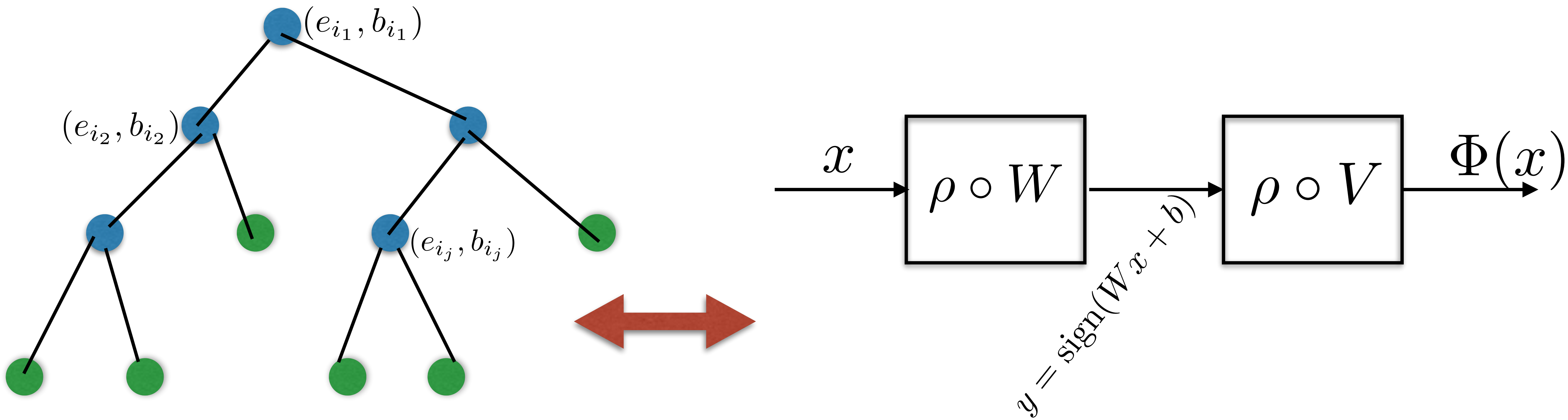
Stat 212b: Topics in Deep Learning
Lecture 10

Joan Bruna
UC Berkeley

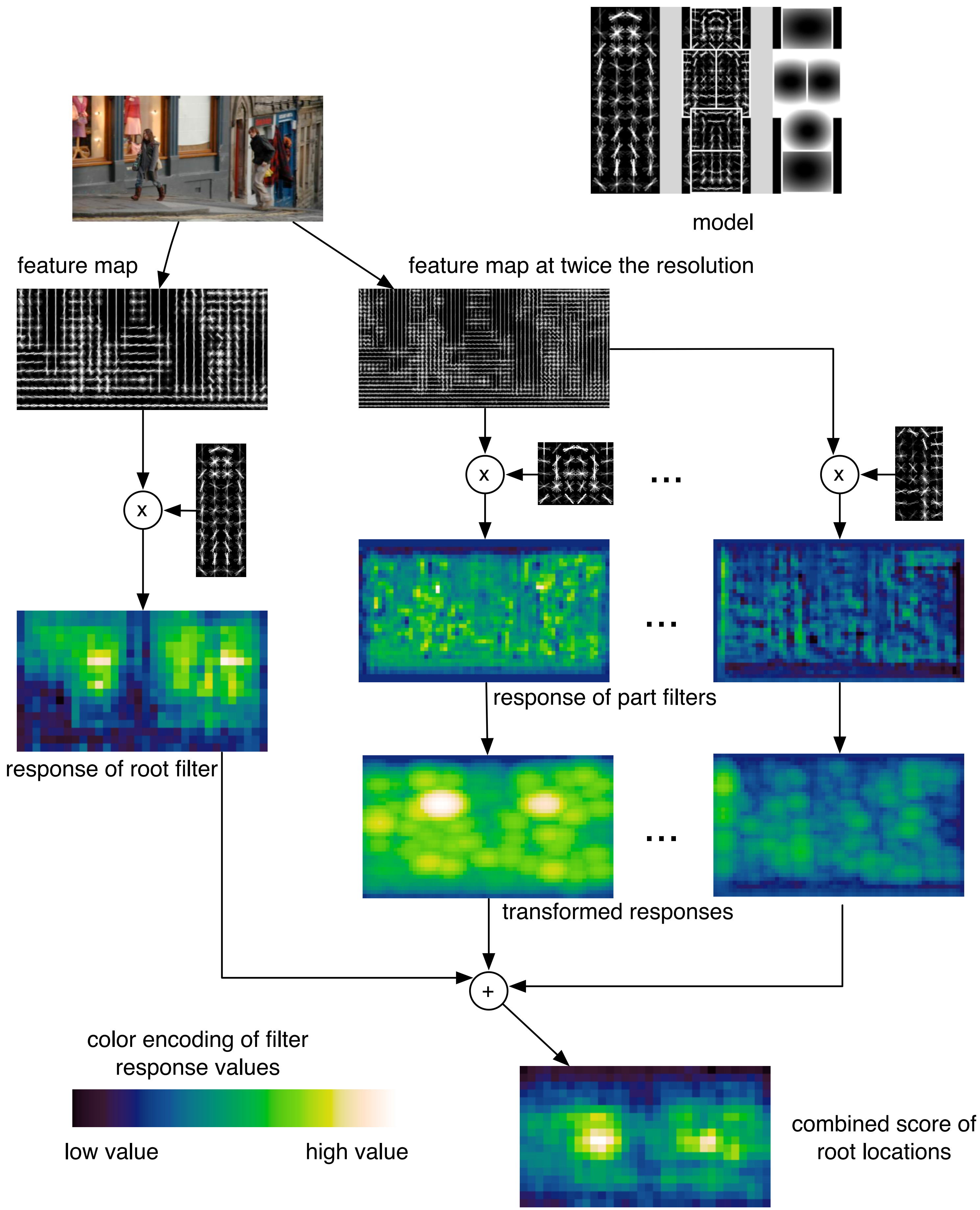
Berkeley
UNIVERSITY OF CALIFORNIA

Review: Embed RF into Deep Neural Networks?



- $\Phi(x) \in \mathbb{R}^L$ is a *one-hot* vector encoding $x \in \Omega^l_\infty$, $l \leq L$. $\left(x \in \Omega^l_\infty \Leftrightarrow \Phi(x)_l = 1, \Phi(x)_k = 0\,, k \neq l\right)$
- The Random Forest is obtained with an ensemble of two-layer networks.
- Training is radically different: greedy in RF versus gradient descent in Deep Learning.

Review: Deformable Parts Model

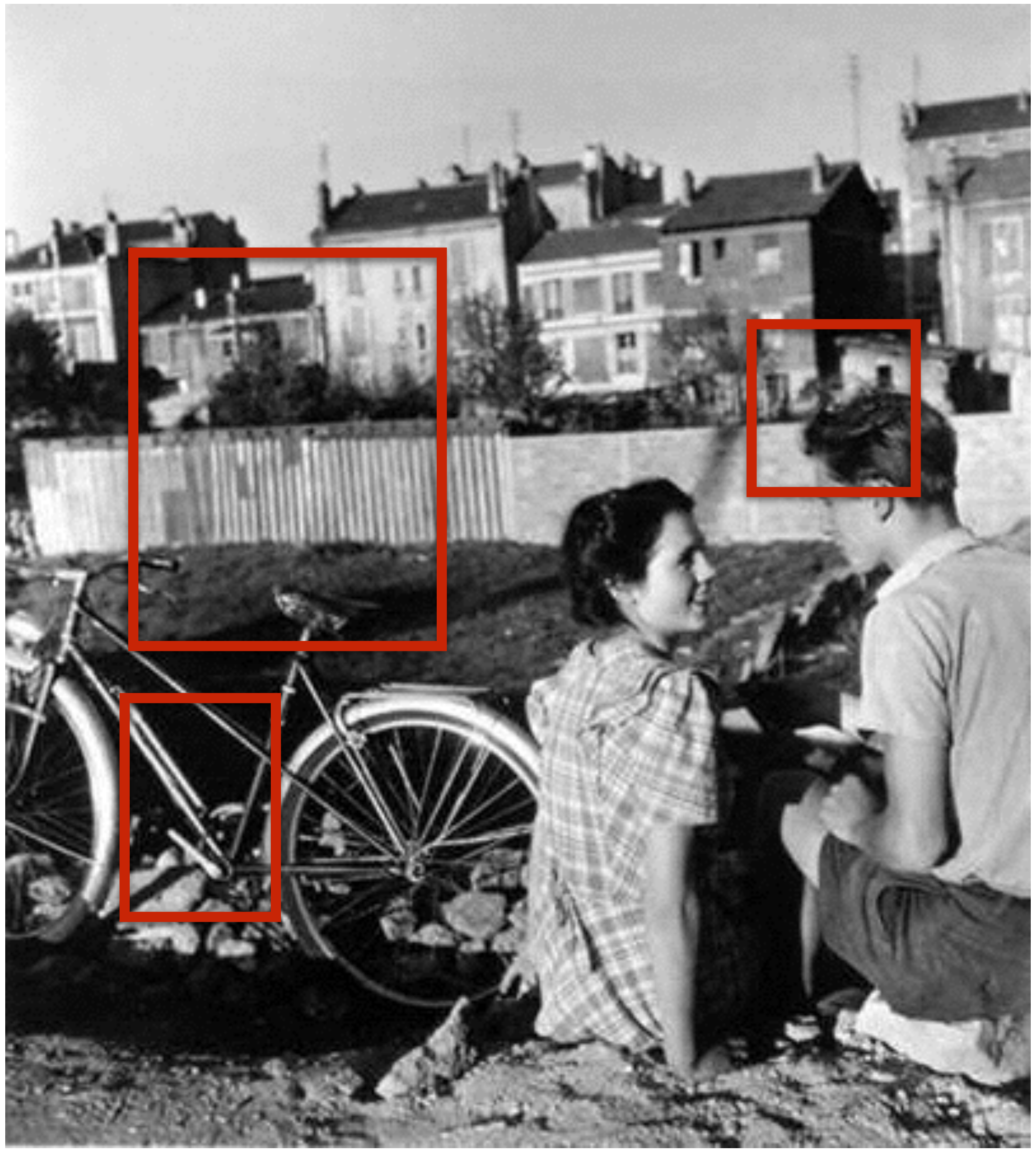


"Object Detection with discriminatively trained Deformable Parts Model", Felzenszwalb, Girshick et al.'10

Provides a Generative Model that is compatible with the Deep Convolutional Architecture.

Can it scale to model high-dimensional variability present in natural images?

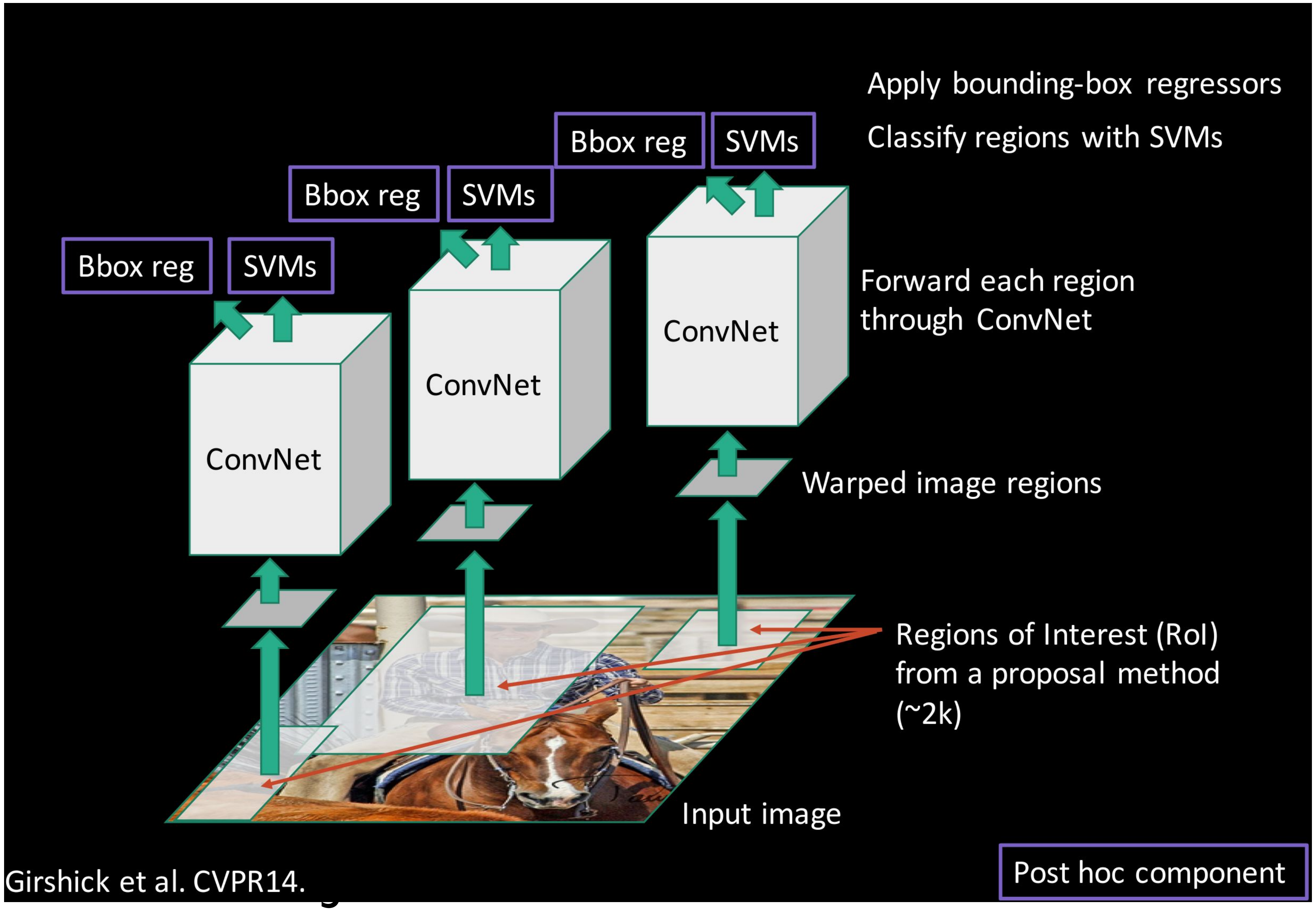
Review: Region-based CNN (R-CNN)



- Suppose that for each bounding box we ask: is there a {house, bicycle, dog, man, …, *none*} ?
- This is standard object classification.

Review: R-CNN [R. Girshick et al, 14-15]

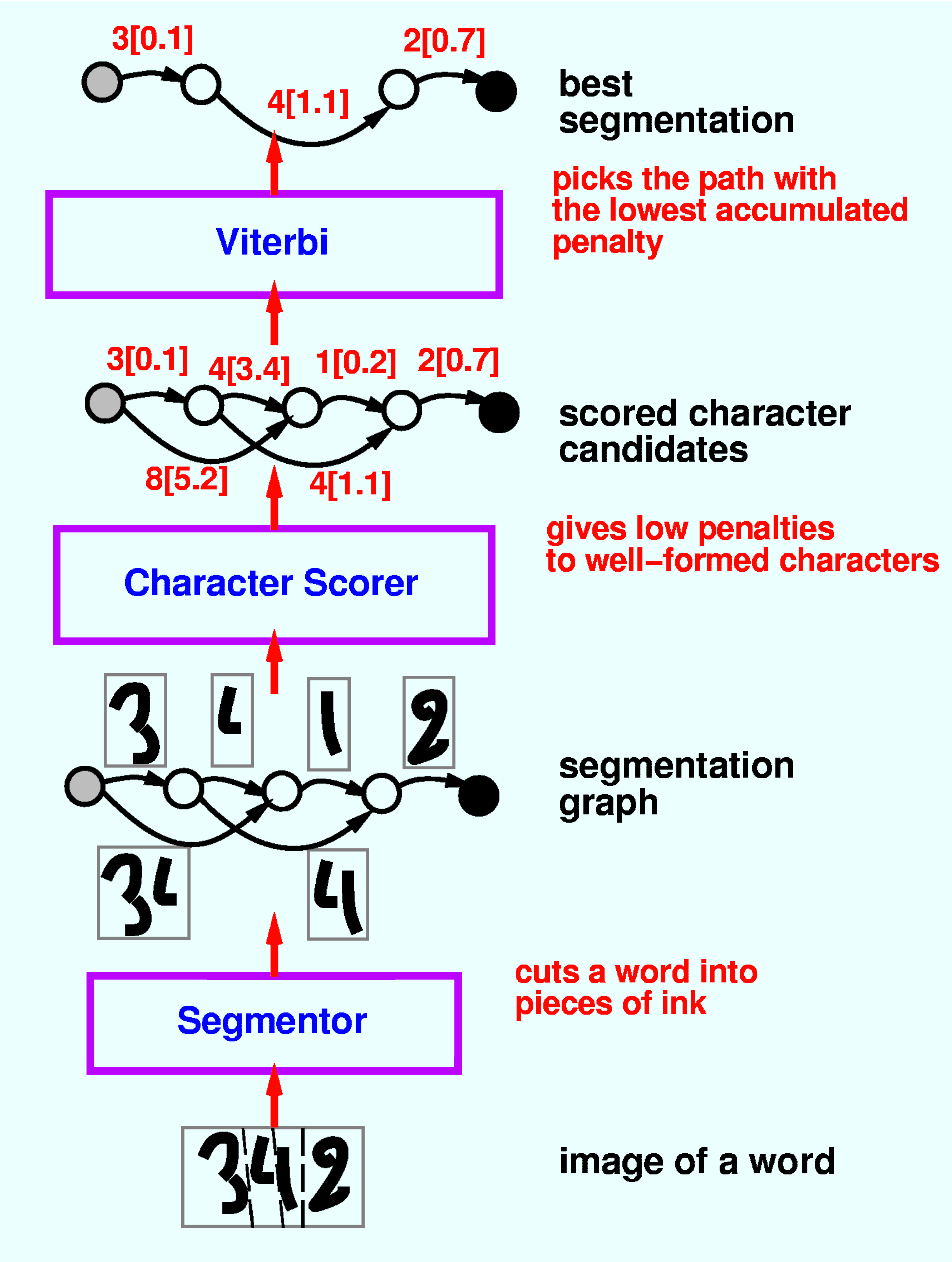
- Rather than testing every possible rectangular region, we rely on a Region Proposal algorithm (which can also be done by a CNN).
- Each proposal region is warped and analyzed with another CNN.



Review: Graph Transformer Network

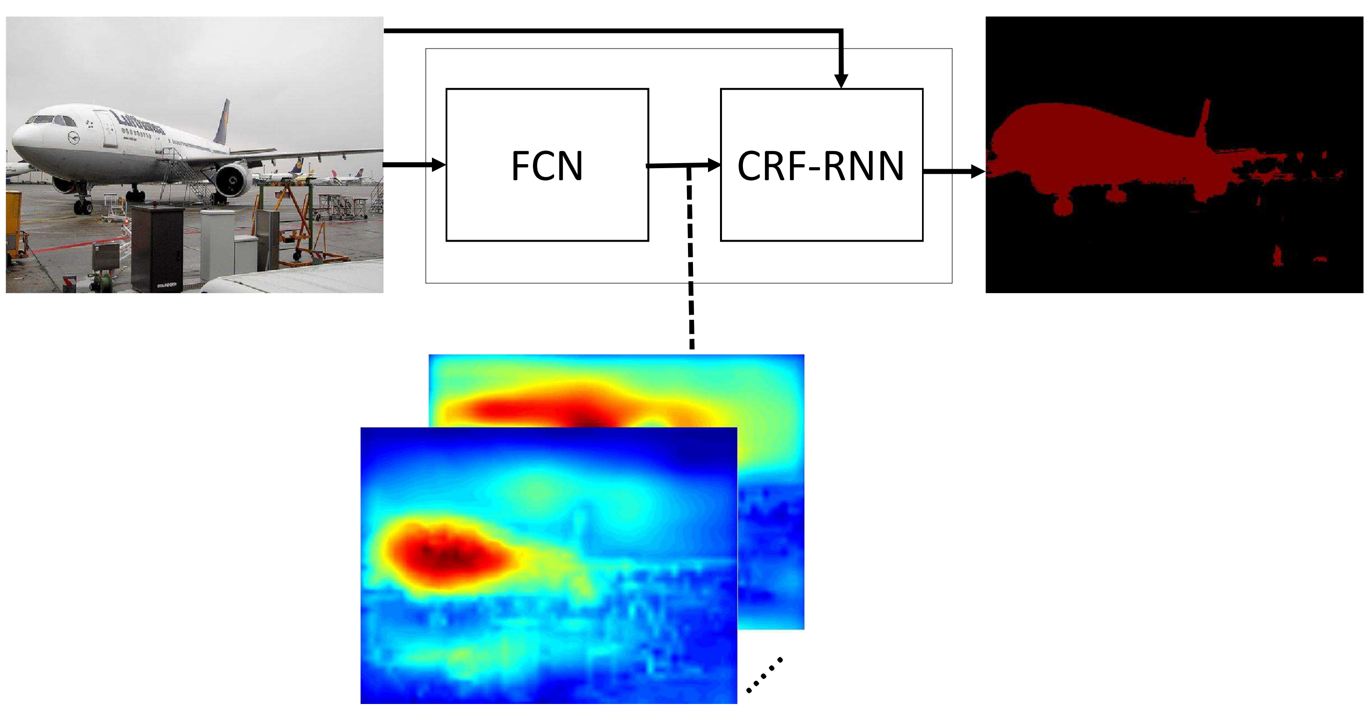
- [Bottou, Bengio & LeCun, '97]
- Graphical model over possible "segmentations" of handwritten characters

- Used commercially to read ~10% checks in the US (1996).



Review: CRFs as Convolutional Neural Networks

- [Zheng et al,'15] approximate the mean-field message passing iterations with CNN layers with shared parameters.
- The system can be efficiently trained end-to-end.



Algorithm 1 Mean-field in dense CRFs [27], broken down to common CNN operations.

$Q_i(l) \leftarrow \frac{1}{Z_i} \exp\left(U_i(l)\right)$ for all i ▷ Initialization

while not converged **do**

$\tilde{Q}_i^{(m)}(l) \leftarrow \sum_{j \neq i} k^{(m)}(\mathbf{f}_i, \mathbf{f}_j) Q_j(l)$ for all m ▷ Message Passing

$\check{Q}_i(l) \leftarrow \sum_m w^{(m)} \tilde{Q}_i^{(m)}(l)$ ▷ Weighting Filter Outputs

$\hat{Q}_i(l) \leftarrow \sum_{l' \in \mathcal{L}} \mu(l, l') \check{Q}_i(l)$ ▷ Compatibility Transform

$\breve{Q}_i(l) \leftarrow U_i(l) - \hat{Q}_i(l)$ ▷ Adding Unary Potentials

$Q_i \leftarrow \frac{1}{Z_i} \exp\left(\breve{Q}_i(l)\right)$ ▷ Normalizing

end while

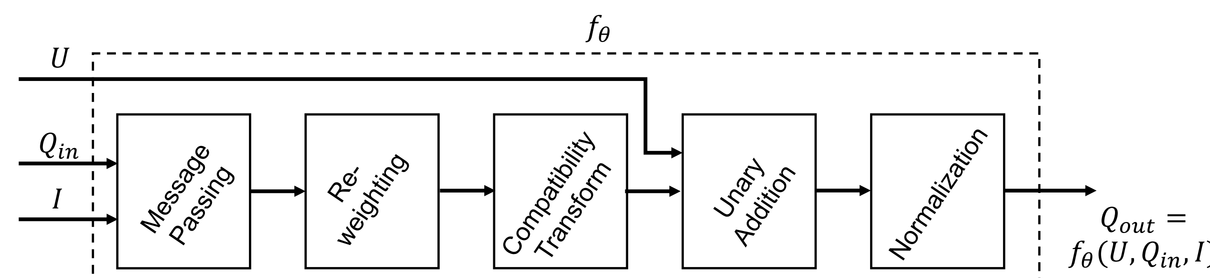


Figure 1. **A mean-field iteration as a CNN.** A single iteration of the mean-field algorithm can be modelled as a stack of common CNN layers.

