Creatis

UNIVERSITÉ DE SHERBROOKE

MEDICAL IMAGING SEGMENTATION USING STATE-OF-THE-ART COMPUTER VISION AND MACHINE LEARNING

BY
PIERRE-MARC JODOIN & CLEMENT ZOTTI

May 2016

SUMMARY

- Segmentation methods
- Neural networks
- Convolutional neural networks
- Results

SEGMENTATION METHODS

Image segmentation

Features : raw pixel values
Method : manual threshold (no learning)

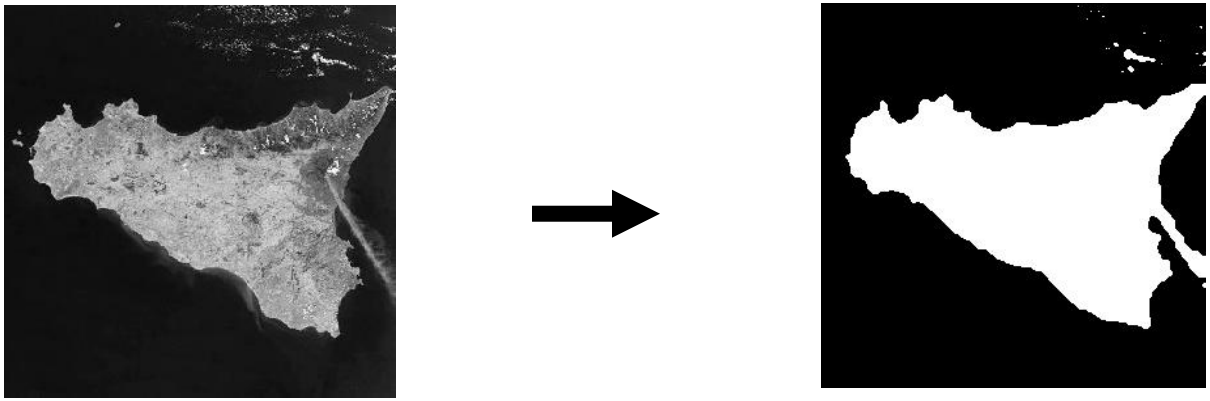


Image segmentation

Features : raw pixel values
Method : active contours (no learning)

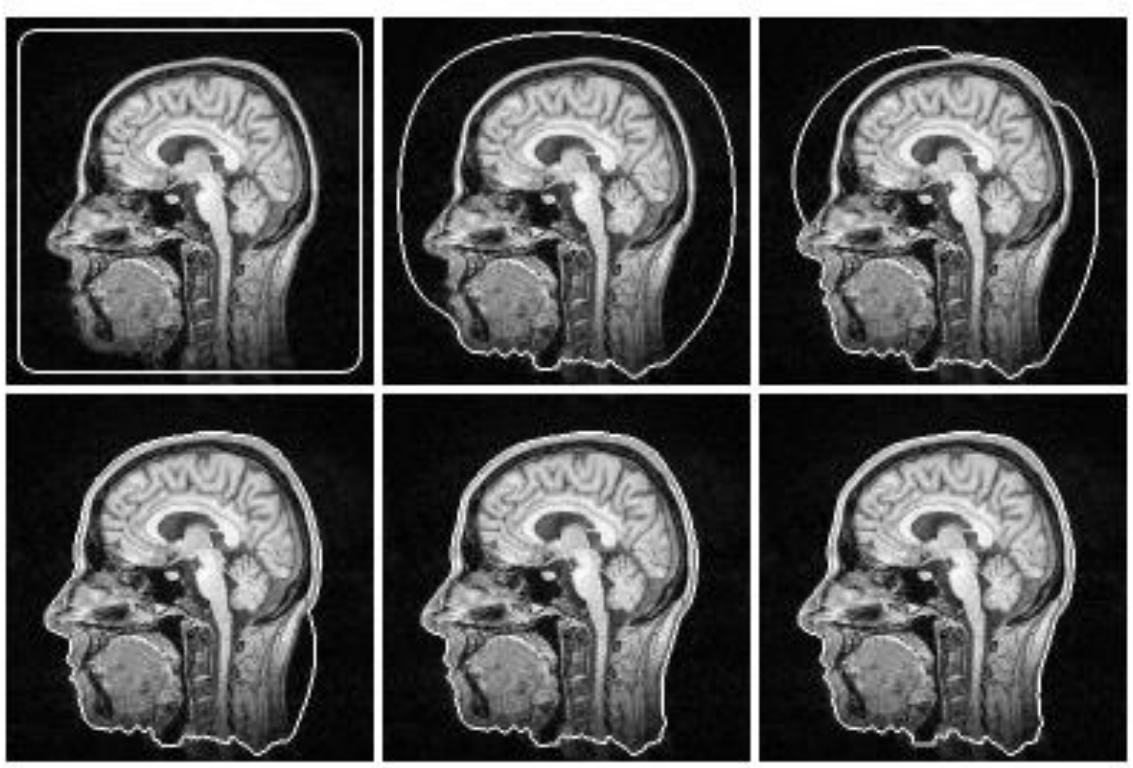
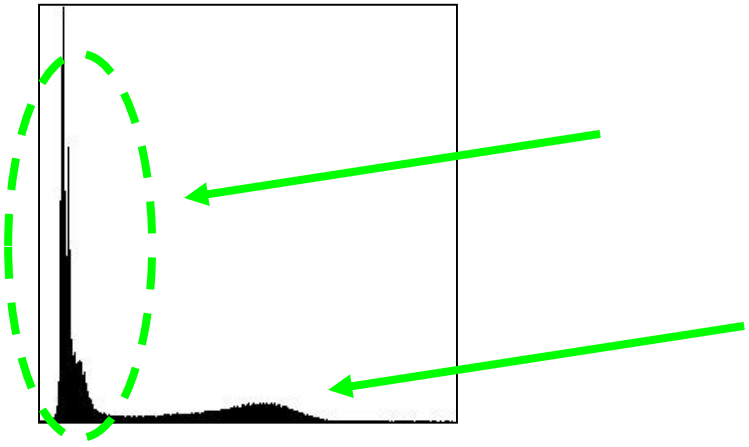


Image segmentation

Features : raw pixel values
Method : K-Means (no learning)

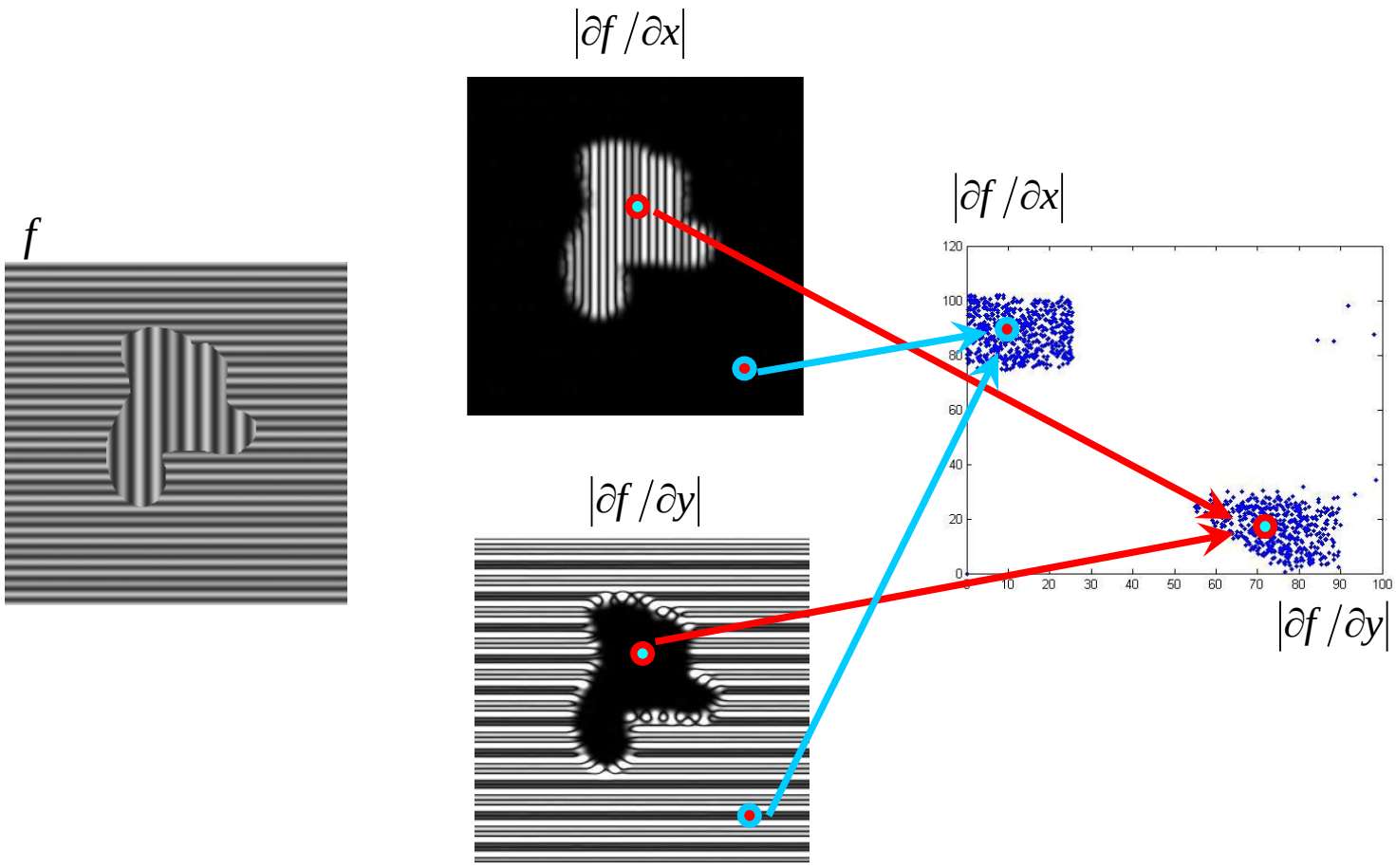
Label field : white = land, black = sea

Histogram



Bummer

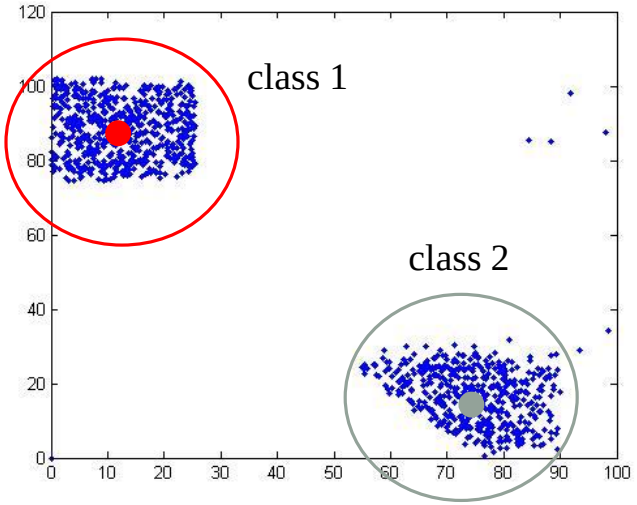
Features : ~~raw pixel values~~ **hand-designed features**
Method : k-means (no learning)



