Convolutional Neural Networks

Computer Vision

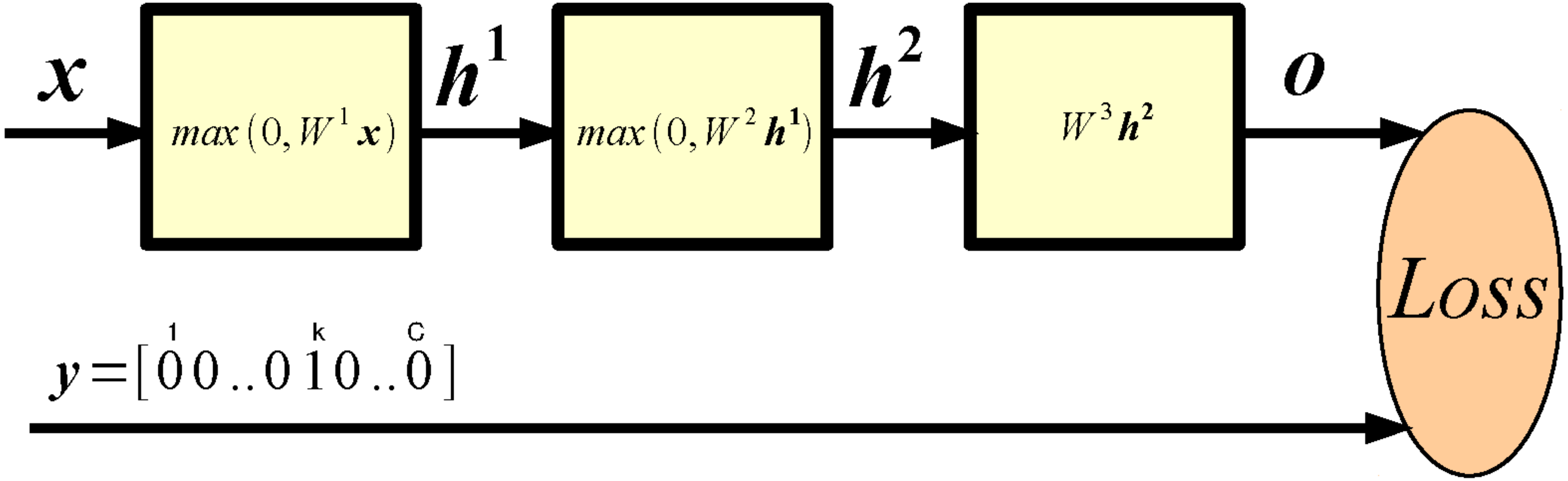
James Hays

Many slides by Marc'Aurelio Ranzato

Outline

- Neural Networks (covered in previous lecture)
- *Convolutional* Neural Networks
- Visualization and interpretation of Deep Networks

How Good is a Network?



Probability of class k given input (softmax):

$$p(c_k = 1|\boldsymbol{x}) = \frac{e^{o_k}}{\sum_{j=1}^{C} e^{o_j}}$$

(Per-sample) **Loss**; e.g., negative log-likelihood (good for classification of small number of classes):

$$L(\boldsymbol{x}, y; \boldsymbol{\theta}) = -\sum_j y_j \log p(c_j|\boldsymbol{x})$$

Training

Learning consists of minimizing the loss (plus some regularization term) w.r.t. parameters over the whole training set.

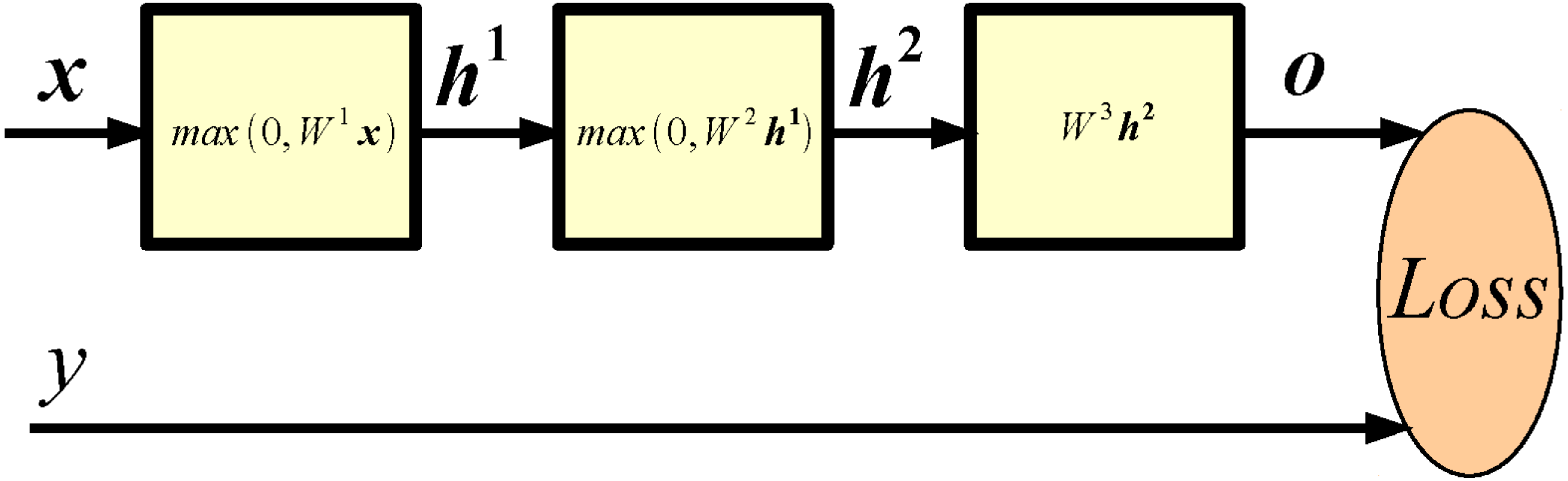
$$\boldsymbol{\theta}^* = arg\ min_{\boldsymbol{\theta}} \sum_{n=1}^{P} L(\boldsymbol{x}^n, y^n; \boldsymbol{\theta})$$

Question: How to minimize a complicated function of the parameters?

Answer: Chain rule, a.k.a. **Backpropagation**! That is the procedure to compute gradients of the loss w.r.t. parameters in a multi-layer neural network.

Rumelhart et al. "Learning internal representations by back-propagating.." Nature 1986

Key Idea: Wiggle To Decrease Loss



Let's say we want to decrease the loss by adjusting $W^1_{i,j}$.
We could consider a very small $\epsilon = 1\text{e-}6$ and compute:

$$L(\boldsymbol{x}, y; \boldsymbol{\theta})$$

$$L(\boldsymbol{x}, y; \boldsymbol{\theta} \backslash W^1_{i,j}, W^1_{i,j} + \epsilon)$$

Then, update:

$$W^1_{i,j} \leftarrow W^1_{i,j} + \epsilon \, sgn(L(\boldsymbol{x}, y; \boldsymbol{\theta}) - L(\boldsymbol{x}, y; \boldsymbol{\theta} \backslash W^1_{i,j}, W^1_{i,j} + \epsilon))$$

Ranzato

Derivative w.r.t. Input of Softmax

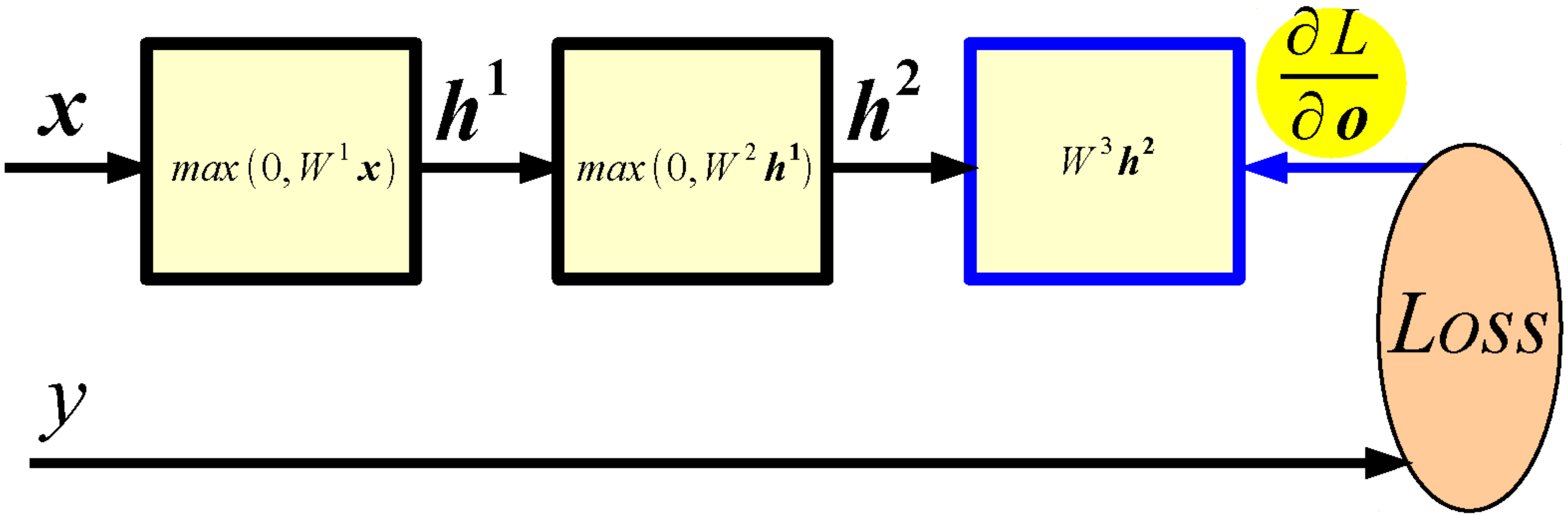
$$p(c_k = 1|\boldsymbol{x}) = \frac{e^{o_k}}{\sum_j e^{o_j}}$$

$$L(\boldsymbol{x}, y; \boldsymbol{\theta}) = -\sum_j y_j \log p(c_j|\boldsymbol{x}) \qquad \boldsymbol{y} = [\overset{1}{0}\, 0 \,..\, 0\, \overset{k}{1}\, 0 \,..\, \overset{C}{0}]$$

By substituting the fist formula in the second, and taking the derivative w.r.t. $\boldsymbol{o}$ we get:

$$\frac{\partial L}{\partial o} = p(c|\boldsymbol{x}) - \boldsymbol{y}$$

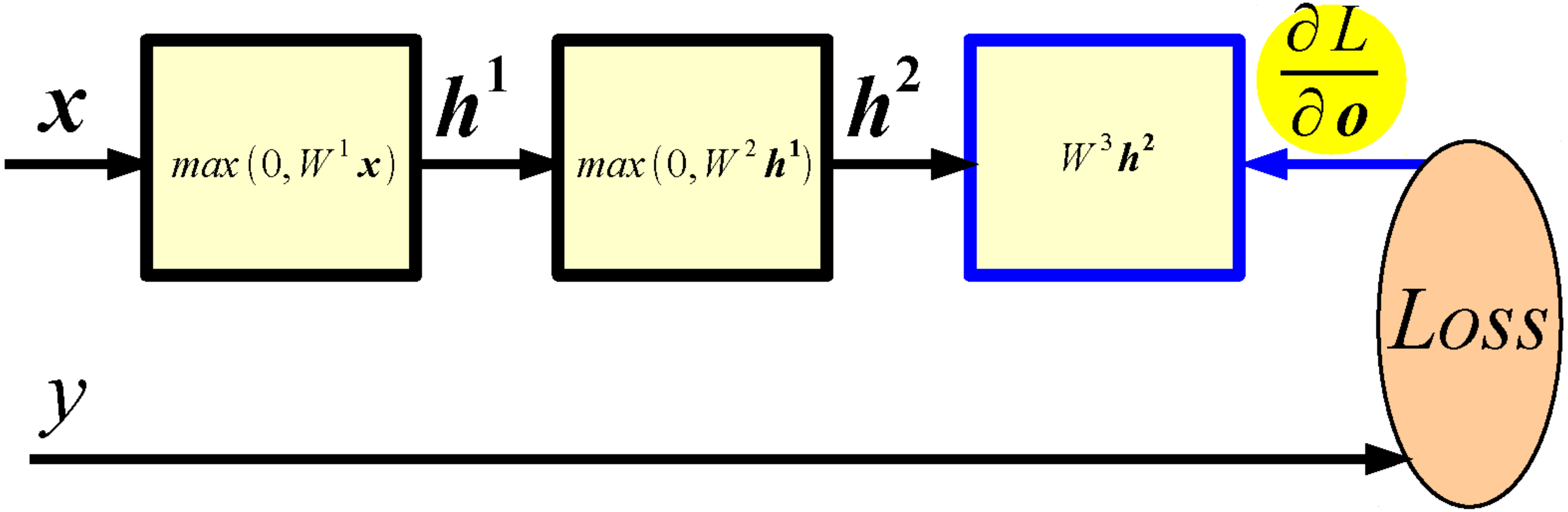
Backward Propagation



Given $\partial L / \partial \boldsymbol{o}$ and assuming we can easily compute the Jacobian of each module, we have:

$$\frac{\partial L}{\partial W^3} = \frac{\partial L}{\partial \boldsymbol{o}} \frac{\partial \boldsymbol{o}}{\partial W^3} \qquad \frac{\partial L}{\partial \boldsymbol{h}^2} = \frac{\partial L}{\partial \boldsymbol{o}} \frac{\partial \boldsymbol{o}}{\partial \boldsymbol{h}^2}$$

Backward Propagation

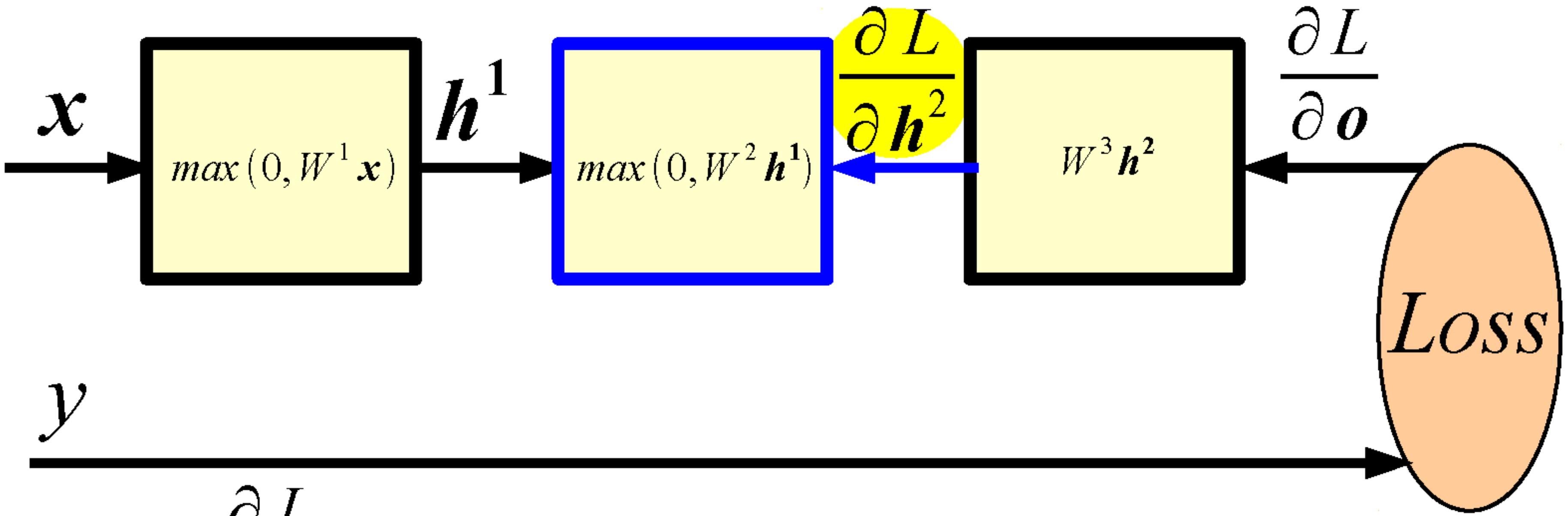


Given $\partial L / \partial \boldsymbol{o}$ and assuming we can easily compute the Jacobian of each module, we have:

$$\frac{\partial L}{\partial W^3} = \frac{\partial L}{\partial \boldsymbol{o}} \frac{\partial \boldsymbol{o}}{\partial W^3} \qquad \frac{\partial L}{\partial \boldsymbol{h}^2} = \frac{\partial L}{\partial \boldsymbol{o}} \frac{\partial \boldsymbol{o}}{\partial \boldsymbol{h}^2}$$

$$\frac{\partial L}{\partial W^3} = (p(c|\boldsymbol{x}) - \boldsymbol{y})\ \boldsymbol{h}^{2\,T} \qquad \frac{\partial L}{\partial \boldsymbol{h}^2} = W^{3\,T}(p(c|\boldsymbol{x}) - \boldsymbol{y})$$

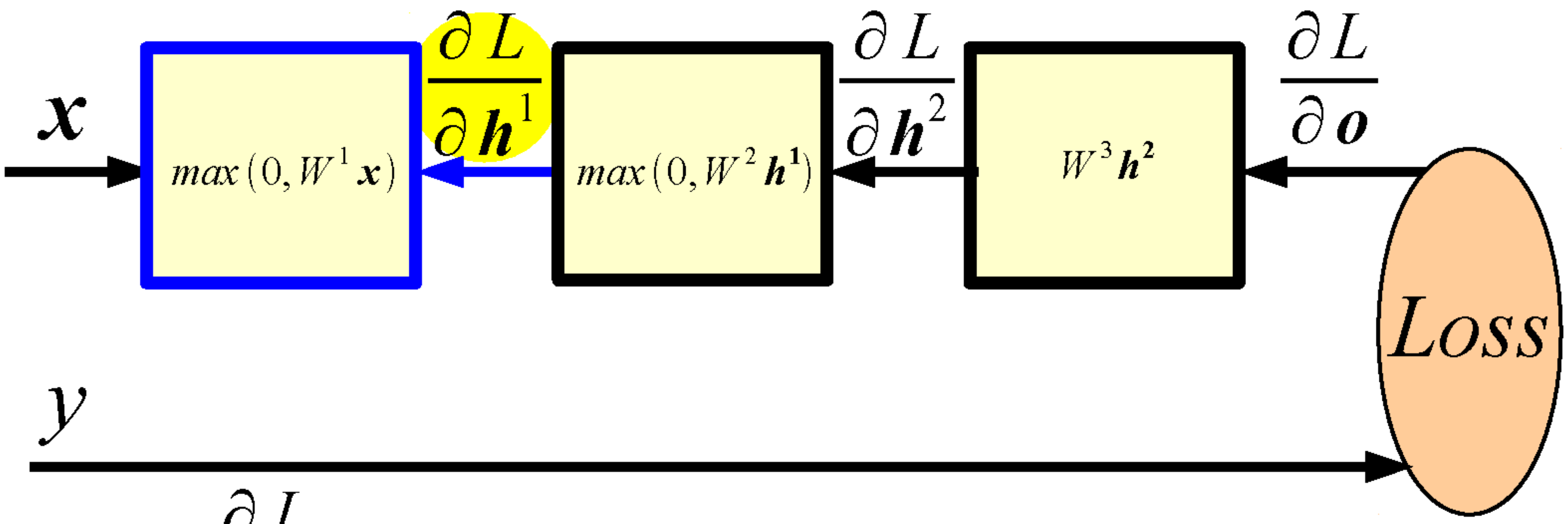
Backward Propagation



Given $\frac{\partial L}{\partial \boldsymbol{h}^2}$ we can compute now:

$$\frac{\partial L}{\partial W^2} = \frac{\partial L}{\partial \boldsymbol{h}^2} \frac{\partial \boldsymbol{h}^2}{\partial W^2} \qquad \frac{\partial L}{\partial \boldsymbol{h}^1} = \frac{\partial L}{\partial \boldsymbol{h}^2} \frac{\partial \boldsymbol{h}^2}{\partial \boldsymbol{h}^1}$$

Backward Propagation



Given $\frac{\partial L}{\partial \boldsymbol{h}^1}$ we can compute now:

$$\frac{\partial L}{\partial W^1} = \frac{\partial L}{\partial \boldsymbol{h}^1} \frac{\partial \boldsymbol{h}^1}{\partial W^1}$$

Ranzato

This all seems pretty complicated. Why are we using Neural Networks? James's rough assessment:

Learning method	Ease of configuration
Neural Network	1
Nearest Neighbor	10
Linear SVM	10
Non-linear SVM	5
Decision Tree or Random Forest	4

This all seems pretty complicated. Why are we using Neural Networks? James's rough assessment:

Learning method	Ease of configuration	Ease of interpretation
Neural Network	1	1
Nearest Neighbor	10	10
Linear SVM	10	9
Non-linear SVM	5	4
Decision Tree or Random Forest	4	4

This all seems pretty complicated. Why are we using Neural Networks? James's rough assessment:

Learning method	Ease of configuration	Ease of interpretation	Speed / memory when training
Neural Network	1	1	1
Nearest Neighbor	10	10	8
Linear SVM	10	9	10
Non-linear SVM	5	4	2
Decision Tree or Random Forest	4	4	4

This all seems pretty complicated. Why are we using Neural Networks? James's rough assessment:

Learning method	Ease of configuration	Ease of interpretation	Speed / memory when training	Speed / memory at test time
Neural Network	1	1	1	6
Nearest Neighbor	10	10	8	4
Linear SVM	10	9	10	10
Non-linear SVM	5	4	2	2
Decision Tree or Random Forest	4	4	4	8

This all seems pretty complicated. Why are we using Neural Networks? James's rough assessment:

Learning method	Ease of configuration	Ease of interpretation	Speed / memory when training	Speed / memory at test time	Accuracy w/ lots of data
Neural Network	1	1	1	6	10
Nearest Neighbor	10	10	8	4	7
Linear SVM	10	9	10	10	5
Non-linear SVM	5	4	2	2	8
Decision Tree or Random Forest	4	4	4	8	7

This all seems pretty complicated. Why are we using Neural Networks? James's rough assessment:

Learning method	Ease of configuration	Ease of interpretation	Speed / memory when training	Speed / memory at test time	Accuracy w/ lots of data
Neural Network	1	1	1	6	10
Nearest Neighbor	10	10	8	4	7
Linear SVM	10				
Non-linear SVM	5				
Decision Tree or Random Forest	4				

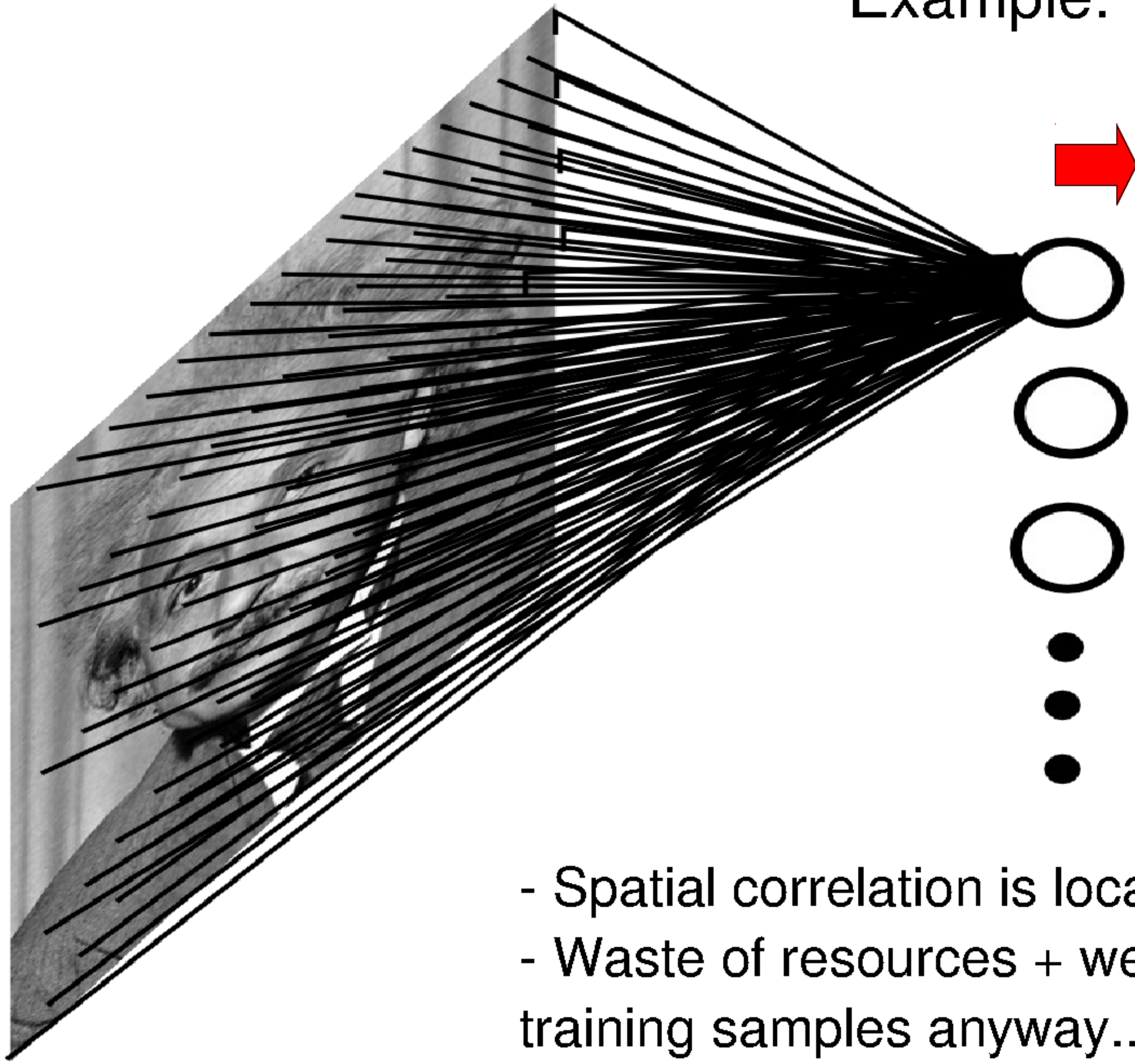
Representation design matters more for all of these

Outline

- Supervised Neural Networks
- Convolutional Neural Networks
- Examples
- Tips

Fully Connected Layer

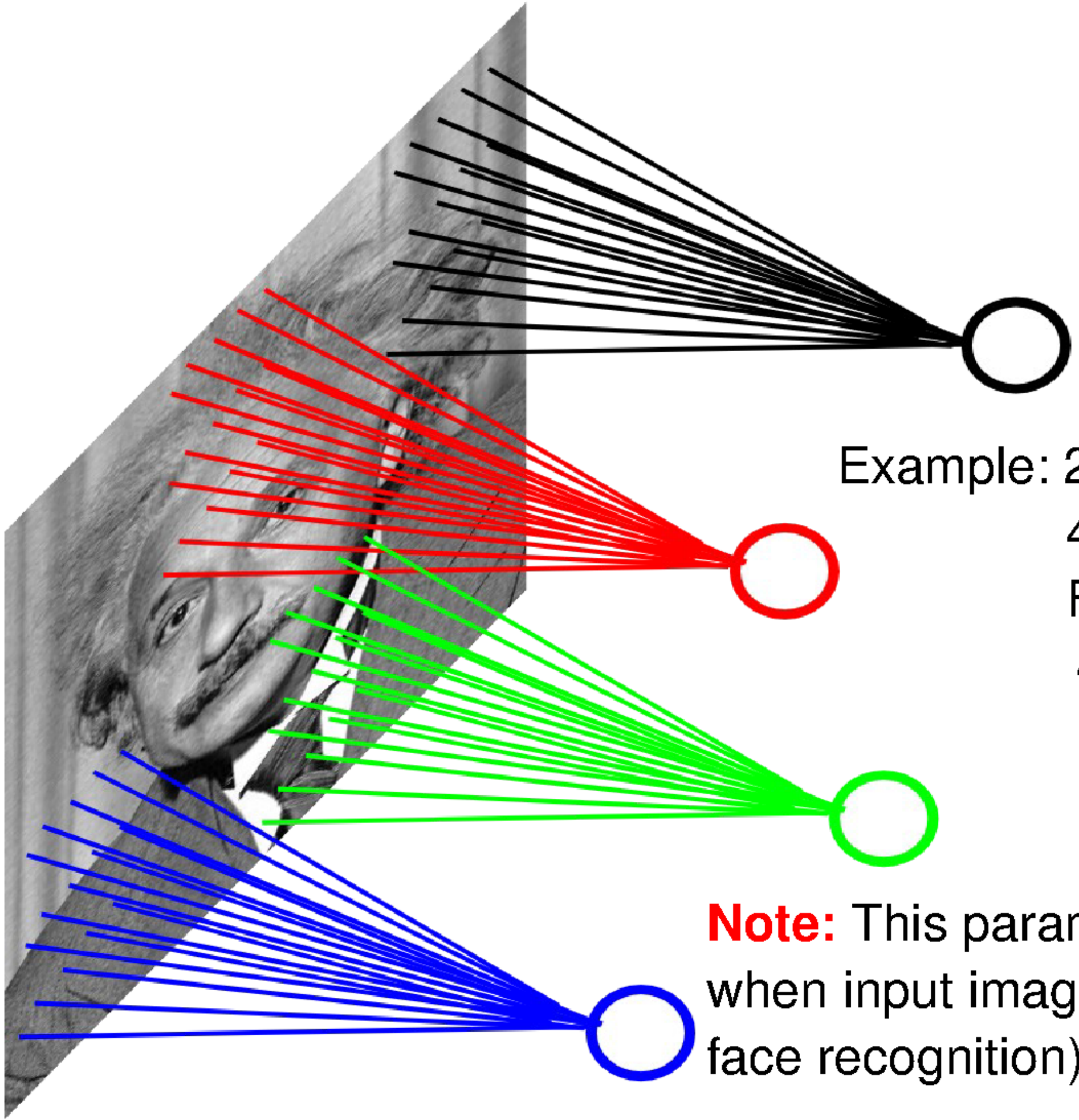
Example: 200x200 image
40K hidden units
→ **~2B parameters**!!!



