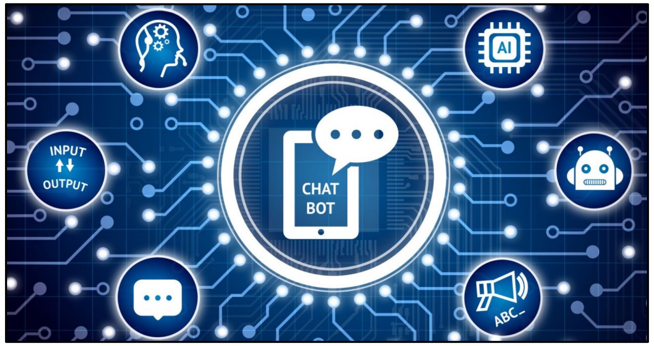
Understanding Foundational Models via the Lens of Training Dynamics

Yuandong Tian
Research Scientist and Manager

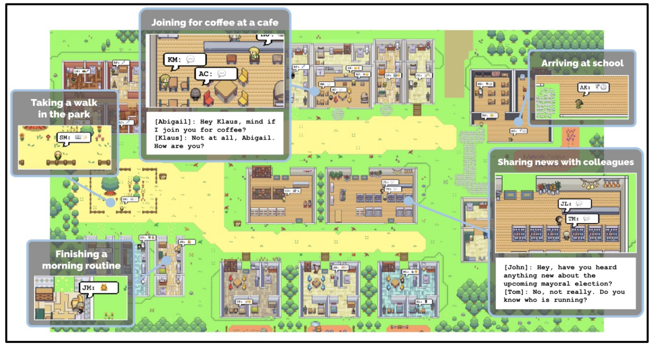
Meta AI (FAIR)

Large Language Models (LLMs)



Conversational AI

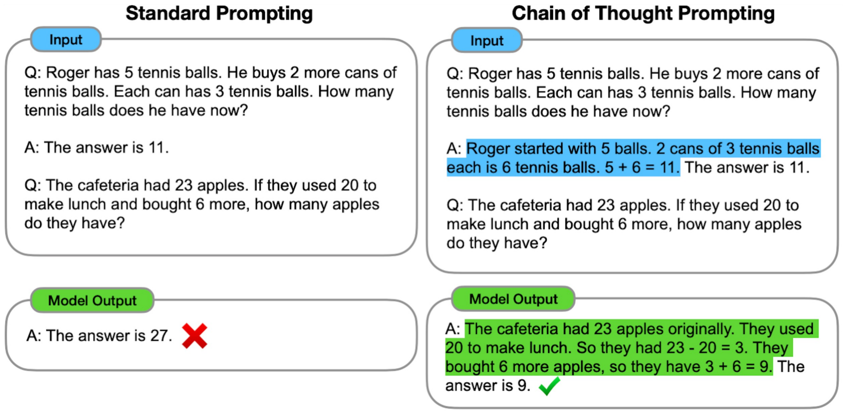
Content Generation


Joining for coffee at a cafe
KM:
AC:
Taking a walk in the park
SM:
Arriving at school
AK:
Sharing news with colleagues
JL:
TM:
Finishing a morning routine
JM:

[Abigail]: Hey Klaus, mind if I join you for coffee?
[Klaus]: Not at all, Abigail. How are you?

[John]: Hey, have you heard anything new about the upcoming mayoral election?
[Tom]: No, not really. Do you know who is running?

AI Agents



Standard Prompting

Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The answer is 27. ✗

Chain of Thought Prompting

Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. 5 + 6 = 11. The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

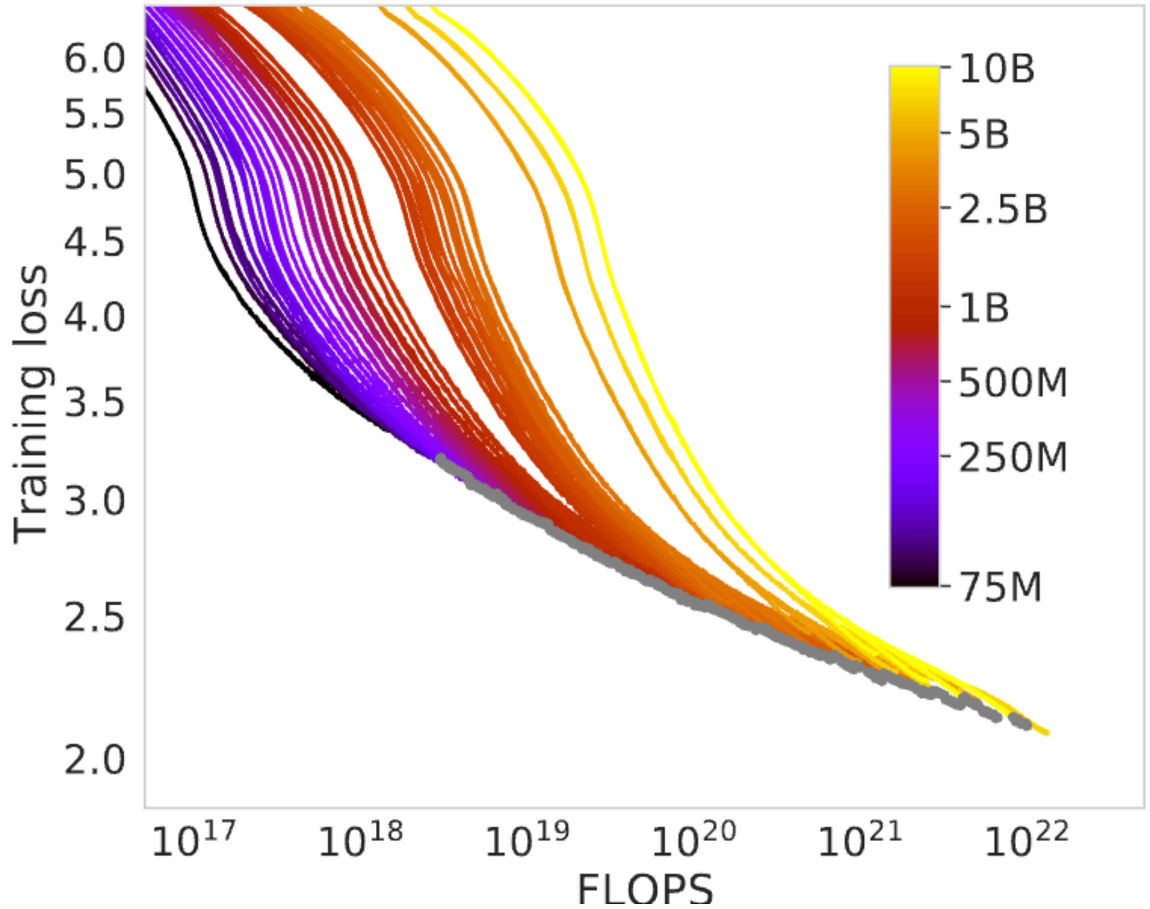
Model Output

A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had 23 - 20 = 3. They bought 6 more apples, so they have 3 + 6 = 9. The answer is 9. ✓

Reasoning

Planning

What does the future look like?



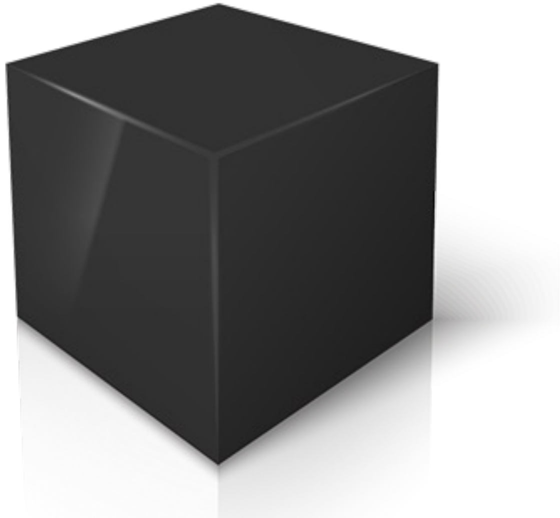
More data

More compute

Larger models

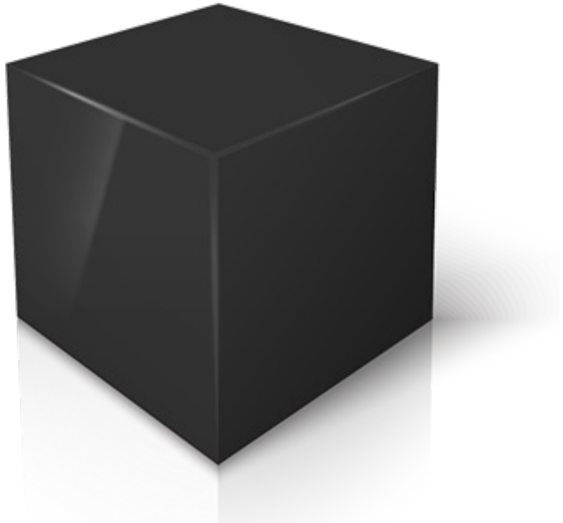
Are we going to blindly believe in scaling laws?

Black-box versus White-box



Black box

Black-box versus White-box



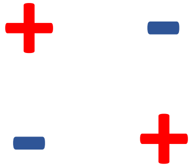
Black box

White box

Three Angles

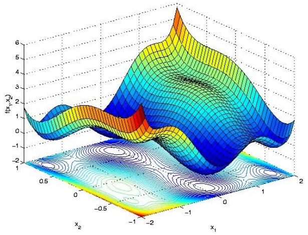
Understanding how Deep Models work

Expressibility



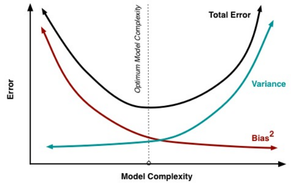
“Neural Network is a universal approximator”
“Deep Models can express functions more efficiently than shallow ones”

Optimization



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

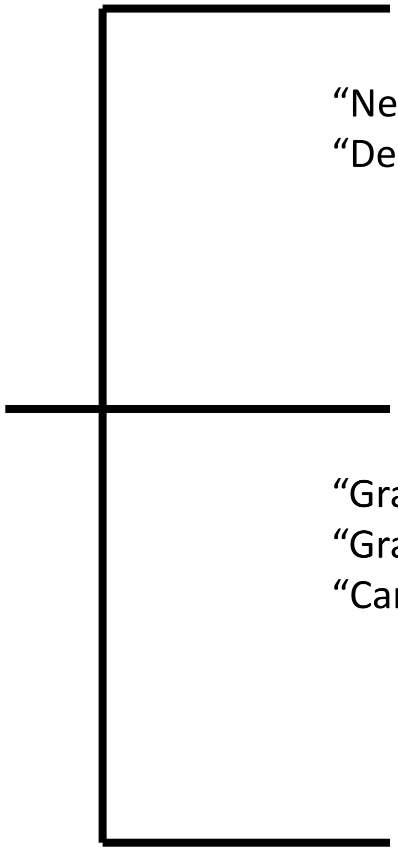
Generalization



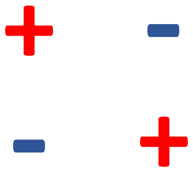
“Does zero training error often lead to overfitting?”
“More parameters might lead to overfitting.”

Three Angles

Understanding how Deep Models work

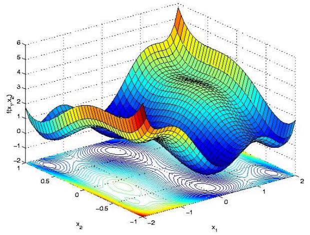


Expressibility



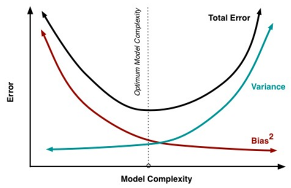
“Neural Network is a universal approximator”
“Deep Models can express functions more efficiently than shallow ones”

Optimization



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

Generalization

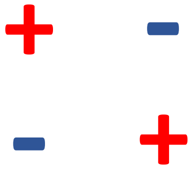


“Does zero training error often lead to overfitting?”
“More parameters might lead to overfitting.”

Which path should we take?

Three Angles – What to pick?

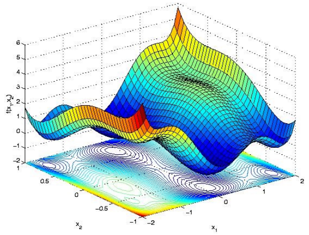
Expressibility



Architecture ✓

training dynamics ✗

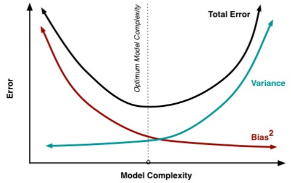
Optimization



Architecture ✗

training dynamics ✓

Generalization



Architecture ✗

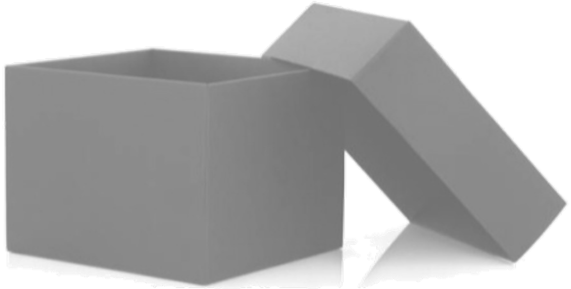
training dynamics ✗

How about

Architecture ✓

training dynamics ✓

Start From the First Principle



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

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

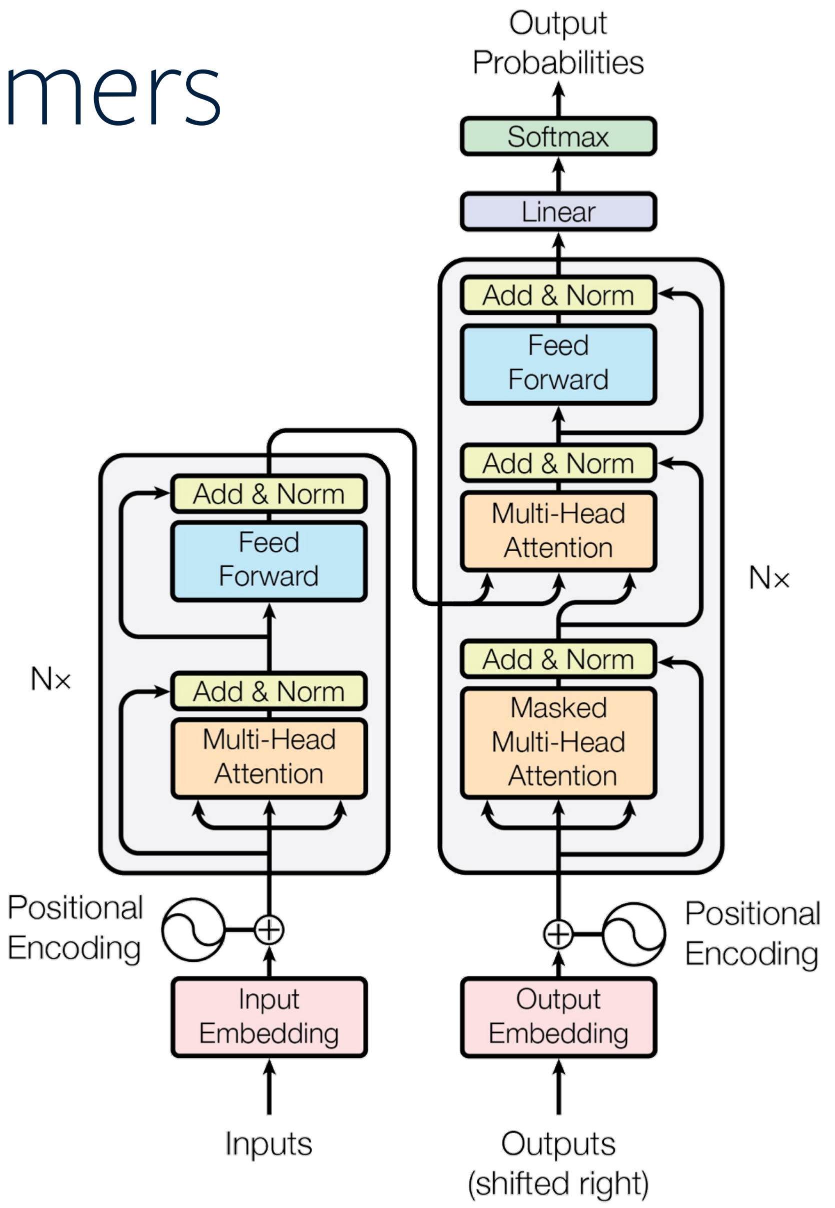
- **First principle** → Understand the behavior of the neural networks by checking the gradient **dynamics** induced by the neural **architectures**.

