Learning Deep Low-Dimensional Models from High-Dimensional Data: From Theory to Practice

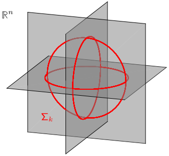
(White-Box Transformers via Sparse Rate Reduction)

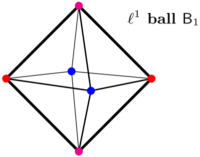
Sam Buchanan

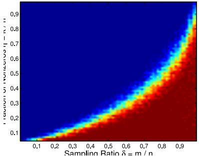
University of California, Berkeley

October 19, 2025

High-Dimensional Data Analysis: Sparse Reconstruction







Sparse recovery: **structured** signals, **linear** measurements

$$\boldsymbol{x} = \boldsymbol{A}\boldsymbol{z}_o, \quad \boldsymbol{z}_o \text{ sparse}, \quad \boldsymbol{A} \in \mathbb{R}^{m \times n} \text{ random}$$

with **convex** optimization

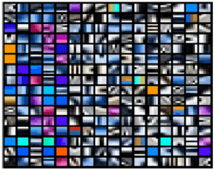
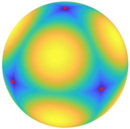
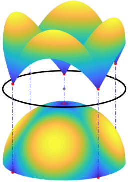
$$\boldsymbol{z}_\star = \arg\min_{\boldsymbol{z} \in \mathbb{R}^n} \frac{1}{2} \|\boldsymbol{x} - \boldsymbol{A}\boldsymbol{z}\|_2^2 + \lambda \|\boldsymbol{z}\|_1$$

and provable (high probability) guarantees

$$\boldsymbol{z}_\star = \boldsymbol{z}_o \text{ when measurements} \gtrsim \text{sparsity} \times \log\left(\frac{\text{dimension}}{\text{sparsity}}\right)$$

Wright and Ma 2022

Representation (Dictionary) Learning

Dictionary learning: **structured** signals, **bilinear** measurements

$$\boldsymbol{X} = \boldsymbol{A}_o \boldsymbol{Z}_o \in \mathbb{R}^{n \times p}, \quad \boldsymbol{Z}_o \text{ sparse and random}, \quad \boldsymbol{A}_o^* \boldsymbol{A}_o \approx \boldsymbol{I}$$

with (efficient) **nonconvex** optimization

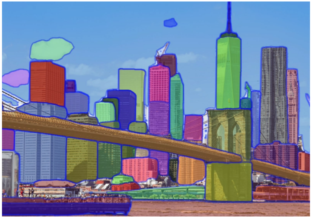
$$\boldsymbol{a}_\star = \underset{\|\boldsymbol{a}\|_2 = 1}{\arg\min} \; \|\boldsymbol{X}^* \boldsymbol{a}\|_1$$

and provable (high probability) guarantees

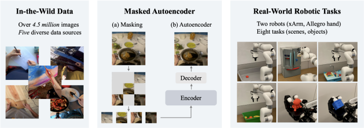
$$\boldsymbol{a}_\star \approx (\boldsymbol{A}_o)_j \text{ when observations} \geq \mathrm{poly}(\text{expected sparsity})$$

Wright and Ma 2022

Modern (Deep) Representation Learning

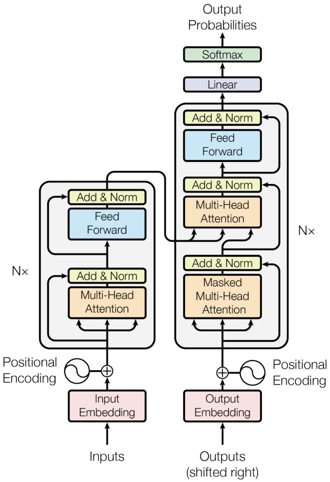


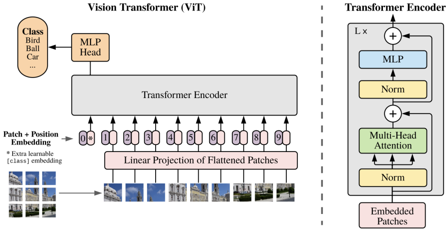
Perceiving the physical world $\implies$ **nonlinear signals**!
Nonlinearity demands **deeper** representations.



Kirillov et al. 2023; Yi, Zeng, **B.**, Ma 2023; Radosavovic et al. 2022; Wang et al. 2023

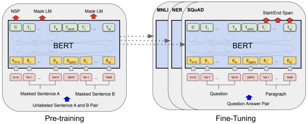
Transformers: Modern Representation Learning's Workhorse



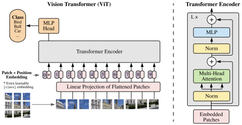


Vaswani et al. 2017, Dosovitskiy et al. 2021

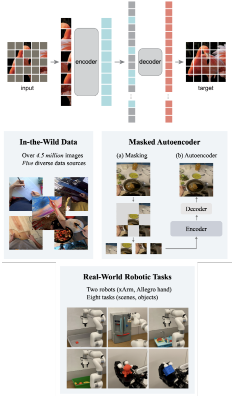
Transformers: A Universal Backbone

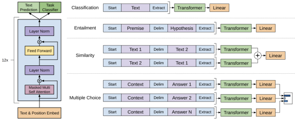


BERT

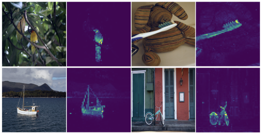


ViT





GPT



DINO

TF + NLP

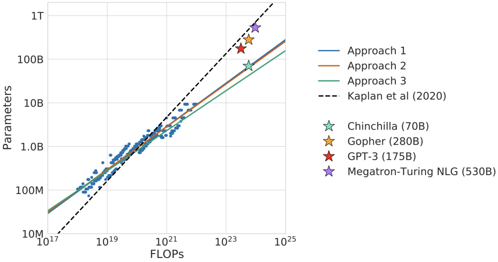
TF + Vision

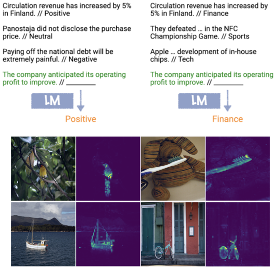
TF + Robotics

Devlin et al. 2018, Radford et al. 2018, Dosovitskiy et al. 2021, Caron et al. 2021, He et al. 2021, Radosavovic et al. 2022

Shortcomings of Black-Box Models?

Transformers are **empirically-designed** (or **“black-box”** models).



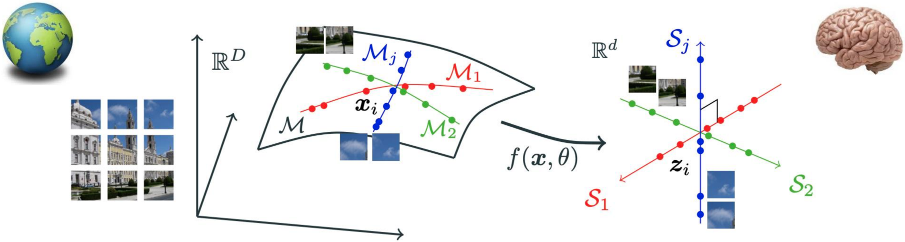


How to understand such “emergent” phenomena?
What to do when things go wrong?