- Spatial correlation is local
- Waste of resources + we have not enough training samples anyway..

Ranzato

Locally Connected Layer

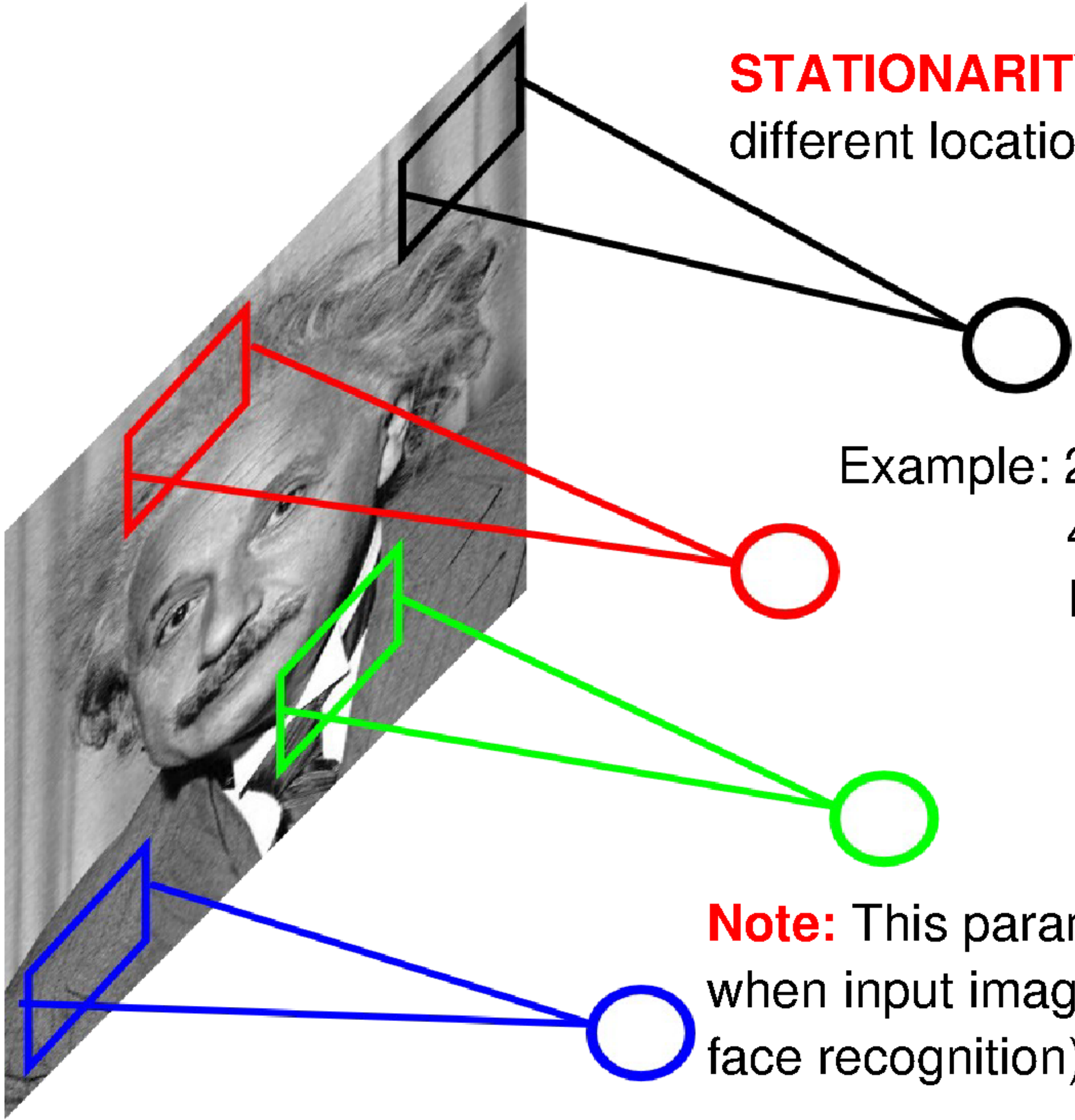


Example: 200x200 image
40K hidden units
Filter size: 10x10
4M parameters

Note: This parameterization is good when input image is registered (e.g., face recognition).

Ranzato

Locally Connected Layer



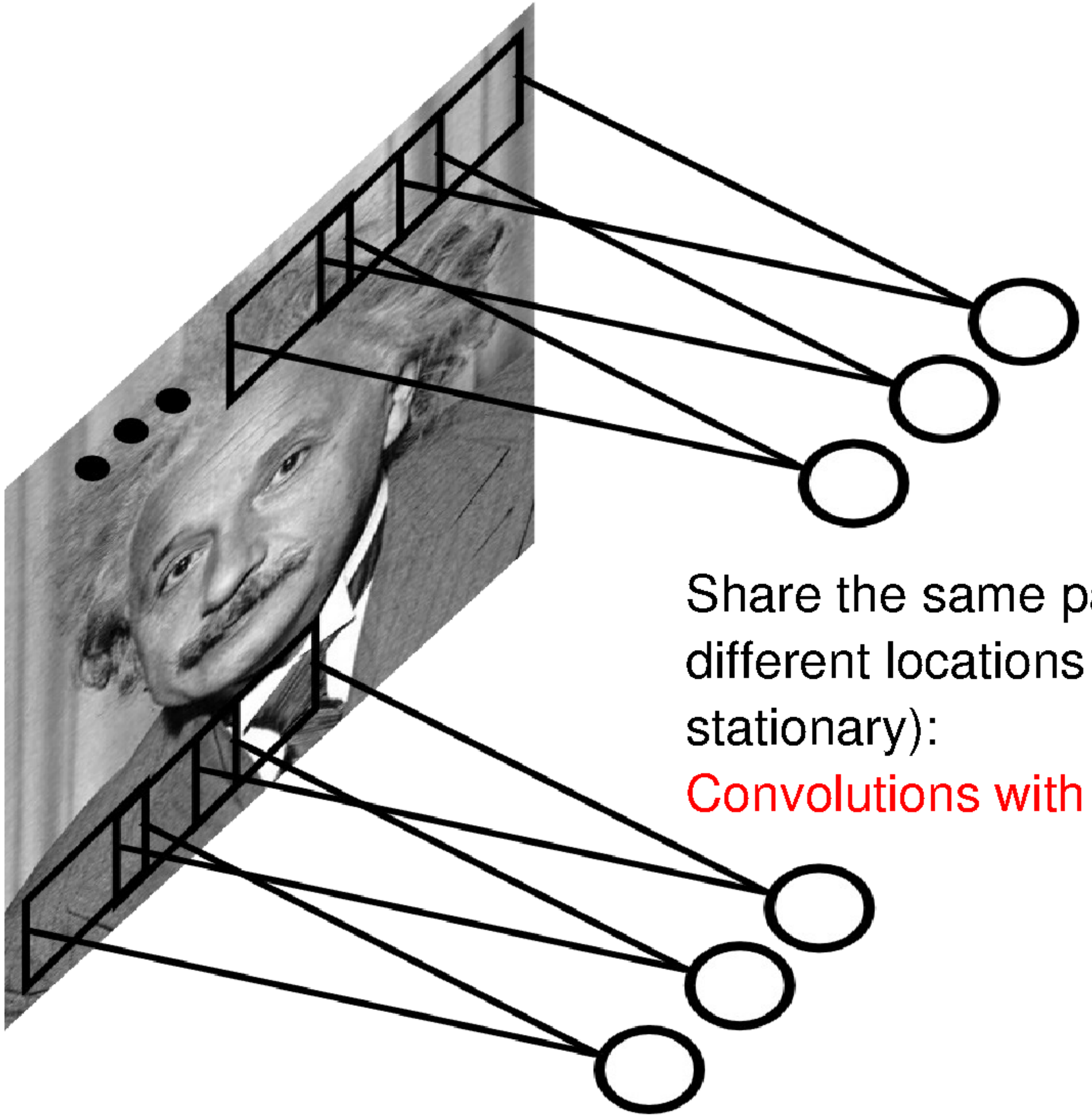
STATIONARITY? Statistics is similar at different locations

Example: 200x200 image
40K hidden units
Filter size: 10x10
4M parameters

Note: This parameterization is good when input image is registered (e.g., face recognition).

Ranzato

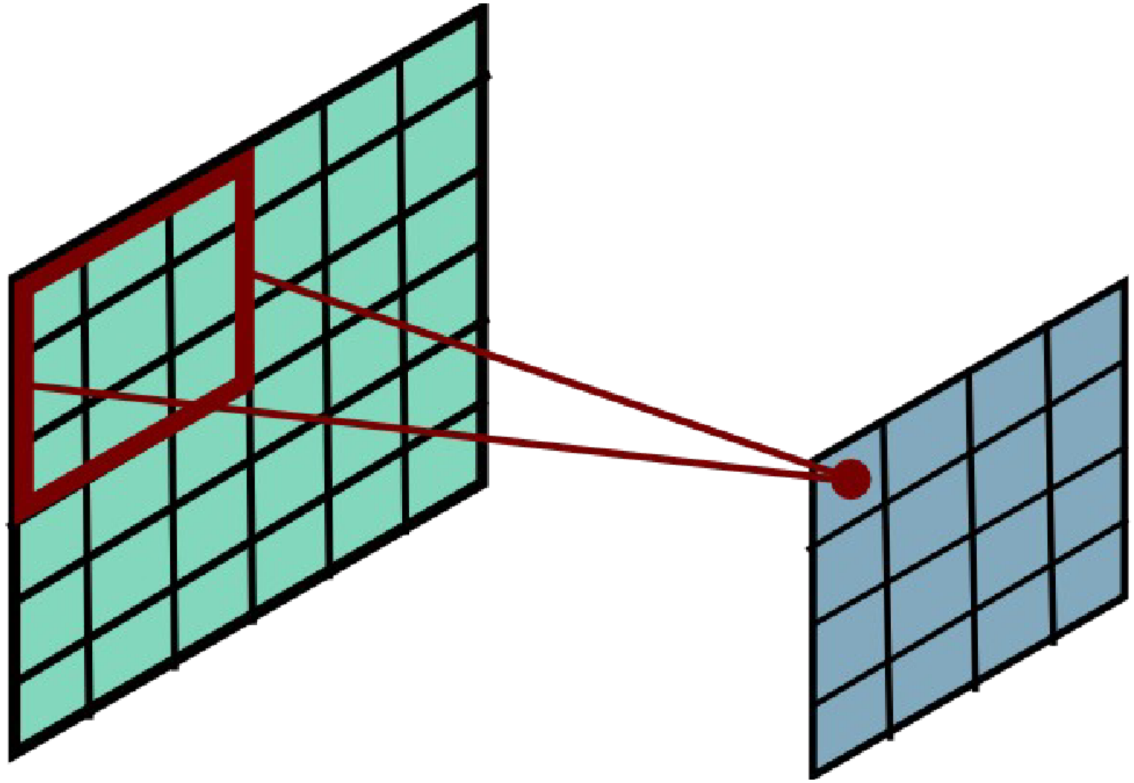
Convolutional Layer



Share the same parameters across different locations (assuming input is stationary):
Convolutions with learned kernels

