# Deep Learning for Computer Vision

Andrii Liubonko

Grammarly*

* The opinions expressed in this presentation and on the following slides are solely those of the presenter and not necessarily those of Grammarly

# Logistics

4 units

2 types of homework

- paper review
- mini-project

01 december, 23:59, approve for a paper
09 december, 23:59, paper review (SOFT, PENALTY 30%)
*30 december, 23:59, deadline* *(HARD)*

https://github.com/lyubonko/ucu2020cv

# Overview of the course

**Unit I** (26 Nov, 11.30-14.30)

[T] Intro to Convolution Neural Networks (CNNs)
[P] pytorch

**Unit II** (28 Nov, 15.00-18.00)

[T] CNNs in depth
[P] classification

**Unit III** (10 Dec, 14.00-17.00)

[T] Attention in CV
[P] Transformers

**Unit IV** (11 Dec, 10.00-13.00)

[T] Object Detection
[P] project structure, detection

# Overview of the course

**Unit I** (26 Nov, 11.30-14.30)
[T] Intro to Convolution Neural Networks (CNNs)
[P] pytorch

**Unit II** (28 Nov, 15.00-18.00)
[T] CNNs in depth
[P] simple net, classification

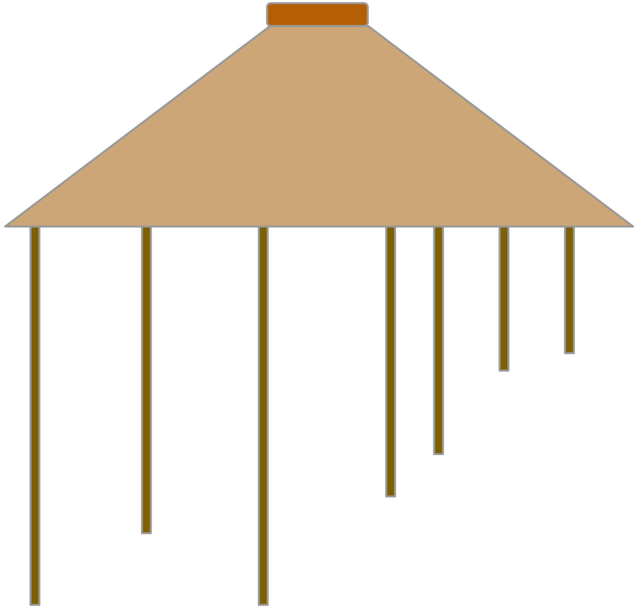
**Unit III** (10 Dec, 14.00-17.00)
[T] Attention in CV
[P] Transformers

**Unit IV** (11 Dec, 10.00-13.00)
[T] Object Detection
[P] project structure, detection

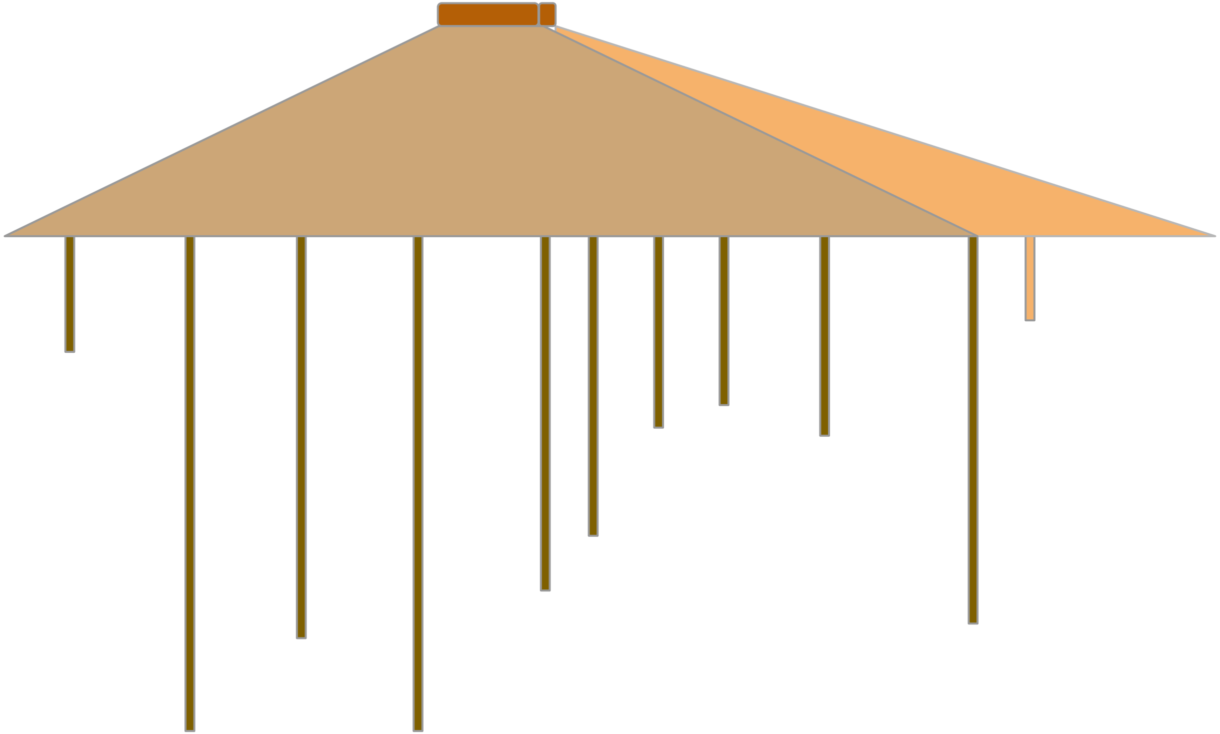
# Goals of the Course

- working knowledge of essential elements/blocks of Convolutional Neural Networks (CNNs)
- modern CNNs architectures
- attention in Computer Vision (current trend)
- get deeper with one particular problem (Object Detection)

