Computer Perception With Deep Learning

Yann LeCun
Center for Data Science &
Courant Institute of Mathematical Sciences
New York University
yann@cs.nyu.edu http://yann.lecun.com

Deep Learning = Learning Representations/Features

- The traditional model of pattern recognition (since the late 50's)
  - Fixed/engineered features (or fixed kernel) + trainable classifier

hand-crafted Feature Extractor

“Simple” Trainable Classifier

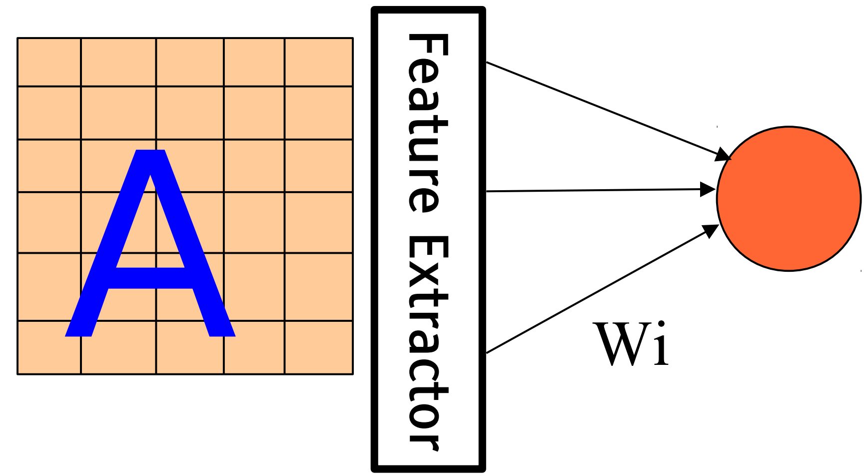
- End-to-end learning / Feature learning / Deep learning
  - Trainable features (or kernel) + trainable classifier

Trainable Feature Extractor

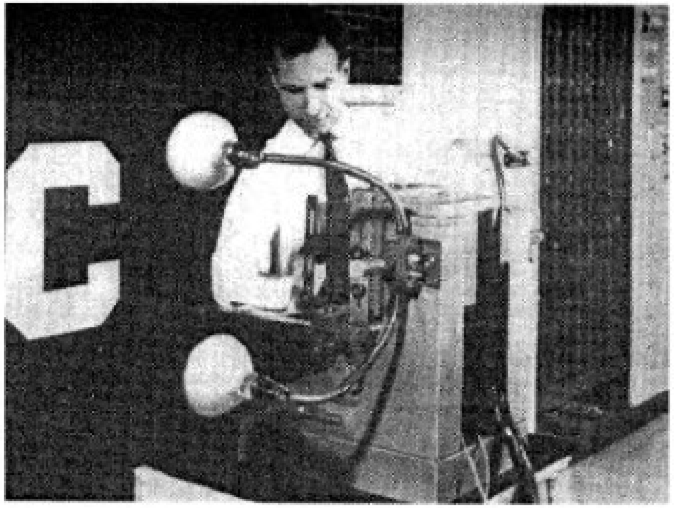
Trainable Classifier

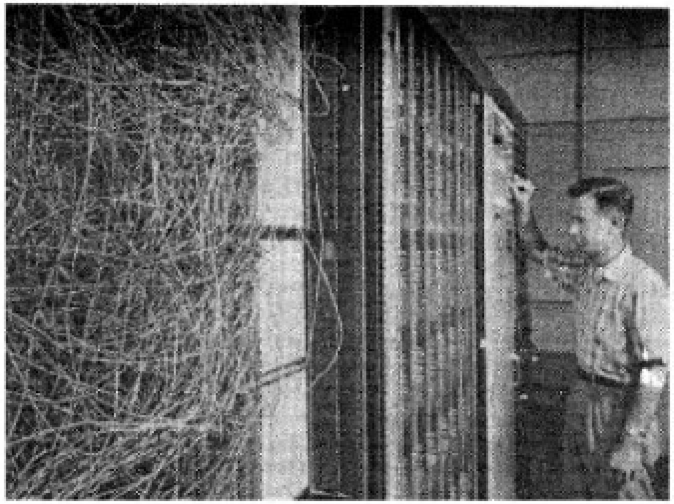
This Basic Model has not evolved much since the 50's

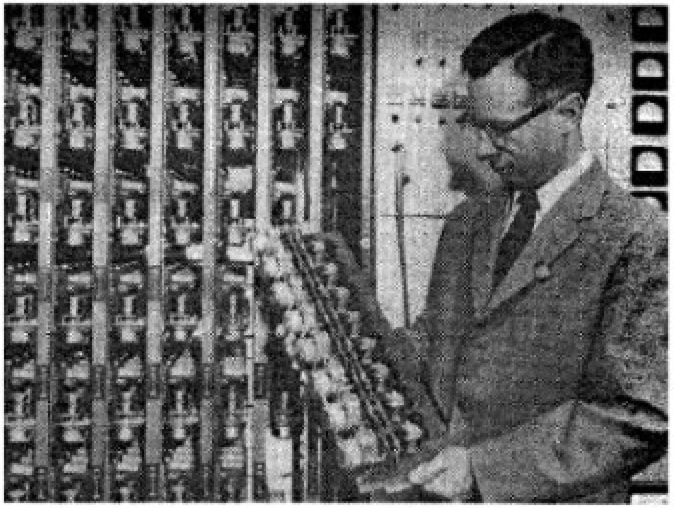
- The first learning machine: the Perceptron
  - Built at Cornell in 1960
- The Perceptron was a linear classifier on top of a simple feature extractor
- The vast majority of practical applications of ML today use glorified linear classifiers or glorified template matching.
- Designing a feature extractor requires considerable efforts by experts.



$$y = sign\left(\sum_{i=1}^{N} W_i F_i(X) + b\right)$$



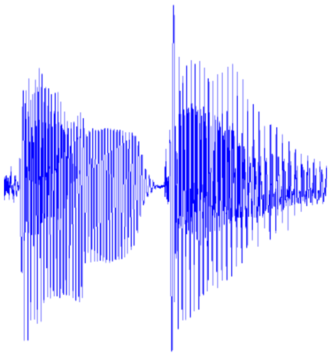




Architecture of "Mainstream" Pattern Recognition Systems

Modern architecture for pattern recognition

- Speech recognition: early 90's – 2011



MFCC → Mix of Gaussians → Classifier →

fixed | unsupervised | supervised

- Object Recognition: 2006 - 2012

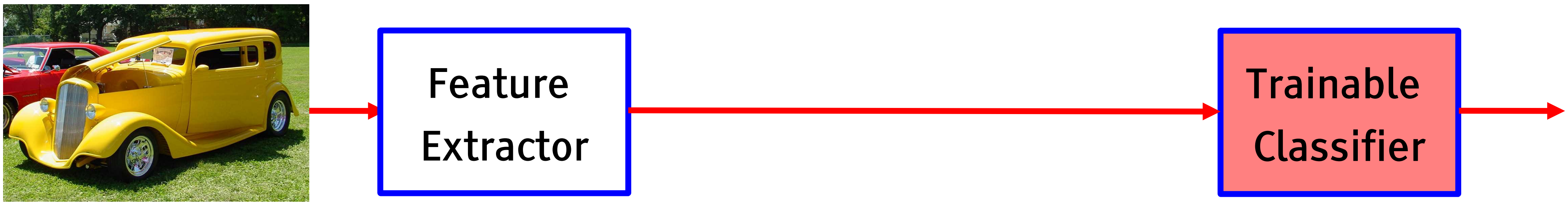
SIFT HoG → K-means Sparse Coding → Pooling → Classifier →

fixed | unsupervised | supervised

Low-level Features | Mid-level Features

Deep Learning = Learning Representations/Features

- Traditional Pattern Recognition: Fixed/Handcrafted Feature Extractor

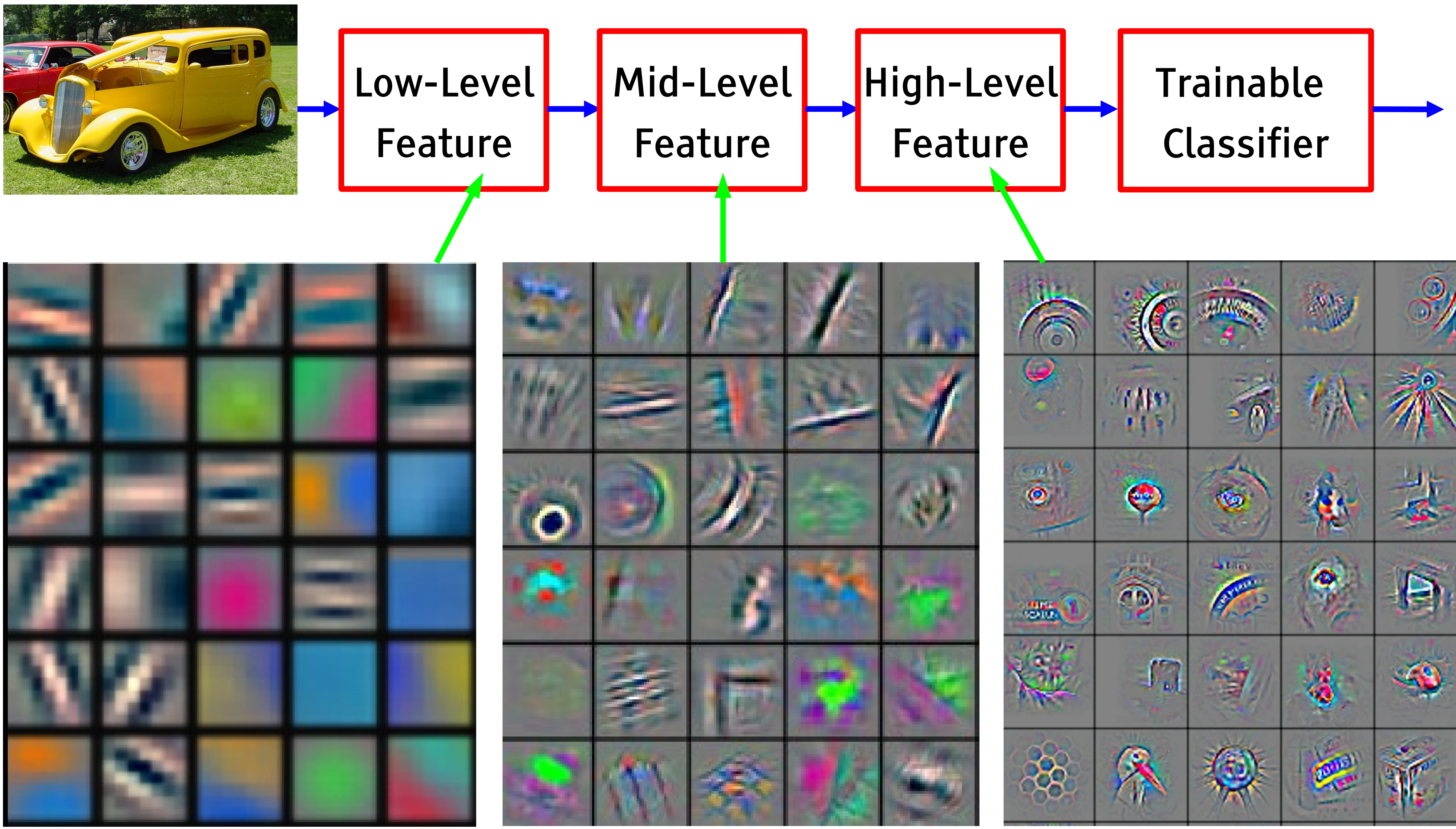


- Mainstream Modern Pattern Recognition: Unsupervised mid-level features

- Deep Learning: Representations are hierarchical and trained

Deep Learning = Learning Hierarchical Representations

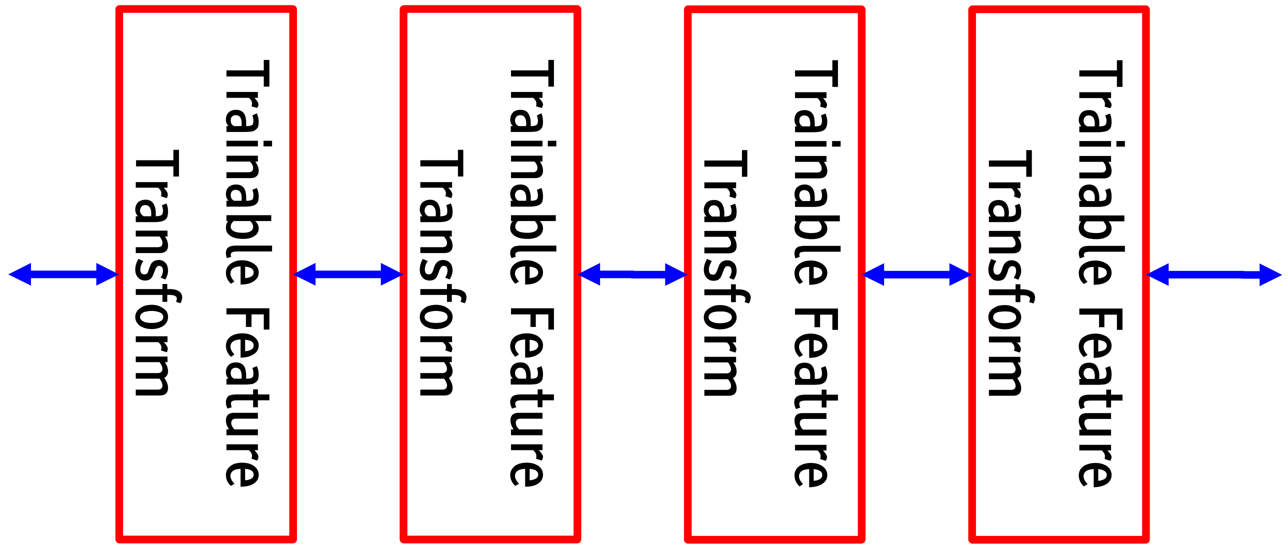
- It's deep if it has more than one stage of non-linear feature transformation



Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

Trainable Feature Hierarchy

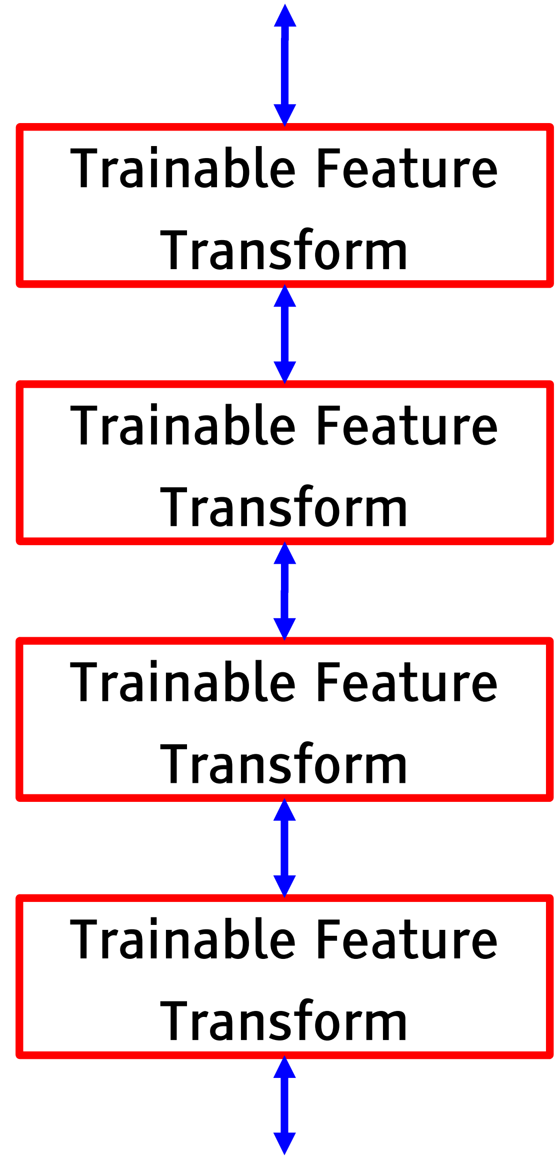
- Hierarchy of representations with increasing level of abstraction
- Each stage is a kind of trainable feature transform
- Image recognition
  - Pixel → edge → texton → motif → part → object
- Text
  - Character → word → word group → clause → sentence → story
- Speech
  - Sample → spectral band → sound → … → phone → phoneme → word



Learning Representations: a challenge for ML, CV, AI, Neuroscience, Cognitive Science...

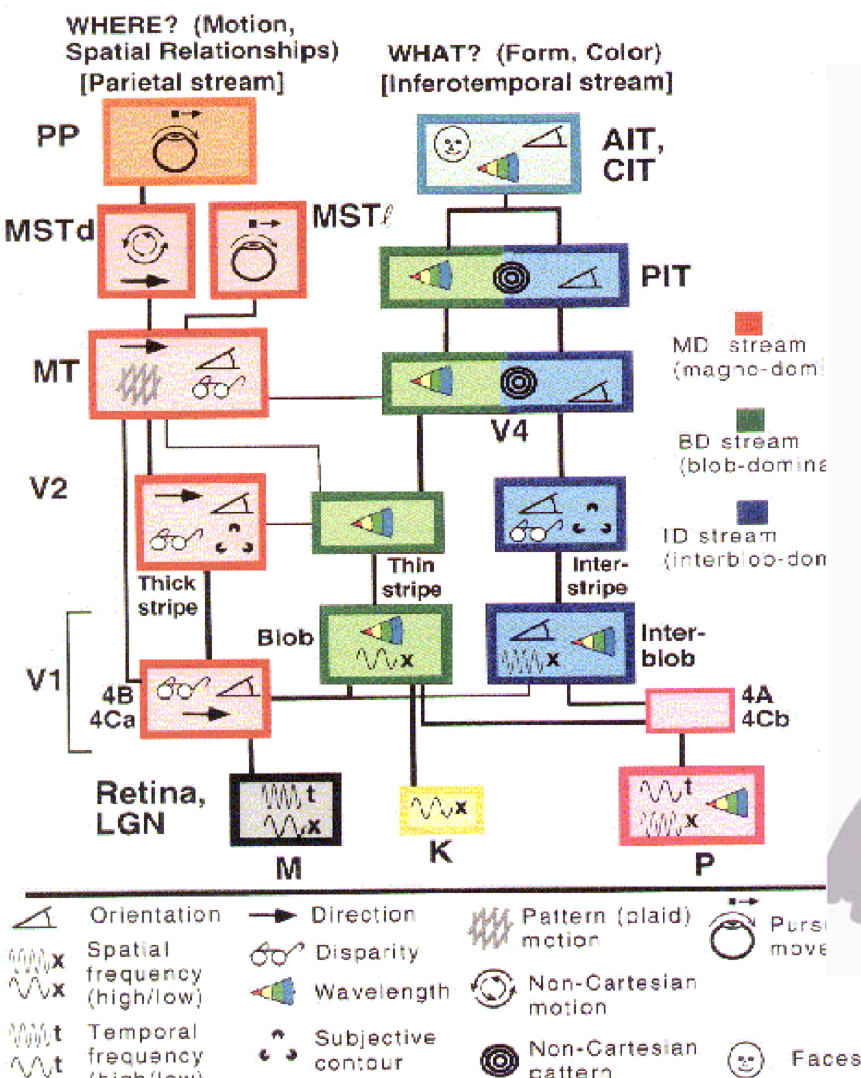
- How do we learn representations of the perceptual world?
  - How can a perceptual system build itself by looking at the world?
  - How much prior structure is necessary
- ML/AI: how do we learn features or feature hierarchies?
  - What is the fundamental principle? What is the learning algorithm? What is the architecture?
- Neuroscience: how does the cortex learn perception?
  - Does the cortex “run” a single, general learning algorithm? (or a small number of them)
- CogSci: how does the mind learn abstract concepts on top of less abstract ones?

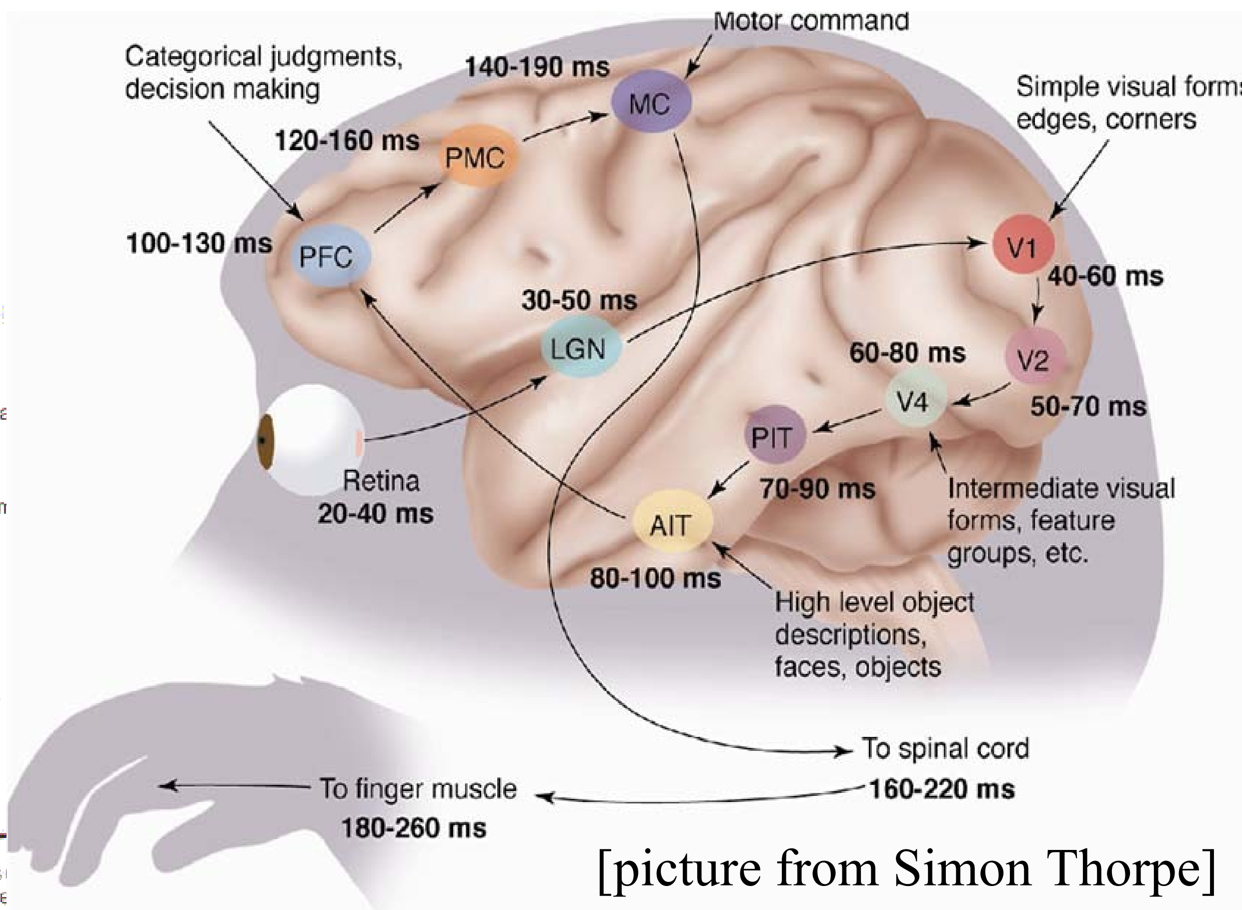
- Deep Learning addresses the problem of learning hierarchical representations with a single algorithm
  - or perhaps with a few algorithms



The Mammalian Visual Cortex is Hierarchical

- The ventral (recognition) pathway in the visual cortex has multiple stages
- Retina - LGN - V1 - V2 - V4 - PIT - AIT ....
- Lots of intermediate representations





[picture from Simon Thorpe]

[Gallant & Van Essen]

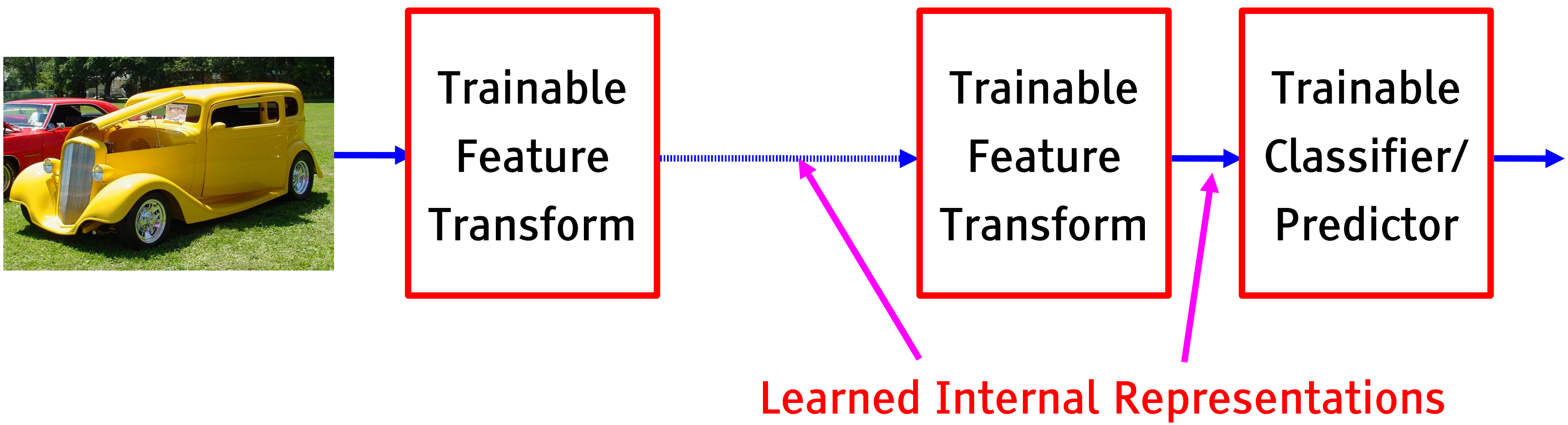
Let's be inspired by nature, but not too much

- It's nice imitate Nature,
- But we also need to understand
  - How do we know which details are important?
  - Which details are merely the result of evolution, and the constraints of biochemistry?
- For airplanes, we developed aerodynamics and compressible fluid dynamics.
  - We figured that feathers and wing flapping weren't crucial
- QUESTION: What is the equivalent of aerodynamics for understanding intelligence?

L'Avion III de Clément Ader, 1897
(Musée du CNAM, Paris)
His "Eole" took off from the ground in 1890,
13 years before the Wright Brothers, but you
probably never heard of it (unless you are french).

Trainable Feature Hierarchies: End-to-end learning

- A hierarchy of trainable feature transforms
  - Each module transforms its input representation into a higher-level one.
  - High-level features are more global and more invariant
  - Low-level features are shared among categories



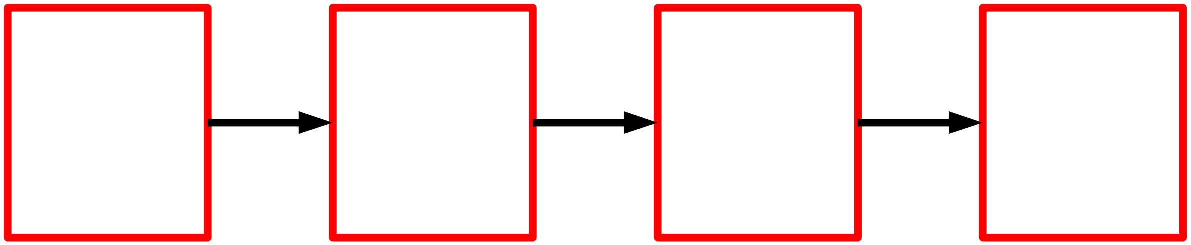
- How can we make all the modules trainable and get them to learn appropriate representations?

Deep Learning is Inevitable for Three Reasons

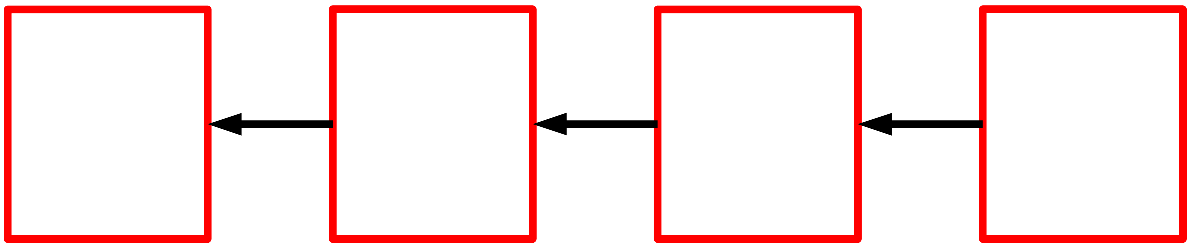
- There is more data being produced than all the human brains on the planet can process (let alone design models for)
- 1. Building traditional models is expensive
  - designing feature extractors is long, painful and expensive
  - Industry needs to build more and more models
  - The process must be automated → Deep Learning
- 2. Computers power is increasing, data size in increasing
  - Learning algorithms are already better than humans at “designing” models from data
  - It can only get better as machine become more powerful
  - Human-based design doesn't scale!
- 3. It is the direction of history
  - The history of pattern recognition/AI in the last decades show a clear motion away from “hand engineering” and towards machine learning.
- Soon, most of the knowledge in the world will have to be derived by machines

Three Types of Deep Architectures

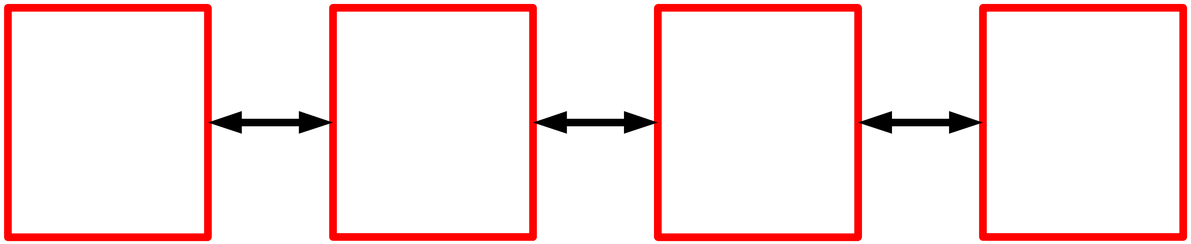
- **Feed-Forward**: multilayer neural nets, convolutional nets



- **Feed-Back**: Stacked Sparse Coding, Deconvolutional Nets [Zeiler et al.]



- **Bi-Drectional**: Deep Boltzmann Machines, Stacked Auto-Encoders



Three Types of Training Protocols

- Purely Supervised
  - Initialize parameters randomly
  - Train in supervised mode
    - typically with SGD, using backprop to compute gradients
  - Used in most practical systems for speech and image recognition
- Unsupervised, layerwise + supervised classifier on top
  - Train each layer unsupervised, one after the other
  - Train a supervised classifier on top, keeping the other layers fixed
  - Good when very few labeled samples are available
- Unsupervised, layerwise + global supervised fine-tuning
  - Train each layer unsupervised, one after the other
  - Add a classifier layer, and retrain the whole thing supervised
  - Good when label set is poor (e.g. pedestrian detection)
- Unsupervised pre-training often uses regularized auto-encoders

Do we really need deep architectures?

- Theoretician's dilemma: "We can approximate any function as close as we want with shallow architecture. Why would we need deep ones?"

$$y = \sum_{i=1}^{P} \alpha_i K(X, X^i) \qquad y = F(W^1.F(W^0.X))$$

  - kernel machines (and 2-layer neural nets) are "universal".
- Deep learning machines

$$y = F(W^K.F(W^{K-1}.F(....F(W^0.X)...)))$$

- Deep machines are more efficient for representing certain classes of functions, particularly those involved in visual recognition
  - they can represent more complex functions with less "hardware"
- We need an efficient parameterization of the class of functions that are useful for "AI" tasks (vision, audition, NLP...)

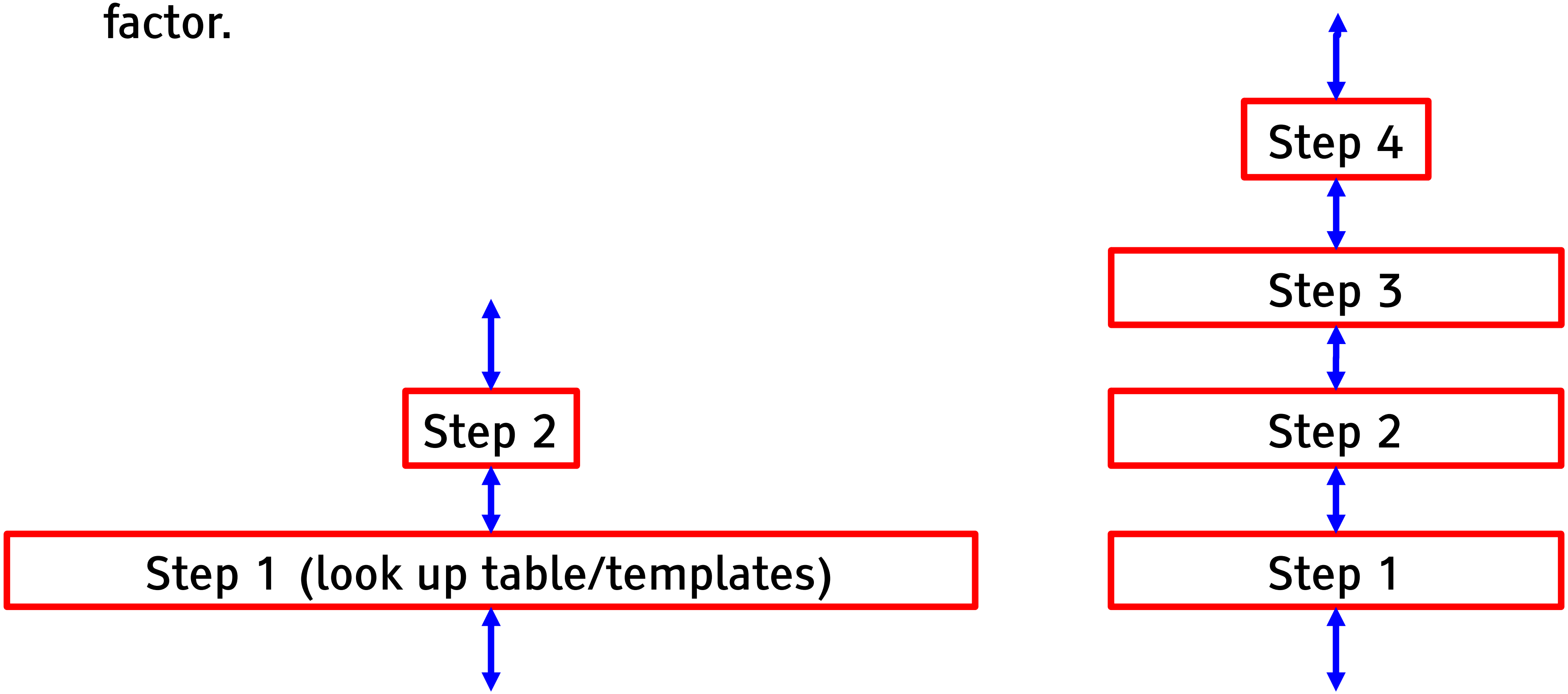
Why would deep architectures be more efficient?

[Bengio & LeCun 2007 “Scaling Learning Algorithms Towards AI”]

- A deep architecture trades space for time (or breadth for depth)
  - more layers (more sequential computation),
  - but less hardware (less parallel computation).
- Example1: N-bit parity
  - requires N-1 XOR gates in a tree of depth log(N).
  - Even easier if we use threshold gates
  - requires an exponential number of gates of we restrict ourselves to 2 layers (DNF formula with exponential number of minterms).
- Example2: circuit for addition of 2 N-bit binary numbers
  - Requires O(N) gates, and O(N) layers using N one-bit adders with ripple carry propagation.
  - Requires lots of gates (some polynomial in N) if we restrict ourselves to two layers (e.g. Disjunctive Normal Form).
  - Bad news: almost all boolean functions have a DNF formula with an exponential number of minterms O(2^N).....

Shallow vs Deep == lookup table vs multi-step algorithm

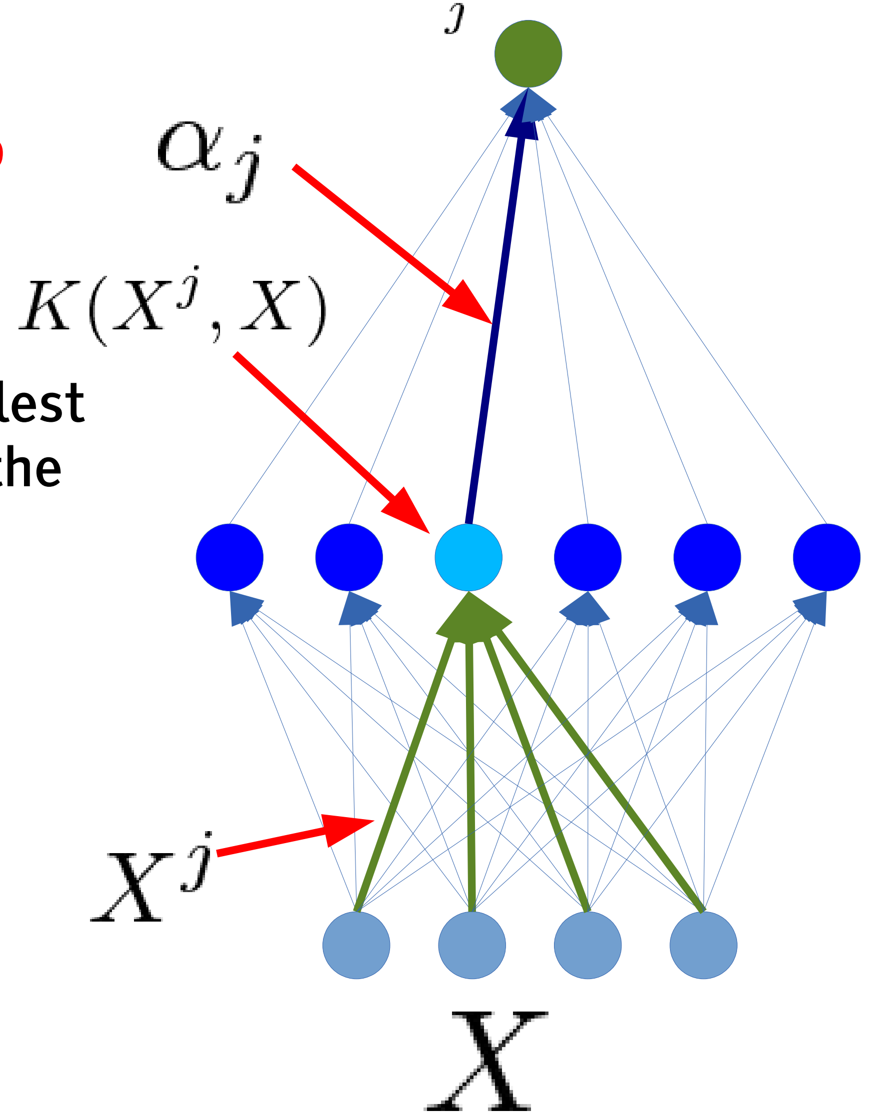
- "shallow & wide" vs "deep and narrow" == "more memory" vs "more time"
  - Look-up table vs algorithm
  - Few functions can be computed in two steps without an exponentially large lookup table
  - Using more than 2 steps can reduce the "memory" by an exponential factor.



Which Models are Deep?

- 2-layer models are not deep (even if you train the first layer)
  - Because there is no feature hierarchy
- Neural nets with 1 hidden layer are not deep
- SVMs and Kernel methods are not deep
  - Layer1: kernels; layer2: linear
  - The first layer is “trained” in with the simplest unsupervised method ever devised: using the samples as templates for the kernel functions.
  - “glorified template matching”
- Classification trees are not deep
  - No hierarchy of features. All decisions are made in the input space

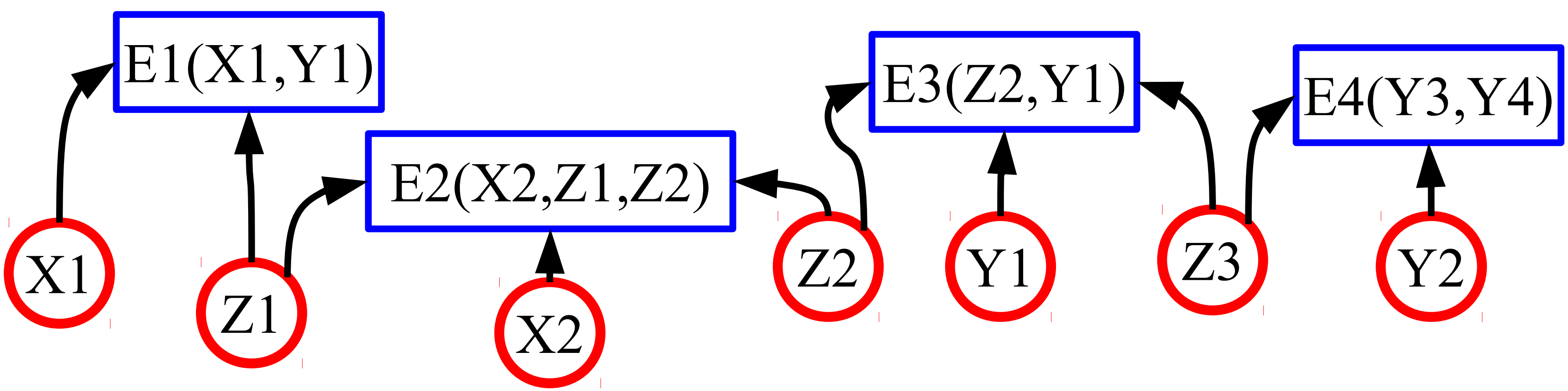
$$G(X,\alpha)=\sum_j \alpha_j K(X^j,X)$$



Are Graphical Models Deep?

- There is no opposition between graphical models and deep learning.
  - Many deep learning models are formulated as factor graphs
  - Some graphical models use deep architectures inside their factors
- Graphical models can be deep (but most are not).
- Factor Graph: sum of energy functions
  - Over inputs X, outputs Y and latent variables Z. Trainable parameters: W

$$-\log P(X,Y,Z/W) \propto E(X,Y,Z,W) = \sum_i E_i(X,Y,Z,W_i)$$



- Each energy function can contain a deep network
- The whole factor graph can be seen as a deep network

Deep Learning: A Theoretician's Nightmare?

- Deep Learning involves non-convex loss functions
  - With non-convex losses, all bets are off
  - Then again, every speech recognition system ever deployed has used non-convex optimization (GMMs are non convex).
- But to some of us all “interesting” learning is non convex
  - Convex learning is invariant to the order in which sample are presented (only depends on asymptotic sample frequencies).
  - Human learning isn't like that: we learn simple concepts before complex ones. The order in which we learn things matter.

Deep Learning: A Theoretician's Nightmare?

- No generalization bounds?
  - Actually, the usual VC bounds apply: most deep learning systems have a finite VC dimension
  - We don't have tighter bounds than that.
  - But then again, how many bounds are tight enough to be useful for model selection?
- It's hard to prove anything about deep learning systems
  - Then again, if we only study models for which we can prove things, we wouldn't have speech, handwriting, and visual object recognition systems today.

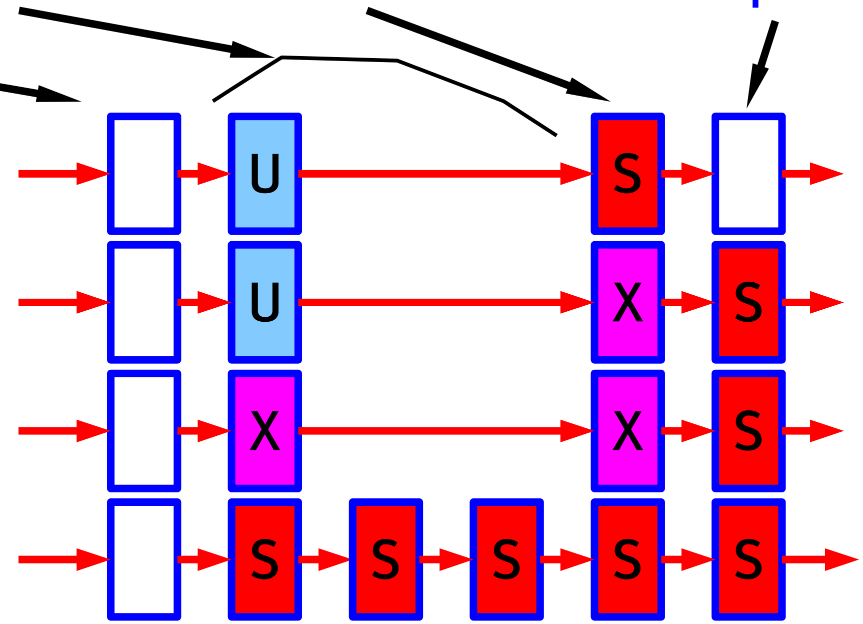
Deep Learning and Feature Learning Today

- Deep Learning has been the hottest topic in speech recognition in the last 2 years
  - A few long-standing performance records were broken with deep learning methods
  - Google, Baidu, IBM and Microsoft have have deployed DL-based speech recognition system in their products (many use convolutional networks)
  - All the major academic and industrial players in speech recognition have projects on deep learning
- Deep Learning is becoming the hottest topic in Computer Vision
  - Feature engineering is the bread-and-butter of a large portion of the CV community, which creates some resistance to feature learning
  - But the record holders on ImageNet and Semantic Segmentation are convolutional nets
- Deep Learning is becoming hot in Natural Language Processing
- Deep Learning/Feature Learning in Applied Mathematics
  - The connection with Applied Math is through sparse coding, non-convex optimization, stochastic gradient algorithms, etc...

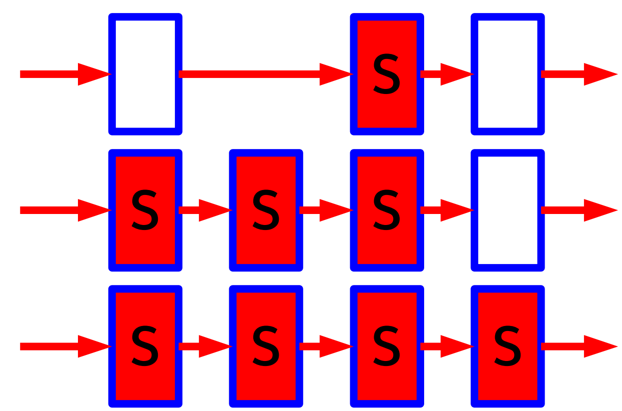
In Several Fields, Feature Learning Has Caused Revolutions: Speech Recognition, Handwriting Recogntiion

- U= unsupervised, S=supervised, X=unsupervised+supervised
- Low-level feat. → mid-level feat. → classifier → contextual post-proc

- Speech Recognition
  - Early 1980s: Dyn. time Warping
  - Late 1980s: Gaussian Mix. Model
  - 1990s: discriminative GMM
  - 2010: deep neural nets

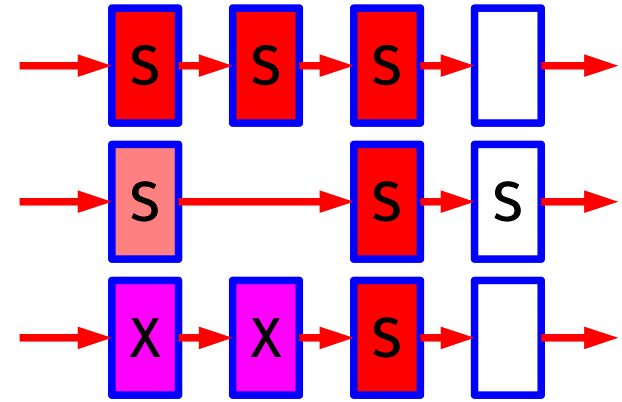


- Handwriting Recognition and OCR
  - Early 80's: features+classifier
  - Late 80's: supervised convnet
  - Mid 90's: convnet+CRF

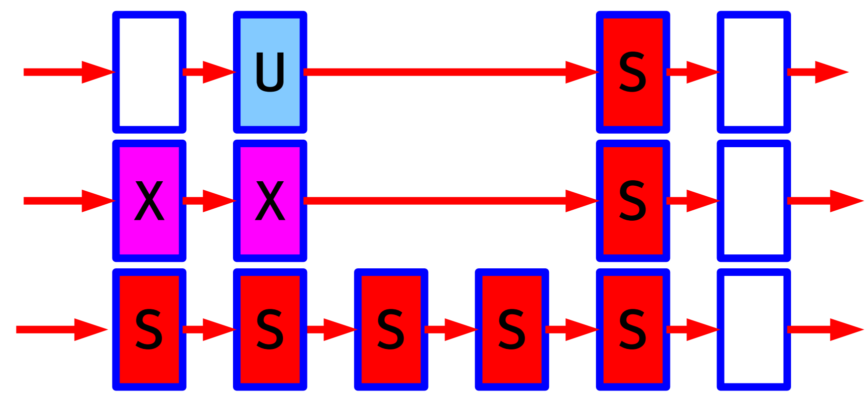


In Several Fields, Feature Learning Has Caused Revolutions: Object Detection, Object Recognition, Scene Labeling

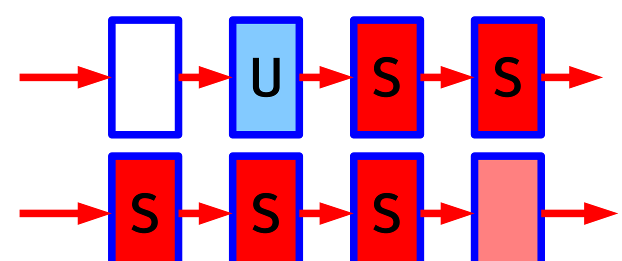
- **Face & People Detection (1993-now)**
  - Supervised ConvNet on pixels (93, 94, 05, 07)
  - Selected Haar features + Adaboost (2001)
  - Unsup+Sup ConvNet on raw pixels (2011)



- **Object Recognition**
  - SIFT/HoG+sparse code+pool+SVM (06)
  - unsup+sup convnet (07,10)
  - supervised convnet (2012)



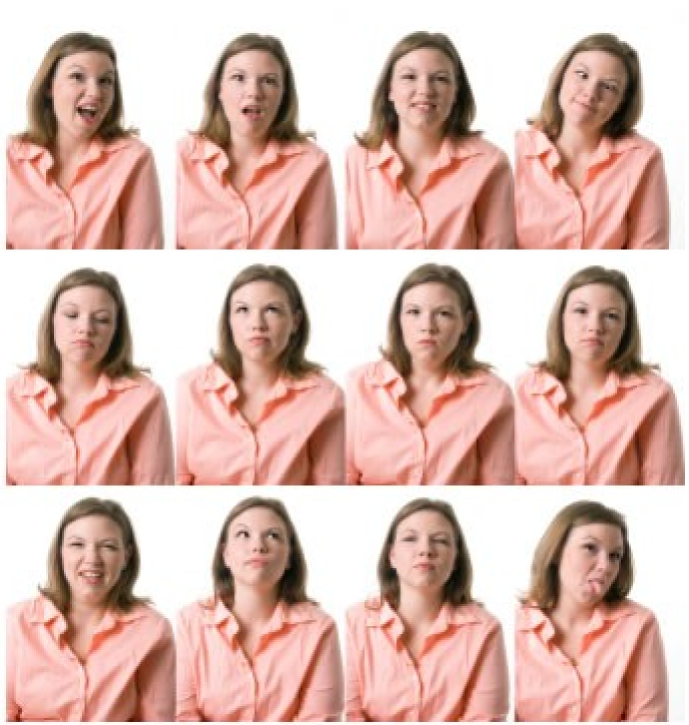
- **Semantic Segmentation / scene labeling**
  - unsup mid-lvl, CRF (2009, 10, 11, 12)
  - supervised convnet (2008, 12, 13)

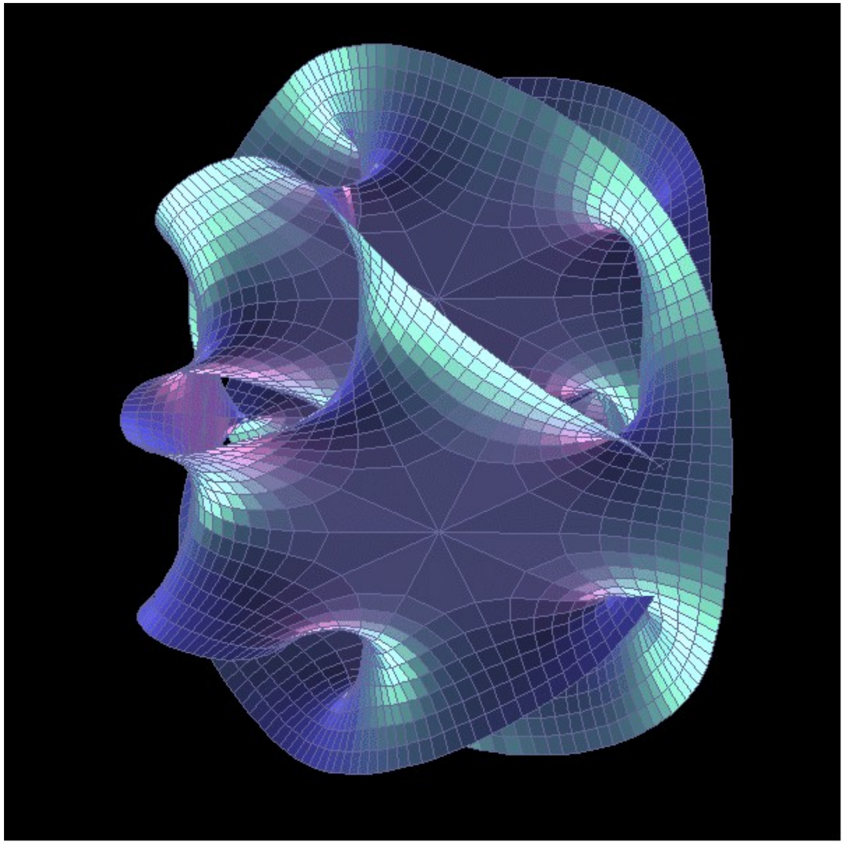


What Are
Good Feature?

Discovering the Hidden Structure in High-Dimensional Data
The manifold hypothesis

- Learning Representations of Data:
  - Discovering & disentangling the independent explanatory factors
- The Manifold Hypothesis:
  - Natural data lives in a low-dimensional (non-linear) manifold
  - Because variables in natural data are mutually dependent





Discovering the Hidden Structure in High-Dimensional Data

- Example: all face images of a person
  - 1000x1000 pixels = 1,000,000 dimensions
  - But the face has 3 cartesian coordinates and 3 Euler angles
  - And humans have less than about 50 muscles in the face
  - Hence the manifold of face images for a person has <56 dimensions
- The perfect representations of a face image:
  - Its coordinates on the face manifold
  - Its coordinates away from the manifold
- We do not have good and general methods to learn functions that turns an image into this kind of representation

Ideal Feature Extractor

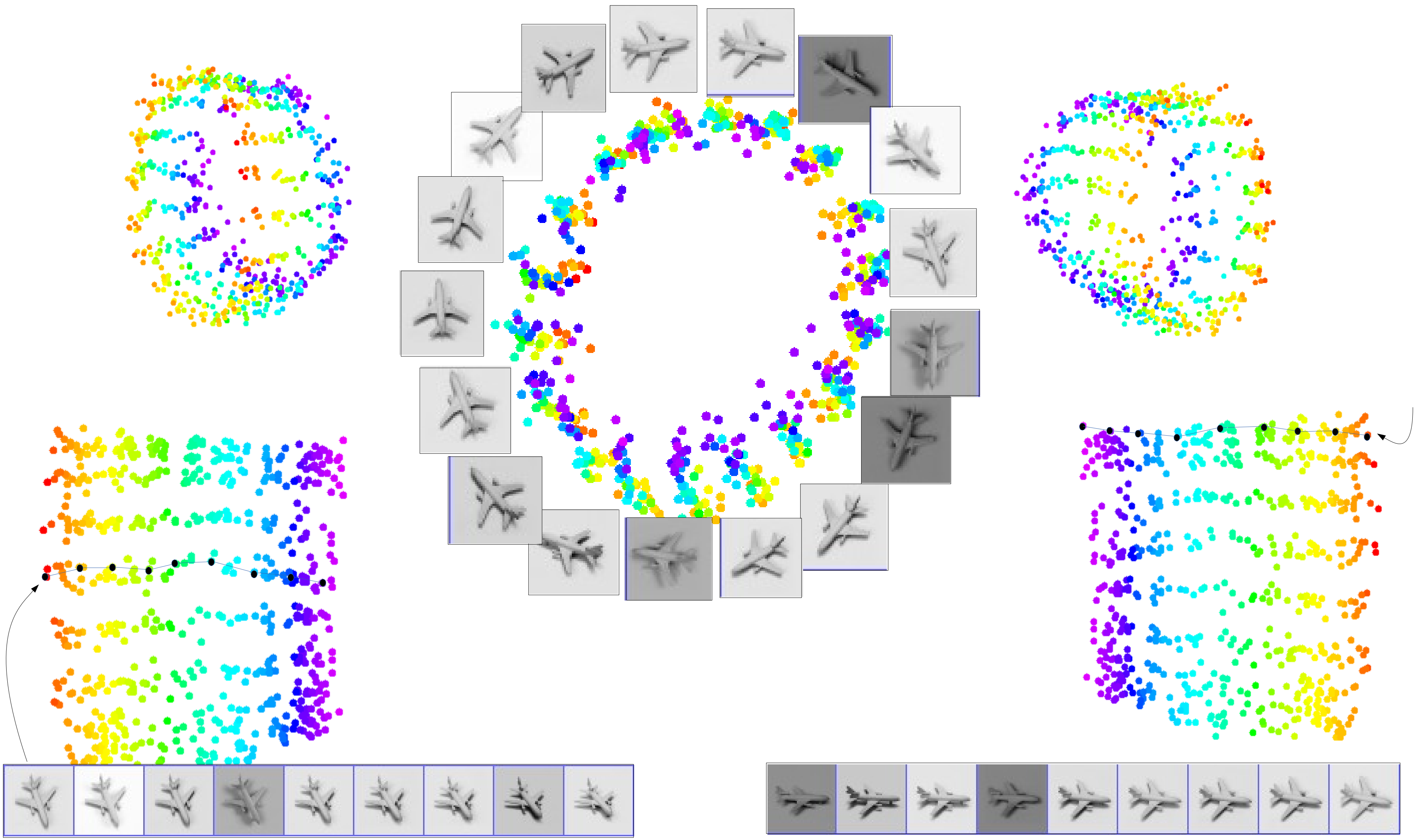
$$\begin{bmatrix} 1.2 \\ -3 \\ 0.2 \\ -2\ldots \end{bmatrix}$$

Face/not face
Pose
Lighting
Expression

Data Manifold & Invariance: Some variations must be eliminated

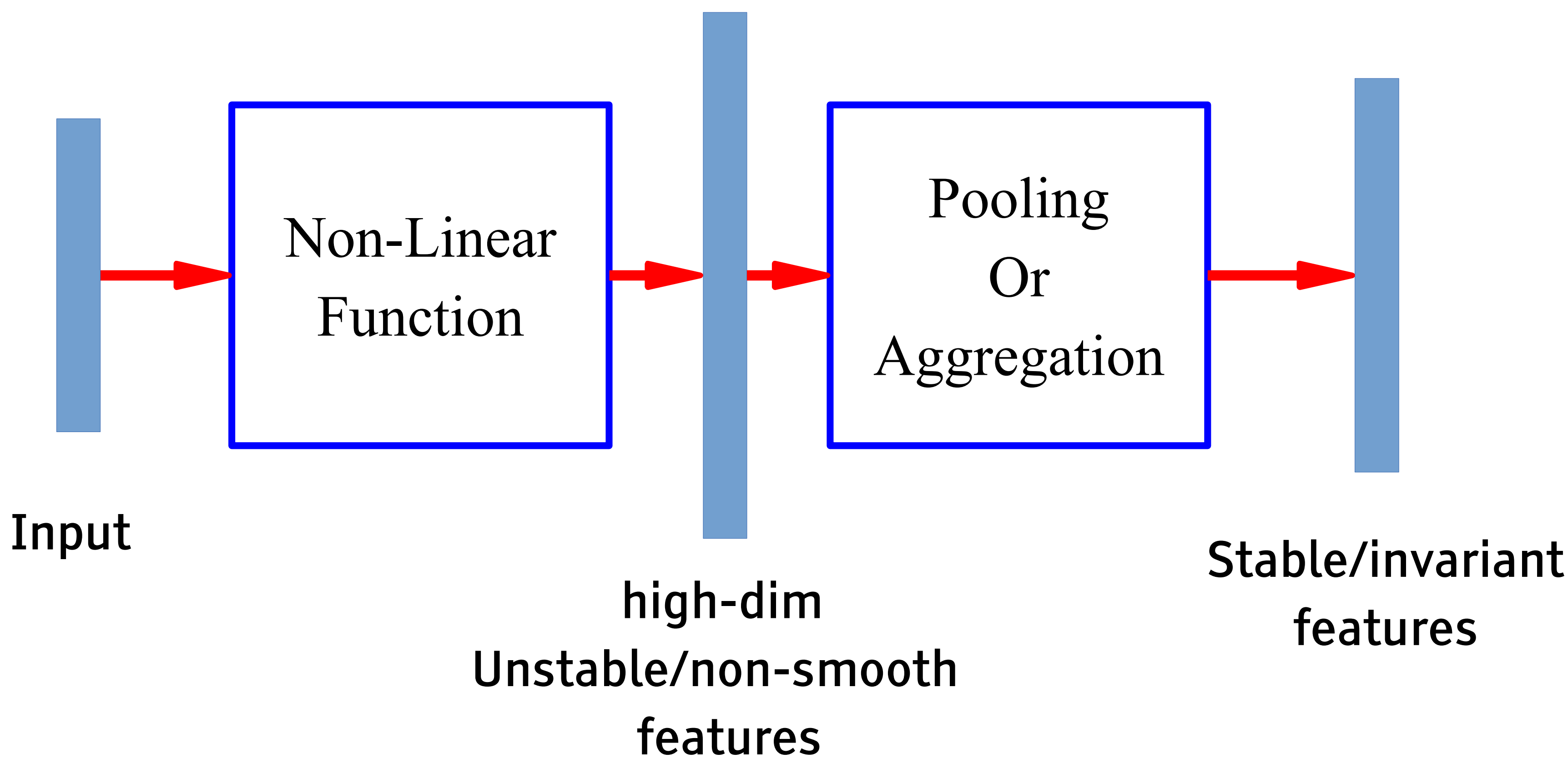
- Azimuth-Elevation manifold. Ignores lighting.

[Hadsell et al. CVPR 2006]



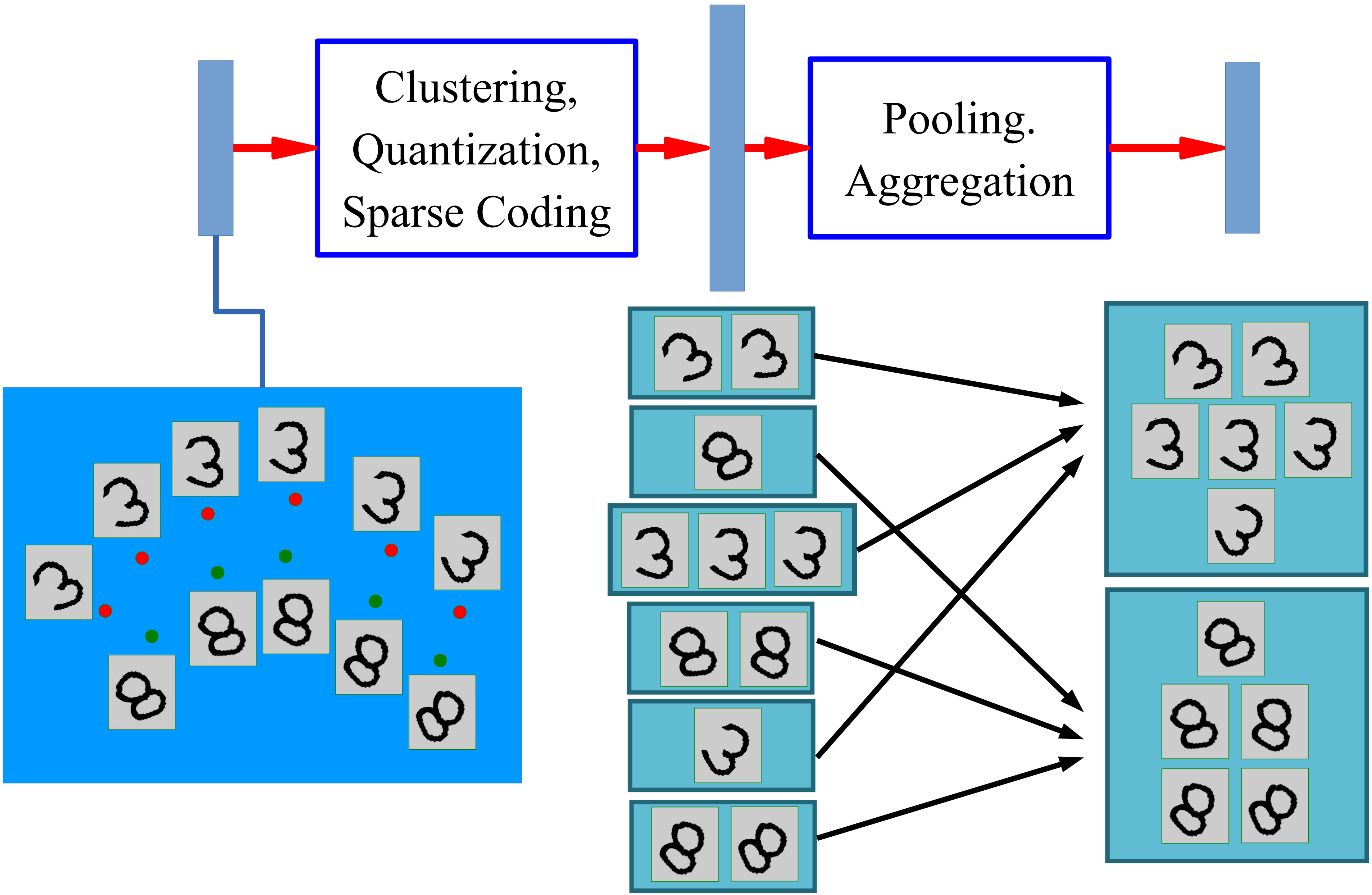
Basic Idea for Invariant Feature Learning

- Embed the input non-linearly into a high(er) dimensional space
  - In the new space, things that were non separable may become separable
- Pool regions of the new space together
  - Bringing together things that are semantically similar. Like pooling.

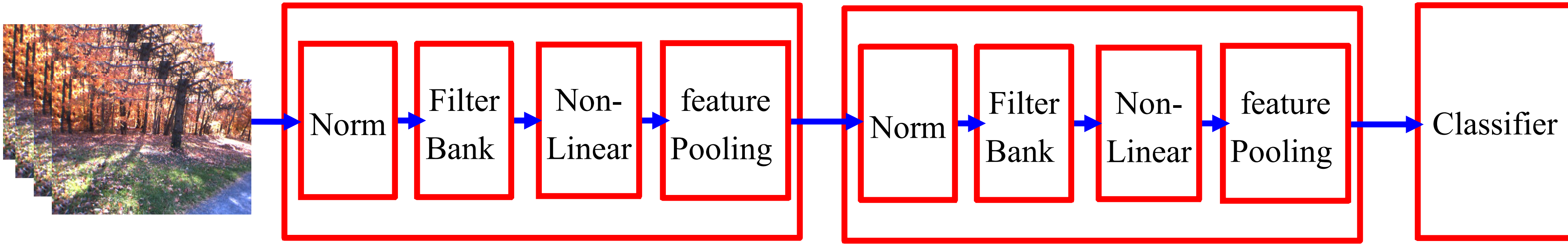


Sparse Non-Linear Expansion → Pooling

- Use clustering to break things apart, pool together similar things



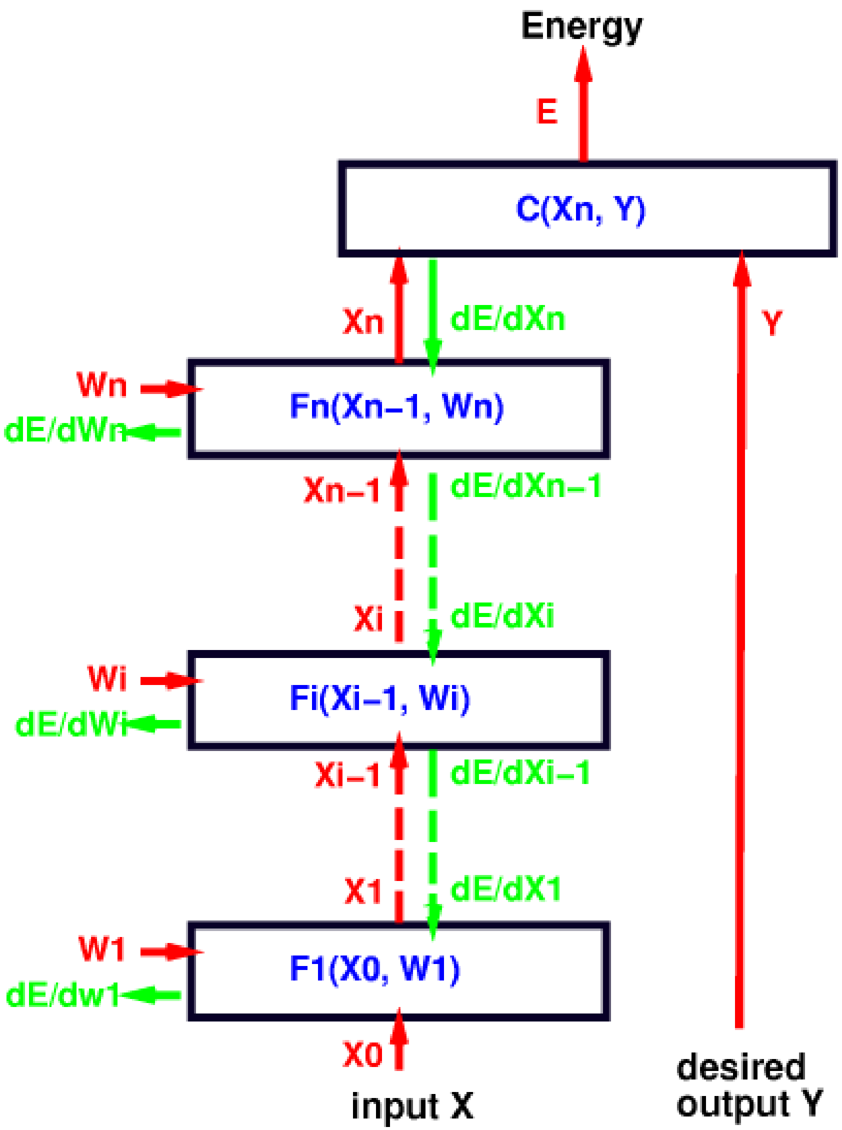
Overall Architecture:
Normalization → Filter Bank → Non-Linearity → Pooling



- Stacking multiple stages of
  - [Normalization → Filter Bank → Non-Linearity → Pooling].
- Normalization: variations on whitening
  - Subtractive: average removal, high pass filtering
  - Divisive: local contrast normalization, variance normalization
- Filter Bank: dimension expansion, projection on overcomplete basis
- Non-Linearity: sparsification, saturation, lateral inhibition....
  - Rectification (ReLU), Component-wise shrinkage, tanh, winner-takes-all
- Pooling: aggregation over space or feature type
  - $$X_i; \quad L_p : \sqrt[p]{X_i^p}; \quad PROB : \frac{1}{b}\log\left(\sum_i e^{bX_i}\right)$$

Deep Supervised Learning
(modular approach)

Multimodule Systems: Cascade



- Complex learning machines can be built by assembling modules into networks
- Simple example: sequential/layered feed-forward architecture (cascade)
- Forward Propagation:

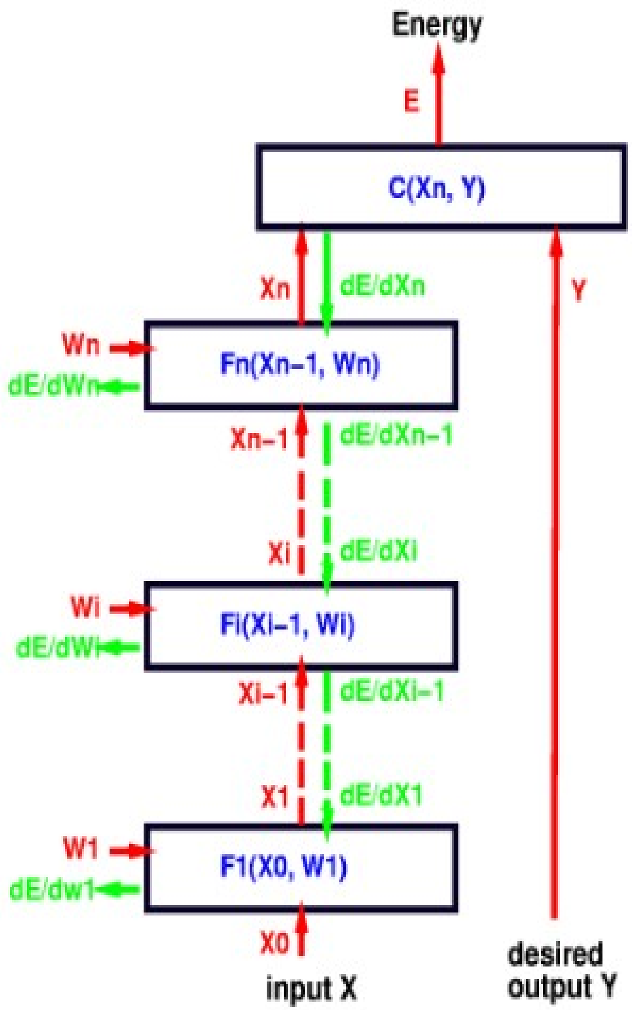
$$\text{let } X = X_0,$$

$$X_i = F_i(X_{i-1}, W_i) \quad \forall i \in [1, n]$$

$$E(Y, X, W) = C(X_n, Y)$$

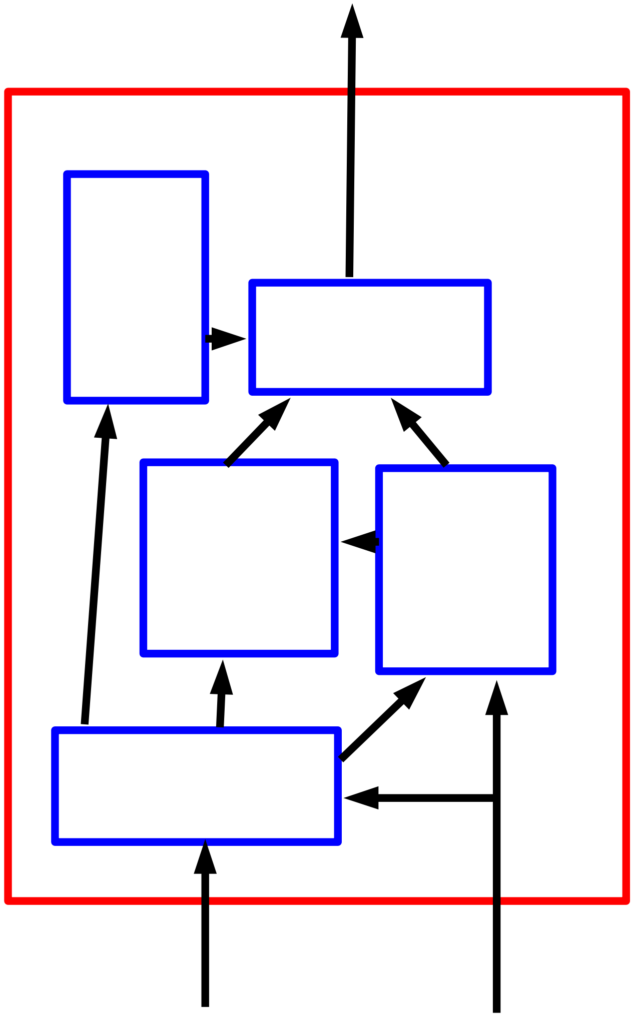
Back Propagation

To compute all the derivatives, we use a backward sweep called the **back-propagation algorithm** that uses the recurrence equation for $\frac{\partial E}{\partial X_i}$



- $\frac{\partial E}{\partial X_n} = \frac{\partial C(X_n, Y)}{\partial X_n}$
- $\frac{\partial E}{\partial X_{n-1}} = \frac{\partial E}{\partial X_n} \frac{\partial F_n(X_{n-1}, W_n)}{\partial X_{n-1}}$
- $\frac{\partial E}{\partial W_n} = \frac{\partial E}{\partial X_n} \frac{\partial F_n(X_{n-1}, W_n)}{\partial W_n}$
- $\frac{\partial E}{\partial X_{n-2}} = \frac{\partial E}{\partial X_{n-1}} \frac{\partial F_{n-1}(X_{n-2}, W_{n-1})}{\partial X_{n-2}}$
- $\frac{\partial E}{\partial W_{n-1}} = \frac{\partial E}{\partial X_{n-1}} \frac{\partial F_{n-1}(X_{n-2}, W_{n-1})}{\partial W_{n-1}}$
- ....etc, until we reach the first module.
- we now have all the $\frac{\partial E}{\partial W_i}$ for $i \in [1, n]$.