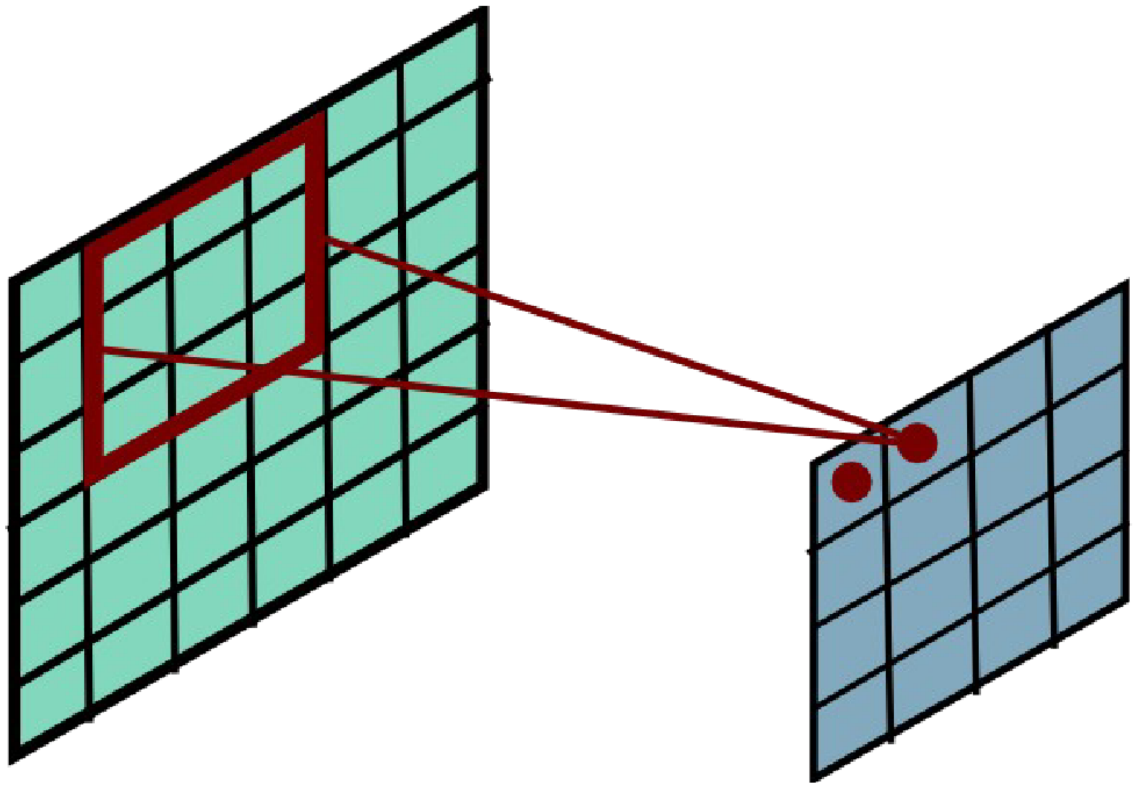
Ranzato

Convolutional Layer



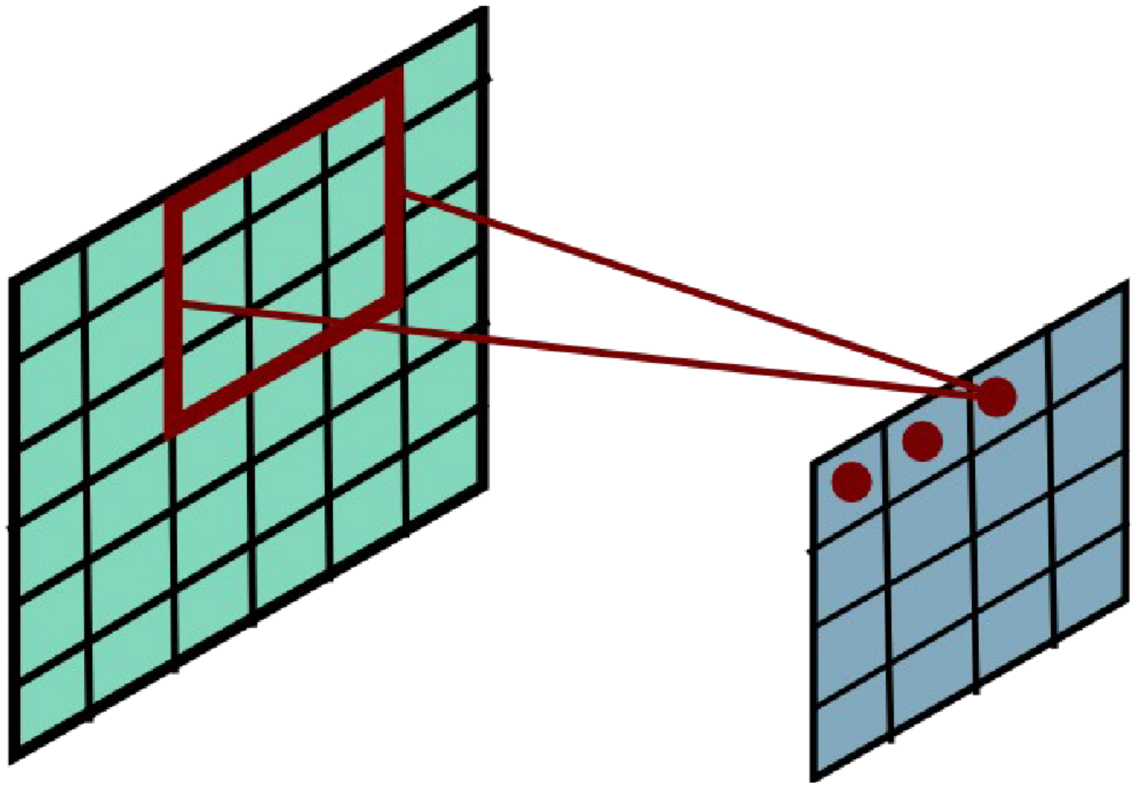
Ranzato

Convolutional Layer



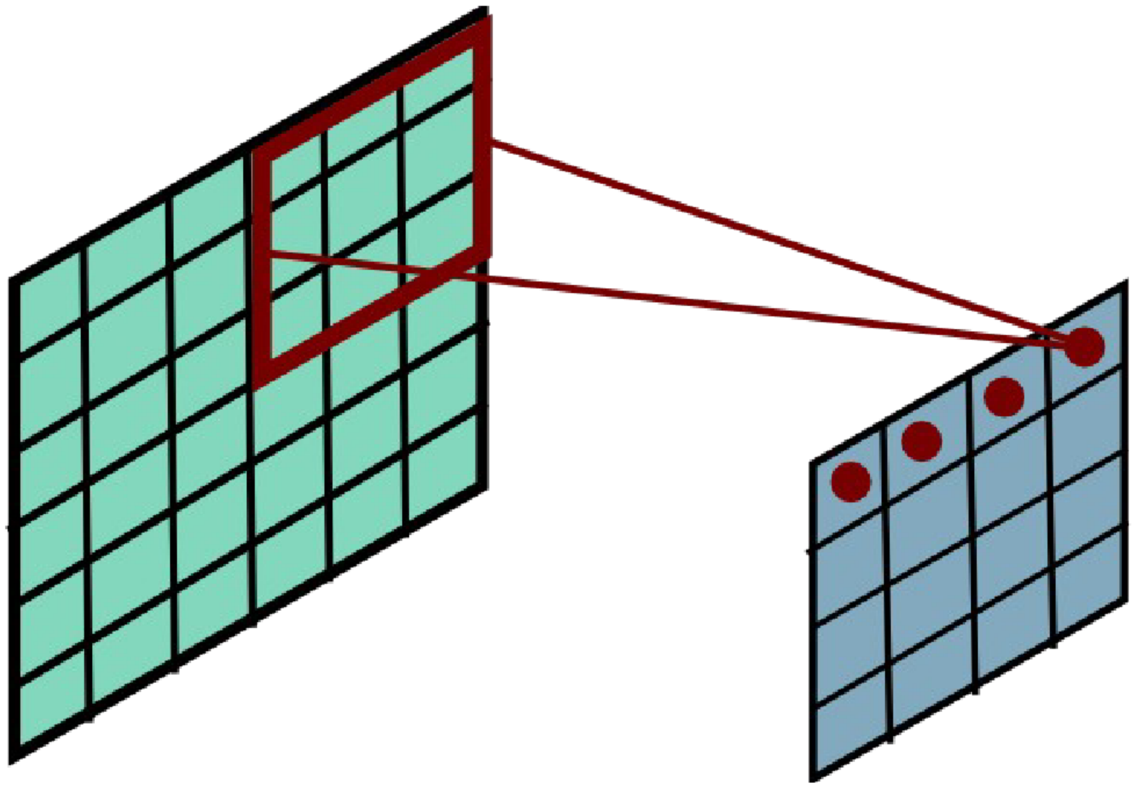
Ranzato

Convolutional Layer



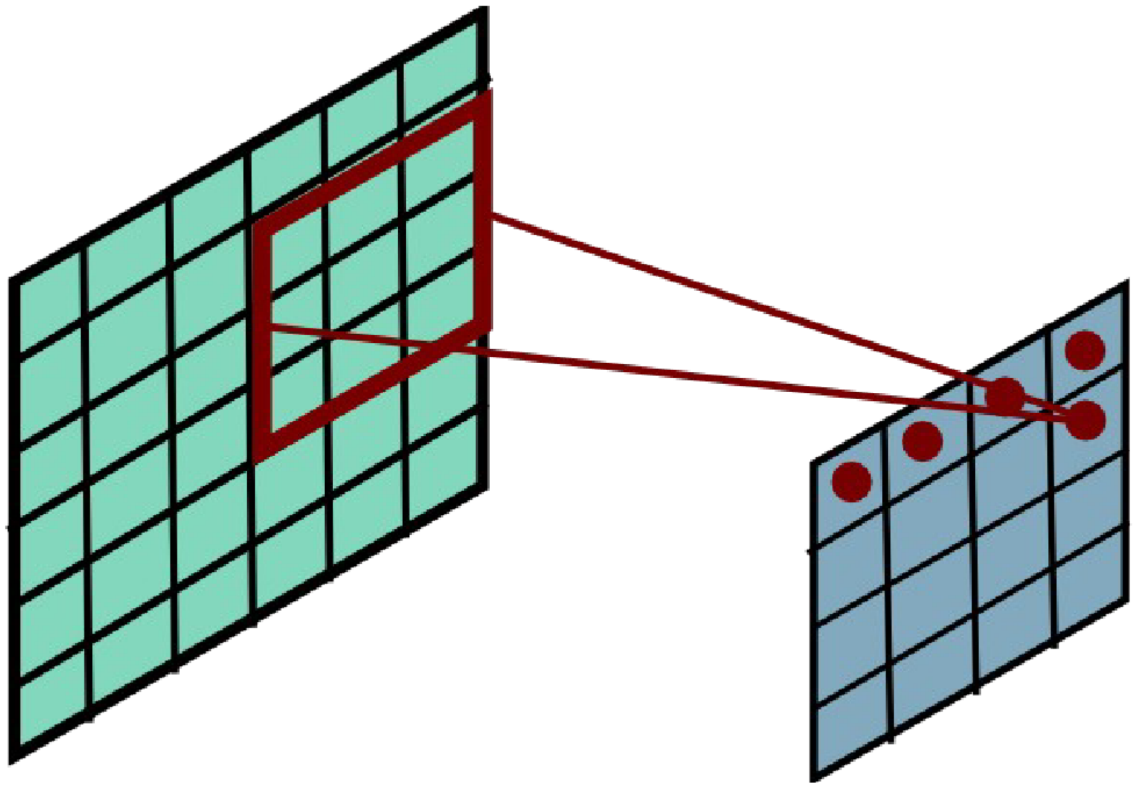
Ranzato

Convolutional Layer

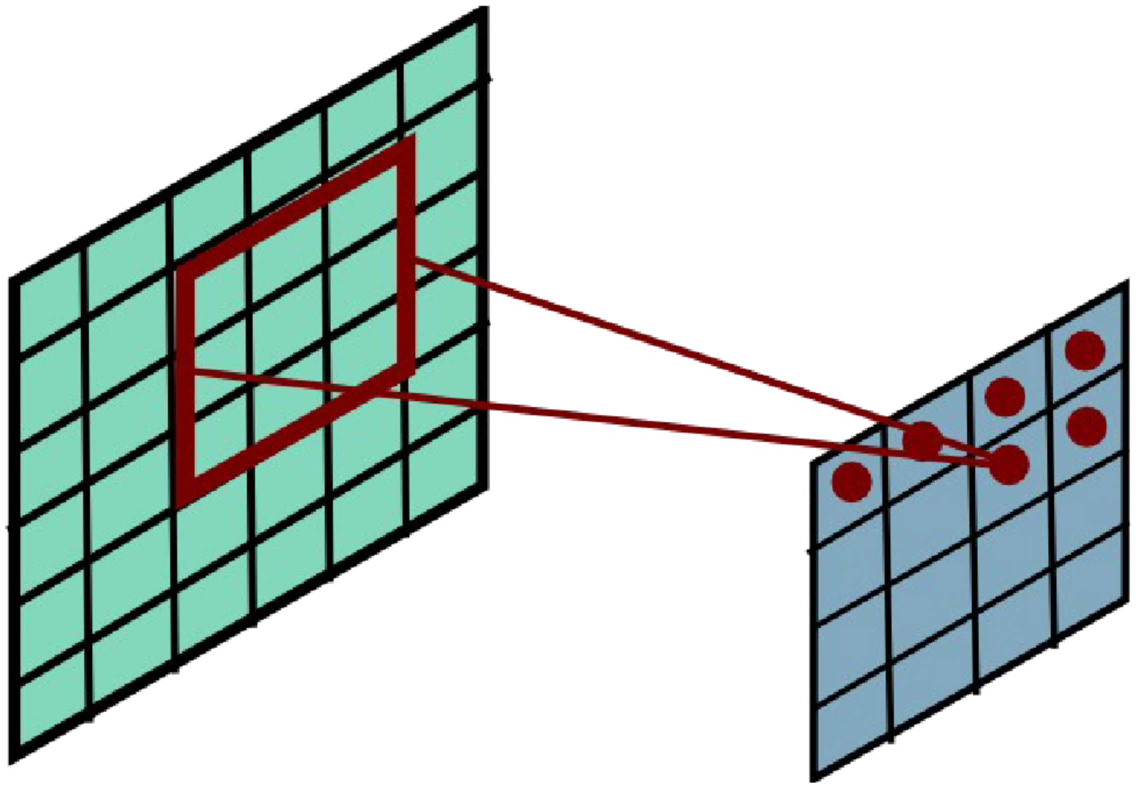


Ranzato

Convolutional Layer

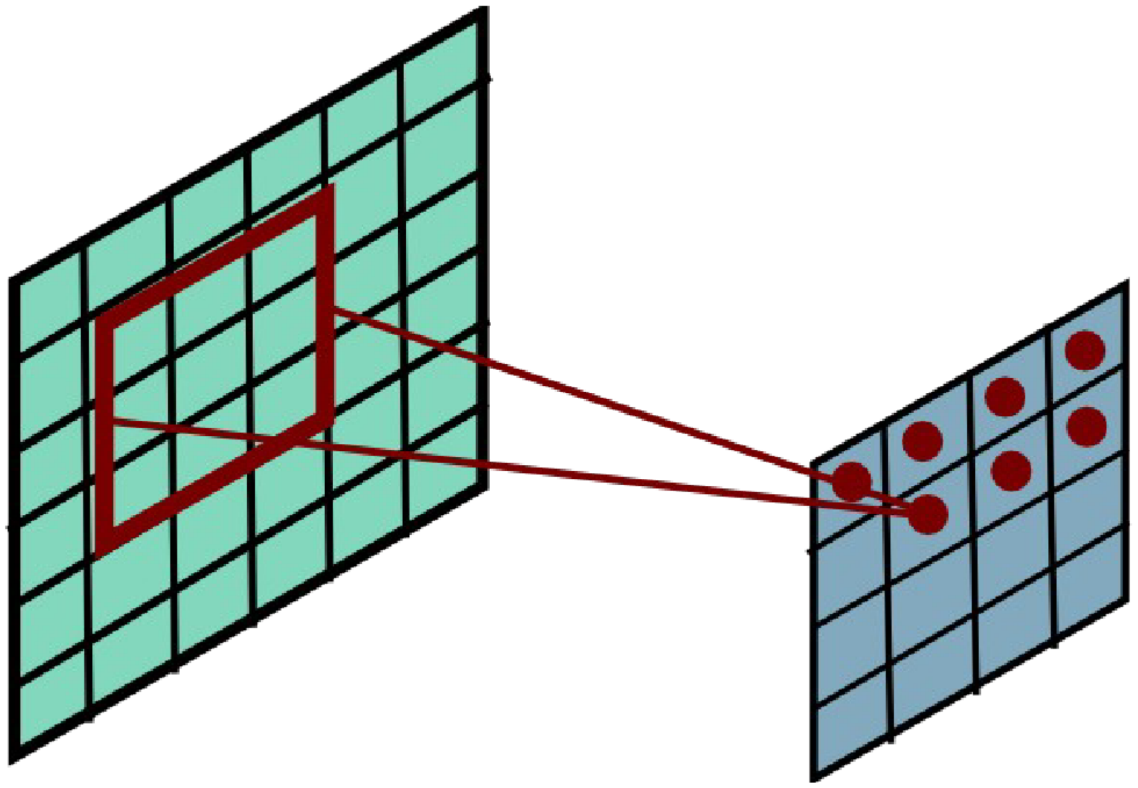


Convolutional Layer



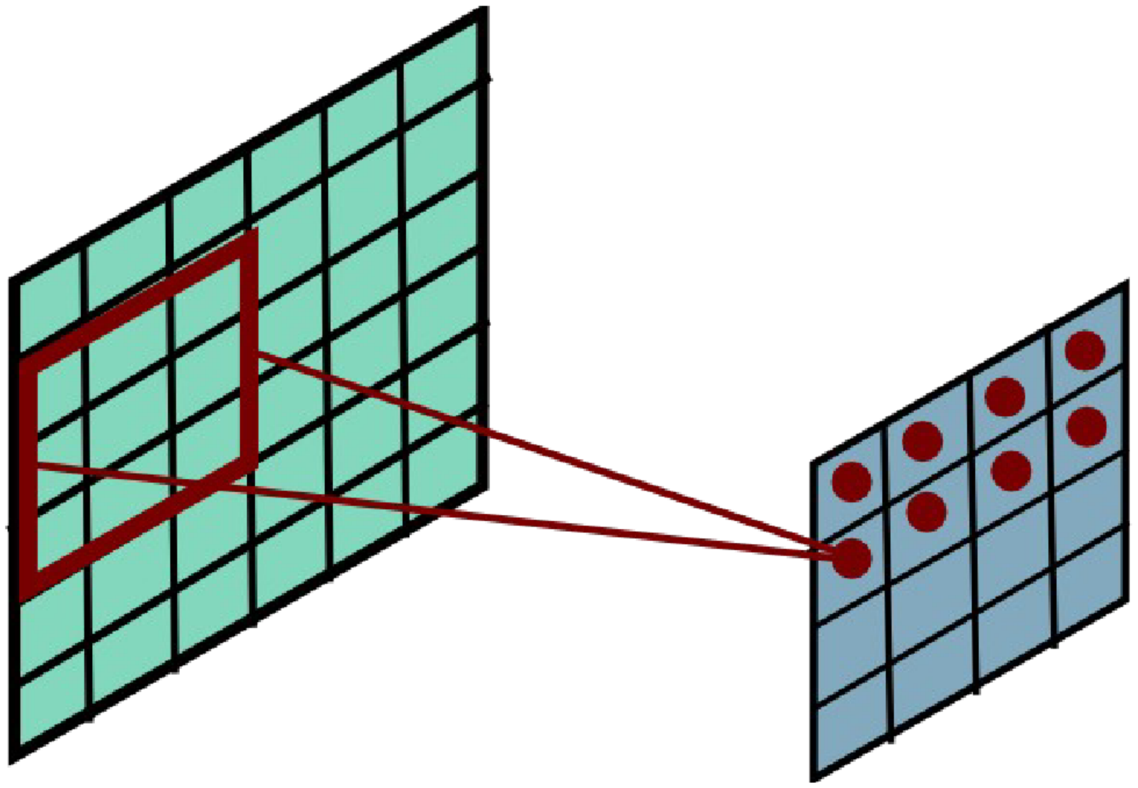
Ranzato

Convolutional Layer



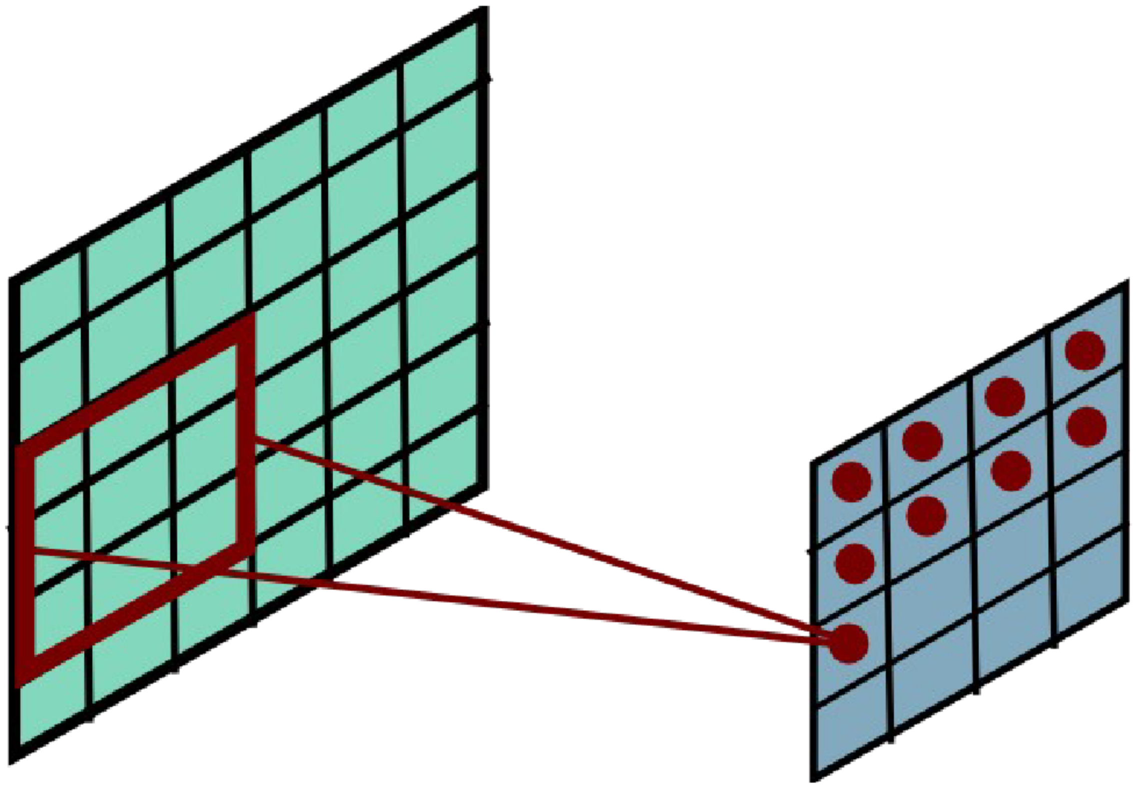
Ranzato

Convolutional Layer



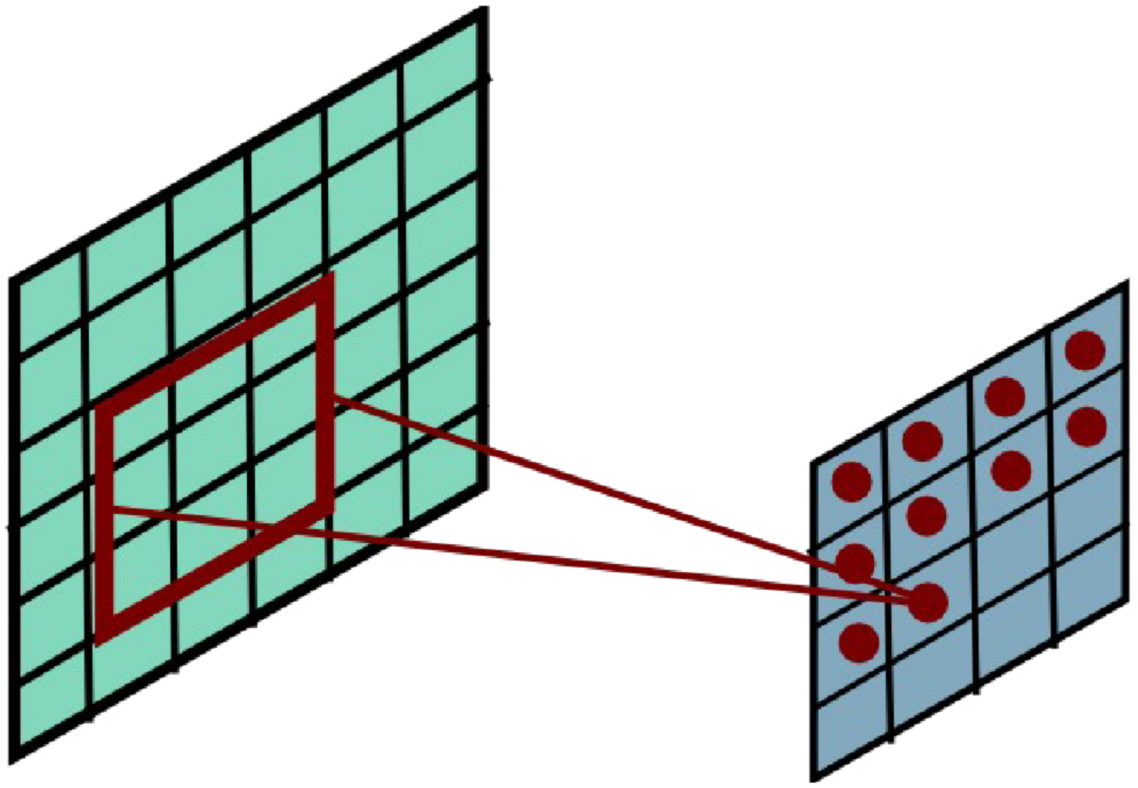
Ranzato

Convolutional Layer



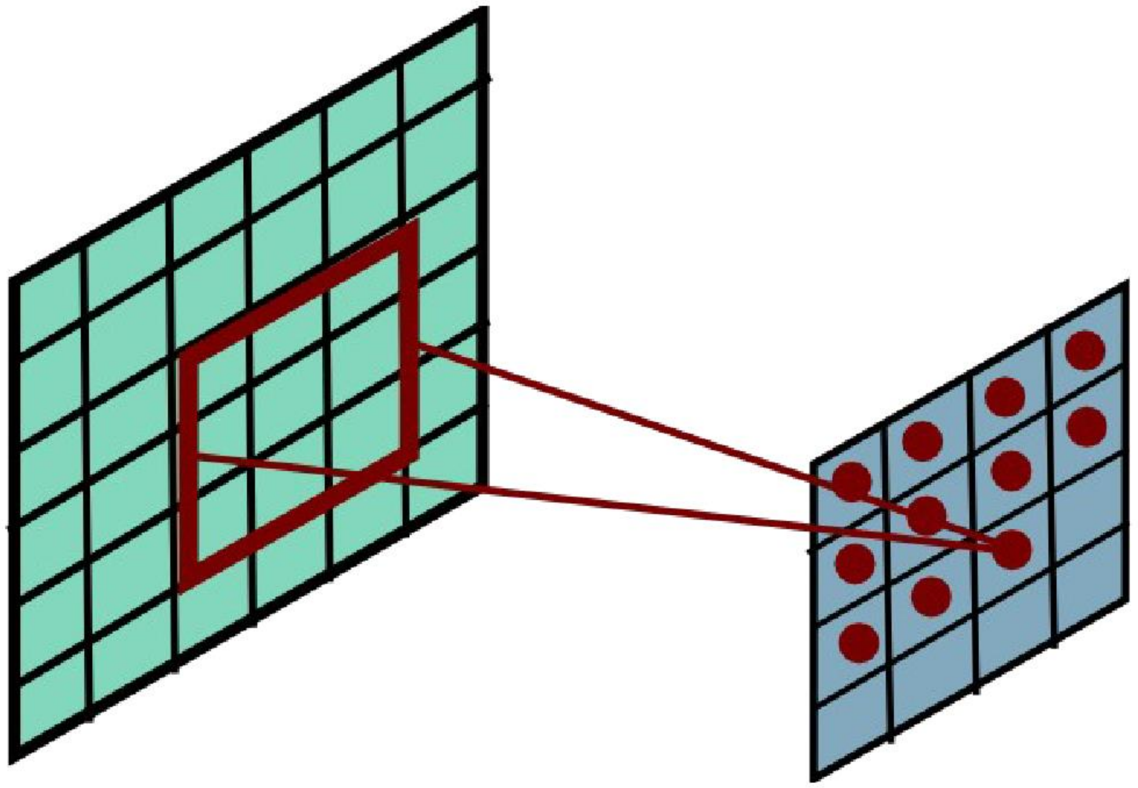
Ranzato

Convolutional Layer



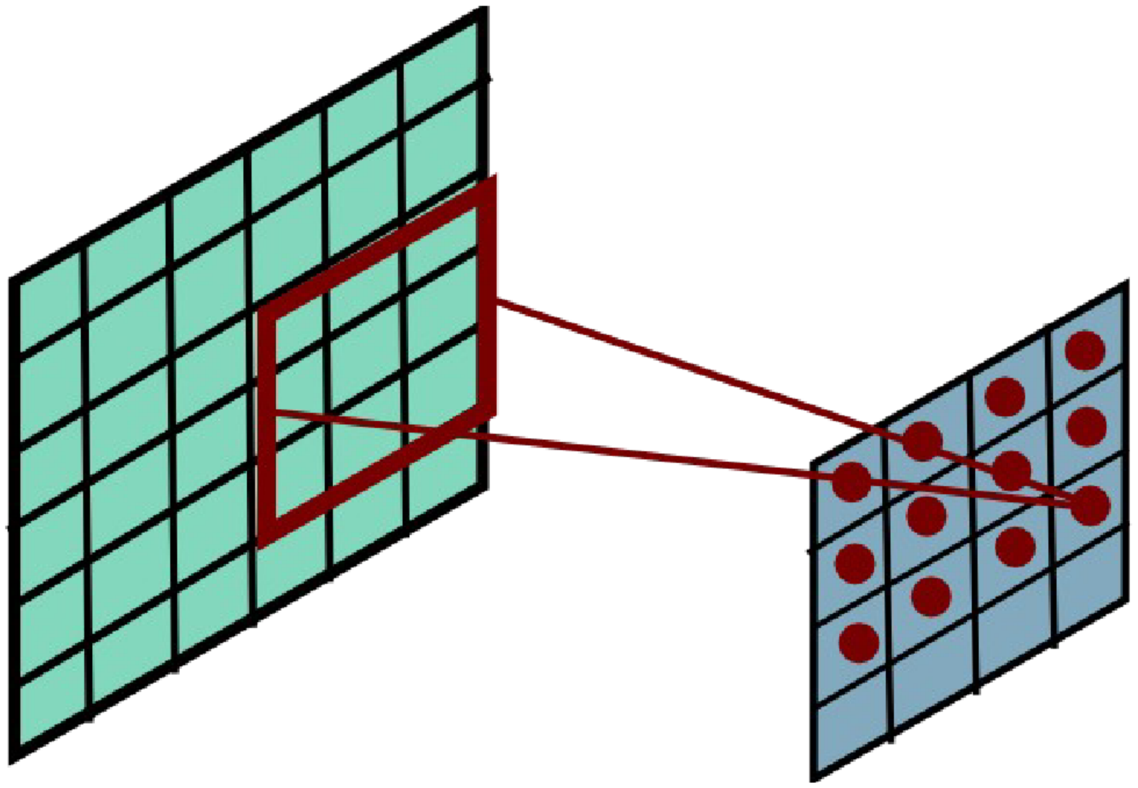
Ranzato

Convolutional Layer



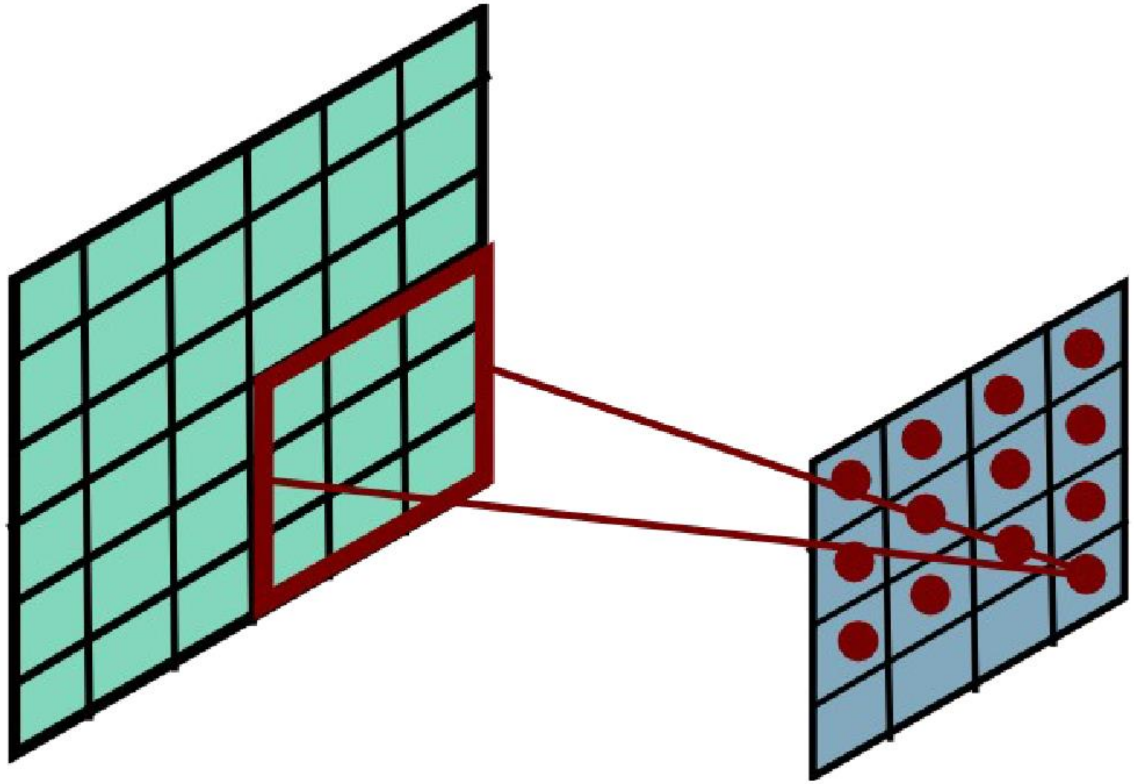
Ranzato

Convolutional Layer



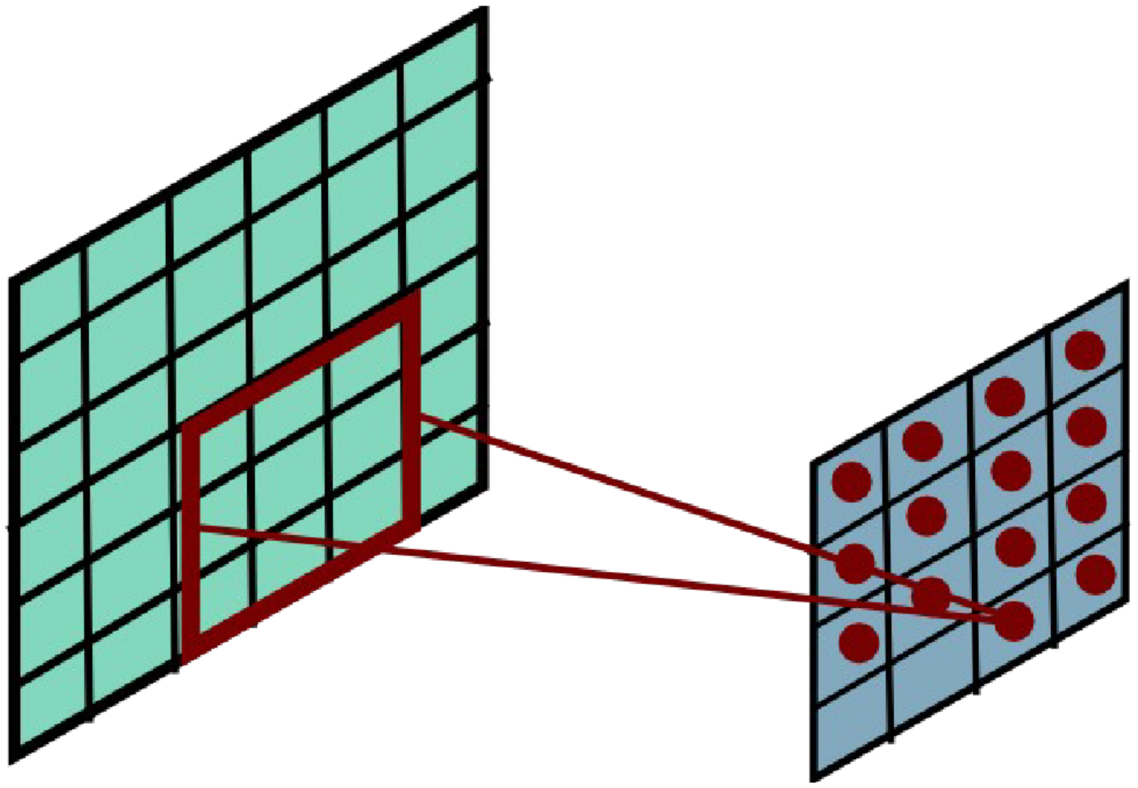
Ranzato

Convolutional Layer



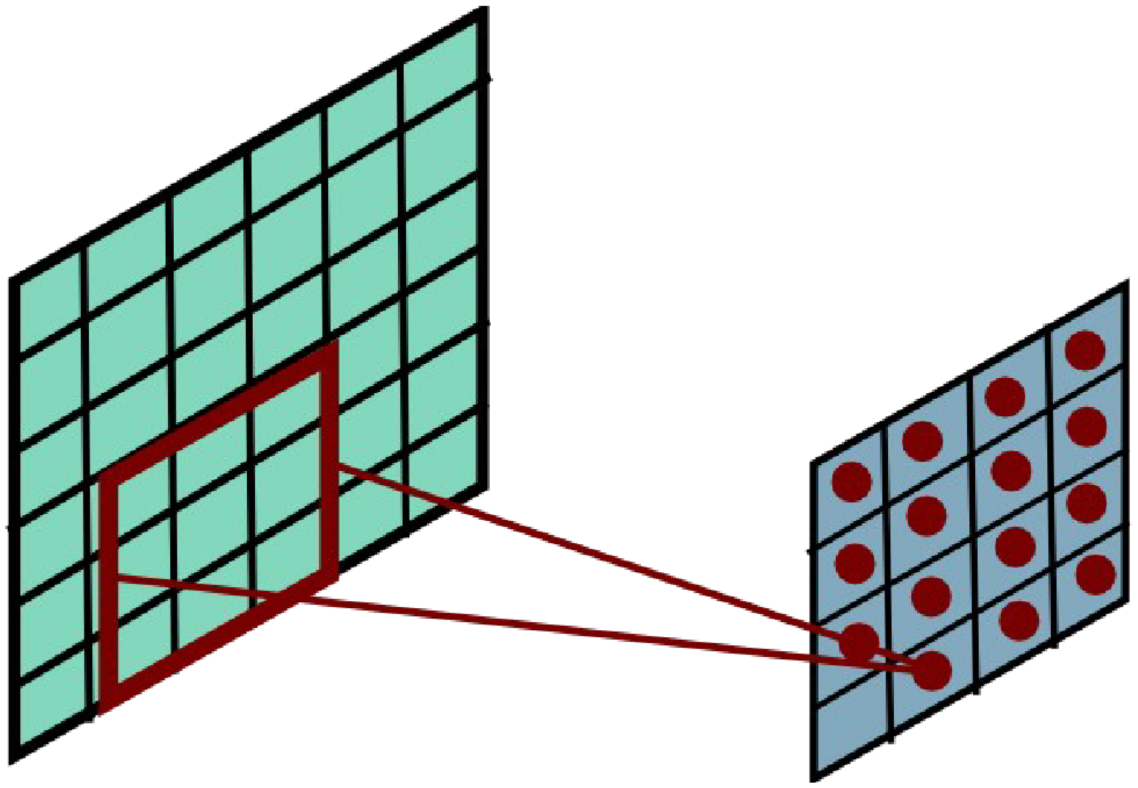
Ranzato

Convolutional Layer



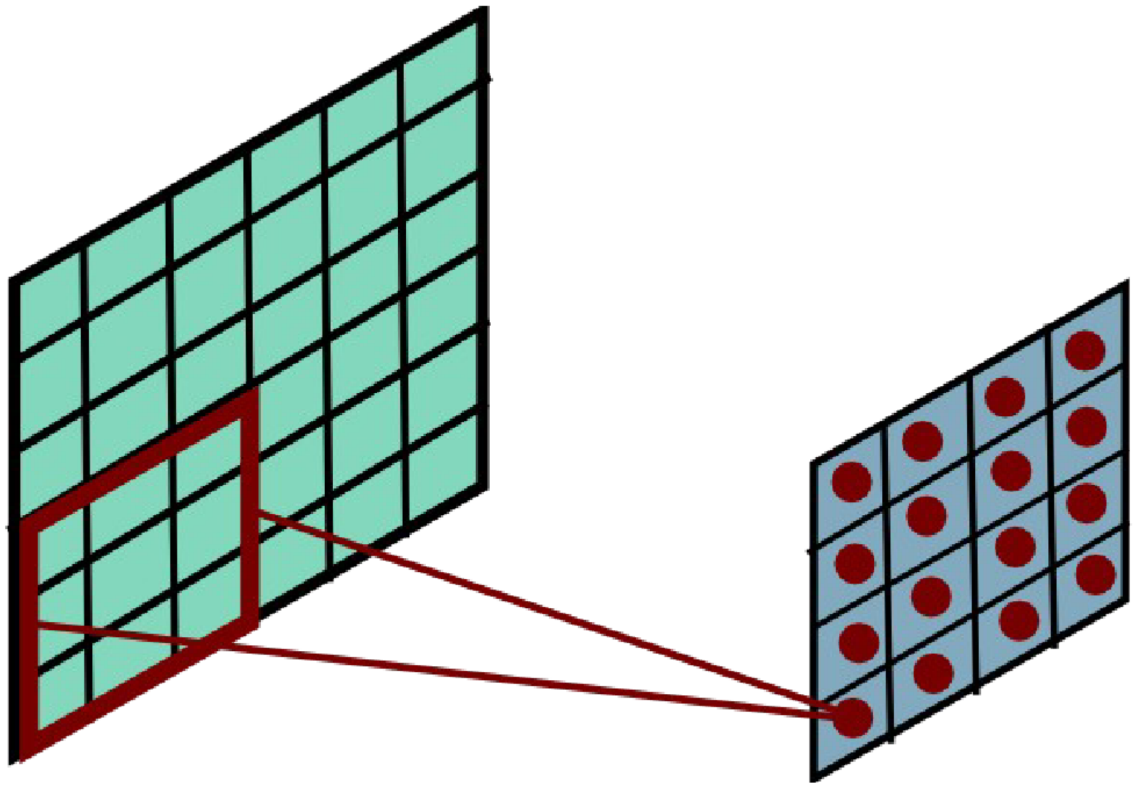
Ranzato

Convolutional Layer



Ranzato

Convolutional Layer



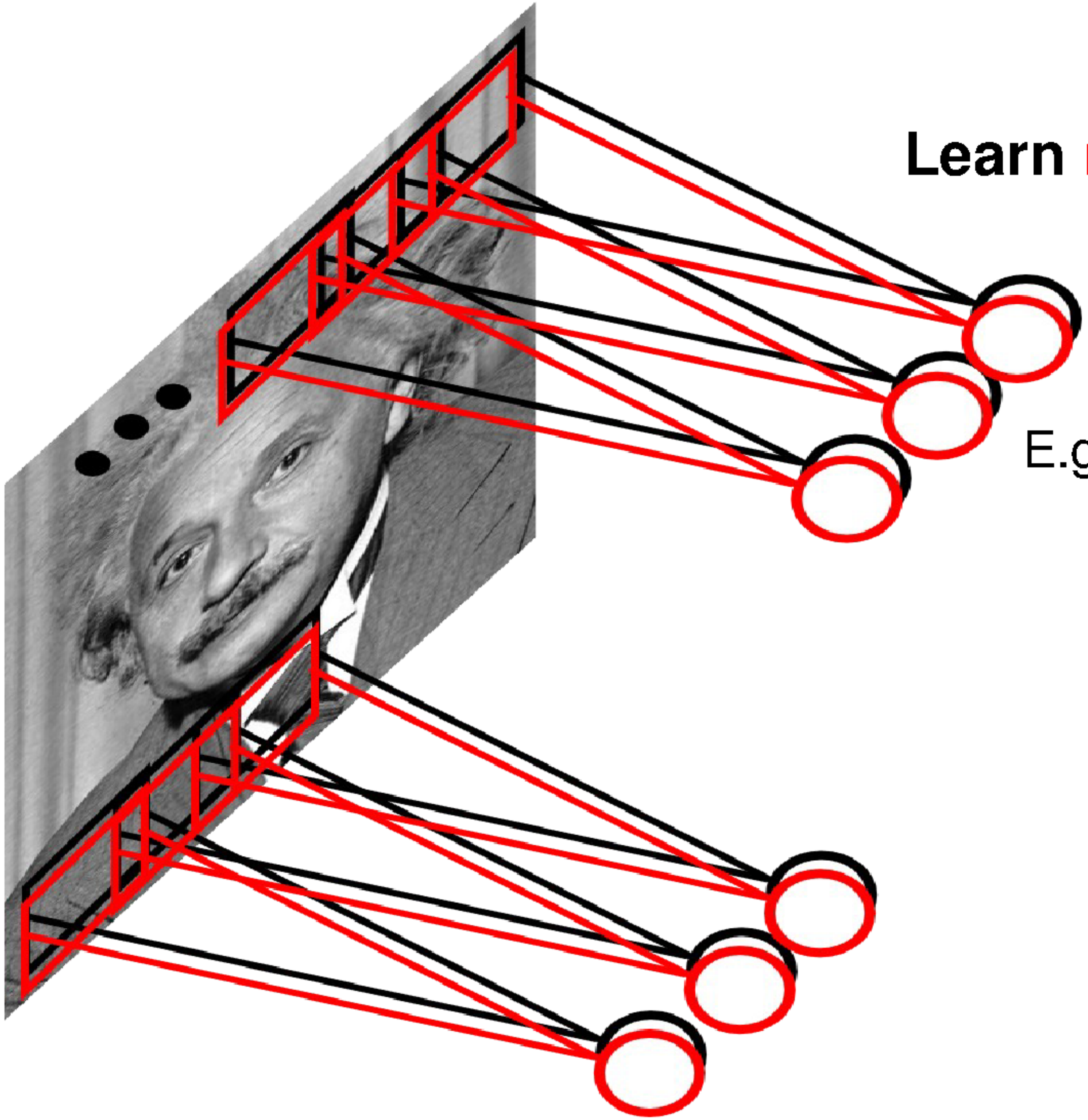
Ranzato

Convolutional Layer

$$* \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} =$$

Ranzato

Convolutional Layer



Learn multiple filters.

