Towards Better Understanding of Representation Collapsing in Representation Learning

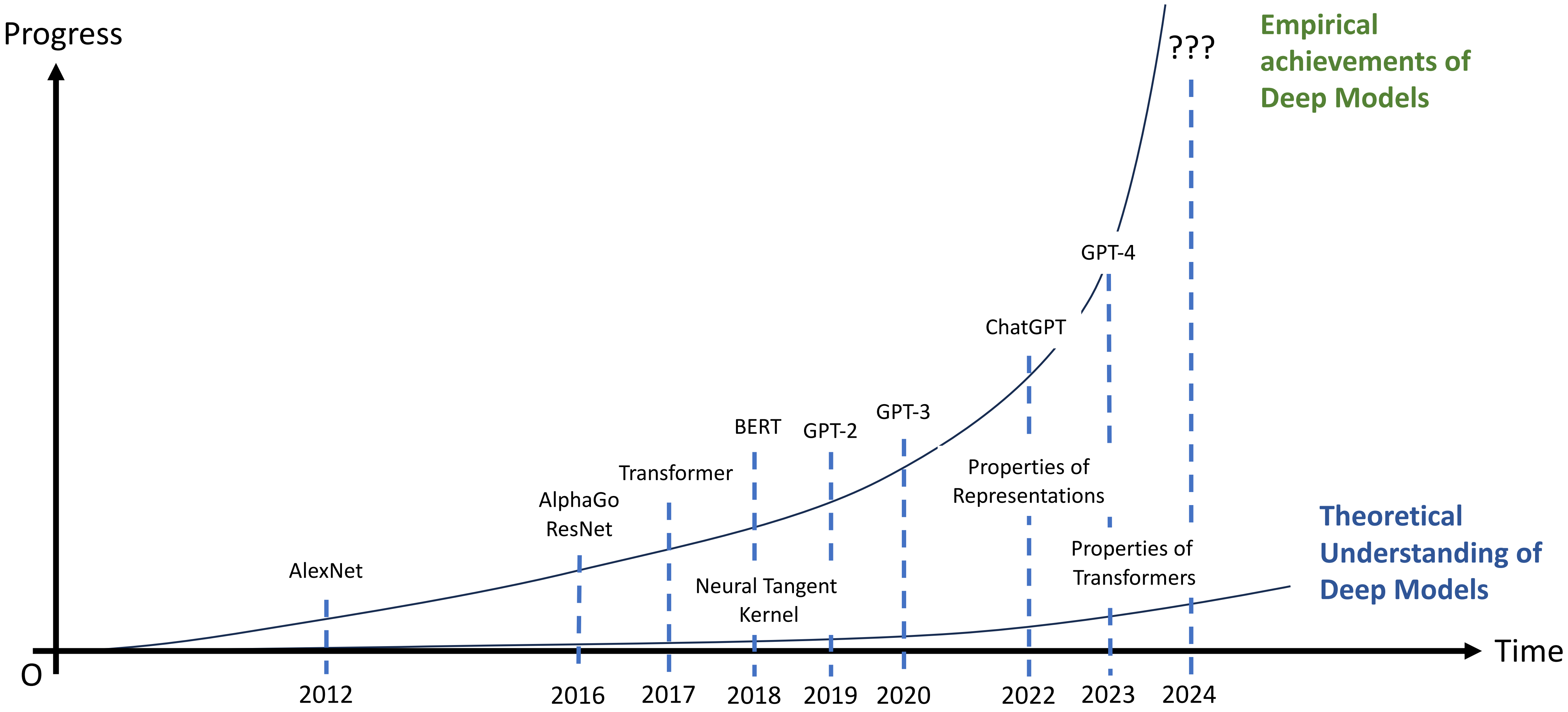
Yuandong Tian

Research Scientist

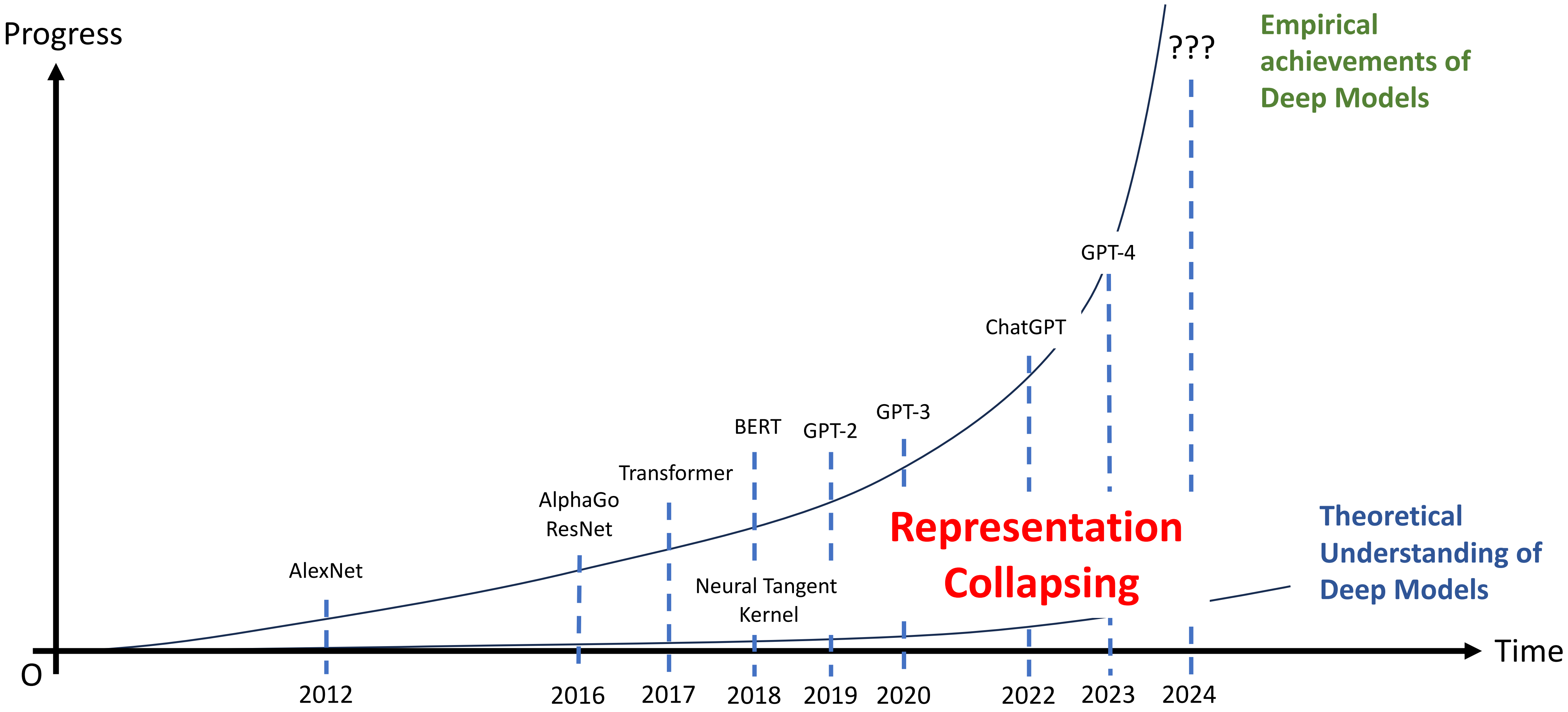
Meta AI (FAIR)

Great Empirical Success of Deep Models

A sharp difference between theory and practice

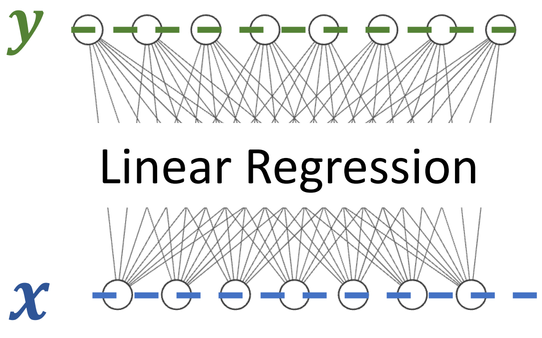


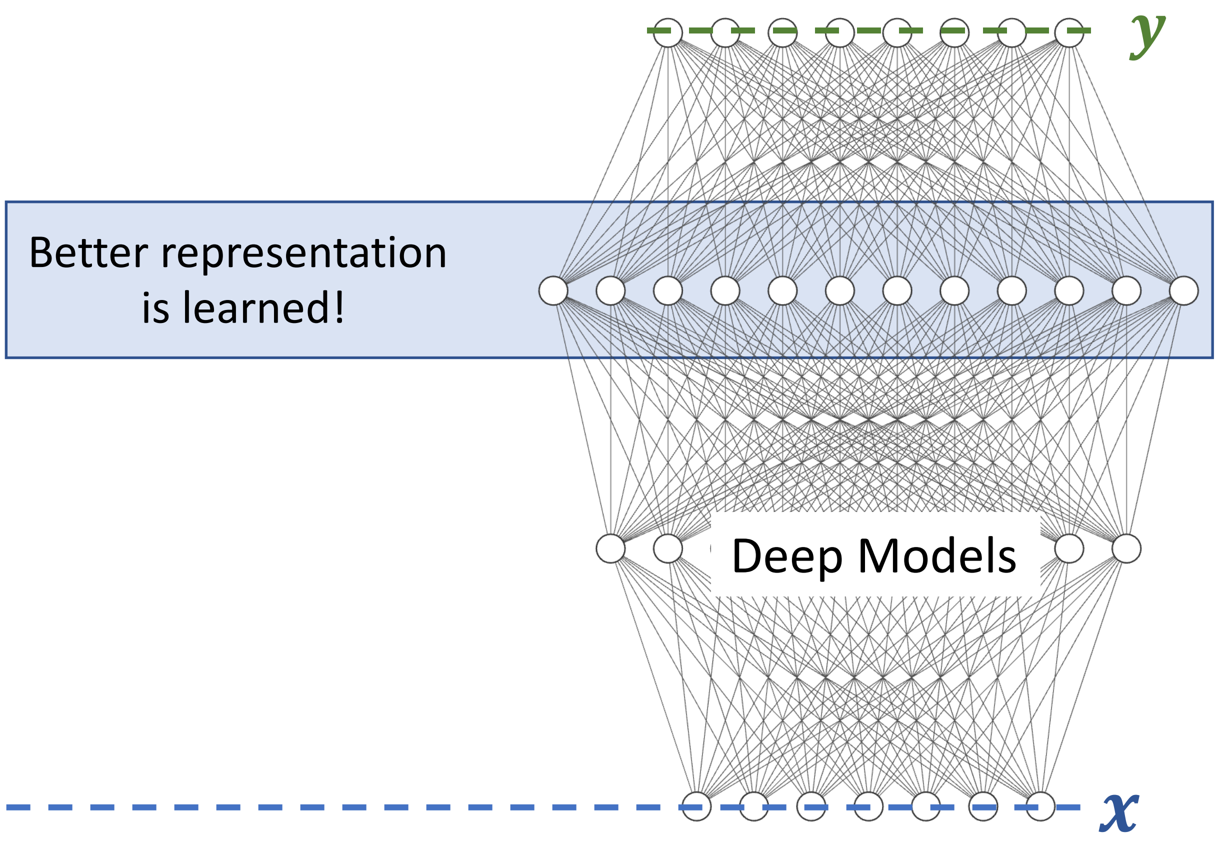
A sharp difference between theory and practice



What Deep Learning Brings?

Same loss function
Different representation



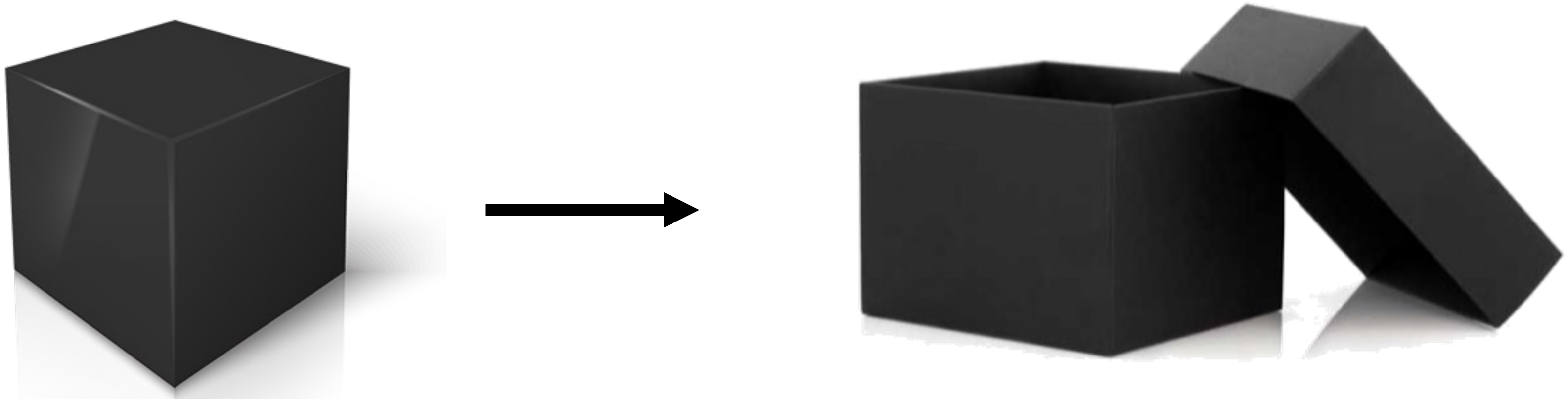


Before deep learning era

Deep learning era

Overall Research Philosophy

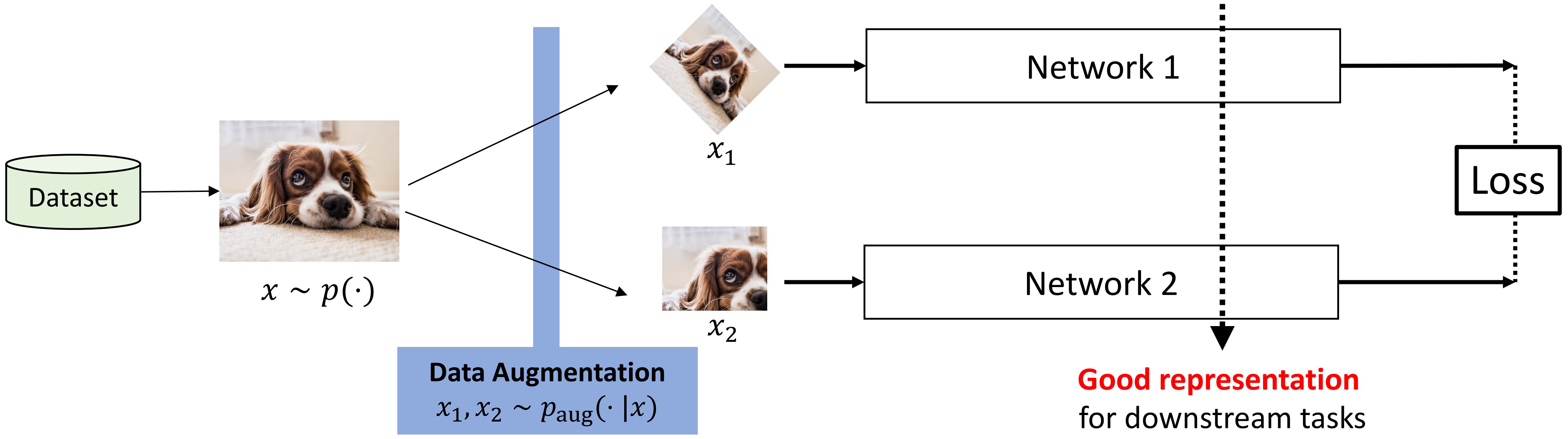
- Analyze the property of *training loss* **plus** *Neural architecture*
- Propose novel loss / architectures that lead to empirical improvement

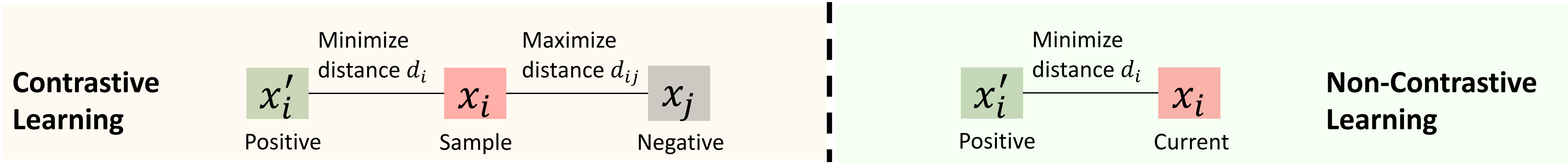


Blackbox

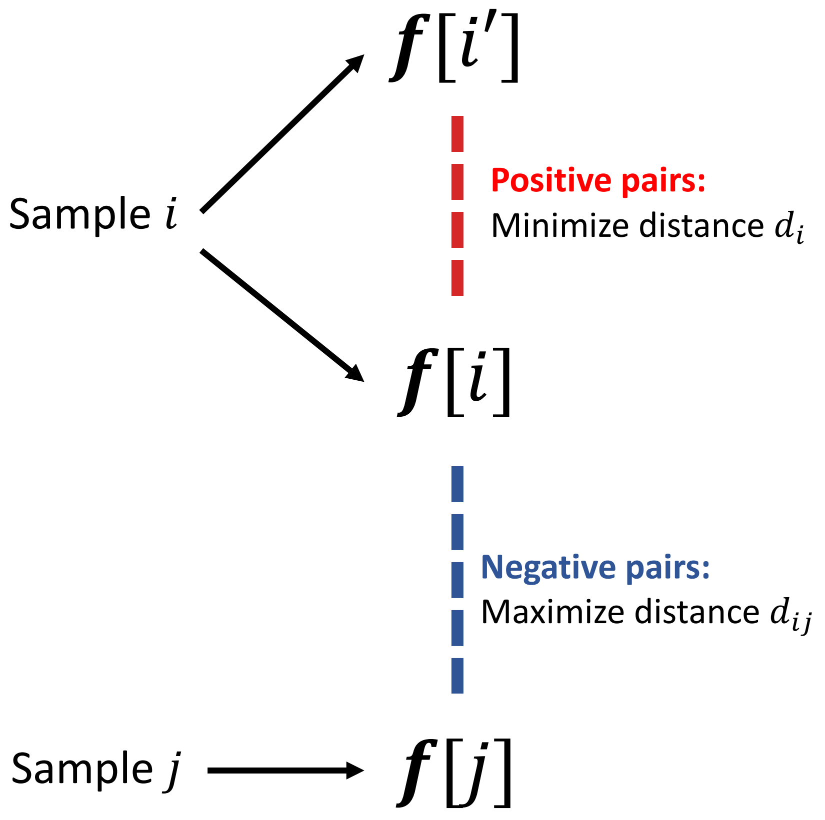
Open the Blackbox

Contrastive versus Non-contrastive Learning





Formulation of Contrastive Learning



InfoNCE loss:

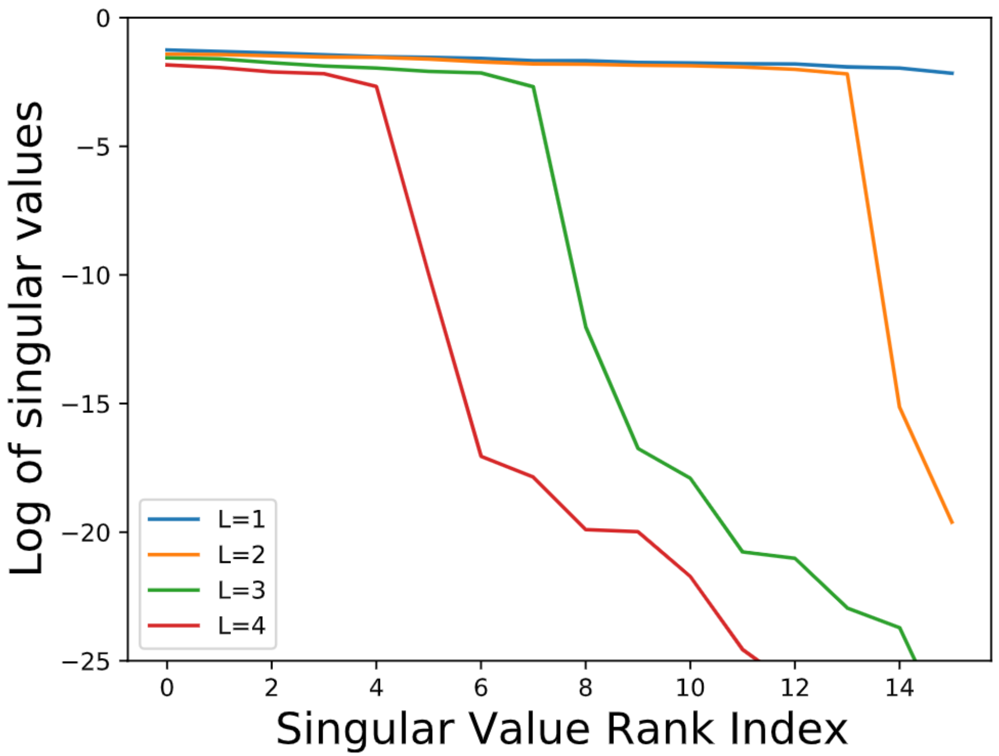
$$\mathcal{L}_{nce} := -\tau \sum_{i=1}^{N} \log \frac{\exp(-d_i^2/\tau)}{\epsilon \exp\left(-d_i^2/\tau\right) + \sum_{j\neq i} \exp(-d_{ij}^2/\tau)}$$

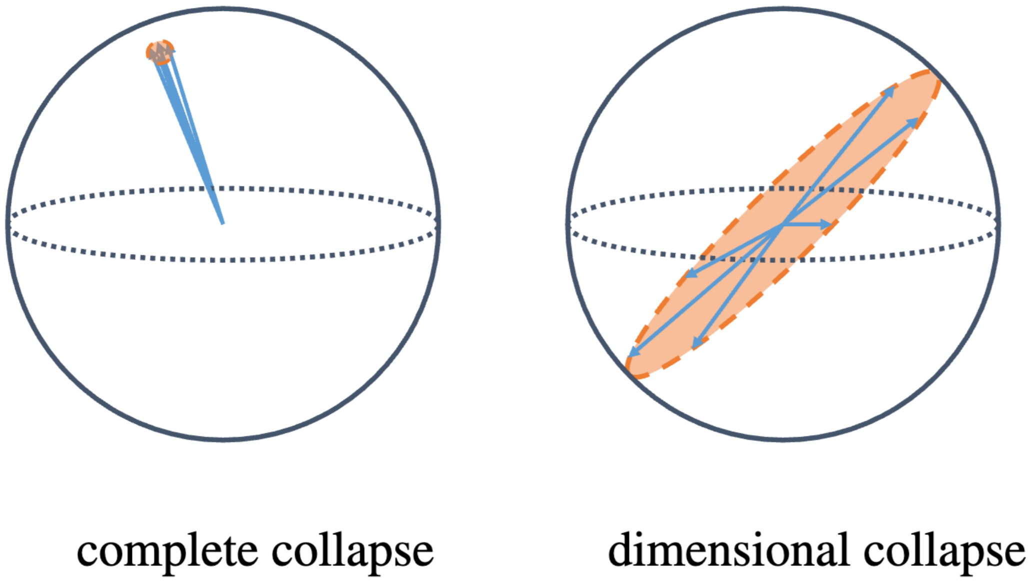
Intra-view distance $d_i^2 = \|\boldsymbol{f}[i] - \boldsymbol{f}[i']\|_2^2/2$

