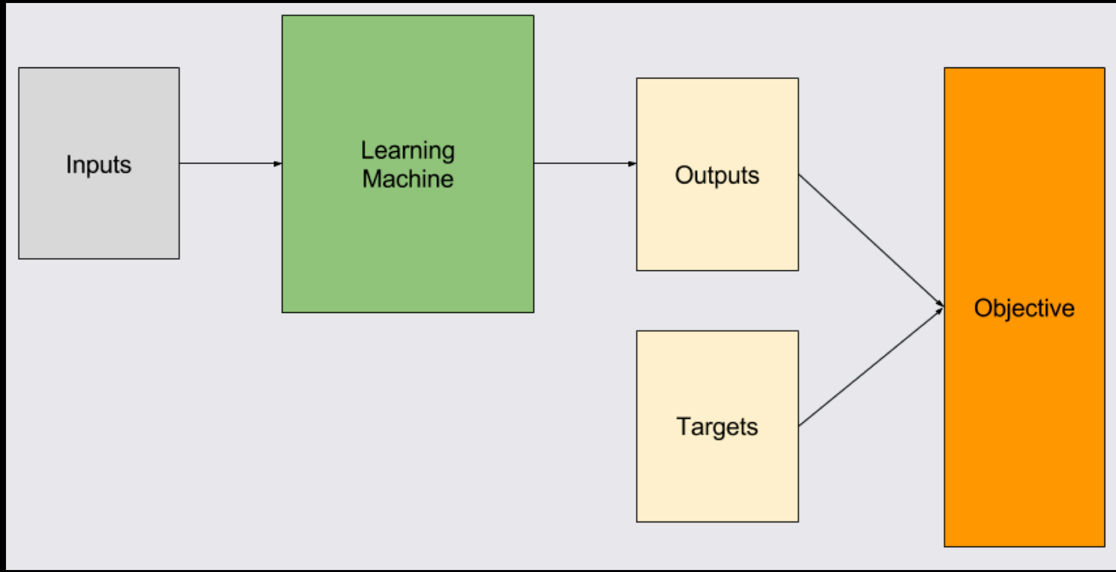
Unsupervised Learning

Lecture 12

Motivation

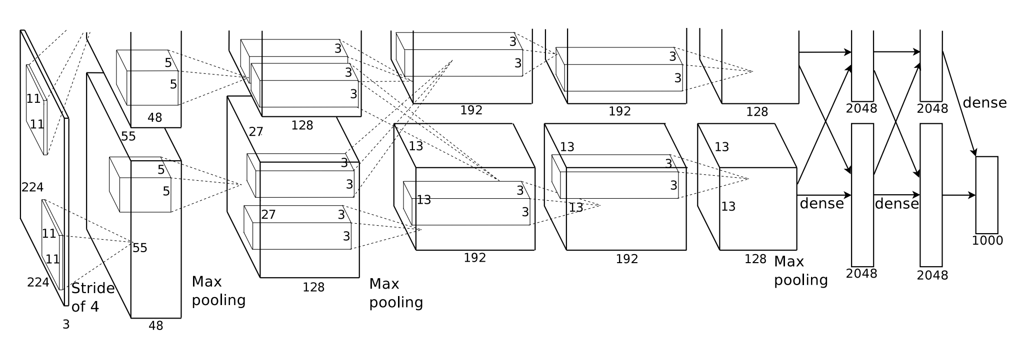
- Goal: learn structure of data to yield generic representation, useful for many different tasks
- Supervised learning: given labels/targets for a particular task
  - Task strongly constrains objective function



- In unsupervised learning, where do labels/output targets come from?
  - What is the objective?

Motivation

- Most successes obtained with supervised models, e.g. Convnets



- Unsupervised learning methods less successful

- But likely to be very important in long-term

Different Perspectives on Unsupervised Learning

1. Density estimation

Given data {x}, build p(x).

E.g. k-Means, PCA, RBMs, sparse coding etc.

Assumption: good model of data requires representation that will be generically useful

2. Train model to use "free-labels" from task that is somewhat similar to one we actually care about

- Often called "self-supervised" learning
- Free-labels often come from exploiting knowledge of domain

Historical Note

- Deep Learning revival started in ~2006
  - Hinton & Salakhudinov Science paper on RBMs

- Unsupervised Learning was focus from 2006-2012

- In ~2012 great results in vision, speech with supervised methods appeared
  - Less interest in unsupervised learning

Arguments for Unsupervised Learning

- Want to be able to exploit unlabeled data
  - Vast amount of it often available
  - Essentially free

- Good regularizer for supervised learning
  - Helps generalization
  - Transfer learning
  - Zero / one-shot learning

Another Argument for Unsupervised Learning

When we're learning to see, nobody's telling us what the right answers are — we just look. Every so often, your mother says "that's a dog", but that's very little information.

You'd be lucky if you got a few bits of information — even one bit per second — that way. The brain's visual system has 10^{14} neural connections. And you only live for 10^9 seconds.

So it's no use learning one bit per second. You need more like 10^5 bits per second. And there's only one place you can get that much information: from the input itself.

— Geoffrey Hinton, 1996

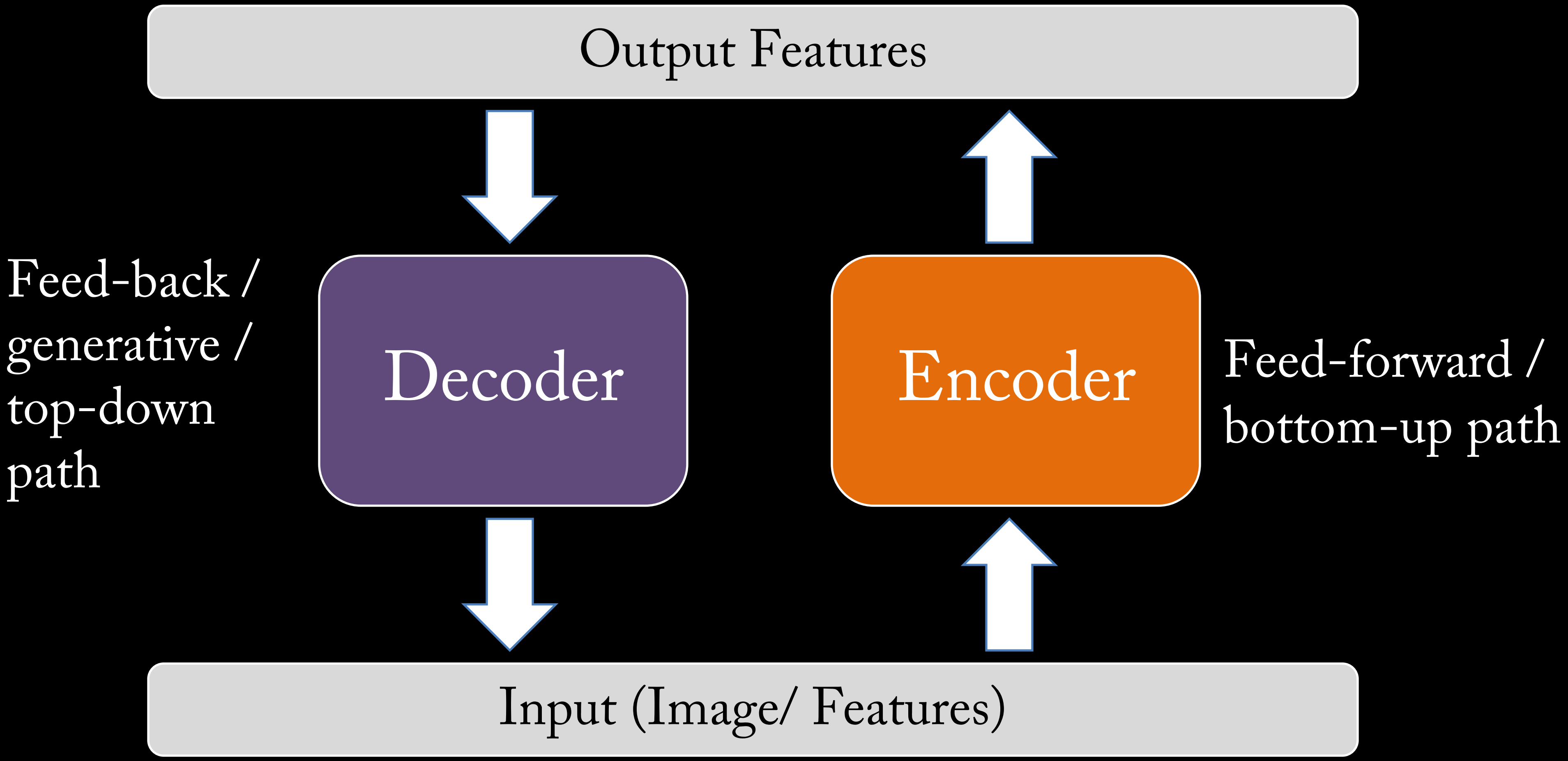
Taxonomy of Approaches

- Autoencoder (most common)
  - RBMs / DBMs
  - Denoising autoencoders
  - Predictive sparse decomposition
- Decoder-only
  - Sparse coding
  - Deconvolutional Nets

Loss involves some kind of reconstruction error

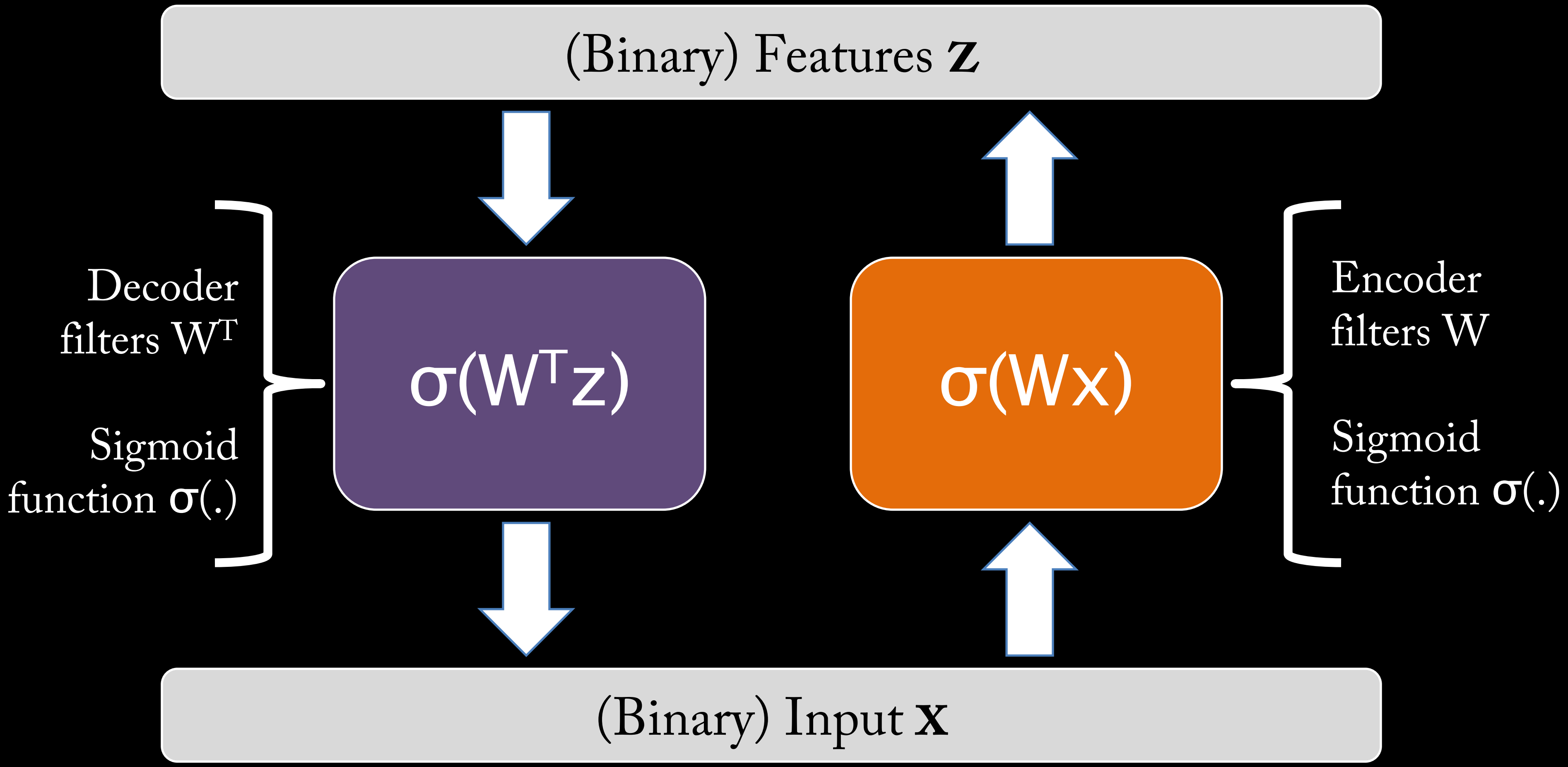
- Encoder-only
  - Implicit or self-supervision, e.g. from video
- Adversarial Networks [to be covered in class on 12/16]

Auto-Encoder



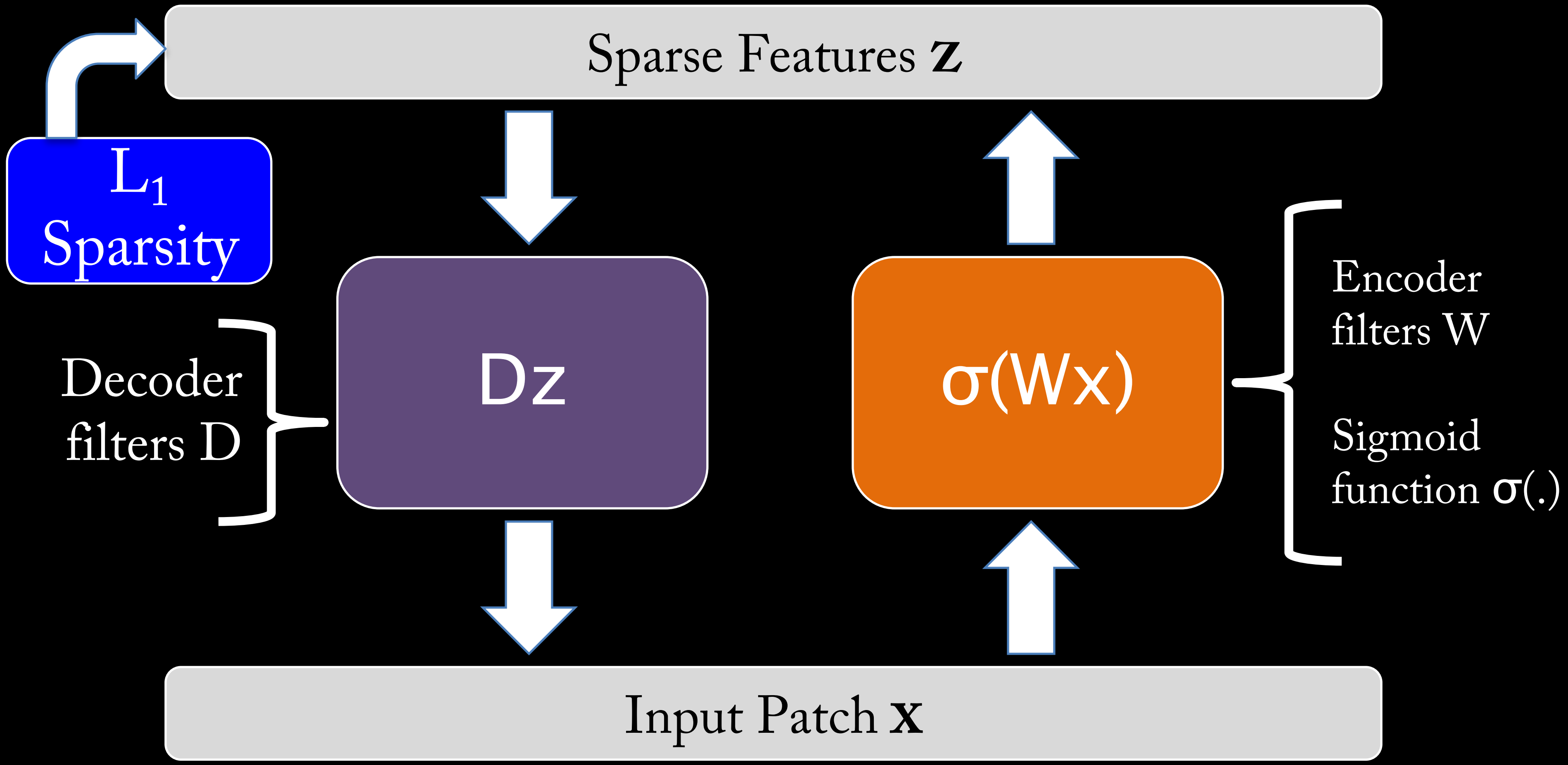
Auto-Encoder Example 1

- Restricted Boltzmann Machine [Hinton '02]



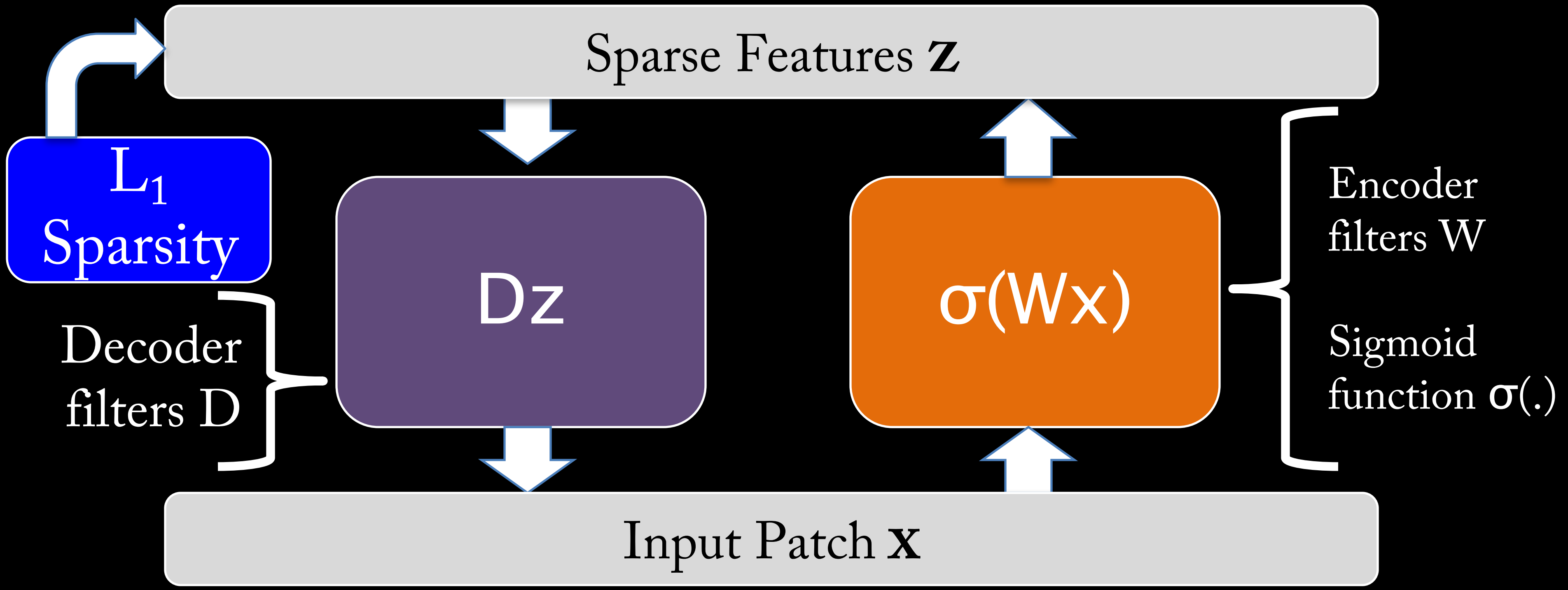
Auto-Encoder Example 2

- Predictive Sparse Decomposition [Ranzato et al., '07]



Auto-Encoder Example 2

- Predictive Sparse Decomposition [Kavukcuoglu et al., '09]

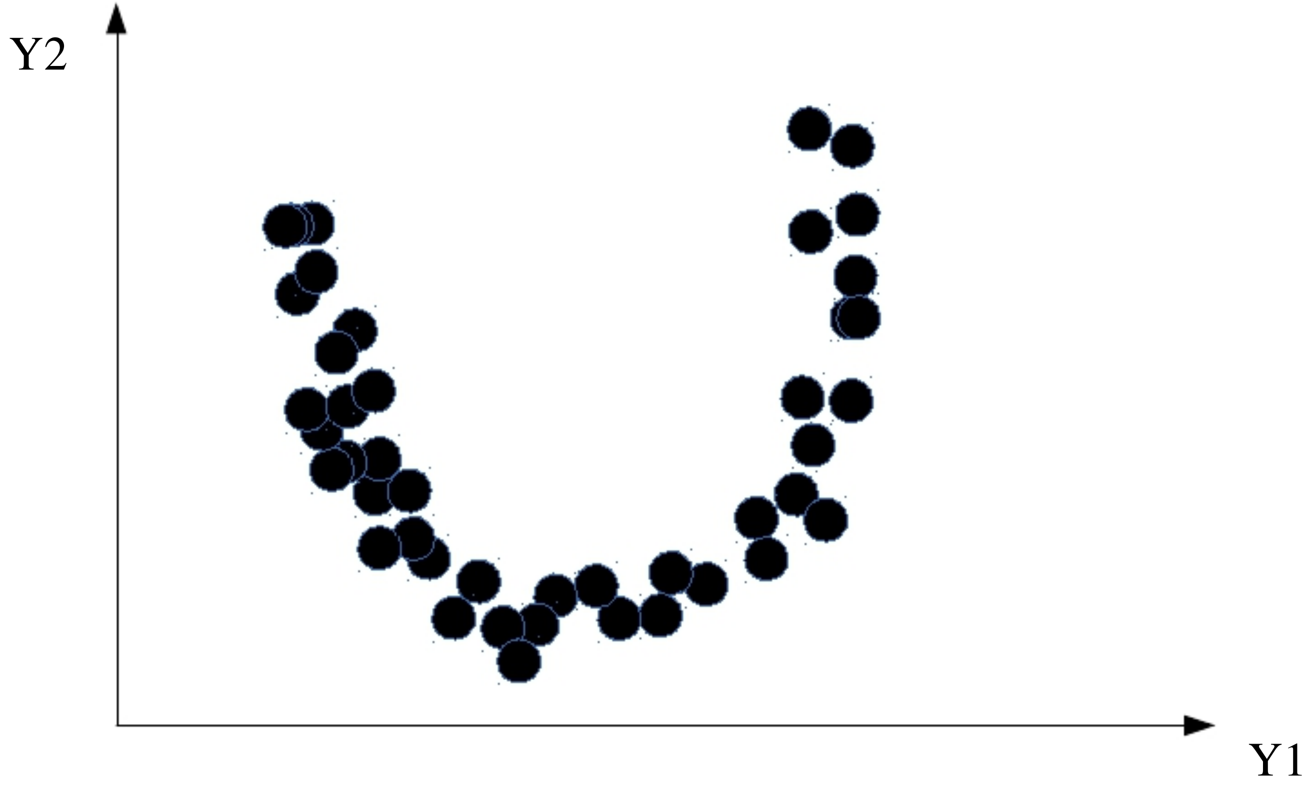


$$\min_{D,W,z} \underbrace{\|Dz - x\|_2^2 + \lambda|z|_1}_{Decoder} + \underbrace{\|\sigma(Wx) - z\|_2^2}_{Encoder}$$

Training

Energy-Based Unsupervised Learning

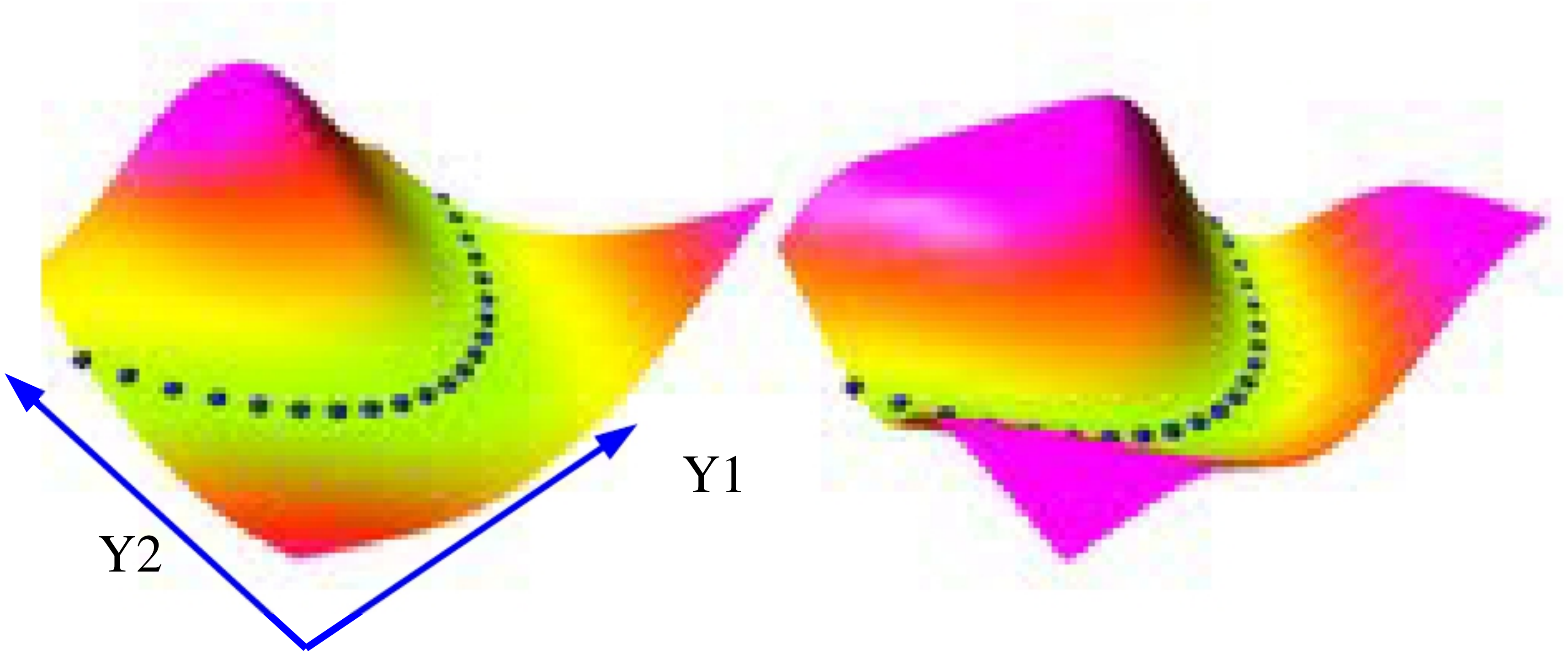
- Learning an energy function (or contrast function) that takes
  - Low values on the data manifold
  - Higher values everywhere else



Capturing Dependencies Between Variables with an Energy Function

Y LeCun
MA Ranzato

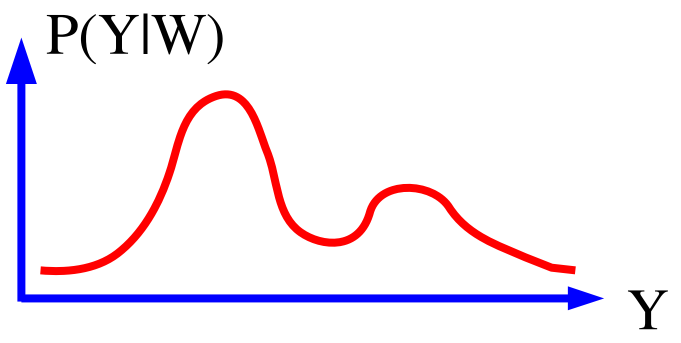
- **The energy surface is a "contrast function" that takes low values on the data manifold, and higher values everywhere else**
  - Special case: energy = negative log density
  - Example: the samples live in the manifold $Y_2 = Y_1{}^2$



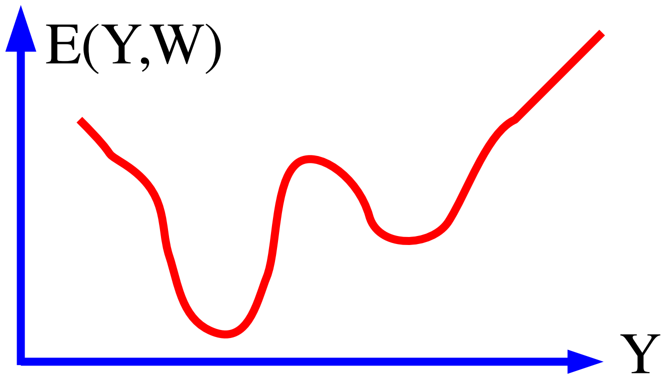
Y LeCun
MA Ranzato

- **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

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



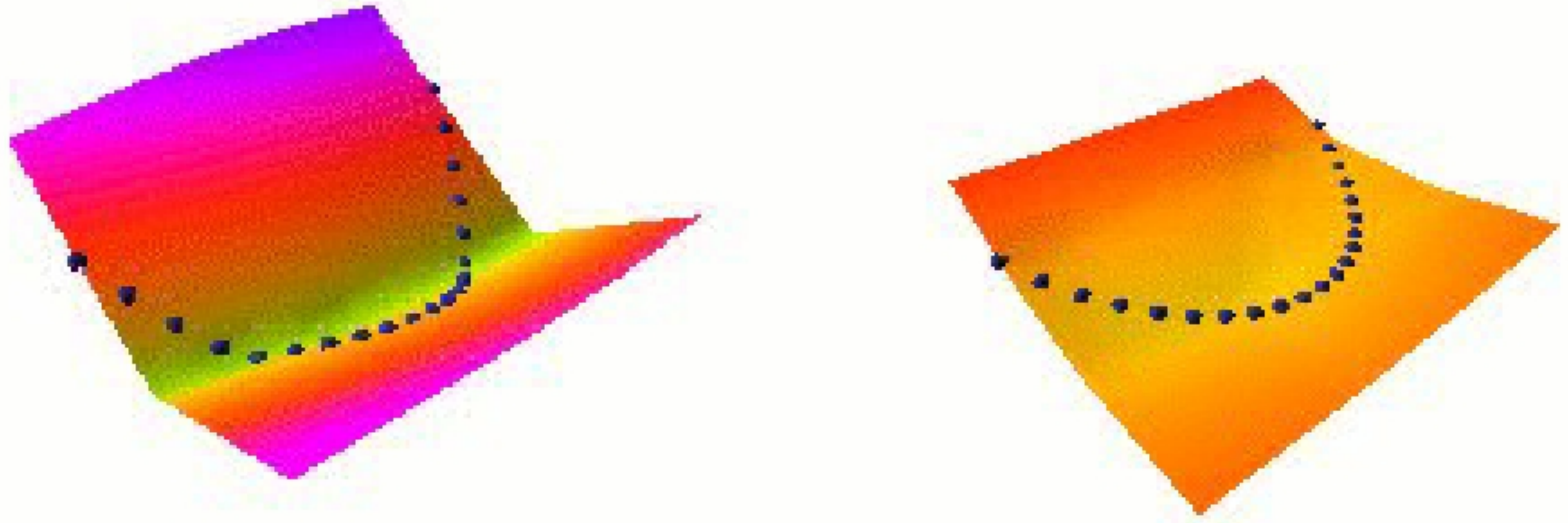
$$E(Y,W) \propto -\log P(Y|W)$$



Learning the Energy Function

Y LeCun
MA Ranzato

- **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

Y LeCun
MA Ranzato

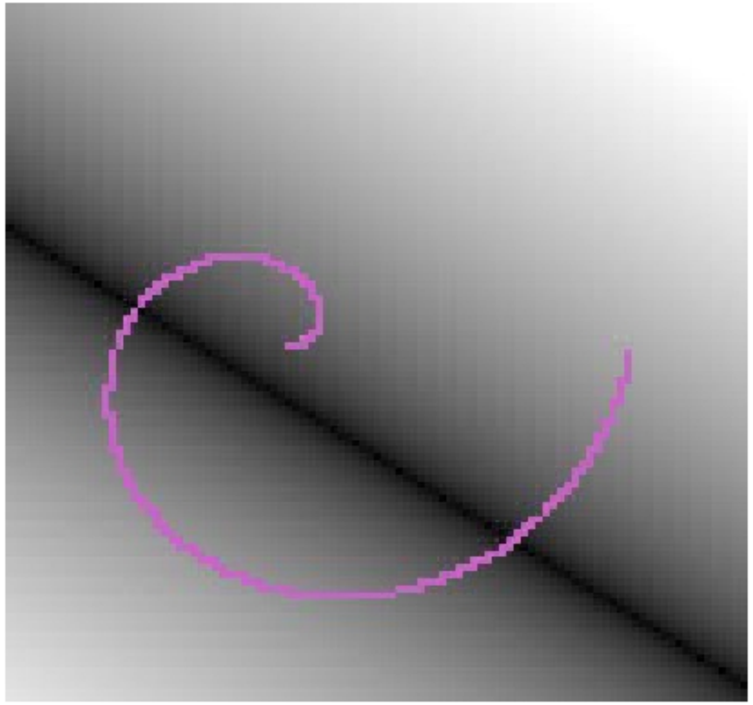
- **1. build the machine so that the volume of low energy stuff is constant**
  - PCA, K-means, GMM, square ICA
- **2. push down of the energy of data points, push up everywhere else**
  - Max likelihood (needs tractable partition function)
- **3. push down of the energy of data points, push up on chosen 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