Any Architecture works

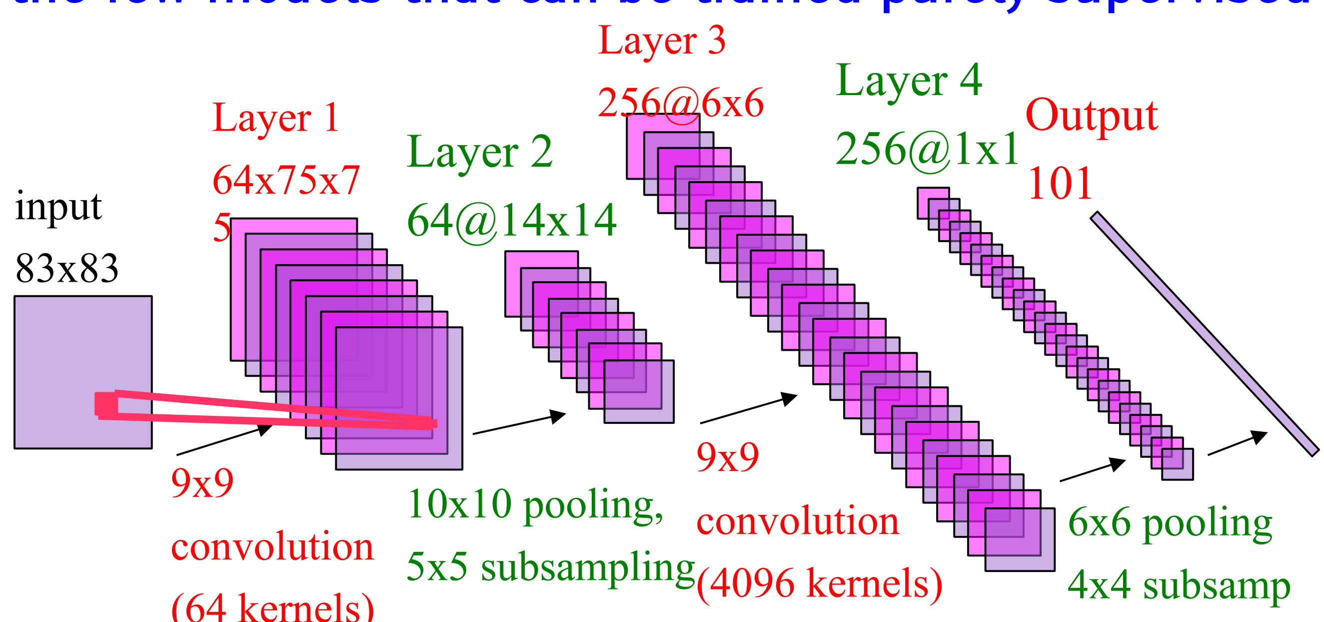


- **Any connection is permissible**
  - Networks with loops must be "unfolded in time".
- **Any module is permissible**
  - As long as it is continuous and differentiable almost everywhere with respect to the parameters, and with respect to non-terminal inputs.

Convolutional Networks

Convolutional Nets

- Are deployed in many practical applications
  - Image reco, speech reco, Google's and Baidu's photo taggers
- Have won several competitions
  - ImageNet, Kaggle Facial Expression, Kaggle Multimodal Learning, German Traffic Signs, Connectomics, Handwriting....
- Are applicable to array data where nearby values are correlated
  - Images, sound, time-frequency representations, video, volumetric images, RGB-Depth images,.....
- One of the few models that can be trained purely supervised



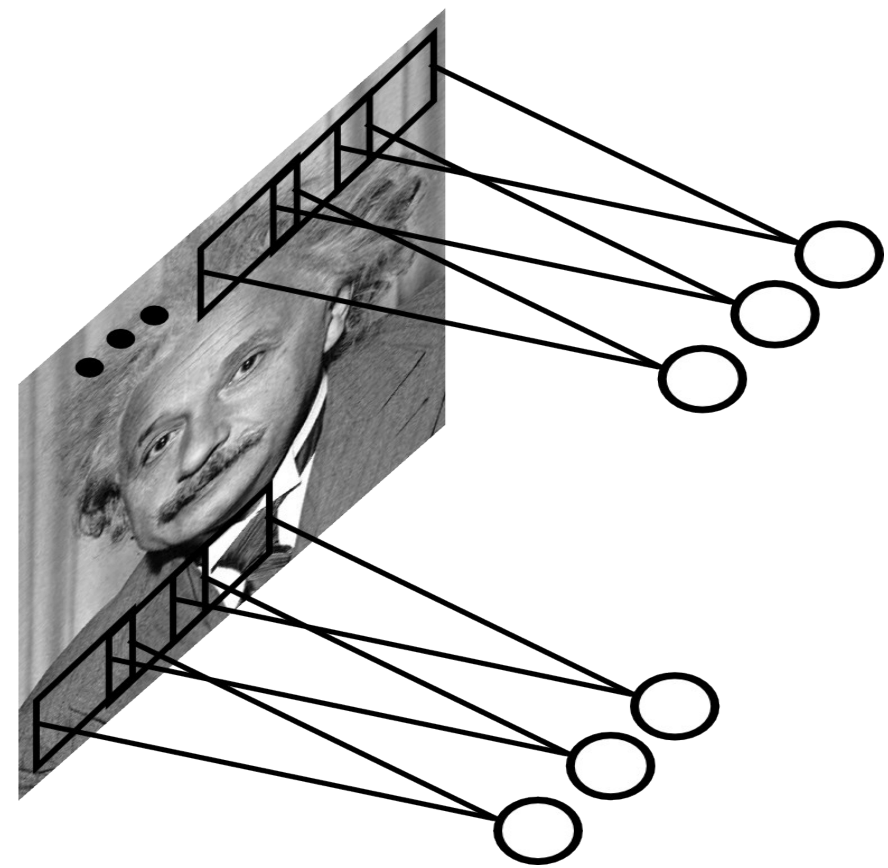
Shared Weights & Convolutions: Exploiting Stationarity and Local Correlations

- Features that are useful on one part of the image and probably useful elsewhere.
- All units share the same set of weights
- Shift equivariant processing:
  - When the input shifts, the output also shifts but stays otherwise unchanged.
- Convolution
  - with a learned kernel (or filter)
  - Non-linearity: ReLU (rectified linear)
- The filtered "image" Z is called a feature map

$$A_{ij} = \sum_{kl} W_{kl} X_{i+j.\, k+l}$$

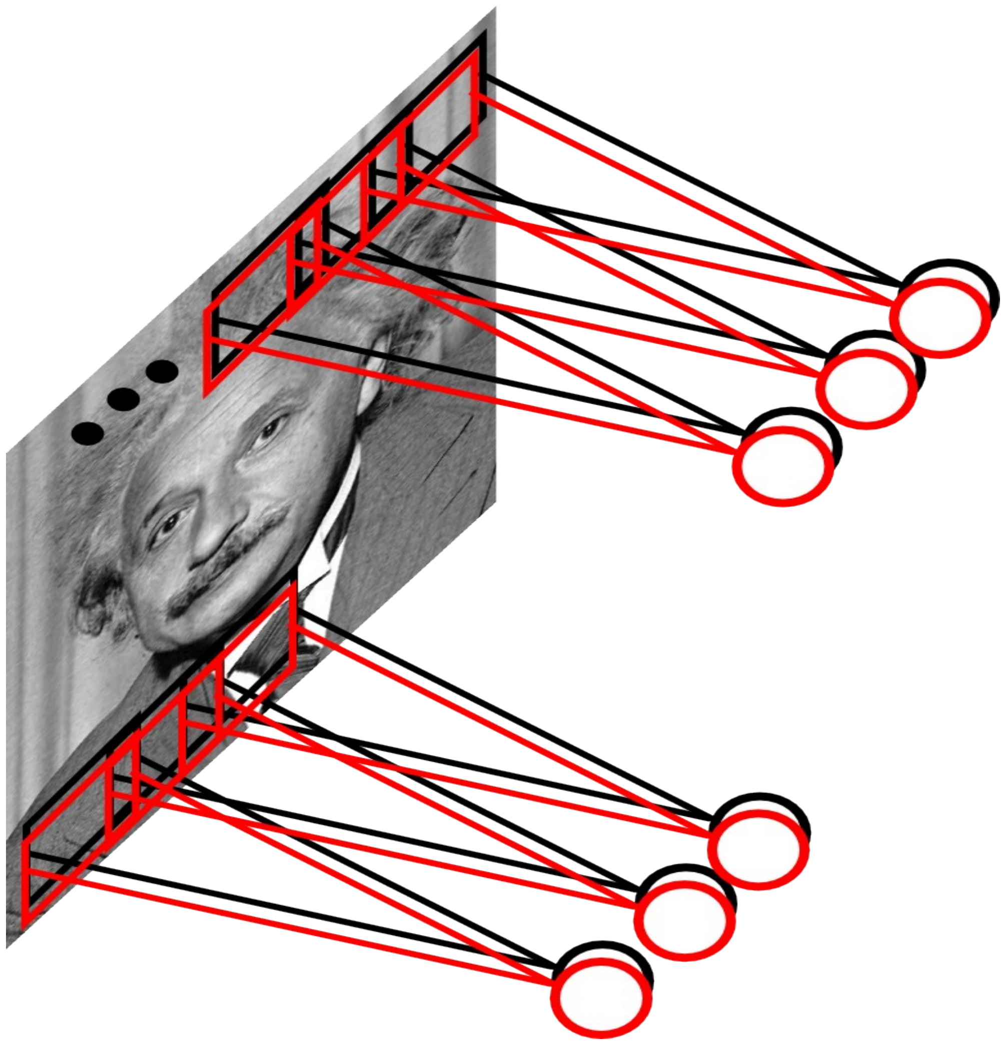
$$Z_{ij} = max(0, A_{ij})$$

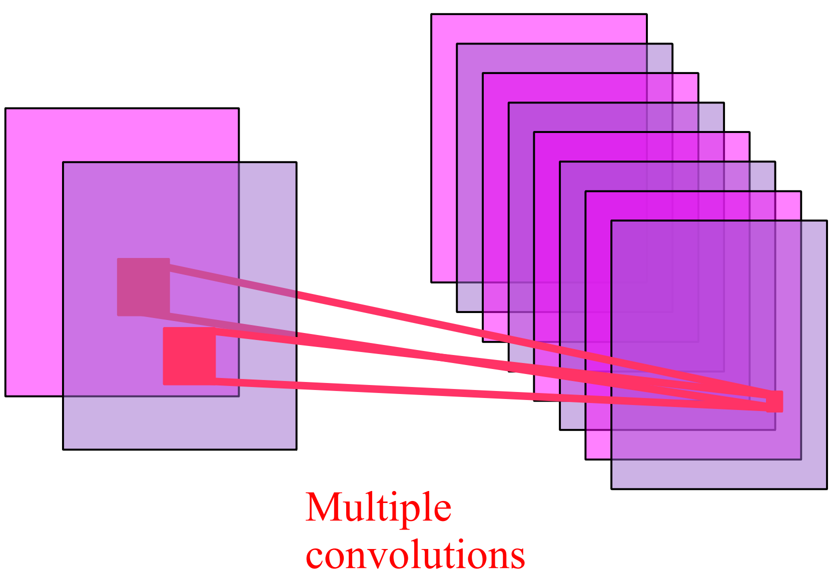
- Example: 200x200 image
  - 400,000 hidden units with 10x10 fields = 1000 params
  - 10 feature maps of size 200x200, 10 filters of size 10x10



Multiple Convolutions with Different Kernels

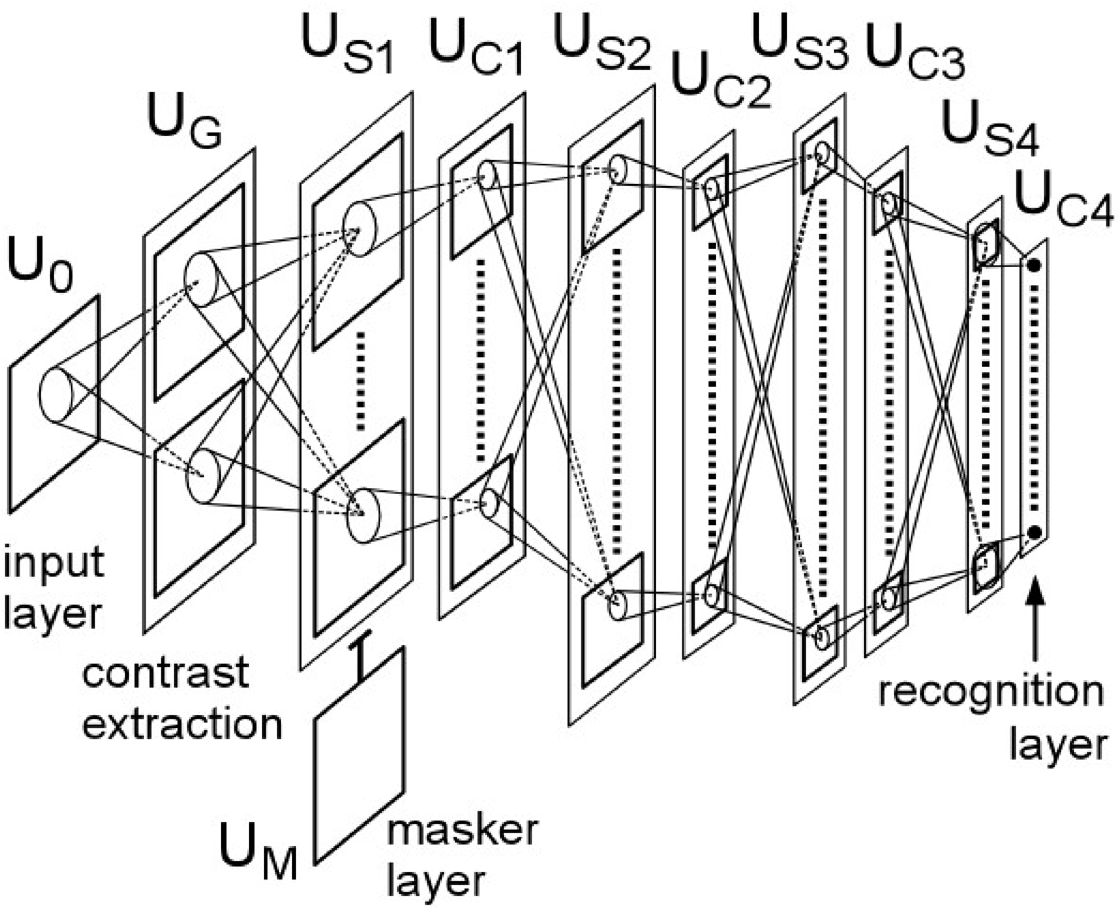
- Detects multiple motifs at each location
- The collection of units looking at the same patch is akin to a feature vector for that patch.
- The result is a 3D array, where each slice is a feature map.

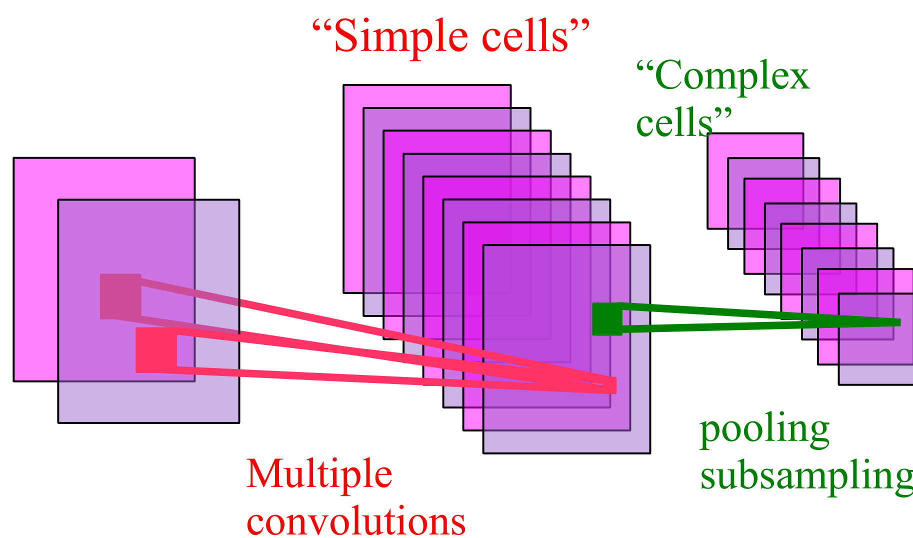




Early Hierarchical Feature Models for Vision

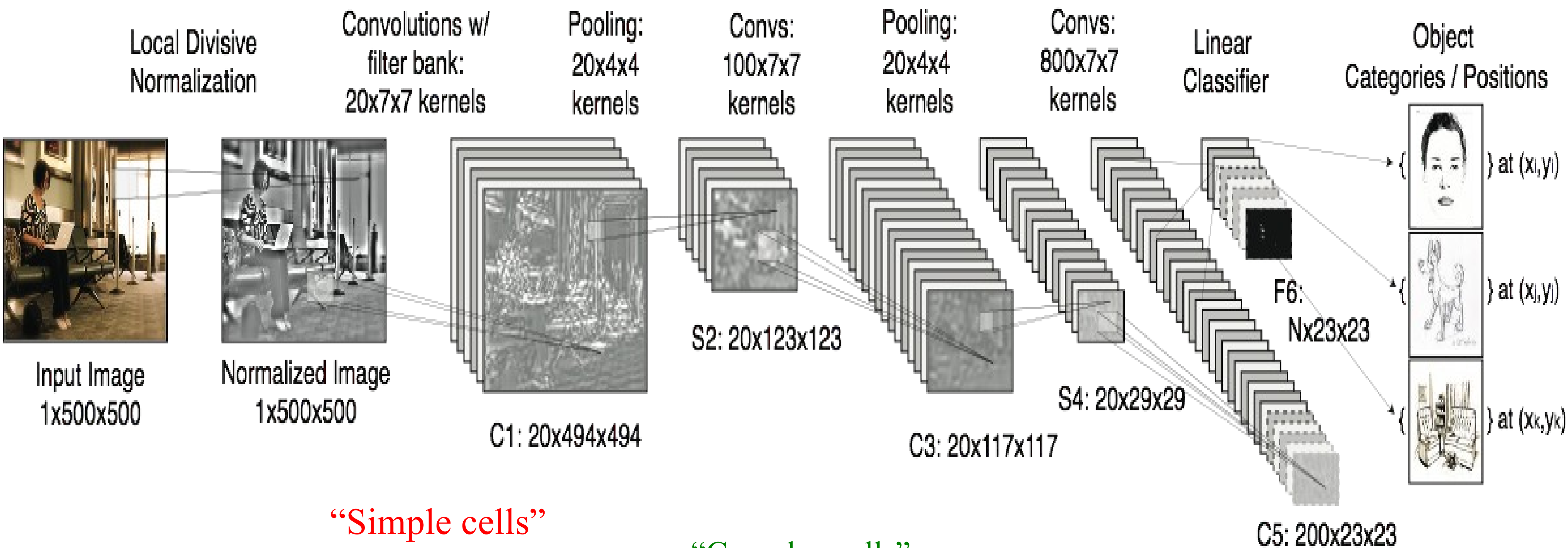
- [Hubel & Wiesel 1962]:
  - simple cells detect local features
  - complex cells "pool" the outputs of simple cells within a retinotopic neighborhood.

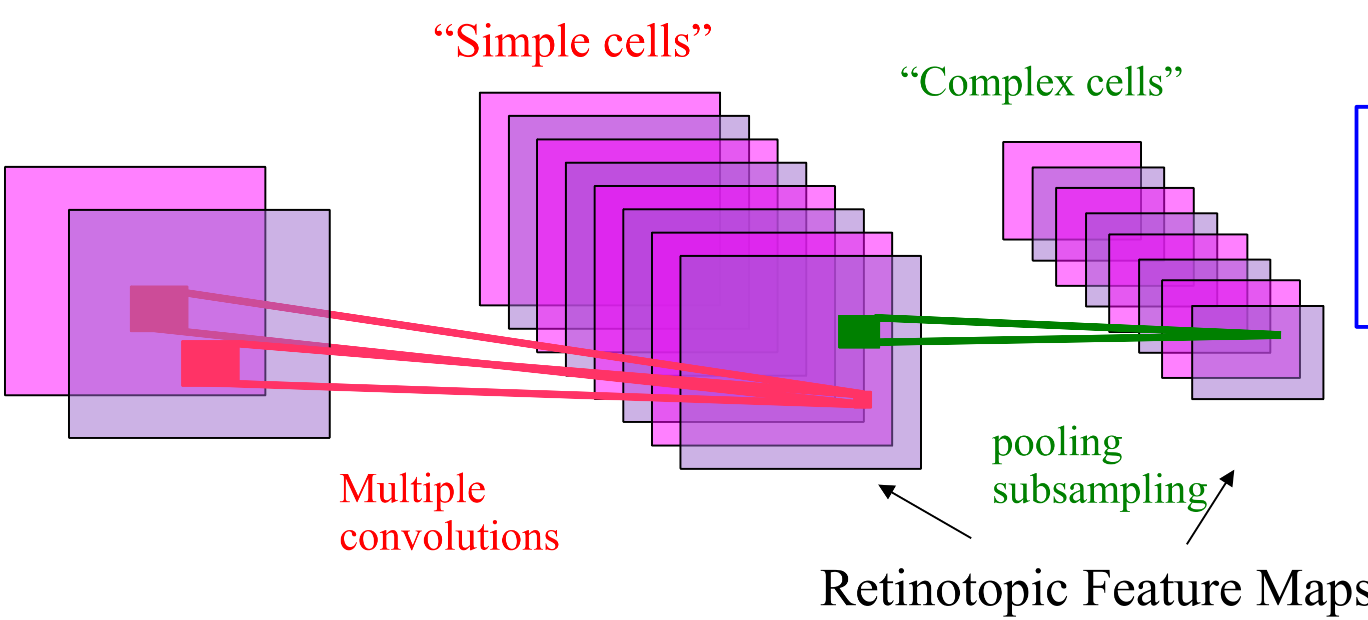




Cognitron & Neocognitron [Fukushima 1974-1982]

The Convolutional Net Model
(Multistage Hubel-Wiesel system)



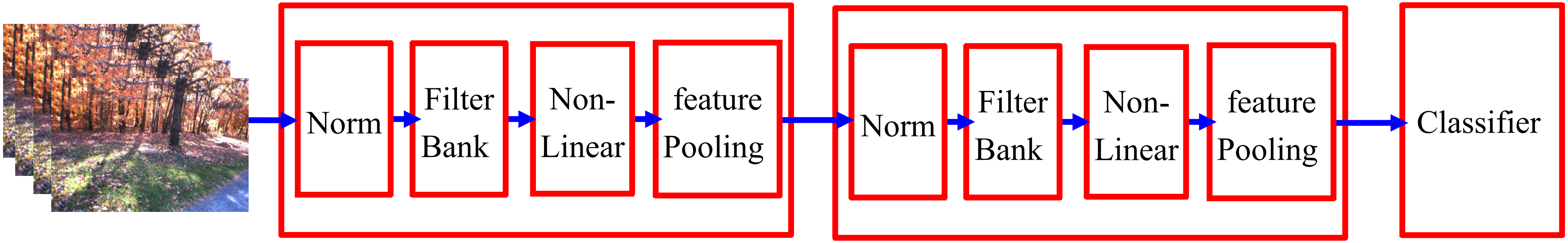


- Training is supervised
- With stochastic gradient descent

[LeCun et al. 89]

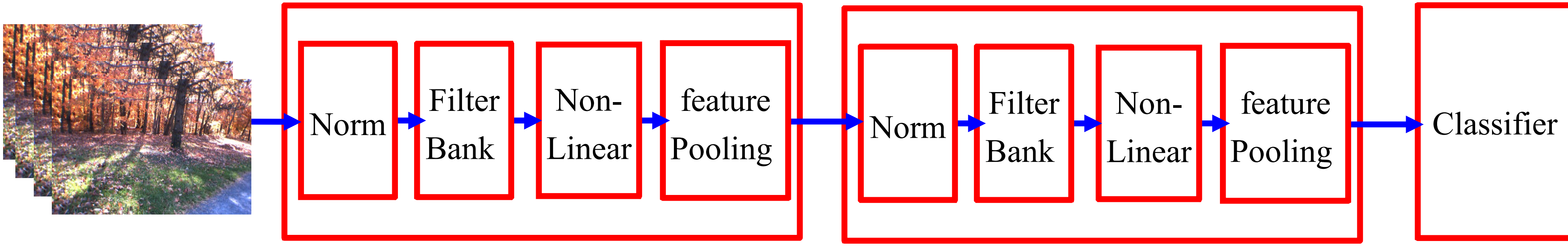
[LeCun et al. 98]

Feature Transform:
Normalization → Filter Bank → Non-Linearity → Pooling



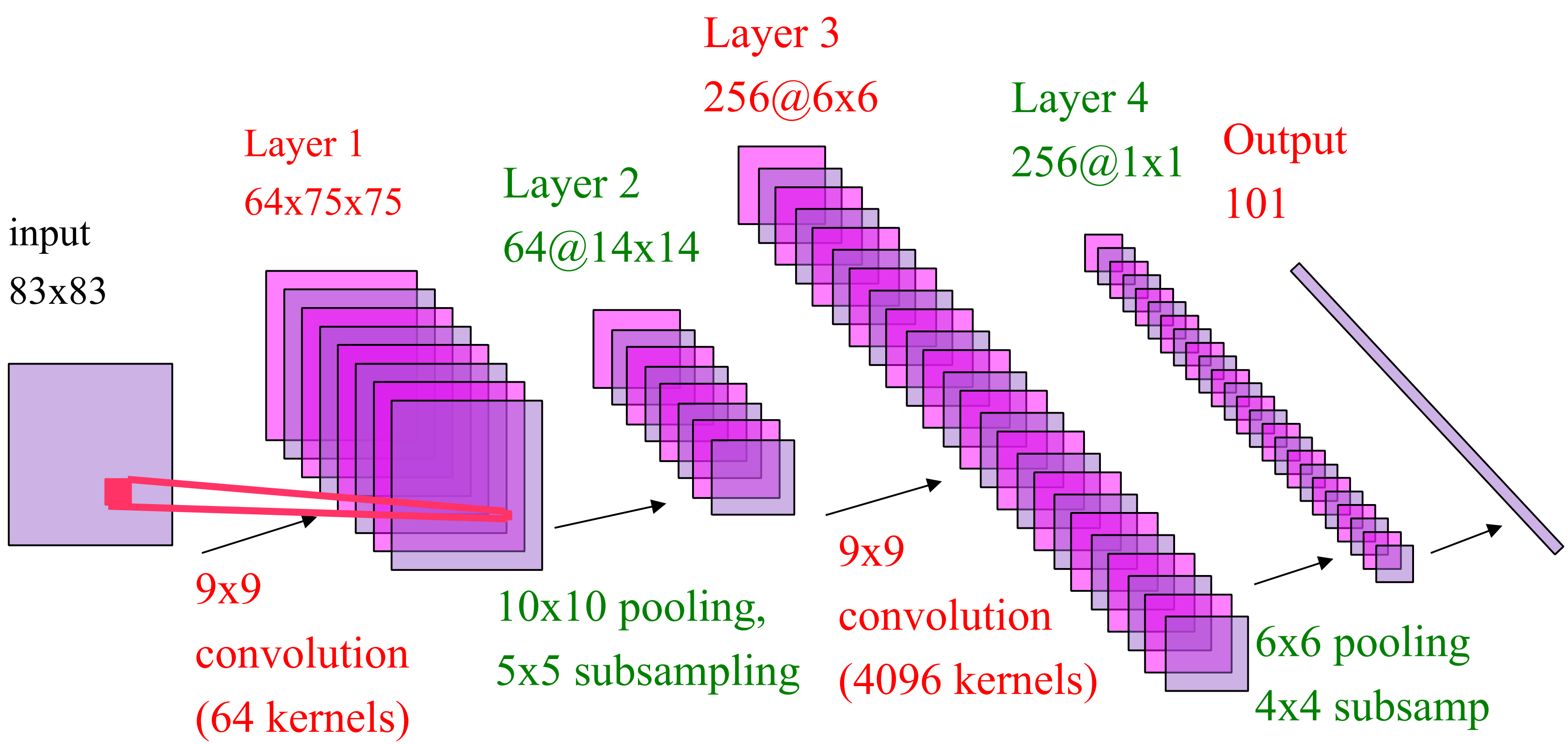
- Stacking multiple stages of
  - [Normalization → Filter Bank → Non-Linearity → Pooling].
- Normalization: variations on whitening
  - Subtractive: average removal, high pass filtering
  - Divisive: local contrast normalization, variance normalization
- Filter Bank: dimension expansion, projection on overcomplete basis
- Non-Linearity: sparsification, saturation, lateral inhibition....
  - Rectification, Component-wise shrinkage, tanh, winner-takes-all
- Pooling: aggregation over space or feature type, subsampling
  - $X_i; \quad L_p: \sqrt[p]{X_i^p}; \quad PROB: \frac{1}{b}\log\left(\sum_i e^{bX_i}\right)$

Feature Transform:
Normalization → Filter Bank → Non-Linearity → Pooling



- Filter Bank → Non-Linearity = Non-linear embedding in high dimension
- Feature Pooling = contraction, dimensionality reduction, smoothing
- Learning the filter banks at every stage
- Creating a hierarchy of features
- Basic elements are inspired by models of the visual (and auditory) cortex
  - Simple Cell + Complex Cell model of [Hubel and Wiesel 1962]
  - Many "traditional" feature extraction methods are based on this
  - SIFT, GIST, HoG, SURF...
- [Fukushima 1974-1982], [LeCun 1988-now],
  - since the mid 2000: Hinton, Seung, Poggio, Ng,....

Convolutional Network (ConvNet)

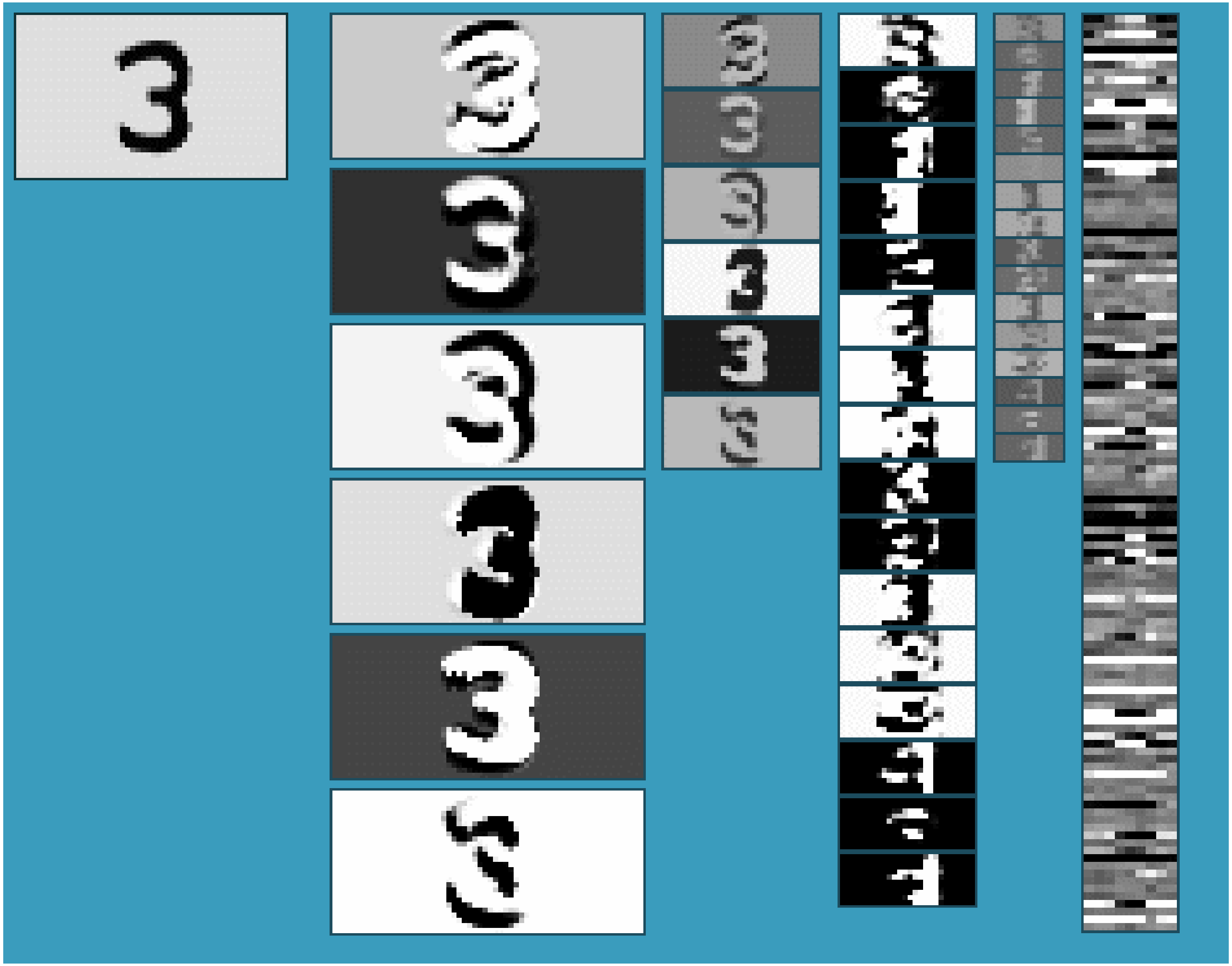


- Non-Linearity: half-wave rectification, shrinkage function, sigmoid
- Pooling: average, L1, L2, max
- Training: Supervised (1988-2006), Unsupervised+Supervised (2006-now)

Convolutional Network (vintage 1990)

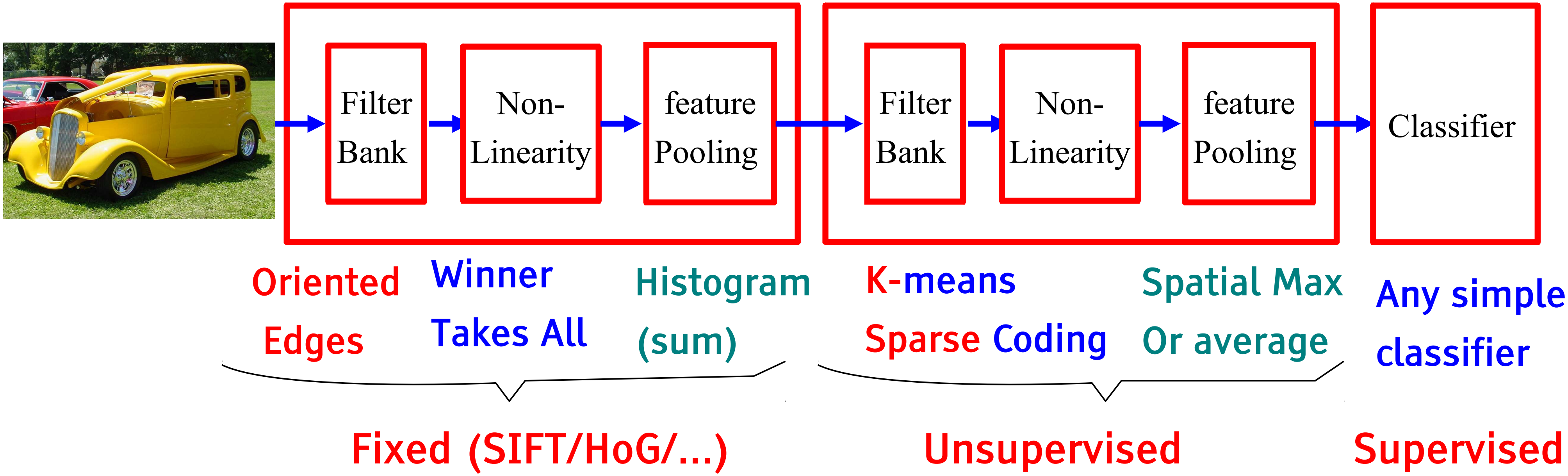
- filters → tanh → average-tanh → filters → tanh → average-tanh → filters → tanh

Curved manifold



Flatter manifold

"Mainstream" object recognition pipeline 2006-2012: somewhat similar to ConvNets



- Fixed Features + unsupervised mid-level features + simple classifier
  - SIFT + Vector Quantization + Pyramid pooling + SVM
    - [Lazebnik et al. CVPR 2006]
  - SIFT + Local Sparse Coding Macrofeatures + Pyramid pooling + SVM
    - [Boureau et al. ICCV 2011]
  - SIFT + Fisher Vectors + Deformable Parts Pooling + SVM
    - [Perronin et al. 2012]

Tasks for Which Deep Convolutional Nets are the Best

- Handwriting recognition MNIST (many), Arabic HWX (IDSIA)
- OCR in the Wild [2011]: StreetView House Numbers (NYU and others)
- Traffic sign recognition [2011] GTSRB competition (IDSIA, NYU)
- Asian handwriting recognition [2013] ICDAR competition (IDSIA)
- Pedestrian Detection [2013]: INRIA datasets and others (NYU)
- Volumetric brain image segmentation [2009] connectomics (IDSIA, MIT)
- Human Action Recognition [2011] Hollywood II dataset (Stanford)
- Object Recognition [2012] ImageNet competition (Toronto)
- Scene Parsing [2012] Stanford bgd, SiftFlow, Barcelona datasets (NYU)
- Scene parsing from depth images [2013] NYU RGB-D dataset (NYU)
- Speech Recognition [2012] Acoustic modeling (IBM and Google)
- Breast cancer cell mitosis detection [2011] MITOS (IDSIA)

- The list of perceptual tasks for which ConvNets hold the record is growing.
- Most of these tasks (but not all) use purely supervised convnets.

Commercial Applications of Convolutional Nets

- Form Reading: AT&T 1994
- Check reading: AT&T 1996 (read 10-20% of all US checks in 2000)
- Handwriting recognition: Microsoft early 2000
- Face and person detection: NEC 2005
- Face and License Plate Detection: Google/StreetView 2009
- Gender and age recognition: NEC 2010 (vending machines)
- OCR in natural images: Google 2013 (StreetView house numbers)
- Photo tagging: Google 2013
- Image Search by Similarity: Baidu 2013

- Suspected applications from Google, Baidu, Microsoft, IBM.....
  - Speech recognition, porn filtering,....

Ideas from Neuroscience and Psychophysics

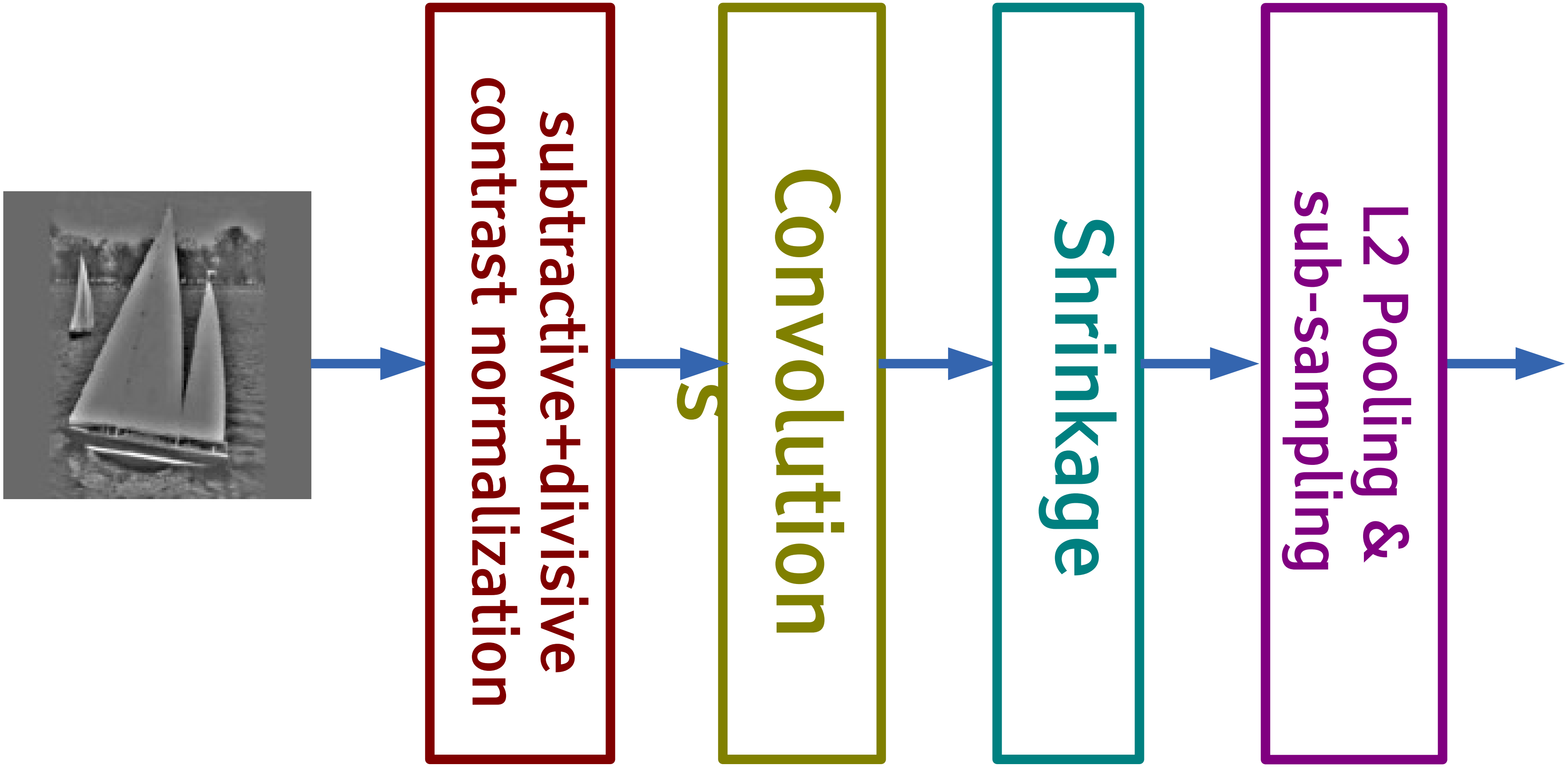
- The whole architecture: simple cells and complex cells
- Local receptive fields
- Self-similar receptive fields over the visual field (convolutions)
- Pooling (complex cells)
- Non-Linearity: Rectified Linear Units (ReLU)
- LGN-like band-pass filtering and contrast normalization in the input
- Divisive contrast normalization (from Heeger, Simoncelli....)
  - Lateral inhibition
- Sparse/Overcomplete representations (Olshausen-Field....)
- Inference of sparse representations with lateral inhibition
- Sub-sampling ratios in the visual cortex
  - between 2 and 3 between V1-V2-V4
- Crowding and visual metamers give cues on the size of the pooling areas

Simple ConvNet Applications with State-of-the-Art Performance

- **Traffic Sign Recognition (GTSRB)**
  - German Traffic Sign Reco Bench
  - 99.2% accuracy
  - #1: IDSIA; #2 NYU

- **House Number Recognition (Google)**
  - Street View House Numbers
  - 94.3 % accuracy

One Stage: Contrast Norm → Filter Bank → Shrinkage → L2 Pooling



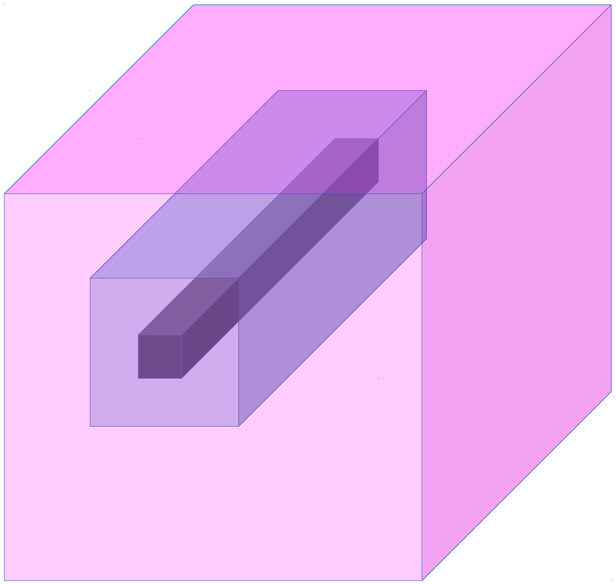
THIS IS **ONE STAGE** OF THE CONVNET

Results on Caltech101 with sigmoid non-linearity

Single Stage System: $[64.F^{9\times9}_{CSG} - R/N/P^{5\times5}]$ **- log_reg**					
R/N/P	$R_{abs} - N - P_A$	$R_{abs} - P_A$	$N - P_M$	$N - P_A$	P_A
U^+	54.2%	50.0%	44.3%	18.5%	14.5%
R^+	54.8%	47.0%	38.0%	16.3%	14.3%
U	52.2%	43.3%(±1.6)	44.0%	17.2%	13.4%
R	53.3%	31.7%	32.1%	15.3%	12.1%(±2.2)
G	52.3%				
Two Stage System: $[64.F^{9\times9}_{CSG} - R/N/P^{5\times5}] - [256.F^{9\times9}_{CSG} - R/N/P^{4\times4}]$ **- log_reg**					
R/N/P	$R_{abs} - N - P_A$	$R_{abs} - P_A$	$N - P_M$	$N - P_A$	P_A
U^+U^+	65.5%	60.5%	61.0%	34.0%	32.0%
R^+R^+	64.7%	59.5%	60.0%	31.0%	29.7%
UU	63.7%	46.7%	56.0%	23.1%	9.1%
RR	62.9%	33.7%(±1.5)	37.6%(±1.9)	19.6%	8.8%
GT	55.8%	← like HMAX model			
Single Stage: $[64.F^{9\times9}_{CSG} - R/N/P^{5\times5}]$ **- PMK-SVM**					
U	64.0%				
Two Stages: $[64.F^{9\times9}_{CSG} - R/N/P^{5\times5}] - [256.F^{9\times9}_{CSG} - R/N]$ **- PMK-SVM**					
UU	52.8%				

Local Contrast Normalization

- Performed on the state of every layer, including the input
- Subtractive Local Contrast Normalization
  - Subtracts from every value in a feature a Gaussian-weighted average of its neighbors (high-pass filter)
- Divisive Local Contrast Normalization
  - Divides every value in a layer by the standard deviation of its neighbors over space and over all feature maps
- Subtractive + Divisive LCN performs a kind of approximate whitening.



The Effect of Architectural Elements

- Pyramid pooling on last layer: 1% improvement over regular pooling
- Shrinkage non-linearity + lateral inhibition: 1.6% improvement over tanh
- Discriminative term in sparse coding: 2.8% improvement

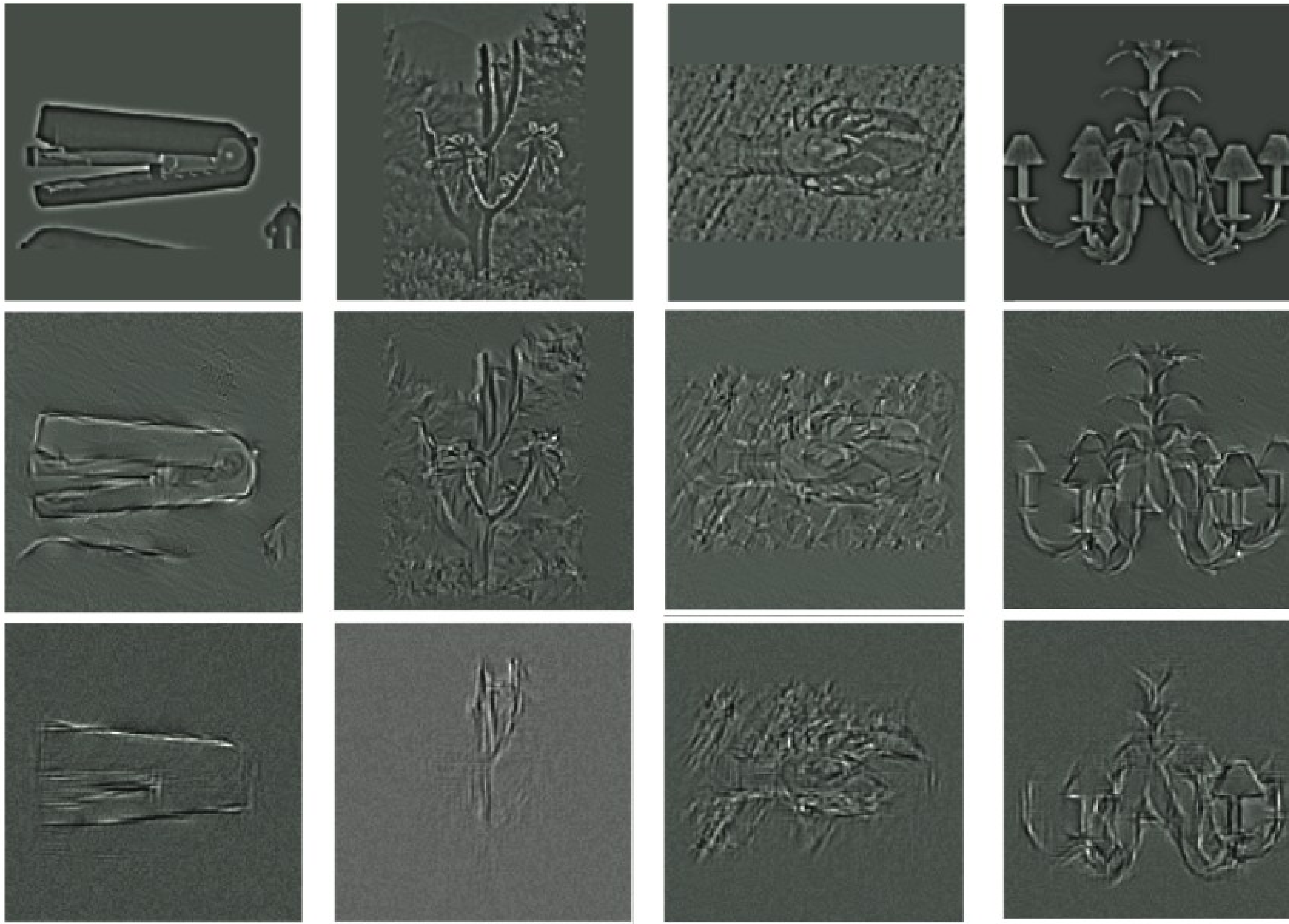
Architecture	**Protocol**	%
(1) $F_{\tanh} - R_{abs} - N - P_A^{pyr}$	$\mathbf{R^+R^+}$	65.4 ± 1.0
(2) $F_{\tanh} - R_{abs} - N - P_A^{pyr}$	$\mathbf{U^+U^+}$	66.2 ± 1.0
(3) $F_{si} - R_{abs} - N - P_A$	$\mathbf{R^+R^+}$	63.3 ± 1.0
(4) $F_{si} - R_{abs} - N - P_A$	$\mathbf{UU}$	60.4 ± 0.6
(5) $F_{si} - R_{abs} - N - P_A$	$\mathbf{U^+U^+}$	66.4 ± 0.5
(6) $F_{si} - R_{abs} - N - P_A^{pyr}$	$\mathbf{U^+U^+}$	67.8 ± 0.4
(7) $F_{si} - R_{abs} - N - P_A$	$\mathbf{DD}$	66.0 ± 0.3
(8) $F_{si} - R_{abs} - N - P_A$	$\mathbf{D^+D^+}$	68.7 ± 0.2
(9) $F_{si} - R_{abs} - N - P_A^{pyr}$	$\mathbf{D^+D^+}$	$\mathbf{70.6 \pm 0.3}$

What does Local Contrast Normalization Do?

Original

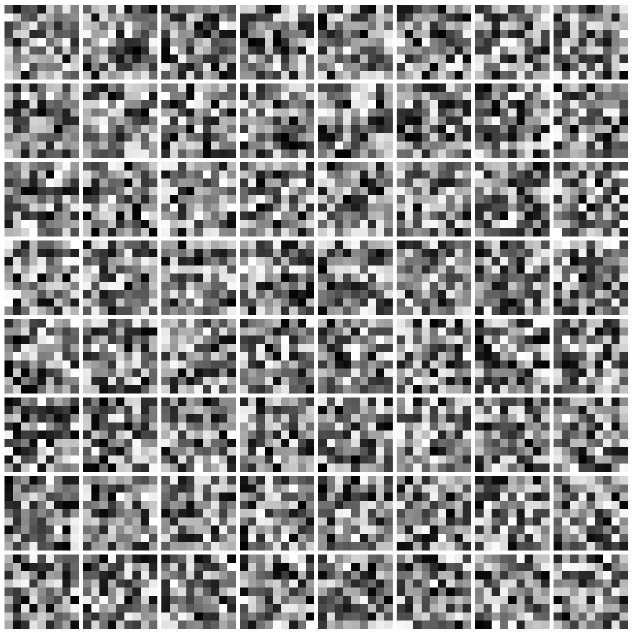
Reconstuction
With LCN

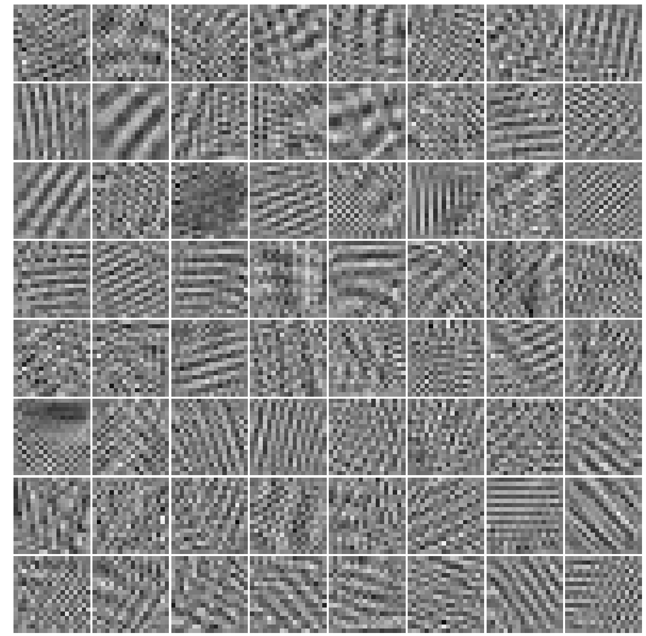
Reconstruction
Without LCN



Why Do Random Filters Work?

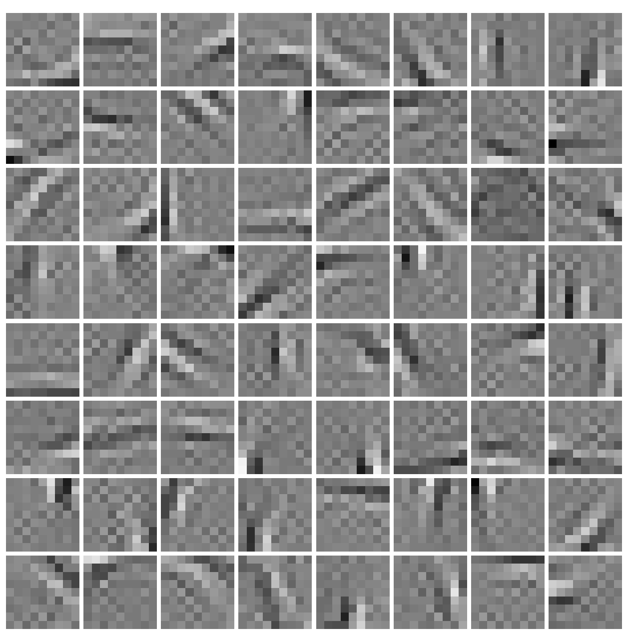
Random Filters For Simple Cells

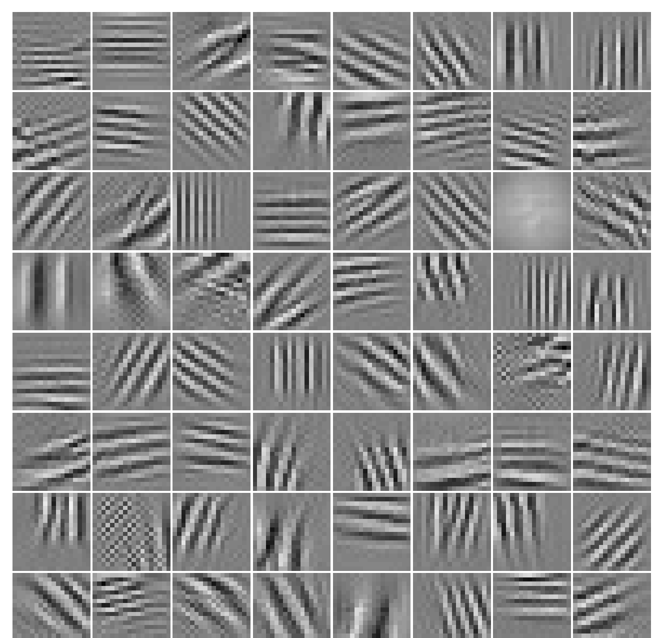




Optimal Stimuli for each Complex Cell

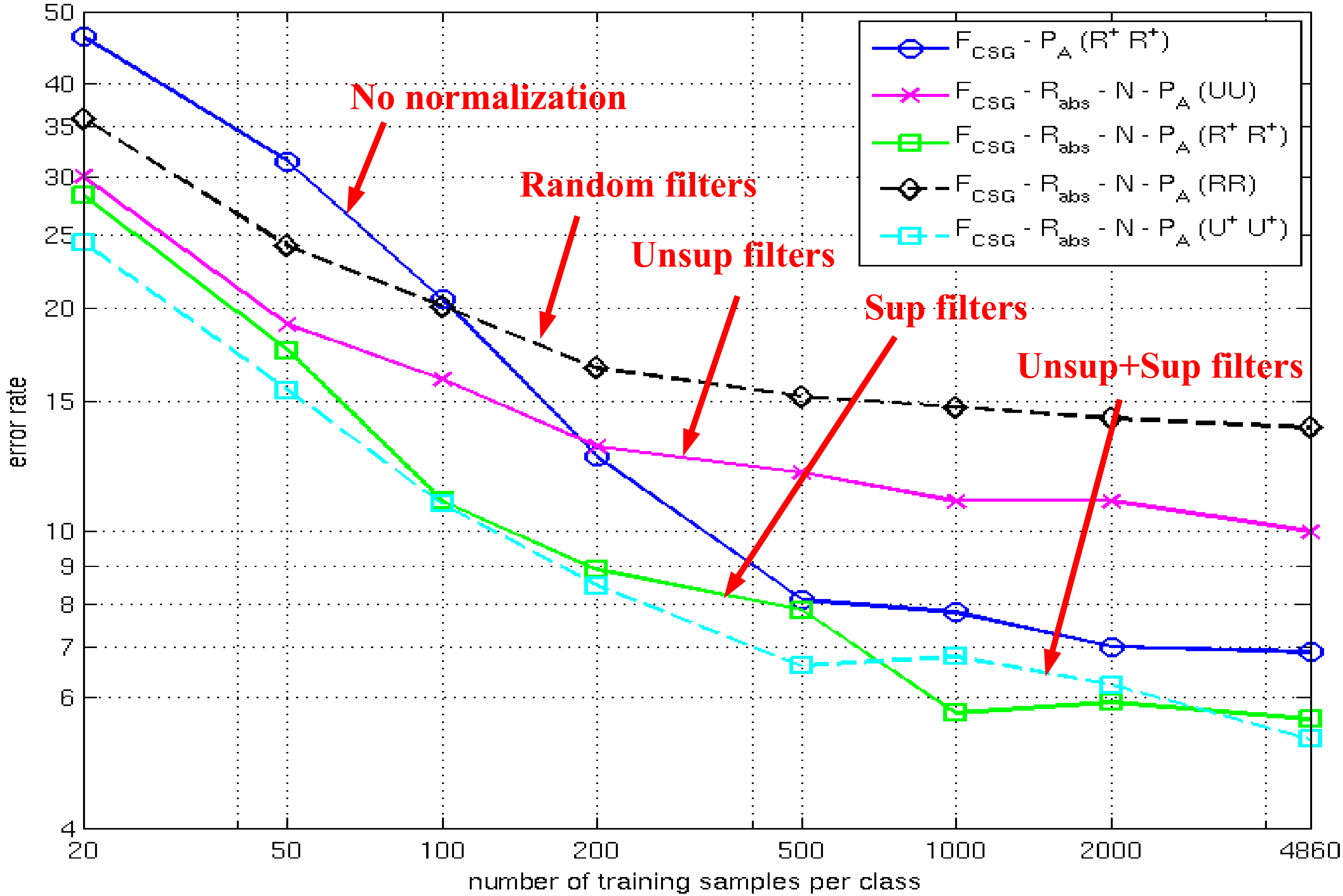
Trained Filters For Simple Cells





Small NORB dataset

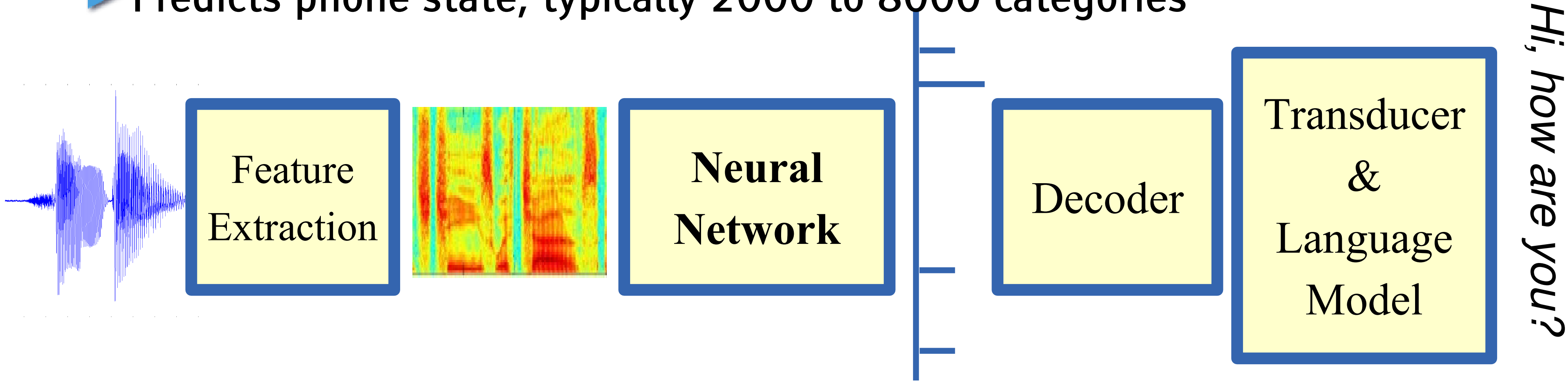
- Two-stage system: error rate versus number of labeled training samples



Deep Learning and Convolutional Networks in Speech, Audio, and Signals

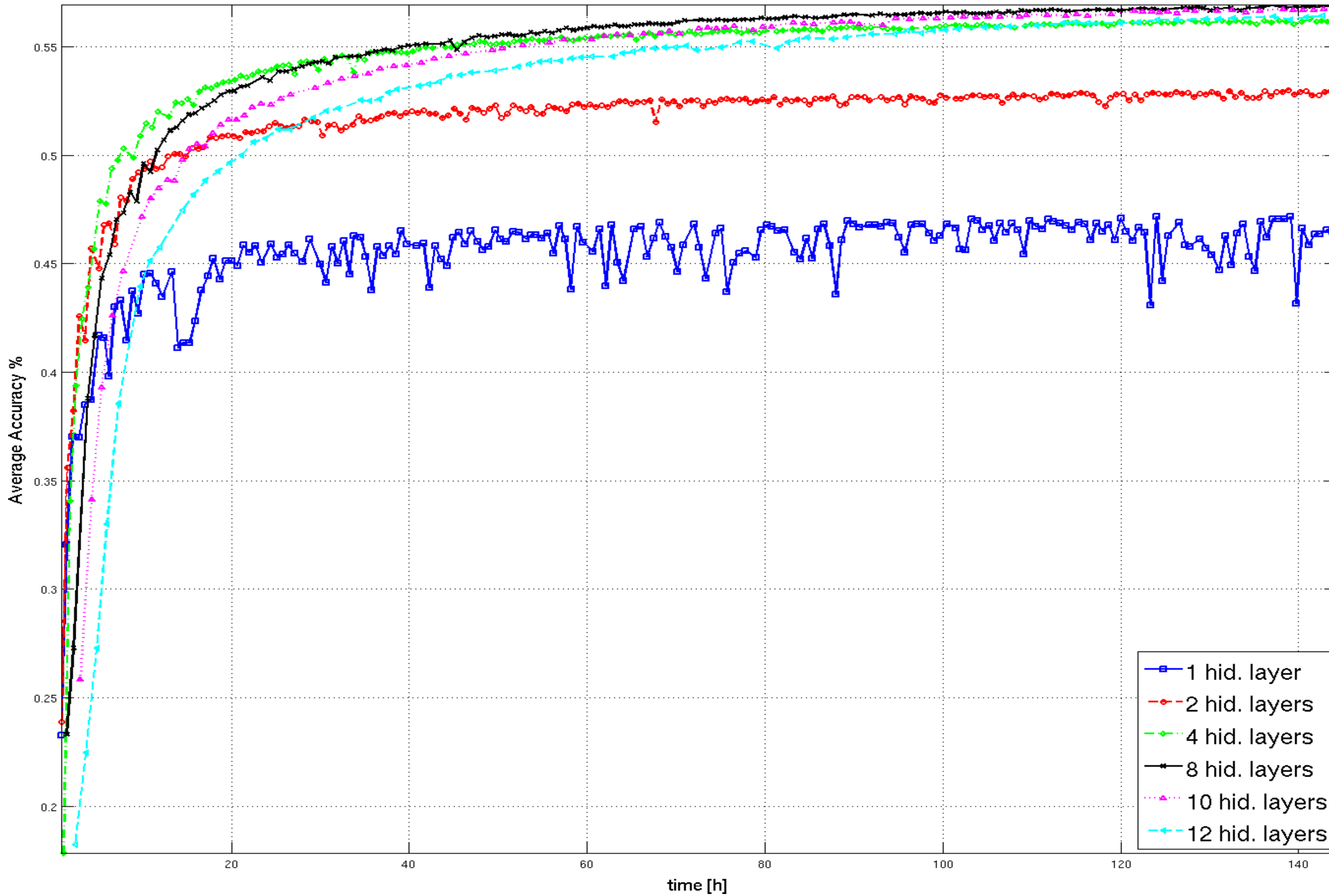
Acoustic Modeling in Speech Recognition (Google)

- A typical speech recognition architecture with DL-based acoustic modeling
  - Features: log energy of a filter bank (e.g. 40 filters)
  - Neural net acoustic modeling (convolutional or not)
  - Input window: typically 10 to 40 acoustic frames
  - Fully-connected neural net: 10 layers, 2000-4000 hidden units/layer
  - But convolutional nets do better....
  - Predicts phone state, typically 2000 to 8000 categories



Mohamed et al. “DBNs for phone recognition” NIPS Workshop 2009
Zeiler et al. “On rectified linear units for speech recognition” ICASSP 2013

Training Curves (Google)



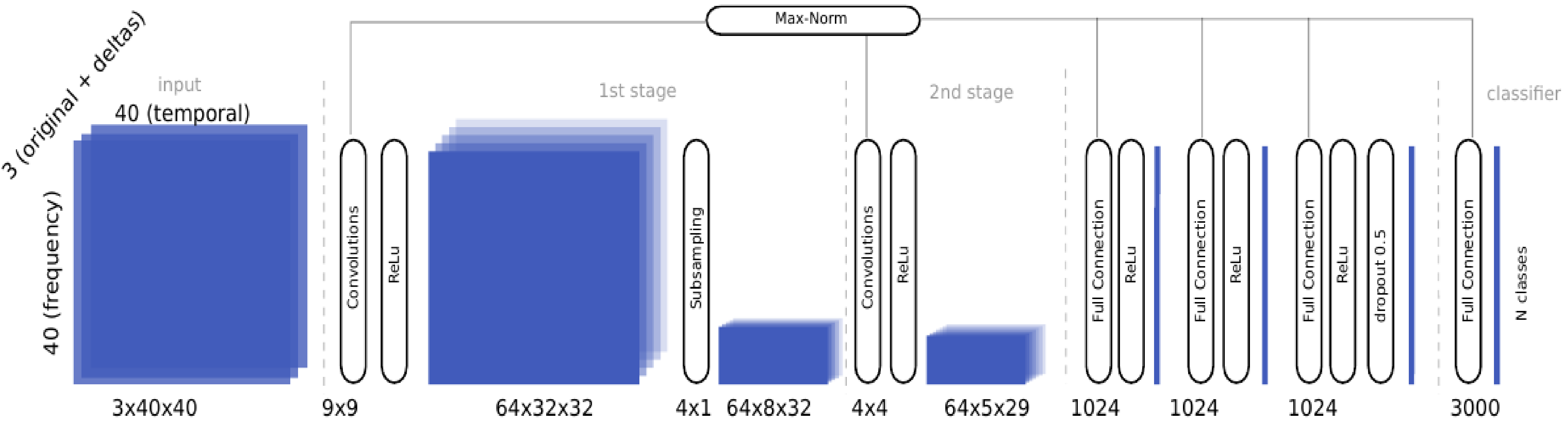
Word Error Rate (Google)

Number of hidden layers	Word Error Rate %
1	16
2	12.8
4	11.4
8	10.9
12	11.1

GMM baseline: 15.4%

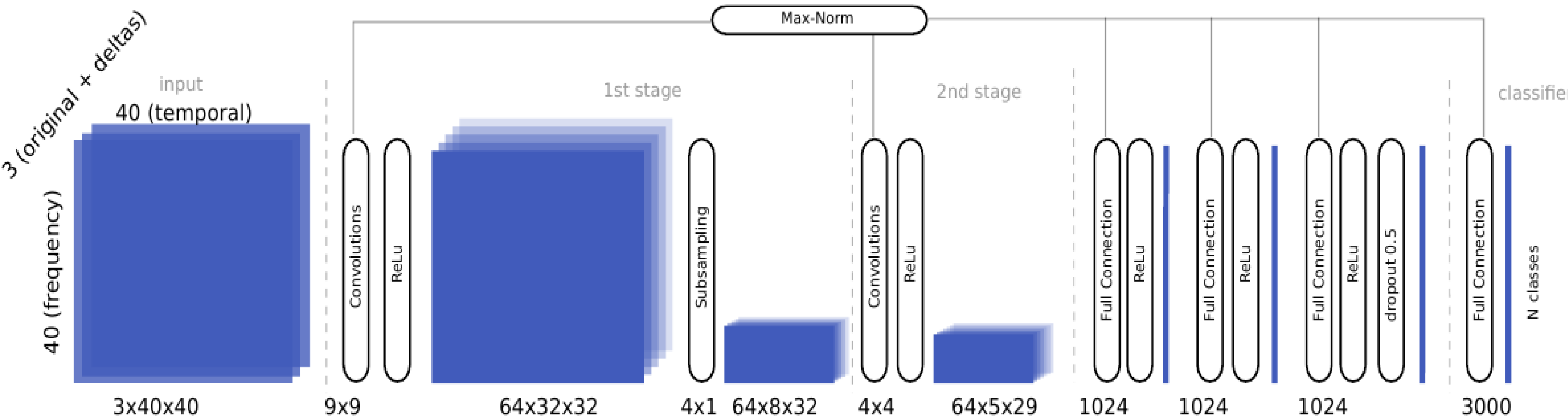
Zeiler et al. “On rectified linear units for speech recognition” ICASSP 2013

Speech Recognition with Convolutional Nets (NYU/IBM)



- Acoustic Model: ConvNet with 7 layers. 54.4 million parameters.
- Classifies acoustic signal into 3000 context-dependent subphones categories
- ReLU units + dropout for last layers
- Trained on GPU. 4 days of training

Speech Recognition with Convolutional Nets (NYU/IBM)

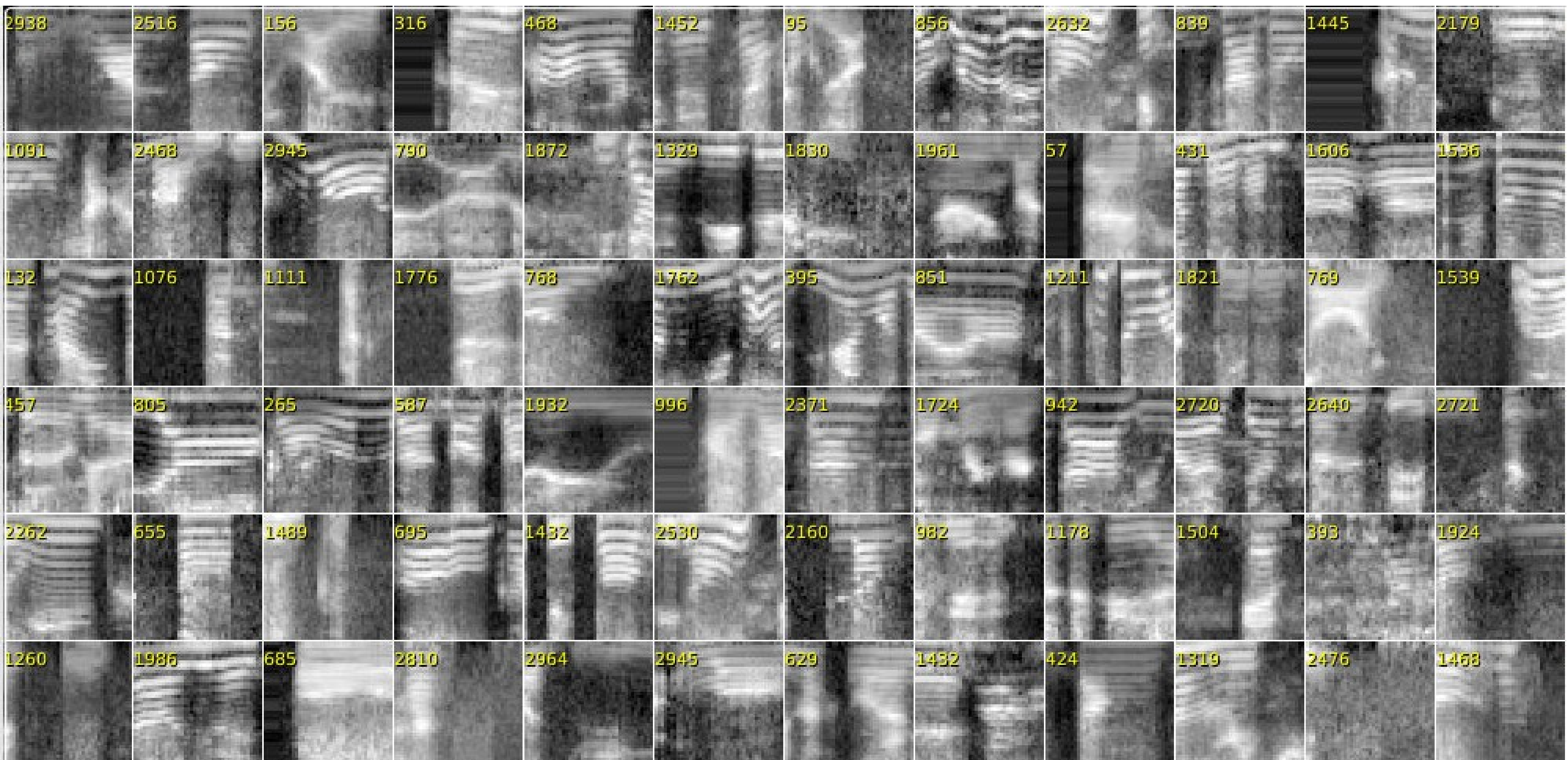


- **Subphone-level classification error (sept 2013):**
  - Cantonese: phone: 20.4% error; subphone: 33.6% error (IBM DNN: 37.8%)
- **Subphone-level classification error (march 2013)**
  - Cantonese: subphone: 36.91%
  - Vietnamese: subphone 48.54%
  - Full system performance (token error rate on conversational speech):
    - 76.2% (52.9% substitution, 13.0% deletion, 10.2% insertion)

Speech Recognition with Convolutional Nets (NYU/IBM)

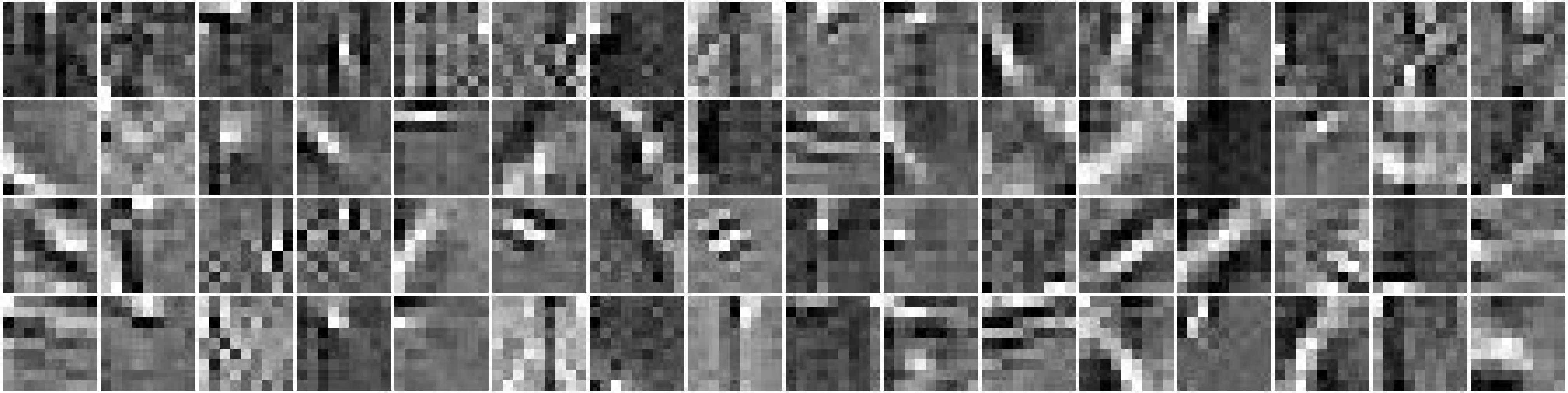
Training samples.

