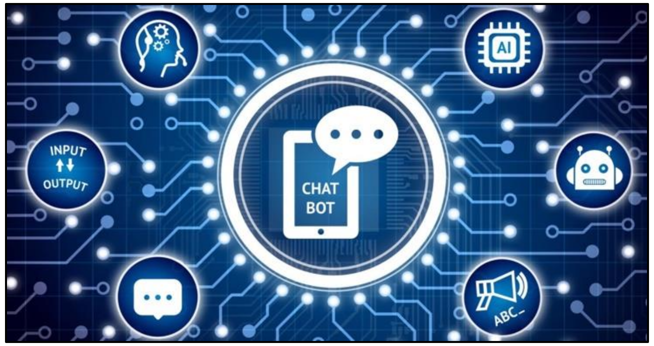
Exploring Training Mechanism in Transformers via the Lens of Training Dynamics

Yuandong Tian
Research Scientist Director

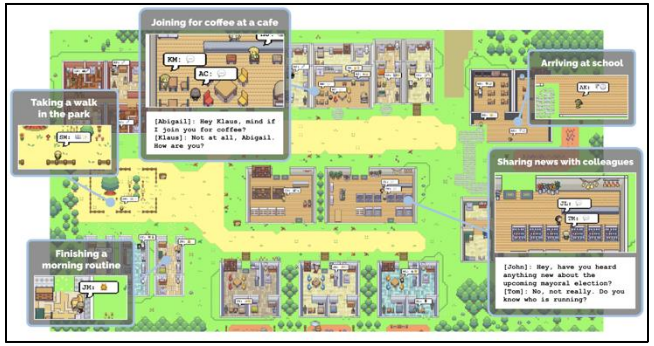
Meta GenAI

Large Language Models (LLMs)

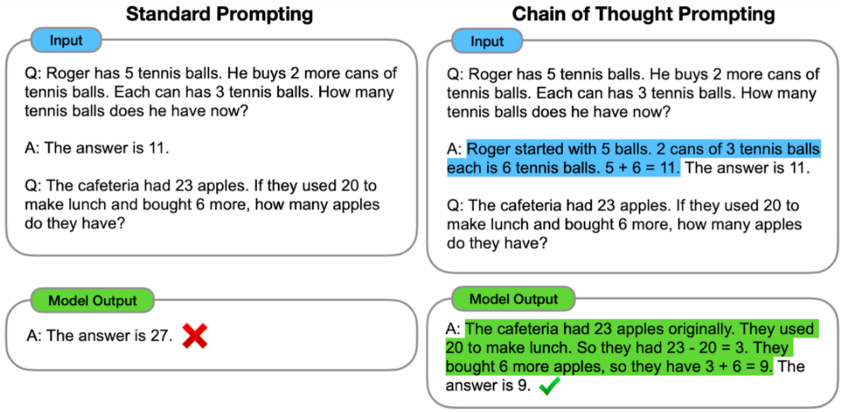


Conversational AI

Content Generation



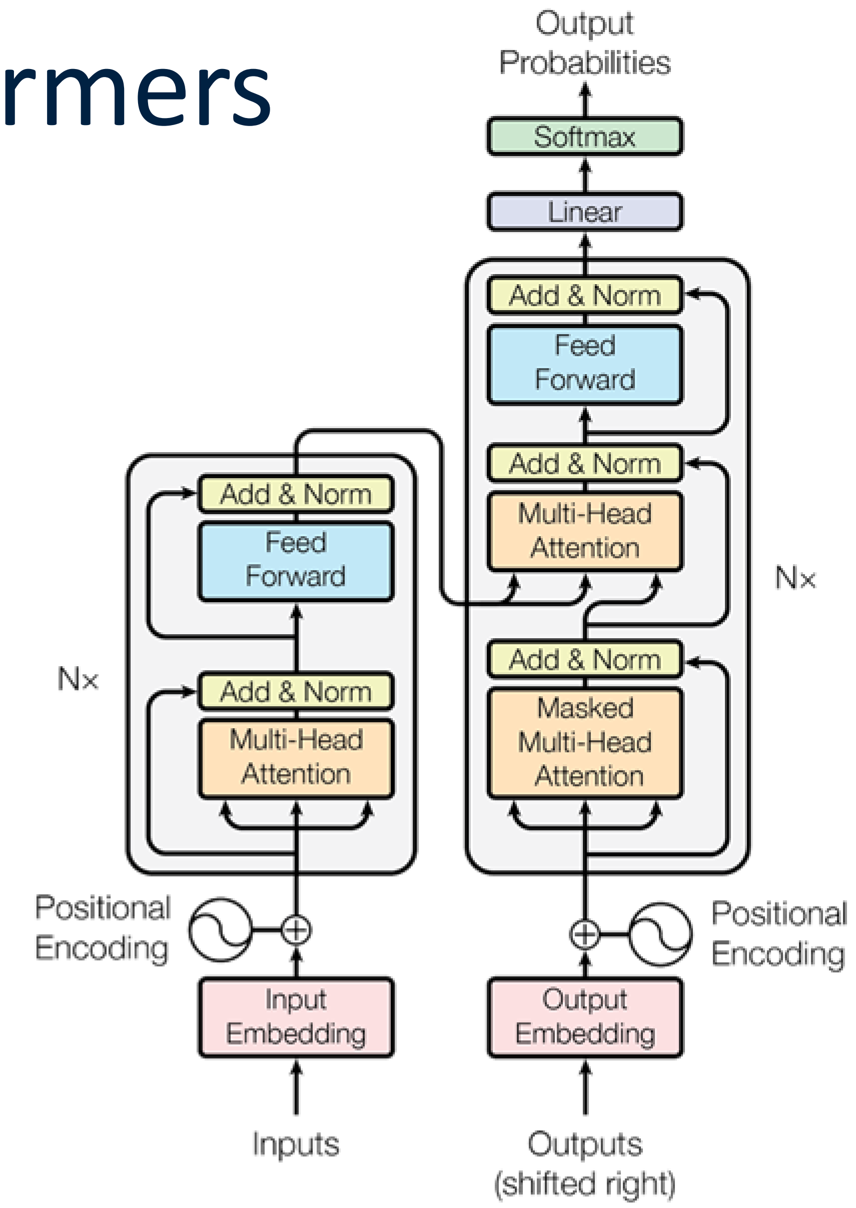
AI Agents



Reasoning

Planning

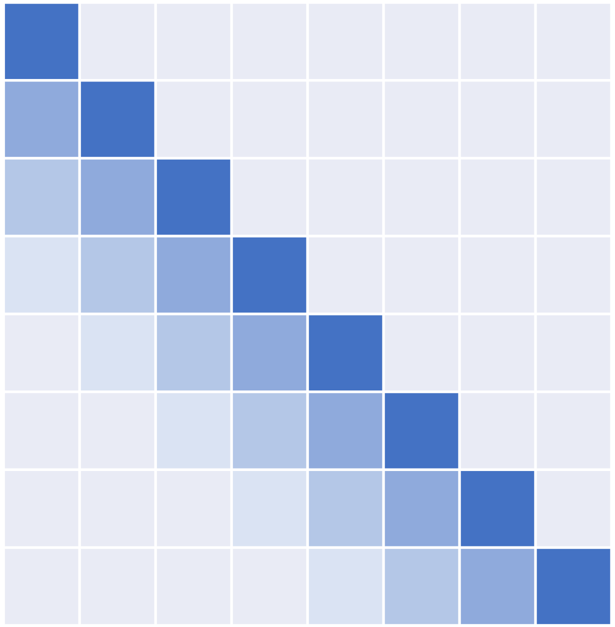
Transformers



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

Key K

Query Q

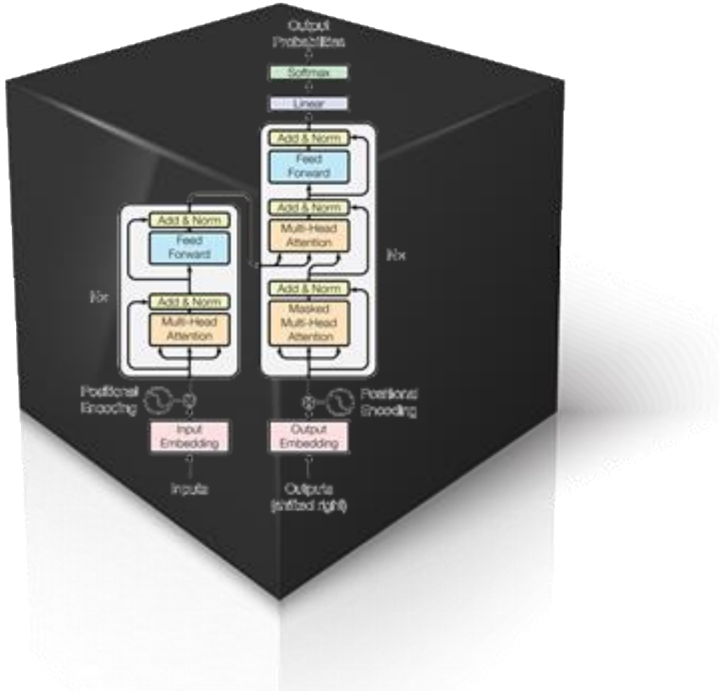


Attention mechanism

How does Transformer work?

Input

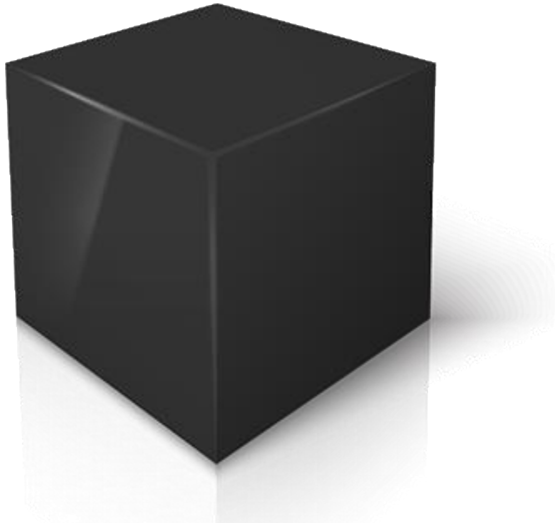
This is an apple



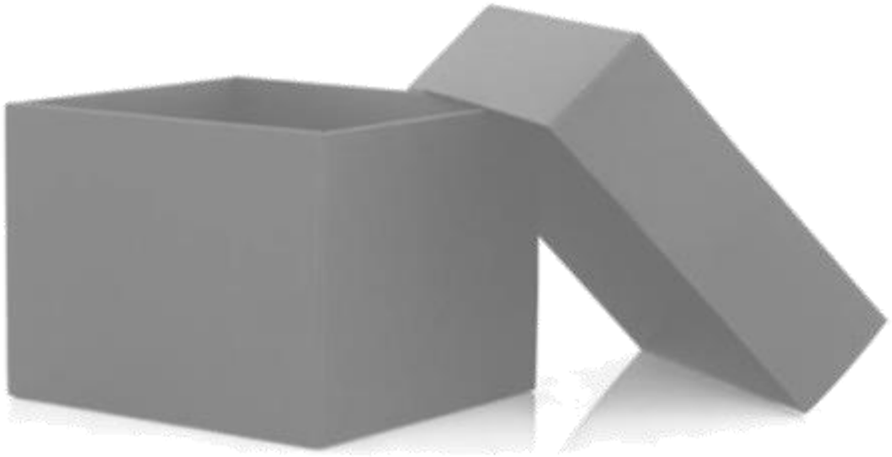
"Some Nonlinear Transformation"

Output

Black-box versus White-box



Black box



White box

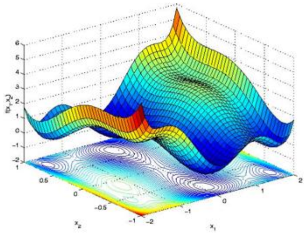
Three Angles

Understanding how Deep Models work

Expressibility

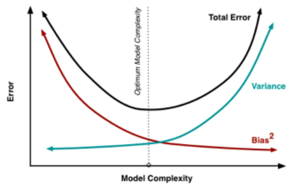
"Neural Network is a universal approximator"
"Deep Models can express functions more efficiently than shallow ones"

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."

Three Angles

Understanding how Deep Models work

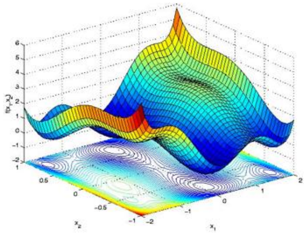
Expressibility

\+ -

\- +

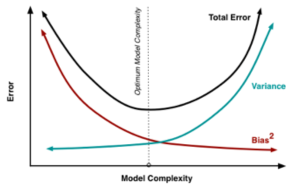
"Neural Network is a universal approximator"
"Deep Models can express functions more efficiently than shallow ones"

Optimization



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

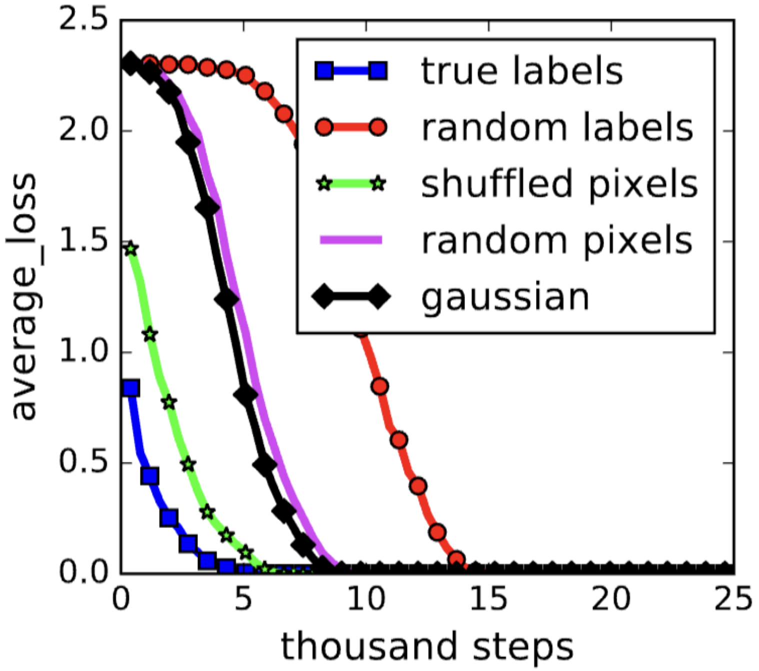
Generalization



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

Which path should we take?

Rethinking Generalization



(a) learning curves

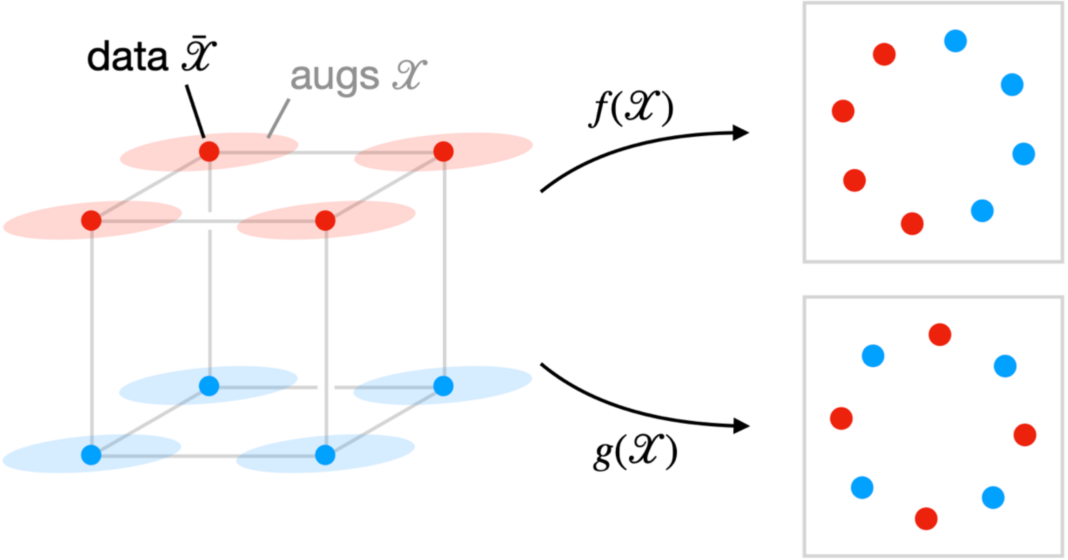
model	# params	random crop	weight decay	train accuracy	test accuracy
Inception	1,649,402	yes	yes	100.0	89.05
		yes	no	100.0	89.31
		no	yes	100.0	86.03
		no	no	100.0	85.75
(fitting random labels)		no	no	100.0	9.78
Inception w/o BatchNorm	1,649,402	no	yes	100.0	83.00
		no	no	100.0	82.00
(fitting random 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



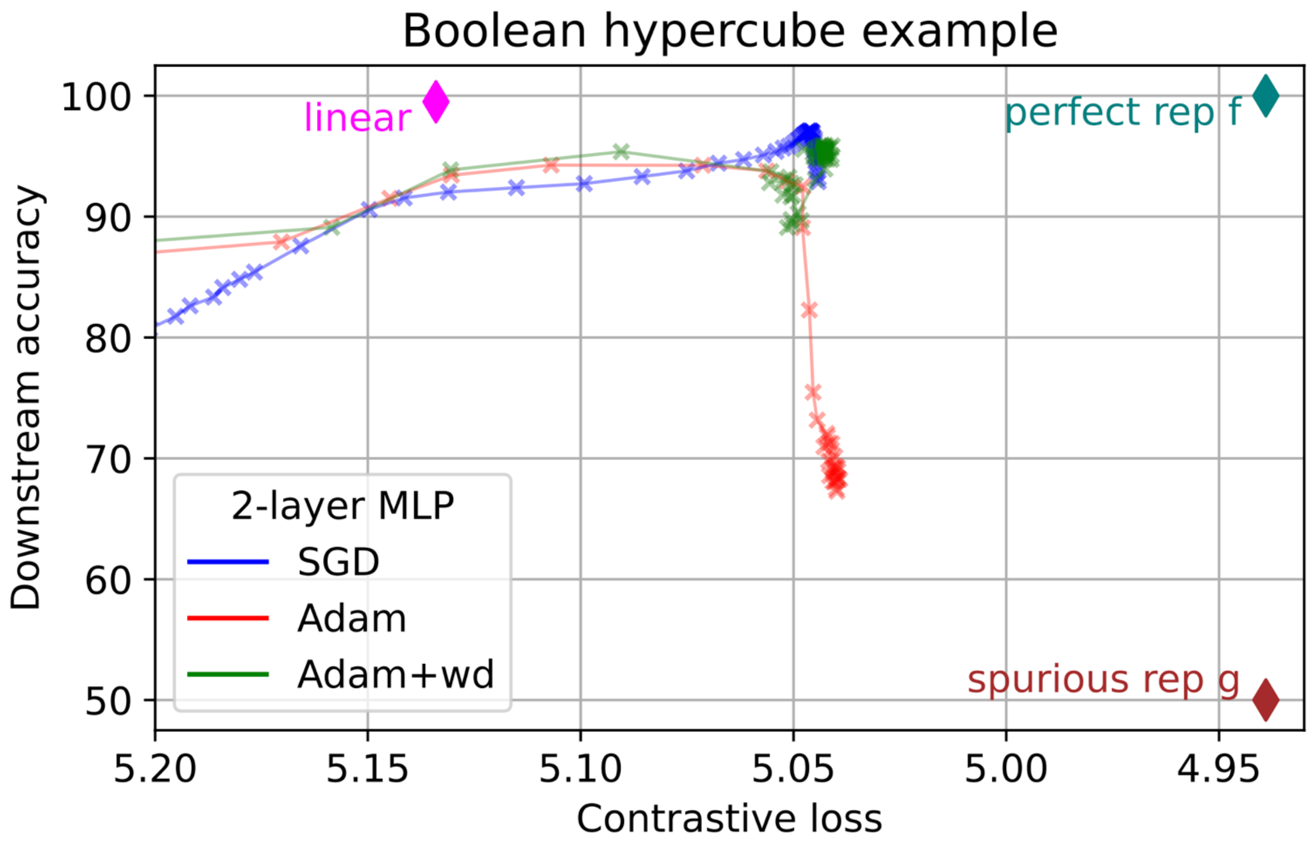
Pretraining: $L_{\mathrm{cont}}(g) \approx L_{\mathrm{cont}}(f)$

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

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

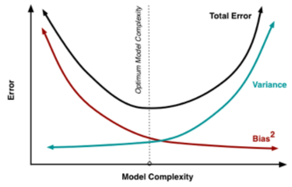


Representation	Contrastive loss	Accuracy (%)
$\exists f$ (perfect)	4.939	100
$\exists g$ (spurious)	4.939	50
MLP + Adam	5.039 ± 0.001	74.1 ± 4.3
MLP + Adam + wd	5.040 ± 0.002	89.5 ± 4.9
Linear	5.134 ± 0.002	99.5 ± 0.1

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

Lesson learned?