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

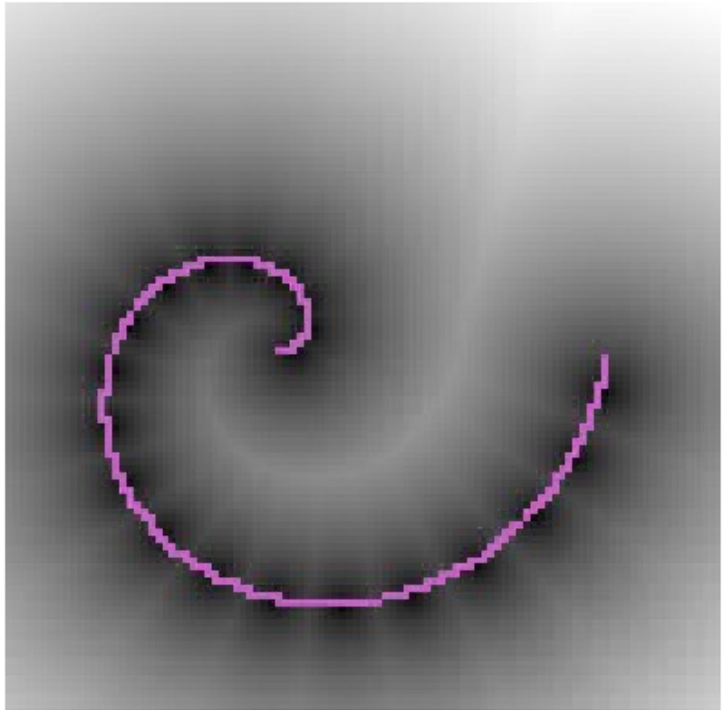
PCA

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



K-Means,
Z constrained to 1-of-K code

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



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

Y LeCun
MA Ranzato

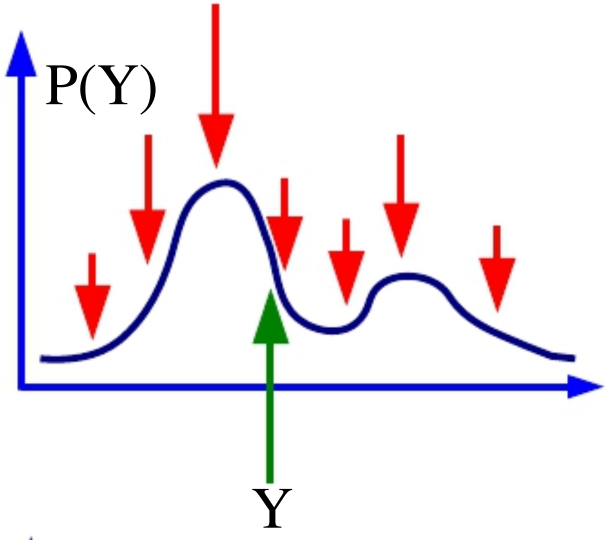
- **Max likelihood (requires a tractable partition function)**

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

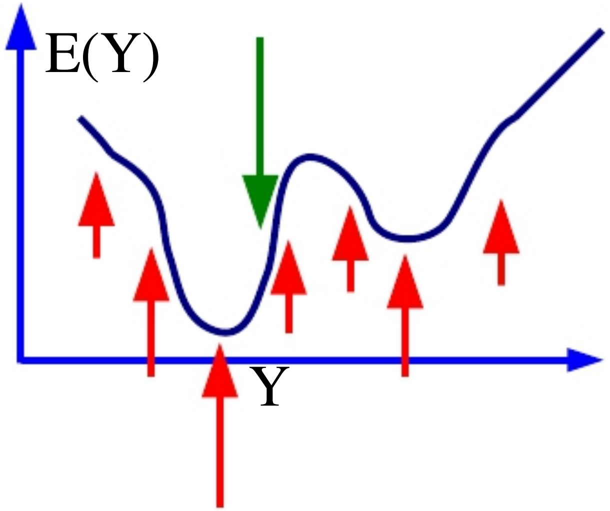


Minimizing -log P(Y,W) on training samples

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

make this small

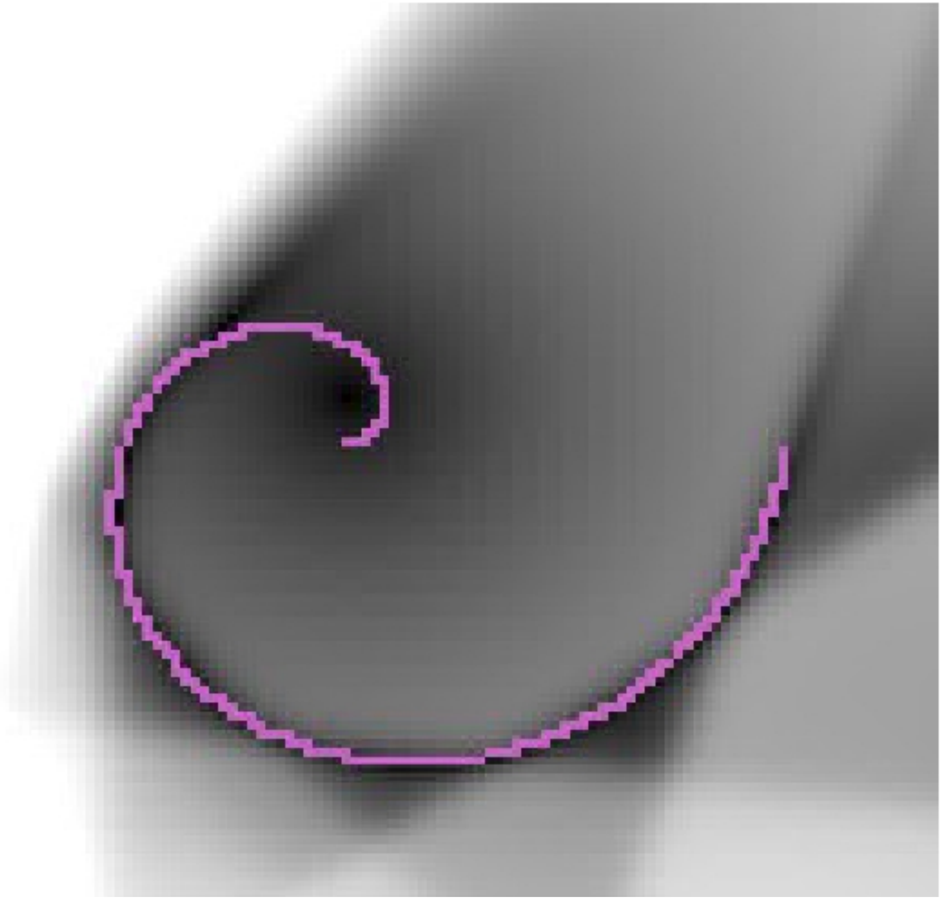
make this big



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

Y LeCun
MA Ranzato

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



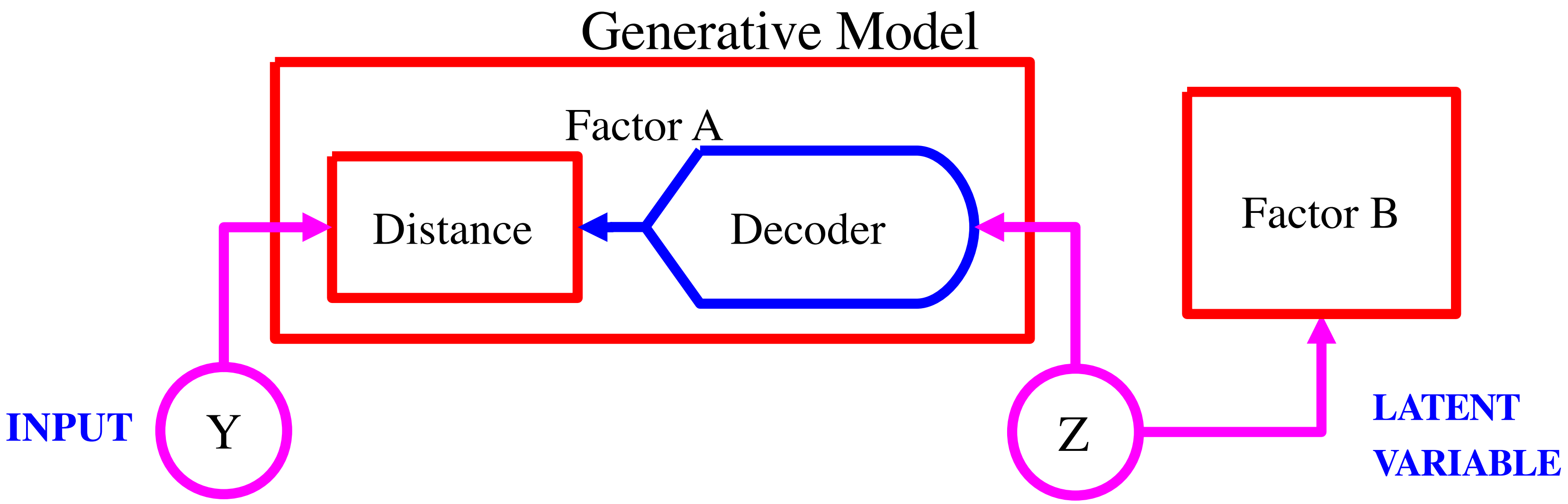
Y LeCun
MA Ranzato

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

How to Speed Up Inference in a Generative Model?

Y LeCun
MA Ranzato

- **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**

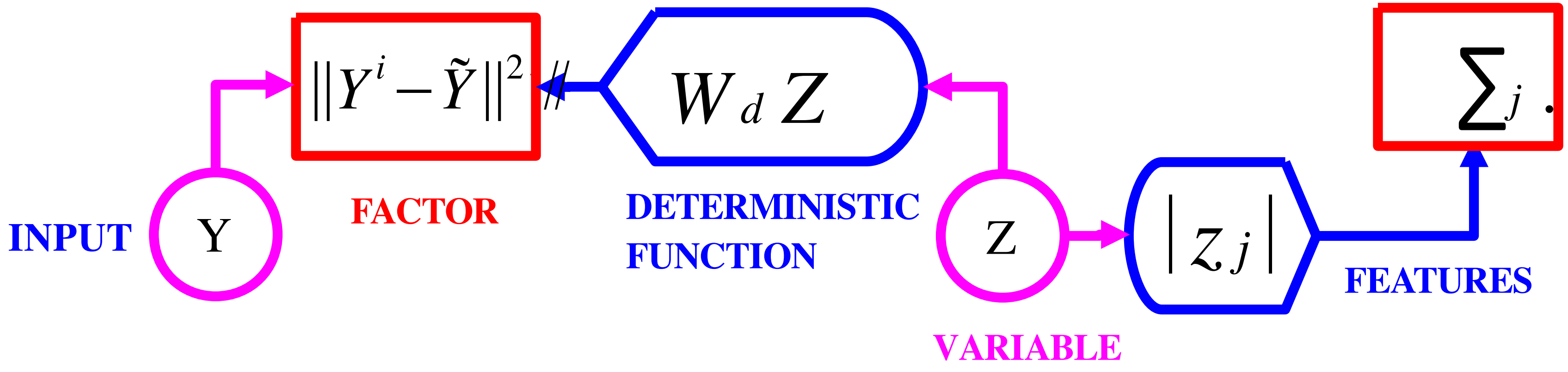


Sparse Coding & Sparse Modeling

[Olshausen & Field 1997]

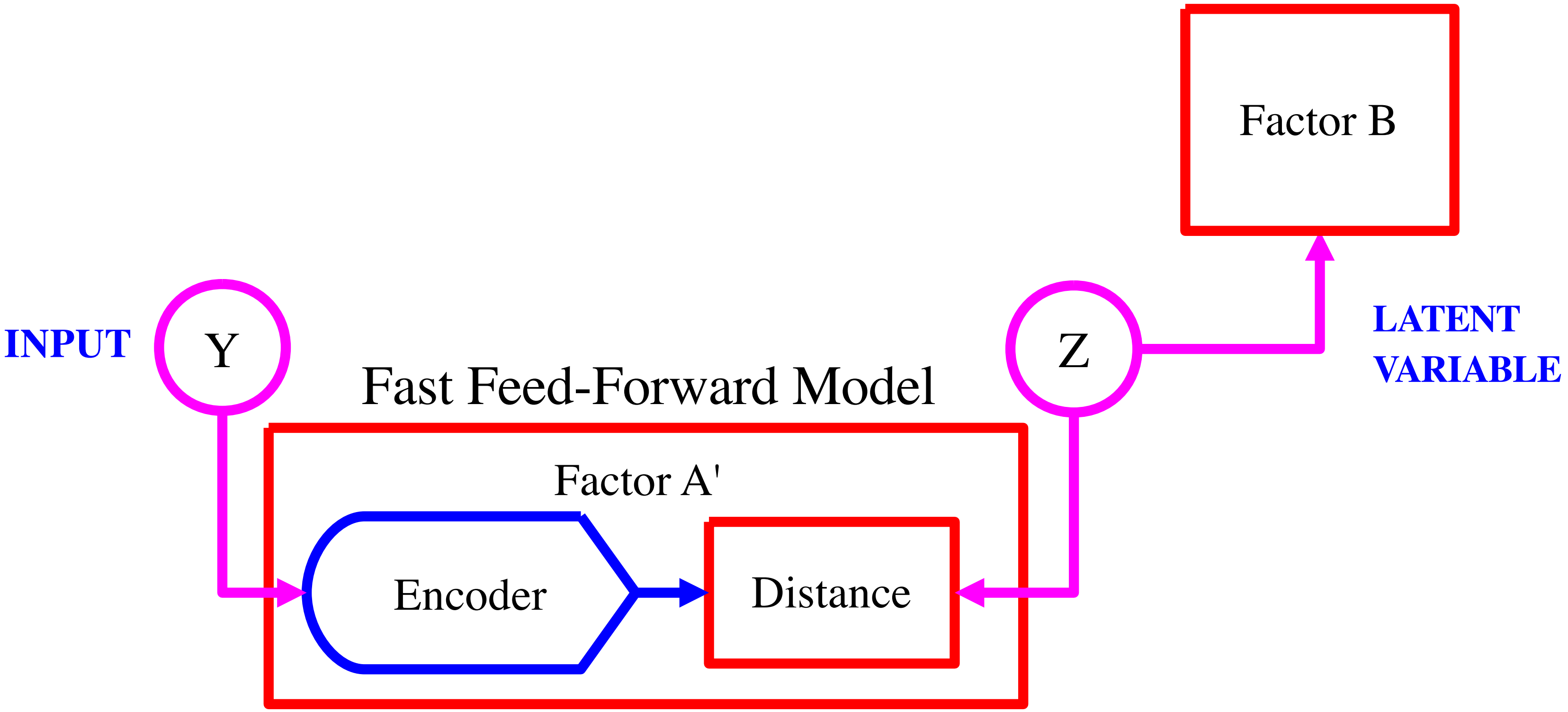
- **Sparse linear reconstruction**
- **Energy = reconstruction_error + code_prediction_error + code_sparsity**

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



- **Inference is slow** $\quad Y \rightarrow \hat{Z} = \mathrm{argmin}_Z\, E(Y, Z)$

Examples: most ICA models, Product of Experts

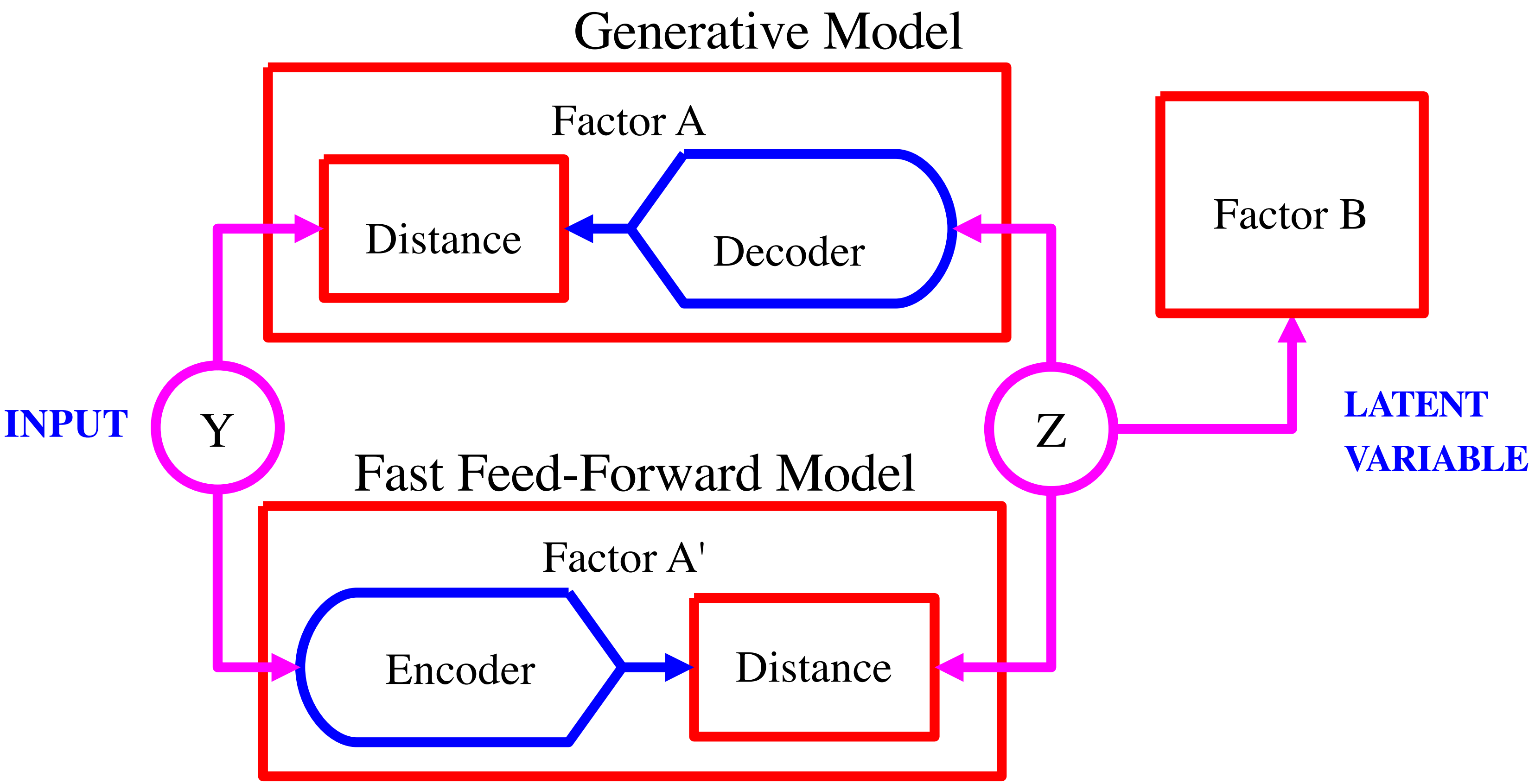


Encoder-Decoder Architecture

Y LeCun
MA Ranzato

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

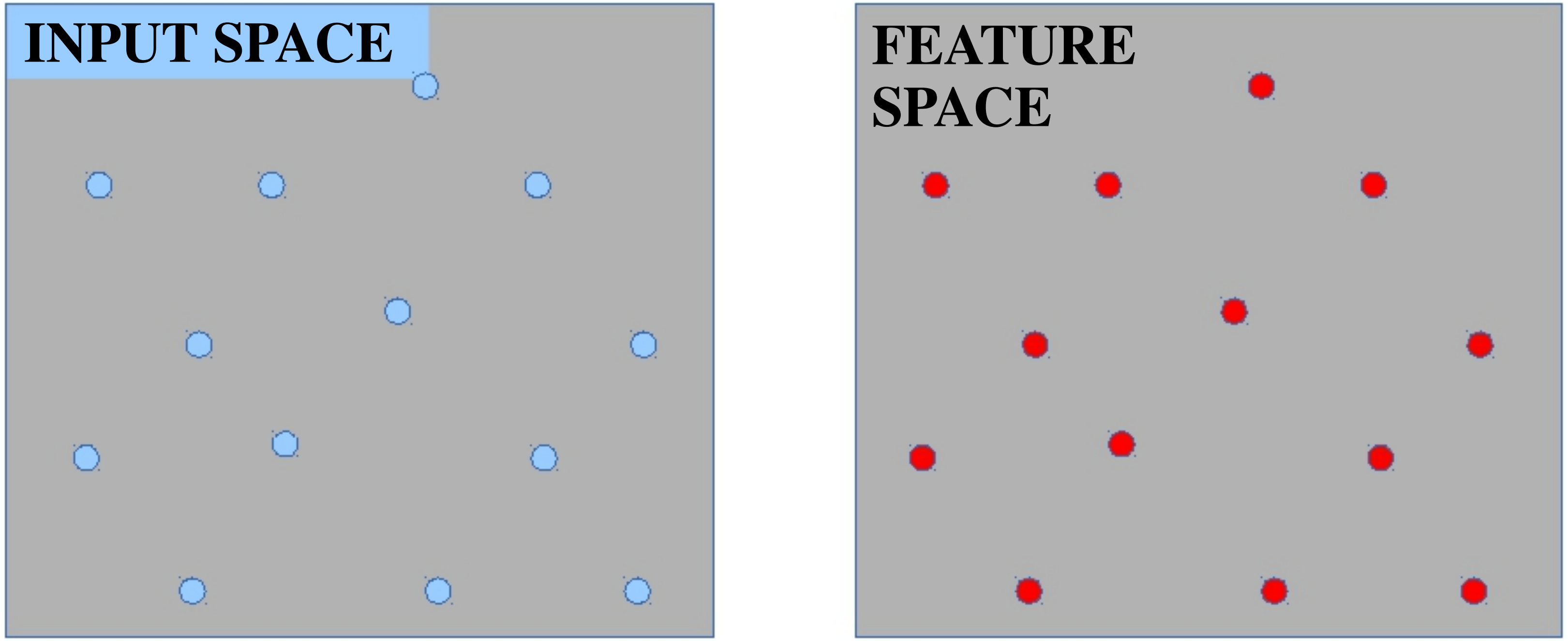
- Train a “simple” feed-forward function to predict the result of a complex optimization on the data points of interest



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

Why Limit the Information Content of the Code?

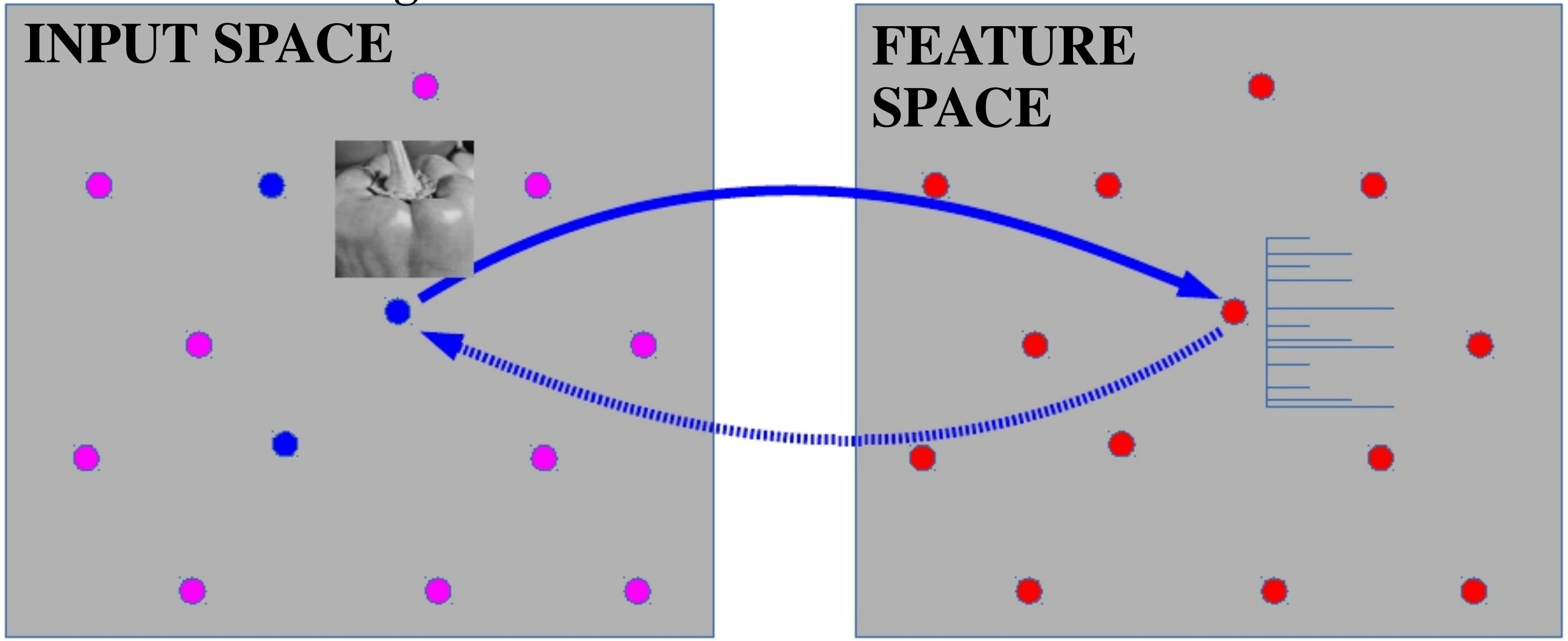
- Training sample
- Input vector which is NOT a training sample
- Feature vector



Why Limit the Information Content of the Code?

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

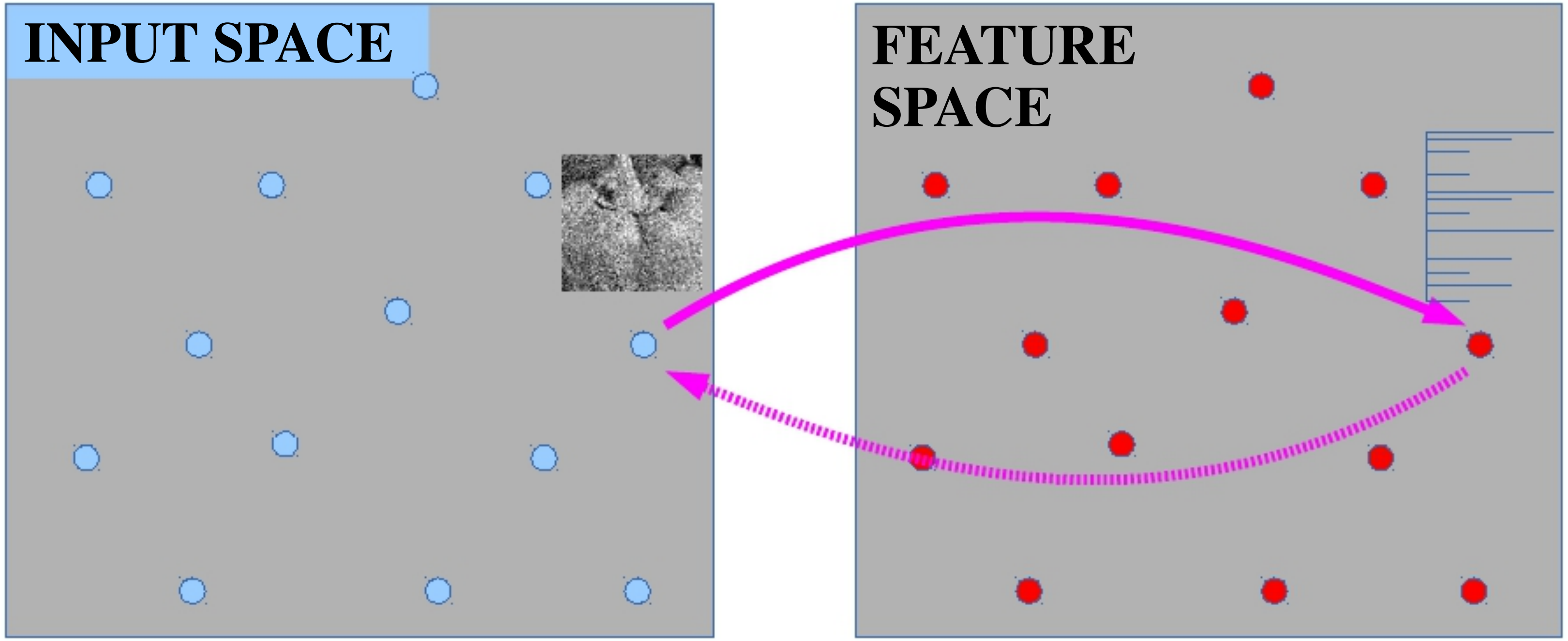
Training based on minimizing the reconstruction error over the training set



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.

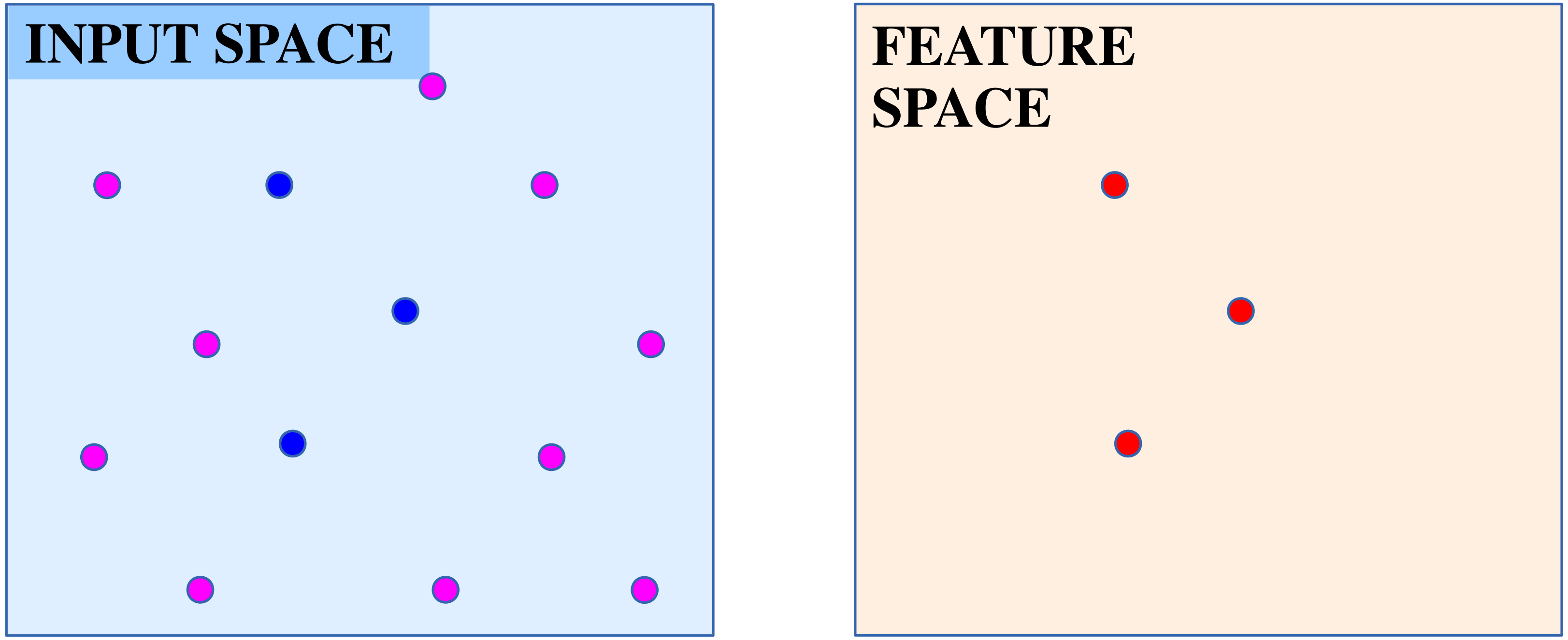


Why Limit the Information Content of the Code?

Y LeCun
MA Ranzato

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

IDEA: reduce number of available codes.

