Deep Learning 2
Neural Net Basics

Computer Vision

James Hays

Many slides by Marc'Aurelio Ranzato

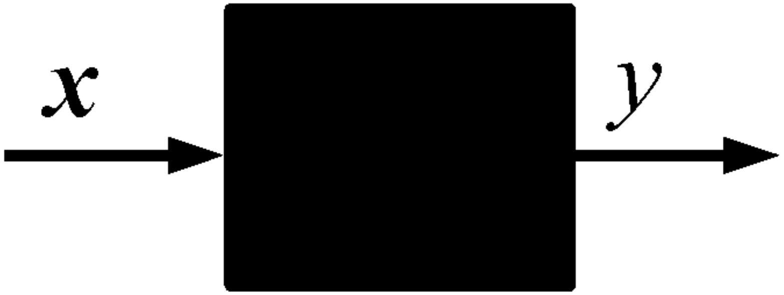
Supervised Learning

$\{(\boldsymbol{x}^i, y^i), i=1 \ldots P\}$ training dataset

$\boldsymbol{x}^i$ i-th input training example

y^i i-th target label

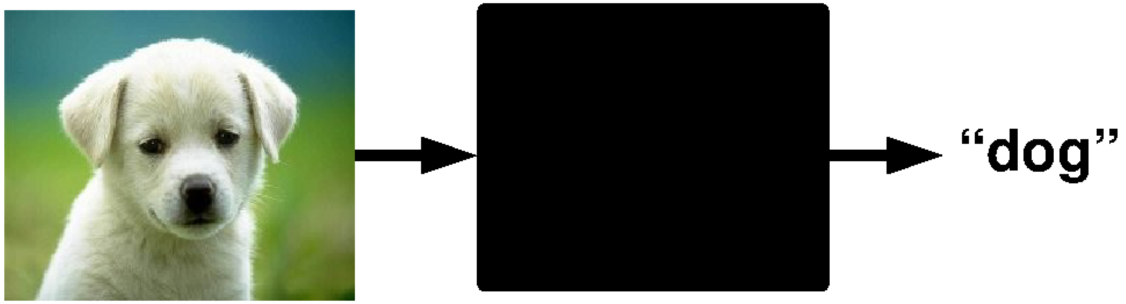
P number of training examples



Goal: predict the target label of unseen inputs.

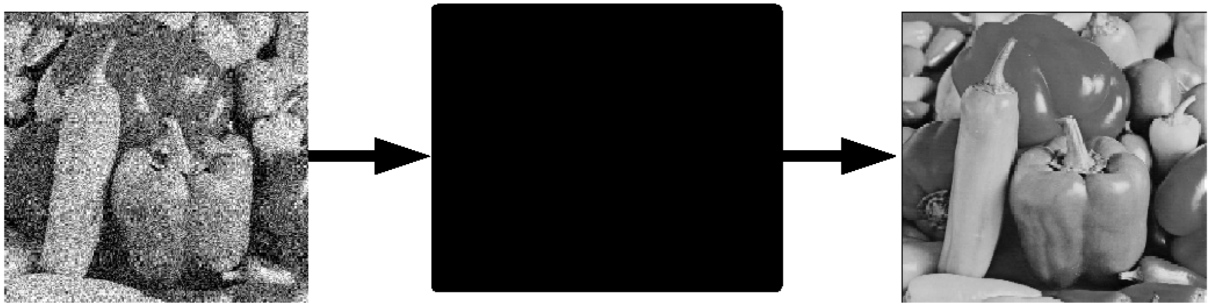
Supervised Learning: Examples

Classification



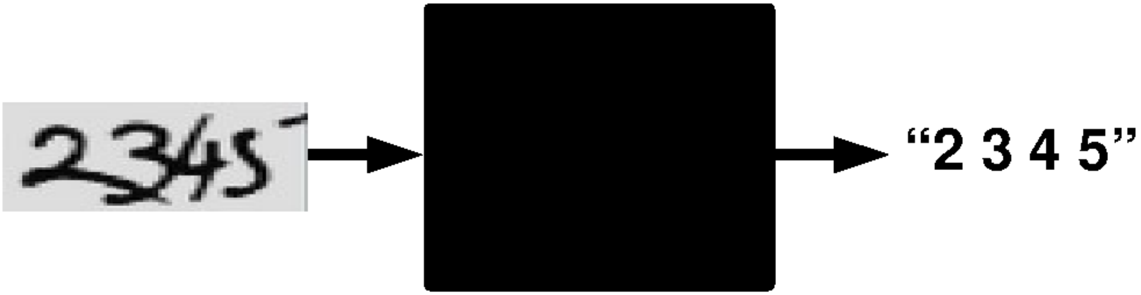
classification

Denoising



regression

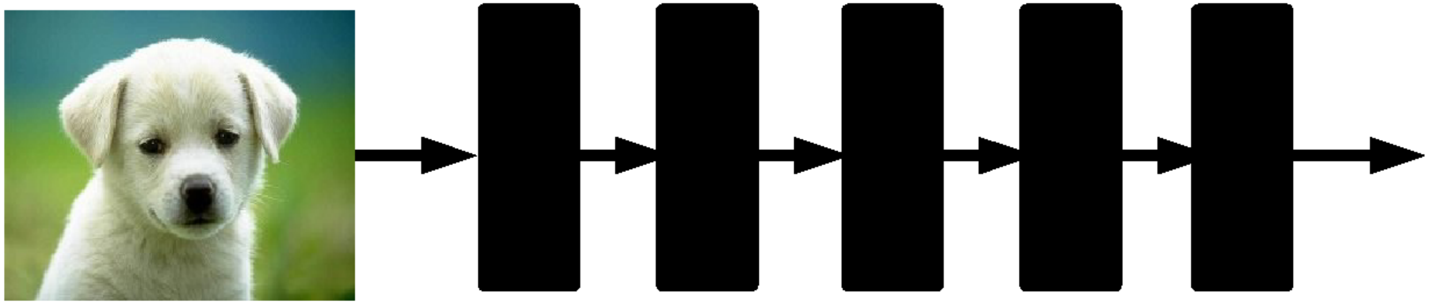
OCR



structured prediction

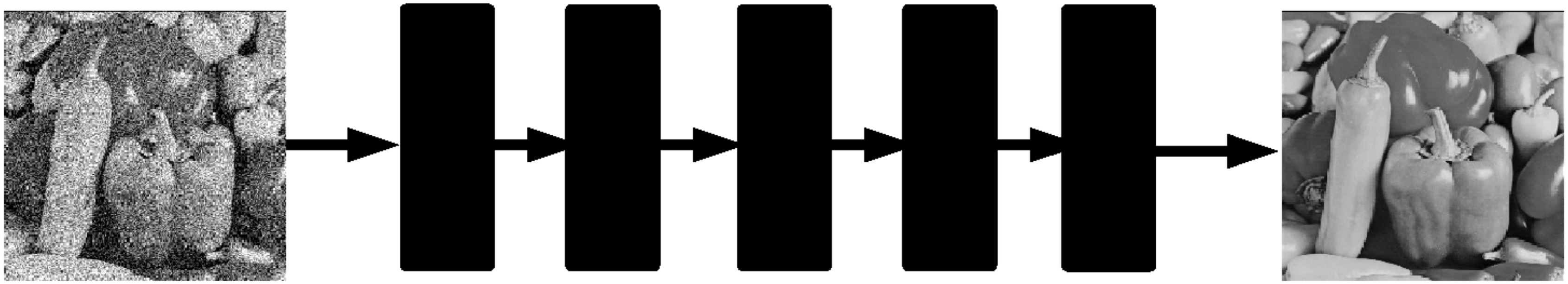
Supervised Deep Learning

Classification

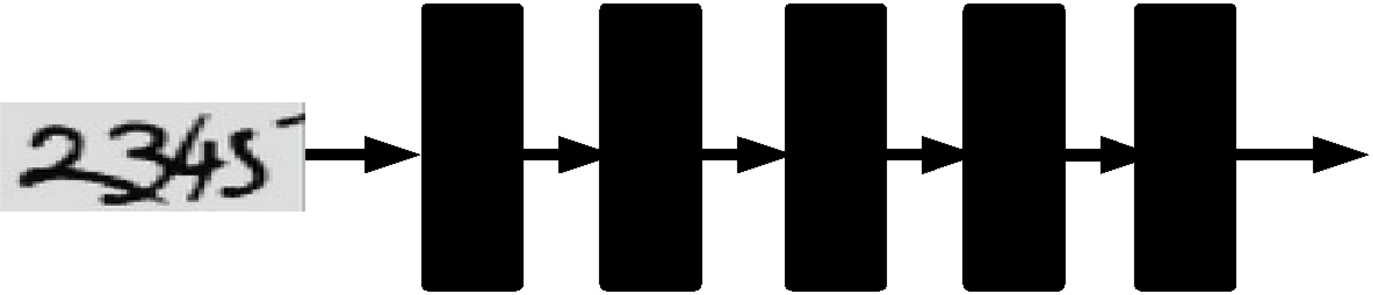


“dog”

Denoising



OCR

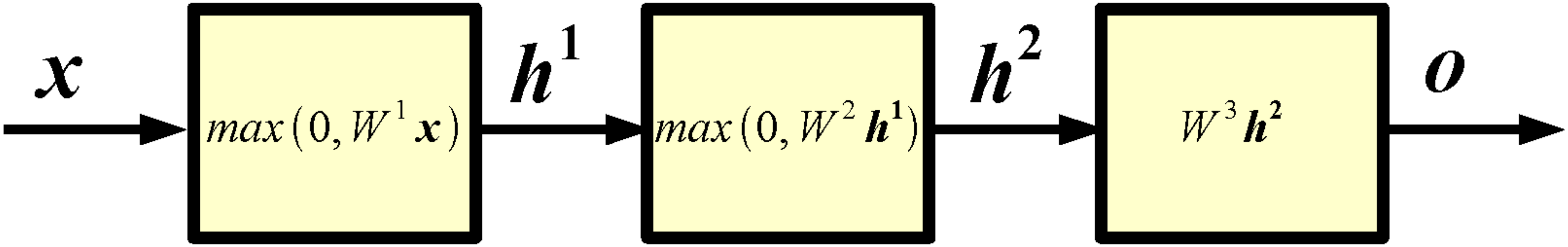


“2 3 4 5”

Outline

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

Neural Networks: example



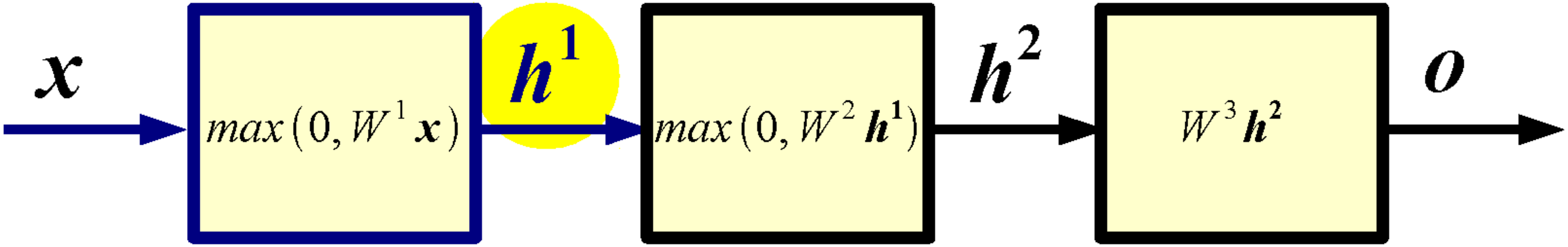
$\boldsymbol{x}$	input
$\boldsymbol{h}^1$	1-st layer hidden units
$\boldsymbol{h}^2$	2-nd layer hidden units
$\boldsymbol{o}$	output

Example of a 2 hidden layer neural network (or 4 layer network, counting also input and output).

Forward Propagation

Def.: Forward propagation is the process of computing the output of the network given its input.

Forward Propagation



$\boldsymbol{x} \in R^D \quad W^1 \in R^{N_1 \times D} \quad \boldsymbol{b}^1 \in R^{N_1} \quad \boldsymbol{h}^1 \in R^{N_1}$

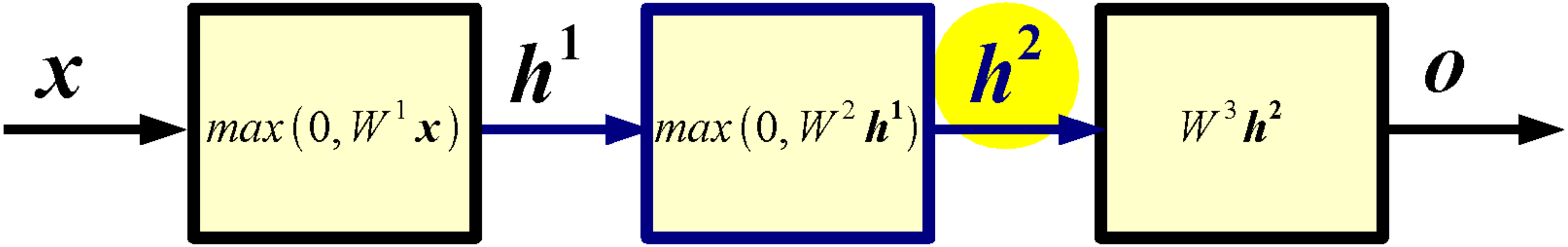
$$\boldsymbol{h}^1 = max(0, W^1 \boldsymbol{x} + \boldsymbol{b}^1)$$

W^1 1-st layer weight matrix or weights

$\boldsymbol{b}^1$ 1-st layer biases

The non-linearity $u = max(0, v)$ is called **ReLU** in the DL literature. Each output hidden unit takes as input all the units at the previous layer: each such layer is called "**fully connected**".

Forward Propagation



$\boldsymbol{x}$

$max(0, W^1 \boldsymbol{x})$

$\boldsymbol{h}^1$

$max(0, W^2 \boldsymbol{h}^1)$

$\boldsymbol{h}^2$

$W^3 \boldsymbol{h}^2$

$\boldsymbol{o}$

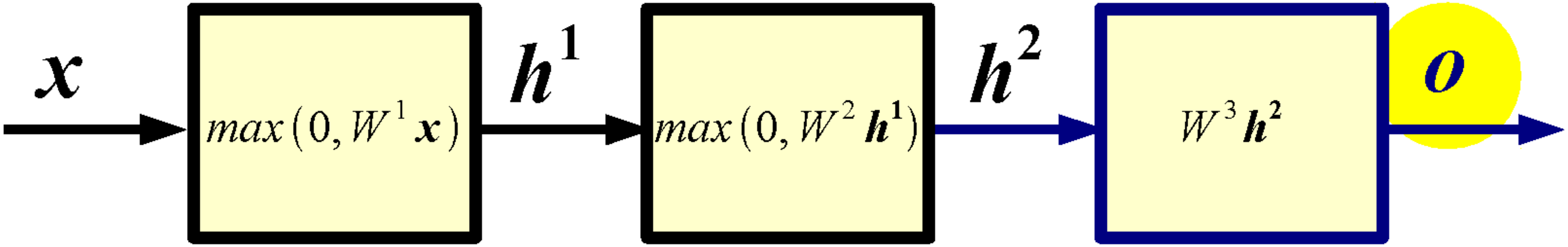
$\boldsymbol{h}^1 \in R^{N_1} \quad W^2 \in R^{N_2 \times N_1} \quad \boldsymbol{b}^2 \in R^{N_2} \qquad \boldsymbol{h}^2 \in R^{N_2}$

$$\boldsymbol{h}^2 = max(0, W^2 \boldsymbol{h}^1 + \boldsymbol{b}^2)$$

W^2 2-nd layer weight matrix or weights

$\boldsymbol{b}^2$ 2-nd layer biases

Forward Propagation



$\boldsymbol{x}$ $max(0, W^1 \boldsymbol{x})$ $\boldsymbol{h}^1$ $max(0, W^2 \boldsymbol{h}^1)$ $\boldsymbol{h}^2$ $W^3 \boldsymbol{h}^2$ $\boldsymbol{o}$

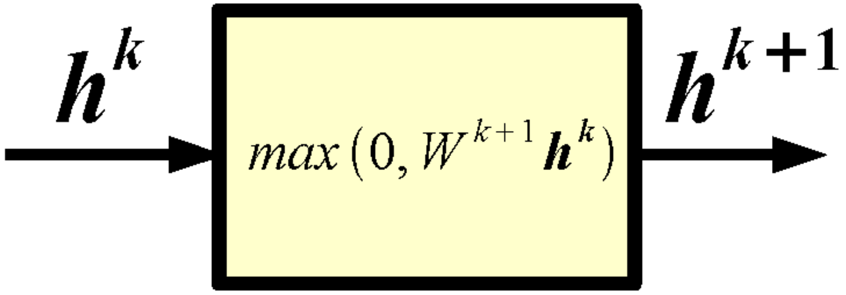
$\boldsymbol{h}^2 \in R^{N_2}$ $W^3 \in R^{N_3 \times N_2}$ $\boldsymbol{b}^3 \in R^{N_3}$ $\boldsymbol{o} \in R^{N_3}$

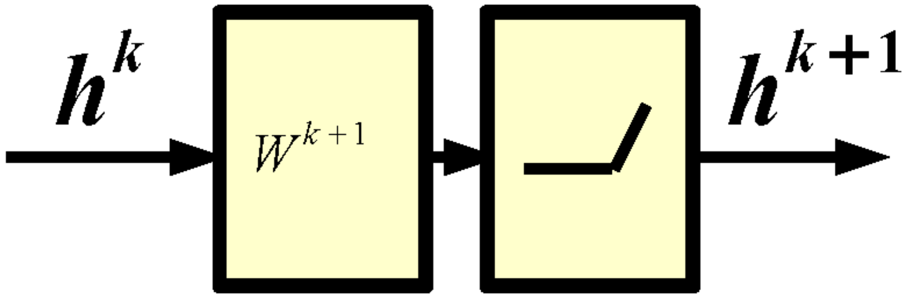
$$\boldsymbol{o} = max(0, W^3 \boldsymbol{h}^2 + \boldsymbol{b}^3)$$

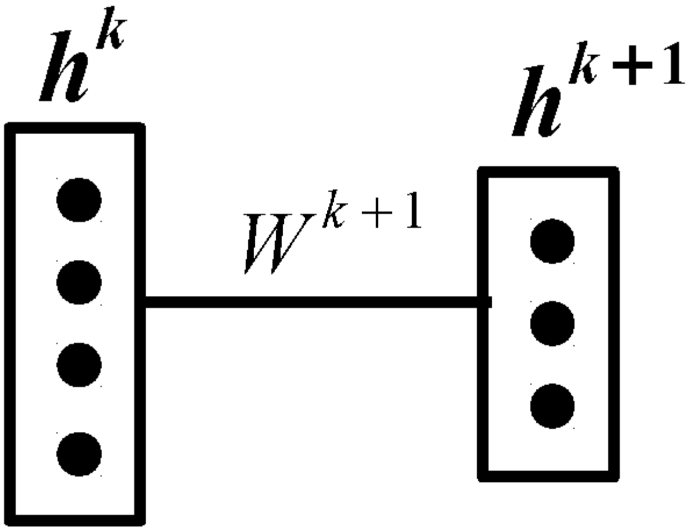
W^3 3-rd layer weight matrix or weights

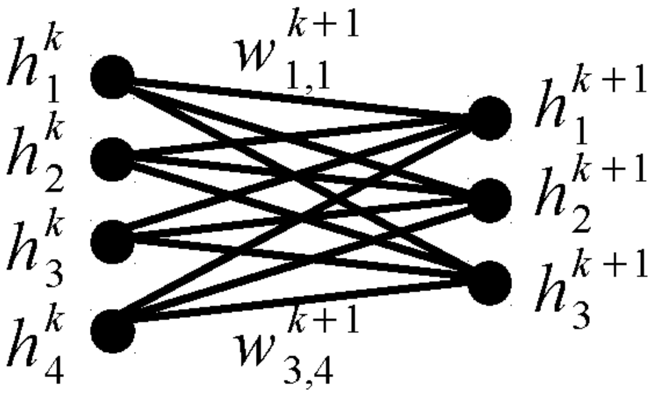
$\boldsymbol{b}^3$ 3-rd layer biases

Alternative Graphical Representation

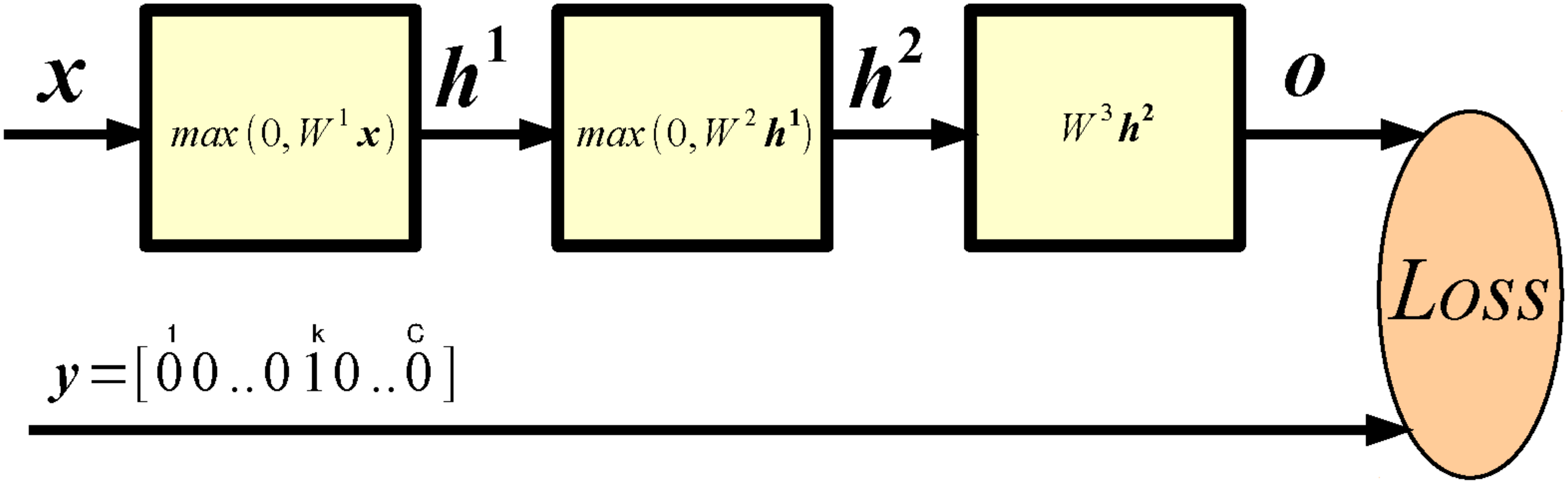








How Good is a Network?