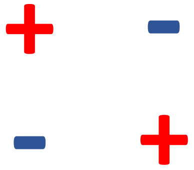
Generalization



Architecture ✗

training dynamics ✗

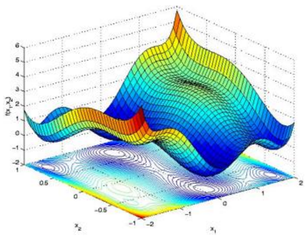
Expressibility



Architecture ✓

training dynamics ✗

Optimization



Architecture ✗

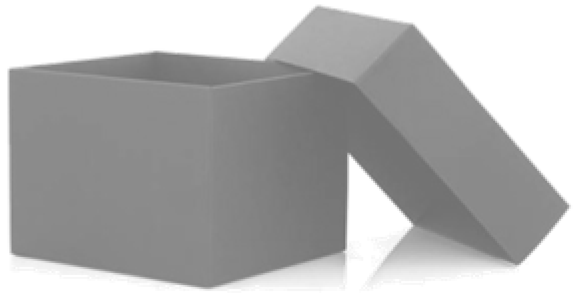
training dynamics ✓

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 ✓

Roadmap of Theoretical Analysis

Fix Representation, check how Self-attention works

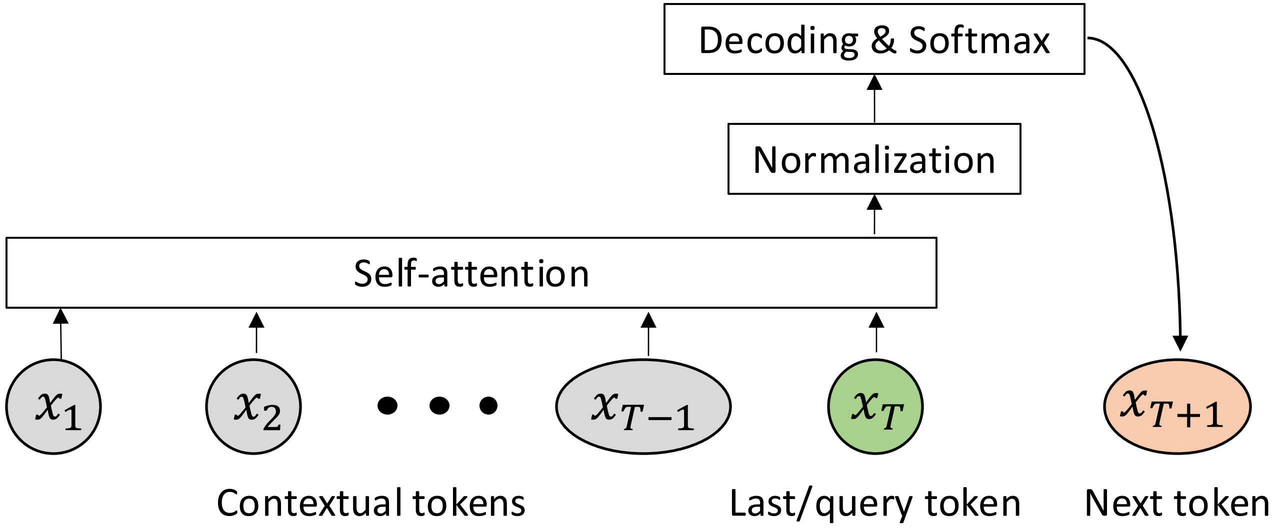
Check what representation it learns

Roadmap of Theoretical Analysis

Fix Representation, check how Self-attention works

Check what representation it learns

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

<u>Objective:</u>

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

[*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)

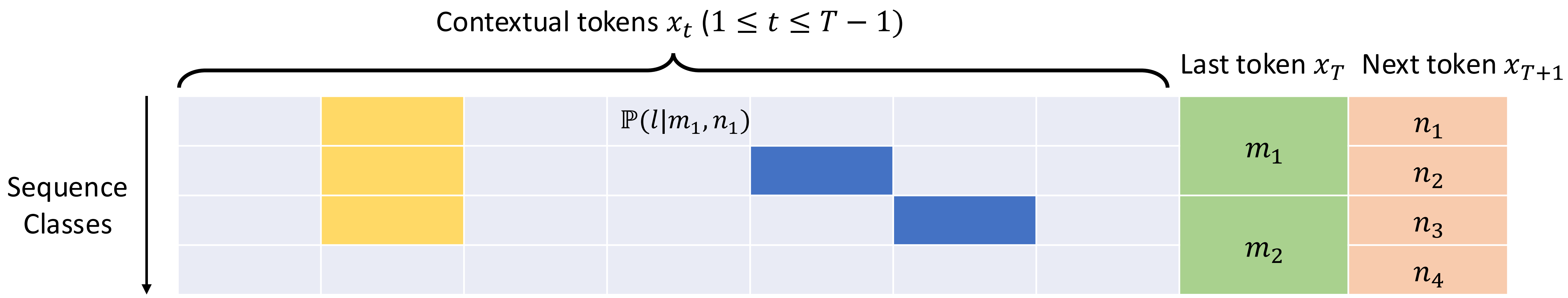
- Then the dynamics becomes easier to analyze

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

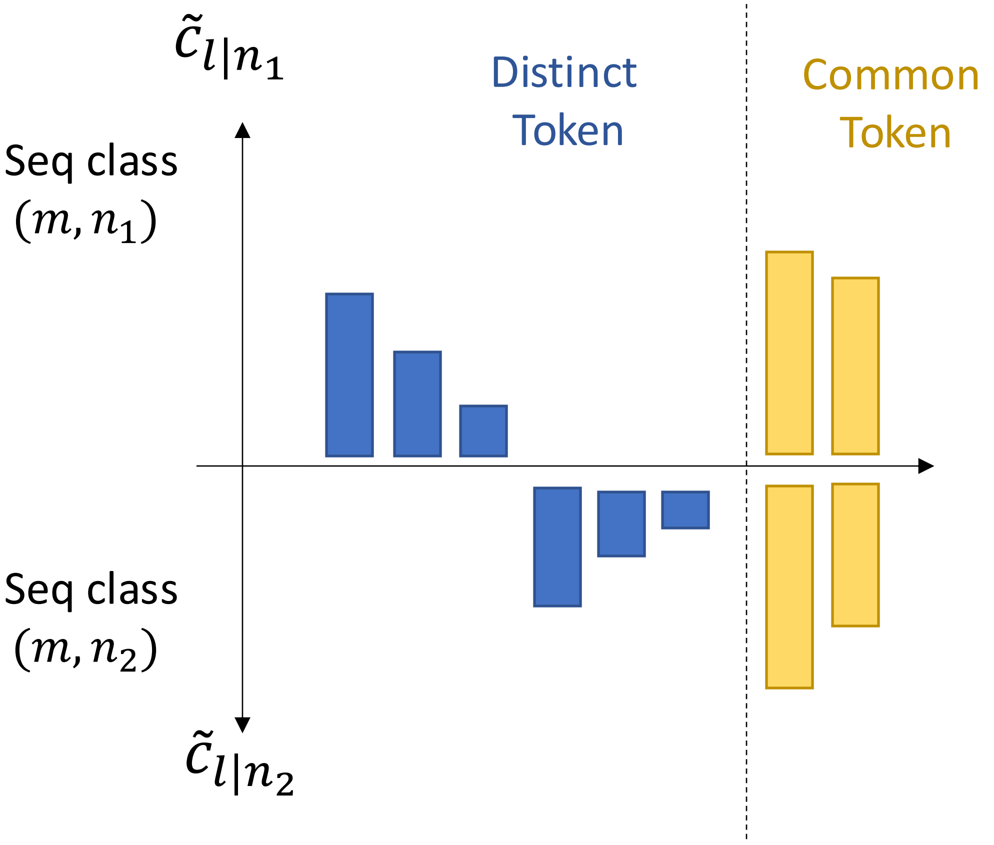
$\mathbb{P}(l|m, n) = \mathbb{P}(l|n)$ is the conditional probability of token l given last token $x_T = m$ and $x_{T+1} = n$

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

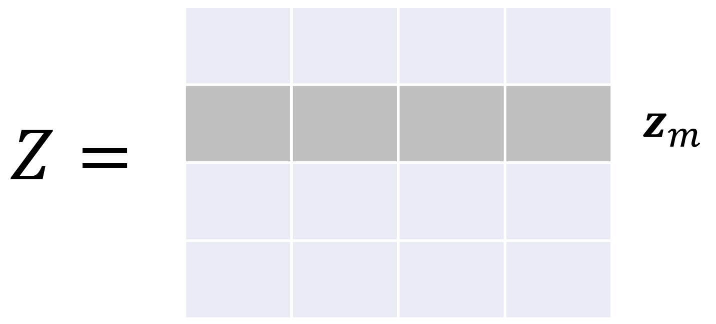
At initialization



Co-occurrence probability

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

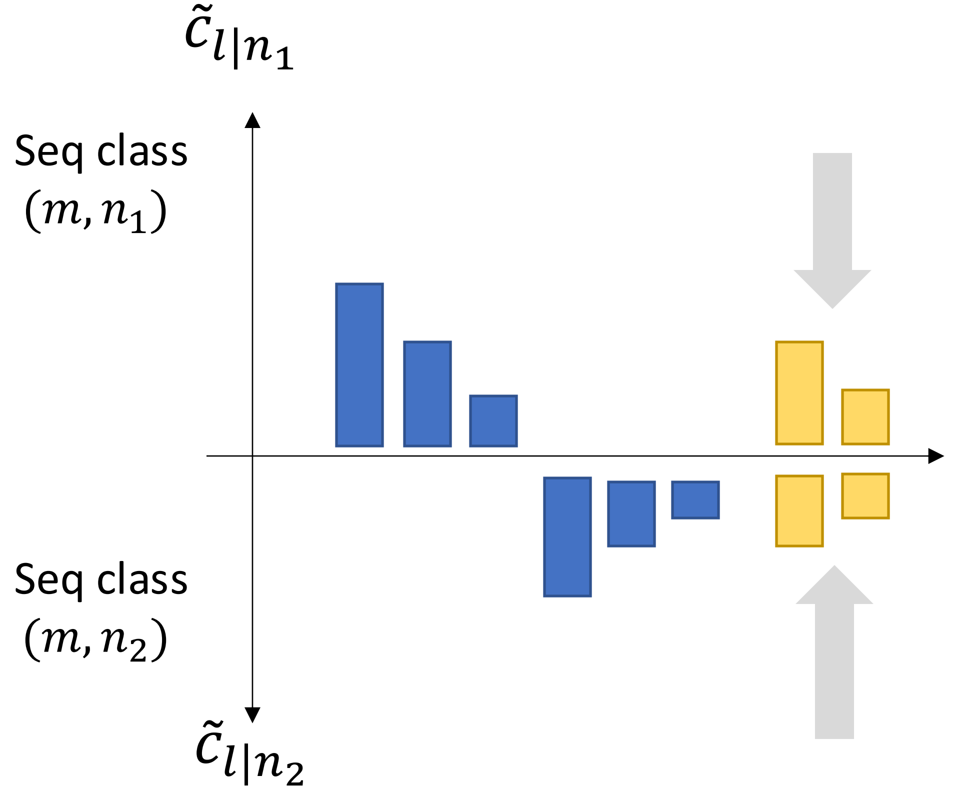
Initial condition: $z_{ml}(0) = 0$



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

Overall Picture of the Training Dynamics

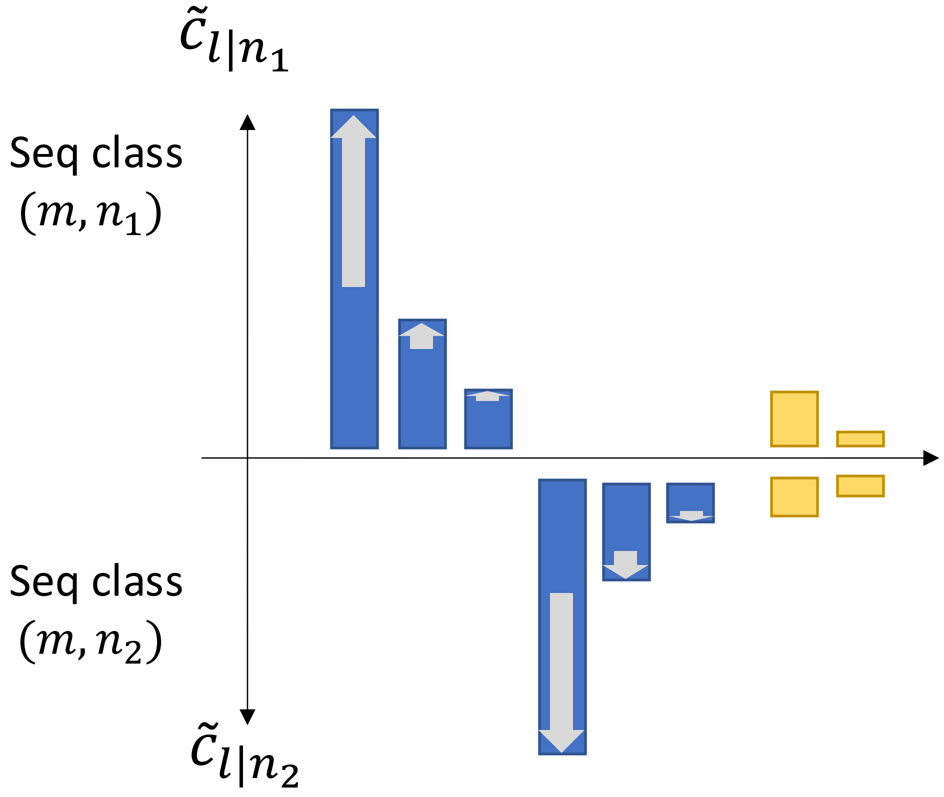
Common Token Suppression



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

Overall Picture of the Training Dynamics

Winners-emergence



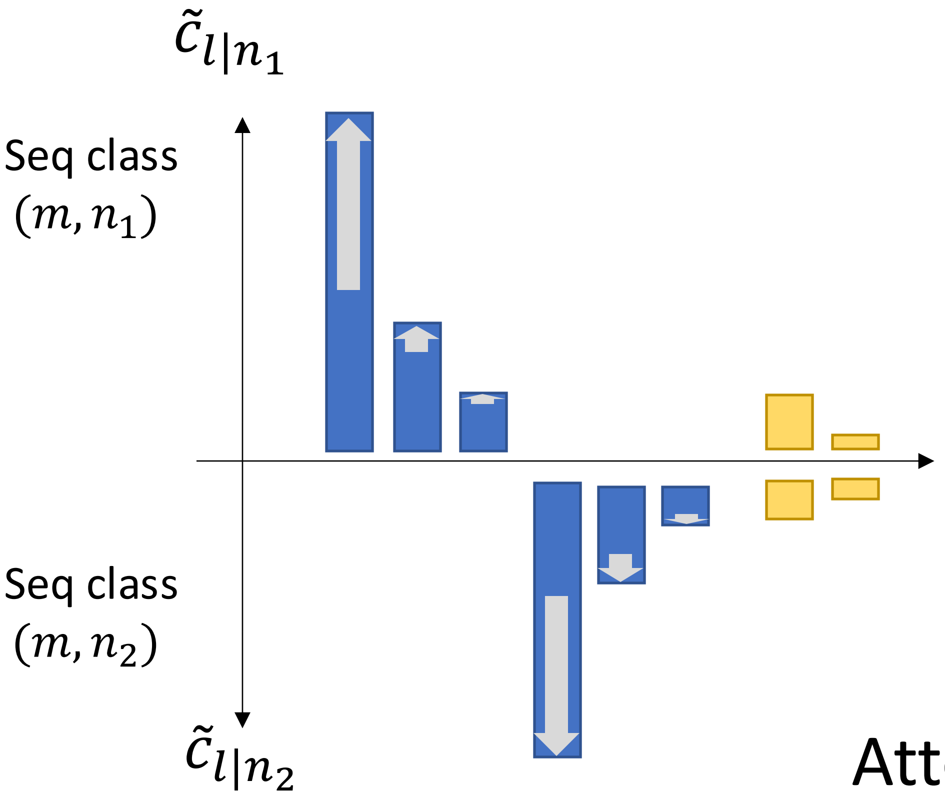
(a) $\dot{z}_{ml} < 0$, for common token l

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

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

Overall Picture of the Training Dynamics

<u>Winners-emergence</u>



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

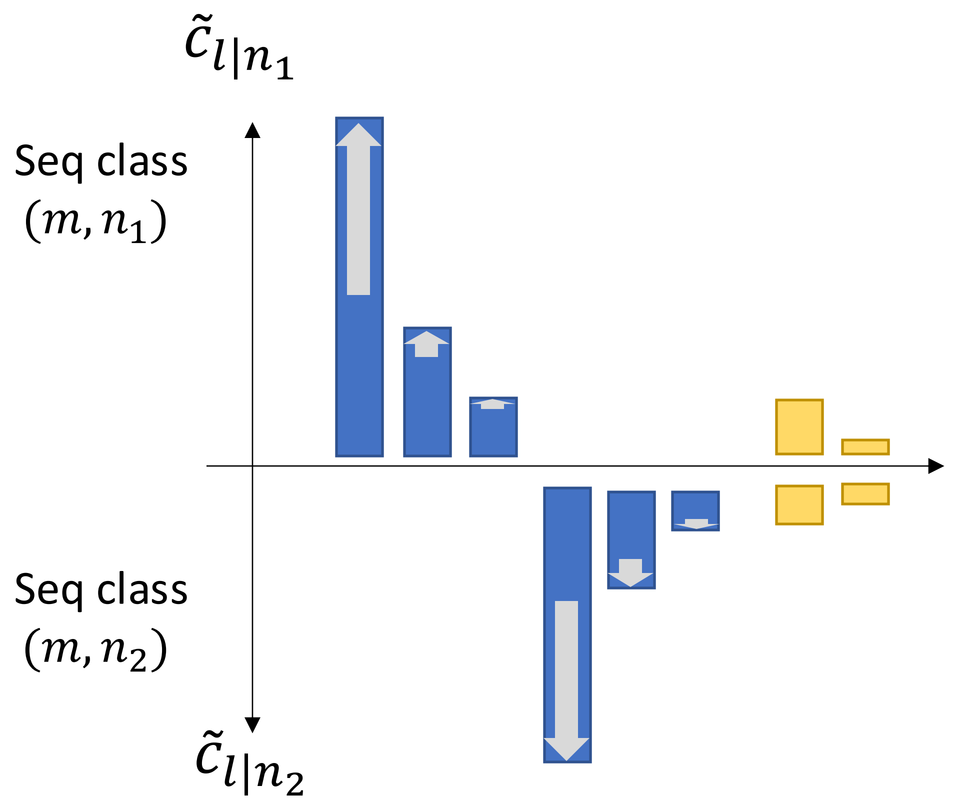
(b) $\dot{z}_{ml} > 0$, for distinct token l

(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:

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

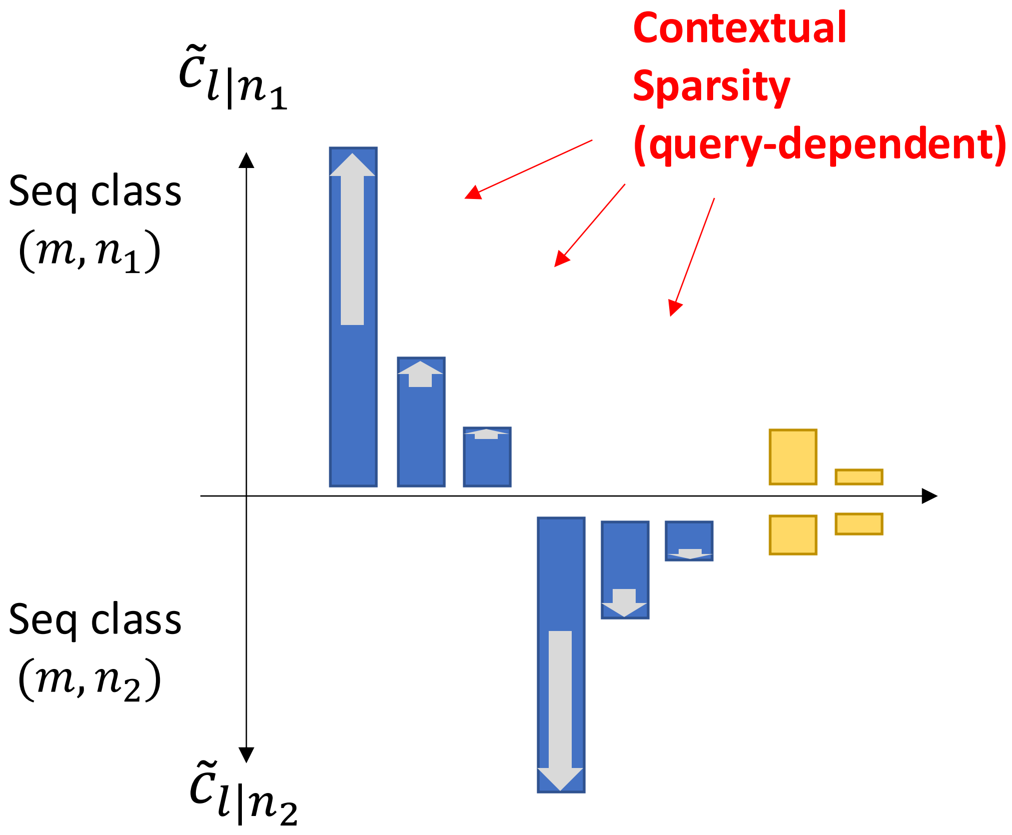
If l_0 is the dominant token: $r_{l_0/l|n}(0) > 0$ for all $l \neq l_0$ then

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

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:

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

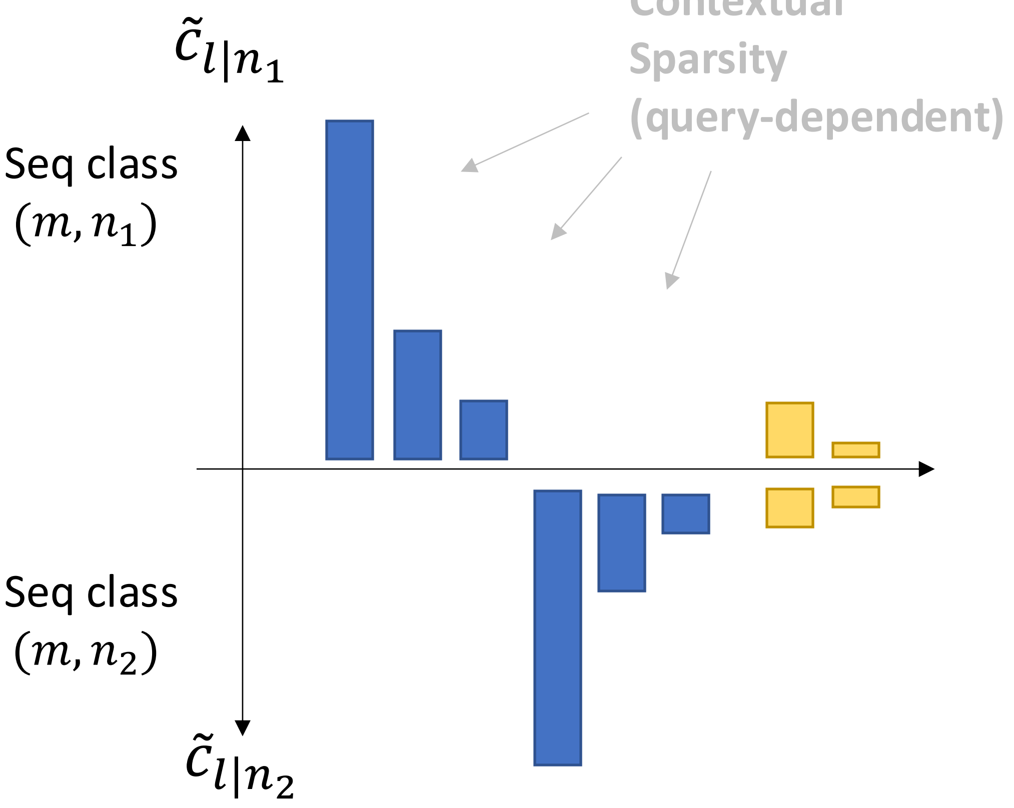
If l_0 is the dominant token: $r_{l_0/l|n}(0) > 0$ for all $l \neq l_0$ then

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

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**:

