Deep Learning 1
Neural Net Basics

Many slides by James Heys,
Marc'Aurelio Ranzato

1

Outline

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

2

Supervised Learning

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

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.

Ranzato

3

Supervised Learning: Examples

Classification

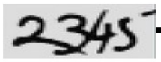
"dog"

classification

Denoising

regression

OCR



"2 3 4 5"

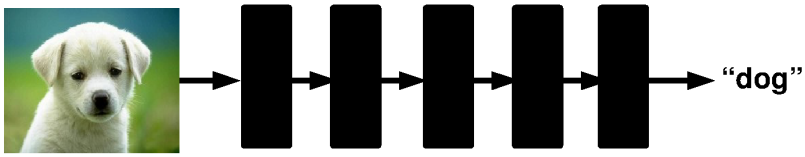
structured prediction

Ranzato

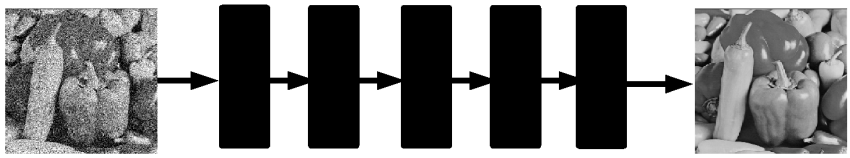
4

Supervised Deep Learning

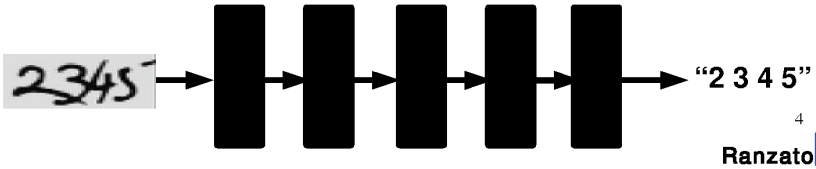
Classification



Denoising



OCR



4

Ranzato

5

Outline

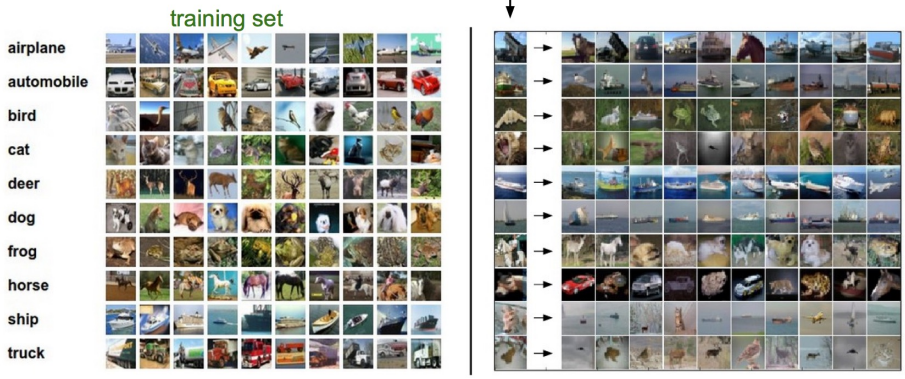
- Supervised Neural Networks
- Convolutional Neural Networks
- Examples
- Tips

5

Ranzato

6

k-Nearest Neighbor

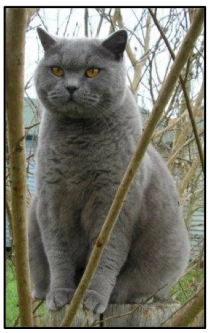


Slide credit:Fei-Fei Li

7

Linear Classification

1. define a **score function**



class scores

Slide credit:Fei-Fei Li

8

Linear Classification

1. define a **score function**

data (image)

$$f(x_i, W, b) = Wx_i + b$$

class scores

"weights"

"bias vector"

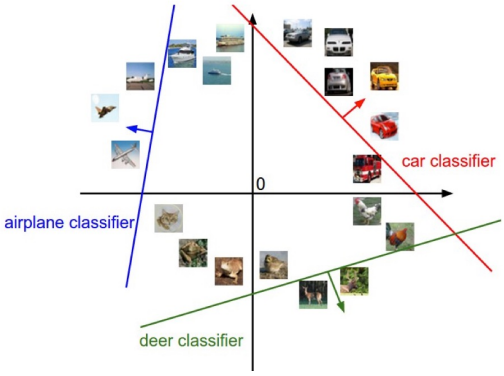
"parameters"

Slide credit:Fei-Fei Li

9

Interpreting a Linear Classifier

$$f(x_i, W, b) = Wx_i + b$$



Slide credit:Fei-Fei Li

10

Loss

2. Define a **loss function** (or cost function, or objective)

- scores, label $f(x_i, W)$ y_i ⟶ *loss.* L_i

Example:

$f(x_i, W) = [13, -7, 11]$

$y_i = 0$

Question: if you were to assign a single number to how "unhappy" you are with these scores, what would you do?

Slide credit:Fei-Fei Li

11

Multiclass SVM Loss

2. Define a **loss function** (or cost function, or objective)
One (of many ways) to do it: **Multiclass SVM Loss**

$$L_i = \sum_{j \neq y_i} \max(0, f(x_i, W)_j - f(x_i, W)_{y_i} + \Delta)$$

loss due to example i

sum over all incorrect labels

difference between the correct class score and incorrect class score

Slide credit:Fei-Fei Li

12

$$L_i = \sum_{j \neq y_i} \max(0, f(x_i, W)_j - f(x_i, W)_{y_i} + \Delta)$$

loss due to example i

sum over all incorrect labels

difference between the correct class score and incorrect class score

Slide credit:Fei-Fei Li

13

L2 Regularization

$$R(W) = \sum_k \sum_l W_{k,l}^2$$

$$L = \frac{1}{N} \sum_i \sum_{j \neq y_i} \left[\max(0, f(x_i; W)_j - f(x_i; W)_{y_i} + \Delta) \right] + \lambda R(W)$$

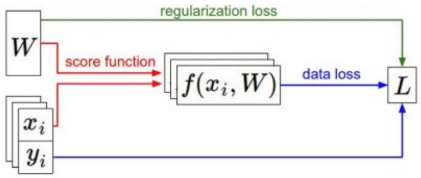
Regularization strength

Slide credit:Fei-Fei Li

14

Putting it all together:

Linear Classification



SVM:

$$L = \frac{1}{N}\sum_i \sum_{j \neq y_i} \left[\max(0, f(x_i; W)_j - f(x_i; W)_{y_i} + \Delta)\right] + \lambda \sum_k \sum_l W_{k,l}^2$$

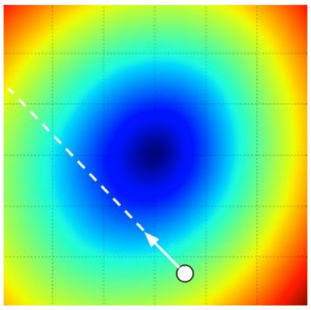
Softmax:

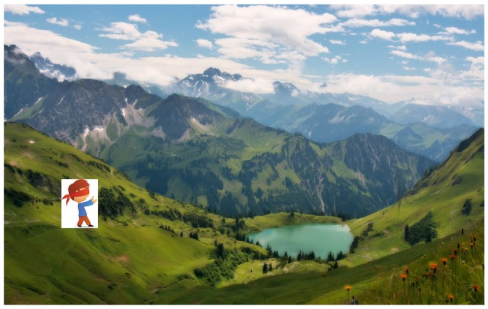
$$L = \frac{1}{N}\sum_i -\log\left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}\right) + \lambda \sum_k \sum_l W_{k,l}^2$$

Slide credit:Fei-Fei Li

15

Optimization Landscape





Slide credit:Fei-Fei Li

16

Gradient Descent

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

Numerical gradient: slow :(, approximate :(, easy to write :)
Analytic gradient: fast :), exact :), error-prone :(

In practice: Derive analytic gradient, check your implementation with numerical gradient

Slide credit:Fei-Fei Li

17

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?

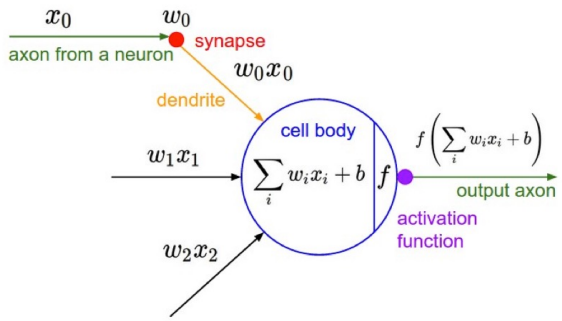
Answer: later...

6

Ranzato

18

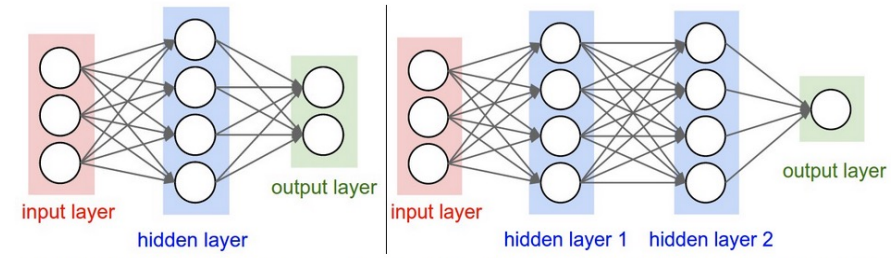
A Single Neuron
Activation Functions



Slide credit:Fei-Fei Li

19

Neural Network Structure



Left: A 2-layer Neural Network (one hidden layer of 4 neurons (or units) and one output layer with 2 neurons), and three inputs. Right: A 3-layer neural network with three inputs, two hidden layers of 4 neurons each and one output layer. Notice that in both cases there are connections (synapses) between neurons across layers, but not within a layer.

20

Activation functions

Step

$$g(a) = \begin{cases} 0 & a < 0 \\ 1 & a \geq 0 \end{cases}$$

Sigmoidal

$$g(a) = \frac{1}{1 + \exp(-a)}$$

Hyperbolic tangent

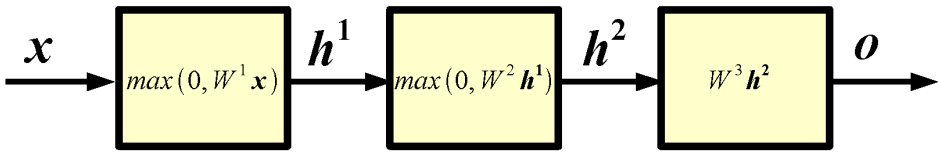
$$g(a) = \frac{\exp(a) - \exp(-a)}{\exp(a) + \exp(-a)}$$

Rectified Linear Unit (RELU)

$$g(a) = \max(0, a)$$

21

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).

