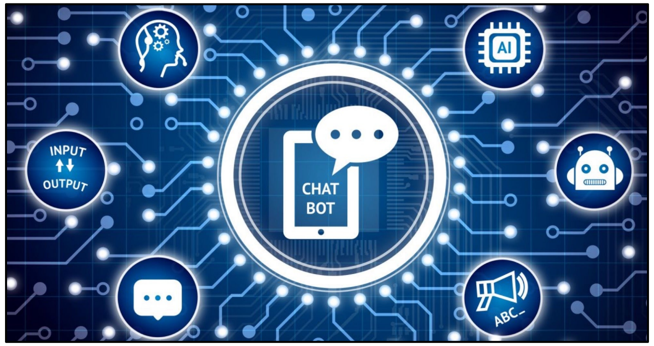
Demystifying Self-Attention Mechanism in Feature Composition and Logic Reasoning via the Lens of Training Dynamics

Yuandong Tian
Research Scientist

Meta AI (FAIR)

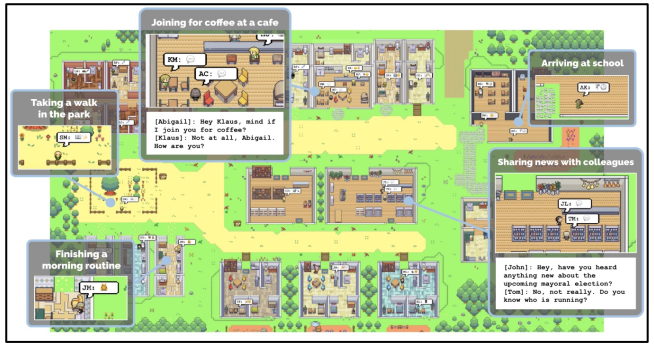
facebook Artificial Intelligence

Large Language Models (LLMs)

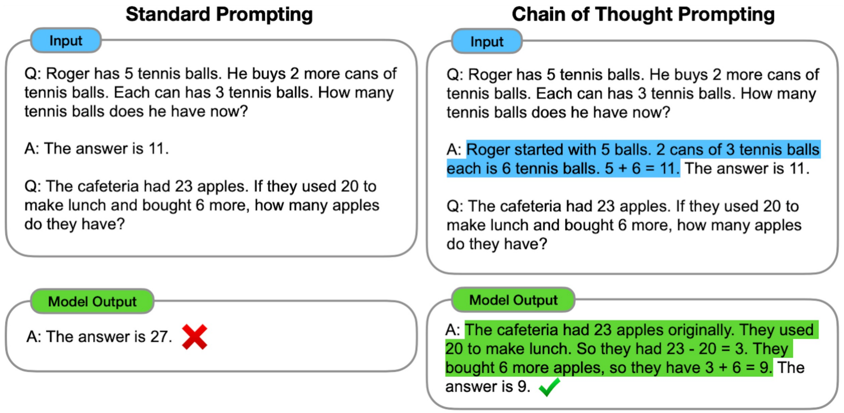


Conversational AI

Content Generation



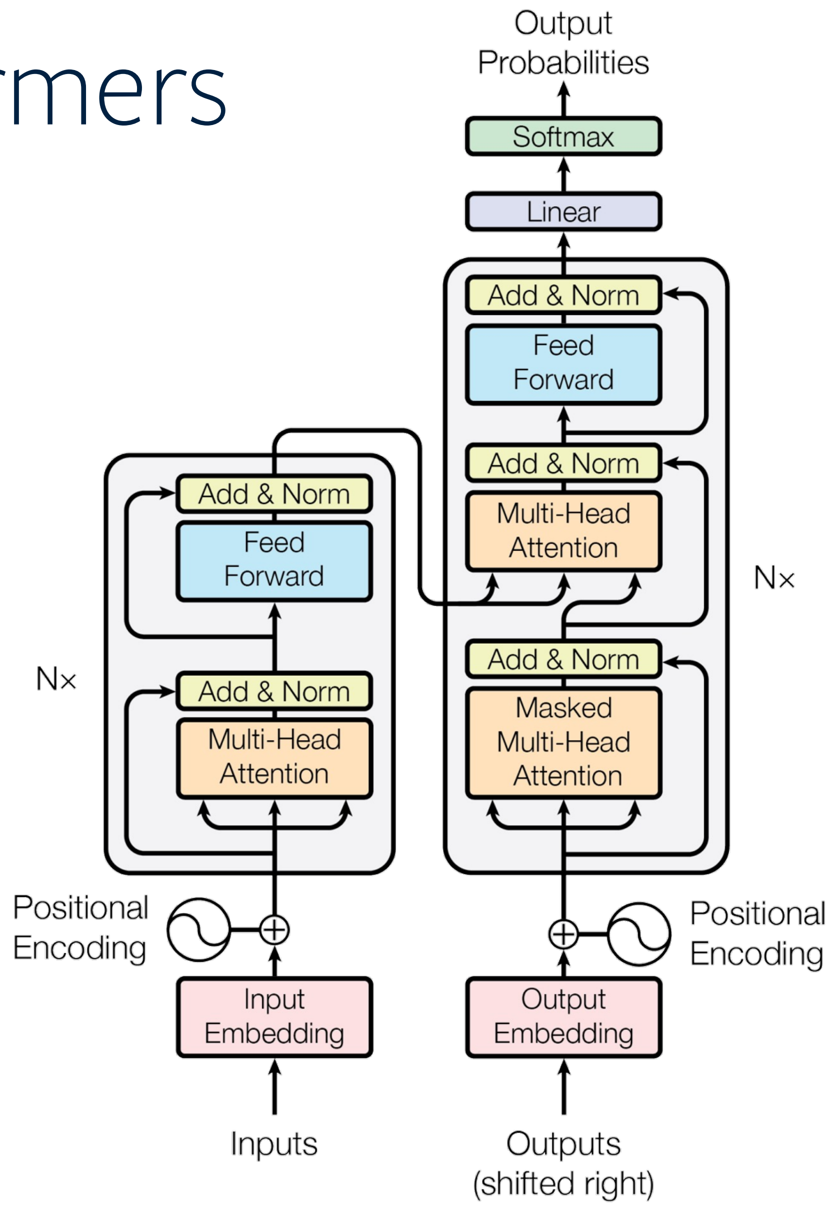
AI Agents

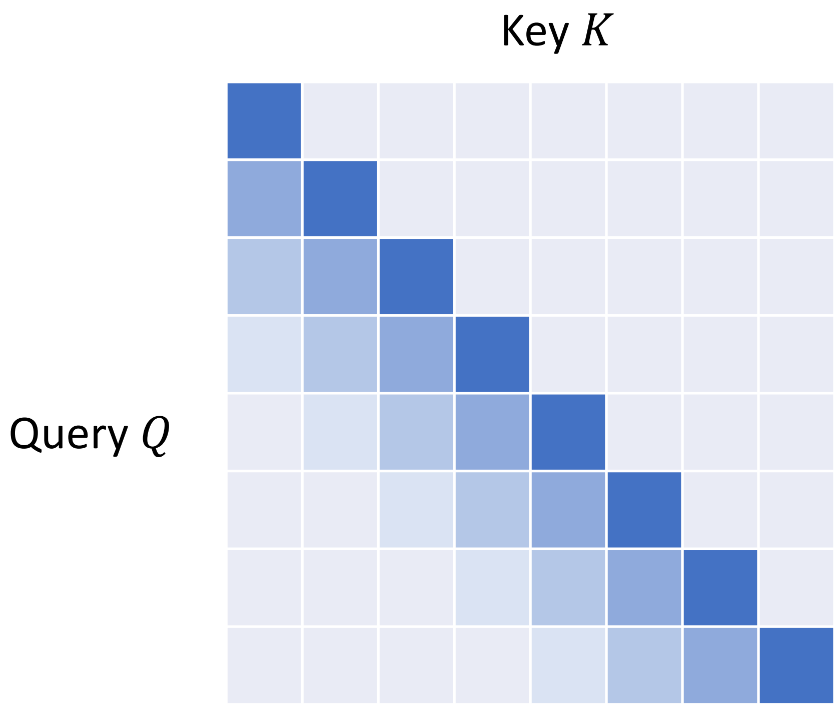


Reasoning

Planning

Transformers



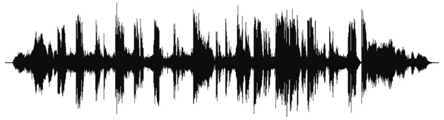


Attention mechanism

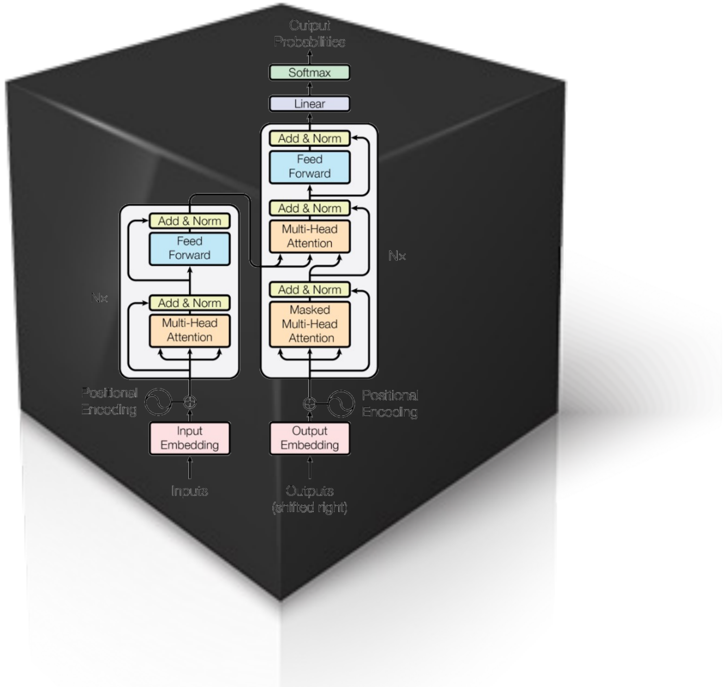
[A. Vaswani et al, Attention is all you need, NeurIPS’17]

How does Transformer work?

Input



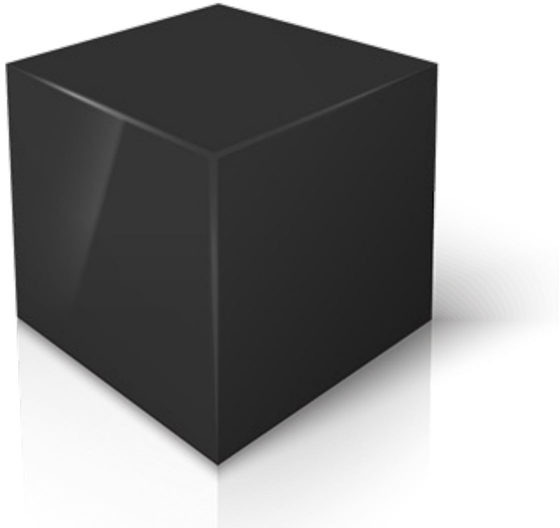
This is an apple



"Some Nonlinear Transformation"

Output

Black-box versus White-box



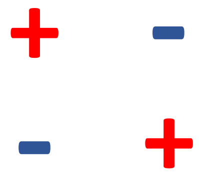
Black box

White box

Three Angles

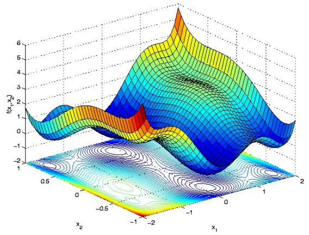
Understanding how Deep Models work

Expressibility



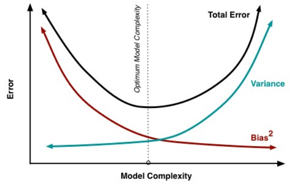
"Neural Network is a universal approximator"
"Deep Models can express functions more efficiently than shallow ones"

Optimization



"Gradient vanishing/exploding"
"Gradient Descent might get stuck at saddle point / local minima"
"Can GD/SGD go to global optima? How fast?"

Generalization



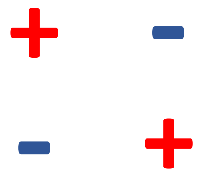
"Does zero training error often lead to overfitting?"
"More parameters might lead to overfitting."

facebook Artificial Intelligence

Three Angles

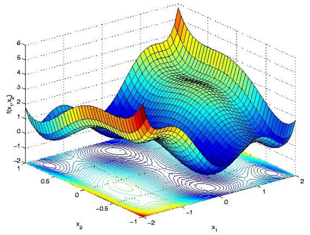
Understanding how Deep Models work

Expressibility



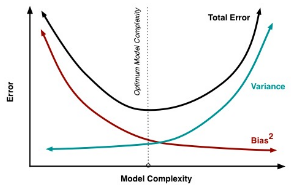
"Neural Network is a universal approximator"
"Deep Models can express functions more efficiently than shallow ones"

Optimization



"Gradient vanishing/exploding"
"Gradient Descent might get stuck at saddle point / local minima"
"Can GD/SGD go to global optima? How fast?"

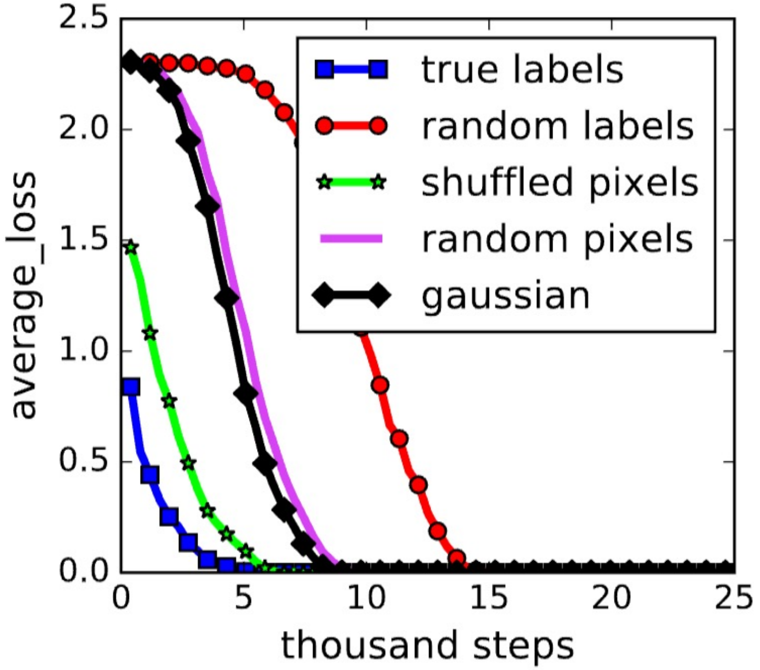
Generalization



"Does zero training error often lead to overfitting?"
"More parameters might lead to overfitting."

Which path should we take?