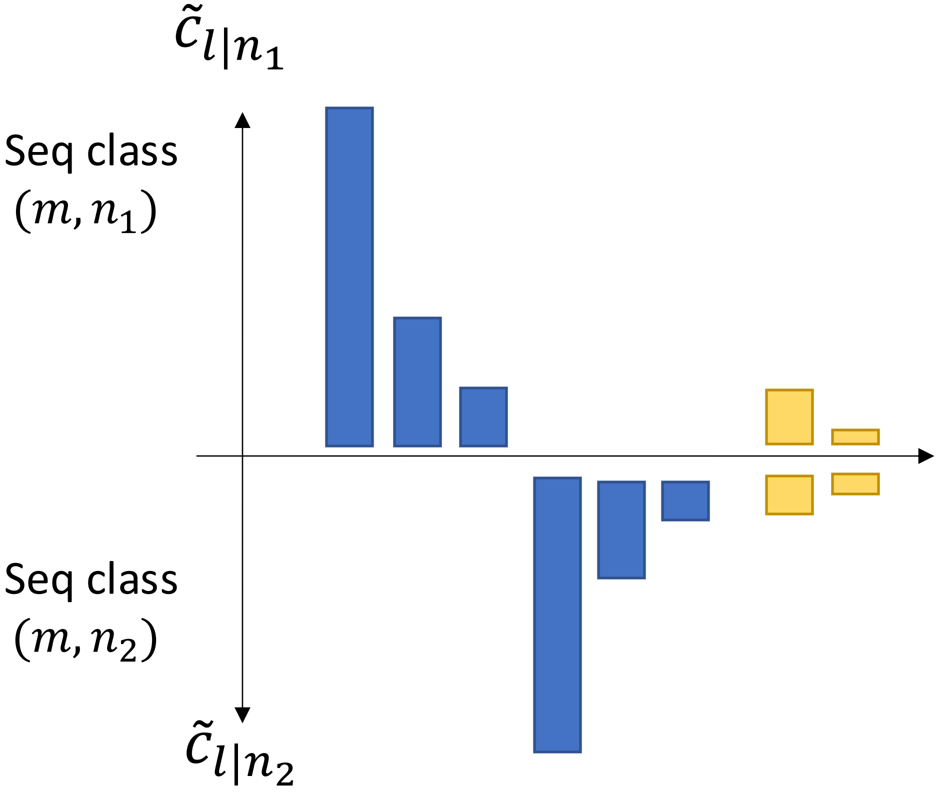
When $t \geq t_0 = O\left(\frac{2K\ln M}{\eta_Y}\right)$, $B_n(t) = O(\ln\ln t)$

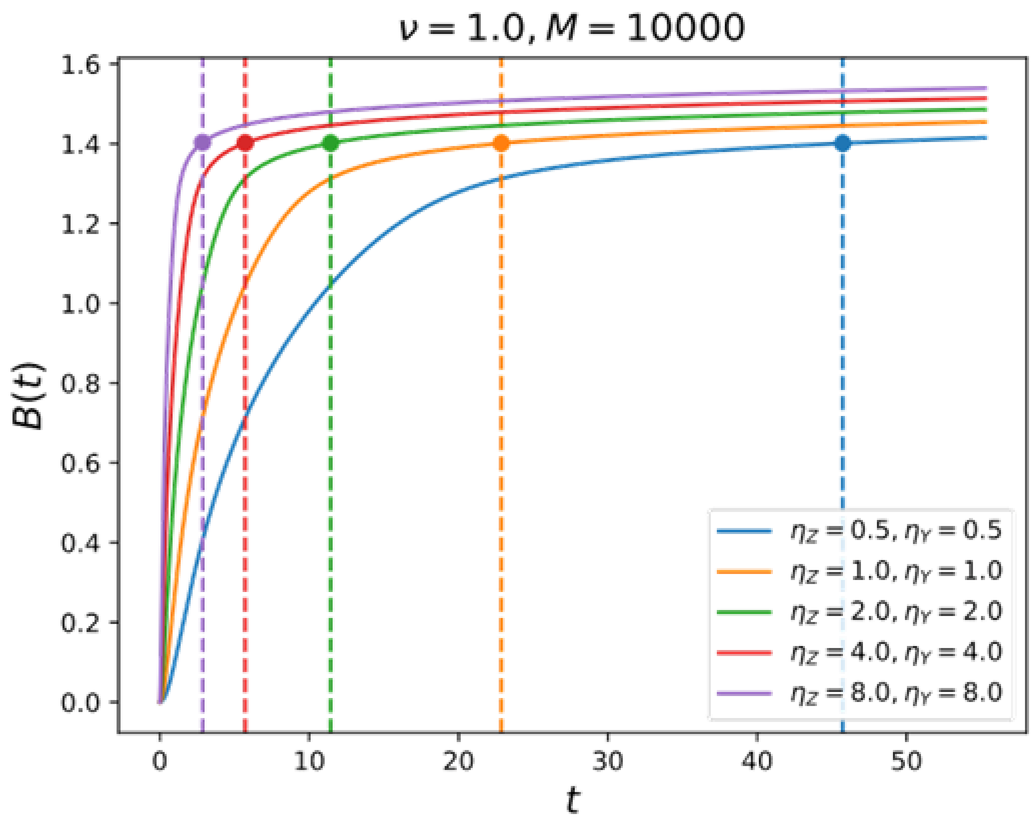
(1) η_Z and η_Y are large, $B_n(t)$ is large and attention is sparse

(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

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

Simple Real-world Experiments

WikiText2
(original parameterization)

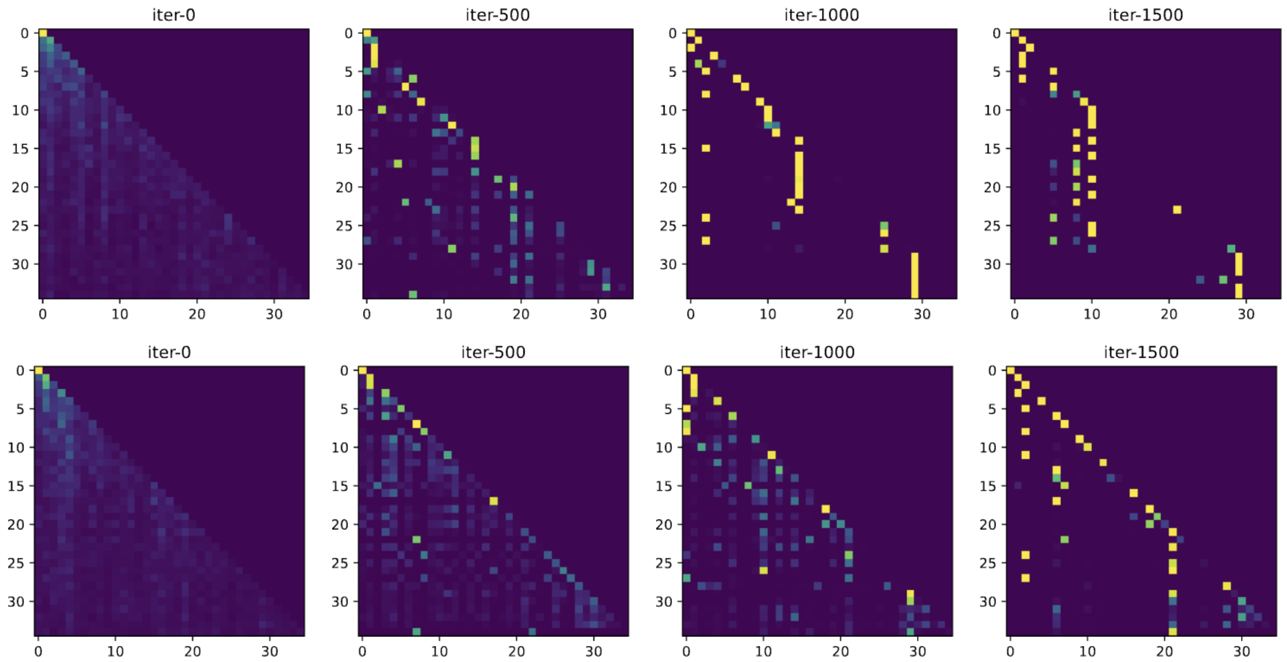


Figure 7: Attention patterns in the lowest self-attention layer for 1-layer (top) and 3-layer (bottom) Transformer trained on WikiText2 using SGD (learning rate is 5). Attention becomes sparse over training.

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

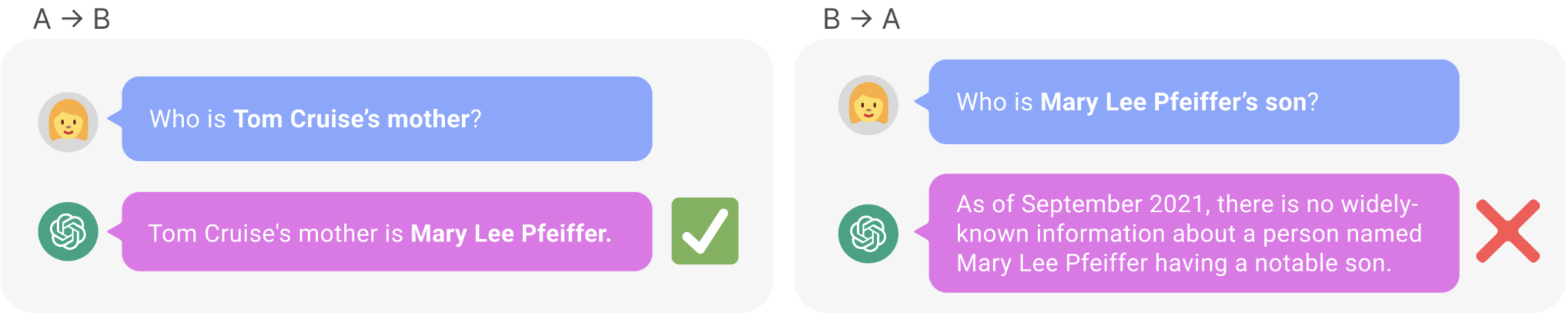


Figure 1: **Inconsistent knowledge in GPT-4.** GPT-4 correctly gives the name of Tom Cruise's mother (left). Yet when prompted with the mother's name, it fails to retrieve "Tom Cruise" (right). We hypothesize this ordering effect is due to the Reversal Curse. Models trained on "*A* is *B*" (e.g. "Tom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

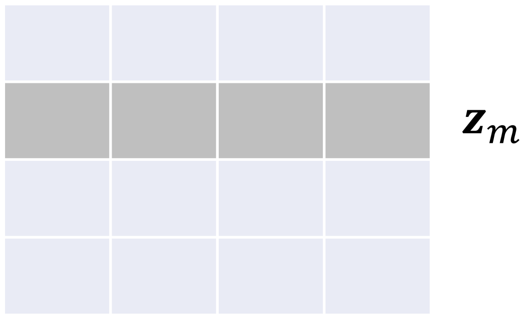
[L. Berglund et al, *The Reversal Curse: LLMs trained on "A is B" fail to learn "B is A"*, ICLR 2024]

How to explain “Reversal Curse”?

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

is **not** symmetric

$Z =$



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

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

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

for some constant $c > 0$ *and*

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

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

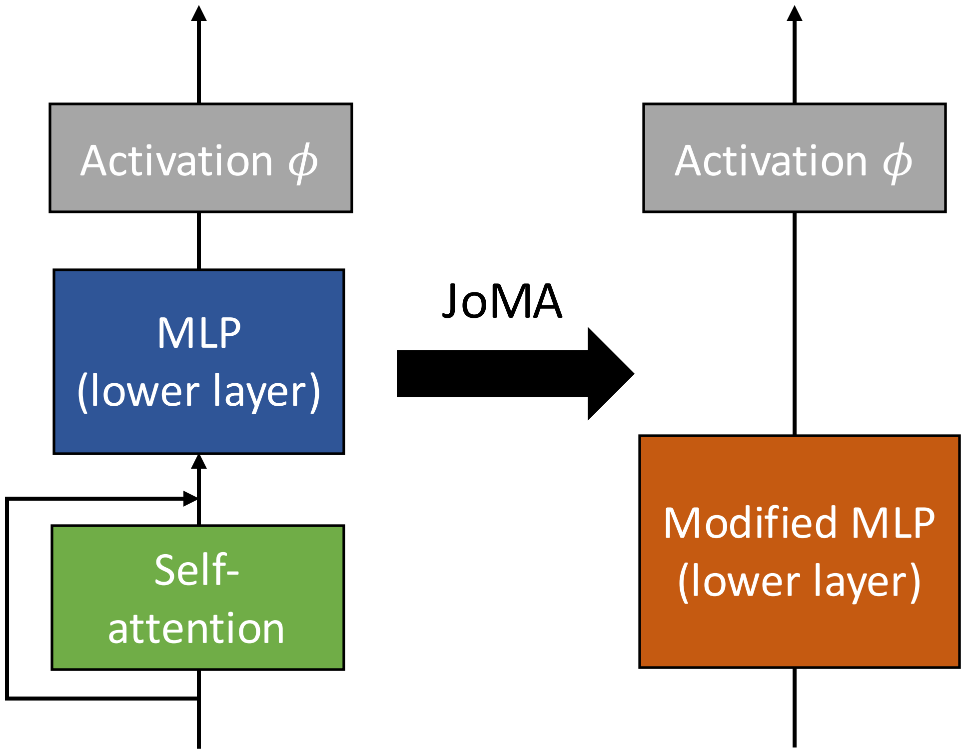
How to get rid of the assumptions?

- 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

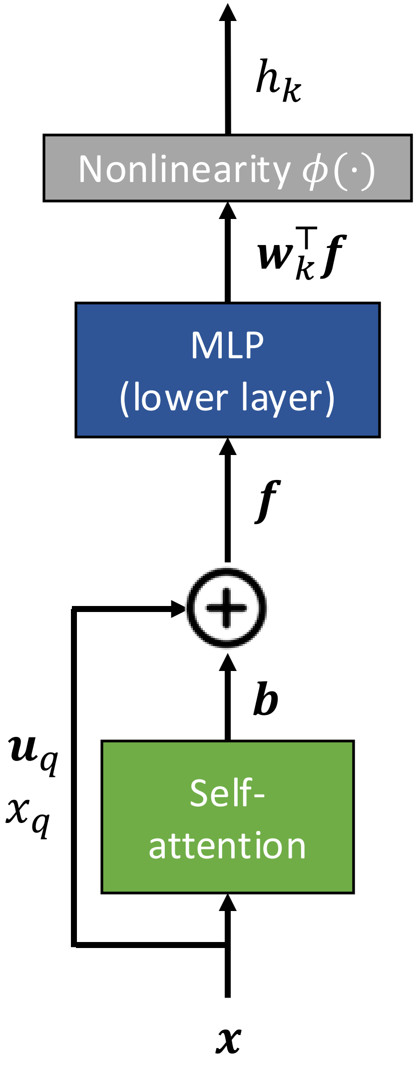


Main Contributions:

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”

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

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

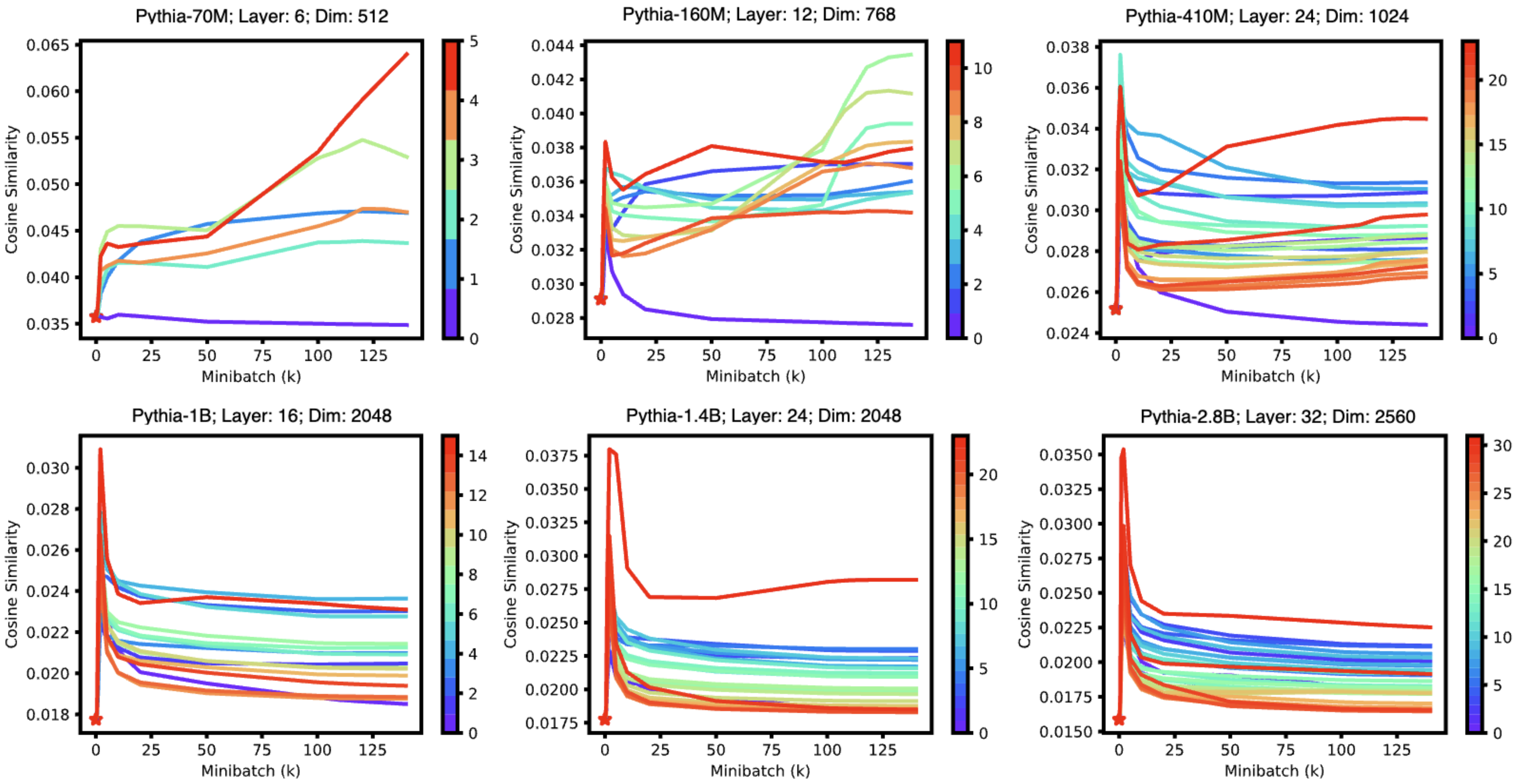
U_C and $\boldsymbol{u}_q$ are embeddings

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

- 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.



JoMA Dynamics

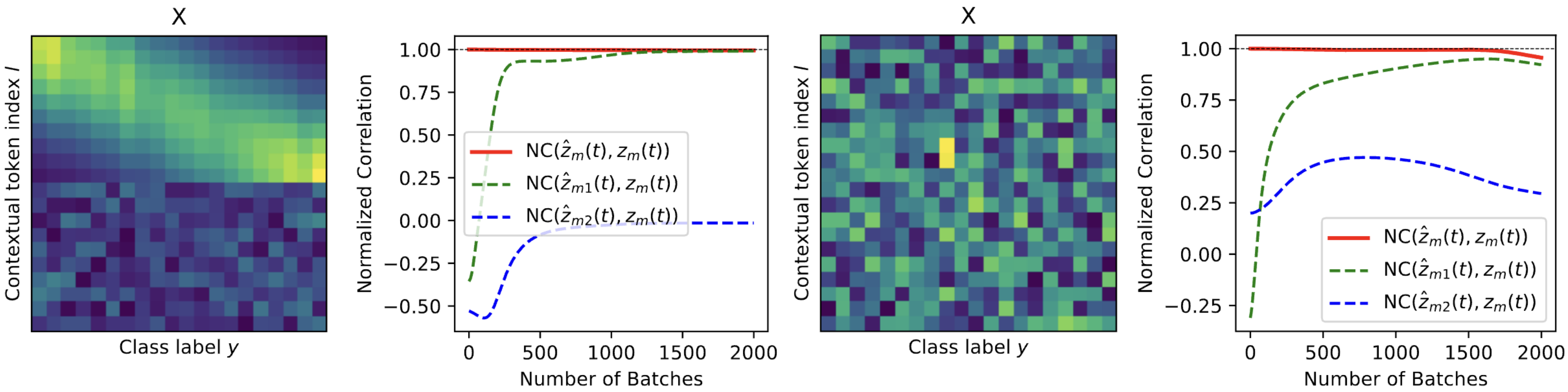
Theorem 1 (JoMA). *Let* $\boldsymbol{v}_k := U_C^\top \boldsymbol{w}_k$, *then the dynamics of Eqn. 3 satisfies the invariants:*

- *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.

