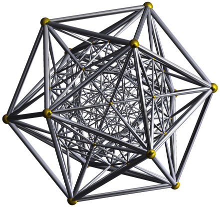
ICASSP 2022 Short Course One on
Low-Dimensional Models for High-Dimensional Data

Lecture 5: Design Deep Networks for Pursuing Low-Dimensional Structures

Sam Buchanan, Yi Ma, Qing Qu
John Wright, Yuqian Zhang, Zhihui Zhu

May 26, 2022



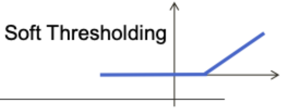
Outline

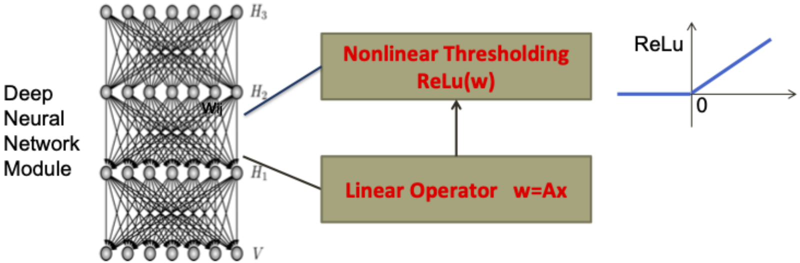
ISTA: Sparse Recovery via ℓ^1 (Wright and Ma, 2022)

CONTEXT – Basic algorithm (ISTA)

Algorithm 8.1 Iterative Soft-Thresholding Algorithm (ISTA) for BPDN

1: **Problem:** $\min_{\boldsymbol{x}} \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2 + \lambda\|\boldsymbol{x}\|_1$, given $\boldsymbol{y} \in \mathbb{R}^d$, $\boldsymbol{A} \in \mathbb{R}^{d \times n}$.
2: **Input:** $\boldsymbol{x}_0 \in \mathbb{R}^n$ and $L \geq \lambda_{\max}(\boldsymbol{A}^T\boldsymbol{A})$.
3: **while** $\boldsymbol{x}_k$ not converged $(k = 1, 2, \ldots)$ **do**
4: $\quad \boldsymbol{w}_k \leftarrow \boldsymbol{x}_k - \frac{1}{L}\boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{x}_k - \boldsymbol{y})$.
5: $\quad \boldsymbol{x}_{k+1} \leftarrow \mathrm{soft}(\boldsymbol{w}_k, \lambda/L)$.
6: **end while**
7: **Output:** $\boldsymbol{x}_\star \leftarrow \boldsymbol{x}_k$.





Learned ISTA (Gregor and LeCun, ICML 2010)

CONTEXT – Learned ISTA (LISTA)

If only interested in one instance: $y = Ax$ AND with many training data: $\{(y_i, x_i)\}$.
We can optimize the optimization path of ISTA using supervised learning:

Algorithm 3 LISTA::fprop

LISTA :: **fprop**(X, Z, W_e, S, θ)
;; Arguments are passed by reference.
;; variables $Z(t)$, $C(t)$ and B are saved for bprop.
$B = W_e X$; $Z(0) = h_\theta(B)$
for $t = 1$ to T **do**
  $C(t) = B + SZ(t-1)$
  $Z(t) = h_\theta(C(t))$
end for
$Z = Z(T)$

Algorithm 4 LISTA::bprop

LISTA :: **bprop**$(Z^*, X, Z, W_e, S, \theta, \delta X, \delta W_e, \delta S, \delta\theta)$
;; Arguments are passed by reference.
;; Variables $Z(t)$, $C(t)$, and B were saved in fprop.
Initialize: $\delta B = 0$; $\delta S = 0$; $\delta\theta = 0$
$\delta Z(T) = (Z(T) - Z^*)$
for $t = T$ down to 1 **do**
  $\delta C(t) = h'_\theta(C(t)).\delta Z(t)$
  $\delta\theta = \delta\theta - \text{sign}(C(t)).\delta C(t)$
  $\delta B = \delta B + \delta C(t)$
  $\delta S = \delta S + \delta C(t) Z(t-1)^T$
  $\delta Z(t-1) = S^T \delta C(t)$
end for
$\delta B = \delta B + h'_\theta(B).\delta Z(0)$
$\delta\theta = \delta\theta - \text{sign}(B).h'_\theta(B)\delta Z(0)$
$\delta W_e = \delta B X^T$; $\delta X = W_e^T \delta B$

Learning fast approximations of sparse coding, K. Gregor and Y. LeCun, ICML 2010.

Sparse Dictionary Learning via ℓ^4: the MSP Algorithm

Solve a sparsifying $\boldsymbol{A}$ for $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{X}$ by solving the program:

$$\max_{\boldsymbol{A}} \phi(\boldsymbol{A}) \doteq \|\boldsymbol{A}\boldsymbol{Y}\|_4^4, \quad \boldsymbol{A} \in O(n;\mathbb{R}), \tag{1}$$

via projected gradient descent: $\boldsymbol{A}_{t+1} = \mathcal{P}_{O(n)}[\boldsymbol{A}_t + \alpha \nabla_{\boldsymbol{A}} \phi(\boldsymbol{A}_t)]$.

The Matching, Stretching, and Projection (MSP) Algorithm:

1: **Initialize** $\boldsymbol{A}_0 \in O(n,\mathbb{R})$ — initialize $\boldsymbol{A}_0$ for iteration
2: **for** $t = 0, 1, \ldots$
3: $\quad \nabla_{\boldsymbol{A}} \phi(\boldsymbol{A}_t) = 4(\boldsymbol{A}_t \boldsymbol{Y})^{\circ 3} \boldsymbol{Y}^*$ — Matching and Stretching
4: $\quad \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^* = \mathsf{SVD}[\nabla_{\boldsymbol{A}} \phi(\boldsymbol{A}_t)]$
5: $\quad \boldsymbol{A}_{t+1} = \boldsymbol{U}\boldsymbol{V}^*$ — Project $\boldsymbol{A}$ onto orthogonal group
6: **end for**
7: **output:** $\boldsymbol{A}_\star$.

Complete dictionary learning via L4-Norm maximization over the orthogonal group, Zhai et. al. JMLR 2020.

"Deep Networks" as Power Iteration for Low-Dim

"Fixed point" (not gradient descent) interpretation:

$$\boldsymbol{A}_{t+1} = \mathcal{P}_{O(n)}[\nabla_{\boldsymbol{A}} \phi(\boldsymbol{A}_t)] = \mathcal{P}_{O(n)}[(\boldsymbol{A}_t \boldsymbol{Y})^{\circ 3} \boldsymbol{Y}^*].$$

Define "layer" operators and states iteratively:

$$\delta \boldsymbol{A}_{t+1} \doteq \boldsymbol{A}_{t+1} \boldsymbol{A}_t^* \quad \text{and} \quad \boldsymbol{Z}_{t+1} \doteq \boldsymbol{A}_{t+1} \boldsymbol{Y} = \delta \boldsymbol{A}_{t+1} \boldsymbol{Z}_t.$$

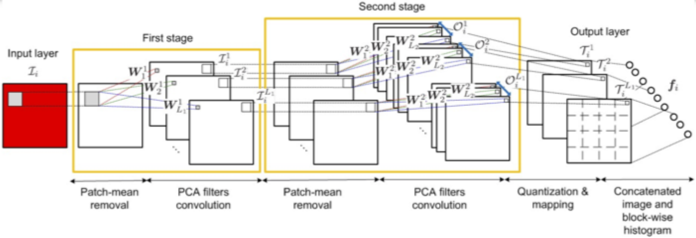
Forward-constructed "deep network" interpretation:

$$\delta \boldsymbol{A}_{t+1} = \mathcal{P}_{O(n)}[(\boldsymbol{Z}_t)^{\circ 3} \boldsymbol{Z}_t^*], \quad \boldsymbol{X} \leftarrow \delta \boldsymbol{A}_{t+1} \delta \boldsymbol{A}_t \cdots \delta \boldsymbol{A}_1 \boldsymbol{Y}.$$

Table: Fixed Points: PCA (Power iteration), ICA (FastICA), and DL (MSP).

	Objectives	Constraint Sets	Algorithms
Power Iter.	$\varphi(\boldsymbol{a}) \doteq \frac{1}{2} \\|\boldsymbol{a}^* \boldsymbol{Y}\\|_2^2$	$\boldsymbol{w} \in \mathbb{S}^{n-1}$	$\boldsymbol{a}_{t+1} = \mathcal{P}_{\mathbb{S}^{n-1}} [\nabla_{\boldsymbol{a}} \varphi(\boldsymbol{a}_t)]$
FastICA	$\psi(\boldsymbol{a}) \doteq \frac{1}{4} \mathrm{kurt}[\boldsymbol{a}^* \boldsymbol{y}]$	$\boldsymbol{a} \in \mathbb{S}^{n-1}$	$\boldsymbol{a}_{t+1} = \mathcal{P}_{\mathbb{S}^{n-1}} [\nabla_{\boldsymbol{a}} \psi(\boldsymbol{a}_t)]$
MSP	$\phi(\boldsymbol{A}) \doteq \frac{1}{4} \\|\boldsymbol{A}^* \boldsymbol{Y}\\|_4^4$	$\boldsymbol{A} \in St(k, n; \mathbb{R})$	$\boldsymbol{A}_{t+1} = \mathcal{P}_{St(k,n;\mathbb{R})} [\nabla_{\boldsymbol{A}} \phi(\boldsymbol{A}_t)]$

The Simplest Network Models - PCANet



Two layers forward-constructed without back-propagation:

- **simplest data-adaptive mapping (PCA),**
- **simplest nonlinear activation (binary),**
- **simplest pooling (histogram).**

Recognition rates (%) on FERET dataset.

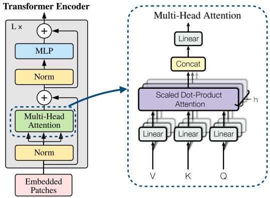
Probe sets	*Fb*	*Fc*	*Dup-I*	*Dup-II*	Avg.
LBP [18]	93.00	51.00	61.00	50.00	63.75
DMMA [25]	98.10	98.50	81.60	83.20	89.60
P-LBP [21]	98.00	98.00	90.00	85.00	92.75
POEM [26]	99.60	99.50	88.80	85.00	93.20
G-LQP [27]	99.90	100	93.20	91.00	96.03
LGBP-LGXP [28]	99.00	99.00	94.00	93.00	96.25
sPOEM+POD [29]	99.70	100	94.90	94.00	97.15
GOM [30]	99.90	100	95.70	93.10	97.18
PCANet-1 (Trn. CD)	99.33	99.48	88.92	84.19	92.98
PCANet-2 (Trn. CD)	99.67	99.48	95.84	94.02	97.25
PCANet-1	99.50	98.97	89.89	86.75	93.78
PCANet-2	99.58	100	95.43	94.02	97.26

PCANet, Chan and Ma et. al. in IEEE Trans. On Image Processing, 2015

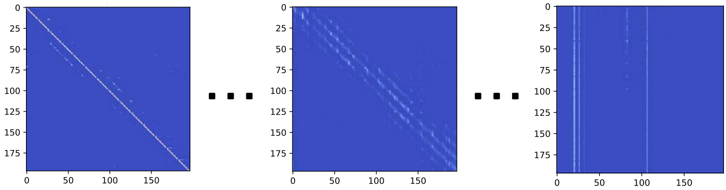
PCANet: A Simple deep learning baseline for image classification? Chan et. al. TIP, 2015.

Pursuit of Sparsity via Transformer Layers

[Vaswani et al., 2017, Dosovitskiy et al., 2020]



"Attention" matrix

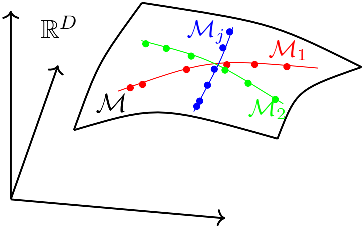


Earlier Layers ⟷ Later Layers

$$A = \mathrm{Softmax}\left(qk^{\top}/\sqrt{D_h}\right)$$

High-Dim Data with Mixed Nonlinear Low-Dim Structures

Figure: **High-dimensional Real-World Data**: data samples $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m]$ in $\mathbb{R}^D$ lying on a mixture of low-dimensional submanifolds $\boldsymbol{X} \subset \cup_{j=1}^{k} \mathcal{M}_j \subset \mathbb{R}^D$.



The main objective of learning from (samples of) such real-world data:

seek a most compact and structured representation of the data.

Fitting Class Labels via a Deep Network

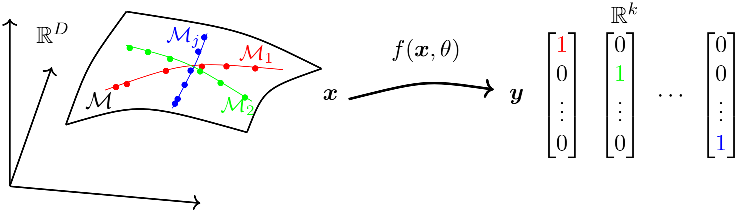


Figure: **Black Box DNN for Classification:** $\boldsymbol{y}$ is the class label of $\boldsymbol{x}$ represented as a "one-hot" vector in $\mathbb{R}^k$. To learn a nonlinear mapping $f(\cdot, \theta) : \boldsymbol{x} \mapsto \boldsymbol{y}$, say modeled by a deep network, using cross-entropy (CE) loss.

$$\min_{\theta \in \Theta} \mathsf{CE}(\theta, \boldsymbol{x}, \boldsymbol{y}) \doteq -\mathbb{E}[\langle \boldsymbol{y}, \log[f(\boldsymbol{x}, \theta)]\rangle] \approx -\frac{1}{m}\sum_{i=1}^{m}\langle \boldsymbol{y}_i, \log[f(\boldsymbol{x}_i, \theta)]\rangle. \tag{2}$$

Prevalence of **neural collapse** *during the terminal phase of deep learning training*, Papyan, Han, and Donoho, 2020.

