# Deep Learning for Computer Vision

Andrii Liubonko

Samsung R&D Institute Ukraine

July 24, 2018

# Course overview

**Unit I** *DL overview. Classification. Modern CNNs.*

**Unit II** *Object Object Localization. Detection. Semantic Segmentation.*

**Unit III** *Metrics learning. Perspective.*

# Contents

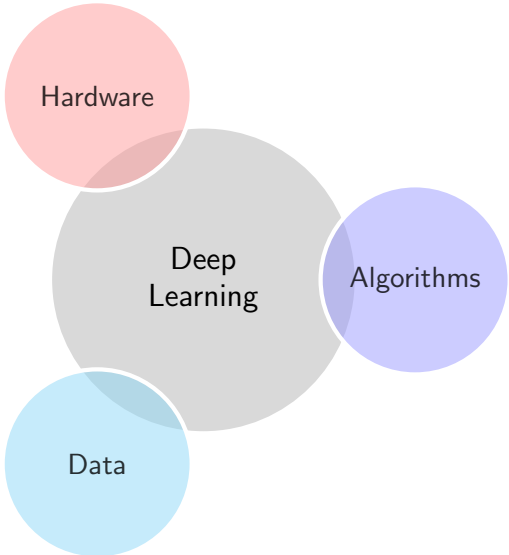
Deep Learning for Computer Vision.

Quick and dirty ML overview.

Basic elements of CNN
- Motivation
- Structure
- Structure [convolution layer]
- Structure [fully-connected layer]
- Case study

Summary

# Deep Learning for Computer Vision.

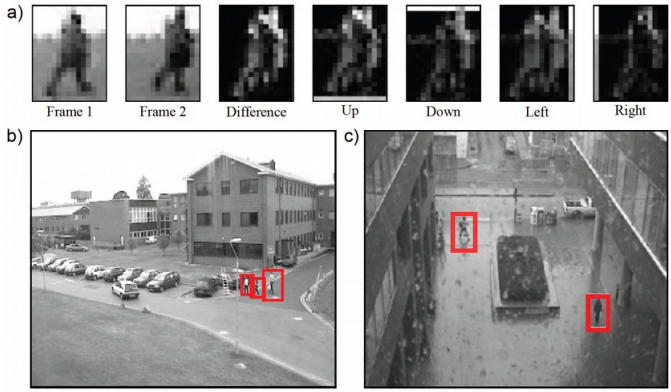


# Deep Learning for Computer Vision.

*Classification [e.g genre classification]*

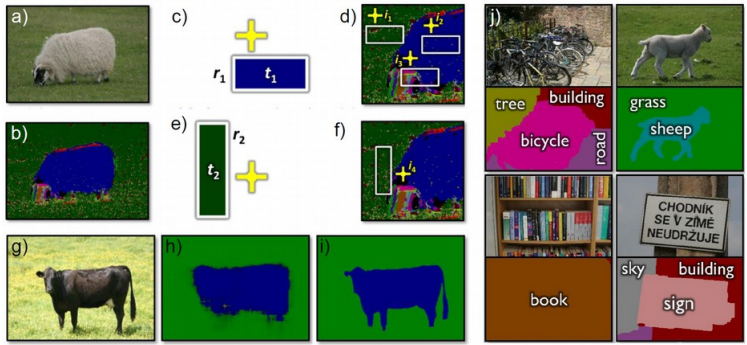
# Deep Learning for Computer Vision.

*Object detection [e.g. pedestrian detection]*



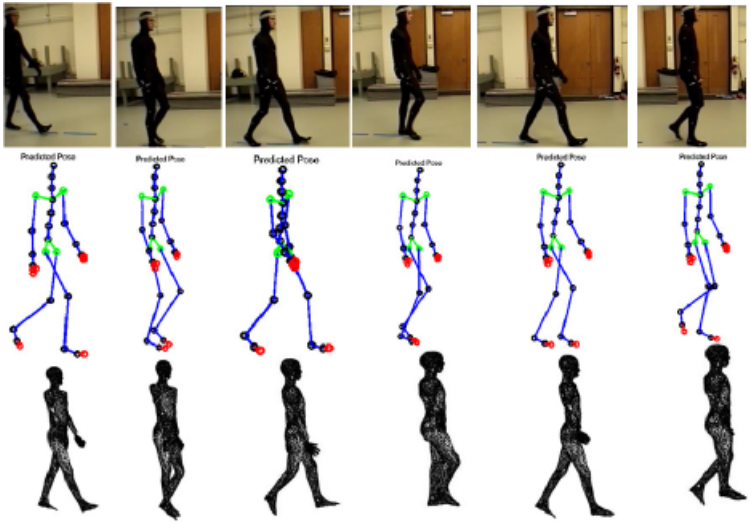
# Deep Learning for Computer Vision.

*Semantic segmentation*



# Deep Learning for Computer Vision.

*Human pose estimation*



# Contents

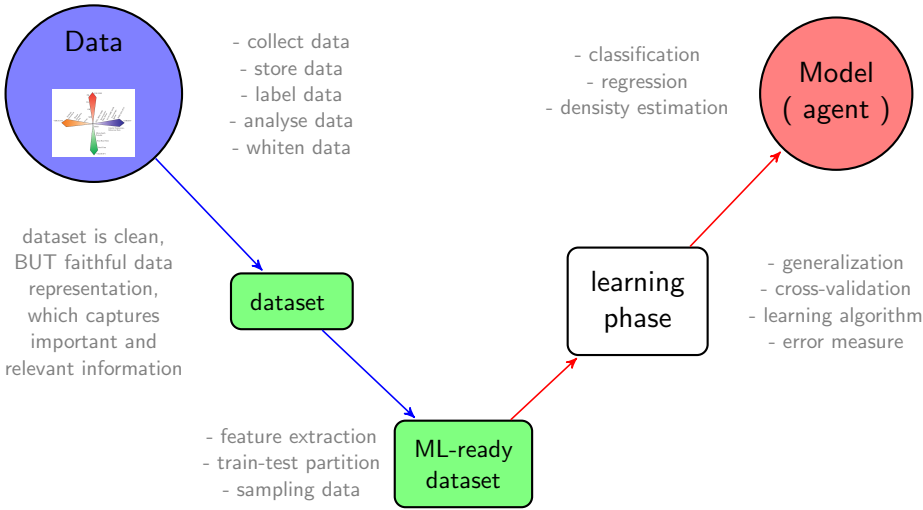
Deep Learning for Computer Vision.

Quick and dirty ML overview.