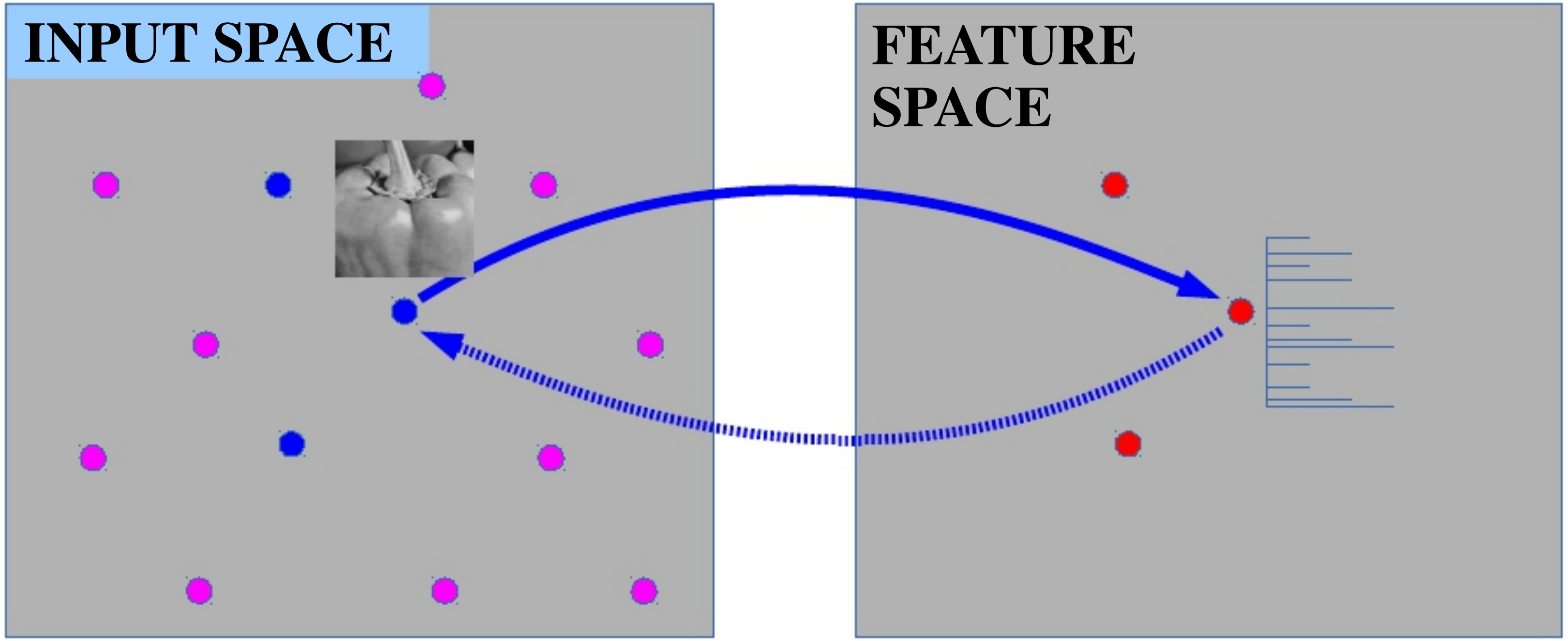


Why Limit the Information Content of the Code?

Y LeCun
MA Ranzato

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

IDEA: reduce number of available codes.

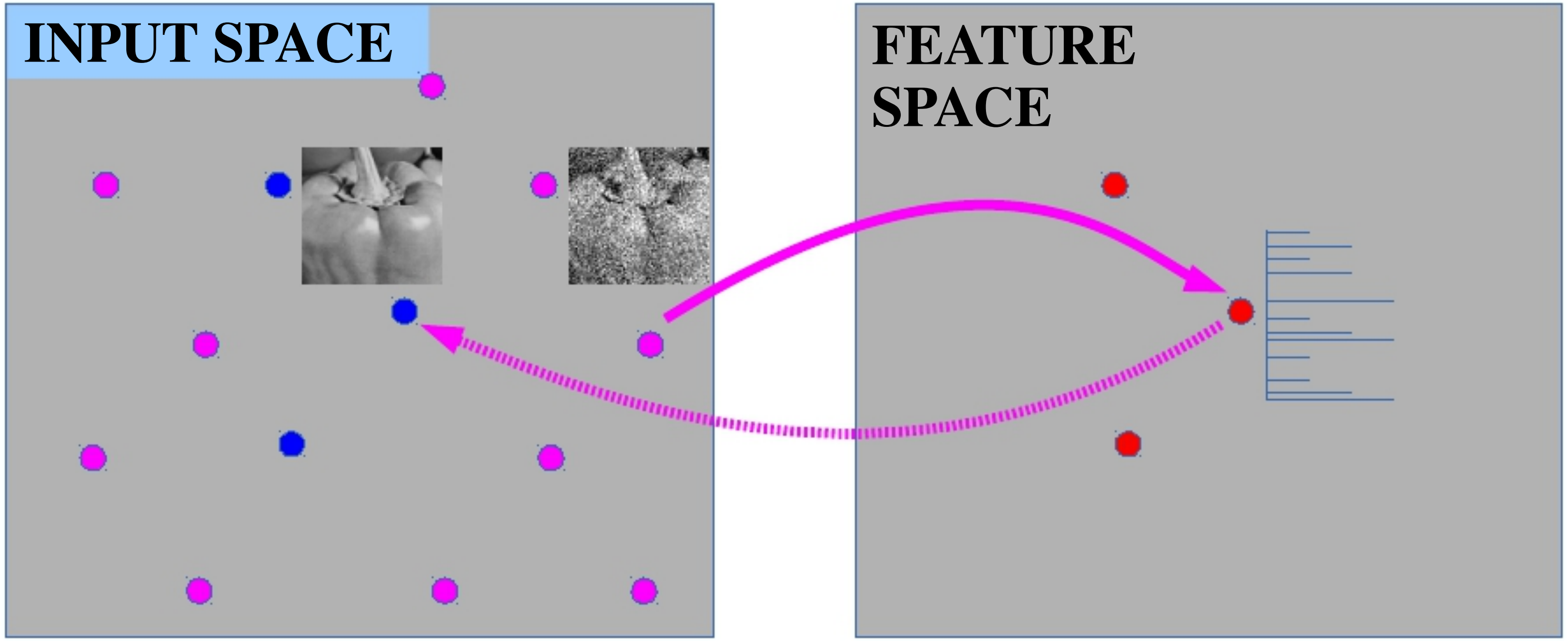


Why Limit the Information Content of the Code?

Y LeCun
MA Ranzato

- 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

Y LeCun
MA Ranzato

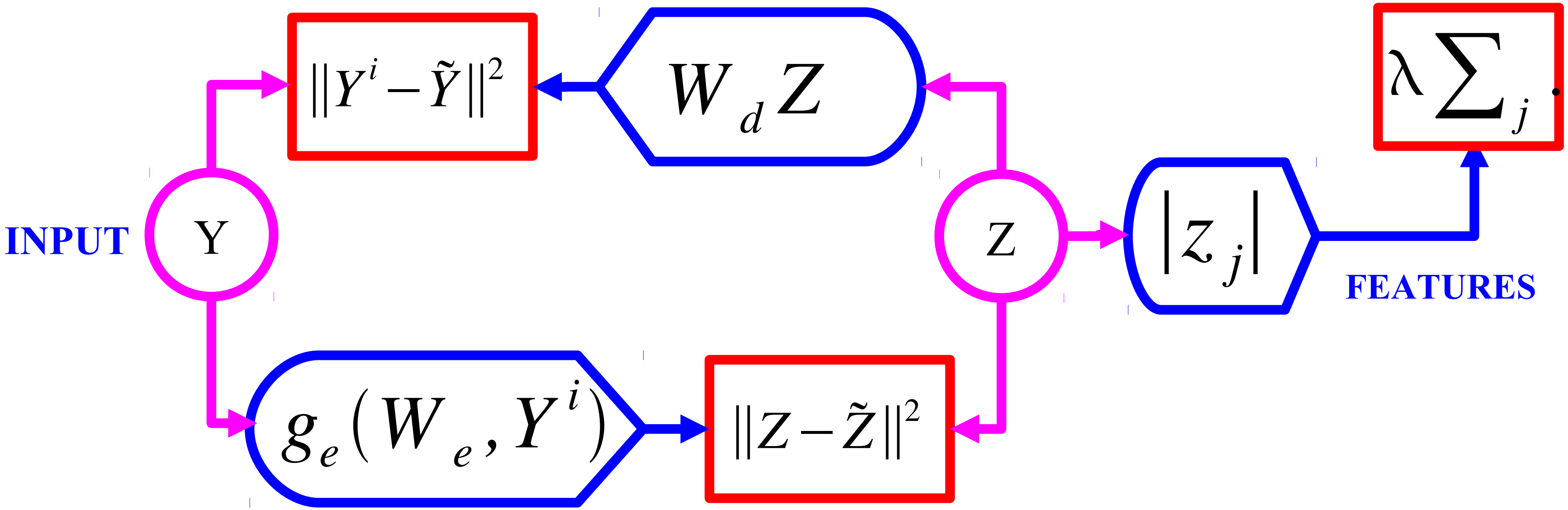
[Kavukcuoglu, Ranzato, LeCun, 2008 → arXiv:1010.3467],

Prediction the optimal code with a trained encoder

Energy = reconstruction_error + code_prediction_error + code_sparsity

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

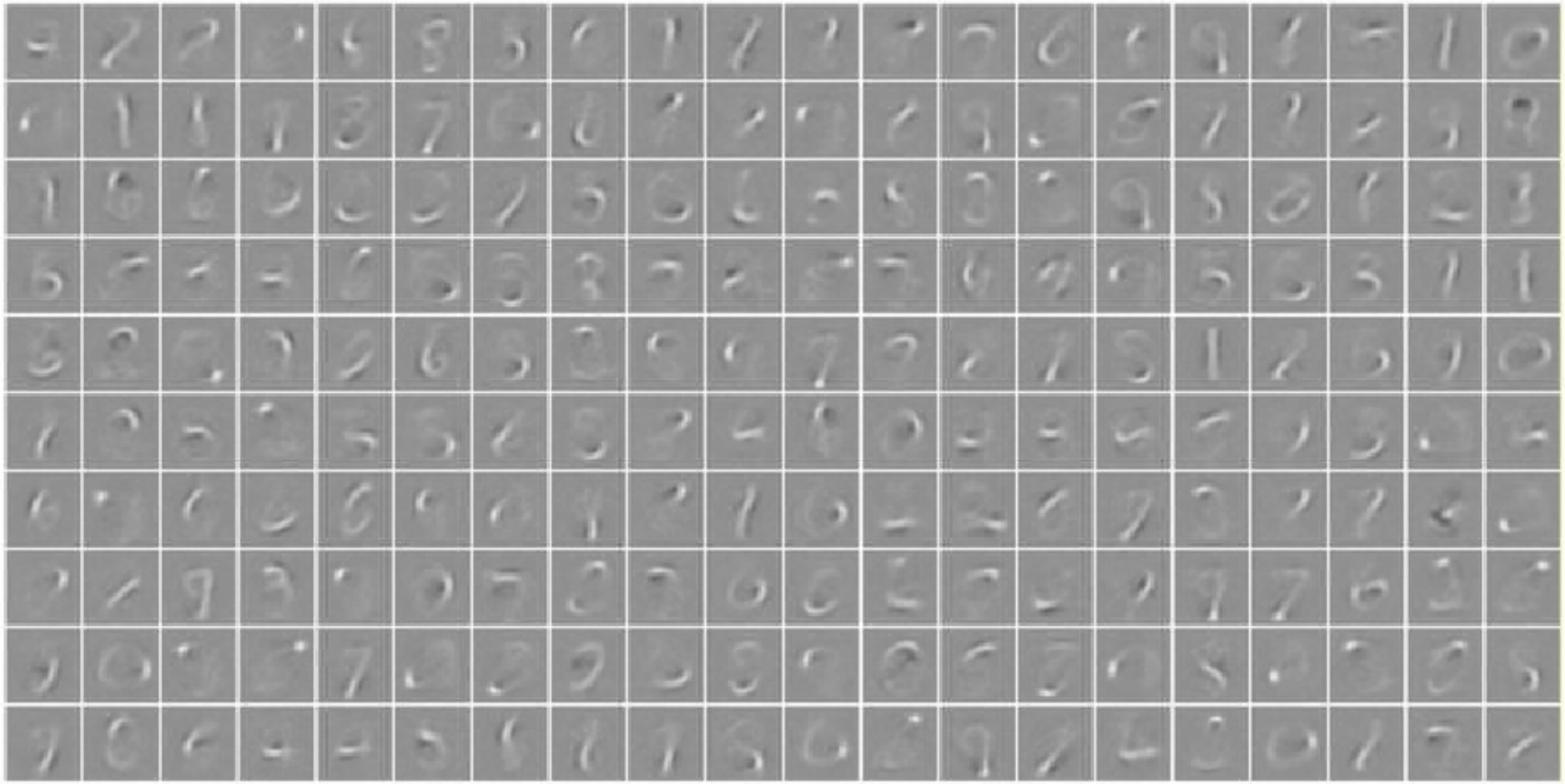
$$g_e(W_e, Y^i) = shrinkage(W_e Y^i)$$



PSD: Basis Functions on MNIST

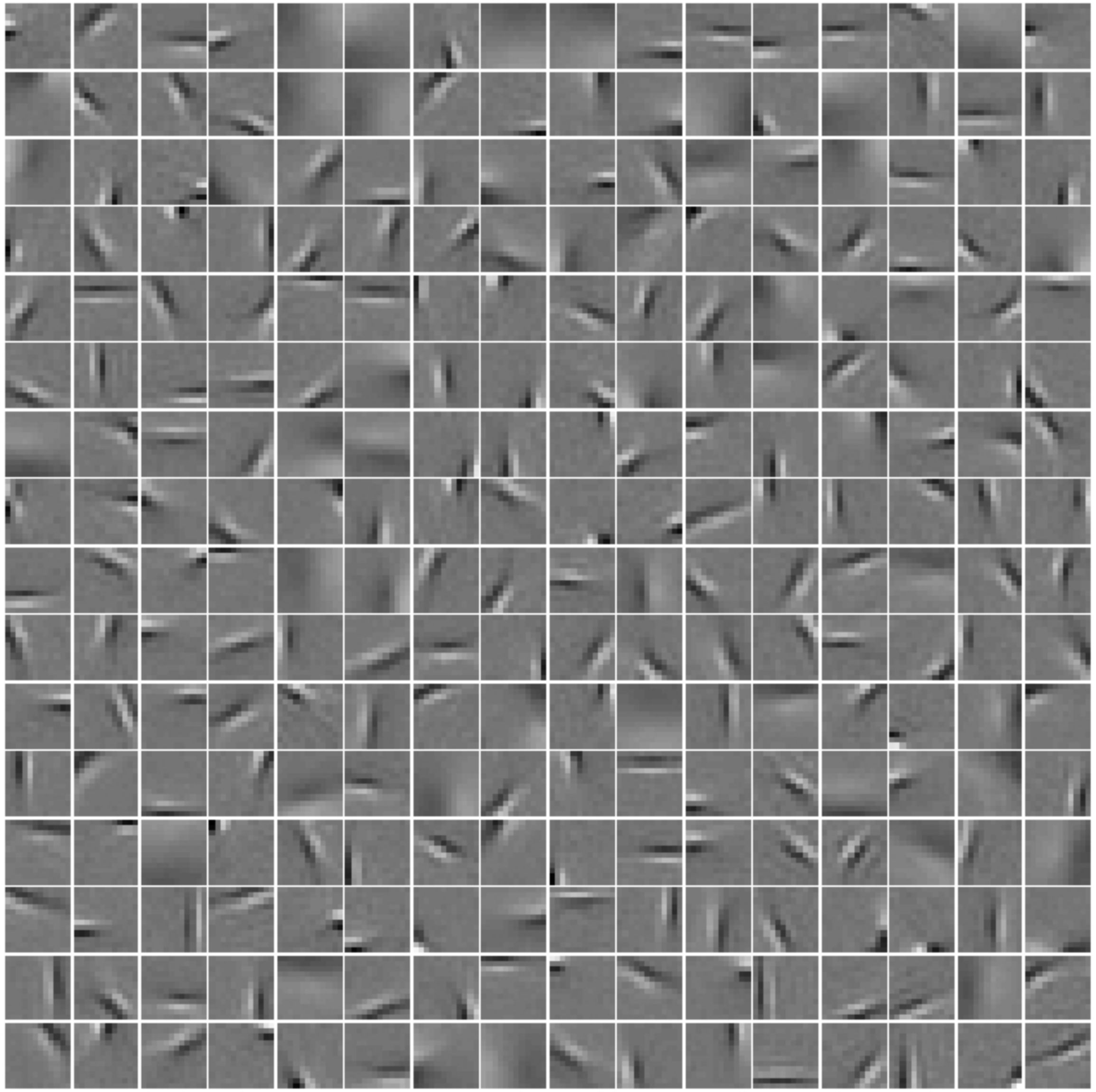
Y LeCun
MA Ranzato

- Basis functions (and encoder matrix) are digit parts



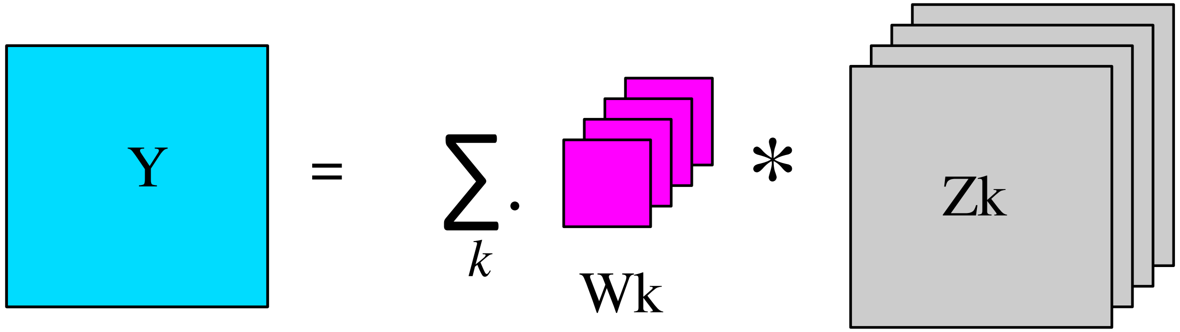
Learned Features on natural patches: V1-like receptive fields

Y LeCun
MA Ranzato



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]

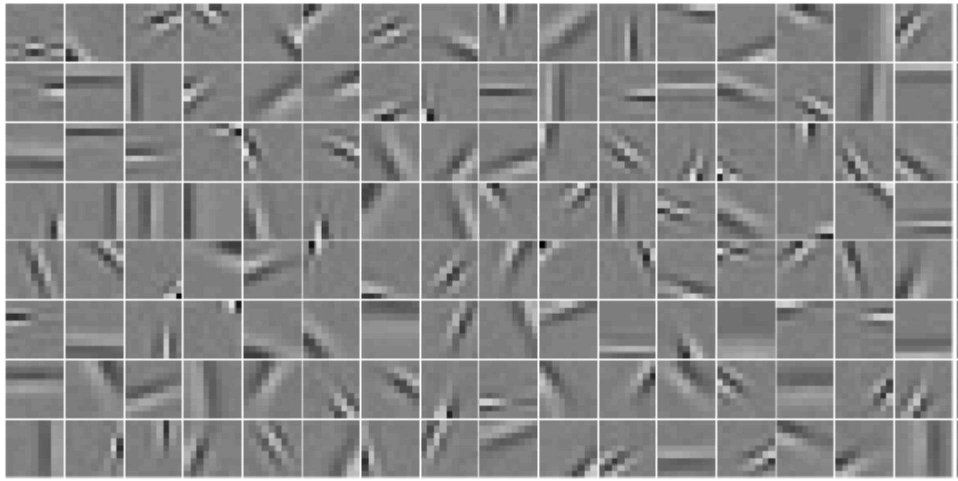
Convolutional PSD: Encoder with a soft sh() Function

Y LeCun
MA Ranzato

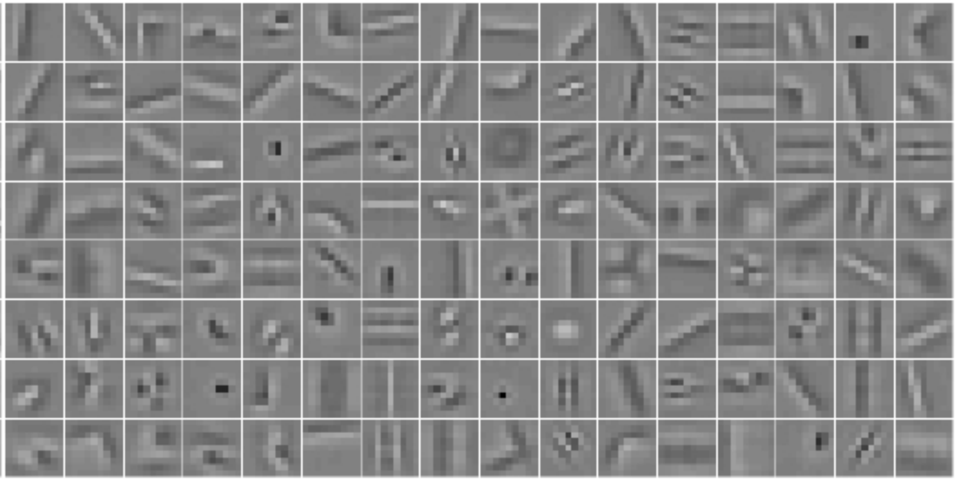
Convolutional Formulation

- 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

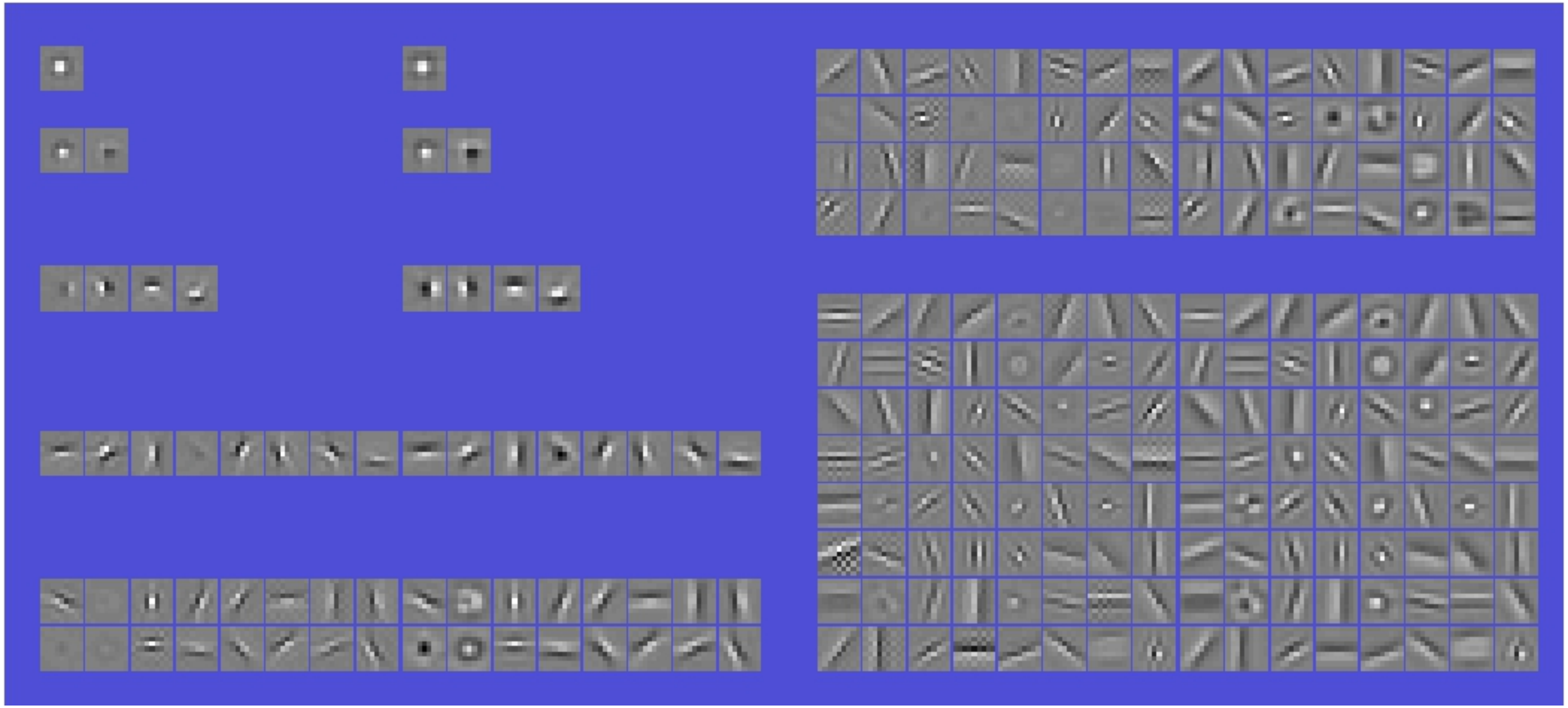


- ***CONVOLUTIONAL*** learning

Convolutional Sparse Auto-Encoder on Natural Images

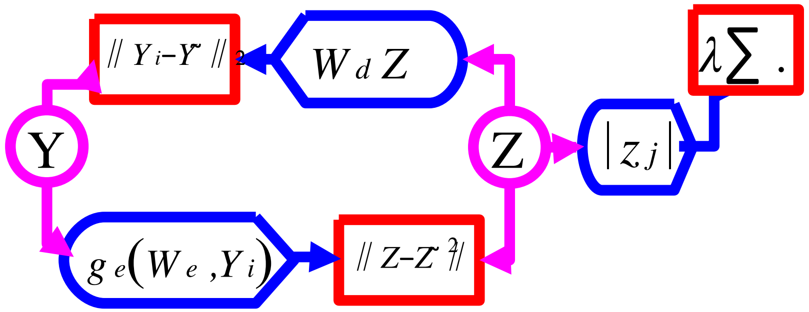
Y LeCun
MA Ranzato

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



Using PSD to Train a Hierarchy of Features

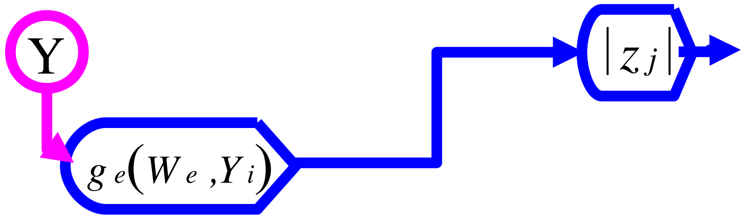
- **Phase 1: train first layer using PSD**



FEATURES

Using PSD to Train a Hierarchy of Features

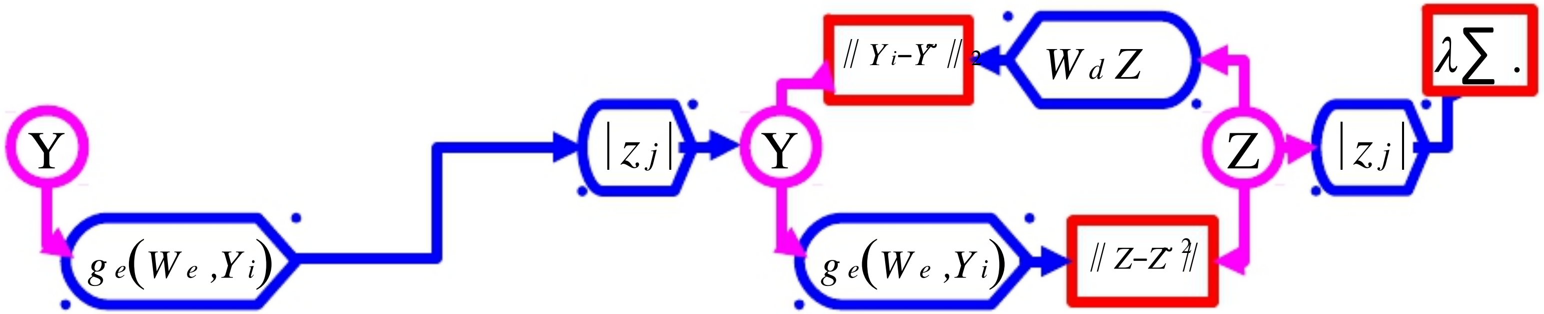
- **Phase 1: train first layer using PSD**
- **Phase 2: use encoder + absolute value as feature extractor**