$\boldsymbol{x}$
$max(0, W^1 \boldsymbol{x})$
$\boldsymbol{h}^1$
$max(0, W^2 \boldsymbol{h}^1)$
$\boldsymbol{h}^2$
$W^3 \boldsymbol{h}^2$
$\boldsymbol{o}$
$Loss$
$\boldsymbol{y} = [\overset{1}{0}\,0\,..\,0\,\overset{k}{1}\,0\,..\,\overset{C}{0}]$

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.

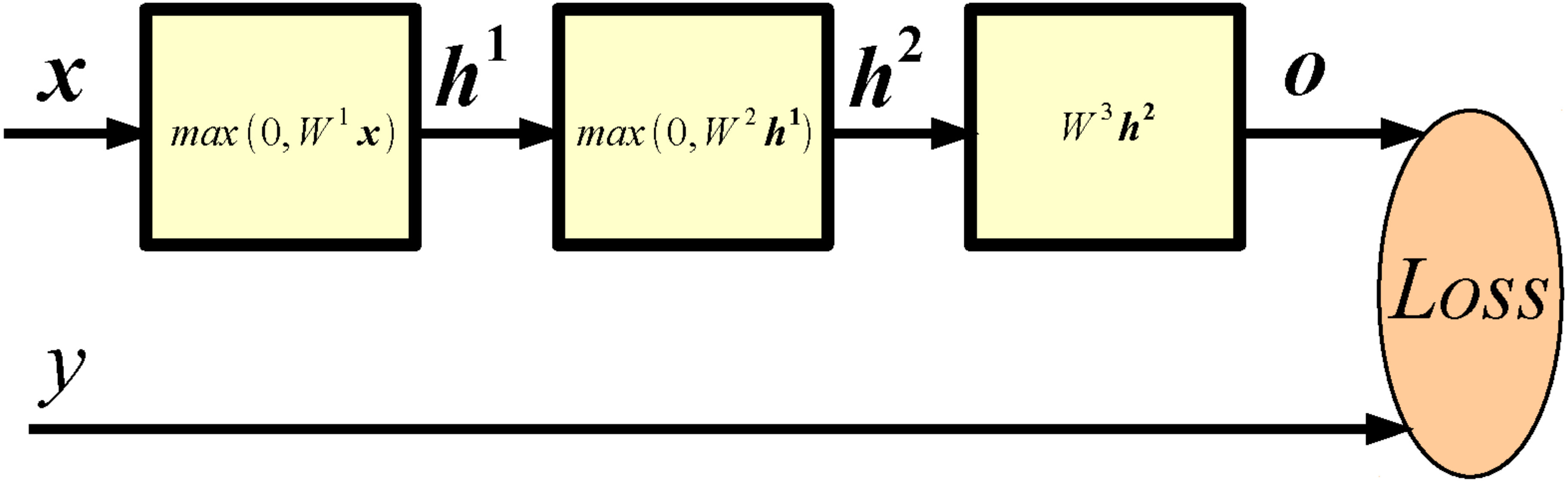
$$\boldsymbol{\theta}^* = arg\ min_{\boldsymbol{\theta}} \sum_{n=1}^{P} L(\mathbf{x}^n, y^n; \boldsymbol{\theta})$$

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:

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

Derivative w.r.t. Input of Softmax

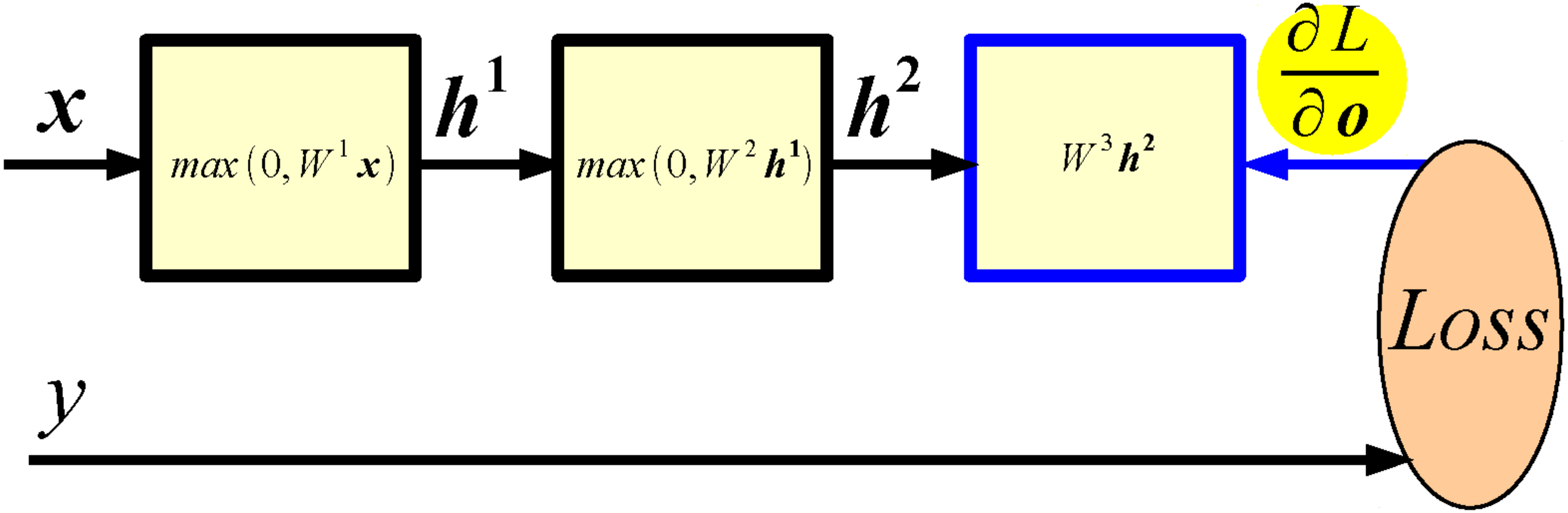
$$p(c_k=1|\boldsymbol{x})=\frac{e^{o_k}}{\sum_j e^{o_j}}$$

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

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

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

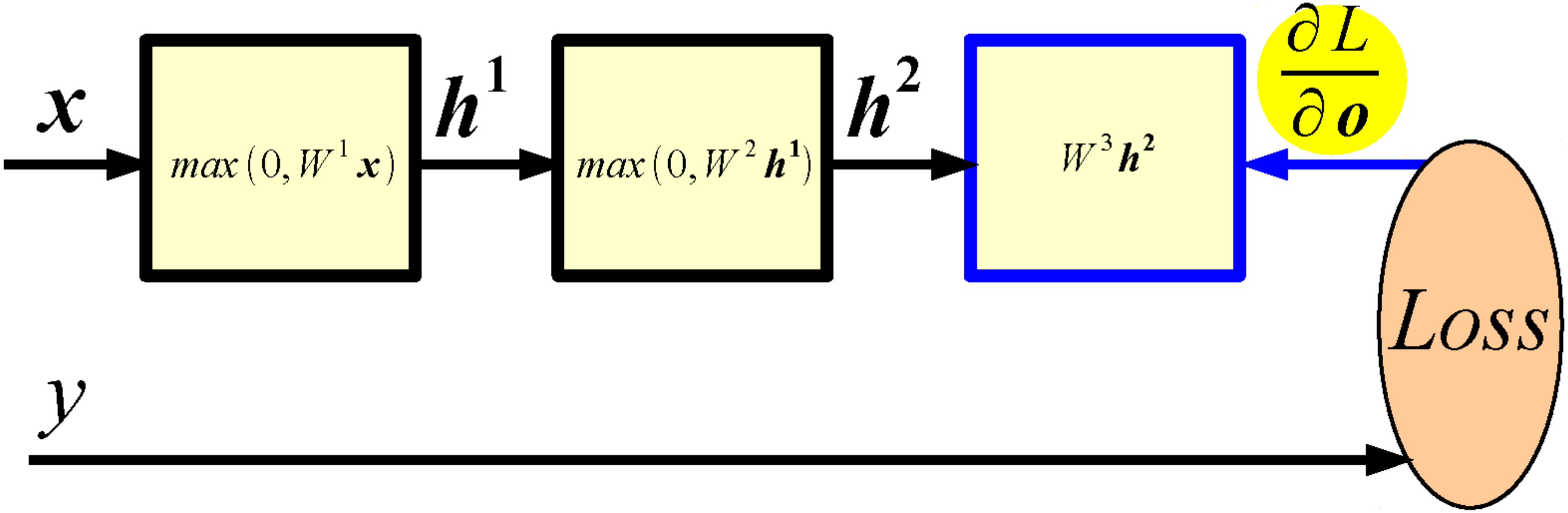
Backward Propagation


x
$\max(0, W^1 \boldsymbol{x})$
$\boldsymbol{h}^1$
$\max(0, W^2 \boldsymbol{h}^1)$
$\boldsymbol{h}^2$
$W^3 \boldsymbol{h}^2$
$\frac{\partial L}{\partial \boldsymbol{o}}$
$Loss$
y

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

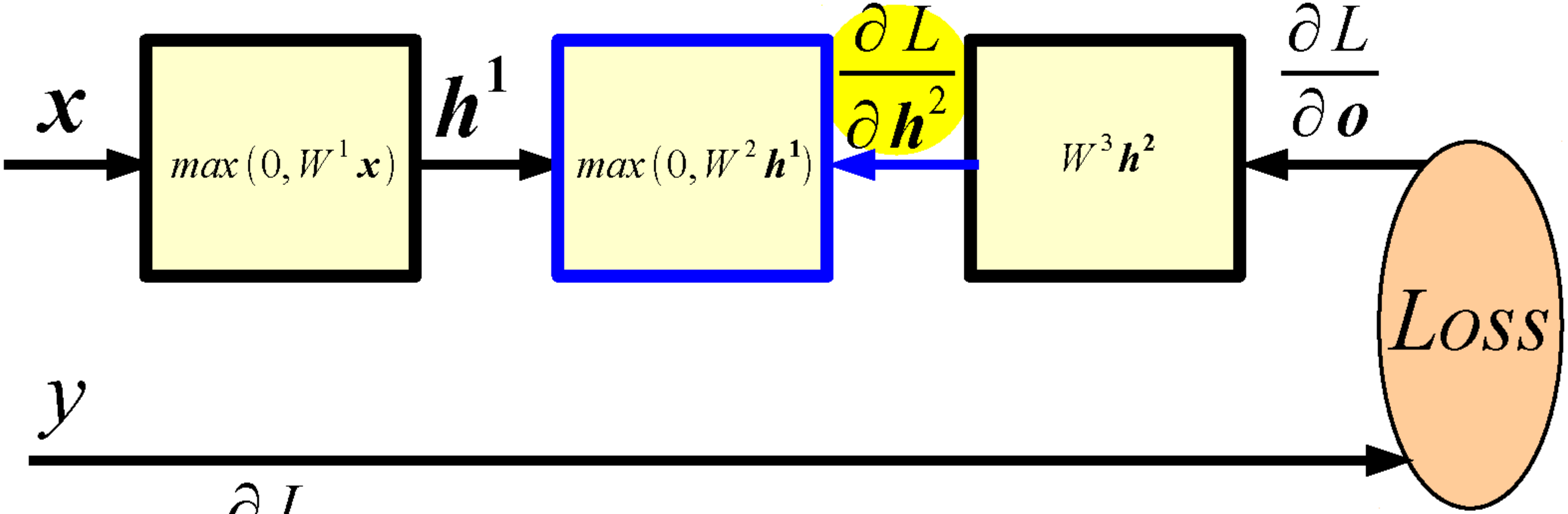

x
$max(0, W^1 \boldsymbol{x})$
$\boldsymbol{h}^1$
$max(0, W^2 \boldsymbol{h}^1)$
$\boldsymbol{h}^2$
$W^3 \boldsymbol{h}^2$
$\frac{\partial L}{\partial \boldsymbol{o}}$
$Loss$
y

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

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

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

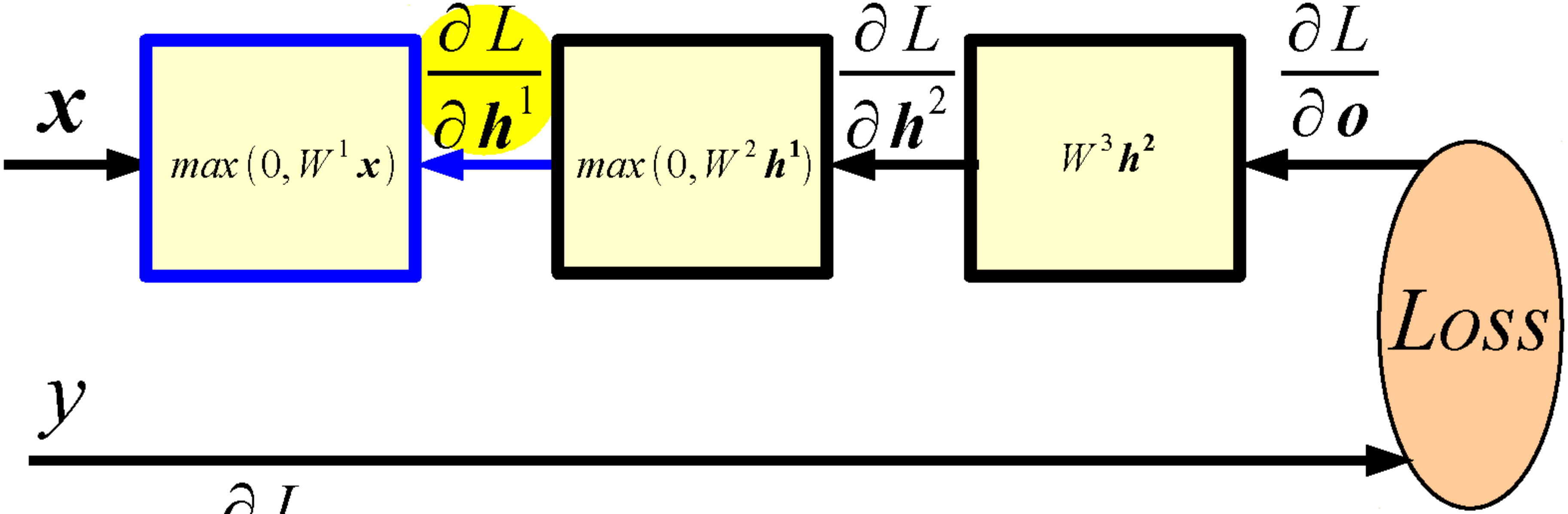
Backward Propagation


x
$\max(0, W^1 x)$
h^1
$\max(0, W^2 h^1)$
$\frac{\partial L}{\partial h^2}$
$W^3 h^2$
$\frac{\partial L}{\partial o}$
Loss
y

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

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

Backward Propagation


$\boldsymbol{x}$
$\frac{\partial L}{\partial \boldsymbol{h}^1}$
$\frac{\partial L}{\partial \boldsymbol{h}^2}$
$\frac{\partial L}{\partial \boldsymbol{o}}$
$max(0, W^1 \boldsymbol{x})$
$max(0, W^2 \boldsymbol{h}^1)$
$W^3 \boldsymbol{h}^2$
$Loss$
y

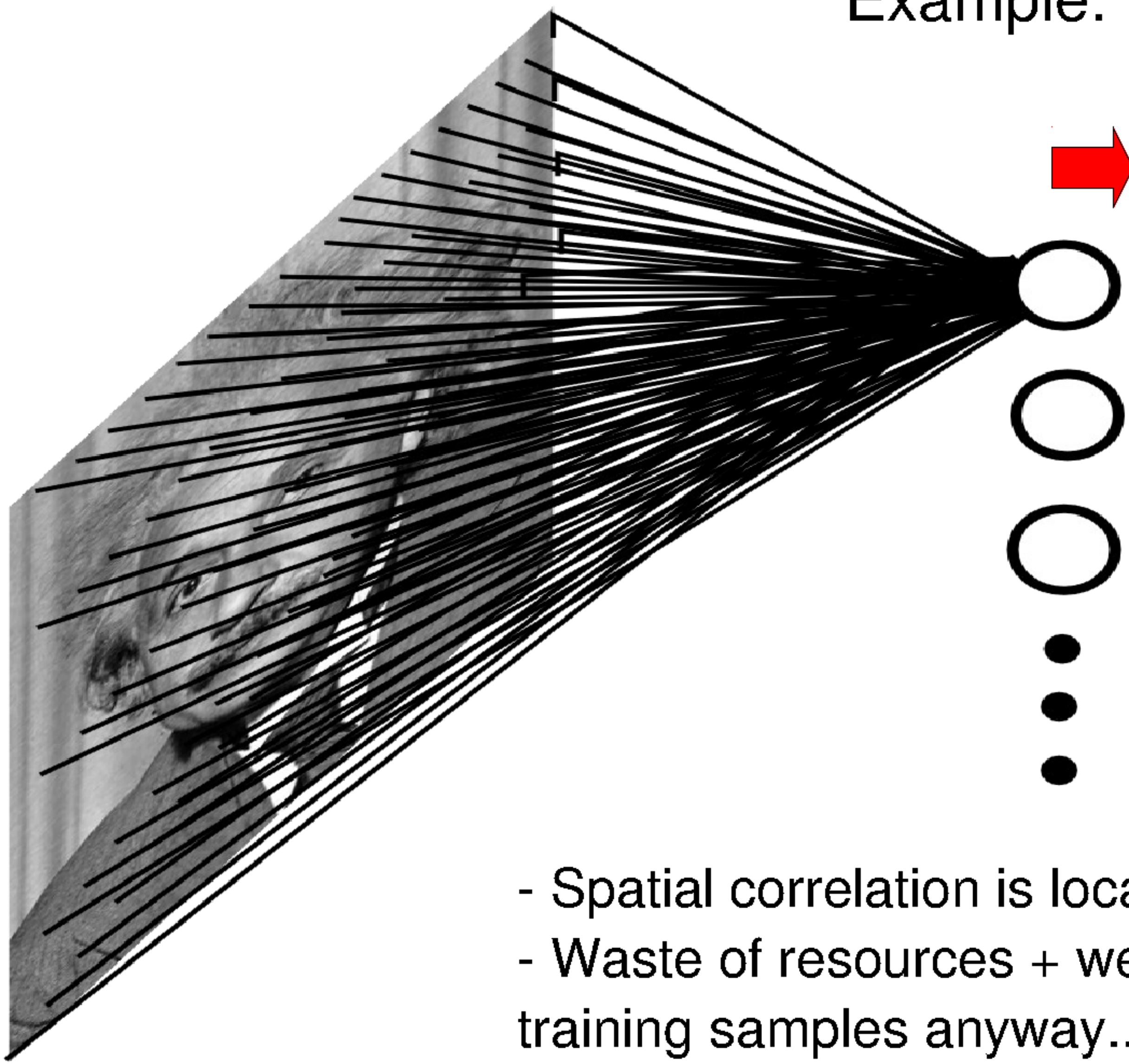
Given $\frac{\partial L}{\partial \boldsymbol{h}^1}$ we can compute now:

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

Outline

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

Fully Connected Layer



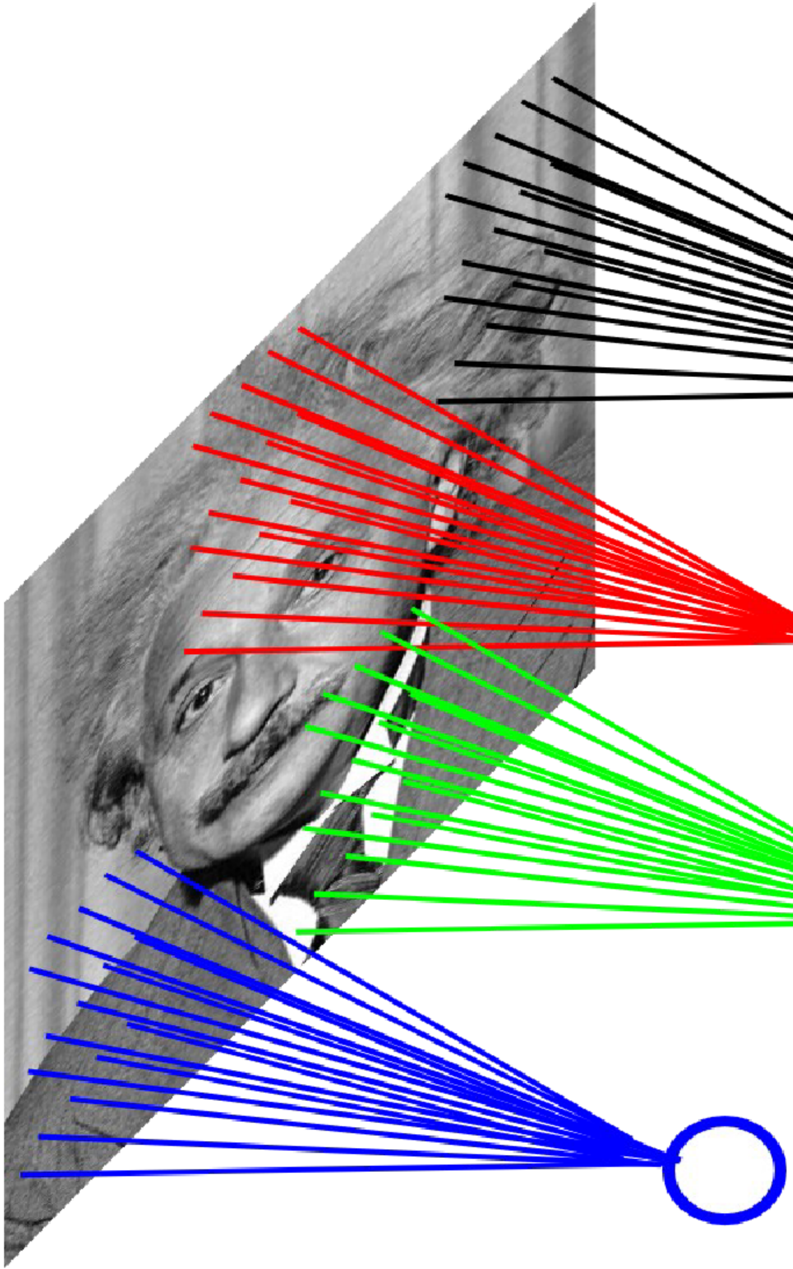
Example: 200x200 image

40K hidden units

~2B parameters!!!

