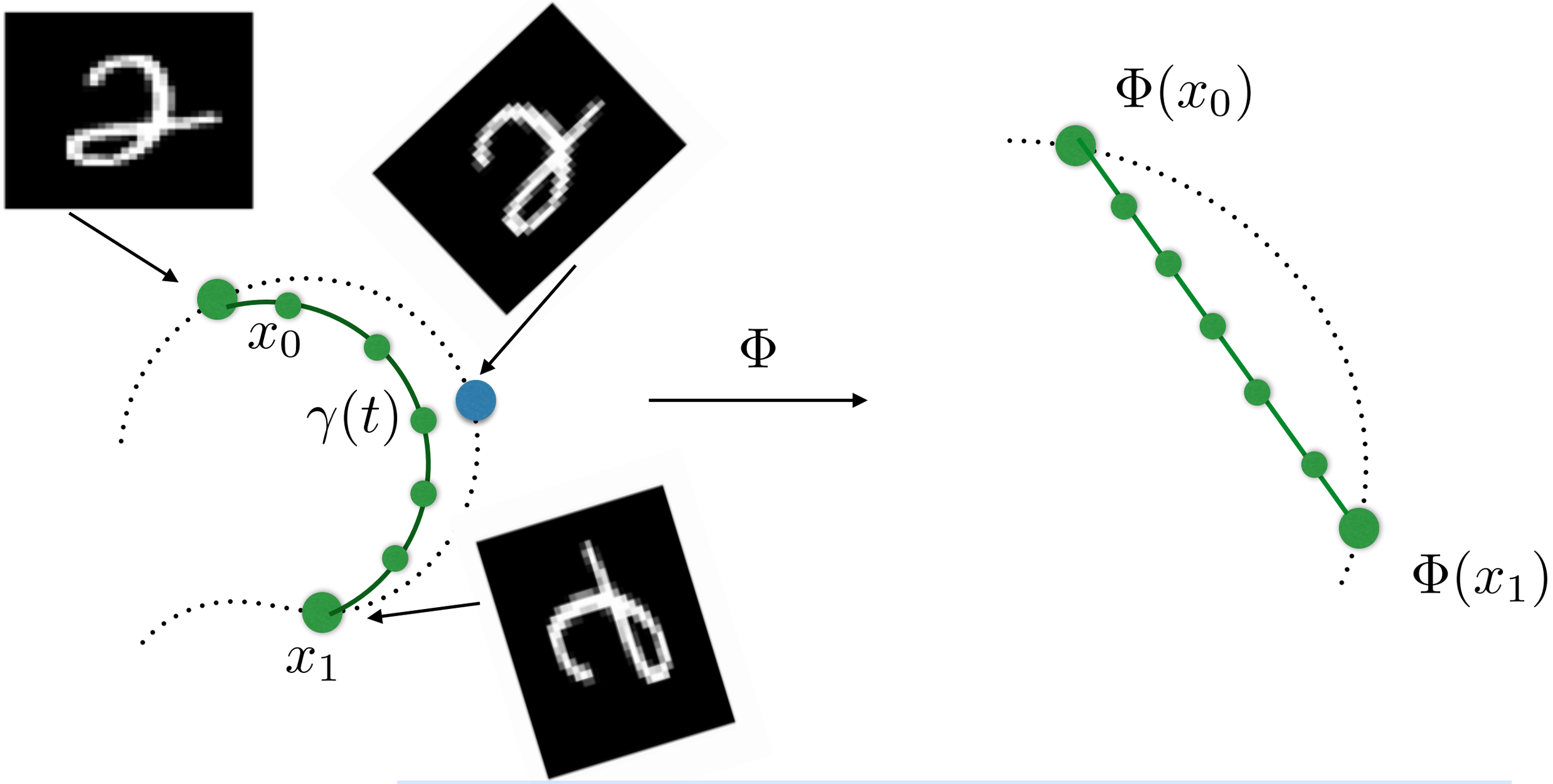
Stat 212b: Topics in Deep Learning
Lecture 8

Joan Bruna
UC Berkeley

Review: Invariance, Linearization and Geodesics

- Algorithm from [Henaff & Simoncelli '16]:



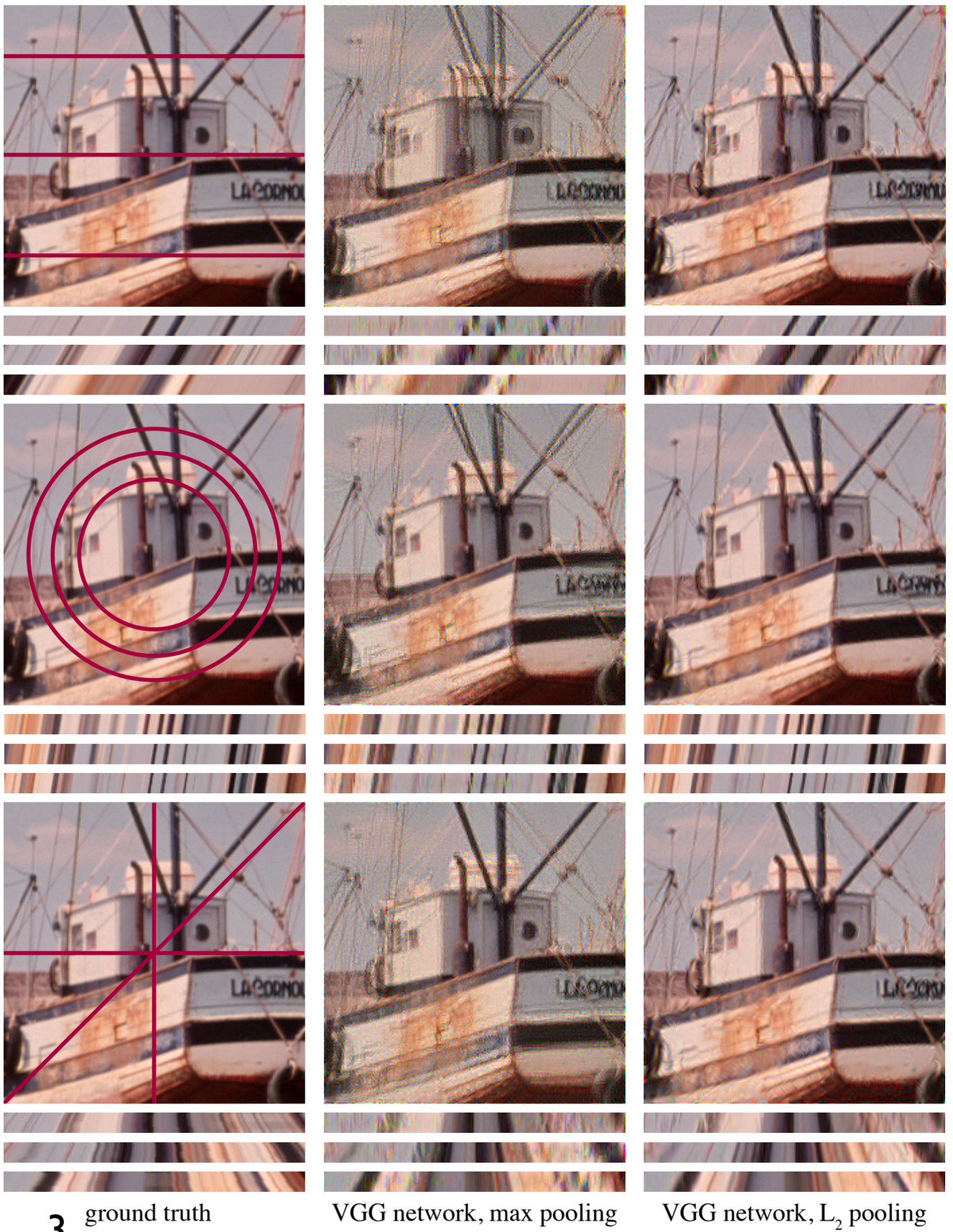
$$\min_{\gamma(0)=x_0,\gamma(1)=x_1} \int_0^1 |\dot{\gamma}(t)|dt + \int_0^1 |(\dot{\Phi\gamma})(t)|dt$$

Review:Invariance, Linearization and Geodesics

- On pertained CNNs (VGG oxford net), linearization is empirically verified for various groups.

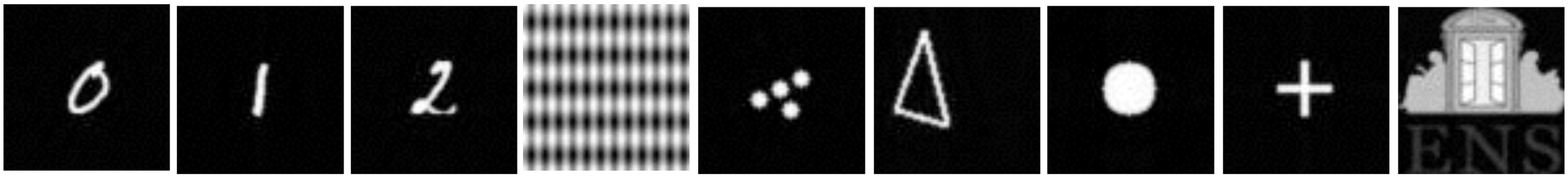
- Continuous transformation groups are better linearized with energy pooling than with max-pooling

[Henaff and Simoncelli'16]



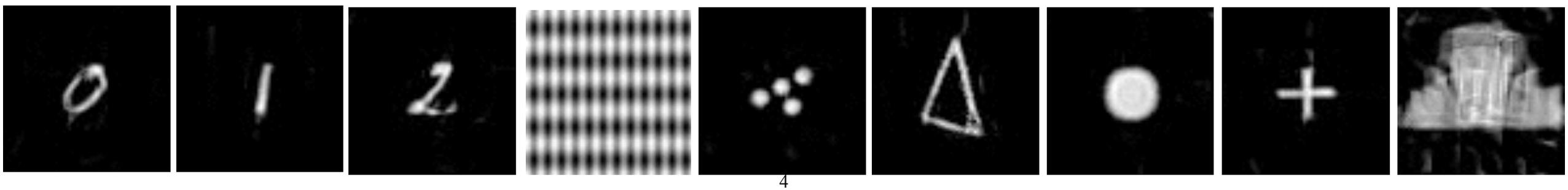
Review: Sparse Shape Scattering Reconstructions

Original images of N^2 pixels:



$m = 1$, $2^J = N$: reconstruction from $O(\log_2 N)$ scattering coeff.

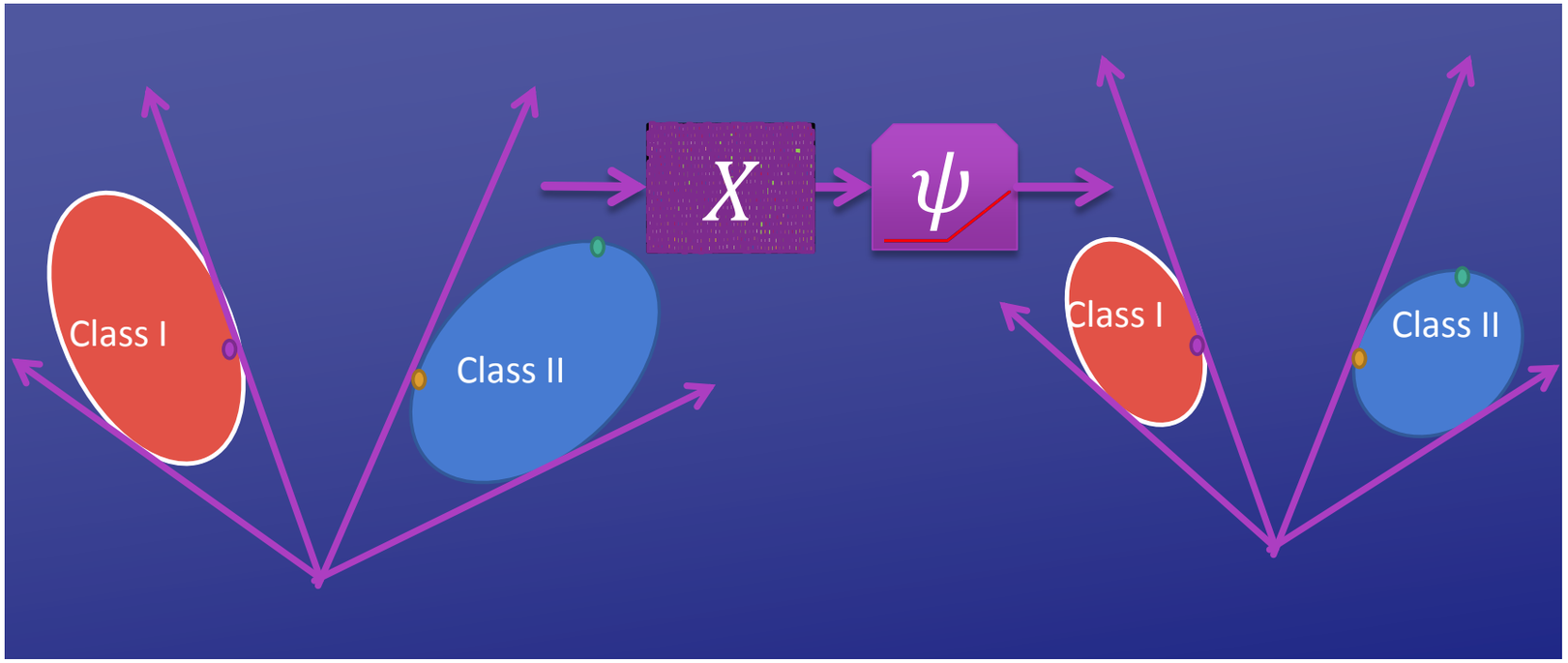
$m = 2$, $2^J = N$: reconstruction from $O(\log_2^2 N)$ scattering coeff.