Inter-view distance $d_{ij}^2 = \|\boldsymbol{f}[i] - \boldsymbol{f}[j]\|_2^2/2$

Representation Collapses in Contrastive Learning

Shouldn't contrastive learning make full use of all dimensions? The answer is **No…**

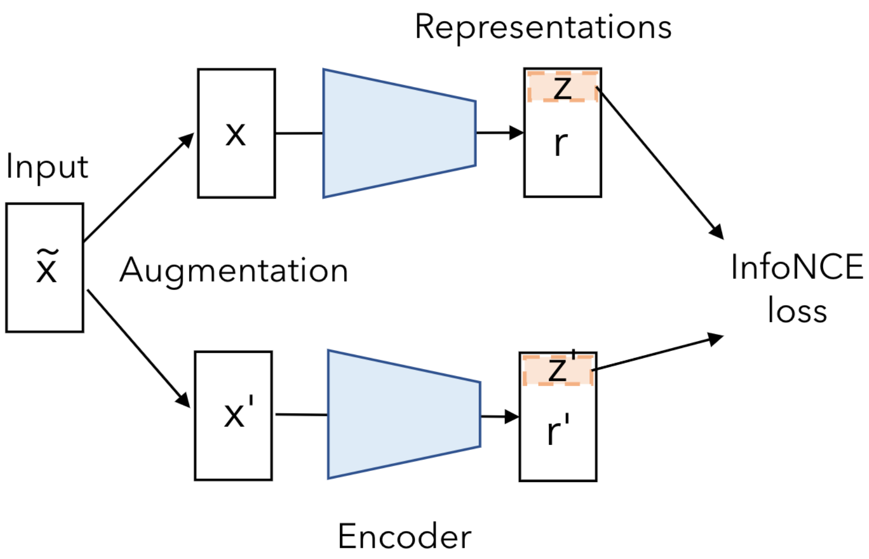




DirectCLR *[L. Jing, P. Vincent, Y. LeCun, **Y. Tian**, Understanding Dimensional Collapse in Contrastive Self-supervised Learning, ICLR'22]*

Representation Collapses in Contrastive Learning

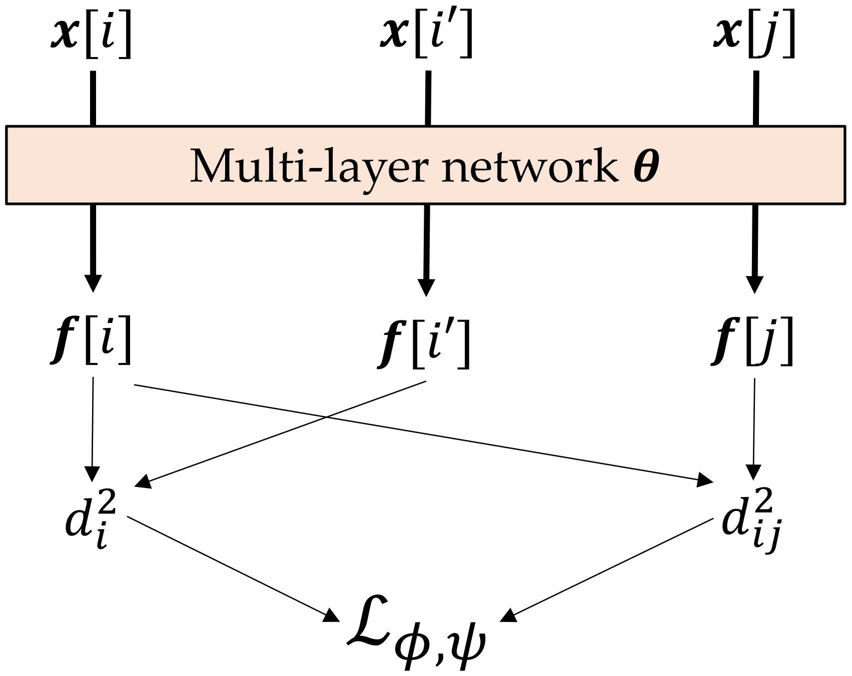
If things are collapsed during training, why not just pick a subset of the dimensions directly?



Loss function	Projector	Top-1 Accuracy
SimCLR	2-layer nonlinear projector	66.5
SimCLR	1-layer linear projector	61.1
SimCLR	no projector	51.5
DirectCLR	no projector	62.7

A family of contrastive losses

General Loss function we consider (ϕ, ψ are monotonous increasing functions)



$$\min_{\boldsymbol{\theta}} \mathcal{L}_{\phi,\psi}(\boldsymbol{\theta}) := \sum_{i=1}^{N} \phi\left(\sum_{j \neq i} \psi(d_i^2 - d_{ij}^2)\right)$$

Intra-view distance $d_i^2 = \|\boldsymbol{f}[i] - \boldsymbol{f}[i']\|_2^2/2$

Inter-view distance $d_{ij}^2 = \|\boldsymbol{f}[i] - \boldsymbol{f}[j]\|_2^2/2$

[Y. Tian, Understanding Deep Contrastive Learning via Coordinate-wise Optimization, NeurIPS'22]

Example: InfoNCE

$$\mathcal{L}_{nce} := -\tau \sum_{i=1}^{N} \log \frac{\exp(-d_i^2/\tau)}{\epsilon \exp\left(-d_i^2/\tau\right) + \sum_{j \neq i} \exp(-d_{ij}^2/\tau)}$$

$$= \tau \sum_{i=1}^{N} \log \left(\epsilon + \sum_{j \neq i} \exp\left(\frac{d_i^2 - d_{ij}^2}{\tau} \right) \right)$$

$$\phi(x) = \tau \log(\epsilon + x) \qquad \psi(x) = \exp(x/\tau)$$

Coordinate-wise Optimization

Minimizing contrastive loss $\mathcal{L}_{\phi,\psi}$ ⇔ Coordinate-wise optimization:

$$\alpha_t := \arg\min_{\alpha \in \mathcal{A}} \mathcal{E}_\alpha(\boldsymbol{\theta}_t) - \mathcal{R}(\alpha)$$

$$\boldsymbol{\theta}_{t+1} := \boldsymbol{\theta}_t + \eta \nabla_{\boldsymbol{\theta}} \mathcal{E}_{\alpha_t}(\boldsymbol{\theta}_t)$$

Max-player $\boldsymbol{\theta}$

Learns the representation to maximize contrastiveness.

Min-player α

Find distinct sample pairs that share similar representation (**hard negative pairs**)

The Energy Function $\mathcal{E}_\alpha(\boldsymbol{\theta})$

The energy $\mathcal{E}_\alpha$ is defined as the *trace* of ***contrastive covariance*** $\mathbb{C}_\alpha$:

$$\mathcal{E}_\alpha(\boldsymbol{\theta}) := \mathrm{tr}\,\mathbb{C}_\alpha[\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}), \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x})]$$

The contrastive covariance $\mathbb{C}_\alpha[\boldsymbol{x}, \boldsymbol{y}] := \Sigma_0 - \Sigma_{\mathrm{aug}}$

Inter-sample $\Sigma_0 := \sum_{i,j} \alpha_{ij}(\boldsymbol{x}[i] - \boldsymbol{x}[j])(\boldsymbol{y}[i] - \boldsymbol{y}[j])^T$

Intra-sample $\Sigma_{\mathrm{aug}} := \sum_i \left(\sum_{j \neq i} \alpha_{ij}\right)(\boldsymbol{x}[i] - \boldsymbol{x}[i'])(\boldsymbol{y}[i] - \boldsymbol{y}[i'])^T$

A general family

Contrastive Loss	$\phi(x)$	$\psi(x)$
InfoNCE (Oord et al., 2018)	$\tau \log(\epsilon + x)$	$e^{x/\tau}$
MINE (Belghazi et al., 2018)	$\log(x)$	e^{x}
Triplet (Schroff et al., 2015)	x	$[x + \epsilon]_+$
Soft Triplet (Tian et al., 2020c)	$\tau \log(1 + x)$	$e^{x/\tau+\epsilon}$
N+1 Tuplet (Sohn, 2016)	$\log(1 + x)$	e^{x}
Lifted Structured (Oh Song et al., 2016)	$[\log(x)]_+^2$	$e^{x+\epsilon}$
(Coria et al., 2020)	x	$\text{sigmoid}(cx)$
(Ji et al., 2021)	linear	linear

Different Losses, Same Energy Function

Contrastive Loss	$\phi(x)$	$\psi(x)$
InfoNCE (Oord et al., 2018)	$\tau \log(\epsilon + x)$	$e^{x/\tau}$
MINE (Belghazi et al., 2018)	$\log(x)$	e^{x}
Triplet (Schroff et al., 2015)	x	$[x+\epsilon]_+$
Soft Triplet (Tian et al., 2020c)	$\tau \log(1 + x)$	$e^{x/\tau+\epsilon}$
N+1 Tuplet (Sohn, 2016)	$\log(1 + x)$	e^{x}
Lifted Structured (Oh Song et al., 2016)	$[\log(x)]_+^2$	$e^{x+\epsilon}$
(Coria et al., 2020)	x	$\text{sigmoid}(cx)$
(Ji et al., 2021)	linear	linear

Different loss functions (ϕ, ψ) corresponds to the **same energy function $\mathcal{E}$**

How the min player α operates are different.

How min player $\boldsymbol{\alpha}$ is determined?

If $\psi(x) = e^{x/\tau}$, then we have $\alpha(\boldsymbol{\theta}) \coloneqq \arg\min_{\alpha \in \mathcal{A}} \mathcal{E}_\alpha(\boldsymbol{\theta}) - \mathcal{R}(\alpha)$

where the feasible set $\mathcal{A} \coloneqq \left\{ \alpha\colon \ \forall i, \sum_{j \neq i} \alpha_{ij} = \tau^{-1} \xi_i \phi'(\xi_i), \alpha_{ij} \geq 0 \right\}$

and entropy regularization term $\mathcal{R}(\alpha) \coloneqq 2\tau \sum_{i=1}^{N} H(\alpha_{i\cdot})$

$$\xi_i \coloneqq \sum_{j \neq i} \psi(d_i^2 - d_{ij}^2)$$

For infoNCE with $\epsilon = 0$, solving the optimization problem yields:

$$\alpha_{ij}(\boldsymbol{\theta}) = \frac{\exp\left(-d_{ij}^2/\tau\right)}{\sum_{j \neq i} \exp\left(-d_{ij}^2/\tau\right)}$$

We put more weights on **small $\boldsymbol{d_{ij}}$**, i.e., distinct samples with similar representations

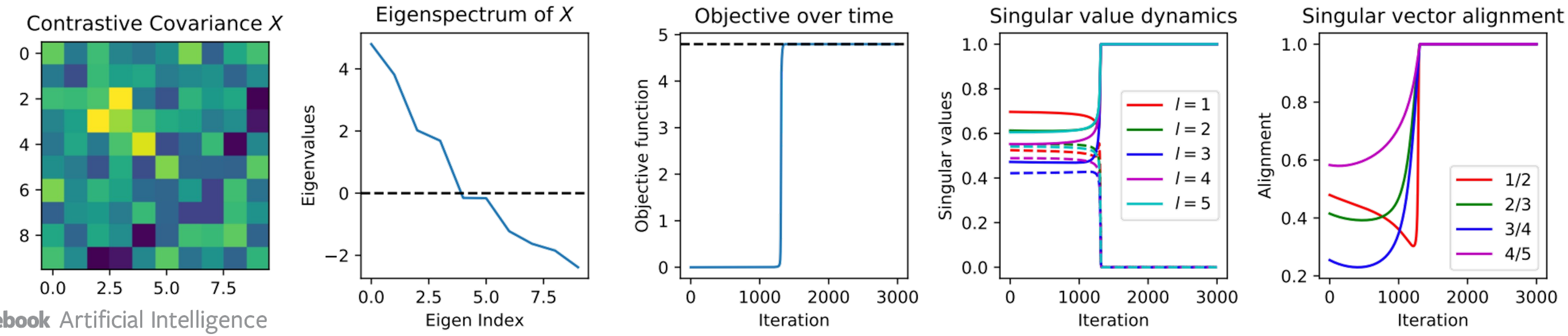
Feature Collapsing! Deep linear case with fixed $\boldsymbol{\alpha}$

If $\boldsymbol{f_\theta}(\boldsymbol{x}) = W_L W_{L-1} \ldots W_1 x$, then almost all local optima are global and it is PCA

Theorem 3 (Representation Learning with `DeepLin` is PCA). *If $\lambda_{\max}(X_\alpha) > 0$, then for any local maximum $\boldsymbol{\theta} \in \Theta$ of Eqn. 11 whose $W_{>1}^\top W_{>1}$ has distinct maximal eigenvalue:*

- *there exists a set of unit vectors $\{\boldsymbol{v}_l\}_{l=0}^{L}$ so that $W_l = \boldsymbol{v}_l \boldsymbol{v}_{l-1}^\top$ for $1 \le l \le L$, in particular, $\boldsymbol{v}_0$ is the unit eigenvector corresponding to $\lambda_{\max}(X_\alpha)$,*
- *$\boldsymbol{\theta}$ is global optimal with objective $\mathcal{E}^* = \lambda_{\max}(X_\alpha)$.*

1. Nearby weights align
2. All W_l has rank-1 structure



Coordinate-wise Optimization

Minimizing $\mathcal{L}_{\phi,\psi}$ ⇔ Coordinate-wise optimization:

$$\alpha_t \coloneqq \arg\min_{\alpha \in \mathcal{A}} \mathcal{E}_\alpha(\boldsymbol{\theta}_t) - \mathcal{R}(\alpha)$$

$$\boldsymbol{\theta}_{t+1} \coloneqq \boldsymbol{\theta}_t + \eta \nabla_{\boldsymbol{\theta}} \mathcal{E}_{\alpha_t}(\boldsymbol{\theta}_t)$$

Coordinate-wise Optimization

Minimizing $\mathcal{L}_{\phi,\psi}$ ⇔ Coordinate-wise optimization:

$$\alpha_t \coloneqq \arg\min_{\alpha \in \mathcal{A}} \mathcal{E}_\alpha(\boldsymbol{\theta}_t) - \mathcal{R}(\alpha)$$

$$\boldsymbol{\theta}_{t+1} \coloneqq \boldsymbol{\theta}_t + \eta \nabla_{\boldsymbol{\theta}} \mathcal{E}_{\alpha_t}(\boldsymbol{\theta}_t)$$

Proposed: Pair-weighed CL ($\boldsymbol{\alpha}$-CL)

The min player α can be optimized by a loss function, or ***directly*** specified:

$$\alpha_t = \alpha(\boldsymbol{\theta}_t)$$ **Pairwise importance**

$$\boldsymbol{\theta}_{t+1} := \boldsymbol{\theta}_t + \eta \nabla_{\boldsymbol{\theta}} \mathcal{E}_{\alpha_t}(\boldsymbol{\theta}_t)$$

Experimental Results

	CIFAR-10			*STL-10*		
	100 epochs	300 epochs	500 epochs	100 epochs	300 epochs	500 epochs
$\mathcal{L}_{quadratic}$	63.59 ± 2.53	73.02 ± 0.80	73.58 ± 0.82	55.59 ± 4.00	64.97 ± 1.45	67.28 ± 1.21
$\mathcal{L}_{nce}$	84.06 ± 0.30	87.63 ± 0.13	87.86 ± 0.12	78.46 ± 0.24	82.49 ± 0.26	83.70 ± 0.12
backprop $\alpha(\boldsymbol{\theta})$	83.42 ± 0.25	87.18 ± 0.19	87.48 ± 0.21	77.88 ± 0.17	81.86 ± 0.30	83.19 ± 0.16
α-CL-r_H	84.27 ± 0.24	87.75 ± 0.25	87.92 ± 0.24	78.53 ± 0.35	82.62 ± 0.15	83.74 ± 0.18
α-CL-r_γ	83.72 ± 0.19	87.51 ± 0.11	87.69 ± 0.09	78.22 ± 0.28	82.19 ± 0.52	83.47 ± 0.34
α-CL-r_s	84.72 ± 0.10	86.62 ± 0.17	86.74 ± 0.15	76.95 ± 1.06	80.64 ± 0.77	81.65 ± 0.59
α-CL-direct	$\mathbf{85.09 \pm 0.13}$	$\mathbf{88.00 \pm 0.12}$	$\mathbf{88.16 \pm 0.12}$	$\mathbf{79.38 \pm 0.16}$	$\mathbf{82.99 \pm 0.15}$	$\mathbf{84.06 \pm 0.24}$

- (α-CL-r_H) Entropy regularizer $r_H(\alpha_{ij}) = -2\tau\alpha_{ij}\log\alpha_{ij}$;
- (α-CL-r_γ) Inverse regularizers $r_\gamma(\alpha_{ij}) = \frac{2\tau}{1-\gamma}\alpha_{ij}^{1-\gamma}$ ($\gamma > 1$).
- (α-CL-r_s) Square regularizer $r_s(\alpha_{ij}) = -\frac{\tau}{2}\alpha_{ij}^2$.
- (α-CL-direct) Directly setting α: $\alpha_{ij} = \exp(-d_{ij}^p/\tau)$ ($p > 1$).

Experimental Results

More datasets

	CIFAR-100		
	100 epochs	300 epochs	500 epochs
$\mathcal{L}_{nce}$	55.696 ± 0.368	59.706 ± 0.360	59.892 ± 0.340
α-CL-direct	$\mathbf{57.144 \pm 0.150}$	$\mathbf{60.110 \pm 0.187}$	$\mathbf{60.330 \pm 0.194}$

Backbone = ResNet50

Dataset	Method	100 epochs	300 epochs	500 epochs
CIFAR-10	$\mathcal{L}_{nce}$	86.388 ± 0.157	89.974 ± 0.138	90.194 ± 0.232
	α-CL-direct	$\mathbf{87.406 \pm 0.227}$	$\mathbf{90.228 \pm 0.185}$	$\mathbf{90.366 \pm 0.209}$
CIFAR-100	$\mathcal{L}_{nce}$	60.162 ± 0.482	65.400 ± 0.310	65.532 ± 0.297
	α-CL-direct	$\mathbf{62.650 \pm 0.181}$	$\mathbf{65.630 \pm 0.263}$	$\mathbf{65.636 \pm 0.269}$
STL-10	$\mathcal{L}_{nce}$	81.635 ± 0.244	86.570 ± 0.174	$\mathbf{87.900 \pm 0.222}$
	α-CL-direct	$\mathbf{82.850 \pm 0.171}$	$\mathbf{86.870 \pm 0.178}$	87.653 ± 0.175

Nonlinear Setting

CL with linear model connects with classic approaches.

Where does the magic of deep models come from?

Nonlinearity!

[Y. Tian, Understanding the Role of Nonlinearity in Training Dynamics of Contrastive Learning, ICLR'23]

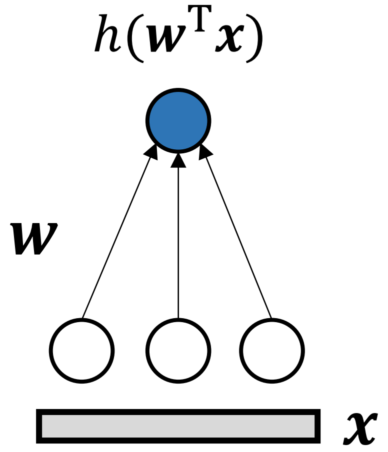
Overview of Nonlinear Analysis

- One and Two-layer nonlinear networks
- Homogenous activations: $h(x) = h'(x)x$
  - Linear, ReLU, leaky ReLU and monomial activations $h(x) = x^p$ (with additional constant)
- Training Dynamics / Critical Point Analysis
  - Statistics of local optima.
  - Dynamics of weights during training

[Y. Tian, Understanding the Role of Nonlinearity in Training Dynamics of Contrastive Learning, ICLR'23]

Nonlinear Setting

One-layer nonlinear network: $f_{\boldsymbol{\theta}}(\boldsymbol{x}) = h(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x})$

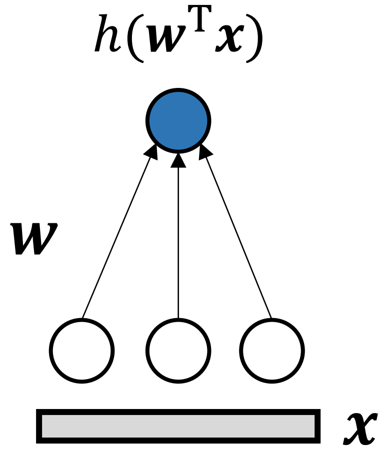


1-layer network

$$\max_{\|\boldsymbol{w}\|_2=1} \mathbb{C}_\alpha[f_{\boldsymbol{\theta}}] = \mathbb{C}_\alpha[h(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x})]$$

Nonlinear Setting

One-layer nonlinear network: $f_{\boldsymbol{\theta}}(\boldsymbol{x}) = h(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x})$



1-layer network

$$\max_{\|\boldsymbol{w}\|_2=1} \mathbb{C}_\alpha[f_{\boldsymbol{\theta}}] = \mathbb{C}_\alpha[h(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x})]$$

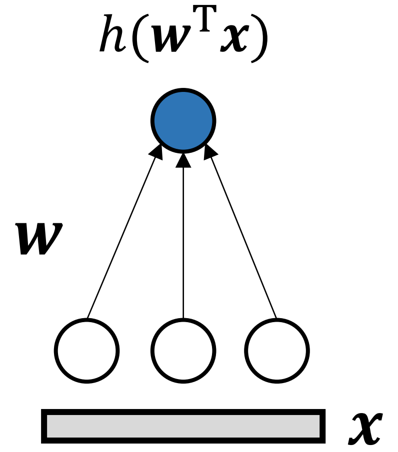
Homogeneity: $\mathbb{C}_\alpha\left[h\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}\right)\right] = \boldsymbol{w}^{\mathrm{T}}\mathbb{C}_\alpha[\tilde{\boldsymbol{x}}^{\boldsymbol{w}}]\boldsymbol{w}$

$\tilde{\boldsymbol{x}}^{\boldsymbol{w}} \coloneqq \boldsymbol{x} \cdot h'(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x})$ is the ***gated*** data point

Similar to covariance matrix in PCA, but now the matrix is not constant.