7

Ranzato

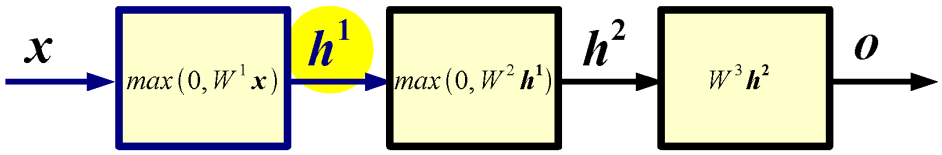
22

Forward Propagation

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

23

Forward Propagation



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

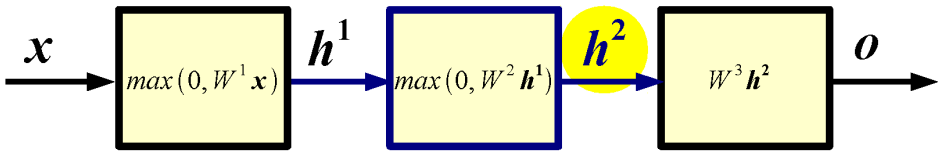
$$h^1 = max(0, W^1 x + b^1)$$

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**".

24

Forward Propagation



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

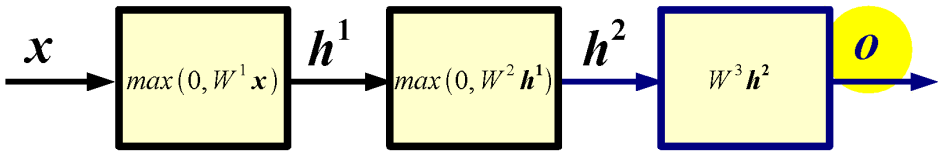
$$\boldsymbol{h^2} = max(0, W^2 \boldsymbol{h^1} + \boldsymbol{b^2})$$

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

10
Ranzato

25

Forward Propagation



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

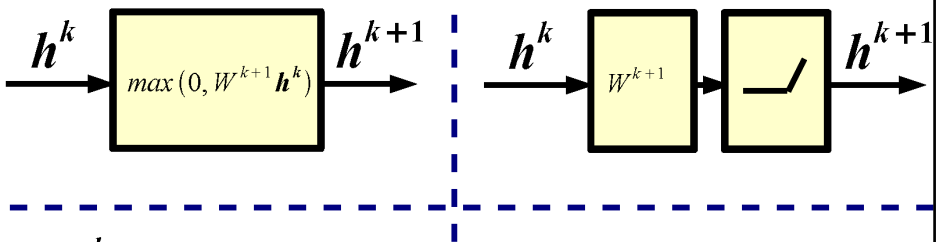
$$\boldsymbol{o} = max(0, W^3 \boldsymbol{h^2} + \boldsymbol{b^3})$$

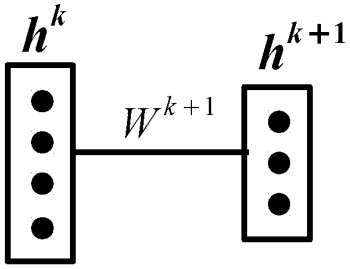
W^3 3-rd layer weight matrix or weights
$\boldsymbol{b^3}$ 3-rd layer biases

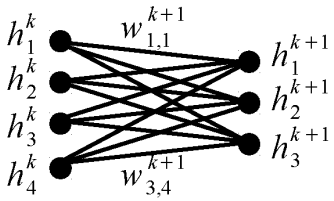
11
Ranzato

26

Alternative Graphical Representation







12

Ranzato

27

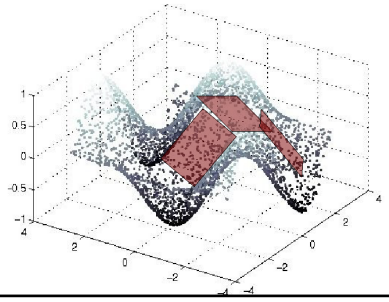
Interpretation

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.



13

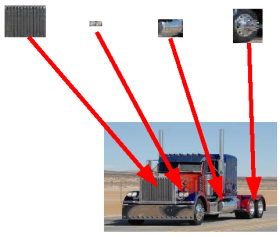
28

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)

14

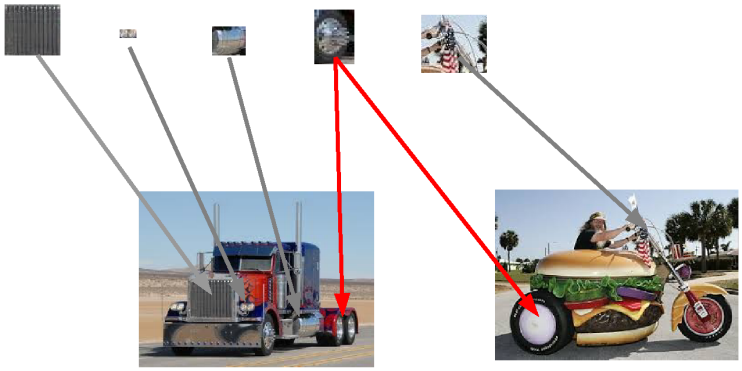
Ranzato

29

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

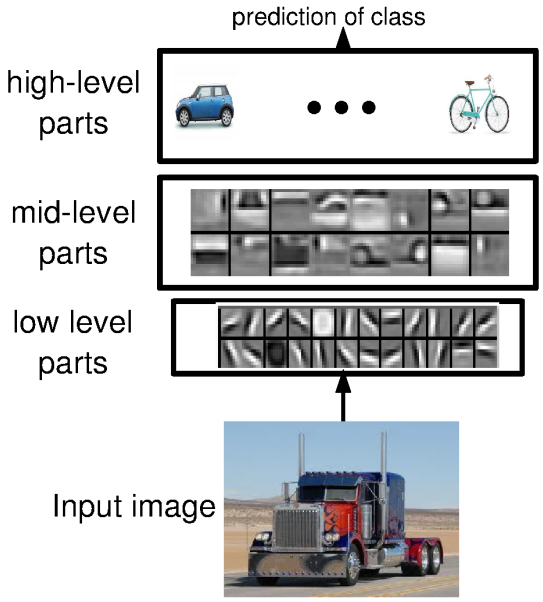


15

Ranzato

30

Interpretation



- distributed representations
- feature sharing
- compositionality

Lee et al. "Convolutional DBN's ..." ICML 2009

16

Ranzato

31

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?

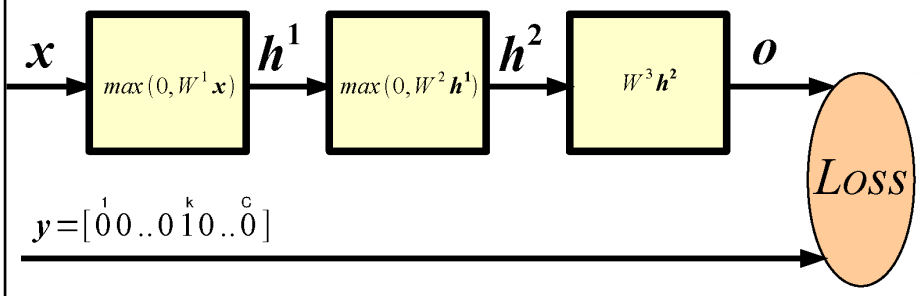
Answer: Weight matrices and biases are learned.
First, we need to define a measure of quality of the current mapping.
Then, we need to define a procedure to adjust the parameters.

17

Ranzato

32

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):

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

18 Ranzato

33

Neural network properties

- **Theorem (Universal function approximators):** A two-layer network with a sufficient number of neurons can approximate any continuous function to any desired accuracy
- **Practical considerations:**
  - Can be seen as learning the features
  - Large number of neurons
    - Danger for overfitting
  - Hill-climbing procedure can get stuck in bad local optima

Approximation by Superpositions of Sigmoidal Function, 1989

Slide credit: Pieter Abeel and Dan Klein

34

Training

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

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

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

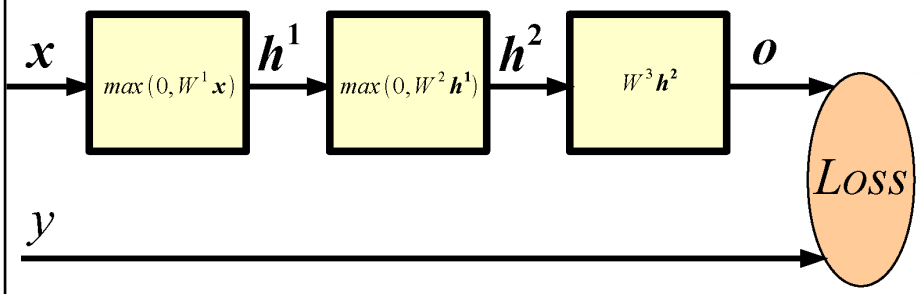
Answer: Chain rule, a.k.a. **Backpropagation**! That is the procedure to compute gradients of the loss w.r.t. parameters in a multi-layer neural network.

