

Deep Learning Neural Net Basics

Computer Vision

James Hays

Outline

- Neural Networks
- *Convolutional* Neural Networks
- Variants
 - Detection
 - Segmentation
 - Siamese Networks
- Visualization of Deep Networks

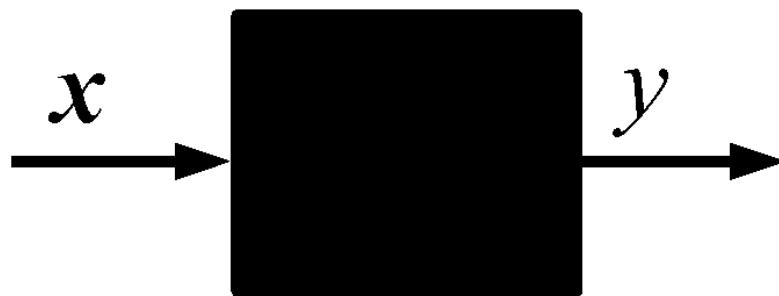
Supervised Learning

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

\mathbf{x}^i i-th input training example

y^i i-th target label

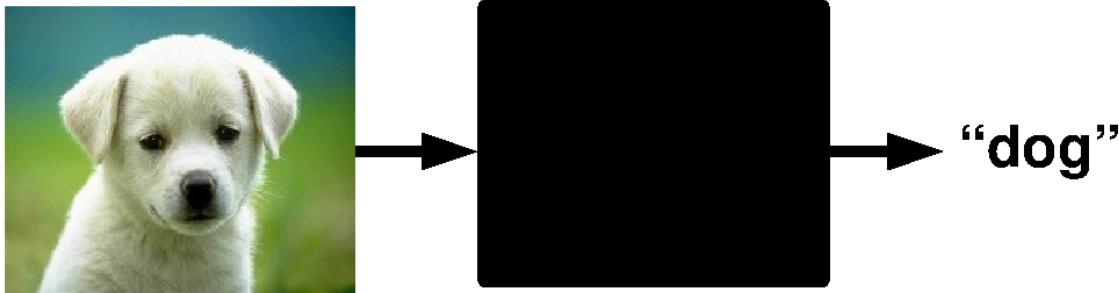
P number of training examples



Goal: predict the target label of unseen inputs.

Supervised Learning: Examples

Classification



classification

Denoising



regression

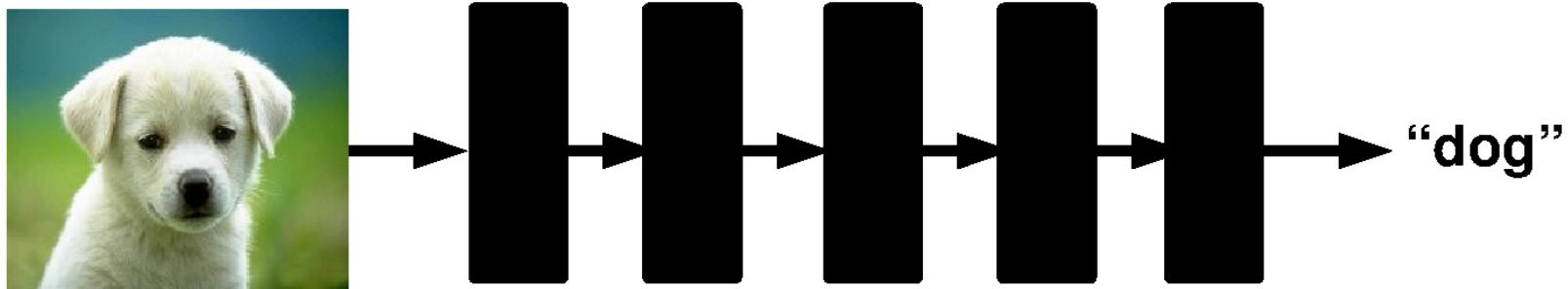
OCR



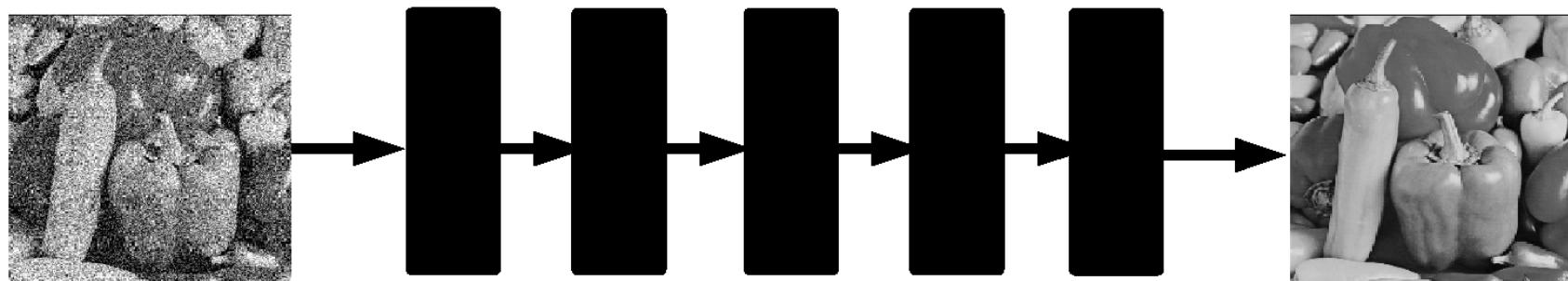
structured
prediction

Supervised Deep Learning

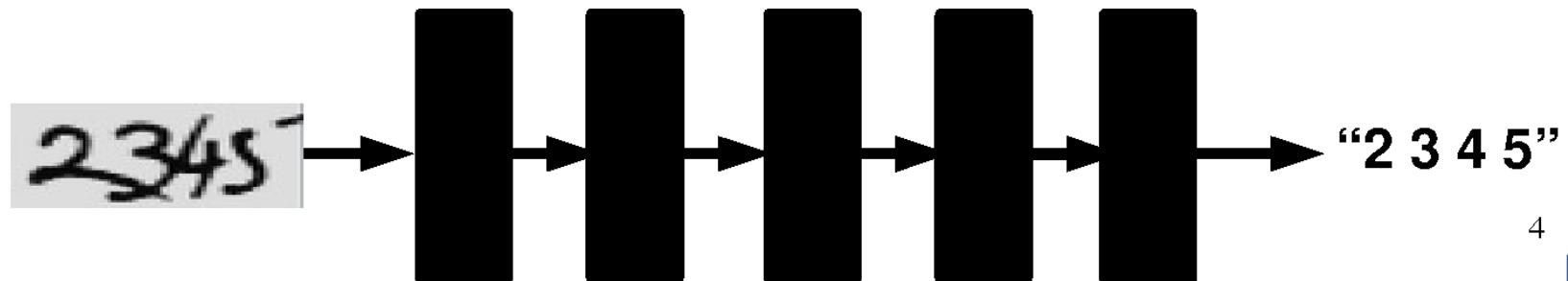
Classification



Denoising



OCR



Outline

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

Neural Networks

Assumptions (for the next few slides):

- The input image is vectorized (disregard the spatial layout of pixels)
- The target label is discrete (classification)

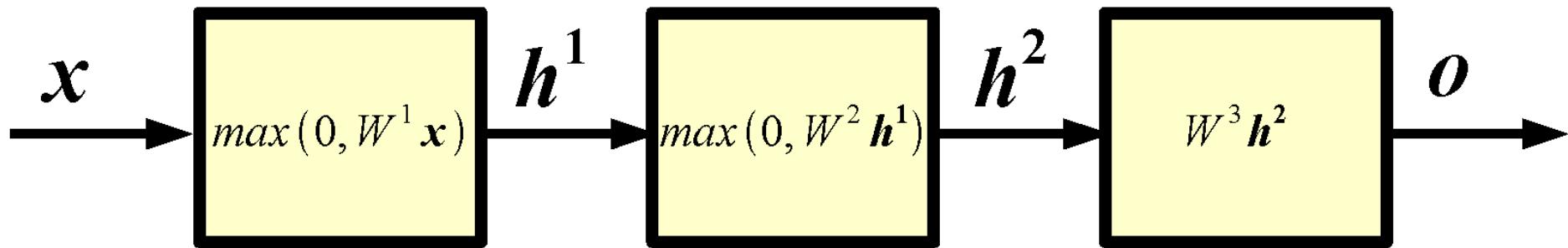
Question: what class of functions shall we consider to map the input into the output?

Answer: composition of simpler functions.

Follow-up questions: Why not a linear combination? What are the “simpler” functions? What is the interpretation?

Answer: later...

Neural Networks: example



x input

h^1 1-st layer hidden units

h^2 2-nd layer hidden units

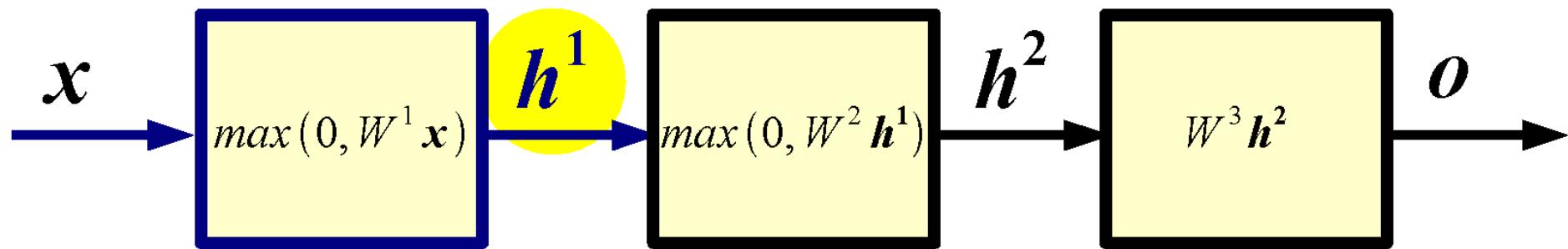
o output

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



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

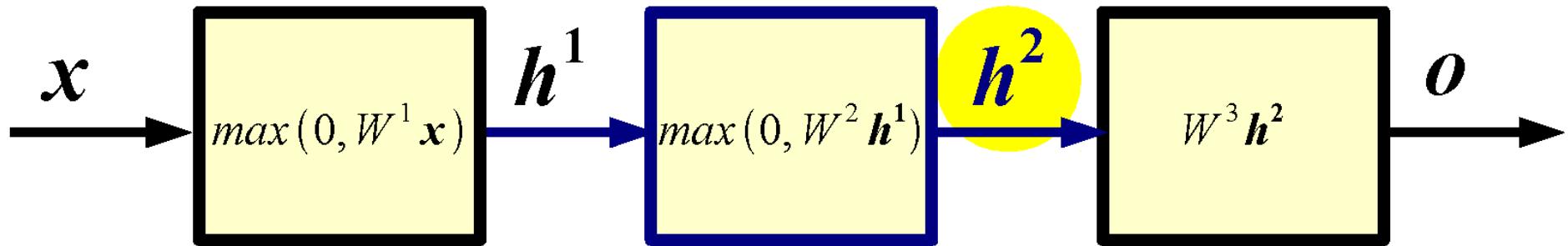
$$h^1 = \max(0, W^1 x + b^1)$$

W^1 1-st layer weight matrix or weights

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



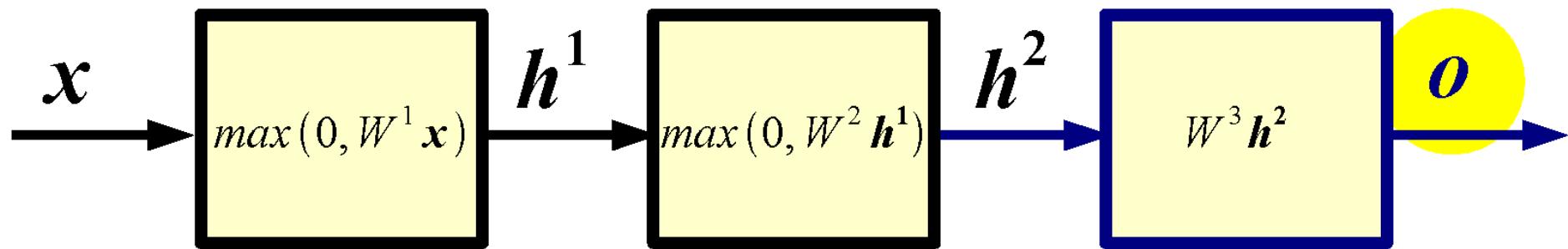
$$h^1 \in R^{N_1} \quad W^2 \in R^{N_2 \times N_1} \quad b^2 \in R^{N_2} \quad h^2 \in R^{N_2}$$

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

W^2 2-nd layer weight matrix or weights

b^2 2-nd layer biases

Forward Propagation



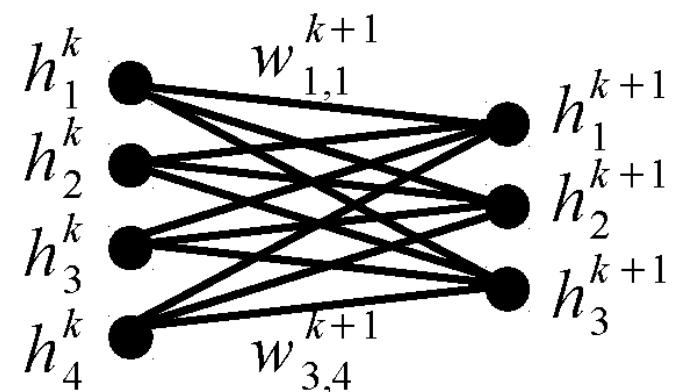
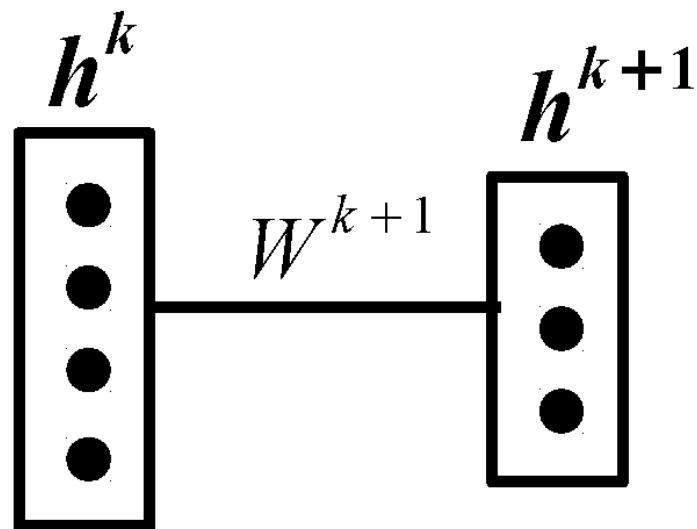
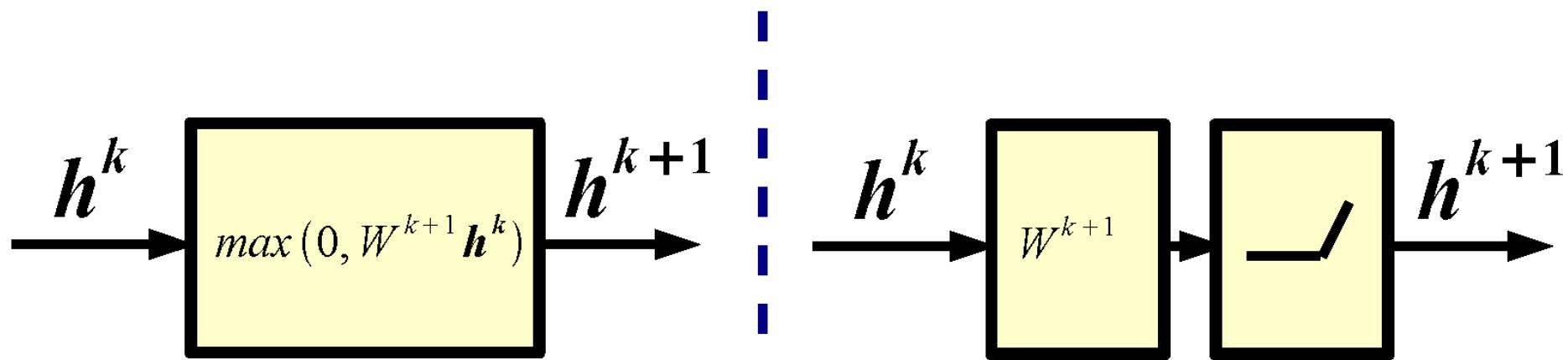
$$h^2 \in R^{N_2} \quad W^3 \in R^{N_3 \times N_2} \quad b^3 \in R^{N_3} \quad o \in R^{N_3}$$

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

W^3 3-rd layer weight matrix or weights

b^3 3-rd layer biases

Alternative Graphical Representation



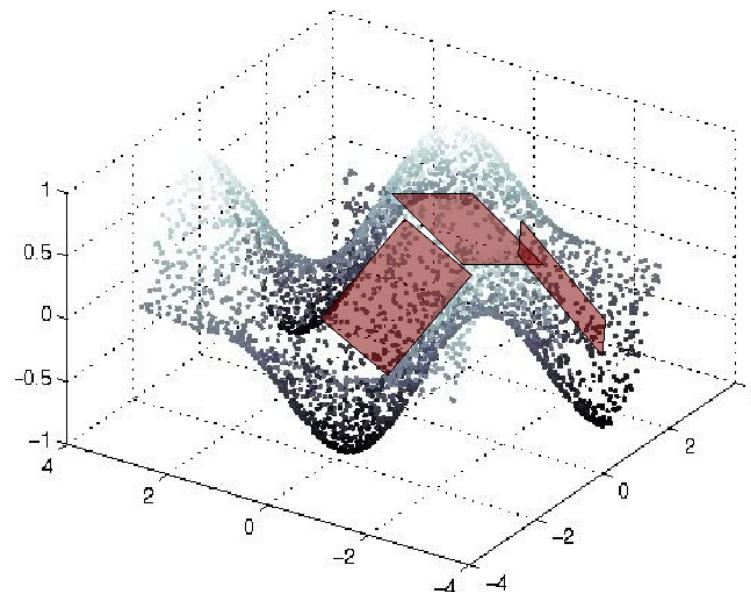
Interpretation

Question: Why can't the mapping between layers be linear?

Answer: Because composition of linear functions is a linear function. Neural network would reduce to (1 layer) logistic regression.

Question: What do ReLU layers accomplish?

Answer: Piece-wise linear tiling: mapping is locally linear.

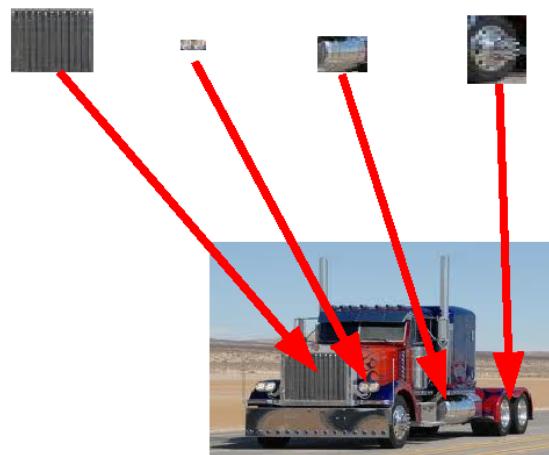


Interpretation

Question: Why do we need many layers?

Answer: When input has hierarchical structure, the use of a hierarchical architecture is potentially more efficient because intermediate computations can be re-used. DL architectures are efficient also because they use **distributed representations** which are shared across classes.