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

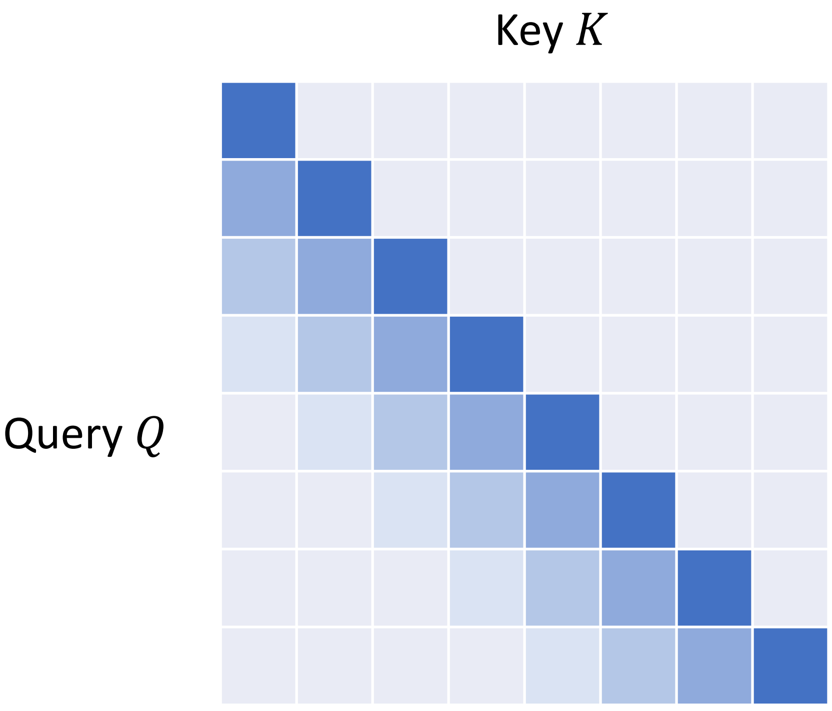
Architecture ✓

training dynamics ✓

Transformers



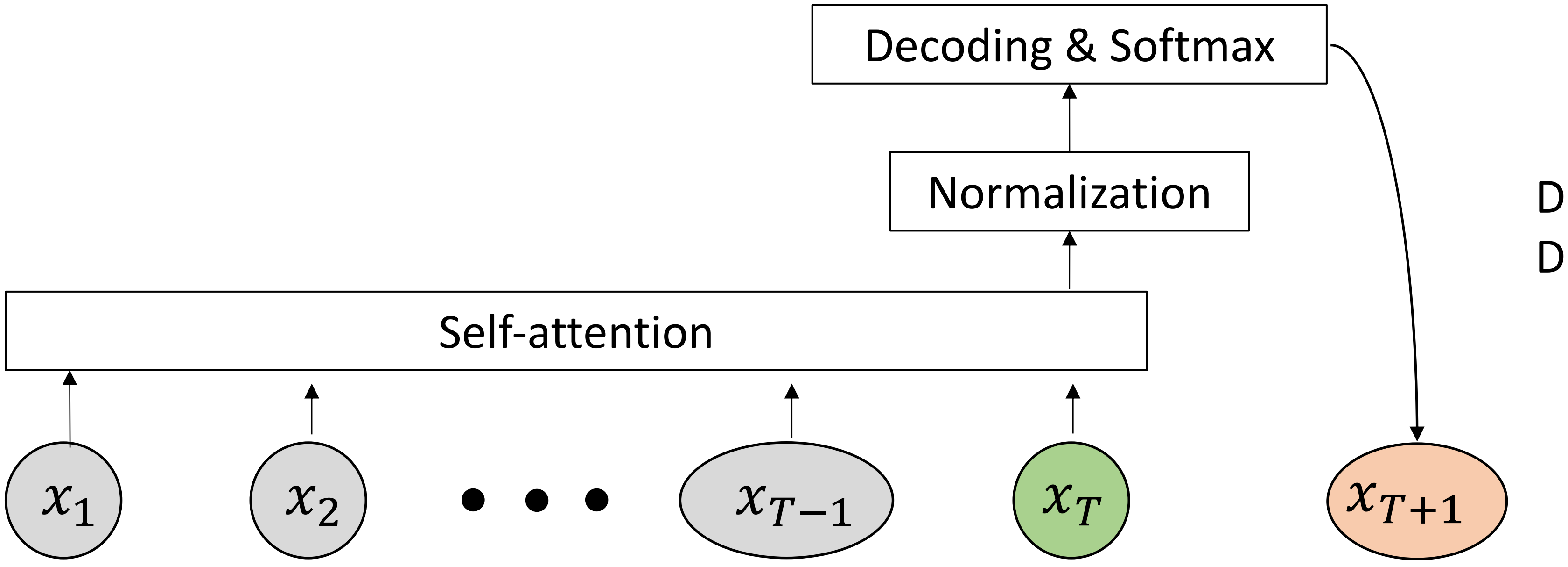
[A. Vaswani et al, Attention is all you need, NeurIPS'17]



Attention mechanism

facebook Artificial Intelligence

Understanding Attention in 1-layer Setting



Contextual tokens

Last/query token

Next token

Different next token =
Different classes in classification

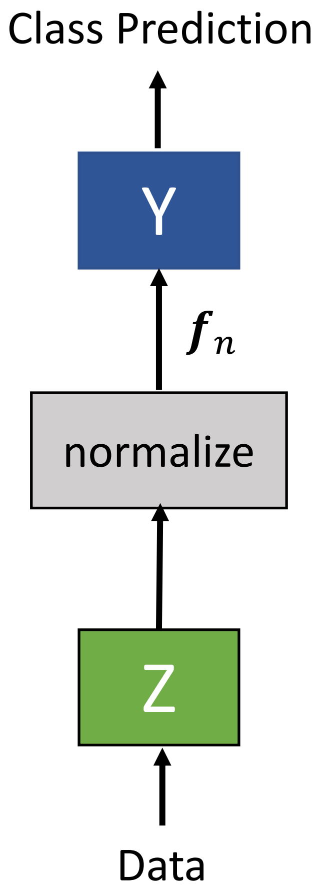
[*Y. Tian et al*, Scan and Snap: Understanding Training Dynamics and Token Composition in 1-layer Transformer, *NeurIPS'23]*

Reparameterization

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

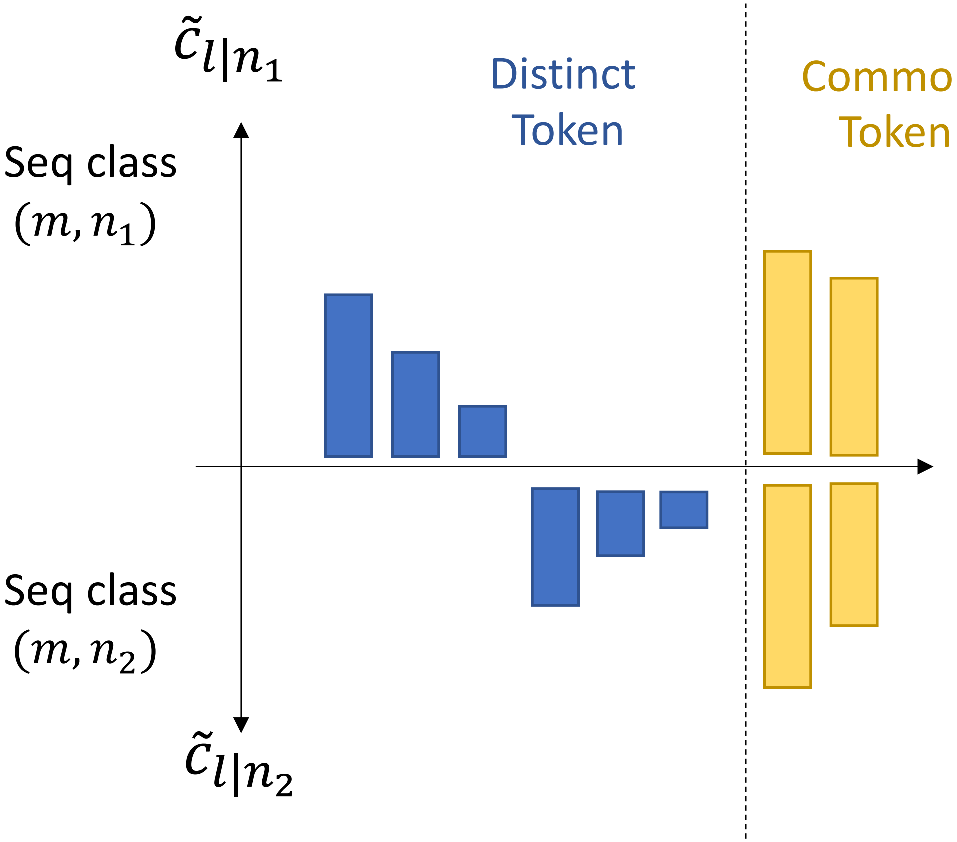
- Reparameterize the problem with independent variable Y and Z
  - $Y = UW_V^T U^T$ (*Merging the embedding with weight matrix*)
  - $Z = UW_Q W_K^T U^T$ (pairwise logits of self-attention matrix)

- Then the dynamics becomes easier to analyze



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

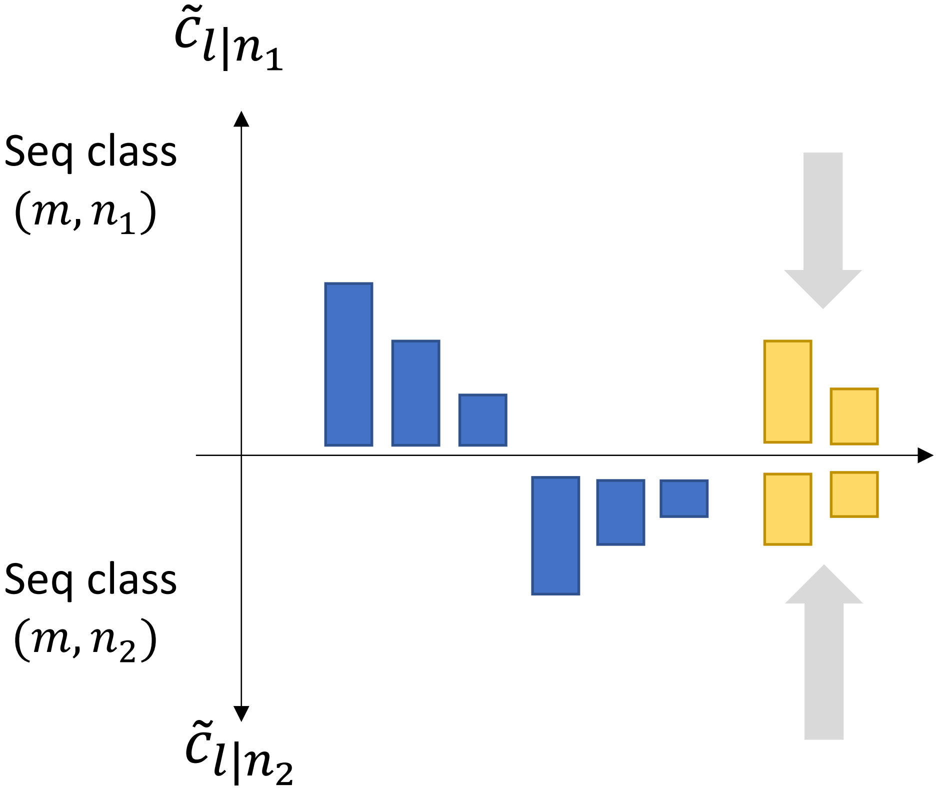
*Pairwise **attention score** between token l and query m*

Distinct tokens: Tokens that only appear in a single class.

Common tokens: Tokens that appear in multiple classes.

Overall Picture of the Training Dynamics

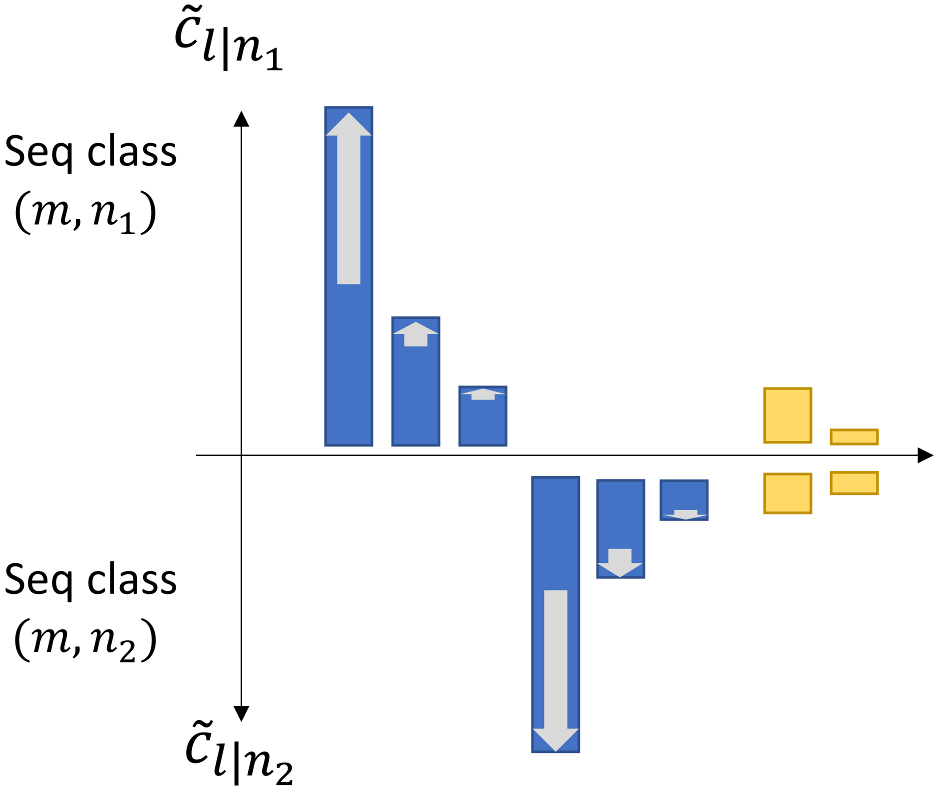
Common Token Suppression



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

Overall Picture of the Training Dynamics

Winners-emergence



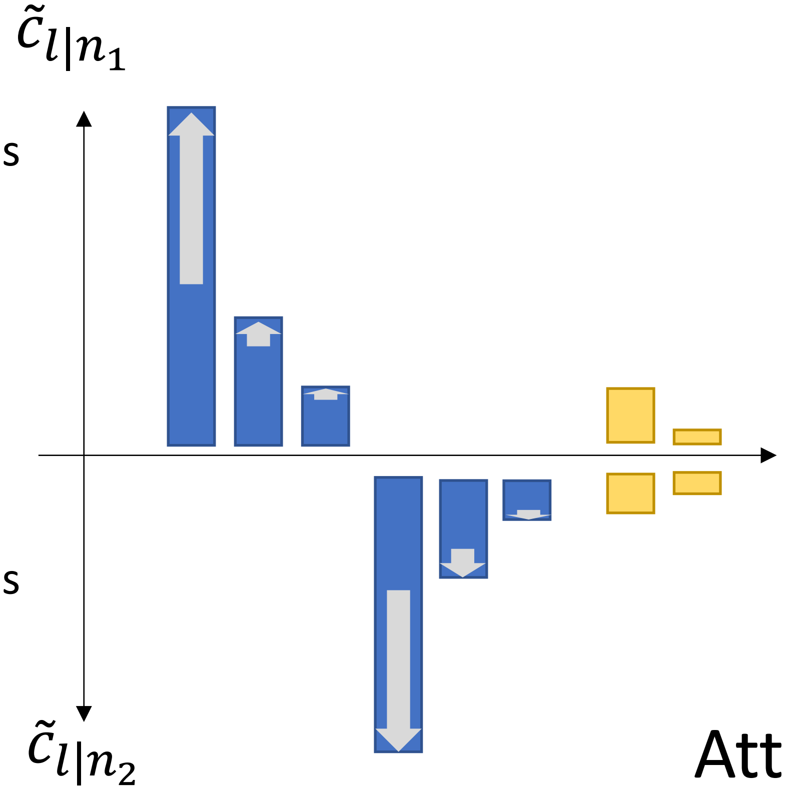
(a) $\dot{z}_{ml} < 0$, for common token l

(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

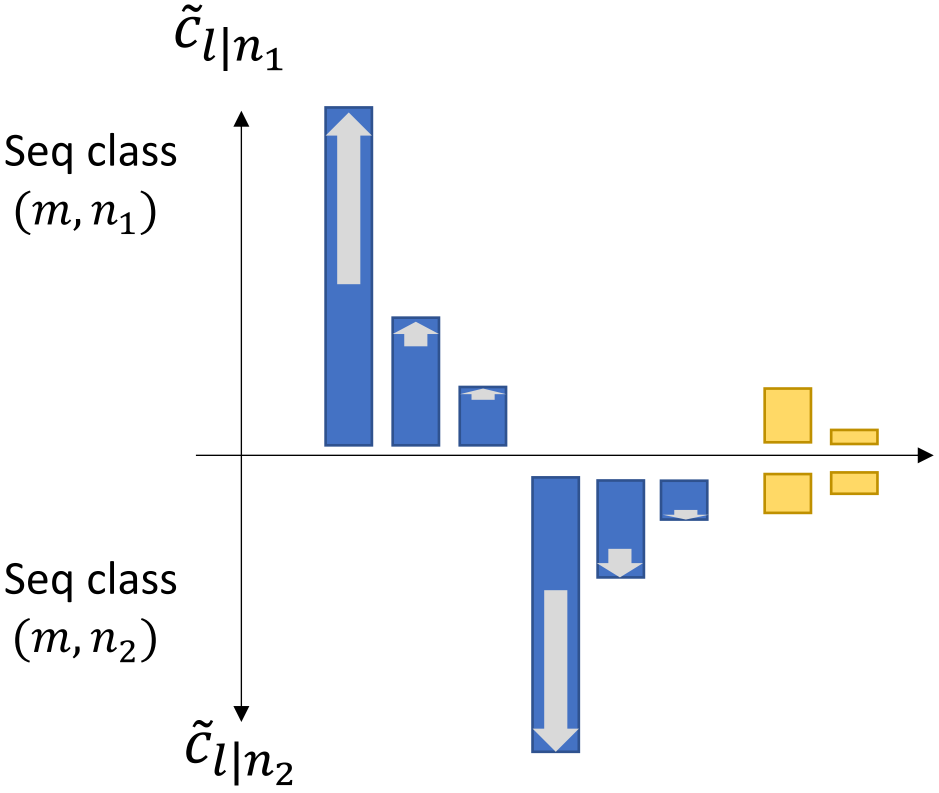
(b) $\dot{z}_{ml} > 0$, for distinct token l