Competing partial theoretical interpretations, e.g. [Vidal 2022] [Bai et al. 2023] [Geshkovski et al. 2023]

Hoffmann et al. 2022, Brown et al. 2020

Representations: What and How to Learn?



The main objective of learning:
Identify low-dimensional structures in sensed data of the world and transform to a compact and structured representation.

Outline

1. Analytical Models
   - Geometry and Sparsity
   - Optimization and Neural Networks
2. Deep Representation Learning
   - Transformers for Visual Data
   - Objectives for Representation Learning
   - Unrolled Optimization for Representation Learning
   - Compression and Self-Attention
   - Sparsification and MLP
   - Coding Rate Reduction Transformer
   - Experimental Results on CRATE
3. Conclusions for the Tutorial

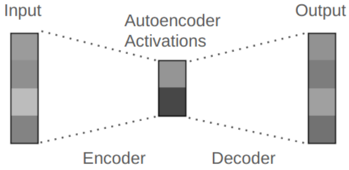
Outline

1. Analytical Models
   - Geometry and Sparsity
   - Optimization and Neural Networks
2. Deep Representation Learning
   - Transformers for Visual Data
   - Objectives for Representation Learning
   - Unrolled Optimization for Representation Learning
   - Compression and Self-Attention
   - Sparsification and MLP
   - Coding Rate Reduction Transformer
   - Experimental Results on CRATE
3. Conclusions for the Tutorial

A Low-Dimensional Subspace

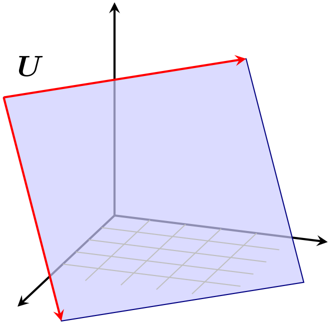
Canonical example:
subspace $\boldsymbol{U} \in \mathbb{R}^{D \times d}$:

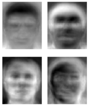
$$\boldsymbol{x} \xrightarrow{\quad f = \boldsymbol{U}^\top \quad} \boldsymbol{z} \xrightarrow{\quad g = \boldsymbol{U} \quad} \hat{\boldsymbol{x}}$$



Principal component analysis:

$$\min_{\boldsymbol{U}} \mathbb{E}\left[\left\|\boldsymbol{x} - \boldsymbol{U}\boldsymbol{U}^\top \boldsymbol{x}\right\|_2^2\right]$$





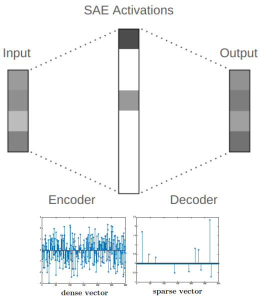
Sparsity and Sparse Coding

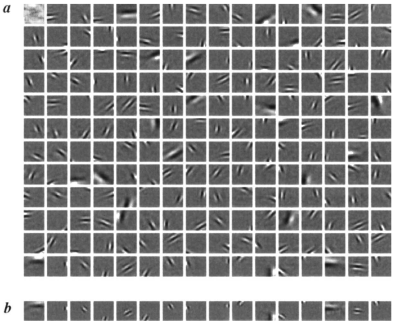
A significant generalization: unions of subspaces

ℓ^0 "norm": **# of nonzero entries**, $\|\boldsymbol{x}\|_0 = |\{i \mid x_i \neq 0\}|$.

Given (learned) $\boldsymbol{A} \in \mathbb{R}^{D \times d}$, represent $\boldsymbol{x}$ as a sparse code:

$$f(\boldsymbol{x}) = \min_{\boldsymbol{z}} \|\boldsymbol{x} - \boldsymbol{A}\boldsymbol{z}\|_2^2 + \|\boldsymbol{z}\|_0\,; \qquad \boldsymbol{x} \xrightarrow{f} \boldsymbol{z} \xrightarrow{g=\boldsymbol{A}} \hat{\boldsymbol{x}}$$





Olshausen and Field 1996

How to Learn: Optimization for Low-Dim Structures

We can compute sparse coding with (proximal) gradient descent

$$f(\boldsymbol{x}) = \arg\min_{\boldsymbol{z} \geq 0} \|\boldsymbol{x} - \boldsymbol{A}\boldsymbol{z}\|_2^2 + \lambda \|\boldsymbol{z}\|_1$$

1. Given the current code $\boldsymbol{z}^\ell$, gradient descent to better fit $\boldsymbol{x}$;
2. Without moving too much, sparsify the updated code

Then $f(\boldsymbol{x}) = \boldsymbol{z}^\infty$, where

$$\boldsymbol{z}^{\ell+1} = \mathrm{ReLU}\left(\eta \boldsymbol{A}^\top \boldsymbol{x} + \left(\boldsymbol{I} - \eta \boldsymbol{A}^\top \boldsymbol{A}\right) \boldsymbol{z}^\ell - \lambda\eta\mathbf{1}\right)$$

Unrolled Optimization: From Objectives to Deep Networks

Recall the sparse coding objective:

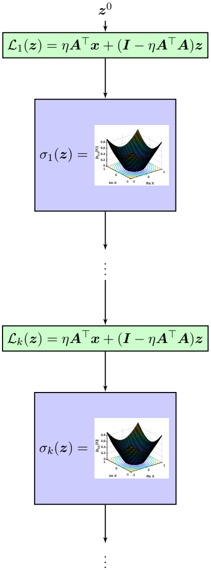
$$f(\boldsymbol{x}) = \arg\min_{\boldsymbol{z} \geq 0} \|\boldsymbol{x} - \boldsymbol{A}\boldsymbol{z}\|_2^2 + \lambda \|\boldsymbol{z}\|_1$$

Then $f(\boldsymbol{x}) = \boldsymbol{z}^\infty$, where

$$\boldsymbol{z}^{\ell+1} = \mathrm{ReLU}\left(\eta \boldsymbol{A}^\top \boldsymbol{x} + \left(\boldsymbol{I} - \eta \boldsymbol{A}^\top \boldsymbol{A}\right) \boldsymbol{z}^\ell - \lambda \eta \mathbf{1}\right)$$

Truncate the network, and **learn** its parameters using data. $(\boldsymbol{A} \to \boldsymbol{A}^\ell)$

This approach is called LISTA [Gregor and Lecun, 2010].
$\Longrightarrow$ each layer learns its own dictionary!