Basic elements of CNN
- Motivation
- Structure
- Structure [convolution layer]
- Structure [fully-connected layer]
- Case study

Summary

# Quick and dirty ML overview.



# Quick and dirty ML overview.[Types of learning]

- Supervised
- Unsupervised
- Semi-supervised
- RL

# Quick and dirty ML overview.[Types of learning]

**The goal of supervised learning** is to learn a mapping from inputs $\mathbf{x}$ to outputs $y$, given a labeled set of input-output pairs

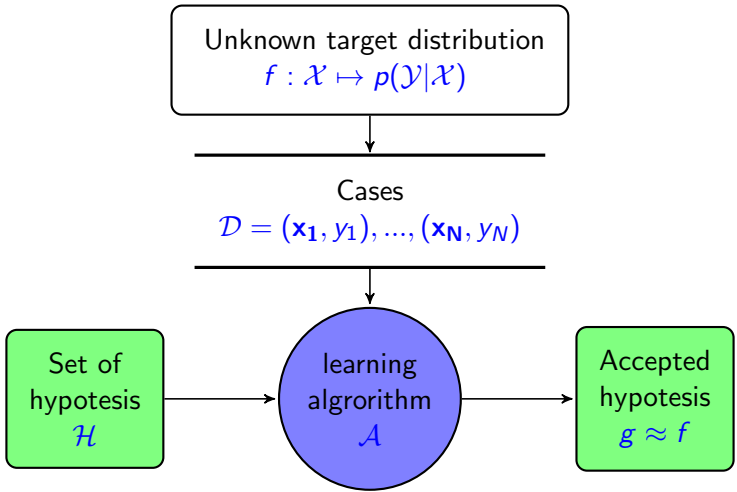
$$\mathcal{D} = (\mathbf{x_i}, y_i)_{i=1}^{N}$$

$\mathcal{D}$ is called the **training set**, and $N$ is the number of training examples.

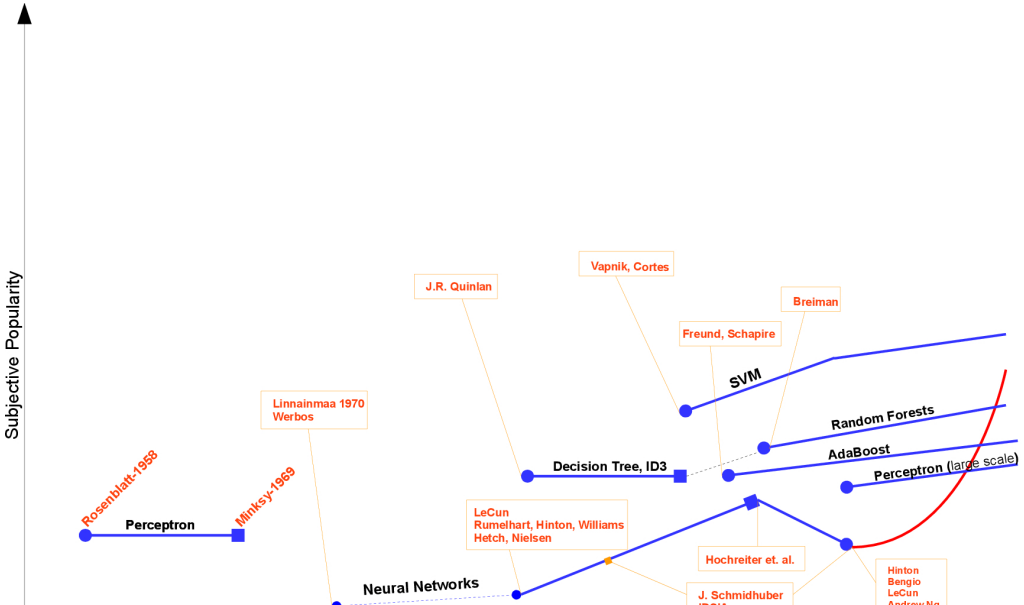
In the simplest setting, each training input $\mathbf{x_i}$ is a $D$-dimensional vector of numbers. These are called **features** or attributes.

In general $\mathbf{x_i}$ could be a complex structured object, such as an image, a sentence, an email message, a time series, a molecular shape, a graph, etc.

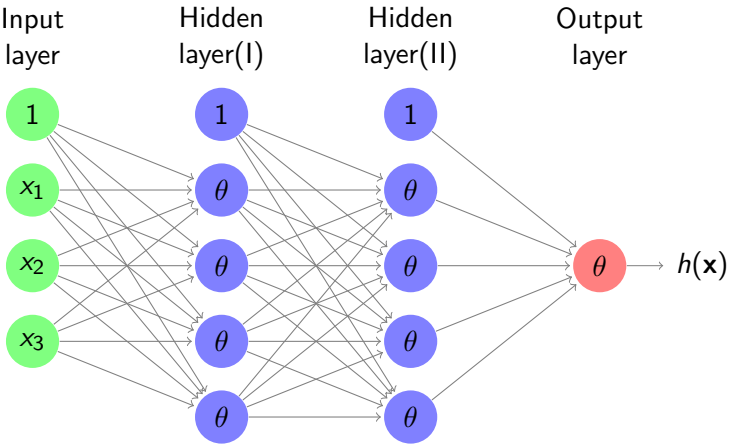
# Quick and dirty ML overview.[Types of learning]



# Quick and dirty ML overview.[Types of learning]



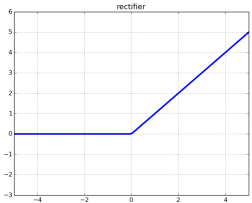
# Quick and dirty ML overview. [NN - Architecture]



# Quick and dirty ML overview. [NN - Architecture]

parameters in NN:

$$\omega_{ij}^{(l)} = \begin{cases} 1 \leq l \leq L & \text{layers} \\ 0 \leq i \leq d^{(l-1)} & \text{inputs} \\ 1 \leq j \leq d^{(l)} & \text{outputs} \end{cases}$$




activation:

$$x_j^{(l)} = \theta(s_j^{(l)}) = \theta\Big(\sum_{i=0}^{d^{(l-1)}} \omega_{ij}^{(l)} x_i^{(l-1)}\Big)$$

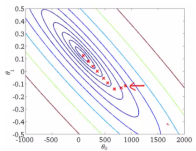
$$\theta(s) = RELU(s) = max(0, x)$$

# Quick and dirty ML overview. [NN - Architecture]

Define **Loss (or Cost) function**:

$$L_{logloss} = \frac{1}{N}\sum_{n=1}^{N}\sum_{k=1}^{M} y_{nk} \cdot log(p_{nk})$$

$$L_{rmse} = \frac{1}{N}\sum_{n=1}^{N} \parallel F(\mathbf{x_n}) - y_i \parallel_2$$




**Gradient Descent (GD)** minimizes:

$$L_{train}(\omega) = \frac{1}{N}\sum_{n=1}^{N} e(F(\mathbf{x_n}), y_n)$$

by iterative steps along $-\nabla L_{train}$:

$$\Delta\omega = -\eta \nabla L_{train}(\omega)$$
$$\omega_{prev} = \omega_{next} + \Delta\omega$$

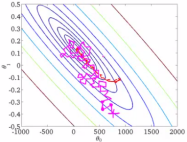
If $\nabla L_{train}(\omega)$ is based on all examples $\{\mathbf{x_n}, y_n\}$ then it is called **batch gradient descent**

# Quick and dirty ML overview. [NN - Architecture]

Define **Loss (or Cost) function**:

$$L_{logloss} = \frac{1}{N}\sum_{n=1}^{N}\sum_{k=1}^{M} y_{nk} \cdot log(p_{nk})$$

$$L_{rmse} = \frac{1}{N}\sum_{n=1}^{N} \parallel F(\mathbf{x_n}) - y_i \parallel_2$$