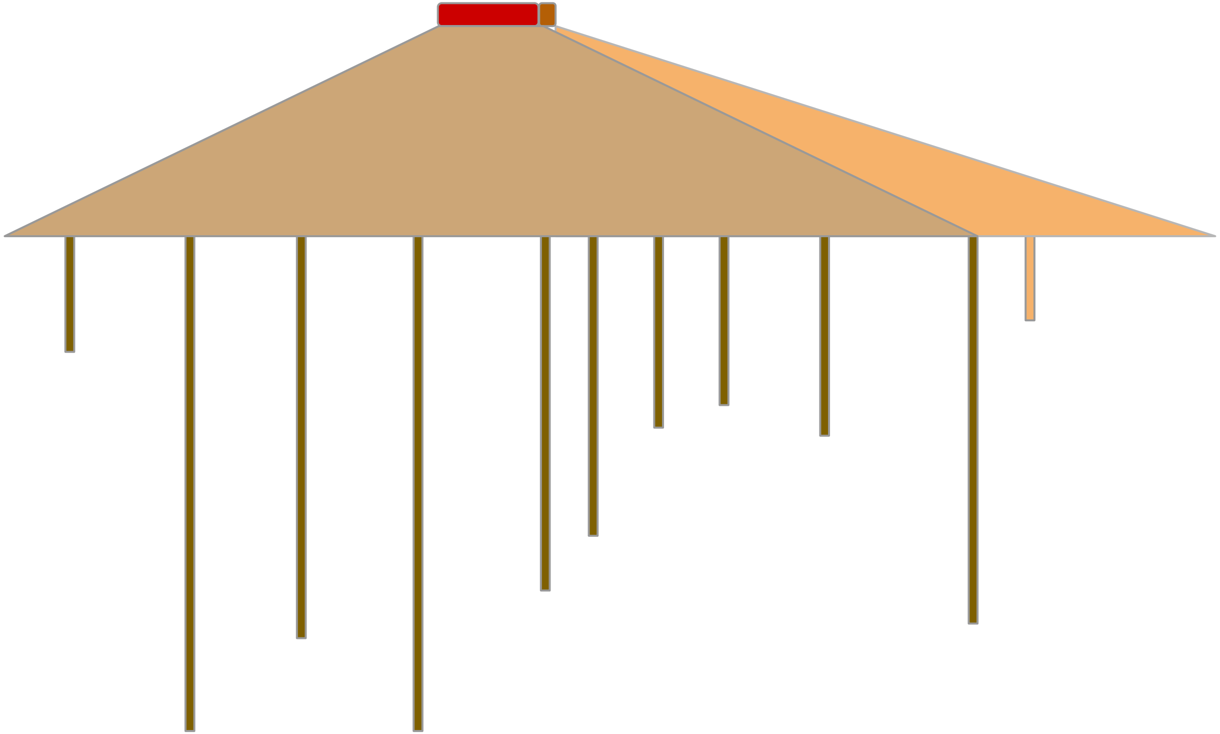
# Goals of the Course



# Goals of the Course

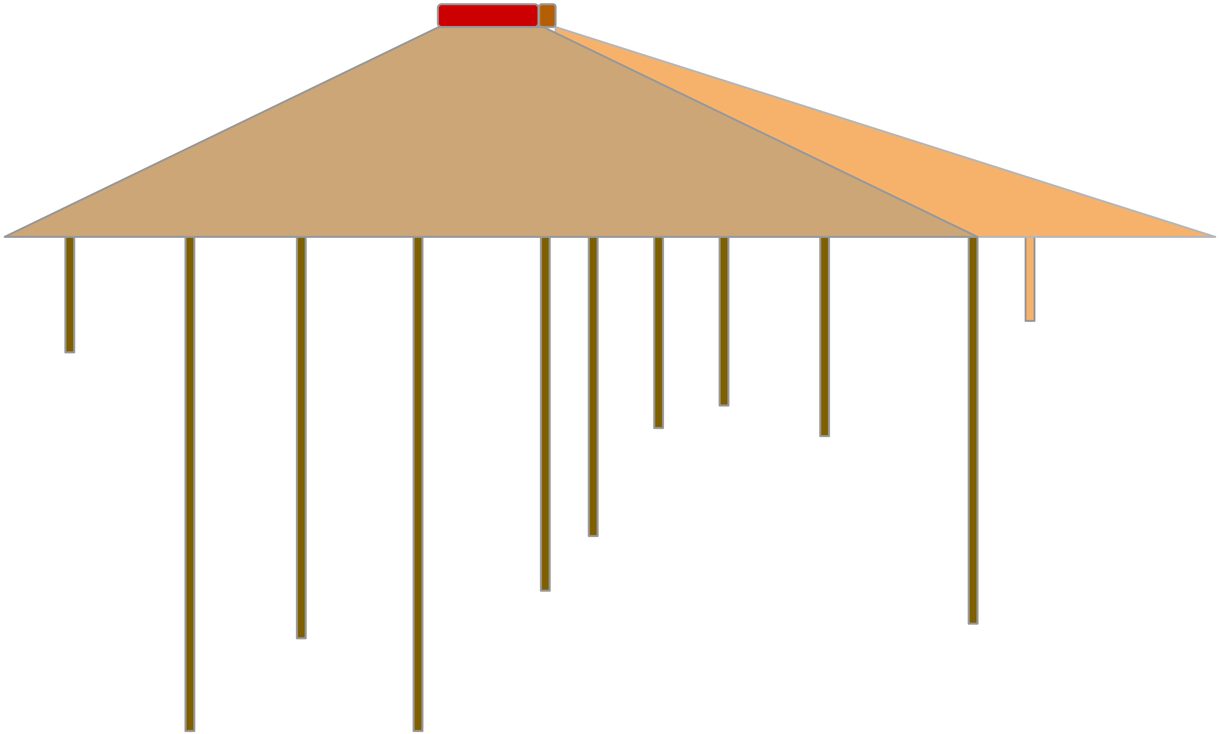


# Goals of the Course



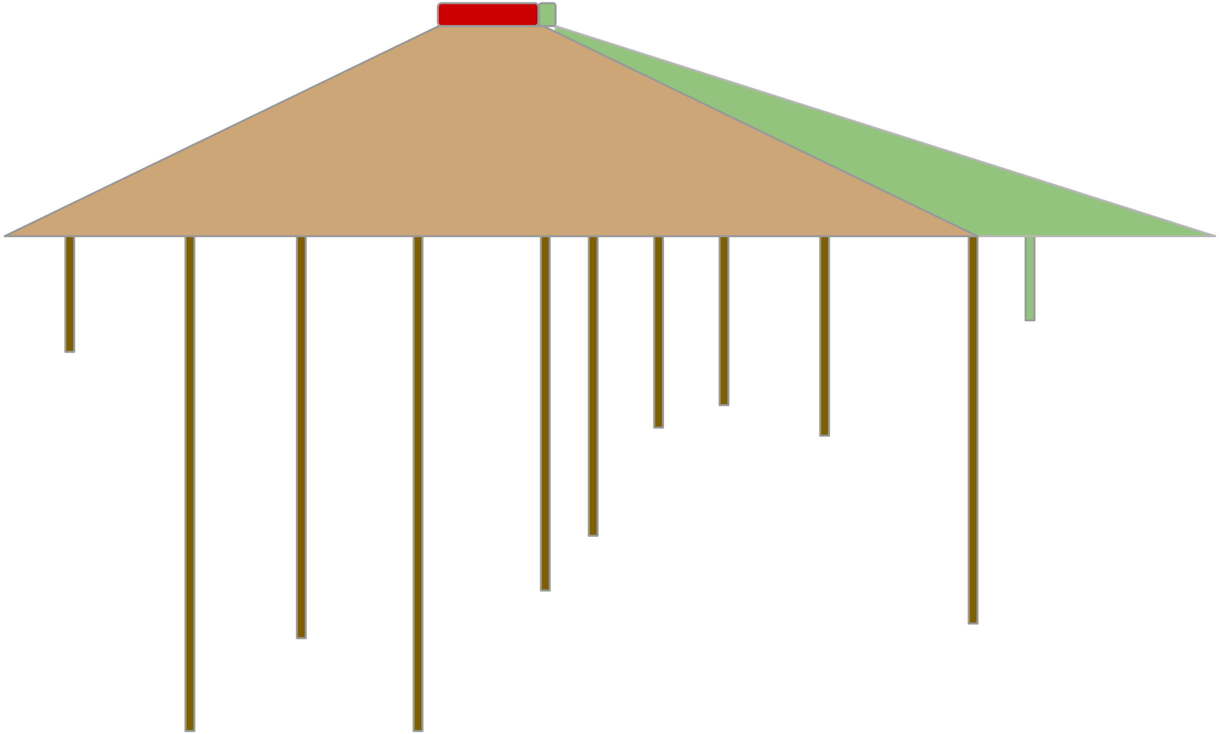
- essential CNNs elements/blocks

# Goals of the Course



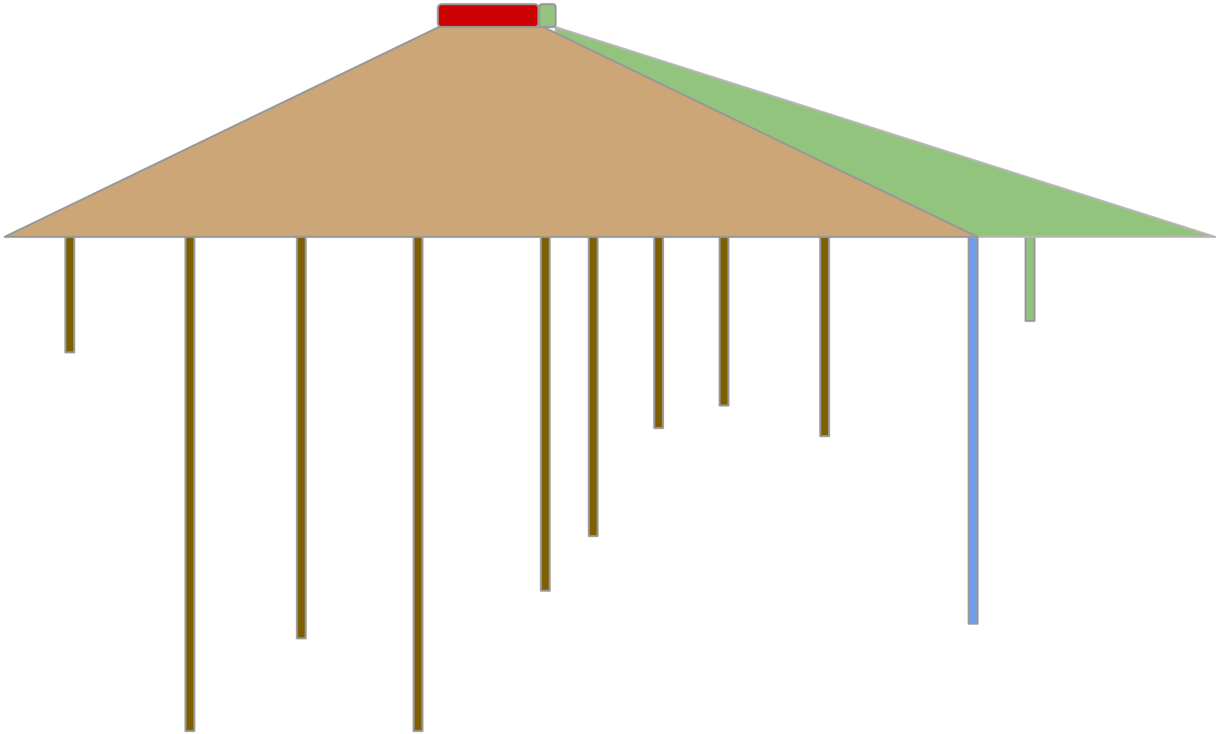
- essential CNNs elements/blocks
- modern CNNs architectures

# Goals of the Course



- essential CNNs elements/blocks
- modern CNNs architectures
- attention in CV

# Goals of the Course



- essential CNNs elements/blocks
- modern CNNs architectures
- attention in CV
- Object Detection

# Content of today lecture

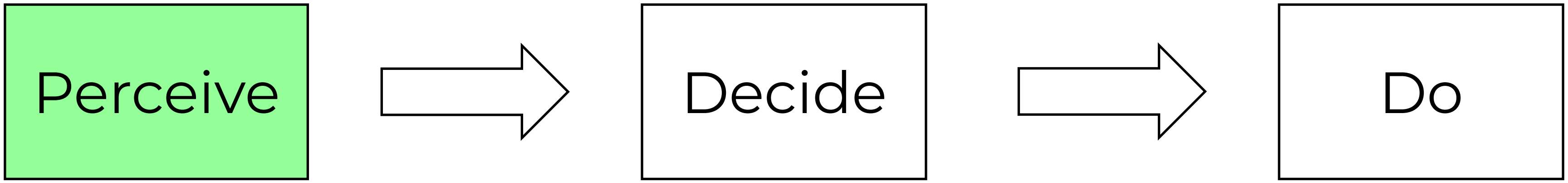
- **Intro**

- DL review
  - neural networks
  - training & testing
  - supervised, semi-supervised, self-training

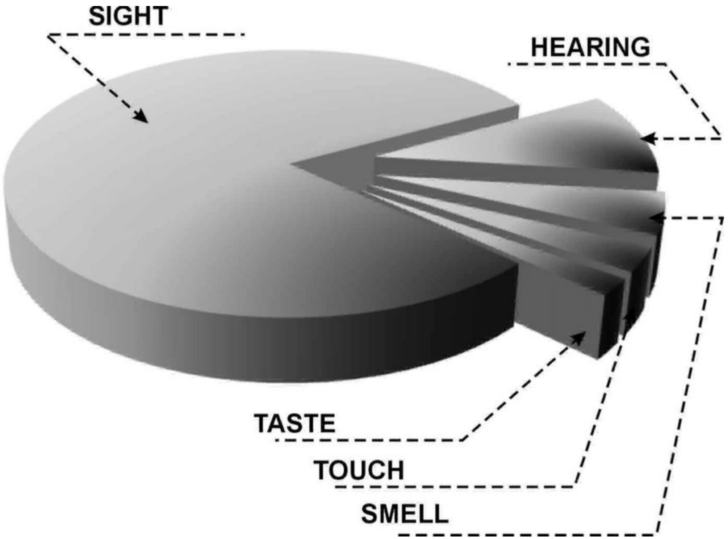
- Main components of CNNs (motivation and details)
  - convolutional layer
  - pooling layer

- Datasets
  - ImageNet

# Intro



# Intro

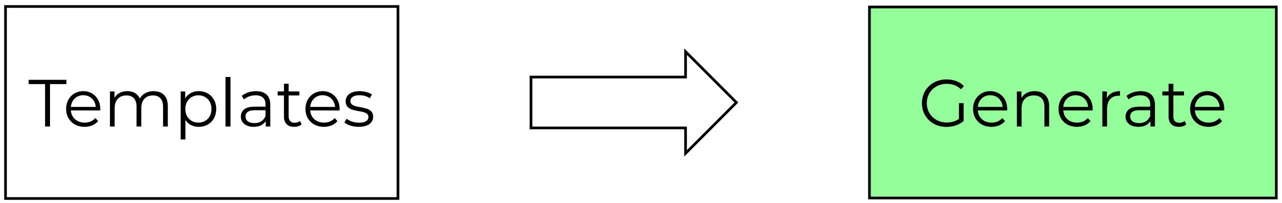



*The goal of* ***computer vision*** *is to extract useful information from visual input (images, video)*

# Intro

- indoor/outdoor? [image classification]
- Where are the objects? [object detection]
- How far is the object ? [depth estimation ]
- What people are doing? [activity recognition]
- Is the state of the environment normal? [anomaly detection ]
- ...

# Intro



# Intro

Generative adversarial network (GAN)

2011.09055

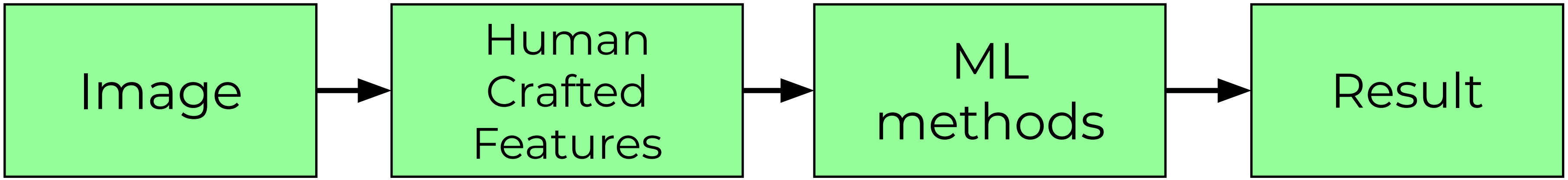
# Intro

what humans see

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | 3 | 2 | 5 | 4 | 7 | 6 | 9 | 8 |
| 3 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 2 | 1 | 0 | 3 | 2 | 5 | 4 | 7 | 6 |
| 5 | 2 | 3 | 0 | 1 | 2 | 3 | 4 | 5 |
| 4 | 3 | 2 | 1 | 0 | 3 | 2 | 5 | 4 |
| 7 | 4 | 5 | 2 | 3 | 0 | 1 | 2 | 3 |
| 6 | 5 | 4 | 3 | 2 | 1 | 0 | 3 | 2 |
| 9 | 6 | 7 | 4 | 5 | 2 | 3 | 0 | 1 |
| 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

what computers see

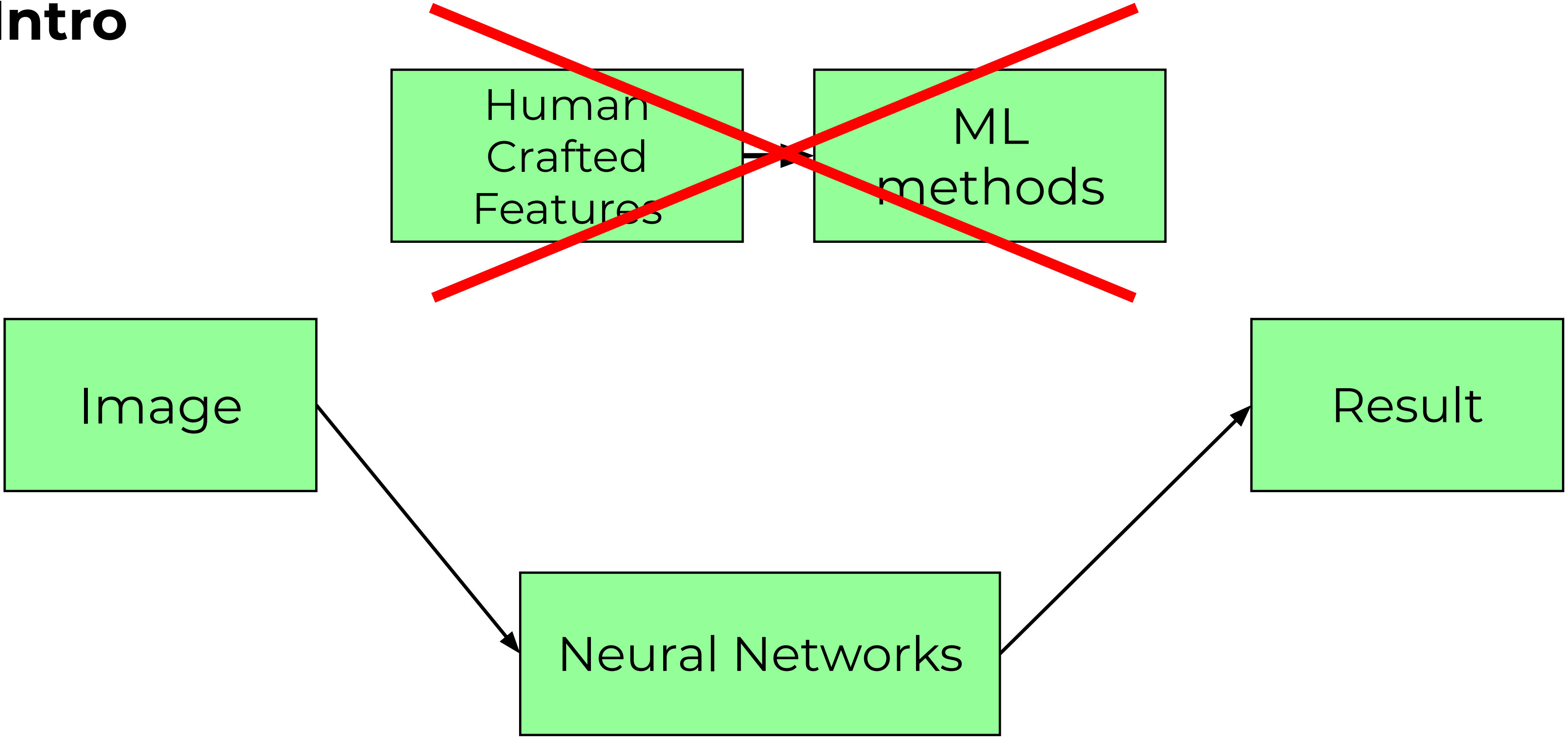
# Intro



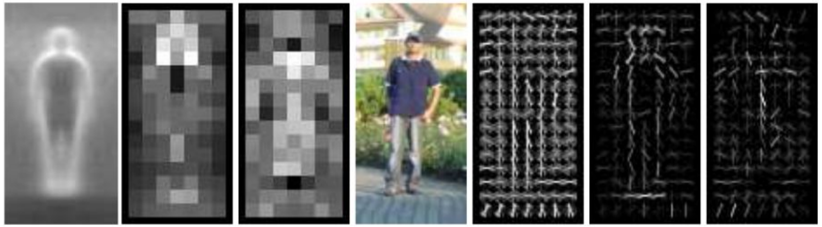

Descriptors:
SIFT, SURF, etc.