(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:

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

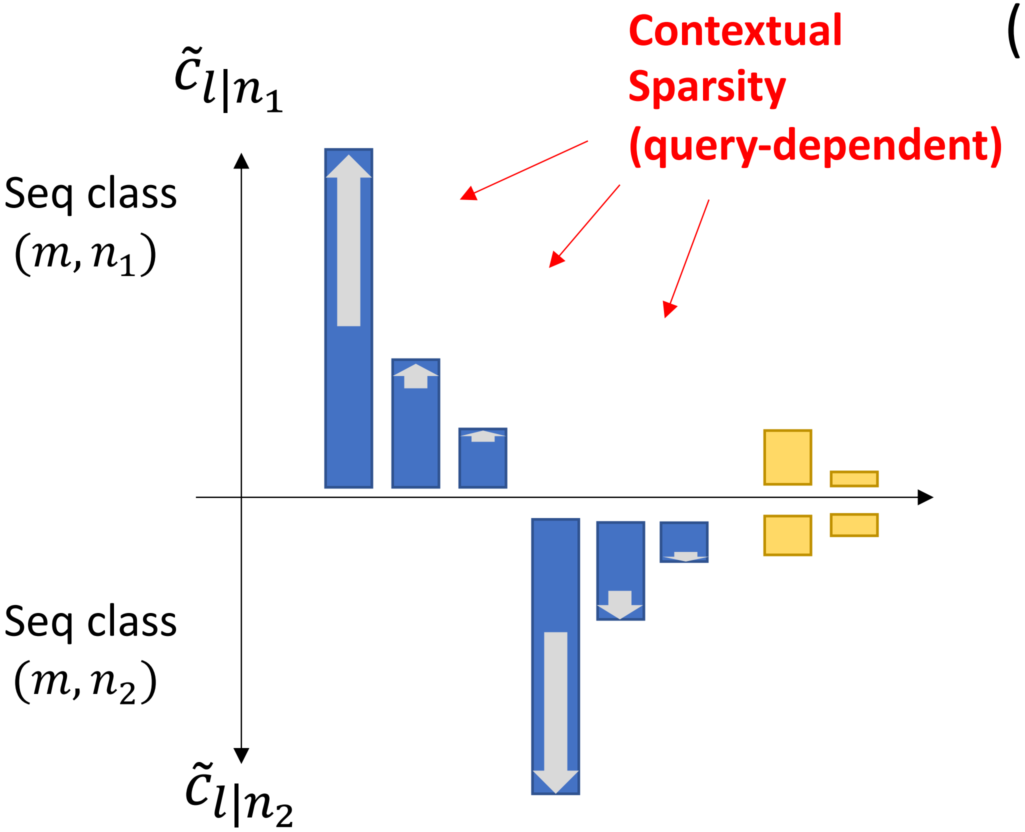
If l_0 is the dominant token: $r_{l_0/l|n}(0) > 0$ for all $l \neq l_0$ then

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

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:

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

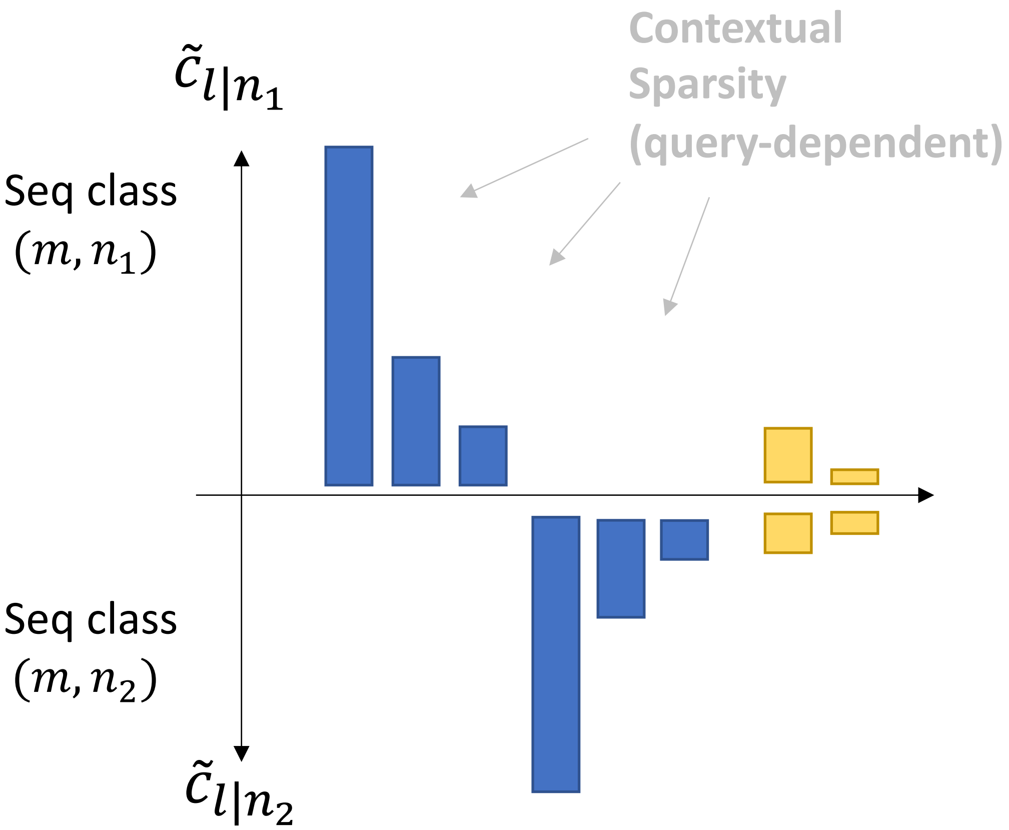
If l_0 is the dominant token: $r_{l_0/l|n}(0) > 0$ for all $l \neq l_0$ then

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

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**:

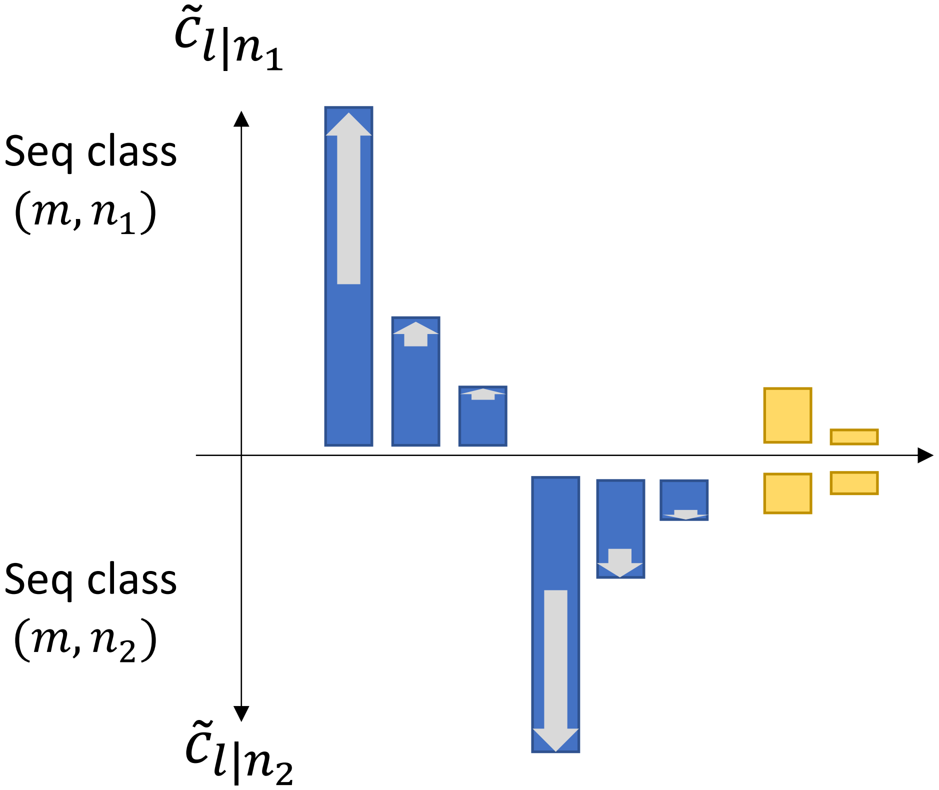
When $t \geq t_0 = O\left(\frac{2K \ln M}{\eta_Y}\right)$, $B_n(t) = O(\ln \ln t)$

(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

Winners-emergence

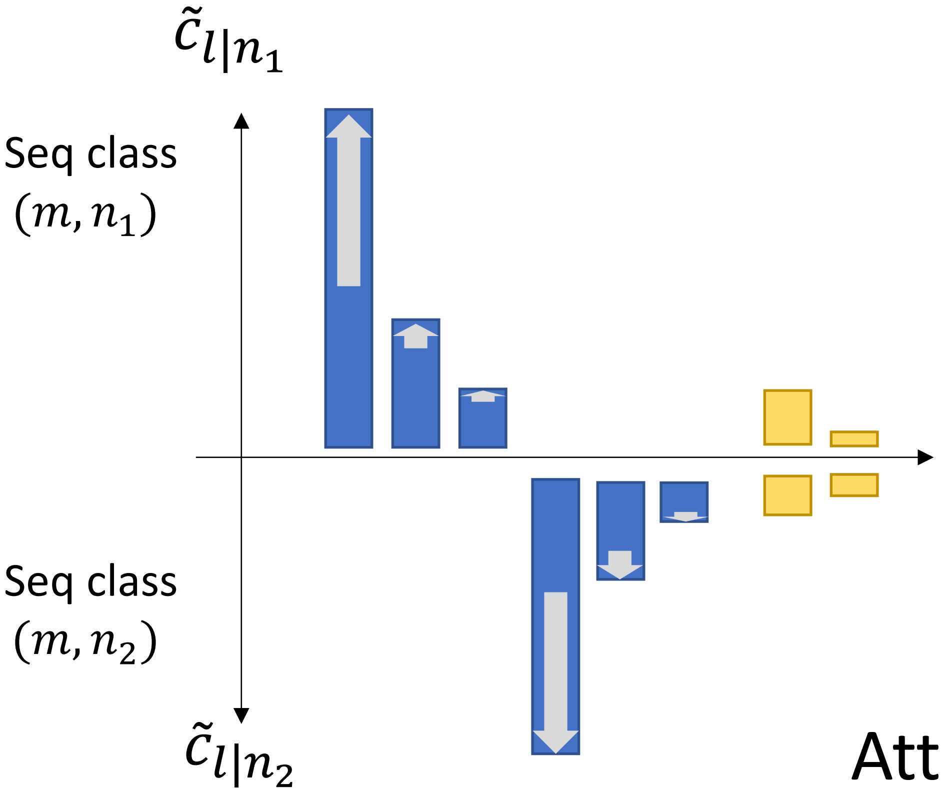


(a) $\dot{z_{ml}} < 0$, for common token l

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

Overall Picture of the Training Dynamics

Winners-emergence



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

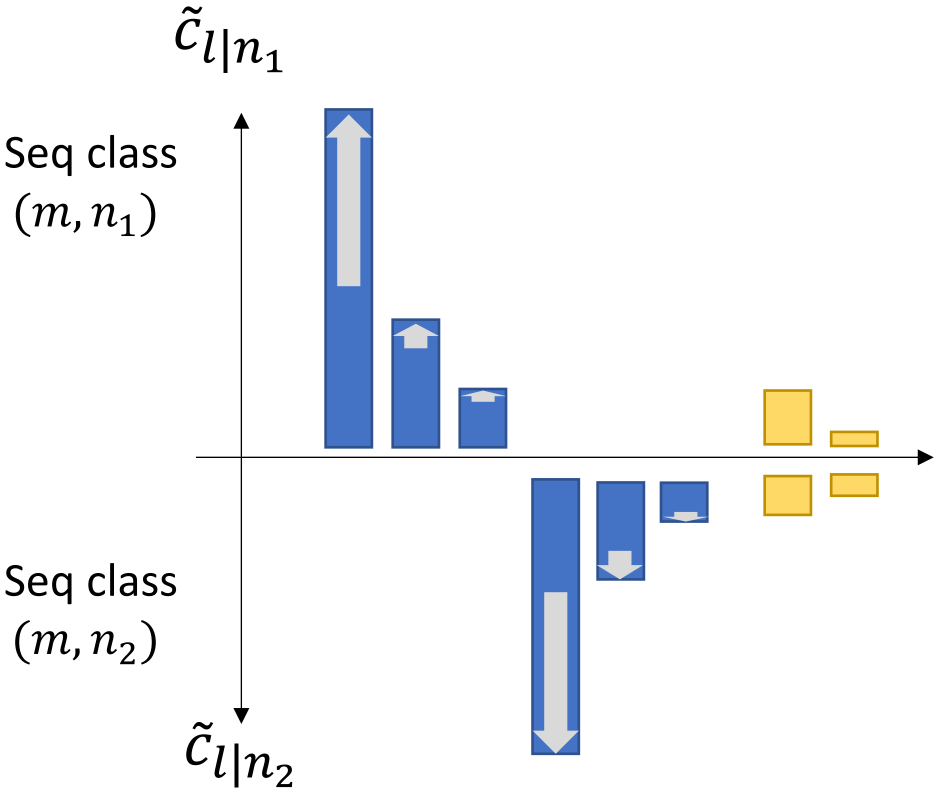
(b) $\dot{z}_{ml} > 0$, for distinct token l

(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:

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

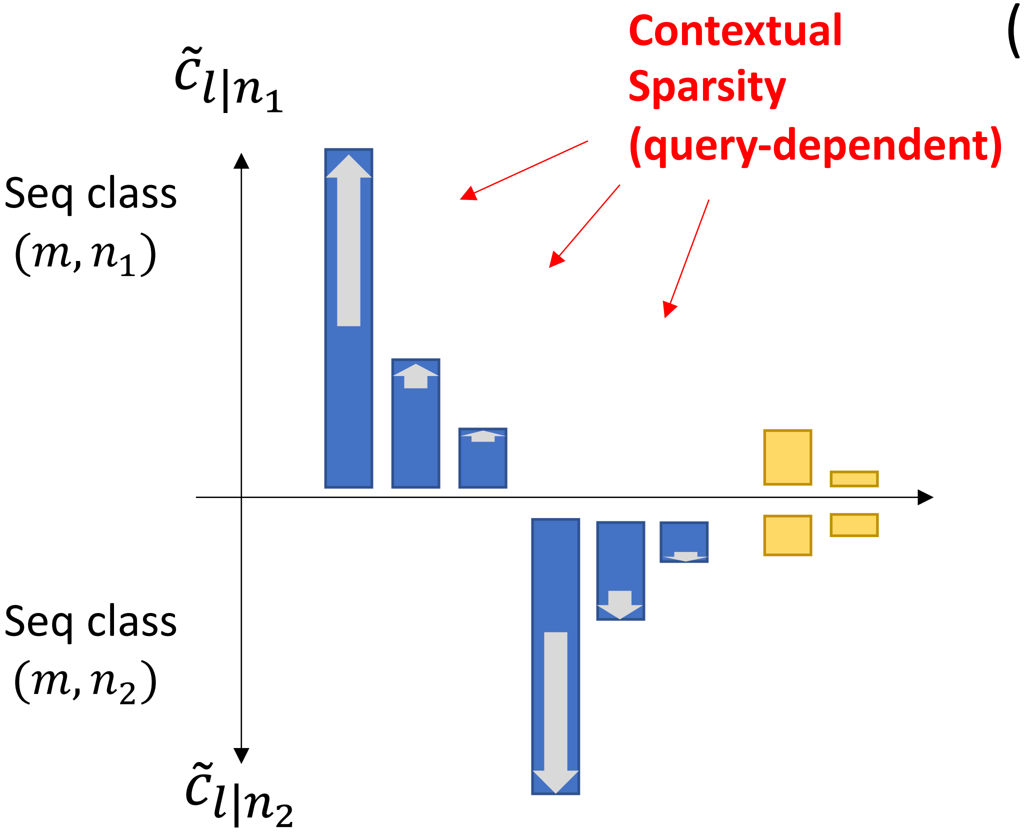
If l_0 is the dominant token: $r_{l_0/l|n}(0) > 0$ for all $l \neq l_0$ then

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

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

Overall Picture of the Training Dynamics

Winners-emergence



Contextual Sparsity (query-dependent)

(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:

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

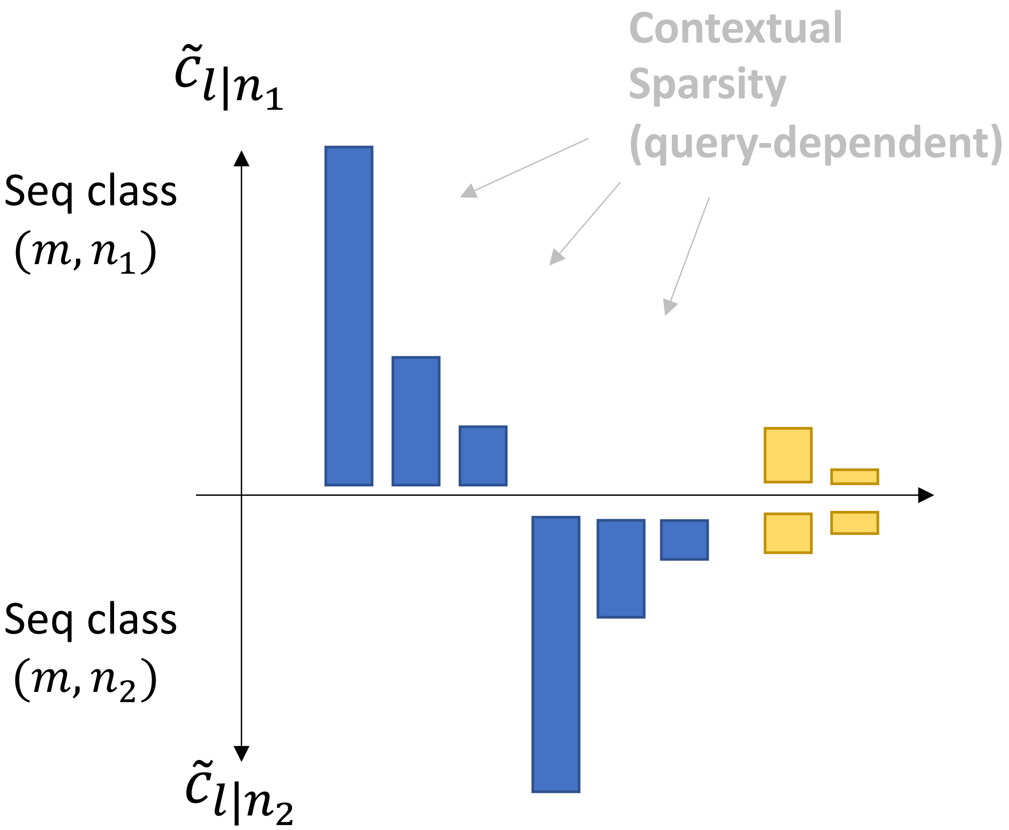
If l_0 is the dominant token: $r_{l_0/l|n}(0) > 0$ for all $l \neq l_0$ then

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

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

Simple Real-world Experiments

WikiText2
(original parameterization)

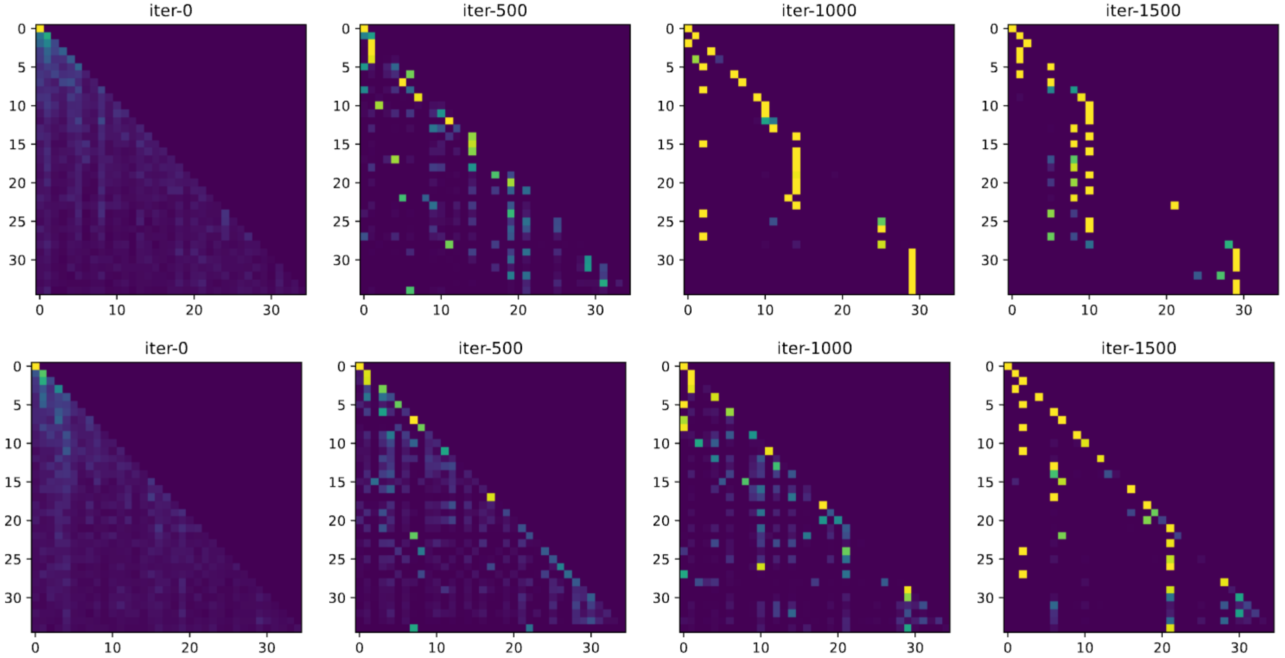


Figure 7: Attention patterns in the lowest self-attention layer for 1-layer (top) and 3-layer (bottom) Transformer trained on WikiText2 using SGD (learning rate is 5). Attention becomes sparse over training.

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]

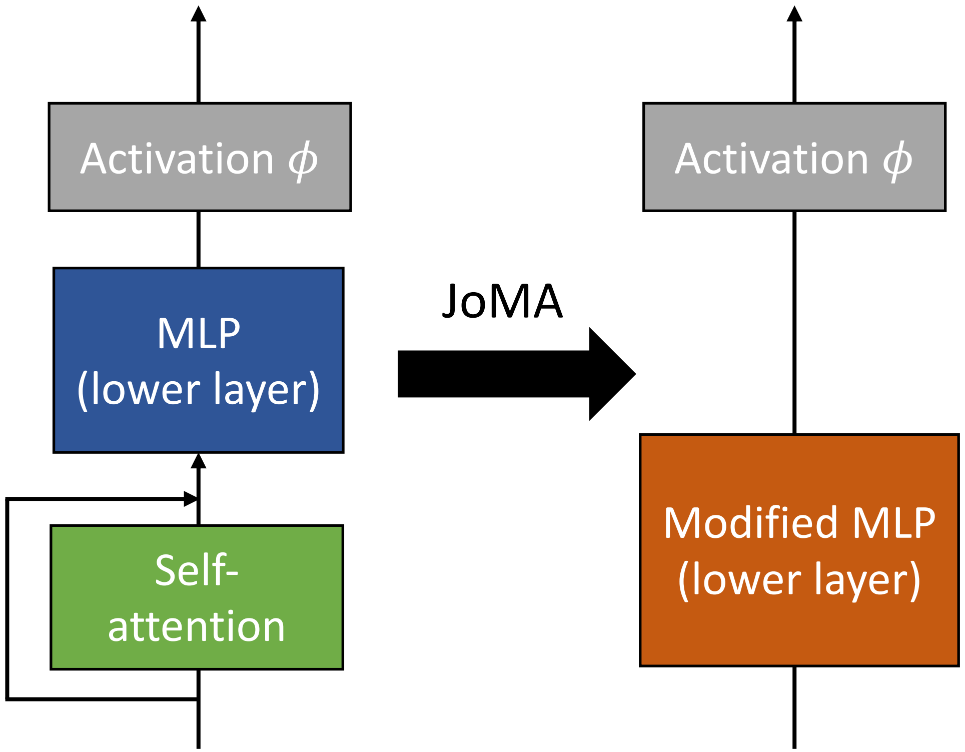
[G. Xiao et al, *Efficient Streaming Language Models with Attention Sinks*, ICLR’24]

Follow-up works

- Scan & Snap has Multiple 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

- How to get rid of them?
- Follow-up work: **JoMA**

JoMA: JOint Dynamics of MLP/Attention layers

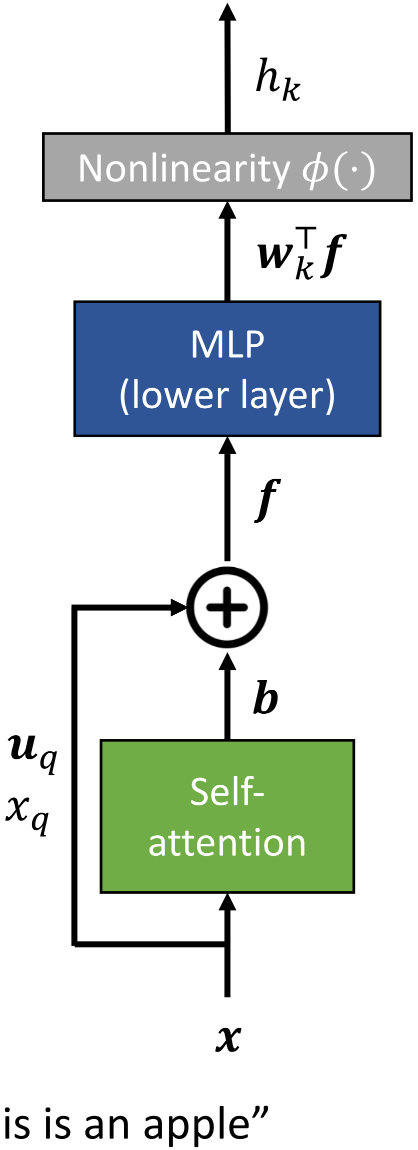


Main Contributions:

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



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

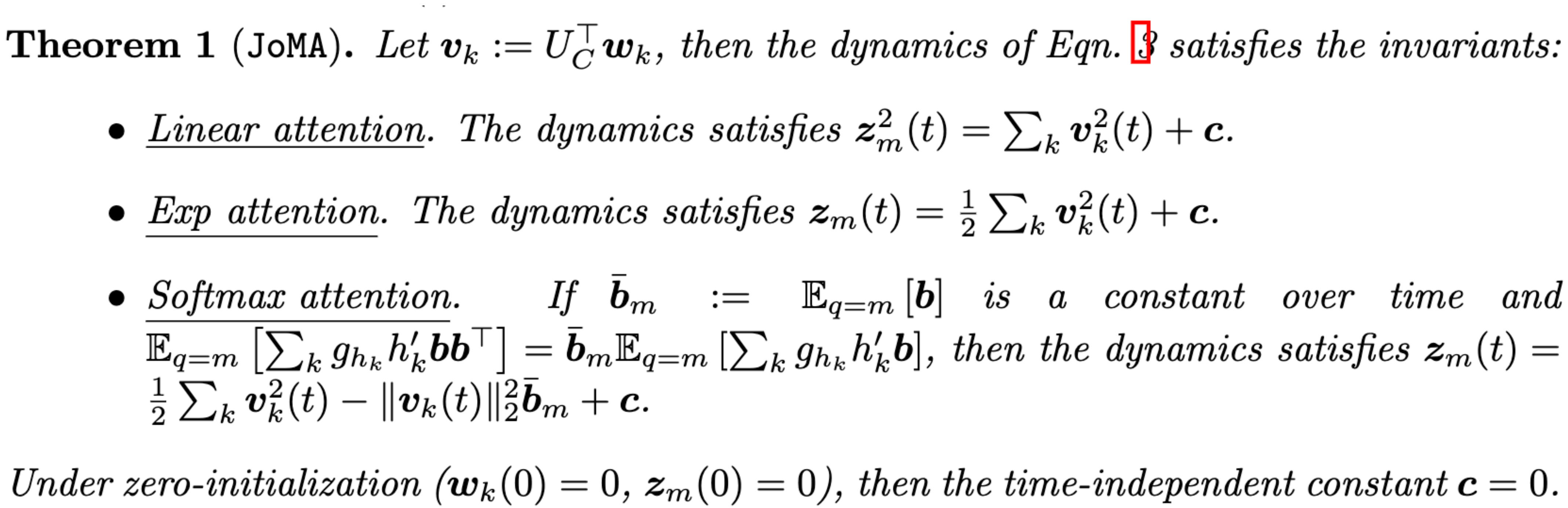
$$\boldsymbol{f} = U_C \boldsymbol{b} + \boldsymbol{u}_q$$

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: $b_l = x_l z_{ql}$

JoMA Dynamics



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

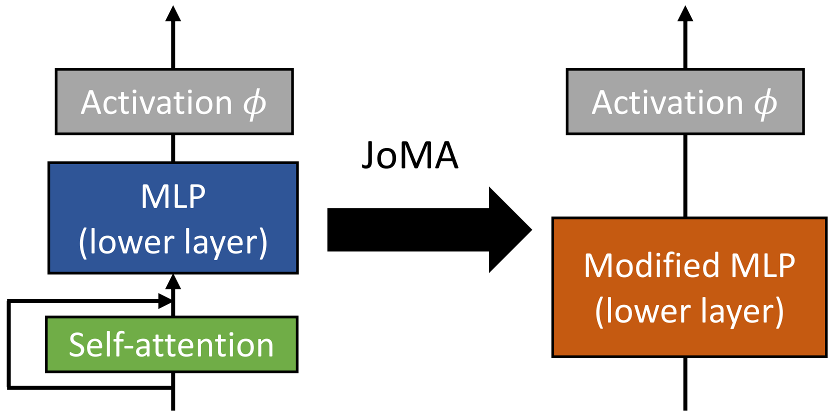
- *Linear attention. The dynamics satisfies* $\boldsymbol{z}_m^2(t) = \sum_k \boldsymbol{v}_k^2(t) + \boldsymbol{c}$.
- *Exp attention. The dynamics satisfies* $\boldsymbol{z}_m(t) = \frac{1}{2}\sum_k \boldsymbol{v}_k^2(t) + \boldsymbol{c}$.
- *Softmax attention. If* $\bar{\boldsymbol{b}}_m := \mathbb{E}_{q=m}[\boldsymbol{b}]$ *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* $\boldsymbol{c} = 0$.

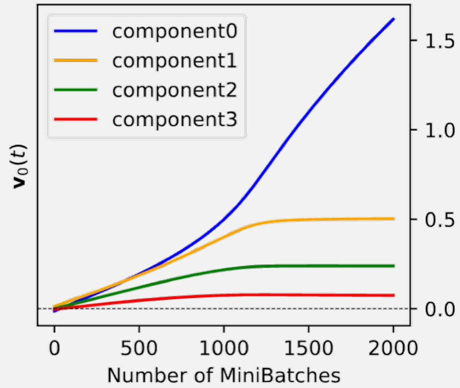
There is residual connection.
Joint dynamics works for any learning rates between self-attention and MLP layer.
No assumption on the data distribution.

Implication of Theorem 1

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

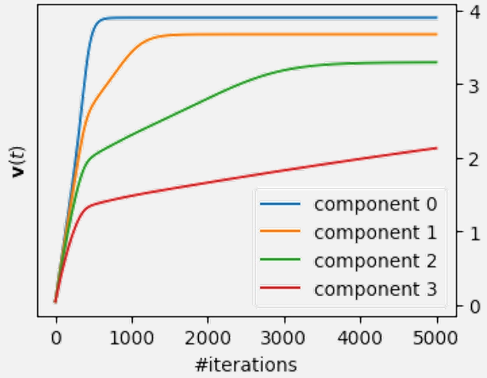


Linear case ($\phi = \text{Id}, K = 1$)



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 $\Delta_{lm} = \underset{\substack{\uparrow \\ \textbf{Discriminancy}}}{\mathbb{E}[g|l,m]} \cdot \underset{\substack{\uparrow \\ \textbf{CoOccurrence}}}{\mathbb{P}[l|m]}$

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

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

JoMA for Linear Activation

Linear

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

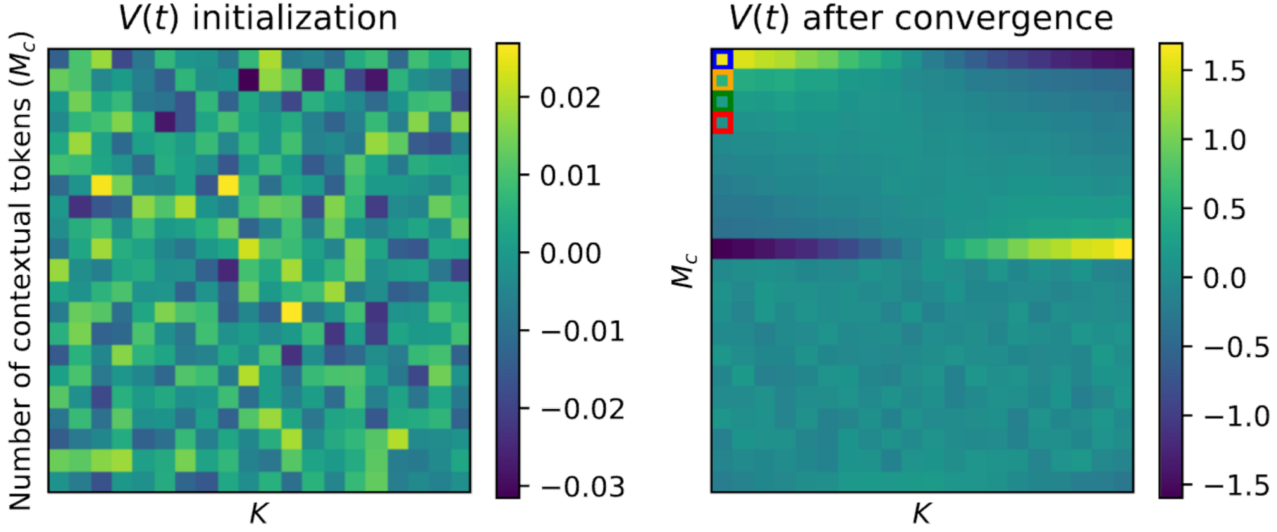
Modified MLP (lower layer)

Theorem 2

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

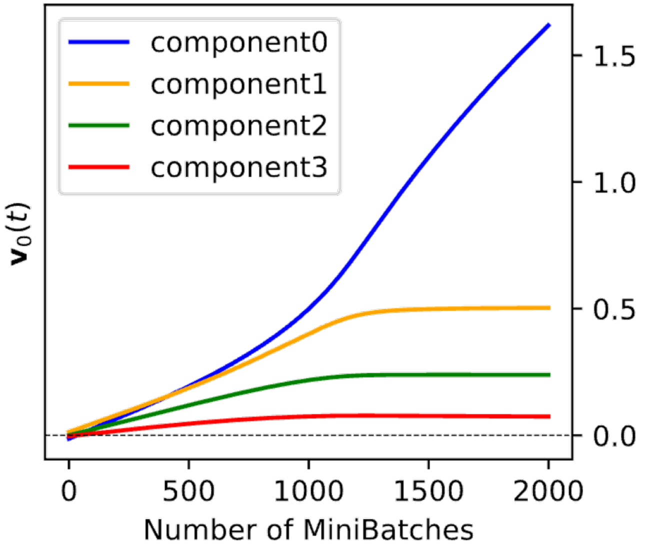
$$\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}\mathrm{d}t \in [-1,1]$$

Only 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 ($K > 1$)?