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

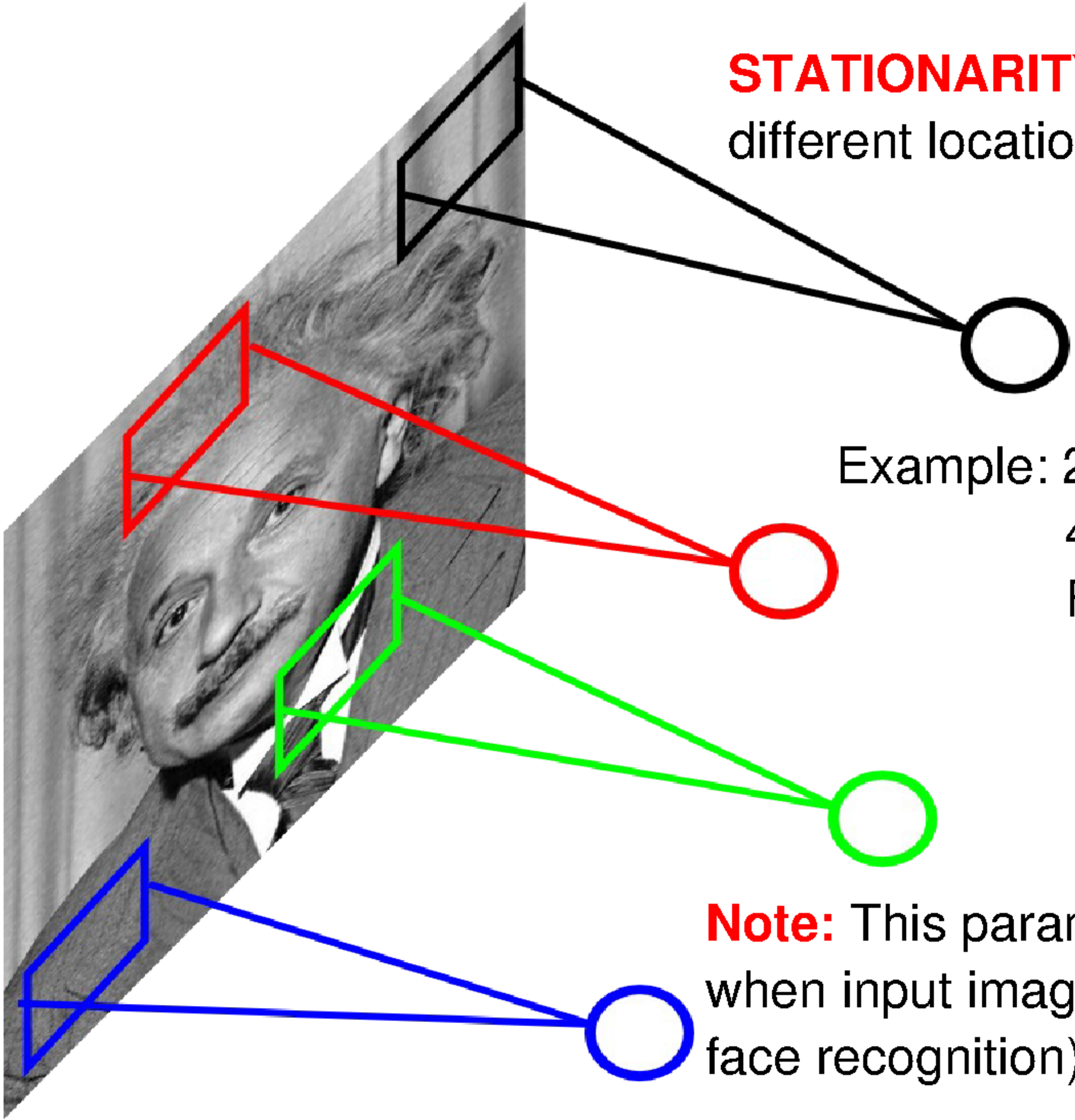
Locally Connected Layer



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).

Locally Connected Layer



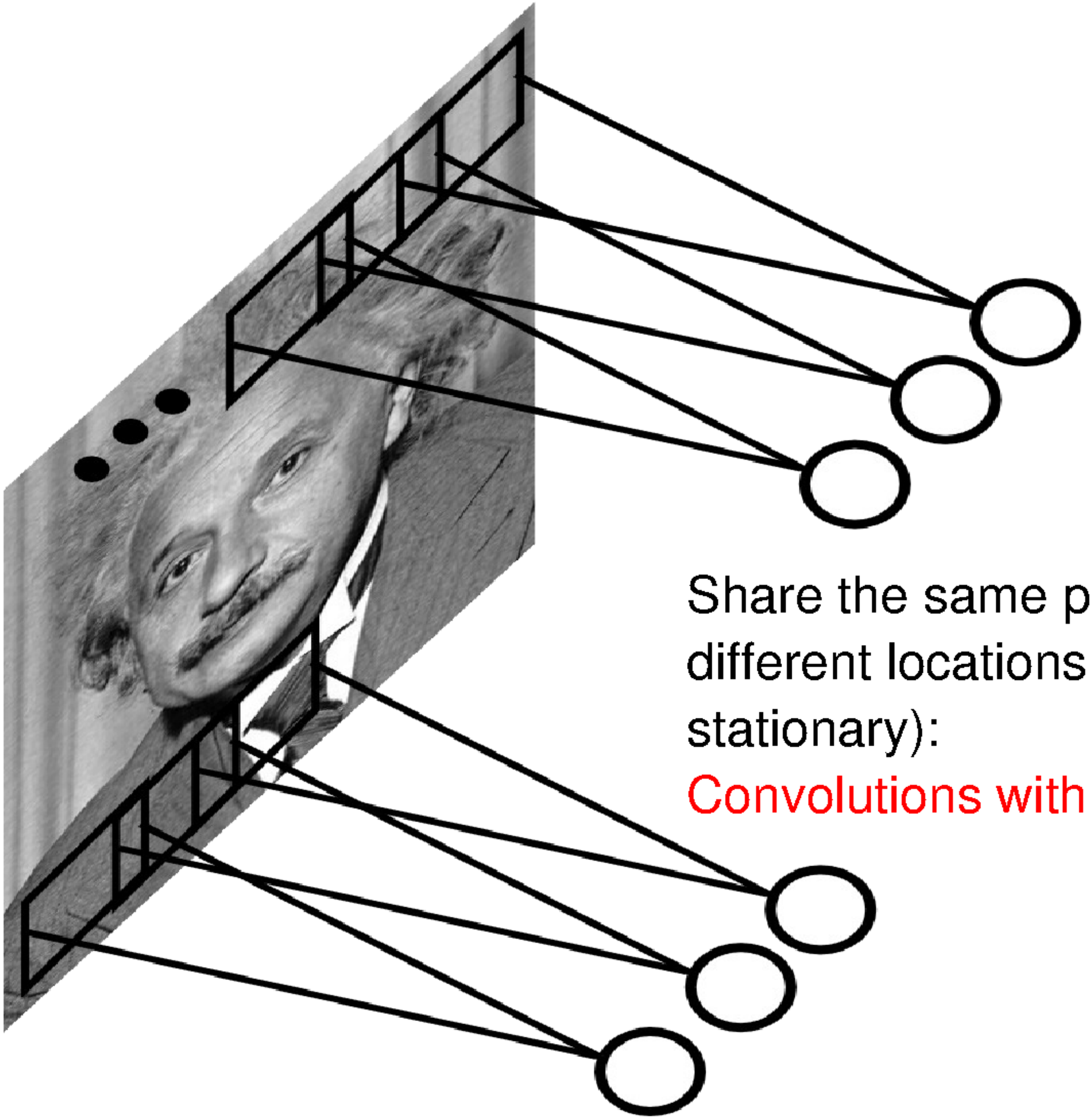
STATIONARITY? Statistics is similar at different locations

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

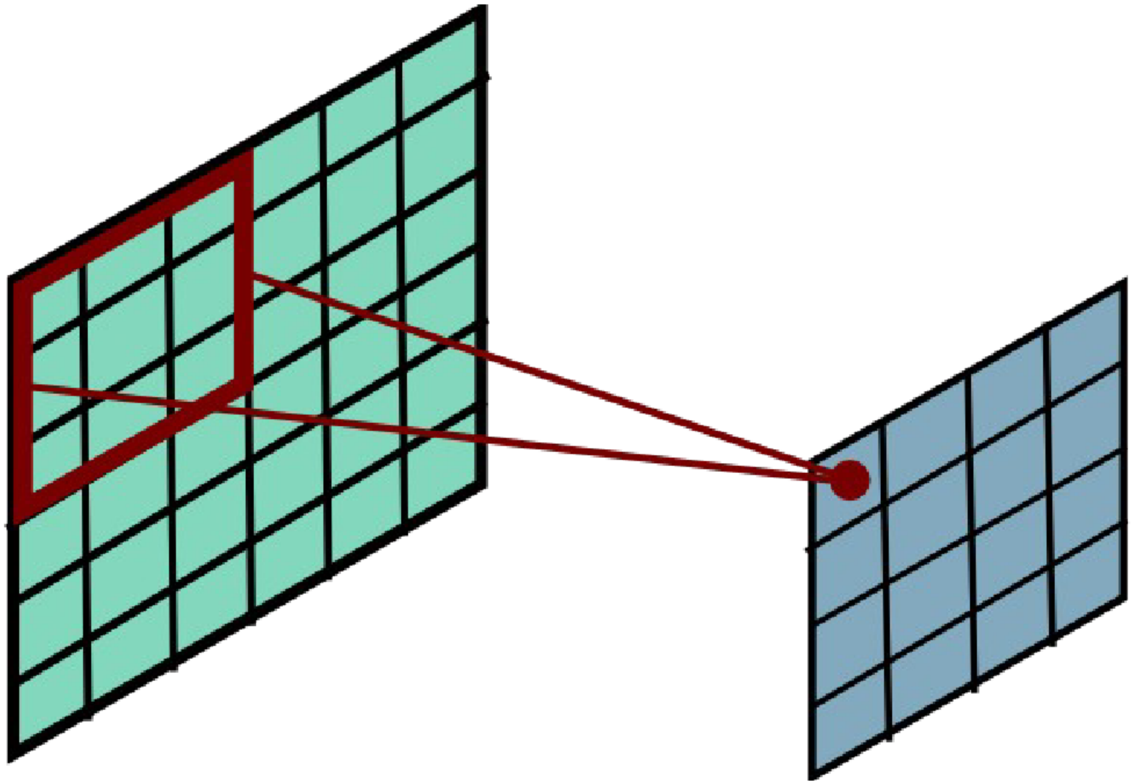
Convolutional Layer



Share the same parameters across different locations (assuming input is stationary):