**Gradient Descent (GD)** minimizes:

$$L_{train}(\omega) = \frac{1}{N}\sum_{n=1}^{N} e(F(\mathbf{x_n}), y_n)$$

by iterative steps along $-\nabla L_{train}$:

$$\Delta\omega = -\eta \nabla L_{train}(\omega)$$
$$\omega_{prev} = \omega_{next} + \Delta\omega$$

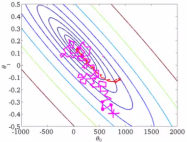
If $\nabla L_{train}(\omega)$ is based on all examples $\{\mathbf{x_n}, y_n\}$ then it is called **stochastic (mini-batch) gradient descent - SGD**

# Quick and dirty ML overview. [NN - Architecture]

Define **Loss (or Cost) function**:

$$L_{logloss} = \frac{1}{N}\sum_{n=1}^{N}\sum_{k=1}^{M} y_{nk} \cdot log(p_{nk})$$

$$L_{rmse} = \frac{1}{N}\sum_{n=1}^{N} \parallel F(\mathbf{x_n}) - y_i \parallel_2$$




**Gradient Descent (GD)** minimizes:

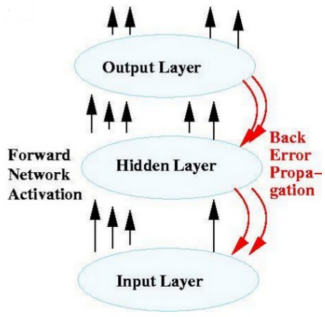
$$L_{train}(\omega) = \frac{1}{N}\sum_{n=1}^{N} e(F(\mathbf{x_n}), y_n)$$

by iterative steps along $-\nabla L_{train}$:

$$\Delta\omega = -\eta \nabla L_{train}(\omega)$$
$$\omega_{prev} = \omega_{next} + \Delta\omega$$

If $\nabla L_{train}(\omega)$ is based on all examples $\{\mathbf{x_n}, y_n\}$ then it is called **momentum, rmsprop, adagrad, adam**

# Quick and dirty ML overview. [NN - Architecture]



# Contents

Deep Learning for Computer Vision.

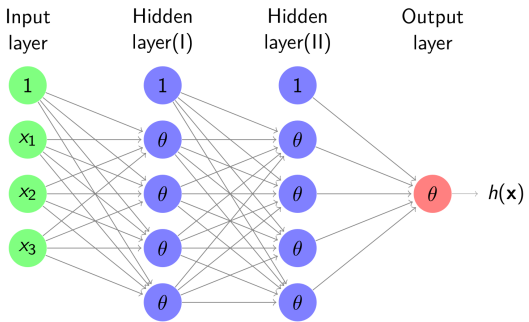
Quick and dirty ML overview.

Basic elements of CNN
- Motivation
- Structure
- Structure [convolution layer]
- Structure [fully-connected layer]
- Case study

Summary

# Basic elements of CNN

Image processing with standard Feed-forward Neural Networks




Input layer
$64 \times 64 = 4096$

Weights: $4096\times$ (size of second layer)

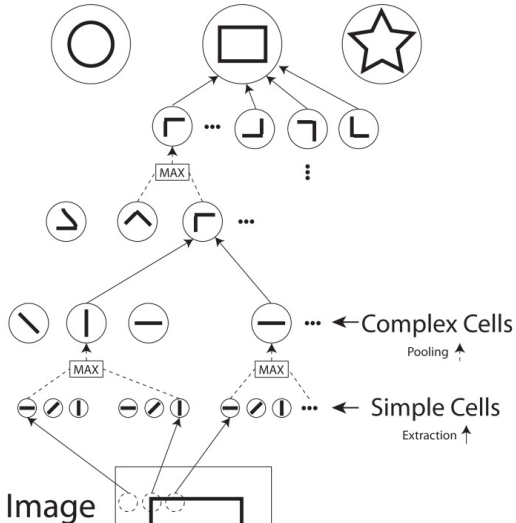
$=>$ millions of parameters (for just one layer)

Need for alternative architecture

# Basic elements of CNN

## Motivation

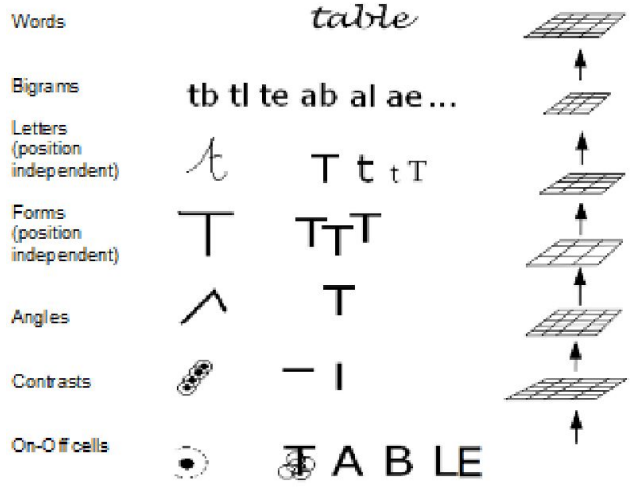
Hubel & Wiesel (1960s) => Hierarchical organization



# Basic elements of CNN

## Motivation

Hubel & Wiesel (1960s) => Hierarchical organization



# Contents

Deep Learning for Computer Vision.

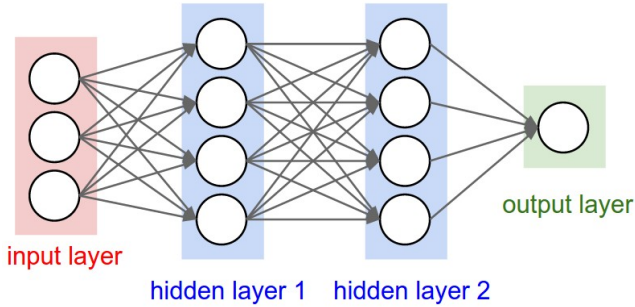
Quick and dirty ML overview.

Basic elements of CNN
- Motivation
- Structure
- Structure [convolution layer]
- Structure [fully-connected layer]
- Case study