Nonlinear Setting

One-layer nonlinear network: $f_{\boldsymbol{\theta}}(\boldsymbol{x}) = h(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x})$



1-layer network

$$\max_{\|\boldsymbol{w}\|_2=1} \mathbb{C}_\alpha[f_{\boldsymbol{\theta}}] = \mathbb{C}_\alpha[h(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x})]$$

Homogeneity: $\mathbb{C}_\alpha\left[h\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}\right)\right] = \boldsymbol{w}^{\mathrm{T}}A(\boldsymbol{w})\boldsymbol{w}$

$\tilde{\boldsymbol{x}}^{\boldsymbol{w}} := \boldsymbol{x} \cdot h'(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x})$ is the ***gated*** data point

$$\max_{\|w\|_2=1} \boldsymbol{w}^{\mathrm{T}}A(\boldsymbol{w})\boldsymbol{w}$$

Training Dynamics

$$\max_{\|w\|_2=1} \boldsymbol{w}^{\mathrm{T}} A(\boldsymbol{w}) \boldsymbol{w}$$

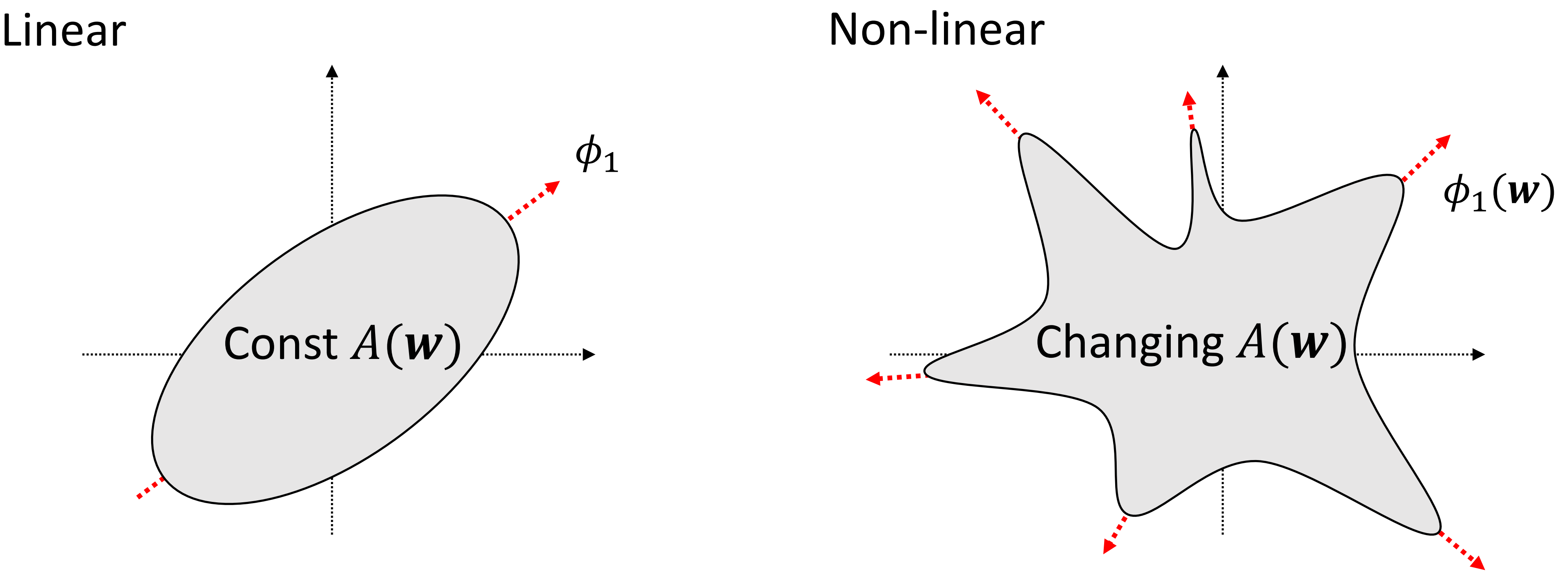
$P_{\boldsymbol{w}}^{\perp} := I - \boldsymbol{w}\boldsymbol{w}^T$ is the projection matrix

$$\dot{\boldsymbol{w}} = P_{\boldsymbol{w}}^{\perp} A(\boldsymbol{w}) \boldsymbol{w}$$

Very much like power iteration, but $\boldsymbol{A}(\boldsymbol{w})$ changes over $\boldsymbol{w}$!

1-layer 1-node nonlinear network

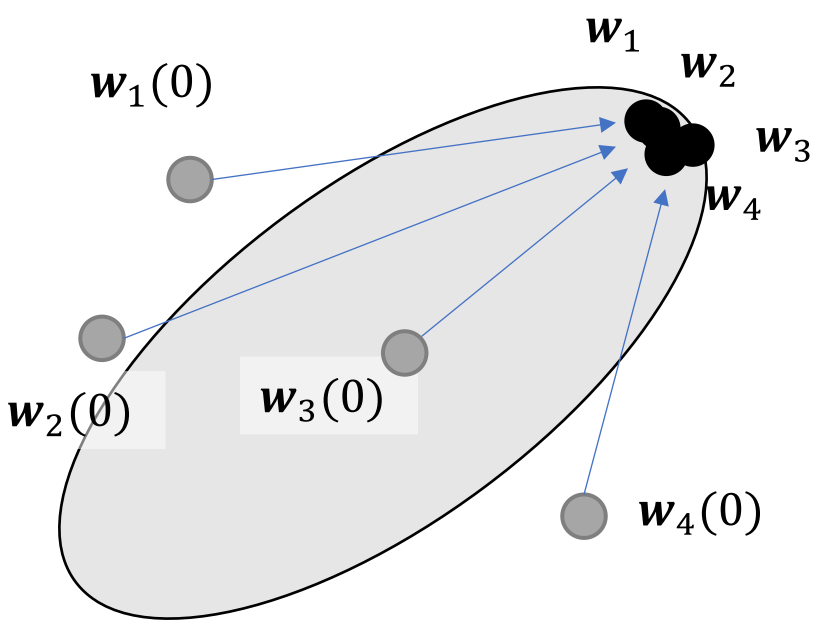
$\dot{\boldsymbol{w}} = P_{\boldsymbol{w}}^{\perp} A(\boldsymbol{w})\boldsymbol{w}$

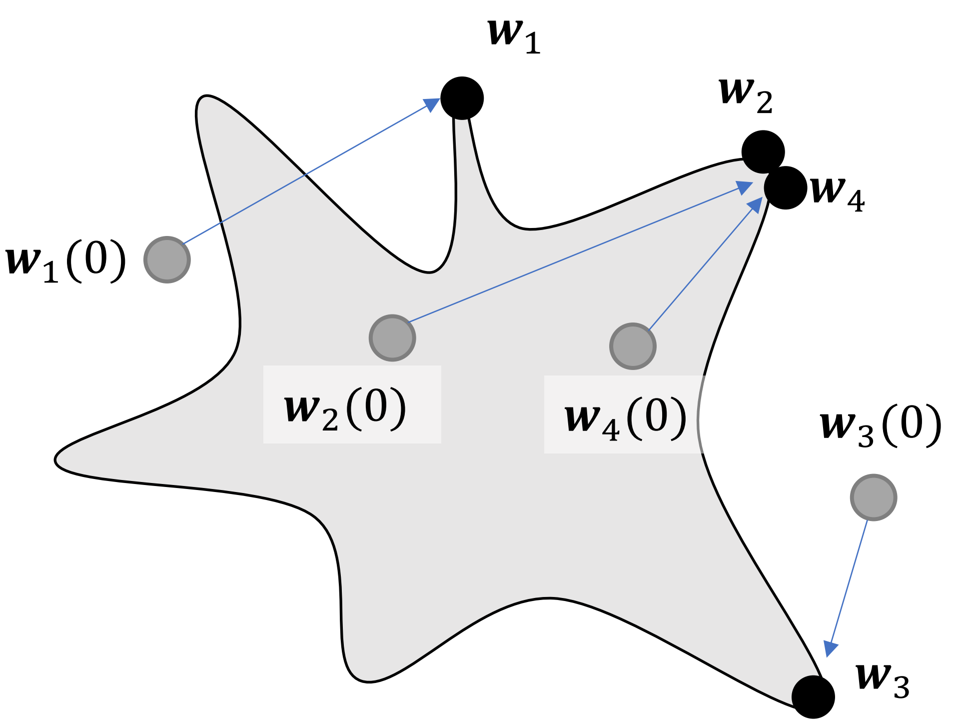


$\phi_1(\boldsymbol{w})$: Largest eigenvector of $A(\boldsymbol{w}) = \mathbb{C}_\alpha[\tilde{\boldsymbol{x}}^{\boldsymbol{w}}]$

Multiple largest eigenvectors!

1-layer multiple node nonlinear network





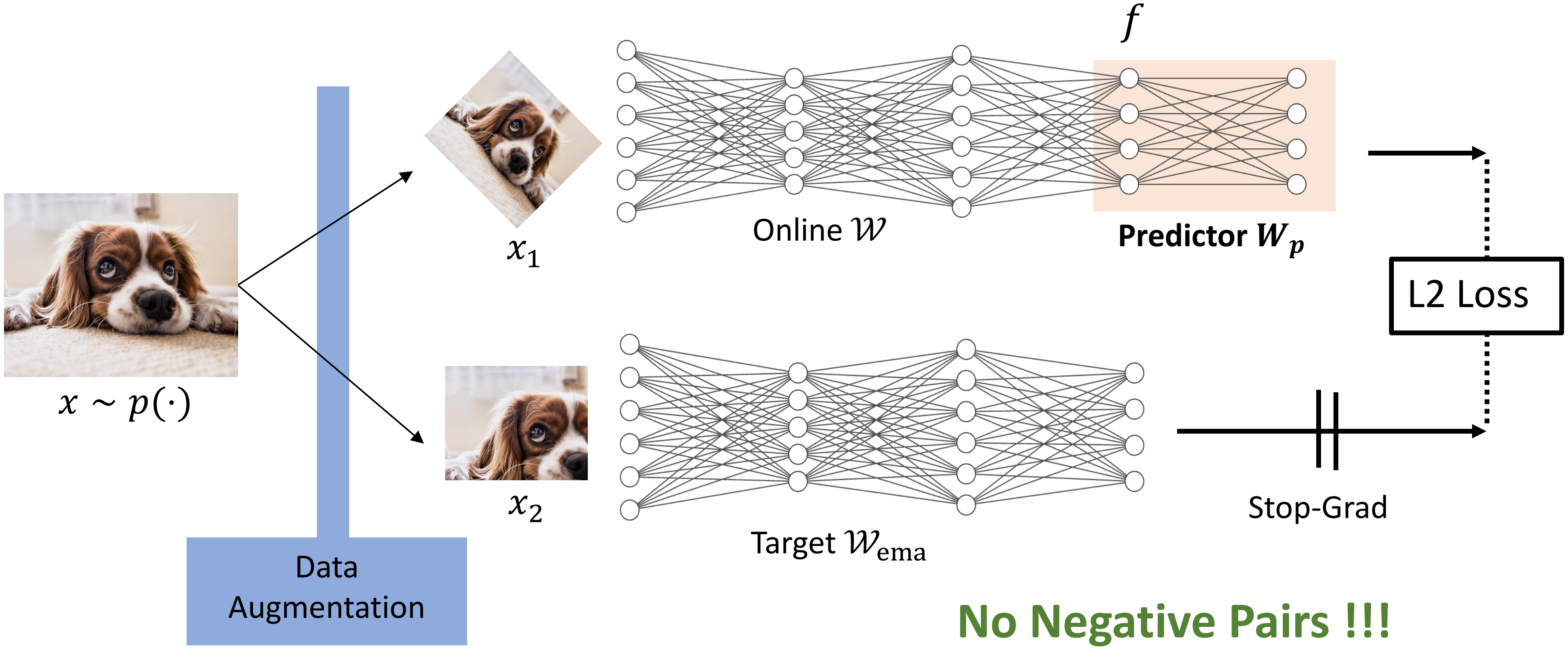
Linear model

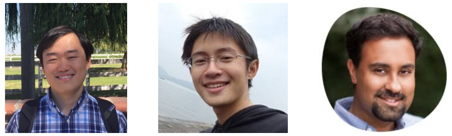
1. Every $\boldsymbol{w}_k$ converges to the global maximal eigenvector
2. More nodes do NOT help.

Nonlinear model

1. Each $\boldsymbol{w}_k$ can converge to different patterns
2. More nodes with diverse initialization learn more patterns!

Why Non-contrastive Learning doesn't collapse?





DirectPred [**Y. Tian** et al, Understanding Self-Supervised Learning Dynamics without Contrastive Pairs, ***ICML'21 Outstanding Paper Honorable Mentions***]

No Predictor / No Stop-Gradient do not work

If there is no EMA ($W = W_a$), then the dynamics becomes:

No Predictor

$$\dot{W} = -(X' + \eta I)W$$

PSD matrix

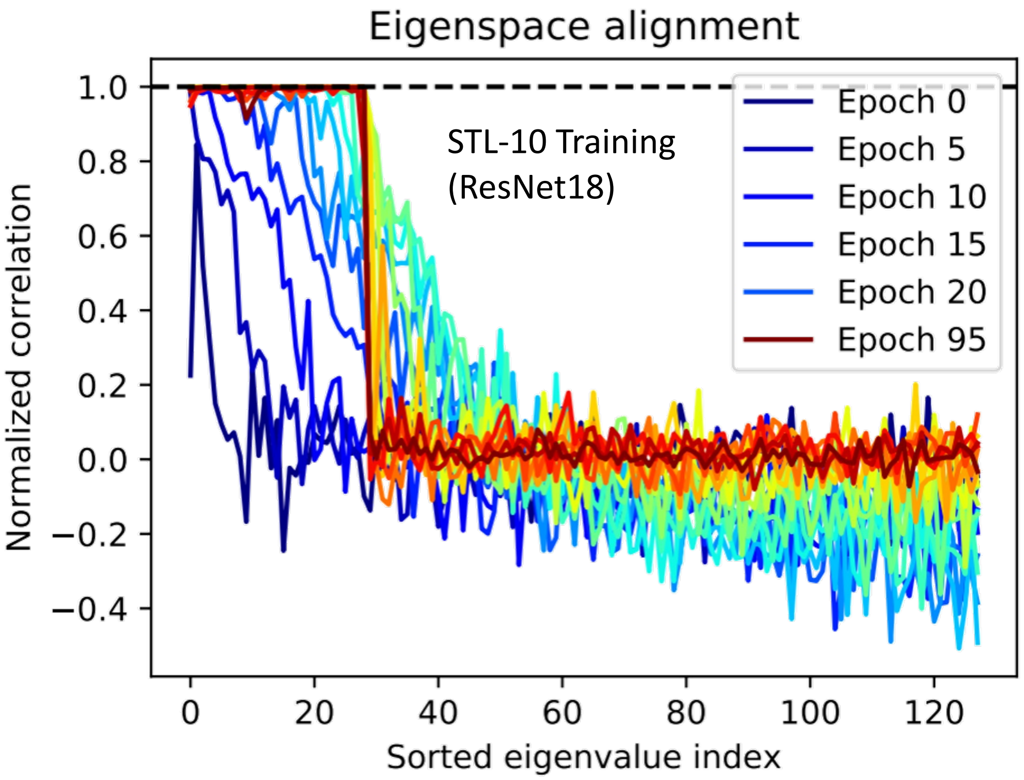
No Stop-Gradient (Here $\widetilde{W_p} := W_p - I$)

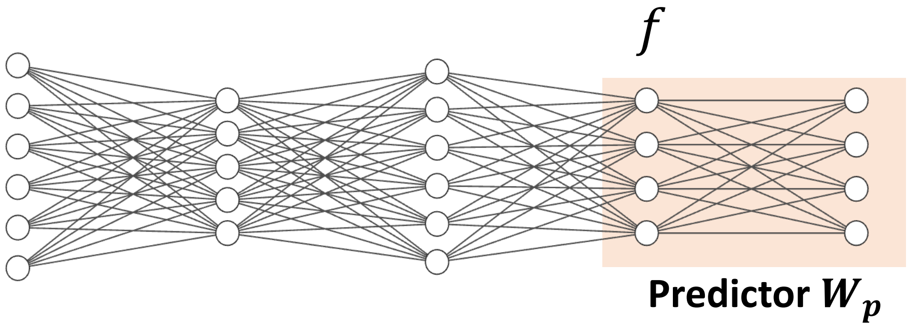
$$\frac{\mathrm{d}}{\mathrm{d}t}\mathrm{vec}(W) = -\left[X' \otimes (W_p^\intercal W_p + I) + X \otimes \tilde{W}_p^\intercal \tilde{W}_p + \eta I_{n_1 n_2}\right] \mathrm{vec}(W)$$

PSD matrix

In both cases, $W \rightarrow 0$

Why Non-contrastive Learning doesn't collapse?





Theorem 3: Under certain conditions,

$$FW_p - W_pF \to 0 \text{ when } t \to +\infty$$

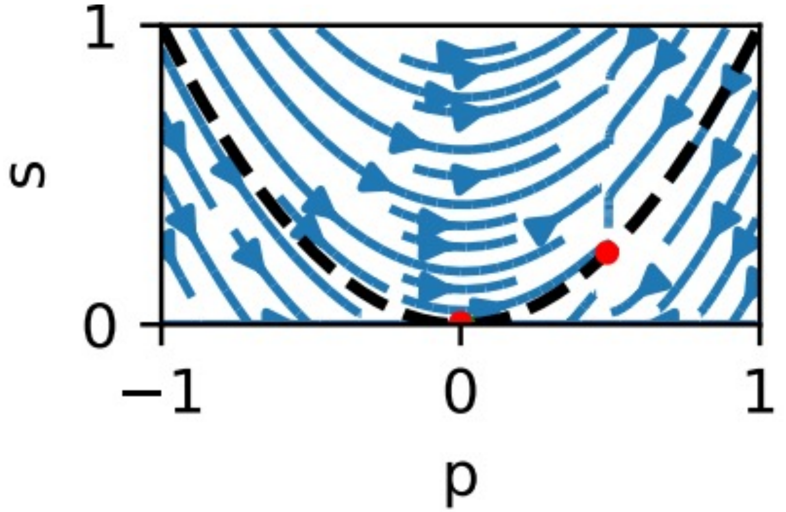
and the eigenspace of W_p and F gradually **aligns**.

$F := \mathbb{E}[\boldsymbol{f}\boldsymbol{f}^T]$ is the statistics of the input before W_p

Why Non-contrastive Learning doesn't collapse?

When eigenspace aligns, the dynamics becomes decoupled:

$$\begin{aligned} \dot{p}_j &= \alpha_p s_j \left[\tau - (1+\sigma^2)p_j\right] - \eta p_j \\ \dot{s}_j &= 2p_j s_j \left[\tau - (1+\sigma^2)p_j\right] - 2\eta s_j \\ s_j \dot{\tau} &= \beta(1-\tau)s_j - \tau \dot{s}_j/2. \end{aligned}$$



Where p_j and s_j are eigenvalues of W_p and F

Invariance holds: $s_j(t) = \alpha_p^{-1} p_j^2(t) + e^{-2\eta t} c_j$

Why Non-contrastive Learning doesn’t collapse?

1D dynamics of the eigenvalue p_j of W_p:

$$\dot{p}_j = p_j^2 \left[\tau(t) - (1 + \sigma^2) p_j\right] - \eta p_j$$

EMA

Variance due to data augmentation

Weight Decay

Why Non-contrastive Learning doesn't collapse?

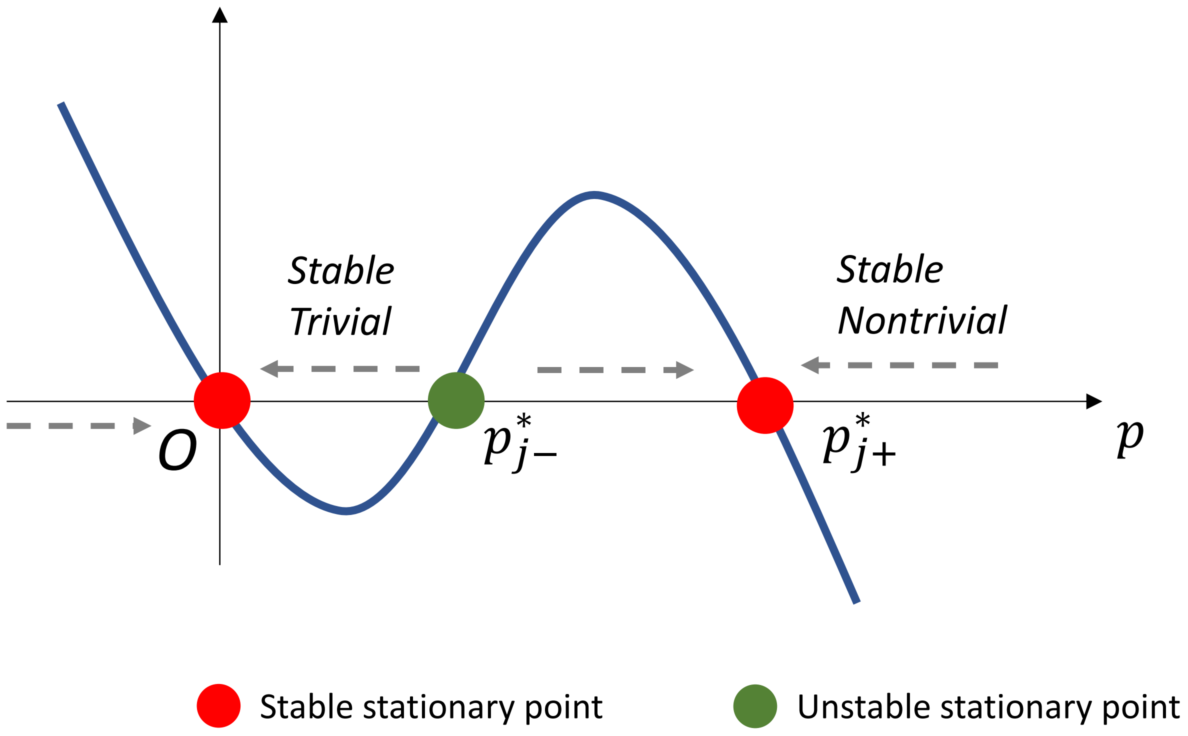
1D dynamics of the eigenvalue p_j of W_p:

$$\dot{p}_j = p_j^2\left[\tau(t) - (1+\sigma^2)p_j\right] - \eta p_j$$

EMA

Variance due to data augmentation

Weight Decay



Why Non-contrastive Learning doesn't collapse?