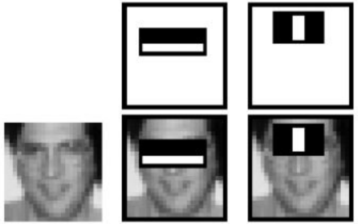
Linear, SVM,
Trees

# Intro

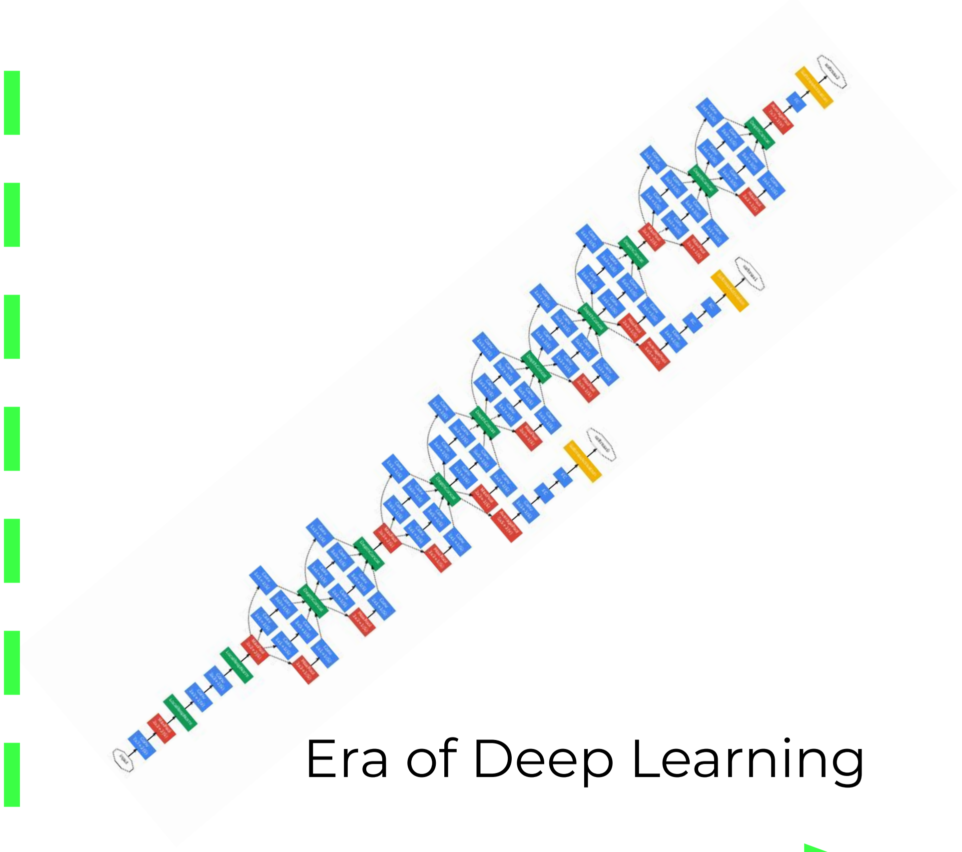

Human Crafted Features
ML methods
Image
Result
Neural Networks

# Intro





Era of Human-Crafter Features



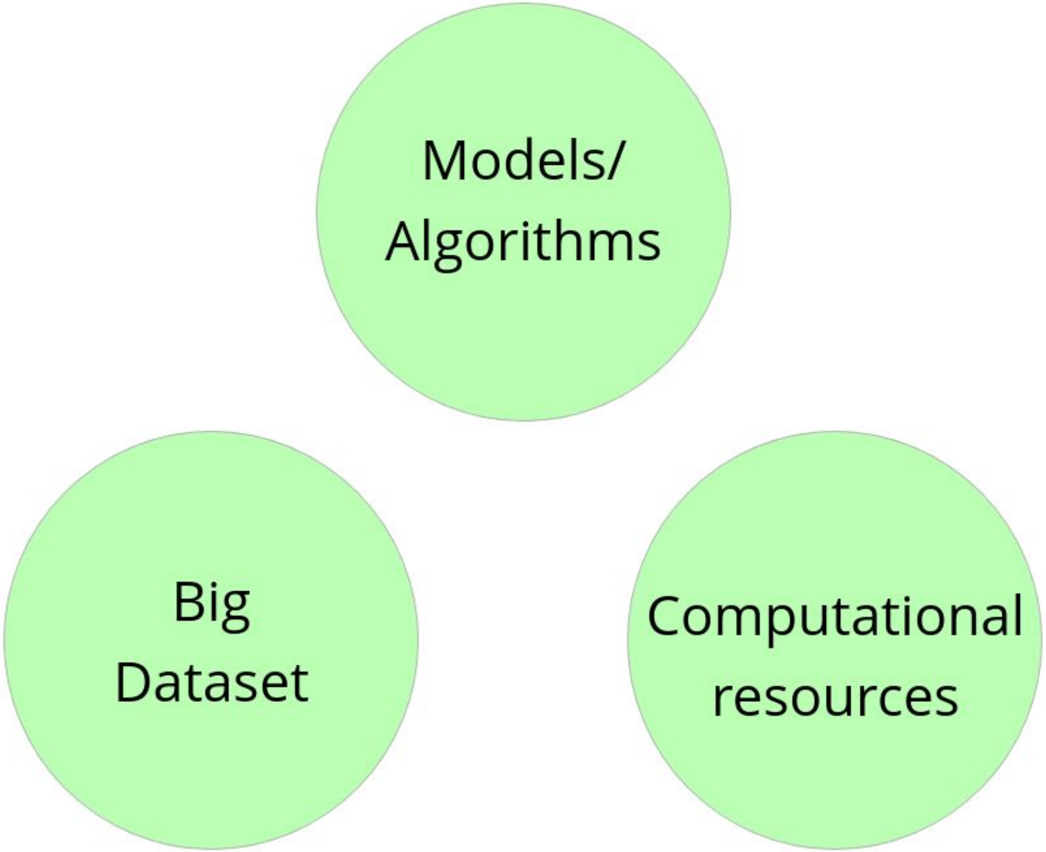
Era of Deep Learning

1986
BackProp

1998
LeNet

2012
AlexNet

# Intro



# Content

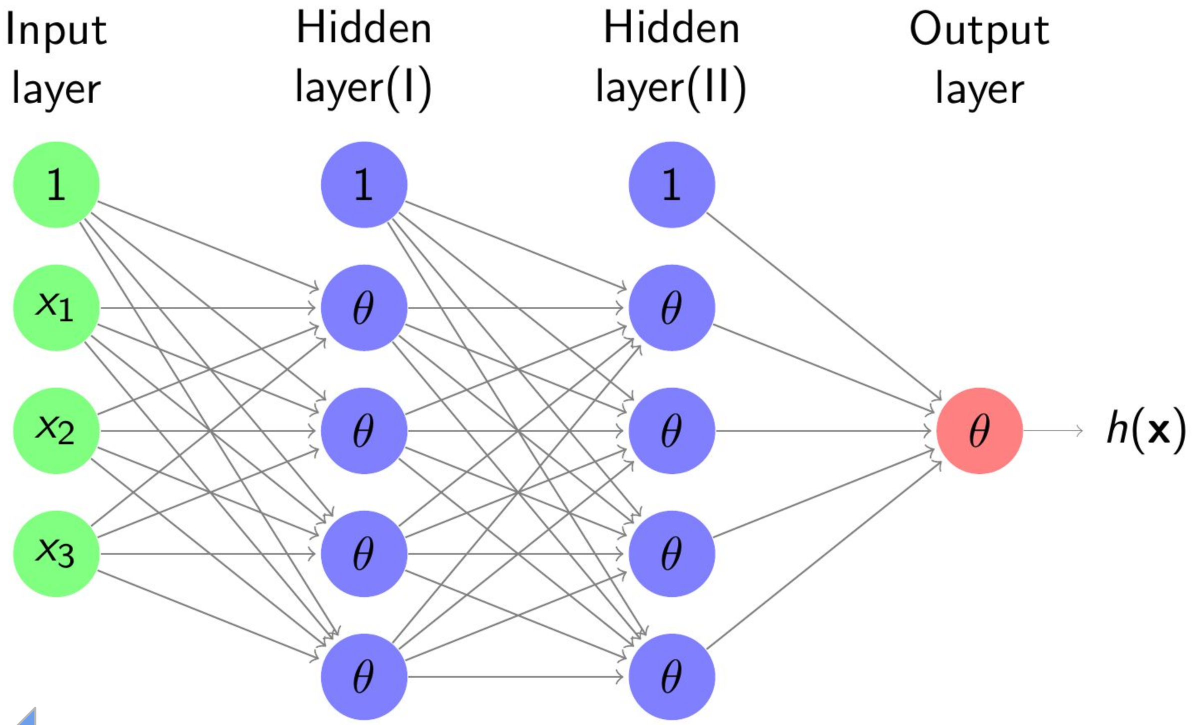
- Intro

- **DL review**
  - neural networks
  - training & testing
  - loss functions
  - supervised, semi-supervised

- Main components of CNNs (motivation and details)
  - convolutional layer
  - pooling layer