Review: Random Networks

- If $\angle(x, y)$ is small, then $\beta(x, y) \approx 0$:
  distances are approx. shrunk by 2, angles are preserved.

- If $\angle(x, y)$ is large, then $\beta(x, y) \approx 0.5$:
  distances are shrunk by a smaller factor.

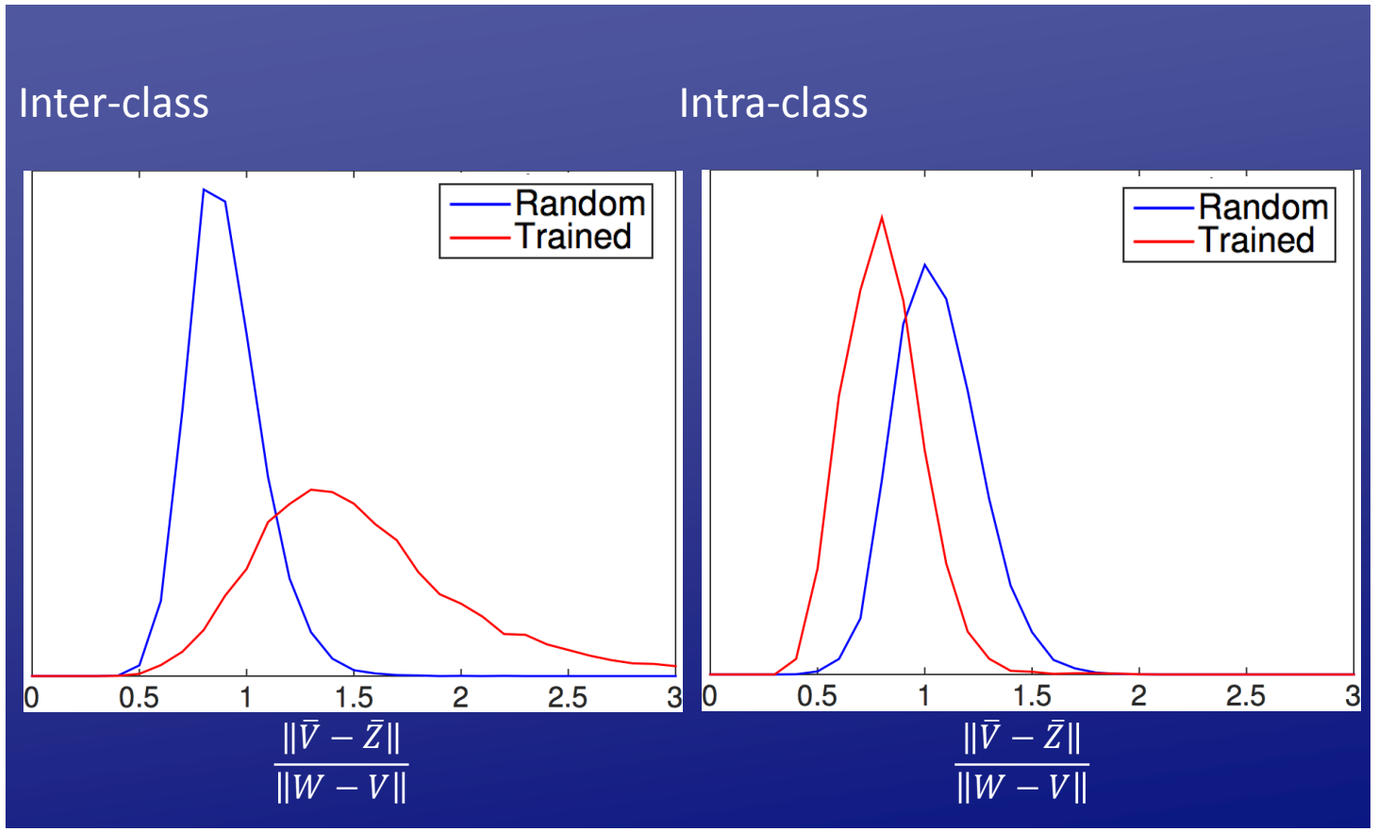


[Raja Giryes]

points with small angles between them become closer
than points with larger angles between them

The result can be cascaded since gaussian mean width is approximately preserved by each layer.

Review: Role of Training?



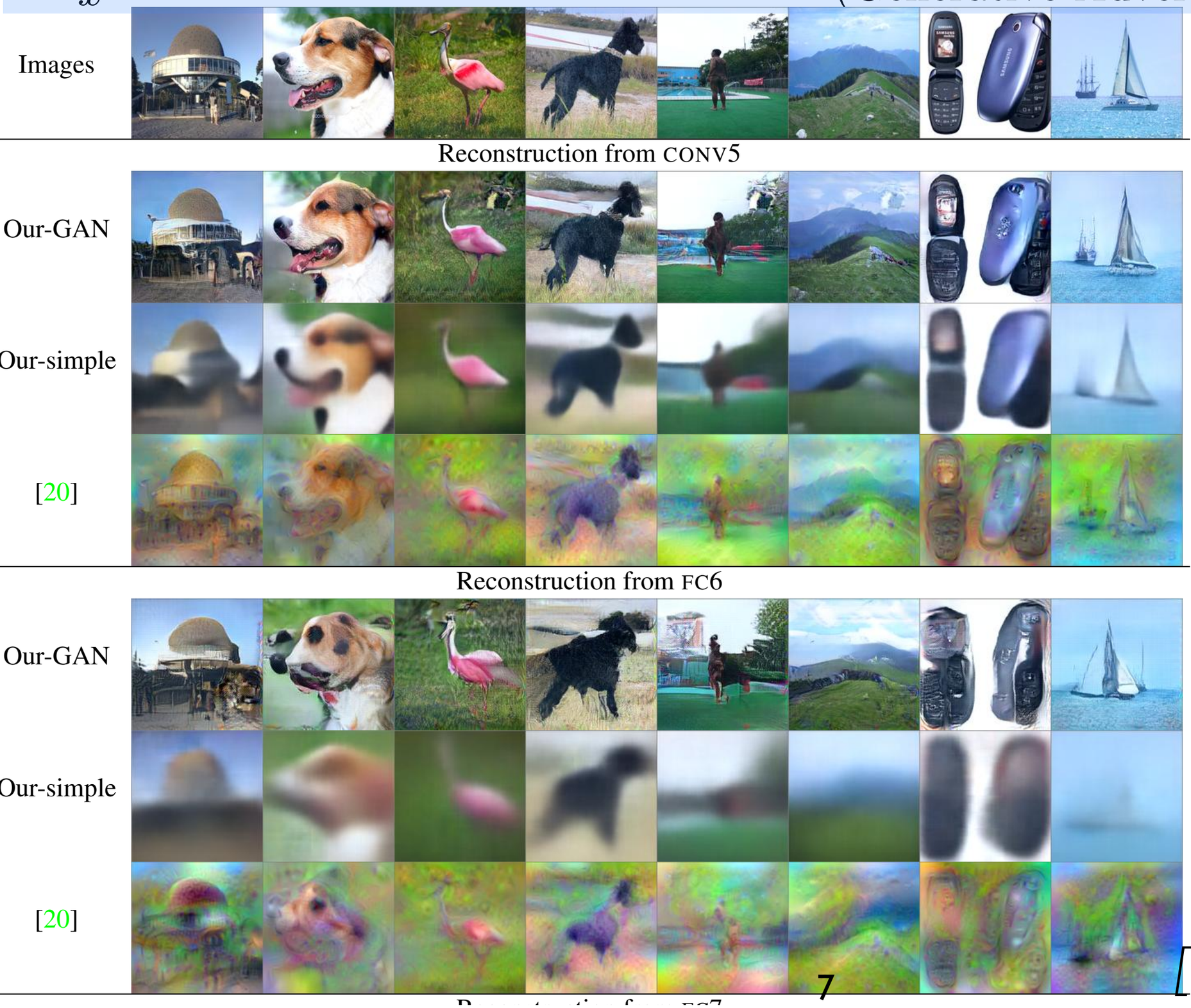
[Raja Giryes]

- Training the network does not affect the *bulk* of distances
- However, it critically changes the behavior at the boundary points:
  - Inter-class distances expand (as expected).
  - Intra-class distances shrink (as expected).

Review: Empirical Recovery

$$\min_x \|\Phi(x) - \Phi(x_0)\|^2 + \mathcal{R}(x)$$

$\mathcal{R}(x)$: Regularization with "learnt" prior (Generative Adversarial Networks, TBD)



[Dosovitsky & Brox'15]

Objectives

- Inference models and Deep Networks
  - Clustering and Dictionary Learning
  - From Unsupervised to Supervised Sparsity
  - From Inference to Deep Neural Networks
  - Examples

- Fisher Vectors and Pyramid Kernels

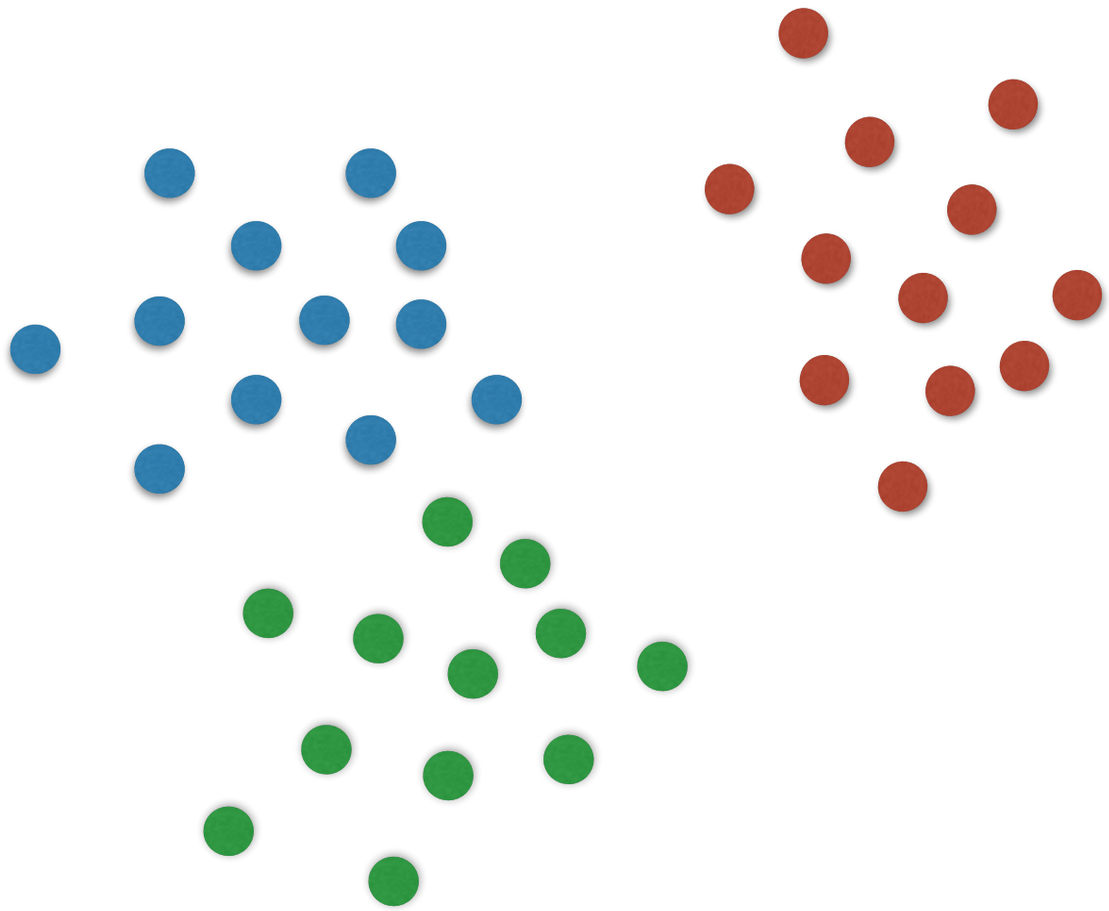
- Random Forests and CART

Selection Models

- Q: How to increase separation between classes?