19

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

35

Key Idea: Wiggle To Decrease Loss



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

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

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

Then, update:

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

20

Ranzato

36

Derivative w.r.t. Input of Softmax

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

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

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

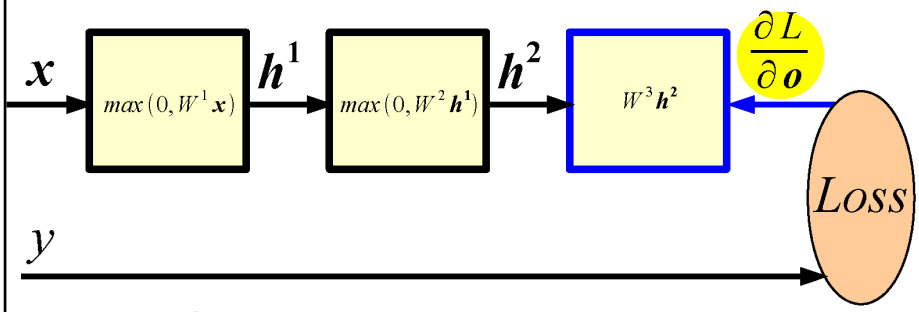
$$\frac{\partial L}{\partial o} = p(c|\boldsymbol{x}) - \boldsymbol{y}$$

21

Ranzato

37

Backward Propagation



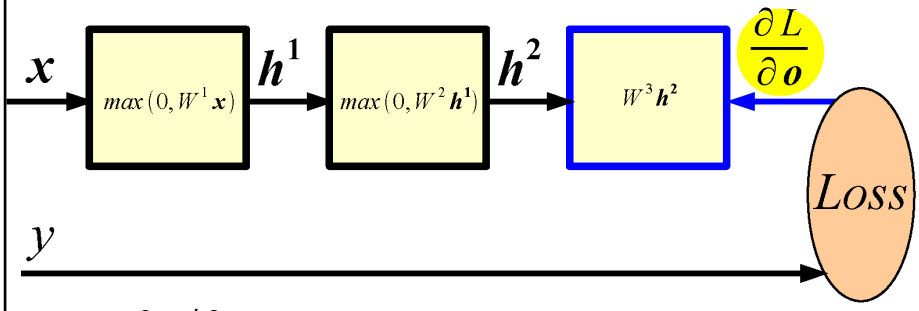
Given $\partial L/\partial \boldsymbol{o}$ and assuming we can easily compute the Jacobian of each module, we have:

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

22

38

Backward Propagation



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

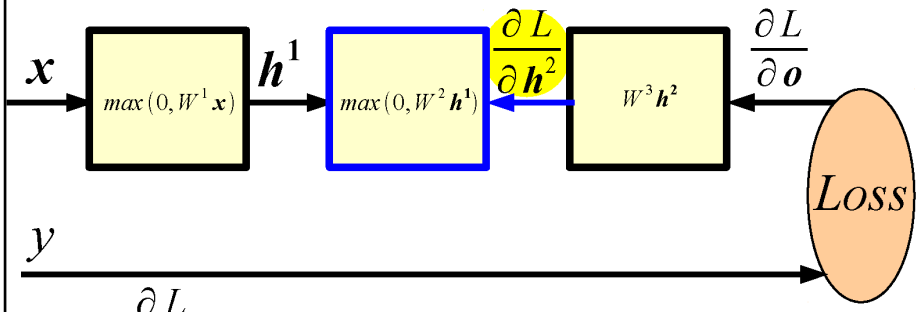
$$\frac{\partial L}{\partial W^3} = \frac{\partial L}{\partial \boldsymbol{o}} \frac{\partial \boldsymbol{o}}{\partial W^3} \qquad \frac{\partial L}{\partial \boldsymbol{h}^2} = \frac{\partial L}{\partial \boldsymbol{o}} \frac{\partial \boldsymbol{o}}{\partial \boldsymbol{h}^2}$$

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

23

39

Backward Propagation



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

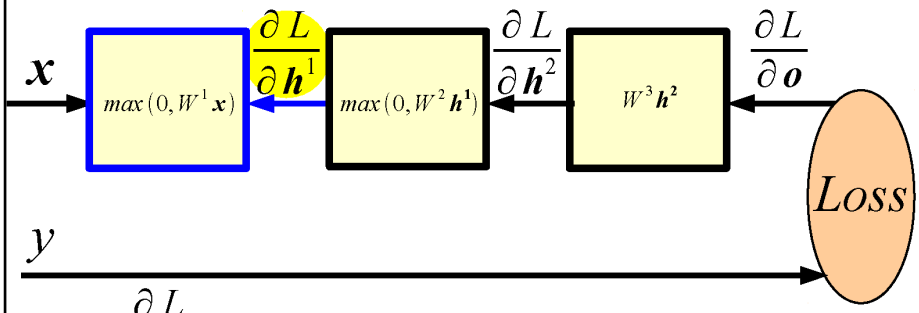
$$\frac{\partial L}{\partial W^2} = \frac{\partial L}{\partial \boldsymbol{h}^2} \frac{\partial \boldsymbol{h}^2}{\partial W^2} \qquad \frac{\partial L}{\partial \boldsymbol{h}^1} = \frac{\partial L}{\partial \boldsymbol{h}^2} \frac{\partial \boldsymbol{h}^2}{\partial \boldsymbol{h}^1}$$

24

Ranzato

40

Backward Propagation



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

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

25

Ranzato

41

Backward Propagation

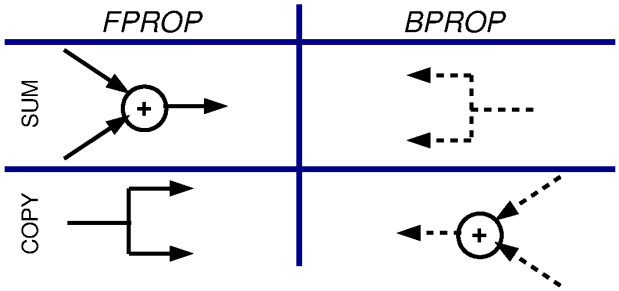
Question: Does BPROP work with ReLU layers only?

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.,:



26

Ranzato

42

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:

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

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

Note: there are many other variants...

27

Ranzato

43

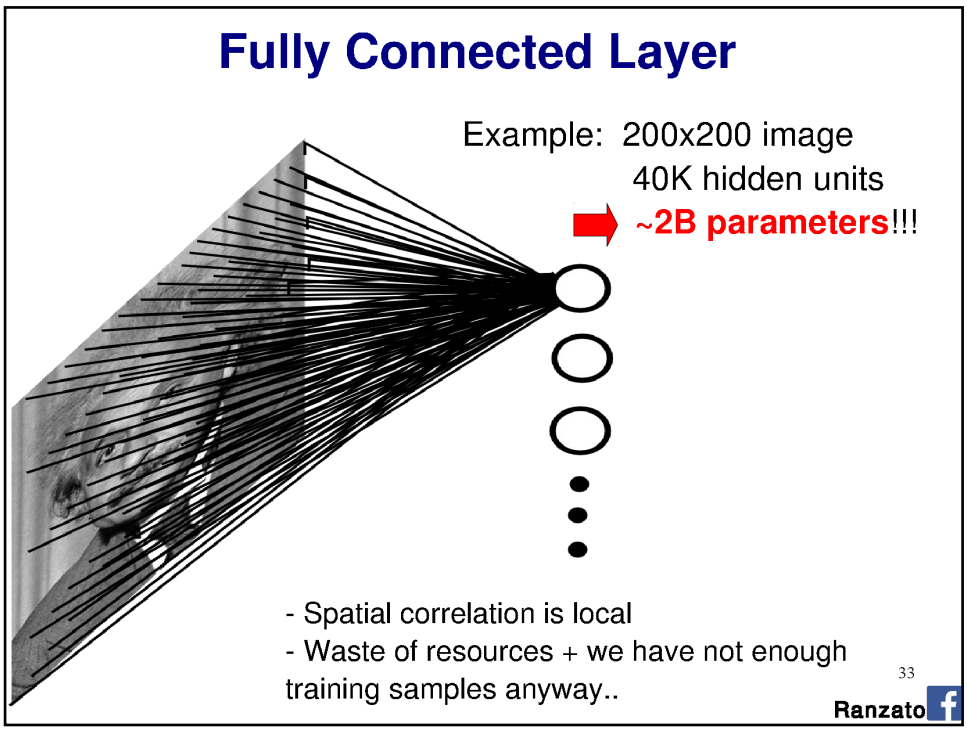
Outline

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

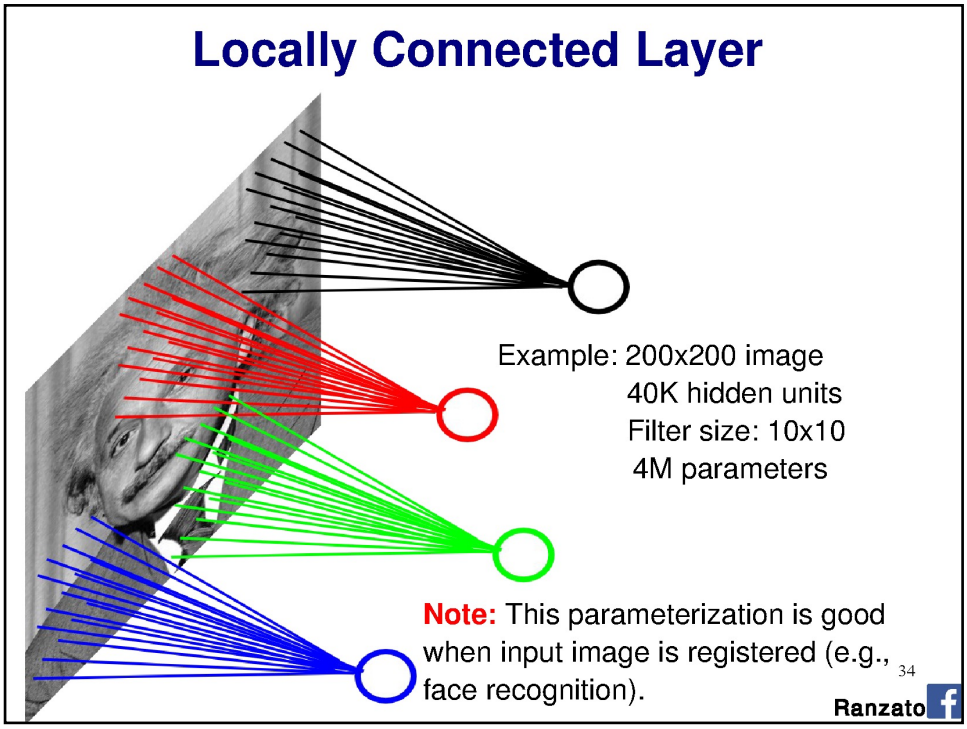
32

Ranzato

44

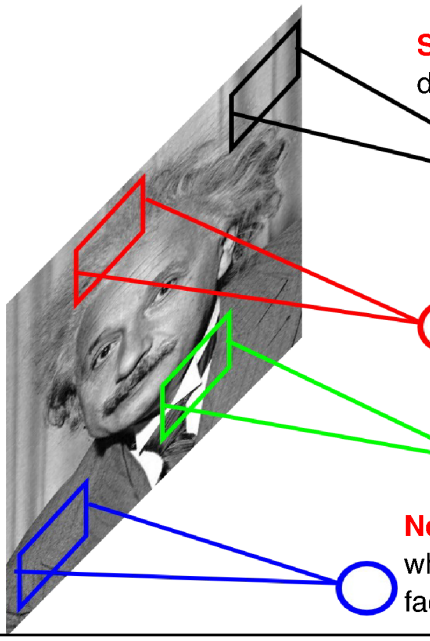

Fully Connected Layer
Example: 200x200 image
40K hidden units
~2B parameters!!!
- Spatial correlation is local
- Waste of resources + we have not enough training samples anyway..
33
Ranzato