- 40 MEL-frequency Cepstral Coefficients
- Window: 40 frames, 10ms each



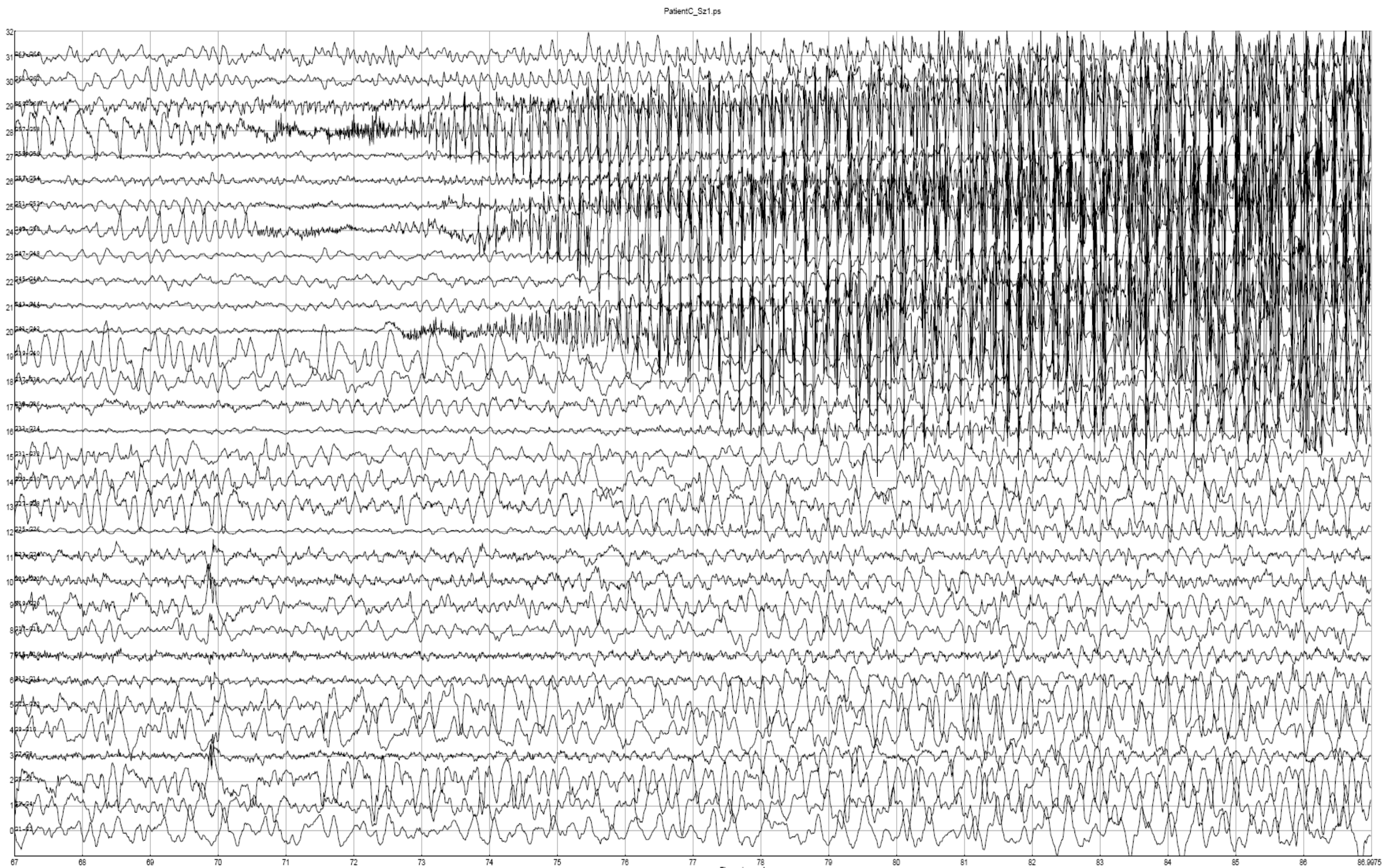
Speech Recognition with Convolutional Nets (NYU/IBM)

- Convolution Kernels at Layer 1:
  - 64 kernels of size 9x9



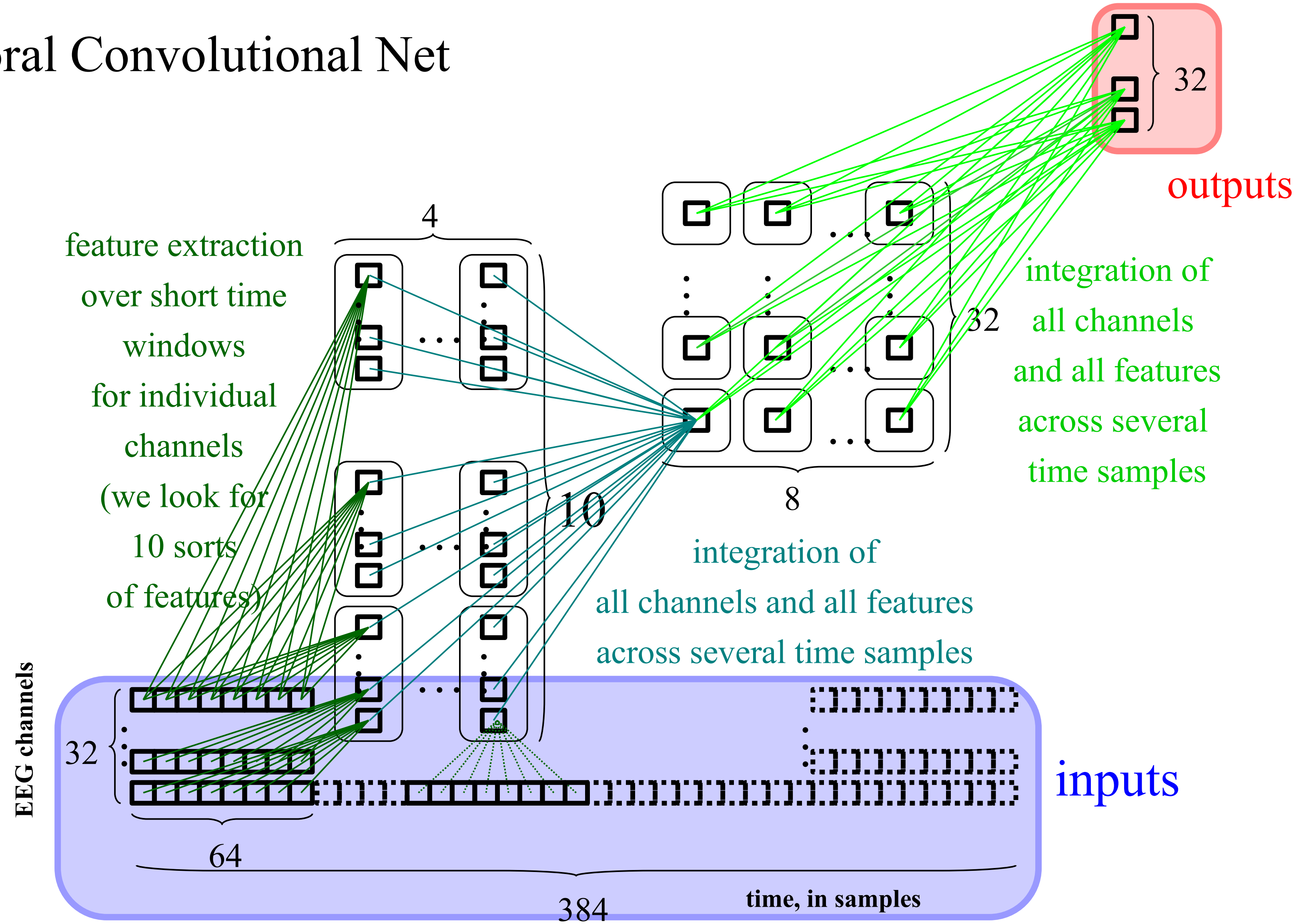
Prediction of Epilepsy Seizures from Intra-Cranial EEG

Piotr Mirowski, Deepak Mahdevan (NYU Neurology), Yann LeCun



Epilepsy Prediction

Temporal Convolutional Net

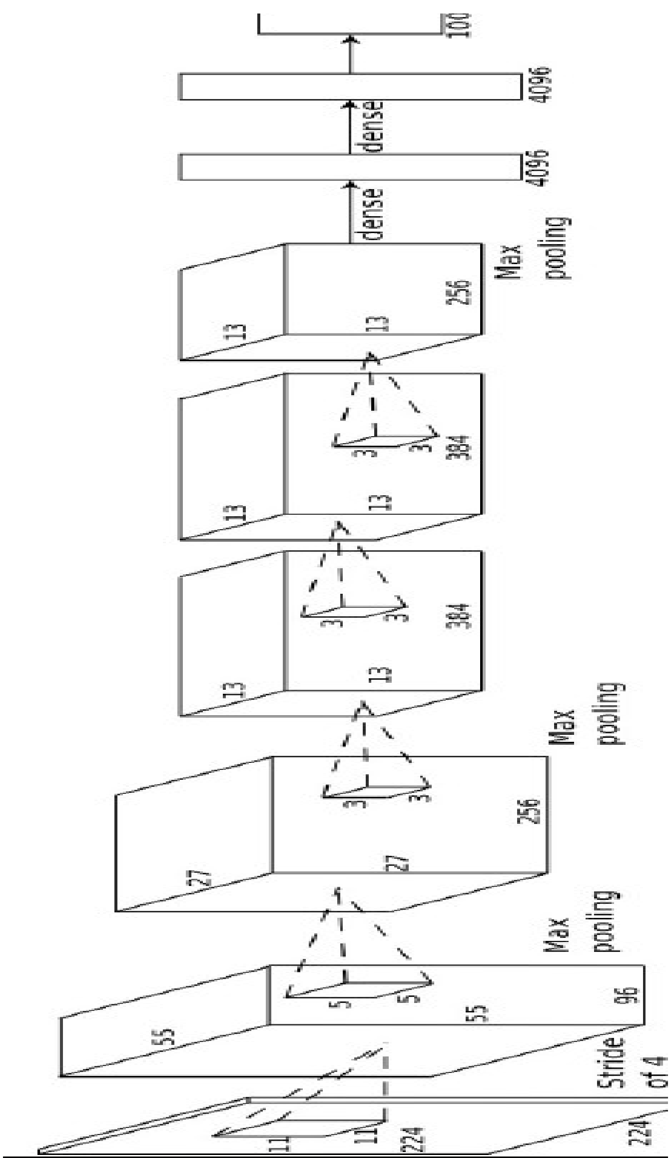


Convolutional Networks
In
Visual Object Recognition

Object Recognition [Krizhevsky, Sutskever, Hinton 2012]

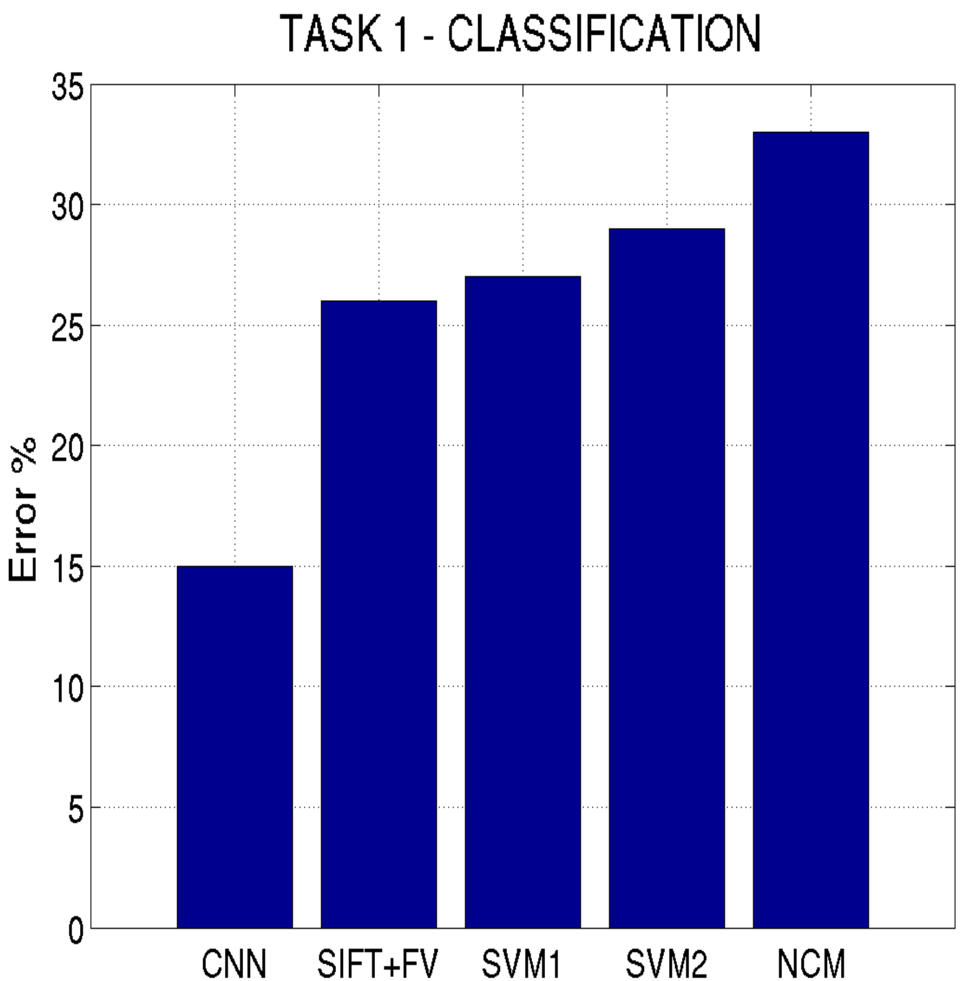
- Won the 2012 ImageNet LSVRC. 60 Million parameters, 832M MAC ops

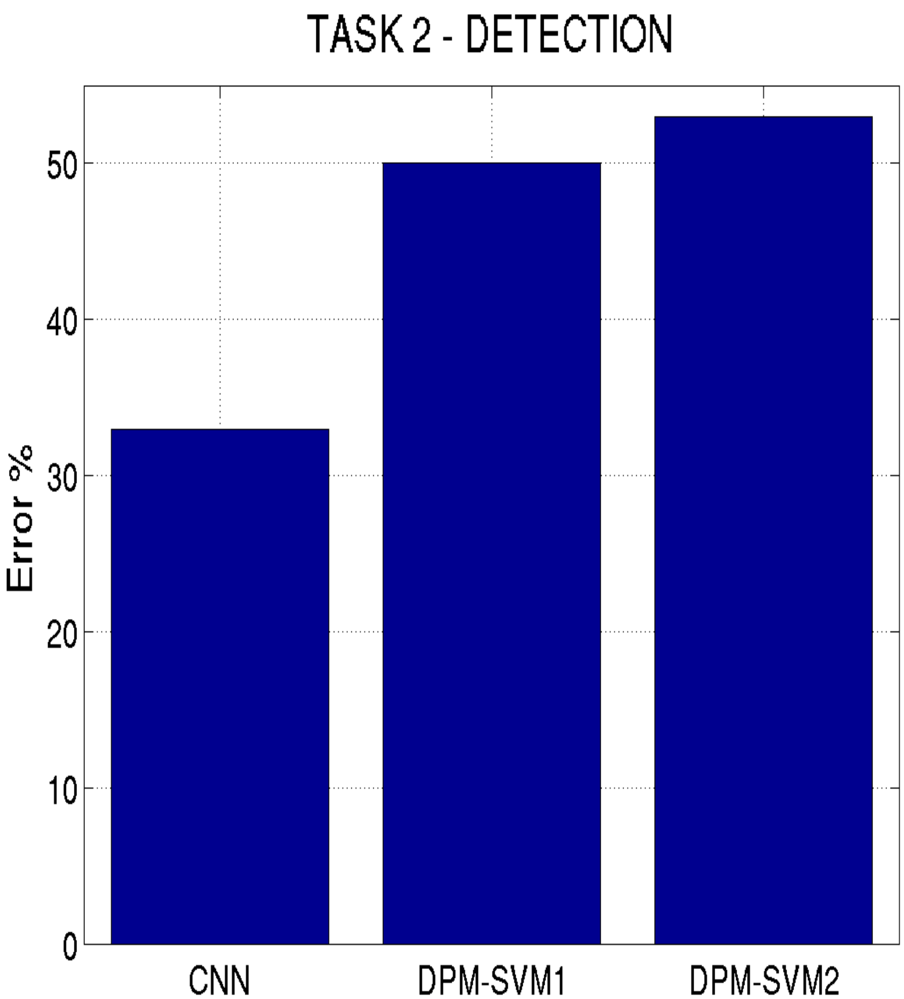
4M	FULL CONNECT	4Mflop
16M	FULL 4096/ReLU	16M
37M	FULL 4096/ReLU	37M
	MAX POOLING	
442K	CONV 3x3/ReLU 256fm	74M
1.3M	CONV 3x3ReLU 384fm	224M
884K	CONV 3x3/ReLU 384fm	149M
	MAX POOLING 2x2sub	
	LOCAL CONTRAST NORM	
307K	CONV 11x11/ReLU 256fm	223M
	MAX POOL 2x2sub	
	LOCAL CONTRAST NORM	
35K	CONV 11x11/ReLU 96fm	105M



Object Recognition: ILSVRC 2012 results

- ImageNet Large Scale Visual Recognition Challenge
- 1000 categories, 1.5 Million labeled training samples



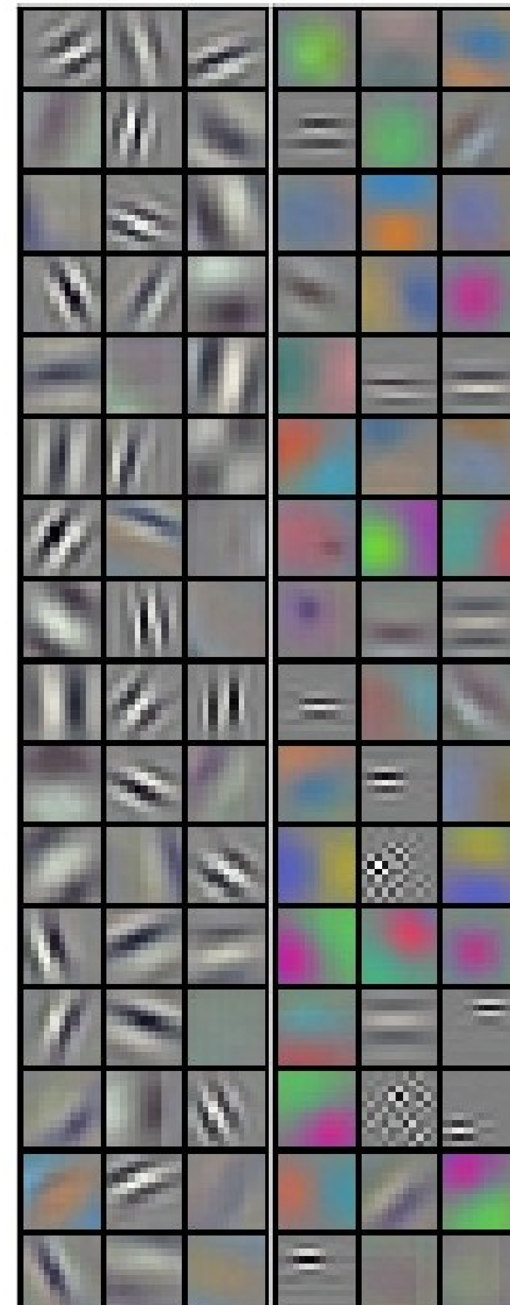


Object Recognition [Krizhevsky, Sutskever, Hinton 2012]

- Method: large convolutional net
  - 650K neurons, 832M synapses, 60M parameters
  - Trained with backprop on GPU
  - Trained “with all the tricks Yann came up with in the last 20 years, plus dropout” (Hinton, NIPS 2012)
  - Rectification, contrast normalization,...
- Error rate: 15% (whenever correct class isn't in top 5)
- Previous state of the art: 25% error

- A REVOLUTION IN COMPUTER VISION

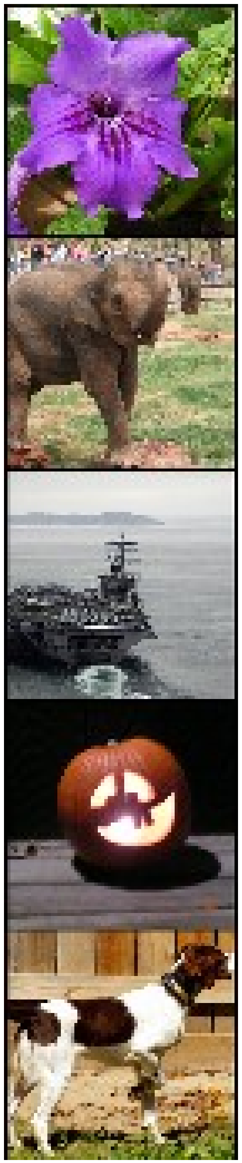
- Acquired by Google in Jan 2013
- Deployed in Google+ Photo Tagging in May 2013



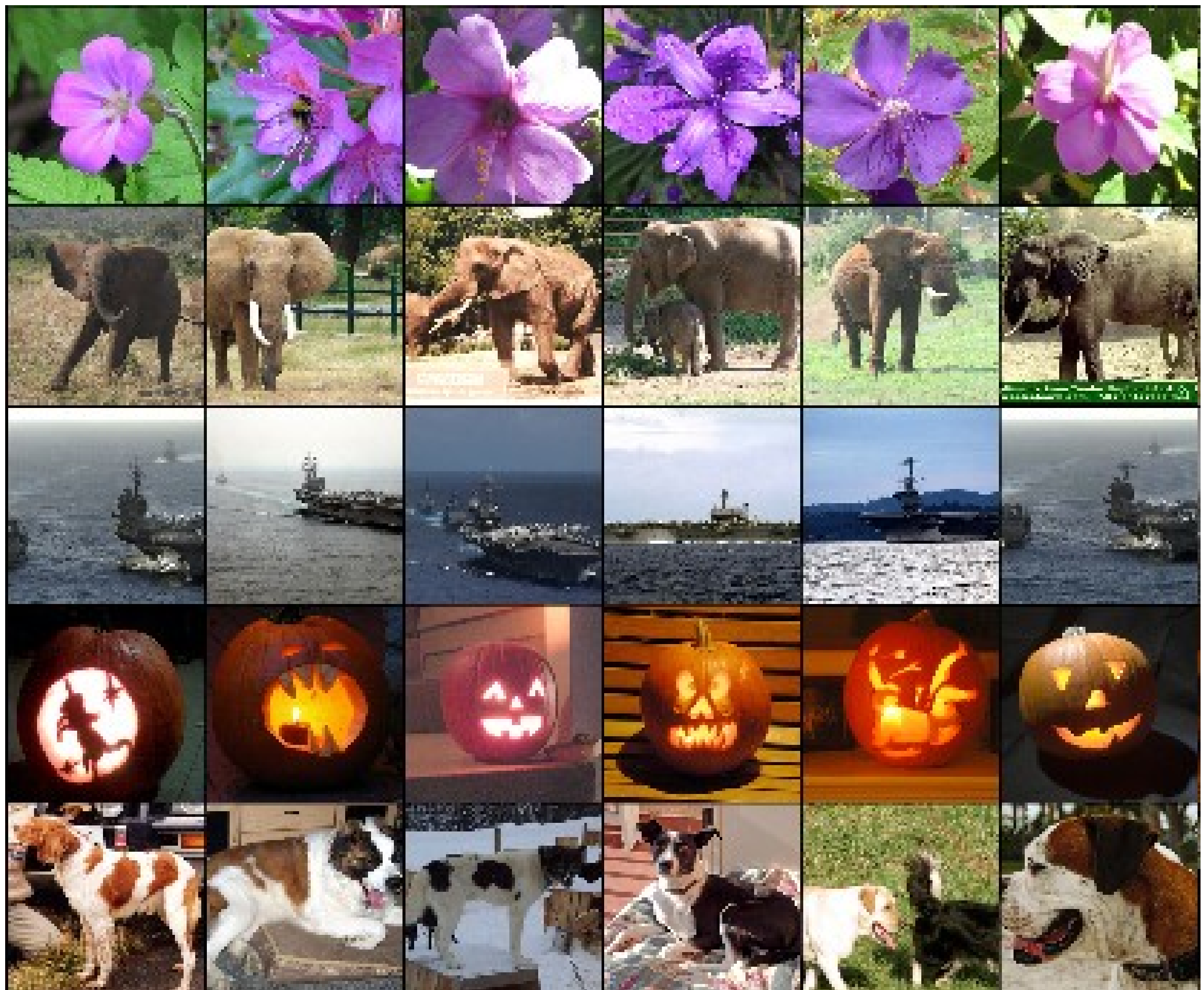
Object Recognition [Krizhevsky, Sutskever, Hinton 2012]

Object Recognition [Krizhevsky, Sutskever, Hinton 2012]

TEST IMAGE

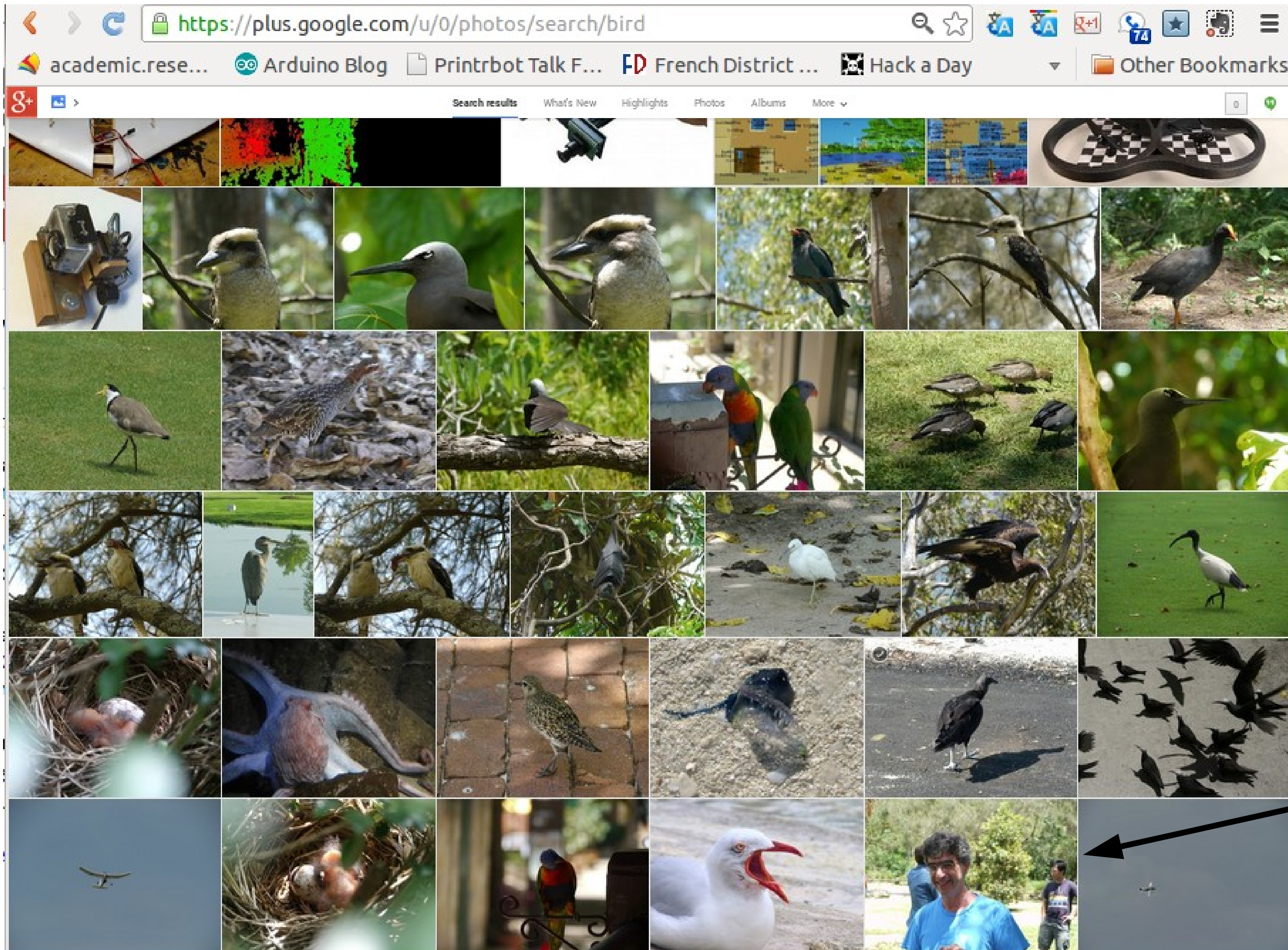


RETRIEVED IMAGES



ConvNet-Based Google+ Photo Tagger

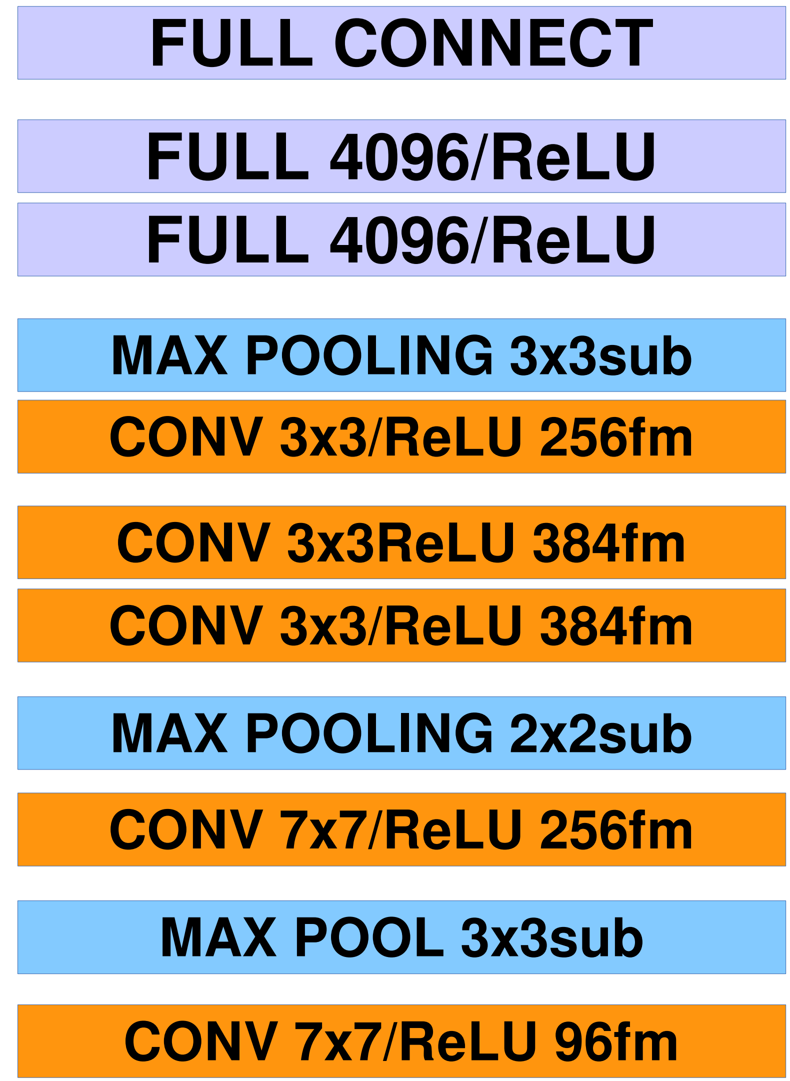
- Searched my personal collection for "bird"



Samy
Bengio
???

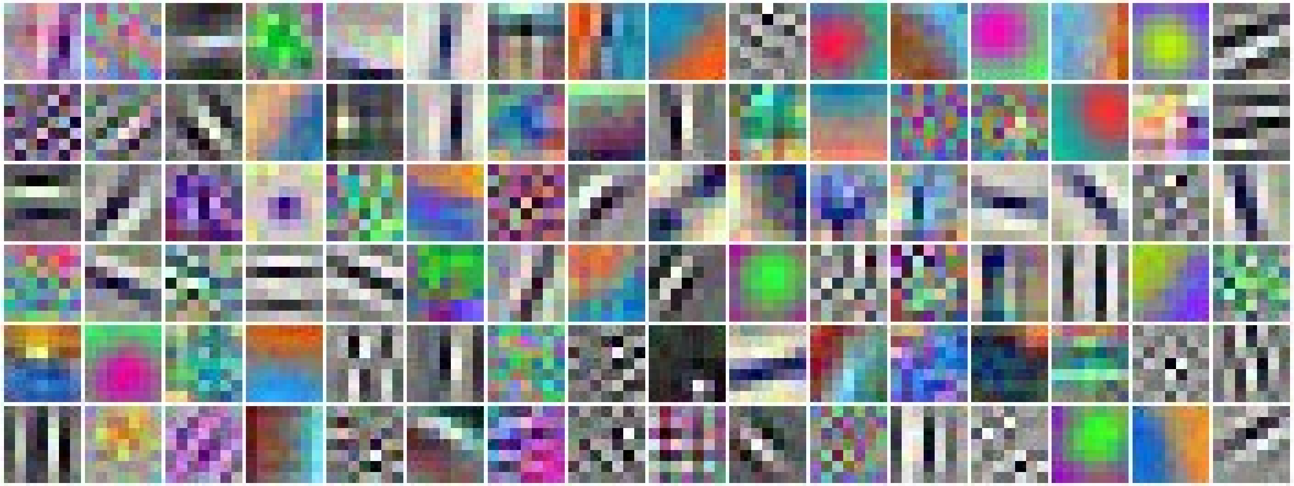
ConvNet Trained on ImageNet at NYU

- [Sermanet, Zhang, Zhang, LeCun 2013, in preparation]
- Trained on GPU using Torch7
- Uses a number of new tricks
- 17.4% error (top 5) on ImageNet with a single network
  - vs Krizhevksy's 18.2%
- Real-time demo!

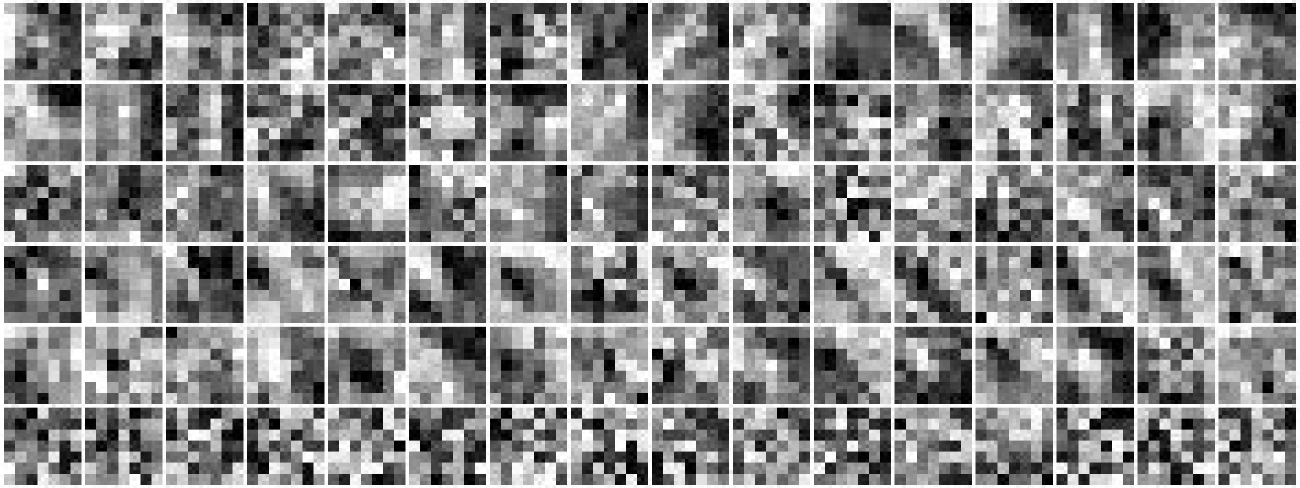


Layer 1 and Layer 2 Kernels

- Layer 1: 3x64 kernels, RGB->64 feature maps, 7x7 Kernels

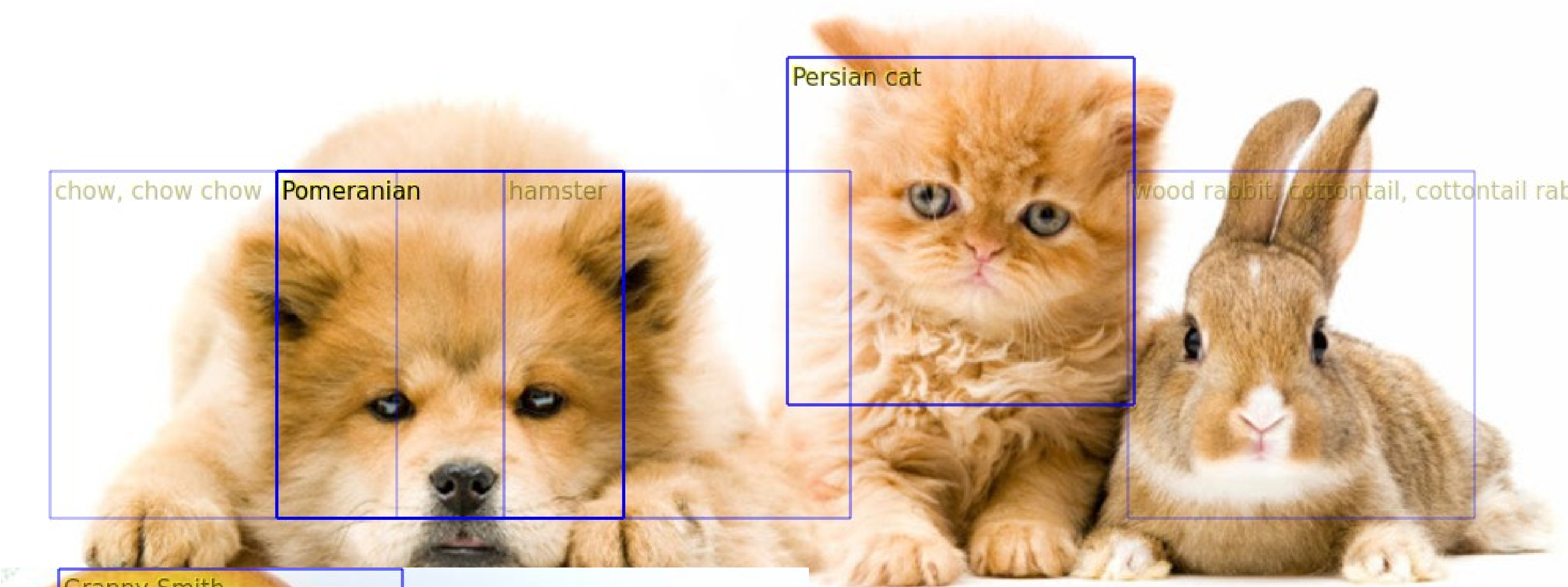


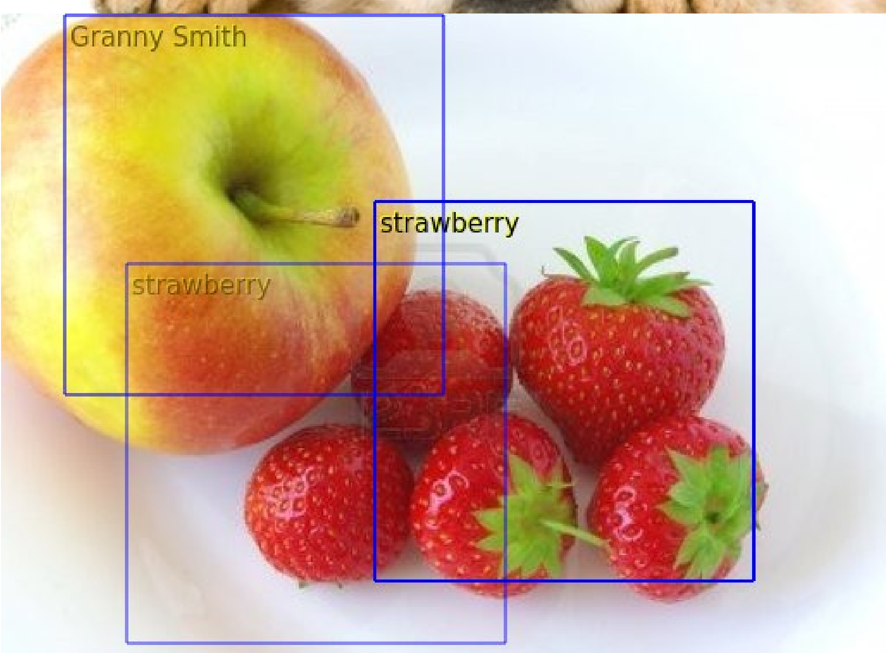
- Layer 2: 64x256 kernels, 9x9

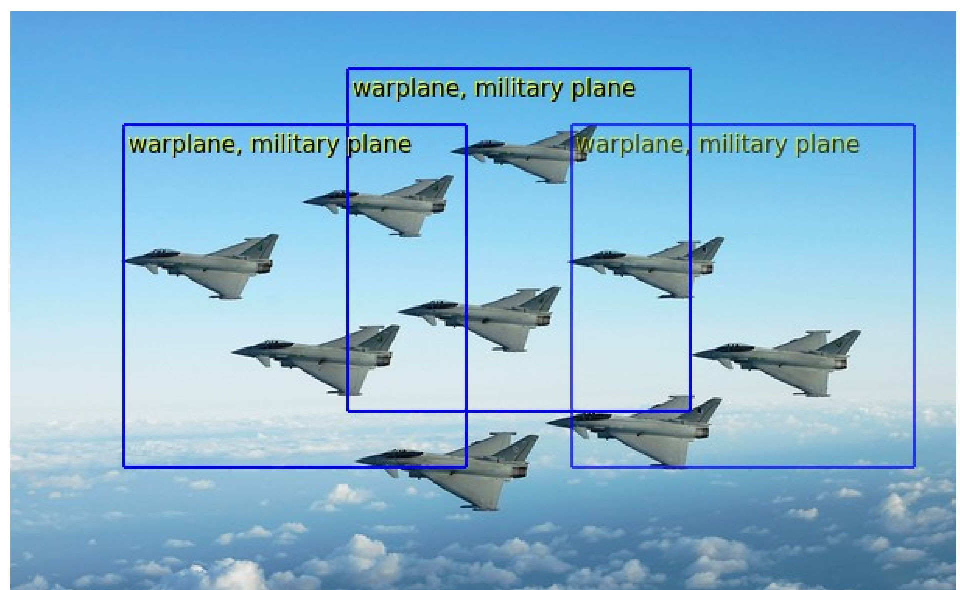


Results: detection with sliding window

- Network trained for recognition with 1000 ImageNet classes

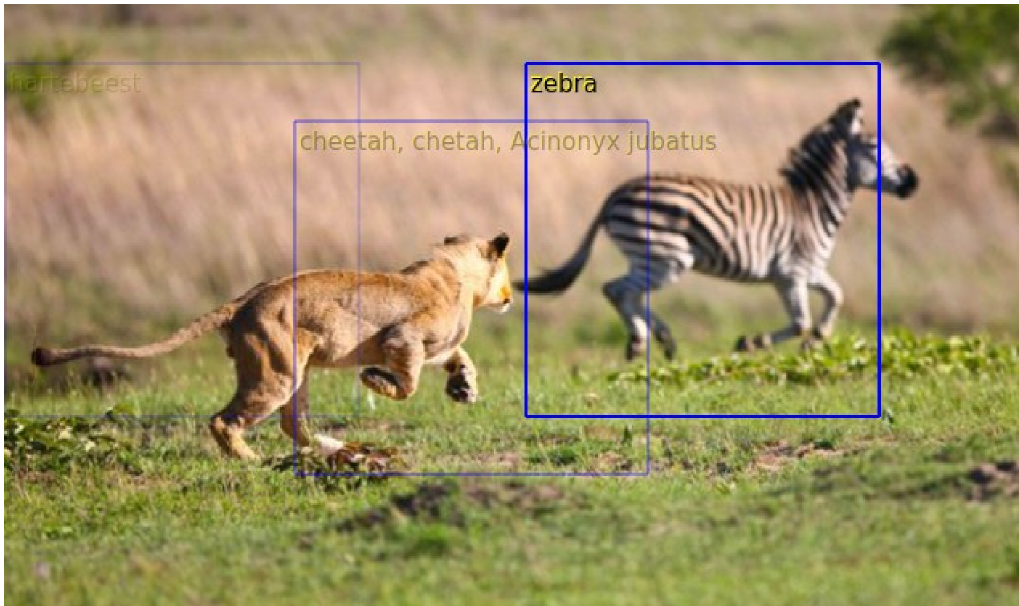


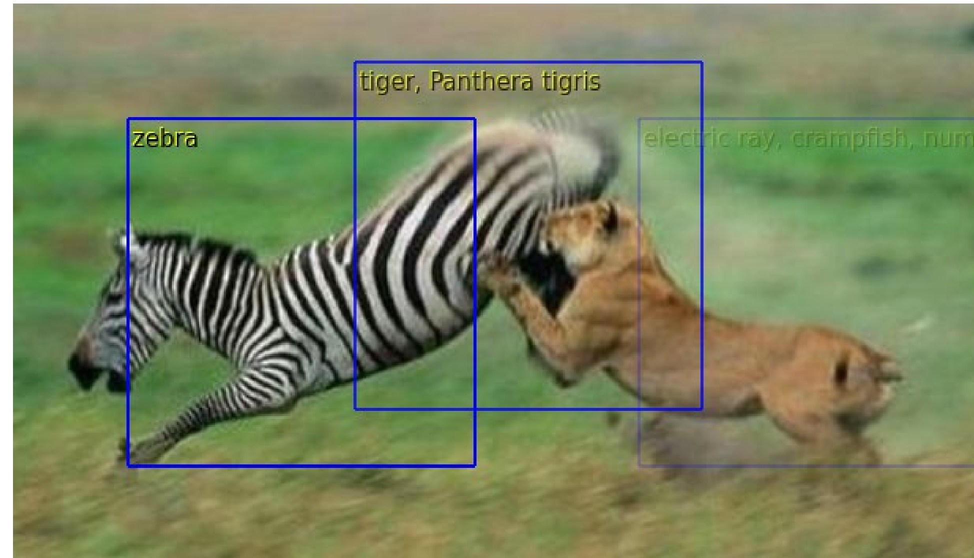


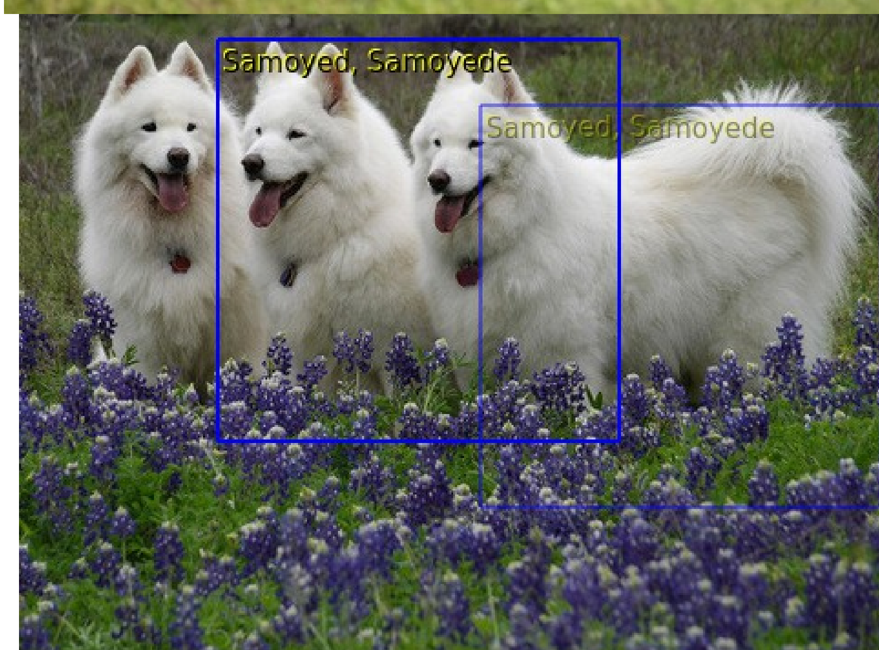


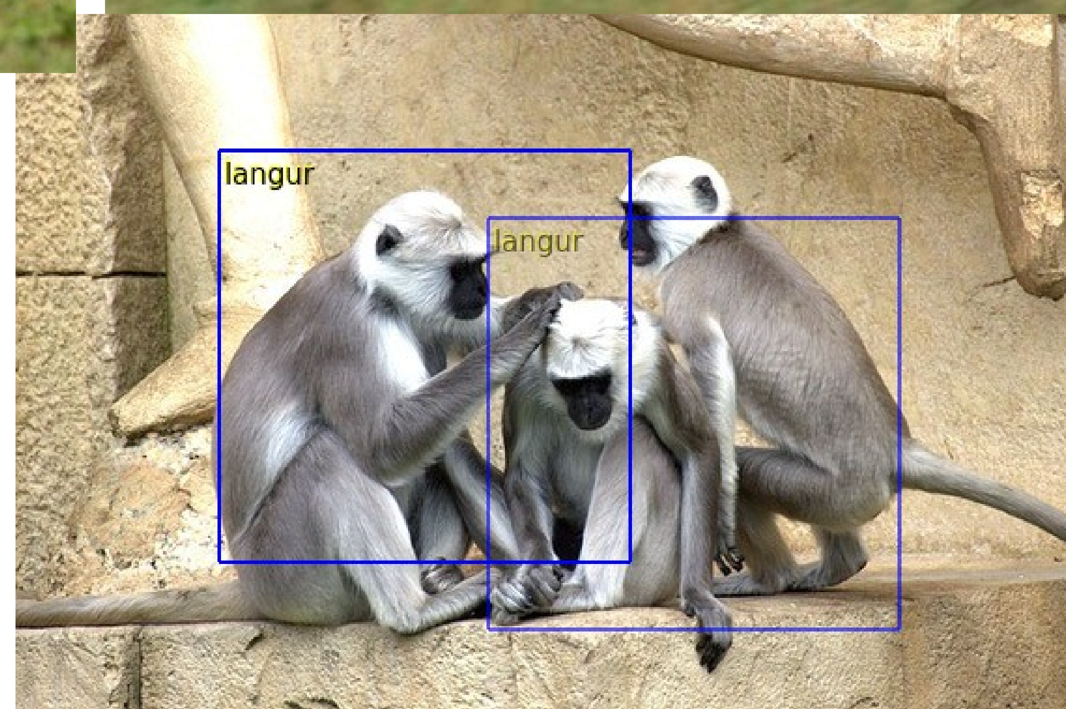
Results: detection with sliding window

- Network trained for recognition with 1000 ImageNet classes

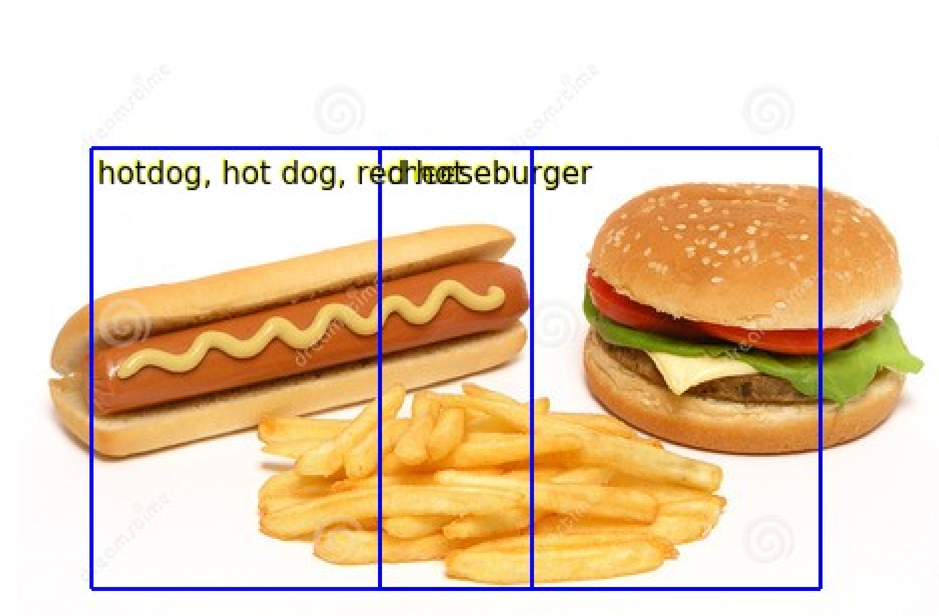


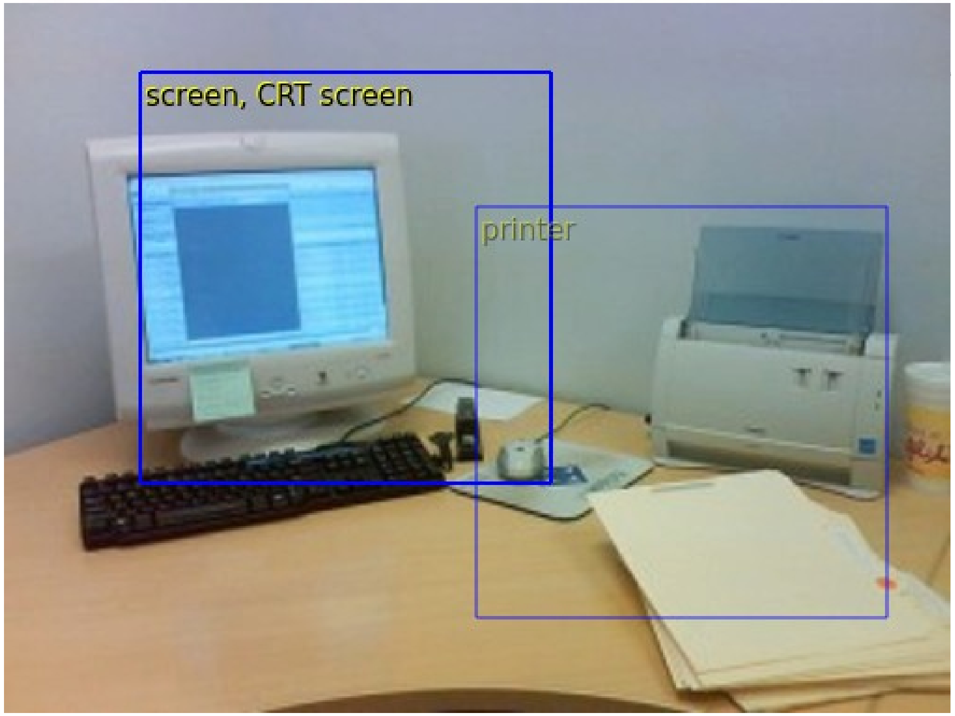


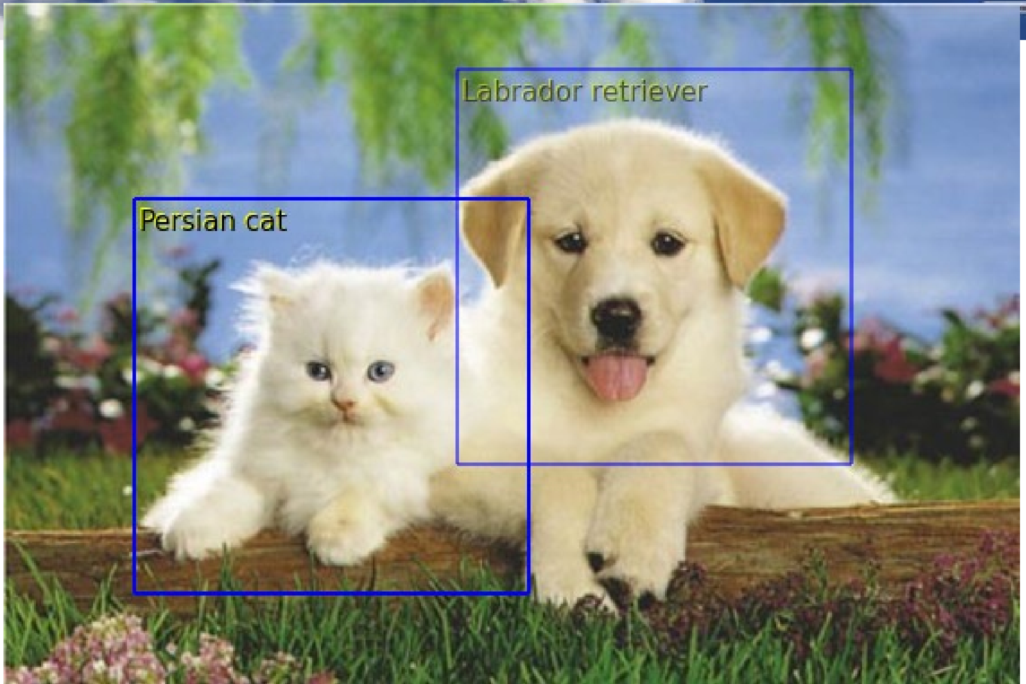




Results: detection with sliding window

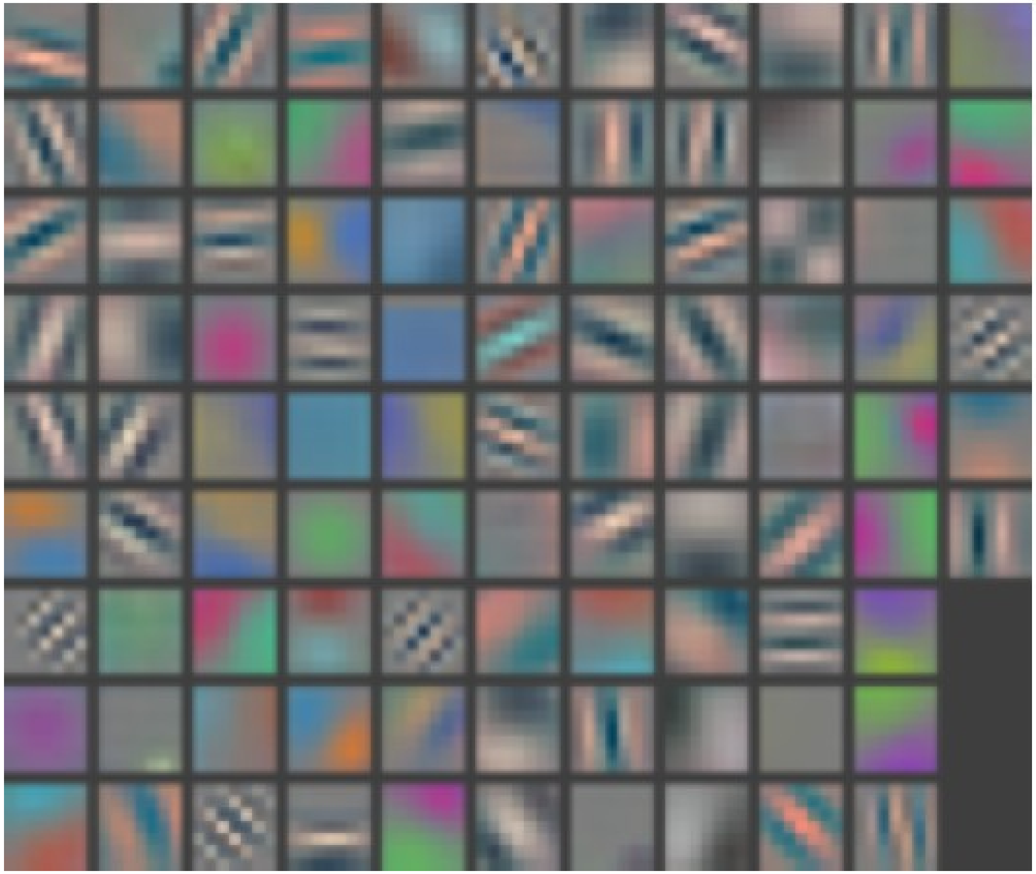






Another ImageNet-trained ConvNet at NYU [Zeiler & Fergus 2013]

- Convolutional Net with 8 layers, input is 224x224 pixels
  - conv-pool-conv-pool-conv-conv-conv-full-full-full
  - Rectified-Linear Units (ReLU): y = max(0,x)
  - Divisive contrast normalization across features [Jarrett et al. ICCV 2009]
- Trained on ImageNet 2012 training set
  - 1.3M images, 1000 classes
  - 10 different crops/flips per image
- Regularization: Dropout
  - [Hinton 2012]
  - zeroing random subsets of units
- Stochastic gradient descent
  - for 70 epochs (7-10 days)
  - With learning rate annealing

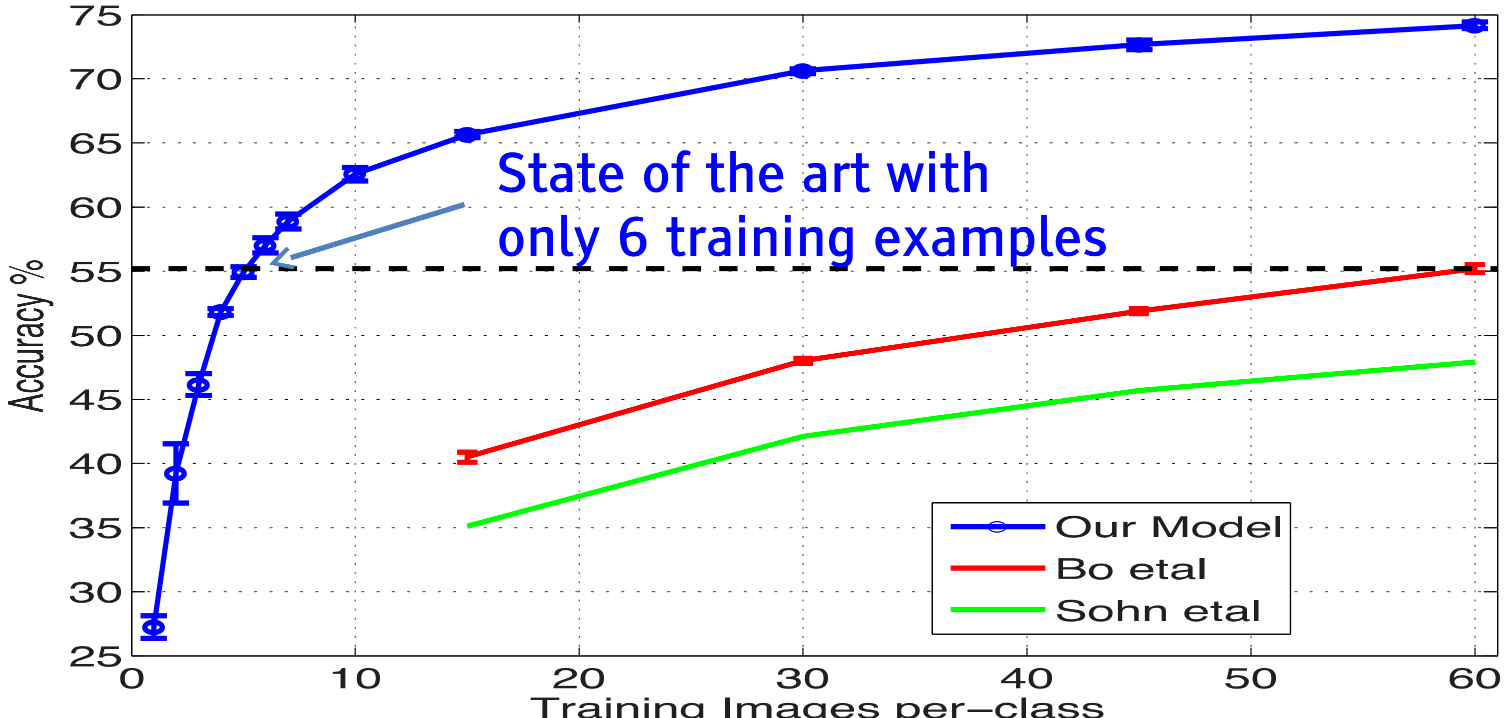


ConvNet trained on ImageNet [Zeiler & Fergus 2013]

Error %	Val Top-1	Val Top-5	Test Top-5
Deng *et al.* SIFT + FV [7]	— —	— —	26.2
Krizhevsky *et al.* [12], 1 convnet	40.7	18.2	— —
Krizhevsky *et al.* [12], 5 convnets	38.1	16.4	16.4
*Krizhevsky *et al.* [12], 1 convnets	39.0	16.6	— —
*Krizhevsky *et al.* [12], 7 convnets	36.7	15.4	15.3
Our replication of [12], 1 convnet	41.7	19.0	— —
1 convnet - our model	38.4 ± 0.05	16.5 ± 0.05	— —
5 convnets - our model (a)	36.7	15.3	15.3
1 convnet - tweaked model (b)	37.5	16.0	16.1
6 convnets, (a) & (b) combined	**36.0**	**14.7**	**14.8**

Features are generic: Caltech 256

- Network first trained on ImageNet.
- Last layer chopped off
- Last layer trained on Caltech 256,
- first layers N-1 kept fixed.
- State of the art accuracy with only 6 training samples/class



# Train	Acc % 15/class	Acc % 30/class	Acc % 45/class	Acc % 60/class
Sohn *et al.* [16]	35.1	42.1	45.7	47.9
Bo *et al.* [3]	40.5 ± 0.4	48.0 ± 0.2	51.9 ± 0.2	55.2 ± 0.3
Non-pretr.	9.0 ± 1.4	22.5 ± 0.7	31.2 ± 0.5	38.8 ± 1.4
ImageNet-pretr.	**65.7 ± 0.2**	**70.6 ± 0.2**	**72.7 ± 0.4**	**74.2 ± 0.3**

3: [Bo, Ren, Fox. CVPR, 2013]   16: [Sohn, Jung, Lee, Hero ICCV 2011]

Features are generic: PASCAL VOC 2012

- Network first trained on ImageNet.
- Last layer trained on Pascal VOC, keeping N-1 first layers fixed.