Objectives

- Embeddings

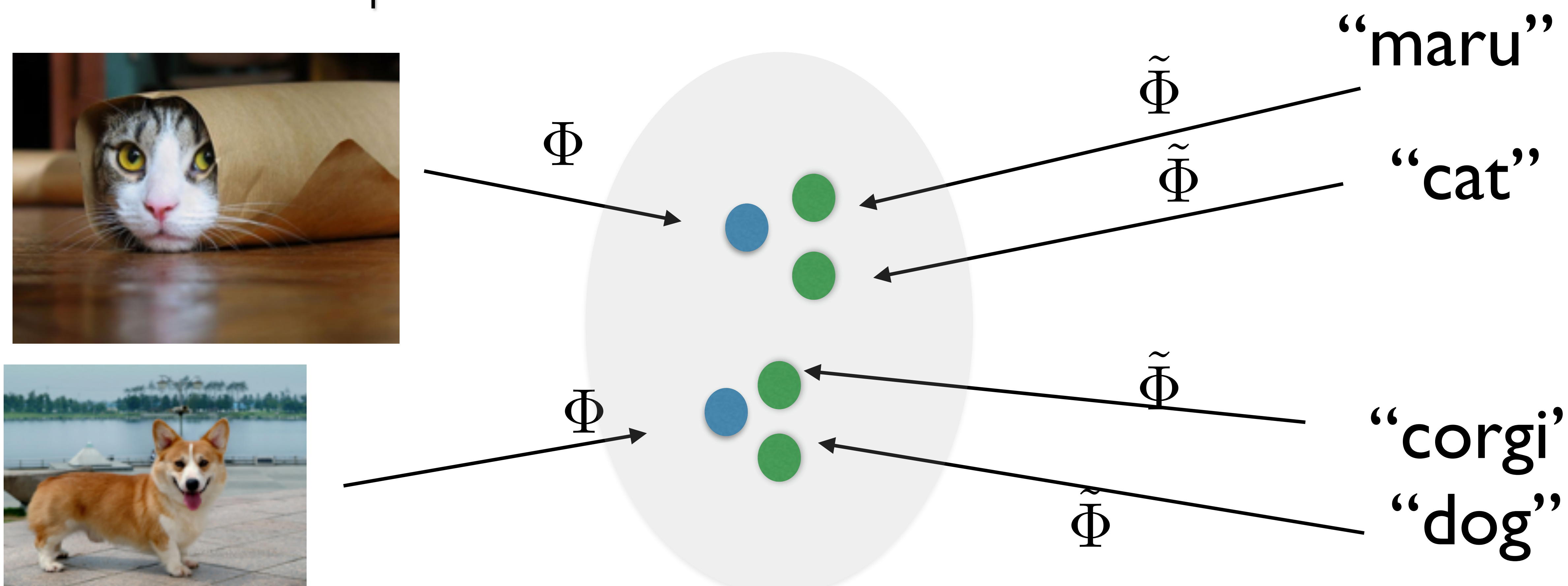
- Extensions to Non-Euclidean Domains
  - Locally Connected Networks
  - Spectral Networks
  - Spatial Transformer Networks

- Representations of Stationary Processes
  - Scattering Moments
  - Properties and Applications
  - Texture Synthesis
  - CNNs for Texture Representation.

Embeddings

- Q: Can we use a CNN to learn a metric $\|\Phi(x) - \Phi(x')\|$ with specific properties?

- Ex: metric compatible with object categories and/or transformations.
- Ex: metric compatible with a retrieval task:

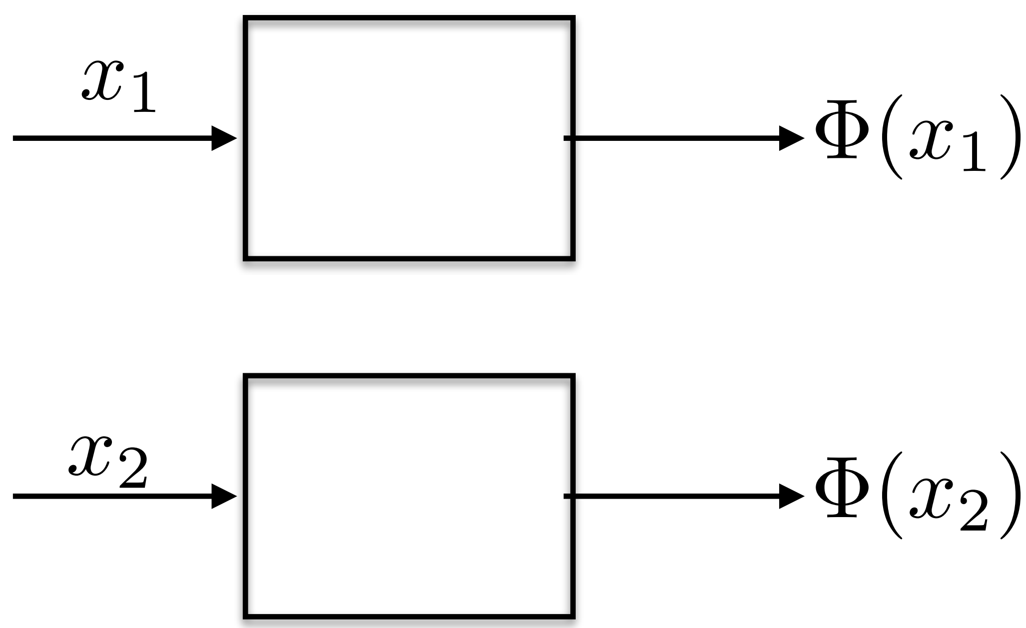


Embeddings with “Siamese” Architectures

- *positive* pairs $(x_1, x_2) \in \mathcal{X} \times \mathcal{X}$: $(x_1, x_2) \sim q_{pos}$
- *negative* pairs $(x_1, x_2) \in \mathcal{X} \times \mathcal{X}$: $(x_1, x_2) \sim q_{neg}$
- *Idea*: we want to push closer positive pairs and push farther negative pairs:

Hinge Embedding Loss:

$$\min_{\Phi} \mathbb{E}_{x \sim q_{pos}} \|\Phi(x_1) - \Phi(x_2)\|^2 + \lambda \mathbb{E}_{x \sim q_{neg}} \max(0, M - \|\Phi(x_1) - \Phi(x_2)\|^2)$$

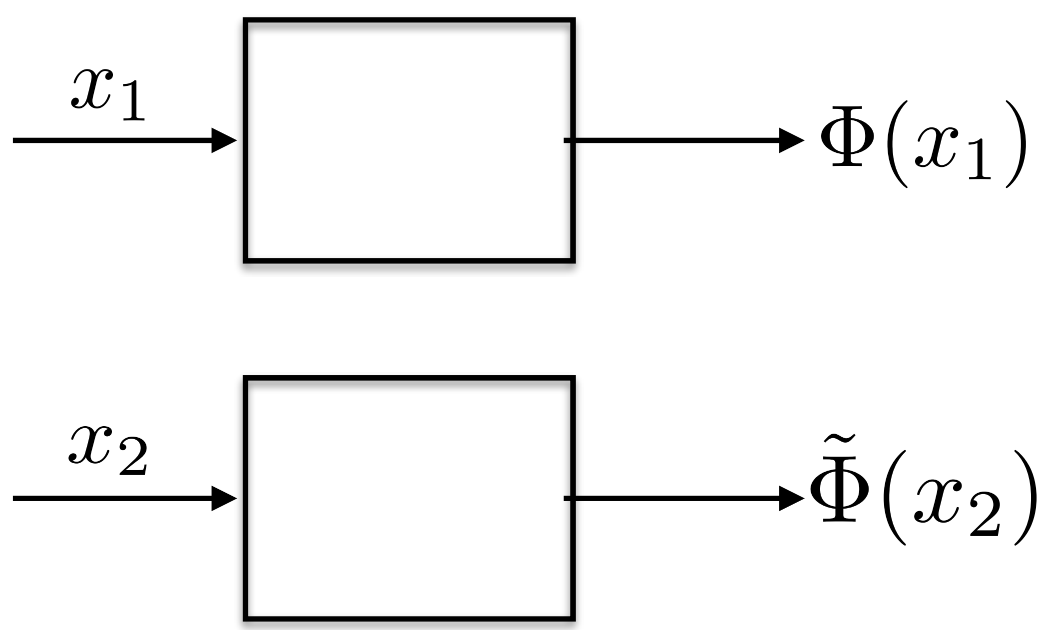


Embeddings with General Architectures

- *positive* pairs $(x_1, x_2) \in \mathcal{X} \times \mathcal{X}$: $(x_1, x_2) \sim q_{pos}$
- *negative* pairs $(x_1, x_2) \in \mathcal{X} \times \mathcal{X}$: $(x_1, x_2) \sim q_{neg}$
- *Idea*: we want to push closer positive pairs and push farther negative pairs:

Hinge Embedding Loss:

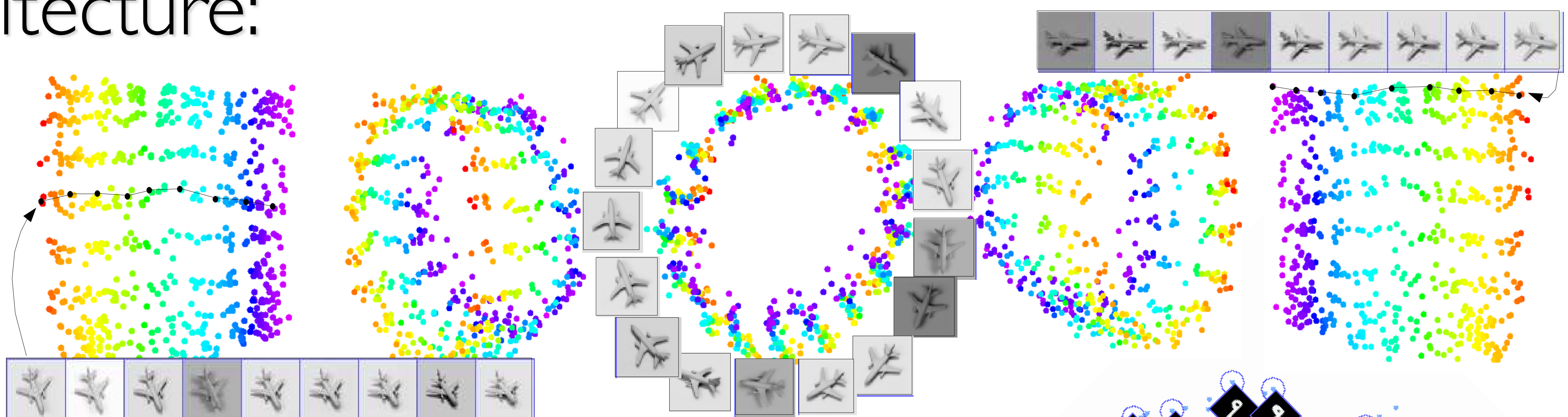
$$\min_{\Phi, \tilde{\Phi}} \mathbb{E}_{x \sim q_{pos}} \|\Phi(x_1) - \tilde{\Phi}(x_2)\|^2 + \lambda \mathbb{E}_{x \sim q_{neg}} \max(0, M - \|\Phi(x_1) - \tilde{\Phi}(x_2)\|^2)$$

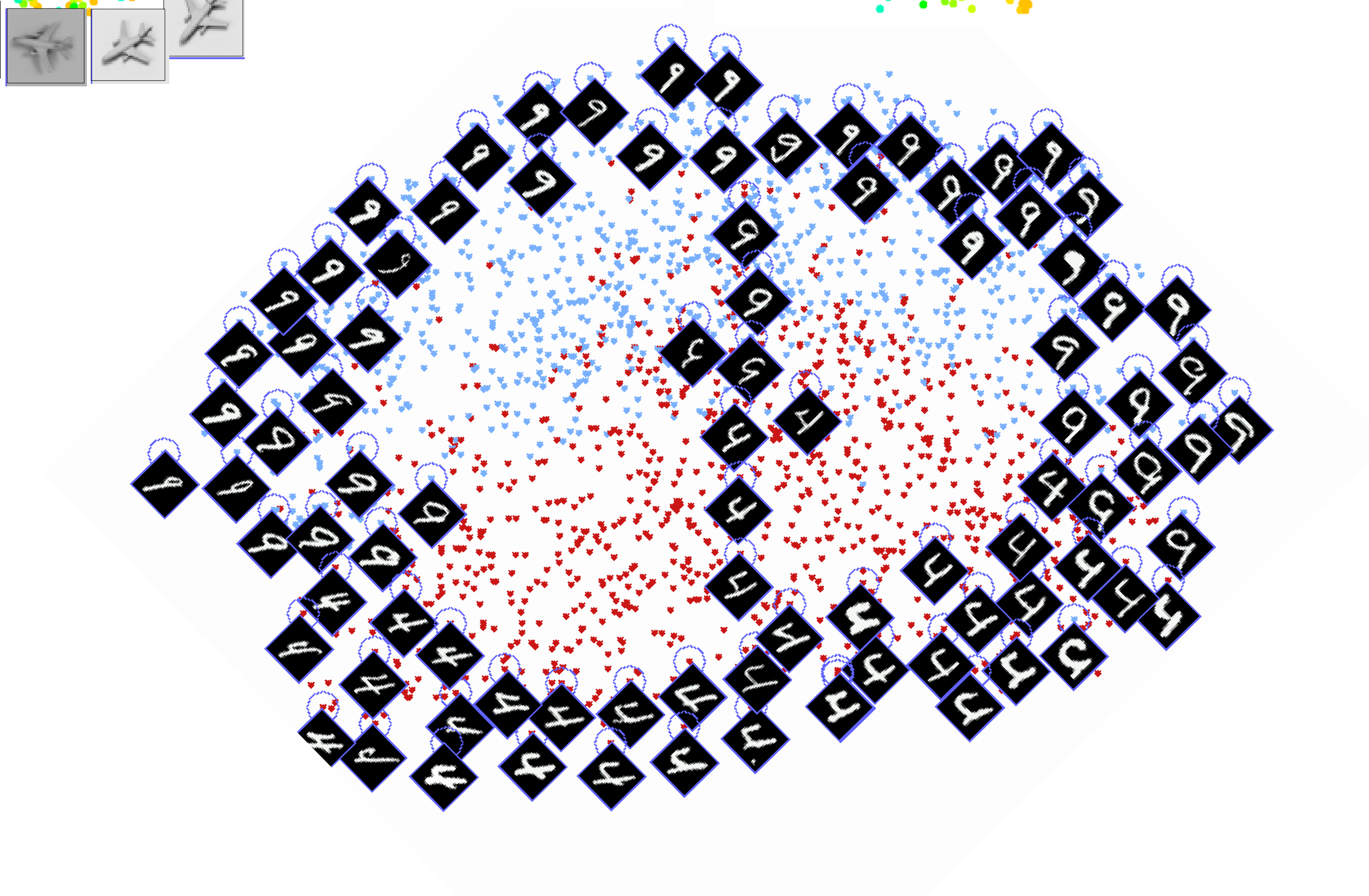


- The "contrastive" term can be replaced by other losses.

Dimensionality Reduction and Embedding

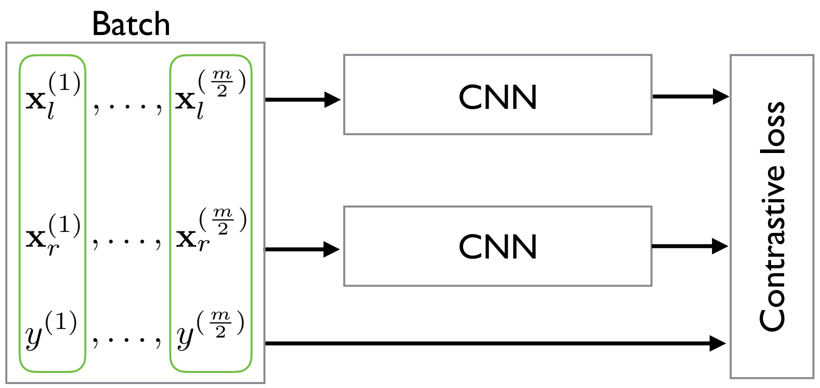
- [DrLiM, Hasdell et al, '06] considered a setup where $\Phi(x)$ is a low-dimensional embedding using a siamese CNN architecture:



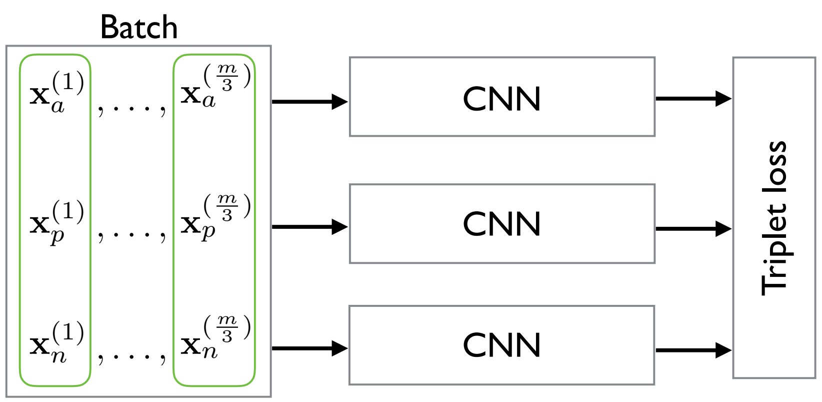


Semantic Embedding and metric learning

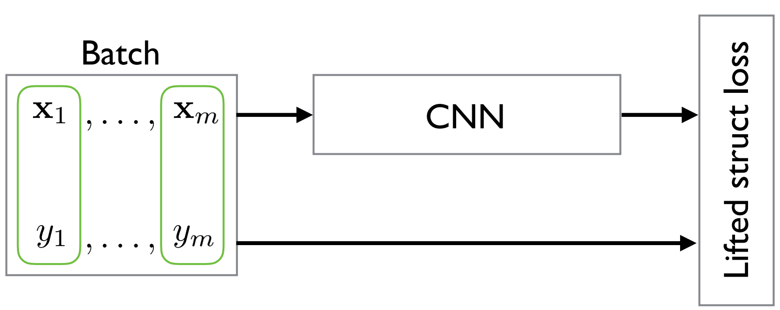
- Given labeled data, one may learn a metric of the form $d(x, x') = \|\Phi(x) - \Phi(x')\|$ that is compatible with labels.



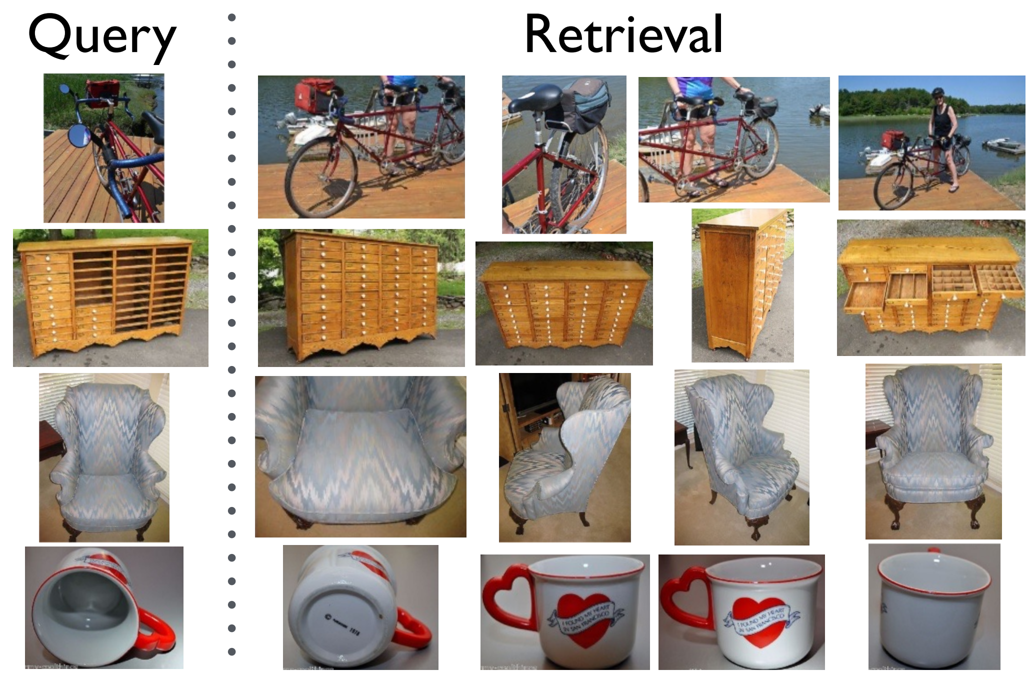
(a) Training network with contrastive embedding [14]



(b) Training network with triplet embedding [39, 31]



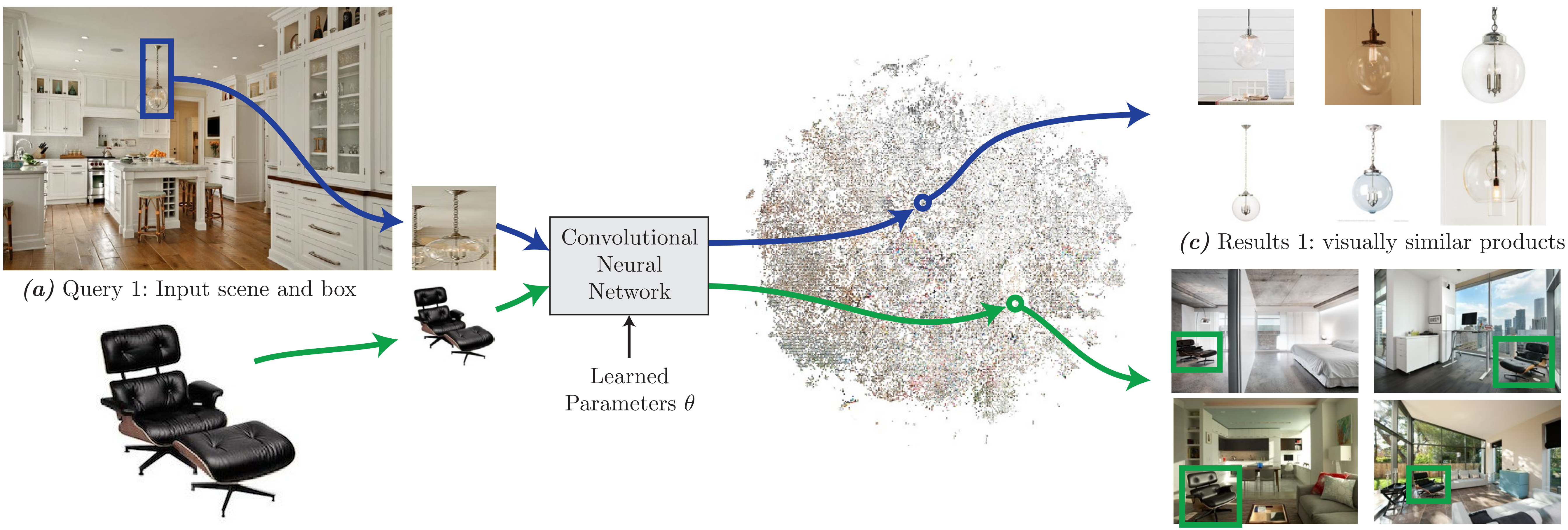
(c) Training network with lifted structure embedding



"Deep Metric Learning via Lifted Structured Feature Embedding", Oh Song et al.'15

Semantic Embedding and metric learning

- Given labeled data, one may learn a metric of the form $d(x, x') = \|\Phi(x) - \Phi(x')\|$ that is compatible with labels.



