From CNN to ViT: Evaluating the Impact of Attention Block On the Learned Internal Representation

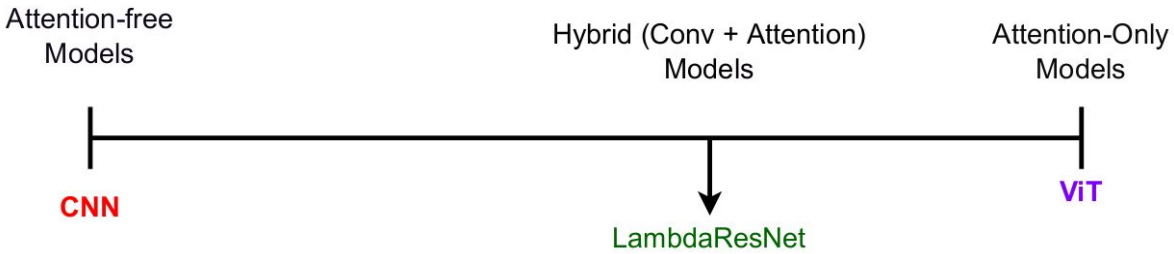
Animesh Basak Chowdhury (abc586)

Nandan Kumar Jha (nj2049)

Executive summary

Problem Statement (Goal): Self-attention operator exhibits quadratic complexity. How much Self-attention do we really need?

Solution/Approach: Investigate the entire spectrum of CNN(attention-free) model to Lambda Networks (conv + self-attention) to ViT(pure-attention) w/ HSIC metric

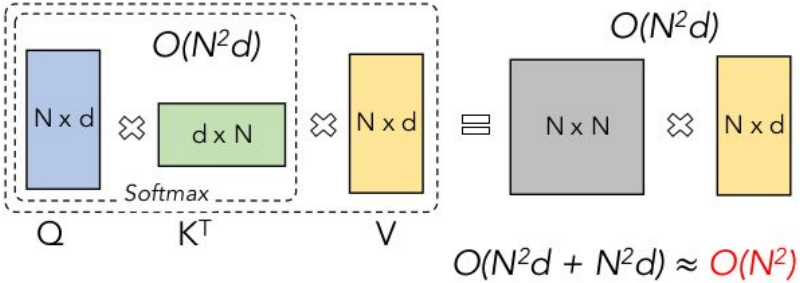


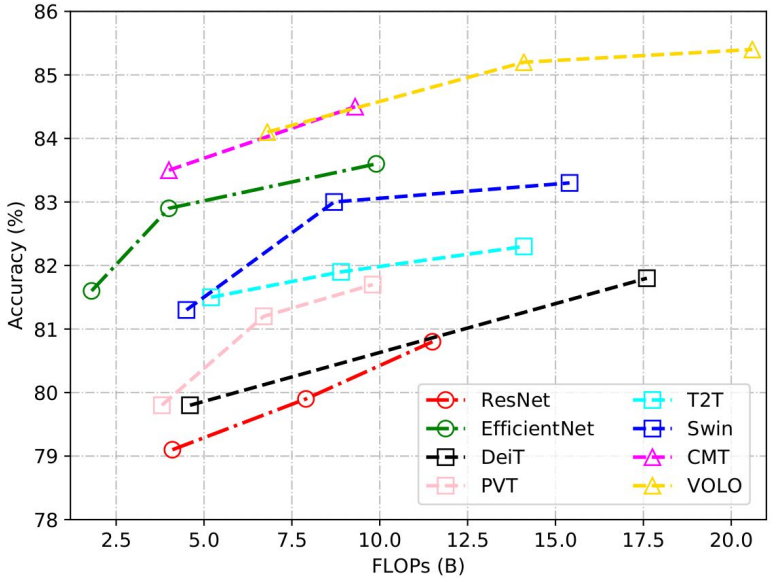
Value/Benefit: Visualizing the (learned) internal representation is a *self-explanatory, effective, and robust* way for evaluating the impact of different architectures and operators

Problem Statement: CNN vs Transformer

CNNs are inferior to pure-attention based transformer models

Self-attention: Quadratic computational and memory complexity

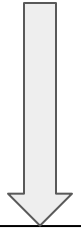




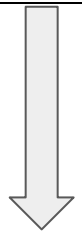
[Han, TPAMI'22]

Problem Statement: How good are the hybrid models?

1Q: Do we really need pure attention based models?