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

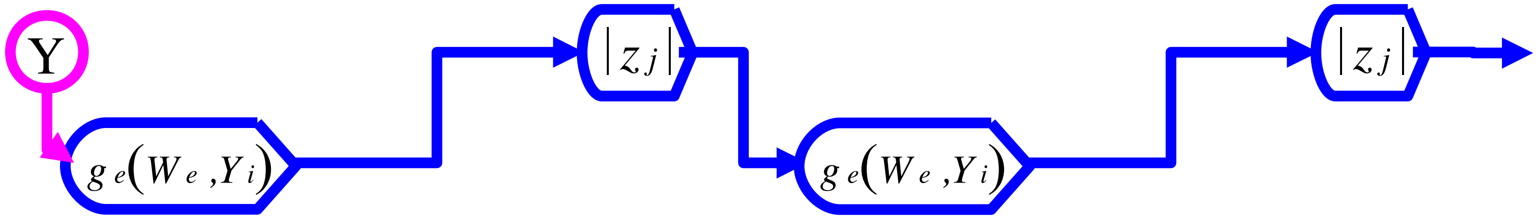


FEATURES

Using PSD to Train a Hierarchy of Features

Y LeCun
MA Ranzato

- **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**

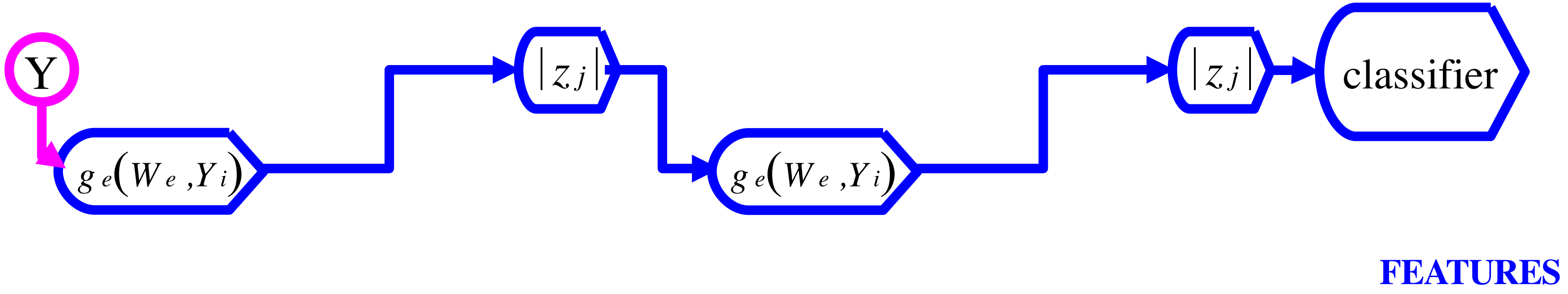


FEATURES

Using PSD to Train a Hierarchy of Features

Y LeCun
MA Ranzato

- 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



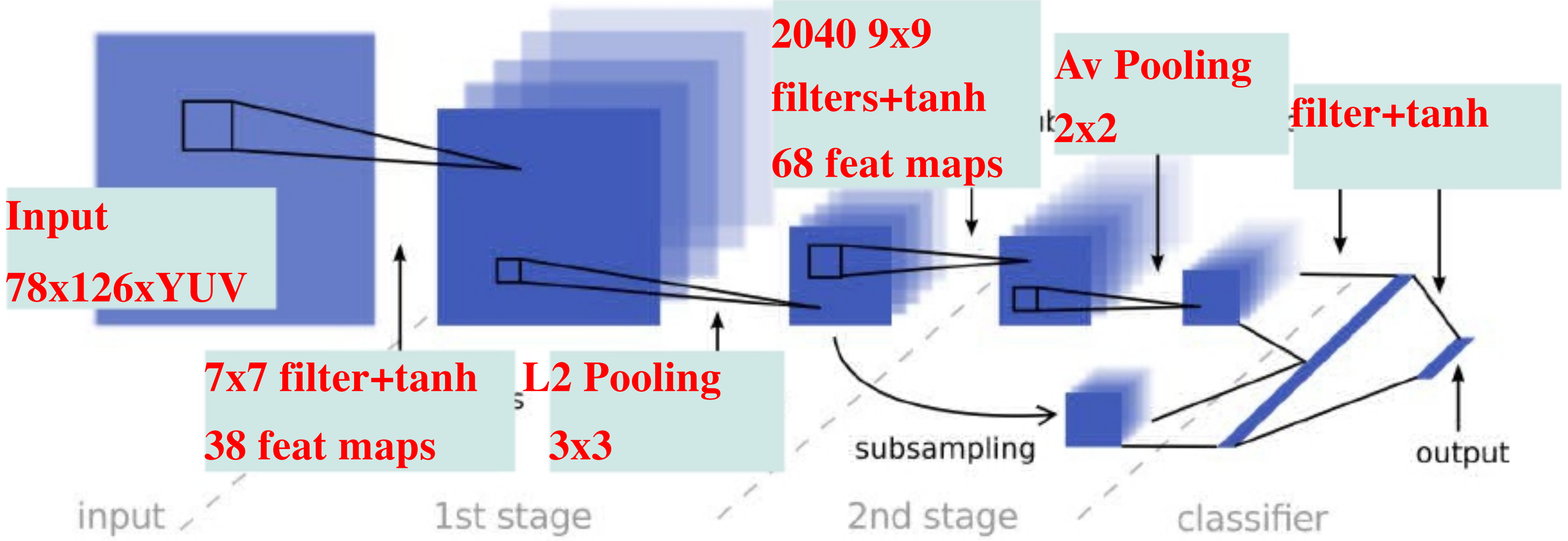
Pedestrian Detection, Face Detection

Y LeCun
MA Ranzato

[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

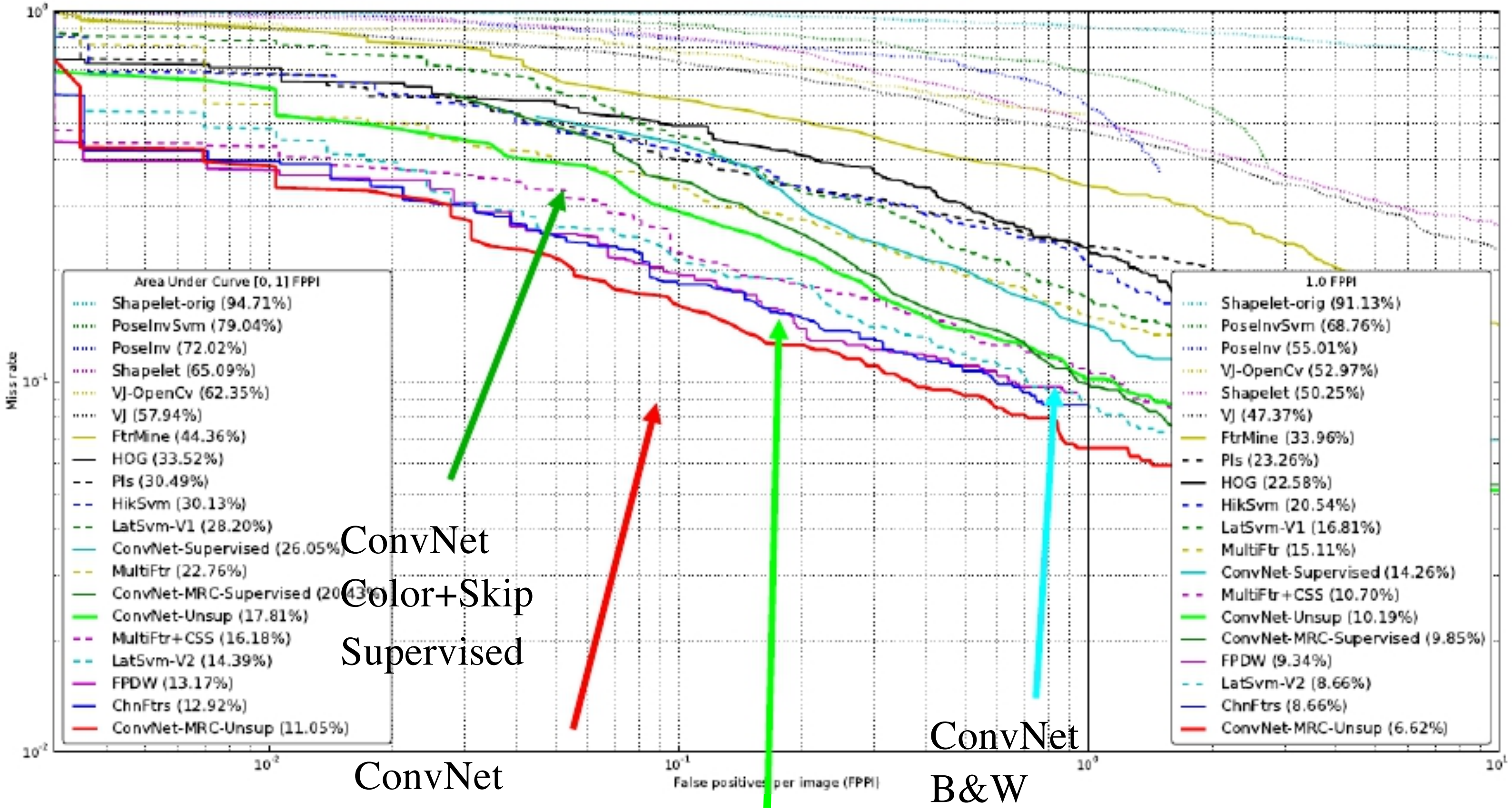


Task	Single-Stage features	Multi-Stage features	Improvement %
Pedestrians detection (INRIA)	14.26%	9.85%	31%
Traffic Signs classification (GTSRB) [33]	1.80%	0.83%	54%
House Numbers classification (SVHN) [32]	5.54%	5.36%	3.2%

[Sermanet, Chintala, LeCun CVPR 2013]

Pedestrian Detection: INRIA Dataset. Miss rate vs false positives

Y LeCun
MA Ranzato



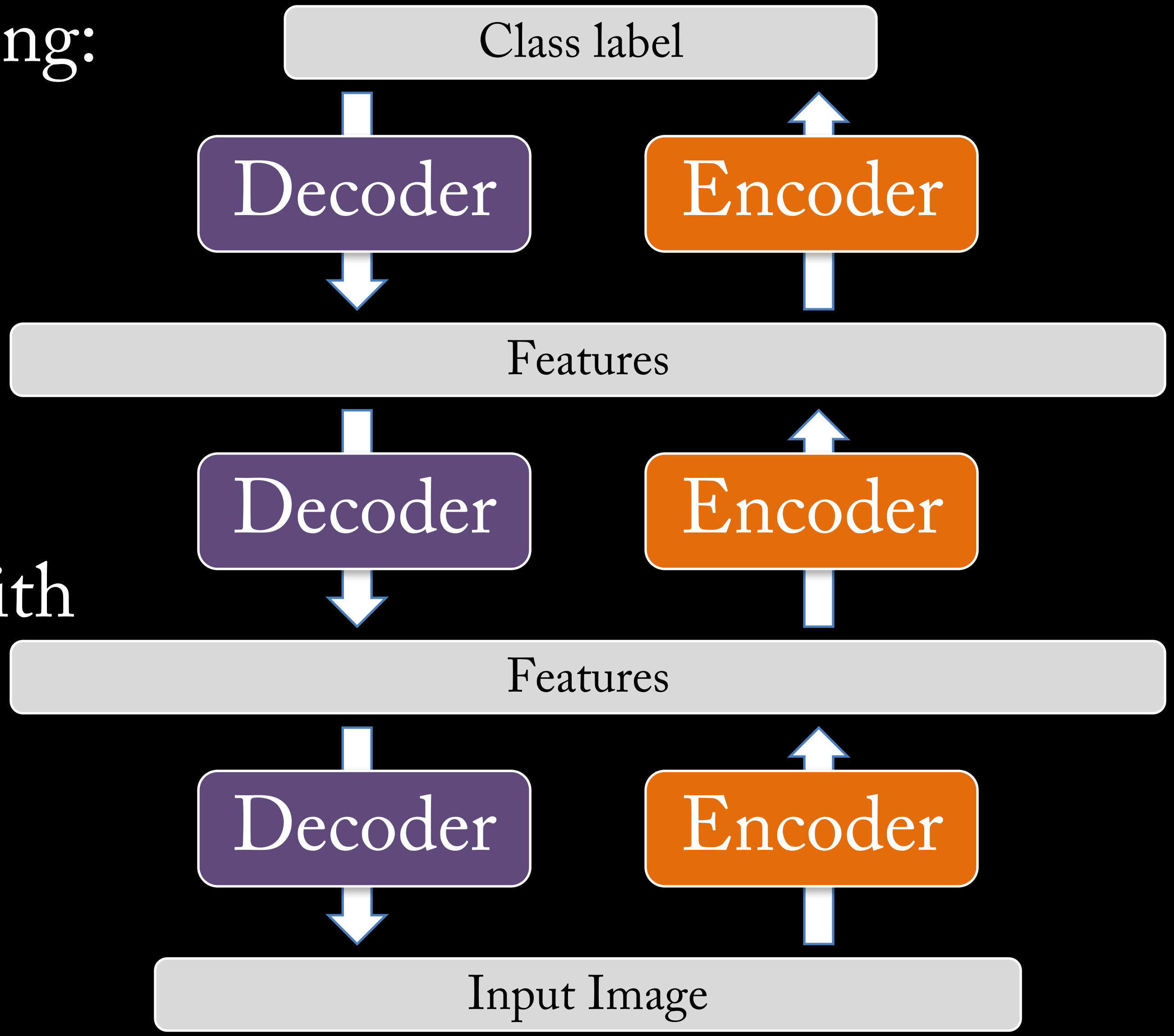
[Kavukcuoglu et al. NIPS 2010] [Sermanet et al. ArXiv 2012]

Stacked Auto-Encoders

Two phase training:

1. Unsupervised layer-wise pre-training
2. Fine-tuning with labeled data

[Hinton & Salakhutdinov Science '06]



Training phase 2: Supervised Fine-Tuning

- Remove decoders
- Use feed-forward path

- Gives standard(Convolutional) Neural Network

- Can fine-tune with backprop

[Hinton & Salakhutdinov Science '06]

Class label

Encoder

Features

Encoder

Features

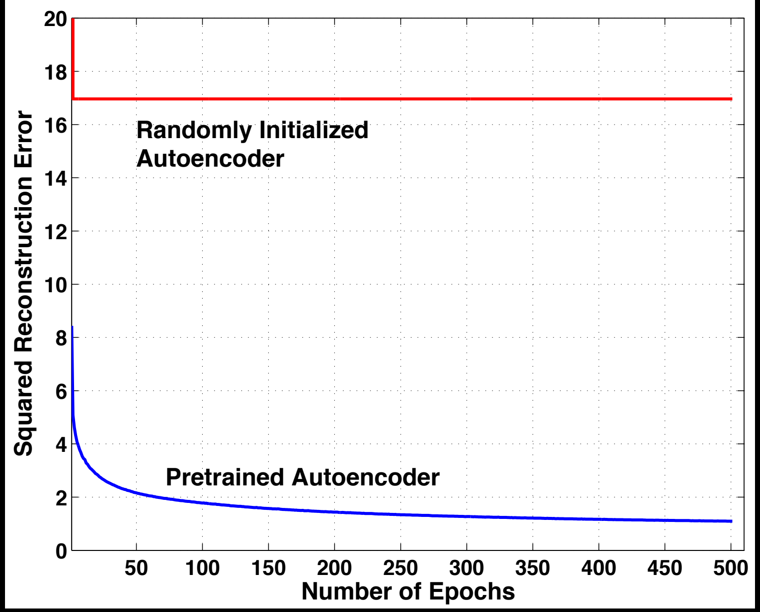
Encoder

Input Image

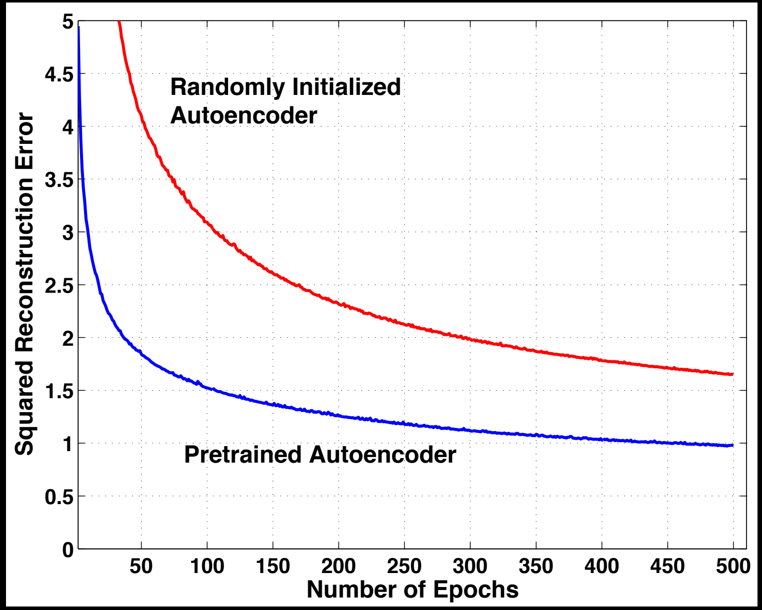
Effects of Pre-Training

- From [Hinton & Salakhudinov, Science 2006]

Big network



Small network



See also: Why Does Unsupervised Pre-training Help Deep Learning?
Dumitru Erhan, Yoshua Bengio ,Aaron Courville, Pierre-Antoine Manzagol PIERRE-Pascal Vincent, Sammy Bengio, JMLR 2010

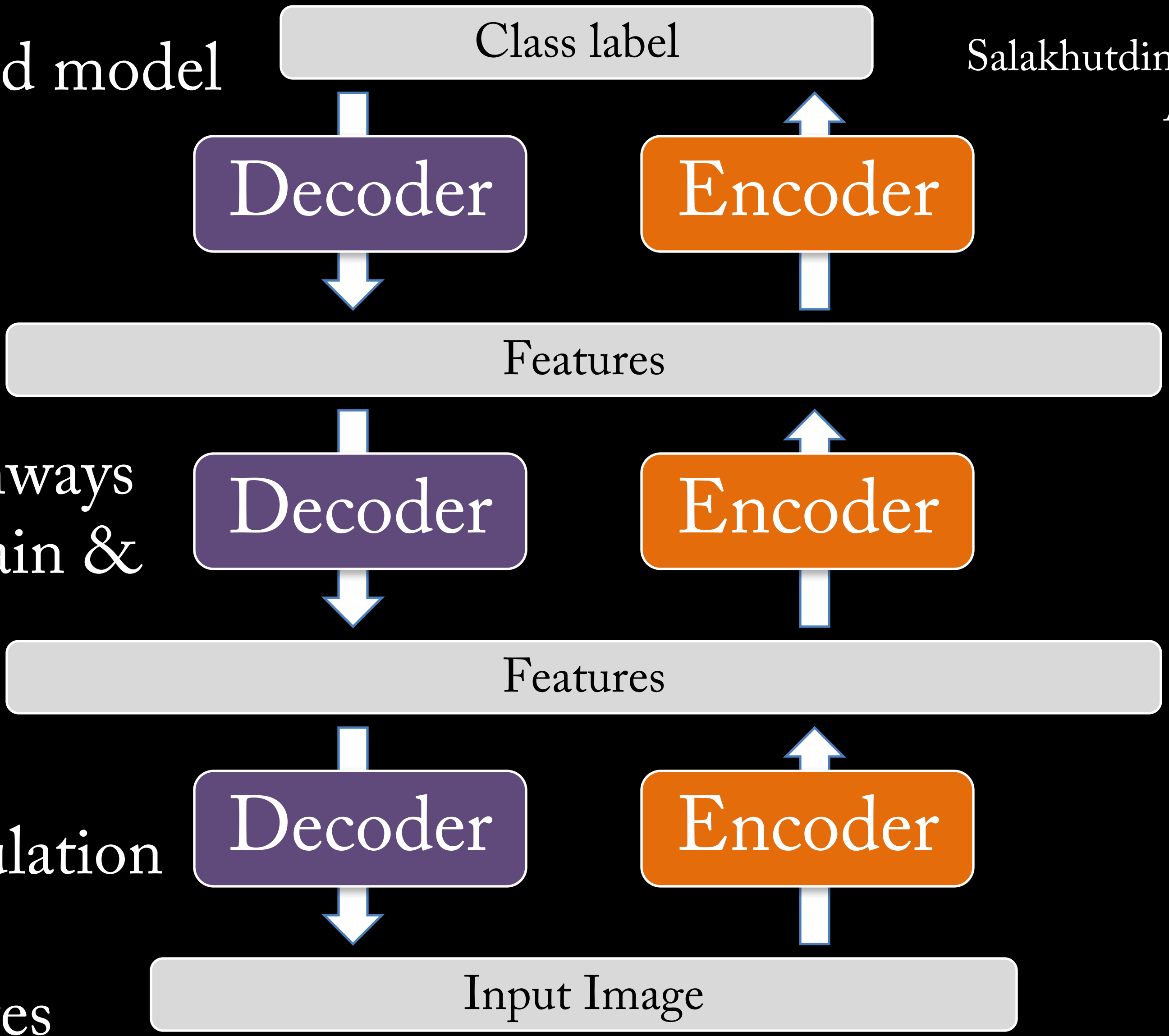
Deep Boltzmann Machines

Undirected model

Salakhutdinov & Hinton
AISTATS'09

Both pathways used at train & test time

TD modulation of BU features



Shape Boltzmann Machine

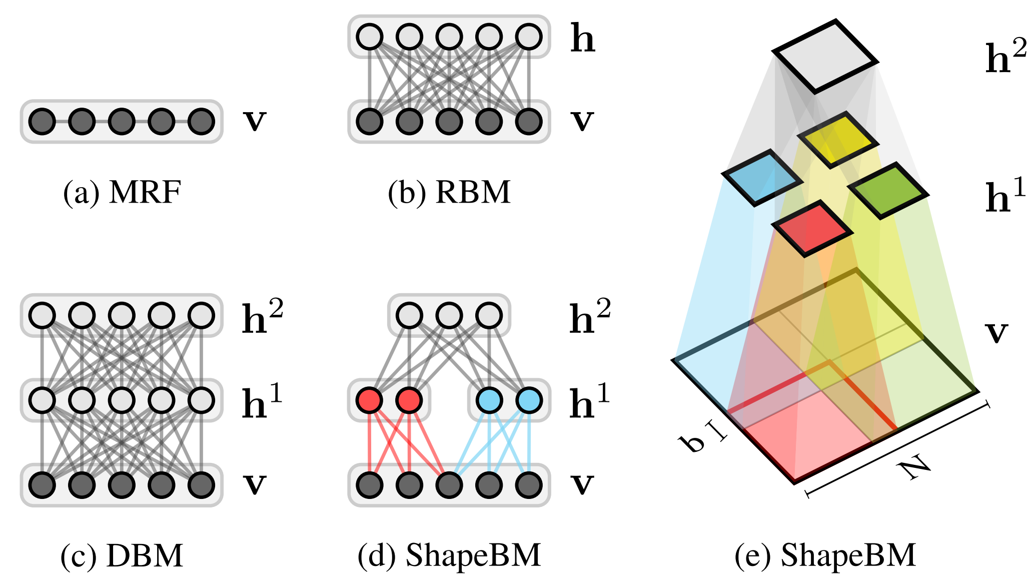
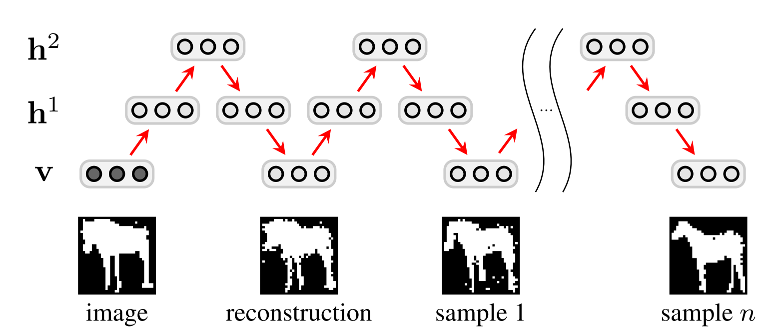
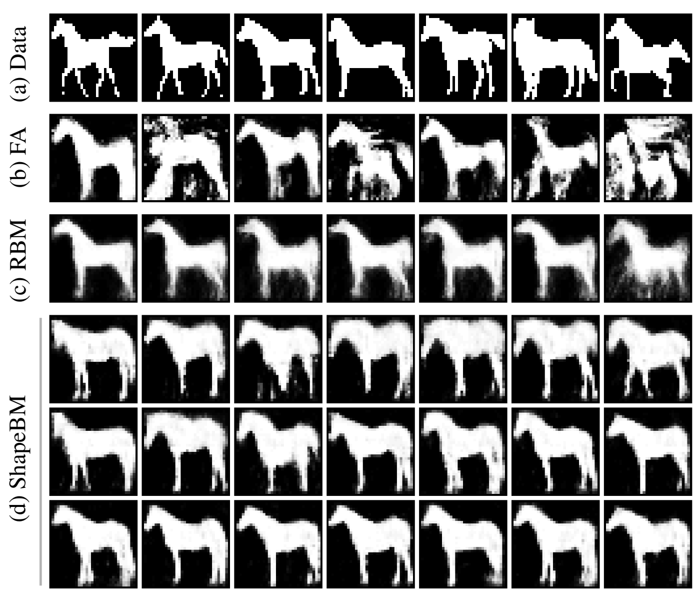


Figure 2. **Undirected models of shape:** (a) 1D slice of a Markov Random Field. (b) Restricted Boltzmann Machine in 1D. (c) Deep Boltzmann Machine in 1D. (d) 1D slice of a Shape Boltzmann Machine. (e) Shape Boltzmann Machine in 2D.





“The Shape Boltzmann Machine: a Strong Model of Object Shape”, Ali Eslami, Nicolas Heess and John Winn, CVPR 2012

Decoder-Only Models

- Examples:
  - Sparse coding
  - Deconvolutional Networks [Zeiler & Fergus, '10]

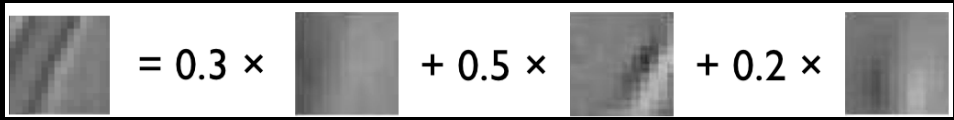
- No encoder to compute features

- So need to perform optimization
  - Can be relatively fast

Sparse Coding (Patch-based)

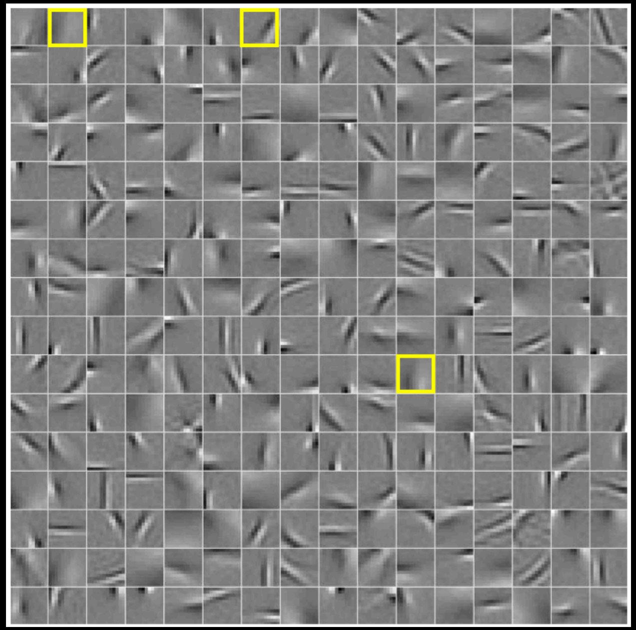
- Over-complete linear decomposition of input y using dictionary D

Input



y

$$C(y,D) = \underset{z}{\operatorname{argmin}} \; \frac{\lambda}{2}\|Dz - y\|_2^2 + |z|_1$$

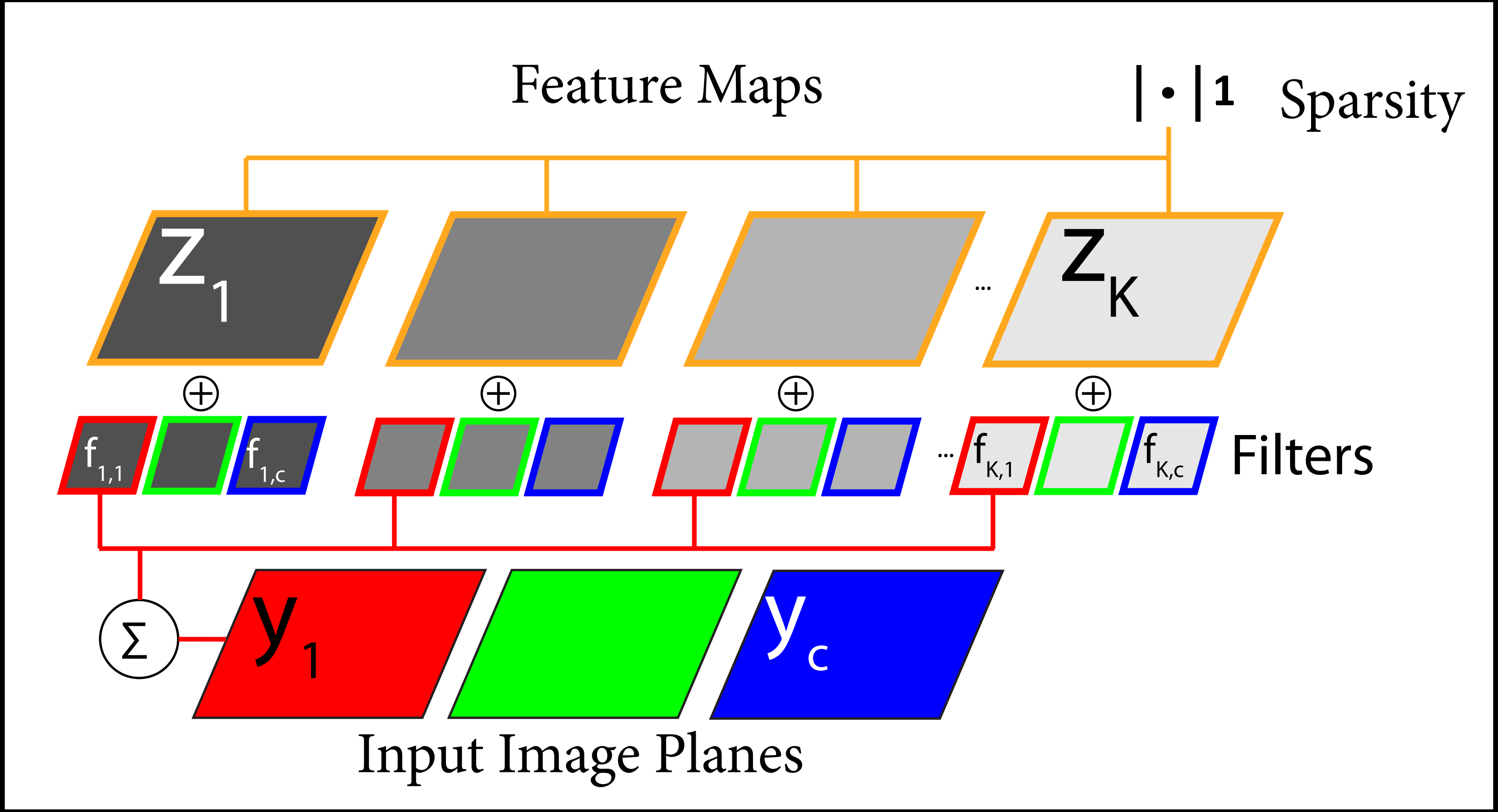


Dictionary D

- ℓ_1 regularization yields solutions with few non-zero elements
- Output is sparse vector: $z = [0, 0.3, 0, \ldots, 0.5, \ldots, 0.2, \ldots, 0]$

Deconvolutional Network Layer

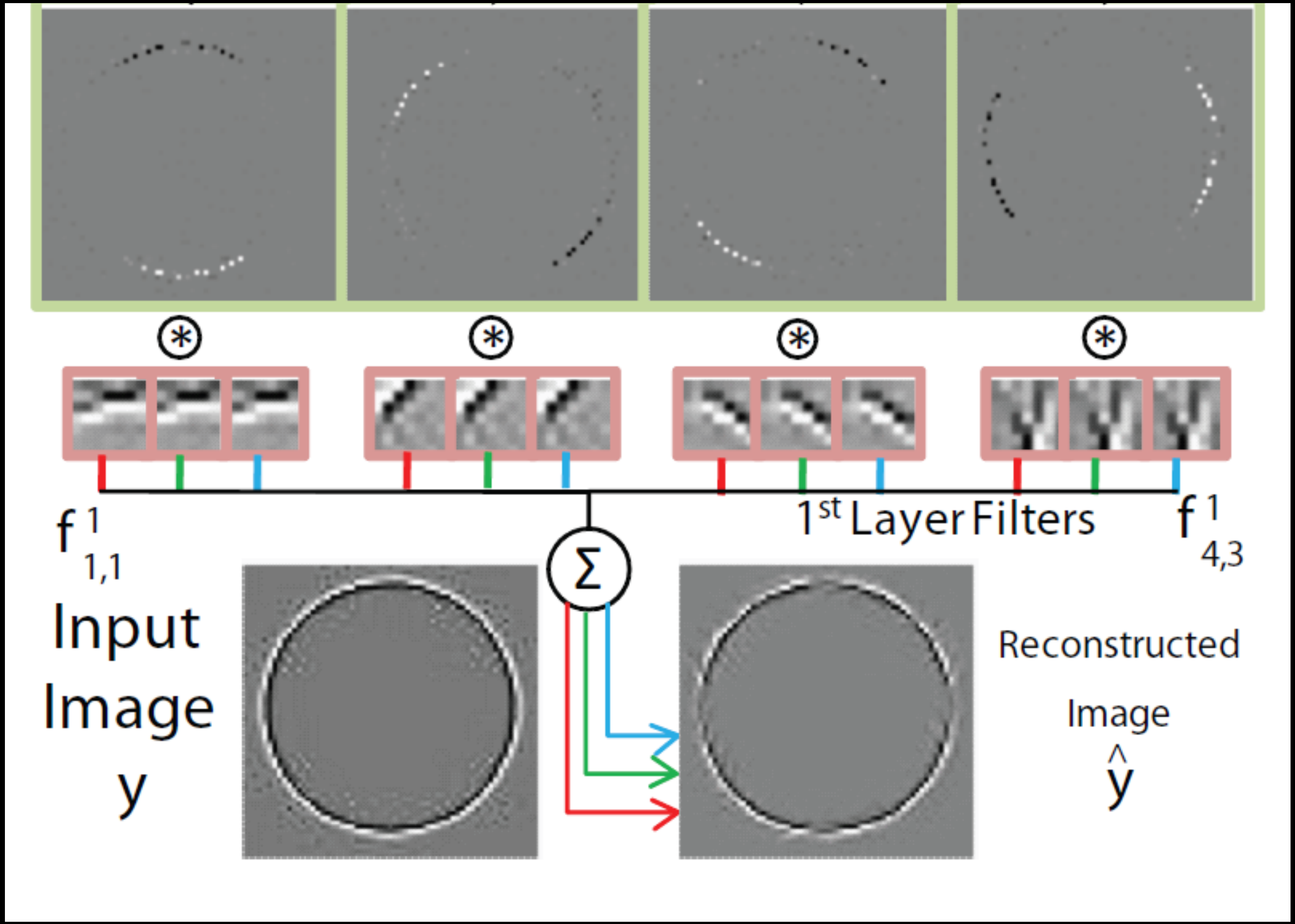
- Convolutional form of sparse coding [Zeiler & Fergus, CVPR 2010]. Also Kavukcuoglu et al. NIPS 2010



