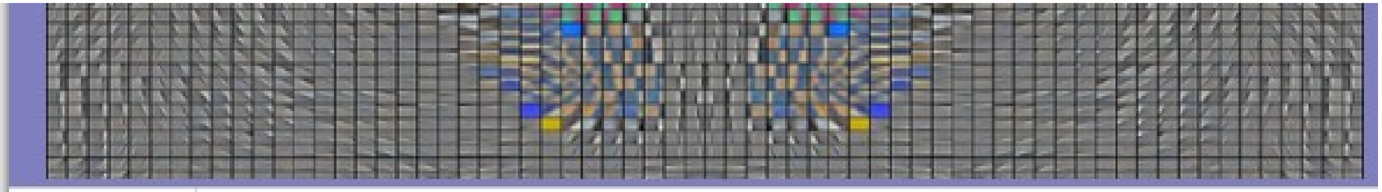
Image Classification with Deep Learning

Marc'Aurelio Ranzato

Facebook A.I. Research



22
days since
Tutorial Date

TUTORIAL ON DEEP LEARNING FOR VISION

A tutorial in conjunction with the Intl. Conference in Computer Vision (CVPR) 2014.
Monday June 23, 2014
Grand Ballroom 2
Columbus, Ohio

Schedule

Morning Session: foundations

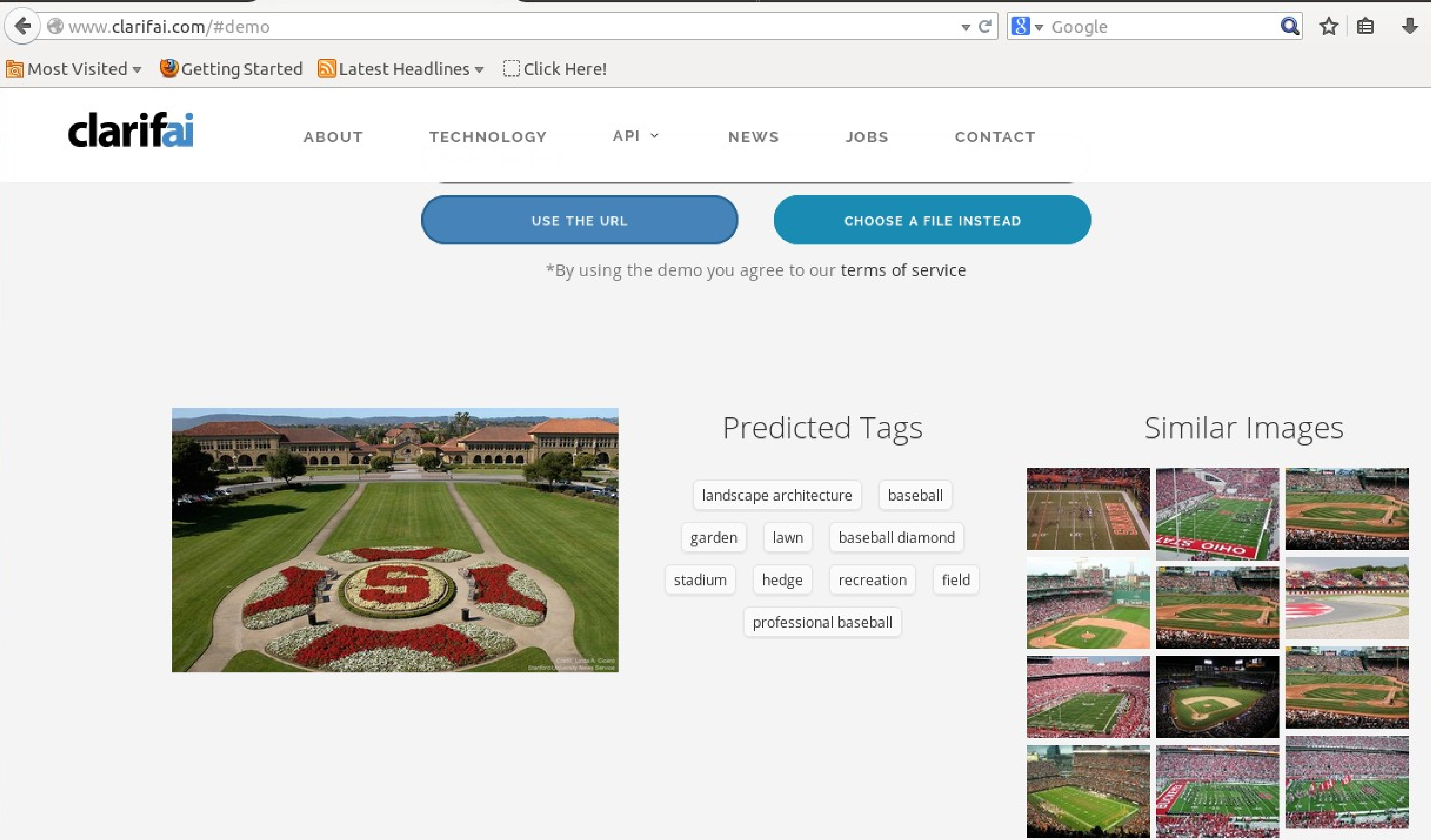
- 8.30-9.00 Introduction - Honglak Lee (University of Michigan)
- 9.00-10.00 Supervised learning - Marc'Aurelio Ranzato (Facebook A.I. Research)
- 10.00-10.30 Coffee Break
- 10.30-11.30 Unsupervised learning - Graham Taylor (University of Guelph)
- 11.30-12.30 Practical tools
    - Torch7 - Marc'Aurelio Ranzato (Facebook A.I. Research)
    - Theano/Pylearn2 - Presented by Ian Goodfellow (Univ. of Montreal)
    - Caffe - Presented by Yangqing Jia (Google Research). [additional material]

12.30-1.30 Lunch

Afternoon Session: advanced topics

- 1.30-2.15 Object detection - Pierre Sermanet (Google)
- 2.15-3.00 Regression Methods for Localization - Alex Toshev (Google Research)
- 3.00-3.30 Coffee Break
- 3.30-4.00 Large Scale Classification and GPU Parallelization - Alex Krizhevsky (Google)
- 4.00-4.30 Learning Transformations from videos - Roland Memisevic (Univ. of Montreal)
- 4.30-5.15 Multi-modal & Multi-task Learning - Honglak Lee (Univ. of Michigan)
- 5.15-6.00 Structured Prediction - Yann LeCun (NYU and Facebook A.I. Research)
- 6.00-6.15 Q&A

https://sites.google.com/site/deeplearningcvpr2014/



The methods we are going to talk about today are used by several companies for a variety of applications, such as classification, retrieval, detection, etc.

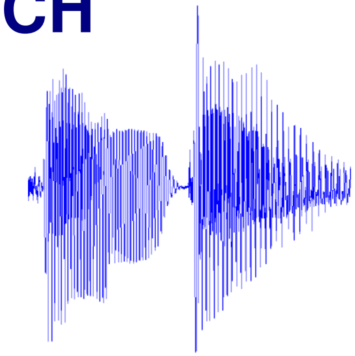
Traditional Pattern Recognition

VISION

→ SIFT/HOG → K-Means/ pooling → classifier → "car"

fixed | unsupervised | supervised

SPEECH



→ MFCC → Mixture of Gaussians → classifier → \ˈd ē p\

fixed | unsupervised | supervised

NLP

This burrito place is yummy and fun! → Parse Tree Syntactic → n-grams → classifier → "+"

fixed | unsupervised | supervised

Ranzato

Hierarchical Compositionality (DEEP)

VISION

pixels → edge → texton → motif → part → object

SPEECH

sample → spectral band → formant → motif → phone → word

NLP

character → word → NP/VP/.. → clause → sentence → story

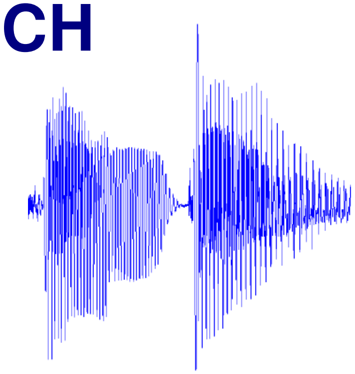
Traditional Pattern Recognition

VISION

SIFT/HOG → K-Means/ pooling → classifier → "car"

fixed | unsupervised | supervised

SPEECH



MFCC → Mixture of Gaussians → classifier → \ˈd ē p\

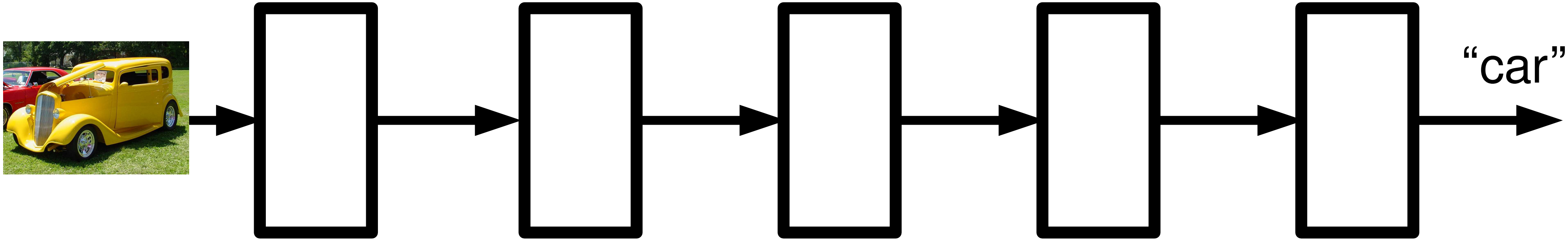
fixed | unsupervised | supervised

NLP

This burrito place is yummy and fun! → Parse Tree Syntactic → n-grams → classifier → "+"

fixed | unsupervised | supervised

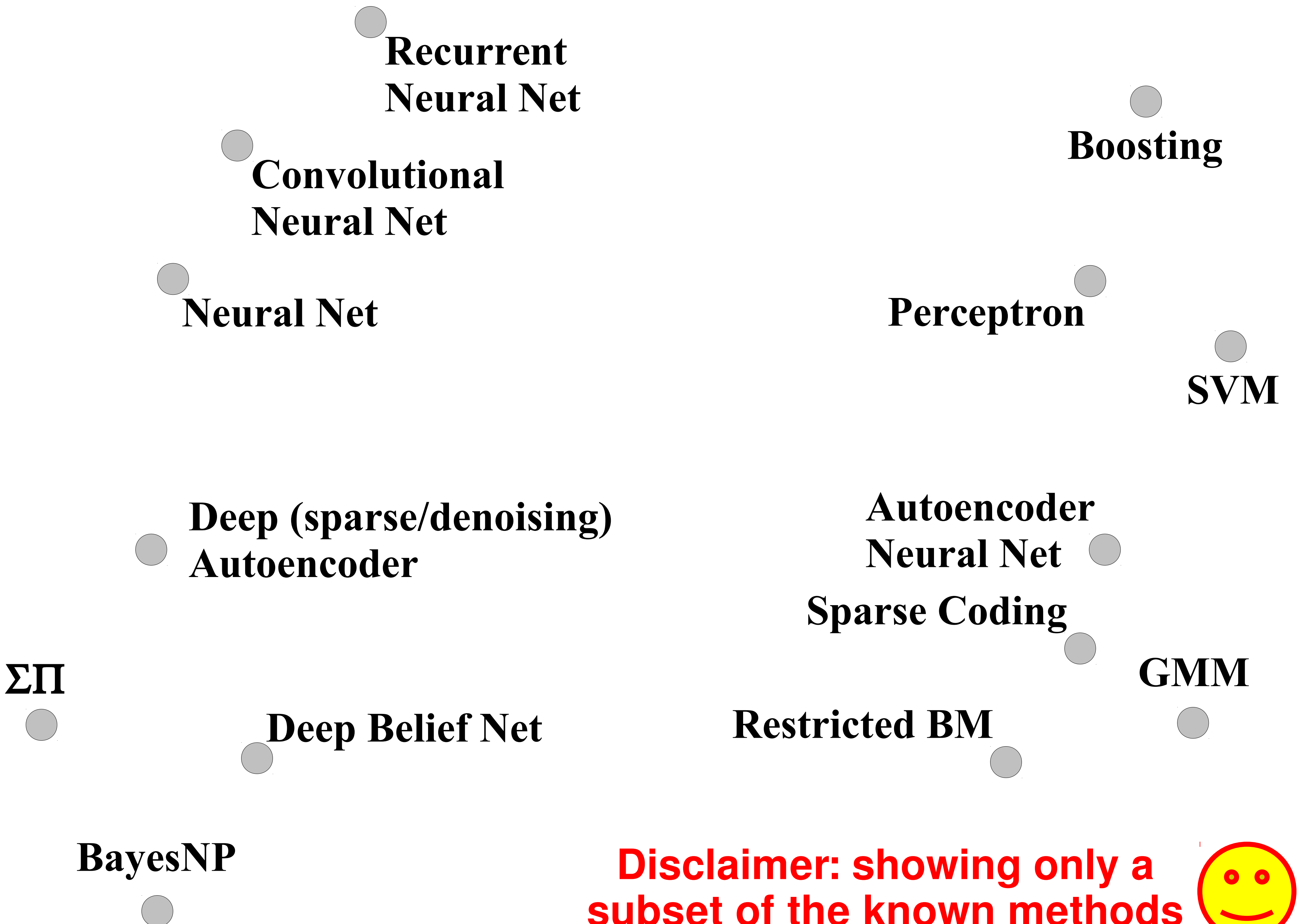
Deep Learning

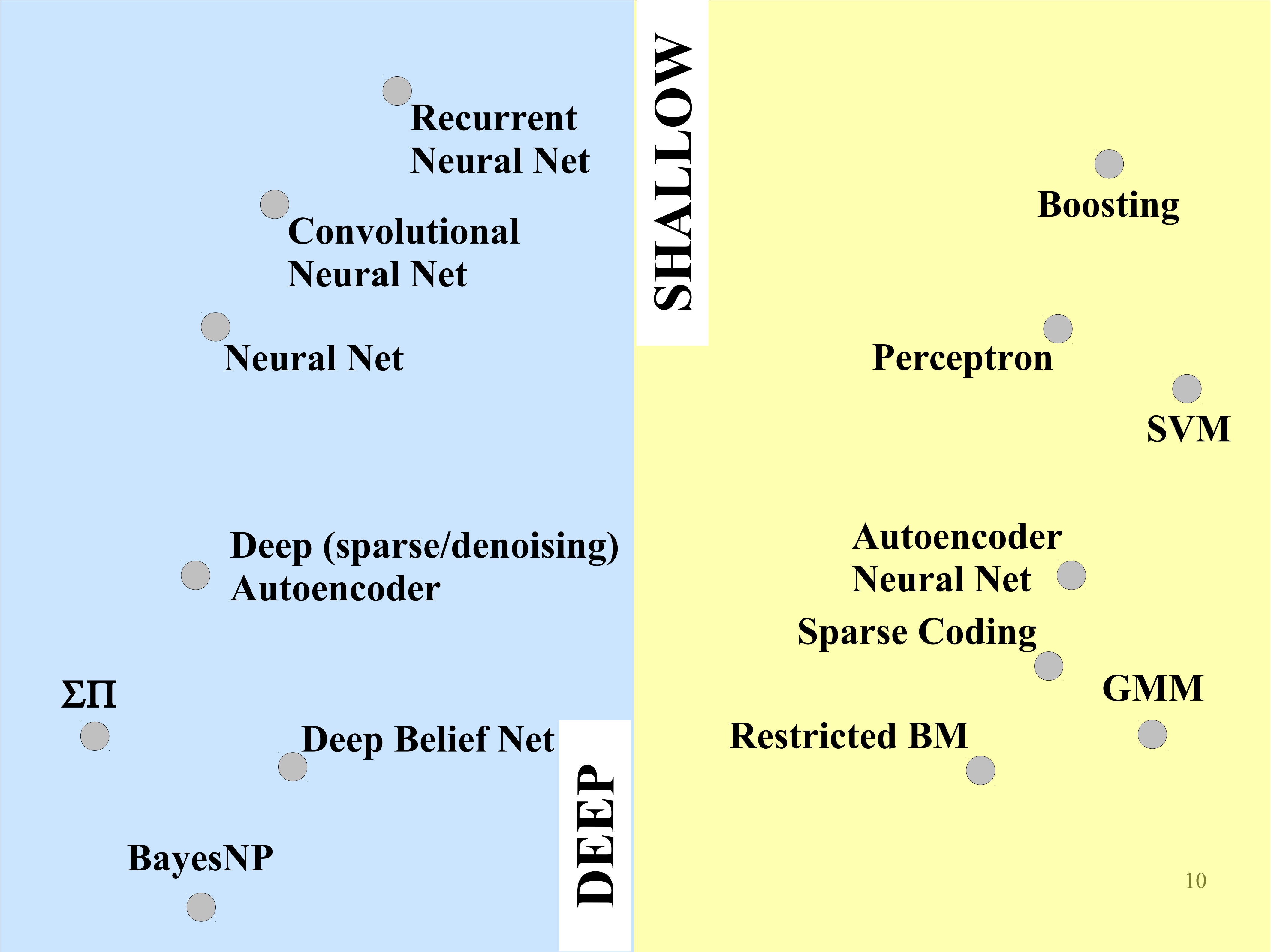


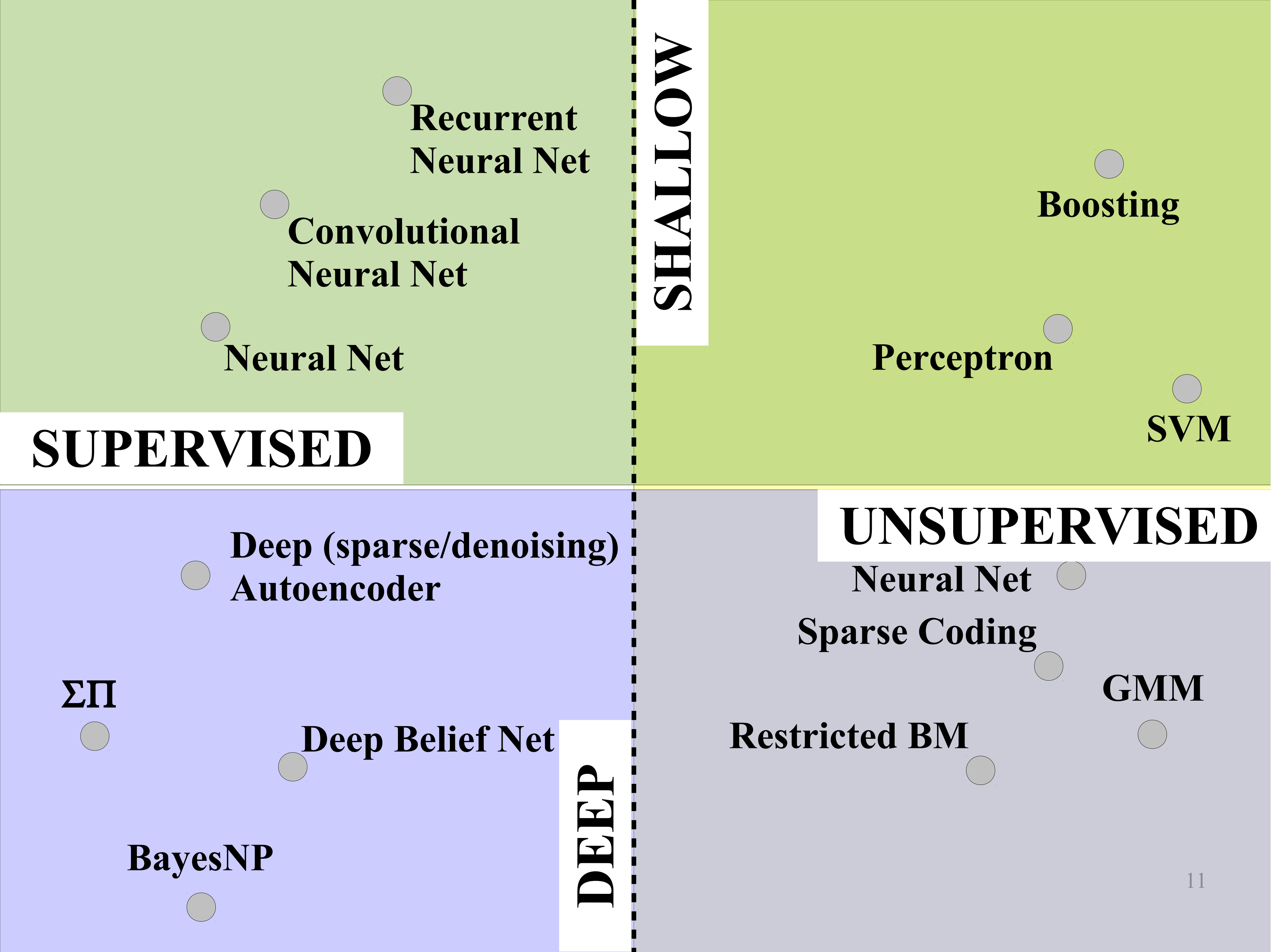
What is Deep Learning

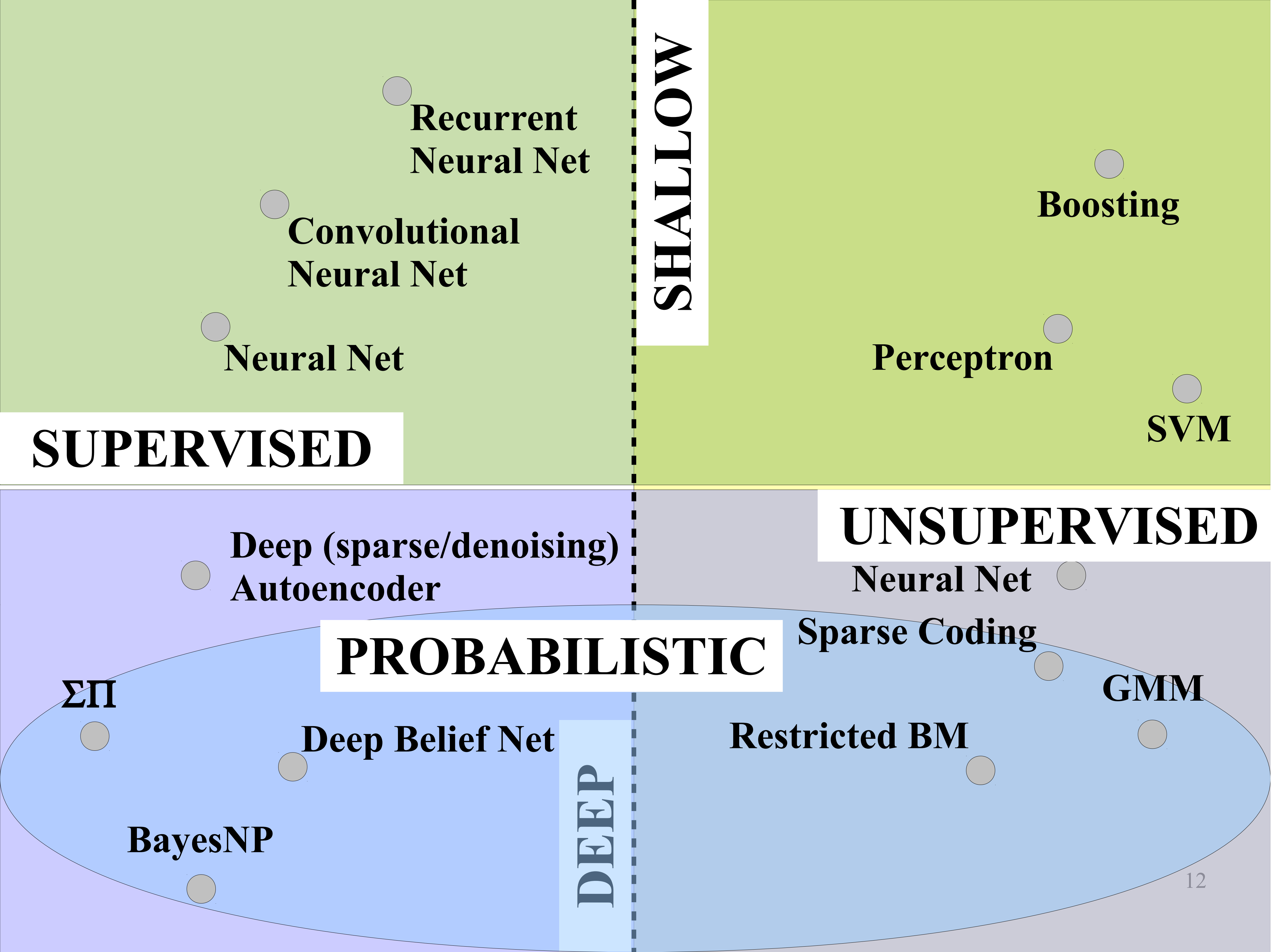
- Cascade of non-linear transformations
- End to end learning
- General framework (any hierarchical model is deep)

THE SPACE OF MACHINE LEARNING METHODS


Recurrent Neural Net
Boosting
Convolutional Neural Net
Neural Net
Perceptron
SVM
Deep (sparse/denoising) Autoencoder
Autoencoder Neural Net
Sparse Coding
ΣΠ
GMM
Deep Belief Net
Restricted BM
BayesNP
Disclaimer: showing only a subset of the known methods


SHALLOW
DEEP
Recurrent Neural Net
Convolutional Neural Net
Neural Net
Deep (sparse/denoising) Autoencoder
ΣΠ
Deep Belief Net
BayesNP
Boosting
Perceptron
SVM
Autoencoder Neural Net
Sparse Coding
GMM
Restricted BM


SHALLOW
DEEP
SUPERVISED
UNSUPERVISED
Recurrent Neural Net
Convolutional Neural Net
Neural Net
Boosting
Perceptron
SVM
Deep (sparse/denoising) Autoencoder
ΣΠ
Deep Belief Net
BayesNP
Neural Net
Sparse Coding
GMM
Restricted BM

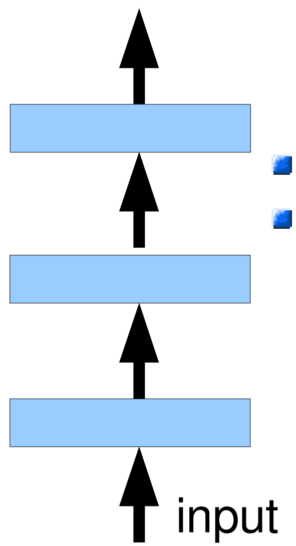


SHALLOW
Recurrent
Neural Net
Convolutional
Neural Net
Neural Net
Boosting
Perceptron
SVM
SUPERVISED
UNSUPERVISED
Deep (sparse/denoising)
Autoencoder
Neural Net
Sparse Coding
PROBABILISTIC
ΣΠ
GMM
Deep Belief Net
Restricted BM
BayesNP
DEEP

Deep Learning is B I G

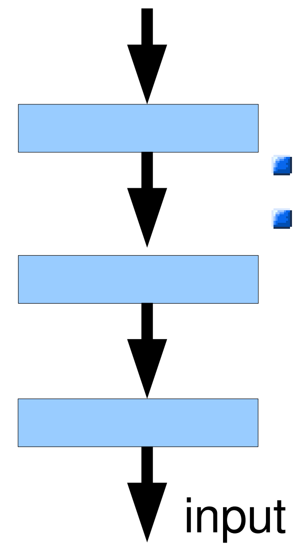
- Main types of deep architectures

feed-forward



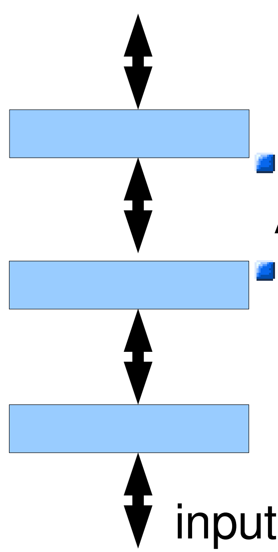
- Neural nets
- Conv Nets

Feed-back



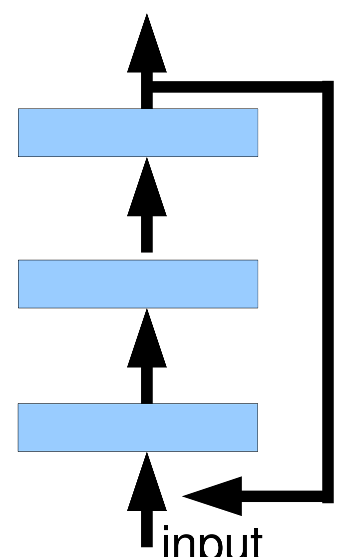
- Hierar. Sparse Coding
- Deconv Nets

Bi-directional



- Stacked Auto-encoders
- DBM

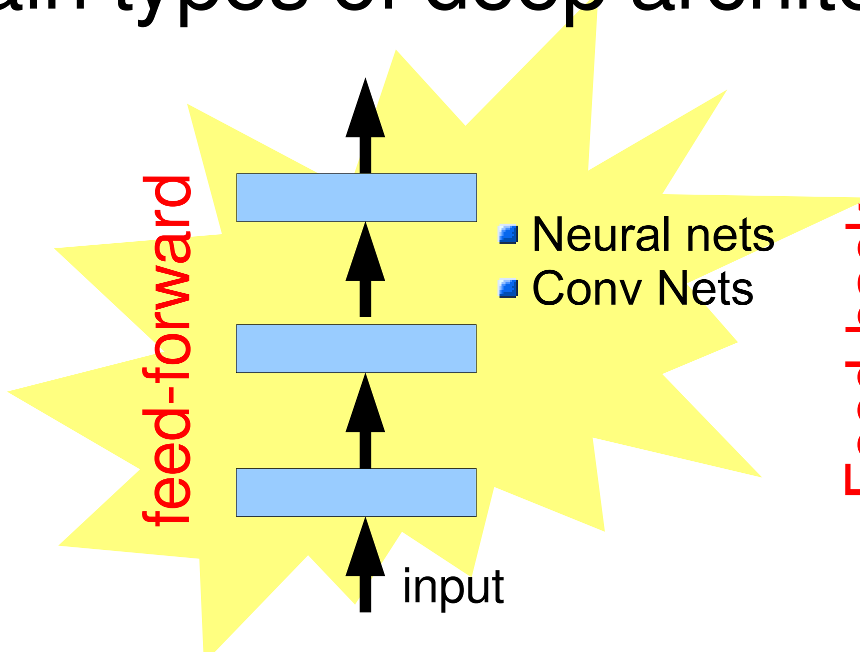
Recurrent



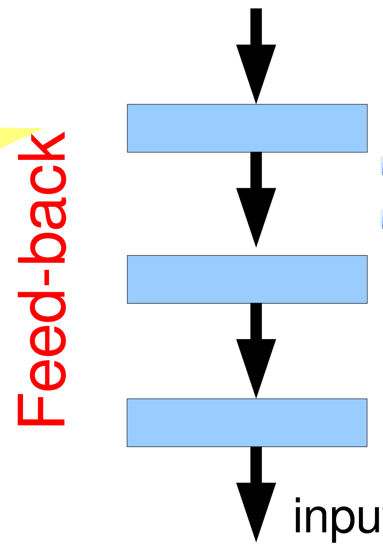
- Recurrent Neural nets
- Recursive Nets
- LISTA

Deep Learning is B I G

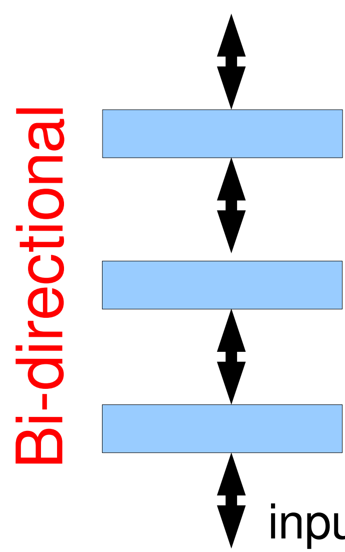
- Main types of deep architectures



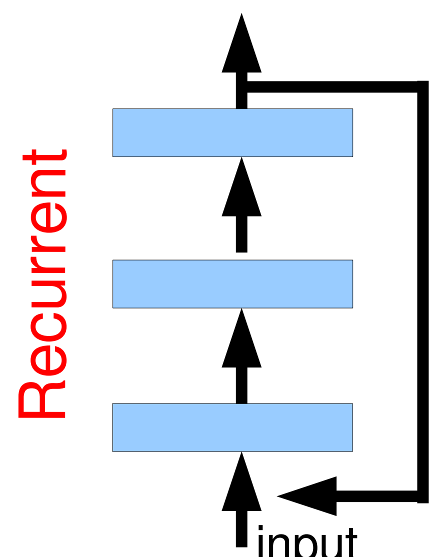
- Neural nets
- Conv Nets



- Hierar. Sparse Coding
- Deconv Nets



- Stacked Auto-encoders
- DBM



- Recurrent Neural nets
- Recursive Nets
- LISTA

Deep Learning is B I G

- Main types of learning protocols
  - Purely supervised
    - Backprop + SGD
    - Good when there is lots of labeled data.
  - Layer-wise unsupervised + superv. linear classifier
    - Train each layer in sequence using regularized auto-encoders or RBMs
    - Hold fix the feature extractor, train linear classifier on features
    - Good when labeled data is scarce but there is lots of unlabeled data.
  - Layer-wise unsupervised + supervised backprop
    - Train each layer in sequence
    - Backprop through the whole system
    - Good when learning problem is very difficult.

Deep Learning is B I G

- Main types of learning protocols
  - Purely supervised
    - Backprop + SGD
    - Good when there is lots of labeled data.
  - Layer-wise unsupervised + superv. linear classifier
    - Train each layer in sequence using regularized auto-encoders or RBMs
    - Hold fix the feature extractor, train linear classifier on features
    - Good when labeled data is scarce but there is lots of unlabeled data.
  - Layer-wise unsupervised + supervised backprop
    - Train each layer in sequence
    - Backprop through the whole system
    - Good when learning problem is very difficult.

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)

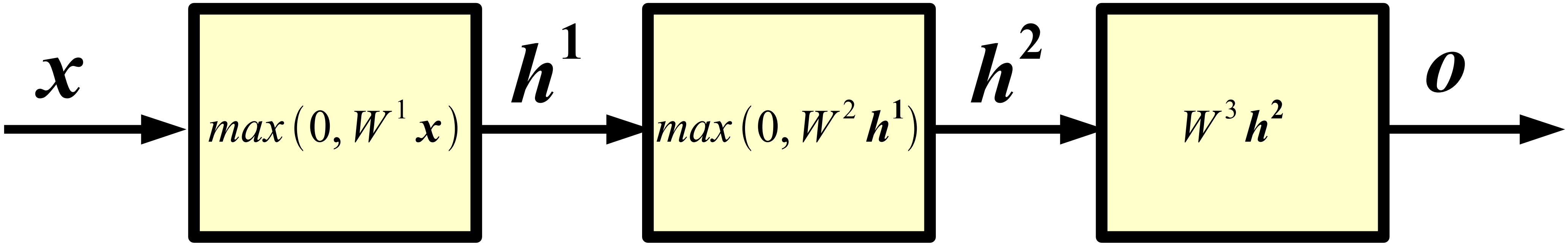
Question: what class of functions shall we consider to map the input into the output?