45


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).
34
Ranzato

46

Locally Connected Layer



STATIONARITY? Statistics is similar at different locations

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

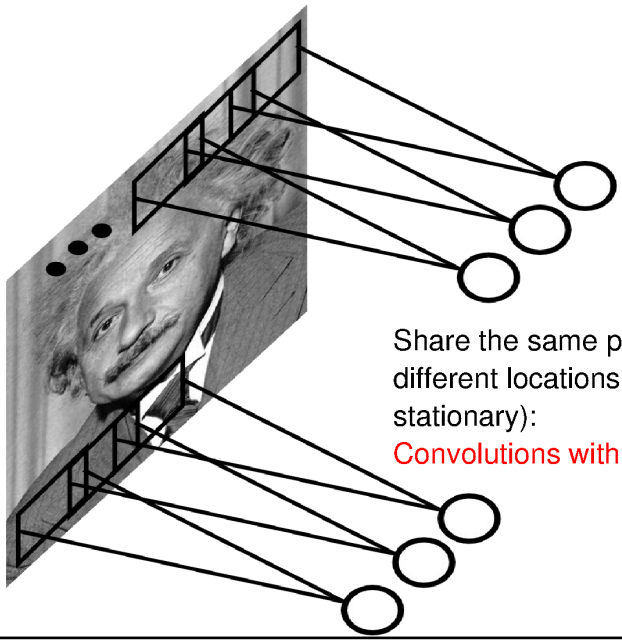
Note: This parameterization is good when input image is registered (e.g., face recognition).

35

Ranzato

47

Convolutional Layer



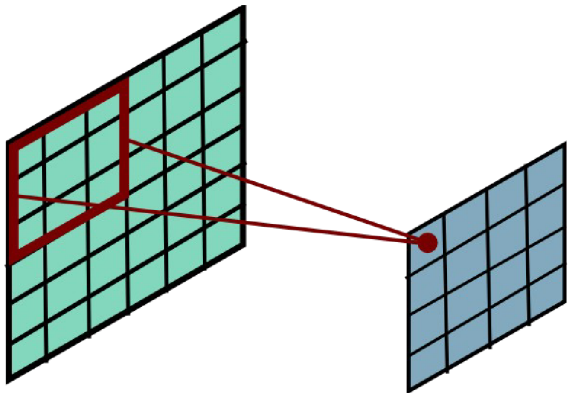
Share the same parameters across different locations (assuming input is stationary):
Convolutions with learned kernels