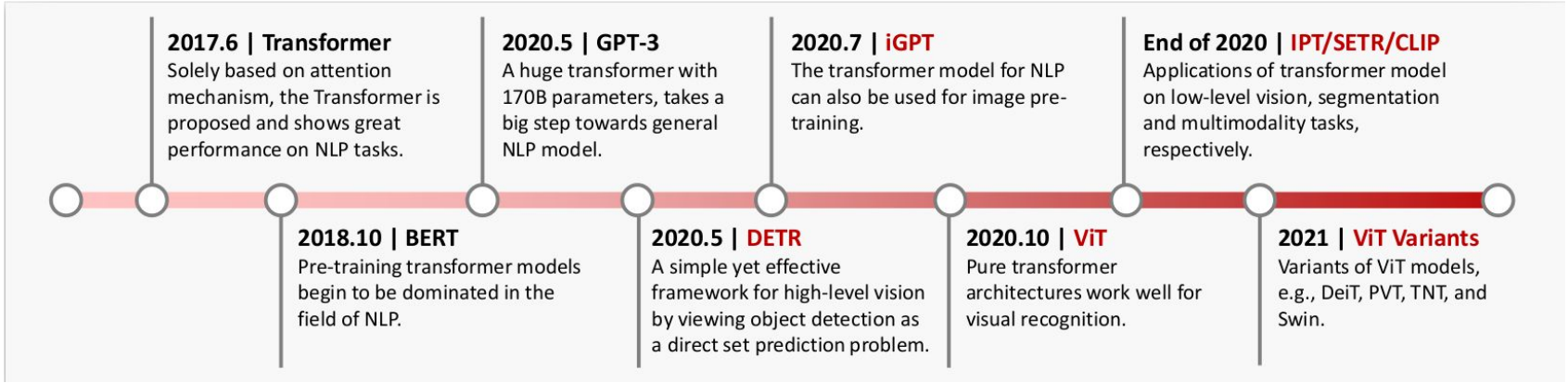


2Q: How good are the Hybrid models (CNN w/ self-attention in fewer layers)?

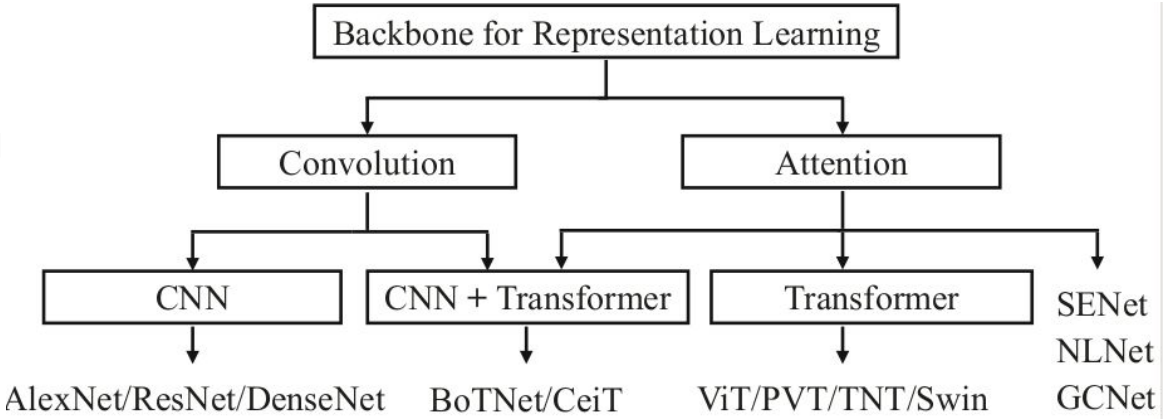


3Q: How self-attention changes the (learned)internal representation?

Background: Model Complexity (CNN vs ViT)

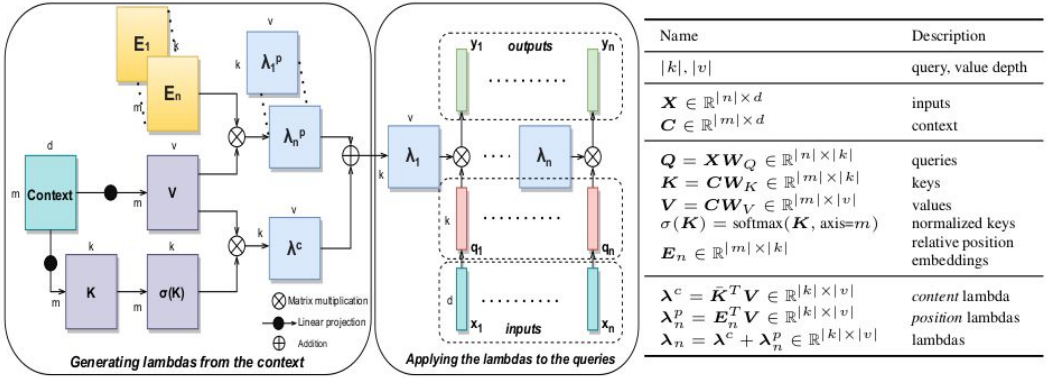


[A Survey on Vision Transformer: Han, TPAMI'22]