[0 0 1 0 0 0 0 1 0 0 1 1 0 0 1 0 ...] truck feature

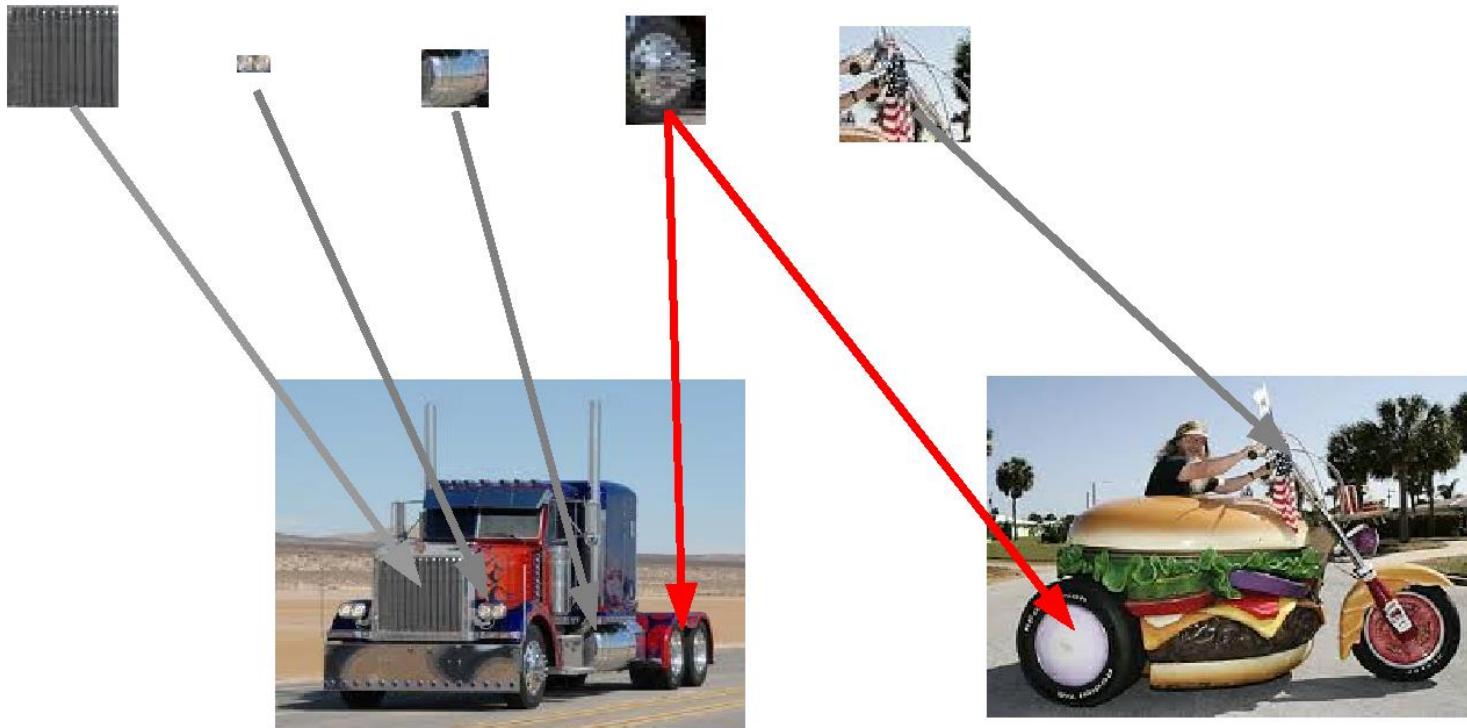


Exponentially more efficient than a 1-of-N representation (a la k-means)

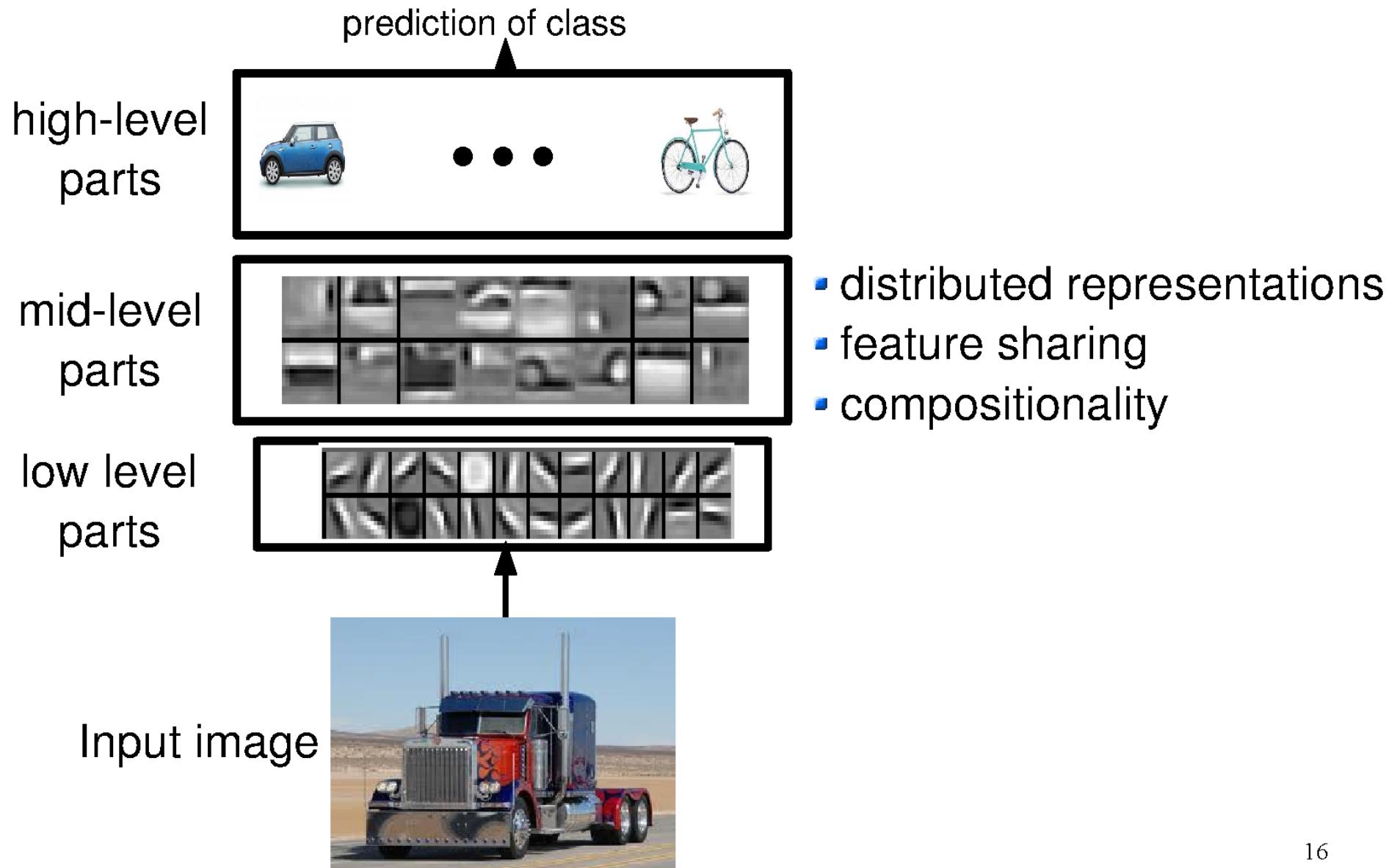
Interpretation

[1 1 0 0 0 1 0 1 0 0 0 0 1 1 0 1 ...] motorbike

[0 0 1 0 0 0 0 1 0 0 1 1 0 0 1 0 ...] truck



Interpretation



Interpretation

Question: What does a hidden unit do?

Answer: It can be thought of as a classifier or feature detector.

Question: How many layers? How many hidden units?

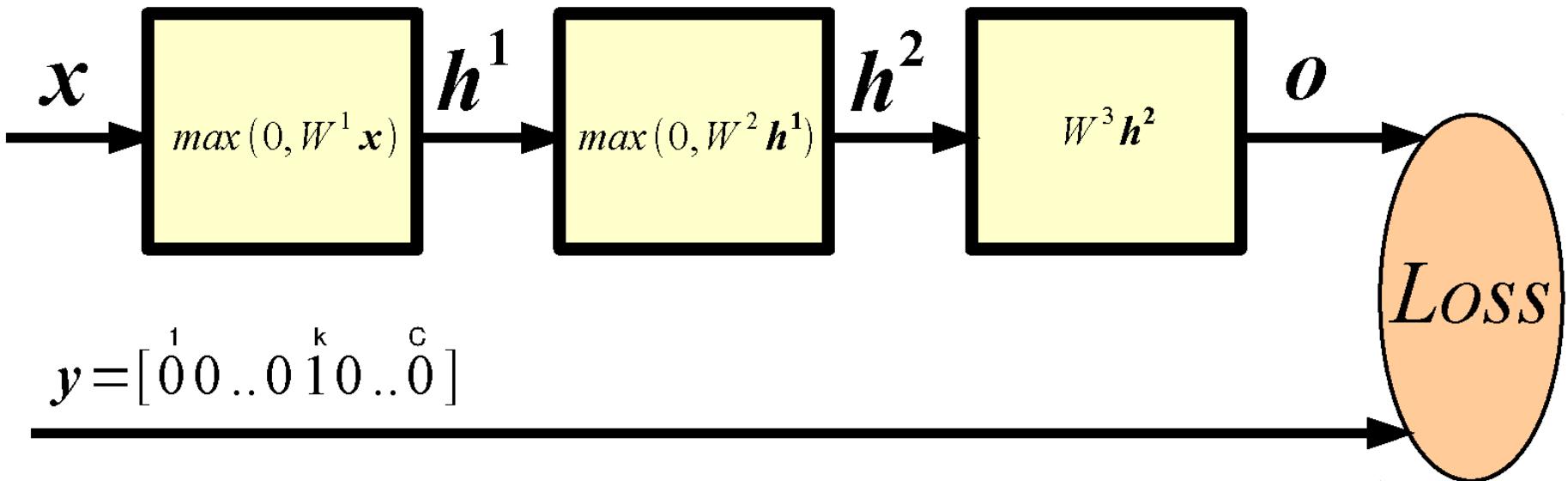
Answer: Cross-validation or hyper-parameter search methods are the answer. In general, the wider and the deeper the network the more complicated the mapping.

Question: How do I set the weight matrices?

Answer: Weight matrices and biases are learned.

First, we need to define a measure of quality of the current mapping.
Then, we need to define a procedure to adjust the parameters.

How Good is a Network?



Probability of class k given input (softmax):

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

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

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

Training

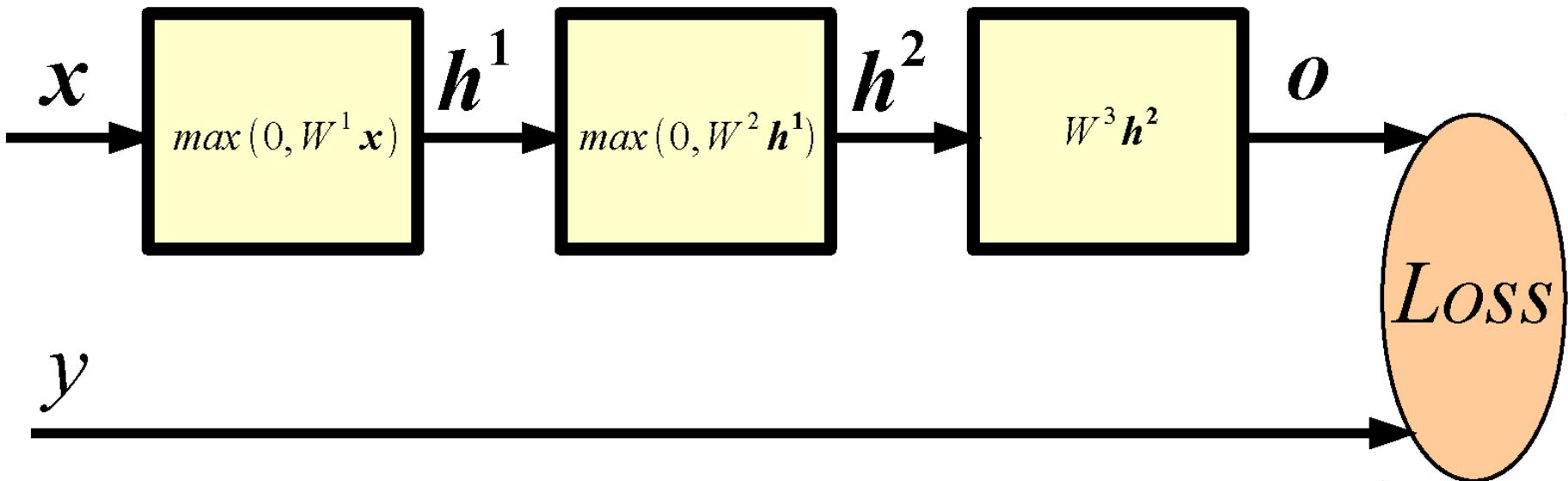
Learning consists of minimizing the loss (plus some regularization term) w.r.t. parameters over the whole training set.

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

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.

Key Idea: Wiggle To Decrease Loss



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

$$L(x, y; \theta)$$

$$L(x, y; \theta \setminus W_{i,j}^1, W_{i,j}^1 + \epsilon)$$

Then, update:

$$W_{i,j}^1 \leftarrow W_{i,j}^1 + \epsilon \operatorname{sgn}(L(x, y; \theta) - L(x, y; \theta \setminus W_{i,j}^1, W_{i,j}^1 + \epsilon))$$

Derivative w.r.t. Input of Softmax

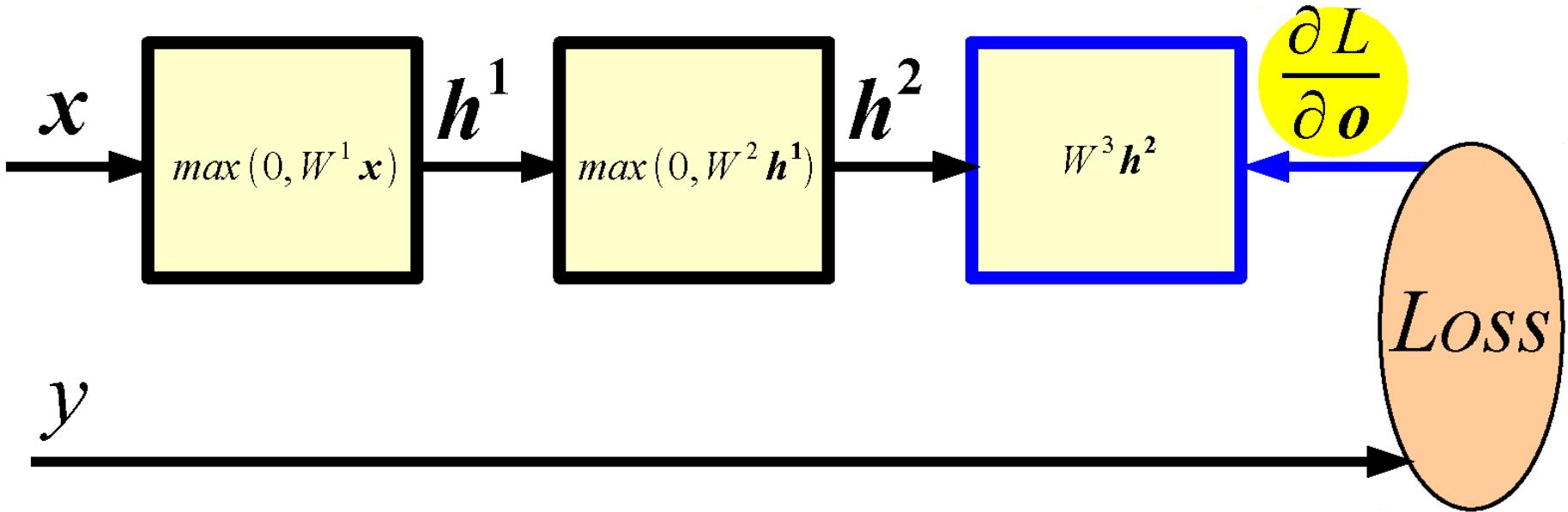
$$p(c_k=1|\mathbf{x}) = \frac{e^{o_k}}{\sum_j e^{o_j}}$$

$$L(\mathbf{x}, \mathbf{y}; \boldsymbol{\theta}) = -\sum_j y_j \log p(c_j|\mathbf{x}) \quad \mathbf{y} = [0^1 0^0 \dots 0^k 1^0 \dots 0^c]$$

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

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

Backward Propagation

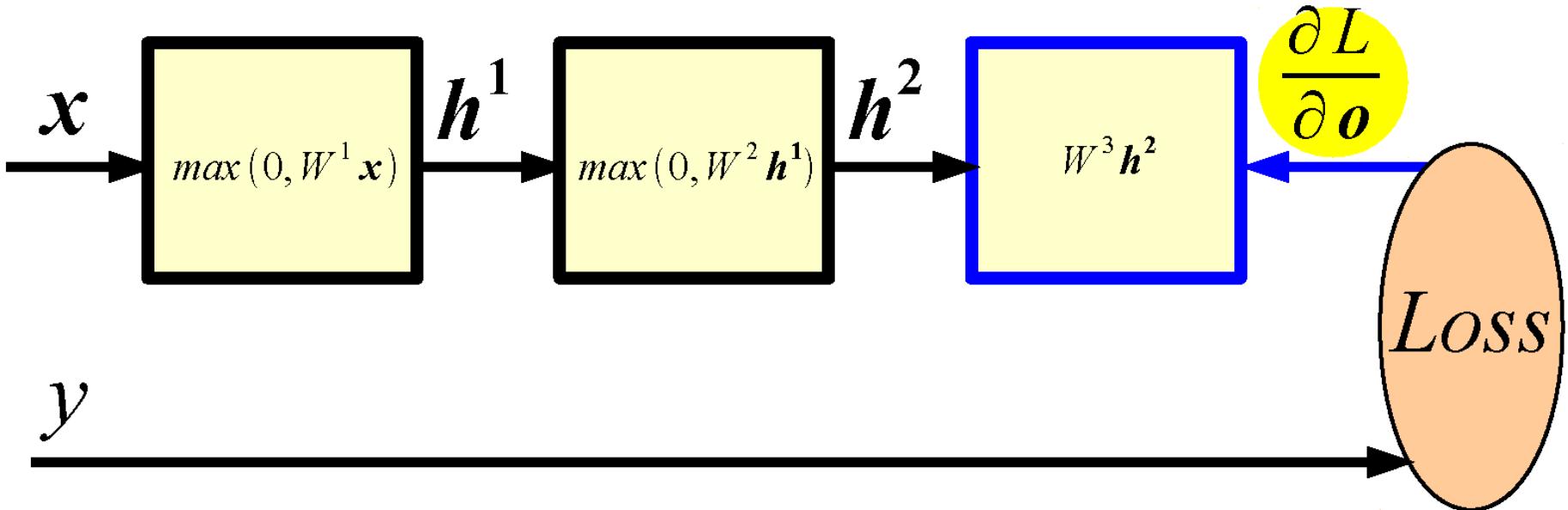


Given $\frac{\partial L}{\partial \mathbf{o}}$ and assuming we can easily compute the Jacobian of each module, we have:

$$\frac{\partial L}{\partial W^3} = \frac{\partial L}{\partial \mathbf{o}} \frac{\partial \mathbf{o}}{\partial W^3}$$

$$\frac{\partial L}{\partial h^2} = \frac{\partial L}{\partial \mathbf{o}} \frac{\partial \mathbf{o}}{\partial h^2}$$

Backward Propagation



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

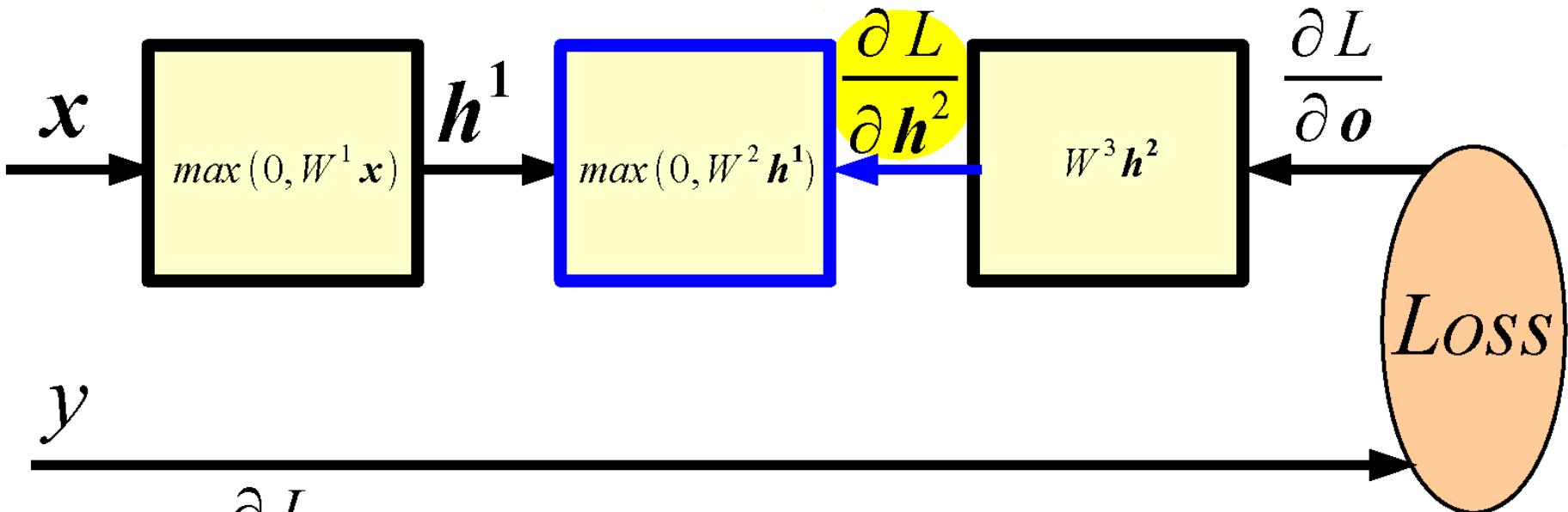
$$\frac{\partial L}{\partial W^3} = \frac{\partial L}{\partial \mathbf{o}} \frac{\partial \mathbf{o}}{\partial W^3}$$

$$\frac{\partial L}{\partial \mathbf{h}^2} = \frac{\partial L}{\partial \mathbf{o}} \frac{\partial \mathbf{o}}{\partial \mathbf{h}^2}$$

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

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

Backward Propagation

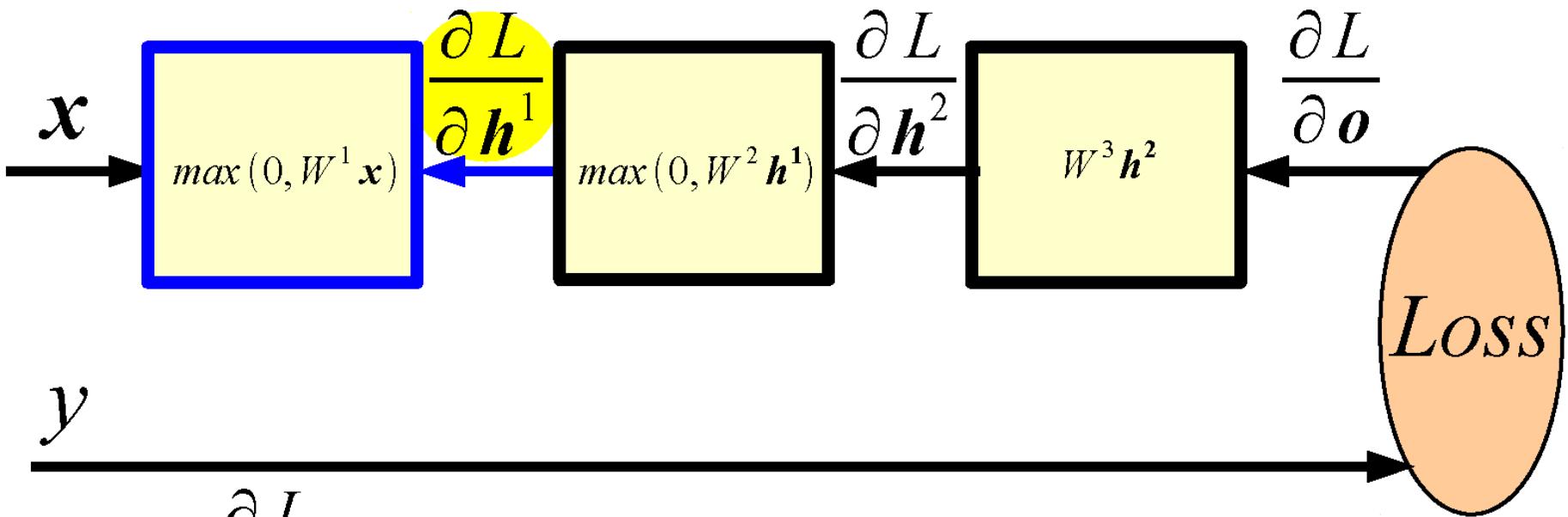


Given $\frac{\partial L}{\partial h^2}$ we can compute now:

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

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

Backward Propagation



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

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

Backward Propagation

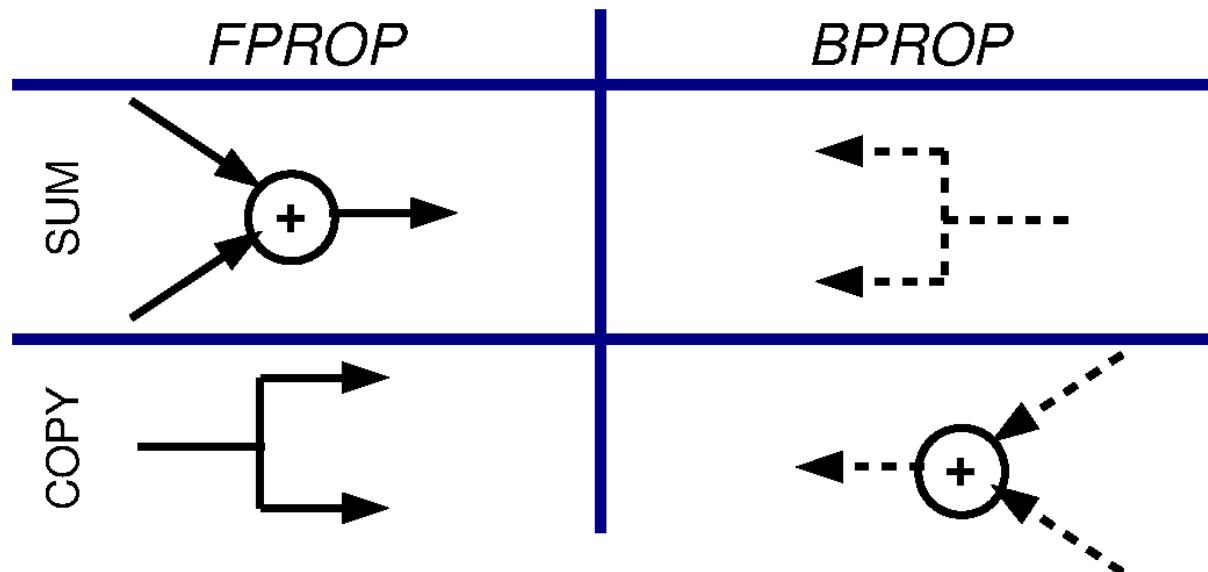
Question: Does BPROP work with ReLU layers only?

