

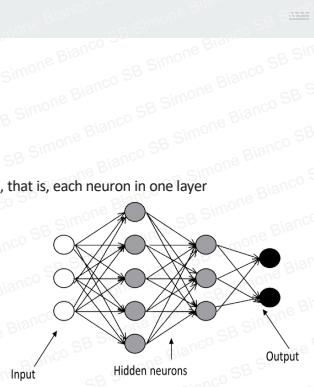
# Machine Learning for Modelling: *Supervised Learning*

Simone Bianco

1

## Multi-layer perceptron

- Neurons are divided in layers
    - of three different kinds: input, hidden, output
    - Each layer is connected to the next one (without cycles)
    - Multilayer perceptrons usually mean fully connected: each neuron is connected to all neurons in the next layer.
    - Input data is used to set the input layer
    - The output layer produces the final outcome
    - For classification, one output neuron for each class
  - Backpropagation is the most used training algorithm
    - Numerically fit the weights in order to produce output values as close as possible to the target values



3

# Convolutional Neural Networks (CNNs)

2

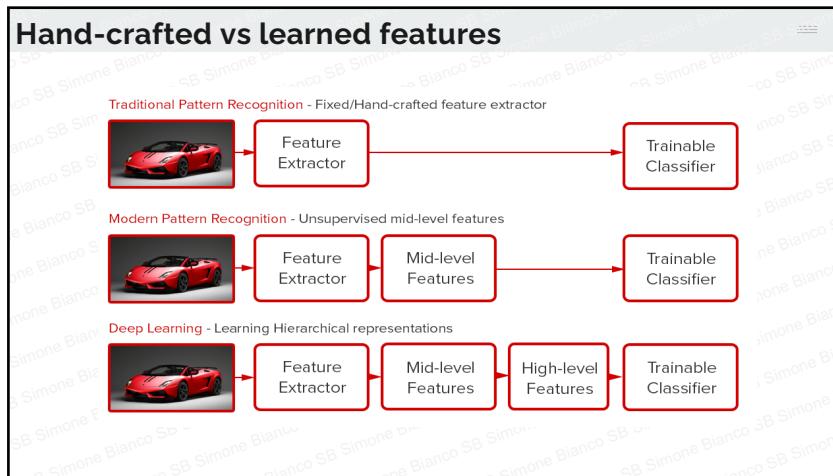
## Multi-layer perceptron

Network structure	Type of decision region	Solution to exclusive-OR problem	Classes with meshed regions	Most general decision surface shapes
Single layer	Single hyperplane			
Two layers	Open or closed convex regions			
Three layers	Arbitrary complexity (limited by the number of nodes)			

**FIGURE 12.23**  
Types of decision regions that can be formed by single- and multilayer feed-forward networks with one and two layers of hidden units and two inputs. (Lippman)

4

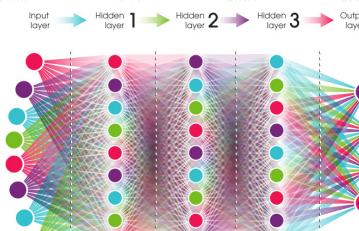
## Hand-crafted vs learned features



5

## What is Deep learning?

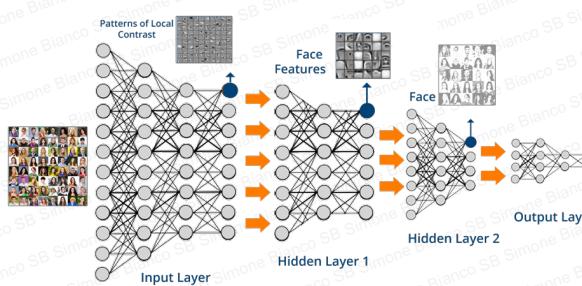
- Is a new area of Machine Learning research (ML), with the objective of moving ML closer to Artificial Intelligence
- It is about learning multiple levels (a hierarchy) of representations and abstractions



6

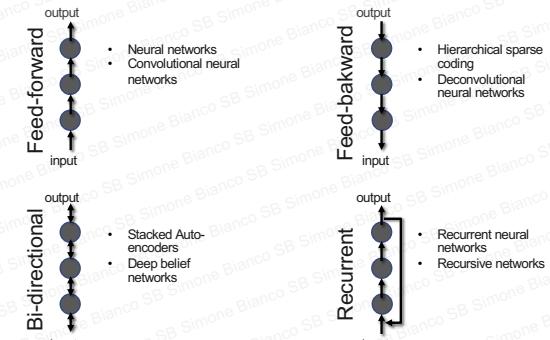
## Feature hierarchy

- Hierarchy of representations with increasing level of abstraction
- Each stage is a kind of trainable non-linear feature transform
- Can share the lower-level representations for multiple tasks