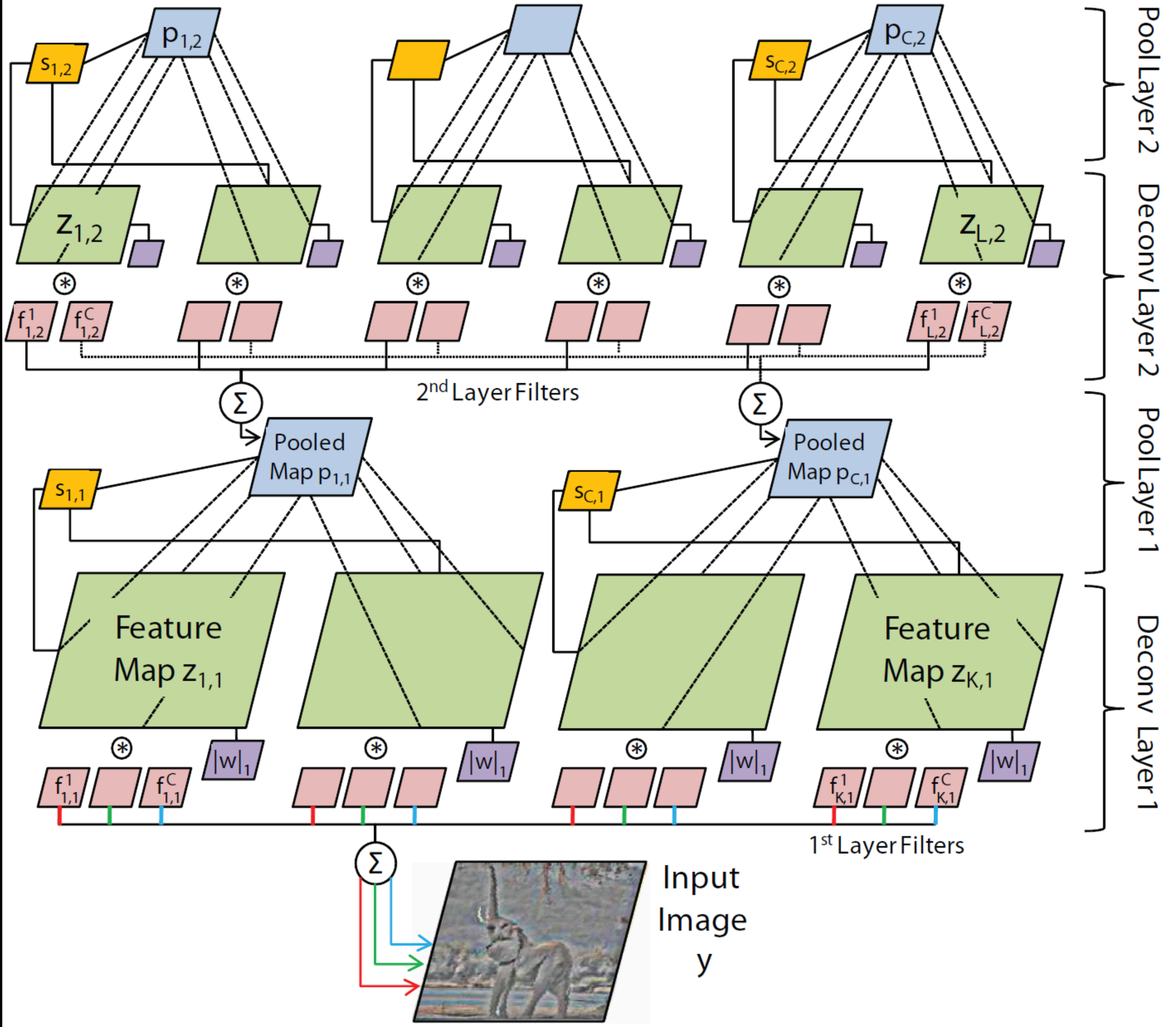
Toy Example

Feature maps

Filters

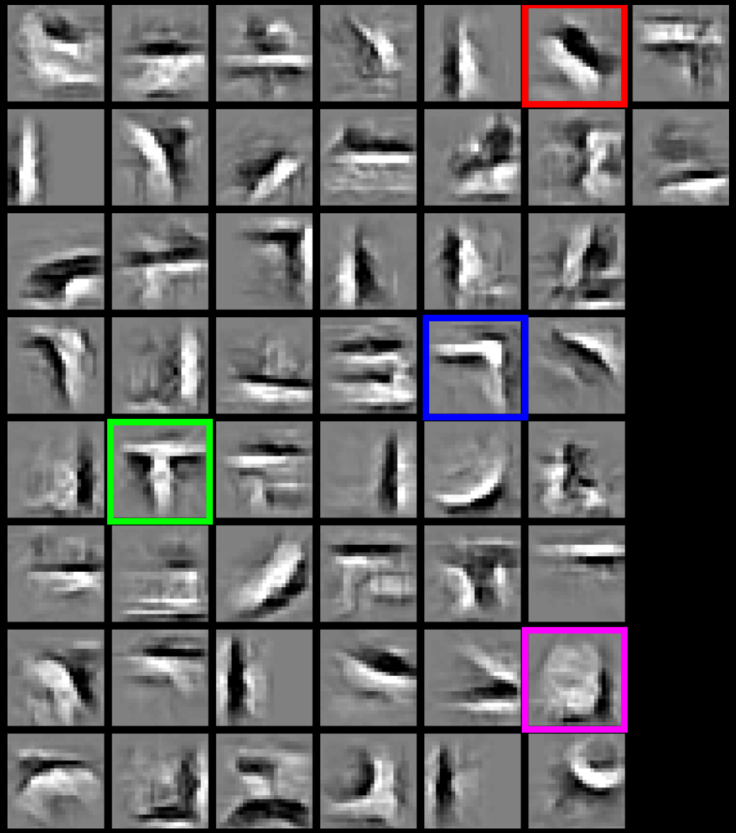


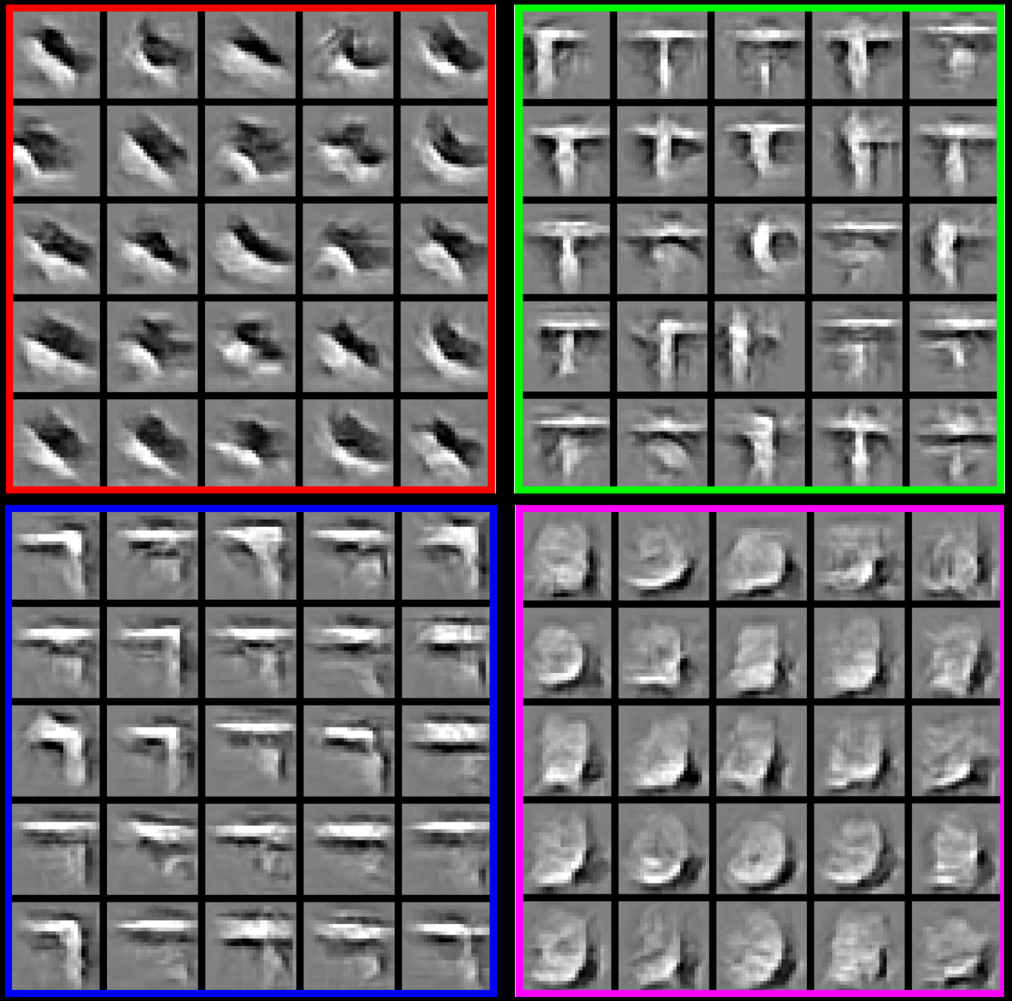
Overall Architecture (2 layers)



Layer 2 Filters

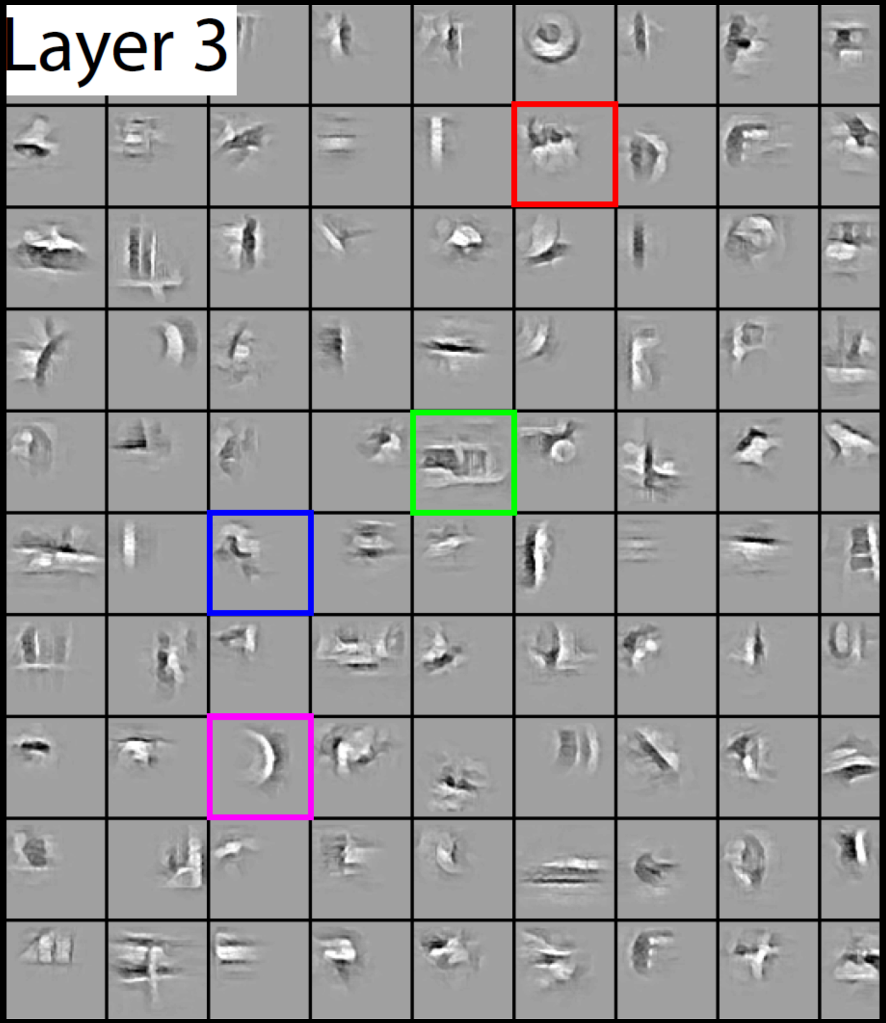
- 50 filters/feature maps, showing max for each map projected down to image

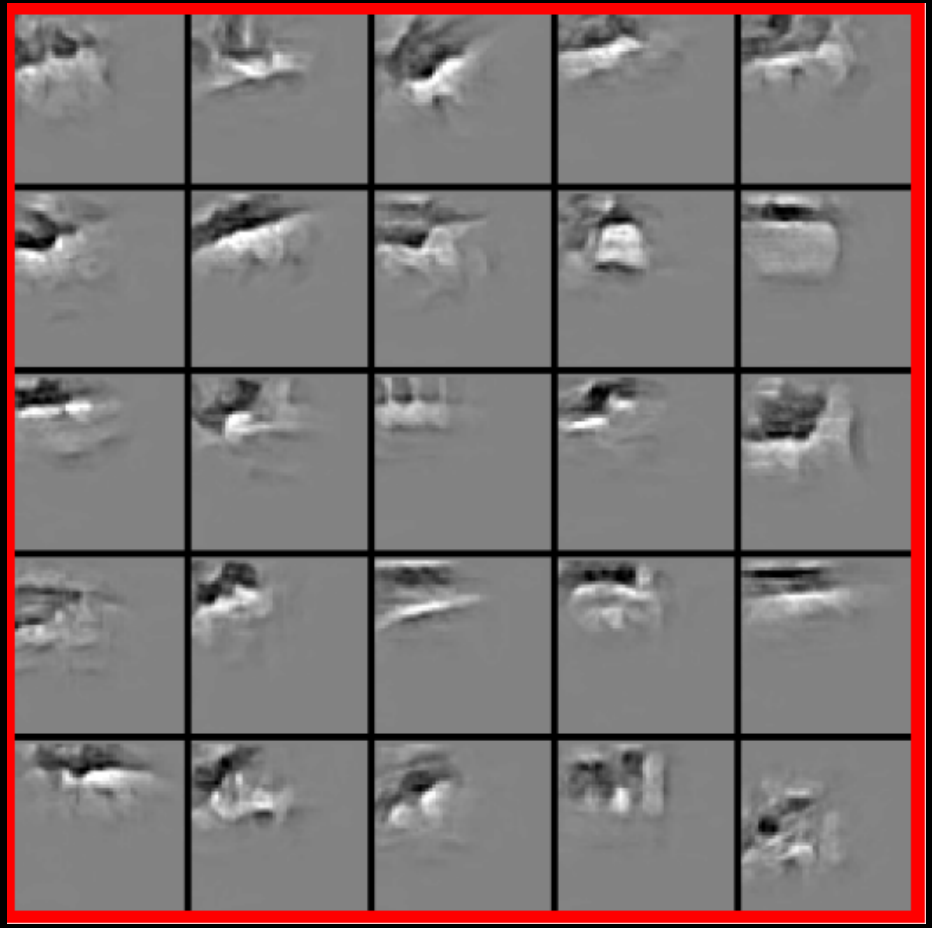




Layer 3 filters

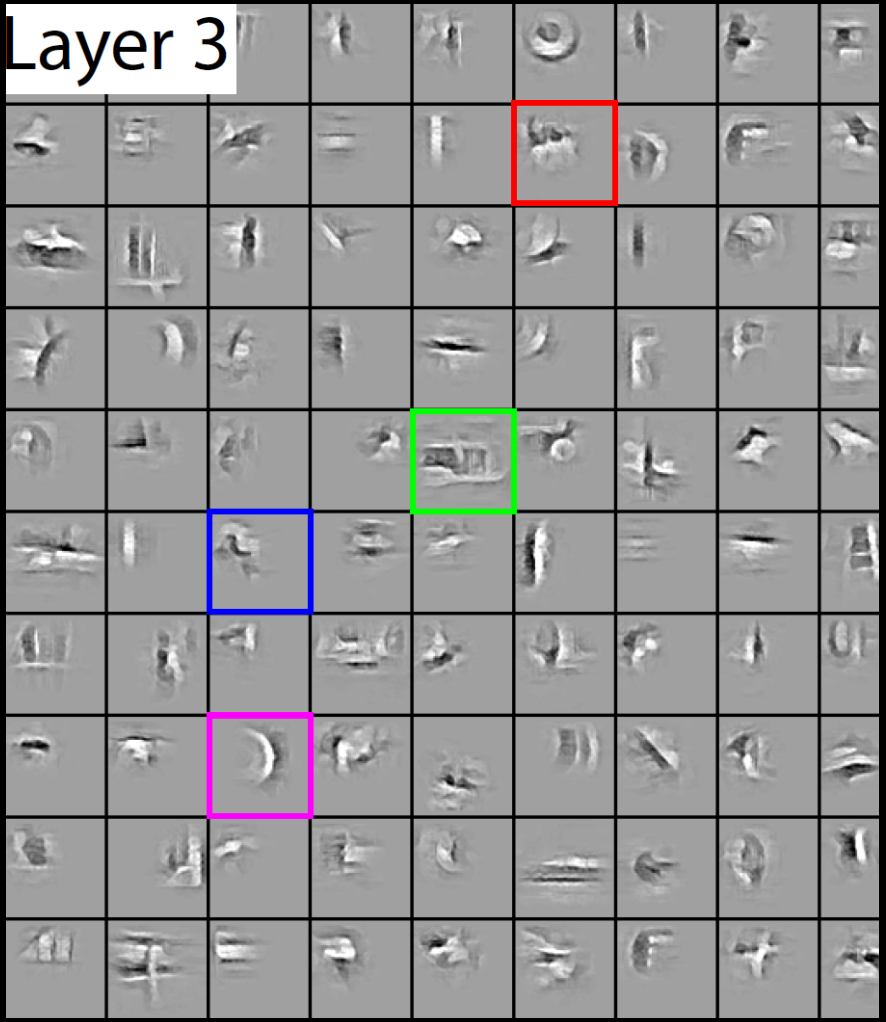
- 100 filters/feature maps, showing max for each map





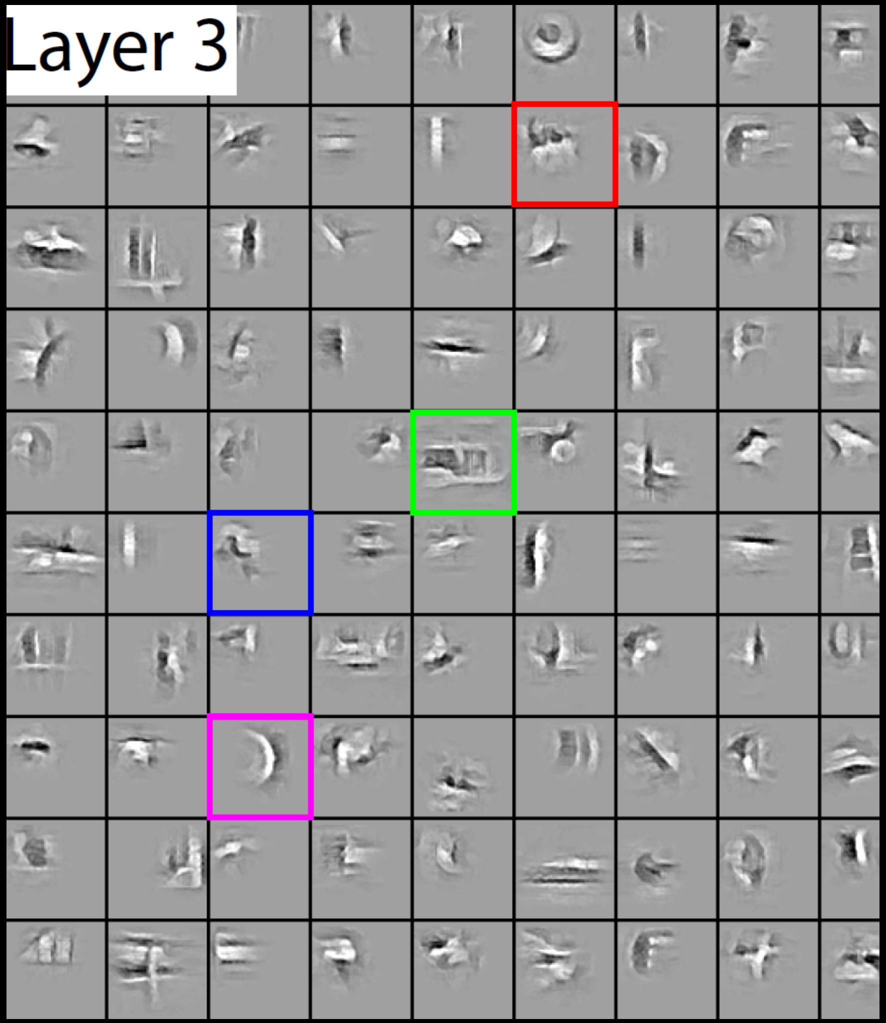
Layer 3 filters

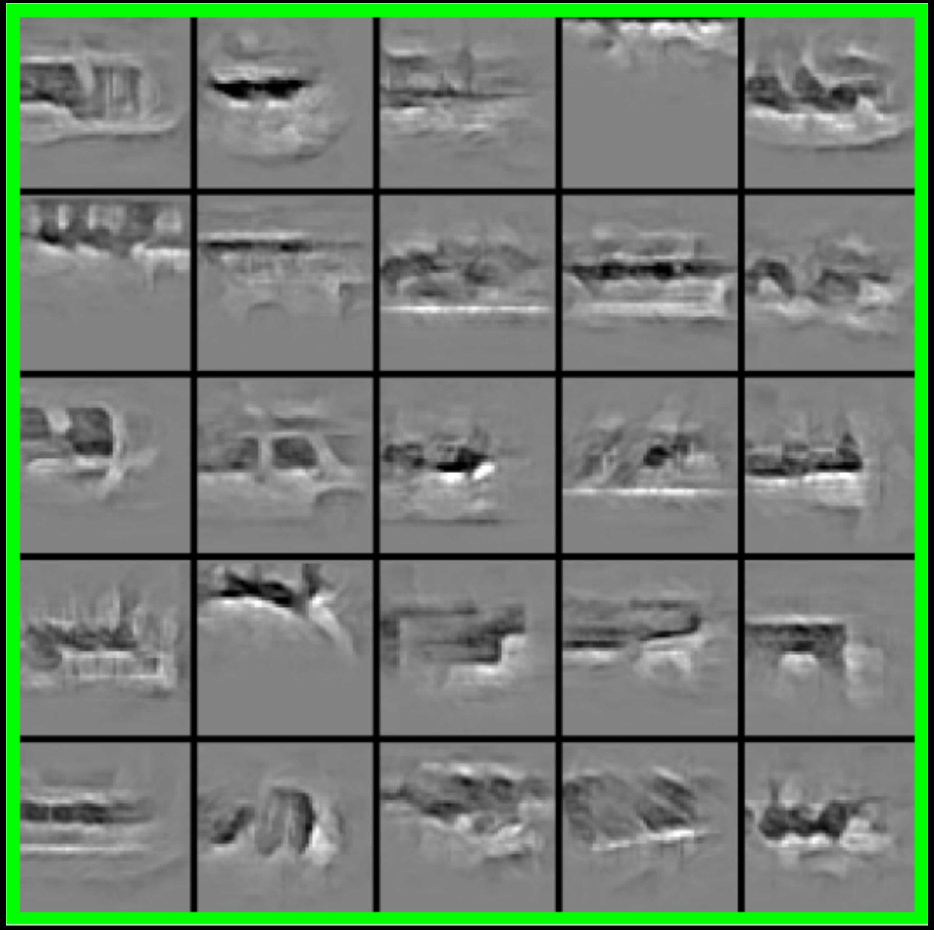
- 100 filters/feature maps, showing max for each map



Layer 3 filters

- 100 filters/feature maps, showing max for each map





Results on Caltech 101

- Comparison to other methods using Lazebnik's SPM with hard vector quantization

Our model - layer 1	66.5%	
Chen *et al.* layer-1+2	$65.7 \pm 0.7\%$	Convolutional Sparse Coding
Kavukcuoglu *et al.*	$65.7 \pm 0.7\%$	
Zeiler *et al.* layer-1+2	$66.9 \pm 1.1\%$	
Boureau *et al.* (Hard)	$70.9 \pm 1.0\%$	Other approaches using SPM with Hard quantization
Jarrett *et al.*	$65.6 \pm 1.0\%$	
Lazebnik *et al.*	$64.6 \pm 0.7\%$	
Lee *et al.* layer-1+2	$65.4 \pm 0.5\%$	

Encoder-Only Models

- In vision setting, essentially a convnet trained without explicit class labels
- But still use feed-forward convnet to predict labels (of some kind)
- What kinds of labels?
  - Need to be "free", i.e. zero or minimal human effort required to obtain
  - Typically exploit some property of images/video
- Often called self-supervised learning
- Note: NOT generic approach -- only valid for image/video domain

Self-Supervised Learning

- Unsupervised feature learning by augmenting single images, Alexey Dosovitskiy, Jost Tobias Springenberg and Thomas Brox, NIPS 2014
- Colorful Image Colorization, Richard Zhang, Philip Isola, Alexei Efros, ECCV 2016
- Unsupervised Visual Representation Learning by Context Prediction, Carl Doersch, Abhinav Gupta, Alexei Efros, ICCV 2015
- Unsupervised Learning of Visual Representations using Videos, Xiaolong Wang, Abhinav Gupta, ICCV 2015

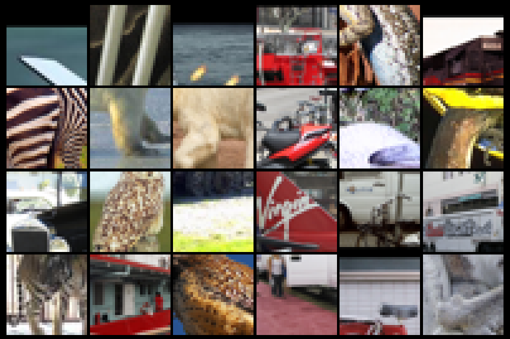
Self-Supervised Learning

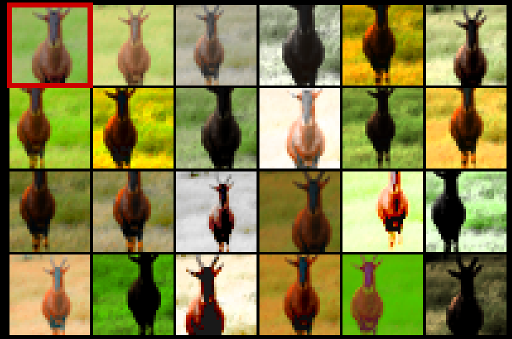
- Unsupervised feature learning by augmenting single images, Alexey Dosovitskiy, Jost Tobias Springenberg and Thomas Brox, NIPS 2014
- Colorful Image Colorization, Richard Zhang, Philip Isola, Alexei Efros, ECCV 2016
- Unsupervised Visual Representation Learning by Context Prediction, Carl Doersch, Abhinav Gupta, Alexei Efros, ICCV 2015
- Unsupervised Learning of Visual Representations using Videos, Xiaolong Wang, Abhinav Gupta, ICCV 2015

Unsupervised Learning of Transformations

[Unsupervised feature learning by augmenting single images, Alexey Dosovitskiy, Jost Tobias Springenberg and Thomas Brox, NIPS 2014]

- Take patches from images
- For each patch, make lots of peturbed versions
- Treat each patch + perturbed copies as a separate classs
- Train supervised convnet
- Introducing prior knowledge about irrelevant transformations via perturbations

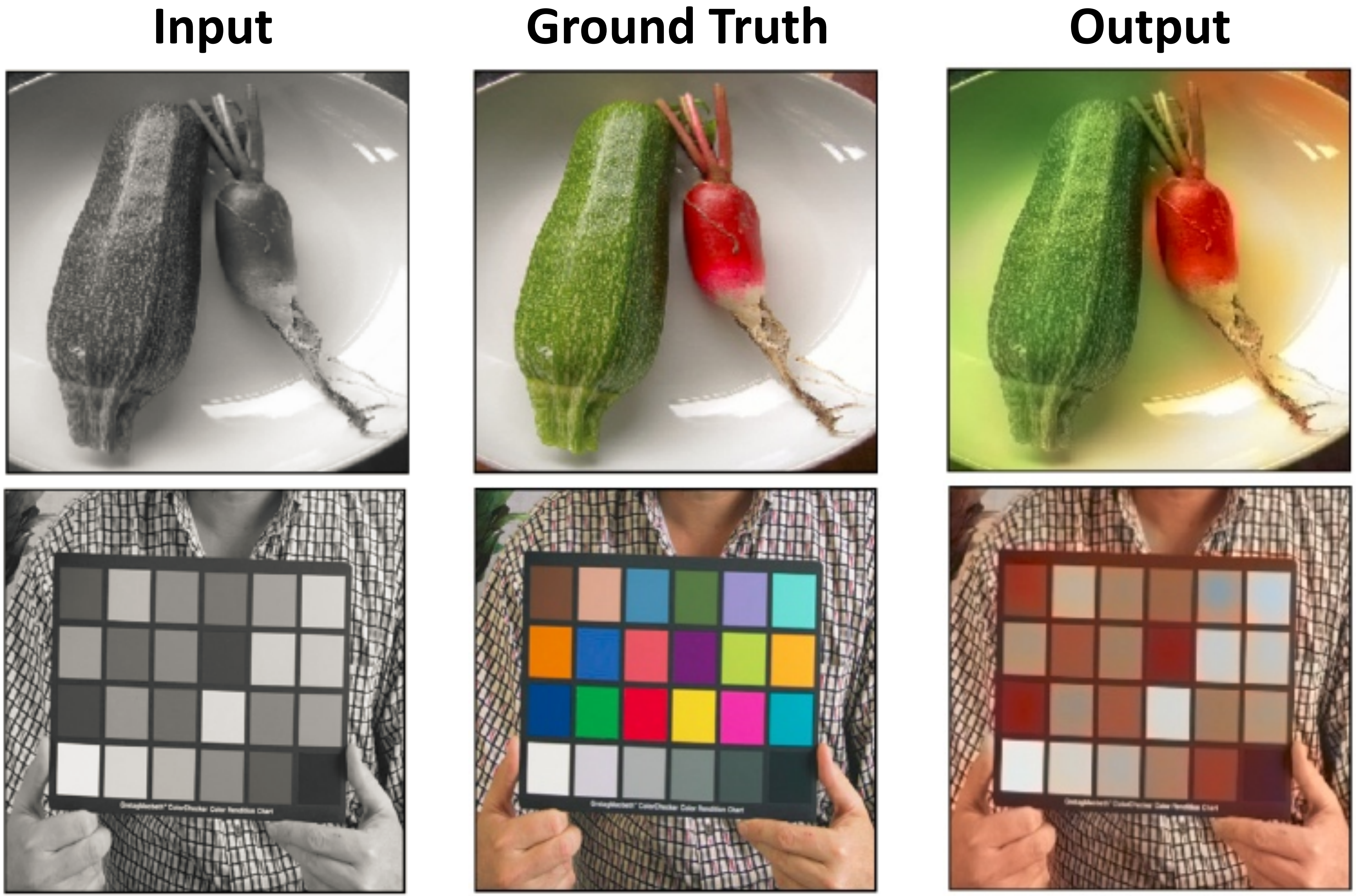




	STL-10	CIFAR-10-reduced	CIFAR-10	Caltech-101
K-means [6]	60.1 ± 1	70.7 ± 0.7	82.0	—
Multi-way local pooling [5]	—	—	—	77.3 ± 0.6
Slowness on videos [25]	61.0	—	—	74.6
Receptive field learning [16]	—	—	[83.11][1]	75.3 ± 0.7
Hierarchical Matching Pursuit (HMP) [3]	64.5 ± 1	—	—	—
Multipath HMP [4]	—	—	—	$\mathbf{82.5 \pm 0.5}$
Sum-Product Networks [8]	62.3 ± 1	—	[83.96][1]	—
View-Invariant K-means [15]	63.7	$\mathbf{72.6 \pm 0.7}$	**81.9**	—
This paper	$\mathbf{67.4 \pm 0.6}$	69.3 ± 0.4	77.5	76.6 ± 0.7 [2]

Self-Supervised Learning

- Unsupervised feature learning by augmenting single images, Alexey Dosovitskiy, Jost Tobias Springenberg and Thomas Brox, NIPS 2014
- Colorful Image Colorization, Richard Zhang, Philip Isola, Alexei Efros, ECCV 2016
- Unsupervised Visual Representation Learning by Context Prediction, Carl Doersch, Abhinav Gupta, Alexei Efros, ICCV 2015
- Unsupervised Learning of Visual Representations using Videos, Xiaolong Wang, Abhinav Gupta, ICCV 2015