Surveys: Monga et al. 2020; Shlezinger et al. 2022; Chen et al. 2022

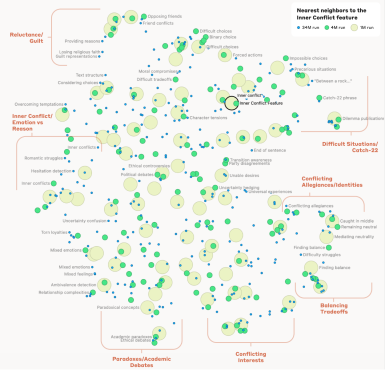
Special Case: Sparse Autoencoders

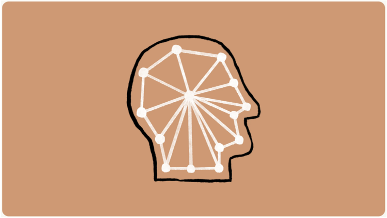
Truncate after one iteration & learn a dictionary $\boldsymbol{D}$ for decoding:

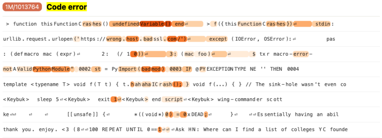
$$f(\boldsymbol{x}) = \mathrm{ReLU}\left(\boldsymbol{A}^\top \boldsymbol{x} - \boldsymbol{b}\right)$$

$$\boldsymbol{x} \xrightarrow{\quad f \quad} \boldsymbol{z} \xrightarrow{\quad g=\boldsymbol{D} \quad} \hat{\boldsymbol{x}}$$

Interpretable representations from massive-scale models!







Templeton, Conerly et al. 2024

Using Unrolled Optimization for Deep Learning

Unrolled optimization:

- Given objective function $\mathcal{R}^\ell$, improve it on input $\boldsymbol{Z}^\ell$ by taking optimization step:

$$\boldsymbol{Z}^{\ell+1} \leftarrow \boldsymbol{Z}^{\ell} - \kappa \nabla_{\boldsymbol{Z}} \mathcal{R}^\ell(\boldsymbol{Z}^\ell)$$

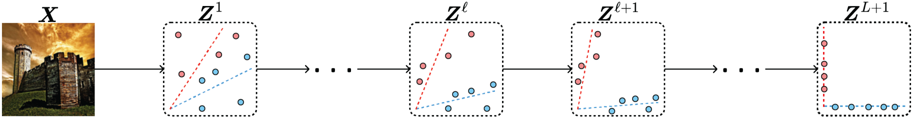
  (...or similar)

- Collection of objective functions $(\mathcal{R}^\ell)_{\ell=1}^L$ + optimization strategies
  $\implies$ data processing algorithm
- *New:* collection of objective functions + optimization strategies
  $\implies$ deep network architecture!

$$\begin{tikzcd}
\boxed{\boldsymbol{Z}^1} \arrow[d, "-\kappa \nabla_{\boldsymbol{Z}} \mathcal{R}^1(\boldsymbol{Z}^1)"] \\
\boxed{\boldsymbol{Z}^2} \arrow[d, "-\kappa \nabla_{\boldsymbol{Z}} \mathcal{R}^2(\boldsymbol{Z}^2)"] \\
\cdots \arrow[d, "-\kappa \nabla_{\boldsymbol{Z}} \mathcal{R}^{L-1}(\boldsymbol{Z}^{L-1})"] \\
\boxed{\boldsymbol{Z}^L} \arrow[d, "-\kappa \nabla_{\boldsymbol{Z}} \mathcal{R}^L(\boldsymbol{Z}^L)"] \\
\boxed{\boldsymbol{Z}^{L+1}}
\end{tikzcd}$$

From Unrolled Optimization to Deep Architectures

Constructing deep networks:
Design objectives $\mathcal{R}^\ell$ and optimization strategies s.t. unrolling yields *compact & structured* deep representation!



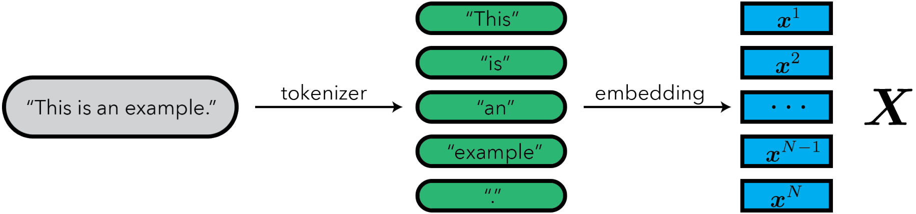
Previously: Did this for ResNets (ReduNet).
Next: *How do we do this for transformers?* What does it buy us?

Outline

1. Analytical Models
   - Geometry and Sparsity
   - Optimization and Neural Networks
2. Deep Representation Learning
   - Transformers for Visual Data
   - Objectives for Representation Learning
   - Unrolled Optimization for Representation Learning
   - Compression and Self-Attention
   - Sparsification and MLP
   - Coding Rate Reduction Transformer
   - Experimental Results on CRATE
3. Conclusions for the Tutorial

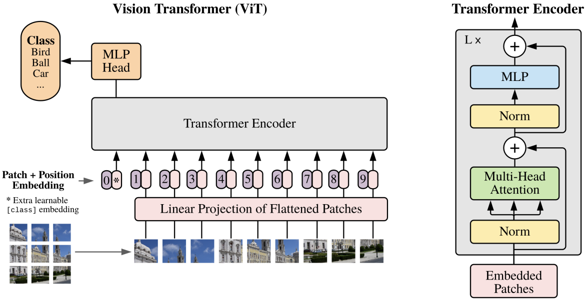
Scaling Data Processing

Data format:
Sequences of *tokens* → *embeddings* $\boldsymbol{X} = \left[\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\right] \in \mathbb{R}^{D \times N}$



Processing Images as Token Sequences with ViT

Recall: the Vision Transformer (ViT) processes images as a **sequence of patches**.

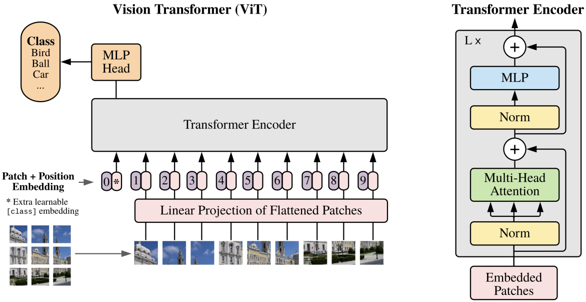


$$f_{\mathrm{ViT}} = f^L \circ \underbrace{f^{L-1}}_{\text{transformer layer}} \circ \cdots \circ f^1 \circ \underbrace{f^{\mathrm{pre}}}_{\text{tokenization}}$$

Dosovitskiy et al. 2021

Processing Images with ViT (Tokenization)

Recall: the Vision Transformer (ViT) processes images as a **sequence of patches**.