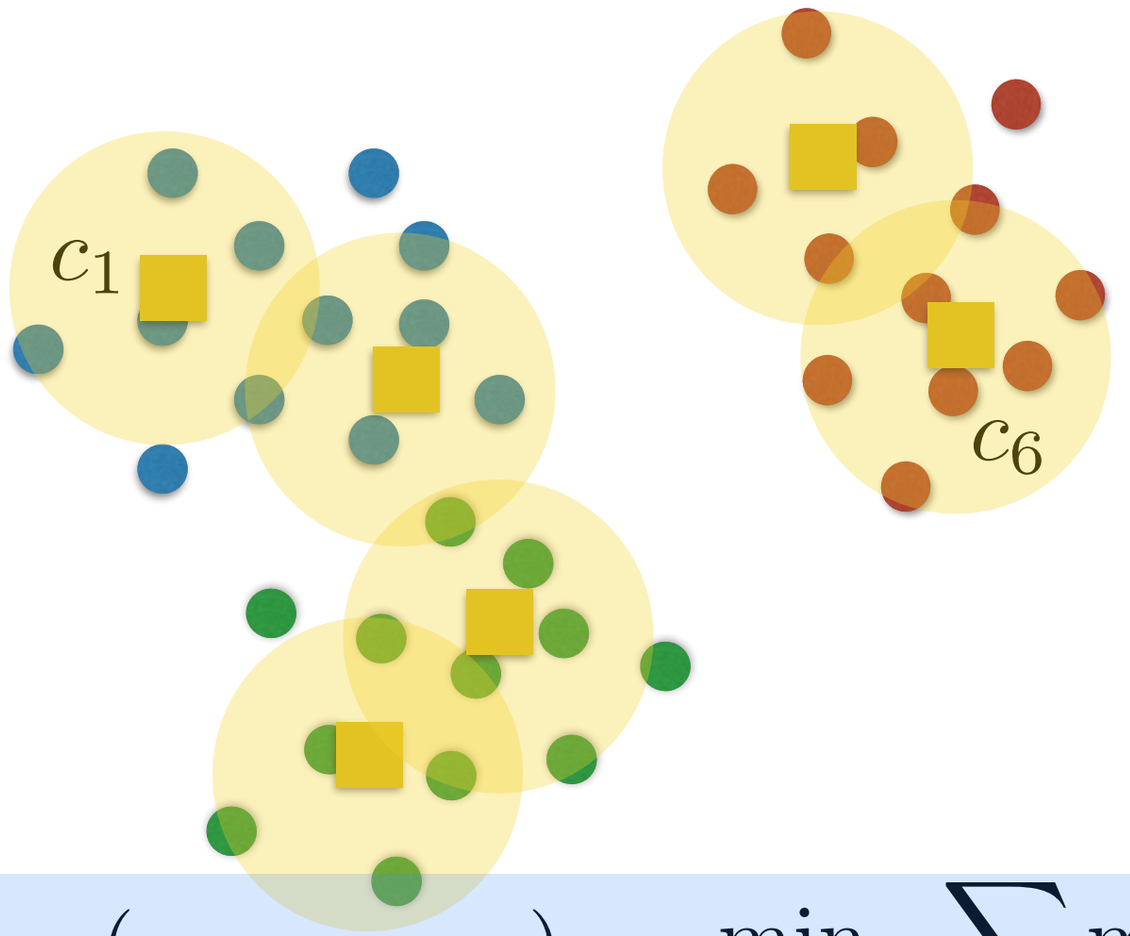
Selection Models

- Q: How to increase separation between classes?
- The simplest model is K-means clustering:



Selection Models

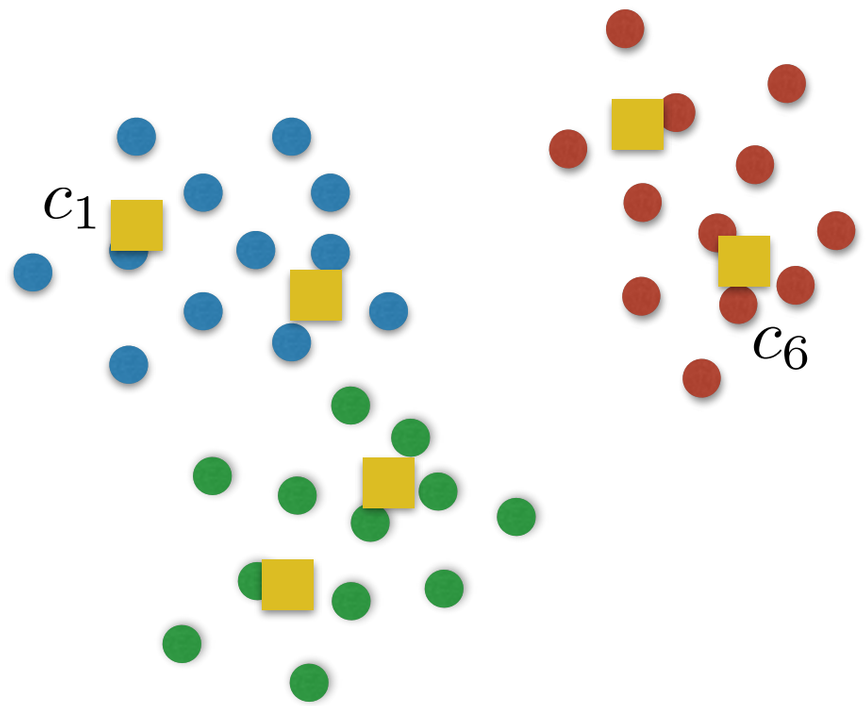
- Q: How to increase separation between classes?
- The simplest model is K-means clustering:



$K = 6$

Given data $X = (x_1, \ldots, x_n)$, $\min_{c_1,\ldots,c_K} \sum_{i \leq n} \min_j \|x_i - c_j\|^2$

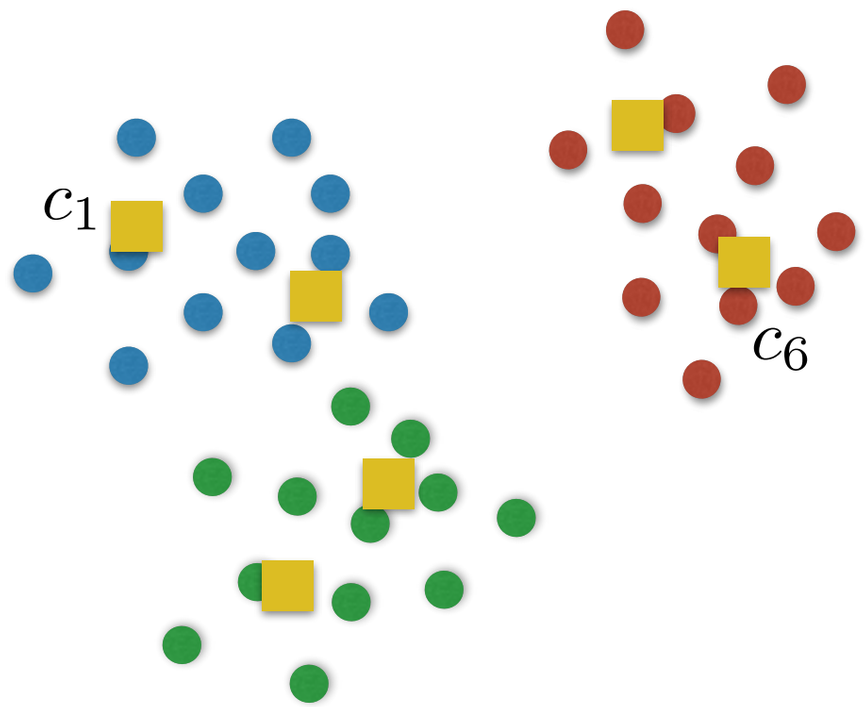
Selection Models



- K-means defines a mapping:

$$\Phi \;:\; \mathbb{R}^m \to \mathbb{R}^K$$
$$x \mapsto e_{k(x)} \;,\; k(x) = \arg\min_j \|x - c_j\|$$

Selection Models



- K-means defines a mapping:

$$\Phi \; : \; \mathbb{R}^m \to \mathbb{R}^K$$
$$x \mapsto e_{k(x)} \; , \; k(x) = \arg\min_j \|x - c_j\|$$

- Assuming power-normalized data ($\|x\| = 1$), Φ maximally separates points falling into different clusters:
  ($\langle \Phi(x), \Phi(y) \rangle = 0$ in that case)

Selection Models

- The K-means encoding is extremely naïve: log(K) bits encoding which region of input space we fall into (piecewise constant encoding)
  - It is nevertheless a very competitive encoding for small image patches.

Selection Models

- The K-means encoding is extremely naïve: log(K) bits encoding which region of input space we fall into (piecewise constant encoding)
  - It is nevertheless a very competitive encoding for small image patches.
- A strictly richer model is the union of subspaces model or dictionary learning:

$$\min_{D=(d_1,\ldots,d_K),\|d_k\|\leq 1,z} \sum_{i\leq n} \|x_i - Dz_i\|^2 + \lambda \mathcal{R}(z_i)$$

$\mathcal{R}(z)$: sparsity-promoting

$\mathcal{R}(z) = \|z\|_0$ (NP-Hard)

$\mathcal{R}(z) = \|z\|_1$ (Tractable)

Selection Models

- For a given dictionary D, the *sparse coding* is defined as the mapping

$$\begin{aligned} \Phi \; : \; \mathbb{R}^m &\to \mathbb{R}^K \\ x &\mapsto \Phi(x) = \arg\min_z \|x - Dz\|^2 + \lambda \mathcal{R}(z) \; . \end{aligned}$$

Selection Models

- For a given dictionary D, the *sparse coding* is defined as the mapping

$$\begin{aligned} \Phi \; : \; & \mathbb{R}^m \to \mathbb{R}^K \\ & x \mapsto \Phi(x) = \arg\min_z \|x - Dz\|^2 + \lambda \mathcal{R}(z) \; . \end{aligned}$$

- A particularly attractive choice is $\mathcal{R}(z) = \|z\|_1$
  - in that case $\Phi(x)$ requires solving a convex program.
  - Lasso estimator [Tibshirani,'96]
  - Rich theory in the statistical community.
  - Extensions: Group Lasso, Hierarchical Lasso, etc.

Proximal Splitting

- The sparse coding involves minimizing a function of the form

$$\min_z h_1(z) + h_2(z)$$

$h_1(z) = \|x - Dz\|^2$ convex and smooth (differentiable)

$h_2(z) = \lambda\|z\|_1$ convex but non-smooth

Proximal Splitting

- The sparse coding involves minimizing a function of the form

$$\min_z h_1(z) + h_2(z)$$

  - $h_1(z) = \|x - Dz\|^2$ convex and smooth (differentiable)
  - $h_2(z) = \lambda\|z\|_1$ convex but non-smooth

- A solution can be obtained by alternatively minimizing each term:

Fact: Let $h : \mathbb{R}^m \to \mathbb{R}$ be a convex function. For every $z \in \mathbb{R}^m$,

$$\min_y h(y) + \frac{1}{2}\|z - y\|^2$$

has unique solution, denoted $\text{prox}_h(z)$.

(prox_h is a non-expansive operator for all h)

Forward-Backward Splitting

- It can be shown that if h_1 is convex and differentiable with Lipschitz gradient, and h_2 is convex, then the solutions of

$$\min_z h_1(z) + h_2(z)$$

are characterized by the fixed points of

$$z = \text{prox}_{\gamma h_2}(z - \gamma \nabla h_1(z)) \; \forall \; \gamma \geq 0.$$

Forward-Backward Splitting

- It can be shown that if h_1 is convex and differentiable with Lipschitz gradient, and h_2 is convex, then the solutions of

$$\min_z h_1(z) + h_2(z)$$

are characterized by the fixed points of

$$z = \text{prox}_{\gamma h_2}(z - \gamma \nabla h_1(z)) \; \forall \; \gamma \geq 0.$$

- These can be found by iterating

$$z_{n+1} = \text{prox}_{\gamma_n h_2}(z_n - \gamma_n \nabla h_1(z_n))$$

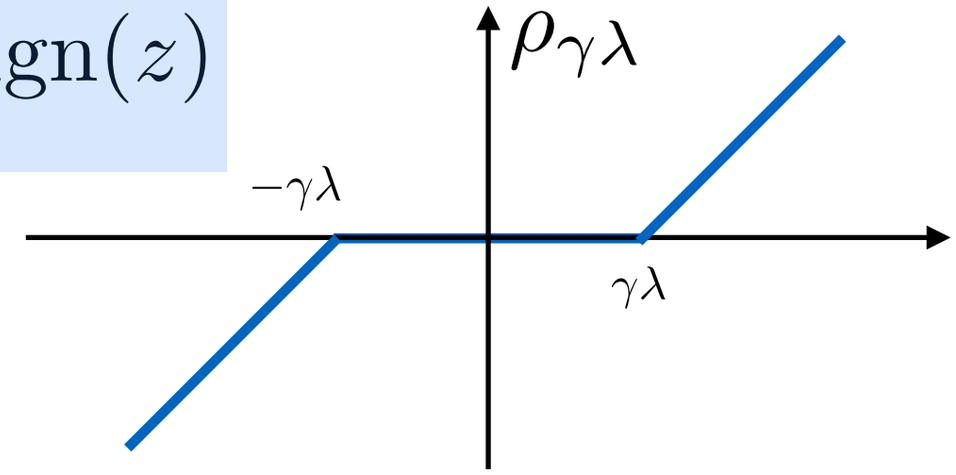
  - by properly adjusting the rate γ_n these method is proven to converge to its unique solution.