Answer: Nope, any a.e. differentiable transformation works.

Question: What's the computational cost of BPROP?

Answer: About twice FPROP (need to compute gradients w.r.t. input and parameters at every layer).

Note: FPROP and BPROP are dual of each other. E.g.,:



Optimization

Stochastic Gradient Descent (on mini-batches):

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \frac{\partial L}{\partial \boldsymbol{\theta}}, \eta \in (0, 1)$$

Stochastic Gradient Descent with Momentum:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \Delta$$

$$\Delta \leftarrow 0.9 \Delta + \frac{\partial L}{\partial \boldsymbol{\theta}}$$

Note: there are many other variants...

Outline

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

Hollow Face Illusion

- https://en.wikipedia.org/wiki/Hollow-Face_illusion



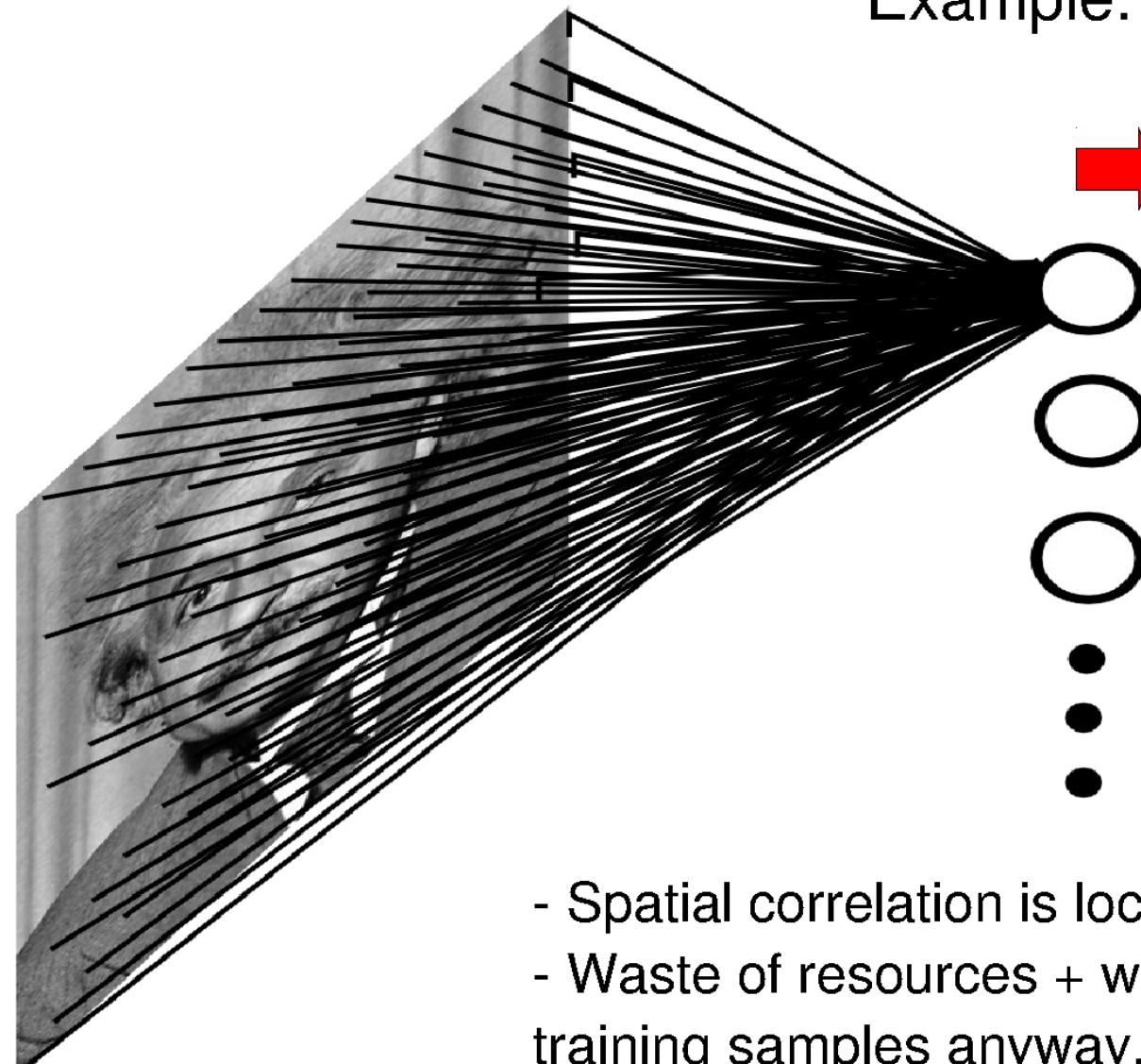
www.grand-illusions.com

Outline

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

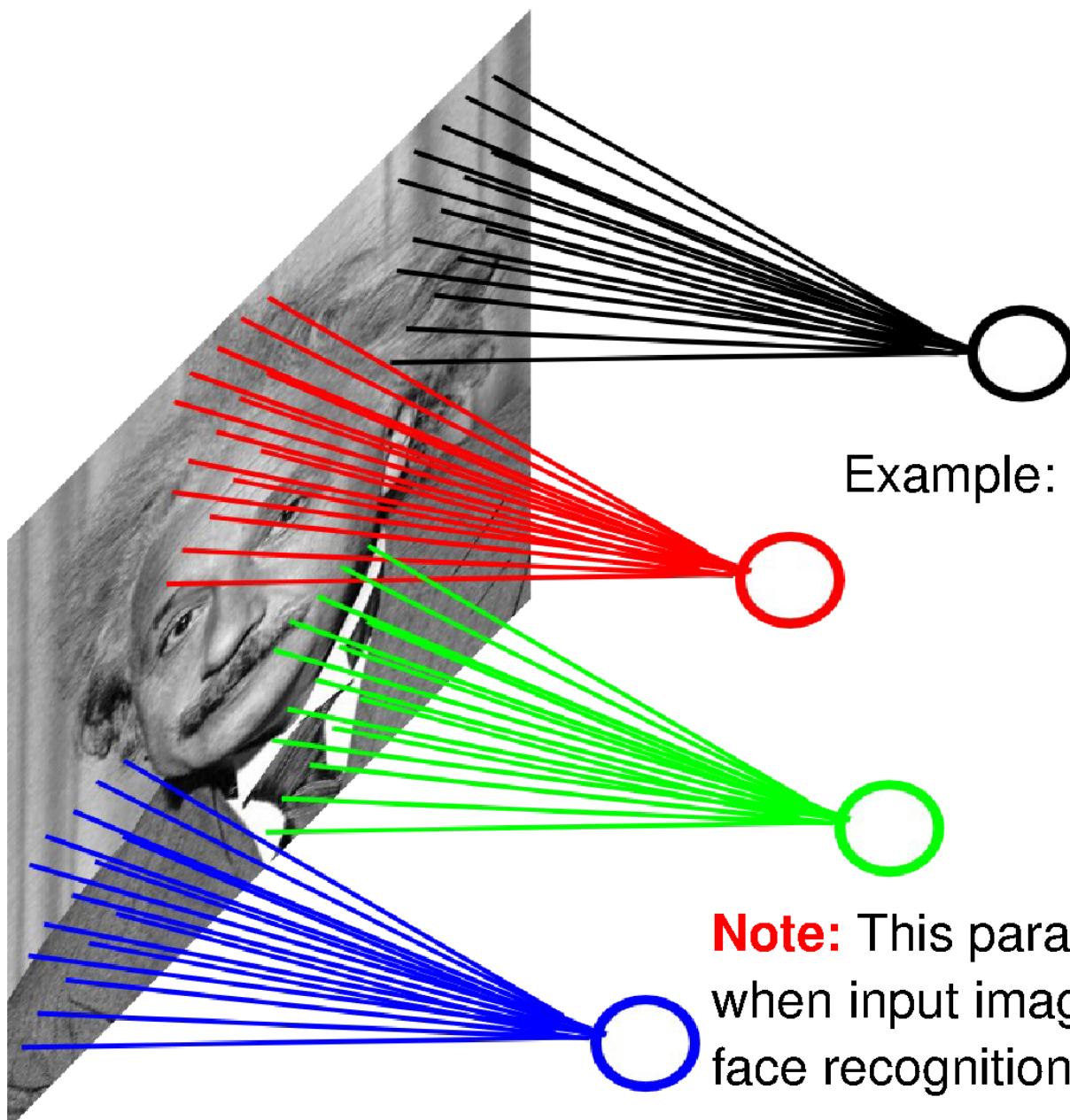
Fully Connected Layer

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



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

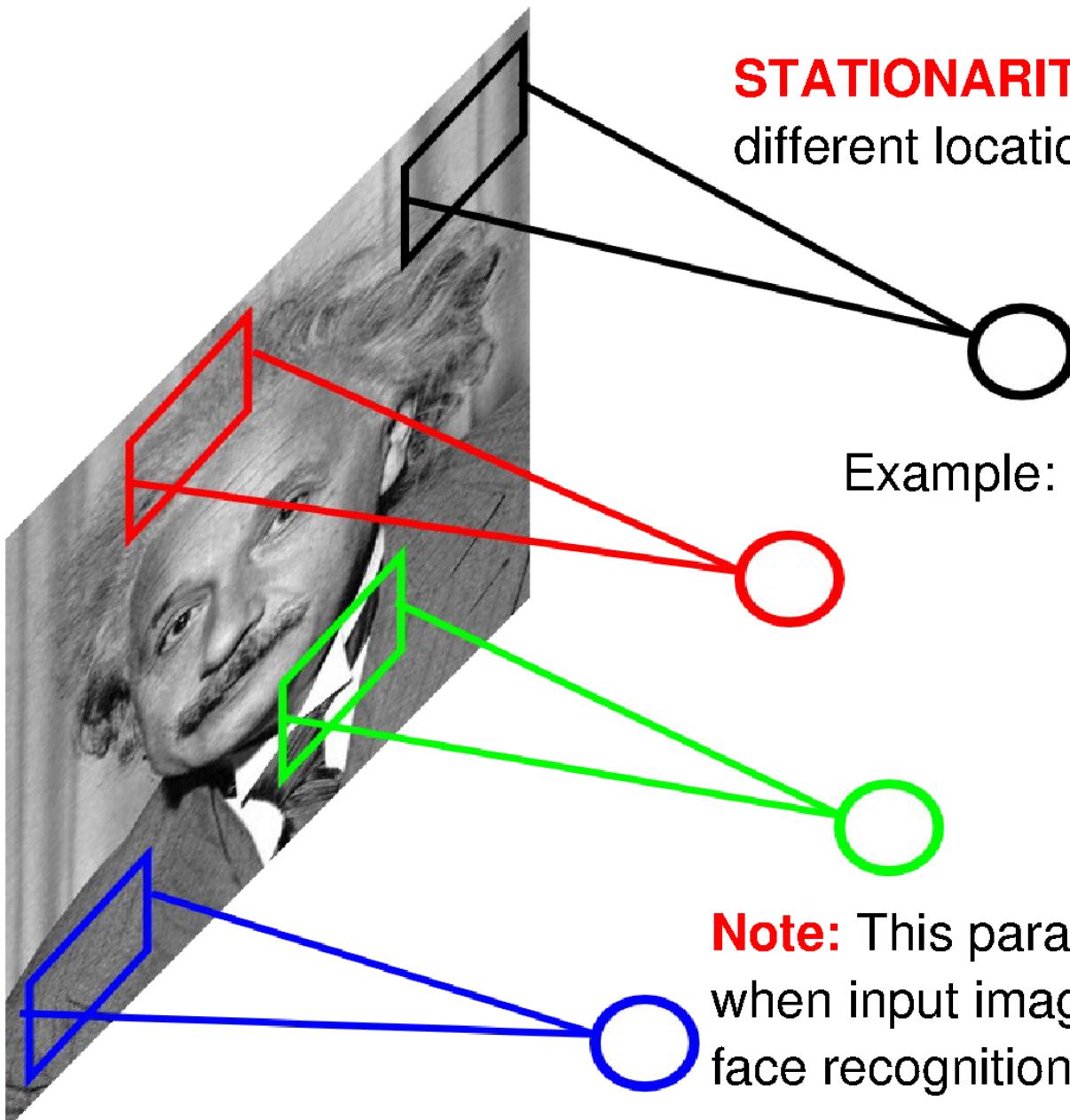
Locally Connected Layer



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

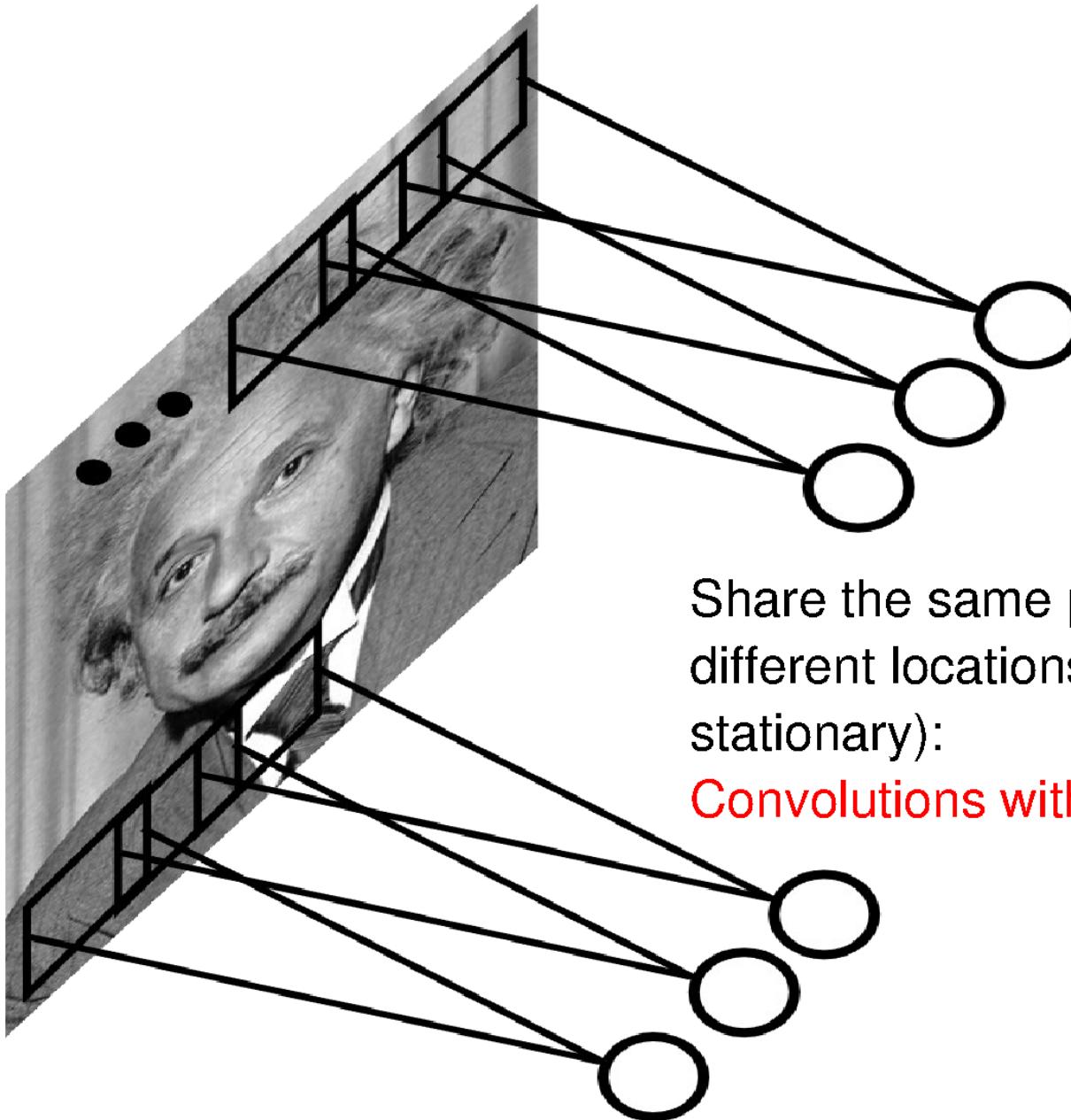


STATIONARITY? Statistics is similar at different locations

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).