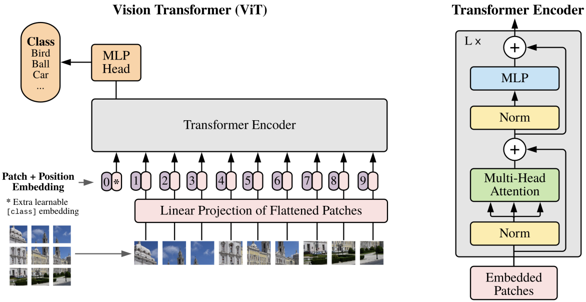
$$\underbrace{f^{\text{pre}}}_{\text{tokenization}} \qquad \boldsymbol{X} = [\boldsymbol{x}_1, \dots, \boldsymbol{x}_N] \in \mathbb{R}^{D \times N}$$

$$\boldsymbol{Z}^1 = [\boldsymbol{z}_{\text{cls}}, \boldsymbol{W}^{\text{pre}} \boldsymbol{X}] + \boldsymbol{E}_{\text{pos}} = [\boldsymbol{z}_{\text{cls}}, \boldsymbol{z}_1, \dots, \boldsymbol{z}_N] \in \mathbb{R}^{d \times (N+1)}$$

Dosovitskiy et al. 2021

Processing Images with ViT (TF Block)

Recall: the Vision Transformer (ViT) processes images as a **sequence of patches**.



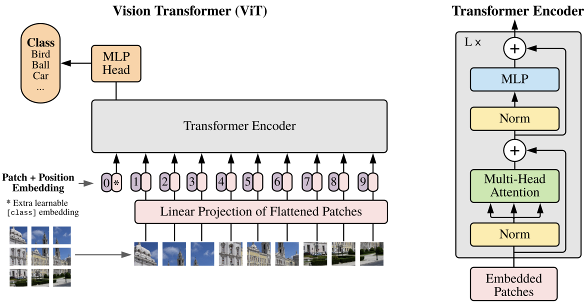
$$\underbrace{f^\ell}_{\text{TF layer}}$$

$$\boldsymbol{Z}^{\ell+1/2} = \mathtt{MHSA}(\mathtt{LN}(\boldsymbol{Z}^\ell)) + \boldsymbol{Z}^\ell$$
$$f^\ell(\boldsymbol{Z}^\ell) = \mathtt{MLP}(\mathtt{LN}(\boldsymbol{Z}^{\ell+1/2})) + \boldsymbol{Z}^{\ell+1/2}$$

Dosovitskiy et al. 2021

Processing Images with ViT (TF Block)

Recall: the Vision Transformer (ViT) processes images as a **sequence of patches**.



$$\underbrace{f^{\ell}}_{\text{TF layer}}$$

$$\boldsymbol{Z}^{\ell+1/2} = \mathtt{MHSA}(\mathtt{LN}(\boldsymbol{Z}^{\ell})) + \boldsymbol{Z}^{\ell}$$

$$f^{\ell}(\boldsymbol{Z}^{\ell}) = \mathtt{MLP}(\mathtt{LN}(\boldsymbol{Z}^{\ell+1/2})) + \boldsymbol{Z}^{\ell+1/2}$$

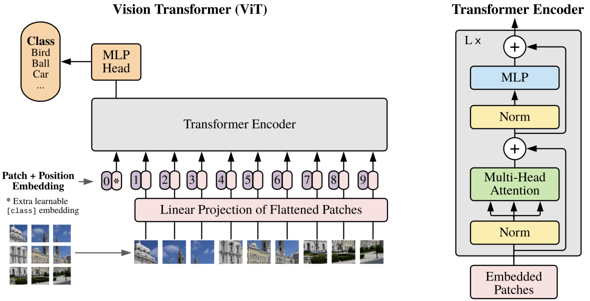
$$\mathtt{SA}(\boldsymbol{Z}; \boldsymbol{W}_Q, \boldsymbol{W}_K, \boldsymbol{W}_V) = \boldsymbol{W}_V \boldsymbol{Z} \mathrm{softmax}((\boldsymbol{W}_Q \boldsymbol{Z})^* (\boldsymbol{W}_K \boldsymbol{Z}))$$

$$\mathtt{MHSA}(\boldsymbol{Z}) = \sum_{h=1}^{H} \boldsymbol{W}_{O,h} \mathtt{SA}(\boldsymbol{Z}; \boldsymbol{W}_{Q,h}, \boldsymbol{W}_{K,h}, \boldsymbol{W}_{V,h})$$

Dosovitskiy et al. 2021

Processing Images with ViT (TF Block)

Recall: the Vision Transformer (ViT) processes images as a **sequence of patches**.



$$\underbrace{f^{\ell}}_{\text{TF layer}}$$

$$\boldsymbol{Z}^{\ell+1/2} = \mathtt{MHSA}(\mathtt{LN}(\boldsymbol{Z}^{\ell})) + \boldsymbol{Z}^{\ell}$$

$$f^{\ell}(\boldsymbol{Z}^{\ell}) = \mathtt{MLP}(\mathtt{LN}(\boldsymbol{Z}^{\ell+1/2})) + \boldsymbol{Z}^{\ell+1/2}$$

$$\mathtt{MLP}(\boldsymbol{Z}) = \boldsymbol{W}_{\mathrm{down}}\sigma(\boldsymbol{W}_{\mathrm{up}}\boldsymbol{Z})$$

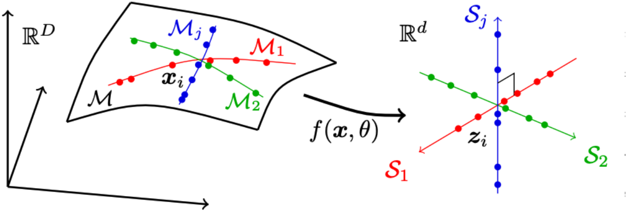
Dosovitskiy et al. 2021

Recall (Yi's Lecture): Coding Rate Reduction

Rate reduction (for non-tokenized data):

$$\Delta R(\boldsymbol{Z} \mid \boldsymbol{\Pi}) := R(\boldsymbol{Z}) - \underbrace{\sum_{k=1}^{K} \frac{n_k}{n} R(\boldsymbol{Z}_k)}_{R_c(\boldsymbol{Z}|\boldsymbol{\Pi})}.$$

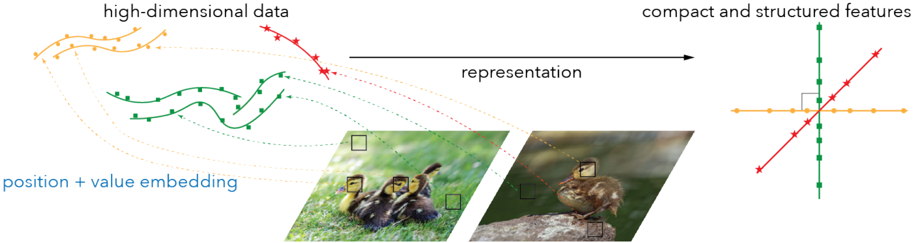
Promotes compression of *features (of samples)* against class-wise (learned) low-rank GMM.