1D dynamics of the eigenvalue p_j of W_p:

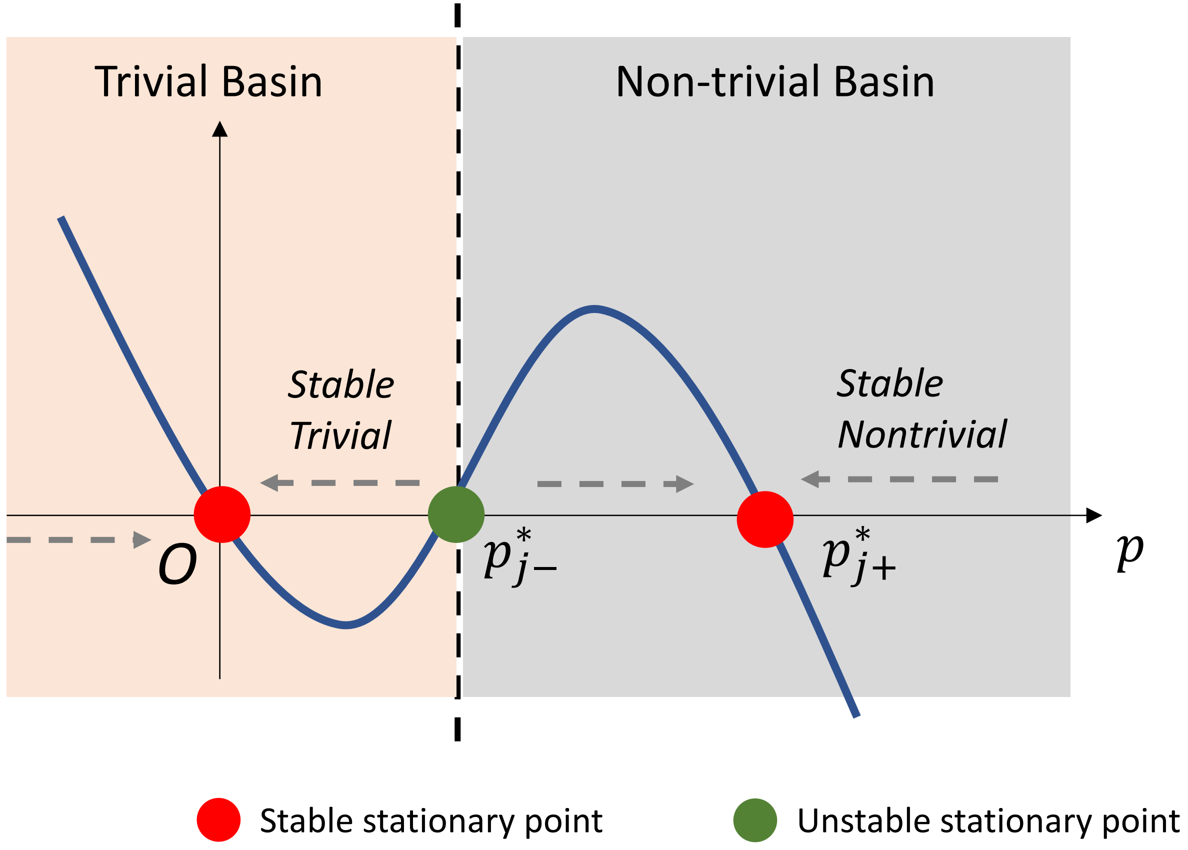
$$\dot{p}_j = p_j^2 \left[\tau(t) - (1+\sigma^2)p_j\right] - \eta p_j$$

EMA

Variance due to data augmentation

Weight Decay

$$p_{j-}^* = \frac{\tau - \sqrt{\tau^2 - 4\eta(1+\sigma^2)}}{2(1+\sigma^2)} \sim \frac{\eta}{\tau}$$



DirectPred

- Directly setting linear W_p rather than relying on gradient update.

1. Estimate $\hat{F} = \rho\hat{F} + (1-\rho)E[\boldsymbol{f}\boldsymbol{f}^T]$
2. Eigen-decompose $\hat{F} = \hat{U}\Lambda_F\hat{U}^T$, $\Lambda_F = \text{diag}\,[s_1, s_2, \ldots, s_d]$
3. Set W_p following the invariance:

$$p_j = \sqrt{s_j} + \epsilon \max_j s_j, \quad W_p = \hat{U}\text{diag}[p_j]\hat{U}^\intercal$$

Guaranteed Eigenspace Alignment ☺

Performance of **DirectPred** on STL-10/CIFAR-10

Downstream Classification Top-1	Number of epochs 100	300	500
STL-10			
DirectPred	$\mathbf{77.86 \pm 0.16}$	78.77 ± 0.97	78.86 ± 1.15
DirectPred (freq=5)	77.54 ± 0.11	$\mathbf{79.90 \pm 0.66}$	$\mathbf{80.28 \pm 0.62}$
SGD baseline	75.06 ± 0.52	75.25 ± 0.74	75.25 ± 0.74
CIFAR-10			
DirectPred	$\mathbf{85.21 \pm 0.23}$	$\mathbf{88.88 \pm 0.15}$	89.52 ± 0.04
DirectPred (freq=5)	84.93 ± 0.29	88.83 ± 0.10	$\mathbf{89.56 \pm 0.13}$
SGD baseline	84.49 ± 0.20	88.57 ± 0.15	89.33 ± 0.27

Performance of **DirectPred** on ImageNet

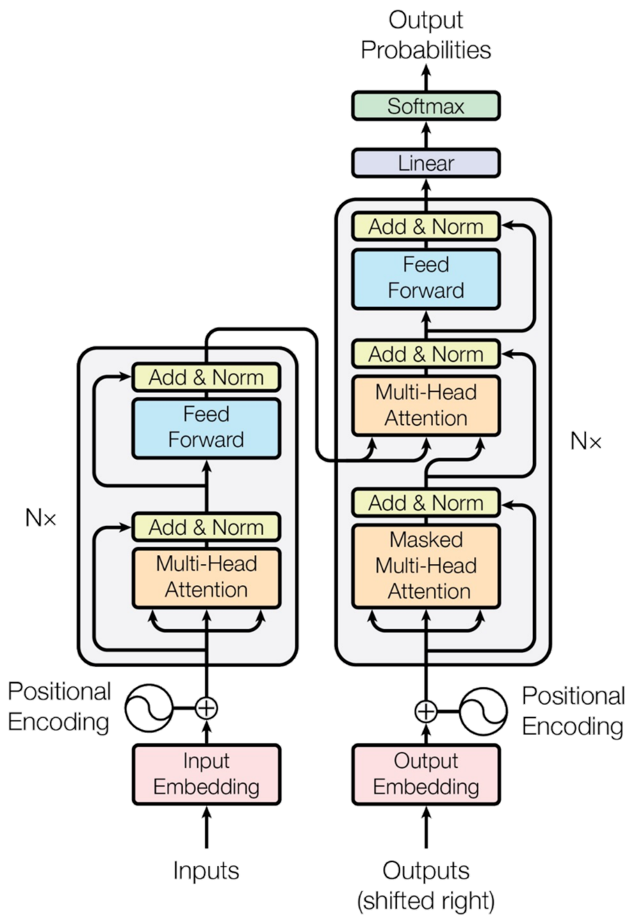
Downstream classification (ImageNet):

BYOL variants	*Accuracy (60 ep)*		*Accuracy (300 ep)*	
	Top-1	Top-5	Top-1	Top-5
2-layer predictor*	**64.7**	**85.8**	**72.5**	90.8
linear predictor	59.4	82.3	69.9	89.6
DirectPred	64.4	**85.8**	72.4	**91.0**

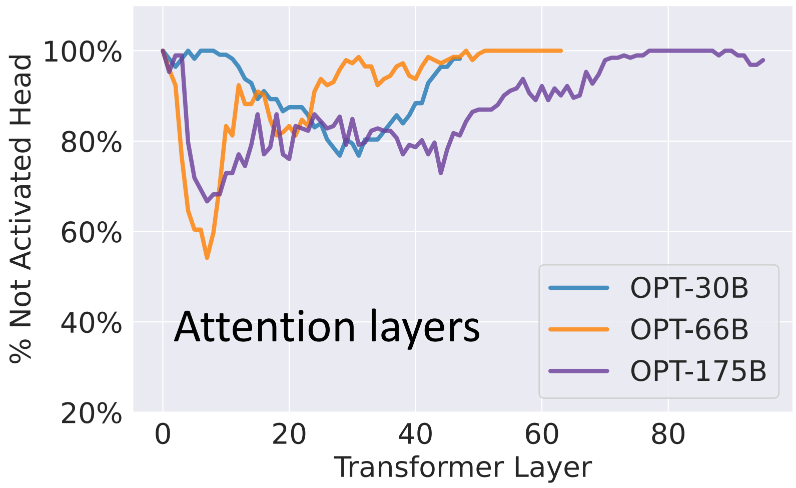
* 2-layer predictor is BYOL default setting.

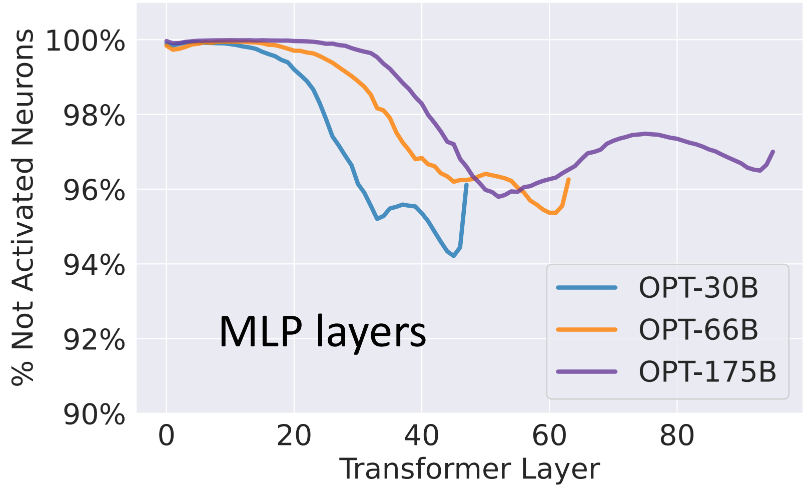
DirectPred using linear predictor is better than SGD with linear predictor, and is comparable with 2-layer predictor.

Let's check Collapsing ("sparsity") in Transformers!



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

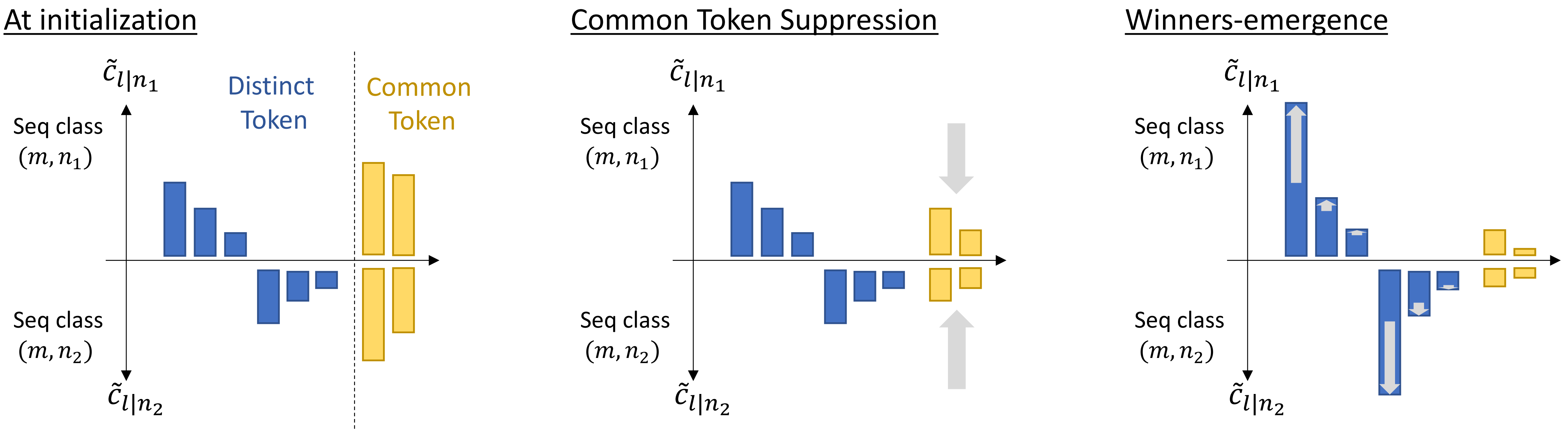




[Z. Liu et al, *Deja vu: Contextual sparsity for efficient LLMs at inference time*, ICML'23 (oral)]

Representation Collapses ("sparsity") in Self-Attention

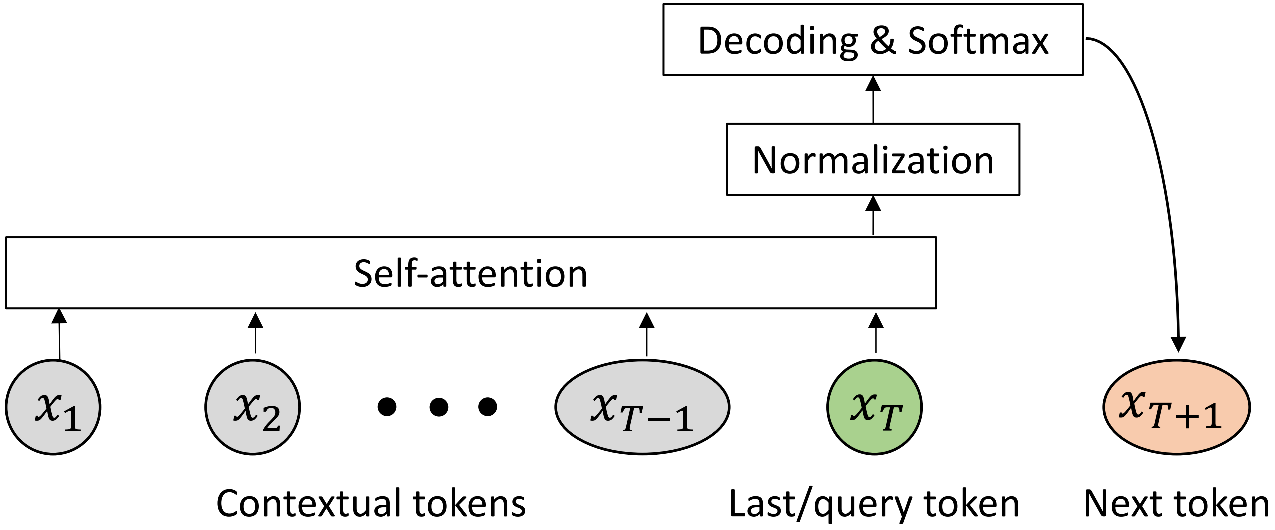
One layer Transformer, linear MLP



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

Understanding Attention in 1-layer Setting

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



Self-attention

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

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

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

Objective:

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

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

Reparameterization

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

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

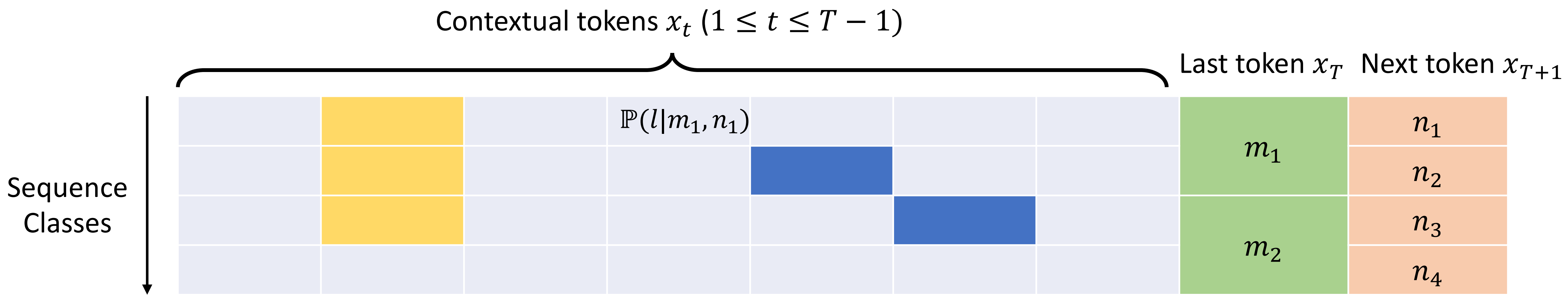
- Then the dynamics becomes easier to analyze

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 $1 \leq t \leq T$
$x_{T+1} \in [K]$
$K \ll M$



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

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

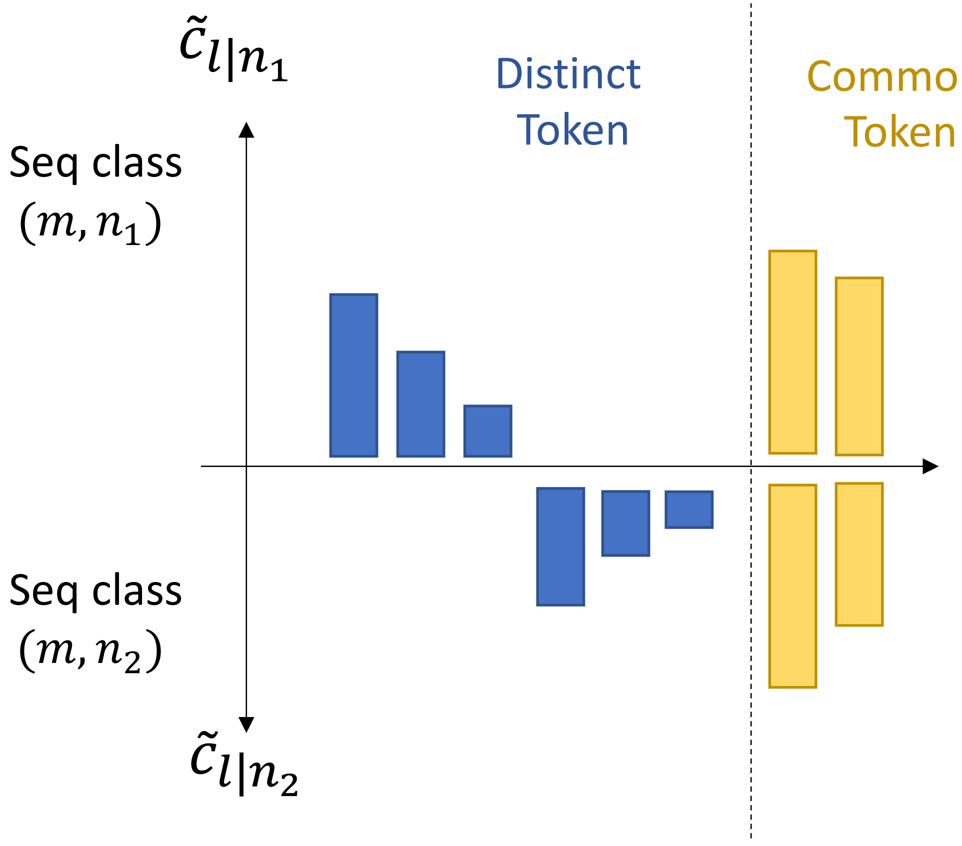
$\mathbb{P}(l|m,n) = \mathbb{P}(l|n)$ is the conditional probability of token l given last token $x_T = m$ and $x_{T+1} = n$