Summary

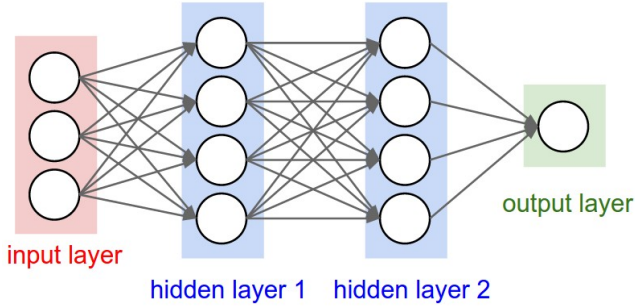
# Basic elements of CNN

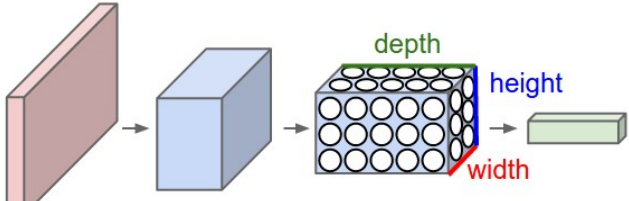
## Structure



# Basic elements of CNN

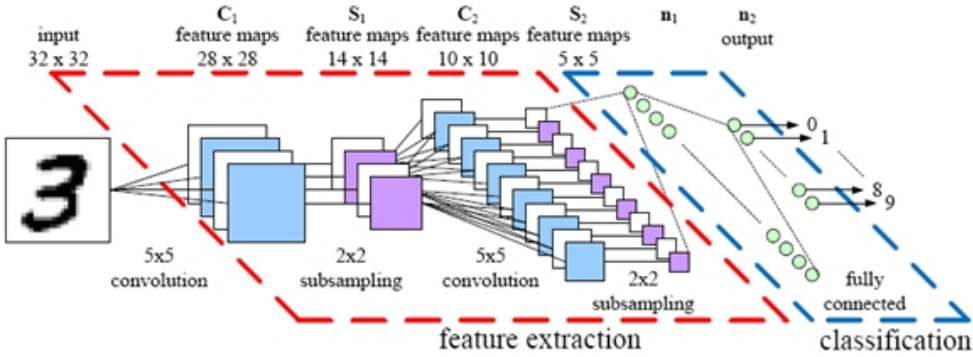
## Structure



# Basic elements of CNN

## Structure



# Basic elements of CNN

## Structure

- INPUT holds the raw pixel values of the image.
- CONV layer computes the output of neurons that are connected to local regions in the input, each computing a dot product between their weights and the region they are connected to in the input volume.
- POOL layer performs a downsampling operation along the spatial dimensions (width, height).
- FC (i.e. fully-connected) layer computes the class scores. As with ordinary Neural Networks and as the name implies, each neuron in this layer is connected to all the numbers in the previous volume.

# Contents

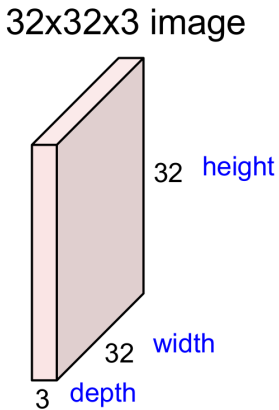
Deep Learning for Computer Vision.

Quick and dirty ML overview.

Basic elements of CNN
- Motivation
- Structure
- Structure [convolution layer]
- Structure [fully-connected layer]
- Case study

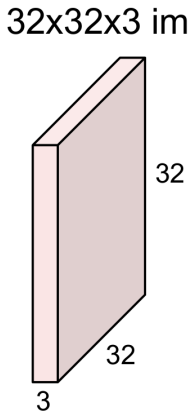
Summary

# Basic elements of CNN

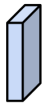



[ from *Fei-Fei Li & Andrej Karpathy & Justin Johnson "CNN for Computer Vision"*]

# Basic elements of CNN



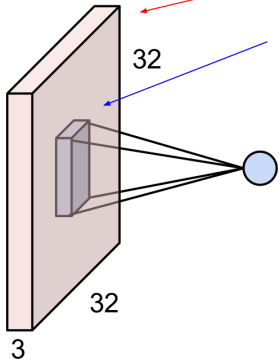

5x5x3 filter



**Convolve** the filter with the image
i.e. “slide over the image spatially,
computing dot products”

[ from *Fei-Fei Li & Andrej Karpathy & Justin Johnson "CNN for Computer Vision"*]

# Basic elements of CNN

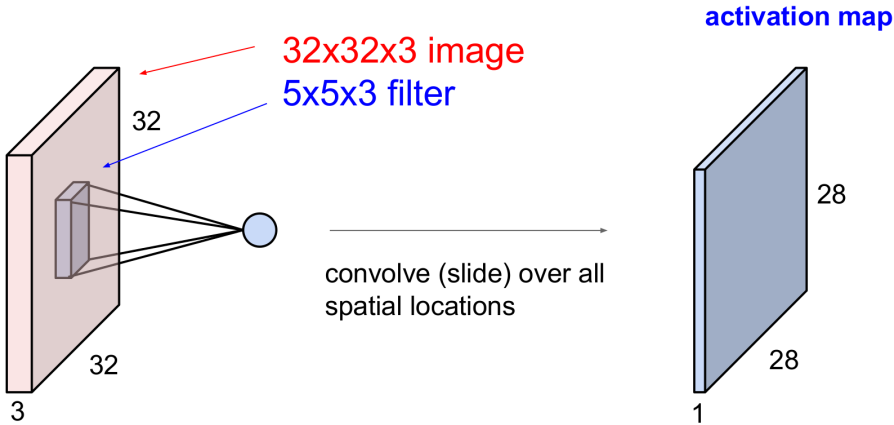



**1 number:**
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. 5*5*3 = 75-dimensional dot product + bias)

$$w^T x + b$$

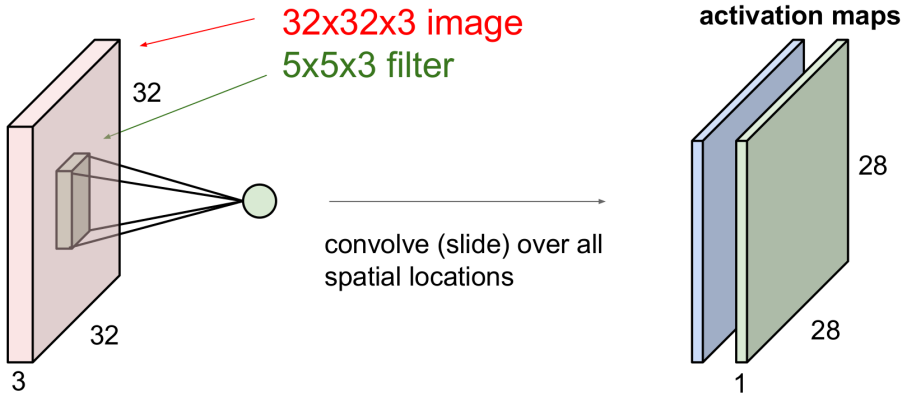
[ from *Fei-Fei Li & Andrej Karpathy & Justin Johnson "CNN for Computer Vision"*]

# Basic elements of CNN



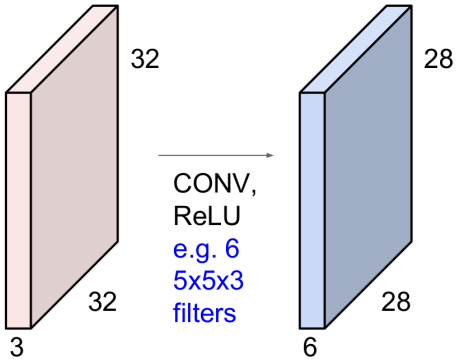

[ from *Fei-Fei Li & Andrej Karpathy & Justin Johnson "CNN for Computer Vision"*]