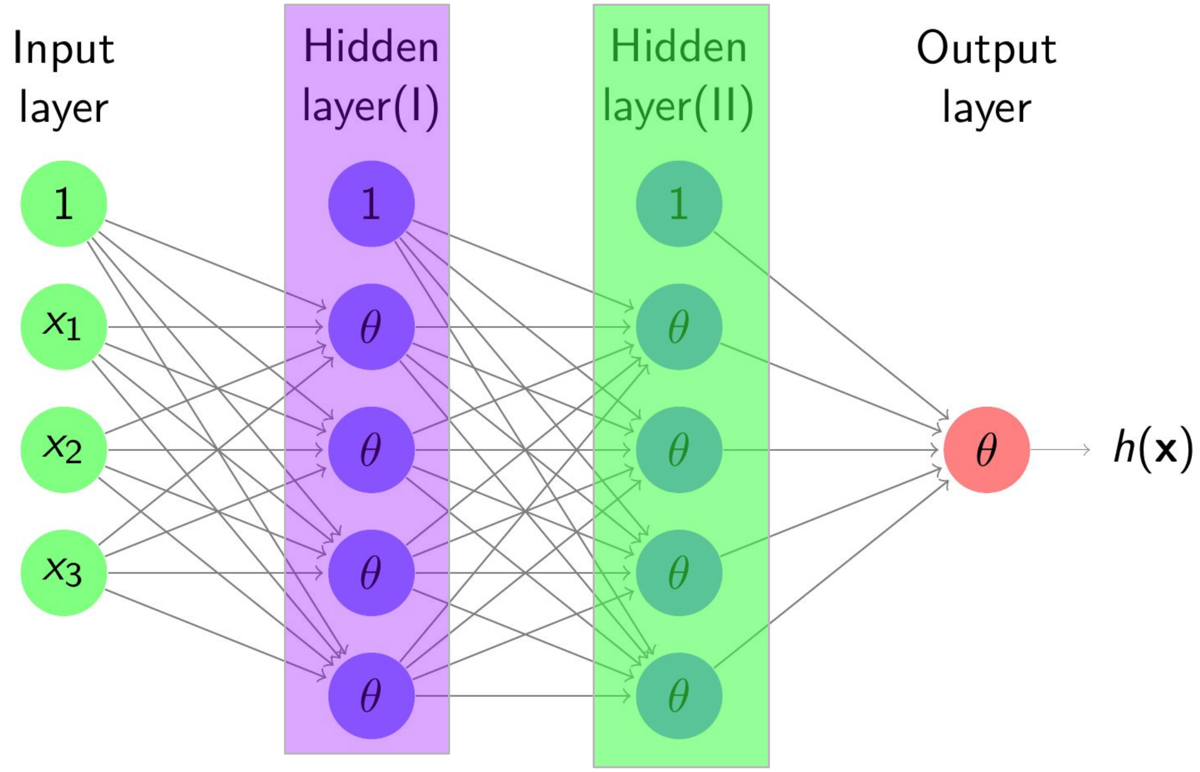
- Datasets & Metrics
  - ImageNet

# Neural Networks




Back-propagate error signal to get derivatives for learning

# Neural Networks

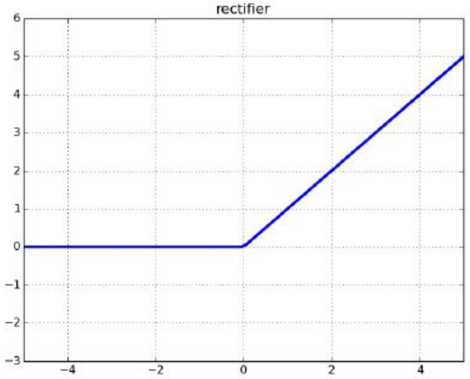



$$\{f(\boldsymbol{x};\boldsymbol{\theta}) = \mathbf{W}_L \boldsymbol{\sigma}_L(\mathbf{W}_{L-1} \cdots \boldsymbol{\sigma}_2(\mathbf{W}_2 \boldsymbol{\sigma}_1(\mathbf{W}_1 \boldsymbol{x}))) \mid \boldsymbol{\theta} = \{\mathbf{W}_1, \ldots, \mathbf{W}_L\}\}$$

# Neural Networks

parameters in NN:

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




activation:

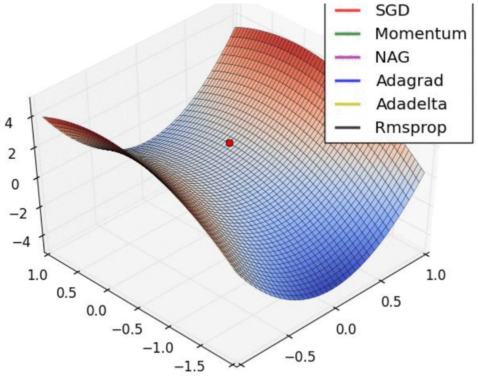
$$x_j^{(l)} = \sigma(s_j^{(l)}) = \sigma\Big(\sum_{i=0}^{d^{(l-1)}} W_l^{ij} x_i^{(l-1)}\Big)$$

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

# Neural Networks

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

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



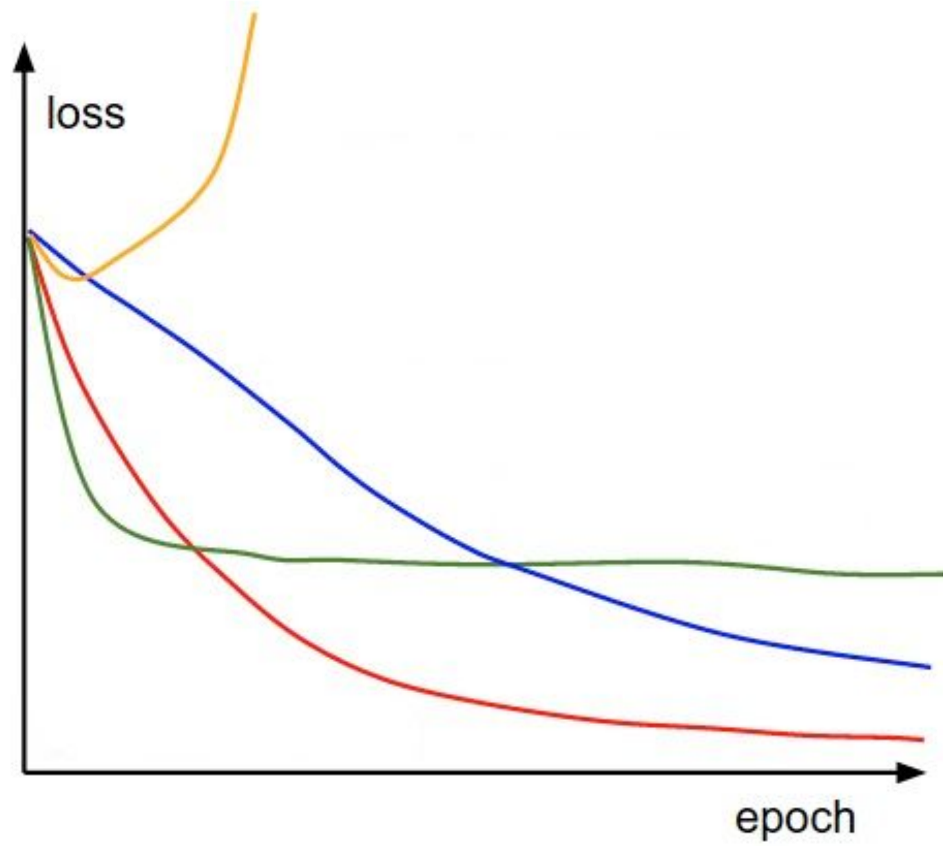

**Gradient Descent (GD)** minimizes:

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

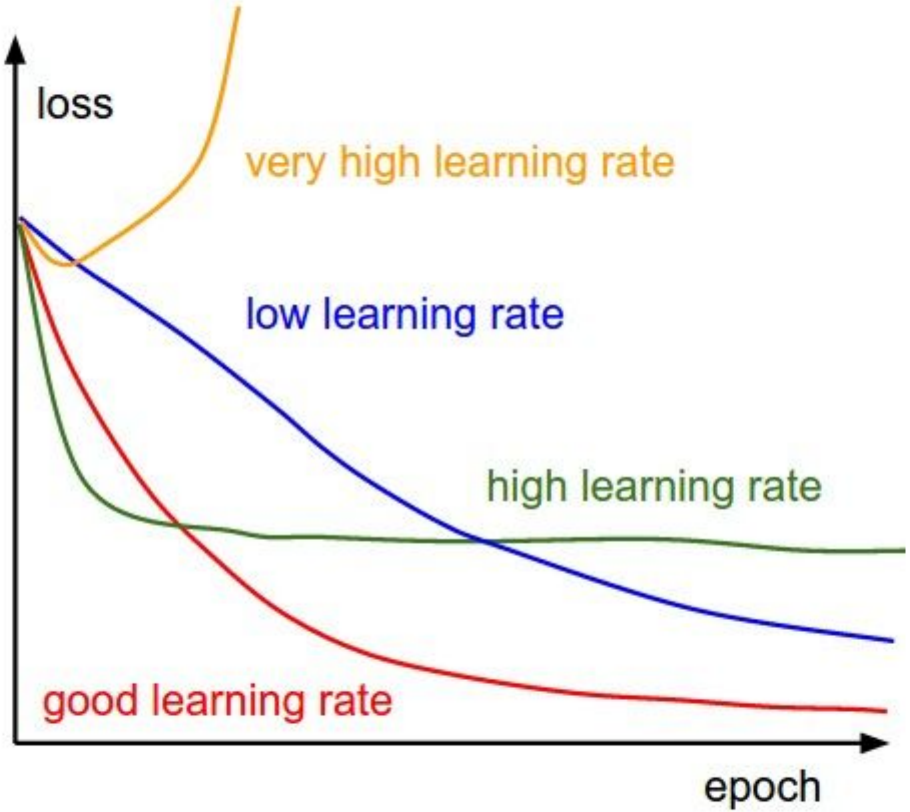
by iterative steps along $-\nabla L_{train}$:

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

# Neural Networks

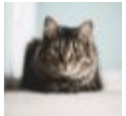


# Neural Networks



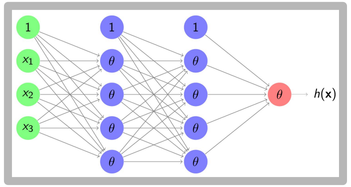
# Neural Networks (supervised way)

$\{$  , plane $\}$

$\{$  , cat $\}$

...

$F(\mathbf{x_n})$




$e(F(\mathbf{x_n}), y_n)$

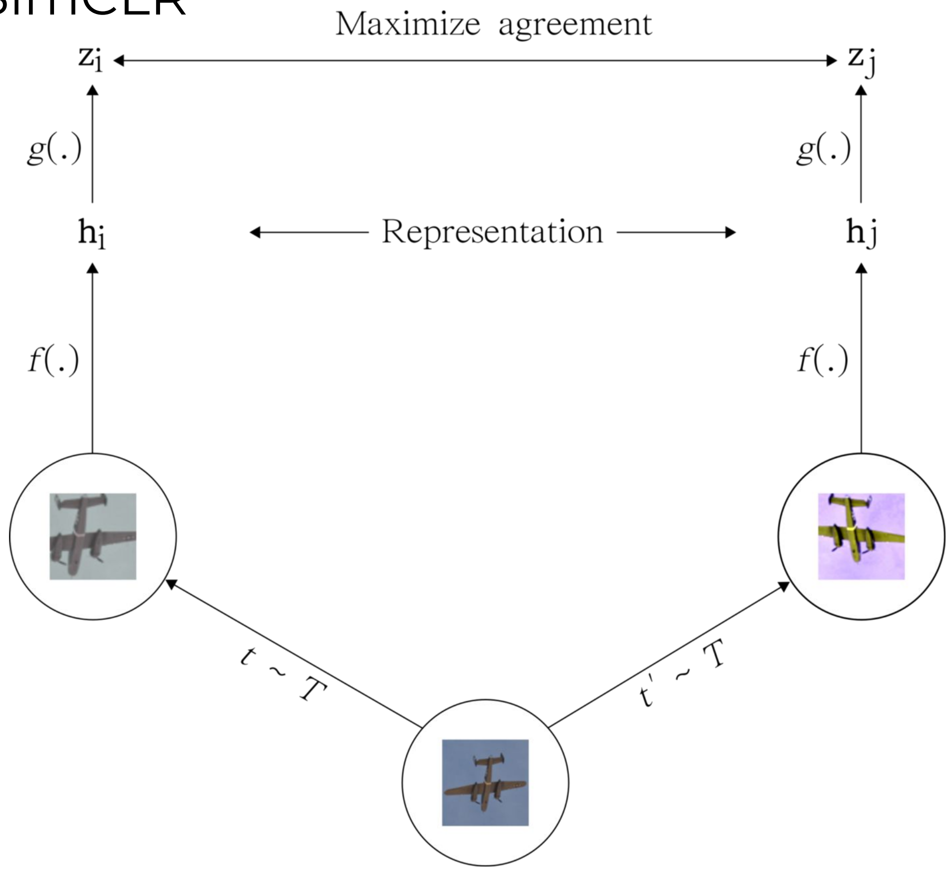
$e(F(\mathbf{x_n}), y_n)$

...