Rate Reduction for Token Sequences

Rate reduction for tokenized data:
Parameterize the GMM covariances $\boldsymbol{\Sigma}_k = \boldsymbol{U}_k \boldsymbol{U}_k^\top$.

$$\Delta R(\boldsymbol{Z} \mid \underbrace{\boldsymbol{U}_{[K]}}_{:=(\boldsymbol{U}_k)_{k=1}^K}) := R(\boldsymbol{Z}) - \underbrace{\sum_{k=1}^{K} R(\boldsymbol{U}_k^\top \boldsymbol{Z})}_{:=R_c(\boldsymbol{Z} \mid \boldsymbol{U}_{[K]})}$$

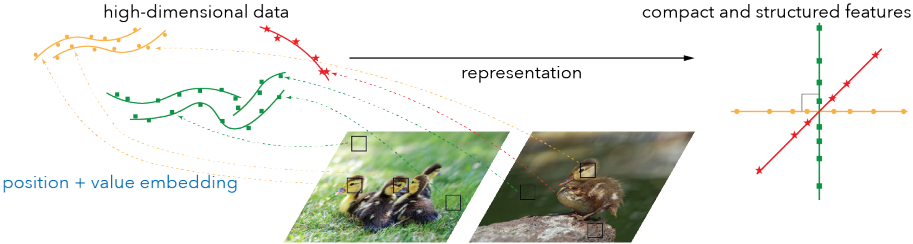


Sparse Rate Reduction

To be maximally structured, ask $\boldsymbol{Z}$ (hence $\boldsymbol{U}_k$) to be *sparse*!

Objective to maximize: **Sparse Rate Reduction**

$$\mathrm{SRR}(\boldsymbol{Z} \mid \boldsymbol{U}_{[K]}) := R(\boldsymbol{Z}) - R_c(\boldsymbol{Z} \mid \boldsymbol{U}_{[K]}) - \lambda \|\boldsymbol{Z}\|_1$$



Unrolling the Sparse Rate Reduction

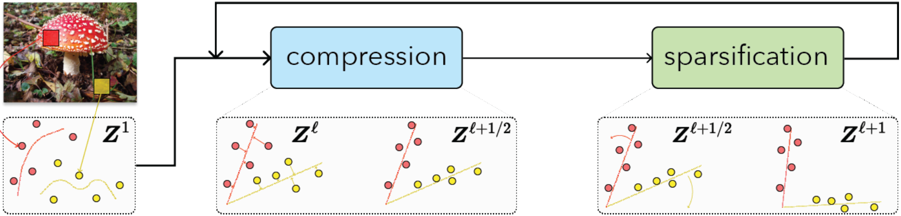
Proposed optimization strategy:

Two-step (prox-like) iteration.

$$\boldsymbol{Z}^{\ell} \mapsto \boldsymbol{Z}^{\ell+1/2} \mapsto \boldsymbol{Z}^{\ell+1}$$

$$\boldsymbol{Z}^{\ell+1/2} \approx \boldsymbol{Z}^{\ell} - \kappa \nabla_{\boldsymbol{Z}} R_c(\boldsymbol{Z}^{\ell} \mid \boldsymbol{U}_{[K]}^{\ell}) \qquad \text{(compression)}$$

$$\boldsymbol{Z}^{\ell+1} \approx \operatorname*{arg\,max}_{\boldsymbol{Z} \colon \boldsymbol{Z}^{\ell+1/2} = \boldsymbol{D}^{\ell}\boldsymbol{Z}} \{R(\boldsymbol{Z}) - \lambda \|\boldsymbol{Z}\|_1\} \qquad \text{(sparsification)}$$



Parameters: $(\boldsymbol{U}_k^{\ell})_{k=1}^{K} \subseteq \mathbb{R}^{d \times p}$, $\boldsymbol{D}^{\ell} \in \mathbb{R}^{d \times d}$.

Gradient of Compression Objective

Define $\alpha := p/(n\varepsilon^2)$.
If $(\boldsymbol{U}_k)_{k=1}^K \approx$ orthogonal + p/w $\approx$ orthogonal + $\approx$ support $\boldsymbol{Z}$:

$$
\begin{aligned}
\nabla_{\boldsymbol{Z}} R_c(\boldsymbol{Z} \mid \boldsymbol{U}_{[K]}) &= \sum_{k=1}^{K} \alpha(\boldsymbol{U}_k \boldsymbol{U}_k^\top \boldsymbol{Z})(\boldsymbol{I}_n + \alpha(\boldsymbol{U}_k^\top \boldsymbol{Z})^\top (\boldsymbol{U}_k^\top \boldsymbol{Z}))^{-1} \\
&\approx \sum_{k=1}^{K} \alpha \boldsymbol{U}_k(\boldsymbol{U}_k^\top \boldsymbol{Z})(\boldsymbol{I}_d - \alpha(\boldsymbol{U}_k^\top \boldsymbol{Z})^\top (\boldsymbol{U}_k^\top \boldsymbol{Z})) \\
&= \alpha \left[\left(\sum_{k=1}^{K} \boldsymbol{U}_k \boldsymbol{U}_k^\top \right) \boldsymbol{Z} - \alpha \sum_{k=1}^{K} \boldsymbol{U}_k (\boldsymbol{U}_k^\top \boldsymbol{Z})(\boldsymbol{U}_k^\top \boldsymbol{Z})^\top (\boldsymbol{U}_k^\top \boldsymbol{Z}) \right] \\
&\approx \alpha \left[\boldsymbol{Z} - \alpha \sum_{k=1}^{K} \boldsymbol{U}_k (\boldsymbol{U}_k^\top \boldsymbol{Z})(\boldsymbol{U}_k^\top \boldsymbol{Z})^\top (\boldsymbol{U}_k^\top \boldsymbol{Z}) \right]
\end{aligned}
$$

Gradient shaping/"non-parametric autoregression":

$$
\nabla_{\boldsymbol{Z}} R_c(\boldsymbol{Z}) \approx \alpha \left[\boldsymbol{Z} - \alpha \sum_{k=1}^{K} \boldsymbol{U}_k (\boldsymbol{U}_k^\top \boldsymbol{Z}) \operatorname{softmax}\left\{ (\boldsymbol{U}_k^\top \boldsymbol{Z})^\top (\boldsymbol{U}_k^\top \boldsymbol{Z}) \right\} \right]
$$

Multi-head Subspace Self-Attention

$$\nabla_{\boldsymbol{Z}} R_c(\boldsymbol{Z}) \approx \alpha \left[\boldsymbol{Z} - \alpha \sum_{k=1}^{K} \boldsymbol{U}_k (\boldsymbol{U}_k^\top \boldsymbol{Z}) \operatorname{softmax}\left\{ (\boldsymbol{U}_k^\top \boldsymbol{Z})^\top (\boldsymbol{U}_k^\top \boldsymbol{Z}) \right\} \right]$$

Multi-head Subspace Self-Attention (MSSA):

$$\mathrm{MSSA}(\boldsymbol{Z} \mid \boldsymbol{U}_{[K]}) := \alpha \left[\boldsymbol{U}_1, \ldots, \boldsymbol{U}_K \right] \begin{bmatrix} (\boldsymbol{U}_1^\top \boldsymbol{Z}) \operatorname{softmax}\{ (\boldsymbol{U}_1^\top \boldsymbol{Z})^\top (\boldsymbol{U}_1^\top \boldsymbol{Z}) \} \\ \vdots \\ (\boldsymbol{U}_K^\top \boldsymbol{Z}) \operatorname{softmax}\{ (\boldsymbol{U}_K^\top \boldsymbol{Z})^\top (\boldsymbol{U}_K^\top \boldsymbol{Z}) \} \end{bmatrix}$$

$$\boxed{\boldsymbol{Z}^{\ell+1/2} := \underbrace{(1 - \alpha\kappa)\boldsymbol{Z}^{\ell}}_{\text{residual}} + \underbrace{\alpha\kappa \, \mathrm{MSSA}(\boldsymbol{Z}^{\ell} \mid \boldsymbol{U}_{[K]}^{\ell})}_{\text{attention-like}}}$$