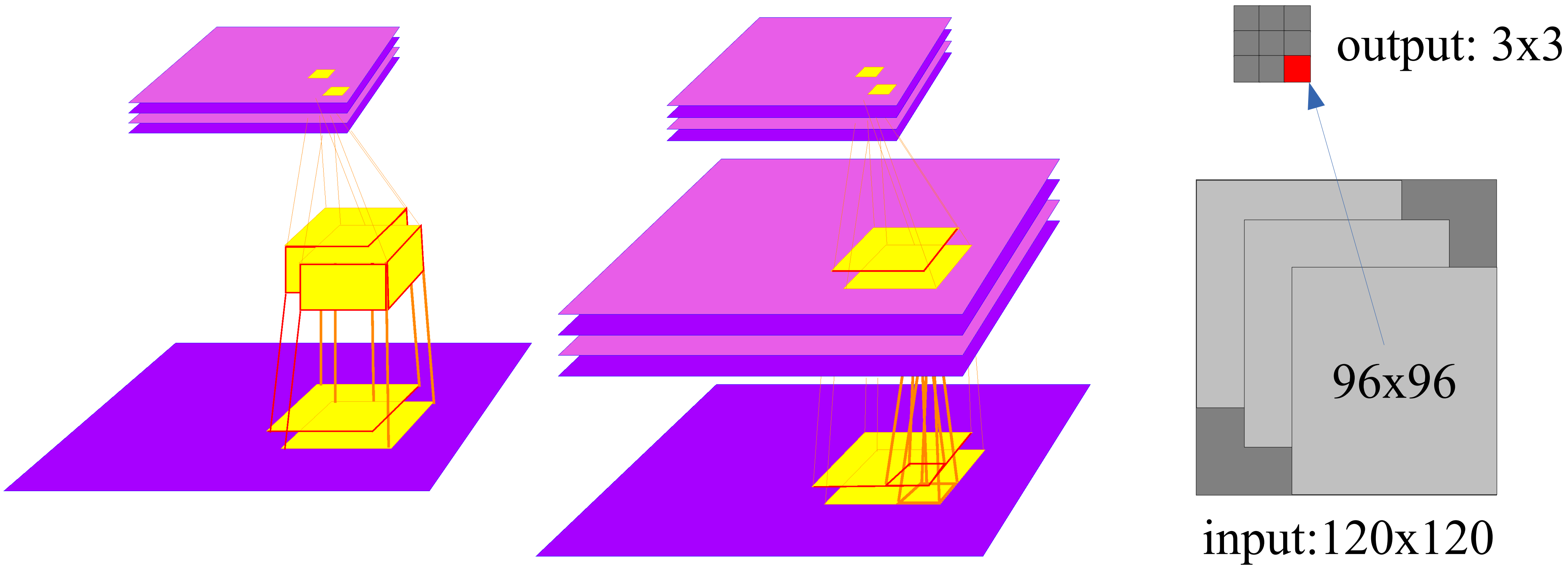
Acc %	[15]	[19]	Ours	Acc %	[15]	[19]	Ours
Airplane	92.0	**97.3**	96.0	Dining table	63.2	**77.8**	67.7
Bicycle	74.2	**84.2**	77.1	Dog	68.9	83.0	**87.8**
Bird	73.0	80.8	**88.4**	Horse	78.2	**87.5**	86.0
Boat	77.5	85.3	**85.5**	Motorbike	81.0	**90.1**	85.1
Bottle	54.3	**60.8**	55.8	Person	91.6	**95.0**	90.9
Bus	85.2	**89.9**	85.8	Potted plant	55.9	**57.8**	52.2
Car	81.9	**86.8**	78.6	Sheep	69.4	79.2	**83.6**
Cat	76.4	89.3	**91.2**	Sofa	65.4	**73.4**	61.1
Chair	65.2	**75.4**	65.0	Train	86.7	**94.5**	91.8
Cow	63.2	**77.8**	74.4	Tv/monitor	77.4	**80.7**	76.1
Mean	74.3	**82.2**	79.0	# won	0	**15**	5

[15] K. Sande, J. Uijlings, C. Snoek, and A. Smeulders. Hybrid coding for selective search. In PASCAL VOC Classification Challenge 2012,

[19] S. Yan, J. Dong, Q. Chen, Z. Song, Y. Pan, W. Xia, Z. Huang, Y. Hua, and S. Shen. Generalized hierarchical matching for sub-category aware object classification. In PASCAL VOC Classification Challenge 2012

Convolutional Networks
In
Image Segmentation
And Object Detection

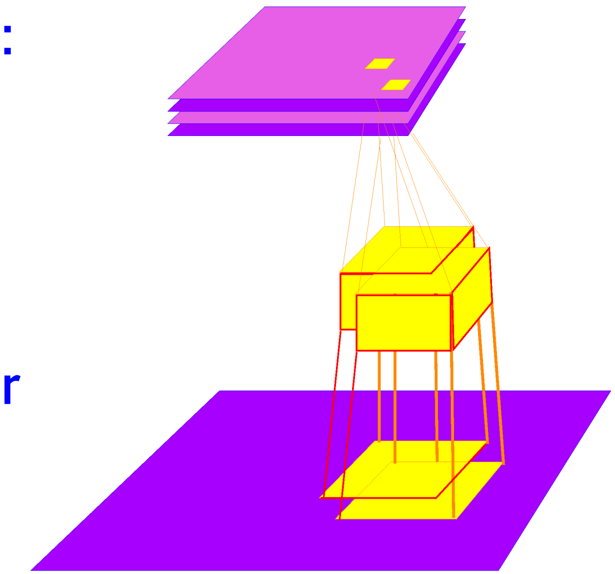
Applying a ConvNet on Sliding Windows is Very Cheap!

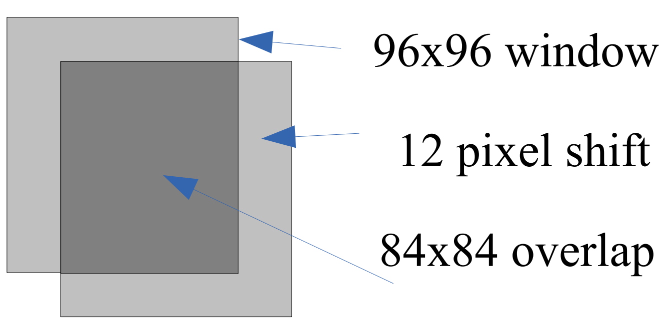


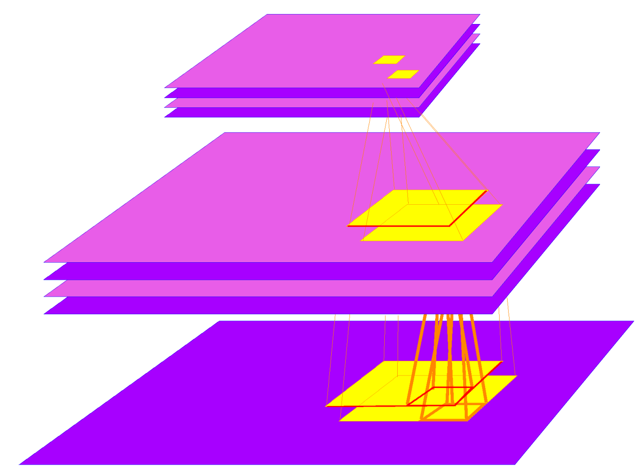
- Traditional Detectors/Classifiers must be applied to every location on a large input image, at multiple scales.
- Convolutional nets can replicated over large images very cheaply.
- The network is applied to multiple scales spaced by 1.5.

Building a Detector/Recognizer: Replicated Convolutional Nets

- Computational cost for replicated convolutional net:
  - 96x96 -> 4.6 million multiply-accumulate operations
  - 120x120 -> 8.3 million multiply-accumulate ops
  - 240x240 -> 47.5 million multiply-accumulate ops
  - 480x480 -> 232 million multiply-accumulate ops
- Computational cost for a non-convolutional detector of the same size, applied every 12 pixels:
  - 96x96 -> 4.6 million multiply-accumulate operations
  - 120x120 -> 42.0 million multiply-accumulate operations
  - 240x240 -> 788.0 million multiply-accumulate ops
  - 480x480 -> 5,083 million multiply-accumulate ops

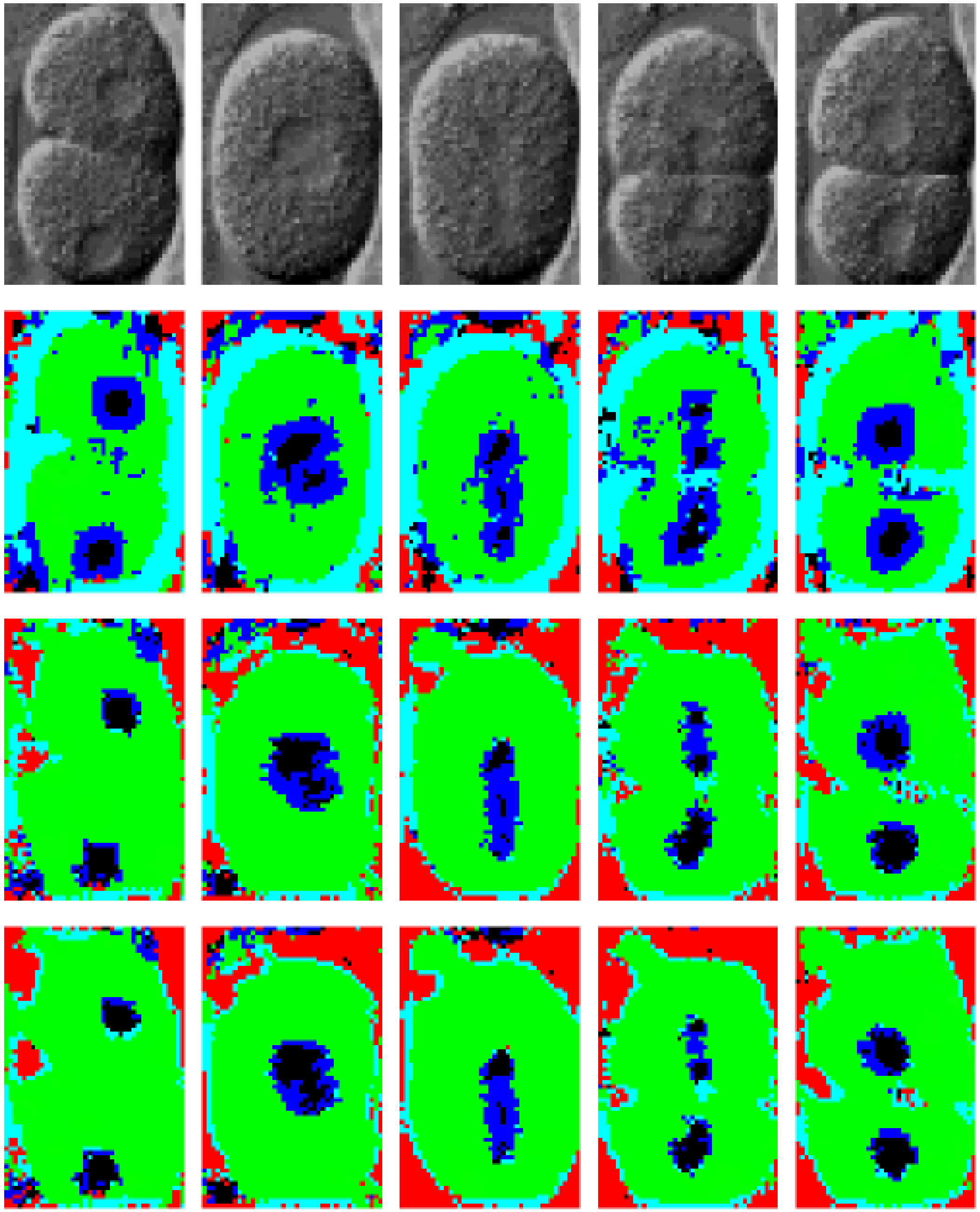






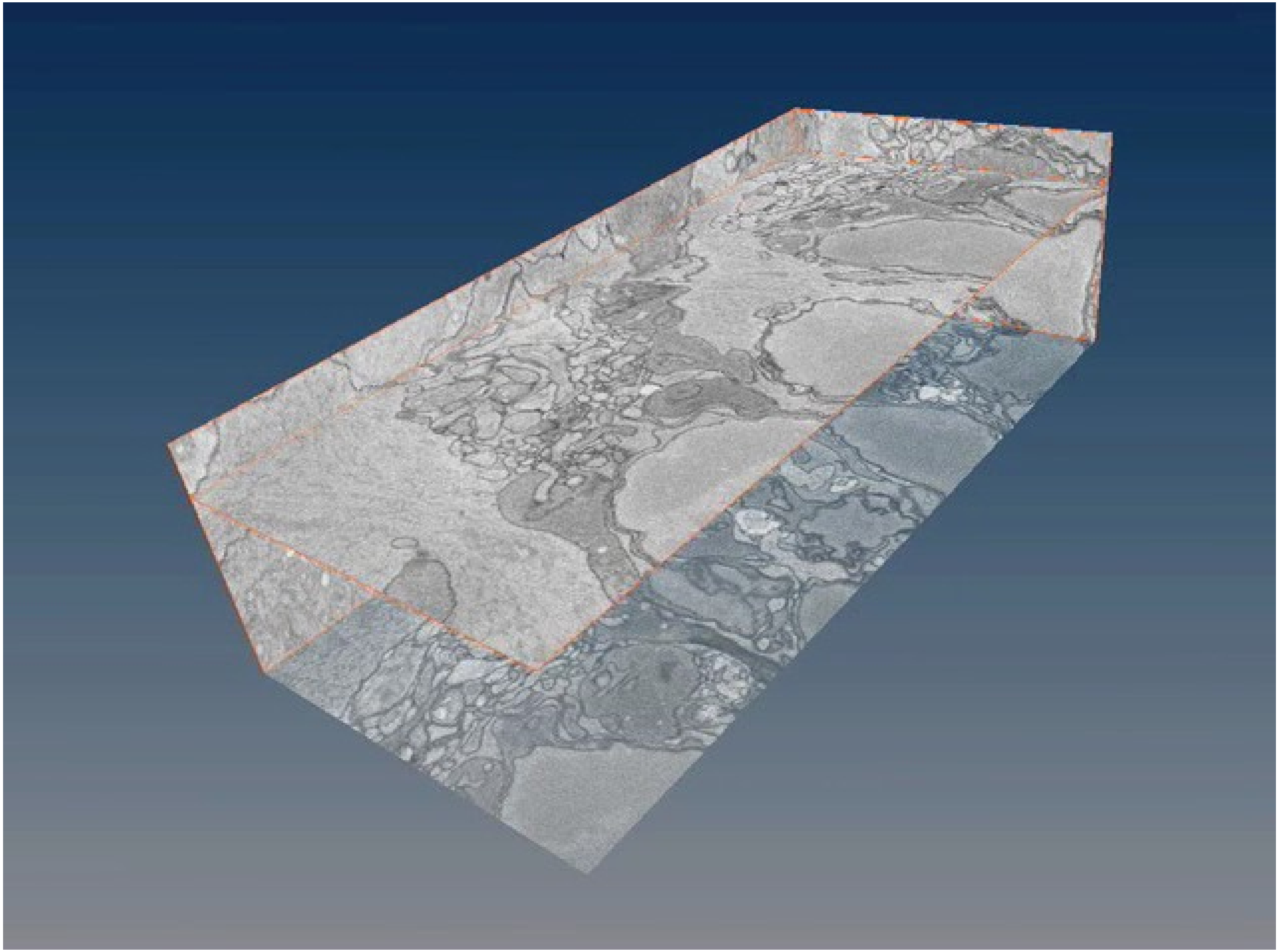
ConvNets for Image Segmentation

- Biological Image Segmentation
  - [Ning et al. IEEE-TIP 2005]
- Pixel labeling with large context using a convnet
- Cleanup using a kind of conditional random field (CRF)
  - Similar to a field of expert



ConvNet in Connectomics
[Jain, Turaga, Seung 2007-present]

- 3D ConvNet
- Volumetric
- Images
-
- Each voxel labeled as "membrane" or "non-membrane" using a 7x7x7 voxel neighborhood

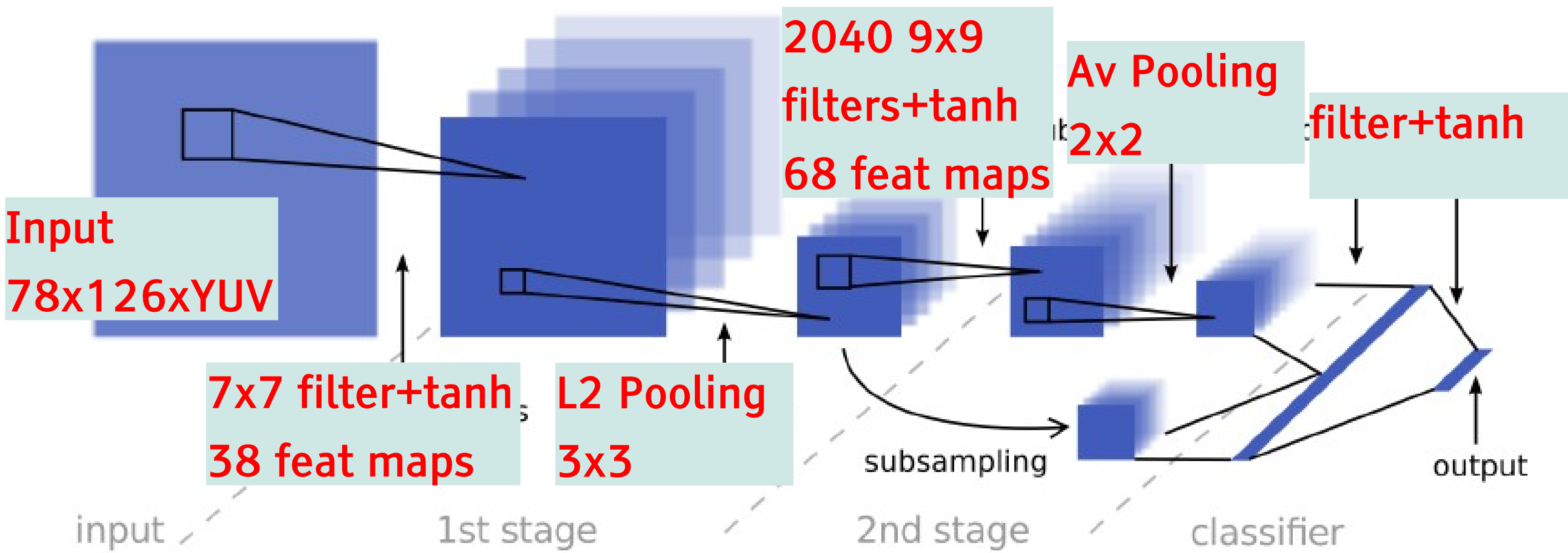


Pedestrian Detection, Face Detection

[Osadchy,Miller LeCun JMLR 2007],[Kavukcuoglu et al. NIPS 2010] [Sermanet et al. CVPR 2013]

ConvNet Architecture with Multi-Stage Features

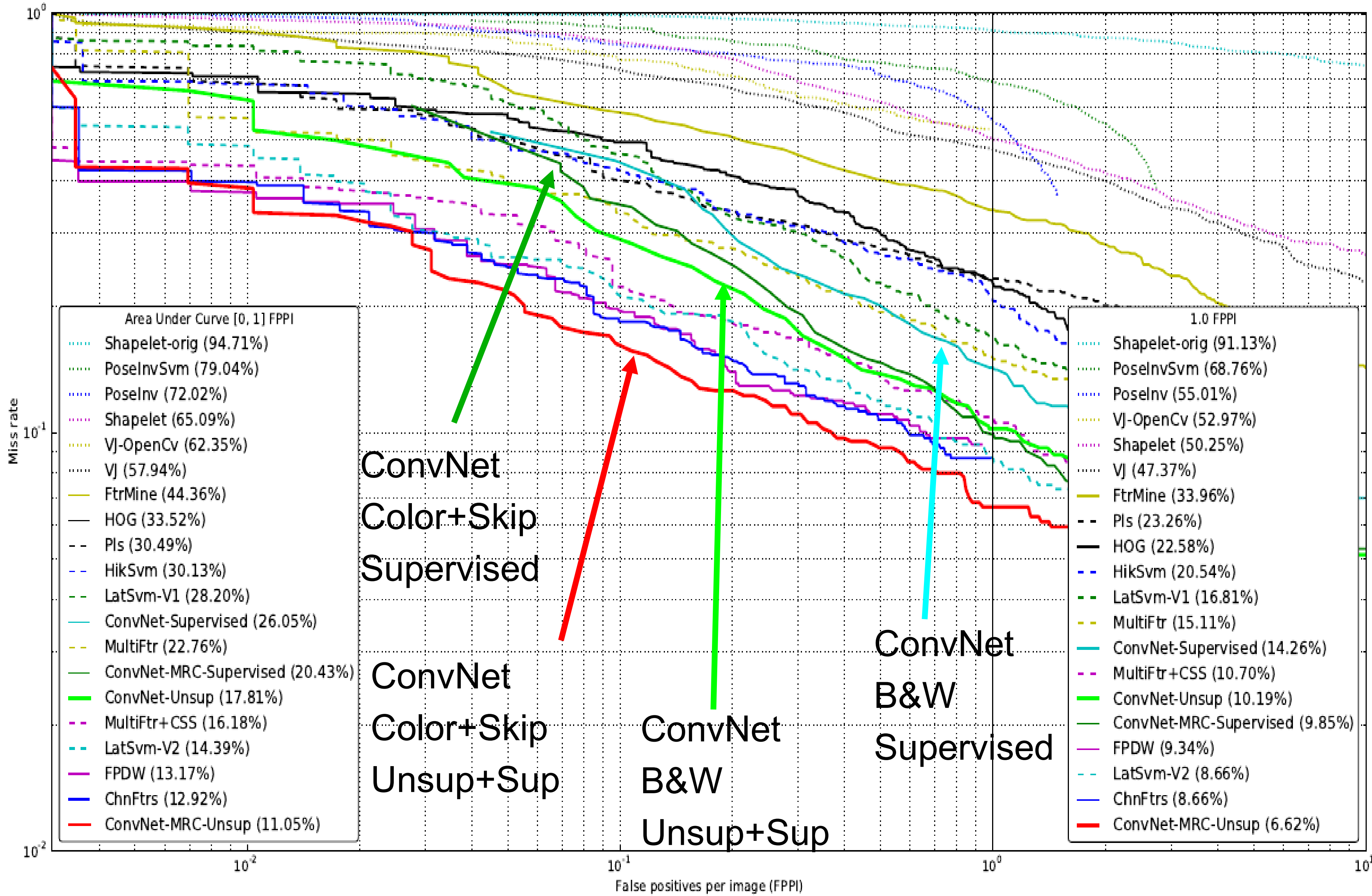
- Feature maps from all stages are pooled/subsampled and sent to the final classification layers
  - Pooled low-level features: good for textures and local motifs
  - High-level features: good for “gestalt” and global shape



Task	Single-Stage features	Multi-Stage features	Improvement %
Pedestrians detection (INRIA)	14.26%	9.85%	31%
Traffic Signs classification (GTSRB) [33]	1.80%	0.83%	54%
House Numbers classification (SVHN) [32]	5.54%	5.36%	3.2%

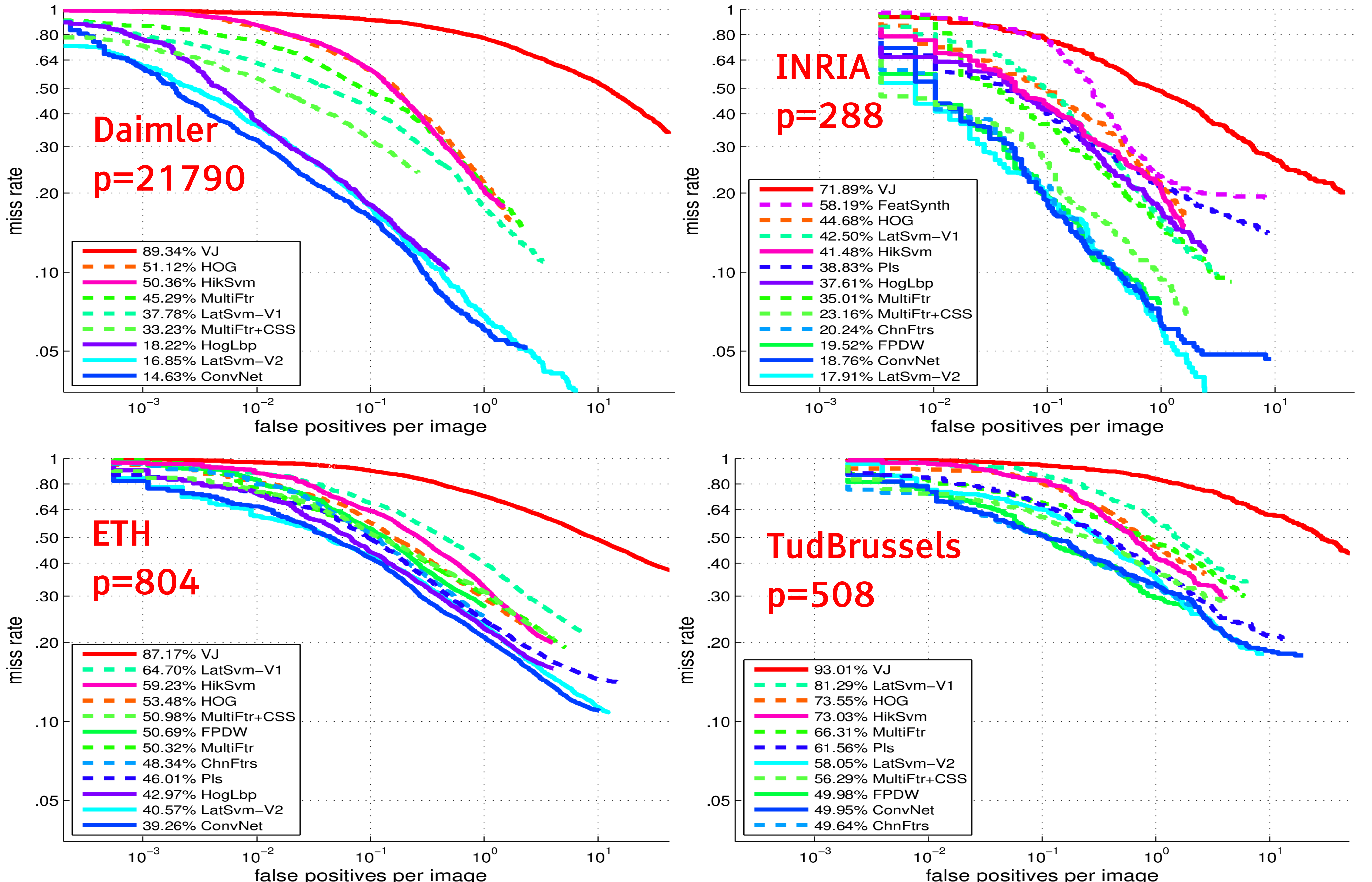
[Sermanet, Chintala, LeCun CVPR 2013]

Pedestrian Detection: INRIA Dataset. Miss rate vs false positives

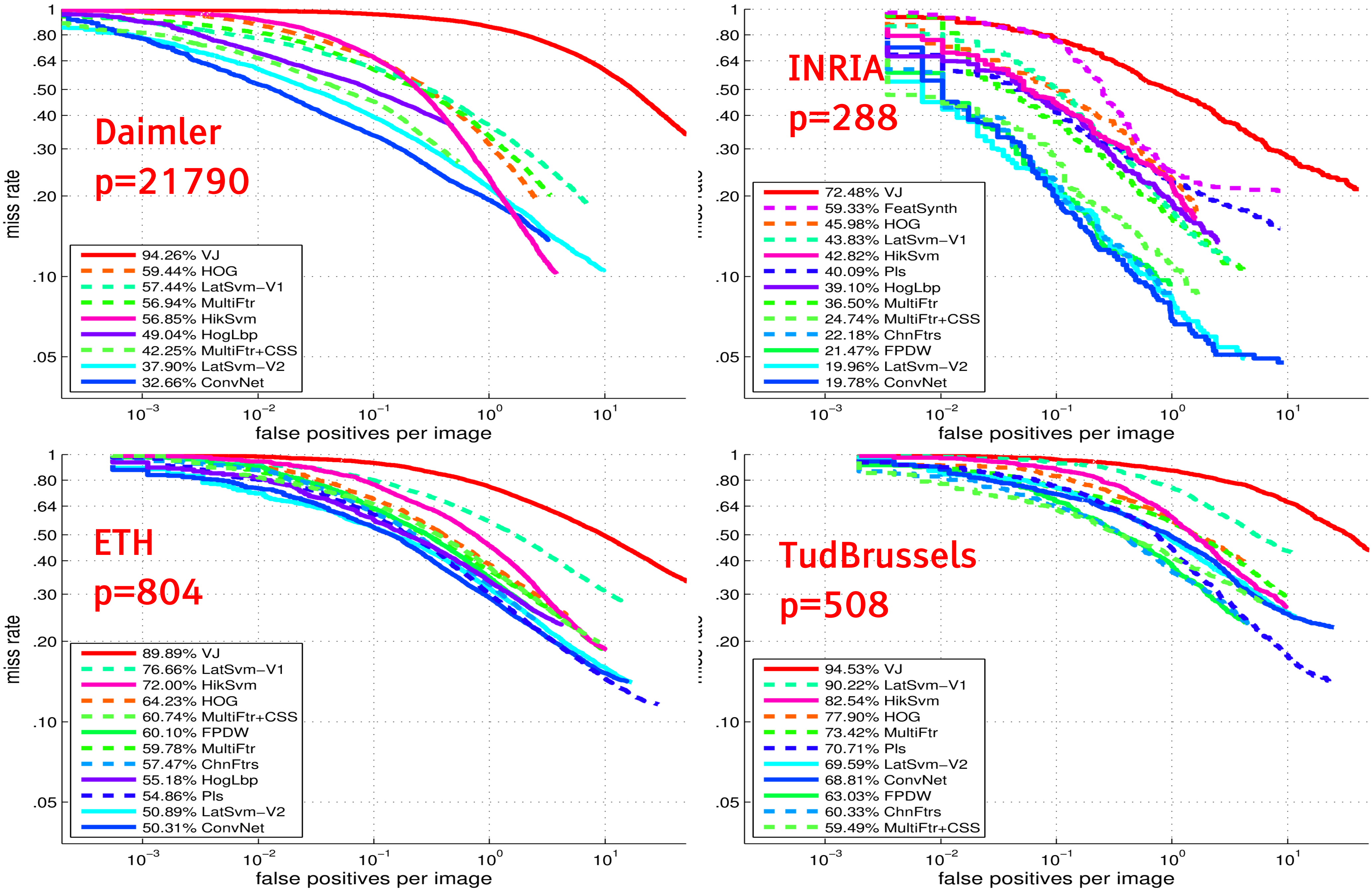


[Kavukcuoglu et al. NIPS 2010] [Sermanet et al. ArXiv 2012]

Results on "Near Scale" Images (>80 pixels tall, no occlusions)

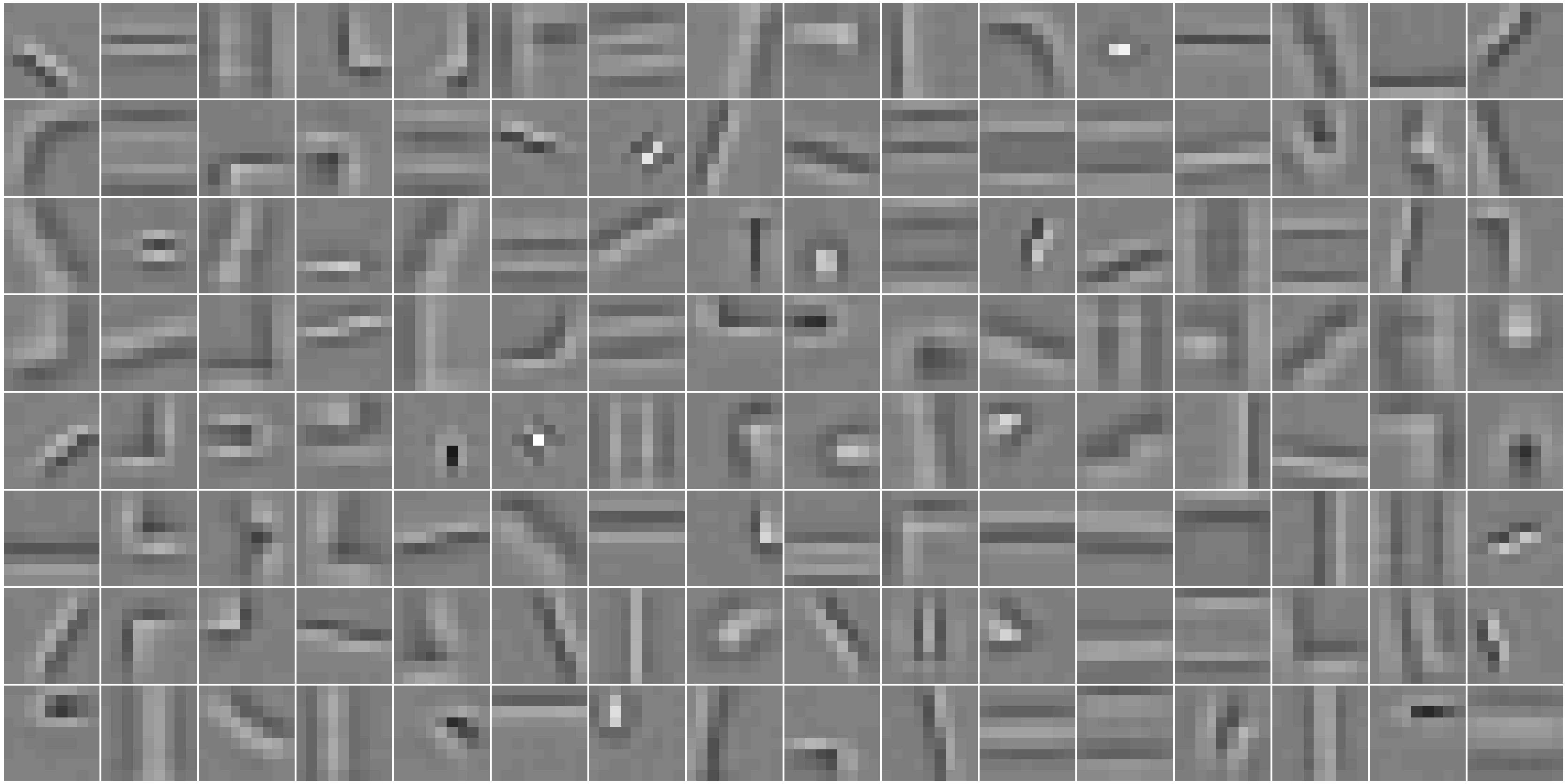


Results on "Reasonable" Images (>50 pixels tall, few occlusions)



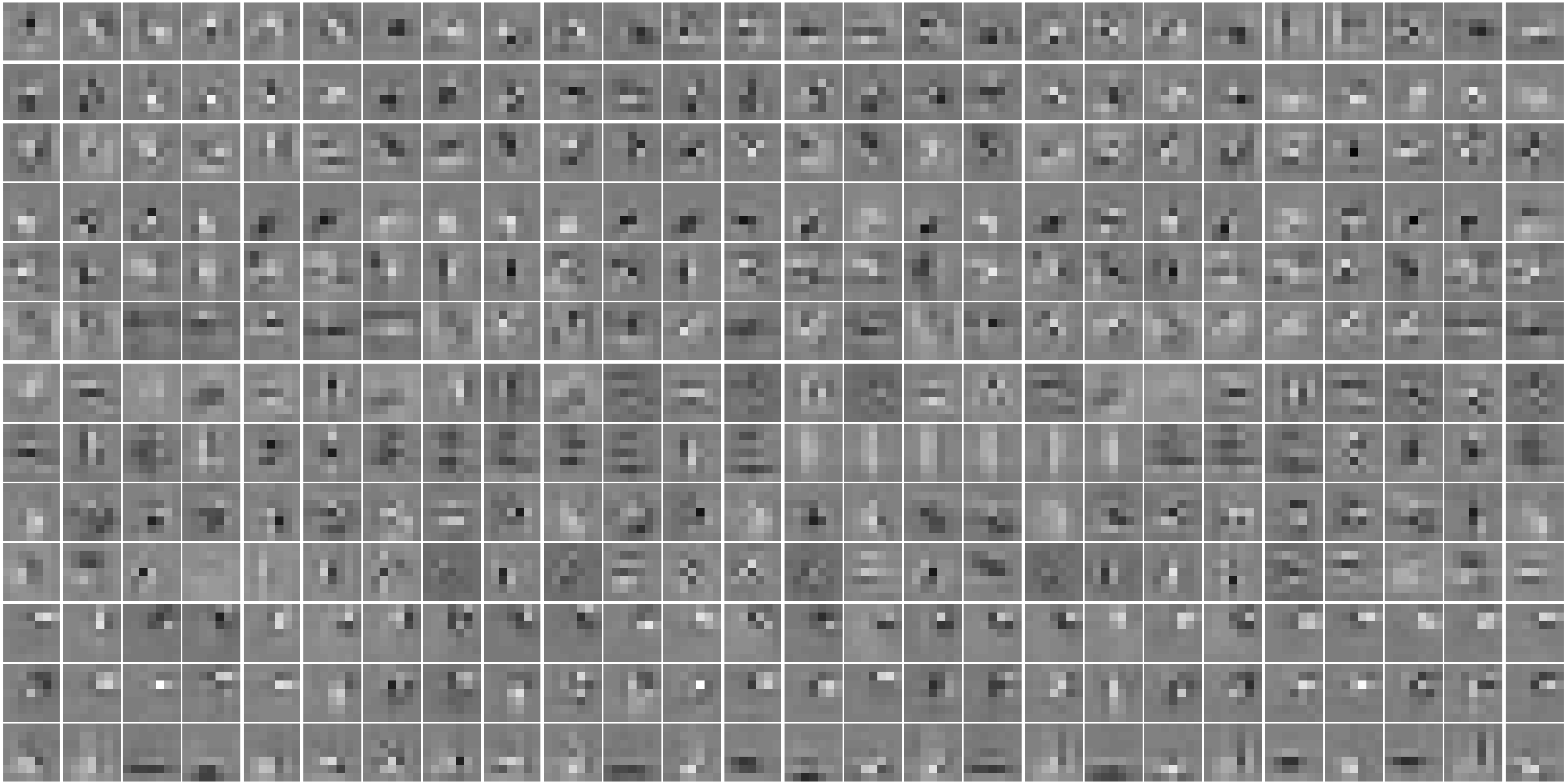
Unsupervised pre-training with convolutional PSD

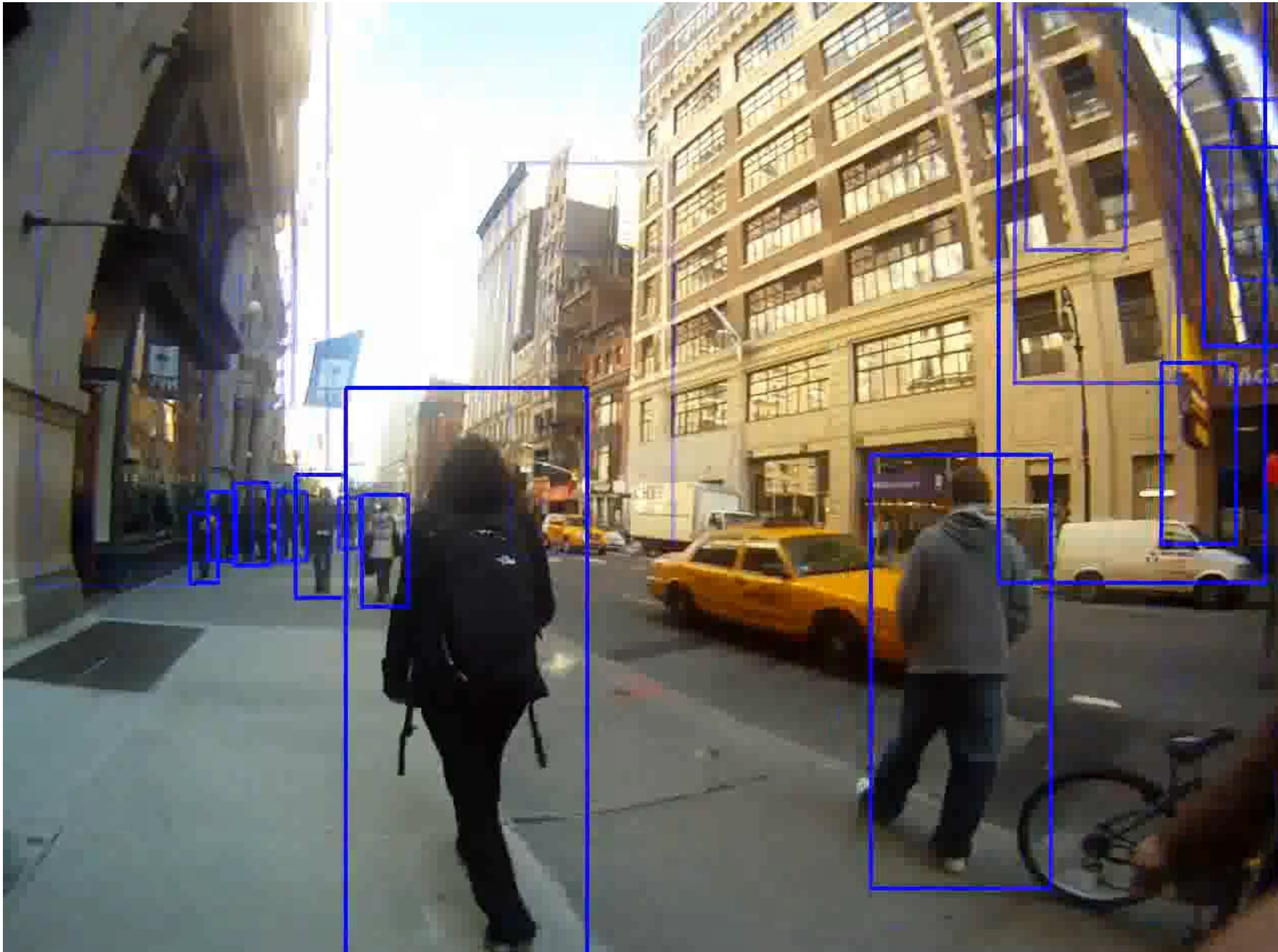
- 128 stage-1 filters on Y channel.
- Unsupervised training with convolutional predictive sparse decomposition



Unsupervised pre-training with convolutional PSD

- Stage 2 filters.
- Unsupervised training with convolutional predictive sparse decomposition

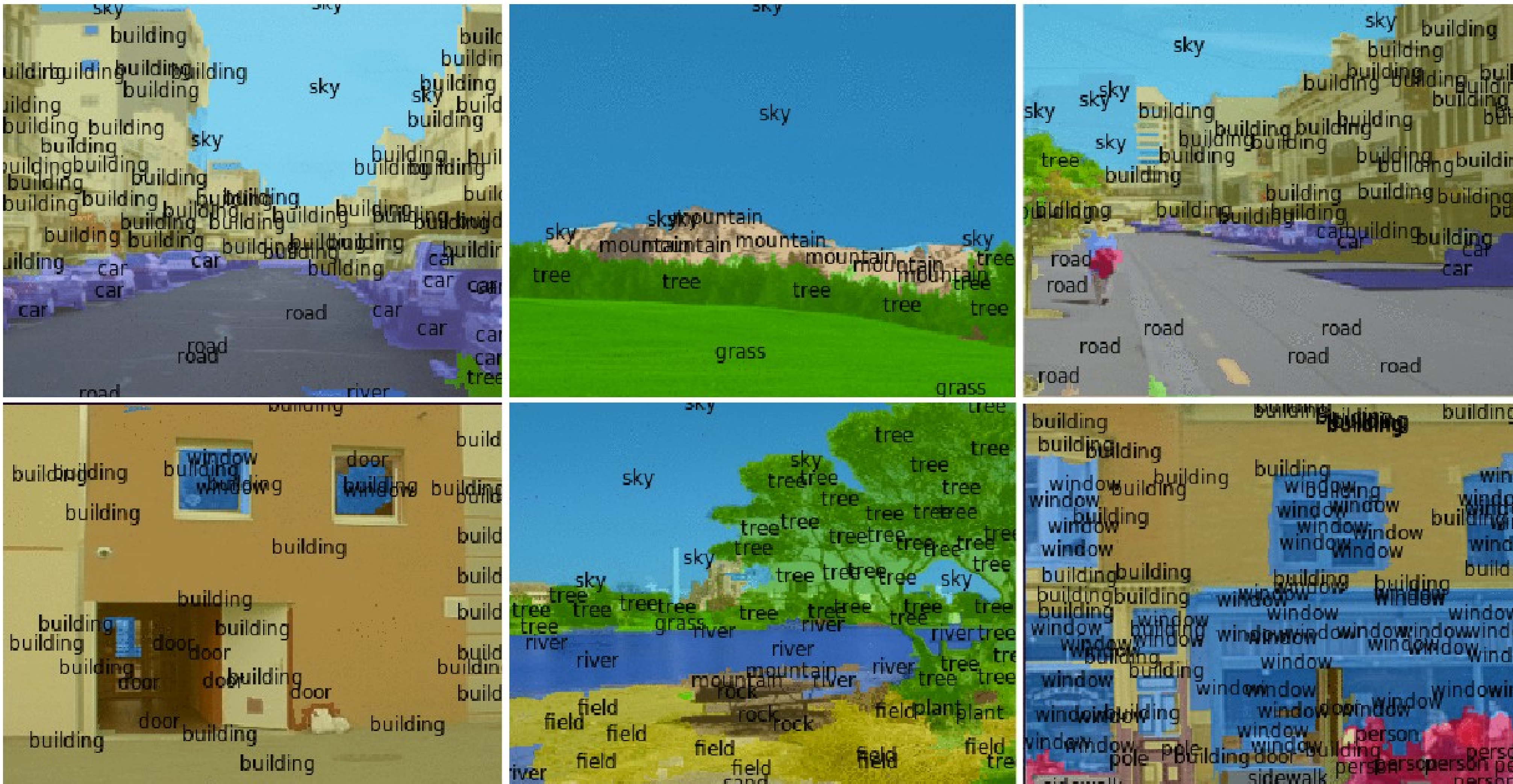




Convolutional Networks
In
Semantic Segmentation,
Scene Labeling

Semantic Labeling: Labeling every pixel with the object it belongs to

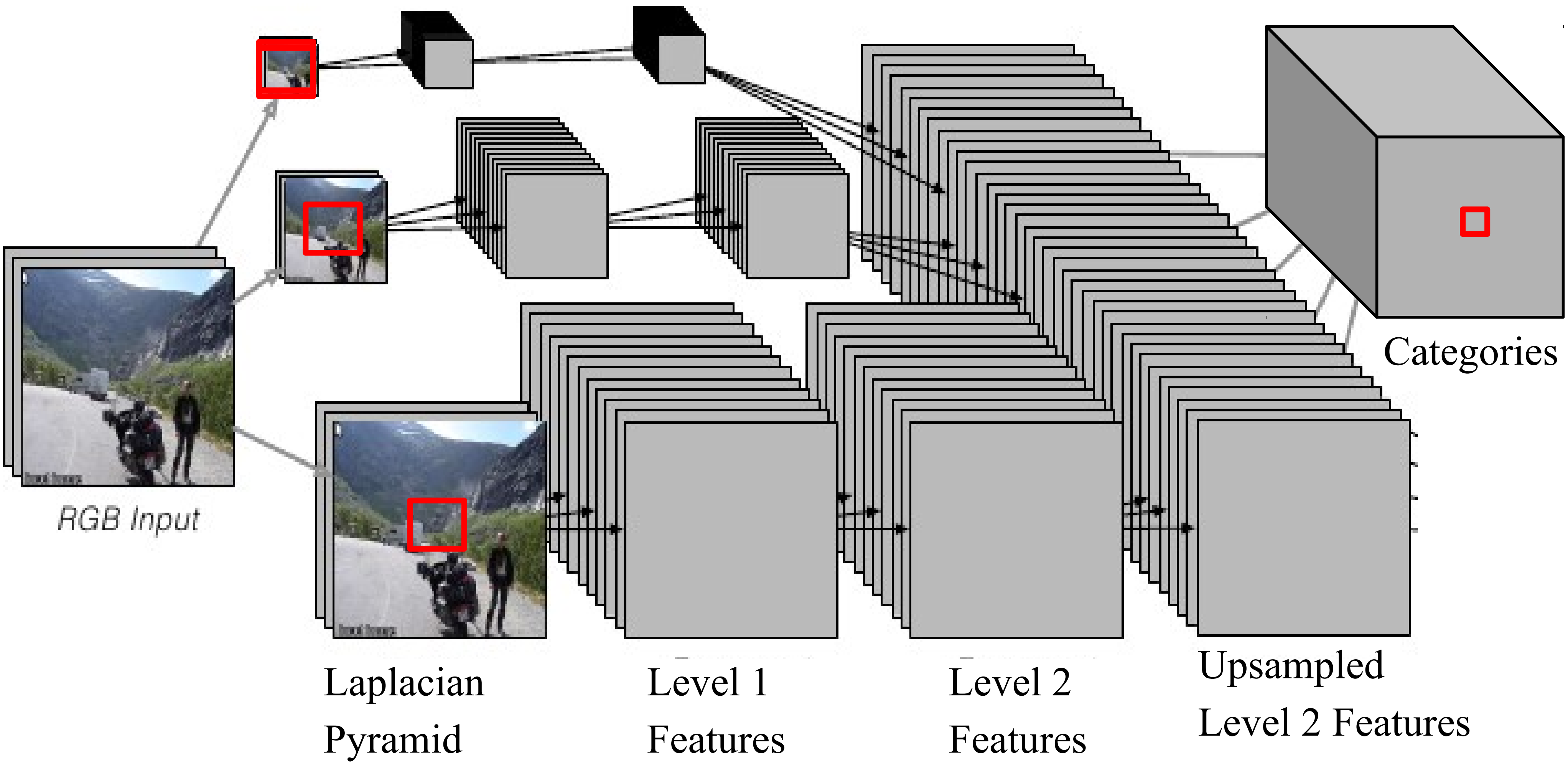
- Would help identify obstacles, targets, landing sites, dangerous areas
- Would help line up depth map with edge maps



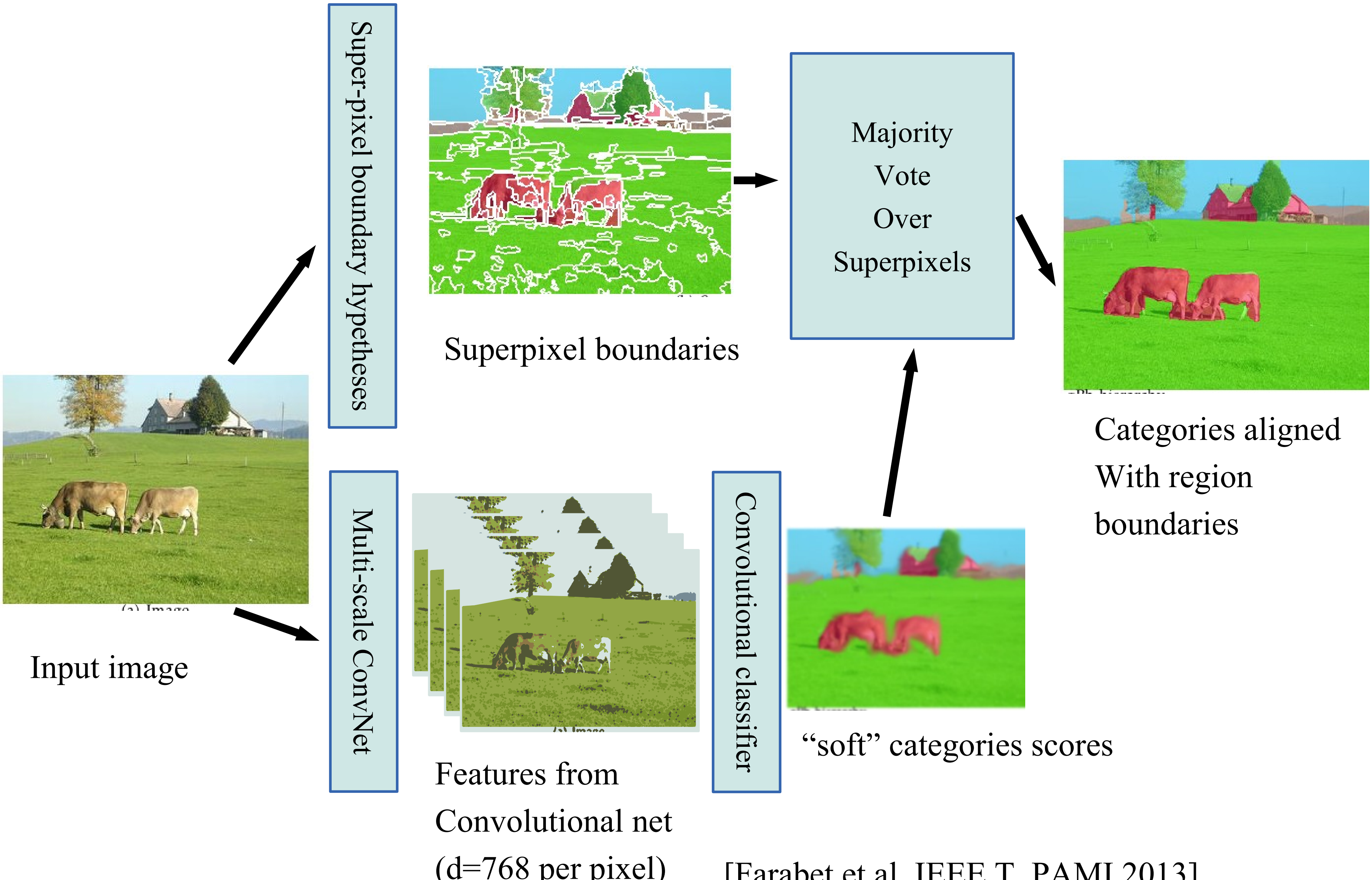
[Farabet et al. ICML 2012, PAMI 2013]

Scene Parsing/Labeling: ConvNet Architecture

- Each output sees a large input context:
  - 46x46 window at full rez; 92x92 at ½ rez; 184x184 at ¼ rez
  - [7x7conv]->[2x2pool]->[7x7conv]->[2x2pool]->[7x7conv]->
  - Trained supervised on fully-labeled images



Method 1: majority over super-pixel regions



[Farabet et al. IEEE T. PAMI 2013]

Scene Parsing/Labeling: Performance

- Stanford Background Dataset [Gould 1009]: 8 categories

	Pixel Acc.	Class Acc.	CT (sec.)
Gould *et al.* 2009 [14]	76.4%	-	10 to 600s
Munoz *et al.* 2010 [32]	76.9%	66.2%	12s
Tighe *et al.* 2010 [46]	77.5%	-	10 to 300s
Socher *et al.* 2011 [45]	78.1%	-	?
Kumar *et al.* 2010 [22]	79.4%	-	$< 600s$
Lempitzky *et al.* 2011 [28]	**81.9%**	72.4%	$> 60s$
singlescale convnet	66.0 %	56.5 %	0.35s
multiscale convnet	78.8 %	72.4%	0.6s
multiscale net + superpixels	80.4%	74.56%	**0.7s**
multiscale net + gPb + cover	80.4%	75.24%	61s
multiscale net + CRF on gPb	81.4%	**76.0%**	60.5s

[Farabet et al. IEEE T. PAMI 2013]

Scene Parsing/Labeling: Performance

	Pixel Acc.	Class Acc.
Liu *et al.* 2009 [31]	74.75%	-
Tighe *et al.* 2010 [44]	76.9%	29.4%
raw multiscale net[1]	67.9%	45.9%
multiscale net + superpixels[1]	71.9%	**50.8%**
multiscale net + cover[1]	72.3%	**50.8%**
multiscale net + cover[2]	**78.5%**	29.6%

- SIFT Flow Dataset
- [Liu 2009]:
- 33 categories

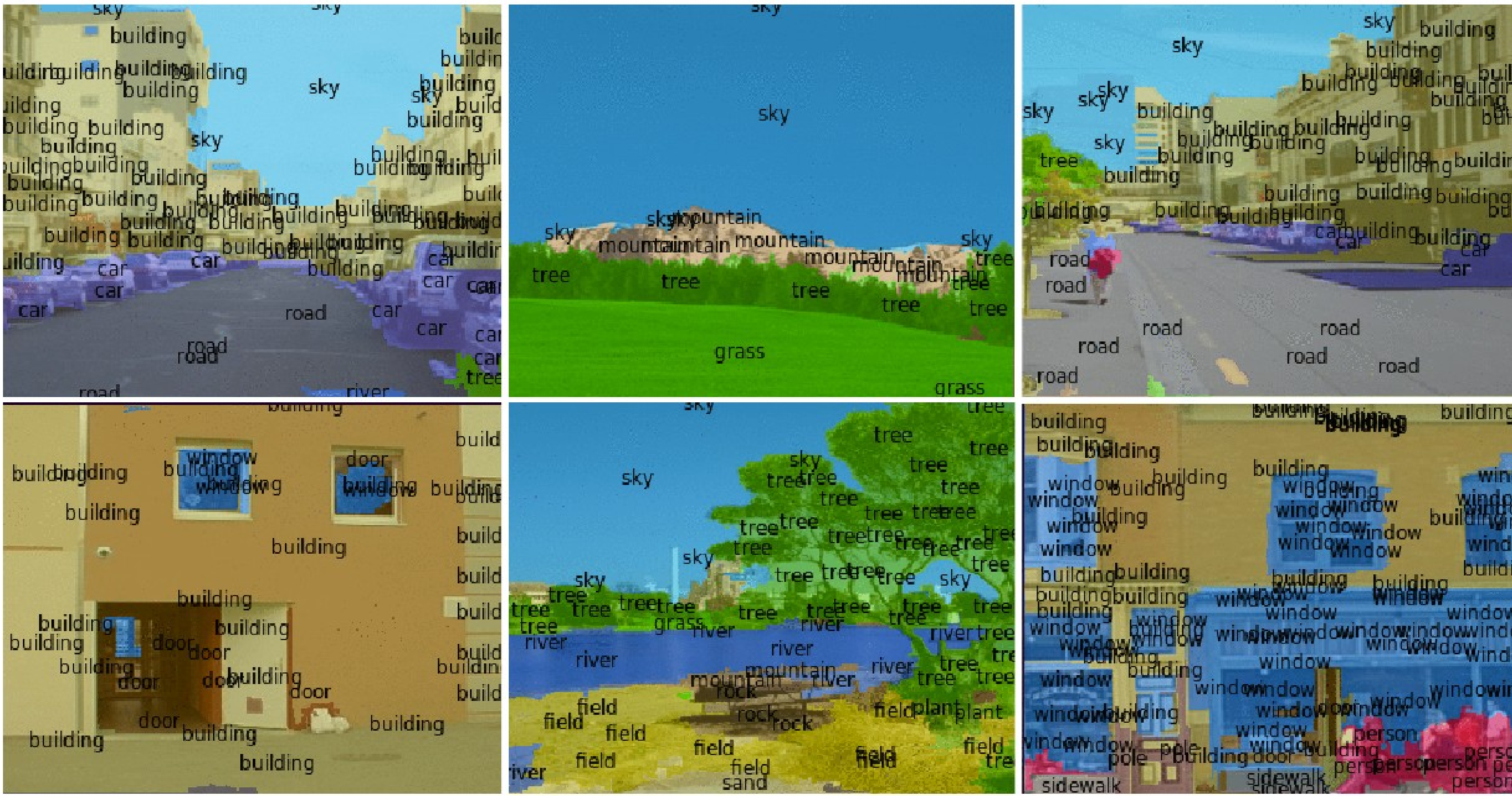
- Barcelona dataset
- [Tighe 2010]:
- 170 categories.

