (*SoftMax-Off-by-One, Miller et al. 2023*)

$$\text{SoftMax}(x)_i = \frac{e^{x_i}}{e^{x_1} + \sum_{j=2}^{N} e^{x_j}}, \quad x_1 \gg x_j, j \in 2, \ldots, N$$

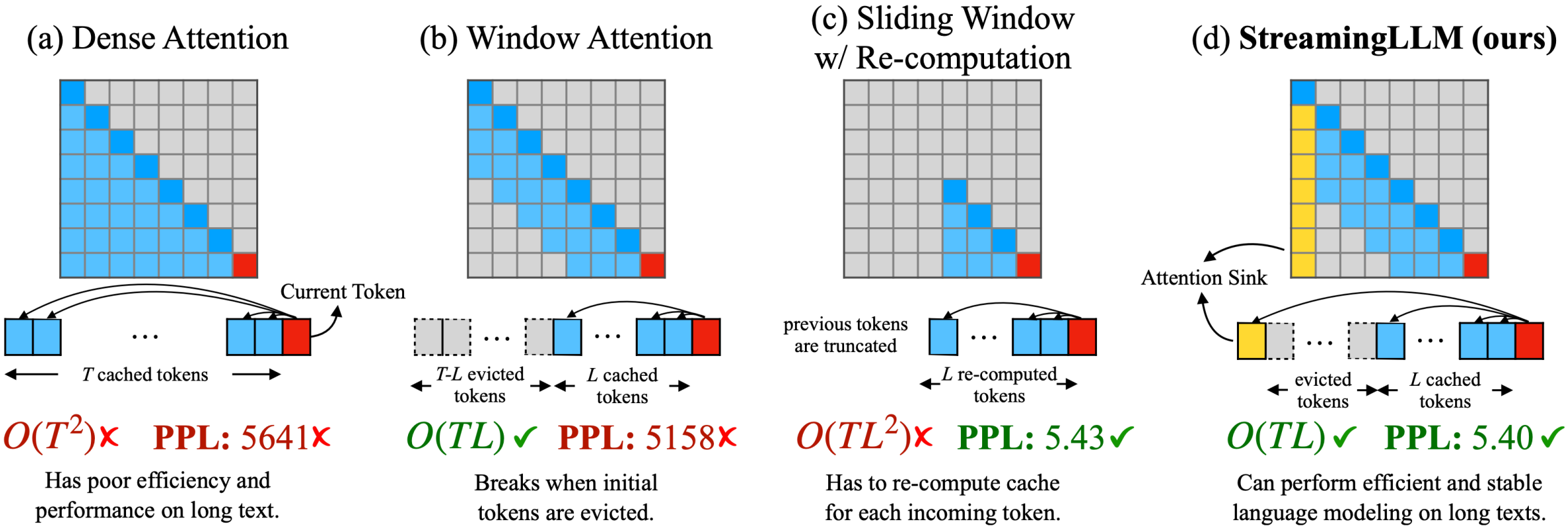
- **Why initial tokens?** Their visibility to subsequent tokens, rooted in autoregressive language modeling.

- The model learns a bias towards their **absolute position** rather than the semantics.

Llama-2-13B	PPL (↓)
0+1024 (window)	5158.07
4+1024	5.40
4"\n"+1020	5.6

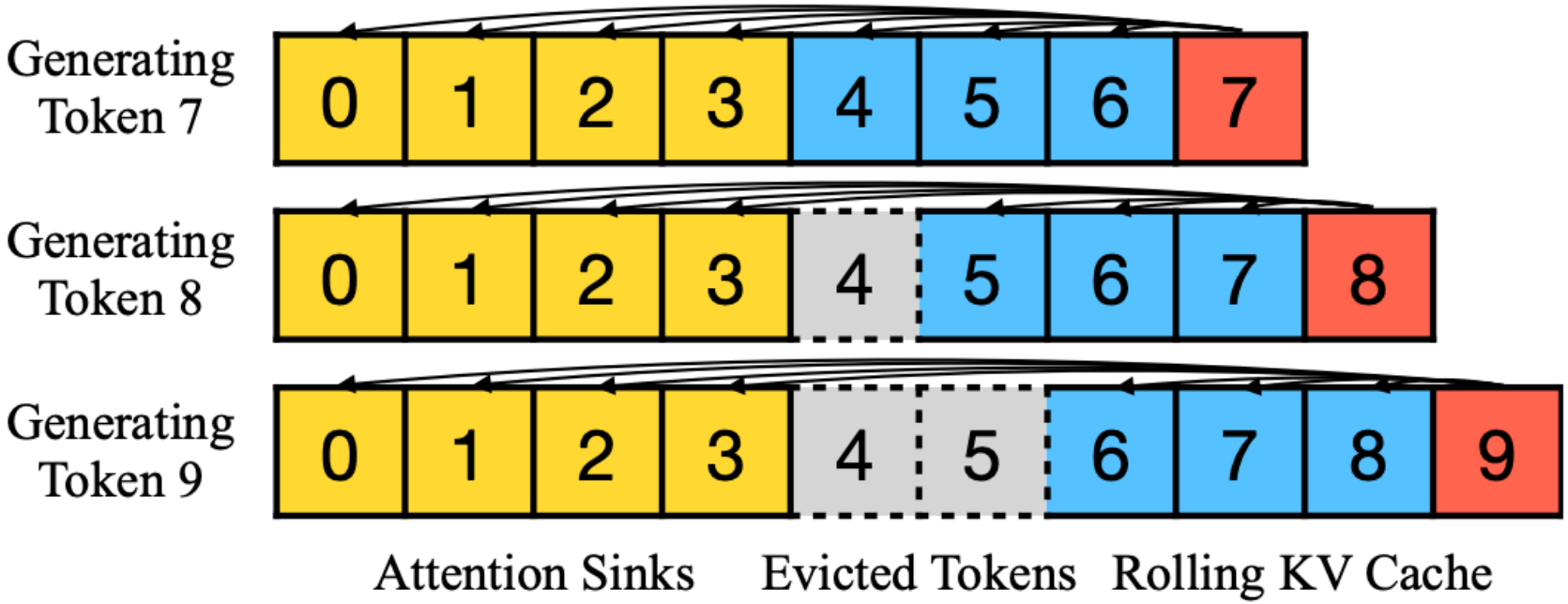
[G. Xiao et al, *Efficient Streaming Language Models with Attention Sinks,* ICLR'24]

StreamingLLM



[G. Xiao et al, *Efficient Streaming Language Models with Attention Sinks*, ICLR'24]

