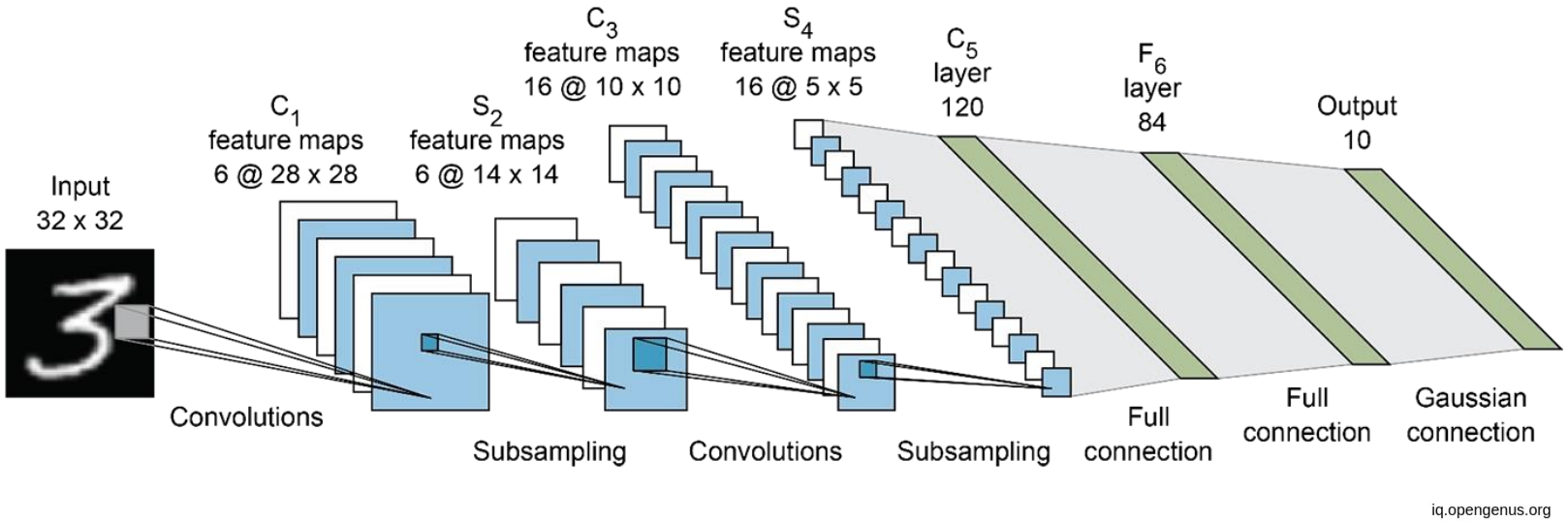
Some of the following slides, marked with “EPFL”, are from the CS-442 course by prof. Pascal Fua

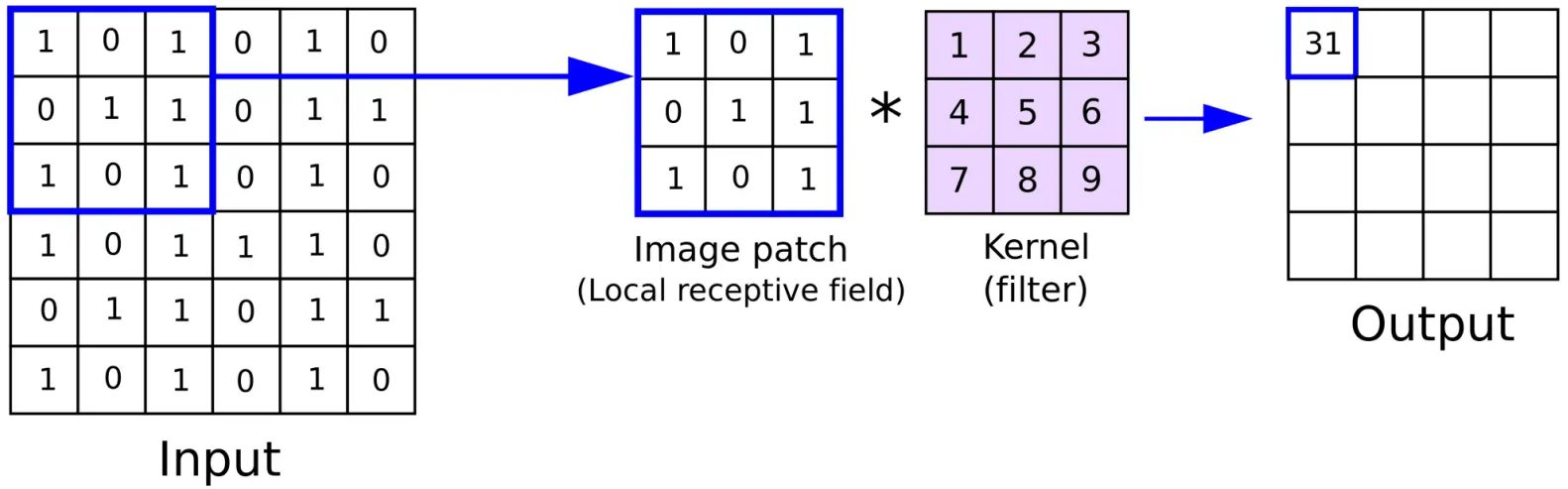
Computer Vision

Federico Stella

Object detection

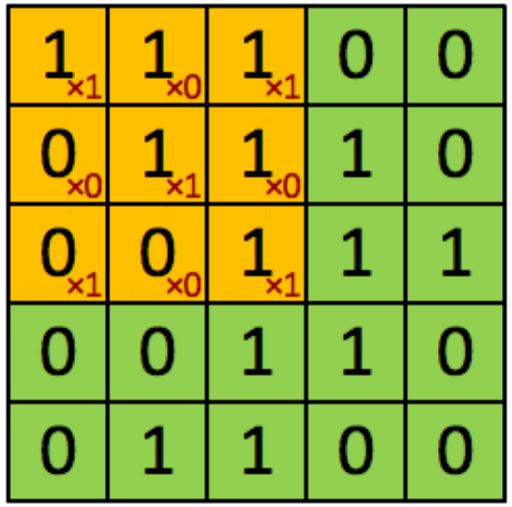
Convolutions

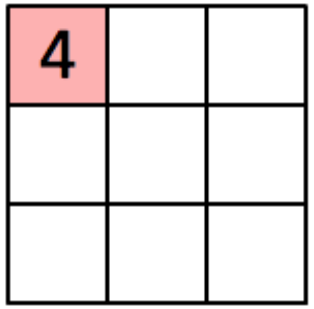




https://www.superannotate.com/blog/guide-to-convolutional-neural-networks

Convolutions*





*In the next slides, I refer to convolutions and correlations interchangeably, since it does not impact the core of the explanations

Convolutions: hand-crafted filters

3x3 Mean filter

$$\begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix} = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

5x5 Mean filter

$$\begin{bmatrix} 1/25 & 1/25 & 1/25 & 1/25 & 1/25 \\ 1/25 & 1/25 & 1/25 & 1/25 & 1/25 \\ 1/25 & 1/25 & 1/25 & 1/25 & 1/25 \\ 1/25 & 1/25 & 1/25 & 1/25 & 1/25 \\ 1/25 & 1/25 & 1/25 & 1/25 & 1/25 \end{bmatrix} = \frac{1}{25}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Mean filter

Original

Corrupted with Gaussian noise

3x3 Mean filter

5x5 Mean filter

Mean filter

Original

Corrupted with impulse noise

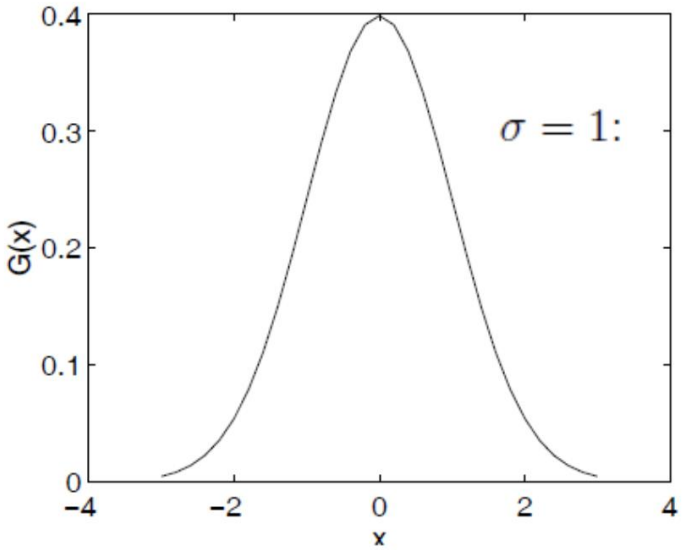
3x3 Mean filter

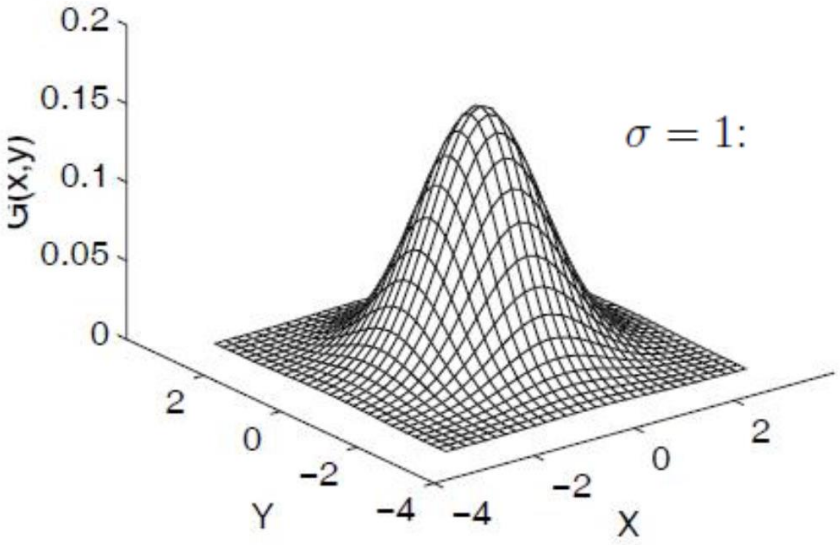
5x5 Mean filter

Other filters? Gaussian

$$G(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}}$$

$$G(x,y) = G(x)G(y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

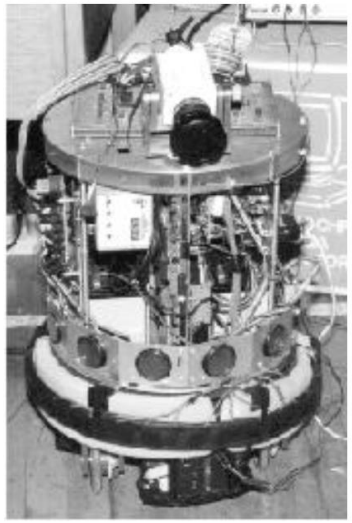




What does it do?

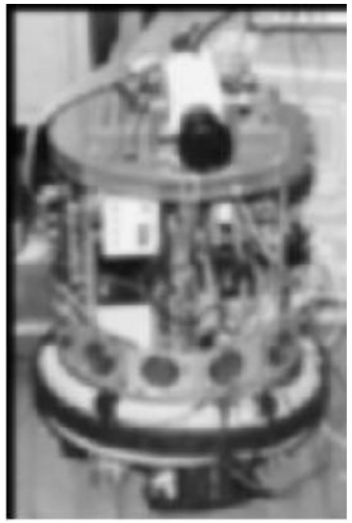
Gaussian filter: smoothing

Original



Sigma 1

Sigma 2



Sigma 4

Non-linear filters? Median

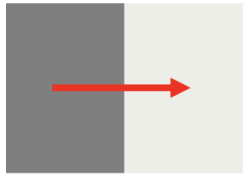
Original

Corrupted with impulse noise

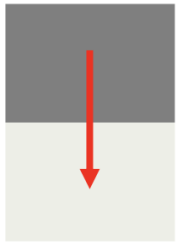
Median 3x3

Median 3x3 twice

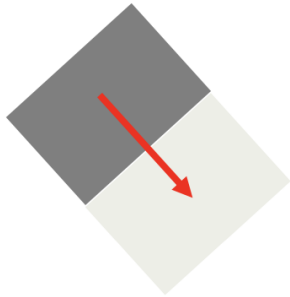
Edge detection



Vertical Edge



Horizontal Edge



Diagonal Edge

$$\nabla I(x,y) = \frac{\partial I(x,y)}{\partial x}\mathbf{i} + \frac{\partial I(x,y)}{\partial y}\mathbf{j}$$

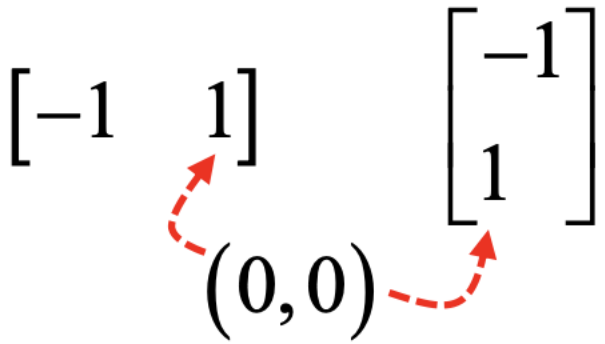
Finite-difference as a filter

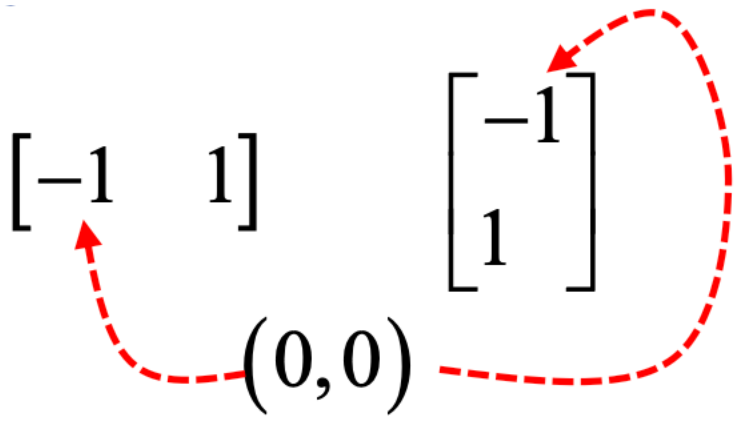
We can approximate the gradient as forward or backward differences

$$\frac{\partial I(x,y)}{\partial x} \cong I_x(i,j) = I(i,j) - I(i,j-1), \quad \frac{\partial I(x,y)}{\partial y} \cong I_y(i,j) = I(i,j) - I(i-1,j)$$

$$\frac{\partial I(x,y)}{\partial x} \cong I_x(i,j) = I(i,j+1) - I(i,j), \quad \frac{\partial I(x,y)}{\partial y} \cong I_y(i,j) = I(i+1,j) - I(i,j)$$

Which can be reformulated as correlation kernels





Finite-difference as a filter

Central differences are also possible

$$I_x(i,j) = I(i,j+1) - I(i,j-1), \qquad I_y(i,j) = I(i+1,j) - I(i-1,j)$$

$$\begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \quad (0,0) \quad \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$$

Smoother gradient: mean filter

We can apply a mean filter in the perpendicular direction compared to the one of the derivative

$$\frac{1}{3}\begin{bmatrix} -1 & 1 \\ -1 & 1 \\ -1 & 1 \end{bmatrix} \qquad \frac{1}{3}\begin{bmatrix} -1 & -1 & -1 \\ 1 & 1 & 1 \end{bmatrix}$$

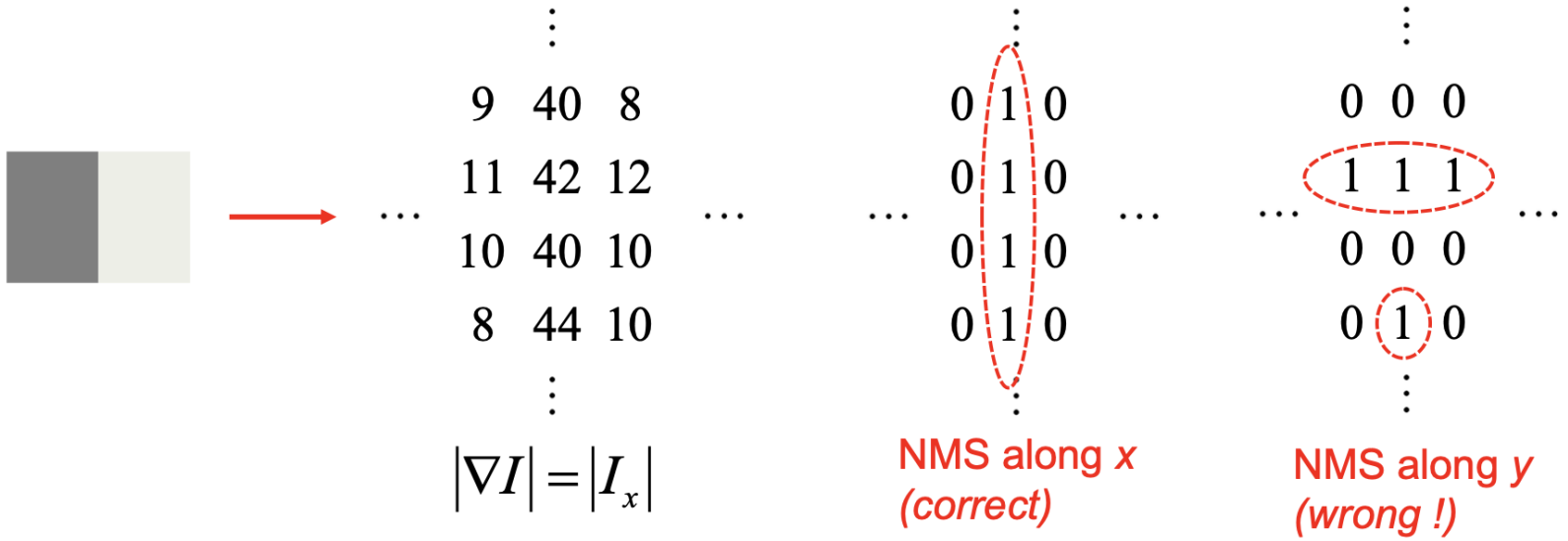
$$\frac{1}{3}\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \qquad \frac{1}{3}\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

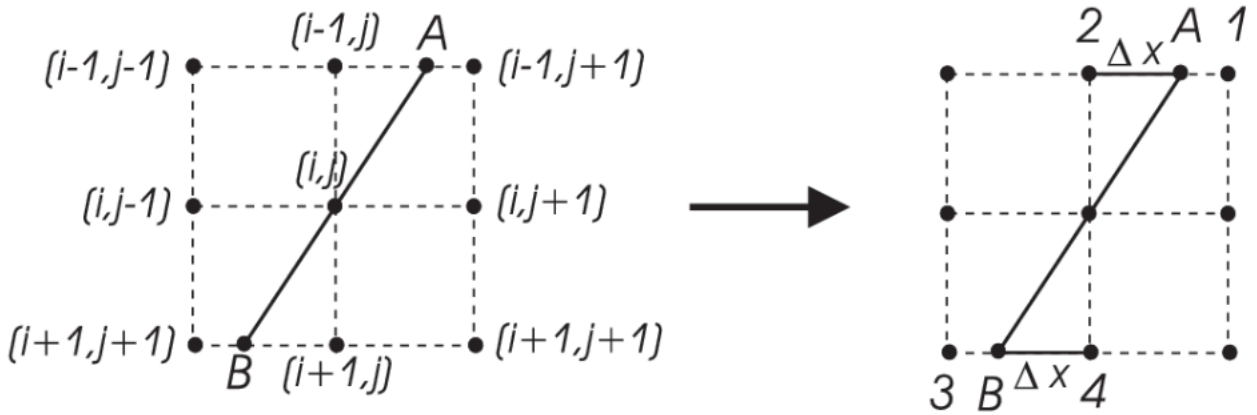
Prewitt operator

$$\frac{1}{4}\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad \frac{1}{4}\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Sobel operator
(weighs the central contribution more)

Non-maxima suppression

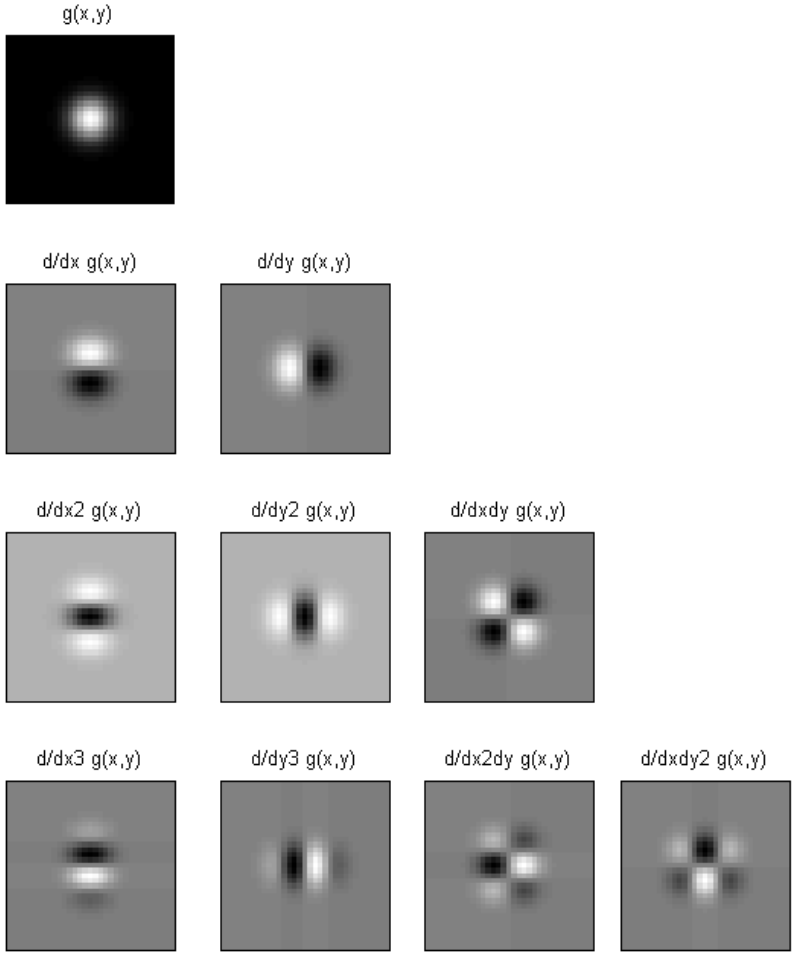




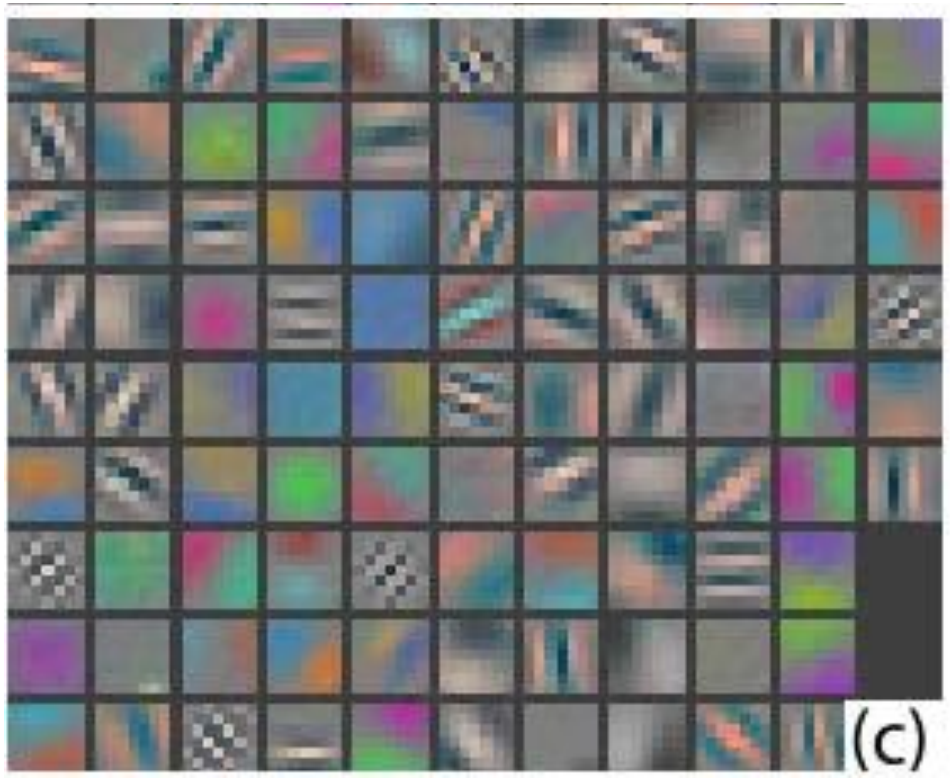
Sobel operator

https://towardsdatascience.com/magic-of-the-sobel-operator-bbbcb15af20d

Spoilers on CNN filters



Derivatives



Learned filters

CNN filters oftentimes learn to spot edges and similar features

Hough transform

(for lines)

$$y - mx - c = 0$$

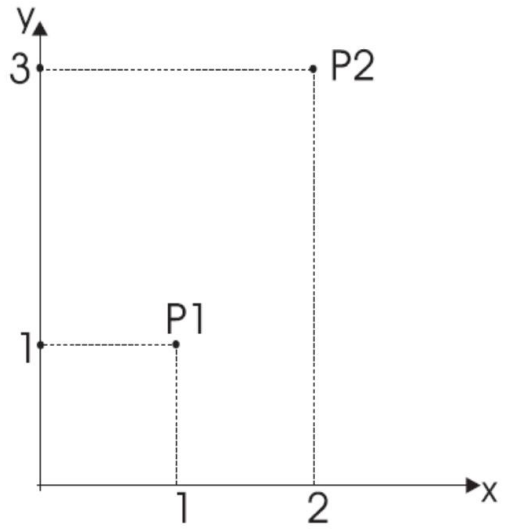
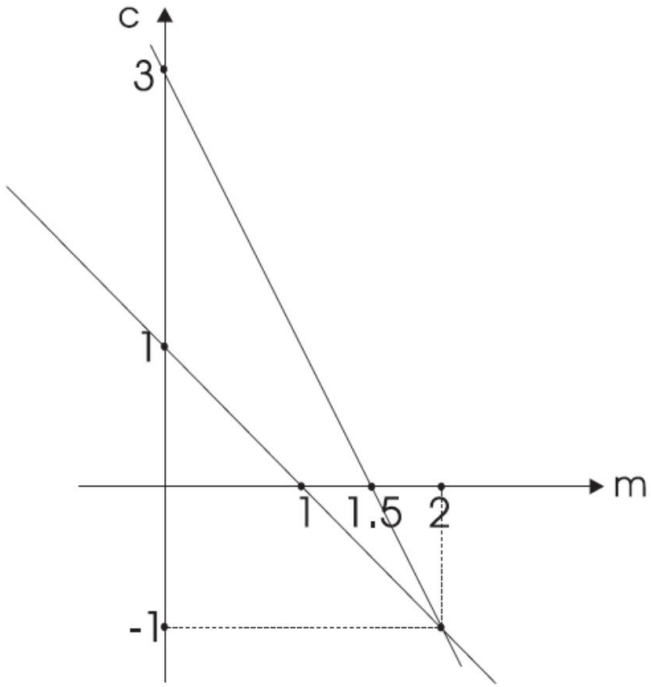