Input
Ground Truth
Output

Predicting Labels from Data

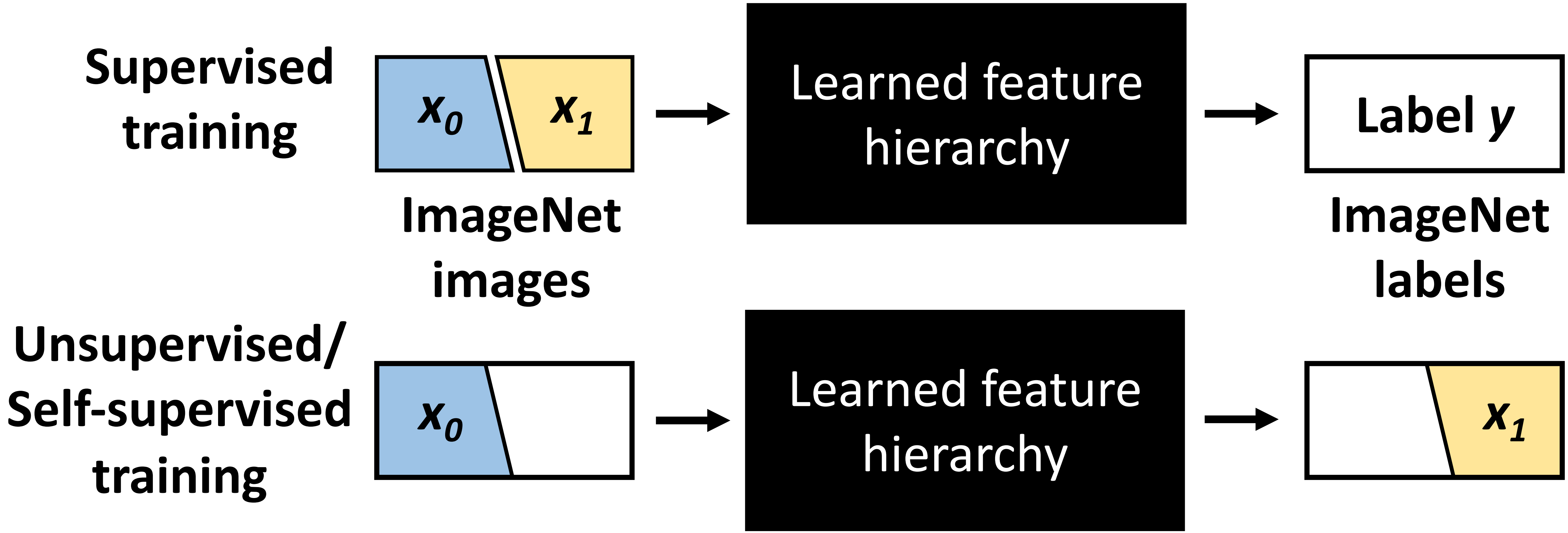
Supervised training

Data *x* → Learned feature hierarchy → **Label *y***

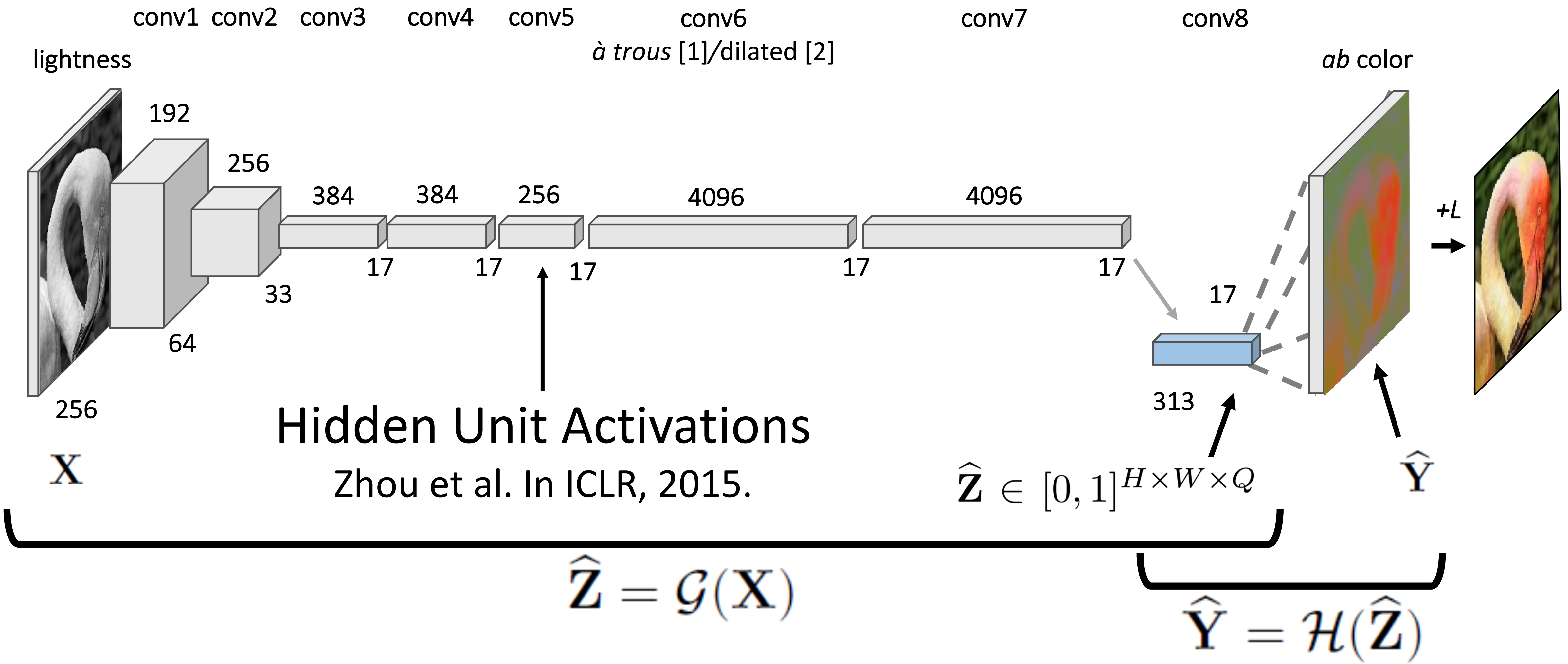
ImageNet images

ImageNet labels

Predicting Data from Data

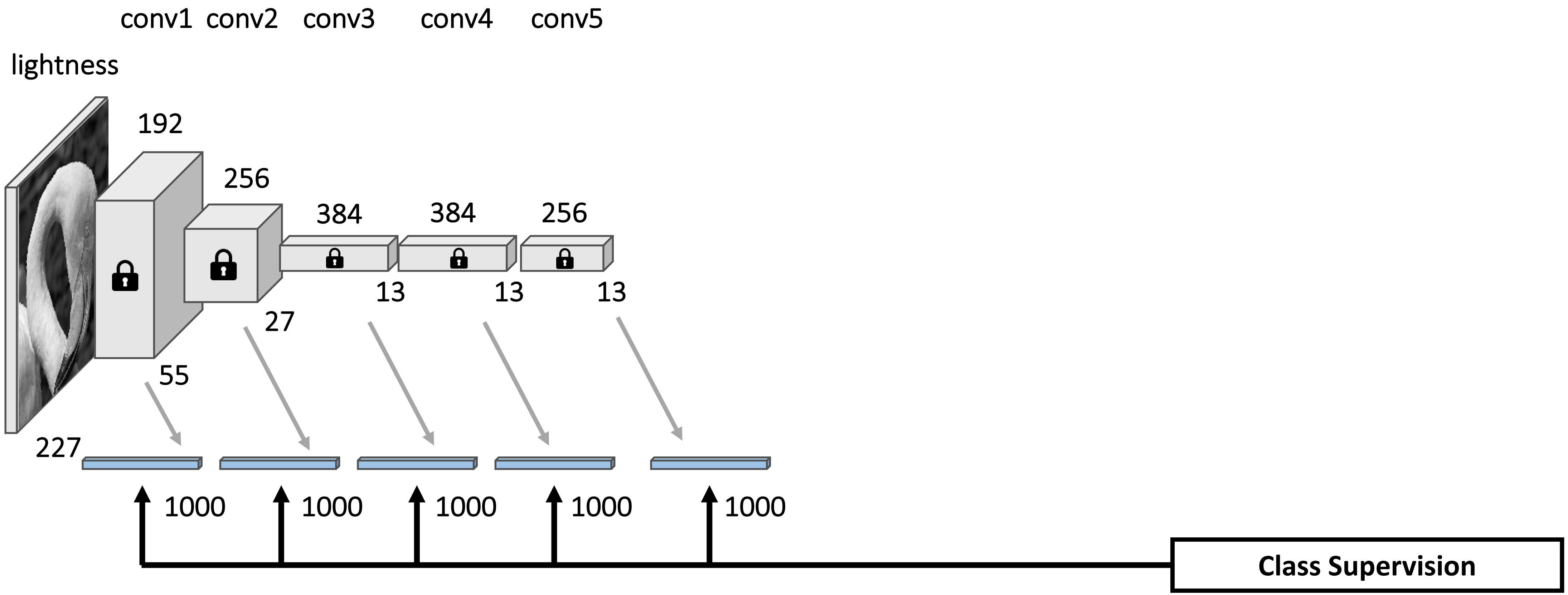


Cross-Channel Encoder



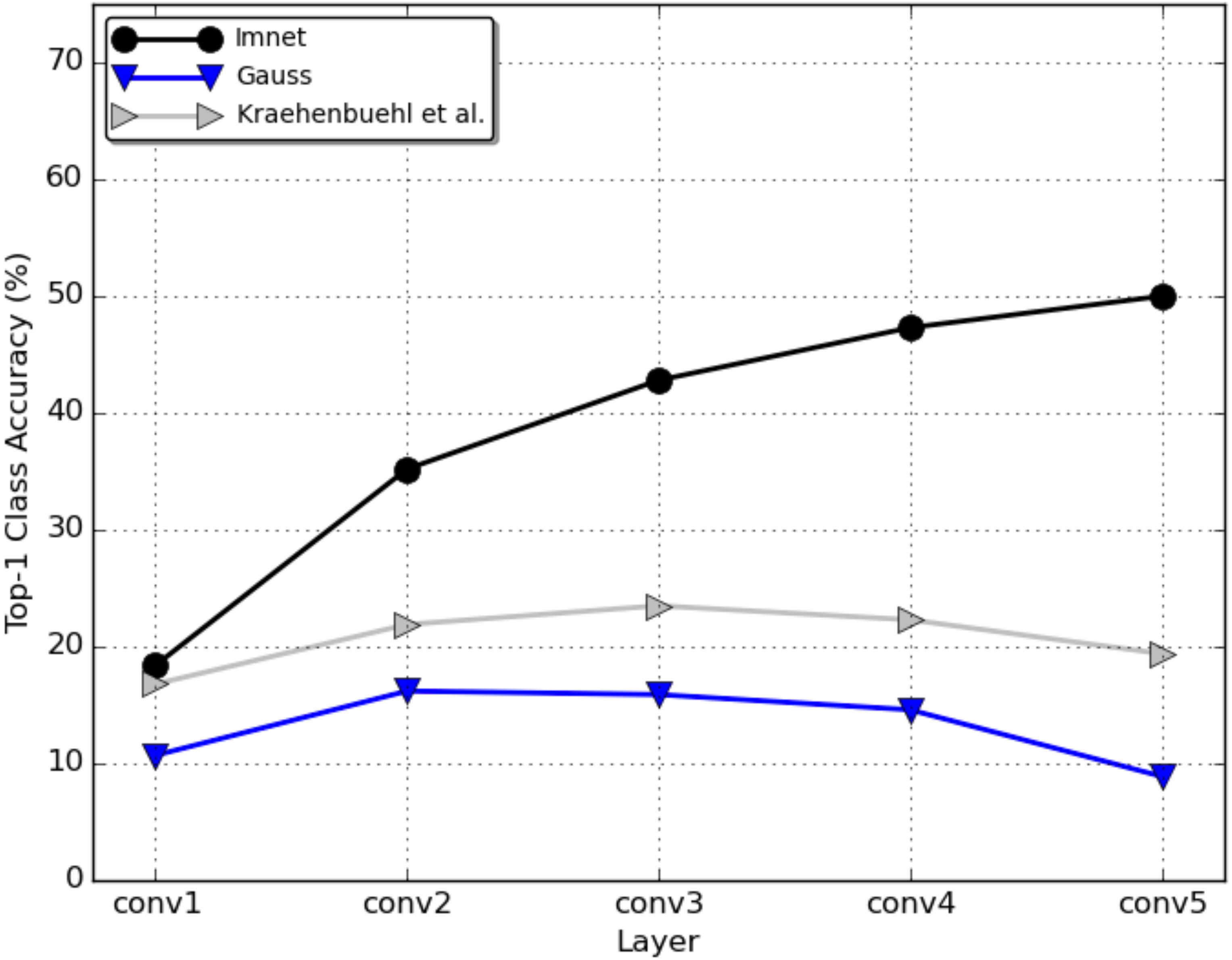
[1] Chen *et al.* In arXiv, 2016.
[2] Yu and Koltun. In ICLR, 2016

Task Generalization: ILSVRC linear classification

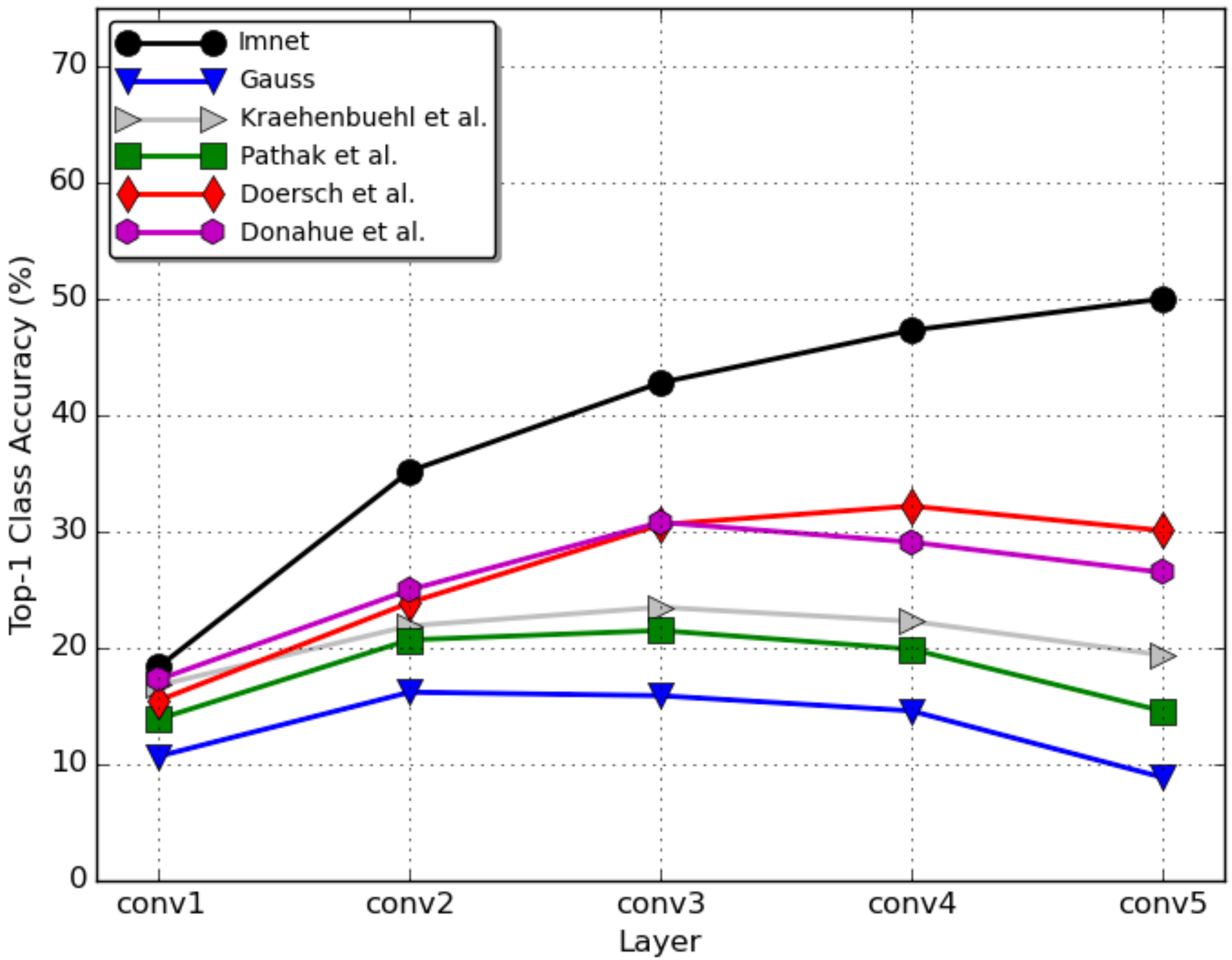


Are semantic classes ***linearly separable***
in the learned feature space?

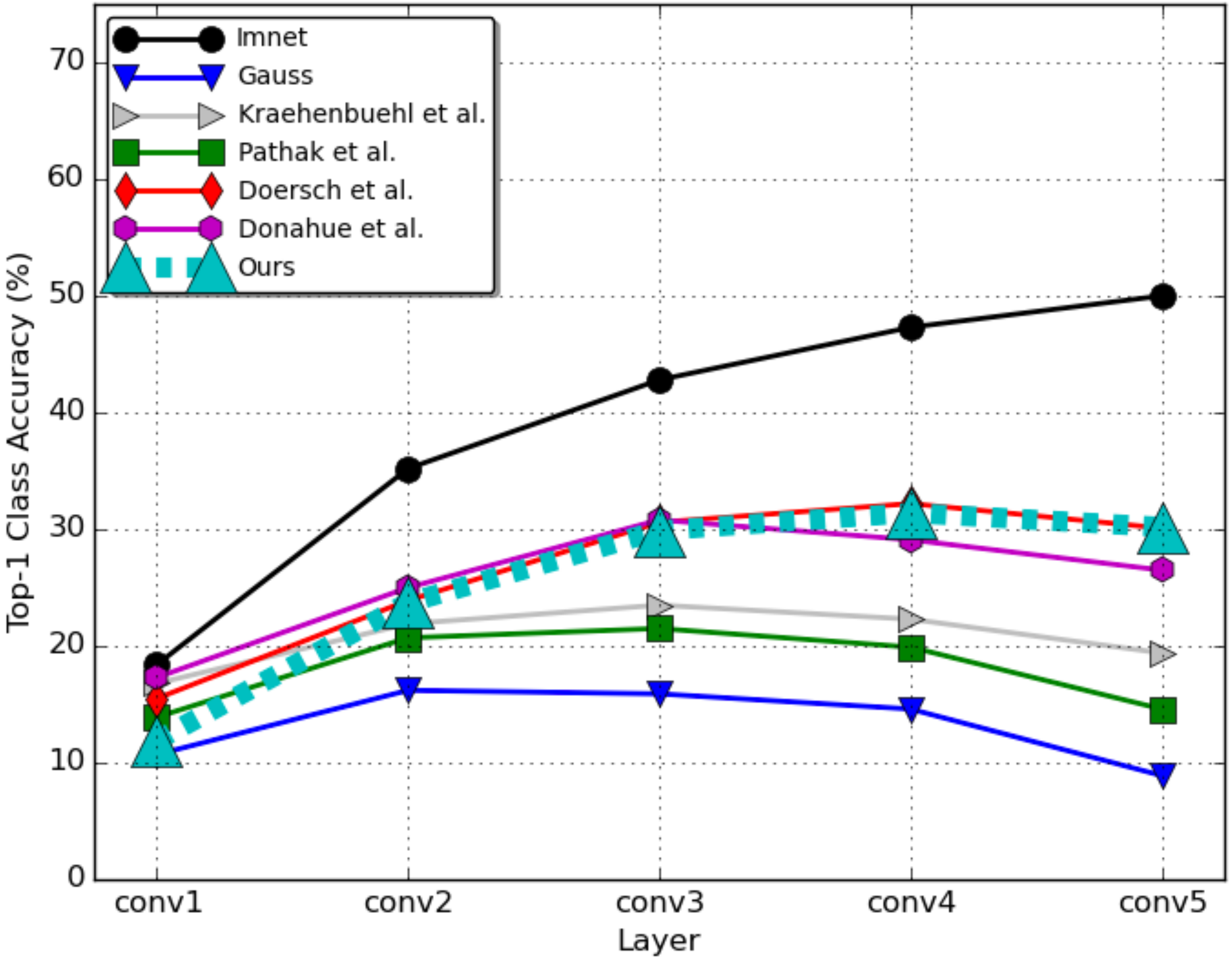
Task Generalization: ILSVRC linear classification



Task Generalization: ILSVRC linear classification



Task Generalization: ILSVRC linear classification

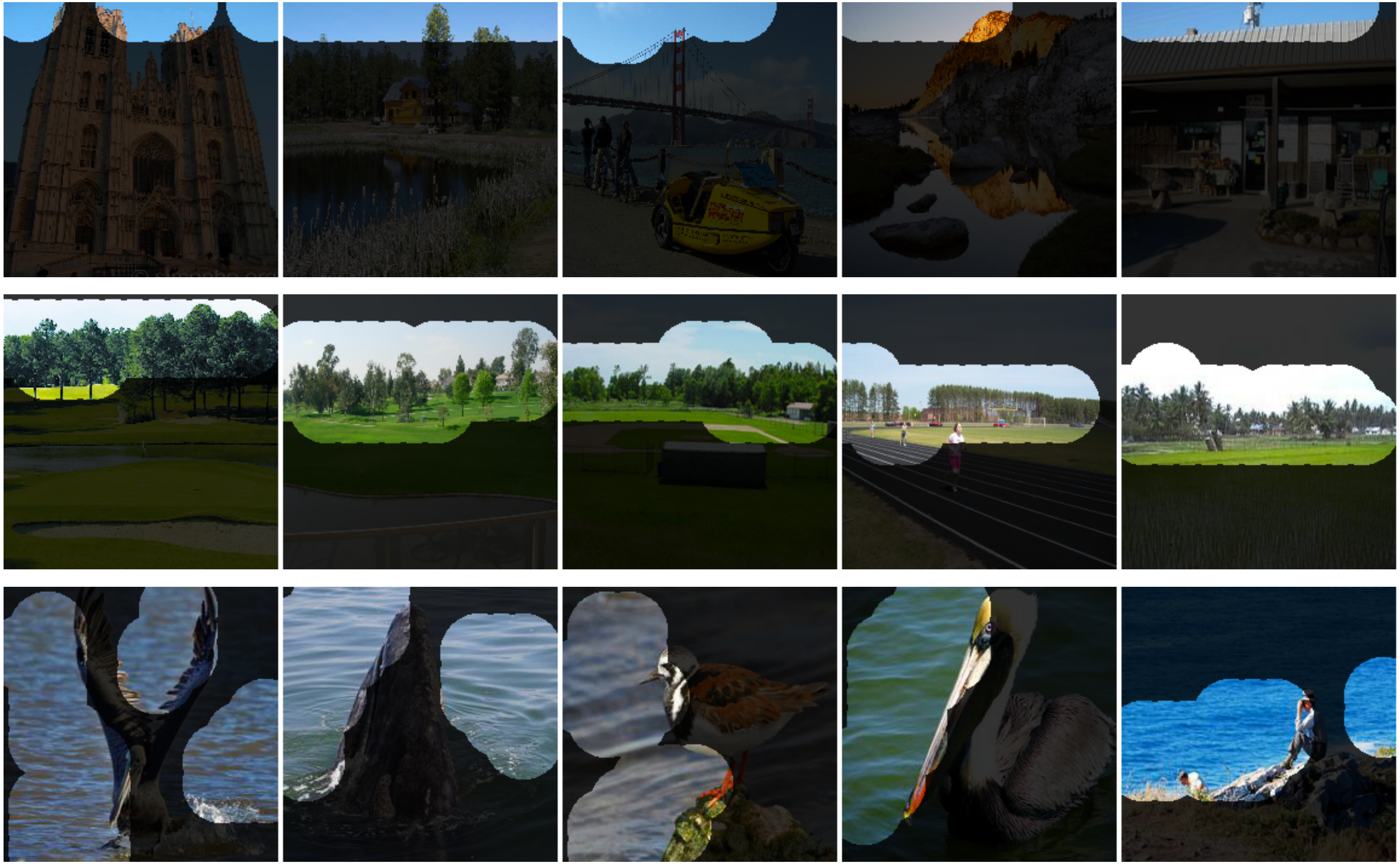


Hidden Unit (conv5) Activations

sky

trees

water

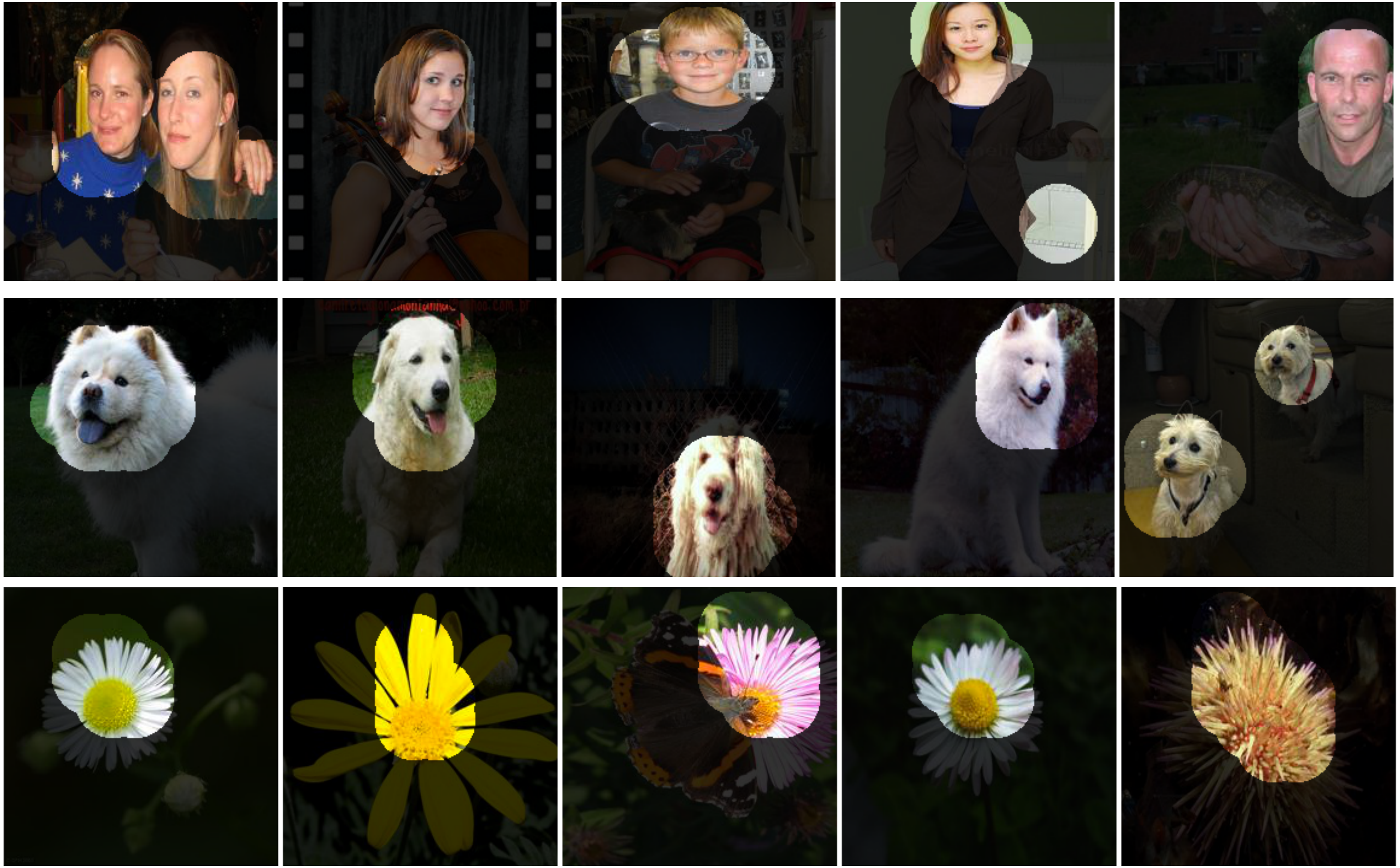


Hidden Unit (conv5) Activations

faces

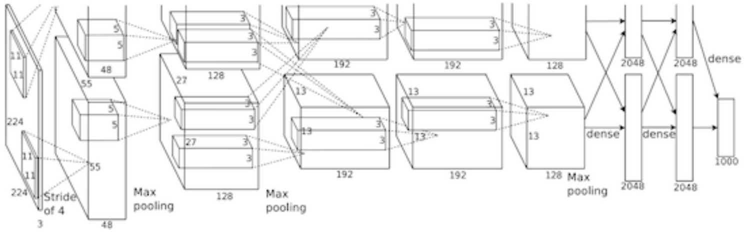
dog faces

flowers



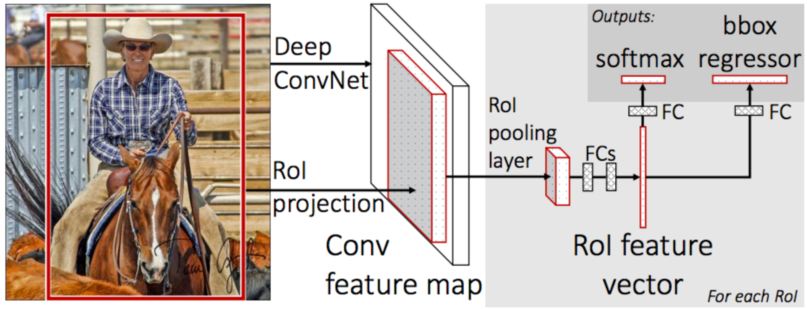
Dataset & Task Generalization on PASCAL VOC

Does the feature representation
transfer to other datasets and tasks?



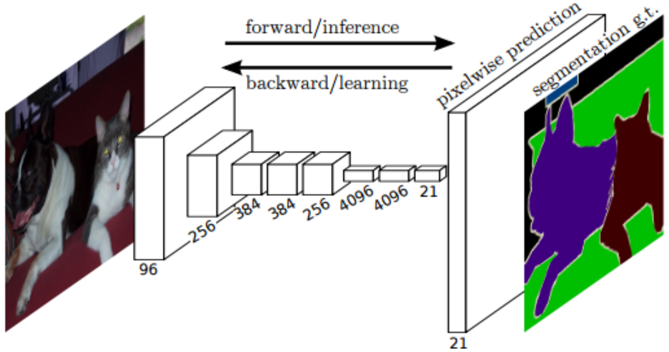
Classification

Krähenbühl et al. In ICLR, 2016.



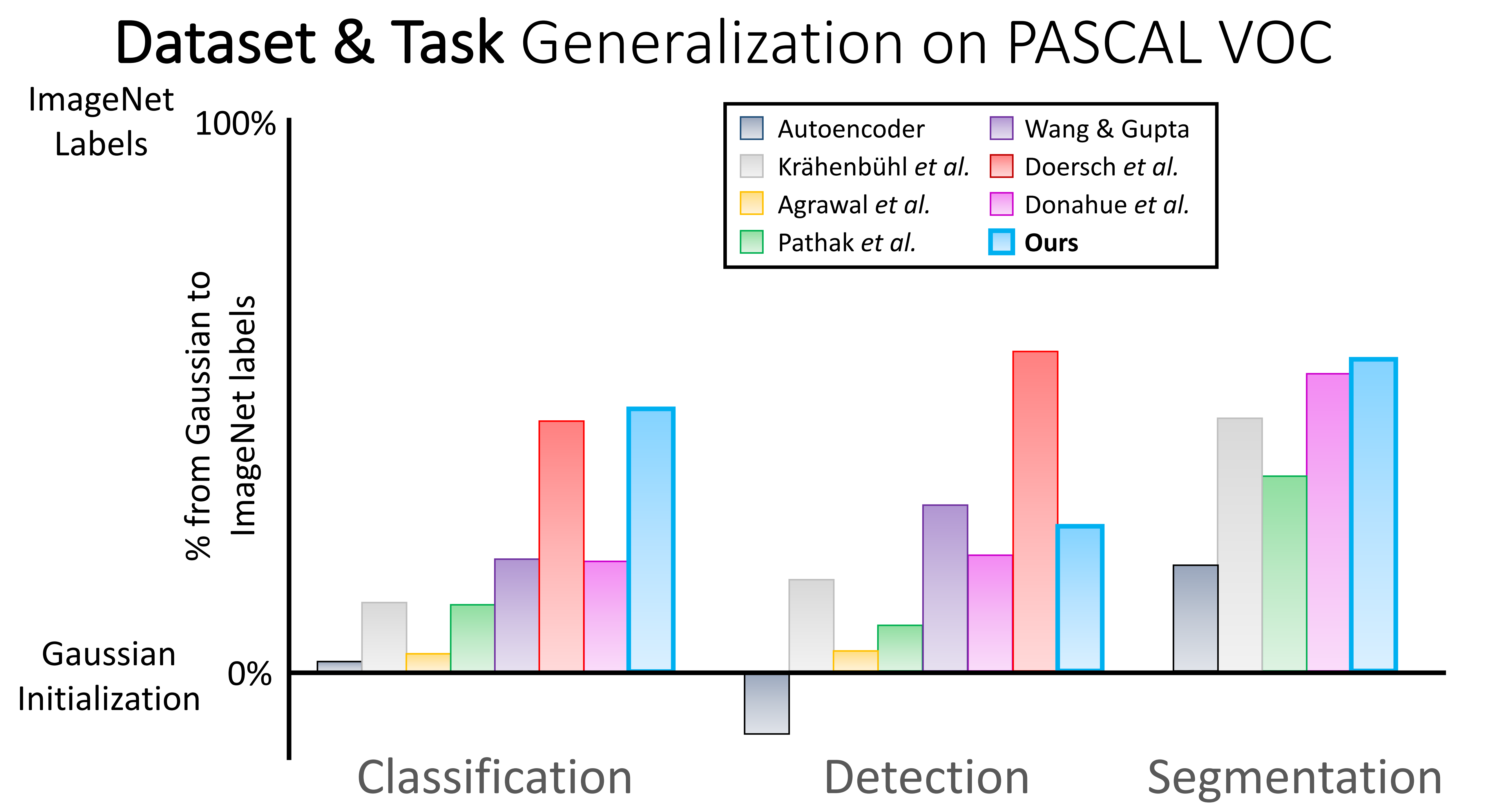
Detection

Fast R-CNN. Girshick. In ICCV, 2015.