Iterative Shrinkage-Thresholding Block

If $\boldsymbol{D}^\ell \approx$ complete incoherent dictionary then

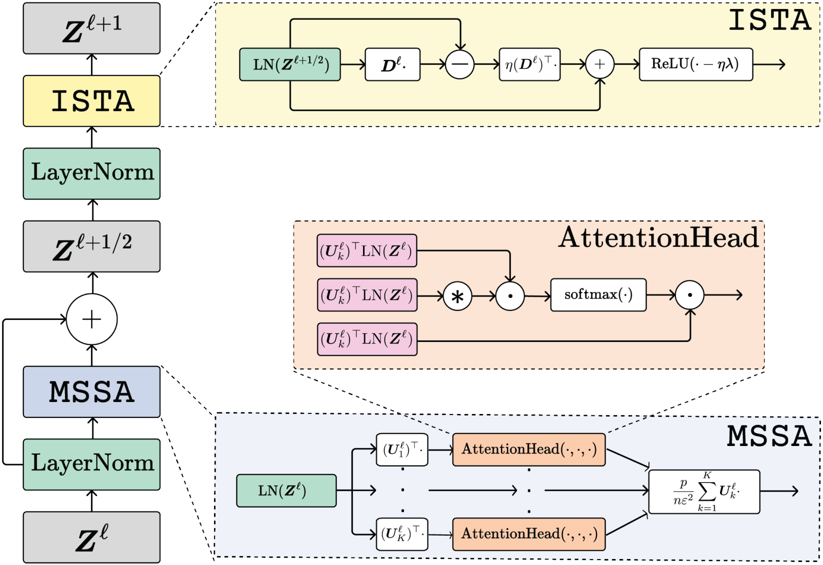
$$\boldsymbol{Z}^{\ell+1/2} \approx \boldsymbol{D}^\ell \boldsymbol{Z} \implies R(\boldsymbol{Z}) \approx R(\boldsymbol{Z}^{\ell+1/2})$$

Can simplify the prox-like step:

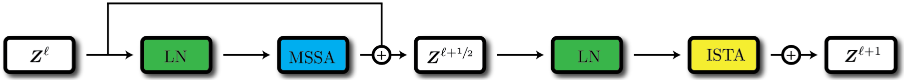
$$\begin{aligned}\boldsymbol{Z}^{\ell+1} &\approx \operatorname*{arg\,max}_{\boldsymbol{Z}\colon \boldsymbol{Z}^{\ell+1/2} \approx \boldsymbol{D}^\ell \boldsymbol{Z}} \{R(\boldsymbol{Z}) - \lambda \|\boldsymbol{Z}\|_1\} \approx \operatorname*{arg\,min}_{\substack{\boldsymbol{Z} \\ \boldsymbol{Z}^{\ell+1/2} \approx \boldsymbol{D}^\ell \boldsymbol{Z}}} \|\boldsymbol{Z}\|_1 \\ &\approx \operatorname*{arg\,min}_{\boldsymbol{Z}} \left\{ \frac{1}{2} \|\boldsymbol{Z}^{\ell+1/2} - \boldsymbol{D}^\ell \boldsymbol{Z}\|_2^2 + \lambda' \|\boldsymbol{Z}\|_1 \right\}\end{aligned}$$

$$\boxed{\boldsymbol{Z}^{\ell+1} := \mathrm{ISTA}(\boldsymbol{Z}^{\ell+1/2}) := \mathrm{ProxGD}(\underbrace{\boldsymbol{Z}^{\ell+1/2}}_{\text{iterate}}, \underbrace{\boldsymbol{Z}^{\ell+1/2}}_{\text{target}}, \underbrace{\boldsymbol{D}^\ell}_{\text{dict.}})}$$

CRATE Architecture

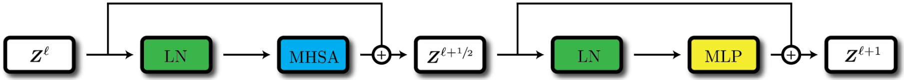


Comparing CRATE and Regular Transformer

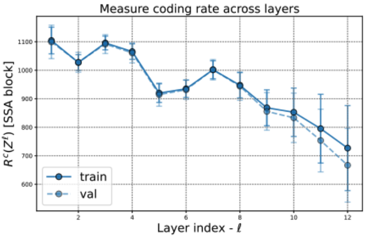


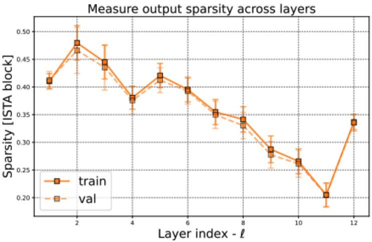
Three practical differences:

- MSSA sets $W_{Q,k} = W_{K,k} = W_{V,k} = U_k^\top$
- ISTA sets $W_{\mathrm{up}} = W_{\mathrm{down}}^\top = D$
- In ISTA the residual connection is moved inside ReLU



Do CRATE Models Behave According to Theory?





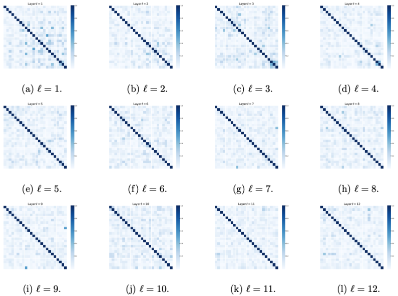


Figure 17: We visualize the $[U_1^\ell, \ldots, U_K^\ell]^* [U_1^\ell, \ldots, U_K^\ell] \in \mathbb{R}^{pK \times pK}$ at different layers. The (i, j)-th

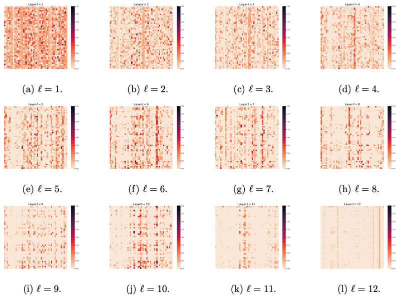


Figure 16: Visualizing layer-wise token Z^ℓ representations at each layer ℓ. To enhance the visual

Can CRATE Perform Well in Practice?