Image Space



Parameter Space

Hough transform

(for lines)

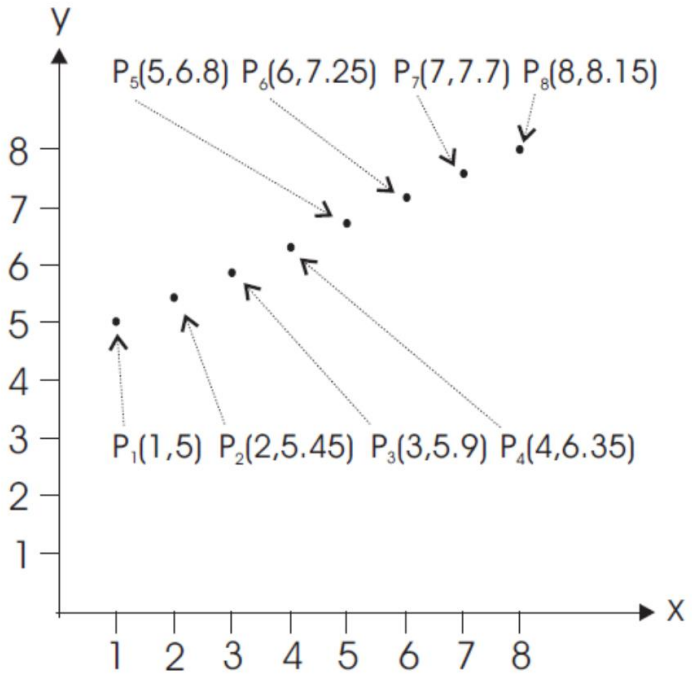
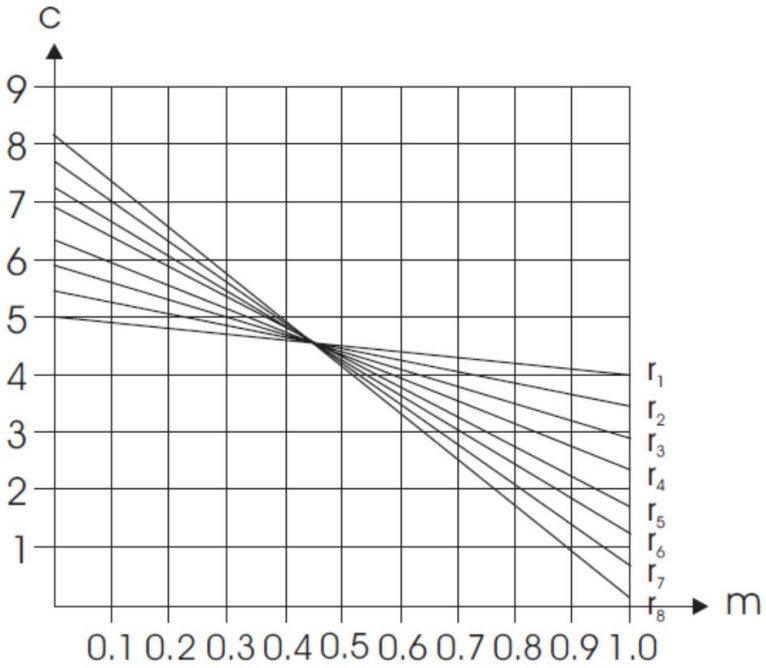


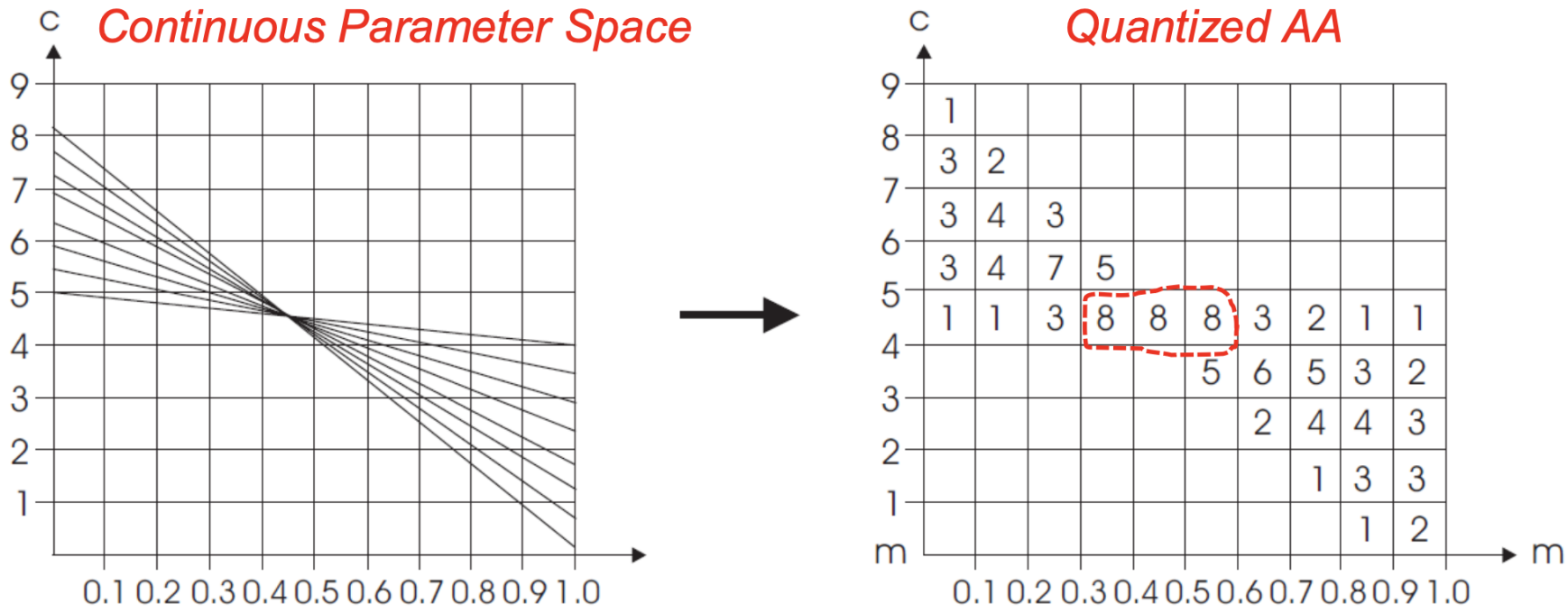
Image Space



Parameter Space

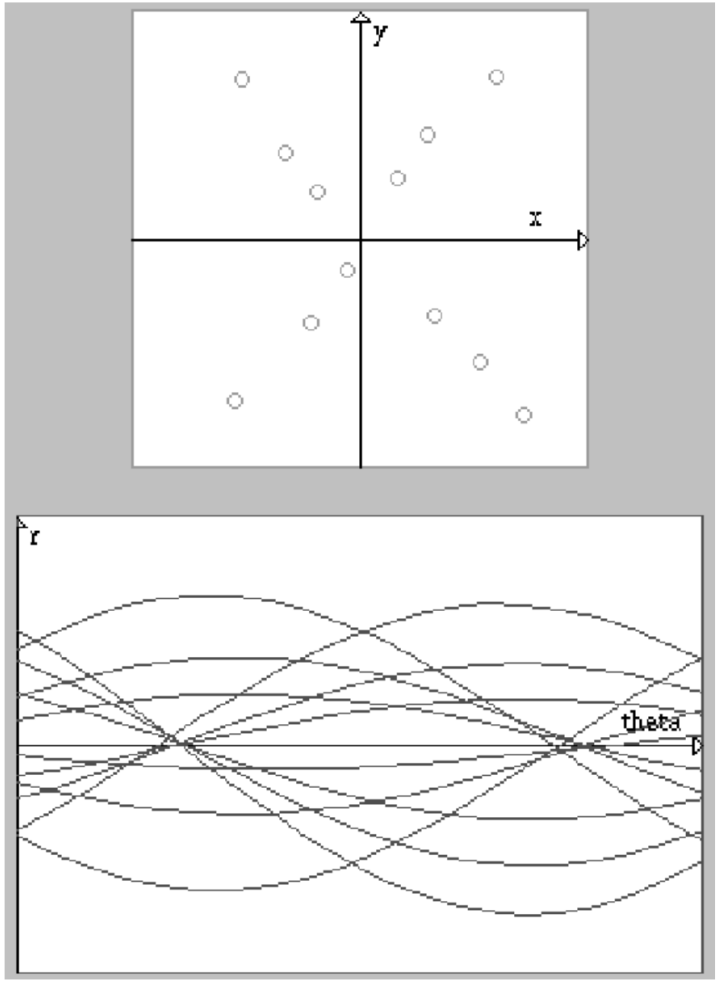
Hough transform

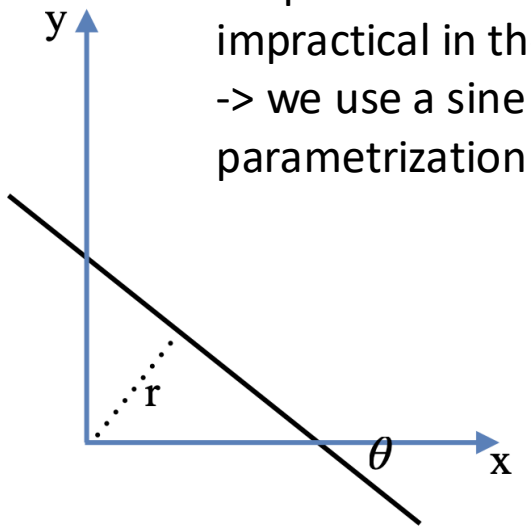
(for lines)



Hough transform

(for lines)

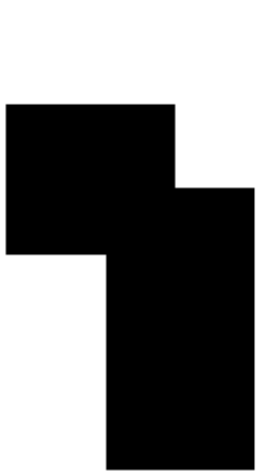


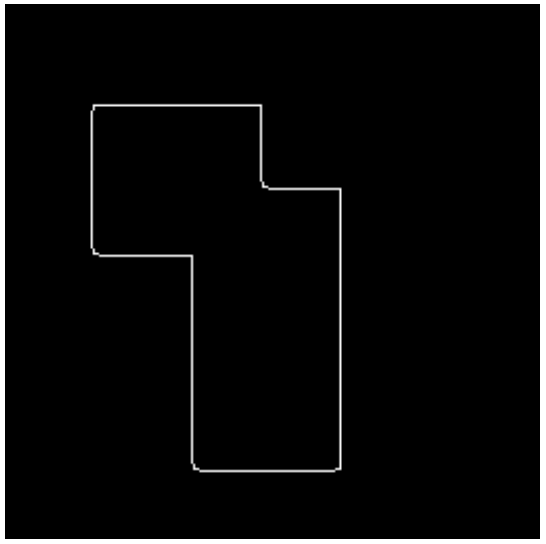


m spans an infinite range, so it's impractical in the accumulator array -> we use a sine and cosine parametrization

$$x\cos(\theta) + y\sin(\theta) = r \ , \ 0 \leq \theta \leq \pi$$

Synthetic Lines





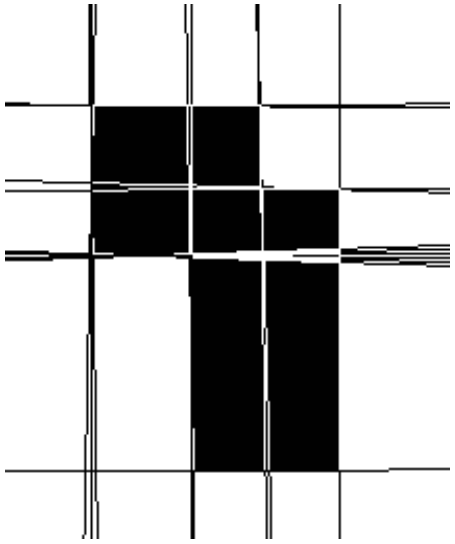
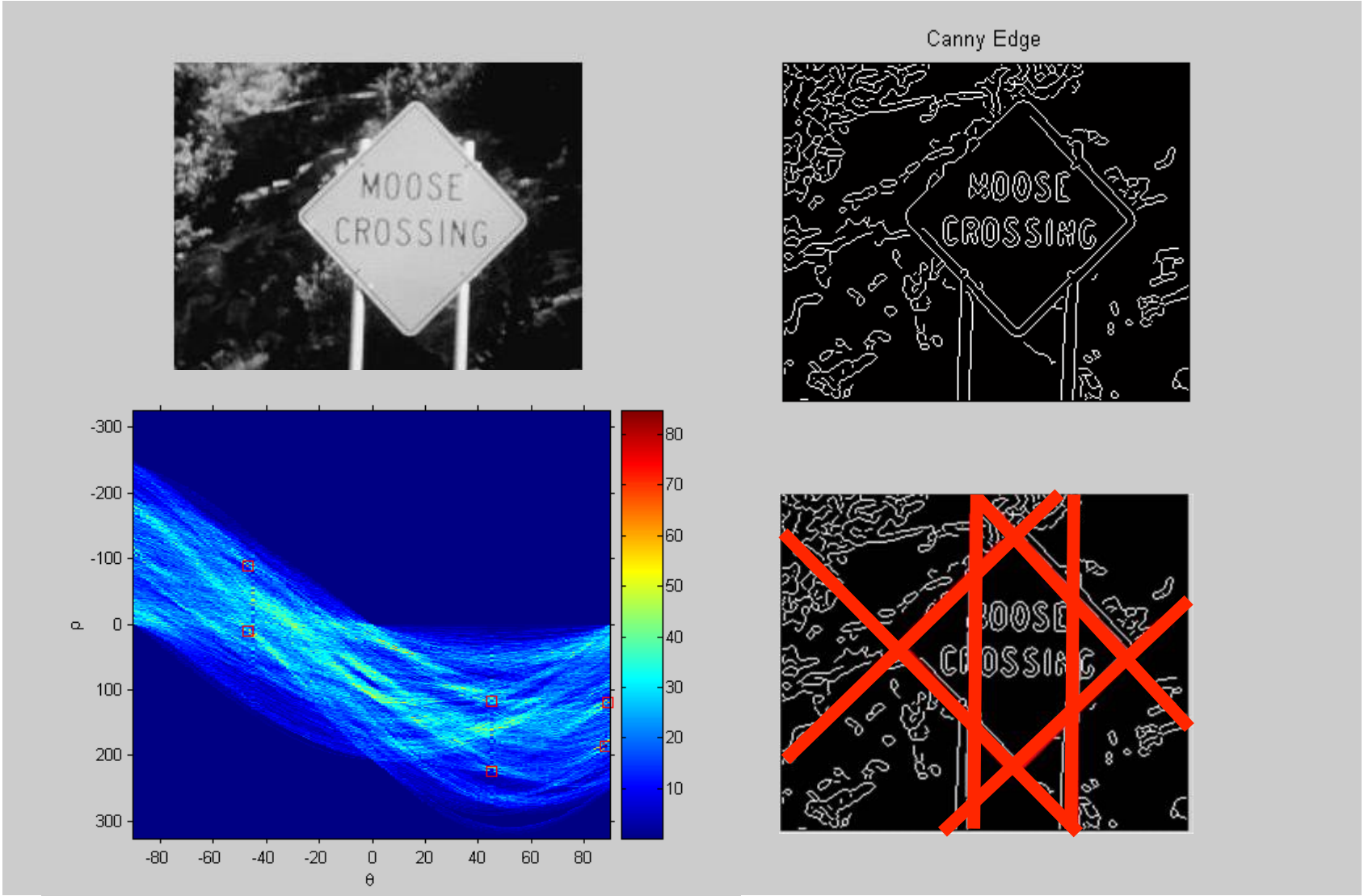


Image Contours Accumulator Lines

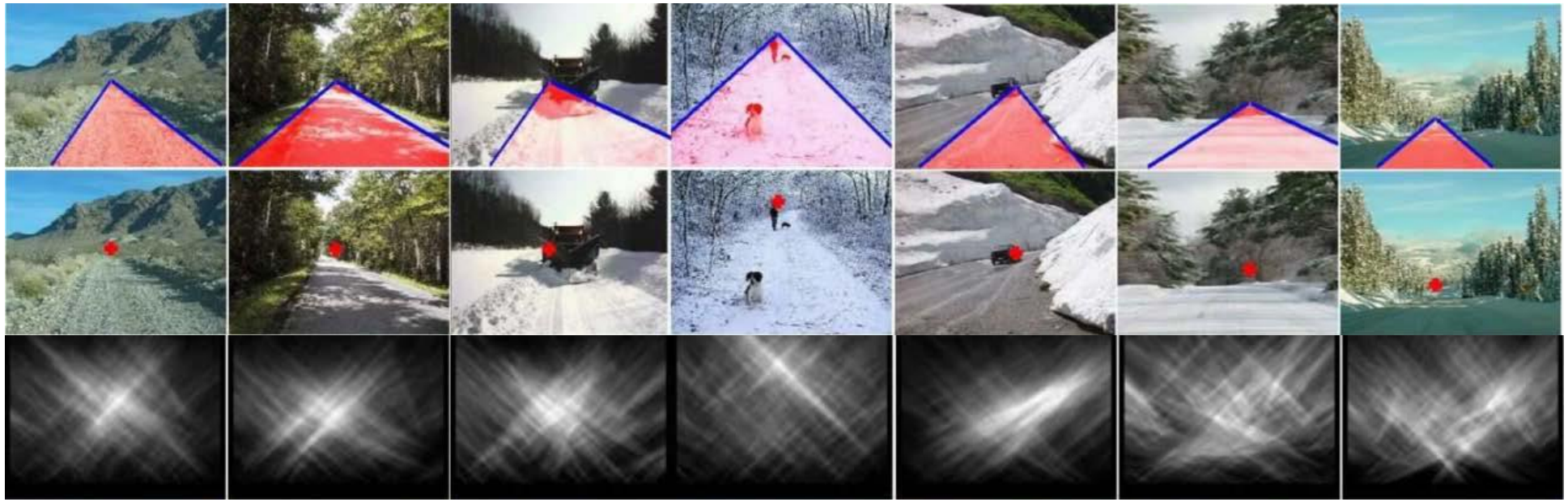
Once the contour points are associated to individual lines, you can perform least squares fitting.

Real Lines


MOOSE
CROSSING
Canny Edge
MOOSE
CROSSING
-300
-200
-100
0
100
200
300
ρ
-80
-60
-40
-20
0
20
40
60
80
θ
80
70
60
50
40
30
20
10

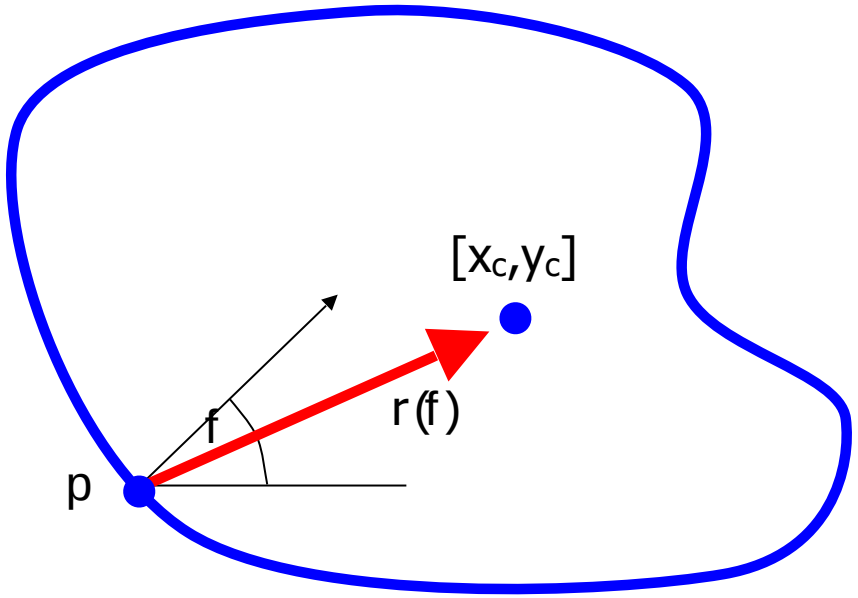
Road Lines

Road Edges



Kong et al. CVPR'09

Generalized Hough

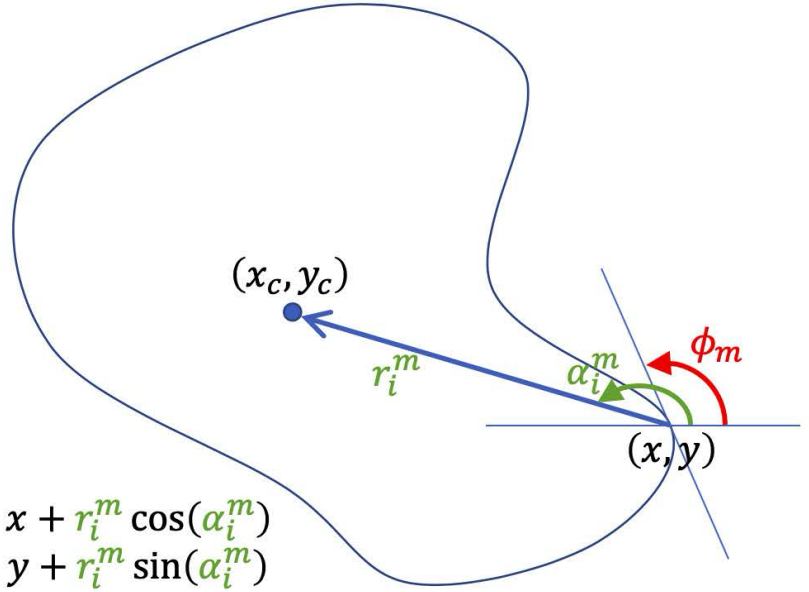


- We want to find a shape defined by its boundary points in terms of the location of a reference point $[x_c,y_c]$.
- For every boundary point p, we can compute the displacement vector $r = [x_c,y_c] – p$ as a function of local gradient orientation f.

Ballard. , PR'81

R-Table

ϕ	$R(\phi_i)$
ϕ_1	$(r_1^1, \alpha_1^1), (r_2^1, \alpha_2^1), \ldots\ldots (r_{n1}^1, \alpha_{n1}^1)$
ϕ_2	$(r_1^2, \alpha_1^2), (r_2^2, \alpha_2^2), \ldots\ldots (r_{n2}^2, \alpha_{n2}^2)$
..	.....
ϕ_m	$(r_1^m, \alpha_1^m), (r_2^m, \alpha_2^m), \ldots\ldots (r_i^m, \alpha_i^m) \ldots\ldots (r_{nm}^m, \alpha_{nm}^m)$
..	.....
ϕ_M	$(r_1^M, \alpha_1^M), (r_2^M, \alpha_2^M), \ldots\ldots (r_{nM}^M, \alpha_{nM}^M)$

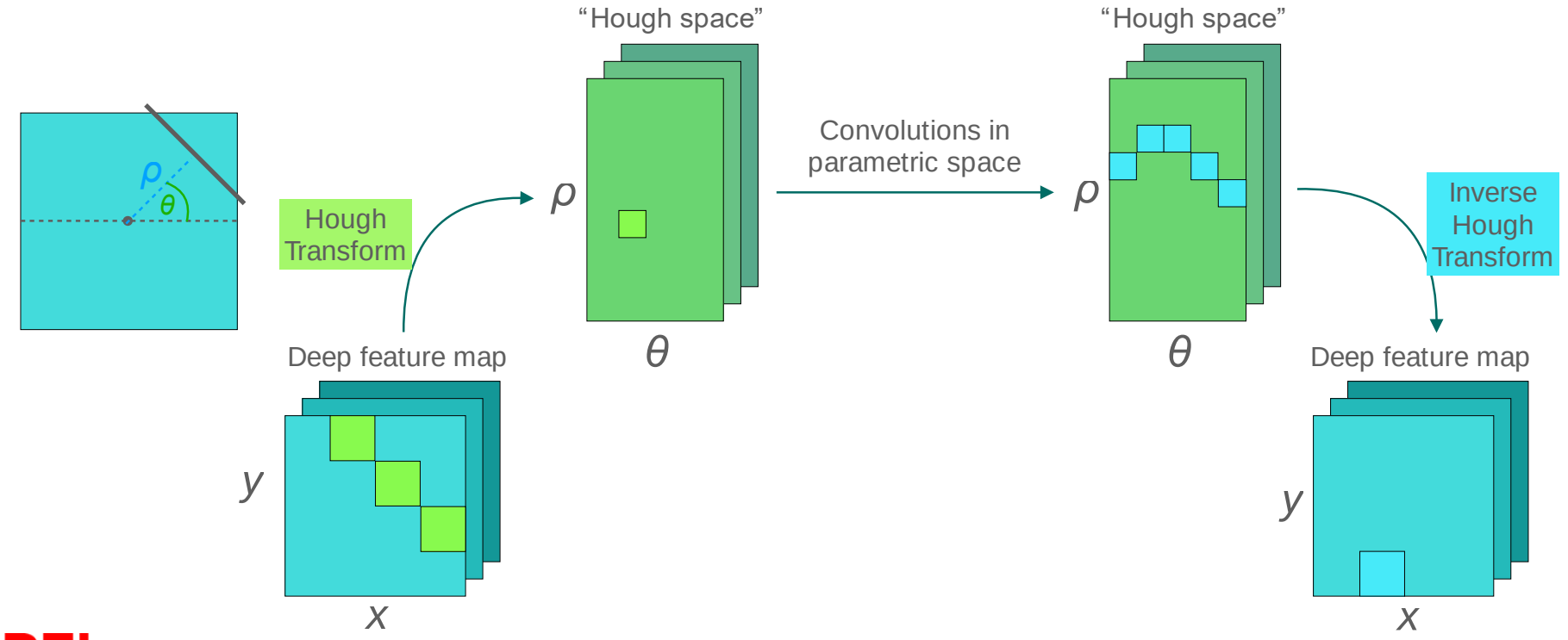


$$x_c = x + r_i^m \cos(\alpha_i^m)$$
$$y_c = y + r_i^m \sin(\alpha_i^m)$$

Set of potential displacement vectors r,a given the boundary orientation f.