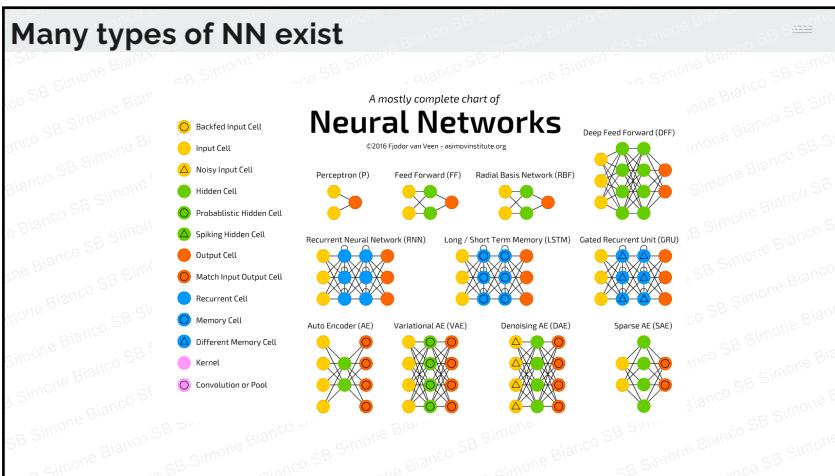
7

## Main types of deep architectures



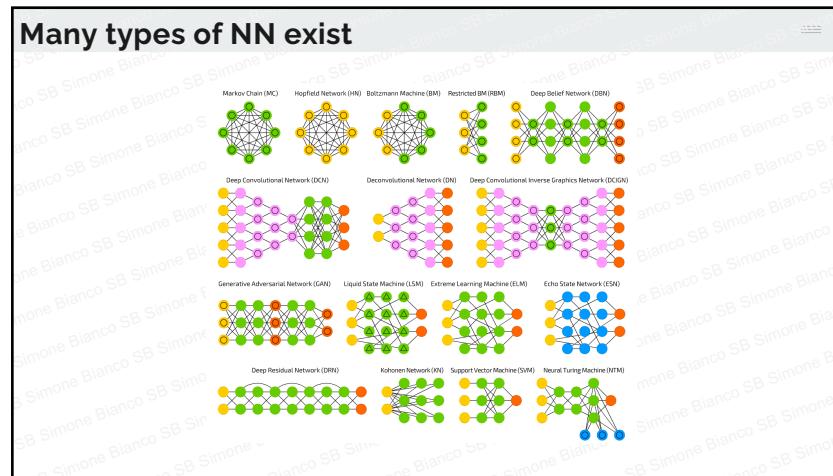
8

## Many types of NN exist



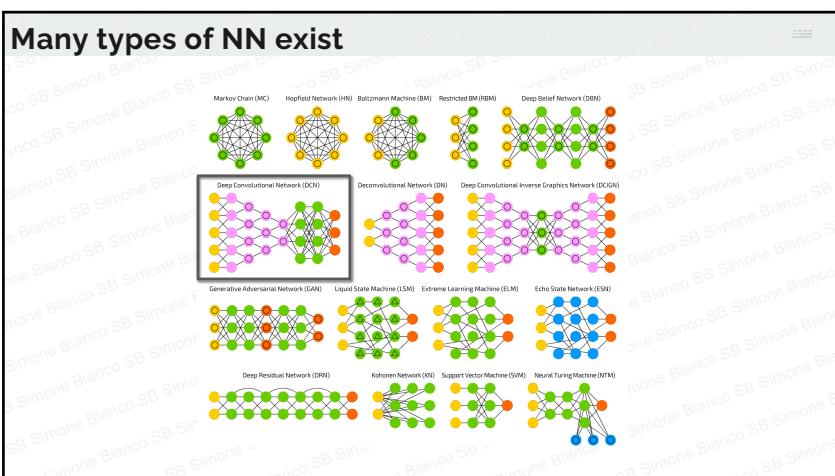
9

## Many types of NN exist



10

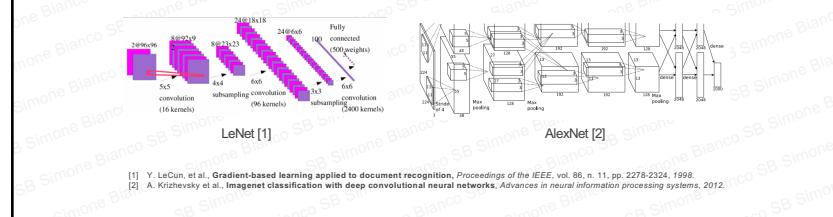
## Many types of NN exist



11

## Convolutional Neural Networks (CNNs)

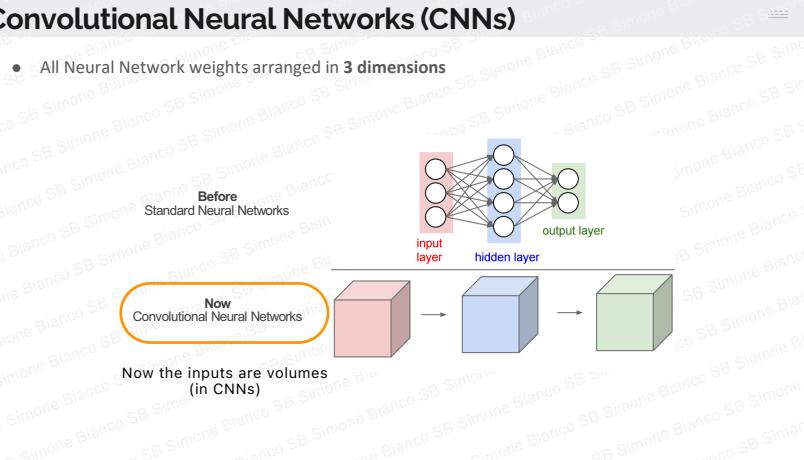
- Convolutional Neural Networks are **biologically-inspired architectures** made up of neural networks stuff
- CNNs are the most used architectures in computer vision
- CNNs stacks multiple stages of feature extractors
- Higher stages compute more global, more invariant features