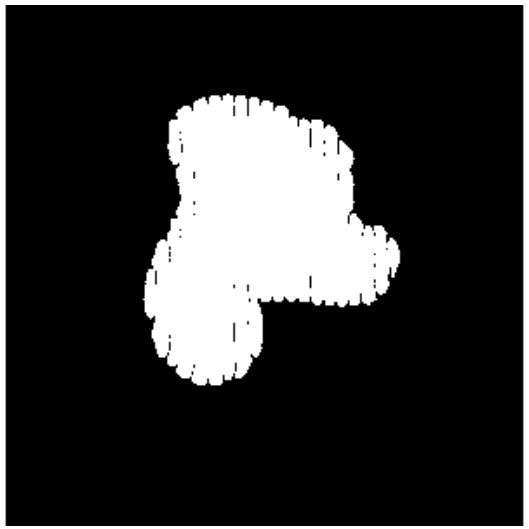
Bummer

Features : ~~raw pixel values~~ **hand-designed features**
Method : k-means (no learning)

$|\partial f / \partial x|$

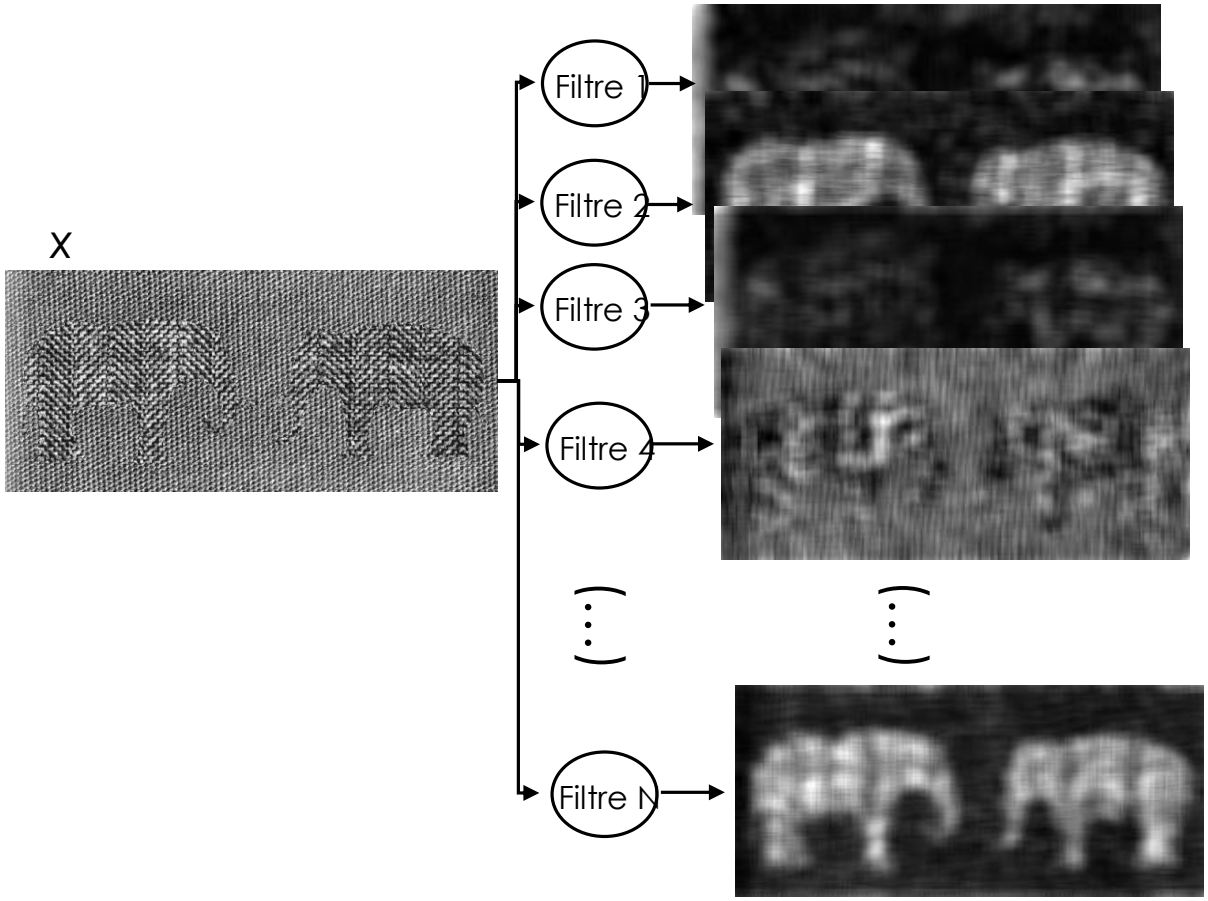


$|\partial f / \partial y|$

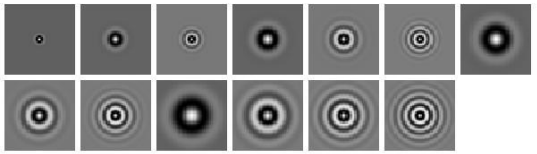


Bummer again!

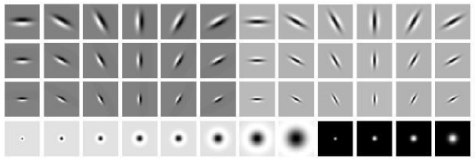
Features : which hand-designed features?
Method : hand-designed approach (no learning)



« texton filters »



« *steerable filters* »



Gabor filters

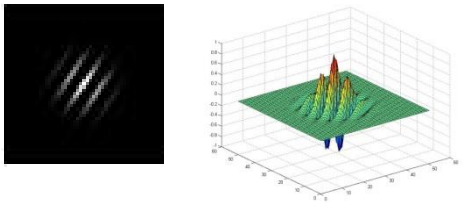
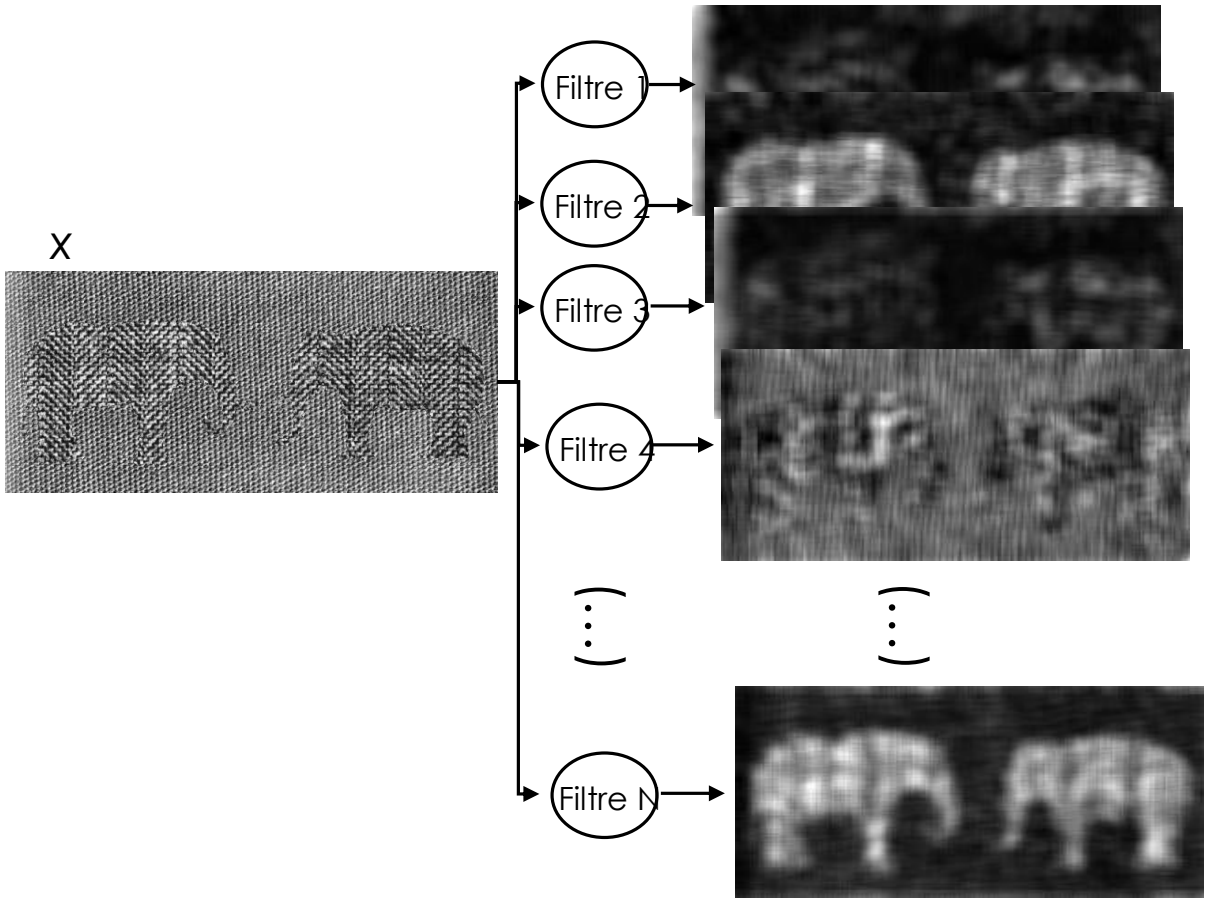