E.g.: 200x200 image
100 Filters
Filter size: 10x10
10K parameters

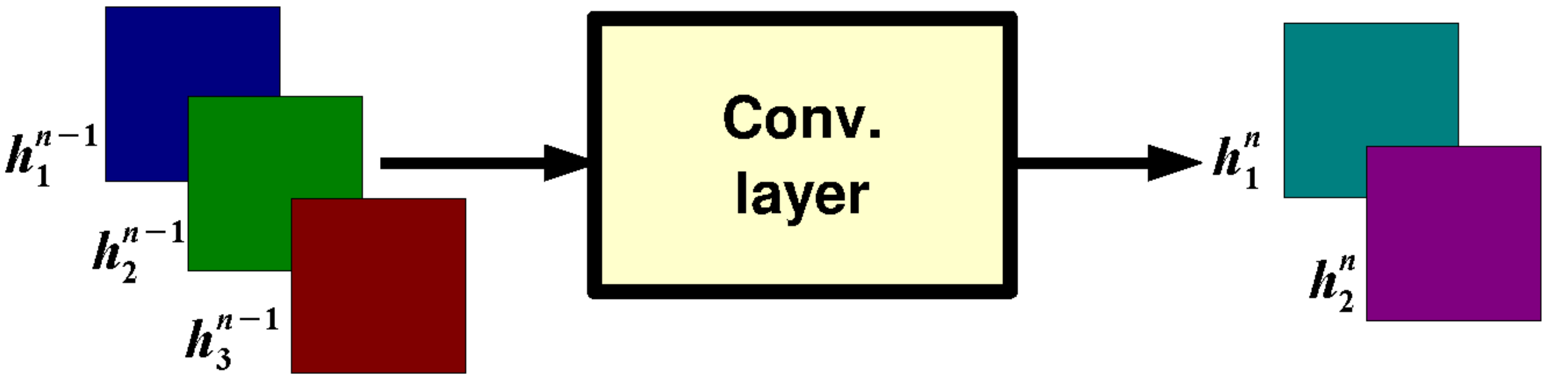
Convolutional Layer

$$h_j^n = max\,(0, \sum_{k=1}^{K} h_k^{n-1} * w_{kj}^n)$$

output feature map

input feature map

kernel



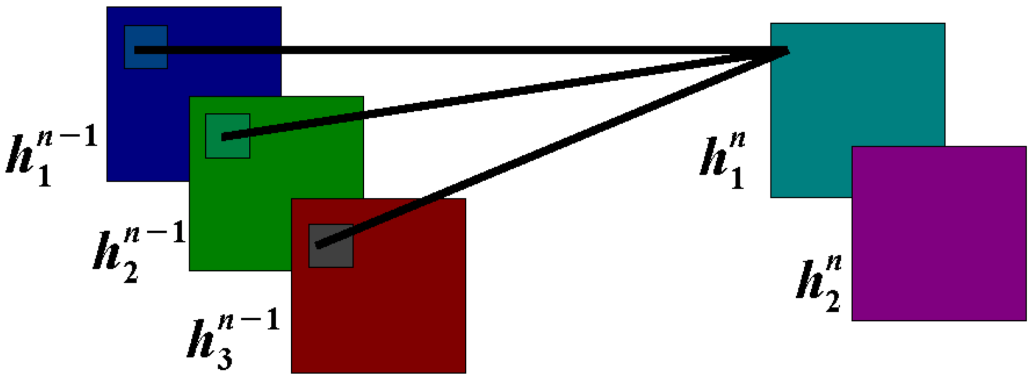
Convolutional Layer

$$h_j^n = max\,(0, \sum_{k=1}^{K} h_k^{n-1} * w_{kj}^n)$$

output feature map

input feature map

kernel



Ranzato

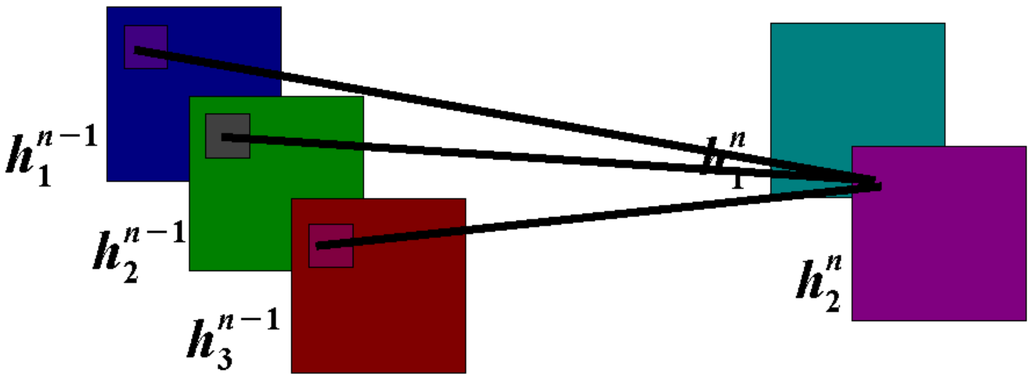
Convolutional Layer

$$h_j^n = max\left(0, \sum_{k=1}^{K} h_k^{n-1} * w_{kj}^n\right)$$

output feature map

input feature map

kernel



Convolutional Layer

Question: What is the size of the output? What's the computational cost?

Answer: It is proportional to the number of filters and depends on the stride. If kernels have size KxK, input has size DxD, stride is 1, and there are M input feature maps and N output feature maps then:
- the input has size M@DxD
- the output has size N@(D-K+1)x(D-K+1)
- the kernels have MxNxKxK coefficients (which have to be learned)
- cost: M*K*K*N*(D-K+1)*(D-K+1)

Question: How many feature maps? What's the size of the filters?

Answer: Usually, there are more output feature maps than input feature maps. Convolutional layers can increase the number of hidden units by big factors (and are expensive to compute).
The size of the filters has to match the size/scale of the patterns we want to detect (task dependent).

Ranzato

Key Ideas

A standard neural net applied to images:

- scales quadratically with the size of the input
- does not leverage stationarity

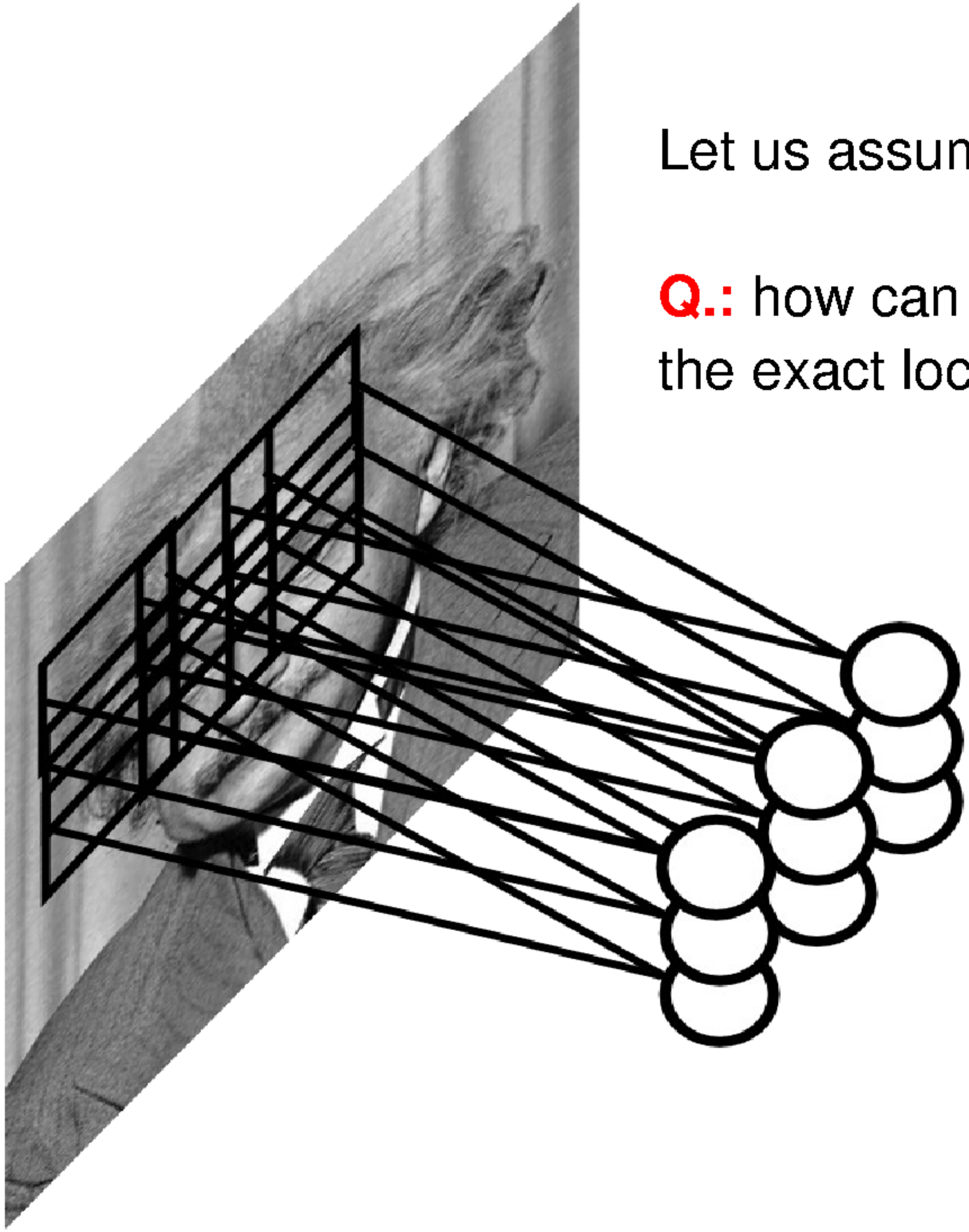
Solution:

- connect each hidden unit to a small patch of the input
- share the weight across space

This is called: **convolutional layer.**
A network with convolutional layers is called **convolutional network.**

LeCun et al. “Gradient-based learning applied to document recognition” IEEE 1998

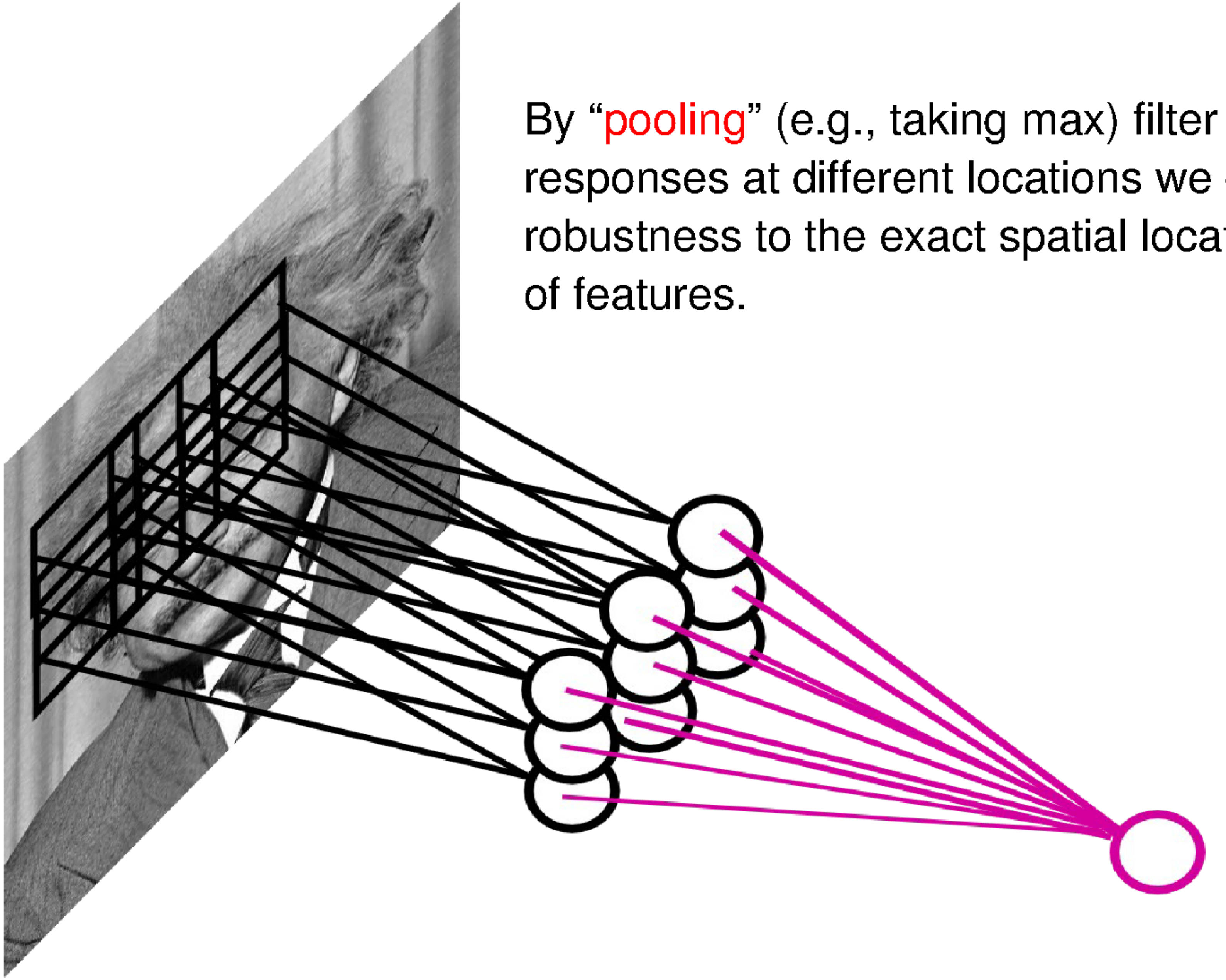
Pooling Layer



Let us assume filter is an "eye" detector.

Q.: how can we make the detection robust to the exact location of the eye?

Pooling Layer



By "pooling" (e.g., taking max) filter responses at different locations we gain robustness to the exact spatial location of features.

Ranzato

Pooling Layer: Examples

Max-pooling:

$$h_j^n(x,y) = max_{\bar{x} \in N(x), \bar{y} \in N(y)} h_j^{n-1}(\bar{x}, \bar{y})$$

Average-pooling:

$$h_j^n(x,y) = 1/K \sum_{\bar{x} \in N(x), \bar{y} \in N(y)} h_j^{n-1}(\bar{x}, \bar{y})$$

L2-pooling:

$$h_j^n(x,y) = \sqrt{\sum_{\bar{x} \in N(x), \bar{y} \in N(y)} h_j^{n-1}(\bar{x}, \bar{y})^2}$$

L2-pooling over features:

$$h_j^n(x,y) = \sqrt{\sum_{k \in N(j)} h_k^{n-1}(x, y)^2}$$

Pooling Layer

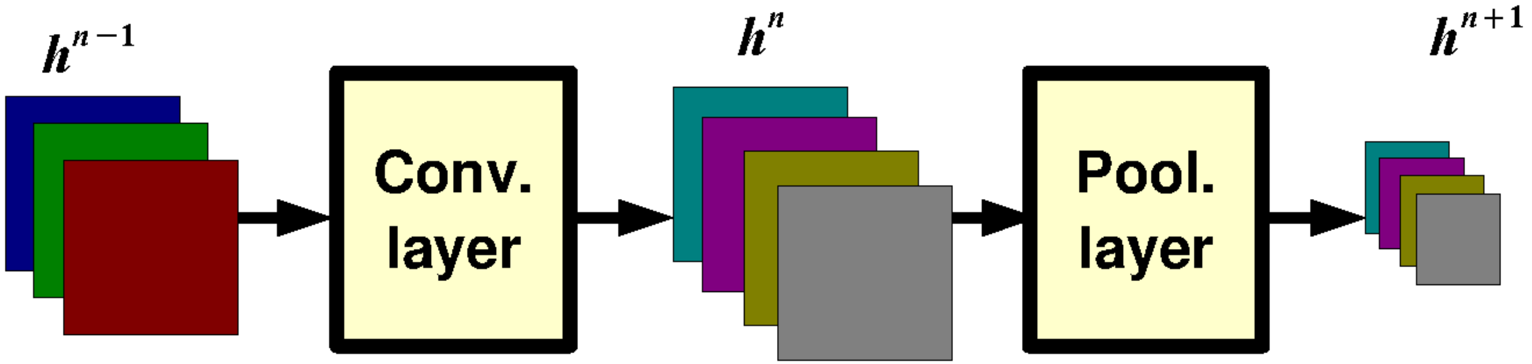
Question: What is the size of the output? What's the computational cost?

Answer: The size of the output depends on the stride between the pools. For instance, if pools do not overlap and have size KxK, and the input has size DxD with M input feature maps, then:
- output is M@(D/K)x(D/K)
- the computational cost is proportional to the size of the input (negligible compared to a convolutional layer)

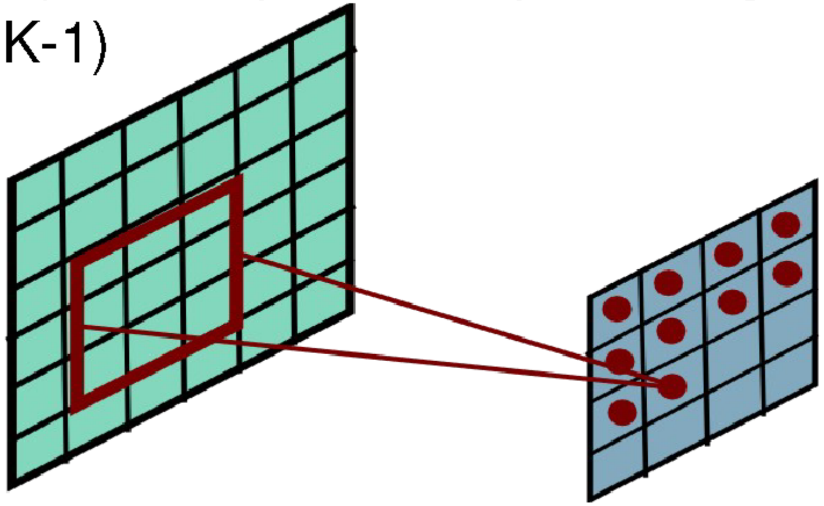
Question: How should I set the size of the pools?

Answer: It depends on how much “invariant” or robust to distortions we want the representation to be. It is best to pool slowly (via a few stacks of conv-pooling layers).

Pooling Layer: Receptive Field Size

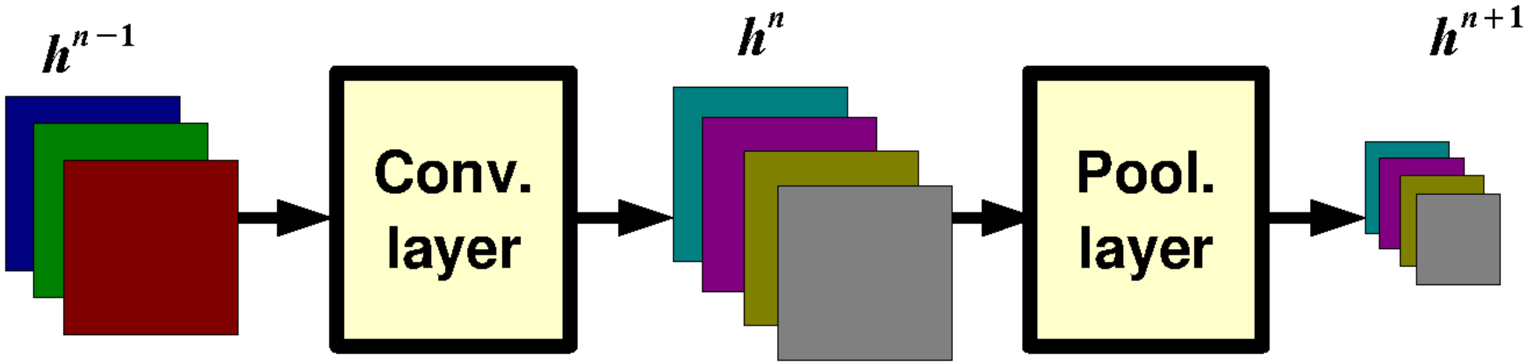


If convolutional filters have size KxK and stride 1, and pooling layer has pools of size PxP, then each unit in the pooling layer depends upon a patch (at the input of the preceding conv. layer) of size: (P+K-1)x(P+K-1)

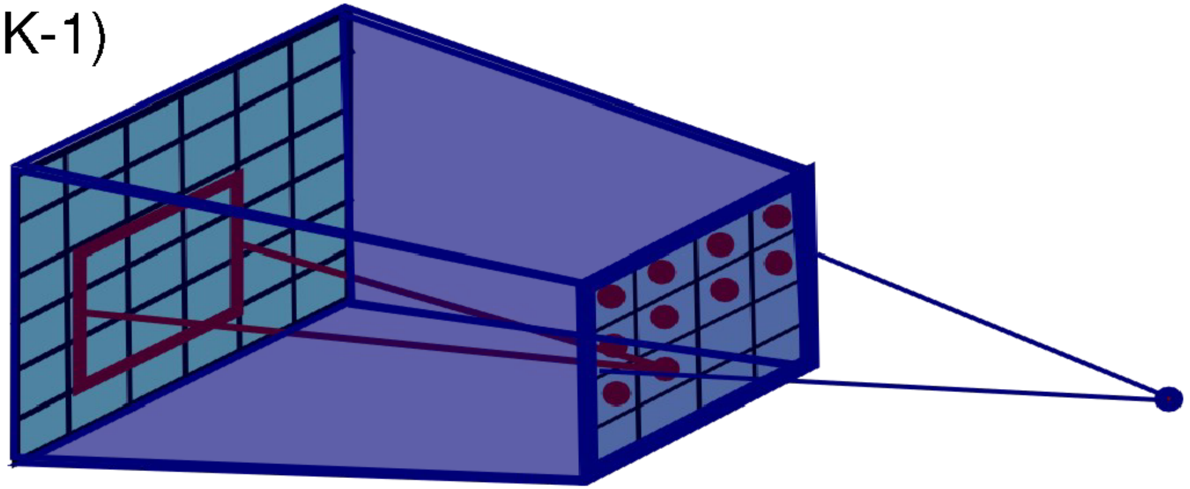


Ranzato

Pooling Layer: Receptive Field Size



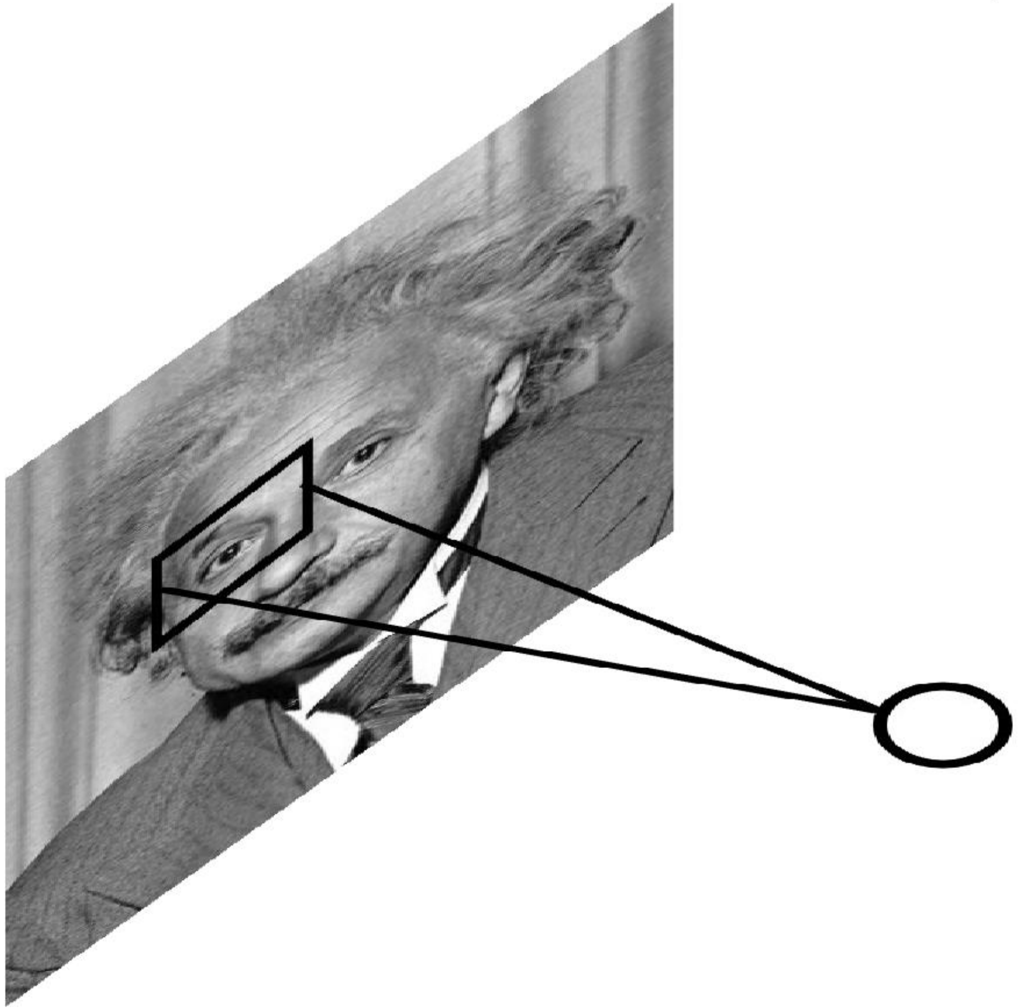
If convolutional filters have size KxK and stride 1, and pooling layer has pools of size PxP, then each unit in the pooling layer depends upon a patch (at the input of the preceding conv. layer) of size: (P+K-1)x(P+K-1)



Ranzato

Local Contrast Normalization

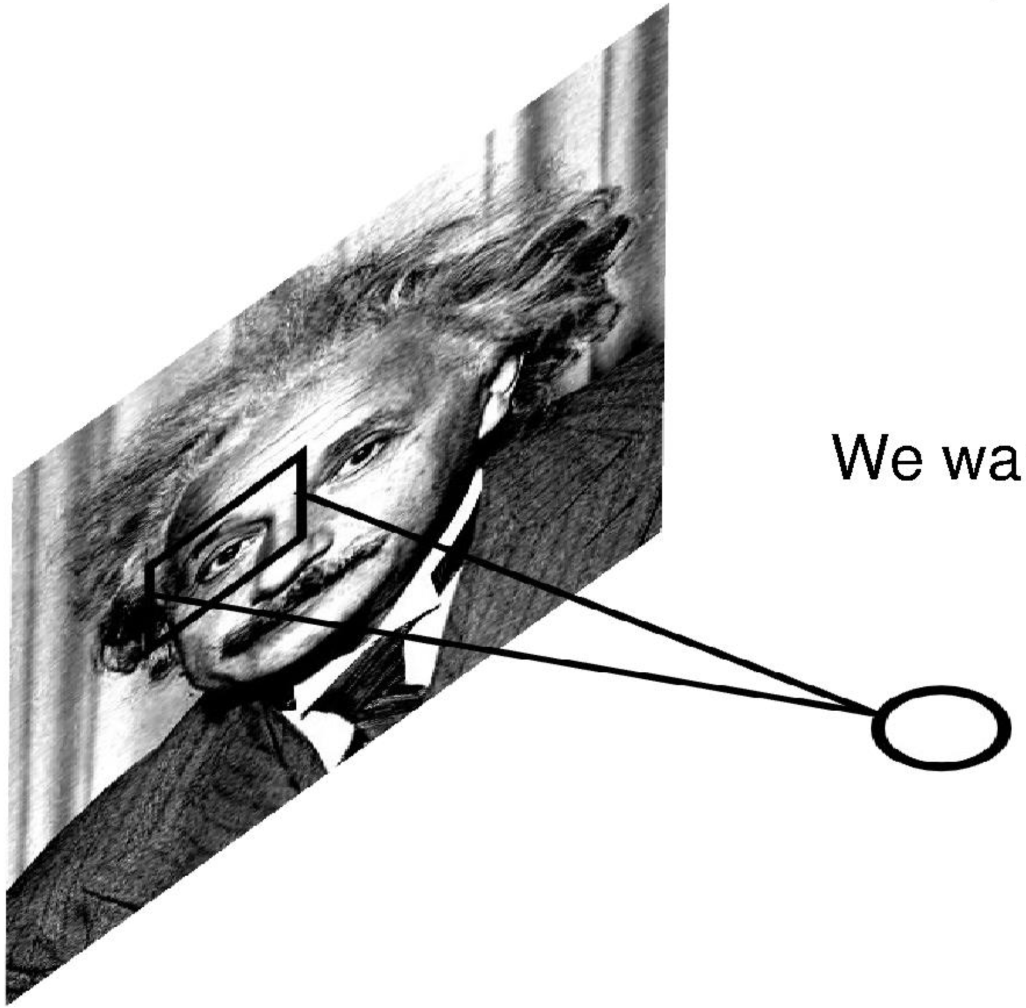
$$h^{i+1}(x,y) = \frac{h^i(x,y) - m^i(N(x,y))}{\sigma^i(N(x,y))}$$



Ranzato

Local Contrast Normalization

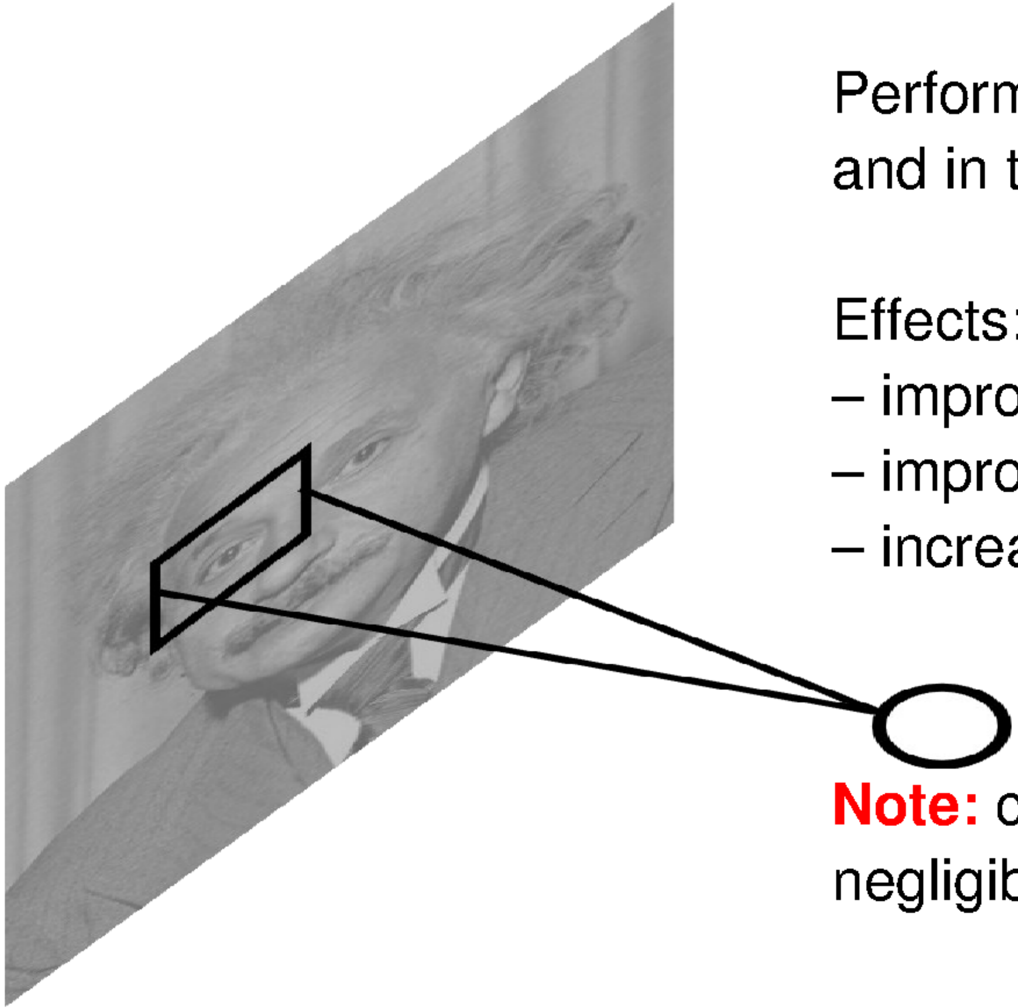
$$h^{i+1}(x,y) = \frac{h^i(x,y) - m^i(N(x,y))}{\sigma^i(N(x,y))}$$



We want the same response.