Background: Hybrid Network w/ Linear Self-attention [Bello, ICLR'21]

Architecture	Params (M)	Throughput	top-1
C → C → C → C	25.6	7240 ex/s	76.9
L → C → C → C	25.5	1880 ex/s	77.3
L → L → C → C	25.0	1280 ex/s	77.2
L → L → L → C	21.7	1160 ex/s	77.8
L → L → L → L	15.0	1160 ex/s	78.4
C → L → L → L	15.1	2200 ex/s	78.3
C → C → L → L	15.4	4980 ex/s	78.3
C → C → C → L	18.8	7160 ex/s	77.3



Name	Description
$\lvert k\rvert, \lvert v\rvert$	query, value depth
$\boldsymbol{X} \in \mathbb{R}^{\lvert n\rvert \times d}$	inputs
$\boldsymbol{C} \in \mathbb{R}^{\lvert m\rvert \times d}$	context
$\boldsymbol{Q} = \boldsymbol{X}\boldsymbol{W}_Q \in \mathbb{R}^{\lvert n\rvert \times \lvert k\rvert}$	queries
$\boldsymbol{K} = \boldsymbol{C}\boldsymbol{W}_K \in \mathbb{R}^{\lvert m\rvert \times \lvert k\rvert}$	keys
$\boldsymbol{V} = \boldsymbol{C}\boldsymbol{W}_V \in \mathbb{R}^{\lvert m\rvert \times \lvert v\rvert}$	values
$\sigma(\boldsymbol{K}) = \text{softmax}(\boldsymbol{K}, \text{axis}{=}m)$	normalized keys
$\boldsymbol{E}_n \in \mathbb{R}^{\lvert m\rvert \times \lvert k\rvert}$	relative position embeddings
$\boldsymbol{\lambda}^c = \bar{\boldsymbol{K}}^T\boldsymbol{V} \in \mathbb{R}^{\lvert k\rvert \times \lvert v\rvert}$	*content* lambda
$\boldsymbol{\lambda}_n^p = \boldsymbol{E}_n^T\boldsymbol{V} \in \mathbb{R}^{\lvert k\rvert \times \lvert v\rvert}$	*position* lambdas
$\boldsymbol{\lambda}_n = \boldsymbol{\lambda}^c + \boldsymbol{\lambda}_n^p \in \mathbb{R}^{\lvert k\rvert \times \lvert v\rvert}$	lambdas

Lambda convolution are used in selective layers only!

$$\boldsymbol{\lambda}_n = \sum_m (\bar{\boldsymbol{k}}_m + \boldsymbol{e}_{nm})\boldsymbol{v}_m^T = \underbrace{\bar{\boldsymbol{K}}^T\boldsymbol{V}}_{\text{content lambda}} + \underbrace{\boldsymbol{E}_n^T\boldsymbol{V}}_{\text{position lambda}} \in \mathbb{R}^{|k|\times|v|}$$

Model	Block Configuration	Lambda layers in c4
LambdaResNet-50	[3-4-6-3]	3
LambdaResNet-101	[3-4-23-3]	6, 12, 18
LambdaResNet-152	[3-8-36-3]	5, 10, 15, 20, 25, 30
LambdaResNet-200	[3-24-36-3]	5, 10, 15, 20, 25, 30
LambdaResNet-270	[4-29-53-4]	8, 16, 24, 32, 40, 48
LambdaResNet-350	[4-36-72-4]	10, 20, 30, 40, 50, 60
LambdaResNet-420	[4-44-87-4]	10, 20, 30, 40, 50, 60, 70, 80

Most of the benefits come from the placing lambda-layers in the last stages of the networks.