12

## Convolutional Neural Networks (CNNs)

- All Neural Network weights arranged in **3 dimensions**



13

## Convolutional Neural Networks (CNNs)

- All Neural Network weights arranged in **3 dimensions**
- Convolutional Neural Networks are still Neural Networks but:

- Local connectivity

CNNs exploit the use of Convolution Kernels to analyse local regions  
Rather than connecting each neuron to the whole image,  
each neuron in a CNNs only connects to a small region of the image  
( $5 \times 5 \times 3$  patch with only 75 weights)

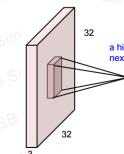


Image: 32x32x3 volume

Before: full connectivity  $32 \times 32 \times 3 = 3072$  weights

Now: one neuron will connect to, e.g.  $5 \times 5 \times 3$  patch and only have  $5 \times 5 \times 3 = 75$  weights

Note that connectivity is:

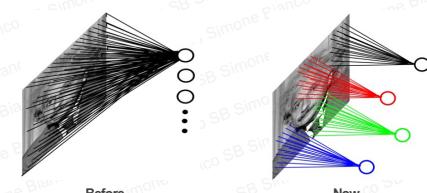
- Local in space ( $5 \times 5$  inside  $32 \times 32$ )
- Full in space (all 3 depth channels)

The region size is defined by the kernel size ( $5 \times 5$  in this case)

14

## Convolutional Neural Networks (CNNs)

- All Neural Network weights arranged in **3 dimensions**
- Convolutional Neural Networks are still Neural Networks but:
  - Local connectivity

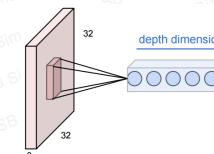


15

## Convolutional Neural Networks (CNNs)

- All Neural Network weights arranged in **3 dimensions**
- Convolutional Neural Networks are still Neural Networks but:

- Local connectivity



Before: hidden layer of 200 neurons

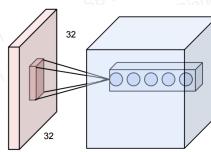
Now: output volume of depth 200

Multiple neurons all looking at the same region of the input volume, stacked along depth

16

## Convolutional Neural Networks (CNNs)

- All Neural Network weights arranged in **3 dimensions**
- Convolutional Neural Networks are still Neural Networks but:
  - Local connectivity



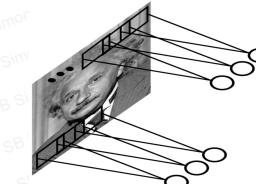
**Depth = # of hidden neurons**

These form a single [1 x 1 x depth] «depth column» in the output feature map volume

17

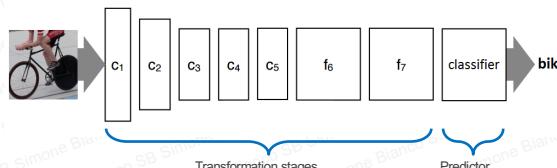
## Convolutional Neural Networks (CNNs)

- All Neural Network weights arranged in **3 dimensions**
- Convolutional Neural Networks are still Neural Networks but:
  - Local connectivity
  - Weights sharing (i.e., weights in the layer are shared across spatial positions)



18

## Overall CNN architecture



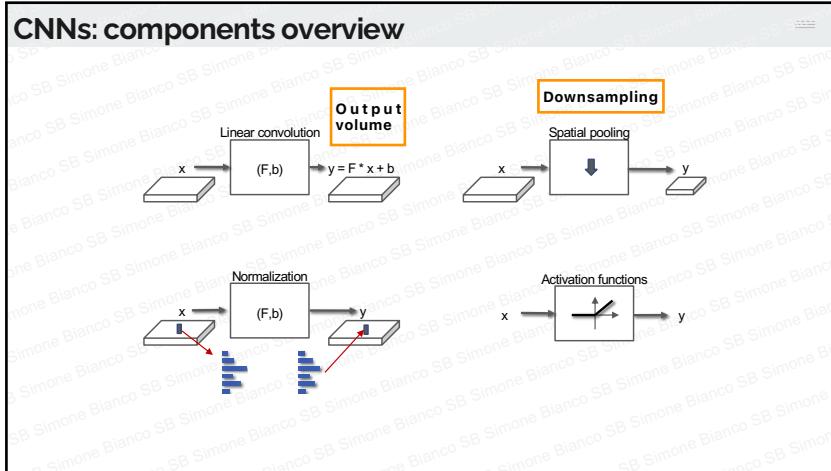
19

## CNNs: key ideas

- Feed-forward feature extraction
- Supervised training of convolutional filters by back-propagating classification error
- With respect to traditional Neural Networks, CNNs have other special layers:
  - Spatial pooling
  - Local response normalization
- These layers are useful to:
  - Reduce computational burden
  - Increase invariance
  - Ease the optimization