Image Segmentation

Features : texture features (filters)
Method : K-means (no learning)



K-means

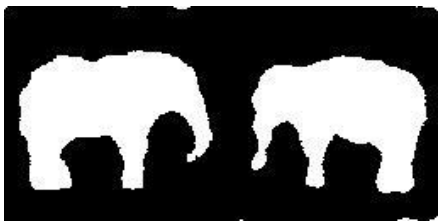
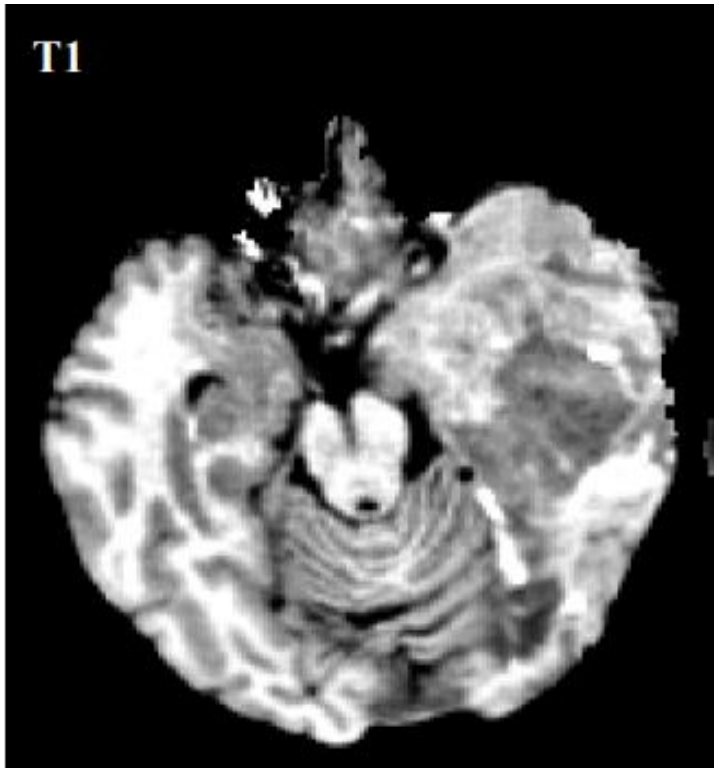


Image Segmentation

Sometimes, there are no solutions…



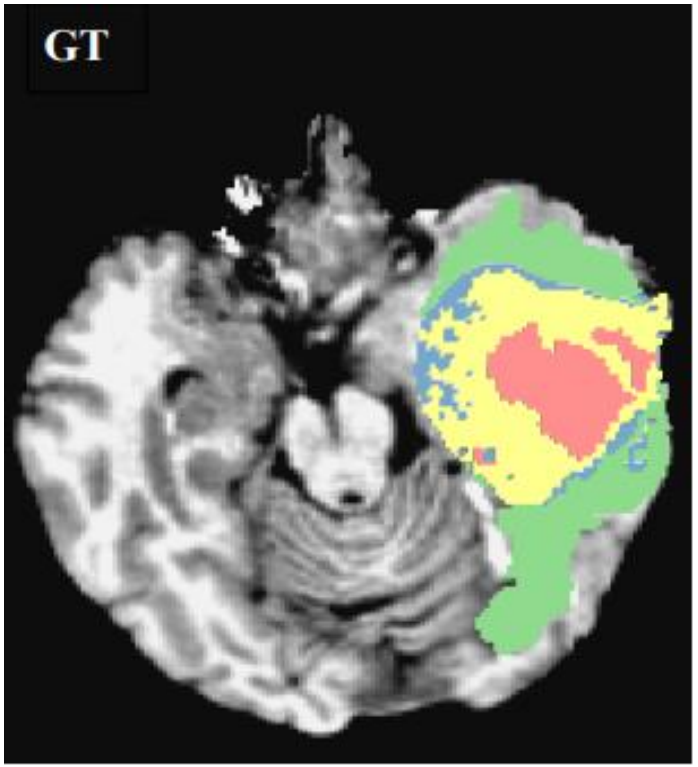
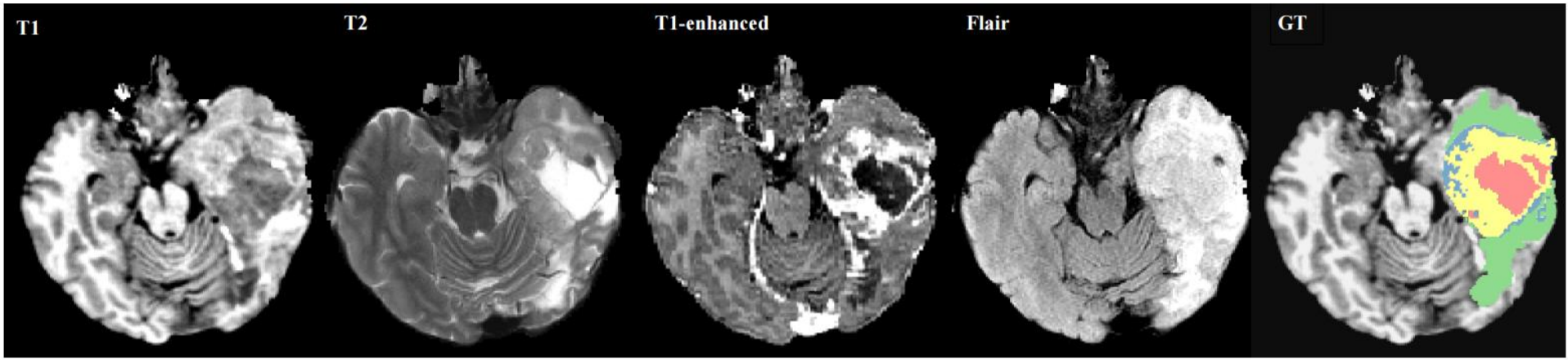


Image Segmentation

Only way out : more modalities

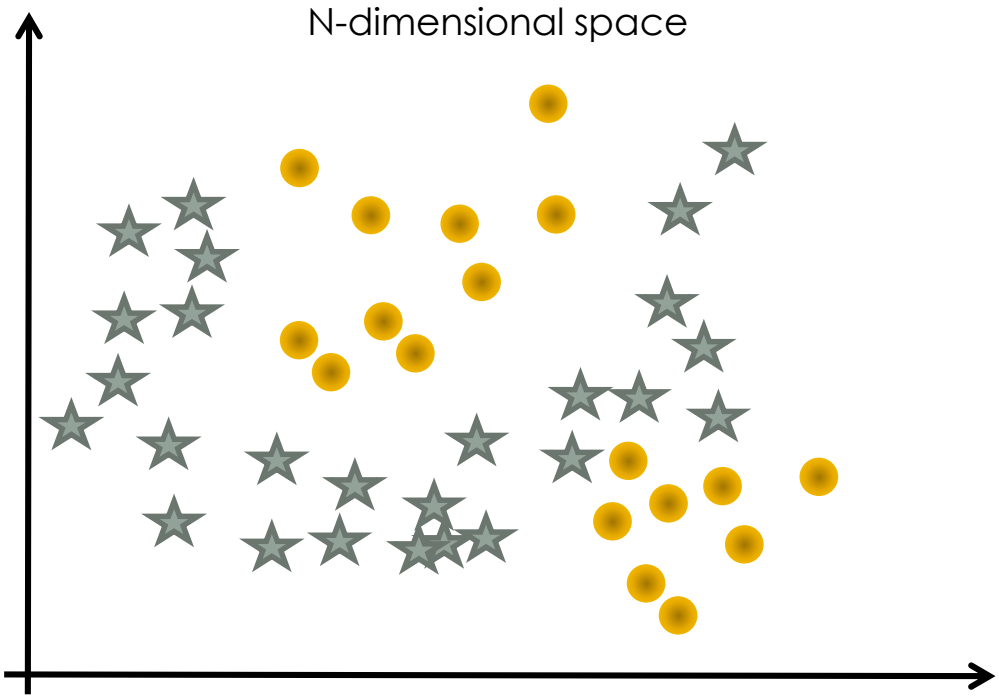
Features : hand-designed features
Method : k-means (no learning)



Again bummer!