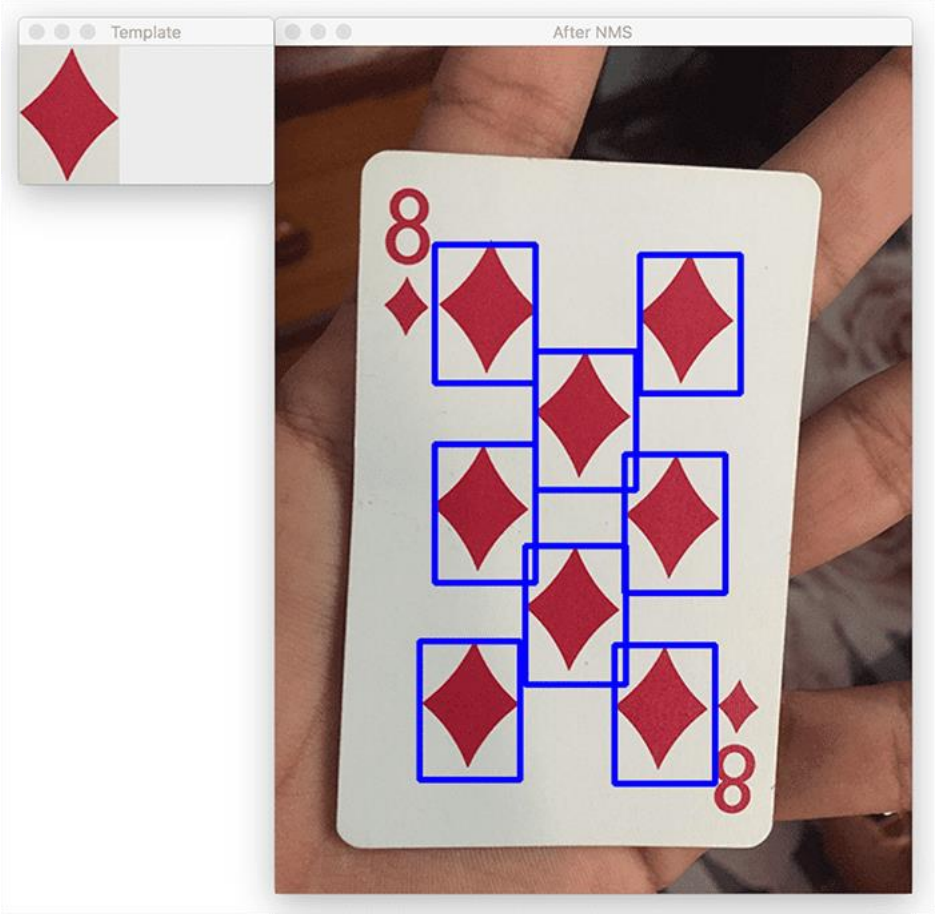
--> Generalized template matching.

Deep Hough Transform



Zhao, PAMI'21

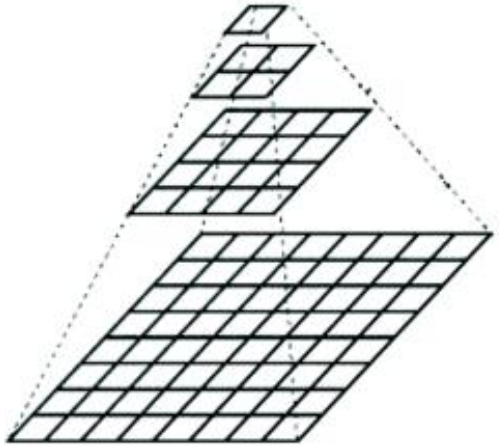
Early object detection: template matching

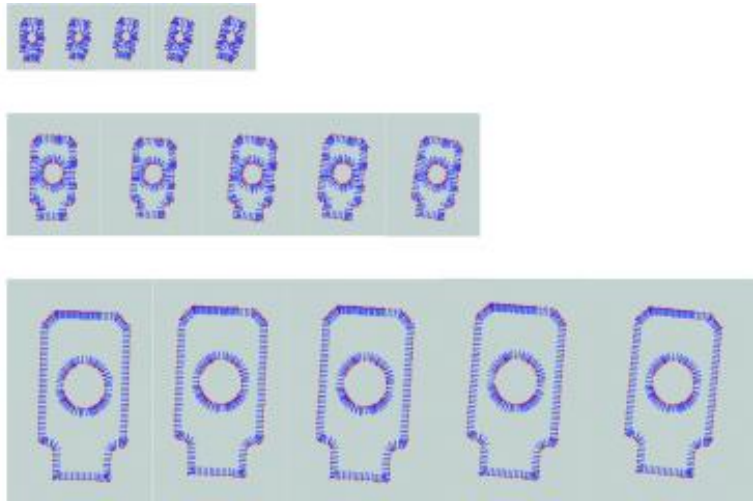


It's a "manually defined" convolution operation

https://pyimagesearch.com/2021/03/29/multi-template-matching-with-opencv/

Fast template matching with image pyramid





Full search is only performed at the top level, the subsequent levels refine the location

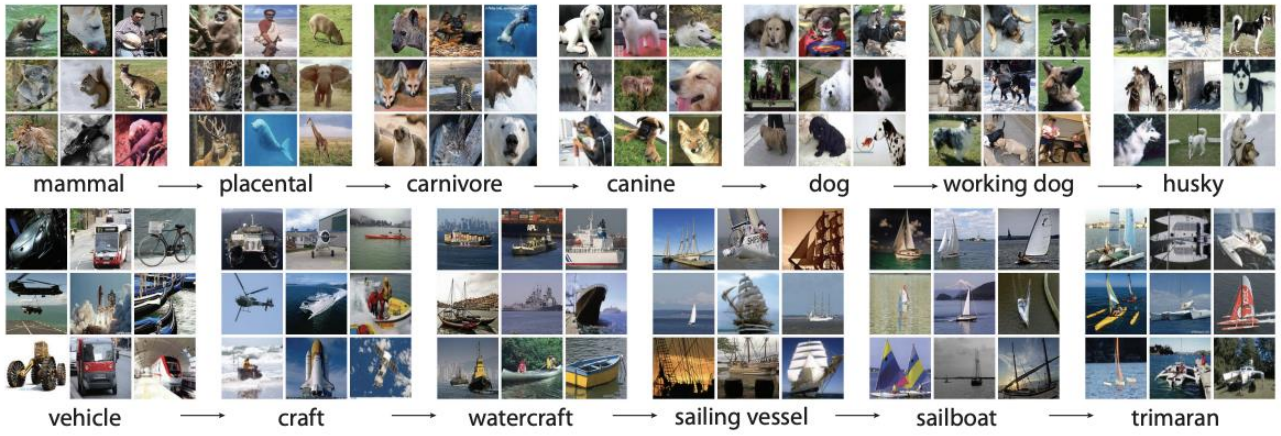
https://link.springer.com/chapter/10.1007/978-3-030-27538-9_43

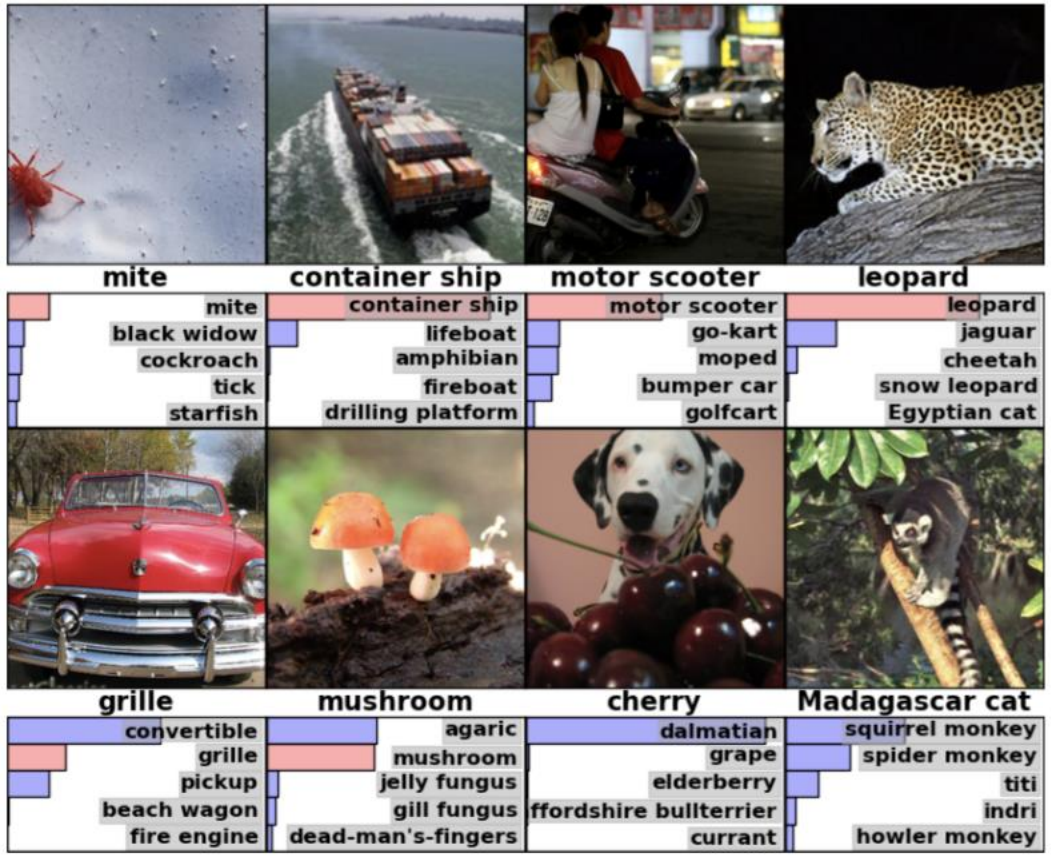
But wait, neural networks can do convolution!

Instead of manually defining templates and filters, we can let backpropagation do its job in CNNs

ImageNet
Large Scale Visual Recognition Challenge

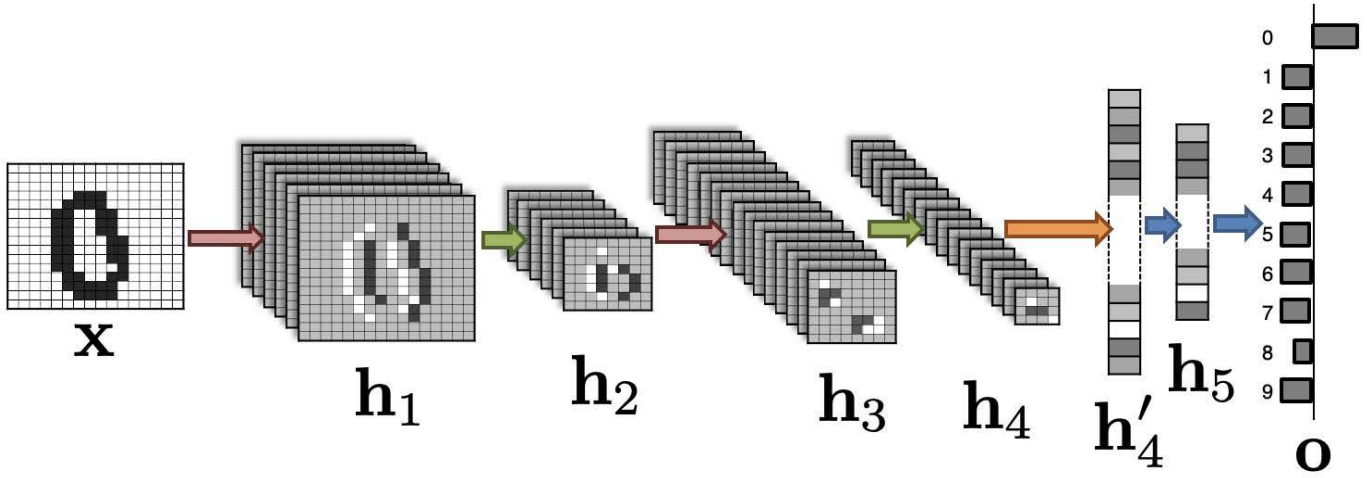
(Deng Ji et al, CVPR 2009)

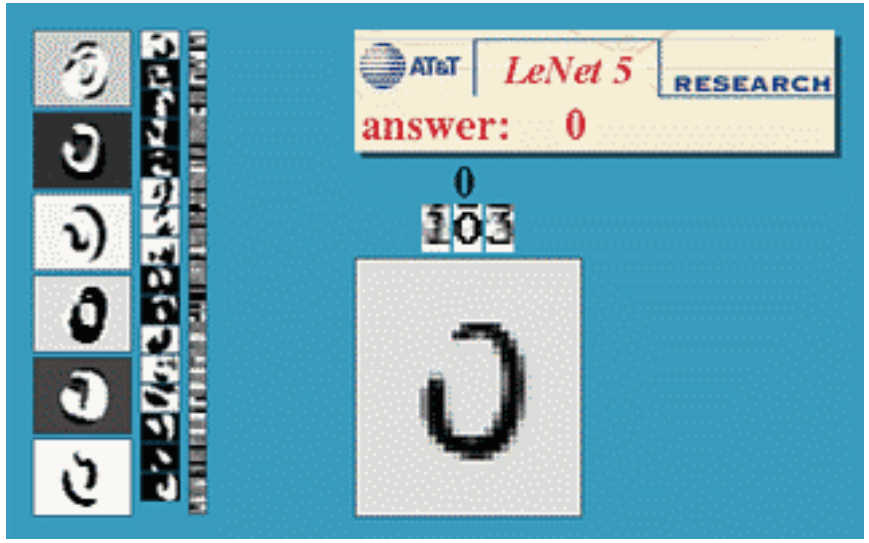




Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, L. D. Jackel, Backpropagation Applied to Handwritten Zip Code Recognition, AT&T Bell Laboratories, 1989

LeNet (1989-1999)

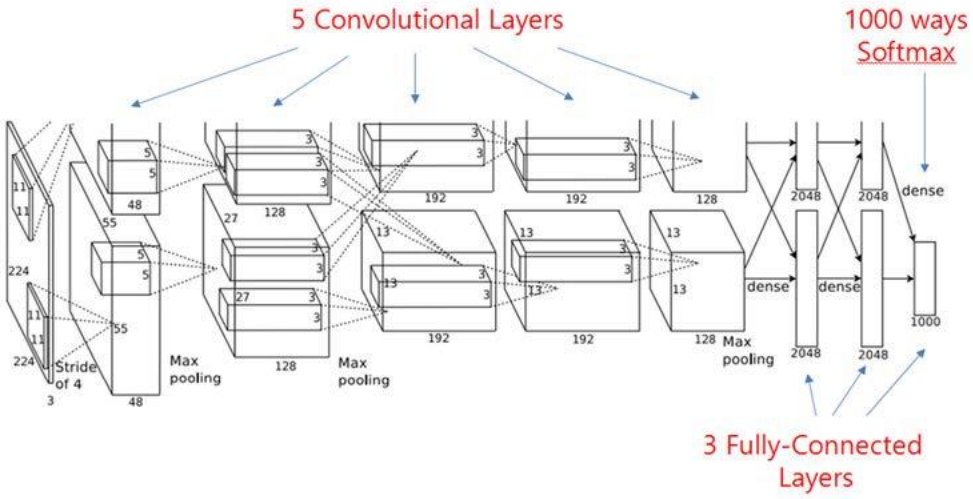




Applied backpropagation to convolutional layers and started obtaining interesting results, but too slow for large scale!

Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, NeurIPS 2012

AlexNet (2012)



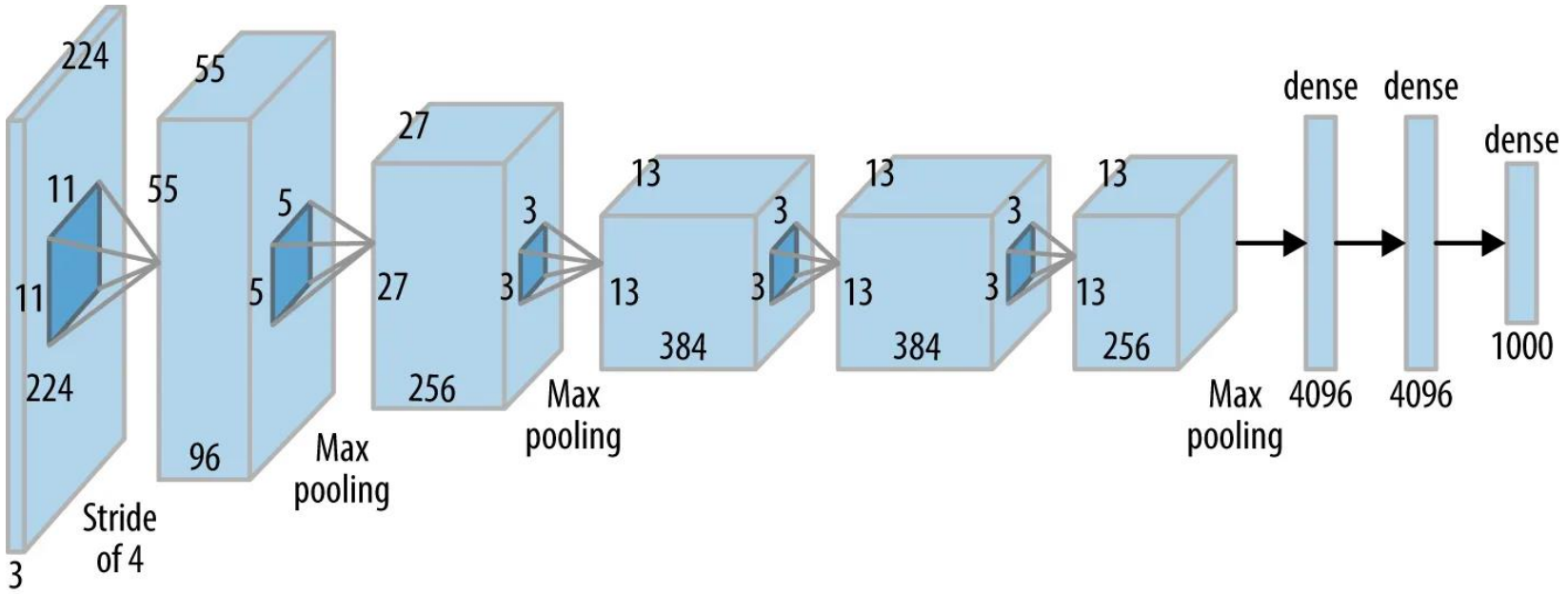
Task: Image classification

Training images: Large Scale Visual Recognition Challenge 2010

Training time: 2 weeks on 2 GPUs

Major Breakthrough: Training large networks has now been shown to be practical!!

AlexNet (2012)



They absolutely smashed existing methods, achieving a top-5 error of ~15% (down from ~25% the year before)
They did not invent something particularly new, the piece were already there, but they created a fast GPU implementation of CNNs, which made the lengthy training possible. An AI revolution.

Relatively deep architecture, achieved using ReLUs, to avoid the vanishing gradient problem

AlexNet = CNN + backprop + GPU + ReLU + maxpool + dropout

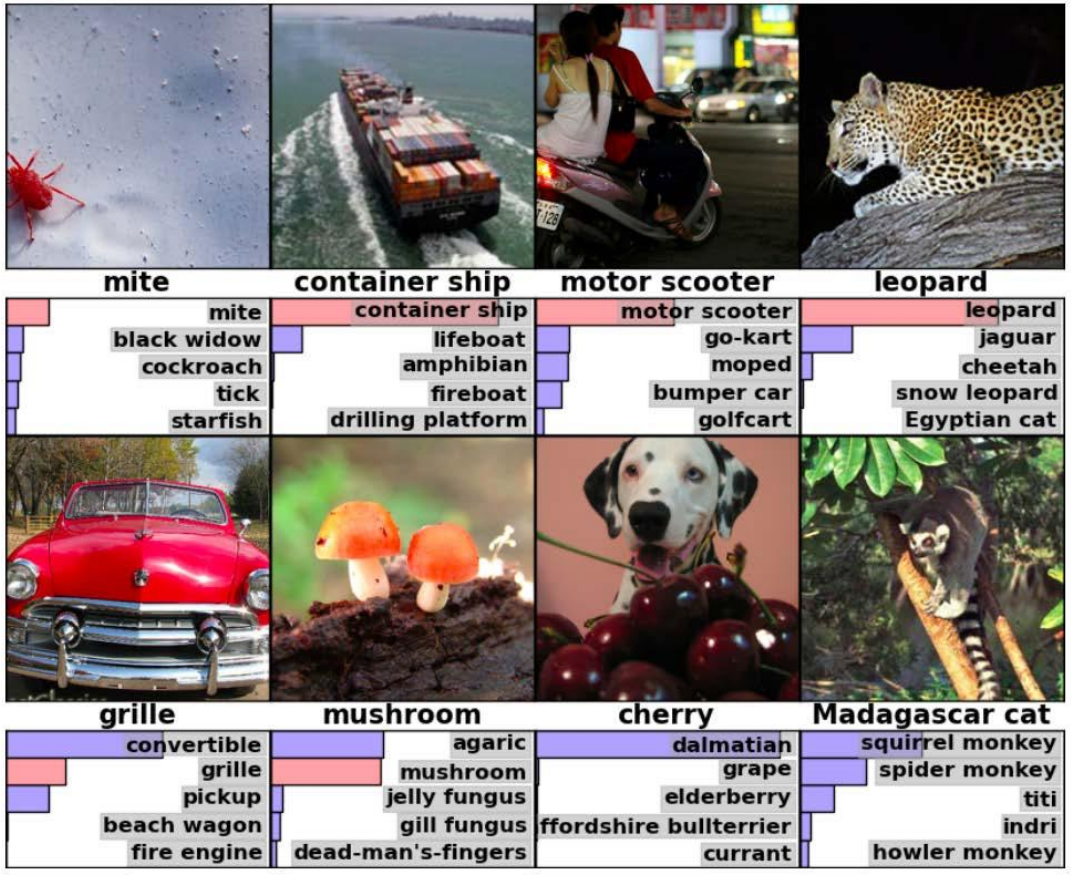
https://medium.com/analytics-vidhya/concept-of-alexnet-convolutional-neural-network-6e73b4f9ee30

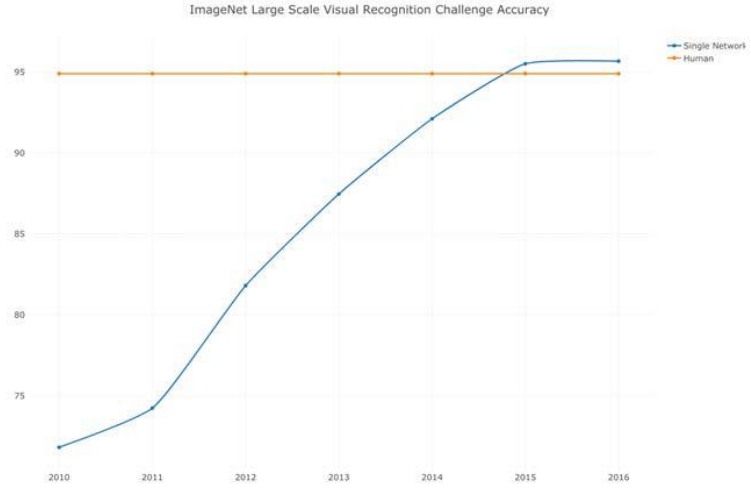
AlexNet (2012)

AlexNet Network - Structural Details												
Input			Output			Layer	Stride	Pad	Kernel size		in	out	# of Param
227	227	3	55	55	96	conv1	4	0	11	11	3	96	34944
55	55	96	27	27	96	maxpool1	2	0	3	3	96	96	0
27	27	96	27	27	256	conv2	1	2	5	5	96	256	614656
27	27	256	13	13	256	maxpool2	2	0	3	3	256	256	0
13	13	256	13	13	384	conv3	1	1	3	3	256	384	885120
13	13	384	13	13	384	conv4	1	1	3	3	384	384	1327488
13	13	384	13	13	256	conv5	1	1	3	3	384	256	884992
13	13	256	6	6	256	maxpool5	2	0	3	3	256	256	0
						fc6			1	1	9216	4096	37752832
						fc7			1	1	4096	4096	16781312
						fc8			1	1	4096	1000	4097000
Total													**62,378,344**

https://medium.com/analytics-vidhya/concept-of-alexnet-convolutional-neural-network-6e73b4f9ee30

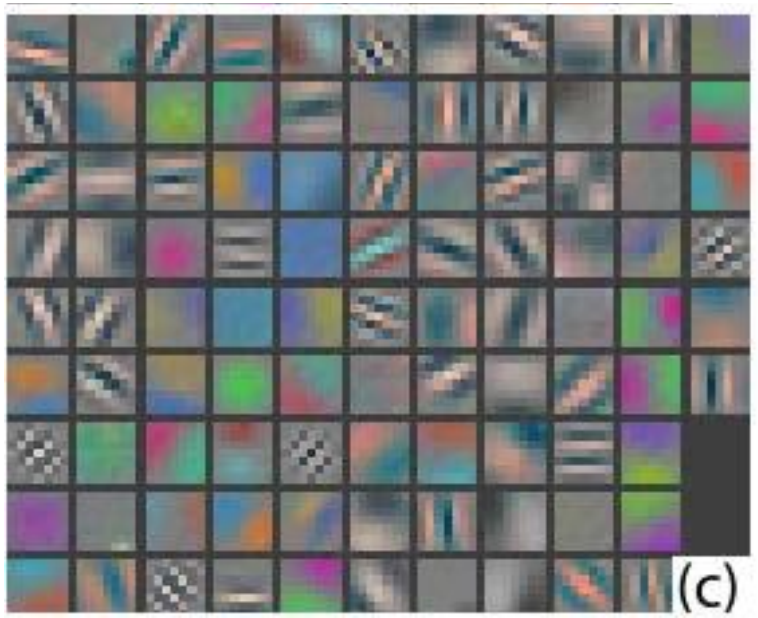
AlexNet Results



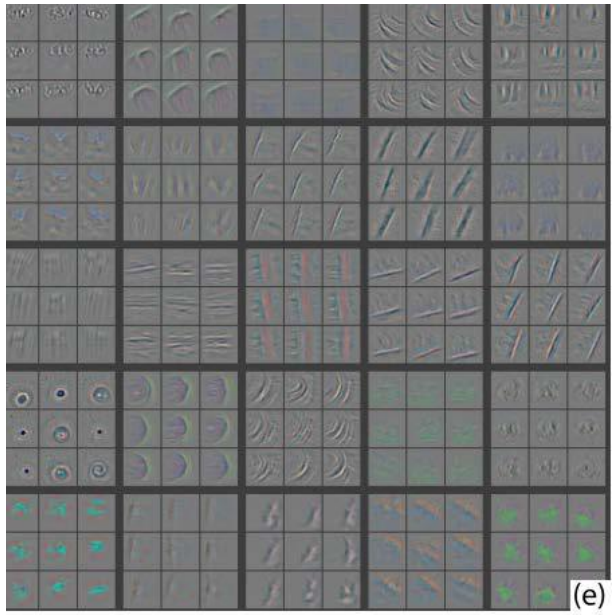


- At the 2012 ImageNet Large Scale Visual Recognition Challenge, AlexNet achieved a top-5 error of 15.3%, more than 10.8% lower than the runner up.
- Since 2015, networks outperform humans on this task.

