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:

- mini-project [70 % of FINAL SCORE]
- paper review [30 % of FINAL SCORE]

important dates:

24 December, 23:59 : mini-project deadline (SOFT)
paper review deadline (SOFT)
14 January, 23:59 : mini-project deadline (HARD, PENALTY 30%)
paper review deadline (HARD, PENALTY 30%)

course repo:
https://github.com/lyubonko/ucu2021cv

Overview of the course

Unit I (19 Nov, 09.00-12.00)

[T] Intro to Convolution Neural Networks (CNNs)

[P] pytorch

Unit II (20 Nov, 09.30-12.30)

[T] CNNs in depth

[P] classification

Unit III (17 Dec, 13.00-16.00)

[T] Attention in CV

[P] Transformers

Unit IV (18 Dec, 09.30-12.30)

[T] Object Detection

[P] project structure, detection

Overview of the course

Unit I (19 Nov, 09.00-12.00)
- [T] Intro to Convolution Neural Networks (CNNs)
- [P] pytorch

Unit II (20 Nov, 09.30-12.30)
- [T] CNNs in depth
- [P] classification

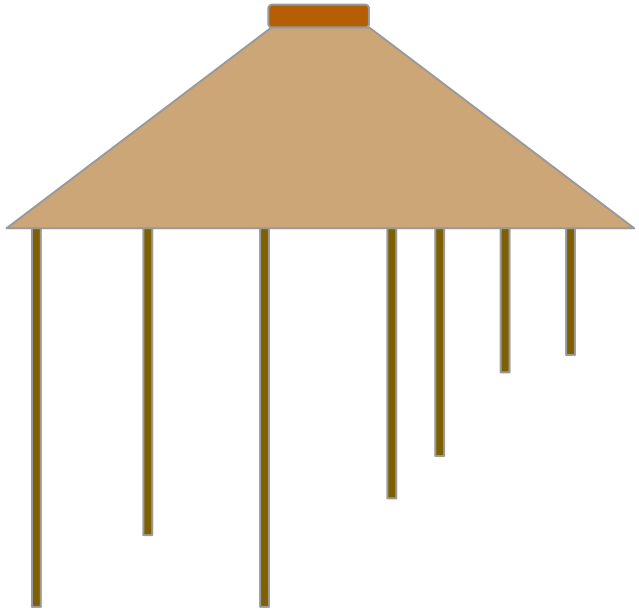
Unit III (17 Dec, 13.00-16.00)
- [T] Attention in CV
- [P] Transformers

Unit IV (18 Dec, 09.30-12.30)
- [T] Object Detection
- [P] project structure, detection

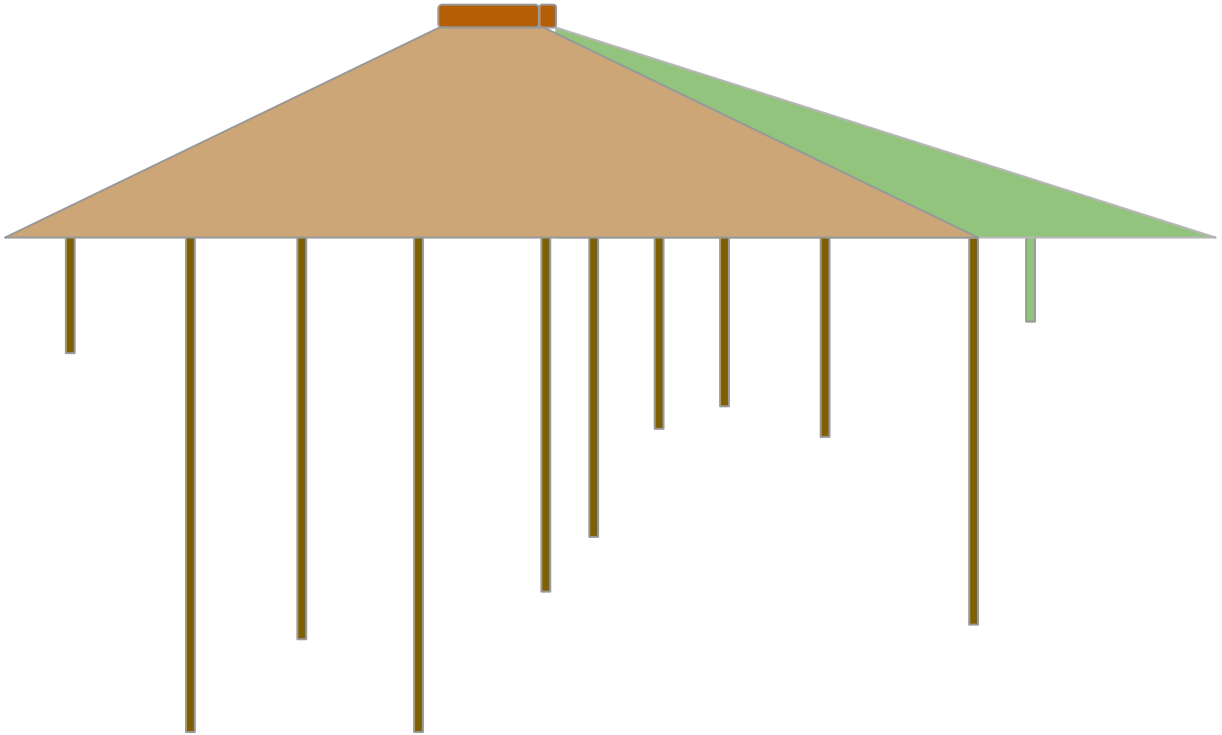
Goals of the Course

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

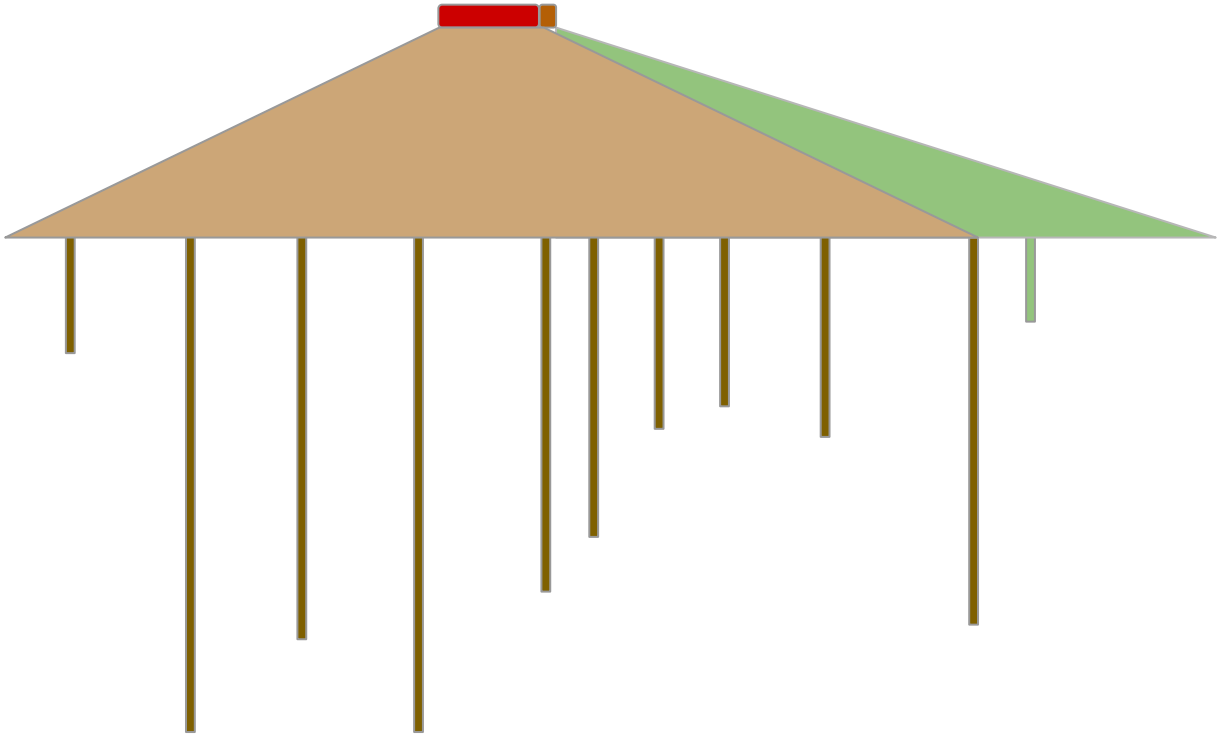
Goals of the Course



Goals of the Course

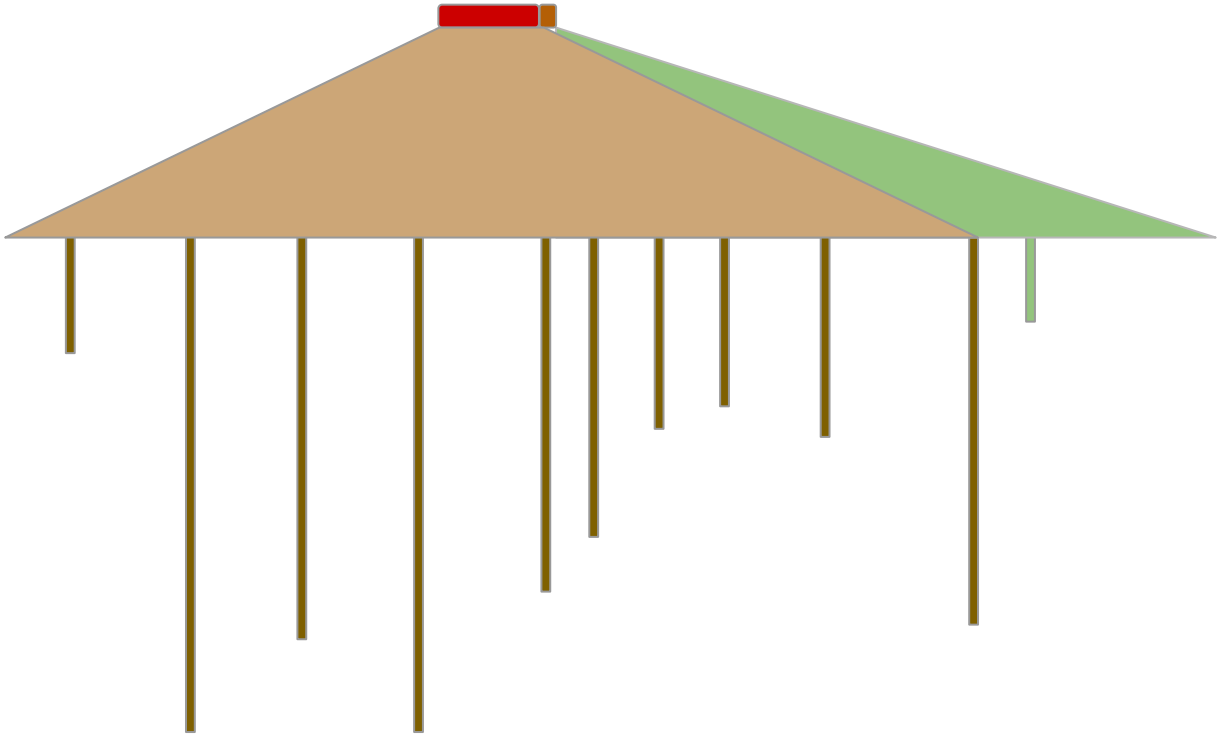


Goals of the Course



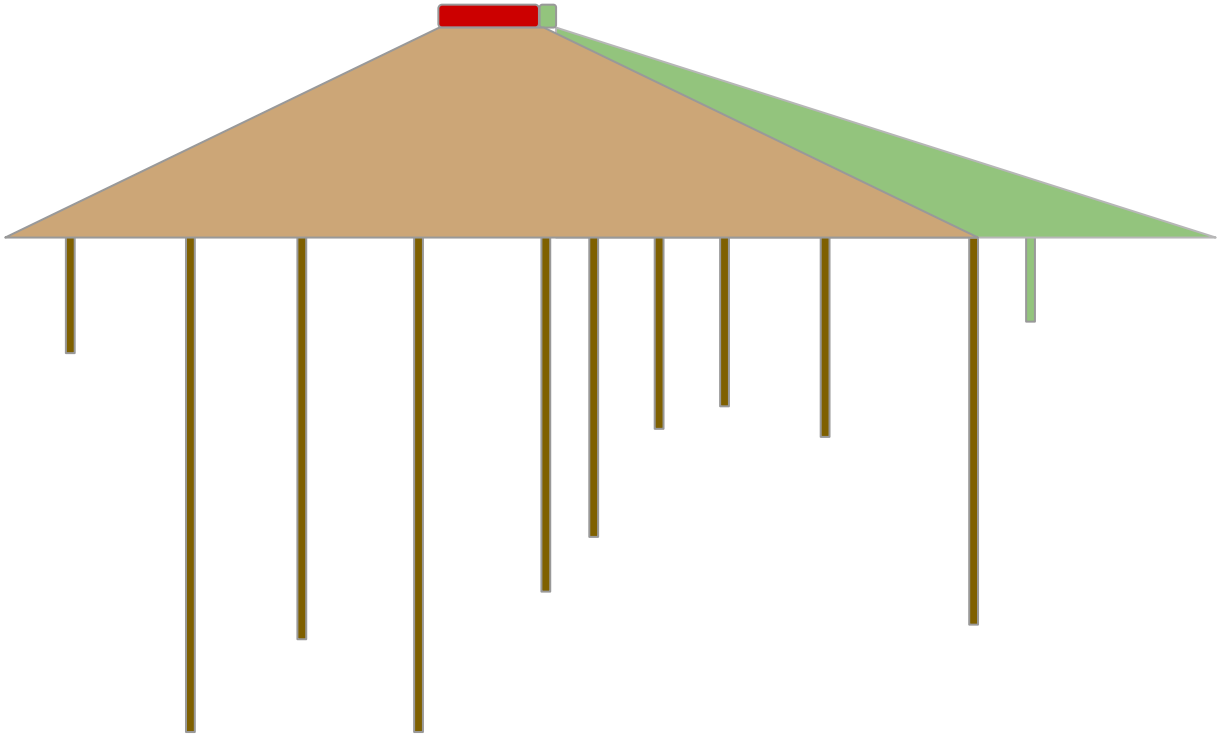
- essential CNNs elements/blocks

Goals of the Course



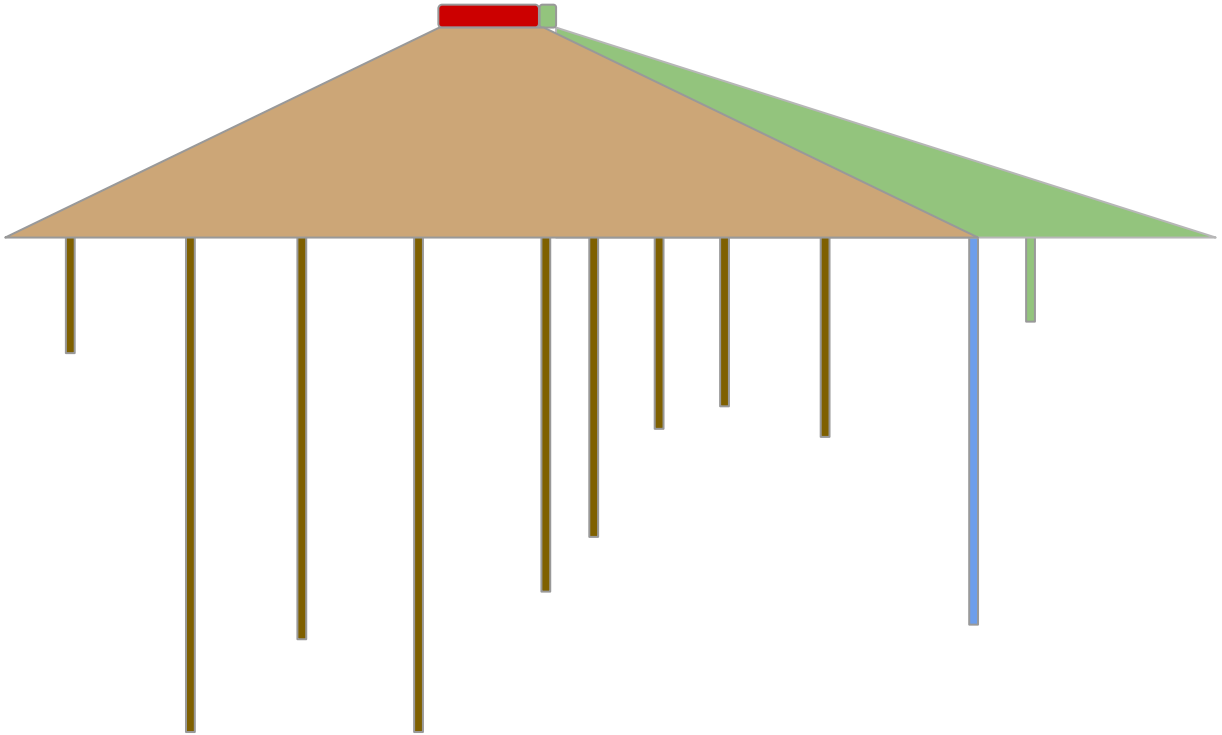
- essential CNNs elements/blocks
- modern CNNs architectures