Rethinking Generalization



(a) learning curves

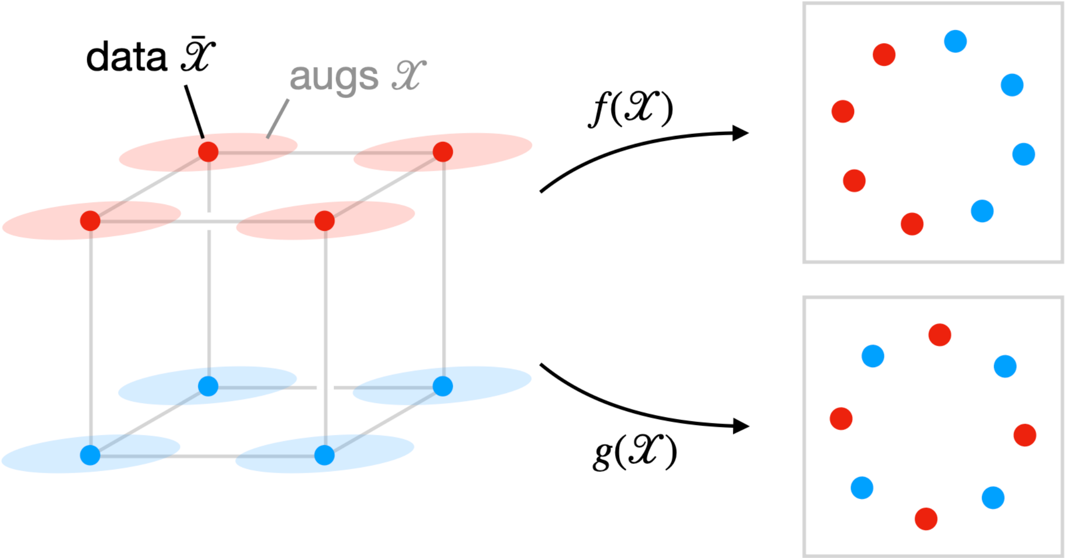
model	# params	random crop	weight decay	train accuracy	test accuracy
Inception	1,649,402	yes	yes	100.0	89.05
		yes	no	100.0	89.31
		no	yes	100.0	86.03
		no	no	100.0	85.75
(fitting random labels)		no	no	100.0	9.78
Inception w/o BatchNorm	1,649,402	no	yes	100.0	83.00
		no	no	100.0	82.00
(fitting random labels)		no	no	100.0	10.12
Alexnet	1,387,786	yes	yes	99.90	81.22
		yes	no	99.82	79.66
		no	yes	100.0	77.36
		no	no	100.0	76.07
(fitting random labels)		no	no	99.82	9.86
MLP 3x512	1,735,178	no	yes	100.0	53.35
		no	no	100.0	52.39
(fitting random labels)		no	no	100.0	10.48
MLP 1x512	1,209,866	no	yes	99.80	50.39
		no	no	100.0	50.51
(fitting random labels)		no	no	99.34	10.61

Generalization bound failed: $Test\ Error \leq Train\ Error + ???$

[C. Zhang et al, *Understanding deep learning requires rethinking generalization,* ICLR 2017]

Inductive Bias Really Matters

A self-supervised contrastive learning example



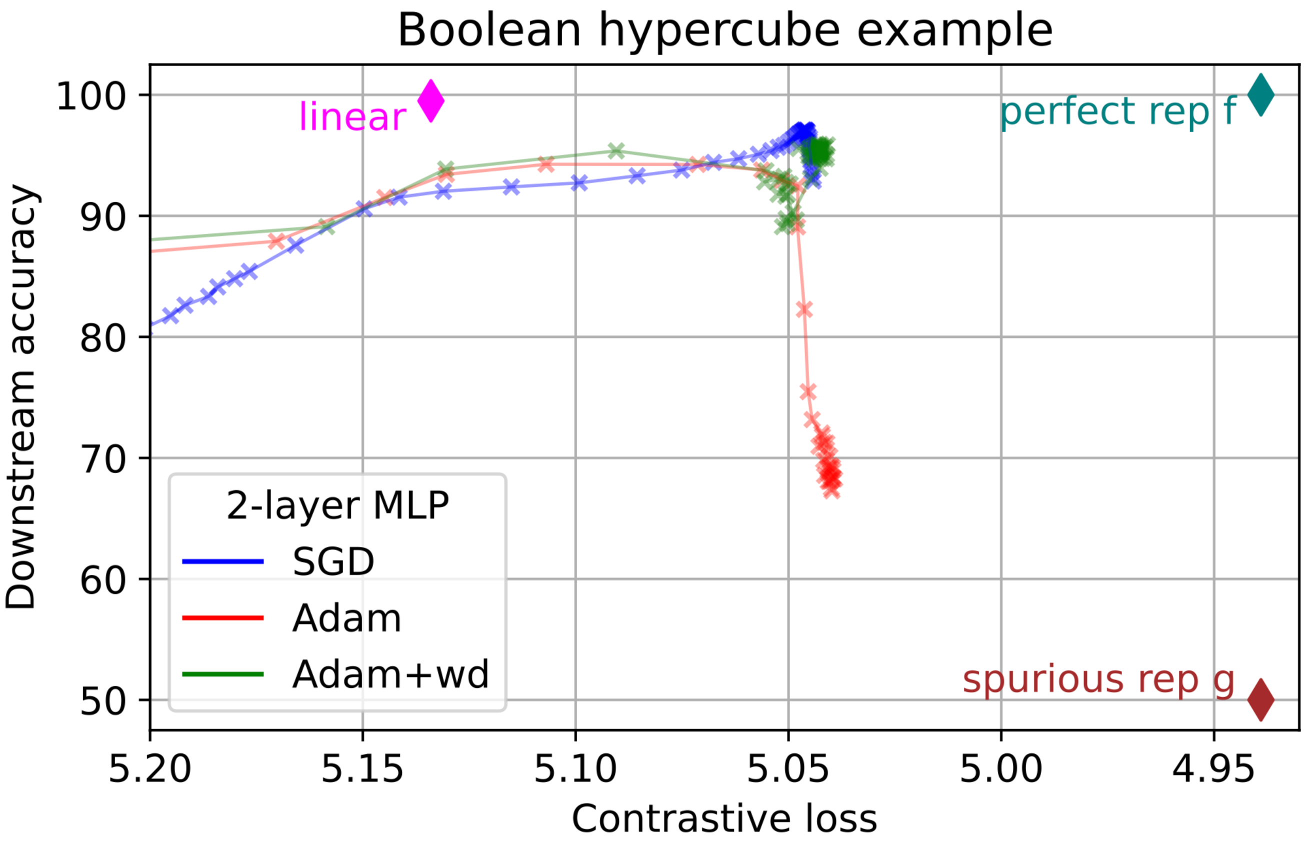
Pretraining: $L_{\text{cont}}(g) \approx L_{\text{cont}}(f)$

Downstream: $L_{\text{clf}}(g) \gg L_{\text{clf}}(f)$

SSL Pertraining loss doesn’t really reflect downstream loss

[N. Saunshi et al, *Understanding Contrastive Learning Requires Incorporating Inductive Biases,* ICML 2022]

Inductive Bias Really Matters

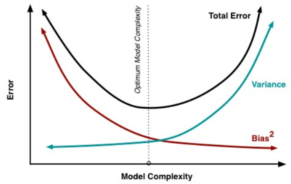


Representation	Contrastive loss	Accuracy (%)
$\exists f$ (perfect)	4.939	100
$\exists g$ (spurious)	4.939	50
MLP + Adam	5.039 ± 0.001	74.1 ± 4.3
MLP + Adam + wd	5.040 ± 0.002	89.5 ± 4.9
Linear	5.134 ± 0.002	99.5 ± 0.1

[N. Saunshi et al, *Understanding Contrastive Learning Requires Incorporating Inductive Biases,* ICML 2022]

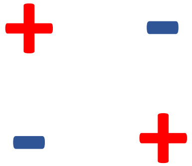
Lesson learned?

Generalization



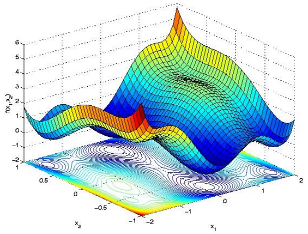
Architecture ✗
training dynamics ✗

Expressibility



Architecture ✓
training dynamics ✗

Optimization

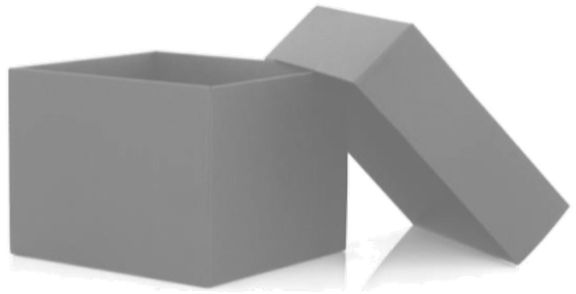


Architecture ✗
training dynamics ✓

How about

Architecture ✓
training dynamics ✓

Start From the First Principle



- Training follows Gradient and its variants (SGD, Adams, etc)

$$\dot{\boldsymbol{w}} := \frac{\mathrm{d}\boldsymbol{w}}{\mathrm{d}t} = -\nabla_{\boldsymbol{w}} J(\boldsymbol{w})$$

- **First principle** → Understand the behavior of the neural networks by checking the gradient **dynamics** induced by the neural **architectures**.

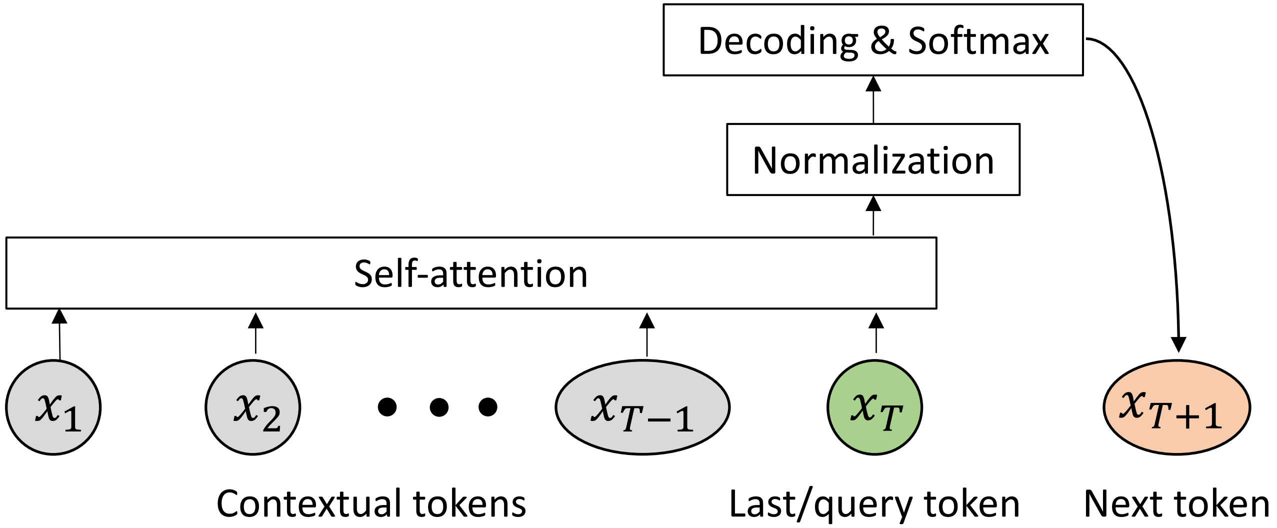
- Sounds complicated.. Is that possible? **Yes**

Architecture ✓

training dynamics ✓

Understanding Attention in 1-layer Setting

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



Self-attention

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

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

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

Objective:

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

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

Reparameterization

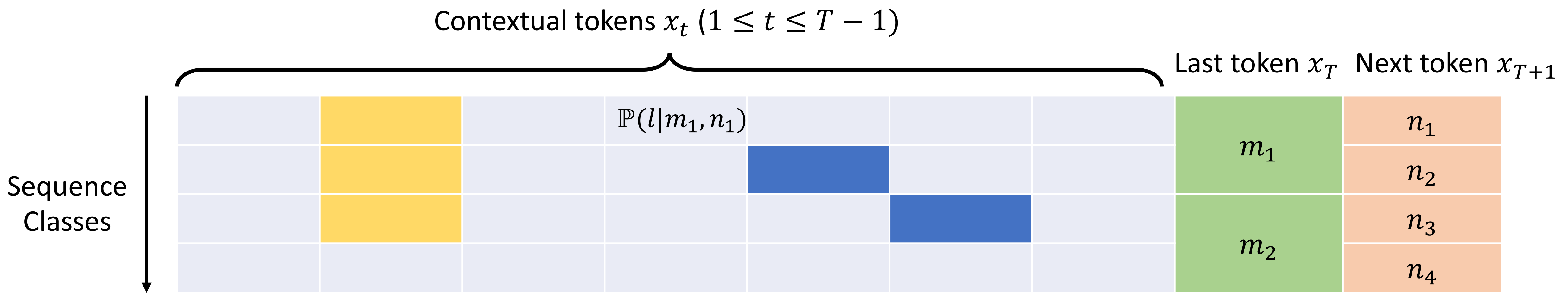
- Parameters W_K, W_Q, W_V, U makes the dynamics complicated.
- Reparameterize the problem with independent variable Y and Z
  - $Y = UW_V^T U^T$
  - $Z = UW_Q W_K^T U^T$ (pairwise logits of self-attention matrix)
- Then the dynamics becomes easier to analyze

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 $\mathbb{P}(l|n) > 0$

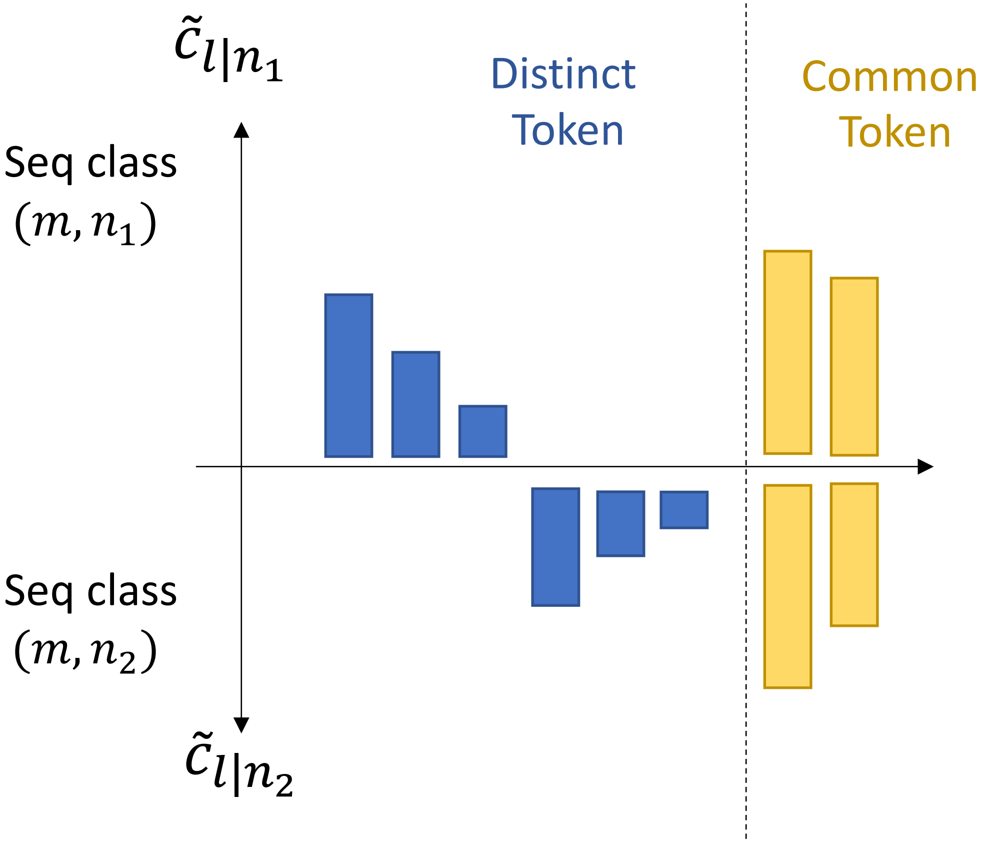
$\mathbb{P}(l|m,n) = \mathbb{P}(l|n)$ is the conditional probability of token l given last token $x_T = m$ and $x_{T+1} = n$