	Pixel Acc.	Class Acc.
Tighe *et al.* 2010 [44]	66.9%	7.6%
raw multiscale net[1]	37.8%	**12.1%**
multiscale net + superpixels[1]	44.1%	**12.4%**
multiscale net + cover[1]	46.4%	**12.5%**
multiscale net + cover[2]	**67.8%**	9.5%

[Farabet et al. IEEE T. PAMI 2012]

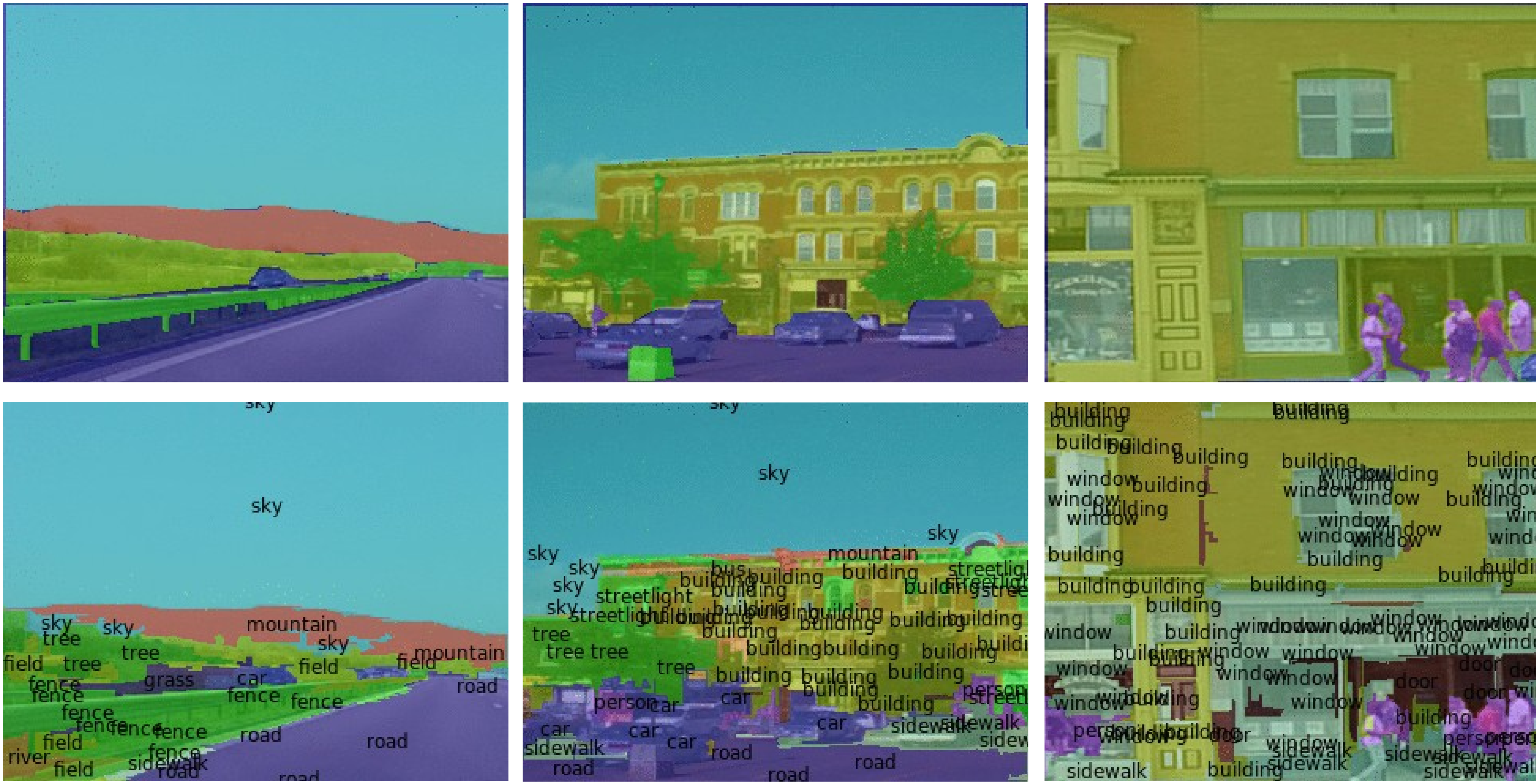
Scene Parsing/Labeling: SIFT Flow dataset (33 categories)

- Samples from the SIFT-Flow dataset (Liu)



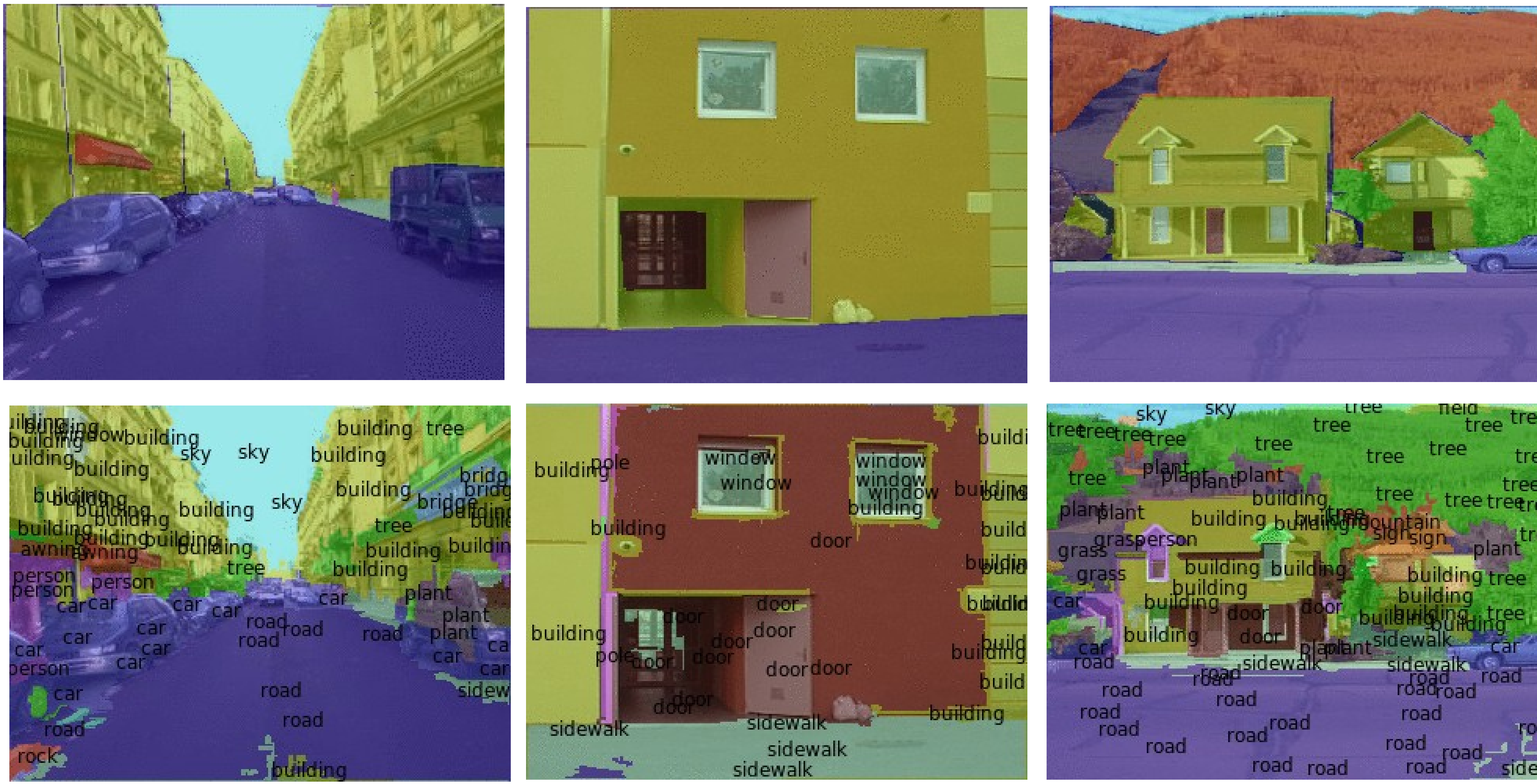
[Farabet et al. ICML 2012, PAMI 2013]

Scene Parsing/Labeling: SIFT Flow dataset (33 categories)



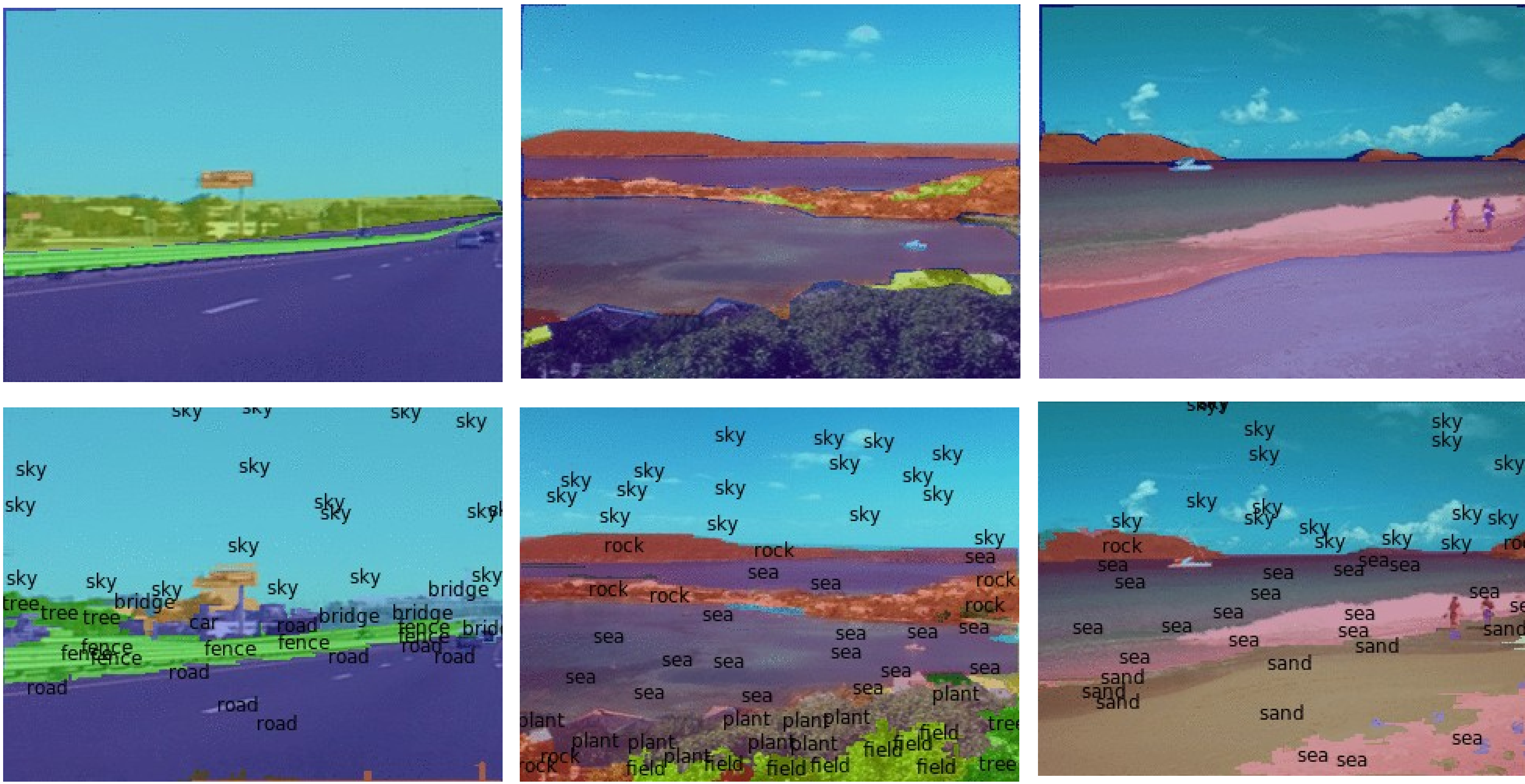
[Farabet et al. ICML 2012, PAMI 2013]

Scene Parsing/Labeling



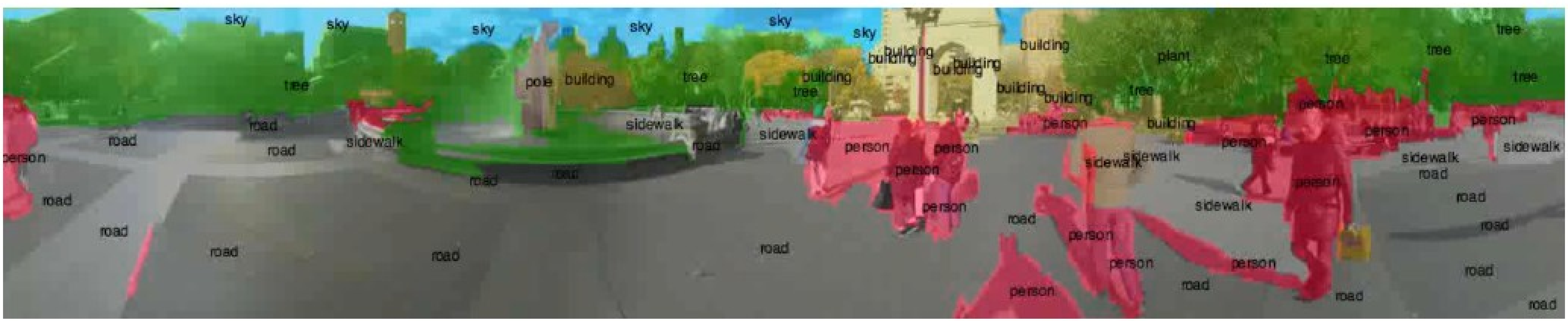
[Farabet et al. ICML 2012, PAMI 2013]

Scene Parsing/Labeling



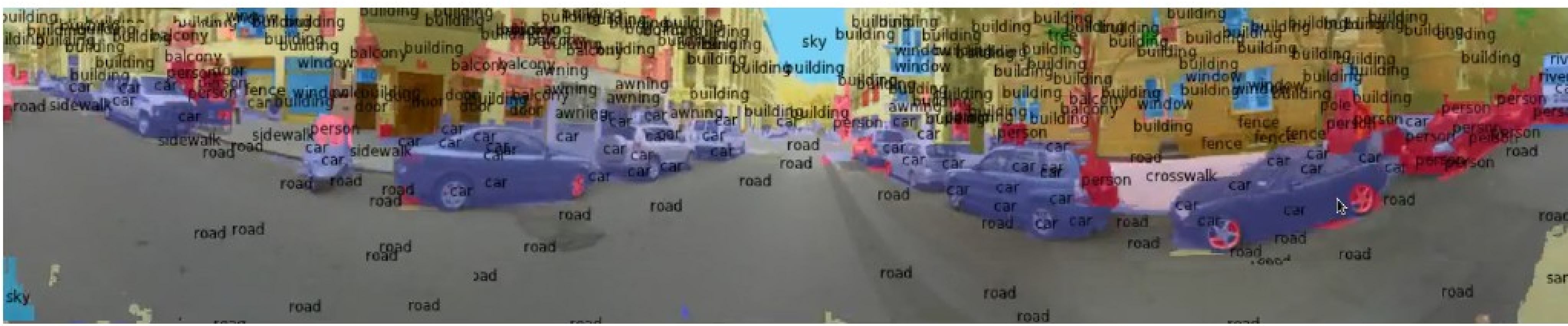
[Farabet et al. ICML 2012, PAMI 2013]

Scene Parsing/Labeling



[Farabet et al. ICML 2012, PAMI 2013]

Scene Parsing/Labeling



[Farabet et al. ICML 2012, PAMI 2013]

Scene Parsing/Labeling

- No post-processing
- Frame-by-frame
- ConvNet runs at 50ms/frame on Virtex-6 FPGA hardware
  - But communicating the features over ethernet limits system performance

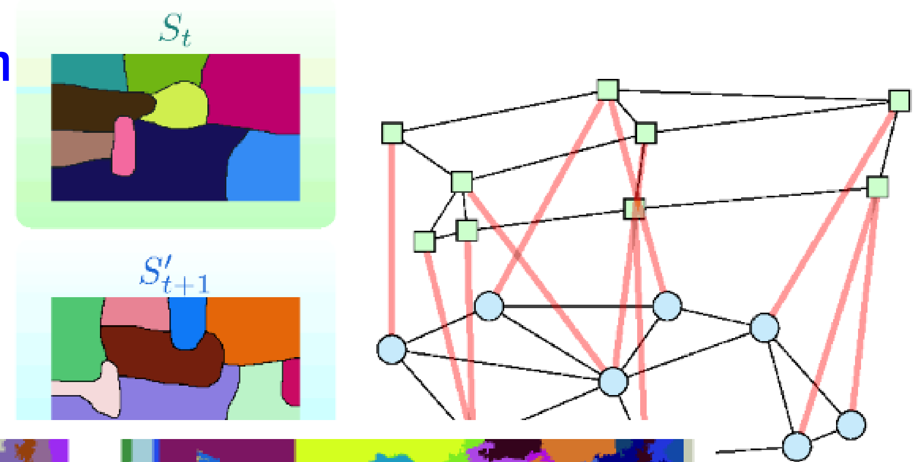
Scene Parsing/Labeling: Temporal Consistency

- Causal method for temporal consistency

[Couprie, Farabet, Najman, LeCun ICLR 2013, ICIP 2013]

Temporal Consistency

- Spatio-Temporal Super-Pixel segmentation
  - [Couprie et al ICIP 2013]
  - [Couprie et al JMLR under review]
  - Majority vote over super-pixels



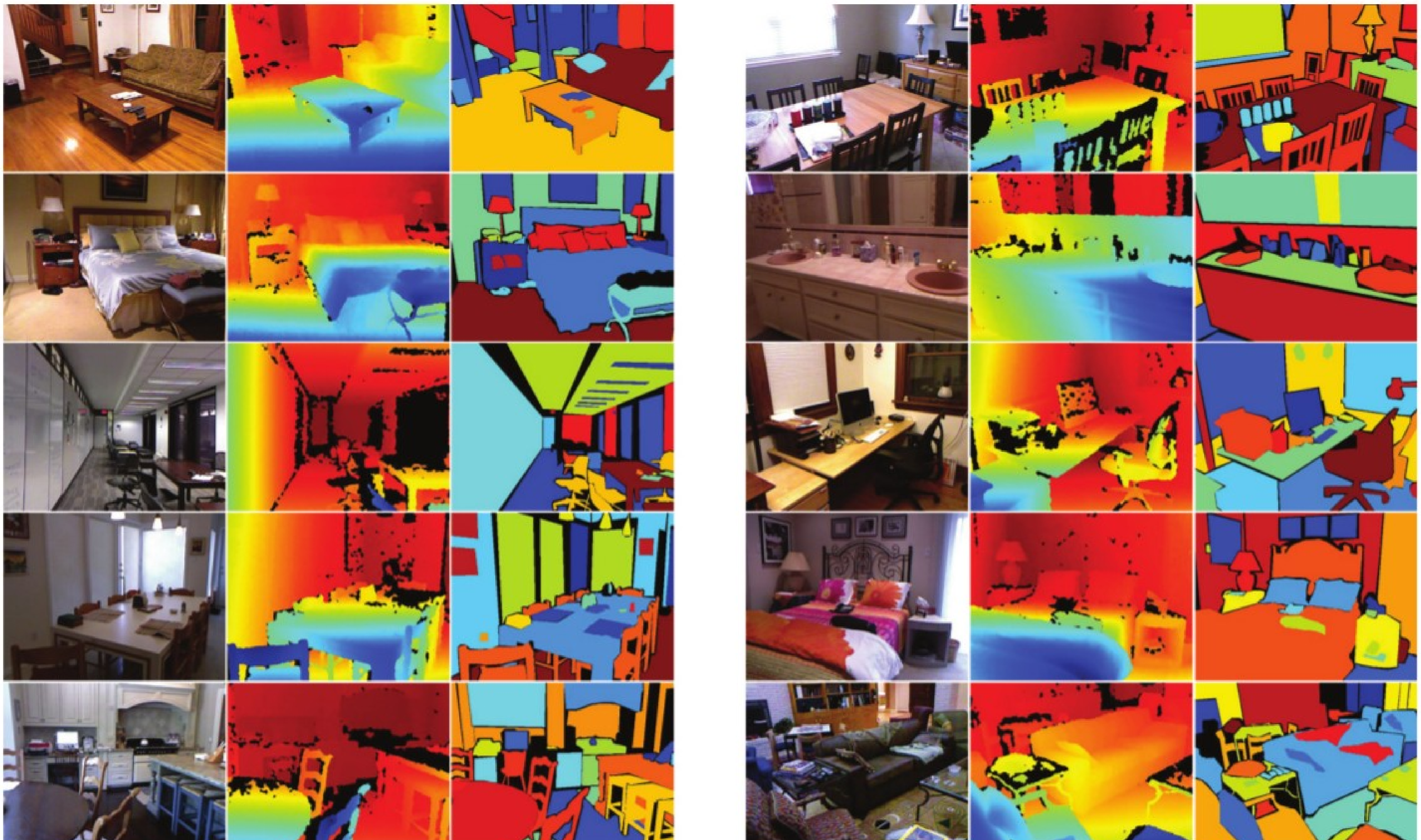
Independent segmentations S'_1, S'_2 and S'_3

Temporally consistent segmentations $S_1 (= S'_1)$, S_2, and S_3

NYU RGB-Depth Indoor Scenes Dataset

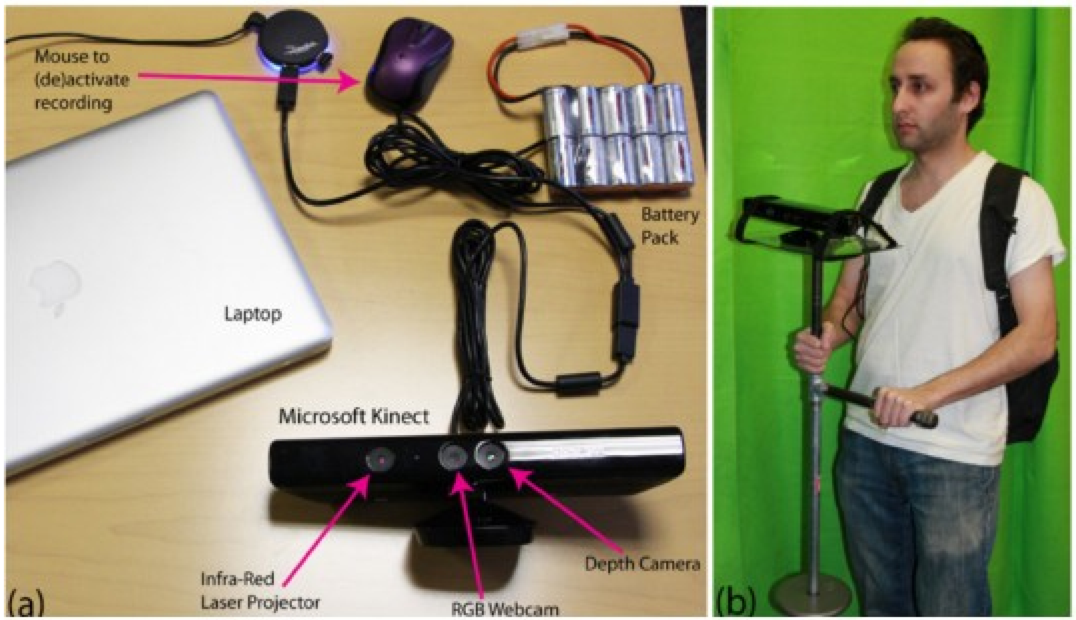
- 407024 RGB-D images of apartments
- 1449 labeled frames, 894 object categories

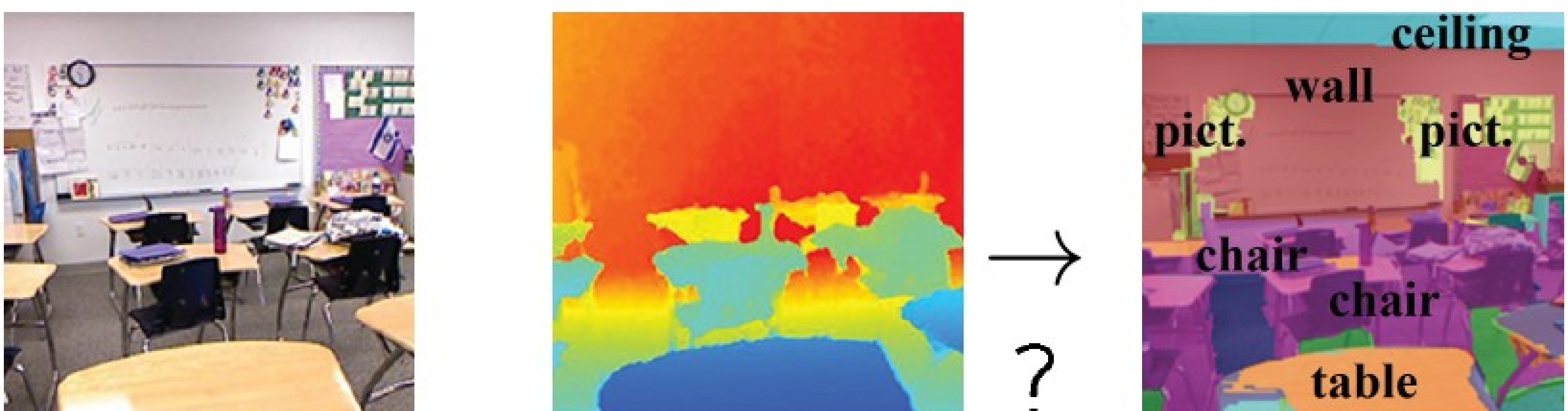
[Silberman et al. 2012]



NYU RGB-D Dataset

- Captured with a Kinect on a steadycam





Results

	Class Occurrences	Multiscale Convnet Acc. Farabet et al. (2013)	MultiScl. Cnet +depth Acc.
bed	4.4%	30.3	**38.1**
objects	7.1 %	**10.9**	8.7
chair	3.4%	**44.4**	34.1
furnit.	12.3%	28.5	**42.4**
ceiling	1.4%	33.2	**62.6**
floor	9.9%	68.0	**87.3**
deco.	3.4%	38.5	**40.4**
sofa	3.2%	**25.8**	24.6
table	3.7%	**18.0**	10.2
wall	24.5%	**89.4**	86.1
window	5.1%	**37.8**	15.9
books	2.9%	**31.7**	13.7
TV	1.0%	**18.8**	6.0
unkn.	17.8%	-	-
Avg. Class Acc.	-	35.8	**36.2**
Pixel Accuracy (mean)	-	51.0	**52.4**
Pixel Accuracy (median)	-	51.7	**52.9**
Pixel Accuracy (std. dev.)	-	15.2	15.2

Results

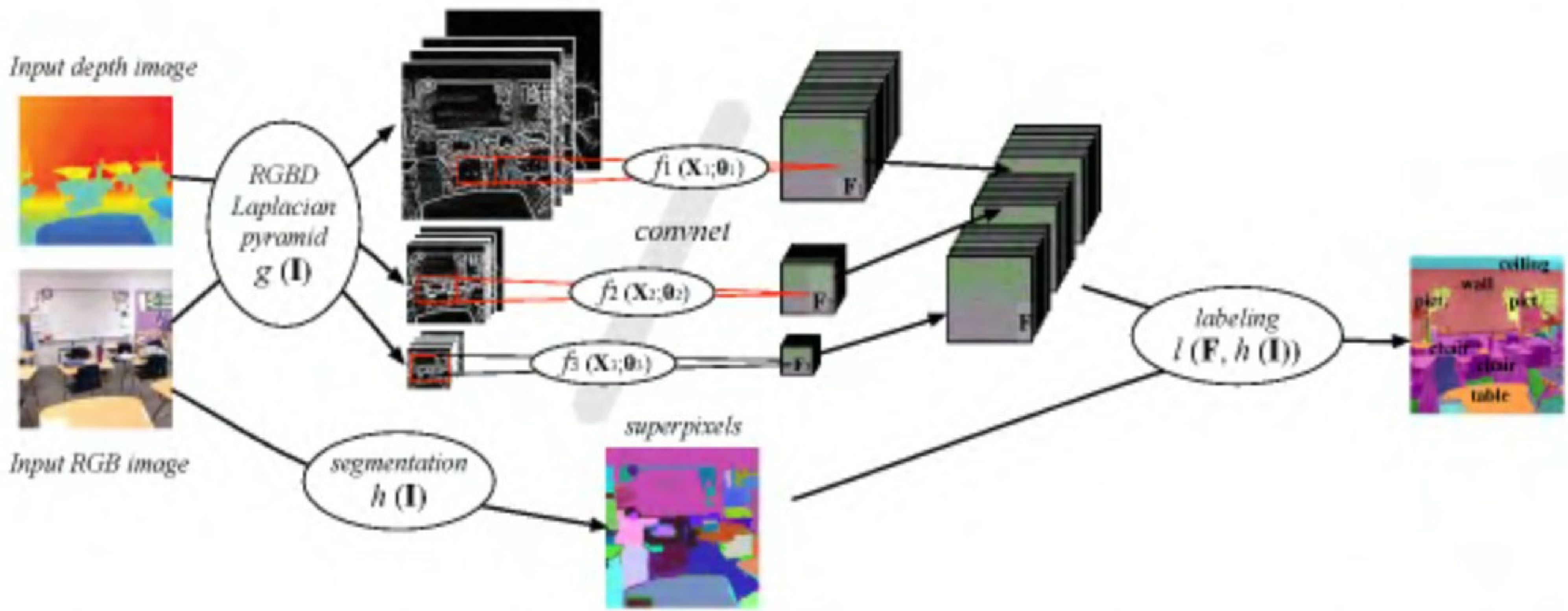
- **Depth helps a bit**
  - Helps a lot for floor and props
  - Helps surprisingly little for structures, and hurts for furniture

	Ground	Furniture	Props	Structure	Class Acc.	Pixel Acc.	Comput. time (s)
Silberman et al. (2012)	68	**70**	**42**	59	59.6	58.6	>3
Cadena and Kosecka (2013)	**87.9**	**64.1**	31.0	77.8	**65.2**	**66.9**	1.7
Multiscale convnet	68.1	51.1	29.9	**87.8**	59.2	63.0	**0.7**
Multiscale+depth convnet	**87.3**	45.3	**35.5**	**86.1**	**63.5**	**64.5**	**0.7**

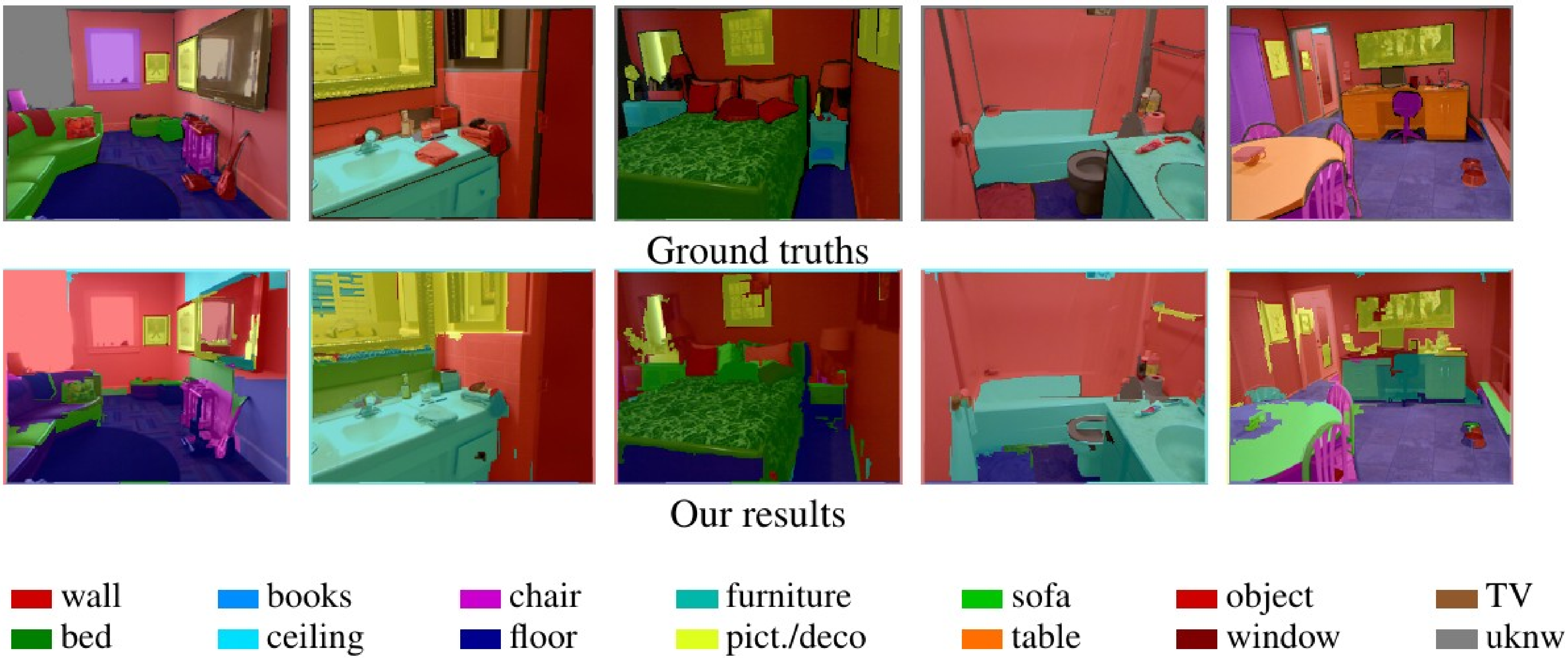
[C. Cadena, J. Kosecka “Semantic Parsing for Priming Object Detection in RGB-D Scenes” Semantic Perception Mapping and Exploration (SPME), Karlsruhe 2013]

Architecture for indoor RGB-D Semantic Segmentation

- Similar to outdoors semantic segmentation method
  - Convnet with 4 input channels
  - Vote over superpixels

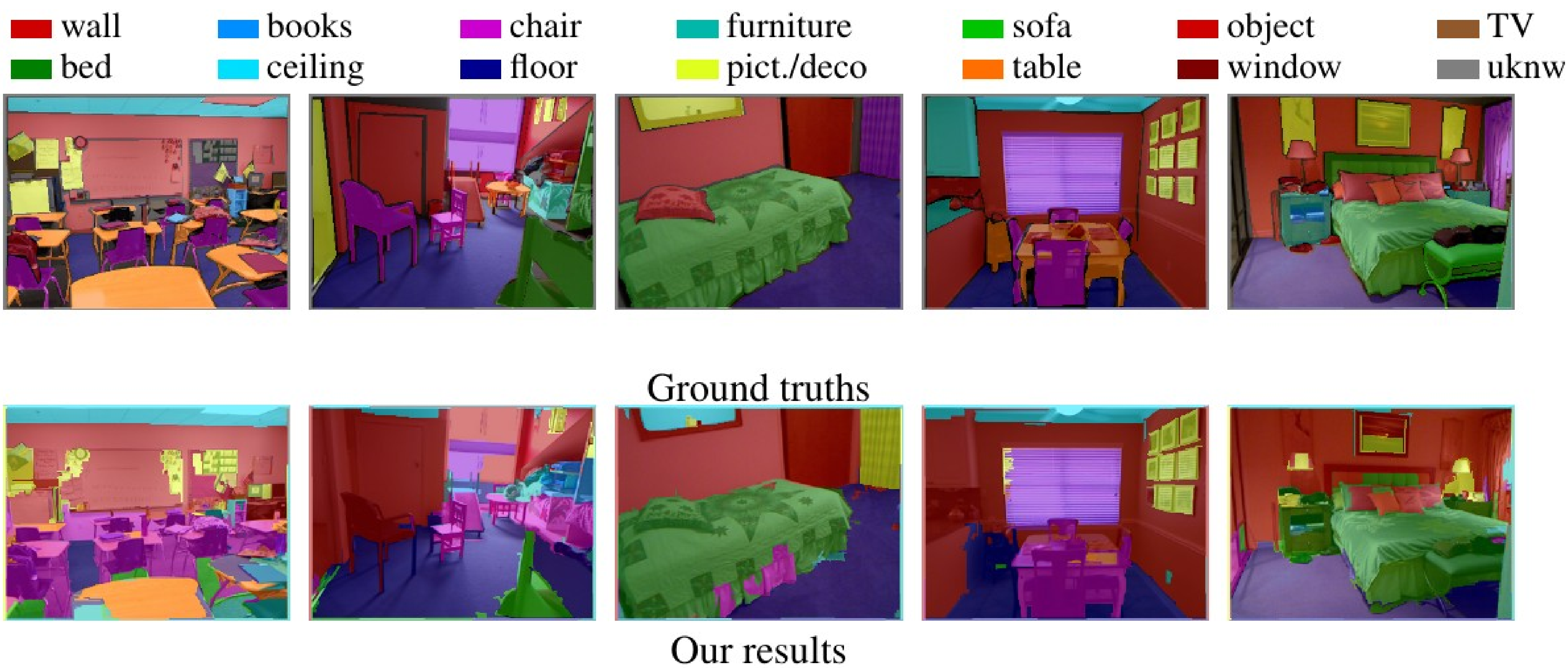


Scene Parsing/Labeling on RGB+Depth Images



[Couprie, Farabet, Najman, LeCun ICLR 2013, ICIP 2013]

Scene Parsing/Labeling on RGB+Depth Images



[Couprie, Farabet, Najman, LeCun ICLR 2013, ICIP 2013]

Labeling Videos

- Temporal consistency

(a) Output of the Multiscale convnet trained using depth information - frame by frame

(b) Results smoothed temporally using Couprie et al. (2013a)

[Couprie, Farabet, Najman, LeCun ICLR 2013]
[Couprie, Farabet, Najman, LeCun ICIP 2013]
[Couprie, Farabet, Najman, LeCun submitted to JMLR]

Semantic Segmentation on RGB+D Images and Videos

[Couprie, Farabet, Najman, LeCun ICLR 2013, ICIP 2013]

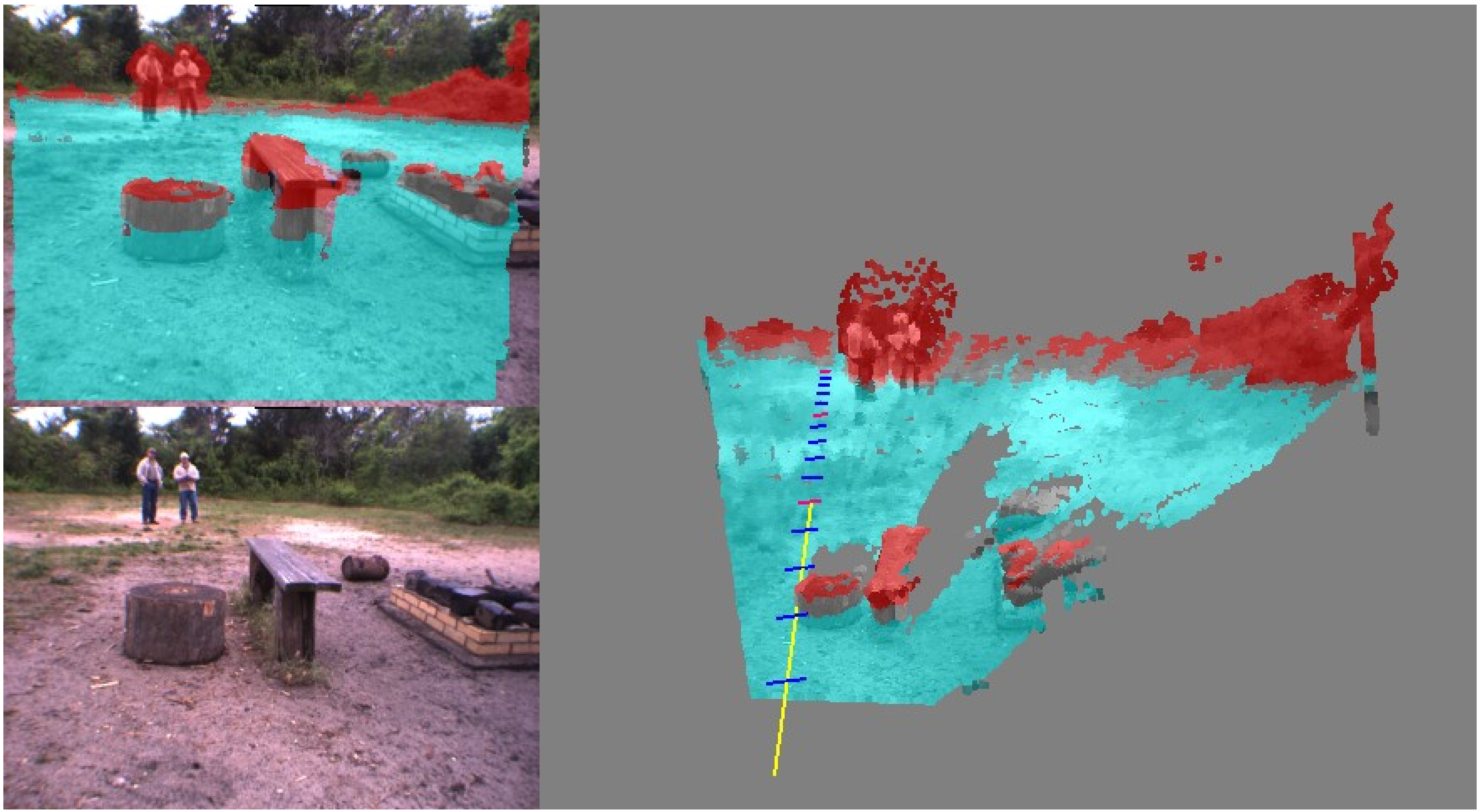
ConvNet Scene Labeling For Vision-Based Navigation Of Off-Road Robots

LAGR project: vision-based navigation for off-road robot

- Getting a robot to drive autonomously in unknown terrain solely from vision (camera input).
- Our team (NYU/Net-Scale Technologies Inc.) was one of 8 participants funded by DARPA
- All teams received identical robots and can only modify the software (not the hardware)
- The robot is given the GPS coordinates of a goal, and must drive to the goal as fast as possible. The terrain is unknown in advance. The robot is run 3 times through the same course.
- Long-Range Obstacle Detection with on-line, self-trained ConvNet
- Uses temporal consistency!

Obstacle Detection at Short Range: Stereovision

Obstacles overlaid with camera image

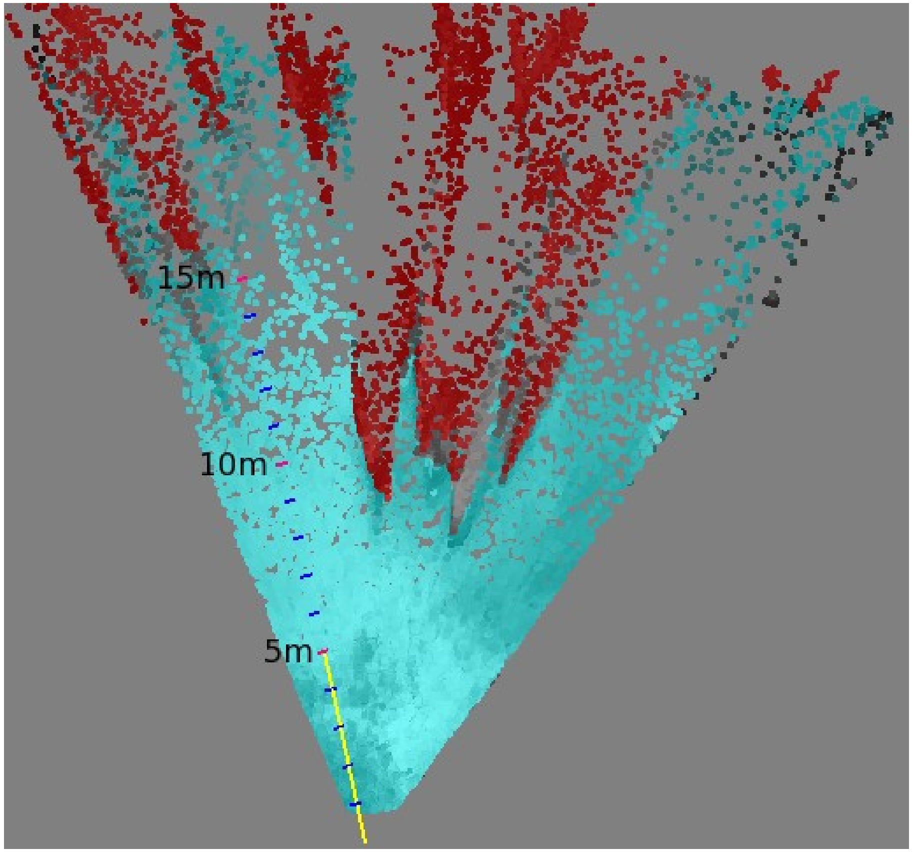


Camera image

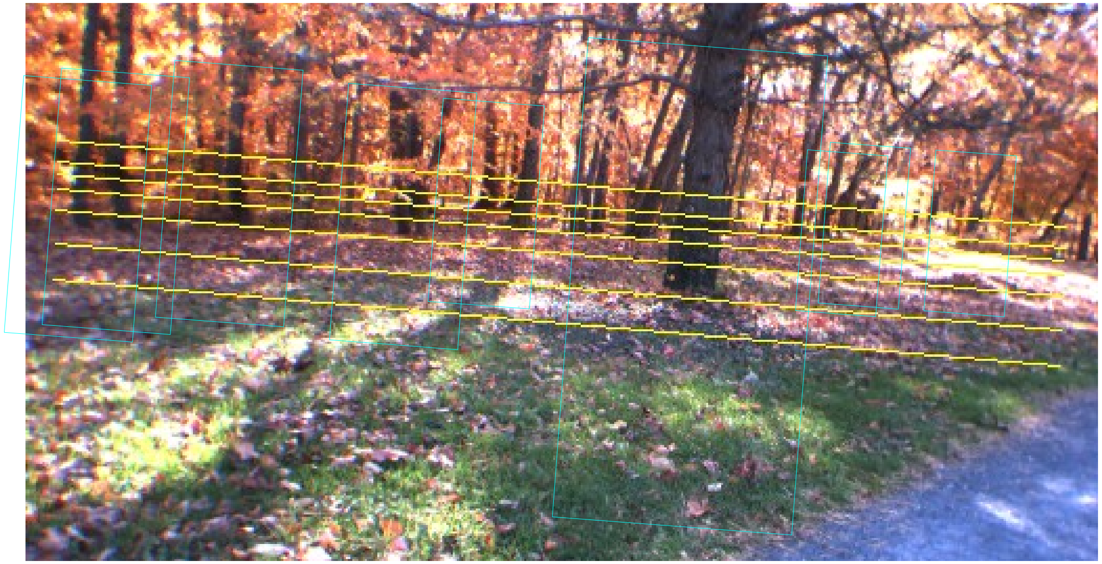
Detected obstacles (red)

But Stereovision Doesn't work at long range

- Stereo is only good up to about 10 meters.
- But not seeing past 10 meters is like driving in a fog or a snowstorm!

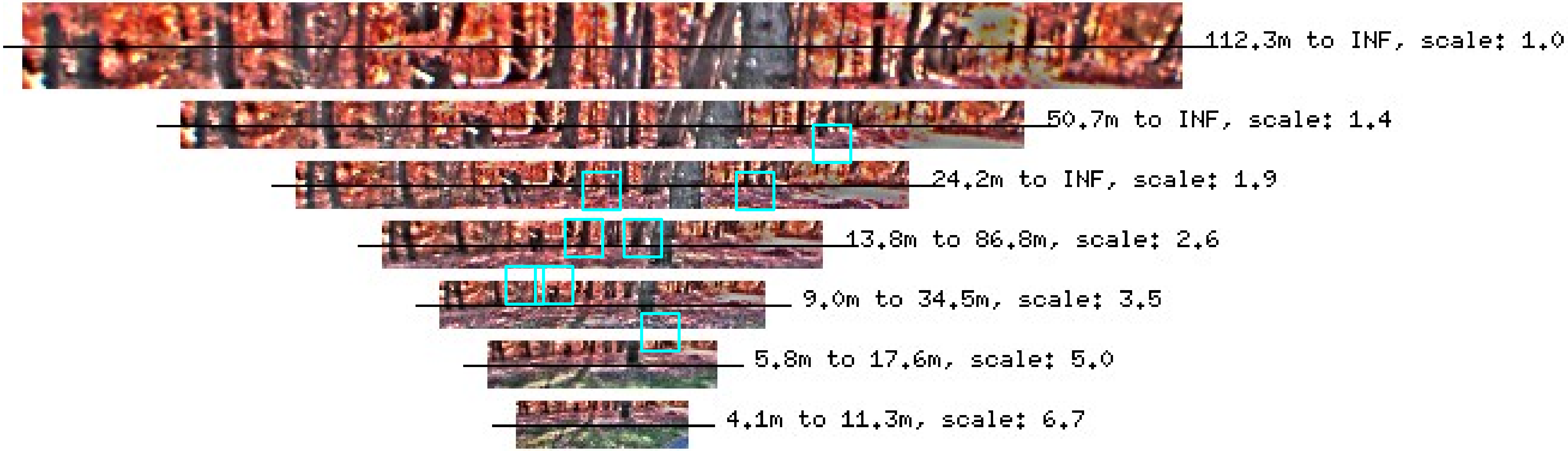


Long Range Vision with a Convolutional Net



Pre-processing (125 ms)

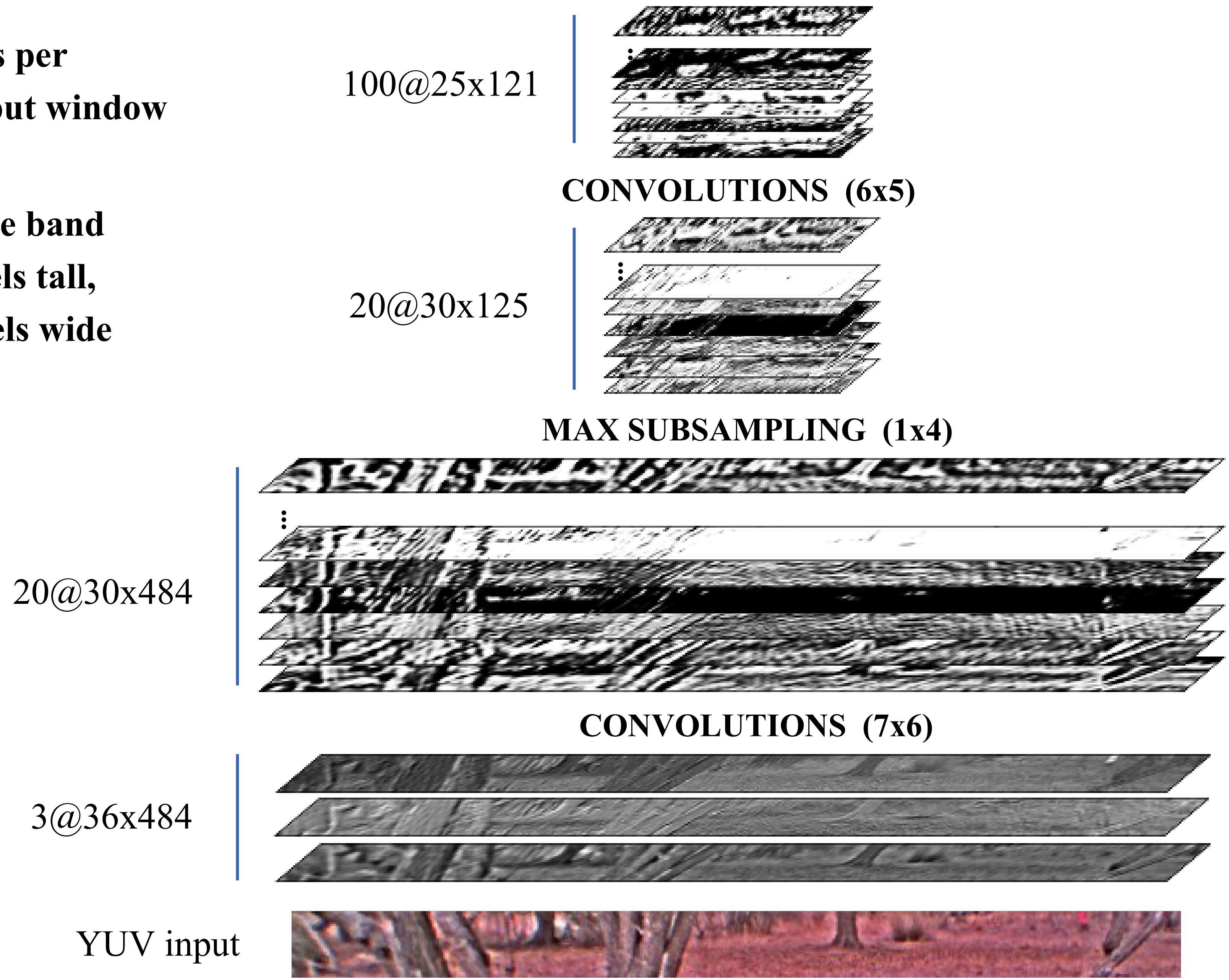
- Ground plane estimation
- Horizon leveling
- Conversion to YUV + local contrast normalization
- Scale invariant pyramid of distance-normalized image "bands"



Convolutional Net Architecture

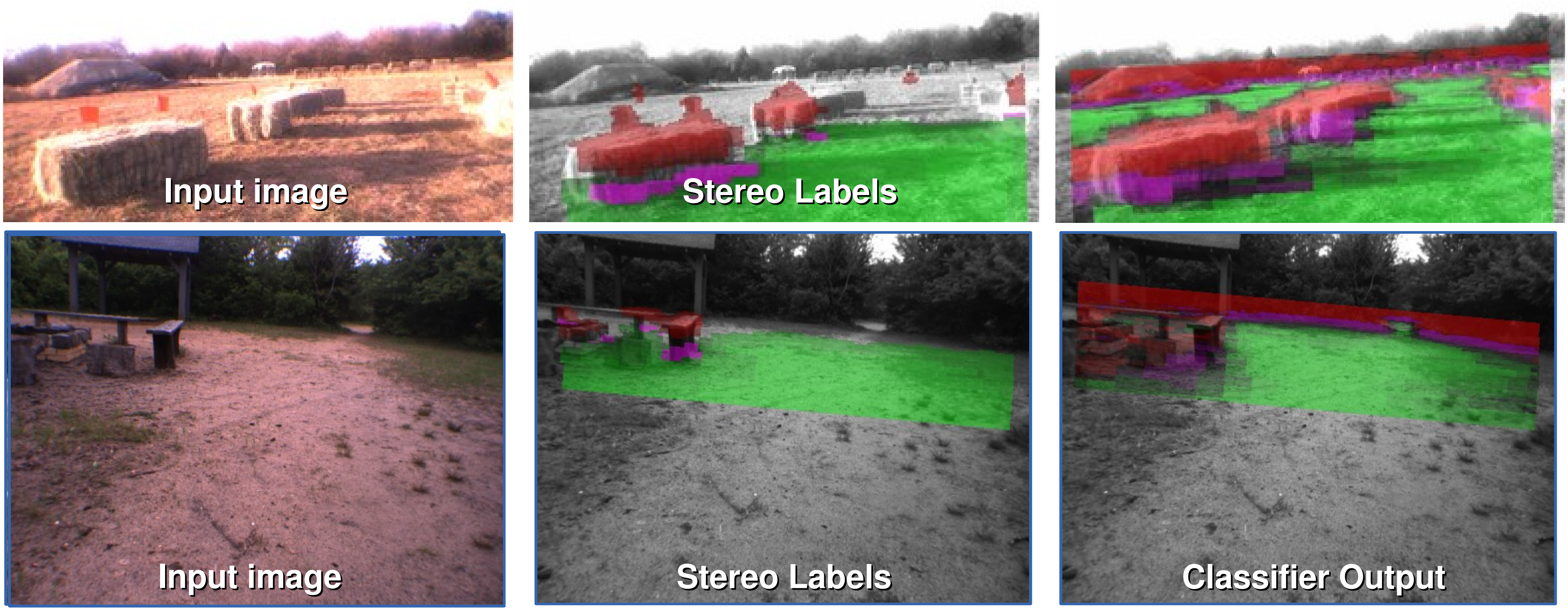
100 features per
3x12x25 input window

YUV image band
20-36 pixels tall,
36-500 pixels wide

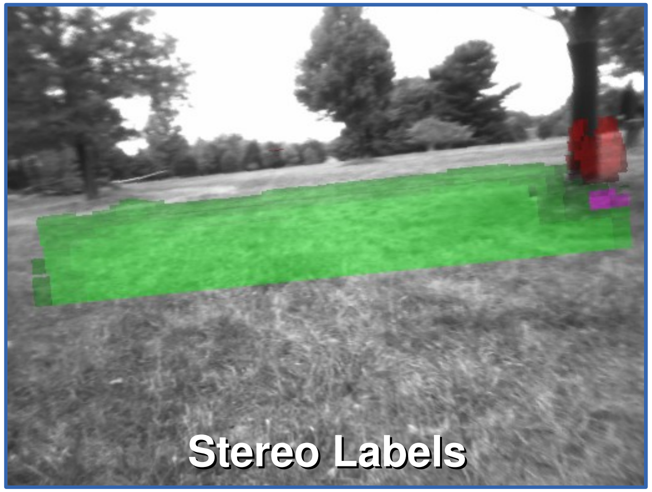


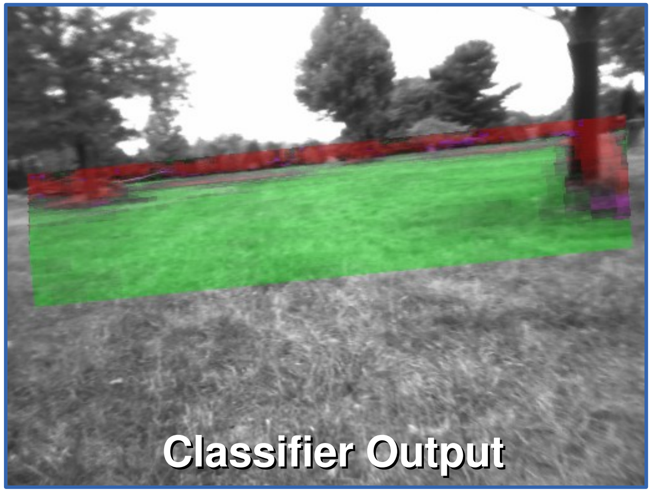
Scene Labeling with ConvNet + online learning

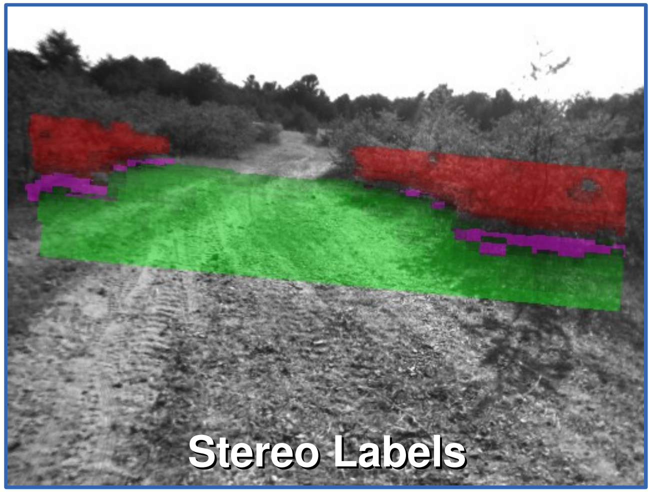
- **Image Labeling for Off-Road Robots [Hadsell JFR 2008]**
  - ConvNet labels pixels as one of 3 categories
  - Traversible/flat (green), non traversible (red), foot of obstacle (purple)
  - Labels obtained from stereo vision and SLAM

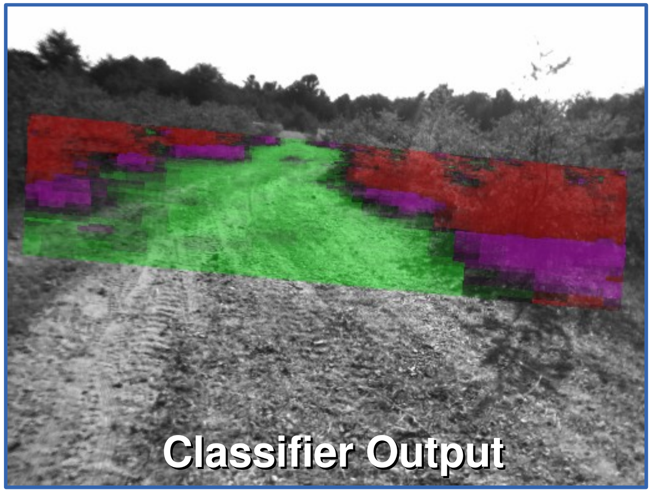


Long Range Vision Results

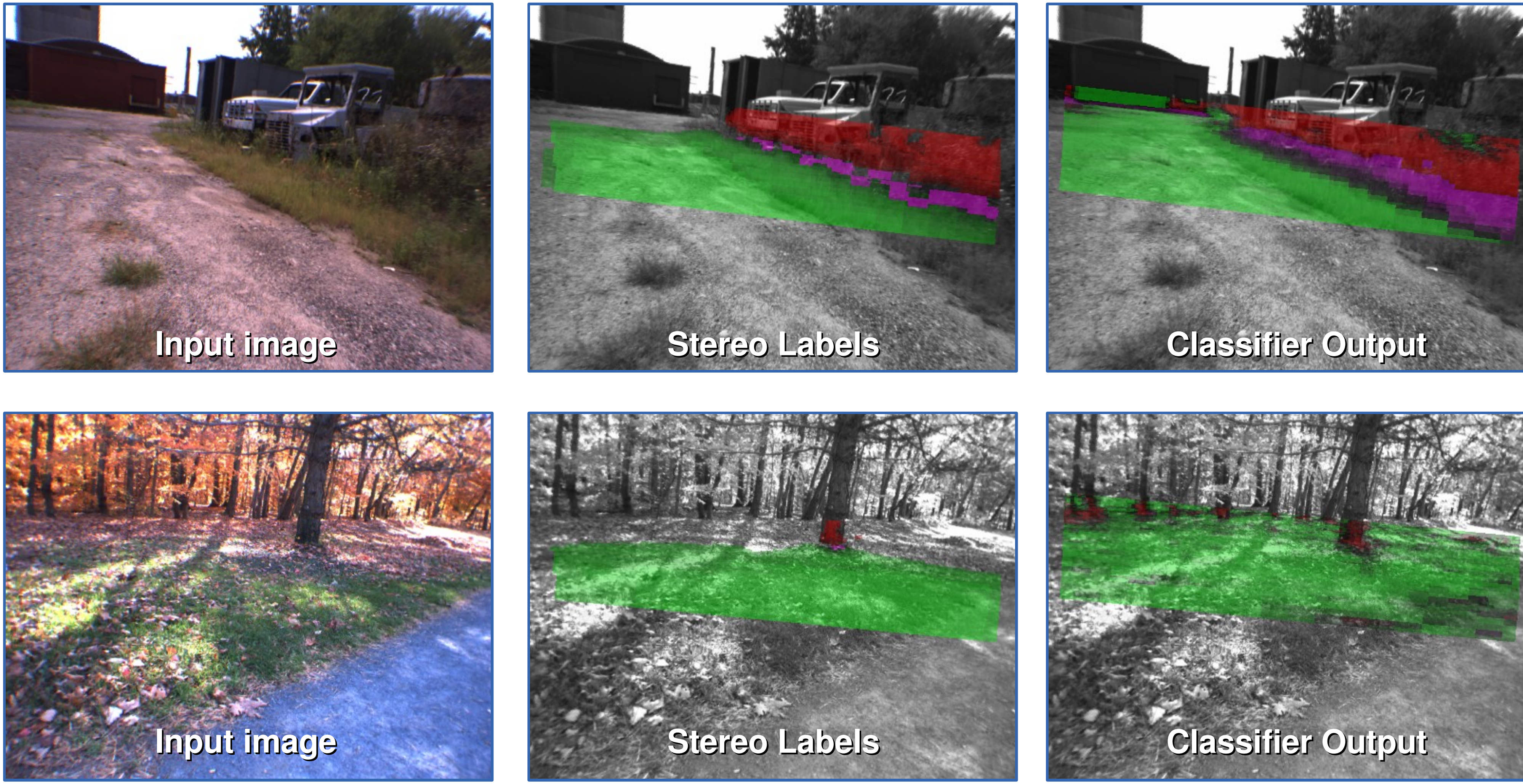








Long Range Vision Results



Hardware Acceleration for Convolutional Networks

Specialized Hardware can Accelerate Convolutions

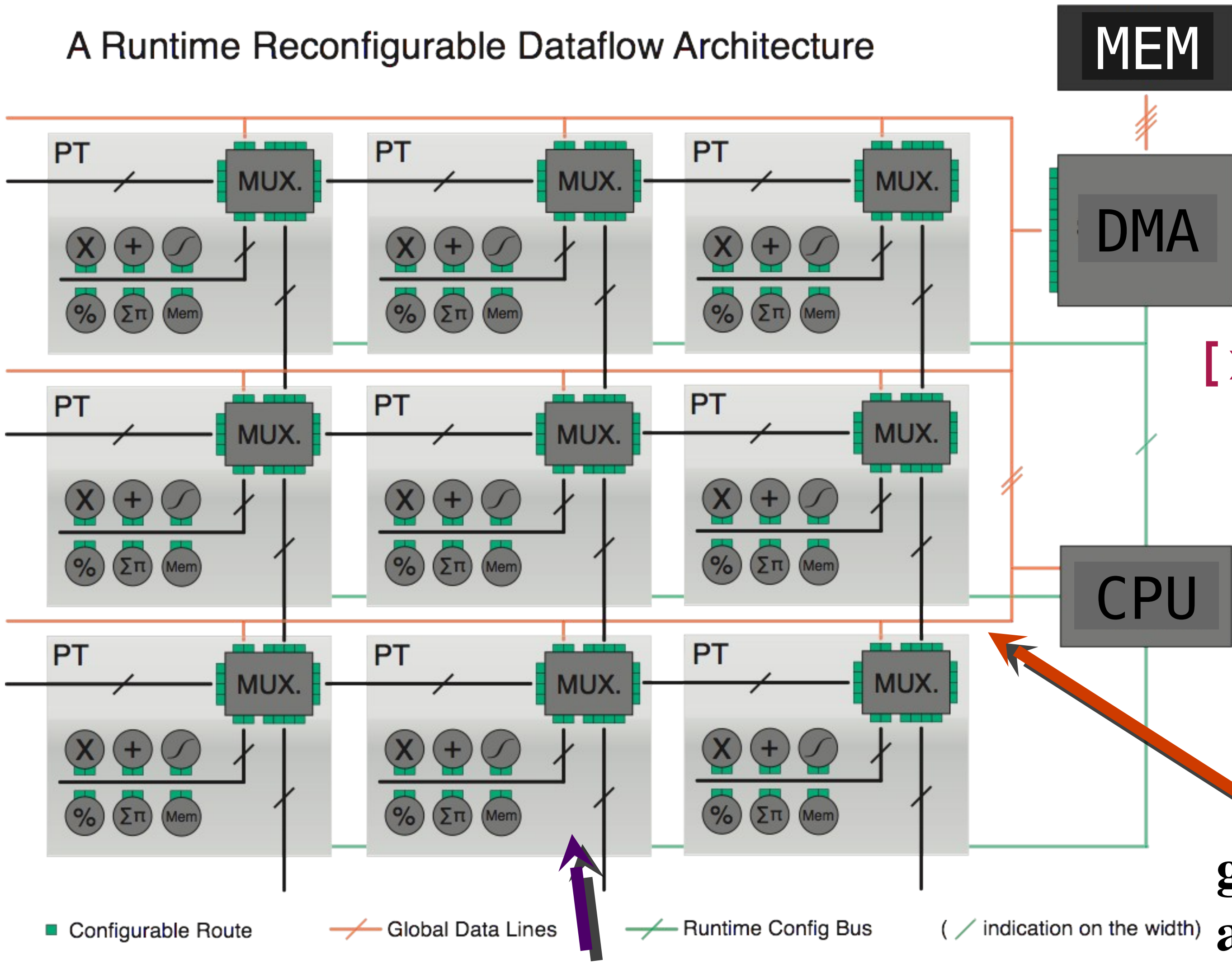
- Large-scale convolutional networks are trained on GPUs
  - Generally implemented on NVidia GPUs using CUDA
  - But exploiting all the power of GPUs for ConvNets is very difficult
  - The memory architecture is not particularly well suited for convolutions
- In the near future, other multi-core architectures may become competitive
  - e.g. Intel Xeon Phi, TI ARM+DSP, Xilinx ARM+FPGA, ARM GPUs.
- Major hardware manufacturers are exploring how to support convolutional nets more efficiently with their hardware.
  - Direct support for convnet operations in mobile CPU/GPU, as well as in high-end CPU/GPU may soon become available
- But dedicated ConvNet hardware is also on the way
  - Mostly for embedded applications (smart cameras, robots...)

NeuFlow architecture (NYU + Purdue)

- Collaboration NYU-Purdue: Eugenio Culurciello's e-Lab.
- Running on Picocomputing 8x10cm high-performance FPGA board
  - Virtex 6 LX240T: 680 MAC units, 20 neuflow tiles
- Full scene labeling at 20 frames/sec (50ms/frame) at 320x240

board with Virtex-6

NewFlow: Architecture



Multi-port memory controller (DMA)

[x12 on a V6 LX240T]

RISC CPU, to reconfigure tiles and data paths, at runtime

global network-on-chip to allow fast reconfiguration

grid of passive processing tiles (PTs)

[x20 on a Virtex6 LX240T]

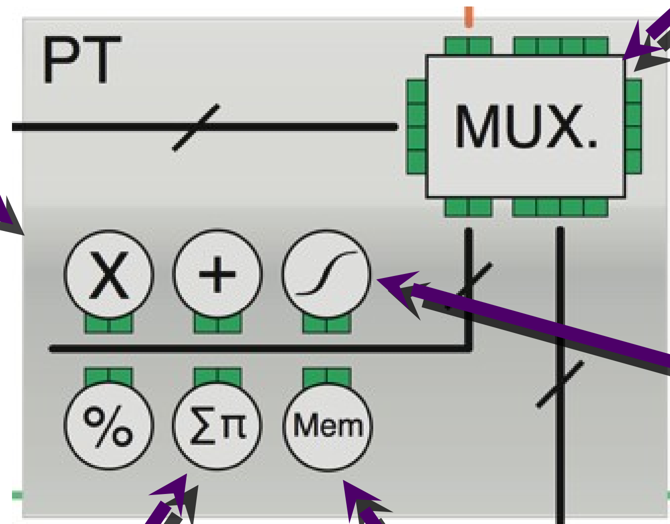
NewFlow: Processing Tile Architecture

Term-by:-term streaming operators (MUL,DIV,ADD,SUB,MAX)

[x8,2 per tile]

configurable router, to stream data in and out of the tile, to neighbors or DMA ports

[x20]



configurable piece-wise linear or quadratic mapper

[x4]

full 1/2D parallel convolver with 100 MAC units

[x4]

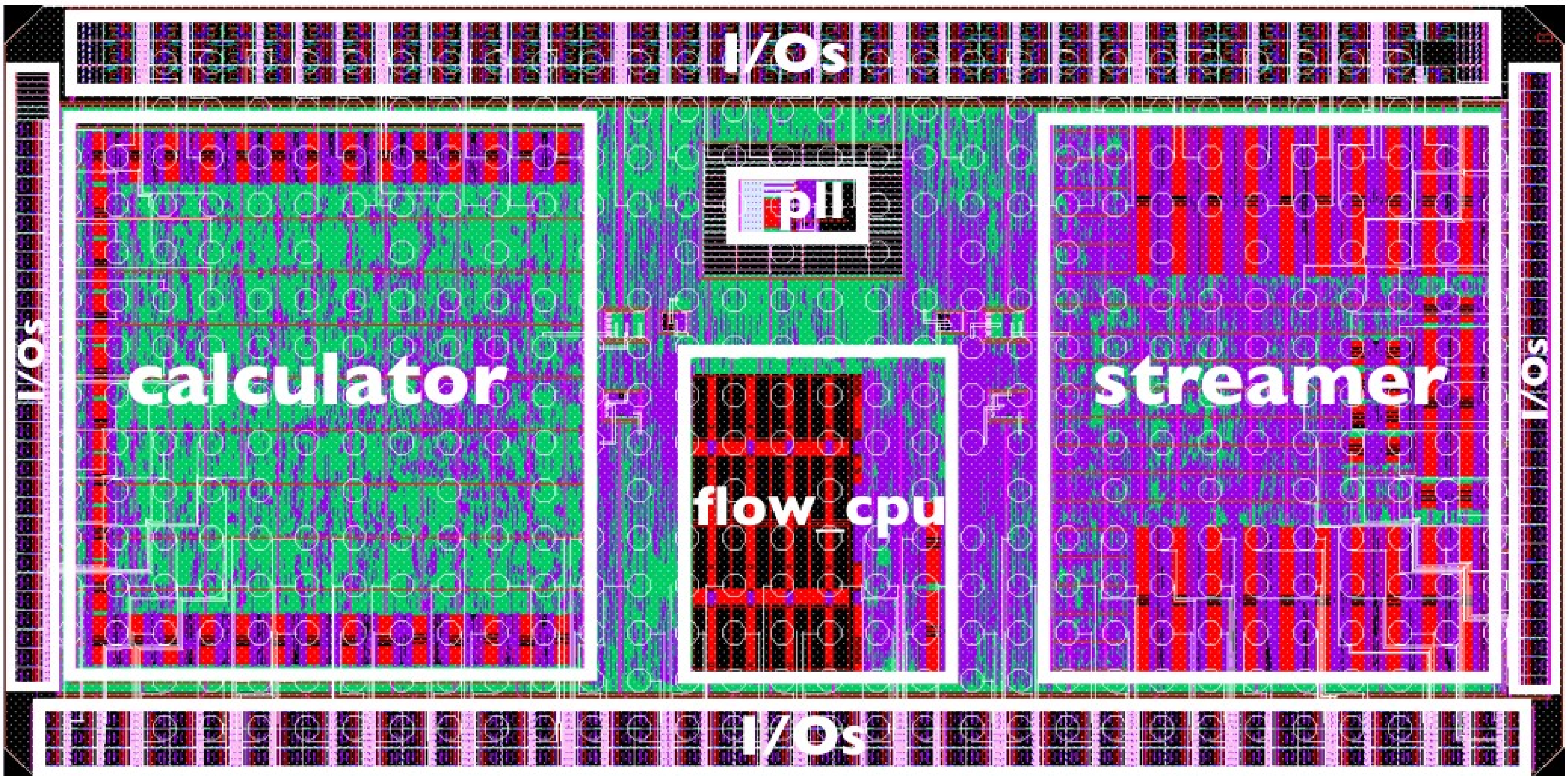
configurable bank of FIFOs , for stream buffering, up to 10kB per PT

[x8]

[Virtex6 LX240T]

NewFlow ASIC: 2.5x5 mm, 45nm, 0.6Watts, >300GOPS

- Collaboration Purdue-NYU: Eugenio Culurciello's e-Lab
- Suitable for vision-enabled embedded and mobile devices
- Status: first samples were received, but fabrication was botched...



[Pham, Jelaca, Farabet, Martini, LeCun, Culurciello 2012]

NewFlow: Performance

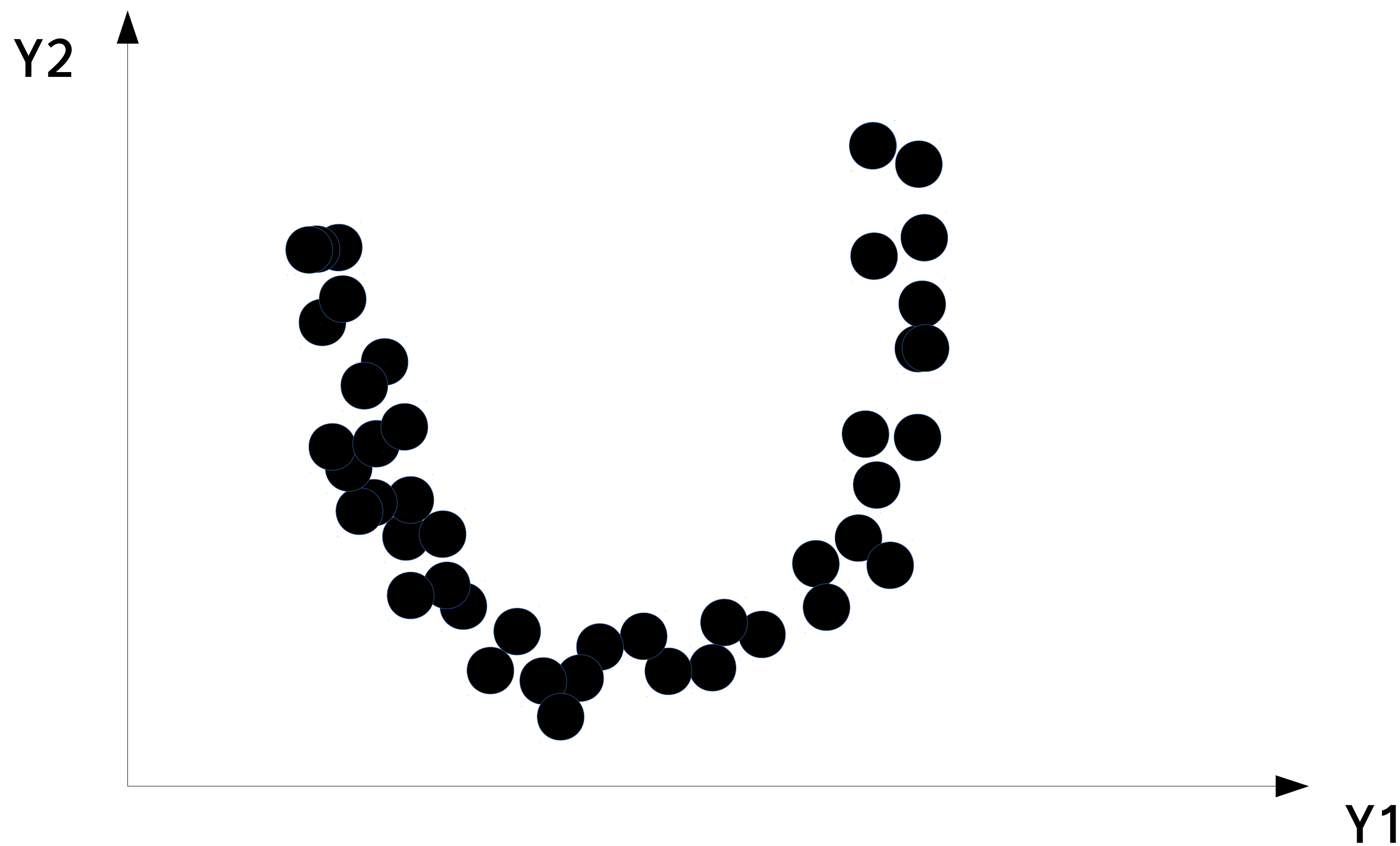
	Intel I7 4 cores	neuFlow Virtex4	neuFlow Virtex 6	nVidia GT335m	NeuFlow ASIC 45nm	nVidia GTX480
Peak GOP/sec	40	40	160	182	160	1350
Actual GOP/sec	12	37	147	54	147	294
FPS	14	46	182	67	182	374
Power (W)	50	10	10	30	0.6	220
Embed? (GOP/s/W)	0.24	3.7	14.7	1.8	245	1.34

- NeuFlow Virtex6 can run the semantic labeling system at 50ms/frame

Unsupervised Learning: Disentangling the independent, explanatory factors of variation

Energy-Based Unsupervised Learning

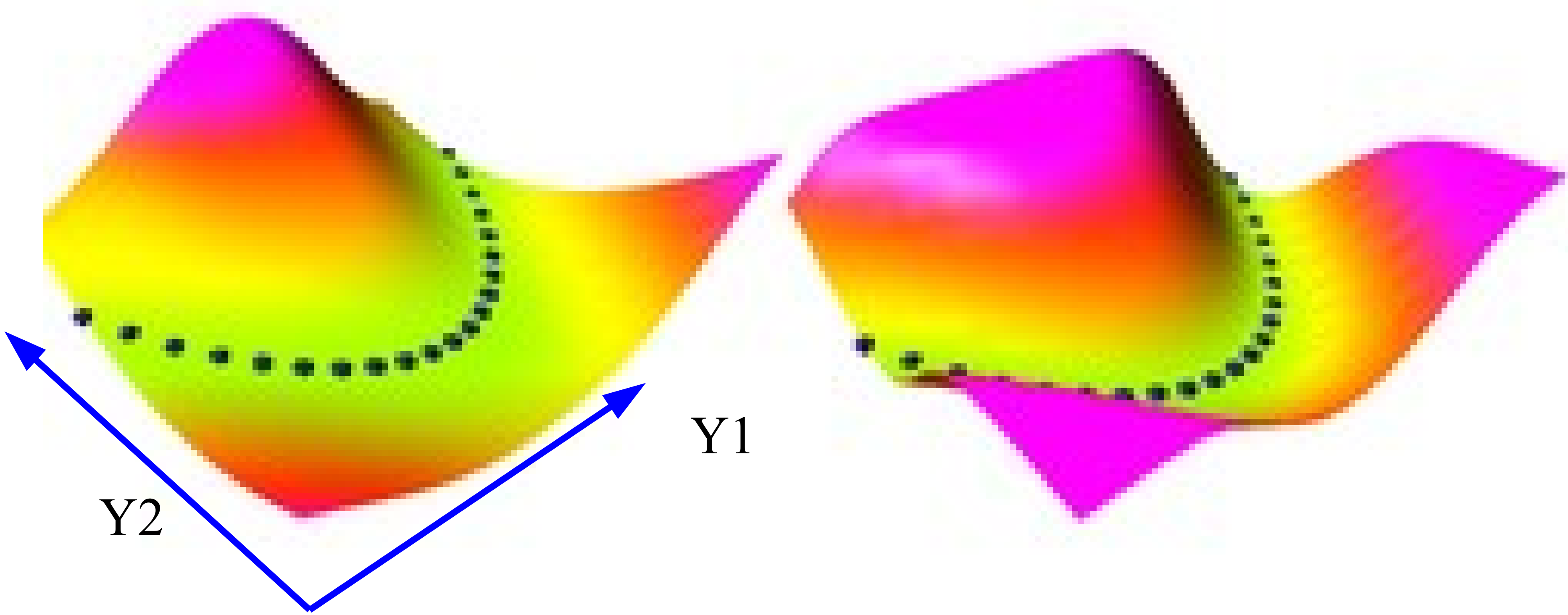
- Learning an energy function (or contrast function) that takes
  - Low values on the data manifold
  - Higher values everywhere else



Capturing Dependencies Between Variables with an Energy Function

- The energy surface is a “contrast function” that takes low values on the data manifold, and higher values everywhere else
  - Special case: energy = negative log density
  - Example: the samples live in the manifold

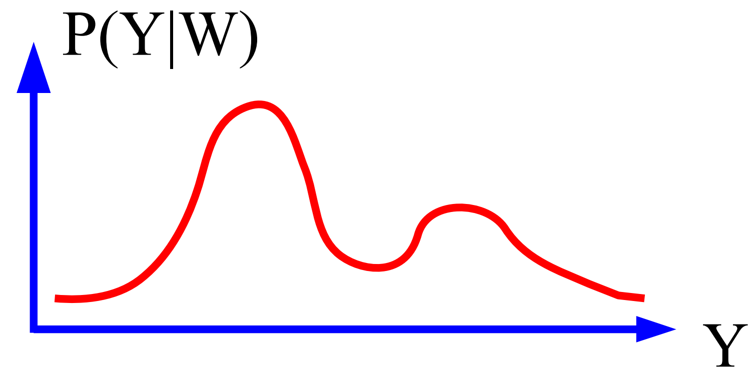
$$Y_2 = (Y_1)^2$$



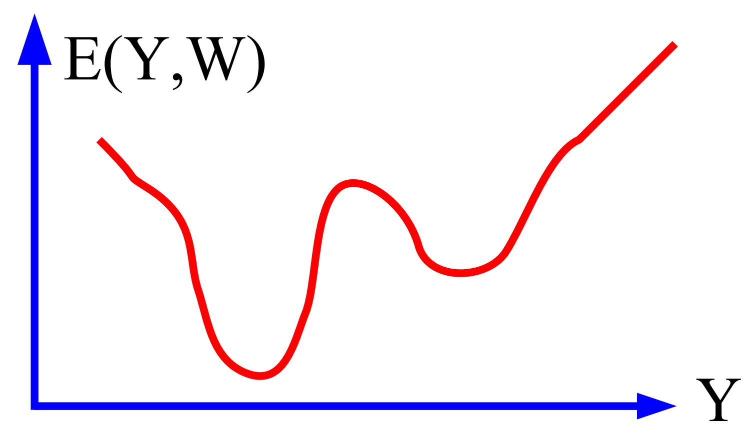
Transforming Energies into Probabilities (if necessary)

- The energy can be interpreted as an unnormalized negative log density
- Gibbs distribution: Probability proportional to exp(-energy)
  - Beta parameter is akin to an inverse temperature
- Don't compute probabilities unless you absolutely have to
  - Because the denominator is often intractable

$$P(Y|W) = \frac{e^{-\beta E(Y,W)}}{\int_y e^{-\beta E(y,W)}}$$

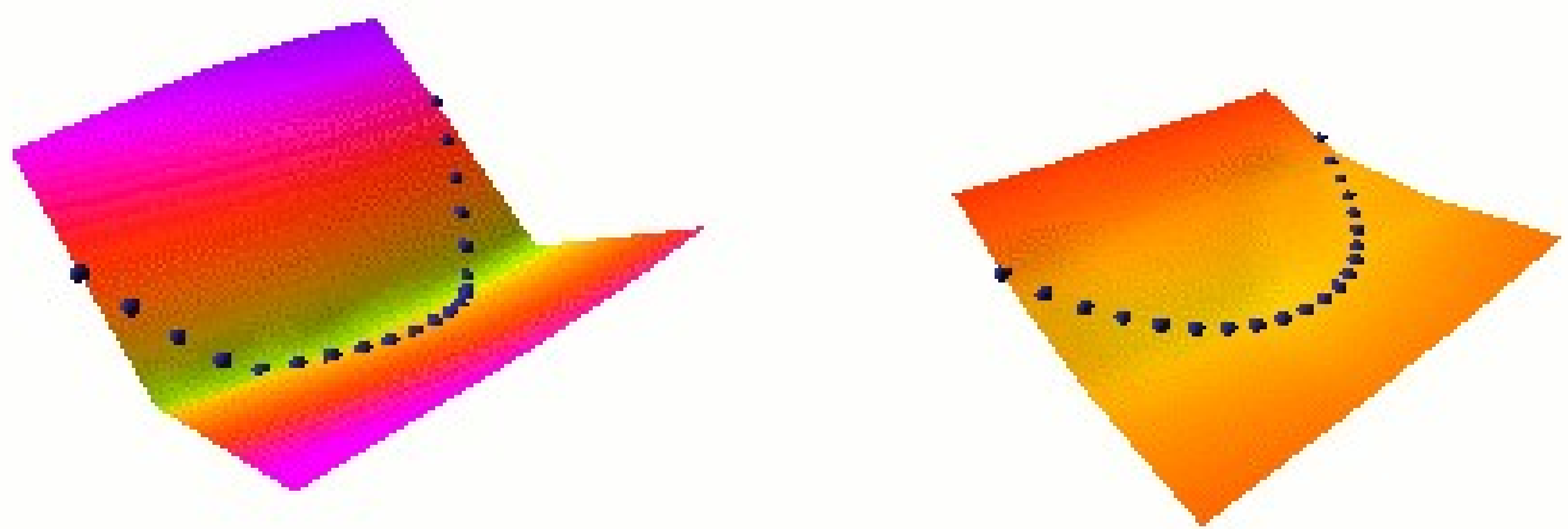


$$E(Y,W) \propto -\log P(Y|W)$$



Learning the Energy Function

- parameterized energy function E(Y,W)
  - Make the energy low on the samples
  - Make the energy higher everywhere else
  - Making the energy low on the samples is easy
  - But how do we make it higher everywhere else?