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

# Neural Networks (self-supervised way)

## SimCLR




$$\ell_{i,j} = -\log \frac{\exp(\mathrm{sim}(\boldsymbol{z}_i, \boldsymbol{z}_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(\mathrm{sim}(\boldsymbol{z}_i, \boldsymbol{z}_k)/\tau)}$$

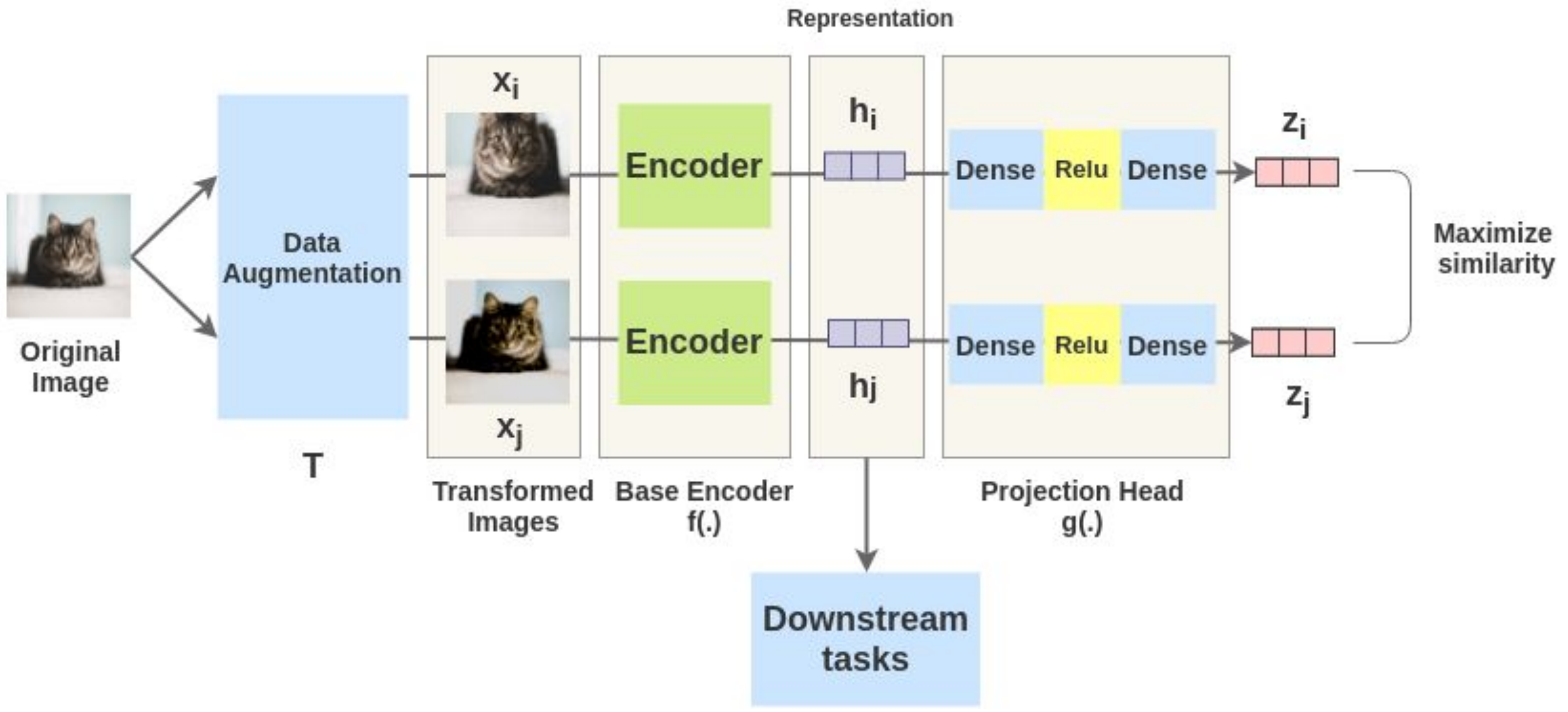
$$\boldsymbol{z}_i = g(\boldsymbol{h}_i) = W^{(2)}\sigma(W^{(1)}\boldsymbol{h}_i)$$

$$\boldsymbol{h}_i = f(\tilde{\boldsymbol{x}}_i) = \mathrm{ResNet}(\tilde{\boldsymbol{x}}_i)$$

2002.05709

# Self-supervised

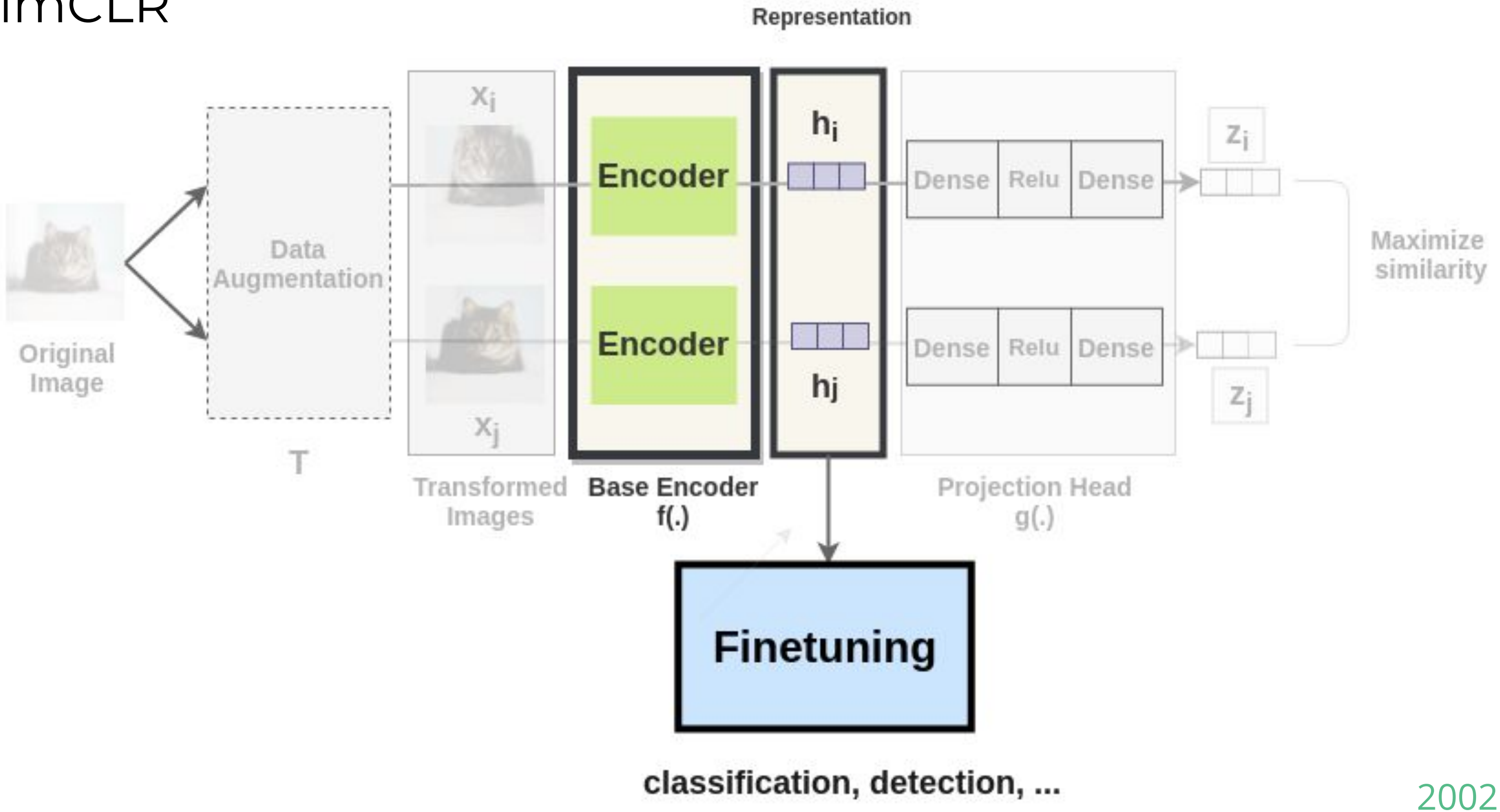
## SimCLR




2002.05709

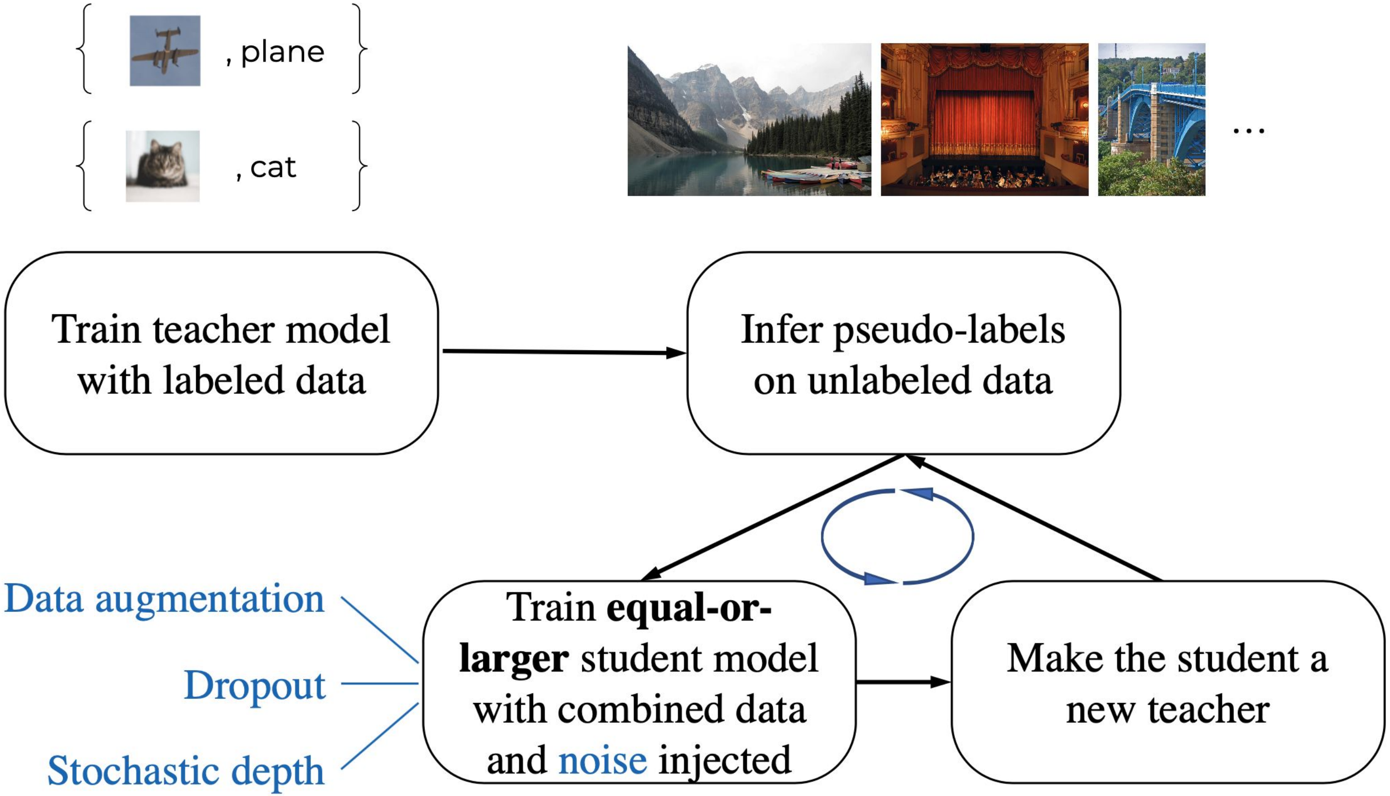
# Self-supervised

SimCLR




2002.05709

# Neural Networks (semi-supervised)




1911.04252

# Content

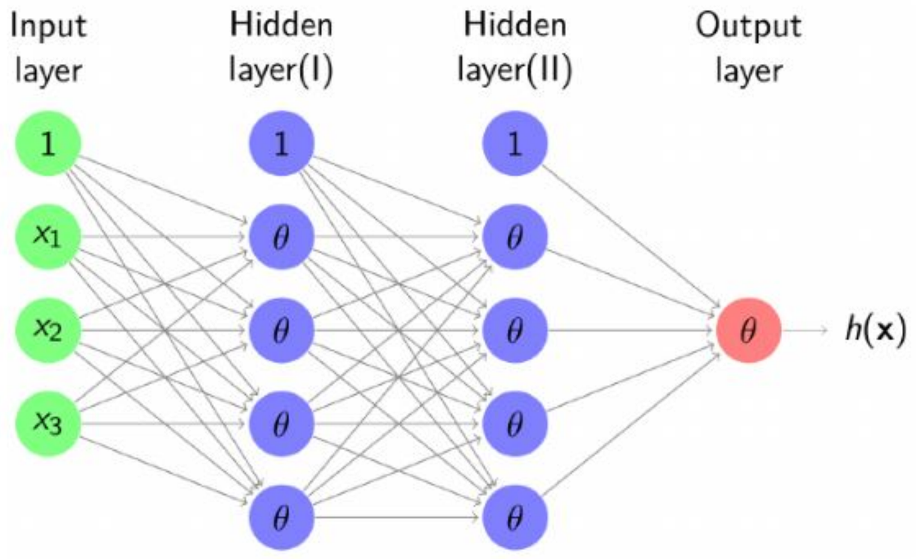
- Intro

- DL review
  - neural networks
  - training & testing
  - supervised, semi-supervised

- **Main components of CNNs (motivation and details)**
  - convolutional layer
  - pooling layer