Proximal Splitting and ISTA

- When $h_2(z) = \lambda \|z\|_1$, the proximal operator becomes

$$\text{prox}_{\gamma h_2}(z) = \max(0, |z| - \gamma\lambda) \cdot \text{sign}(z)$$

$\rho_{\gamma\lambda}$: soft thresholding

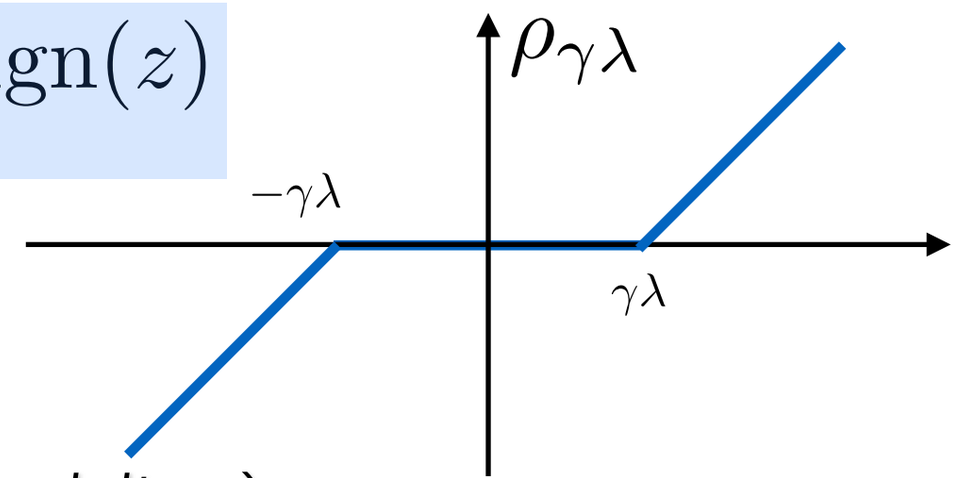


Proximal Splitting and ISTA

- When $h_2(z) = \lambda \|z\|_1$, the proximal operator becomes

$$\text{prox}_{\gamma h_2}(z) = \max(0, |z| - \gamma\lambda) \cdot \text{sign}(z)$$

$\rho_{\gamma\lambda}$: soft thresholding



- ISTA algorithm (*iterative soft thresholding*):

$$z_{n+1} = \text{prox}_{\gamma_n h_2}(z_n - \gamma_n \nabla h_1(z_n))$$

$$\nabla h_1(z_n) = -D^T(x - Dz_n)$$

$$z_{n+1} = \rho_{\gamma\lambda}((\mathbf{1} - \gamma D^T D)z_n + \gamma D^T x)$$

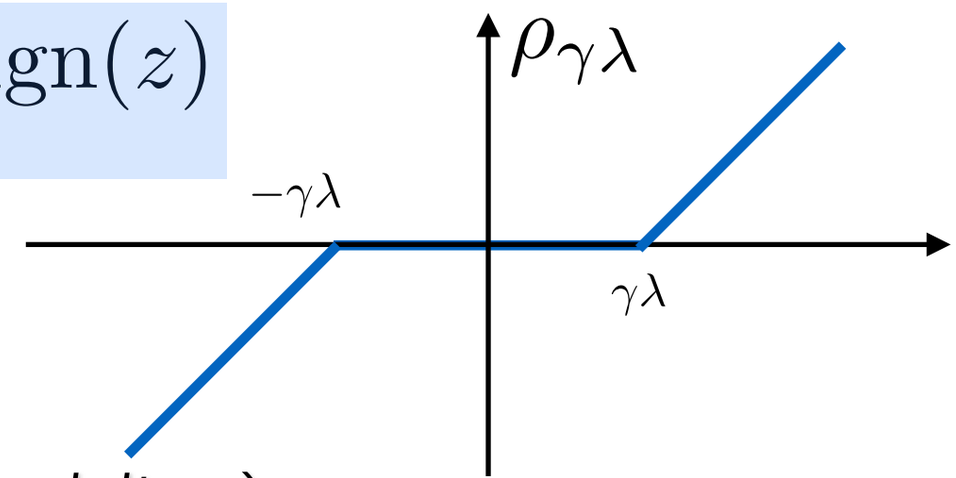
- converges in sublinear time $O(1/n)$ if $\gamma \in (0, 1/\|D^T D\|)$

Proximal Splitting and ISTA

- When $h_2(z) = \lambda \|z\|_1$, the proximal operator becomes

$$\text{prox}_{\gamma h_2}(z) = \max(0, |z| - \gamma\lambda) \cdot \text{sign}(z)$$

$\rho_{\gamma\lambda}$: soft thresholding



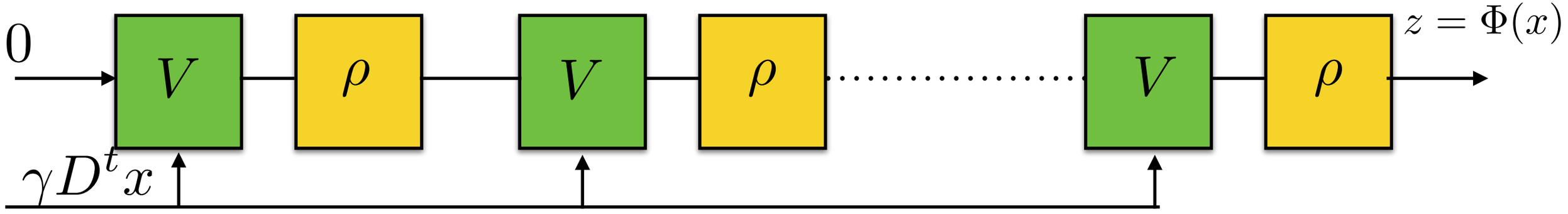
- ISTA algorithm (*iterative soft thresholding*):

$$z_{n+1} = \text{prox}_{\gamma_n h_2}(z_n - \gamma_n \nabla h_1(z_n))$$
$$\nabla h_1(z_n) = -D^T(x - Dz_n)$$
$$z_{n+1} = \rho_{\gamma\lambda}((\mathbf{1} - \gamma D^T D)z_n + \gamma D^T x)$$

  - converges in sublinear time $O(1/n)$ if $\gamma \in (0, 1/\|D^T D\|)$

- FISTA [*Beck and Teboulle,'09*]:
  - adds Nesterov momentum.
  - proven accelerated convergence $O(1/n^2)$

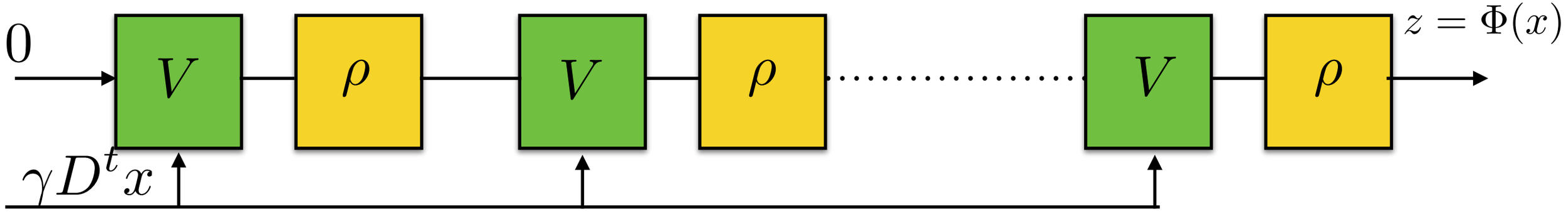
Sparse Coding with (F)ISTA



$Vz = (\mathbf{1} - \gamma D^T D)z + \gamma D^T x)$: linear with bias

ρ: pointwise non-linearity

Sparse Coding with (F)ISTA



$Vz = (\mathbf{1} - \gamma D^T D)z + \gamma D^T x)$: linear with bias

ρ: pointwise non-linearity

- Lasso can be cast as a (very) deep network, with
  - Shared weights, adapted to the dictionary.

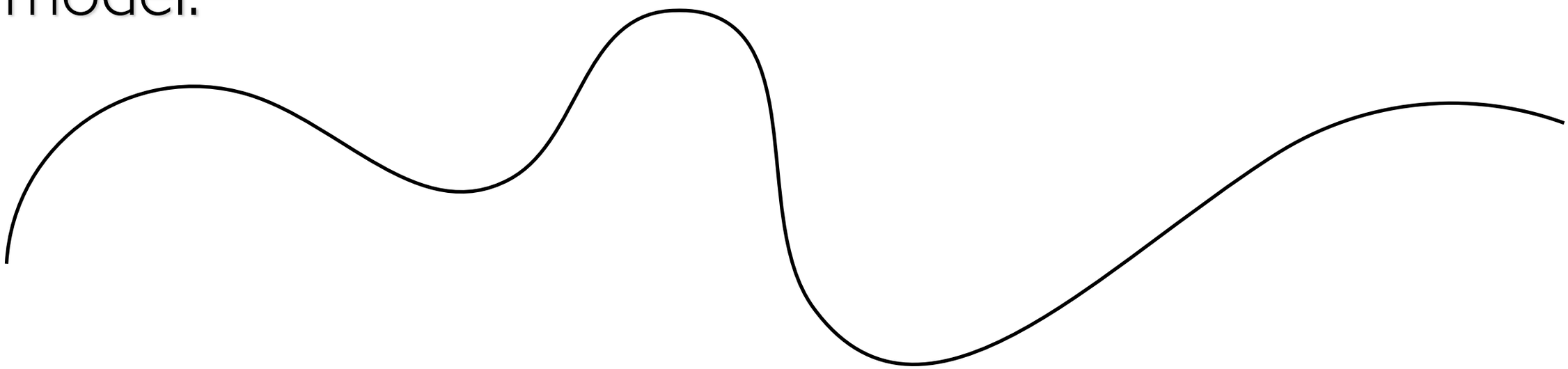
$$A = \mathbf{1} - \gamma D^T D \quad , \quad B = \gamma D^T$$
$$\Phi_{n+1}(x) = \rho(A\Phi_n(x) + Bx)$$

  - Note that A is a contraction $(\|Ax\| \leq \|x\|)$, but the affine term may increase the separation:

$$\begin{aligned}\|\Phi_{k+1}(x) - \Phi_{k+1}(x')\| &\leq \|A(\Phi_k(x) - \Phi_k(x'))\| + \|B(x - x')\| \\ &\leq \|\Phi_k(x) - \Phi_k(x')\| + \|B(x - x')\|\end{aligned}$$

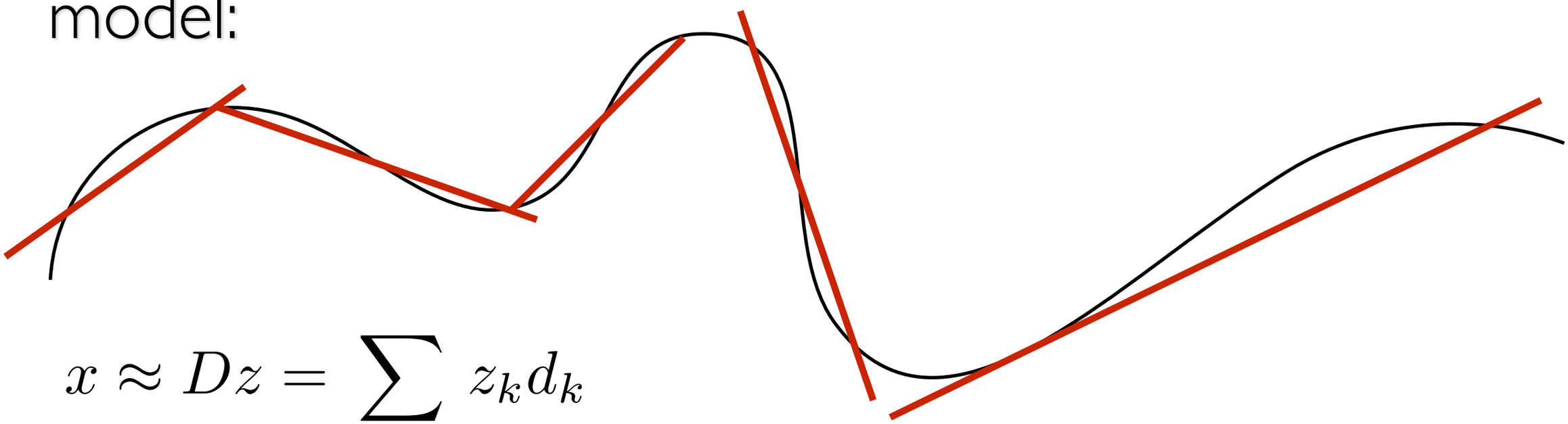
Geometric Interpretation

- Dictionary learning is a locally linear approximation model:



Geometric Interpretation

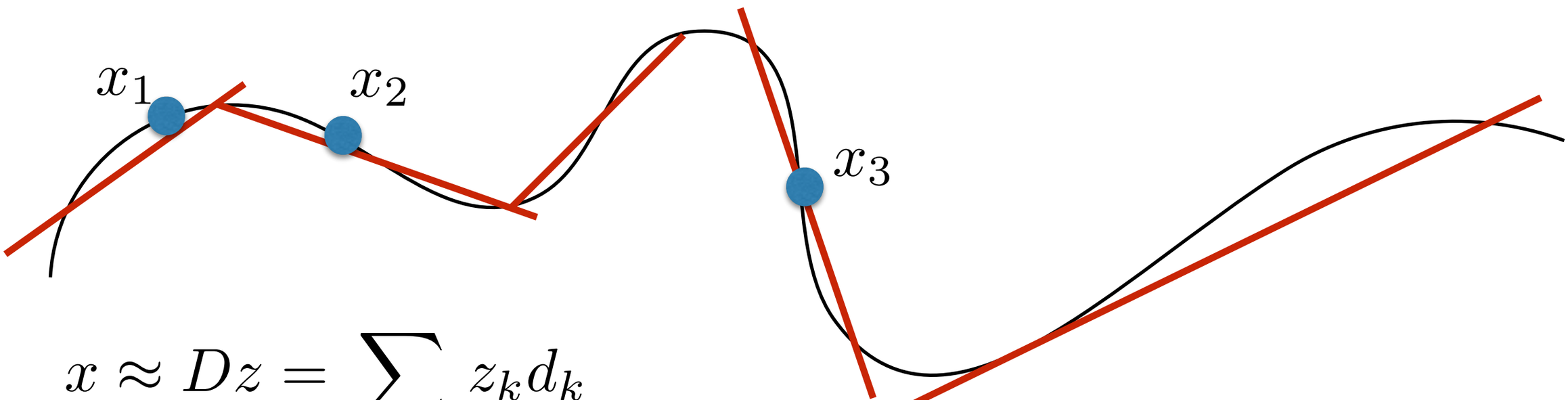
- Dictionary learning is a locally linear approximation model:



$$x \approx Dz = \sum_{z_k \neq 0} z_k d_k$$

Geometric Interpretation

- Orthogonalization of different linear pieces:

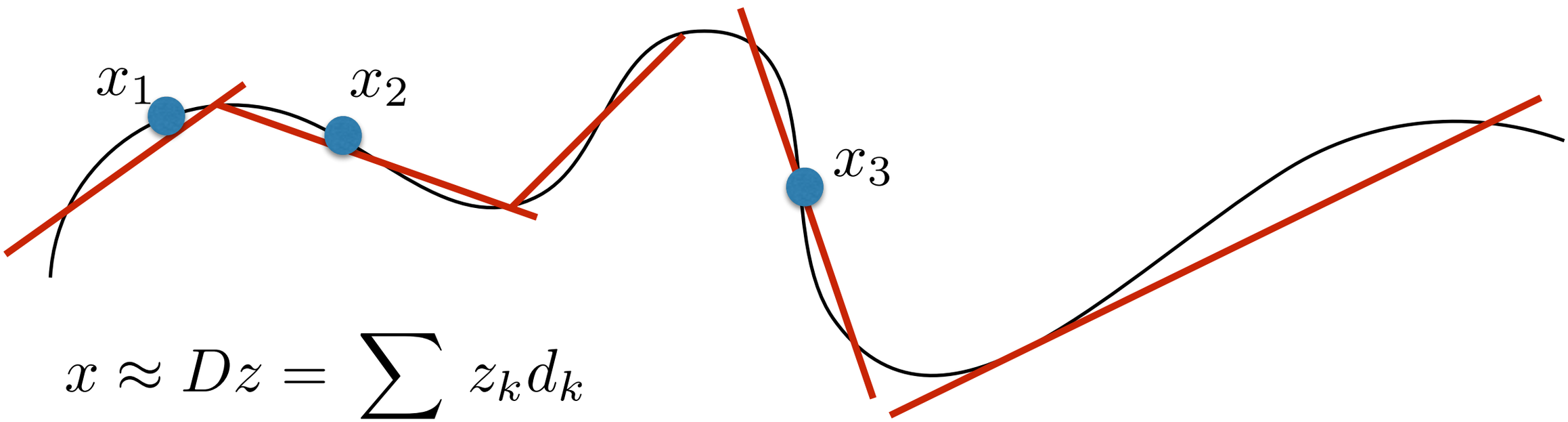


$$x \approx Dz = \sum z_k d_k$$

$$z_k \neq 0$$

Geometric Interpretation

- Orthogonalization of different linear pieces:



$$x \approx Dz = \sum_{\substack{z_k \neq 0}} z_k d_k$$

- If x_1 and x_2 share most dictionary atoms J, then

$$\langle \Phi(x_1), \Phi(x_2) \rangle \approx \langle D_J^T x_1, D_J^T x_2 \rangle = \langle x_1, D_J D_J^T x_2 \rangle$$

If x_1 and x_3 do not share dictionary atoms, then

$$\langle \Phi(x_1), \Phi(x_3) \rangle \approx 0$$

Sparse Coding and Stability

Linear decoder implies geometric instability is preserved in the sparse decomposition

$$\|x - Dz\| \leq \epsilon\|x\| \quad , \quad \|\varphi_\tau x - Dz_\tau\| \leq \epsilon\|x\| \quad , \quad \|x - \varphi_\tau x\| \sim \|x\|$$

$$\Downarrow$$

$$\|x - \varphi_\tau x\| \leq \|Dz - Dz_\tau\| + 2\epsilon$$

$$\Downarrow$$

$$\begin{aligned}\|z - z_\tau\| &\geq \|D\|_\infty^{-1}\|Dz - Dz_\tau\| \\ &\geq \|D\|_\infty^{-1}(\|x - \varphi_\tau x\| - 2\epsilon\|x\|) \\ &\sim \|D\|_\infty^{-1}(1 - 2\epsilon)\|x\|\end{aligned}$$

From unsupervised to supervised selection

- The previous model is unsupervised:
  - Why would a dictionary for reconstruction be useful for recognition or other tasks?
  - Pro: it exploits the local regularity of the data.
  - Cons: sparse coding unaware of stability, sparse dictionaries might be not unique.

From unsupervised to supervised selection

- The previous model is unsupervised:
  - Why would a dictionary for reconstruction be useful for recognition or other tasks?
  - Pro: it exploits the local regularity of the data.
  - Cons: sparse coding unaware of stability, sparse dictionaries might be not unique.

- Q: Can we make a dictionary task-aware? (i.e. supervised dictionary learning)

From unsupervised to supervised selection

- Task-driven dictionary learning [Mairal et al,'12]:

  Suppose we want to predict $y \in \mathcal{Y}$ from $x \in \mathcal{X}$

From unsupervised to supervised selection

- Task-driven dictionary learning [Mairal et al,'12]:

Suppose we want to predict $y \in \mathcal{Y}$ from $x \in \mathcal{X}$

Consider the sparse coding operator

$$\Phi(x; D) = \arg\min_z \frac{1}{2}\|x - Dz\|^2 + \lambda\|z\|_1 + \lambda_2\|z\|_2^2$$

It is Lipschitz with respect to both x and D if $\lambda_2 > 0$, it is differentiable almost everywhere.

From unsupervised to supervised selection

- Task-driven dictionary learning [Mairal et al,'12]:

Suppose we want to predict $y \in \mathcal{Y}$ from $x \in \mathcal{X}$

Consider the sparse coding operator

$$\Phi(x; D) = \arg\min_z \frac{1}{2}\|x - Dz\|^2 + \lambda\|z\|_1 + \lambda_2\|z\|_2^2$$

It is Lipschitz with respect to both x and D if $\lambda_2 > 0$, it is differentiable almost everywhere.

We can construct an estimator $\hat{y}$ from this sparse code:

$\hat{y} = W^T\Phi(x; D)$ (more generally, , $\hat{y} = F(W, \Phi(x; D))$)

$$\min_{D,W} \mathbb{E}_{x,y}\ell(y, \hat{y}(x, W, D))$$

From unsupervised to supervised

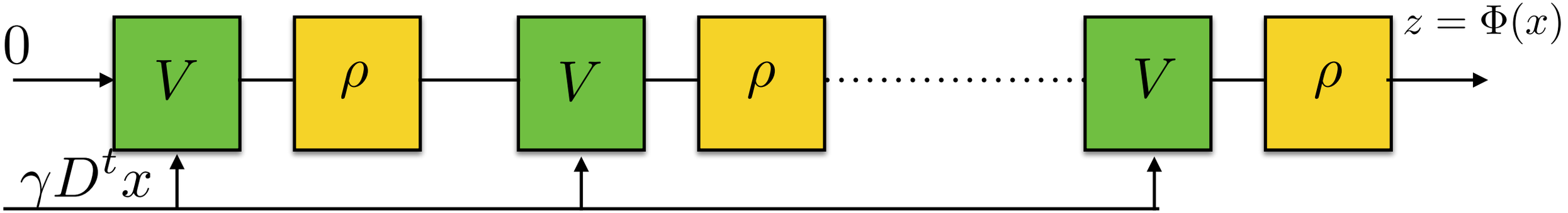
- Half-toning Results from [Mairal et al,'12]:

From supervised Lasso to DNNs

- The Lasso (sparse coding operator) can be implemented as a specific deep network.

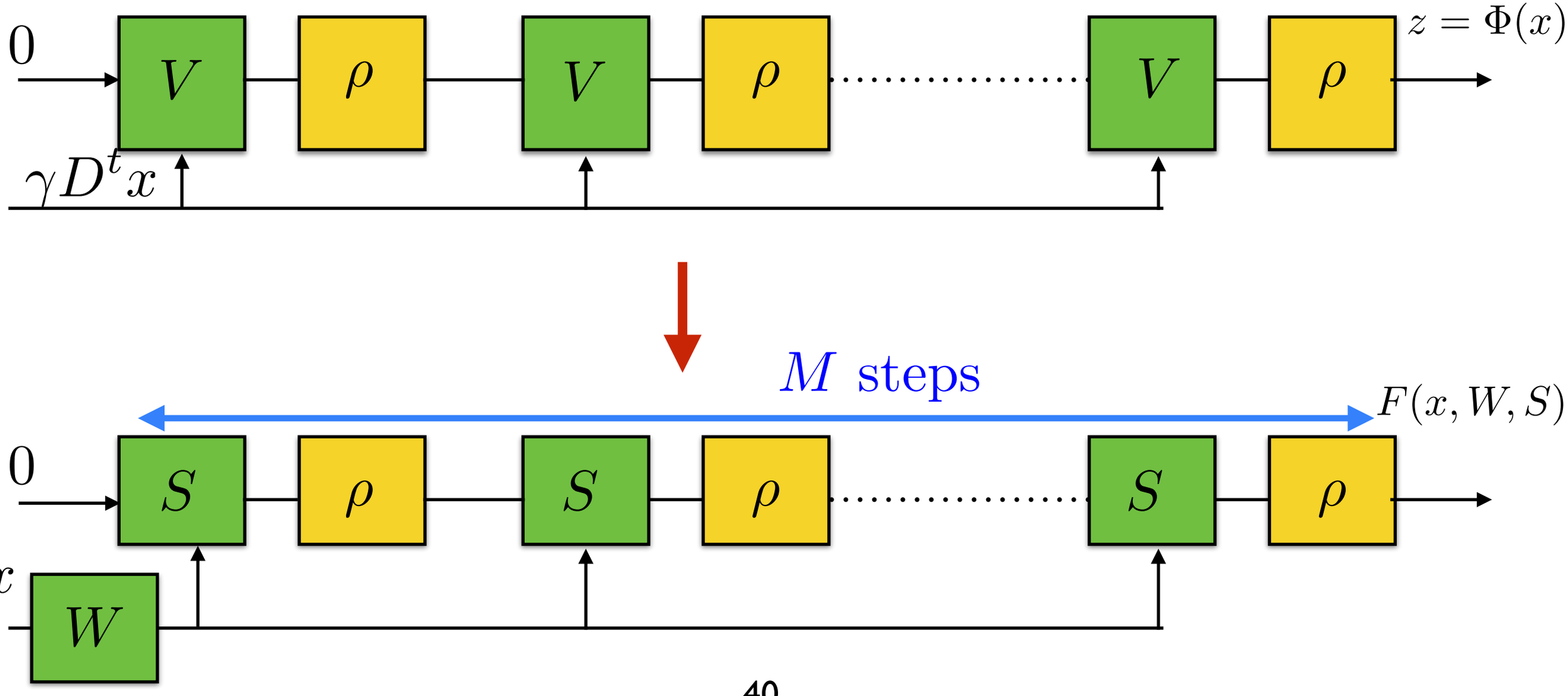
From supervised Lasso to DNNs

- The Lasso (sparse coding operator) can be implemented as a specific deep network
- Can we accelerate the sparse inference with a shallower network, with trained parameters?



From supervised Lasso to DNNs

- The Lasso (sparse coding operator) can be implemented as a specific deep network
- Can we accelerate the sparse inference with a shallower network, with trained parameters?