Data may follow an unknown non-Gaussian distribution

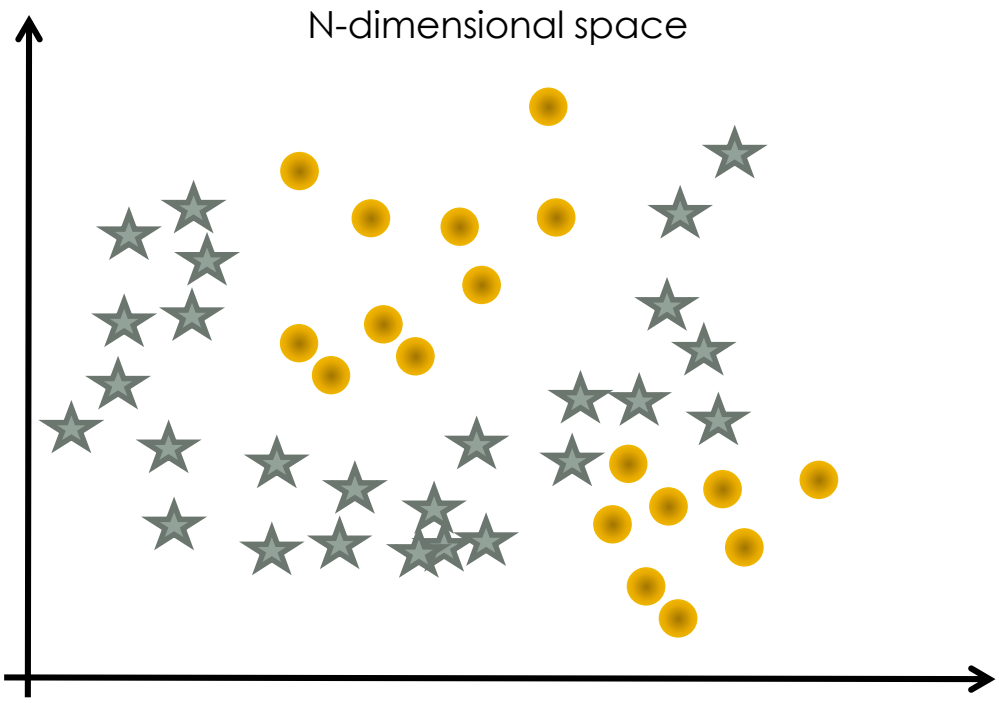
Features : hand-designed features
Method : k-means (no learning)



Again bummer!

Solution 1 : through more modalities!

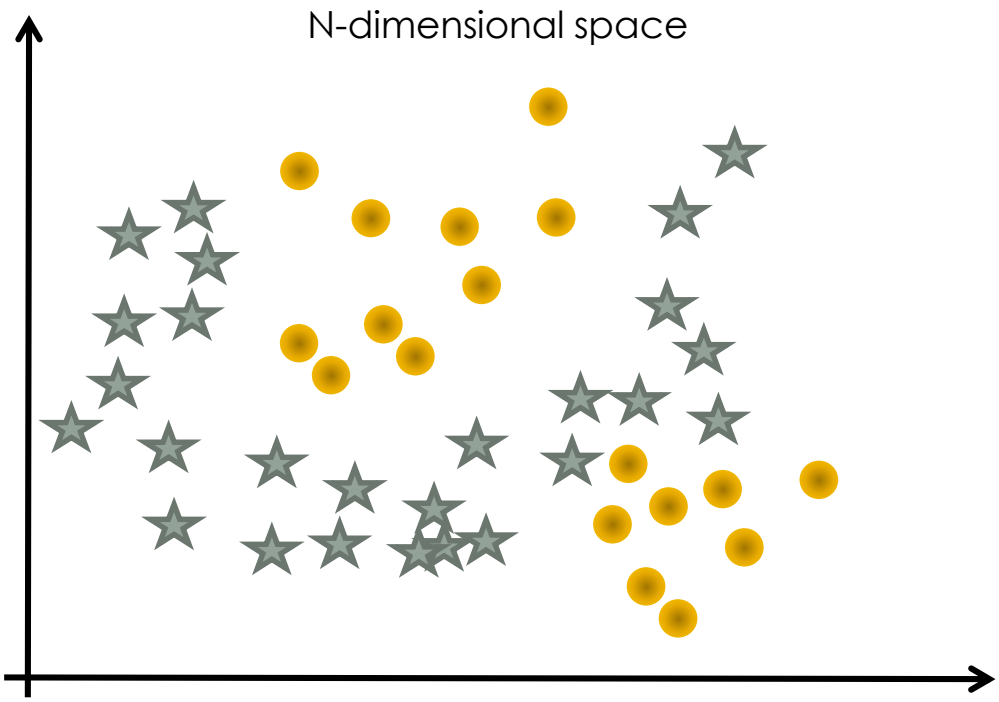
Features : hand-designed features
Method : k-means (no learning)



Again bummer!

Solution 2

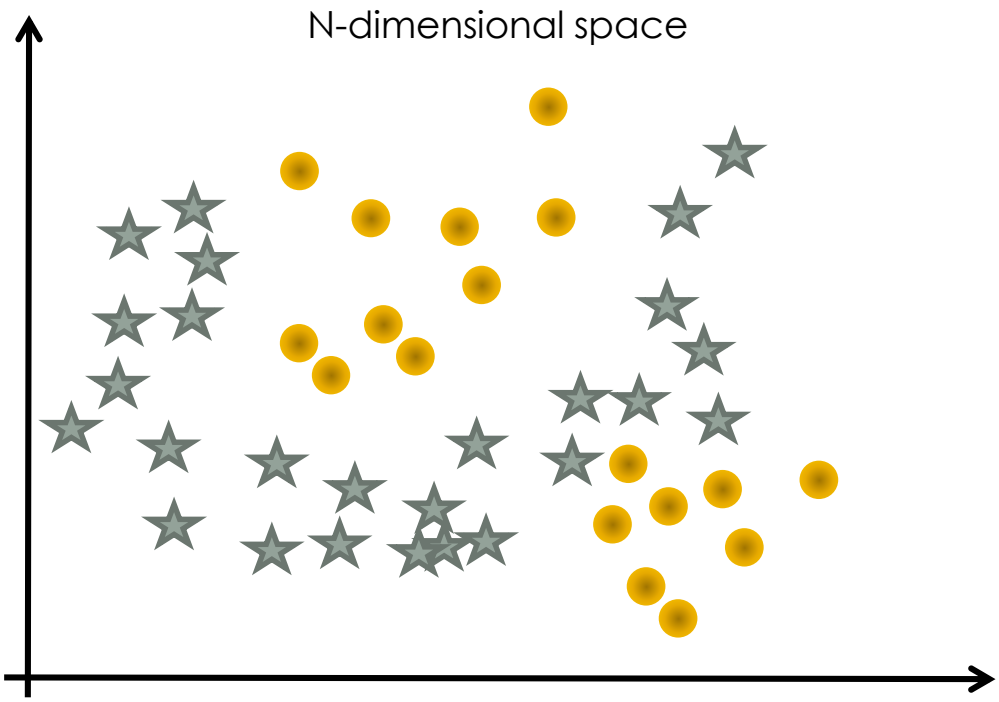
Features : **find more and/or better features**
Method : k-means (no learning)



Again bummer!

Solution 3

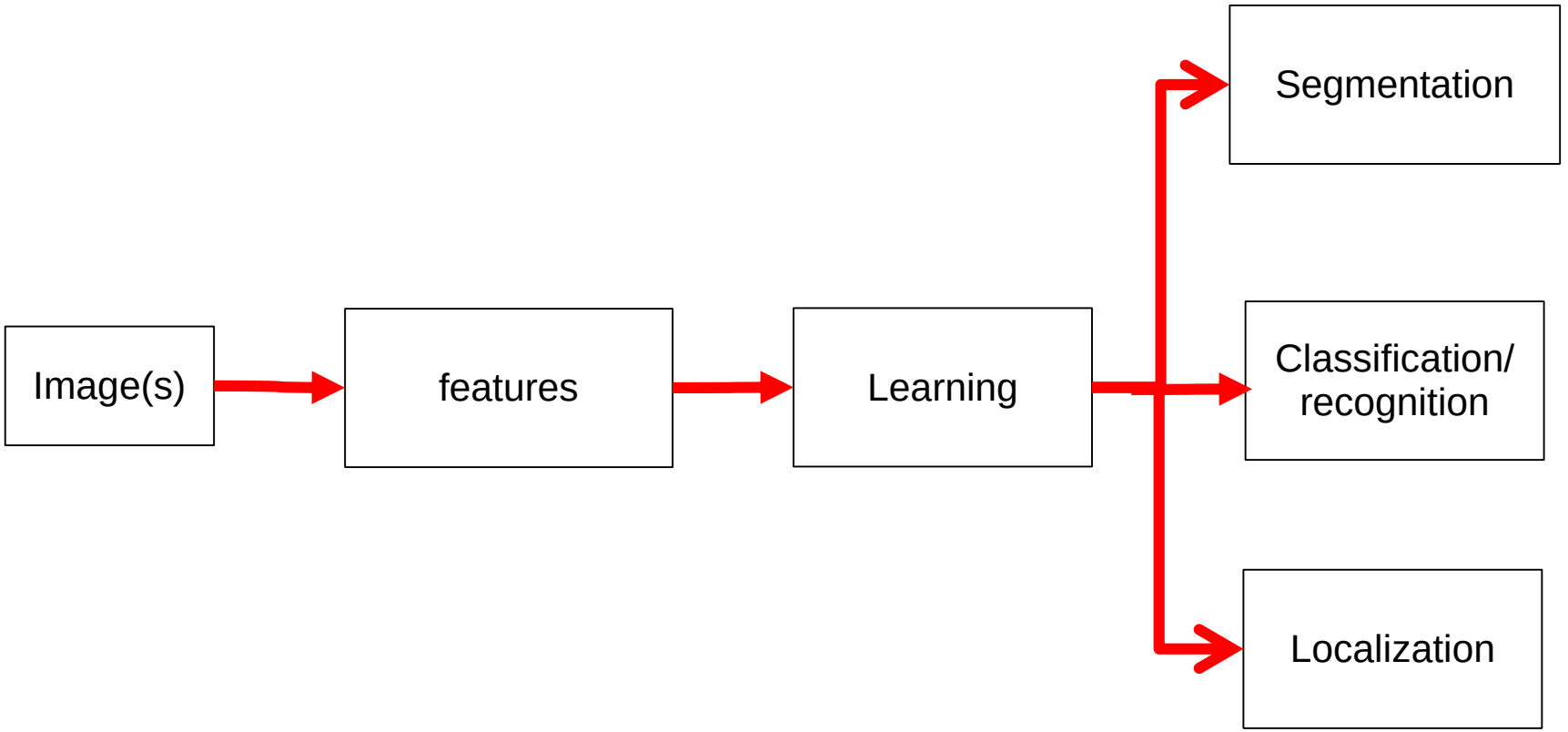
Features : hand-designed features
Method : **machine learning**