(a) Query 1: Input scene and box

(a) Query 2: Product

(b) Project into 256D embedding

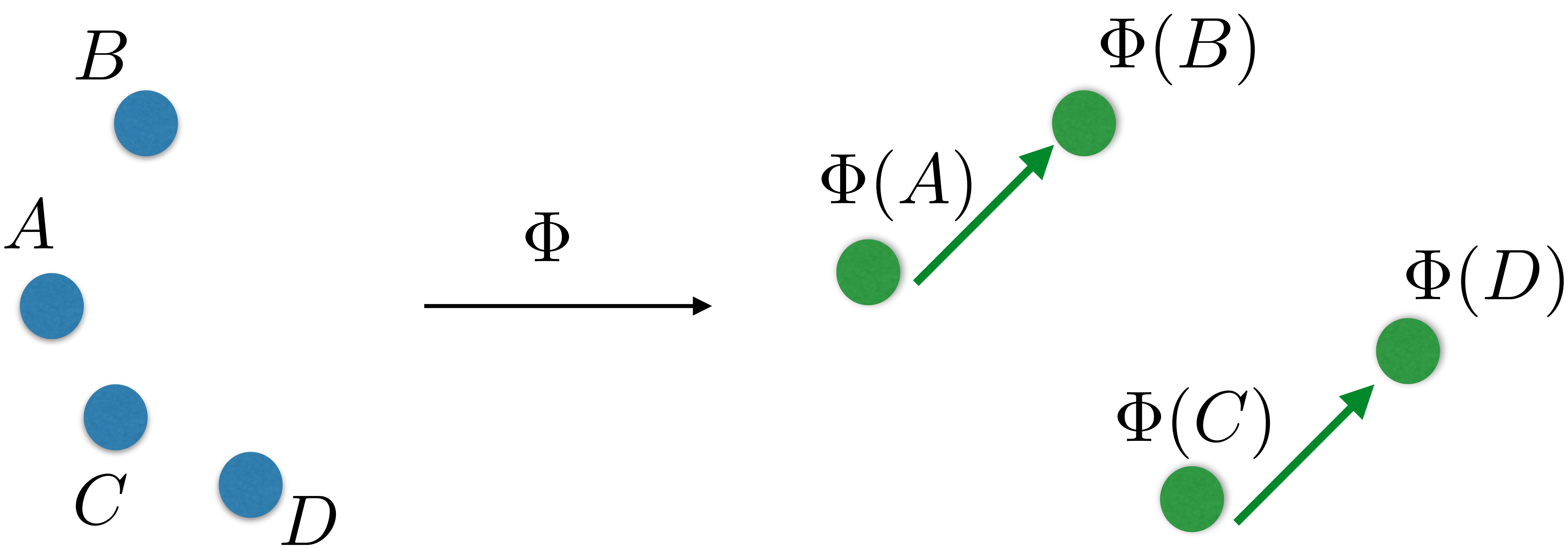
(c) Results 1: visually similar products

(c) Results 2: use of product in-situ

"Learning visual similarity for product design with convolutional neural nets", Bell et al.'15

Application: Visual Analogies

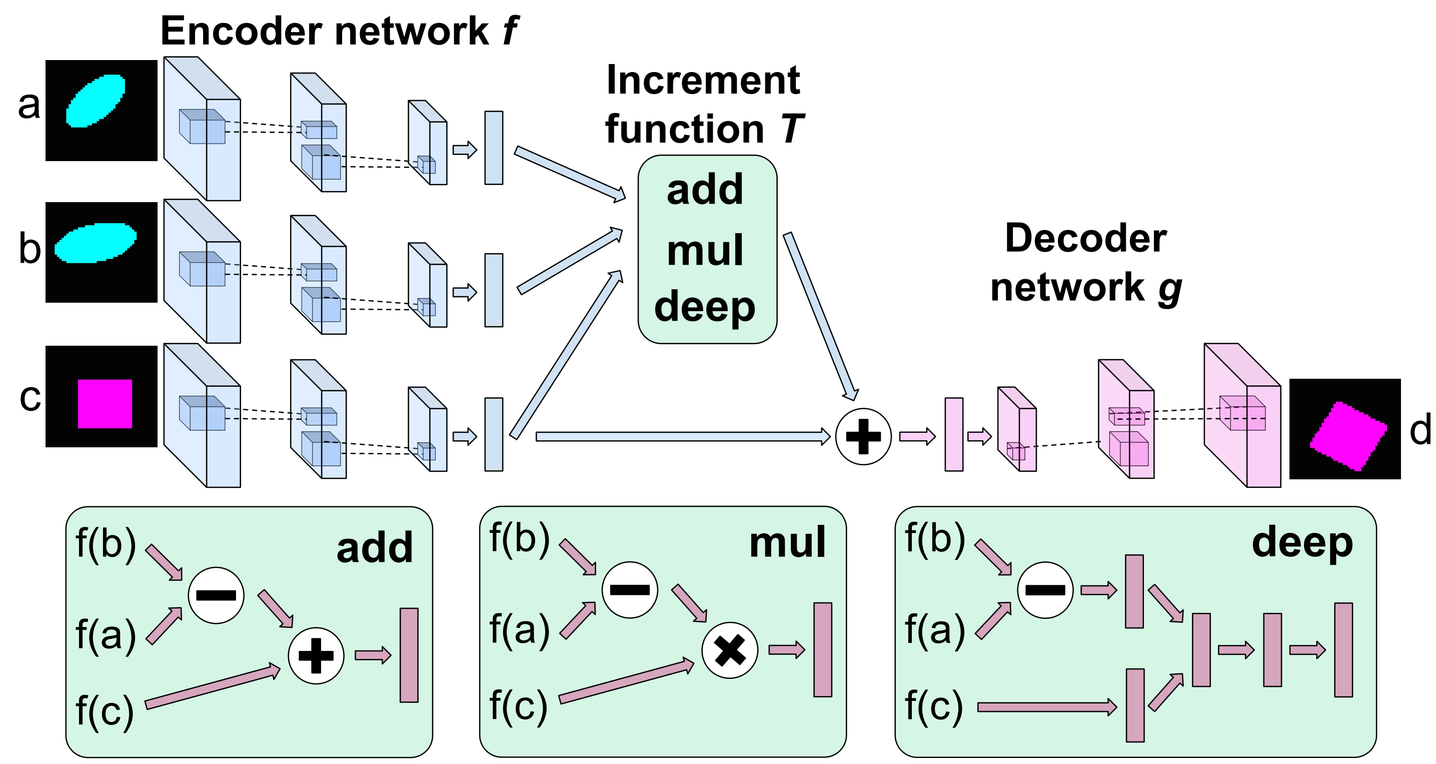
- Analogies are relationships of the form "A is to B as C is to D".
  - E.g. "Paris" is to "France" as "London" is to "UK".
- Q: How to solve analogies using embeddings?
- We can try to *linearize* the analogies:



$$\Phi(D) - \Phi(C) \approx \Phi(B) - \Phi(A)$$

Application: Visual Analogies

- ["Deep Visual Analogy-Making", Reed et al, NIPS'15]



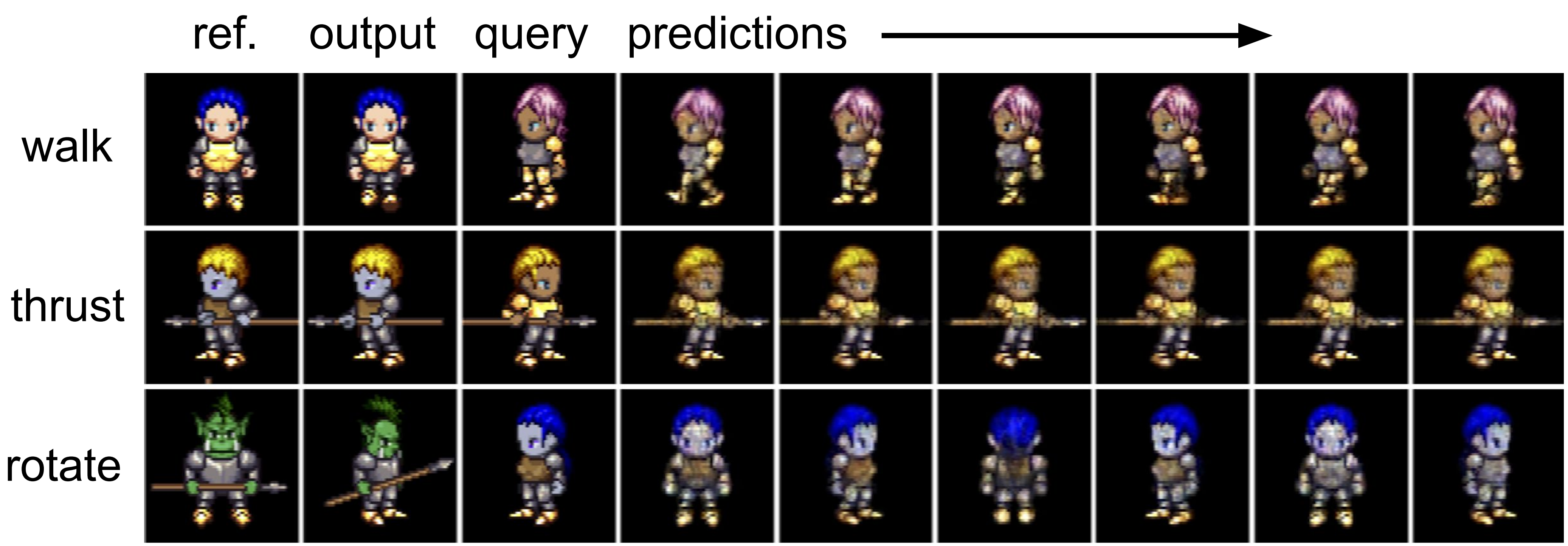
- Given analogy tuples (a, b, c, d), optimize a cost of the form

$$\sum_{(a,b,c,d)} \|d - g(\Phi(b) - \Phi(a) + \Phi(c))\|^2 \ .$$

  - More complicated transformations beyond linear possible.

Application: Visual Analogies

- ["Deep Visual Analogy-Making", Reed et al, NIPS'15]



- Given analogy tuples (a, b, c, d), optimize a cost of the form

$$\sum_{(a,b,c,d)} \|d - g(\Phi(b) - \Phi(a) + \Phi(c))\|^2 \ .$$

  - More complicated transformations beyond linear possible.

Application: One-Shot Learning

- Person recognition with one single training example:

Application: One-Shot Learning

- Person recognition with one single training example:




Application: One-Shot Learning

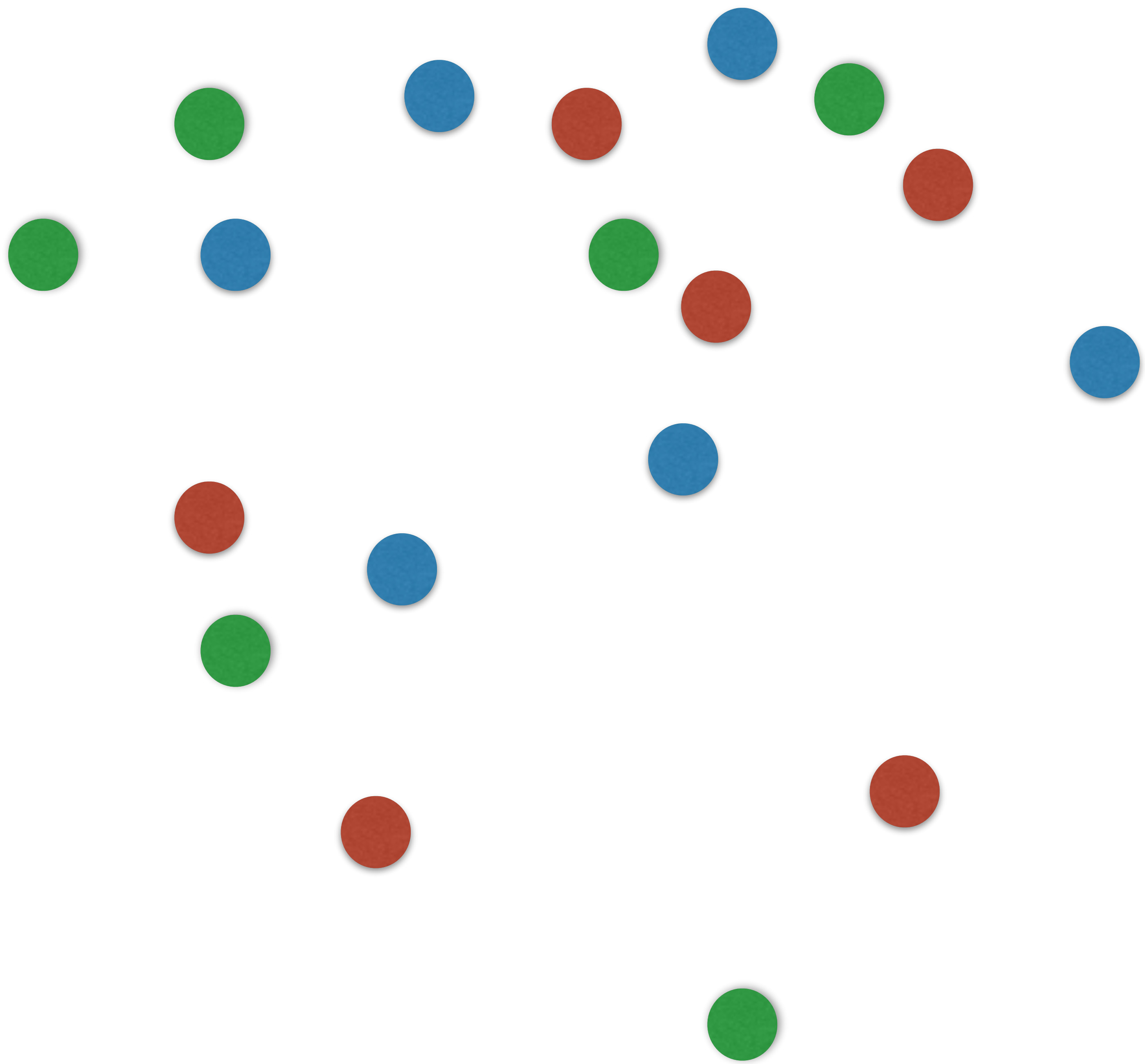
- Person recognition with one single training example:




- Leverage examples from other classes and transfer knowledge

Application: One-shot learning

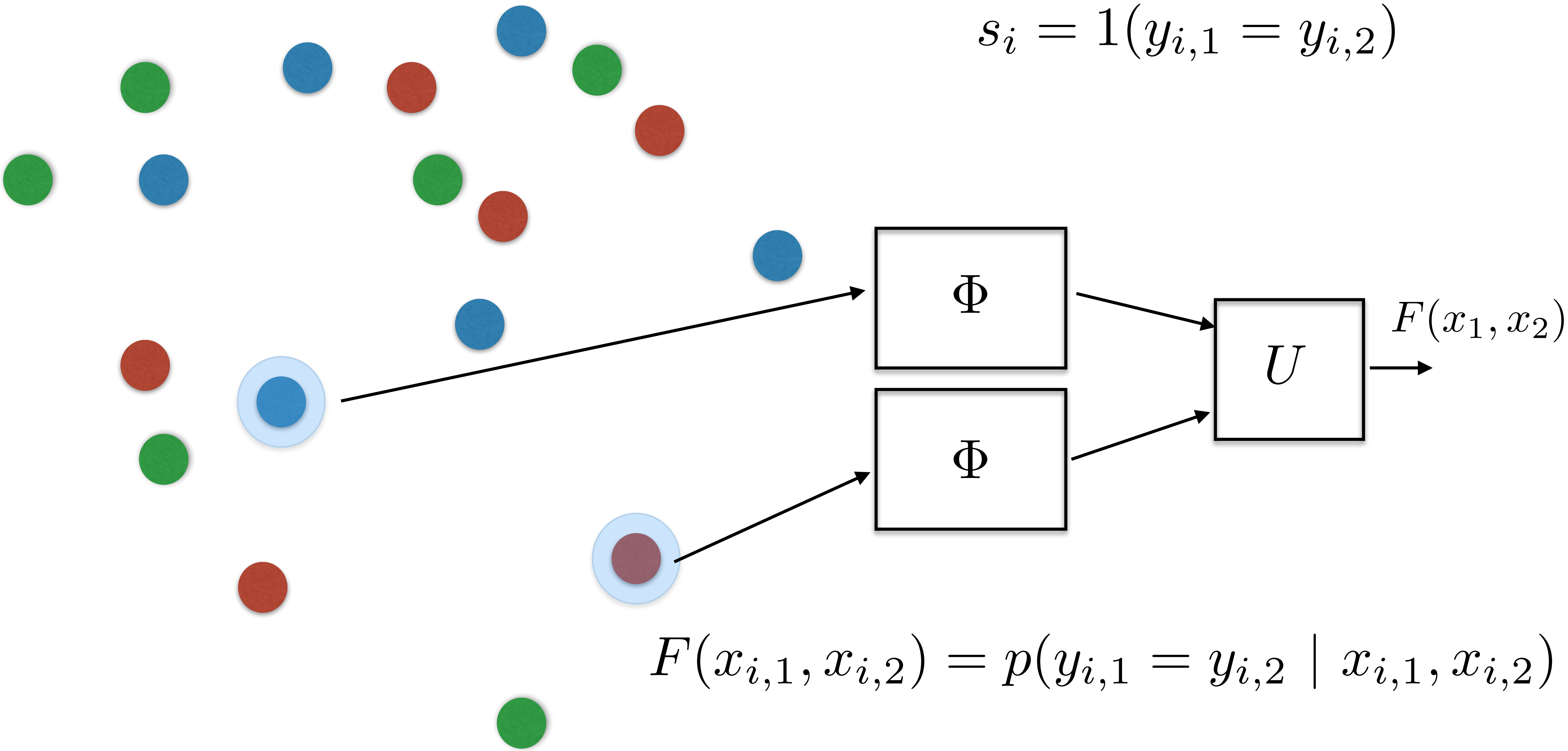
- One-Shot learning with siamese architectures



Application: One-shot learning

- Consider pairs of training examples $(x_{i,1}, y_{i,1}),\ (x_{i,2}, y_{i,2})$

$$s_i = 1(y_{i,1} = y_{i,2})$$

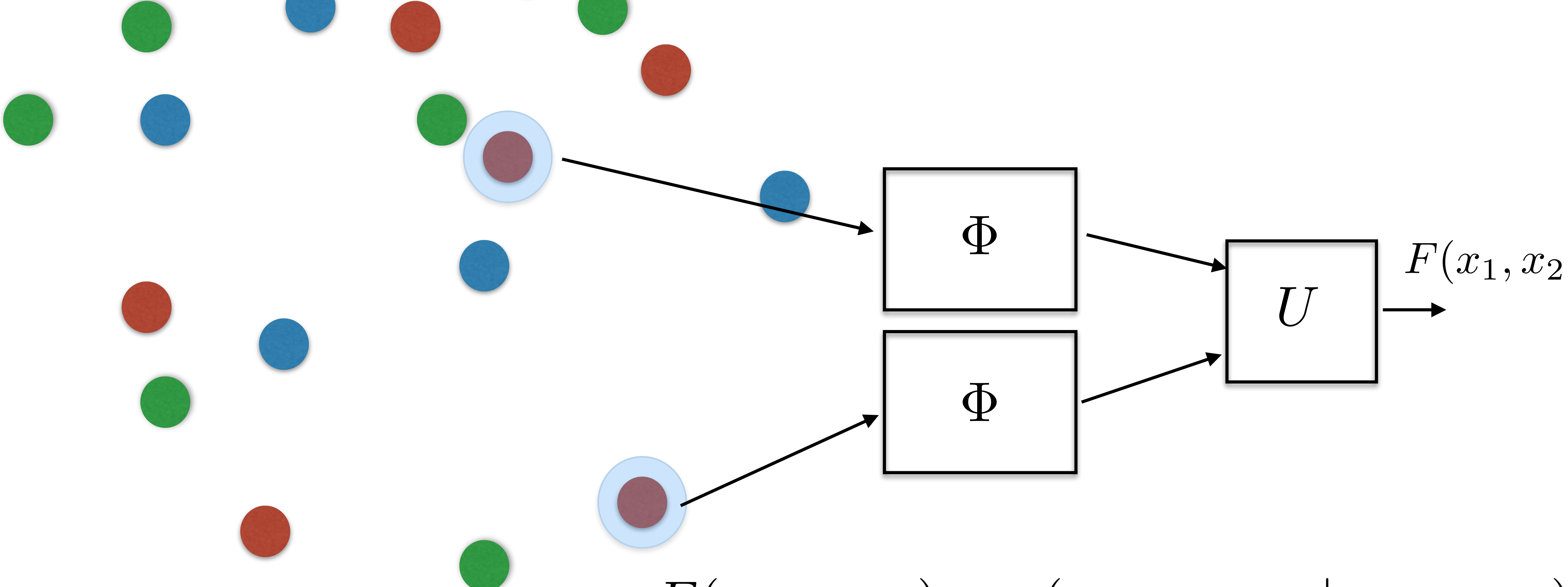


$$F(x_{i,1}, x_{i,2}) = p(y_{i,1} = y_{i,2} \mid x_{i,1}, x_{i,2})$$

Application: One-shot learning

- Consider pairs of training examples $(x_{i,1}, y_{i,1}),\ (x_{i,2}, y_{i,2})$

$$s_i = 1(y_{i,1} = y_{i,2})$$



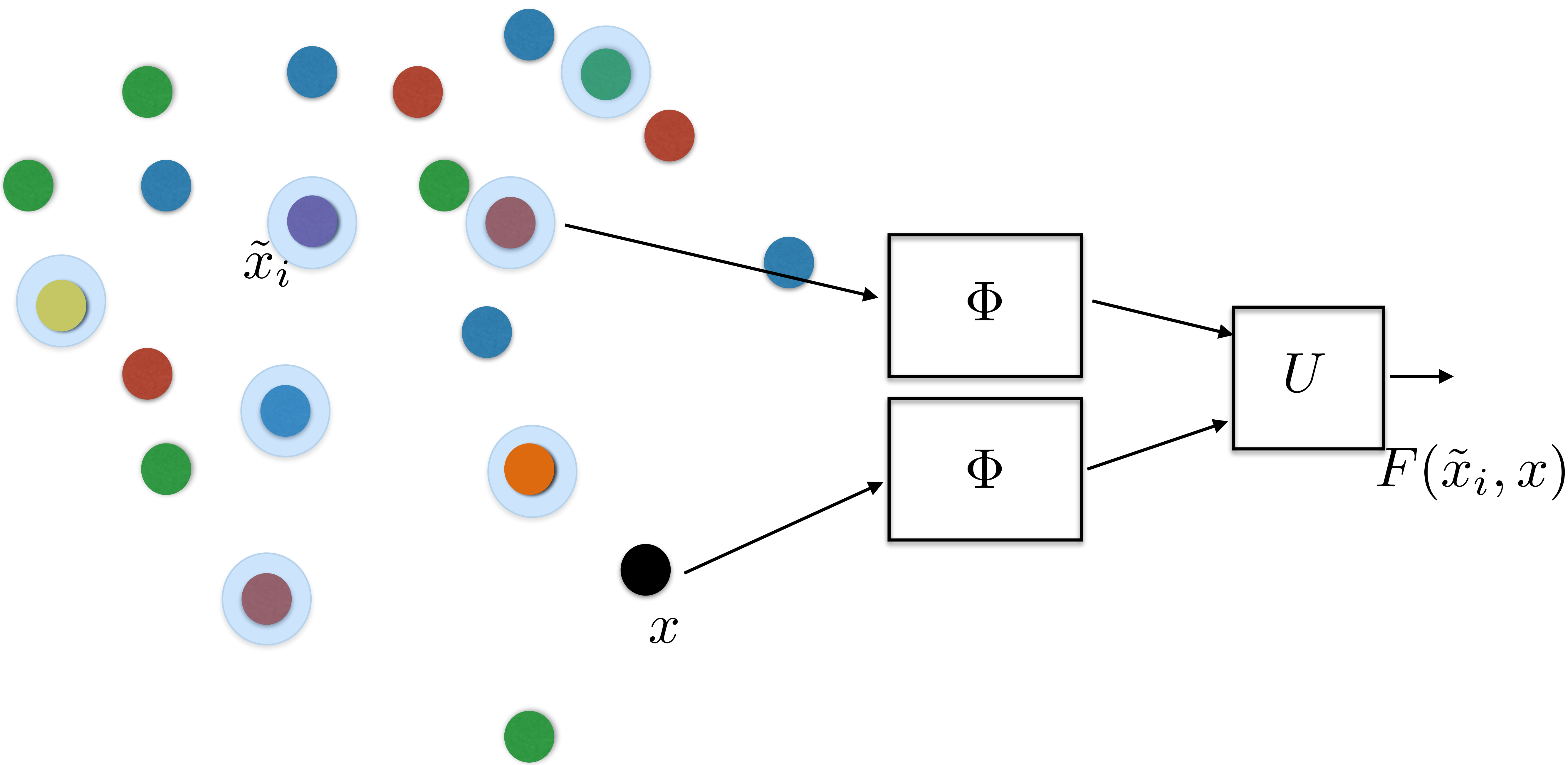
$$F(x_{i,1}, x_{i,2}) = p(y_{i,1} = y_{i,2} \mid x_{i,1}, x_{i,2})$$

- We train the system to detect whether a pair comes from the same class or not.