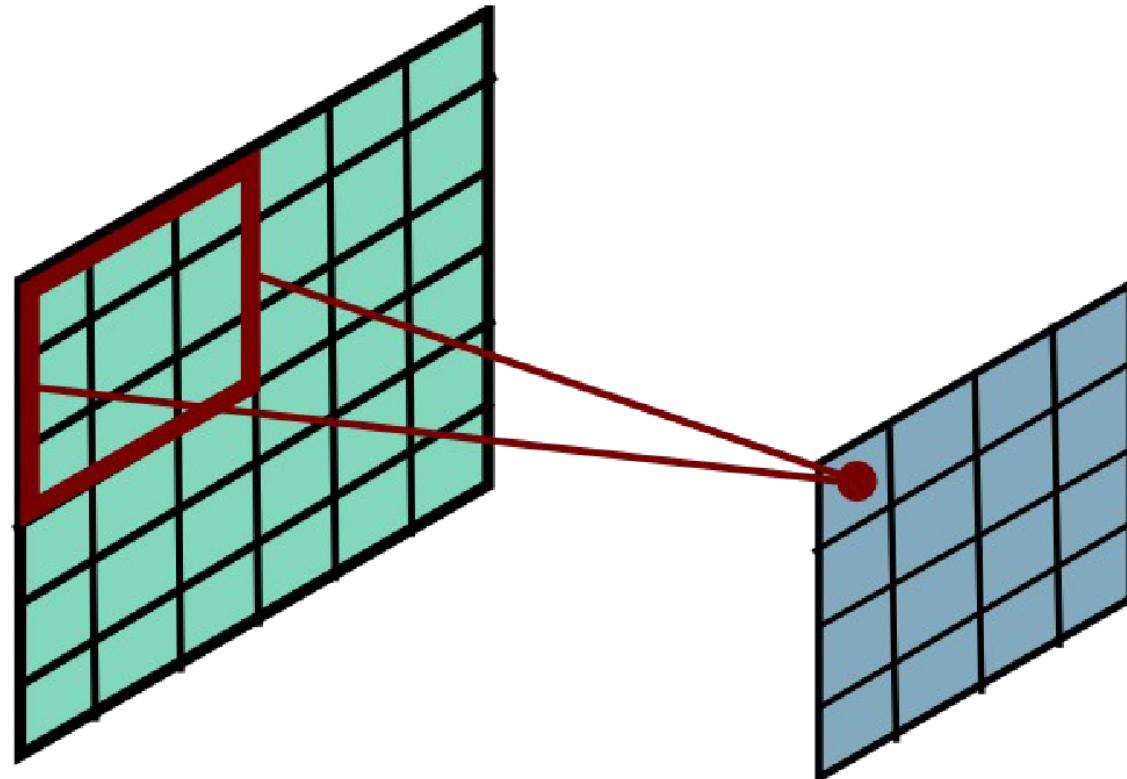
Convolutional Layer



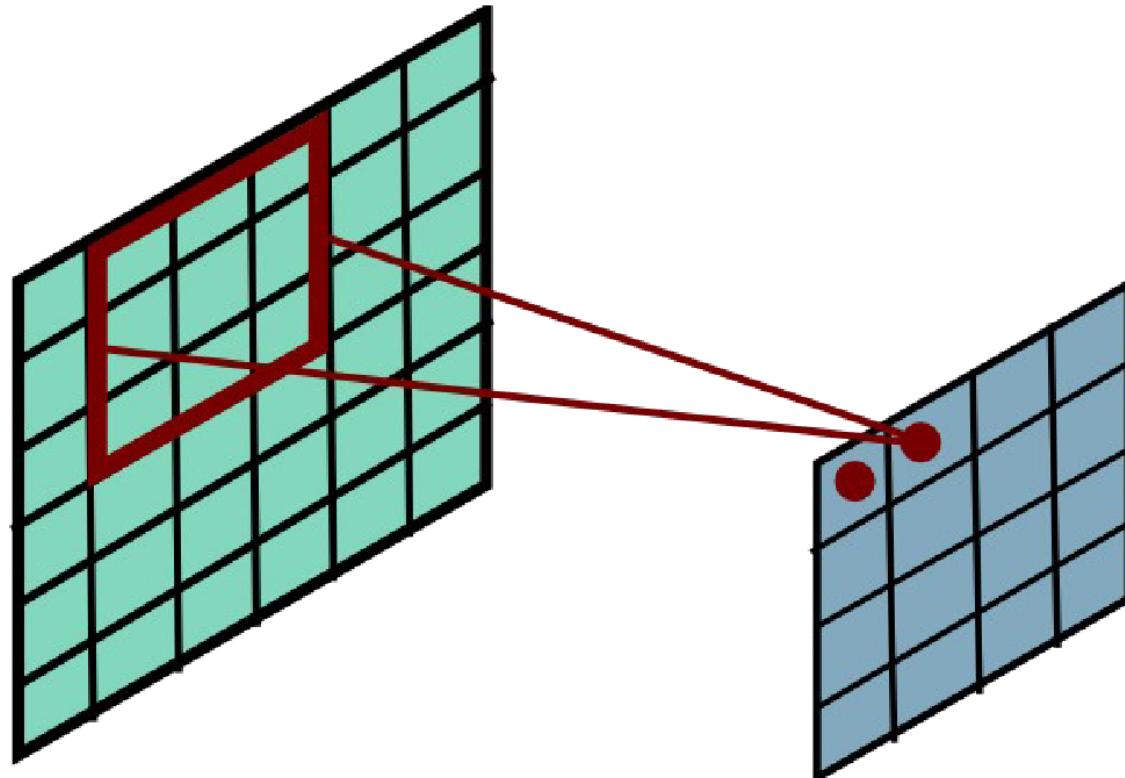
Share the same parameters across
different locations (assuming input is
stationary):