Local Contrast Normalization

$$h^{i+1}(x,y) = \frac{h^i(x,y) - m^i(N(x,y))}{\sigma^i(N(x,y))}$$



Performed also across features and in the higher layers..

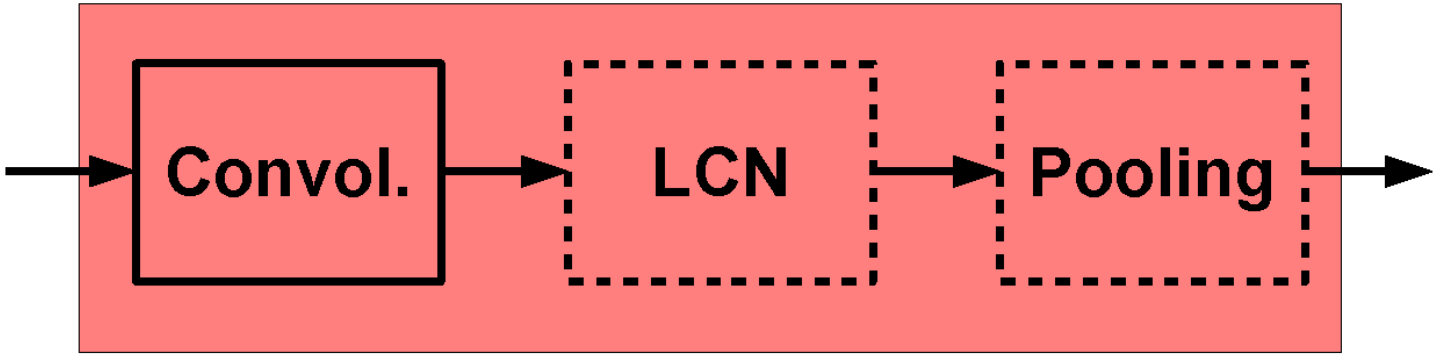
Effects:
– improves invariance
– improves optimization
– increases sparsity

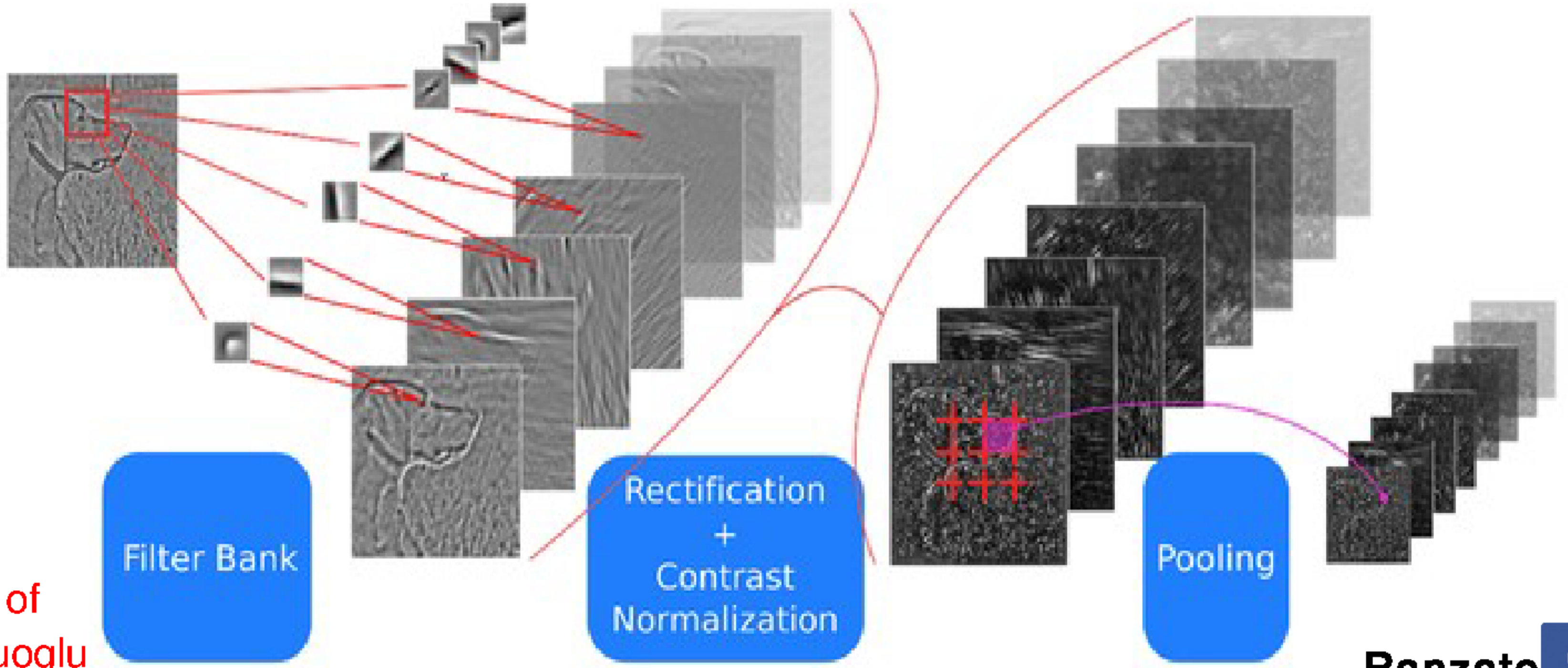
Note: computational cost is negligible w.r.t. conv. layer.

Ranzato

ConvNets: Typical Stage

One stage (zoom)

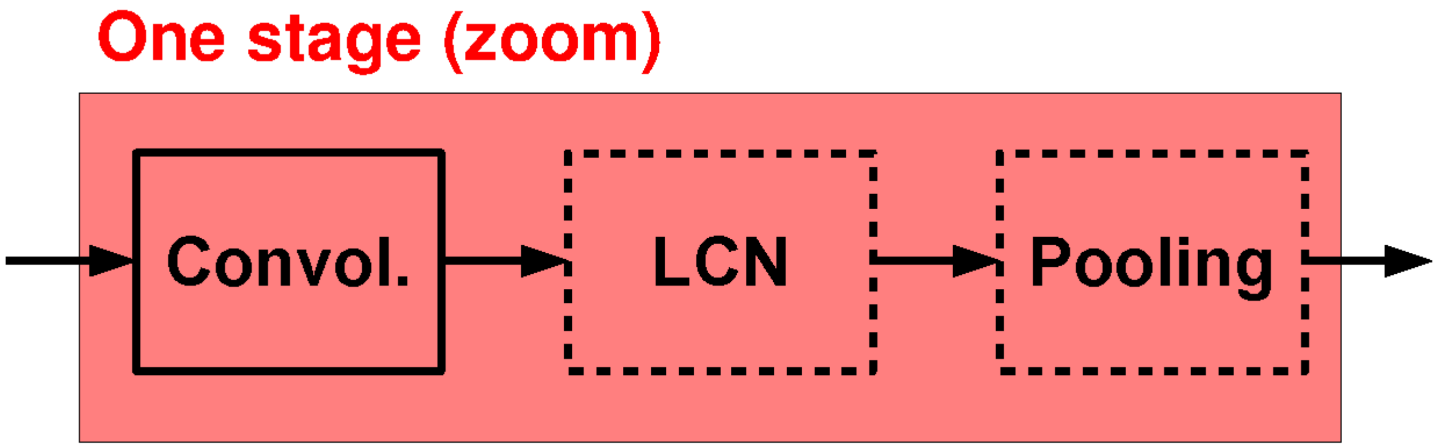




courtesy of
K. Kavukcuoglu

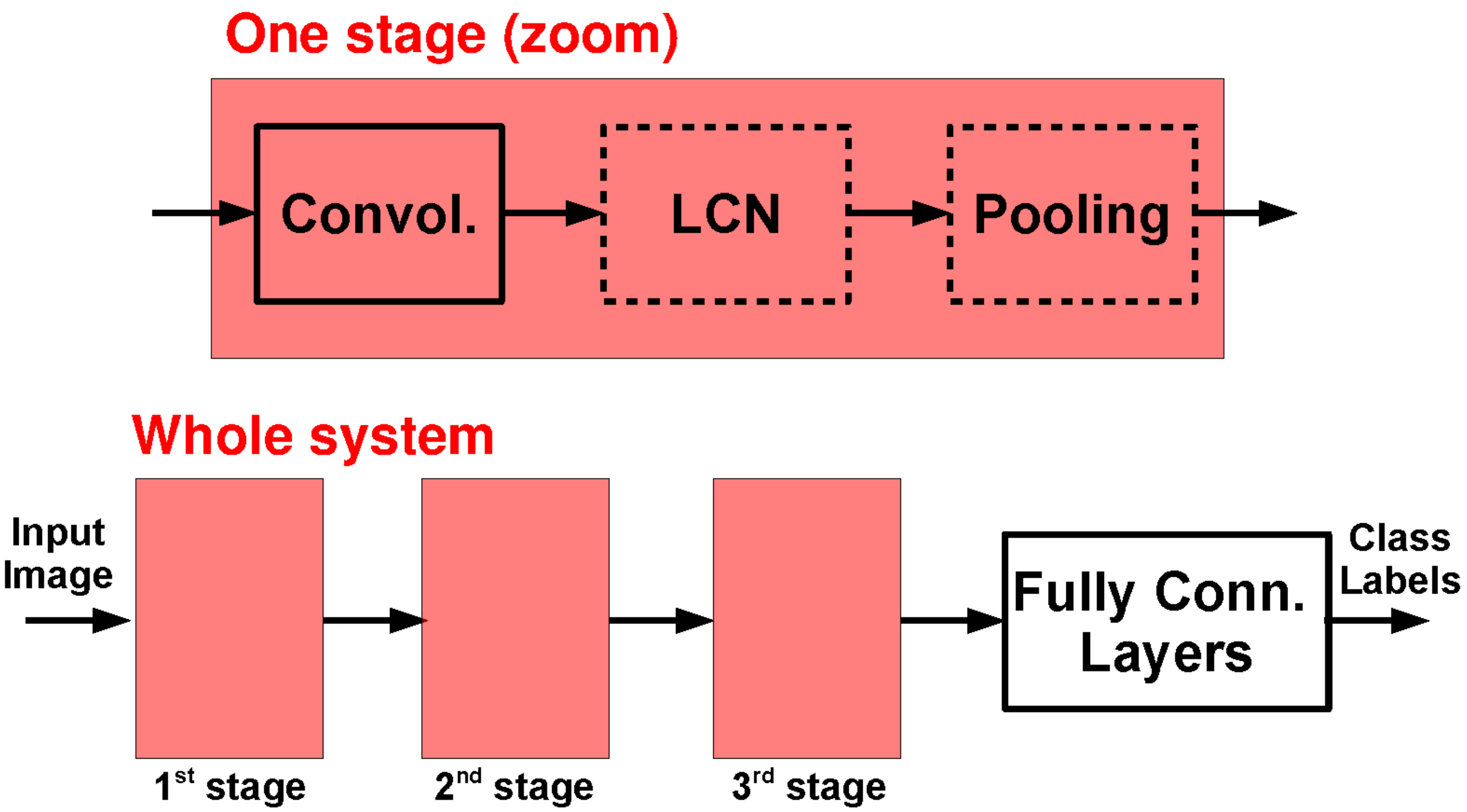
Ranzato

ConvNets: Typical Stage



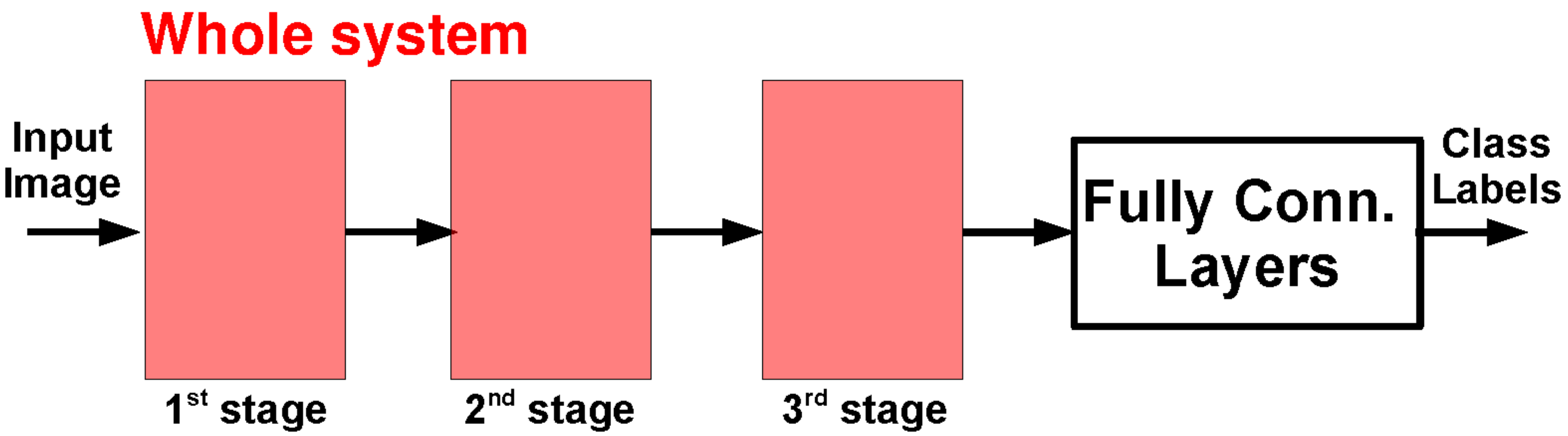
Conceptually similar to: SIFT, HoG, etc.

ConvNets: Typical Architecture



Ranzato

ConvNets: Typical Architecture



Conceptually similar to:

SIFT → K-Means → Pyramid Pooling → SVM

Lazebnik et al. "...Spatial Pyramid Matching..." CVPR 2006

SIFT → Fisher Vect. → Pooling → SVM

Sanchez et al. "Image classifcation with F.V.: Theory and practice" IJCV 2012

ConvNets: Training

All layers are differentiable (a.e.).
We can use standard back-propagation.

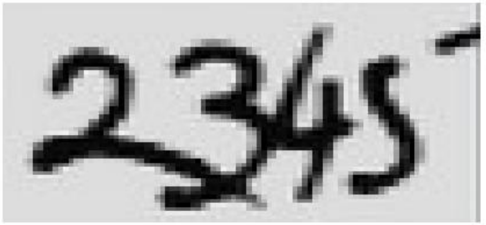
Algorithm:
Given a small mini-batch
- F-PROP
- B-PROP
- PARAMETER UPDATE

Outline

- Supervised Neural Networks
- Convolutional Neural Networks
- Examples
- Tips

CONV NETS: EXAMPLES

- **OCR / House number & Traffic sign classification**



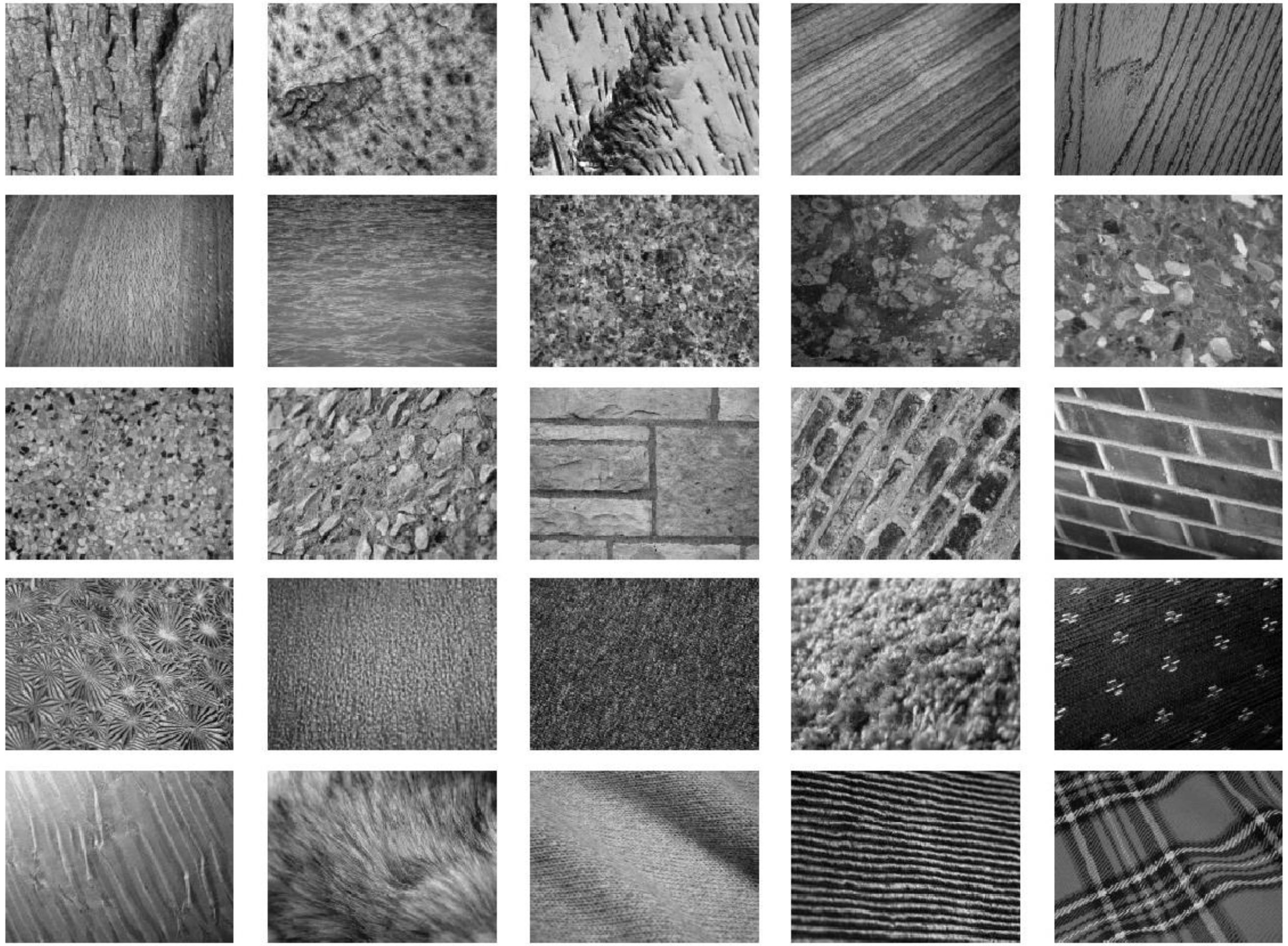
Ciresan et al. “MCDNN for image classification” CVPR 2012

Wan et al. “Regularization of neural networks using dropconnect” ICML 2013

Jaderberg et al. “Synthetic data and ANN for natural scene text recognition” arXiv 2014

CONV NETS: EXAMPLES

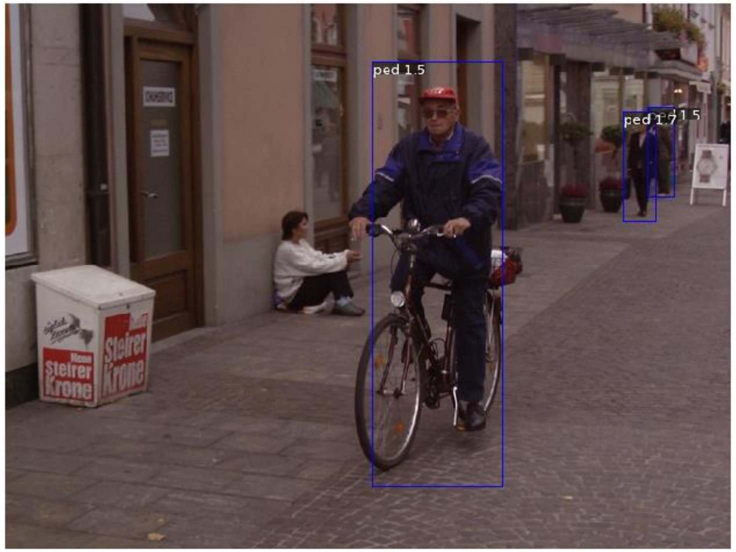
- Texture classification

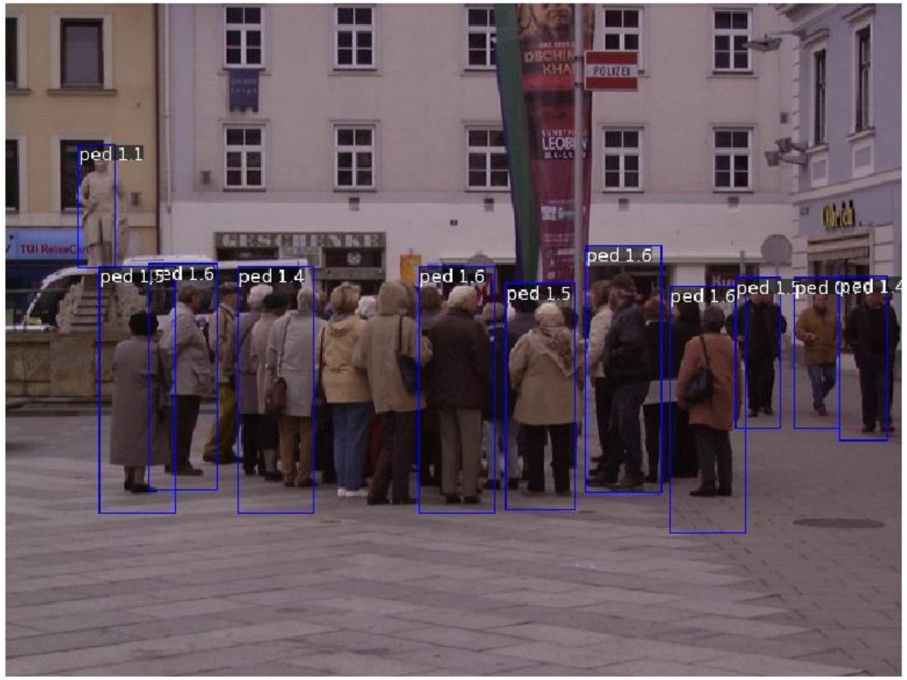


Sifre et al. “Rotation, scaling and deformation invariant scattering...” CVPR 2013

CONV NETS: EXAMPLES

- **Pedestrian detection**

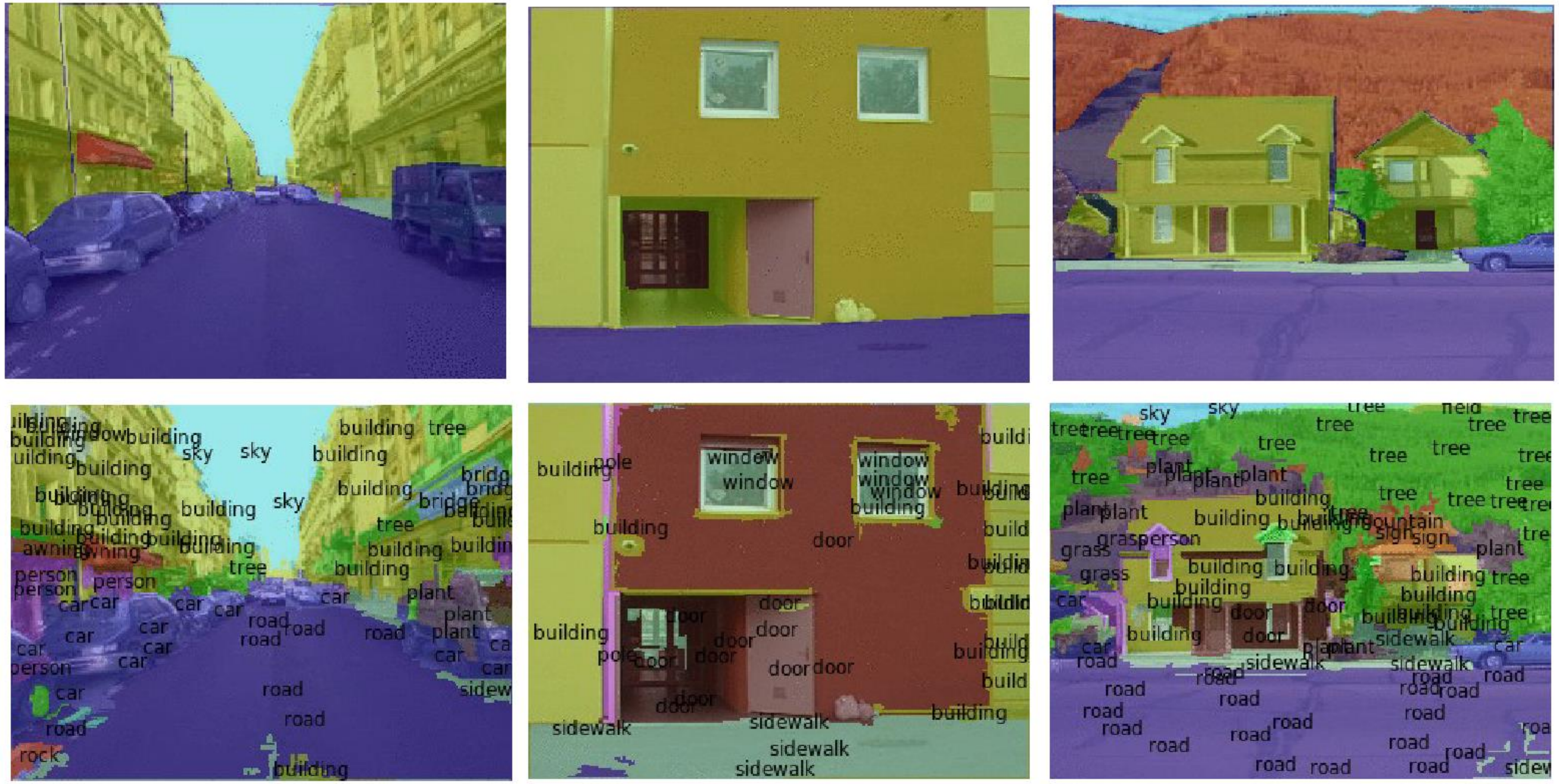




Sermanet et al. “Pedestrian detection with unsupervised multi-stage..” CVPR 2013

CONV NETS: EXAMPLES

- **Scene Parsing**

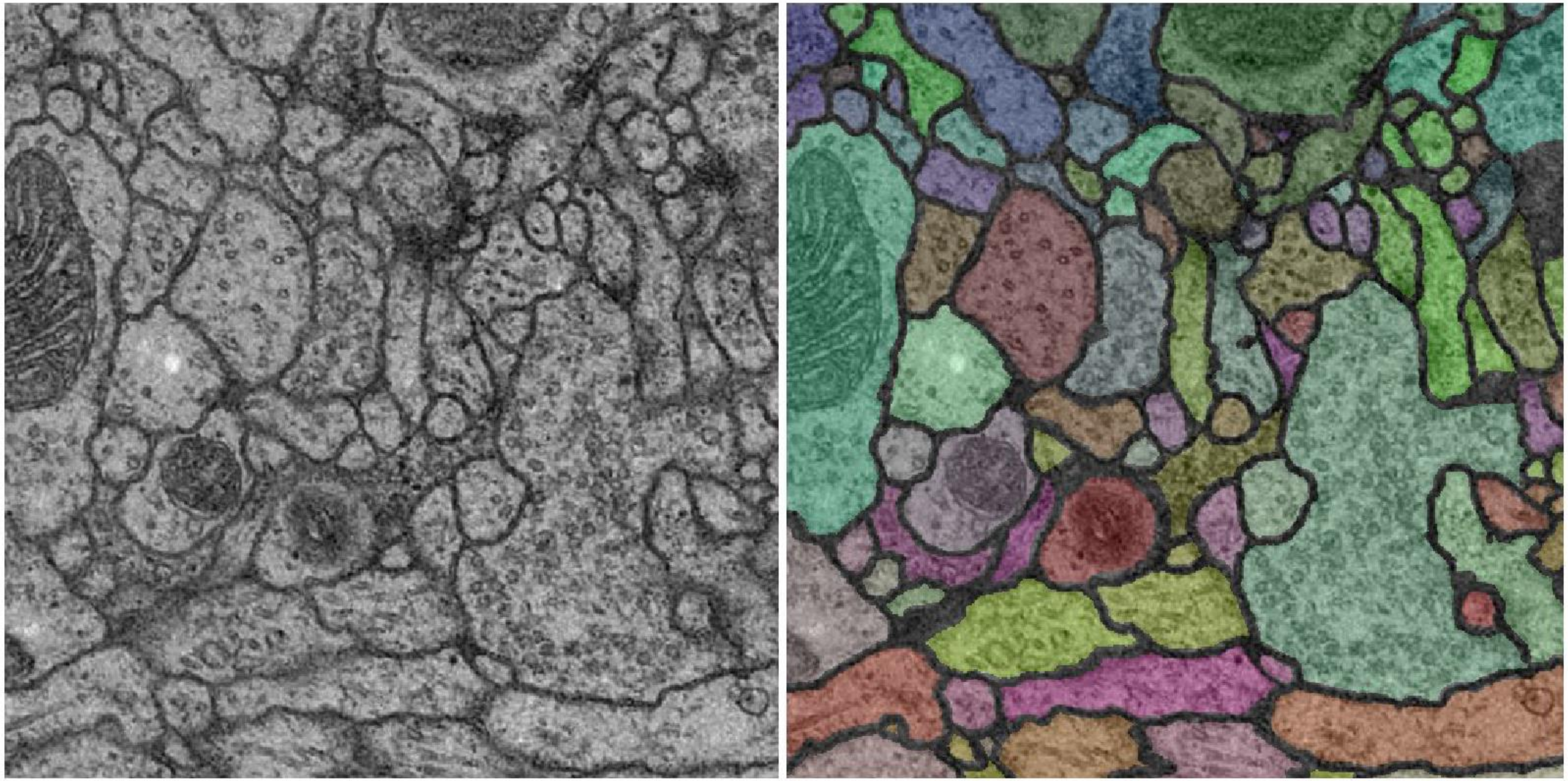


Farabet et al. "Learning hierarchical features for scene labeling" PAMI 2013
Pinheiro et al. "Recurrent CNN for scene parsing" arxiv 2013