Application: One-shot learning

- Now, given one training example $\tilde{x}_i$ from each new class and a query x, we estimate the label as

$$\hat{y}(x) = \arg\max_i F(\tilde{x}_i, x) \ .$$



Application: One-Shot Learning

- [G. Koch,'15] uses a CNN siamese architecture on the Omniglot dataset:

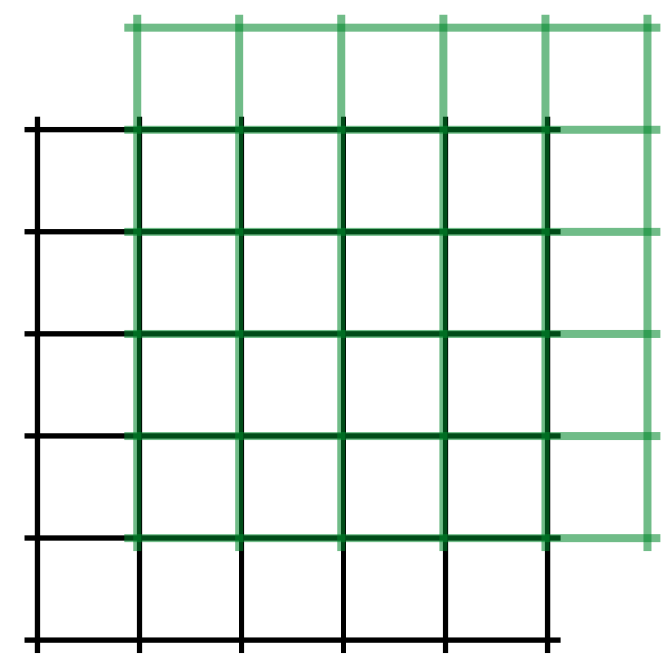
Extensions to non-Euclidean Domain

- So far, we have been able to define convolutional operators to our inputs of the form

$$x(u, \lambda)\ ,\ u \in \mathcal{G} \qquad \mathcal{G} : \mathbb{R}^d \ \ (d = 1, 2, 3)$$

$$\mathcal{G} : \Omega^d \ \ (d = 1, 2, 3)\ ,\ \Omega : \ \text{discrete grid}$$

- In all these cases, the translation group acts on $\mathcal{G}$.
(i.e. $\varphi_v \mathcal{G} = \mathcal{G}$ for all translations $\varphi_v \ \ v \in \mathcal{G}$)



Extensions to non-Euclidean Domain

- So far, we have been able to define convolutional operators to our inputs of the form

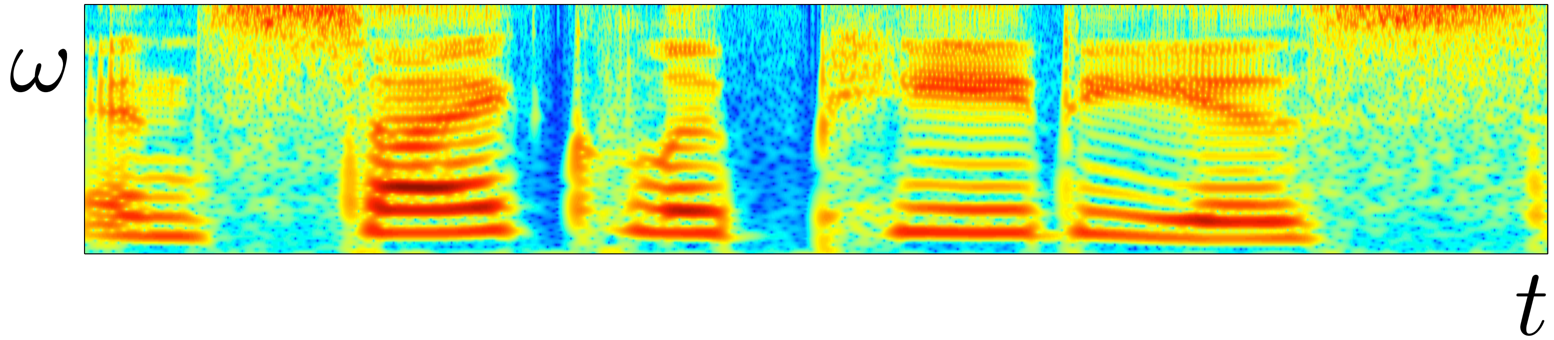
$$x(u, \lambda) \ , \ u \in \mathcal{G} \qquad \begin{array}{l} \mathcal{G} : \mathbb{R}^d \ \ (d = 1, 2, 3) \\ \mathcal{G} : \Omega^d \ \ (d = 1, 2, 3) \ , \ \Omega : \ \text{discrete grid} \end{array}$$

- In all these cases, the translation group acts on $\mathcal{G}$ (i.e. $\varphi_v \mathcal{G} = \mathcal{G}$ for all translations φ_v $v \in \mathcal{G}$)
- Moreover, stability to local deformations and stationarity result in models with localized convolutional operators.
- As a result, the number of parameters to learn is independent of input dimensionality

Limits of Convolutional Networks

- These properties are not present in
  - 3D Mesh data (eg surface tensions)

  - Time-frequency audio representations:

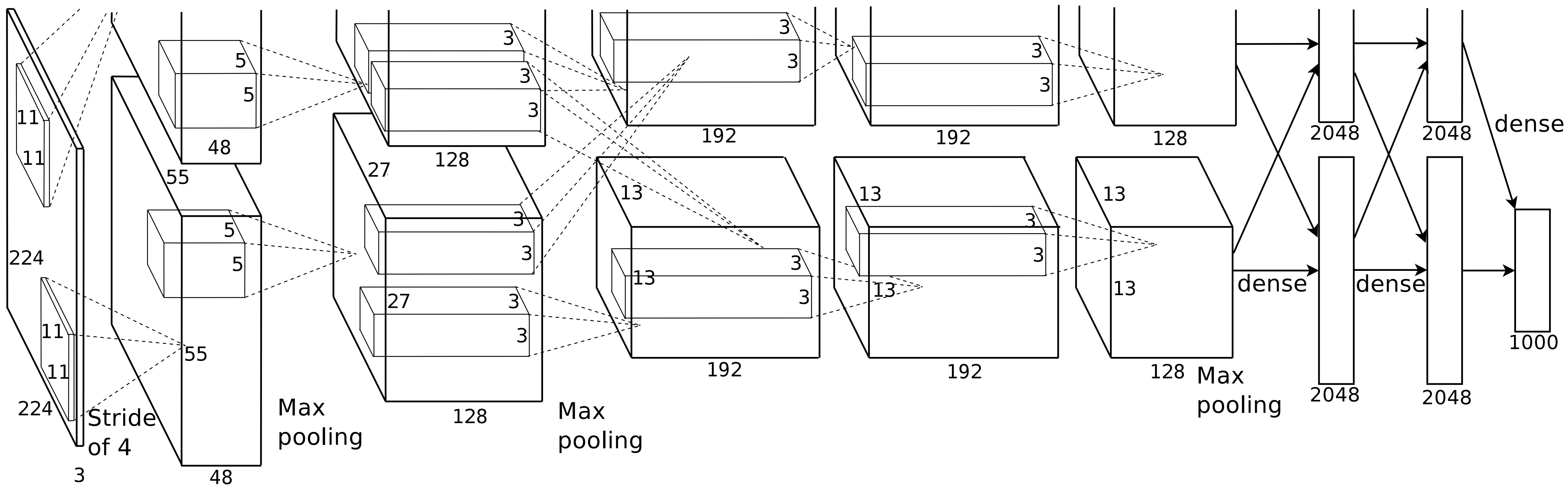


$$x(t,\omega) \text{ is not stationary with respect to } \omega.$$

  - Social Network signals, gene expression, collaborative filtering, etc.

Limits of Convolutional Networks

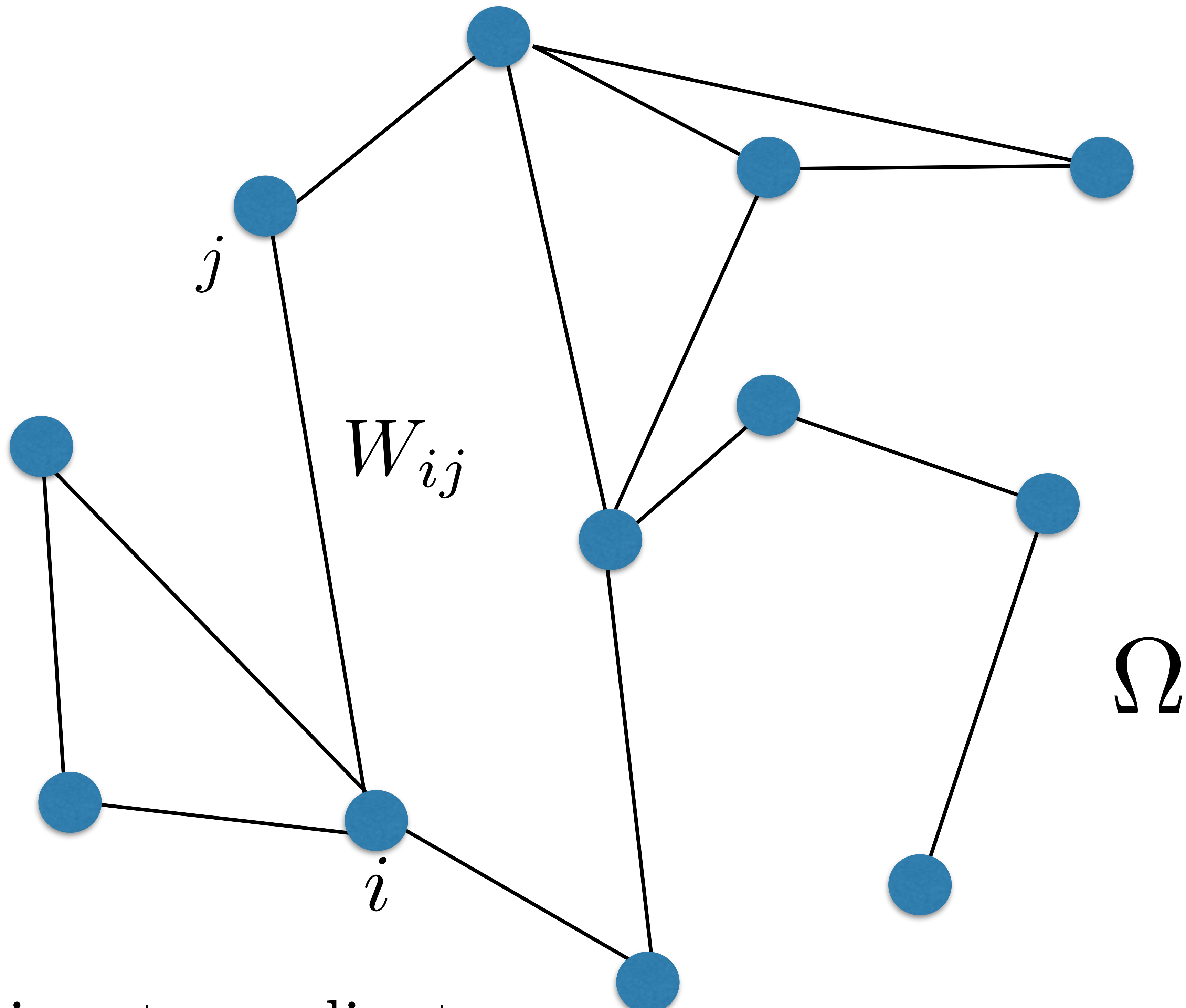
- Intermediate CNN layers



$$x(u, \lambda) \text{ is not stationary with respect to } \lambda.$$

- In general, can we learn with #parameters independent of input size? What architecture?

General Signals: Functions on graphs

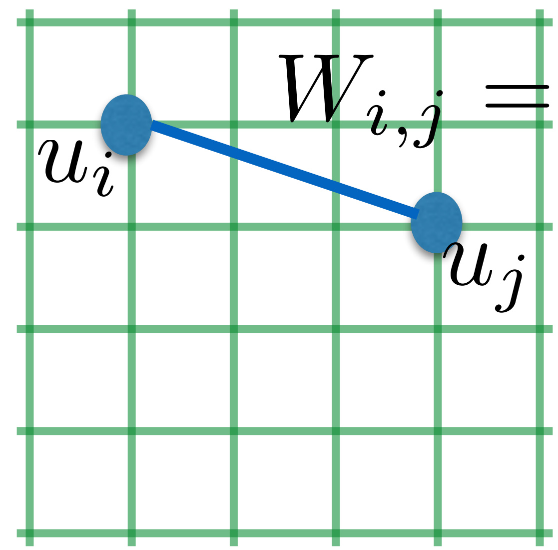


Ω: set of input coordinates

$W_{i,j}$: similarity between coordinates i and j

Feature Similarity

- Similarity can be given by sensing process:
  - (grids, 3D meshes, weather stations)

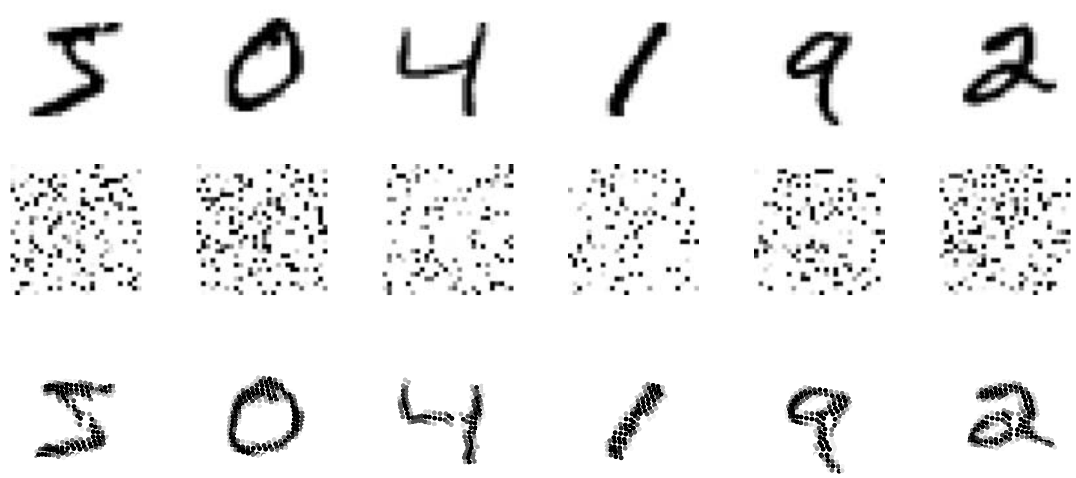


$$W_{i,j} = \exp(-\|u_i - u_j\|^2 / 2\sigma^2)$$

- Or it can also be estimated from the data.

Recovering Graph Structure

- L Observations in dimension N: $X = (x_{i,l})_{i \leq N \,;\, l \leq L}$
- Similarity given by $W_{i,j} = \mathrm{Cov}(|X_i|, |X_j|)$
- Ex: Stationary distributions, in MNIST:



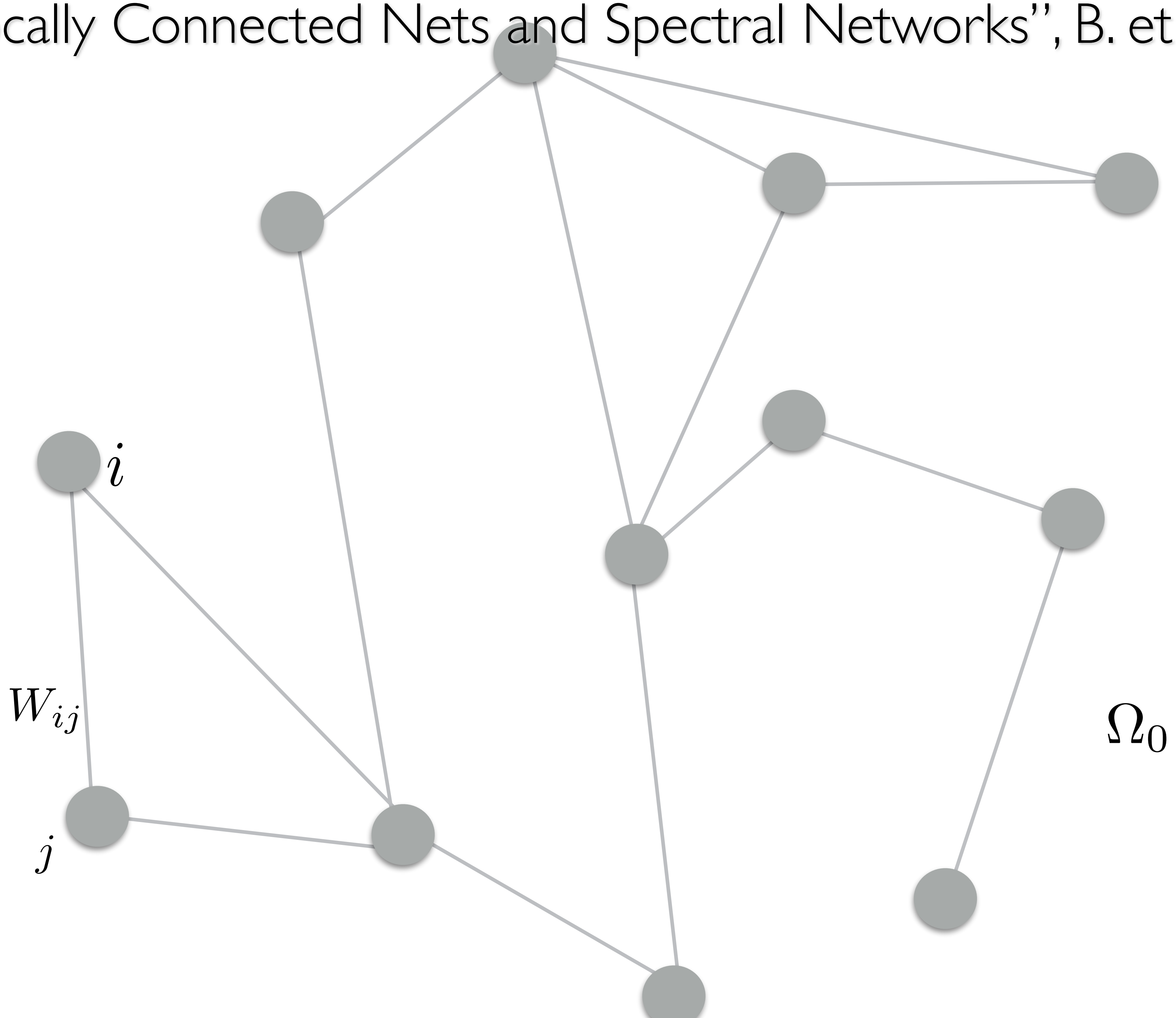
from 'Learning the 2-D topology of images'',
by N.LeRoux, Y. Bengio et al, NIPS 07

- Also in *["Selecting Receptive Fields in Deep Networks", Coates et al, NIPS 11]*.
- Richer statistics can be used to define similarity.

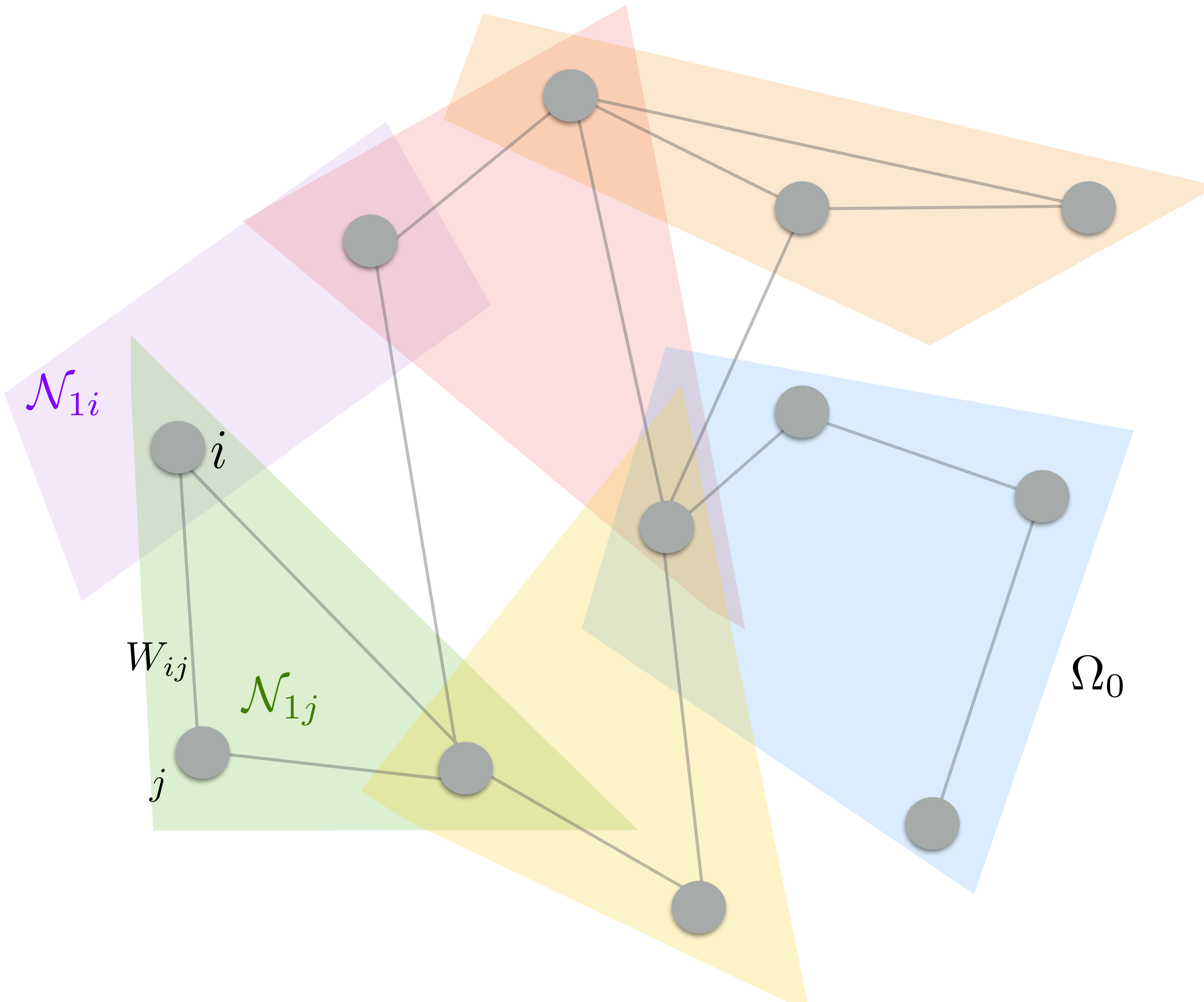
Locally Connected Networks

[“Selecting Receptive Fields in Deep Networks”, Coates & Ng, NIPS’11]