Fitting Class Labels via a Deep Network

In a supervised setting, using cross-entropy (CE) loss:

$$\min_{\theta \in \Theta} \mathrm{CE}(\theta, \boldsymbol{x}, \boldsymbol{y}) \doteq -\mathbb{E}[\langle \boldsymbol{y}, \log[f(\boldsymbol{x}, \theta)] \rangle] \approx -\frac{1}{m} \sum_{i=1}^{m} \langle \boldsymbol{y}_i, \log[f(\boldsymbol{x}_i, \theta)] \rangle. \qquad (3)$$

Issues (an elephant in the room):

- A large deep neural networks can **fit arbitrary data and labels**.
- Statistical and geometric meaning of internal features **not clear**.
- Task/data-dependent and **not robust nor truly invariant**.

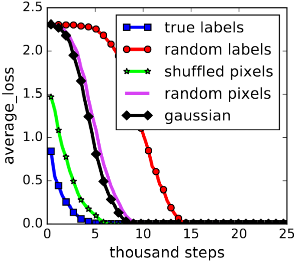


Figure: [Zhang et al, ICLR'17]

What did machines actually "learn" from doing this?

In terms of interpolating, extrapolating, or representing the data?

A Hypothesis: Information Bottleneck

[Tishby & Zaslavsky, 2015]

A feature mapping $f(\boldsymbol{x}, \theta)$ and a classifier $g(\boldsymbol{z})$ trained for downstream classification:

$$\boldsymbol{x} \xrightarrow{f(\boldsymbol{x},\theta)} \boldsymbol{z}(\theta) \xrightarrow{g(\boldsymbol{z})} \boldsymbol{y}.$$

The IB Hypothesis: Features learned in a deep network trying to

$$\max_{\theta \in \Theta} \mathsf{IB}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}(\theta)) \doteq I(\boldsymbol{z}(\theta), \boldsymbol{y}) - \beta I(\boldsymbol{x}, \boldsymbol{z}(\theta)), \quad \beta > 0, \tag{4}$$

where $I(\boldsymbol{z}, \boldsymbol{y}) \doteq H(\boldsymbol{z}) - H(\boldsymbol{z}|\boldsymbol{y})$ and $H(\boldsymbol{z})$ is the entropy of $\boldsymbol{z}$.

- **Minimal** informative features $\boldsymbol{z}$ that most correlate with the label $\boldsymbol{y}$
- Task and label-dependent, consequently sacrificing generalizability, robustness, or transferability

Gap between Theory and Practice (a Bigger Elephant)

For high-dimensional real data,
many statistical and information-theoretic concepts such as entropy, mutual information, K-L divergence, and maximum likelihood:

- curse of **dimensionality** for computation.
- ill-posed for **degenerate** distributions.
- lack guarantees with **finite** (or non-asymptotic) samples.

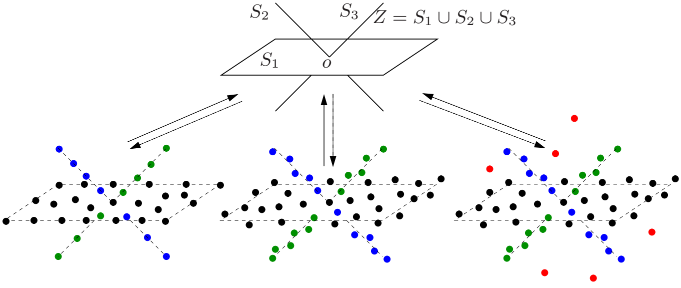
Reality check: principled formulations are replaced with approximate bounds, grossly simplifying assumptions, heuristics, even *ad hoc* tricks and hacks.

How to provide any performance guarantees at all?

A Principled Computational Approach

For high-dim data with mixed **low-dim** structures:

learn to compress, and compress to learn!

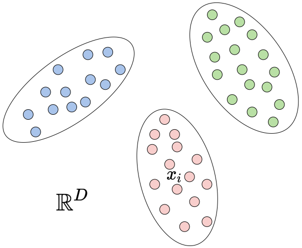


a) sample points   b) noisy samples   c) noisy samples with outliers

Generalized PCA for mixture of subspaces [Vidal, Ma, and Sastry, 2005]

1. Clustering Mixed Data (Interpolation)

Assume data $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m]$ are i.i.d. samples from a mixture of distributions: $p(\boldsymbol{x}, \theta) = \sum_{j=1}^{k} \pi_j p_j(\boldsymbol{x}, \theta)$.



Classic approaches to cluster the data: the maximum-likelihood (ML) estimate via Expectation Maximization (EM):

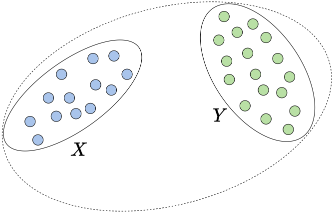
$$\max_{\theta,\pi} \mathbb{E}\Big[\log\Big(\sum_{j=1}^{k} \pi_j p_j(\boldsymbol{x}, \theta)\Big)\Big] \approx \max_{\theta,\pi} \frac{1}{m} \sum_{i=1}^{m} \log\Big(\sum_{j=1}^{k} \pi_j p_j(\boldsymbol{x}_i, \theta)\Big).$$

Difficulties: ML is not well-defined when distributions are degenerate.

Clustering via Compression

[Yi Ma, Harm Derksen, Wei Hong, and John Wright, TPAMI'07]

A Fundamental Idea:
Data belong to mixed low-dim structures should be compressible.



Cluster Criterion:
Whether the number of binary bits required to store the data:

$$\#\text{bits}(\boldsymbol{X} \cup \boldsymbol{Y}) \ \geq \ \#\text{bits}(\boldsymbol{X}) + \#\text{bits}(\boldsymbol{Y})?$$

"The whole is greater than the sum of the parts."
– Aristotle, 320 BC

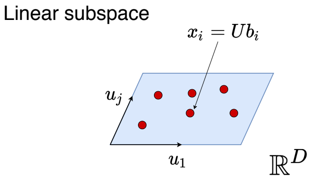
Coding Length Function for Subspace-Like Data

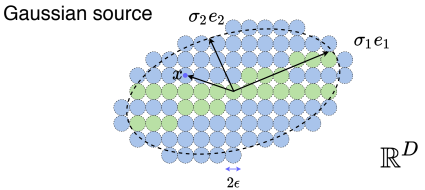
Theorem (Ma, TPAMI'07)

The number of bits needed to encode data $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m] \in \mathbb{R}^{D\times m}$ *up to a precision* $\|\boldsymbol{x} - \hat{\boldsymbol{x}}\|_2 \leq \epsilon$ *is bounded by:*

$$L(\boldsymbol{X}, \epsilon) \doteq \left(\frac{m+D}{2}\right) \log\det\left(\boldsymbol{I} + \frac{D}{m\epsilon^2}\boldsymbol{X}\boldsymbol{X}^\top\right).$$

This can be derived from constructively quantifying SVD of $\boldsymbol{X}$ or by sphere packing vol$(\boldsymbol{X})$ as samples of a noisy Gaussian source.

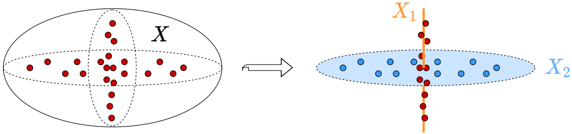




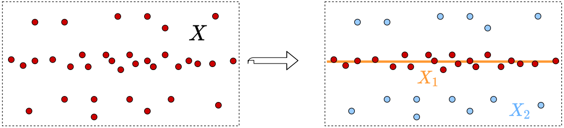
Cluster to Compress

$$L(\boldsymbol{X}) \;\geq\; L^c(\boldsymbol{X}) \doteq L(\boldsymbol{X}_1) + L(\boldsymbol{X}_2) + H(|\boldsymbol{X}_1|, |\boldsymbol{X}_2|)?$$

partitioning:



sifting:



A Greedy Algorithm

Seek a partition of the data $\boldsymbol{X} \to [\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_k]$ such that

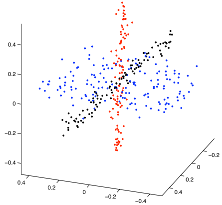
$$\min L^c(\boldsymbol{X}) \doteq L(\boldsymbol{X}_1) + \cdots + L(\boldsymbol{X}_k) + H(|\boldsymbol{X}_1|, \ldots, |\boldsymbol{X}_k|).$$

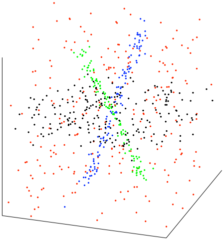
Optimize with a *bottom-up pair-wise* merging algorithm [Ma, TPAMI'07]:

input: the data $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m] \in \mathbb{R}^{D \times m}$ and a distortion $\epsilon^2 > 0$.
initialize $\mathcal{S}$ as a set of sets with a single datum $\{S = \{\boldsymbol{x}\} \mid \boldsymbol{x} \in \boldsymbol{X}\}$.
while $|\mathcal{S}| > 1$ **do**
choose distinct sets $S_1, S_2 \in \mathcal{S}$ such that
$L^c(S_1 \cup S_2) - L^c(S_1, S_2)$ is minimal.
if $L^c(S_1 \cup S_2) - L^c(S_1, S_2) \geq 0$ **then** break;
else $\mathcal{S} := \big(\mathcal{S} \setminus \{S_1, S_2\}\big) \cup \{S_1 \cup S_2\}$.
end
output: $\mathcal{S}$

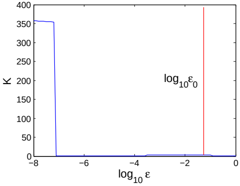
Surprisingly Good Performance

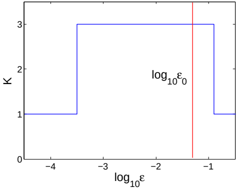
Empirically, **find global optimum and extremely robust to outliers**





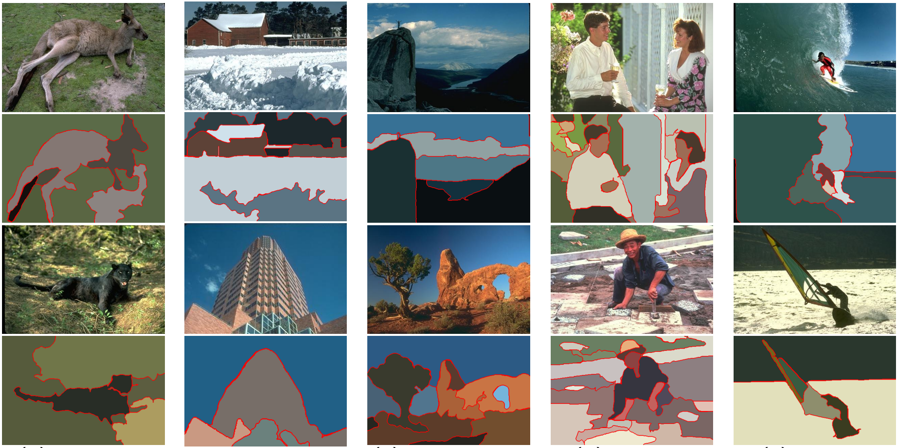
A strikingly sharp **phase transition** w.r.t. quantization ϵ





Natural Image Segmentation [Mobahi et.al., IJCV'09]

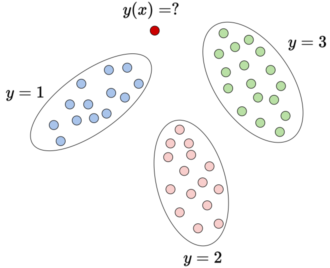
Compression alone, without any supervision, leads to **state of the art** segmentation on natural images (and many other types of data).



(a) Animals (b) Buildings (c) Landscape (d) People (e) Water

2. Classify Mixed Data (Extrapolation)

Assume data $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m]$ are i.i.d. samples from a mixture of distributions: $p(\boldsymbol{x}, \theta) = \sum_{j=1}^{k} \pi_j p_j(\boldsymbol{x}, \theta)$.



Classic approach to classify the data is via maximum a posteriori (MAP) classifier:

$$\hat{y}(\boldsymbol{x}) \;=\; \arg\max_j \log p_j(\boldsymbol{x}, \theta) + \log \pi_j.$$

Difficulties: distributions p_j are hard to estimate and log likelihood is not well-defined when distributions are degenerate.

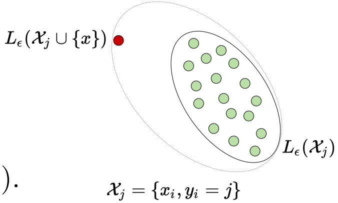
(**probably why SVMs or deep networks prevail instead...**)

Classify to Compress

[Wright, Tao, Lin, Shum, and Ma, NIPS'07]

A Fundamental Idea:
Count additional #bits needed to encode a query sample $\boldsymbol{x}$ with data in each class $\boldsymbol{X}_j$:

$$\delta L_\epsilon(\boldsymbol{x}, j) \doteq L_\epsilon(\boldsymbol{X}_j \cup \{\boldsymbol{x}\}) - L_\epsilon(\boldsymbol{X}_j) + L(j).$$



Classification Criterion: Minimum **I**ncremental **C**oding **L**ength (**MICL**):

$$\hat{y}(\boldsymbol{x}) \; = \; \arg\min_j \, \delta L_\epsilon(\boldsymbol{x}, j).$$

Law of Parsimony: *"Entities should not be multiplied without necessity."*
–William of Ockham

Asymptotic Property of MICL

Theorem (Wright, NIPS'07)

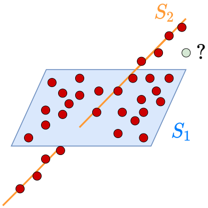
As the number of samples m goes to infinity, the MICL criterion converges at a rate of $O(m^{-1/2})$ to the following criterion:

$$\hat{y}_\epsilon(\boldsymbol{x}) = \arg\max_j \underbrace{L_G\Big(\boldsymbol{x} \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j + \frac{\epsilon^2}{D}\boldsymbol{I}\Big) + \log \pi_j}_{\textit{Regularized MAP}} + \frac{1}{2} D_\epsilon(\boldsymbol{\Sigma}_j),$$

where $D_\epsilon(\boldsymbol{\Sigma}_j) \doteq tr\Big(\boldsymbol{\Sigma}_j\big(\boldsymbol{\Sigma}_j + \frac{\epsilon^2}{D}\boldsymbol{I}\big)^{-1}\Big)$ is the effective dimension.