36

Ranzato

48

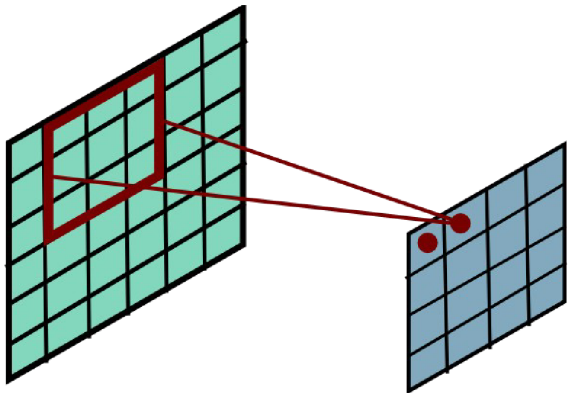
Convolutional Layer



Ranzato

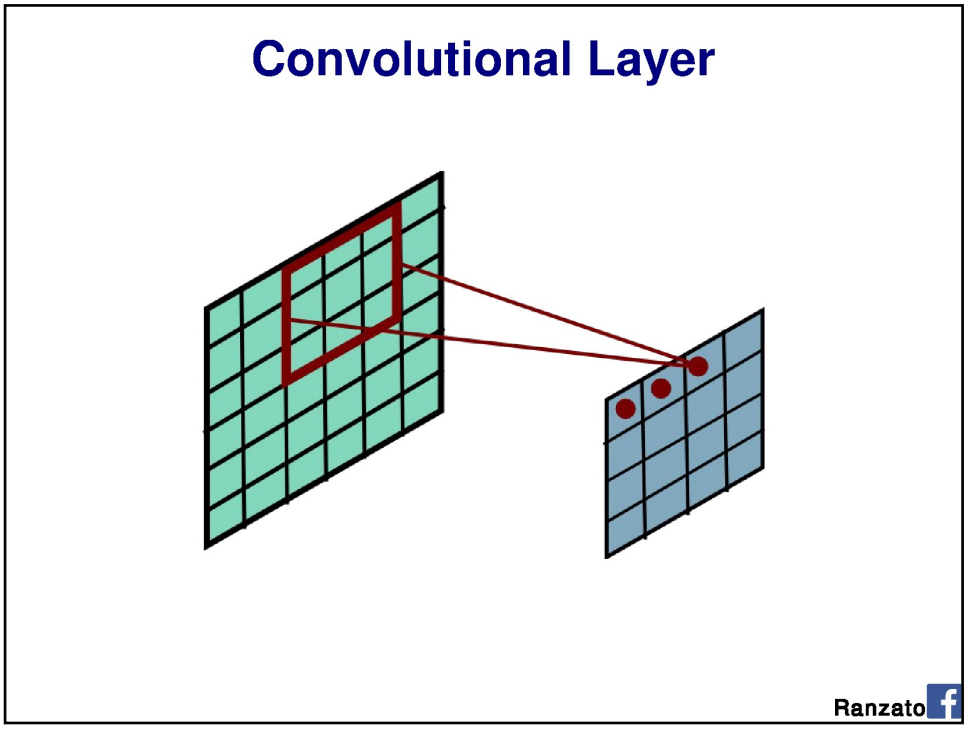
49

Convolutional Layer

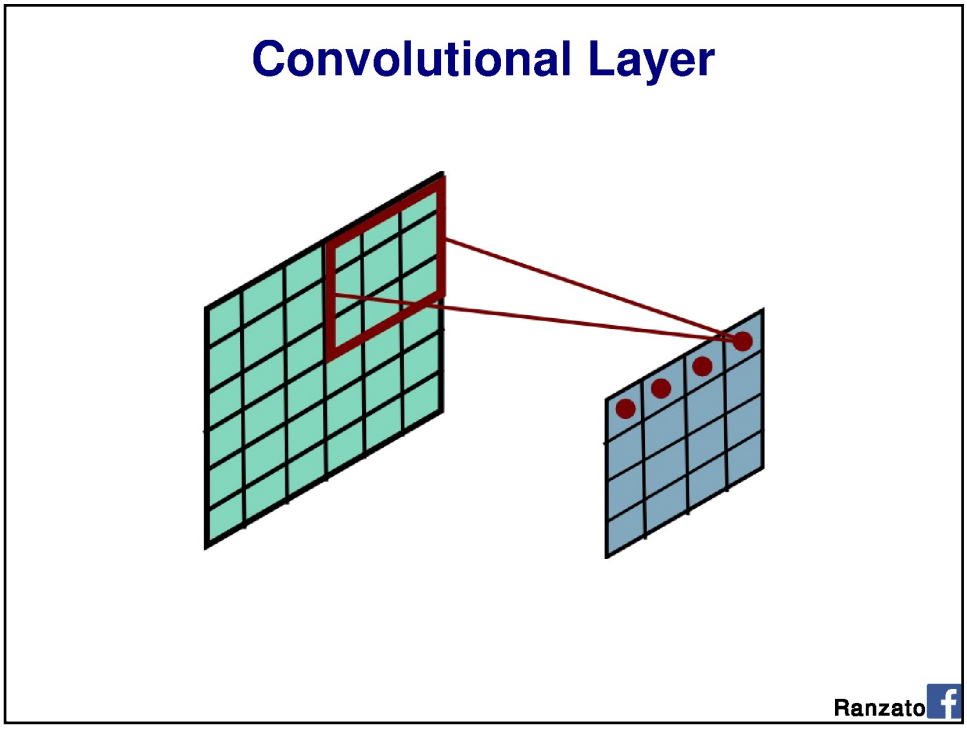


Ranzato

50

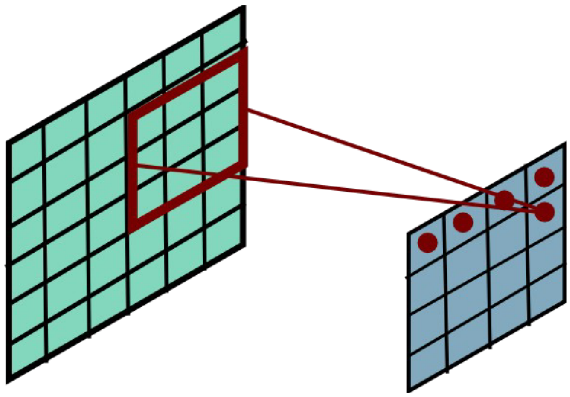

Convolutional Layer
Ranzato

51


Convolutional Layer
Ranzato

52

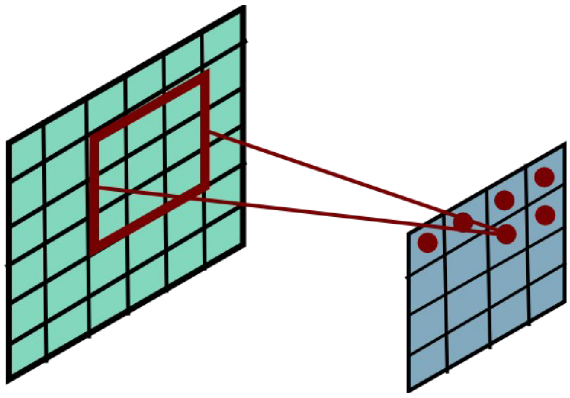
Convolutional Layer



Ranzato

53

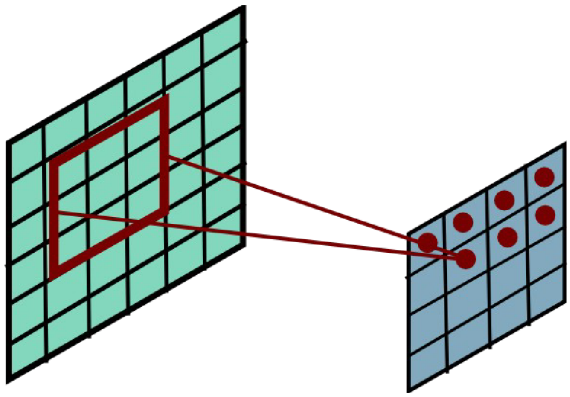
Convolutional Layer



Ranzato

54

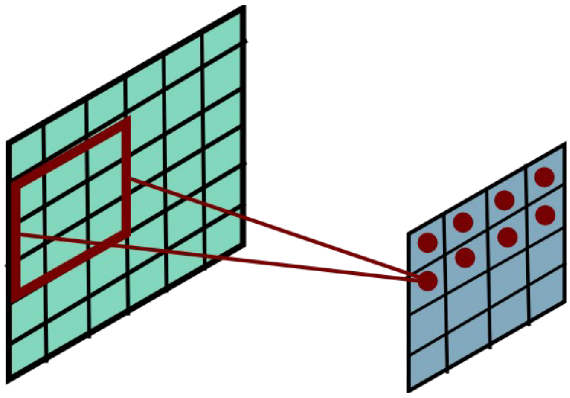
Convolutional Layer



Ranzato

55

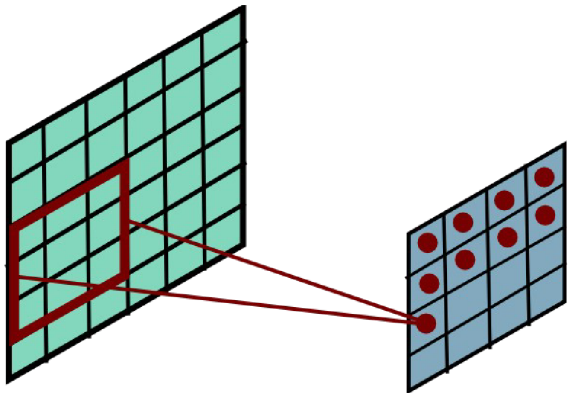
Convolutional Layer



Ranzato

56

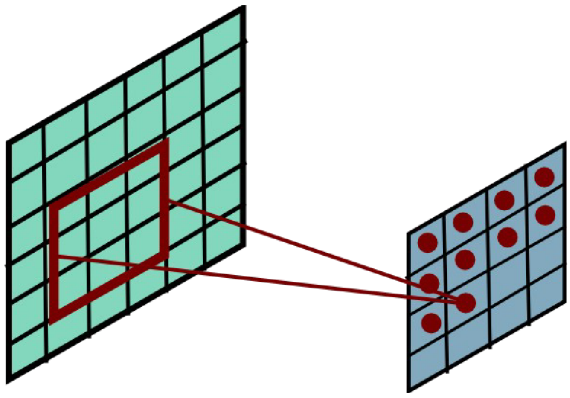
Convolutional Layer



Ranzato

57

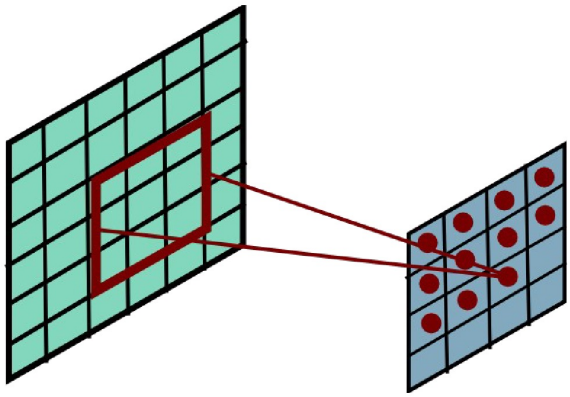
Convolutional Layer



Ranzato

58

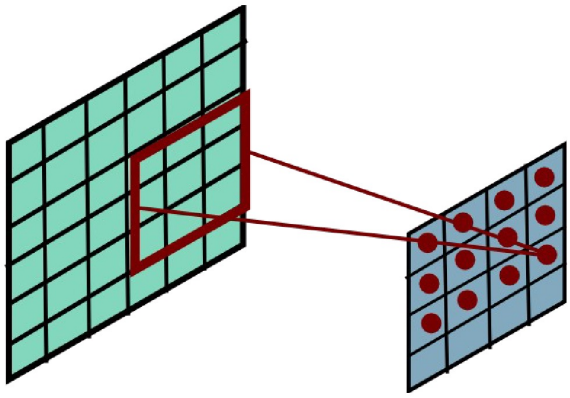
Convolutional Layer



Ranzato

59

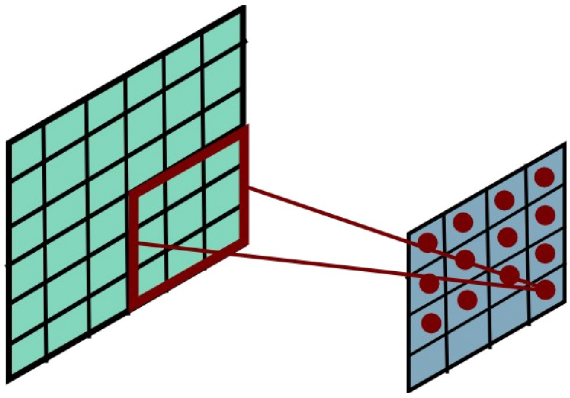
Convolutional Layer



Ranzato

60

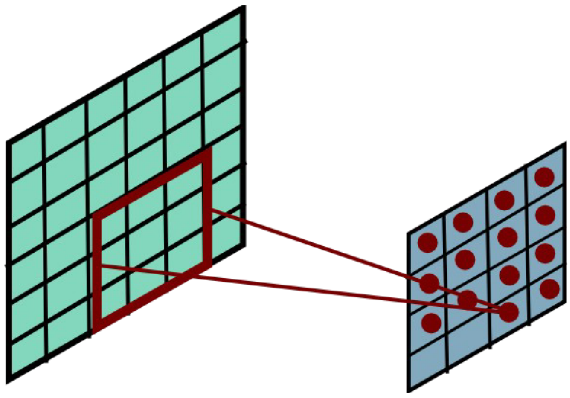
Convolutional Layer



Ranzato

61

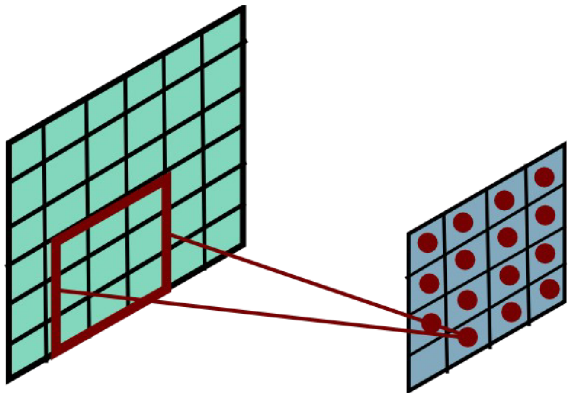
Convolutional Layer



Ranzato

62

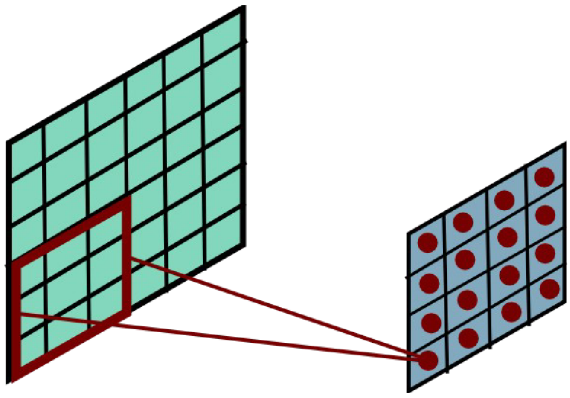
Convolutional Layer



Ranzato

63

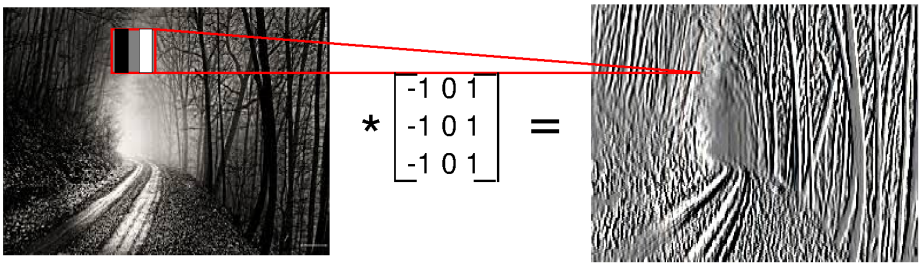
Convolutional Layer



Ranzato

64

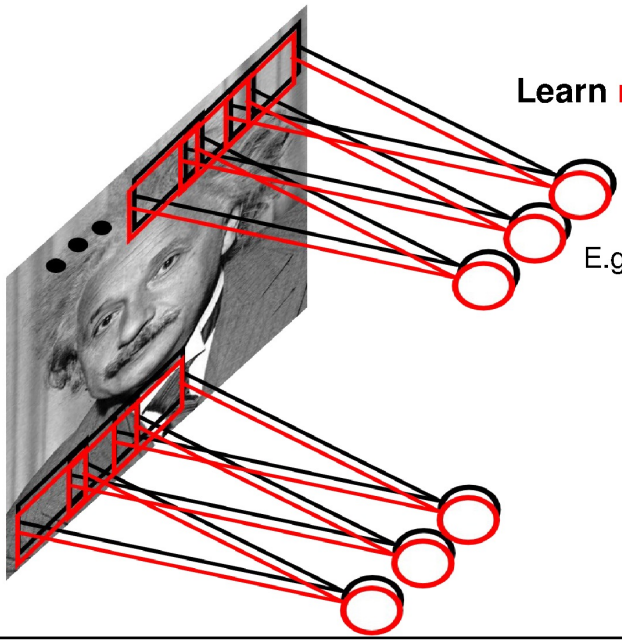
Convolutional Layer



Ranzato

65

Convolutional Layer



Learn multiple filters.