Segmentation

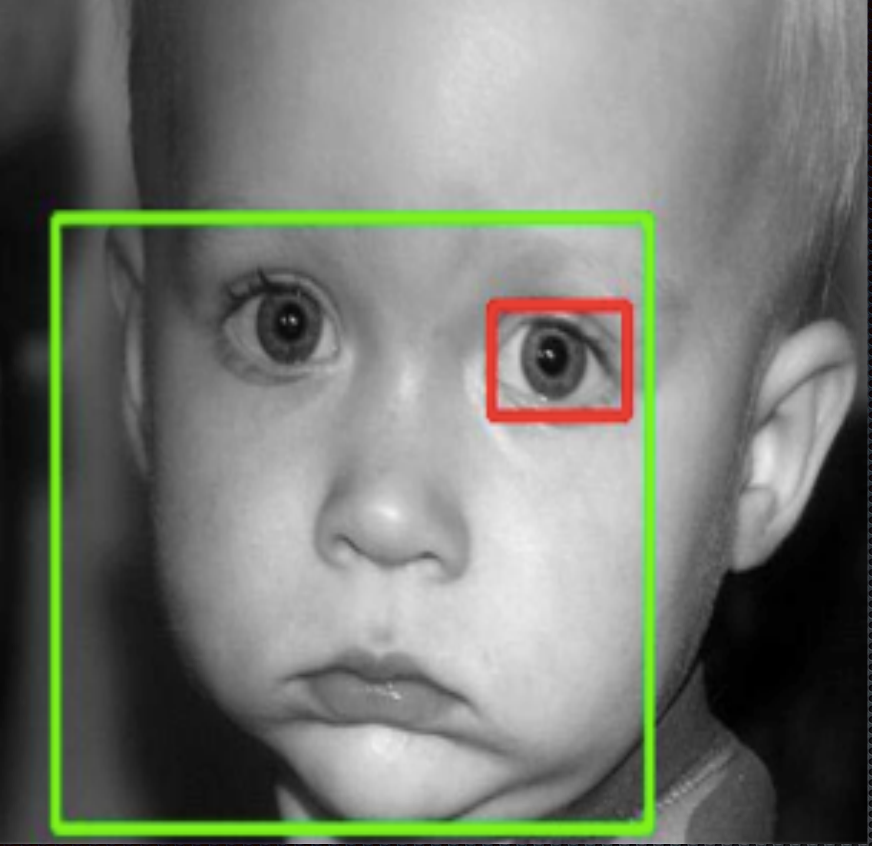
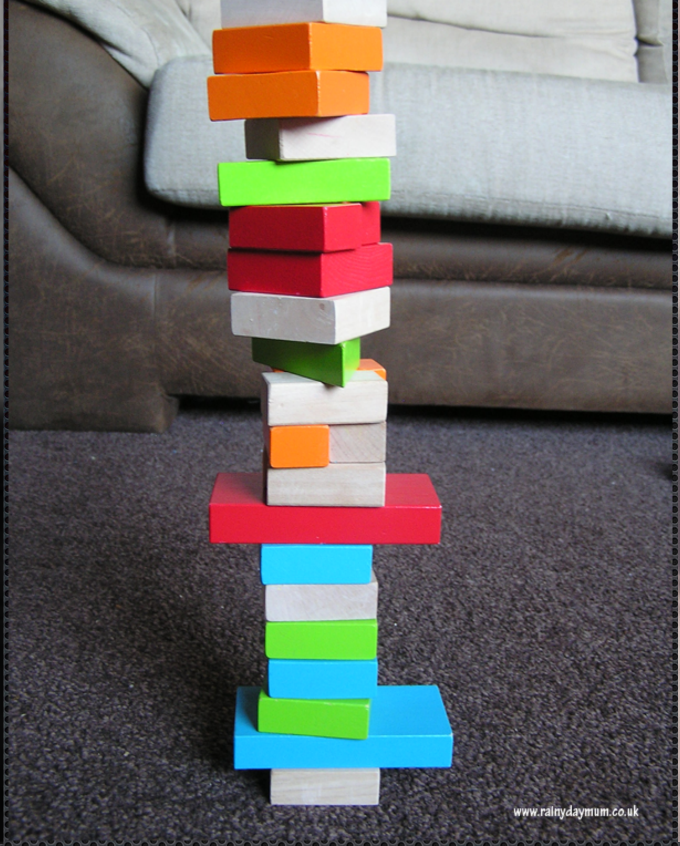
FCNs. Long et al. In CVPR, 2015.



Dataset & Task Generalization on PASCAL VOC
ImageNet Labels
100%
% from Gaussian to ImageNet labels
Gaussian Initialization
0%
Autoencoder
Krähenbühl et al.
Agrawal et al.
Pathak et al.
Wang & Gupta
Doersch et al.
Donahue et al.
Ours
Classification
Detection
Segmentation

Self-Supervised Learning

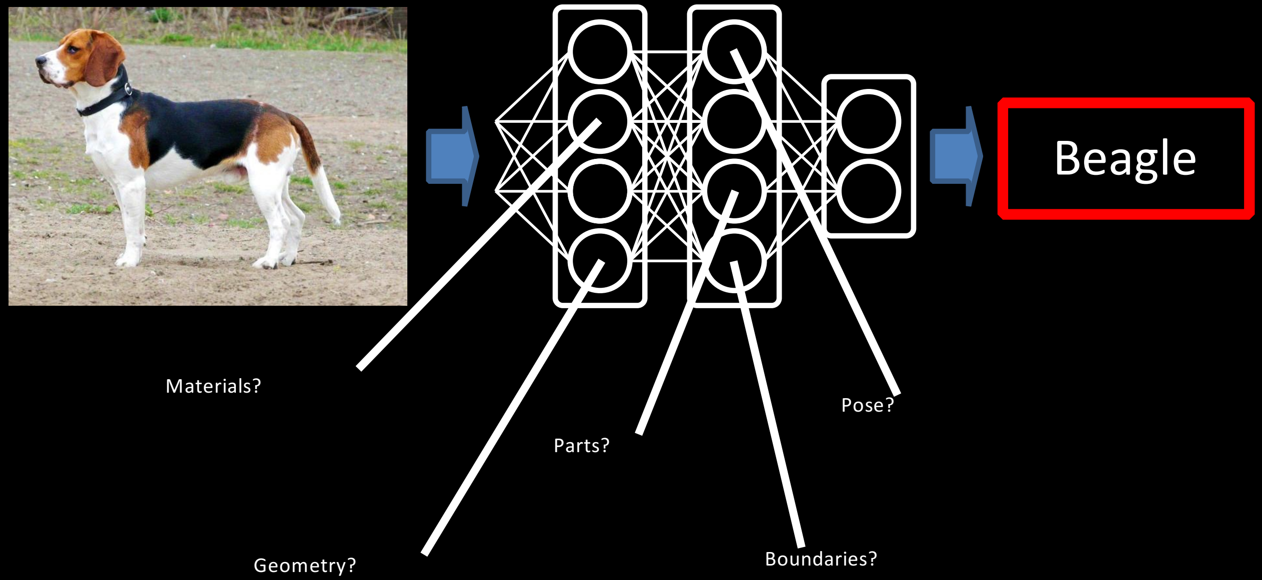
- Unsupervised feature learning by augmenting single images, Alexey Dosovitskiy, Jost Tobias Springenberg and Thomas Brox, NIPS 2014
- Colorful Image Colorization, Richard Zhang, Philip Isola, Alexei Efros, ECCV 2016
- Unsupervised Visual Representation Learning by Context Prediction, Carl Doersch, Abhinav Gupta, Alexei Efros, ICCV 2015
- Unsupervised Learning of Visual Representations using Videos, Xiaolong Wang, Abhinav Gupta, ICCV 2015

Scaling Self-Supervision

Learning Perception and Action without Human Supervision

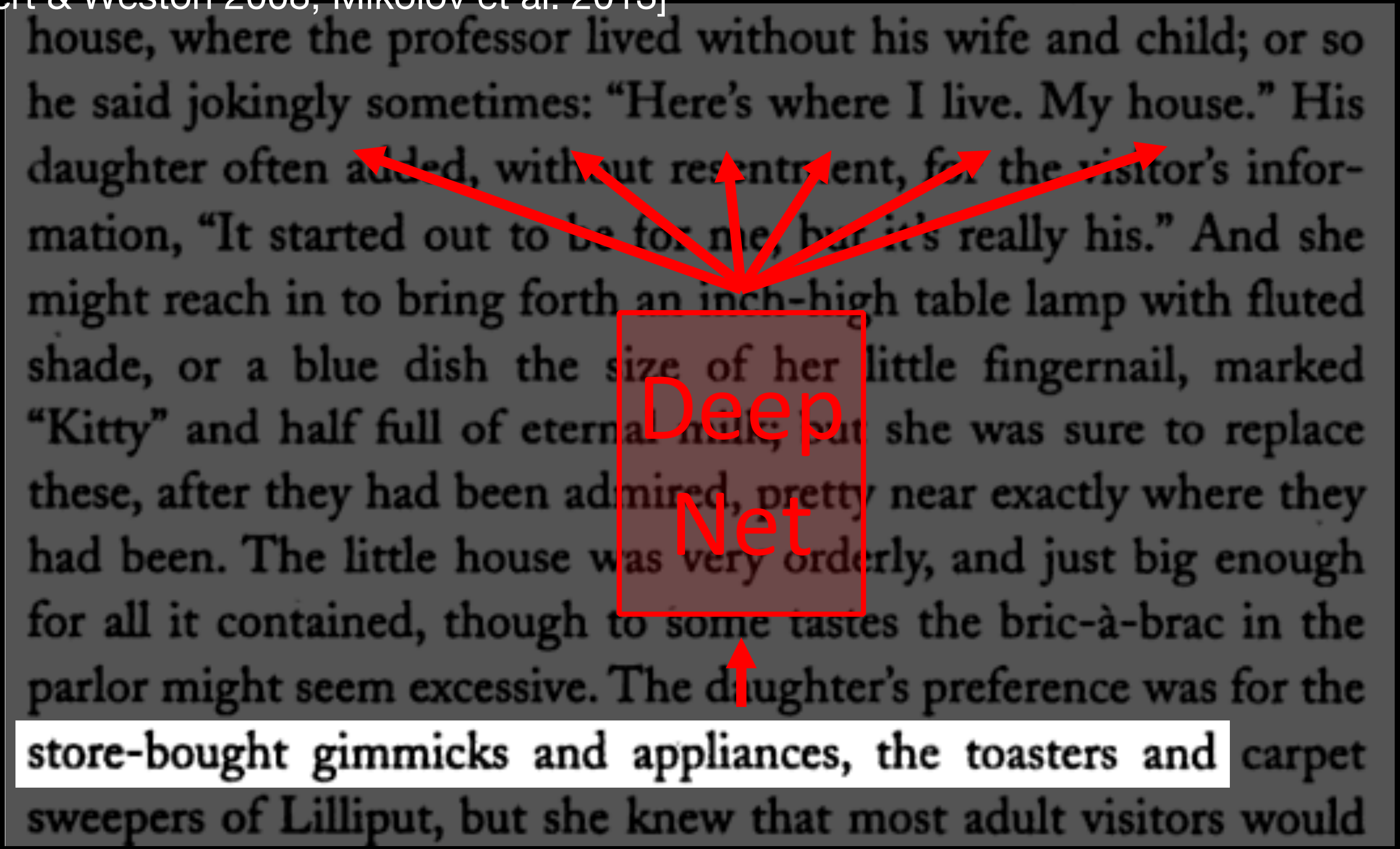
Abhinav Gupta



scale learning to billions of images

context as supervision

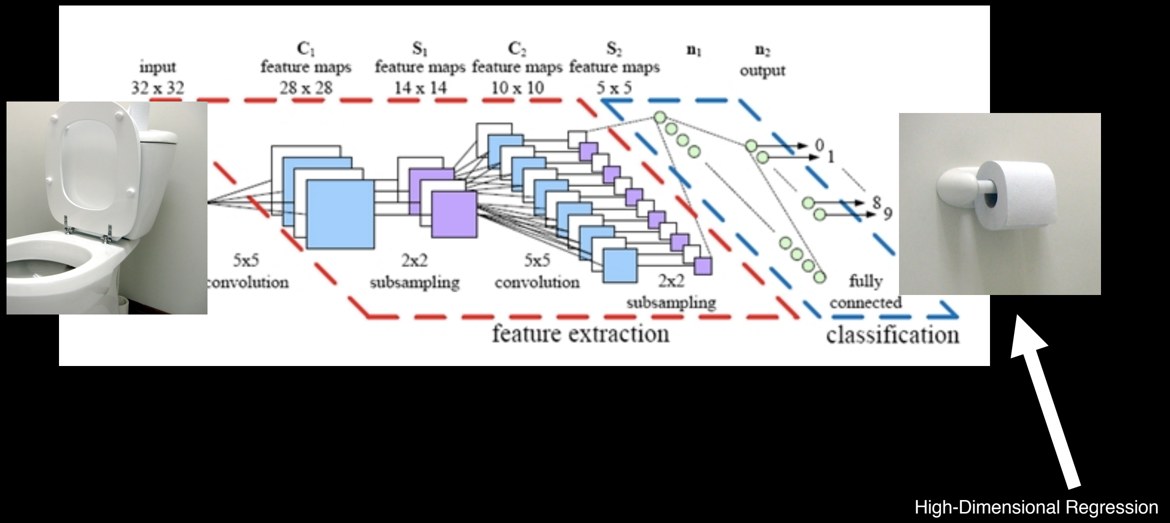
[Collobert & Weston 2008; Mikolov et al. 2013]

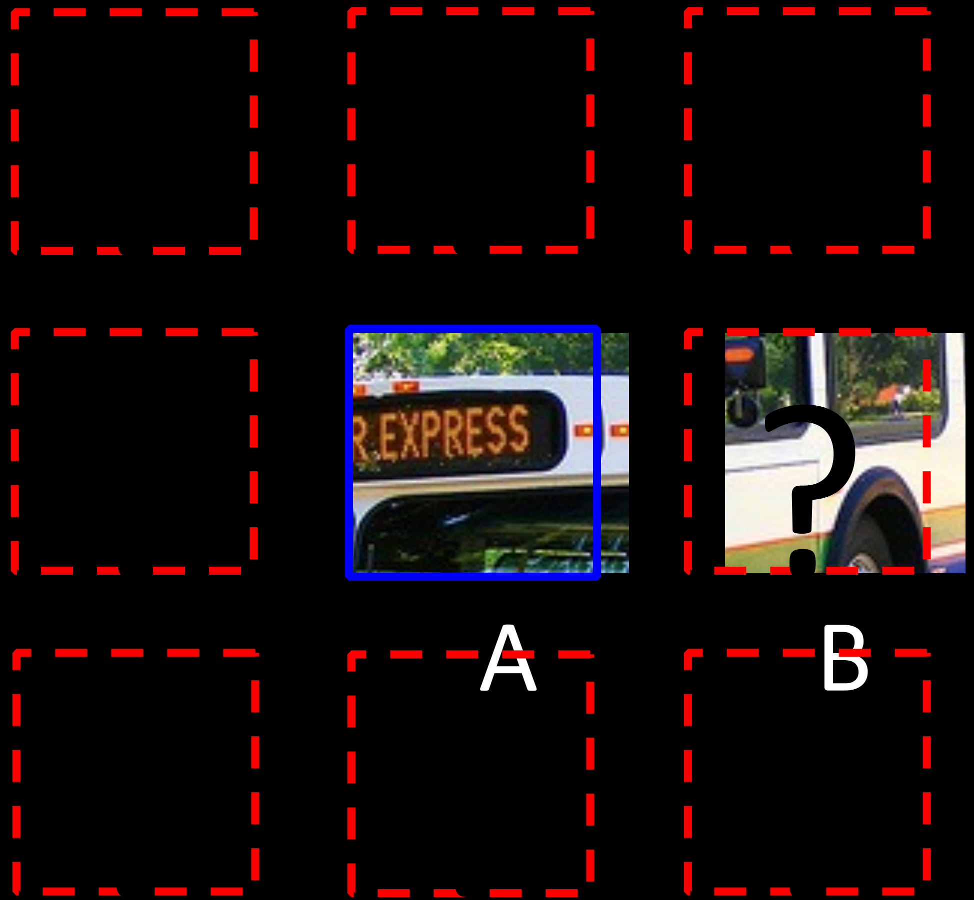


house, where the professor lived without his wife and child; or so he said jokingly sometimes: “Here’s where I live. My house.” His daughter often added, without resentment, for the visitor’s information, “It started out to be for me, but it’s really his.” And she might reach in to bring forth an inch-high table lamp with fluted shade, or a blue dish the size of her little fingernail, marked “Kitty” and half full of eternal milk, but she was sure to replace these, after they had been admired, pretty near exactly where they had been. The little house was very orderly, and just big enough for all it contained, though to some tastes the bric-à-brac in the parlor might seem excessive. The daughter’s preference was for the store-bought gimmicks and appliances, the toasters and carpet sweepers of Lilliput, but she knew that most adult visitors would

idea

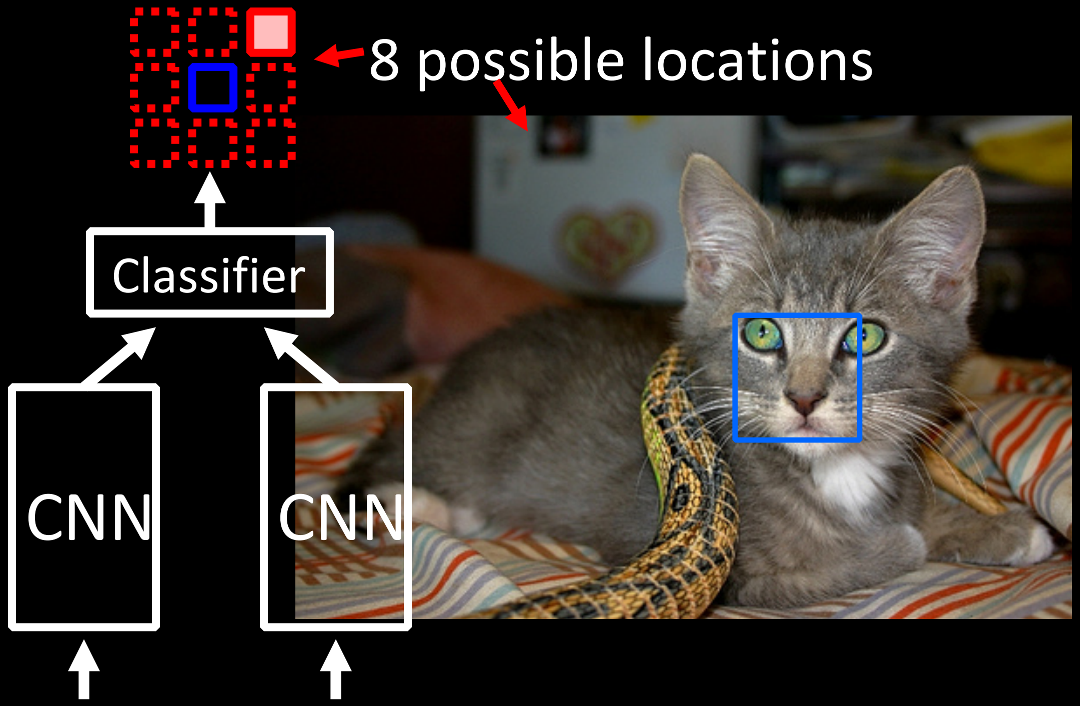
Train a CNN to predict the patches in context

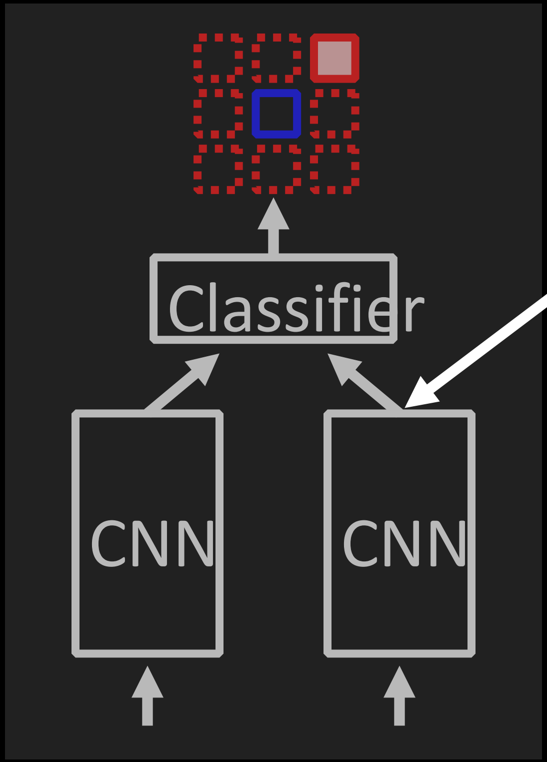



R EXPRESS
?
A
B

Semantics from a non-semantic task

Relative Position Task



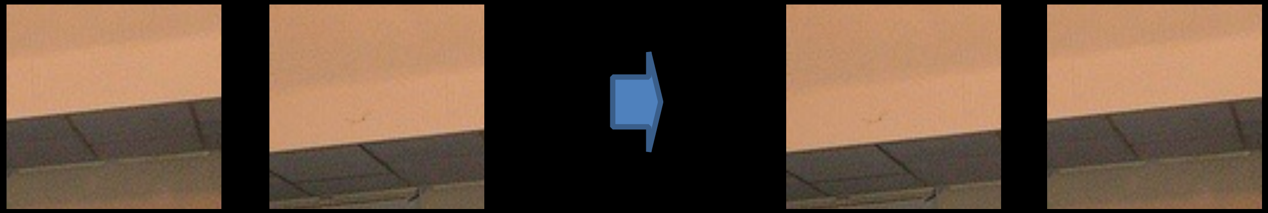


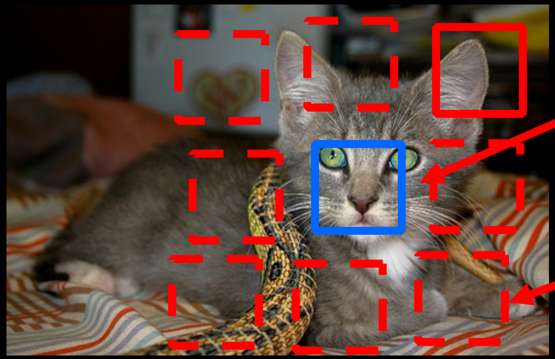
Patch Embedding

Input Nearest Neighbors

Note: connects ***across*** instances!

Avoiding Trivial Shortcuts

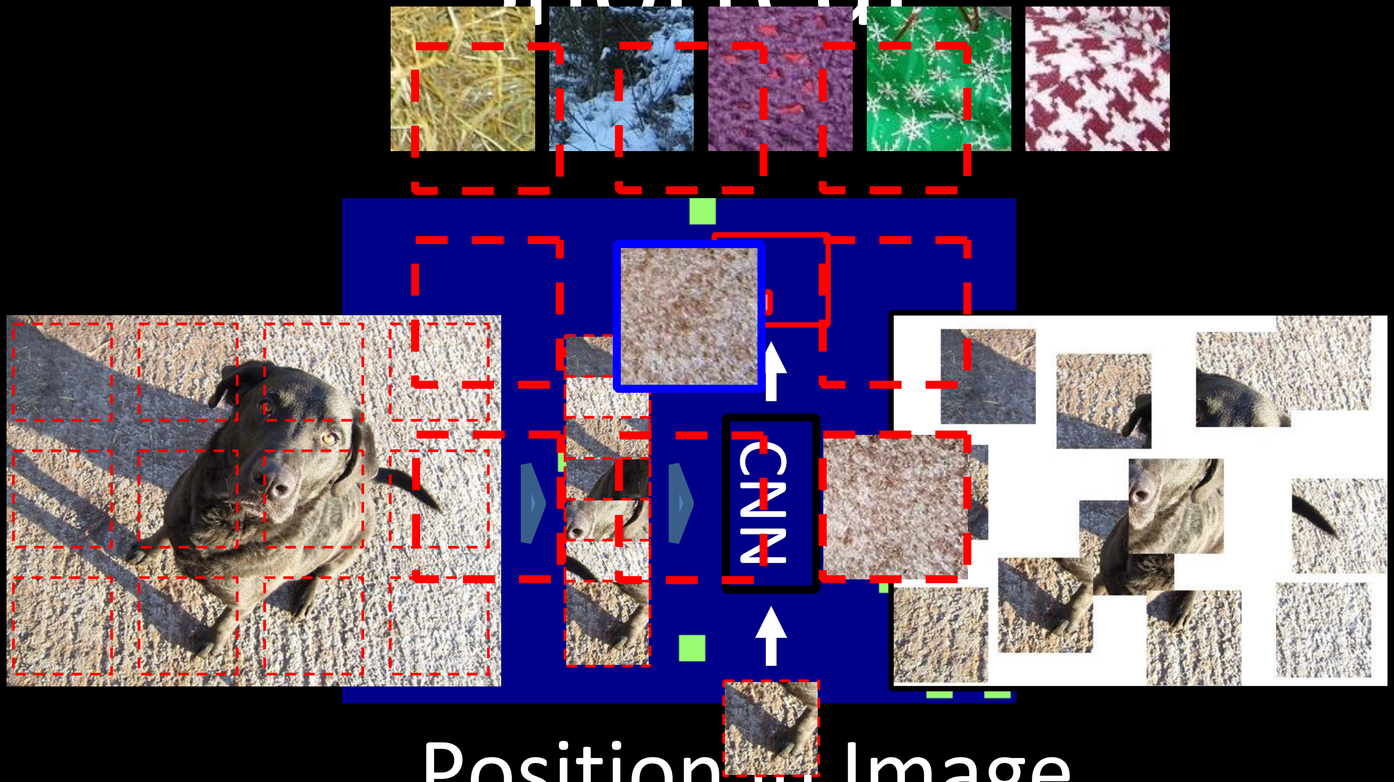




Include a gap

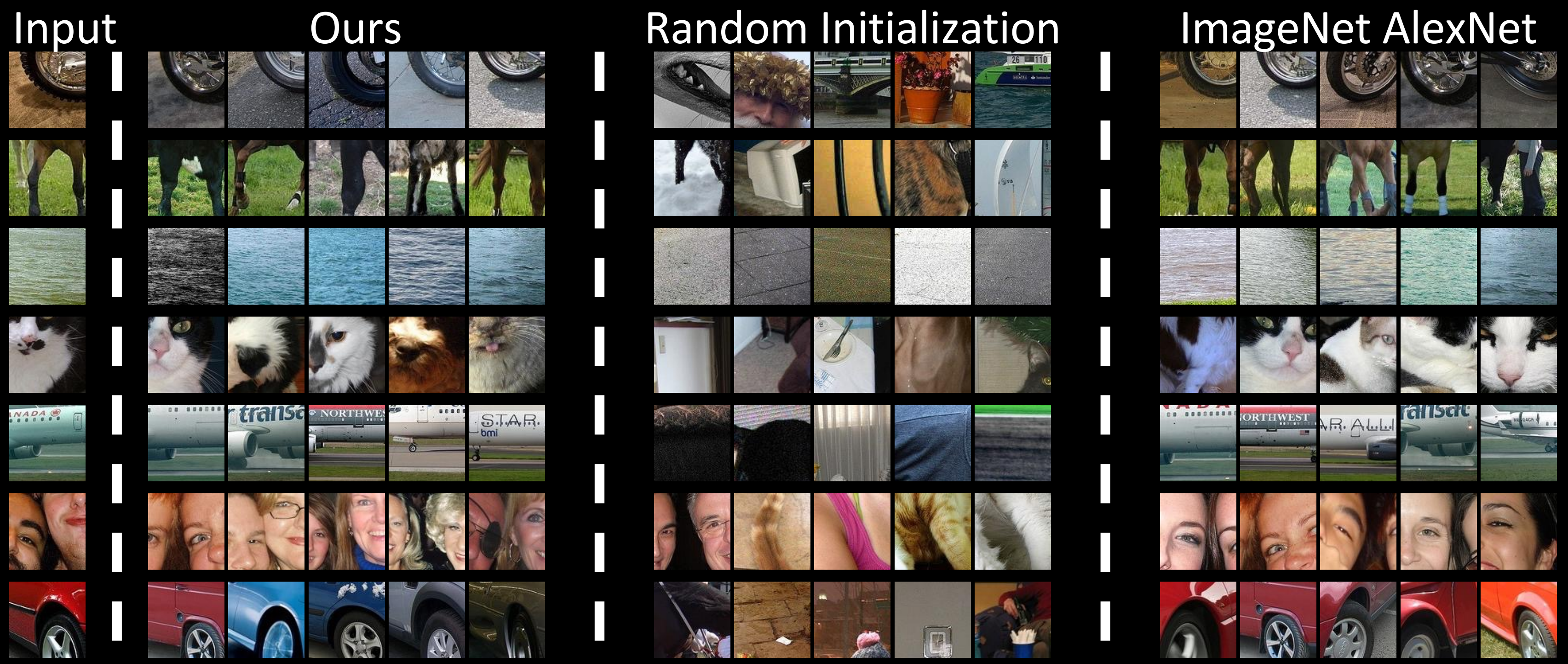
Jitter the patch locations

A Not-So “Trivial” Shortcut



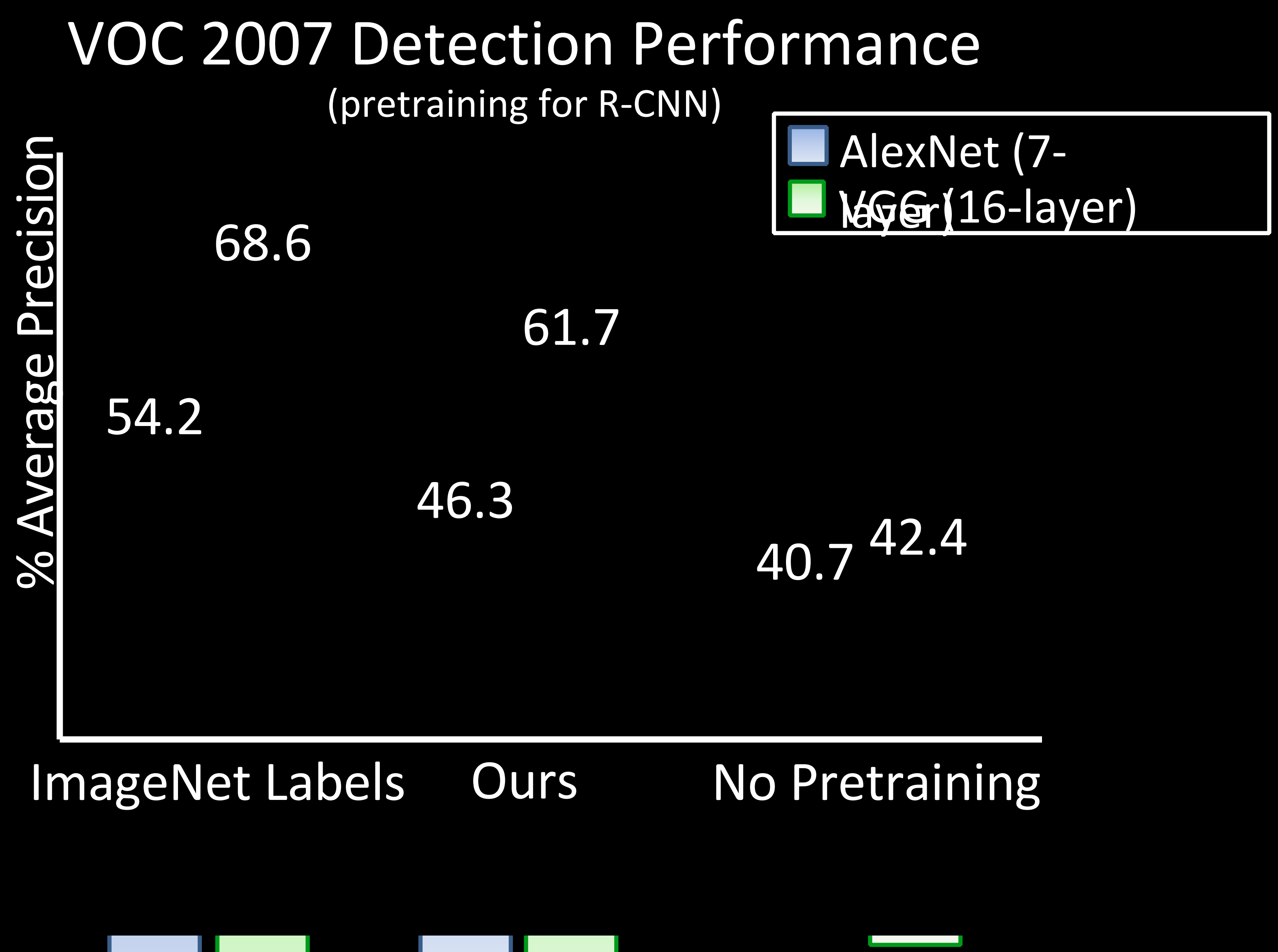
Position in Image

What is learned?



how do we evaluate?