Ranzato

CONV NETS: EXAMPLES

- **Segmentation 3D volumetric images**



Ciresan et al. "DNN segment neuronal membranes..." NIPS 2012
Turaga et al. "Maximin learning of image segmentation" NIPS 2009

CONV NETS: EXAMPLES

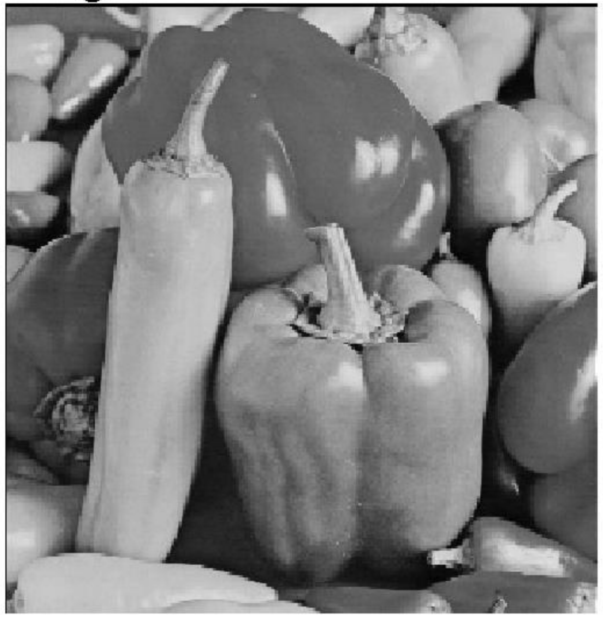
- **Action recognition from videos**

Taylor et al. “Convolutional learning of spatio-temporal features” ECCV 2010
Karpathy et al. “Large-scale video classification with CNNs” CVPR 2014
Simonyan et al. “Two-stream CNNs for action recognition in videos” arXiv 2014

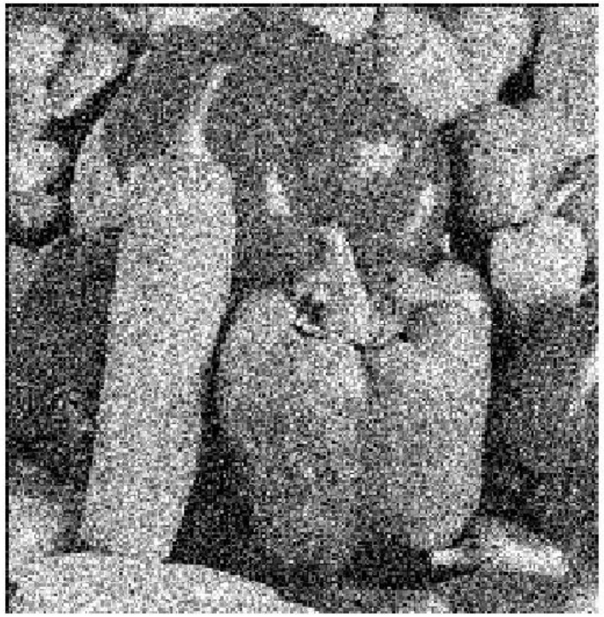
CONV NETS: EXAMPLES

- **Denoising**

original



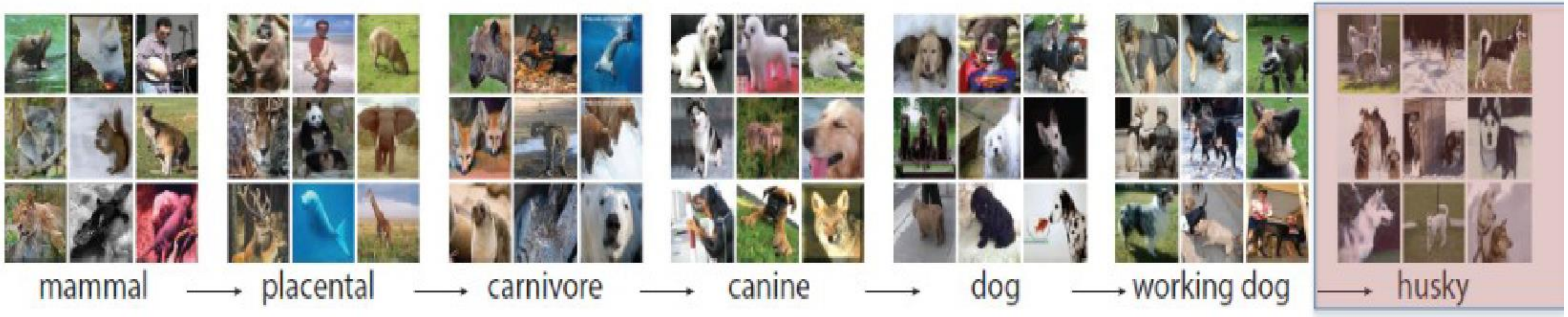
noised



denoised

Burger et al. “Can plain NNs compete with BM3D?” CVPR 2012

Dataset: ImageNet 2012



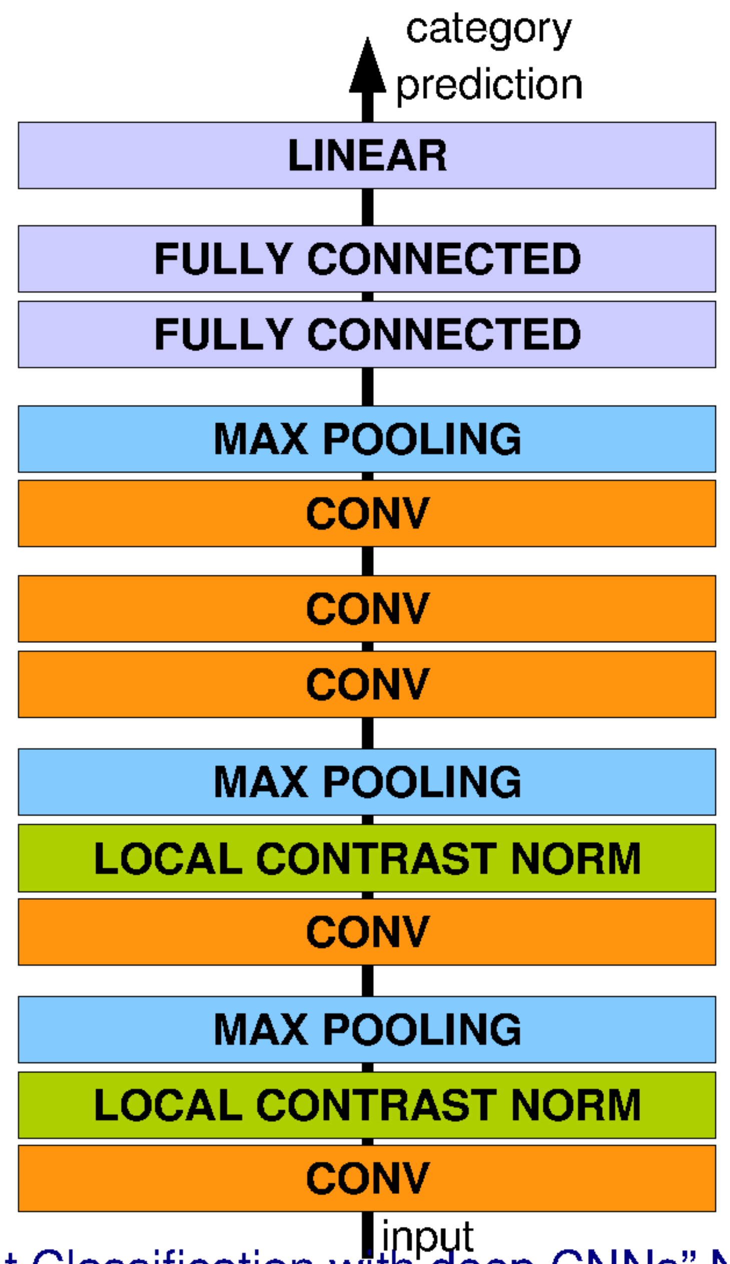
- S: (n) Eskimo dog, **husky** (breed of heavy-coated Arctic sled dog)
  - *direct hypernym* / ***inherited hypernym*** / *sister term*
    - S: (n) working dog (any of several breeds of usually large powerful dogs bred to work as draft animals and guard and guide dogs)
      - S: (n) dog, domestic dog, Canis familiaris (a member of the genus Canis (probably descended from the common wolf) that has been domesticated by man since prehistoric times; occurs in many breeds) *"the dog barked all night"*
        - S: (n) canine, canid (any of various fissiped mammals with nonretractile claws and typically long muzzles)
          - S: (n) carnivore (a terrestrial or aquatic flesh-eating mammal) *"terrestrial carnivores have four or five clawed digits on each limb"*
            - S: (n) placental, placental mammal, eutherian, eutherian mammal (mammals having a placenta; all mammals except monotremes and marsupials)
              - S: (n) mammal, mammalian (any warm-blooded vertebrate having the skin more or less covered with hair; young are born alive except for the small subclass of monotremes and nourished with milk)
                - S: (n) vertebrate, craniate (animals having a bony or cartilaginous skeleton with a segmented spinal column and a large brain enclosed in a skull or cranium)
                  - S: (n) chordate (any animal of the phylum Chordata having a notochord or spinal column)
                    - S: (n) animal, animate being, beast, brute, creature, fauna (a living organism characterized by voluntary movement)
                      - S: (n) organism, being (a living thing that has (or can develop) the ability to act or function independently)
                        - S: (n) living thing, animate thing (a living (or once living) entity)
                          - S: (n) whole, unit (an assemblage of parts that is regarded as a single entity) *"how big is that part compared to the whole?"; "the team is a unit"*
                            - S: (n) object, physical object (a tangible and visible entity; an entity that can cast a shadow) *"it was full of rackets, balls and other objects"*
                              - S: (n) physical entity (an entity that has physical existence)
                                - S: (n) entity (that which is perceived or known or inferred to have its own distinct existence (living or nonliving))

Deng et al. "Imagenet: a large scale hierarchical image database" CVPR 2009

ImageNet

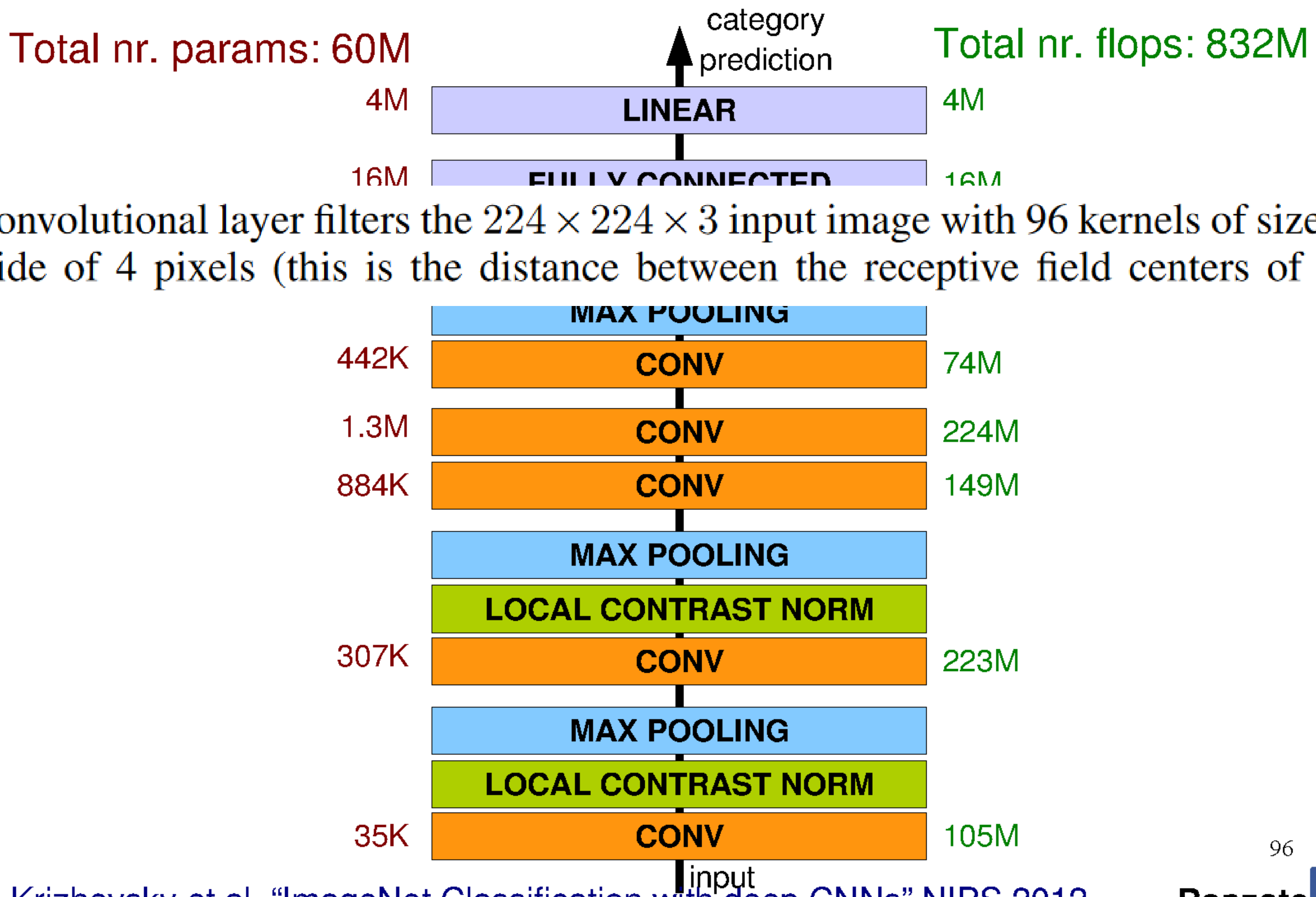
Examples of hammer:

Architecture for Classification



Krizhevsky et al. “ImageNet Classification with deep CNNs” NIPS 2012

Architecture for Classification



The first convolutional layer filters the $224 \times 224 \times 3$ input image with 96 kernels of size $11 \times 11 \times 3$ with a stride of 4 pixels (this is the distance between the receptive field centers of neighboring

Krizhevsky et al. "ImageNet Classification with deep CNNs" NIPS 2012

Ranzato

Optimization

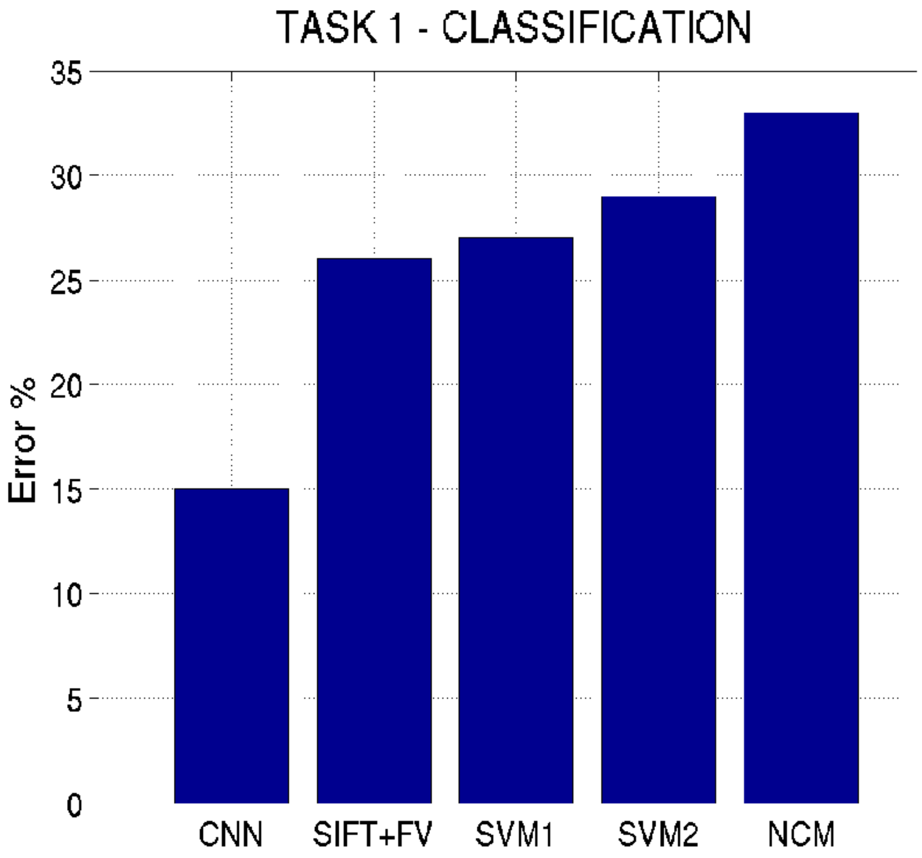
SGD with momentum:

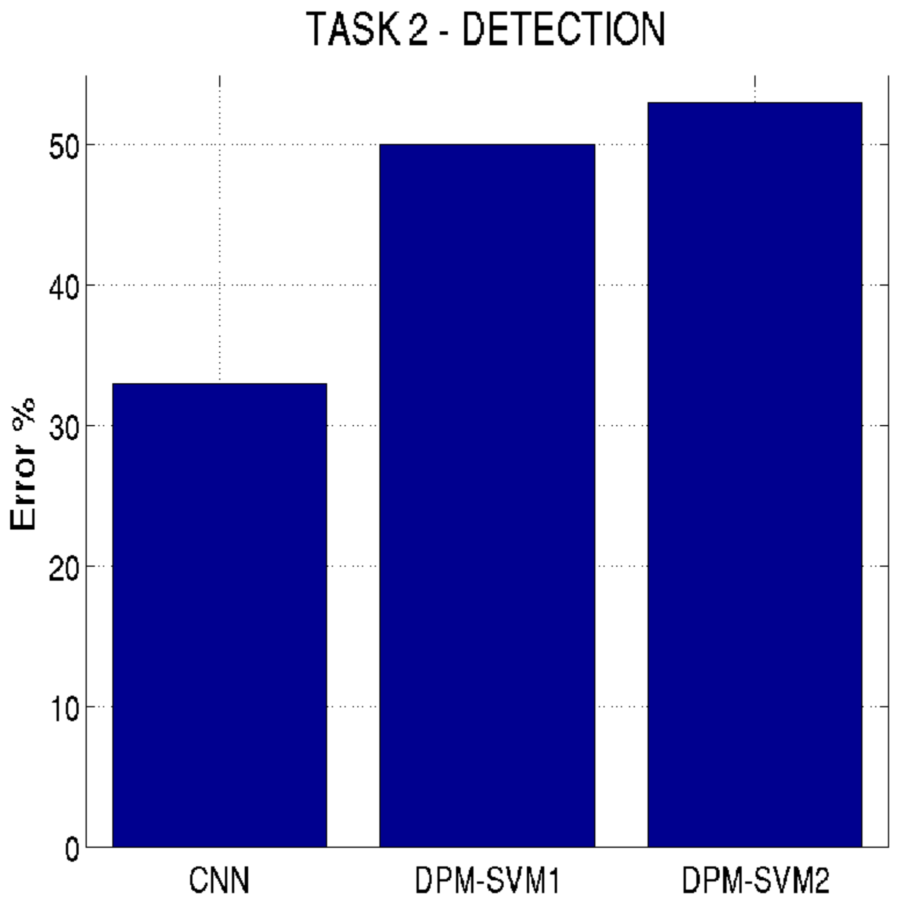
- Learning rate = 0.01
- Momentum = 0.9

Improving generalization by:

- Weight sharing (convolution)
- Input distortions
- Dropout = 0.5
- Weight decay = 0.0005

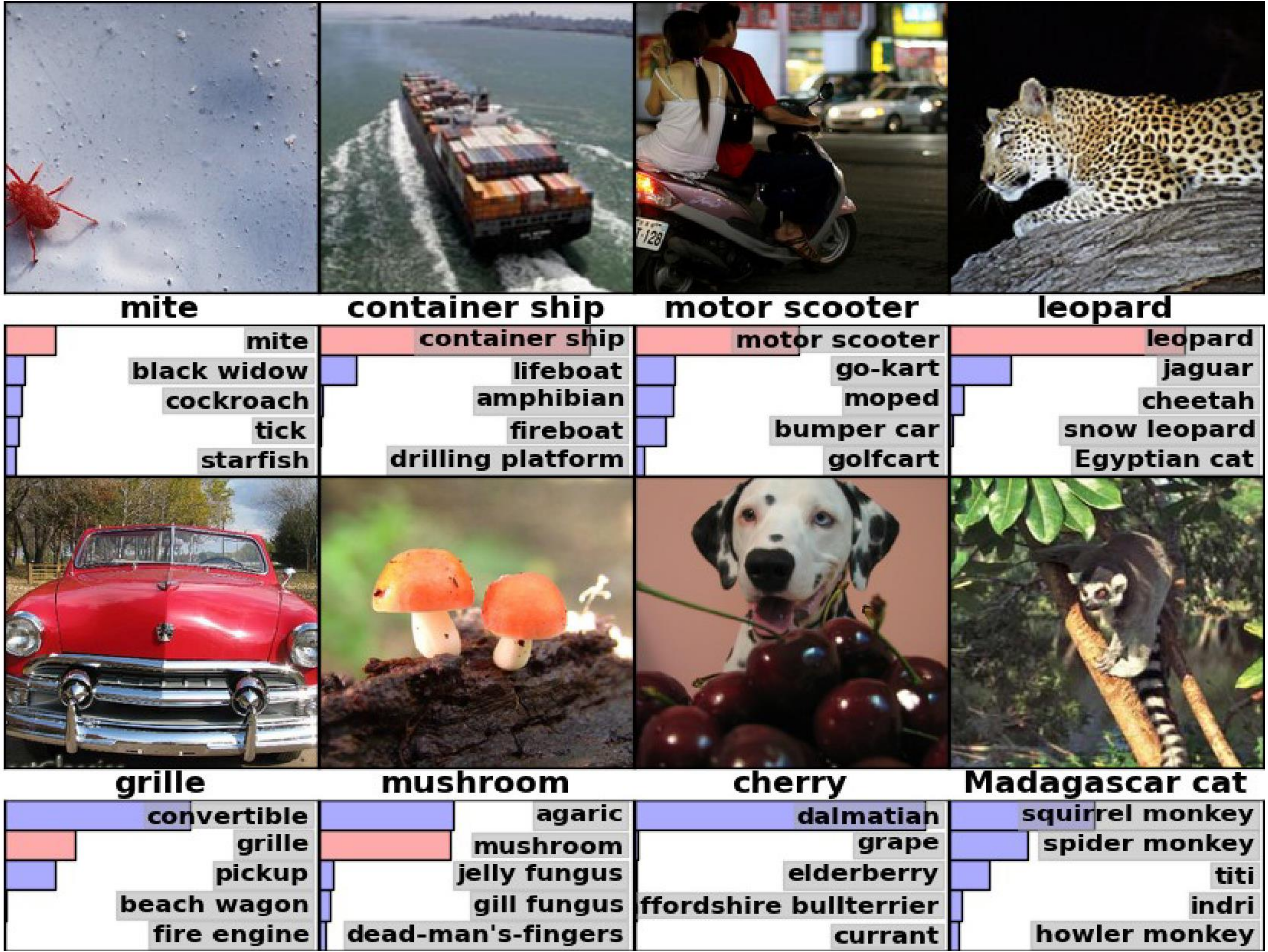
Results: ILSVRC 2012





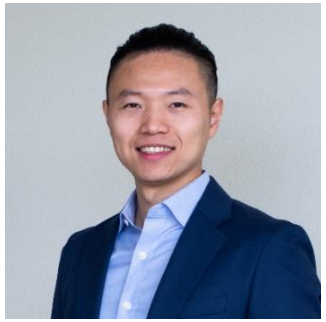
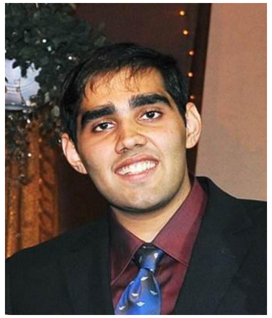
Krizhevsky et al. “ImageNet Classification with deep CNNs” NIPS 2012

Ranzato


mite
mite
black widow
cockroach
tick
starfish
container ship
container ship
lifeboat
amphibian
fireboat
drilling platform
motor scooter
motor scooter
go-kart
moped
bumper car
golfcart
leopard
leopard
jaguar
cheetah
snow leopard
Egyptian cat
grille
convertible
grille
pickup
beach wagon
fire engine
mushroom
agaric
mushroom
jelly fungus
gill fungus
dead-man's-fingers
cherry
dalmatian
grape
elderberry
ffordshire bullterrier
currant
Madagascar cat
squirrel monkey
spider monkey
titi
indri
howler monkey