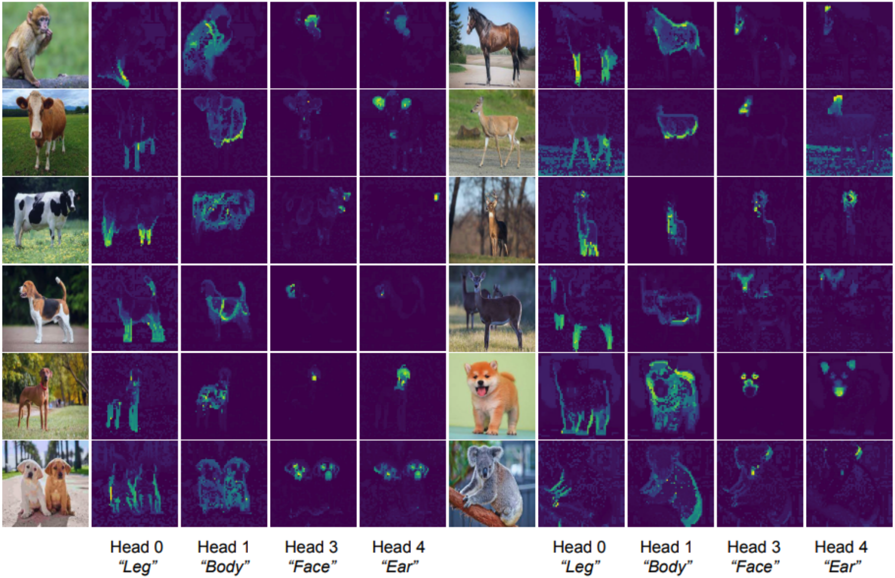
Vision:

Model	CRATE-T	CRATE-S	CRATE-B	CRATE-L	ViT-T	ViT-S
# parameters	6.09M	13.12M	22.80M	77.64M	5.72M	22.05M
ImageNet-1K	66.7	69.2	70.8	71.3	71.5	72.4
ImageNet-1K ReaL	74.0	76.0	76.5	77.4	78.3	78.4
CIFAR10	95.5	96.0	96.8	97.2	96.6	97.2
CIFAR100	78.9	81.0	82.7	83.6	81.8	83.2
Oxford Flowers-102	84.6	87.1	88.7	88.3	85.1	88.5
Oxford-IIIT-Pets	81.4	84.9	85.3	87.4	88.5	88.6

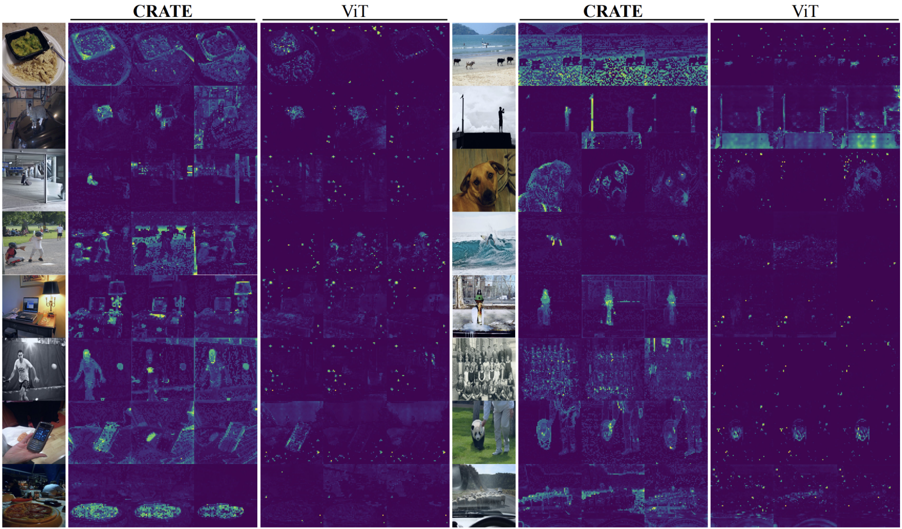
Text:

	#parameters	**OWT**	**LAMBADA**	**WikiText**	**PTB**	**Avg**
GPT2-Base	124M	2.85	4.12	3.89	4.63	3.87
GPT2-Small	64M	3.04	4.49	4.31	5.15	4.25
CRATE-GPT2-Base	60M	3.37	4.91	4.61	5.53	4.61

Interpretability and Emergent Segmentation



Interpretability and Emergent Segmentation



Performance: Semantic Segmentation

Setup: Zero-shot semantic segmentation with CLIP + `MSSA`.

Top left: original image. *Bottom left:* CLIP-ViT features. *Right:* CRATE-CLIP features.

$\approx 5\%$ **better mIoU score than previous approaches!**

Wang et al. 2024

Design Choices in CRATE

More effective way to do sparsification?

What if we use multiple prox iterations with *overcomplete (wide) dictionary* $\boldsymbol{D}^{\ell} \in \mathbb{R}^{d \times m}$, $m > d$?

$\implies$ different architecture!

CRATE-α Sparsification Block

$$\begin{aligned} \boldsymbol{Z}^{\ell+1} &:= \mathrm{ODL}(\boldsymbol{Z}^{\ell+1/2}) \\ &= \boldsymbol{D}^{\ell}\, \mathrm{ProxGD}(\mathrm{ProxGD}(\mathbf{0}, \boldsymbol{Z}^{\ell+1/2}, \boldsymbol{D}^{\ell}), \boldsymbol{Z}^{\ell+1/2}, \boldsymbol{D}^{\ell}) \end{aligned}$$

Performance of CRATE-α

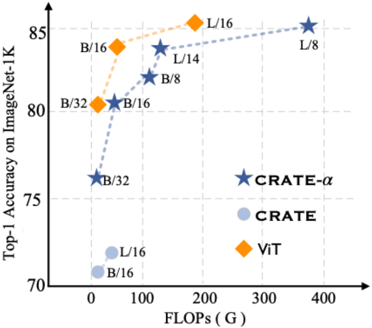
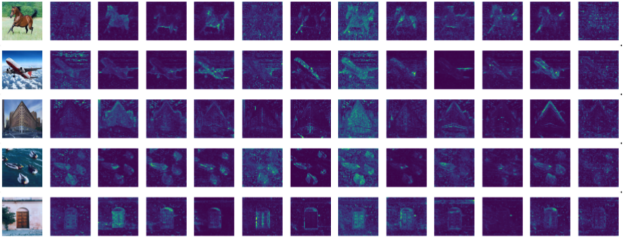


Figure 5: **Visualization of segmentation on COCO val2017 [20] with MaskCut [43].** (*Top row*)

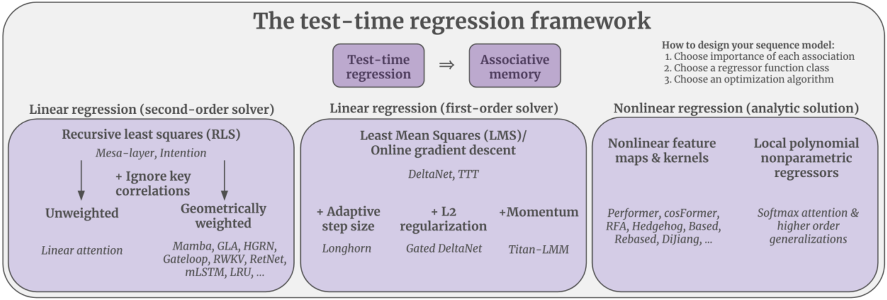
Table 4: The comparison between CRATE and CRATE-α on the NLP task using the OpenWebText dataset.