Seven Strategies to Shape the Energy Function

- 1. build the machine so that the volume of low energy stuff is constant
  - PCA, K-means, GMM, square ICA
- 2. push down of the energy of data points, push up everywhere else
  - Max likelihood (needs tractable partition function)
- 3. push down of the energy of data points, push up on chosen locations
  - contrastive divergence, Ratio Matching, Noise Contrastive Estimation, Minimum Probability Flow
- 4. minimize the gradient and maximize the curvature around data points
  - score matching
- 5. train a dynamical system so that the dynamics goes to the manifold
  - denoising auto-encoder
- 6. use a regularizer that limits the volume of space that has low energy
  - Sparse coding, sparse auto-encoder, PSD
- 7. if E(Y) = ||Y - G(Y)||^2, make G(Y) as "constant" as possible.
  - Contracting auto-encoder, saturating auto-encoder

#1: constant volume of low energy
Energy surface for PCA and K-means

- 1. build the machine so that the volume of low energy stuff is constant
  - PCA, K-means, GMM, square ICA...

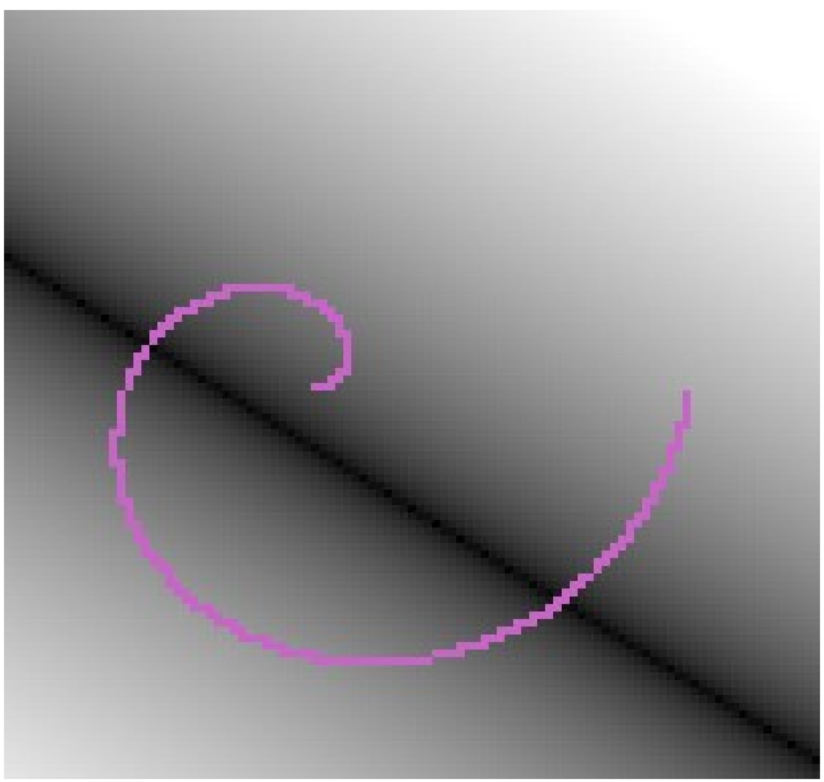
PCA

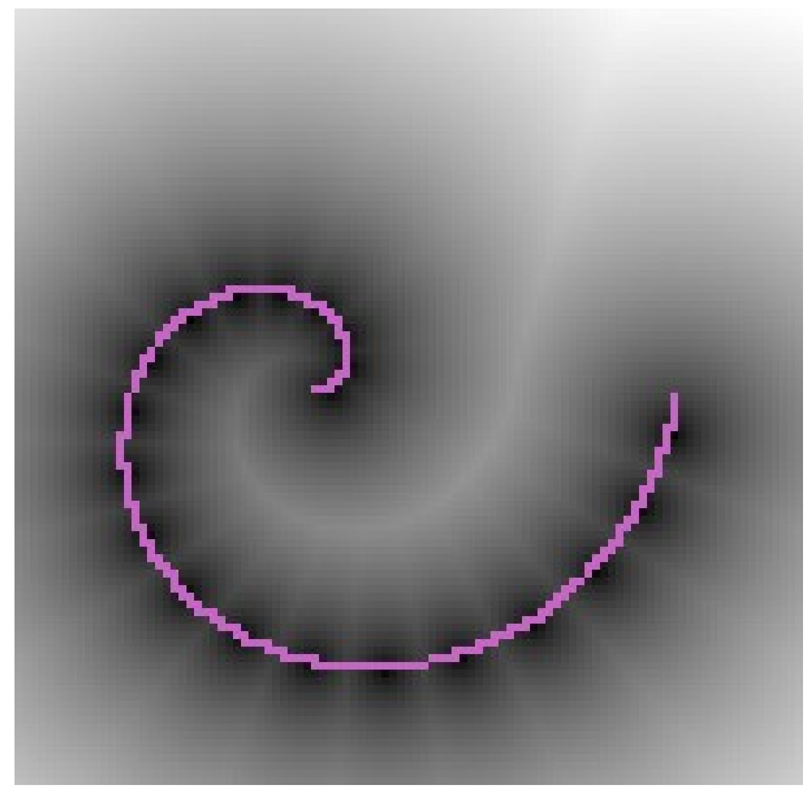
$$E(Y) = \| W^T W Y - Y \|^2$$

K-Means,

Z constrained to 1-of-K code

$$E(Y) = min_z \sum_i \| Y - W_i Z_i \|^2$$





#2: push down of the energy of data points, push up everywhere else

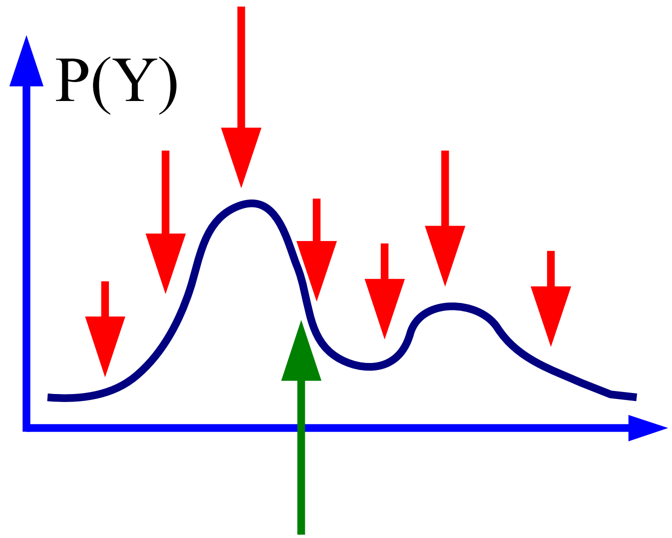
- **Max likelihood (requires a tractable partition function)**

Maximizing P(Y|W) on training samples

make this big

$$P(Y|W) = \frac{e^{-\beta E(Y,W)}}{\int_y e^{-\beta E(y,W)}}$$

make this small

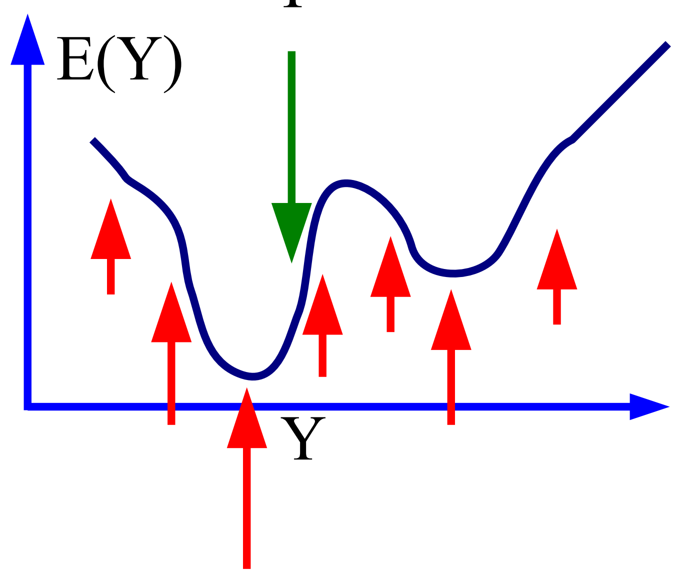


Minimizing -log P(Y,W) on training samples

$$L(Y,W) = E(Y,W) + \frac{1}{\beta}\log\int_y e^{-\beta E(y,W)}$$

make this small

make this big



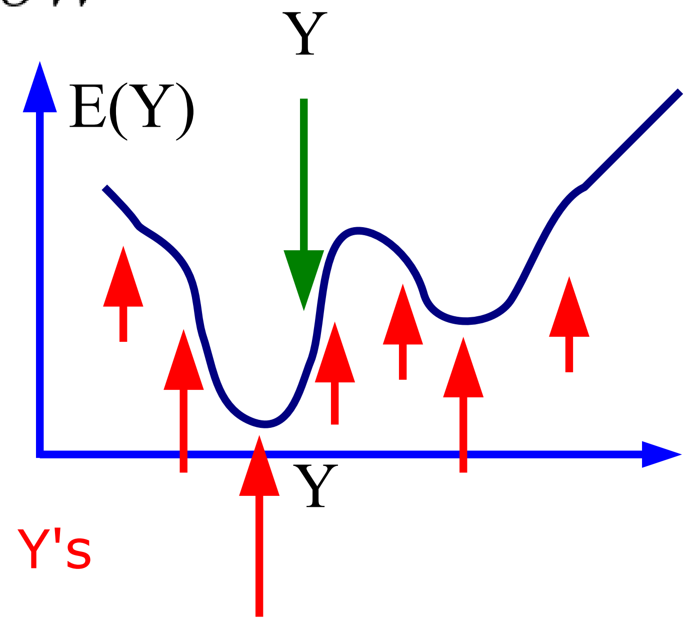
#2: push down of the energy of data points, push up everywhere else

Gradient of the negative log-likelihood loss for one sample Y:

$$\frac{\partial L(Y,W)}{\partial W} = \frac{\partial E(Y,W)}{\partial W} - \int_y P(y|W)\frac{\partial E(y,W)}{\partial W}$$

Gradient descent:

$$W \leftarrow W - \eta \frac{\partial L(Y,W)}{\partial W}$$



Pushes down on the energy of the samples

Pulls up on the energy of low-energy Y's

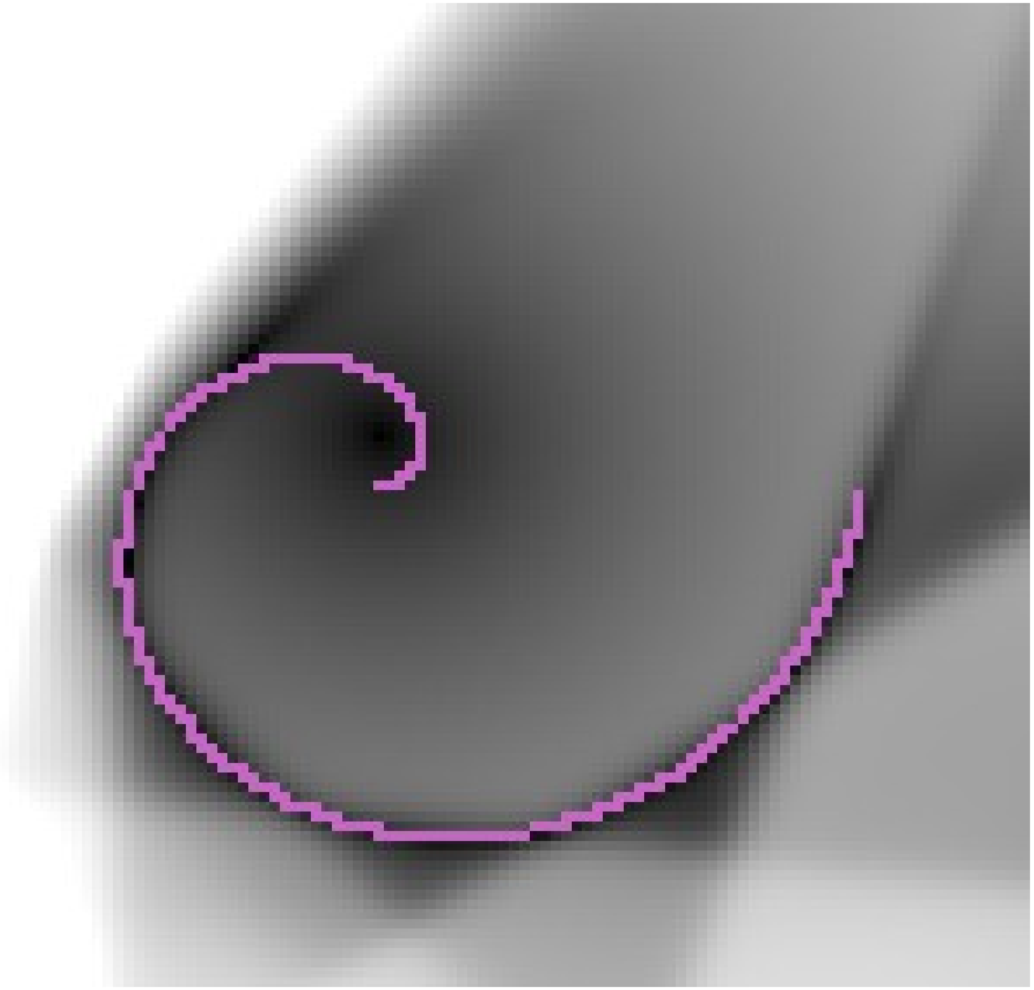
$$W \leftarrow W - \eta \frac{\partial E(Y,W)}{\partial W} + \eta \int_y P(y|W)\frac{\partial E(y,W)}{\partial W}$$

#3. push down of the energy of data points, push up on chosen locations

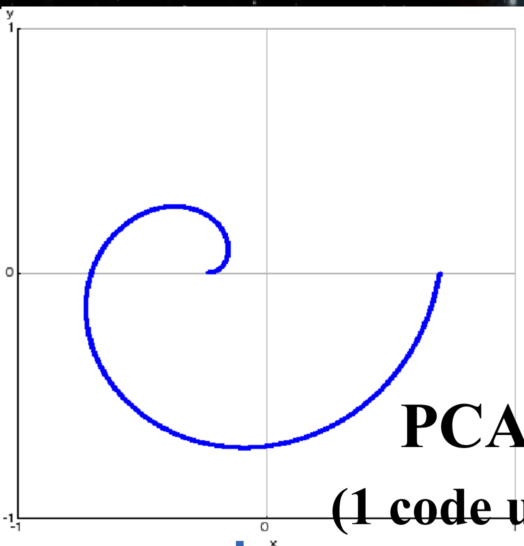
- contrastive divergence, Ratio Matching, Noise Contrastive Estimation, Minimum Probability Flow
- Contrastive divergence: basic idea
  - Pick a training sample, lower the energy at that point
  - From the sample, move down in the energy surface with noise
  - Stop after a while
  - Push up on the energy of the point where we stopped
  - This creates grooves in the energy surface around data manifolds
  - CD can be applied to any energy function (not just RBMs)
- Persistent CD: use a bunch of "particles" and remember their positions
  - Make them roll down the energy surface with noise
  - Push up on the energy wherever they are
  - Faster than CD
- RBM $E(Y,Z) = -Z^T WY$ $\quad E(Y) = -\log \sum_z e^{Z^T WY}$

#6. use a regularizer that limits the volume of space that has low energy

- Sparse coding, sparse auto-encoder, Predictive Saprse Decomposition

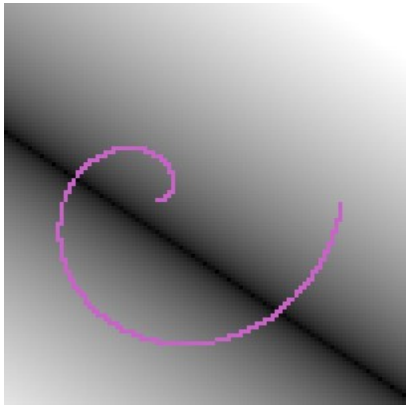


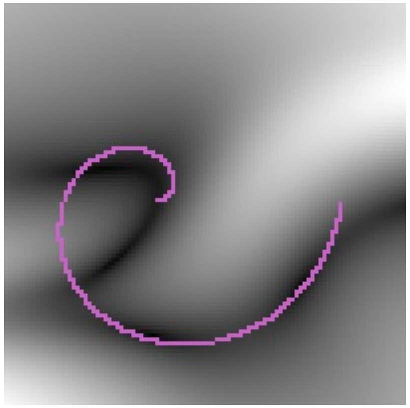
Energy Functions of Various Methods

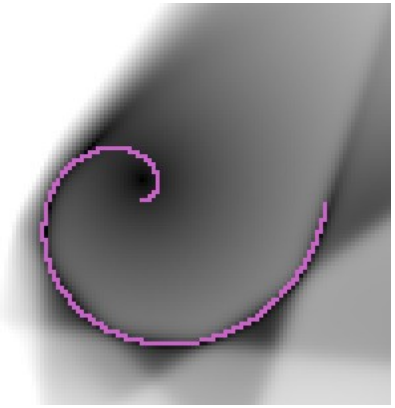


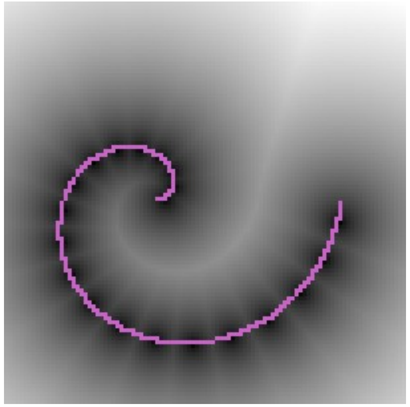
- 2 dimensional toy dataset: spiral
- Visualizing energy surface
  - (black = low, white = high)

	PCA (1 code unit)	**autoencoder (1 code unit)**	**sparse coding (20 code units)**	**K-Means (20 code units)**
encoder	$W'Y$	$\sigma(W_e Y)$	$\sigma(W_e Z)$	$-$
decoder	WZ	$W_d Z$	$W_d Z$	WZ
energy	$\|\|Y-WZ\|\|^2$	$\|\|Y-WZ\|\|^2$	$\|\|Y-WZ\|\|^2$	$\|\|Y-WZ\|\|^2$
loss	$F(Y)$	$F(Y)$	$F(Y)$	$F(Y)$
pull-up	dimens.	dimens.	sparsity	1-of-N code





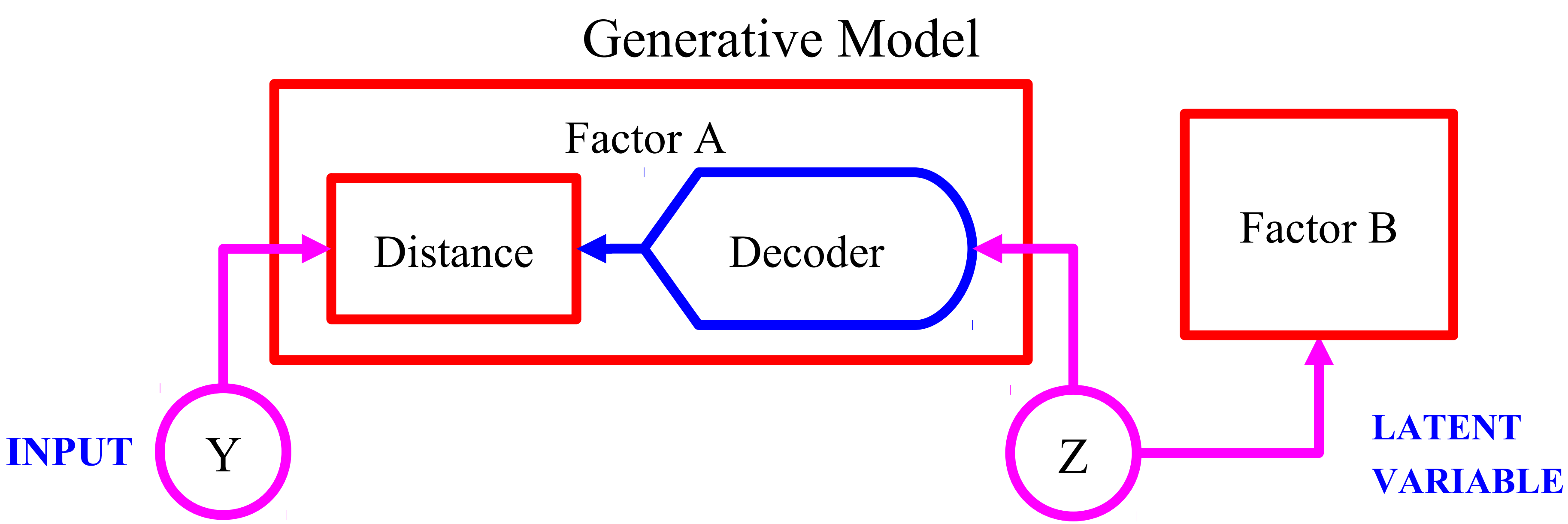




Sparse Modeling,
Sparse Auto-Encoders,
Predictive Sparse Decomposition
LISTA

How to Speed Up Inference in a Generative Model?

- Factor Graph with an asymmetric factor
- Inference Z → Y is easy
  - Run Z through deterministic decoder, and sample Y
- Inference Y → Z is hard, particularly if Decoder function is many-to-one
  - MAP: minimize sum of two factors with respect to Z
  - Z* = argmin_z Distance[Decoder(Z), Y] + FactorB(Z)
- Examples: K-Means (1of K), Sparse Coding (sparse), Factor Analysis

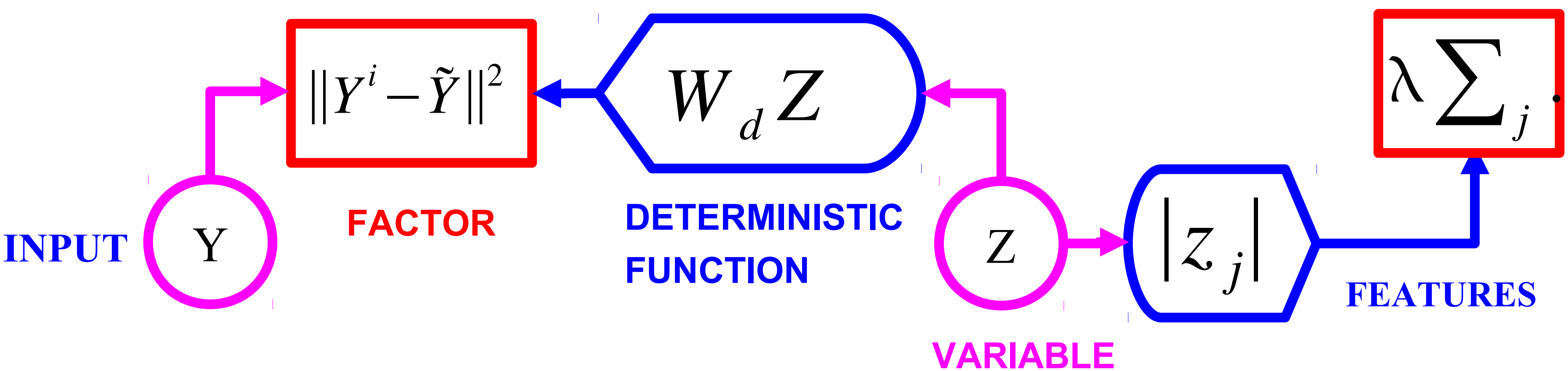


Sparse Coding & Sparse Modeling

[Olshausen & Field 1997]

- Sparse linear reconstruction
- Energy = reconstruction_error + code_prediction_error + code_sparsity

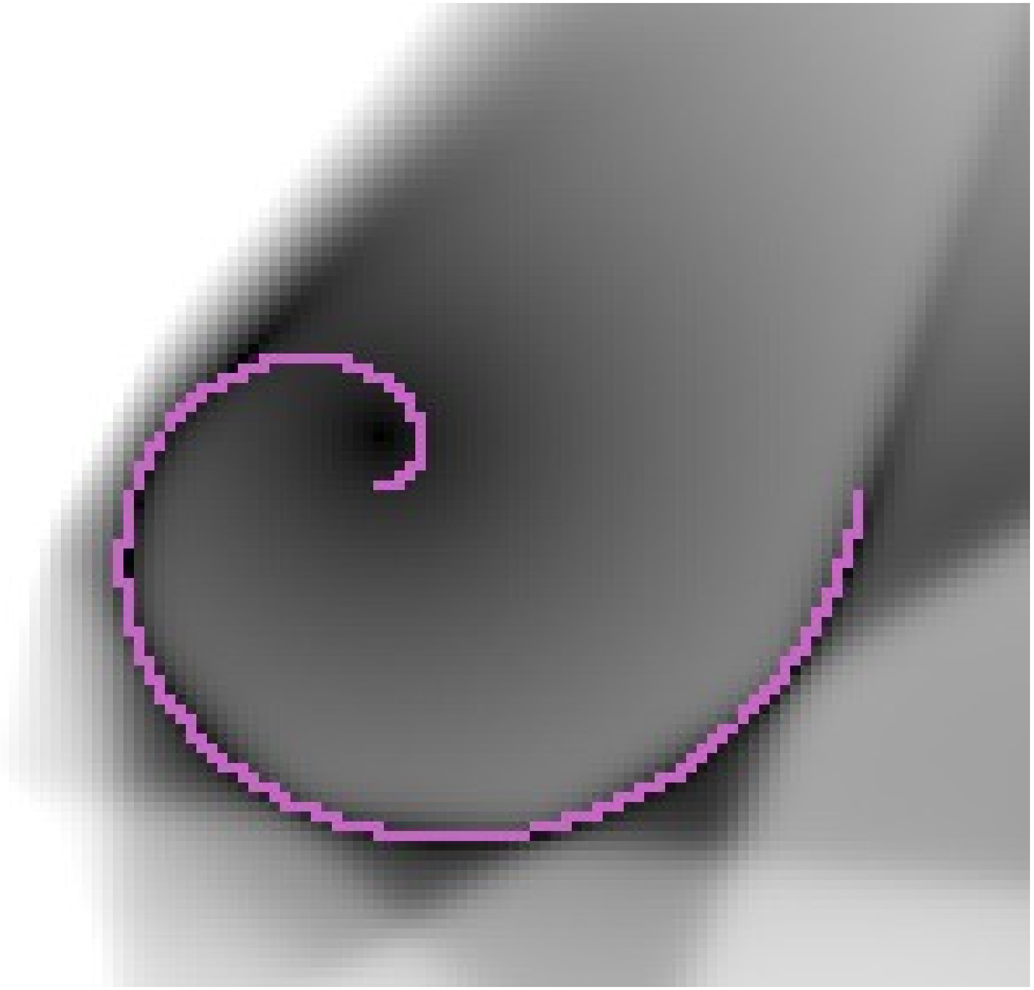
$$E(Y^i, Z) = \|Y^i - W_d Z\|^2 + \lambda \sum_j |z_j|$$



- Inference is slow $Y \rightarrow \hat{Z} = argmin_Z E(Y, Z)$

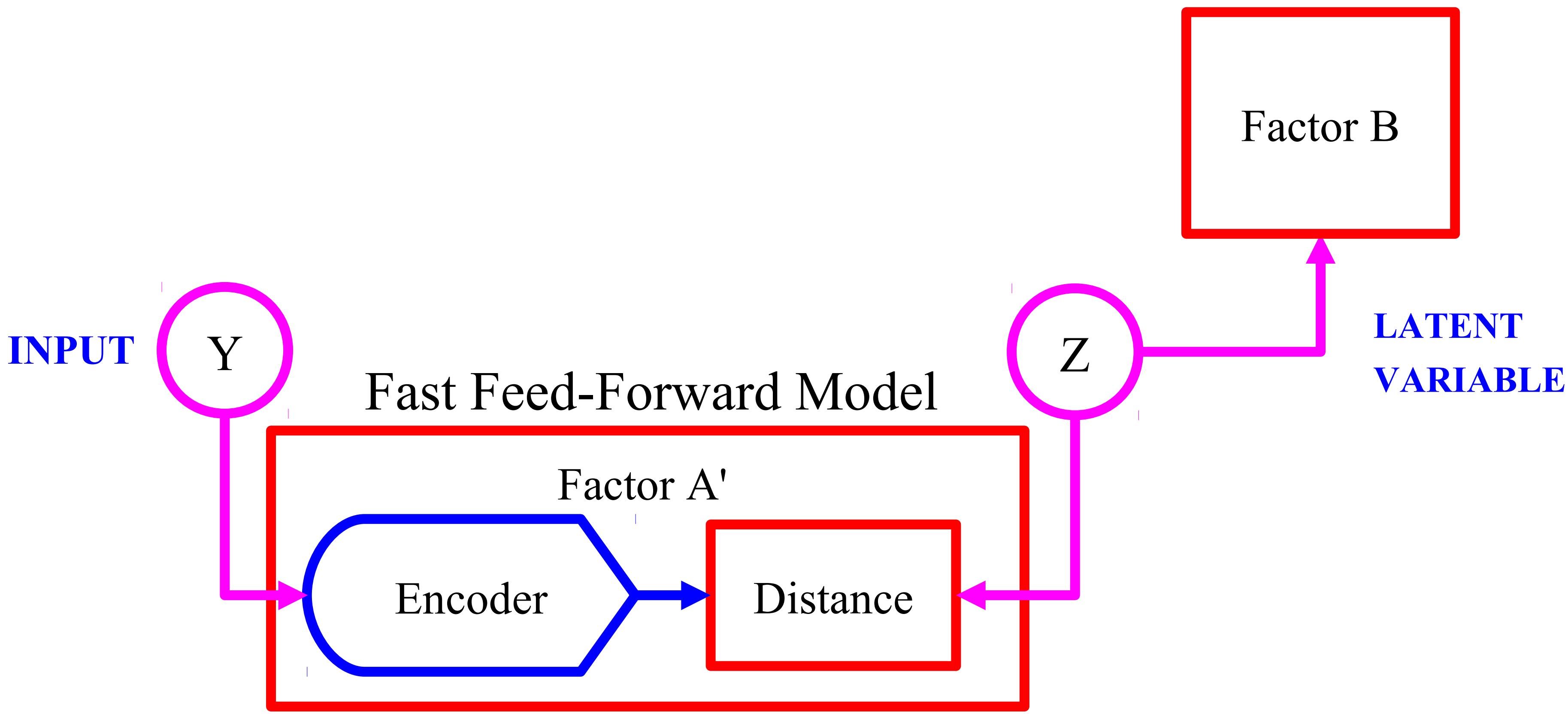
#6. use a regularizer that limits the volume of space that has low energy

- Sparse coding, sparse auto-encoder, Predictive Saprse Decomposition



Encoder Architecture

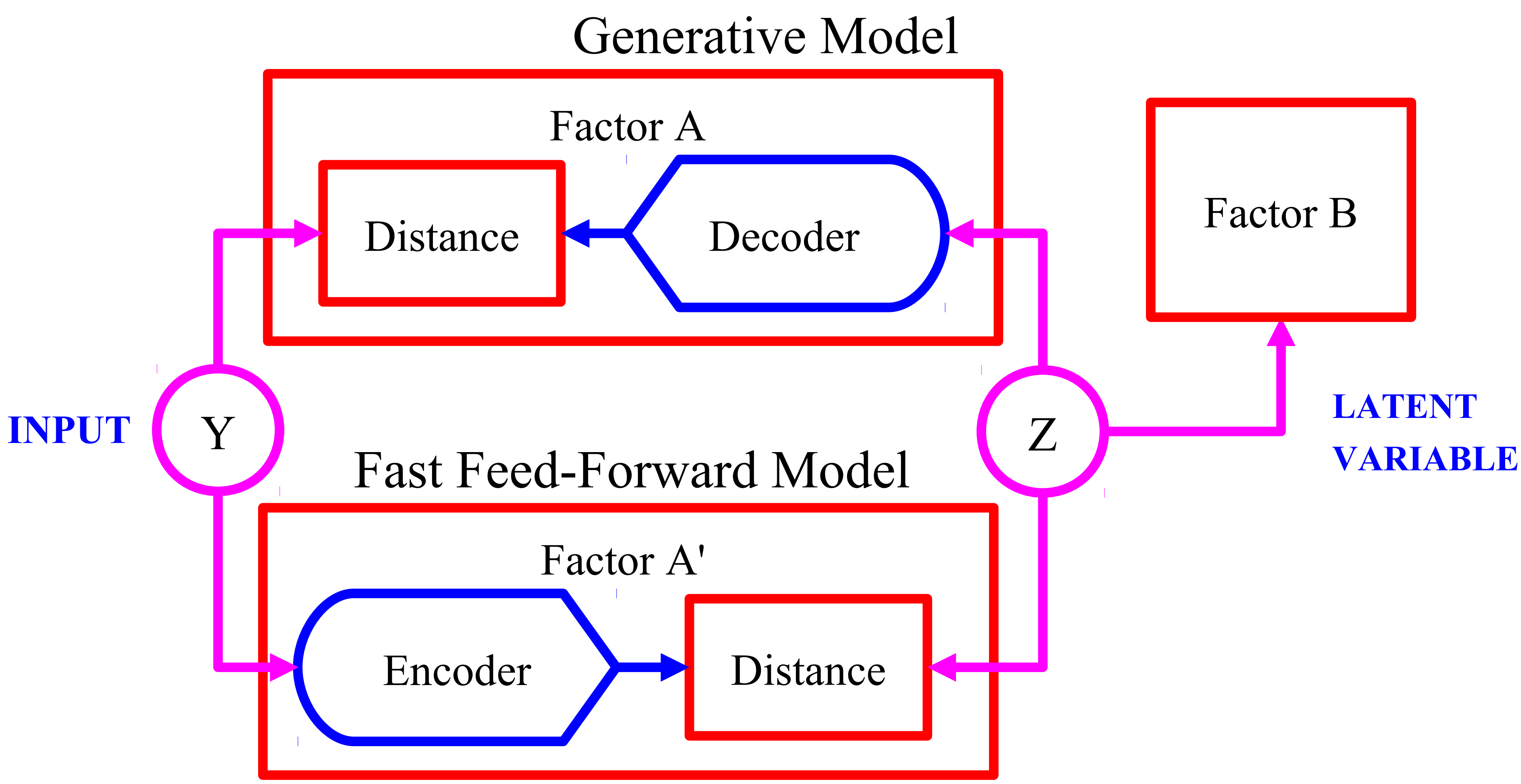
- Examples: most ICA models, Product of Experts



Encoder-Decoder Architecture

[Kavukcuoglu, Ranzato, LeCun, rejected by every conference, 2008-2009]

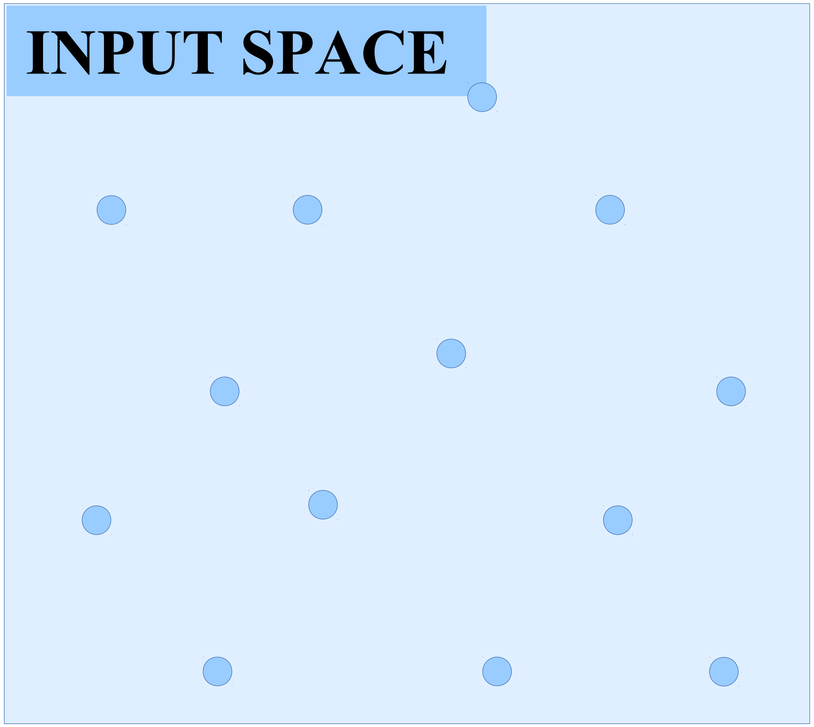
- Train a “simple” feed-forward function to predict the result of a complex optimization on the data points of interest

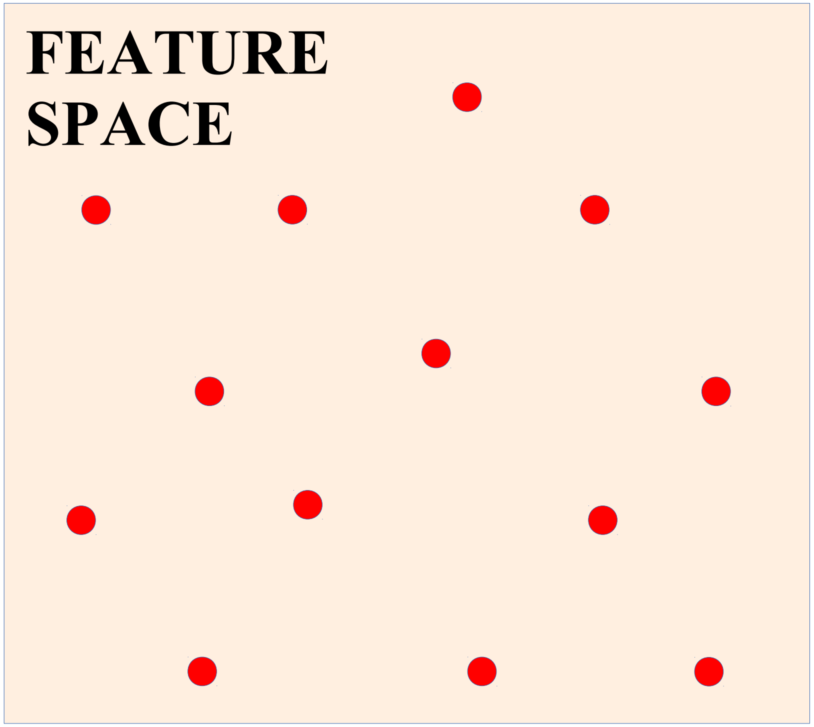


- 1. Find optimal Zi for all Yi; 2. Train Encoder to predict Zi from Yi

Why Limit the Information Content of the Code?

- Training sample
- Input vector which is NOT a training sample
- Feature vector

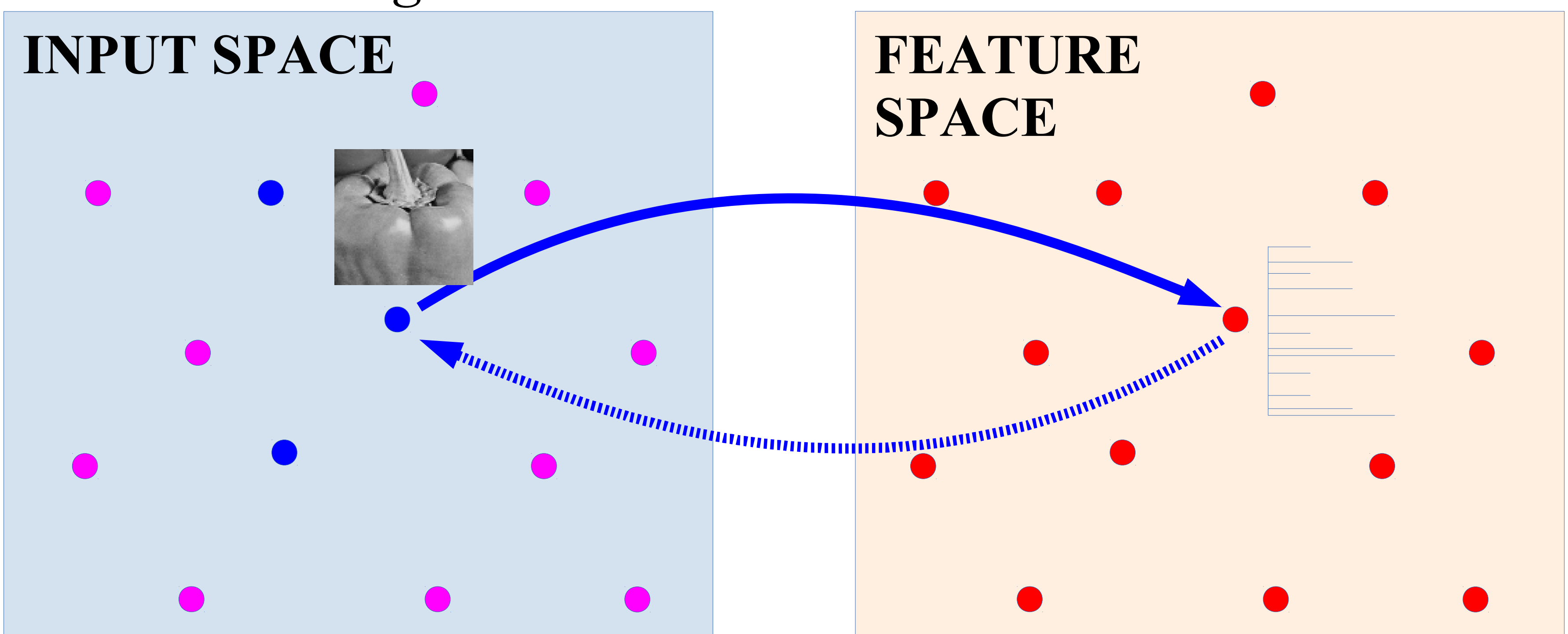




Why Limit the Information Content of the Code?

- **Training sample**
- **Input vector which is NOT a training sample**
- **Feature vector**

Training based on minimizing the reconstruction error over the training set

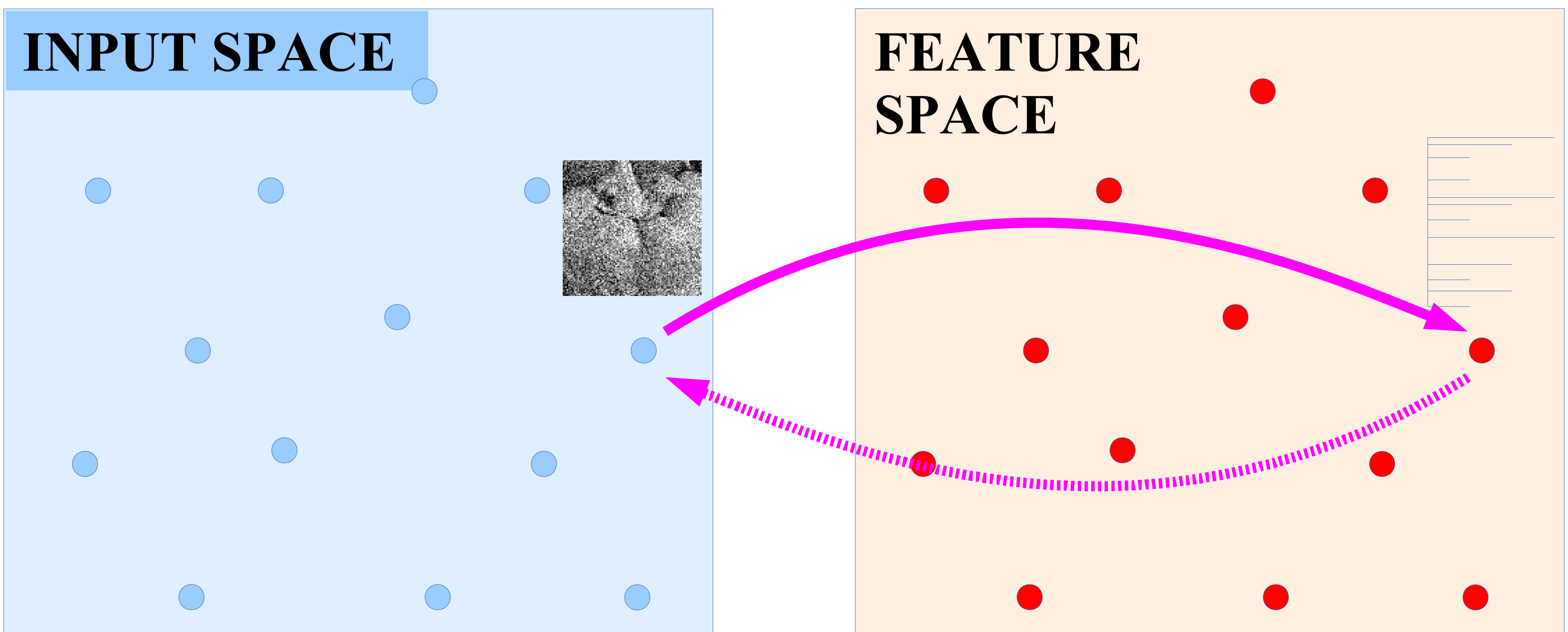


Why Limit the Information Content of the Code?

- **Training sample**
- **Input vector which is NOT a training sample**
- **Feature vector**

BAD: machine does not learn structure from training data!!

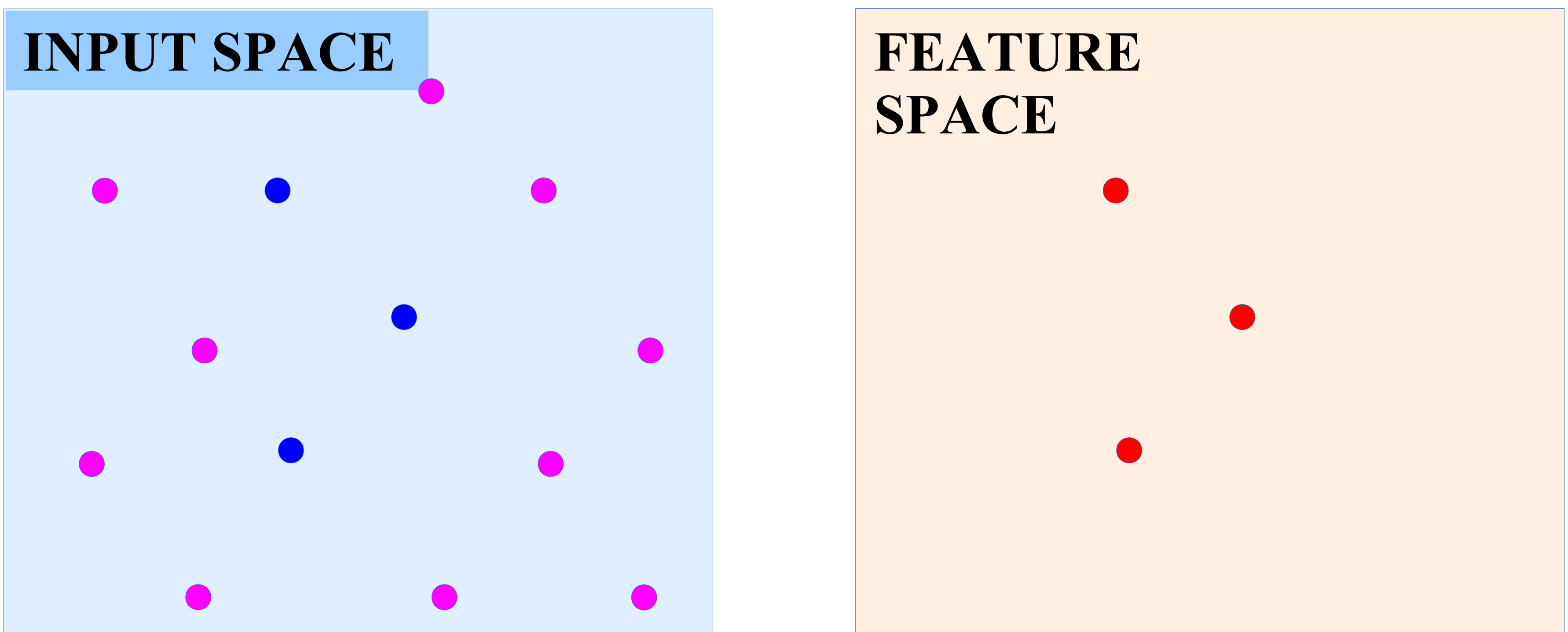
It just copies the data.



Why Limit the Information Content of the Code?

- Training sample
- Input vector which is NOT a training sample
- Feature vector

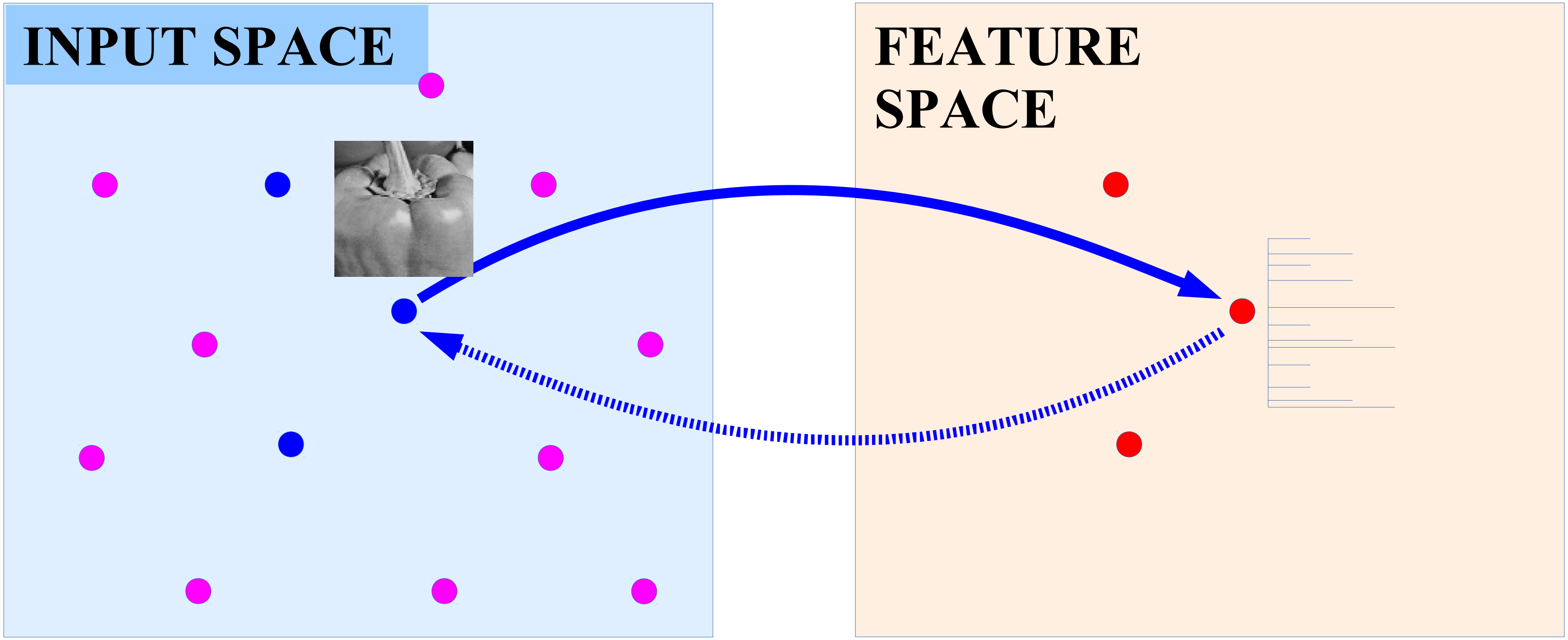
IDEA: reduce number of available codes.



Why Limit the Information Content of the Code?

- Training sample
- Input vector which is NOT a training sample
- Feature vector

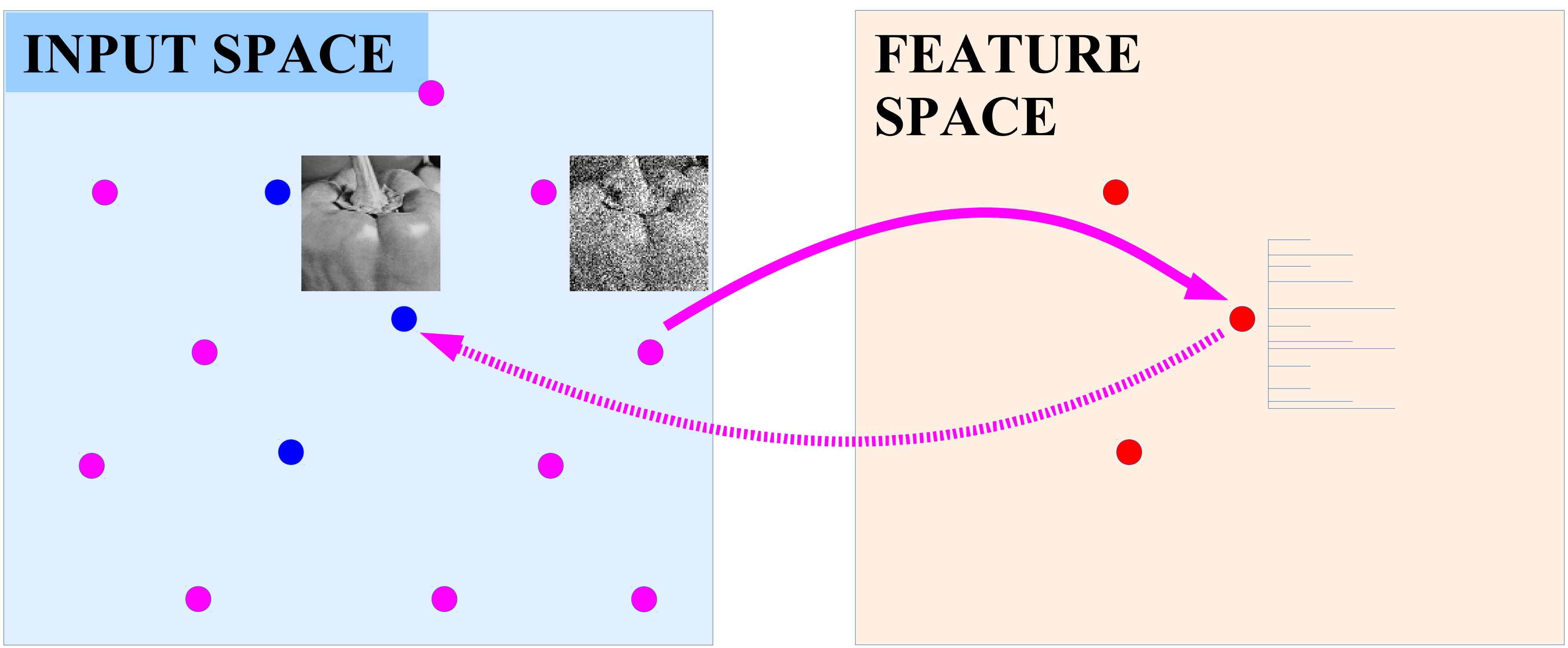
IDEA: reduce number of available codes.



Why Limit the Information Content of the Code?

- **Training sample**
- **Input vector which is NOT a training sample**
- **Feature vector**

IDEA: reduce number of available codes.



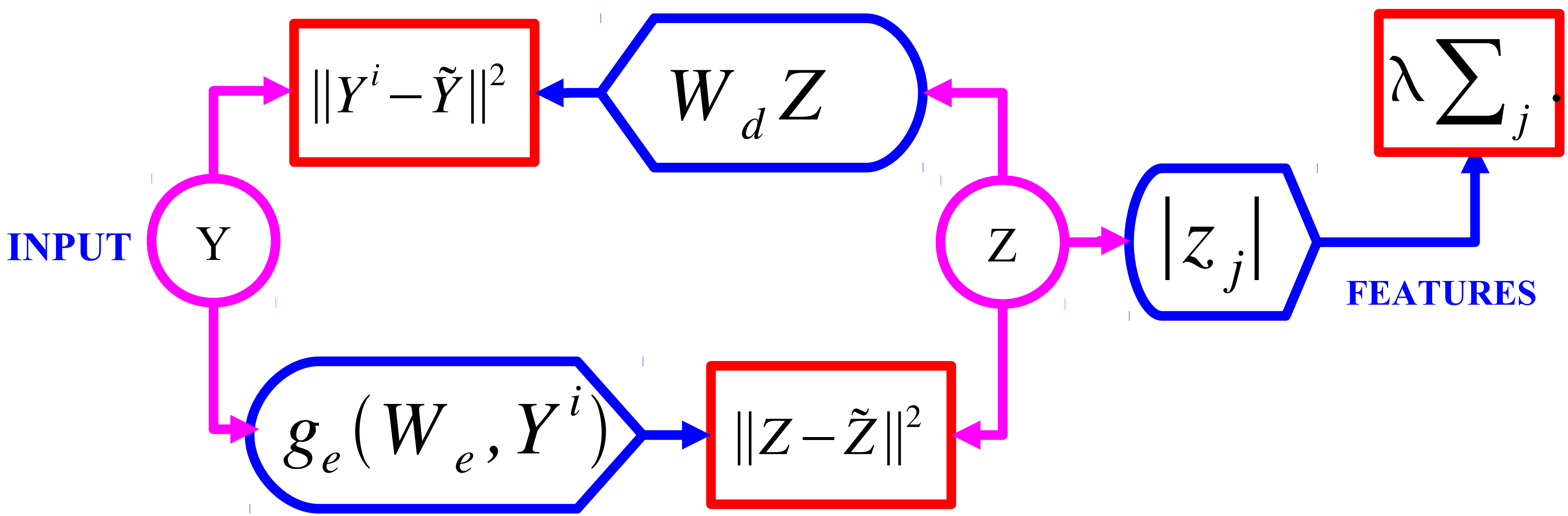
Predictive Sparse Decomposition (PSD): sparse auto-encoder

[Kavukcuoglu, Ranzato, LeCun, 2008 → arXiv:1010.3467],

- Prediction the optimal code with a trained encoder
- Energy = reconstruction_error + code_prediction_error + code_sparsity

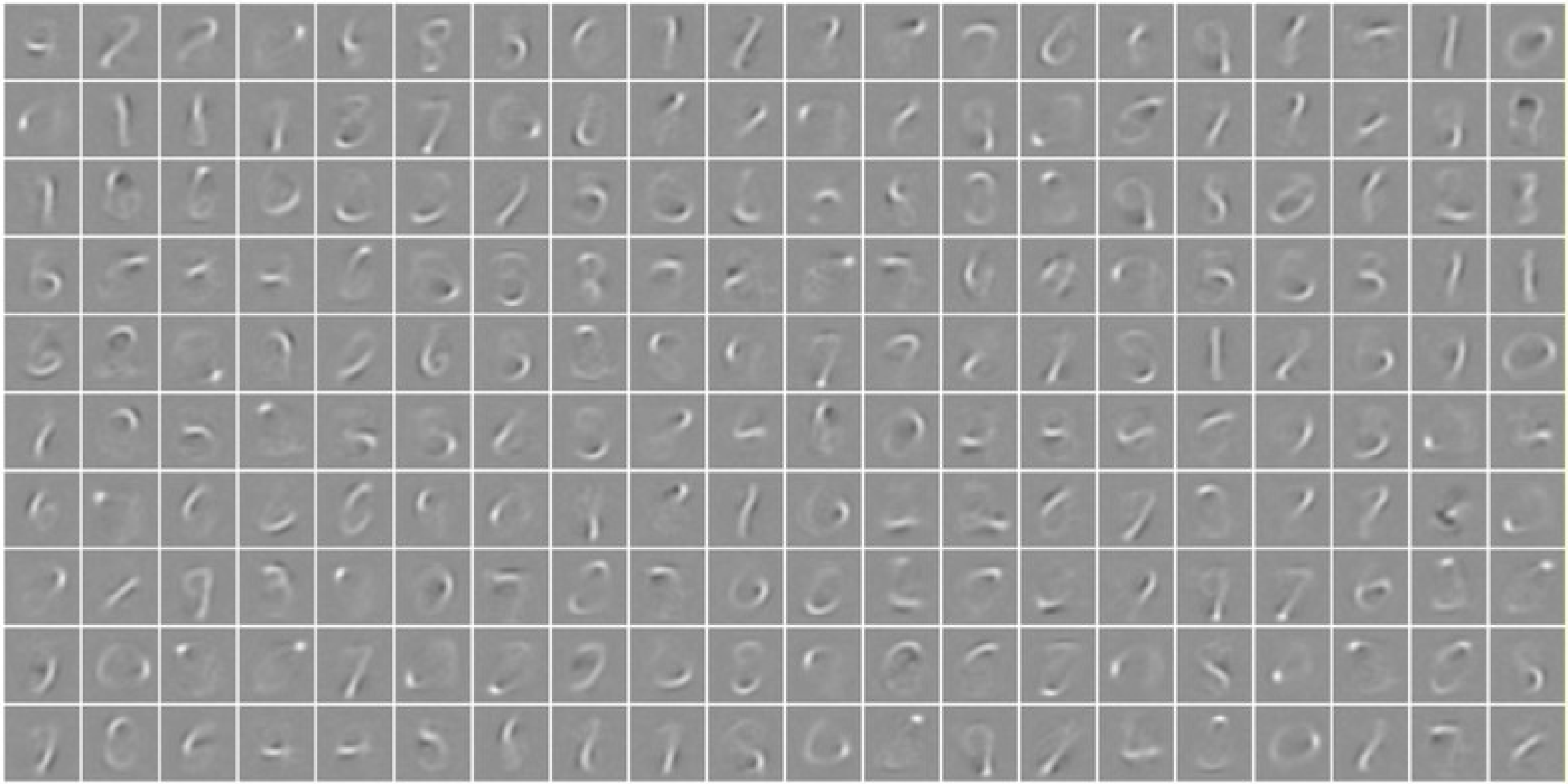
$$E(Y^i, Z) = \|Y^i - W_d Z\|^2 + \|Z - g_e(W_e, Y^i)\|^2 + \lambda \sum_j |z_j|$$

$$g_e(W_e, Y^i) = shrinkage(W_e Y^i)$$



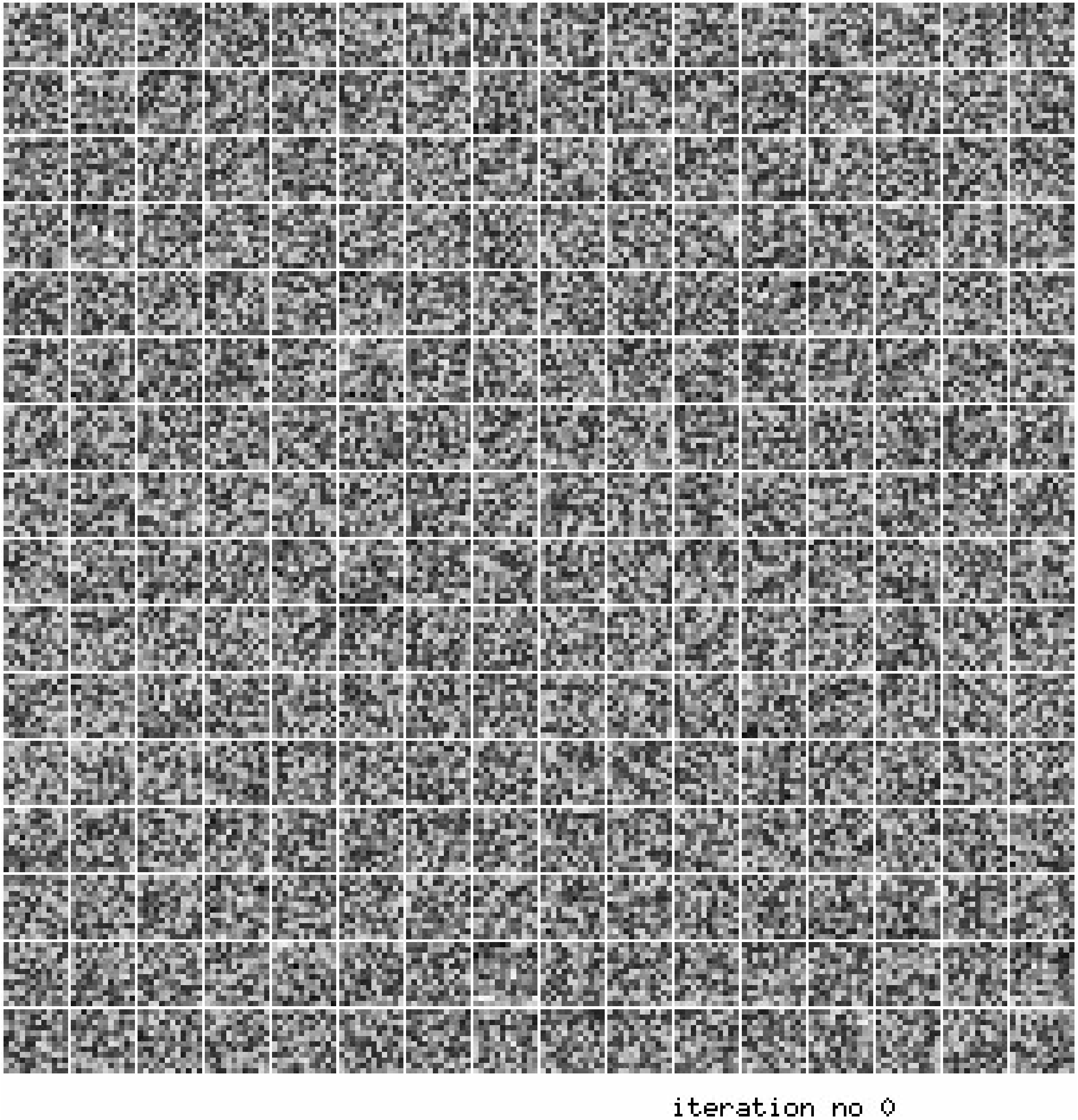
PSD: Basis Functions on MNIST

- Basis functions (and encoder matrix) are digit parts

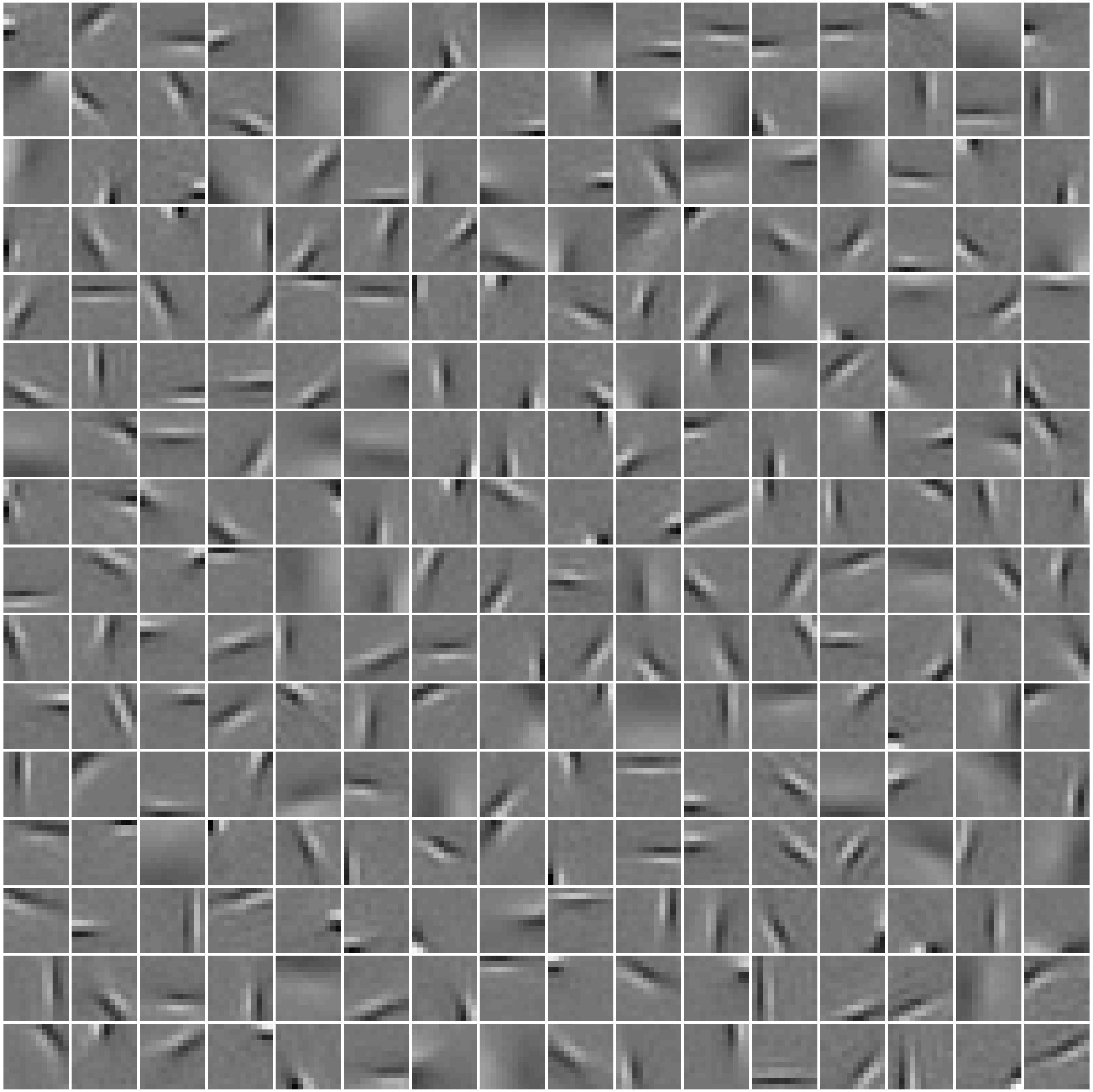


Predictive Sparse Decomposition (PSD): Training

- Training on natural images patches.
  - 12X12
  - 256 basis functions



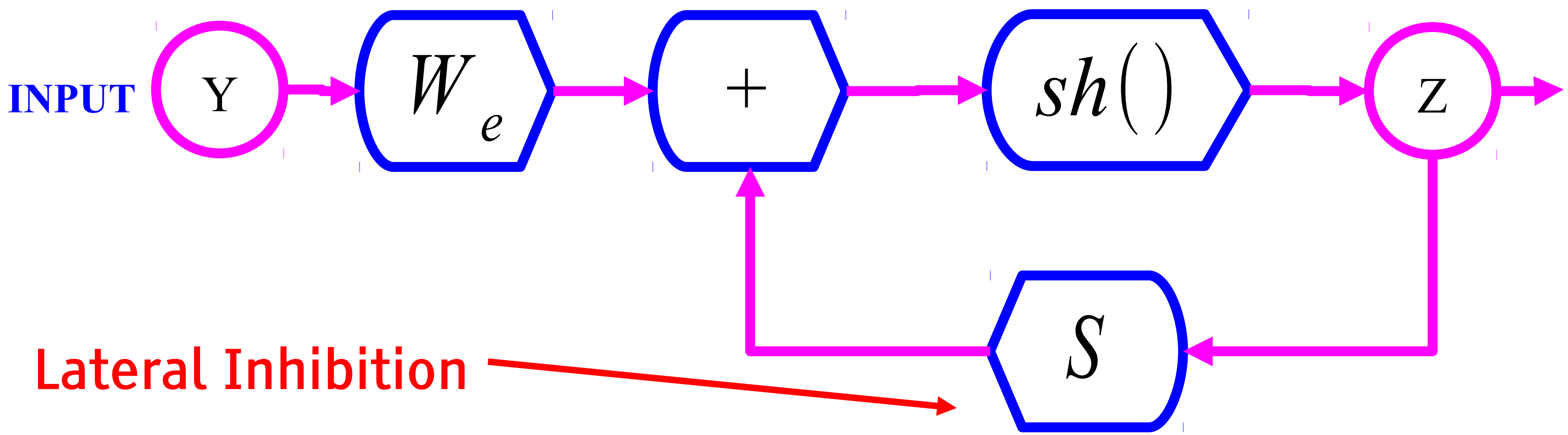
Learned Features on natural patches: V1-like receptive fields