Feature Maps



First convolutional layer

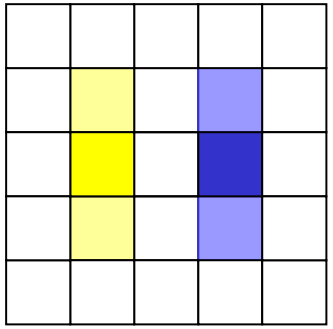


Second convolutional layer

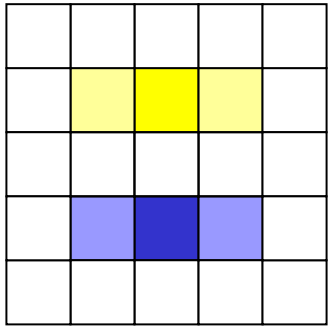
- The trained neural nets compute oriented derivatives, which the brain is also **believed** to do.

Reminder: 3X3 Masks

x derivative



y derivative



$$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \text{ and } \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

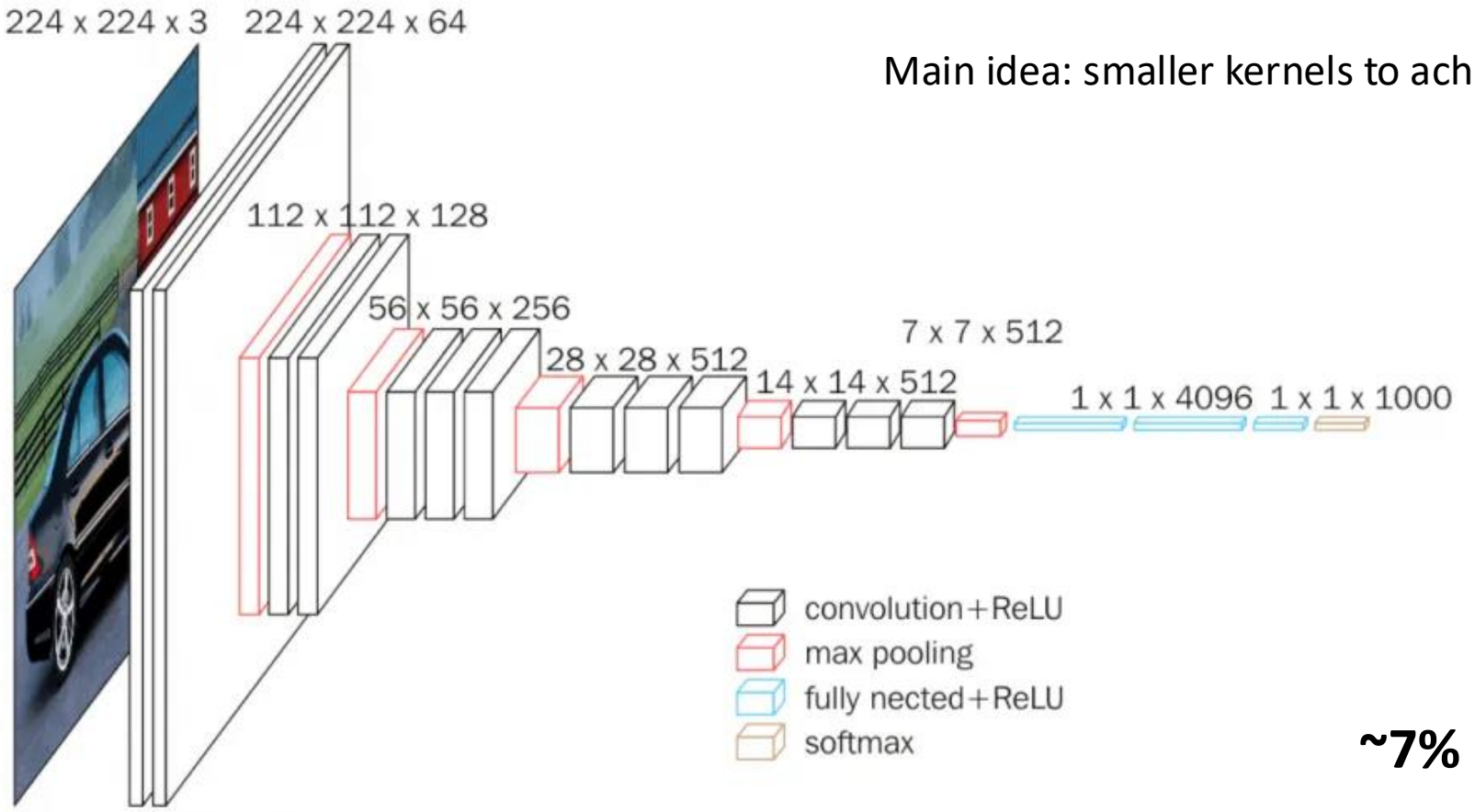
Prewitt operator

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

Sobel operator

VGG (2014)

(from Oxford's Visual Geometry Group)



Main idea: smaller kernels to achieve higher depth

~7% top-5 error

https://medium.com/analytics-vidhya/concept-of-alexnet-convolutional-neural-network-6e73b4f9ee30

VGG (2014)

(from Oxford's Visual Geometry Group)

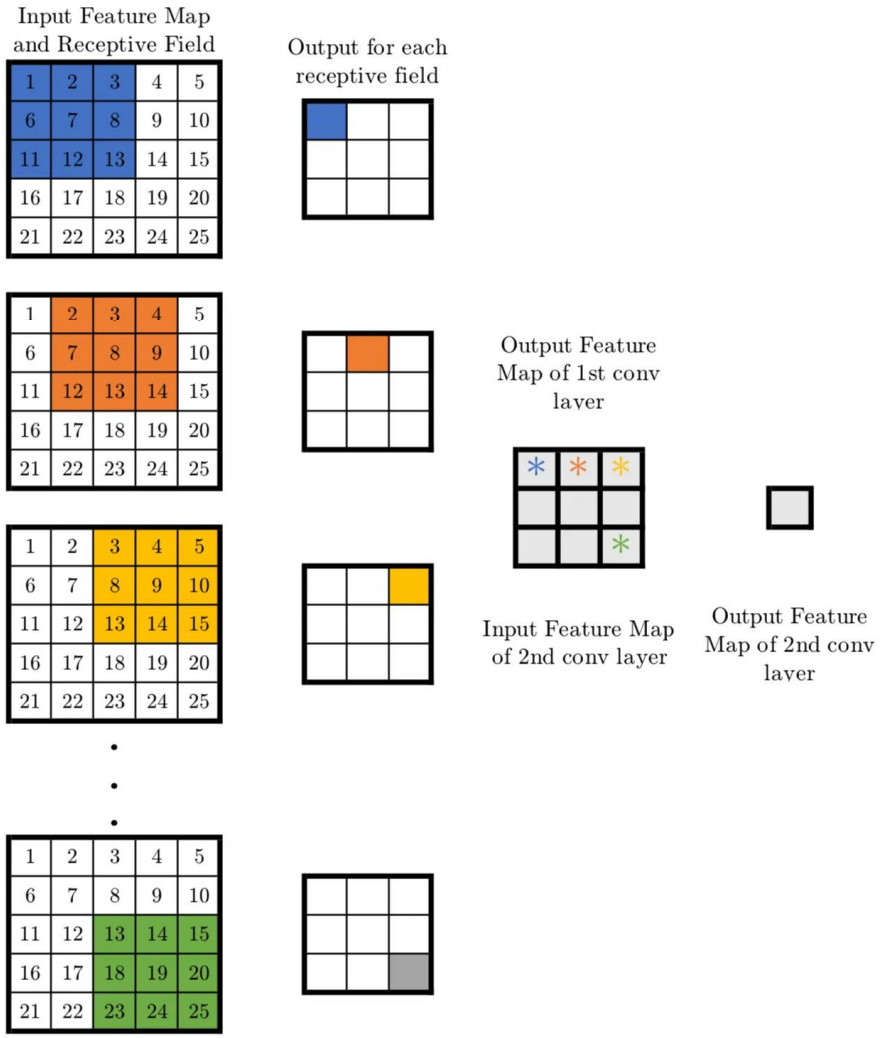
VGG16 - Structural Details													
#	Input Image			output			Layer	Stride	Kernel		in	out	Param
1	224	224	3	224	224	64	conv3-64	1	3	3	3	64	1792
2	224	224	64	224	224	64	conv3064	1	3	3	64	64	36928
	224	224	64	112	112	64	maxpool	2	2	2	64	64	0
3	112	112	64	112	112	128	conv3-128	1	3	3	64	128	73856
4	112	112	128	112	112	128	conv3-128	1	3	3	128	128	147584
	112	112	128	56	56	128	maxpool	2	2	2	128	128	65664
5	56	56	128	56	56	256	conv3-256	1	3	3	128	256	295168
6	56	56	256	56	56	256	conv3-256	1	3	3	256	256	590080
7	56	56	256	56	56	256	conv3-256	1	3	3	256	256	590080
	56	56	256	28	28	256	maxpool	2	2	2	256	256	0
8	28	28	256	28	28	512	conv3-512	1	3	3	256	512	1180160
9	28	28	512	28	28	512	conv3-512	1	3	3	512	512	2359808
10	28	28	512	28	28	512	conv3-512	1	3	3	512	512	2359808
	28	28	512	14	14	512	maxpool	2	2	2	512	512	0
11	14	14	512	14	14	512	conv3-512	1	3	3	512	512	2359808
12	14	14	512	14	14	512	conv3-512	1	3	3	512	512	2359808
13	14	14	512	14	14	512	conv3-512	1	3	3	512	512	2359808
	14	14	512	7	7	512	maxpool	2	2	2	512	512	0
14	1	1	25088	1	1	4096	fc		1	1	25088	4096	102764544
15	1	1	4096	1	1	4096	fc		1	1	4096	4096	16781312
16	1	1	4096	1	1	1000	fc		1	1	4096	1000	4097000
Total													138,423,208

Good performance and small network, still in use for fast models

https://medium.com/analytics-vidhya/concept-of-alexnet-convolutional-neural-network-6e73b4f9ee30

VGG (2014)

(from Oxford's Visual Geometry Group)



https://medium.com/analytics-vidhya/concept-of-alexnet-convolutional-neural-network-6e73b4f9ee30

Bigger and Deeper

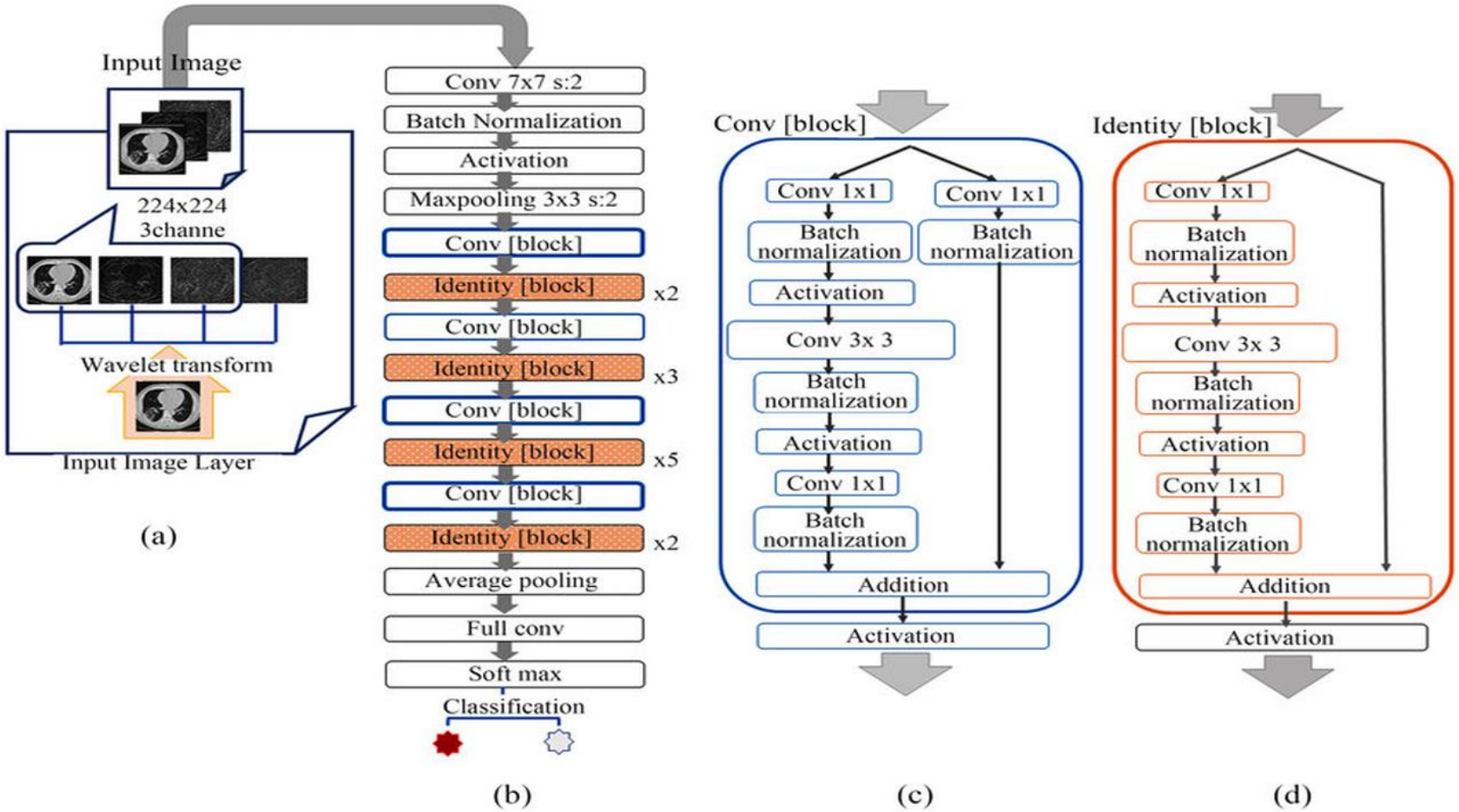
VGG19, 3 weeks of training.

GoogleLeNet.

"It was demonstrated that the representation depth is beneficial for the classification accuracy, and that state-of-the-art performance on the ImageNet challenge dataset can be achieved using a conventional ConvNet architecture."

Simonyan & Zisserman, ICLR'15

ResNet (2015)



~4% top-5 error

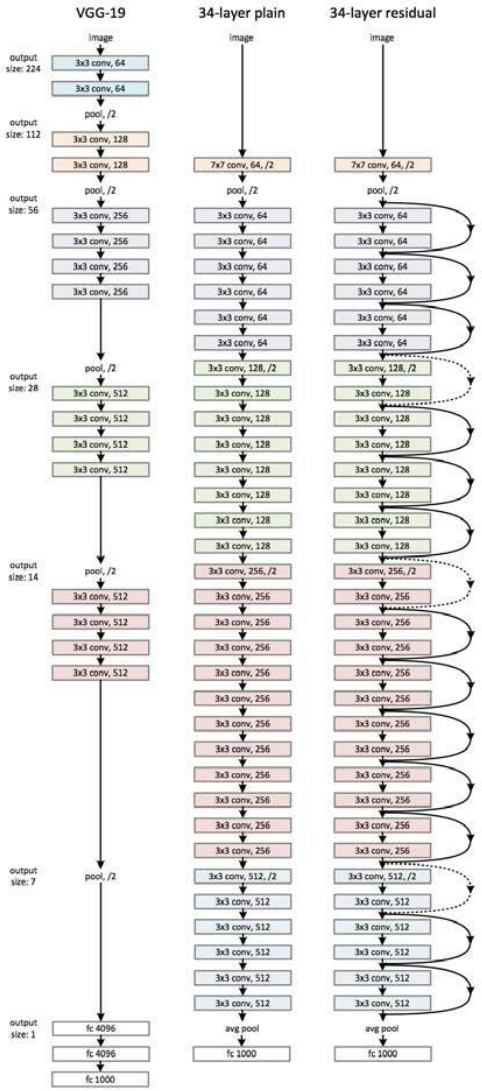
Popular ResNet-50, but deeper models exist

Allow even deeper networks thanks to skip connections, solving the VG problem

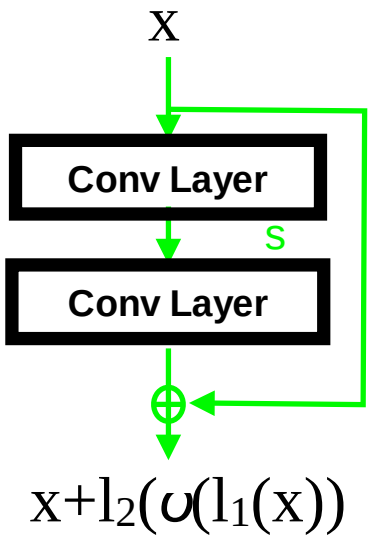
More accurate but way slower than VGG

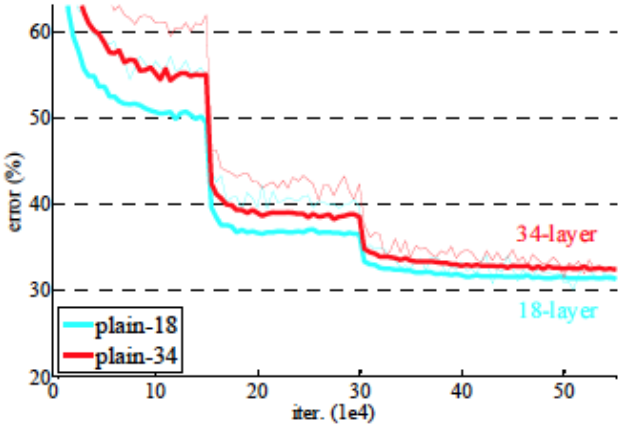
https://medium.com/analytics-vidhya/concept-of-alexnet-convolutional-neural-network-6e73b4f9ee30

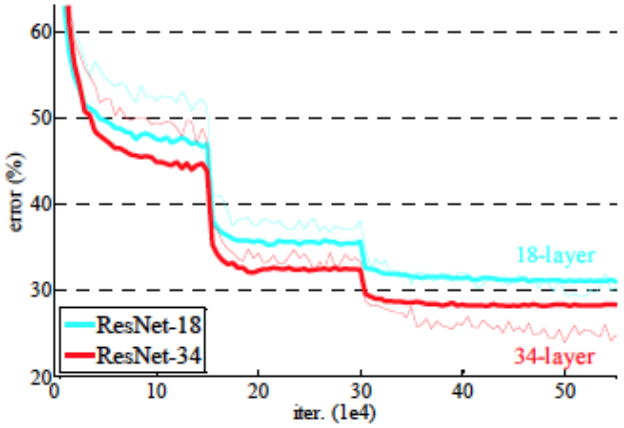
Deeper and Deeper



Resnet

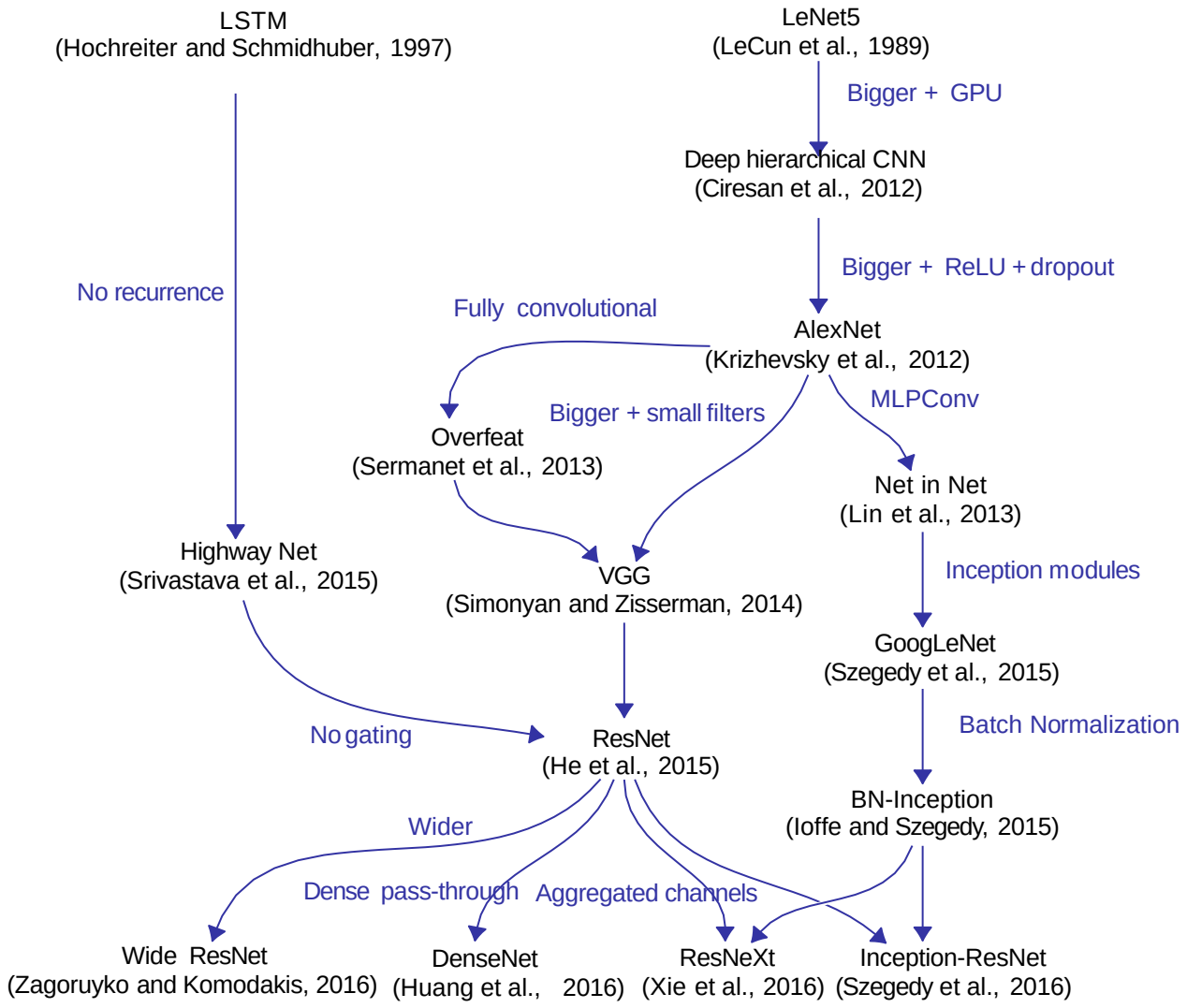






He et al. , CVPR'16

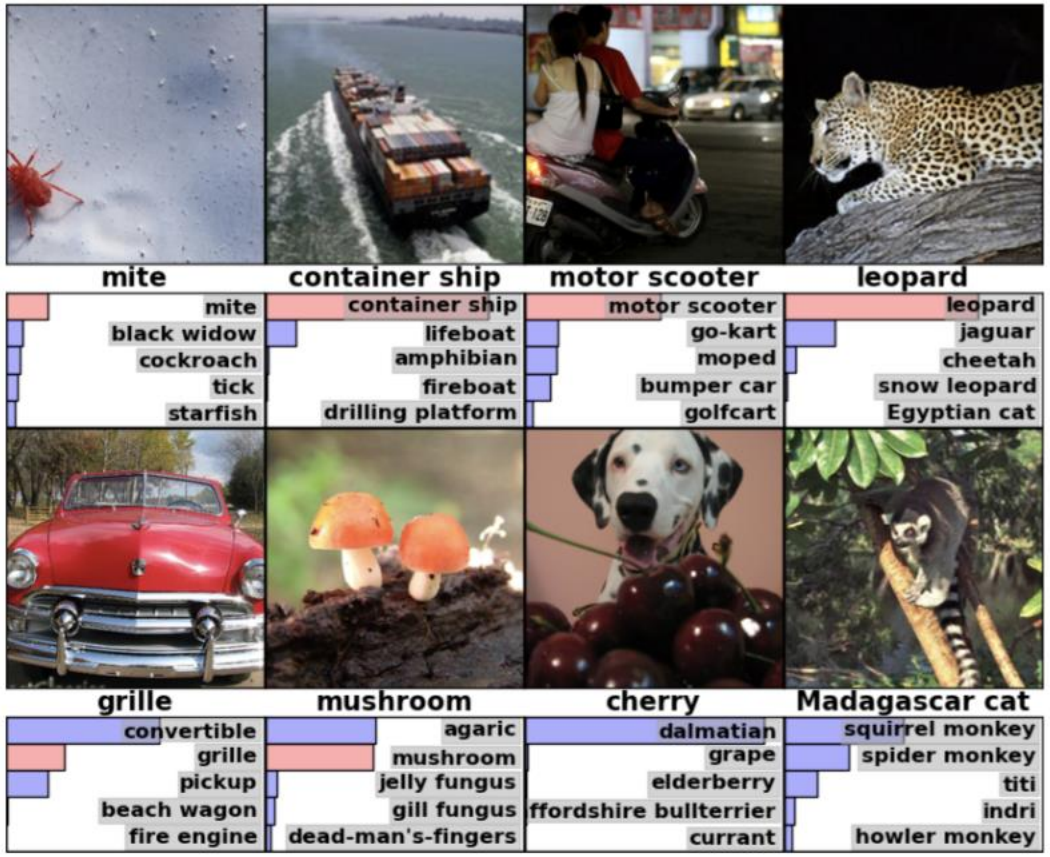
Image Classification Taxonomy 1989 —2016