Verification of JoMA dynamics



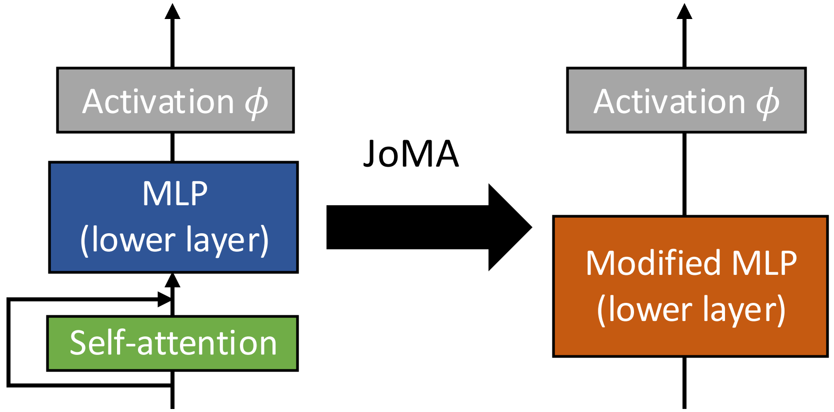
$\boldsymbol{z}_m(t)$: Real attention logits

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

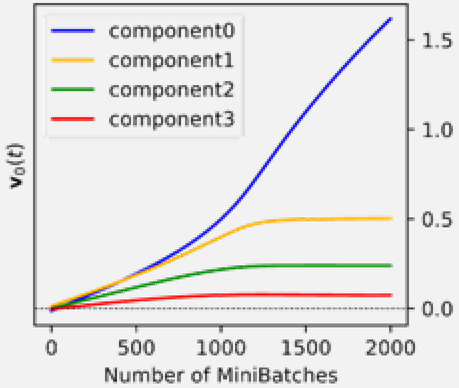
$$\hat{\boldsymbol{z}}_m(t) = \underbrace{\frac{1}{2}\sum_k \boldsymbol{v}_k^2(t)}_{\hat{\boldsymbol{z}}_{m1}(t)} - \underbrace{\|\boldsymbol{v}_k(t)\|_2^2 \bar{\boldsymbol{b}}_m}_{\hat{\boldsymbol{z}}_{m2}(t)} + \boldsymbol{c}$$

Implication of Theorem 1

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

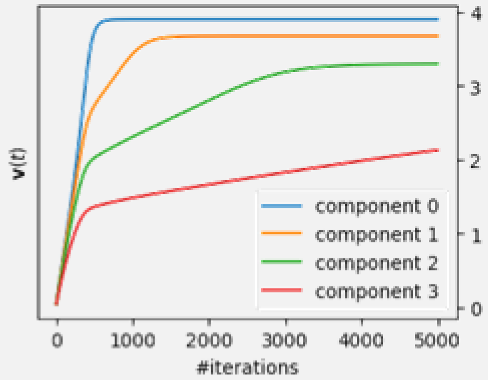


Linear case ($\phi = \mathrm{Id}, K = 1$)



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
Modified MLP (lower layer)

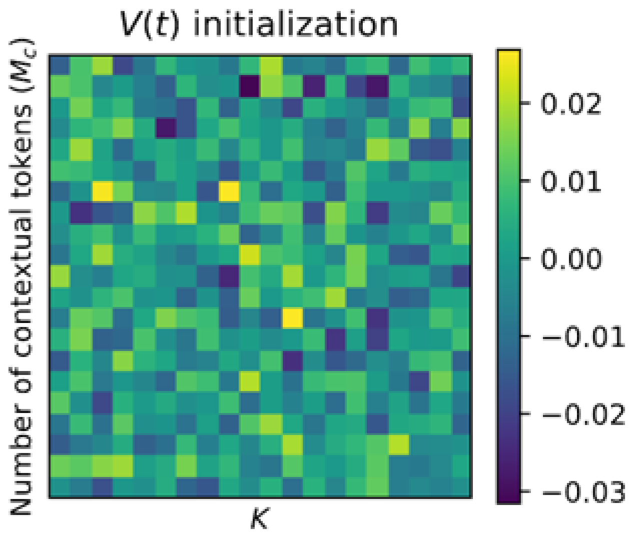
$$\dot{\boldsymbol{v}} = \boldsymbol{\Delta}_m \circ \exp\left(\frac{\boldsymbol{v}^2}{2}\right)$$

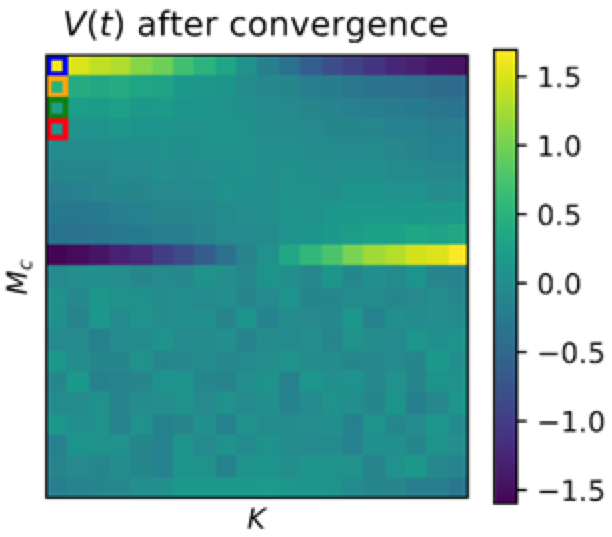
Theorem 2

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

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

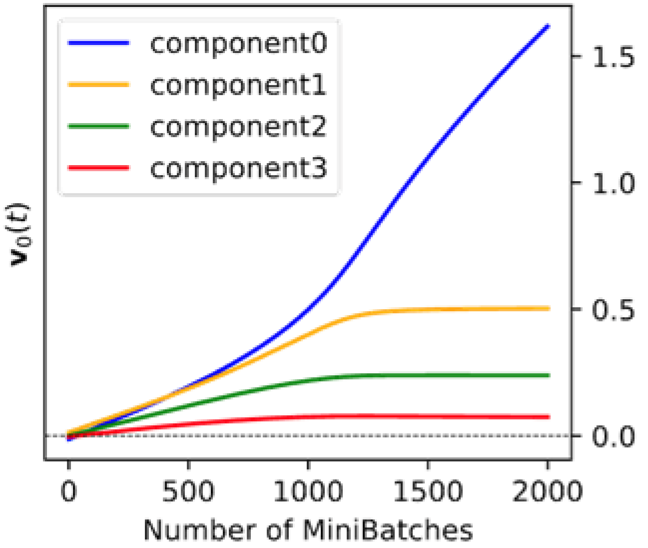
Only the most salient token $l^* = \mathrm{argmax}\,|\Delta_{lm}|$ of $\boldsymbol{v}$ goes to $+\infty$
other components stay finite.





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 ($K > 1$)?

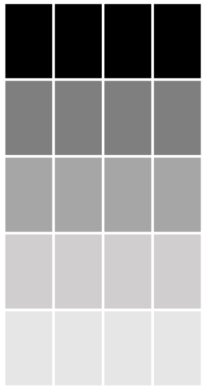
- $V = U_C^\top W \in \mathbb{R}^{M_c \times K}$ and the dynamics becomes

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

We can prove that V gradually becomes low rank

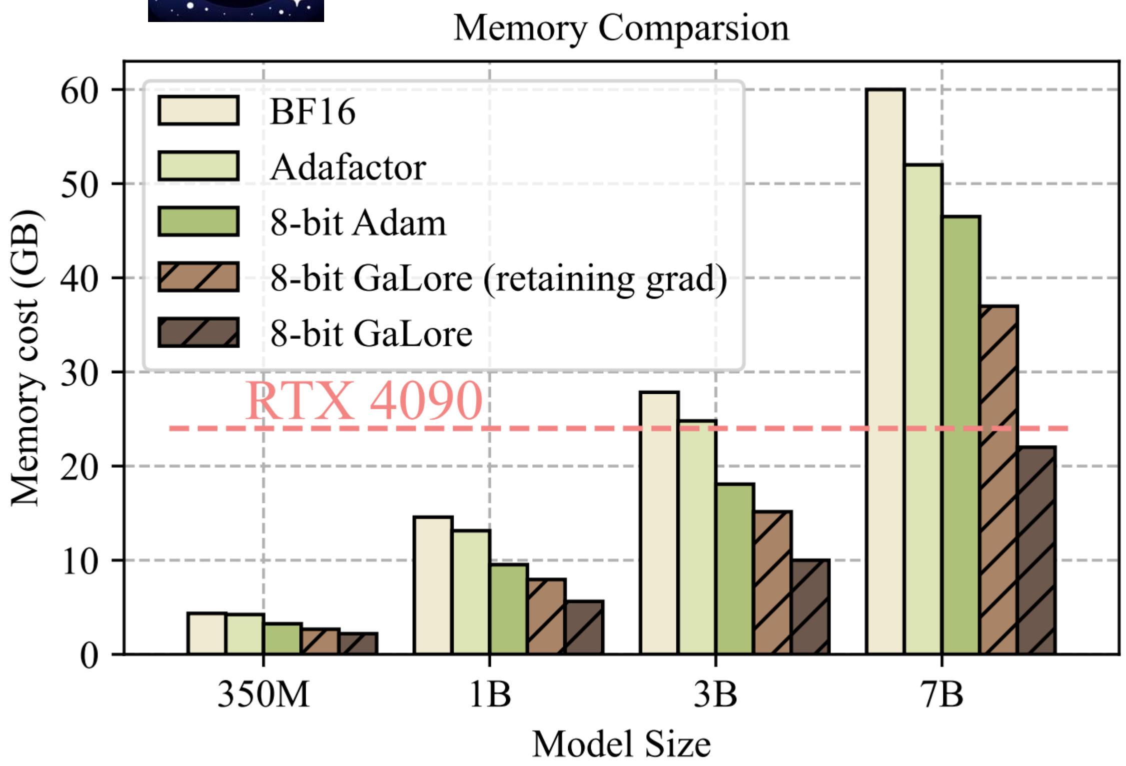
- The growth rate of each row of V varies widely.

$V(t) \rightarrow$



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

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)

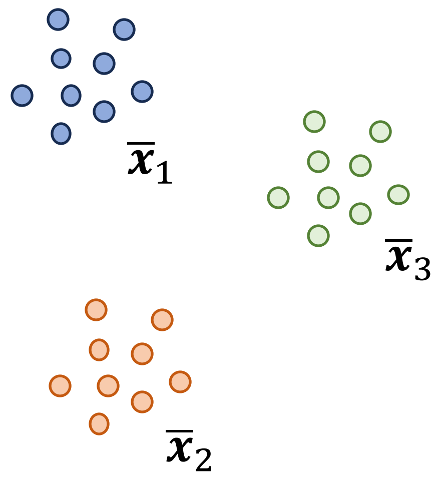
JoMA for Nonlinear Activation

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 $a_c := \mathbb{E}_{q=m,c}[g_{h_k}]\mathbb{P}[c]$, $r_c = \boldsymbol{v}^\top\bar{\boldsymbol{x}}_c + \int_0^t \mathbb{E}_{q=m}[g_{h_k}h_k']\mathrm{d}t$, 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

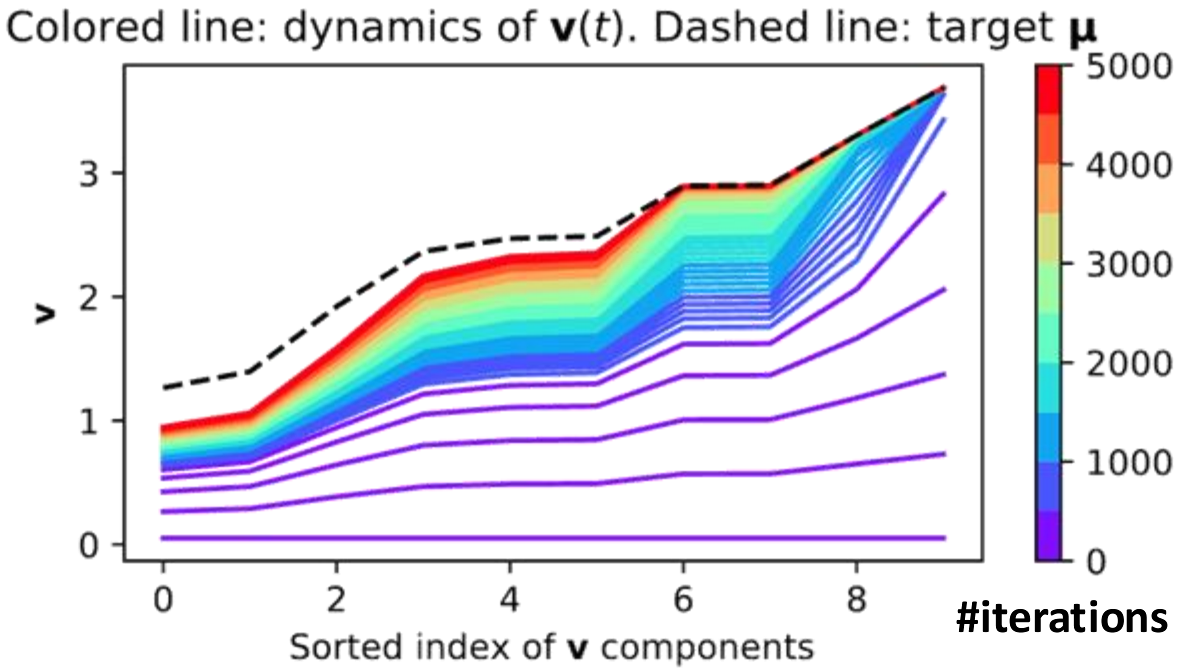
$\dot{\boldsymbol{v}} = (\boldsymbol{\mu} - \boldsymbol{v}) \circ \exp\left(\frac{\boldsymbol{v}^2}{2}\right)$	Nonlinear Modified MLP (lower layer)

Theorem 4

Salient components grow much faster than non-salient ones:

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

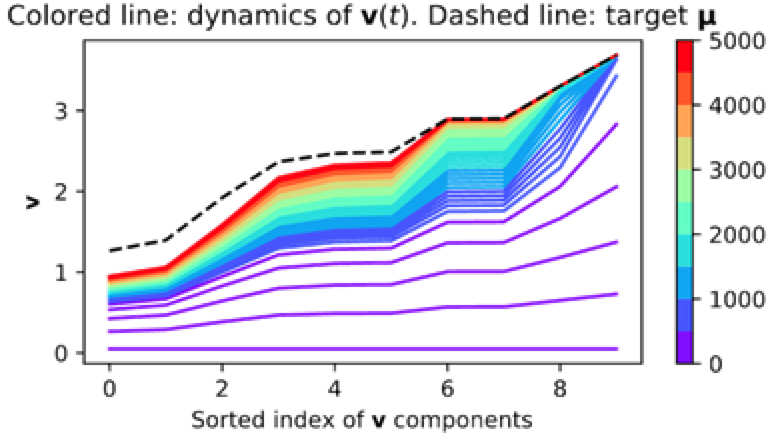
$$\text{ConvergenceRate}(j) \coloneqq \ln 1/\delta_j(t)$$
$$\delta_j(t) \coloneqq 1 - v_j(t)/\mu_j$$

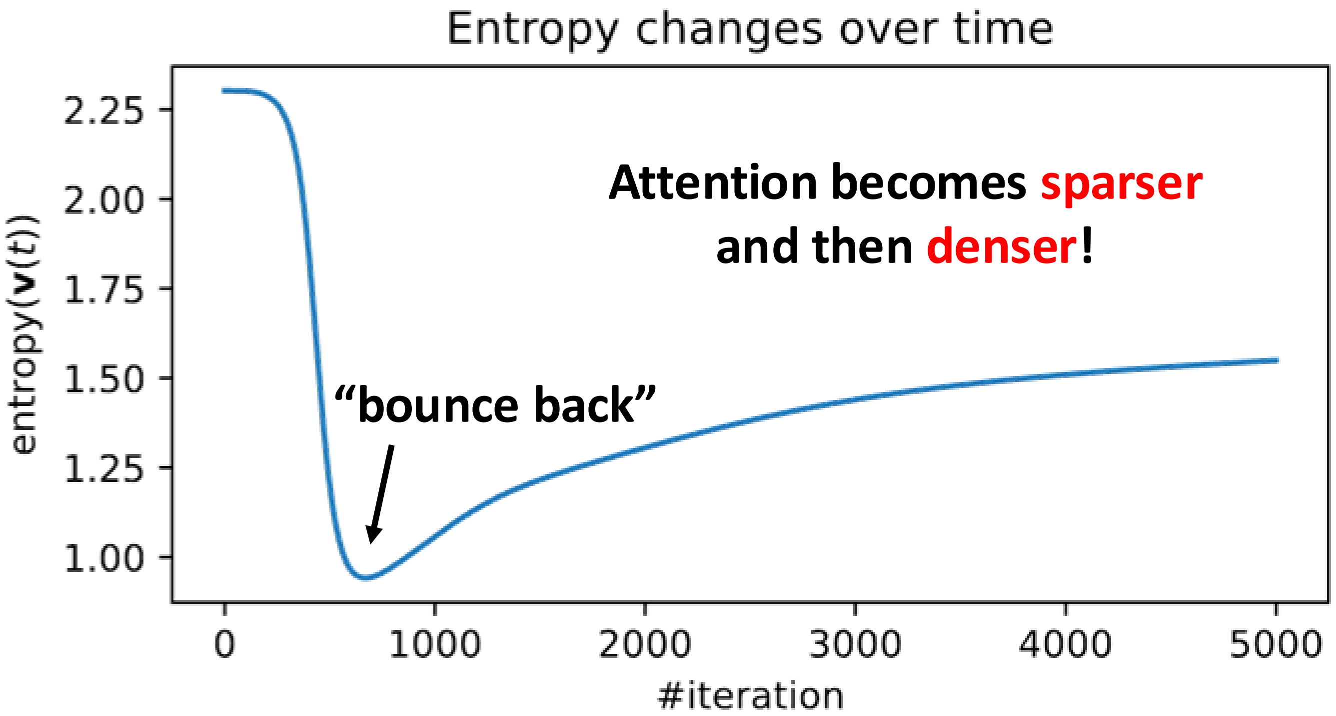


JoMA for Nonlinear activation

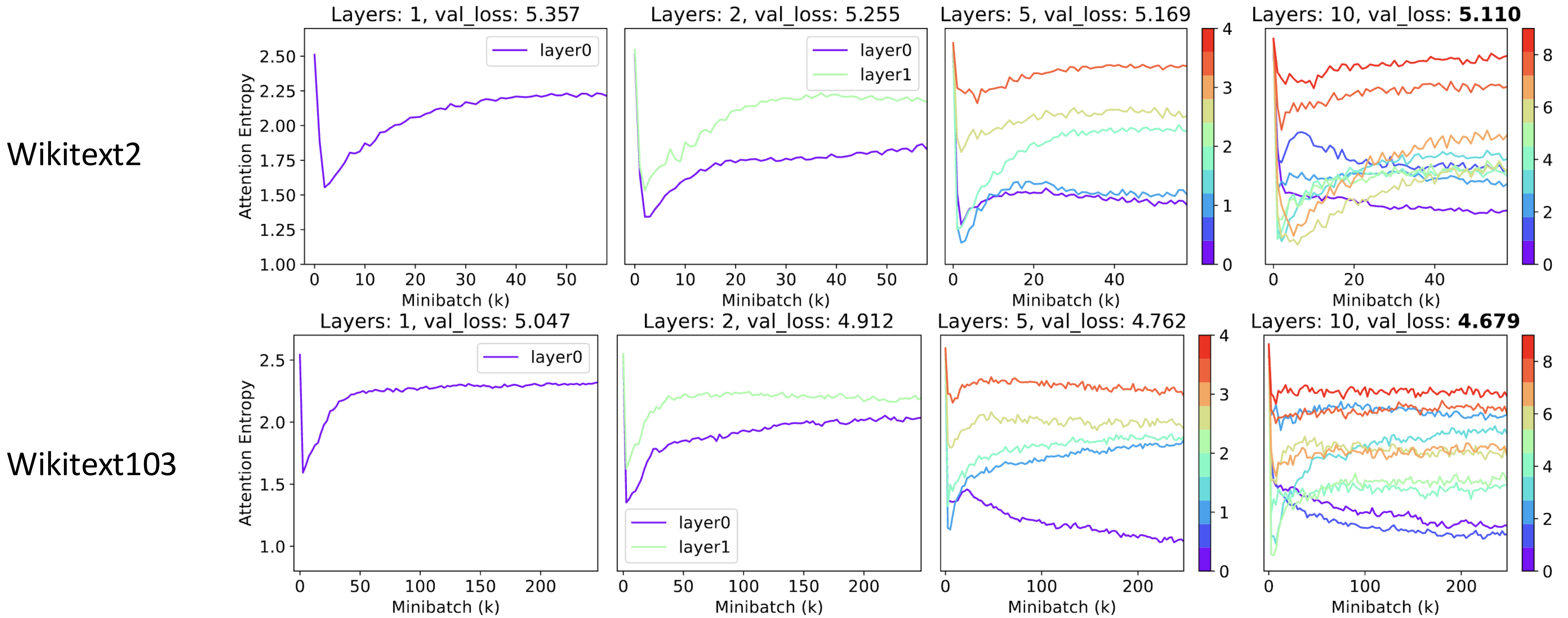
$$\dot{\boldsymbol{v}} = (\boldsymbol{\mu} - \boldsymbol{v}) \circ \exp\left(\frac{\boldsymbol{v}^2}{2}\right)$$

Nonlinear
Modified MLP (lower layer)

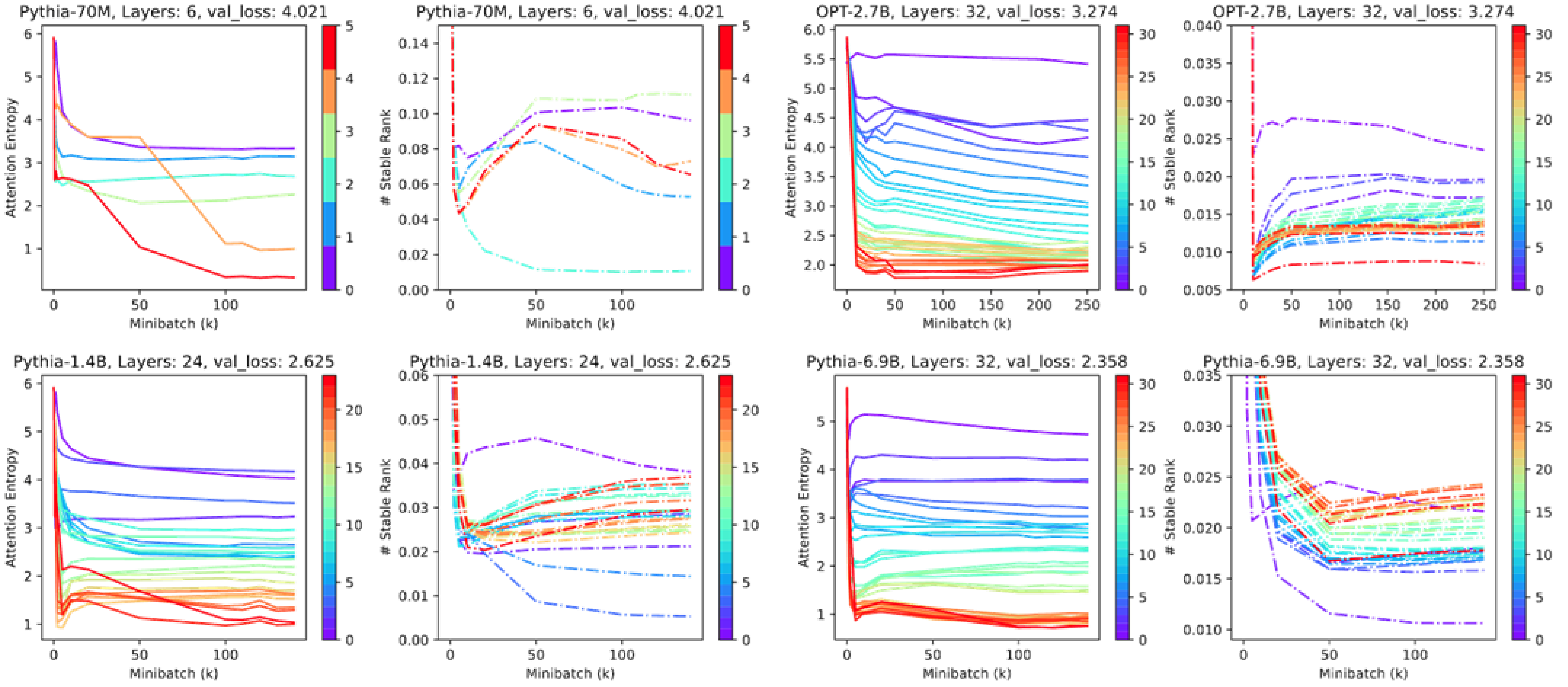




Real-world Experiments



Real-world Experiments



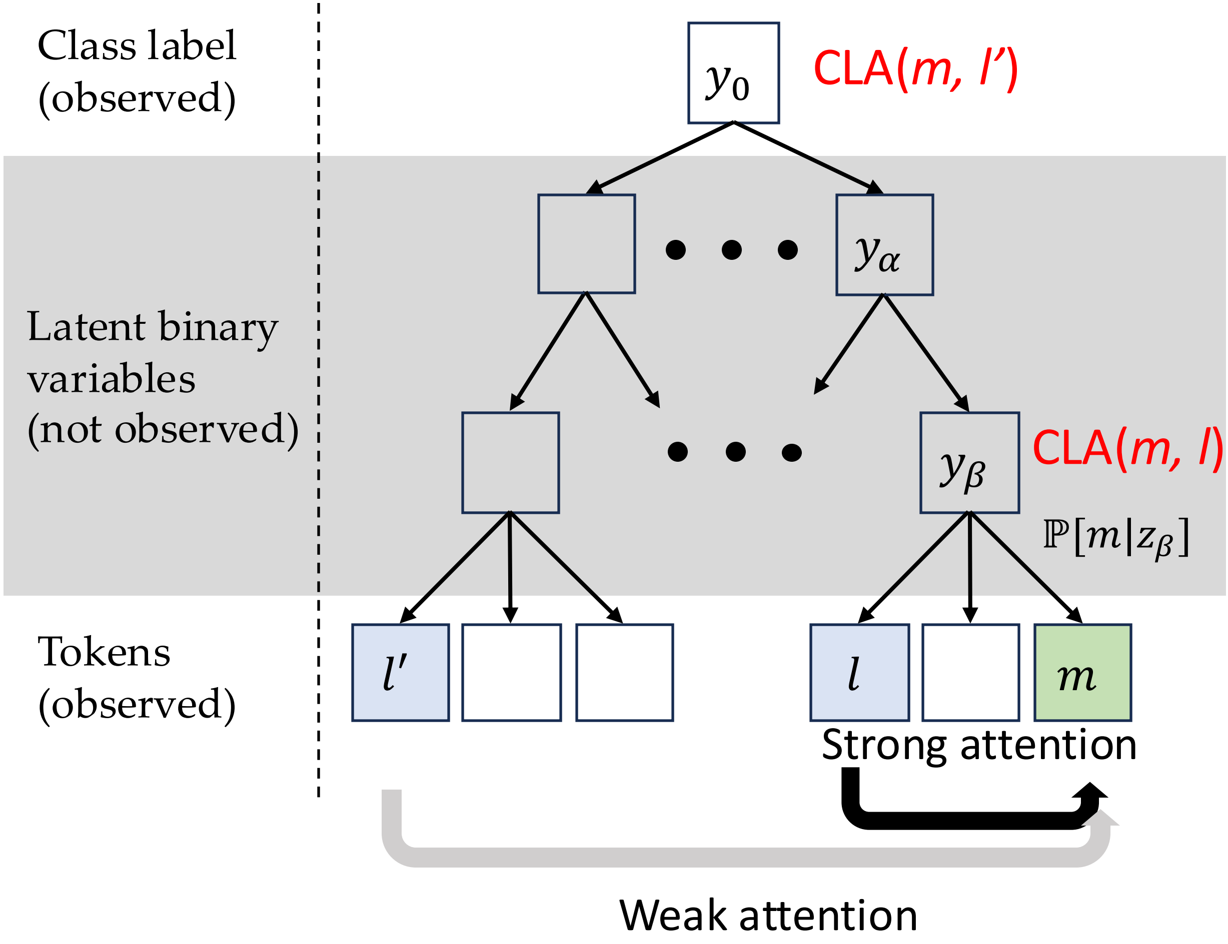
Stable Rank of the lower layer of MLP shows the “bouncing back” effects as well.