StreamingLLM

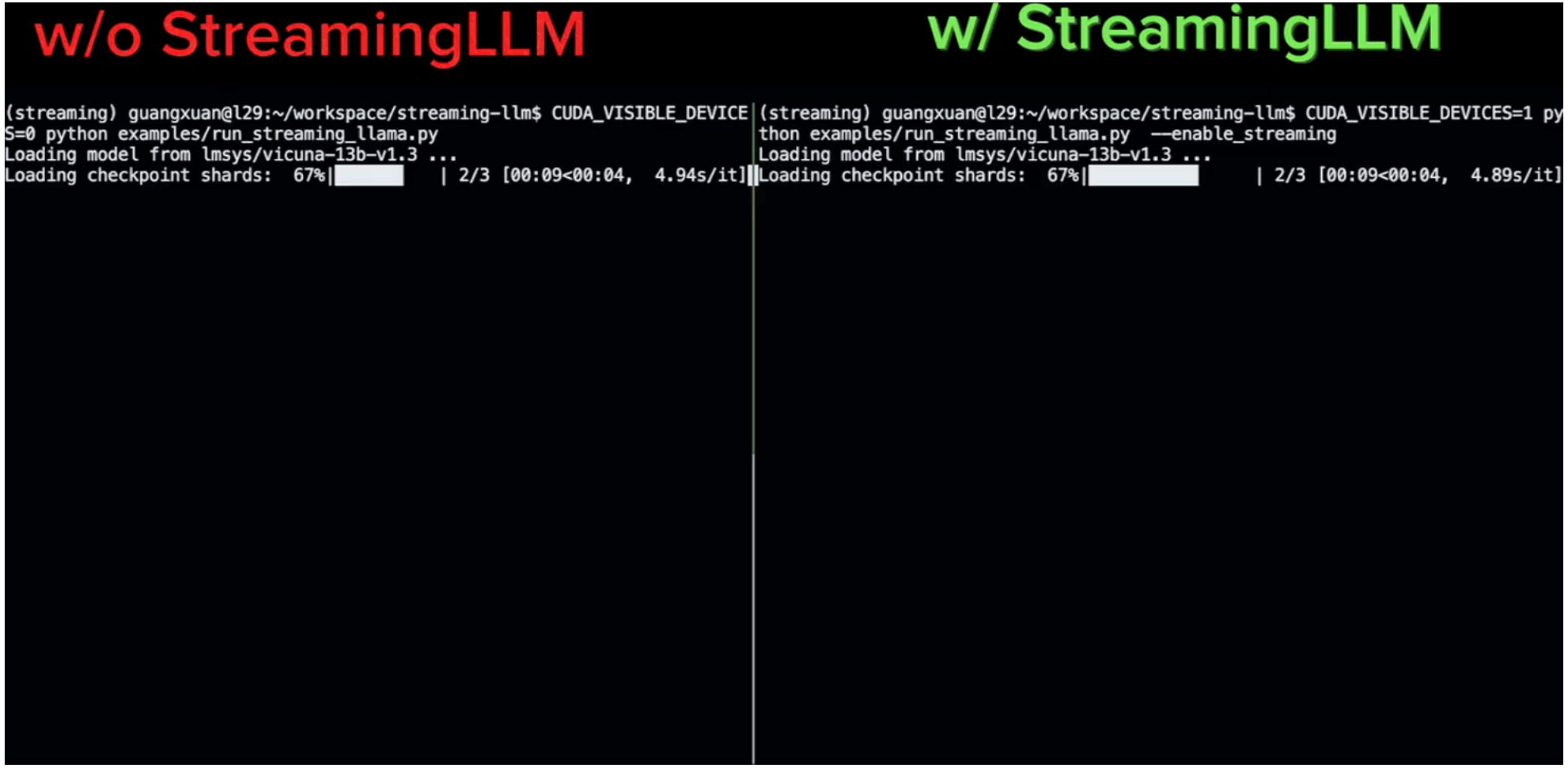


Key design: Position Rolling

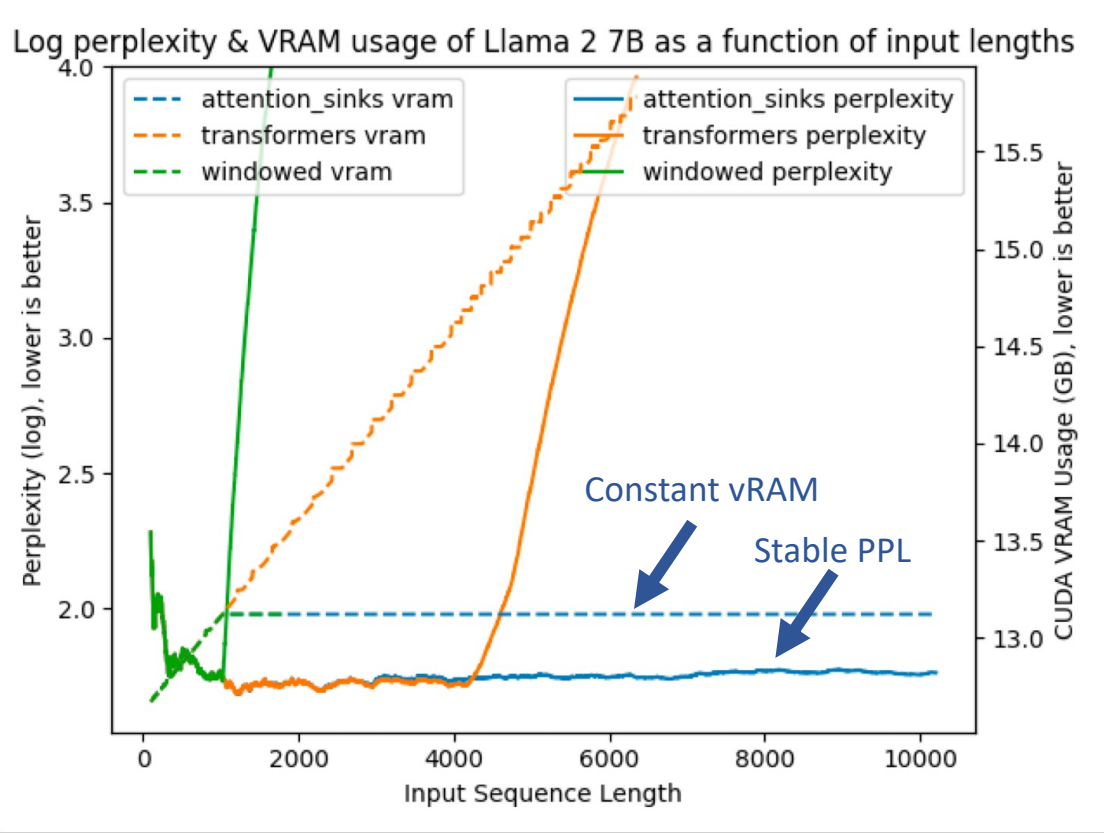
For all tokens, use their positions **within cache** to compute positional encoding!

→ Token distance never exceeds pre-trained context window!