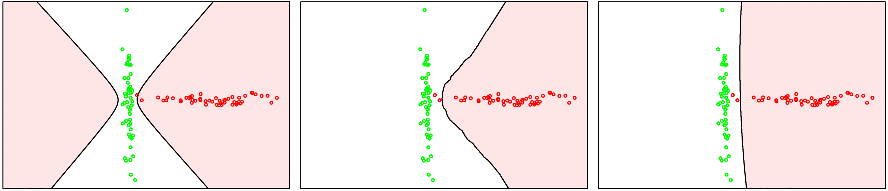
Everything else equal, MICL prefers a class with higher effective dimension.

Err on the side of caution!



Extrapolation of Low-Dim Structure for Classification

Figure: A truly extrapolating (nearest subspace) classifier!



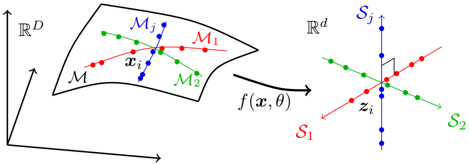
(a) **MICL** (b) k-Nearest Neighbors (c) SVM-RBF

Difficulty in practice: inference computationally costly (non-parametric) and possibly need a kernel (nonlinearity).

Go beyond (non-parametric) data interpolation and extrapolation?

Represent Multi-class Multi-dimensional Data

Given samples
$\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m] \subset \cup_{j=1}^{k} \mathcal{M}_j$,
seek a good representation
$\boldsymbol{Z} = [\boldsymbol{z}_1, \ldots, \boldsymbol{z}_m] \subset \mathbb{R}^d$
through a continuous mapping:
$f(\boldsymbol{x}, \theta) : \boldsymbol{x} \in \mathbb{R}^D \mapsto \boldsymbol{z} \in \mathbb{R}^d$.



Goals of **"re-present"** the data:

- linearization: from nonlinear structures $\cup_{j=1}^{k} \mathcal{M}_j$ to linear $\cup_{j=1}^{k} \mathcal{S}_j$.
- compression: from high-dimensional samples to compact features.
- sparsity: from separable components $\mathcal{M}_j$'s to incoherent $\mathcal{S}_j$'s.
- self-consistent: from compact structured $\boldsymbol{Z}$ back to the data $\boldsymbol{X}$.

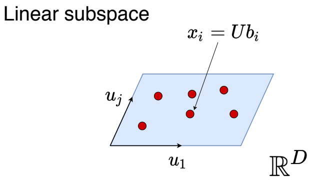
Seeking a Linear Discriminative Representation (LDR)

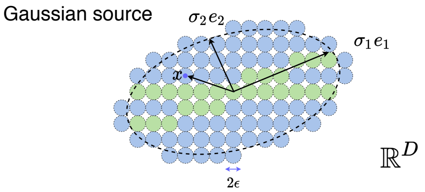
Desiderata: Representation $\boldsymbol{z} = f(\boldsymbol{x}, \theta)$ have the following properties:

1. *Within-Class Compressible:* Features of the same class/cluster should be highly compressed in a **low-dimensional** linear subspace.
2. *Between-Class Discriminative:* Features of different classes/clusters should be in highly **incoherent** linear subspaces.
3. *Maximally Informative:* Dimension (or variance) of features for each class/cluster should be **the same as that of the data**.

Is there a principled objective for all such properties, together?

Compactness Measure for Linear/Gaussian Representation





Theorem (Coding Length, Ma & Derksen TPAMI'07)

The number of bits needed to encode data $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m] \in \mathbb{R}^{D \times m}$ *up to a precision* $\|\boldsymbol{x} - \hat{\boldsymbol{x}}\|_2 \leq \epsilon$ *is bounded by:*

$$L(\boldsymbol{X}, \epsilon) \doteq \left(\frac{m+D}{2}\right) \log\det\left(\boldsymbol{I} + \frac{D}{m\epsilon^2}\boldsymbol{X}\boldsymbol{X}^\top\right).$$

This can be derived from constructively quantifying SVD of $\boldsymbol{X}$ or by sphere packing vol$(\boldsymbol{X})$ as samples of a noisy Gaussian source.

Compactness Measure for Linear/Gaussian Representation

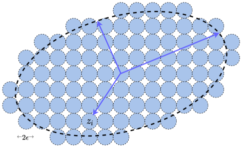
If $\boldsymbol{X}$ is not (piecewise) linear or Gaussian, consider a nonlinear mapping:

$$\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_m] \in \mathbb{R}^{D\times m} \xrightarrow{f(\boldsymbol{x},\theta)} \boldsymbol{Z}(\theta) = [\boldsymbol{z}_1, \boldsymbol{z}_2, \dots, \boldsymbol{z}_m] \in \mathbb{R}^{d\times m}.$$

The average coding length per sample (rate) subject to a distortion ϵ:

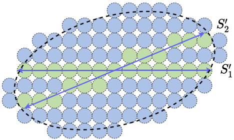
$$R(\boldsymbol{Z},\epsilon) \doteq \frac{1}{2}\log\det\left(\boldsymbol{I} + \frac{d}{m\epsilon^2}\boldsymbol{Z}\boldsymbol{Z}^\top\right). \tag{5}$$

Rate distortion is an intrinsic measure for the volume of all features.



vol(Z)

Compactness Measure for Mixed Linear Representations



The features $\boldsymbol{Z}$ of multi-class data

$$\boldsymbol{X} = \boldsymbol{X}_1 \cup \boldsymbol{X}_2 \cup \cdots \cup \boldsymbol{X}_k \ \subset \cup_{j=1}^{k} \mathcal{M}_j.$$

may be partitioned into multiple subsets:

$$\boldsymbol{Z} = \boldsymbol{Z}_1 \cup \boldsymbol{Z}_2 \cup \cdots \cup \boldsymbol{Z}_k \ \subset \cup_{j=1}^{k} \mathcal{S}_j.$$

W.r.t. this partition, the **average coding rate** is:

$$R^c(\boldsymbol{Z}, \epsilon \mid \boldsymbol{\Pi}) \doteq \sum_{j=1}^{k} \frac{\mathrm{tr}(\boldsymbol{\Pi}_j)}{2m} \log\det\left(\boldsymbol{I} + \frac{d}{\mathrm{tr}(\boldsymbol{\Pi}_j)\epsilon^2} \boldsymbol{Z}\boldsymbol{\Pi}_j\boldsymbol{Z}^\top\right), \qquad (6)$$

where $\boldsymbol{\Pi} = \{\boldsymbol{\Pi}_j \in \mathbb{R}^{m\times m}\}_{j=1}^{k}$ encode the membership of the m samples in the k classes: the diagonal entry $\boldsymbol{\Pi}_j(i,i)$ of $\boldsymbol{\Pi}_j$ is the probability of sample i belonging to subset j. $\Omega \doteq \{\boldsymbol{\Pi} \mid \sum \boldsymbol{\Pi}_j = \boldsymbol{I}, \boldsymbol{\Pi}_j \geq \boldsymbol{0}.\}$

Measure for Linear Discriminative Representation (LDR)

A fundamental idea: maximize the **difference** between the coding rate of <u>all features</u> and the average rate of <u>features in each of the classes</u>:

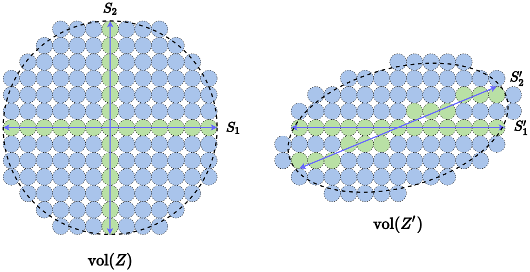
$$\Delta R(\boldsymbol{Z}, \boldsymbol{\Pi}, \epsilon) = \underbrace{\frac{1}{2} \log\det\left(\boldsymbol{I} + \frac{d}{m\epsilon^2}\boldsymbol{Z}\boldsymbol{Z}^\top\right)}_{R} - \underbrace{\sum_{j=1}^{k} \frac{\mathrm{tr}(\boldsymbol{\Pi}_j)}{2m} \log\det\left(\boldsymbol{I} + \frac{d}{\mathrm{tr}(\boldsymbol{\Pi}_j)\epsilon^2}\boldsymbol{Z}\boldsymbol{\Pi}_j\boldsymbol{Z}^\top\right)}_{R^c}.$$

This difference is called **rate reduction**:

- Large R: expand all features $\boldsymbol{Z}$ as **large** as possible.
- Small R^c: compress each class $\boldsymbol{Z}_j$ as **small** as possible.

Slogan: similarity contracts and dissimilarity contrasts!

Interpretation of MCR^2: Sphere Packing and Counting



Example: two subspaces S_1 and S_2 in $\mathbb{R}^2$.

- $\log \#$(green spheres + blue spheres) = rate of span of all samples R.
- $\log \#$(green spheres) = rate of the two subspaces R^c.
- $\log \#$(blue spheres) = rate reduction ΔR.

Comparison to Contrastive Learning

[Hadsell, Chopra, and LeCun, CVPR'06]

When k is large, a randomly chosen **pair** $(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is of high probability belonging to different classes. Minimize the **contrastive loss**:

$$\min -\log \frac{\exp(\langle \boldsymbol{z}_i, \boldsymbol{z}_i' \rangle)}{\sum_{j \neq i} \exp(\langle \boldsymbol{z}_i, \boldsymbol{z}_j \rangle)}.$$

The learned features of such pairs of samples together with their augmentations $\boldsymbol{Z}_i$ and $\boldsymbol{Z}_j$ should have large rate reduction:

$$\max \sum_{ij} \Delta R_{ij} \doteq R\big(\boldsymbol{Z}_i \cup \boldsymbol{Z}_j, \epsilon\big) - \frac{1}{2}\big(R(\boldsymbol{Z}_i, \epsilon) + R(\boldsymbol{Z}_j, \epsilon)\big).$$

MCR^2 contrasts triplets, quadruplets, or any number of sets.

Principle of Maximal Coding Rate Reduction (MCR^2)

[Yu, Chan, You, Song, Ma, NeurIPS2020]

Learn a mapping $f(\boldsymbol{x},\theta)$ (for a given partition $\boldsymbol{\Pi}$):

$$\boldsymbol{X} \xrightarrow{f(\boldsymbol{x},\theta)} \boldsymbol{Z}(\theta) \xrightarrow{\boldsymbol{\Pi},\epsilon} \Delta R(\boldsymbol{Z}(\theta),\boldsymbol{\Pi},\epsilon) \tag{7}$$

so as to **M**aximize the **C**oding **R**ate **R**eduction (**MCR**^{2}):

$$\begin{aligned} \max_{\theta} \quad & \Delta R\big(\boldsymbol{Z}(\theta),\boldsymbol{\Pi},\epsilon\big) = R(\boldsymbol{Z}(\theta),\epsilon) - R^c(\boldsymbol{Z}(\theta),\epsilon \mid \boldsymbol{\Pi}), \\ & \text{subject to} \quad \|\boldsymbol{Z}_j(\theta)\|_F^2 = m_j,\ \boldsymbol{\Pi} \in \Omega. \end{aligned} \tag{8}$$

Since ΔR is *monotonic* in the scale of $\boldsymbol{Z}$, one needs to:
normalize the features $z = f(x,\theta)$ so as to compare $Z(\theta)$ and $Z(\theta')$!

Batch normalization, Sergey Ioffe and Christian Szegedy, 2015.

Layer normalization'16, instance normalization'16; group normalization'18...

Theoretical Justification of the MCR^2 Principle

Theorem (Informal Statement [Yu et.al., NeurIPS2020])

Suppose $\boldsymbol{Z}^\star = \boldsymbol{Z}_1^\star \cup \cdots \cup \boldsymbol{Z}_k^\star$ *is the optimal solution that maximizes the rate reduction* (8)*. We have:*

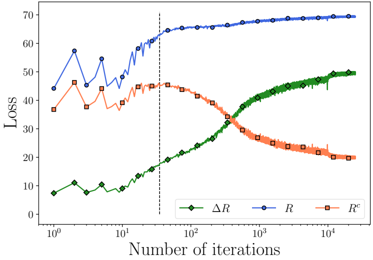
- Between-class Discriminative*: As long as the ambient space is adequately large* ($d \geq \sum_{j=1}^k d_j$)*, the subspaces are all orthogonal to each other,* i.e. $(\boldsymbol{Z}_i^\star)^\top \boldsymbol{Z}_j^\star = \mathbf{0}$ *for* $i \neq j$*.*
- Maximally Informative Representation*: As long as the coding precision is adequately high, i.e.,* $\epsilon^4 < \min_j \left\{ \frac{m_j}{m} \frac{d^2}{d_j^2} \right\}$*, each subspace achieves its maximal dimension, i.e.* $\text{rank}(\boldsymbol{Z}_j^\star) = d_j$*. In addition, the largest* $d_j - 1$ *singular values of* $\boldsymbol{Z}_j^\star$ *are equal.*

A new slogan, beyond Aristotle:

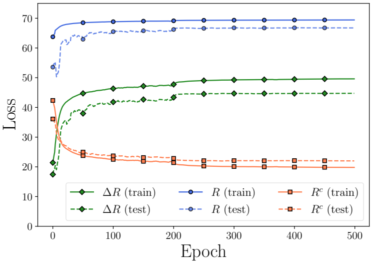
The whole is to be maximally greater than the sum of the parts!