- Datasets & Metrics
  - ImageNet

# Intro to CNN




Input layer
$3 \times 256 \times 256 = 200K$

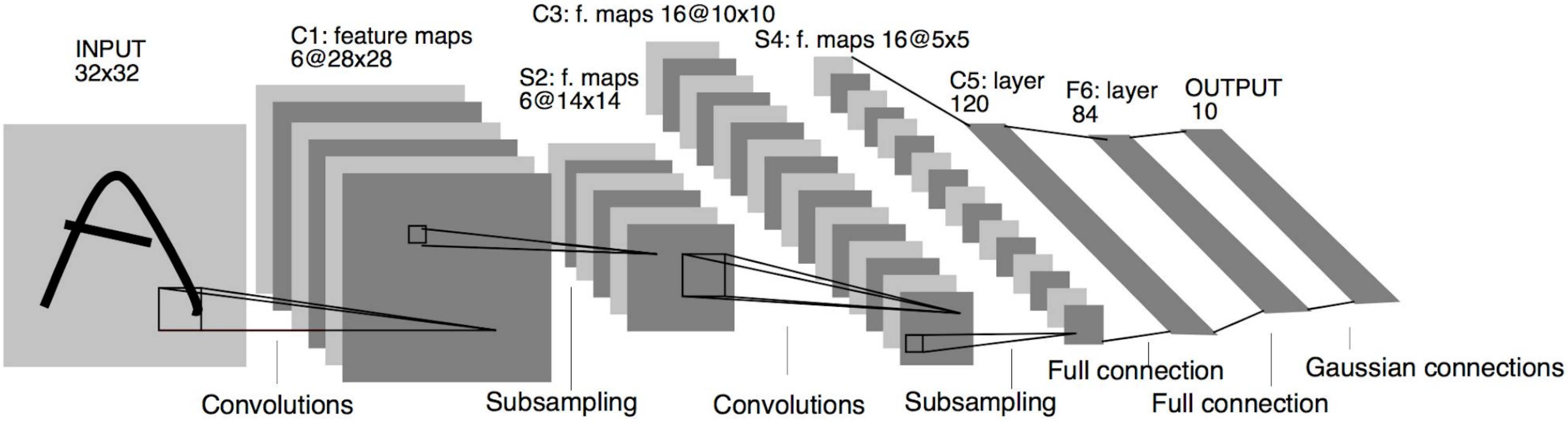
Weights ($\omega_0$):
$200K \times$(size of next layer)

=> millions of parameters
(for just one layer)

Need for alternative architecture

# Intro to CNN

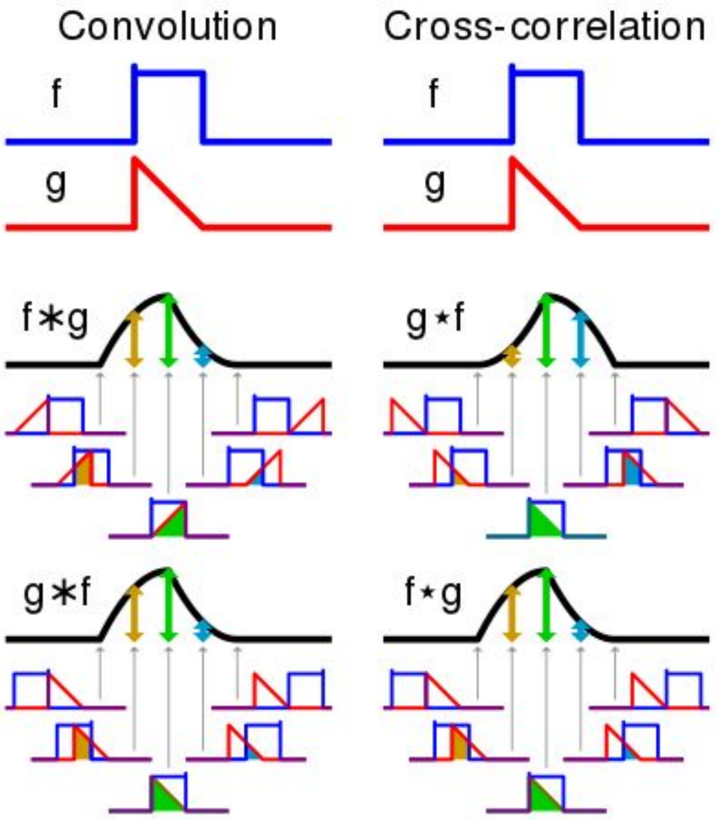
LeNet-5 [1998, paper by LeCun et al.]



# Intro to CNN

- 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.

# Intro to CNN

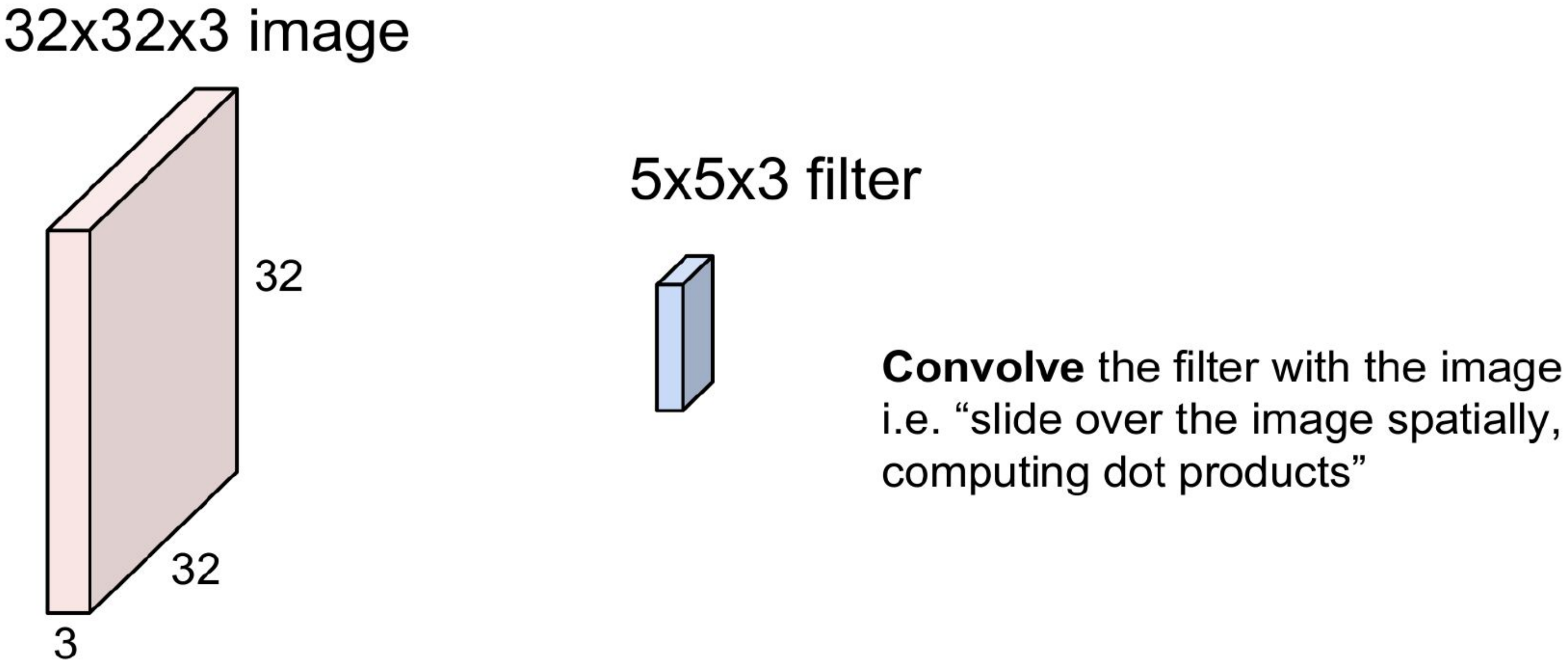



a function derived from two given functions by integration that expresses how the shape of one is modified by the other

$$(f * g)(t) \triangleq \int_{-\infty}^{\infty} f(\tau) g(t-\tau)\, d\tau.$$

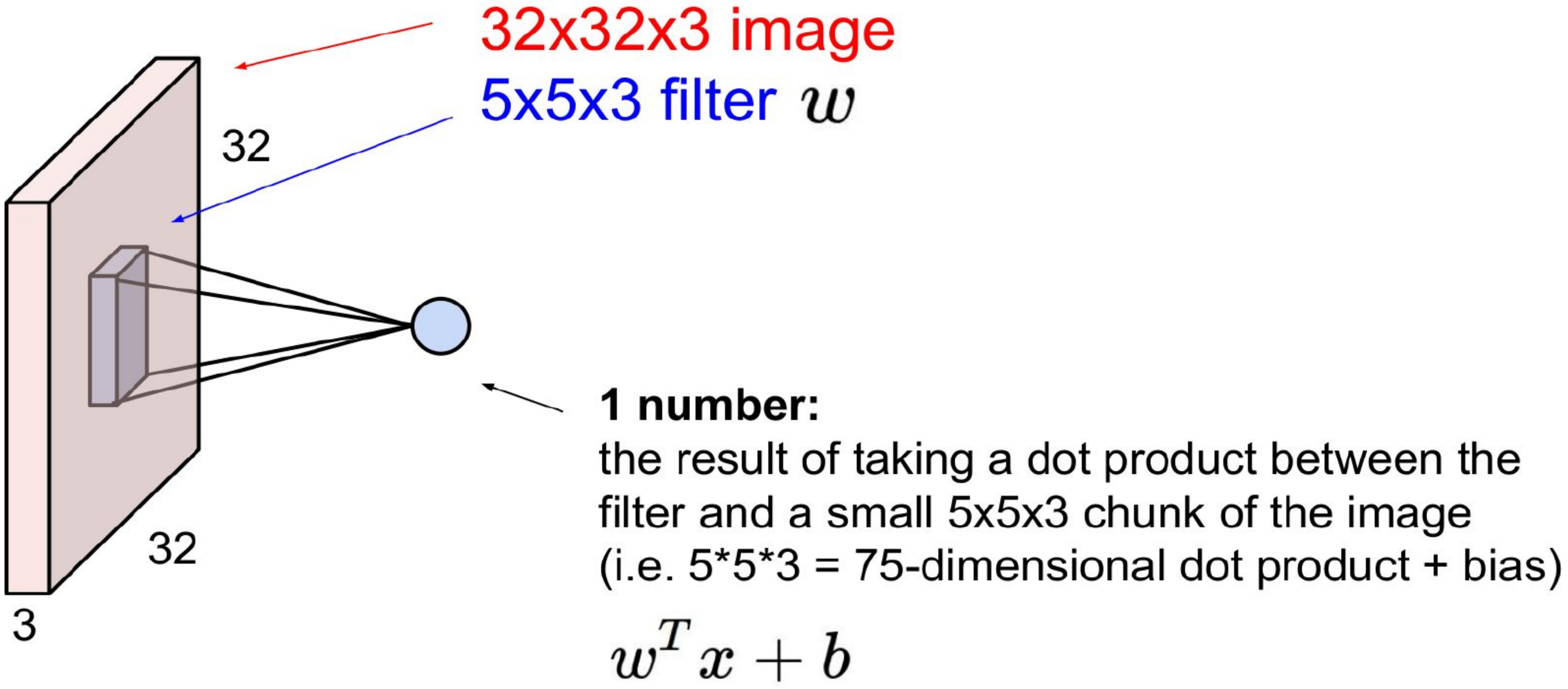
https://en.wikipedia.org/wiki/Convolution