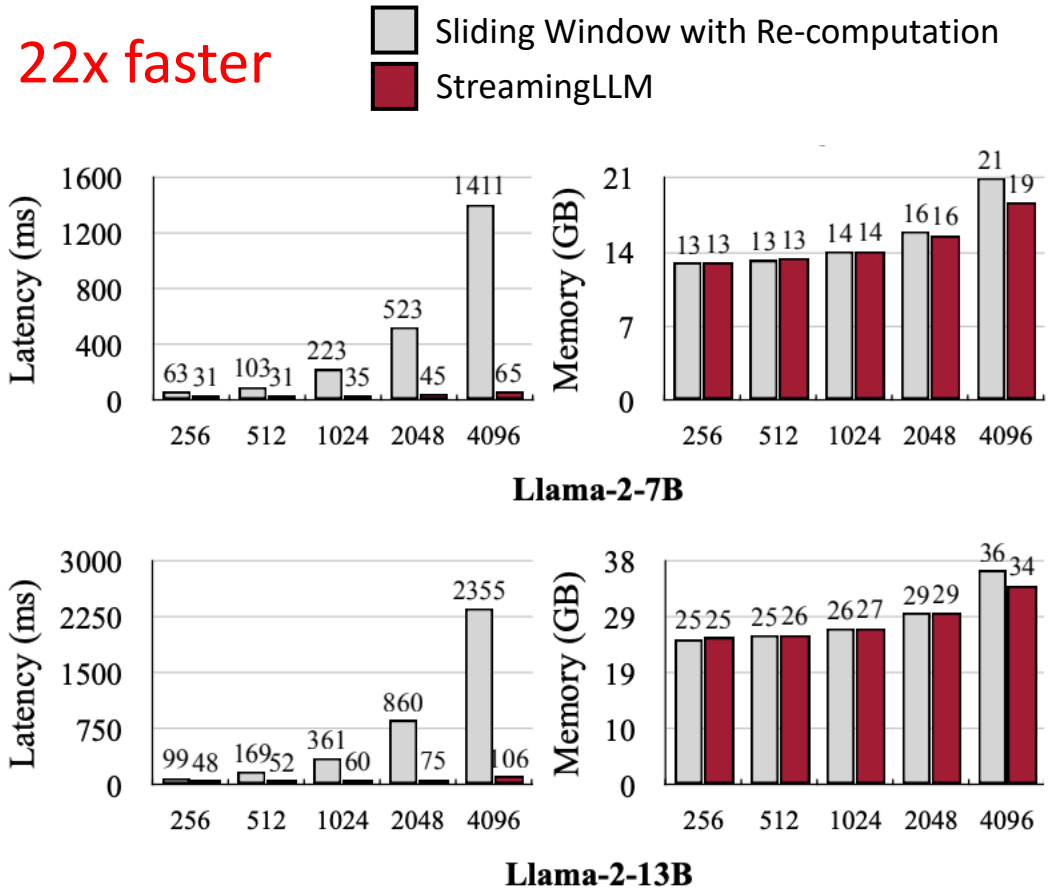
StreamingLLM



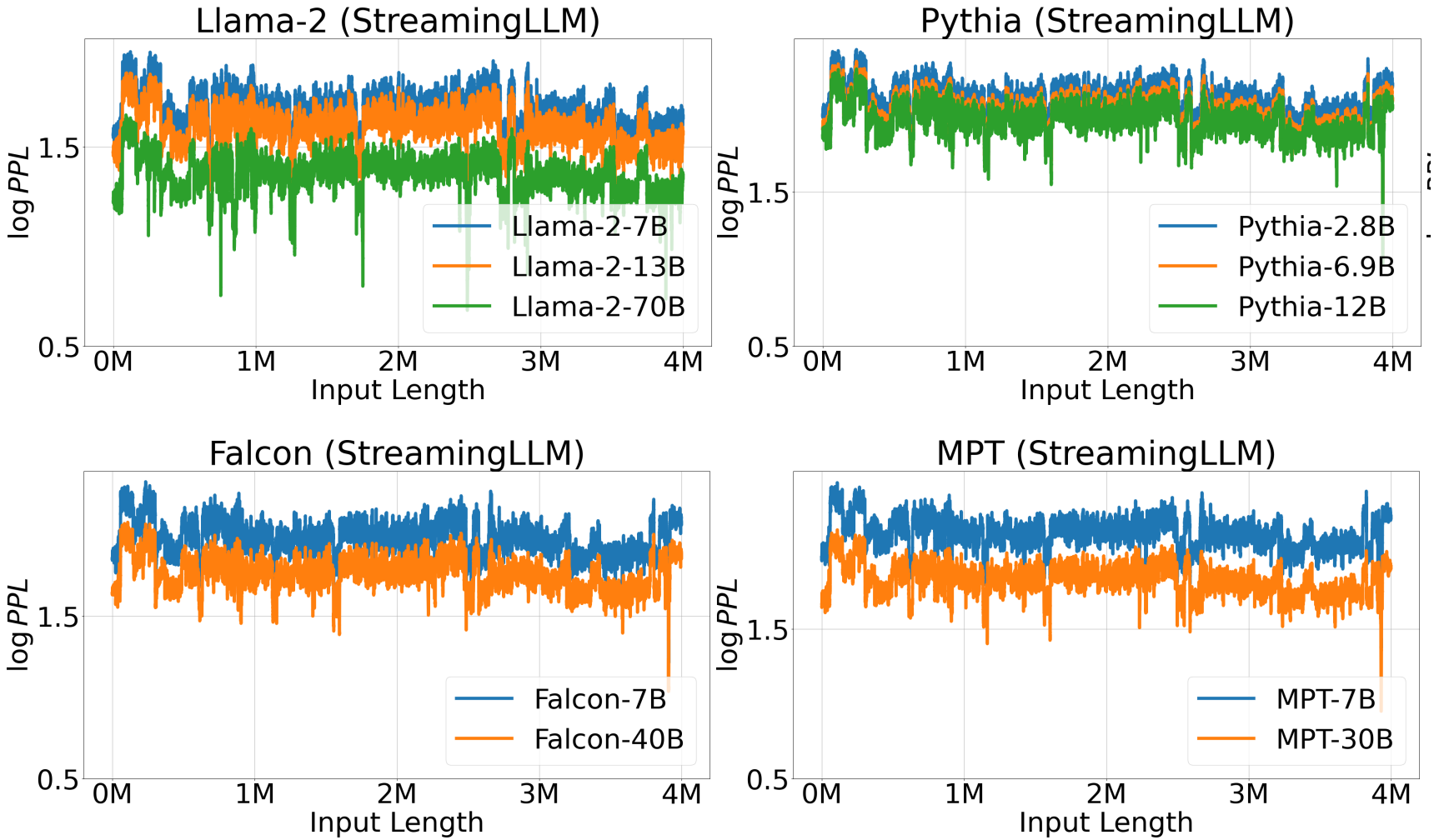
StreamingLLM: stable PPL, constant vRAM

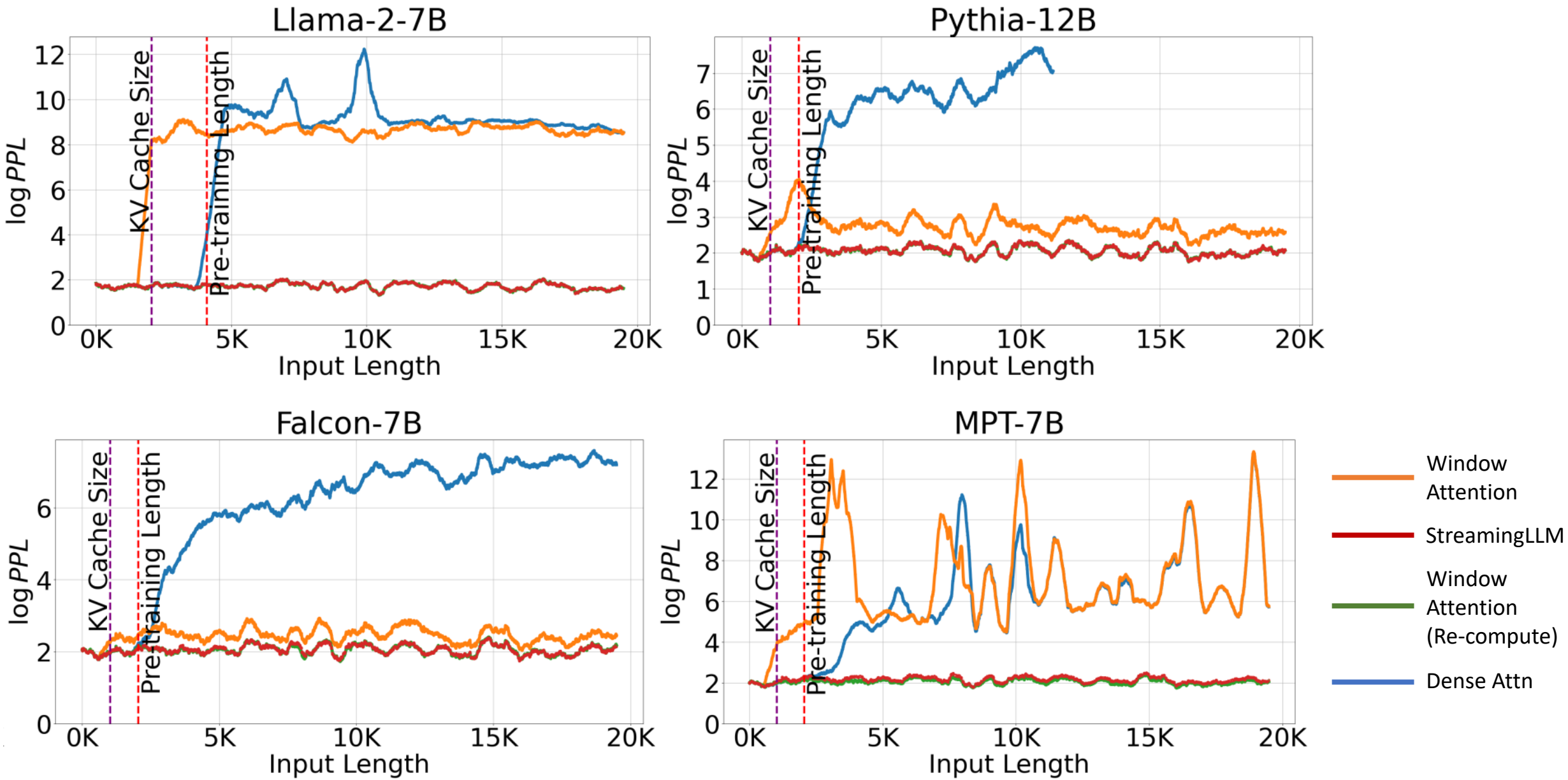


22x faster



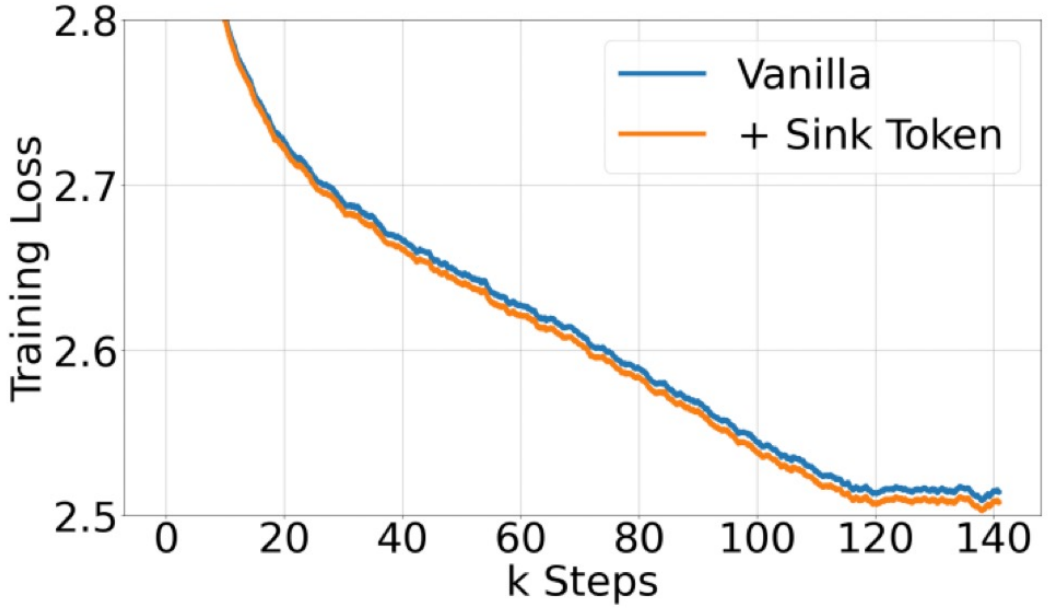
StreamingLLM: Stably Model up to 4 Million Tokens



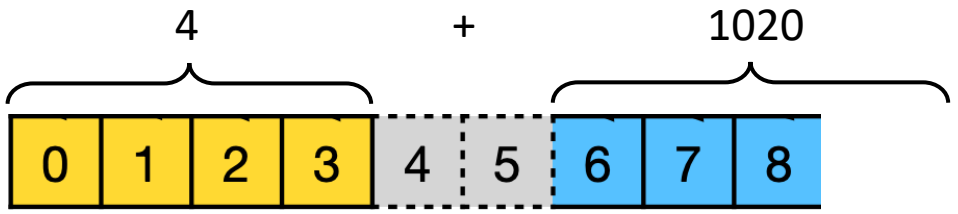

Llama-2-7B
Pythia-12B
Falcon-7B
MPT-7B
log PPL
Input Length
KV Cache Size
Pre-training Length
0K
5K
10K
15K
20K
Window Attention
StreamingLLM
Window Attention (Re-compute)
Dense Attn

Understanding Attention Sinks

- Pre-train with a Dedicated Attention Sink Token

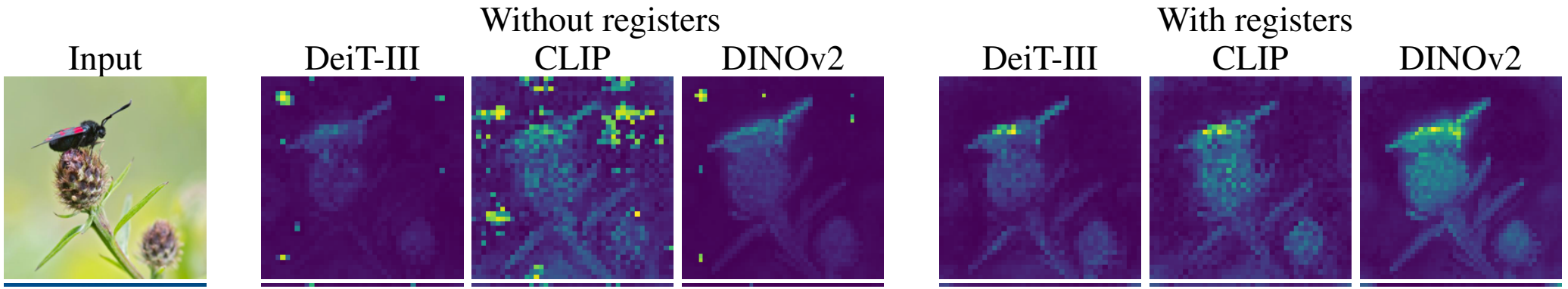


Cache Config	PPL (↓)			
	0+1024	1+1023	2+1022	4+1020
Vanilla	27.87	18.49	18.05	18.05
Zero Sink	29214	19.90	18.27	18.01
Learnable Sink	1235	18.01	18.01	18.02



Understanding Attention Sinks

- Similar Phenomenon in [*Darcet et al. Vision transformers need registers*]
  - *ViT is not decoder-only so there is no preference on initial tokens.*



Does StreamingLLM solve long-context? No

Accuracy (in %) on StreamEval with increasing query-answer distance

Llama-2-7B-32K-Instruct		Cache Config			
Line Distances	Token Distances	4+2044	4+4092	4+8188	4+16380
20	460	85.80	84.60	81.15	77.65
40	920	80.35	83.80	81.25	77.50
60	1380	79.15	82.80	81.50	78.50
80	1840	75.30	77.15	76.40	73.80
100	2300	0.00	61.60	50.10	40.50
150	3450	0.00	68.20	58.30	38.45
200	4600	0.00	0.00	62.75	46.90
400	9200	0.00	0.00	0.00	45.70
600	13800	0.00	0.00	0.00	28.50
800	18400	0.00	0.00	0.00	0.00
1000	23000	0.00	0.00	0.00	0.00