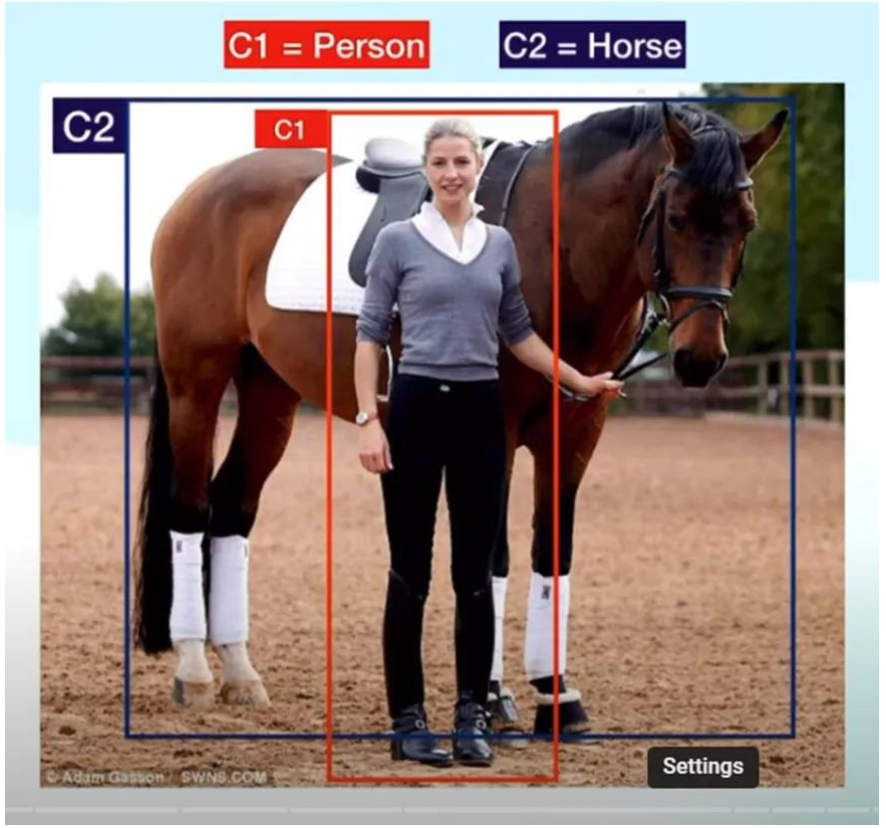


Up to now, object detection is a classification problem

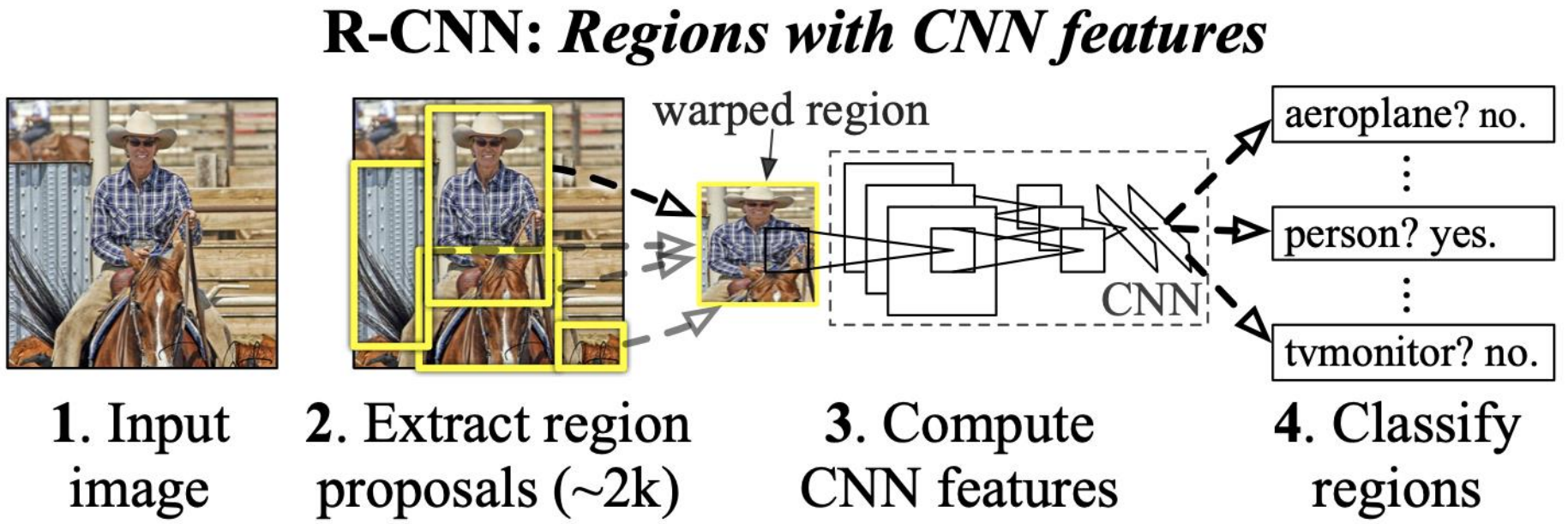
What if it's a regression problem?

We are now trying to find the object's locations, and not simply their existence in the image





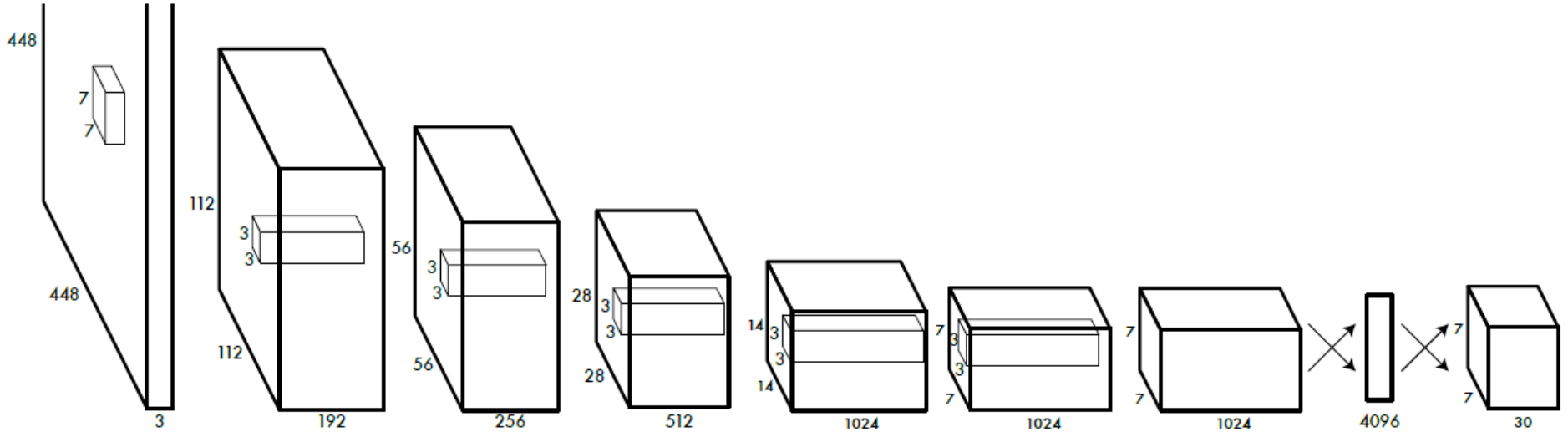
Popular approach: propose regions



Rich feature hierarchies for accurate object detection and semantic segmentation,
Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik, UC Berkeley , CVPR 2014

YOLO(v1) 2016

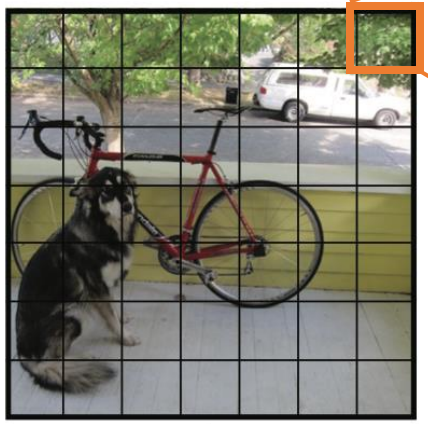
You only look once



Leaky ReLU activations

J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)

YOLOv1 2016



S × S grid on input

Confidence score

Class probabilities

$$[$$

$$p(c_1), p(c_2), \ldots, (c_n),$$ Length n (number of classes)

$$C_1, x_1, y_1, \sqrt{w_1}, \sqrt{h_1},$$

$$C_2, x_2, y_2, \sqrt{w_2}, \sqrt{h_2},$$ Length 5 (4 coordinates and a confidence score)

$$\ldots$$

$$]$$

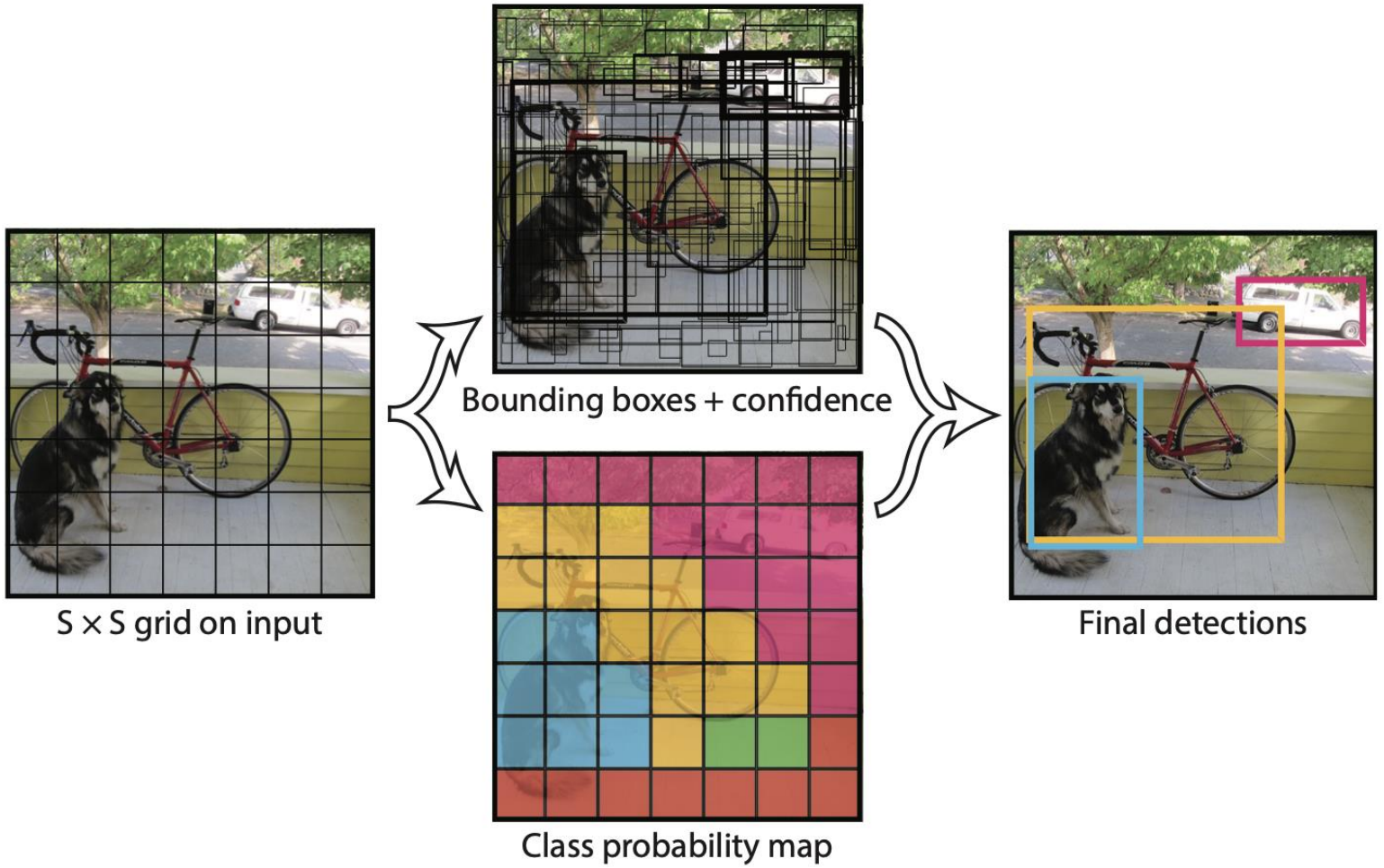
B bounding boxes

Overall size: (Bx5 + n)

In the case of the original YOLOv1, B=2 and n=20

J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)

YOLOv1 2016



J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)

Intersection over Union

$$IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$

https://pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/

YOLOv1 2016

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2$$

J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)

YOLOv1 2016

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

GT

$$C = \begin{cases} 0 \; if \; cell \; i \; has \; no \; GT \; objects \\ IoU \; if \; it \; does \end{cases}$$

J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)

YOLOv1 2016

Pre-training on ImageNet classification at half resolution
Training on Pascal VOC

J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)

YOLOv1 2016

Object detection

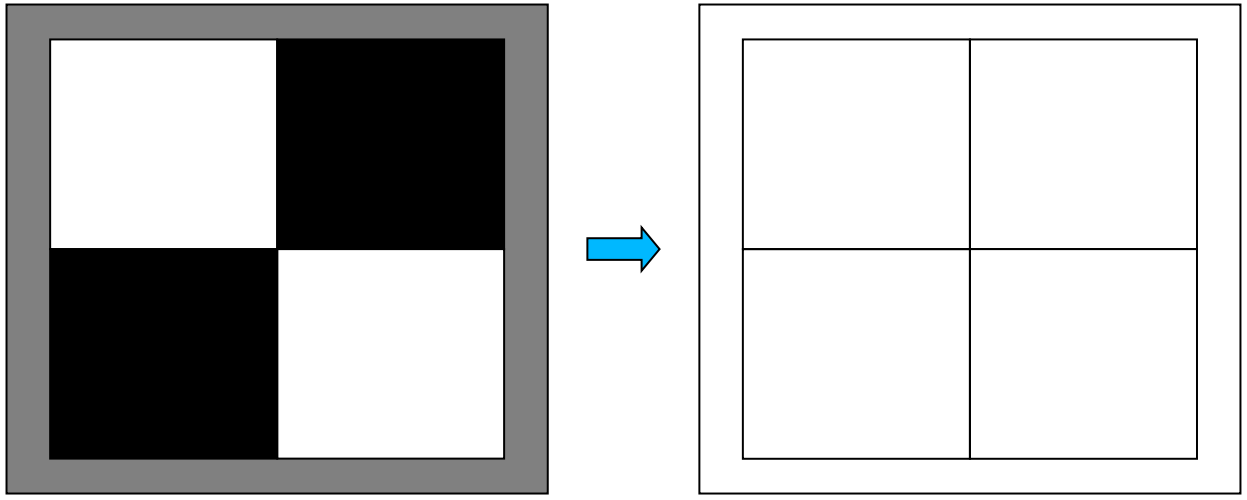
Model	Pascal 2007 mAP	Speed	
DPM v5	33.7	0.07 fps	14.3s
R-CNN	66.0	0.05 fps	20.0s
Fast R-CNN	70.0	0.5 fps	2.0s
Faster R-CNN	73.2	7 fps	0.14s
YOLO	69.0	45 fps	0.02s
100Hz DPM	16.0	100 fps	0.01s
Fast YOLO	16.0	155 fps	0.006s

https://www.youtube.com/watch?v=ag3DLKsl2vk

DEMO

Image Segmentation

Reminder: Edges and Regions



Edges:

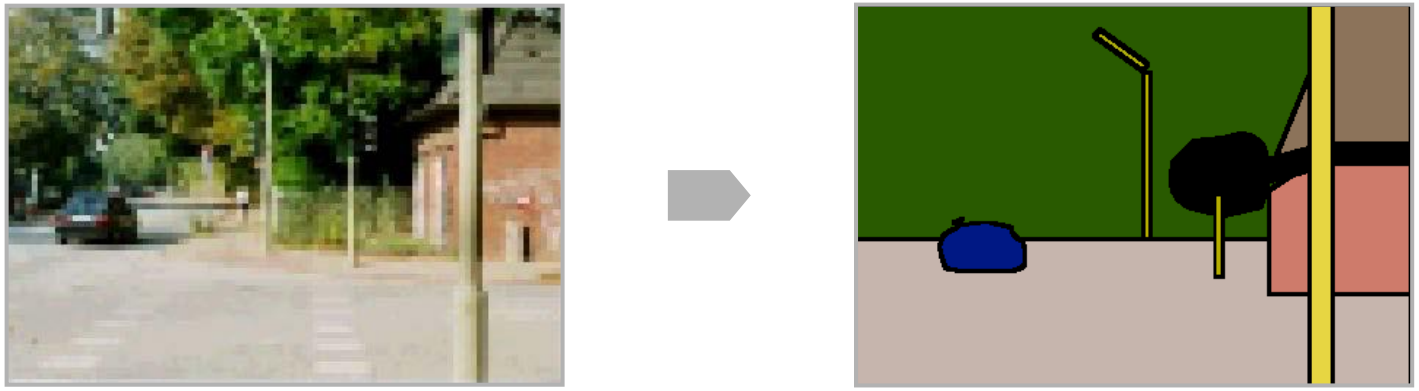
- Boundary between bland image regions.

Regions:

- Homogenous areas between edges.

- Edge/Region Duality.

Region Segmentation



Ideal region: Set of pixels with the same statistical properties and corresponding to the same object.

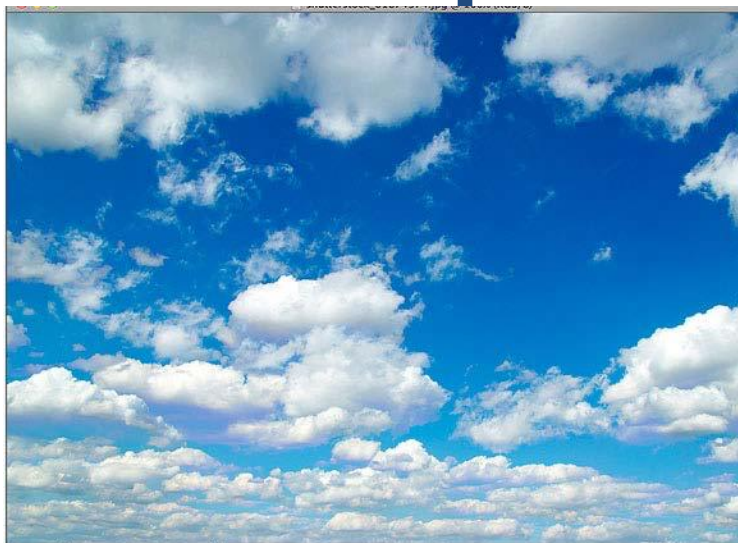
Purpose: Should help with recognition, tracking, image database retrieval, and image compression among other high-level vision tasks.

Application: Automated Driving

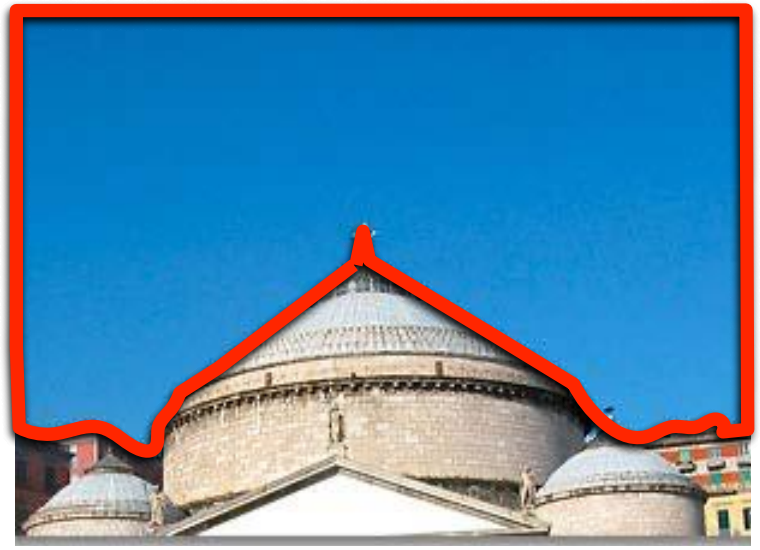
Vu et al. , ArXiv'22

Applications: Photoshop

I find this blue sky too bland!



I prefer this one.



Find the sky region.

Replace it.

https://www.photoshopessentials.com/basics/selections/magic-wand-tool/

In Theory

Look for an image partition such that:

$$I = \bigcup_{i=1}^{m} S_i$$

$$S_i \cap S_j = 0, \forall i \neq j$$

$$H(S_i) = True, \forall i$$

$$H(S_i \cup S_j) = False, \text{ if } S_i \text{ and } S_j \text{ are adjacent.}$$

where H measures homogeneity.

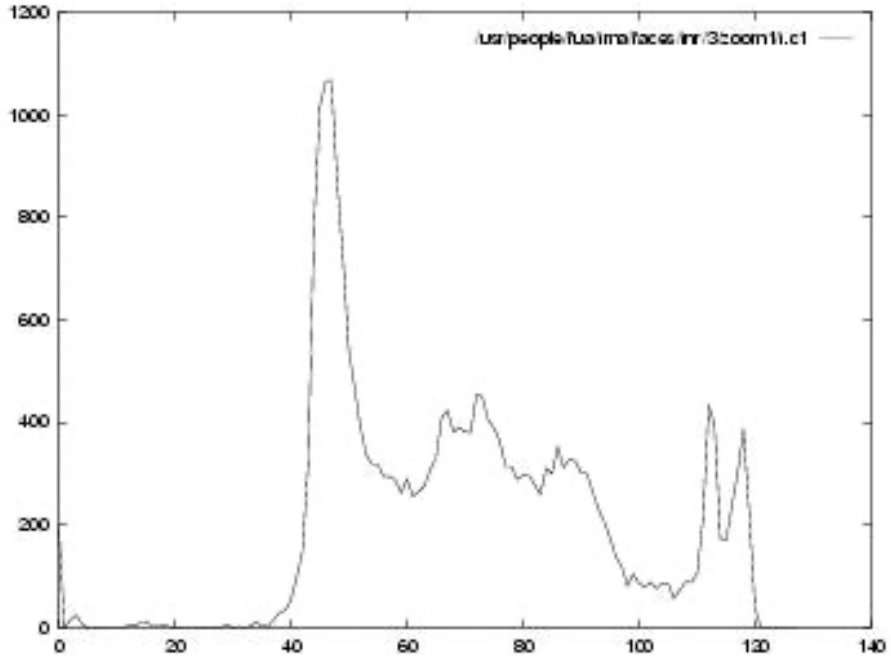
Complex Gray Level Variations

- Simple thresholding and other basic image operations do not suffice.
- The H predicate is difficult to define.

From Simple to Complex Algorithms

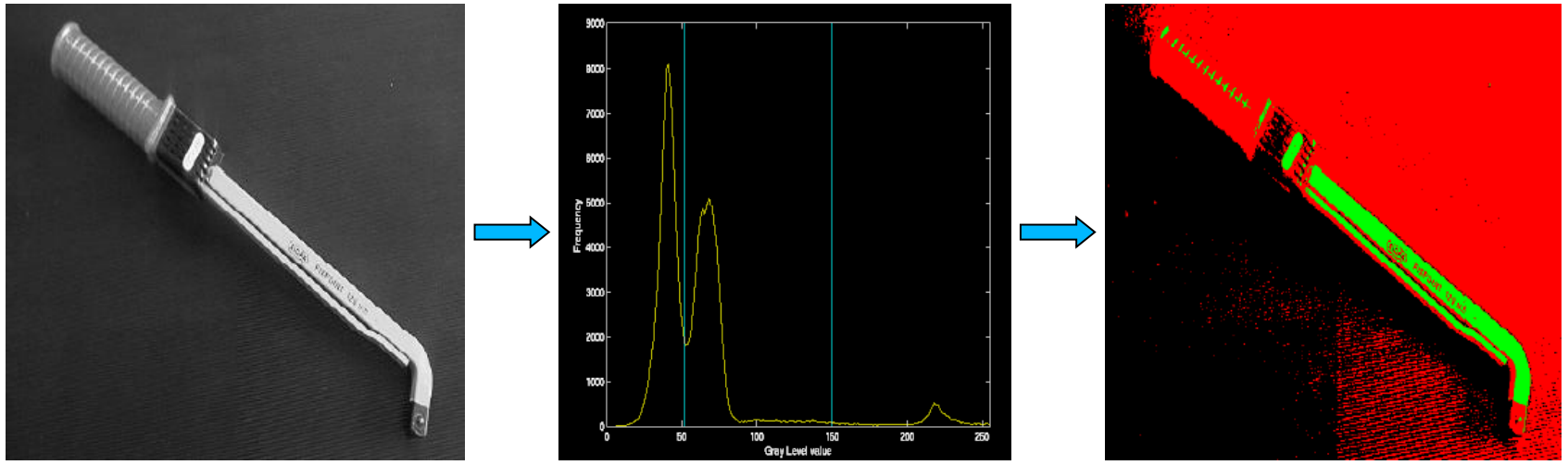
- Region Growing.
- Histogram Splitting.
- K-Means.
- Graph Theoretic Methods.
- Convolutional Neural Nets.