Why is this "bouncing back" property useful?

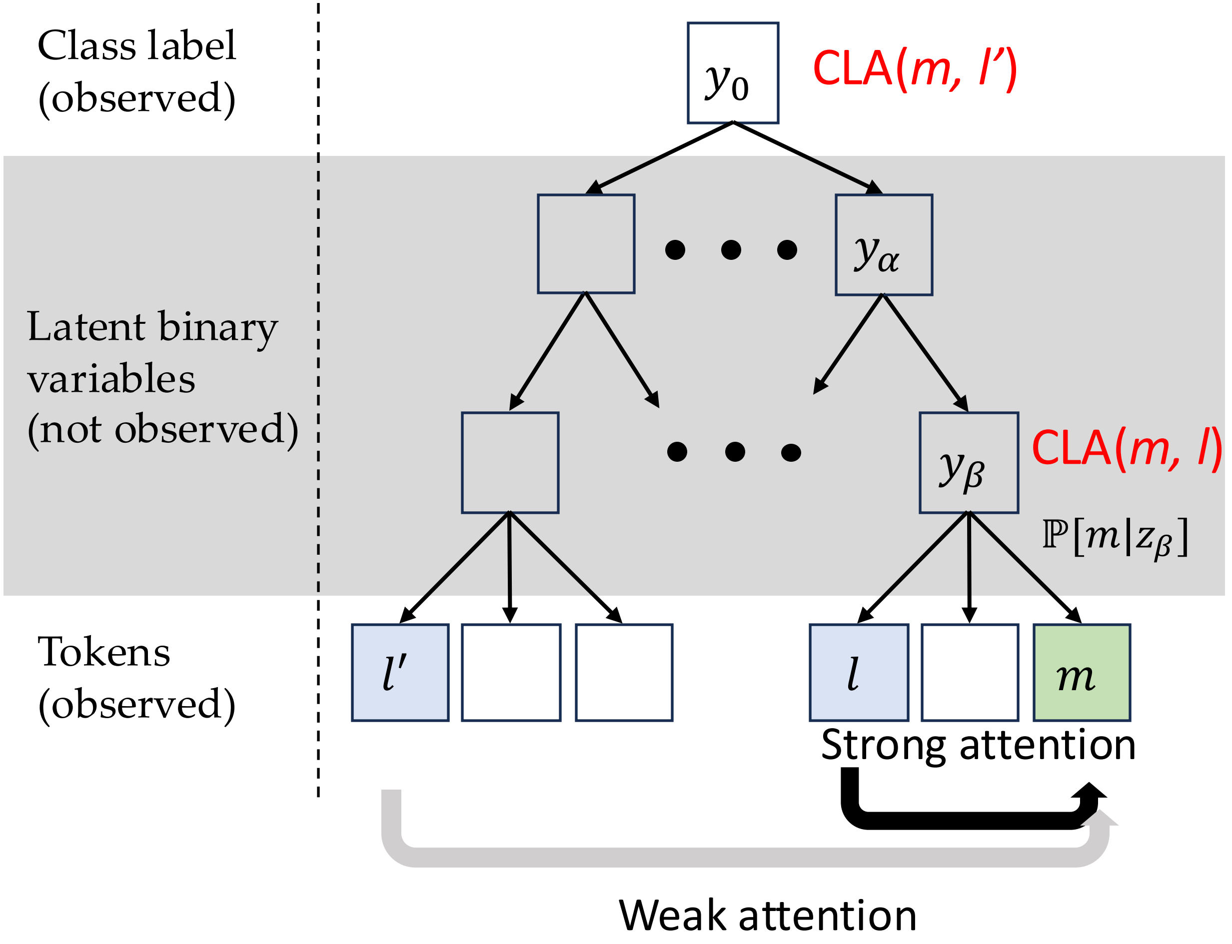
It seems that it only slows down the training??

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

Data Hierarchy & Multilayer Transformer



Data Hierarchy & Multilayer Transformer



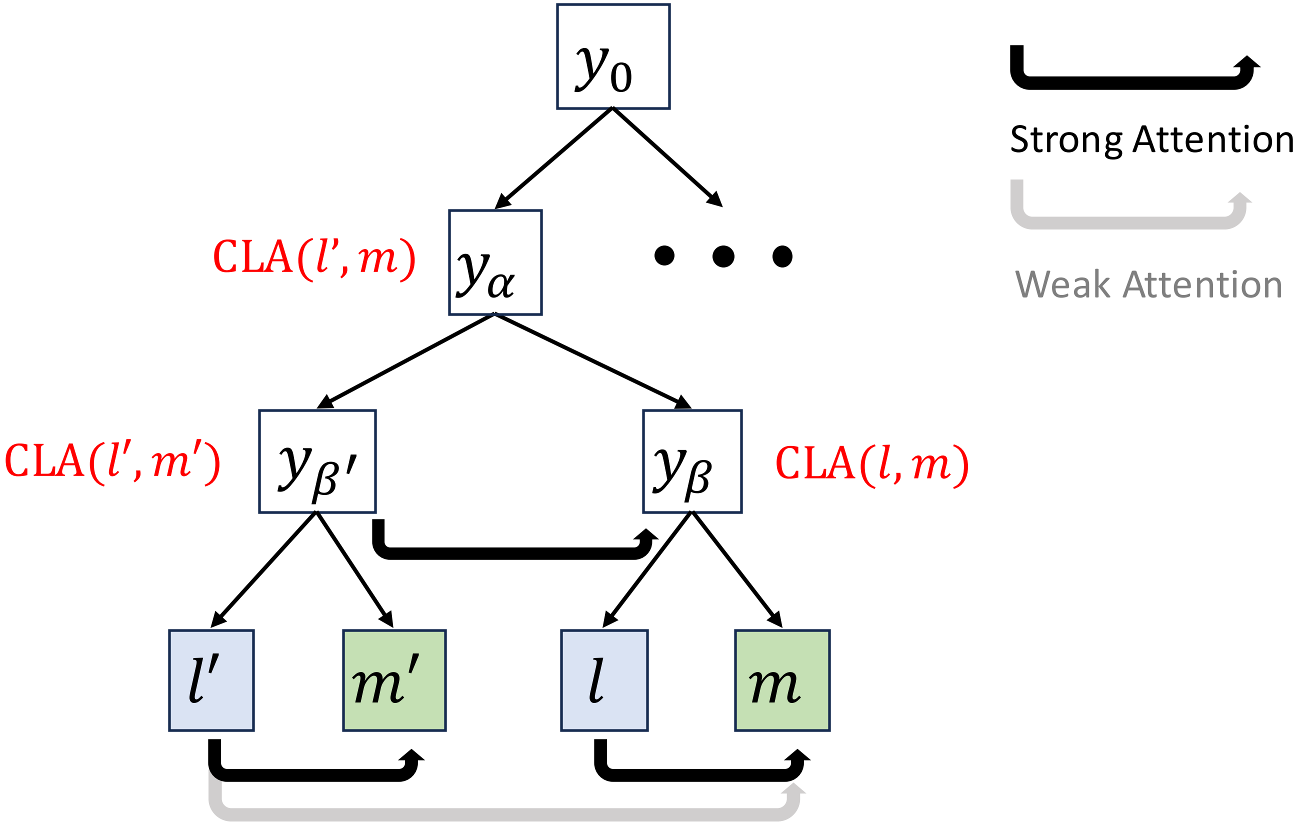
Theorem 5

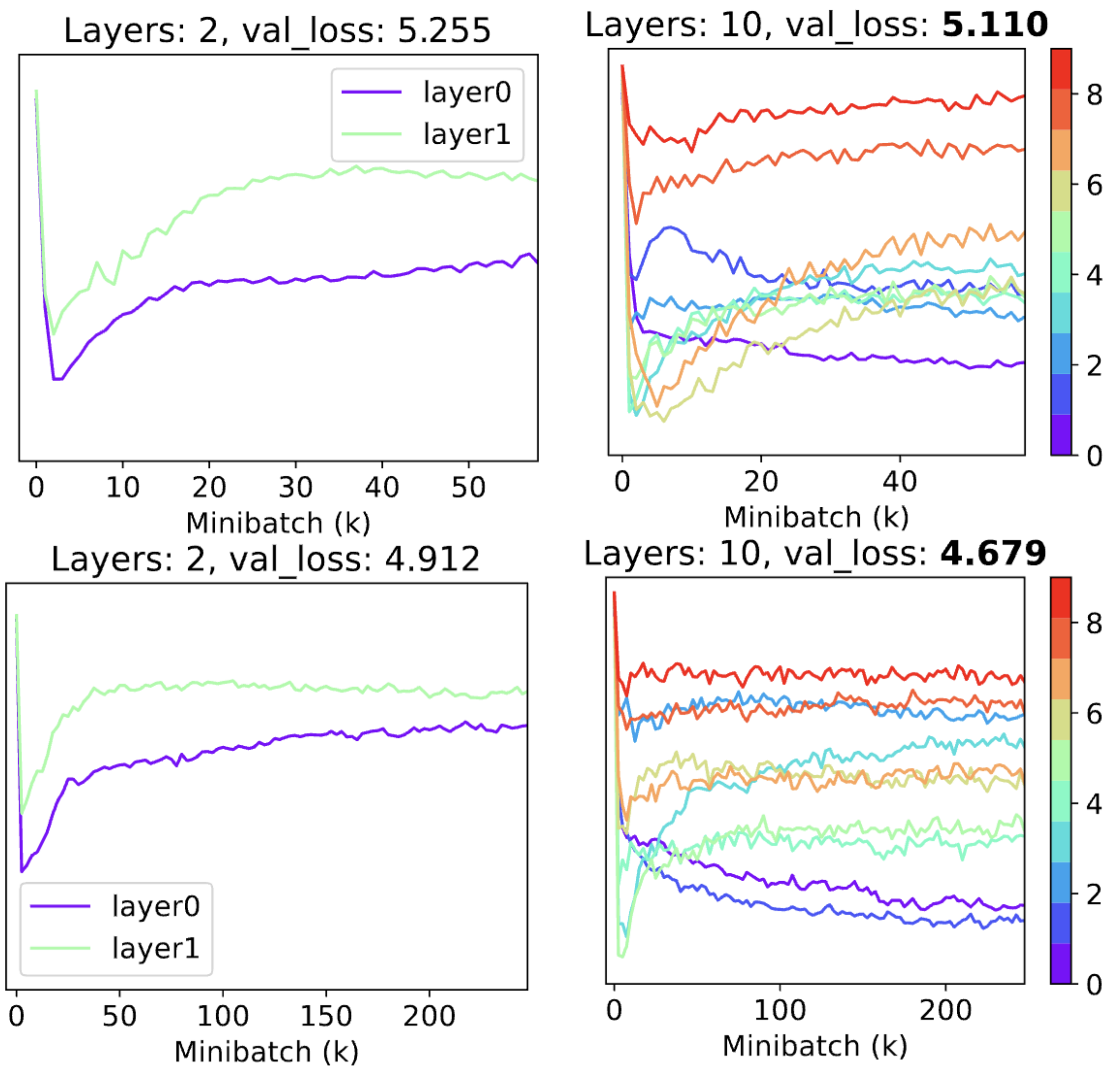
$$\mathbb{P}[l|m] \approx 1 - \frac{H}{L}$$

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

L: total height of the hierarchy

Deep Latent Distribution

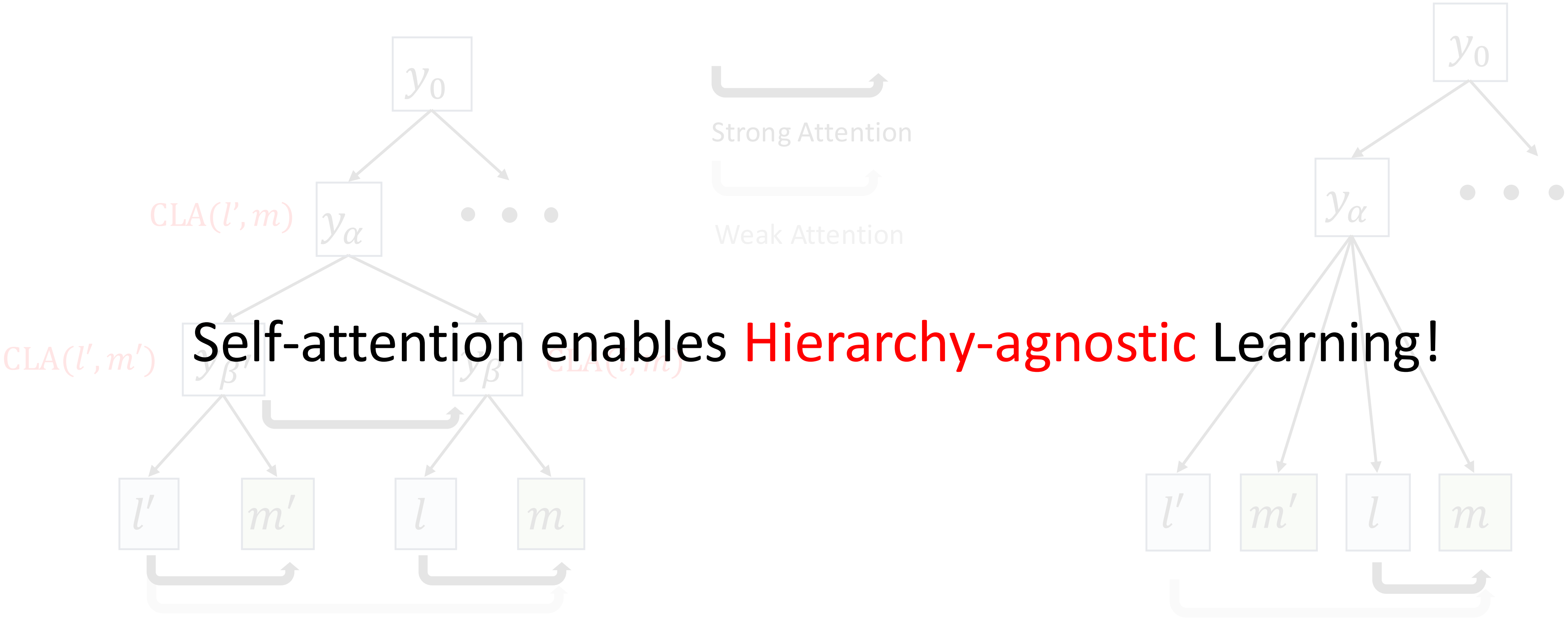




Learning the current hierarchical structure by
slowing down the association of tokens that are not directly correlated

Shallow Latent Distribution

Hierarchy-agnostic Learning



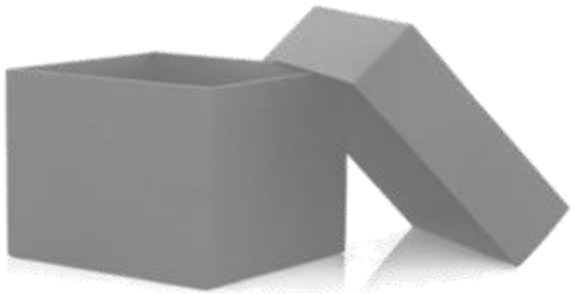
Self-attention enables Hierarchy-agnostic Learning!

Verification of Hierarchical Intuitions

	$C = 20,\ N_{ch} = 2$		$C = 20,\ N_{ch} = 3$		$C = 30,\ N_{ch} = 2$	
(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
	$C = 30\ N_{ch} = 3$		$C = 50,\ N_{ch} = 2$		$C = 50,\ N_{ch} = 3$	
(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

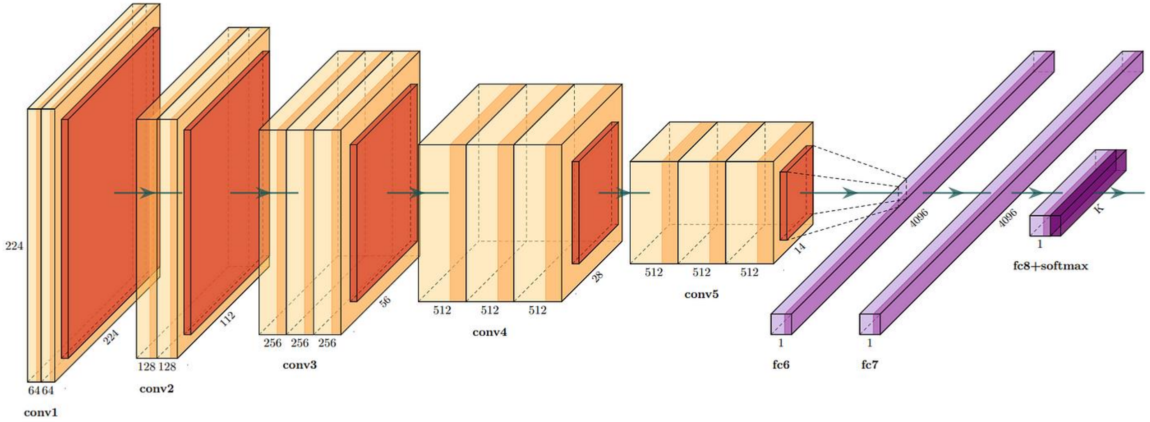
Roadmap of Theoretical Analysis

Fix Representation, check how Self-attention works

Check what representation it learns

Dichotomy: Symbolic and Neural Representation

Neural Representation



Symbolic Representation

$$\nabla \cdot \mathbf{E} = \frac{\rho_v}{\varepsilon} \qquad \text{(Gauss' Law)}$$

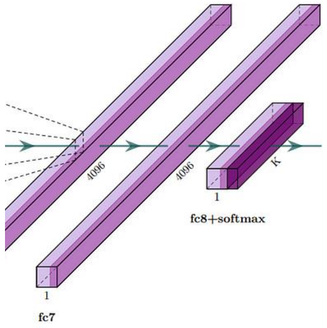
$$\nabla \cdot \mathbf{H} = 0 \qquad \text{(Gauss' Law for Magnetism)}$$

$$\nabla \times \mathbf{E} = -\mu \frac{\partial \mathbf{H}}{\partial t} \qquad \text{(Faraday's Law)}$$

$$\nabla \times \mathbf{H} = \mathbf{J} + \varepsilon \frac{\partial \mathbf{E}}{\partial t} \qquad \text{(Ampere's Law)}$$

Unification of Symbolic and Neural Representation

Emerging Symbolic Structure



Neural
Repres

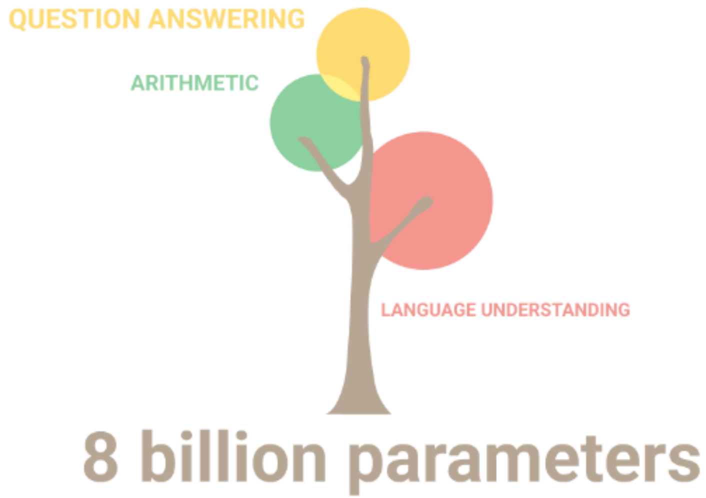
tism)

Symbo
Repres

Deep Models

pere' s Law)

Debate: Is LLM doing retrieval or true reasoning?