Technical Challenges

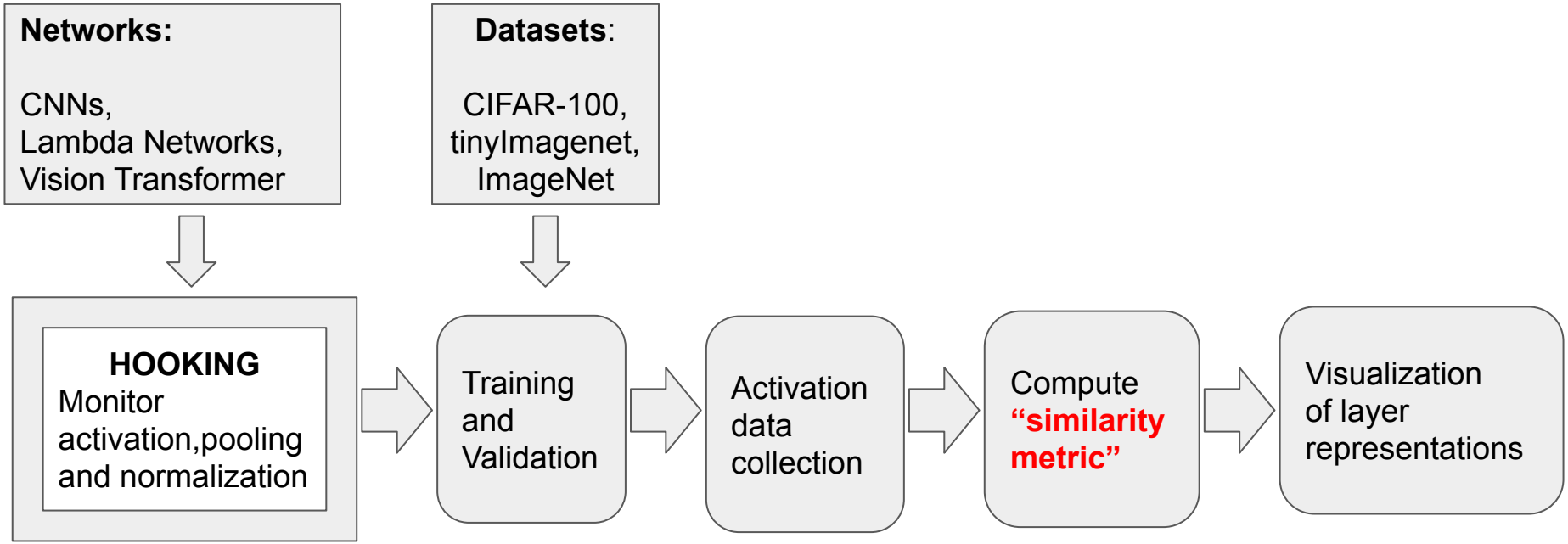
Challenge1: What is the right way to investigate the impact of conv vs self-attention operator?

Hint: A single metric (e.g., Top-1 val accuracy) *is not enough* to describe the impact of different architecture and operators on the learning of networks.

Challenge2: How to visualize the fundamental change on the learned representation of a over-parameterized network with high-dimensional data manifold?

Hint: A distance-metric/function that can capture the similarity-dissimilarity in non-euclidean space/geometry.

Representation similarity in vision models: A visualization approach



Similarity metric

What does it **measure**?

Similarity between two learned representation in high-dimensional space.

Similarity Index	Formula	Invariant to		
		Invertible Linear Transform	Orthogonal Transform	Isotropic Scaling
Linear Reg. (R^2_{LR})	$\|\|Q_Y^{\mathrm{T}} X\|\|_{\mathrm{F}}^2 / \|\|X\|\|_{\mathrm{F}}^2$	Y only	✓	✓
CCA (R^2_{CCA})	$\|\|Q_Y^{\mathrm{T}} Q_X\|\|_{\mathrm{F}}^2 / p_1$	✓	✓	✓
CCA ($\bar{\rho}_{\text{CCA}}$)	$\|\|Q_Y^{\mathrm{T}} Q_X\|\|_* / p_1$	✓	✓	✓
SVCCA (R^2_{SVCCA})	$\|\|(U_Y T_Y)^{\mathrm{T}} U_X T_X\|\|_{\mathrm{F}}^2 / \min(\|\|T_X\|\|_{\mathrm{F}}^2, \|\|T_Y\|\|_{\mathrm{F}}^2)$	If same subspace kept	✓	✓
SVCCA ($\bar{\rho}_{\text{SVCCA}}$)	$\|\|(U_Y T_Y)^{\mathrm{T}} U_X T_X\|\|_* / \min(\|\|T_X\|\|_{\mathrm{F}}^2, \|\|T_Y\|\|_{\mathrm{F}}^2)$	If same subspace kept	✓	✓
PWCCA	$\sum_{i=1}^{p_1} \alpha_i \rho_i / \|\|\alpha\|\|_1, \alpha_i = \sum_j \|\langle \mathbf{h}_i, \mathbf{x}_j \rangle\|$	✗	✗	✓
Linear HSIC	$\|\|Y^{\mathrm{T}} X\|\|_{\mathrm{F}}^2 / (n-1)^2$	✗	✓	✗
Linear CKA	$\|\|Y^{\mathrm{T}} X\|\|_{\mathrm{F}}^2 / (\|\|X^{\mathrm{T}} X\|\|_{\mathrm{F}} \|\|Y^{\mathrm{T}} Y\|\|_{\mathrm{F}})$	✗	✓	✓
RBF CKA	$\mathrm{tr}(KHLH) / \sqrt{\mathrm{tr}(KHKH)\mathrm{tr}(LHLH)}$	✗	✓	✓*