LISTA [Gregor and LeCun,'10]

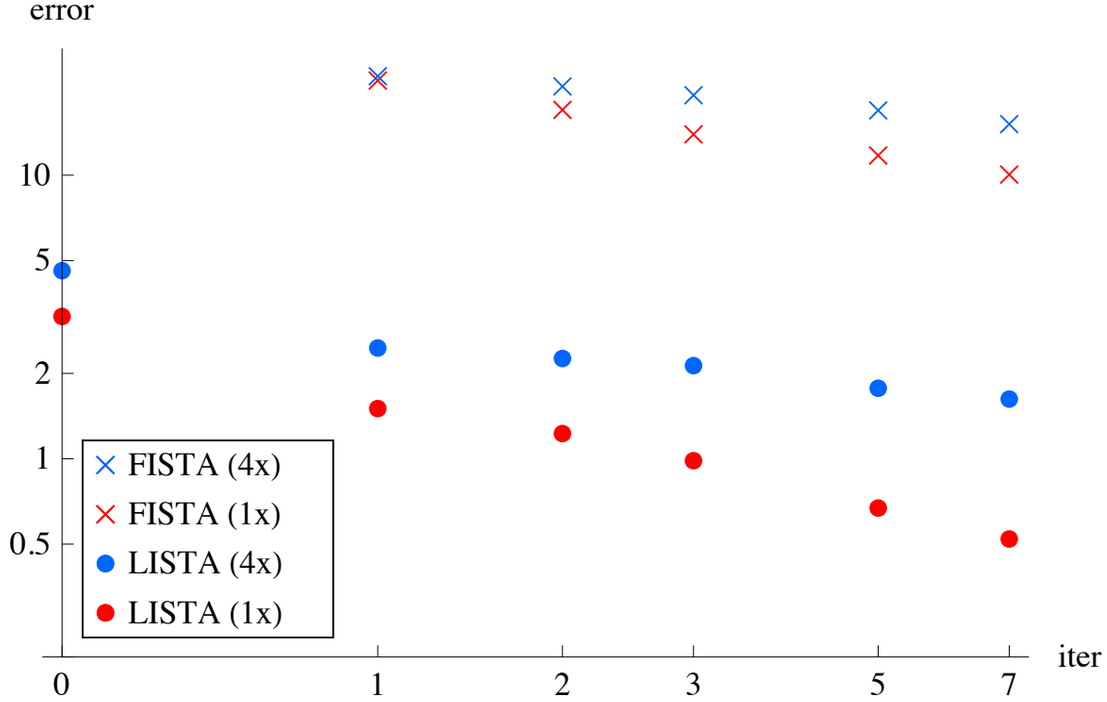
- Explicit Sparse encoder trained to predict the output of the Lasso:

$$\min_{W,S} \frac{1}{n} \sum_{i \leq n} \|\Phi(x_i) - F(x_i, W, S)\|^2$$

LISTA [Gregor and LeCun,'10]

- Explicit Sparse encoder trained to predict the output of the Lasso:

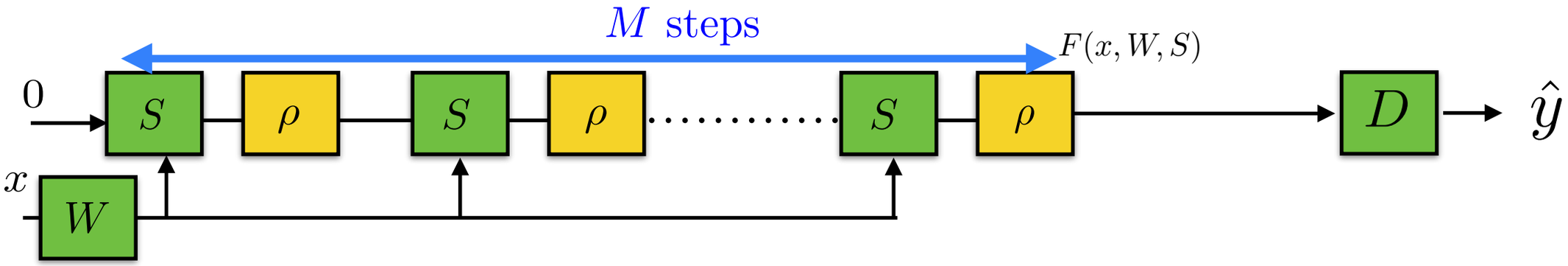
$$\min_{W,S} \frac{1}{n} \sum_{i \leq n} \|\Phi(x_i) - F(x_i, W, S)\|^2$$



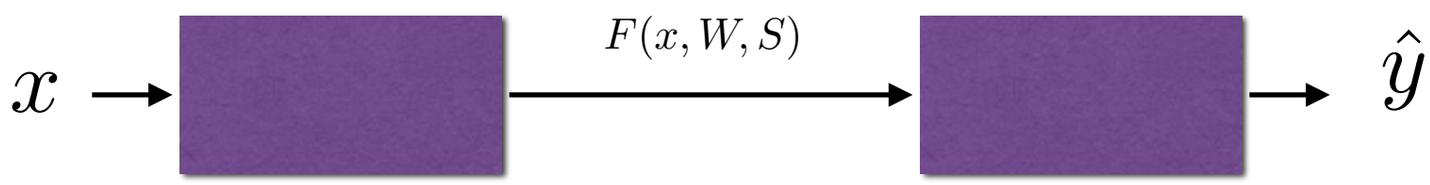
- LISTA adapts to the data distribution and produces much faster approximate sparse codes.

From supervised sparse coding to DNN

- The fast approximation of a sparse code can be plugged-in in a supervised regression or classification task:



$$\min_{\Theta, W} \sum_i \ell(y_i, DF(x_i, \Theta)) + \lambda \|F(x_i, \Theta)\|_1$$



$$\min_{\Theta, \Theta'} \sum_i \ell(y_i, G(F(x_i, \Theta), \Theta')) + \lambda \|F(x_i, \Theta)\|_1$$

From supervised sparse coding to DNN

- The fast approximation of a sparse code can be plugged-in in a supervised regression or classification task.

- For example, [*Sprechmann, Bronstein & Sapiro,' l 2*] in speaker identification experiments using non-negative matrix factorization:

Noise	**Exact**	**RNMF Encoders**	
		(***Supervised***)	(***Discriminative***)
street	0.86	0.91	0.91
restaurant	0.91	0.89	0.90
car	0.90	0.91	0.96
exhibition	0.93	0.91	0.95
train	0.93	0.88	0.96
airport	0.92	0.85	0.98
average	0.91	0.89	0.94

Fisher Kernels

- Recall the generic kernel method for (binary) classification:

$$\hat{y}(x) = \text{sign}\left(\sum_i y_i \lambda_i K(x, x_i)\right)$$

$\{(x_i, y_i)\}_i$: labeled training examples

λ_i: Lagrange multipliers associated to the loss

$K(x, y)$: similarity kernel

Fisher Kernels

- Recall the generic kernel method for (binary) classification:

$$\hat{y}(x) = \text{sign}\left(\sum_i y_i \lambda_i K(x, x_i)\right)$$

$\{(x_i, y_i)\}_i$: labeled training examples

λ_i: Lagrange multipliers associated to the loss

$K(x, y)$: similarity kernel

- Important challenge: how to choose the kernel?

Fisher Kernels

- Key idea: derive the kernel function from a generative probability model of the data $p(x \mid \theta)$.

Fisher Kernels

- Key idea: derive the kernel function from a generative probability model of the data $p(x \mid \theta)$.
- The *Fisher score* is defined as

$$U_x = \nabla_\theta \log p(x \mid \theta)$$

  - It measures how the parameter vector contributes to generating *x*.

Fisher Kernels

- Key idea: derive the kernel function from a generative probability model of the data $p(x \mid \theta)$.
- The *Fisher score* is defined as

$$U_x = \nabla_\theta \log p(x \mid \theta)$$

  - It measures how the parameter vector contributes to generating *x*.

- A natural metric in that probability space is given by the *Fisher Information matrix*:

$$I = \mathbb{E}_{x \sim p(x|\theta)}(U_x U_x^T)$$

Fisher Vectors

- It results in the Fisher Kernel

$$K(x, y) = U_x^T I^{-1} U_y$$

Fisher Vectors

- It results in the Fisher Kernel

$$K(x,y) = U_x^T I^{-1} U_y$$

- If I_θ is symmetric and positive definite, it admits a Cholesky decomposition $I_\theta = L_\theta^T L_\theta$ and thus

$$K(x,y) = \langle \tilde{U}_{x,\theta}, \tilde{U}_{y,\theta} \rangle \quad , \; \tilde{U}_{x,\theta} = L_\theta^{-1} U_x$$

- How to apply it to image classification/retrieval?

Fisher Vectors

- Let $X = \{x_l \; ; \; l = 1 \dots L\}$ be a collection of L *local* descriptors extracted from a single image.

Fisher Vectors

- Let $X = \{x_l \; ; \; l = 1 \dots L\}$ be a collection of L *local* descriptors extracted from a single image.
- Consider a Gaussian Mixture Model (GMM) as generative model:

$$p(x \mid \theta) = \sum_{k \leq K} w_k f(x; \mu_k, \Sigma_k)$$

w_k: mixture weights
$f(x; \mu, \Sigma)$: Multivariate Gaussian density with mean μ and covariance Σ.

Fisher Vectors

- Let $X = \{x_l \ ; \ l = 1 \dots L\}$ be a collection of L *local* descriptors extracted from a single image.
- Consider a Gaussian Mixture Model (GMM) as generative model:

$$p(x \mid \theta) = \sum_{k \leq K} w_k f(x; \mu_k, \Sigma_k)$$

w_k: mixture weights
$f(x; \mu, \Sigma)$: Multivariate Gaussian density with mean μ and covariance Σ.

- If we assume that the x_l are generated independently by $p(x \mid \theta)$ we have that

$$U_X = \frac{1}{L} \sum_l \nabla_\theta \log p(x_l \mid \theta)$$

Fisher Vectors

- By slightly simplifying the model, we obtain normalized Fisher vectors of the form:

$$\tilde{U}_X = (\tilde{U}_{X,k,\mu}, \tilde{U}_{X,k,\sigma})_{k \leq K}$$

$$\tilde{U}_{X,k,\mu} = \frac{1}{L\sqrt{w_k}} \sum_{l \leq L} \gamma_l(k) \frac{x_l - \mu_k}{\sigma_k} \ , \ \tilde{U}_{X,k,\sigma} = \frac{1}{L\sqrt{2w_k}} \sum_{l \leq L} \gamma_l(k) \left(\frac{(x_l - \mu_k)^2}{\sigma_k^2} - 1 \right)$$

$$\gamma_l(k) = \frac{w_k f(x_l; \mu_k, \Sigma_k)}{\sum_{k' \leq K} w_{k'} f(x_l; \mu_{k'}, \Sigma_{k'})} \quad \text{(soft assignment of descriptor } x_l \text{ to Gaussian } k\text{)}$$

["Improving the Fisher Kernel for Large-Scale Image Classification", Perronnin et al,'10]

Fisher Vectors

- By slightly simplifying the model, we obtain normalized Fisher vectors of the form:

$$\tilde{U}_X = (\tilde{U}_{X,k,\mu}, \tilde{U}_{X,k,\sigma})_{k \leq K}$$

$$\tilde{U}_{X,k,\mu} = \frac{1}{L\sqrt{w_k}} \sum_{l \leq L} \gamma_l(k) \frac{x_l - \mu_k}{\sigma_k} \ , \ \tilde{U}_{X,k,\sigma} = \frac{1}{L\sqrt{2w_k}} \sum_{l \leq L} \gamma_l(k) \left(\frac{(x_l - \mu_k)^2}{\sigma_k^2} - 1 \right)$$

$$\gamma_l(k) = \frac{w_k f(x_l; \mu_k, \Sigma_k)}{\sum_{k' \leq K} w_{k'} f(x_l; \mu_{k'}, \Sigma_{k'})} \quad \text{(soft assignment of descriptor } x_l \text{ to Gaussian } k\text{)}$$

- We aggregate over the image not only counts of visual words, but also first and second order statistics within each cluster.

["Improving the Fisher Kernel for Large-Scale Image Classification", Perronnin et al,'10]

Fisher Vectors and VLAD

- Typically, one considers local descriptors such as SIFT (Scale Invariant Feature Transform) of HoG (Histogram of Oriented Gradients).

Fisher Vectors and VLAD

- Typically, one considers local descriptors such as SIFT (Scale Invariant Feature Transform) of HoG (Histogram of Oriented Gradients)

- By properly normalizing the Fisher vectors and improving the spatial aggregation, State-of-the-art results on Image Classification, Detection and Retrieval before CNNs [Perronnin et al, '10].

Fisher Vectors and VLAD

- Typically, one considers local descriptors such as SIFT (Scale Invariant Feature Transform) of HoG (Histogram of Oriented Gradients)

- By properly normalizing the Fisher vectors and improving the spatial aggregation, State-of-the-art results on Image Classification, Detection and Retrieval before CNNs [Perronnin et al, '10].

- VLAD (Vector of Locally Aggregated Descriptors) [Jegou et al, '10] considers only first order statistics and hard assignments.

From VLAD to CNNs

- The VLAD representation thus becomes

$$\Phi(x)(\lambda, k) = \sum_{l \leq L} a_k(\phi_l(x))(\phi_l(x)(\lambda) - c_k(\lambda))$$

ϕ_l: local descriptor at location l
$a_k(\phi_l(x)) = 1$ if c_k is closest to $\phi_l(x)$, 0 otherwise

- How to relate this operation with what a CNN can do?