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



$\boldsymbol{x}$ input

$\boldsymbol{h}^1$ 1-st layer hidden units

$\boldsymbol{h}^2$ 2-nd layer hidden units

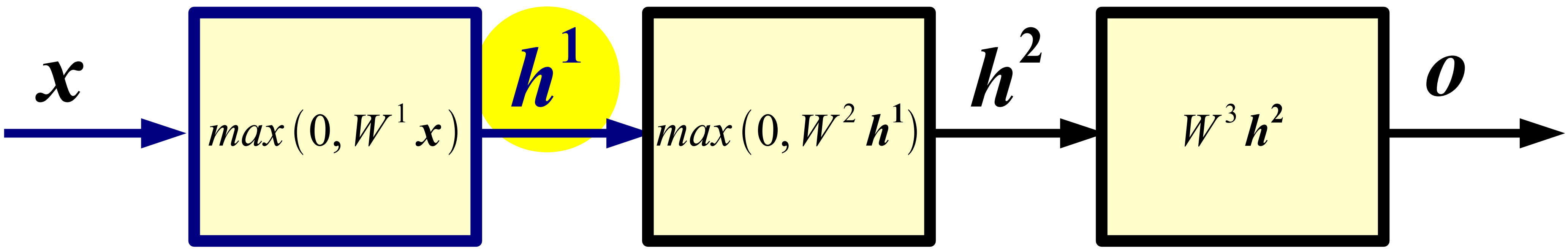
$\boldsymbol{o}$ output

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



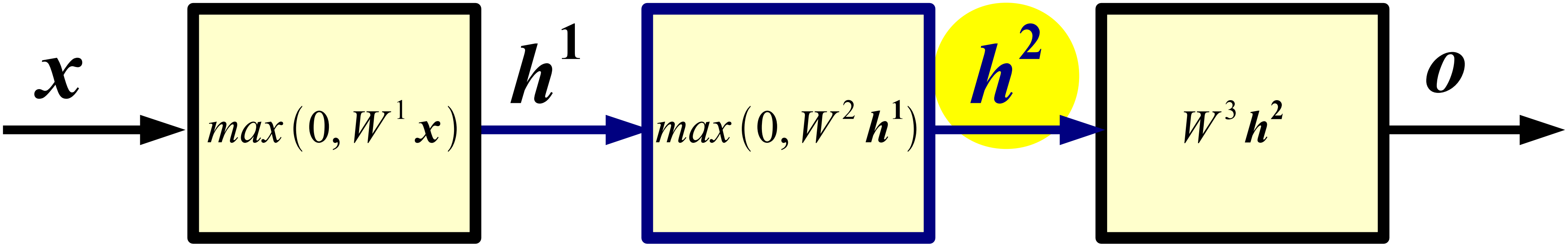
$\boldsymbol{x} \in R^D \quad W^1 \in R^{N_1 \times D} \quad \boldsymbol{b}^1 \in R^{N_1} \quad \boldsymbol{h}^1 \in R^{N_1}$

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

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

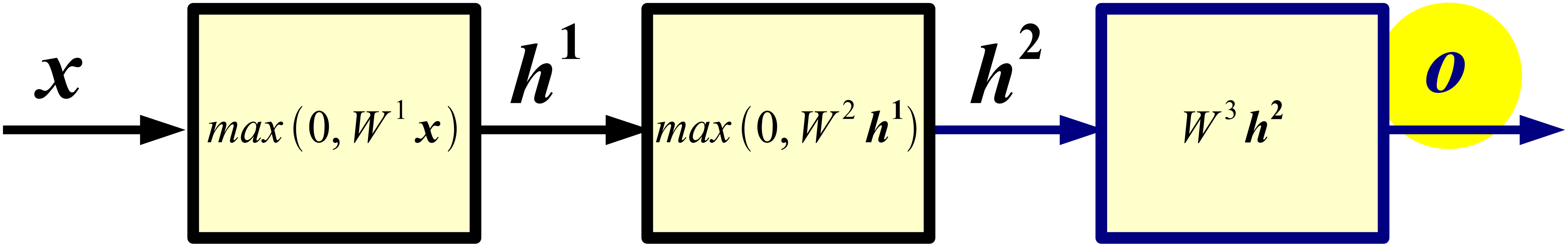


$\boldsymbol{h}^1 \in R^{N_1}$ $\quad W^2 \in R^{N_2 \times N_1}$ $\quad \boldsymbol{b}^2 \in R^{N_2}$ $\quad \boldsymbol{h}^2 \in R^{N_2}$

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

W^2 2-nd layer weight matrix or weights
$\boldsymbol{b}^2$ 2-nd layer biases

Forward Propagation



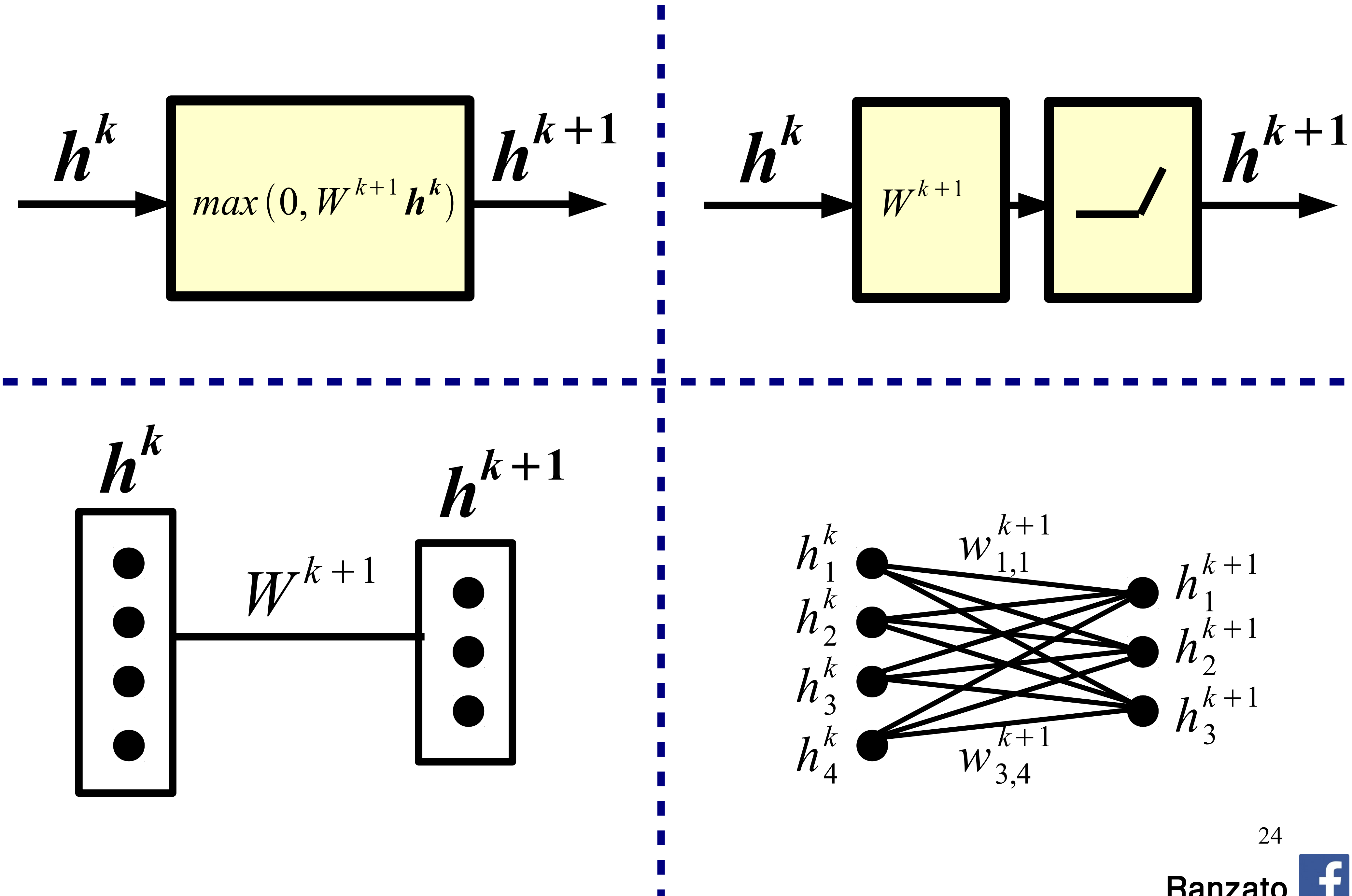
$\boldsymbol{h}^2 \in R^{N_2}$ $\quad W^3 \in R^{N_3 \times N_2}$ $\quad \boldsymbol{b}^3 \in R^{N_3}$ $\quad \boldsymbol{o} \in R^{N_3}$

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

W^3 3-rd layer weight matrix or weights

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

Alternative Graphical Representation



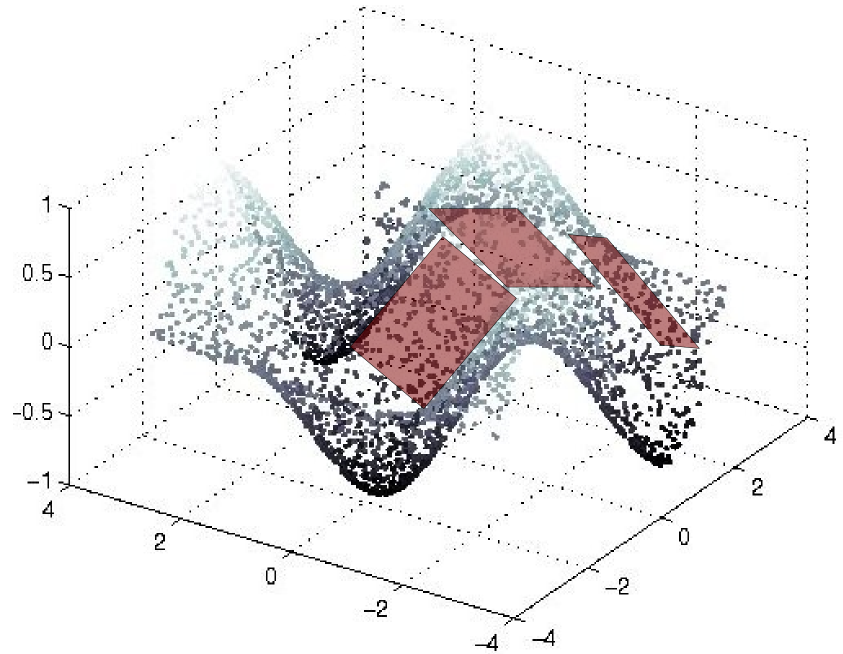
Interpretation

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

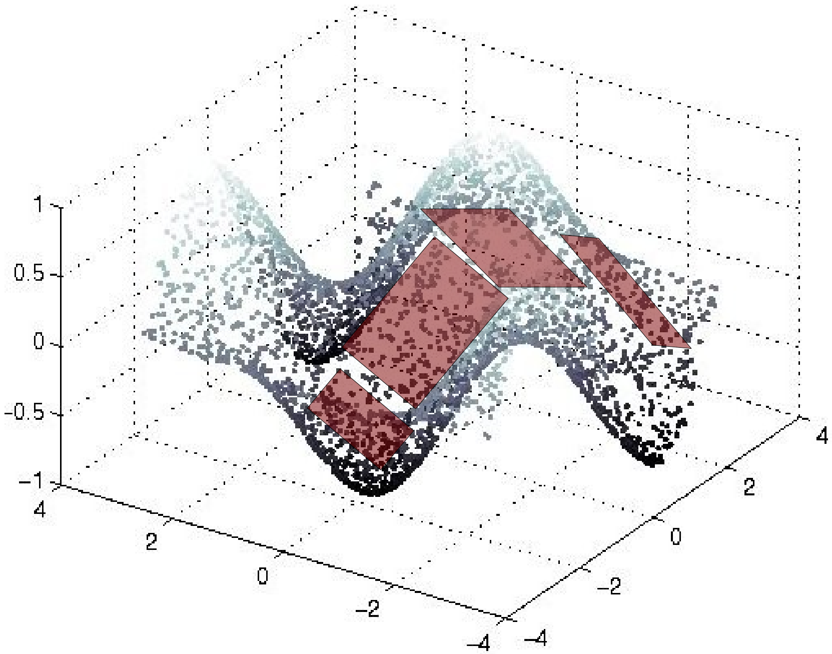
Answer: Because composition of linear functions is a linear function. Neural network would reduce to (1 layer) logistic regression.

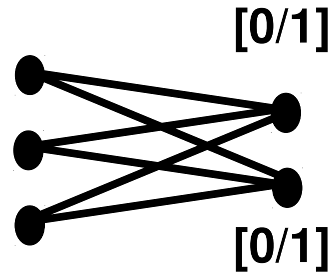
Question: What do ReLU layers accomplish?

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

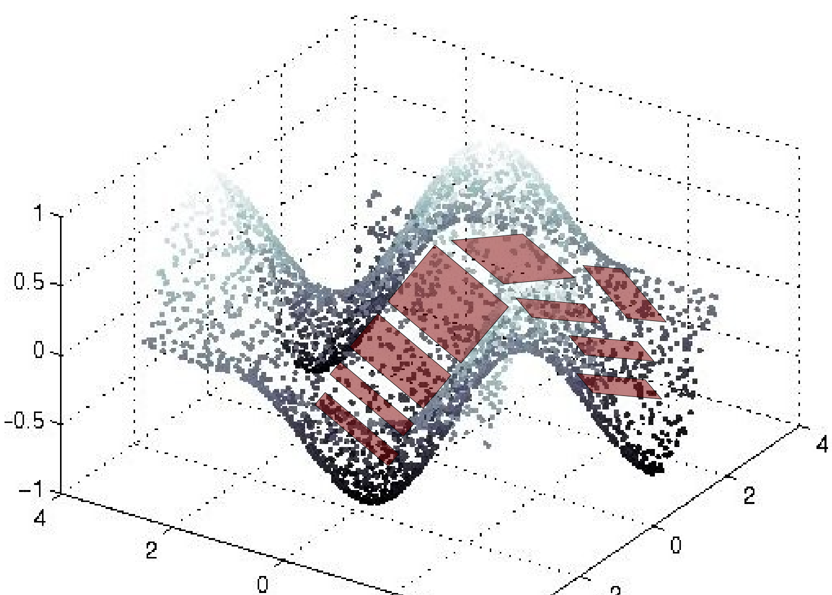


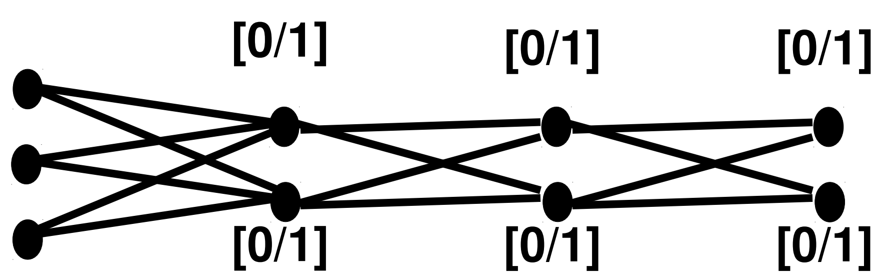
Montufar et al. "On the number of linear regions of DNNs" arXiv 2014





ReLU layers do local linear approximation. Number of planes grows exponentially with number of hidden units. Multiple layers yeild exponential savings in number of parameters (parameter sharing).





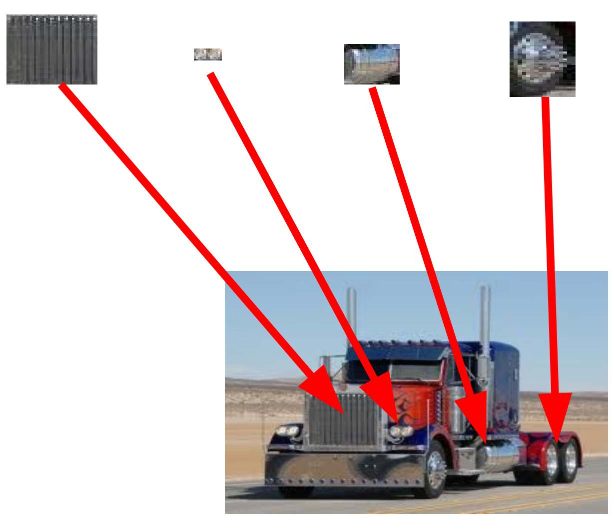
Montufar et al. "On the number of linear regions of DNNs" arXiv 2014

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

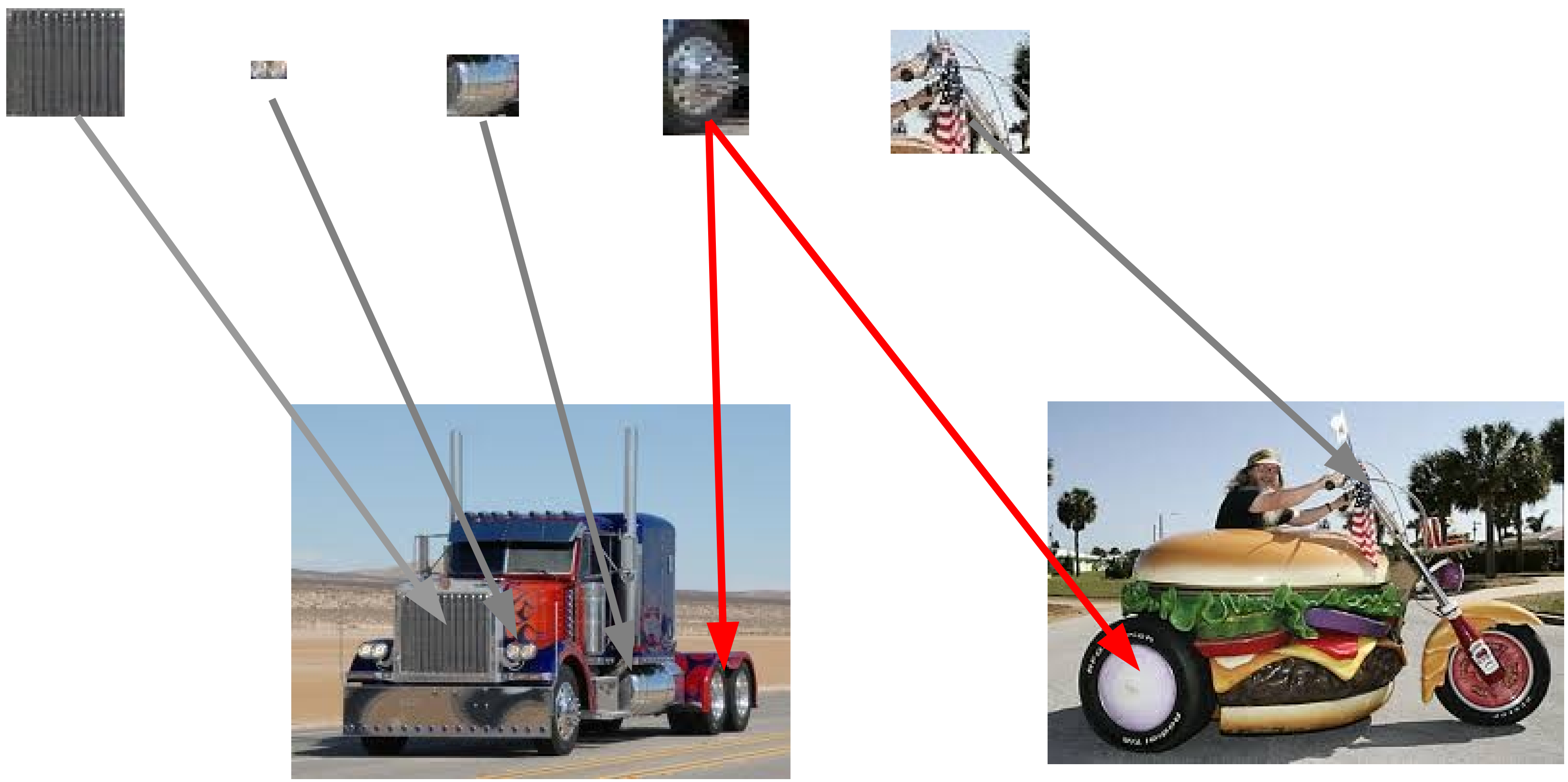


Exponentially more efficient than a 1-of-N representation (a la k-means)

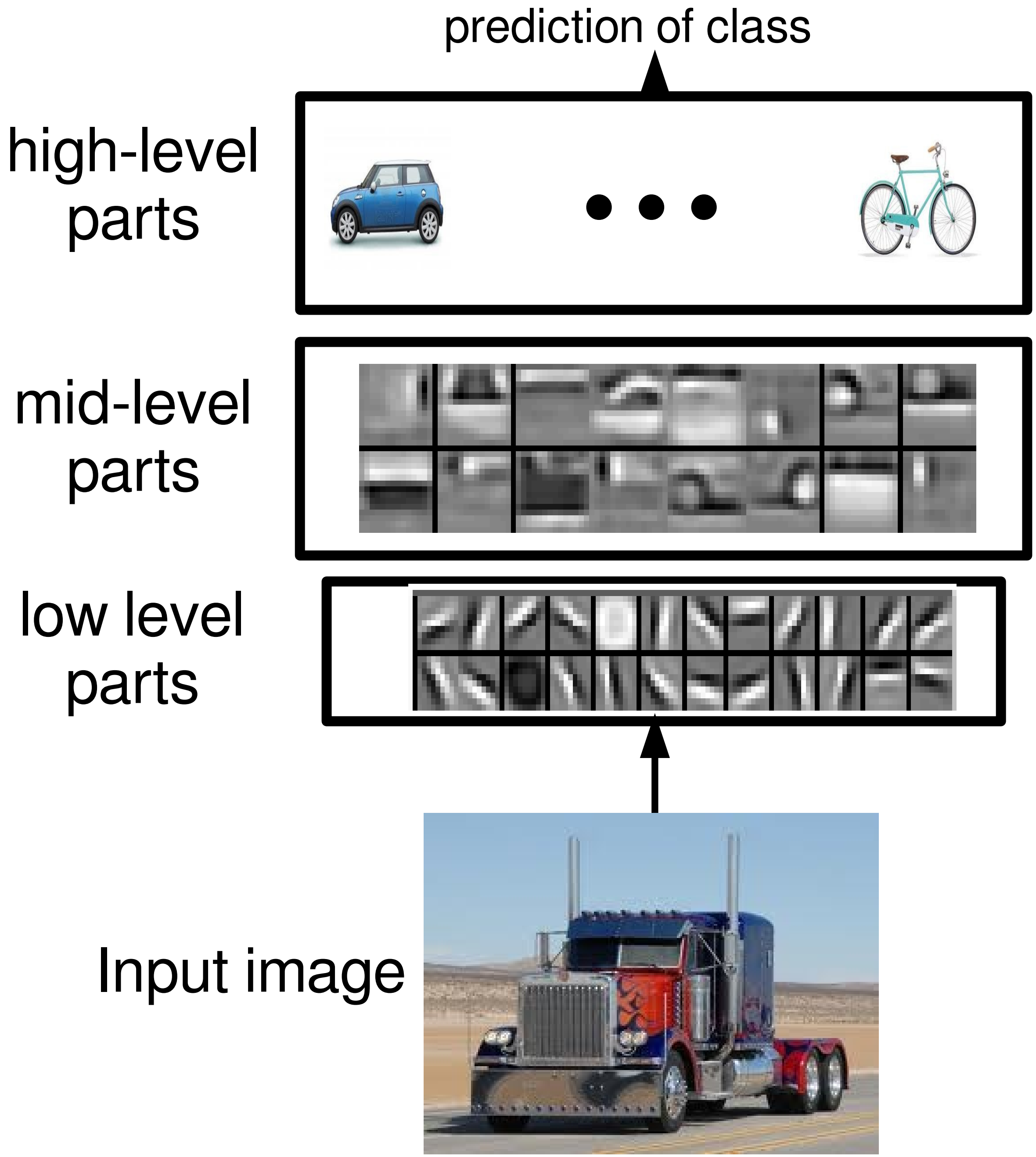
Interpretation

[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



- distributed representations
- feature sharing
- compositionality

Lee et al. “Convolutional DBN's ...” ICML 2009

Interpretation

Question: What does a hidden unit do?

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

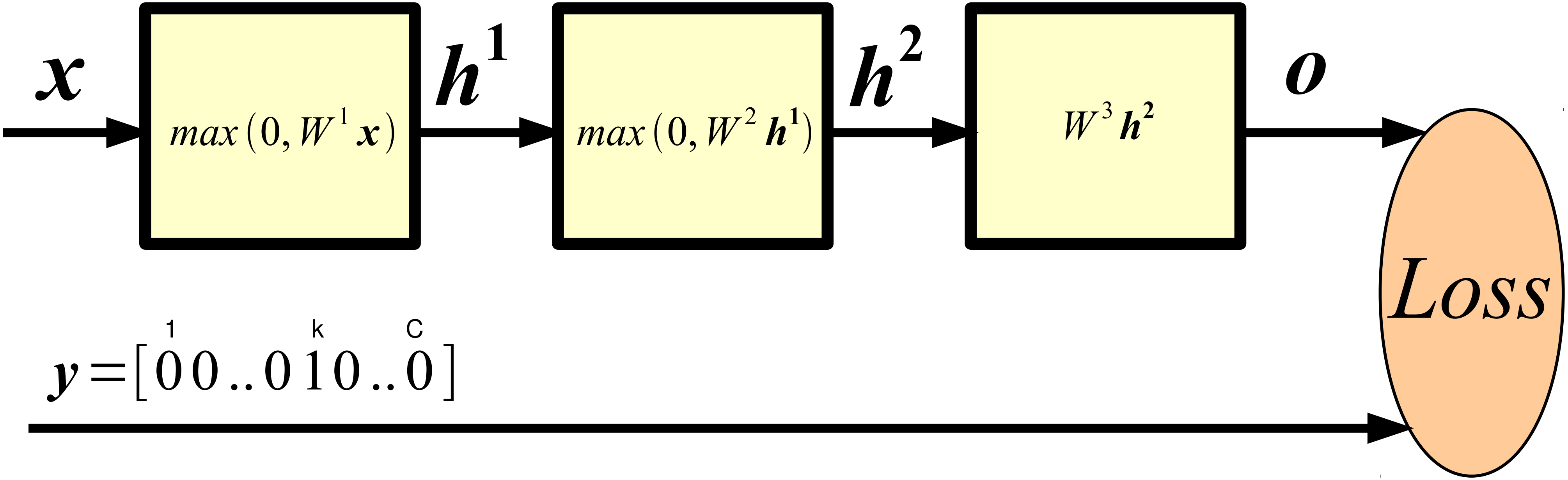
Question: How many layers? How many hidden units?

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):

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

(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.

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

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

Derivative w.r.t. Input of Softmax

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

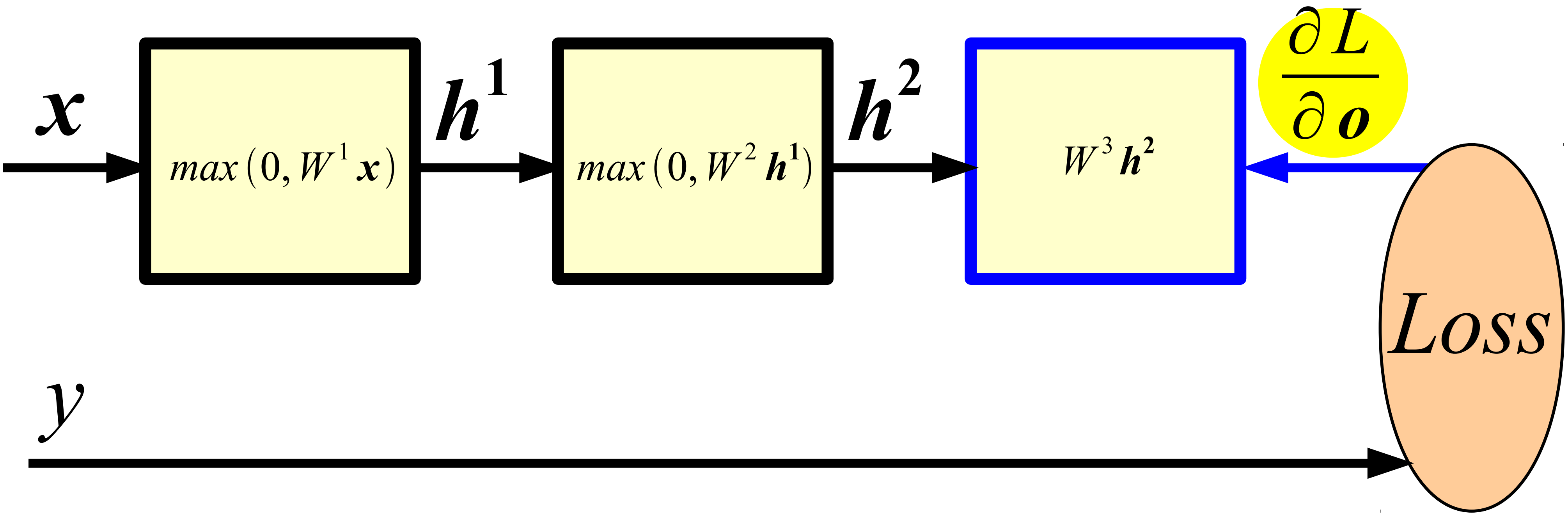
$$L(\boldsymbol{x}, y; \boldsymbol{\theta}) = -\sum_j y_j \log p(c_j | \boldsymbol{x}) \qquad \boldsymbol{y} = [\overset{1}{0}\, 0 \,..\, 0\, \overset{k}{1}\, 0 \,..\, \overset{C}{0}]$$

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

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

HOMEWORK: prove it!

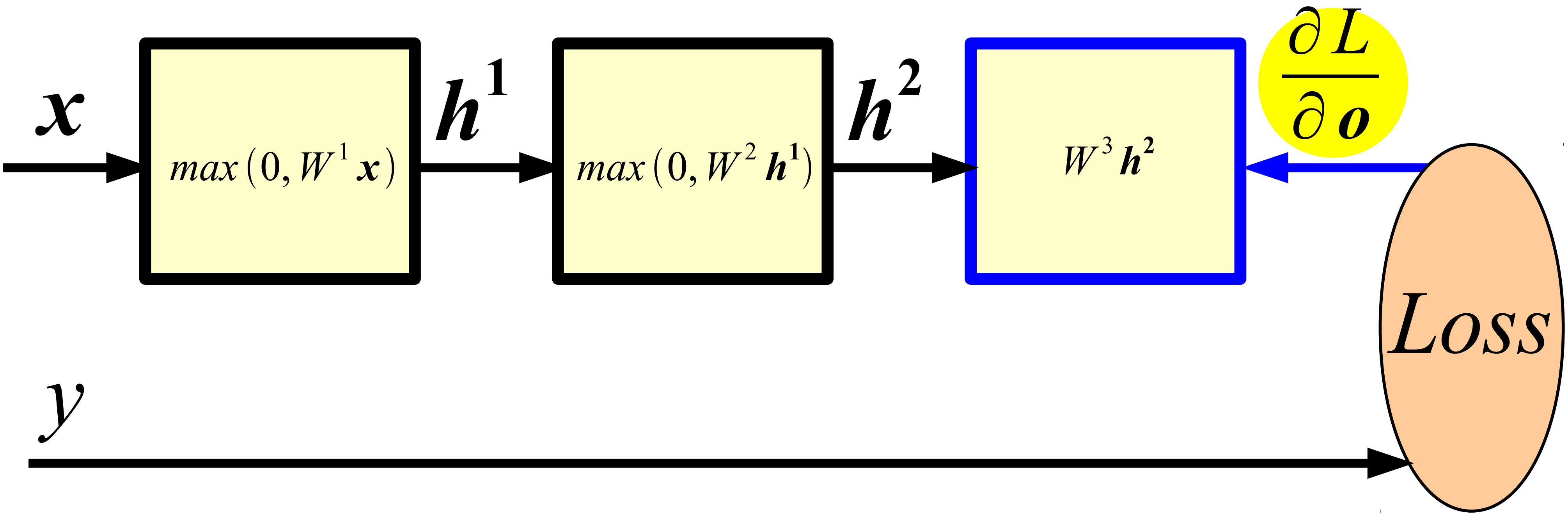
Backward Propagation



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

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

Backward Propagation

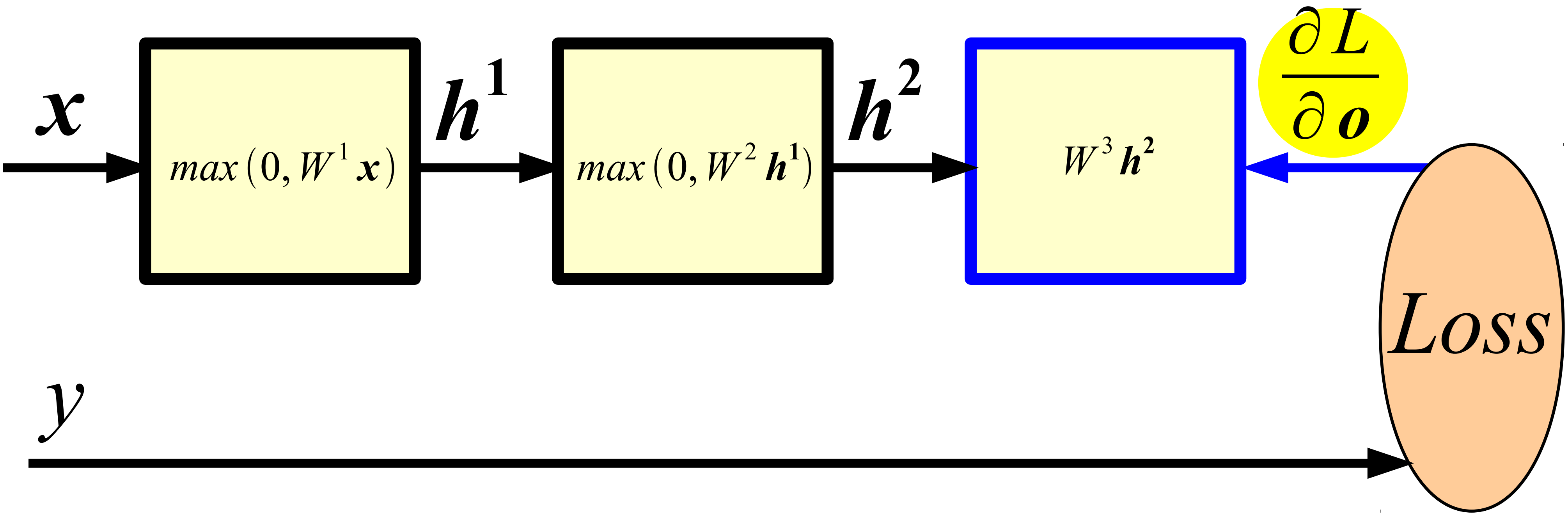


Given $\partial L / \partial \boldsymbol{o}$ and assuming we can easily compute the Jacobian of each module, we have:

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

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

Backward Propagation

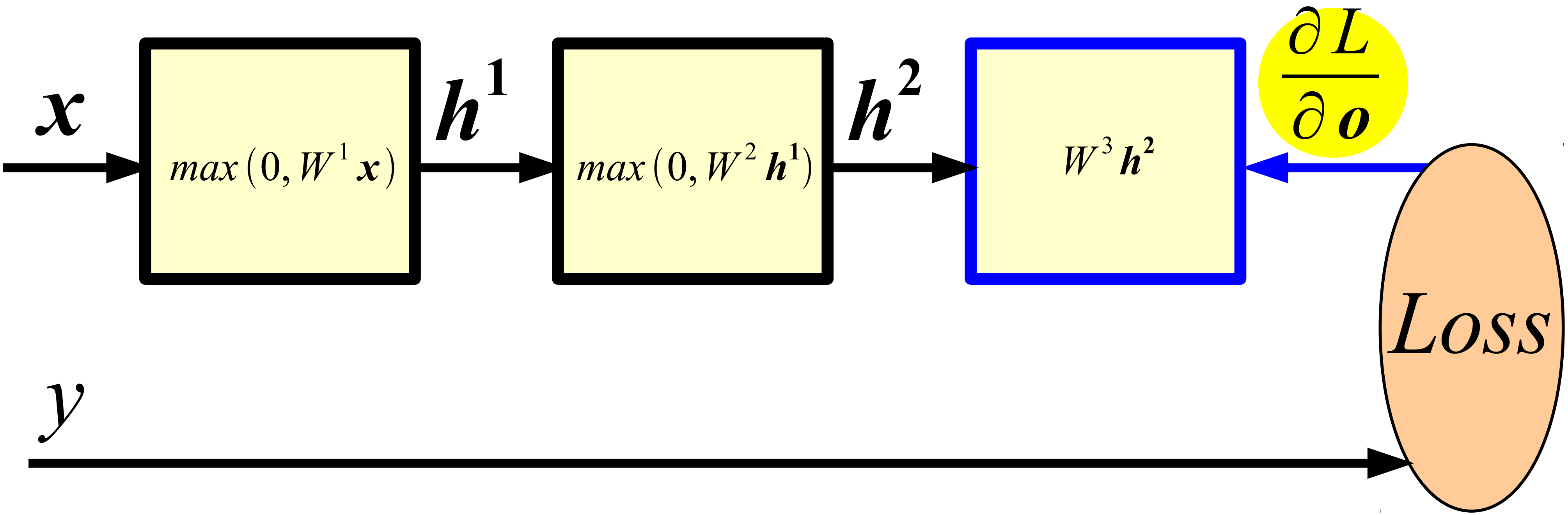


Given $\partial L / \partial \boldsymbol{o}$ and assuming we can easily compute the Jacobian of each module, we have:

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

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

Backward Propagation

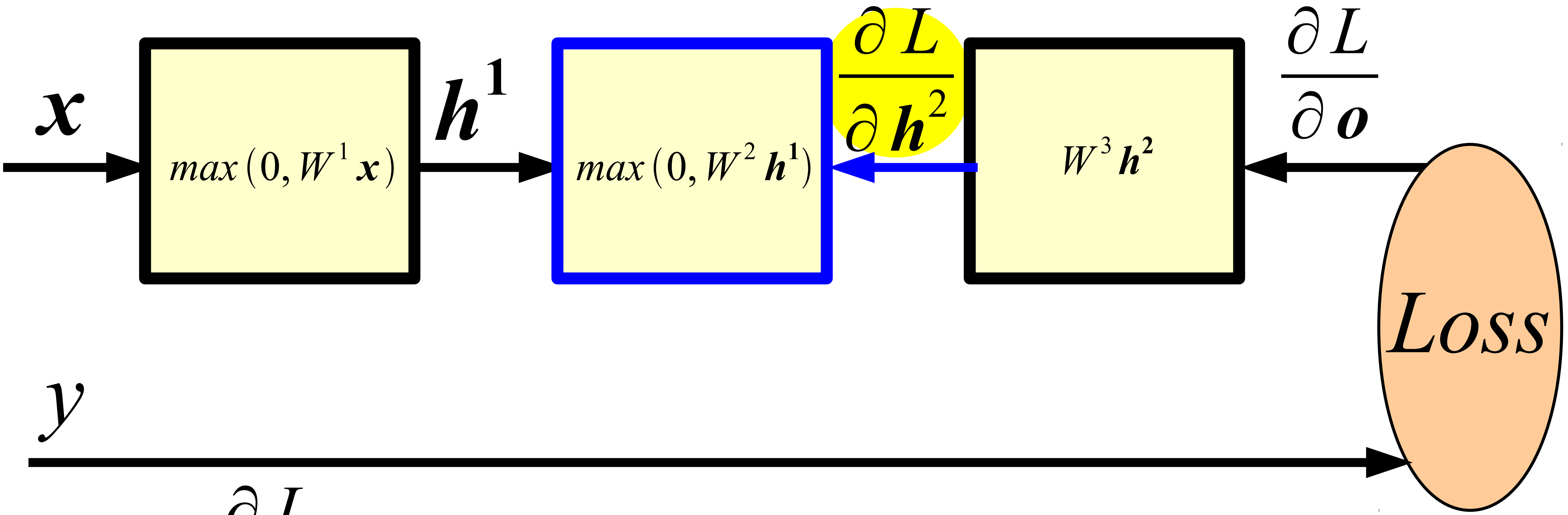


Given $\partial L / \partial \boldsymbol{o}$ and assuming we can easily compute the Jacobian of each module, we have:

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

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

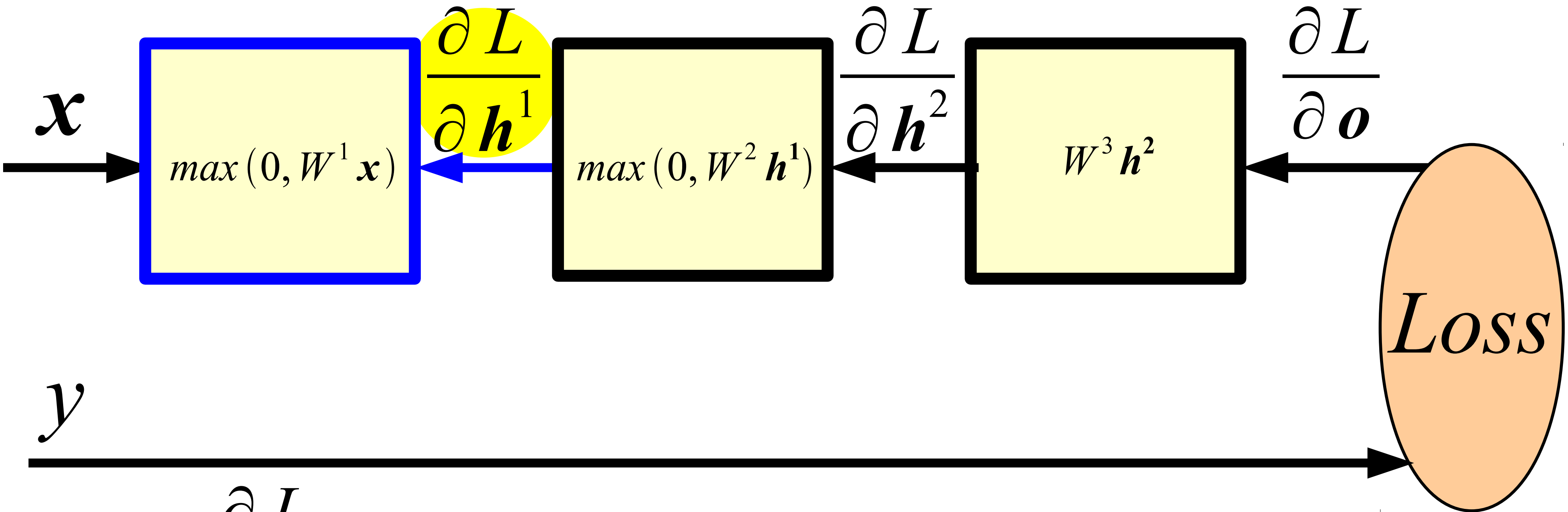
Backward Propagation



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:

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

Backward Propagation

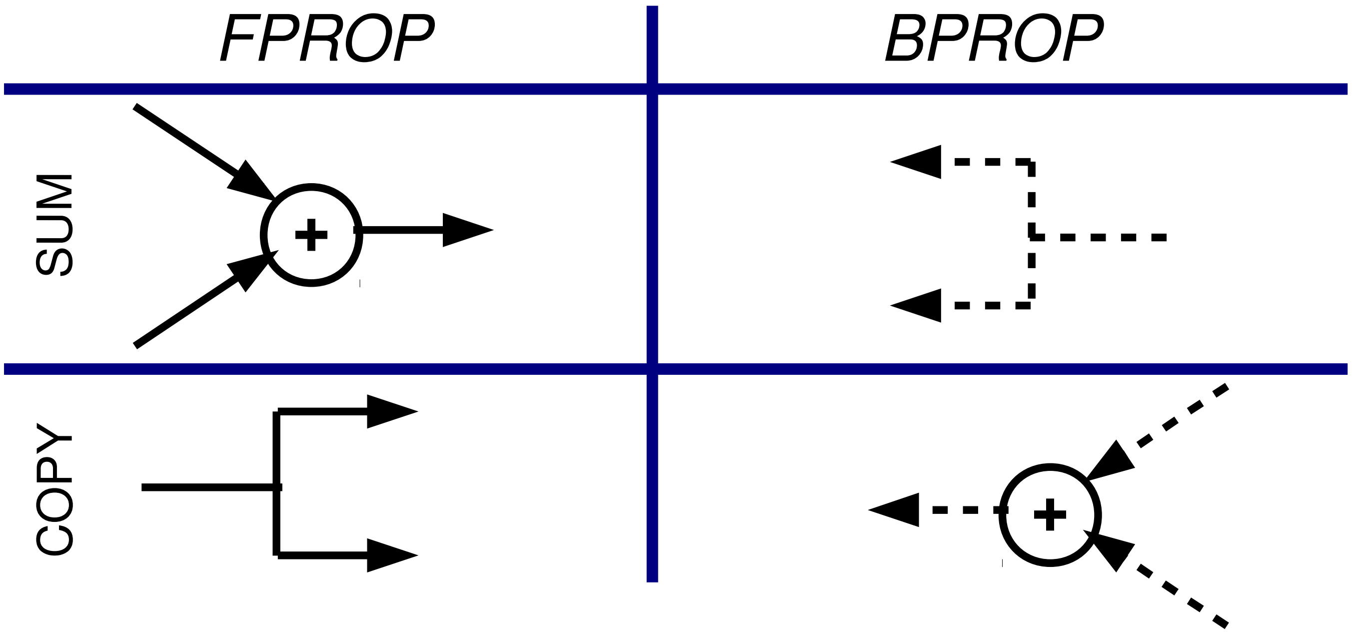
Question: Does BPROP work with ReLU layers only?