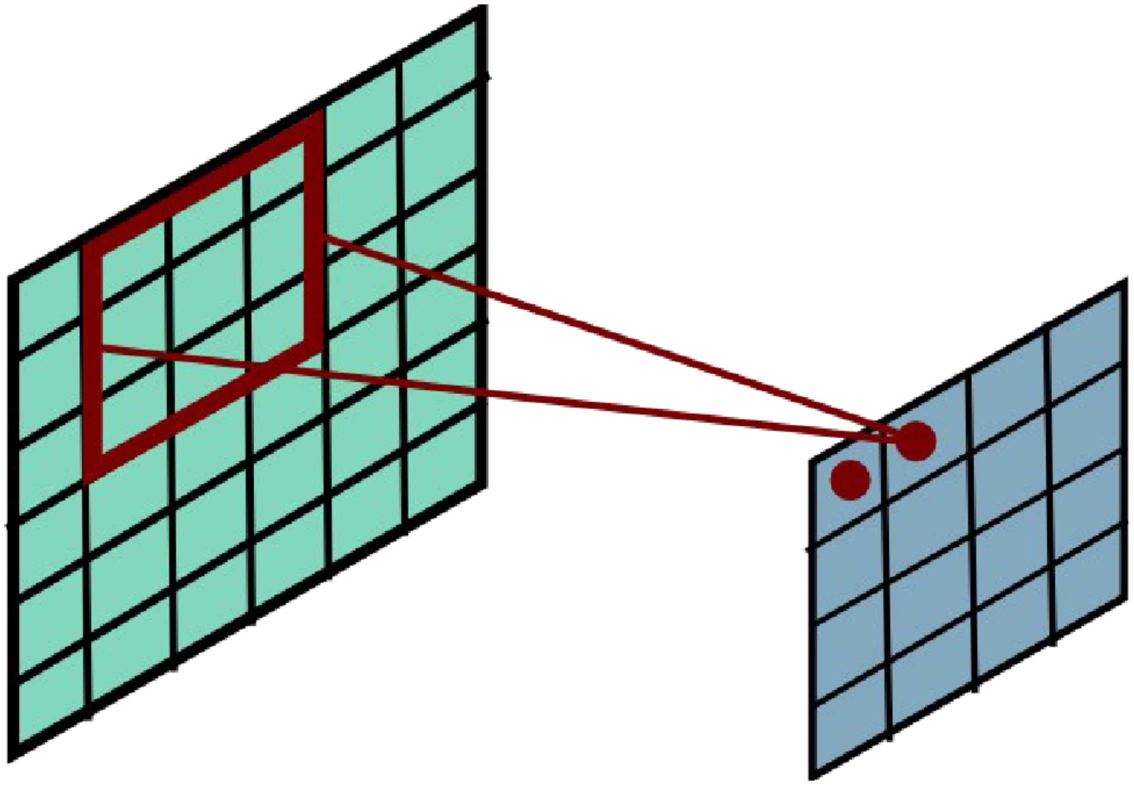
Convolutions with learned kernels

Convolutional Layer

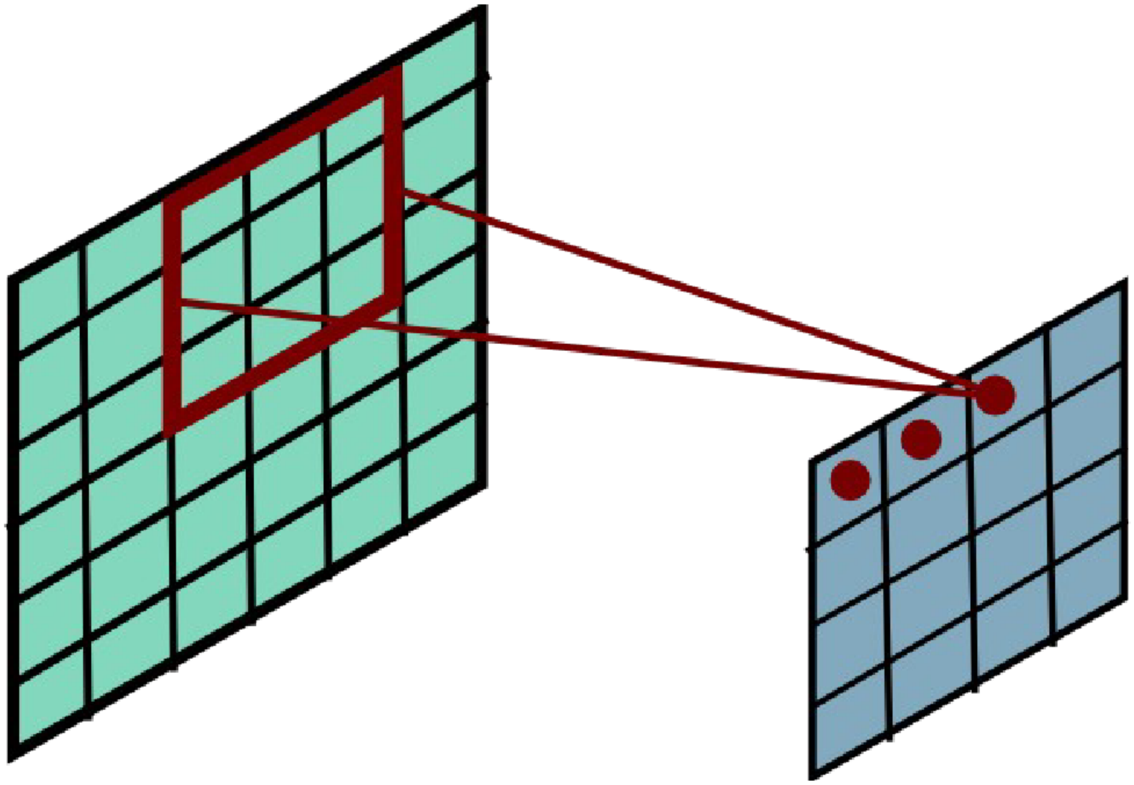


Ranzato

Convolutional Layer

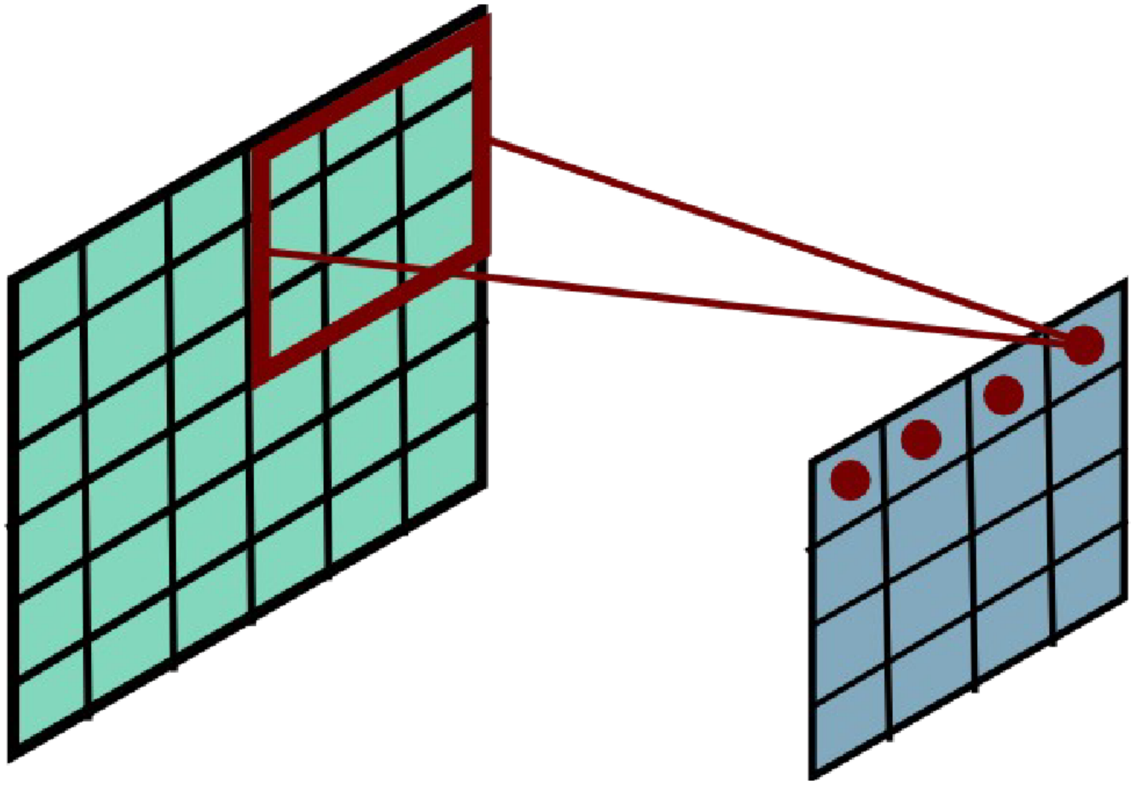


Convolutional Layer



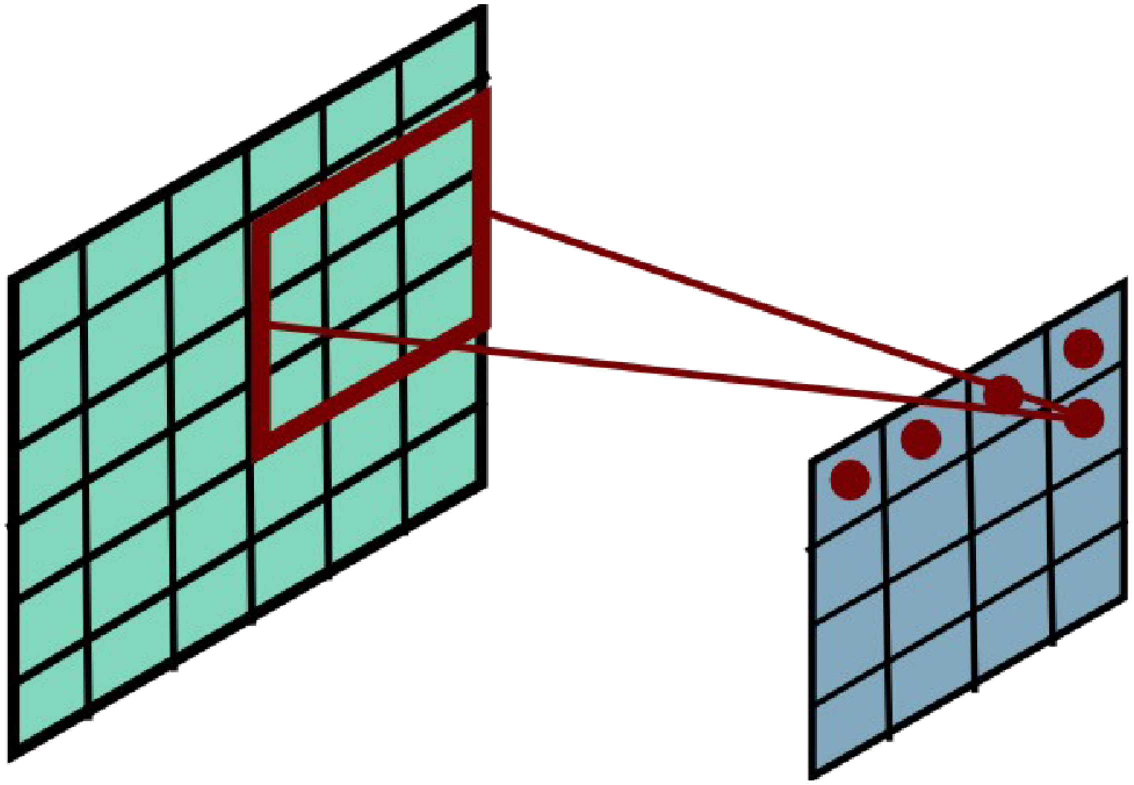
Ranzato

Convolutional Layer



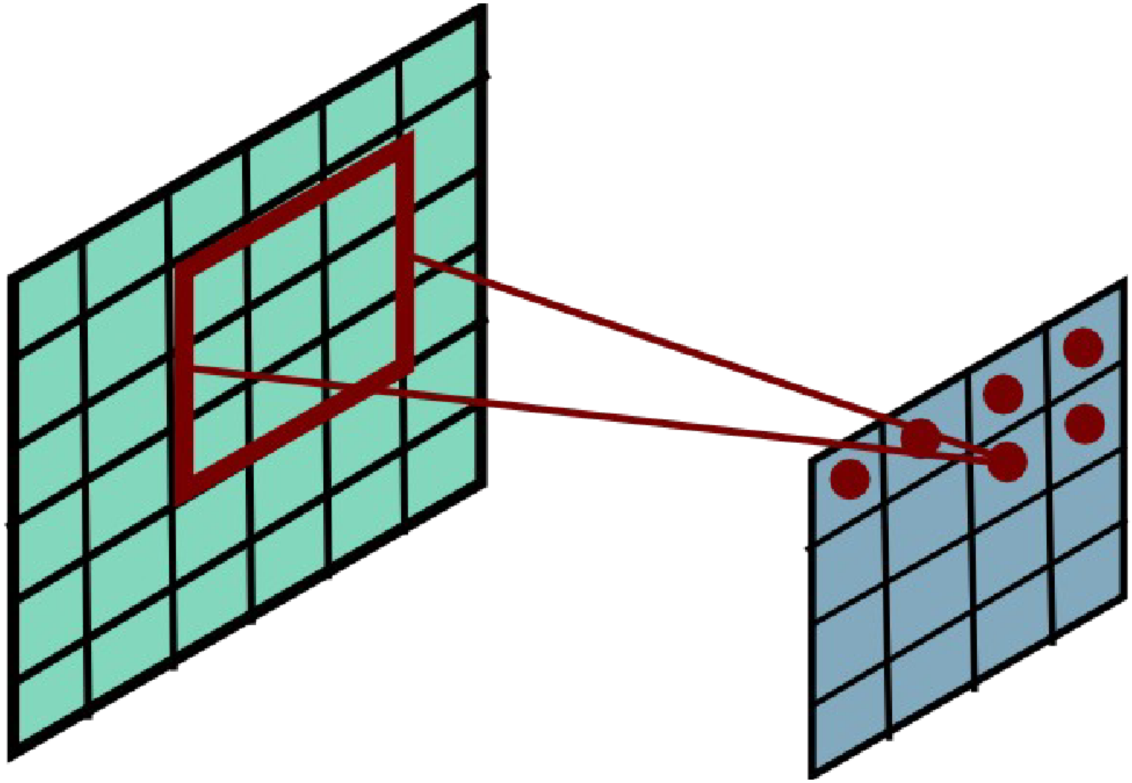
Ranzato

Convolutional Layer



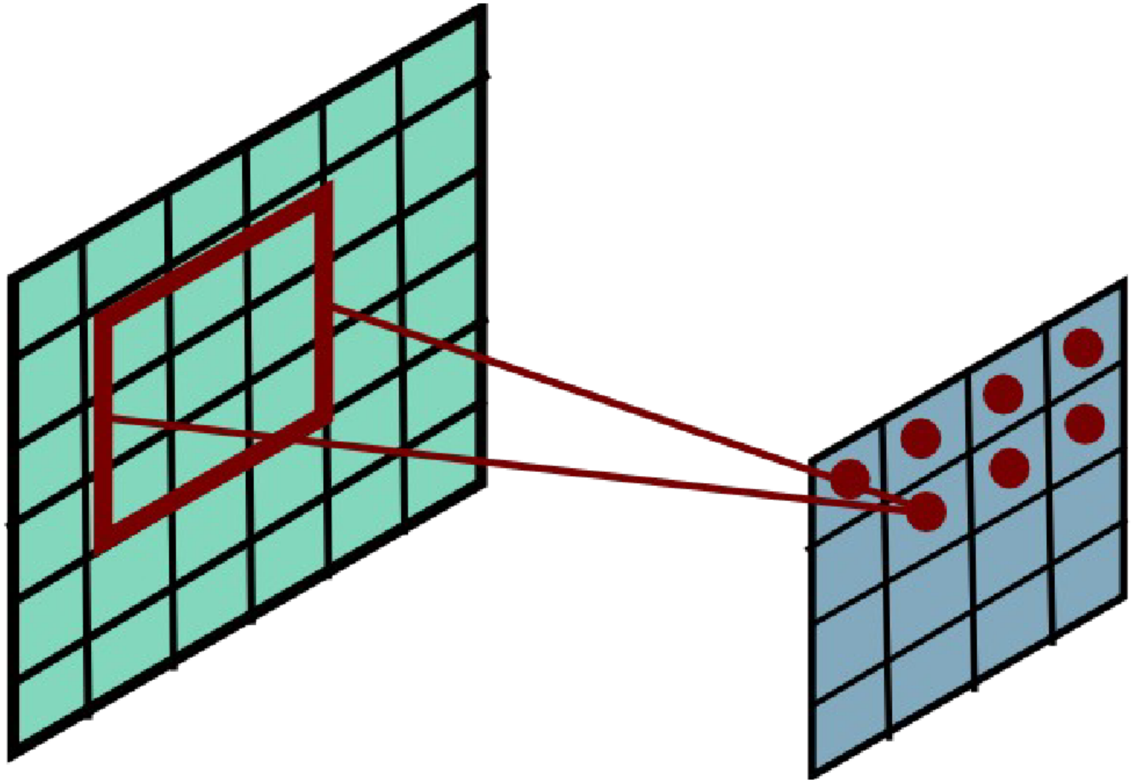
Ranzato

Convolutional Layer



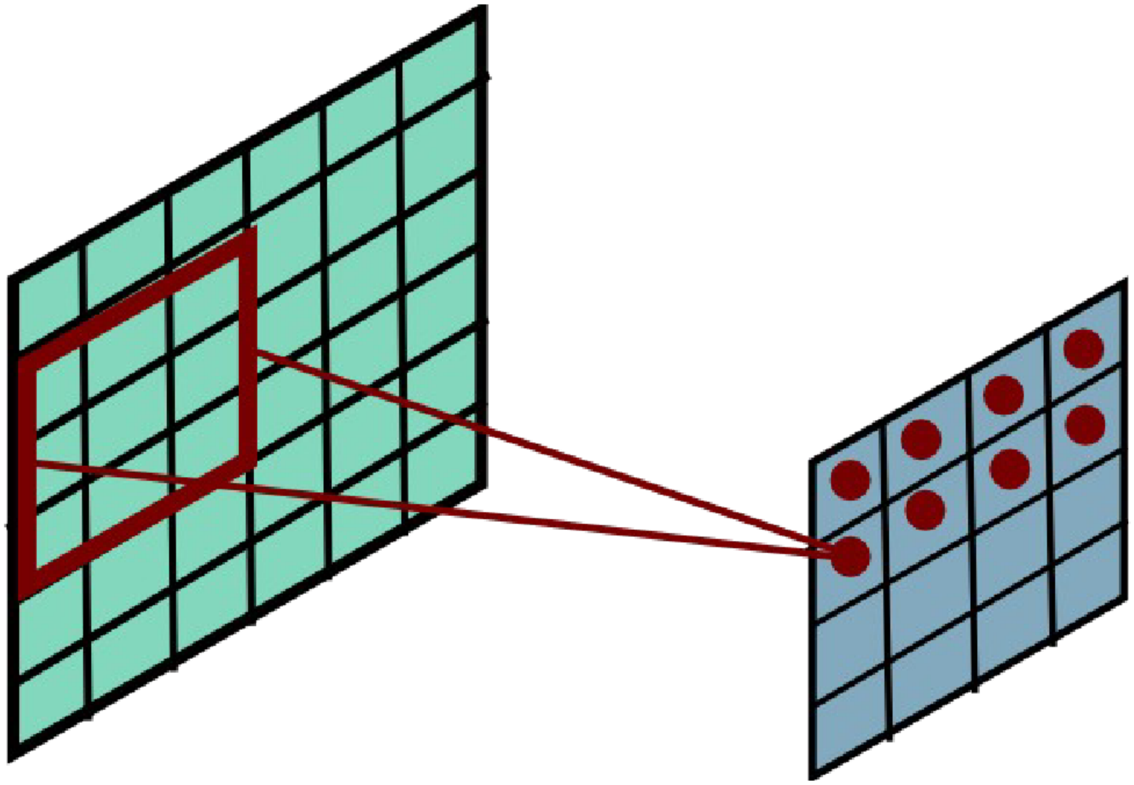
Ranzato

Convolutional Layer



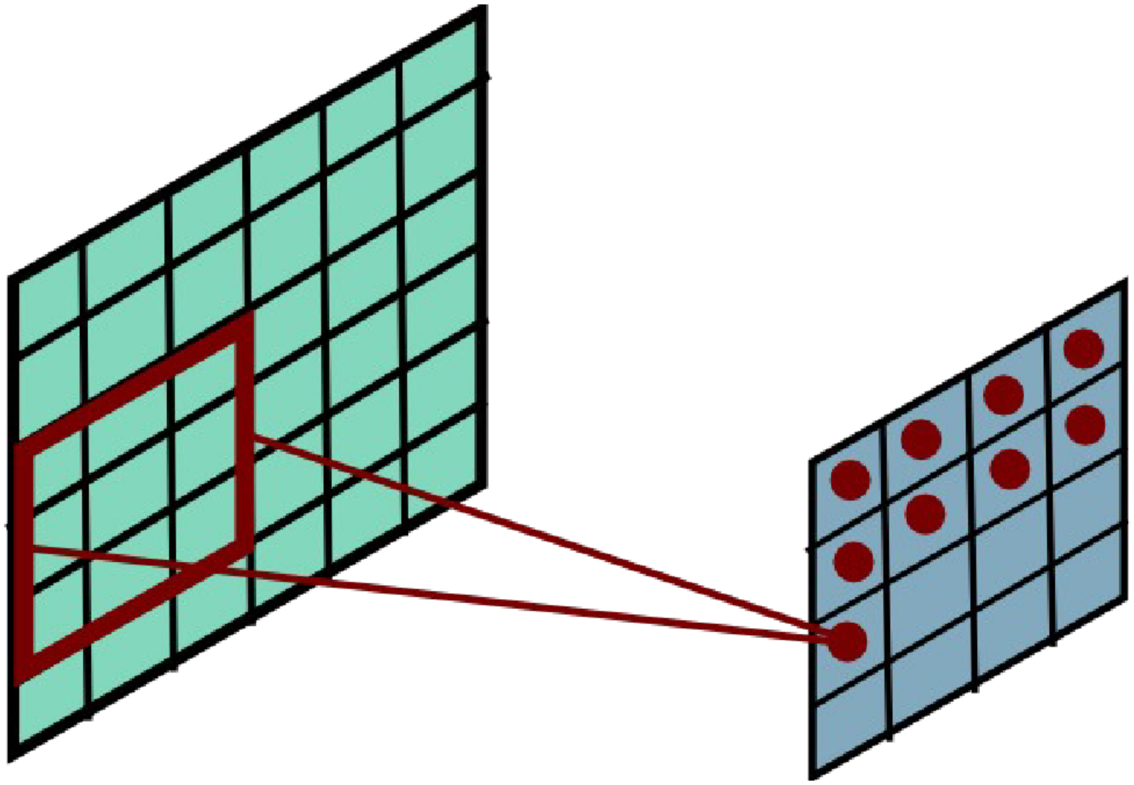
Ranzato

Convolutional Layer



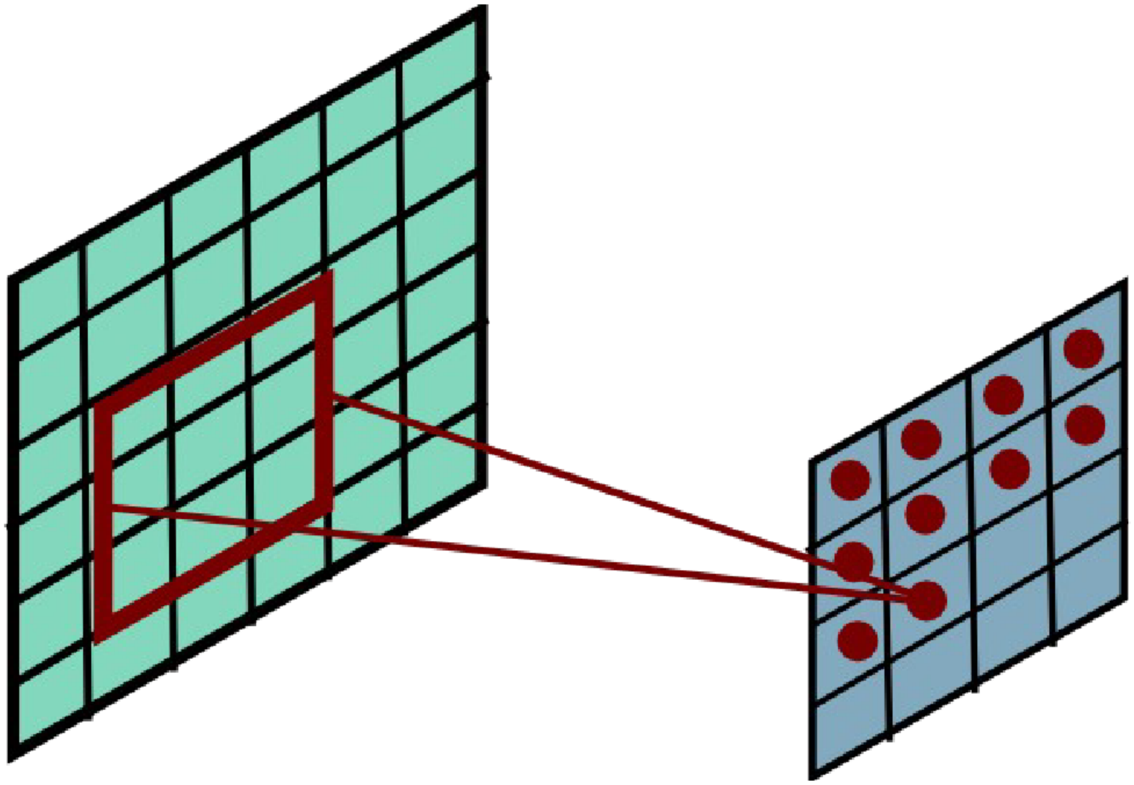
Ranzato

Convolutional Layer

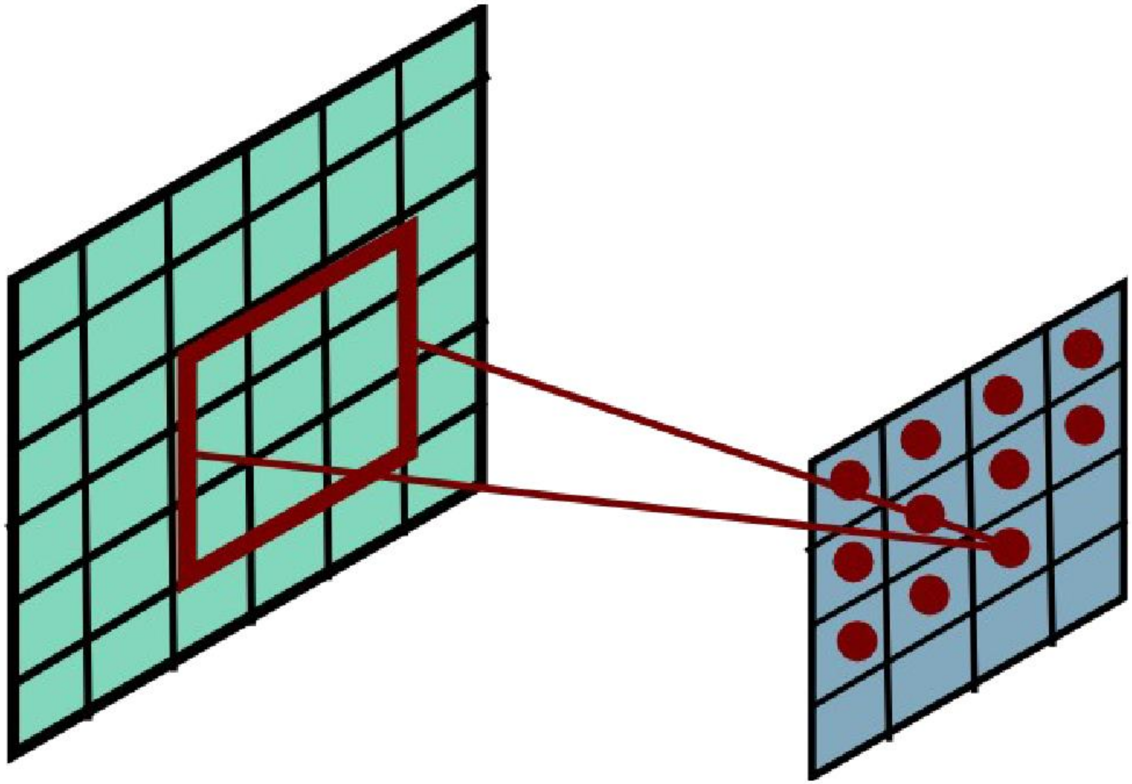


Ranzato

Convolutional Layer

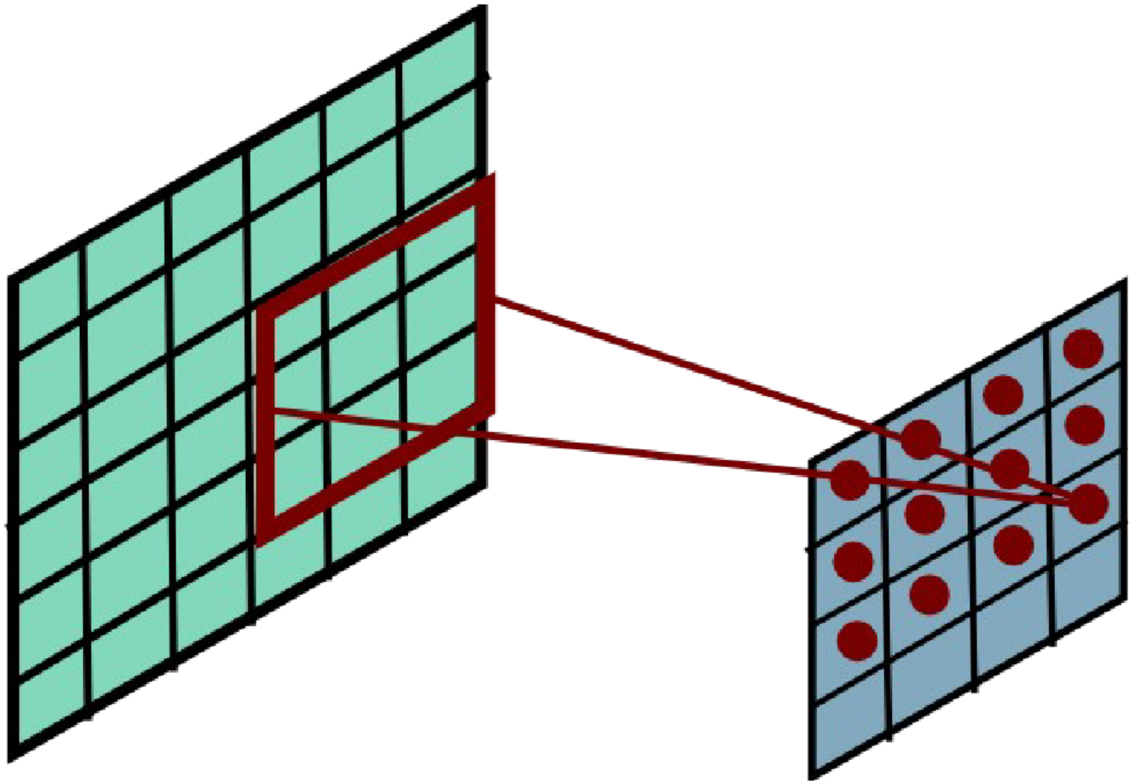


Convolutional Layer



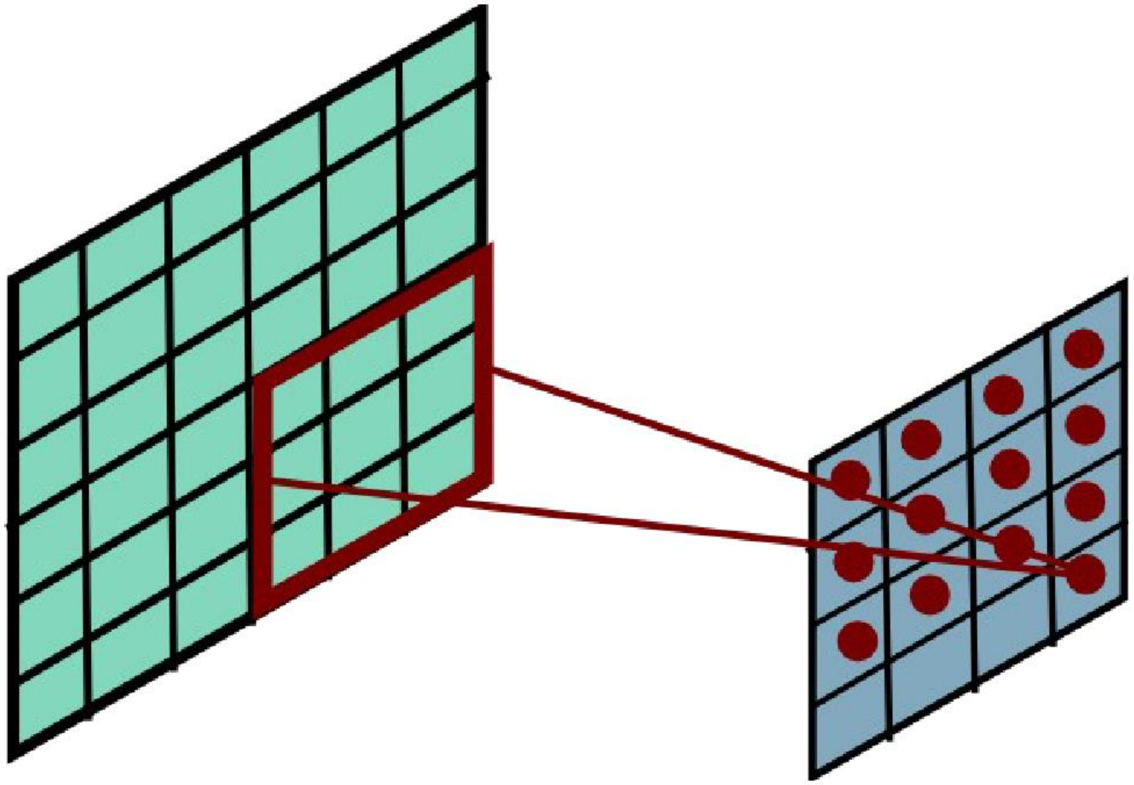
Ranzato

Convolutional Layer



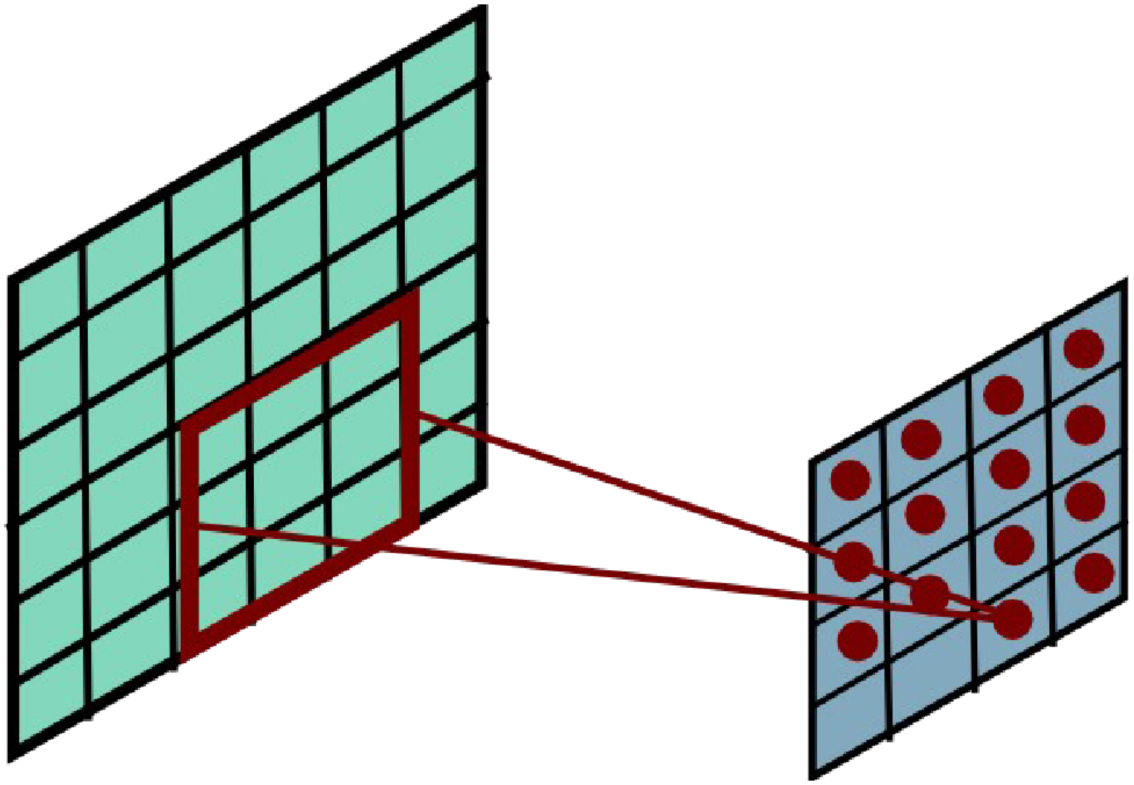
Ranzato

Convolutional Layer