Does StreamingLLM solve long-context? No

Accuracy (in %) on StreamEval with increasing query-answer distance

Llama-2-7B-32K-Instruct		Cache Config			
Line Distances	Token Distances	4+2044	4+4092	4+8188	4+16380
20	460	85.80	84.60	81.15	77.65
40	920	80.35	82.90	81.25	77.60
60	1380	79.15	82.80	81.50	78.50
80	1840	75.30	77.15	76.40	73.80
100	2300	0.00	61.60	50.10	40.50
150	3450	0.00	68.20	58.30	38.45
200	4600	0.00	0.00	62.75	46.90
400	9200	0.00	0.00	0.00	45.70
600	13800	0.00	0.00	0.00	28.50
800	18400	0.00	0.00	0.00	0.00
1000	23000	0.00	0.00	0.00	0.00

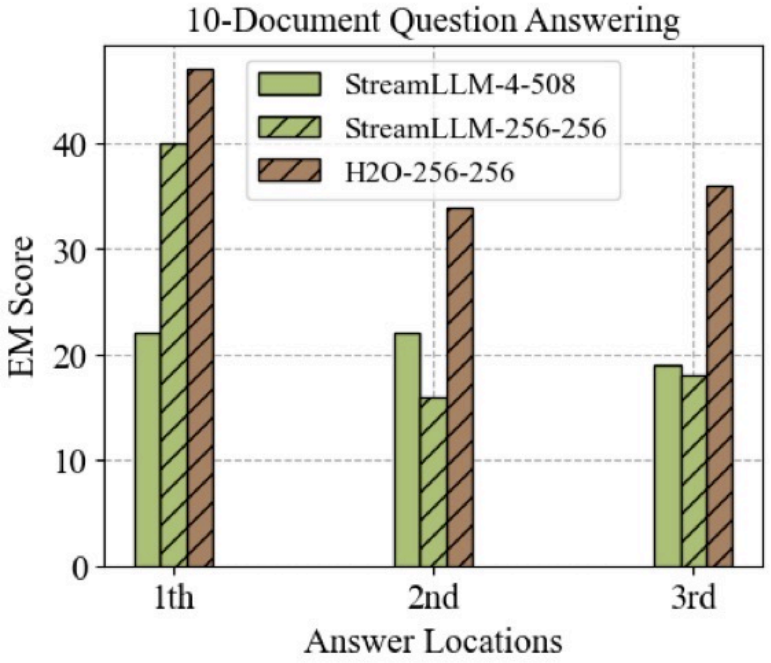
Can we solve the issues of *loss in the middle*?

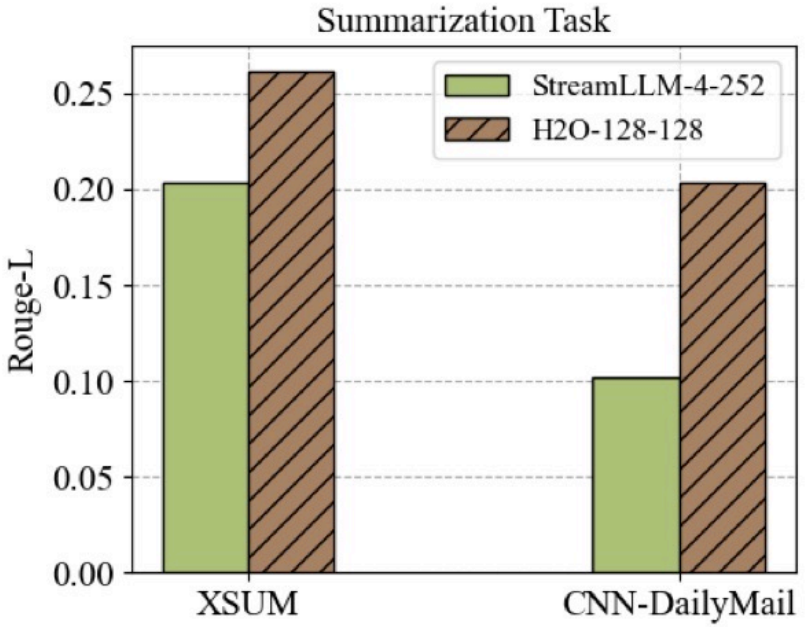
StreamingH2O: Infinite Streaming Ability

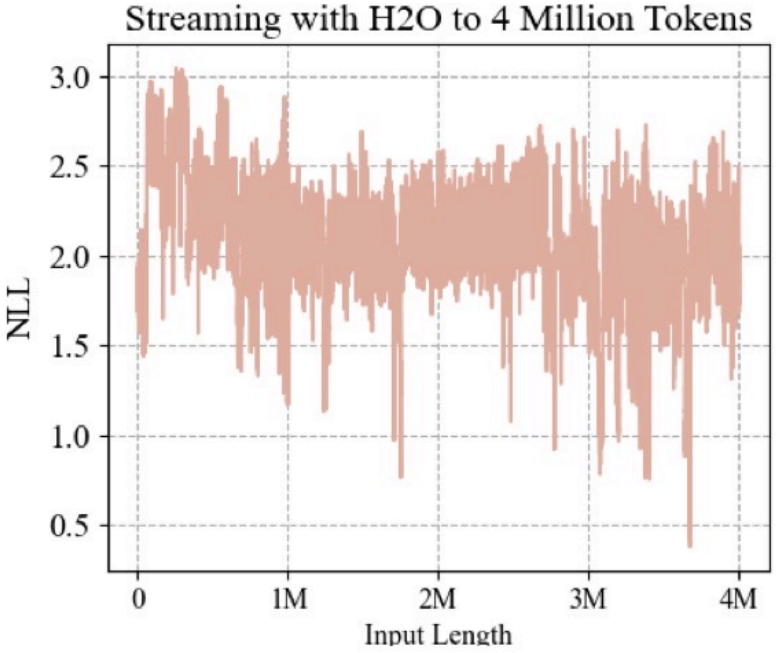
StreamingLLM is a subset of H2O

Applying **position rolling** to H2O!

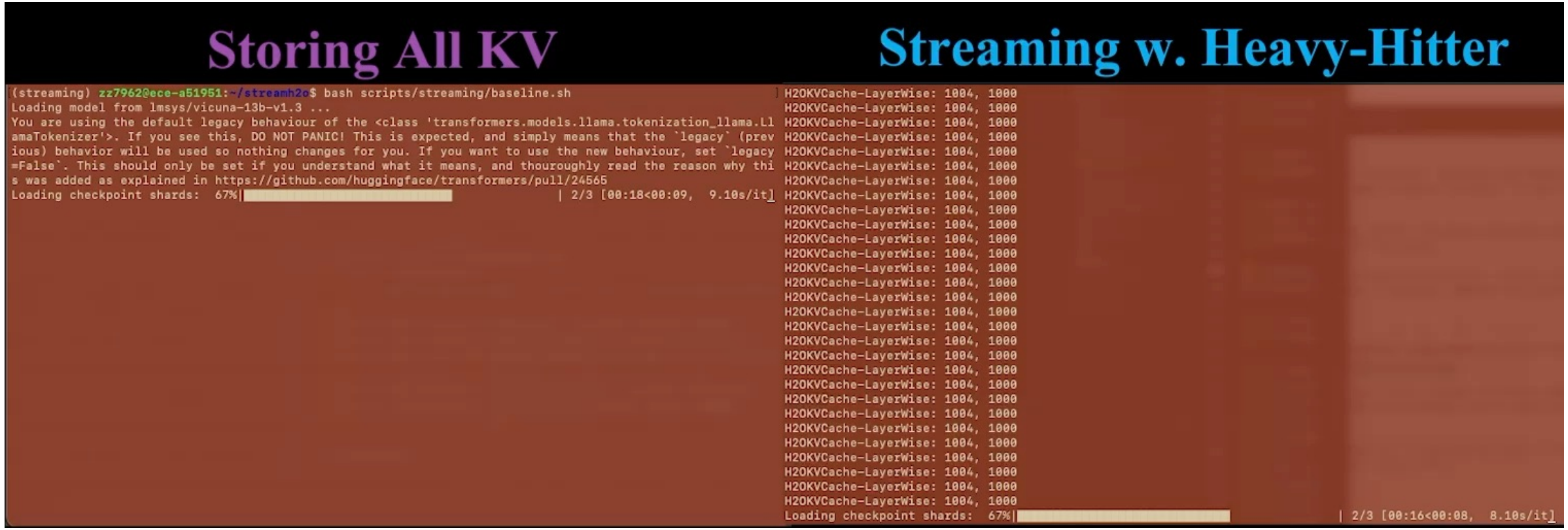
The heavy-hitters of H2O helps mitigate “lost in the middle” issues.