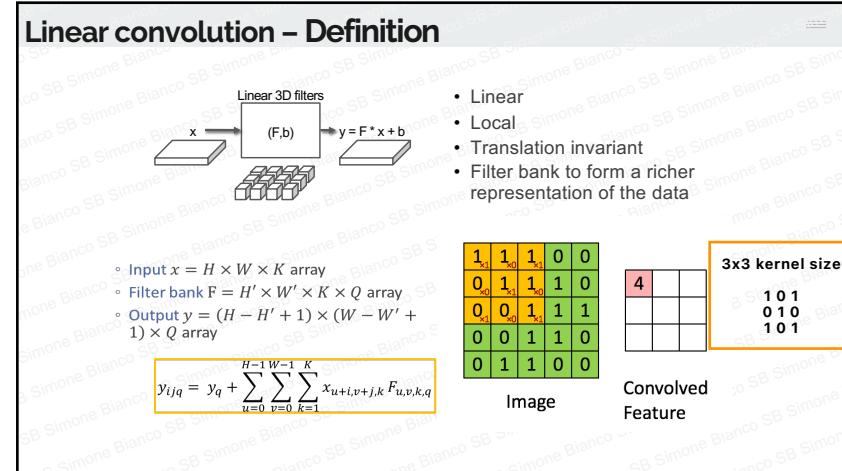
20

## CNNs: components overview



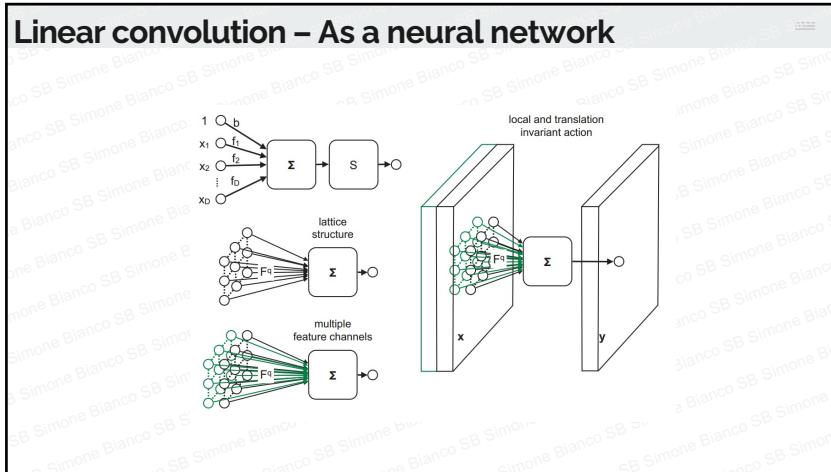
21

## Linear convolution – Definition



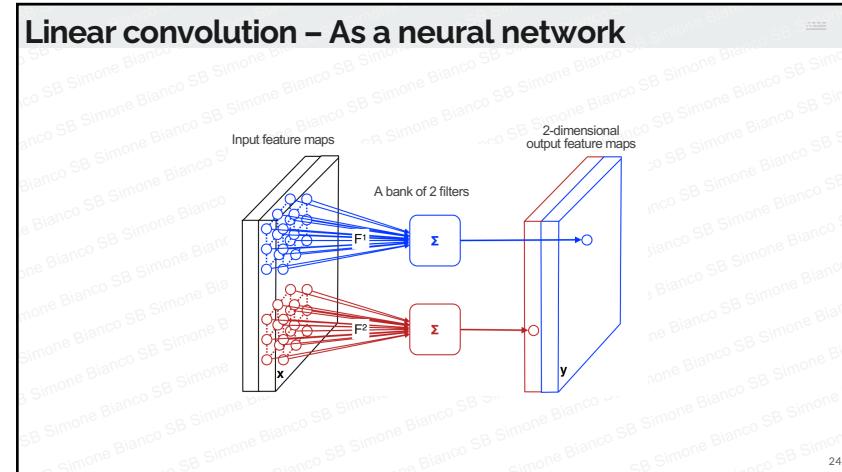
22

## Linear convolution – As a neural network



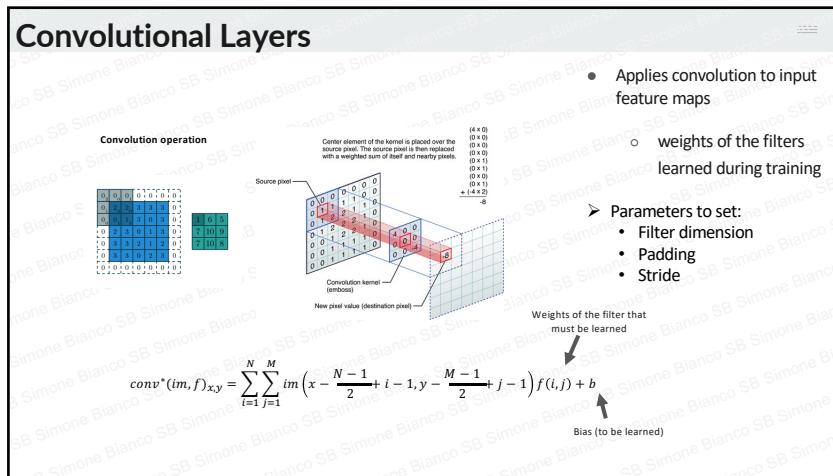
23

## Linear convolution – As a neural network



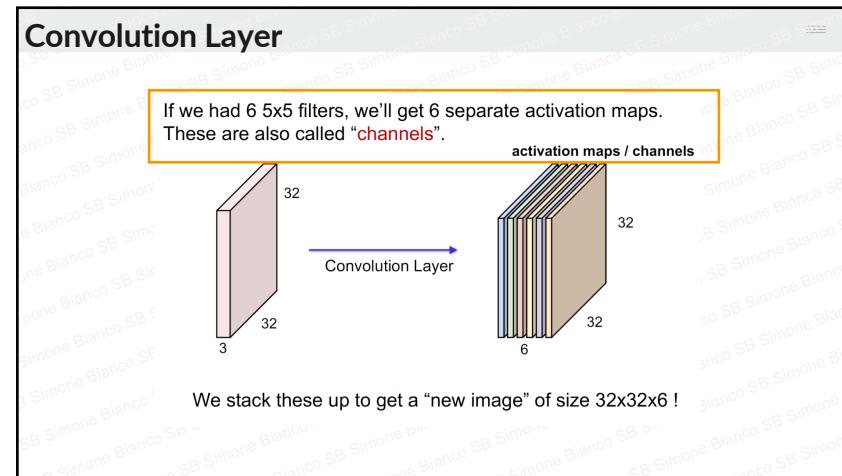
24

## Convolutional Layers



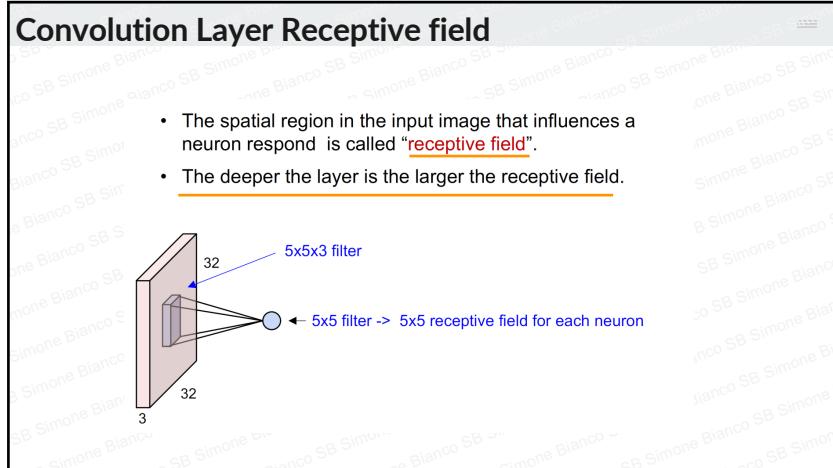
25

## Convolution Layer



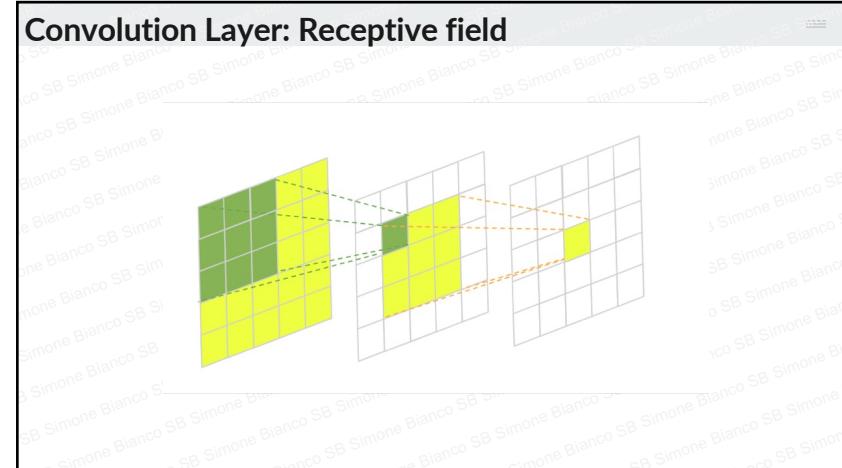
26