Hilbert-Schmidt Independence Criterion(HSIC)

$\langle \mathrm{vec}(XX^{\mathrm{T}}), \mathrm{vec}(YY^{\mathrm{T}}) \rangle$

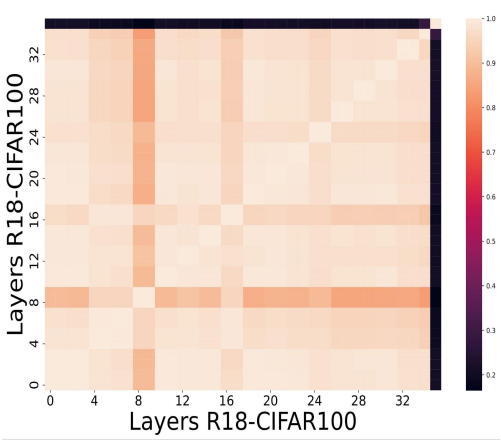
$= \mathrm{tr}(XX^{\mathrm{T}}YY^{\mathrm{T}}) = ||\mathrm{cov}(X^{\mathrm{T}}, Y^{\mathrm{T}})||^2_{\mathrm{F}}$

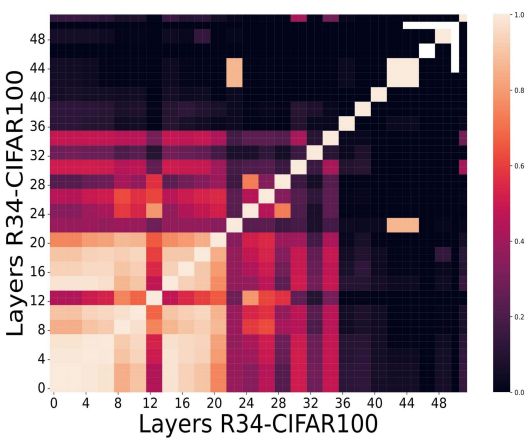
Centered Kernel Alignment (CKA)

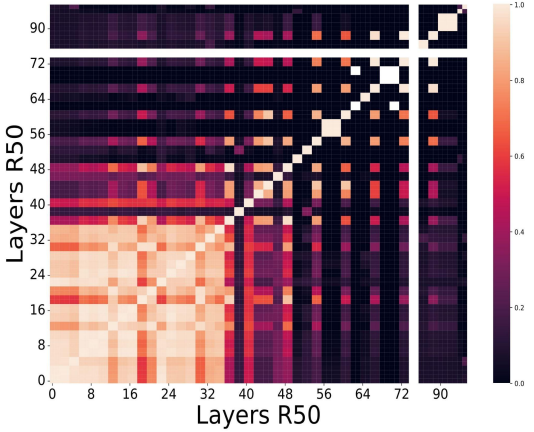
$\mathrm{CKA}(X,Y) = \mathrm{HSIC}(X,Y) / \{\mathrm{HSIC}(X,X) * \mathrm{HSIC}(Y,Y)\}^{1/2}$