LLM shows emergent behaviors!!

https://medium.com/@fenjiro/large-language-models-llms-emergent-abilities-chatgpt-talks-moroccan-dialect-as-an-example-c945f93aa63a

Debate: Is LLM doing retrieval or true reasoning?

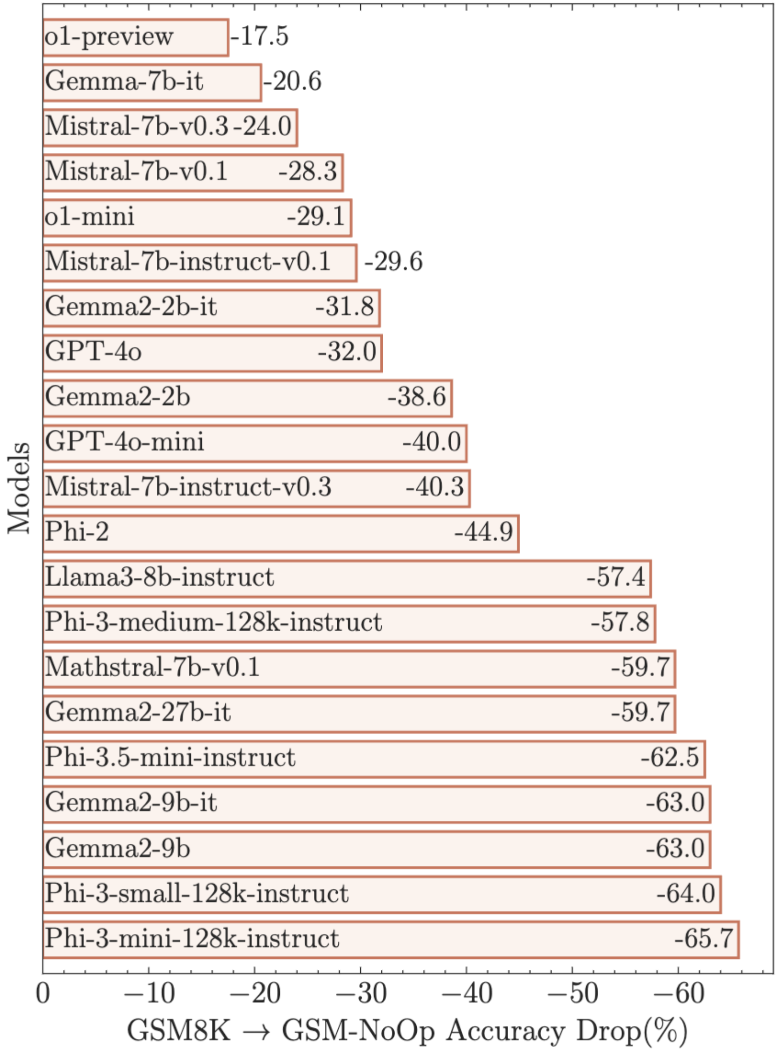
Yann LeCun @ylecun

Do LLMs perform reasoning or approximate retrieval?
There is a continuum between the two, and Auto-Regressive LLMs are largely on the retrieval side.

Subbarao Kambhampati (కంభంపాటి సుబ్బారావు) @rao2z

Emergent Abilities (noun): The preferred euphemism for what your LLM does, when saying "approximate retrieval" sounds too unsexy.

#AIAphorisms



LLM is just doing retrievals!!

Concrete Example: **Modular Addition**

$$a + b = c \bmod d$$

Does neural network have an *implicit table* to do retrieval?

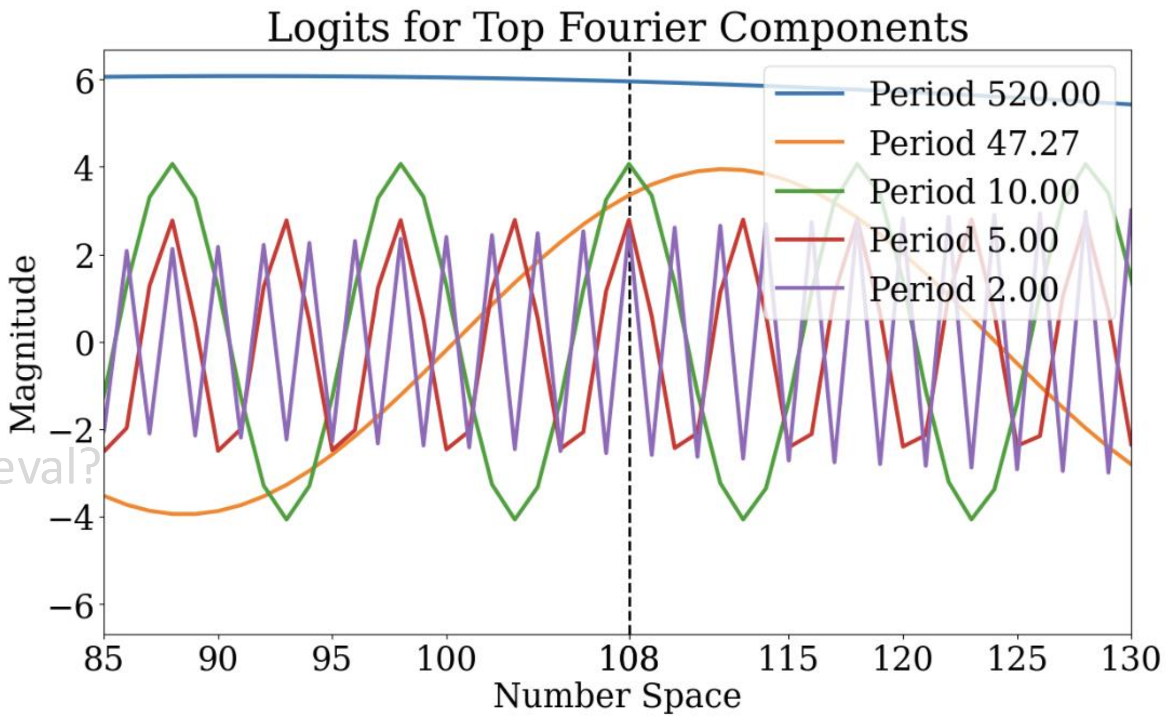
Concrete Example: **Modular Addition**

$$a + b = c \bmod d$$

Does neural network have an *implicit table* to do retrieval?

Learned representation = Fourier basis 🤯

Why? 🤔

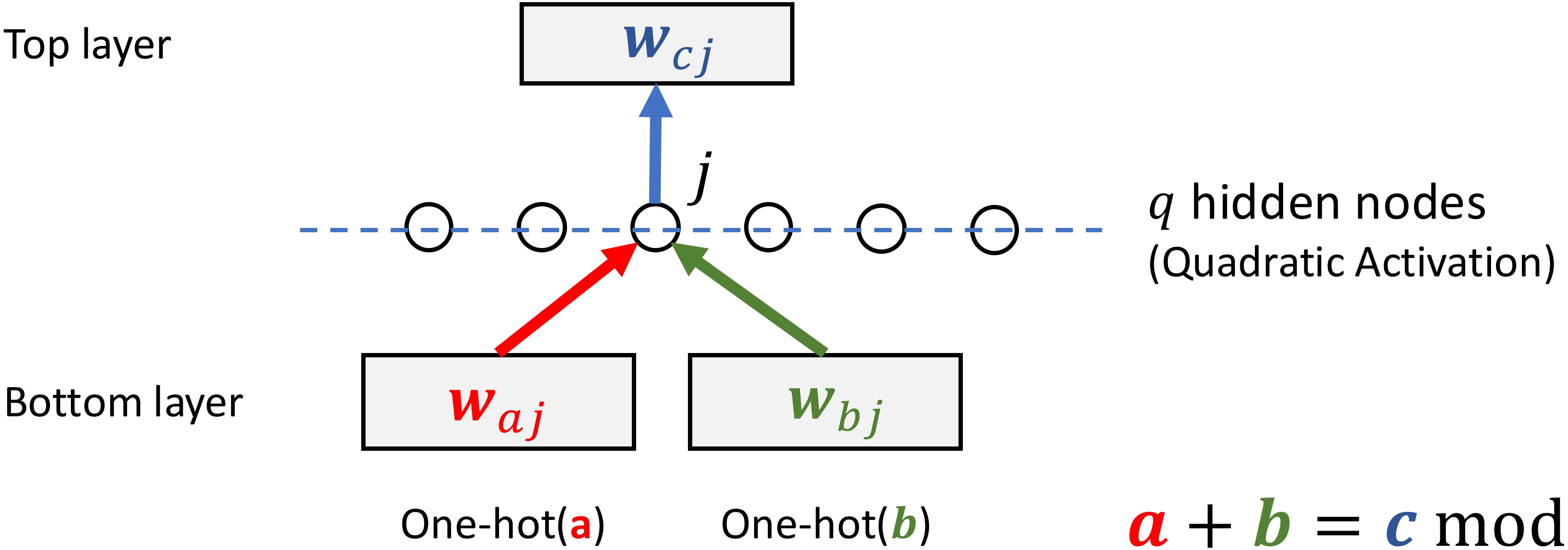


(a) Final logits for top Fourier components

[T. Zhou et al, *Pre-trained Large Language Models Use Fourier Features to Compute Addition*, NeurIPS'24]
[S. Kantamneni, *Language Models Use Trigonometry to Do Addition*, arXiv'25]

Problem Setup

MSE Loss: $Min \ \|\text{Output} - \text{one-hot}(\boldsymbol{c})\|_2$



$$\boldsymbol{a} + \boldsymbol{b} = \boldsymbol{c} \bmod d$$

[Y. Tian, *Composing Global Optimizers to Reasoning Tasks via Algebraic Objects in Neural Nets*, arXiv'24]

(Scaled) Fourier Transform

$$z_{akj} = \sum_{m=0}^{d-1} w_{amj} e^{imk/d}$$

$$z_{bkj} = \sum_{m=0}^{d-1} w_{bmj} e^{imk/d}$$

$$z_{ckj} = \sum_{m=0}^{d-1} w_{cmj} e^{imk/d}$$

k: frequency

$\{W_a, W_b, W_c\}$ are real

⬇

Hermitian condition holds

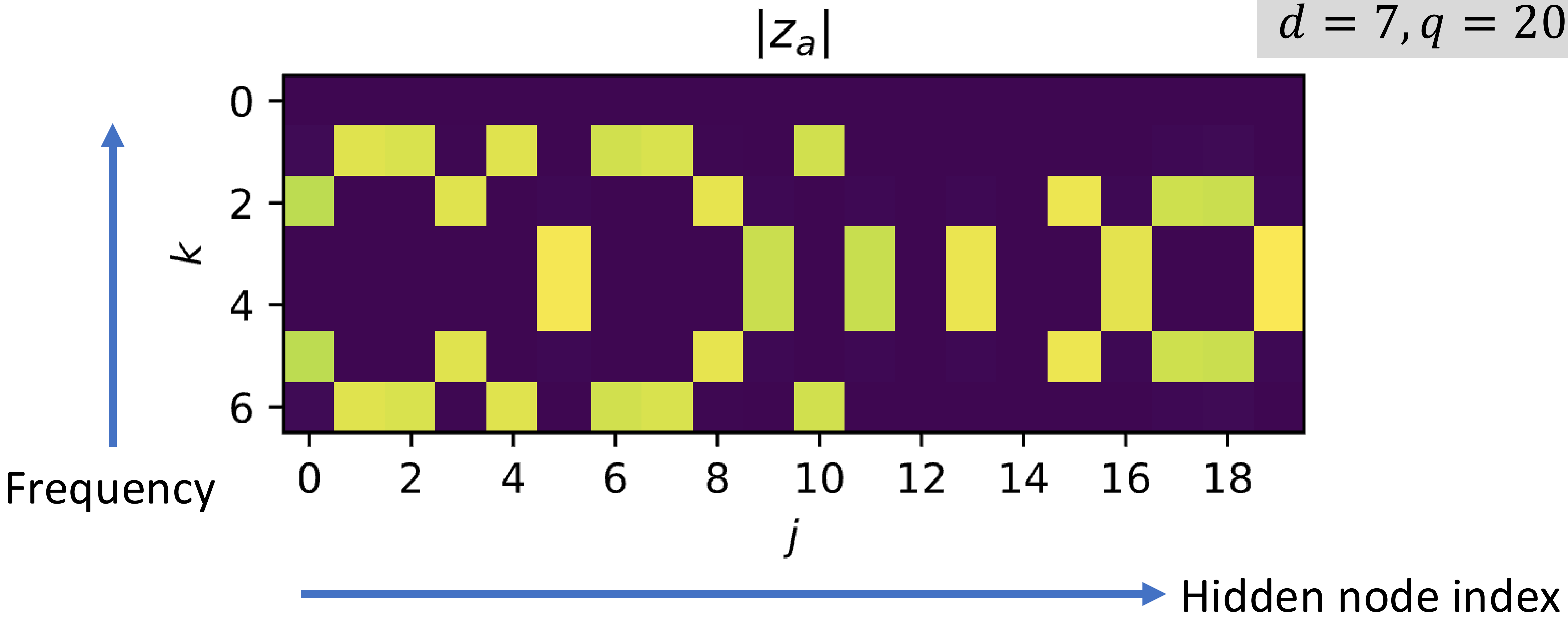
$$z_{akj} = \overline{z_{a,-k,j}}$$

$$z_{bkj} = \overline{z_{b,-k,j}}$$

$$z_{ckj} = \overline{z_{c,-k,j}}$$

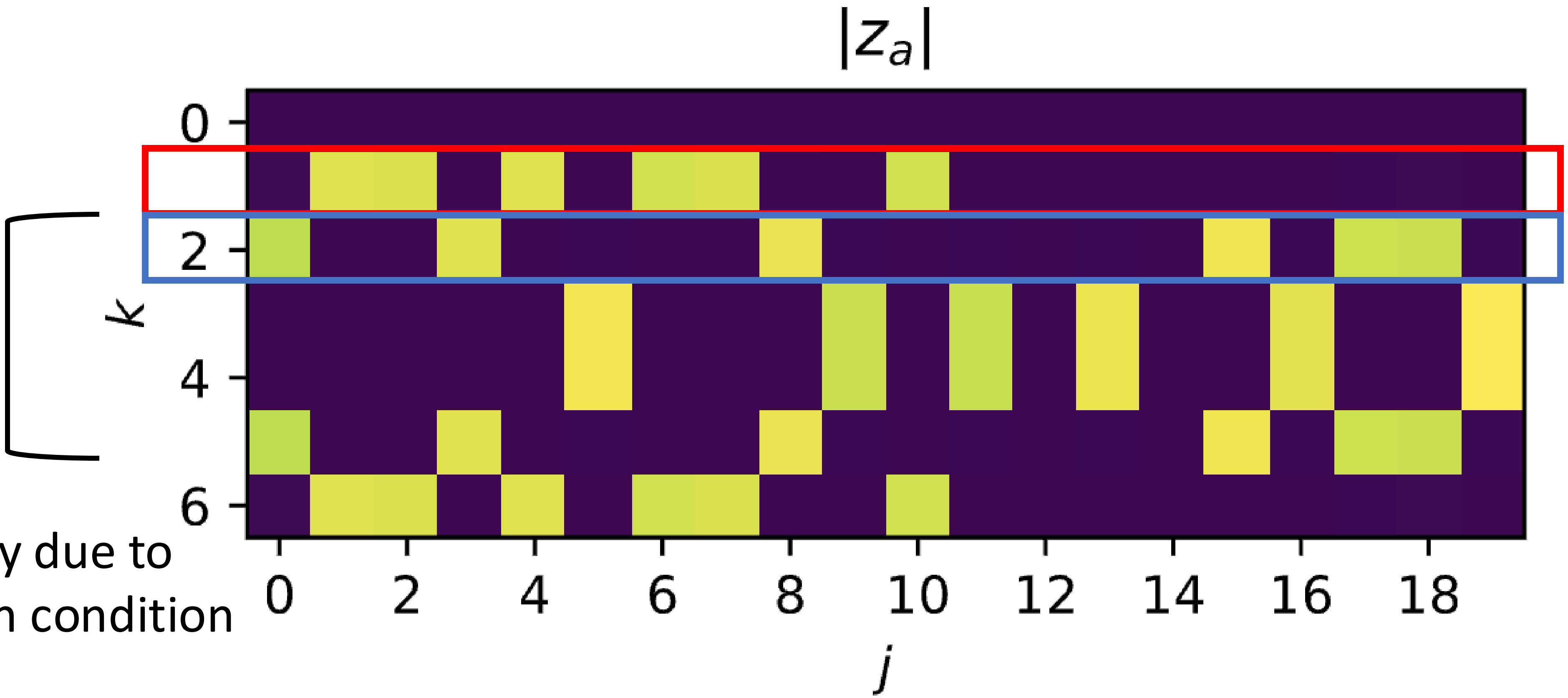
What a Gradient Descent Solution look like?

$d = 7, q = 20$



[Y. Tian, *Composing Global Optimizers to Reasoning Tasks via Algebraic Objects in Neural Nets*, arXiv'24]

What a Gradient Descent Solution look like?

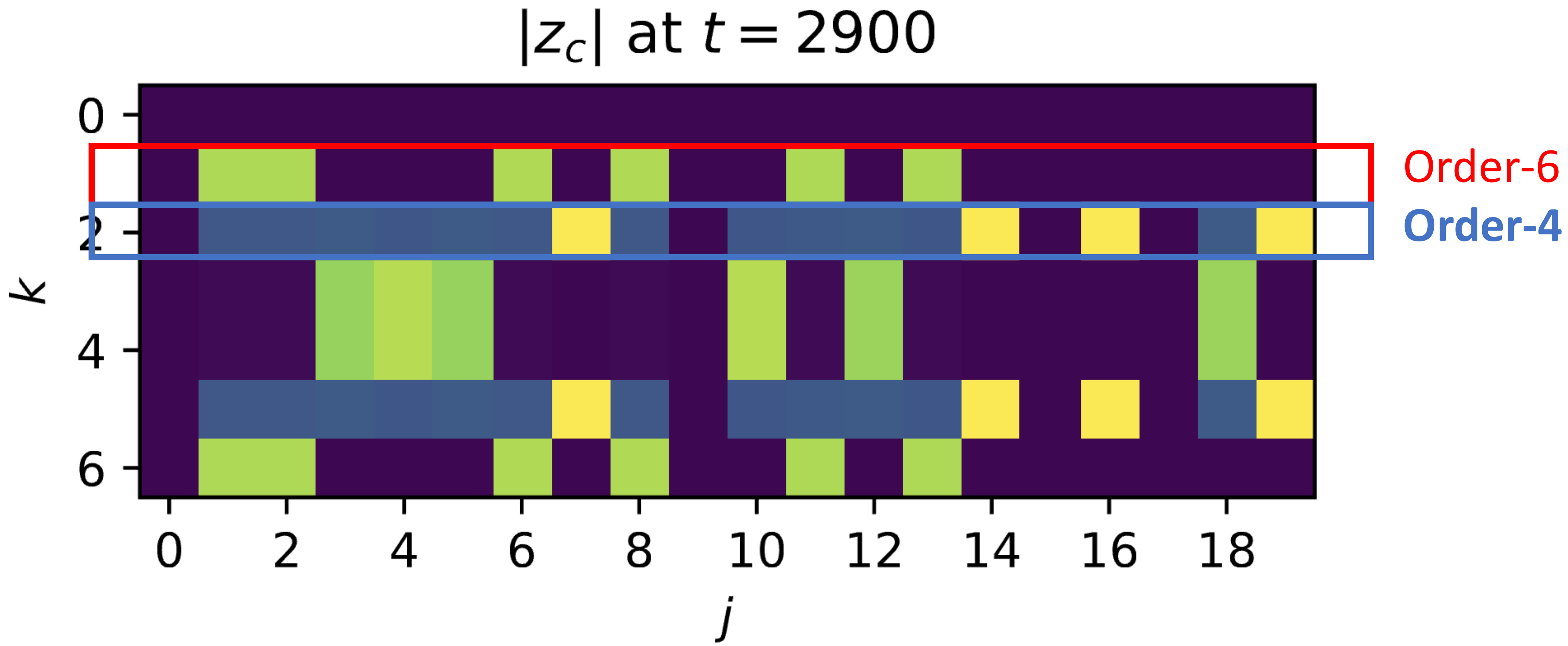


Order-6 solutions

Symmetry due to Hermitian condition

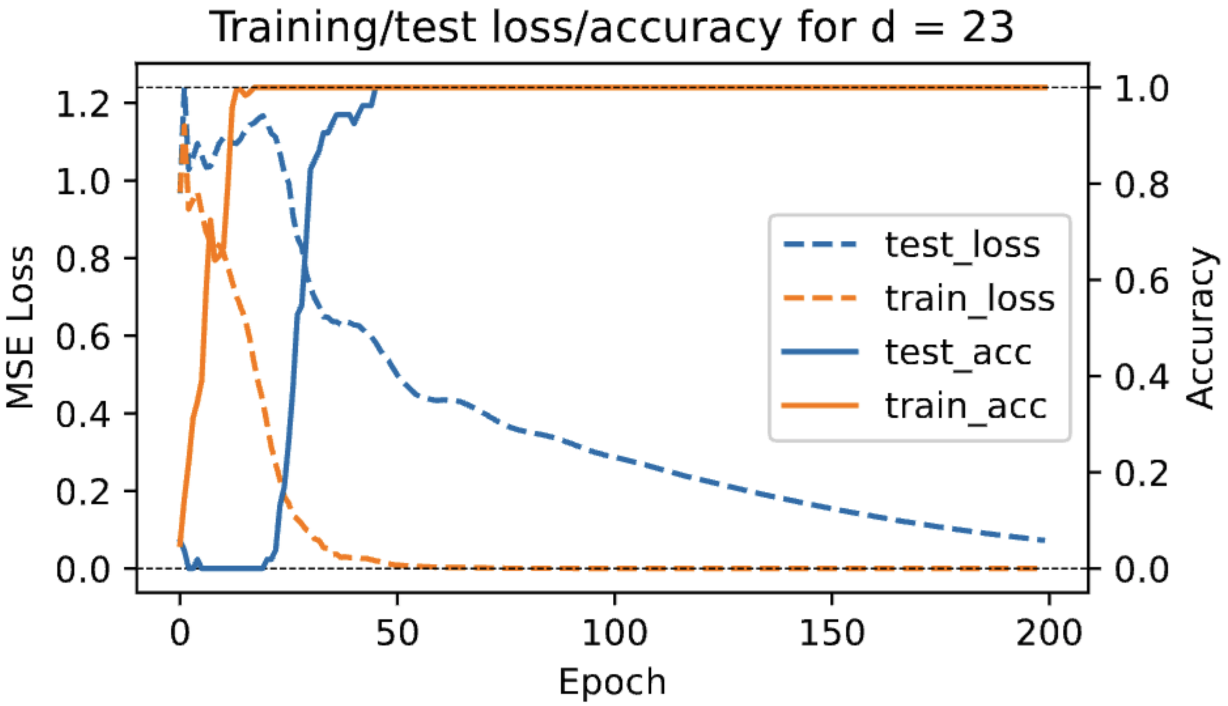
[Y. Tian, *Composing Global Optimizers to Reasoning Tasks via Algebraic Objects in Neural Nets*, arXiv'24]

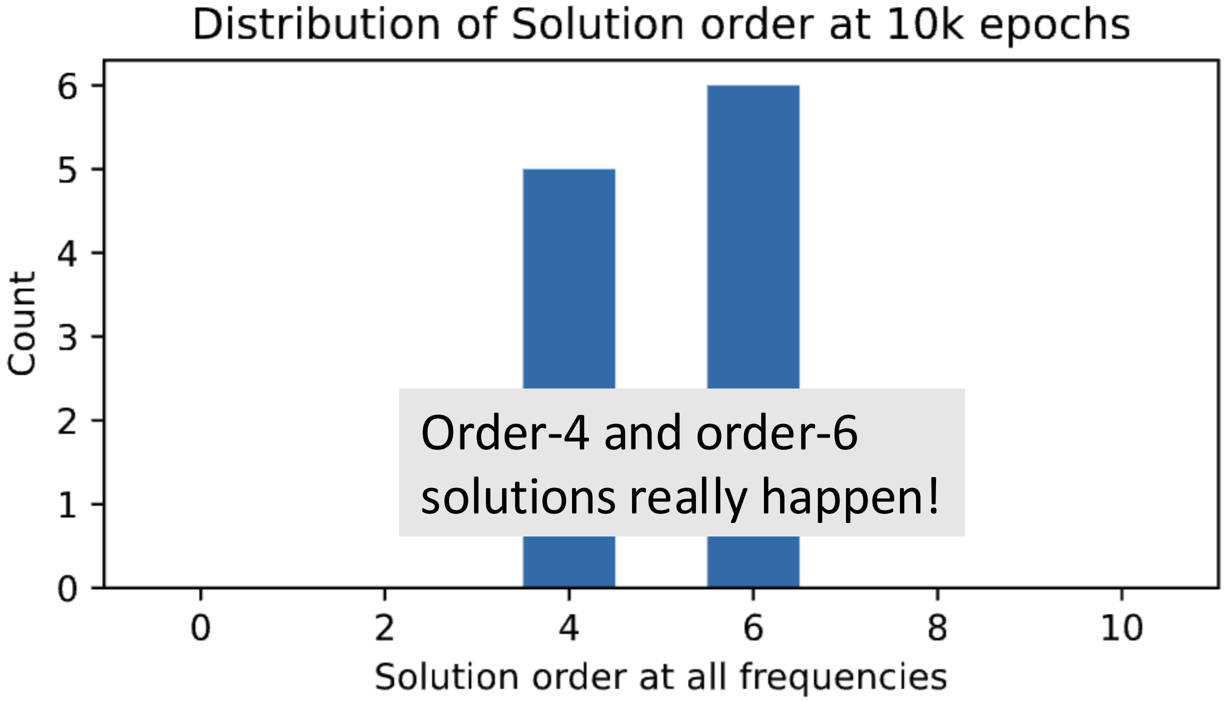
What a Gradient Descent Solution look like?