## Convolution Layer Receptive field



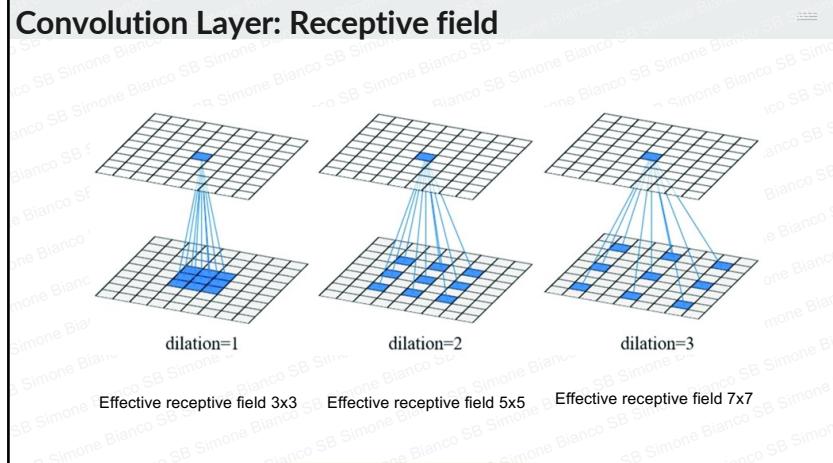
27

## Convolution Layer: Receptive field



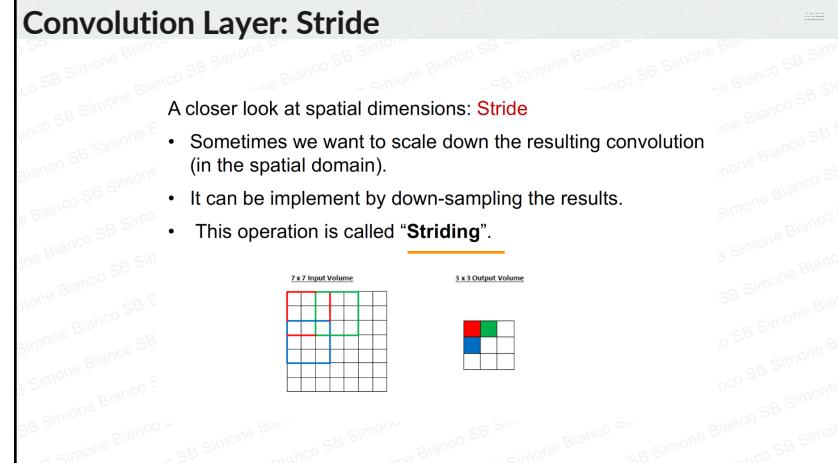
28

## Convolution Layer: Receptive field



29

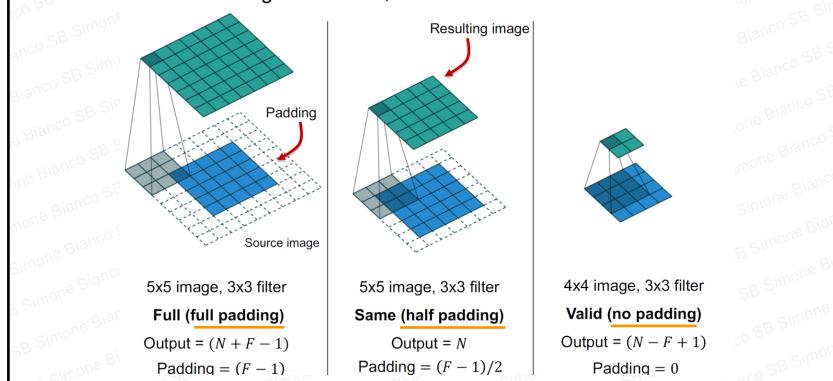
## Convolution Layer: Stride



30

## Convolution Layer: padding

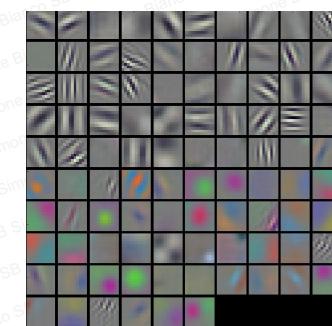
- Assume Image size  $N \times N$ , filter size  $F \times F$ :



31

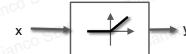
## Linear convolution – Filter bank example

- A bank of 96 filters of the first convolutional layer of AlexNet trained on ILSVRC2012 dataset