Convolutions with learned kernels

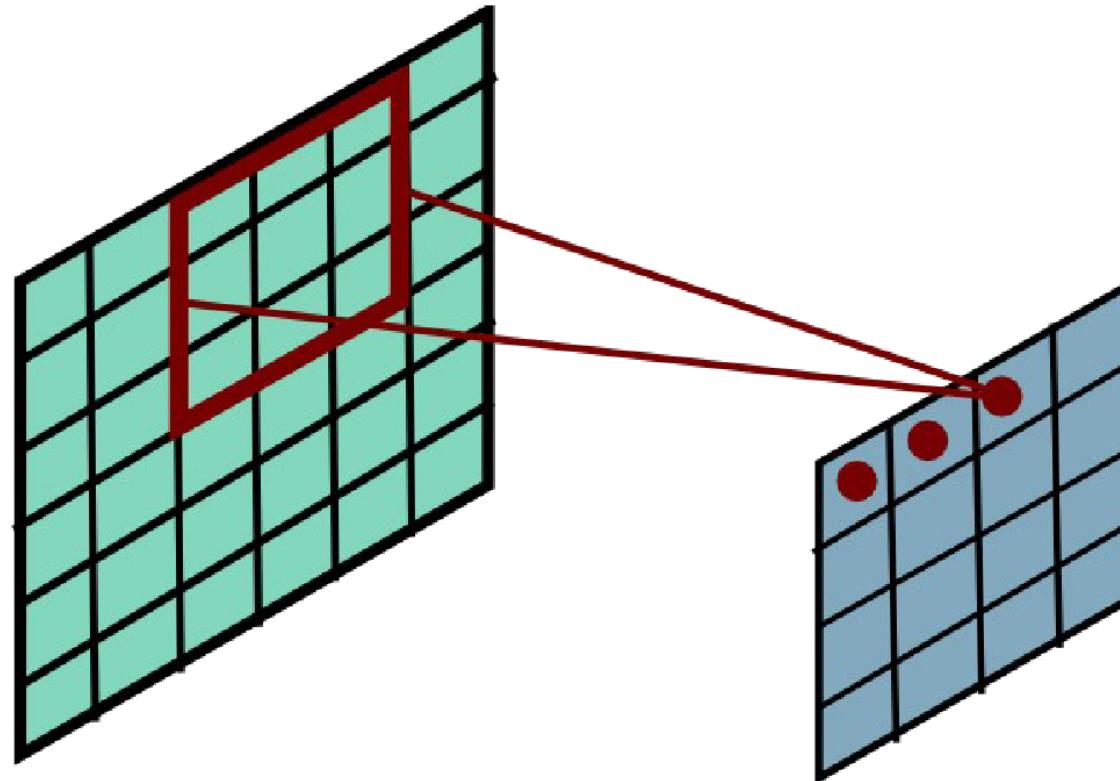
Convolutional Layer



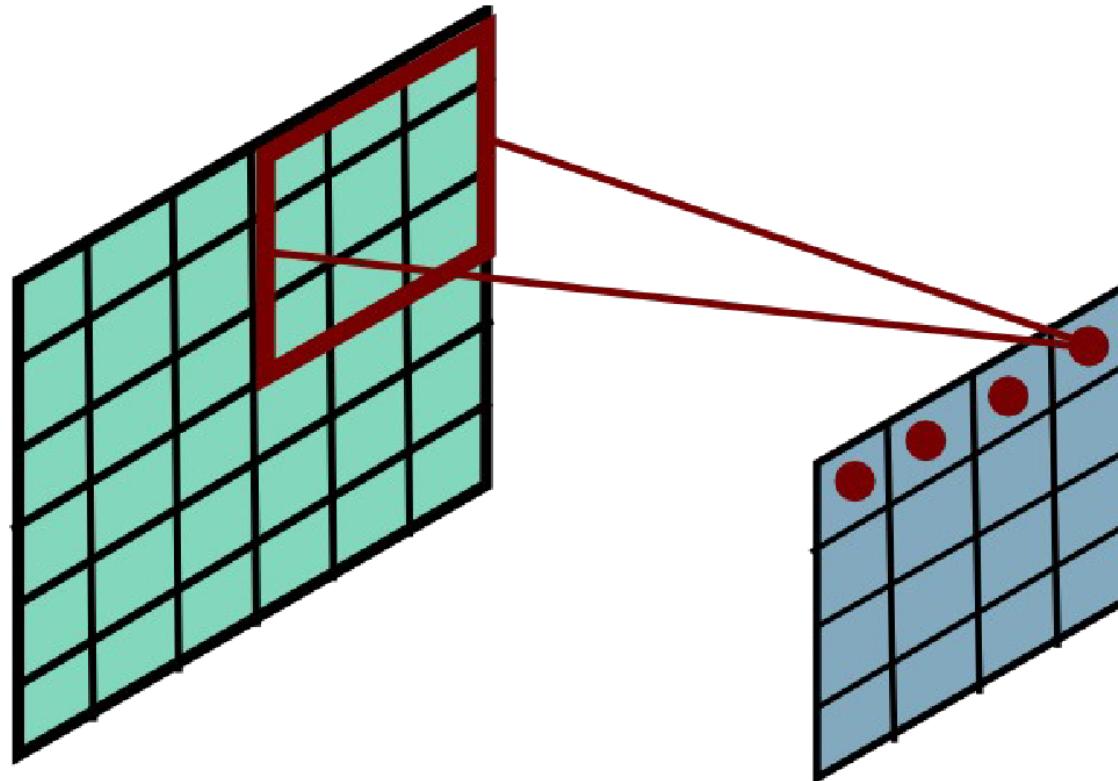
Convolutional Layer



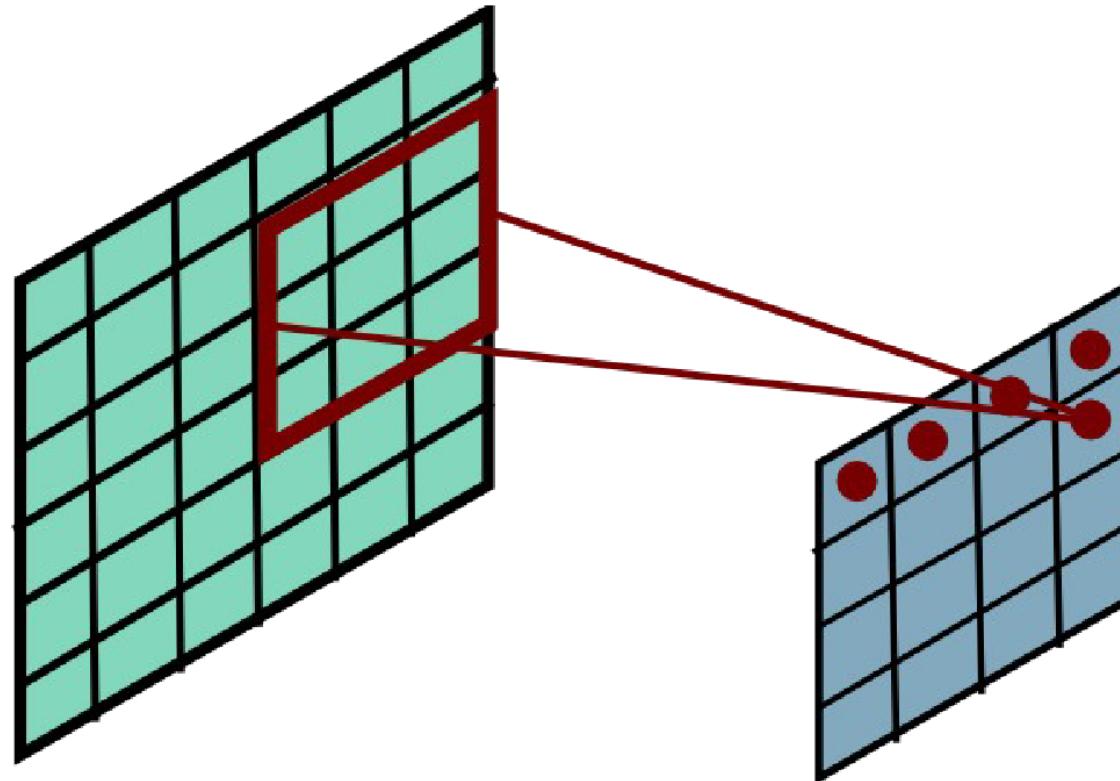
Convolutional Layer



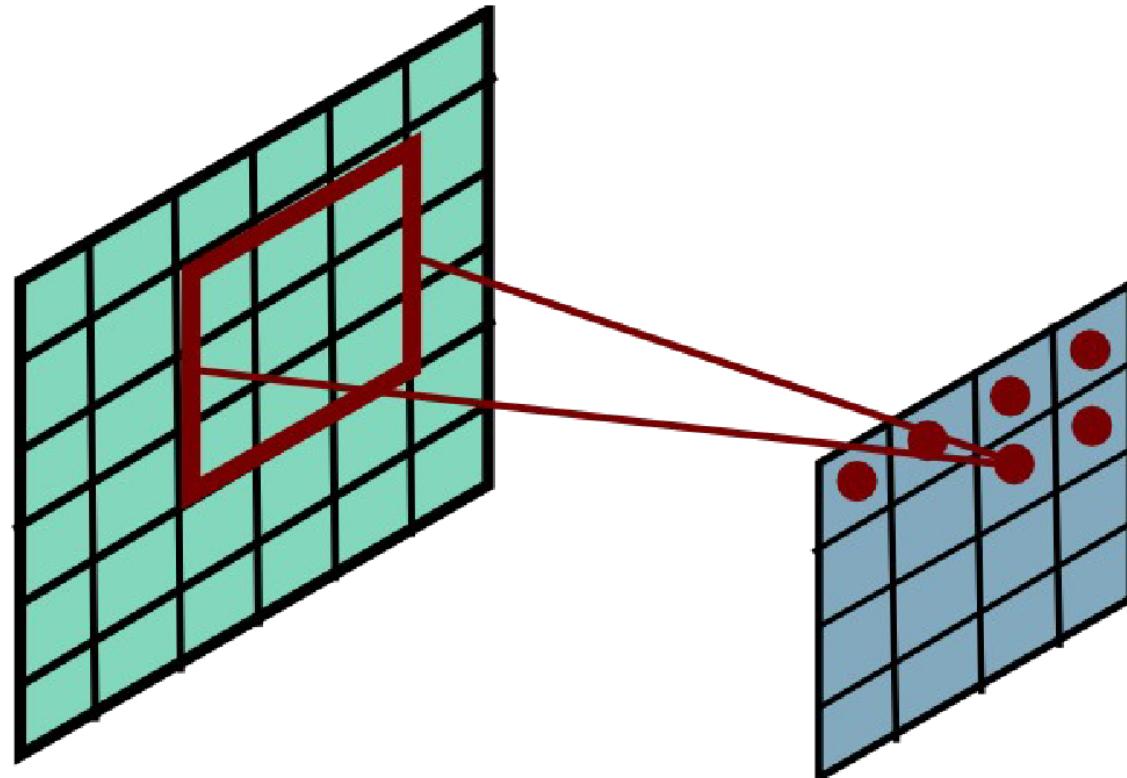
Convolutional Layer



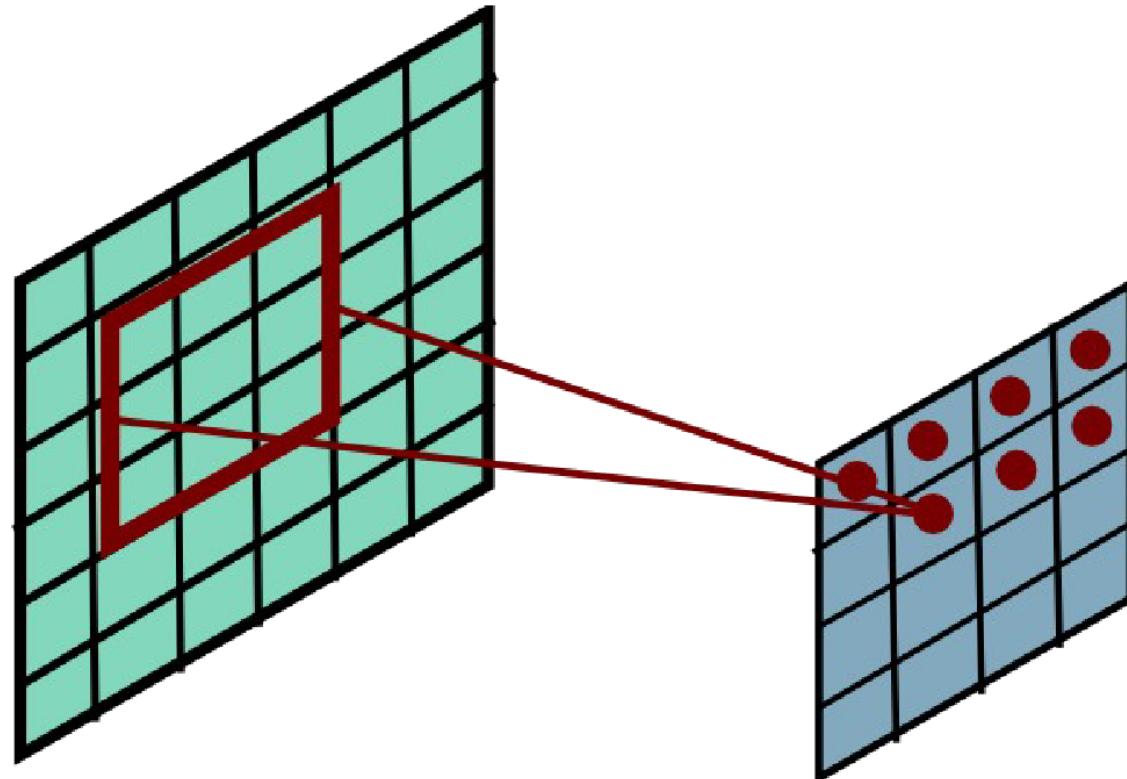
Convolutional Layer



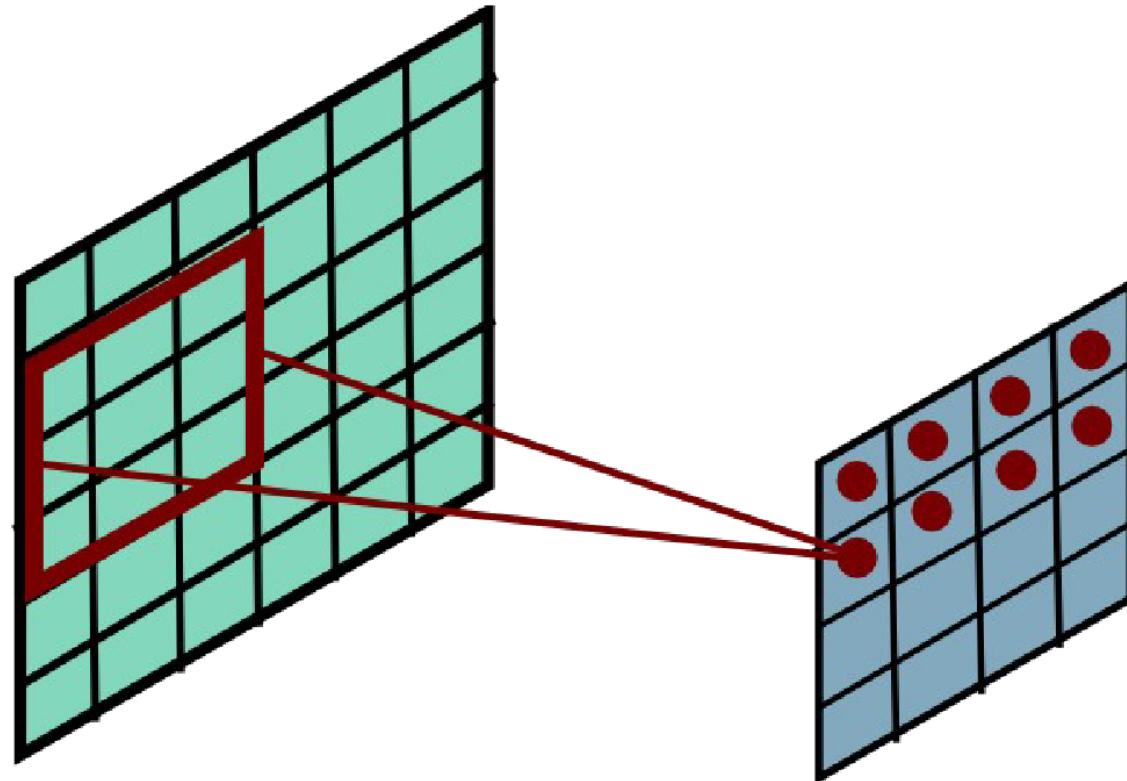
Convolutional Layer



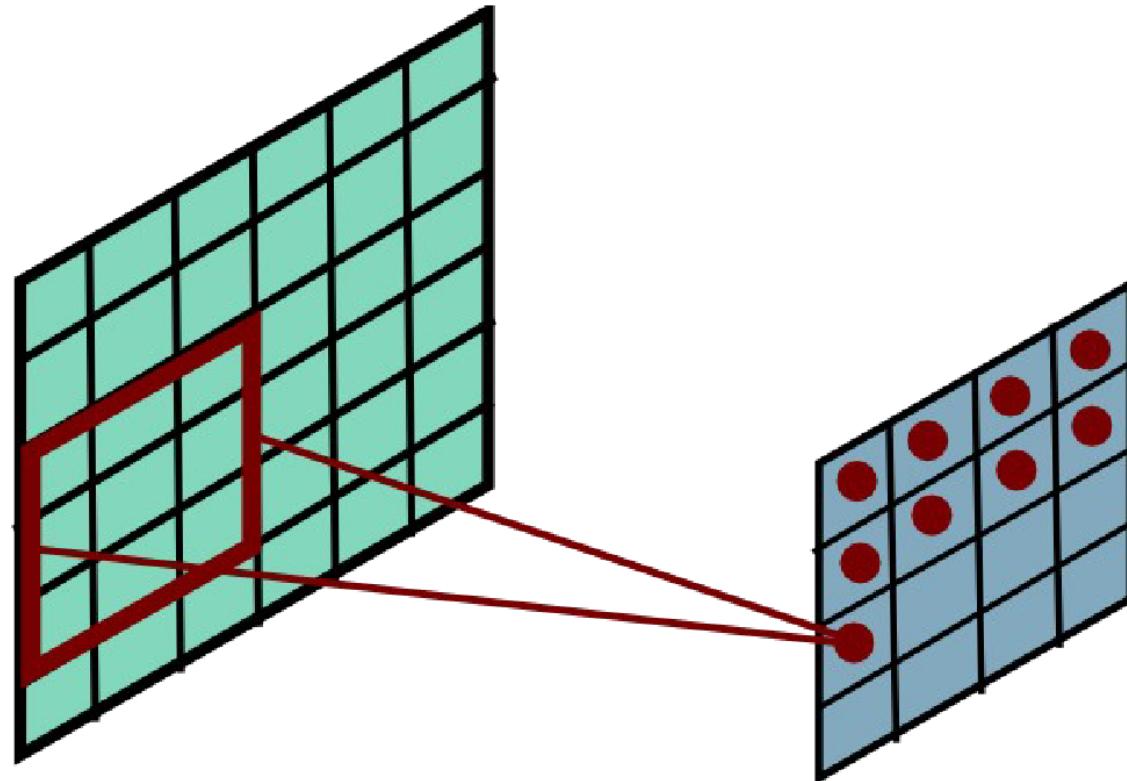
Convolutional Layer



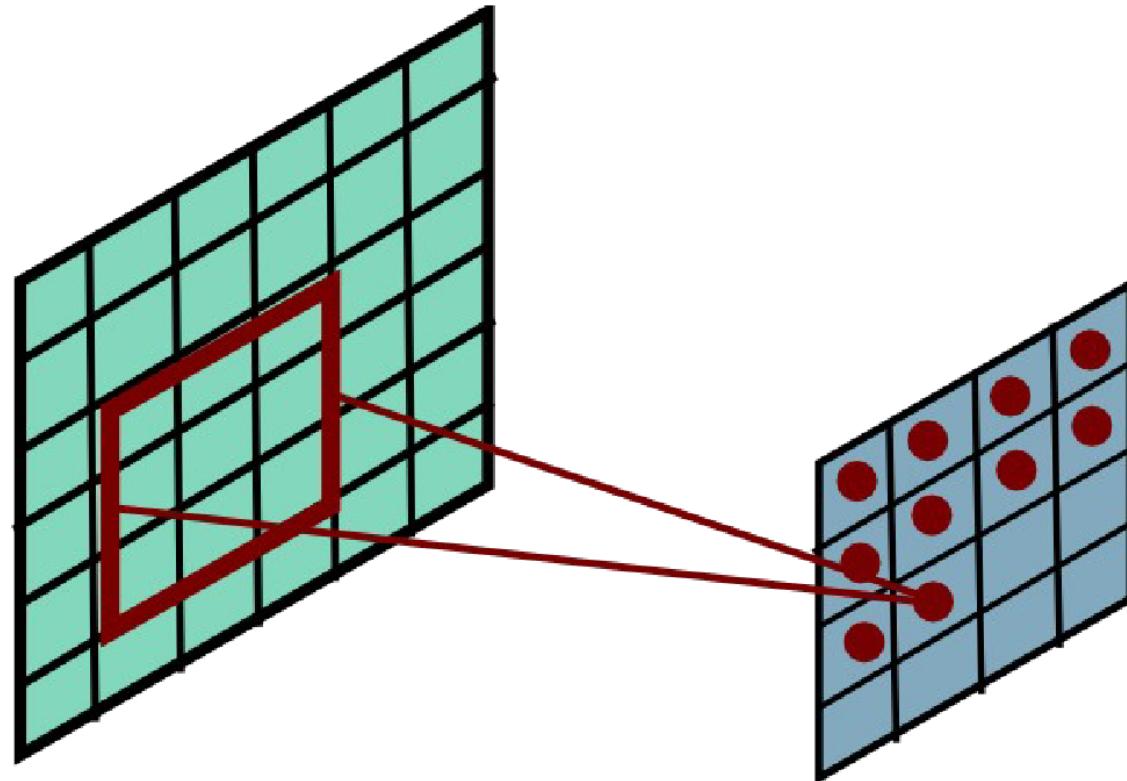
Convolutional Layer



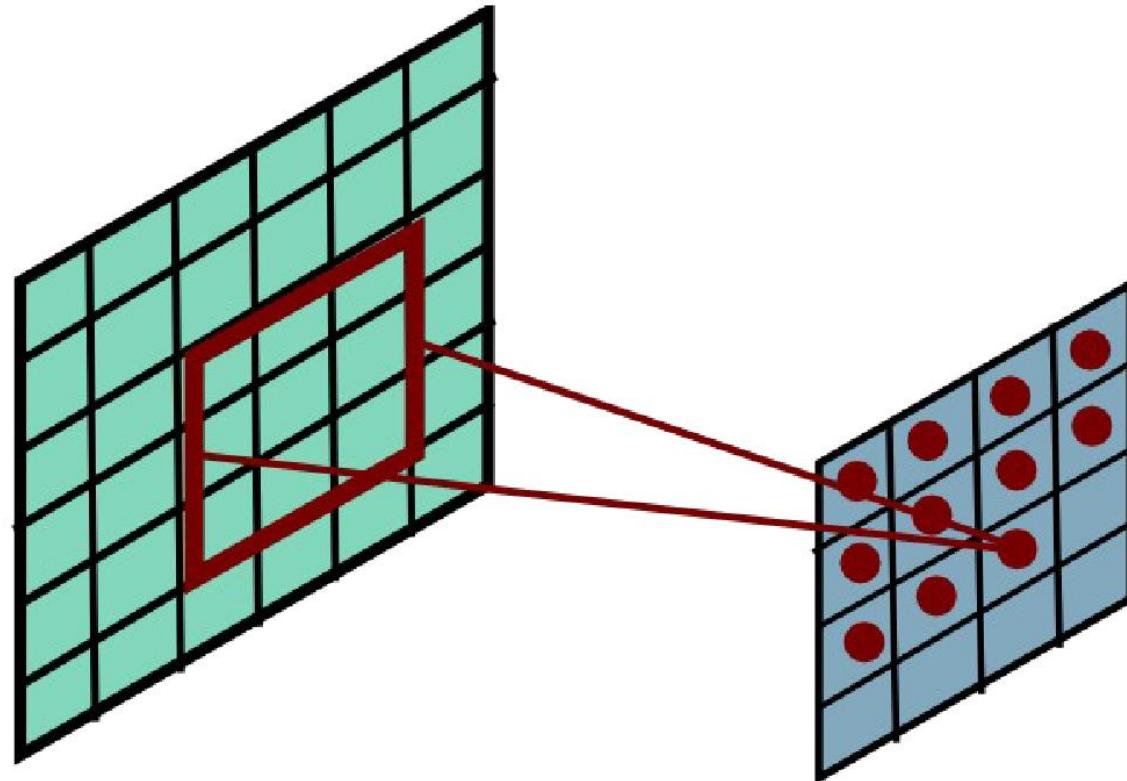
Convolutional Layer



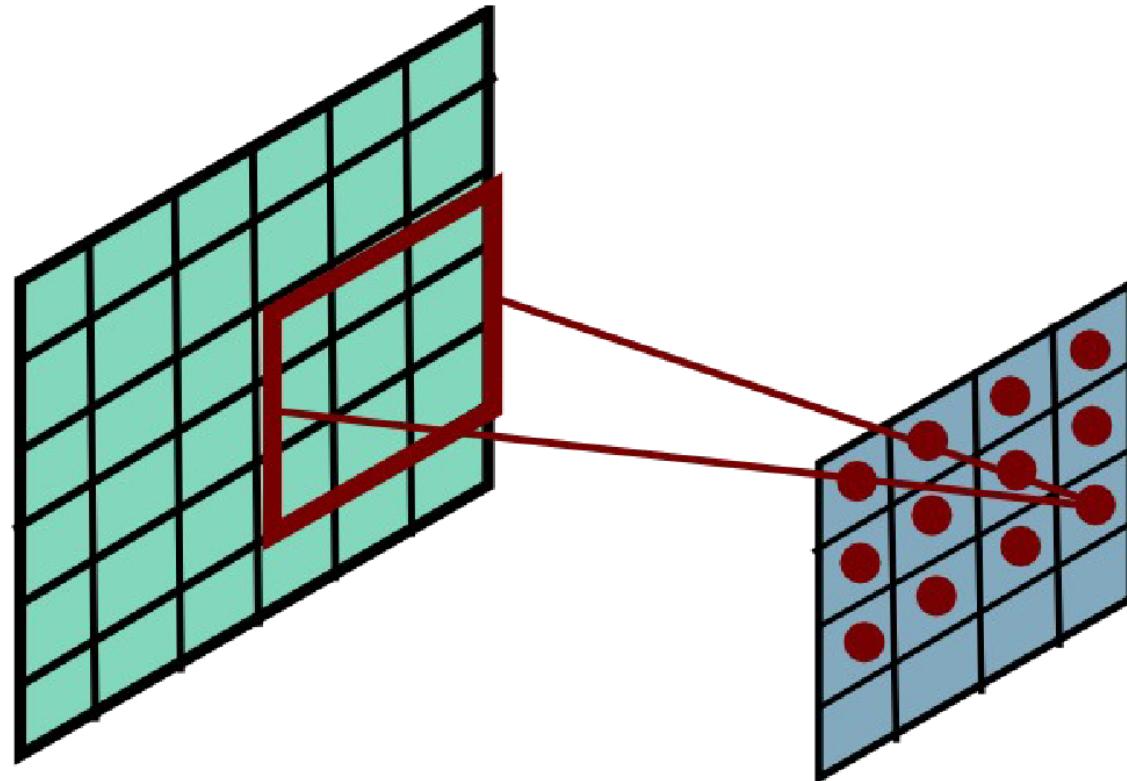
Convolutional Layer



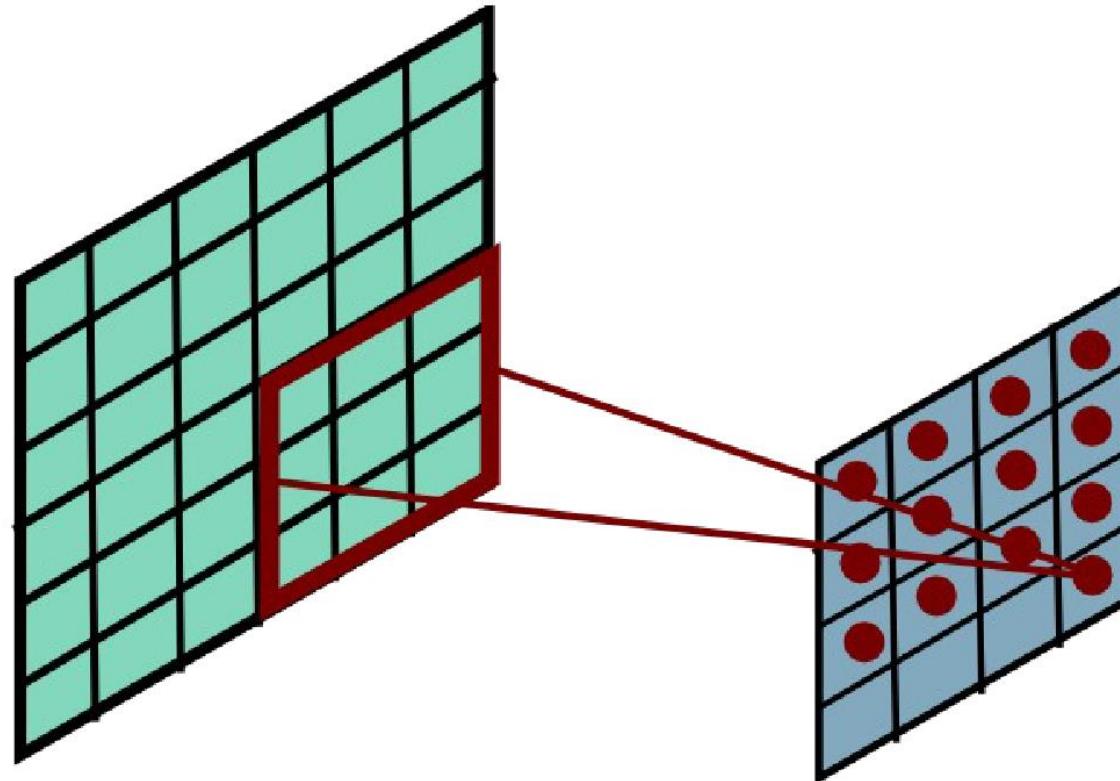
Convolutional Layer



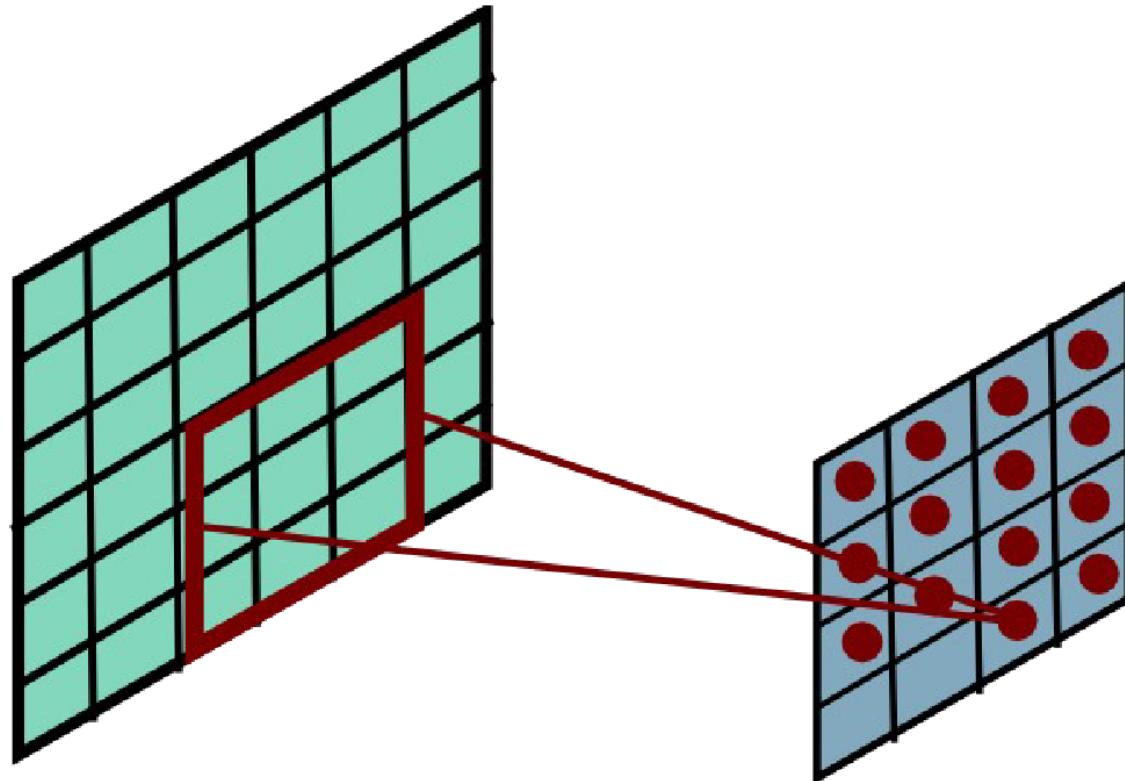
Convolutional Layer



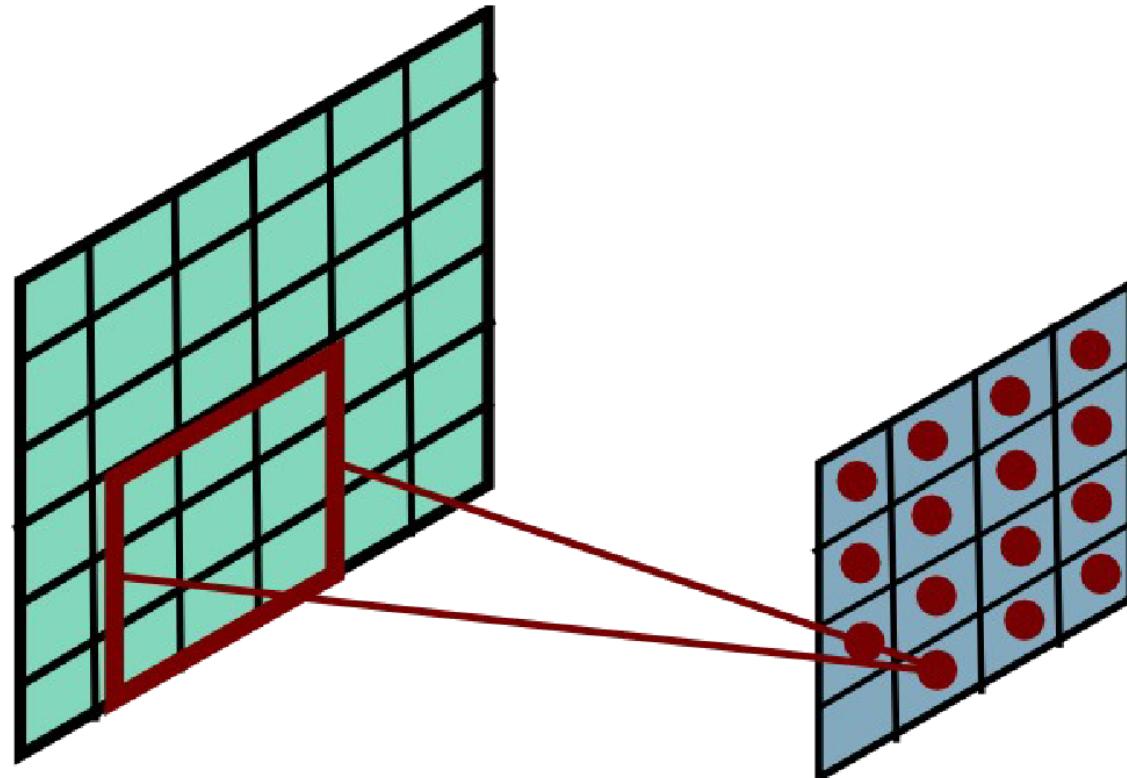
Convolutional Layer



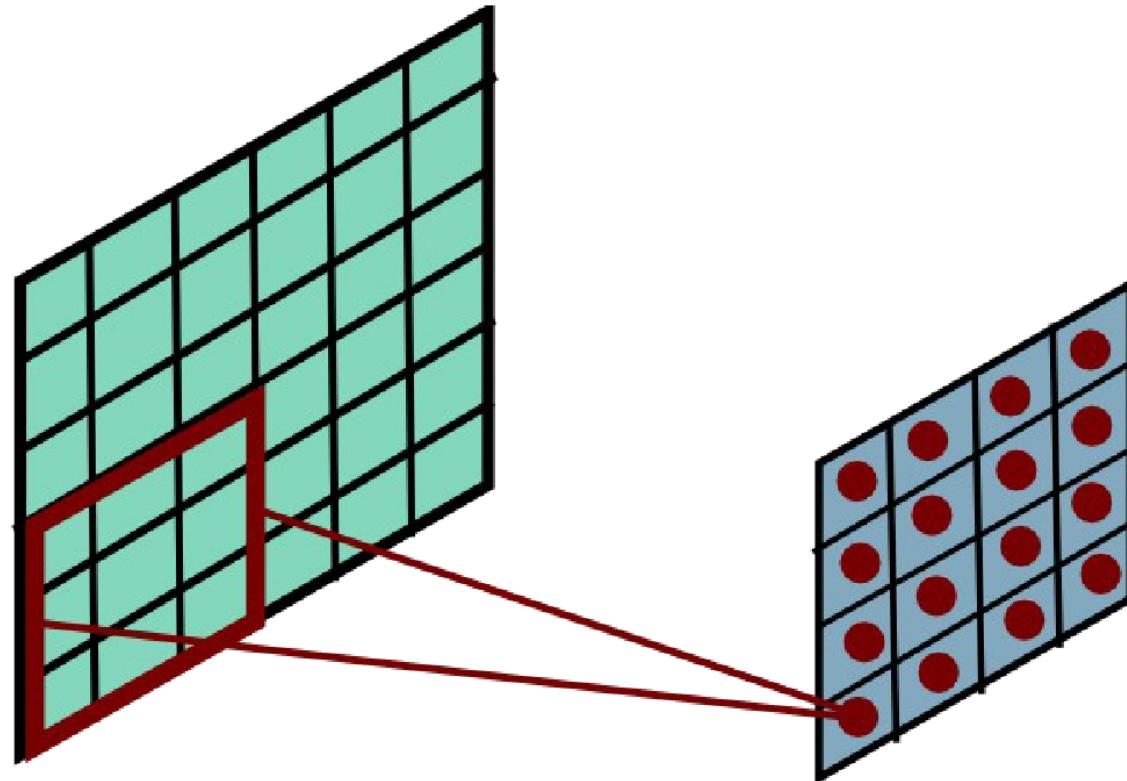
Convolutional Layer



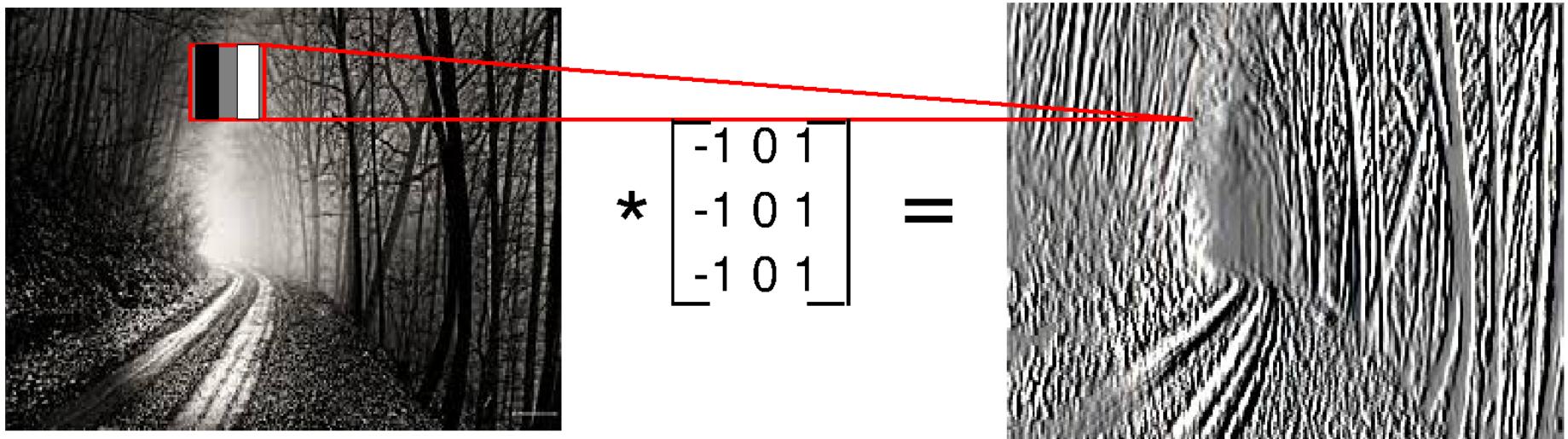
Convolutional Layer



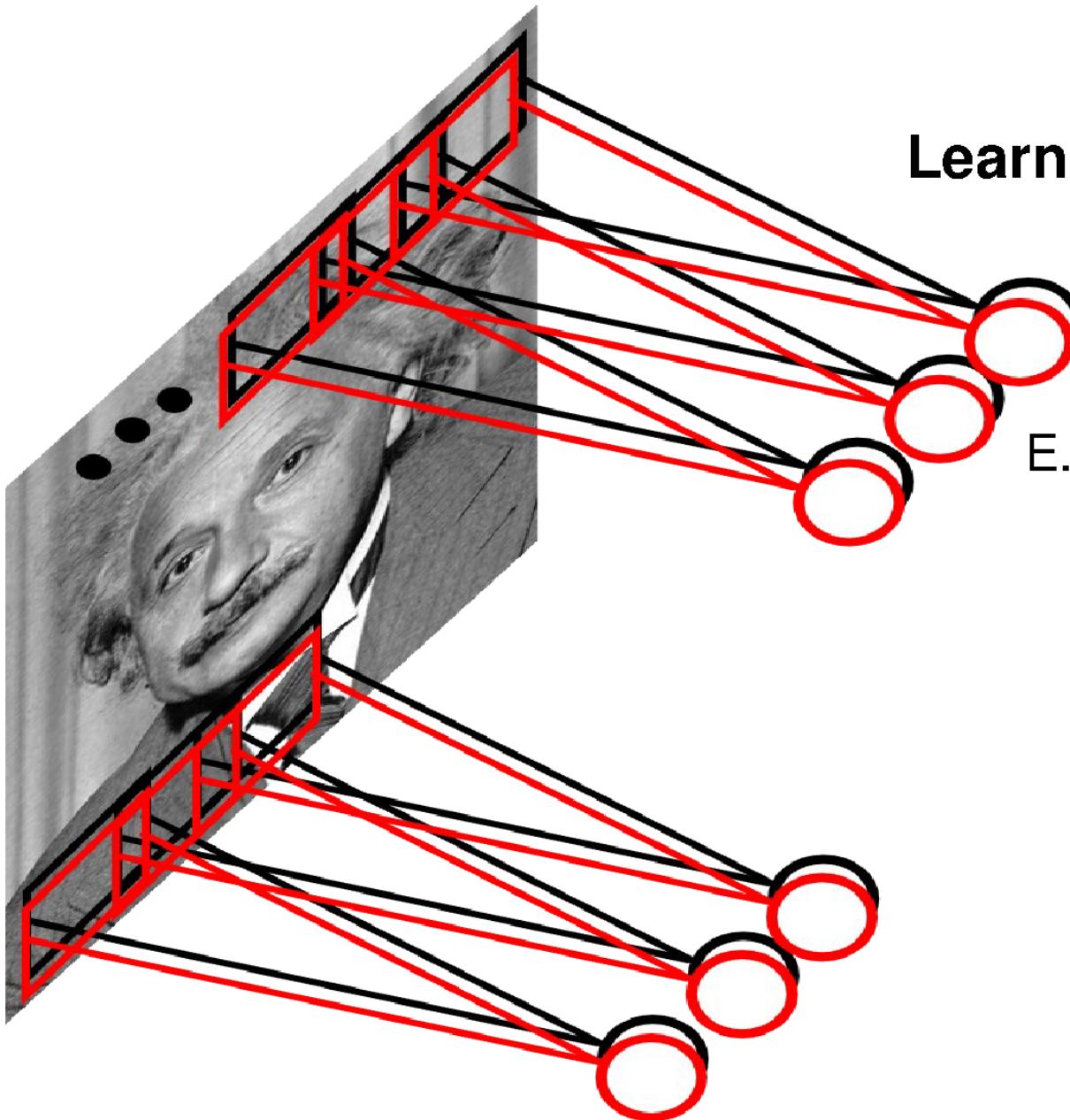
Convolutional Layer



Convolutional Layer



Convolutional Layer



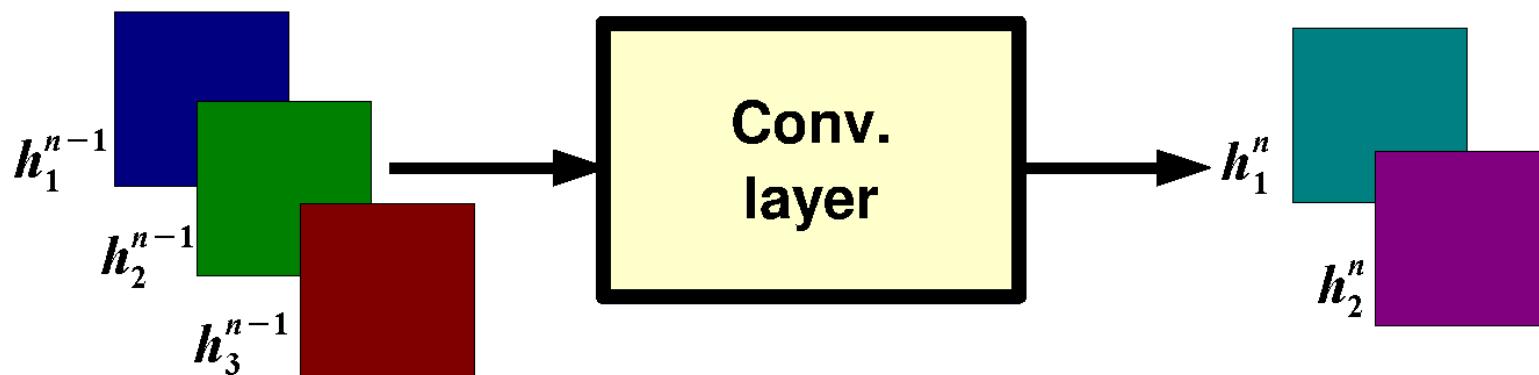
Learn multiple filters.

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

Convolutional Layer

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

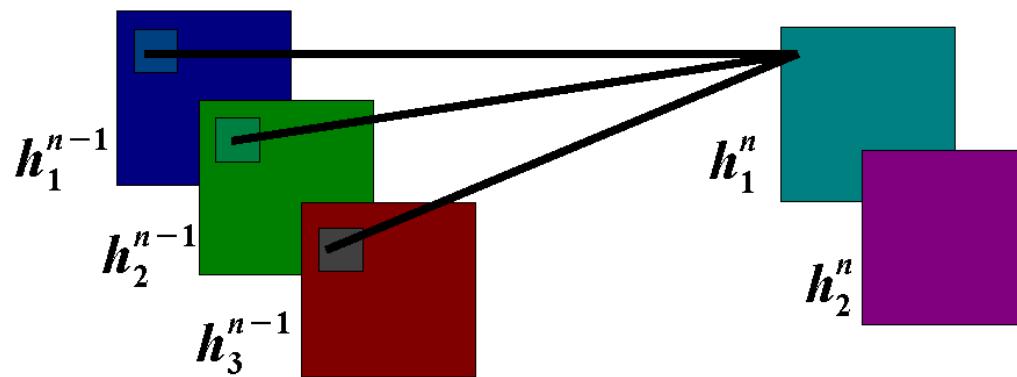
output feature map **input feature map** **kernel**