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

54

Ranzato

66

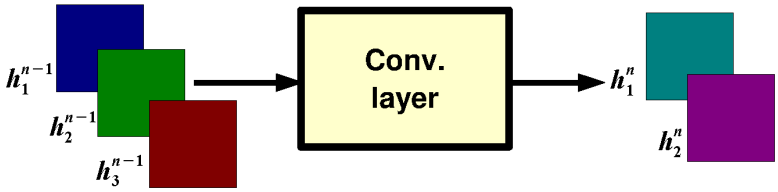
Convolutional Layer

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

output feature map

input feature map

kernel



55

Ranzato

67

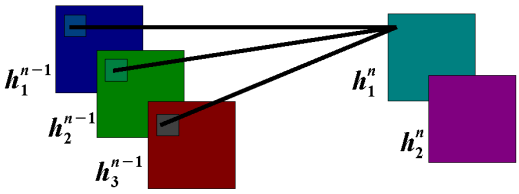
Convolutional Layer

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

output feature map

input feature map

kernel



56

Ranzato

68

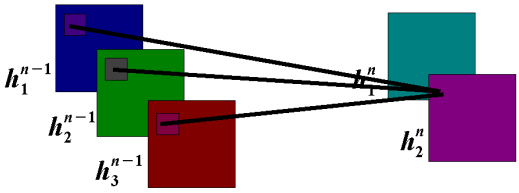
Convolutional Layer

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

output feature map

input feature map

kernel



57

Ranzato

69

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).

58

Ranzato

70

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

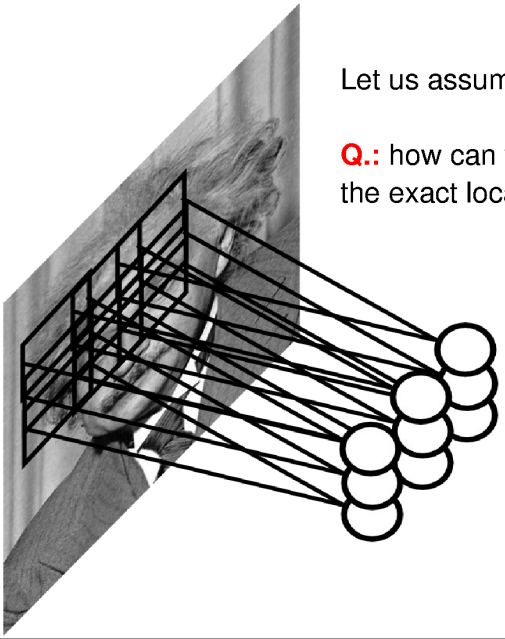
This is called: **convolutional layer.**
A network with convolutional layers is called **convolutional network.**

59

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

71

Pooling Layer



Let us assume filter is an "eye" detector.

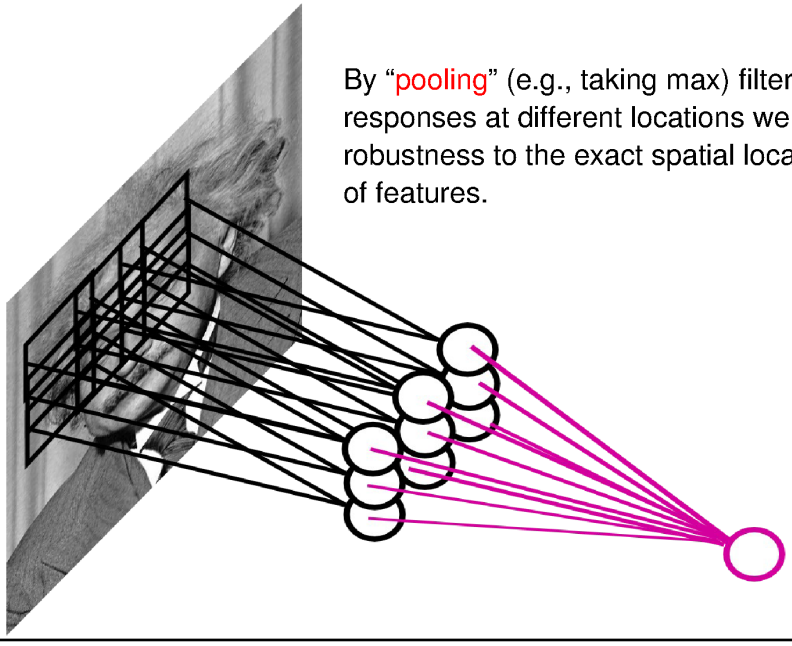
Q.: how can we make the detection robust to the exact location of the eye?

60

Ranzato

72

Pooling Layer



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

61

Ranzato

73

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:

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

62

Ranzato

74

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?

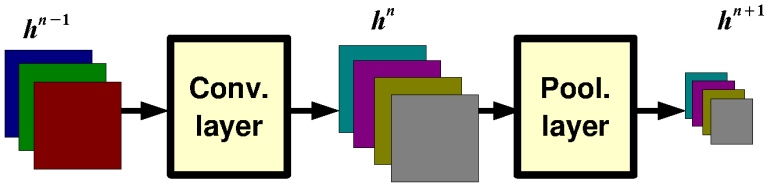
Answer: It depends on how much "invariant" or robust to distortions we want the representation to be. It is best to pool slowly (via a few stacks of conv-pooling layers).

63

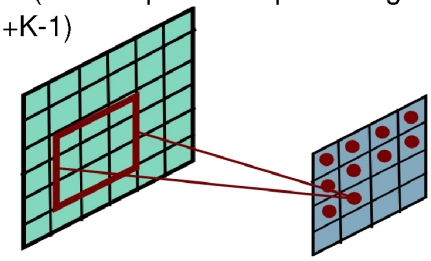
Ranzato

75

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)

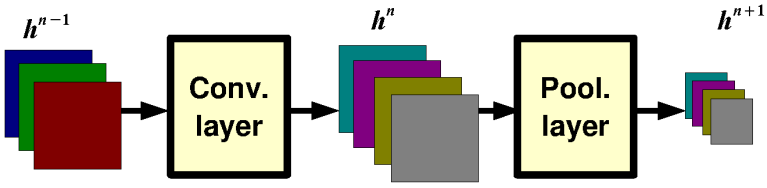


66

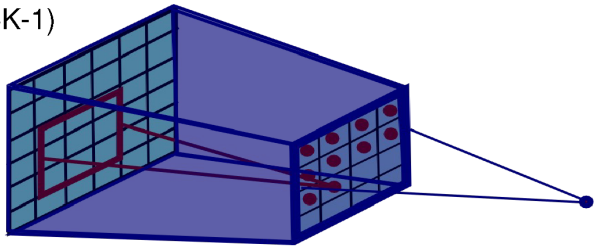
Ranzato

76

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)



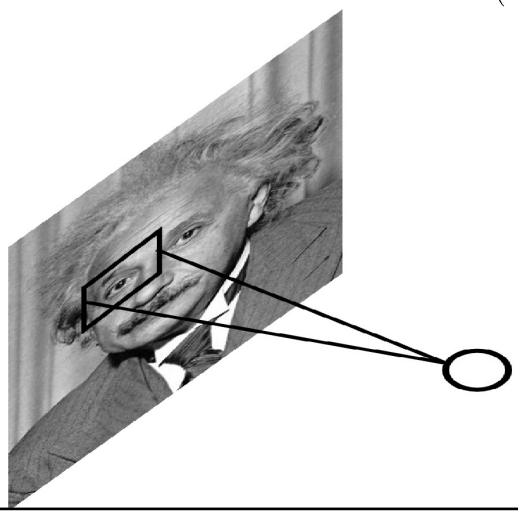
67

Ranzato

77

Local Contrast Normalization

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



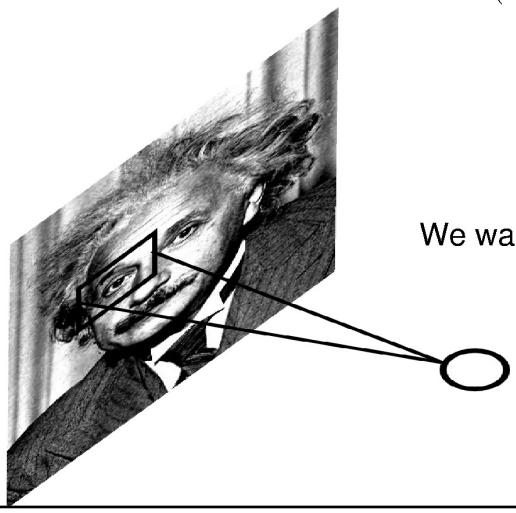
68

Ranzato

78

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.

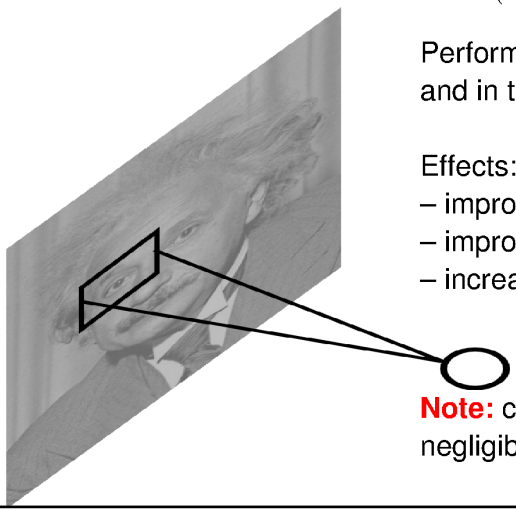
69

Ranzato

79

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.