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

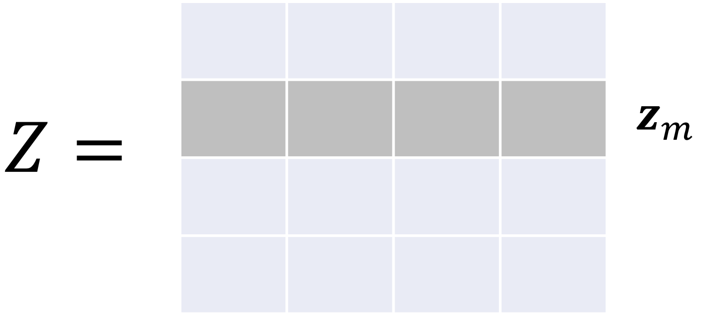
At initialization



Co-occurrence probability

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

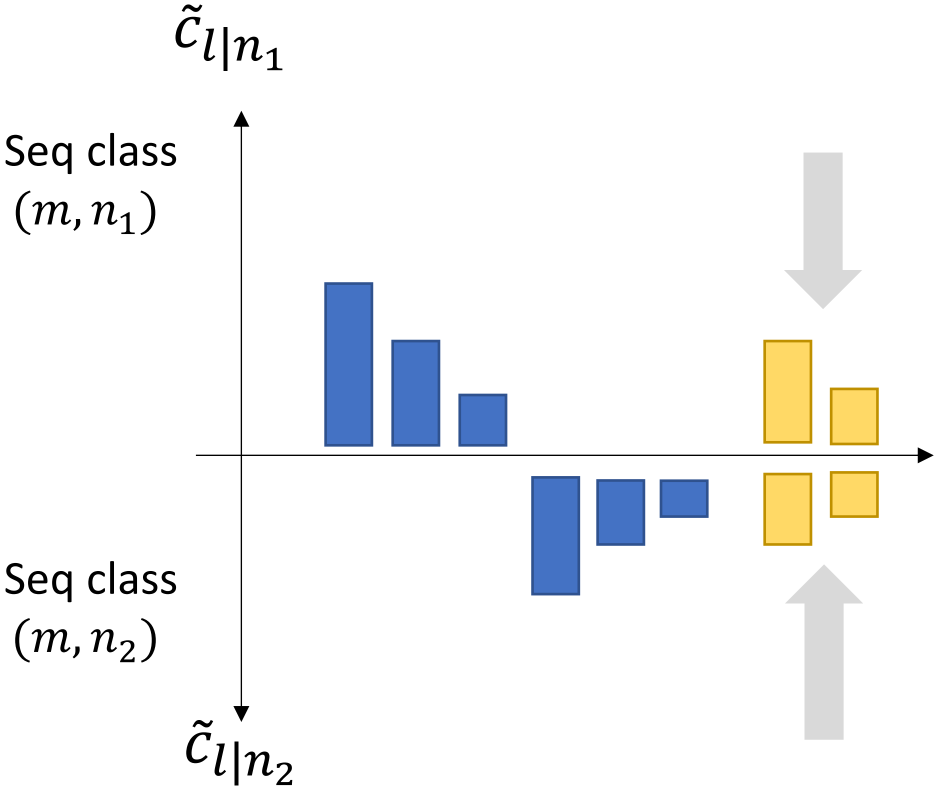
Initial condition: $z_{ml}(0) = 0$



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

Overall Picture of the Training Dynamics

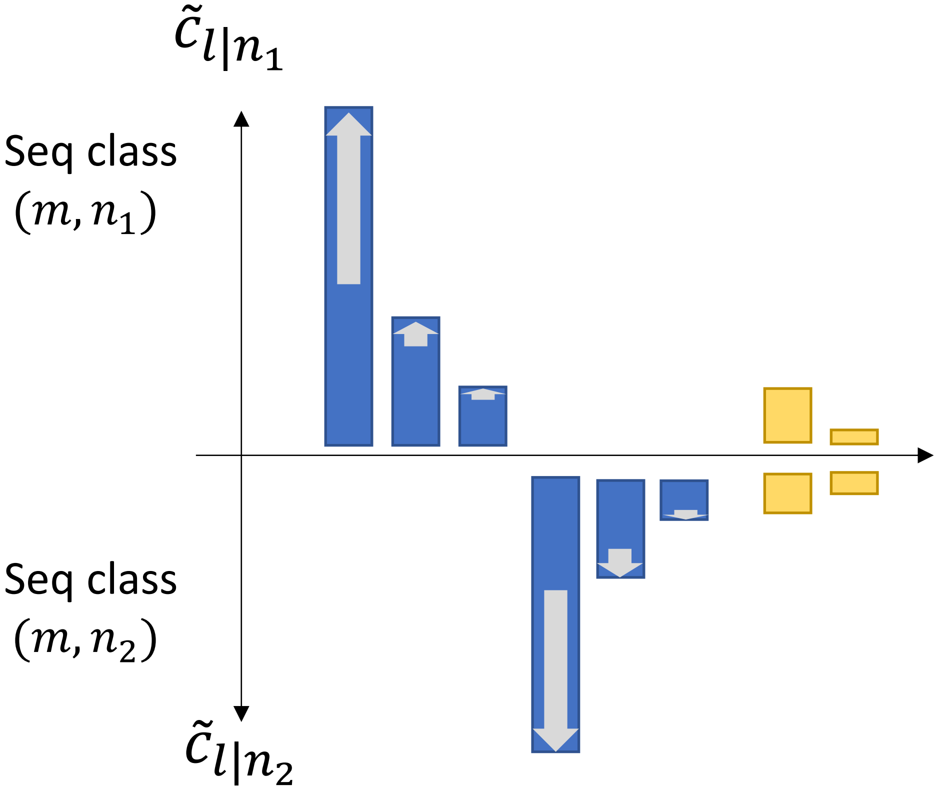
<u>Common Token Suppression</u>



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

Overall Picture of the Training Dynamics

<u>Winners-emergence</u>



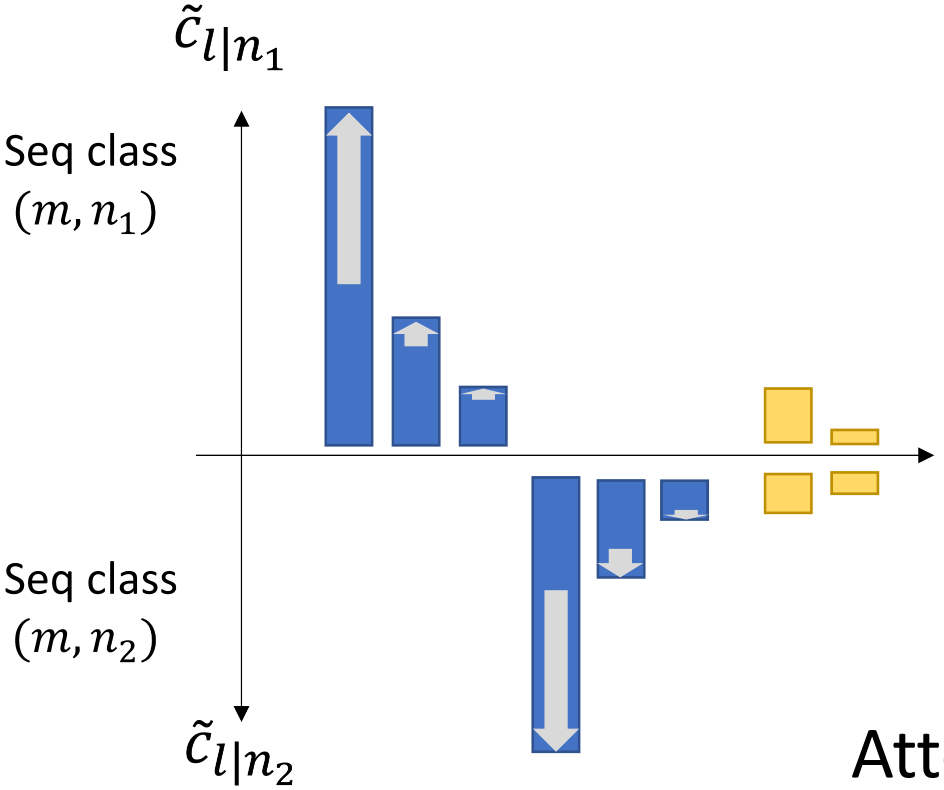
(a) $\dot{z}_{ml} < 0$, for common token l

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

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

Overall Picture of the Training Dynamics

<u>Winners-emergence</u>



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

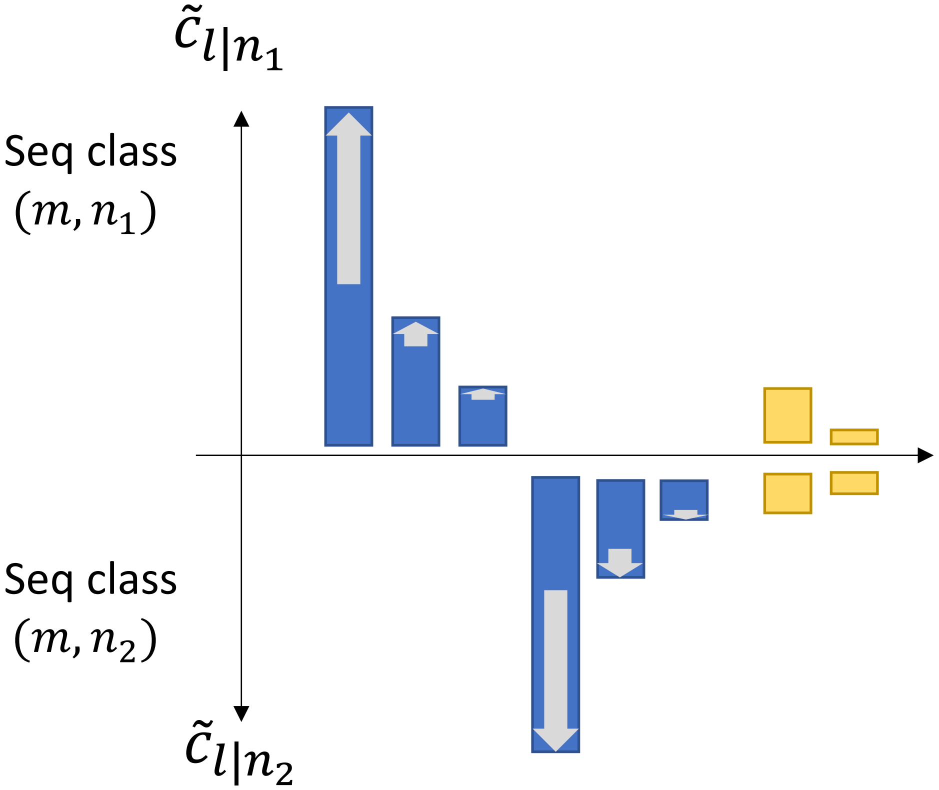
(b) $\dot{z}_{ml} > 0$, for distinct token l

(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

<u>Winners-emergence</u>



(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:

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

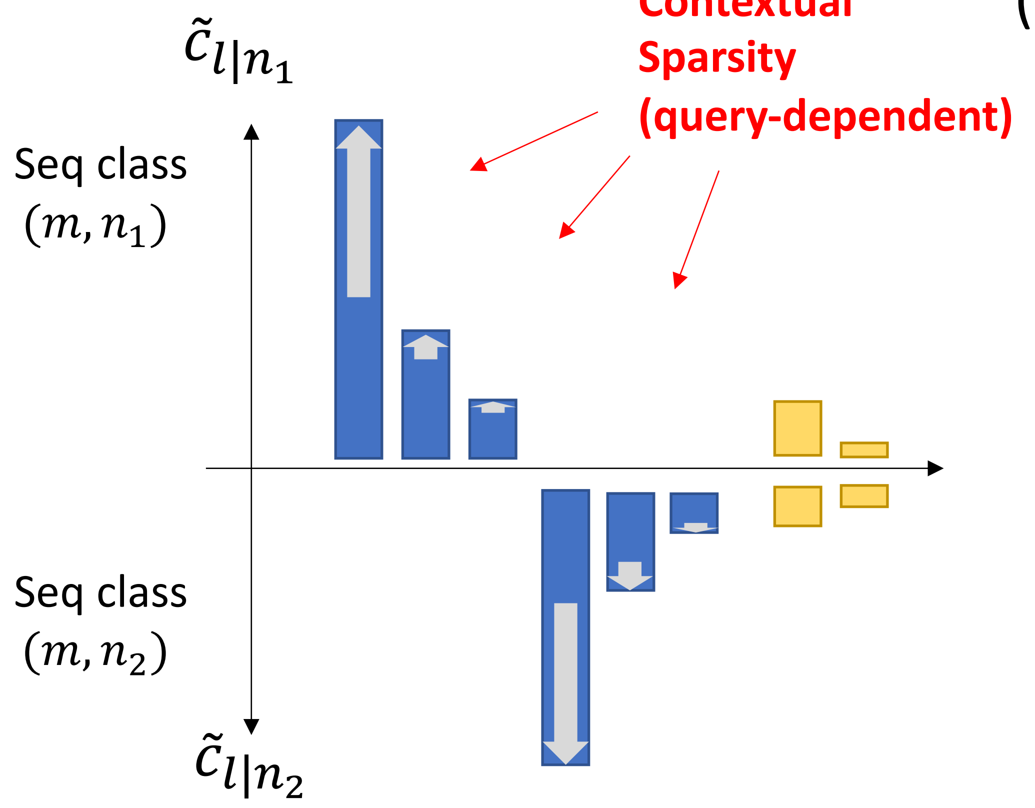
If l_0 is the dominant token: $r_{l_0/l|n}(0) > 0$ for all $l \neq l_0$ then

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

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:

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

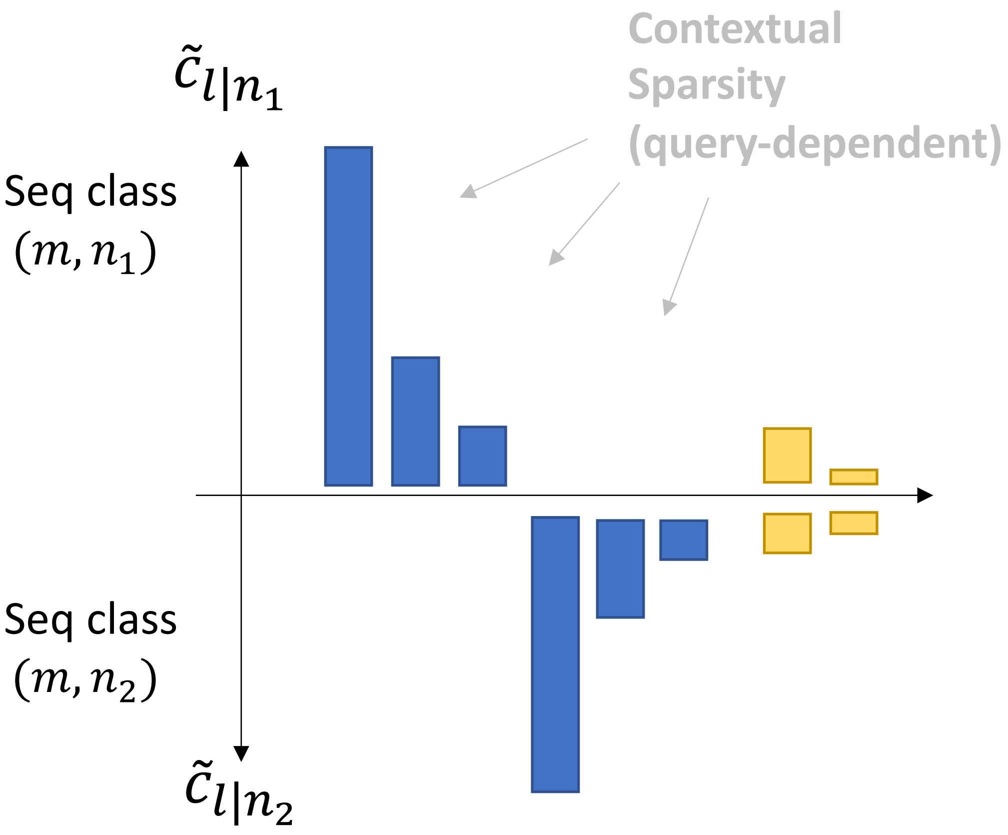
If l_0 is the dominant token: $r_{l_0/l|n}(0) > 0$ for all $l \neq l_0$ then

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

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

Overall Picture of the Training Dynamics

<u>Attention frozen</u>



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**:

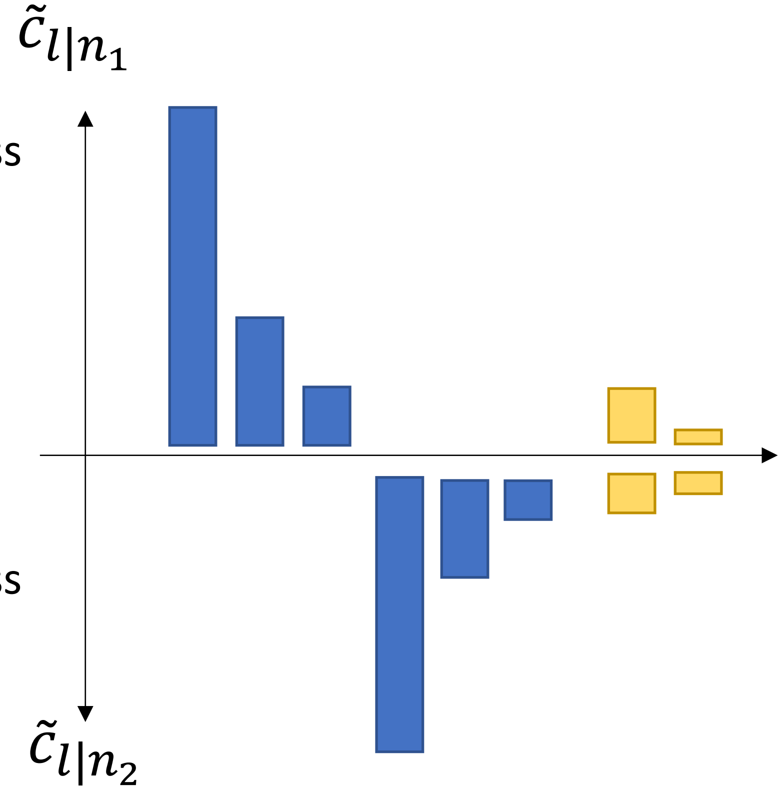
When $t \geq t_0 = O\left(\frac{2K\ln M}{\eta_Y}\right)$, $B_n(t) = O(\ln\ln t)$