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

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

Overall Picture of the Training Dynamics

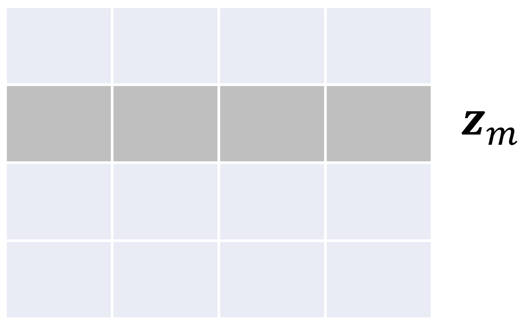
At initialization



Co-occurrence probability

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

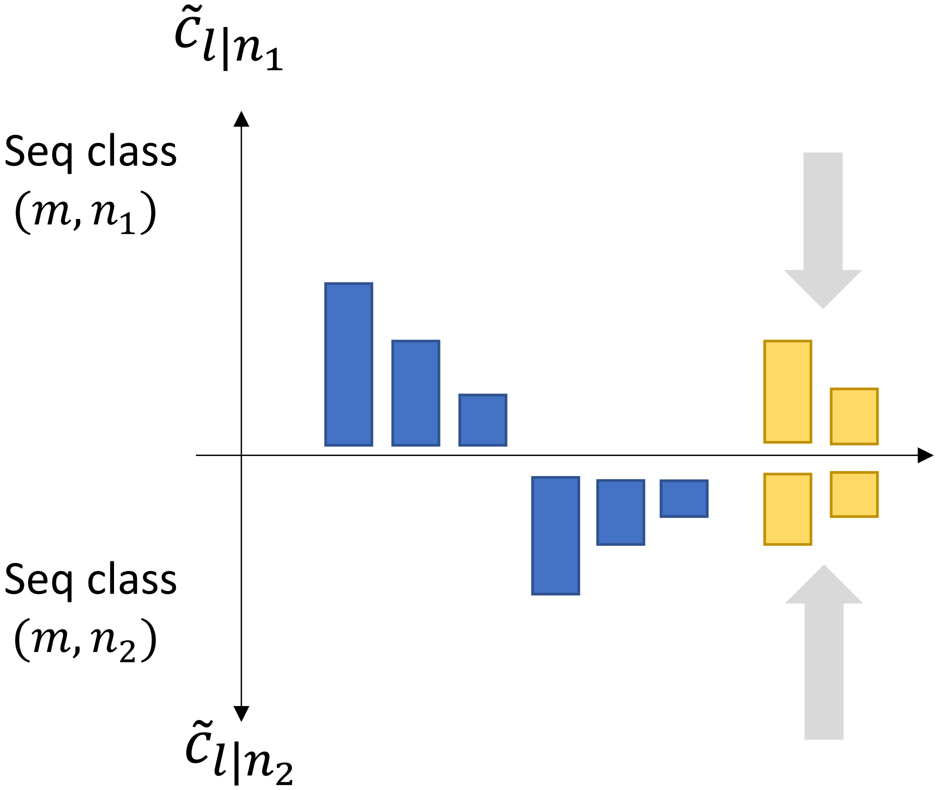
Initial condition: $z_{ml}(0) = 0$

$Z =$ 

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

Overall Picture of the Training Dynamics

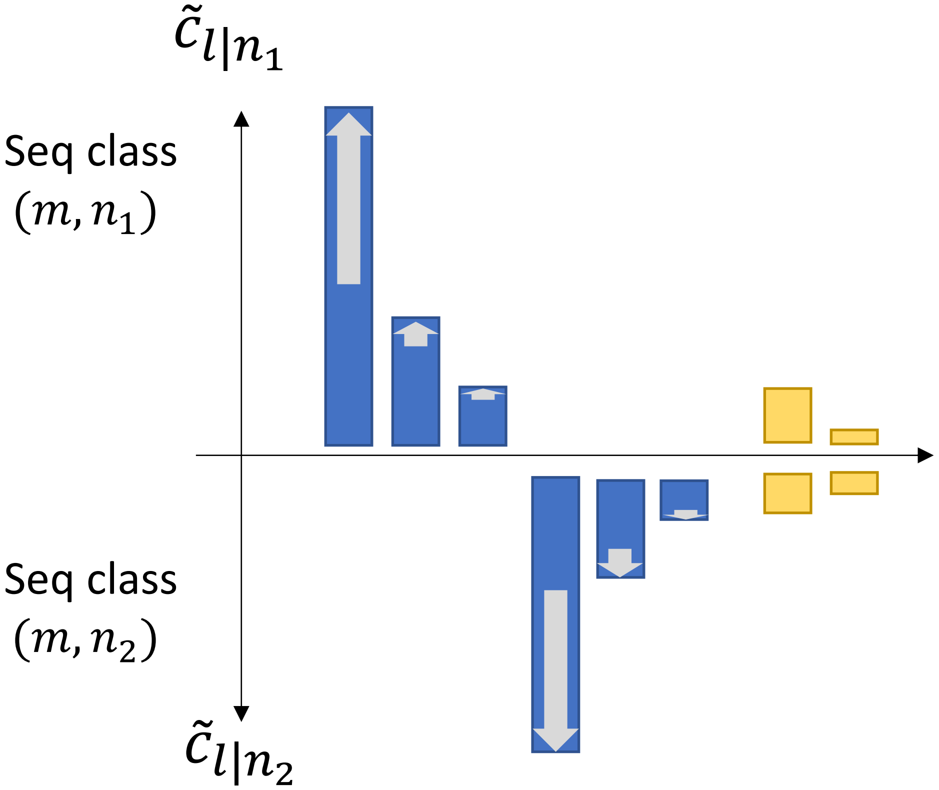
Common Token Suppression



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

Overall Picture of the Training Dynamics

Winners-emergence



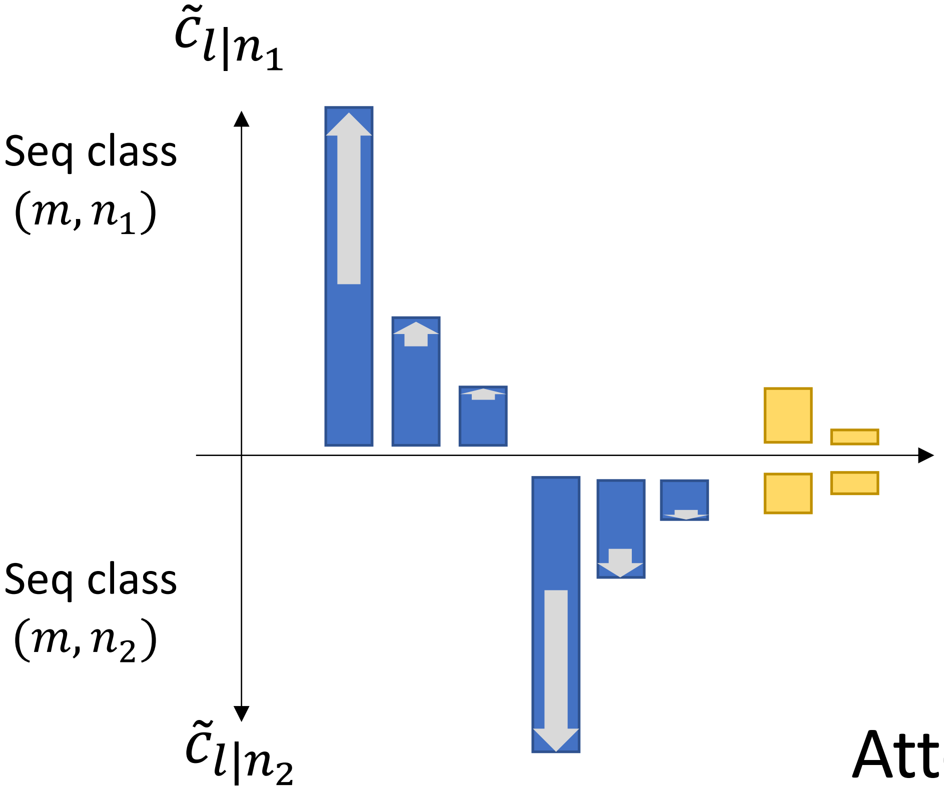
(a) $\dot{z}_{ml} < 0$, for common token l

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

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

Overall Picture of the Training Dynamics

Winners-emergence



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

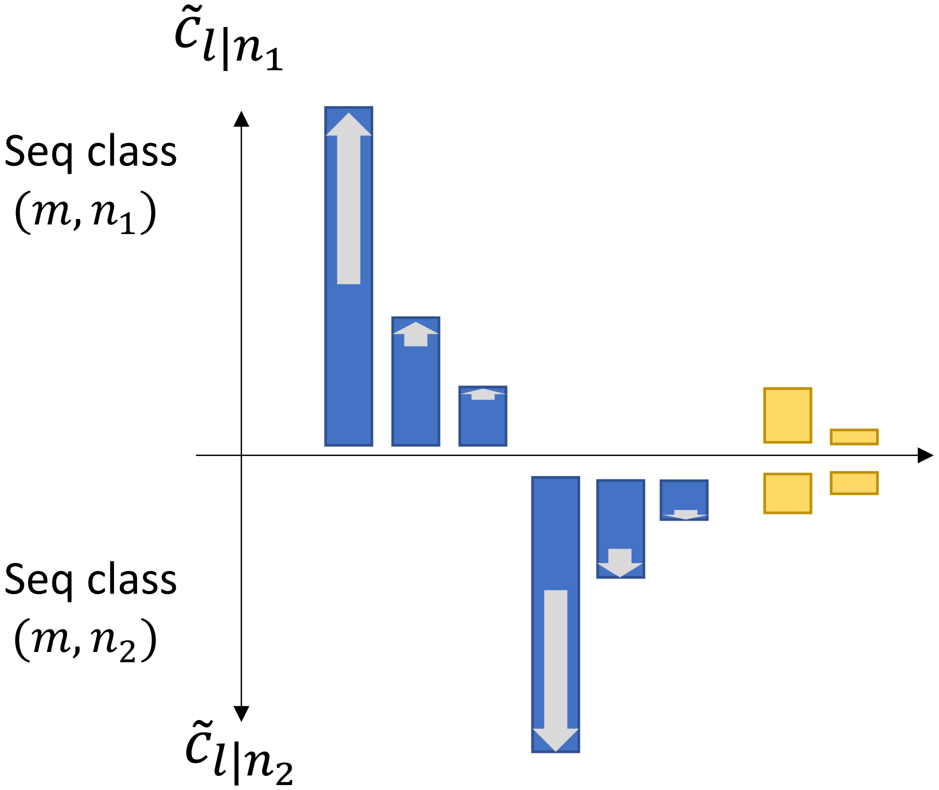
(b) $\dot{z}_{ml} > 0$, for distinct token l

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

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

Overall Picture of the Training Dynamics

Winners-emergence



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

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

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

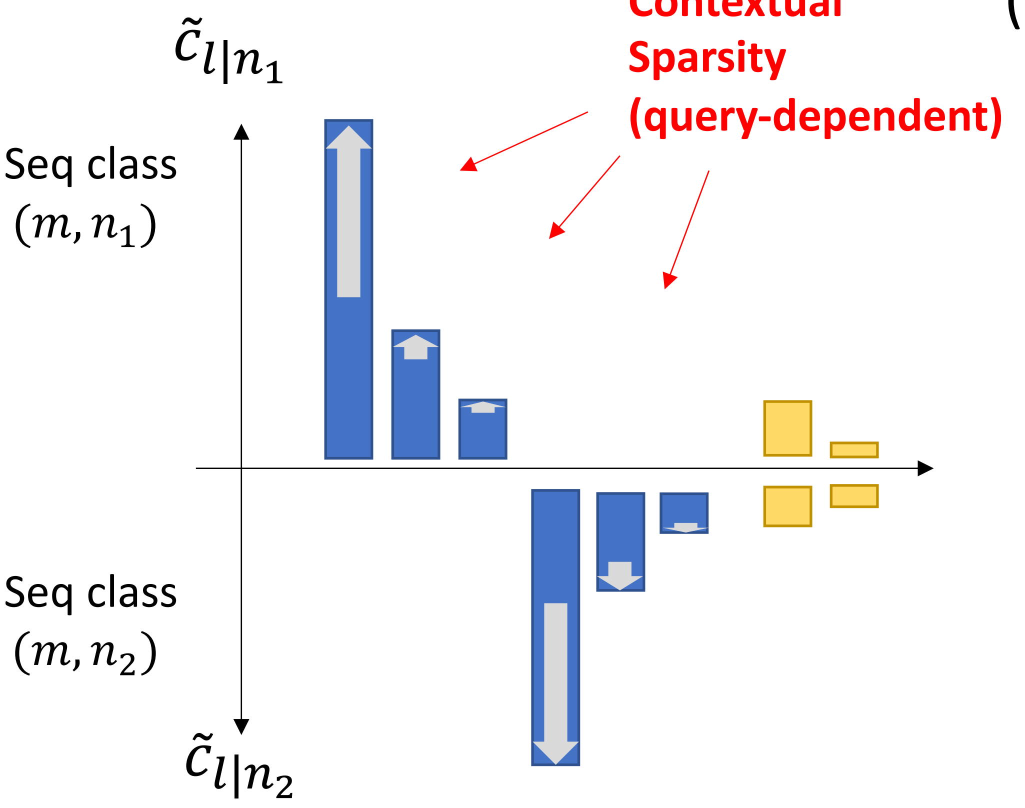
If l_0 is the dominant token: $r_{l_0/l|n}(0) > 0$ for all $l \neq l_0$ then

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

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

Overall Picture of the Training Dynamics

Winners-emergence



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

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

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

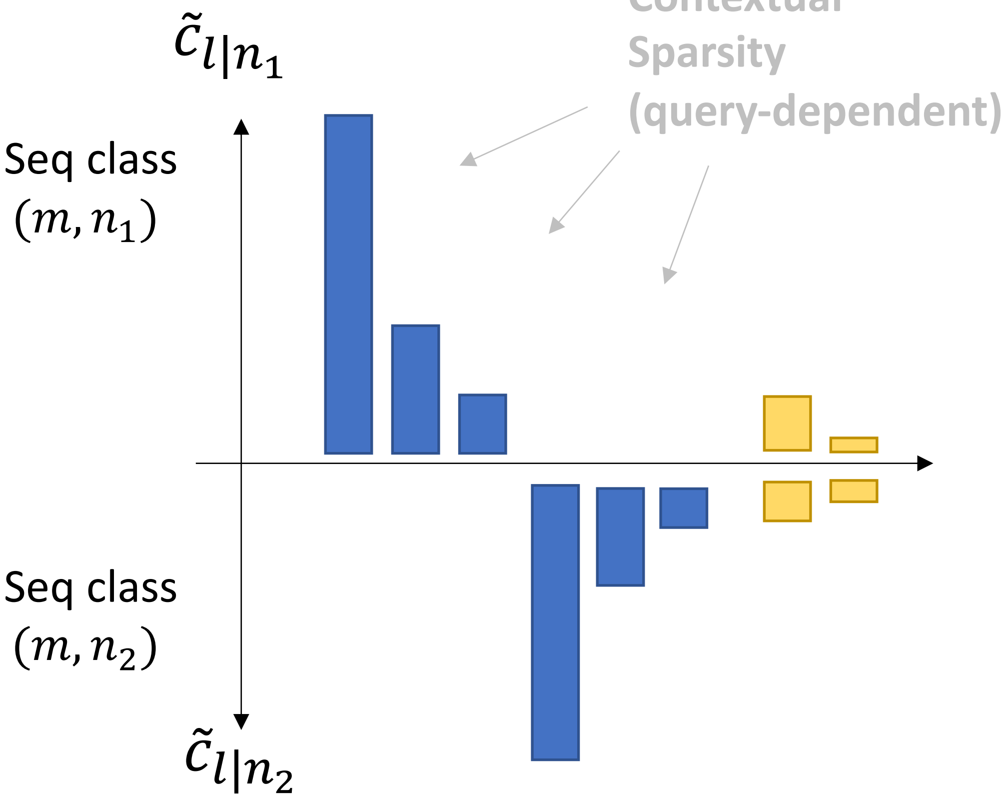
If l_0 is the dominant token: $r_{l_0/l|n}(0) > 0$ for all $l \neq l_0$ then

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

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

Overall Picture of the Training Dynamics

Attention frozen



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

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

Attention **scanning**:

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

Attention **snapping**:

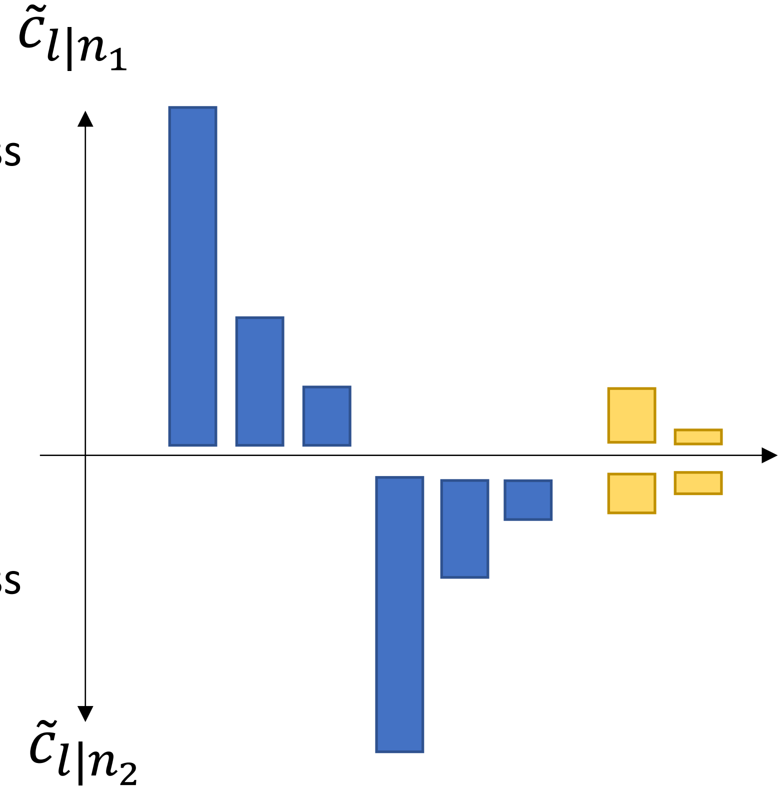
When $t \geq t_0 = O\left(\frac{2K\ln M}{\eta_Y}\right)$, $B_n(t) = O(\ln\ln t)$

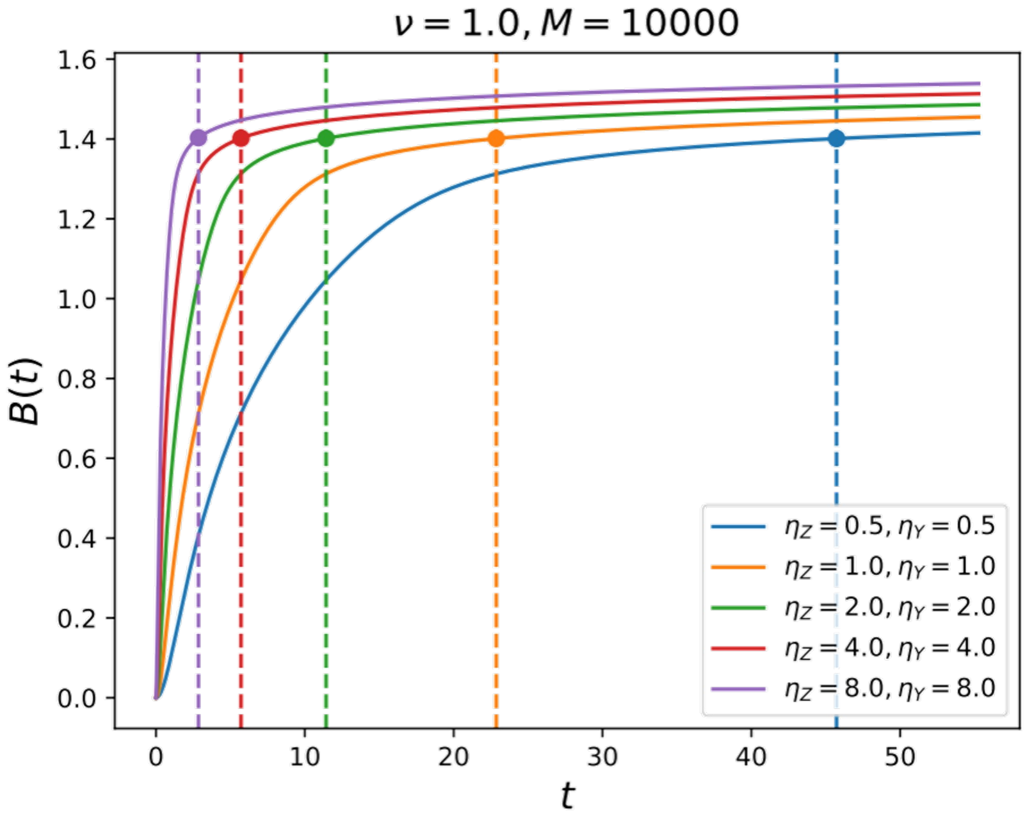
(1) η_z and η_Y are large, $B_n(t)$ is large and attention is sparse

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

Overall Picture of the Training Dynamics

<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)

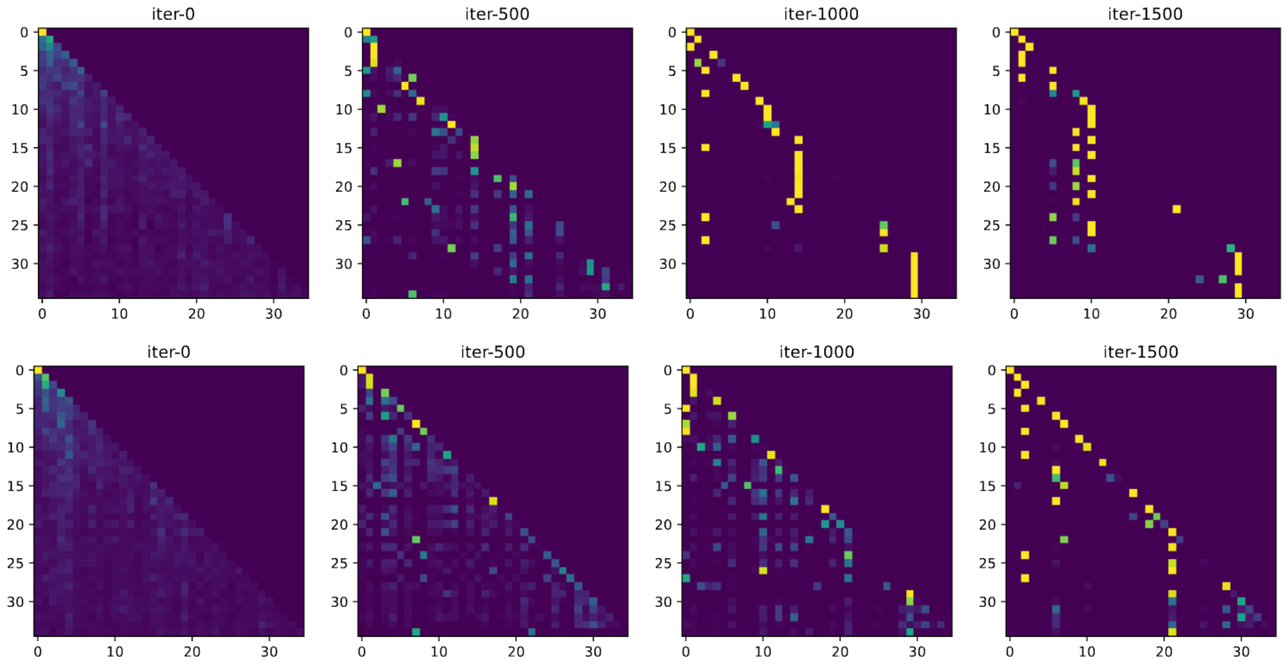


Figure 7: Attention patterns in the lowest self-attention layer for 1-layer (top) and 3-layer (bottom) Transformer trained on WikiText2 using SGD (learning rate is 5). Attention becomes sparse over training.

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]

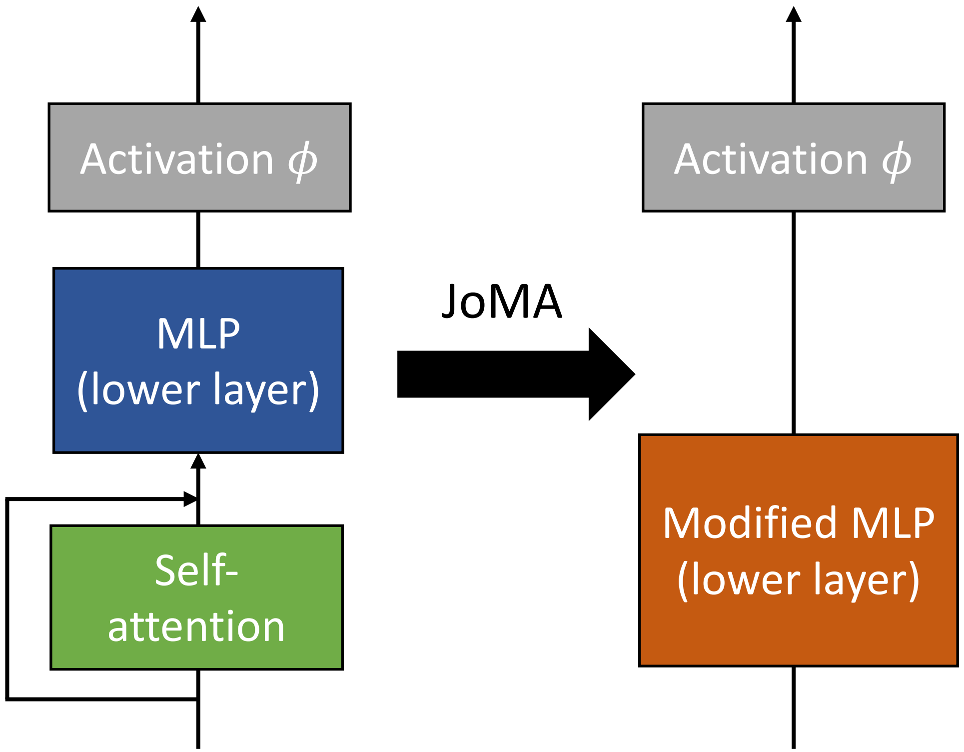
How to get rid of the assumptions?