Goals of the Course



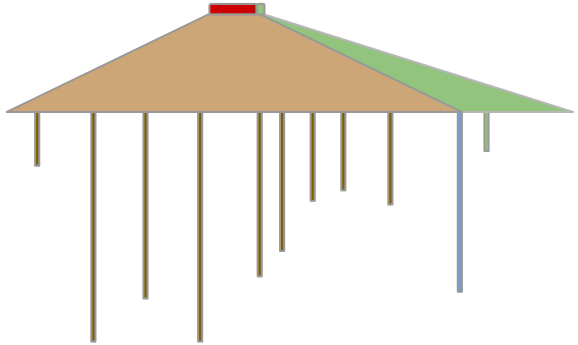
- essential CNNs elements/blocks
- modern CNNs architectures
- attention in CV

Goals of the Course



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

Goals of the Course



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

Content of today lecture

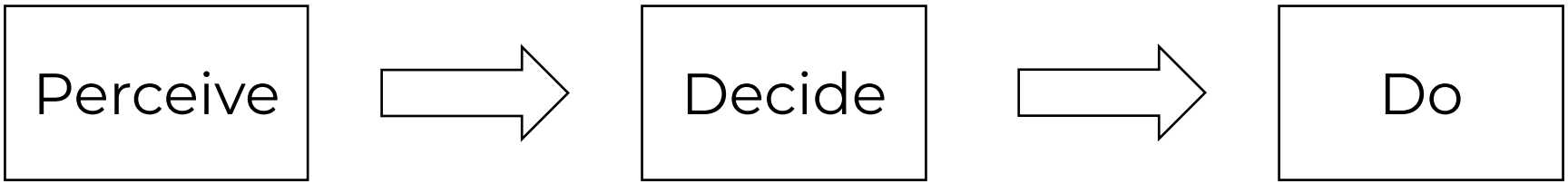
- **Intro, Context**

- ML/DL review
  - training & testing
  - neural networks

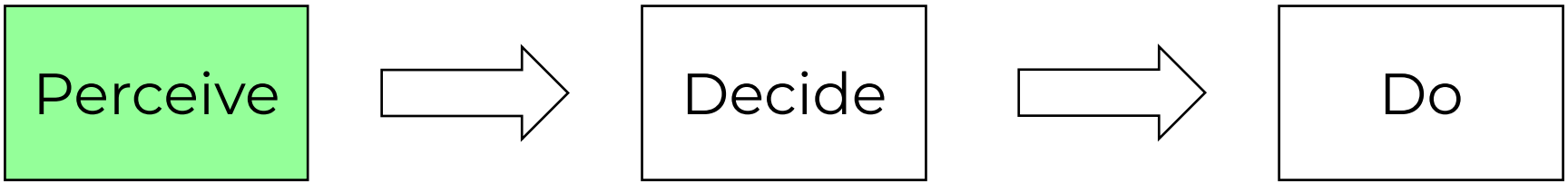
- CV specific setups
  - supervised, semi-supervised,
  - self-supervised, auto-encoders

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

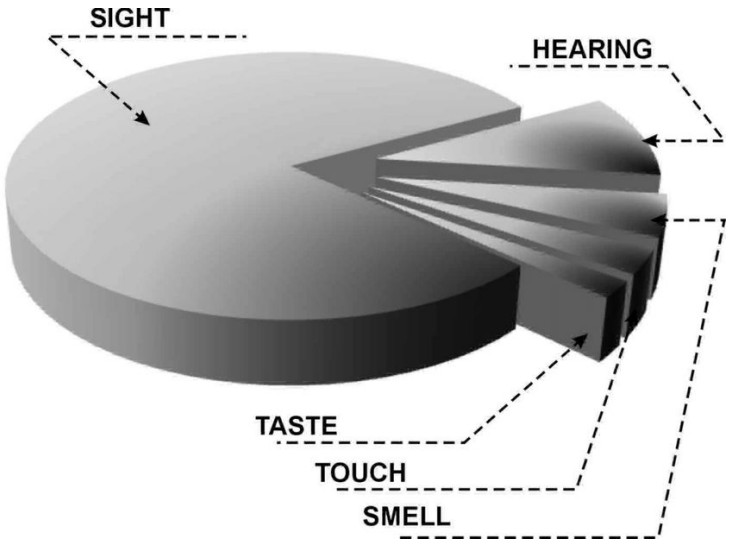
Intro



Intro



Intro



*The goal of **computer vision** is to perceive [extract useful information] from visual input, like images and 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

- scale variations
- occlusions
- illuminations
- deformations
- ....
-

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

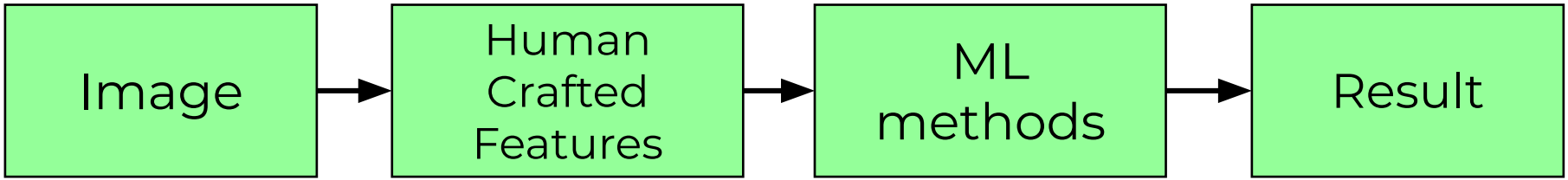
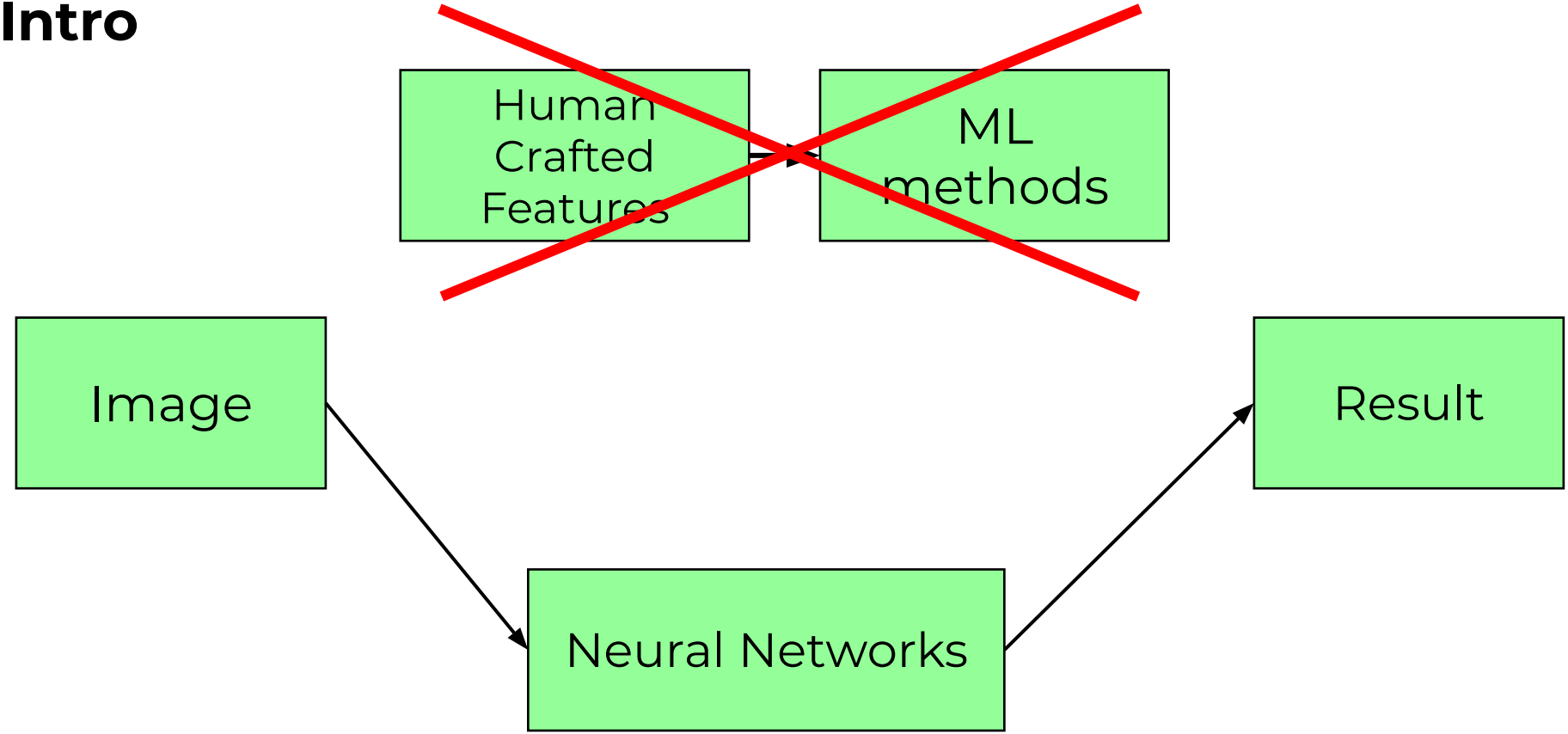
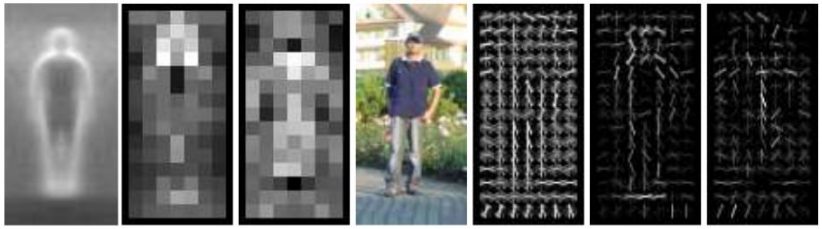


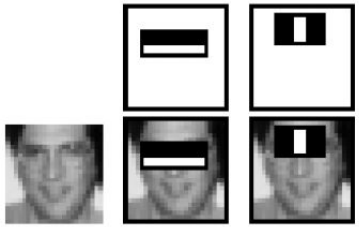
Image
Human Crafted Features
ML methods
Result
Descriptors: SIFT, SURF, etc.
Linear, SVM, Trees

Intro

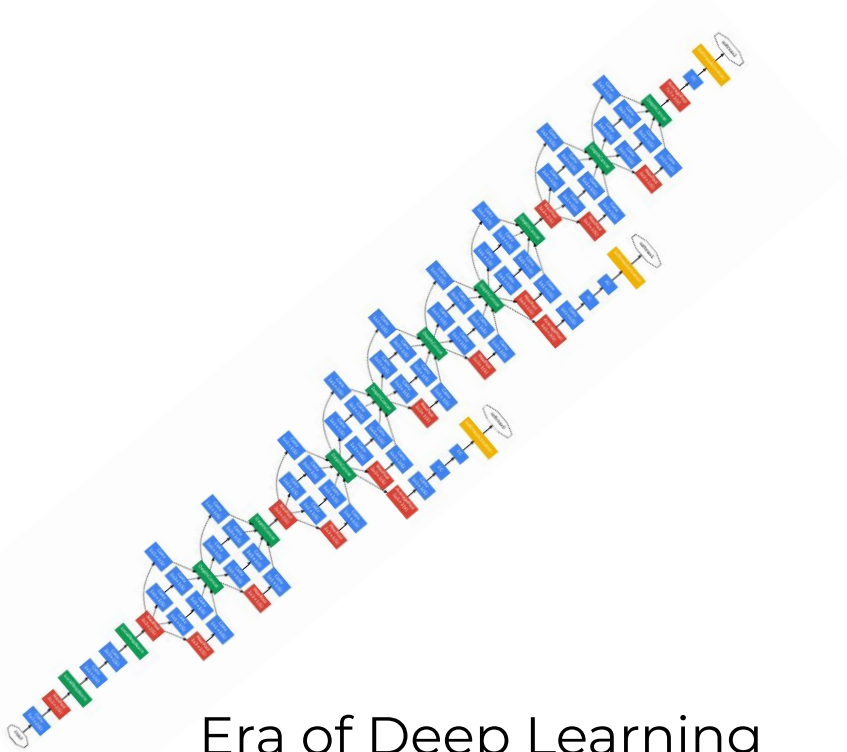

Human Crafted Features
ML methods
Image
Neural Networks
Result

Intro





Era of Human-Crafter Features



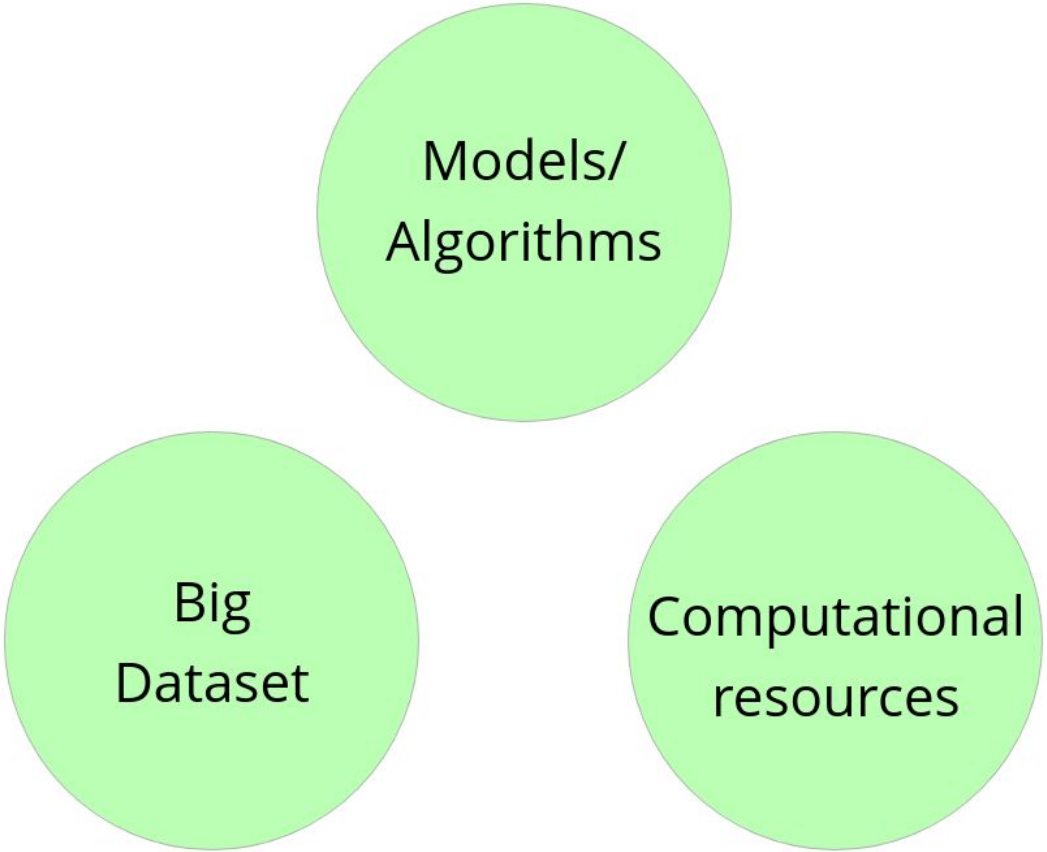
Era of Deep Learning

1986
BackProp

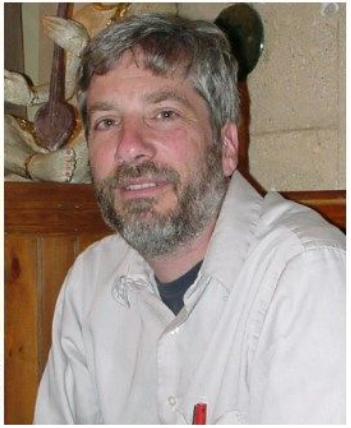
1998
LeNet

2012
AlexNet

Intro



Intro



Vladimir Vapnik **Larry Jackel**

1. Jackel bets (one fancy dinner) that by March 14, 2000, people will understand
quantitatively why big neural nets working on large databases are not so bad. (Understanding means that there will be clear conditions and bounds)

Vapnik bets (one fancy dinner) that Jackel is wrong.

But .. If Vapnik figures out the bounds and conditions, Vapnik still wins the bet.

****

2. Vapnik bets (one fancy dinner) that by March 14, 2005, no one in his right mind will use neural nets that are essentially like those used in 1995.