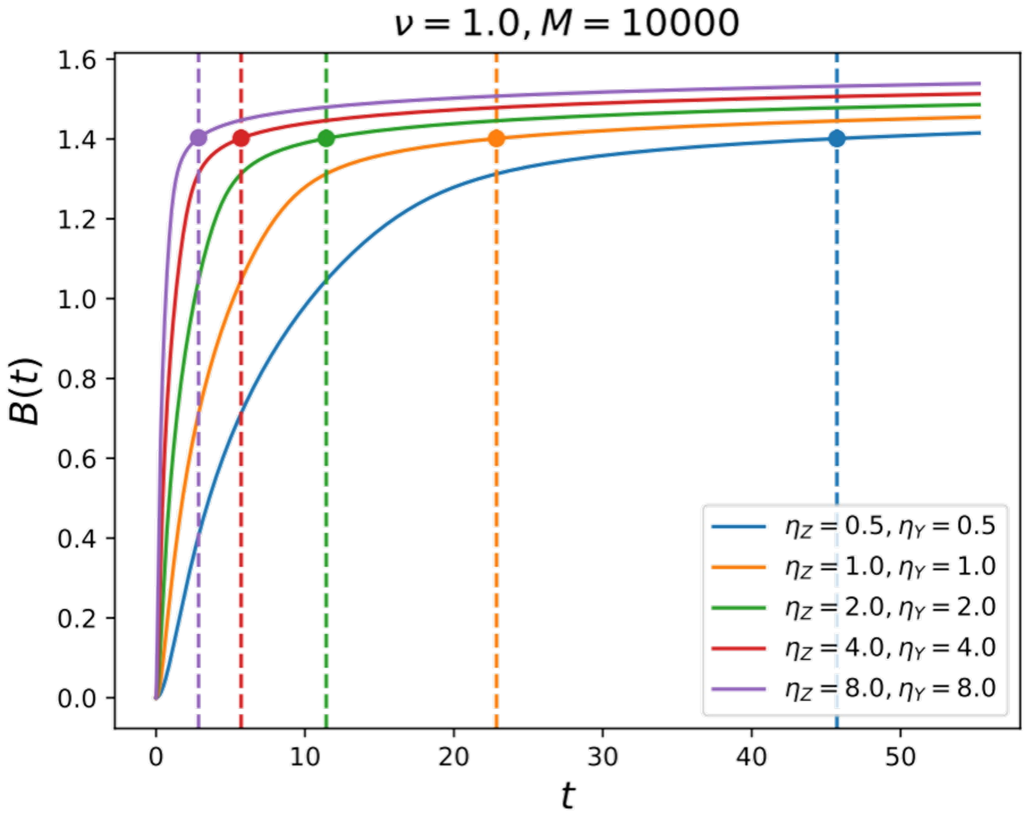
(1) η_Z and η_Y are large, $B_n(t)$ is large and attention is sparse

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

Overall Picture of the Training Dynamics

<u>Attention frozen</u>





Larger learning rate η_z leads to faster phase transition

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

Simple Real-world Experiments

WikiText2
(original parameterization)

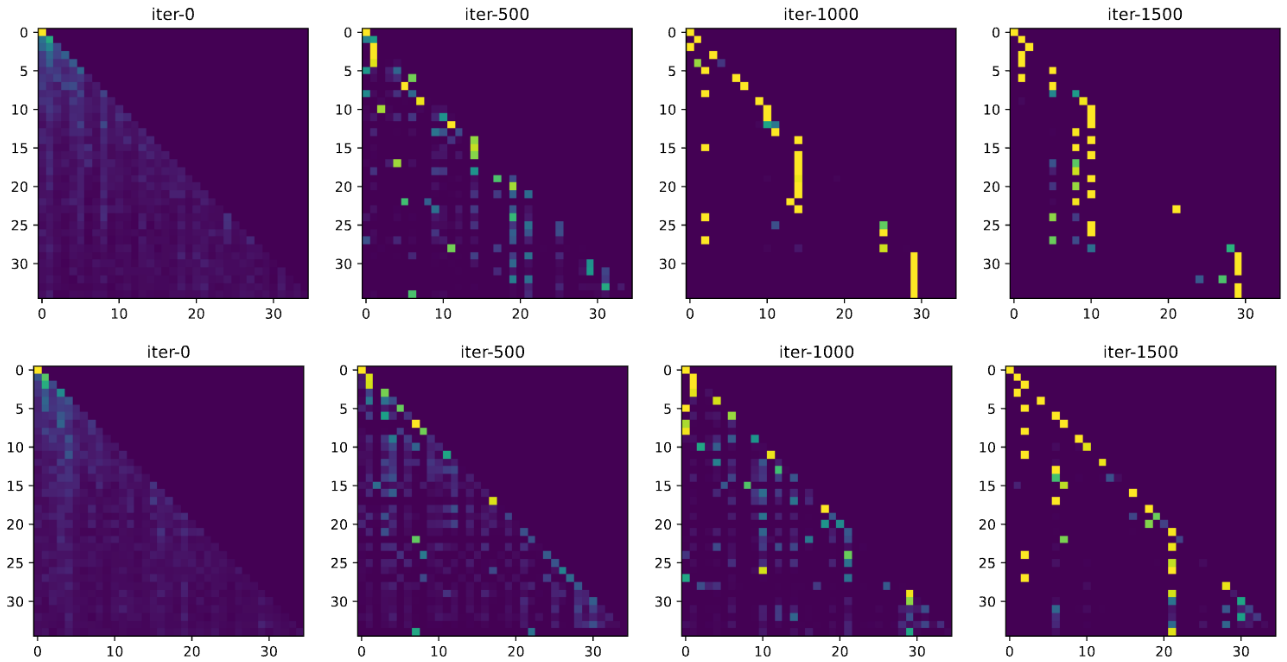


Figure 7: Attention patterns in the lowest self-attention layer for 1-layer (top) and 3-layer (bottom) Transformer trained on WikiText2 using SGD (learning rate is 5). Attention becomes sparse over training.

Further study of sparse attention
→ Deja Vu, H2O and StreamingLLM

[Z. Liu et al, *Deja vu: Contextual sparsity for efficient LLMs at inference time*, ICML'23 (oral)]
[Z. Zhang et al, *H2O: Heavy-Hitter Oracle for Efficient Generative Inference of Large Language Models*, NeurIPS'23]
[G. Xiao et al, *Efficient Streaming Language Models with Attention Sinks*, ICLR'24]

Deal with Reversal Curse

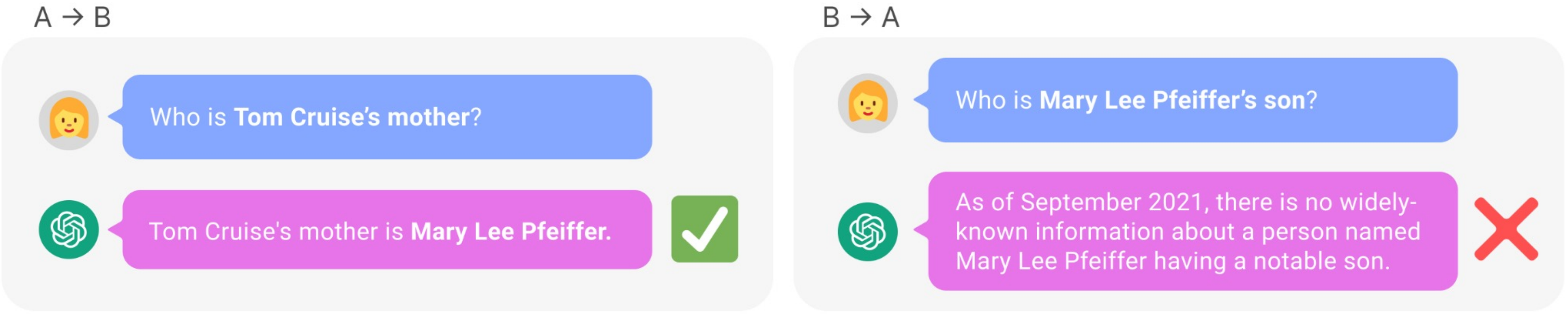


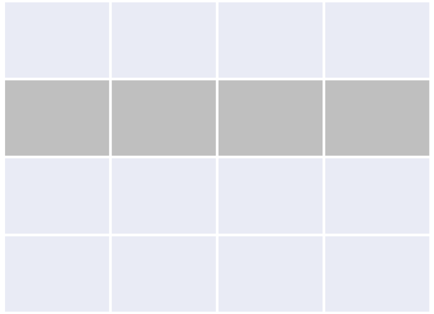
Figure 1: **Inconsistent knowledge in GPT-4.** GPT-4 correctly gives the name of Tom Cruise's mother (left). Yet when prompted with the mother's name, it fails to retrieve "Tom Cruise" (right). We hypothesize this ordering effect is due to the Reversal Curse. Models trained on "*A* is *B*" (e.g. "Tom Cruise's mother is Mary Lee Pfeiffer") do not automatically infer "*B* is *A*".

[L. Berglund et al, *The Reversal Curse: LLMs trained on "A is B" fail to learn "B is A"*, ICLR 2024]

How to explain “Reversal Curse”?

$Z = UW_Q W_K^T U^T$ pairwise logits of self-attention matrix,

is **not** symmetric

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

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

facebook Artificial Intelligence

[H. Zhu et al, *Towards a Theoretical Understanding of the 'Reversal Curse' via Training Dynamics,* arXiv’24]

You only learn what you see in the training set

Theorem 3 (Reversal curse). *Assume we run SGD with batch size 1, and assume* $M \gg 100$ *and* $\frac{1}{M^{0.99}} \ll \eta_Y < 1$. *Let* $t \gtrsim \frac{N \ln M}{\eta_Y}$ *denote the time step which also satisfies* $\ln t \gtrsim \ln(NM/\eta_Y)$. *For training sequence* $(x_1, x_2, x_3) \in \mathcal{D}_{train}$ *at time* t, *we have*

$$p_{\theta(t)}(x_3|x_1, x_2) \geq 1 - \frac{M-1}{2\left(\frac{M\eta_Y t}{N}\right)^c} \xrightarrow{t\to\infty} 1$$

for some constant $c > 0$, *and for any test sequence* $(x_1, x_2, x_3) \in \mathcal{D}_{test}$ *that is not included the training set* $\mathcal{D}_{train}$, *we have*

$$p_{\theta(t)}(x_3|x_1, x_2) \leq \frac{1}{M}.$$

[H. Zhu et al, *Towards a Theoretical Understanding of the 'Reversal Curse' via Training Dynamics,* arXiv'24]

"Chain-of-thoughts" reasoning

Theorem 4 (Necessity of chain-of-thought)**.** *Assume we run SGD with batch size 1, and assume $M \gg 100$ and $\frac{1}{M^{0.99}} \ll \eta_Y < 1$. Let $t \gtrsim \frac{N \ln M}{\eta_Y}$ denote the time step which also satisfies $\ln t \gtrsim \ln(NM/\eta_Y)$. For any test index $i \in \mathcal{I}_{test}$, we have*

$$p_{\theta(t)}(\boldsymbol{B}_i | \boldsymbol{A}_i \rightarrow) \geq 1 - \frac{M-1}{2\left(\frac{M\eta_Y t}{N}\right)^c}, \qquad p_{\theta(t)}(\boldsymbol{C}_i | \boldsymbol{B}_i \rightarrow) \geq 1 - \frac{M-1}{2\left(\frac{M\eta_Y t}{N}\right)^c}$$

for some constant $c > 0$ and

$$p_{\theta(t)}(\boldsymbol{C}_i | \boldsymbol{A}_i \rightsquigarrow) \leq \frac{1}{M}.$$

[H. Zhu et al, *Towards a Theoretical Understanding of the 'Reversal Curse' via Training Dynamics,* arXiv'24]

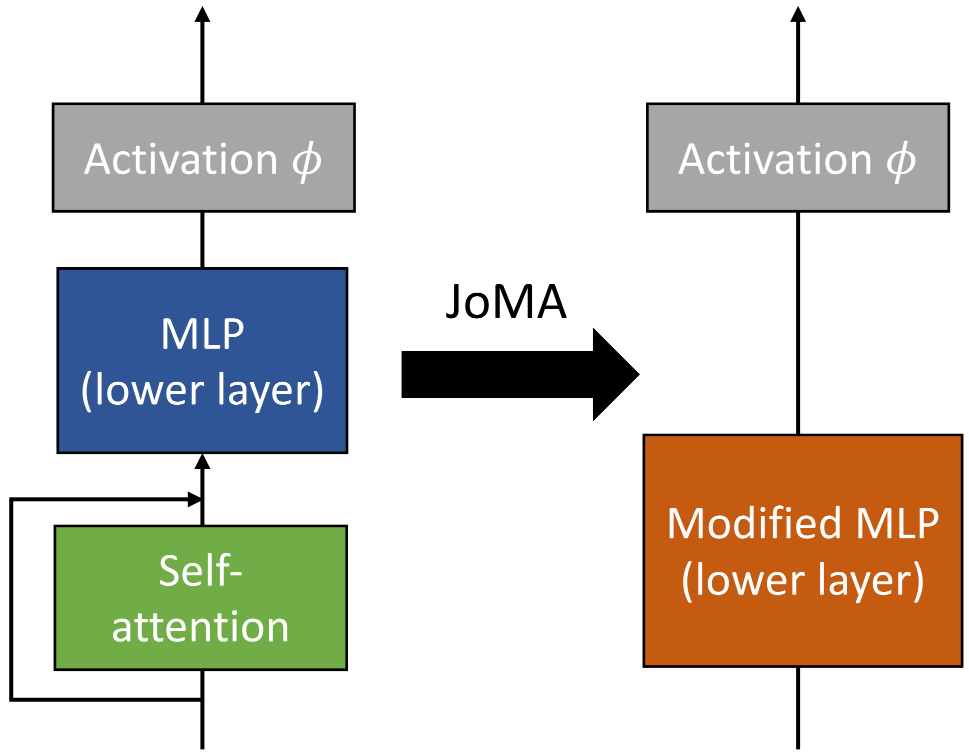
How to get rid of the assumptions?

- 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

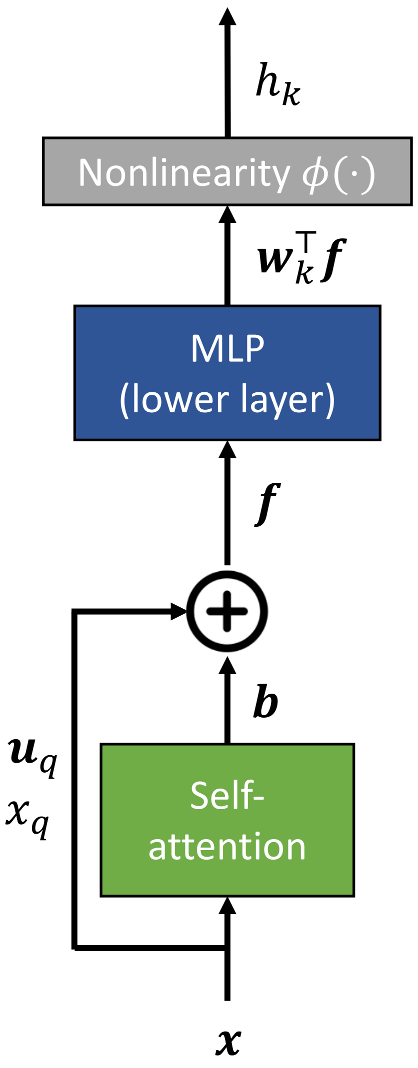


Main Contributions:

1. Find a joint dynamics that connects MLP with self-attention.
2. Understand self-attention behaviors for linear/nonlinear activations.
3. Explain how data hierarchy is learned in multi-layer Transformers.