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):

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

Stochastic Gradient Descent with Momentum:

$$\theta \leftarrow \theta - \eta \Delta$$

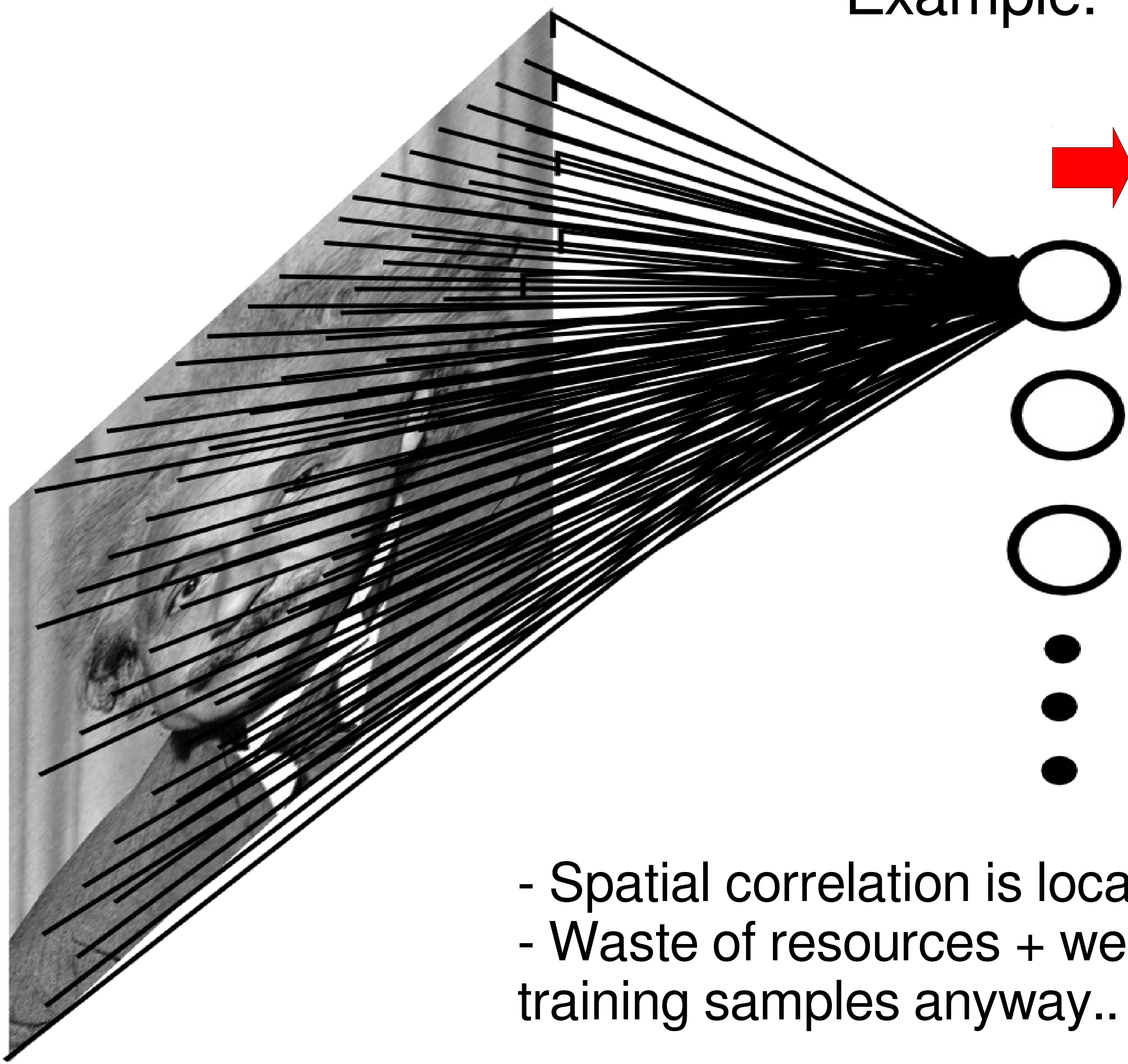
$$\Delta \leftarrow 0.9 \Delta + \frac{\partial L}{\partial \theta}$$

Note: there are many other variants...

Outline

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

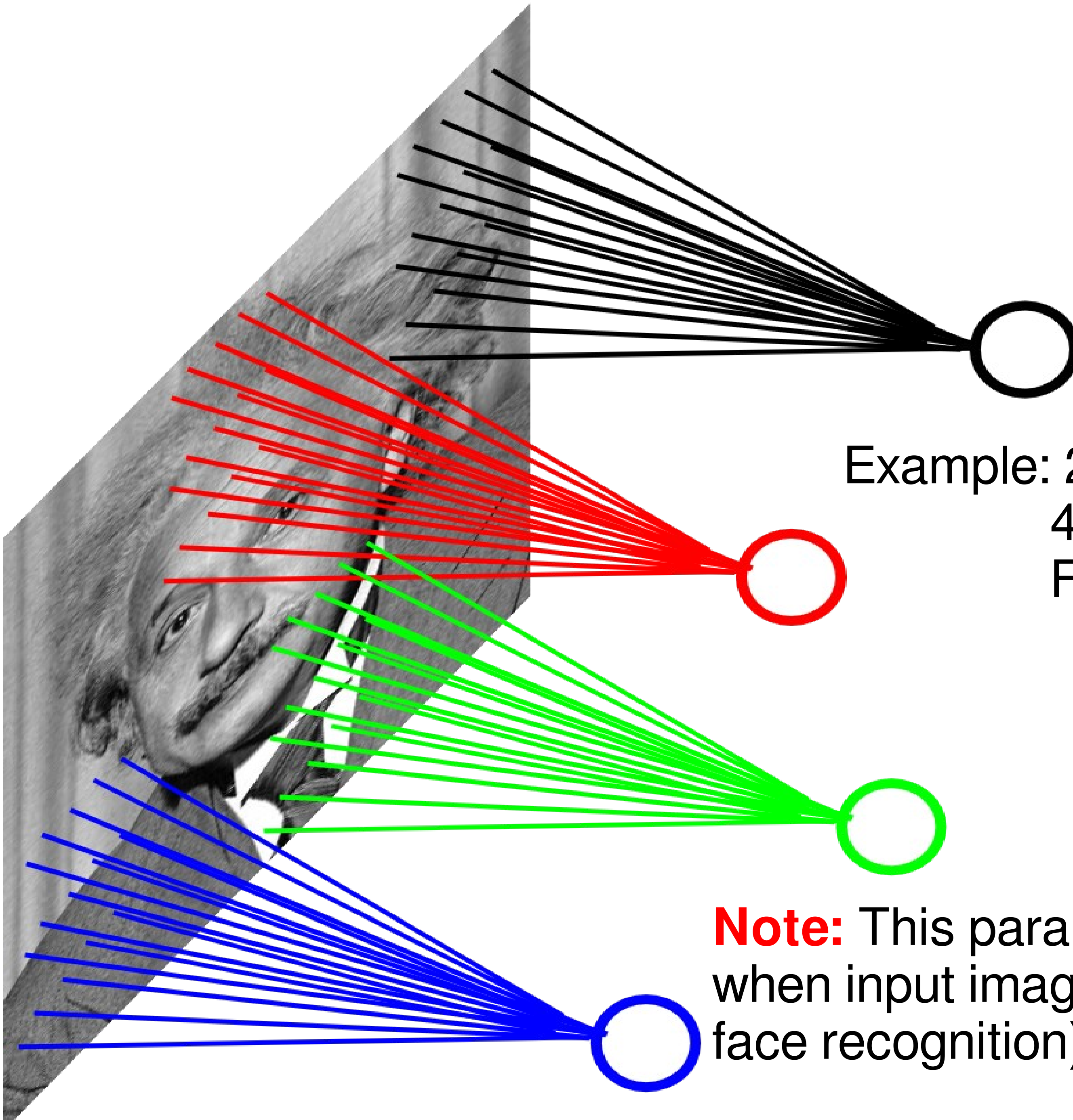
Fully Connected Layer



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

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

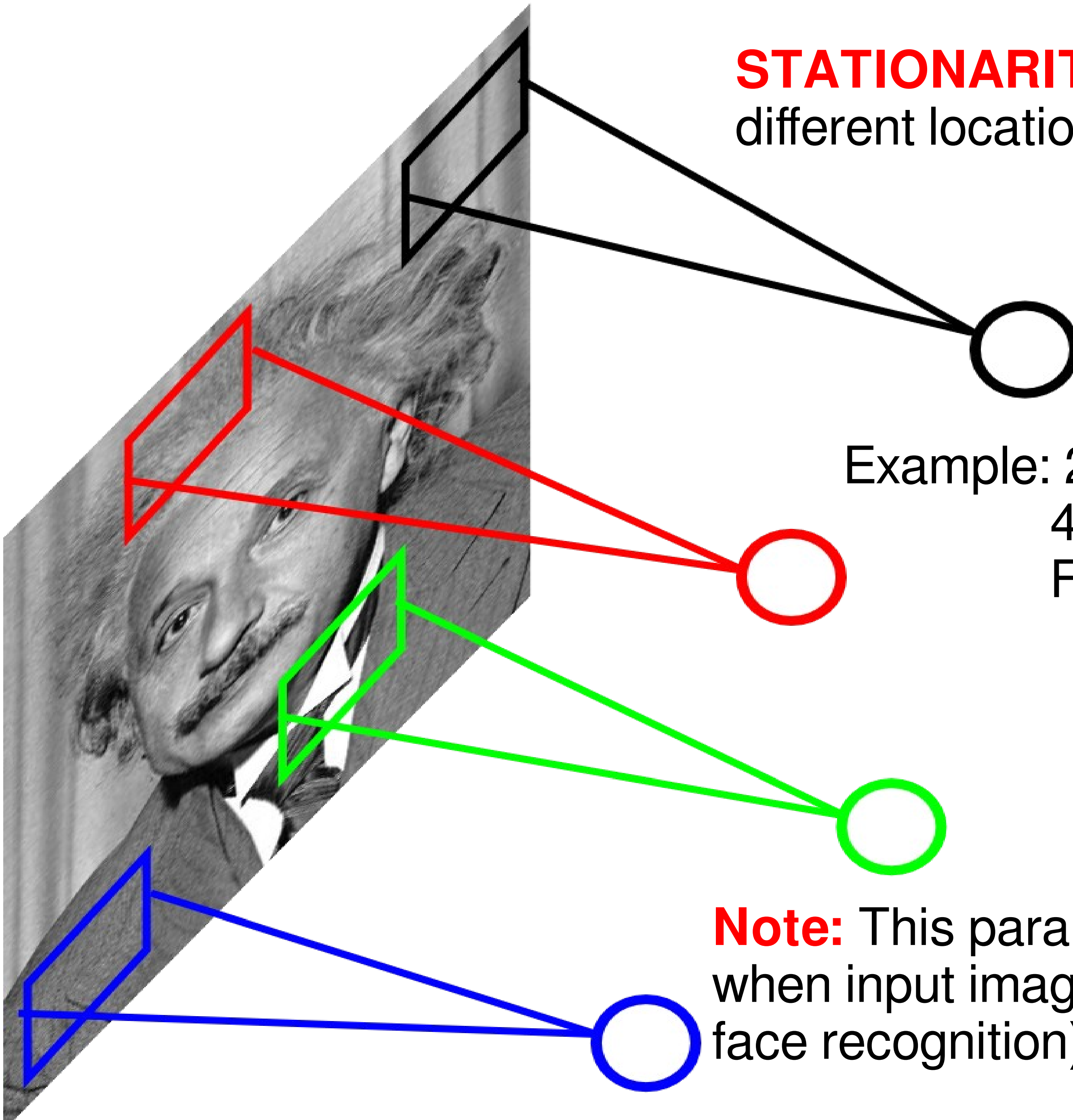
Locally Connected Layer



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



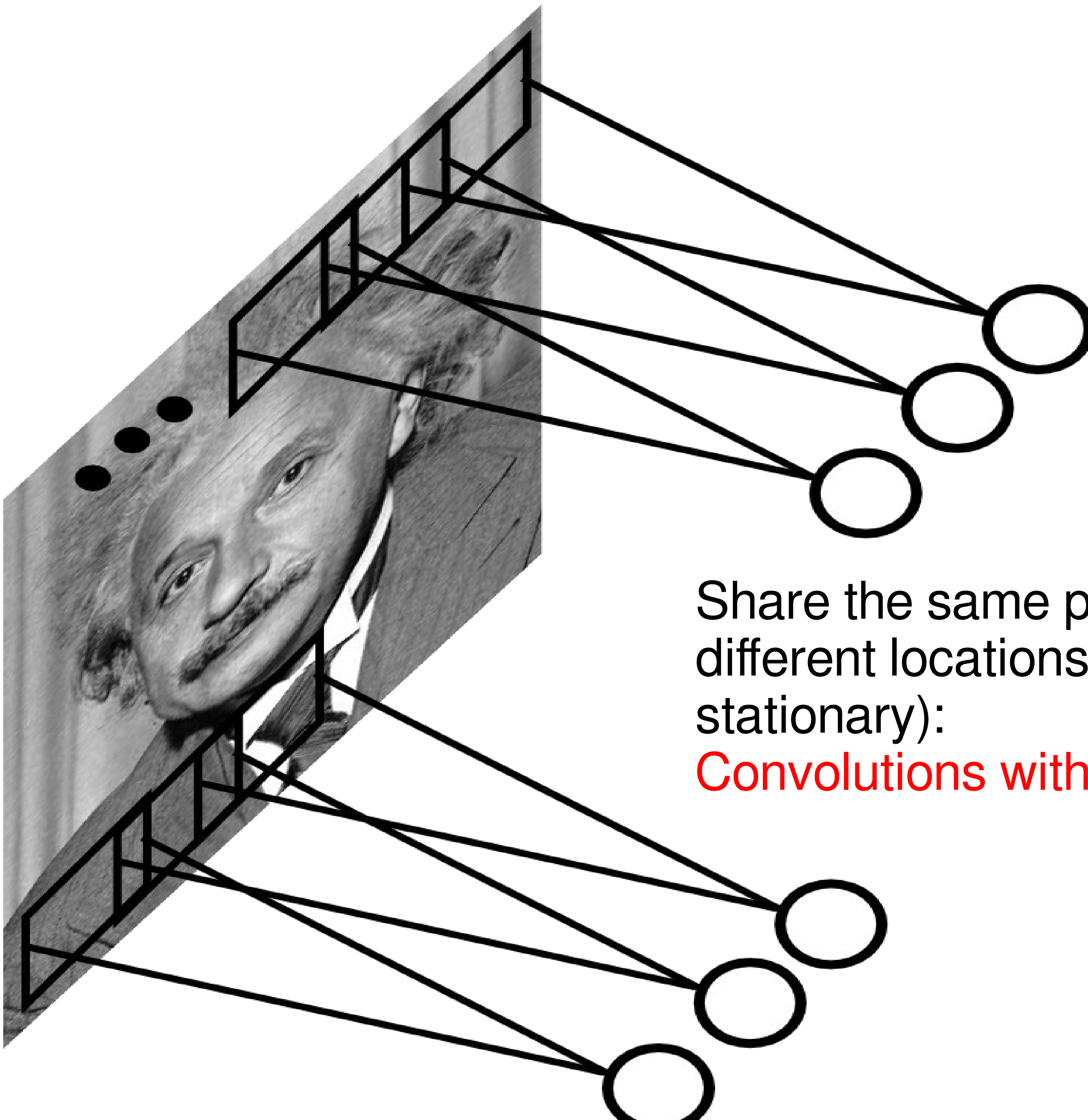
STATIONARITY? Statistics is similar at different locations

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).

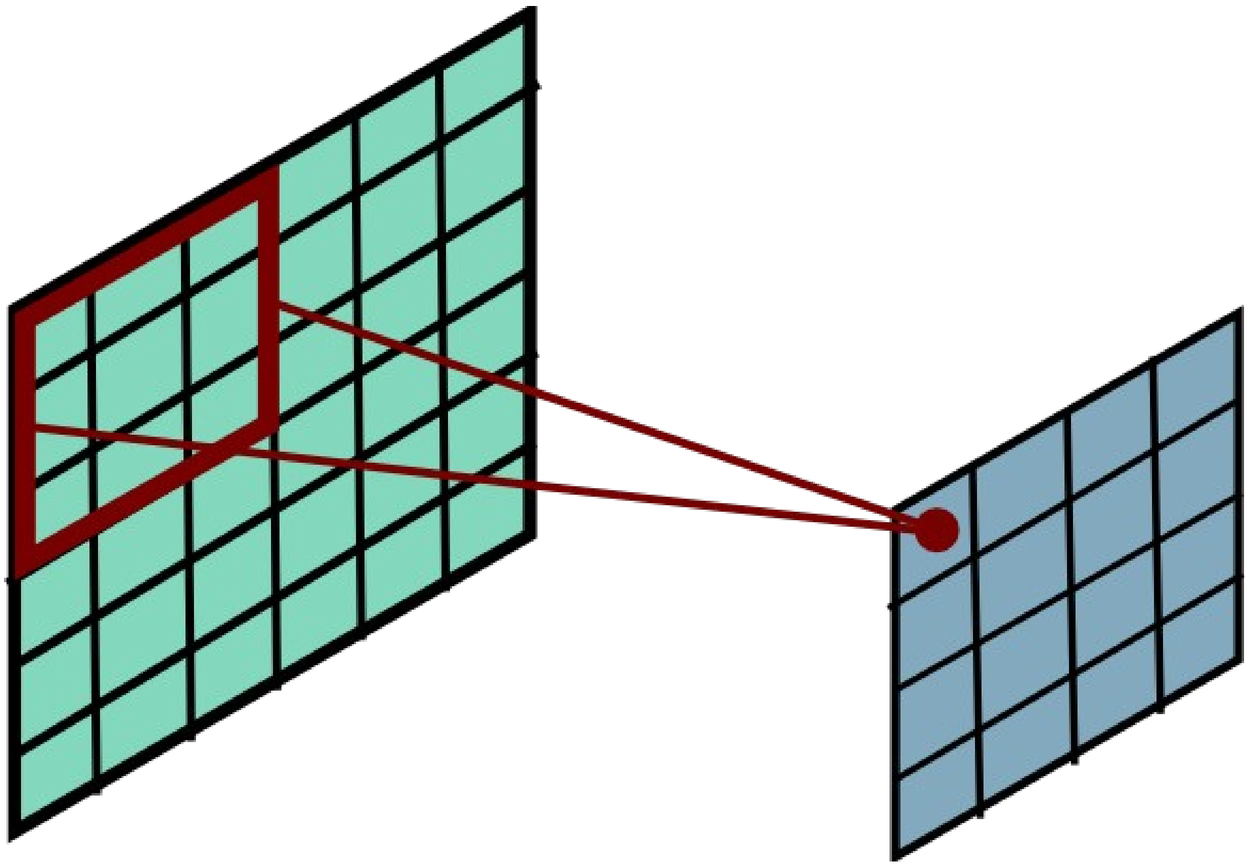
Ranzato

Convolutional Layer

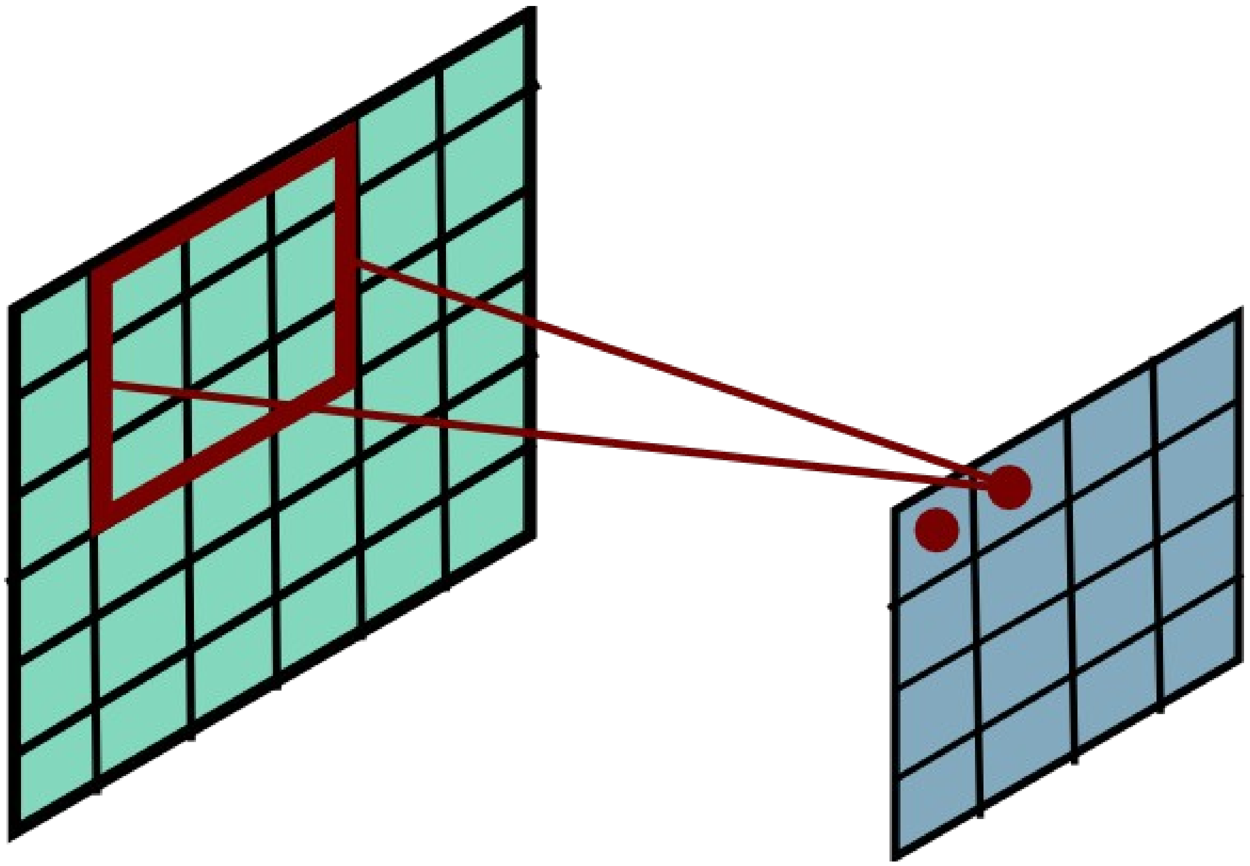


Share the same parameters across different locations (assuming input is stationary):
Convolutions with learned kernels