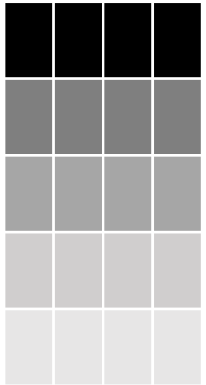
- $V = U_C^\top W \in \mathbb{R}^{M_c \times K}$ and the dynamics becomes

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

We can prove that V gradually becomes low rank

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

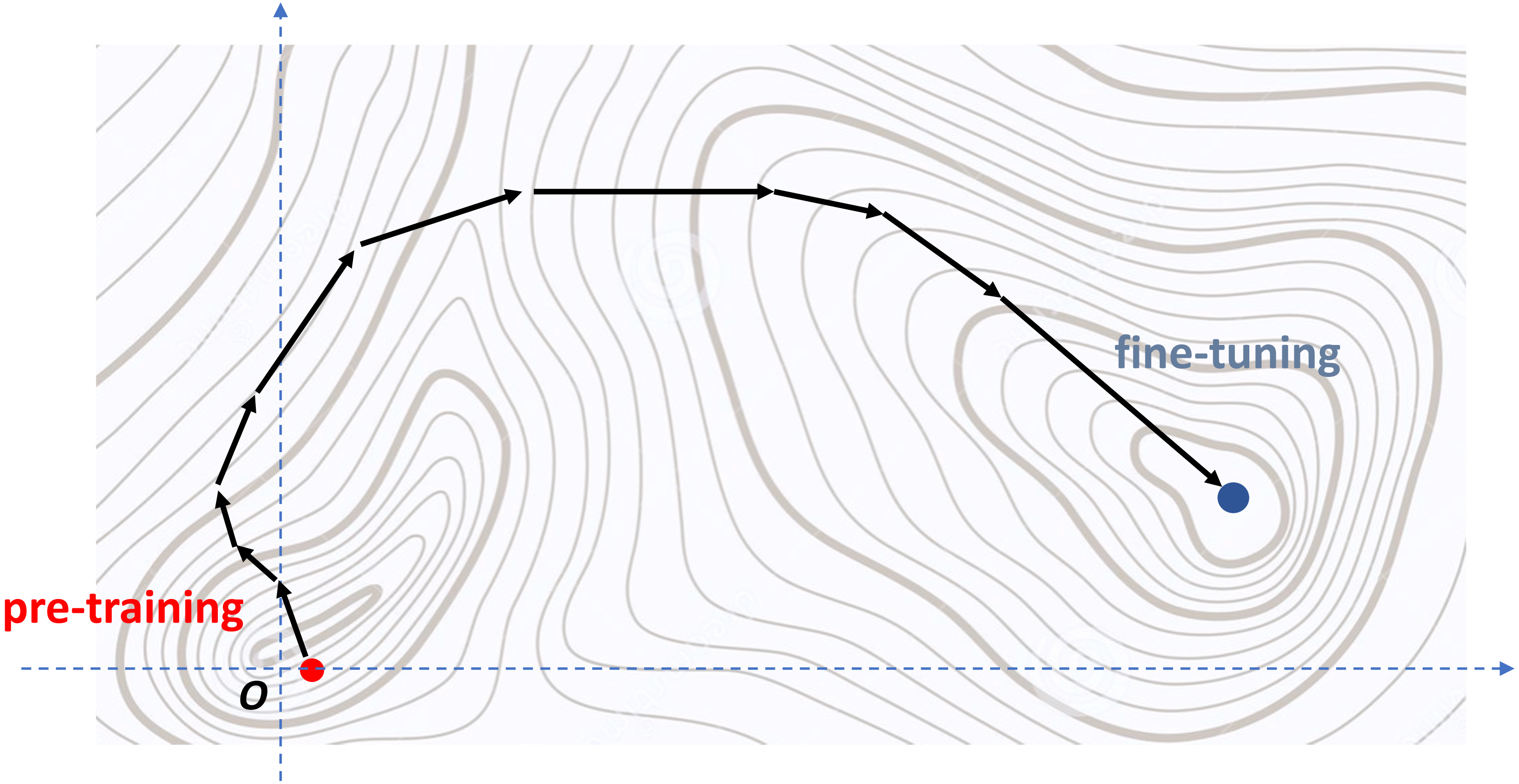
$V(t) \rightarrow$



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

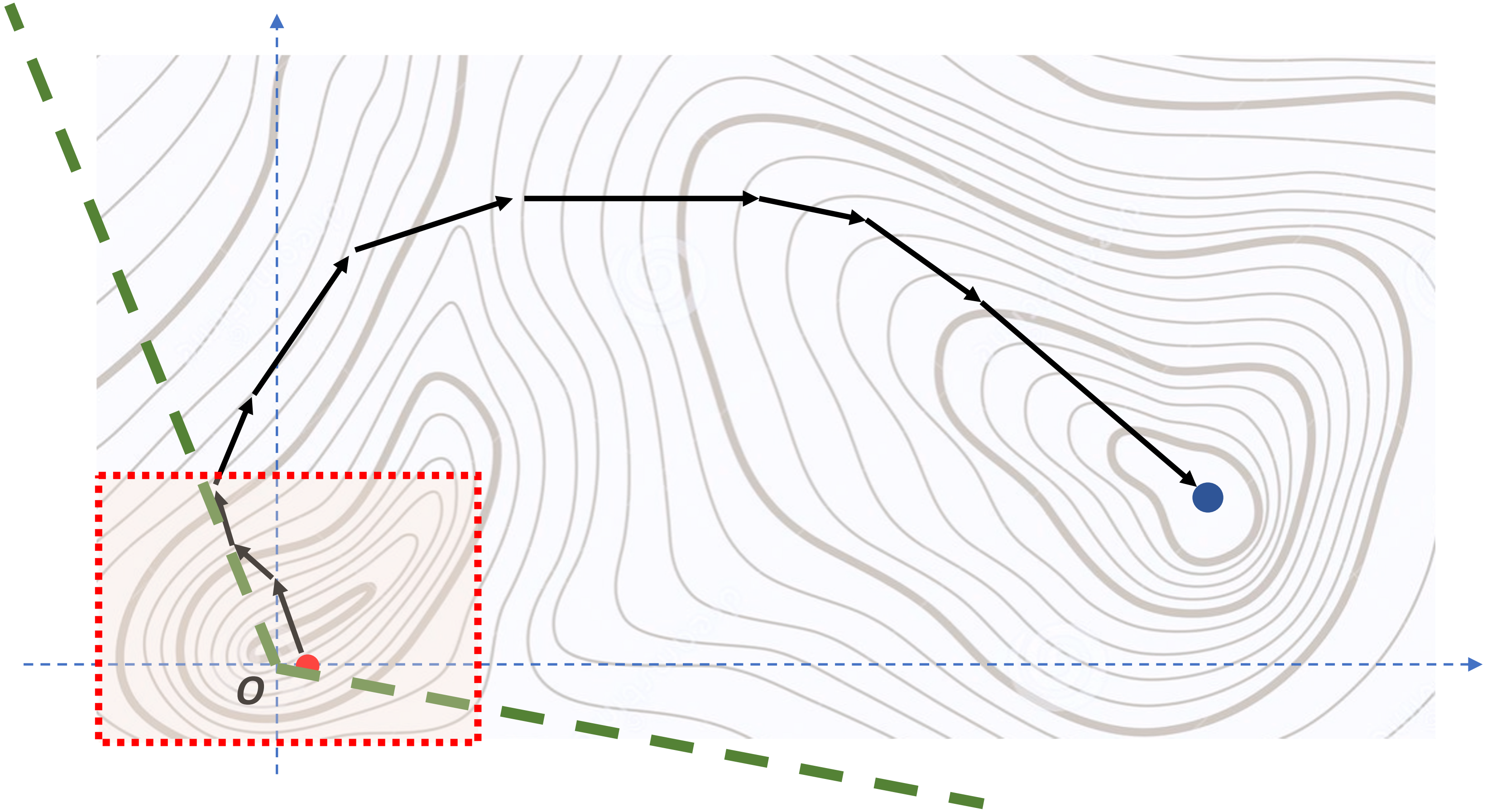
How the Weight Rank Changes over time?

Consider the Entire Training Trajectory …



How the Weight Rank Changes over time?

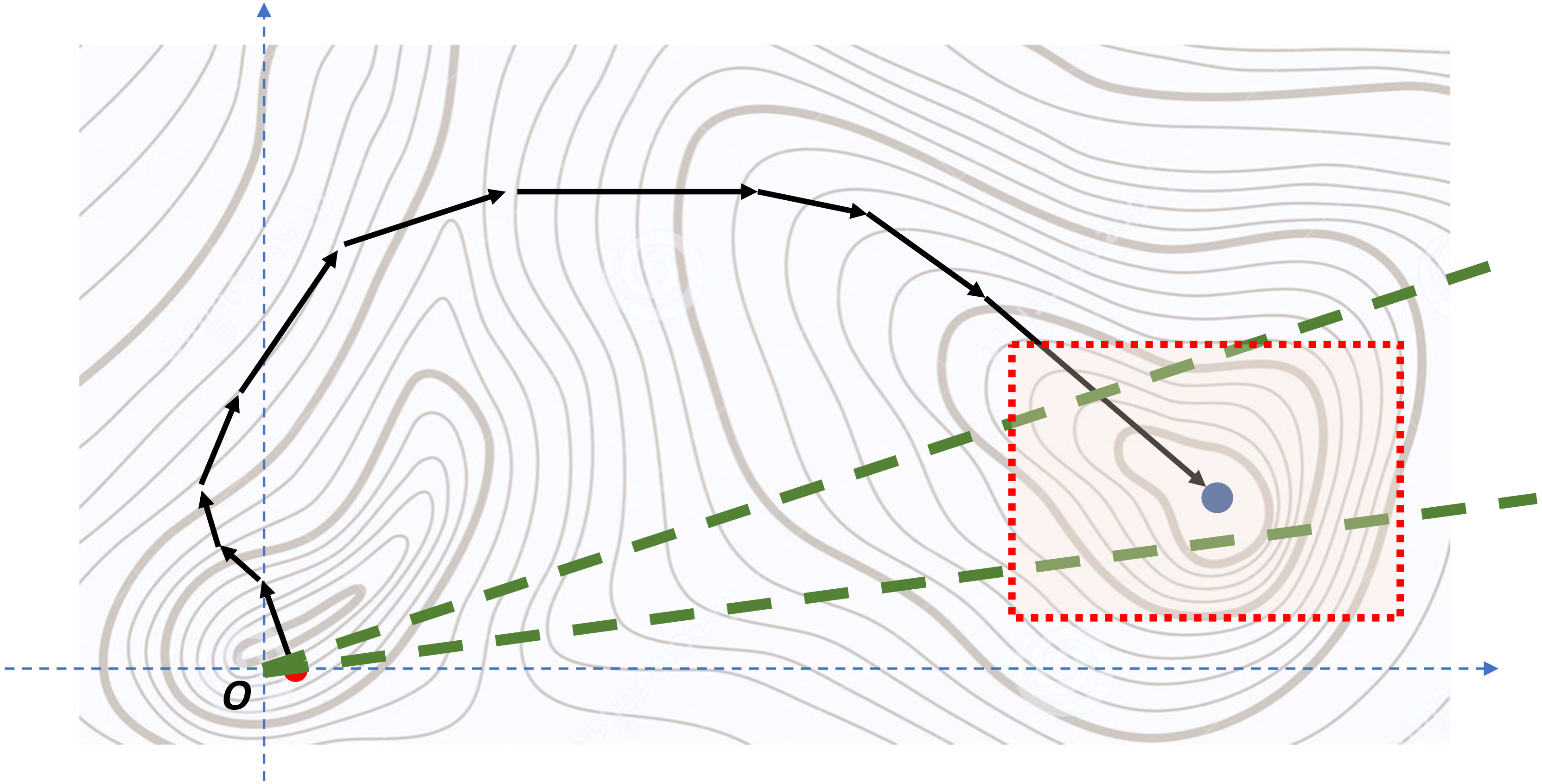
Beginning of Training: Weight subspace changes **a lot**



W high rank (due to random initialization)

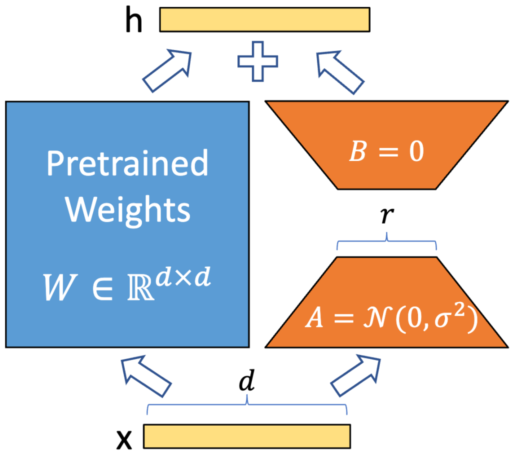
How the Weight Rank Changes over time?

Mid/End of Training: **Weight subspace** changes **little**



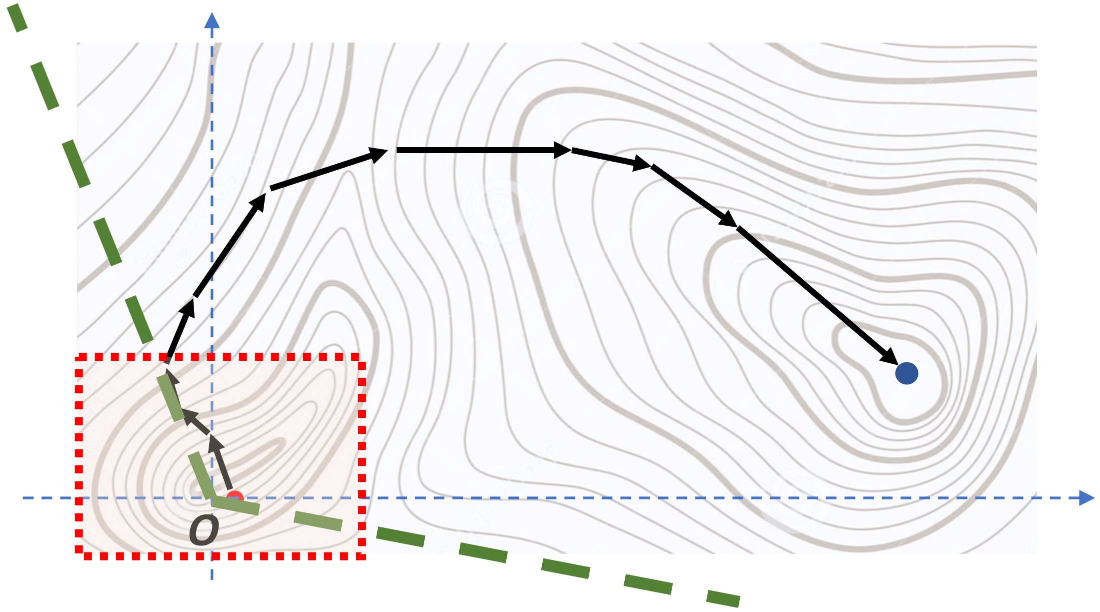
W low rank

Think about LoRA?


h
Pretrained Weights
$W \in \mathbb{R}^{d \times d}$
$B = 0$
r
$A = \mathcal{N}(0, \sigma^2)$
d
x