[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

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

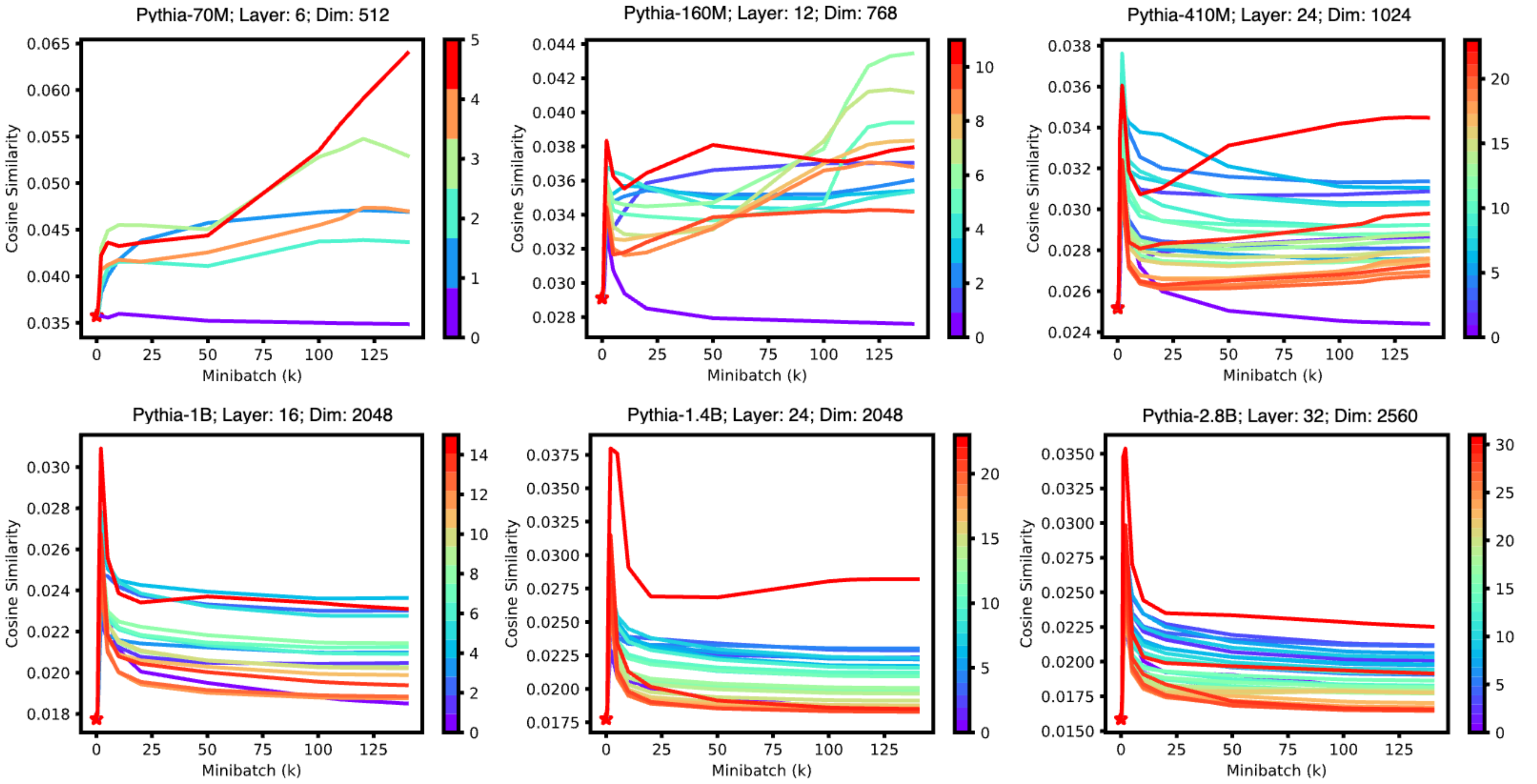
SoftmaxAttn: $b_l = \frac{x_l e^{z_{ql}}}{\sum_l x_l e^{z_{ql}}}$

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.



facebook Artificial Intellig

JoMA Dynamics

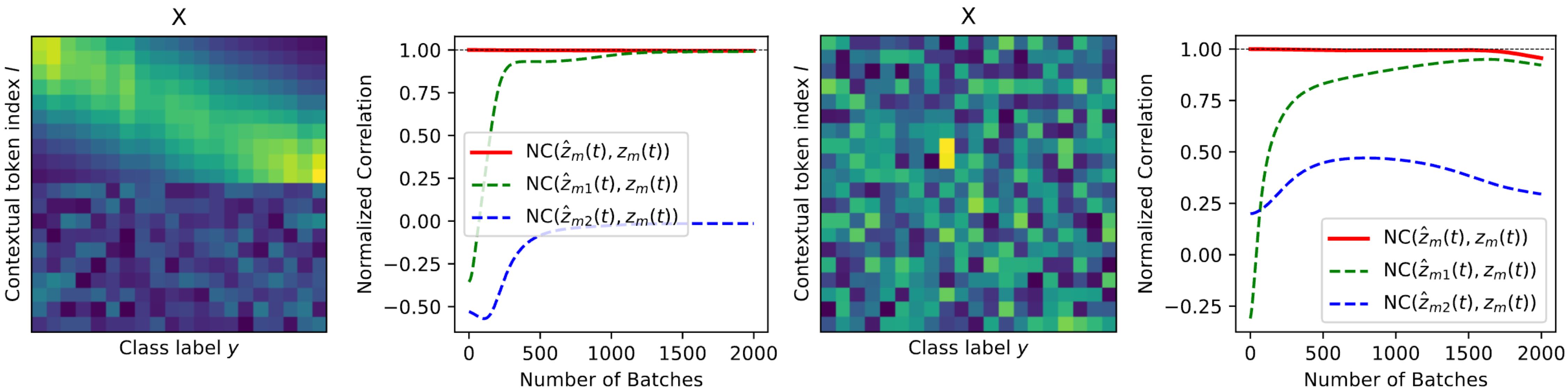
Theorem 1 (JoMA). *Let* $\boldsymbol{v}_k := U_C^\top \boldsymbol{w}_k$, *then the dynamics of Eqn.* $\boxed{3}$ *satisfies the invariants:*

- *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.

Verification of JoMA dynamics



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

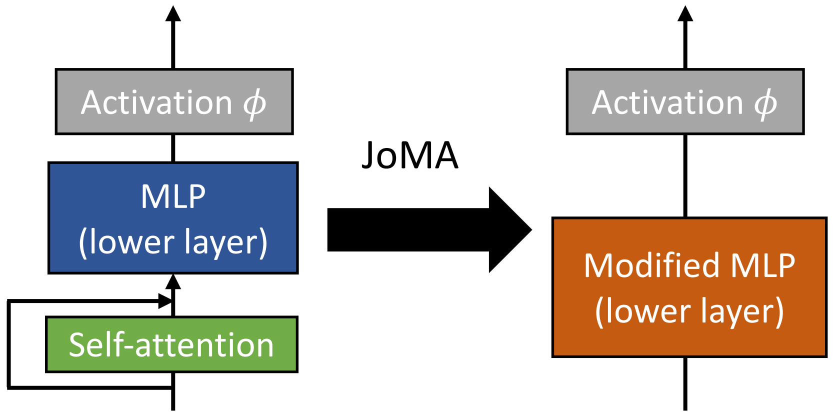
$\hat{\boldsymbol{z}}_m(t)$: Estimated attention logits by JoMA

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

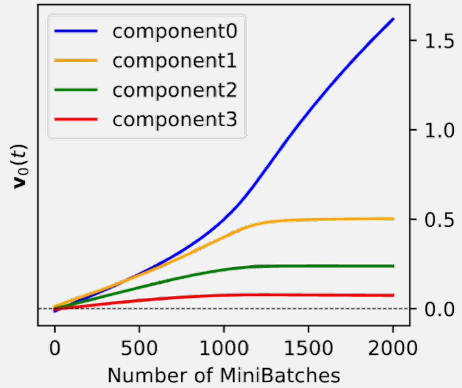
facebook Artificial Intelligence

Implication of Theorem 1

Key idea: folding self-attention into MLP
→ A Transformer block becomes a modified MLP

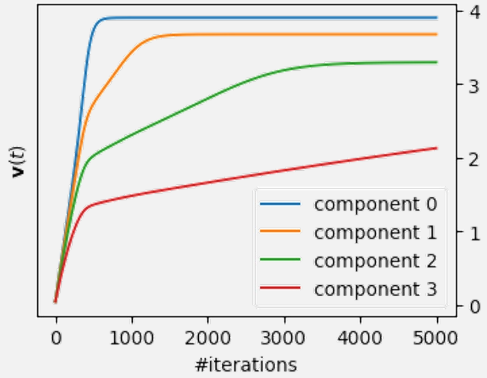


Linear case ($\phi = \mathrm{Id}, K = 1$)



Most salient feature takes all
(Attention becomes sparser)

Nonlinear case (ϕ nonlinear, $K = 1$)



Most salient feature grows, and others catch up
(Attention becomes sparser and denser)

Saliency is defined as $\Delta_{lm} = \mathbb{E}[g|l,m] \cdot \mathbb{P}[l|m]$

Discriminancy **CoOccurrence**

$\Delta_{lm} \approx 0$: **Common** tokens
$|\Delta_{lm}|$ large: **Distinct** tokens

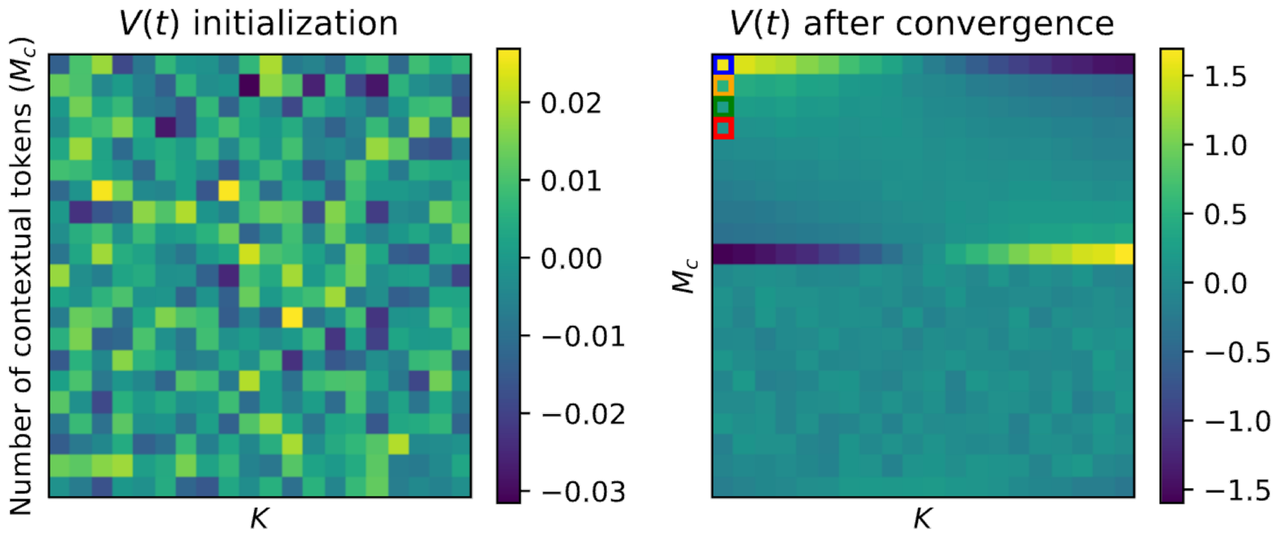
JoMA for Linear Activation

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

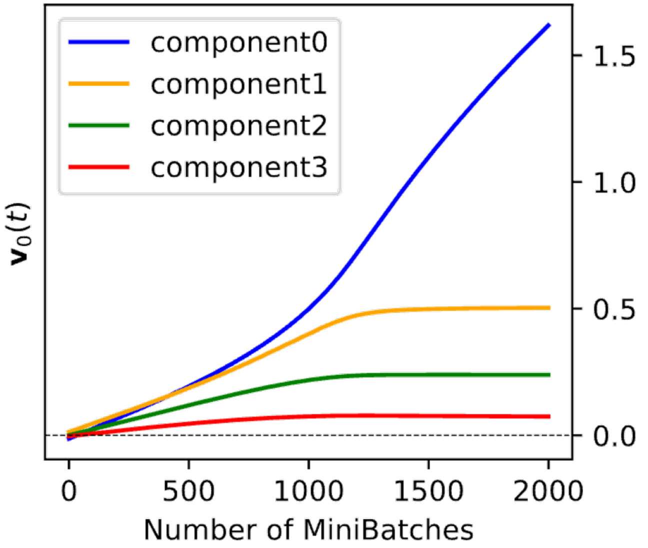
Theorem 2

We can prove $\frac{\mathrm{erf}(v_l(t)/2)}{\Delta_{lm}} = \frac{\mathrm{erf}(v_{l'}(t)/2)}{\Delta_{l'm}}$ $\qquad \mathrm{erf}(x) = \frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}\mathrm{d}t \in [-1,1]$

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



Attention becomes sparser
(Consistent with Scan&Snap)



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

What if we have more nodes ($K > 1$)?

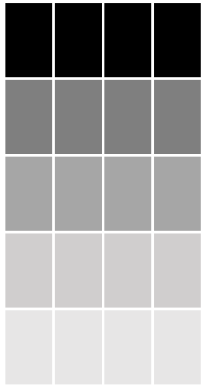
- $V = U_C^\top W \in \mathbb{R}^{M_c \times K}$ and the dynamics becomes

$$\dot{V} = \frac{1}{A}\mathrm{diag}\left(\exp\left(\frac{V \circ V}{2}\right)\mathbf{1}\right)\Delta \qquad \Delta = [\Delta_1, \Delta_2, \dots, \Delta_K], \qquad \Delta_k = \mathbb{E}[g_k \boldsymbol{x}]$$

We can prove that V gradually becomes low rank

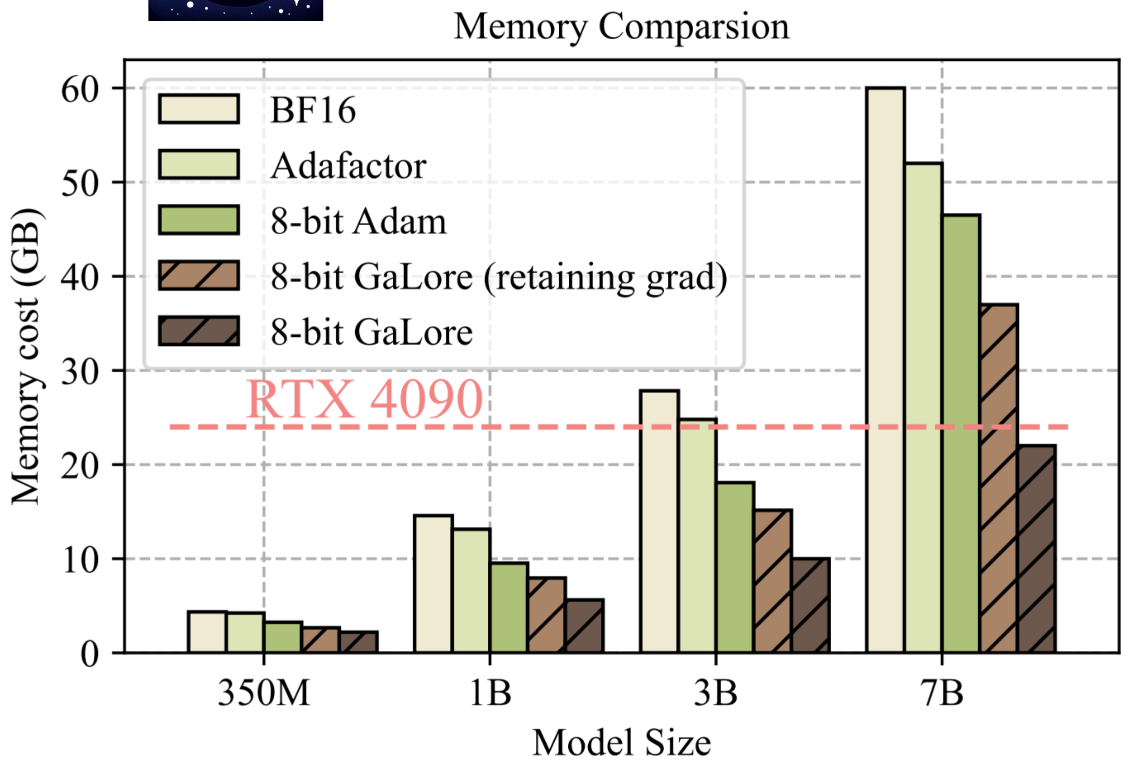
- The growth rate of each row of V varies widely.

$V(t) \rightarrow$



Due to $\exp\left(\frac{V \circ V}{2}\right)$, the weight gradient $\dot{V}$ can be even more low-rank → GaLore

GaLore: Pre-training 7B model on RTX 4090 (24G)



	Rank	Retain grad	Memory	Token/s
8-bit AdamW		Yes	40GB	1434
8-bit GaLore	16	Yes	28GB	1532
8-bit GaLore	128	Yes	29GB	1532
16-bit GaLore	128	Yes	30GB	**1615**
16-bit GaLore	128	No	**18GB**	1587
8-bit GaLore	1024	Yes	36GB	1238

* SVD takes around 10min for 7B model, but runs every T=500-1000 steps.