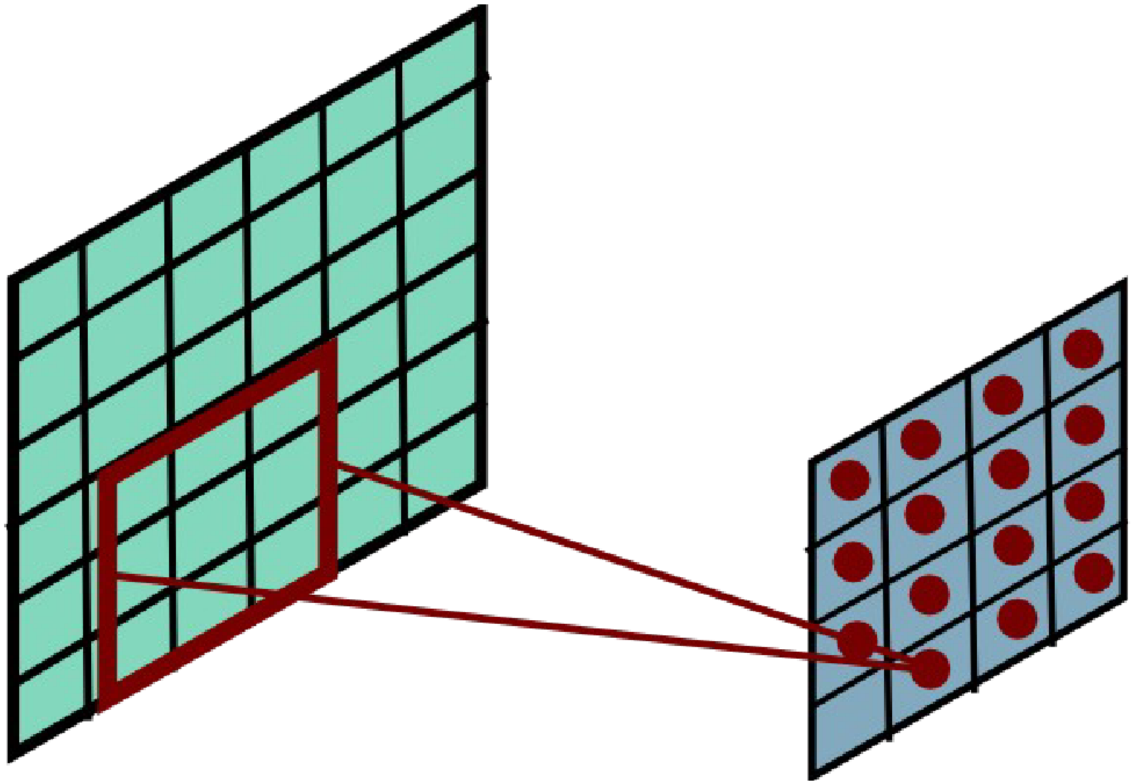


Ranzato

Convolutional Layer

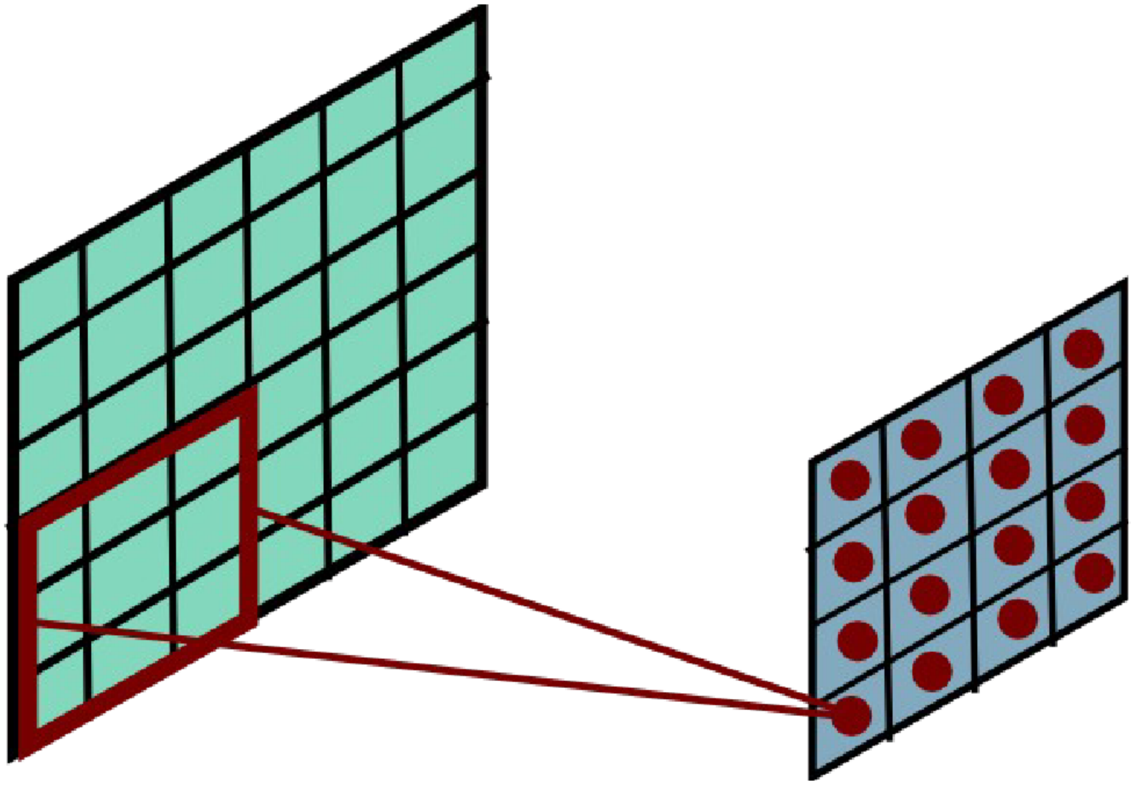


Convolutional Layer



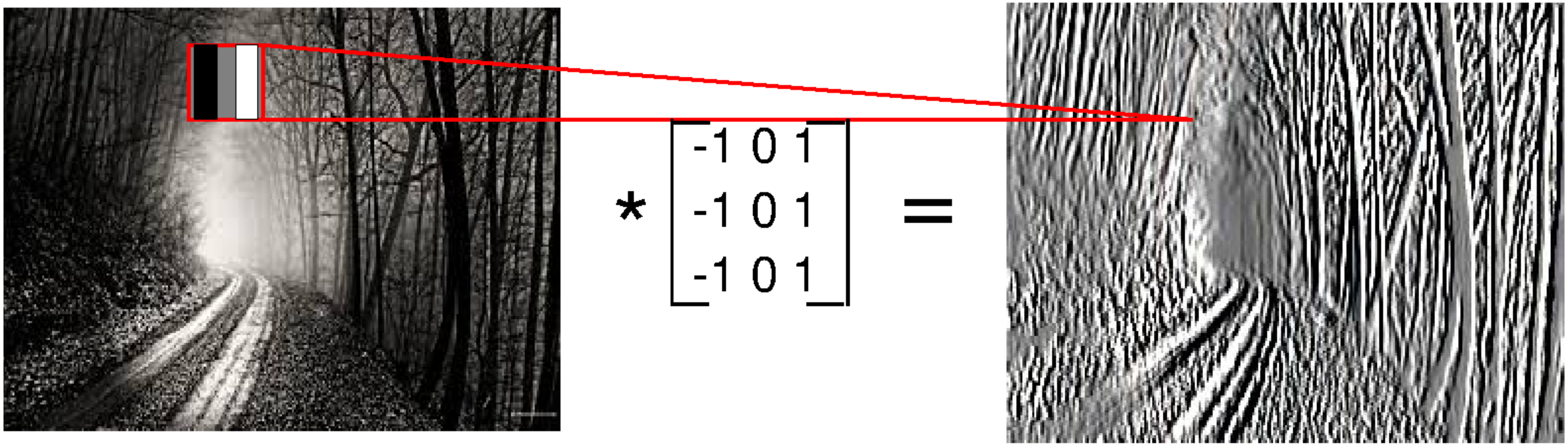
Ranzato

Convolutional Layer



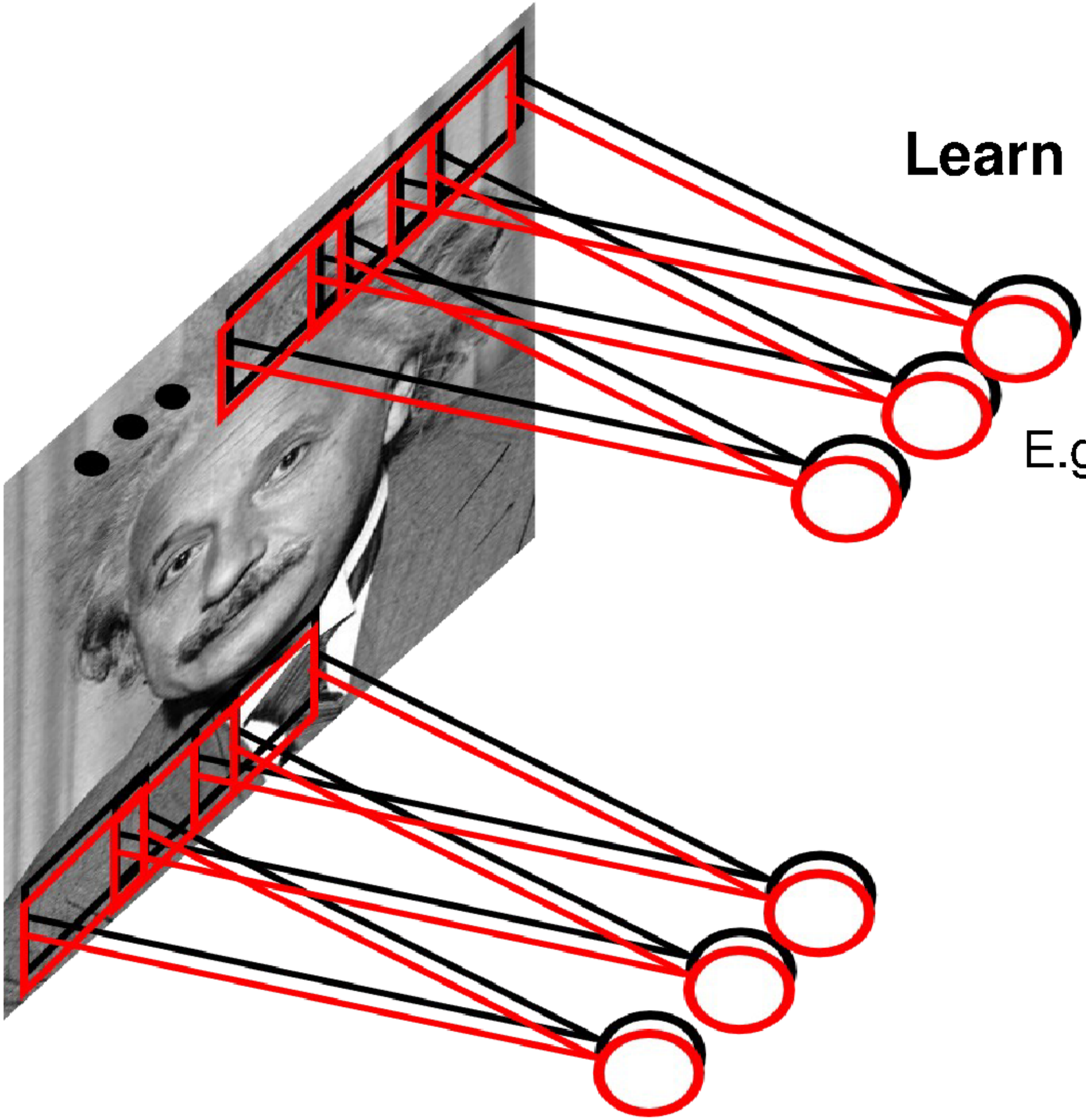
Ranzato

Convolutional Layer



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

Convolutional Layer



Learn multiple filters.

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

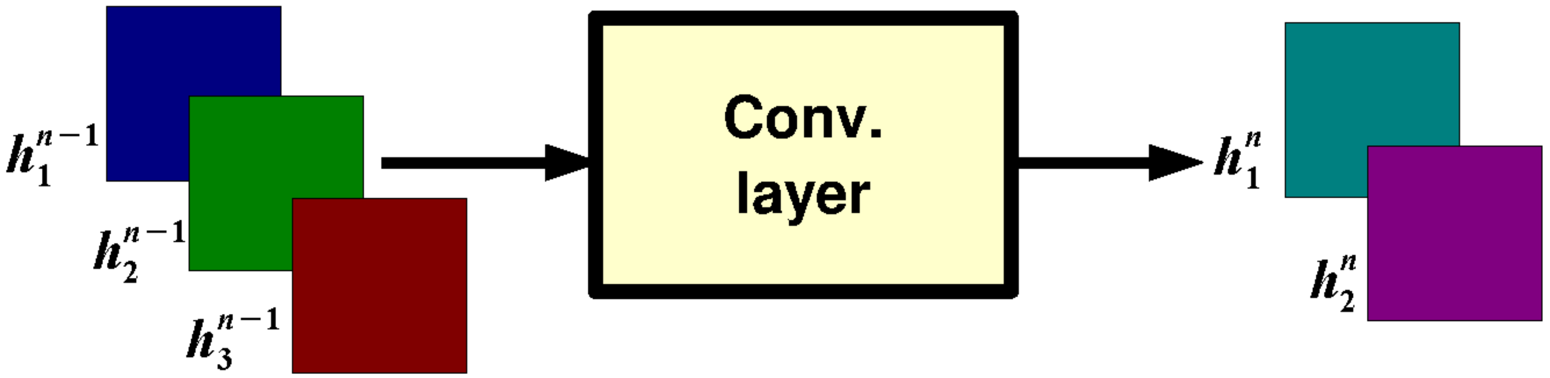
Convolutional Layer

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

output feature map

input feature map

kernel



Conv. layer

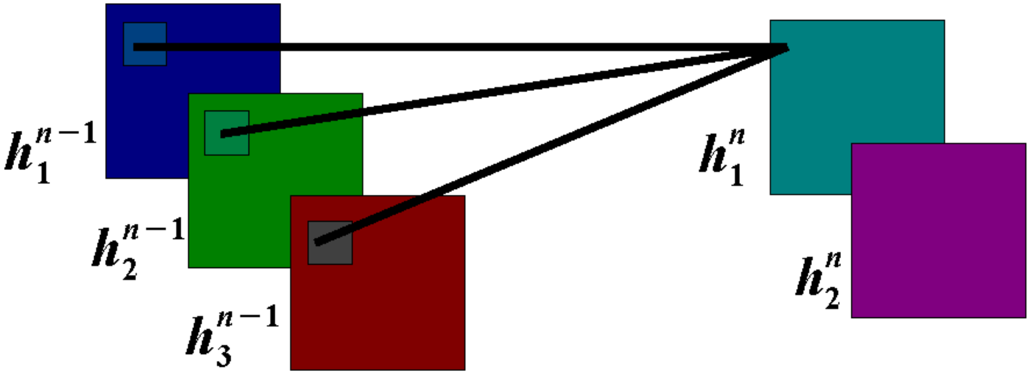
Convolutional Layer

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

output feature map

input feature map

kernel



Ranzato

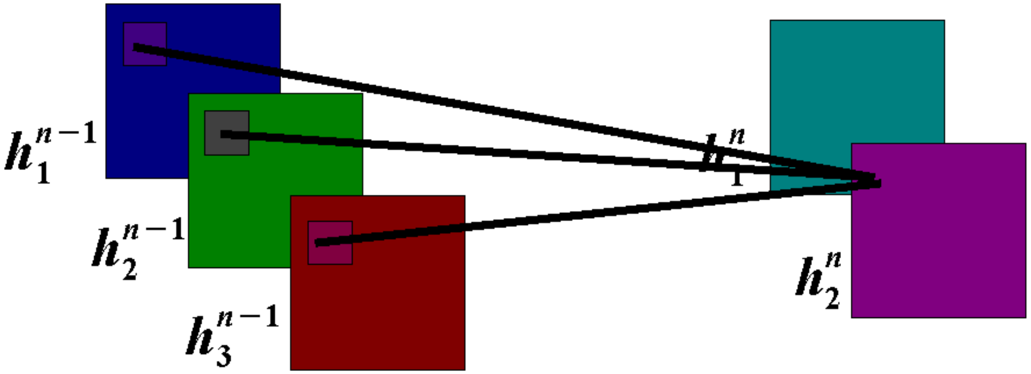
Convolutional Layer

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

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).