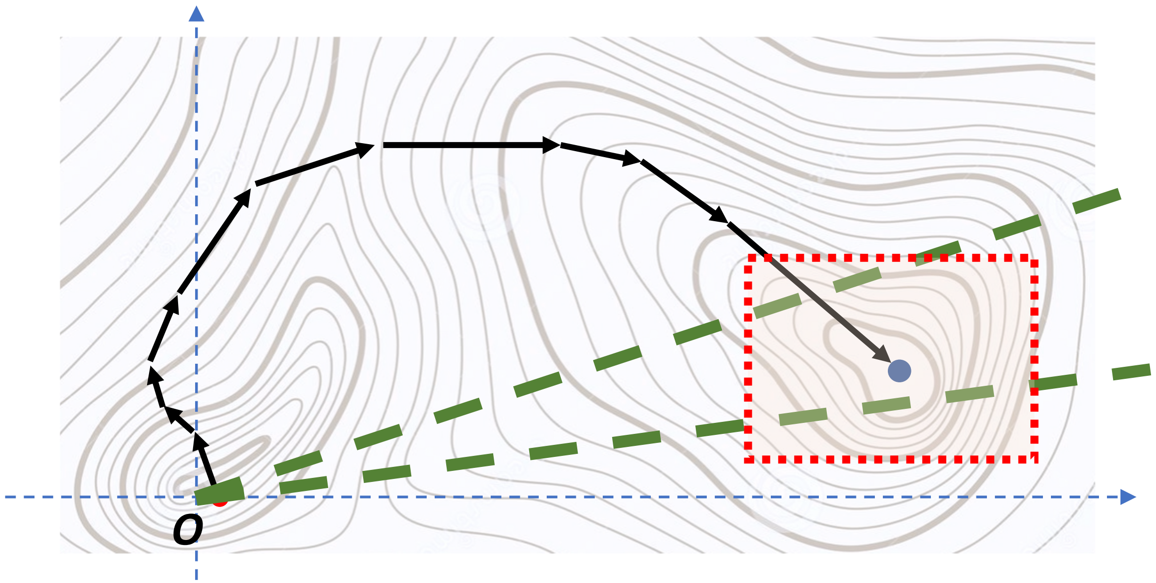
LoRA (Low-rank Adaption)

W high rank (due to random initialization)



LoRA does not work

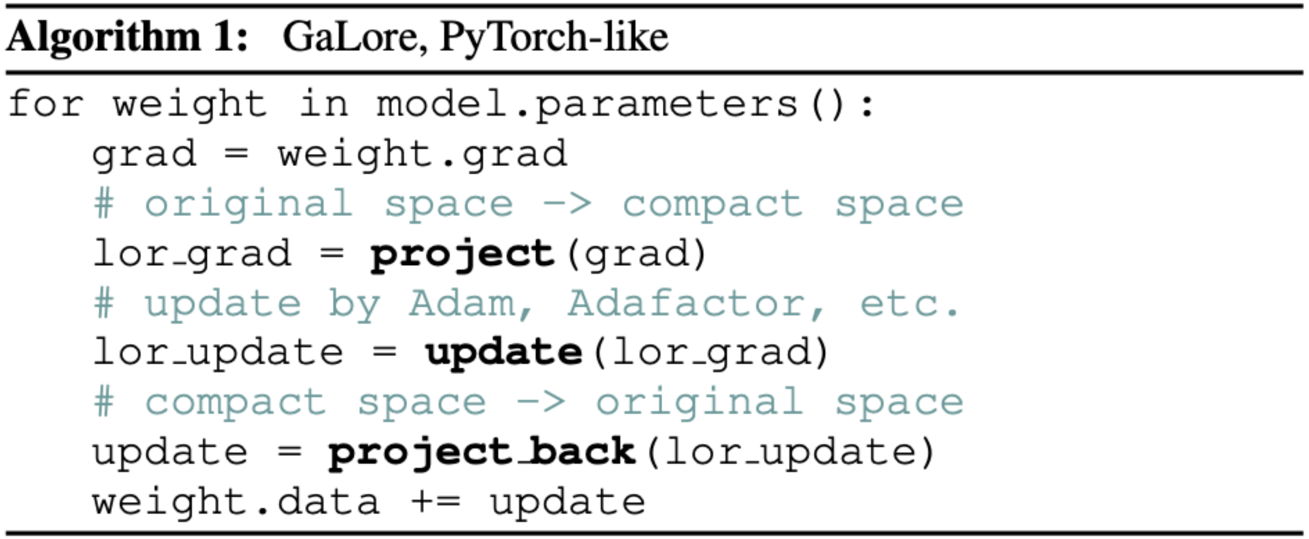
W low rank



LoRA can work

GaLore

low-rank weights → low-rank gradients



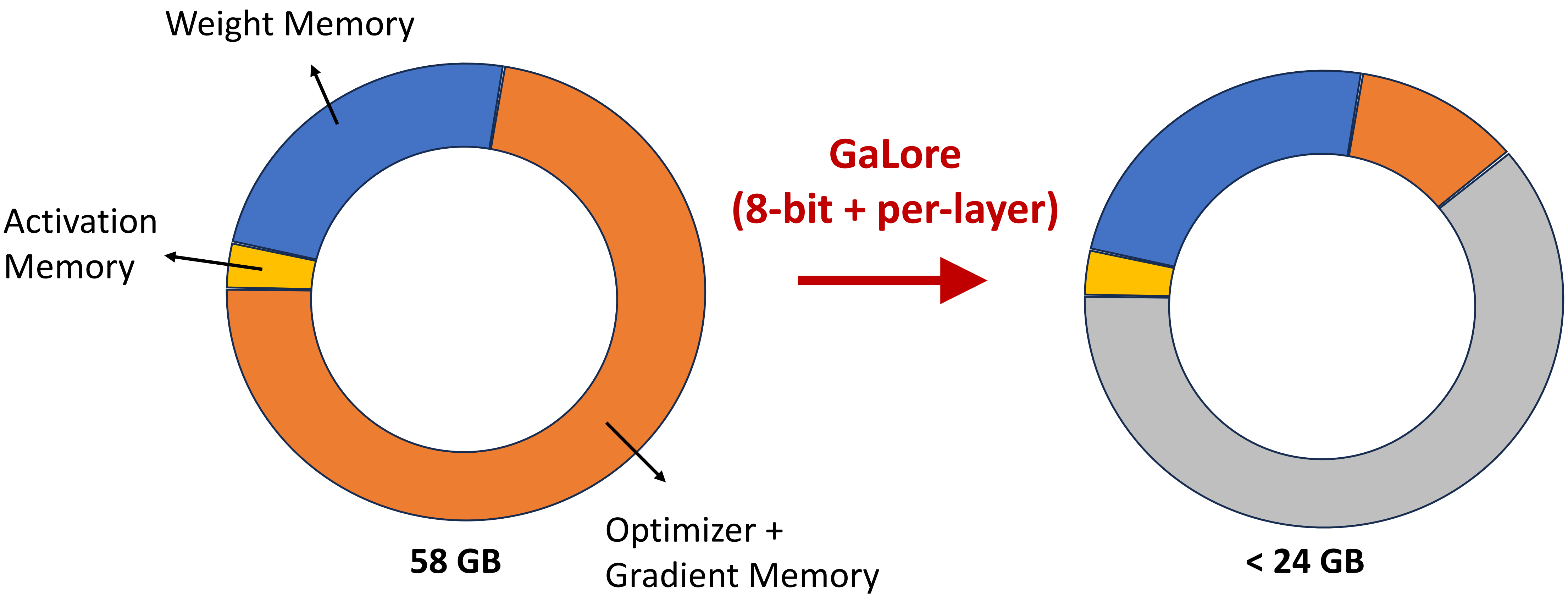
Algorithm 1: GaLore, PyTorch-like

```
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
```

```
$G_t \leftarrow -\nabla_W \phi(W_t)$
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}
$W_{t+1} \leftarrow W_t + \eta \tilde{G}_t$
```

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

Memory Saving in GaLore



Weight Memory

Activation Memory

Optimizer + Gradient Memory

58 GB

GaLore (8-bit + per-layer)

< 24 GB

Reduce optimizer states and weight gradients, Achieve **82.5%** mem reduction

Convergence Analysis on Fixed Projection

For gradient in the following form

$$G = \sum_i A_i - \sum_i B_i W C_i$$

Let $R = P^\top G Q$ be projected gradient (P and Q are fixed) then

$$\boldsymbol{\|R_t\|_F \leq (1 - \eta M)\|R_{t-1}\|_F \to 0}$$

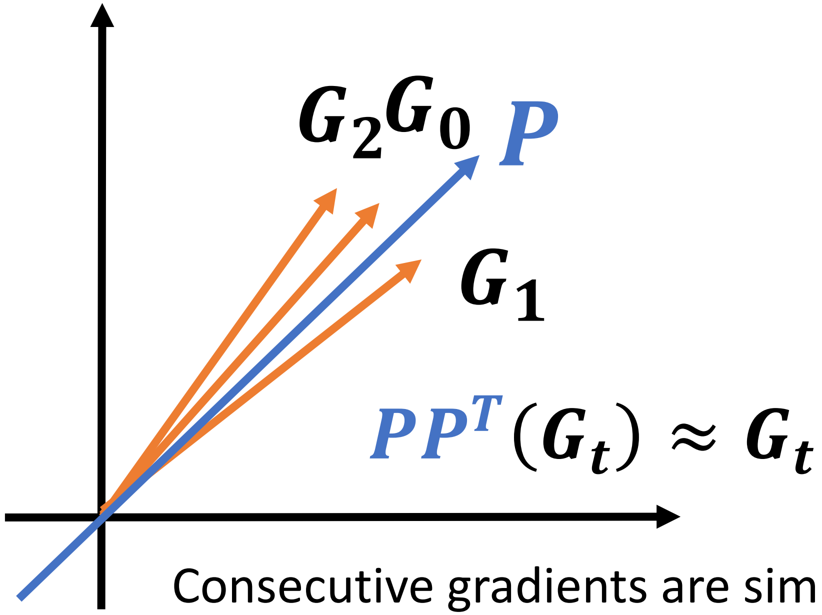
Where $M := \frac{1}{N}\sum_i \min_t \lambda_{\min}(\hat{B}_{it})\lambda_{\min}(\hat{C}_{it}) - L_A - L_B L_C D^2$

$$\hat{B}_{it} = P_t^T B_i(W_t) P_t \qquad \hat{C}_{it} = Q_t^T C_i(W_t) Q_t$$

Does that mean it works? No… $R_t \to 0$ just means the gradient within the subspace vanishes.

How to continue optimization? Change the projection from time to time!

How often to change P_t?



Consecutive gradients are similar

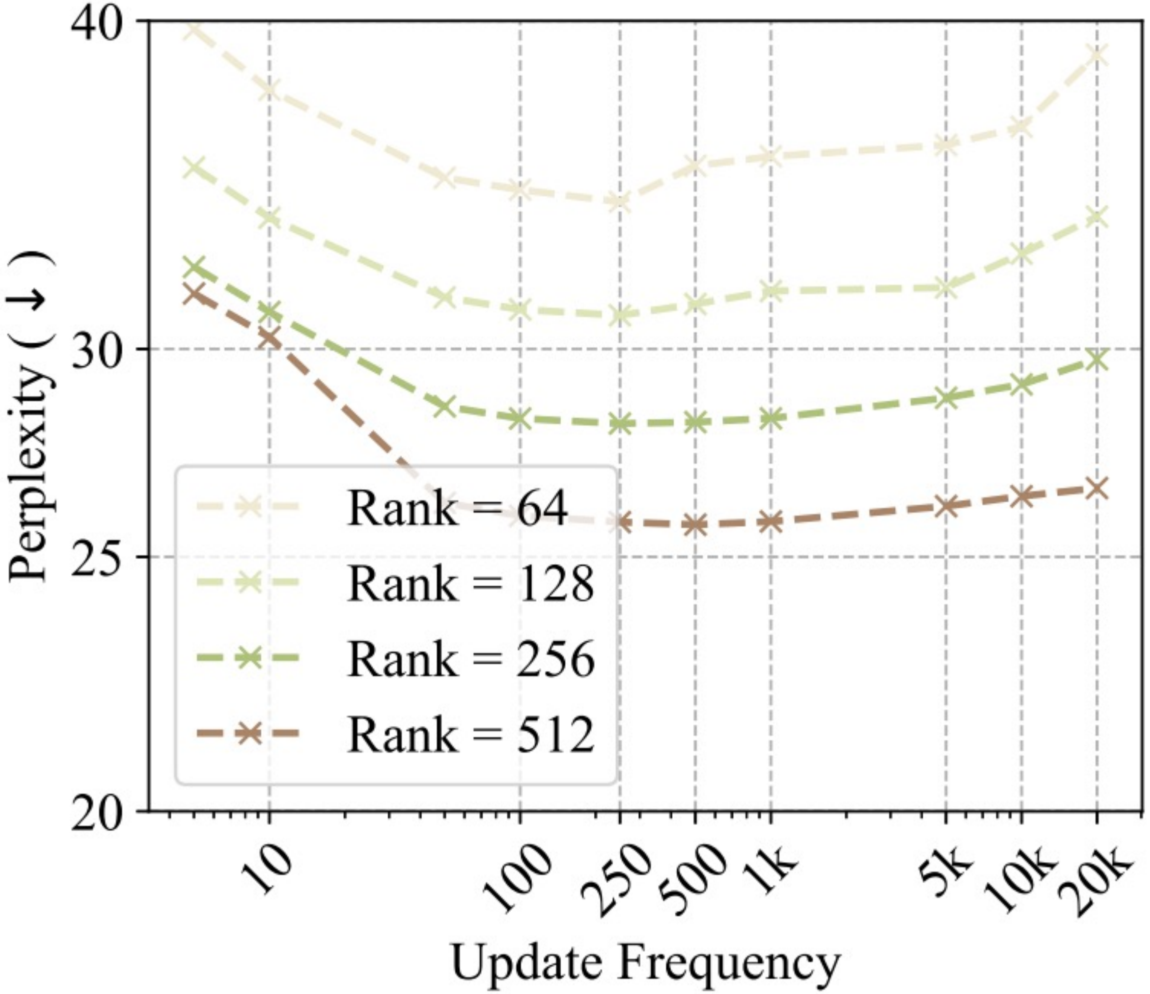
For every T iterations:

Compute and store P_t= SVD(G_t)

P_t is the projection matrix.

No need to change P_t every iteration!

Training 130M models ($d_{\text{model}} = 768$)



Pre-training Results (LLaMA 7B) on C4

7B model trained on up to 150K steps and 19.7 B tokens

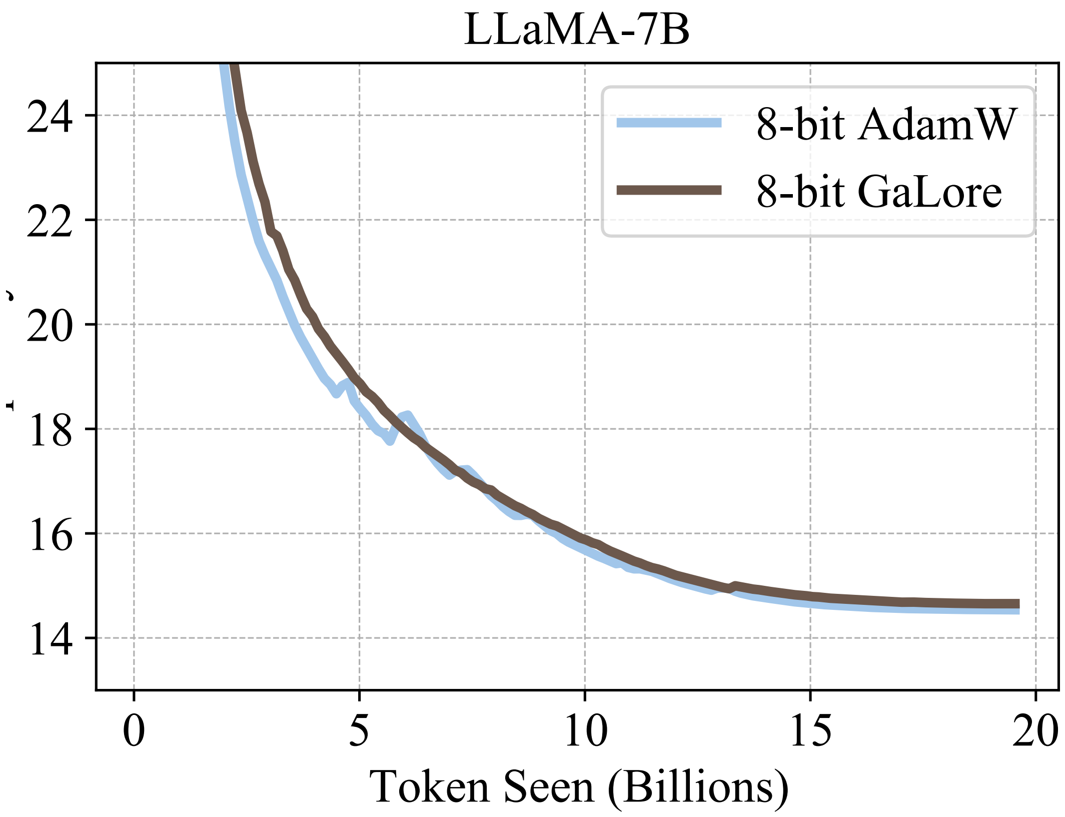
	Mem	40K	80K	120K	150K
8-bit GaLore	18G	17.94	15.39	14.95	14.65
8-bit Adam	26G	18.09	15.47	14.83	14.61
Tokens (B)		5.2	10.5	15.7	19.7

C4 Dataset ⊕

LLaMA-7B ⊕

single RTX 4090

Pre-training - for the first time!