Convolutional Layer

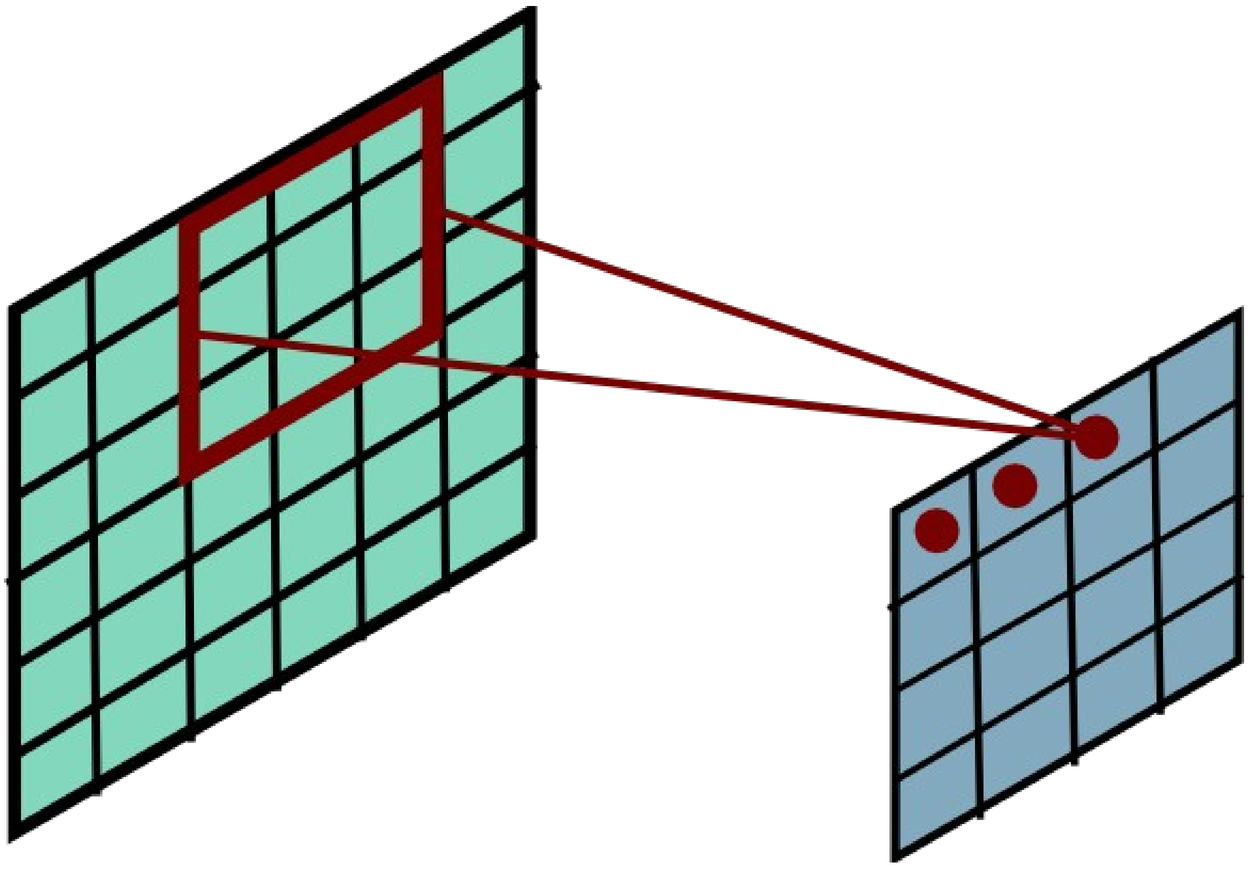


Convolutional Layer



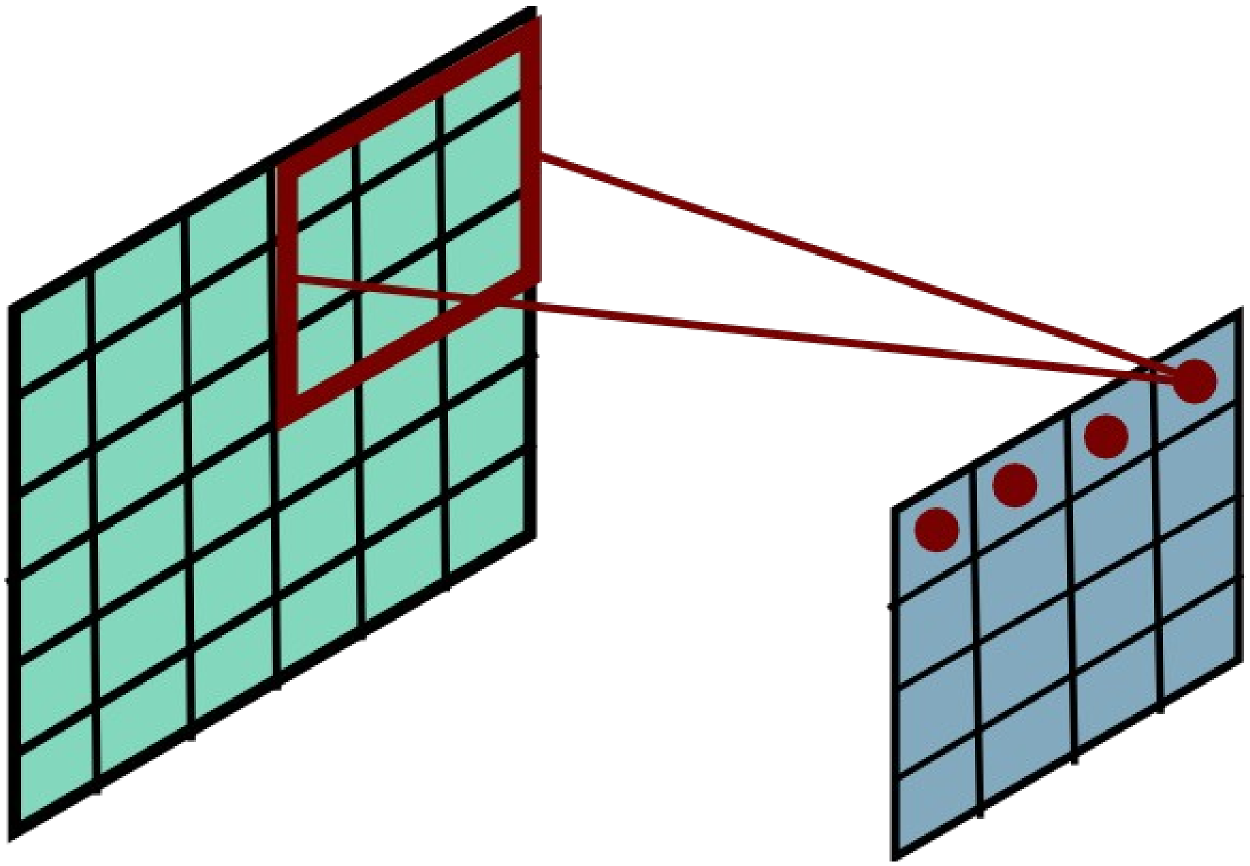
Ranzato

Convolutional Layer



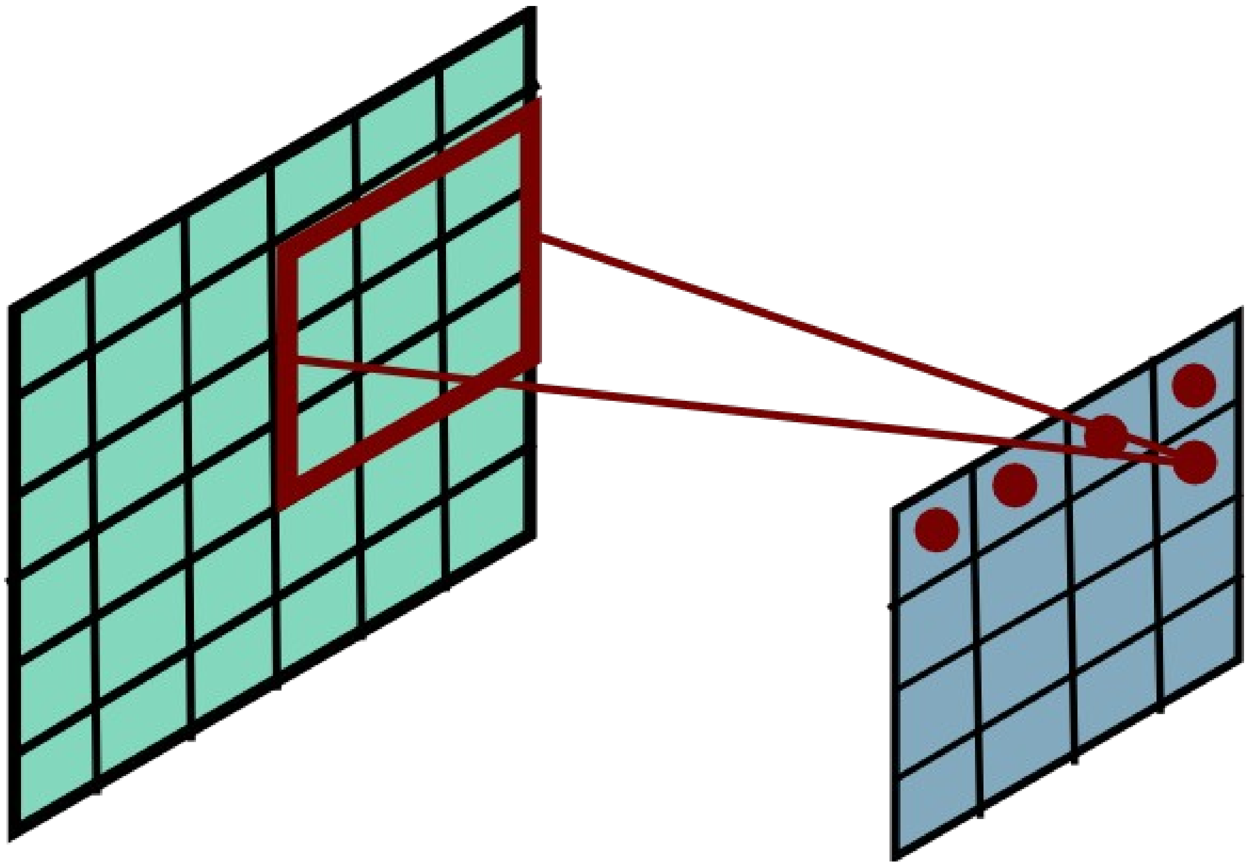
Ranzato

Convolutional Layer



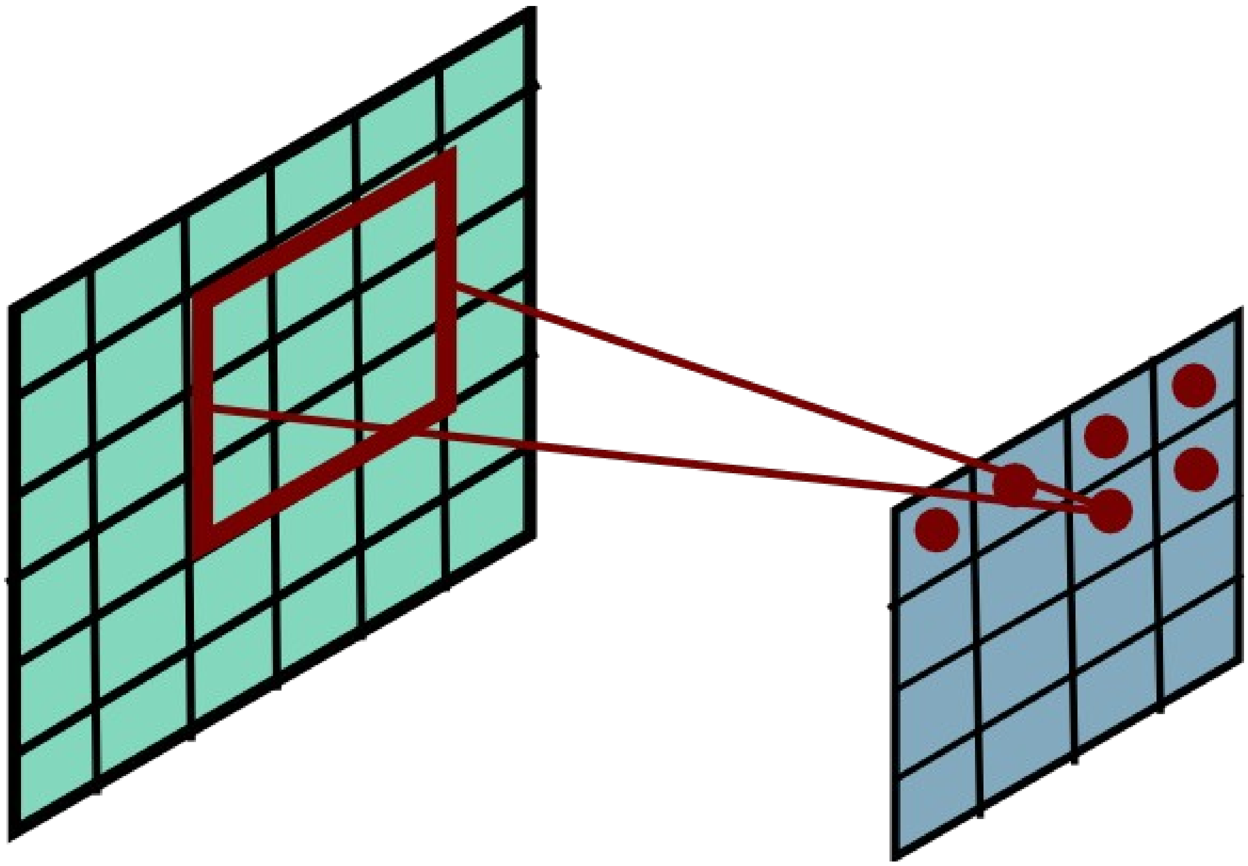
Ranzato

Convolutional Layer

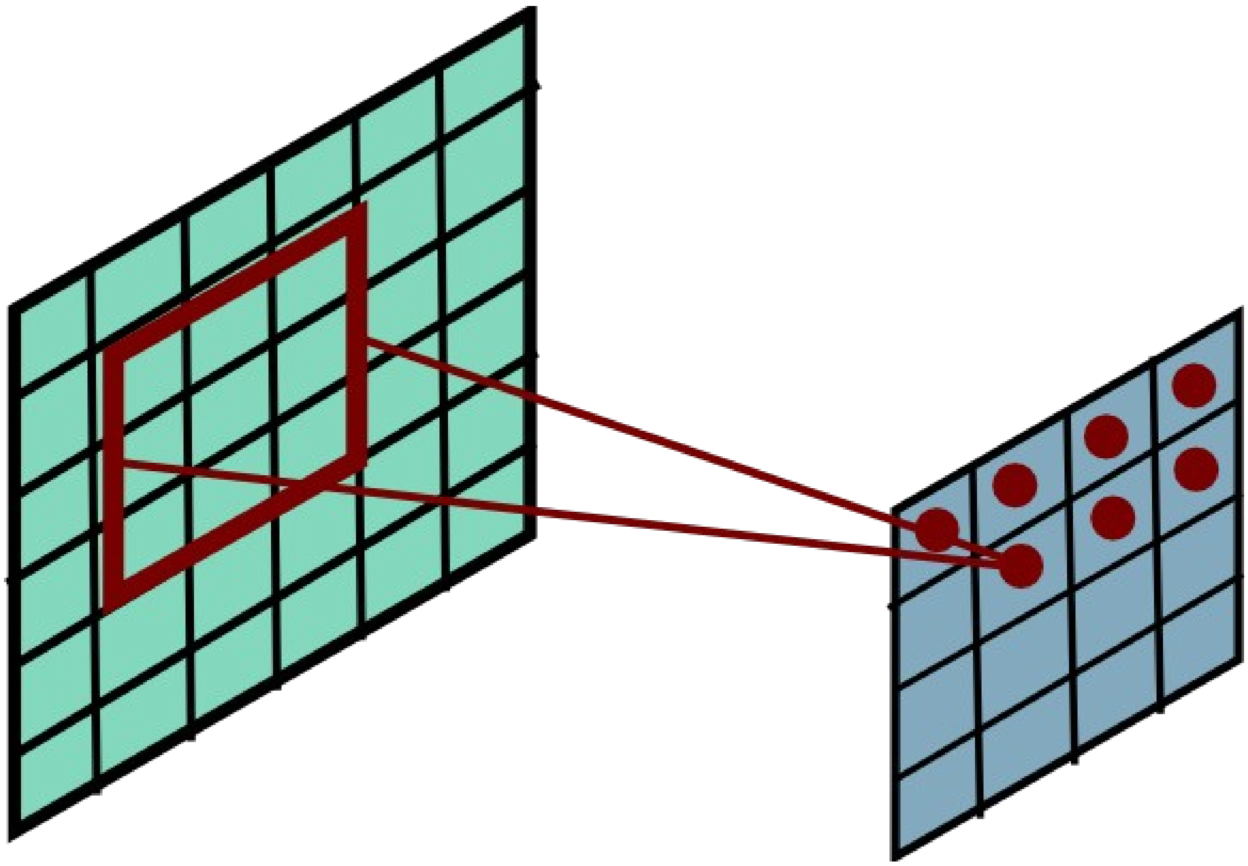


Ranzato

Convolutional Layer

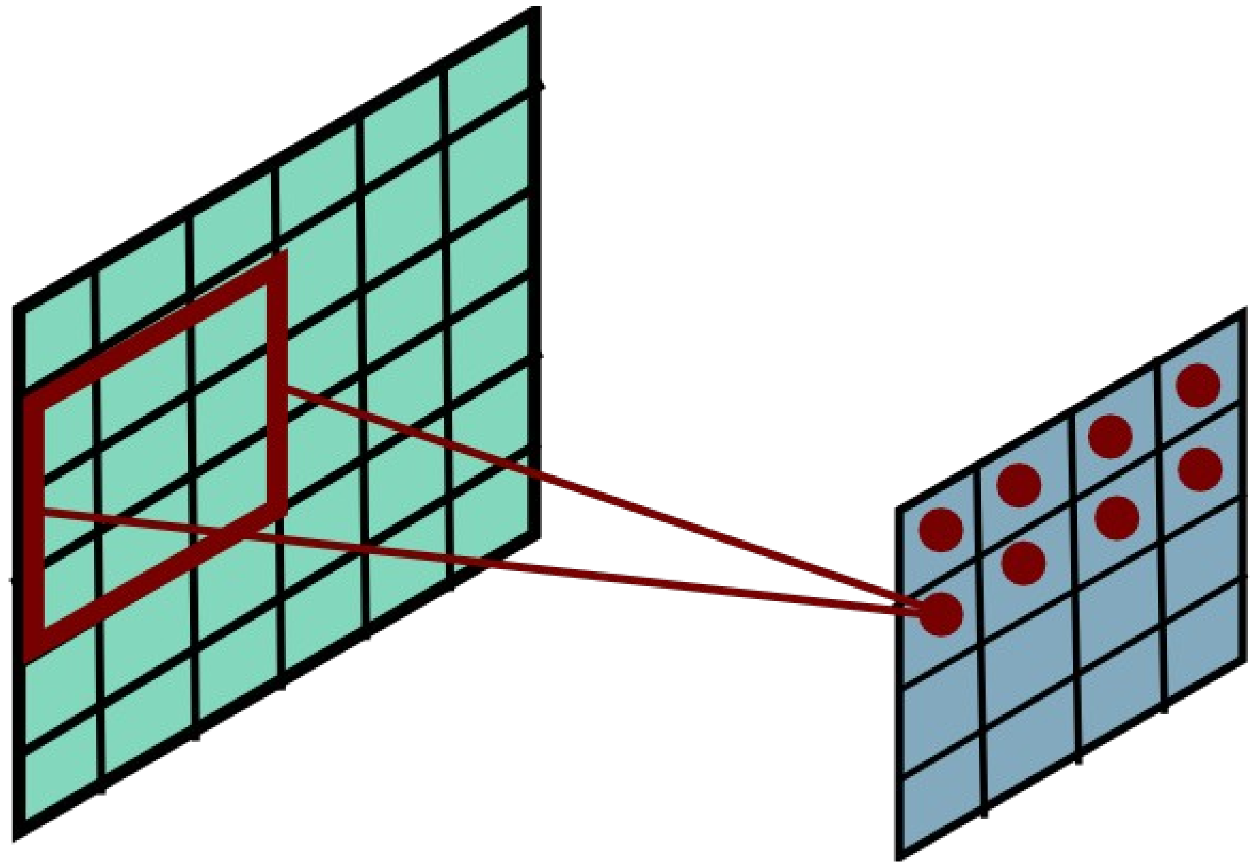


Convolutional Layer

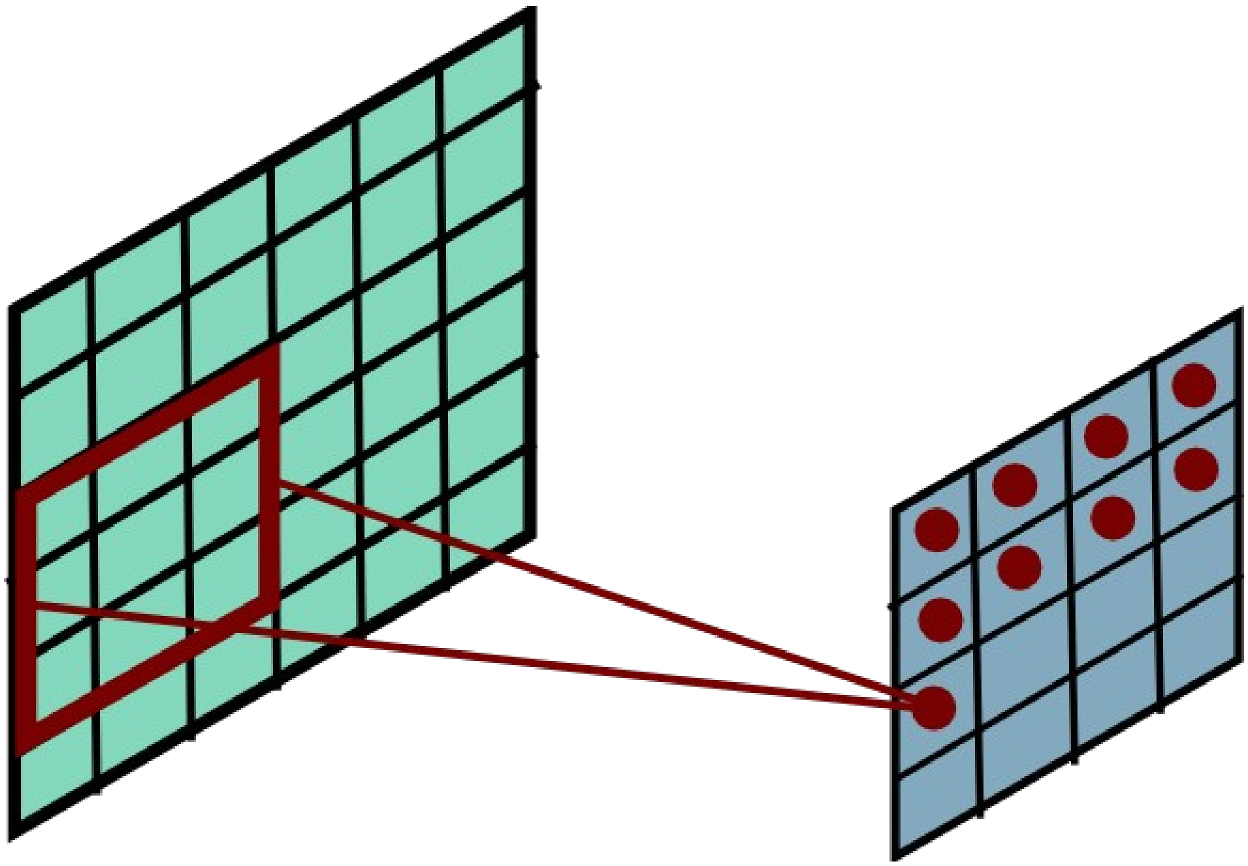


Ranzato

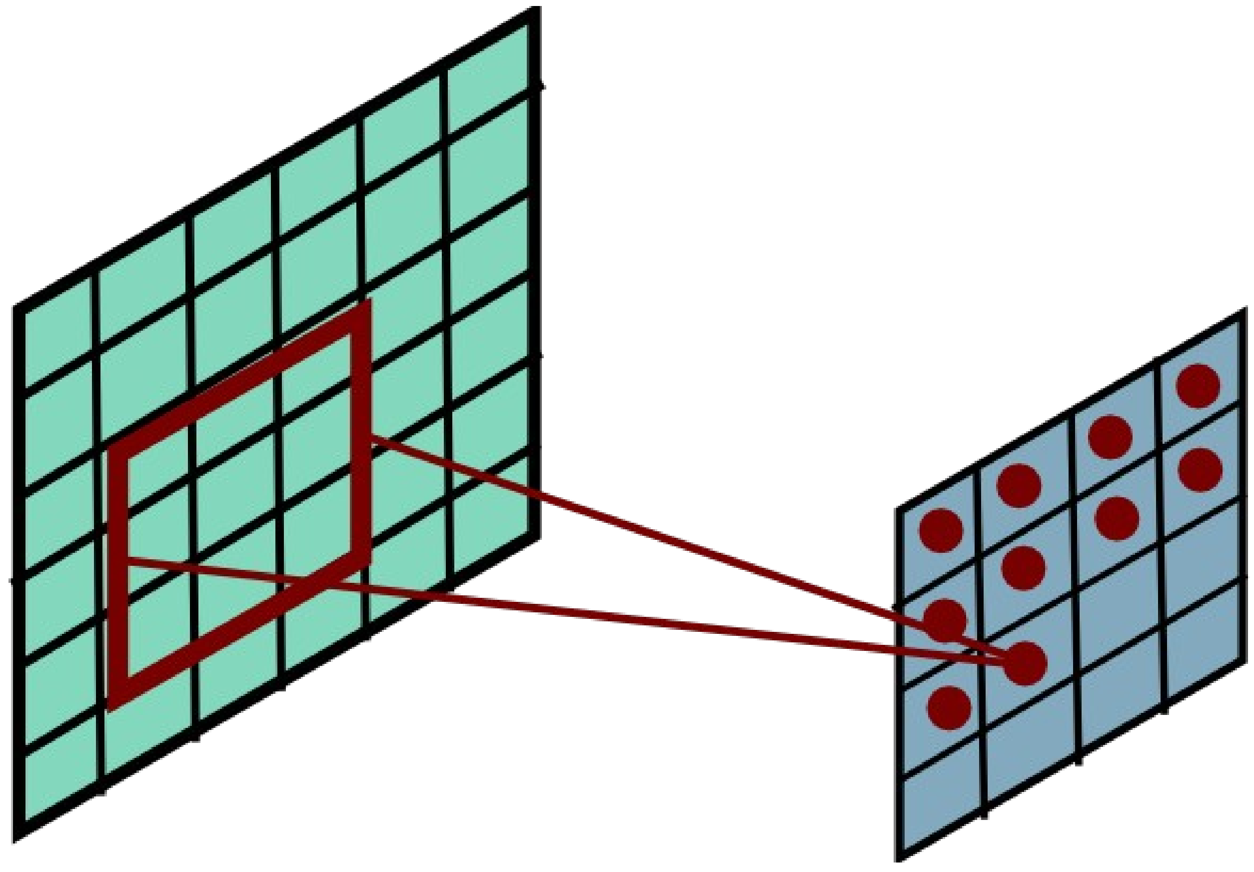
Convolutional Layer



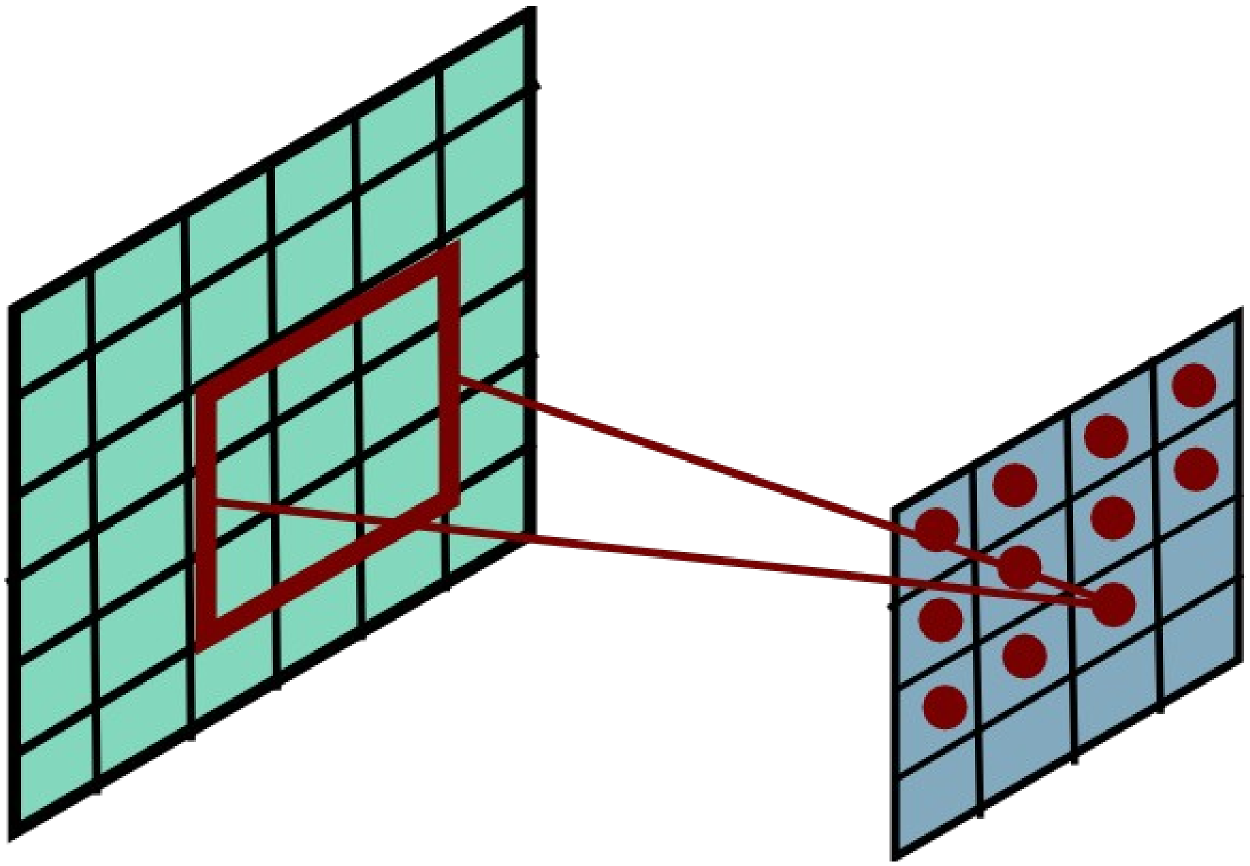
Convolutional Layer



Convolutional Layer

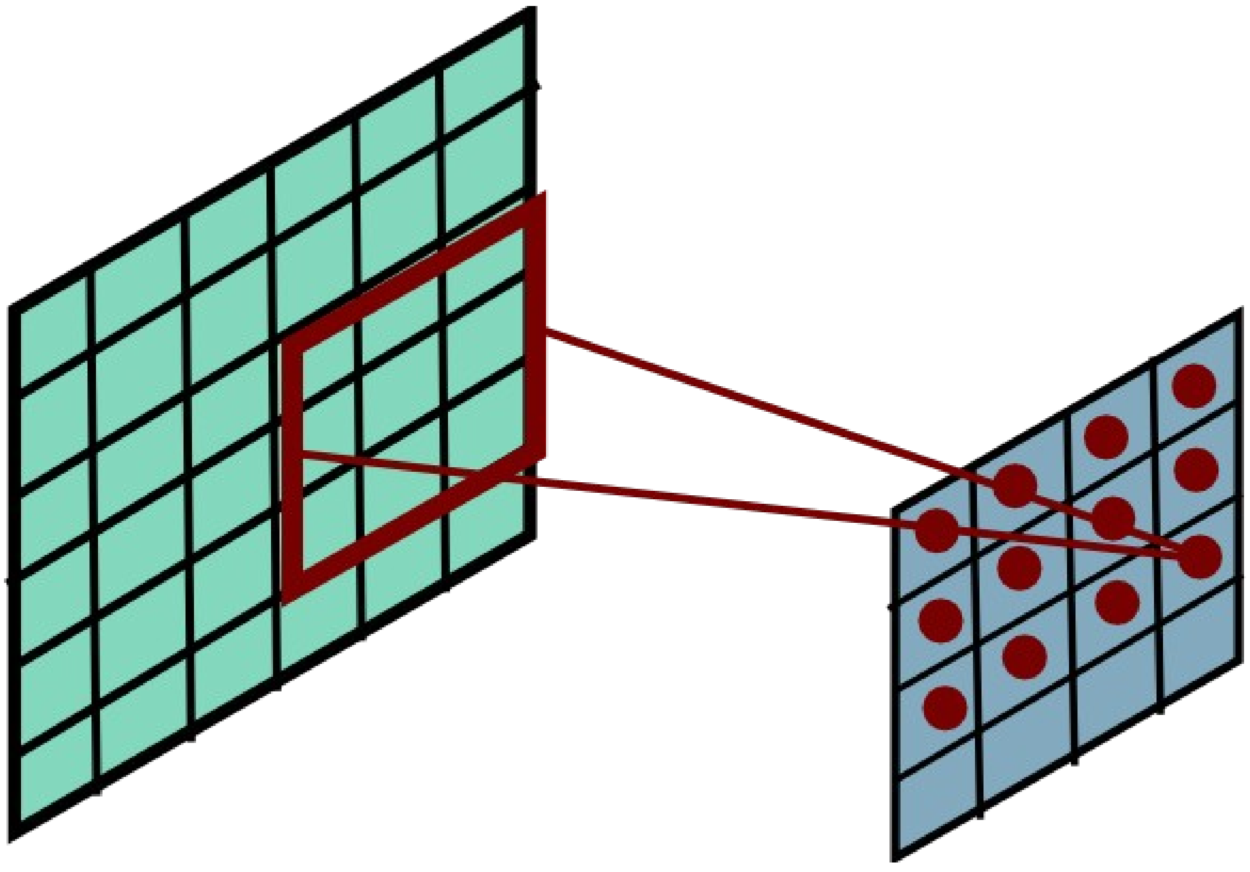


Convolutional Layer

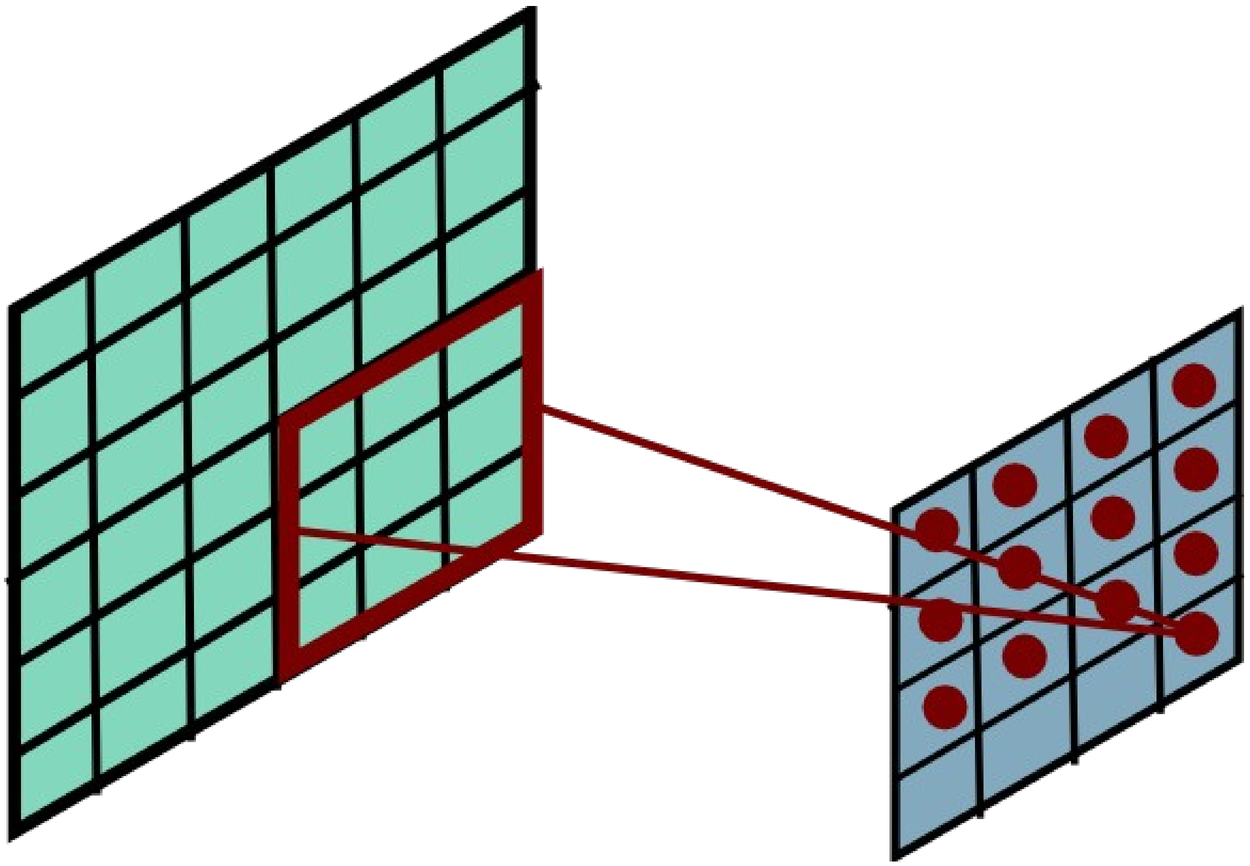


Ranzato

Convolutional Layer

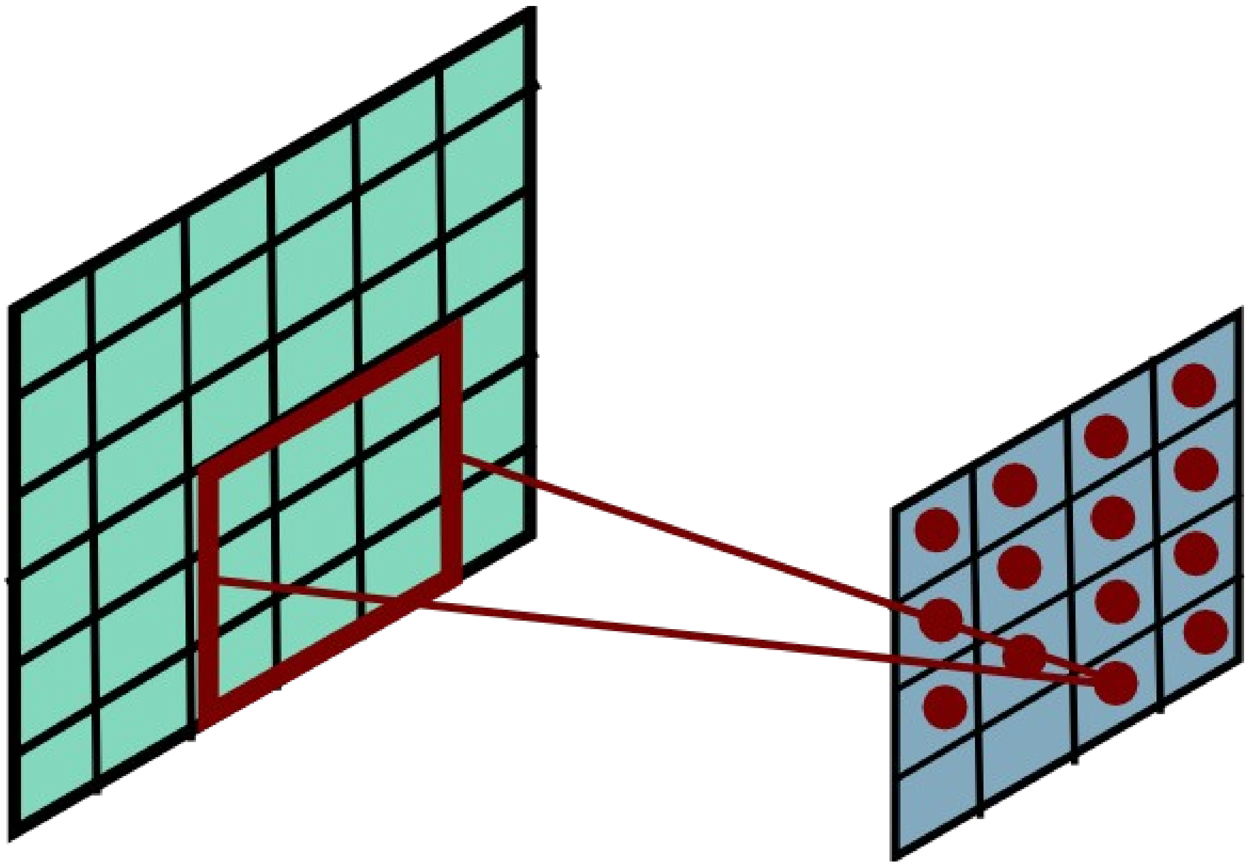


Convolutional Layer



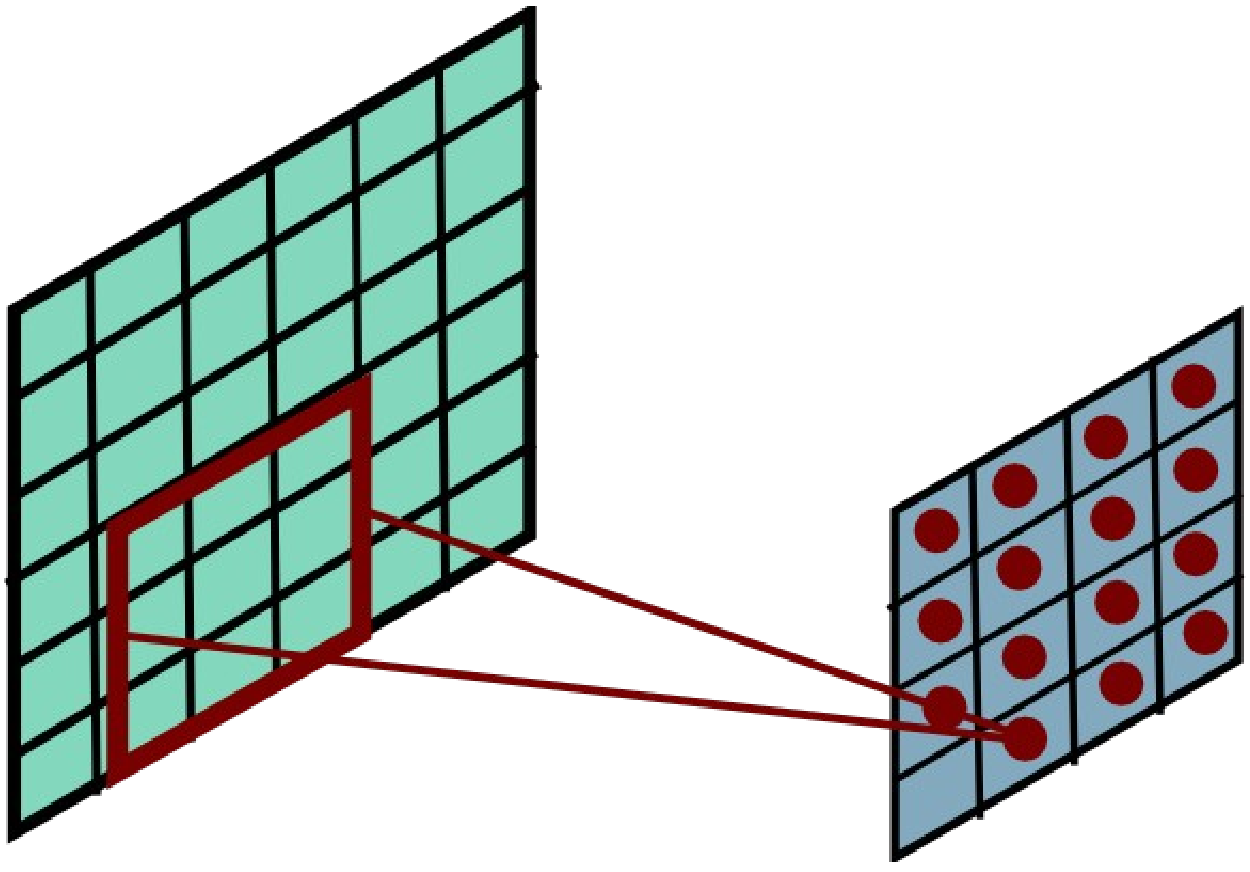
Ranzato

Convolutional Layer



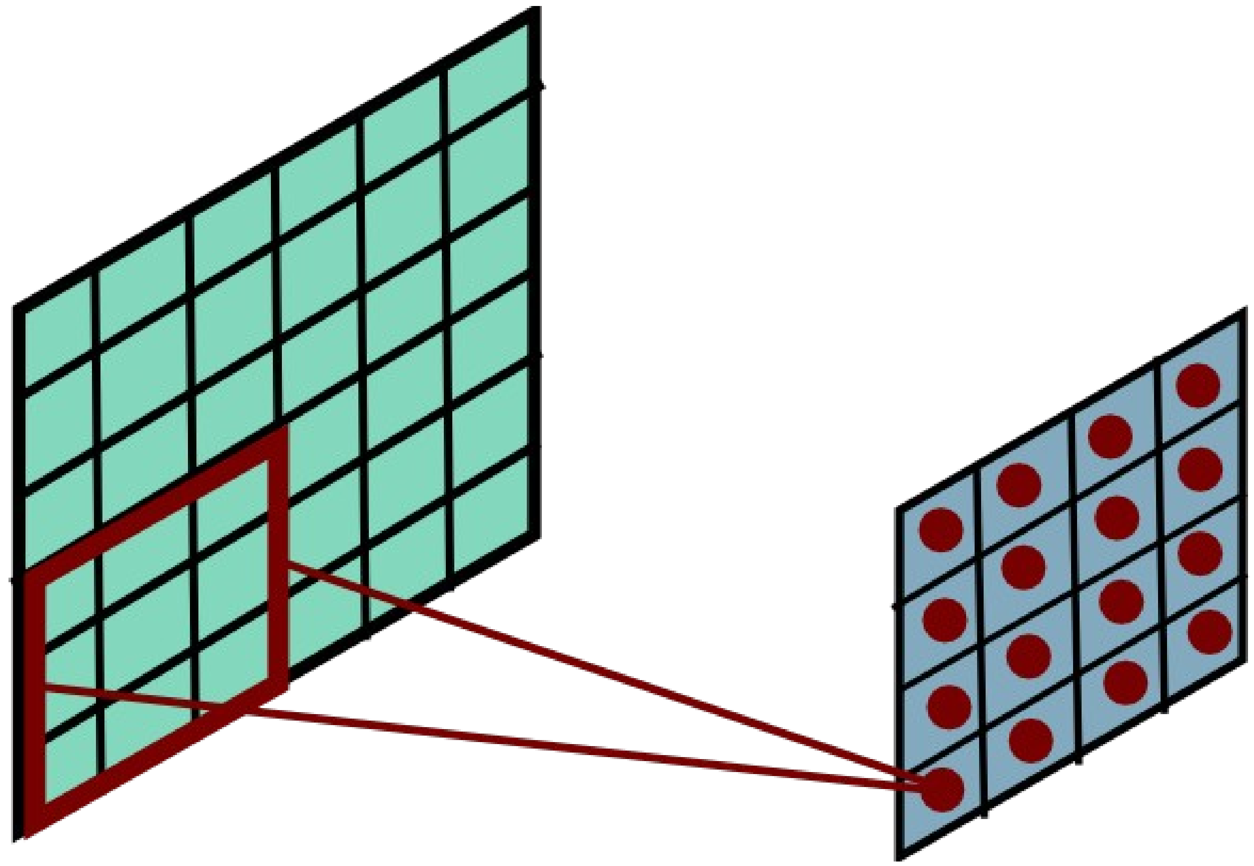
Ranzato

Convolutional Layer



Ranzato

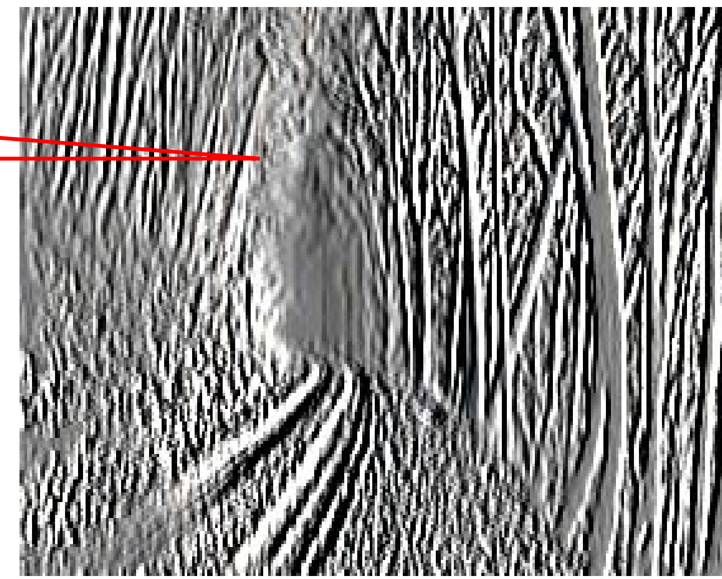
Convolutional Layer



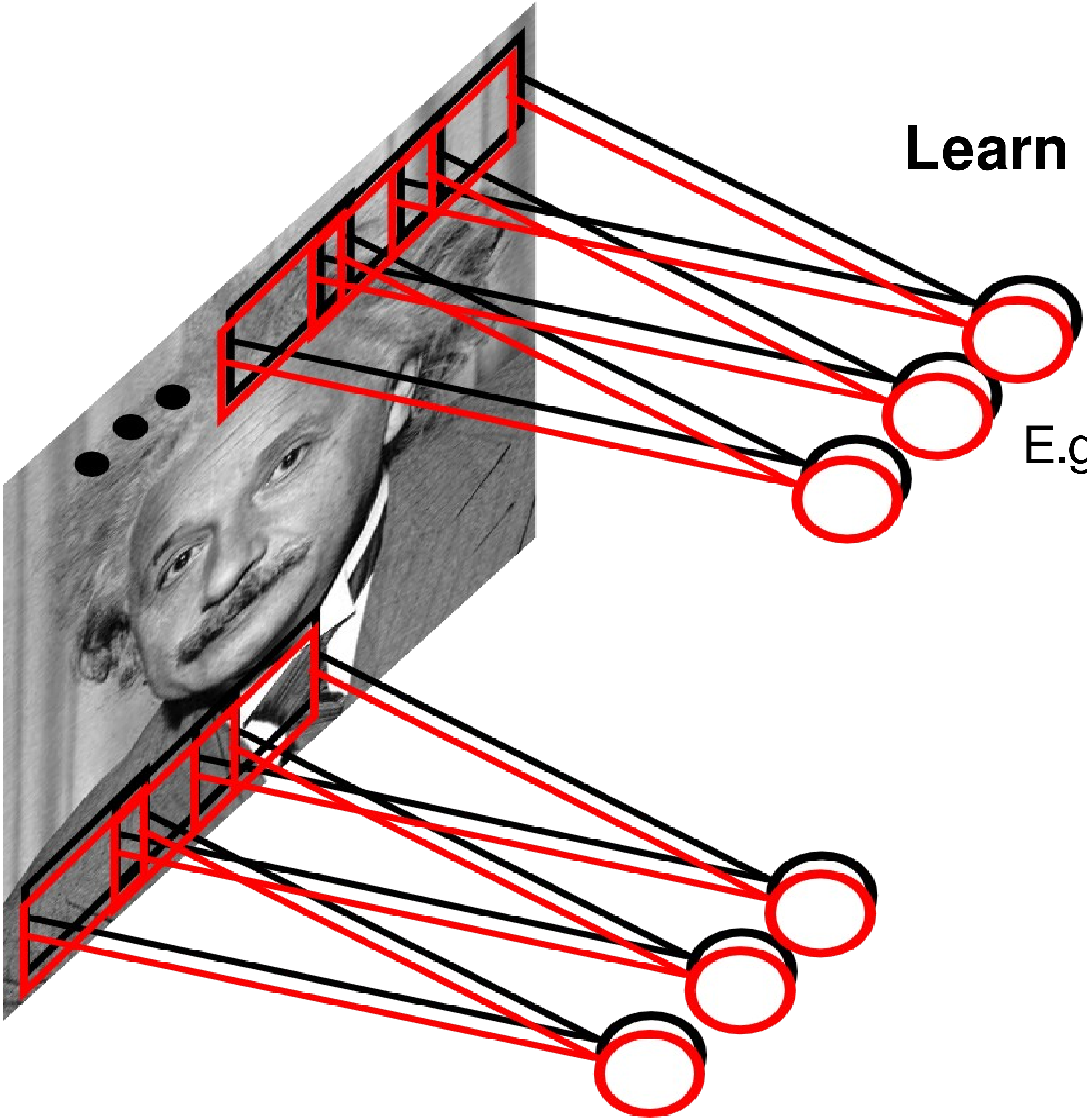
Mathieu et al. “Fast training of CNNs through FFTs” ICLR 2014