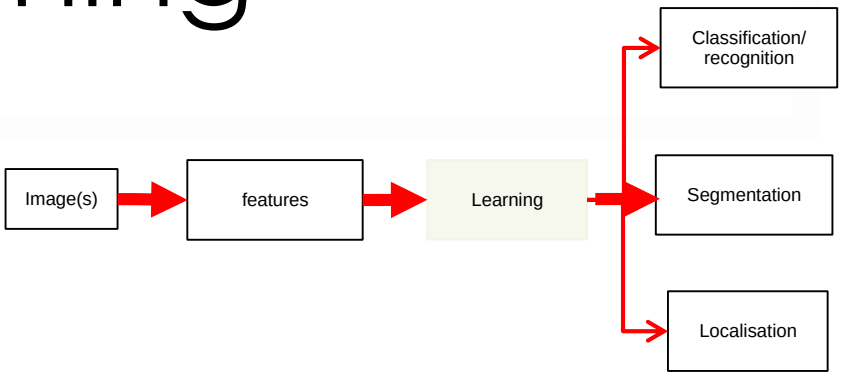
NEURAL NETWORKS

Supervised learning

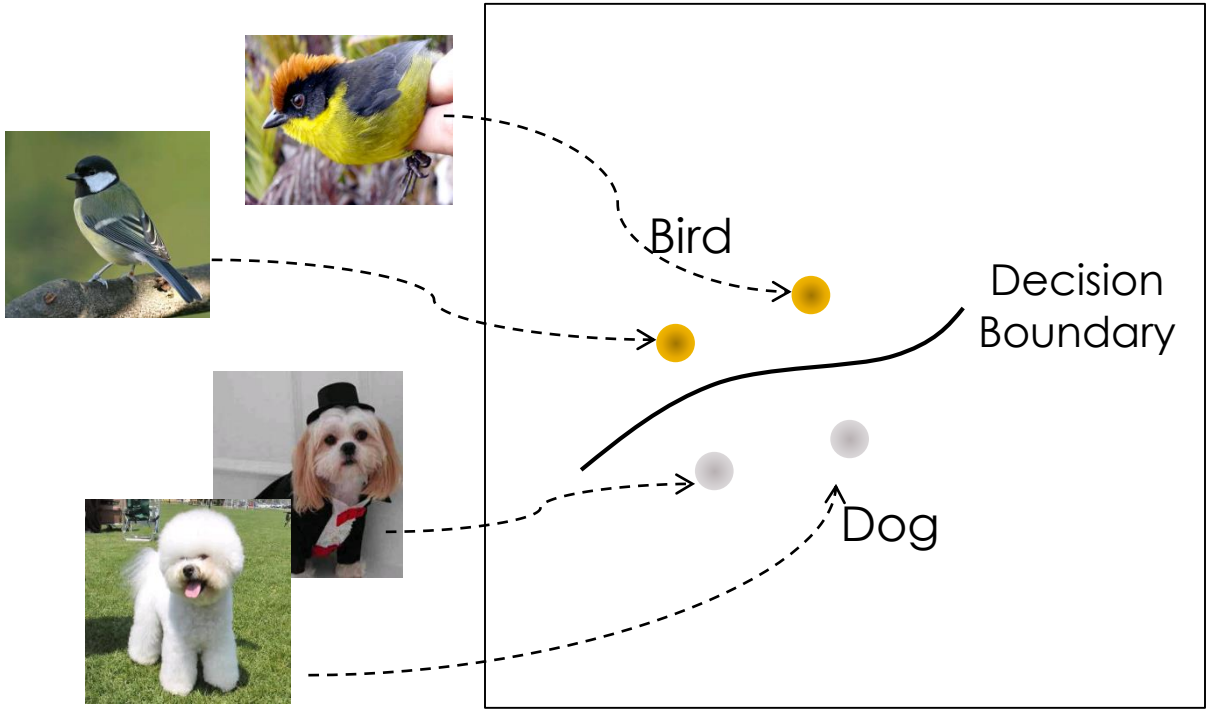
Since 1960



Supervised Learning



Classification / recognition



Supervised Learning

Training set: containing *n* data $Z = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_n, y_n)\}$

$\mathbf{x}_i \in \Re^d$ feature vector of the *i-th* training data

$y_i \in \{+1, -1\}$ is the class label of the *i-th* training data

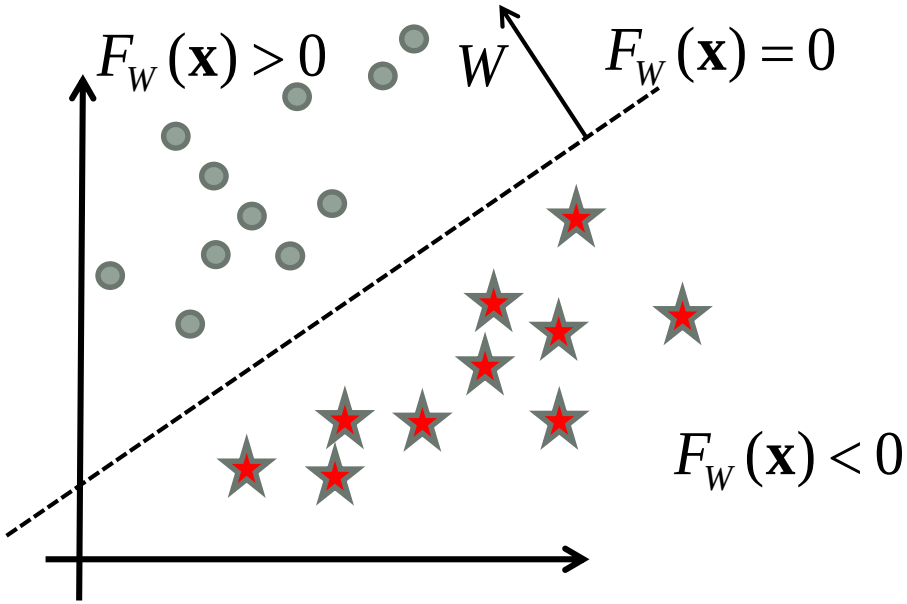
Functions: given Z, we need to *learn* a **classification function**

$$F_W : \Re^d \rightarrow \{-1, +1\}$$

$$F_W(\mathbf{x}) = \begin{cases} +1 & \text{if } \mathbf{x} \text{ is in class } +1 \\ -1 & \text{otherwise.} \end{cases}$$

W are parameters to be estimated.

Hyperplane

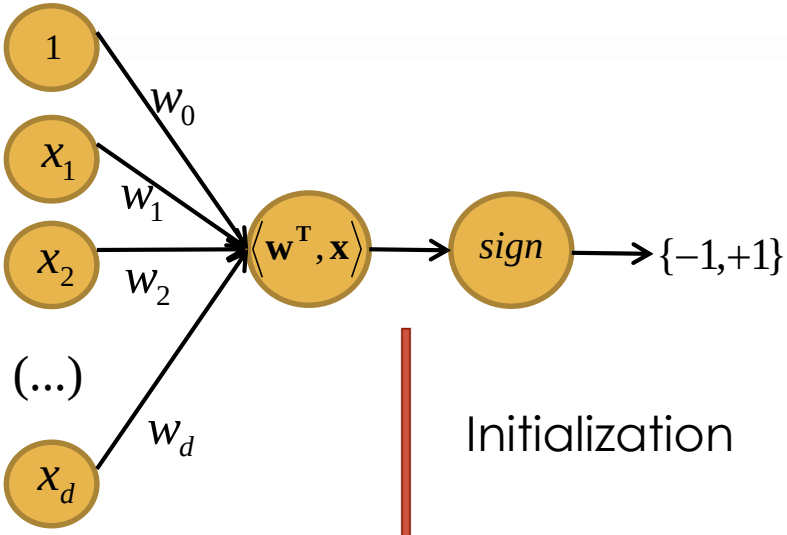


$$x_0 = 1$$

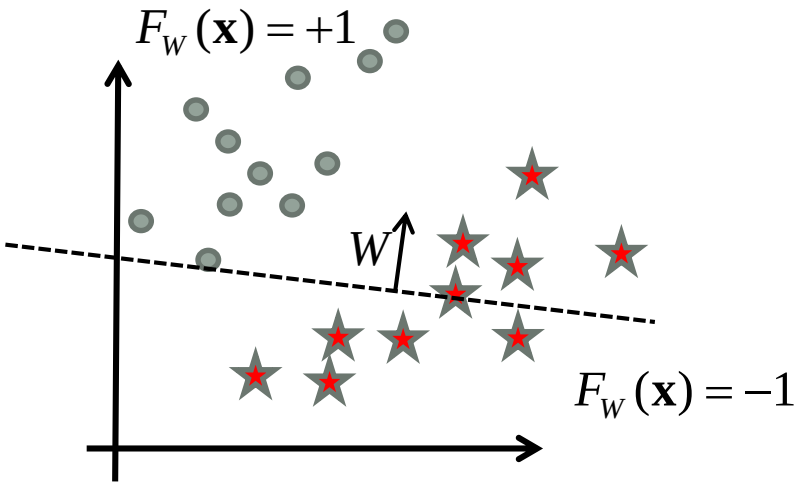
$$F_W(\mathbf{x}) = w_1 x_1 + w_2 x_2 + w_0 x_0$$