facebook Artificial Intelligence

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)

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:

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

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

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)

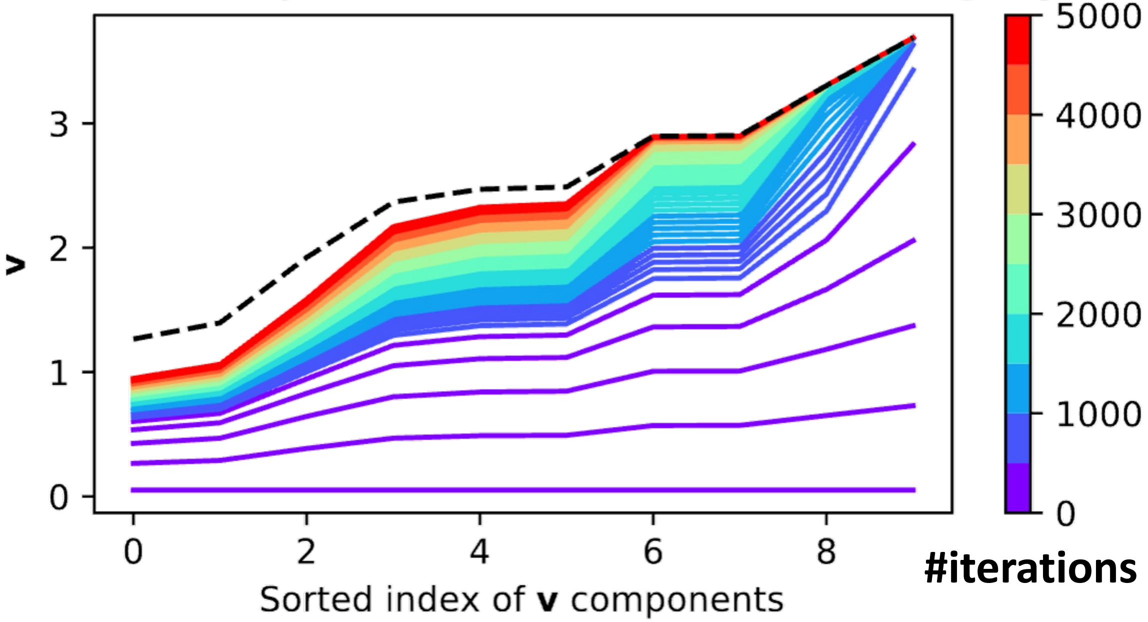
Theorem 4

Salient components grow much faster than non-salient ones:

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

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

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



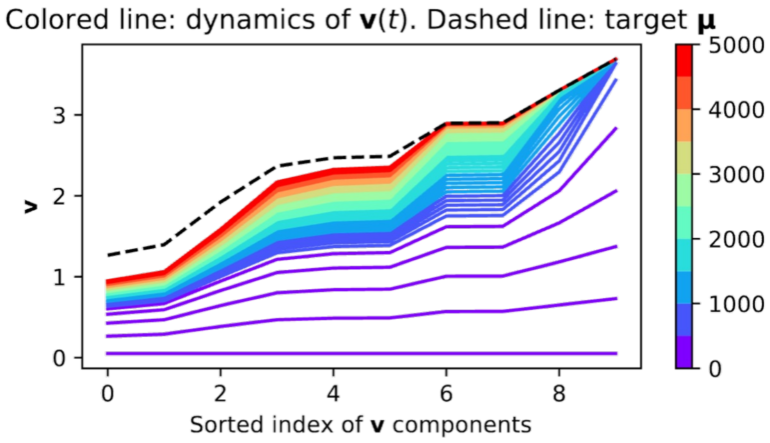
JoMA for Nonlinear activation

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

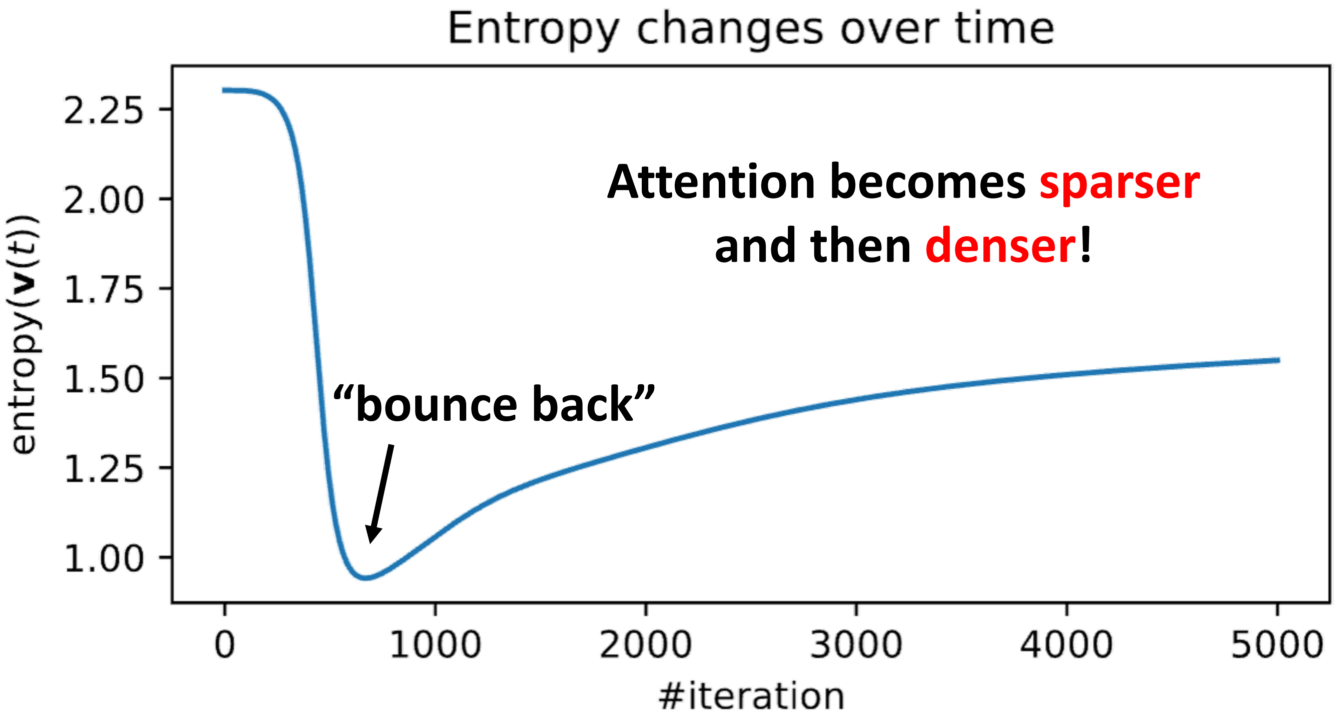
Nonlinear

Modified MLP (lower layer)

How the entropy of attention changes over time?



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

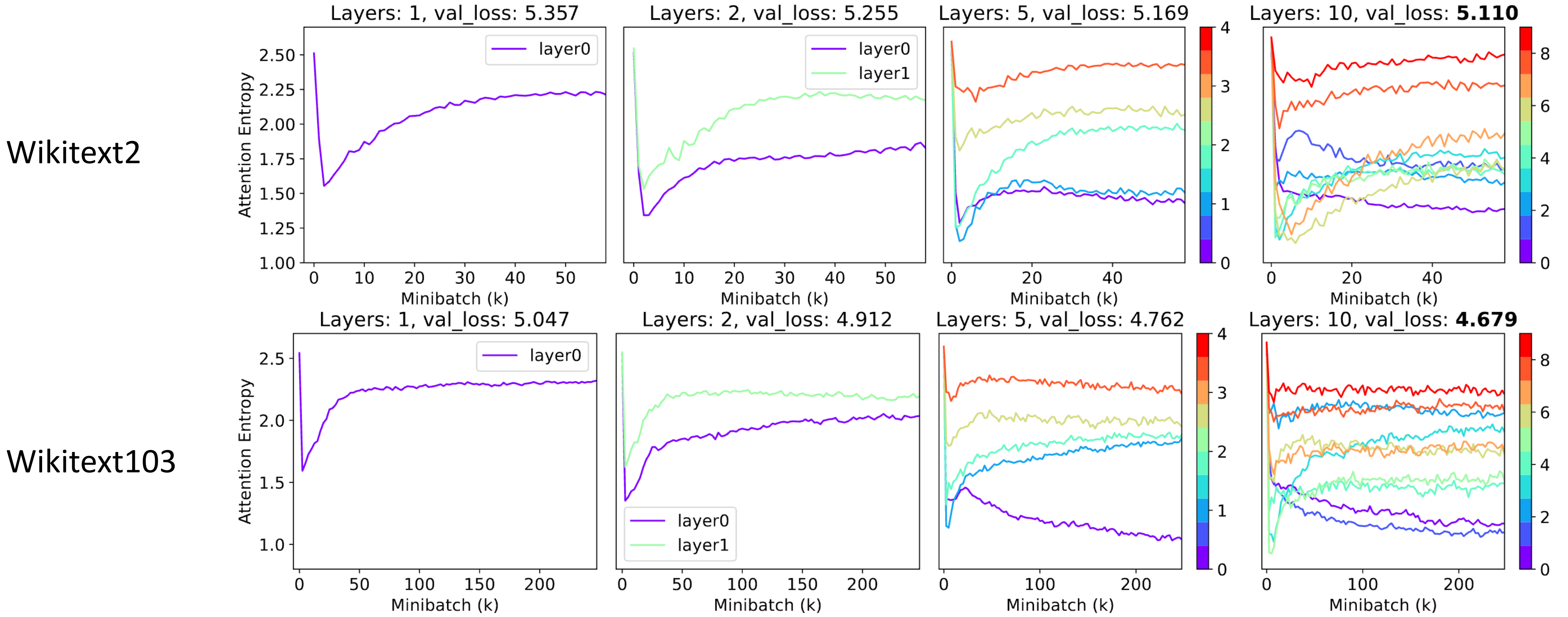


Entropy changes over time

Attention becomes **sparser** and then **denser**!

“bounce back”

Real-world Experiments

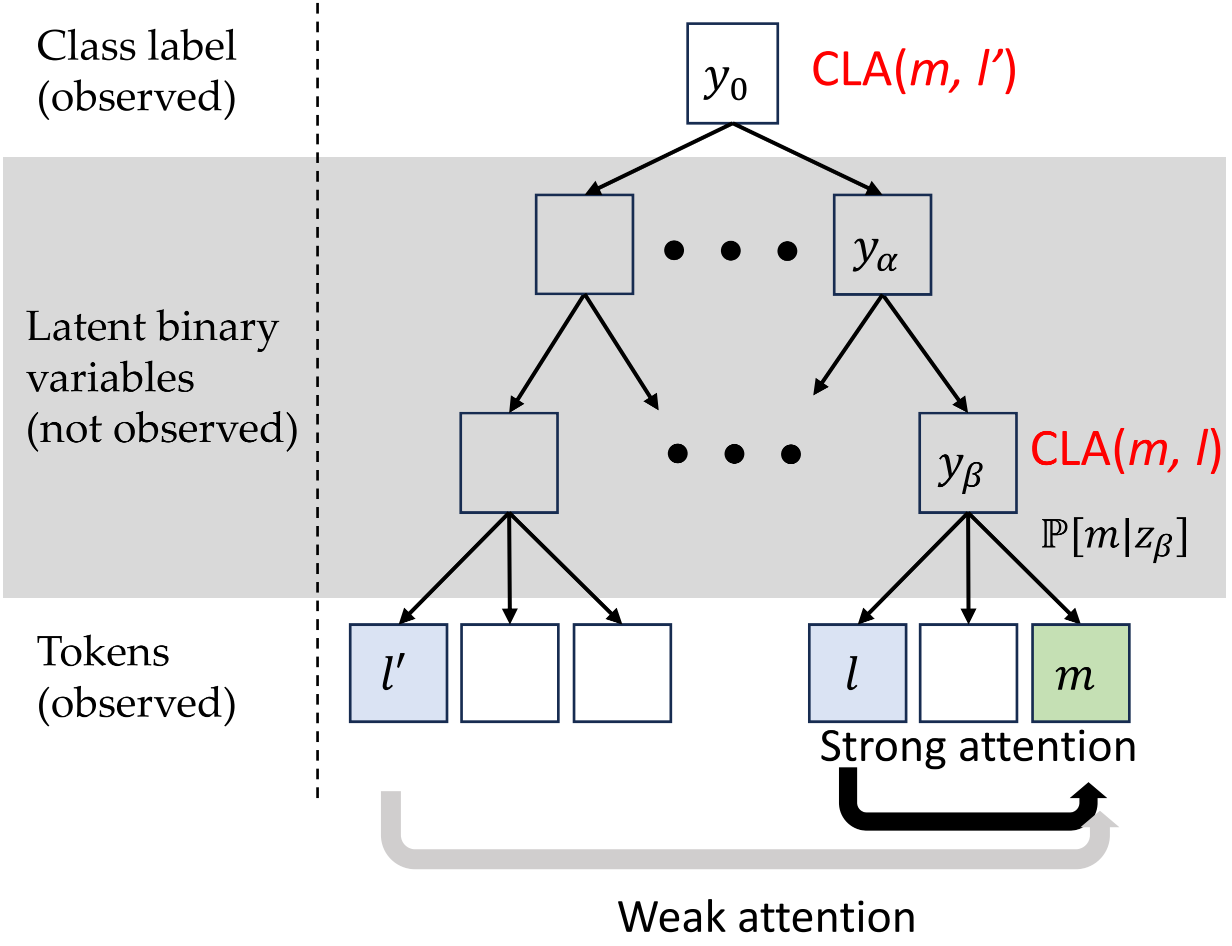


Why is this “bouncing back” property useful?

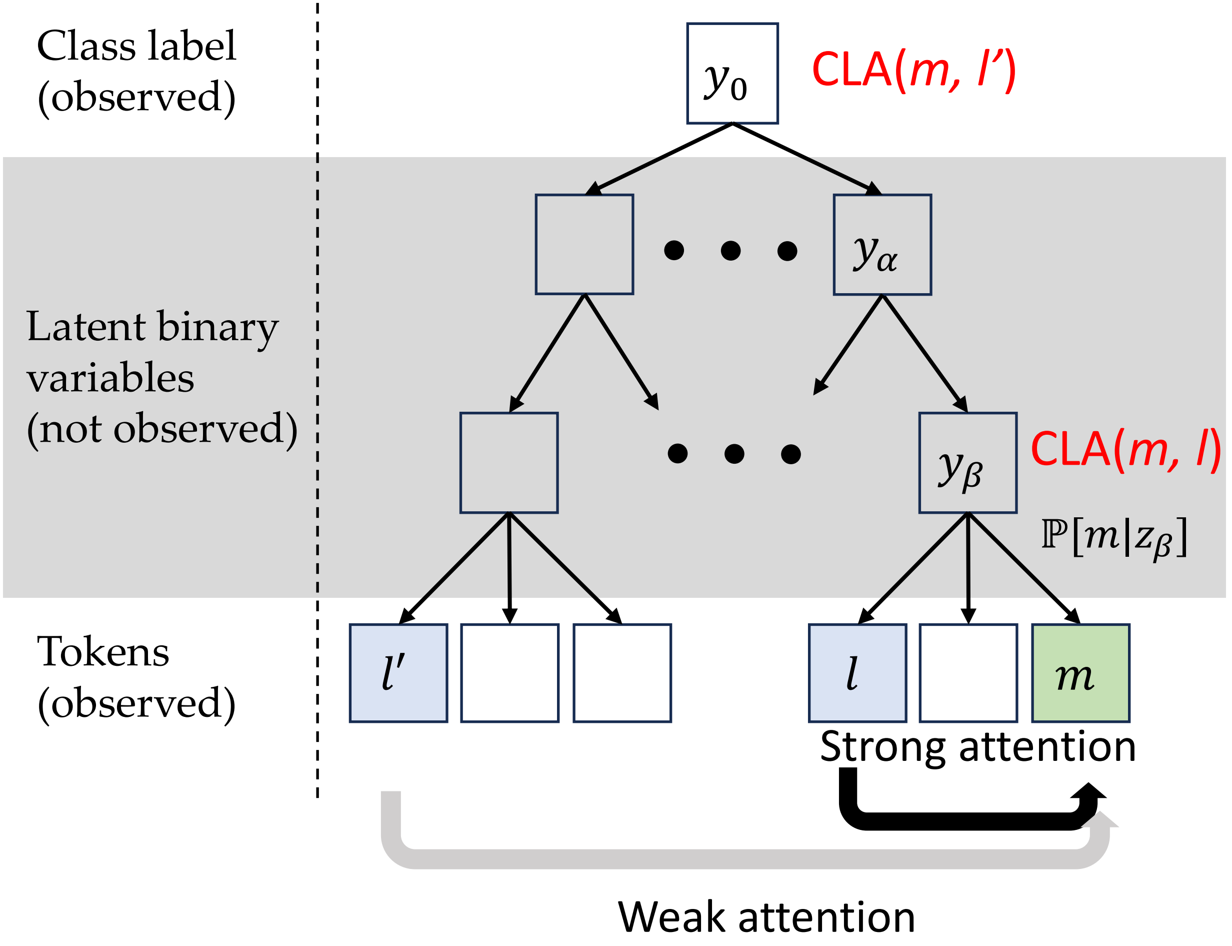
It seems that it only slows down the training??