- 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

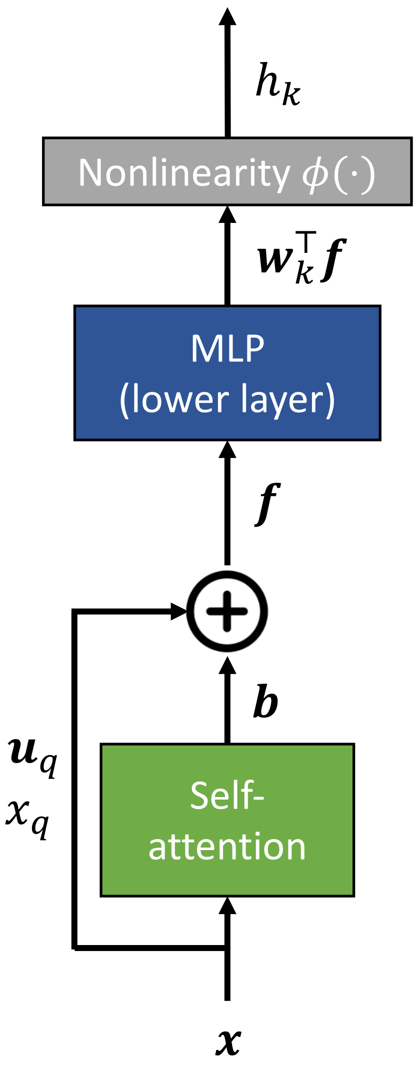


Main Contributions:

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”

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

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

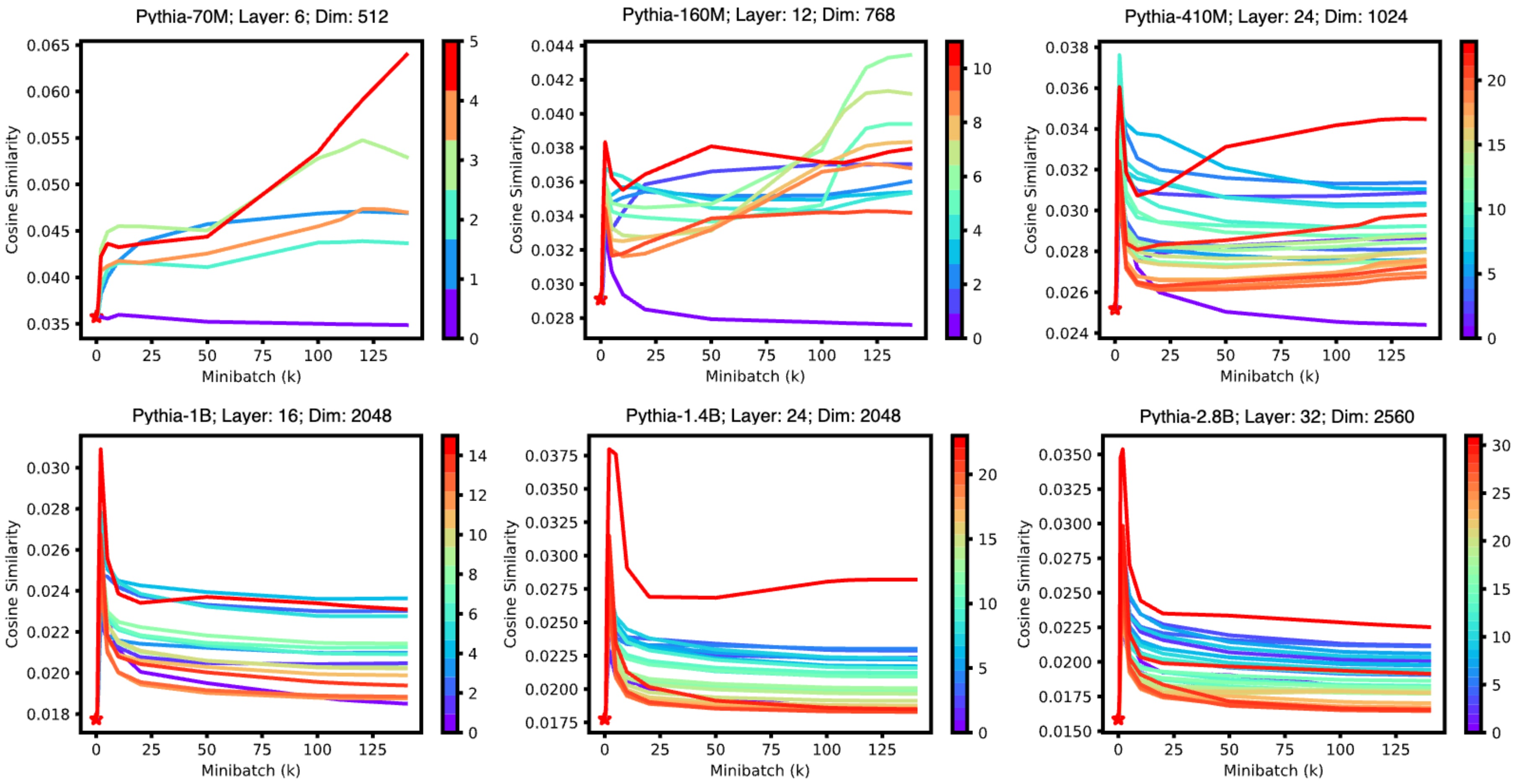
U_C and $\boldsymbol{u}_q$ are embeddings

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

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

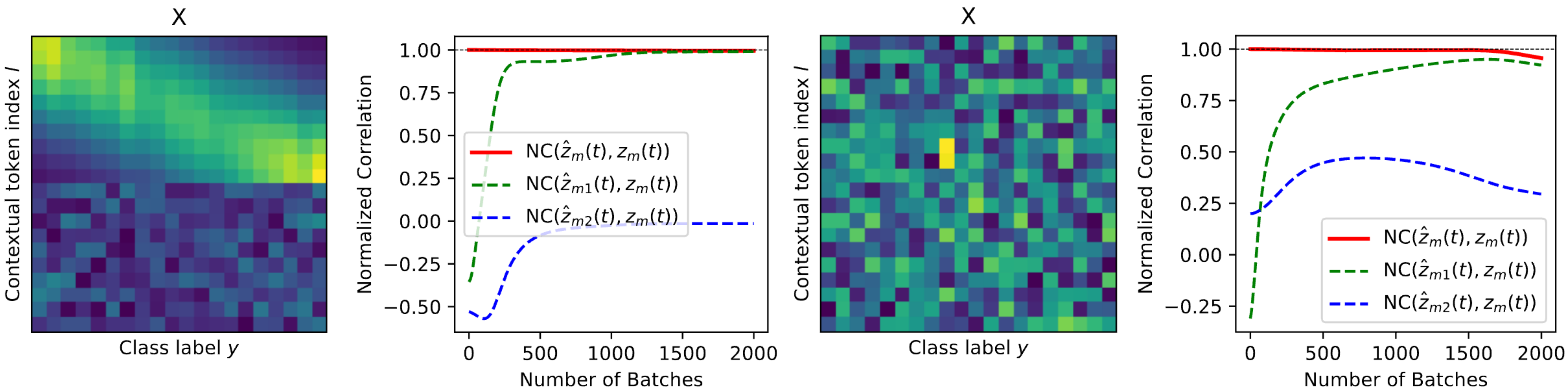
Theorem 1 (JoMA). *Let* $\boldsymbol{v}_k := U_C^\top \boldsymbol{w}_k$*, then the dynamics of Eqn. 3 satisfies the invariants:*

- *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

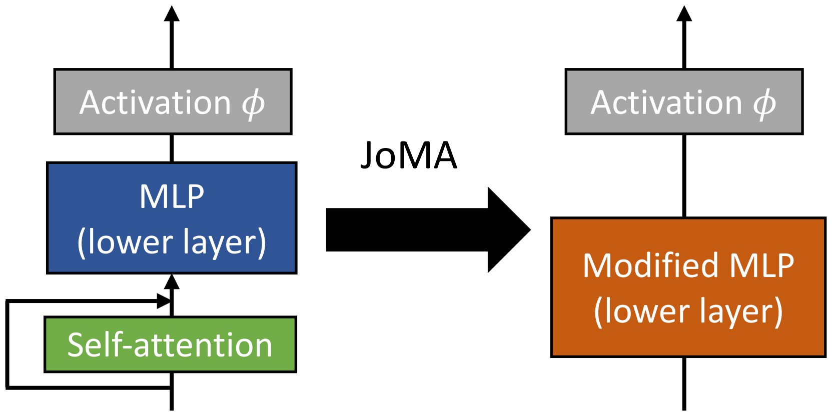


$\boldsymbol{z}_m(t)$: Real attention logits

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

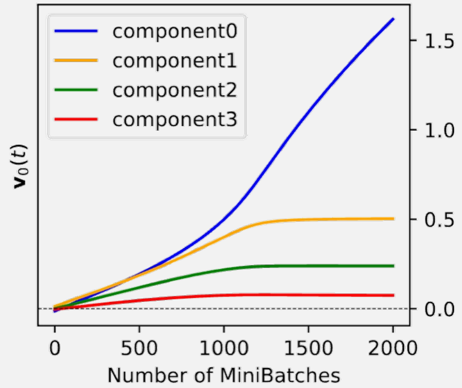
$$\hat{\boldsymbol{z}}_m(t) = \underbrace{\frac{1}{2}\sum_k \boldsymbol{v}_k^2(t)}_{\hat{\boldsymbol{z}}_{m1}(t)} - \underbrace{\|\boldsymbol{v}_k(t)\|_2^2 \overline{\boldsymbol{b}}_m}_{\hat{\boldsymbol{z}}_{m2}(t)} + \boldsymbol{c}$$

Implication of Theorem 1



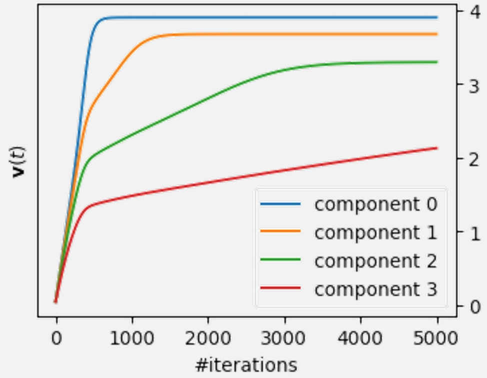
Key idea: folding self-attention into MLP
→ A Transformer block becomes a modified MLP

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

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

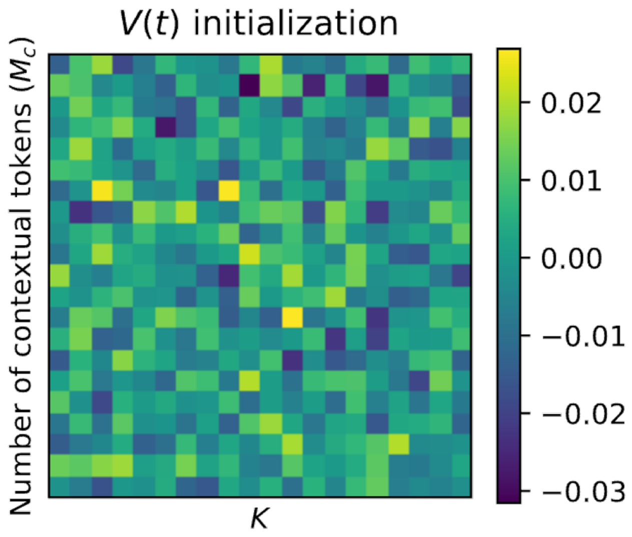
JoMA for Linear Activation

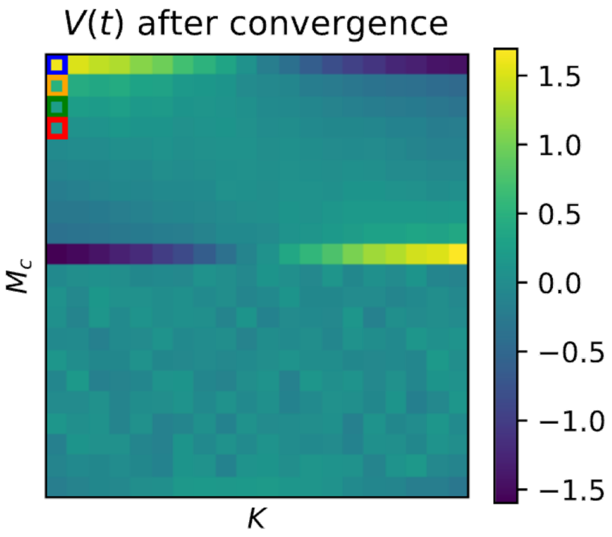
Theorem 2

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

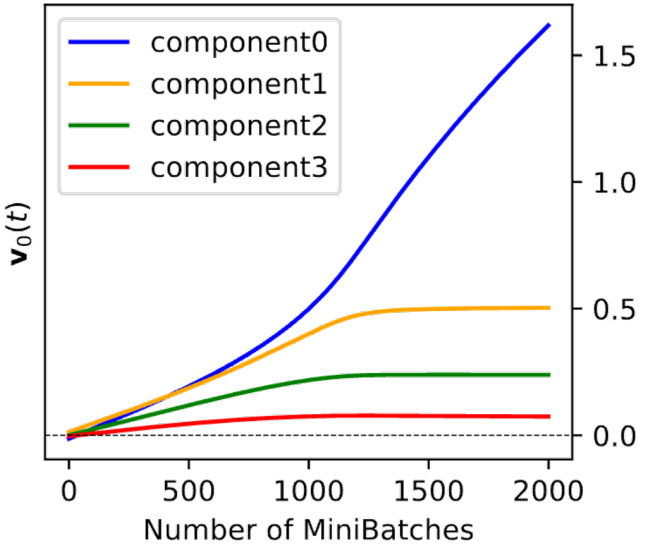
$$\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}\mathrm{d}t \in [-1,1]$$

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 ($K > 1$)?

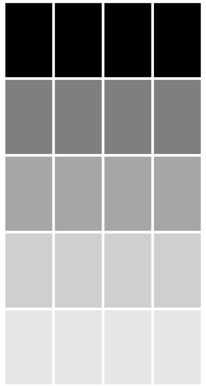
- $V = U_C^\top W \in \mathbb{R}^{M_c \times K}$ and the dynamics becomes

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

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

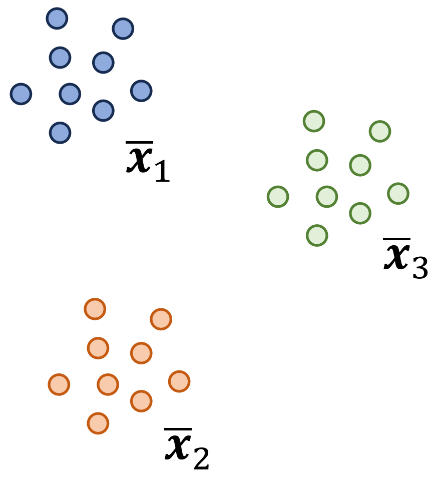
JoMA for Nonlinear Activation

Theorem 3

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

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

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



What does the dynamics look like?

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

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

JoMA for Nonlinear activation

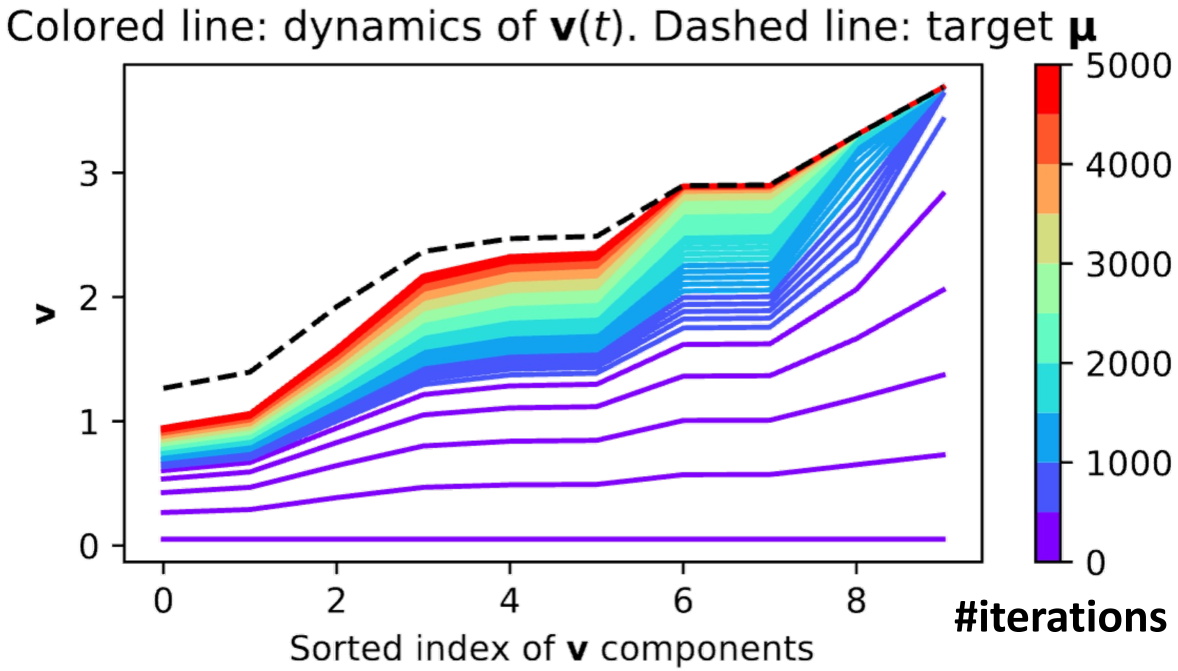
$\dot{\boldsymbol{v}} = (\boldsymbol{\mu} - \boldsymbol{v}) \circ \exp\left(\frac{\boldsymbol{v}^2}{2}\right)$	Nonlinear Modified MLP (lower layer)

Theorem 4

Salient components grow much faster than non-salient ones:

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

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

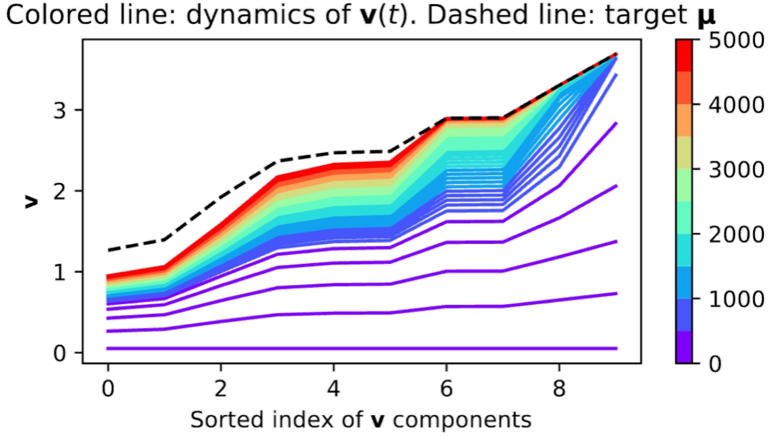


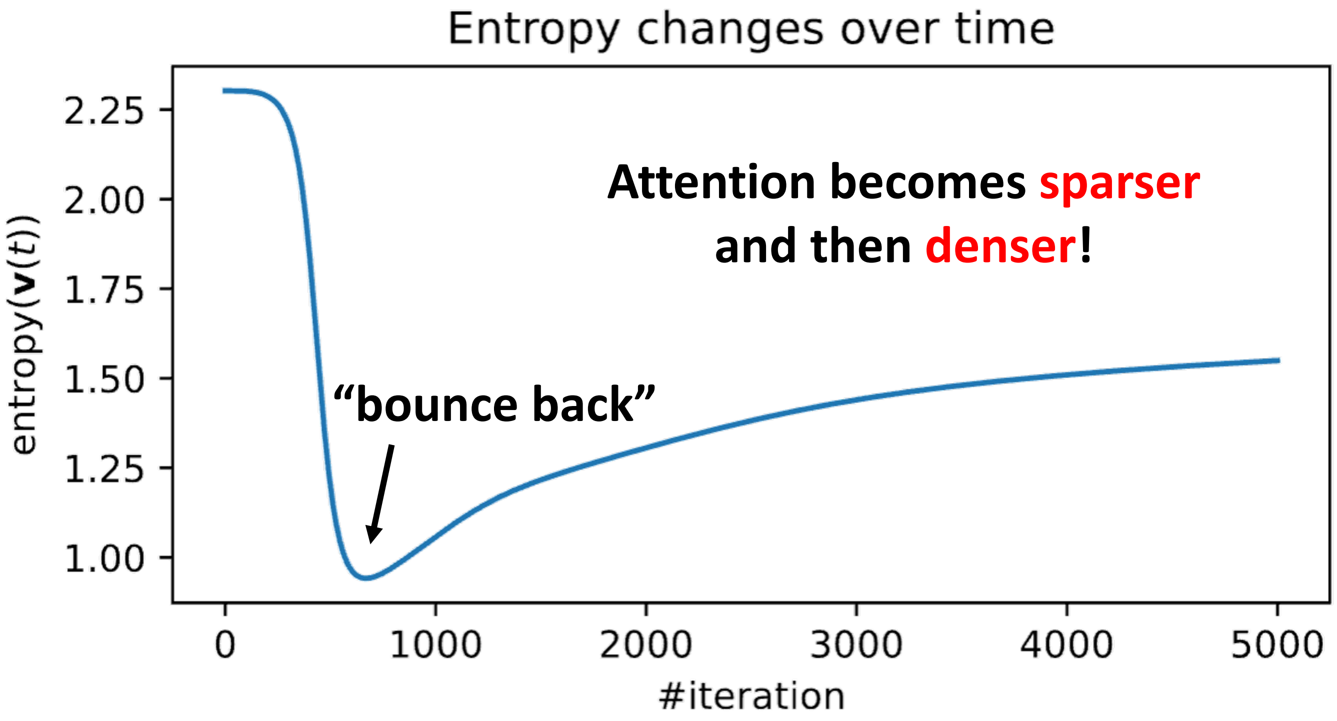
JoMA for Nonlinear activation

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

Nonlinear

Modified MLP (lower layer)