Better Idea: Give the "right" structure to the encoder

- ISTA/FISTA: iterative algorithm that converges to optimal sparse code

[Gregor & LeCun, ICML 2010], [Bronstein et al. ICML 2012], [Rolfe & LeCun ICLR 2013]



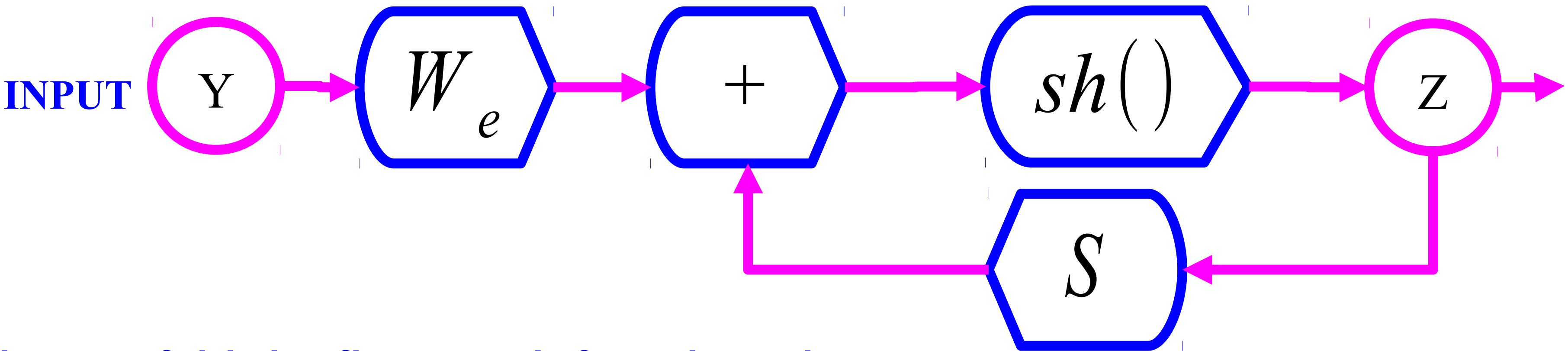
$$Z(t+1) = \mathrm{Shrinkage}_{\lambda/L}\left[Z(t) - \frac{1}{L} W_d^T (W_d Z(t) - Y)\right]$$

$$Z(t+1) = \mathrm{Shrinkage}_{\lambda/L}\left[W_e^T Y + S Z(t)\right]; \quad W_e = \frac{1}{L} W_d; \quad S = I - \frac{1}{L} W_d^T W_d$$

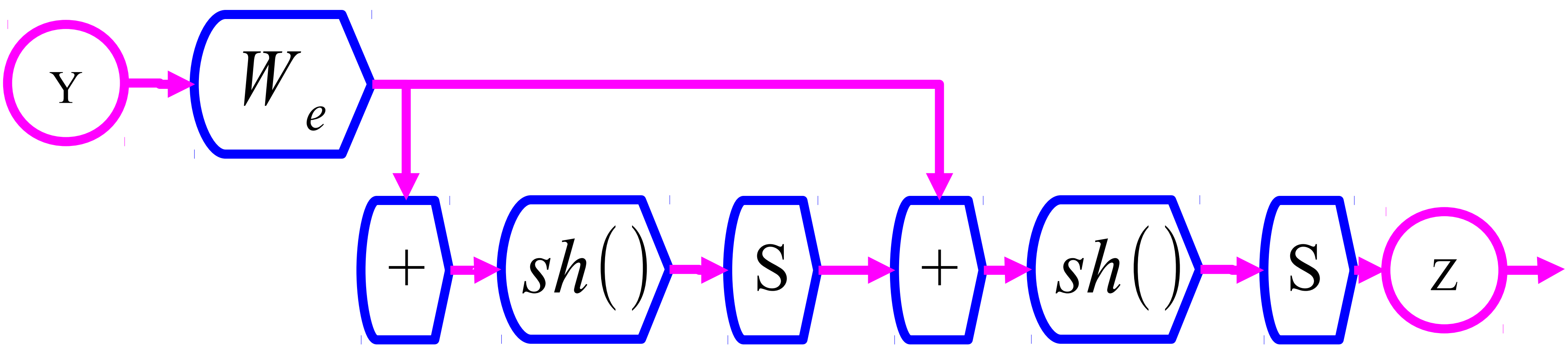
Y LeCun

LISTA: Train We and S matrices to give a good approximation quickly

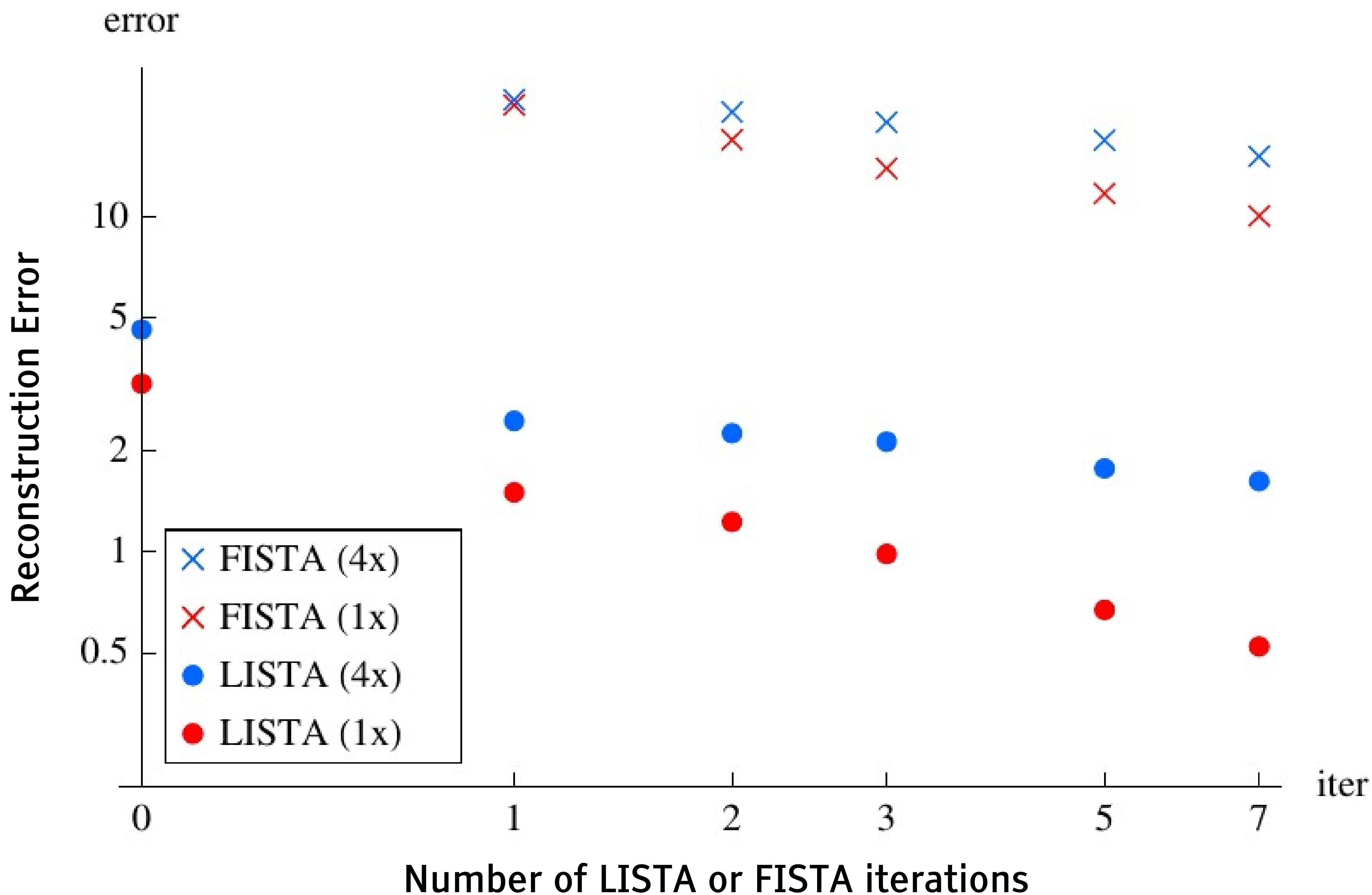
- Think of the FISTA flow graph as a recurrent neural net where We and S are trainable parameters



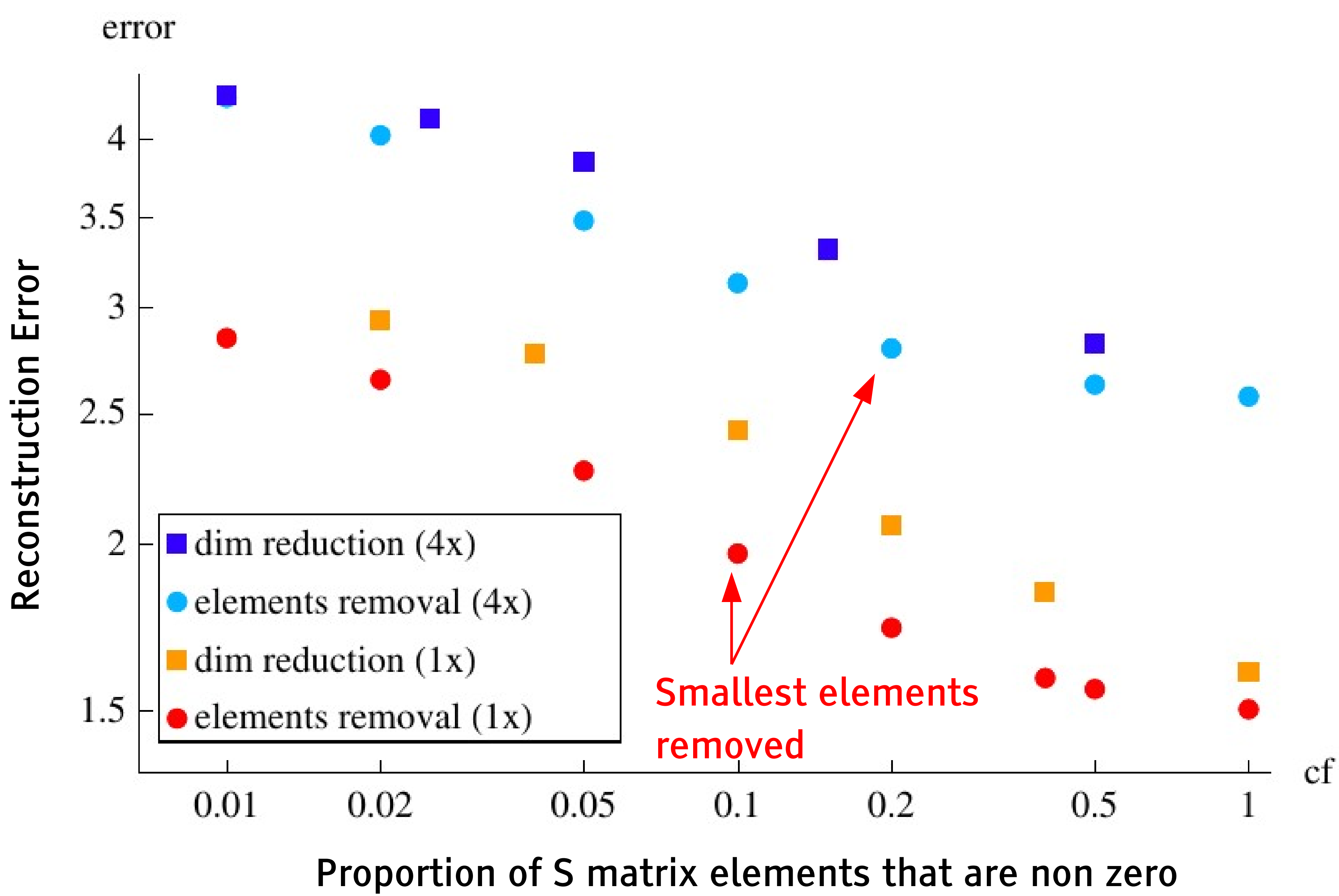
- Time-Unfold the flow graph for K iterations
- Learn the We and S matrices with “backprop-through-time”
- Get the best approximate solution within K iterations



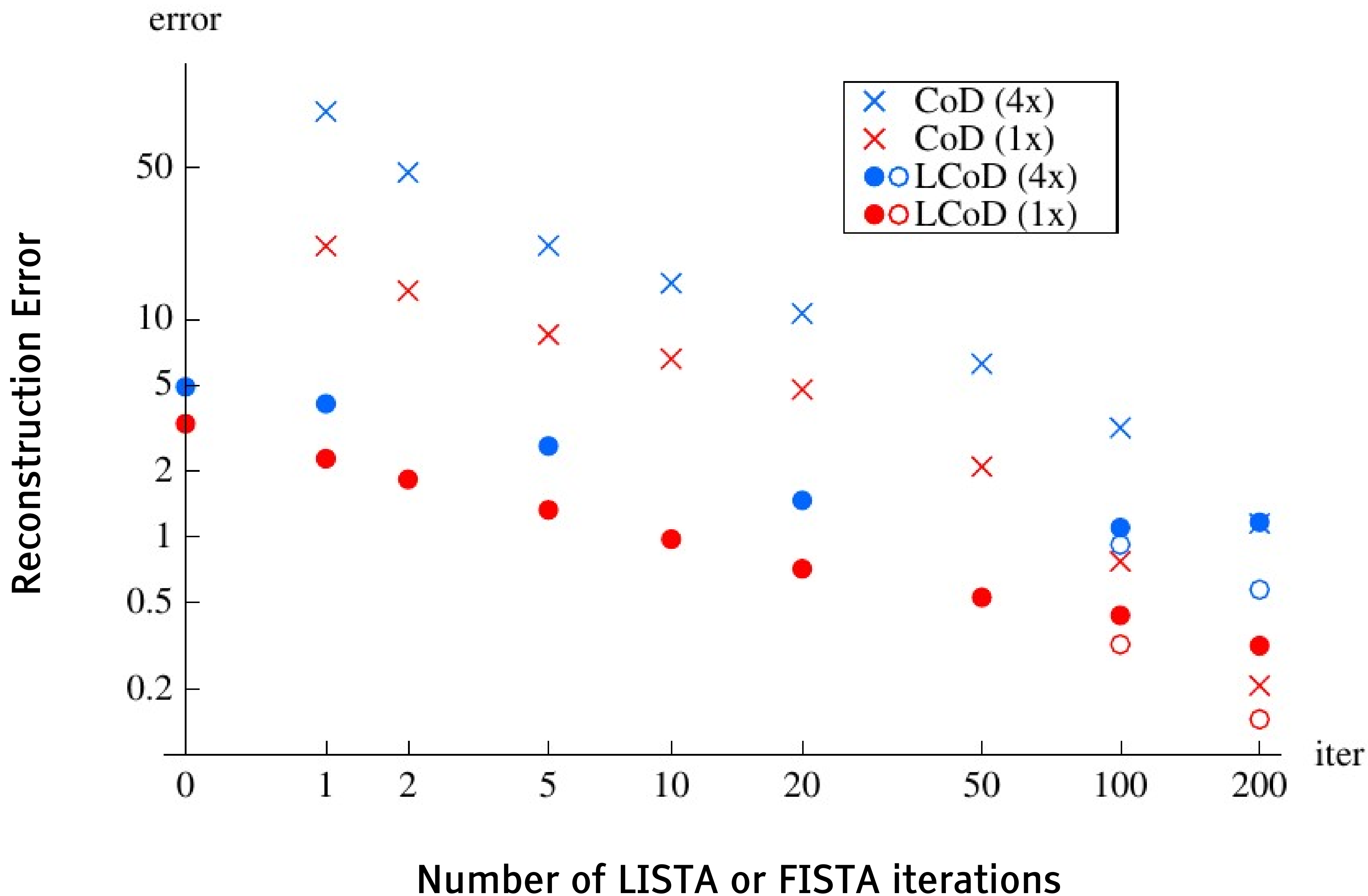
Learning ISTA (LISTA) vs ISTA/FISTA



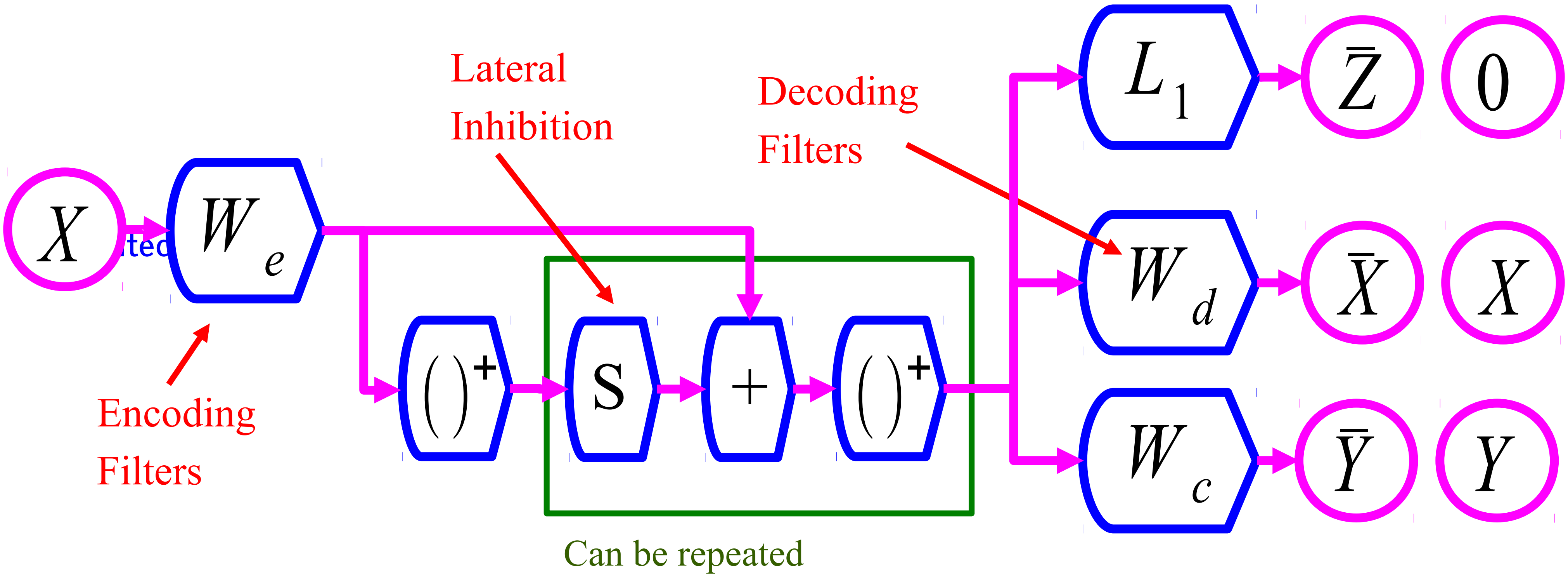
LISTA with partial mutual inhibition matrix



Learning Coordinate Descent (LcoD): faster than LISTA



Discriminative Recurrent Sparse Auto-Encoder (DrSAE)

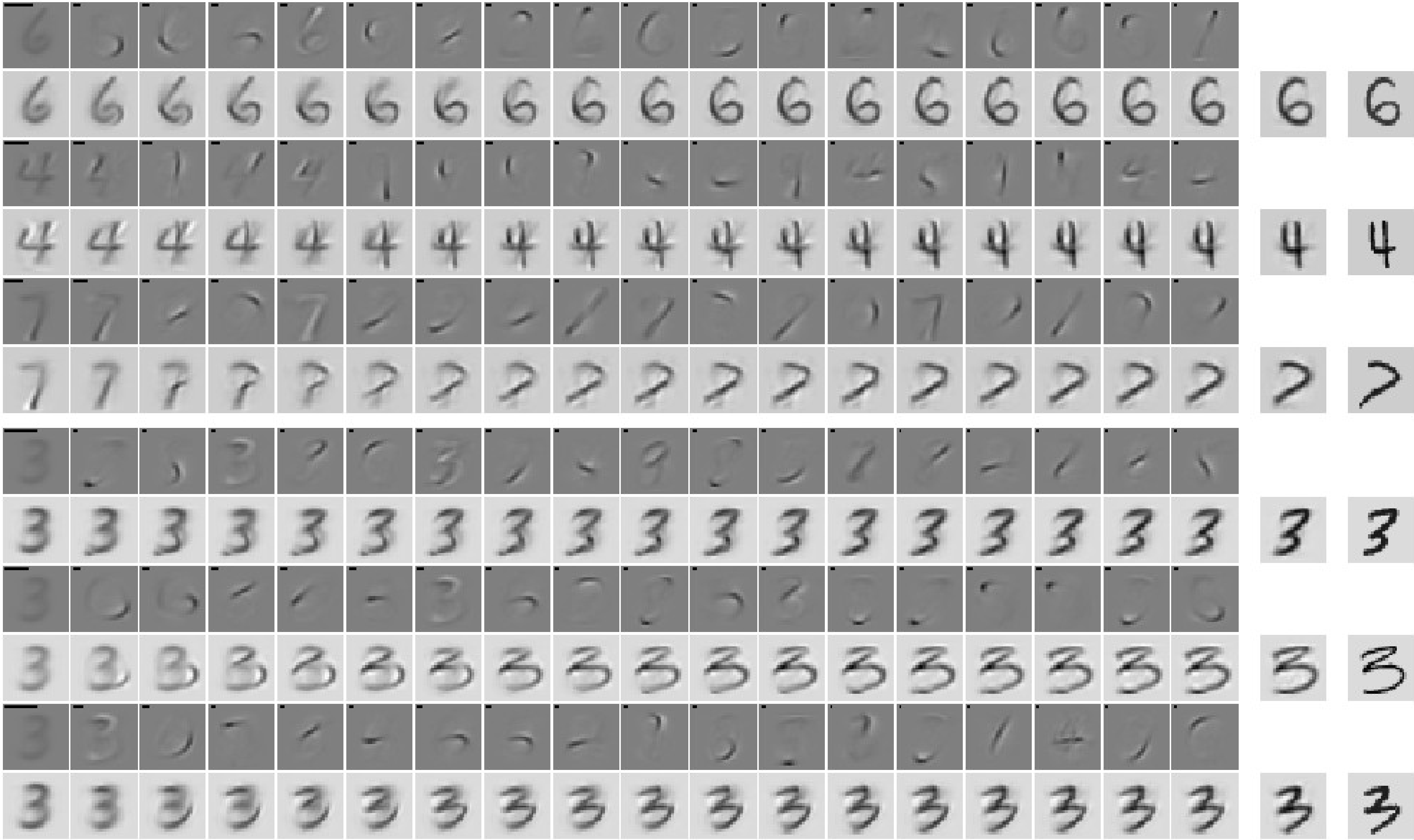


[Rolfe & LeCun ICLR 2013]

- Rectified linear units
- Classification loss: cross-entropy
- Reconstruction loss: squared error
- Sparsity penalty: L1 norm of last hidden layer
- Rows of Wd and columns of We constrained in unit sphere

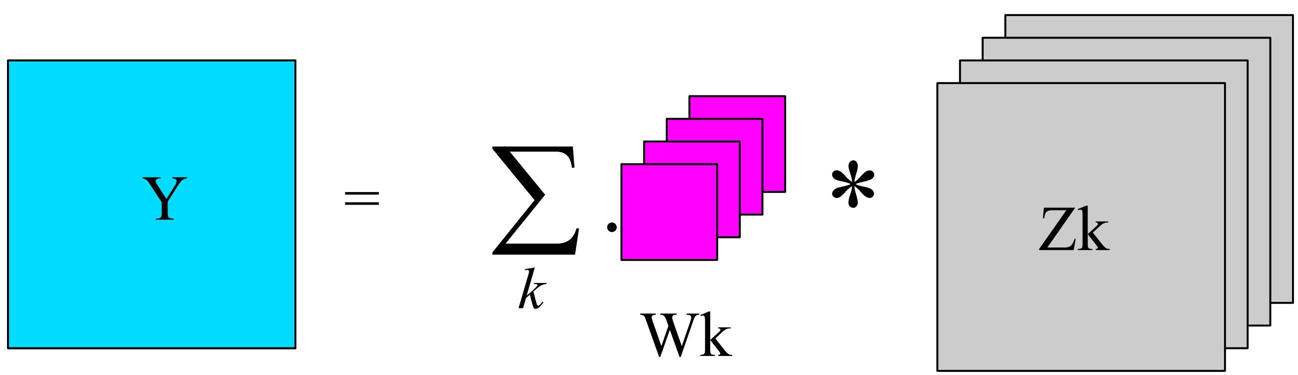
DrSAE Discovers manifold structure of handwritten digits

- Image = prototype + sparse sum of “parts” (to move around the manifold)



Convolutional Sparse Coding

- Replace the dot products with dictionary element by convolutions.
  - Input Y is a full image
  - Each code component Zk is a feature map (an image)
  - Each dictionary element is a convolution kernel
- Regular sparse coding $$E(Y,Z) = ||Y - \sum_k W_k Z_k||^2 + \alpha \sum_k |Z_k|$$
- Convolutional S.C. $$E(Y,Z) = ||Y - \sum_k W_k * Z_k||^2 + \alpha \sum_k |Z_k|$$

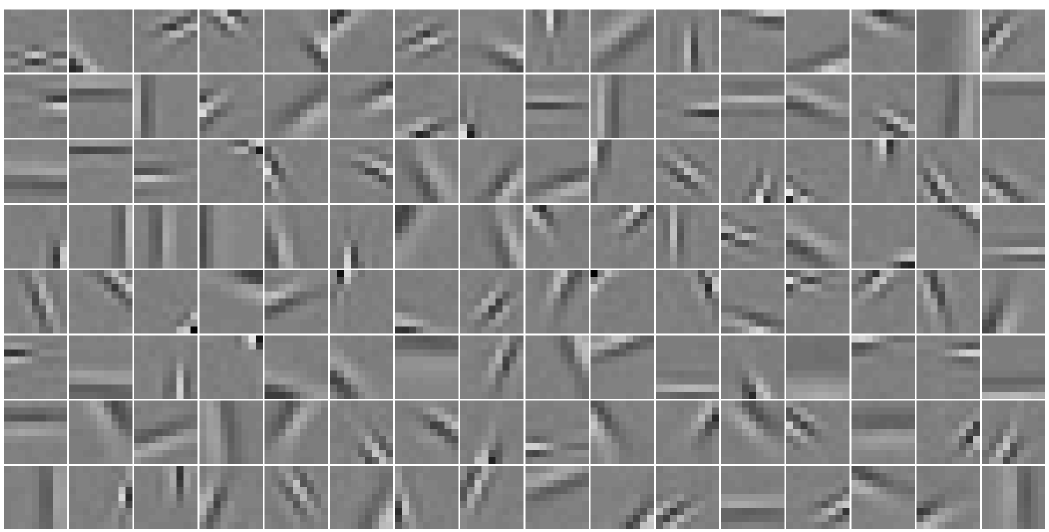


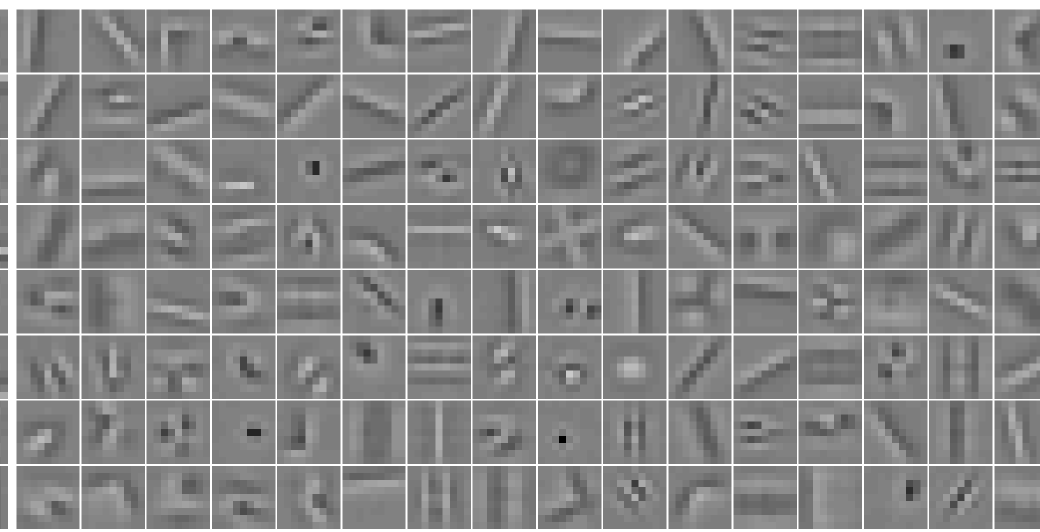
“deconvolutional networks” [Zeiler, Taylor, Fergus CVPR 2010]

Convolutional PSD: Encoder with a soft sh() Function

- **Convolutional Formulation**
  - Extend sparse coding from *PATCH* to *IMAGE*

$$\mathcal{L}(x,z,\mathcal{D}) = \frac{1}{2}||x - \sum_{k=1}^{K} \mathcal{D}_k * z_k||_2^2 + \sum_{k=1}^{K} ||z_k - f(W^k * x)||_2^2 + |z|_1$$

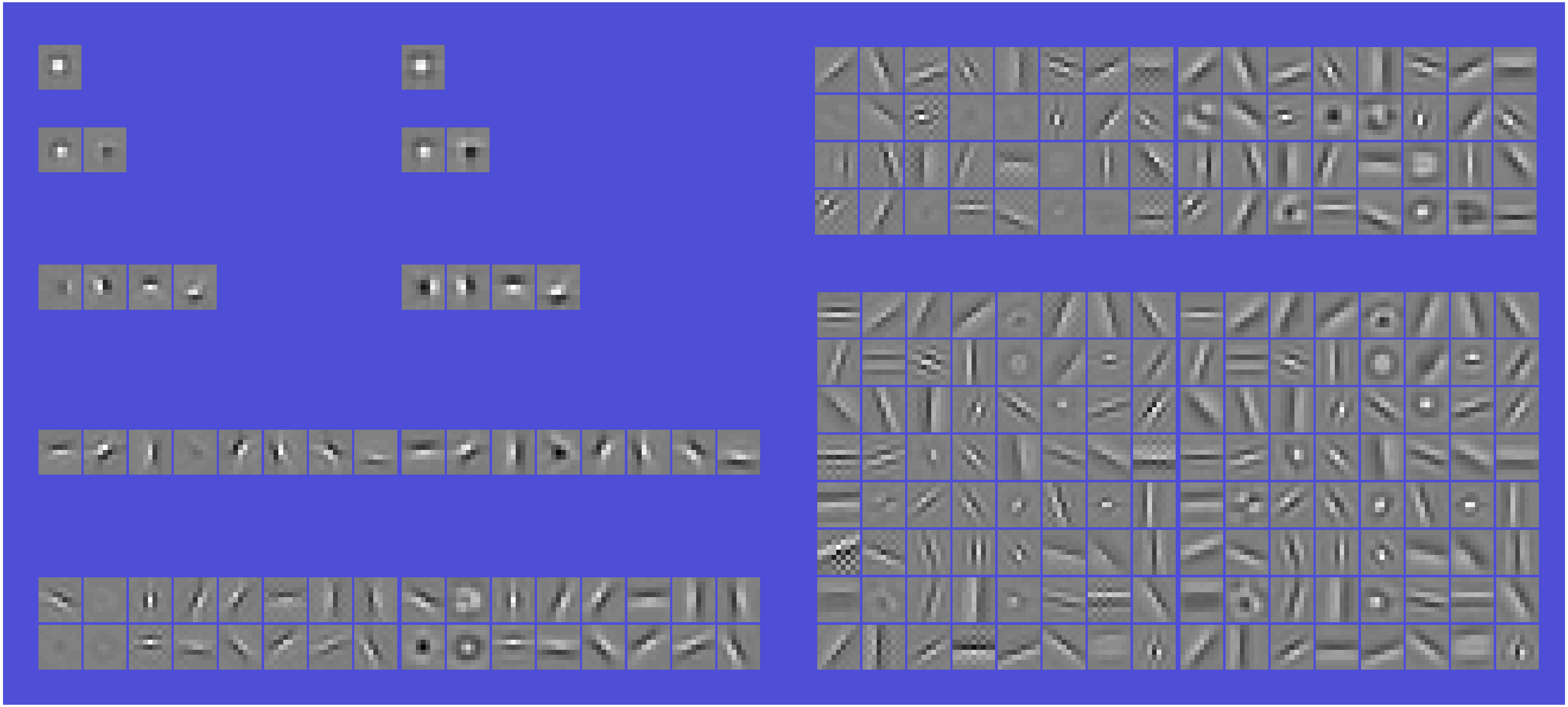




- ***PATCH*** based learning
- ***CONVOLUTIONAL*** learning

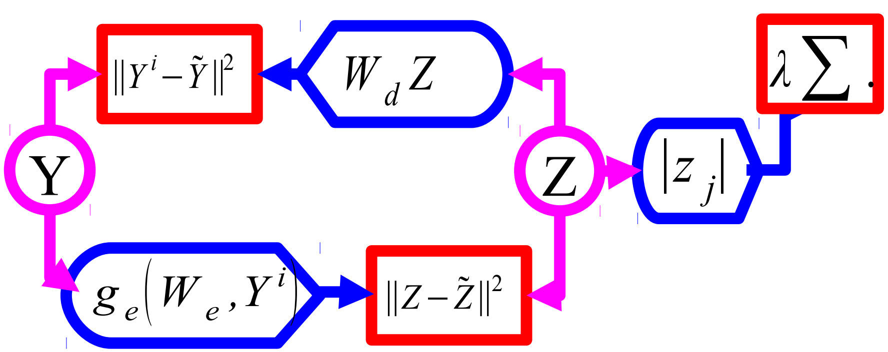
Convolutional Sparse Auto-Encoder on Natural Images

- Filters and Basis Functions obtained with 1, 2, 4, 8, 16, 32, and 64 filters.



Using PSD to Train a Hierarchy of Features

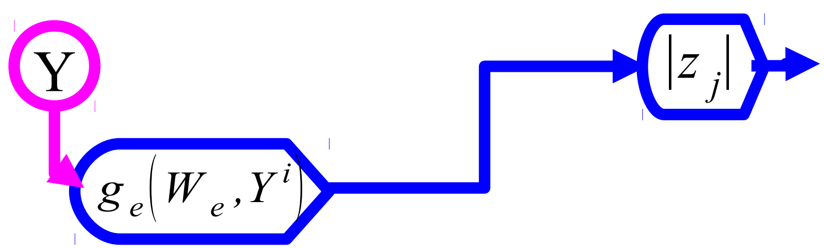
- Phase 1: train first layer using PSD



FEATURES

Using PSD to Train a Hierarchy of Features

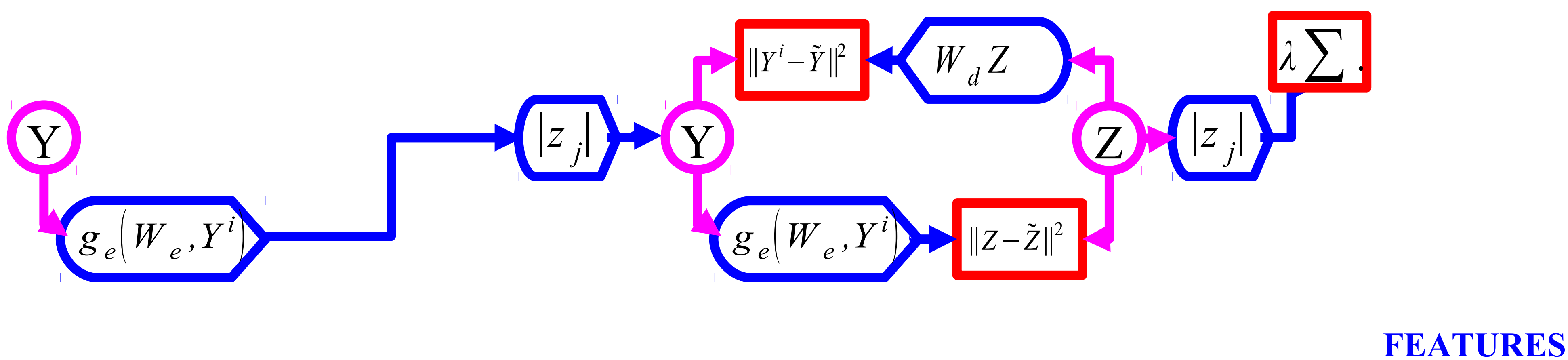
- Phase 1: train first layer using PSD
- Phase 2: use encoder + absolute value as feature extractor



FEATURES

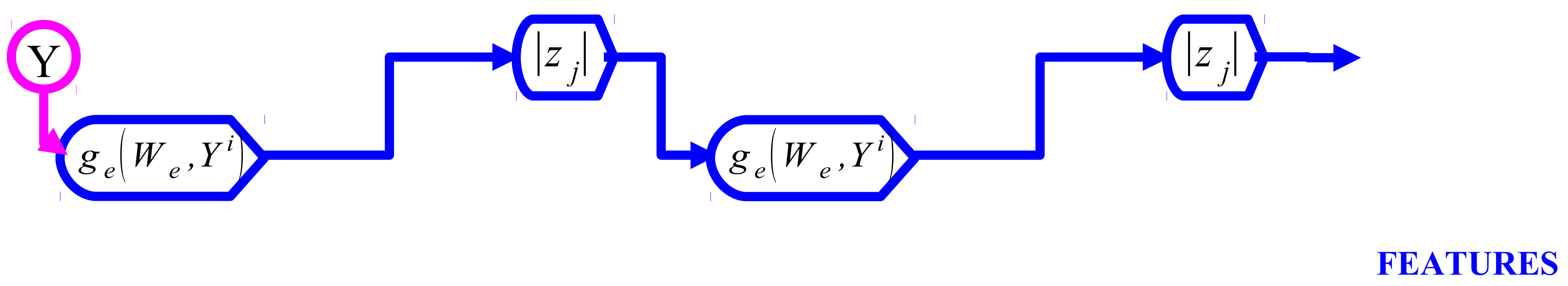
Using PSD to Train a Hierarchy of Features

- Phase 1: train first layer using PSD
- Phase 2: use encoder + absolute value as feature extractor
- Phase 3: train the second layer using PSD



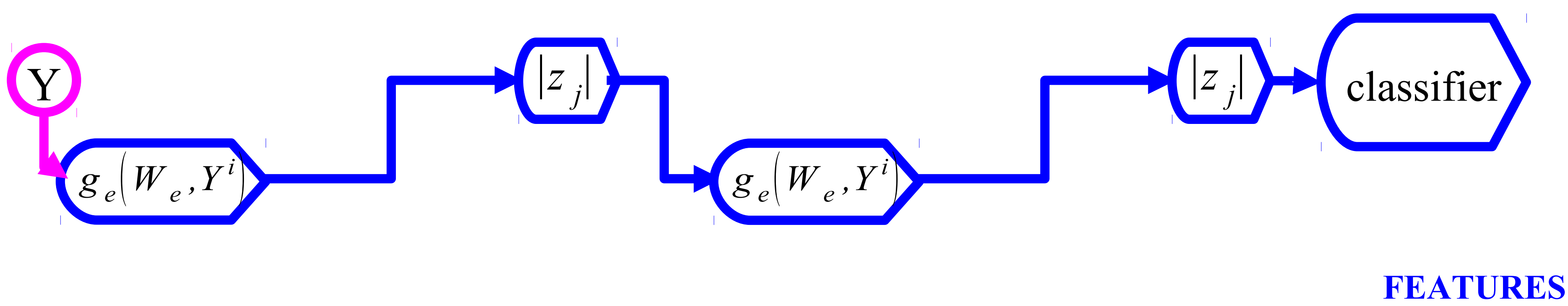
Using PSD to Train a Hierarchy of Features

- Phase 1: train first layer using PSD
- Phase 2: use encoder + absolute value as feature extractor
- Phase 3: train the second layer using PSD
- Phase 4: use encoder + absolute value as 2nd feature extractor

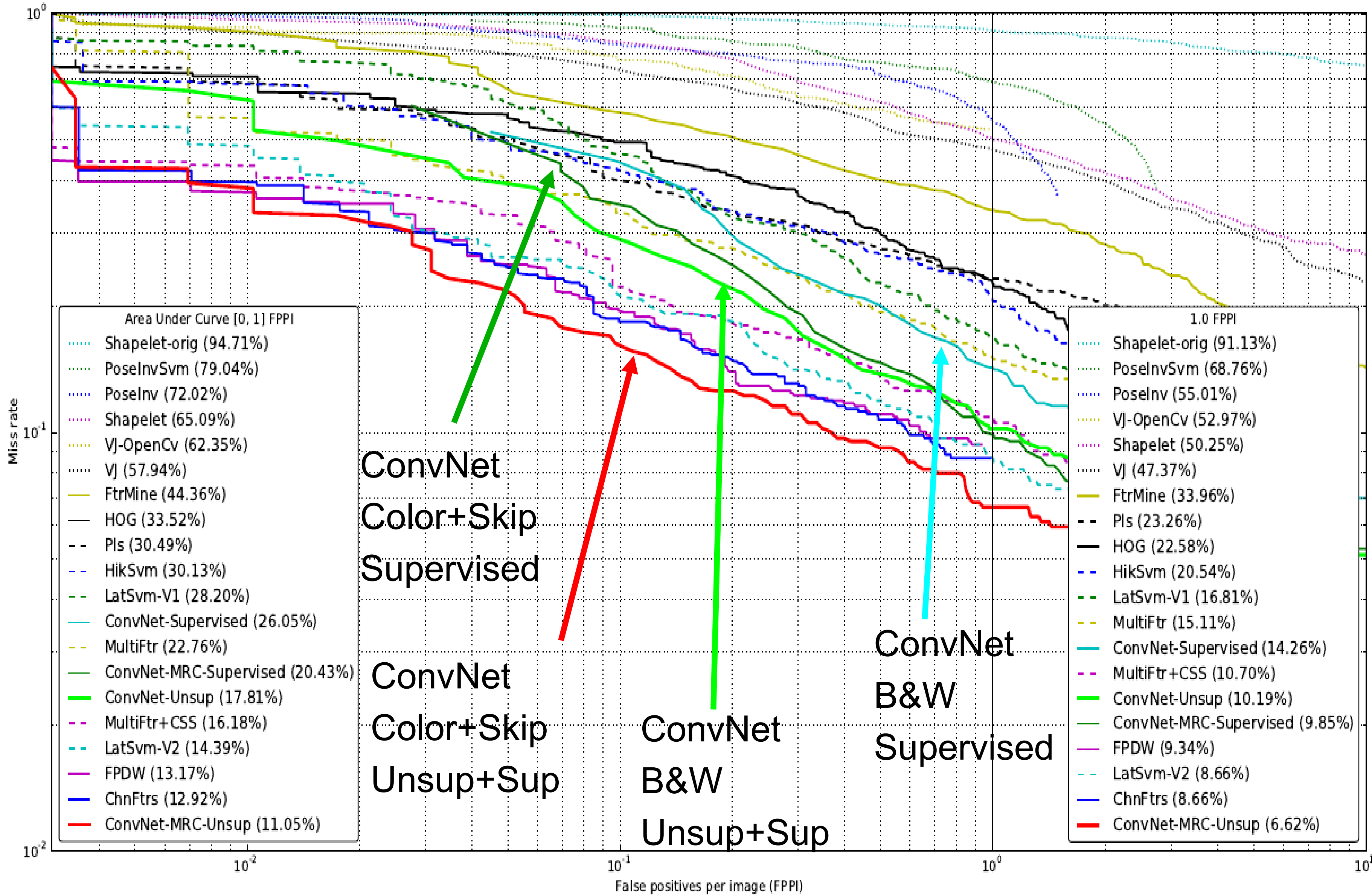


Using PSD to Train a Hierarchy of Features

- Phase 1: train first layer using PSD
- Phase 2: use encoder + absolute value as feature extractor
- Phase 3: train the second layer using PSD
- Phase 4: use encoder + absolute value as 2nd feature extractor
- Phase 5: train a supervised classifier on top
- Phase 6 (optional): train the entire system with supervised back-propagation



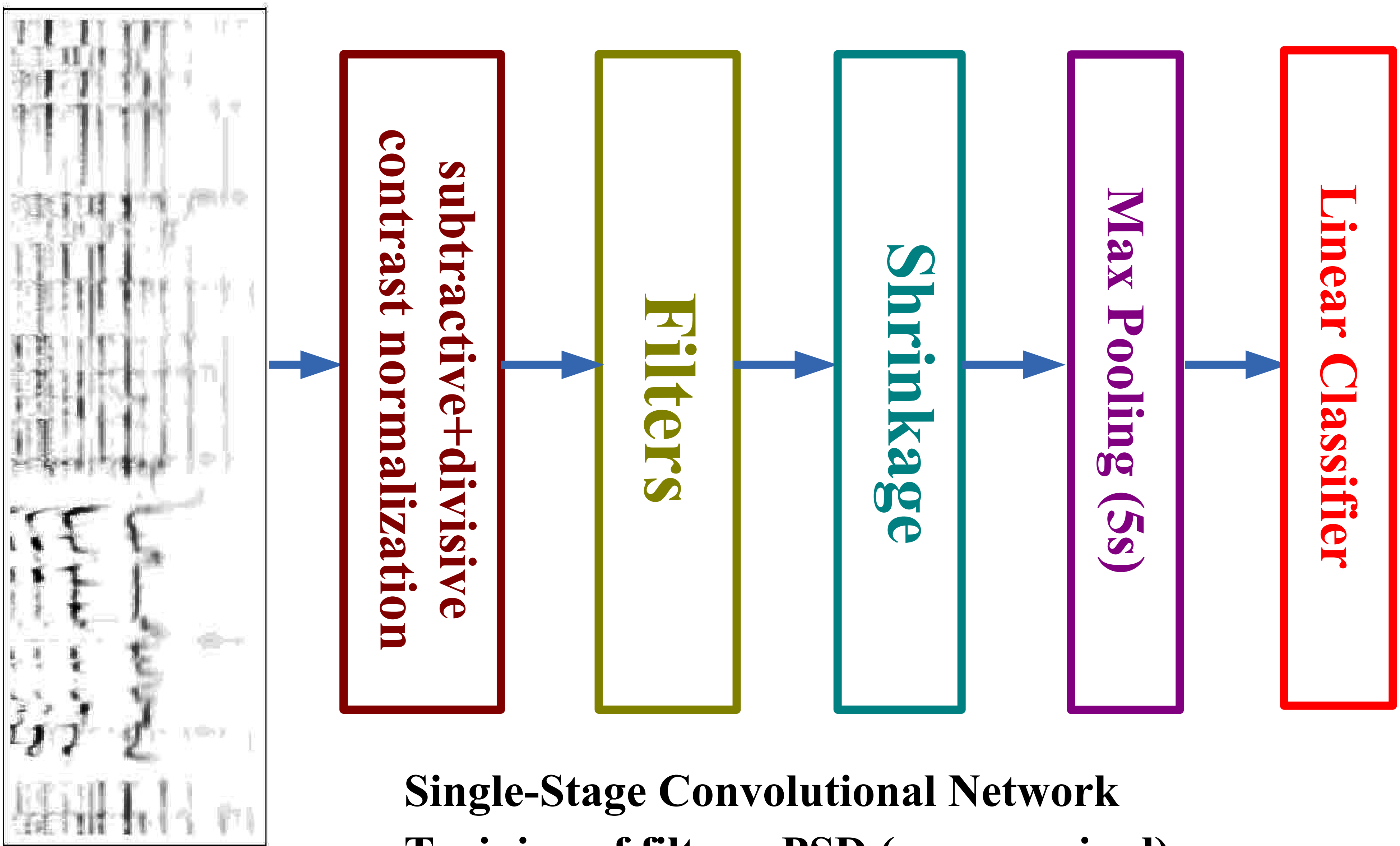
Pedestrian Detection: INRIA Dataset. Miss rate vs false positives



[Kavukcuoglu et al. NIPS 2010] [Sermanet et al. ArXiv 2012]

Musical Genre Classification
With
Unsupervised Convolutional Net

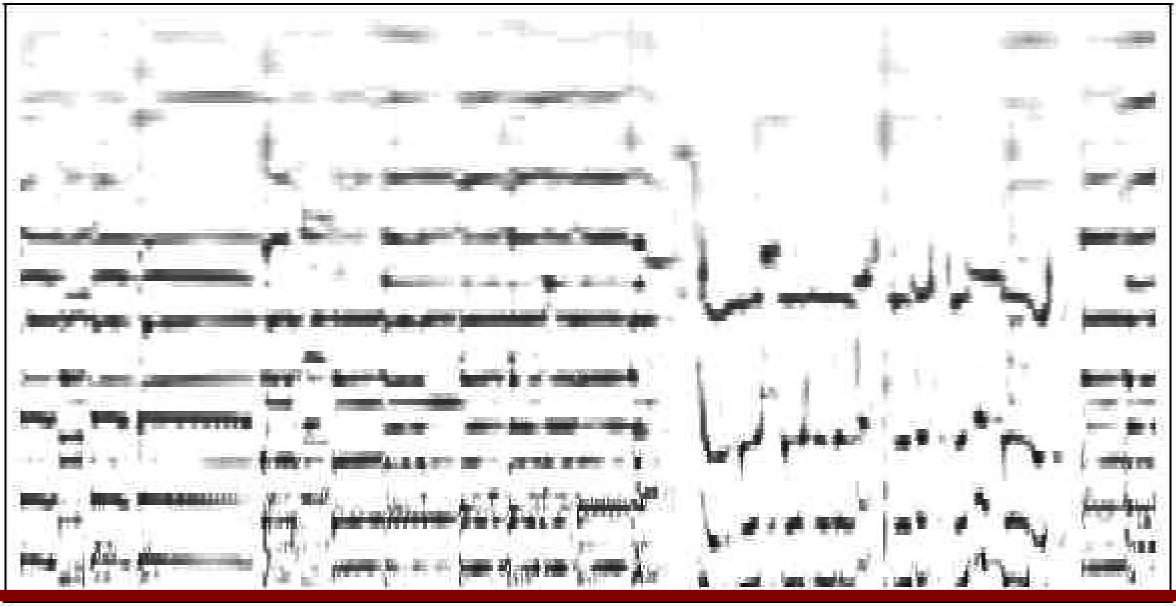
Musical Genre Recognition with PSD Features



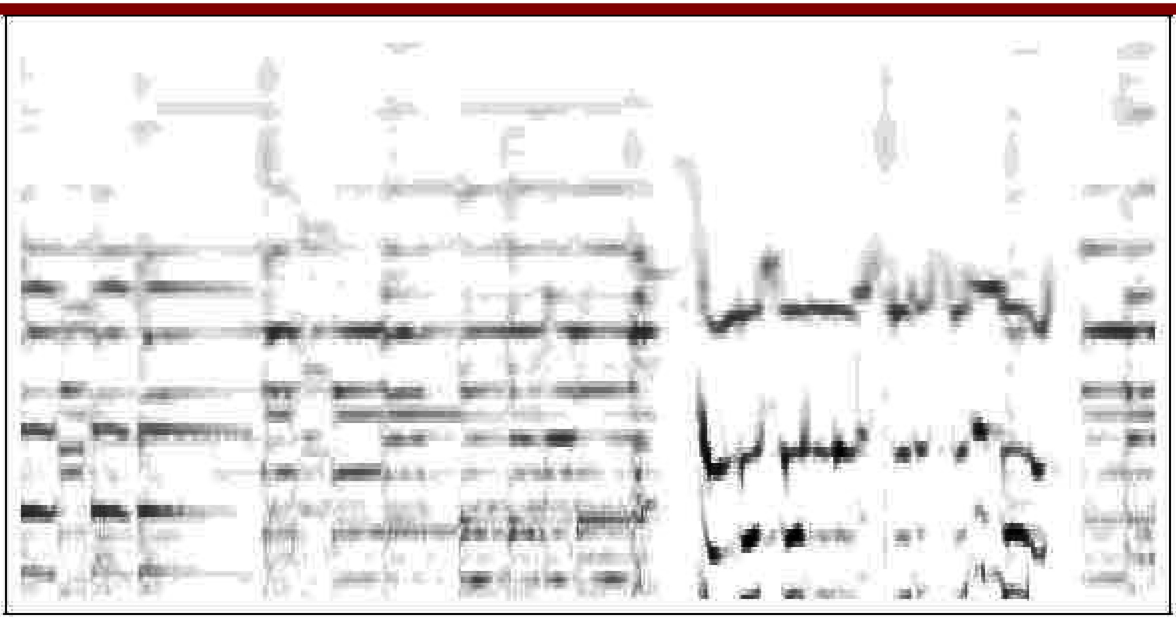
Single-Stage Convolutional Network

Training of filters: PSD (unsupervised)

Constant Q Transform over 46.4 ms → Contrast Normalization

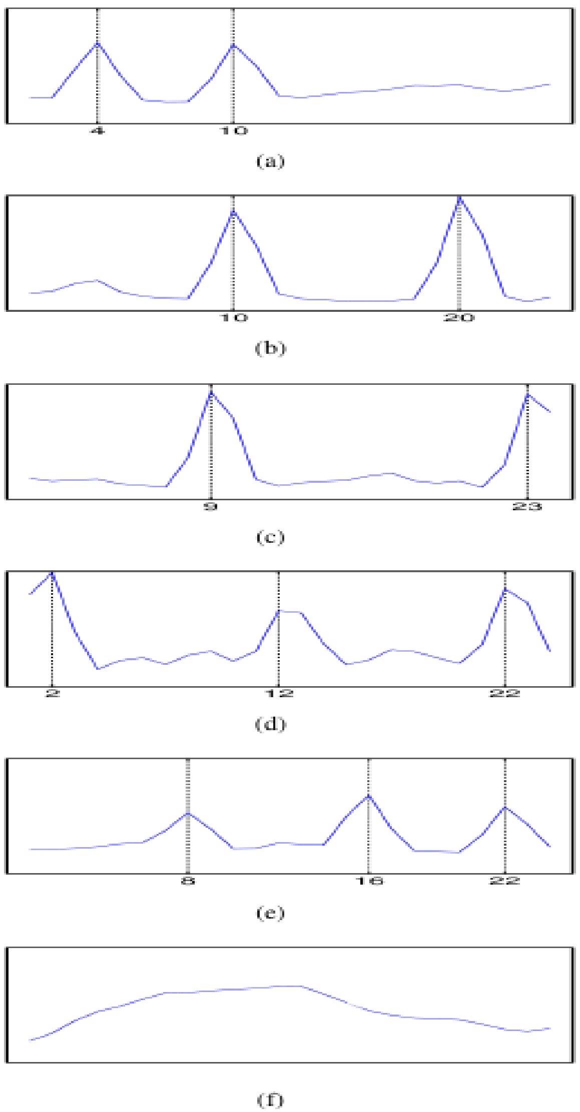


subtractive+divisive contrast normalization



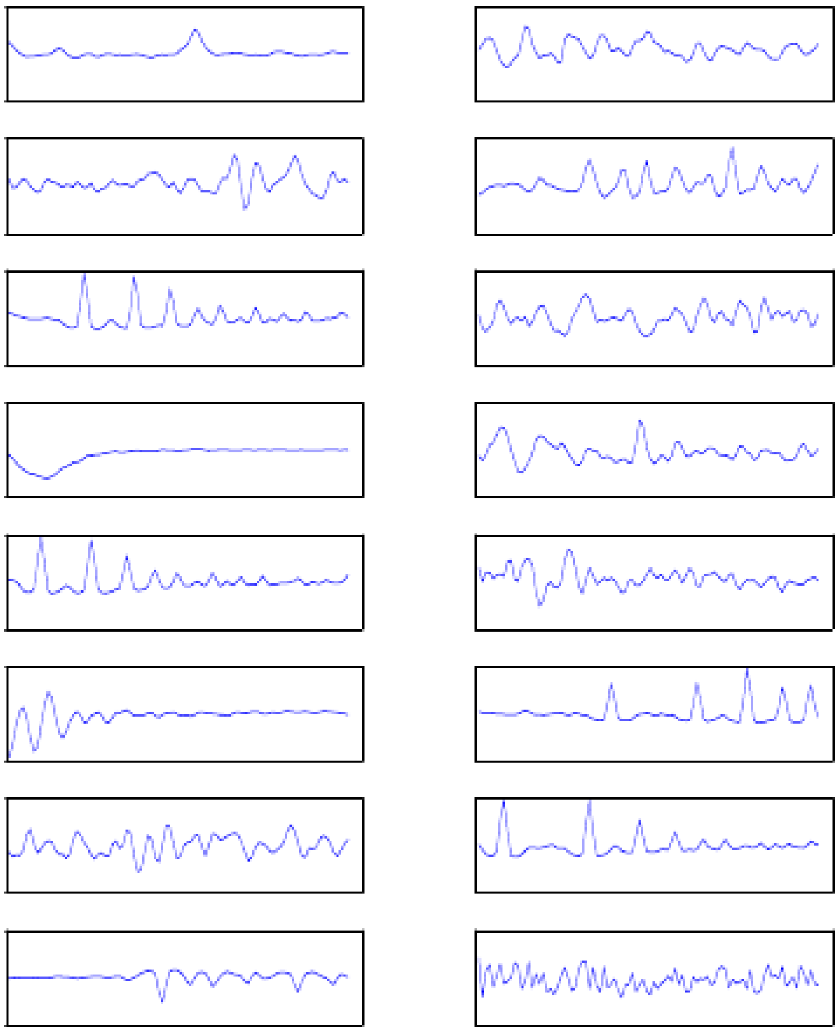
Convolutional PSD Features on Time-Frequency Signals

- Octave-wide features



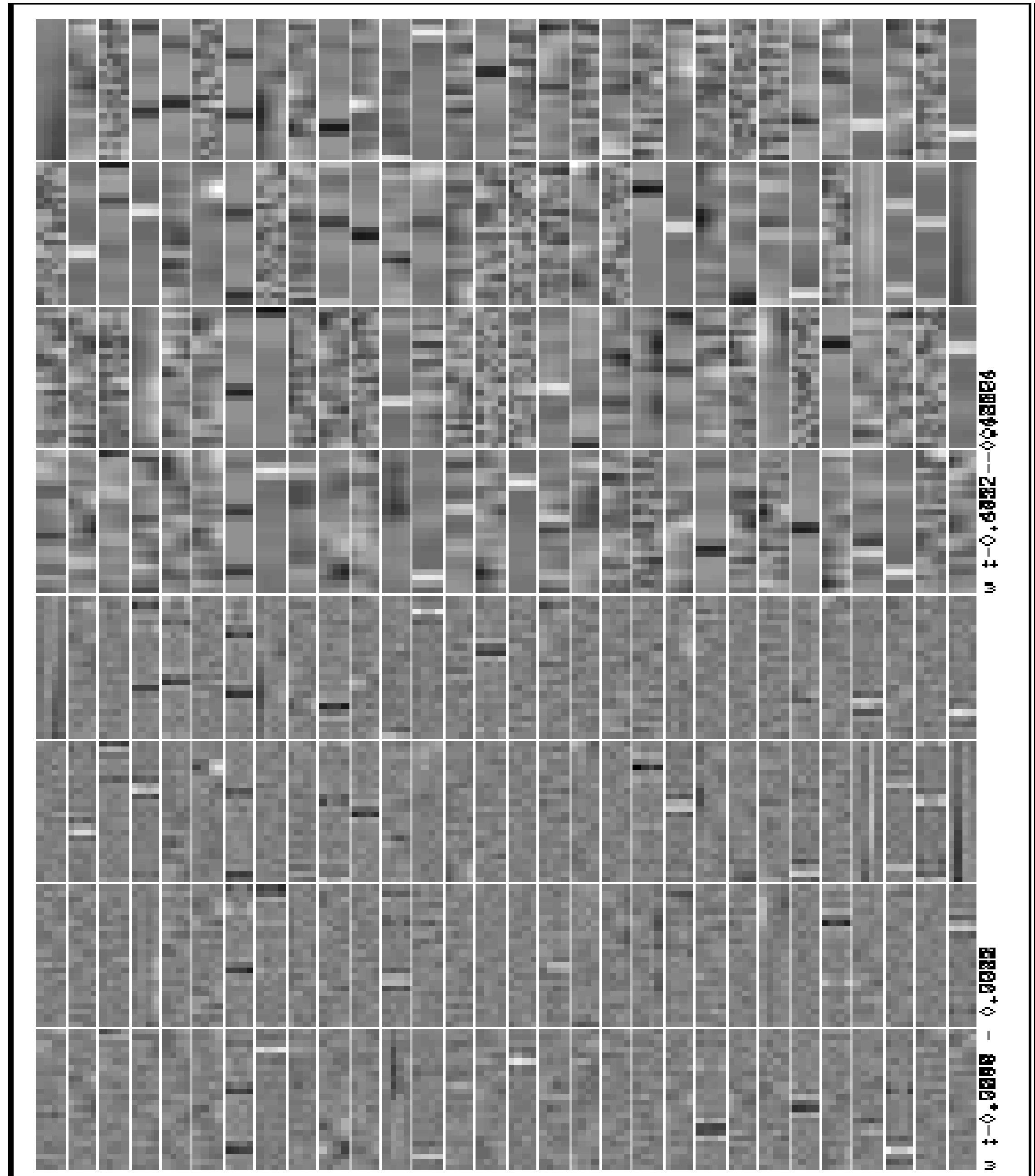
- Minor 3rd
- Perfect 4th
- Perfect 5th
- Quartal chord
- Major triad
- transient

full 4-octave features



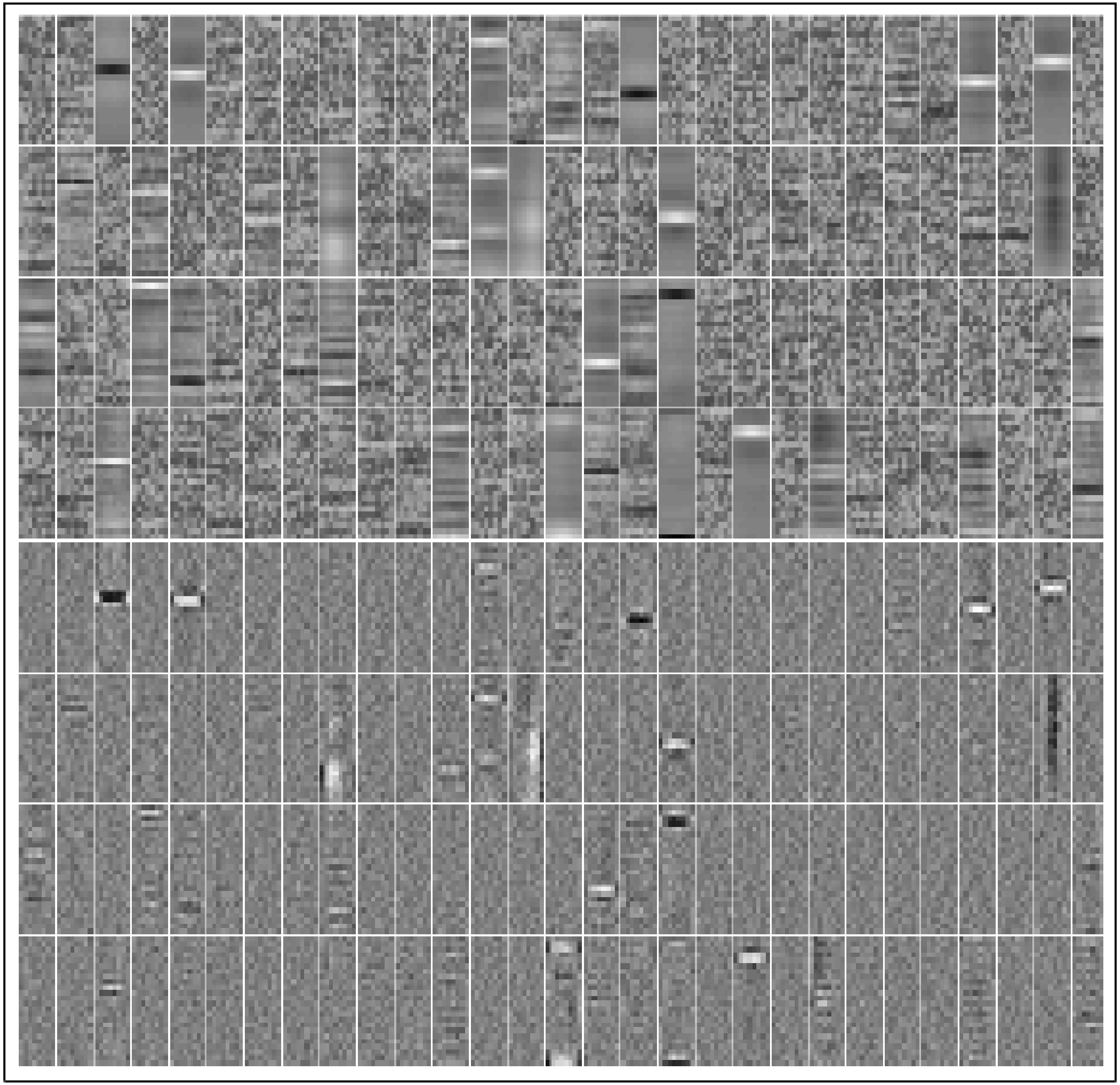
PSD Features on Constant-Q Transform

- Octave-wide features
  - Encoder basis functions
  - Decoder basis functions



Time-Frequency Features

- Octave-wide features on 8 successive acoustic vectors
  - Almost no temporal structure in the filters!



Accuracy on GTZAN dataset (small, old, etc...)

- Accuracy: 83.4%. State of the Art: 84.3%
- Very fast

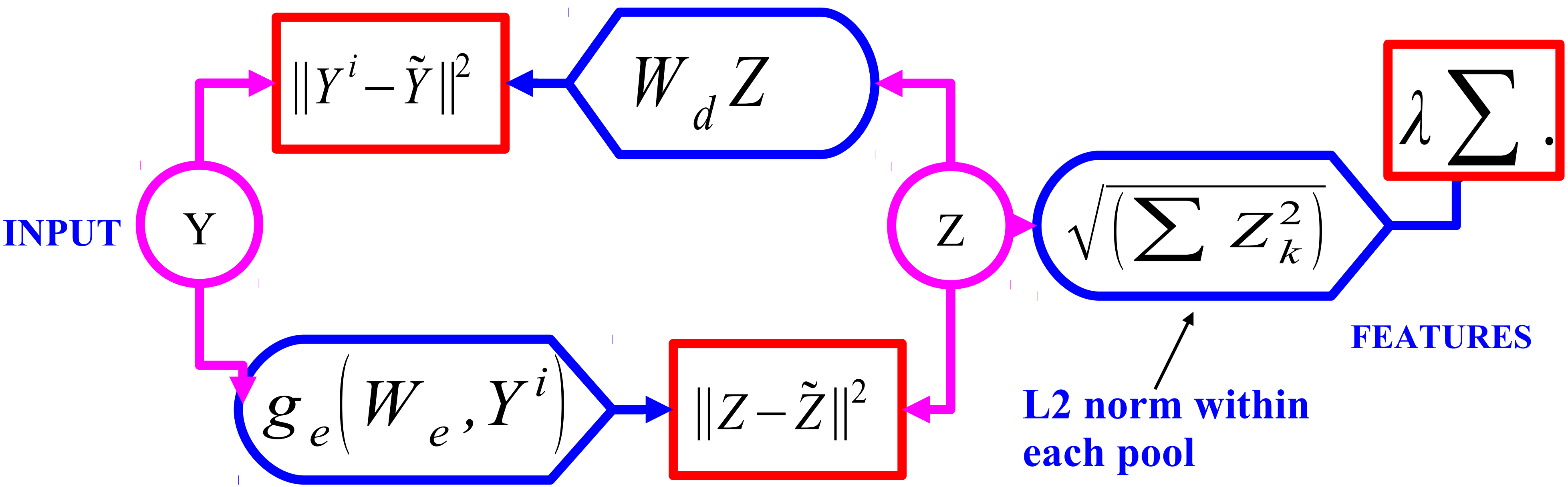
Classifier	Features	Acc. (%)
[redacted]		
RBF-SVM	Learned using DBN [12]	84.3
Linear SVM	Learned using PSD on octaves	**83.4 ± 3.1**
AdaBoost	Many features [2]	83
Linear SVM	Learned using PSD on frames	**79.4 ± 2.8**
SVM	Daubechies Wavelets [19]	78.5
Log. Reg.	Spectral Covariance [3]	77
LDA	MFCC + other [18]	71
Linear SVM	Auditory cortical feat. [25]	70
GMM	MFCC + other [29]	61

Unsupervised Learning: Invariant Features

Learning Invariant Features with L2 Group Sparsity

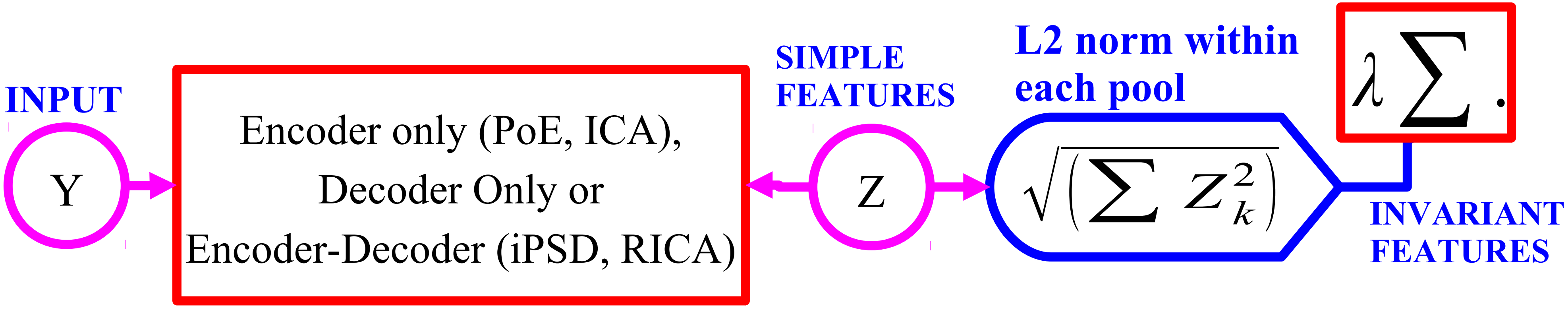
- Unsupervised PSD ignores the spatial pooling step.
- Could we devise a similar method that learns the pooling layer as well?
- Idea [Hyvarinen & Hoyer 2001]: group sparsity on pools of features
  - Minimum number of pools must be non-zero
  - Number of features that are on within a pool doesn't matter
  - Pools tend to regroup similar features

$$E(Y,Z)=\|Y-W_d Z\|^2+\|Z-g_e(W_e,Y)\|^2+\sum_j \sqrt{\sum_{k\in P_j} Z_k^2}$$



Learning Invariant Features with L2 Group Sparsity

- Idea: features are pooled in group.
  - Sparsity: sum over groups of L2 norm of activity in group.
- [Hyvärinen Hoyer 2001]: "subspace ICA"
  - decoder only, square
- [Welling, Hinton, Osindero NIPS 2002]: pooled product of experts
  - encoder only, overcomplete, log student-T penalty on L2 pooling
- [Kavukcuoglu, Ranzato, Fergus LeCun, CVPR 2010]: Invariant PSD
  - encoder-decoder (like PSD), overcomplete, L2 pooling
- [Le et al. NIPS 2011]: Reconstruction ICA
  - Same as [Kavukcuoglu 2010] with linear encoder and tied decoder
- [Gregor & LeCun arXiv:1006:0448, 2010] [Le et al. ICML 2012]
  - Locally-connect non shared (tiled) encoder-decoder



Groups are local in a 2D Topographic Map

- The filters arrange themselves spontaneously so that similar filters enter the same pool.
- The pooling units can be seen as complex cells
- Outputs of pooling units are invariant to local transformations of the input
  - For some it's translations, for others rotations, or other transformations.