Jackel bets (one fancy dinner) that Vapnik is wrong

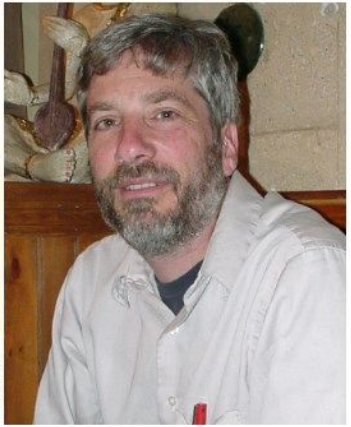
3/14/95
V. Vapnik

3/14//95
L. Jackel

3/14/95
Witnessed by Y. LeCun

Intro

Vladimir Vapnik



Larry Jackel

On the Equivalence between Neural Network and Support Vector Machine

Yilan Chen
Computer Science and Engineering
University of California San Diego
La Jolla, CA
yilan@ucsd.edu

Wei Huang
Engineering and Information Technology
University of Technology Sydney
Ultimo, Australia
weihuang.uts@gmail.com

Lam M. Nguyen
IBM Research
Thomas J. Watson Research Center
Yorktown Heights, NY
LamNguyen.MLTD@ibm.com

Tsui-Wei Weng
Halıcıoğlu Data Science Institute
University of California San Diego
La Jolla, CA
lweng@ucsd.edu

Abstract

Recent research shows that the dynamics of an infinitely wide neural network (NN) trained by gradient descent can be characterized by Neural Tangent Kernel (NTK) [27]. Under the squared loss, the infinite-width NN trained by gradient descent with an infinitely small learning rate is equivalent to kernel regression with NTK [4]. However, the equivalence is only known for ridge regression currently [6], while the equivalence between NN and other kernel machines (KMs), e.g. support vector machine (SVM), remains unknown. Therefore, in this work, we propose to establish the equivalence between NN and SVM, and specifically, the infinitely wide NN trained by soft margin loss and the standard soft margin SVM with NTK trained by subgradient descent. Our main theoretical results include establishing the equivalence between NN and a broad family of ℓ_2 regularized KMs with finite-width bounds, which cannot be handled by prior work, and showing that every finite-width NN trained by such regularized loss functions is approximately a KM. Furthermore, we demonstrate our theory can enable three practical applications, including (i) *non-vacuous* generalization bound of NN via the corresponding KM; (ii) *non-trivial* robustness certificate for the infinite-width NN (while existing robustness verification methods would provide vacuous bounds); (iii) intrinsically more robust infinite-width NNs than those from previous kernel regression. Our code for the experiments are available at `https://github.com/leslie-CH/equiv-nn-svm`.

Content

- Intro, Context

- **ML/DL review**
  - training & testing
  - neural networks

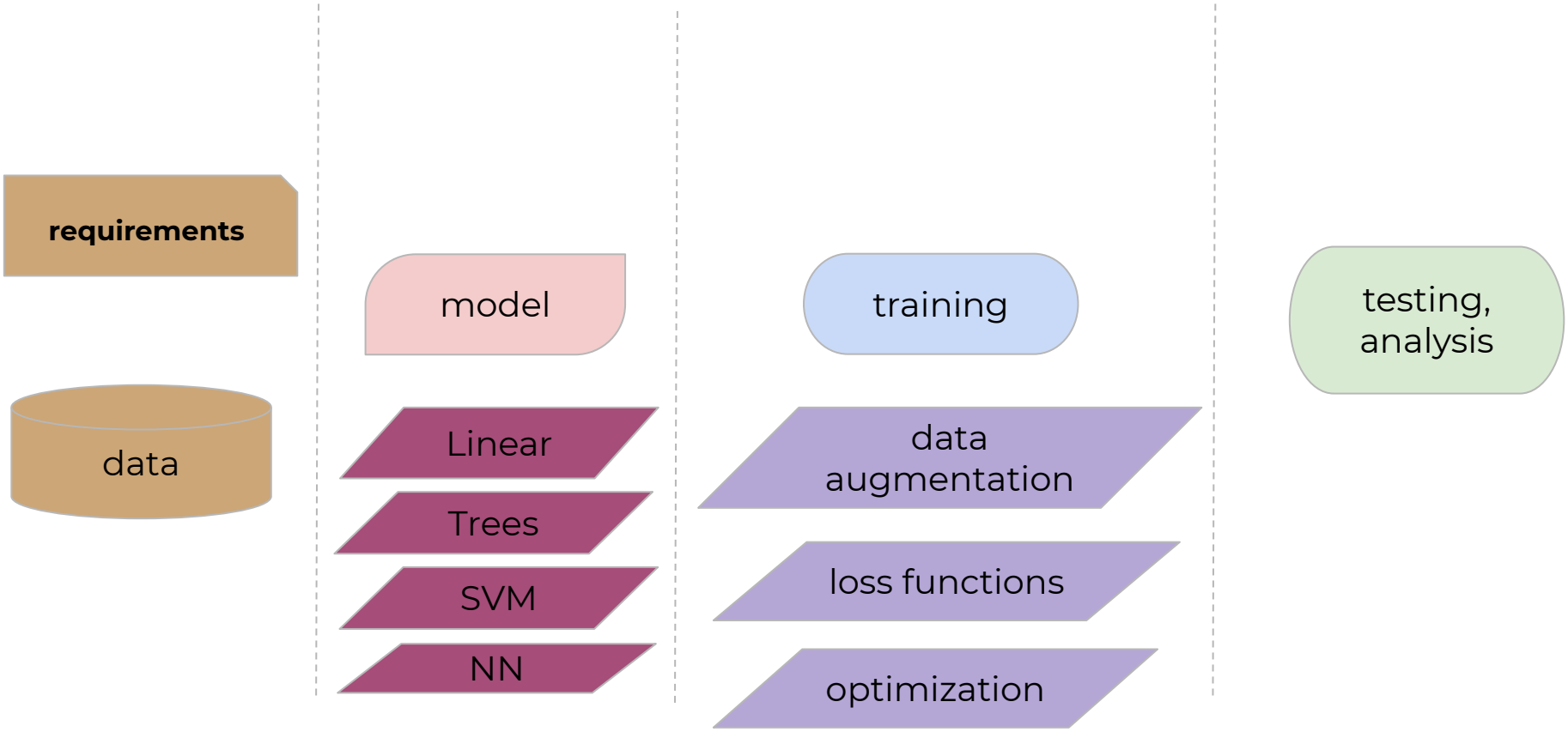
- CV specific setups
  - supervised, semi-supervised,
  - self-supervised, auto-encoders

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

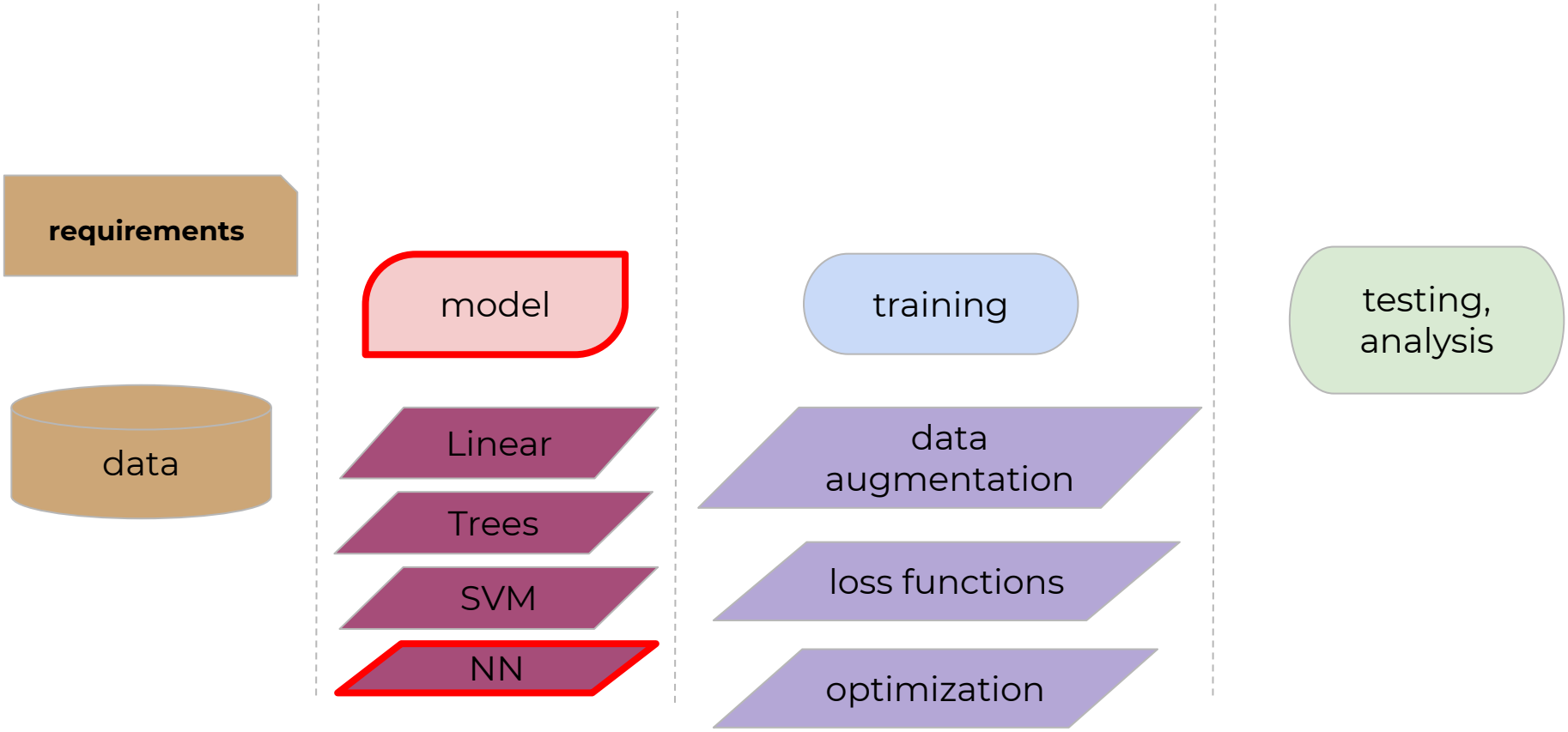
Machine Learning - types of learning

- Supervised
- Unsupervised
- Semi-supervised
- RL

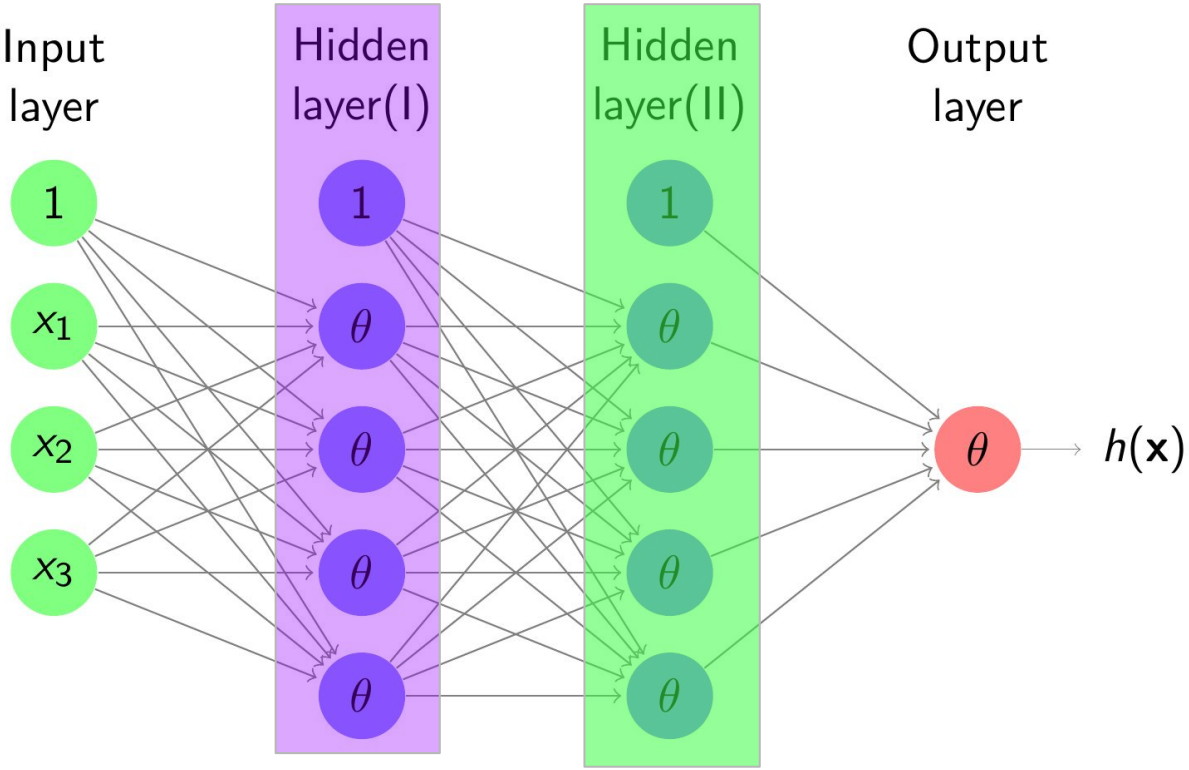
Machine Learning essential blocks



Machine Learning essential blocks

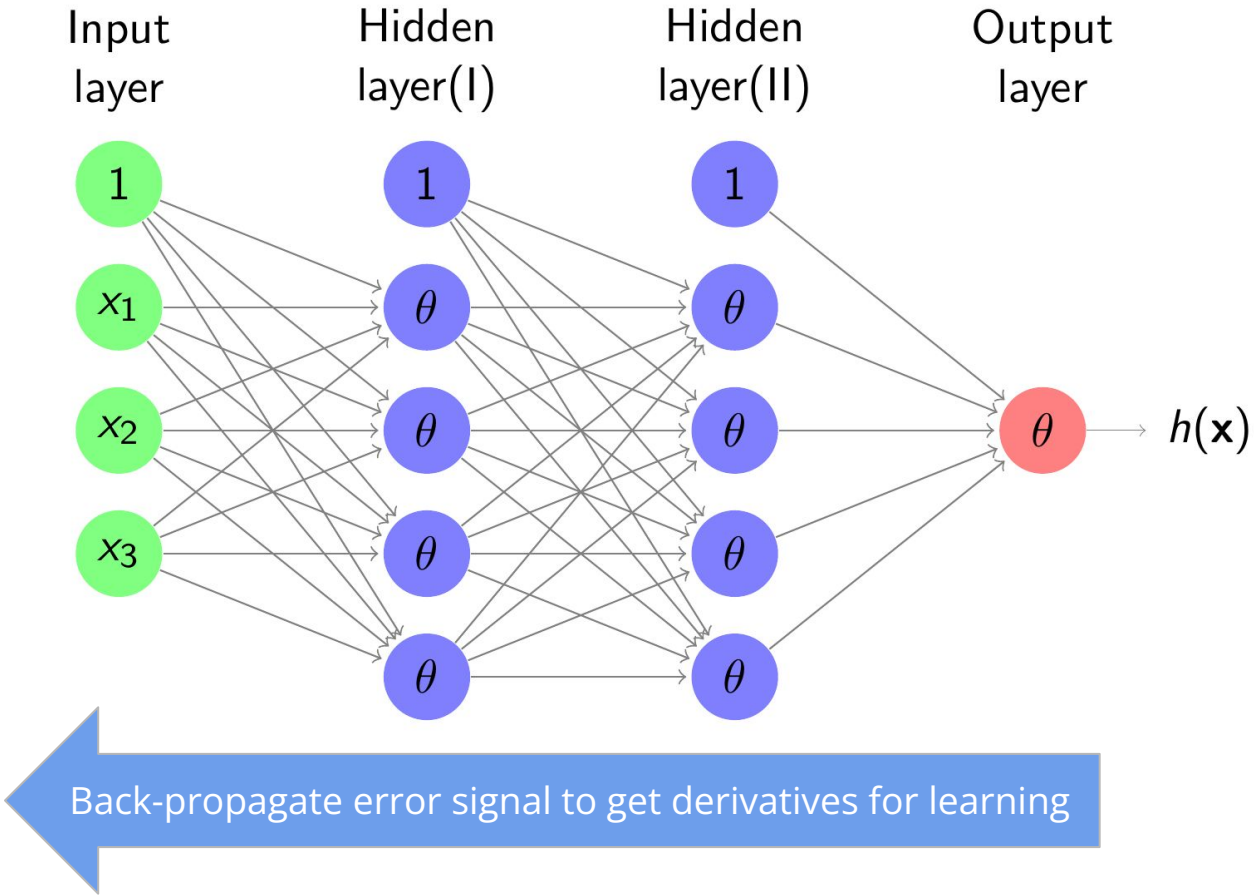


Model [Neural Networks]