# Basic elements of CNN



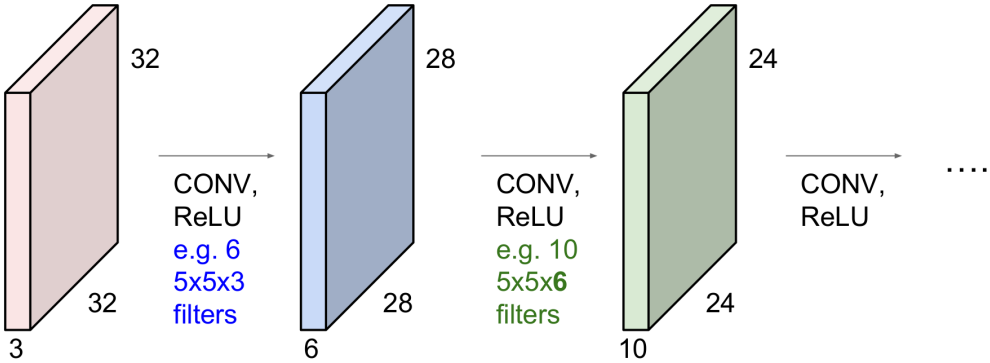

[ from *Fei-Fei Li & Andrej Karpathy & Justin Johnson "CNN for Computer Vision"*]

# Basic elements of CNN



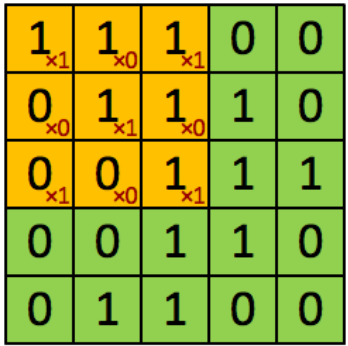

[ from *Fei-Fei Li & Andrej Karpathy & Justin Johnson "CNN for Computer Vision"*]

# Basic elements of CNN



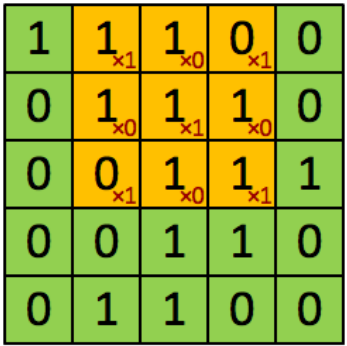

[ from *Fei-Fei Li & Andrej Karpathy & Justin Johnson "CNN for Computer Vision"*]

# Basic elements of CNN



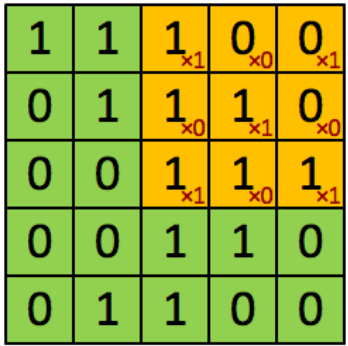

[ from *Fei-Fei Li & Andrej Karpathy & Justin Johnson "CNN for Computer Vision"*]

# Basic elements of CNN



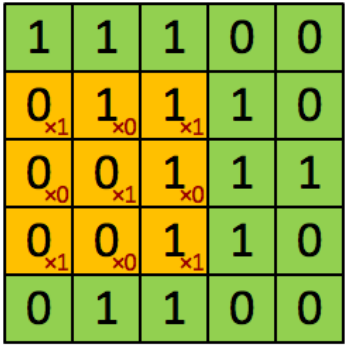

[ from *Fei-Fei Li & Andrej Karpathy & Justin Johnson "CNN for Computer Vision"*]

# Basic elements of CNN



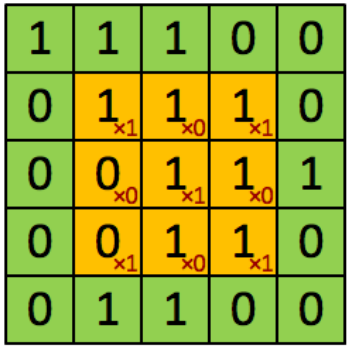

[ from *Fei-Fei Li & Andrej Karpathy & Justin Johnson "CNN for Computer Vision"*]

# Basic elements of CNN



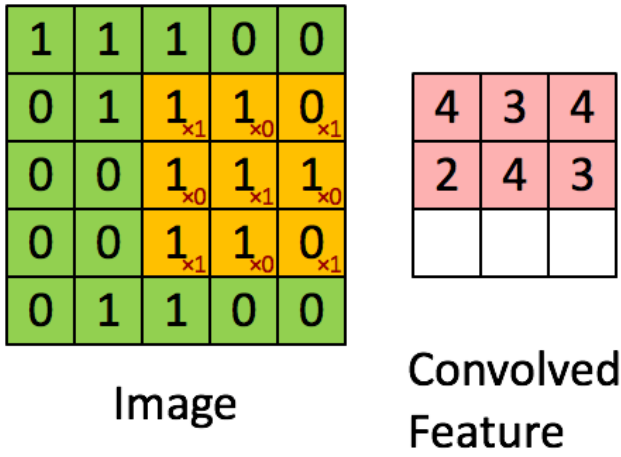

[ from *Fei-Fei Li & Andrej Karpathy & Justin Johnson "CNN for Computer Vision"*]

# Basic elements of CNN



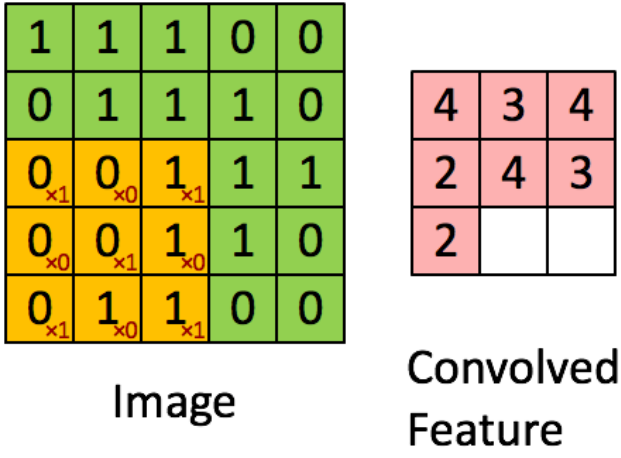

[ from *Fei-Fei Li & Andrej Karpathy & Justin Johnson "CNN for Computer Vision"*]

# Basic elements of CNN



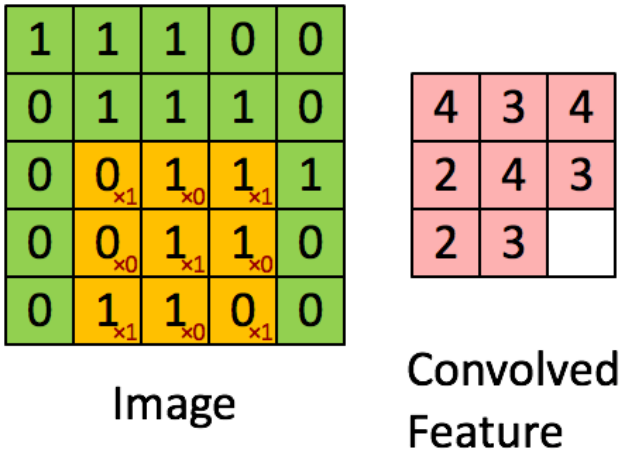

[ from *Fei-Fei Li & Andrej Karpathy & Justin Johnson "CNN for Computer Vision"*]