- Each one is a  $11 \times 11$  pixels 3D filter (it applies to a RGB image)



32

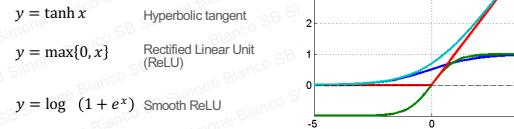
## Activation functions – Definition



- Scalar non-linearity

$$y = \frac{1}{1 + e^{-x}}$$

Sigmoid



$$y = \tanh x$$

Hyperbolic tangent

$$y = \max\{0, x\}$$

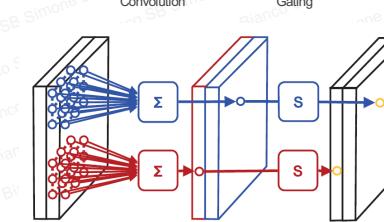
Rectified Linear Unit (ReLU)

$$y = \log(1 + e^x)$$

Smooth ReLU

33

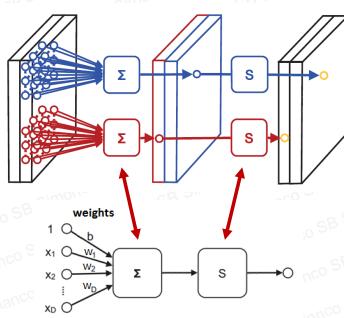
## Activation functions – As neural network