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

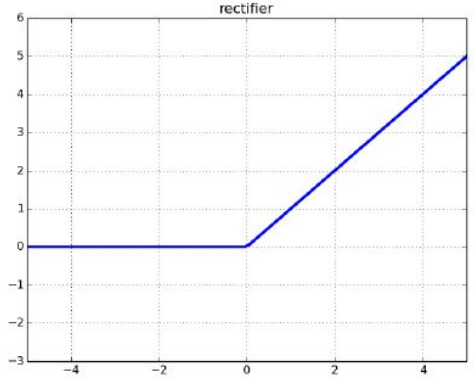
Model [Neural Networks]



Model [Neural Networks]

parameters in NN:

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

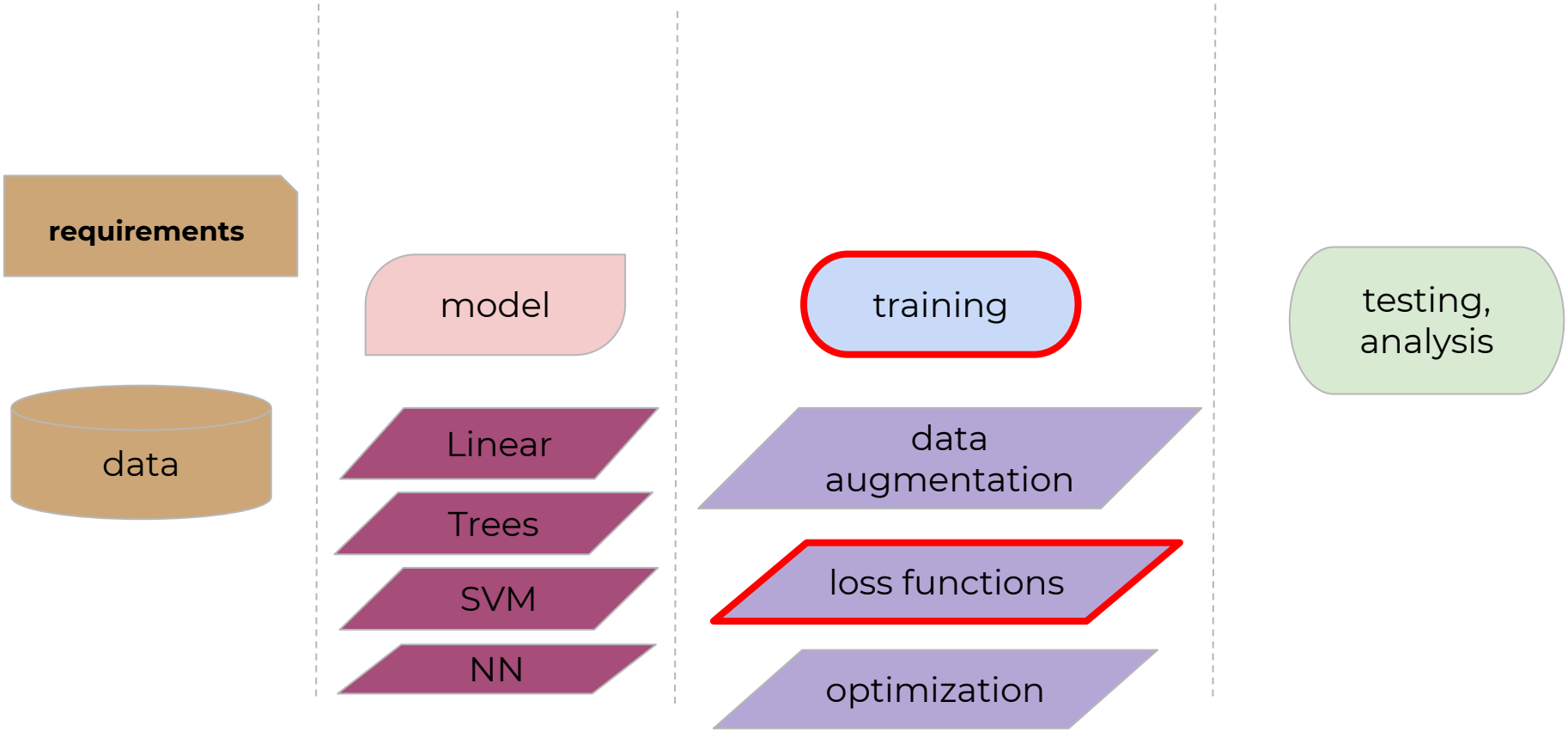


activation:

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

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

Machine Learning essential blocks



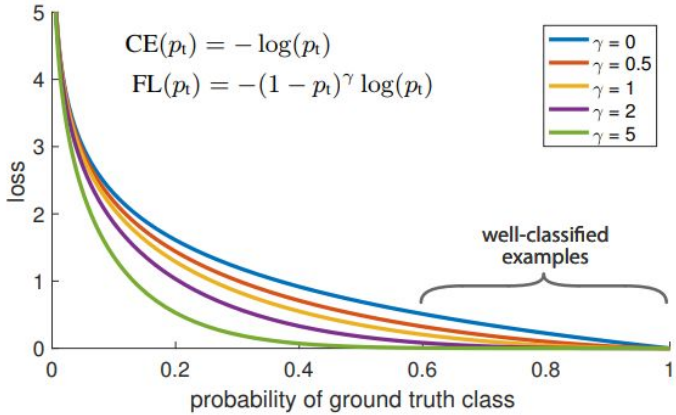
Loss function

Losses for classification, regression:

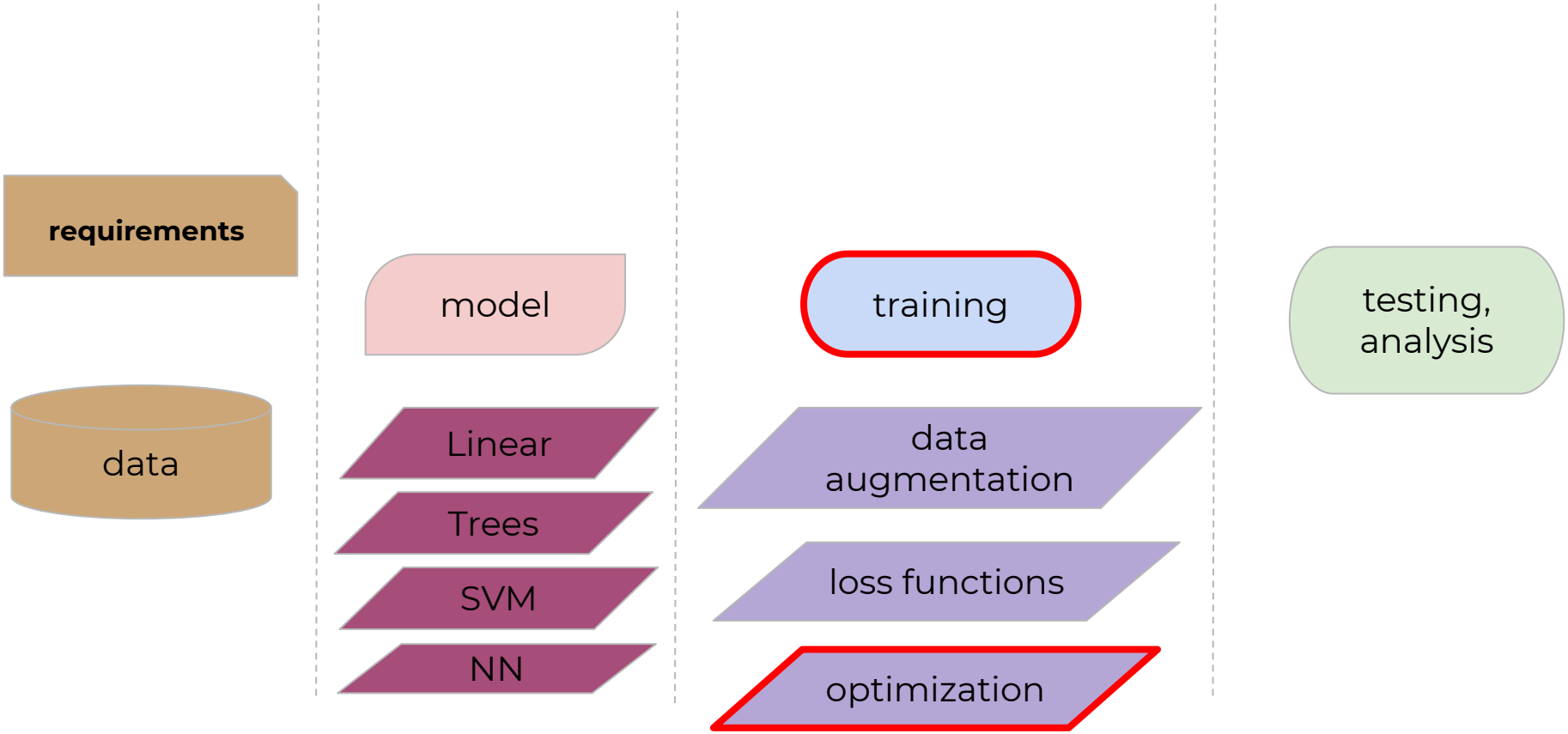
$$L_{logloss} = \frac{1}{N}\sum_{n=1}^{N}\sum_{k=1}^{M} y_{nk} \cdot log(p_{nk})$$

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

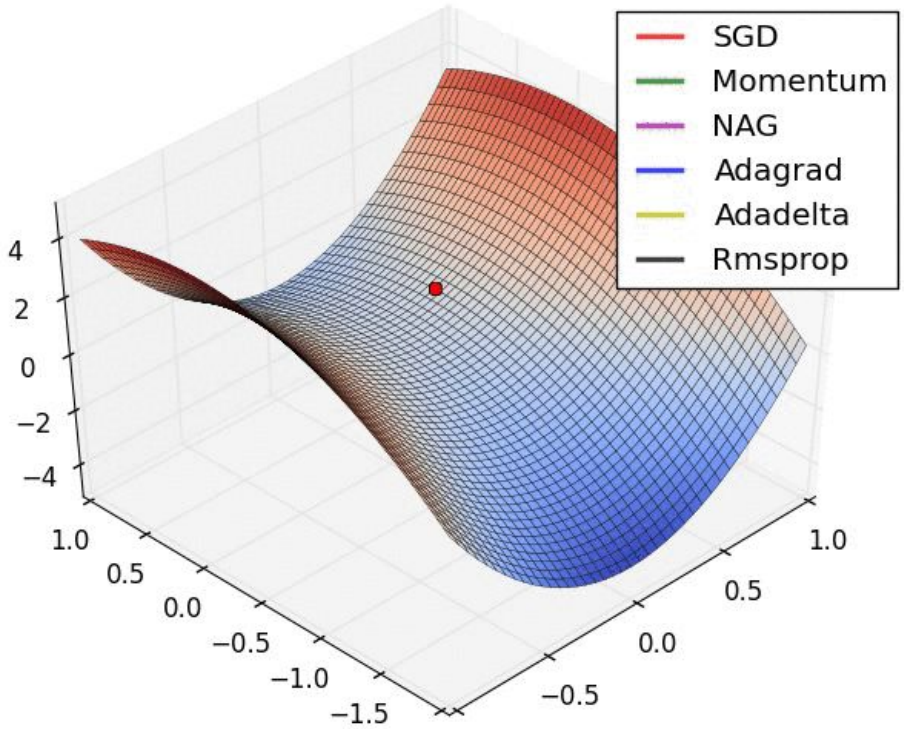
Focal Loss:



Machine Learning essential blocks



Optimization



Optimization

Input: Training set $\mathcal{S} \triangleq \cup_{i=1}^{n}\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}$, Loss function $l : \mathcal{W} \times \mathcal{X} \times \mathcal{Y} \to \mathbb{R}_+$, Batch size b, Step size $\eta > 0$, Neighborhood size $\rho > 0$.
Output: Model trained with SAM
Initialize weights $\boldsymbol{w}_0$, $t = 0$;
while *not converged* **do**
  Sample batch $\mathcal{B} = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), ...(\boldsymbol{x}_b, \boldsymbol{y}_b)\}$;
  Compute gradient $\nabla_{\boldsymbol{w}} L_{\mathcal{B}}(\boldsymbol{w})$ of the batch's training loss;
  Compute $\hat{\boldsymbol{\epsilon}}(\boldsymbol{w})$ per equation 2;
  Compute gradient approximation for the SAM objective (equation 3): $\boldsymbol{g} = \nabla_w L_{\mathcal{B}}(\boldsymbol{w})|_{\boldsymbol{w}+\hat{\boldsymbol{\epsilon}}(\boldsymbol{w})}$;
  Update weights: $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta \boldsymbol{g}$;
  $t = t + 1$;
end
return $\boldsymbol{w}_t$

Algorithm 1: SAM algorithm

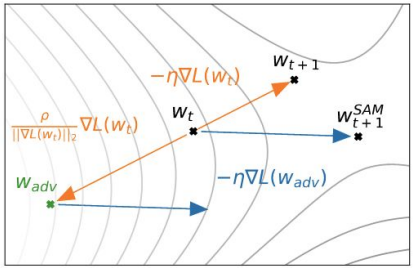
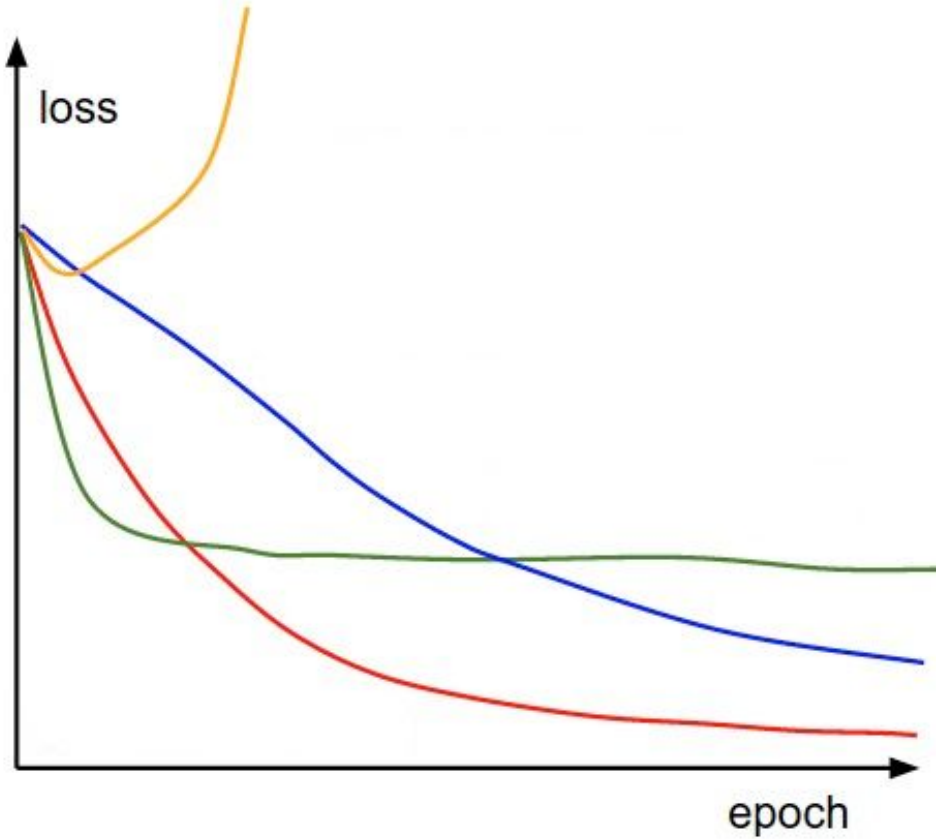


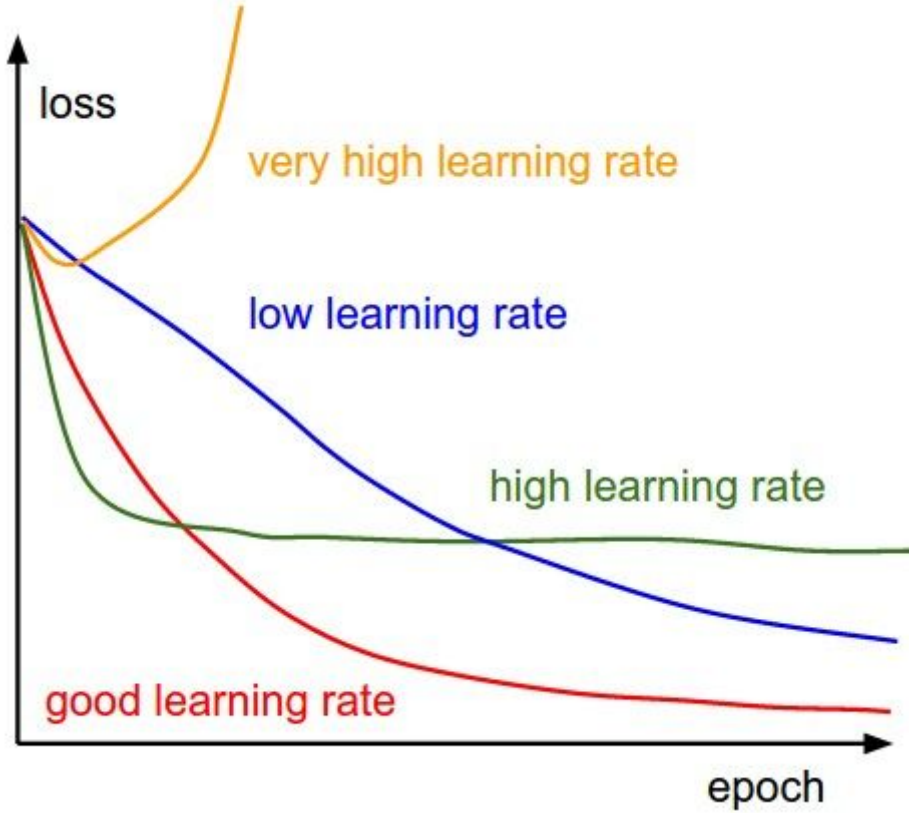
Figure 2: Schematic of the SAM parameter update.