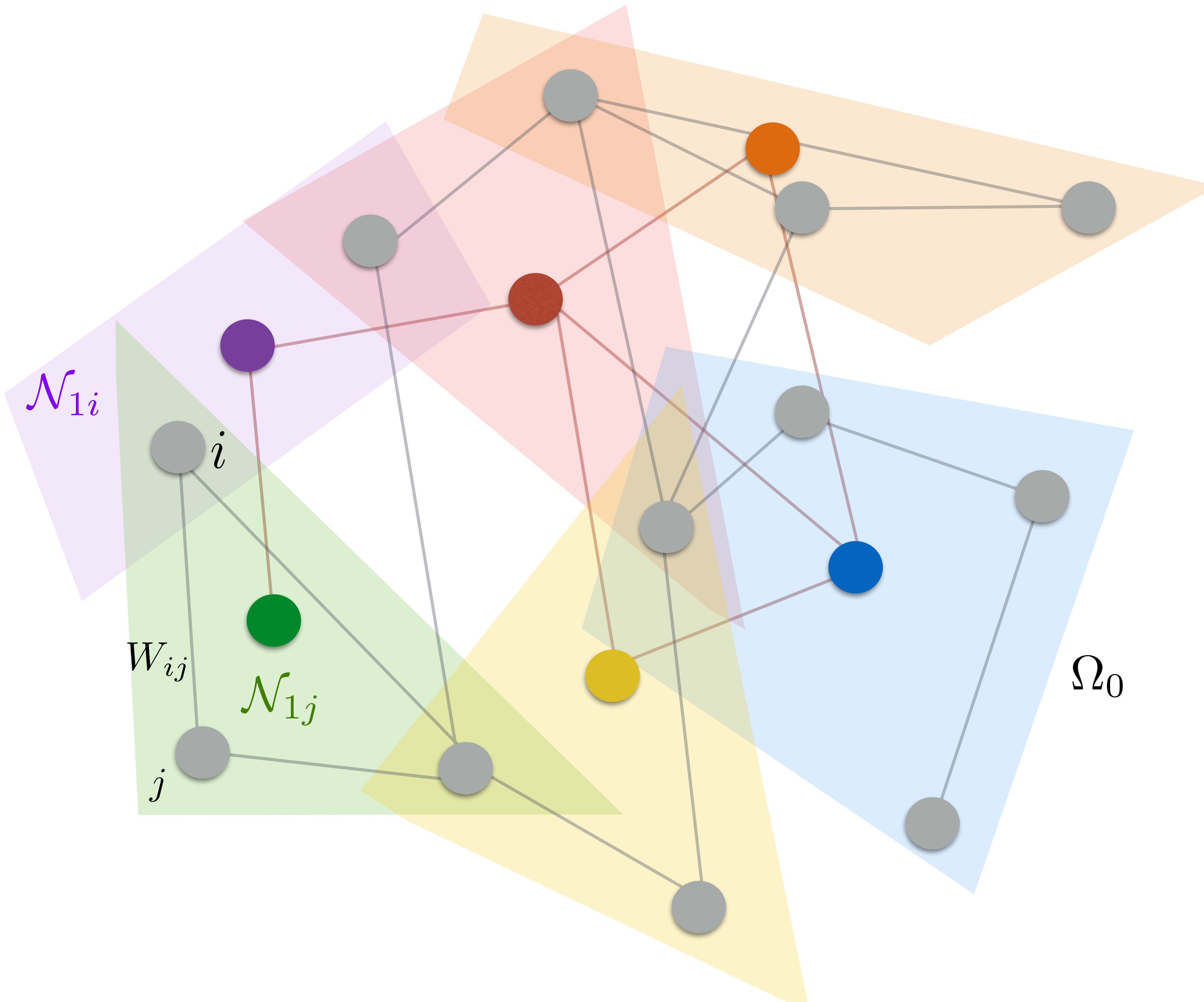
[“Locally Connected Nets and Spectral Networks”, B. et al, ICLR’14]



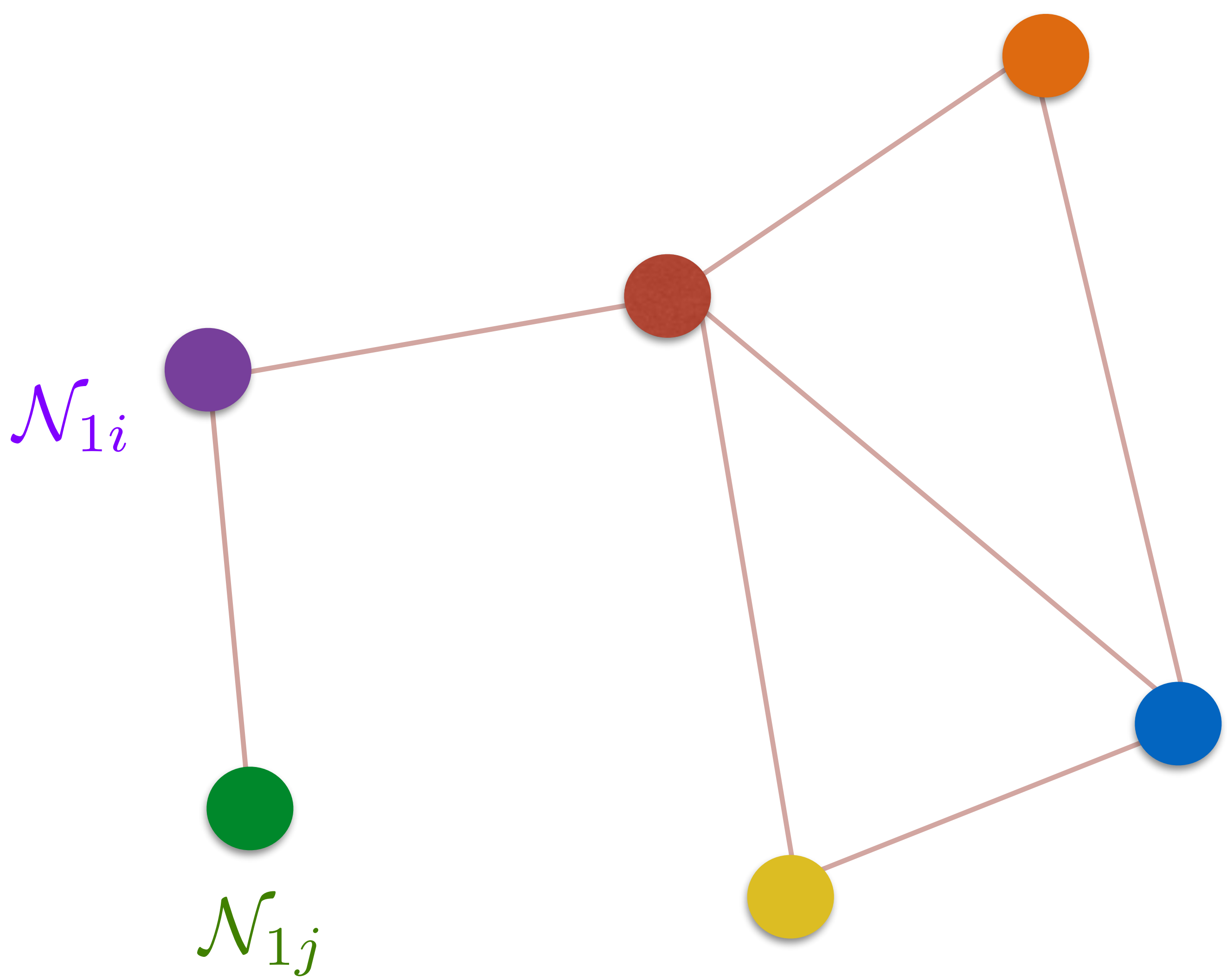
Locally Connected Networks



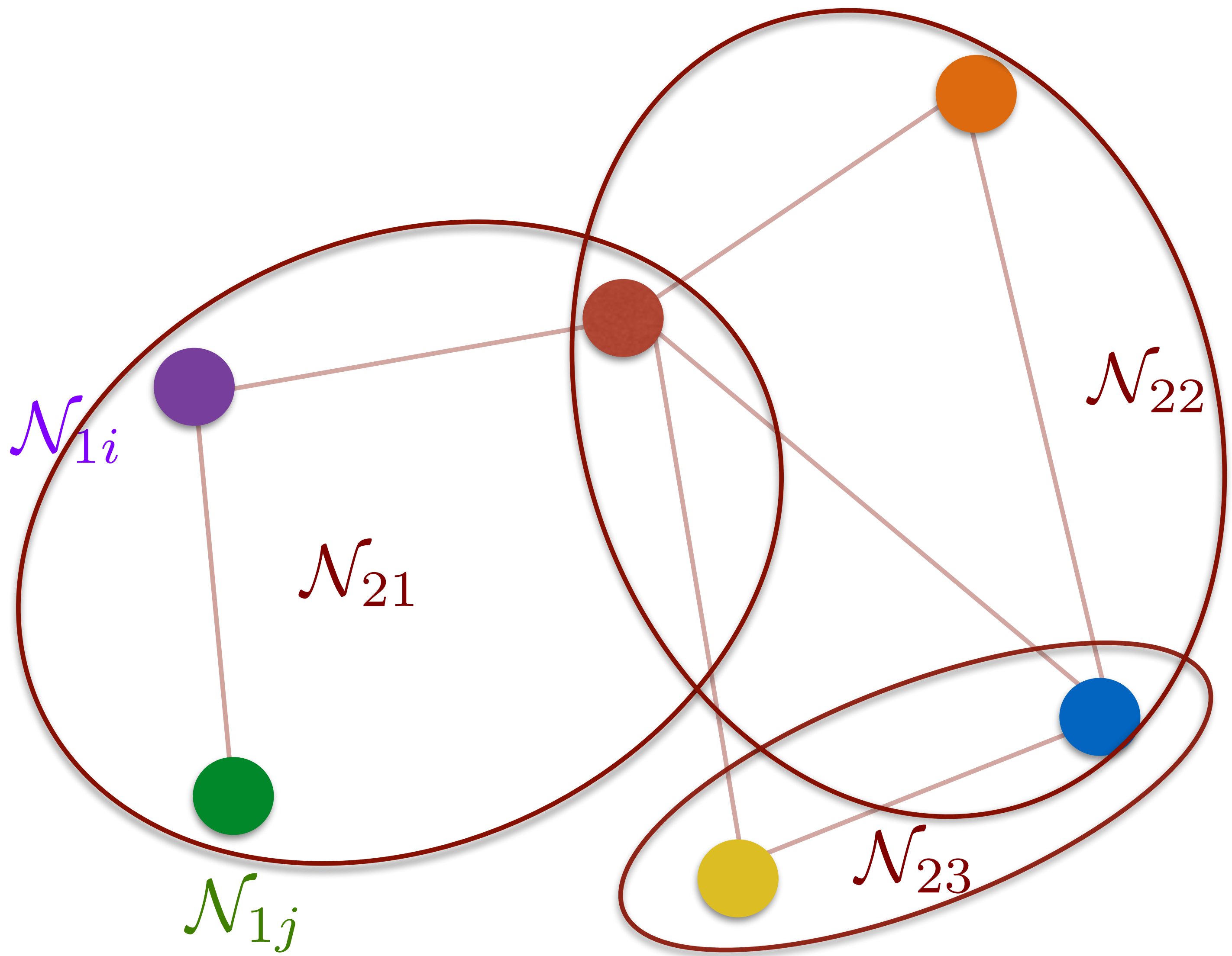
Locally Connected Networks



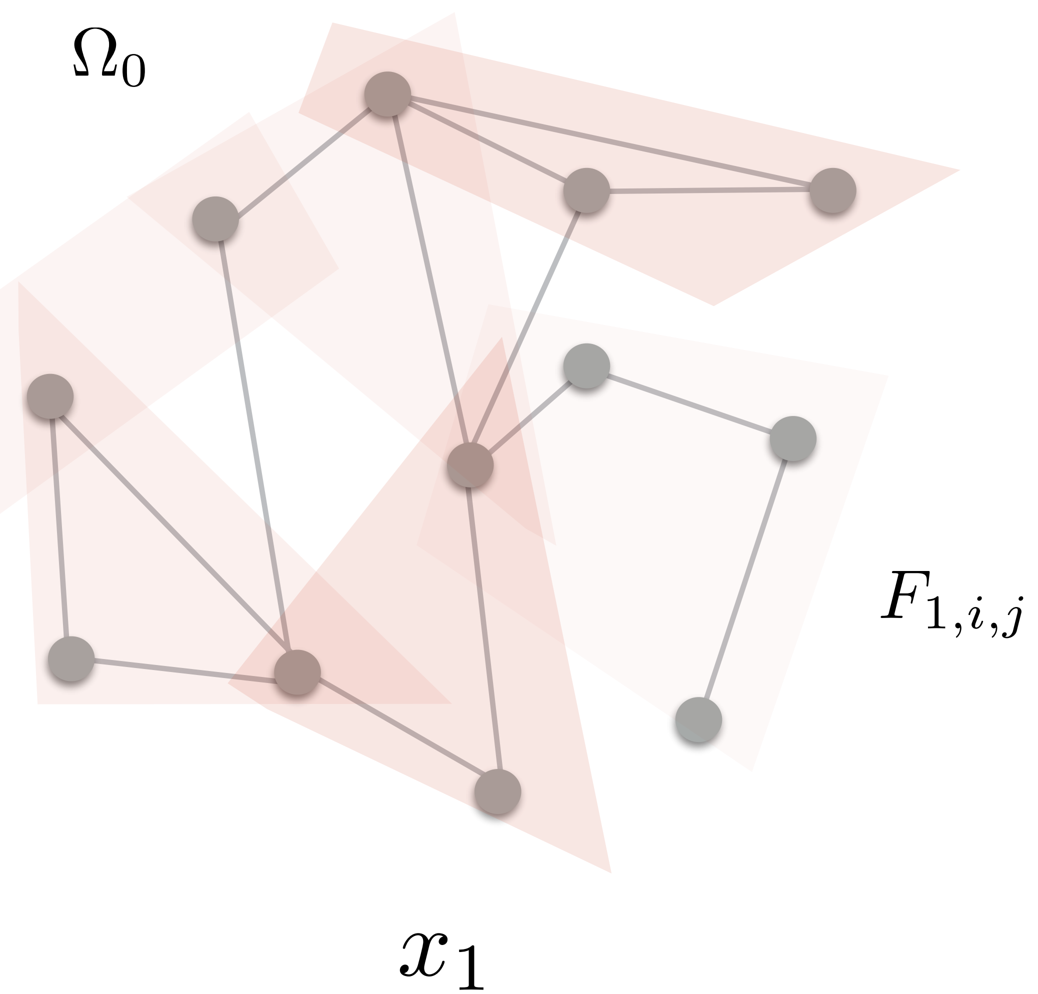
Locally Connected Networks



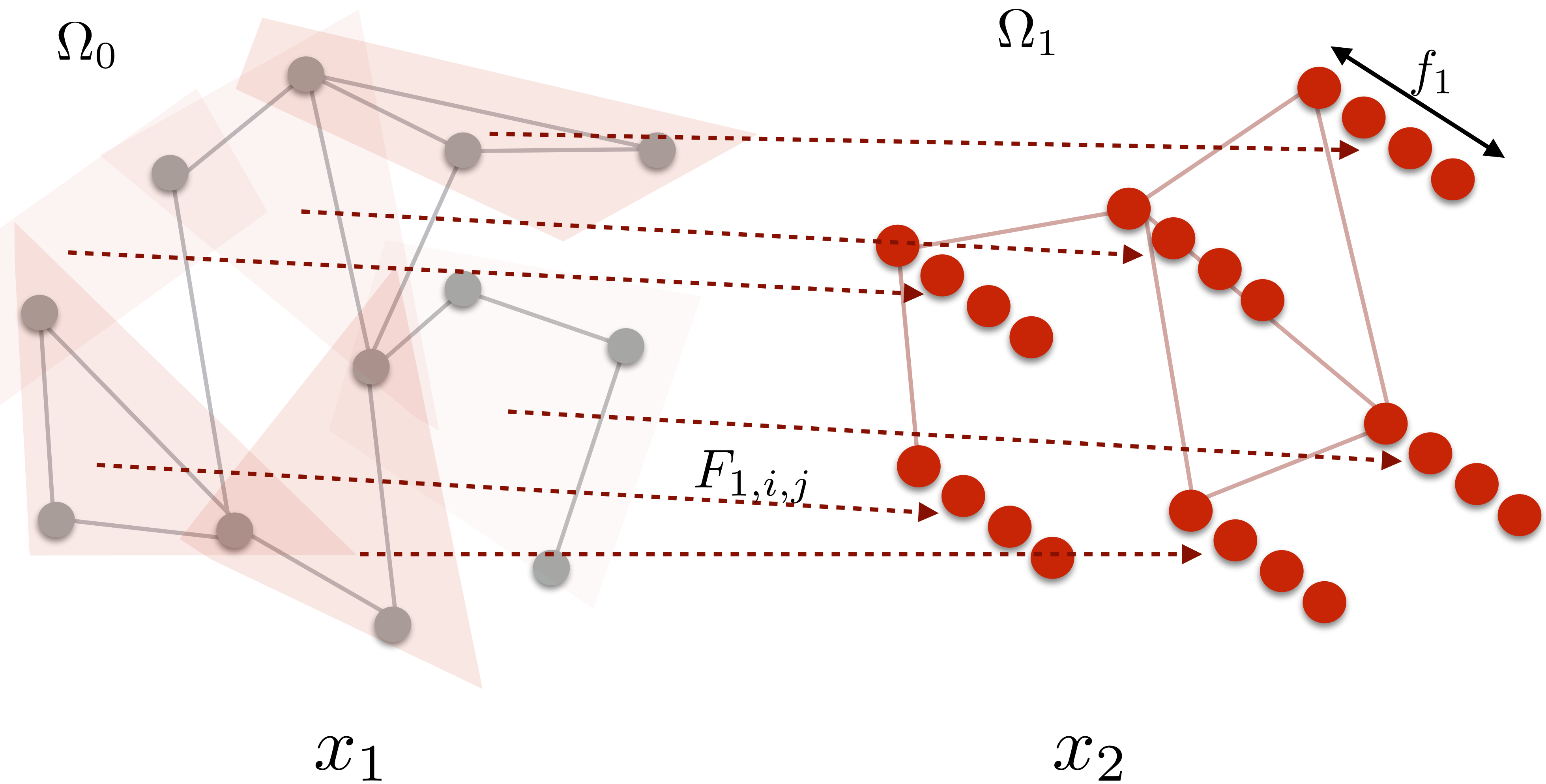
Locally Connected Networks



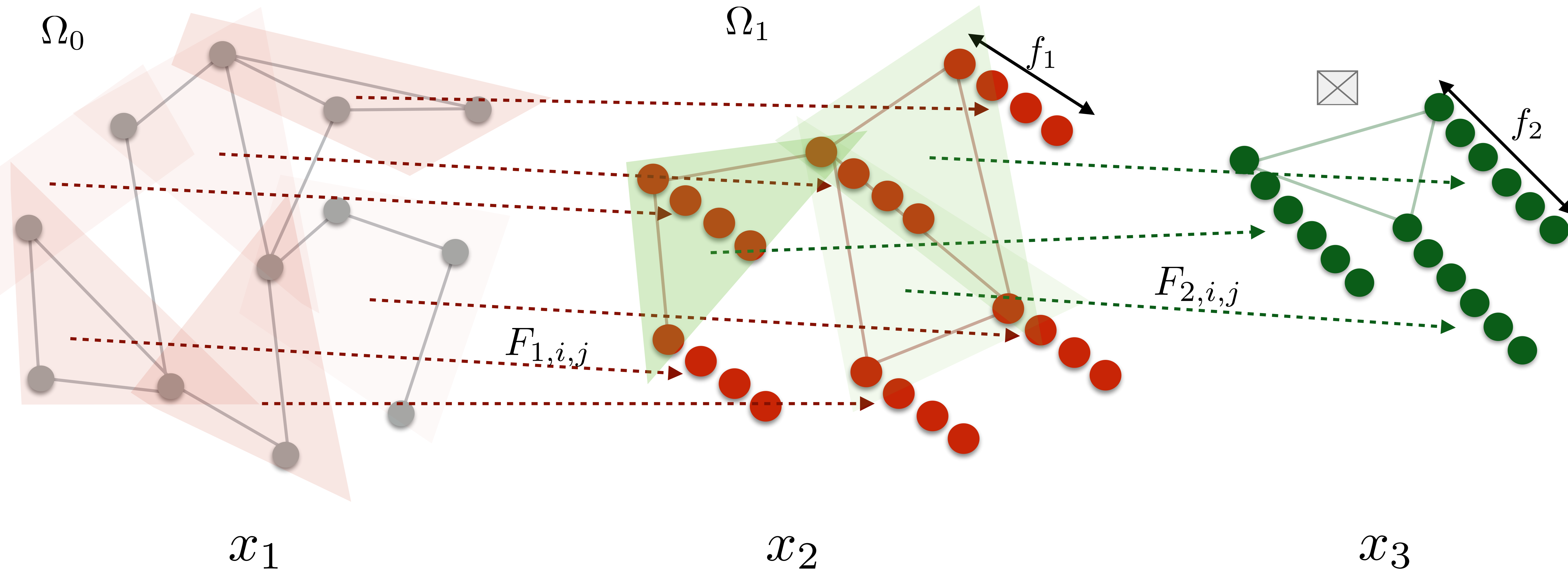
Locally Connected Networks



Locally Connected Networks



Locally Connected Networks



- Hierarchical Clustering of Graph
- This gives O(n) parameters per feature map.

Spectral Networks

- In $\mathbb{R}^d$, convolutions are diagonalized in Fourier domain:

$$x * h = \mathcal{F}^{-1}\mathrm{diag}(\mathcal{F}h)\mathcal{F}x \ ,$$

where $\mathcal{F}_{k,l} = \exp\left(\frac{-2\pi i(k \cdot l)}{N^d}\right) \ .$

Spectral Networks

- In $\mathbb{R}^d$, convolutions are diagonalized in Fourier domain:

$$x * h = \mathcal{F}^{-1}\mathrm{diag}(\mathcal{F}h)\mathcal{F}x \ ,$$

where $\mathcal{F}_{k,l} = \exp\left(\frac{-2\pi i(k \cdot l)}{N^d}\right) \ .$

- Fourier basis can be defined as the eigenbasis of Laplacian operator:

$$\Delta x(u) = \sum_{j \leq d} \frac{\partial^2 x}{\partial u_j^2}(u) \ .$$

Graph Laplacian

- We can define the Laplacian on an undirected graph:

$$\Delta = (I - \tilde{W}) \; , \; \tilde{W} = D^{-1/2} W D^{-1/2} \; , \; D = \mathrm{diag}(W\mathbf{1})$$

$$(\Delta x)_k = x_k - \sum_j \tilde{w}_{kj} x_j$$

measures smoothness in the graph

Graph Laplacian

- We can define the Laplacian on an undirected graph:

$$\Delta = (I - \tilde{W}) \,,\ \tilde{W} = D^{-1/2} W D^{-1/2} \,,\ D = \mathrm{diag}(W\mathbf{1})$$

$$(\Delta x)_k = x_k - \sum_j \tilde{w}_{kj} x_j$$

measures smoothness in the graph

- Δ is positive definite and symmetric. $\Delta = V \mathrm{diag}(\lambda) V^T$

Graph Laplacian

- We can define the Laplacian on an undirected graph:

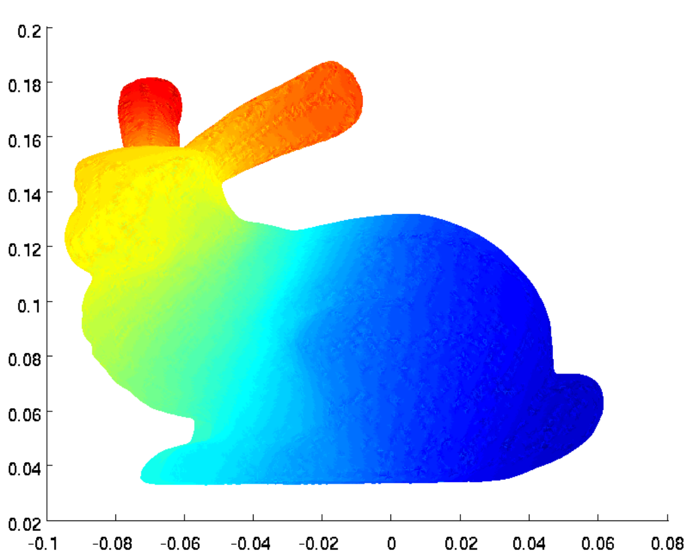
$$\boxed{\Delta = (I - \tilde{W})}\ , \quad \tilde{W} = D^{-1/2} W D^{-1/2}\ , \quad D = \mathrm{diag}(W\mathbf{1})$$

$$(\Delta x)_k = x_k - \sum_j \tilde{w}_{kj} x_j$$

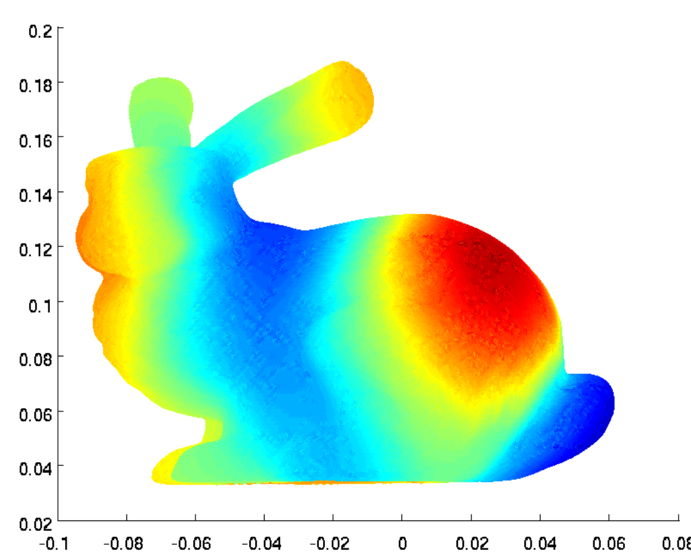
measures smoothness in the graph

- Δ is positive definite and symmetric. $\quad \Delta = V \mathrm{diag}(\lambda) V^T$

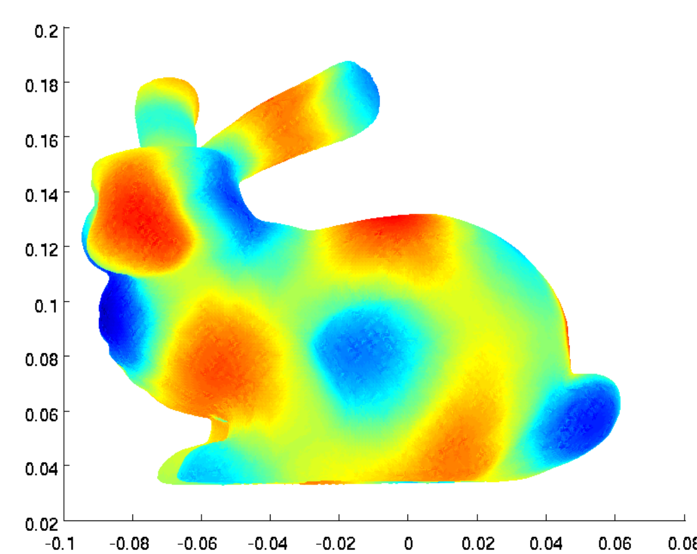
- "Fourier basis" of the graph: V : Eigenvectors of Δ



v_2



v_{10}



v_{30}

Spectral Networks

- "Convolution" on a graph: Linear Operator commuting with Δ:

$$x *_G h := V \mathrm{diag}(h) V^T x$$

  - Filter coefficients *h* are specified in the spectral domain.