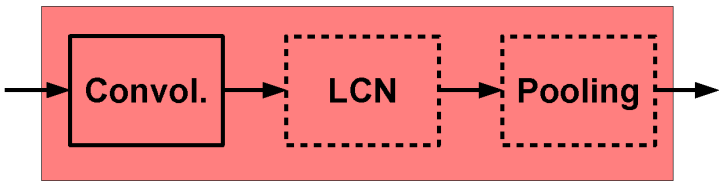
70

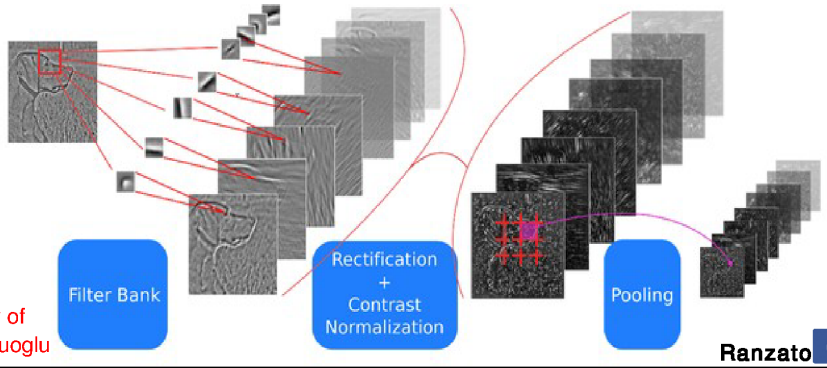
Ranzato

80

ConvNets: Typical Stage

One stage (zoom)





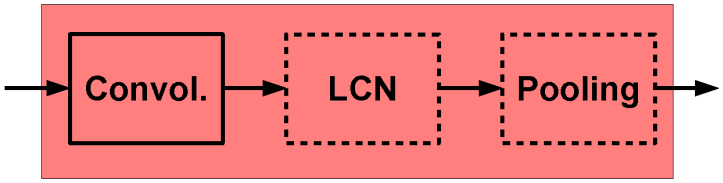
courtesy of
K. Kavukcuoglu

Ranzato

81

ConvNets: Typical Stage

One stage (zoom)

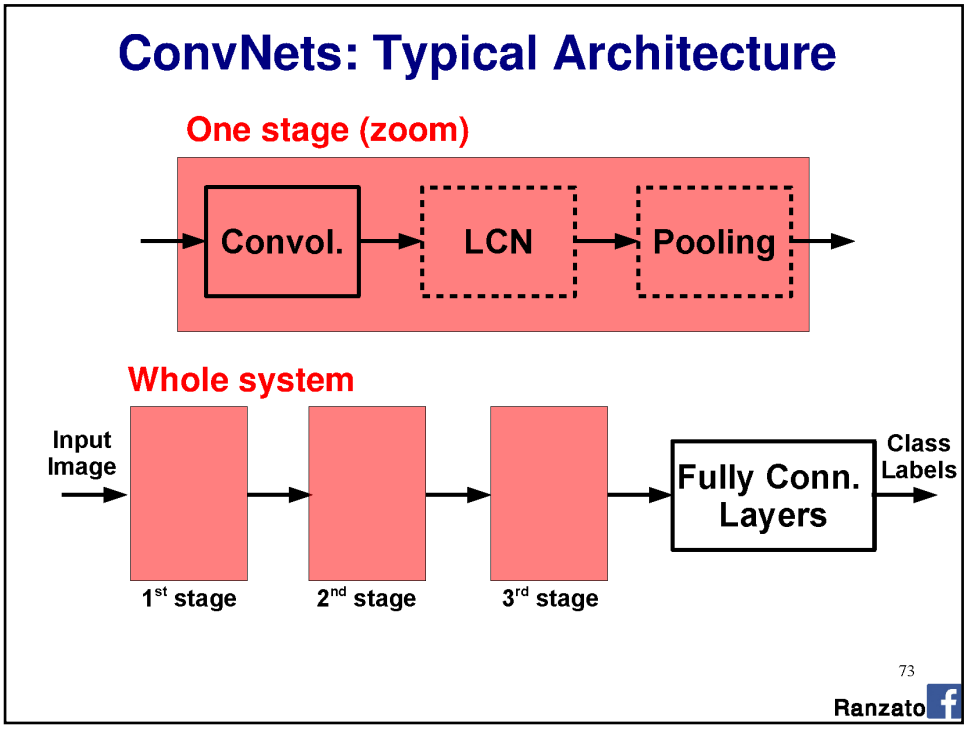


Conceptually similar to: SIFT, HoG, etc.

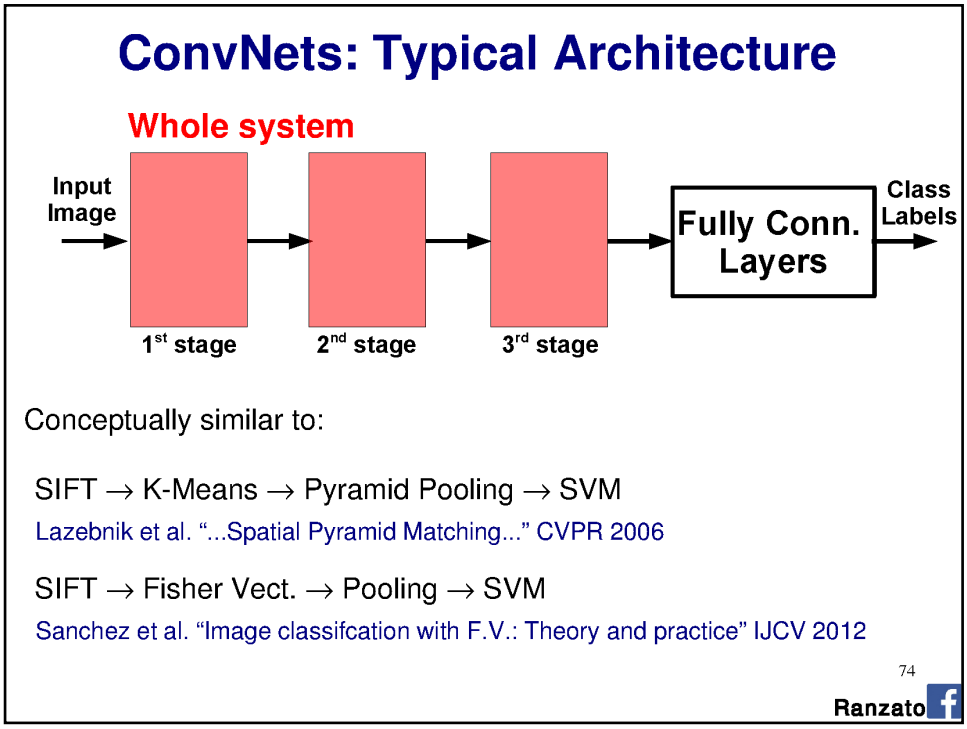
72

Ranzato

82


ConvNets: Typical Architecture
One stage (zoom)
Convol.
LCN
Pooling
Whole system
Input Image
1st stage
2nd stage
3rd stage
Fully Conn. Layers
Class Labels
73
Ranzato

83


ConvNets: Typical Architecture
Whole system
Input Image
1st stage
2nd stage
3rd stage
Fully Conn. Layers
Class Labels
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
74
Ranzato

84

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

85

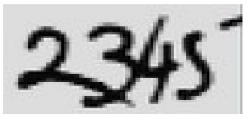
Outline

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

86

CONV NETS: EXAMPLES

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



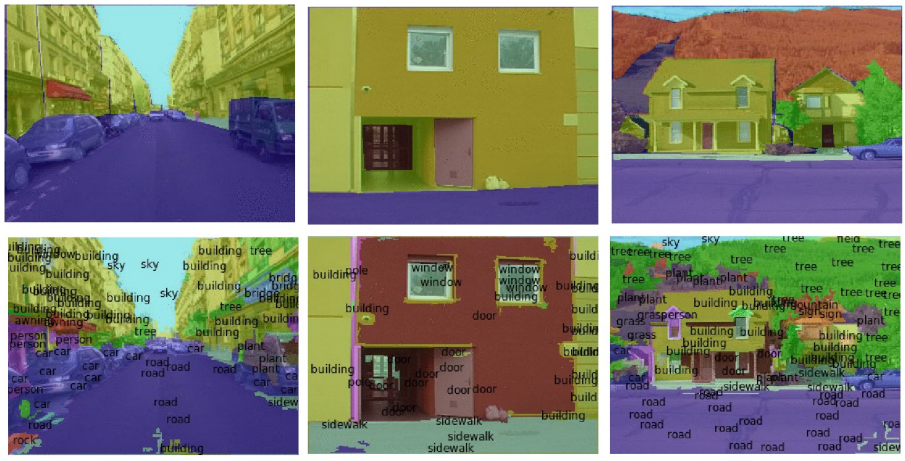
Ciresan et al. "MCDNN for image classification" CVPR 2012
Wan et al. "Regularization of neural networks using dropconnect" ICML 2013
Jaderberg et al. "Synthetic data and ANN for natural scene text recognition" arXiv 2014