NetVLAD [Arandjelovic et al,'15]

- Replace the hard cluster assignments with a softmax assignment of the form

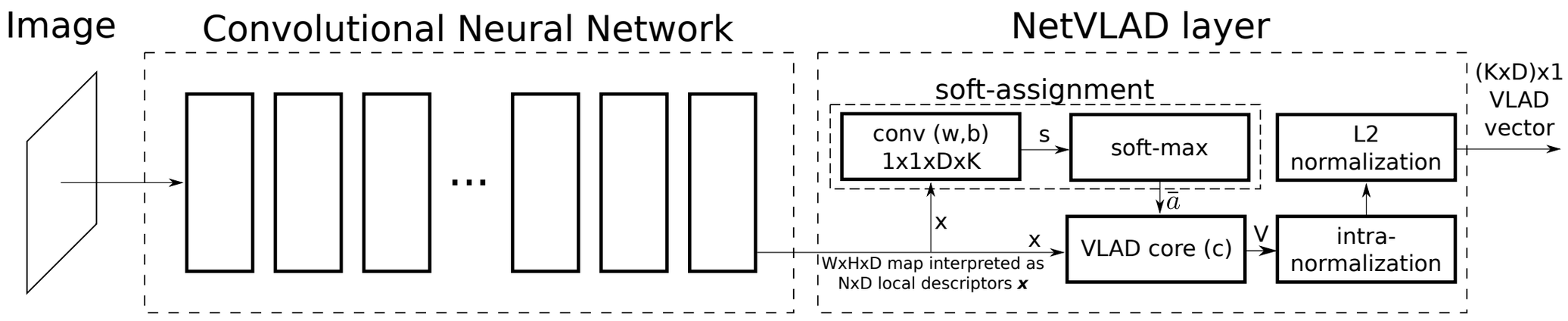
$$\overline{a}_k(\phi_l(x)) = \frac{e^{-\alpha\|\phi_l(x)-c_k\|^2}}{\sum_{k'} e^{-\alpha\|\phi_l(x)-c_{k'}\|^2}}$$

NetVLAD [Arandjelovic et al,'15]

- Replace the hard cluster assignments with a softmax assignment of the form

$$\overline{a}_k(\phi_l(x)) = \frac{e^{-\alpha\|\phi_l(x)-c_k\|^2}}{\sum_{k'} e^{-\alpha\|\phi_l(x)-c_{k'}\|^2}}$$

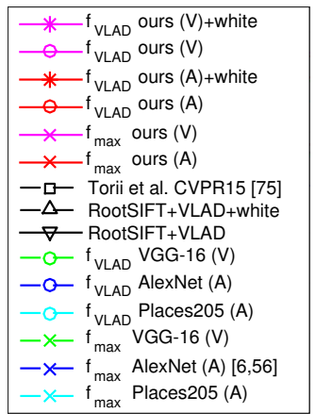
- Replace the local descriptors $\phi_l(x)$ by a few convolutional layers and make the centers c_k trainable.



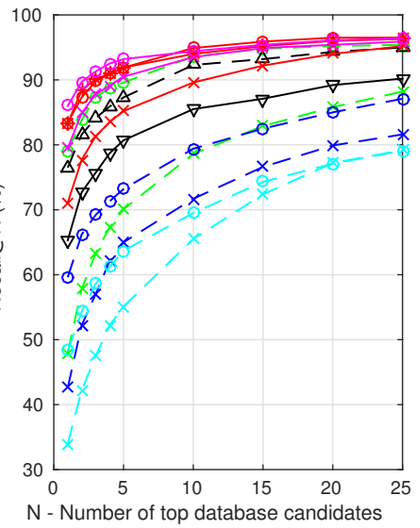
NetVLAD [Arandjelovic et al,'15]

- Examples of Retrieval Results (Tokyo dataset).

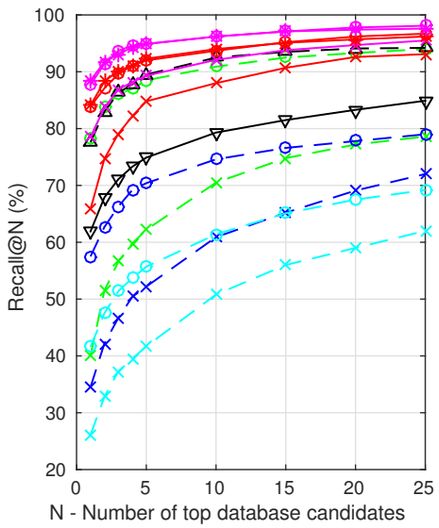
NetVLAD [Arandjelovic et al,'15]



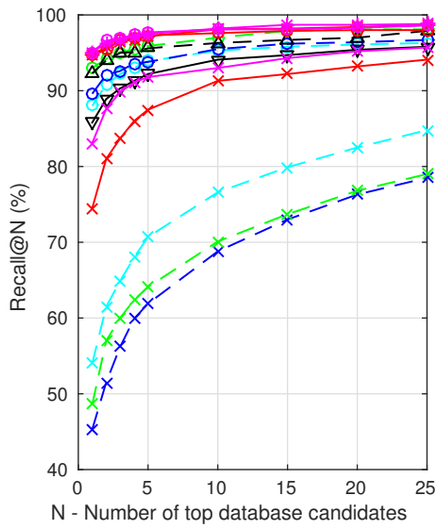
(a) Legend



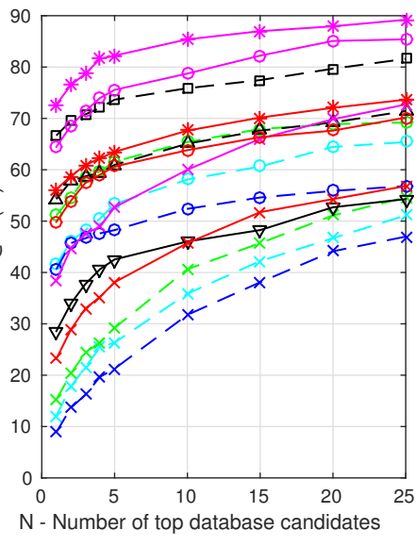
(b) Pitts30k-test



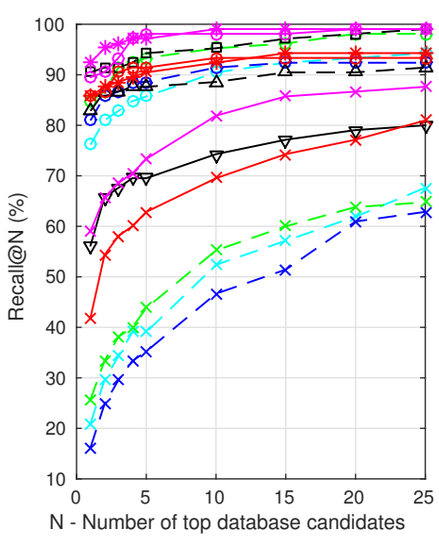
(c) Pitts250k-test



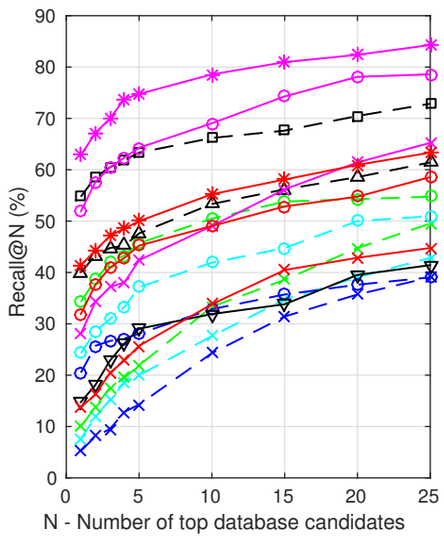
(d) TokyoTM-val



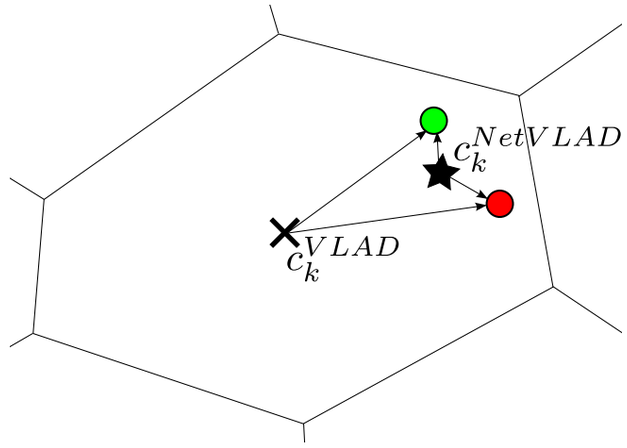
(e) Tokyo 24/7 all queries



(f) Tokyo 24/7 daytime



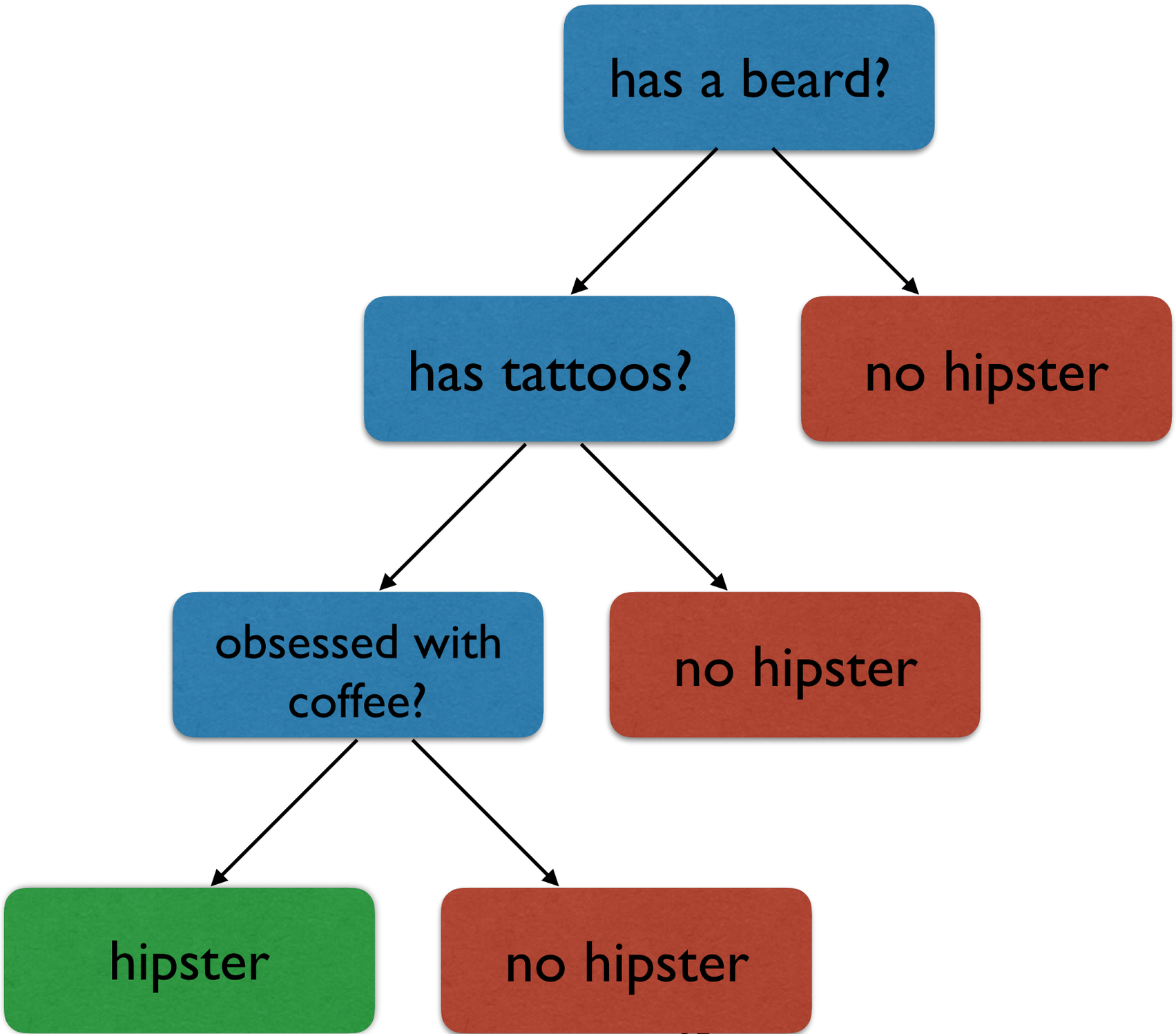
(g) Tokyo 24/7 sunset/night



- More general pooling mechanism
- Training end-to-end again brings substantial gains.