2010.01412,
JAX(official), pytorch

Neural Networks



Neural Networks



Content

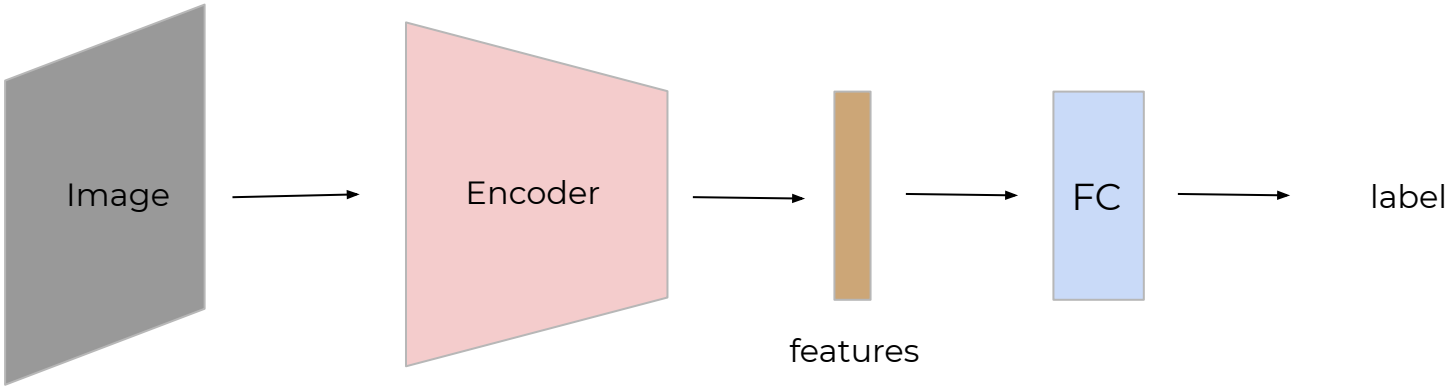
- **Intro, Context**

- ML/DL review
  - training & testing
  - neural networks

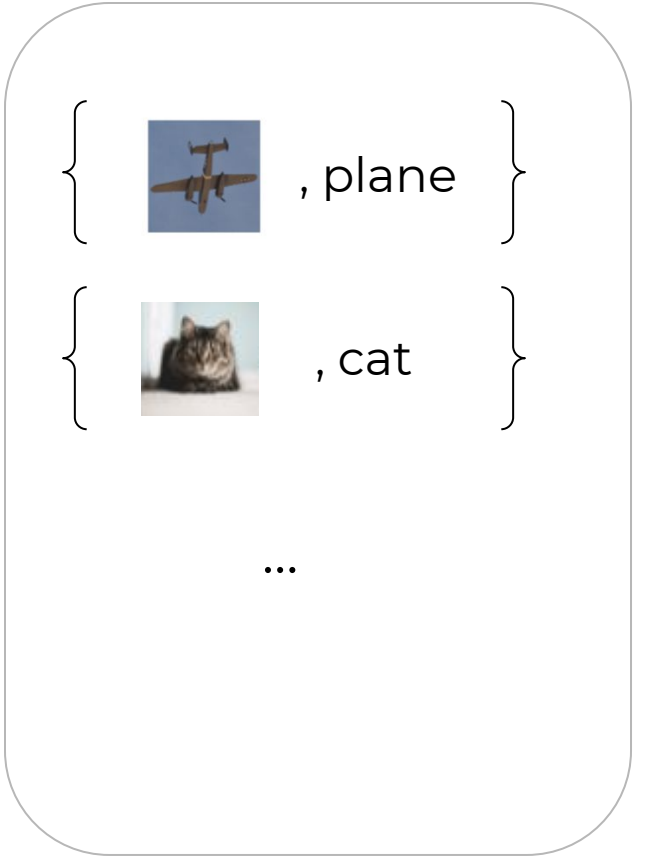
- **CV specific setups**
  - supervised, semi-supervised,
  - self-supervised, auto-encoders

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

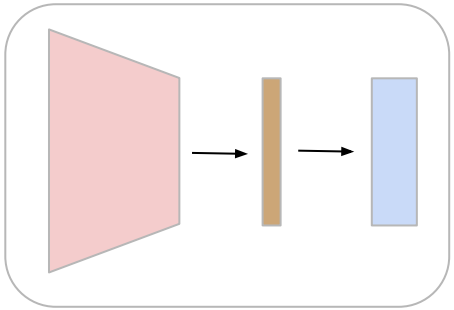
Neural Networks (supervised way)



Neural Networks (supervised way)



$F(\mathbf{x_n})$



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

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

...

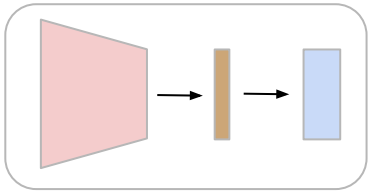
$$L_{train}(\omega) = \frac{1}{N}\sum_{n=1}^{N} e(F(\mathbf{x_n}), y_n)$$

Neural Networks (semi-supervised)

{  , plane }

{  , cat }

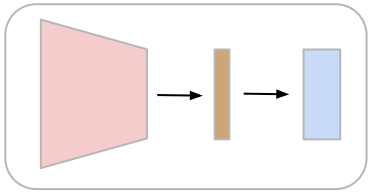
...

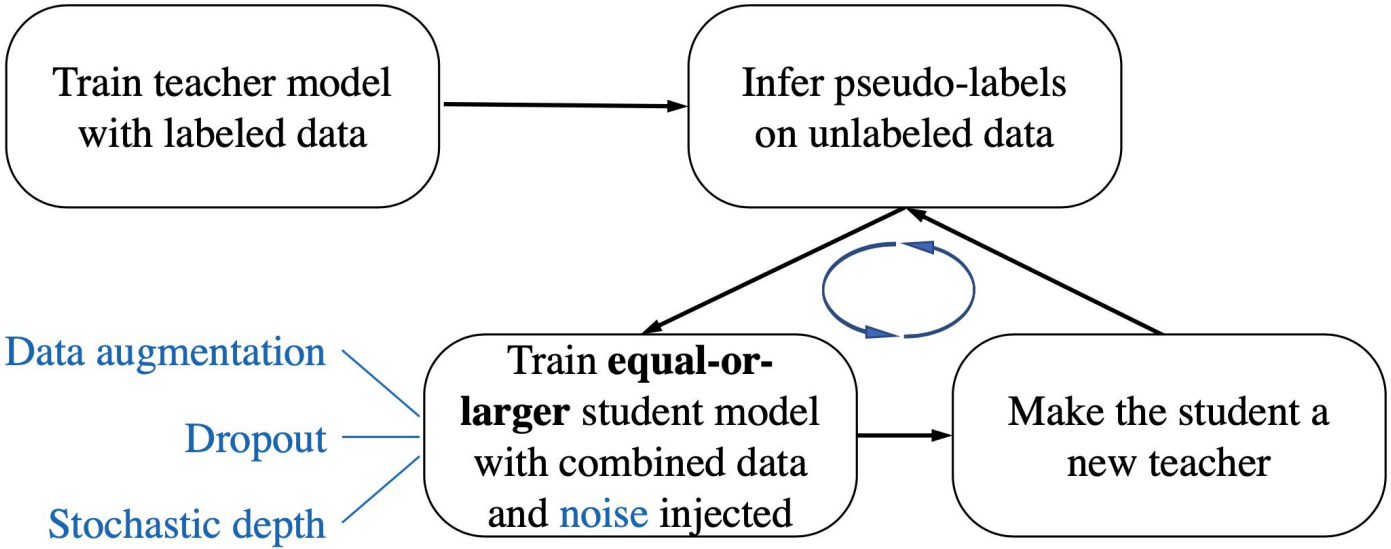


Train teacher model with labeled data

1911.04252

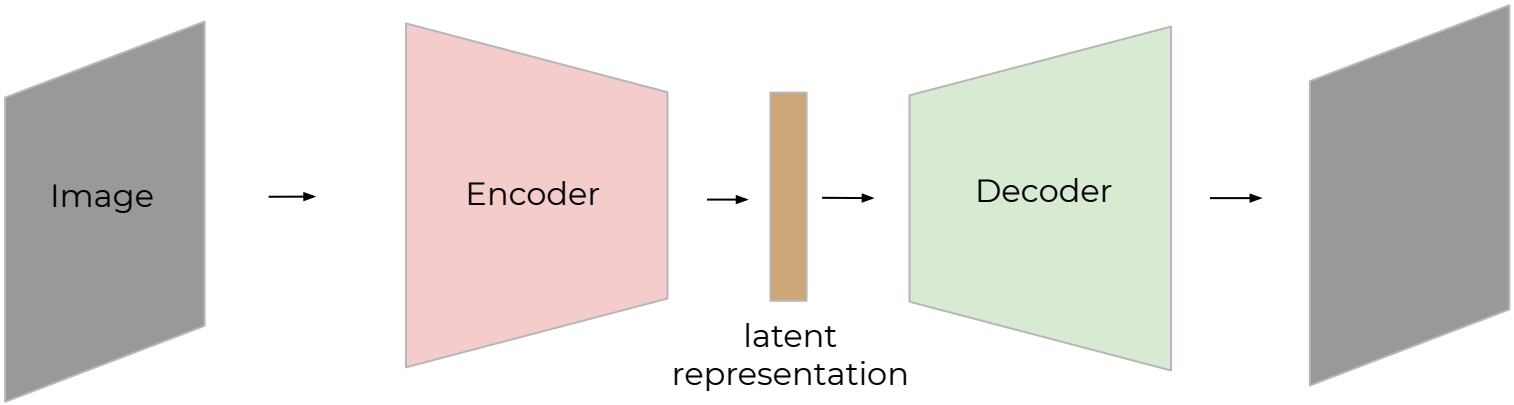
Neural Networks (semi-supervised)



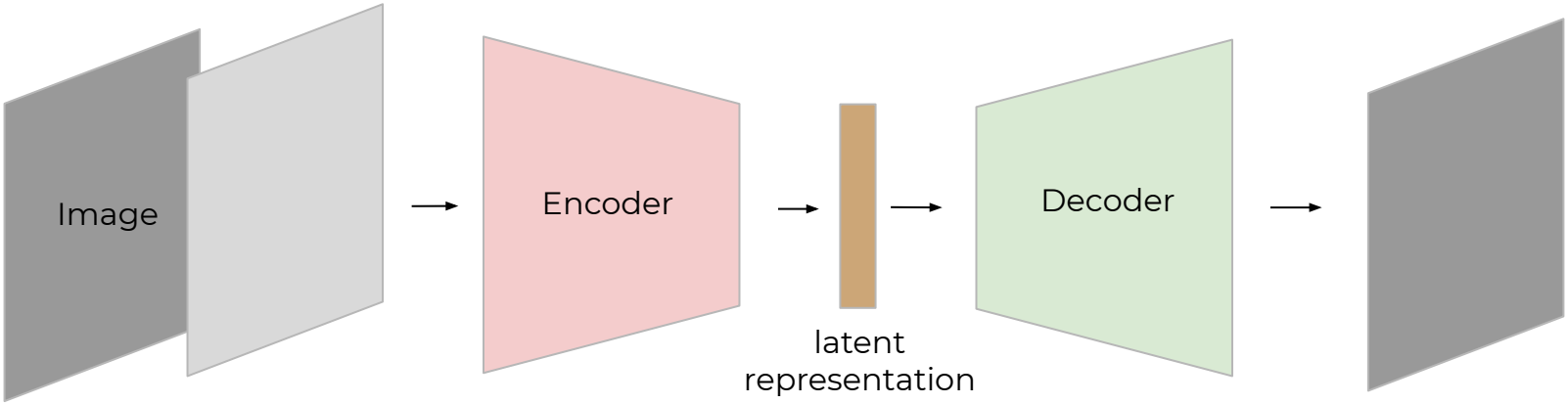


1911.04252

Neural Networks (auto-encoder way)

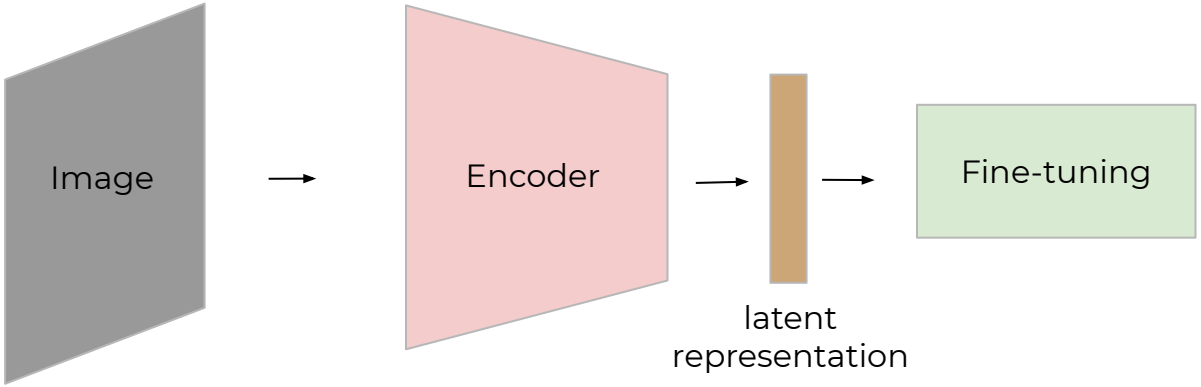


Neural Networks (auto-encoder way)



Neural Networks (auto-encoder way)

Fine-tuning

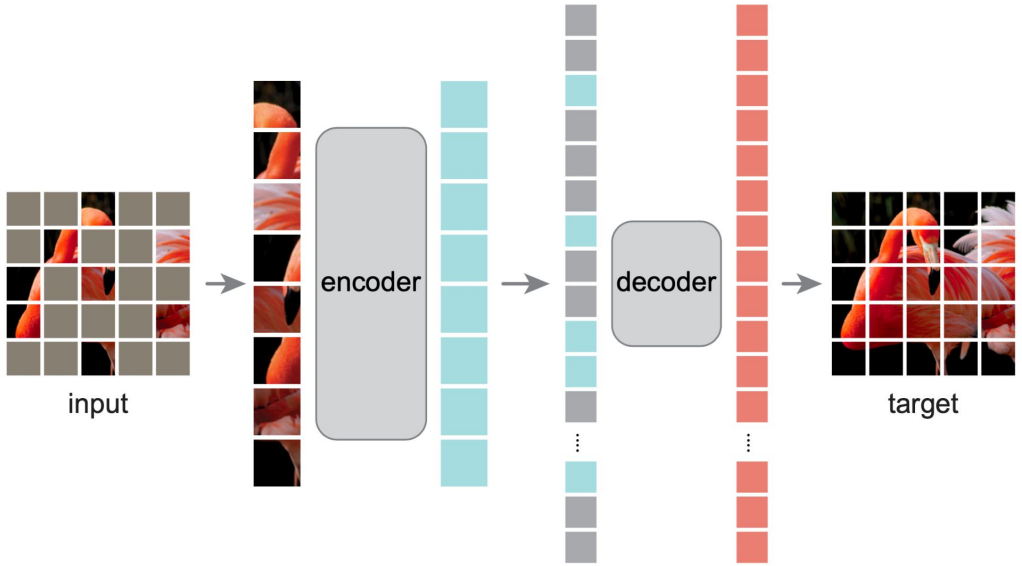


Downstream tasks:

- classification
- detection
- ....