# Basic elements of CNN



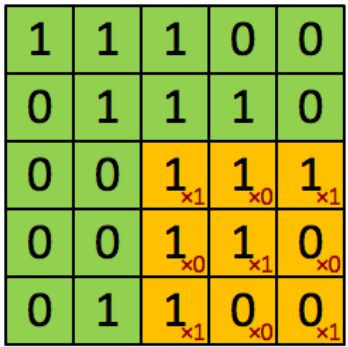

[ from *Fei-Fei Li & Andrej Karpathy & Justin Johnson "CNN for Computer Vision"*]

# Basic elements of CNN



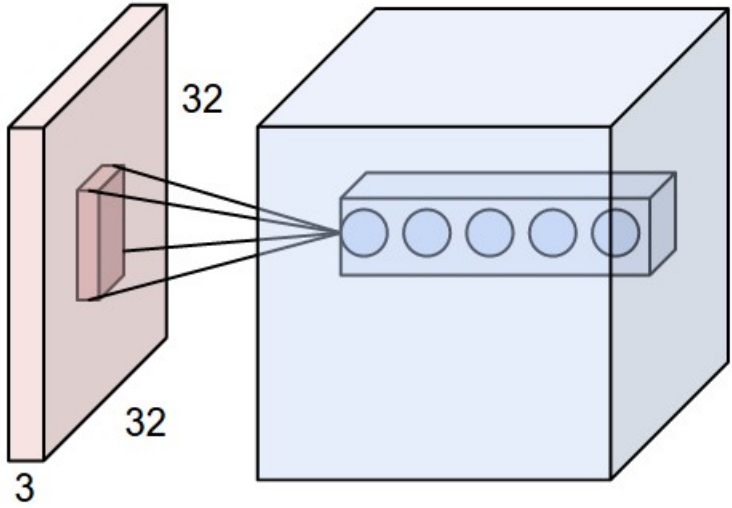

[ from *Fei-Fei Li & Andrej Karpathy & Justin Johnson "CNN for Computer Vision"*]

# Basic elements of CNN




[ from *Fei-Fei Li & Andrej Karpathy & Justin Johnson "CNN for Computer Vision"*]

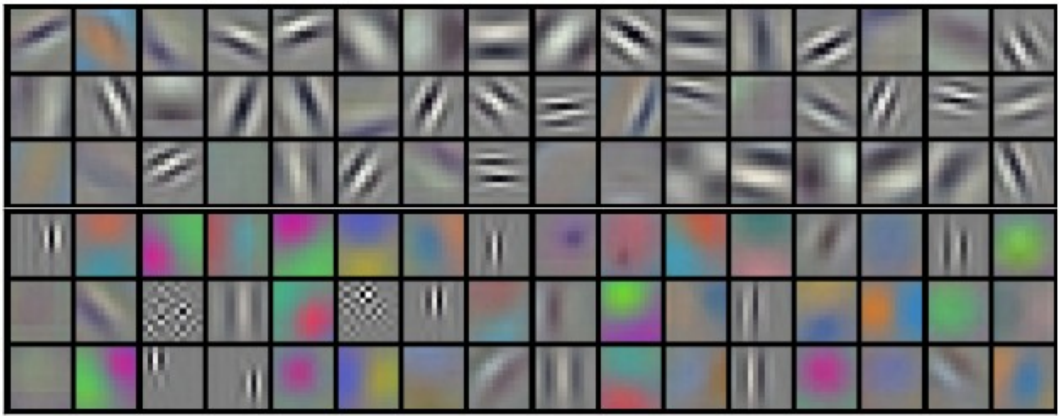
# Basic elements of CNN




[ from *Fei-Fei Li & Andrej Karpathy & Justin Johnson "CNN for Computer Vision"*]

# Basic elements of CNN

## Structure [convolution layer]



[ from *Fei-Fei Li & Andrej Karpathy & Justin Johnson "CNN for Computer Vision"*]

# Basic elements of CNN

- Accepts a volume of size $W1 \times H1 \times D1$
- Requires four hyperparameters:
  - Number of filters $K$,
  - their spatial extent $F$,
  - the stride $S$,
  - the amount of zero padding $P$.
- Produces a volume of size $W2 \times H2 \times D2$ where:
  - $W2 = (W1 - F + 2P)/S + 1$,
  - $H2 = (H1 - F + 2P)/S + 1$
  - $D2 = K$
- With parameter sharing, it introduces $F \times F \times D1$ weights per filter, for a total of $(F \times F \times D1) \times K$ weights and $K$ biases.
- In the output volume, the $d$-th depth slice (of size $W2 \times H2$) is the result of performing a valid convolution of the $d$-th filter over the input volume with a stride of $S$, and then offset by $d$-th bias.

# Contents

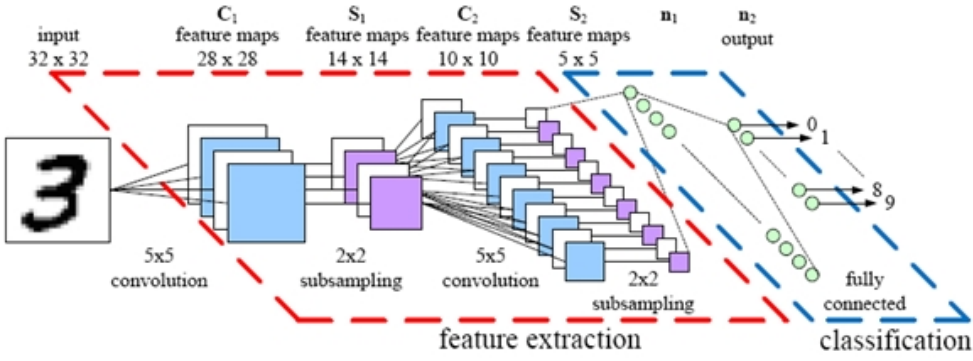
Deep Learning for Computer Vision.

Quick and dirty ML overview.

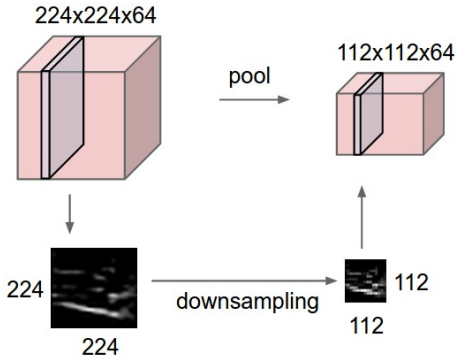
Basic elements of CNN
- Motivation
- Structure
- Structure [convolution layer]
- Structure [fully-connected layer]
- Case study

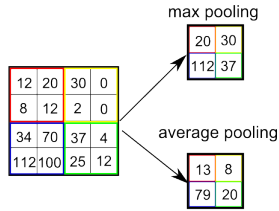
Summary

# Basic elements of CNN



# Basic elements of CNN




[ from *Fei-Fei Li & Andrej Karpathy & Justin Johnson "CNN for Computer Vision"*]

# Basic elements of CNN

- Accepts a volume of size $W1 \times H1 \times D1$
- Requires three hyperparameters:
  - their spatial extent $F$,
  - the stride $S$,
- Produces a volume of size $W2 \times H2 \times D2$ where:
  - $W2 = (W1 - F)/S + 1$
  - $H2 = (H1 - F)/S + 1$
  - $D2 = D1$

# Contents

Deep Learning for Computer Vision.