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

Ranzato

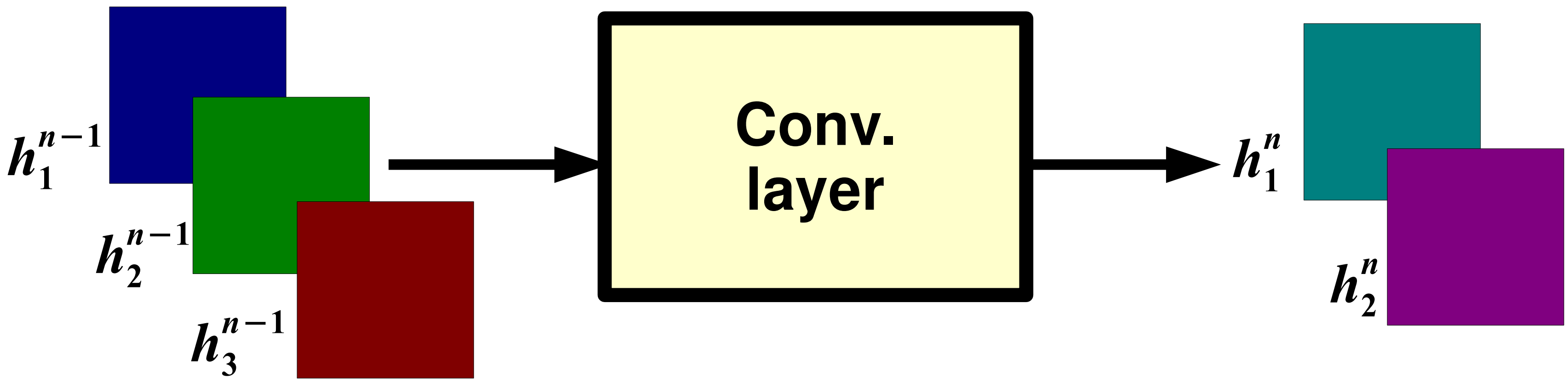
Convolutional Layer

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

output feature map

input feature map

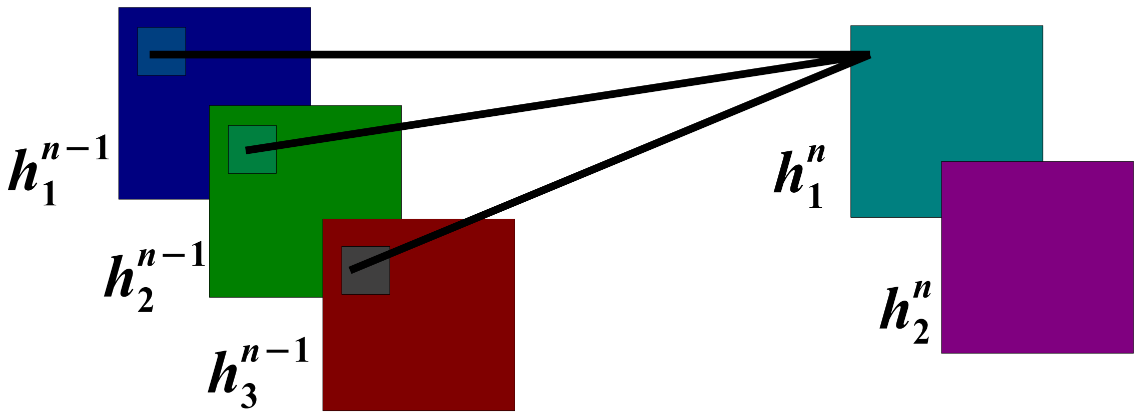
kernel



Convolutional Layer

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

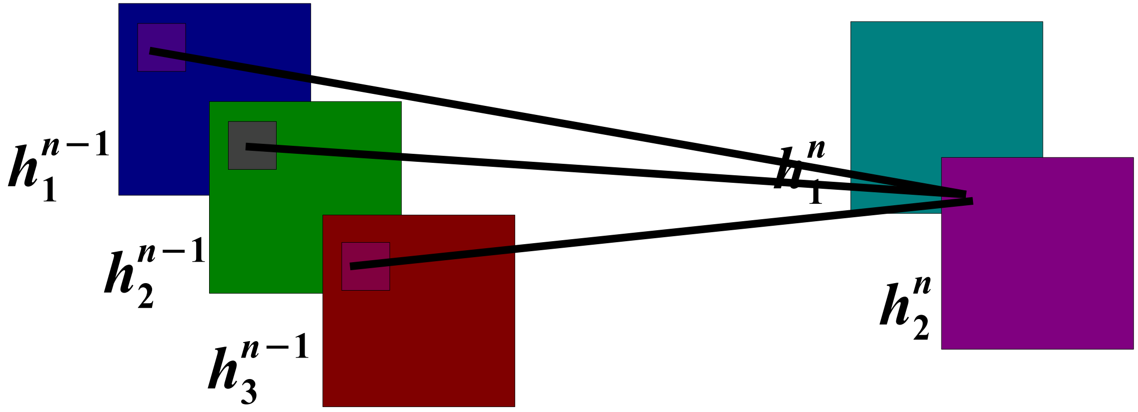
output feature map **input feature map** **kernel**



Convolutional Layer

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

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

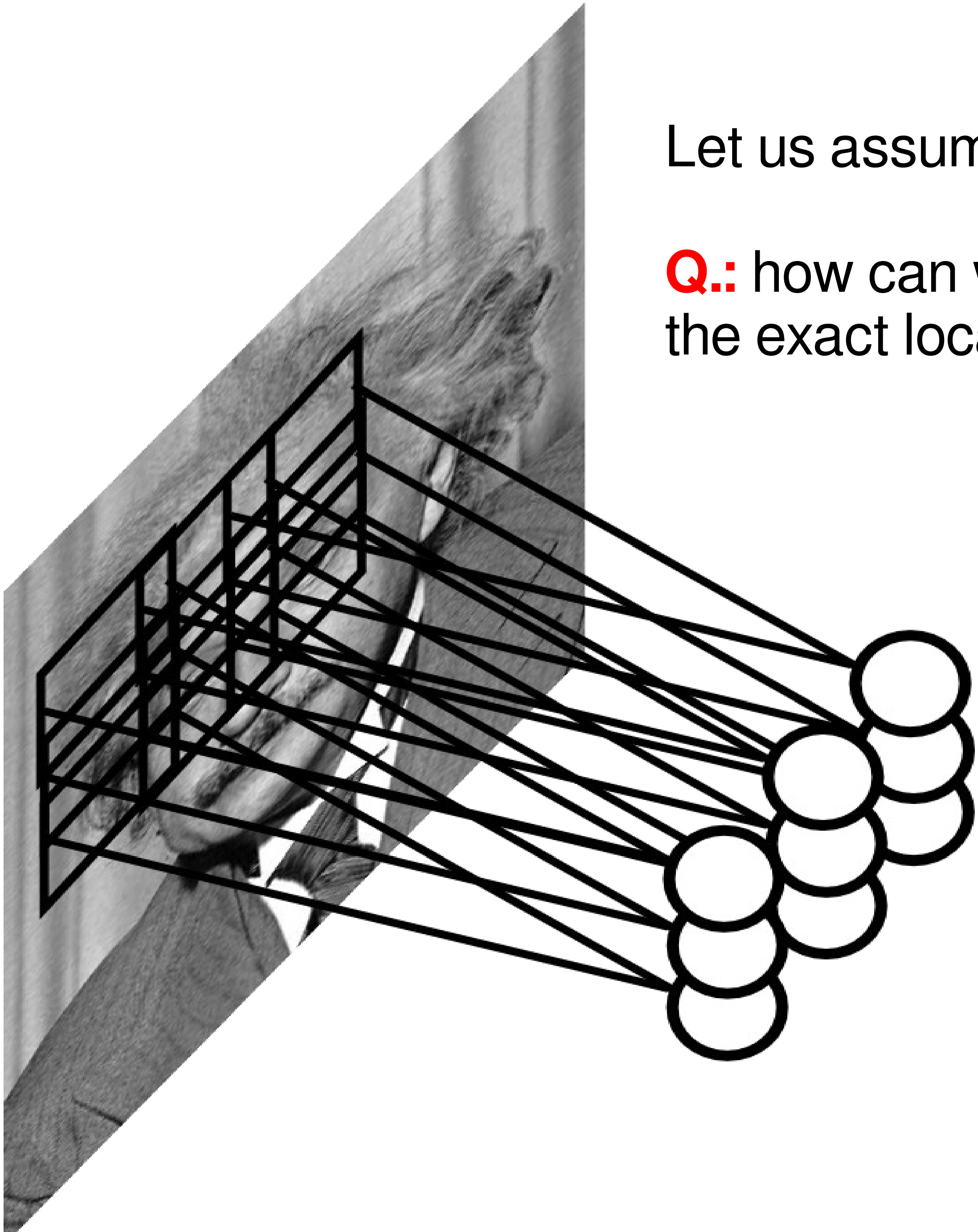
Solution:

- 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

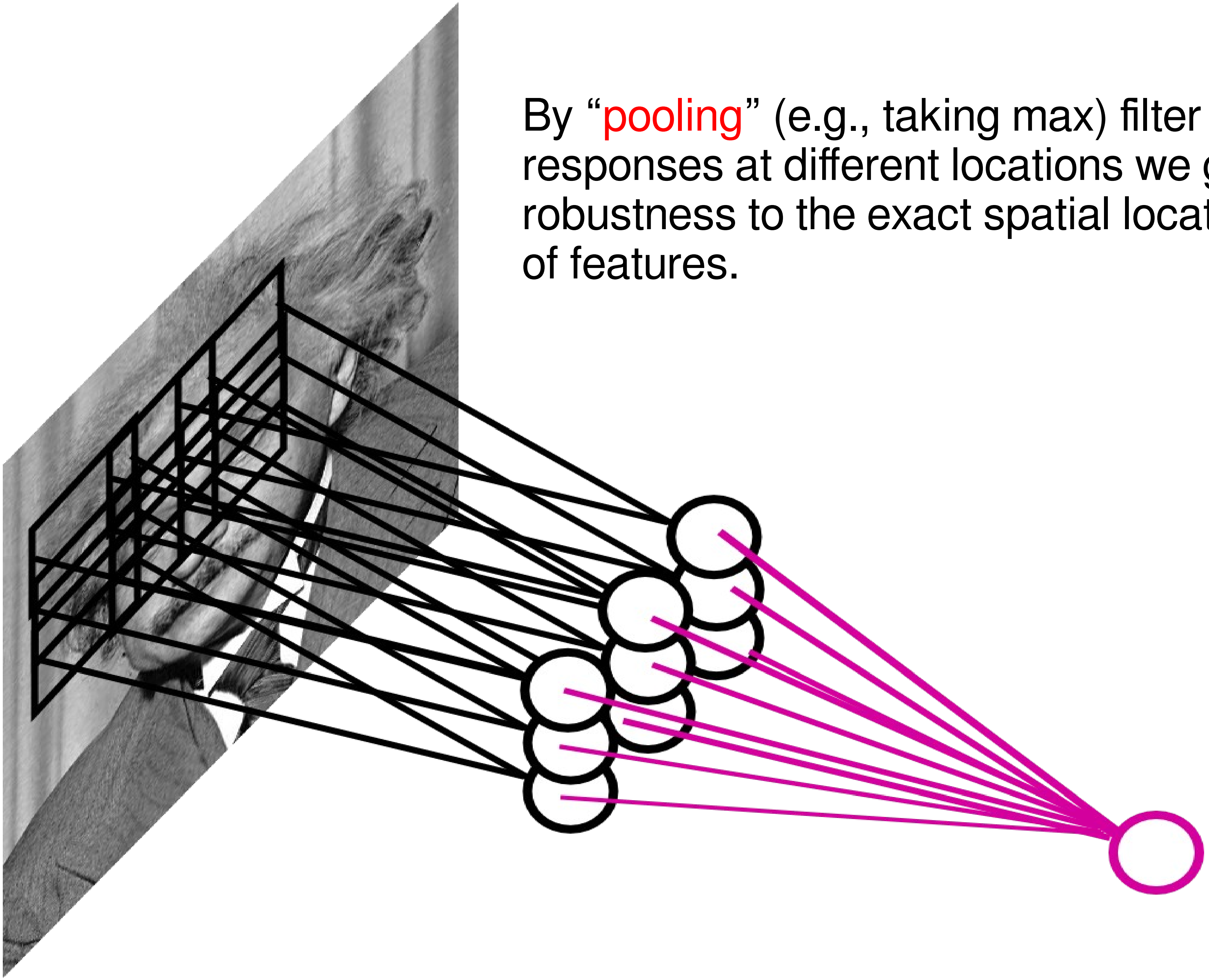


Let us assume filter is an "eye" detector.

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?

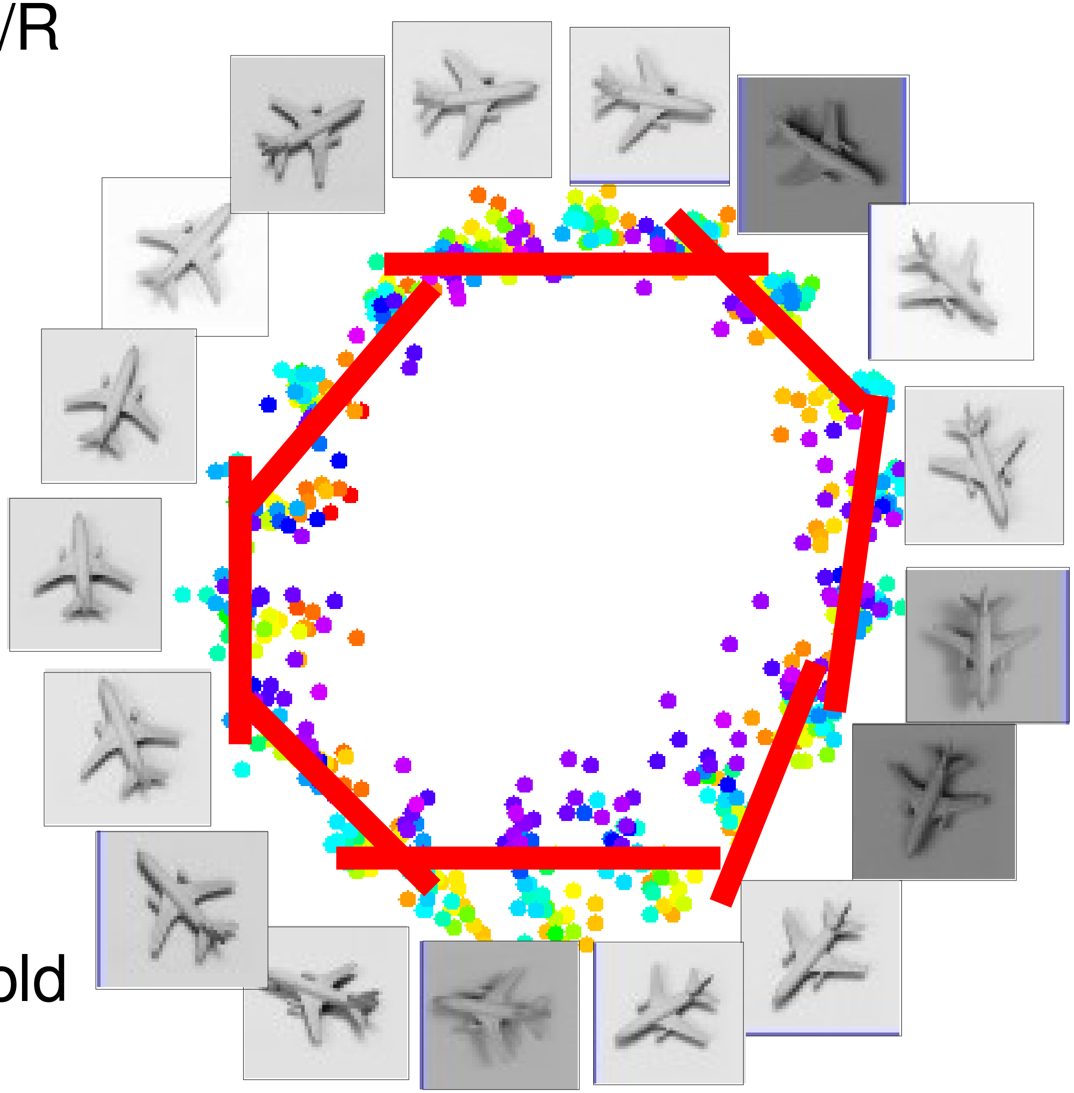
Answer: The size of the output depends on the stride between the pools. For instance, if pools do not overlap and have size KxK, and the input has size DxD with M input feature maps, then:
- output is M@(D/K)x(D/K)
- the computational cost is proportional to the size of the input (negligible compared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

Task: detect orientation L/R

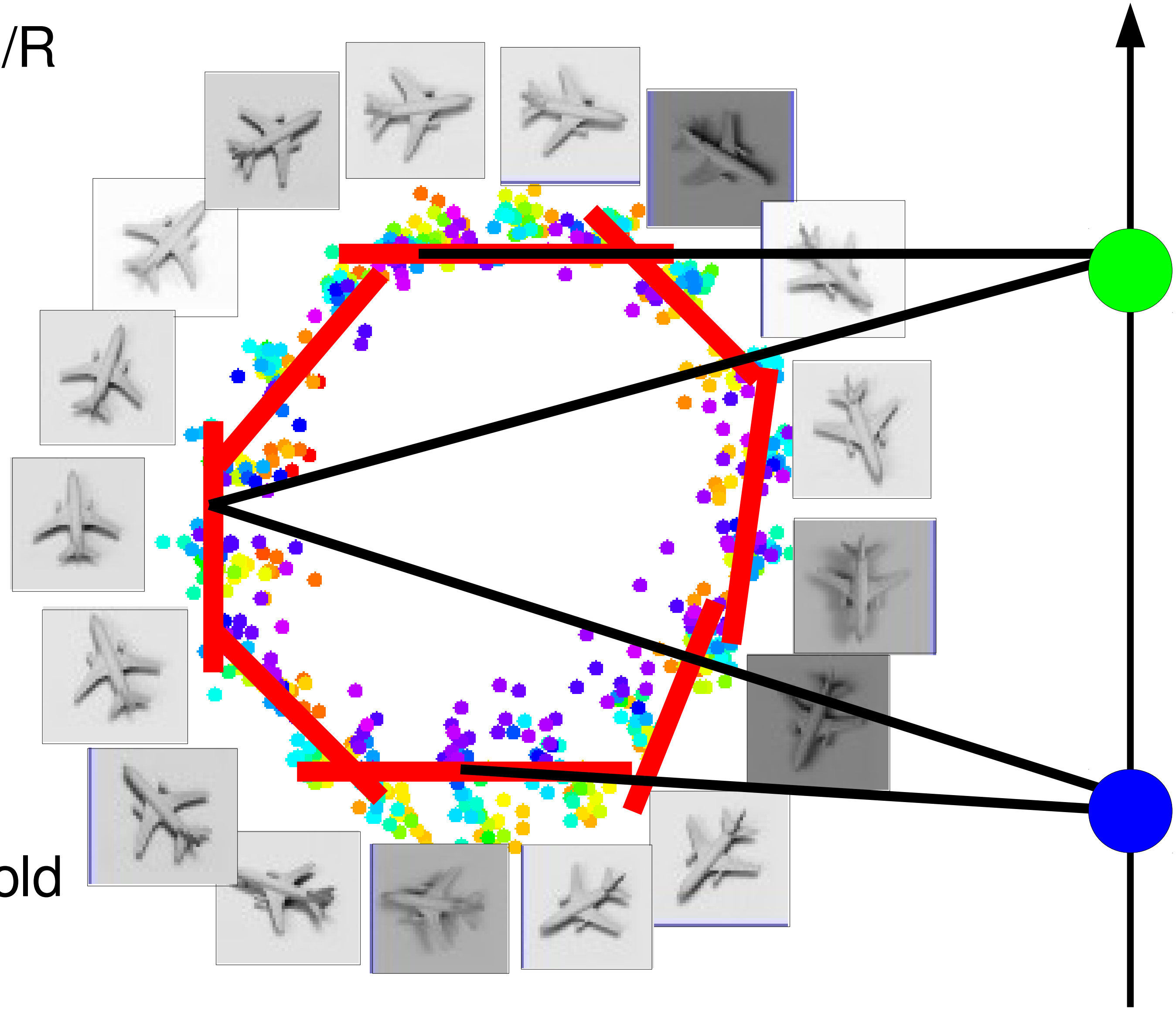


Conv layer: linearizes manifold

Ranzato

Pooling Layer: Interpretation

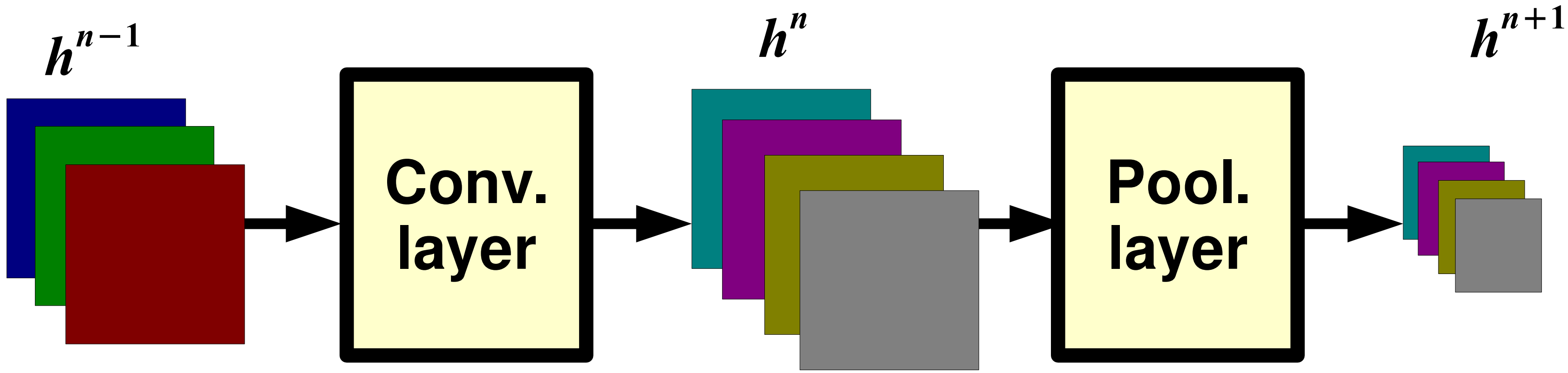
Task: detect orientation L/R



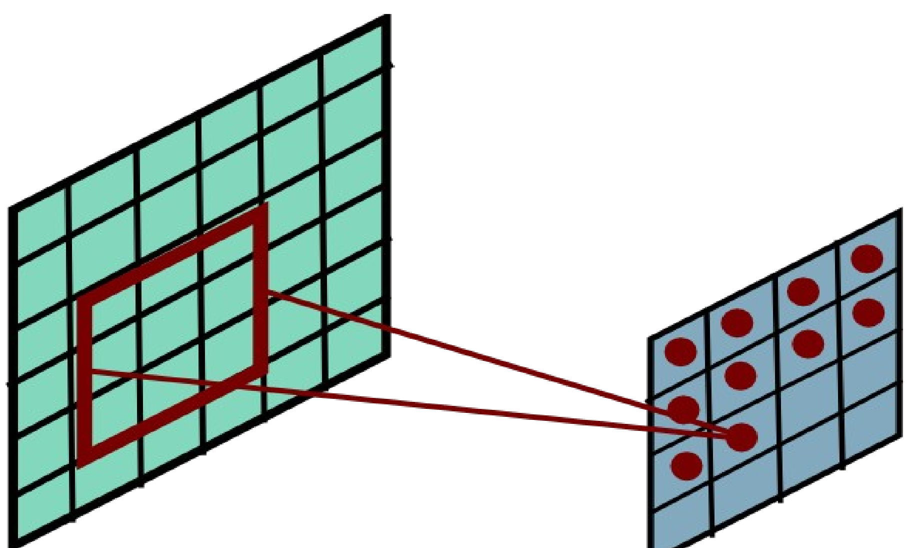
Conv layer:
linearizes manifold

Pooling layer:
collapses manifold

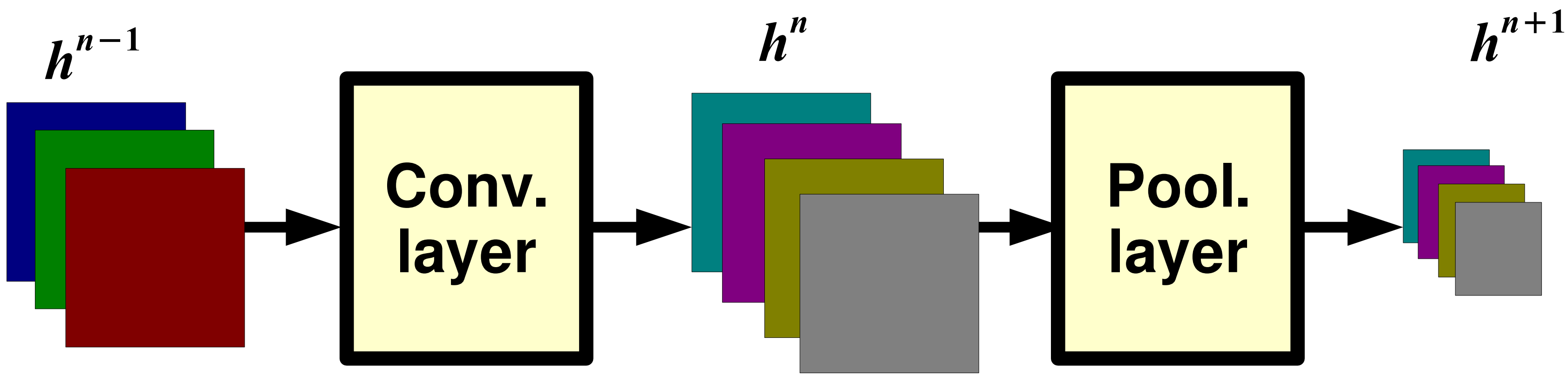
Pooling Layer: Receptive Field Size



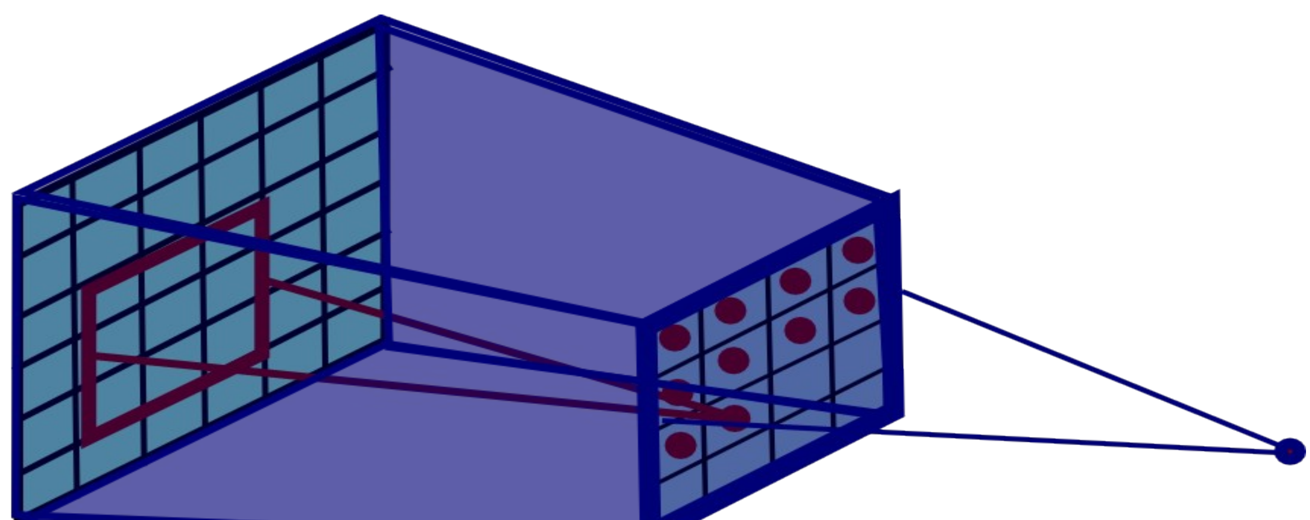
If convolutional filters have size KxK and stride 1, and pooling layer has pools of size PxP, then each unit in the pooling layer depends upon a patch (at the input of the preceding conv. layer) of size: (P+K-1)x(P+K-1)



Pooling Layer: Receptive Field Size



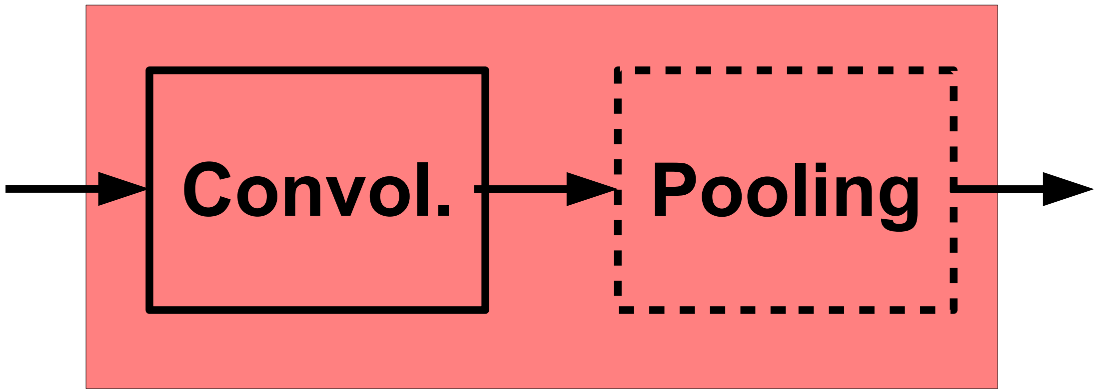
If convolutional filters have size KxK and stride 1, and pooling layer has pools of size PxP, then each unit in the pooling layer depends upon a patch (at the input of the preceding conv. layer) of size: (P+K-1)x(P+K-1)

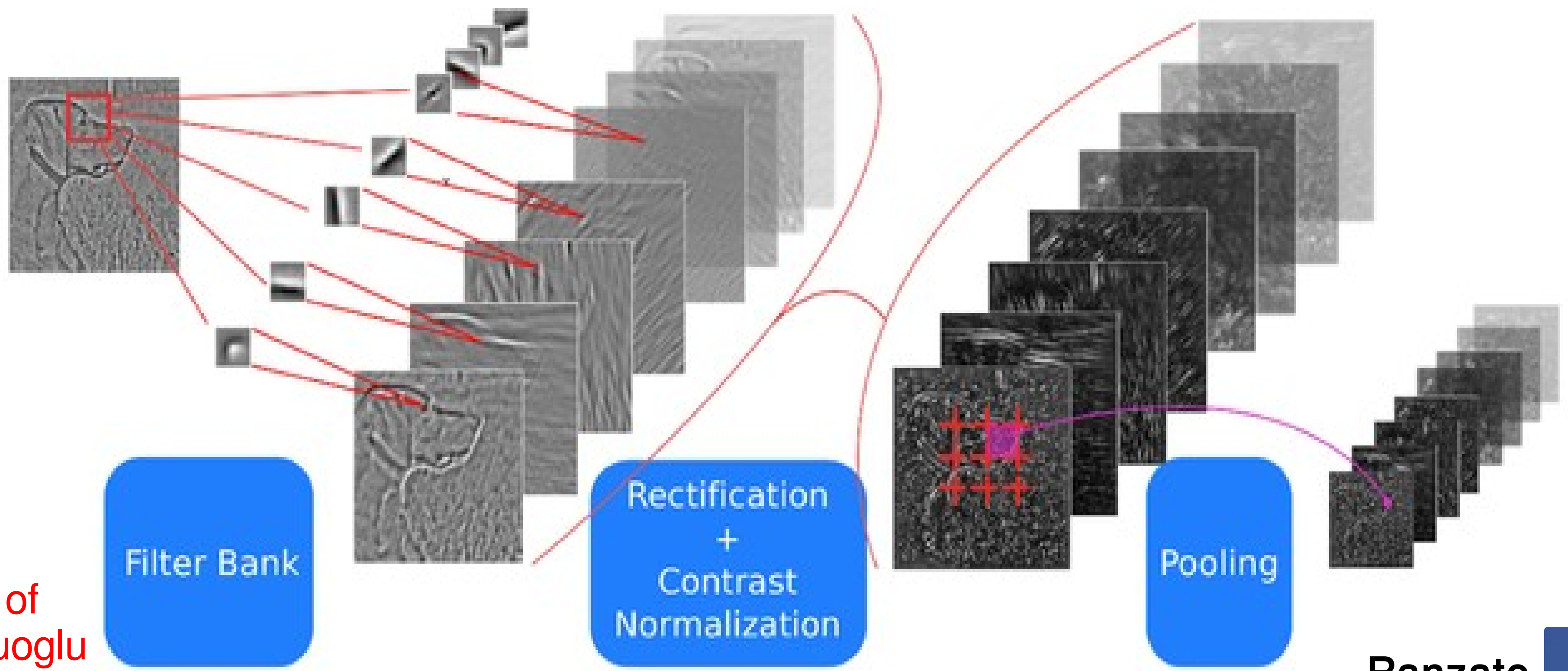


Ranzato

ConvNets: Typical Stage

One stage (zoom)



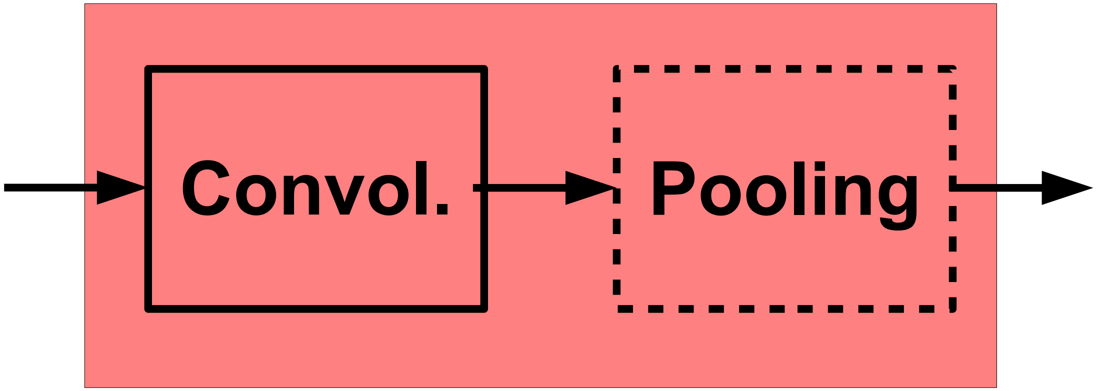


courtesy of
K. Kavukcuoglu

Ranzato

ConvNets: Typical Stage

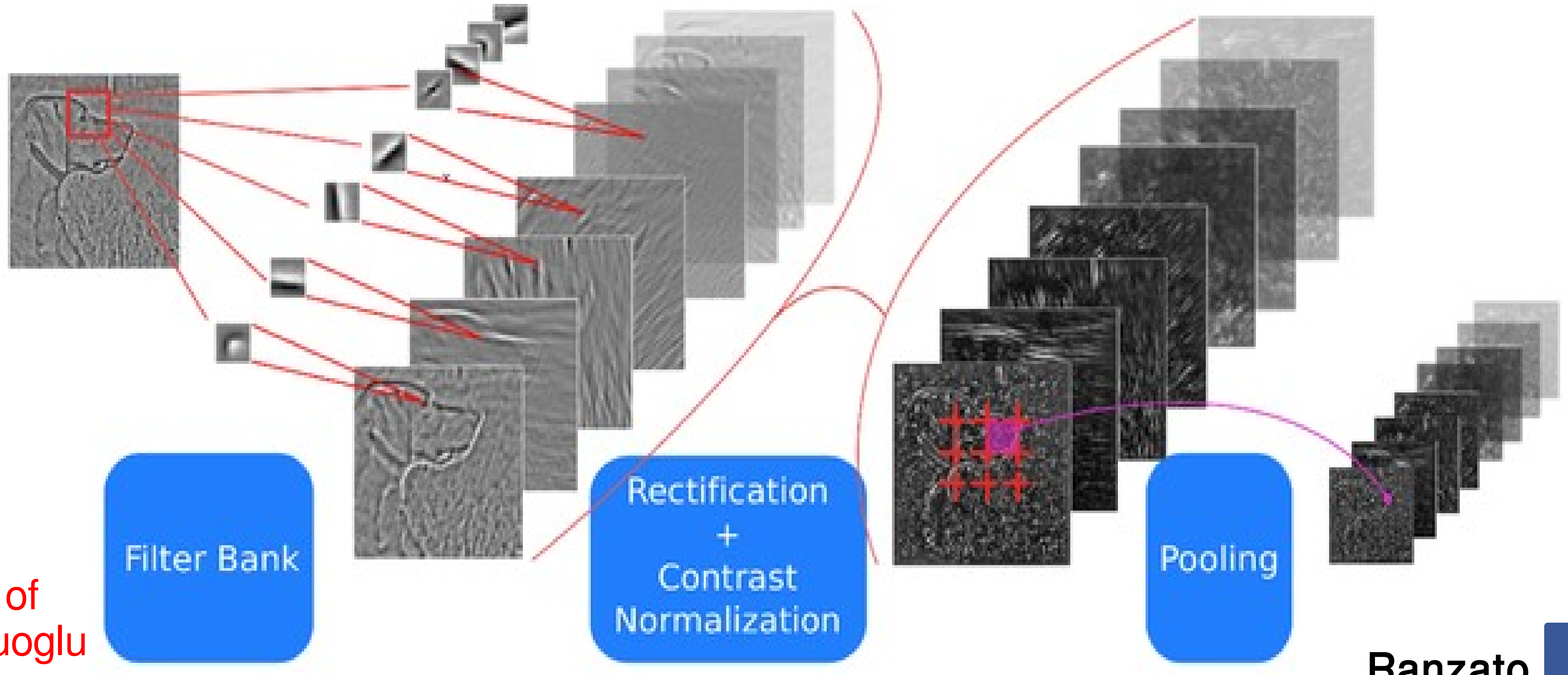
One stage (zoom)



Conceptually similar to: SIFT, HoG, etc.