Image Histogram



Number of pixels that have a given gray level.

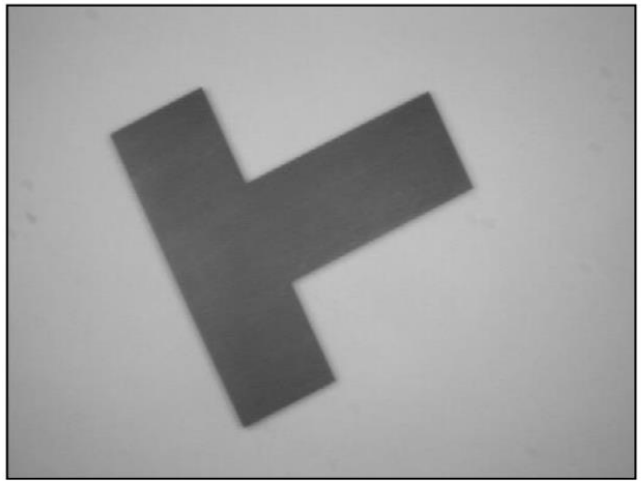
Histogram Splitting



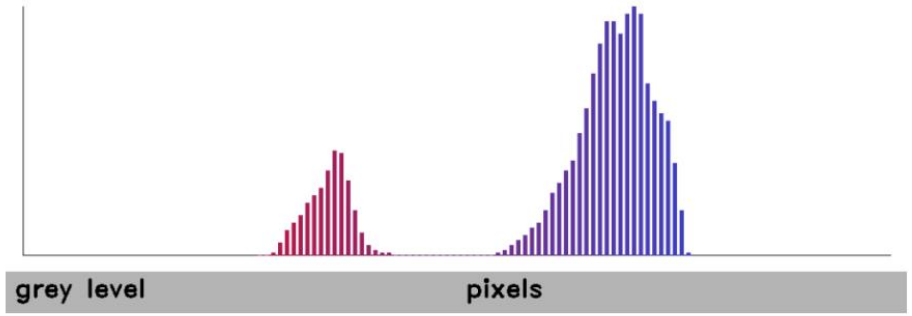
- Groups of similar pixels appear as bumps in the brightness histogram
- Split the histogram at local-minima
- Label pixels according to which bump they belong to

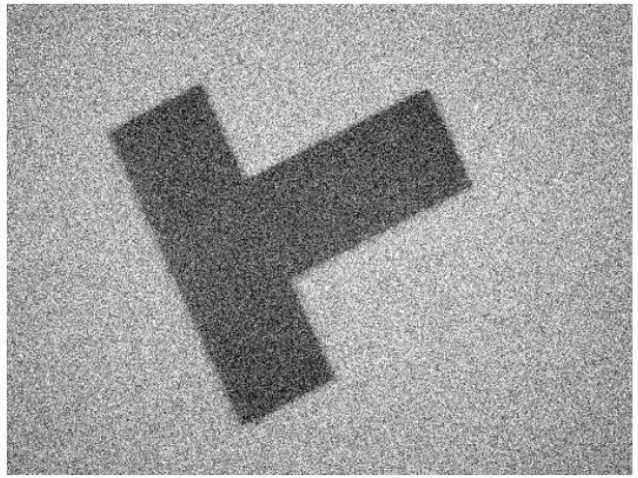
—> Cannot stop there, must go on.

Histogram Splitting

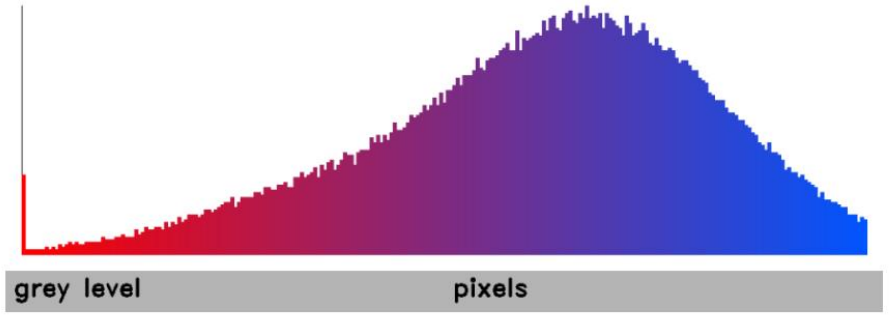


Noiseless Image
and its Histogram



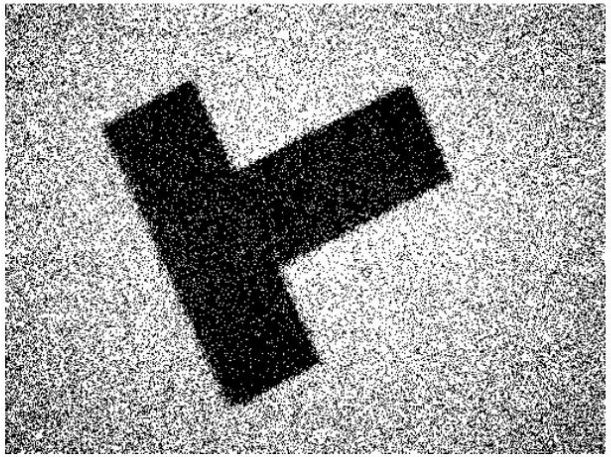


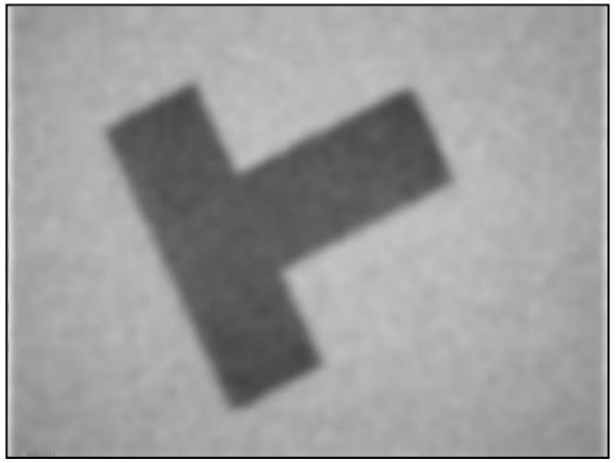
Noisy Image and
its Histogram



Histogram Splitting

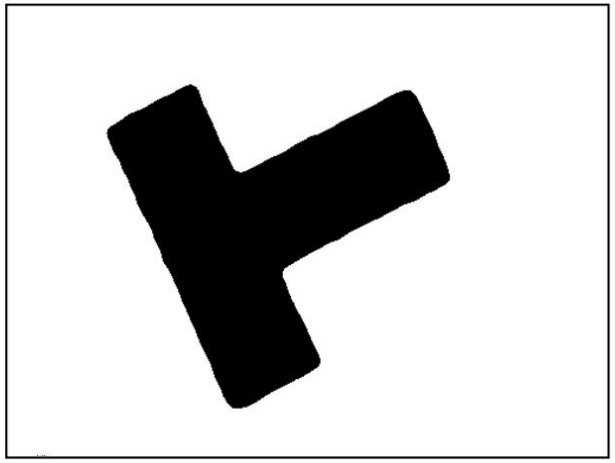
Binarization of the Noisy Image



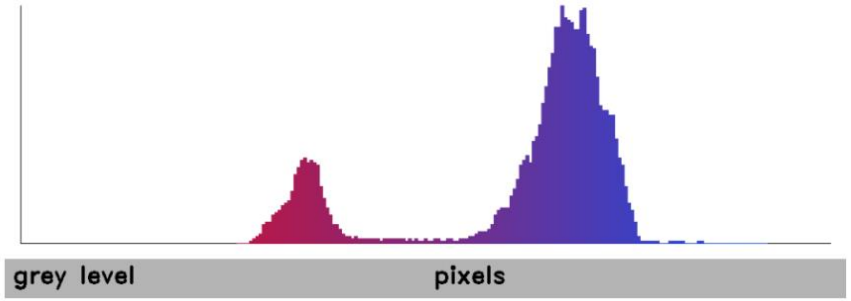


Filtered Image

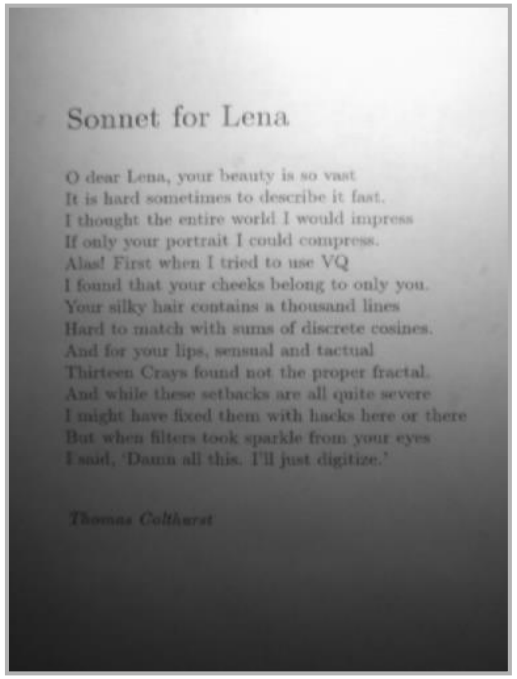
Binarization of the Filtered Image

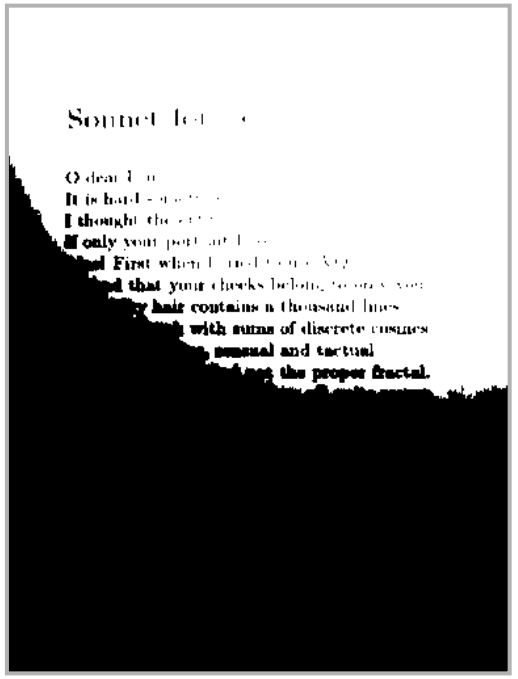


Histogram of the Filtered Image



No Global Threshold

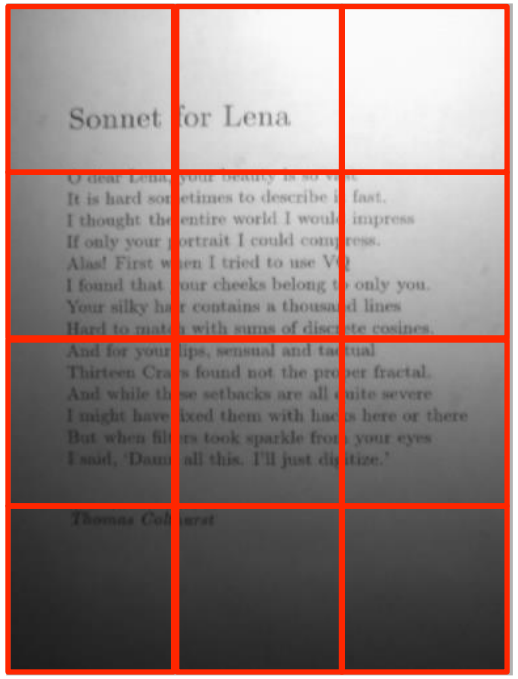


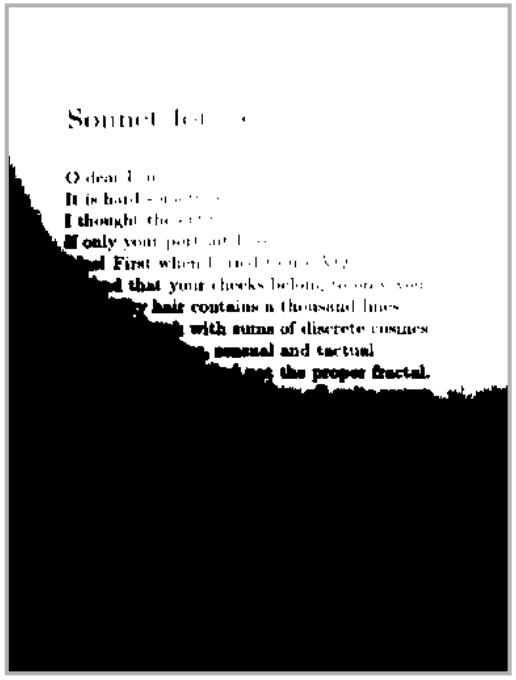


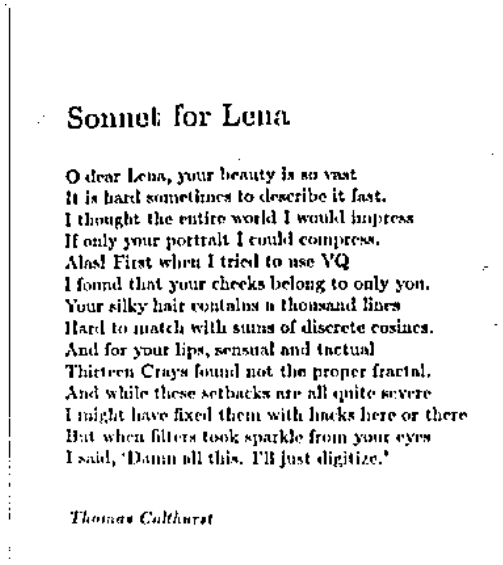
Because of the complex illumination:

- The top of the page is much lighter than the bottom.
- No global threshold can be found.
- Local thresholds can account for this.

Use Local Histograms







Global Local

- Compute a histogram in each red square.
- Find a local threshold.

Using Color

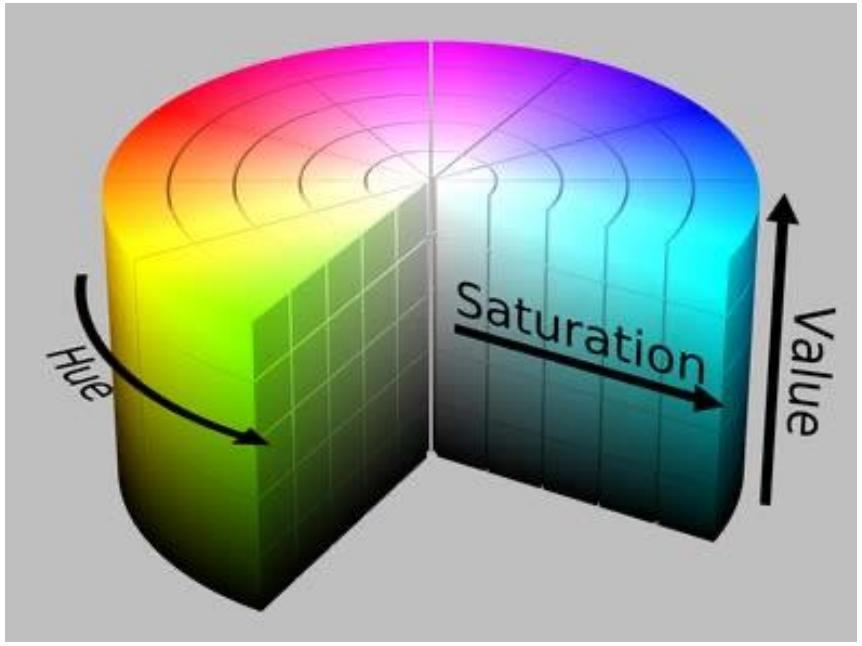
RGB

R

B

G

HSV Space



Hue / Saturation / Value

$$H = \arccos\left(\frac{0.5(2r - g - b)}{\sqrt{(r-g)^2 + (r-g)(g-b)}}\right) \text{ if } b < g$$

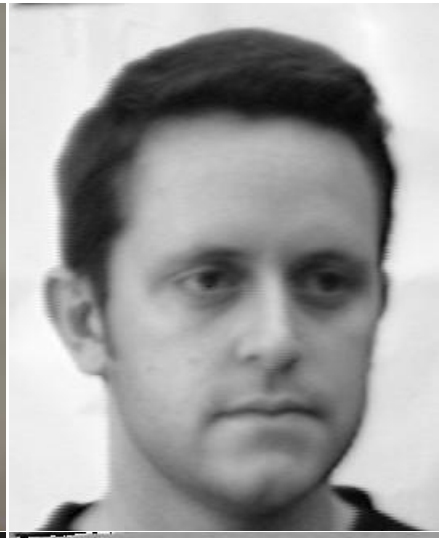
$$S = \max(r, g, b) - \min(r, g, b)$$

$$V = R + G + B$$

HSV Images

RGB

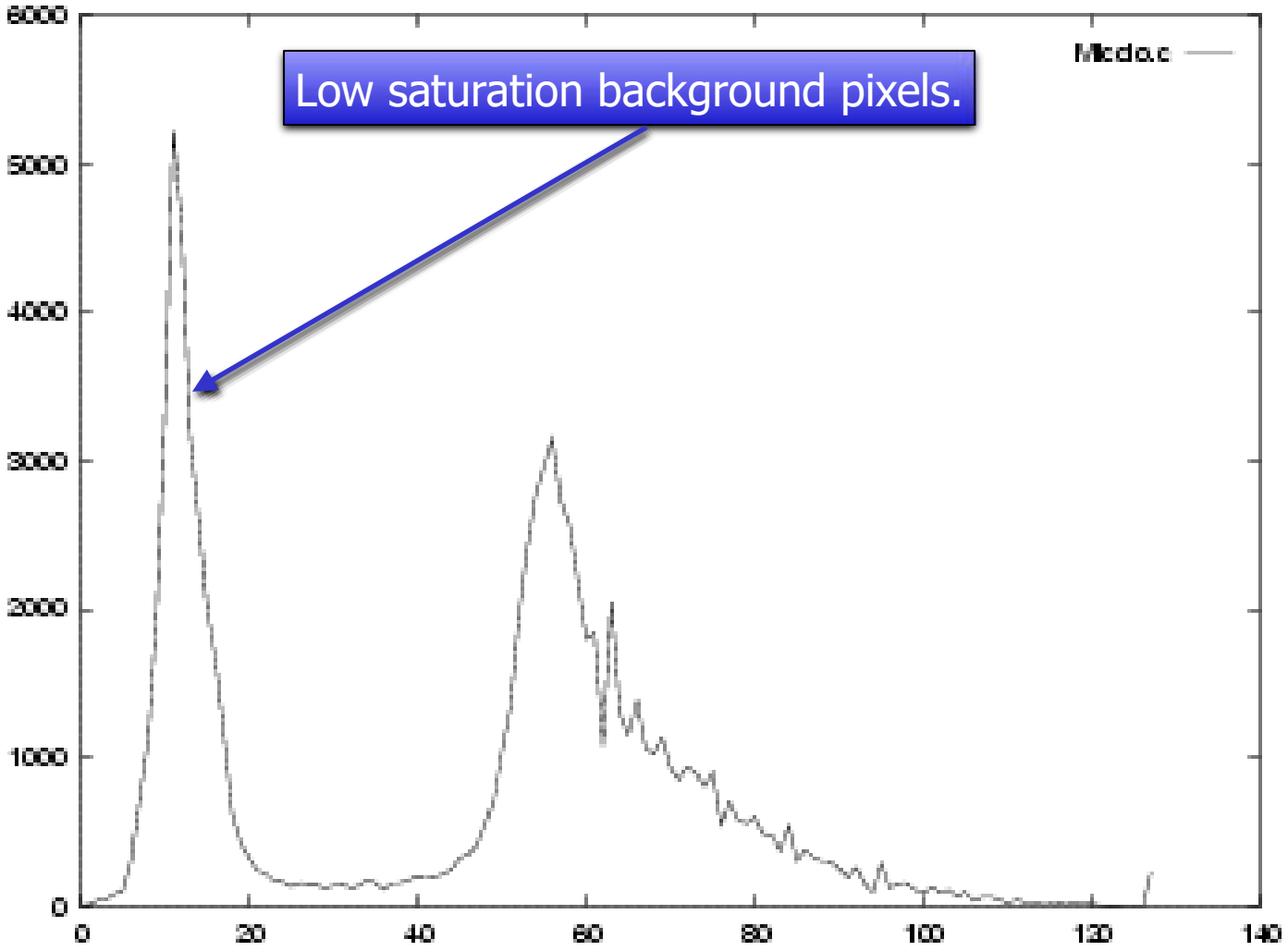
Value



Saturation

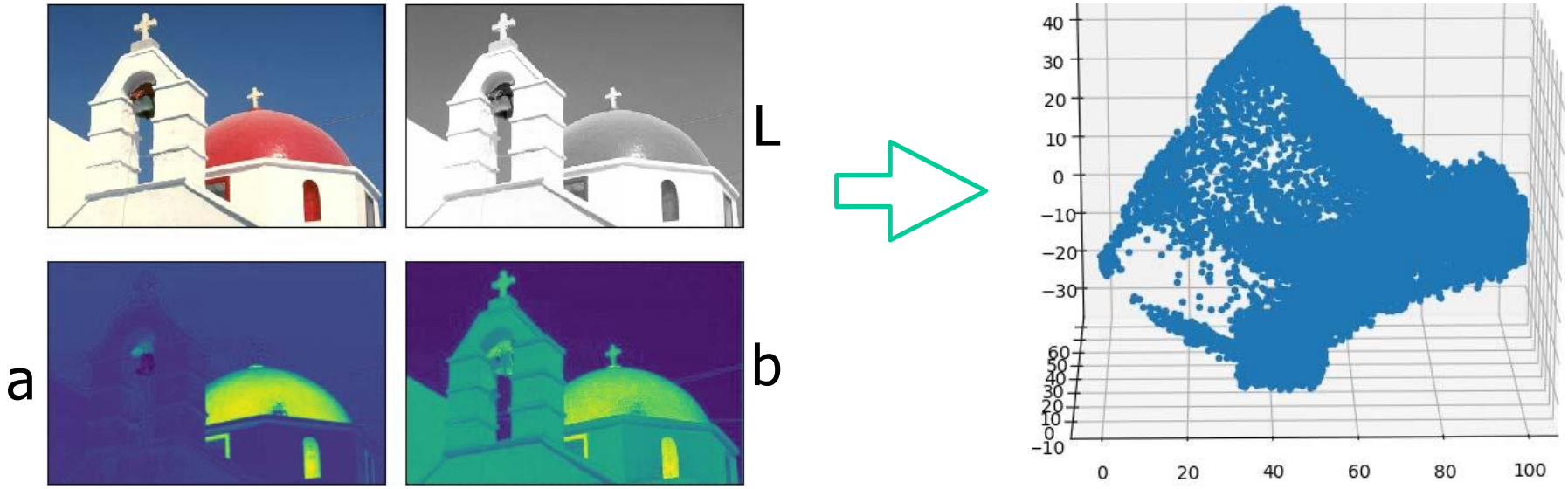
Hue

Saturation Histogram



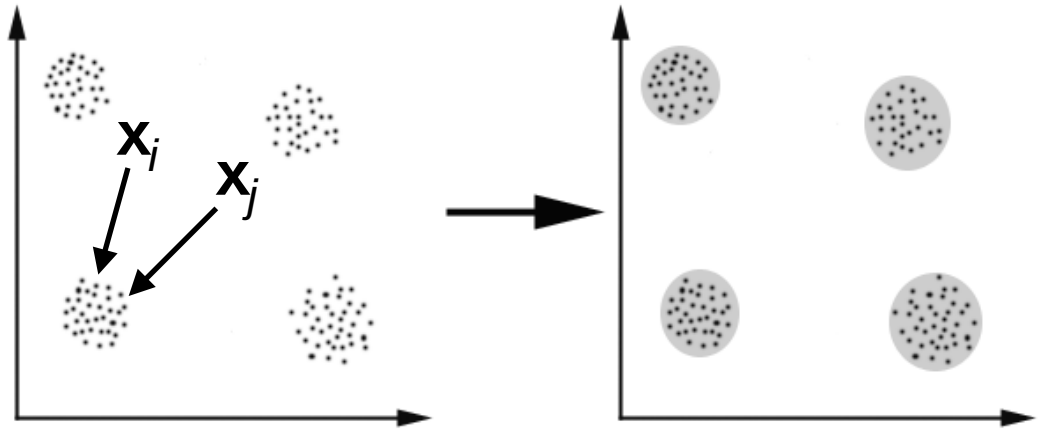
- This histogram is a lot easier to split!
- It makes it easy to segment the head from the background.

Segmentation as Clustering



- Each pixel has 2 spatial coordinates and 1 gray level or 3 color components.
- Segmentation can be understood as clustering in
  - 1D space (G);
  - 3D space (x,y,G), (H,S,V), (L,a,b);
  - 5D space (x,y,L,a,b).
- Ideally each cluster should be as compact as possible.

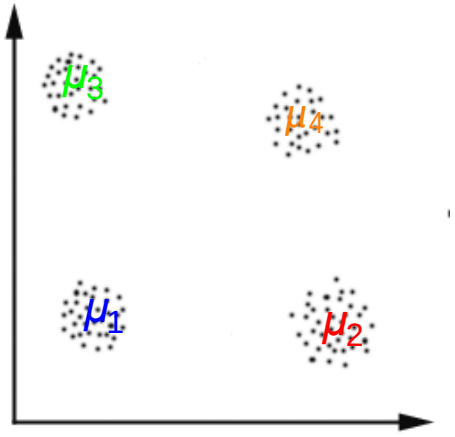
K-Means Clustering



Given a set of input samples:

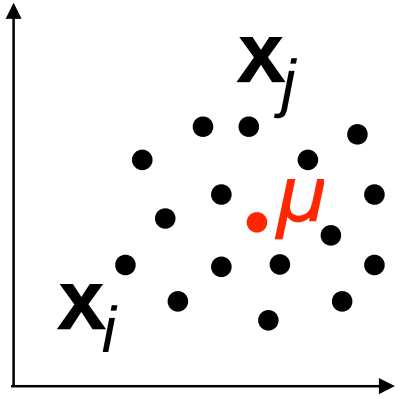
- Group the samples into K clusters.
- K is assumed to be known/given.
- In the example above, each sample is a point in 2D.
- In images, samples can be points in 1D, 3D, or 5D.