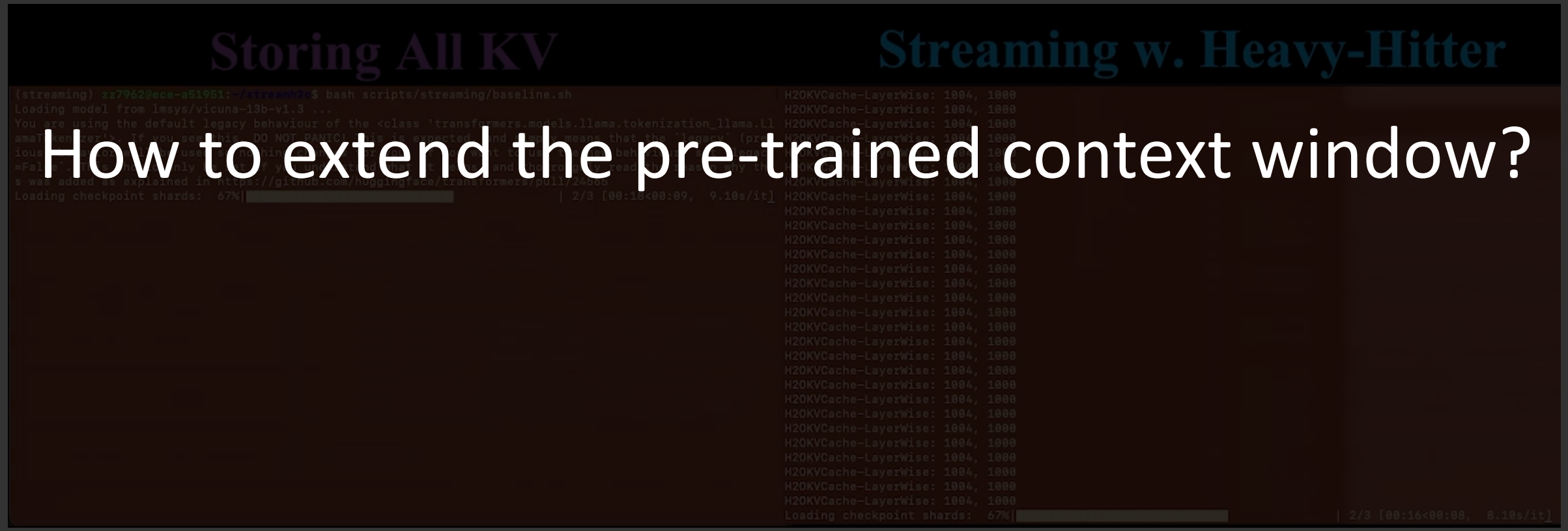




StreamingH2O: Infinite Streaming Ability



StreamingH2O: Infinite Streaming Ability



Extending context window for pre-trained models

Model Size	Context Window	Method	Evaluation Context Window Size 2048	4096	8192	16384	32768
7B	2048	None	7.20	$> 10^3$	$> 10^3$	$> 10^3$	$> 10^3$

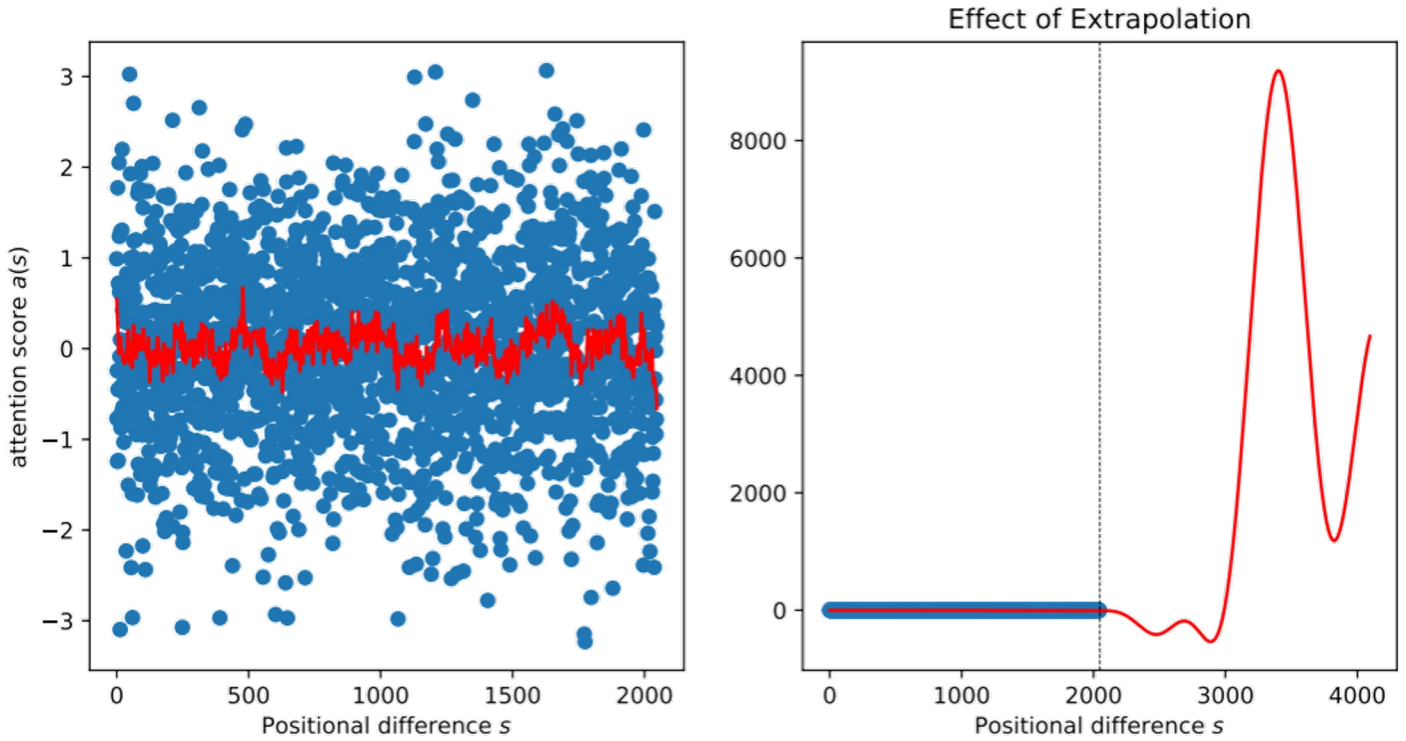
Key issues

Pre-trained model (e.g., LLaMA) cannot go beyond its pre-trained context window

[*S. Chen et al*, Extending Context Window of Large Language Models via Positional Interpolation, *arXiv*]