Note: after one stage the number of feature maps is usually increased (conv. layer) and the spatial resolution is usually decreased (stride in conv. and pooling layers). Receptive field gets bigger.

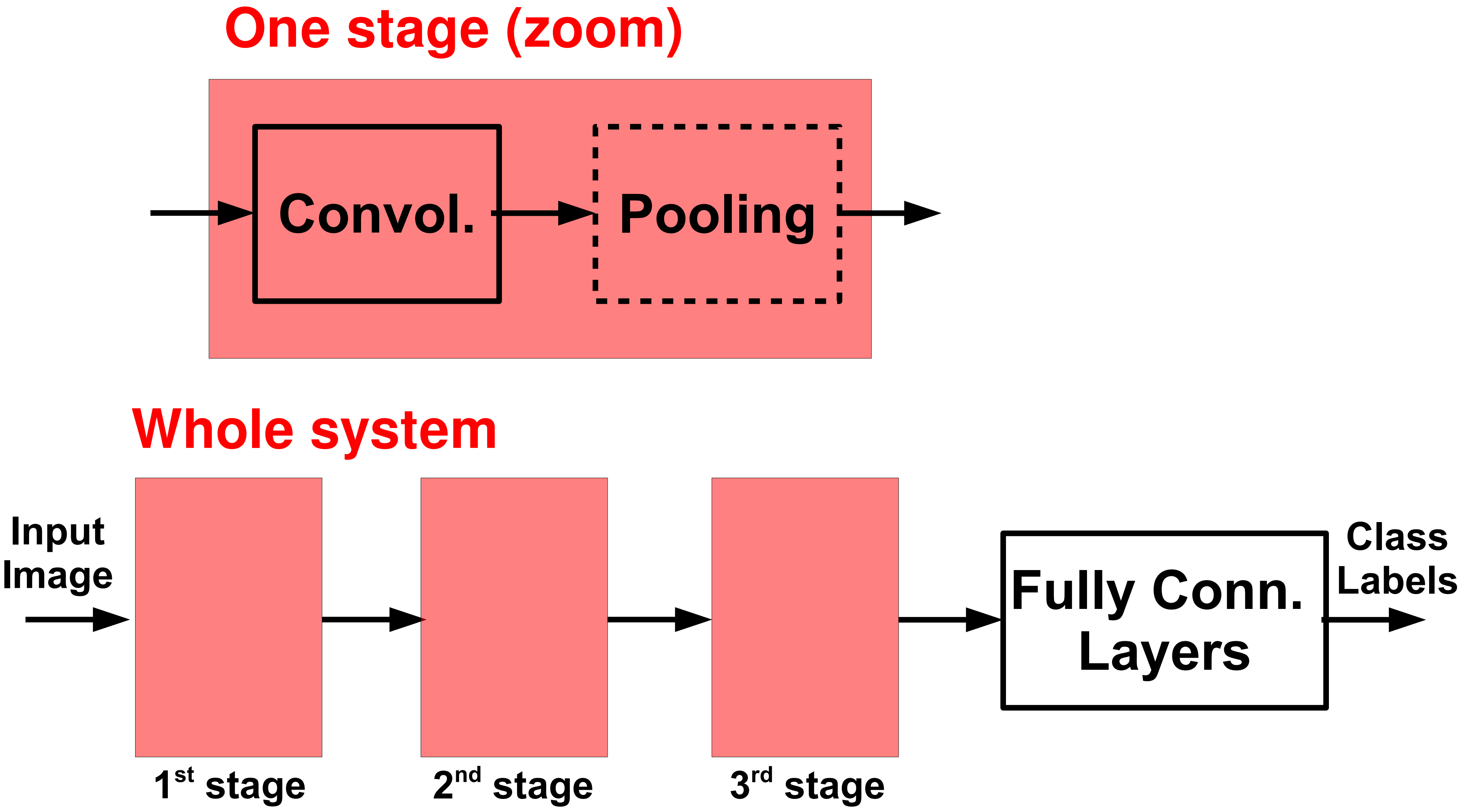
Reasons:
- gain invariance to spatial translation (pooling layer)
- increase specificity of features (approaching object specific units)



courtesy of
K. Kavukcuoglu

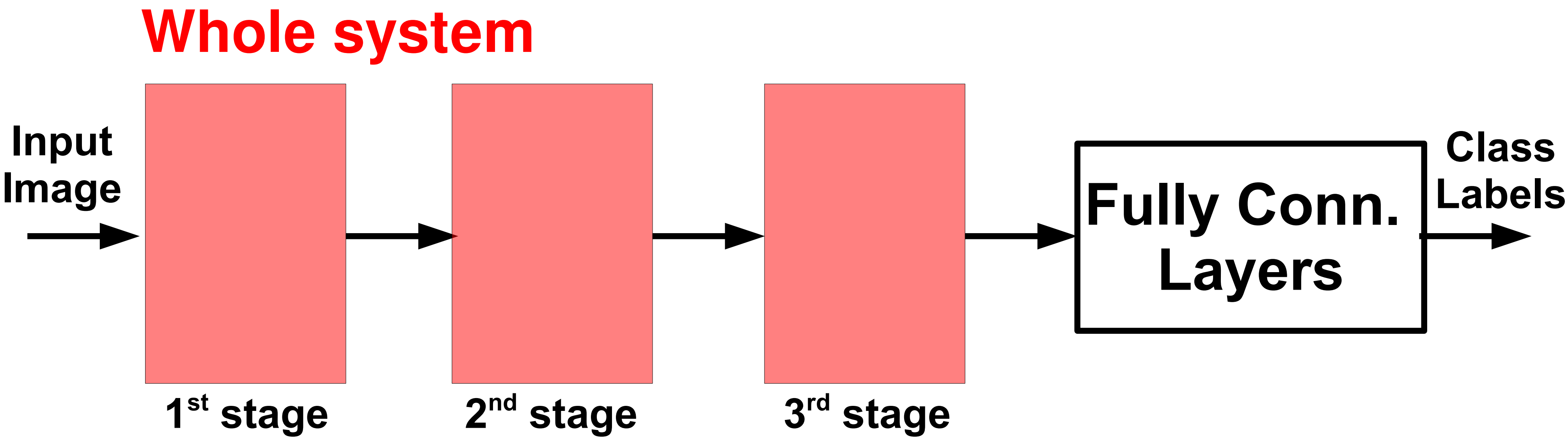
Ranzato

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.

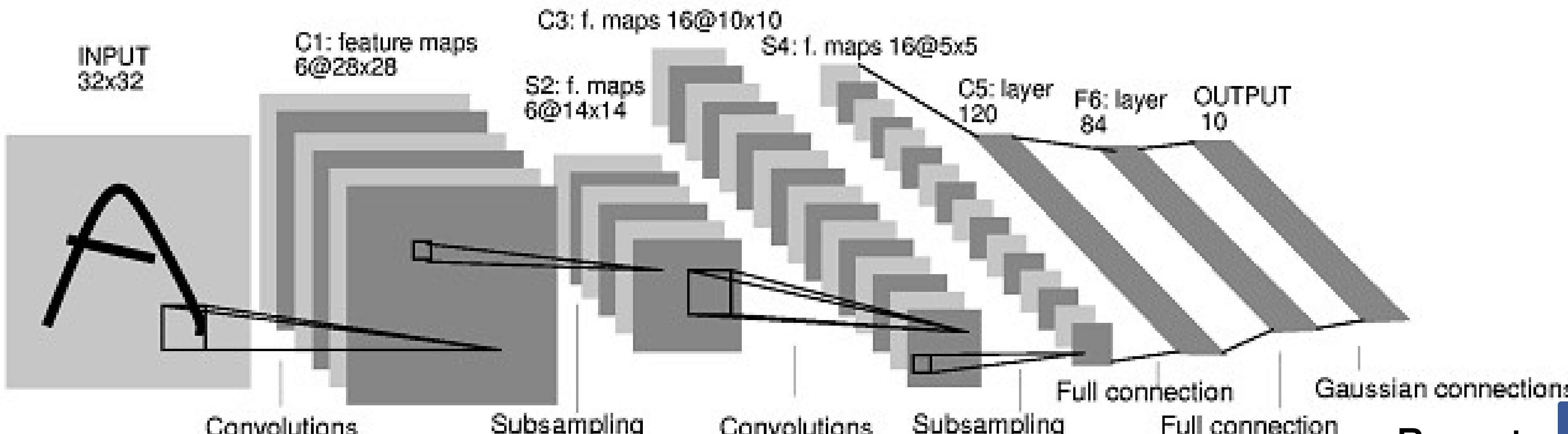
Algorithm:

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

Note: After several stages of convolution-pooling, the spatial resolution is greatly reduced (usually to about 5x5) and the number of feature maps is large (several hundreds depending on the application).

It would not make sense to convolve again (there is no translation invariance and support is too small). Everything is vectorized and fed into several fully connected layers.

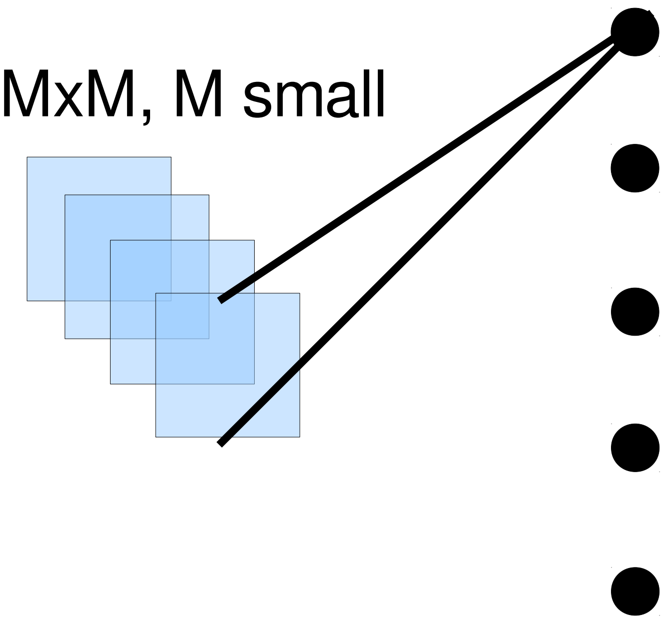
If the input of the fully connected layers is of size Nx5x5, the first fully connected layer can be seen as a conv. layer with 5x5 kernels.
The next fully connected layer can be seen as a conv. layer with 1x1 kernels.



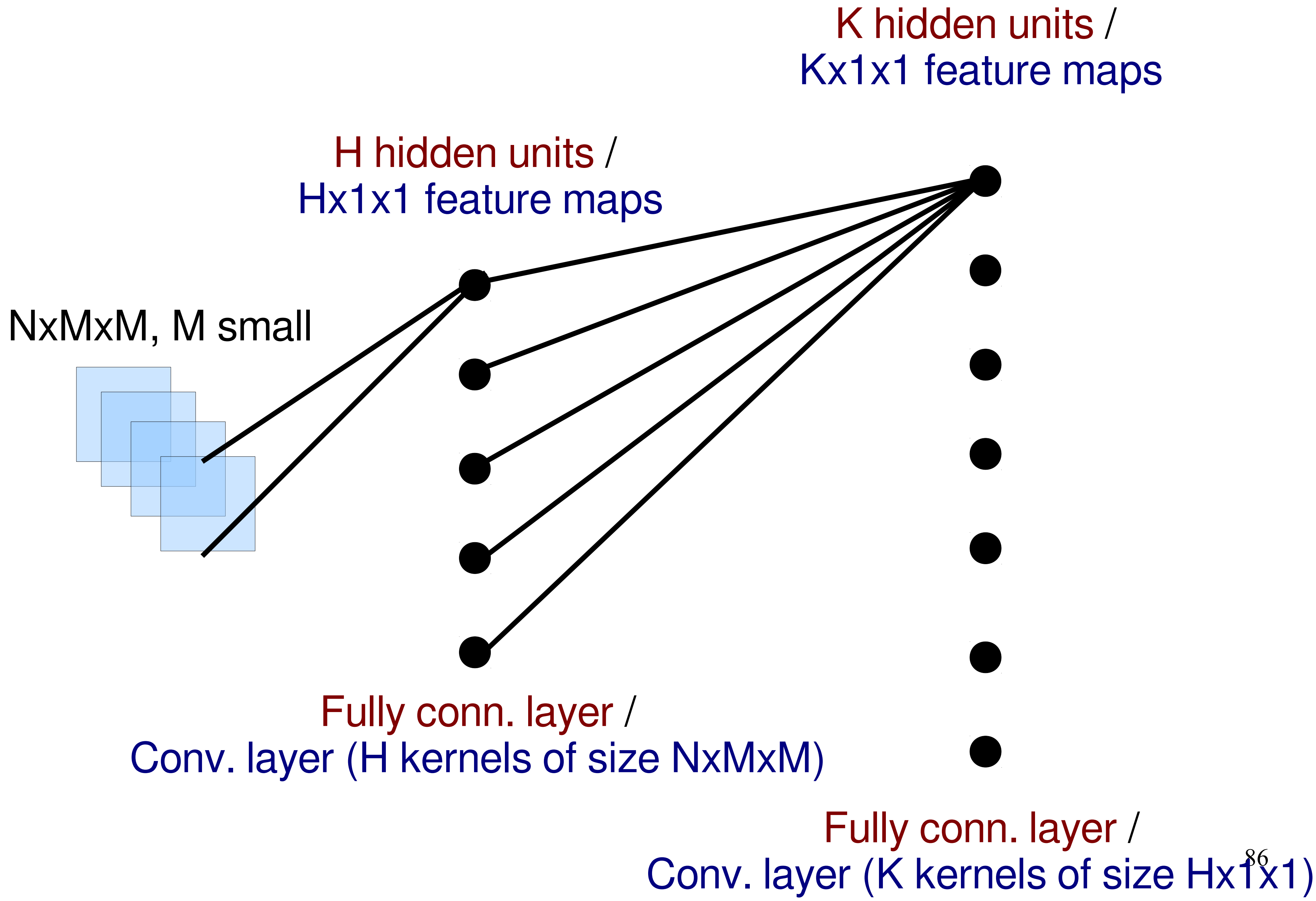
Ranzato

H hidden units /
Hx1x1 feature maps

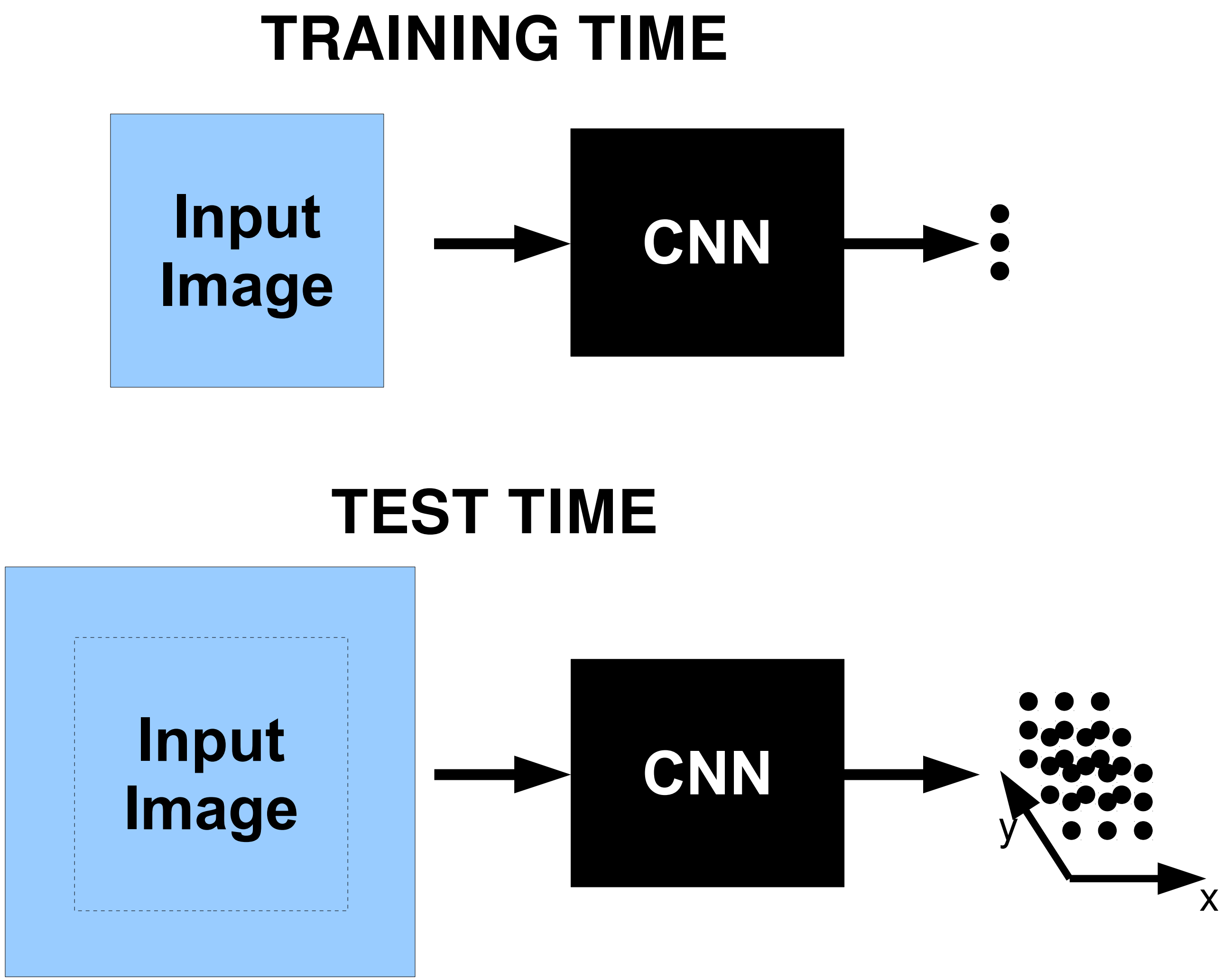
NxMxM, M small



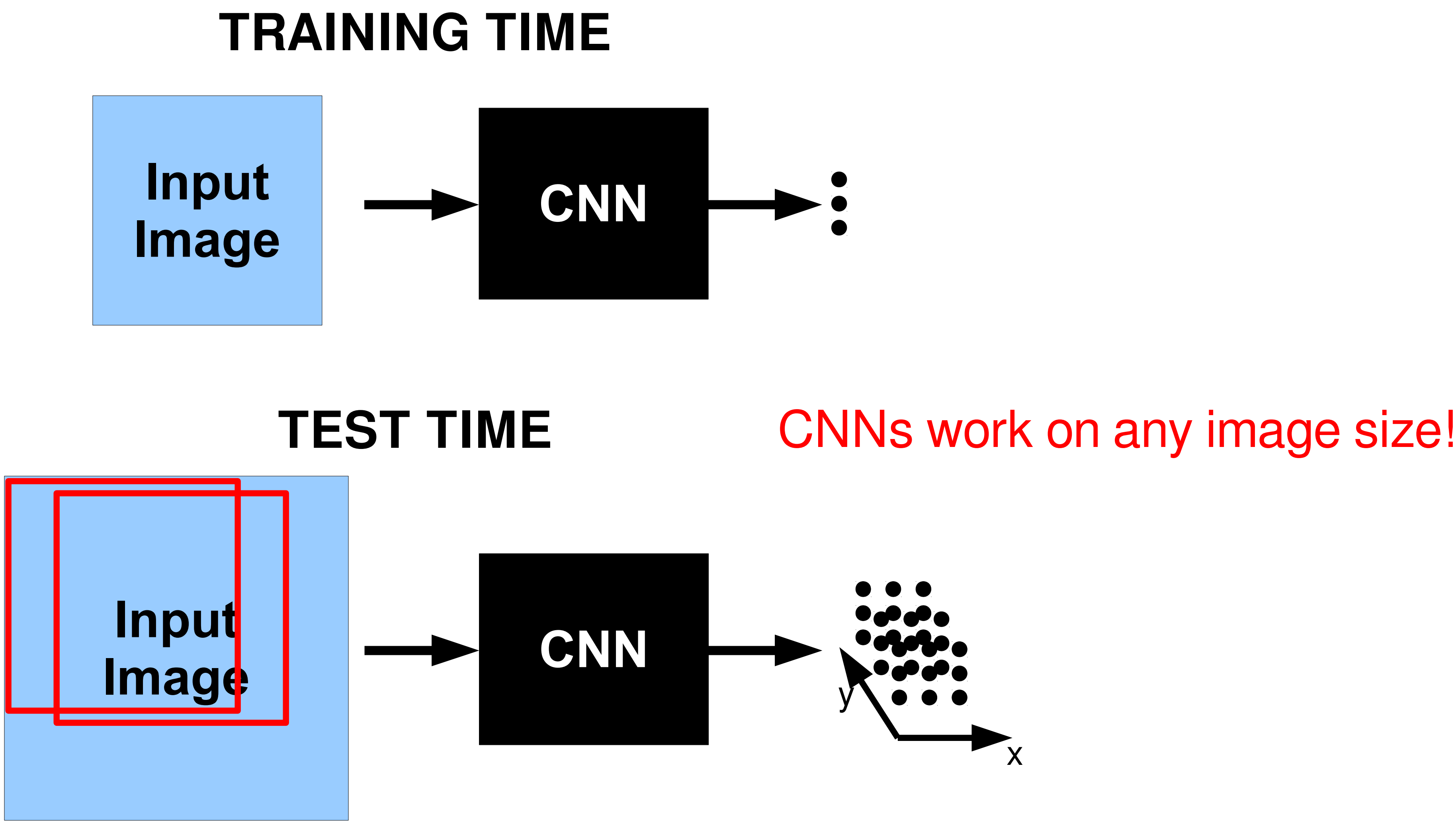
Fully conn. layer /
Conv. layer (H kernels of size NxMxM)


K hidden units /
Kx1x1 feature maps
H hidden units /
Hx1x1 feature maps
NxMxM, M small
Fully conn. layer /
Conv. layer (H kernels of size NxMxM)
Fully conn. layer /
Conv. layer (K kernels of size Hx1x1)

Viewing fully connected layers as convolutional layers enables efficient use of convnets on bigger images (no need to slide windows but unroll network over space as needed to re-use computation).



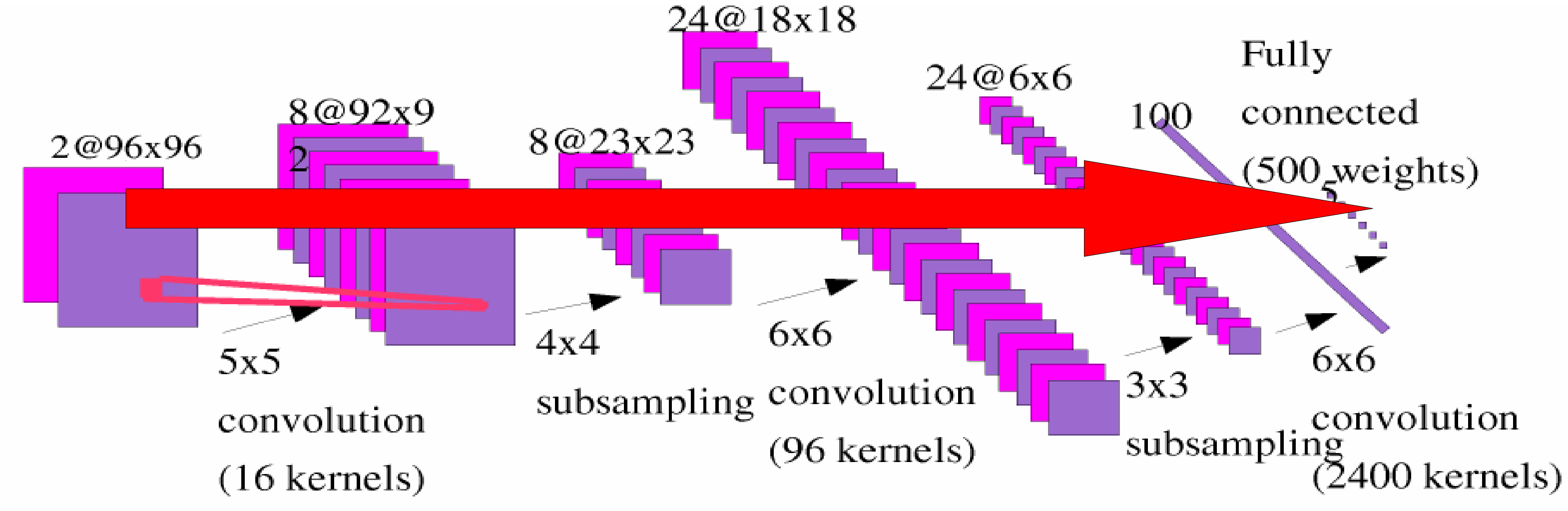
Viewing fully connected layers as convolutional layers enables efficient use of convnets on bigger images (no need to slide windows but unroll network over space as needed to re-use computation).



Unrolling is order of magnitudes more eficient than sliding windows!

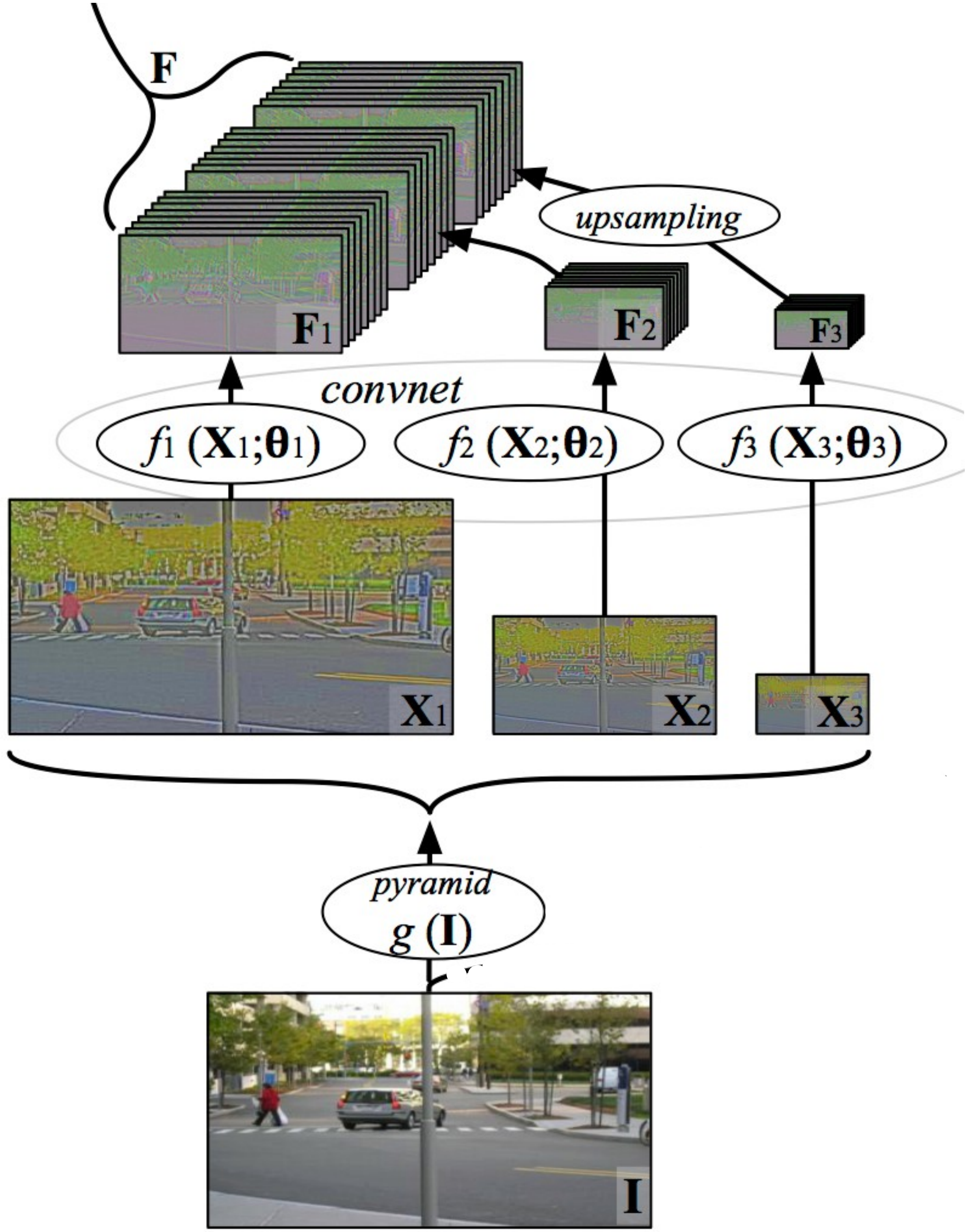
ConvNets: Test

At test time, run only is forward mode (FPROP).