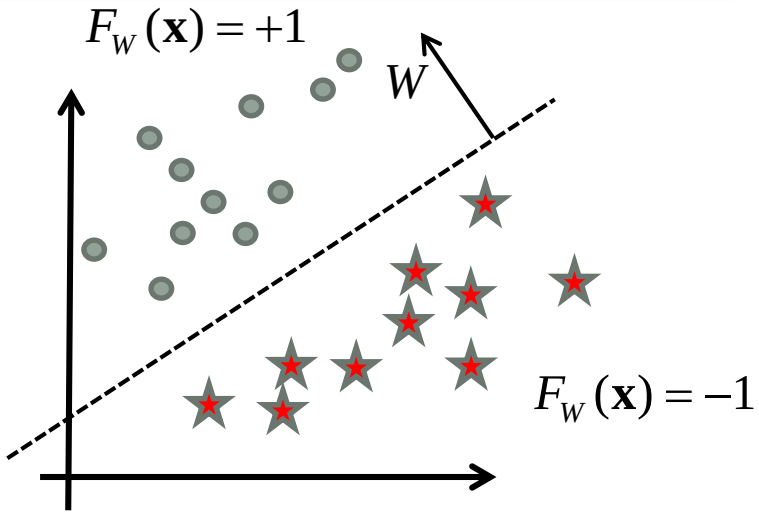
$$= \left\langle \mathbf{w}^{\mathbf{T}}, \mathbf{x} \right\rangle$$

Perceptron (1957)



Initialization





Training

Training

Goal: given training data $Z = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_n, y_n)\}$, estimate W such that

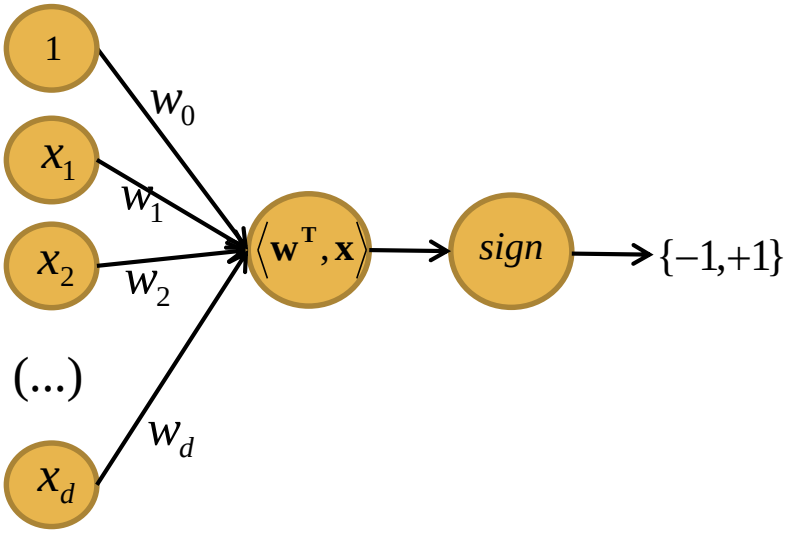
$$F_W(\mathbf{x}_i) = y_i \qquad \forall i$$

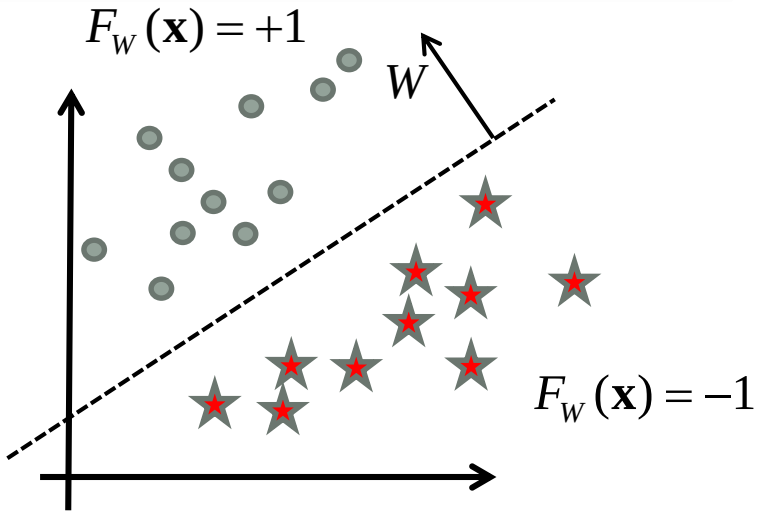
In other words, minimize the **training error**

$$E(W) = \frac{1}{n} \sum_{i=1}^{n} l(F_W(\mathbf{x}_i), y_i)$$

$$W = \arg\min_W = \frac{1}{n} \sum_{i=1}^{n} l(F_W(\mathbf{x}_i), y_i)$$

Perceptron (1957)

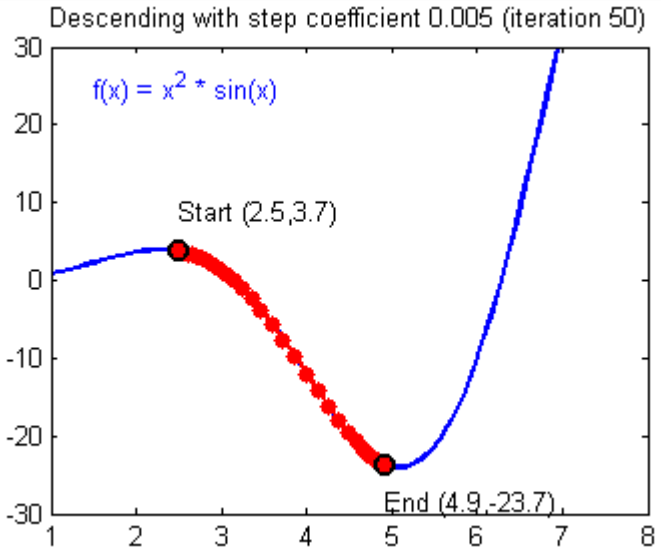




$$E(W) = \sum_{\mathbf{x}_i \in \Psi} - y_i \langle \mathbf{w}^T, \mathbf{x}_i \rangle$$

where Ψ is the set of all wrongly classified samples

Perceptron (1957)



Gradient descent.

$$W^{[k+1]} = W^{[k]} - \eta \nabla E\left(W^{[k]}\right)$$

Gradient of the error function

Step size or "learning rate".

Batch perceptron

Initialize W
k=0
DO k=k+1

$$W = W - \eta \sum_{\mathbf{x}_i \in \Psi} - y_i \mathbf{x}_i$$

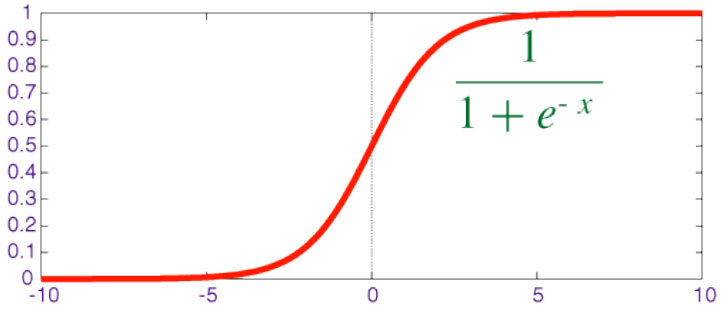
UNTIL all samples are correctly classified

Probabilistic formulation

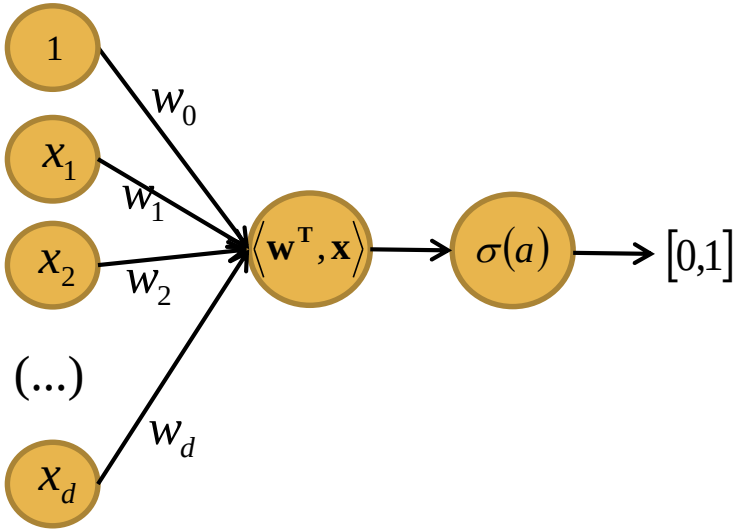
From sign to sigmoid

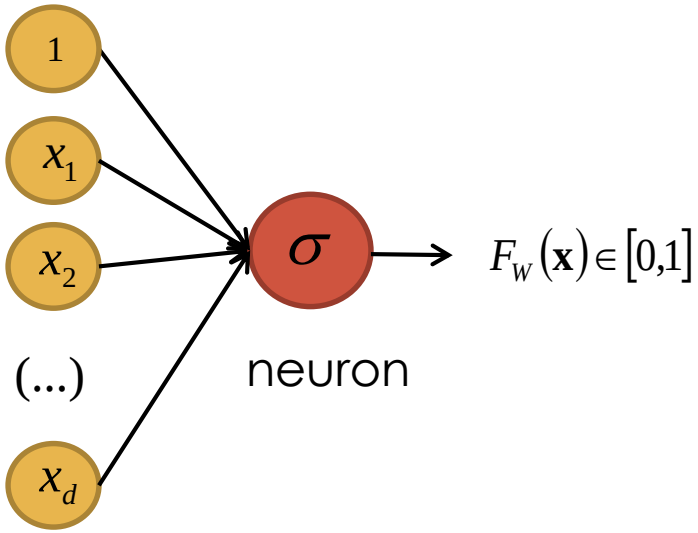
$$p(C_1|\mathbf{x}) = \sigma(a) = \frac{1}{1+\exp(-a)} \qquad a = \frac{p(\mathbf{x}|C_1)p(C_1)}{p(\mathbf{x}|C_2)p(C_2)}$$