Convolutional Layer

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

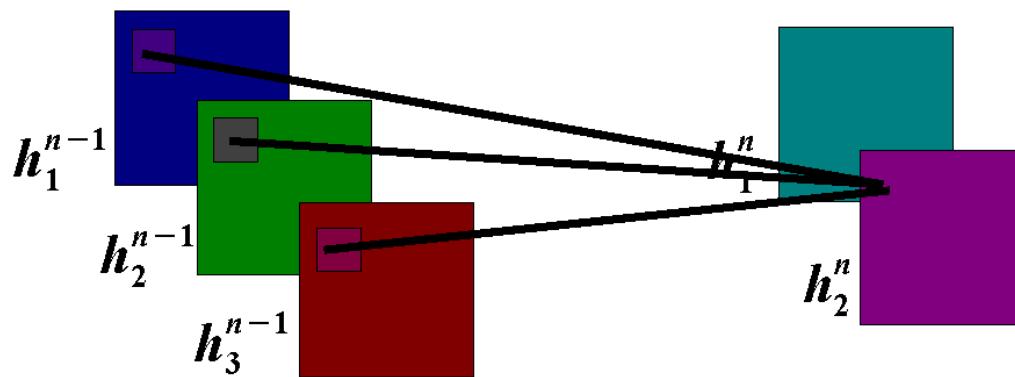
output feature map **input feature map** **kernel**



Convolutional Layer

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

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 $K \times K$, input has size $D \times D$, stride is 1, and there are M input feature maps and N output feature maps then:

- the input has size $M @ D \times D$
- the output has size $N @ (D - K + 1) \times (D - K + 1)$
- the kernels have $M \times N \times K \times K$ 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

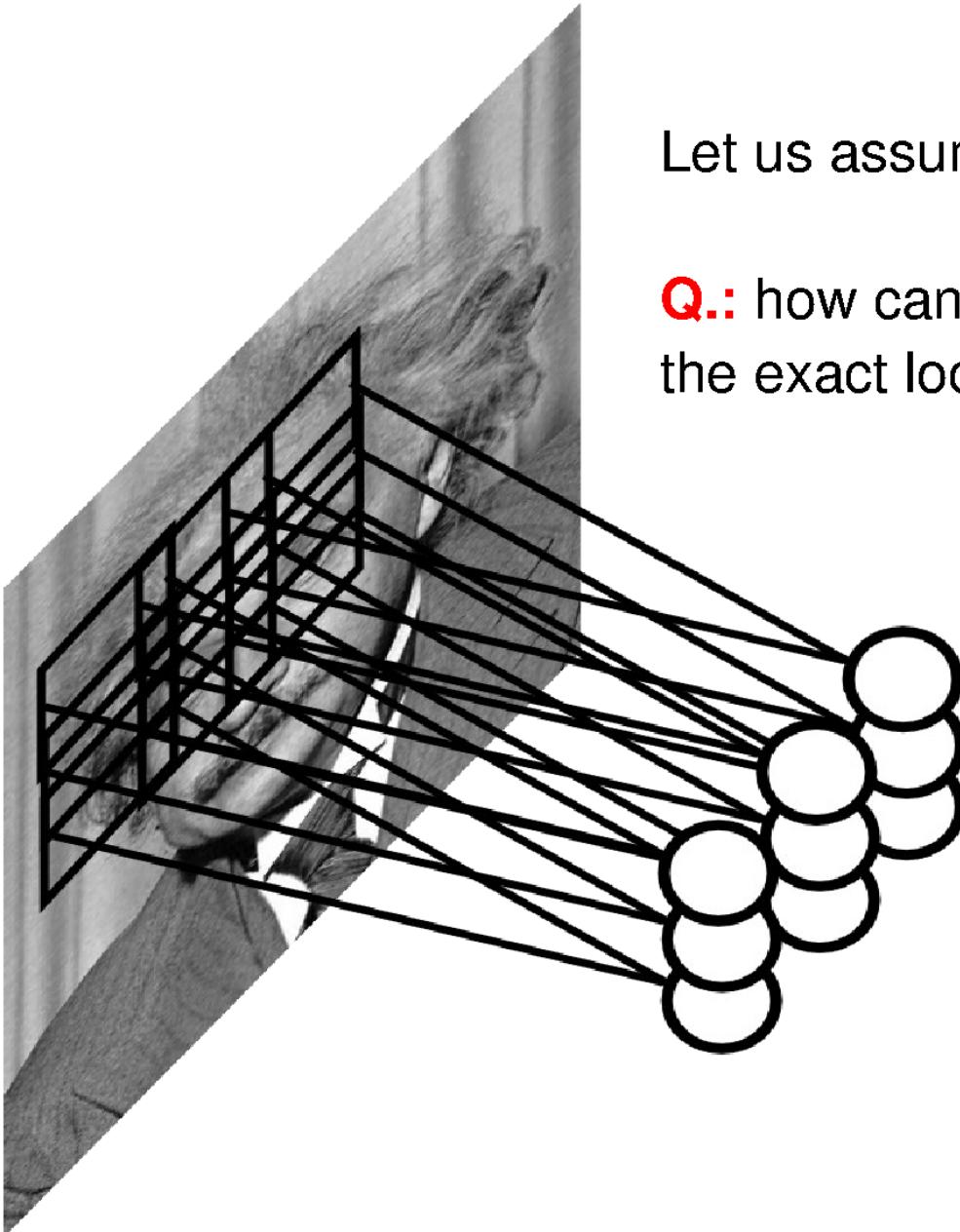
Solution:

- 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**.

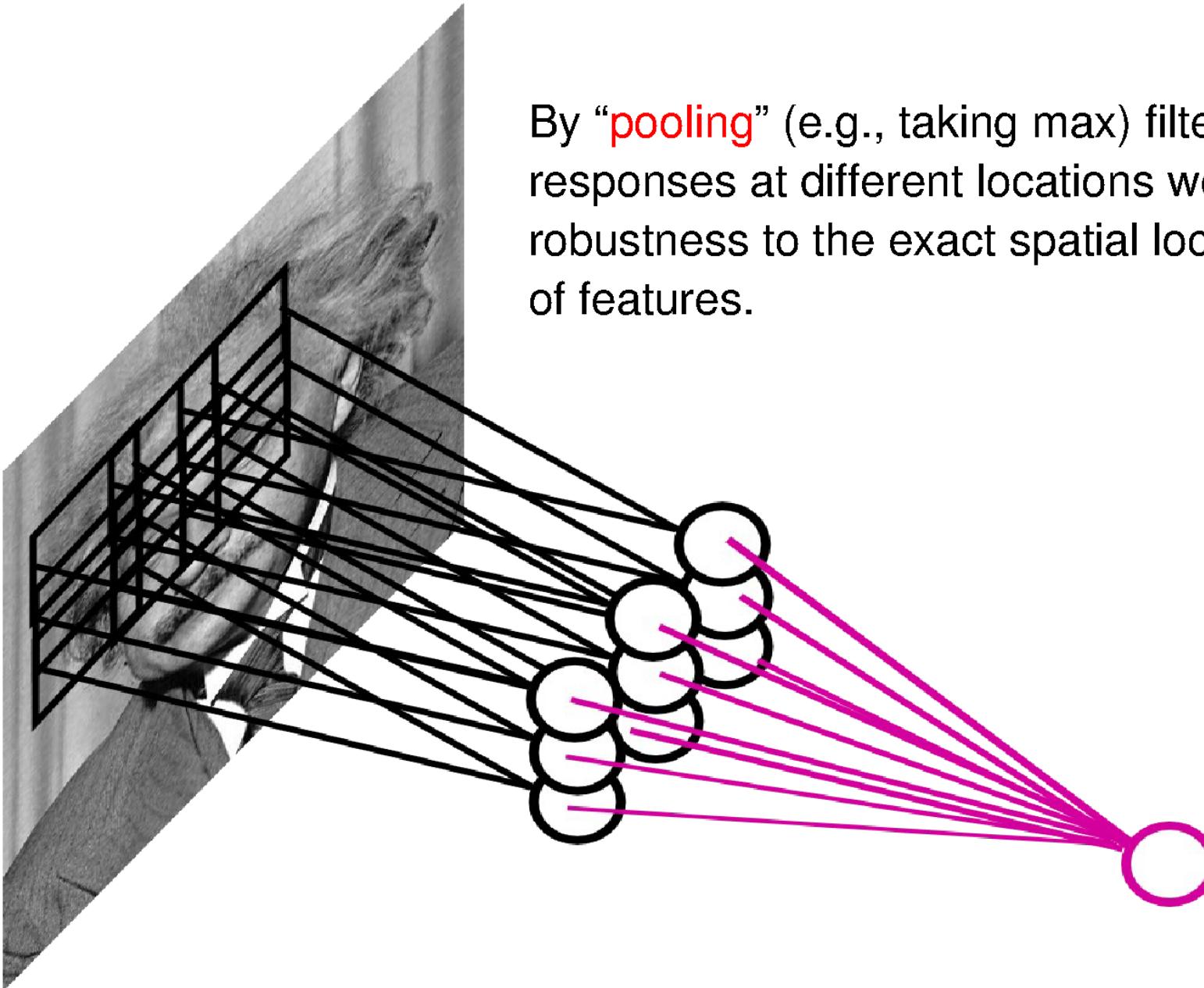
Pooling Layer



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?

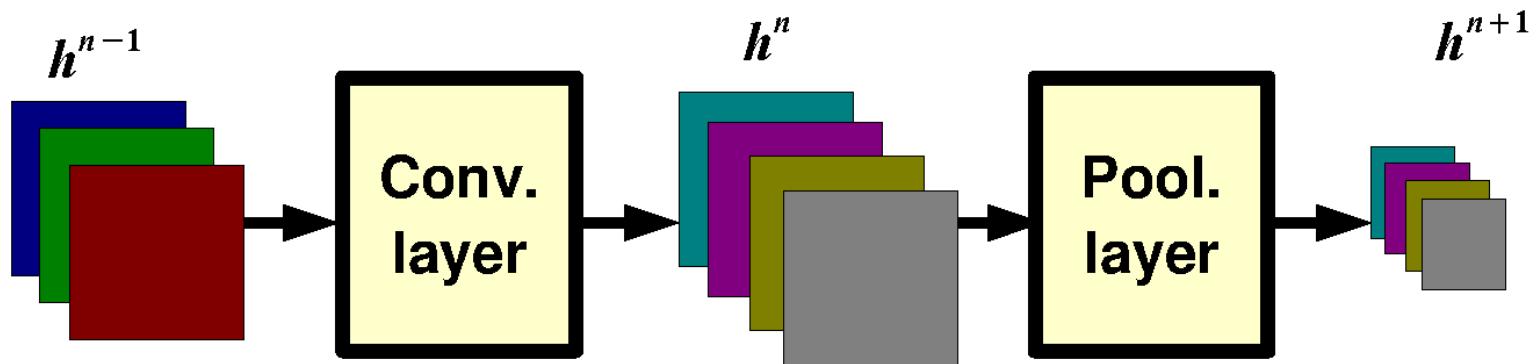
Answer: The size of the output depends on the stride between the pools. For instance, if pools do not overlap and have size $K \times K$, and the input has size $D \times D$ with M input feature maps, then:

- output is $M @ (D/K) \times (D/K)$
- the computational cost is proportional to the size of the input (negligible compared to a convolutional layer)

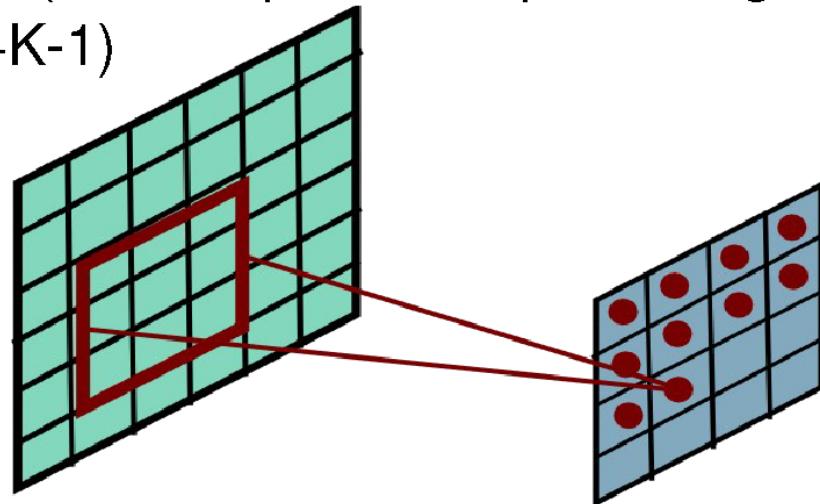
Question: How should I set the size of the pools?

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).