Experiment I: Supervised Deep Learning

Experimental Setup: Train $f(\boldsymbol{x}, \theta)$ as ResNet18 on the CIFAR10 dataset, feature $\boldsymbol{z}$ dimension $d = 128$, precision $\epsilon^2 = 0.5$.



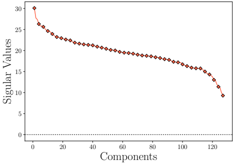
(a)



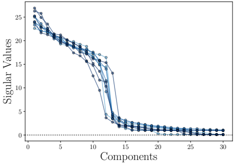
(b)

Figure: (a). Evolution of $R, R^c, \Delta R$ during the training process; (b). Training loss versus testing loss.

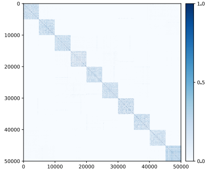
Visualization of Learned Representations Z



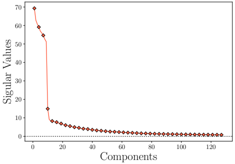
(a) MCR^2 (overall)



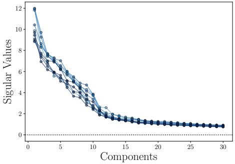
(b) MCR^2 (PCA of every class)



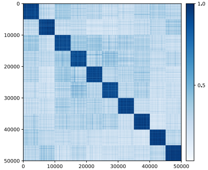
(c) MCR^2 (cosine similarity)



(d) CE (overall)



(e) CE (PCA of every class)



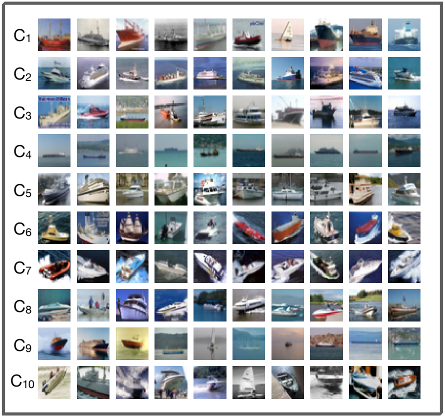
(f) CE (cosine similarity)

Figure: PCA of learned representations from MCR^2 and cross-entropy.

No neural collapse!

Visualization - Samples along Principal Components

(a) Bird



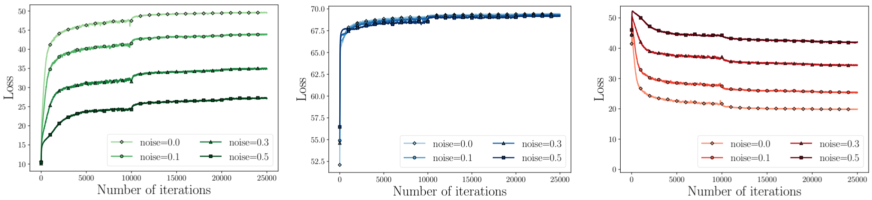
(b) Ship

Figure: Top-10 "principal" images for class - "Bird" and "Ship" in the CIFAR10.

Experiment II: Robustness to Label Noise

	Ratio=0.0	Ratio=0.1	Ratio=0.2	Ratio=0.3	Ratio=0.4	Ratio=0.5
CE Training	0.939	0.909	0.861	0.791	0.724	0.603
MCR^2 Training	**0.940**	**0.911**	**0.897**	**0.881**	**0.866**	**0.843**

Table 1: Classification results with features learned with labels corrupted at different levels.



(a) $\Delta R\big(\boldsymbol{Z}(\theta), \boldsymbol{\Pi}, \epsilon\big)$ (b) $R(\boldsymbol{Z}(\theta), \epsilon)$ (c) $R^c(\boldsymbol{Z}(\theta), \epsilon \mid \boldsymbol{\Pi})$

Figure: Evolution of $R, R^c, \Delta R$ of MCR^2 during training with corrupted labels.

Represent only what can be jointly compressed.

Deep Networks from Optimizing Rate Reduction

$$\boldsymbol{X} \xrightarrow{f(\boldsymbol{x},\theta)} \boldsymbol{Z}(\theta); \quad \max_{\theta} \Delta R(\boldsymbol{Z}(\theta), \boldsymbol{\Pi}, \epsilon).$$

Final features learned by MCR^2 are more interpretable and robust, **but**:

- The borrowed deep network (e.g. ResNet) is still a "black box"!
- Why is a "deep" architecture necessary, and how wide and deep?
- What are the roles of the "linear and nonlinear" operators?
- Why "multi-channel" convolutions?
- ...

Replace black box networks with entirely "white box" networks?

Projected Gradient Ascent for Rate Reduction

Recall the rate reduction objective:

$$\max_{\boldsymbol{Z}} \Delta R(\boldsymbol{Z}) \doteq \underbrace{\frac{1}{2}\log\det\left(\boldsymbol{I} + \alpha \boldsymbol{Z}\boldsymbol{Z}^*\right)}_{R(\boldsymbol{Z})} - \underbrace{\sum_{j=1}^{k}\frac{\gamma_j}{2}\log\det\left(\boldsymbol{I} + \alpha_j \boldsymbol{Z}\boldsymbol{\Pi}^j\boldsymbol{Z}^*\right)}_{R_c(\boldsymbol{Z},\boldsymbol{\Pi})}, \quad (9)$$

where $\alpha = d/(m\epsilon^2)$, $\alpha_j = d/(\mathrm{tr}(\boldsymbol{\Pi}^j)\epsilon^2)$, $\gamma_j = \mathrm{tr}(\boldsymbol{\Pi}^j)/m$ for $j = 1, \ldots, k$.

Consider directly maximizing ΔR with **projected gradient ascent** (PGA):

$$\boldsymbol{Z}_{\ell+1} \ \propto \ \boldsymbol{Z}_\ell + \eta \cdot \left.\frac{\partial \Delta R}{\partial \boldsymbol{Z}}\right|_{\boldsymbol{Z}_\ell} \quad \text{subject to} \quad \boldsymbol{Z}_{\ell+1} \subset \mathbb{S}^{d-1}. \qquad (10)$$

Gradients of the Two Terms

The derivatives $\frac{\partial R(\boldsymbol{Z})}{\partial \boldsymbol{Z}}$ and $\frac{\partial R_c(\boldsymbol{Z},\boldsymbol{\Pi})}{\partial \boldsymbol{Z}}$ are:

$$\frac{1}{2}\frac{\partial \log\det(\boldsymbol{I} + \alpha \boldsymbol{Z}\boldsymbol{Z}^*)}{\partial \boldsymbol{Z}}\bigg|_{\boldsymbol{Z}_\ell} = \underbrace{\alpha(\boldsymbol{I} + \alpha \boldsymbol{Z}_\ell \boldsymbol{Z}_\ell^*)^{-1}}_{\boldsymbol{E}_\ell \,\in \mathbb{R}^{d\times d}} \boldsymbol{Z}_\ell, \tag{11}$$

$$\frac{1}{2}\frac{\partial \left(\gamma_j \log\det(\boldsymbol{I} + \alpha_j \boldsymbol{Z}\boldsymbol{\Pi}^j \boldsymbol{Z}^*)\right)}{\partial \boldsymbol{Z}}\bigg|_{\boldsymbol{Z}_\ell} = \gamma_j \underbrace{\alpha_j(\boldsymbol{I} + \alpha_j \boldsymbol{Z}_\ell \boldsymbol{\Pi}^j \boldsymbol{Z}_\ell^*)^{-1}}_{\boldsymbol{C}_\ell^j \,\in \mathbb{R}^{d\times d}} \boldsymbol{Z}_\ell \boldsymbol{\Pi}^j. \tag{12}$$

Hence the gradient $\frac{\partial \Delta R(\boldsymbol{Z})}{\partial \boldsymbol{Z}}$ is:

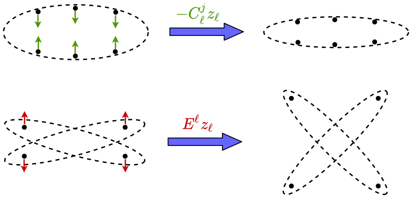
$$\frac{\partial \Delta R}{\partial \boldsymbol{Z}}\bigg|_{\boldsymbol{Z}_\ell} = \underbrace{\boldsymbol{E}_\ell}_{\text{Expansion}} \boldsymbol{Z}_\ell - \sum_{j=1}^{k} \gamma_j \underbrace{\boldsymbol{C}_\ell^j}_{\text{Compression}} \boldsymbol{Z}_\ell \boldsymbol{\Pi}^j \quad \in \mathbb{R}^{d\times m}. \tag{13}$$

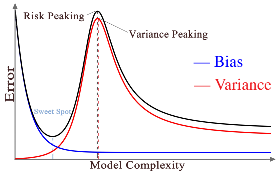
Interpretation of the Linear Operators $\boldsymbol{E}$ and $\boldsymbol{C}^j$

For any $\boldsymbol{z}_\ell \in \mathbb{R}^d$, we have

$$\boldsymbol{E}_\ell \boldsymbol{z}_\ell = \alpha(\boldsymbol{z}_\ell - \boldsymbol{Z}_\ell \boldsymbol{q}_\ell^*) \quad \text{with} \quad \boldsymbol{q}_\ell^* \doteq \arg\min_{\boldsymbol{q}_\ell} \alpha \|\boldsymbol{z}_\ell - \boldsymbol{Z}_\ell \boldsymbol{q}_\ell\|_2^2 + \|\boldsymbol{q}_\ell\|_2^2.$$

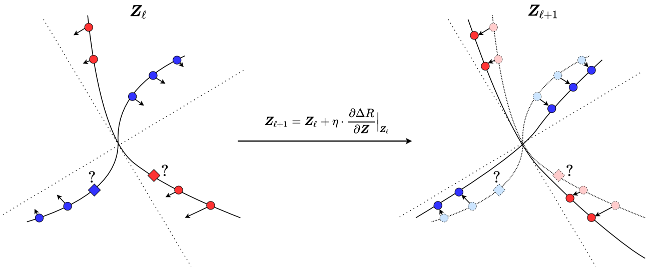
$\boldsymbol{E}_\ell \boldsymbol{z}_\ell$ and $\boldsymbol{C}_\ell^j \boldsymbol{z}_\ell$ are the "residuals" of $\boldsymbol{z}_\ell$ against the subspaces spanned by columns of $\boldsymbol{Z}_\ell$ and $\boldsymbol{Z}_\ell^j$, respectively.





Such "auto" ridge regressions **do not overfit** even with redundant random regressors, due to a "double descent" risk [Yang, ICML'20]!

Incremental Deformation via Gradient Flow



Extrapolate the gradient $\frac{\partial \Delta R(\boldsymbol{Z})}{\partial \boldsymbol{Z}}$ from training samples $\boldsymbol{Z}$ to all $\boldsymbol{z} \in \mathbb{R}^d$:

$$\frac{\partial \Delta R}{\partial \boldsymbol{Z}}\bigg|_{\boldsymbol{Z}_\ell} = \boldsymbol{E}_\ell \boldsymbol{Z}_\ell - \sum_{j=1}^{k} \gamma_j \boldsymbol{C}_\ell^j \boldsymbol{Z}_\ell \underbrace{\boldsymbol{\Pi}^j}_{\text{known}} \in \mathbb{R}^{d \times m}, \tag{14}$$

$$g(\boldsymbol{z}_\ell, \boldsymbol{\theta}_\ell) \doteq \boldsymbol{E}_\ell \boldsymbol{z}_\ell - \sum_{j=1}^{k} \gamma_j \boldsymbol{C}_\ell^j \boldsymbol{z}_\ell \underbrace{\boldsymbol{\pi}^j(\boldsymbol{z}_\ell)}_{\text{unknown}} \in \mathbb{R}^{d}. \tag{15}$$

Estimate of the Membership $\boldsymbol{\pi}^j(\boldsymbol{z}_\ell)$

Estimate the membership $\boldsymbol{\pi}^j(\boldsymbol{z}_\ell)$ with "softmax" on the residuals $\|\boldsymbol{C}_\ell^j \boldsymbol{z}_\ell\|$:

$$\boldsymbol{\pi}^j(\boldsymbol{z}_\ell) \approx \widehat{\boldsymbol{\pi}}^j(\boldsymbol{z}_\ell) \doteq \frac{\exp\left(-\lambda\|\boldsymbol{C}_\ell^j \boldsymbol{z}_\ell\|\right)}{\sum_{j=1}^k \exp\left(-\lambda\|\boldsymbol{C}_\ell^j \boldsymbol{z}_\ell\|\right)} \quad \in [0,1]. \tag{16}$$

Thus the weighted residuals for contracting:

$$\boldsymbol{\sigma}\Big([\boldsymbol{C}_\ell^1 \boldsymbol{z}_\ell, \ldots, \boldsymbol{C}_\ell^k \boldsymbol{z}_\ell]\Big) \;\doteq\; \sum_{j=1}^k \gamma_j \boldsymbol{C}_\ell^j \boldsymbol{z}_\ell \cdot \widehat{\boldsymbol{\pi}}^j(\boldsymbol{z}_\ell) \quad \in \mathbb{R}^d. \tag{17}$$

Many alternatives, e.g. enforcing all features to be in the first quadrant:

$$\boldsymbol{\sigma}(\boldsymbol{z}_\ell) \;\approx\; \boldsymbol{z}_\ell - \sum_{j=1}^k \mathsf{ReLU}\big(\boldsymbol{P}_\ell^j \boldsymbol{z}_\ell\big), \tag{18}$$

The ReduNet for Optimizing Rate **Reduction**

Iterative projected gradient ascent (PGA) :

$$\boldsymbol{z}_{\ell+1} \;\propto\; \boldsymbol{z}_\ell + \eta \cdot \underbrace{\Big[\boldsymbol{E}_\ell \boldsymbol{z}_\ell + \boldsymbol{\sigma}\big([\boldsymbol{C}_\ell^1 \boldsymbol{z}_\ell, \ldots, \boldsymbol{C}_\ell^k \boldsymbol{z}_\ell]\big)\Big]}_{g(\boldsymbol{z}_\ell, \boldsymbol{\theta}_\ell)} \quad \text{s.t.} \quad \boldsymbol{z}_{\ell+1} \in \mathbb{S}^{d-1}, \qquad (19)$$

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = \phi^L \circ \phi^{L-1} \circ \cdots \circ \phi^0(\boldsymbol{x}), \text{ with } \phi^\ell(\boldsymbol{z}_\ell, \boldsymbol{\theta}_\ell) \doteq \mathcal{P}_{\mathbb{S}^{d-1}}[\boldsymbol{z}_\ell + \eta \cdot g(\boldsymbol{z}_\ell, \boldsymbol{\theta}_\ell)].$$

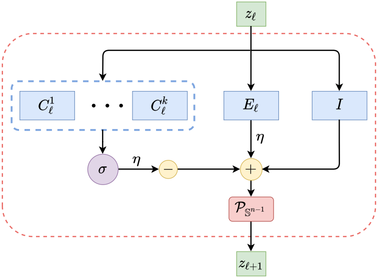


Figure: One layer of the **ReduNet**: one PGA iteration.