Sigmoid

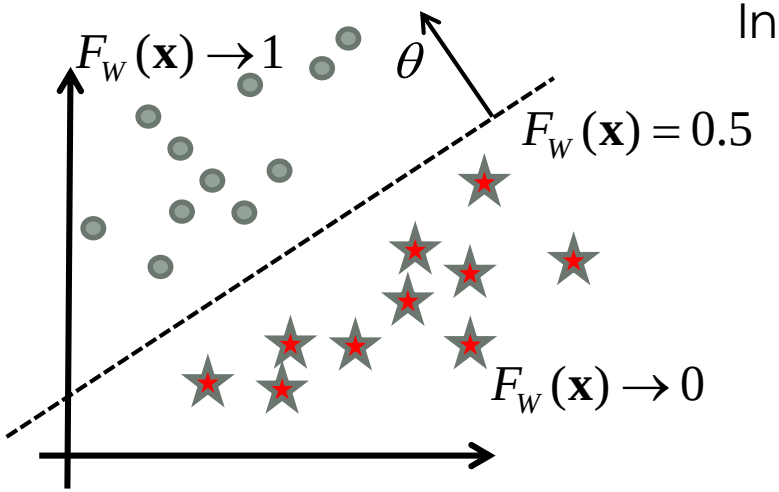


Neuron nonlinearity= activation function



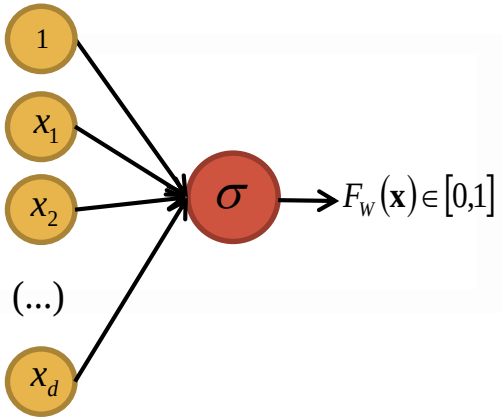


Input layer



Error function

Cross-entropy



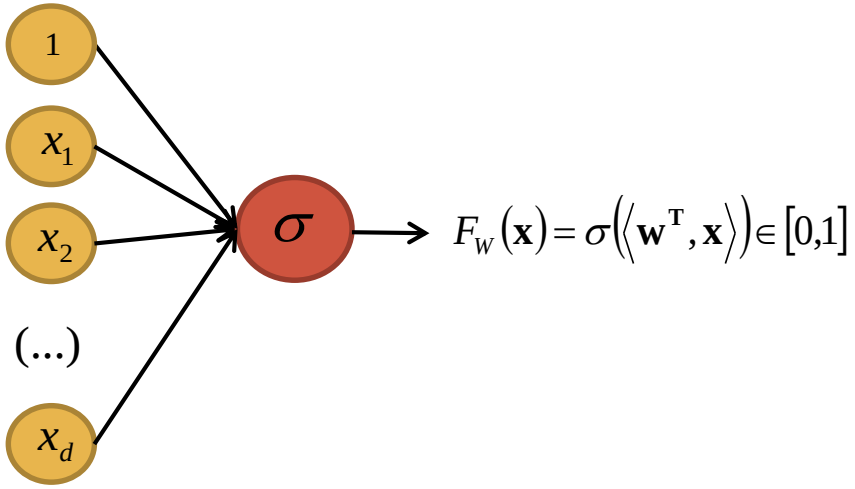
Try to minimize the negative log-likelihood

$$E(W) = -\ln P(\mathbf{y} \mid X)$$

$$= \sum_{i=1}^{n} y_i \ln F_W(\mathbf{x}_i) + (1 - y_i) \ln(1 - F_W(\mathbf{x}_i))$$

Single Layer Network

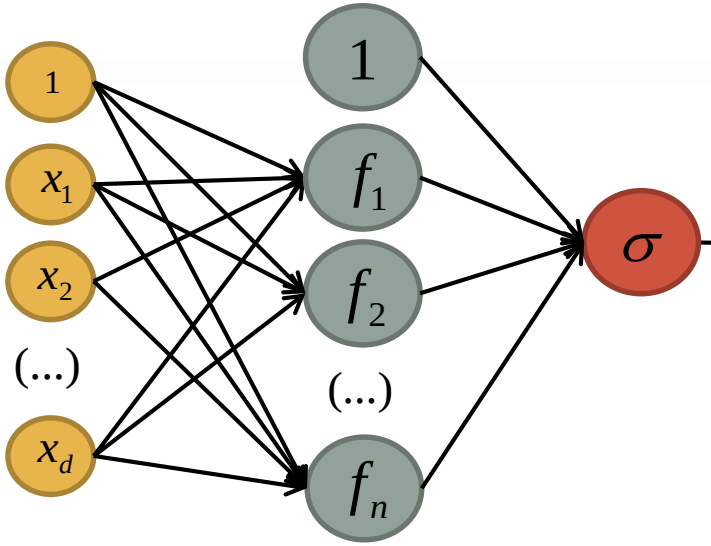
Cross-entropy



$$E(W) = \sum_{i=1}^{n} y_i \ln F_W(\mathbf{x}_i) + (1 - y_i) \ln(1 - F_W(\mathbf{x}_i))$$

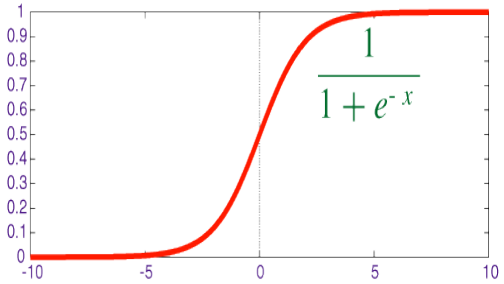
$$W^{[k+1]} = W^{[k]} - \eta \nabla E\left(W^{[k]}\right)$$

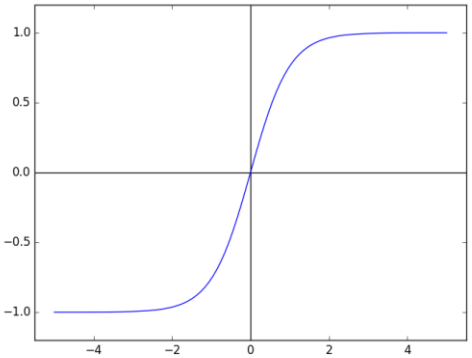
Multi Layer Network

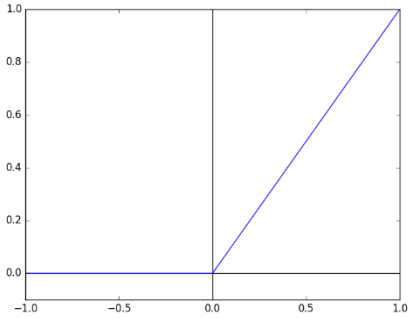


$$F_W = \sigma(\mathbf{w}_1^{\mathbf{T}} \cdot h(W_2^{\mathbf{T}} \cdot \mathbf{x}))$$

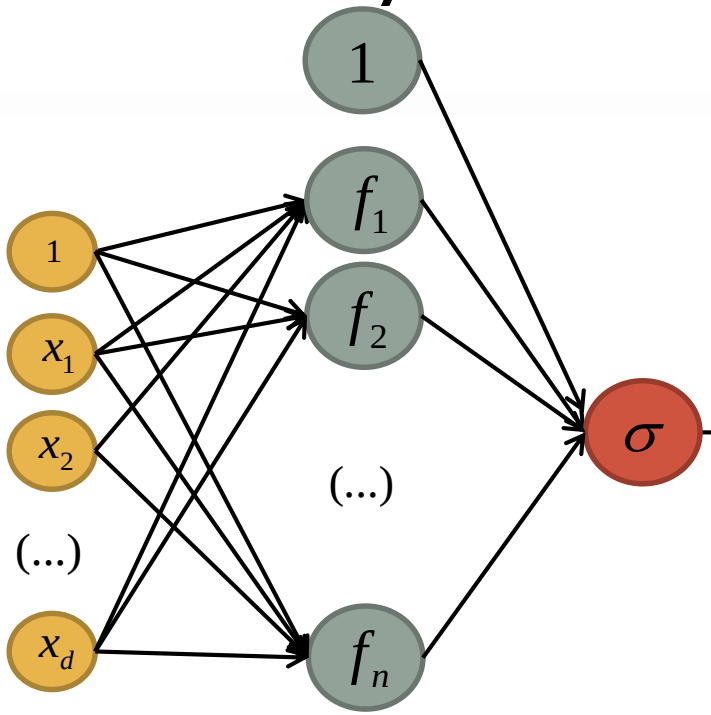
h : activation function(σ(a), tanh(a) or ReLu(a))