82

87

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

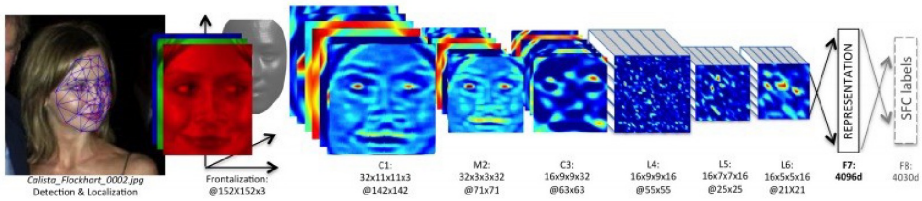
85

Ranzato

88

CONV NETS: EXAMPLES

- **Face Verification & Identification**



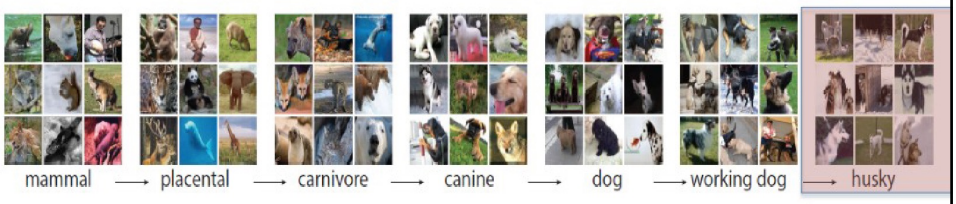
Taigman et al. "DeepFace..." CVPR 2014

92

Ranzato

89

Dataset: ImageNet 2012



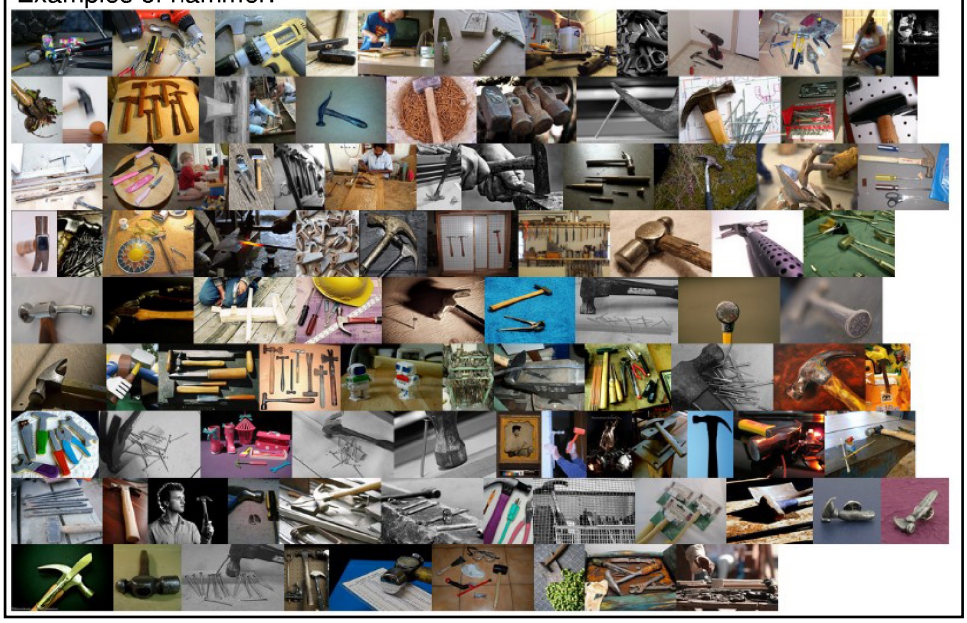
- S: (n) Eskimo dog, husky (breed of heavy-coated Arctic sled dog)
  - direct hypernym / inherited hypernym / sister term
    - S: (n) working dog (any of several breeds of usually large powerful dogs bred to work as draft animals and guard and guide dogs)
      - S: (n) dog, domestic dog, Canis familiaris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

90

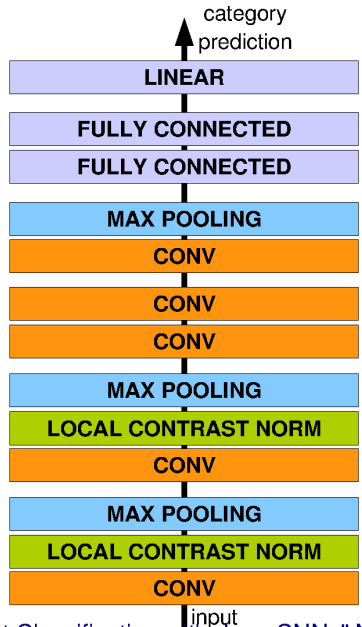
ImageNet

Examples of hammer:



91

Architecture for Classification



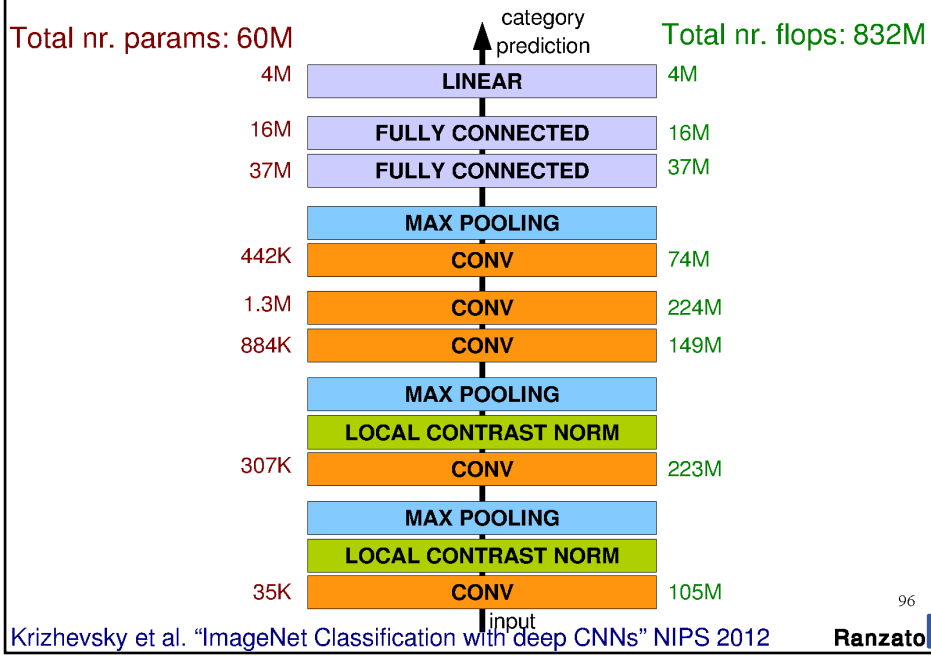
95

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

Ranzato

92

Architecture for Classification



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

96

Ranzato

93

Optimization

SGD with momentum:

- Learning rate = 0.01
- Momentum = 0.9

Improving generalization by:

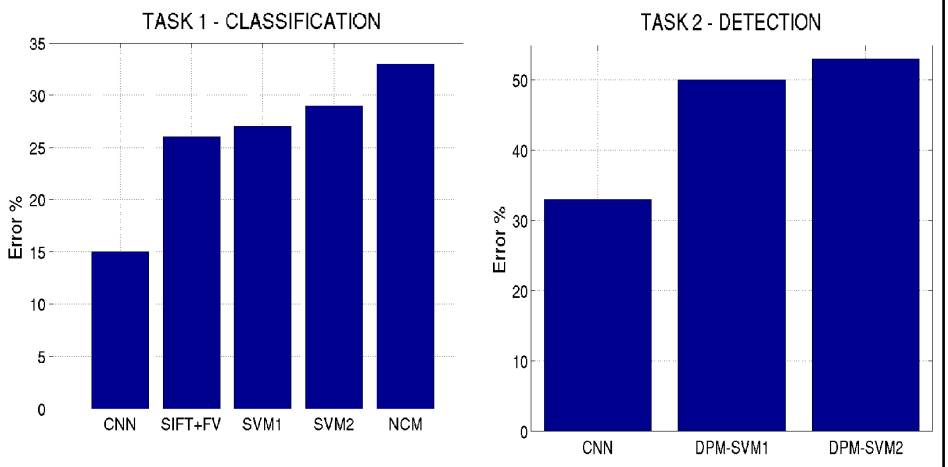
- Weight sharing (convolution)
- Input distortions
- Dropout = 0.5
- Weight decay = 0.0005

97

Ranzato

94

Results: ILSVRC 2012

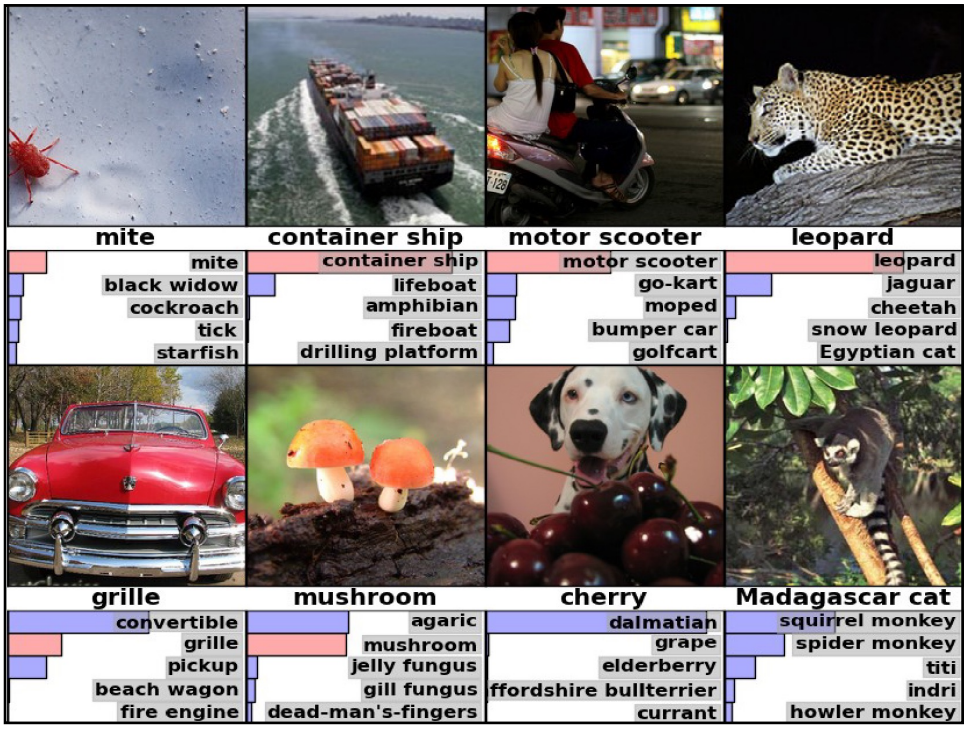


98

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

Ranzato

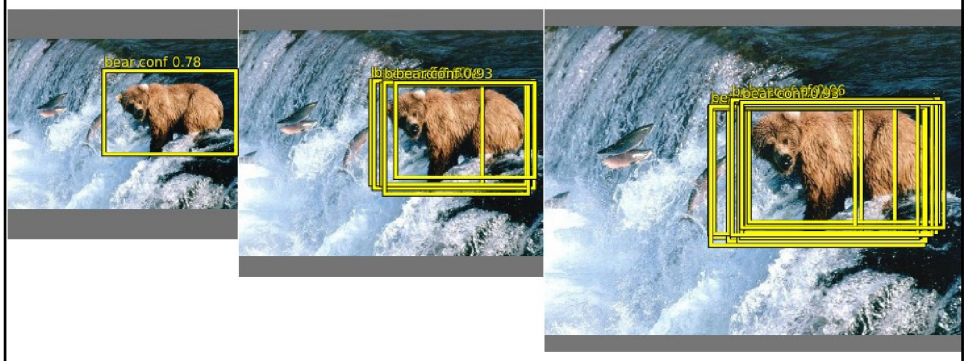
95



96

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

91

Ranzato

97