K-Means Clusters



- Cluster k is formed by the points $\{\mathbf{x}_{i_1^k}, \ldots, \mathbf{x}_{i_{n^k}^k}\}$.
- μ_k is the **center of gravity** of cluster k.

The mean of points $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, $\mathbf{x}_i \in \mathbb{R}^D$ is

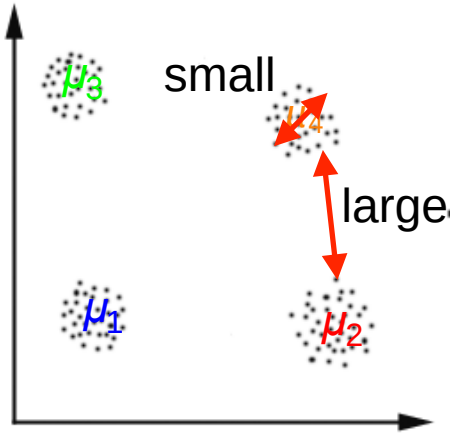


In 2D

$$\mu = \frac{1}{N}\sum_{i=1}^{N} \mathbf{x}_i \,, \mu \in \mathbb{R}^D$$

- If the $\mathbf{x}_i$ were physical points of equal mass, m would be their center of gravity.
- This applies in any dimension.

Formalization



- Cluster k is formed by the points $\{\mathbf{x}_{i_1^k}, \ldots, \mathbf{x}_{i_{n^k}^k}\}$.
- μ_k is the **center of gravity** of cluster k.

- The distances between the points within a cluster should be small.
- The distances across clusters should be large.
- This can be encoded via the distance to cluster centers $\{\mu_1, \ldots, \mu_K\}$:

$$\text{—> Minimize} \sum_{k=1}^{K} \sum_{j=1}^{n_k} (\mathbf{x}_{i_j^k} - \mu_k)^2$$

where $\{\mathbf{x}_{i_1^k}, \ldots, \mathbf{x}_{i_{n^k}^k}\}$ are the n^k samples that belong to cluster k.

Difficult Minimization Problem

Minimize

$$\sum_{k=1}^{K} \sum_{j=1}^{n_k} (\mathbf{x}_{i_j^k} - \mu_k)^2$$

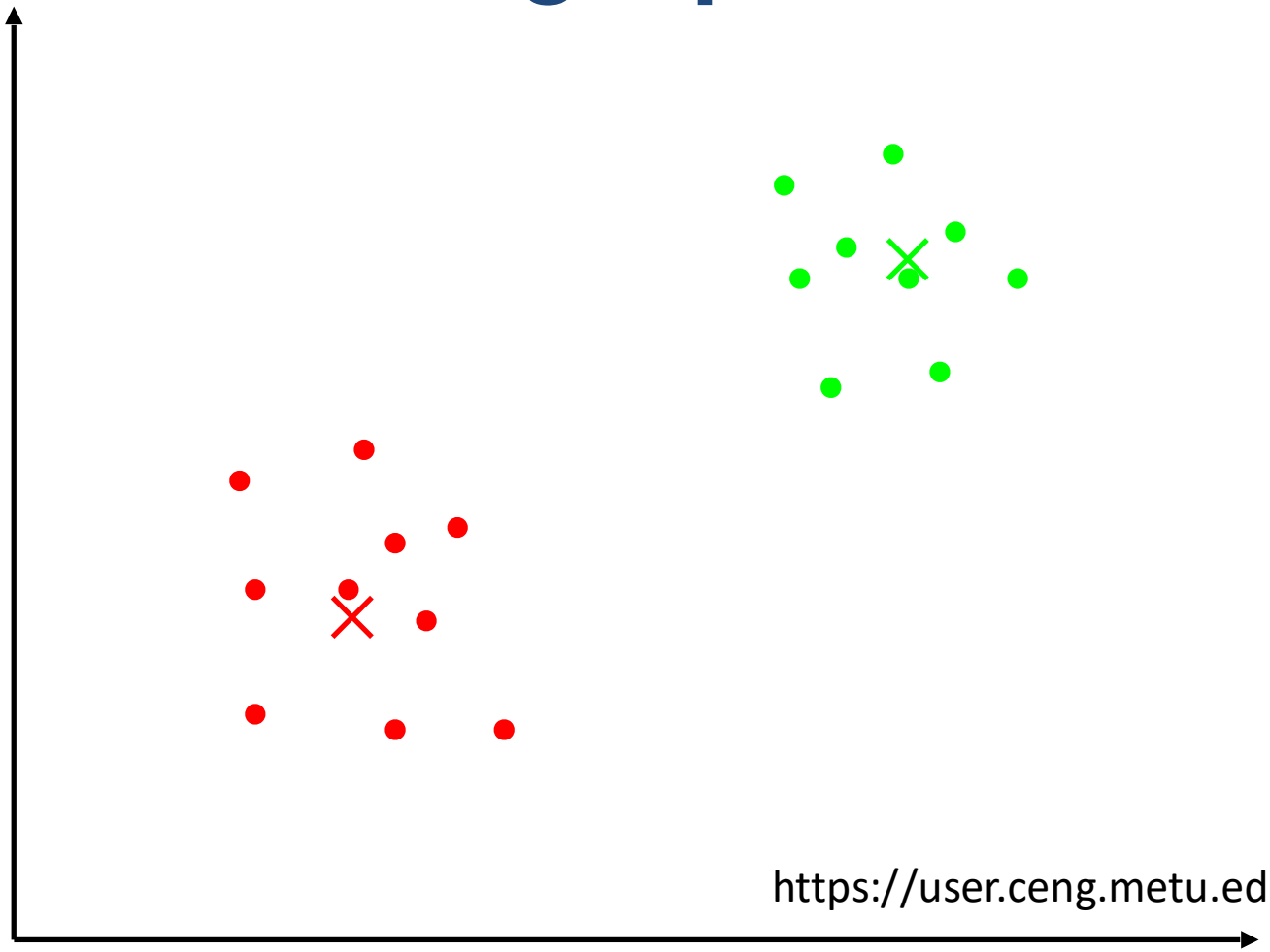
but:

- We don't know what points belong to what cluster.
- We don't know the center of gravity of the clusters.

Simple Solution to the Problem

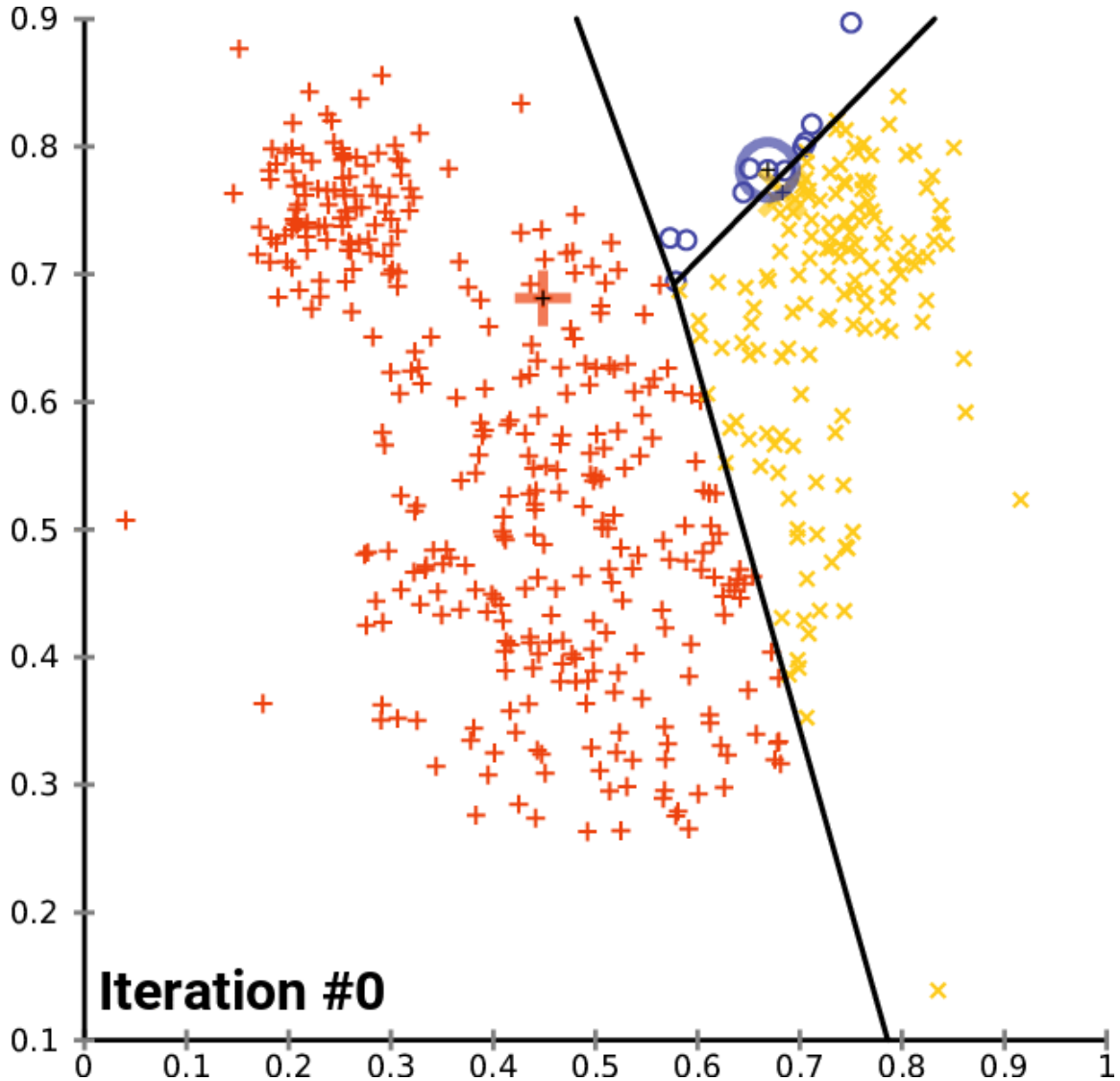
1. Initialize $\{\mu_1, \ldots, \mu_K\}$, randomly if need be.
2. Until convergence
   - 2.1. Assign each point $\mathbf{x}_i$ to the nearest center μ_k
   - 2.2. Update each center μ_k given the points assigned to it

Alternating Optimization

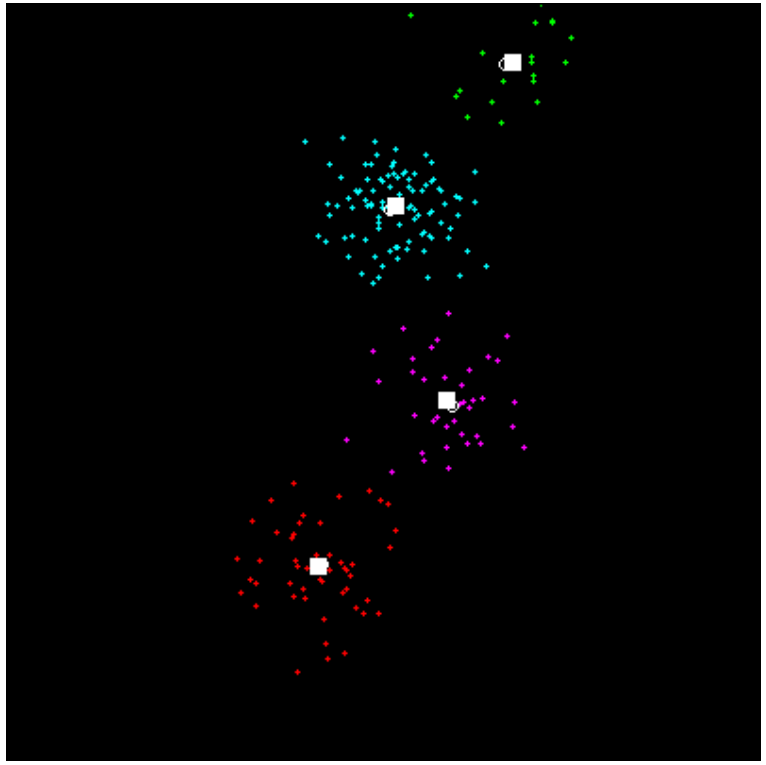


Demo
https://user.ceng.metu.edu.tr/~akifakkus/courses/ceng574/k-means/

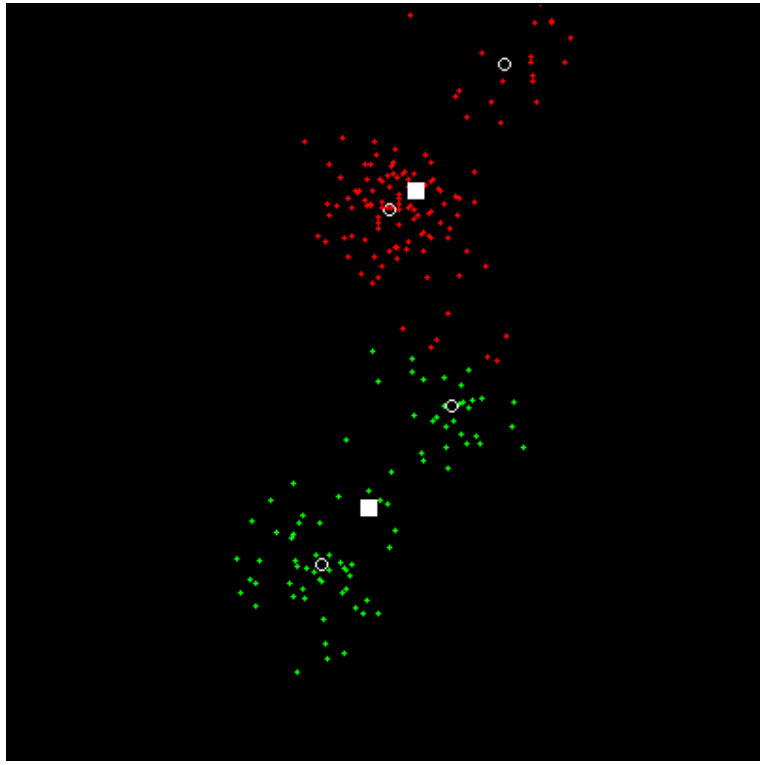
- Initialize
- Associate point to centers
- Recompute centers


0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Iteration #0

Initial Conditions Matter



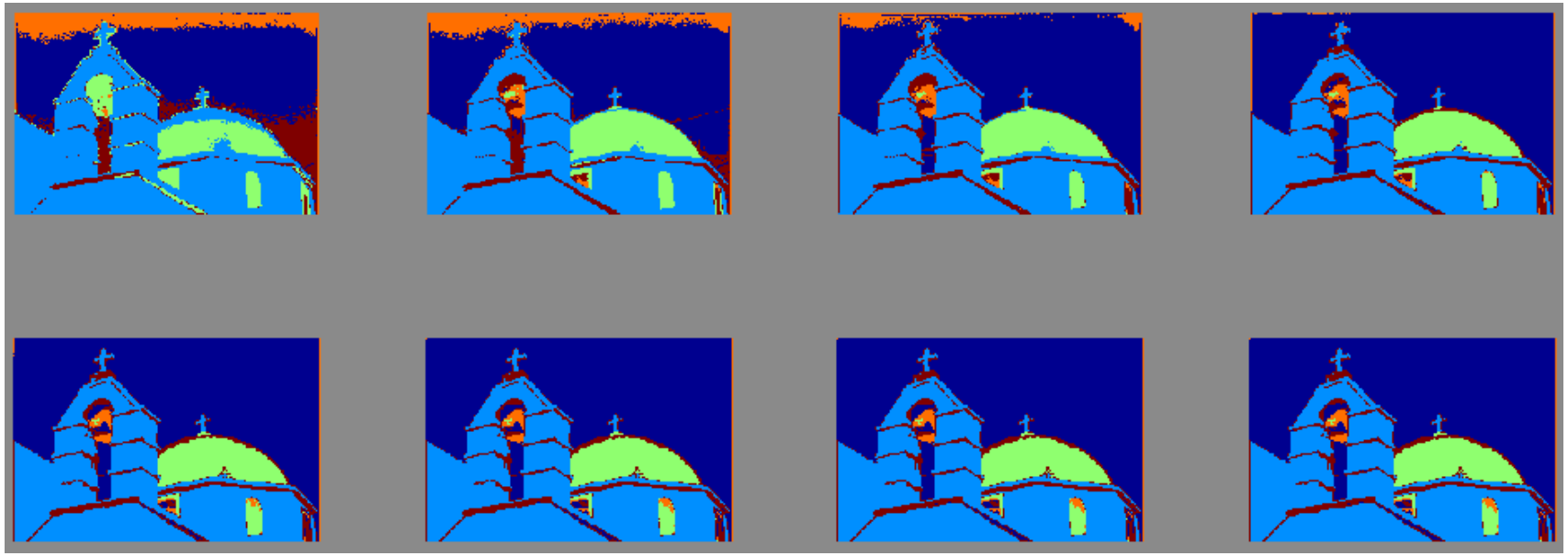
Initially, the points are assigned to the clusters at random—> Success.



Initially, the points are assigned to the closest cluster —> Failure.

—> In practice try several different random initialization and keep the one that yields the best result in term of the sum of square distances.

Color Only (3D)



8 iterations for k=5

Color Only (3D)

Different Initializations for k=5

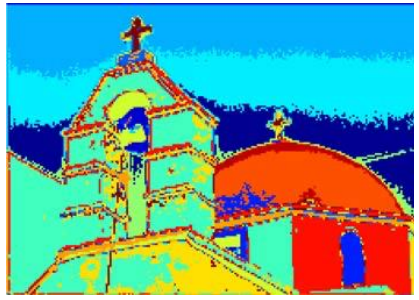



Different values of k


K=3

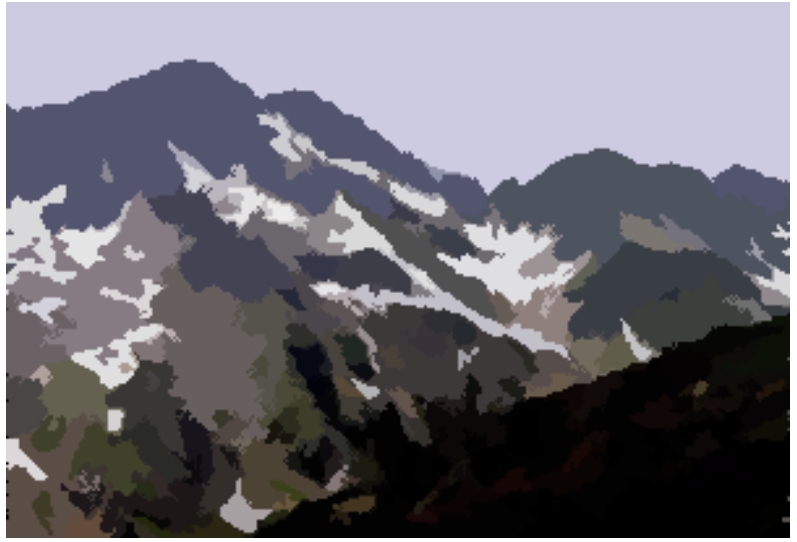

K=5


K=8


K=15

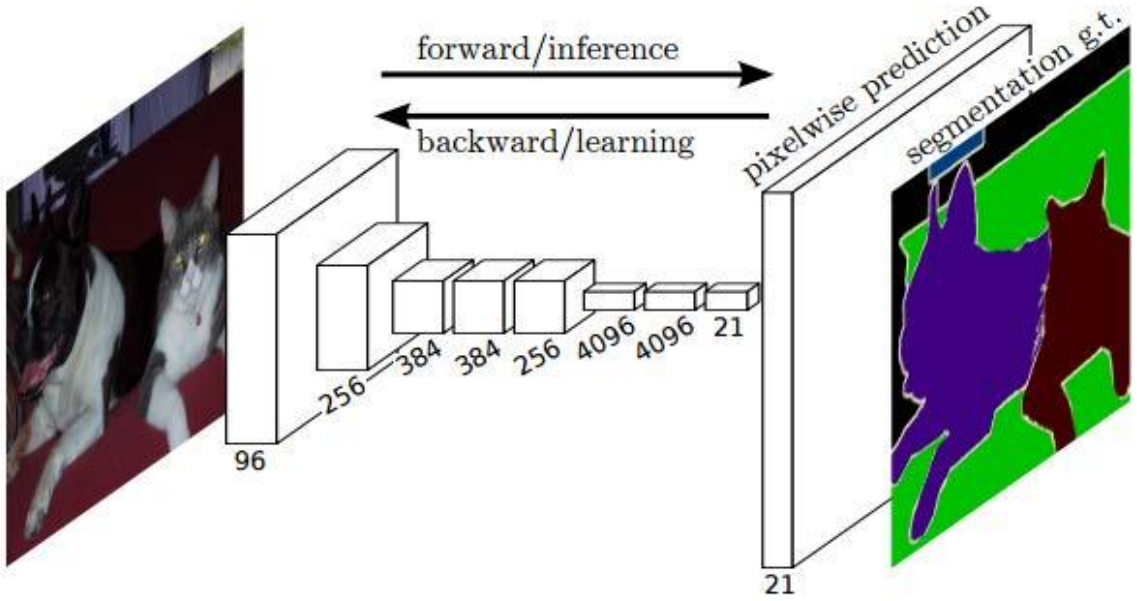
- Different results for different initializations.
- For k well chosen, the results are good for this image because it features bright and distinctive colors.

More Results



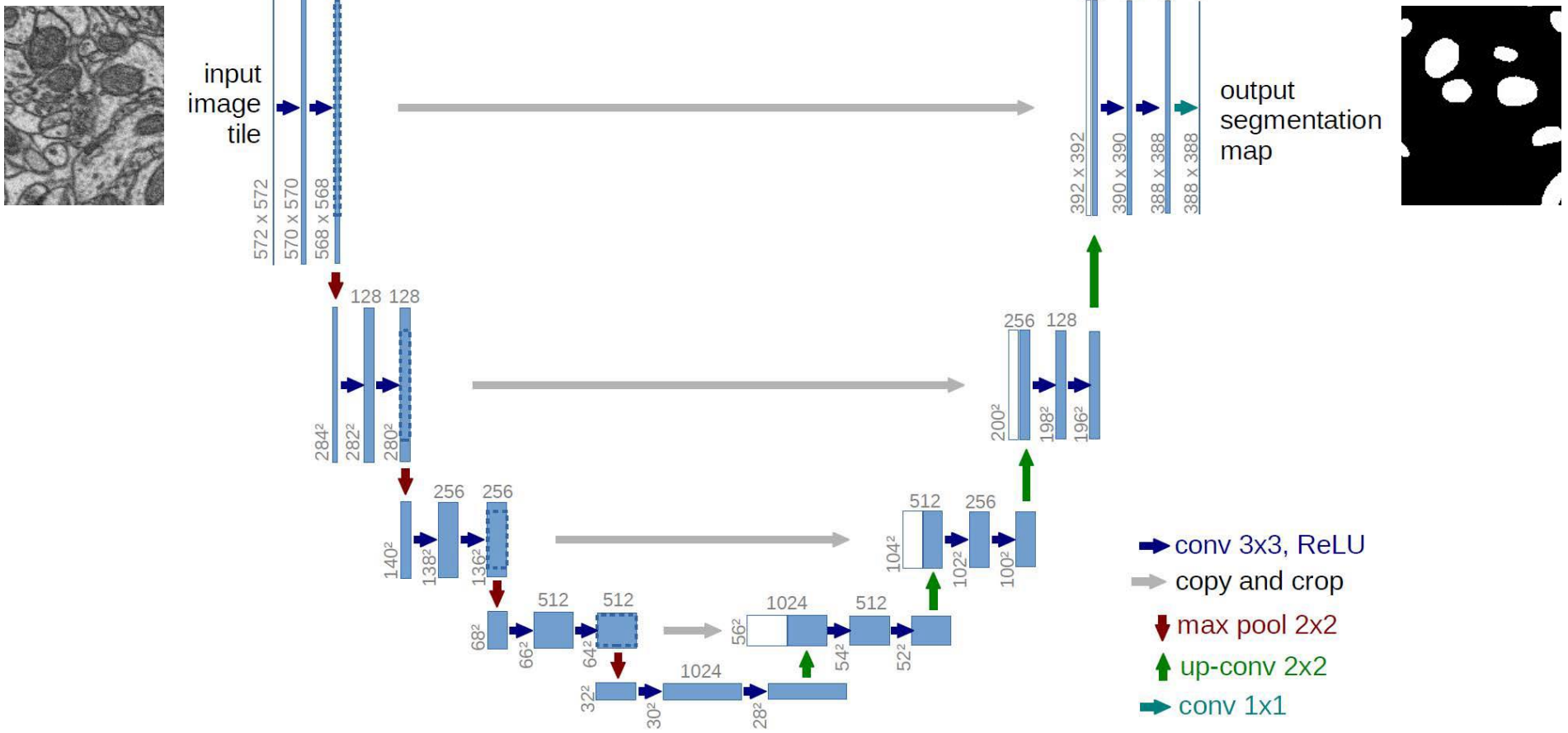
DEMO

Convolutional Neural Nets



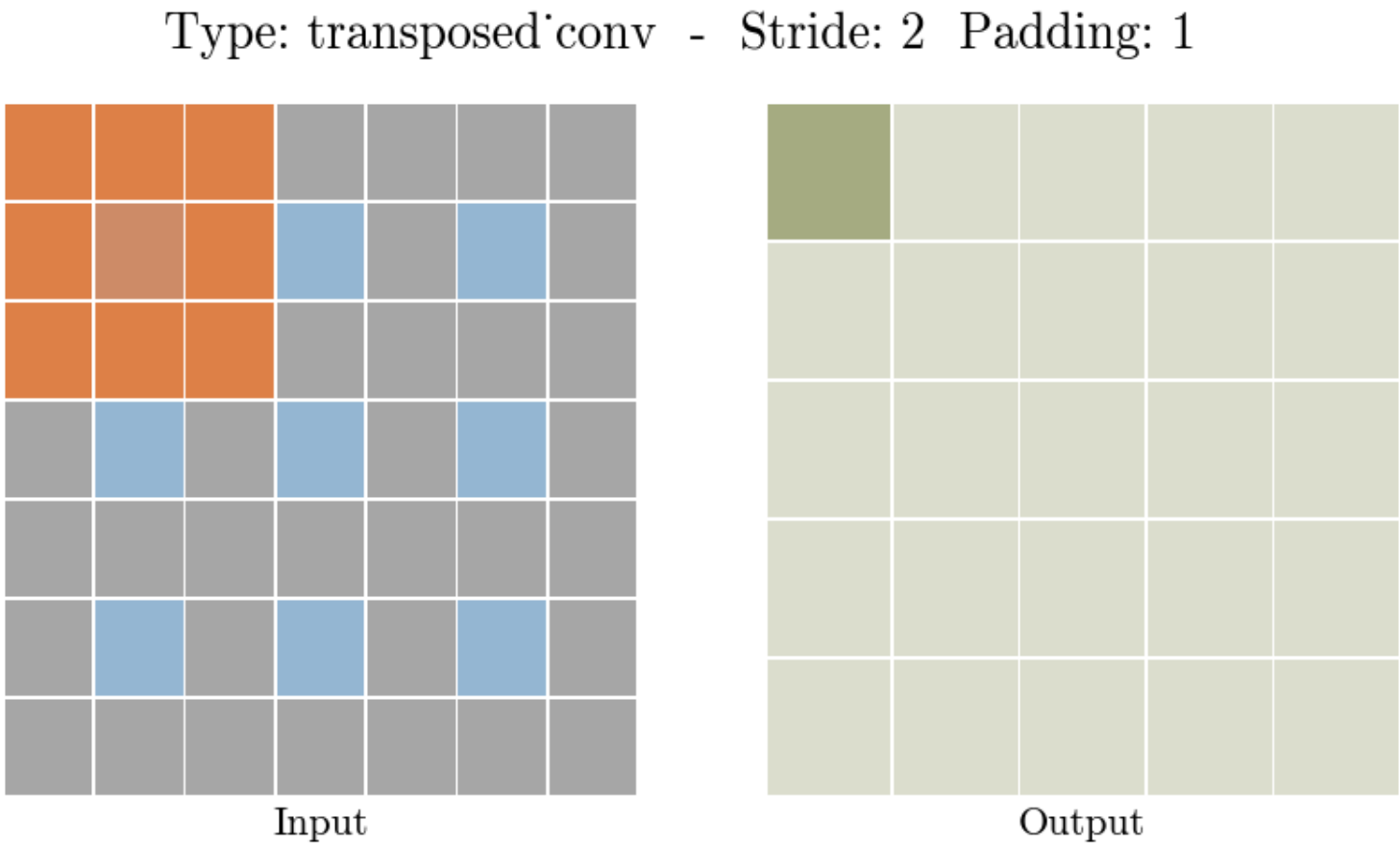
- Connect input layer to output one made of segmentation labels.
- Need layers that both downscale and upscale.
- Connect the lower layers directly to the upper ones.