Key Ideas

A standard neural net applied to images:

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

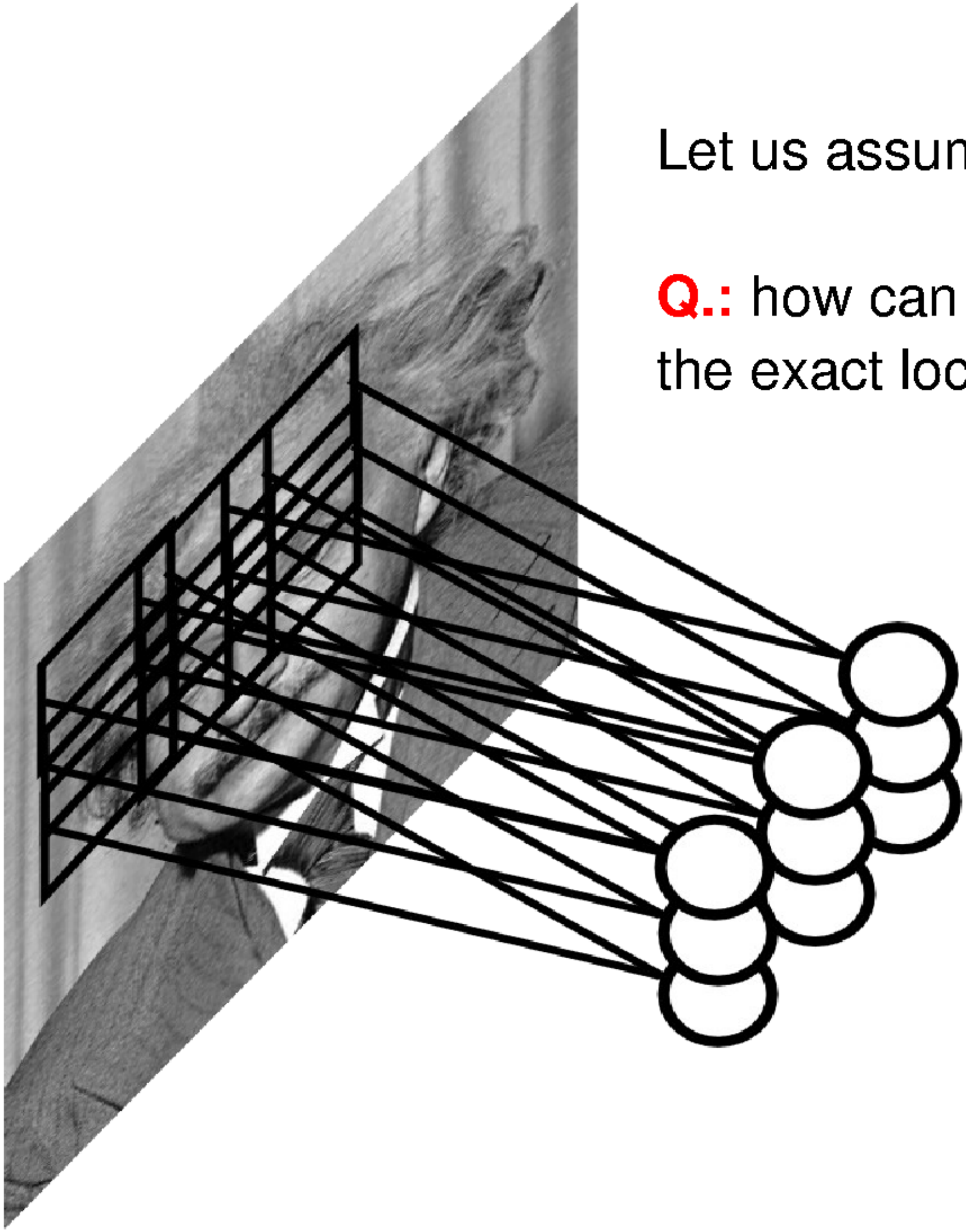
Solution:

- 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

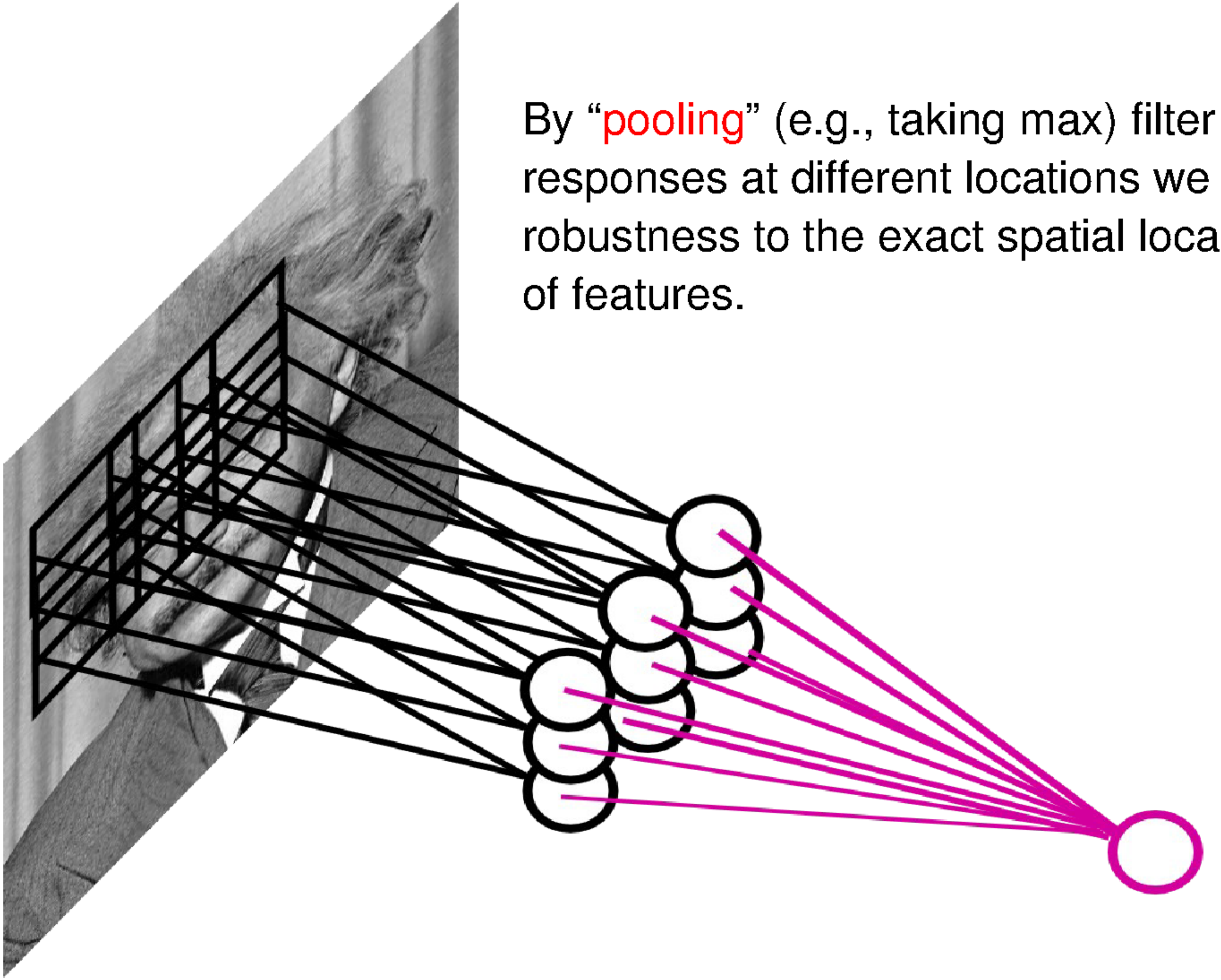
Pooling Layer



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.

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

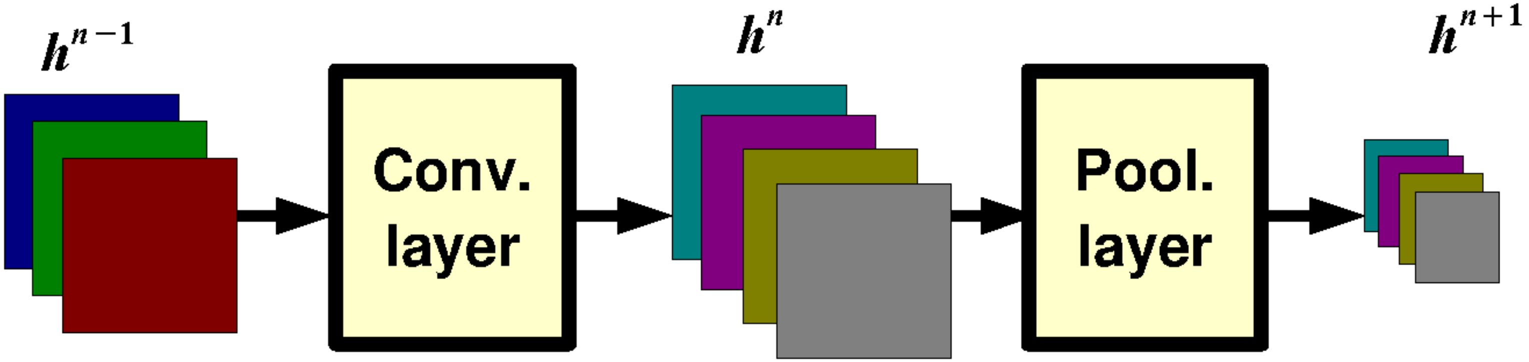
Question: What is the size of the output? What's the computational cost?

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)

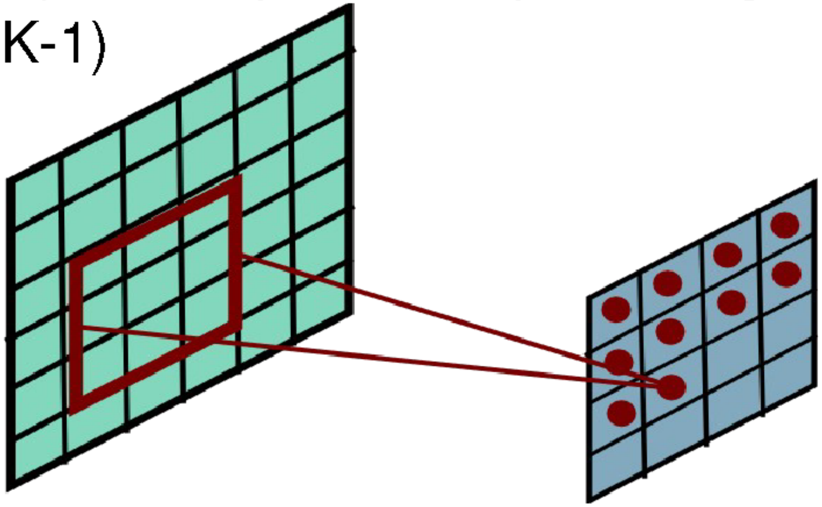
Question: How should I set the size of the pools?

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).

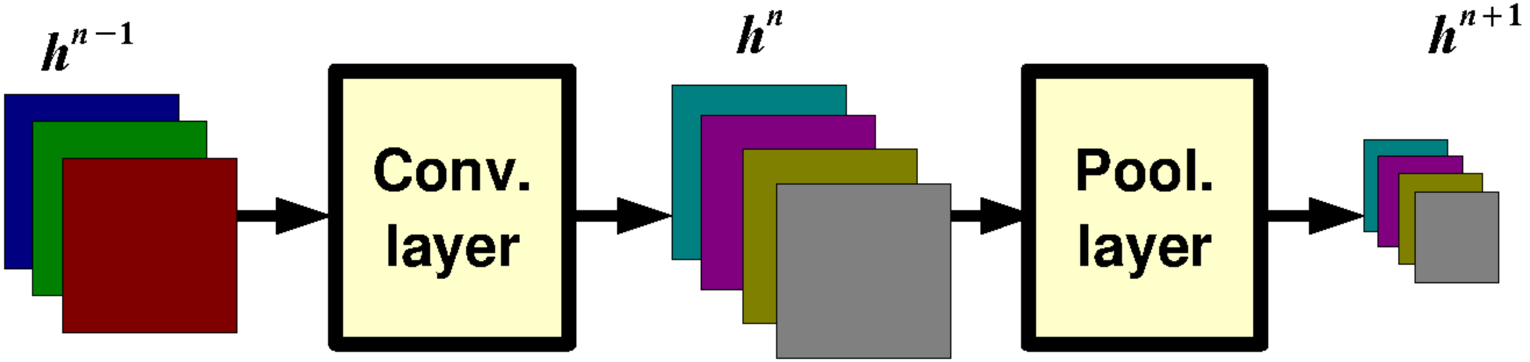
Pooling Layer: Receptive Field Size



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)

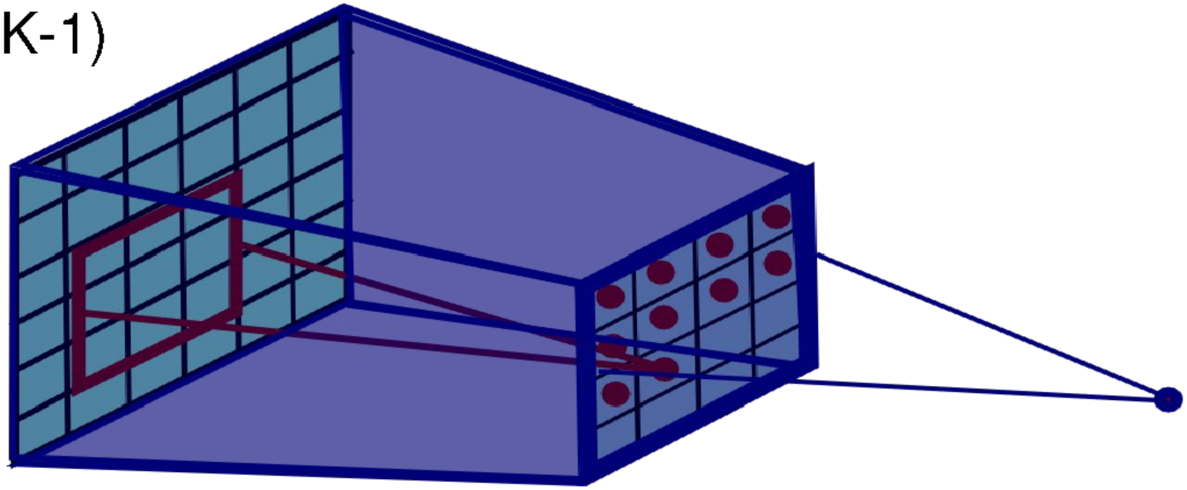


Pooling Layer: Receptive Field Size



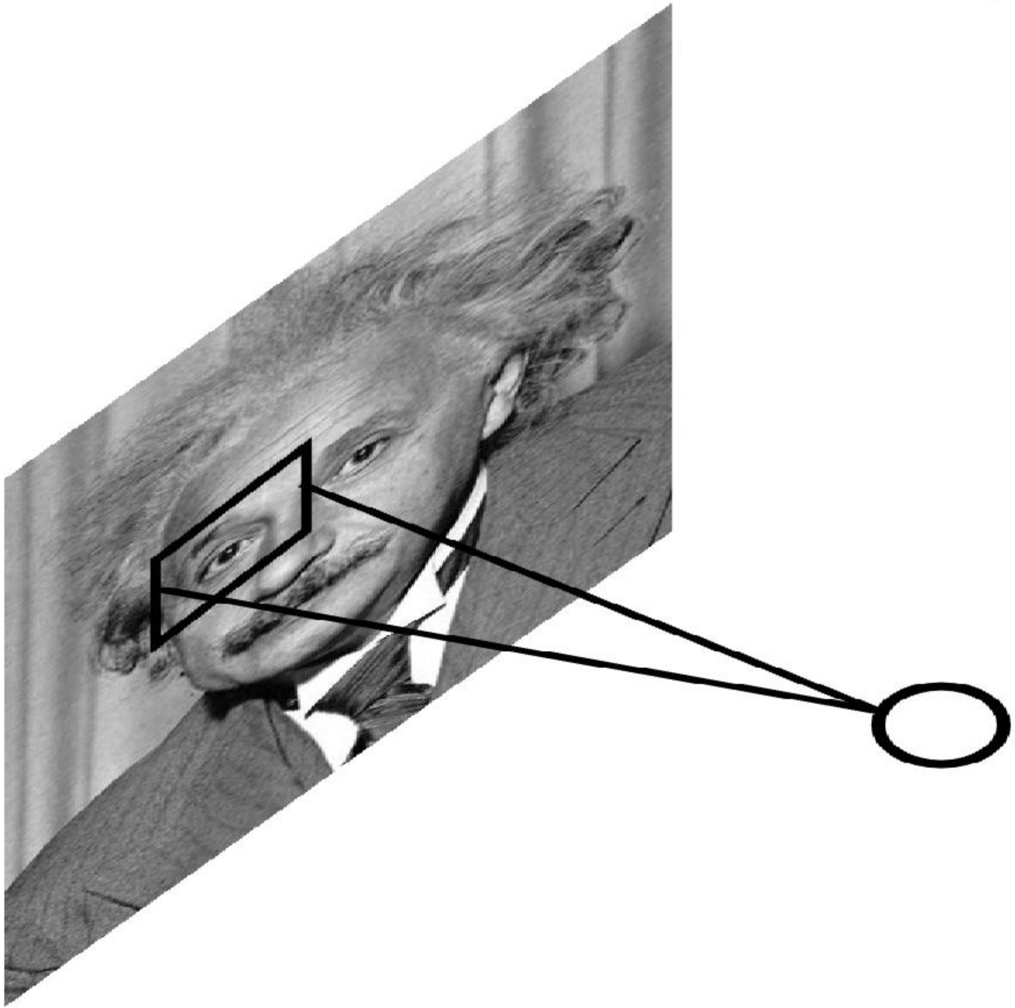
h^{n-1} → Conv. layer → h^{n} → Pool. layer → h^{n+1}

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)



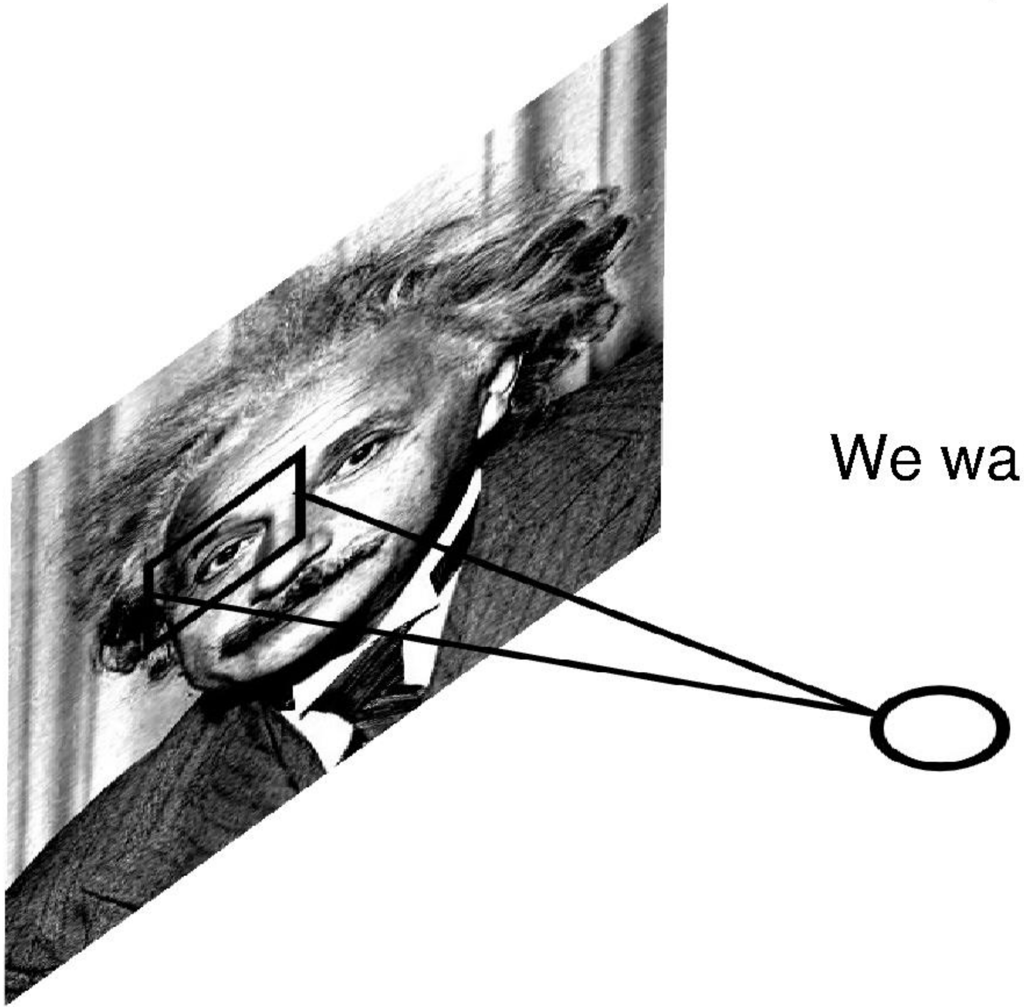
Local Contrast Normalization

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



Local Contrast Normalization

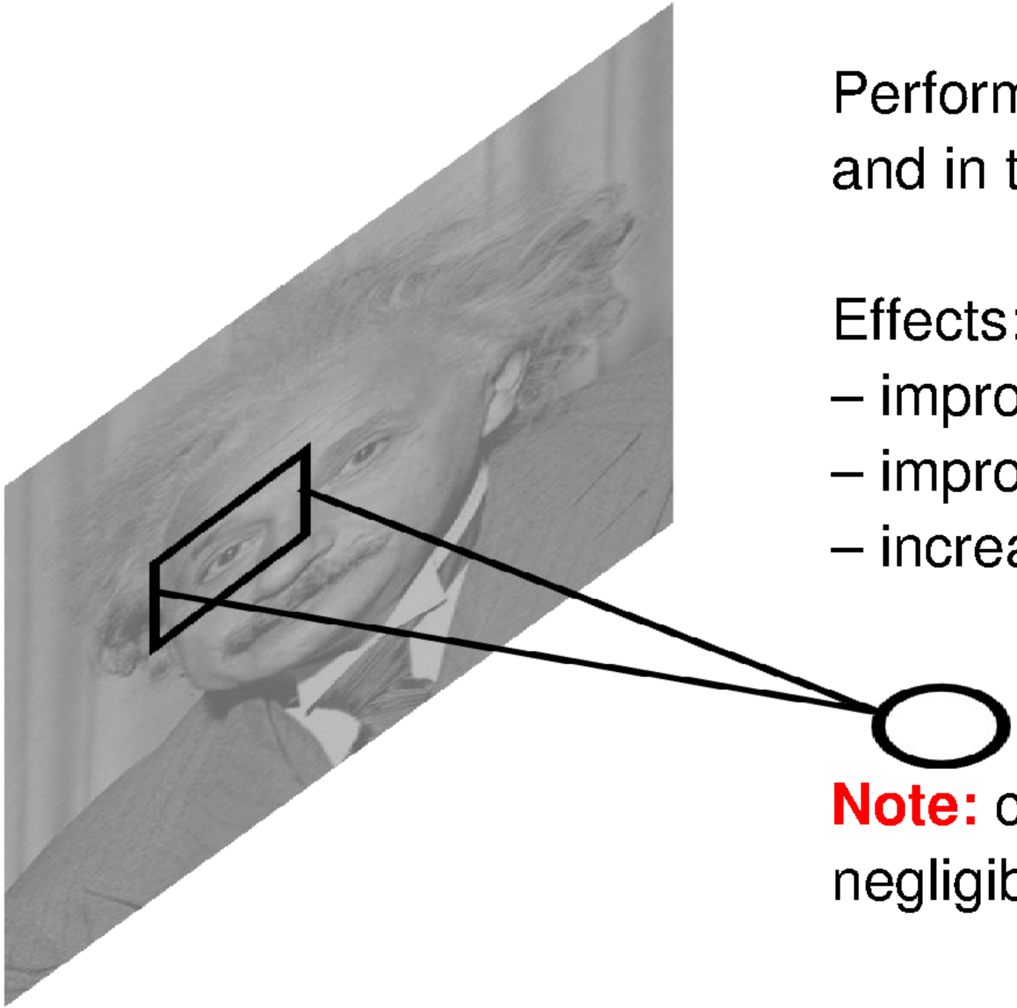
$$h^{i+1}(x,y) = \frac{h^i(x,y) - m^i(N(x,y))}{\sigma^i(N(x,y))}$$