Neural Networks (auto-encoder way)

MAE



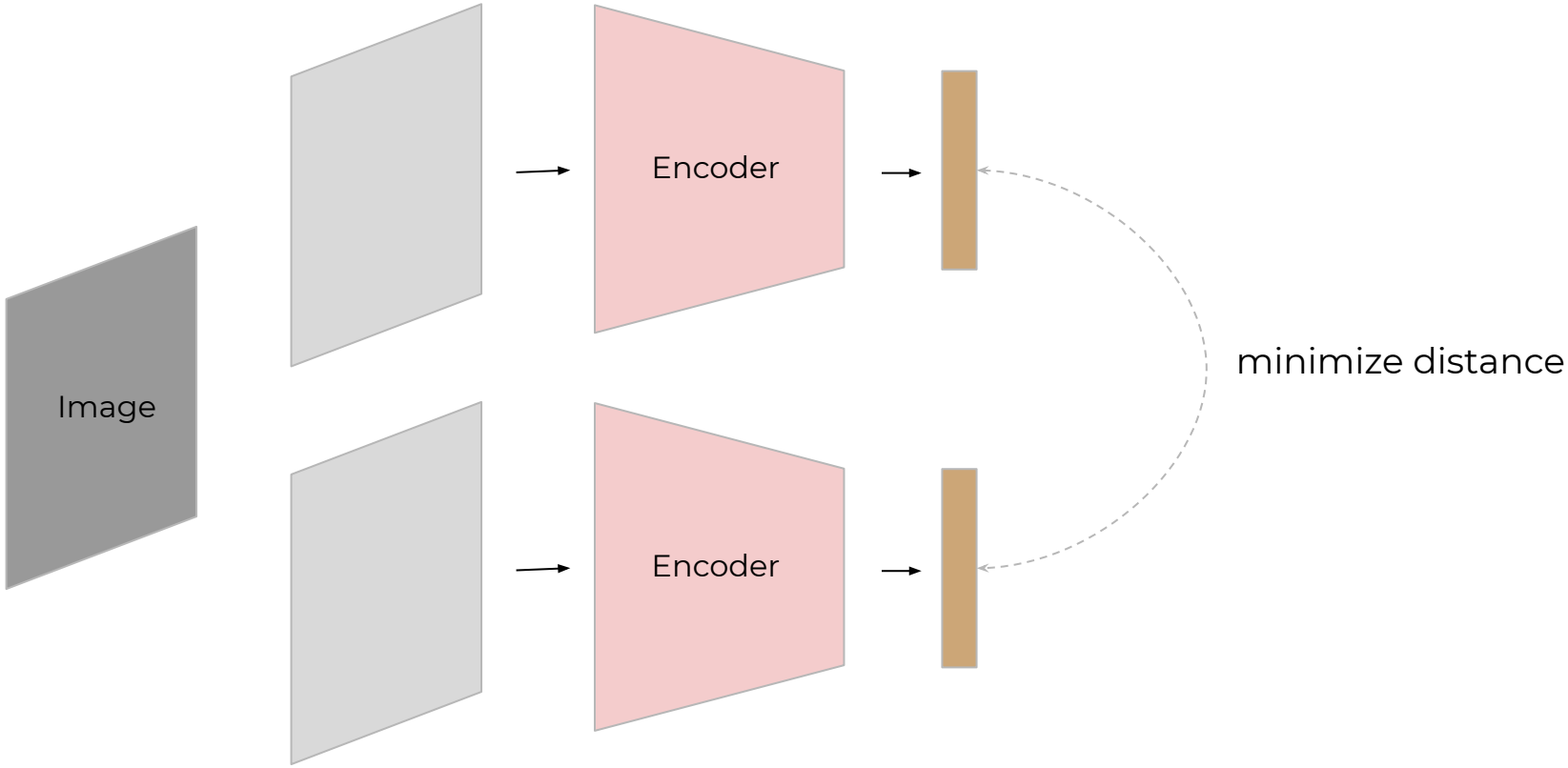
2111.06377

Neural Networks (auto-encoder way)

MAE

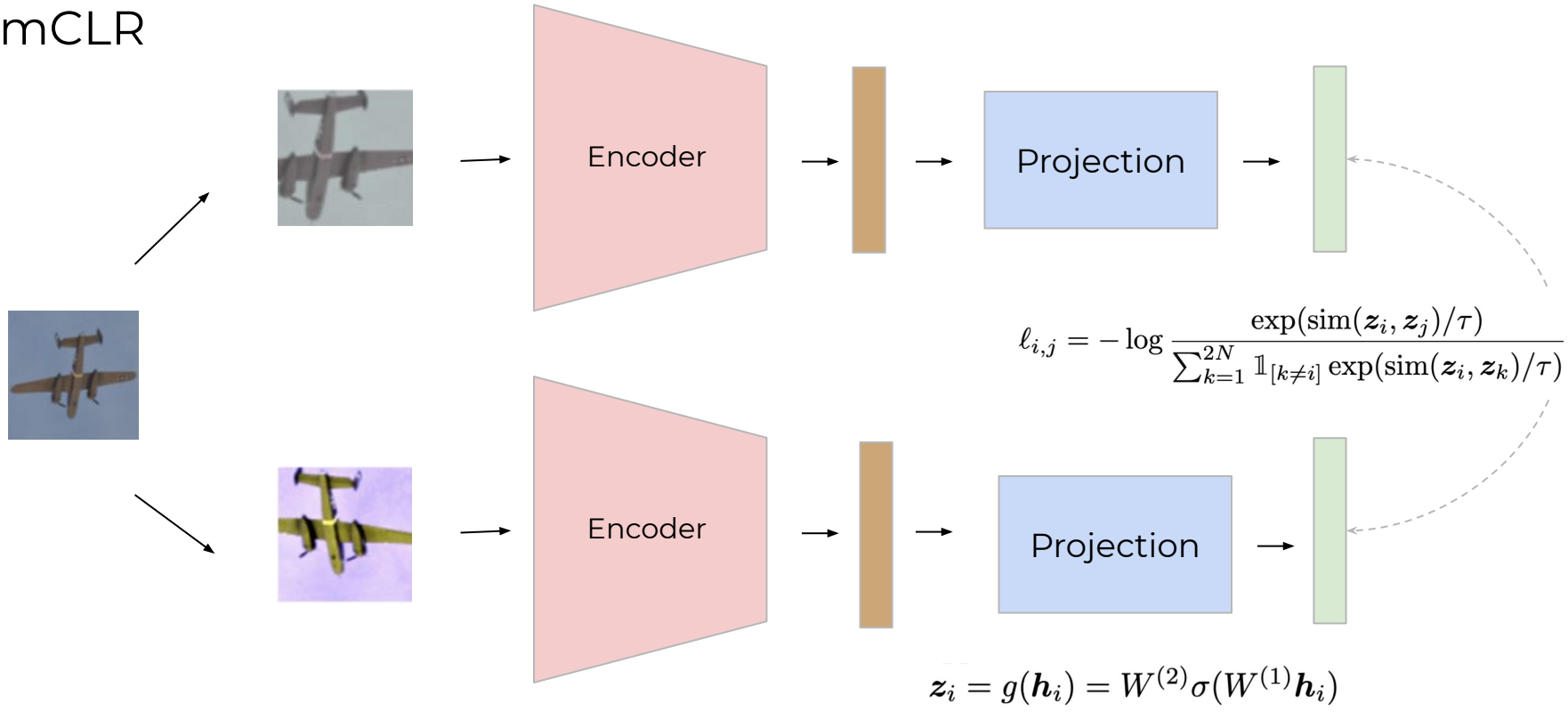
2111.06377

Neural Networks (self-supervised way)



Neural Networks (self-supervised way)

SimCLR



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

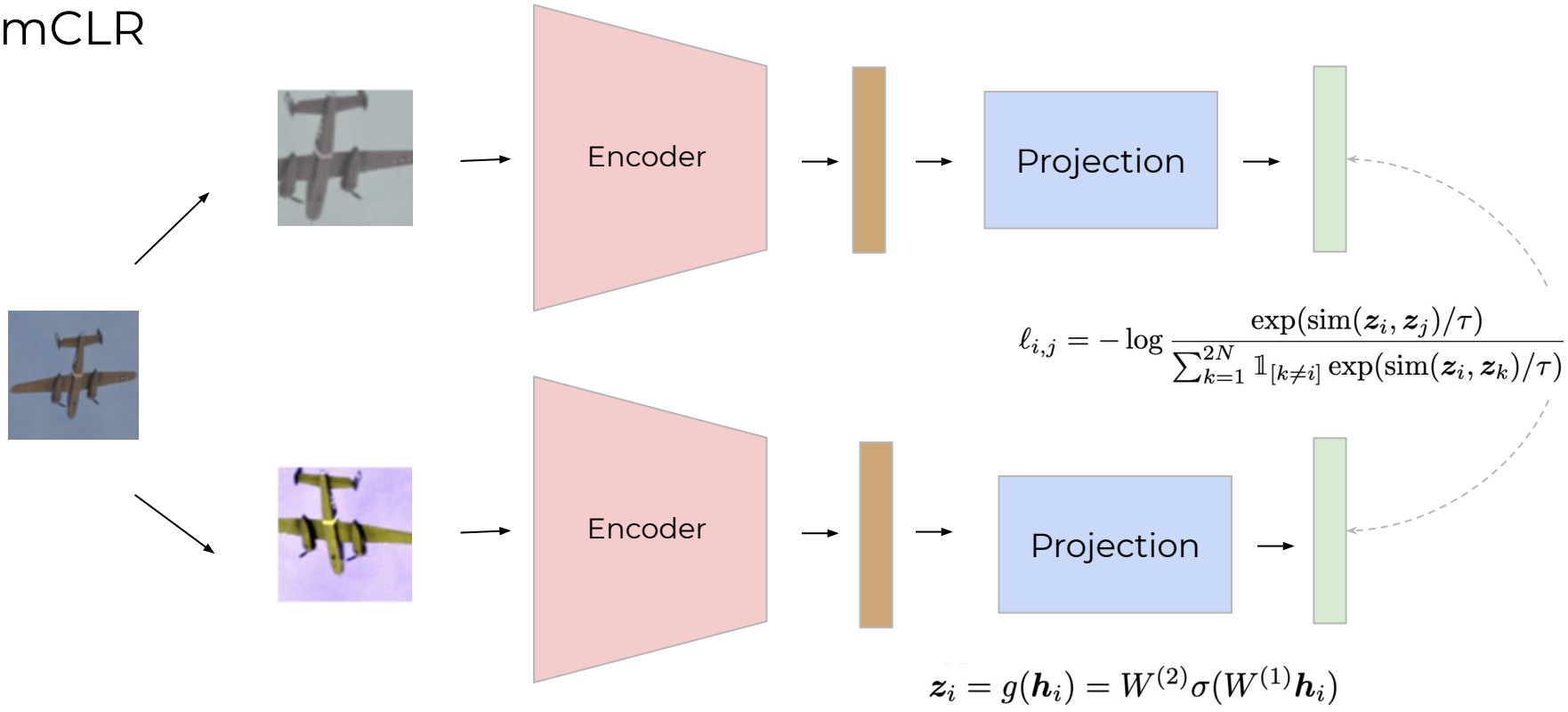
$$\boldsymbol{z}_i = g(\boldsymbol{h}_i) = W^{(2)}\sigma(W^{(1)}\boldsymbol{h}_i)$$

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

2002.05709

Neural Networks (self-supervised way)

SimCLR



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

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

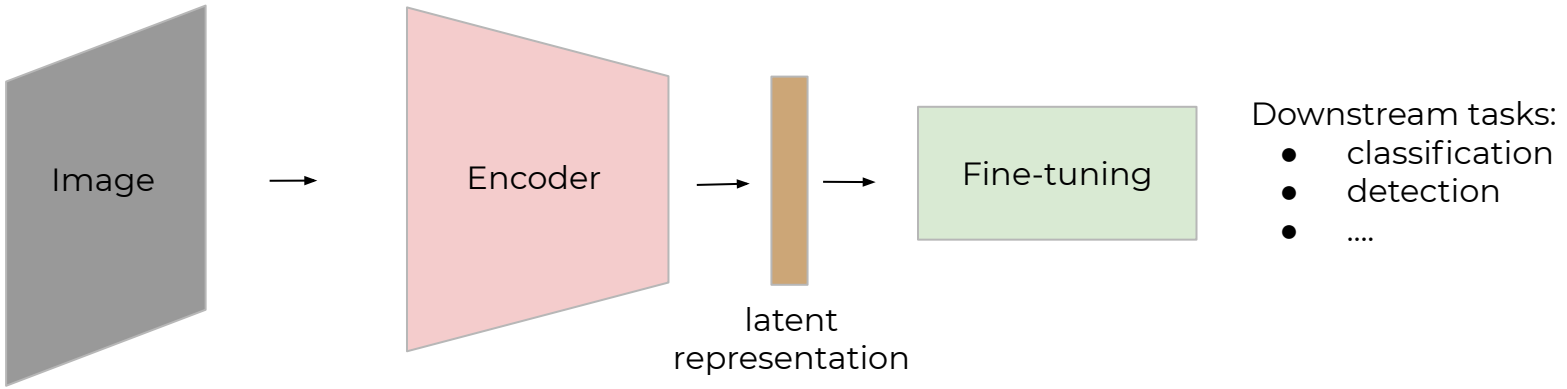
$$\boldsymbol{h}_i = f(\tilde{\boldsymbol{x}}_i) = \text{ResNet}(\tilde{\boldsymbol{x}}_i)$$

2002.05709

Neural Networks (self-supervised way)

SimCLR

Fine-tuning



2002.05709

Content

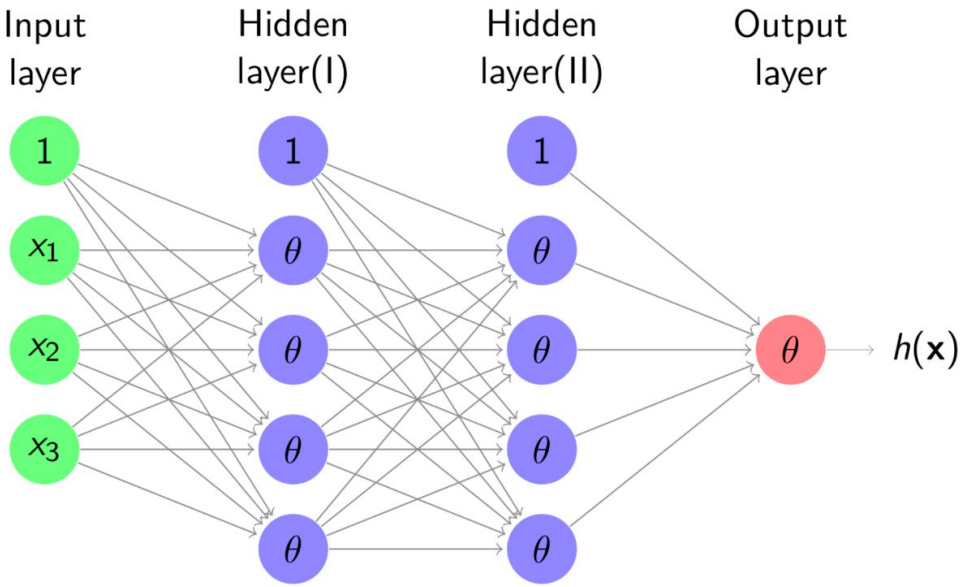
- Intro, Context

- ML/DL review
  - training & testing
  - neural networks

- CV specific setups
  - supervised, semi-supervised,
  - self-supervised, auto-encoders

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

Intro to CNN



Input layer
$64 \times 64 = 4096$

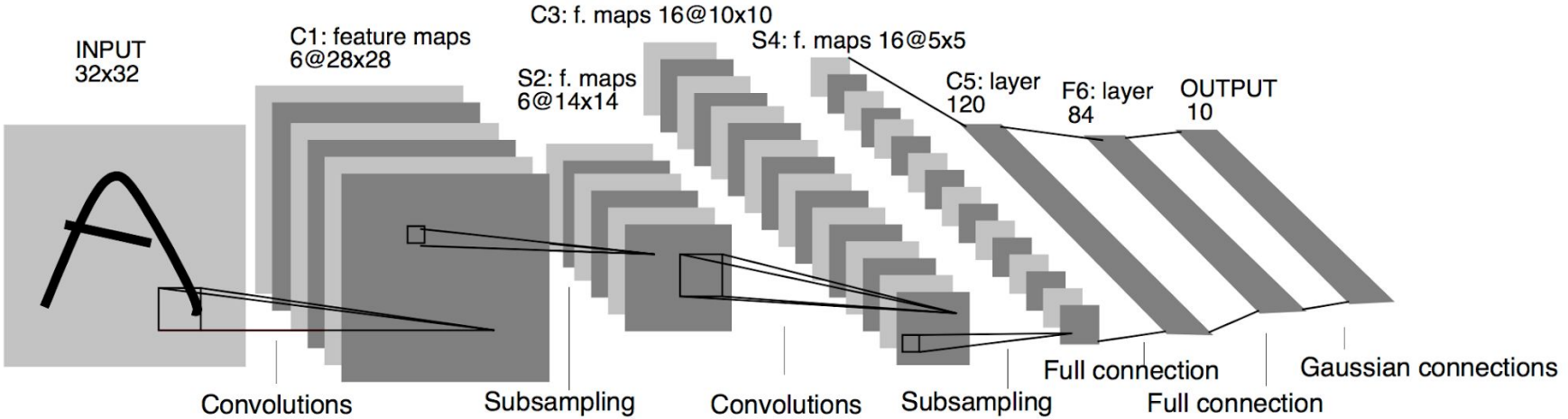
Weights: $4096\times$ (size of second layer)