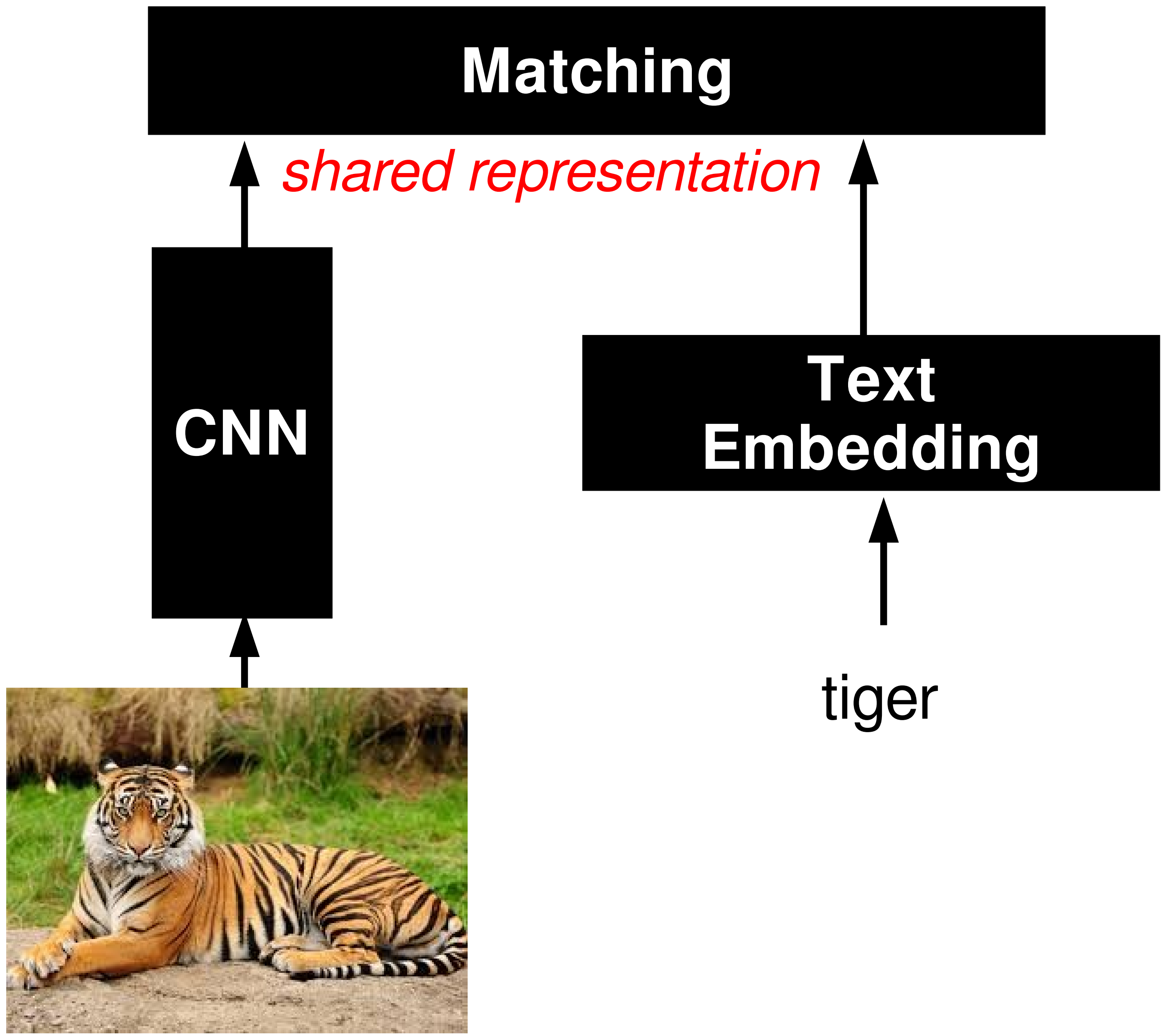
Ranzato

Fancier Architectures: Multi-Scale



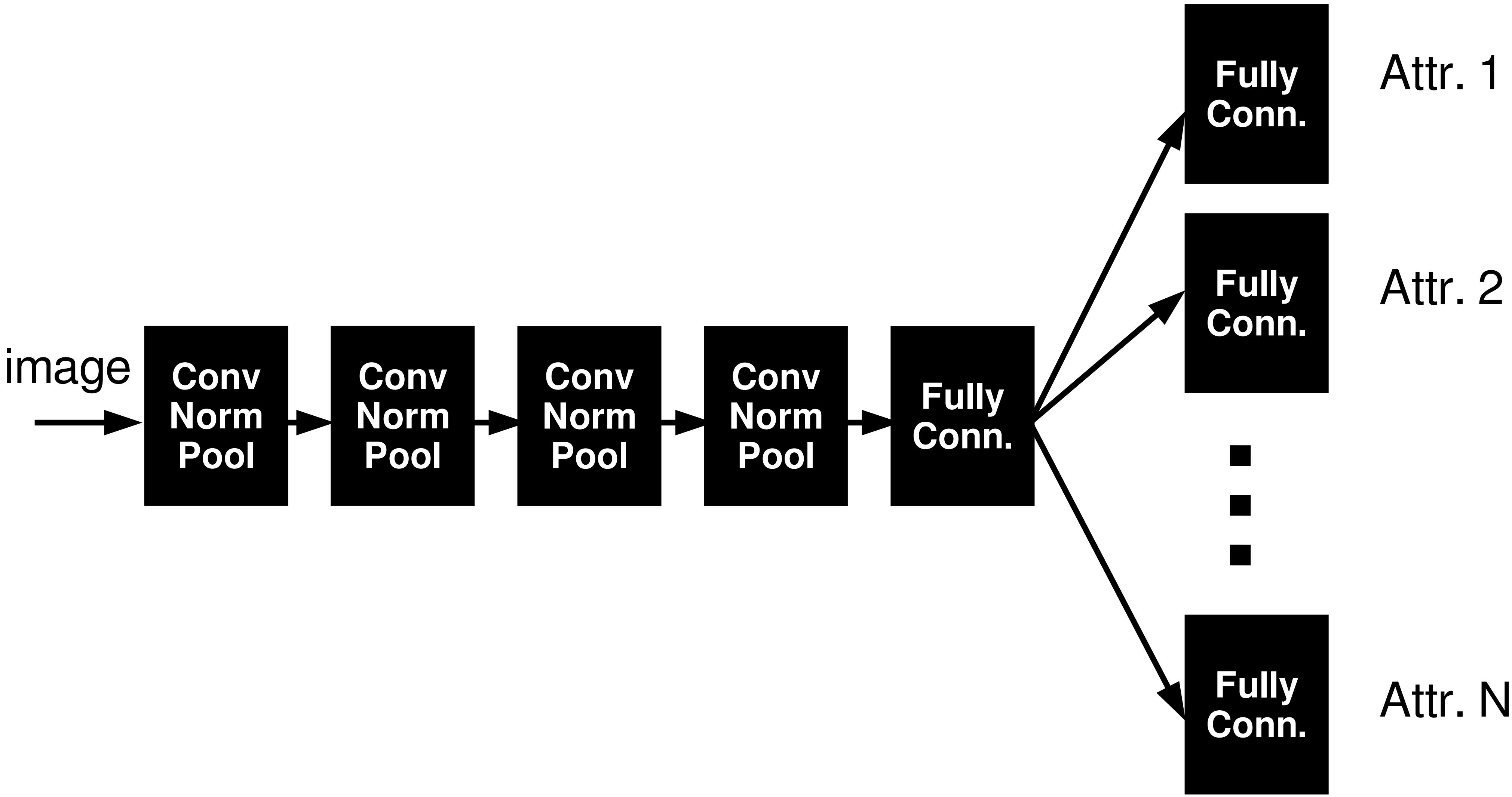
Farabet et al. “Learning hierarchical features for scene labeling” PAMI 2013

Fancier Architectures: Multi-Modal



Frome et al. “Devise: a deep visual semantic embedding model” NIPS 2013

Fancier Architectures: Multi-Task



Zhang et al. “PANDA..” CVPR 2014

Fancier Architectures: Multi-Task

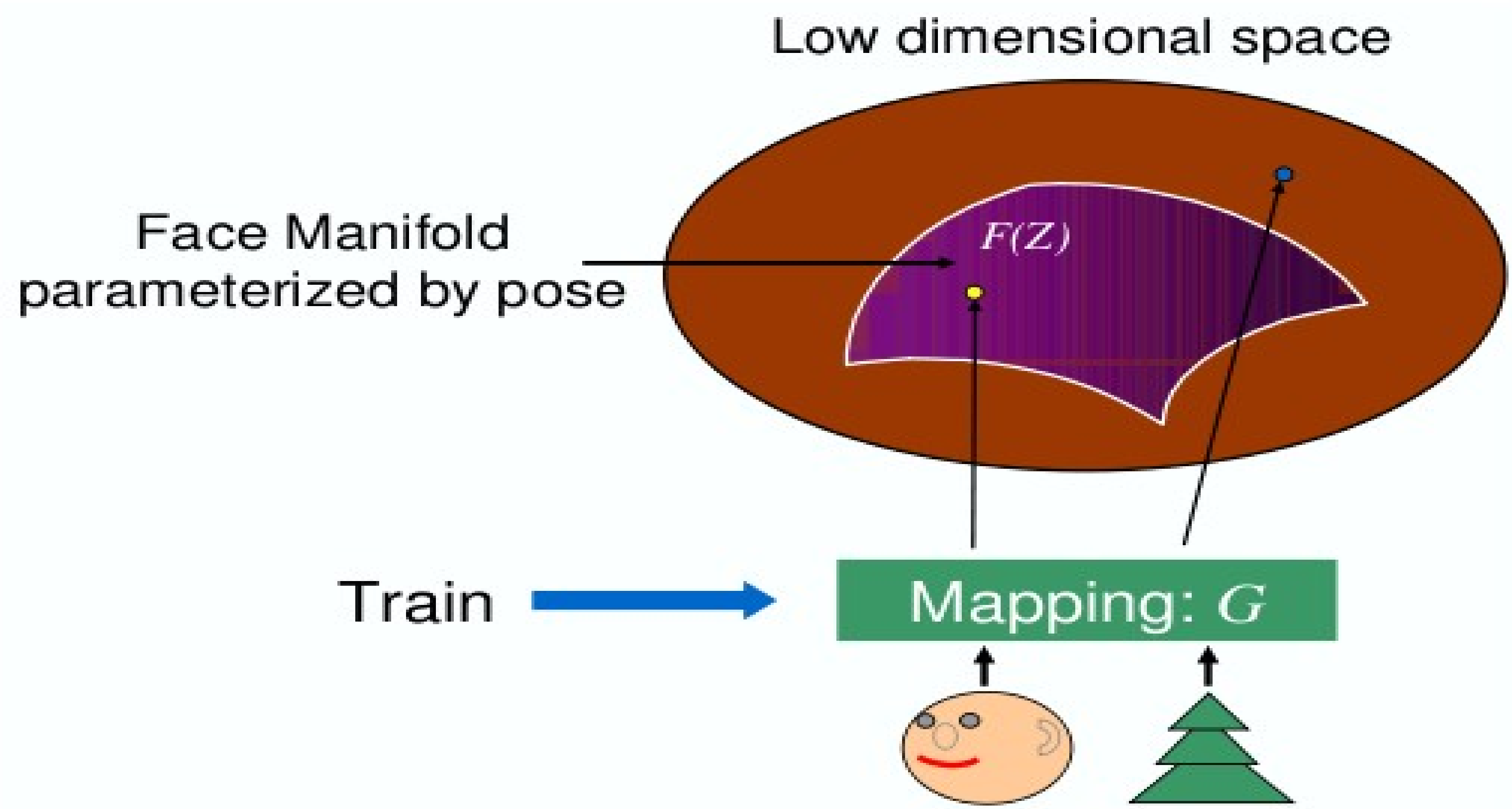
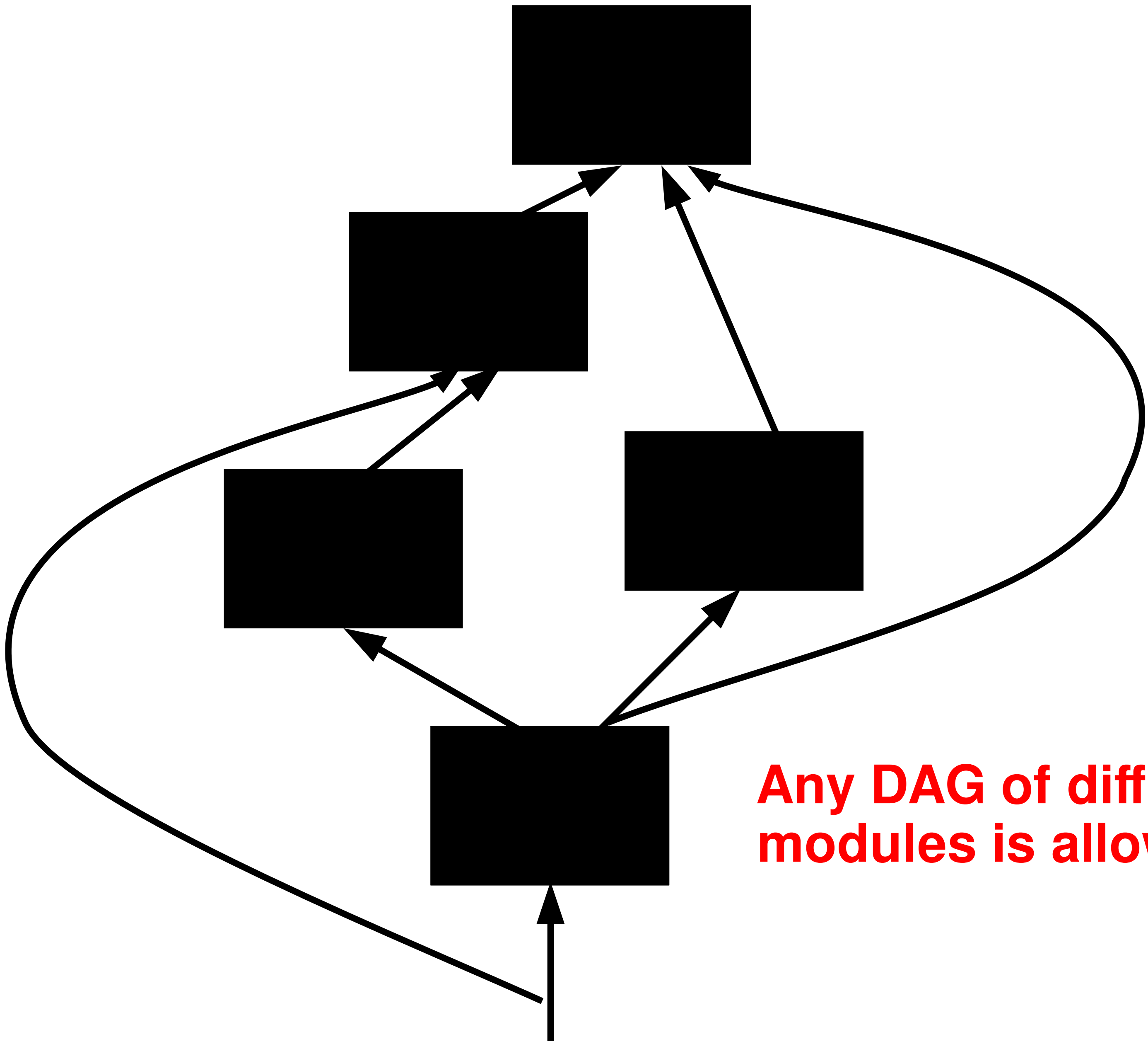


Figure 1: Manifold Mapping—Training

Osadchy et al. “Synergistic face detection and pose estimation..” JMLR 2007

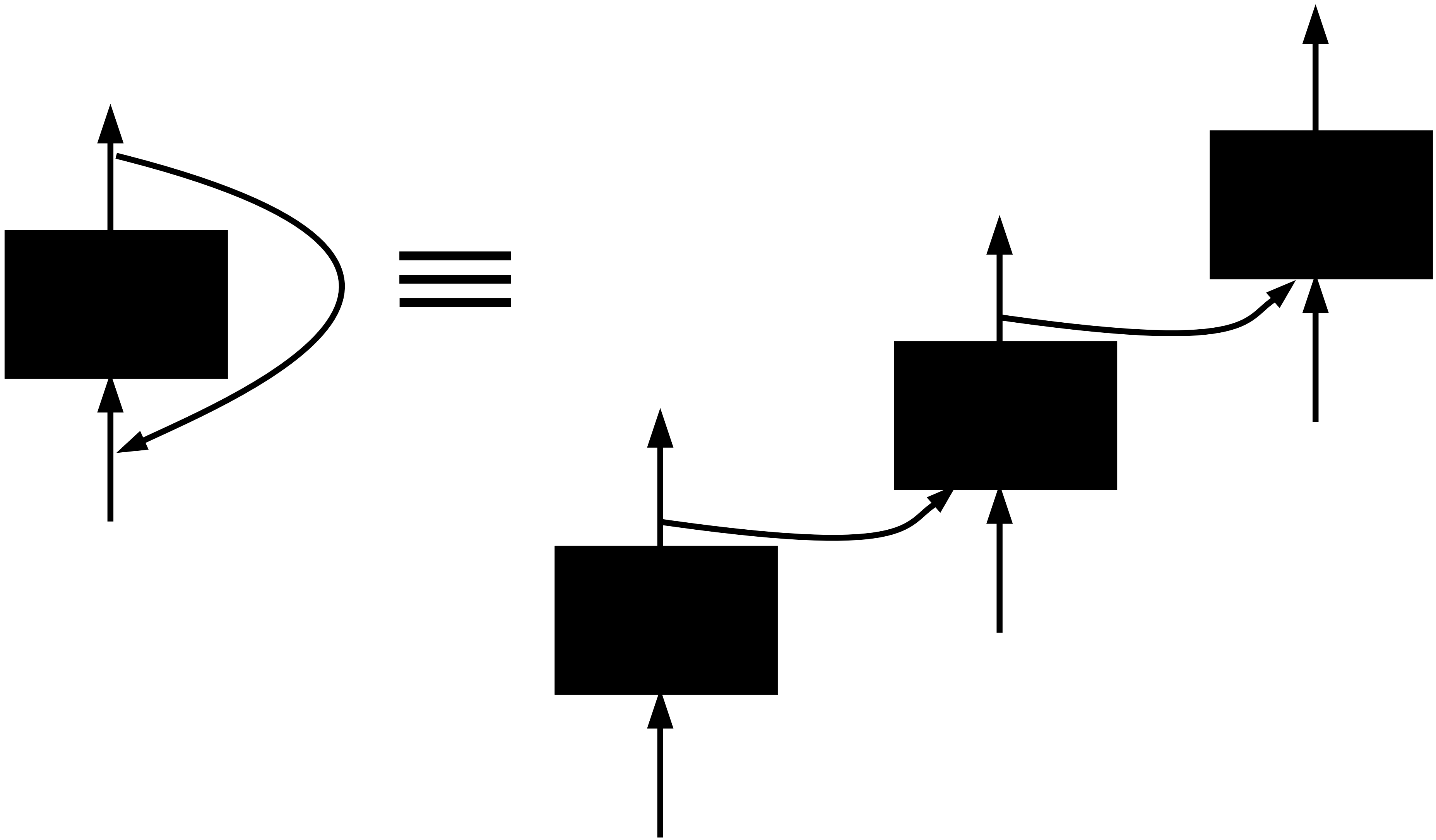
Fancier Architectures: Generic DAG



Any DAG of differentialble modules is allowed!

Fancier Architectures: Generic DAG

If there are cycles (RNN), one needs to un-roll it.



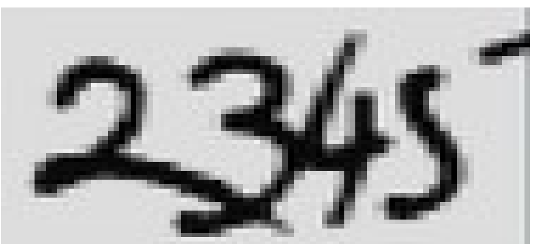
Pinheiro, Collobert "Recurrent CNN for scene labeling" ICML 2014
Graves "Offline Arabic handwriting recognition.." Springer 2012

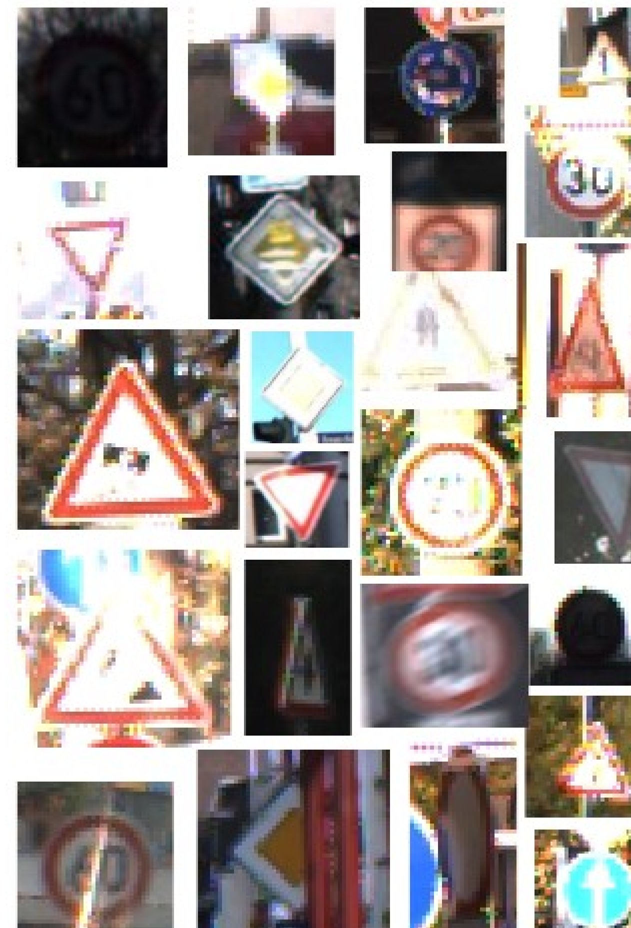
Outline

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

CONV NETS: EXAMPLES

- OCR / House number & Traffic sign classification

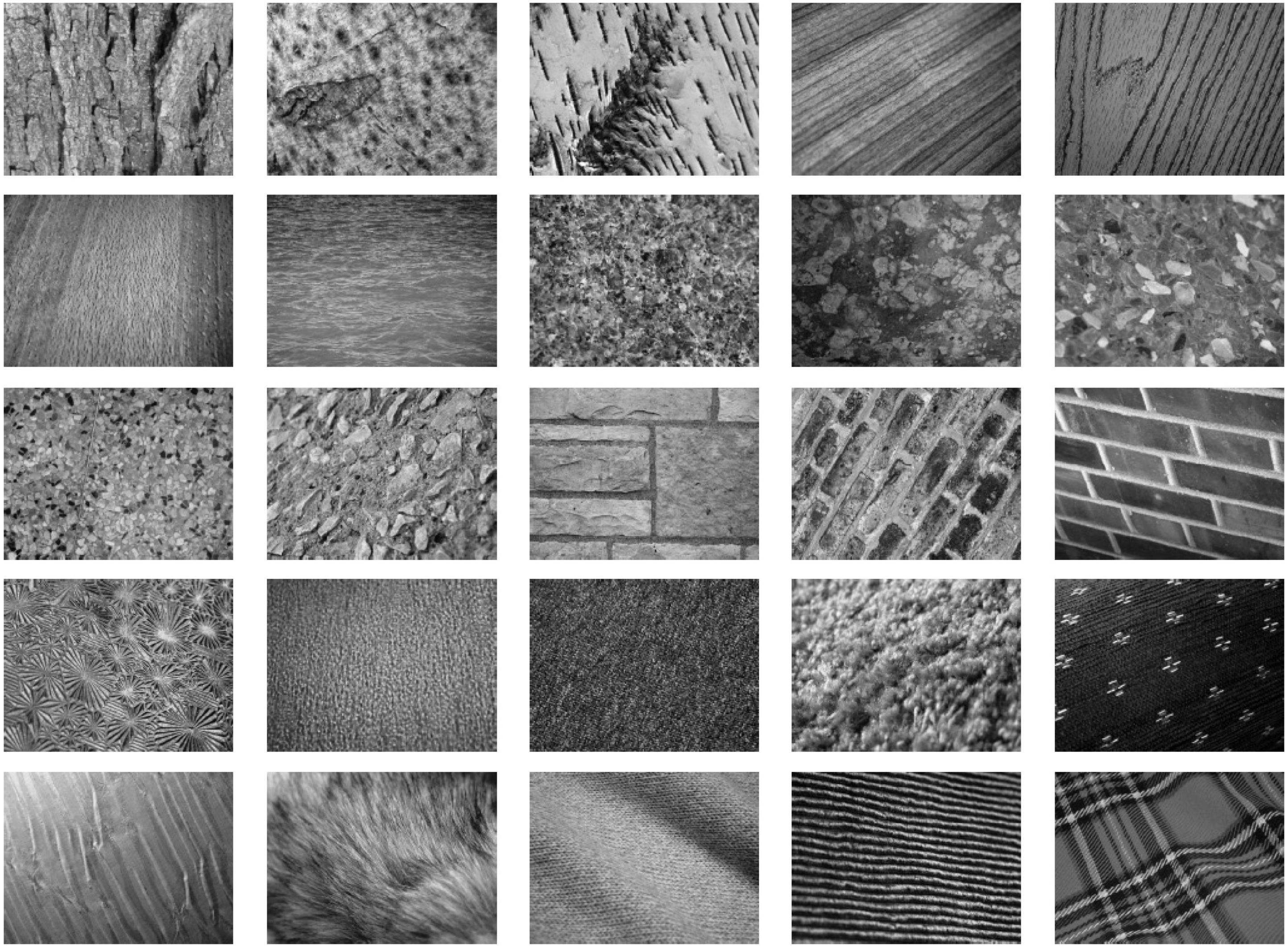




Ciresan et al. “MCDNN for image classification” CVPR 2012
Wan et al. “Regularization of neural networks using dropconnect” ICML 2013
Goodfellow et al. “Multi-digit nuber recognition from StreetView...” ICLR 2014
Jaderberg et al. “Synthetic data and ANN for natural scene text recognition” arXiv 2014

CONV NETS: EXAMPLES

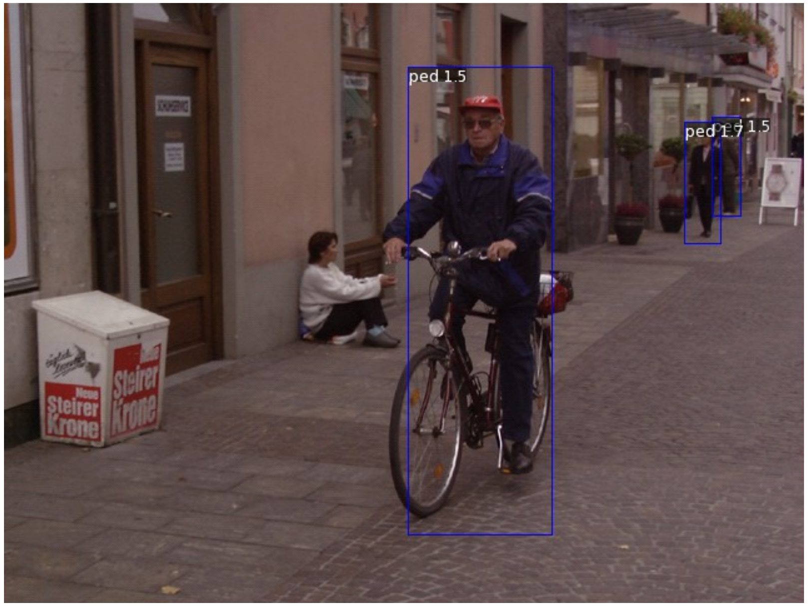
- **Texture classification**

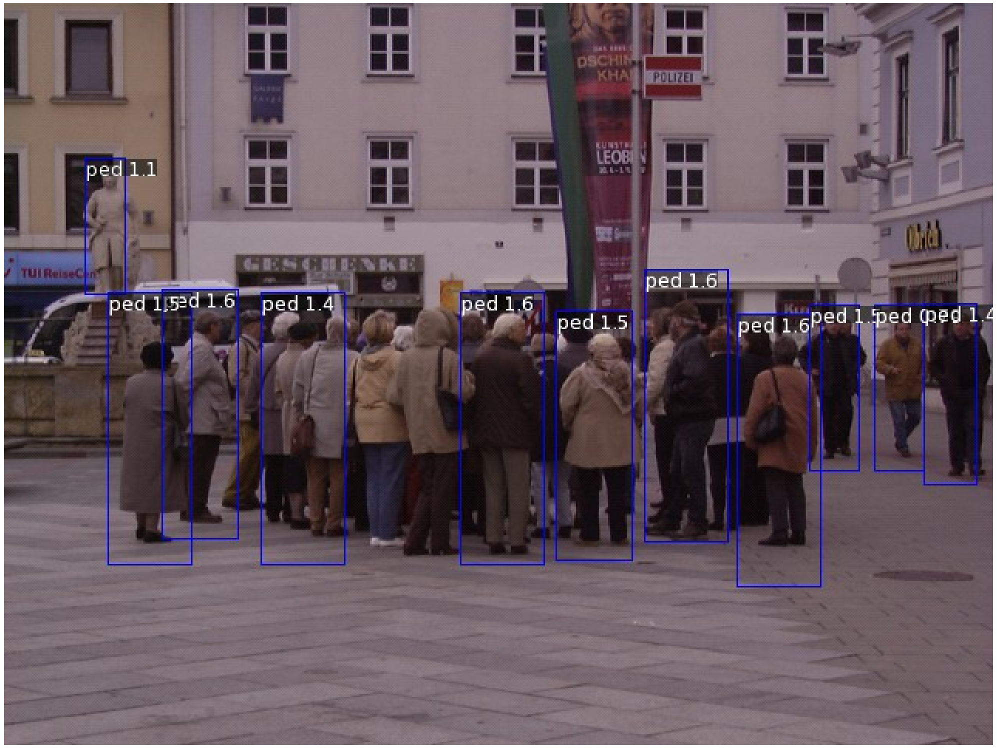


Sifre et al. “Rotation, scaling and deformation invariant scattering...” CVPR 2013