- Use as pre-trained network for VOC object detection.
- Compare to ConvNet trained with ImageNet and ConvNet without Pre-training.

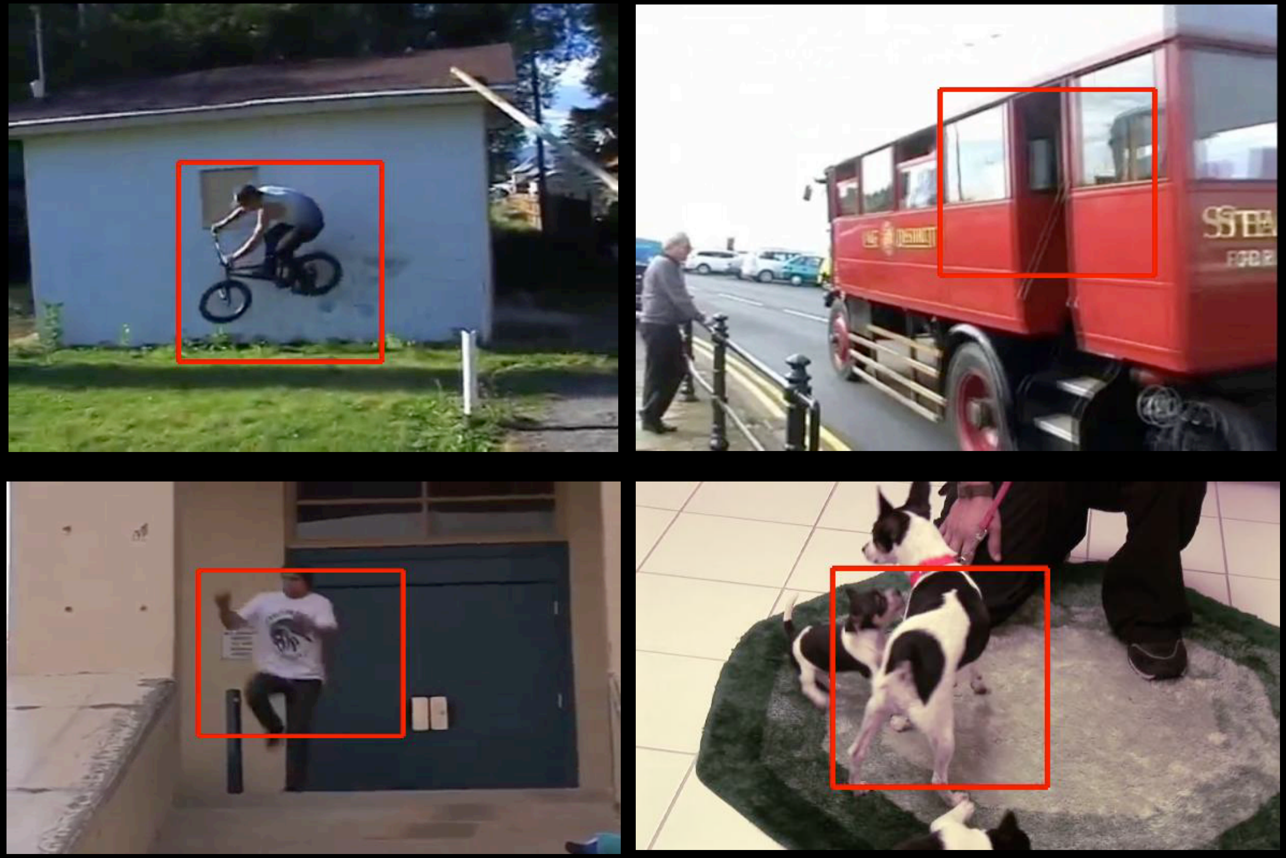


VOC 2007 Detection Performance
(pretraining for R-CNN)
AlexNet (7-layer)
VGG (16-layer)
% Average Precision
68.6
54.2
61.7
46.3
40.7
42.4
ImageNet Labels
Ours
No Pretraining

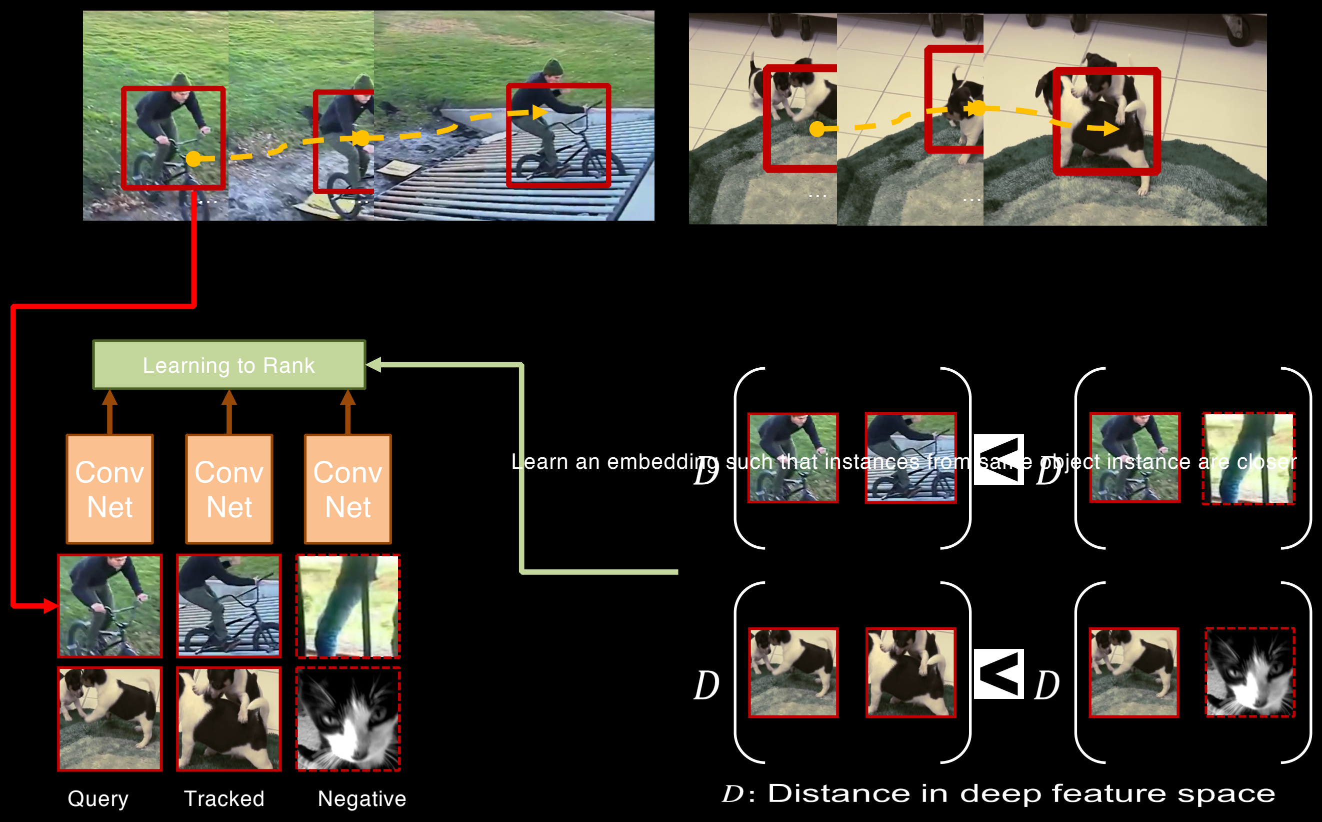
take-home summary

- Surprising: Within image-context leads to across-image similarities.
- Significantly better than no pre-training, but the performance is still below ImageNet pre-training.

Videos: Use tracking



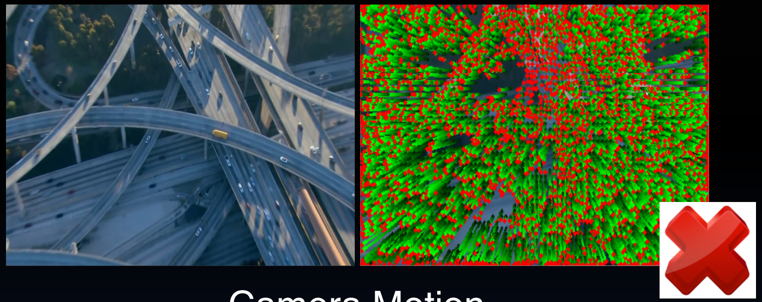
Approach: Outline



Patch Mining In Videos

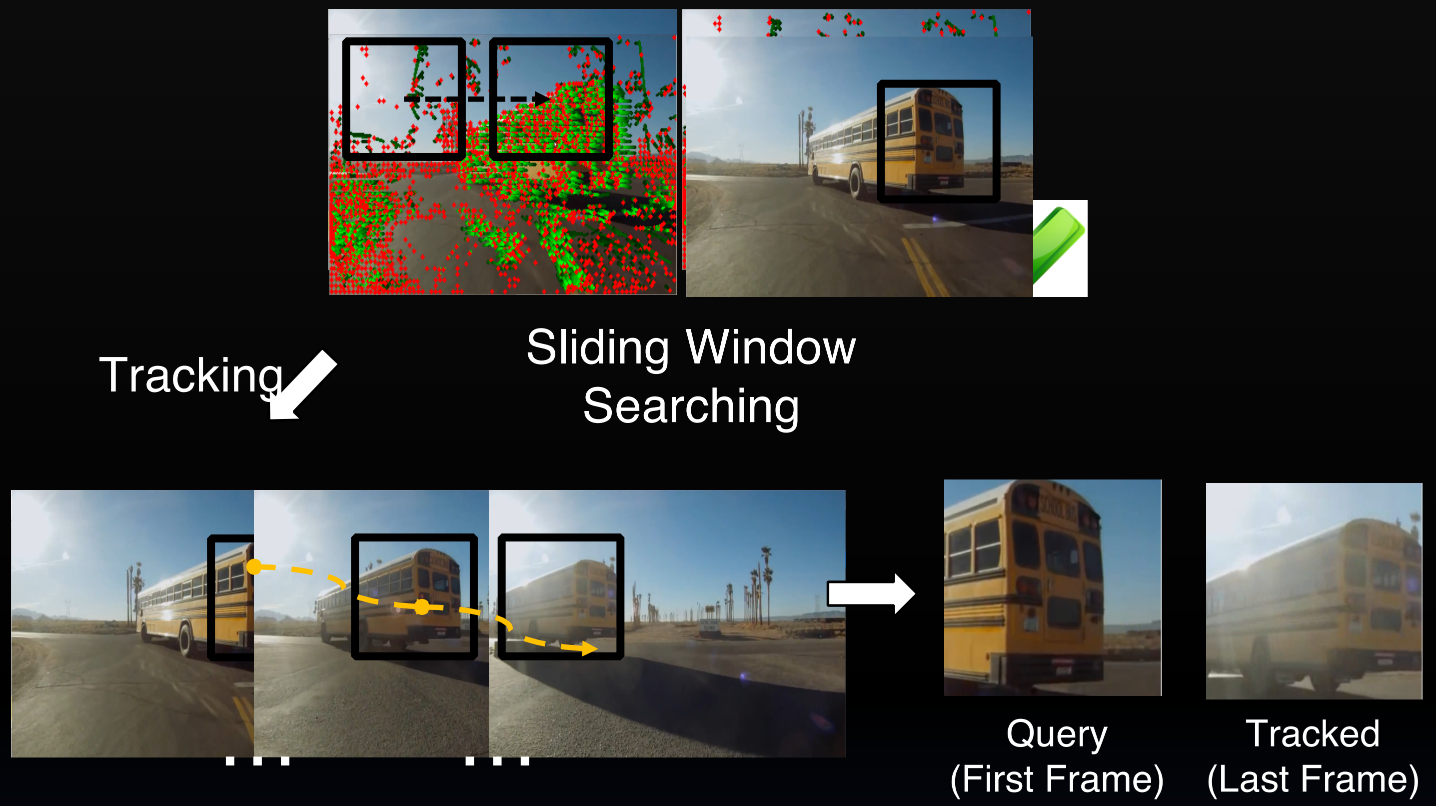
- Track 8M patches in 100K videos from YouTube.
- Use tracking algorithms with no learning.

Small Motion



Camera Motion

Patch Mining In Videos

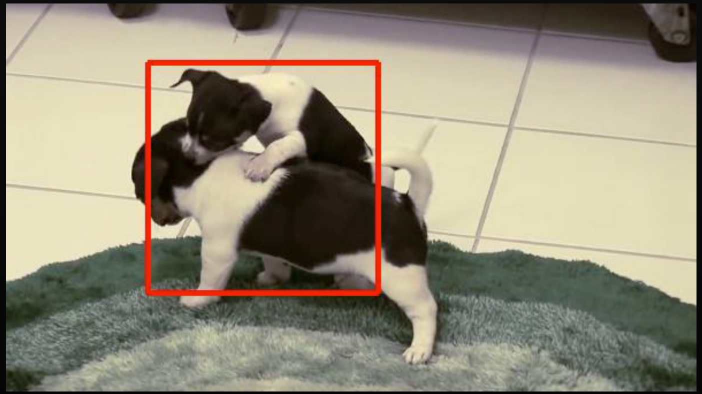


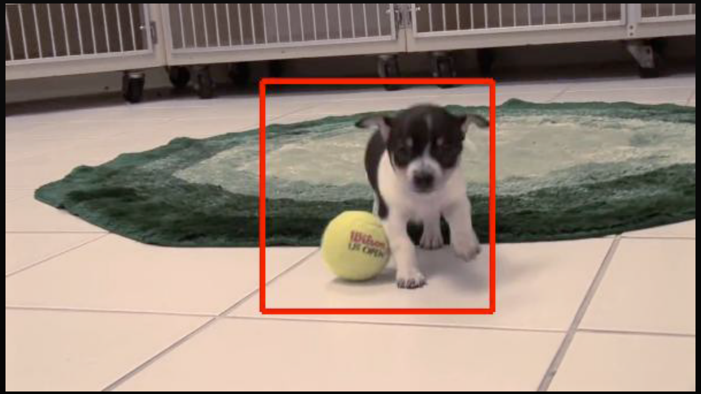
Some Examples

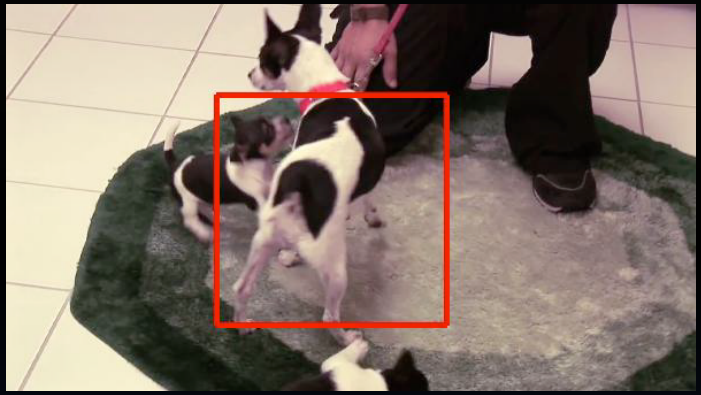


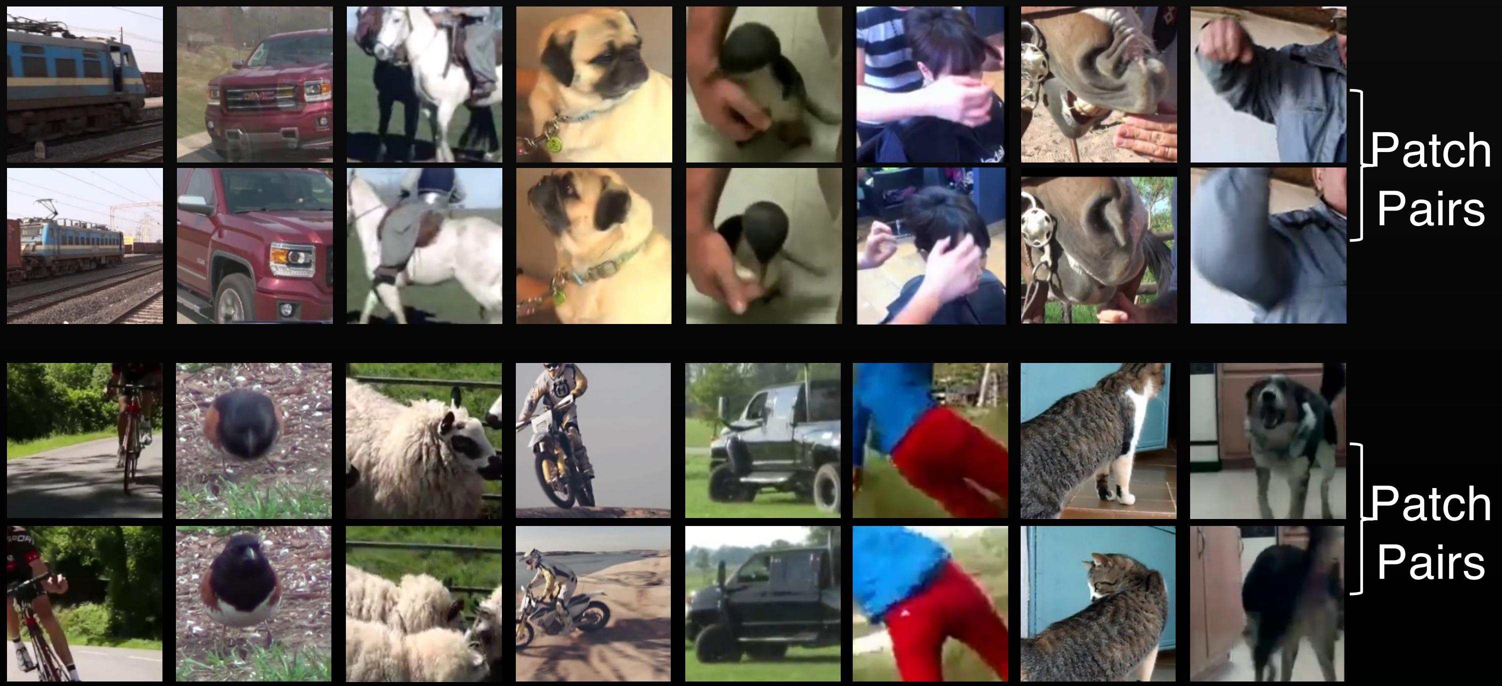


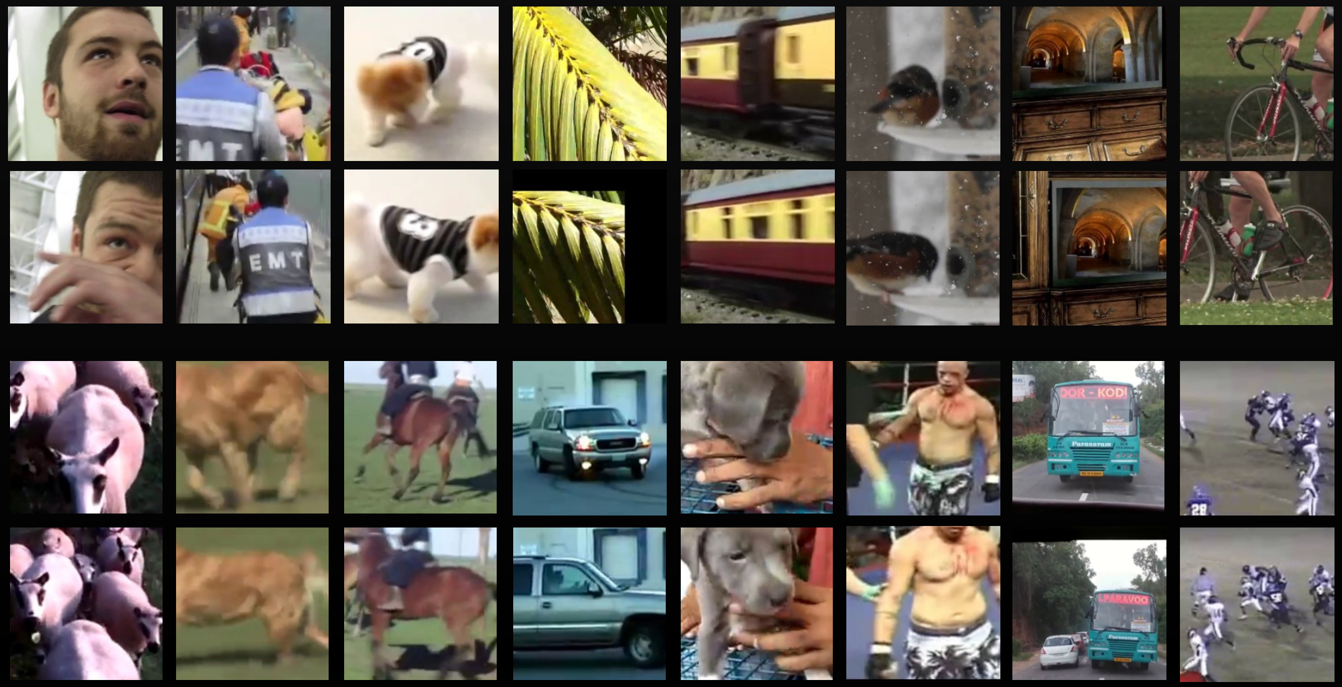

Dog








Patch
Pairs
Patch
Pairs



Patch Pairs

Patch Pairs

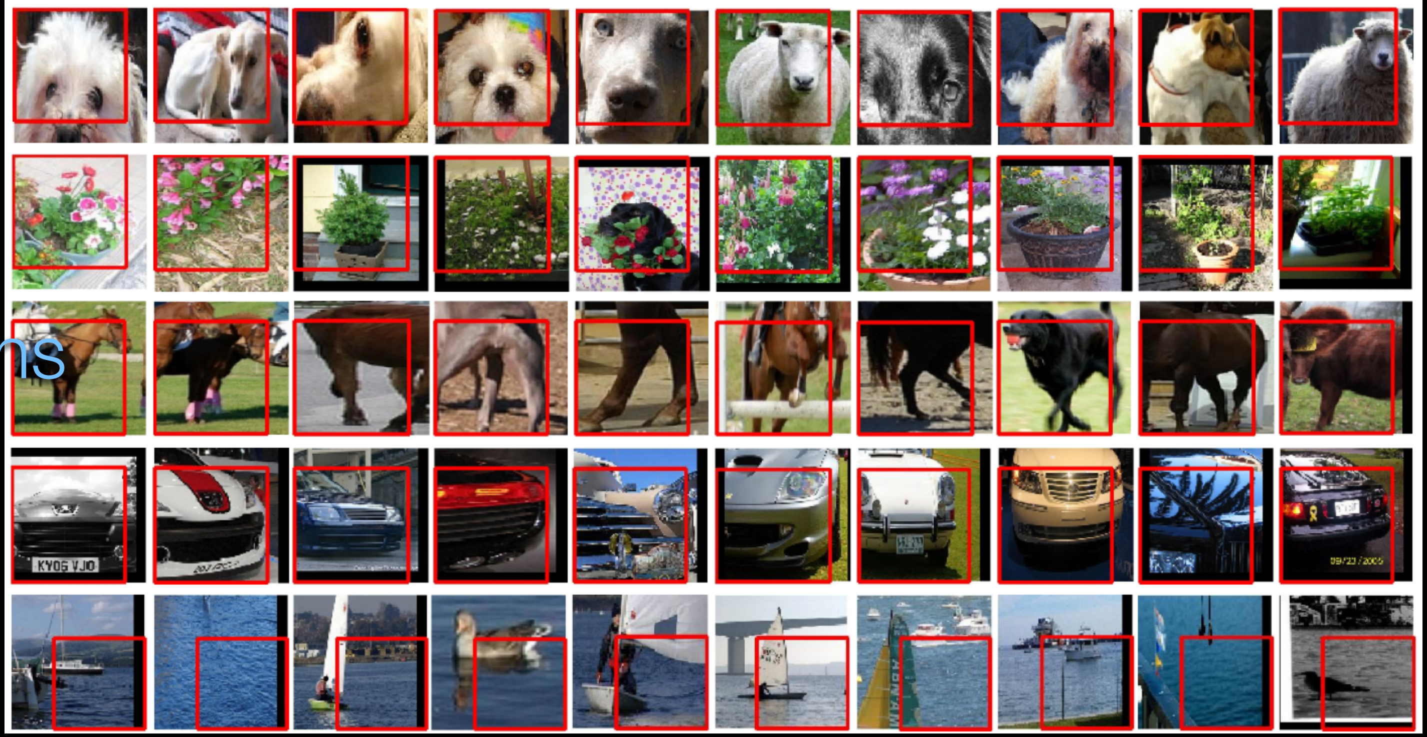
LEARNING VIA VIDEOS

- Space of negatives is huge might take lot of time for network to learn
- Hard Negative Mining for Triplet Sampling
  - Random Selection (150K iterations)
  - Hard Negatives: Negative patches giving high loss
  - Backprop only for Hard Negatives
  - Hard Negative Mining for 200K iterations

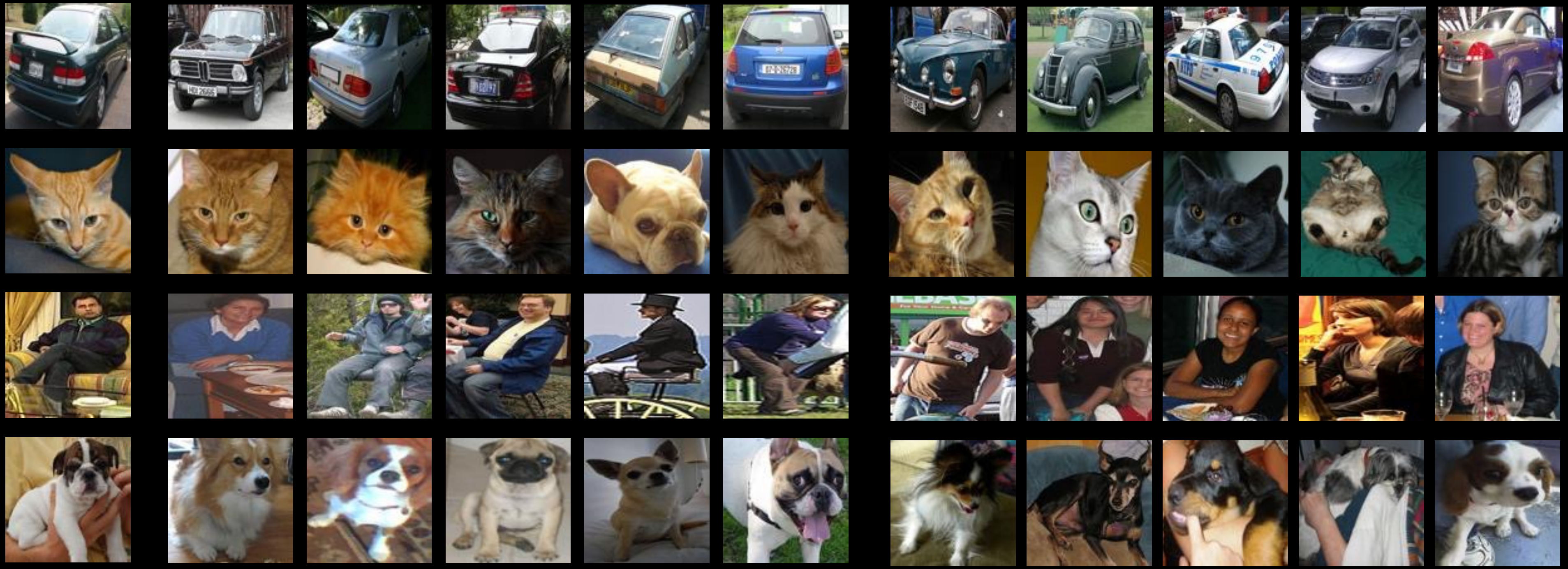
What does the network learn?

- trained using 8M tracks from 100K YouTube Videos

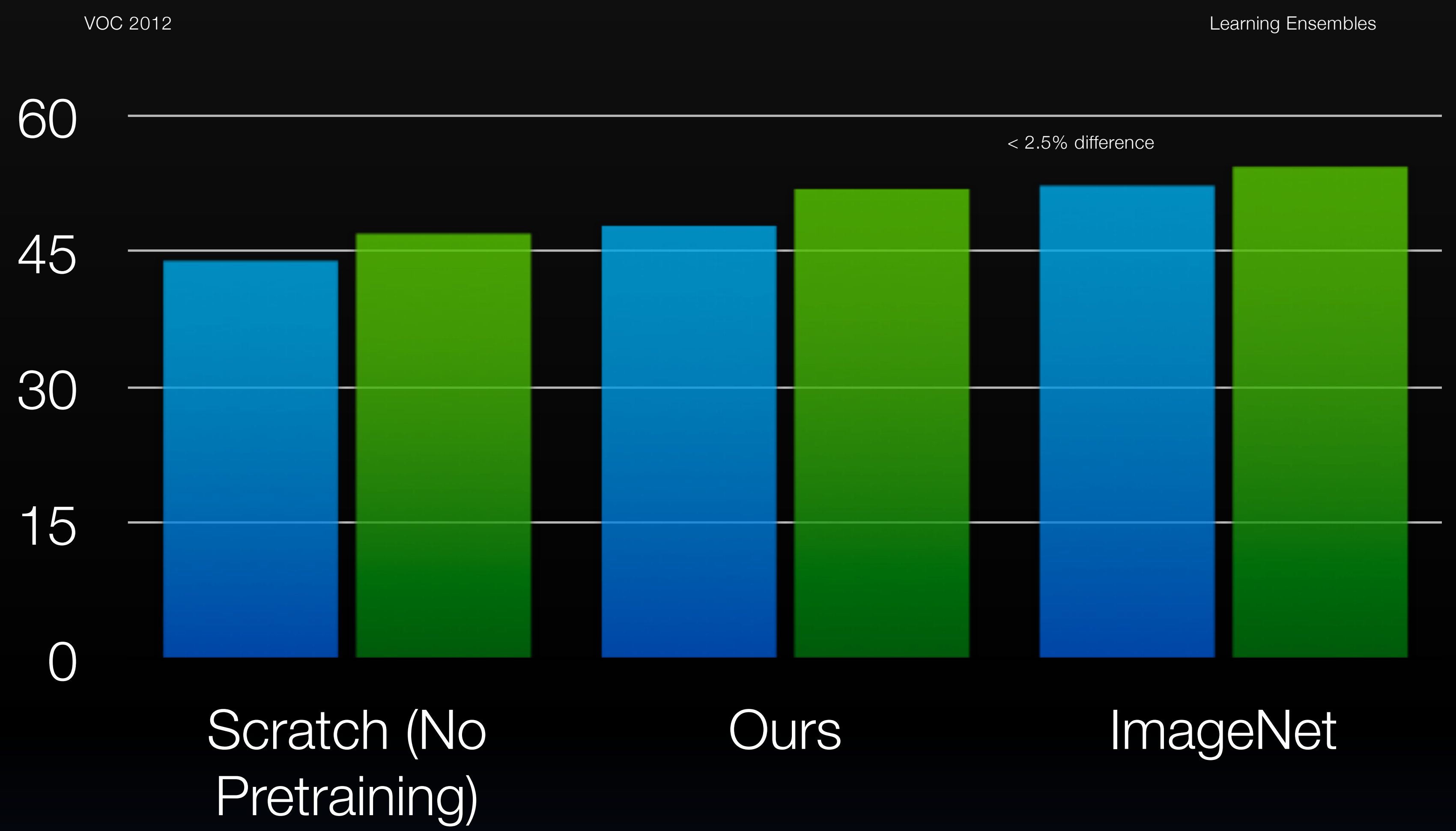
Pool5 Neurons



Nearest Neighbor



Query (a) Imagenet AlexNet (b) Unsupervised AlexNet



VOC 2012
Learning Ensembles
< 2.5% difference
60
45
30
15
0
Scratch (No Pretraining)
Ours
ImageNet

Summary

- Unsupervised learning big unsolved problem in ML/AI
- Lots of active research
- Auto-encoder appealing idea but performance is underwhelming
- Self-supervised methods interesting, but not generic
- New approaches offer promise, e.g. generative adversarial nets and variational auto-encoders
  - Will be covered by Emily Denton on 12/15