The ReduNet versus ResNet or ResNeXt

Iterative projected gradient ascent (PGA):

$$\boldsymbol{z}_{\ell+1} \propto \boldsymbol{z}_\ell + \eta \cdot \underbrace{\Big[\boldsymbol{E}_\ell \boldsymbol{z}_\ell + \boldsymbol{\sigma}\big([\boldsymbol{C}_\ell^1 \boldsymbol{z}_\ell, \dots, \boldsymbol{C}_\ell^k \boldsymbol{z}_\ell]\big)\Big]}_{g(\boldsymbol{z}_\ell, \boldsymbol{\theta}_\ell)} \quad \text{s.t.} \quad \boldsymbol{z}_{\ell+1} \in \mathbb{S}^{d-1}, \qquad (20)$$

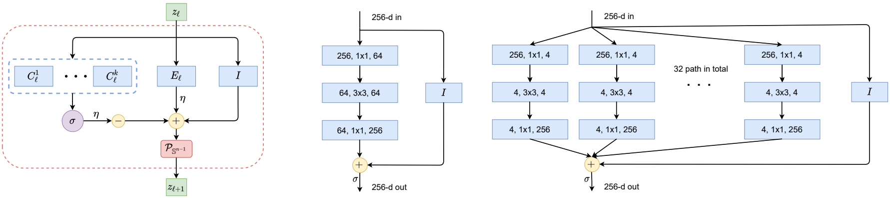


Figure: Left: **ReduNet**. Middle and Right: **ResNet** [He et. al. 2016] and **ResNeXt** [Xie et. al. 2017] (hundreds of layers).

Forward construction instead of back propagation!

The ReduNet versus Mixture of Experts

Approximate iterative projected gradient ascent (PGA) :

$$\boldsymbol{z}_{\ell+1} \;\propto\; \boldsymbol{z}_\ell + \eta \cdot \underbrace{\left[\boldsymbol{E}_\ell \boldsymbol{z}_\ell + \boldsymbol{\sigma}\left([\boldsymbol{C}_\ell^1 \boldsymbol{z}_\ell, \ldots, \boldsymbol{C}_\ell^k \boldsymbol{z}_\ell]\right)\right]}_{g(\boldsymbol{z}_\ell, \boldsymbol{\theta}_\ell)} \quad \text{s.t.} \quad \boldsymbol{z}_{\ell+1} \in \mathbb{S}^{d-1}, \qquad (21)$$

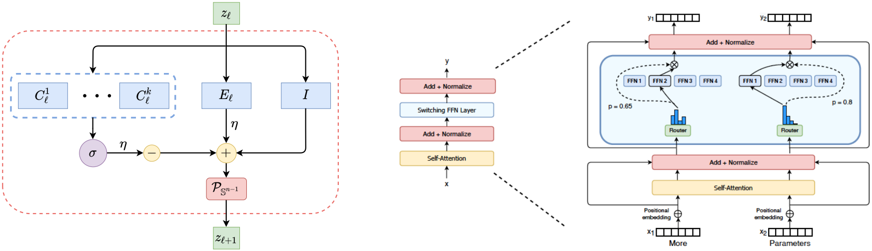


Figure: Left: **ReduNet** layer. Right: **Mixture of Experts** [Shazeer et. al. 2017] or **Switched Transformer** [Fedus et. al. 2021] (1.7 trillion parameters).

Forward construction instead of back propagation!

ReduNet Features for Mixture of Gaussians

$L =$ **2000**-Layers ReduNet: $m = 500, \eta = 0.5, \epsilon = 0.1.$

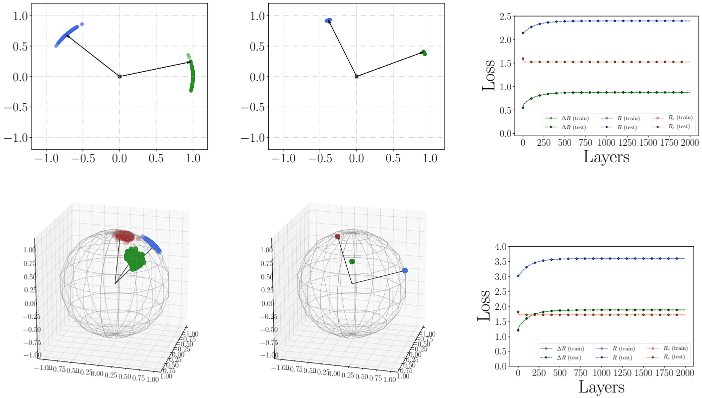


Figure: Left: original samples $\boldsymbol{X}$ and ReduNet features $\boldsymbol{Z} = f(\boldsymbol{Z}, \boldsymbol{\theta})$ for 2D and 3D Mixture of Gaussians. Right: plots for the progression of values of the rates.

Group Invariant Classification

Feature mapping $f(\boldsymbol{x}, \boldsymbol{\theta})$ is invariant to a group of transformations:

$$\textit{Group Invariance:} \quad f(\boldsymbol{x} \circ \mathfrak{g}, \boldsymbol{\theta}) \sim f(\boldsymbol{x}, \boldsymbol{\theta}), \quad \forall \mathfrak{g} \in \mathbb{G}, \tag{22}$$

where "$\sim$" indicates two features belonging to the same equivalent class.

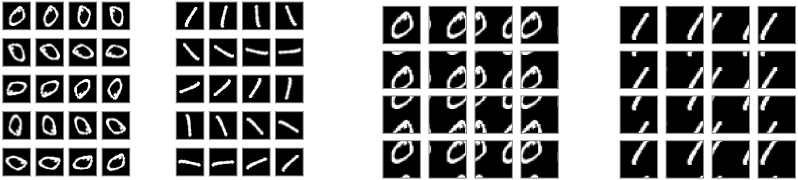


Figure: Left: 1D rotation $\mathbb{S}^1$; Right: 2D cyclic translation $\mathcal{T}^2$.

1. Fooling CNNs with simple transformations, Engstrom et.al., 2017.

2. Why do deep convolutional networks generalize so poorly to small image transformations? Azulay & Weiss, 2018.

Group Invariant Classification

Feature mapping $f(\boldsymbol{x}, \boldsymbol{\theta})$ is invariant to a group of transformations:

$$\textit{Group Invariance:} \quad f(\boldsymbol{x} \circ \mathfrak{g}, \boldsymbol{\theta}) \sim f(\boldsymbol{x}, \boldsymbol{\theta}), \quad \forall \mathfrak{g} \in \mathbb{G}, \tag{23}$$

where "$\sim$" indicates two features belonging to the same equivalent class.

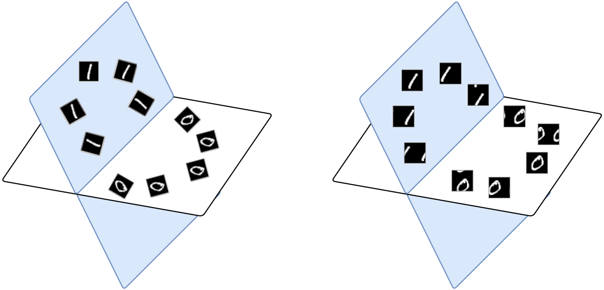


Figure: **Embed all equivariant samples to the same subspace.**

Circulant Matrix and Convolution

Given a vector $\boldsymbol{z} = [z_0, z_1, \ldots, z_{n-1}]^* \in \mathbb{R}^n$, we may arrange all its circular shifted versions in a circulant matrix form as

$$\operatorname{circ}(\boldsymbol{z}) \quad \doteq \quad \begin{bmatrix} z_0 & z_{n-1} & \cdots & z_2 & z_1 \\ z_1 & z_0 & z_{n-1} & \cdots & z_2 \\ \vdots & z_1 & z_0 & \ddots & \vdots \\ z_{n-2} & \vdots & \ddots & \ddots & z_{n-1} \\ z_{n-1} & z_{n-2} & \cdots & z_1 & z_0 \end{bmatrix} \quad \in \mathbb{R}^{n\times n}. \tag{24}$$

A circular (or cyclic) convolution:

$$\operatorname{circ}(\boldsymbol{z}) \cdot \boldsymbol{x} = \boldsymbol{z} \circledast \boldsymbol{x}, \quad \text{where} \quad (\boldsymbol{z} \circledast \boldsymbol{x})_i = \sum_{j=0}^{n-1} x_j z_{i+n-j \bmod n}. \tag{25}$$

Convolutions from Cyclic Shift Invariance

Given a set of sample vectors $\boldsymbol{Z} = [\boldsymbol{z}^1, \ldots, \boldsymbol{z}^m]$, construct the ReduNet from cyclic-shift augmented families $\boldsymbol{Z} = [\text{circ}(\boldsymbol{z}^1), \ldots, \text{circ}(\boldsymbol{z}^m)]$.

Proposition (Convolution Structures of $\boldsymbol{E}$ and $\boldsymbol{C}^j$)

The linear operator in the ReduNet:

$$\boldsymbol{E} = \alpha\Big(\boldsymbol{I} + \alpha \sum_{i=1}^{m} \text{circ}(\boldsymbol{z}^i)\text{circ}(\boldsymbol{z}^i)^*\Big)^{-1}$$

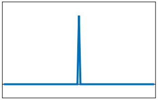
is a circulant matrix and represents a circular convolution:

$$\boldsymbol{E}\boldsymbol{z} = \boldsymbol{e} \circledast \boldsymbol{z},$$

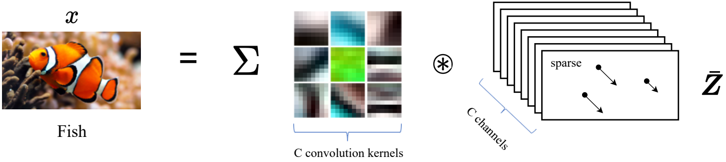
where $\boldsymbol{e}$ is the first column vector of $\boldsymbol{E}$. Similarly, the operators $\boldsymbol{C}^j$ associated with subsets $\boldsymbol{Z}^j$ are also circular convolutions.

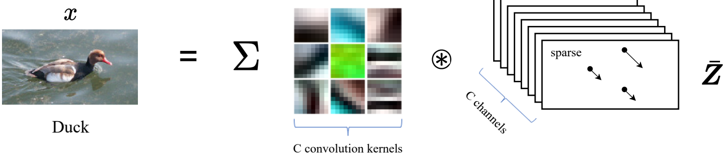
Tradeoff between Invariance and Separability

A problem with separability: superposition of shifted "delta" functions can generate any other signals: $\text{span}[\text{circ}(\boldsymbol{x})] = \mathbb{R}^n$!



A necessary assumption: $\boldsymbol{x}$ is **sparsely generated** from incoherent dictionaries for different classes: $\boldsymbol{x} = [\text{circ}(\mathcal{D}_1), \text{circ}(\mathcal{D}_2), \ldots, \text{circ}(\mathcal{D}_k)]\bar{\boldsymbol{z}}$.



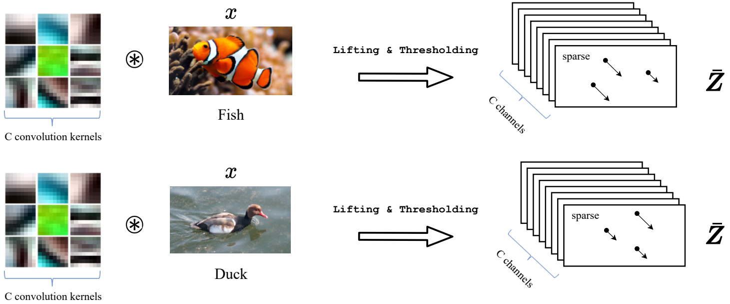


Tradeoff between Invariance and Separability

A basic idea: estimate sparse codes $\bar{\boldsymbol{z}}$ by taking their responses to multiple analysis filters $\boldsymbol{k}_1, \ldots, \boldsymbol{k}_C \in \mathbb{R}^n$ [Rubinstein & Elad 2014]:

$$\bar{\boldsymbol{z}} = \boldsymbol{\tau}\left[\boldsymbol{k}_1 \circledast \boldsymbol{x}, \ldots, \boldsymbol{k}_C \circledast \boldsymbol{x}\right]^* \quad \in \mathbb{R}^{C \times n}. \tag{26}$$

for some entry-wise "sparsity-promoting" operator $\boldsymbol{\tau}(\cdot)$.