Filters are followed by non-linear operators (e.g. gating function)

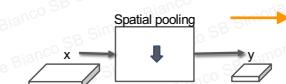
34

## Activation functions – As neural network



35

## Spatial pooling



- Pooling and sub-sampling
- Encodes translation invariance
- Pooling computes the average/max of the features in a neighborhood
- It is applied channel-by-channel

### max pooling

12	20	30	0
8	12	2	0
34	70	37	4
112	100	25	12

20	30
112	37

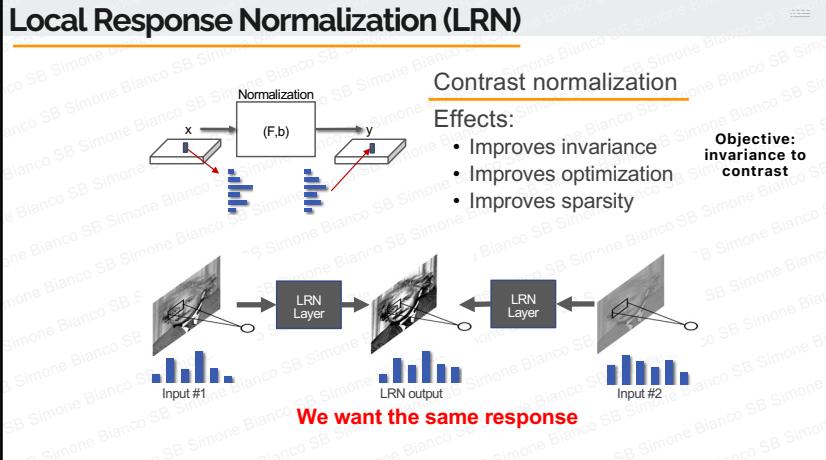
13	8
79	20

$$y_{ijk} = \max_{pq \in \Omega_{ij}} x_{pqk} \quad \text{Max pooling}$$

$$y_{ijk} = \frac{1}{|\Omega_{ij}|} \sum_{pq \in \Omega_{ij}} x_{pqk} \quad \text{Average pooling}$$

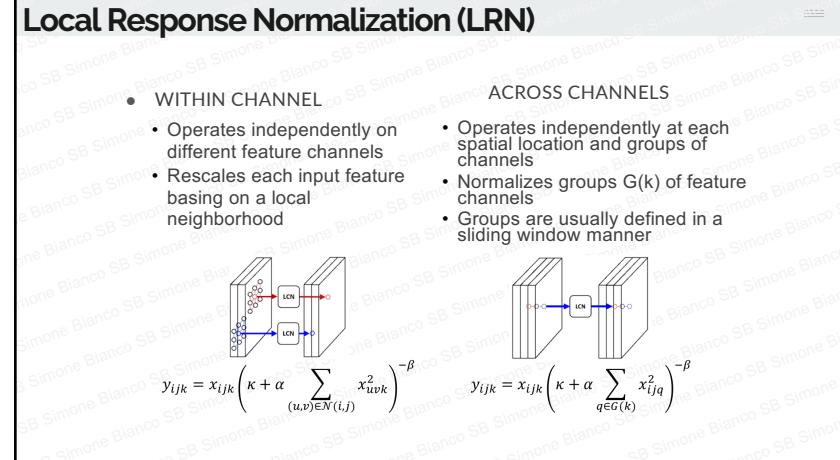
36

## Local Response Normalization (LRN)



37

## Local Response Normalization (LRN)



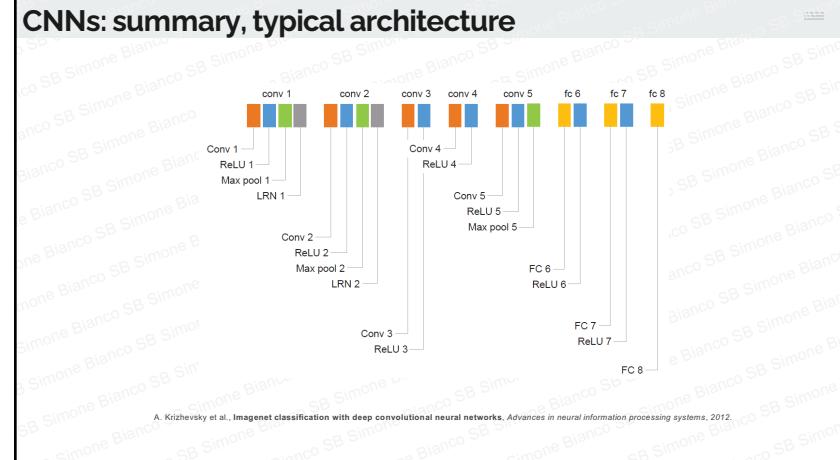
38

## CNNs: summary

- CNNs are **biologically-inspired** architectures
- Two key differences to Standard Neural Nets: neurons arranged in **3D volumes** have **local connectivity, share weights**
- Differently from traditional Neural Networks, CNNs have other special layers
- Spatial pooling
- Local response normalization