Third-party evaluation by @llamafactory_ai

[J. Zhao et al, *GaLore: Memory-Efficient LLM Training by Gradient Low-Rank Projection*, ICML'24]

Memory Saving with GaLore

Algorithm 1: GaLore, PyTorch-like

```
for weight in model.parameters():
    grad = weight.grad
    # original space -> compact space
    lor_grad = project(grad)
    # update by Adam, Adafactor, etc.
    lor_update = update(lor_grad)
    # compact space -> original space
    update = project_back(lor_update)
    weight.data += update
```

GaLore

$G_t \leftarrow -\nabla_W \phi(W_t)$

If t % T == 0:

  Compute $P_t = \mathrm{SVD}(G_t) \in \mathbb{R}^{m \times r}$

$R_t \leftarrow P_t^T G_t$ *{project}*

$\tilde{R}_t \leftarrow \rho(R_t)$ *{Adam in low-rank}*

$\tilde{G}_t \leftarrow P_t \tilde{R}_t$ *{project-back}*

$W_{t+1} \leftarrow W_t + \eta \tilde{G}_t$

Memory Usage	Weight (W)	Optim States (M_t, V_t)	Projection (P)	Total
Full-rank	mn	$2mn$	0	$3mn$
Low-rank adaptor	$mn + mr + nr$	$2(mr + nr)$	0	$mn + 3(mr + nr)$
GaLore	mn	$2nr$	mr	$mn + mr + 2nr$

W_t R_t P_t

Pre-training Results (LLaMA 7B)

Params	Hidden	Intermediate	Heads	Layers	Steps	Data amount
60M	512	1376	8	8	10K	1.3 B
130M	768	2048	12	12	20K	2.6 B
350M	1024	2736	16	24	60K	7.8 B
1 B	2048	5461	24	32	100K	13.1 B
7 B	4096	11008	32	32	150K	19.7 B

	Mem	**40K**	**80K**	**120K**	**150K**
8-bit GaLore	18G	17.94	15.39	14.95	14.65
8-bit Adam	26G	18.09	15.47	14.83	14.61
Tokens (B)		5.2	10.5	15.7	19.7

* Experiments are conducted on 8 x 8 A100

	60M	**130M**	**350M**	**1B**
Full-Rank	34.06 (0.36G)	25.08 (0.76G)	18.80 (2.06G)	15.56 (7.80G)
GaLore	**34.88** (0.24G)	**25.36** (0.52G)	**18.95** (1.22G)	**15.64** (4.38G)
Low-Rank	78.18 (0.26G)	45.51 (0.54G)	37.41 (1.08G)	142.53 (3.57G)
LoRA	34.99 (0.36G)	33.92 (0.80G)	25.58 (1.76G)	19.21 (6.17G)
ReLoRA	37.04 (0.36G)	29.37 (0.80G)	29.08 (1.76G)	18.33 (6.17G)
r/d_{model}	128 / 256	256 / 768	256 / 1024	512 / 2048
Training Tokens	1.1B	2.2B	6.4B	13.1B

* On LLaMA 1B, ppl is better (~14.97) with ½ rank (1024/2048)

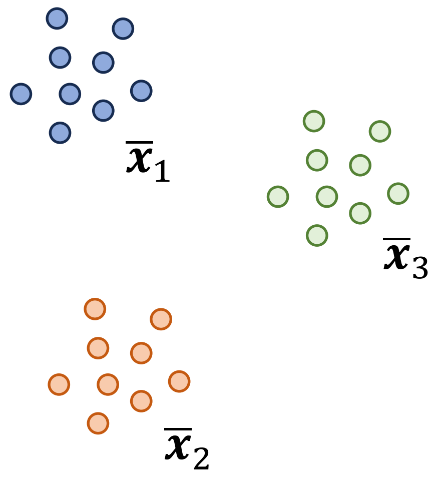
JoMA for Nonlinear Activation

Theorem 3

If $\boldsymbol{x}$ is sampled from a mixture of C isotropic distributions, (i.e., "local salient/non-salient map"), then

$$\dot{\boldsymbol{v}} = \frac{1}{\|\boldsymbol{v}\|_2}\sum_c a_c\theta_1(r_c)\bar{\boldsymbol{x}}_c + \frac{1}{\|\boldsymbol{v}\|_2^3}\sum_c a_c\theta_2(r_c)\boldsymbol{v}$$

Here $a_c \coloneqq \mathbb{E}_{q=m,c}\left[g_{h_k}\right]\mathbb{P}[c]$, $r_c = \boldsymbol{v}^\top\bar{\boldsymbol{x}}_c + \int_0^t \mathbb{E}_{q=m}\left[g_{h_k}h'_k\right]\mathrm{d}t$, and θ_1 and θ_2 depends on nonlinearity



What does the dynamics look like?

$$\dot{\boldsymbol{v}} = (\boldsymbol{\mu} - \boldsymbol{v}) \circ \exp\left(\frac{\boldsymbol{v}^2}{2}\right)$$

$\boldsymbol{\mu} \sim \bar{\boldsymbol{x}}_c$: Critical point due to nonlinearity (one of the cluster centers)

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)

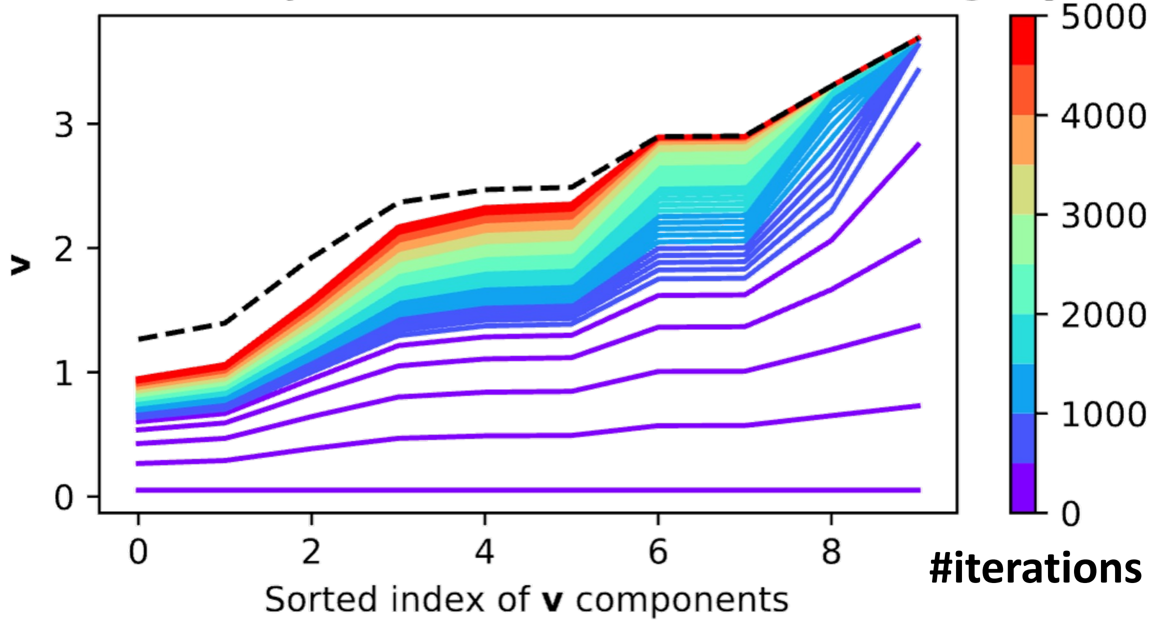
Theorem 4

Salient components grow much faster than non-salient ones:

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

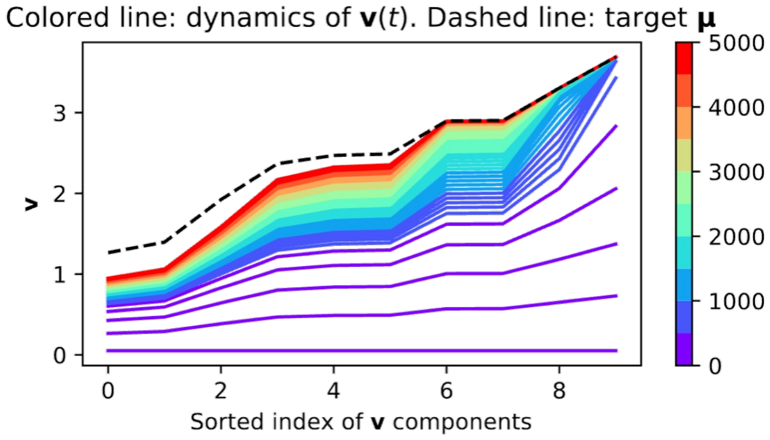
$$\mathrm{ConvergenceRate}(j) \coloneqq \ln 1/\delta_j(t)$$
$$\delta_j(t) \coloneqq 1 - v_j(t)/\mu_j$$

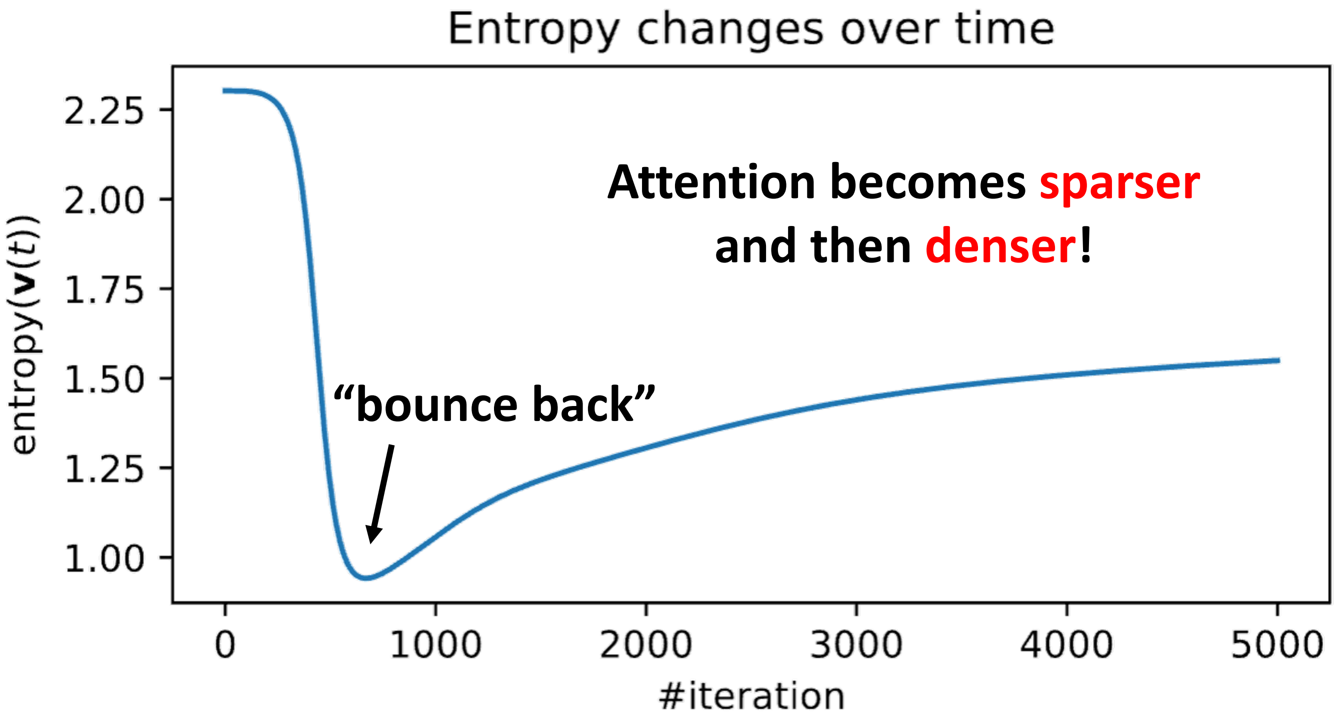
Colored line: dynamics of $\mathbf{v}(t)$. Dashed line: target $\boldsymbol{\mu}$