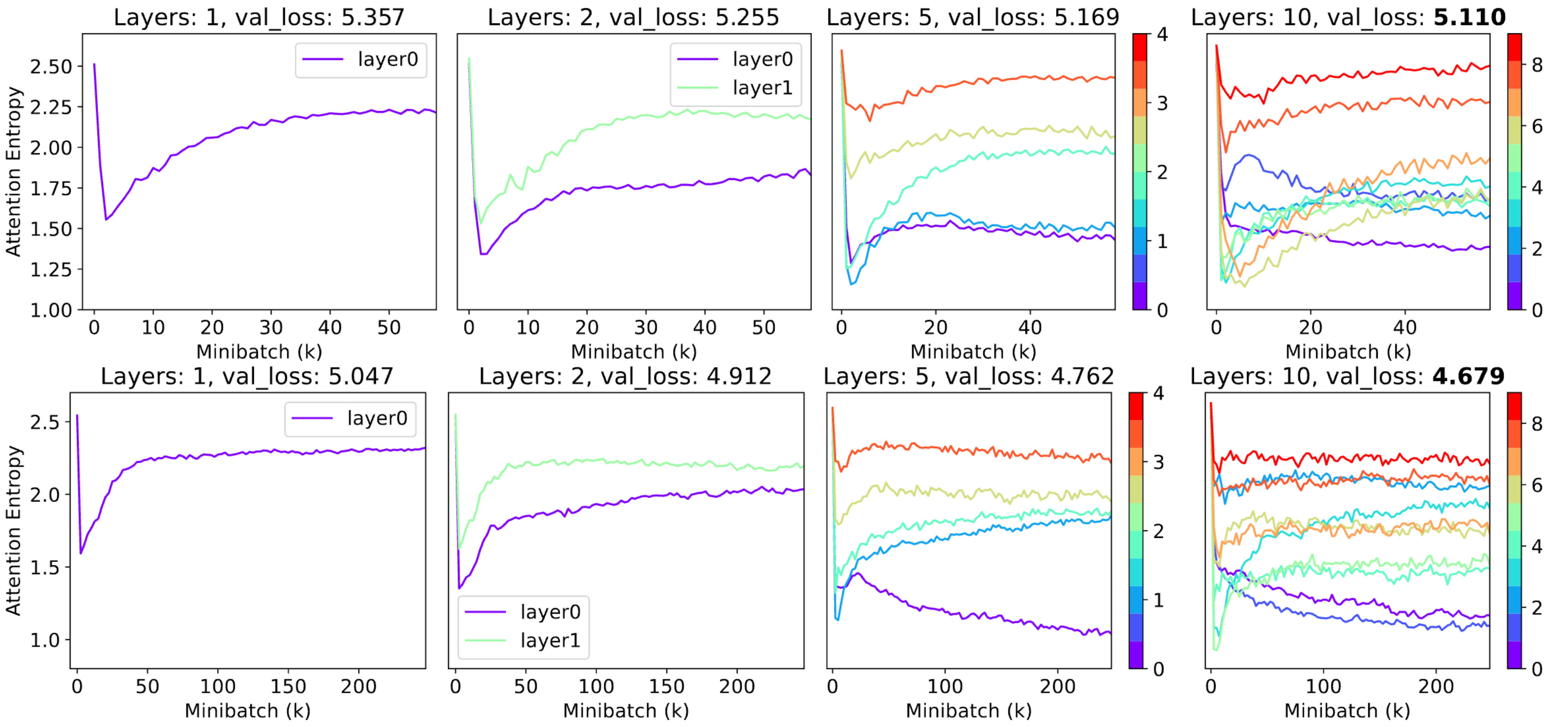




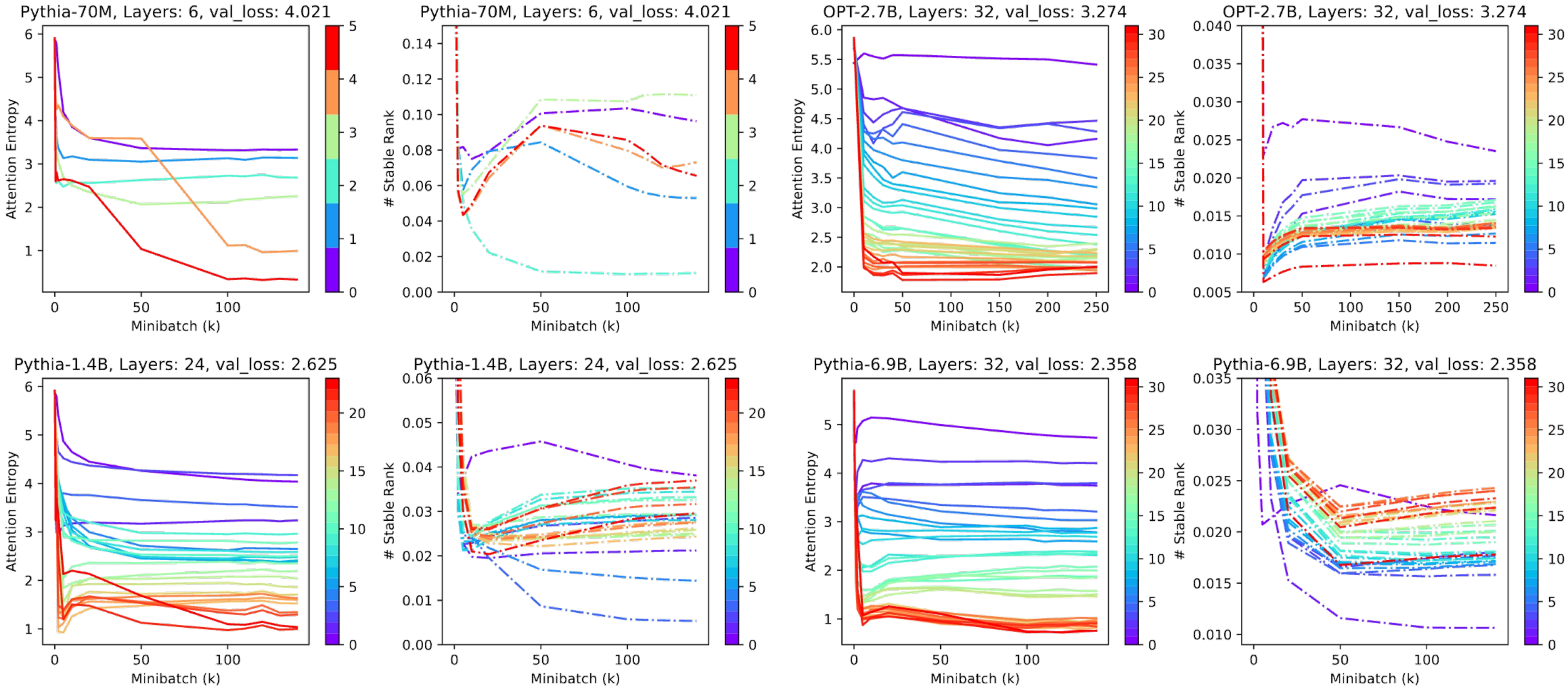
Real-world Experiments

Wikitext2

Wikitext103



Real-world Experiments



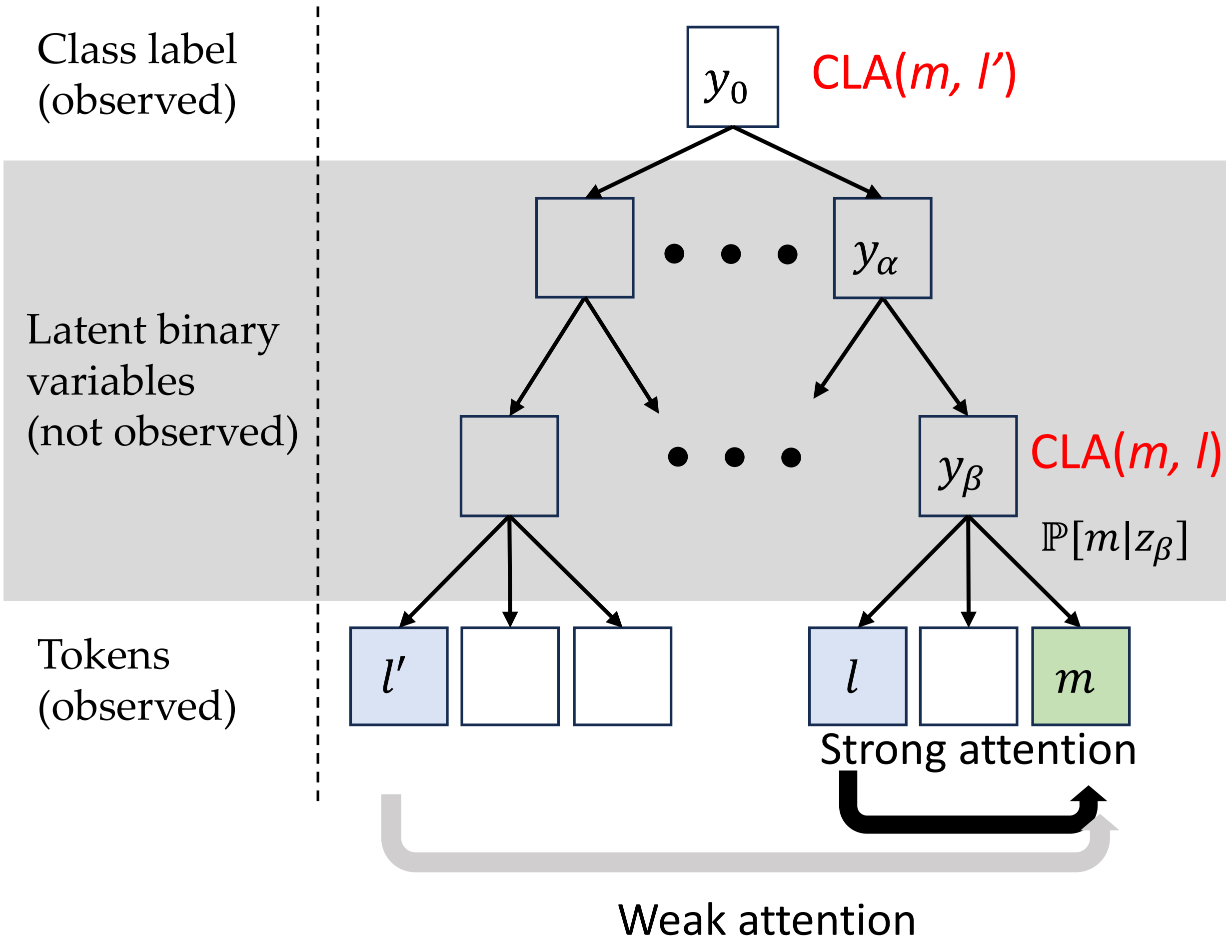
Stable Rank of the lower layer of MLP shows the “bouncing back” effects as well.

Why is this “bouncing back” property useful?

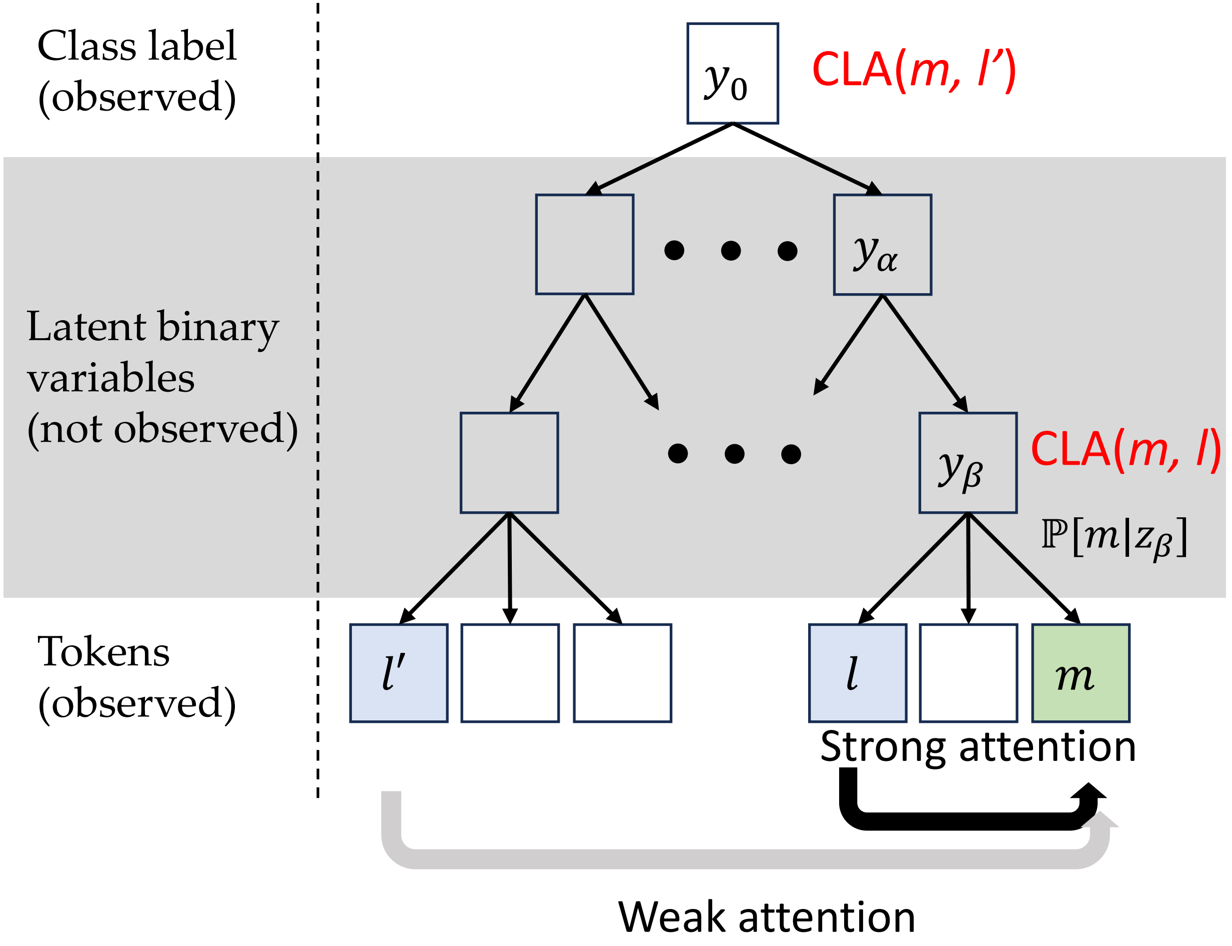
It seems that it only slows down the training??

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

Data Hierarchy & Multilayer Transformer



Data Hierarchy & Multilayer Transformer



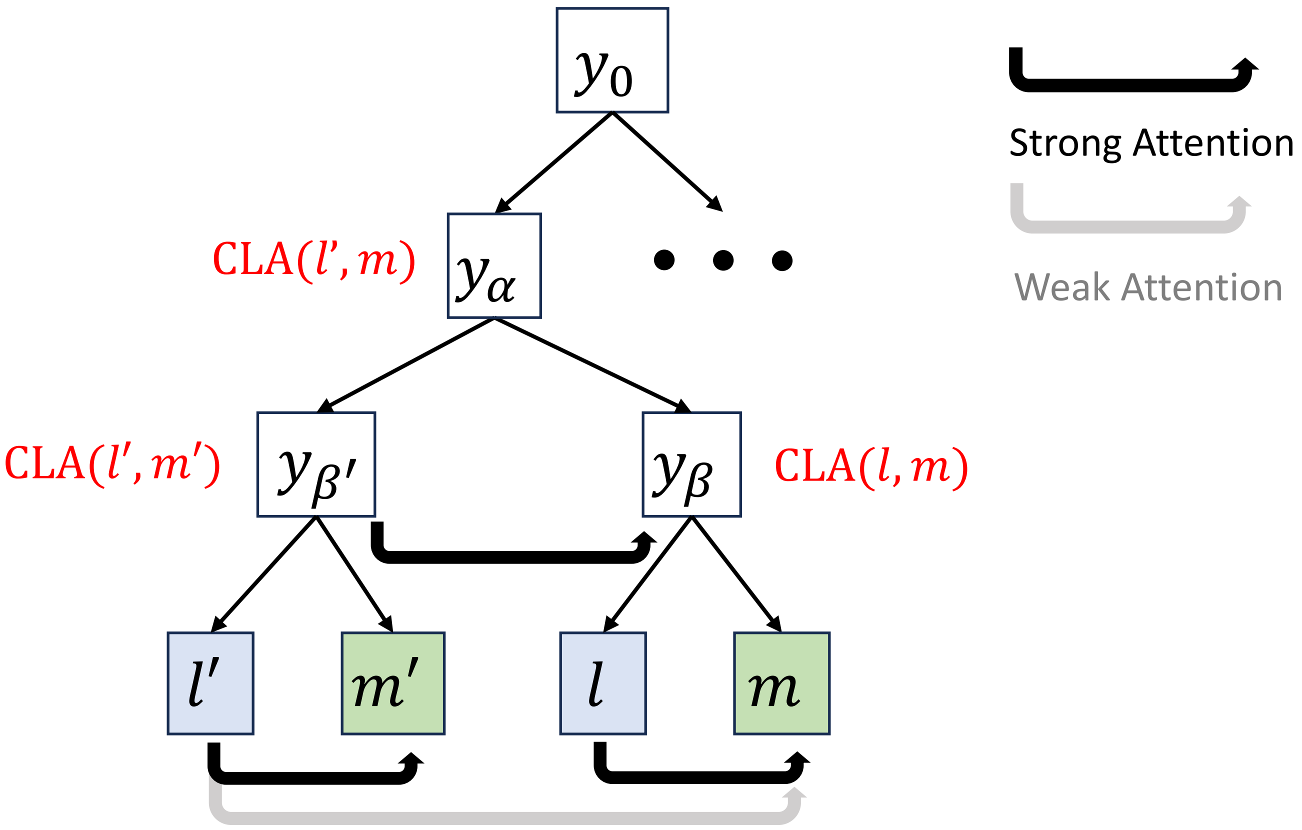
Theorem 5

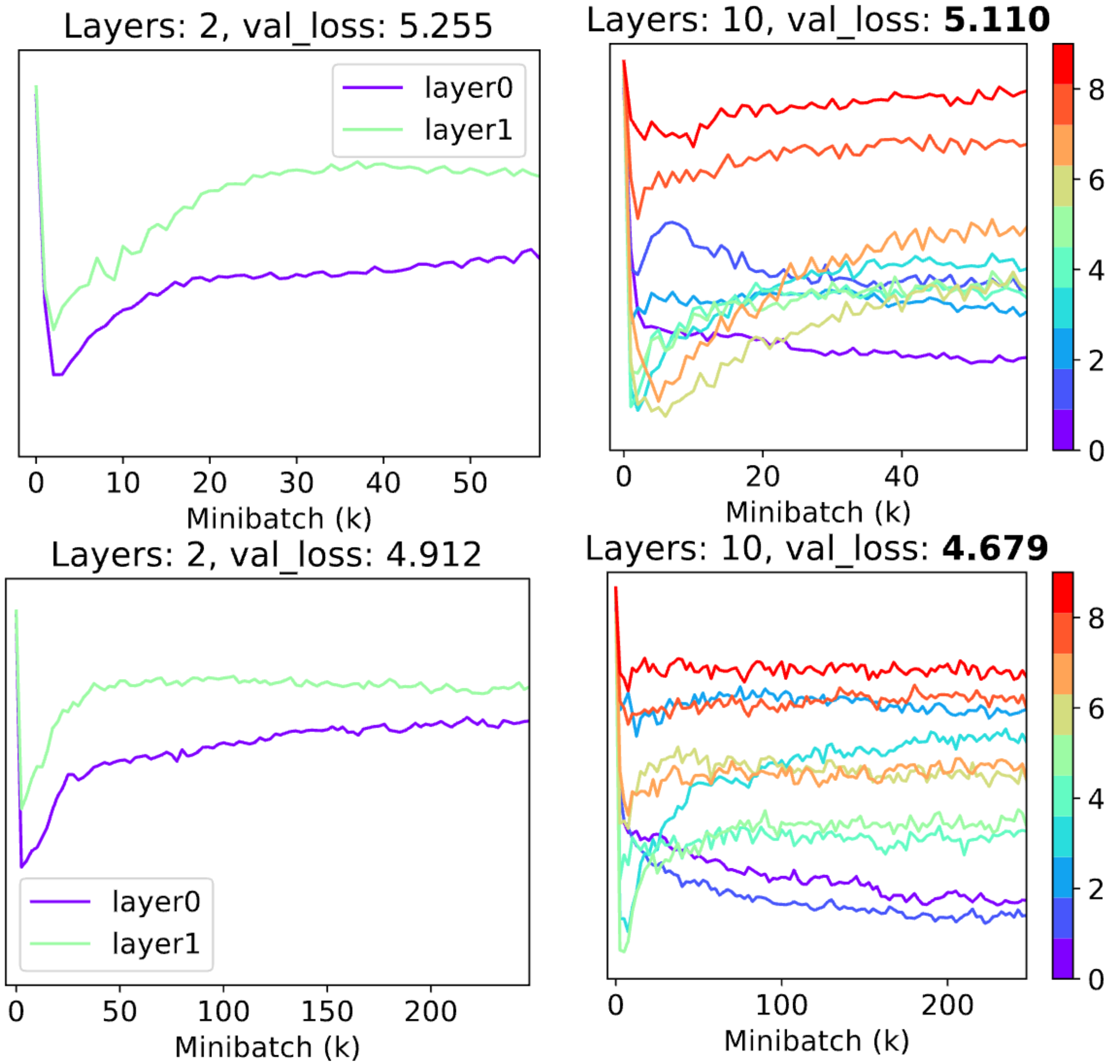
$$\mathbb{P}[l|m] \approx 1 - \frac{H}{L}$$