Multi-Channel Convolutions

Given a set of multi-channel sparse codes $\bar{\boldsymbol{Z}} = [\bar{\boldsymbol{z}}^1, \ldots, \bar{\boldsymbol{z}}^m]$, construct the ReduNet from their circulant families $\bar{\boldsymbol{Z}} = [\mathrm{circ}(\bar{\boldsymbol{z}}^1), \ldots, \mathrm{circ}(\bar{\boldsymbol{z}}^m)]$.

Proposition (Convolution Structures of $\bar{\boldsymbol{E}}$ and $\bar{\boldsymbol{C}}^j$)

The linear operator in the ReduNet:

$$\bar{\boldsymbol{E}} = \alpha\Big(\boldsymbol{I} + \alpha \sum_{i=1}^{m} \mathrm{circ}(\bar{\boldsymbol{z}}^i)\mathrm{circ}(\bar{\boldsymbol{z}}^i)^*\Big)^{-1} \in \mathbb{R}^{Cn \times Cn}$$

is a block circulant matrix and represents a multi-channel convolution:

$$\bar{\boldsymbol{E}}(\bar{\boldsymbol{z}}) = \bar{\boldsymbol{e}} \circledast \bar{\boldsymbol{z}} \ \in \mathbb{R}^{Cn},$$

where $\bar{e}$ is the first slice of $\bar{\boldsymbol{E}}$. Similarly, the operators $\bar{\boldsymbol{C}}^j$ associated with subsets $\bar{\boldsymbol{Z}}^j$ are also multi-channel circular convolutions.

Multi-Channel Convolutions

$$\bar{\boldsymbol{E}}(\bar{\boldsymbol{z}}) = \bar{\boldsymbol{e}} \circledast \bar{\boldsymbol{z}} \ \in \mathbb{R}^{Cn}, \quad \bar{\boldsymbol{C}}^j(\bar{\boldsymbol{z}}) = \bar{\boldsymbol{c}}^j \circledast \bar{\boldsymbol{z}} \ \in \mathbb{R}^{Cn}:$$

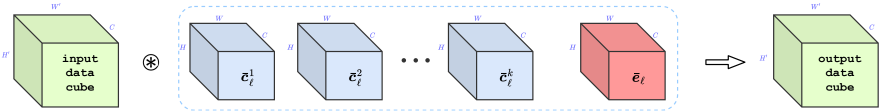


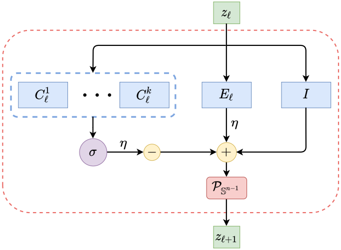
Figure: $\bar{\boldsymbol{E}}$ **and** $\bar{\boldsymbol{C}}^j$ **are automatically multi-channel convolutions!**

The Convolution ReduNet versus Scattering Network

Iterative projected gradient ascent (PGA) for invariant rate reduction:

$$\bar{\boldsymbol{z}}_{\ell+1} \ \propto \ \bar{\boldsymbol{z}}_{\ell} + \eta \cdot \underbrace{\left[\bar{\boldsymbol{E}}_{\ell}\bar{\boldsymbol{z}}_{\ell} + \boldsymbol{\sigma}\Big([\bar{\boldsymbol{C}}_{\ell}^{1}\bar{\boldsymbol{z}}_{\ell}, \ldots, \bar{\boldsymbol{C}}_{\ell}^{k}\bar{\boldsymbol{z}}_{\ell}]\Big)\right]}_{g(\bar{\boldsymbol{z}}_{\ell}, \boldsymbol{\theta}_{\ell})}, \tag{27}$$

with each layer being a fixed number of multi-channel convolutions!



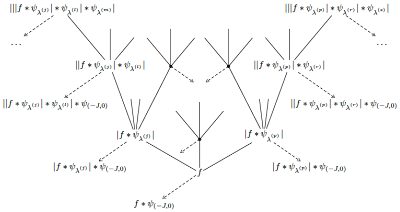


Fig. 2: Scattering network architecture based on wavelet filters and the modulus non-linearity. The elements of the feature vector $\Phi_W(f)$ in (1) are indicated at the tips of the arrows.

Figure: Left: **ReduNet** layer. Right: **Scatterting Network** [J. Bruna and S. Mallat, 2013] [T. Wiatowski and H. Blcskei, 2018] (**only 2-3 layers**).

Fast Computation in Spectral Domain

Fact: all circulant matrices can be simultaneously diagonalized by the *discrete Fourier transform* $\boldsymbol{F}$: $\text{circ}(\boldsymbol{z}) = \boldsymbol{F}^* \boldsymbol{D} \boldsymbol{F}$.

$$\left(\boldsymbol{I} + \sum_{i=1}^{m} \text{circ}(\bar{\boldsymbol{z}}^i)\text{circ}(\bar{\boldsymbol{z}}^i)^*\right)^{-1} = \left(\boldsymbol{I} + \begin{bmatrix} \boldsymbol{F}^* & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \ddots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{F}^* \end{bmatrix} \begin{bmatrix} \boldsymbol{D}_{11} & \cdots & \boldsymbol{D}_{1C} \\ \vdots & \ddots & \vdots \\ \boldsymbol{D}_{C1} & \cdots & \boldsymbol{D}_{CC} \end{bmatrix} \begin{bmatrix} \boldsymbol{F} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \ddots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{F} \end{bmatrix}\right)^{-1} \in \mathbb{R}^{nC \times nC}$$

where $\boldsymbol{D}_{ij}$ are all diagonal of size n.

Computing the inverse is $O(C^3 n)$ in the spectral domain, instead of $O(C^3 n^3)$! *Learning convolutional networks for invariant classification is* ***naturally far more efficient in the spectral domain****!*

Nature: In visual cortex, neurons encode and transmit information in frequency, hence called "spiking neurons" [Softky & Koch, 1993; Eliasmith & Anderson, 2003].

A "White Box" Deep Convolutional ReduNet by Construction (Spectral Domain)

Require: $\bar{\boldsymbol{Z}} \in \mathbb{R}^{C\times T\times m}$, $\boldsymbol{\Pi}$, $\epsilon > 0$, λ, and a learning rate η.

Set $\alpha = \frac{C}{m\epsilon^2}$, $\{\alpha_j = \frac{C}{\mathrm{tr}(\boldsymbol{\Pi}^j)\epsilon^2}\}_{j=1}^k$, $\{\gamma_j = \frac{\mathrm{tr}(\boldsymbol{\Pi}^j)}{m}\}_{j=1}^k$.

Set $\bar{\boldsymbol{V}}_0 = \{\bar{\boldsymbol{v}}_0^i(p) \in \mathbb{C}^C\}_{p=0,i=1}^{T-1,m} \doteq \mathrm{DFT}(\bar{\boldsymbol{Z}}) \in \mathbb{C}^{C\times T\times m}$.

for $\ell = 1, 2, \ldots, L$ **do**

  for $p = 0, 1, \ldots, T-1$ **do**

    Compute $\bar{\mathcal{E}}_\ell(p) \in \mathbb{C}^{C\times C}$ and $\{\bar{\mathcal{C}}_\ell^j(p) \in \mathbb{C}^{C\times C}\}_{j=1}^k$ as

    $\bar{\mathcal{E}}_\ell(p) \doteq \alpha \cdot \left[\boldsymbol{I} + \alpha \cdot \bar{\boldsymbol{V}}_{\ell-1}(p) \cdot \bar{\boldsymbol{V}}_{\ell-1}(p)^*\right]^{-1}$,

    $\bar{\mathcal{C}}_\ell^j(p) \doteq \alpha_j \cdot \left[\boldsymbol{I} + \alpha_j \cdot \bar{\boldsymbol{V}}_{\ell-1}(p) \cdot \boldsymbol{\Pi}^j \cdot \bar{\boldsymbol{V}}_{\ell-1}(p)^*\right]^{-1}$;

  end for

  for $i = 1, \ldots, m$ **do**

    for $p = 0, 1, \ldots, T-1$ **do**

      Compute $\{\bar{\boldsymbol{p}}_\ell^{ij}(p) \doteq \bar{\mathcal{C}}_\ell^j(p) \cdot \bar{\boldsymbol{v}}_\ell^i(p) \in \mathbb{C}^{C\times 1}\}_{j=1}^k$;

    end for

    Let $\{\bar{\boldsymbol{P}}_\ell^{ij} = [\bar{\boldsymbol{p}}_\ell^{ij}(0), \ldots, \bar{\boldsymbol{p}}_\ell^{ij}(T-1)] \in \mathbb{C}^{C\times T}\}_{j=1}^k$;

    Compute $\left\{\widehat{\boldsymbol{\pi}}_\ell^{ij} = \frac{\exp(-\lambda\|\bar{\boldsymbol{P}}_\ell^{ij}\|_F)}{\sum_{j=1}^k \exp(-\lambda\|\bar{\boldsymbol{P}}_\ell^{ij}\|_F)}\right\}_{j=1}^k$;

    for $p = 0, 1, \ldots, T-1$ **do**

      $\bar{\boldsymbol{v}}_\ell^i(p) = \bar{\boldsymbol{v}}_{\ell-1}^i(p) + \eta\left(\bar{\mathcal{E}}_\ell(p)\bar{\boldsymbol{v}}_\ell^i(p) - \sum_{j=1}^k \gamma_j \cdot \widehat{\boldsymbol{\pi}}_\ell^{ij} \cdot \bar{\mathcal{C}}_\ell^j(p) \cdot \bar{\boldsymbol{v}}_\ell^i(p)\right)$;

    end for

    $\bar{\boldsymbol{v}}_\ell^i = \bar{\boldsymbol{v}}_\ell^i \,/\, \|\bar{\boldsymbol{v}}_\ell^i\|_F$;

  end for

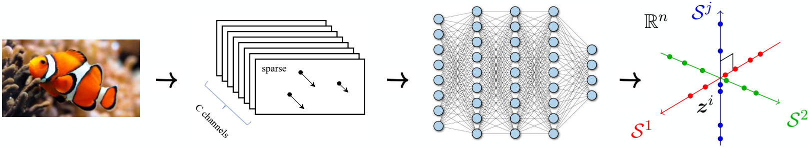
  Set $\bar{\boldsymbol{Z}}_\ell = \mathrm{IDFT}(\bar{\boldsymbol{V}}_\ell)$ as the feature at the ℓ-th layer;

  $\frac{1}{2T}\sum_{p=0}^{T-1}\left(\log\det[\boldsymbol{I} + \alpha\bar{\boldsymbol{V}}_\ell(p) \cdot \bar{\boldsymbol{V}}_\ell(p)^*] - \frac{\mathrm{tr}\left(\boldsymbol{\Pi}^j\right)}{m}\log\det[\boldsymbol{I} + \alpha_j\bar{\boldsymbol{V}}_\ell(p) \cdot \boldsymbol{\Pi}^j \cdot \bar{\boldsymbol{V}}_\ell(p)^*]\right)$;

end for

Ensure: features $\bar{\boldsymbol{Z}}_L$, the learned filters $\{\bar{\mathcal{E}}_\ell(p)\}_{\ell,p}$ and $\{\bar{\mathcal{C}}_\ell^j(p)\}_{j,\ell,p}$.

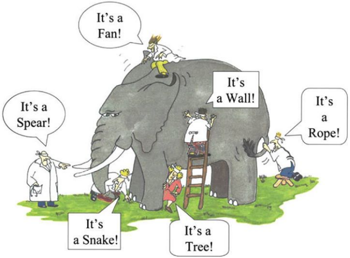
Overall Process (the Elephant)



Multi-Class Signals → **Lifting & Sparse Coding** → **Invariant Rate Reduction** → Incoherent Subspaces

Necessary components:

- **sparse coding** for class separability;
- **deep networks** maximize rate reduction;
- **spectral computing** for shift-invariance;
- convolution, normalization, nonlinearity...



Experiment: 1D Cyclic Shift Invariance of 0 and 1

2000 training samples, 1980 testing, $C = 5$, $L =$**3500**-layers ReduNet.

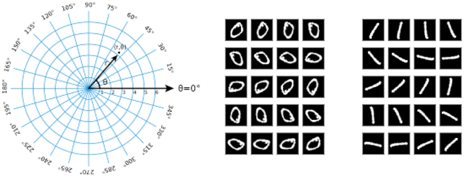


Figure: Left: Multi-channel feature representation of an image in polar coordinates. Right: Example of training/testing samples.

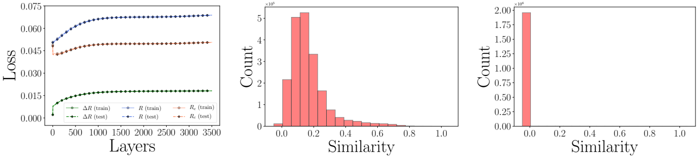


Figure: Left: Rates along the layers; Middle: cross-class cosine similarity among trainings; Right: similarity among testings.

Experiment: 1D Cyclic Shift Invariance of 0 and 1

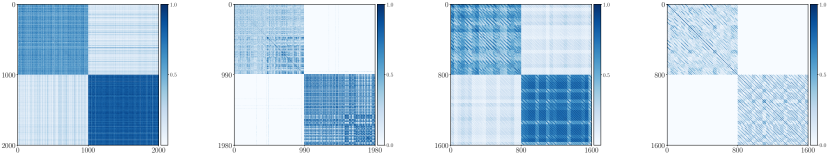


Figure: Left two: heat maps for training and testing. Right two: heat maps for one pair of samples at every possible shift.