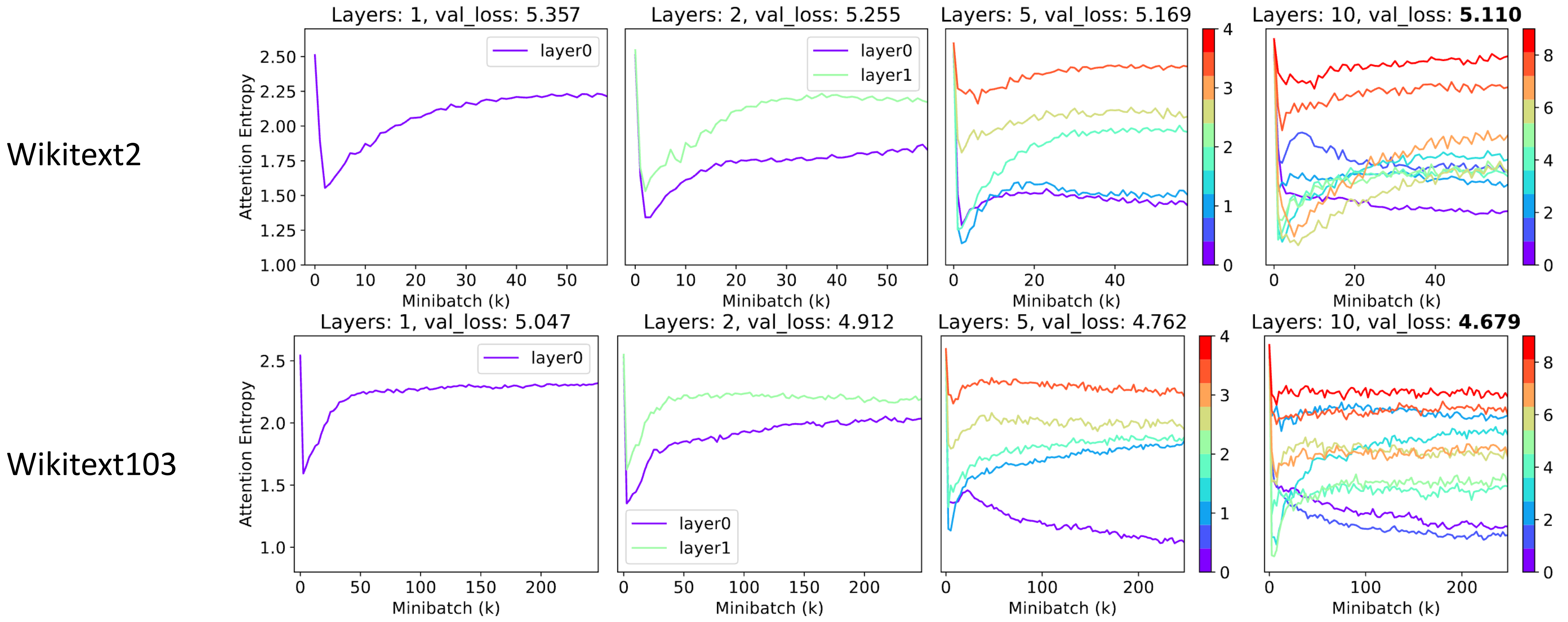
JoMA for Nonlinear activation

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

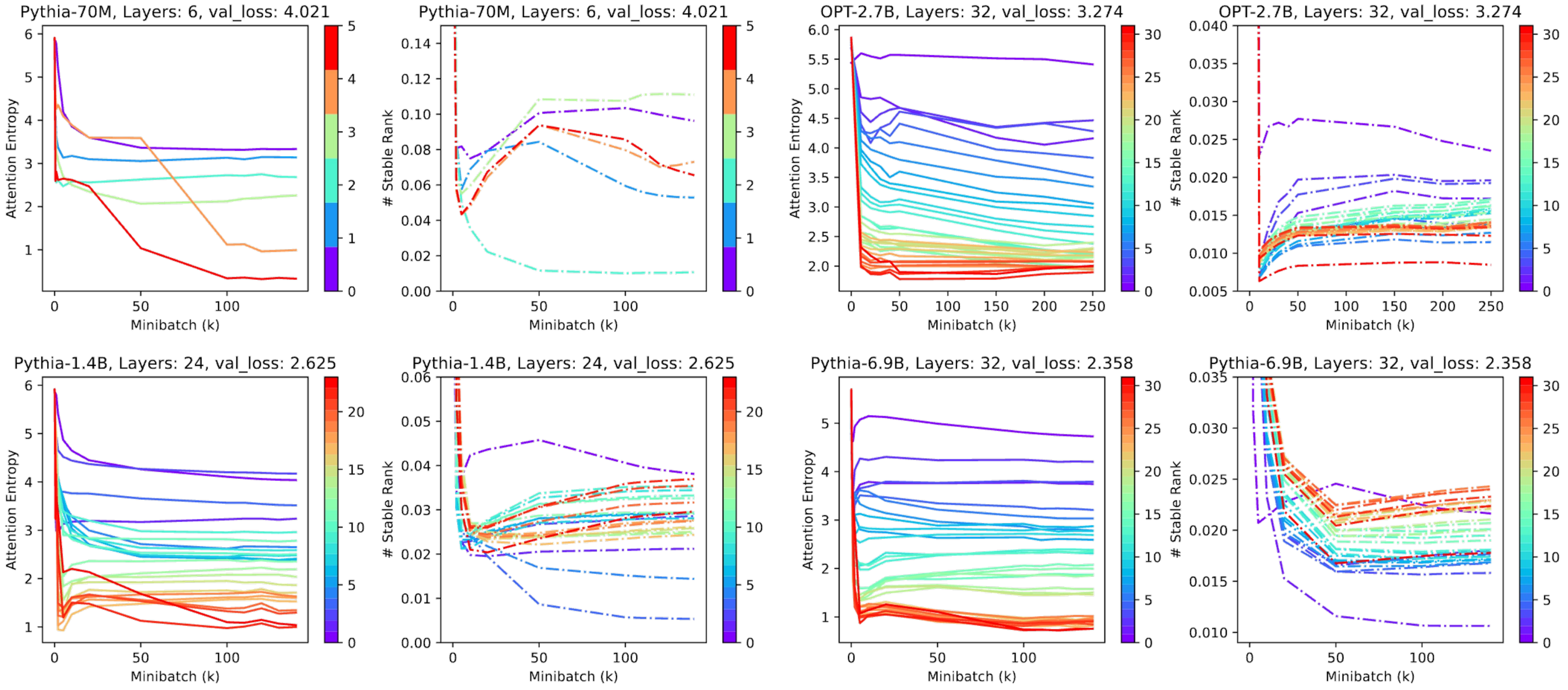




Real-world Experiments



Real-world Experiments



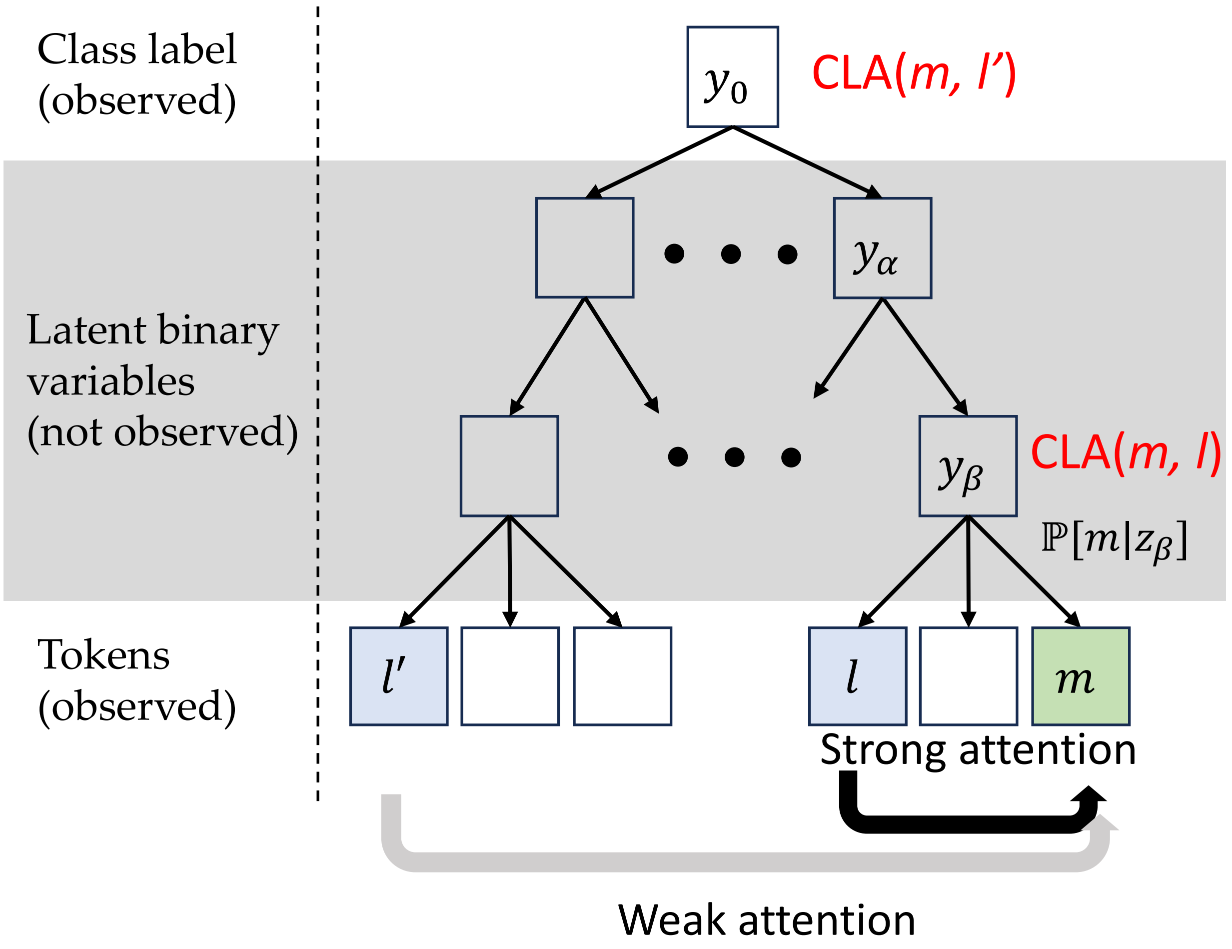
Stable Rank of the lower layer of MLP shows the “bouncing back” effects as well.

Why is this “bouncing back” property useful?

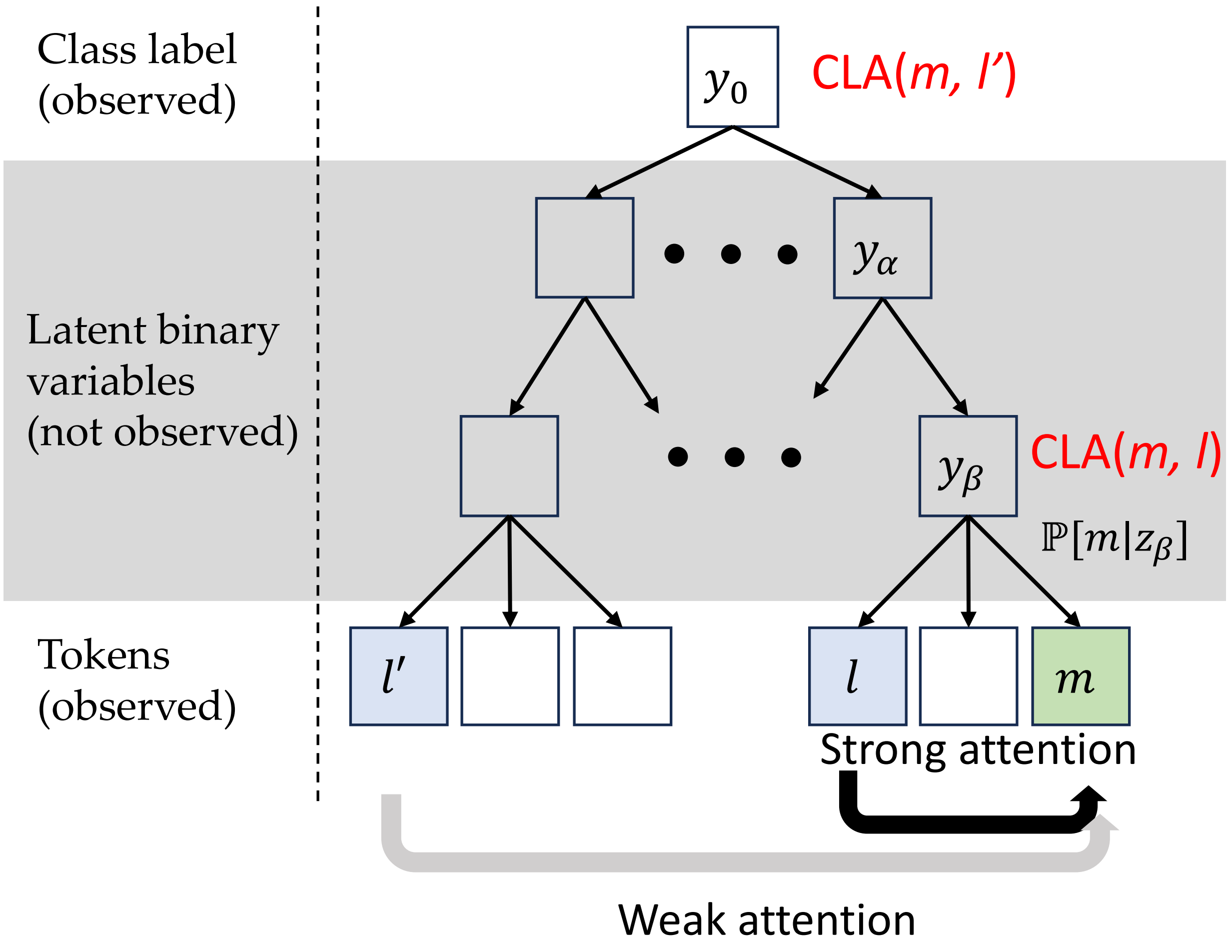
It seems that it only slows down the training??

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

Data Hierarchy & Multilayer Transformer



Data Hierarchy & Multilayer Transformer



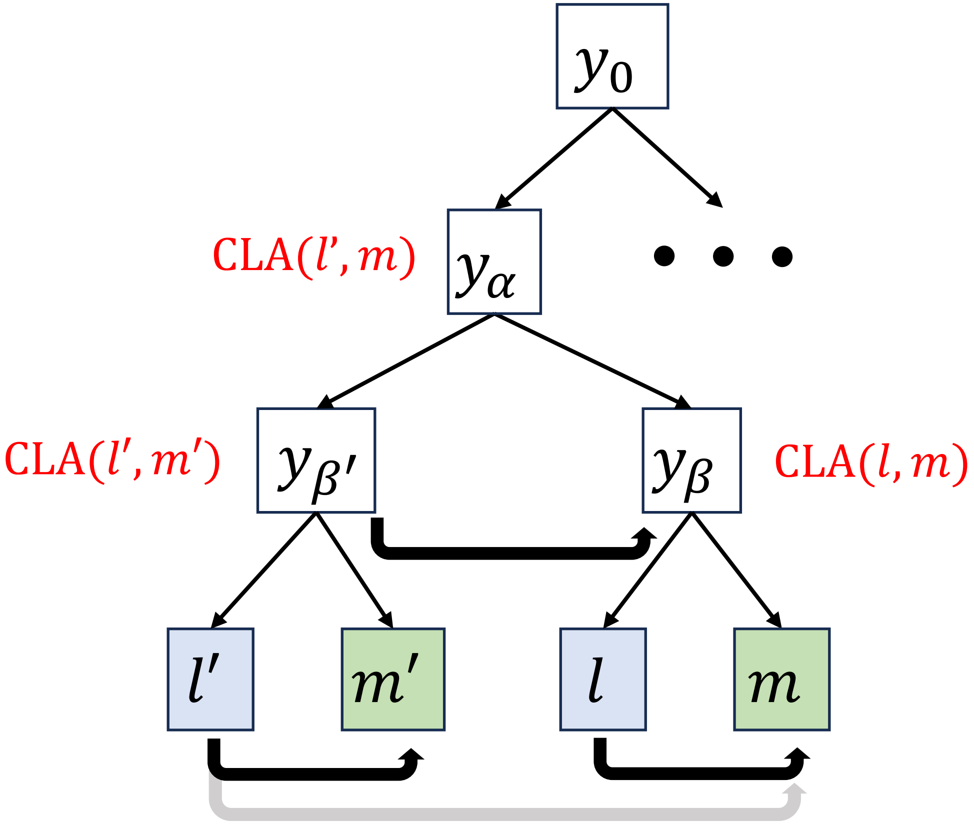
Theorem 5

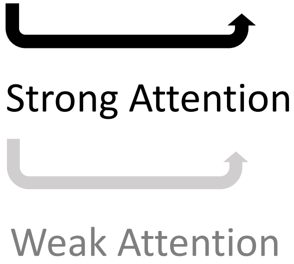
$$\mathbb{P}[l|m] \approx 1 - \frac{H}{L}$$

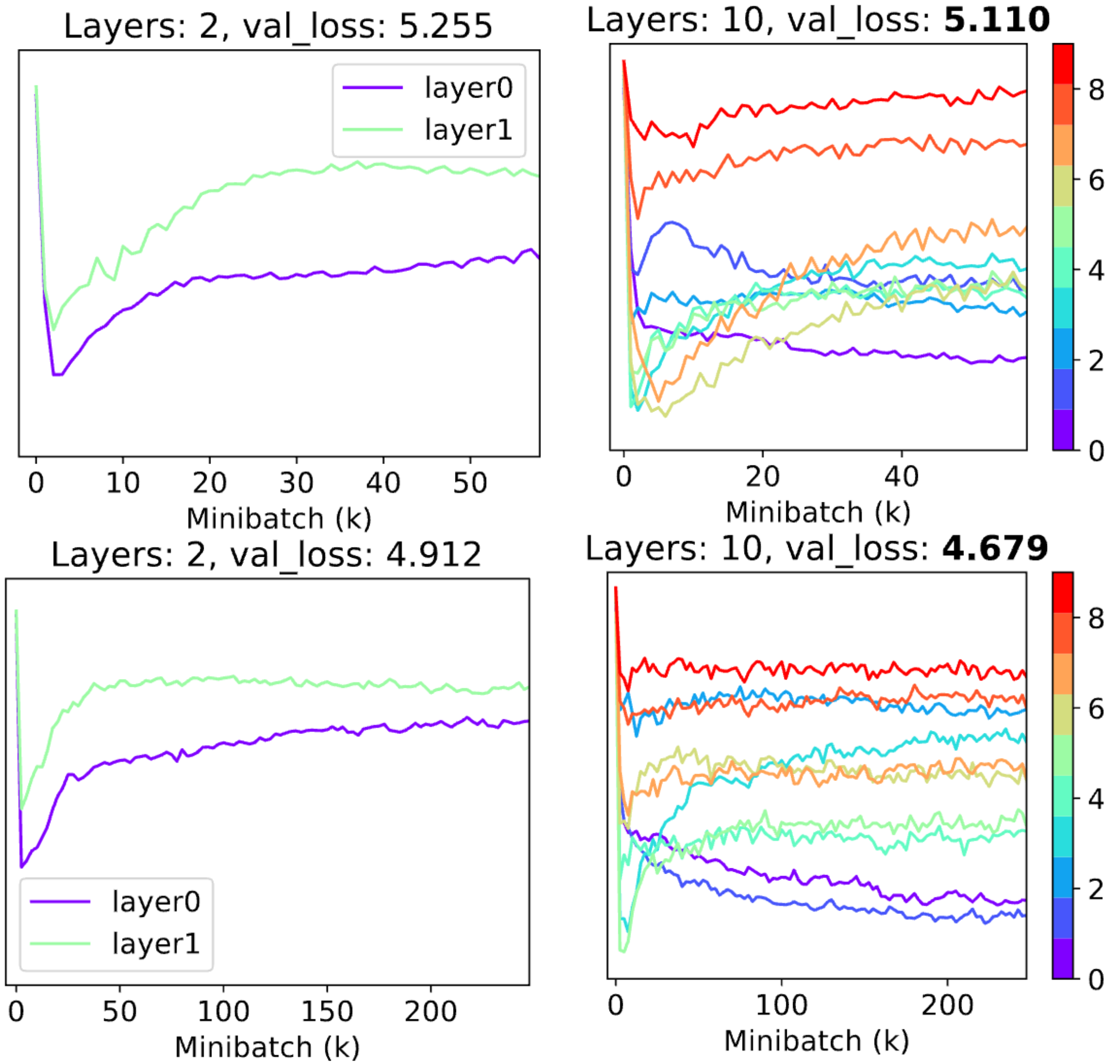
H: height of the common latent ancestor (CLA) of l & m