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

Need for alternative architecture

Intro to CNN

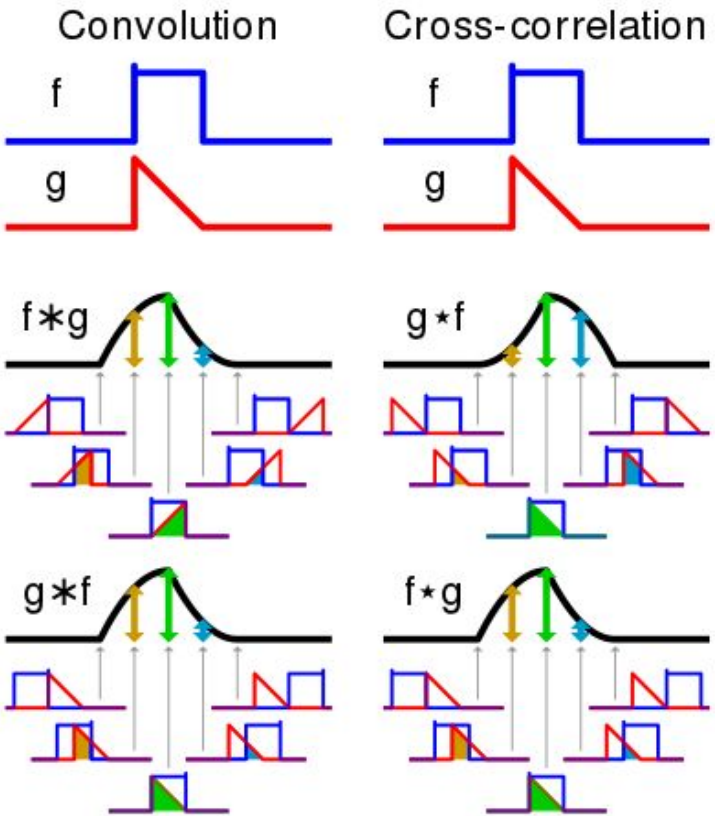
LeNet-5 [1998, paper by LeCun et al.]



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

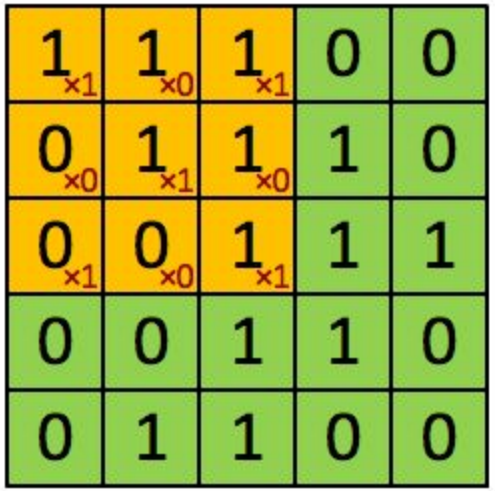
1	0	1
0	1	0
1	0	1

Kernel

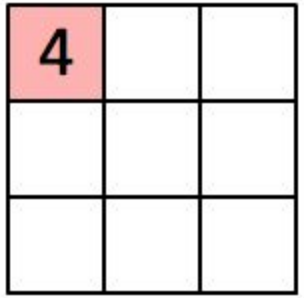
1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Image

Intro to CNN



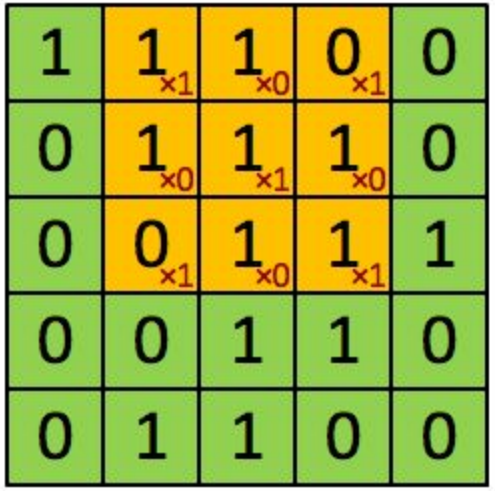
Image



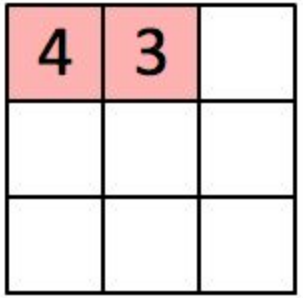
Convolved Feature

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

Intro to CNN



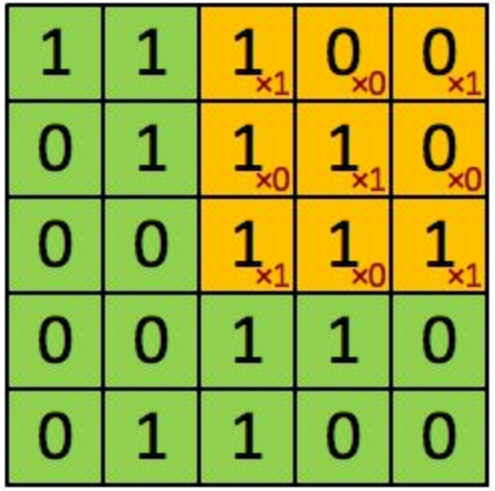
Image



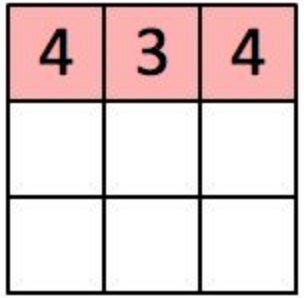
Convolved Feature

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

Intro to CNN



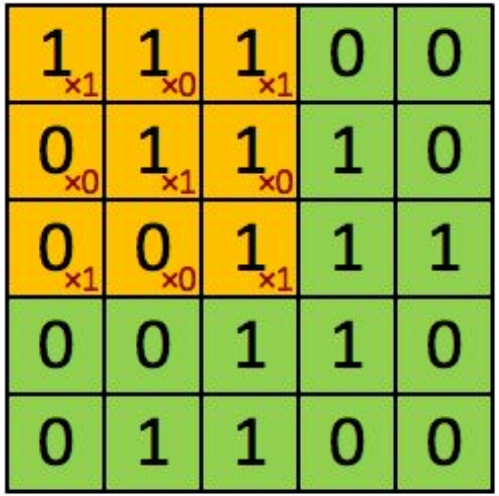
Image



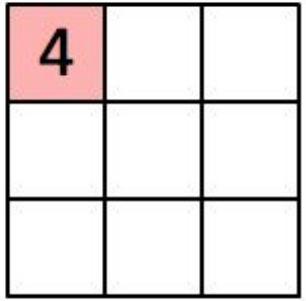
Convolved Feature

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

Intro to CNN



Image

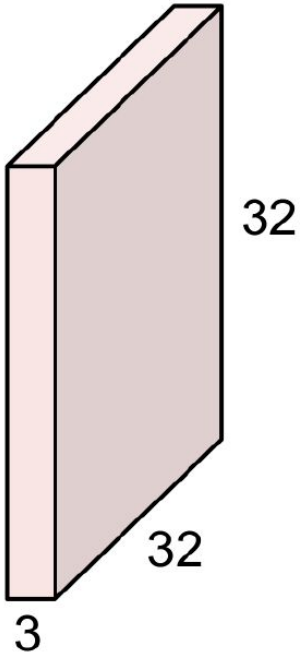


Convolved Feature

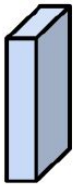
images from http://cs231n.stanford.edu/

Intro to CNN

32x32x3 image



5x5x3 filter

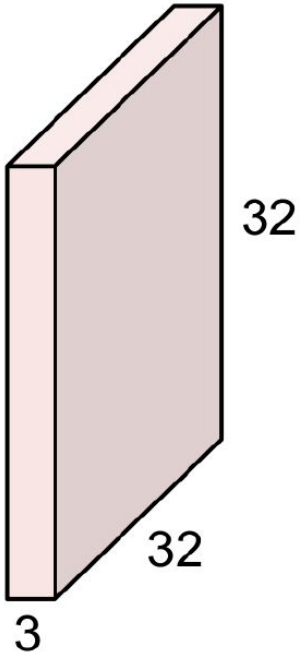


Convolve the filter with the image
i.e. “slide over the image spatially,
computing dot products”

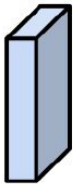
images from http://cs231n.stanford.edu/

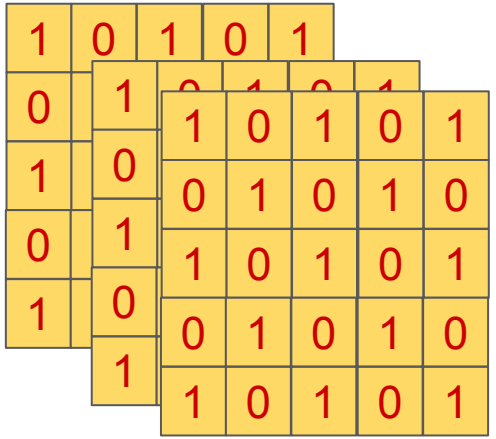
Intro to CNN

32x32x3 image



5x5x3 filter

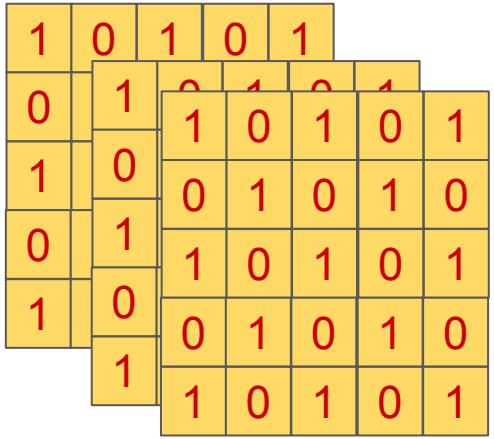


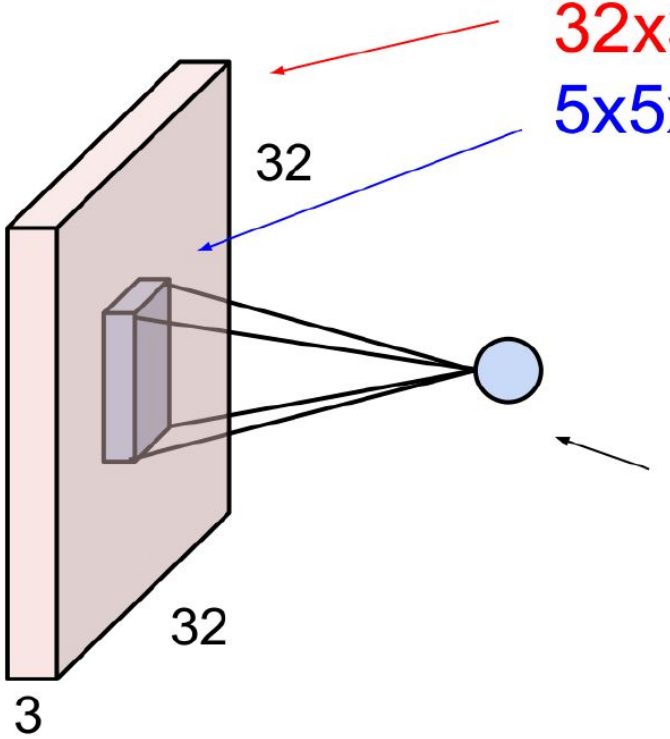


Convolve the filter with the image
i.e. “slide over the image spatially, computing dot products”

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

Intro to 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)