# Intro to CNN




*images from http://cs231n.stanford.edu/*

# Intro to CNN

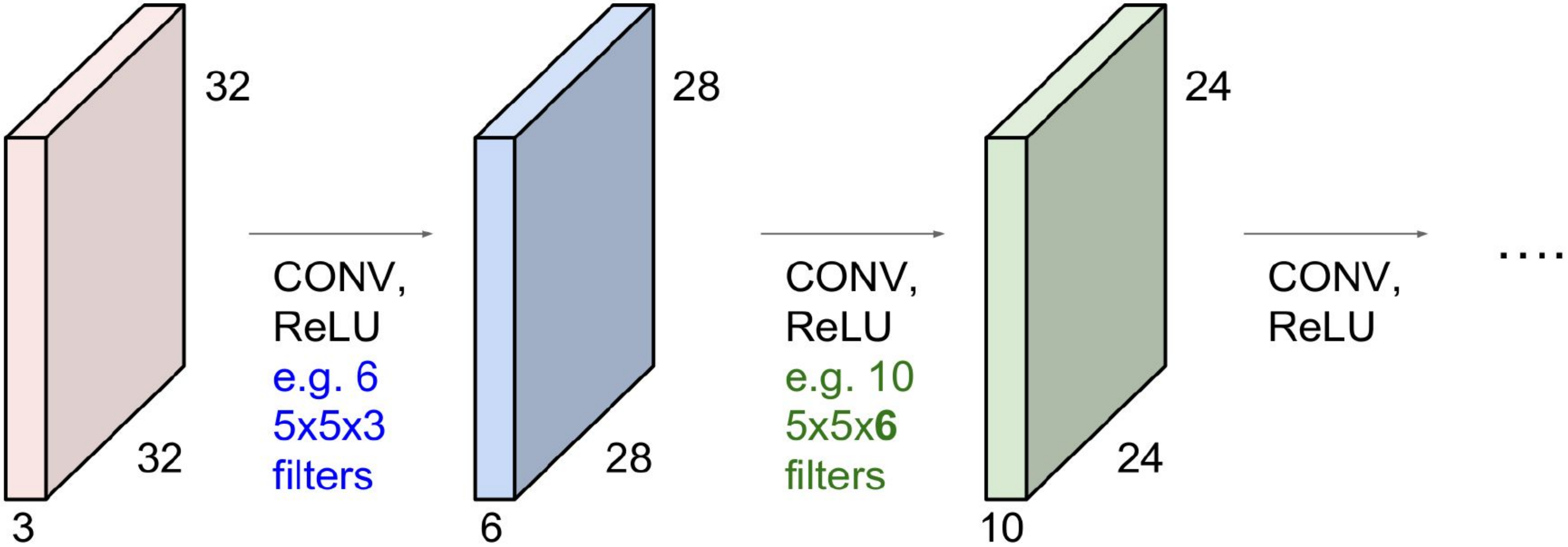



*images from http://cs231n.stanford.edu/*

# Intro to CNN

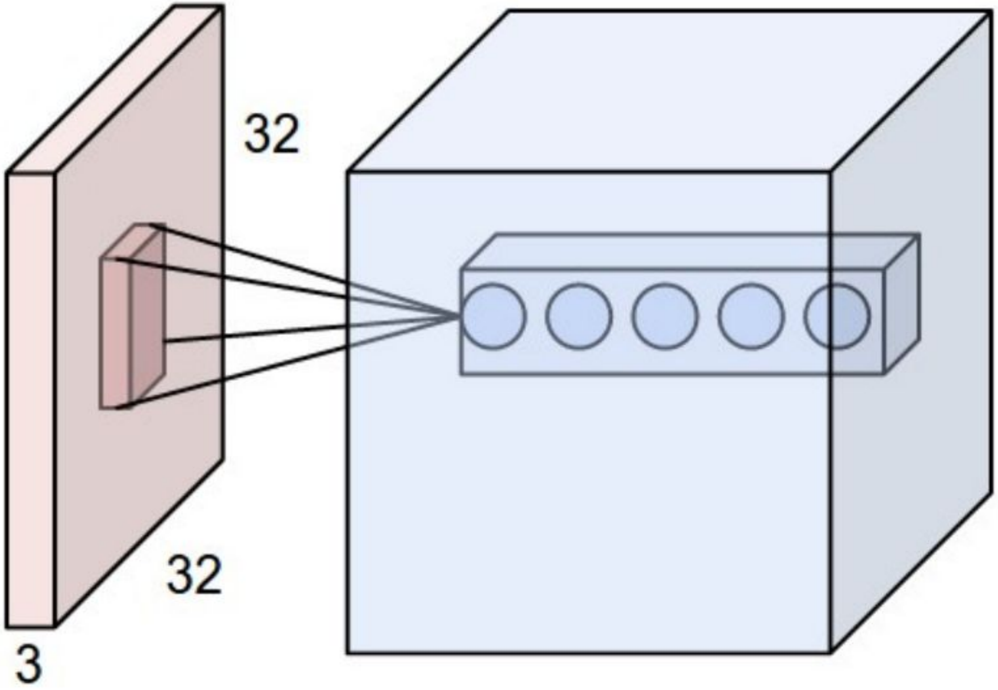

*images from http://cs231n.stanford.edu/*

# Intro to CNN



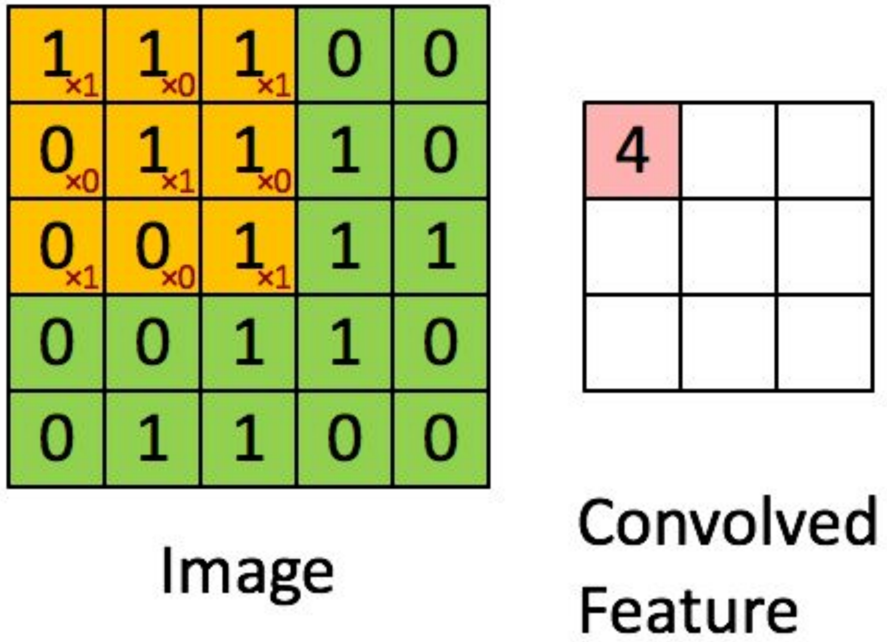

*images from http://cs231n.stanford.edu/*

# Intro to CNN



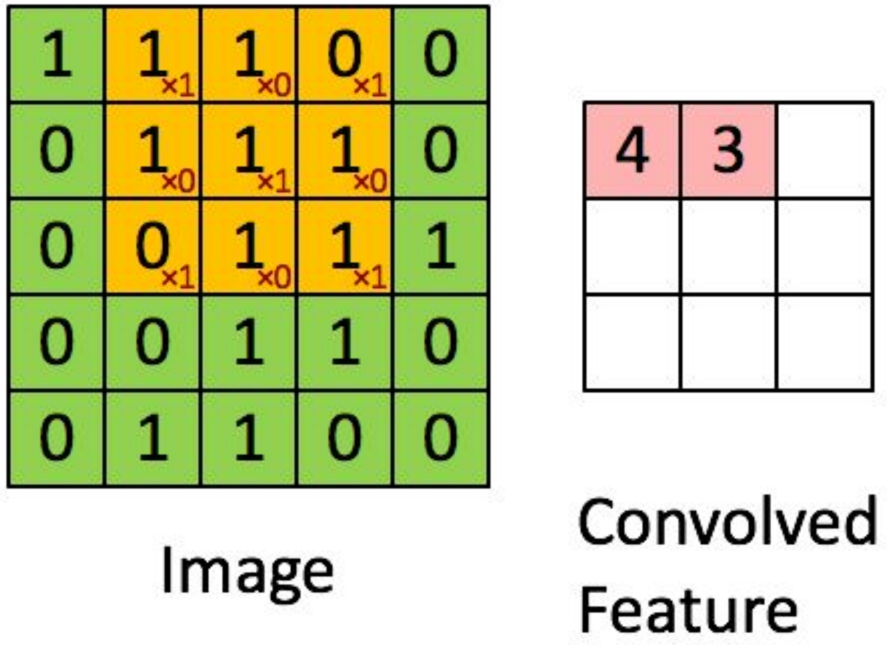

images from http://cs231n.stanford.edu/

# Intro to CNN



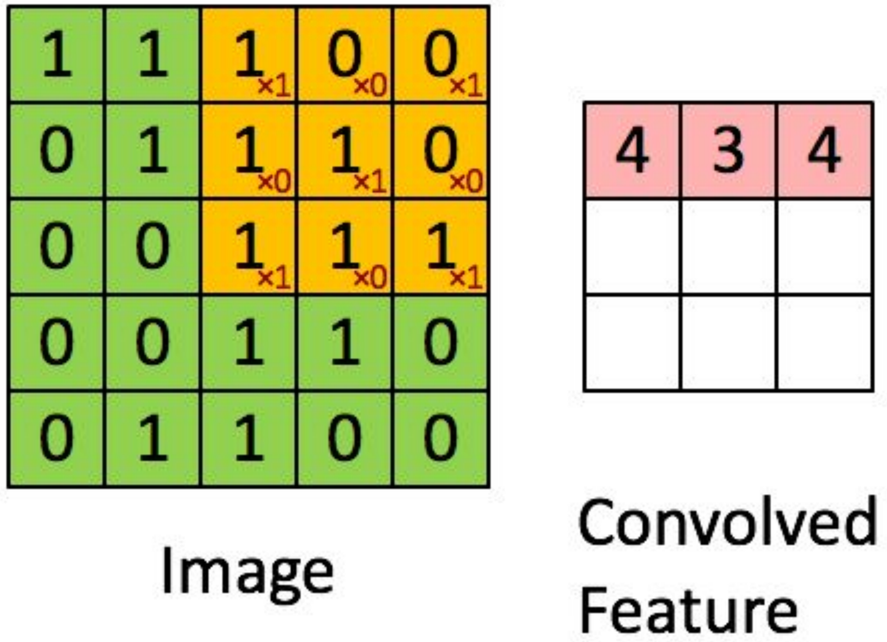

*images from http://cs231n.stanford.edu/*

# Intro to CNN



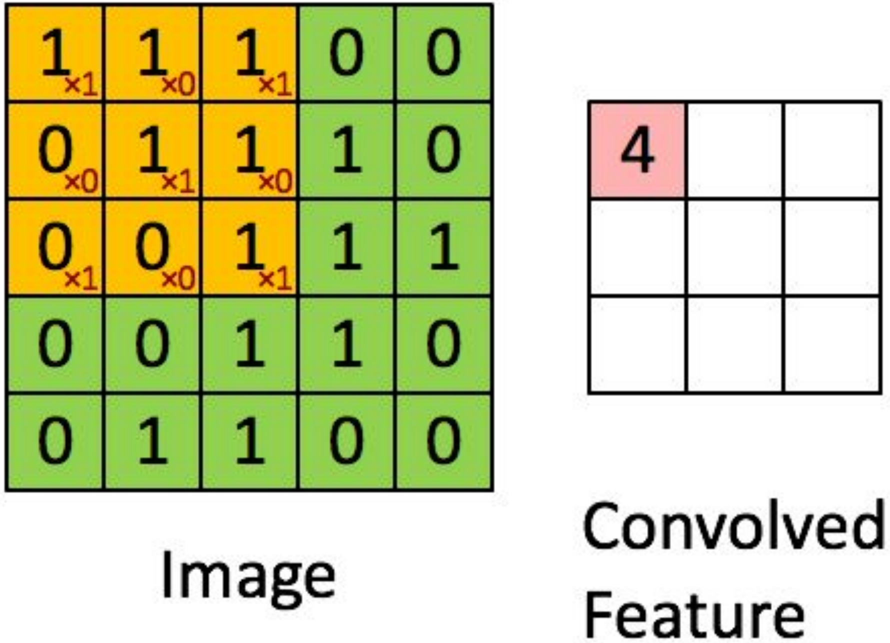

*images from http://cs231n.stanford.edu/*

# Intro to CNN




*images from http://cs231n.stanford.edu/*

# Intro to CNN


1 ×1 | 1 ×0 | 1 ×1 | 0 | 0
0 ×0 | 1 ×1 | 1 ×0 | 1 | 0
0 ×1 | 0 ×0 | 1 ×1 | 1 | 1
0 | 0 | 1 | 1 | 0
0 | 1 | 1 | 0 | 0
4
Image
Convolved Feature

# Intro to CNN

We can use one single convolutional layer to modify a certain image

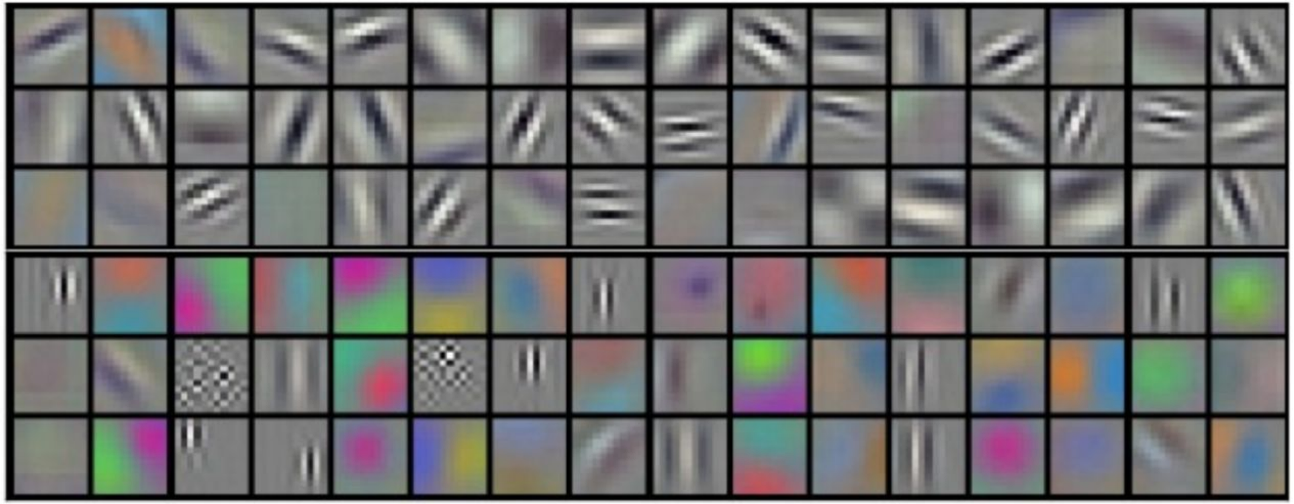

$$\begin{bmatrix} 1. & 1. & 1. \\ 1. & 1. & 1. \\ 1. & 1. & 1. \end{bmatrix}$$


$$\begin{bmatrix} 1. & 2. & 1. \\ 0. & 0. & 0. \\ -1. & -2. & -1. \end{bmatrix}$$

$$\begin{bmatrix} 0. & -1. & 0. \\ -1. & 5. & -1. \\ 0. & -1. & 0. \end{bmatrix}$$

# Intro to CNN

In training, we don't specify kernels.
We learn kernels!