Visualizing CNNs

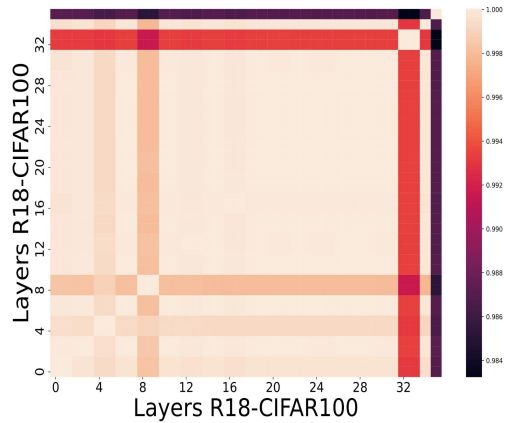
Random weights

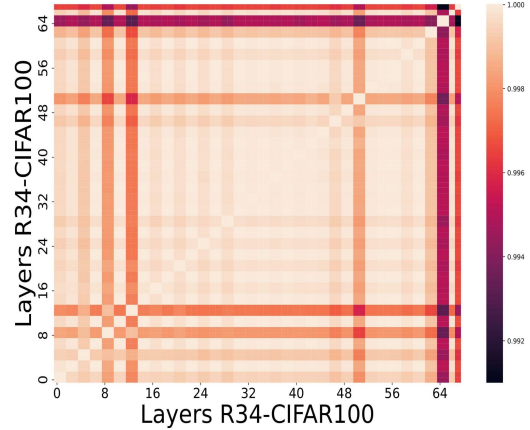


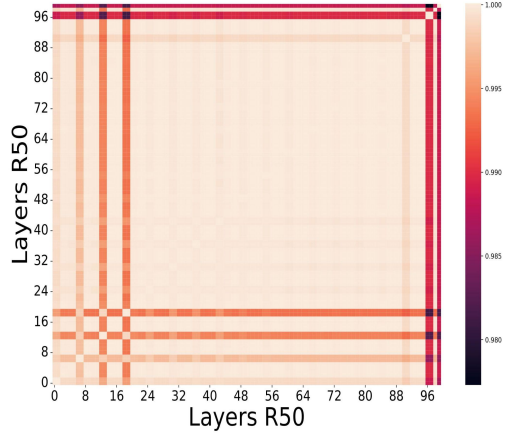




Pre-trained weights

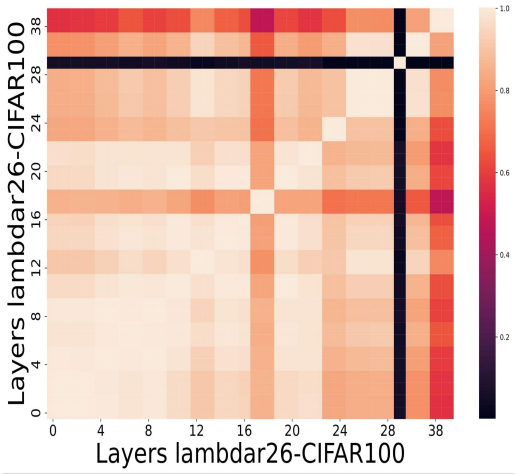


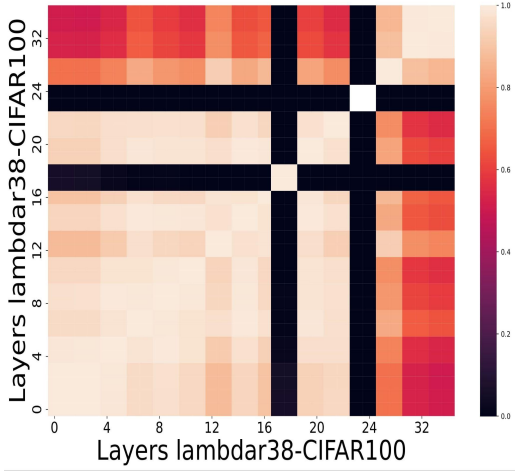


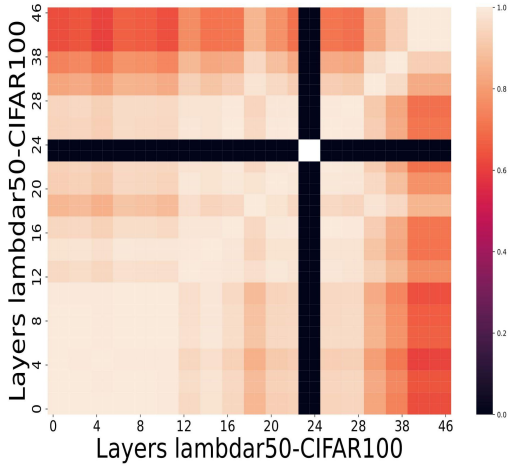


Visualizing Lambda Networks

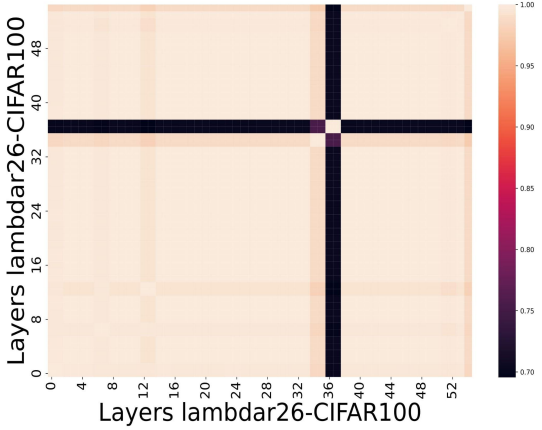
Random weights

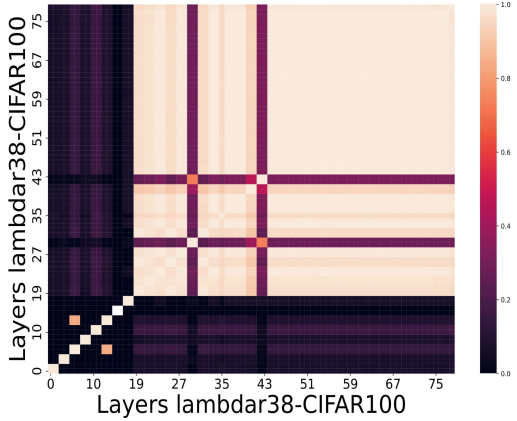


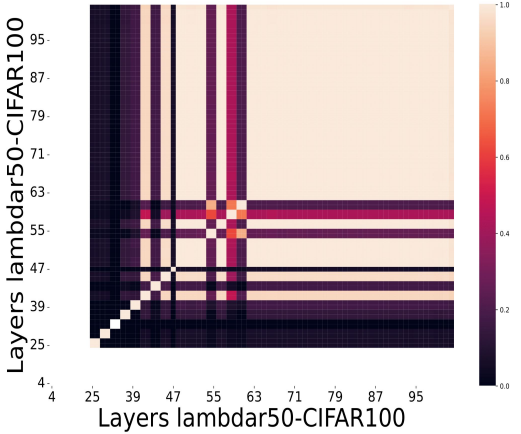




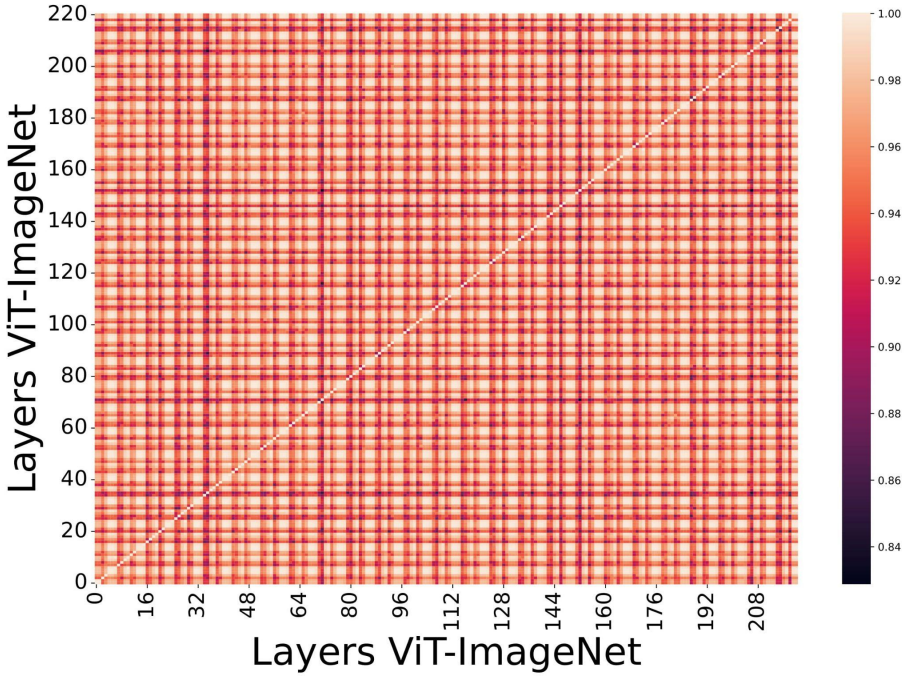
Pre-trained weights



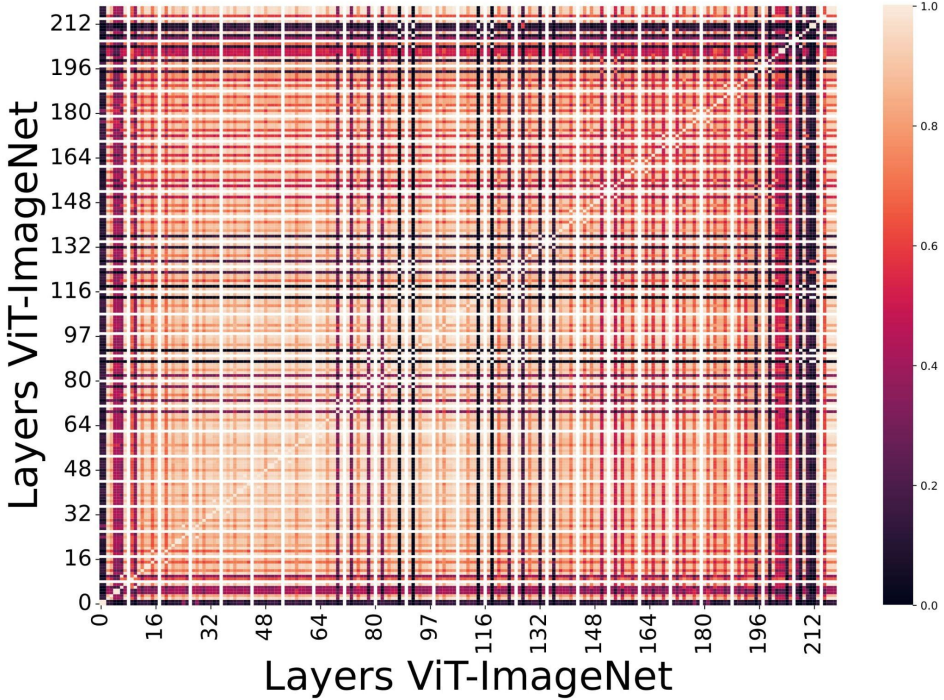