Table: Network performance on digits with **all rotations**.

	ReduNet	ReduNet (invariant)
Acc (Original Test Data)	0.983	0.996
Acc (Test with All Shifts)	0.707	0.993

1. Fooling CNNs with simple transformations, Engstrom et.al., 2017.
2. Why do deep convolutional networks generalize so poorly to small image transformations? Azulay & Weiss, 2018.

Experiment: 1D Cyclic Shift Invariance of All 10 Digits

100 training samples, 100 testing, $C = 20$, $L =$**40**-layers ReduNet.

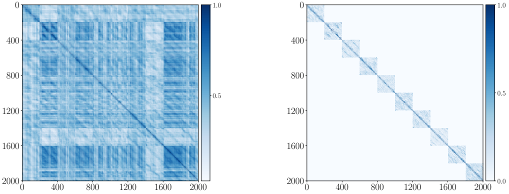


Figure: Heatmaps of cosine similarity among shifted training data $\boldsymbol{X}_{\text{shift}}$ (left) and learned features $\boldsymbol{Z}_{\text{shift}}$ (right).

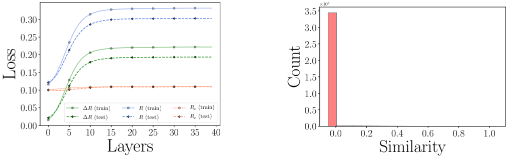
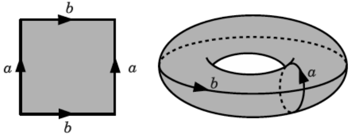


Figure: Left: Rates evolution with iterations; Right: histograms of the cosine similarity (in absolute value) between all pairs of features across different classes.

Experiment: 2D Cyclic Translation Invariance

1000 for training, 500 for testing, $C = 5$, $L =$**2000**-layers ReduNet.



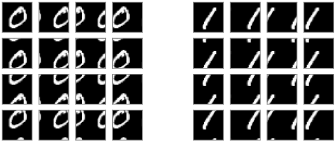


Table: Network performance on digits with **all translations**.

	ReduNet	ReduNet (invariant)
Acc (Original Test Data)	0.980	0.975
Acc (Test with All Shifts)	0.540	0.909

1. Fooling CNNs with simple transformations, Engstrom et.al., 2017.

2. Why do deep convolutional networks generalize so poorly to small image transformations? Azulay & Weiss, 2018.

Experiment: 2D Cyclic Trans. Invariance of All 10 Digits

100 training samples, 100 testing, $C = 75$, $L =$**25**-layers ReduNet.

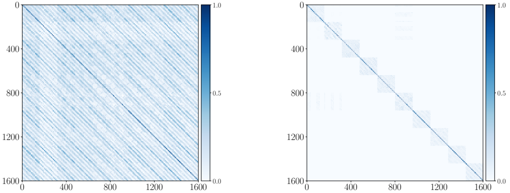


Figure: Heatmaps of cosine similarity among shifted training data $\boldsymbol{X}_{\text{shift}}$ (left) and learned features $\boldsymbol{Z}_{\text{shift}}$ (right).

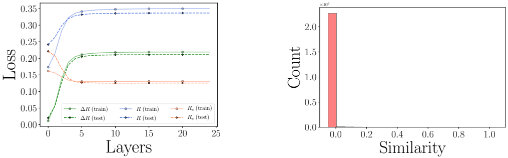


Figure: Left: Rates evolution with iterations; Right: histograms of the cosine similarity (in absolute value) between all pairs of features across different classes.

Experiment: Back Propagation of ReduNet (MNIST)

2D cyclic trans. of 10 digits, 500 training samples, all testing, $C = 16$, $L =$**30**-layers invariant ReduNet.

Initialization	Backpropagation	Test Accuracy
✓	✗	89.8%
✗	✓	93.2%
✓	✓	97.8%

Table: Test accuracy of 2D translation-invariant ReduNet, ReduNet-bp (without initialization), and ReduNet-bp (with initialization) on the MNIST dataset.

- **Backprop:** the ReduNet architecture *can* be fine-tuned by SGD and achieves better standard accuracy after back propagation;
- **Initialization:** using ReduNet for initialization can achieve better performance than the same architecture with random initialization.

Experiment: Back Propagation of ReduNet (CIFAR-10)

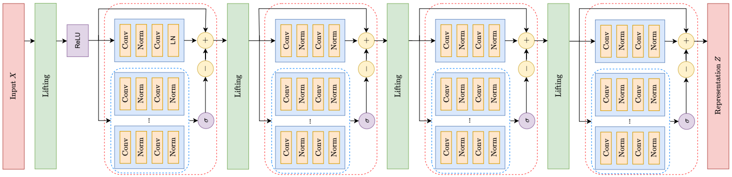


Figure: A ReduNet-inspired architecture

Table: Classification performance of ReduNet-inspired architecture on CIFAR10.

ReLU	Train Acc	Test Acc
✓	0.9997	0.8327
✗	0.9970	0.6542

Conclusions: Learn to Compress and Compress to Learn!

Principles of Parsimony:

- Clustering via compression: $\min_{\mathbf{\Pi}} R^c(\boldsymbol{X}, \mathbf{\Pi})$
- Classification via compression: $\min_{\boldsymbol{\pi}} \delta R^c(\boldsymbol{x}, \boldsymbol{\pi})$
- Representation via maximizing rate reduction: $\max_{\boldsymbol{Z}} \Delta R(\boldsymbol{Z}, \mathbf{\Pi})$
- Deep networks via optimizing rate reduction: $\dot{\boldsymbol{Z}} = \eta \cdot \frac{\partial \Delta R}{\partial \boldsymbol{Z}}$

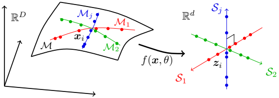
A Unified Framework:

- A principled objective for all settings of learning: **compression**
- A principled approach to interpret deep networks: **optimization**

"Everything should be made as simple as possible, but not simpler."
– Albert Einstein

Conclusions: Learn Linear Discriminative Representations

Compared to conventional deep neural networks:



	Conventional DNNs	Compression ReduNets
Objectives	label fitting	rate reduction
Deep architectures	trial & error	iterative optimization
Layer operators	empirical	projected gradient
Shift invariance	CNNs+augmentation	invariant ReduNets
Initializations	random/pre-design	forward computed
Training/fine-tuning	back prop	forward/back prop
Interpretability	black box	white box
Representations	hidden/latent	incoherent subspaces

Open Problems: Theory

$$\textbf{MCR}^2\text{:}\quad \max_{\boldsymbol{Z} \subset \mathbb{S}^{d-1}, \boldsymbol{\Pi} \in \Omega} \Delta R\big(\boldsymbol{Z}, \boldsymbol{\Pi}, \epsilon\big) = R(\boldsymbol{Z}, \epsilon) - R^c(\boldsymbol{Z}, \epsilon \mid \boldsymbol{\Pi}).$$

- **Phase transition** phenomenon in clustering via compression?
- Statistical justification for **robustness** of MCR^2 to label noise?
- **Optimal configurations** for broader conditions and distributions?
- Fundamental **tradeoff** between sparsity and invariance?
- **Jointly optimizing** both representation $\boldsymbol{Z}$ and clustering $\boldsymbol{\Pi}$?

$$\textbf{Joint Dynamics}\text{:}\quad \dot{\boldsymbol{Z}} = \eta \cdot \frac{\partial \Delta R}{\partial \boldsymbol{Z}}, \quad \dot{\boldsymbol{\Pi}} = \gamma \cdot \frac{\partial \Delta R}{\partial \boldsymbol{\Pi}}.$$

Open Problems: Architectures and Algorithms

ReduNet: $\bar{\boldsymbol{z}}_{\ell+1} \ \propto \ \bar{\boldsymbol{z}}_\ell + \eta \cdot \left[\bar{\boldsymbol{e}}_\ell \circledast \bar{\boldsymbol{z}}_\ell + \boldsymbol{\sigma}\big([\bar{\boldsymbol{c}}_\ell^1 \circledast \bar{\boldsymbol{z}}_\ell, \dots, \bar{\boldsymbol{c}}_\ell^k \circledast \bar{\boldsymbol{z}}_\ell]\big)\right] \in \mathbb{S}^{d-1}.$

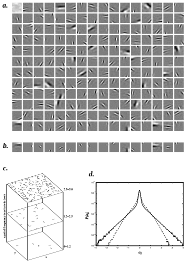
- New architectures from **accelerated** gradient schemes?
- Conditions for channel-wise **separable and short** convolutions?
- Architectures from invariant rate reduction for **other groups**?
- **Transformer**:[1] $\frac{1}{2}\log\det\left(\boldsymbol{I} + \alpha \boldsymbol{Z}\boldsymbol{Z}^*\right) = \frac{1}{2}\log\det\left(\boldsymbol{I} + \alpha \underbrace{\boldsymbol{Z}^*\boldsymbol{Z}}_{\boldsymbol{z}_i^*\boldsymbol{U}^*\boldsymbol{U}\boldsymbol{z}_j?}\right)$?
- Algorithmic architectures (or networks) for optimizing $\boldsymbol{\Pi}$?

[1] Attention: Self-Expression Is All You Need, Rene Vidal, 2022.

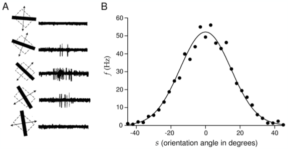
Open Directions: Extensions

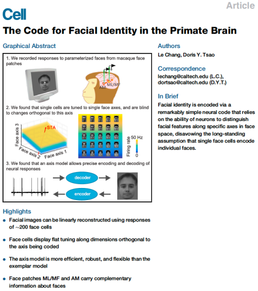
- Data with other **dynamical or graphical** structures.
- Better transferability and robustness w.r.t. **low-dim structures**.
- Combine with a **generative model** (a generator or decoder).
- Sparse coding, spectral computing, subspace embedding in **nature**.[2]

sparse coding in visual cortex



Rate coding hypothesis: the signal conveyed by a neuron is in the *rate* of spiking. Spiking irregularity is largely due to noise and does not convey information.





[2] figures from Bruno Olshausen of Neuroscience Dept., UC Berkeley.

From Open-Loop to Closed-Loop Representation

$$\text{MCR}^2: \quad \boldsymbol{X} \xrightarrow{f(\boldsymbol{x},\theta)} \boldsymbol{Z}(\theta): \quad \max_{\theta} \Delta R(\boldsymbol{Z}(\theta), \boldsymbol{\Pi}, \epsilon).$$

Features learned are more interpretable, independent, rich, and robust.

Nevertheless:

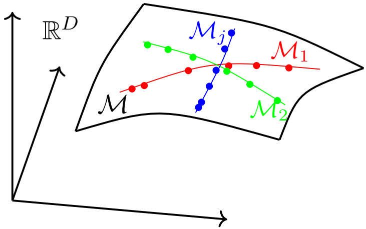
- Anything missing, anything unexpected: $\text{dim}(\boldsymbol{X}_j) = \text{dim}(\boldsymbol{Z}_j)$?
- Can we go from the feature $\boldsymbol{Z}_j$ back to the data $\boldsymbol{X}_j$?
- Is the learned LDR adequate to generate real-world (visual) data?

Can we establish an autoencoding between the data and the LDR:

$$\boldsymbol{X} \xrightarrow{f(\boldsymbol{x},\theta)} \boldsymbol{Z}(\theta) \xrightarrow{g(\boldsymbol{z},\eta)} \hat{\boldsymbol{X}}? \tag{28}$$

Low-dim Autoencoding for High-Dim Data

Assumption: the data $\boldsymbol{X}$ lie on a low-dimensional submanifold $\boldsymbol{X} \subset \mathcal{M}$ or multiple ones: $\boldsymbol{X} \subset \cup_{j=1}^{k} \mathcal{M}_j$ in a high-dimensional space $\in \mathbb{R}^D$:

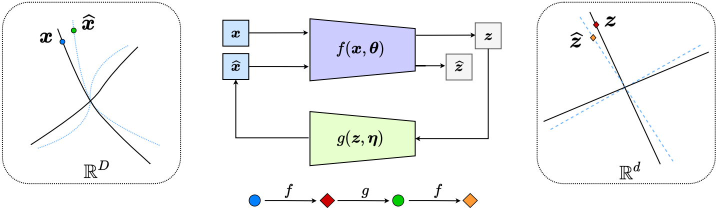


Goal: seeking a low-dim representation $\boldsymbol{Z}$ in $\mathbb{R}^d$ $(d \ll D)$ for the data $\boldsymbol{X}$ on low-dim submanfiolds such that:

$$\boldsymbol{X} \subset \mathbb{R}^D \xrightarrow{f(\boldsymbol{x},\theta)} \boldsymbol{Z} \subset \mathbb{R}^d \xrightarrow{g(\boldsymbol{z},\eta)} \hat{\boldsymbol{X}} \approx \boldsymbol{X} \in \mathbb{R}^D. \tag{29}$$

CTRL: Dual Roles of the Encoder and Decoder

f is both an encoder and sensor; and g is both a decoder and controller. They form a closed-loop feedback control system:



A closed-loop notion of "**self-consistency**" between Z and $\hat{Z}$ is achieved by a pursuit-evasion game between f as a "evader" and g as a "pursuer":

$$\mathcal{D}(\boldsymbol{X}, \hat{\boldsymbol{X}}) \doteq \max_{\theta} \min_{\eta} \sum_{j=1}^{k} \Delta R\big(\underbrace{f(\boldsymbol{X}_j, \theta)}_{\boldsymbol{Z}_j(\theta)}, \underbrace{f(g(f(\boldsymbol{X}_j, \theta), \eta), \theta)}_{\hat{\boldsymbol{Z}}_j(\theta,\eta)}\big). \tag{30}$$

Overall CTRL Objective: Self-Consistency & Parsimony

The overall minimax game between the encoder f and decoder g:

- f *maximizes* the rate reduction of the features $\boldsymbol{Z}$ of the data $\boldsymbol{X}$;
- g *minimizes* the rate reduction of the features $\hat{\boldsymbol{Z}}$ of the decoded $\hat{\boldsymbol{X}}$.