[Y. Tian, *Composing Global Optimizers to Reasoning Tasks via Algebraic Objects in Neural Nets*, arXiv'24]

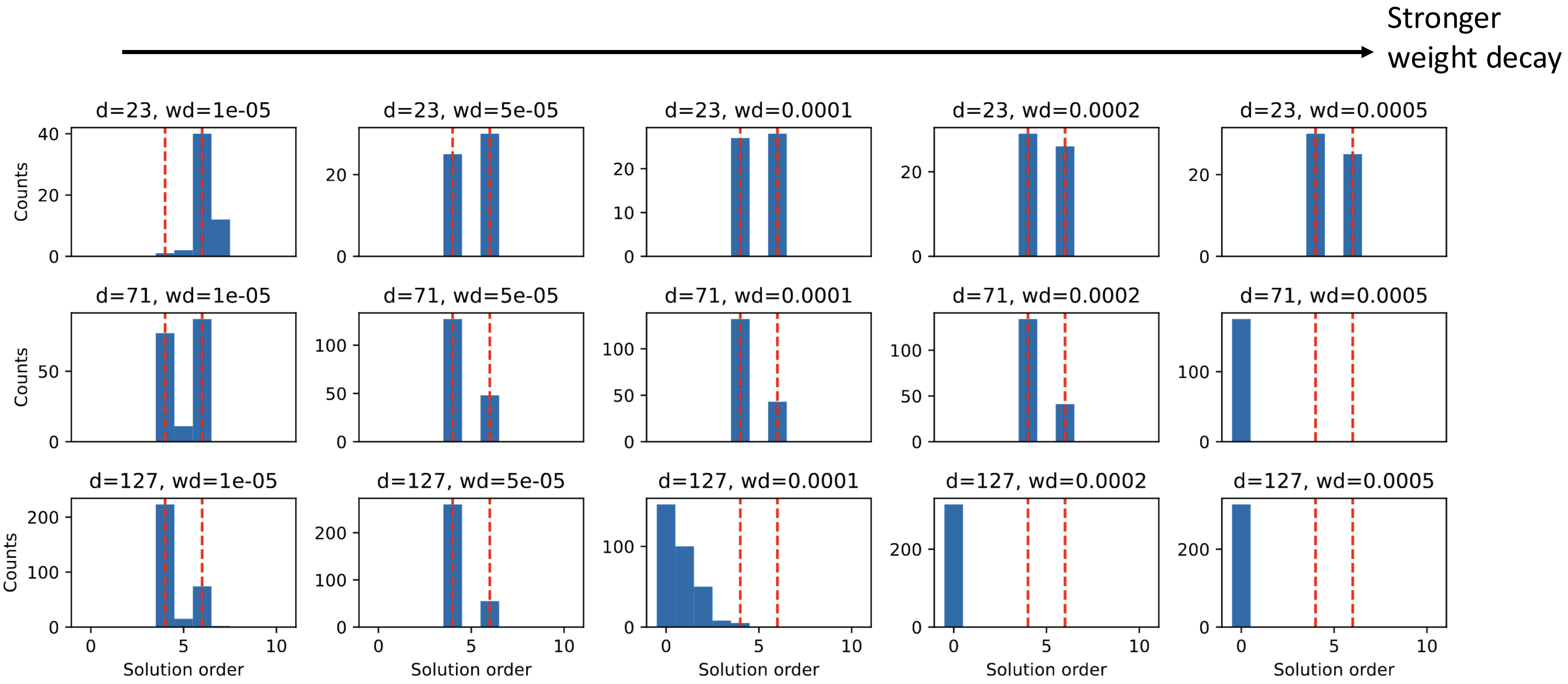
More Statistics on Gradient Descent Solutions





[Y. Tian, *Composing Global Optimizers to Reasoning Tasks via Algebraic Objects in Neural Nets*, arXiv'24]

Effect of Weight Decay



[Y. Tian, *Composing Global Optimizers to Reasoning Tasks via Algebraic Objects in Neural Nets*, arXiv'24]

Why? 🤔

Structure of Loss Functions

MSE loss $\ell(\boldsymbol{z}) = d^{-1} \sum_{k\neq 0} \ell_k(\boldsymbol{z}) + 1 - 1/d$

$$\ell_k(\boldsymbol{z}) = -2r_{kkk} + \sum_{k_1 k_2} \left|r_{k_1 k_2 k}\right|^2 + \frac{1}{4}\left|\sum_{p\in\{a,b\}} \sum_{k'} r_{p,k',-k',k}\right|^2 + \frac{1}{4}\sum_{m\neq 0} \sum_{p\in\{a,b\}} \left|\sum_{k'} r_{p,k',m-k',k}\right|^2$$

Term $r_{k_1 k_2 k}(\boldsymbol{z}) := \sum_j z_{ak_1 j} z_{bk_2 j} z_{ckj}$ and $r_{pk_1 k_2 k}(\boldsymbol{z}) := \sum_j z_{pk_1 j} z_{pk_2 j} z_{ckj}$

[Y. Tian, *Composing Global Optimizers to Reasoning Tasks via Algebraic Objects in Neural Nets*, arXiv'24]

Structure of Loss Functions

MSE loss $\ell(\boldsymbol{z}) = d^{-1} \sum_{k\neq 0} \ell_k(\boldsymbol{z}) + 1 - 1/d$

$$\ell_k(\boldsymbol{z}) = -2r_{kkk} + \sum_{k_1 k_2} \left|r_{k_1 k_2 k}\right|^2 + \frac{1}{4}\left|\sum_{p\in\{a,b\}} \sum_{k'} r_{p,k',-k',k}\right|^2 + \frac{1}{4}\sum_{m\neq 0} \sum_{p\in\{a,b\}} \left|\sum_{k'} r_{p,k',m-k',k}\right|^2$$

Term $r_{k_1 k_2 k}(\boldsymbol{z}) := \sum_j z_{ak_1 j} z_{bk_2 j} z_{ckj}$ and $r_{pk_1 k_2 k}(\boldsymbol{z}) := \sum_j z_{pk_1 j} z_{pk_2 j} z_{ckj}$

Sufficient conditions of Global Optimizers:

R_{g}	R_{c}	R_{n}	R_*
$r_{kkk} = 1$	$r_{k_1 k_2 k} = 0$	$r_{pk',-k',k} = 0$	$r_{pk',m-k',k} = 0$

[Y. Tian, *Composing Global Optimizers to Reasoning Tasks via Algebraic Objects in Neural Nets*, arXiv'24]

How to Optimize?

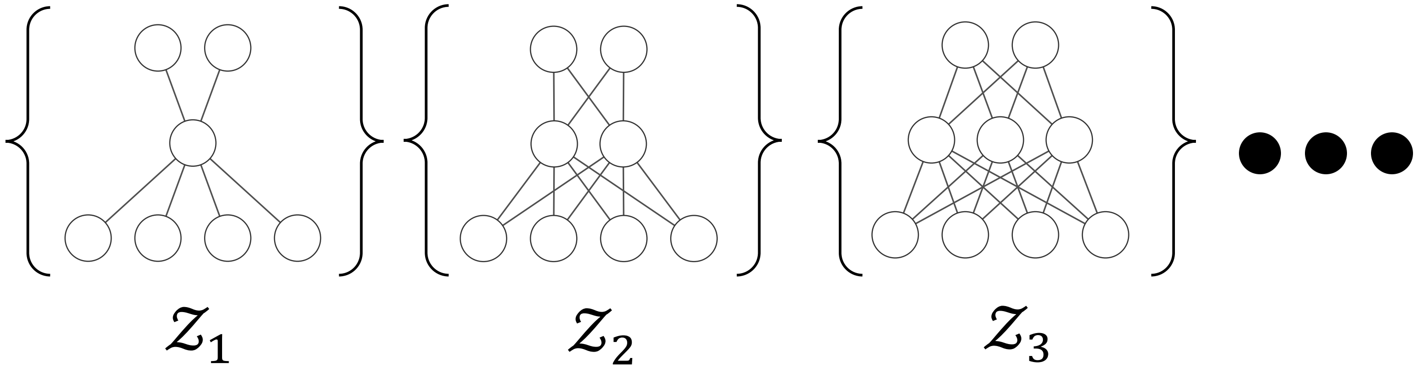
The objective is highly nonlinear !!

However, nice *algebraic structures* exist!

How to Optimize?

The objective is highly nonlinear !!

However, nice *algebraic structures* exist!

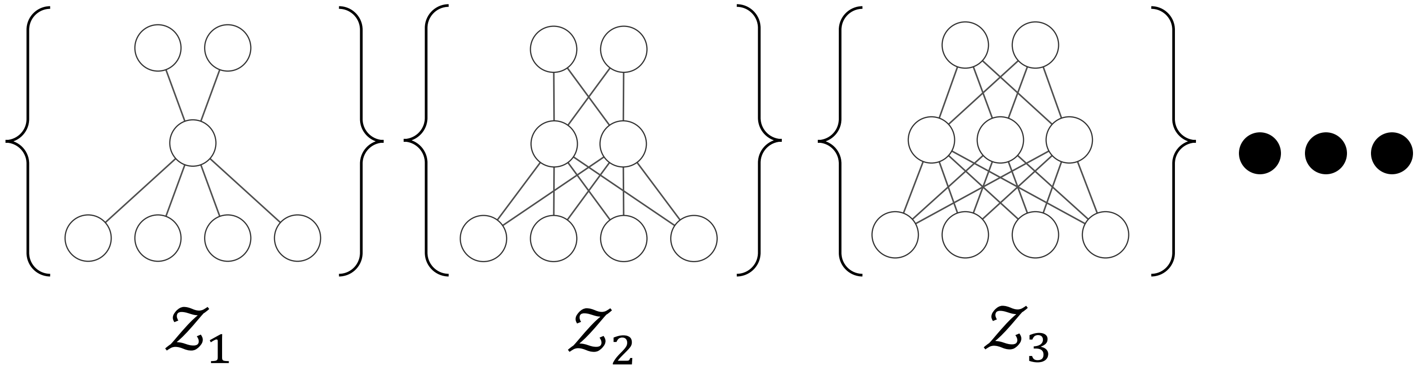


$\mathcal{Z} = \bigcup_{q\geq 0} \mathcal{Z}_q$: All 2-layer networks with different number of hidden nodes

How to Optimize?

The objective is highly nonlinear !!

However, nice *algebraic structures* exist!



$\mathcal{Z} = \cup_{q\geq 0} \mathcal{Z}_q$: All 2-layer networks with different number of hidden nodes

Ring addition $+$: Concatenate hidden nodes

Ring multiplication $*$: Kronecker production along the hidden dimensions

$\langle \mathcal{Z}, +, * \rangle$ is a ***semi-ring***

Ring Homomorphism

A function $r(\boldsymbol{z}): \mathcal{Z} \mapsto \mathbb{C}$ is a *ring homomorphism*, if

- $r(\mathbf{1}) = 1$
- $r(\boldsymbol{z}_1 + \boldsymbol{z}_2) = r(\boldsymbol{z}_1) + r(\boldsymbol{z}_2)$
- $r(\boldsymbol{z}_1 * \boldsymbol{z}_2) = r(\boldsymbol{z}_1) r(\boldsymbol{z}_2)$

Ring Homomorphism

A function $r(\mathbf{z}): \mathcal{Z} \mapsto \mathbb{C}$ is a *ring homomorphism*, if

- $r(\mathbf{1}) = 1$
- $r(\mathbf{z}_1 + \mathbf{z}_2) = r(\mathbf{z}_1) + r(\mathbf{z}_2)$
- $r(\mathbf{z}_1 * \mathbf{z}_2) = r(\mathbf{z}_1) r(\mathbf{z}_2)$

🎯 $r_{k_1 k_2 k}(\mathbf{z})$ and $r_{p k_1 k_2 k}(\mathbf{z})$ are ***ring homomorphisms***!

Ring Homomorphism

A function $r(\mathbf{z}): \mathcal{Z} \mapsto \mathbb{C}$ is a *ring homomorphism*, if

- $r(\mathbf{1}) = 1$
- $r(\mathbf{z}_1 + \mathbf{z}_2) = r(\mathbf{z}_1) + r(\mathbf{z}_2)$
- $r(\mathbf{z}_1 * \mathbf{z}_2) = r(\mathbf{z}_1) r(\mathbf{z}_2)$

$r_{k_1 k_2 k}(\mathbf{z})$ and $r_{p k_1 k_2 k}(\mathbf{z})$ are ***ring homomorphisms***!

MSE Loss

$$\ell_k(\mathbf{z}) = -2 r_{kkk} + \sum_{k_1 k_2} \left| r_{k_1 k_2 k} \right|^2 + \frac{1}{4} \left| \sum_{p \in \{a,b\}} \sum_{k'} r_{p,k',-k',k} \right|^2 + \frac{1}{4} \sum_{m \neq 0} \sum_{p \in \{a,b\}} \left| \sum_{k'} r_{p,k',m-k',k} \right|^2$$

Ring Homomorphism

A function $r(\mathbf{z}): \mathcal{Z} \mapsto \mathbb{C}$ is a *ring homomorphism*, if

- $r(\mathbf{1}) = 1$
- $r(\mathbf{z}_1 + \mathbf{z}_2) = r(\mathbf{z}_1) + r(\mathbf{z}_2)$
- $r(\mathbf{z}_1 * \mathbf{z}_2) = r(\mathbf{z}_1) r(\mathbf{z}_2)$

$r_{k_1k_2k}(\mathbf{z})$ and $r_{pk_1k_2k}(\mathbf{z})$ are ***ring homomorphisms***!

MSE Loss

$$\ell_k(\mathbf{z}) = -2r_{kkk} + \sum_{k_1k_2} \left|r_{k_1k_2k}\right|^2 + \frac{1}{4}\left|\sum_{p\in\{a,b\}}\sum_{k'} r_{p,k',-k',k}\right|^2 + \frac{1}{4}\sum_{m\neq 0}\sum_{p\in\{a,b\}}\left|\sum_{k'} r_{p,k',m-k',k}\right|^2$$

Partial solution $\mathbf{z}_1$ satisfies $r_{k_1k_2k}(\mathbf{z}_1) = 0$

Partial solution $\mathbf{z}_2$ satisfies $r_{pk',-k',k}(\mathbf{z}_2) = 0$

Ring Homomorphism

A function $r(\mathbf{z}): \mathcal{Z} \mapsto \mathbb{C}$ is a *ring homomorphism*, if

- $r(\mathbf{1}) = 1$
- $r(\mathbf{z}_1 + \mathbf{z}_2) = r(\mathbf{z}_1) + r(\mathbf{z}_2)$
- $r(\mathbf{z}_1 * \mathbf{z}_2) = r(\mathbf{z}_1) r(\mathbf{z}_2)$

$r_{k_1k_2k}(\mathbf{z})$ and $r_{pk_1k_2k}(\mathbf{z})$ are ***ring homomorphisms***!

MSE Loss

$$\ell_k(\mathbf{z}) = -2r_{kkk} + \sum_{k_1k_2} \left|r_{k_1k_2k}\right|^2 + \frac{1}{4}\left|\sum_{p\in\{a,b\}}\sum_{k'} r_{p,k',-k',k}\right|^2 + \frac{1}{4}\sum_{m\neq 0}\sum_{p\in\{a,b\}}\left|\sum_{k'} r_{p,k',m-k',k}\right|^2$$

Partial solution $\mathbf{z}_1$ satisfies $r_{k_1k_2k}(\mathbf{z}_1) = 0$

Partial solution $\mathbf{z}_2$ satisfies $r_{pk',-k',k}(\mathbf{z}_2) = 0$

$\mathbf{z} = \mathbf{z}_1 * \mathbf{z}_2$ satisfies both $r_{k_1k_2k}(\mathbf{z}) = r_{pk',-k',k}(\mathbf{z}) = 0$

Composing Global Optimizers from Partial Ones

Partial solution #1

$\boldsymbol{z}_{\text{syn}}^{(k)} \in R_{\text{c}} \cap R_{\text{n}}$ but $\boldsymbol{z}_{\text{syn}}^{(k)} \notin R_*$

Partial solution #2

$\boldsymbol{z}_{\nu}^{(k)} \in R_*$

[Y. Tian, *Composing Global Optimizers to Reasoning Tasks via Algebraic Objects in Neural Nets*, arXiv'24]

Composing Global Optimizers from Partial Ones

Compositing
solutions using
ring multiplication $*$

Partial solution #1

$\boldsymbol{z}_{\text{syn}}^{(k)} \in R_{\text{c}} \cap R_{\text{n}}$ but $\boldsymbol{z}_{\text{syn}}^{(k)} \notin R_*$

Partial solution #2

$\boldsymbol{z}_{\nu}^{(k)} \in R_*$

Better solution

$\boldsymbol{z}_{\text{syn}}^{(k)} * \boldsymbol{z}_{\nu}^{(k)} \in R_{\text{c}} \cap R_{\text{n}} \cap R_*$

[Y. Tian, *Composing Global Optimizers to Reasoning Tasks via Algebraic Objects in Neural Nets*, arXiv'24]

Composing Global Optimizers from Partial Ones

Compositing solutions using ***ring multiplication*** $*$

Compositing solutions using ***ring addition*** $+$

Partial solution #1

$\boldsymbol{z}_{\mathrm{syn}}^{(k)} \in R_{\mathrm{c}} \cap R_{\mathrm{n}}$ but $\boldsymbol{z}_{\mathrm{syn}}^{(k)} \notin R_*$

Partial solution #2

$\boldsymbol{z}_{\nu}^{(k)} \in R_*$

Better solution

$\boldsymbol{z}_{\mathrm{syn}}^{(k)} * \boldsymbol{z}_{\nu}^{(k)} \in R_{\mathrm{c}} \cap R_{\mathrm{n}} \cap R_*$

Global Optimizer to MSE loss $\ell(\boldsymbol{z})$!

$$\boldsymbol{z}_{F6} = \frac{1}{\sqrt[3]{6}} \sum_k \boldsymbol{z}_{\mathrm{syn}}^{(k)} * \boldsymbol{z}_{\nu}^{(k)}$$

[Y. Tian, *Composing Global Optimizers to Reasoning Tasks via Algebraic Objects in Neural Nets*, arXiv'24]

Exemplar constructed global optimizers

Order-6 $\boldsymbol{z}_{F6}$ (2*3)

$$\boldsymbol{z}_{F6} = \frac{1}{\sqrt[3]{6}} \sum_{k=1}^{(d-1)/2} \boldsymbol{z}_{\text{syn}}^{(k)} * \boldsymbol{z}_{\nu}^{(k)} * \boldsymbol{y}_k$$

Exemplar constructed global optimizers

Order-6 $\boldsymbol{z}_{F6}$ (2*3)

$$\boldsymbol{z}_{F6} = \frac{1}{\sqrt[3]{6}} \sum_{k=1}^{(d-1)/2} \boldsymbol{z}_{\text{syn}}^{(k)} * \boldsymbol{z}_{\nu}^{(k)} * \boldsymbol{y}_k$$

Order-4 $\boldsymbol{z}_{F4/6}$ (2*2)
(mixed with order-6)

$$\boldsymbol{z}_{F4/6} = \frac{1}{\sqrt[3]{6}} \hat{\boldsymbol{z}}_{F6}^{(k_0)} + \frac{1}{\sqrt[3]{4}} \sum_{k=1, k \neq k_0}^{(d-1)/2} \boldsymbol{z}_{F4}^{(k)}$$

Exemplar constructed global optimizers

Order-6 $\boldsymbol{z}_{F6}$ (2*3)

$$\boldsymbol{z}_{F6} = \frac{1}{\sqrt[3]{6}} \sum_{k=1}^{(d-1)/2} \boldsymbol{z}_{\text{syn}}^{(k)} * \boldsymbol{z}_{\nu}^{(k)} * \boldsymbol{y}_k$$