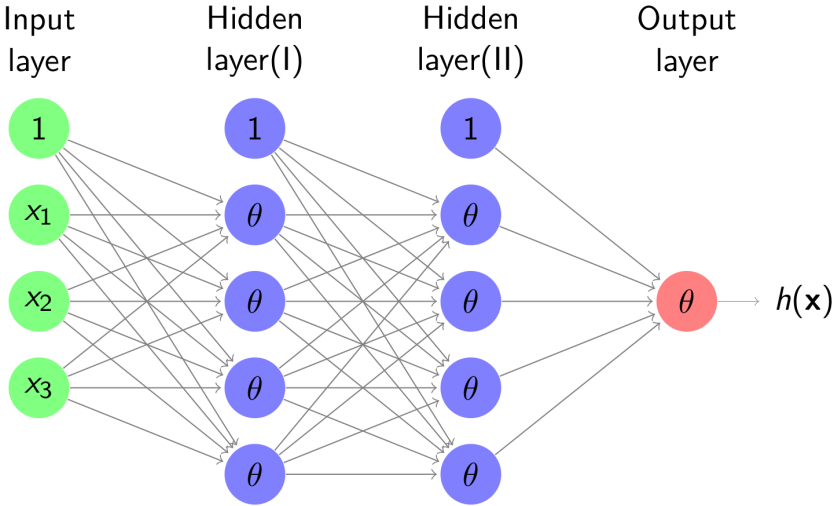
Quick and dirty ML overview.

Basic elements of CNN
- Motivation
- Structure
- Structure [convolution layer]
- Structure [fully-connected layer]
- Case study

Summary

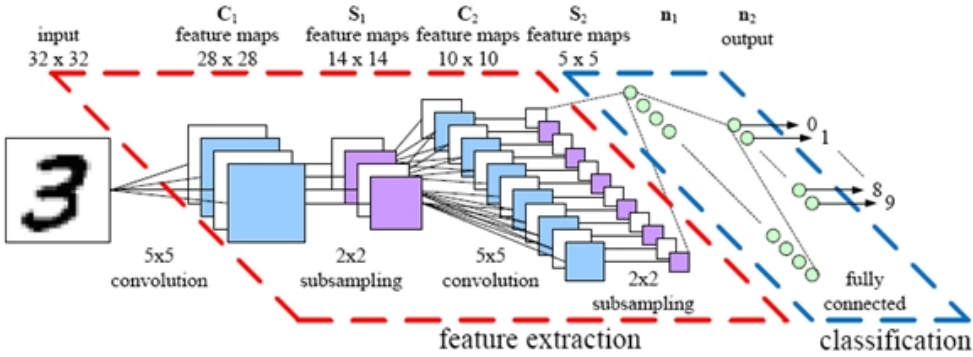
# Basic elements of CNN

## Structure [fully-connected layer]



# Basic elements of CNN

## Structure



# Contents

Deep Learning for Computer Vision.

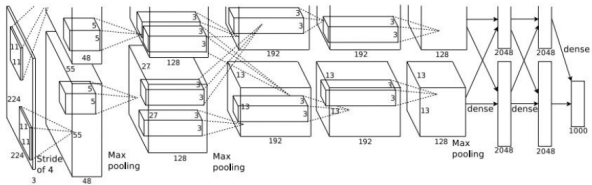
Quick and dirty ML overview.

Basic elements of CNN
- Motivation
- Structure
- Structure [convolution layer]
- Structure [fully-connected layer]
- Case study

Summary

# Case Study: AlexNet

*[Krizhevsky et al. 2012]*




Full (simplified) AlexNet architecture:
[227x227x3] INPUT
[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0
[27x27x96] MAX POOL1: 3x3 filters at stride 2
[27x27x96] NORM1: Normalization layer
[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2
[13x13x256] MAX POOL2: 3x3 filters at stride 2
[13x13x256] NORM2: Normalization layer
[13x13x384] CONV3: 384 3x3 filters at stride 1, pad 1
[13x13x384] CONV4: 384 3x3 filters at stride 1, pad 1
[13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1
[6x6x256] MAX POOL3: 3x3 filters at stride 2
[4096] FC6: 4096 neurons
[4096] FC7: 4096 neurons
[1000] FC8: 1000 neurons (class scores)

**Details/Retrospectives:**
- first use of ReLU
- used Norm layers (not common anymore)
- heavy data augmentation
- dropout 0.5
- batch size 128
- SGD Momentum 0.9
- Learning rate 1e-2, reduced by 10 manually when val accuracy plateaus
- L2 weight decay 5e-4
- 7 CNN ensemble: 18.2% -> 15.4%

[ from *Fei-Fei Li & Andrej Karpathy & Justin Johnson "CNN for Computer Vision"*]

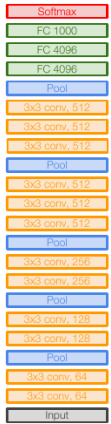
# Basic elements of CNN

## Case study

(not counting biases)

INPUT: [224x224x3] memory: 224*224*3=150K params: 0
CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*3)*64 = 1,728
CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*64)*64 = 36,864
POOL2: [112x112x64] memory: 112*112*64=800K params: 0
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*64)*128 = 73,728
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*128)*128 = 147,456
POOL2: [56x56x128] memory: 56*56*128=400K params: 0
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*128)*256 = 294,912
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
POOL2: [28x28x256] memory: 28*28*256=200K params: 0
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*256)*512 = 1,179,648
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
POOL2: [14x14x512] memory: 14*14*512=100K params: 0
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
POOL2: [7x7x512] memory: 7*7*512=25K params: 0
FC: [1x1x4096] memory: 4096 params: 7*7*512*4096 = 102,760,448
FC: [1x1x4096] memory: 4096 params: 4096*4096 = 16,777,216
FC: [1x1x1000] memory: 1000 params: 4096*1000 = 4,096,000

TOTAL memory: 24M * 4 bytes ~= 96MB / image (only forward! ~*2 for bwd)
TOTAL params: 138M parameters