39

## CNNs: summary, typical architecture

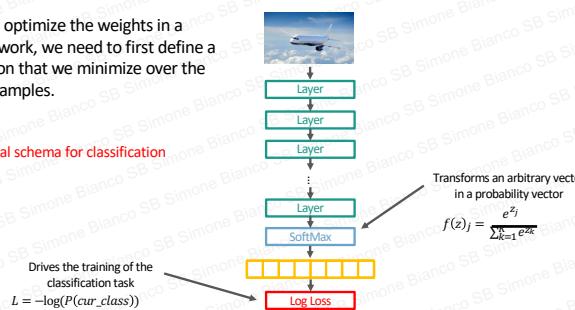


40

## Convolutional Neural Networks (CNNs)

In order to optimize the weights in a neural network, we need to first define a loss function that we minimize over the training examples.

- General schema for classification

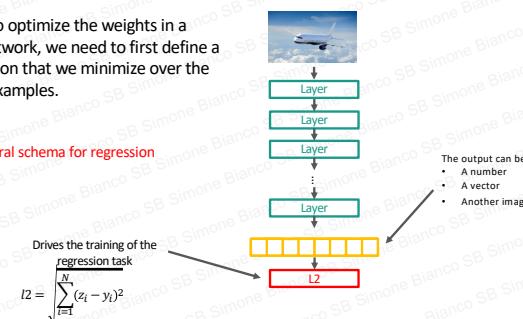


41

## Convolutional Neural Networks (CNNs)

In order to optimize the weights in a neural network, we need to first define a loss function that we minimize over the training examples.

- General schema for regression



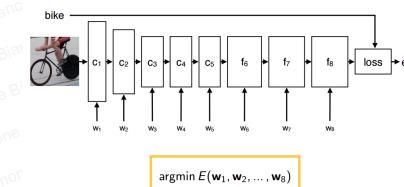
42

## CNN training

43

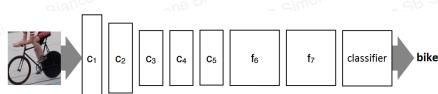
## Training CNNs

- Find CNN's weights  $w$  to minimize the **error** between ground truth labels and estimated labels
- Minimize using Stochastic Gradient Descent (SGD)
- This training method is called Backward-propagation

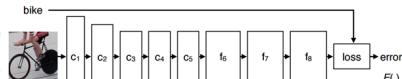


44

## Training CNNs



- A score function maps the raw image pixels to class scores (e.g. a linear function)
- A loss function measures the error between ground truth labels and estimated labels on the training data (e.g. Softmax/Euclidean)



45

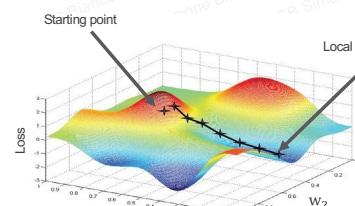
## Training CNNs

- Challenge
  - Many parameters
  - Prone to overfitting
- Key ingredients
  - Very large datasets
  - Stochastic Gradient Descent (SGD)
  - Regularization methods
  - Graphic Processing Unit (GPU)

46

## Stochastic Gradient Descent (SGD)

- Two steps
  1. Forward-propagation
  2. Backward-propagation



47

## Stochastic Gradient Descent (SGD)

- Two steps
  1. Forward-propagation
  2. Backward-propagation
- Refinements
- Epochs: all samples are visited sequentially, but in random order

48

## Stochastic Gradient Descent (SGD)

- Two steps
  1. Forward-propagation
  2. Backward-propagation
- Refinements
- Epochs: all samples are visited sequentially, but in random order
- Validation: evaluate on an held-out validation set to diagnose convergence

49

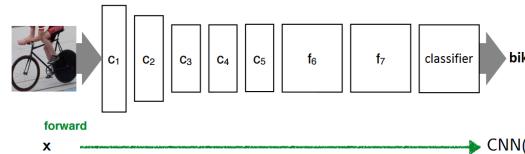
## Stochastic Gradient Descent (SGD)

- Two steps
  1. Forward-propagation
  2. Backward-propagation
- Refinements
- Epochs: all samples are visited sequentially, but in random order
- Validation: evaluate on an held-out validation set to diagnose convergence
- Learning rate: step size for each iteration
- Momentum: previous weight updates are remembered at each iteration

50

## Forward-propagation

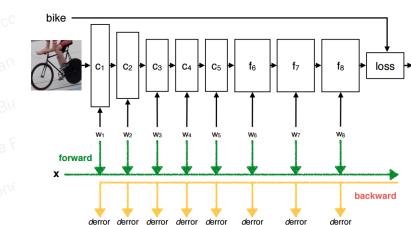
- Process of computing the output of the network given its input



51

## Backward-propagation

- Process of computing gradients of the loss w.r.t. parameters in a multi-layer neural network
- Compute derivatives using Chain Rule



52