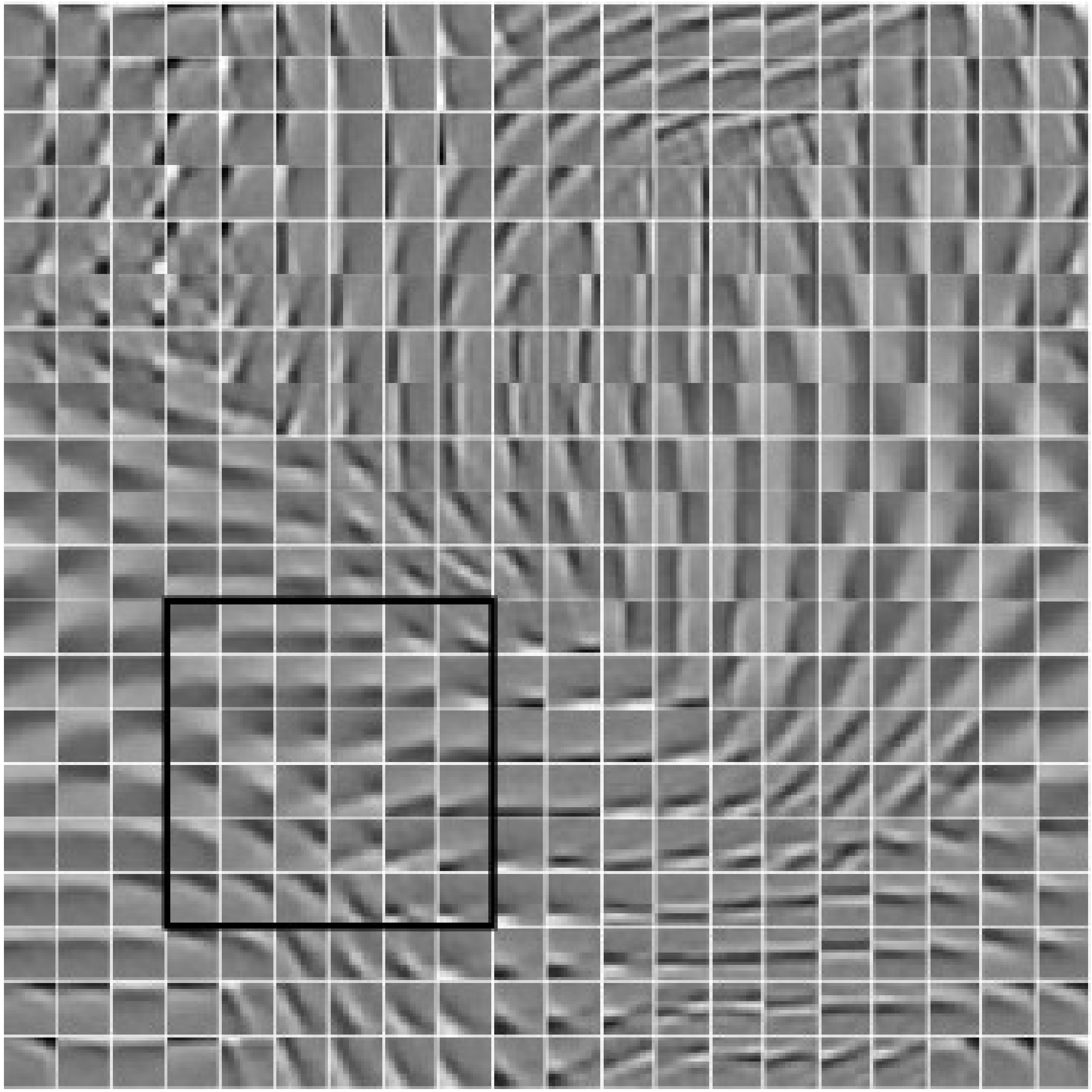


Image-level training, local filters but no weight sharing

- Training on 115x115 images. Kernels are 15x15 (not shared across space!)
  - [Gregor & LeCun 2010]
  - Local receptive fields
  - No shared weights
  - 4x overcomplete
  - L2 pooling
  - Group sparsity over pools

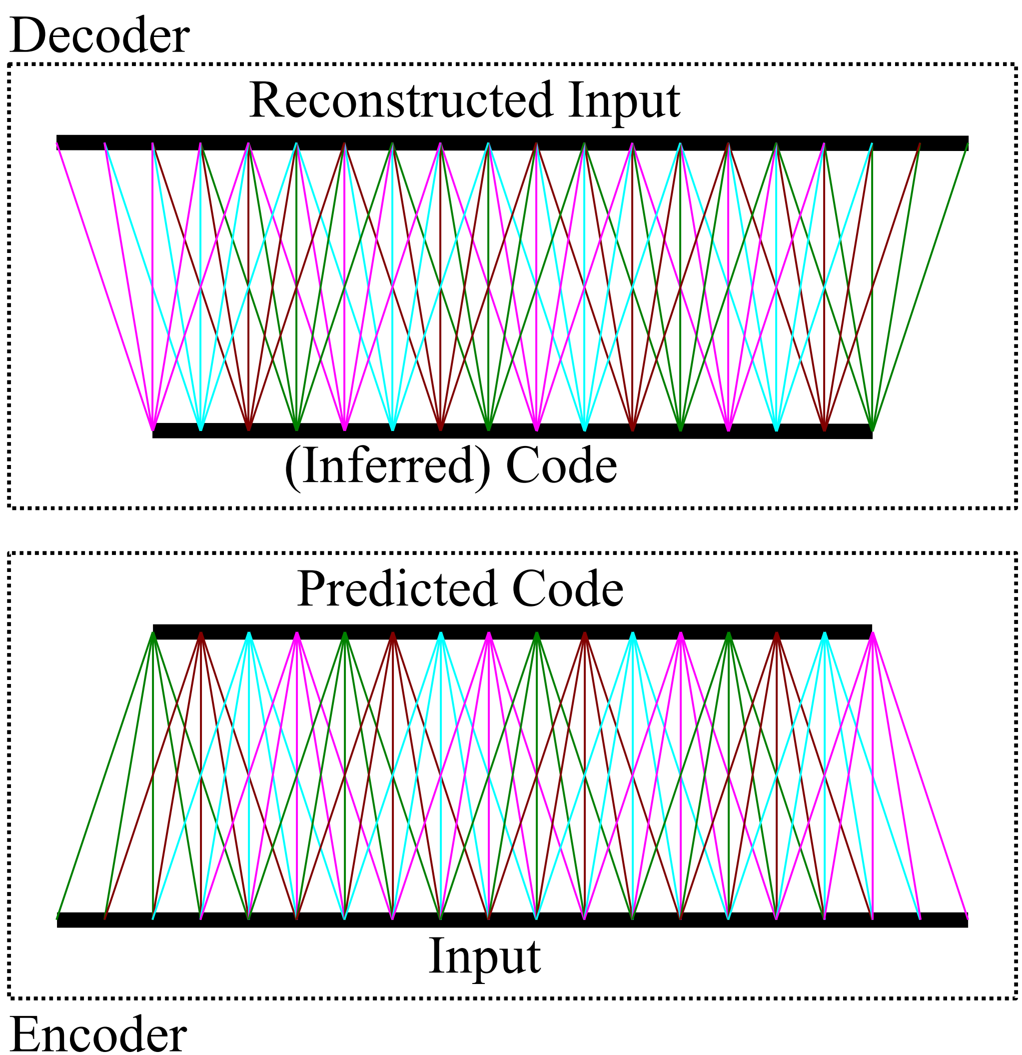
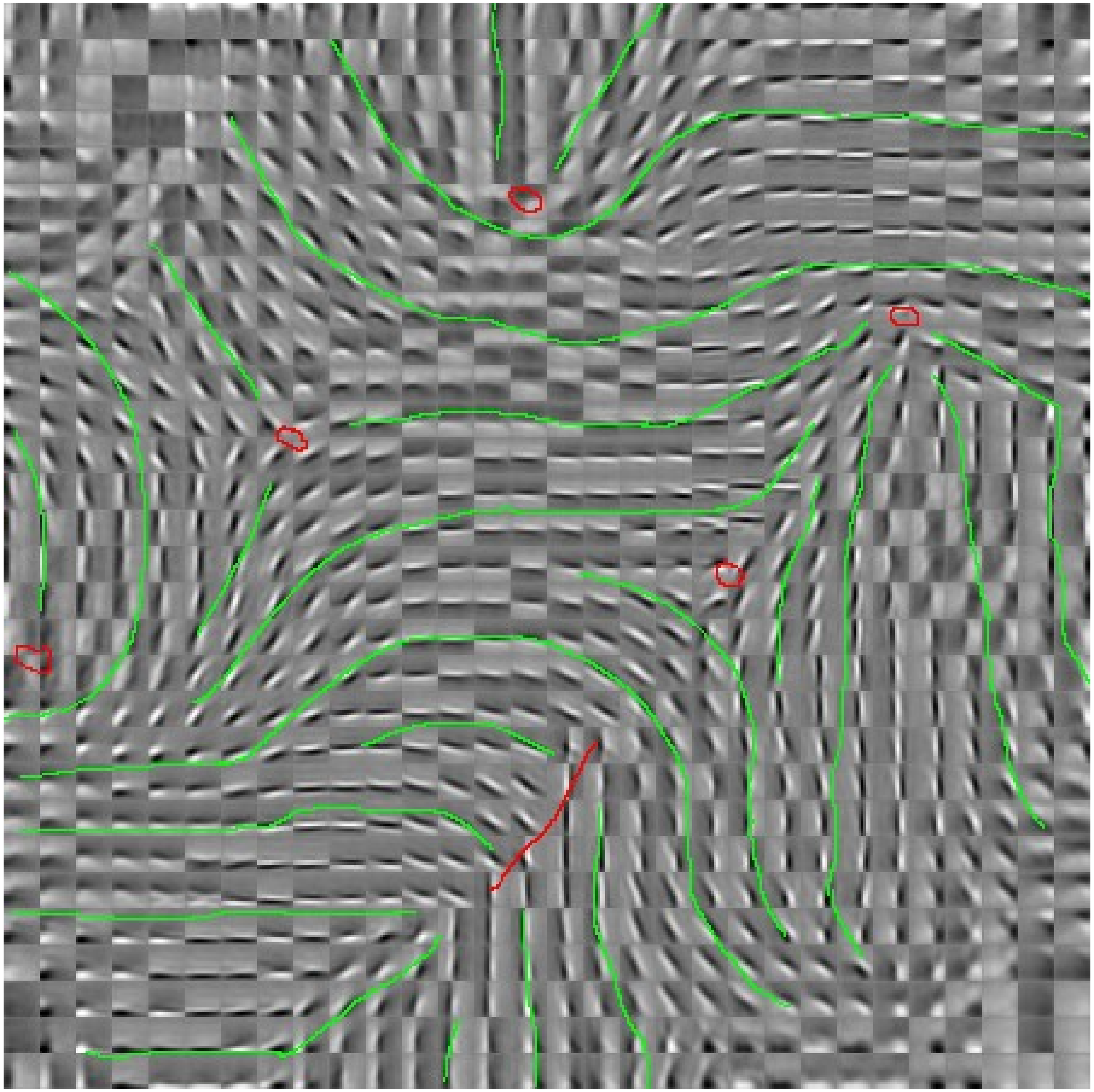


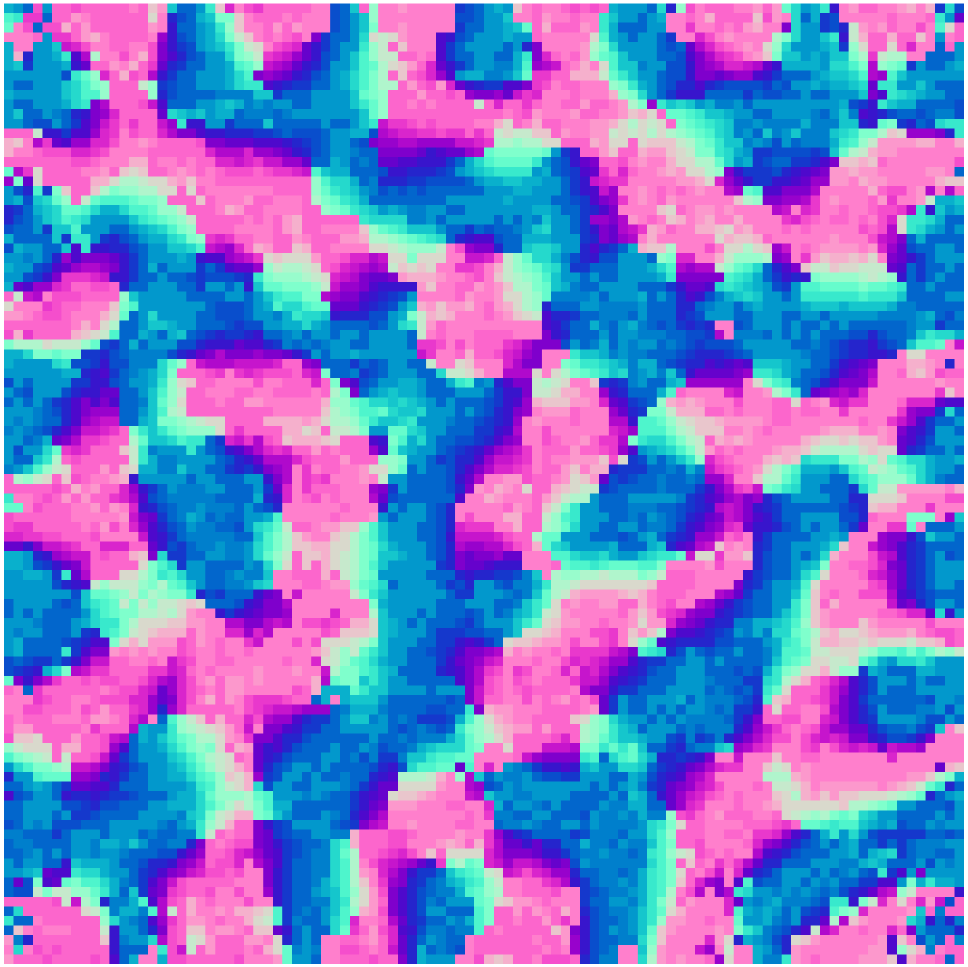
Image-level training, local filters but no weight sharing

- Training on 115x115 images. Kernels are 15x15 (not shared across space!)



Topographic Maps

Obermayer and GG Blasdel, Journal of Neuroscience, Vol 13, 4114-4129 (**Monkey**)

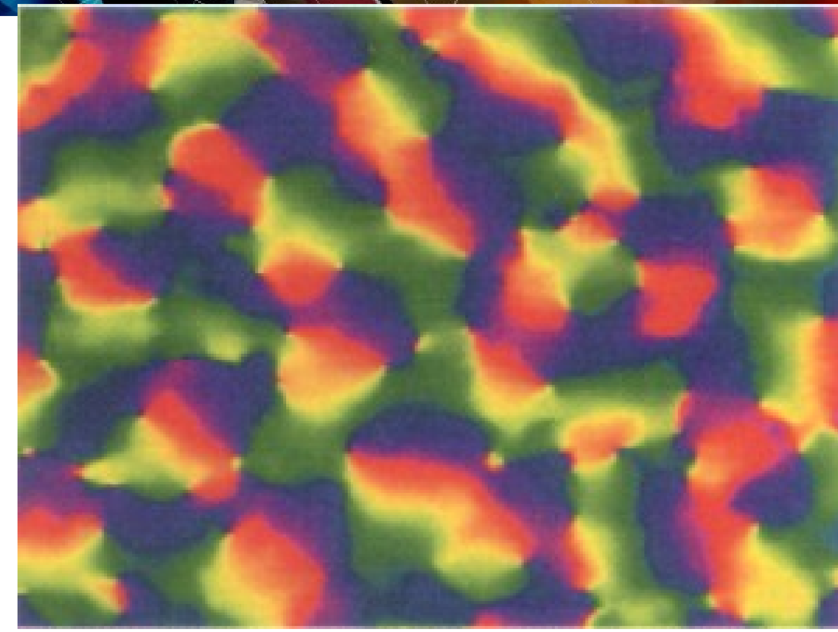


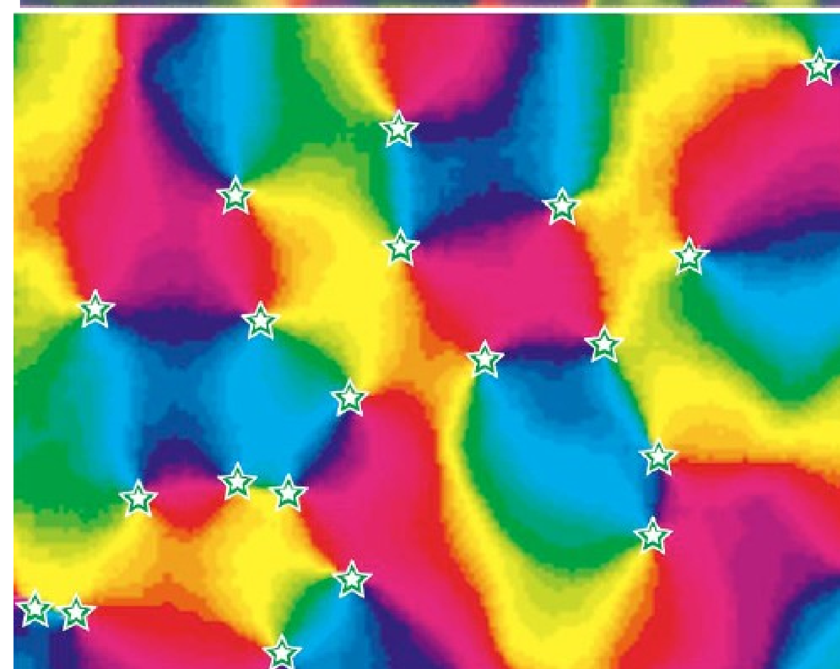
119x119 Image Input

100x100 Code

20x20 Receptive field size

sigma=5

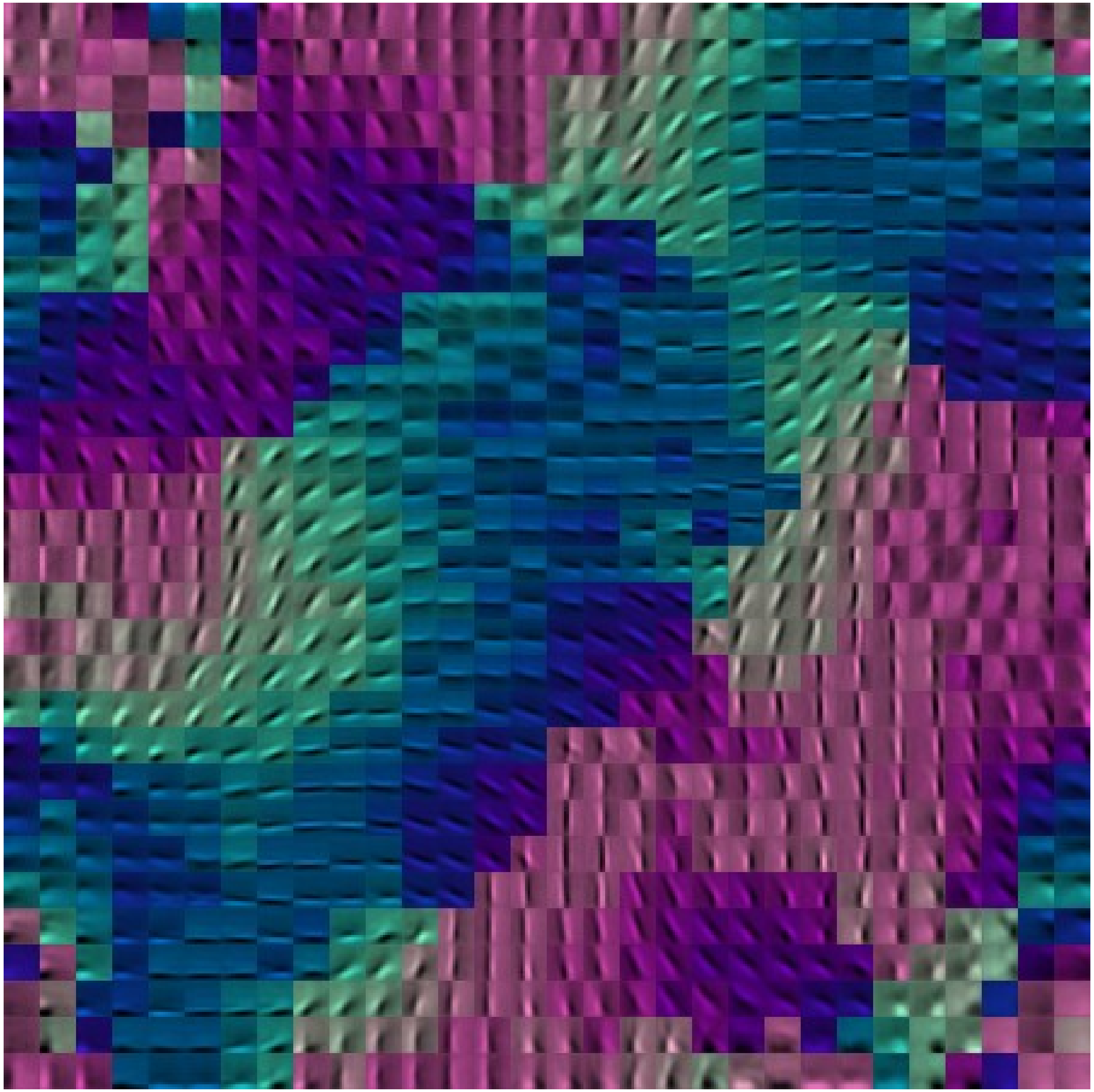




Michael C. Crair, et. al. The Journal of Neurophysiology Vol. 77 No. 6 June 1997, pp. 3381-3385 (**Cat**)

Image-level training, local filters but no weight sharing

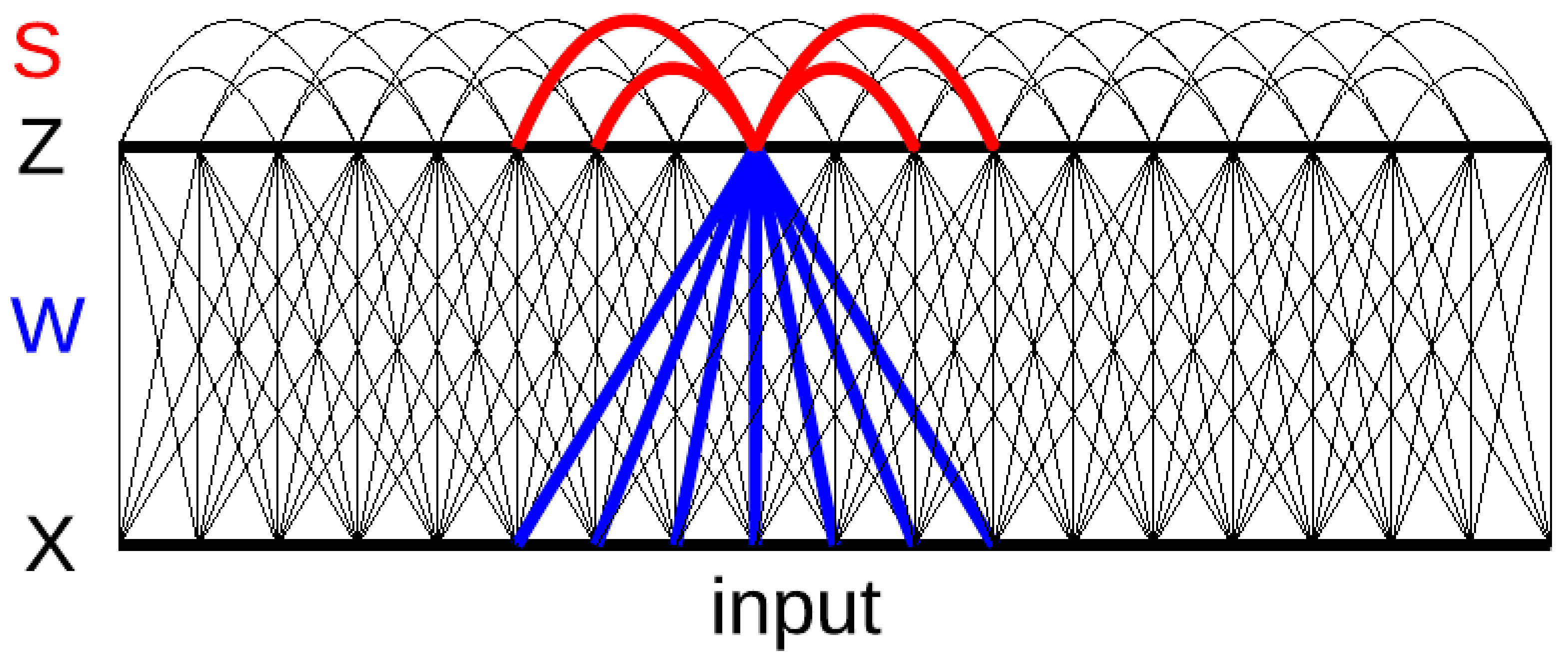
- Color indicates orientation (by fitting Gabors)



Invariant Features Lateral Inhibition

- Replace the L1 sparsity term by a lateral inhibition matrix
- Easy way to impose some structure on the sparsity

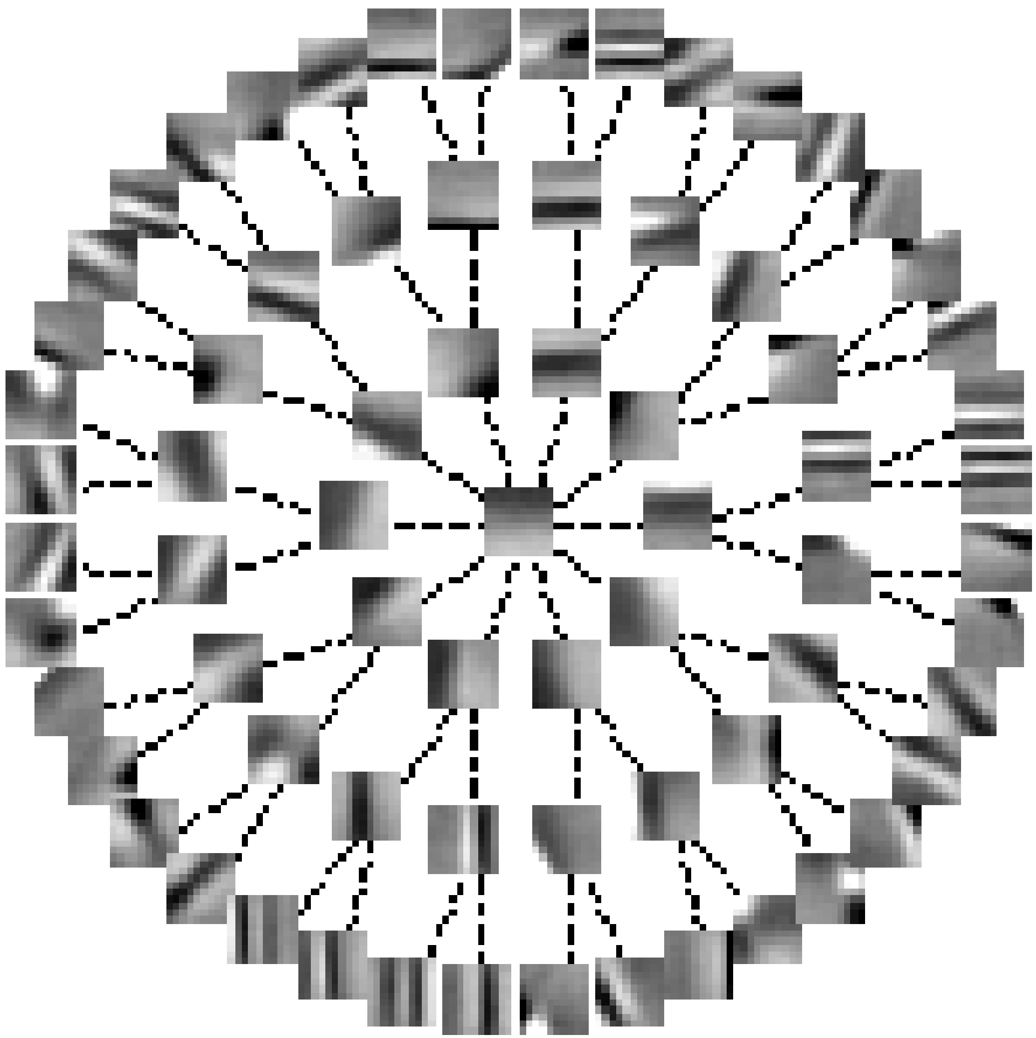
$$\min_{W,Z} \sum_{x \in X} ||Wz - x||^2 + |z|^T S|z|$$



[Gregor, Szlam, LeCun NIPS 2011]

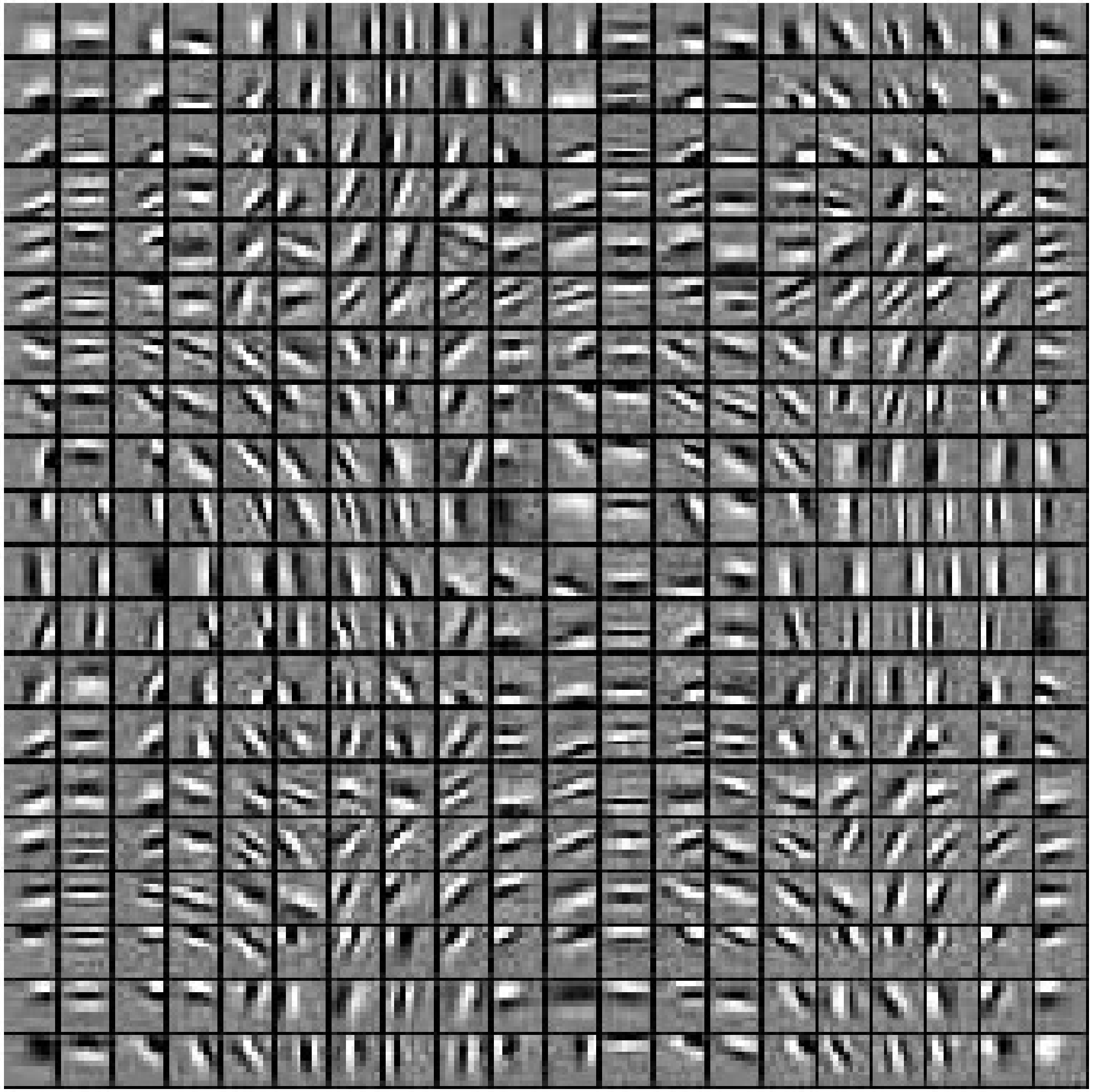
Invariant Features via Lateral Inhibition: Structured Sparsity

- Each edge in the tree indicates a zero in the S matrix (no mutual inhibition)
- Sij is larger if two neurons are far away in the tree



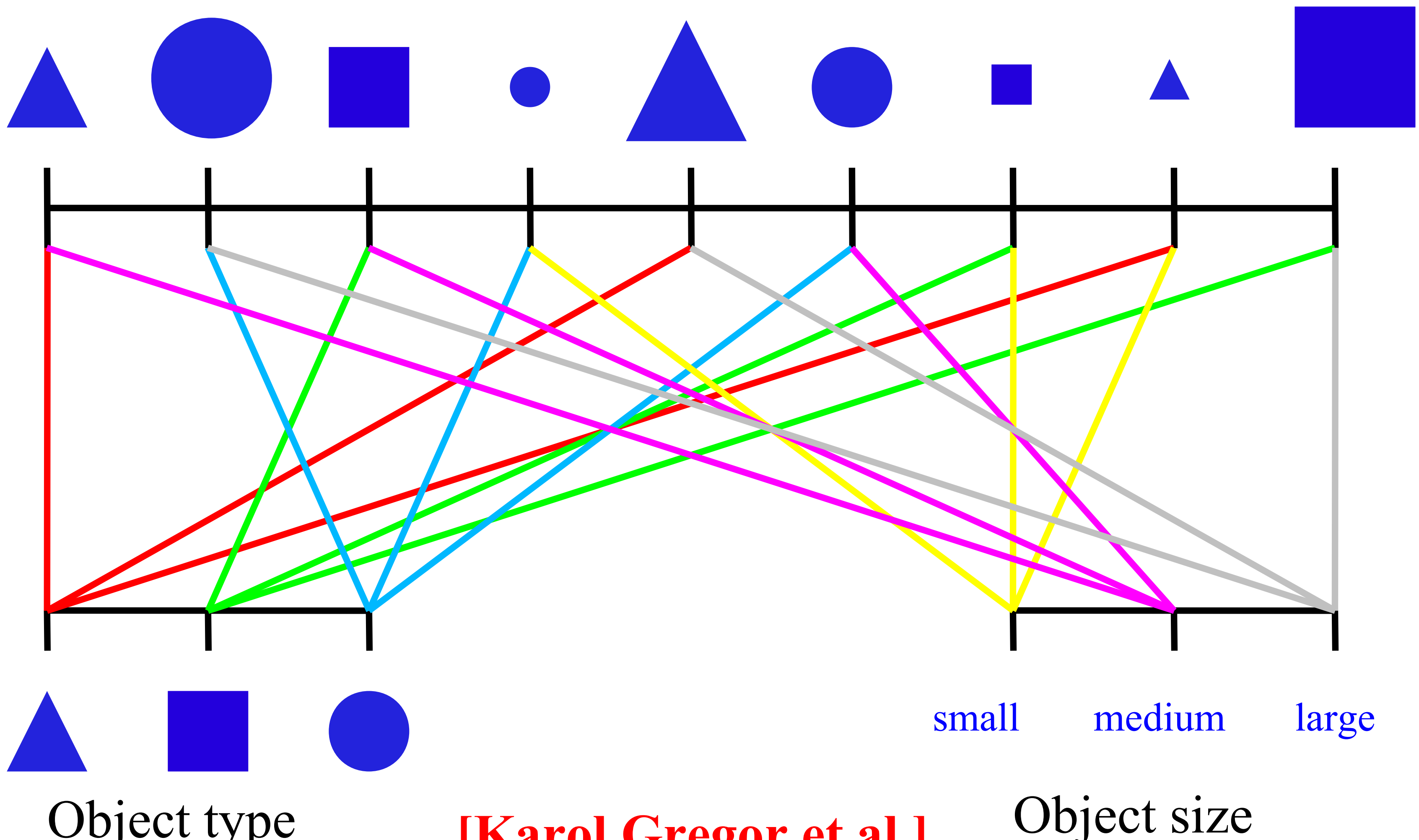
Invariant Features via Lateral Inhibition: Topographic Maps

- Non-zero values in S form a ring in a 2D topology
  - Input patches are high-pass filtered



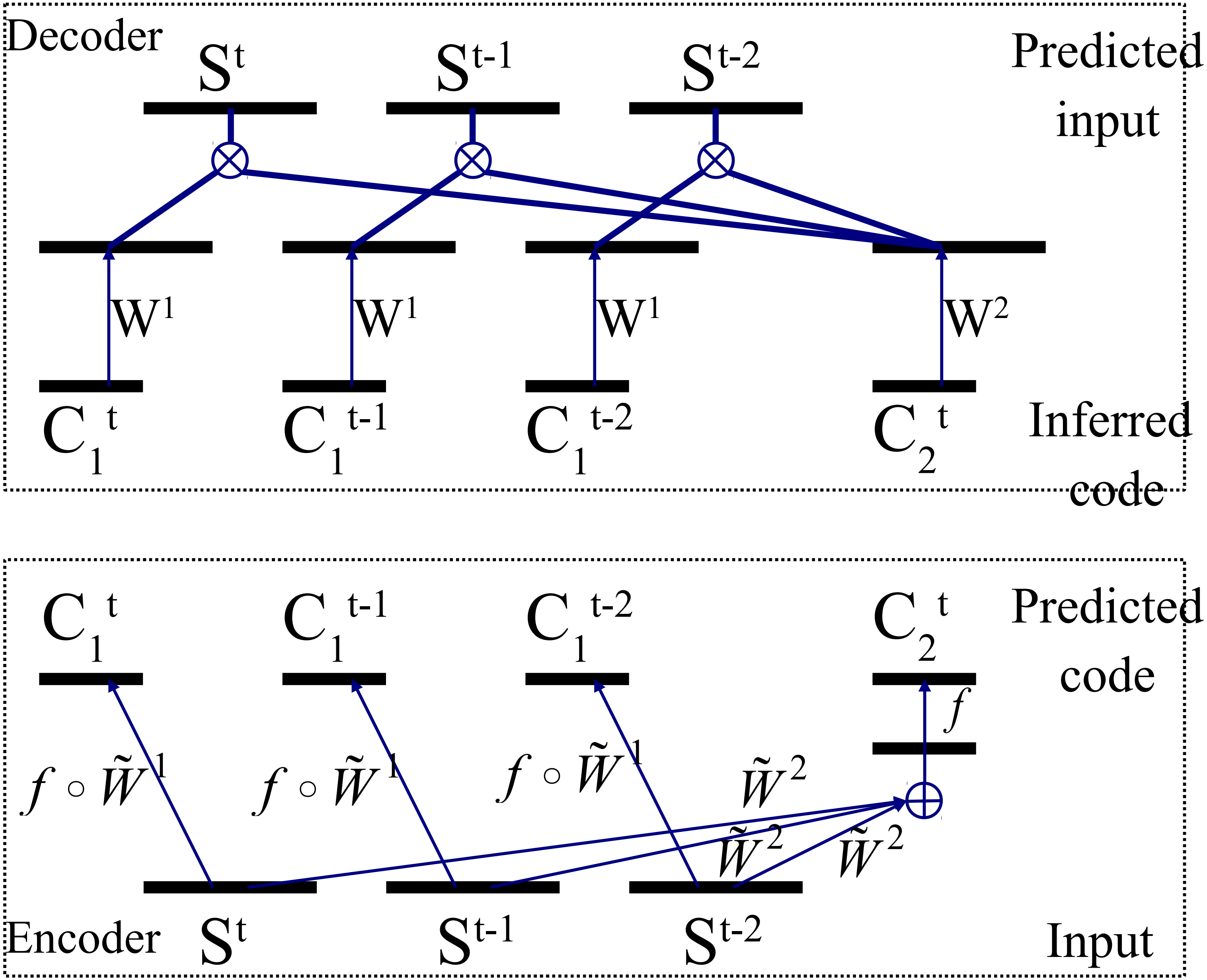
Invariant Features through Temporal Constancy

- Object is cross-product of object type and instantiation parameters
  - Mapping units [Hinton 1981], capsules [Hinton 2011]



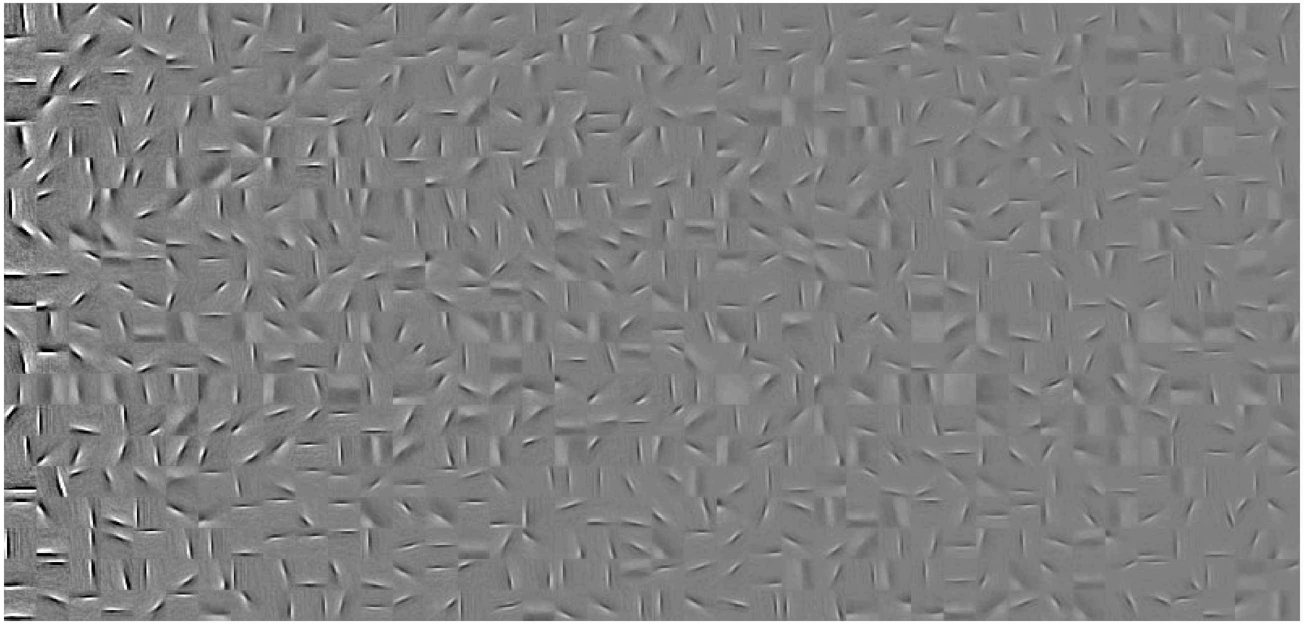
[Karol Gregor et al.]

What-Where Auto-Encoder Architecture

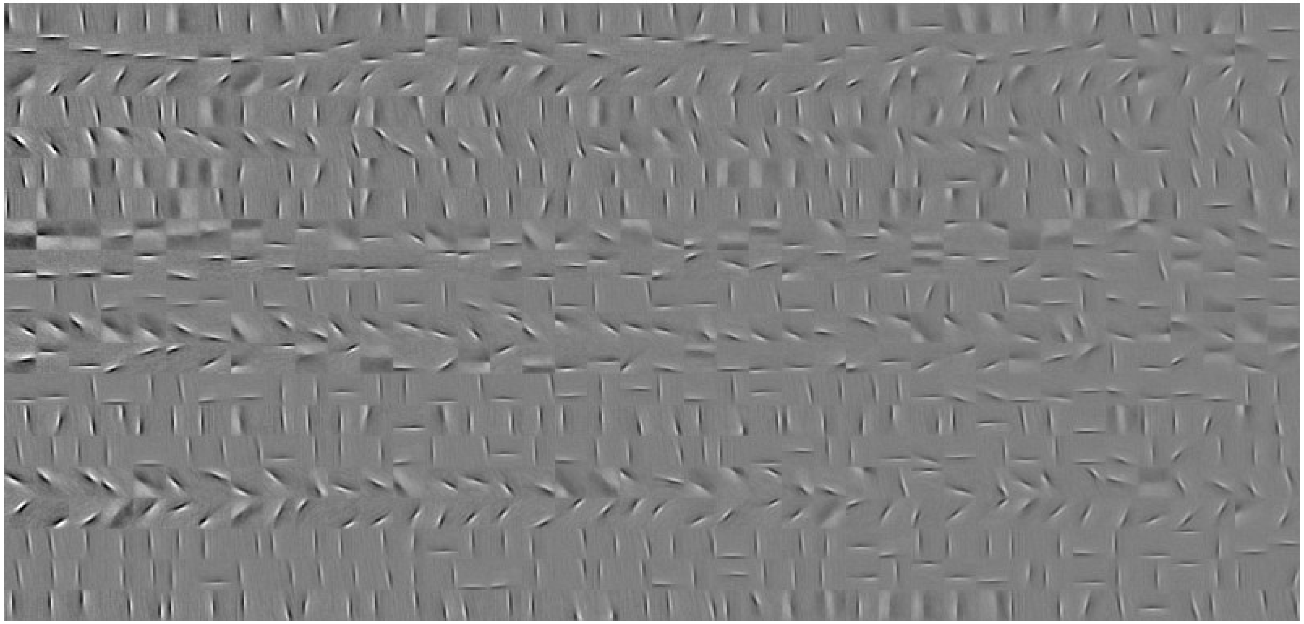


Low-Level Filters Connected to Each Complex Cell

C1
(where)

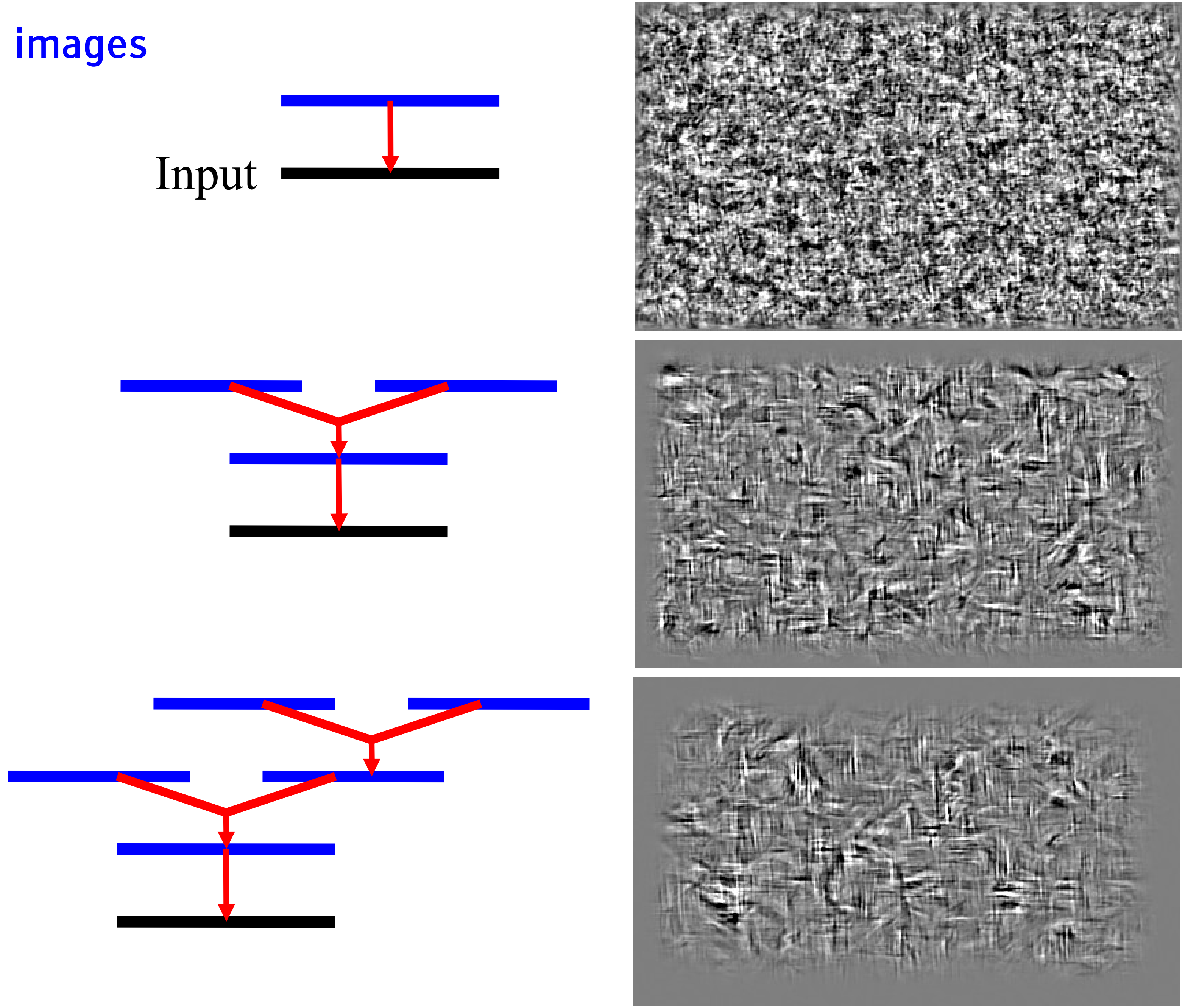


C2
(what)



Generating Images

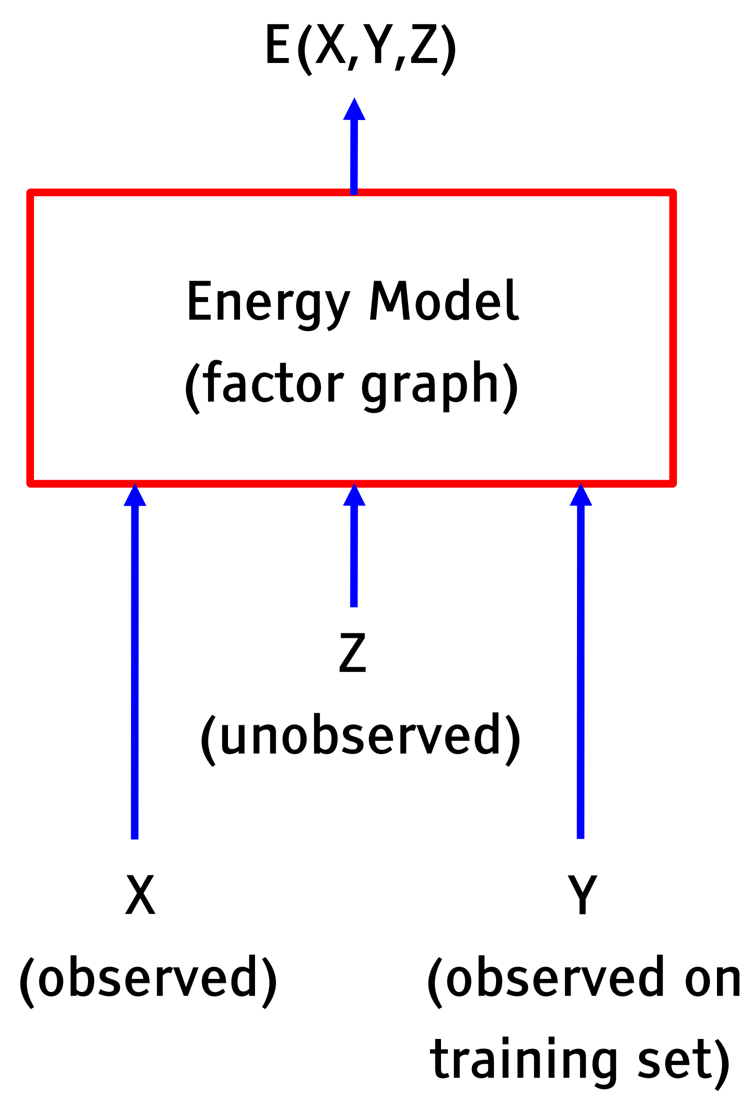
- Generating images



Deep Learning
And
Structured Prediction

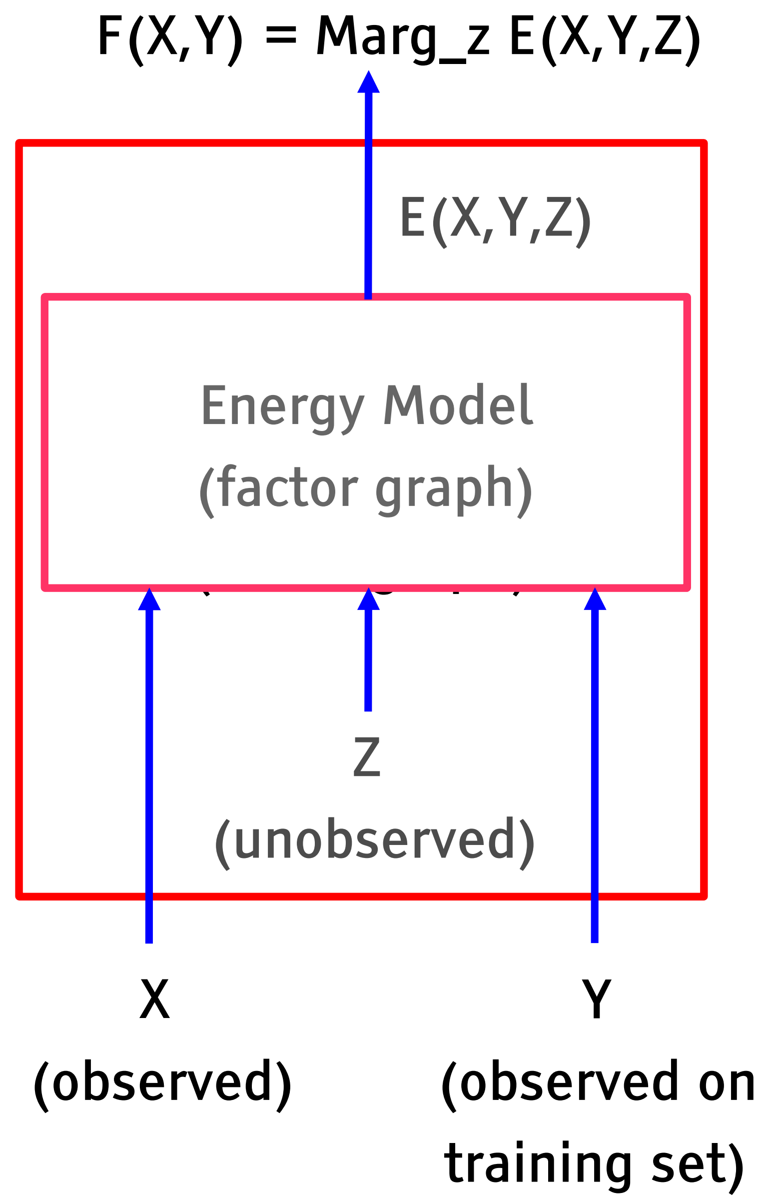
Integrating Deep Learning and Structured Prediction

- Deep Learning systems can be assembled into factor graphs
  - Energy function is a sum of factors
  - Factors can embed whole deep learning systems
  - X: observed variables (inputs)
  - Z: never observed (latent variables)
  - Y: observed on training set (output variables)
- Inference is energy minimization (MAP) or free energy minimization (marginalization) over Z and Y given an X



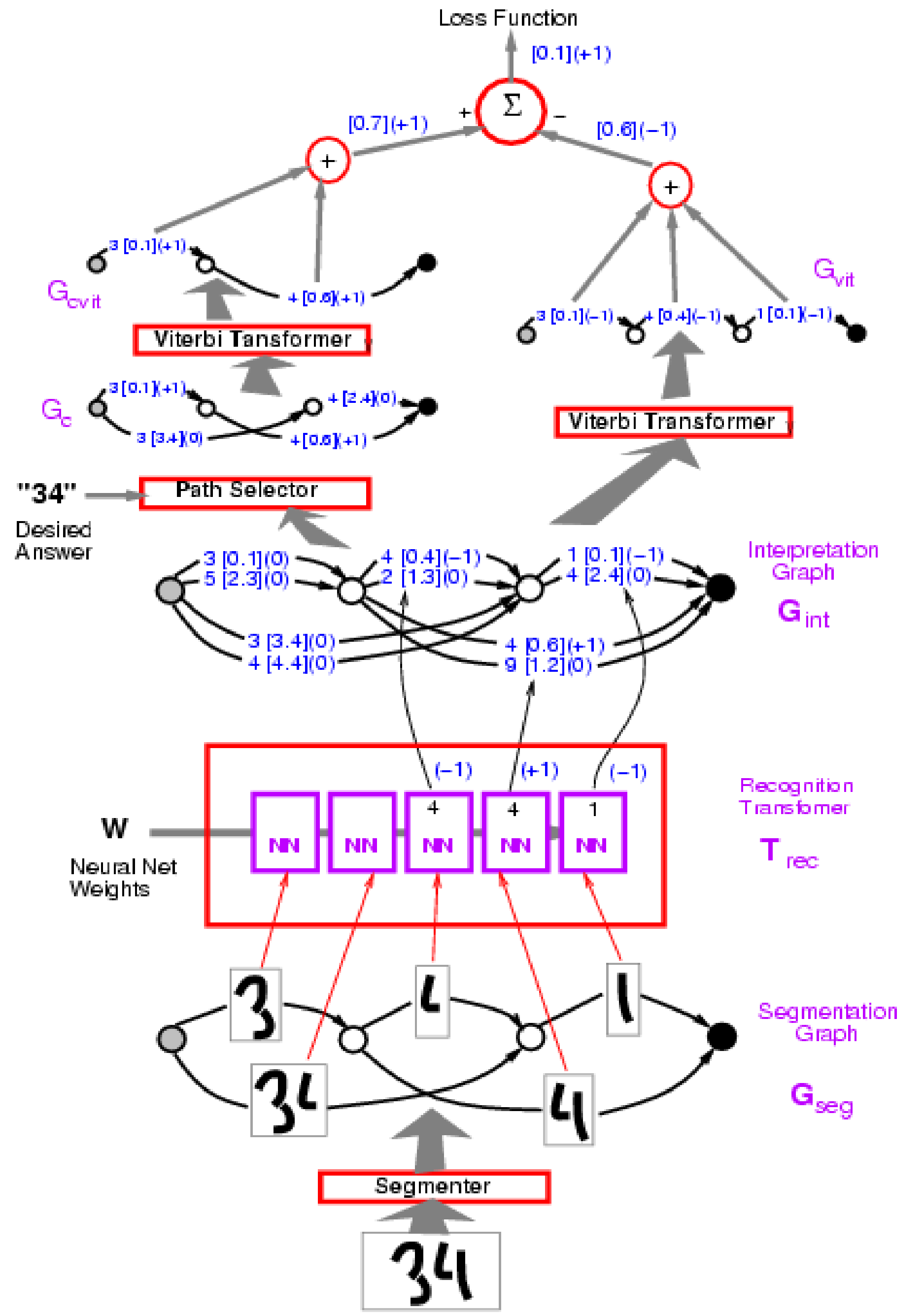
Integrating Deep Learning and Structured Prediction

- Deep Learning systems can be assembled into factor graphs
  - Energy function is a sum of factors
  - Factors can embed whole deep learning systems
  - X: observed variables (inputs)
  - Z: never observed (latent variables)
  - Y: observed on training set (output variables)
- Inference is energy minimization (MAP) or free energy minimization (marginalization) over Z and Y given an X
  - F(X,Y) = MIN_z E(X,Y,Z)
  - F(X,Y) = -log SUM_z exp[-E(X,Y,Z)]



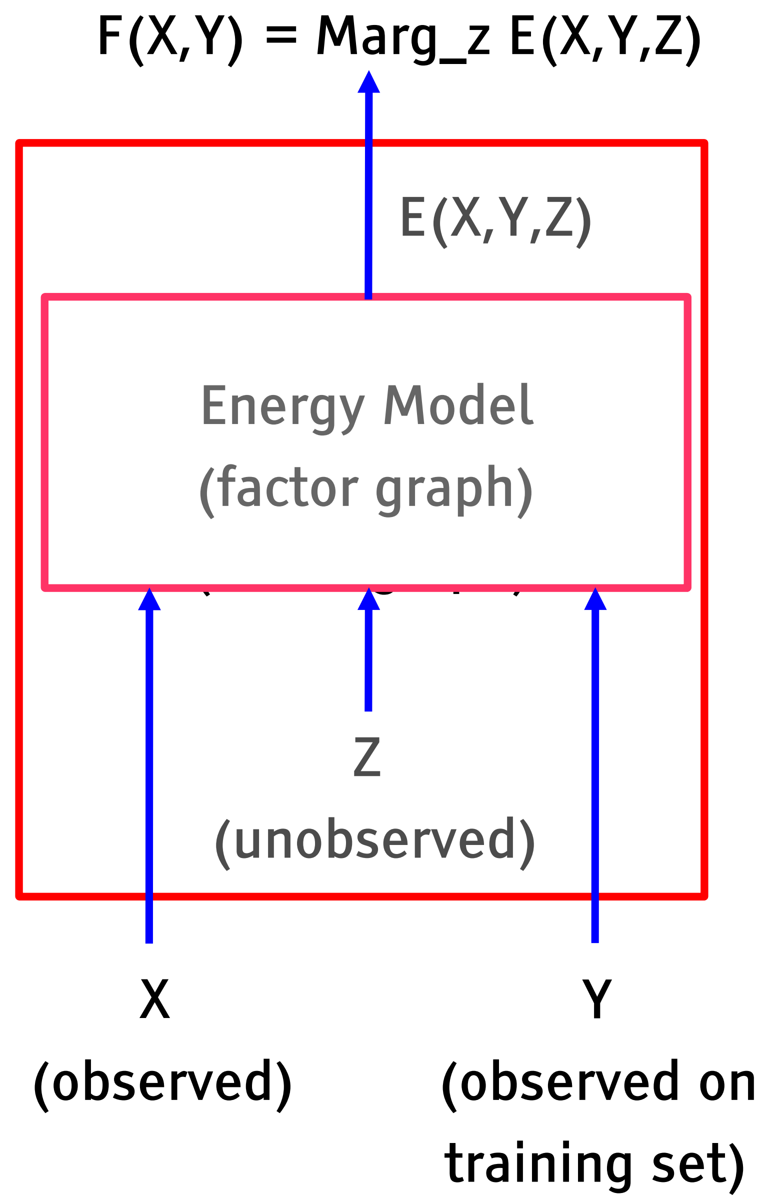
Integrating Deep Learning and Structured Prediction

- [LeCun, Bottou, Bengio, Haffner 1998]
- Integrating deep learning and structured prediction is a very old idea
  - In fact, it predates structured prediction
- Globally-trained convolutional-net + graphical models
  - trained discriminatively at the word level
  - Loss identical to CRF and structured perceptron
  - Compositional movable parts model
- A system like this was reading 10 to 20% of all the checks in the US around 1998



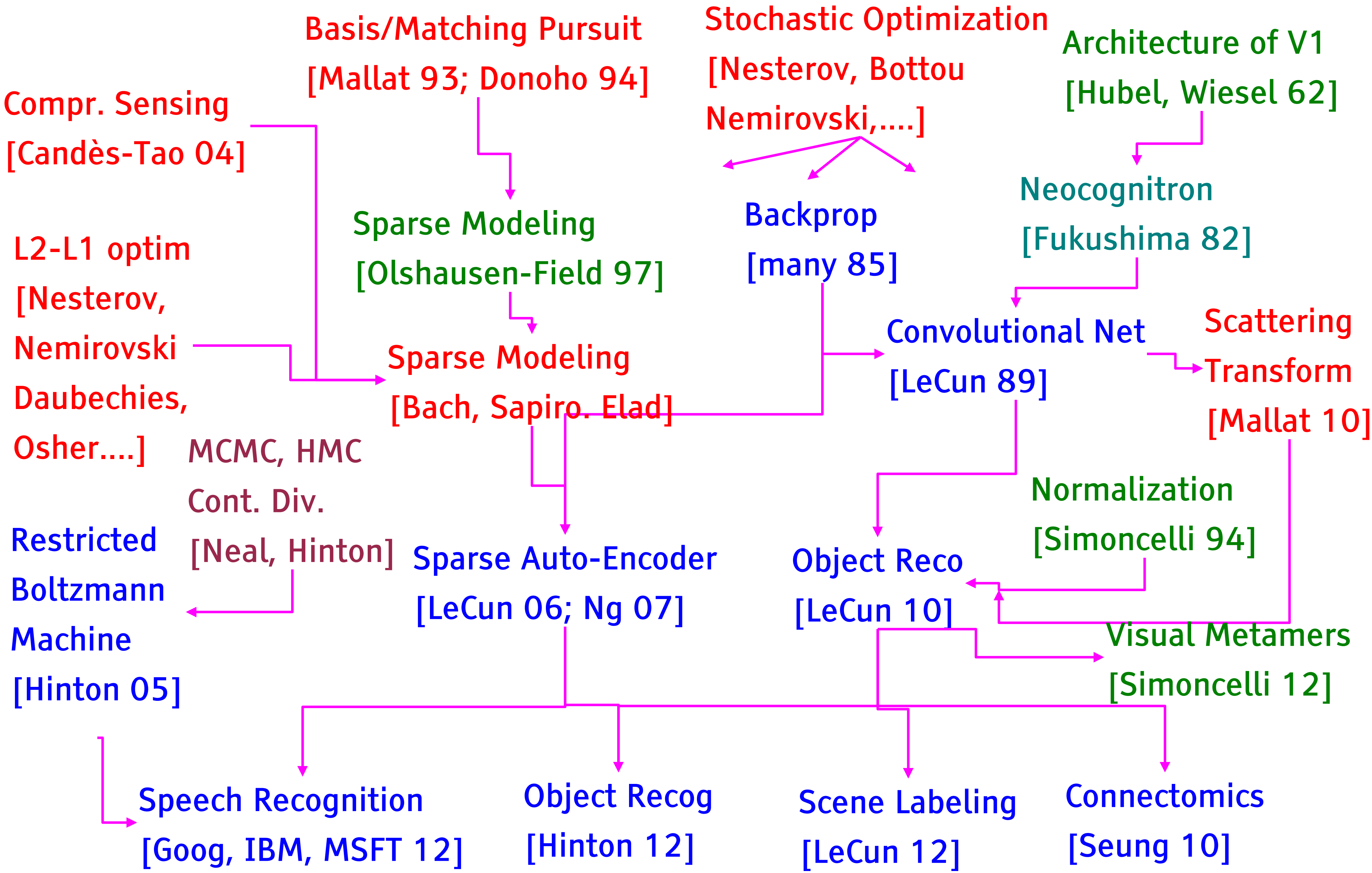
Integrating Deep Learning and Structured Prediction

- Deep Learning systems can be assembled into factor graphs
  - Energy function is a sum of factors
  - Factors can embed whole deep learning systems
  - X: observed variables (inputs)
  - Z: never observed (latent variables)
  - Y: observed on training set (output variables)
- Inference is energy minimization (MAP) or free energy minimization (marginalization) over Z and Y given an X
  - F(X,Y) = MIN_z E(X,Y,Z)
  - F(X,Y) = -log SUM_z exp[-E(X,Y,Z)]



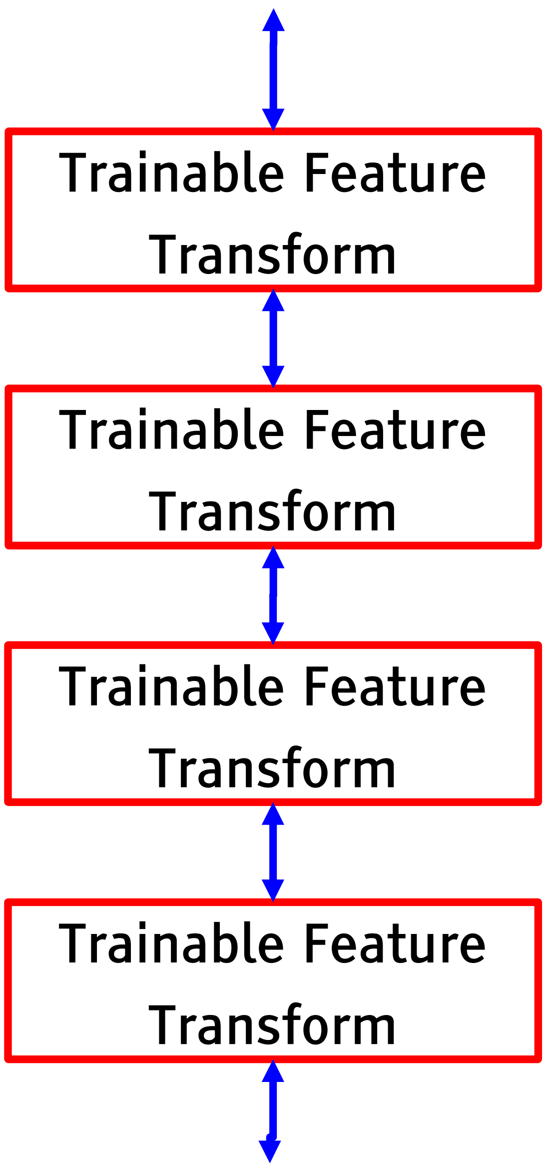
Future Challenges

The Graph of Deep Learning ↔ Sparse Modeling ↔ Neuroscience



The Future: Integrating Feed-Forward and Feedback

- Marrying feed-forward convolutional nets with generative "deconvolutional nets"
  - Deconvolutional networks
    - [Zeiler-Graham-Fergus ICCV 2011]
- Feed-forward/Feedback networks allow reconstruction, multimodal prediction, restoration, etc...
  - Deep Boltzmann machines can do this, but there are scalability issues with training



Towards Practical AI: Challenges

- Applying deep learning to NLP (requires “structured prediction”)
- Video analysis/understanding (requires unsupervised learning)
- High-performance/low power embedded systems for ConvNets (FPGA/ASIC)
- Very-large-scale deep learning (distributed optimization)
- Integrating reasoning with DL (“energy-based models”, recursive neural nets)

- Then we can have
  - Automatically-created high-performance data analytics systems
  - Multimedia content understanding, search and indexing
  - Multilingual speech dialog systems
  - Driver-less cars
  - Autonomous maintenance robots / personal care robots

Future Challenges

- Integrated feed-forward and feedback
  - Deep Boltzmann machine do this, but there are issues of scalability.
- Integrating supervised and unsupervised learning in a single algorithm
  - Again, deep Boltzmann machines do this, but....
- Integrating deep learning and structured prediction (“reasoning”)
  - This has been around since the 1990's but needs to be revived
- Learning representations for complex reasoning
  - “recursive” networks that operate on vector space representations of knowledge [Pollack 90's] [Bottou 2010] [Socher, Manning, Ng 2011]
- Representation learning in natural language processing
  - [Y. Bengio 01],[Collobert Weston 10], [Mnih Hinton 11] [Socher 12]
- Better theoretical understanding of deep learning and convolutional nets
  - e.g. Stephane Mallat's “scattering transform”, work on the sparse representations from the applied math community....