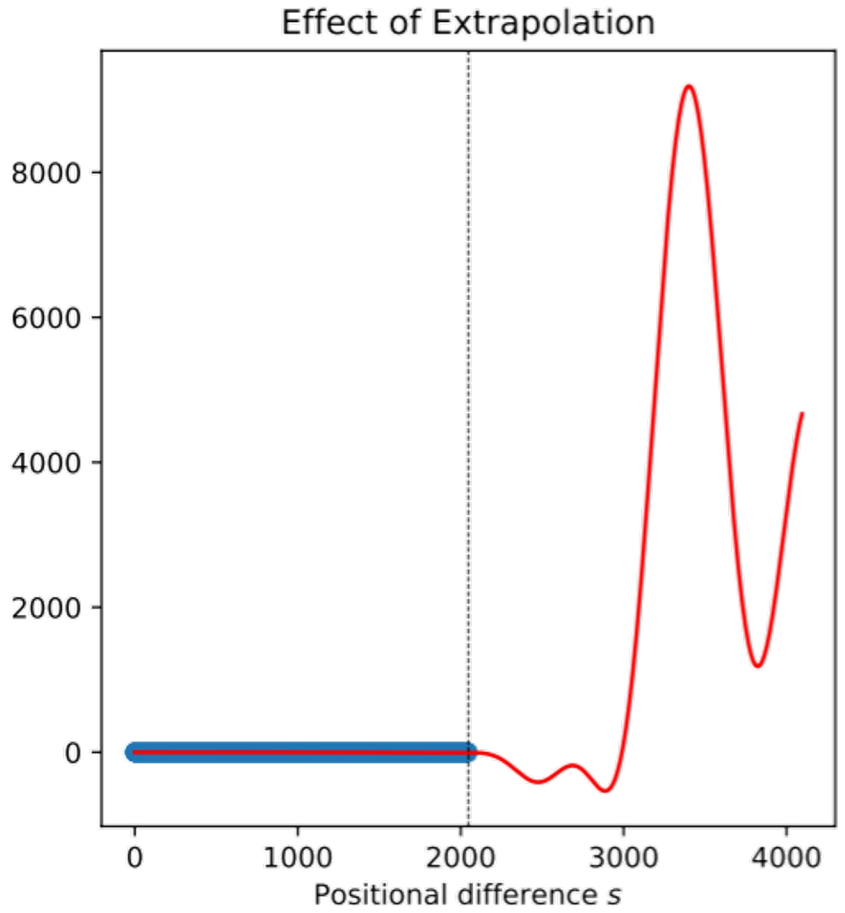
Why? Attention Function $\boldsymbol{a(s)}$ is ill-behaved

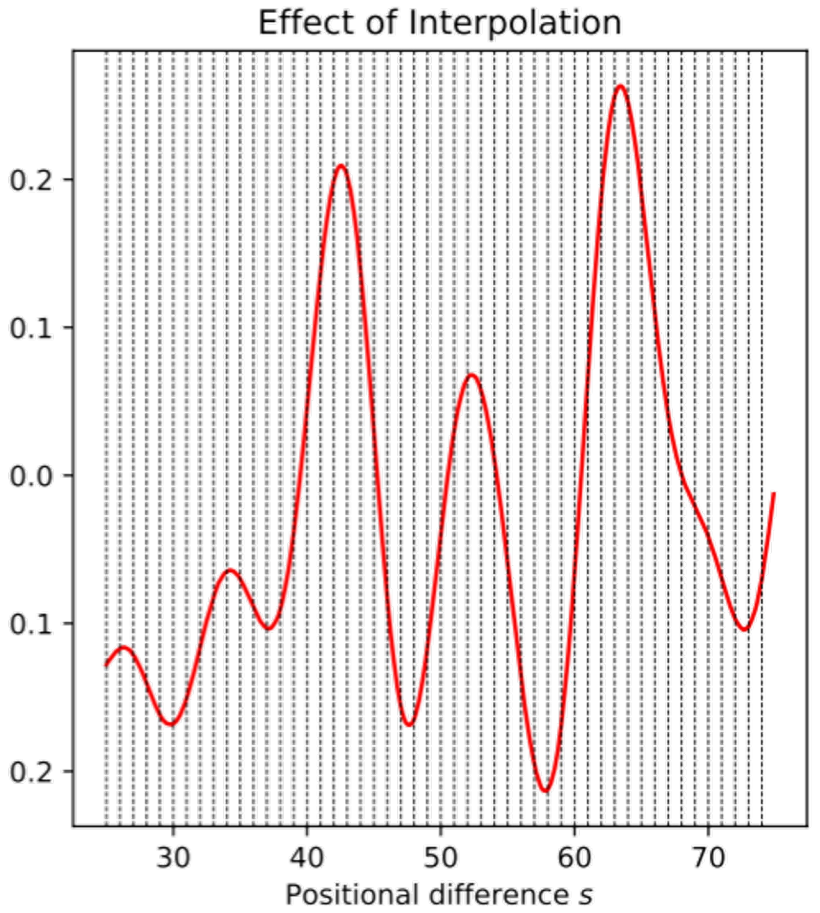
RoPE encoding: $\boldsymbol{f}(\boldsymbol{x}, m) = [(x_0 + \mathrm{i}x_1)e^{im\theta_0}, (x_2 + \mathrm{i}x_3)e^{im\theta_1}, \ldots, (x_{d-2} + \mathrm{i}x_{d-1})e^{im\theta_{d/2-1}}]$
Attention function: $a(\mathrm{s}) = a(m-n) = \mathrm{Re}\langle \boldsymbol{f}(\boldsymbol{q}, m), \boldsymbol{f}(\boldsymbol{k}, n)\rangle$



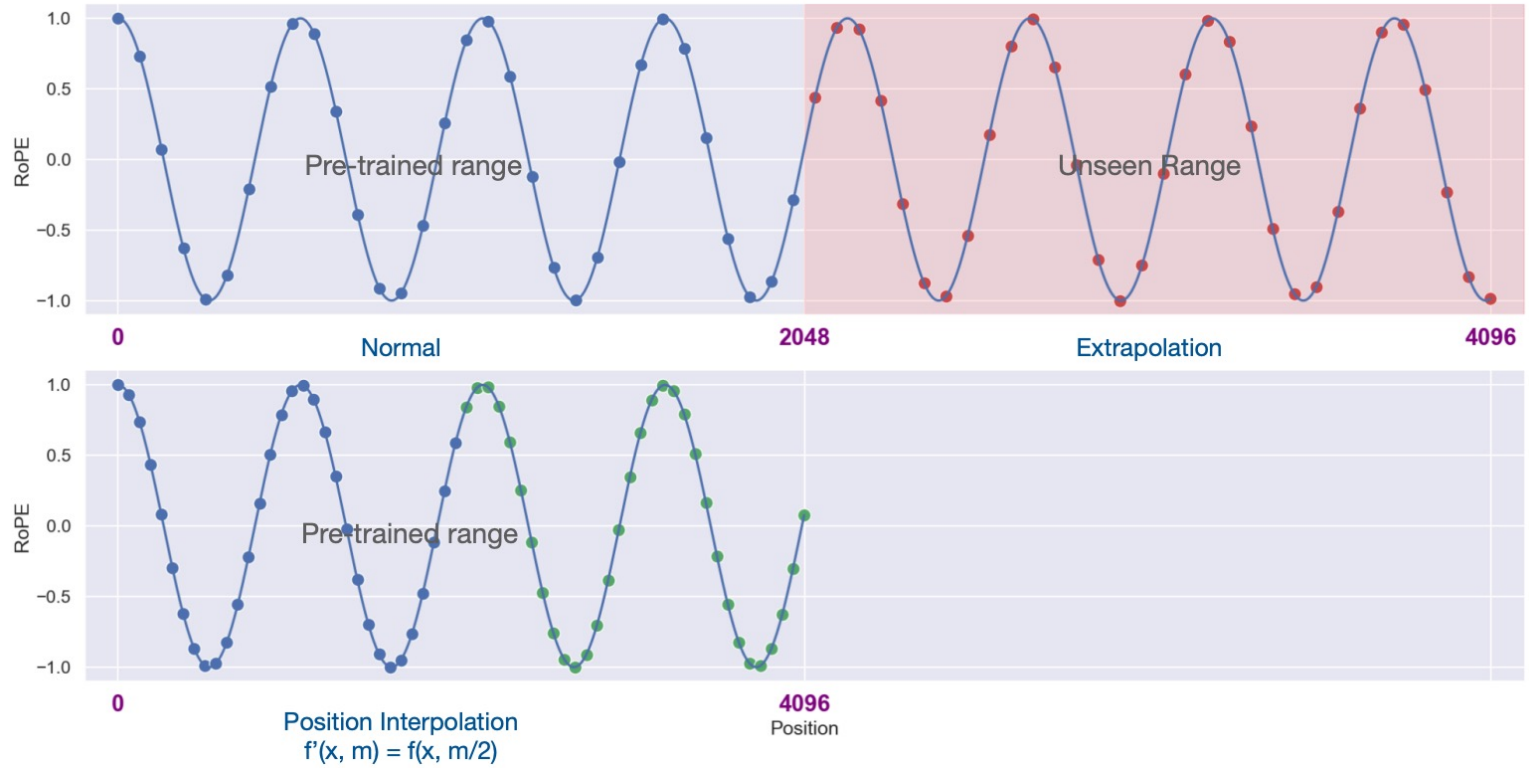
$a(s)$ behaves well within the pre-trained window, but could go crazy when **extrapolating.**