Decision Trees

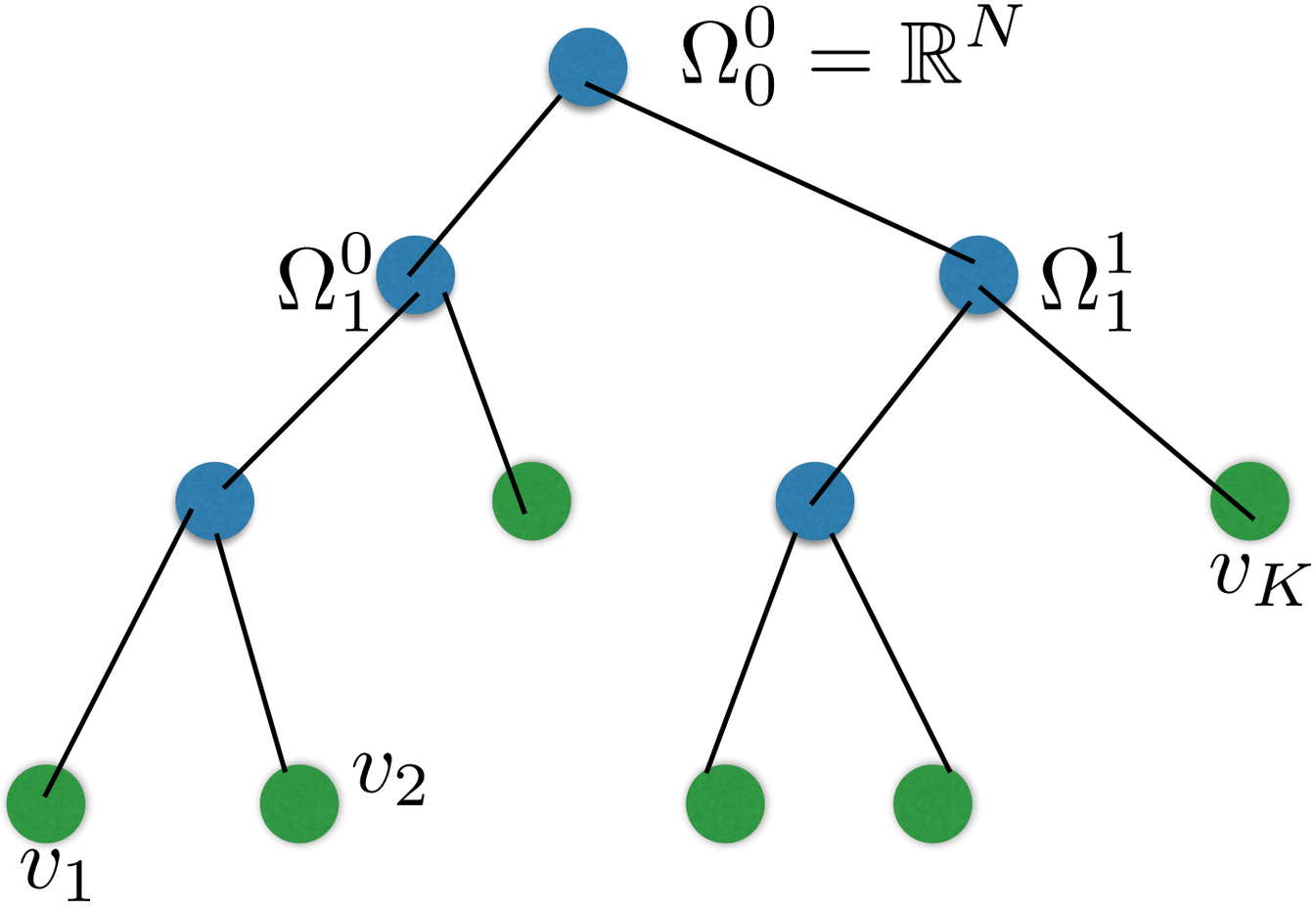
- Typical 20-question game:



Decision Trees

- Let $x = \{(x_1, y_1), \ldots, (x_T, y_T)\}$ be the input data, with $x_i \in \mathbb{R}^N$.
- Each node of the three selects a variable and splits using a threshold.

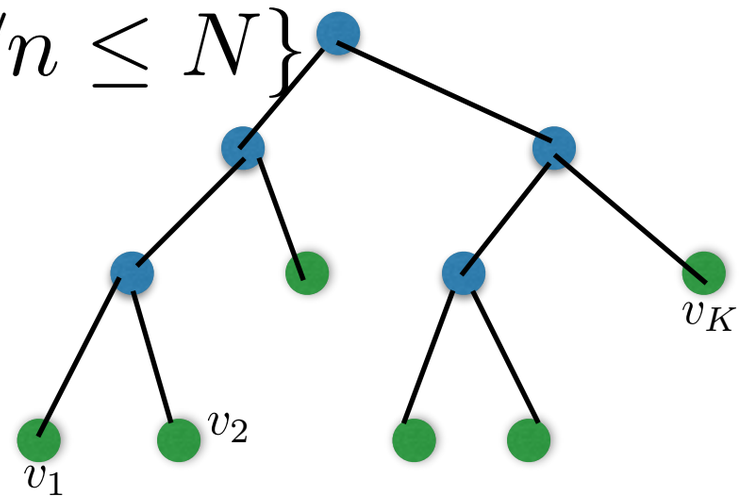
$$\Omega_i^j \subset \mathbb{R}^N \rightarrow (\Omega_{i+1}^l, \Omega_{i+1}^{l+1})$$
$$\Omega_i^j = \Omega_{i+1}^l \cup \Omega_{i+1}^{l+1}, \ \emptyset = \Omega_{i+1}^l \cap \Omega_{i+1}^{l+1}$$



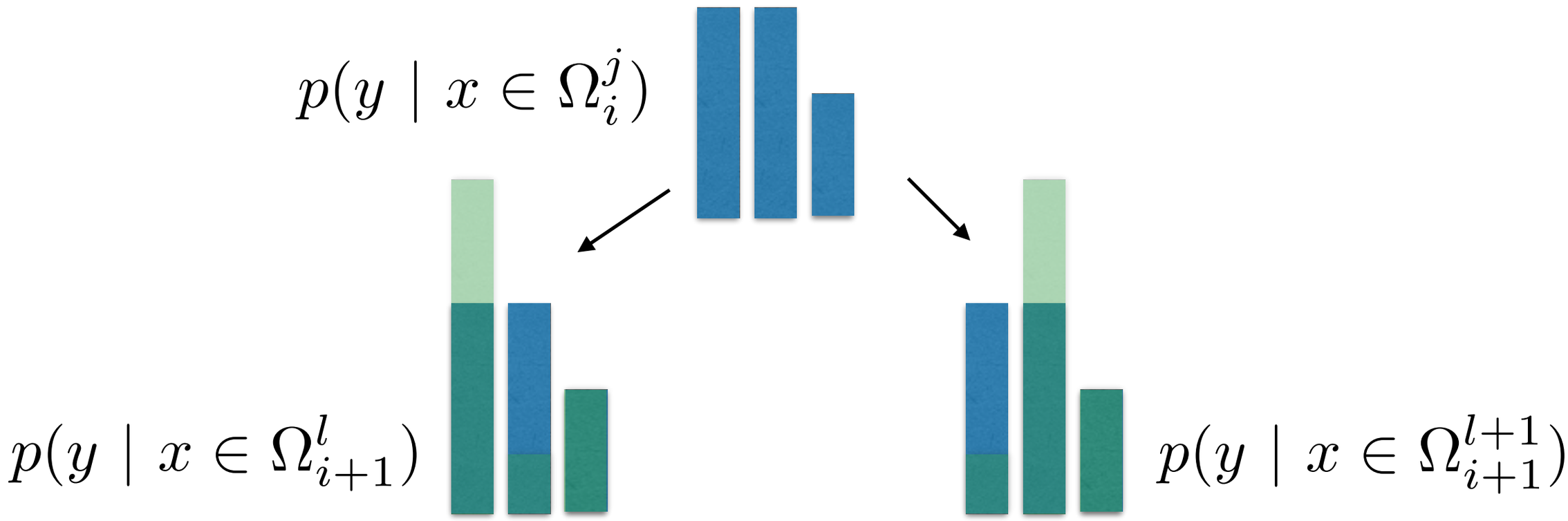
Decision Trees

The leaves $\{v_k\}$ of the tree define a partition of the input space into cubic sections:

$$\Omega_\infty^k = \{x \in \mathbb{R}^N \, ; \, \alpha_{k,n} \leq x_n \leq \beta_{k,n} \; \forall n \leq N\}$$



- Each split optimizes the entropy in the label distribution:



Random Forests

- A decision tree can capture interactions between different variables, but it is very noisy (ie unstable).
- Evaluation and training are extremely efficient.

Random Forests

- A decision tree can capture interactions between different variables, but it is very noisy (ie unstable).
- Evaluation and training are extremely efficient.
- By appropriately introducing *randomization*, we can construct an ensemble of random trees: the so-called *random forests*.

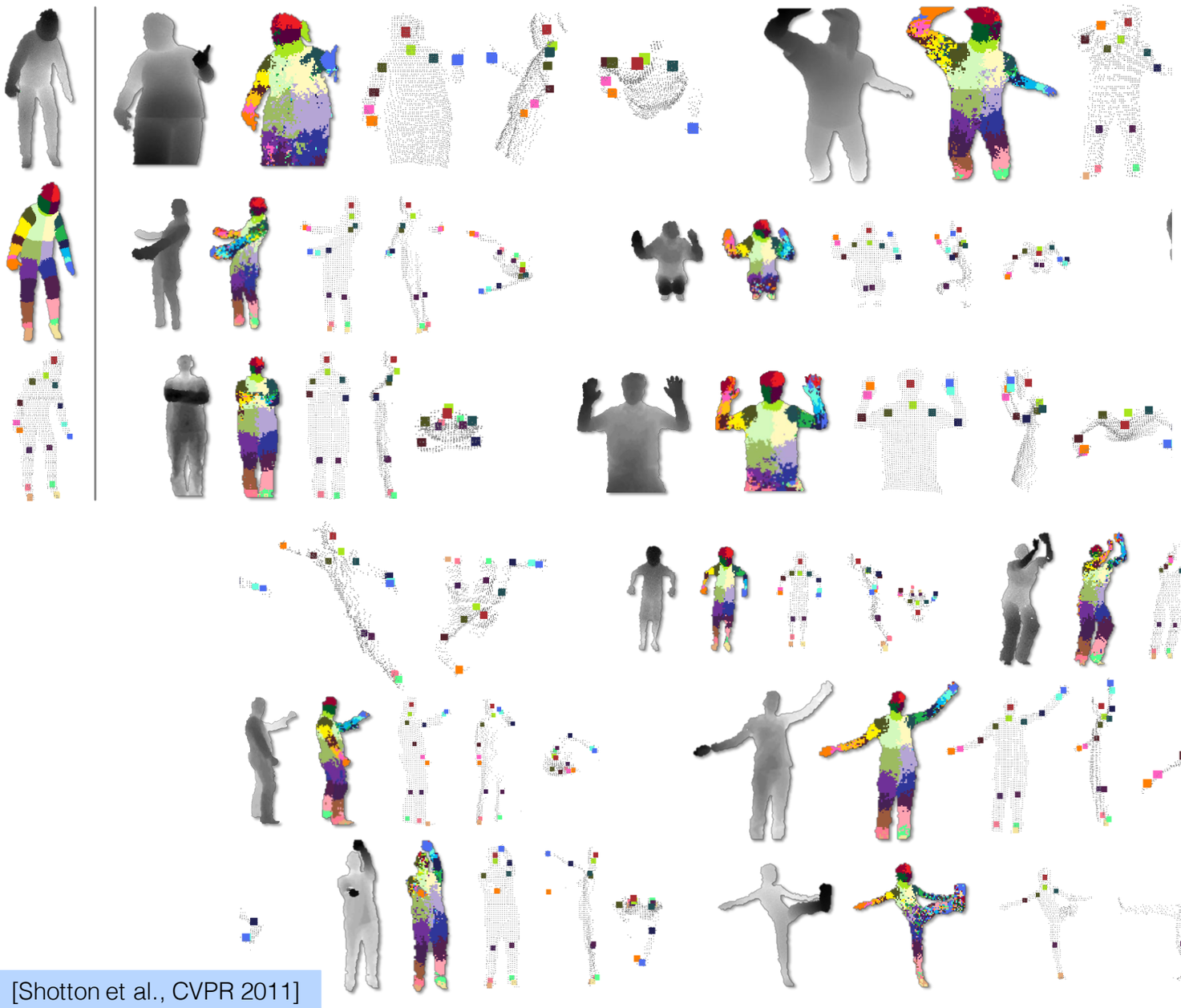
Random Forests

- A decision tree can capture interactions between different variables, but it is very noisy (ie unstable).
- Evaluation and training are extremely efficient.
- By appropriately introducing *randomization*, we can construct an ensemble of random trees: the so-called *random forests*.

- We draw bootstrapped samples of the training set, and each split in the tree is calculated *only on a small random subset of variables (*typically of size $O(\sqrt{N})$).
- The prediction is the aggregate prediction (ie voting) of each tree.

Random Forests

- Successful across a wide range of classification and regression problems.



[Shotton et al., CVPR 2011]

Real-Time Pose Estimation from Kinect measurements (CVPR'11)

LIRIS CNRS

(figure from Ch. Wolf slides)

Random Forests and CNNs

- Random Forests thus also consider piecewise linear regions of the input space.

- However the encoding of these regions is different from that of a deep ReLU network.

- Computationally more efficient
- No gradient descent training
- Less expressive

Random Forests and CNNs

- Random Forests thus also consider piecewise linear regions of the input space.

- However the encoding of these regions is different from that of a deep ReLU network.

- Computationally more efficient
- No gradient descent training
- Less expressive