U-Net for Segmentation



- Same architecture (in more details).
- Train it to produce a segmentation mask instead of a delineation mask.

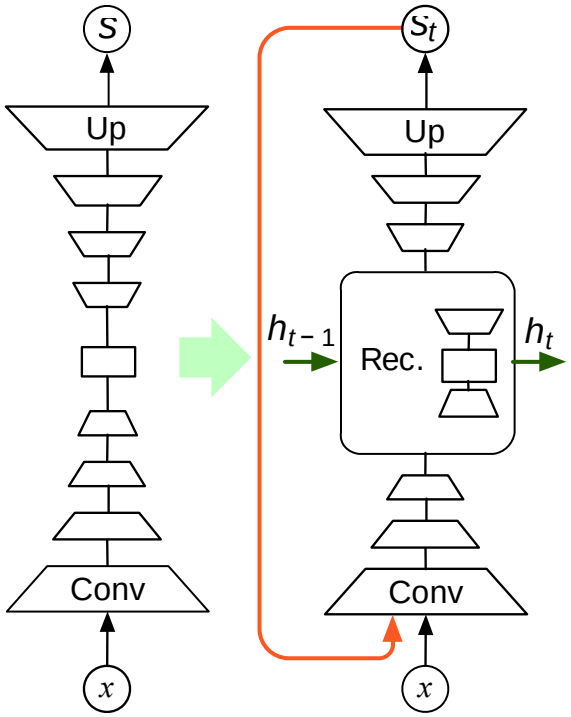
Transposed convolutions

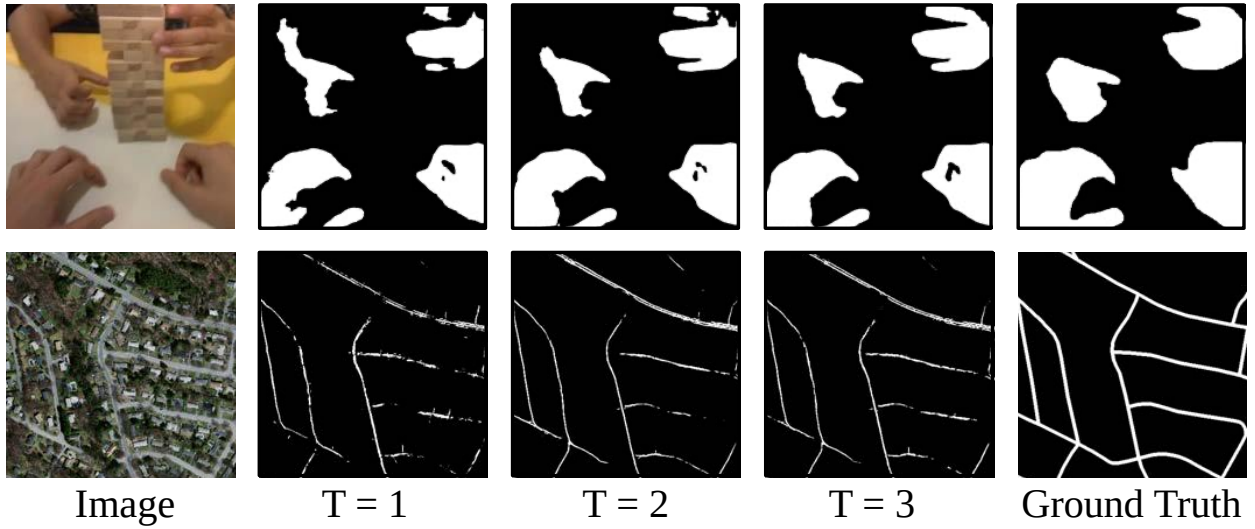


https://towardsdatascience.com/what-is-transposed-convolutional-layer-40e5e6e31c11

In the Street

Recursive Segmentation



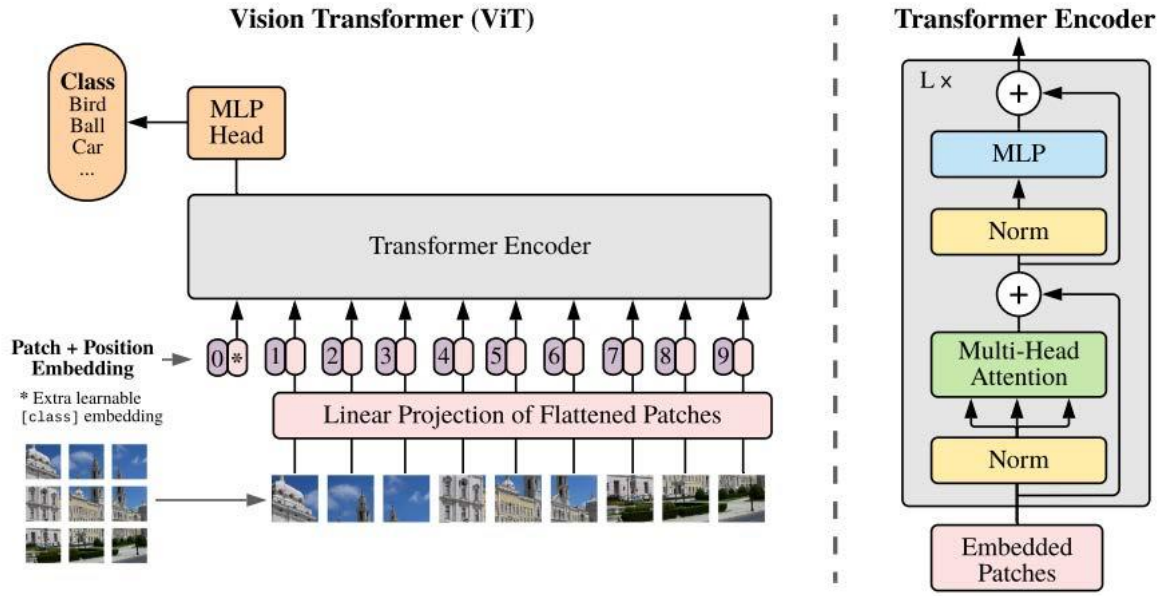


U-Net Rec. U-Net

As for delineation, feeding the output back into the network is an effective way to take context into account.

Wang et al., ICCV'19

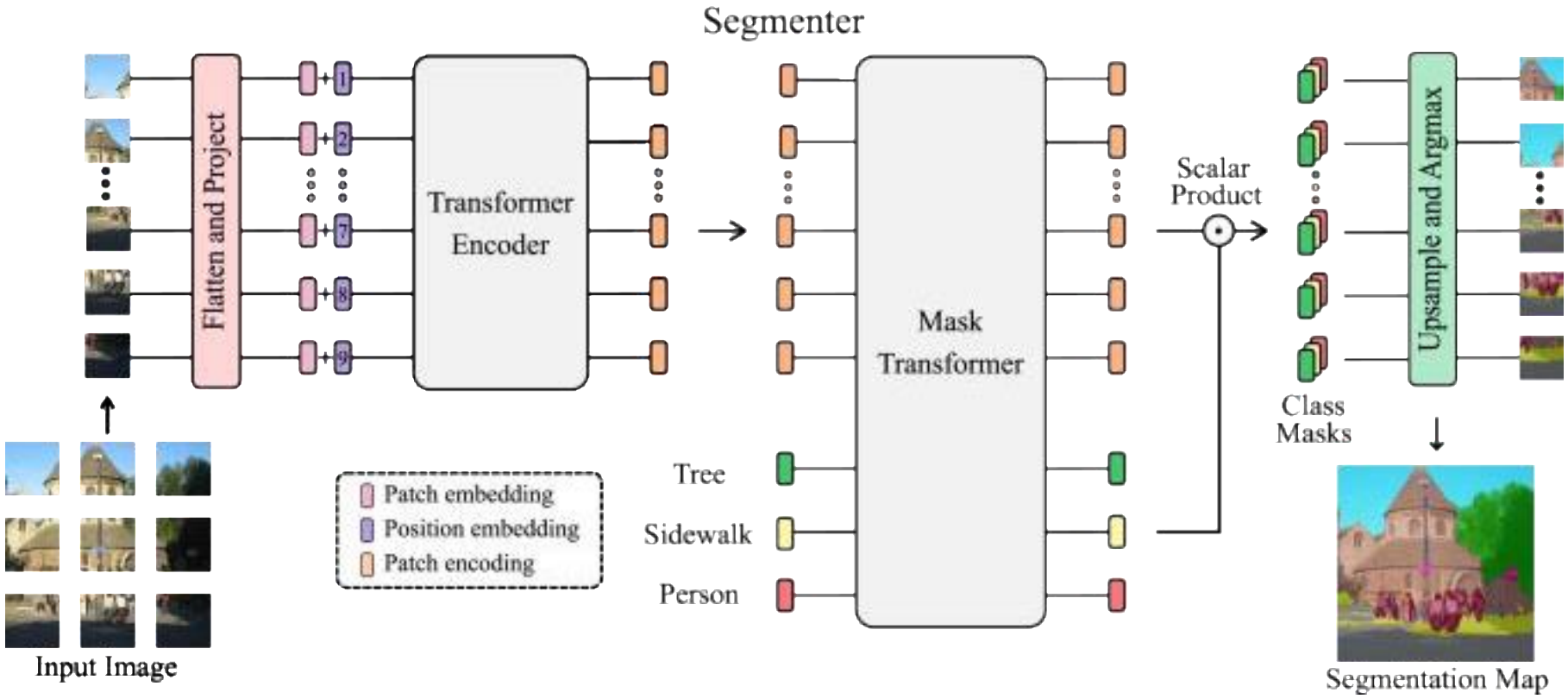
Vision Transformers for Recognition



- Break up the images into square patches.
- Transform each path into a feature vector.
- Feed to a transformer architecture.

Dosovitskiy et al., ICLR'21

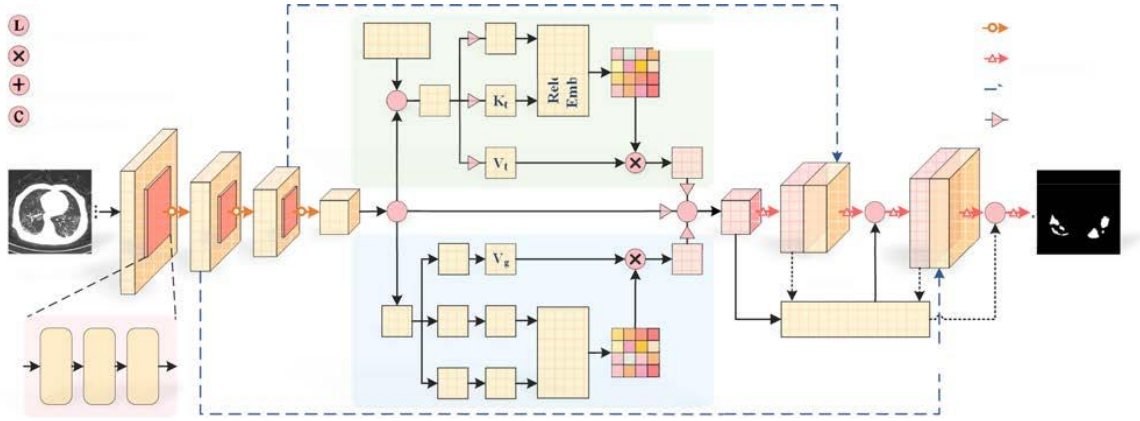
Vision Transformers for Segmentation

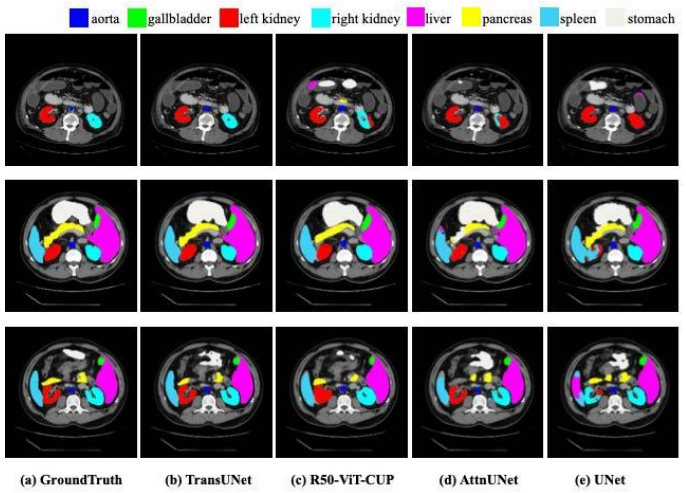


- Replace the "recognition" machinery by a "mask transformer".
- Pros: Good at modeling long range relationships.
- Cons: Flattening the patches looses some amount of information.

Strudel et al., ICCV'21

U-NET + Transformers





- A CNN produces a low-resolution feature vector.
- A transformer operates on that feature vector.
- The upsampling is similar to that of U-Net

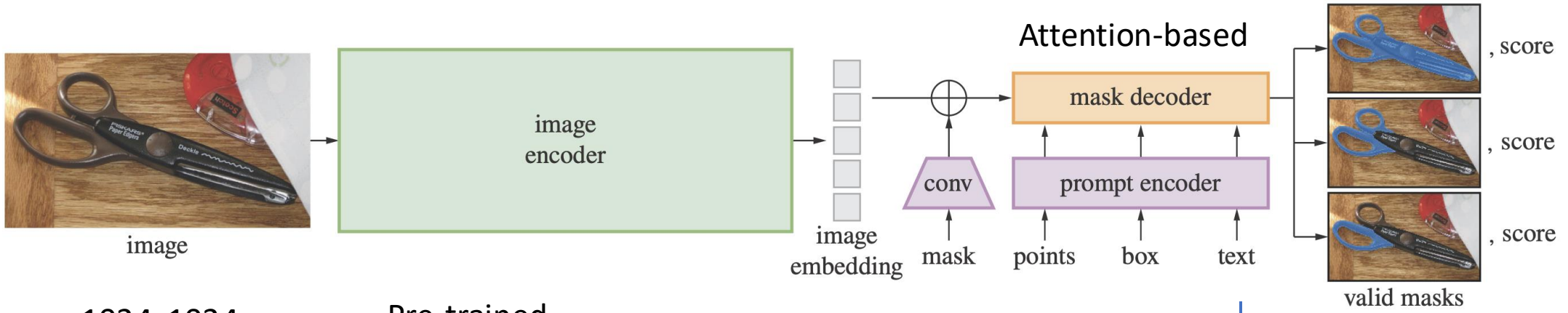
—> Best of both worlds?

Chen et al., TETCI'23

SAM: Segment Anything Model

Meta AI, Kirillov A. et al, 2023

Similar to YOLO, they predict confidence scores as IoU



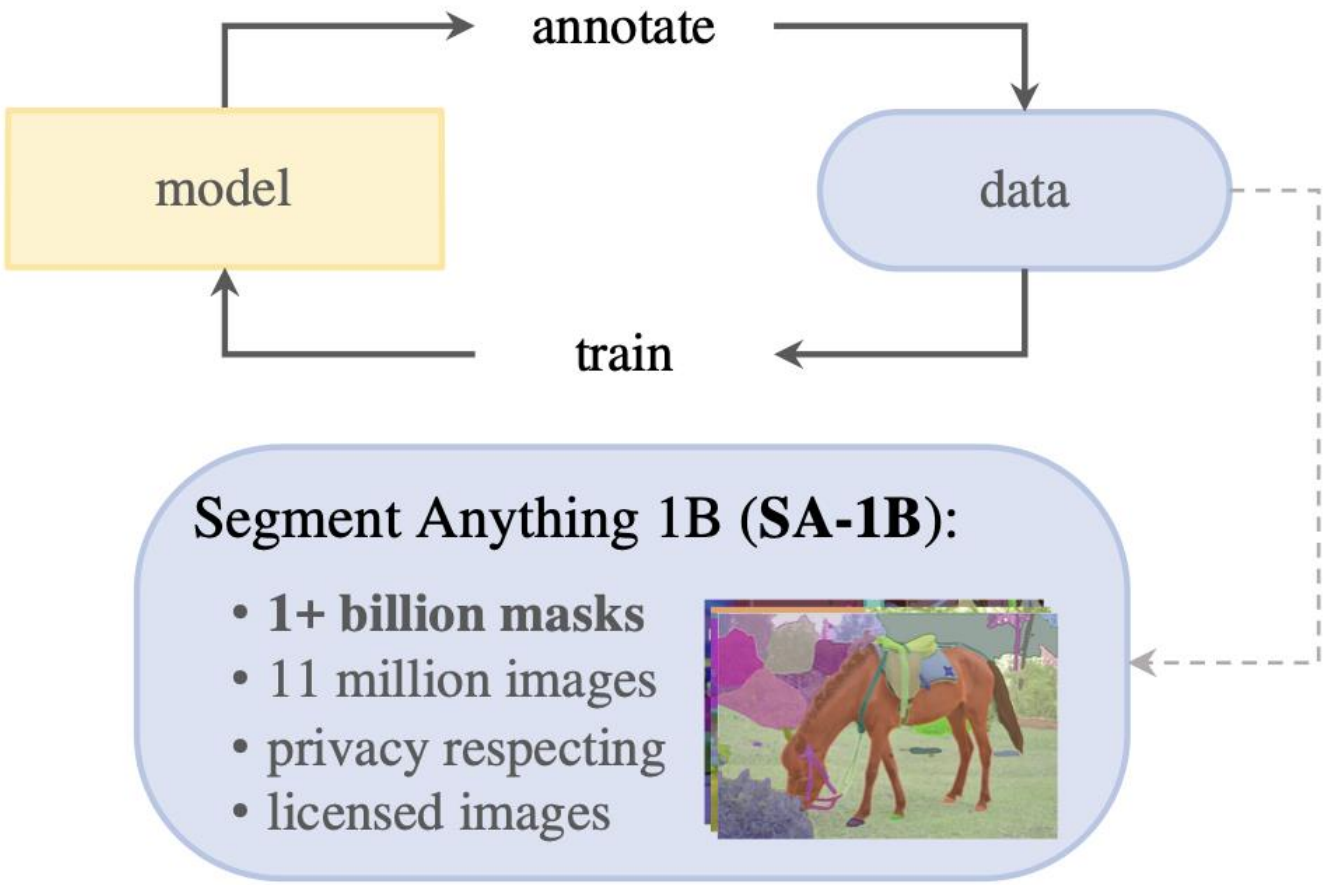
1024x1024

Pre-trained
ViT Masked AutoEncoder
(encoder-only)

Pre-trained
CLIP

Kirillov A. et al, Segment Anything, Meta AI, 2023

SAM: Segment Anything Model



Kirillov A. et al, Segment Anything, Meta AI, 2023

SAM: Segment Anything Model

Kirillov A. et al, Segment Anything, Meta AI, 2023

SAM: Segment Anything Model

Sobel filtering on segmentation masks

Figure 10: Zero-shot edge prediction on BSDS500. SAM was not trained to predict edge maps nor did it have access to BSDS images or annotations during training.

Kirillov A. et al, Segment Anything, Meta AI, 2023

SAM: Segment Anything Model

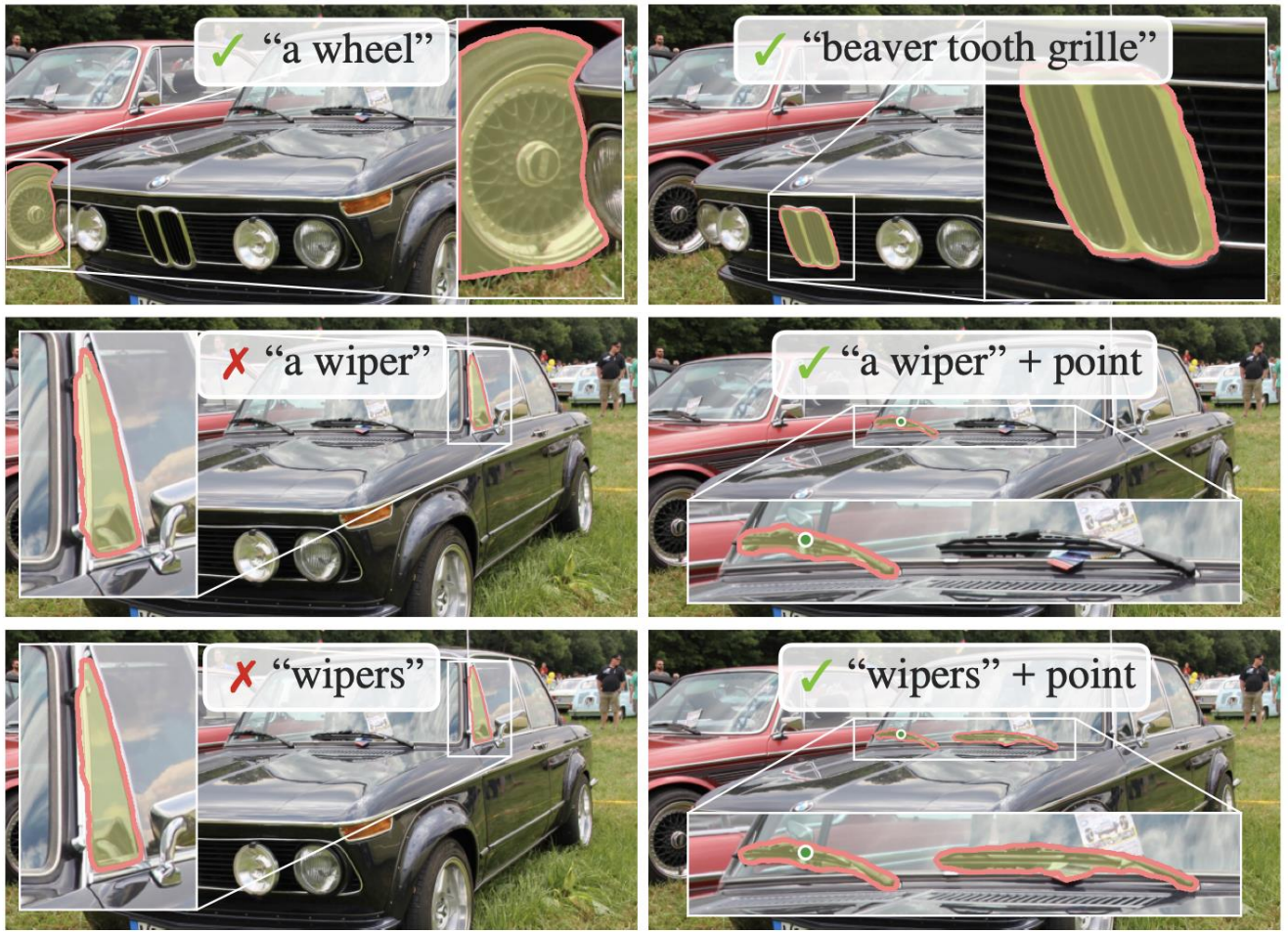


Figure 12: Zero-shot text-to-mask. SAM can work with simple and nuanced text prompts. When SAM fails to make a correct prediction, an additional point prompt can help.

Kirillov A. et al, Segment Anything, Meta AI, 2023

DEMO