L: total height of the hierarchy

Deep Latent Distribution

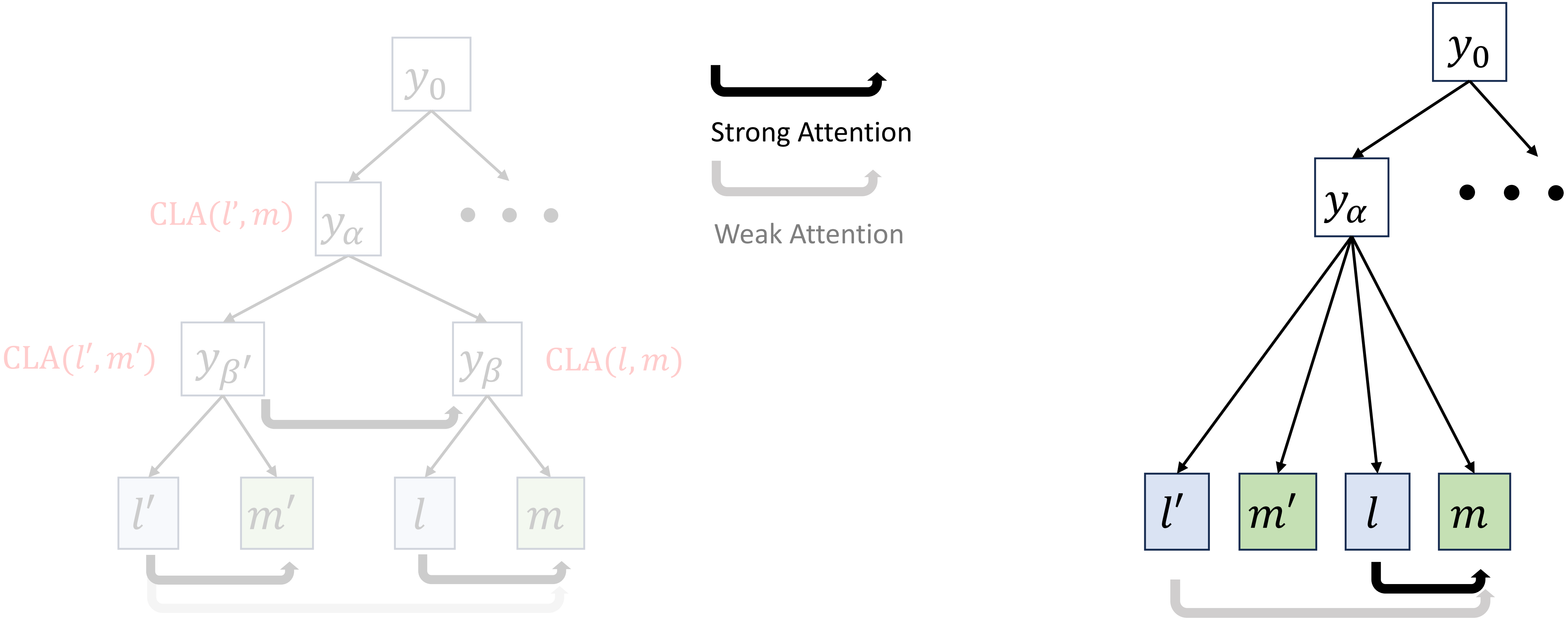




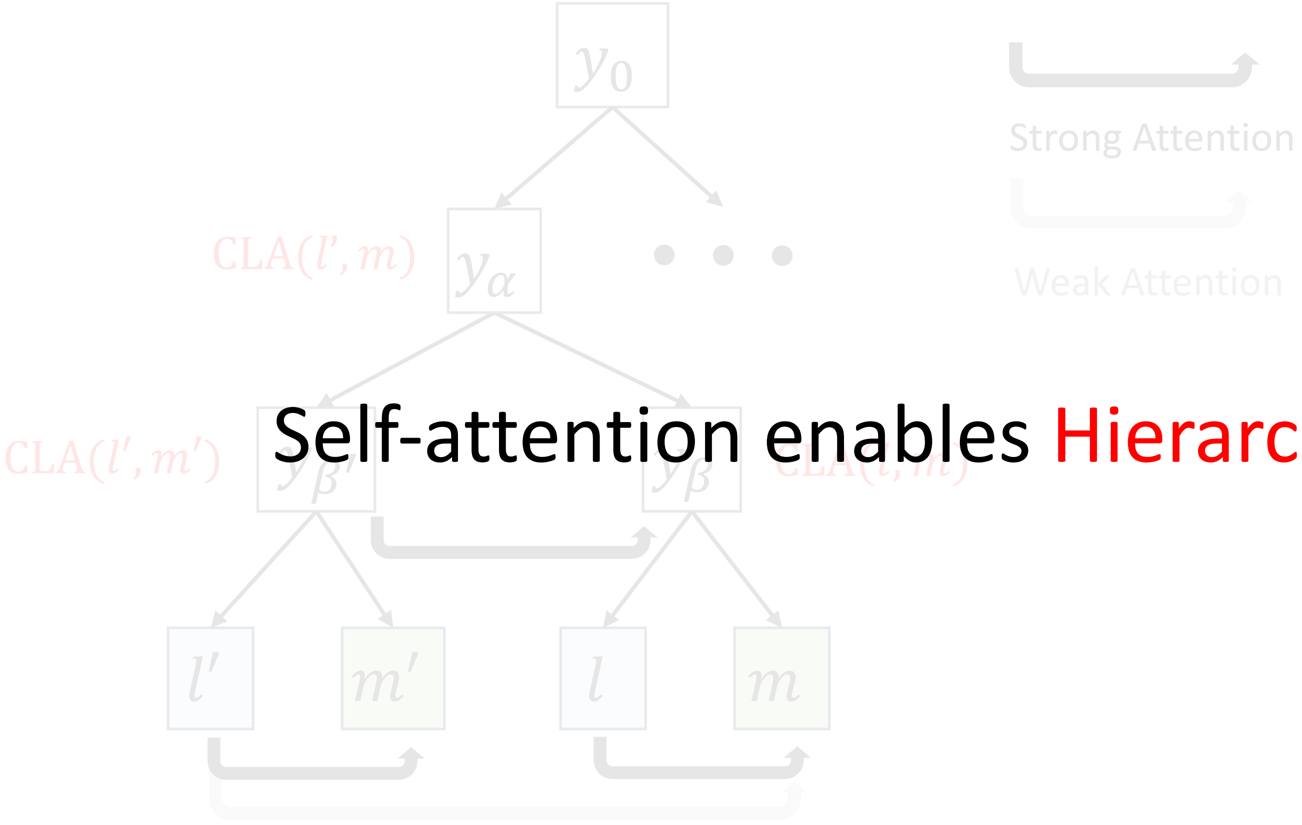


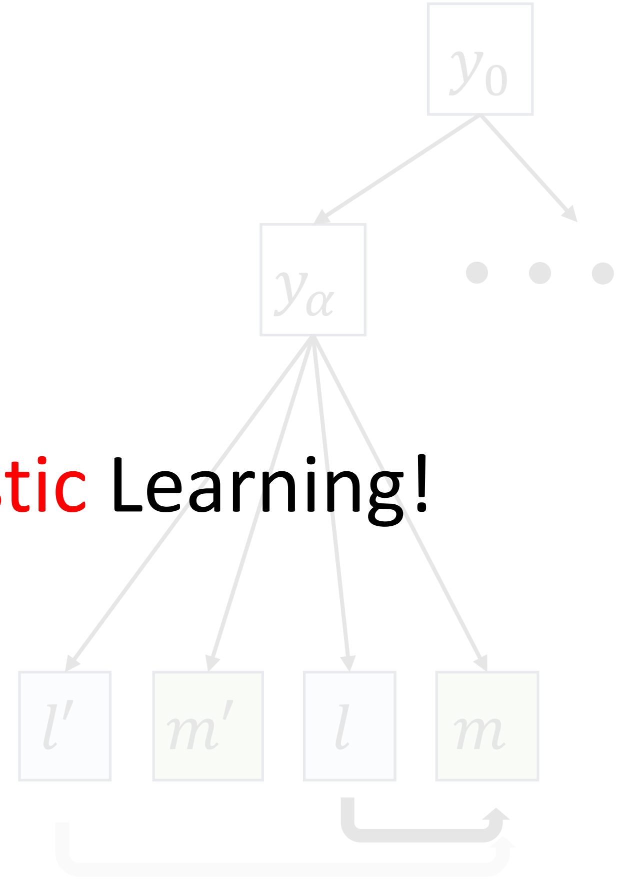
Learning the current hierarchical structure by
slowing down the association of tokens that are not directly correlated

Shallow Latent Distribution



Hierarchy-agnostic Learning





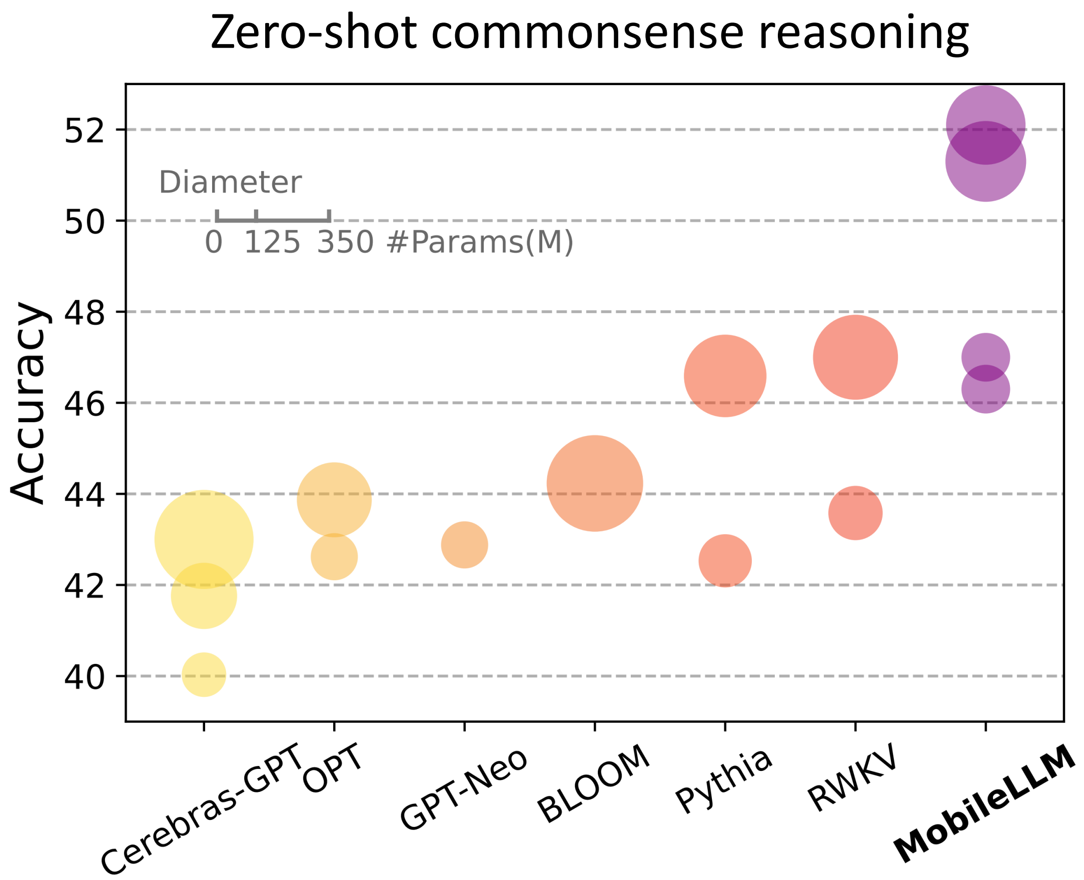
Self-attention enables Hierarchy-agnostic Learning!

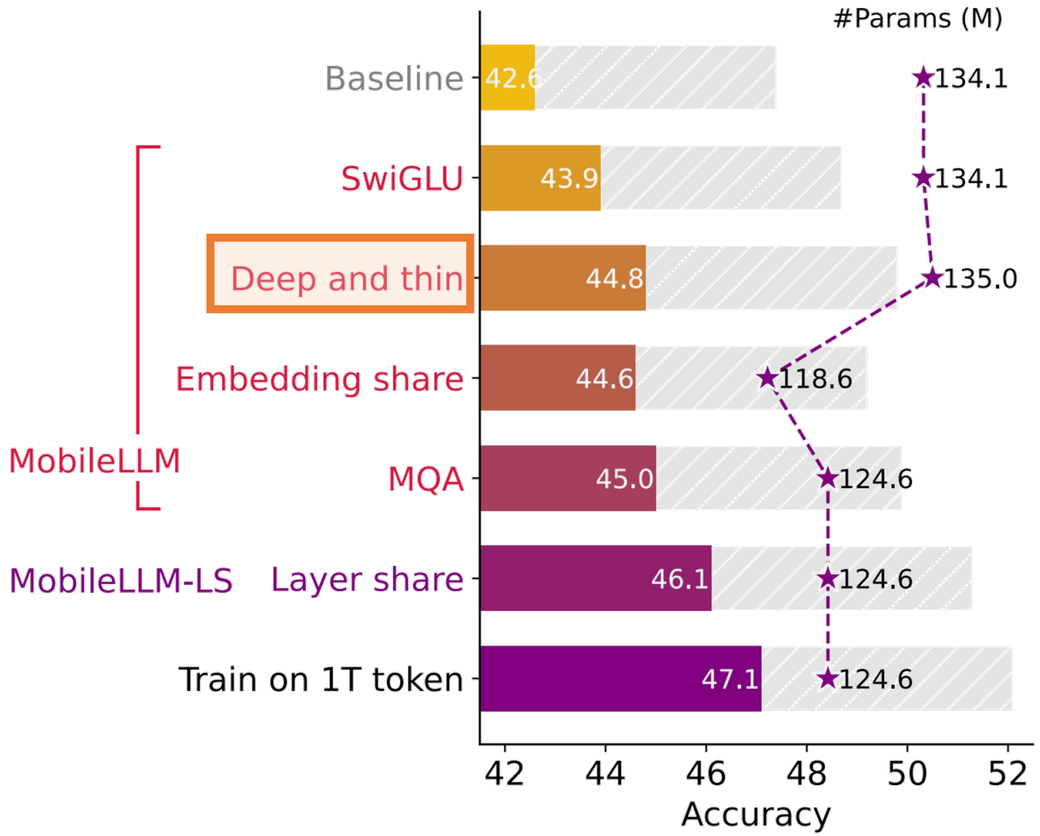
Verification of Hierarchical Intuitions

	$C = 20,\ N_{\text{ch}} = 2$		$C = 20,\ N_{\text{ch}} = 3$		$C = 30,\ N_{\text{ch}} = 2$	
(N_0, N_1)	(10, 20)	(20, 30)	(10, 20)	(20, 30)	(10, 20)	(20, 30)
NCorr ($s = 0$)	0.99 ± 0.01	0.97 ± 0.02	1.00 ± 0.00	0.96 ± 0.02	0.99 ± 0.01	0.94 ± 0.04
NCorr ($s = 1$)	0.81 ± 0.05	0.80 ± 0.05	0.69 ± 0.05	0.68 ± 0.04	0.73 ± 0.08	0.74 ± 0.03
	$C = 30\ N_{\text{ch}} = 3$		$C = 50,\ N_{\text{ch}} = 2$		$C = 50,\ N_{\text{ch}} = 3$	
(N_0, N_1)	(10, 20)	(20, 30)	(10, 20)	(20, 30)	(10, 20)	(20, 30)
NCorr ($s = 0$)	0.99 ± 0.01	0.95 ± 0.03	0.99 ± 0.01	0.95 ± 0.03	0.99 ± 0.01	0.95 ± 0.03
NCorr ($s = 1$)	0.72 ± 0.04	0.66 ± 0.02	0.58 ± 0.02	0.55 ± 0.01	0.64 ± 0.02	0.61 ± 0.04

Table 1: Normalized correlation between the latents and their best matched hidden node in MLP of the same layer. All experiments are run with 5 random seeds.

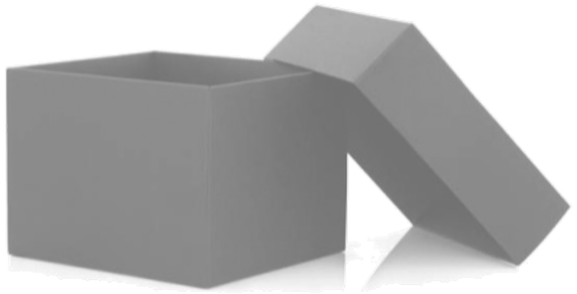
MobileLLM





[Z. Liu et al, *MobileLLM: Optimizing Sub-billion Parameter Language Models for On-Device Use Cases*, ICML'24]

Take away messages



- Architecture ✓ training dynamics ✓
- Nonlinearity is not formidable!
  - Transformer can be analyzed following gradient descent rules
- Property of self-attention
  - Attention becomes sparse over training
  - Inductive bias
    - Favor the learning of strong co-occurred tokens
    - Deter the learning of weakly co-occurred tokens, avoiding spurious correlation.
- Key insights lead to broad applications

Thanks!