CONV NETS: EXAMPLES

- **Pedestrian detection**

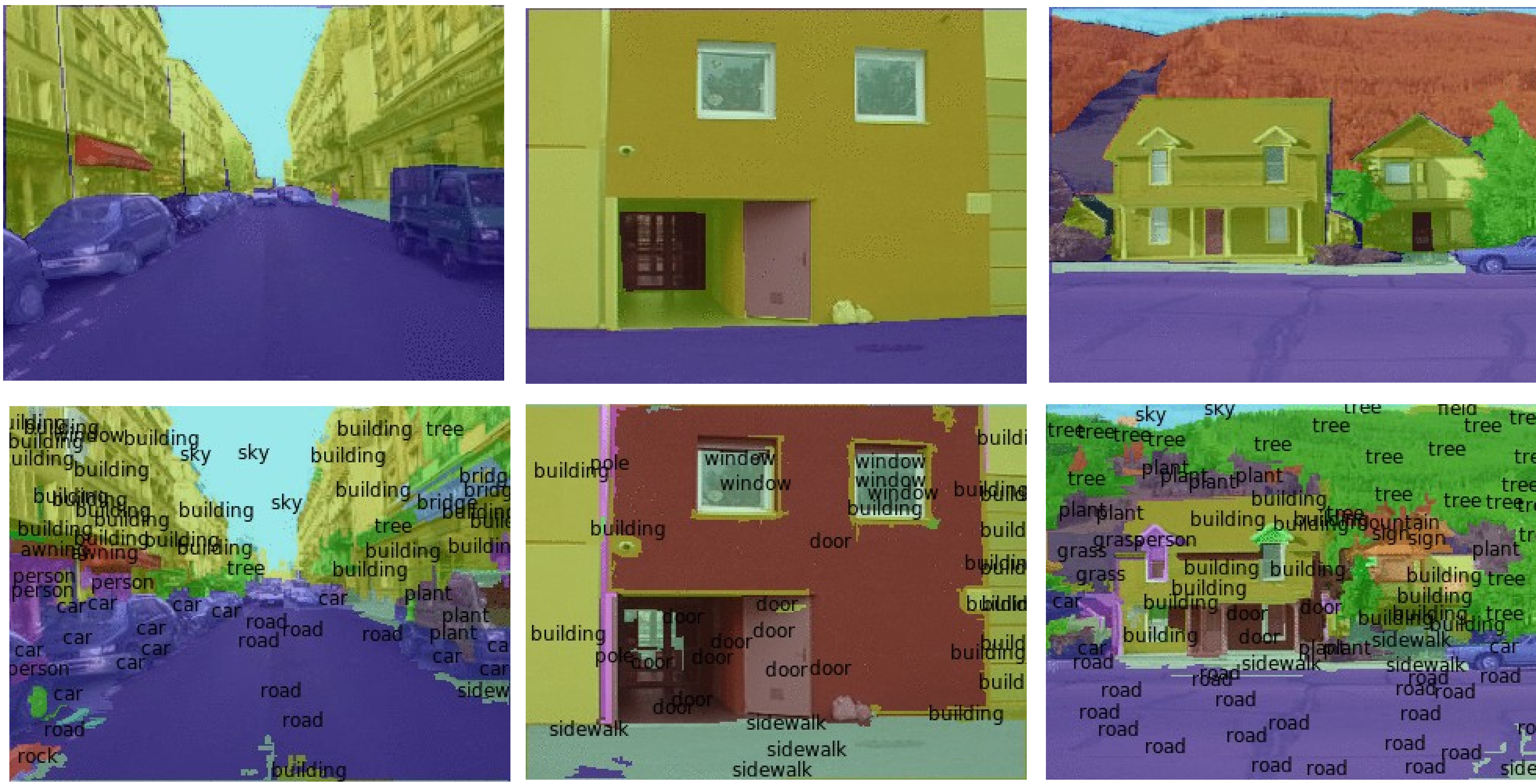




Sermanet et al. “Pedestrian detection with unsupervised multi-stage..” CVPR 2013

CONV NETS: EXAMPLES

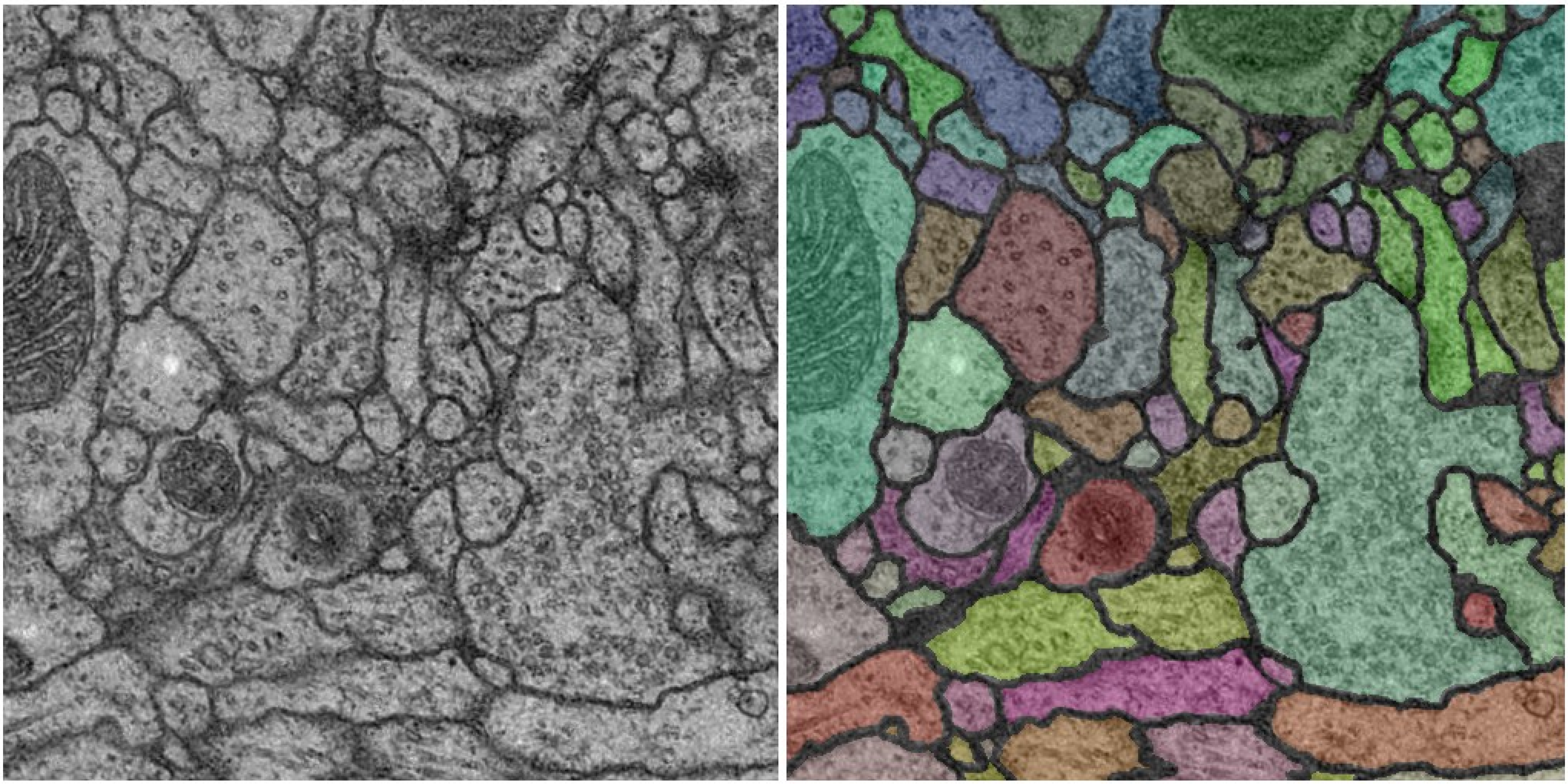
- Scene Parsing



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

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

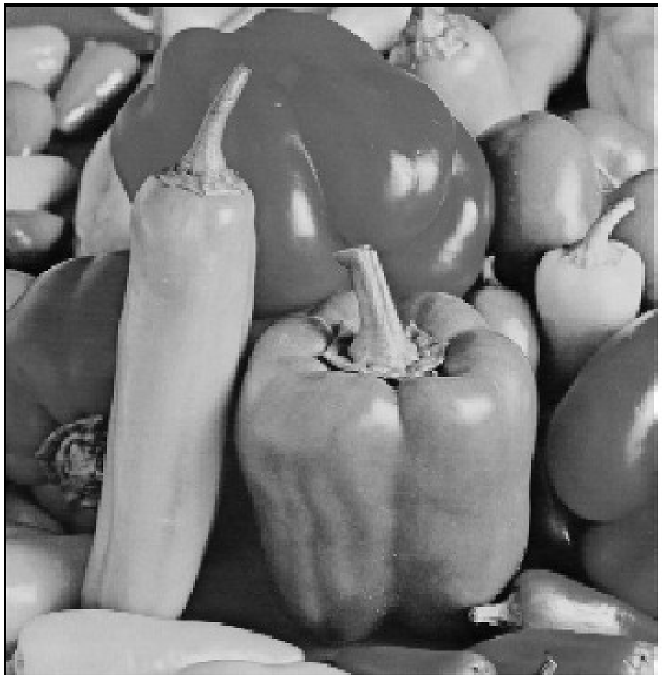
- **Robotics**

Sermanet et al. “Mapping and planning ...with long range perception” IROS 2008

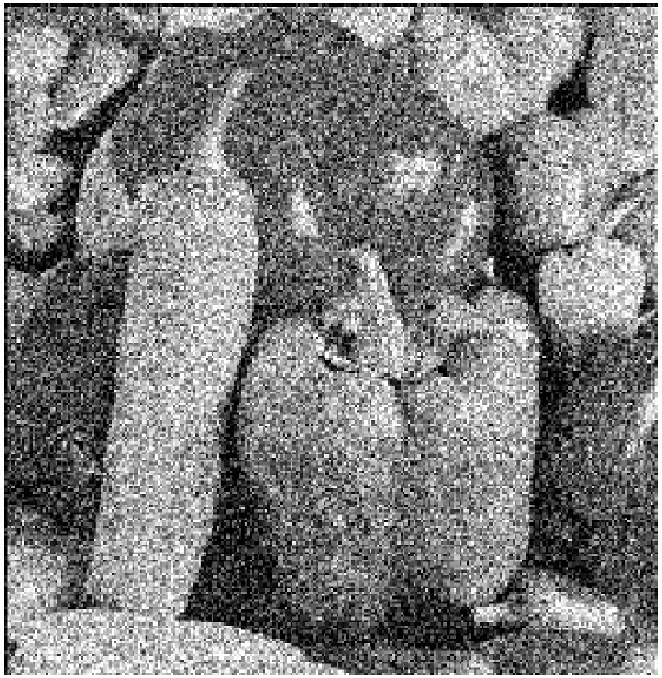
CONV NETS: EXAMPLES

- **Denoising**

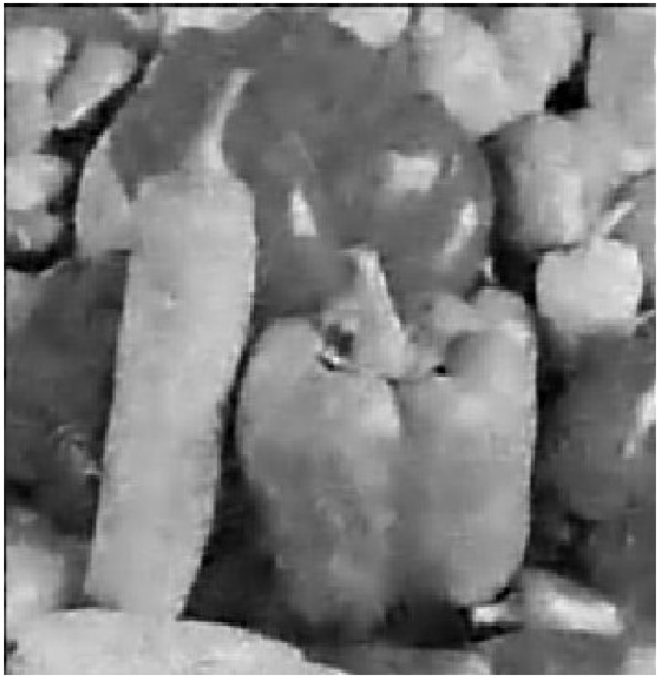
original



noised



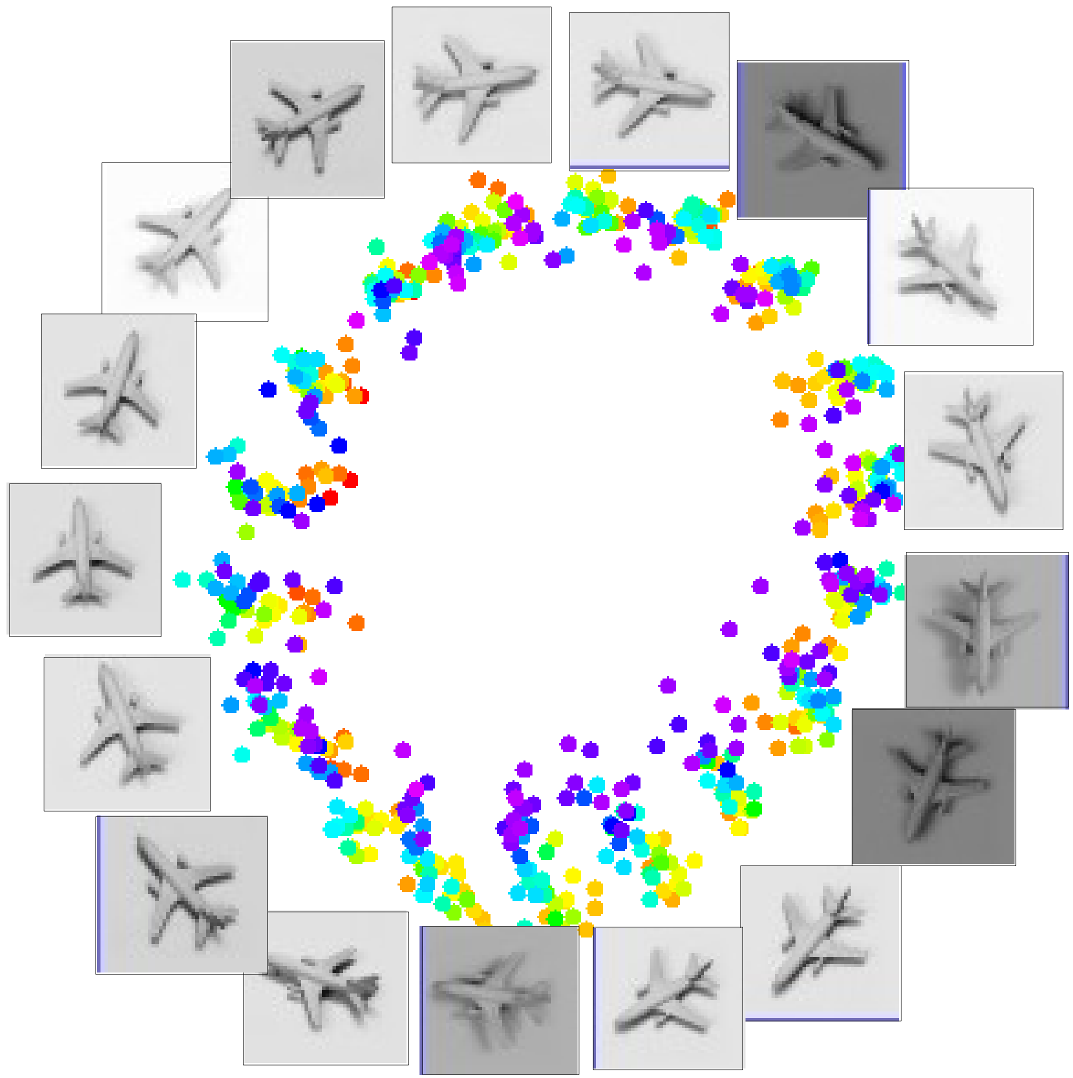
denoised



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

CONV NETS: EXAMPLES

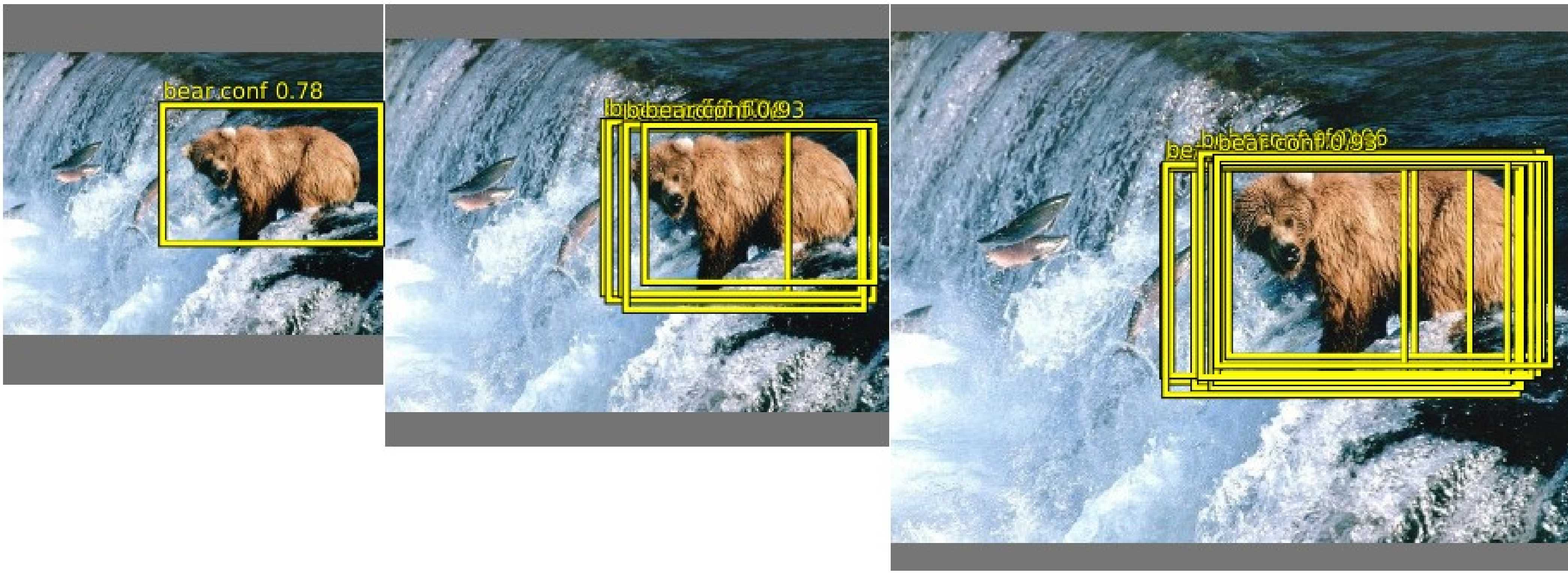
- **Dimensionality reduction / learning embeddings**



Hadsell et al. “Dimensionality reduction by learning an invariant mapping” CVPR 2006

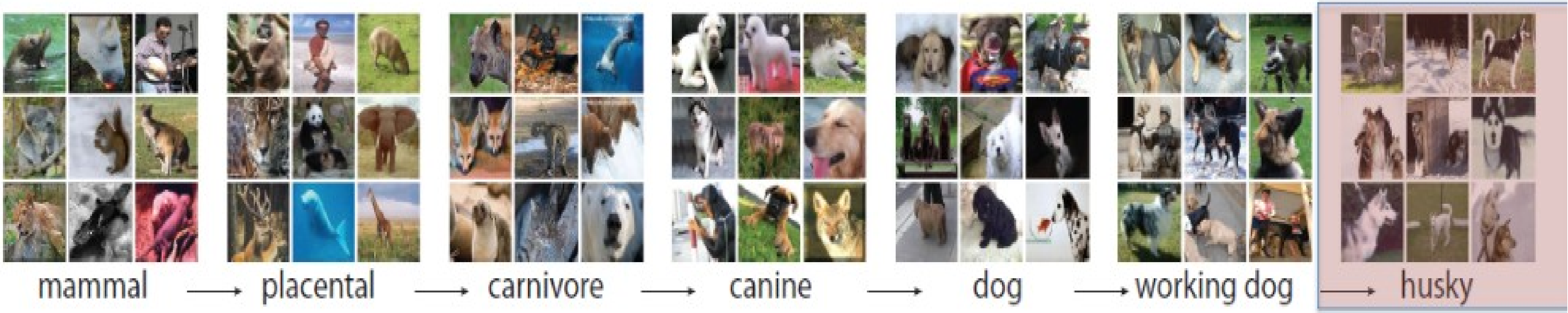
CONV NETS: EXAMPLES

- **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

Dataset: ImageNet 2012

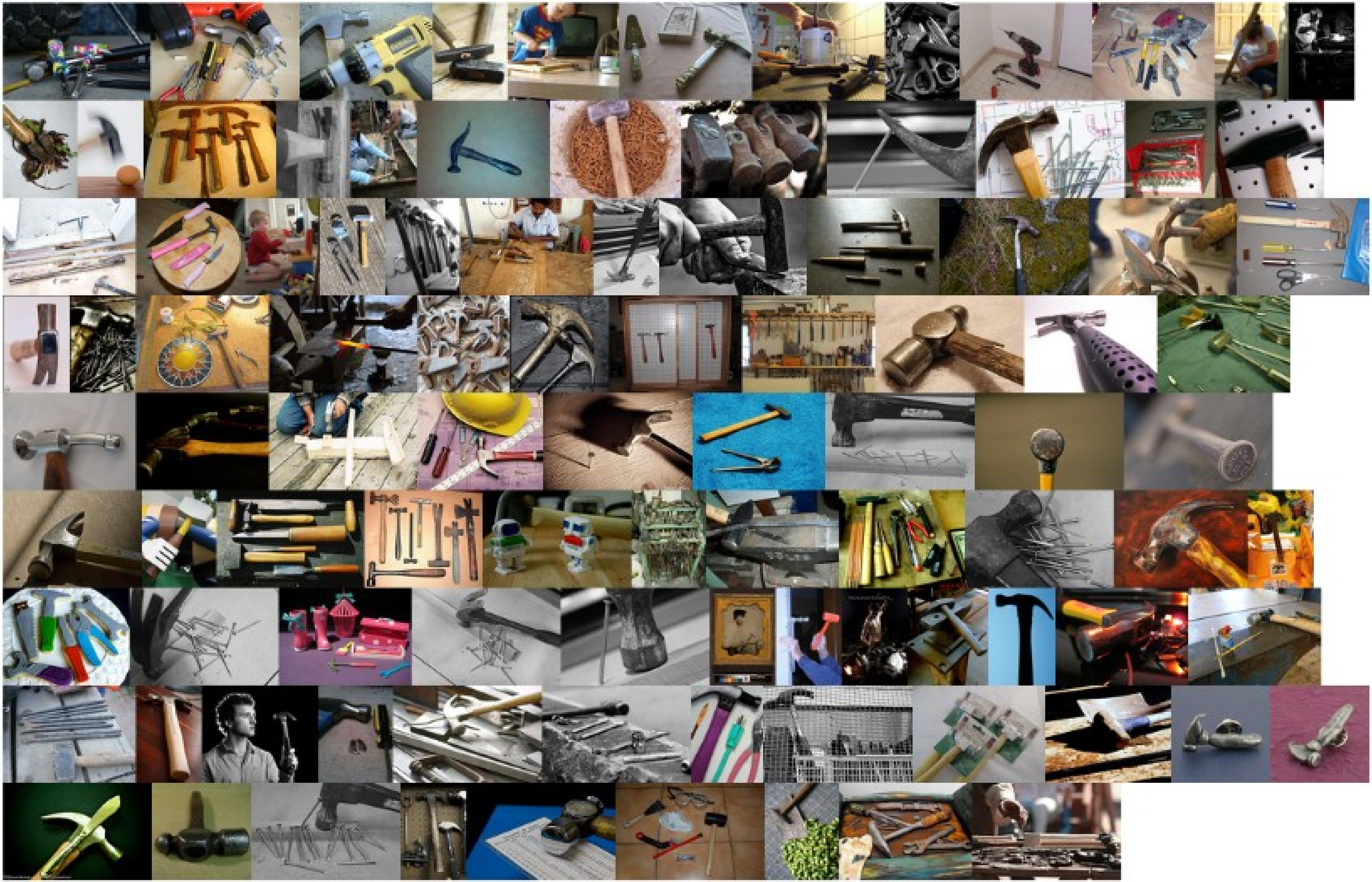


- S: (n) Eskimo dog, **husky** (breed of heavy-coated Arctic sled dog)
  - *direct hypernym* / ***inherited hypernym*** / *sister term*
    - S: (n) working dog (any of several breeds of usually large powerful dogs bred to work as draft animals and guard and guide dogs)
      - S: (n) dog, domestic dog,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

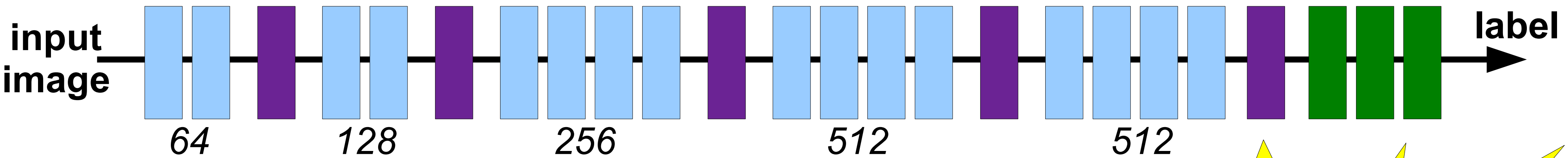
Deng et al. “Imagenet: a large scale hierarchical image database” CVPR 2009

ImageNet

Examples of hammer:



Architecture for Classification

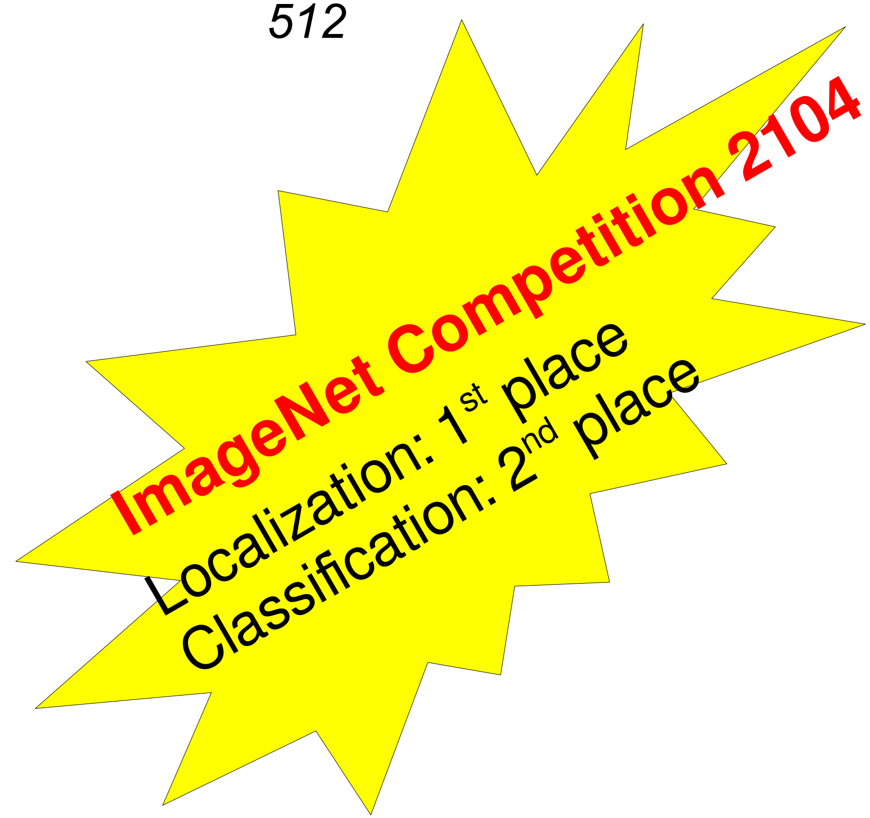


Conv. layer: 3x3 filters

Max pooling layer: 2x2, stride 2

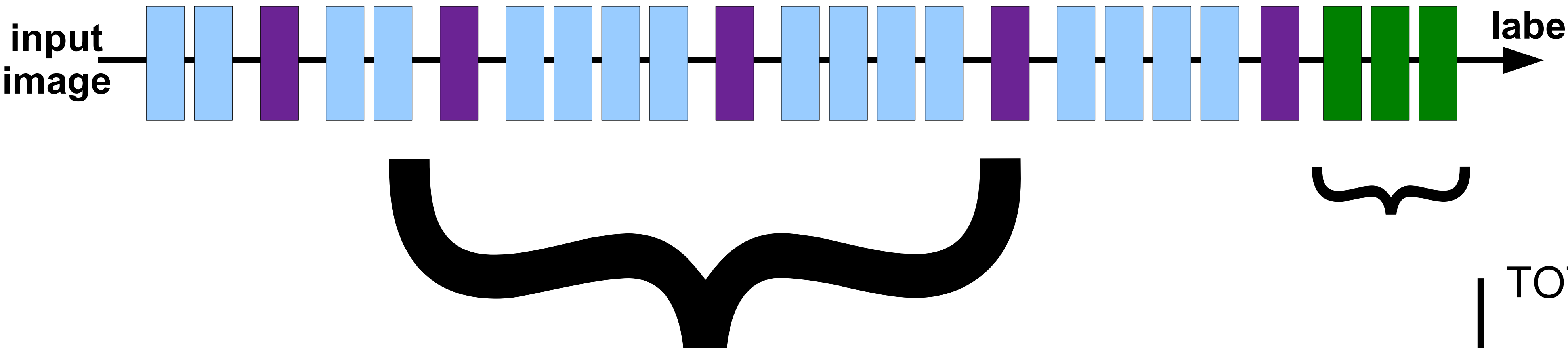
Fully connected layer: 4096 hiddens

24 Layers in total!!!



LeCun et al. “Gradient-based learning applied to OCR ” IEEE 1998
Krizhevsky et al. “ImageNet Classification with deep CNNs” NIPS 2012
Simonyan, Zisserman “Very deep CNN for large scale image recognition” ICLR15

Architecture for Classification

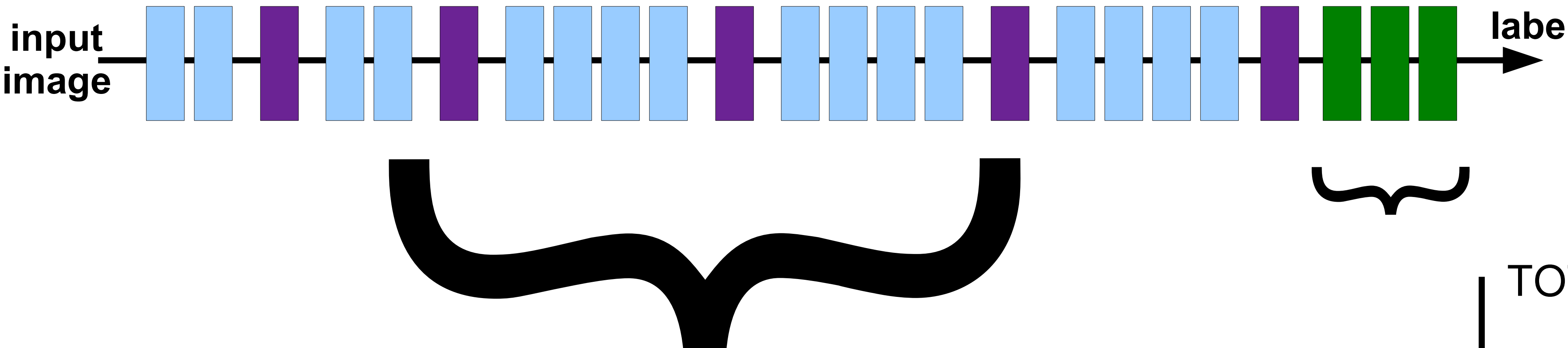


FLOPS: **20G** 0.1G

TOTAL
20G

LeCun et al. “Gradient-based learning applied to OCR ” IEEE 1998
Krizhevsky et al. “ImageNet Classification with deep CNNs” NIPS 2012
Simonyan, Zisserman “Very deep CNN for large scale image recognition” ICLR15

Architecture for Classification

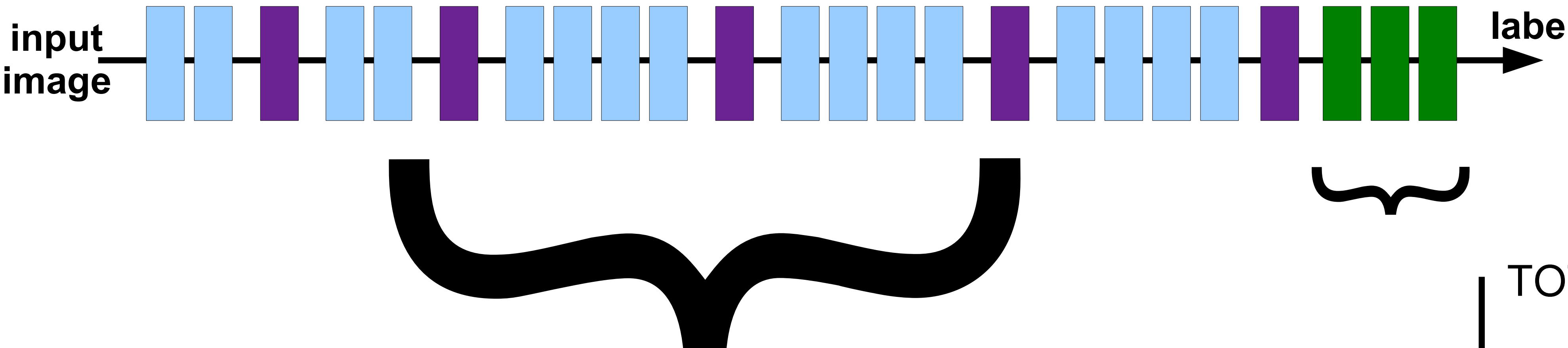


TOTAL

Nr. of parameters: 21M **123M** | 144M

LeCun et al. “Gradient-based learning applied to OCR ” IEEE 1998
Krizhevsky et al. “ImageNet Classification with deep CNNs” NIPS 2012
Simonyan, Zisserman “Very deep CNN for large scale image recognition” ICLR15

Architecture for Classification



TOTAL

Nr. of parameters: 21M 123M | 144M

Data augmentation is key to improve generalization:
- random translation
- left/right flipping
- scaling

LeCun et al. “Gradient-based learning applied to OCR ” IEEE 1998
Krizhevsky et al. “ImageNet Classification with deep CNNs” NIPS 2012
Simonyan, Zisserman “Very deep CNN for large scale image recognition” ICLR15

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

Outline

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

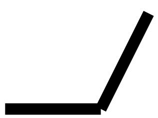
Choosing The Architecture

- Task dependent
- Cross-validation
- [Convolution $\rightarrow$ pooling]* + fully connected layer
- The more data: the more layers and the more kernels
  - Look at the number of parameters at each layer
  - Look at the number of flops at each layer
- Computational resources
- Be creative :)

How To Optimize

- SGD (with momentum) usually works very well

- Pick learning rate by running on a subset of the data
  Bottou "Stochastic Gradient Tricks" Neural Networks 2012
  - Start with large learning rate and divide by 2 until loss does not diverge
  - Decay learning rate by a factor of ~1000 or more by the end of training

- Use  non-linearity

- Initialize parameters so that each feature across layers has similar variance. Avoid units in saturation.

Improving Generalization

- Weight sharing (greatly reduce the number of parameters)

- Data augmentation (e.g., jittering, noise injection, etc.)

- Dropout
  Hinton et al. "Improving Nns by preventing co-adaptation of feature detectors" arxiv 2012

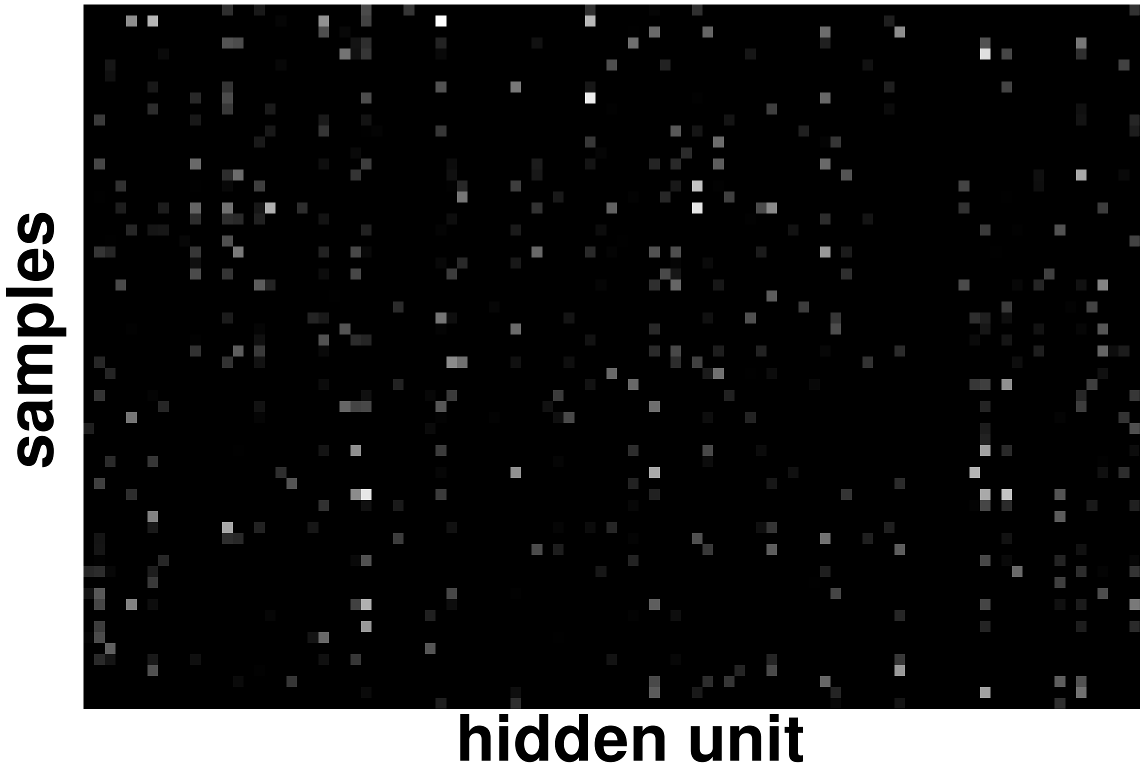
- Weight decay (L2, L1)

- Sparsity in the hidden units

- Multi-task (unsupervised learning)

Good To Know

- Check gradients numerically by finite differences
- Visualize features (feature maps need to be uncorrelated) and have high variance.

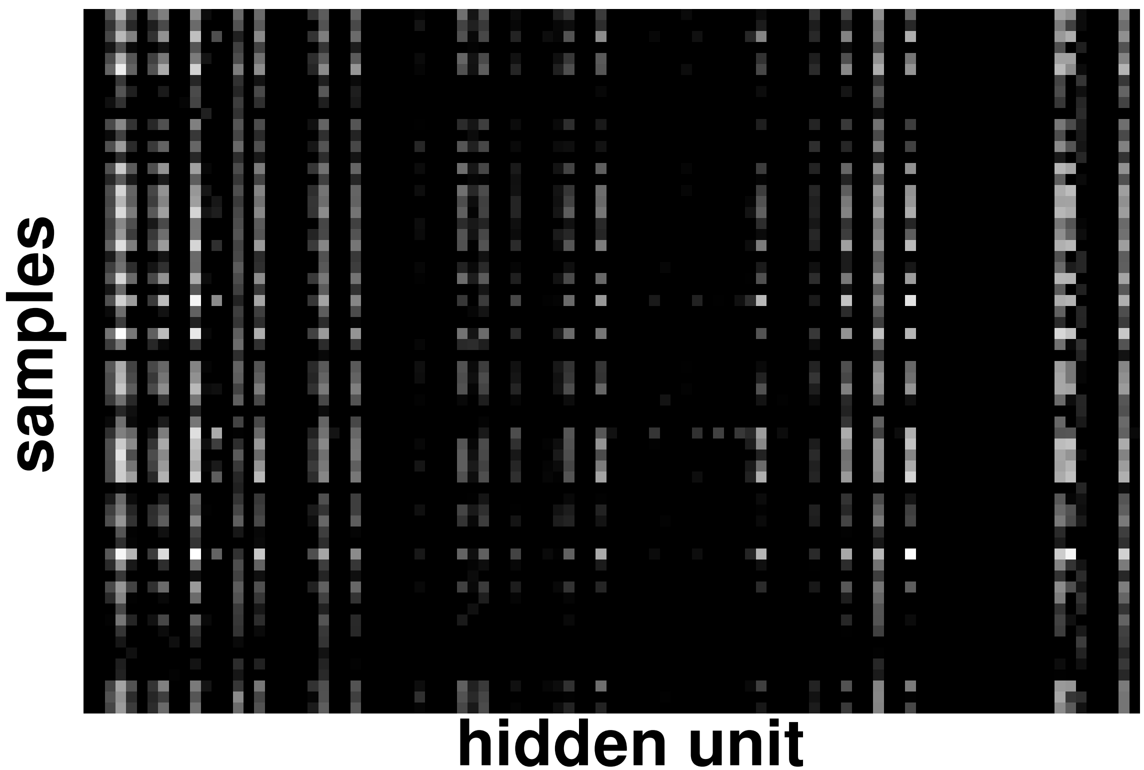


Good training: hidden units are sparse across samples and across features.

Ranzato

Good To Know

- Check gradients numerically by finite differences
- Visualize features (feature maps need to be uncorrelated) and have high variance.

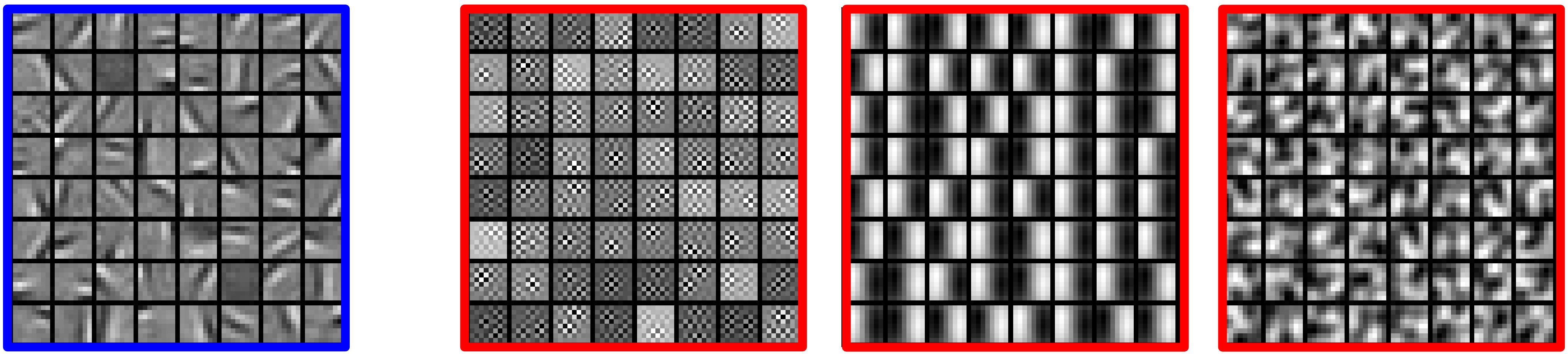


Bad training: many hidden units ignore the input and/or exhibit strong correlations.

Good To Know

- Check gradients numerically by finite differences
- Visualize features (feature maps need to be uncorrelated) and have high variance.
- Visualize parameters

GOOD BAD BAD BAD



too noisy too correlated lack structure

Good training: learned filters exhibit structure and are uncorrelated.

Zeiler, Fergus "Visualizing and understanding CNNs" arXiv 2013
Simonyan, Vedaldi, Zisserman "Deep inside CNNs: visualizing image classification models.." ICLR 2014

Good To Know

- Check gradients numerically by finite differences
- Visualize features (feature maps need to be uncorrelated) and have high variance.
- Visualize parameters
- Measure error on both training and validation set.
- Test on a small subset of the data and check the error $\rightarrow 0$.

What If It Does Not Work?

- Training diverges:
  - Learning rate may be too large $\rightarrow$ decrease learning rate
  - BPROP is buggy $\rightarrow$ numerical gradient checking
- Parameters collapse / loss is minimized but accuracy is low
  - Check loss function:
    - Is it appropriate for the task you want to solve?
    - Does it have degenerate solutions? Check “pull-up” term.
- Network is underperforming
  - Compute flops and nr. params. $\rightarrow$ if too small, make net larger
  - Visualize hidden units/params $\rightarrow$ fix optmization
- Network is too slow
  - Compute flops and nr. params. $\rightarrow$ GPU,distrib. framework, make net smaller

Summary

- Deep Learning = learning hierarhical models. ConvNets are the most successful example. Leverage large labeled datasets.
- Optimization
  - Plain SGD with momentum works well.
- Scaling
  - GPUs
  - Distributed framework (Google)
  - Better optimization techniques
- Generalization on small datasets (curse of dimensionality):
  - data augmentation
  - weight decay
  - dropout
  - unsupervised learning
  - multi-task learning

SOFTWARE

Torch7: learning library that supports neural net training

torch.ch
http://code.cogbits.com/wiki/doku.php (tutorial with demos by C. Farabet)
https://github.com/jhjin/overfeat-torch
https://github.com/facebook/fbcunn/tree/master/examples/imagenet

Python-based learning library (U. Montreal)

- http://deeplearning.net/software/theano/ (does automatic differentiation)

Efficient CUDA kernels for ConvNets (Krizhevsky)

– code.google.com/p/cuda-convnet

Caffe (Yangqing Jia)

– http://caffe.berkeleyvision.org

REFERENCES

Convolutional Nets

– LeCun, Bottou, Bengio and Haffner: Gradient-Based Learning Applied to Document Recognition, Proceedings of the IEEE, 86(11):2278-2324, November 1998

- Krizhevsky, Sutskever, Hinton “ImageNet Classification with deep convolutional neural networks” NIPS 2012

– Jarrett, Kavukcuoglu, Ranzato, LeCun: What is the Best Multi-Stage Architecture for Object Recognition?, Proc. International Conference on Computer Vision (ICCV'09), IEEE, 2009

- Kavukcuoglu, Sermanet, Boureau, Gregor, Mathieu, LeCun: Learning Convolutional Feature Hierachies for Visual Recognition, Advances in Neural Information Processing Systems (NIPS 2010), 23, 2010

– see yann.lecun.com/exdb/publis for references on many different kinds of convnets.

– see http://www.cmap.polytechnique.fr/scattering/ for scattering networks (similar to convnets but with less learning and stronger mathematical foundations)

– see http://www.idsia.ch/~juergen/ for other references to ConvNets and LSTMs.

REFERENCES

Applications of Convolutional Nets

– Farabet, Couprie, Najman, LeCun. Scene Parsing with Multiscale Feature Learning, Purity Trees, and Optimal Covers", ICML 2012

– Pierre Sermanet, Koray Kavukcuoglu, Soumith Chintala and Yann LeCun: Pedestrian Detection with Unsupervised Multi-Stage Feature Learning, CVPR 2013

- D. Ciresan, A. Giusti, L. Gambardella, J. Schmidhuber. Deep Neural Networks Segment Neuronal Membranes in Electron Microscopy Images. NIPS 2012

- Raia Hadsell, Pierre Sermanet, Marco Scoffier, Ayse Erkan, Koray Kavackuoglu, Urs Muller and Yann LeCun. Learning Long-Range Vision for Autonomous Off-Road Driving, Journal of Field Robotics, 26(2):120-144, 2009

– Burger, Schuler, Harmeling. Image Denoisng: Can Plain Neural Networks Compete with BM3D?, CVPR 2012

– Hadsell, Chopra, LeCun. Dimensionality reduction by learning an invariant mapping, CVPR 2006

– Bergstra et al. Making a science of model search: hyperparameter optimization in hundred of dimensions for vision architectures, ICML 2013

REFERENCES

Latest and Greatest Convolutional Nets

– Girshick, Donahue, Darrell, Malick. “Rich feature hierarchies for accurate object detection and semantic segmentation”, arXiv 2014

– Simonyan, Zisserman “Two-stream CNNs for action recognition in videos” arXiv 2014

- Cadieu, Hong, Yamins, Pinto, Ardila, Solomon, Majaj, DiCarlo. “DNN rival in representation of primate IT cortex for core visual object recognition”. arXiv 2014

- Erhan, Szegedy, Toshev, Anguelov “Scalable object detection using DNN” CVPR 2014

- Razavian, Azizpour, Sullivan, Carlsson “CNN features off-the-shelf: and astounding baseline for recognition” arXiv 2014

- Krizhevsky “One weird trick for parallelizing CNNs” arXiv 2014

REFERENCES

Deep Learning in general

– deep learning tutorial @ CVPR 2014 https://sites.google.com/site/deeplearningcvpr2014/

– deep learning tutorial slides at ICML 2013: icml.cc/2013/?page_id=39

– Yoshua Bengio, Learning Deep Architectures for AI, Foundations and Trends in Machine Learning, 2(1), pp.1-127, 2009.

– LeCun, Chopra, Hadsell, Ranzato, Huang: A Tutorial on Energy-Based Learning, in Bakir, G. and Hofman, T. and Schölkopf, B. and Smola, A. and Taskar, B. (Eds), Predicting Structured Data, MIT Press, 2006

"Theory" of Deep Learning

– Mallat: Group Invariant Scattering, Comm. In Pure and Applied Math. 2012

– Pascanu, Montufar, Bengio: On the number of inference regions of DNNs with piece wise linear activations, ICLR 2014

– Pascanu, Dauphin, Ganguli, Bengio: On the saddle-point problem for non-convex optimization, arXiv 2014

- Delalleau, Bengio: Shallow vs deep Sum-Product Networks, NIPS 2011

THANK YOU