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

L: total height of the hierarchy

Deep Latent Distribution

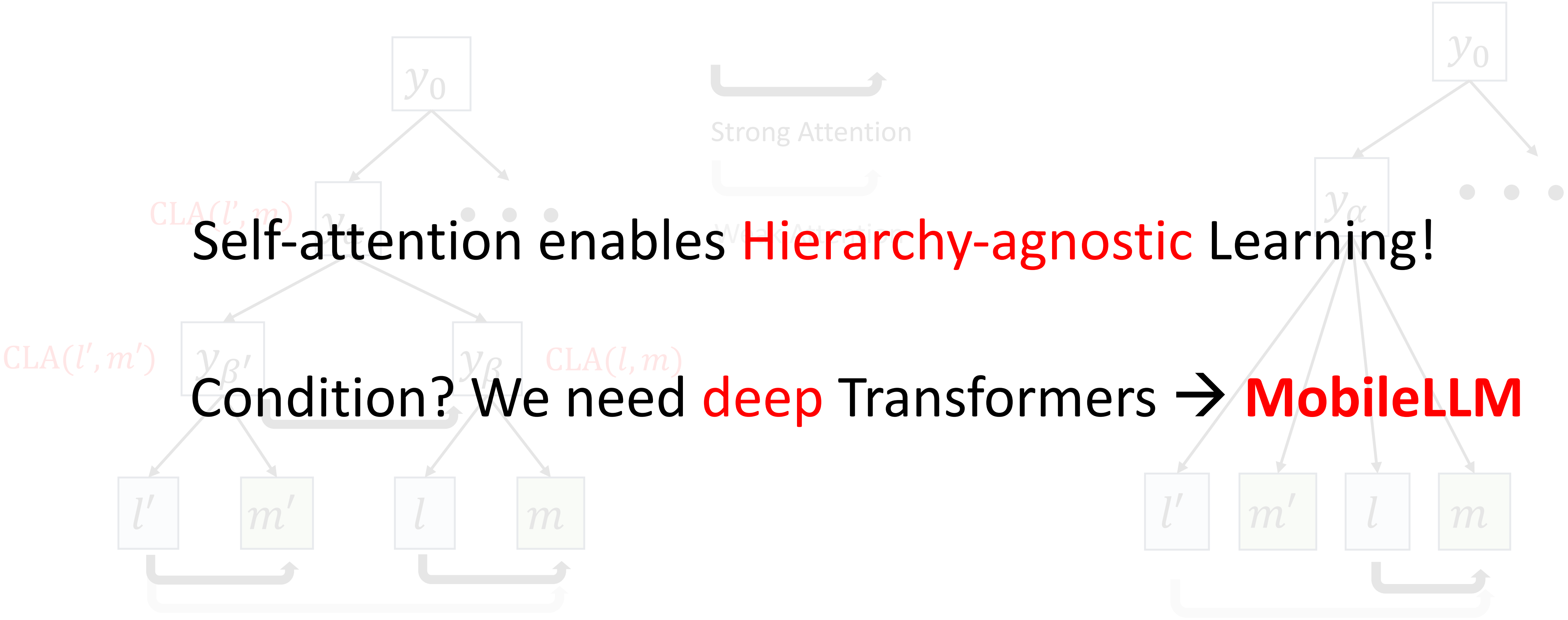




Learning the current hierarchical structure by
slowing down the association of tokens that are not directly correlated

Shallow Latent Distribution

Shallow Latent Distribution



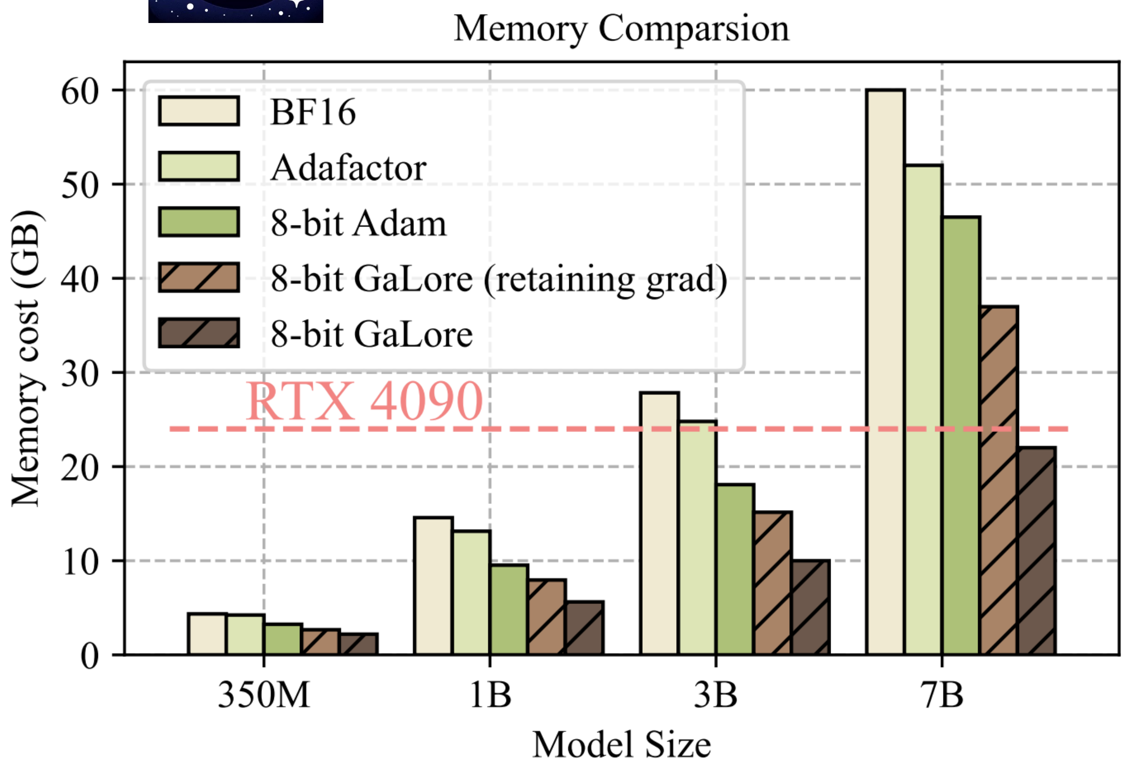
Self-attention enables Hierarchy-agnostic Learning!

Condition? We need deep Transformers → **MobileLLM**

Future Work

- How embedding vectors are learned?
  - In both Scan&Snap and JoMA, we assume embeddings are constant.
- Positional Encoding
- Formulate the dynamics of Multi-layer Transformers
  - How intermediate latent concept gets learned during training?
  - Why we need over-parameterization?

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*, arXiv'24]

Full-rank Training

Regular full-rank training. At time step t, $G_t = -\nabla_W \varphi_t(W_t) \in \mathbb{R}^{m \times n}$ is the backpropagated (negative) gradient matrix. Then the regular pre-training weight update can be written down as follows (η is the learning rate):

$$W_T = W_0 + \eta \sum_{t=0}^{T-1} \tilde{G}_t = W_0 + \eta \sum_{t=0}^{T-1} \rho_t(G_t) \qquad (1)$$

Adam (needs running momentum M_t and variance V_t as optimizer states)

$$M_t = \beta_1 M_{t-1} + (1 - \beta_1) G_t$$
$$V_t = \beta_2 V_{t-1} + (1 - \beta_2) G_t^2$$
$$\tilde{G}_t = M_t / \sqrt{V_t + \epsilon}$$

Memory Usage	Weight (W)	Optim States (M_t, V_t)	Projection (P)	Total
Full-rank	mn	$2mn$	0	$3mn$

Low-rank Adaptor (LoRA)

Low-rank updates.. For a linear layer $W \in \mathbb{R}^{m \times n}$, LoRA and its variants utilize the low-rank structure of the update matrix by introducing a low-rank adaptor AB:

$$W_T = W_0 + B_T A_T, \tag{5}$$

And we optimize B_T and A_T using Adam

Adam (needs running momentum M_t and variance V_t as optimizer states)

$$M_t = \beta_1 M_{t-1} + (1 - \beta_1) G_t$$
$$V_t = \beta_2 V_{t-1} + (1 - \beta_2) G_t^2$$
$$\tilde{G}_t = M_t / \sqrt{V_t + \epsilon}$$

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	$mn + 3(mr + nr)$

Weight (W): mn ↑ W_0, mr ↑ B_T, nr ↑ A_T

Optim States (M_t, V_t): mr ↑ B_T, nr ↑ A_T

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

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

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

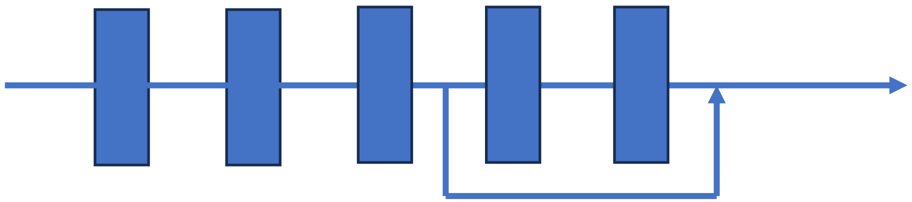
$\tilde{G}_t \leftarrow P_t \tilde{R}_t$ *{project-back}*

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

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	

Why gradient is low-rank?

Reversible models [**Y. Tian**. DDN, arXiv'20]



There exists $K(\boldsymbol{x}; W)$ so that

1. [Forward] $\boldsymbol{y} = K(\boldsymbol{x}; W)\boldsymbol{x}$
2. [Backward] $\boldsymbol{g}_{\boldsymbol{x}} = K^\top(\boldsymbol{x}; W)\boldsymbol{g}_{\boldsymbol{y}}$

Here $K(\boldsymbol{x}; W)$ depends on the input x and weight W in the network $\mathcal{N}$.

Example: Linear, ReLU / LeakyReLU, polynomials

Property of Reversible models

For reversible models trained with ℓ_2 loss or softmax

$$G_t = \frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{a}_i - B_i W_t \boldsymbol{f}_i)\boldsymbol{f}_i^\top$$

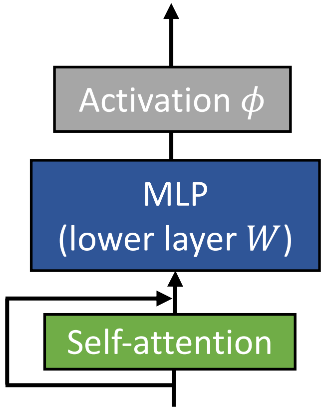
Here B_i are PSD matrices

Gradient becomes low-rank ($\mathrm{sr}(\cdot)$ is stable rank):

$$\mathrm{sr}(G_t) \le \mathrm{sr}\left(G_{t_0}^{\#}\right) + O\left[\left(\frac{1-\eta\lambda_2}{1-\eta\lambda_1}\right)^{2(t-t_0)}\right]$$

$\lambda_1 < \lambda_2$ are two smallest distinct eigenvectors of $S := \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{f}_i\boldsymbol{f}_i^\top \otimes B_i$

Transformer Case



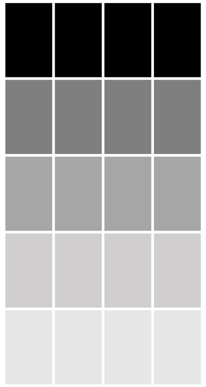
- $V = U_C^\top W \in \mathbb{R}^{M_C \times K}$ and the dynamics becomes

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

We can prove that $V(t)$ 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

Convergence Analysis

If t % T == 0:

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

$$W_t = W_0 + \sum_i \Delta W_{T_i}$$

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

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, then

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

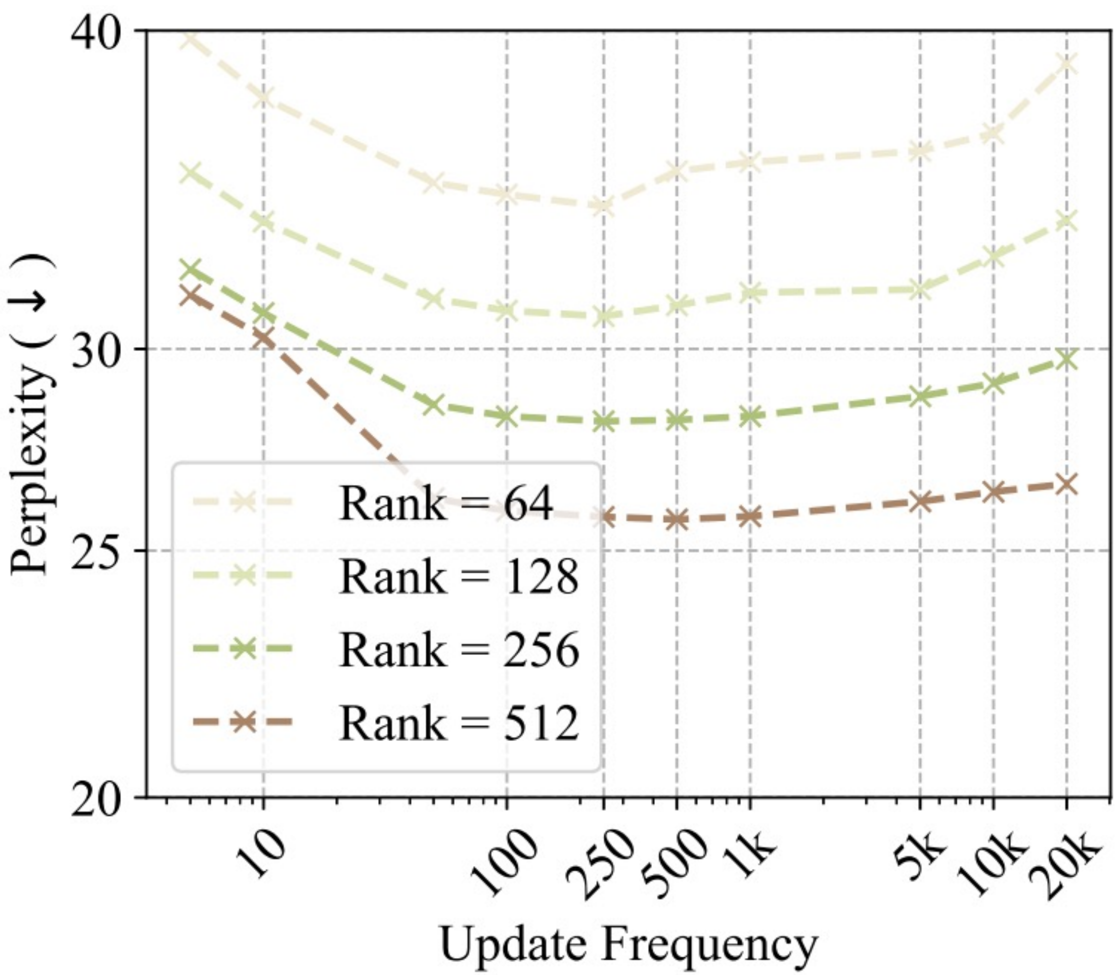
Where $M := \frac{1}{N}\sum_i \min_t \lambda_{\min}(\hat{B}_{it})\lambda_{\min}(\hat{C}_{it}) - L_A - L_B L_C D^2$

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!



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)

Compare with Adafactor

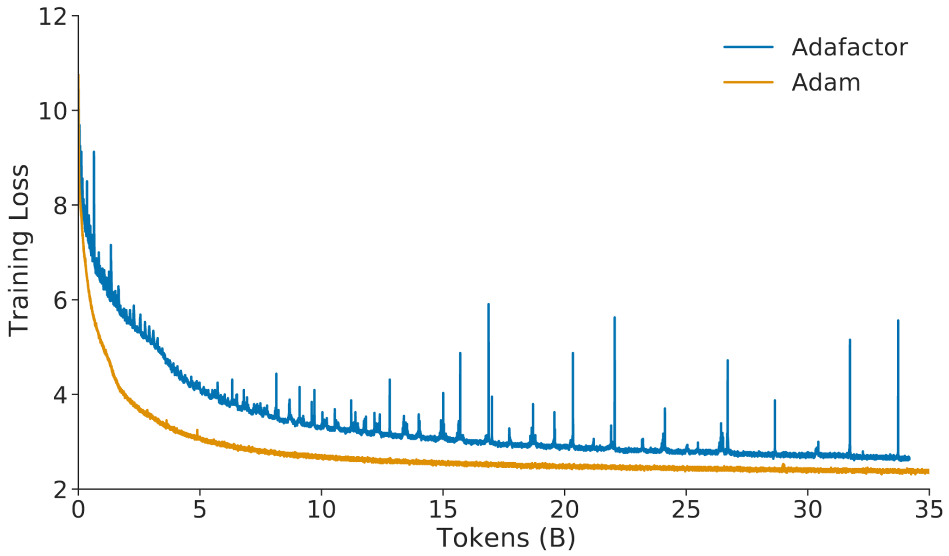
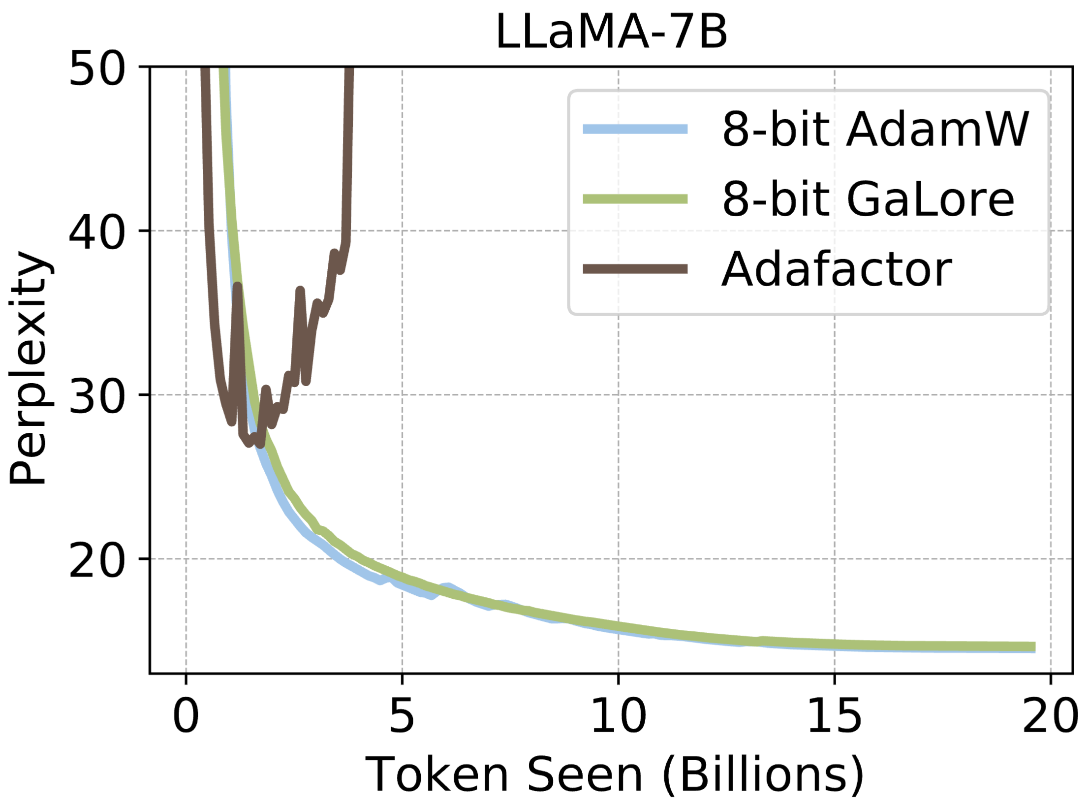


Figure A6 | **7.1B model train with Adafactor and Adam.** We found that training with Adafactor resulted in increased training instabilities at larger scales. This resulted in unhealthy training curves even at smaller learning rates and increased probability of a divergence.

[J. W. Rae, Scaling Language Models: Methods, Analysis & Insights from Training Gopher]



Fine-tuning Results

SQuAD (Bert-Base)

Method	Exact Match	F1
Full-parameter	80.83	88.41
GaLore (r=16)	80.52	88.29
LoRA (r=16)	77.99	86.11

Oaast-SFT (Reporting Perplexity)

Method	Gemma-2b	Phi-2	LLaMA-7B
Full-parameter	4.53	3.81	2.98
GaLore (r=128)	4.51	3.83	2.95
LoRA (r=128)	4.56	4.24	2.94

Belle-1M (Reporting Perplexity)

Method	Gemma-2b	Phi-2	LLaMA-7B
Full-parameter	5.44	2.66	2.27
GaLore (r=128)	5.35	2.62	2.28
LoRA (r=128)	5.37	2.75	2.30

Impact of GaLore

jiaweizzhao / GaLore

Code | Issues 13 | Pull requests 2 | Discussions | Actions | Projects

GaLore Public

Watch 18 | Fork 93 | Starred 927

huggingface / transformers Public

Code | Issues 799 | Pull requests 252 | Actions | Projects 25

FEAT / Optim: Add GaLore optimizer #

Merged younesbelkada merged 44 commits into huggingface:main fro

Conversation 105 | Commits 44 | Checks 3

younesbelkada commented 2 weeks ago • edited

What does this PR do?

As per title, adds the GaLore optimizer from https://github.com/jiaweizzhao/GaLore

FEDML AI
@FEDML_AI

LinkedIn / Twitter post:
🚀 Exciting News! 🚀 #pretraining #finetuning #llm #GaLore #FEDML
🌟 FEDML Nexus AI platform now unlocks the pre-training and fine-tuning of LLaMA-7B on geo-distributed RTX4090s!

📈By supporting the newly developed GaLore as a ready-to-launch job in FEDML Nexus AI, we have enabled the pre-training and fine-tuning of models like LLaMA 7B with a token batch size of 256 on a single RTX 4090, without additional memory optimization.

Hugging Face | Search models, datasets, users... | Models | Datasets | Spaces | Posts | Docs

← Back to blog

GaLore: Advancing Large Model Training on Consumer-grade Hardware

Medium | Search

Memory-efficient LLM Training with GaLore

A novel approach for memory-efficient LLM finetuning, how to use it and what to expect

Geronimo · Follow
6 min read · 1 day ago

MARKTECHPOST

ML News | LLMs | Other AI News | AI Dev Tools | AI Tools | AI Cou

Home › Tech News › AI Paper Summary › Revolutionizing LLM Training with GaLore: A New Machine Learni

Revolutionizing LLM Training with GaLore: A New Machine Learning Approach to Enhance Memory Efficiency without Compromising Performance

By Adnan Hassan - March 10, 2024

Reddit | Y | in

Table 1: Efficient fine-tuning techniques featured in LLAMAFACTORY. Techniques that are compatible with each other are marked with ✓, while those that are not compatible are marked with ✗.

	Freeze-tuning	GaLore	LoRA	DoRA
Mixed precision	✓	✓	✓	✓
Checkpointing	✓	✓	✓	✓
Flash attention	✓	✓	✓	✓
S^2 attention	✓	✓	✓	✓
Quantization	✗	✗	✓	✓
Unsloth	✗	✗	✓	✗

Thanks!