Object Detectors Emerge in Deep Scene CNNs

Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, Antonio Torralba


Massachusetts Institute of Technology

CNN for Object Recognition

Large-scale image classification result on ImageNet

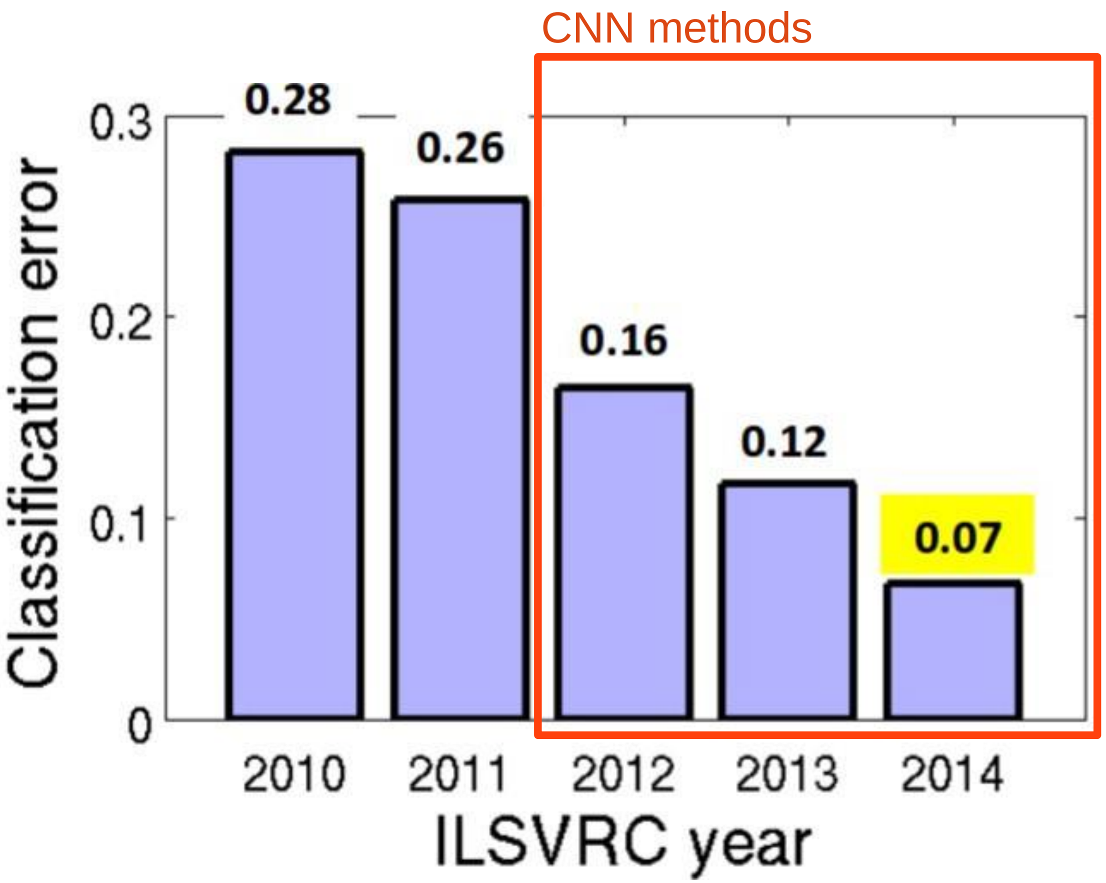
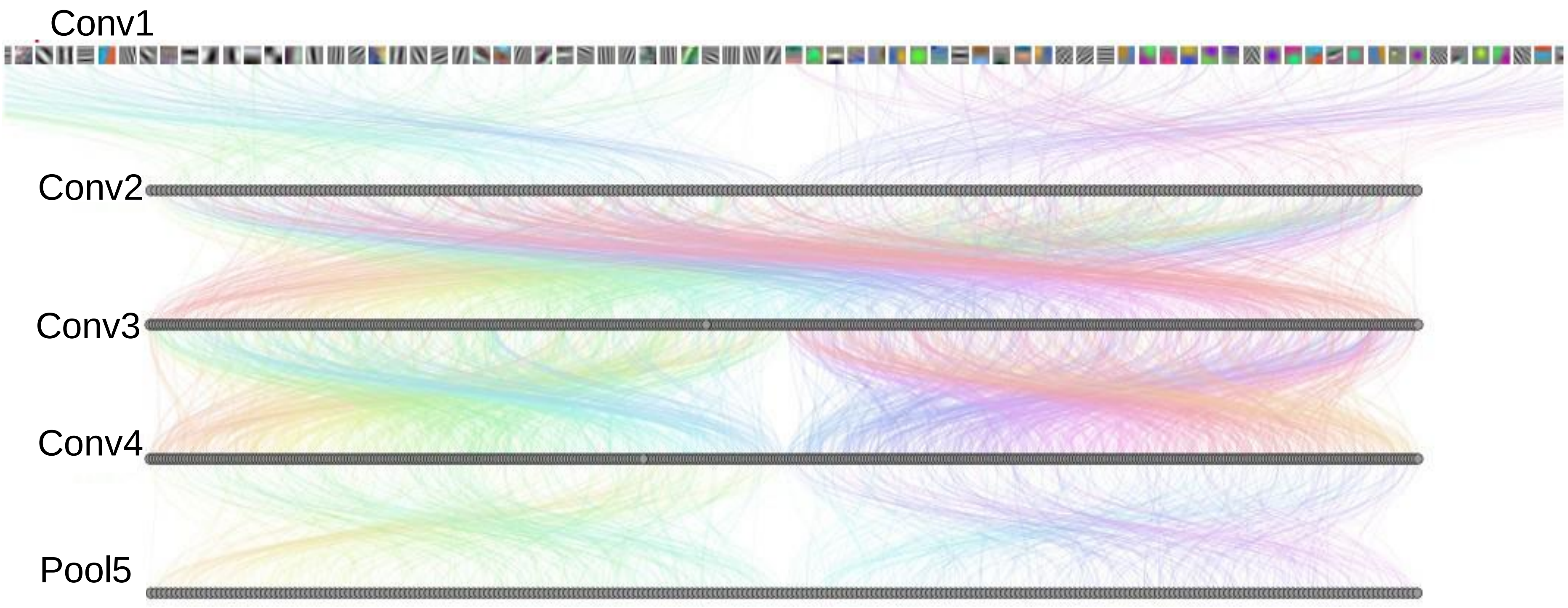


Figure from Olga Russakovsky ECCV'14 workshop

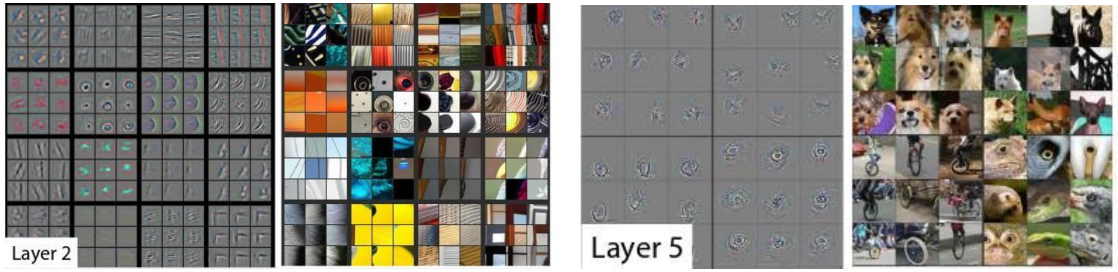
How Objects are Represented in CNN?



DrawCNN: visualizing the units' connections

How Objects are Represented in CNN?

Deconvolution

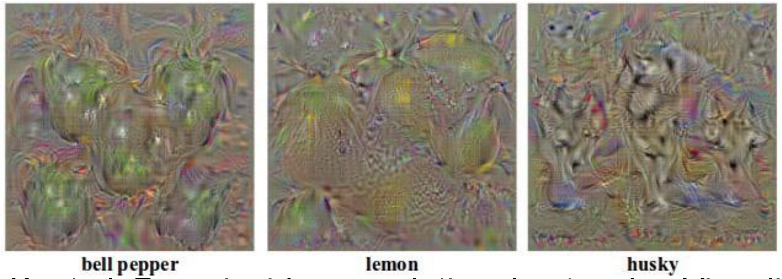


Zeiler, M. et al. Visualizing and Understanding Convolutional Networks,ECCV 2014.

Strong activation image

Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accu-rate object detection and semantic segmentation. CVPR 2014

Back-propagation



Simonyan, K. et al. Deep inside convolutional networks: Visualising image classification models and saliency maps. ICLR workshop, 2014

Object Representations in Computer Vision

Part-based models are used to represent objects and visual patterns.

-Object as a set of parts

-Relative locations between parts

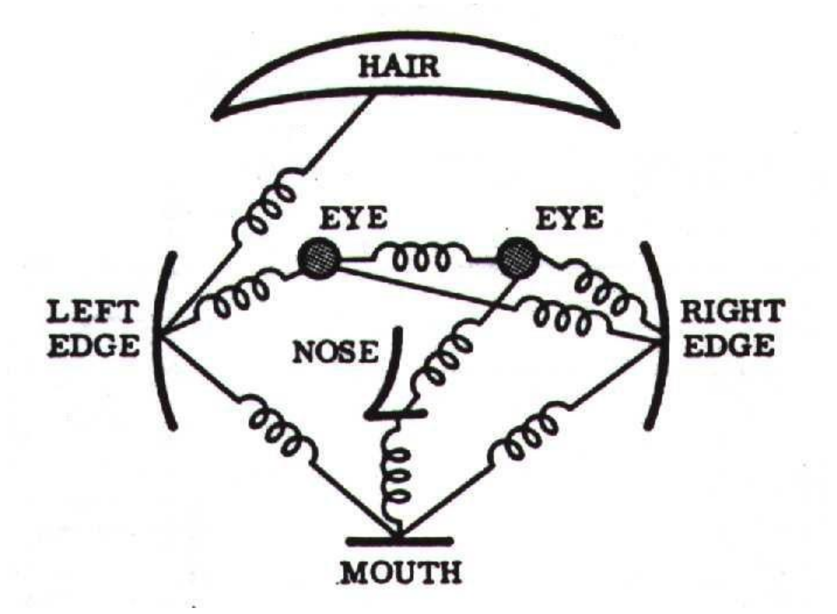


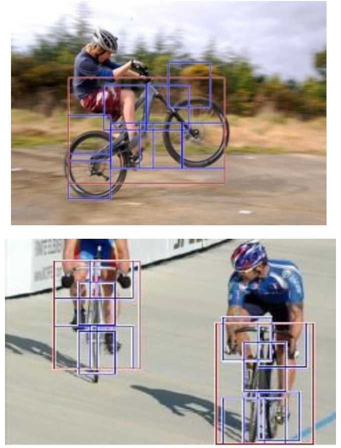
Figure from Fischler & Elschlager (1973)

Object Representations in Computer Vision

Constellation model

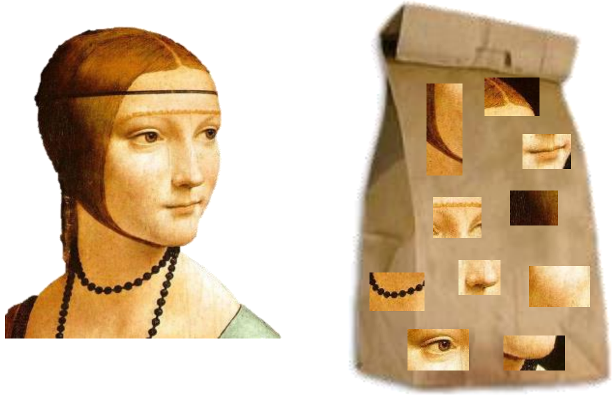
Weber, Welling & Perona (2000),
Fergus, Perona & Zisserman (2003)

Deformable Part model



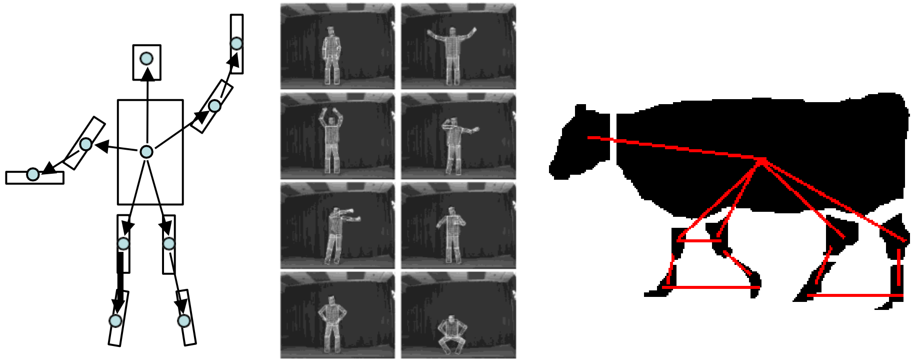
P. Felzenszwalb, R. Girshick, D. McAllester, D. Ramanan (2010)

Bag-of-word model



Lazebnik, Schmid & Ponce(2003), Fei-Fei Perona (2005)

Class-specific graph model



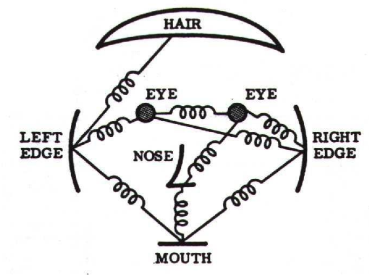
Kumar, Torr and Zisserman (2005), Felzenszwalb & Huttenlocher (2005)

Learning to Recognize Objects

IMAGENET

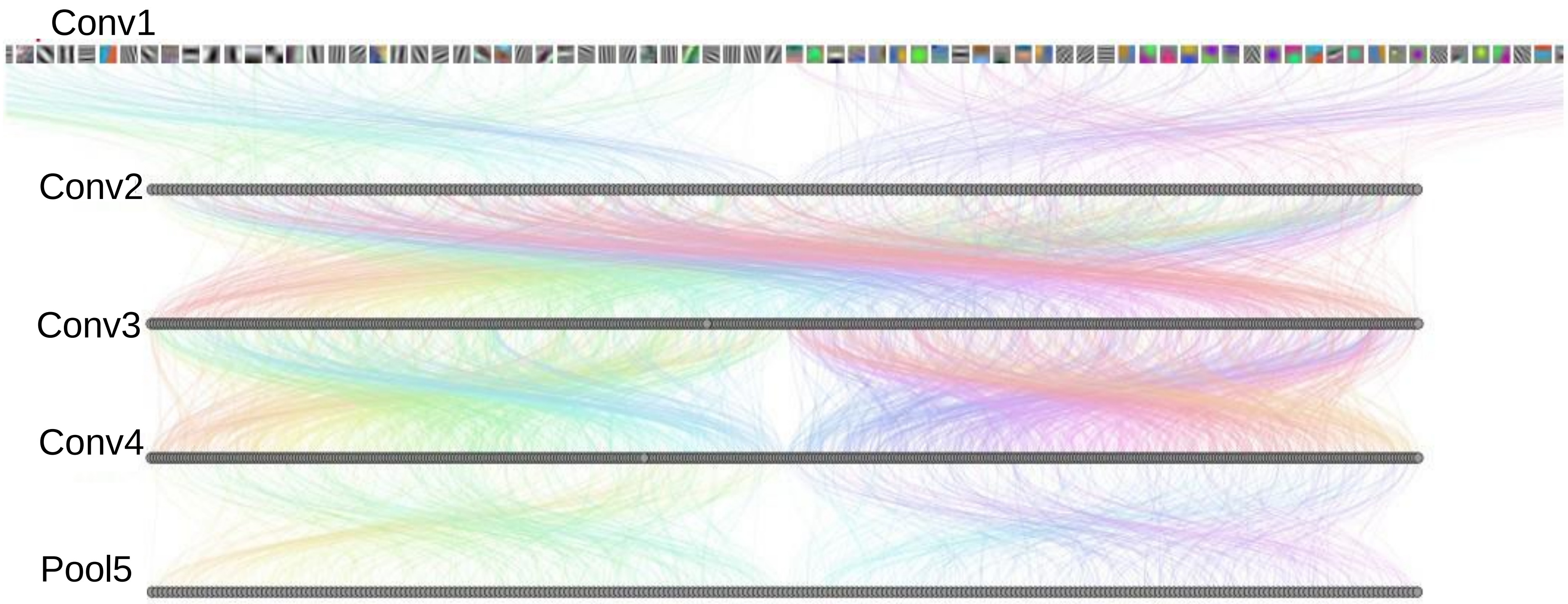
Possible internal representations:

- Object parts
- Textures
- Attributes



How Objects are Represented in CNN?

CNN uses **distributed code** to represent objects.



Agrawal, et al. Analyzing the performance of multilayer neural networks for object recognition. ECCV, 2014
Szegedy, et al. Intriguing properties of neural networks.arXiv preprint arXiv:1312.6199, 2013.
Zeiler, M. et al. Visualizing and Understanding Convolutional Networks, ECCV 2014.

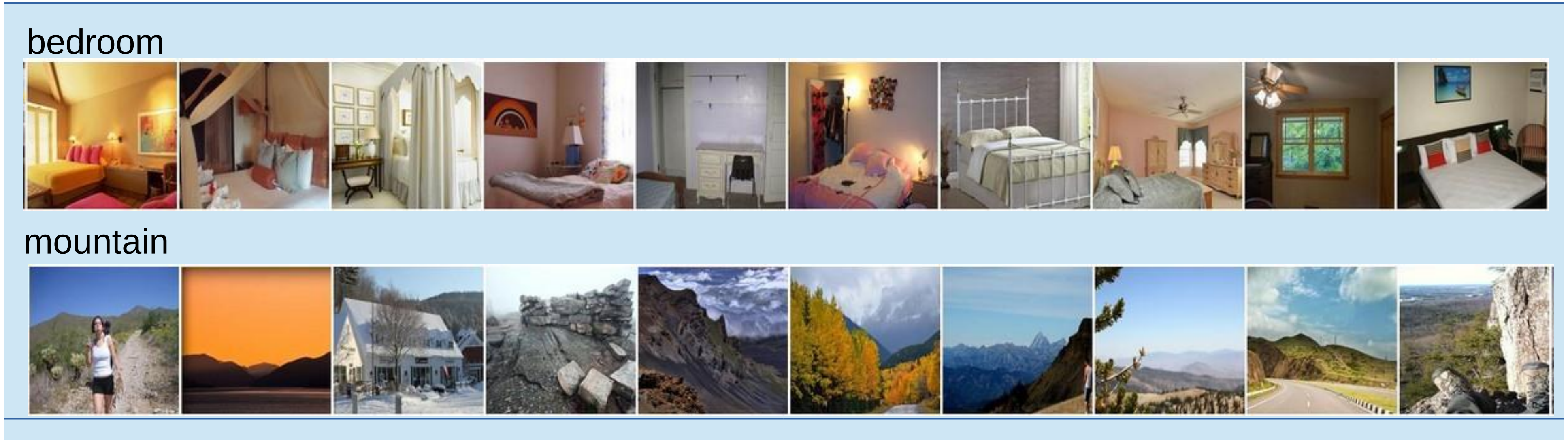
Scene Recognition

Given an image, predict which place we are in.

Bedroom

Harbor

Learning to Recognize Scenes

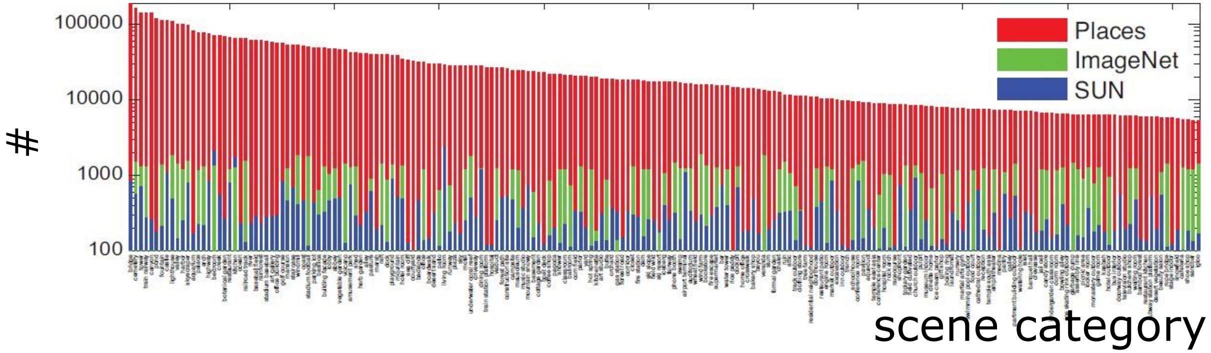


Possible internal representations:

- Objects (scene parts?)
- Scene attributes
- Object parts
- Textures

CNN for Scene Recognition

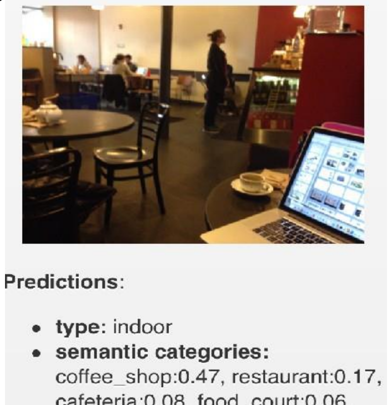
Places Database: 7 million images from 400 scene categories

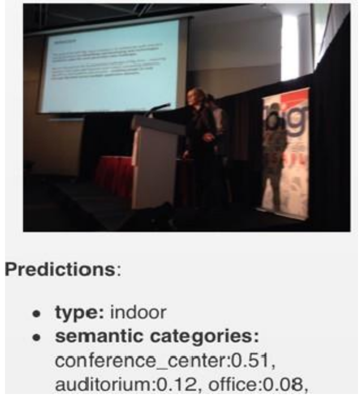


Places-CNN: AlexNet CNN on 2.5 million images from 205 scene categories.

	Places 205	SUN 205
Places-CNN	**50.0%**	**66.2%**
ImageNet CNN feature+SVM	40.8%	49.6%

Scene Recognition Demo: 78% top-5 recognition accuracy in the wild





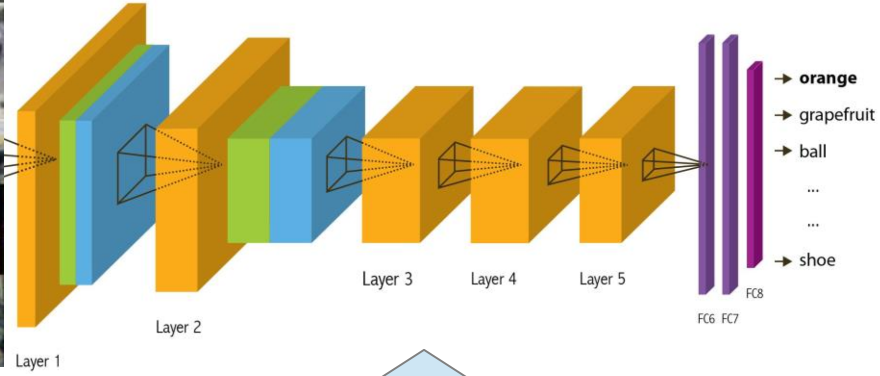
http://places.csail.mit.edu

Zhou, et al. NIPS, 2014.

ImageNet CNN and Places CNN

IMAGENET

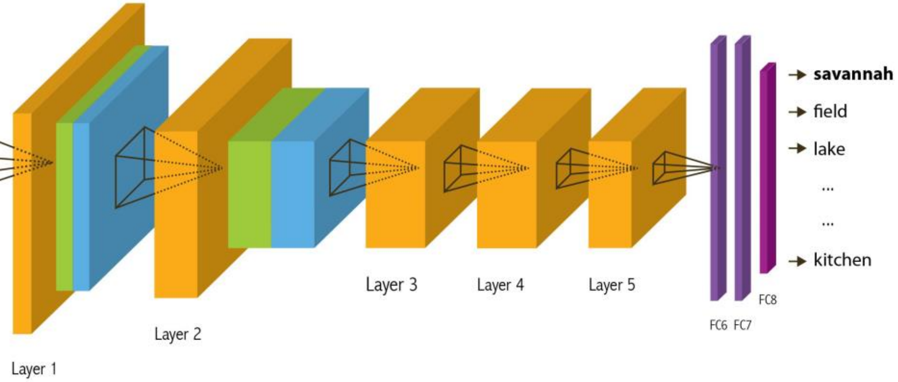
ImageNet CNN for Object Classification



Same architecture: AlexNet

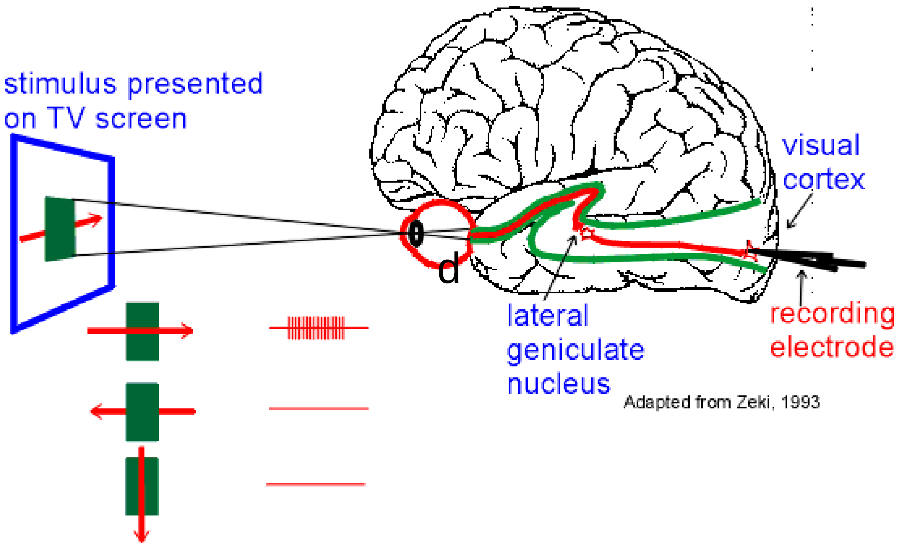
Places

Places CNN for Scene Classification

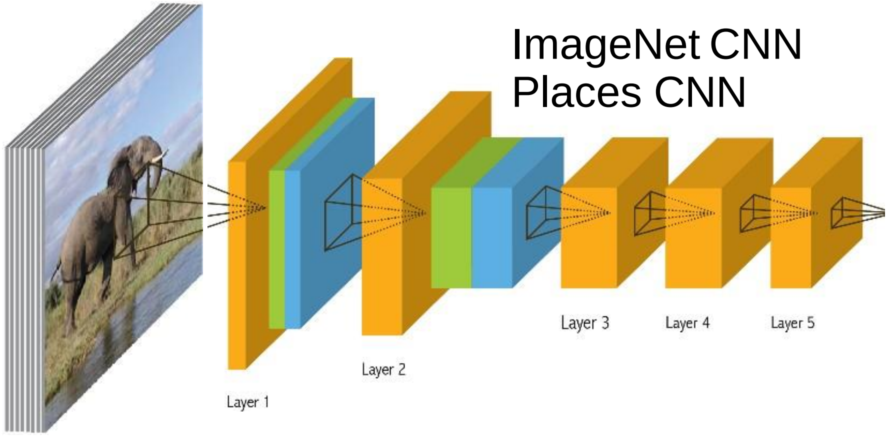


Data-Driven Approach to Study CNN

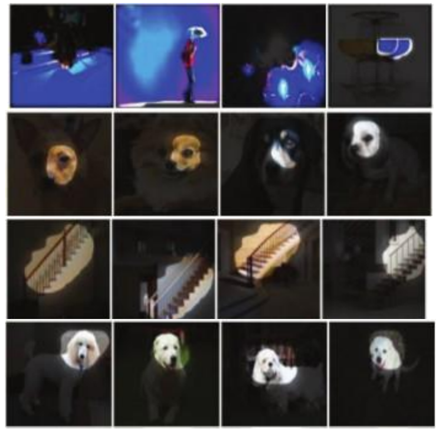
Neuroscientists study brain



200,000 image stimuli of objects and scene categories (ImageNet TestSet+SUN database)