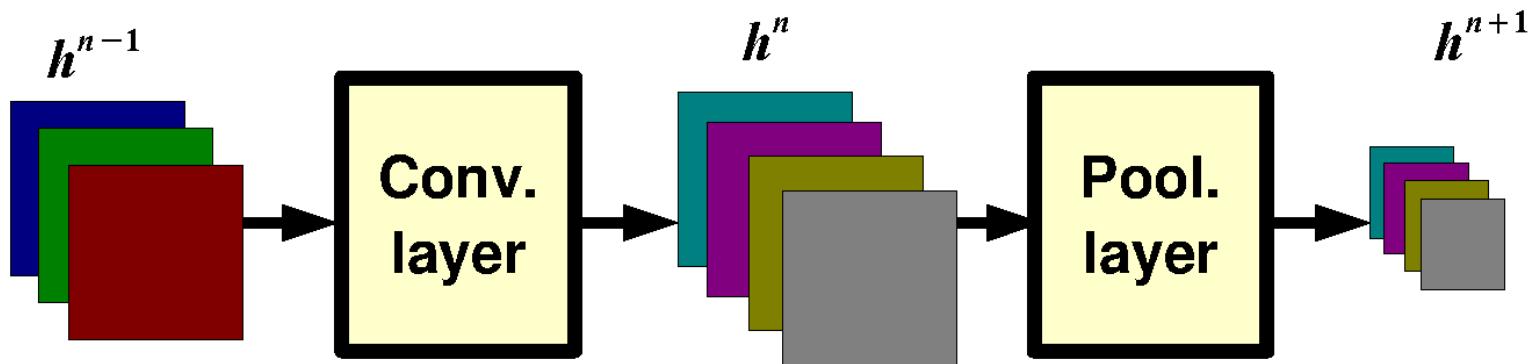
Pooling Layer: Receptive Field Size



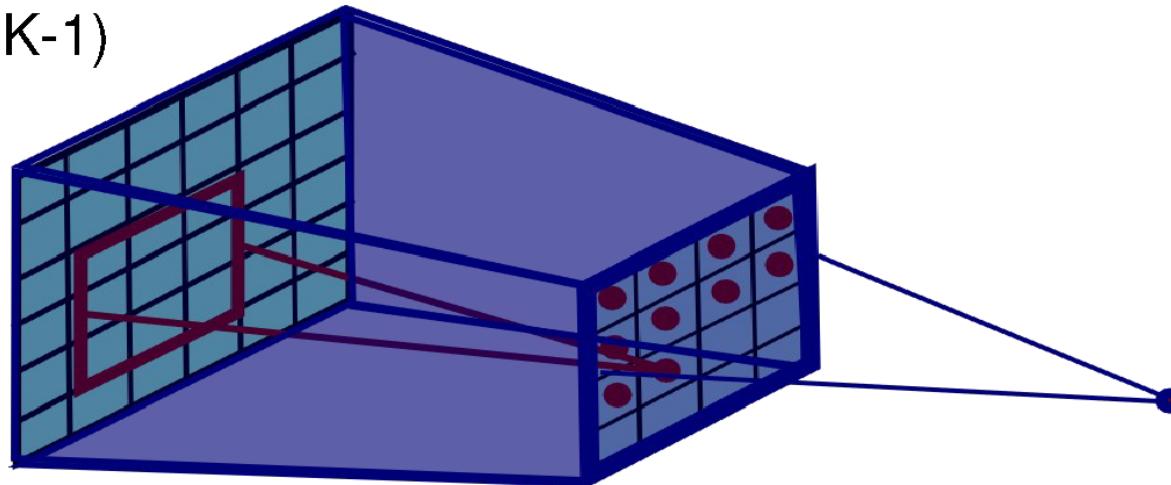
If convolutional filters have size $K \times K$ and stride 1, and pooling layer has pools of size $P \times P$, then each unit in the pooling layer depends upon a patch (at the input of the preceding conv. layer) of size:
 $(P+K-1) \times (P+K-1)$



Pooling Layer: Receptive Field Size

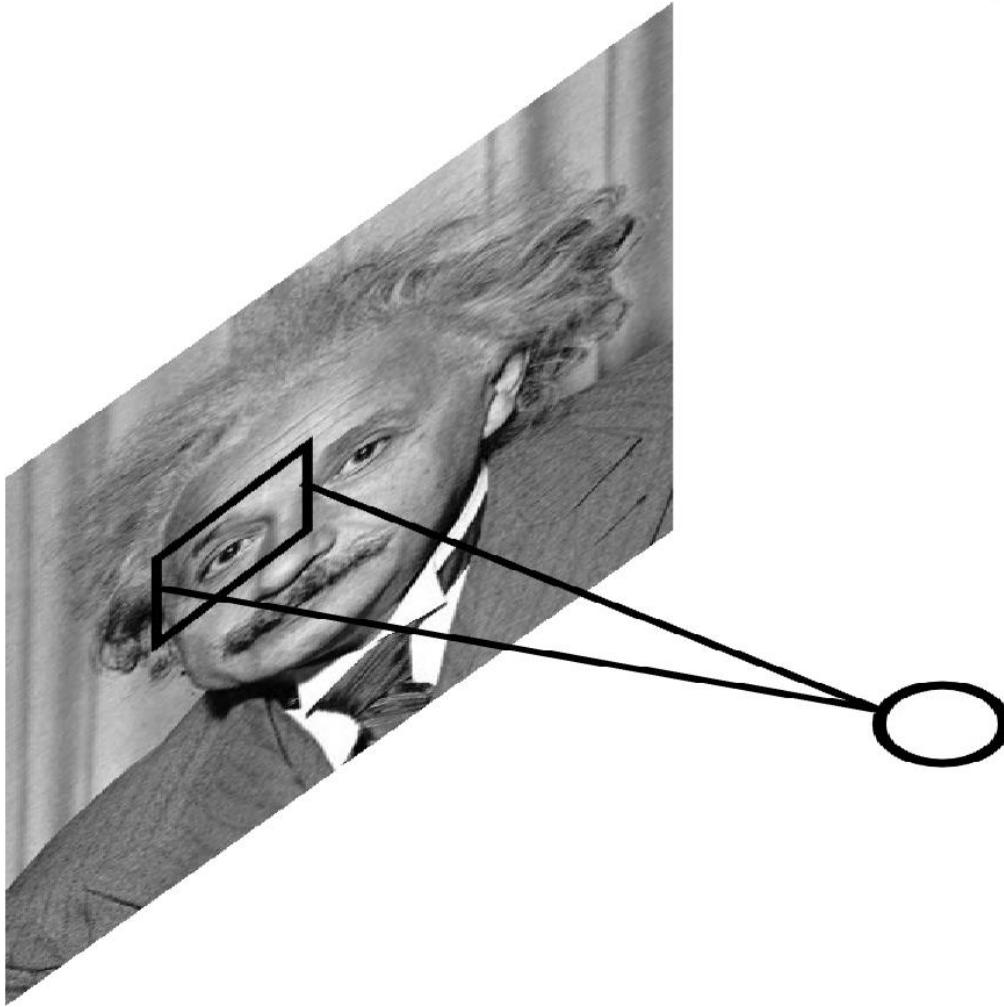


If convolutional filters have size $K \times K$ and stride 1, and pooling layer has pools of size $P \times P$, then each unit in the pooling layer depends upon a patch (at the input of the preceding conv. layer) of size:
 $(P+K-1) \times (P+K-1)$



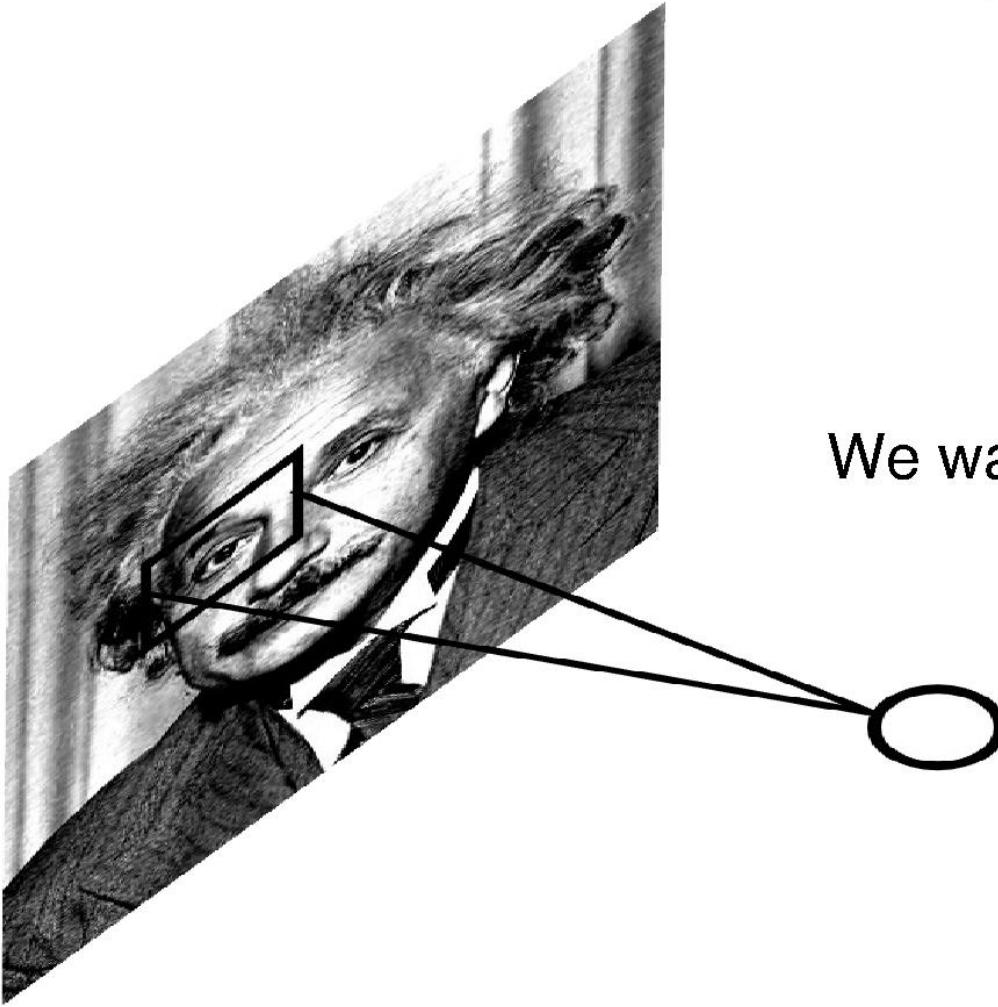
Local Contrast Normalization

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



Local Contrast Normalization

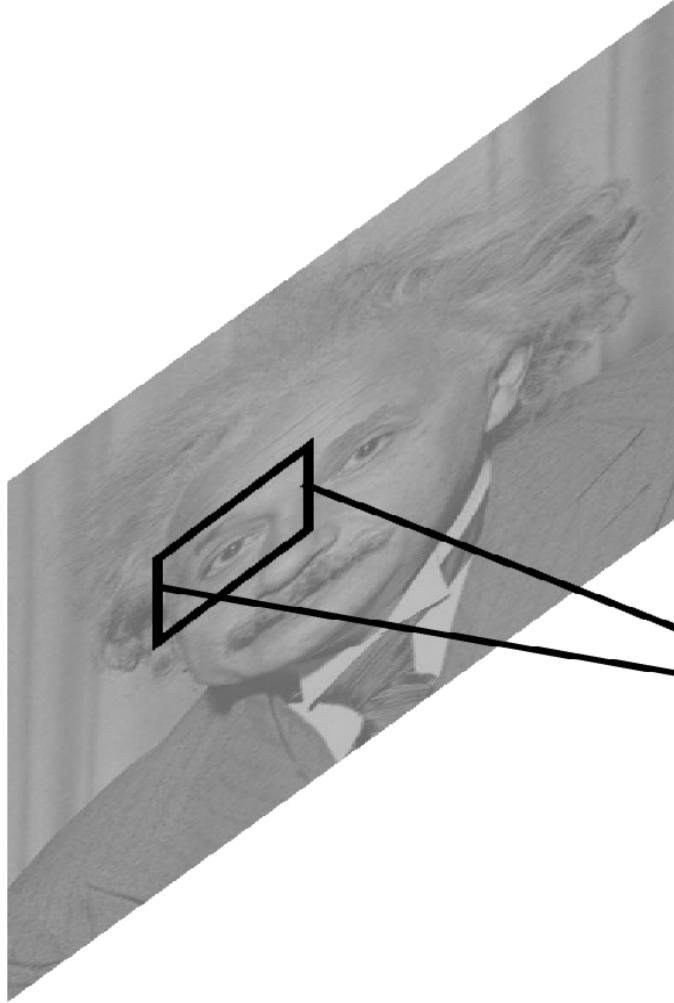
$$h^{i+1}(x, y) = \frac{h^i(x, y) - m^i(N(x, y))}{\sigma^i(N(x, y))}$$



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..

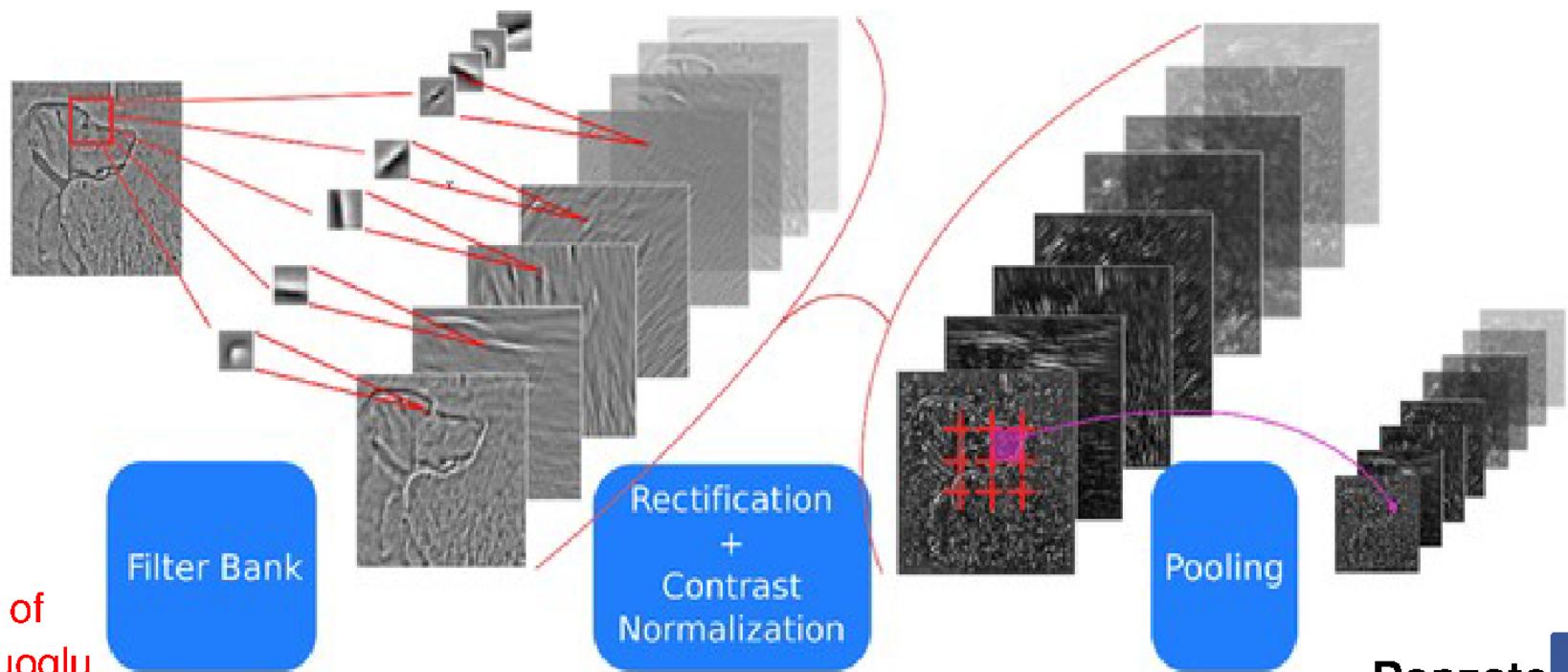
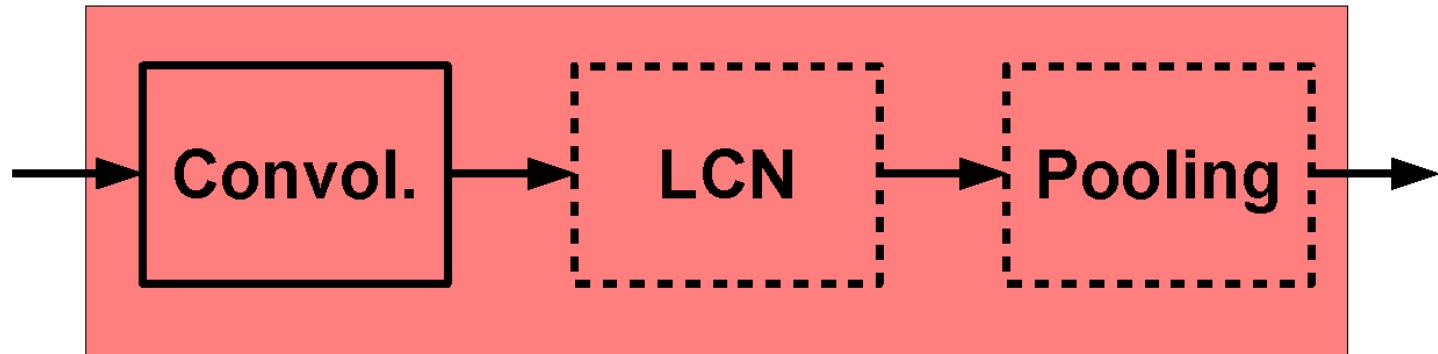
Effects:

- improves invariance
- improves optimization
- increases sparsity

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

ConvNets: Typical Stage

One stage (zoom)

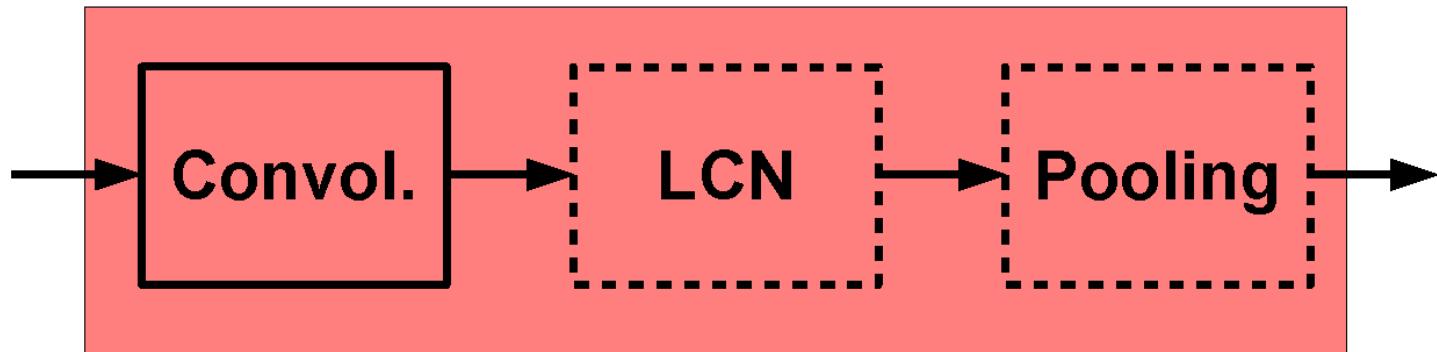


courtesy of
K. Kavukcuoglu

Ranzato

ConvNets: Typical Stage

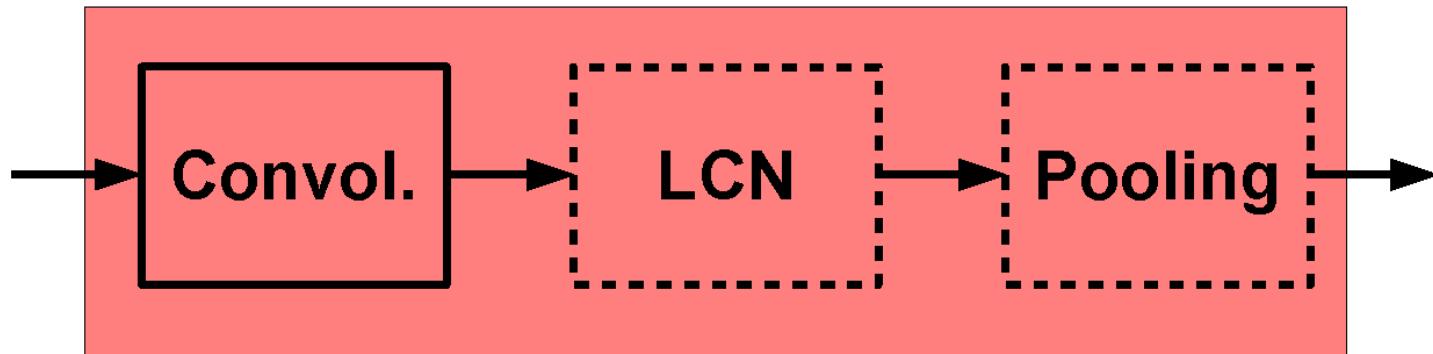
One stage (zoom)



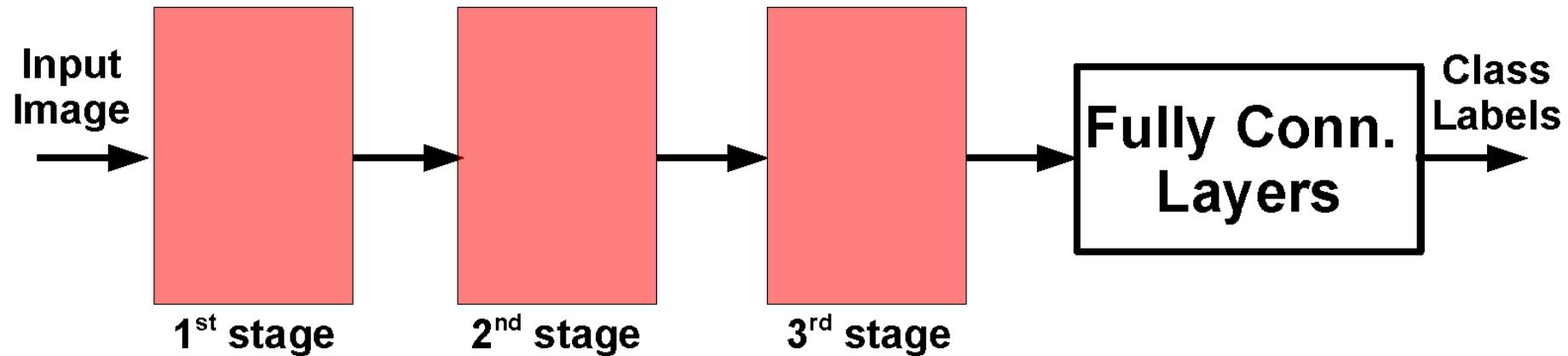
Conceptually similar to: SIFT, HoG, etc.