Interpolation versus Extrapolation





Positional Interpolation (PI)



Interpolated encoding: $\boldsymbol{f}'(\boldsymbol{x}, m) = \boldsymbol{f}(\boldsymbol{x}, \beta m)$. Here $\alpha := L/L'$ is the scaling factor

Or equivalently, $\theta_j' = \alpha c^{-2j/d} = \alpha\theta_j$, where $\theta_j = c^{-2j/d}$, $c = 10000$

Experimental Results

Size	Model Context Window	Method	Evaluation Context Window Size 2048	4096	8192	16384	32768
7B	2048	None	7.20	$> 10^3$	$> 10^3$	$> 10^3$	$> 10^3$
7B	8192	FT	7.21	7.34	7.69	-	-
7B	8192	PI	7.13	6.96	6.95	-	-
7B	16384	PI	7.11	6.93	6.82	6.83	-
7B	32768	PI	7.23	7.04	6.91	6.80	6.77
13B	2048	None	6.59	-	-	-	-
13B	8192	FT	6.56	6.57	6.69	-	-
13B	8192	PI	6.55	6.42	6.42	-	-
13B	16384	PI	6.56	6.42	6.31	6.32	-
13B	32768	PI	6.54	6.40	6.28	6.18	6.09
33B	2048	None	5.82	-	-	-	-
33B	8192	FT	5.88	5.99	6.21	-	-
33B	8192	PI	5.82	5.69	5.71	-	-
33B	16384	PI	5.87	5.74	5.67	5.68	-
65B	2048	None	5.49	-	-	-	-
65B	8192	PI	5.42	5.32	5.37	-	-

With <1000 steps of fine-tuning,
PI can extrapolate up to 8x length of its original context windows

Experimental Results

#fine-tune steps needed to achieve longer context window (measured by passkey retrieval)

Model			Fine-tuning steps					
Size	Context Window	Method	200	400	600	800	1000	10000
7B	8192	FT	1792	2048	2048	2048	2304	2560
33B	8192	FT	1792	2048	1792	2048	2304	-
7B	8192	PI	8192	8192	8192	8192	8192	-
7B	16384	PI	16384	16384	16384	16384	16384	-
7B	32768	PI	32768	32768	18432	32768	32768	-
33B	8192	PI	8192	8192	8192	8192	8192	-
33B	16384	PI	16384	16384	16384	16384	16384	-

200 steps suffice!

Performance remains in LLM benchmarks

Model Size	Context Window	Fine-tune on	BoolQ	PIQA	Race-M	Race-H	WinoGrande
7B	2048	None	76.1	78.9	55.7	42.2	69.6
7B	8192	Pile	73.2	78.2	53.8	41.7	69.0
7B	16384	Pile	69.8	77.6	53.3	40.9	67.8
7B	32768	Pile	64.7	77.2	50.1	39.6	66.9
7B	8192	RedPajama	75.5	77.4	54.5	41.5	68.1
33B	2048	None	81.6	80.2	61.1	45.9	76.2
33B	8192	Pile	80.2	80.7	60.2	45.7	75.9

Future Development Inspired by our work

[Jun. 29, two days after our arXiv release]

"NTK" positional encoding

$$\theta_j' = (\alpha c)^{-2j/d}$$

Extrapolation in high-frequency

Interpolation in low-frequency

No fine-tuning needed.

r/LocalLLaMA

Posts

Posted by u/bloc97 5 months ago

415

NTK-Aware Scaled RoPE allows LLaMA models to have extended (8k+) context size without any fine-tuning and minimal perplexity degradation.

News

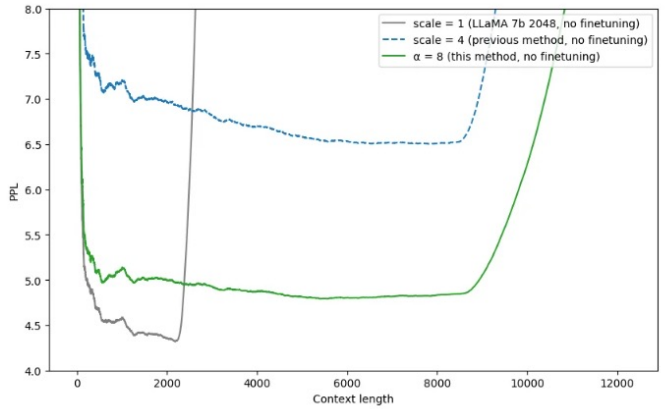
I've seen the posts about SuperHOT and just recently, the paper from Meta which uses RoPE interpolation, and I've noticed an immediate improvement that can be brought to this method. Basically if you apply Neural Tangent Kernel (NTK) theory to this problem, it becomes clear that simply interpolating the RoPE's fourier space "linearly" is very sub-optimal, as it prevents the network to distinguish the order and positions of tokens that are very close by. Borrowing from NTK literature, scaling down the fourier features too much will eventually even prevent succesful finetunes (this is corroborated by the recent paper by Meta that suggests an upper bound of ~600x)

Instead of the simple linear interpolation scheme, I've tried to design a nonlinear interpolation scheme using tools from NTK literature. Basically this interpolation scheme changes the base of the RoPE instead of the scale, which intuitively changes the "spinning" speed which each of the RoPE's dimension vectors compared to the next. Because it does not scale the fourier features directly, all the positions are perfectly distinguishable from eachother, even when taken to the extreme (eg. streched 1million times, which is effectively a context size of 2 Billion)

To my surprise, this method works extremely well, so much so that you don't even need to fine tune the LLaMA 7B model for 4096 context size! The perplexity degradation is minimal. I'm sure with fine tuning this would become even better.

Enough explanations, here's some empirical results. All the perplexity measurements are done on LLaMA 7b with the tau/scrolls - Datasets at Hugging Face dataset (I only used a subset of gov_report).

Here's a graph showing the average perplexity of LLaMA 7b on a set of 40 very long prompts (12k+ context size). Compared to changing the scale (from SuperHOT, which was set to 4), we change a factor alpha, which when equal to 8 provides the same context size increase but with much less perplexity degradation. All without any finetuning!



Graph showing the average perplexity of LLaMA 7b on set of 40 very long prompt (12k+ context size) with previous and new interpolation scheme

Thanks!