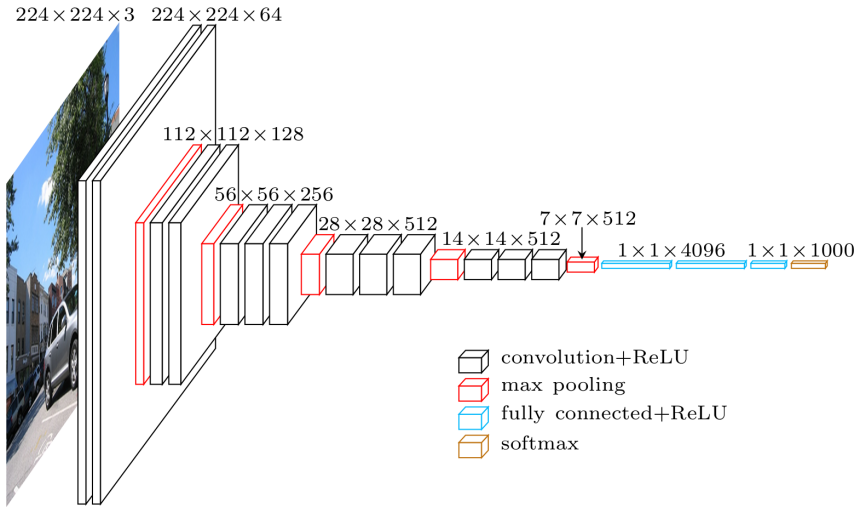



VGG16

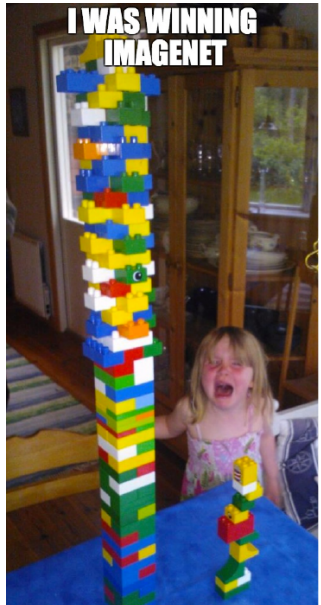
Common names

# Basic elements of CNN

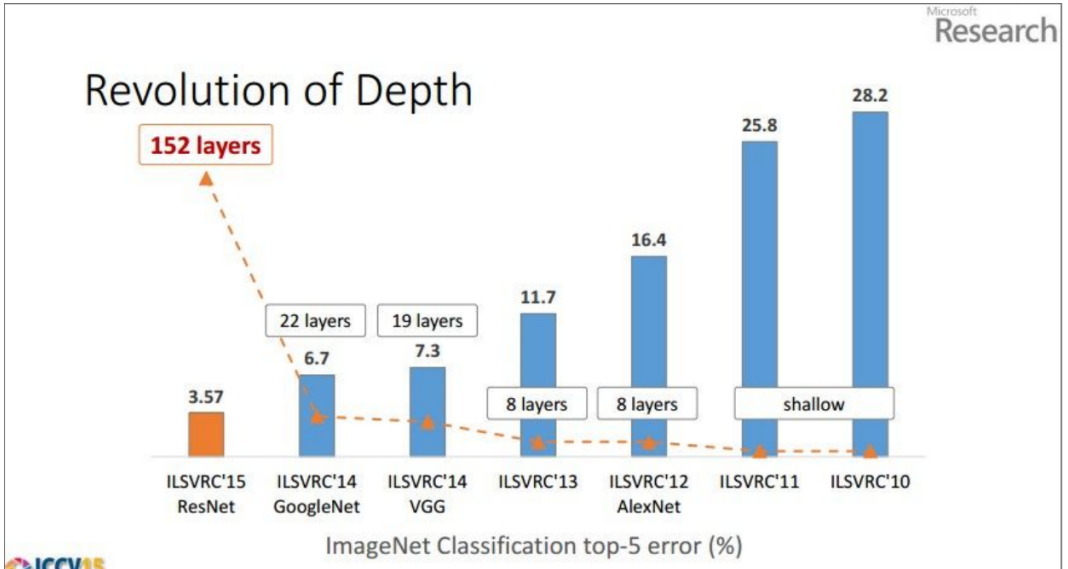
## Case study



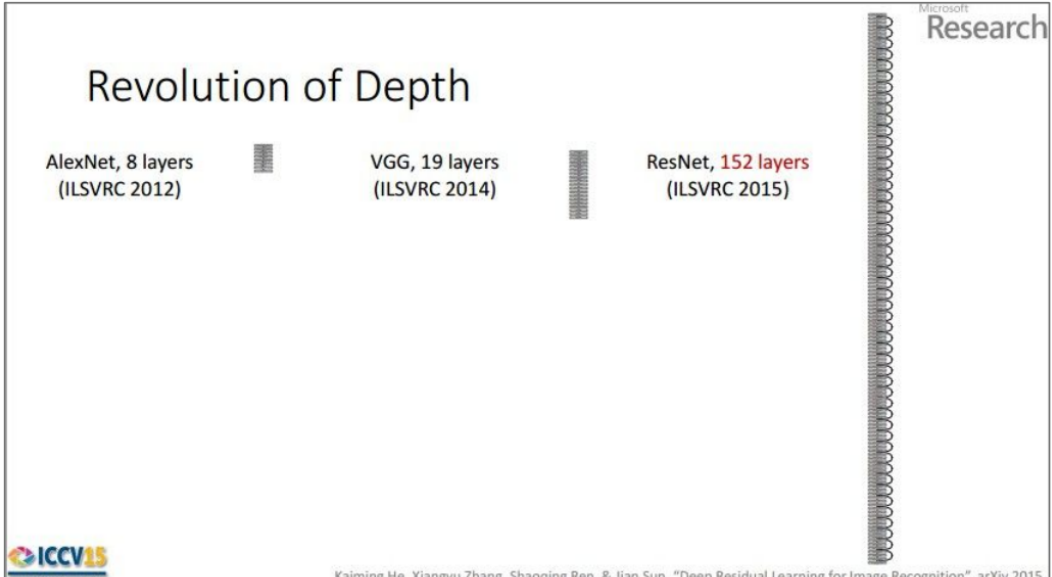
# Basic elements of CNN



# Basic elements of CNN



# Basic elements of CNN



# Contents

Deep Learning for Computer Vision.

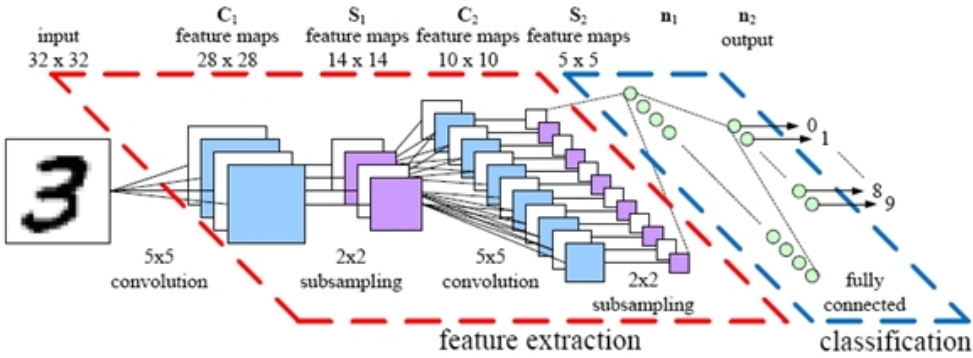
Quick and dirty ML overview.

Basic elements of CNN
- Motivation
- Structure
- Structure [convolution layer]
- Structure [fully-connected layer]
- Case study

Summary

# Summary

## CNN Architecture



# Summary

## CNN Architecture

- INPUT holds the raw pixel values of the image.
- CONV layer computes the output of neurons that are connected to local regions in the input, each computing a dot product between their weights and the region they are connected to in the input volume.
- POOL layer performs a downsampling operation along the spatial dimensions (width, height).
- FC (i.e. fully-connected) layer computes the class scores. As with ordinary Neural Networks and as the name implies, each neuron in this layer is connected to all the numbers in the previous volume.