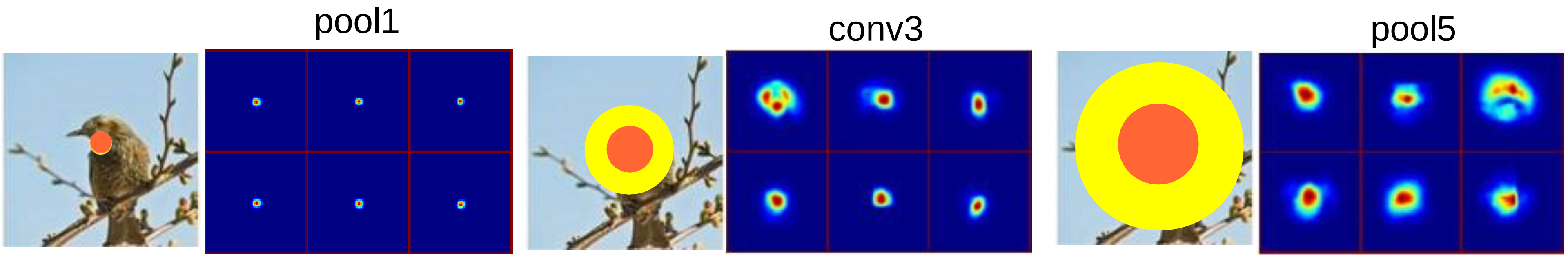
ImageNet CNN
Places CNN



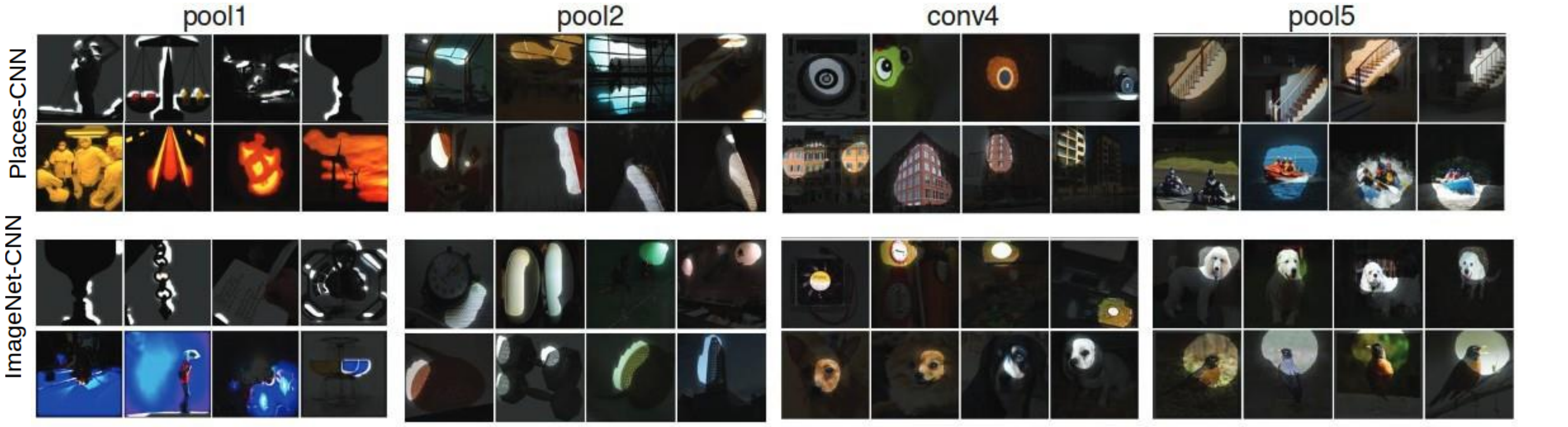
Estimating the Receptive Fields

Estimated receptive fields

Actual size of RF is much smaller than the theoretic size



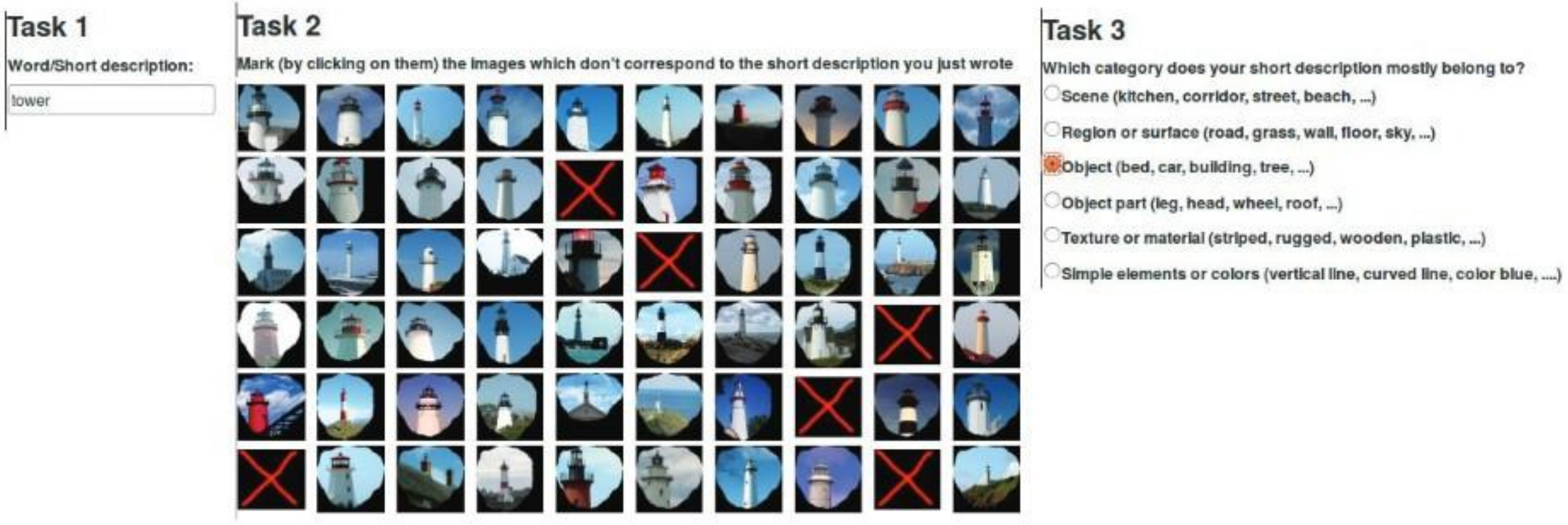
Segmentation using the RF of Units



More semantically meaningful

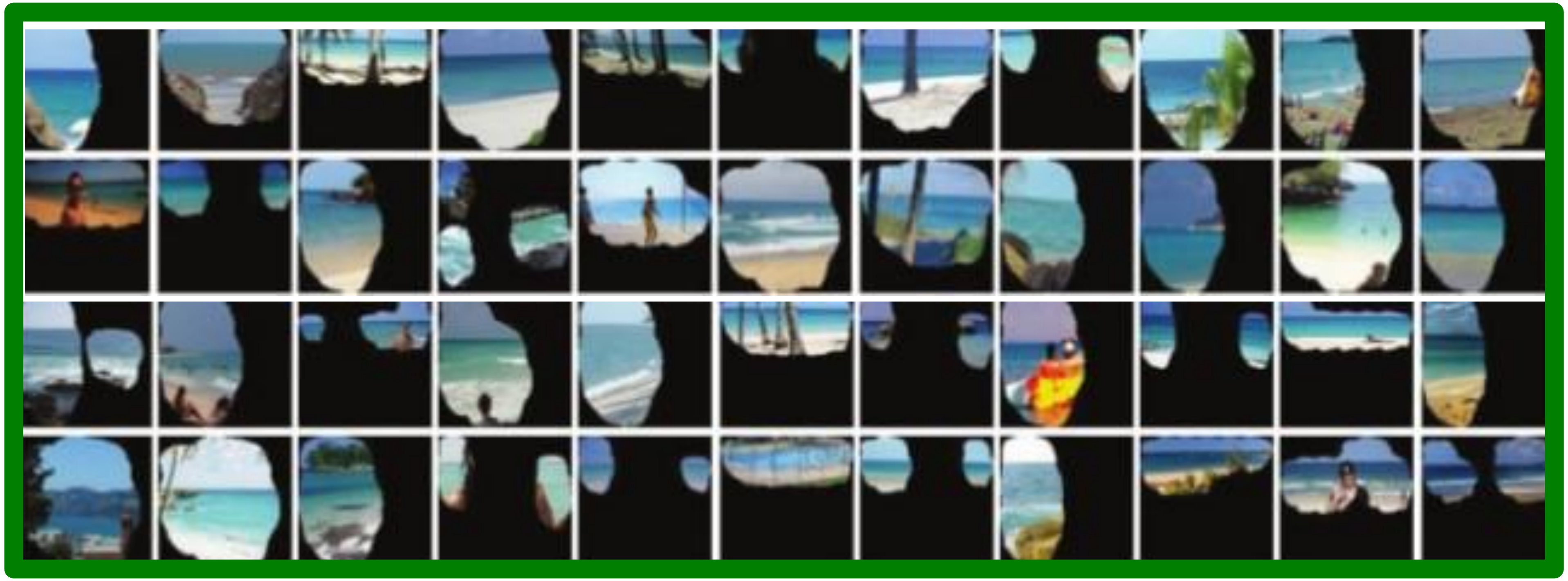
Annotating the Semantics of Units

Top ranked segmented images are cropped and sent to Amazon Turk for annotation.



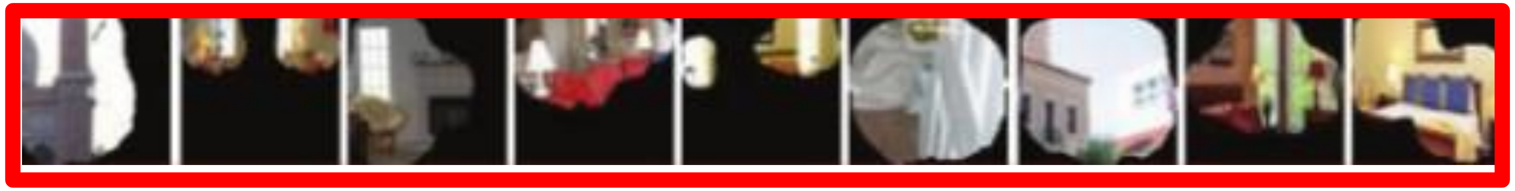
Annotating the Semantics of Units

Pool5, unit 76; Label: ocean; Type: scene; Precision: 93%



Annotating the Semantics of Units

Pool5, unit 13; Label: Lamps; Type: object; Precision: 84%

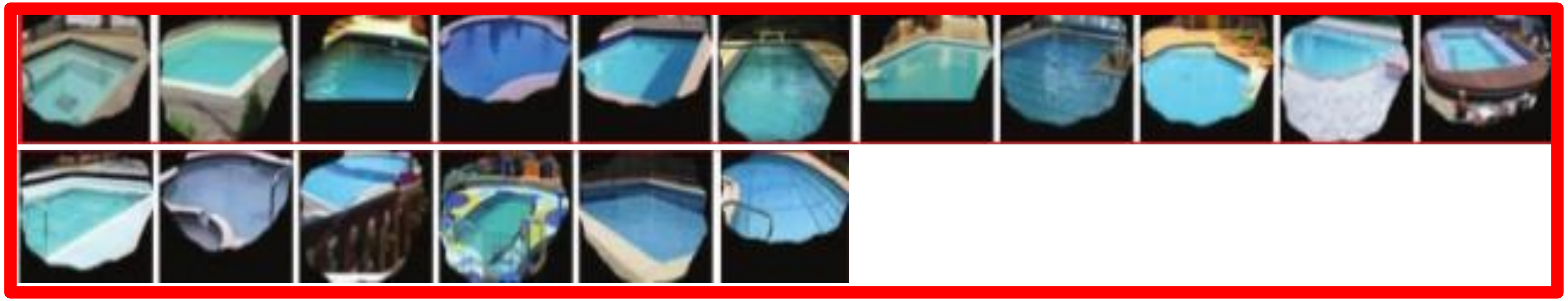


Annotating the Semantics of Units

Pool5, unit 77; Label:legs; Type: object part; Precision: 96%

Annotating the Semantics of Units

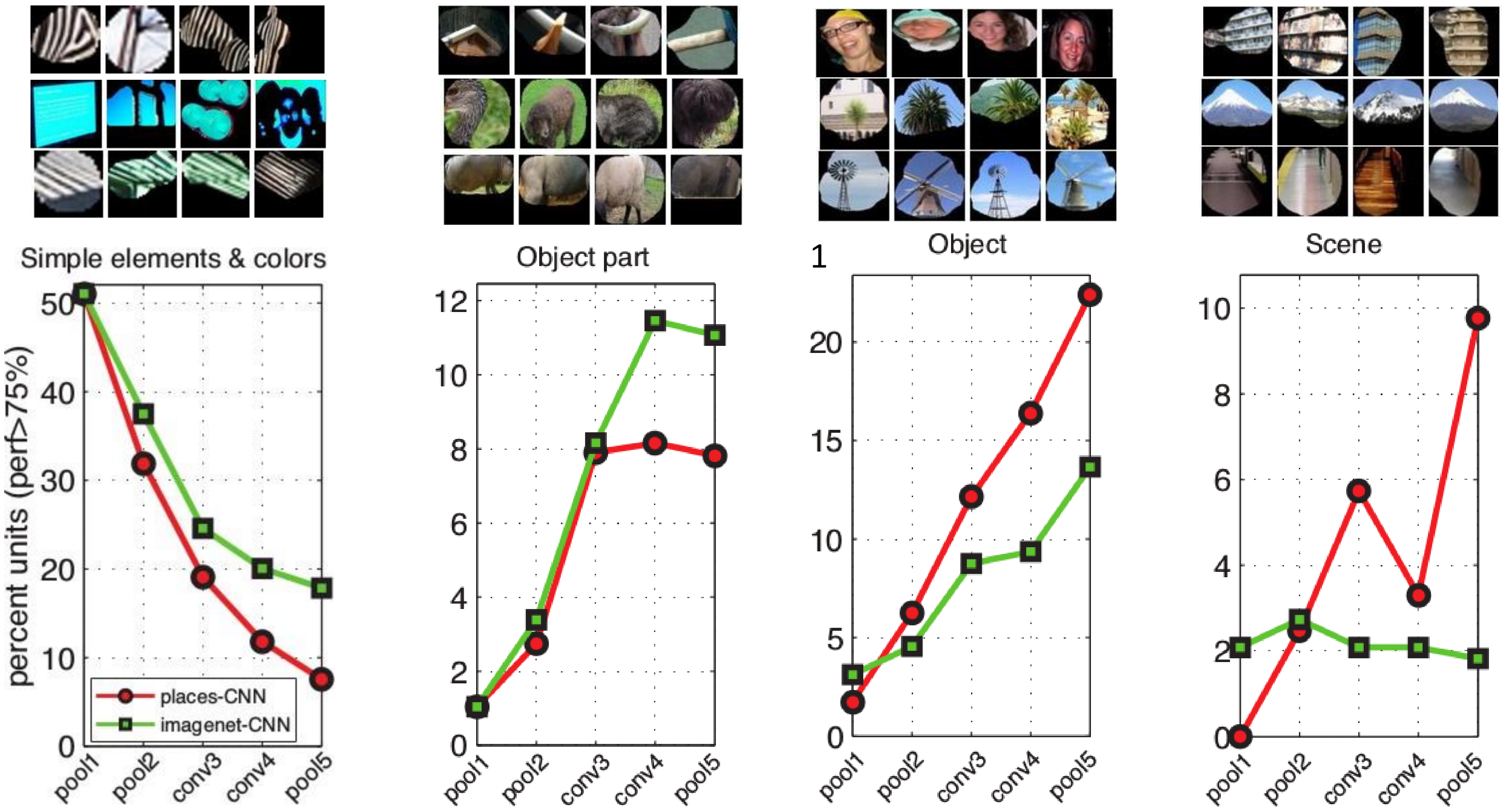
Pool5, unit 112; Label: pool table; Type: object; Precision: 70%



Annotating the Semantics of Units

Pool5, unit 22; Label: dinner table; Type: scene; Precision: 60%

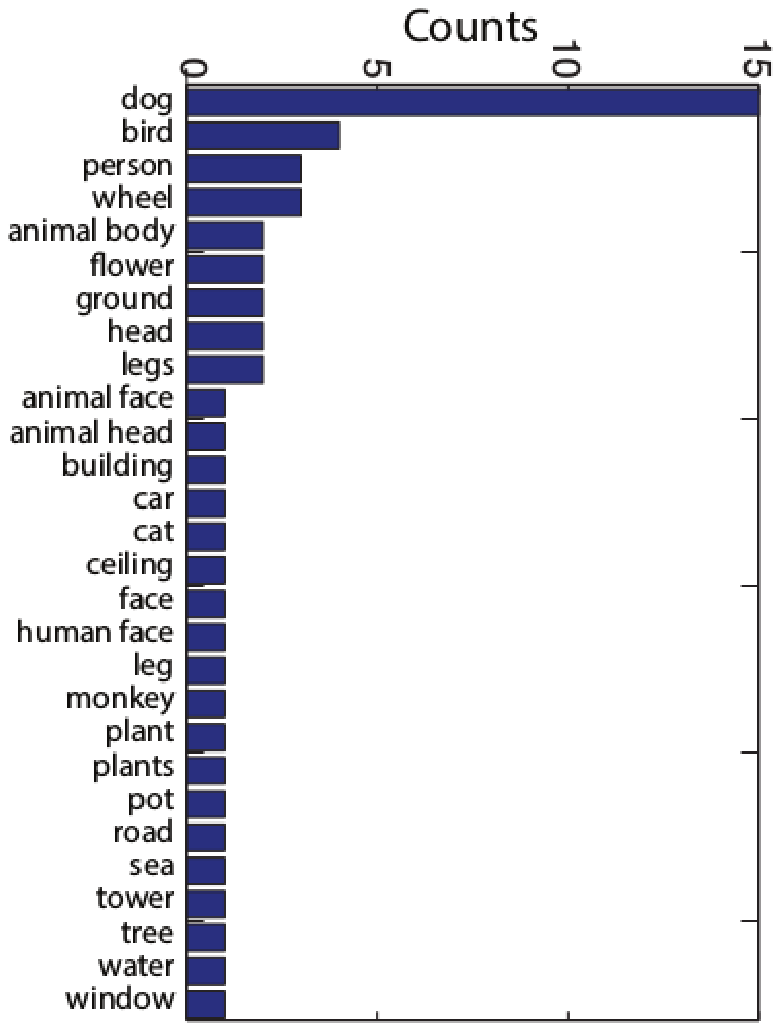
Distribution of Semantic Types at Each Layer



Object detectors emerge within CNN trained to classify scenes, without any object supervision!

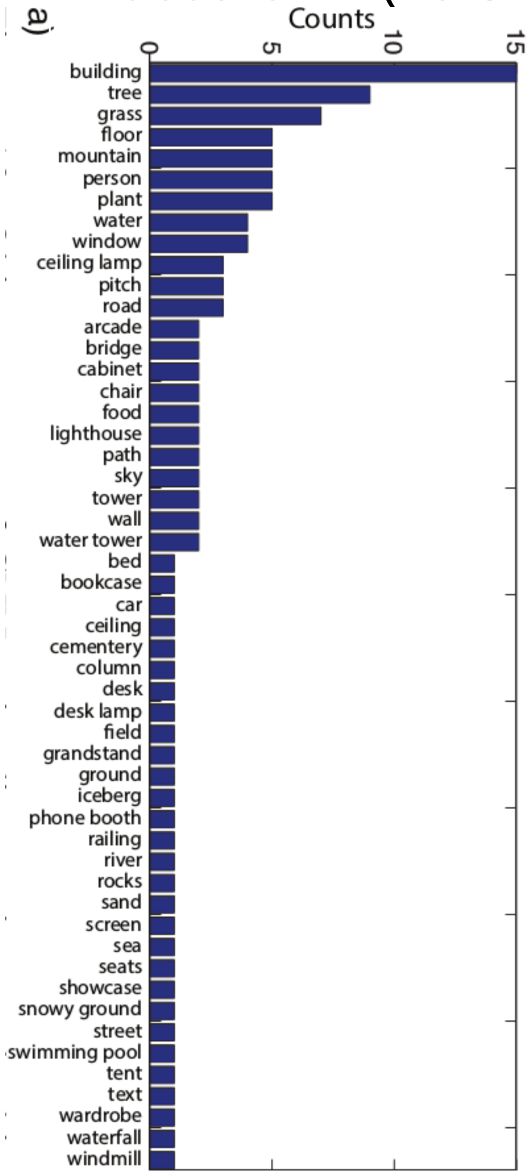
Histogram of Emerged Objects in Pool5

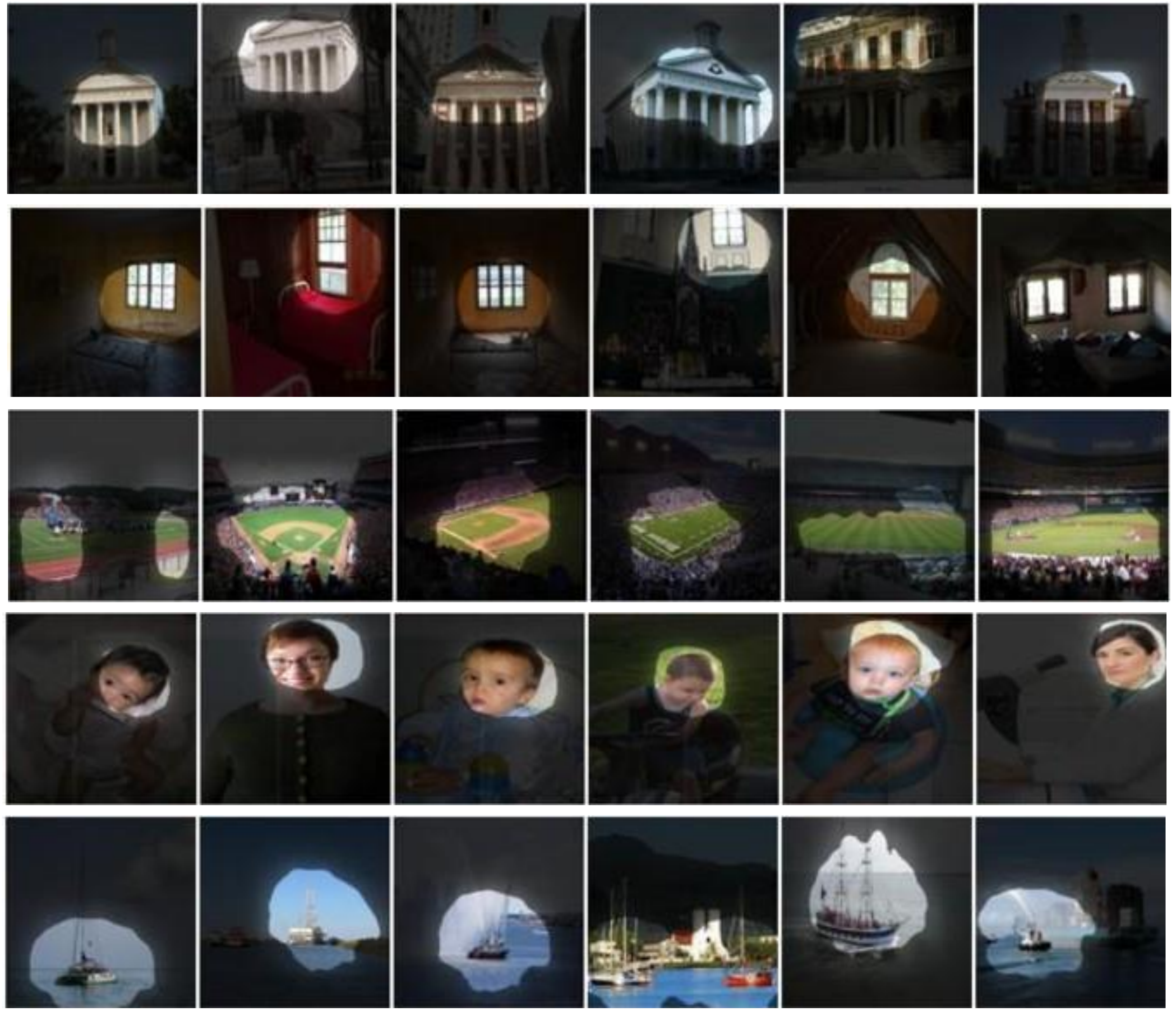
ImageNet-CNN (59/256)



Histogram of Emerged Objects in Pool5

Places-CNN (151/256)

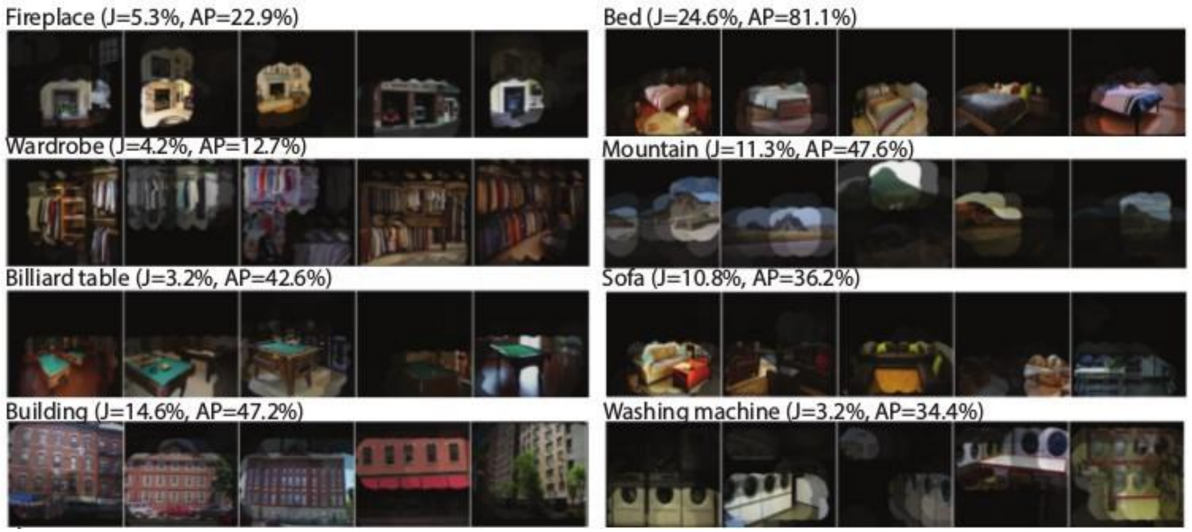


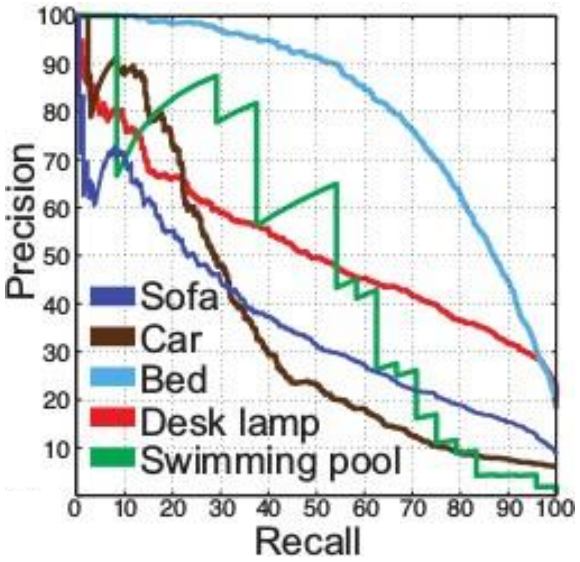



Buildings
56) building
120) arcade
8) bridge
123) building
119) building
9) lighthouse
Furniture
18) billard table
155) bookcase
116) bed
38) cabinet
85) chair
People
3) person
49) person
138) person
100) person
Lighting
55) ceiling lamp
174) ceiling lamp
223) ceiling lamp
13) desk lamp
Nature
195) grass
89) iceberg
140) mountain
159) sand

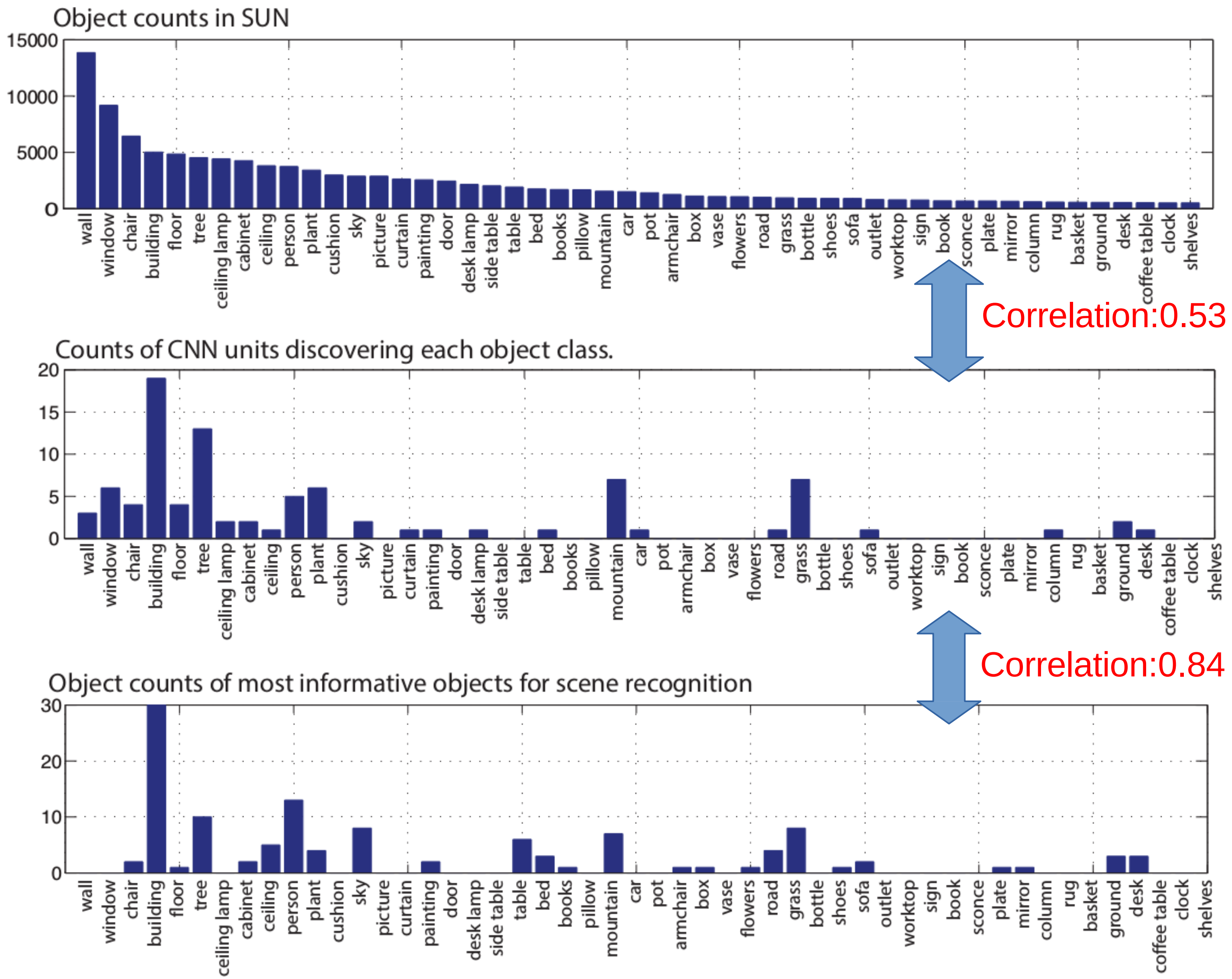
Evaluation on SUN Database

Evaluate the performance of the emerged object detectors





Evaluation on SUN Database



Conclusion

We show that object detectors emerge inside a CNN trained to classify scenes, without any object supervision.

Object detectors for free!

Places database, Places CNN, and unit annotations could be downloaded at

http://places.csail.mit.edu