	GPT-2-base	CRATE-base	CRATE-α-small	CRATE-α-base
Model size	124M	60M	57M	120M
CE val loss	2.85	3.37	3.28	3.14



Aside: Network Operators as Optimization Primitives

- Optimization gives blocks *similar to* blocks in transformer
- Recent work derives linear attention + similar operators as *exact* optimization steps on regression objectives w.r.t. $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}$



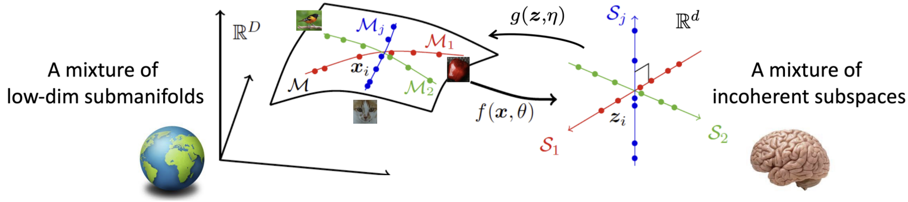
Wang, Shi, and Fox 2025

Outline

1. Analytical Models
   - Geometry and Sparsity
   - Optimization and Neural Networks
2. Deep Representation Learning
   - Transformers for Visual Data
   - Objectives for Representation Learning
   - Unrolled Optimization for Representation Learning
   - Compression and Self-Attention
   - Sparsification and MLP
   - Coding Rate Reduction Transformer
   - Experimental Results on CRATE
3. Conclusions for the Tutorial

Take-Home Message: Low-Dim Structures are Ubiquitous!

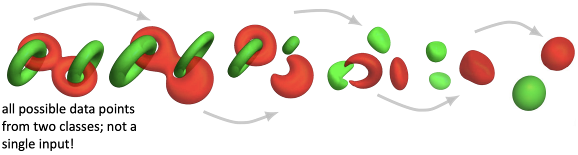
In this tutorial, we have emphasized:

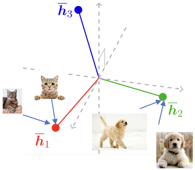


The objective of learning:
Identify low-dim. distributions in sensed data of the world
and *transform* to a compact and structured representation.

All deep networks are simply a means to an end!

S2: Understanding Low-Dimensional Structures in Representation Learning

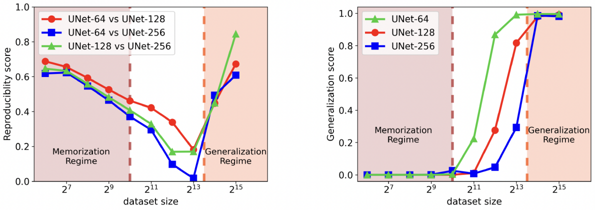




Given enough data and ability to optimize: *inevitable emergence* of low-dim structures in trained deep networks!

Implications for parameter efficiency, transfer learning, ...

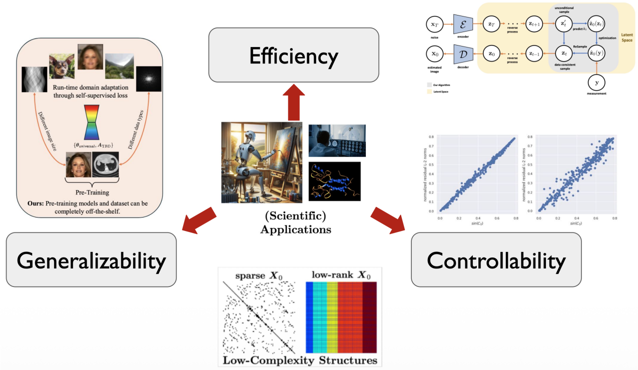
S3: Understanding Low-Dimensional Structures in Diffusion Generative Models



Given enough data and ability to optimize: *inevitable emergence* of low-dim structures in trained deep networks!

Implications for efficiency, controllability, generalizability

S3: Understanding Low-Dimensional Structures in Diffusion Generative Models

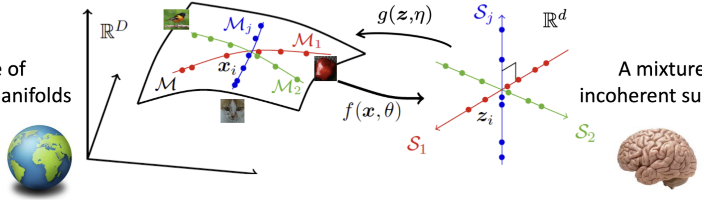


Given enough data and ability to optimize: *inevitable emergence* of low-dim structures in trained deep networks!

Implications for efficiency, controllability, generalizability

S4: Bottom-Up Understanding of Deep Networks for Vision

A mixture of
low-dim submanifolds



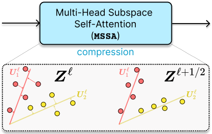
A mixture of
incoherent subspaces

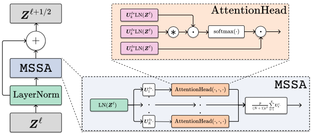
The objective of learning:
Identify low-dim. distributions in sensed data of the world
and *transform* to a compact and structured representation.

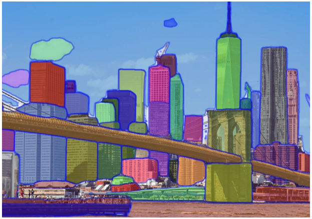
Once we clarify this objective of learning, we can **design networks** to explicitly achieve these functions!

S4: Bottom-Up Understanding of Deep Networks for Vision

1. **Design** $\varphi(z)$ s.t. z optimal $\iff$ good representation
2. **Construct** f **via incremental optimization** of φ
3. Learn any parameters of f from data







More *interpretable* (derivations!) and *less superfluous pieces*!

Kirillov et al. 2023

Thank You! Questions?

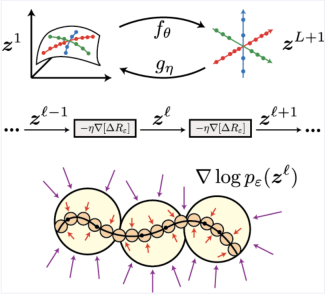
Learning Deep Representations of Data Distributions

Sam Buchanan · Druv Pai · Peng Wang · Yi Ma

A modern fully open-source textbook exploring why and how deep neural networks learn compact and information-dense representations of high-dimensional real-world data.

```
 @book{ldrdd2025,
  title={Learning Deep Representations of Data Distributions},
  author={Buchanan, Sam and Pai, Druv and Wang, Peng and Ma,
Yi},
  month=aug,
  year={2025},
  publisher={Online},
  note={\url{https://ma-lab-berkeley.github.io/deep-
representation-learning-book/}.}
}
```

Read the Book (HTML) | Read the Book (PDF)



Version 1.0
Last Updated: October 17, 2025

https://ma-lab-berkeley.github.io/deep-representation-learning-book