A minimax program to learn a **multi-class LDR** for data $\boldsymbol{X} = \cup_{j=1}^{k} \boldsymbol{X}_j$:

$$\max_{\theta} \min_{\eta} \underbrace{\Delta R\big(f(\boldsymbol{X}, \theta)\big)}_{\text{Expansive encode}} + \underbrace{\Delta R\big(h(\boldsymbol{X}, \theta, \eta)\big)}_{\text{Compressive decode}} + \sum_{j=1}^{k} \underbrace{\Delta R\big(f(\boldsymbol{X}_j, \theta), h(\boldsymbol{X}_j, \theta, \eta)\big)}_{\text{Contrastive \& Contractive}}$$

with $h(\boldsymbol{x}) \doteq f \circ g \circ f(\boldsymbol{x})$, or equivalently

$$\max_{\theta} \min_{\eta} \Delta R\big(\boldsymbol{Z}(\theta)\big) + \Delta R\big(\hat{\boldsymbol{Z}}(\theta, \eta)\big) + \sum_{j=1}^{k} \Delta R\big(\boldsymbol{Z}_j(\theta), \hat{\boldsymbol{Z}}_j(\theta, \eta)\big).$$

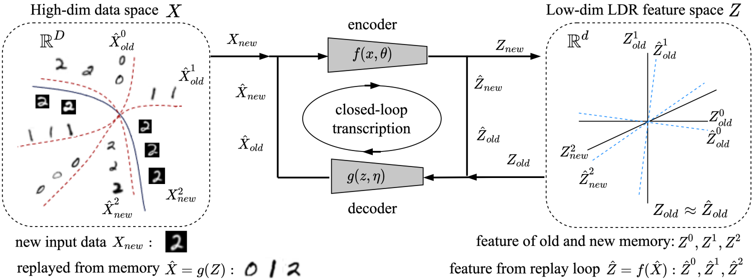
Characteristics of the Overall CTRL Objective

$$\max_{\theta} \min_{\eta} \Delta R\big(\boldsymbol{Z}(\theta)\big) + \Delta R\big(\hat{\boldsymbol{Z}}(\theta,\eta)\big) + \sum_{j=1}^{k} \Delta R\big(\boldsymbol{Z}_j(\theta), \hat{\boldsymbol{Z}}_j(\theta,\eta)\big).$$

- **Simplicity:** all terms are closed-form rate reduction on features.
- **Self-consistency:** enforced by closed-loop encoding and decoding.
- **Structured:** distribution of learned features $\boldsymbol{Z}$ is an LDR.
- **No** need to specify a prior or a surrogate target distribution.
- **No** need of any direct explicit distance between $\boldsymbol{X}$ and $\hat{\boldsymbol{X}}$.
- **No** more approximations or bounds for (KL-, JS-, W-) "distances".
- **No** heuristics or regularizing terms in the objective.

Parsimony and self-consistency are all you need to model $\boldsymbol{X}$?

Incremental Learning via Closed-Loop Transcription



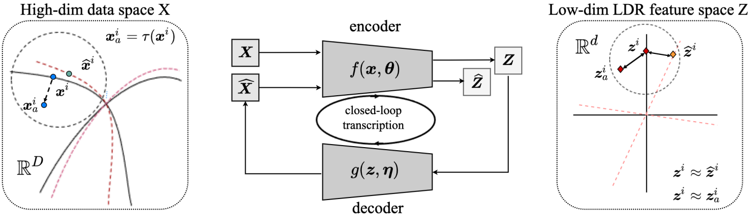
$$\max_{\theta} \min_{\eta} \quad \Delta R(\boldsymbol{Z}) + \Delta R(\hat{\boldsymbol{Z}}) + \Delta R(\boldsymbol{Z}_{new}, \hat{\boldsymbol{Z}}_{new})$$

$$\text{subject to} \quad \Delta R(\boldsymbol{Z}_{old}, \hat{\boldsymbol{Z}}_{old}) = 0. \tag{31}$$

Incremental Learning of Structured Memory, arXiv:2202.05411

under submission...

Unsupervised Learning via Closed-Loop Transcription



$$\max_{\theta} \min_{\eta} \quad R(\boldsymbol{Z}) + \Delta R(\boldsymbol{Z}, \hat{\boldsymbol{Z}}) \tag{32}$$

$$\text{subject to} \quad \sum_{i \in N} \Delta R(\boldsymbol{z}^i_{conv}, \hat{\boldsymbol{z}}^i_{conv}) = 0, \quad \text{and} \quad \sum_{i \in N} \Delta R(\boldsymbol{z}^i, \boldsymbol{z}^i_a) = 0.$$

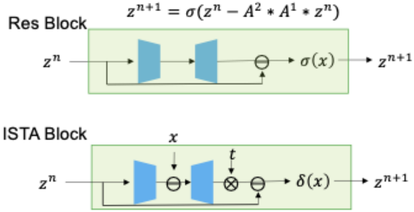
Unsupervised Learning of Structured Representations

under submission...

Whitebox Network as Sparse Dictionary Learning (SDNet)

Use iterative convolution sparse coding (via ISTA) as layers.

Model size, memory, speed, and accuracy all comparable to ResNet.



$$z^{n+1} = \sigma\left(z^n - t \cdot A^T \odot (A * z^n - x)\right)$$

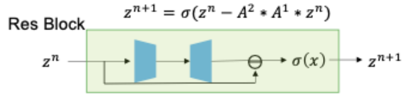
$$z^* = \operatorname*{argmin}_z \lambda \|z\|_1 + \frac{1}{2}\|x - A \odot z\|_2^2$$

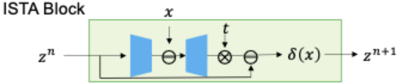
Dataset	Architecture	Model Size	Top-1 Acc	Memory	Speed
CIFAR-10	ResNet-18 [15]	11.2M	95.54%	1.0 GB	1600 n/s
	ResNet-34 [15]	21.1M	95.57%	2.0 GB	1000 n/s
	MDEQ [21]	11.1M	93.80%	2.0 GB	90 n/s
	SCN [9]	0.7M	94.36%	10.0GB	39 n/s
	SDNet-18 (ours)	11.2M	95.20%	1.2 GB	1500 n/s
	SDNet-34 (ours)	21.1M	95.57%	2.4 GB	900 n/s
CIFAR-100	ResNet-18 [15]	11.2M	77.82%	1.0 GB	1600 n/s
	ResNet-34 [15]	21.1M	78.39%	2.0 GB	1000 n/s
	MDEQ [21]	11.2M	74.12%	2.0 GB	90 n/s
	SCN [9]	0.7M	80.07%	10.0GB	39 n/s
	SDNet-18 (ours)	8.2M	78.31%	1.2 GB	1500 n/s
	SDNet-34 (ours)	10.2M	78.48%	2.4 GB	900 n/s
ImageNet	ResNet-18 [15]	11.7M	68.98%	24.1 GB	2100 n/s
	ResNet-34 [15]	21.5M	72.83%	32.3 GB	1400 n/s
	SCN [9]	9.8M	70.42%	95.1 GB	51 n/s
	SDNet-18 (ours)	11.7M	69.47%	37.6 GB	1800 n/s
	SDNet-34 (ours)	21.5M	72.67%	46.4 GB	1200 n/s

More Stable to Noises and More Robust to Perturbations.

under submission...

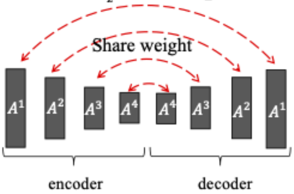
Transcription via Convolution Sparse Autoencoding

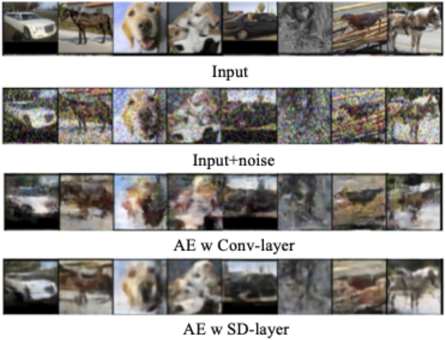




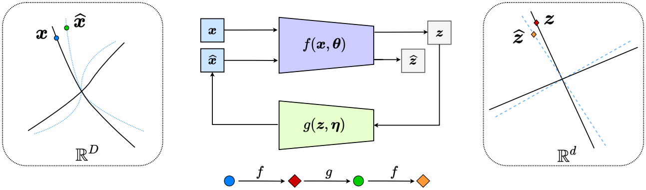
$$z^{n+1} = \sigma\left(z^n - t \cdot A^T \circledast (A * z^n - x)\right)$$

$$z^* = \operatorname*{argmin}_z \lambda \|z\|_1 + \frac{1}{2}\|x - A \circledast z\|_2^2$$





Conclusions: Closed-Loop Transcription to an LDR



- **universality:** transform external real-world data to an internal and structured linear discriminative representation.
- **parsimony:** a good tradeoff in rate reduction via a minimax game between an encoder and a decoder.
- **feedback:** a closed-loop feedback control system between a sensor and a controller.
- **self-consistency:** no need of any prior, surrogate, or error measure in the external data space.

Open Mathematical Problems about CTRL

For the closed-loop minimax rate reduction program:

$$\max_{\theta}\min_{\eta}\Delta R\big(\boldsymbol{Z}(\theta)\big)+\Delta R\big(\hat{\boldsymbol{Z}}(\theta,\eta)\big)+\sum_{j=1}^{k}\Delta R\big(\boldsymbol{Z}_j(\theta),\hat{\boldsymbol{Z}}_j(\theta,\eta)\big).$$

- **optimality:** characterization of the equilibrium points?
- **convergence** of the closed-loop control problem (infinite-dim)?
- **linearization** of distribution supports (plastic manifold learning)?
- **optimal density** of the distributions (*Brascamp-Lieb* inequalities)?
- **correct model selection** (no under- or over-fitting)?
- **guarantees** for approximate sample-wise auto-encoding?

Open Directions: Extensions and Connections

- How to **scale up** to hundreds and thousands of classes? (variational forms for rate reduction, CVPR'22...)
- Internal computational mechanisms for **memory** forming (in Nature)? (incremental/unsupervised learning without catastrophic forgetting.)
- Better **feedback** for generative quality and discriminative property?
- **Whitebox** architectures for closed-loop transcription (ReduNet like)?
- Closed-loop transcription to **other types of low-dim structures**? (dynamical, causal, logical, symbolical, graphical, genetic...)

The principles of parsimony and self-consistency shall always rule!

References: Learning Transcription via Rate Reduction

1. **CTRL: Closed-Loop Transcription to an LDR via Minimaxing Rate Reduction**
   https://arxiv.org/abs/2111.06636 (Entropy 2022)

2. **Incremental Learning** of Structured Memory via Closed-Loop Transcription
   https://arxiv.org/abs/2202.05411 (under submission)

3. **ReduNet:** A Whitebox Deep Network from Rate Reduction (JMLR 2022):
   https://arxiv.org/abs/2105.10446

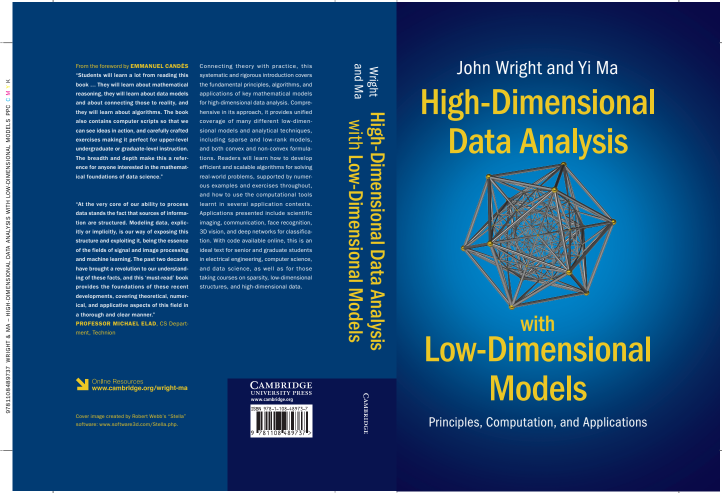
4. **Representation** via Maximal Coding Rate Reduction (NeurIPS 2020):
   https://arxiv.org/abs/2006.08558

5. **Classification** via Minimal Incremental Coding Length (NIPS 2007):
   http://people.eecs.berkeley.edu/~yima/psfile/MICL_SJIS.pdf

6. **Clustering** via Lossy Coding and Compression (TPAMI 2007):
   http://people.eecs.berkeley.edu/~yima/psfile/Ma-PAMI07.pdf

New Textbook: High-Dim Analysis with Low-Dim Models

https://book-wright-ma.github.io/



New Course: Computational Principles for High-Dim Data Analysis

Berkeley EECS 208:
https://pages.github.berkeley.edu/UCB-EECS208/course_site/

- Additional Textbook Information:
  https://book-wright-ma.github.io/
- Course Lectures and Supporting Materials:
  https://book-wright-ma.github.io/Lecture-Slides/

Parsimony and self-consistency are all you need to learn
a **compact and structured** model for real-world data.

Thank you!
Questions, please?

"What I cannot create, I do not understand."
– Richard Feynman

CEU/PDH Certificates

You can receive an CEU/PDH certificate by completing the course and pass the quiz. Here is the quiz/evaluation form:

https://bit.ly/ICASSP22_SC1