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

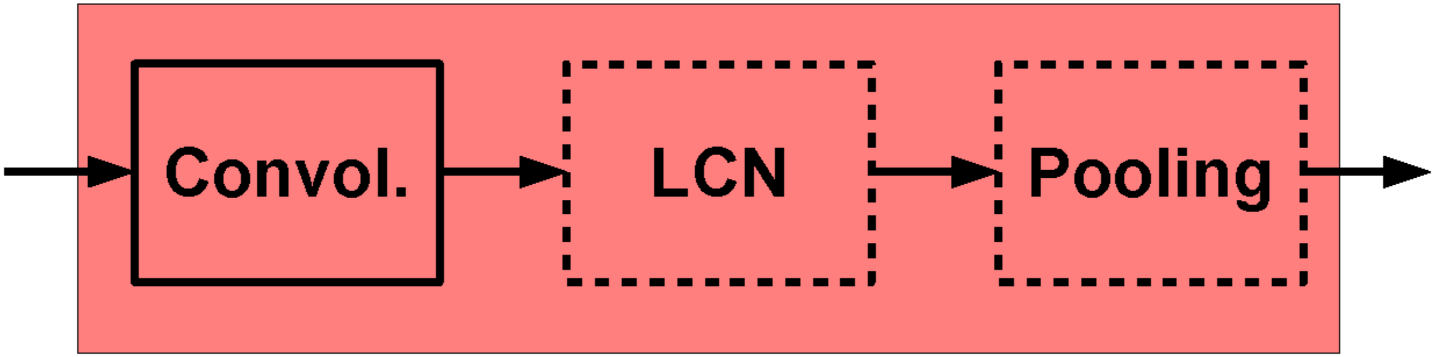
– improves optimization

– increases sparsity

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

ConvNets: Typical Stage

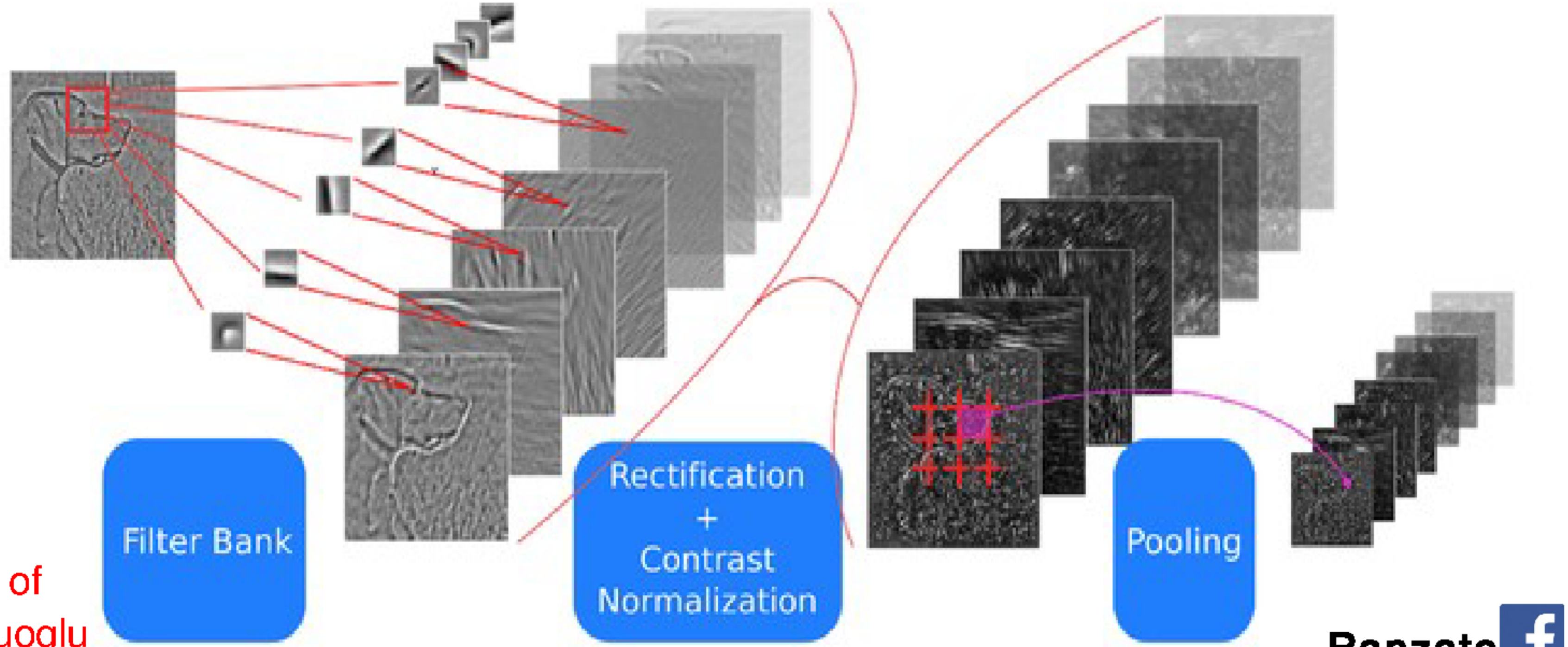
One stage (zoom)



Convol.

LCN

Pooling

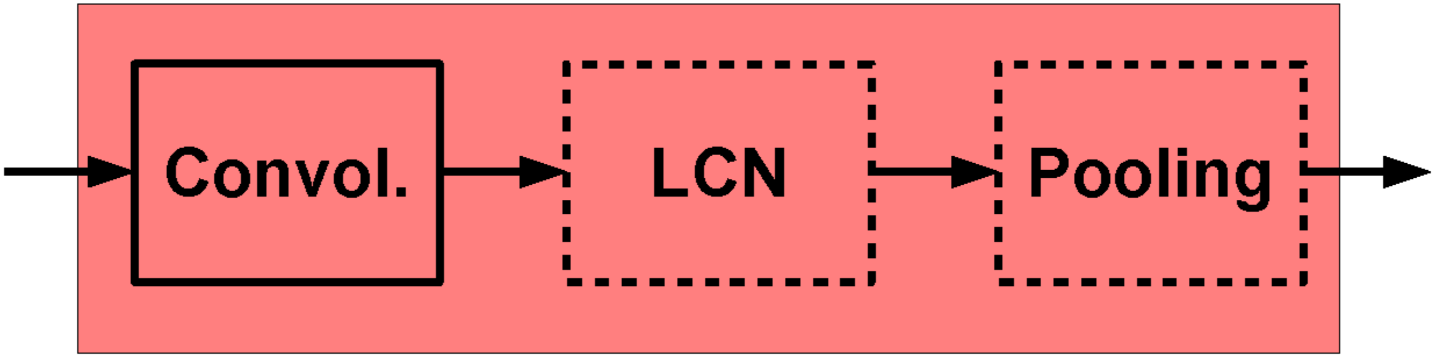


courtesy of
K. Kavukcuoglu

Ranzato

ConvNets: Typical Stage

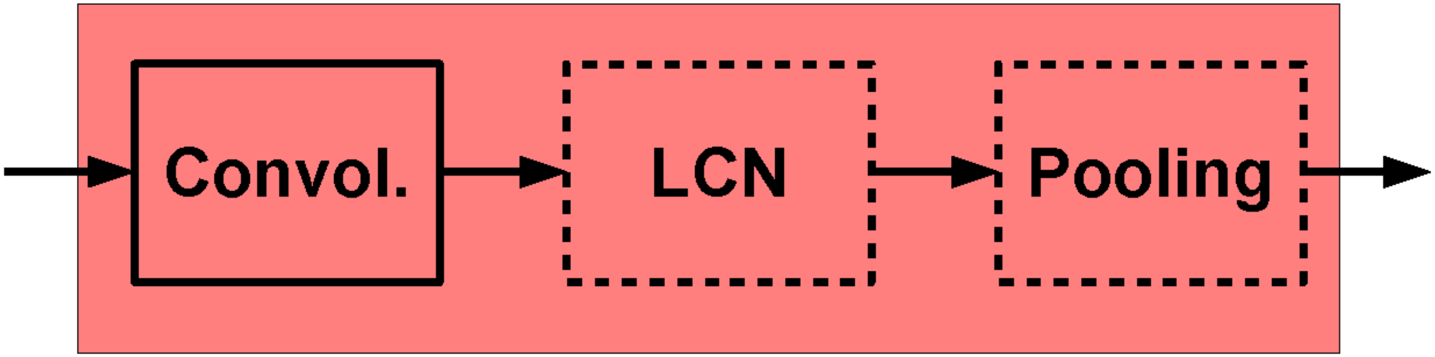
One stage (zoom)



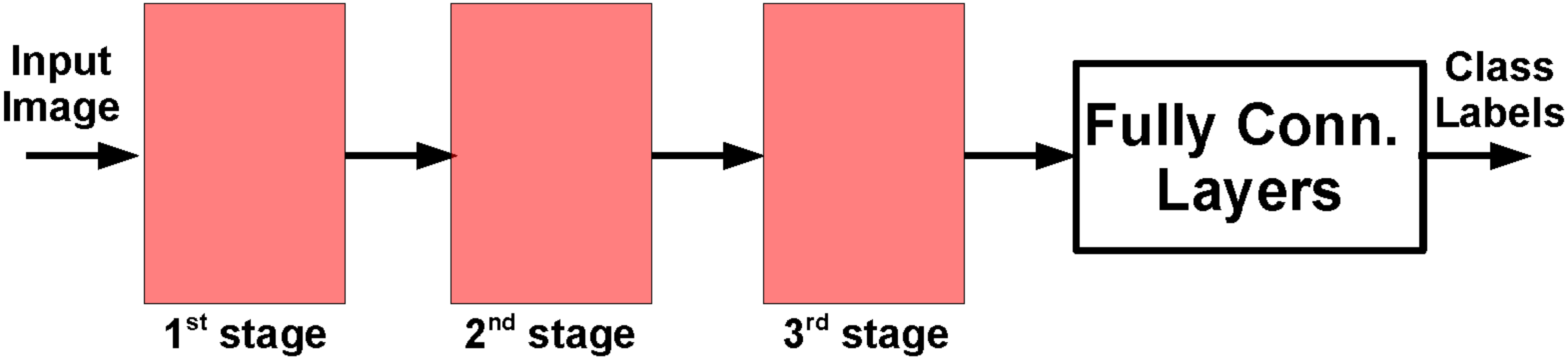
Conceptually similar to: SIFT, HoG, etc.

ConvNets: Typical Architecture

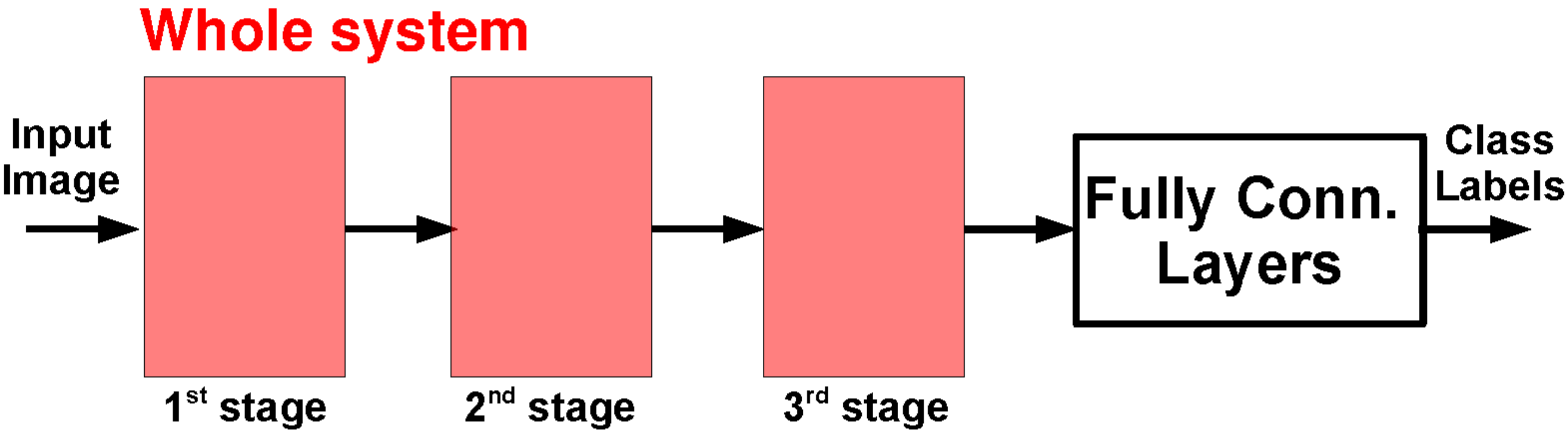
One stage (zoom)



Whole system



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

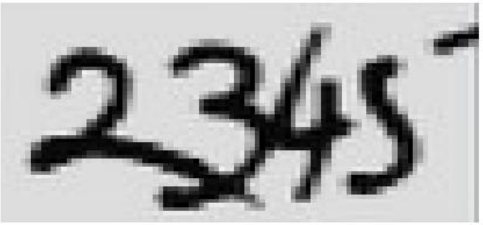
- **F-PROP**
- **B-PROP**
- **PARAMETER UPDATE**

Outline

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

CONV NETS: EXAMPLES

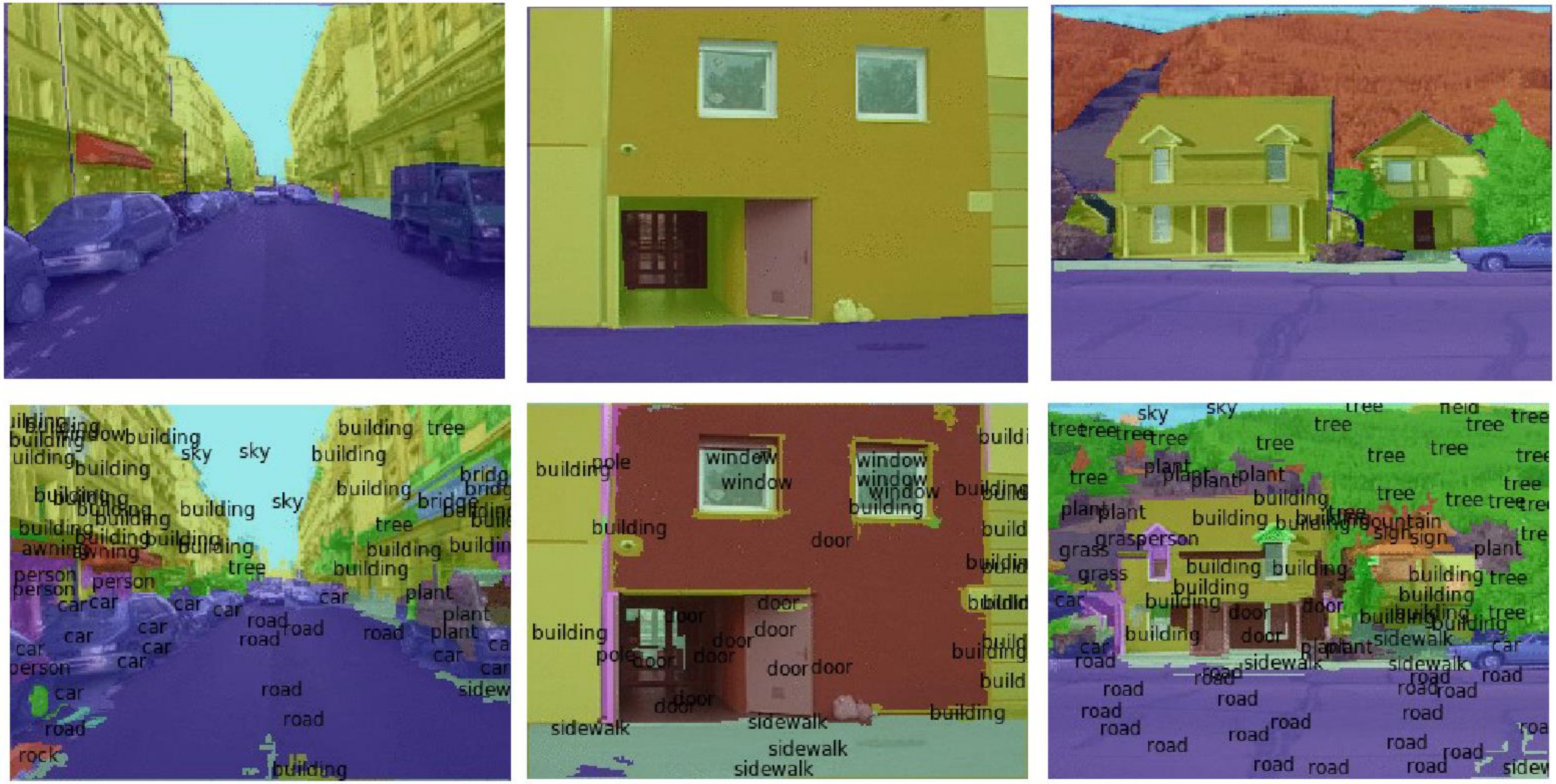
- **OCR / House number & Traffic sign classification**



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

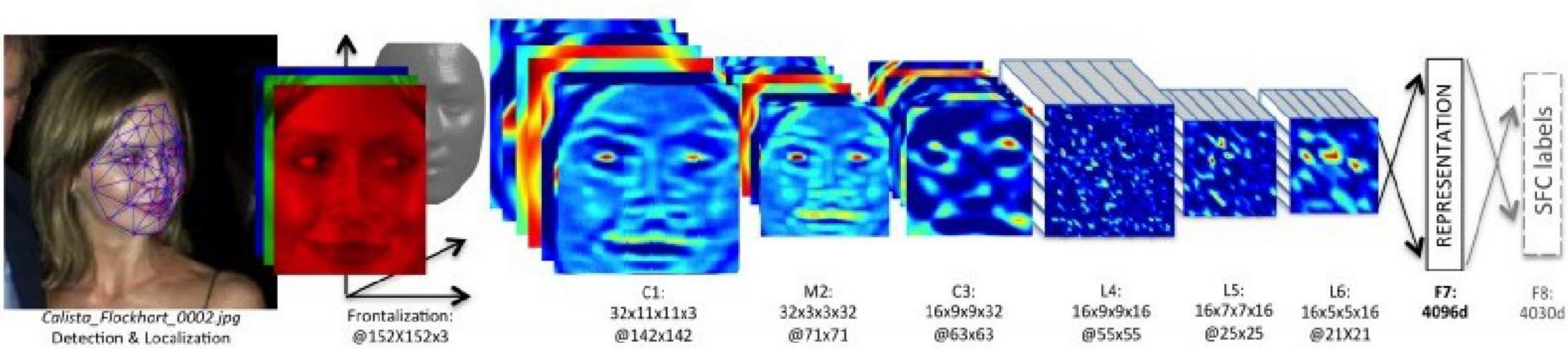
- **Scene Parsing**



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

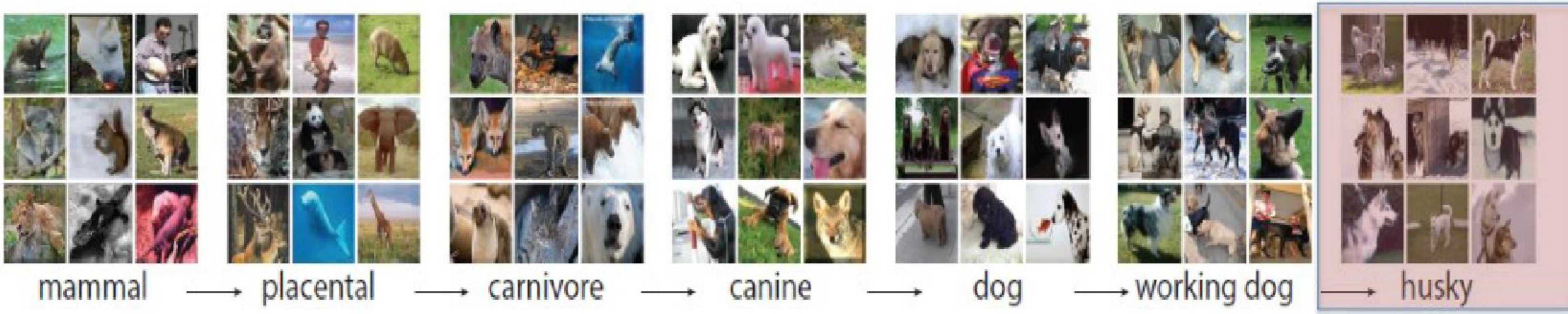
CONV NETS: EXAMPLES

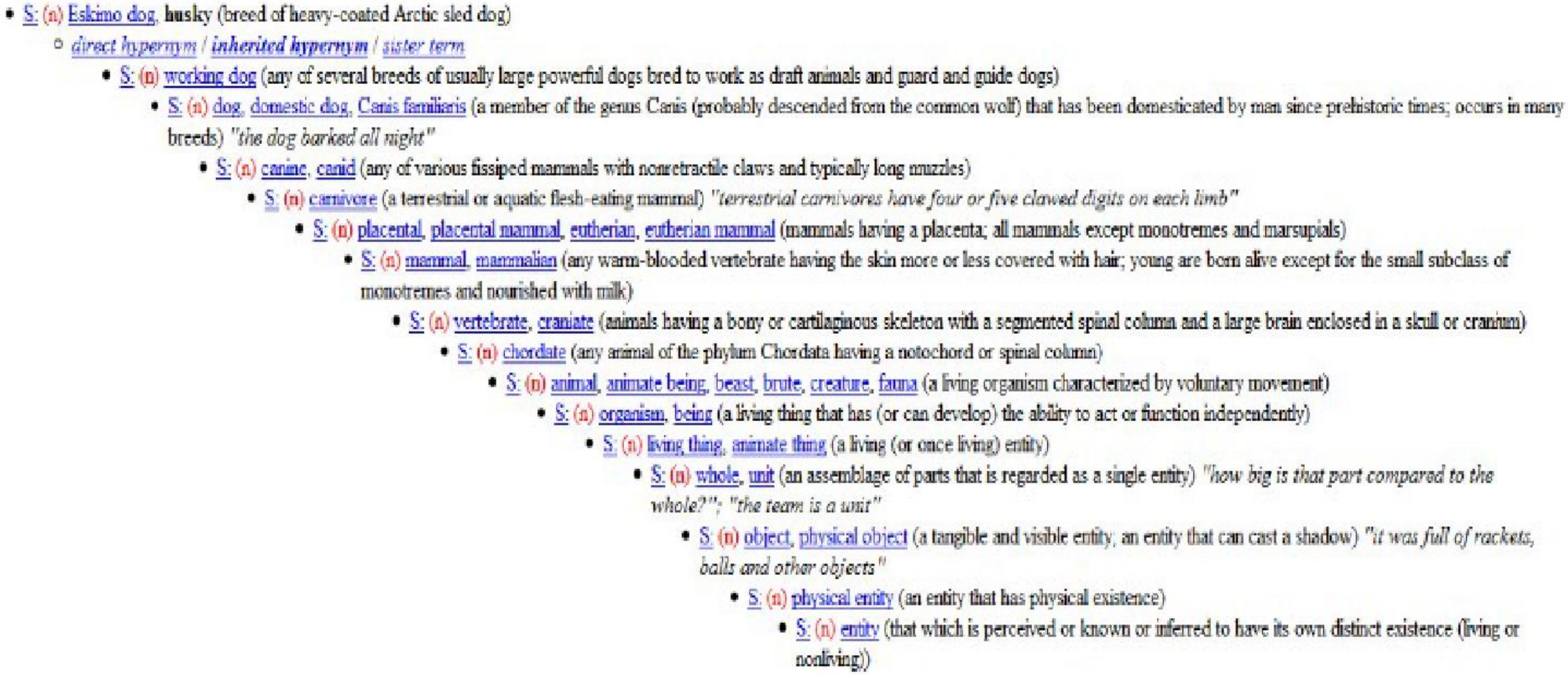
- **Face Verification & Identification**