Spectral Networks

- "Convolution" on a graph: Linear Operator commuting with Δ:

$$x *_G h := V \mathrm{diag}(h) V^T x$$

  – Filter coefficients *h* are specified in the spectral domain.

- Spectral Network: filter bank $(x *_G h_k)_{k \leq K}$

Spectral Networks

- "Convolution" on a graph: Linear Operator commuting with Δ:

$$x *_G h := V \mathrm{diag}(h) V^T x$$

  – Filter coefficients *h* are specified in the spectral domain.

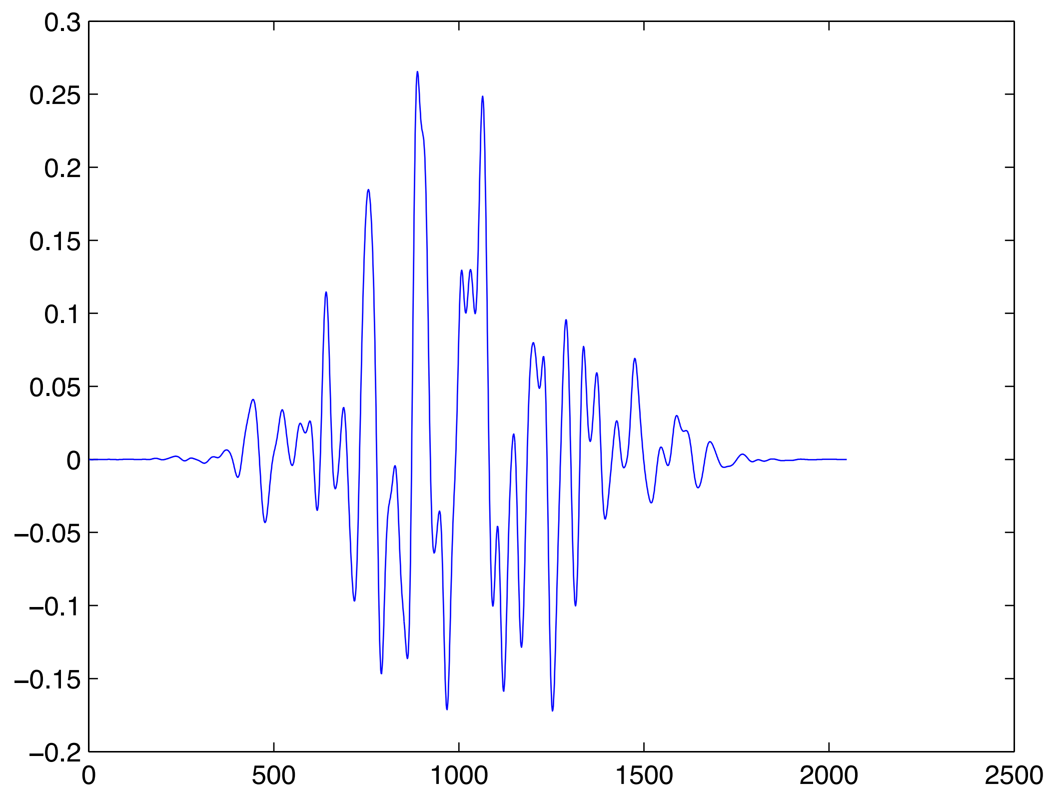
- Spectral Network: filter bank $(x *_G h_k)_{k \leq K}$

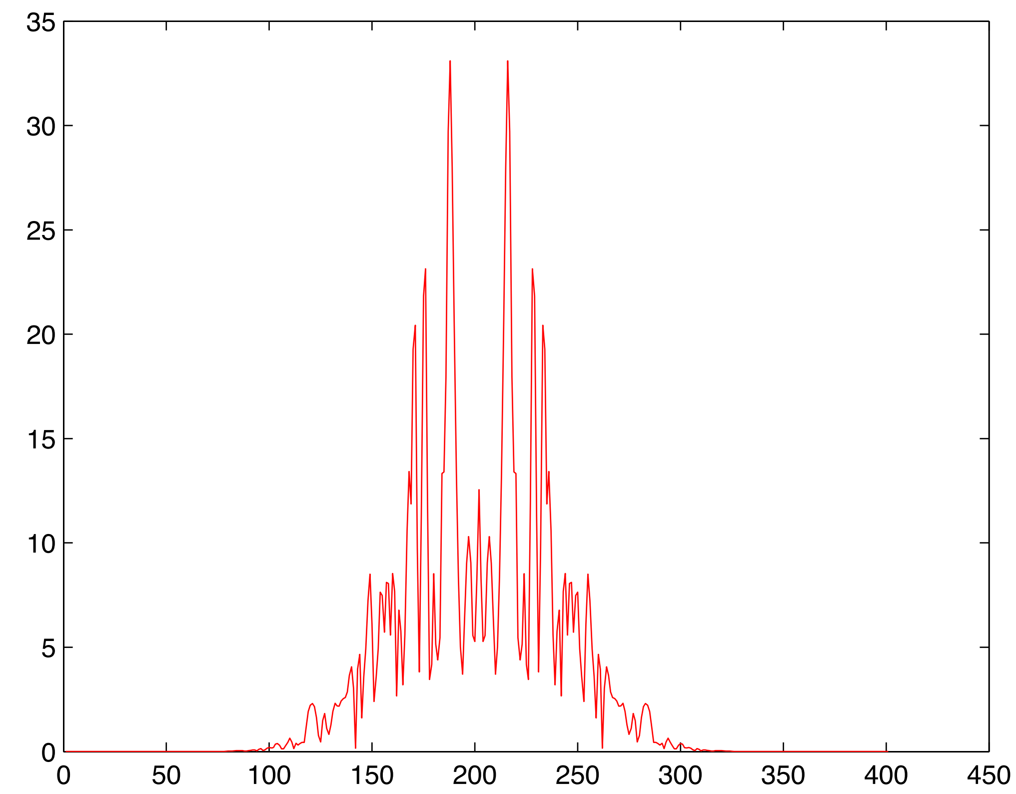
- We still require O(n) parameters per filter.

Spectral Networks

- In R^d, Smoothness and sparsity are dual notions:

$$x \text{ fast decay} \Longleftrightarrow \widehat{x} \text{ smooth}$$

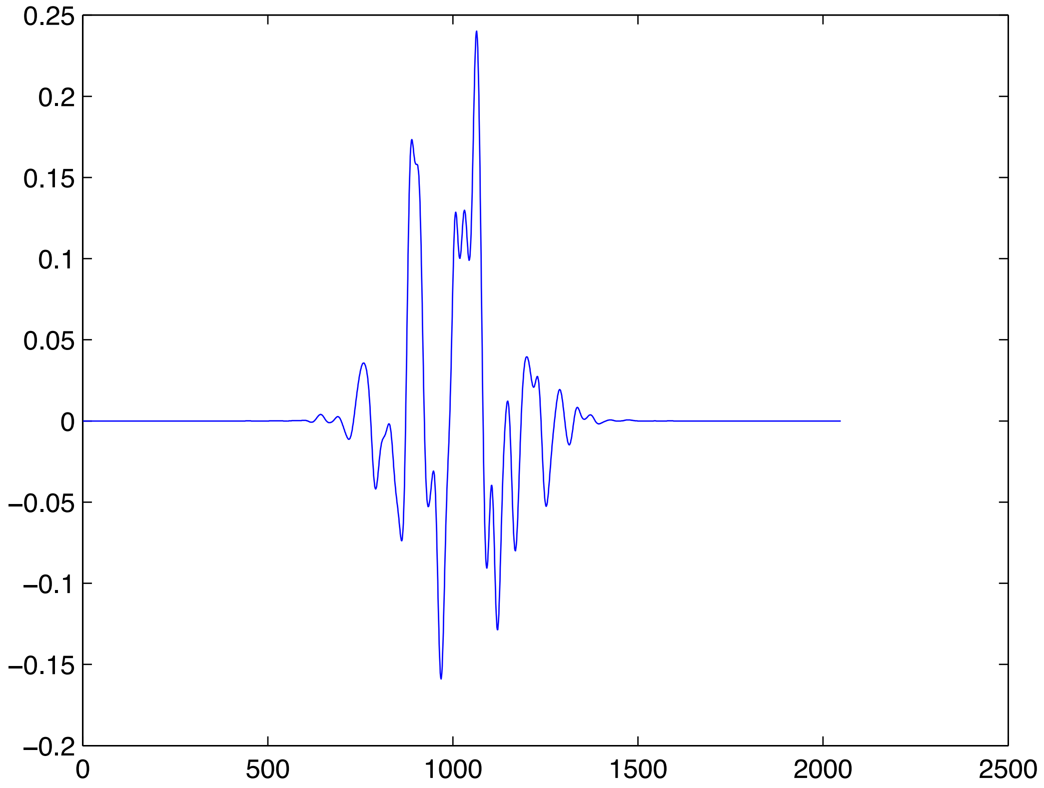


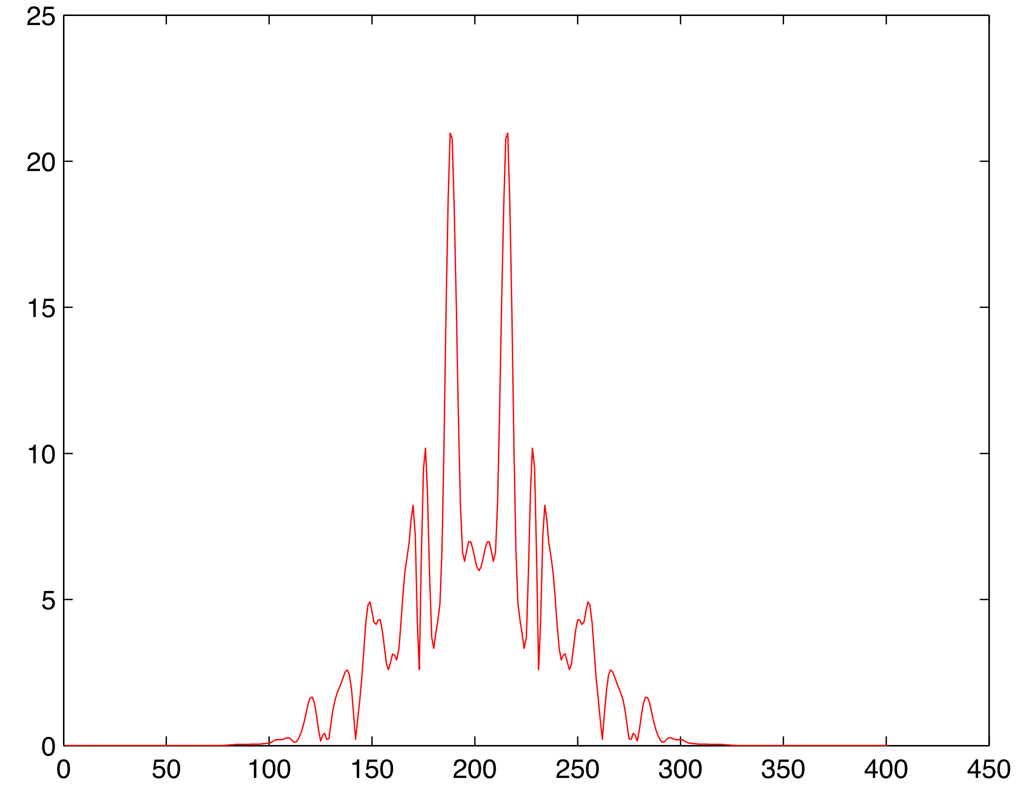


Spectral Networks

- In R^N, Smoothness and sparsity are dual notions:

$$x \text{ fast decay} \iff \widehat{x} \text{ smooth}$$

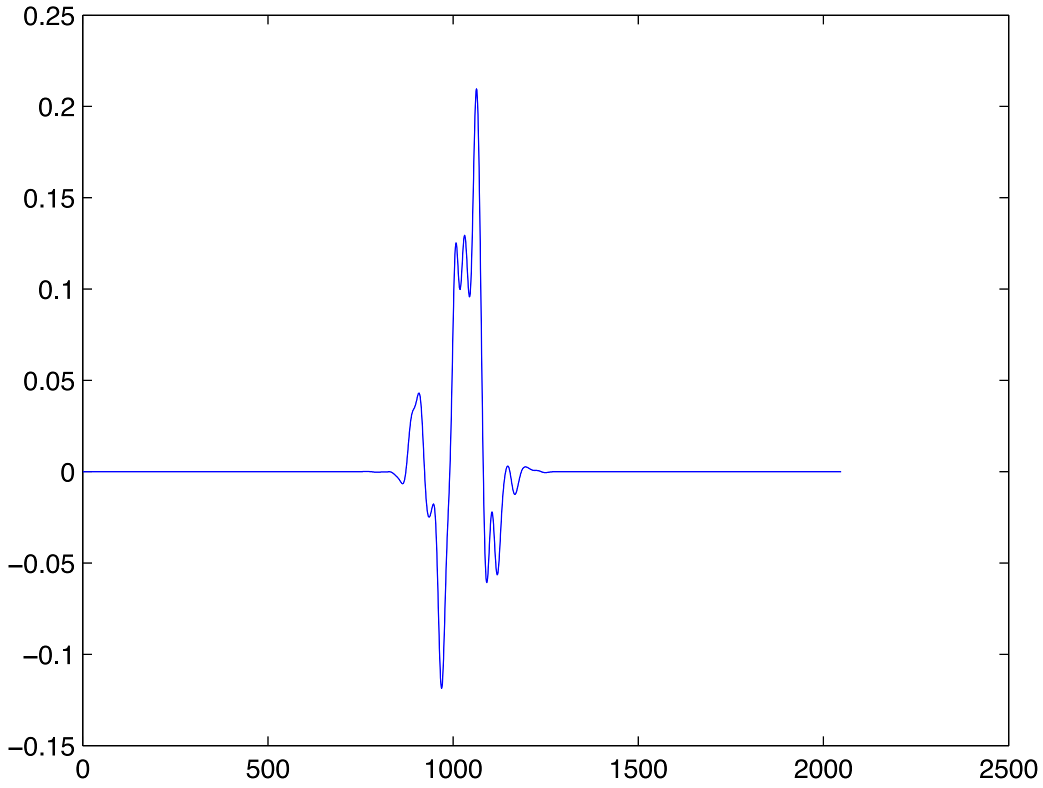


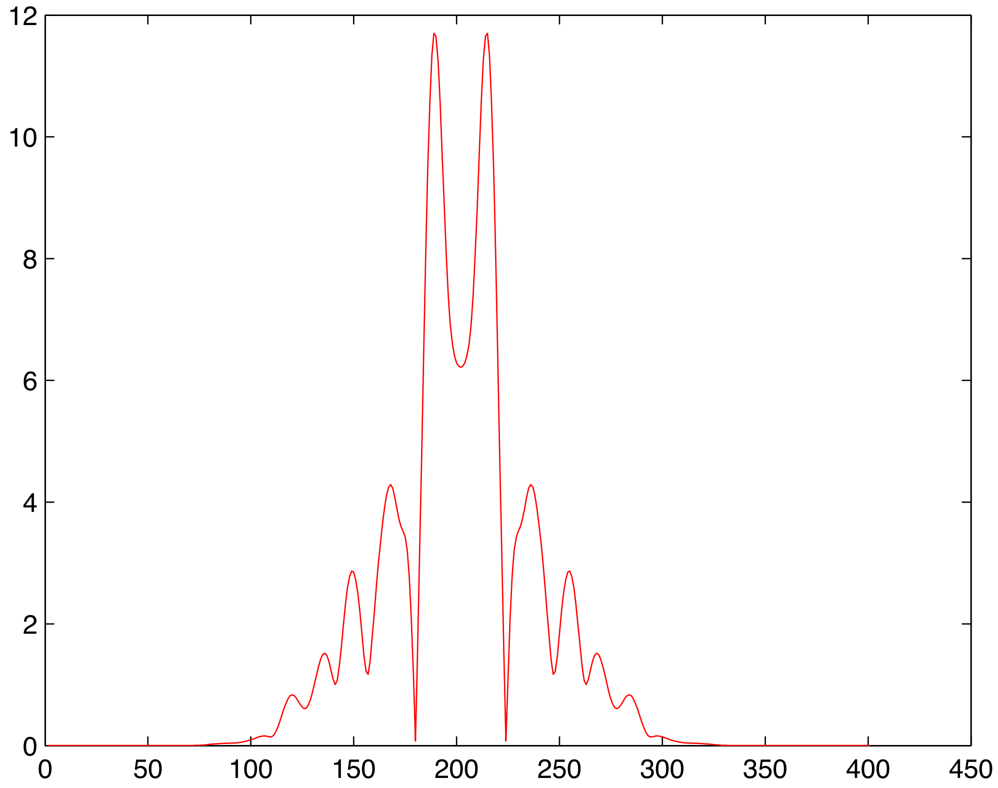


Spectral Networks

- In R^N, Smoothness and sparsity are dual notions:

$$x \text{ fast decay} \Longleftrightarrow \widehat{x} \text{ smooth}$$

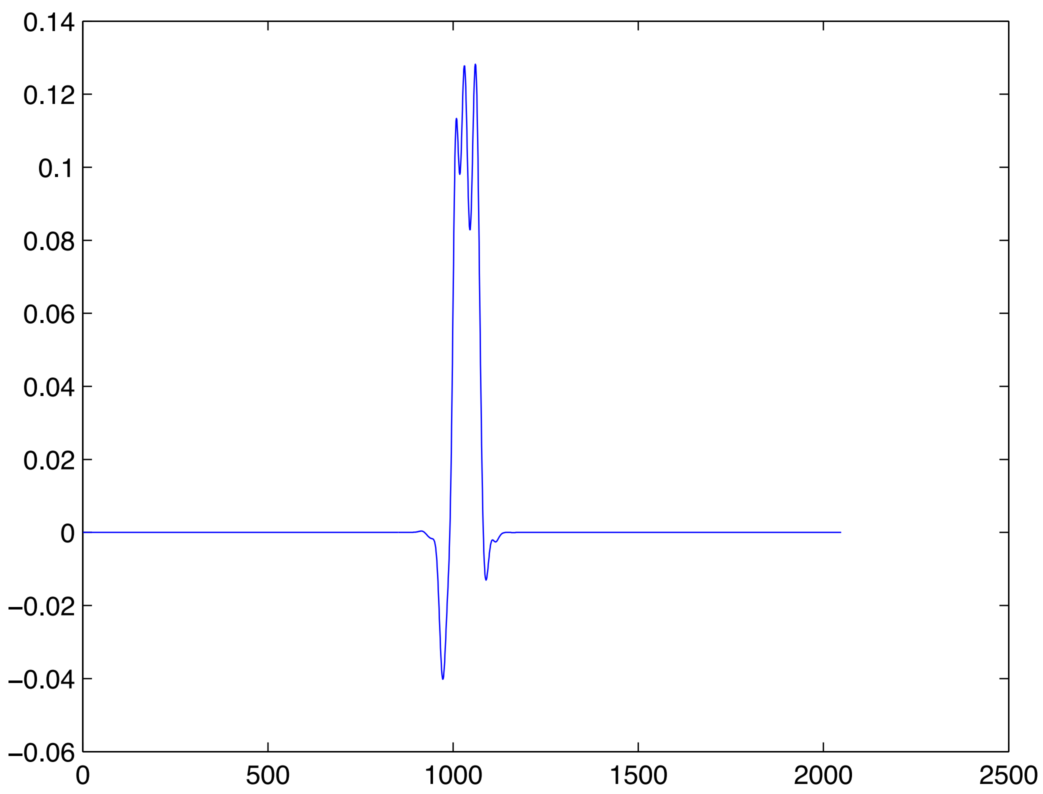


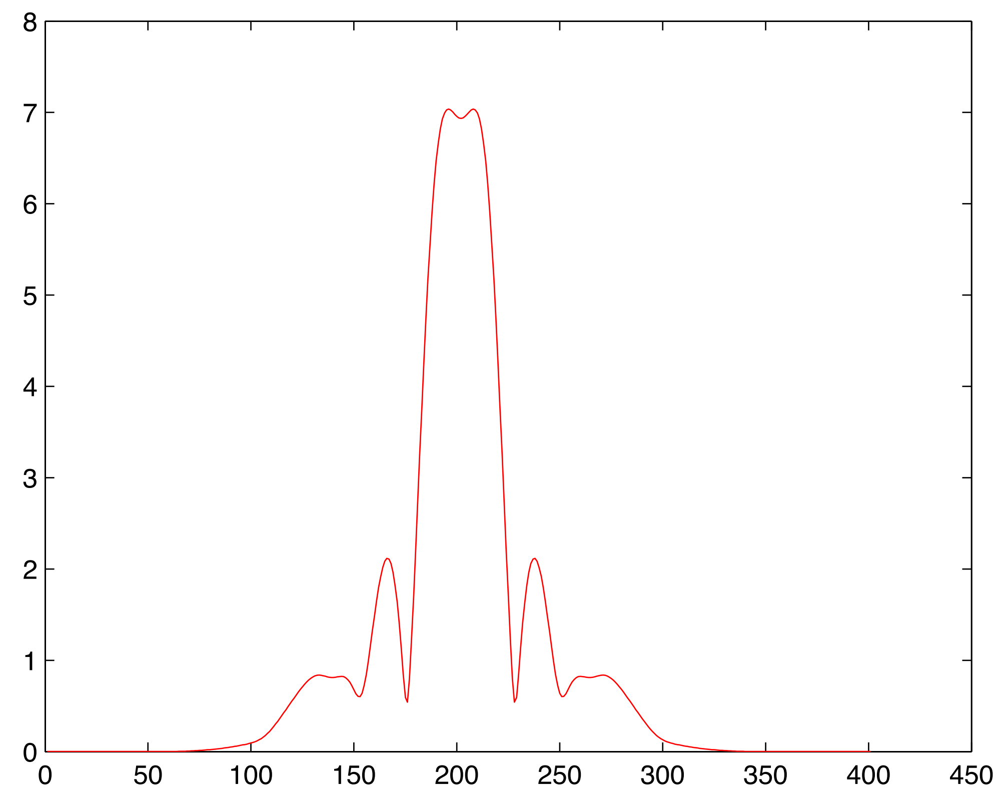


Spectral Networks

- In R^N, Smoothness and sparsity are dual notions:

$$x \text{ fast decay} \iff \widehat{x} \text{ smooth}$$

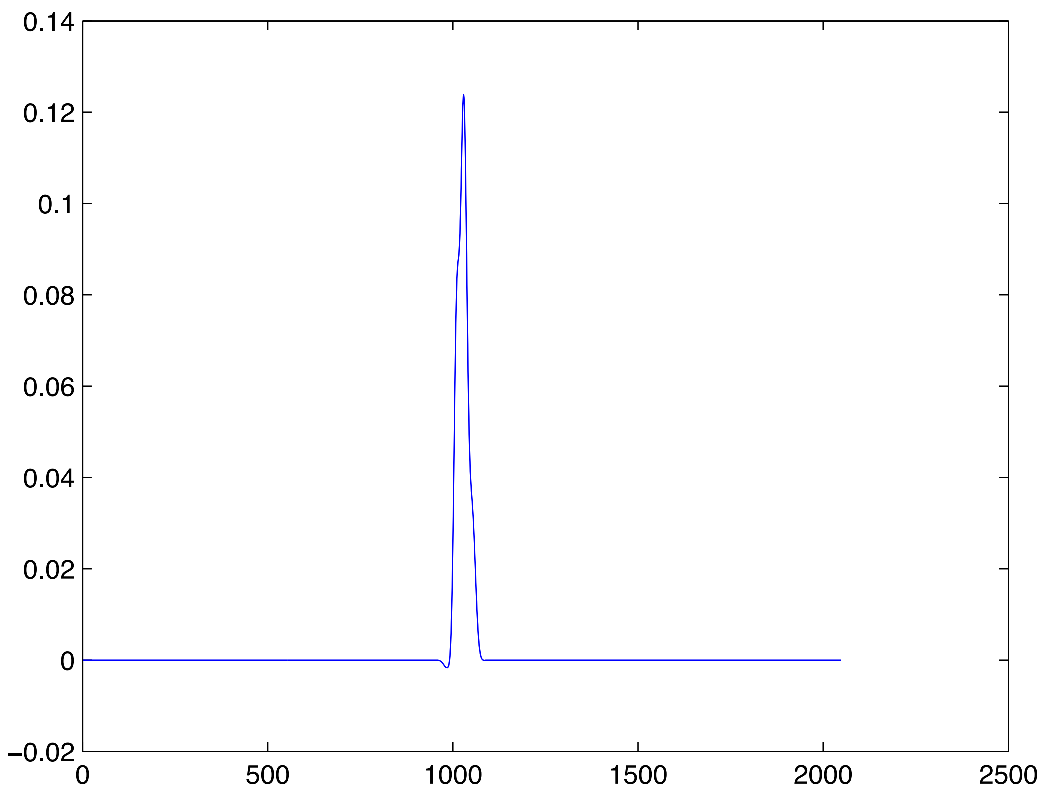


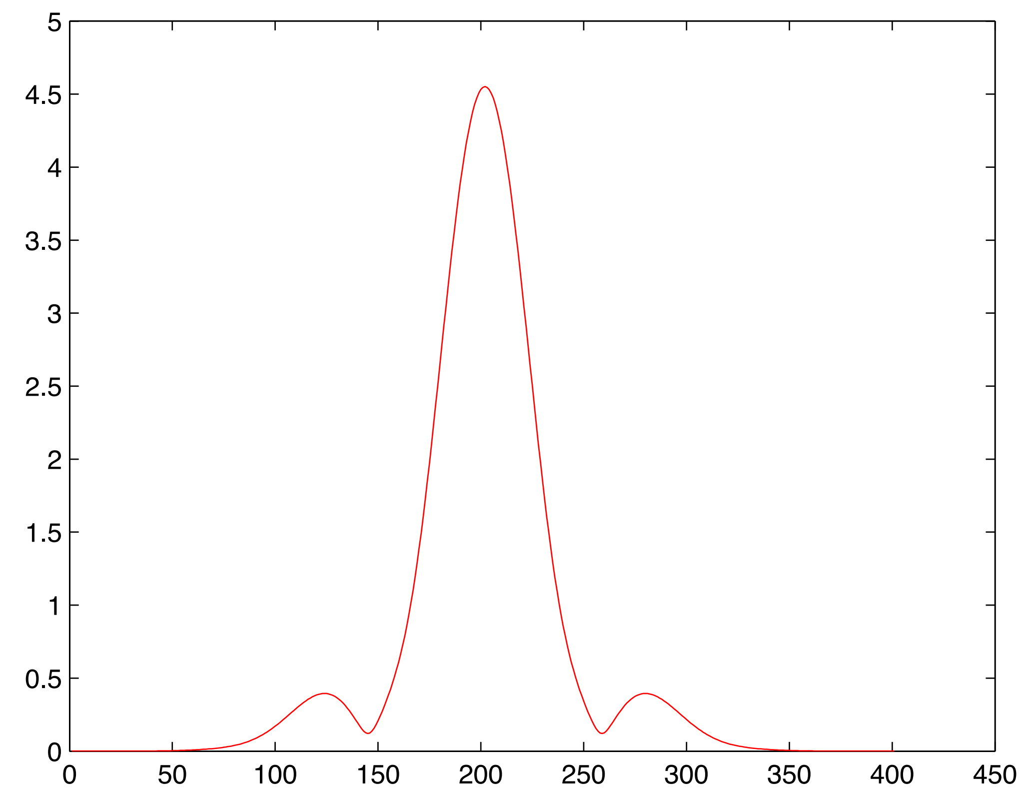


Spectral Networks

- In R^N, Smoothness and sparsity are dual notions:

$$x \text{ fast decay} \Longleftrightarrow \widehat{x} \text{ smooth}$$

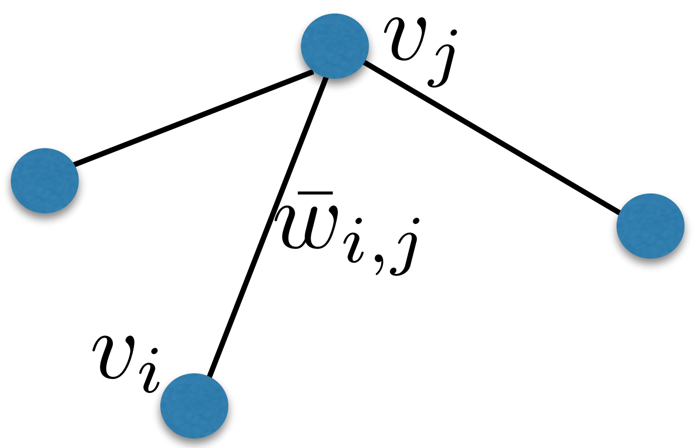




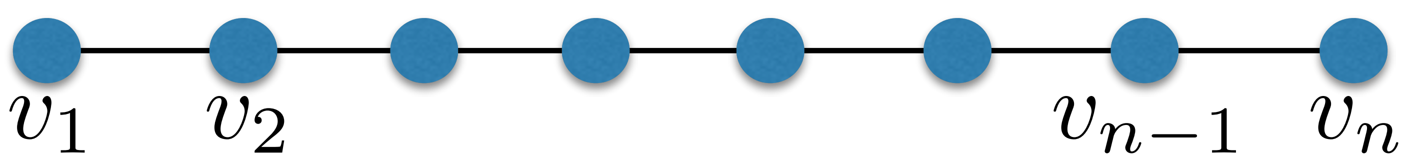
$$h = \mathcal{K}\tilde{h} \ , \ \mathcal{K} : \text{interpolation kernel (eg splines)}$$
$$\mathcal{K} \text{ size } n \times s, \text{ with } s \sim \text{spatial support size}$$

Dual Laplacian Graph

- Smoothness requires a notion of similarity between eigenvectors $V = [v_1 \dots v_n]$ of Laplacian



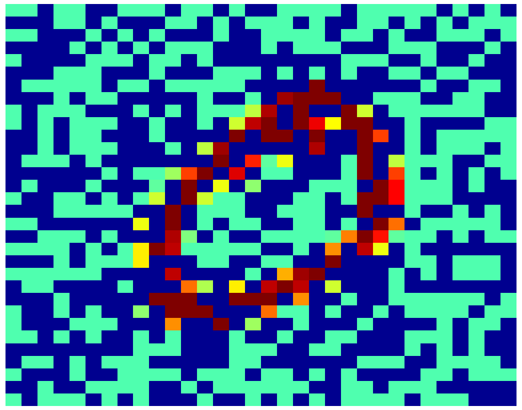
- Simplest Dual Geometry: 1D given by Spectrum

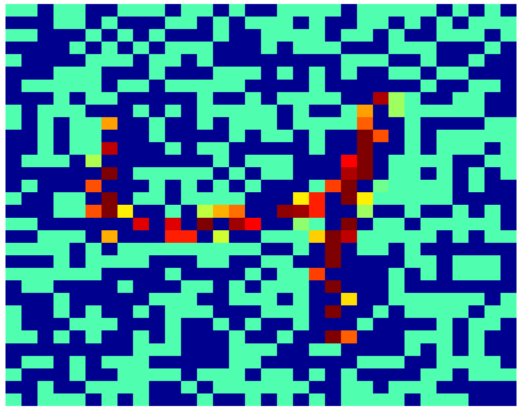


- General construction of dual graph: open problem.
  - Dual construction which enforces spatial decay?

Numerical Experiments

- MNIST random subsampling (400 pixels)

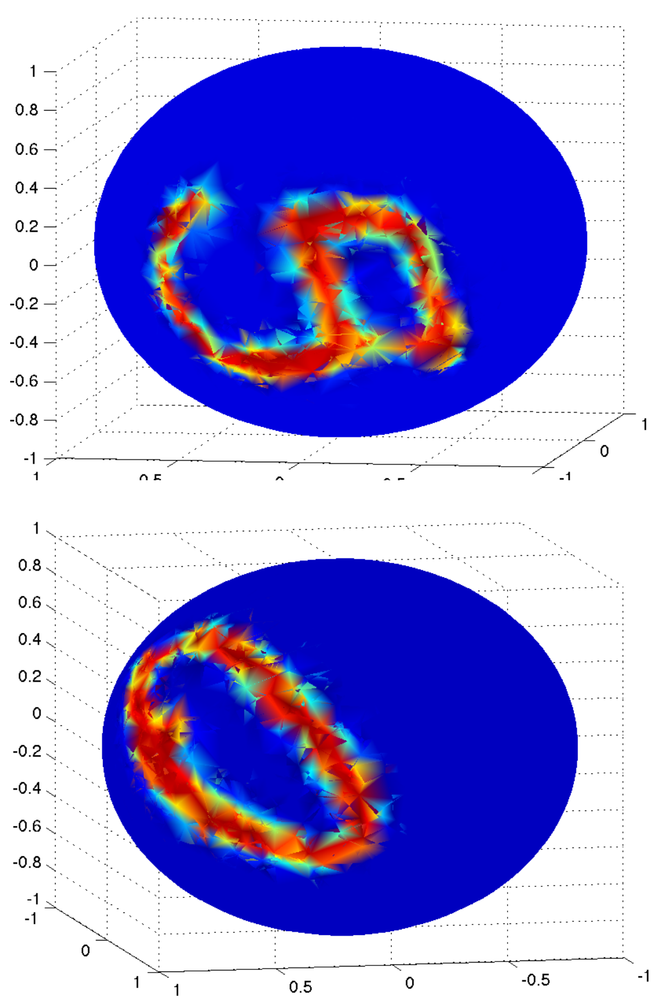


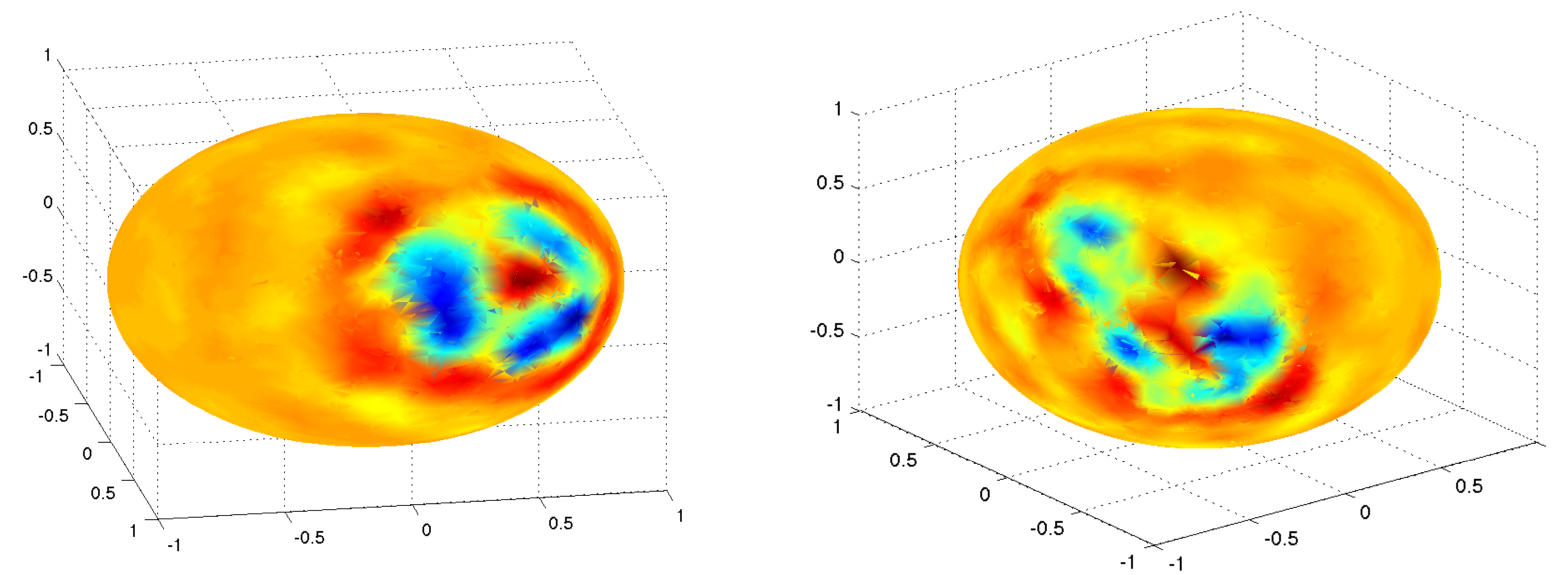


method	Parameters	Error
Nearest Neighbors	N/A	4.11
400-FC800-FC50-10	$3.6 \cdot 10^5$	1.8
400-LRF1600-MP800-10	$7.2 \cdot 10^4$	1.8
400-LRF3200-MP800-LRF800-MP400-10	$1.6 \cdot 10^5$	**1.3**
400-SP1600-10 $(d_1 = 300,\ q = n)$	$3.2 \cdot 10^3$	2.6
400-SP4800-10 $(d_1 = 300,\ q = 20)$	$5 \cdot 10^3$	1.8

Some Experiments

- MNIST projected onto 3D sphere:





learnt "feature maps"

method	Parameters	Error
Nearest Neighbors	N/A	19
4096-FC2048-FC512-9	10^7	**5.6**
4096-LRF4620-MP2000-FC300-9	$8 \cdot 10^5$	**6**
4096-LRF4620-MP2000-LRF500-MP250-9	$2 \cdot 10^5$	6.5
4096-SP32K-MP3000-FC300-9 $(q = n)$	$9 \cdot 10^5$	7
4096-SP32K-MP3000-FC300-9 $(q = 64)$	$9 \cdot 10^5$	**6**

Unknown Similarity

- When graph is unknown, how to estimate it?
  - Unsupervised: use data statistics (eg data covariance).
  - Supervised: use a simple network first and consider:

$$p(y \mid x) = S_m(W_2 \sigma(W_1 x))$$

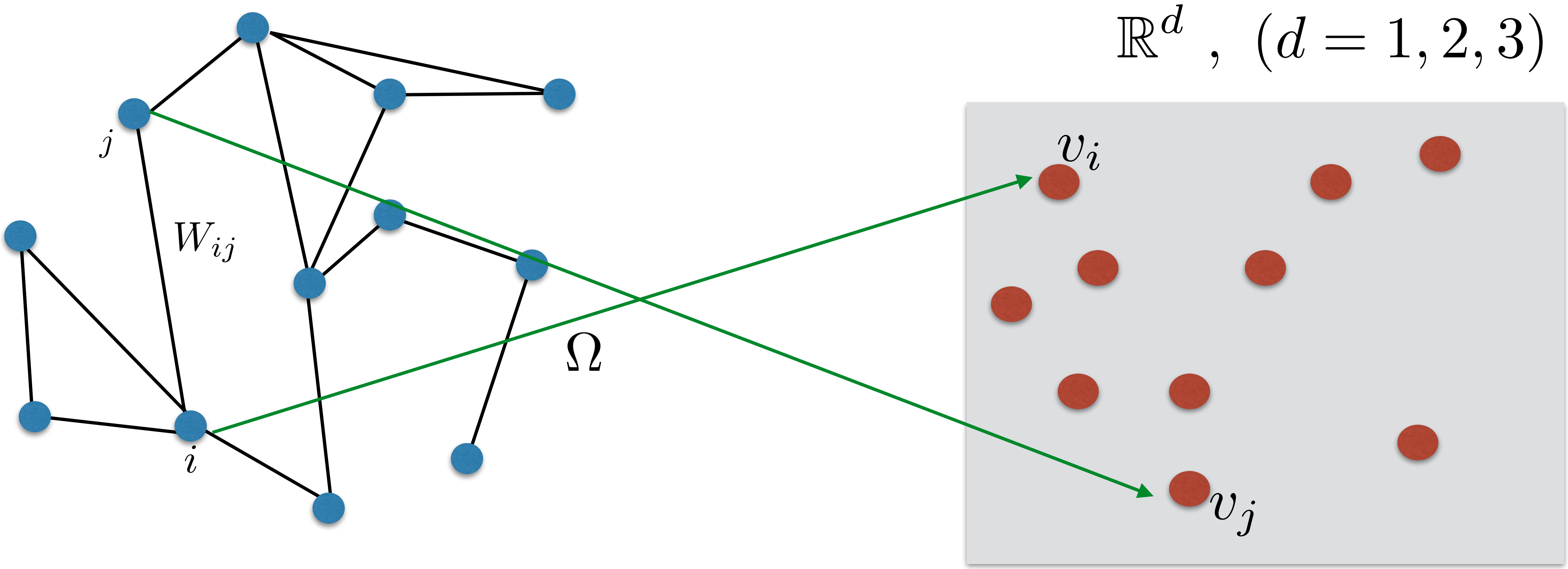
$$d(i,j) = \|W_{1,i} - W_{1,j}\|$$

- Small improvements over Dropout FC baseline:

Graph	Architecture	P_{net}	P_{graph}	R^2
-	FC4000-FC2000-FC1000-FC1000	$22.1 \cdot 10^6$	0	0.2729
Supervised	GC16-P4-GC16-P4-FC1000-FC1000	$3.8 \cdot 10^6$	$3.9 \cdot 10^6$	0.2773
Supervised	GC64-P8-GC64-P8-FC1000-FC1000	$3.8 \cdot 10^6$	$3.9 \cdot 10^6$	0.2580
RBF Kernel	GC64-P8-GC64-P8-FC1000-FC1000	$3.8 \cdot 10^6$	$3.9 \cdot 10^6$	0.2037
RBF Kernel (local)	GC64-P8-GC64-P8-FC1000-FC1000	$3.8 \cdot 10^6$	$3.9 \cdot 10^6$	0.1479

- However, computationally demanding: O(n^2).

Supervised Embedding



$$V=(v_1,\dots,v_N)\in\mathbb{R}^{d\times N}:\ \text{coordinate embedding}$$

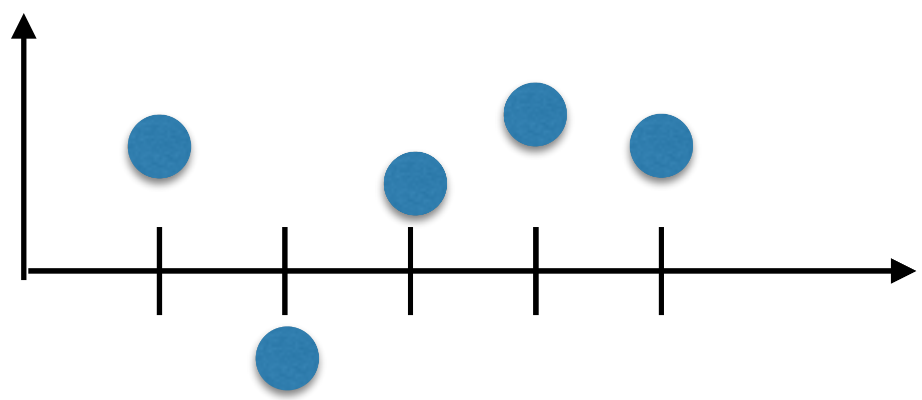
$$x\in L^1(G)\ \mapsto\ \tilde{x}\in L^1(\mathbb{R}^d)$$

$$\tilde{x}(u)=\sum_{i=1}^{N}x(i)\delta(u-v_i)$$

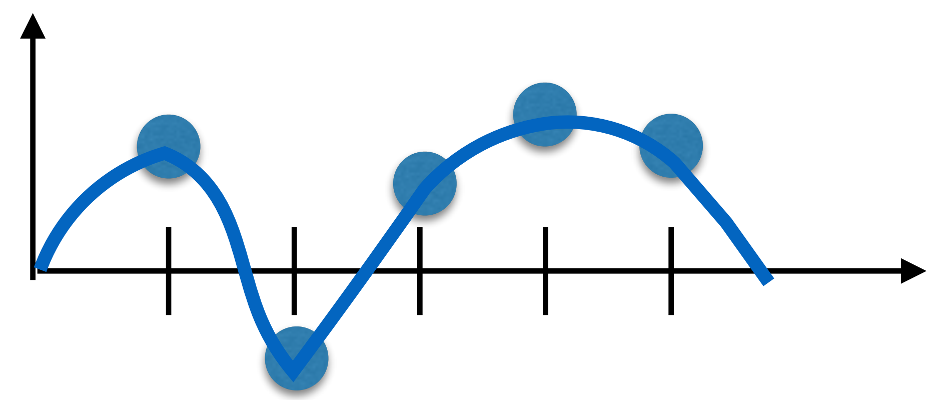
- How to define a convolution in an irregular sampling grid?