ConvNets: Typical Architecture

One stage (zoom)

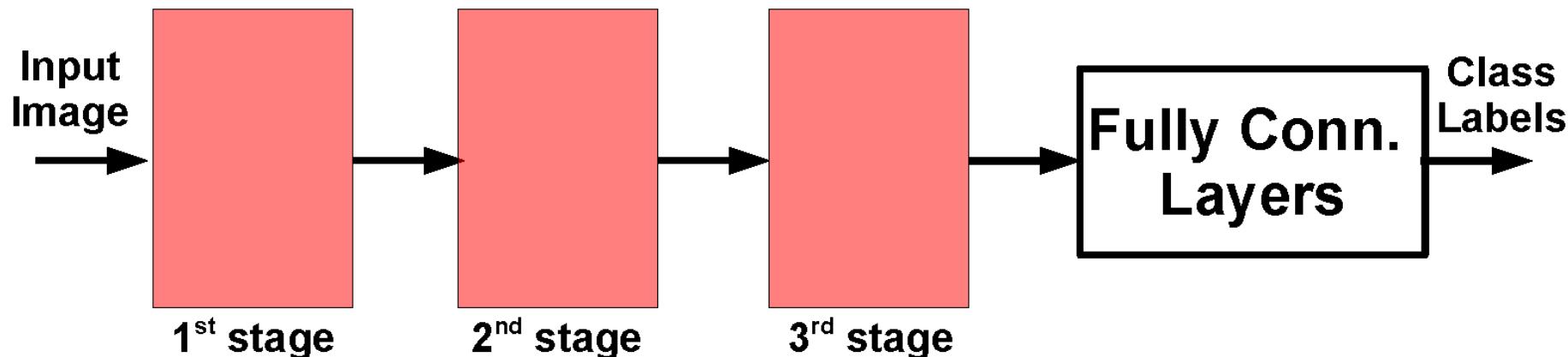


Whole system



ConvNets: Typical Architecture

Whole system



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 classification with F.V.: Theory and practice" IJCV 2012

ConvNets: Training

All layers are differentiable (a.e.).

We can use standard back-propagation.

Algorithm:

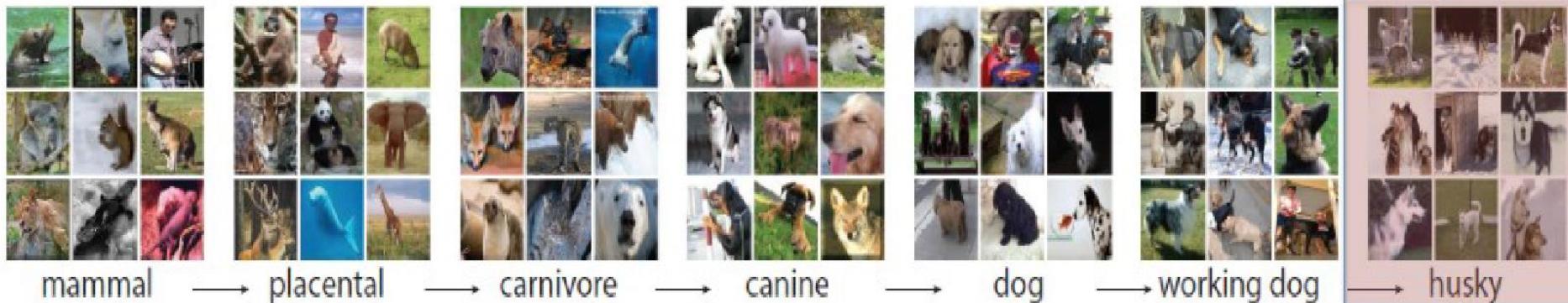
Given a small mini-batch

- F-PROP
- B-PROP
- PARAMETER UPDATE

Outline

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

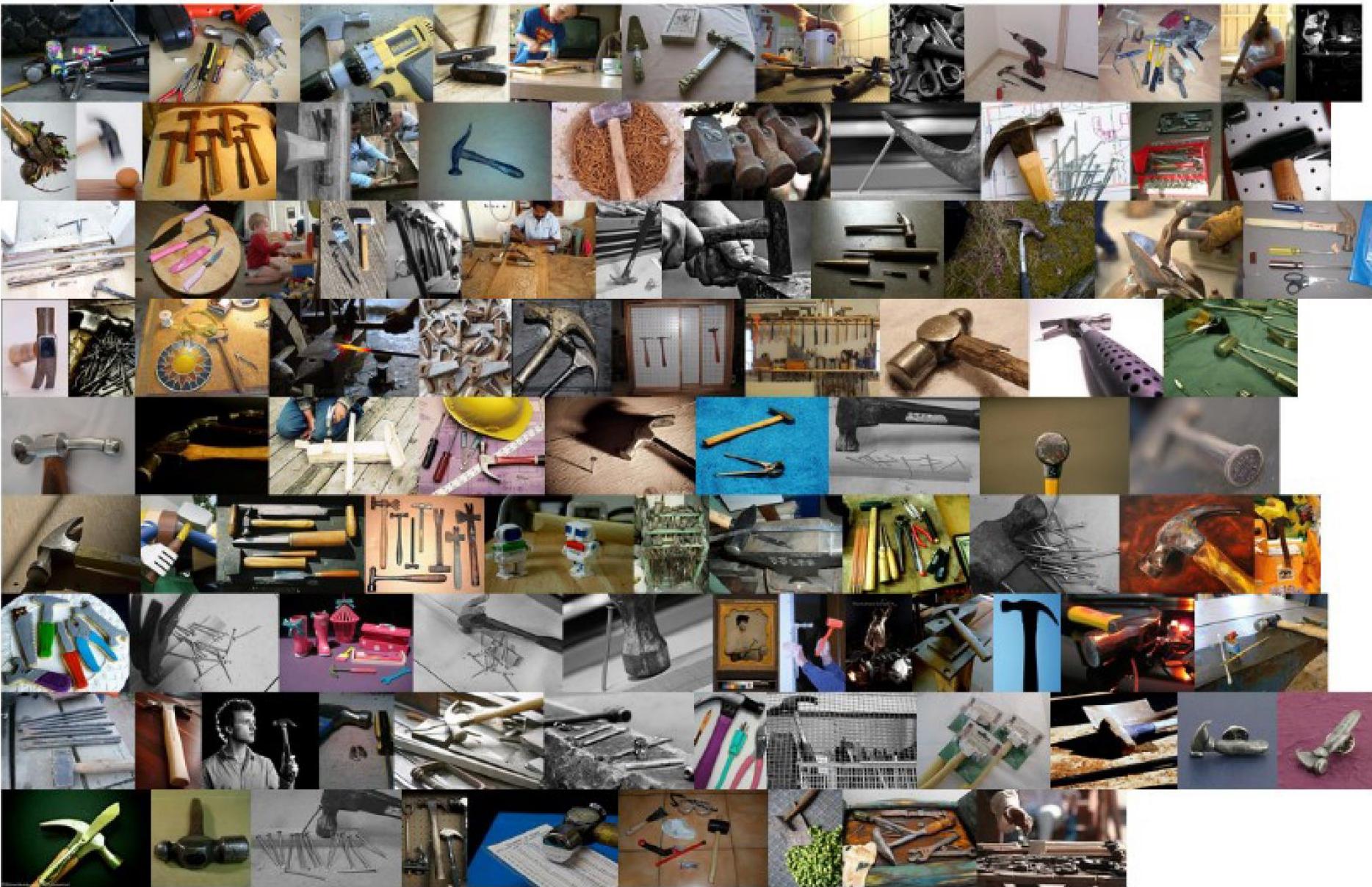
Dataset: ImageNet 2012



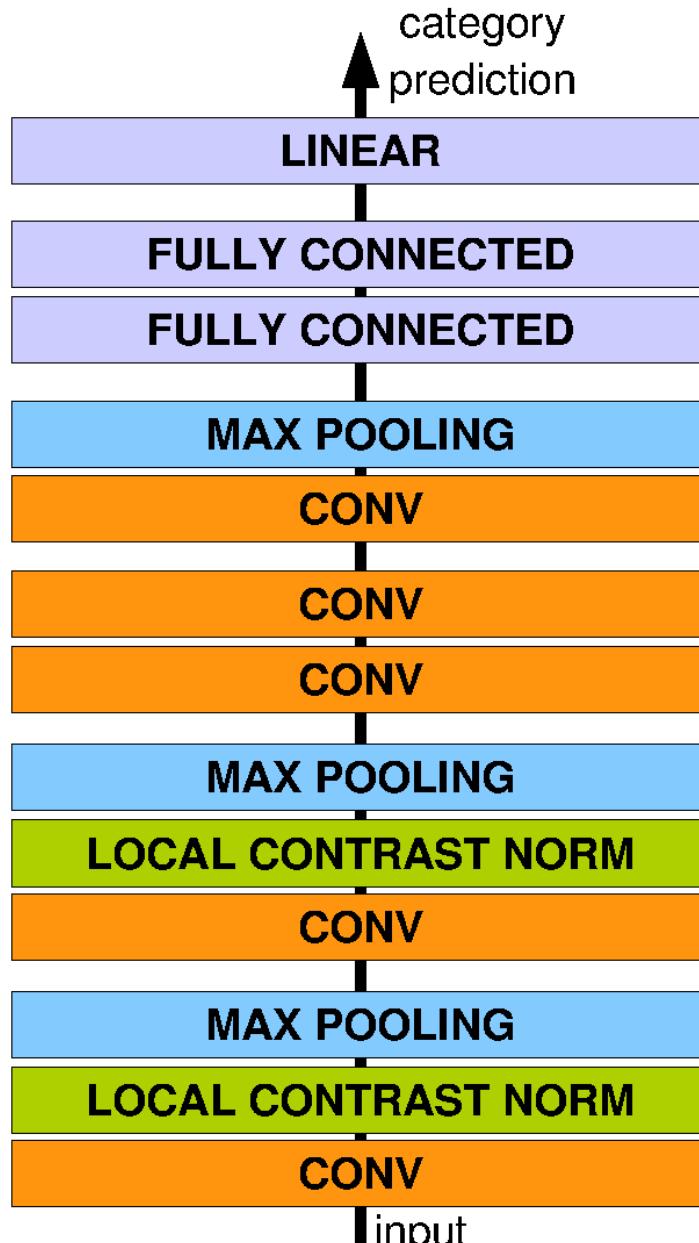
- S: (n) [Eskimo dog](#), [husky](#) (breed of heavy-coated Arctic sled dog)
 - [direct hypernym / inherited hypernym / sister term](#)
- S: (n) [working dog](#) (any of several breeds of usually large powerful dogs bred to work as draft animals and guard and guide dogs)
 - S: (n) [dog](#), [domestic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

ImageNet

Examples of hammer:



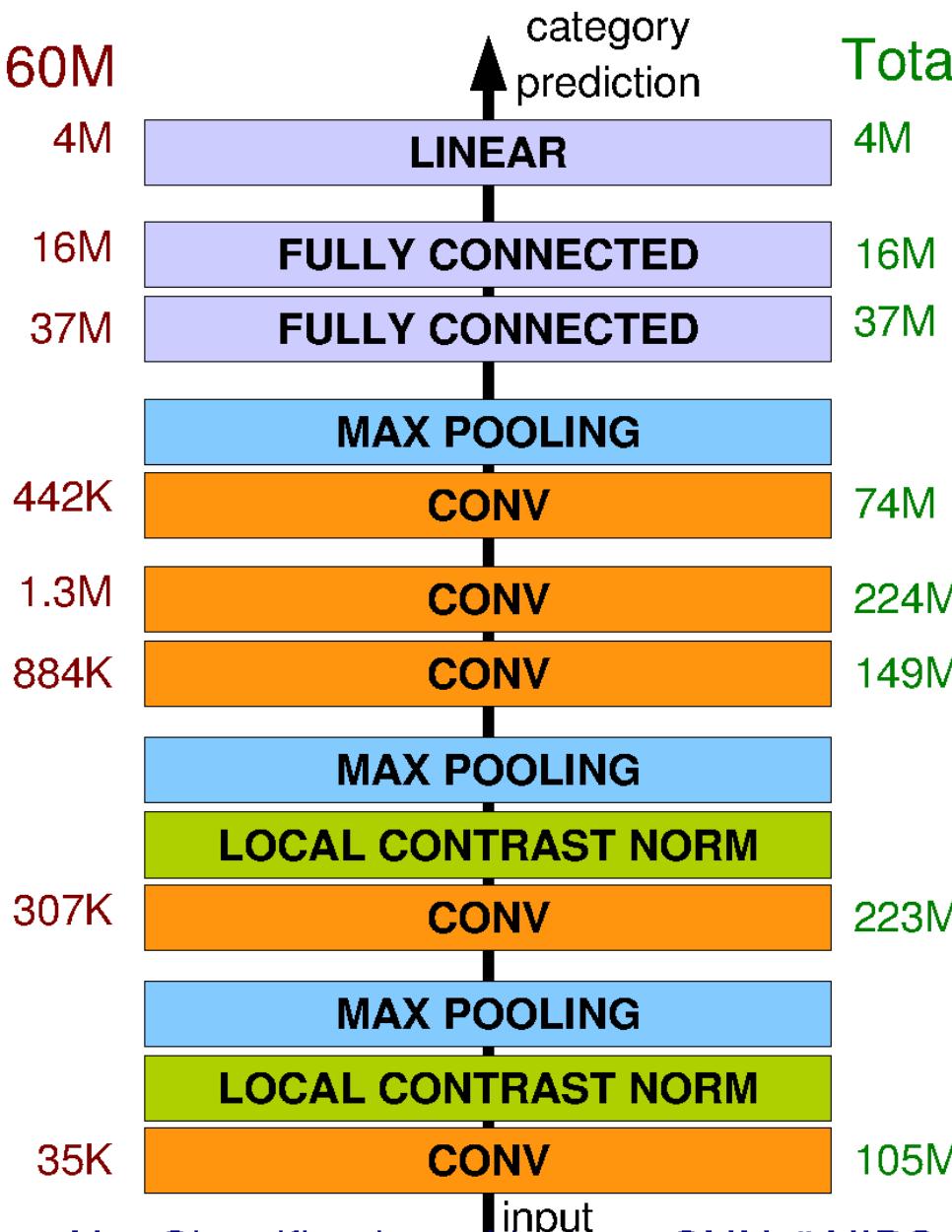
Architecture for Classification



Architecture for Classification

Total nr. params: 60M

Total nr. flops: 832M



Optimization

SGD with momentum:

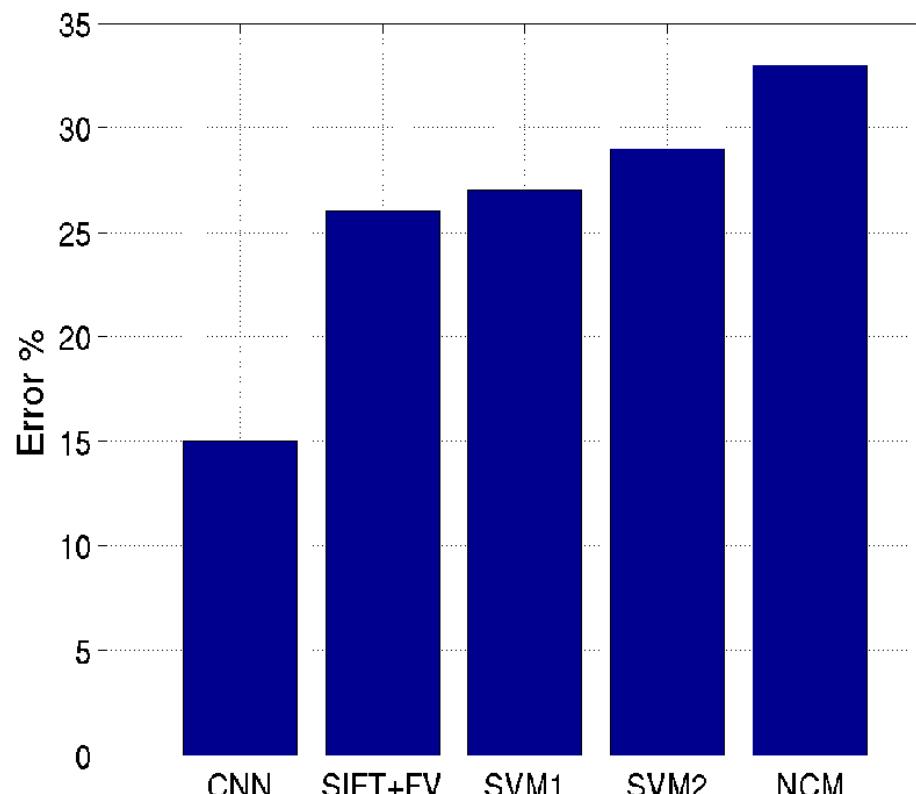
- Learning rate = 0.01
- Momentum = 0.9

Improving generalization by:

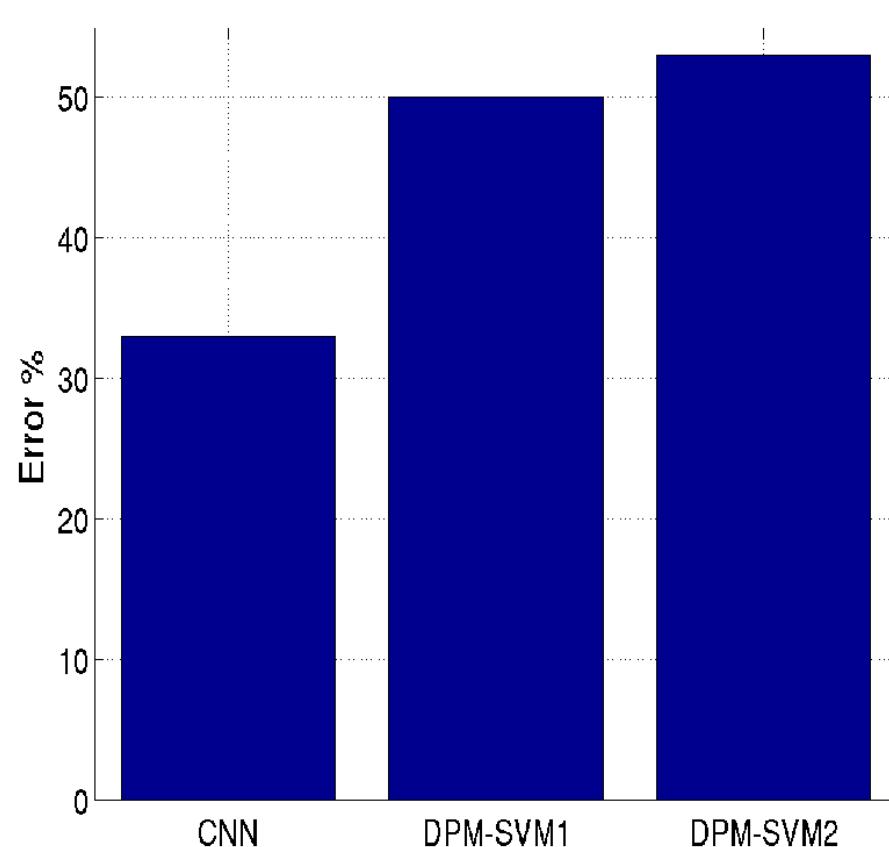
- Weight sharing (convolution)
- Input distortions
- Dropout = 0.5
- Weight decay = 0.0005

Results: ILSVRC 2012

TASK 1 - CLASSIFICATION



TASK 2 - DETECTION





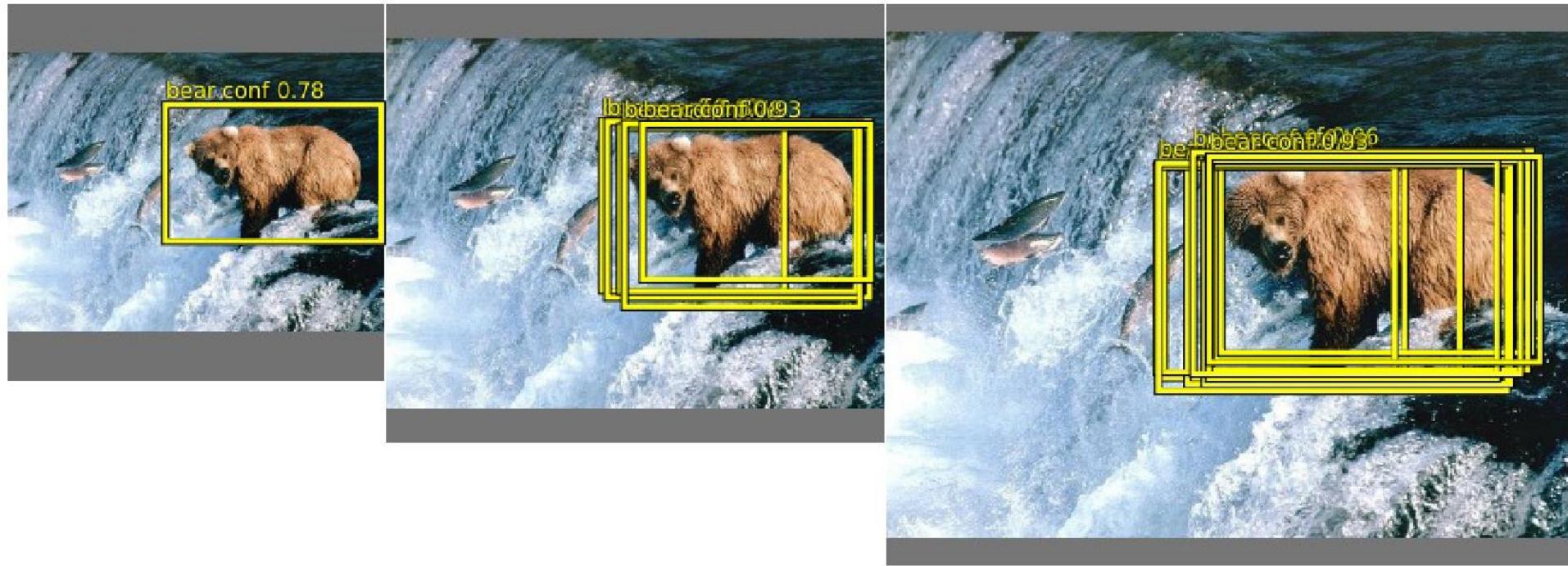
mite	container ship	motor scooter	leopard
black widow cockroach tick starfish	lifeboat amphibian fireboat drilling platform	go-kart moped bumper car golfcart	jaguar cheetah snow leopard Egyptian cat



convertible grille pickup beach wagon fire engine	agaric mushroom jelly fungus gill fungus dead-man's-fingers	dalmatian grape elderberry ffordshire bullterrier currant	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 91

Szegedy et al. “DNN for object detection” NIPS 2013

CONV NETS: EXAMPLES

- Face Verification & Identification