*images from http://cs231n.stanford.edu/*

# Intro to 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.

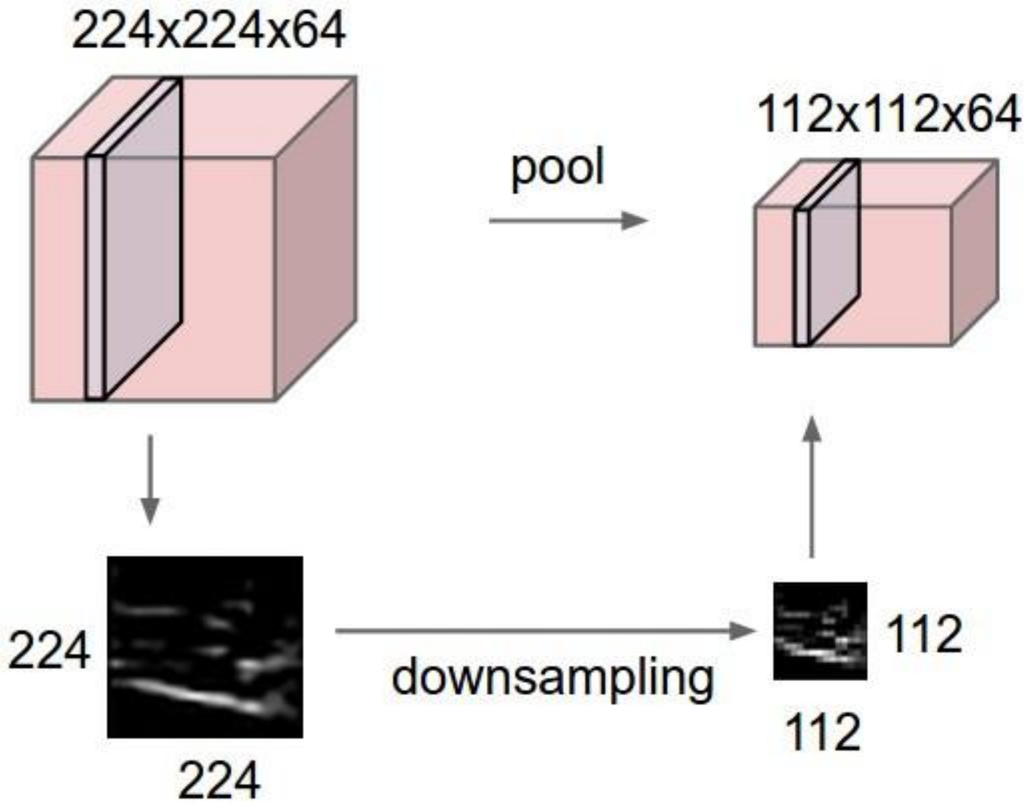
# Intro to CNN

- Convolution leverages four ideas that can help ML systems:
  - Sparse interactions
  - Parameter sharing
  - Equivariant representations $f(g(\mathbf{x})) = g(f(\mathbf{x}))$
  - Ability to work with inputs of variable size

# Intro to CNN

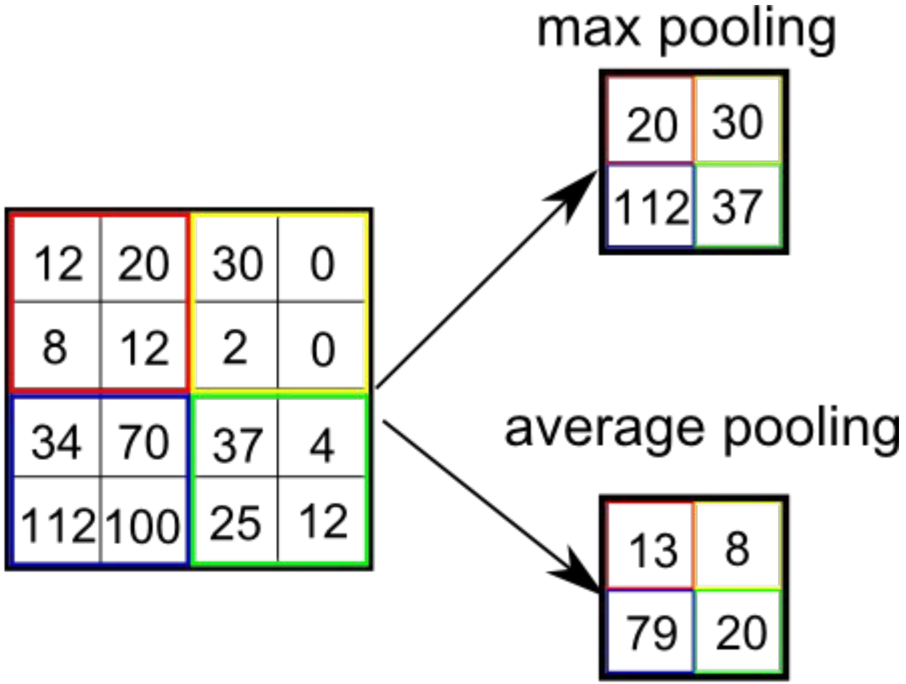
- INPUT holds the raw pixel values of the image.
- CONV layer computes the output of neurons that are connected to local regions in the input, each computing a dot product between their weights and the region they are connected to in the input volume.
- POOL layer performs a downsampling operation along 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.

# Intro to CNN




*images from http://cs231n.stanford.edu/*

# Intro to CNN




*images from http://cs231n.stanford.edu/*

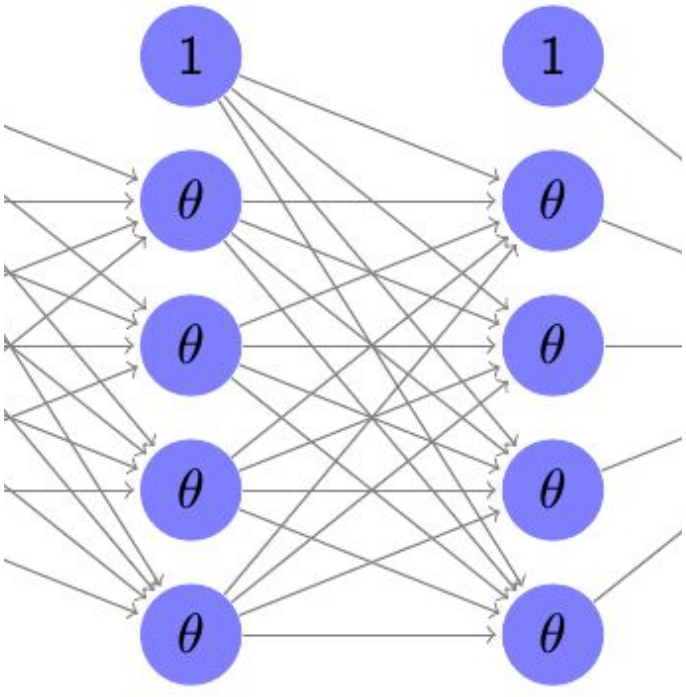
# Intro to CNN

- 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:
  - $W2 = (W1 - F)/S + 1$
  - $H2 = (H1 - F)/S + 1$
  - $D2 = D1$

# Intro to CNN

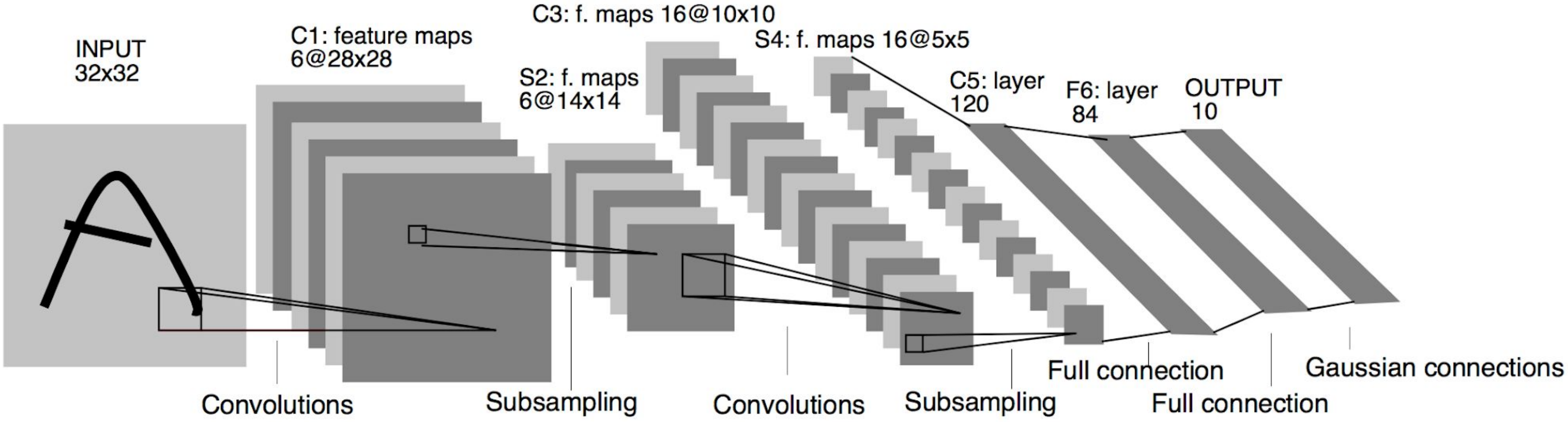
- INPUT holds the raw pixel values of the image.
- CONV layer computes the output of neurons that are connected to local regions in the input, each computing a dot product between their weights and the region they are connected to in the input volume.
- POOL layer performs a downsampling operation along 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.

# Intro to CNN



# Intro to CNN

**LeNet**-5 [1998, paper by LeCun et al.]



# Content

- Intro

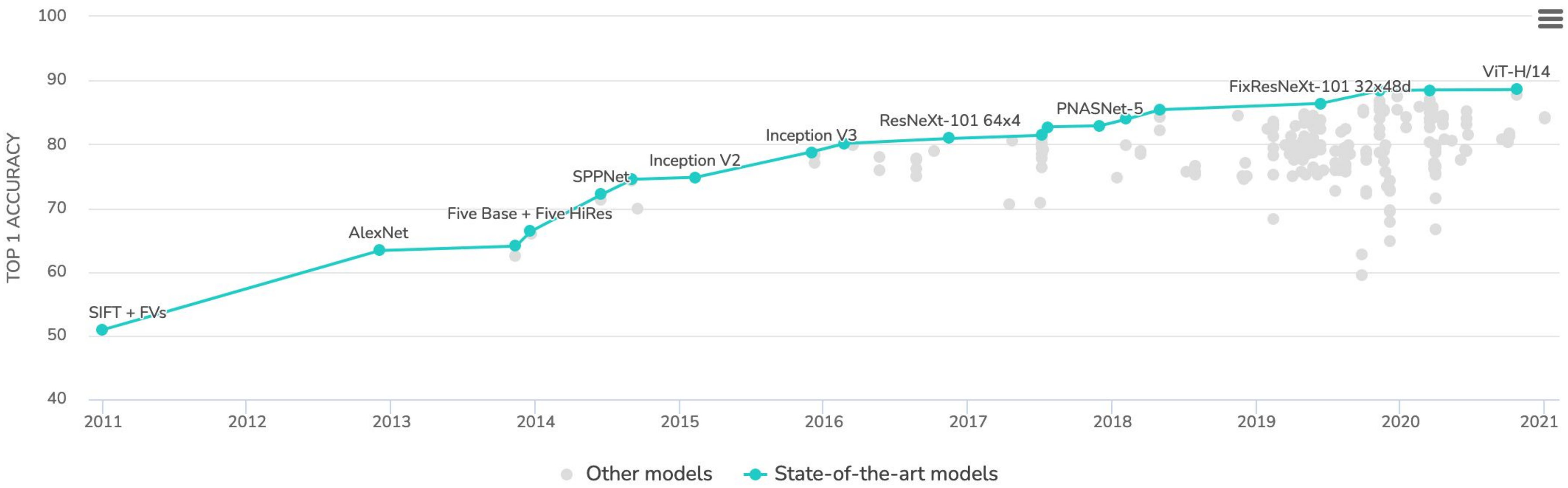
- DL review
  - neural networks
  - training & testing
  - loss functions
  - supervised, semi-supervised

- Main components of CNNs (motivation and details)
  - convolutional layer
  - pooling layer

- **Dataset**
  - ImageNet

- Classes: 1000
- Problem: classification
- Set:
    - 1 200 000 examples

# Intro to CNN




https://paperswithcode.com/sota/image-classification-on-imagenet

# Intro to CNN