Order-4 $\boldsymbol{z}_{F4/6}$ (2*2)
(mixed with order-6)

$$\boldsymbol{z}_{F4/6} = \frac{1}{\sqrt[3]{6}} \hat{\boldsymbol{z}}_{F6}^{(k_0)} + \frac{1}{\sqrt[3]{4}} \sum_{k=1, k\neq k_0}^{(d-1)/2} \boldsymbol{z}_{F4}^{(k)}$$

Perfect memorization
(order-d per frequency)

$$\boldsymbol{z}_a = \sum_{j=0}^{d-1} \boldsymbol{u}_a^j, \qquad \boldsymbol{z}_b = \sum_{j=0}^{d-1} \boldsymbol{u}_b^j$$

$$\boldsymbol{z}_M = d^{-2/3} \boldsymbol{z}_a * \boldsymbol{z}_b$$

Gradient Descent solutions matches with construction

d	%not order-4/6	%non-factorable order-4	%non-factorable order-6	error ($\times 10^{-2}$) order-4	error ($\times 10^{-2}$) order-6	solution distribution (%) in factorable ones: $z^{(k)}_{\nu=\mathrm{i}} * z^{(k)}_{\xi}$	$z^{(k)}_{\nu=\mathrm{i}} * z^{(k)}_{\mathrm{syn},\alpha\beta}$	$z^{(k)}_{\nu} * z^{(k)}_{\mathrm{syn}}$	others
23	$0.0_{\pm 0.0}$	$0.00_{\pm 0.00}$	$5.71_{\pm 5.71}$	$0.05_{\pm 0.01}$	$4.80_{\pm 0.96}$	$47.07_{\pm 1.88}$	$11.31_{\pm 1.76}$	$39.80_{\pm 2.11}$	$1.82_{\pm 1.82}$
71	$0.0_{\pm 0.0}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.03_{\pm 0.00}$	$5.02_{\pm 0.25}$	$72.57_{\pm 0.70}$	$4.00_{\pm 1.14}$	$21.14_{\pm 2.14}$	$2.29_{\pm 1.07}$
127	$0.0_{\pm 0.0}$	$1.50_{\pm 0.92}$	$0.00_{\pm 0.00}$	$0.26_{\pm 0.14}$	$0.93_{\pm 0.18}$	$82.96_{\pm 0.39}$	$2.25_{\pm 0.64}$	$14.13_{\pm 0.87}$	$0.66_{\pm 0.66}$

$q = 512, wd = 5 \cdot 10^{-5}$

Gradient Descent solutions matches with construction

d	%not order-4/6	%non-factorable order-4	%non-factorable order-6	error ($\times 10^{-2}$) order-4	error ($\times 10^{-2}$) order-6	solution distribution (%) in factorable ones: $z^{(k)}_{\nu=\mathrm{i}} * z^{(k)}_{\xi}$	$z^{(k)}_{\nu=\mathrm{i}} * z^{(k)}_{\mathrm{syn},\alpha\beta}$	$z^{(k)}_{\nu} * z^{(k)}_{\mathrm{syn}}$	others
23	0.0 ± 0.0	0.00 ± 0.00	5.71 ± 5.71	0.05 ± 0.01	4.80 ± 0.96	47.07 ± 1.88	11.31 ± 1.76	39.80 ± 2.11	1.82 ± 1.82
71	0.0 ± 0.0	0.00 ± 0.00	0.00 ± 0.00	0.03 ± 0.00	5.02 ± 0.25	72.57 ± 0.70	4.00 ± 1.14	21.14 ± 2.14	2.29 ± 1.07
127	0.0 ± 0.0	1.50 ± 0.92	0.00 ± 0.00	0.26 ± 0.14	0.93 ± 0.18	82.96 ± 0.39	2.25 ± 0.64	14.13 ± 0.87	0.66 ± 0.66

100% of the per-freq solutions are order-4/6

Gradient Descent solutions matches with construction

d	%not	%non-factorable		error ($\times 10^{-2}$)		solution distribution (%) in factorable ones			
	order-4/6	order-4	order-6	order-4	order-6	$\boldsymbol{z}_{\nu=\mathrm{i}}^{(k)} * \boldsymbol{z}_{\xi}^{(k)}$	$\boldsymbol{z}_{\nu=\mathrm{i}}^{(k)} * \boldsymbol{z}_{\mathrm{syn},\alpha\beta}^{(k)}$	$\boldsymbol{z}_{\nu}^{(k)} * \boldsymbol{z}_{\mathrm{syn}}^{(k)}$	others
23	$0.0_{\pm 0.0}$	$0.00_{\pm 0.00}$	$5.71_{\pm 5.71}$	$0.05_{\pm 0.01}$	$4.80_{\pm 0.96}$	$47.07_{\pm 1.88}$	$11.31_{\pm 1.76}$	$39.80_{\pm 2.11}$	$1.82_{\pm 1.82}$
71	$0.0_{\pm 0.0}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.03_{\pm 0.00}$	$5.02_{\pm 0.25}$	$72.57_{\pm 0.70}$	$4.00_{\pm 1.14}$	$21.14_{\pm 2.14}$	$2.29_{\pm 1.07}$
127	$0.0_{\pm 0.0}$	$1.50_{\pm 0.92}$	$0.00_{\pm 0.00}$	$0.26_{\pm 0.14}$	$0.93_{\pm 0.18}$	$82.96_{\pm 0.39}$	$2.25_{\pm 0.64}$	$14.13_{\pm 0.87}$	$0.66_{\pm 0.66}$

95% of the solutions are factorizable into “2*3” or “2*2”

Gradient Descent solutions matches with construction

d	%not	%non-factorable		error ($\times 10^{-2}$)		solution distribution (%) in factorable ones			
	order-4/6	order-4	order-6	order-4	order-6	$\boldsymbol{z}^{(k)}_{\nu=\mathrm{i}} * \boldsymbol{z}^{(k)}_{\xi}$	$\boldsymbol{z}^{(k)}_{\nu=\mathrm{i}} * \boldsymbol{z}^{(k)}_{\mathrm{syn},\alpha\beta}$	$\boldsymbol{z}^{(k)}_{\nu} * \boldsymbol{z}^{(k)}_{\mathrm{syn}}$	others
23	$0.0_{\pm 0.0}$	$0.00_{\pm 0.00}$	$5.71_{\pm 5.71}$	$0.05_{\pm 0.01}$	$4.80_{\pm 0.96}$	$47.07_{\pm 1.88}$	$11.31_{\pm 1.76}$	$39.80_{\pm 2.11}$	$1.82_{\pm 1.82}$
71	$0.0_{\pm 0.0}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.03_{\pm 0.00}$	$5.02_{\pm 0.25}$	$72.57_{\pm 0.70}$	$4.00_{\pm 1.14}$	$21.14_{\pm 2.14}$	$2.29_{\pm 1.07}$
127	$0.0_{\pm 0.0}$	$1.50_{\pm 0.92}$	$0.00_{\pm 0.00}$	$0.26_{\pm 0.14}$	$0.93_{\pm 0.18}$	$82.96_{\pm 0.39}$	$2.25_{\pm 0.64}$	$14.13_{\pm 0.87}$	$0.66_{\pm 0.66}$

Factorization error is very small

Gradient Descent solutions matches with construction

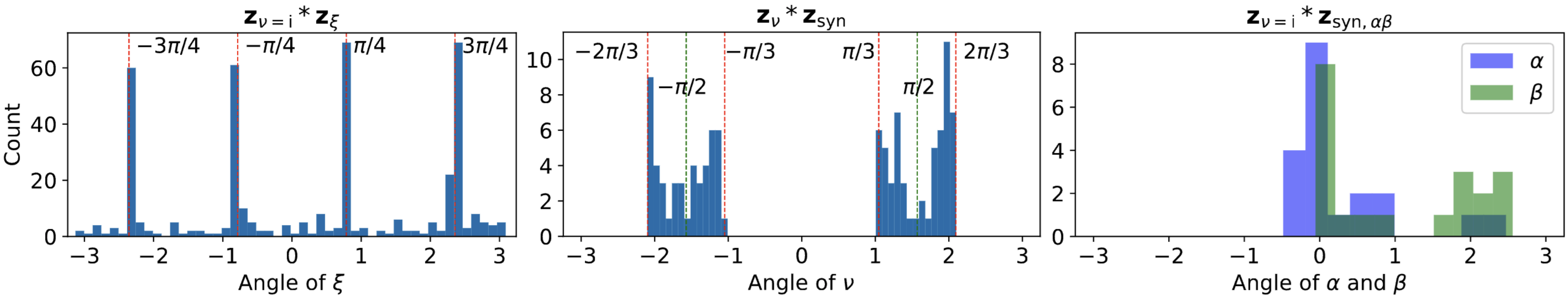
d	%not	%non-factorable		error ($\times 10^{-2}$)		solution distribution (%) in factorable ones			
	order-4/6	order-4	order-6	order-4	order-6	$\boldsymbol{z}^{(k)}_{\nu=\mathrm{i}} * \boldsymbol{z}^{(k)}_{\xi}$	$\boldsymbol{z}^{(k)}_{\nu=\mathrm{i}} * \boldsymbol{z}^{(k)}_{\mathrm{syn},\alpha\beta}$	$\boldsymbol{z}^{(k)}_{\nu} * \boldsymbol{z}^{(k)}_{\mathrm{syn}}$	others
23	$0.0_{\pm 0.0}$	$0.00_{\pm 0.00}$	$5.71_{\pm 5.71}$	$0.05_{\pm 0.01}$	$4.80_{\pm 0.96}$	$47.07_{\pm 1.88}$	$11.31_{\pm 1.76}$	$39.80_{\pm 2.11}$	$1.82_{\pm 1.82}$
71	$0.0_{\pm 0.0}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.03_{\pm 0.00}$	$5.02_{\pm 0.25}$	$72.57_{\pm 0.70}$	$4.00_{\pm 1.14}$	$21.14_{\pm 2.14}$	$2.29_{\pm 1.07}$
127	$0.0_{\pm 0.0}$	$1.50_{\pm 0.92}$	$0.00_{\pm 0.00}$	$0.26_{\pm 0.14}$	$0.93_{\pm 0.18}$	$82.96_{\pm 0.39}$	$2.25_{\pm 0.64}$	$14.13_{\pm 0.87}$	$0.66_{\pm 0.66}$

98% of the solutions can be factorizable into the constructed forms

Gradient Descent solutions matches with construction

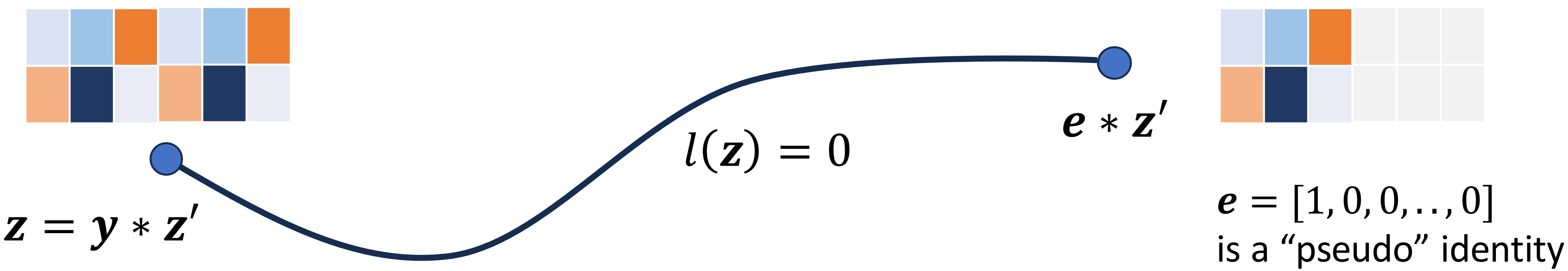
d	%not	%non-factorable		error ($\times 10^{-2}$)		solution distribution (%) in factorable ones			
	order-4/6	order-4	order-6	order-4	order-6	$z^{(k)}_{\nu=\mathrm{i}} * z^{(k)}_{\xi}$	$z^{(k)}_{\nu=\mathrm{i}} * z^{(k)}_{\mathrm{syn},\alpha\beta}$	$z^{(k)}_{\nu} * z^{(k)}_{\mathrm{syn}}$	others
23	0.0±0.0	0.00±0.00	5.71±5.71	0.05±0.01	4.80±0.96	47.07±1.88	11.31±1.76	39.80±2.11	1.82±1.82
					5	72.57±0.70	4.00±1.14	21.14±2.14	2.29±1.07
					8	82.96±0.39	2.25±0.64	14.13±0.87	0.66±0.66

Distribution of the parameters in the solutions



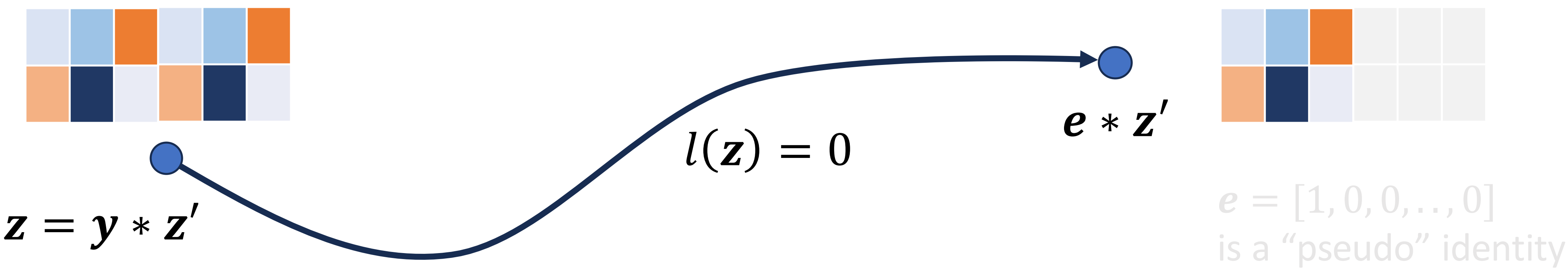
Gradient Dynamics

Theorem [**The Occam's Razer**] If $\boldsymbol{z} = \boldsymbol{y} * \boldsymbol{z}'$ and both $\boldsymbol{z}$ and $\boldsymbol{z}'$ are global optimal, then there exists a path of zero loss connecting $\boldsymbol{z}$ and $\boldsymbol{z}'$.



Gradient Dynamics

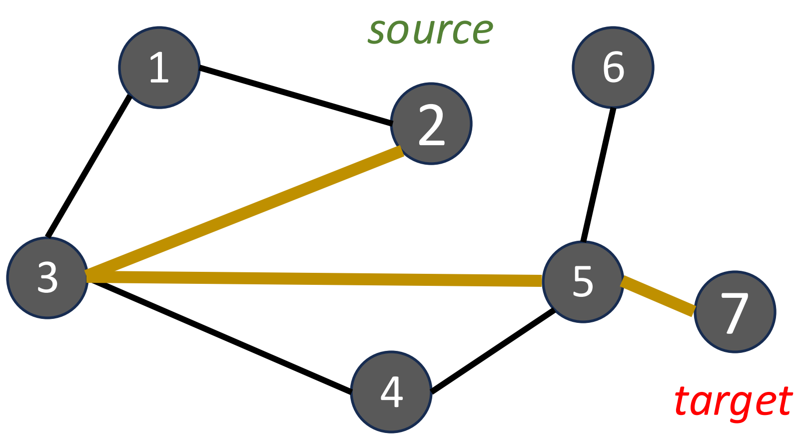
Theorem [**The Occam's Razer**] If $\boldsymbol{z} = \boldsymbol{y} * \boldsymbol{z}'$ and both $\boldsymbol{z}$ and $\boldsymbol{z}'$ are global optimal, then there exists a path of zero loss connecting $\boldsymbol{z}$ and $\boldsymbol{z}'$.



L2 regularization will push the solution to $\boldsymbol{e} * \boldsymbol{z}'$ (simpler solutions), since $\|\boldsymbol{e} * \boldsymbol{z}'\|_2 \leq \|\boldsymbol{y} * \boldsymbol{z}'\|_2$

Another Example: Symbolic from Neural Representation

Task: Learn a 2-layer Transformer for predicting **shortest path** in the graph



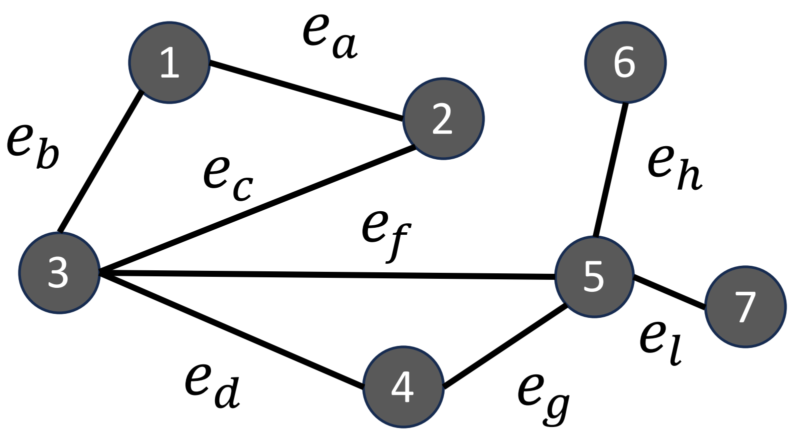
<bos> 1 2 <e> ... <q> [source] [target] <p> [source] [node 1] [node 2] ... [target]

Context

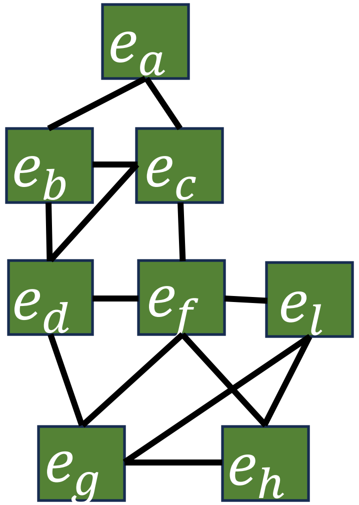
Predicted Shortest path

[A. Cohen et al, *Spectral Journey: How Transformers Predict the Shortest Path,* arXiv'25]

What representations it learns?



Line graph



Normalized Graph Laplacian

$$L = I - D^{-1/2} A D^{-1/2}$$

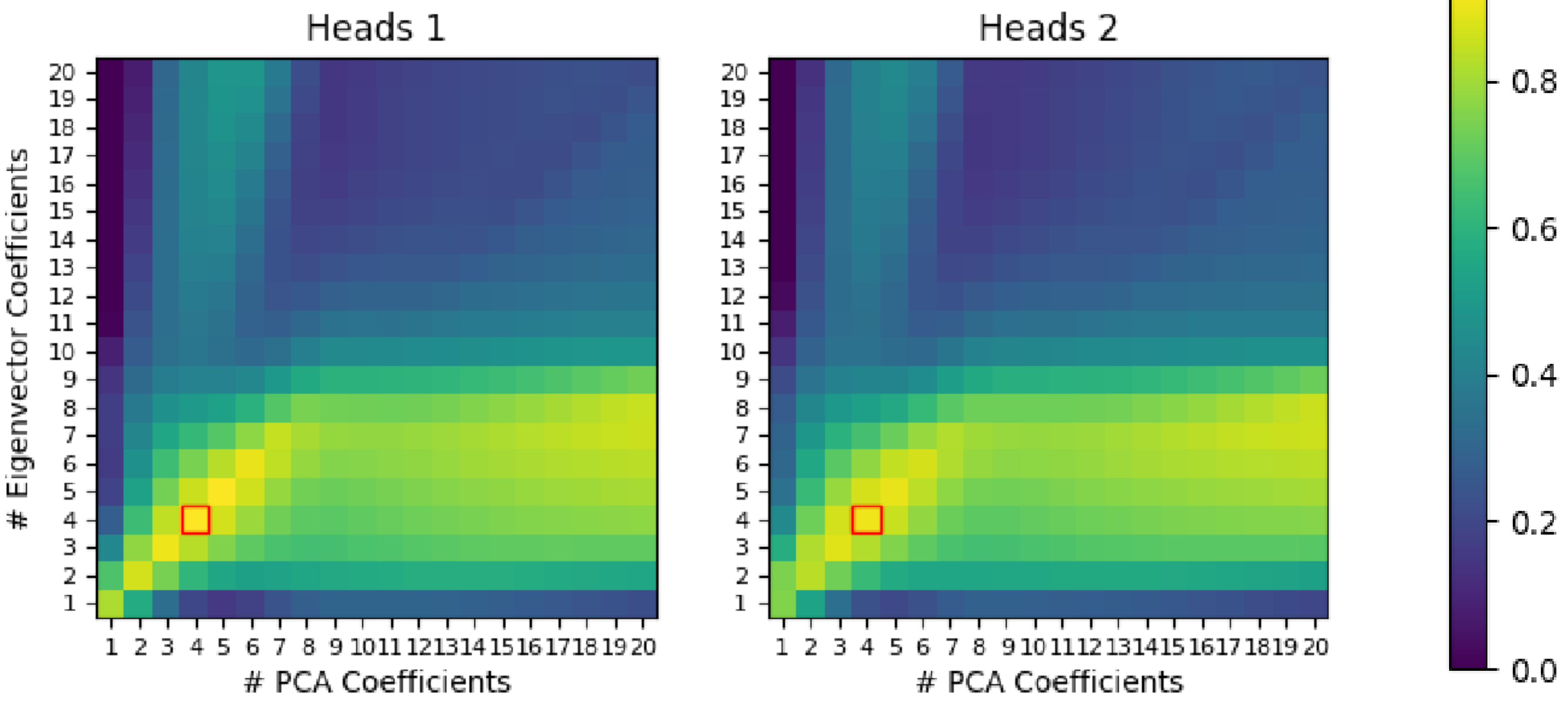
Edge Embedding

Representation after the first Transformer layer (averaged over random edge order)

<bos> 1 2 **<e>** ... <q> [source] [target] <p> [source] [node 1] [node 2] ... [target]

What representations it learns?

Graph Edge Embedding of various dimensions



Computed edge embedding with trained Transformers

Normalized Correlation > 0.9

Spectral Line Navigator (SLN)

<u>Simple Algorithms of Graph Shortest Path</u>

1. Compute Line Graph $\tilde{G}$ of existing graph G
2. Compute eigenvectors of normalized Laplacian $L(\tilde{G})$
3. $i = source$
4. While $i \neq target$ do

   $distance(j, k; i) := \left\| v_{ij} - v_{k,target} \right\|_2$

   Find $j = \text{argmin}_{j,k}\, distance(j, k; i)$

   Let $i = j$

>99% optimal for small random graph (size < 10)

o3-mini-high implementation: https://chatgpt.com/share/67b027f9-fb28-8012-aa64-a1f7479134b7

Possible Implications

Do neural networks end up learning more efficient **symbolic representations** that we don't know?

Does gradient descent lead to a solution that can be reached by **advanced algebraic operations**?

Will gradient descent become **obsolete**, eventually?

Thanks!

Thanks!