Taigman et al. "DeepFace..." CVPR 2014

Dataset: ImageNet 2012





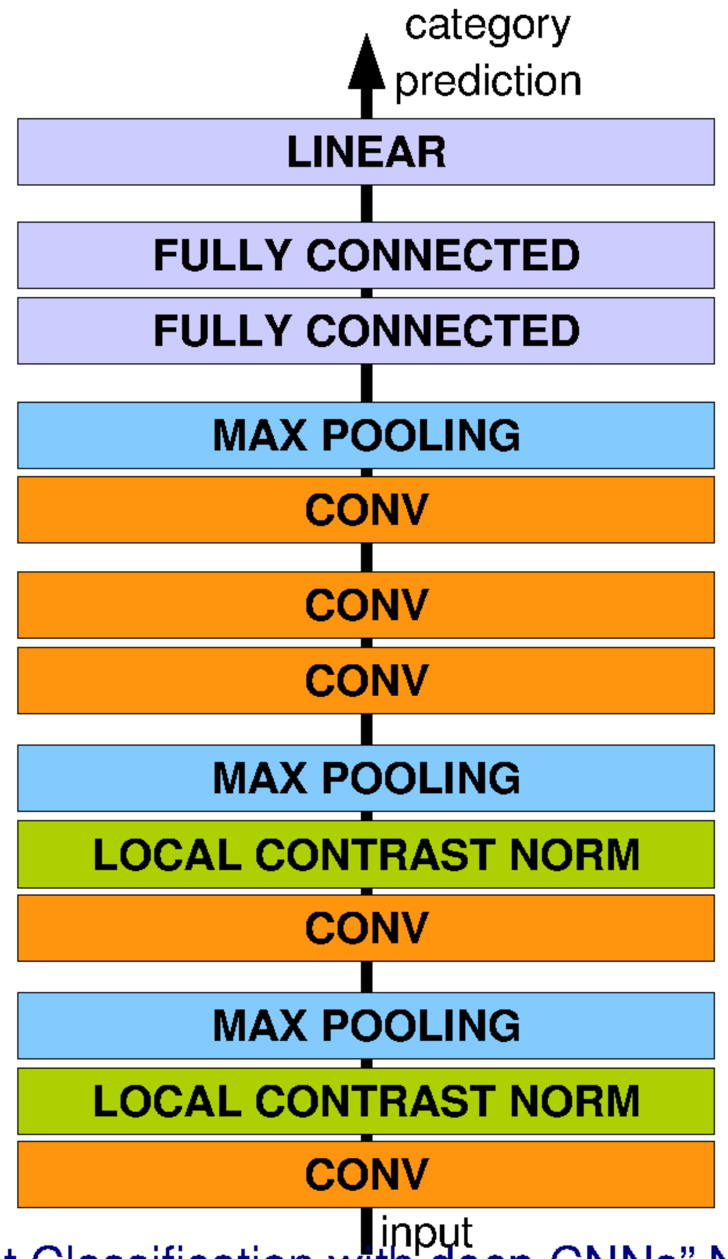
- S: (n) Eskimo dog, **husky** (breed of heavy-coated Arctic sled dog)
  - *direct hypernym* / ***inherited hypernym*** / *sister term*
    - S: (n) working dog (any of several breeds of usually large powerful dogs bred to work as draft animals and guard and guide dogs)
      - S: (n) dog, domestic dog, 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

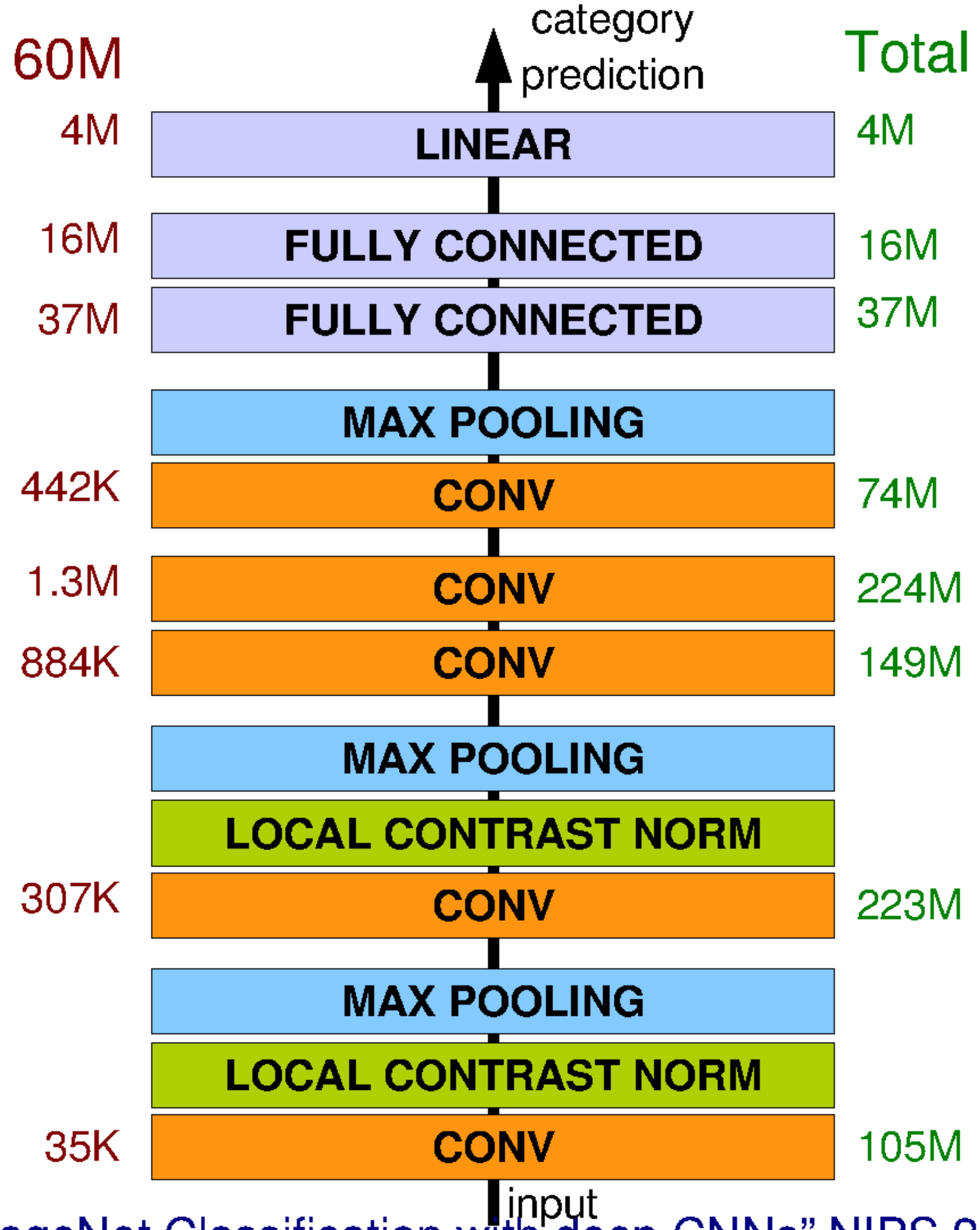


Krizhevsky et al. “ImageNet Classification with deep CNNs” NIPS 2012

Architecture for Classification

Total nr. params: 60M

Total nr. flops: 832M



category prediction

4M	LINEAR	4M
16M	FULLY CONNECTED	16M
37M	FULLY CONNECTED	37M
	MAX POOLING	
442K	CONV	74M
1.3M	CONV	224M
884K	CONV	149M
	MAX POOLING	
	LOCAL CONTRAST NORM	
307K	CONV	223M
	MAX POOLING	
	LOCAL CONTRAST NORM	
35K	CONV	105M

input

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

Optimization

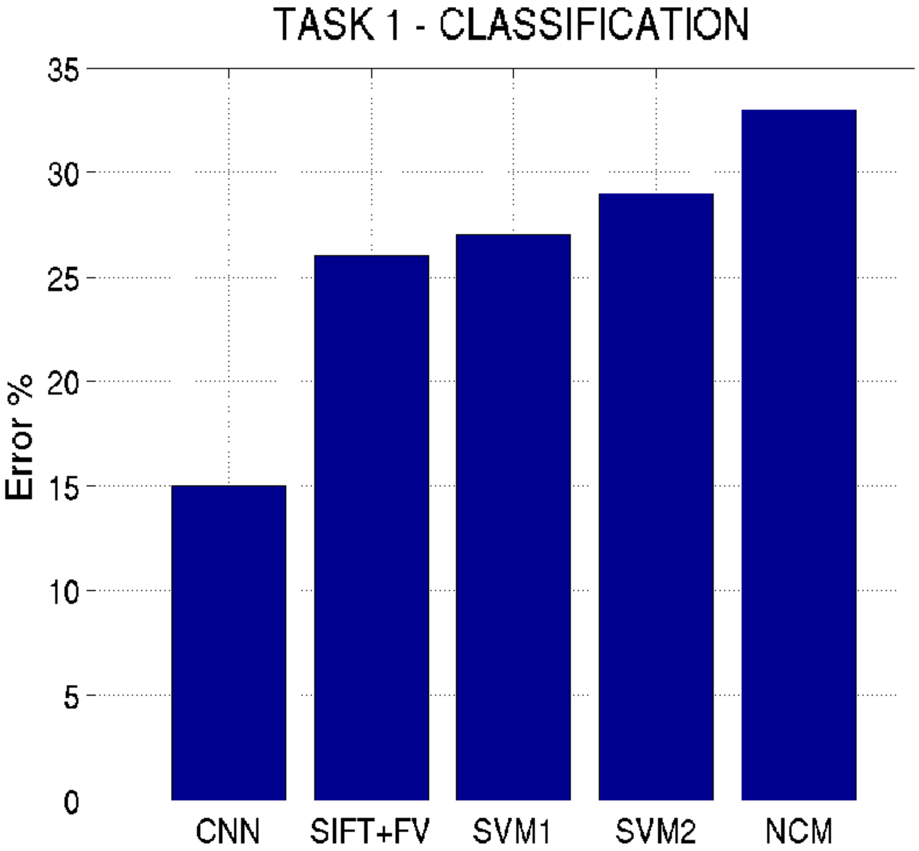
SGD with momentum:

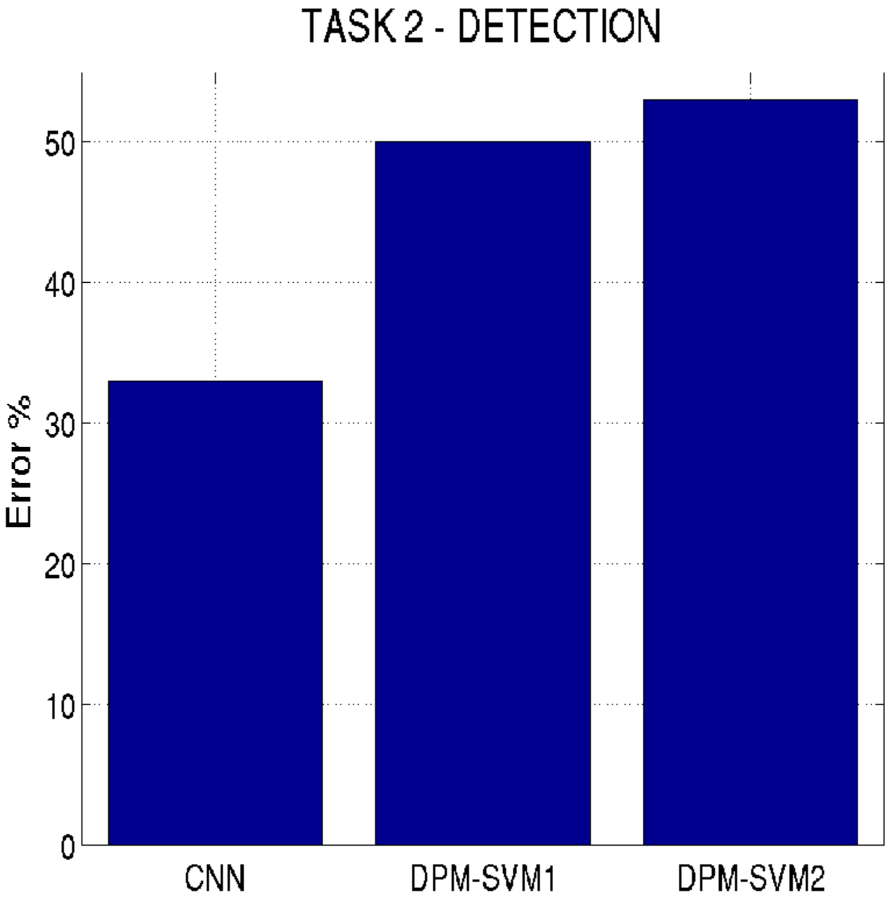
- Learning rate = 0.01
- Momentum = 0.9

Improving generalization by:

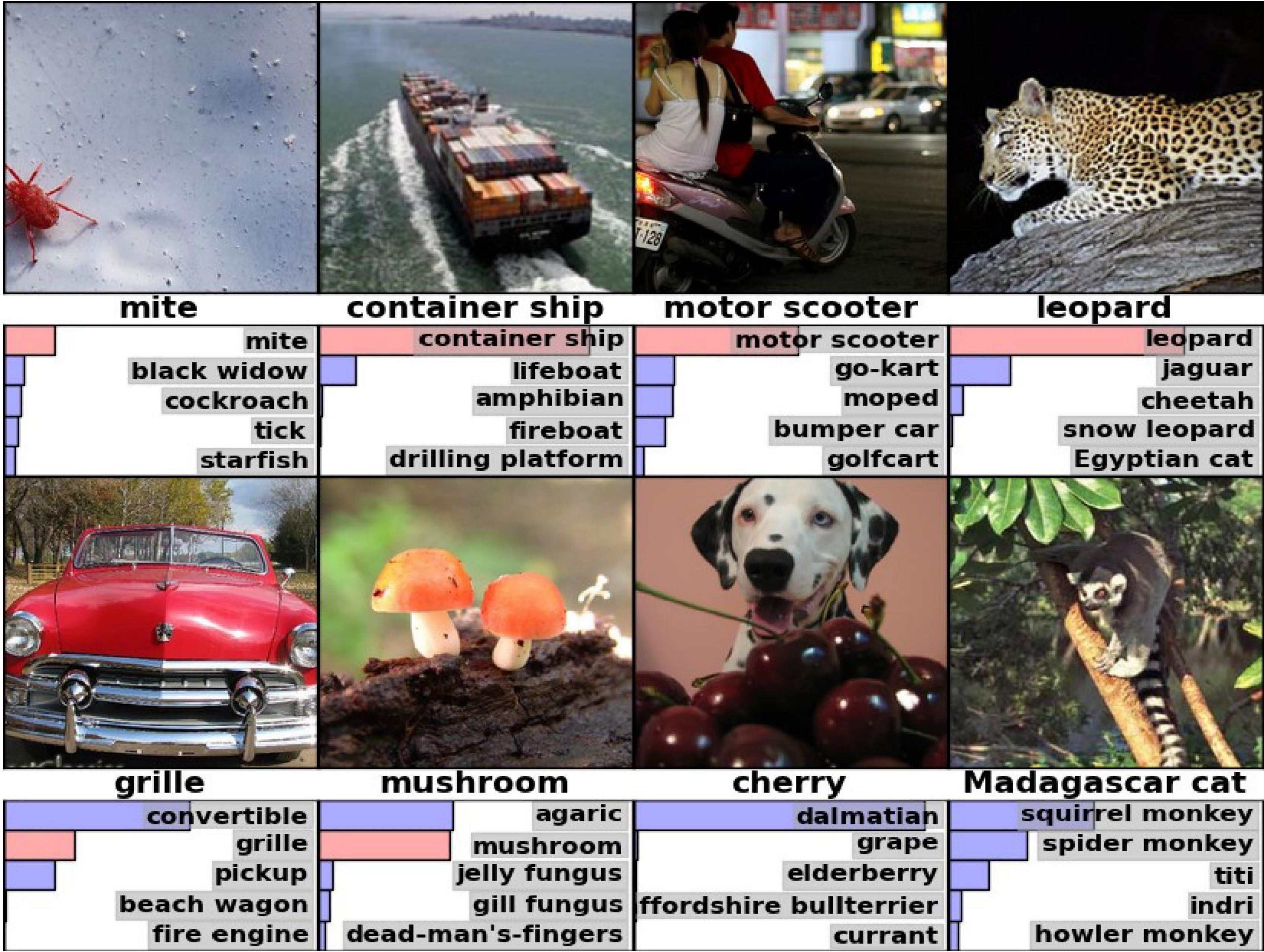
- Weight sharing (convolution)
- Input distortions
- Dropout = 0.5
- Weight decay = 0.0005

Results: ILSVRC 2012



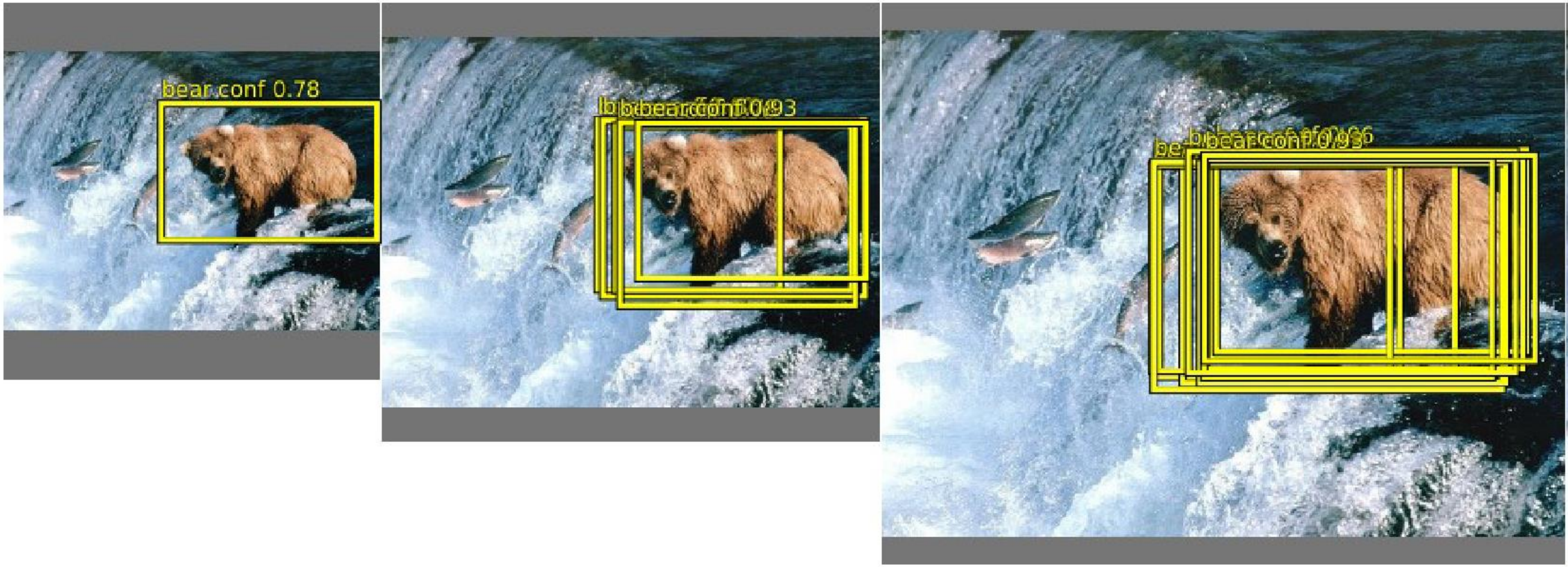


Krizhevsky et al. “ImageNet Classification with deep CNNs” NIPS 2012


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

CONV NETS: EXAMPLES

- **Object detection**



Sermanet et al. “OverFeat: Integrated recognition, localization, ...” arxiv 2013
Girshick et al. “Rich feature hierarchies for accurate object detection...” arxiv 2013
Szegedy et al. “DNN for object detection” NIPS 2013

Ranzato