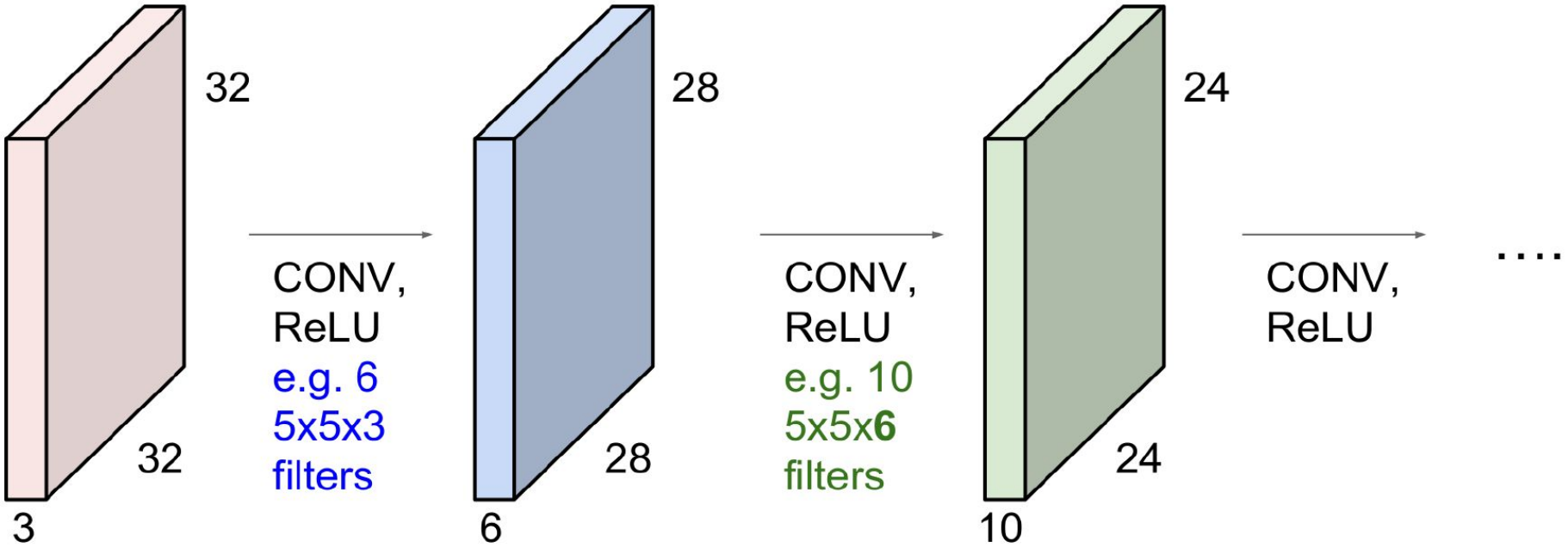
$$w^T x + b$$

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

Intro to CNN

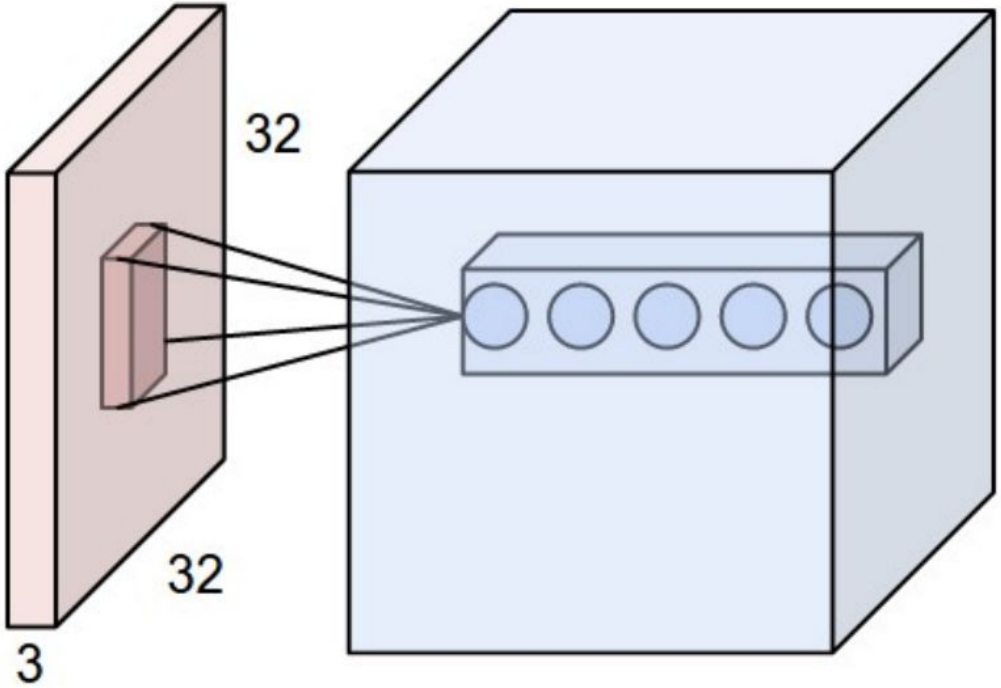
images from http://cs231n.stanford.edu/

Intro to CNN



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

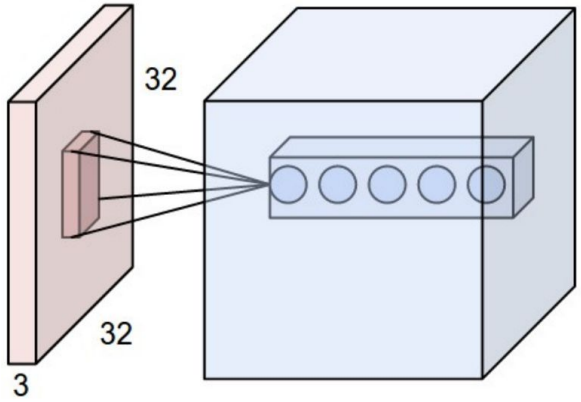
Intro to CNN



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

Intro to CNN

- 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

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

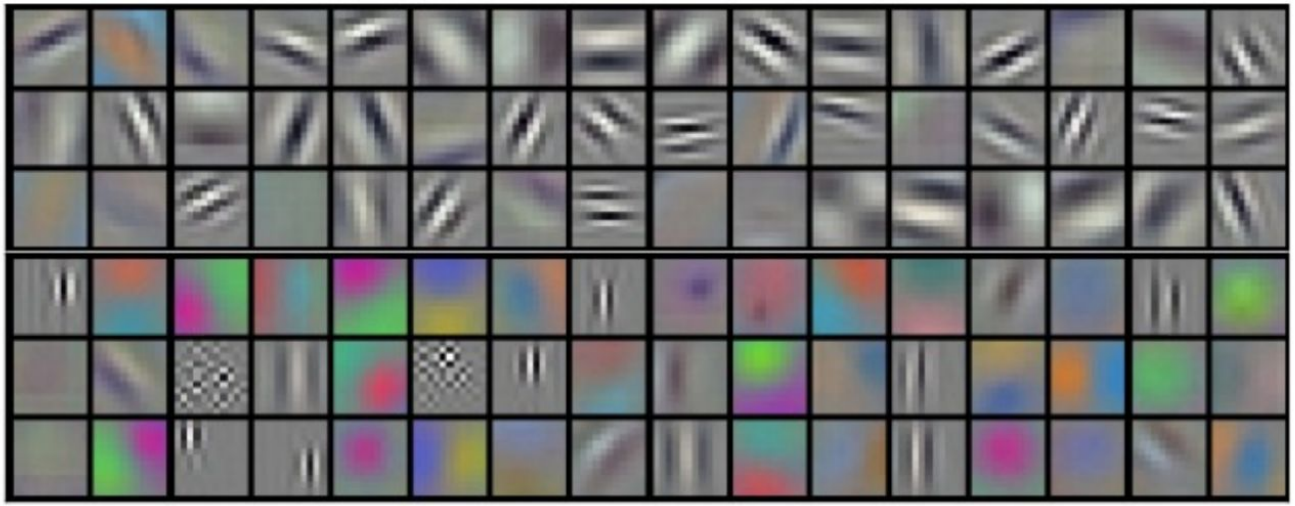
$$\begin{bmatrix} 1. & 1. & 1. \\ 1. & 1. & 1. \\ 1. & 1. & 1. \end{bmatrix}$$

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

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

Intro to CNN

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

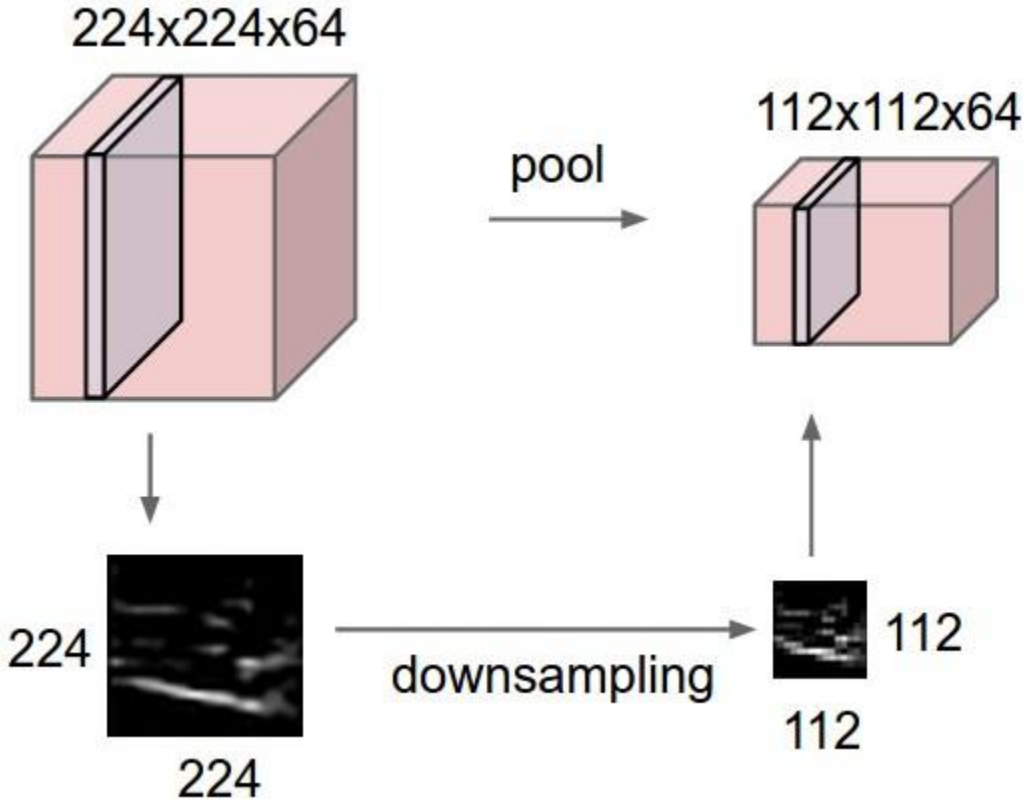


images from http://cs231n.stanford.edu/

Intro to CNN

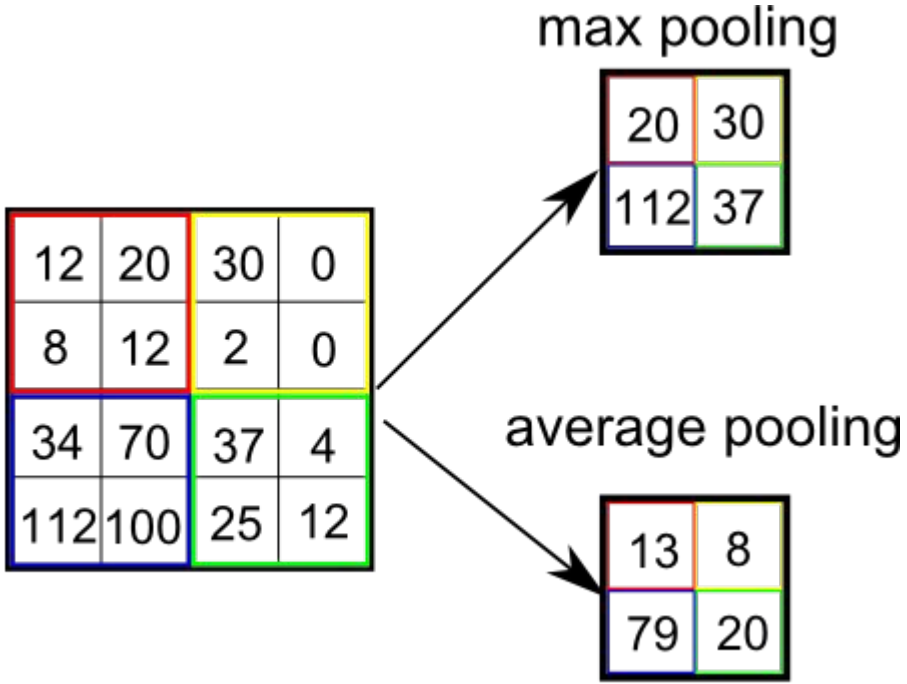
- INPUT holds the raw pixel values of the image.
- CONV layer computes the output of neurons that are connected to local regions in the input, each computing a dot product between their weights and the region they are connected to in the input volume.
- POOL layer performs a downsampling operation along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



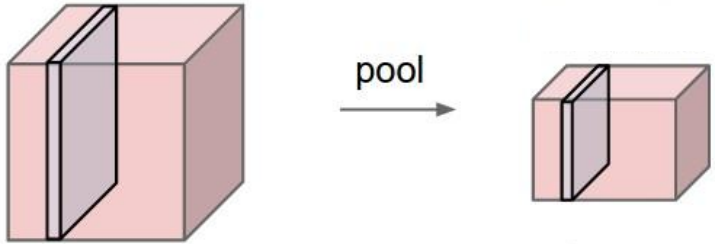
images from http://cs231n.stanford.edu/

Intro to CNN



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

Intro to CNN

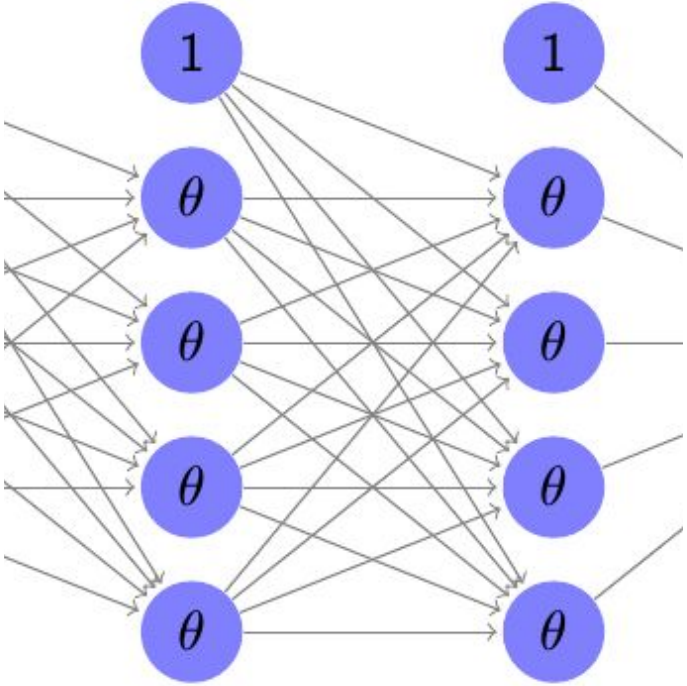


- Accepts a volume of size $W1 \times H1 \times D1$
- Requires three hyperparameters:
  - their spatial extent F,
  - the stride S,
- Produces a volume of size $W2 \times H2 \times D2$ where:
  - $W2 = (W1 - F)/S + 1$
  - $H2 = (H1 - F)/S + 1$
  - $D2 = D1$

Intro to CNN

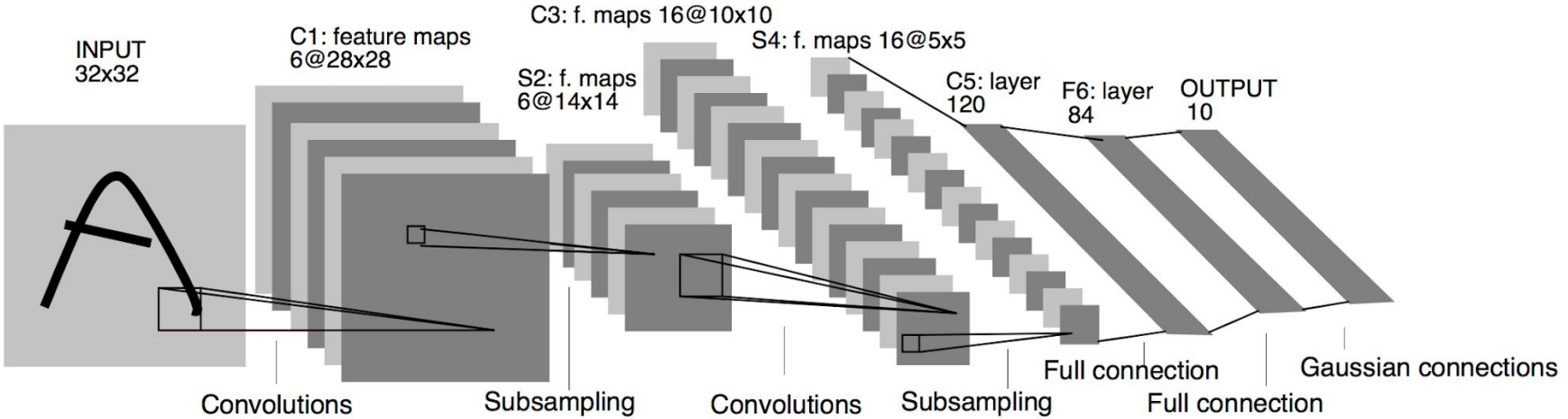
- INPUT holds the raw pixel values of the image.
- CONV layer computes the output of neurons that are connected to local regions in the input, each computing a dot product between their weights and the region they are connected to in the input volume.
- POOL layer performs a downsampling operation along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

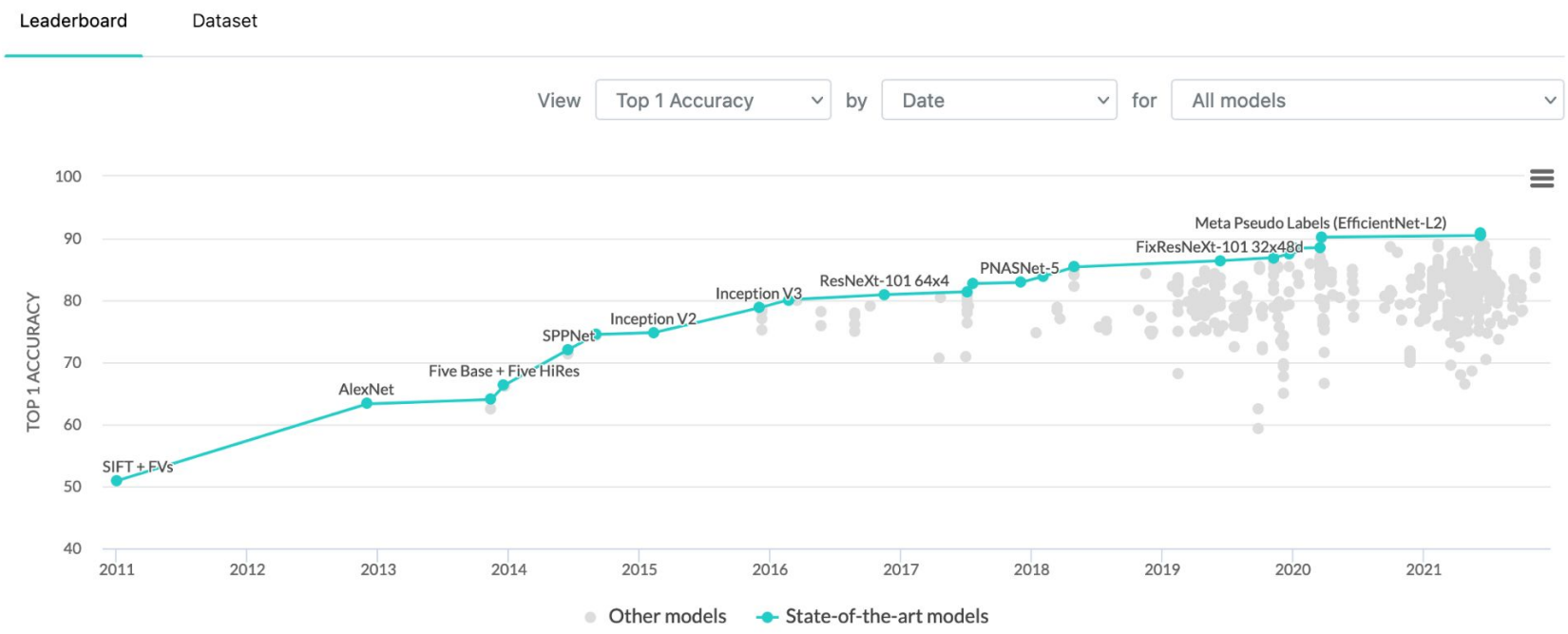


Intro to CNN

CNN explainer

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

Intro to CNN



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

Intro to CNN