Visualizing Vision Transformers

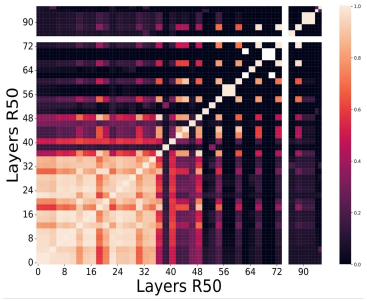


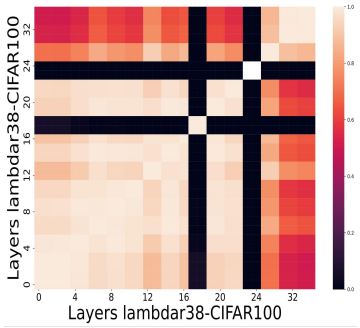
Random weights

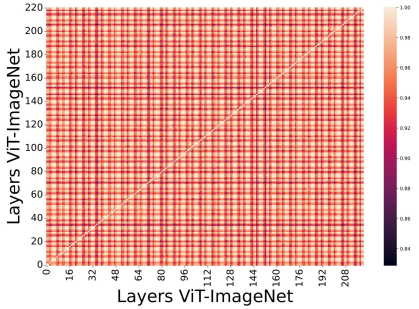


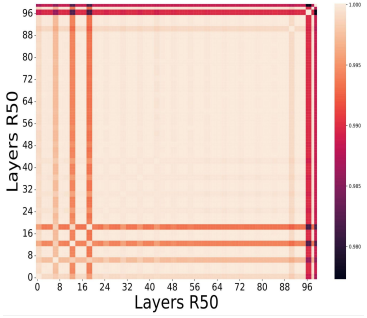
Pre-trained weights

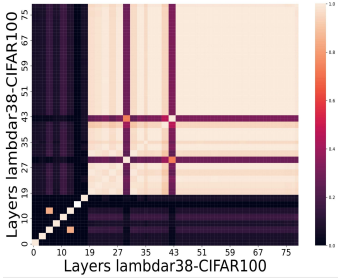
Analyzing representation from CNNs to ViT

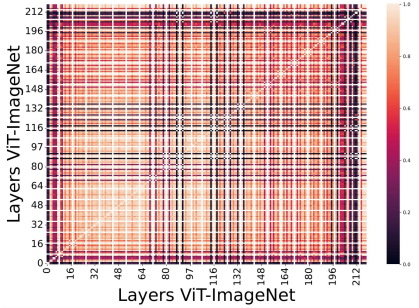












CNNs:
High similarity scores between layers

Lambda Networks:
High similarity scores in layers at later stages

ViTs:
Symmetrical Structures. Certain layers are highly uncorrelated

Conclusion

1) Visualization using **CKA** similarity metric captures **"crucial"** information about learned representations of hidden layers.

2) Our visualization has revealed an important **insight**: CNNs fails to effectively use its learning capacity; whereas self-attention has led to low CKA scores thus learning different representation across layers.

3) Representation learned at deeper layers contribute **more** to the predictive performance.

4) **Selective** use of self-attention in deeper layers of lambda networks maximizes the representation capacity of deeper layers.

Github repo: https://github.com/animeshbchowdhury/visualizing-vision-models