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

Data Hierarchy & Multilayer Transformer



Data Hierarchy & Multilayer Transformer



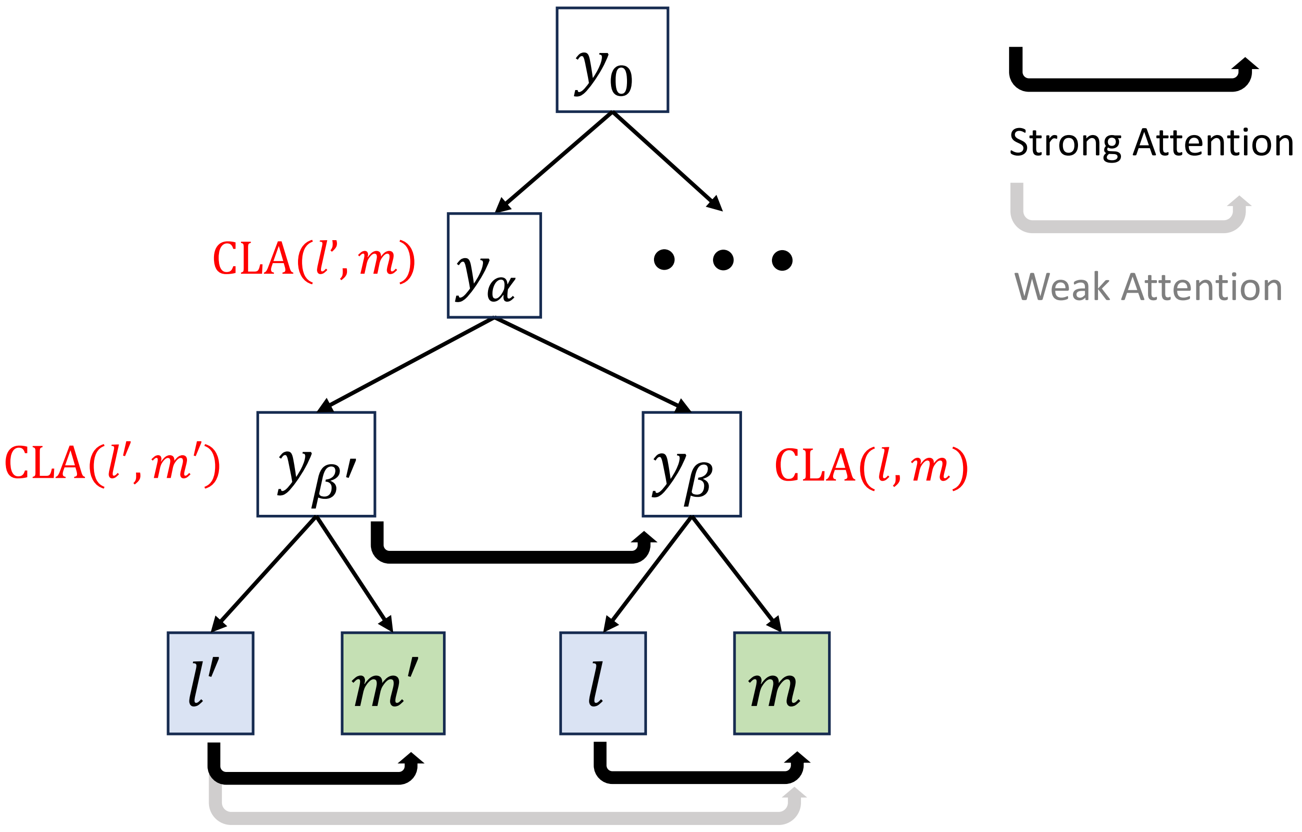
Theorem 5

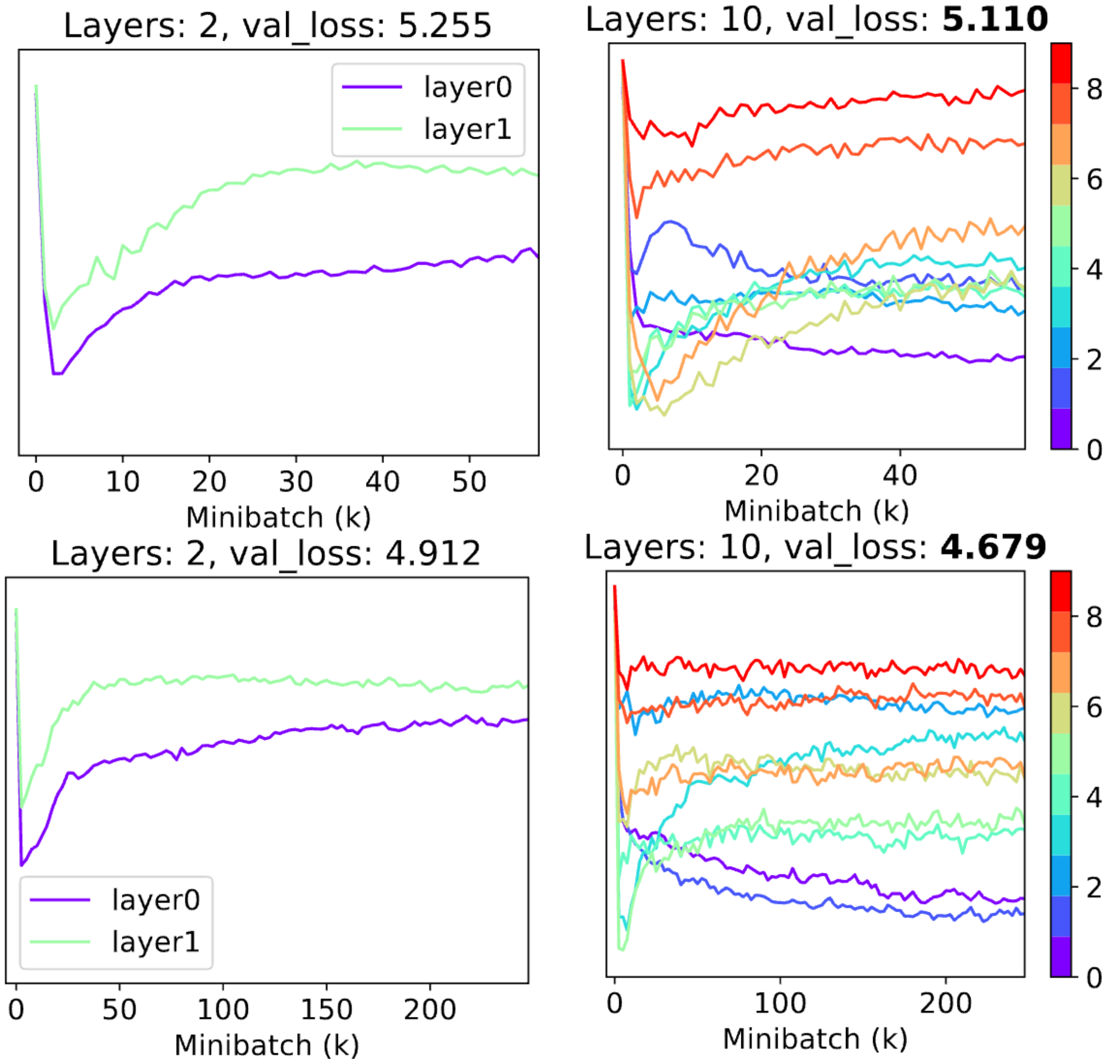
$$\mathbb{P}[l|m] \approx 1 - \frac{H}{L}$$

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

L: total height of the hierarchy

Deep Latent Distribution




Layers: 2, val_loss: 5.255

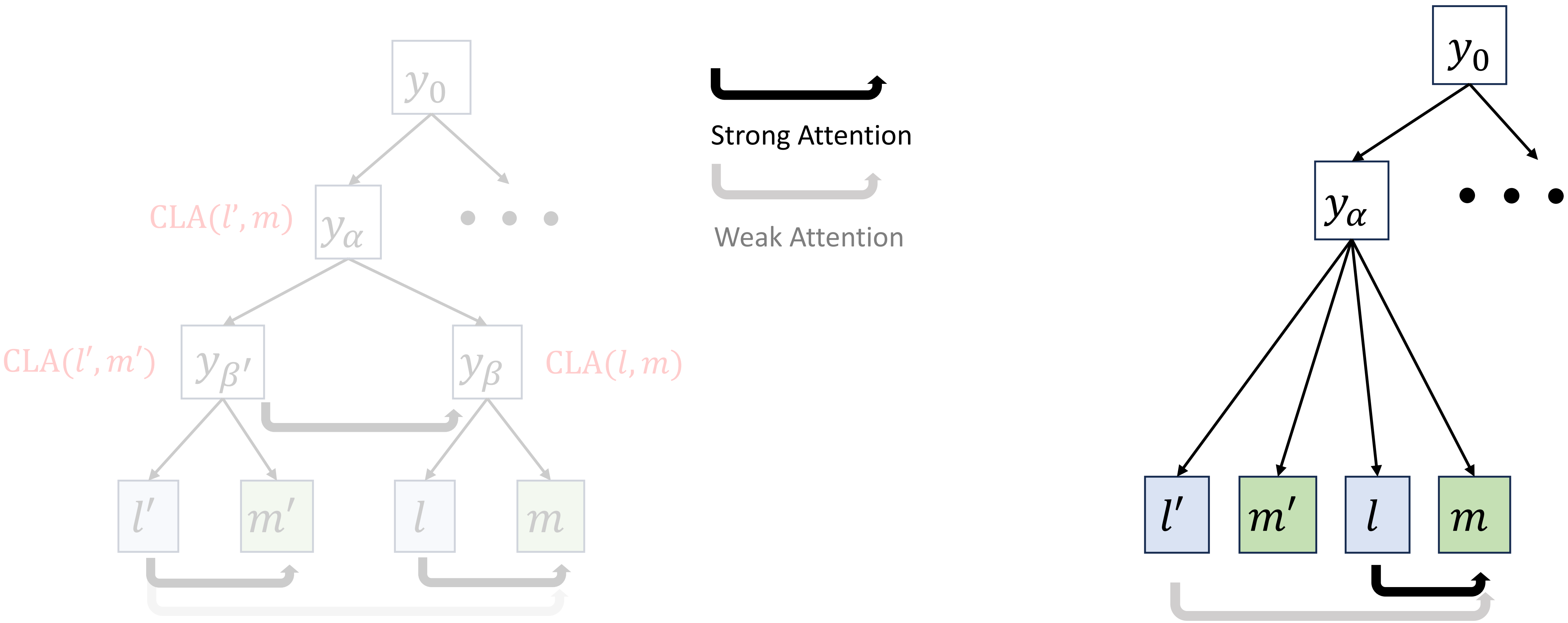
Layers: 10, val_loss: **5.110**

Layers: 2, val_loss: 4.912

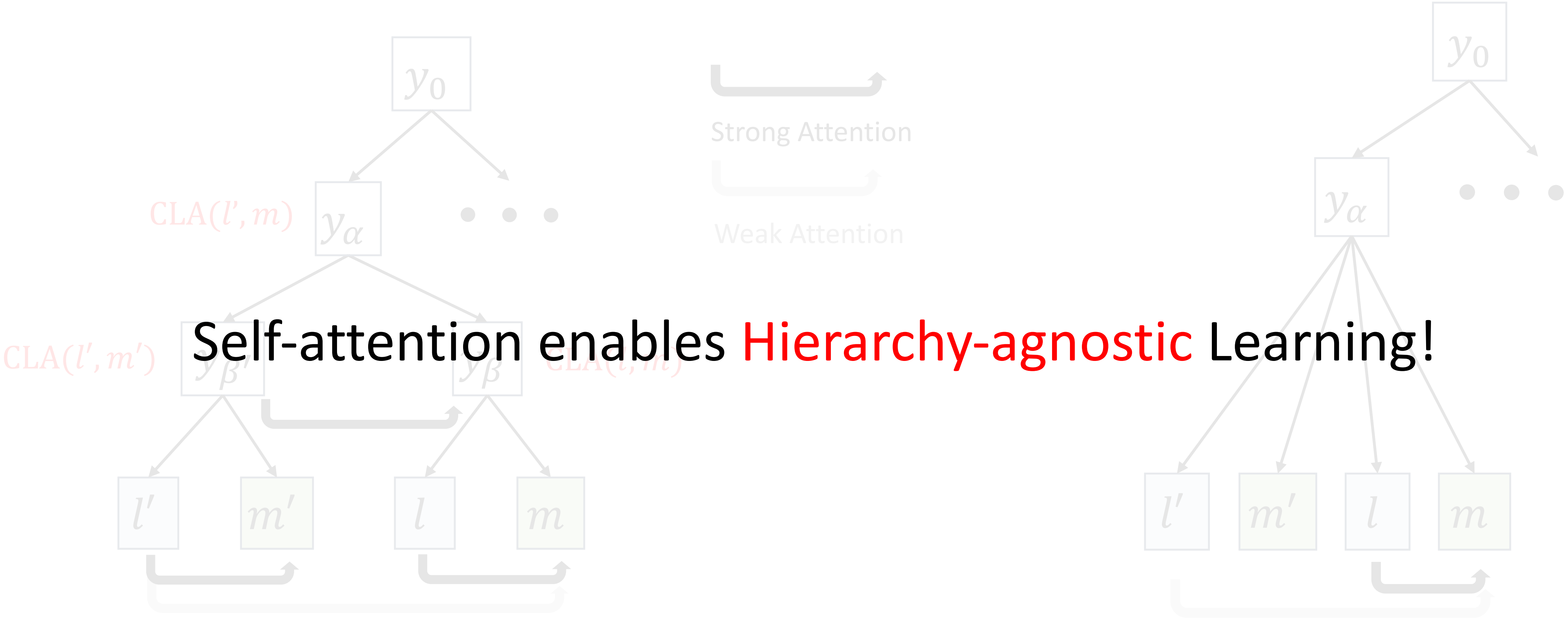
Layers: 10, val_loss: **4.679**

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

Shallow Latent Distribution

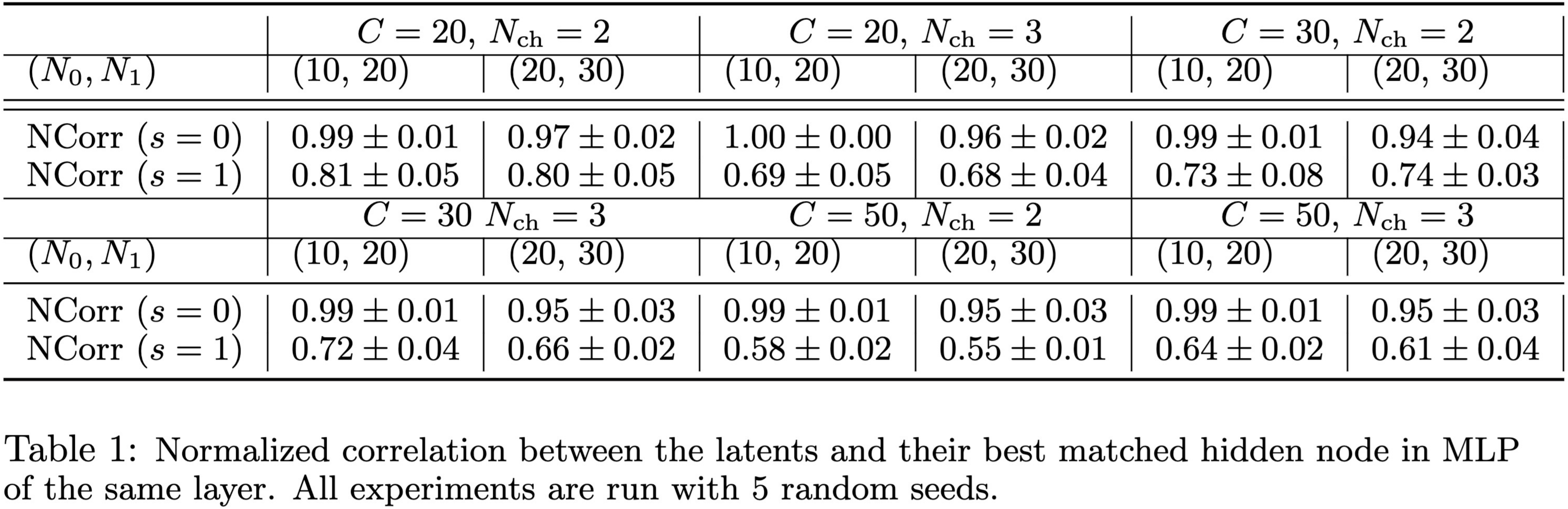


Hierarchy-agnostic Learning



Self-attention enables Hierarchy-agnostic Learning!

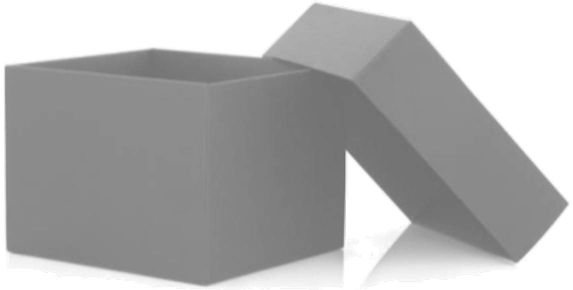
Verification of Hierarchical Intuitions



	$C = 20,\ N_{\text{ch}} = 2$		$C = 20,\ N_{\text{ch}} = 3$		$C = 30,\ N_{\text{ch}} = 2$	
(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.

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!