Supervised Embedding

- Given a compact support kernel ψ defined in a regular grid $\mathcal{G} \subset \mathbb{R}^d$, we extend it to $\mathbb{R}^d$ by interpolating:

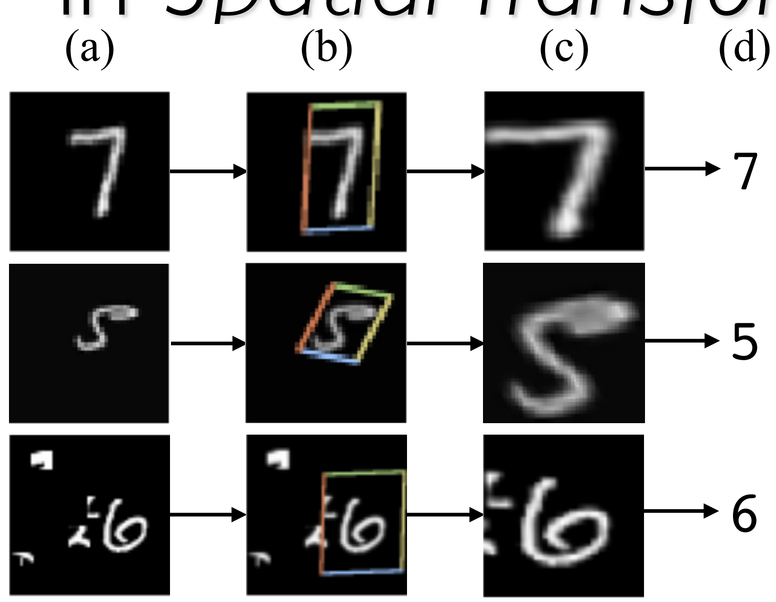


$$\psi(u) \;,\; u \in \mathcal{G}$$



$$\tilde{\psi}(\tilde{u}) \;,\; \tilde{u} \in \mathbb{R}$$

- This idea has also been used within CNN architectures, in *Spatial Transformer Networks*:



"Spatial Transformer Networks", Jaderberg et al,'15

Stationarity Prior

- Two clips. Goal: distinguish which is which.

clip1 clip2 clip ?

Stationarity Prior

- Same experiment. Goal: distinguish which is which.

clip3 clip4 clip ?

Stationarity Prior

- Same experiment. Goal: distinguish which is which.

clip3 clip4 clip ?

- Typically, the latter is harder. Reasons?

"Summary Statistics in auditory perception", McDermott & Simoncelli, Nature Neurosc.'13

Stationarity Prior

- Same experiment. Goal: distinguish which is which.

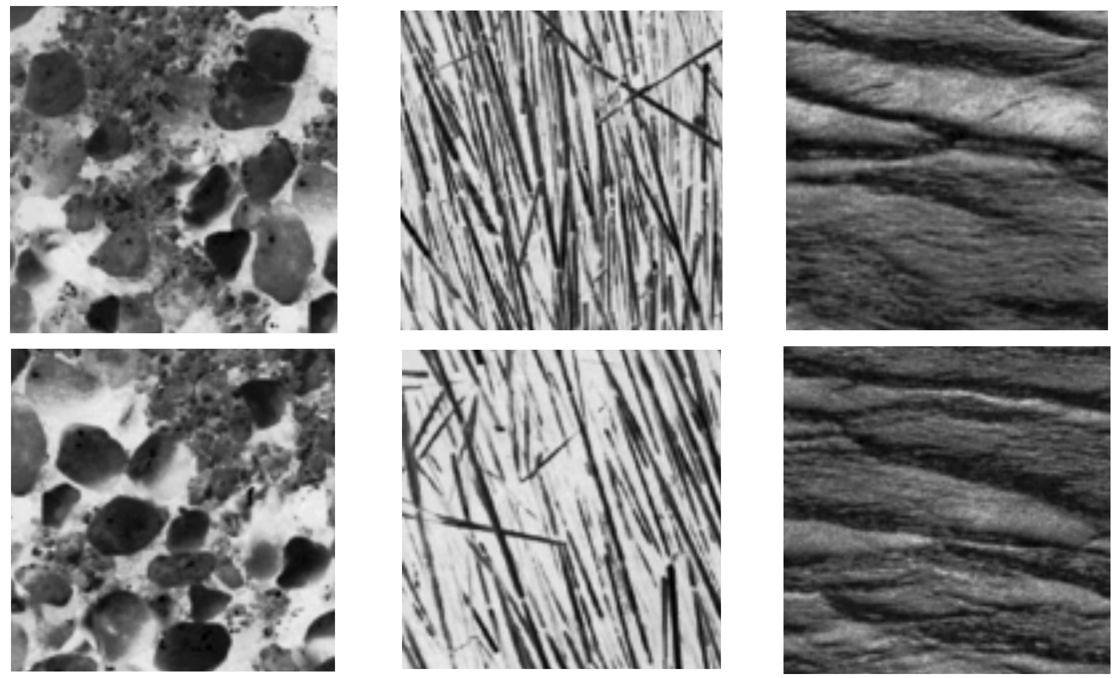
clip3 clip4 clip ?

- Typically, the latter is harder. Reasons?
- Despite having more information, the discrimination is worse because we construct temporal averages in presence of *stationary* inputs.

"Summary Statistics in auditory perception", McDermott & Simoncelli, Nature Neurosc.'13

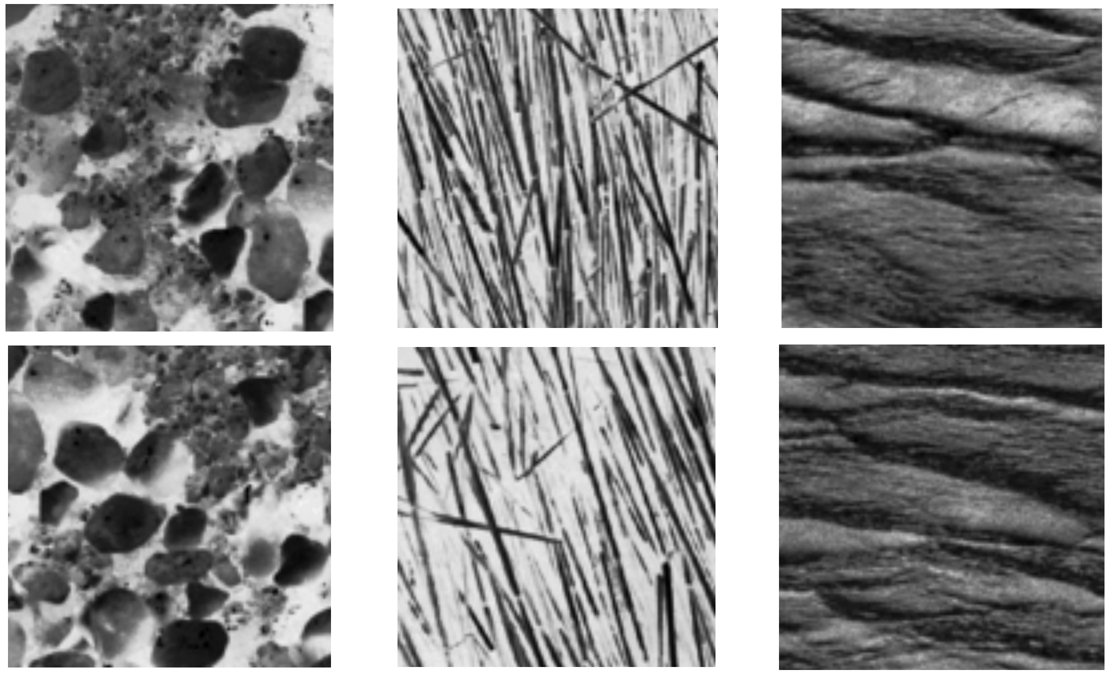
Representation of Stationary Processes

$x(u)$: realizations of a stationary process $X(u)$ (not Gaussian)



Representation of Stationary Processes

$x(u)$: realizations of a stationary process $X(u)$ (not Gaussian)



$$\Phi(X) = \{E(f_i(X))\}_i$$

Estimation from samples $x(n)$: $\widehat{\Phi}(X) = \left\{ \frac{1}{N} \sum_n f_i(x)(n) \right\}_i$

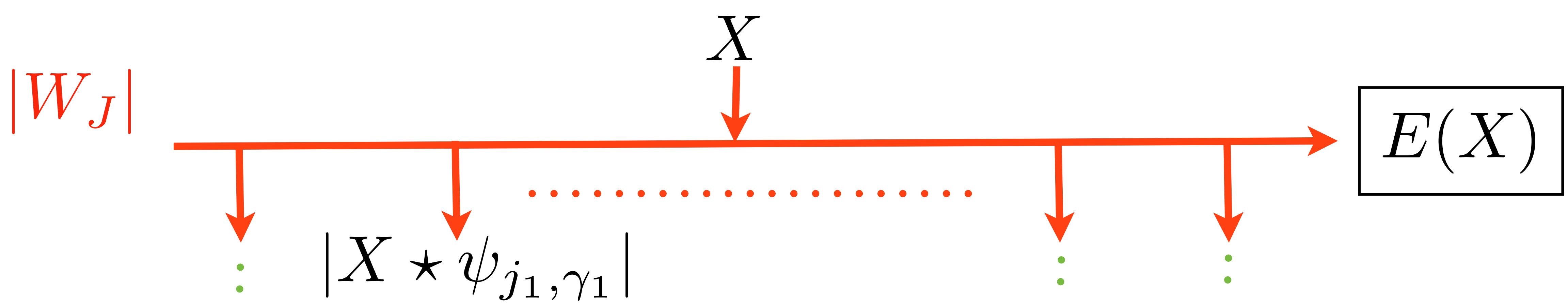
Discriminability: need to capture high-order moments

Stability: $E(\|\widehat{\Phi}(X) - \Phi(X)\|^2)$ small

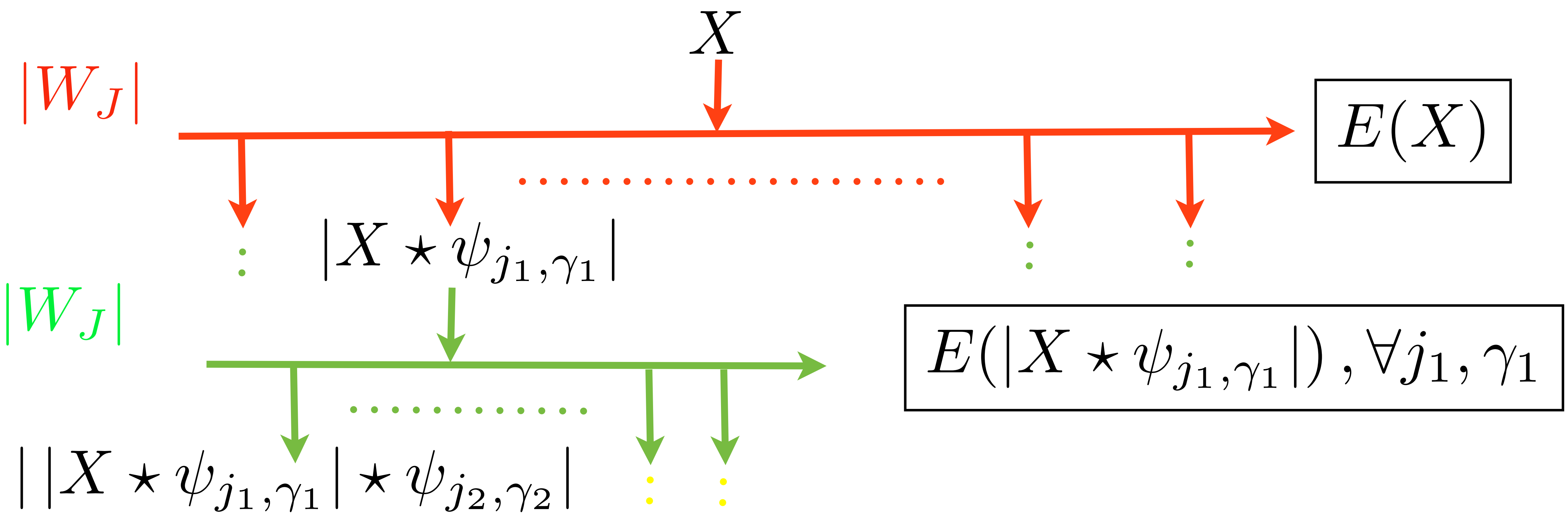
Scattering Moments

$$X$$

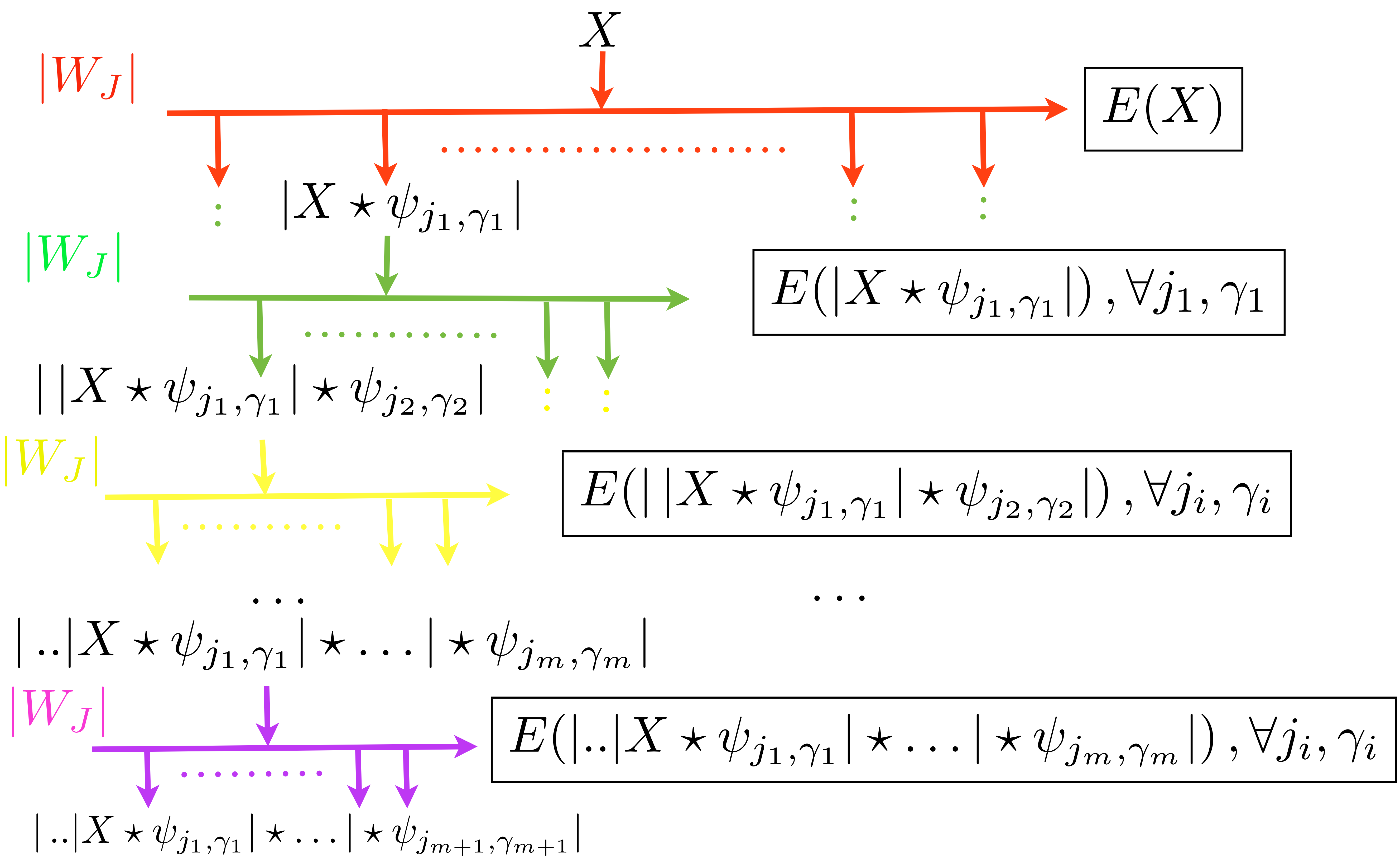
Scattering Moments



Scattering Moments



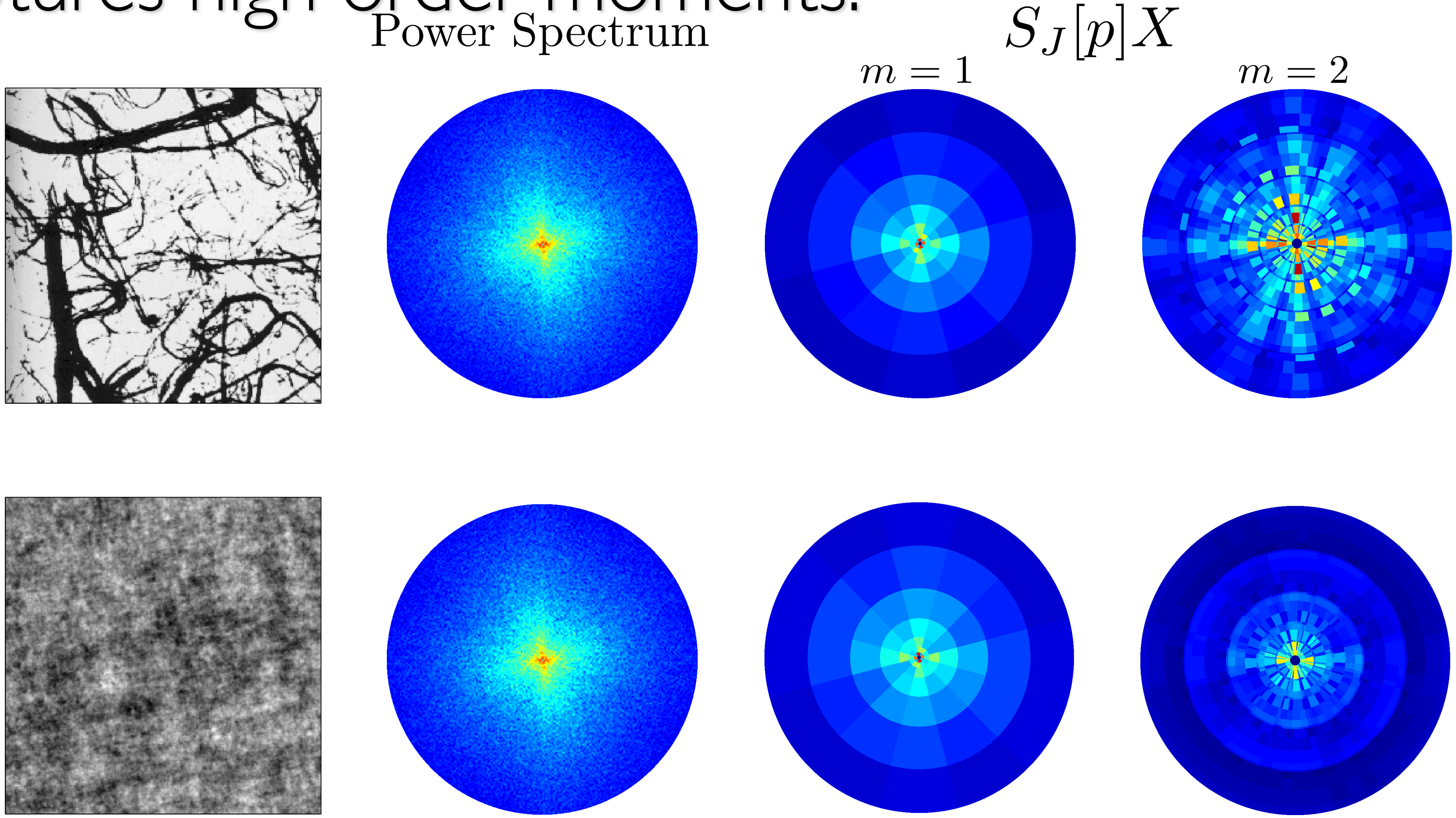
Scattering Moments



Properties of Scattering Moments

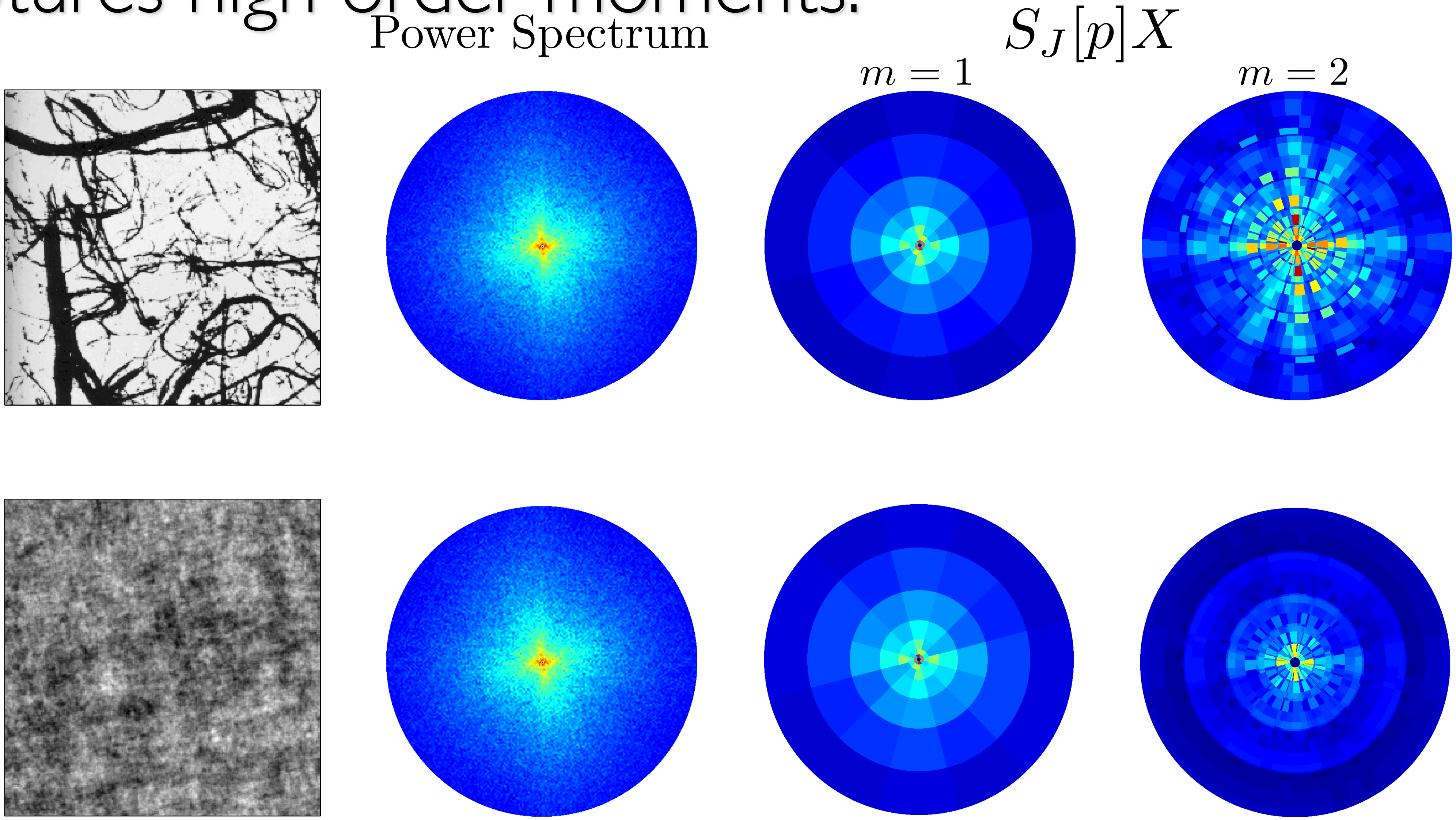
- Captures high order moments:

[Bruna, Mallat, '11,'12]



Properties of Scattering Moments

- Captures high order moments: [Bruna, Mallat, '11,'12]



- Cascading non-linearities is ***necessary*** to reveal higher-order moments.

Consistency of Scattering Moments

Theorem: [B'15] If ψ is a wavelet such that $\|\psi\|_1 \leq 1$, and $X(t)$ is a linear, stationary process with finite energy, then

$$\lim_{N \to \infty} E(\|\hat{S}_N X - SX\|^2) = 0 \ .$$

Consistency of Scattering Moments

Theorem: [B'15] If ψ is a wavelet such that $\|\psi\|_1 \leq 1$, and $X(t)$ is a linear, stationary process with finite energy, then

$$\lim_{N\to\infty} E(\|\hat{S}_N X - SX\|^2) = 0 \ .$$

Corollary: If moreover $X(t)$ is bounded, then

$$E(\|\hat{S}_N X - SX\|^2) \leq C \frac{|X|_\infty^2}{\sqrt{N}} \ .$$

- Although we extract a growing number of features, their global variance goes to 0.
- No variance blow-up due to high order moments.
- Adding layers is critical (here depth is log(N)).