Multi Layer Network



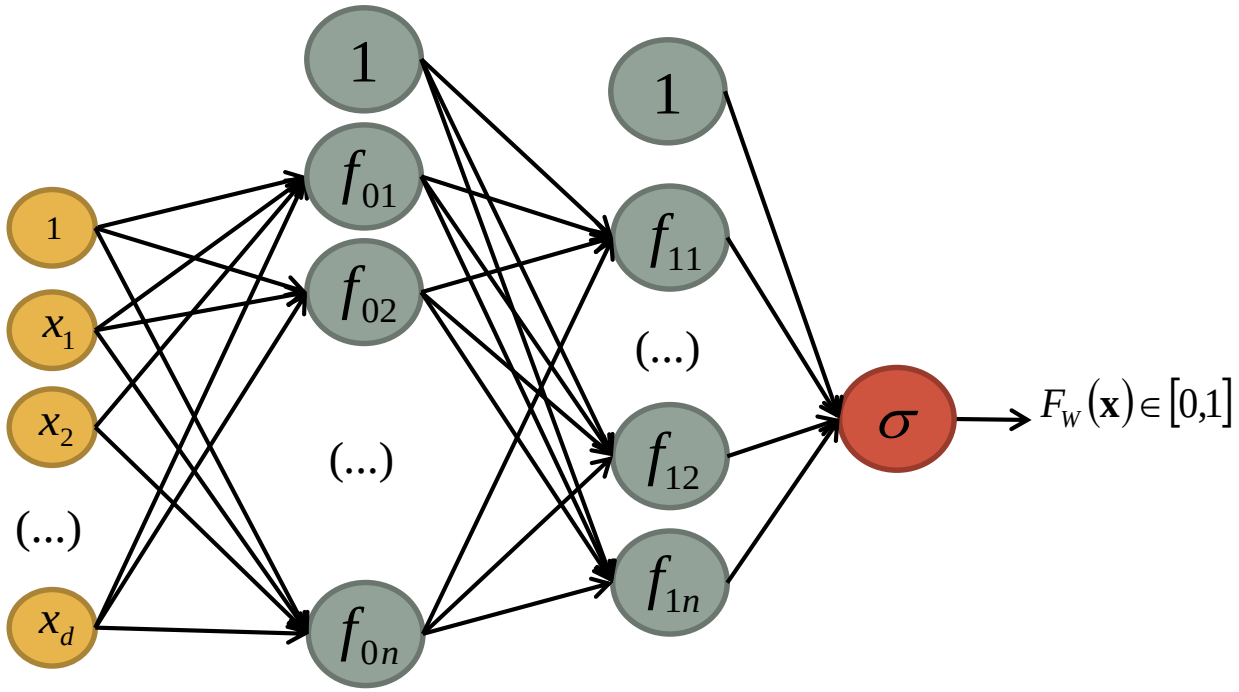
$$F_W = \sigma(\mathbf{w}_1^{\mathbf{T}} \cdot h(W_2^{\mathbf{T}} \cdot \mathbf{x}))$$

$$E(W) = \sum_{i=1}^{n} y_i \ln F_W(\mathbf{x}_i) + (1 - y_i) \ln(1 - F_W(\mathbf{x}_i))$$

$$W^{[k+1]} = W^{[k]} - \eta \nabla E\left(W^{[k]}\right)$$

Back propagation

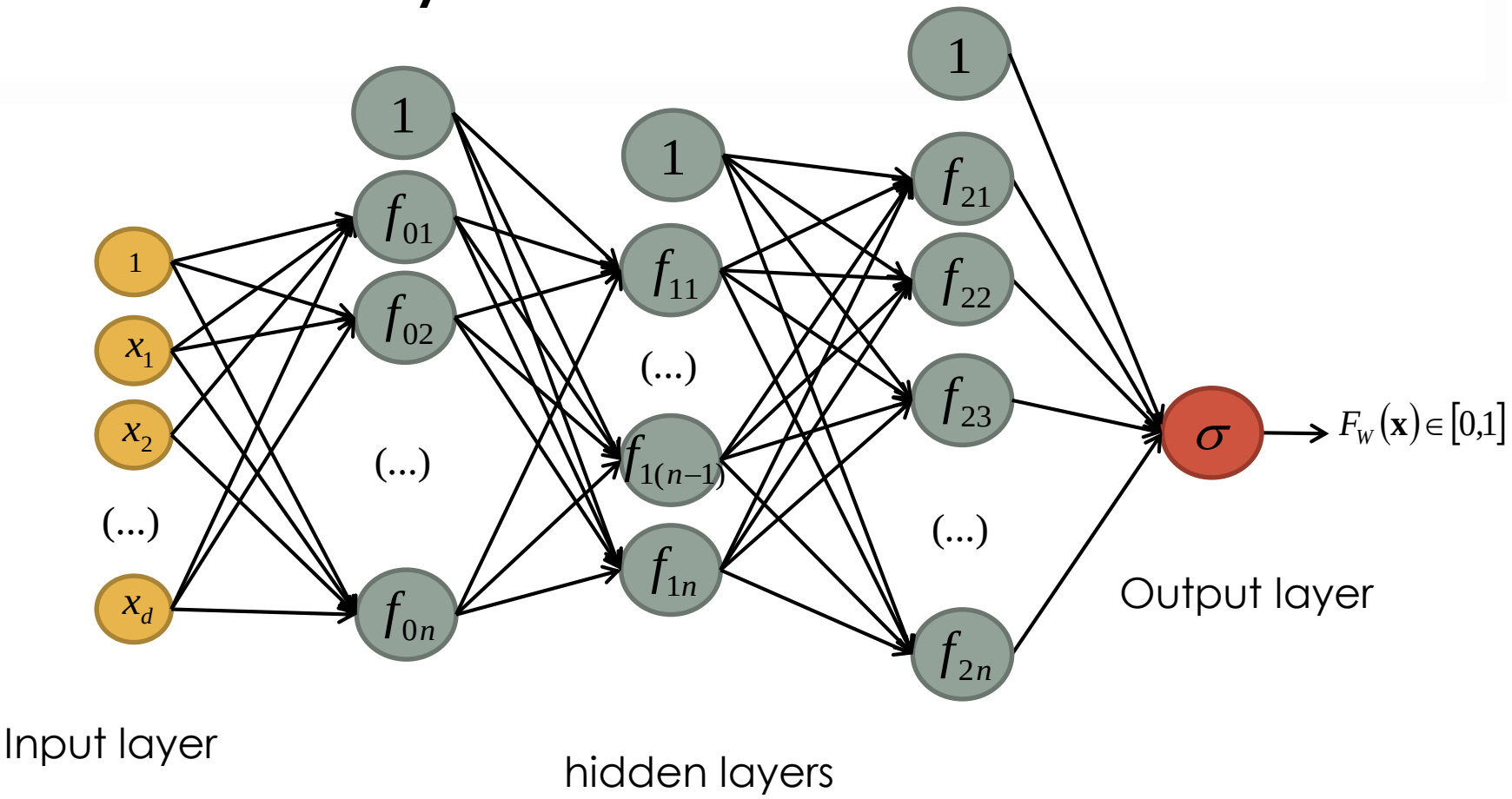
Multi Layer Network



$$E(W) = \sum_{i=1}^{n} y_i \ln F_W(\mathbf{x}_i) + (1 - y_i) \ln(1 - F_W(\mathbf{x}_i))$$

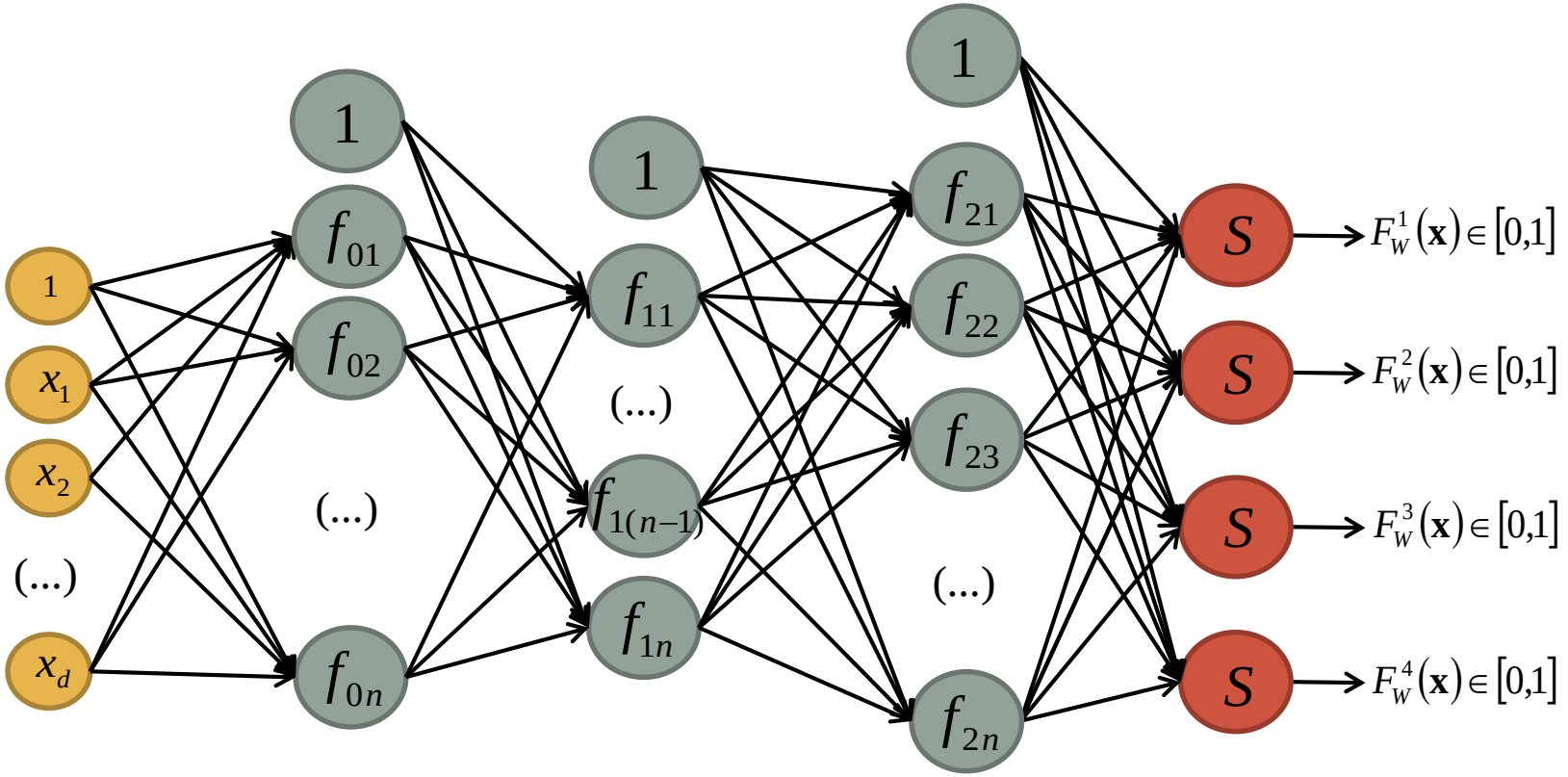
$$W^{[k+1]} = W^{[k]} - \eta \nabla E\left(W^{[k]}\right)$$

Multi Layer Network



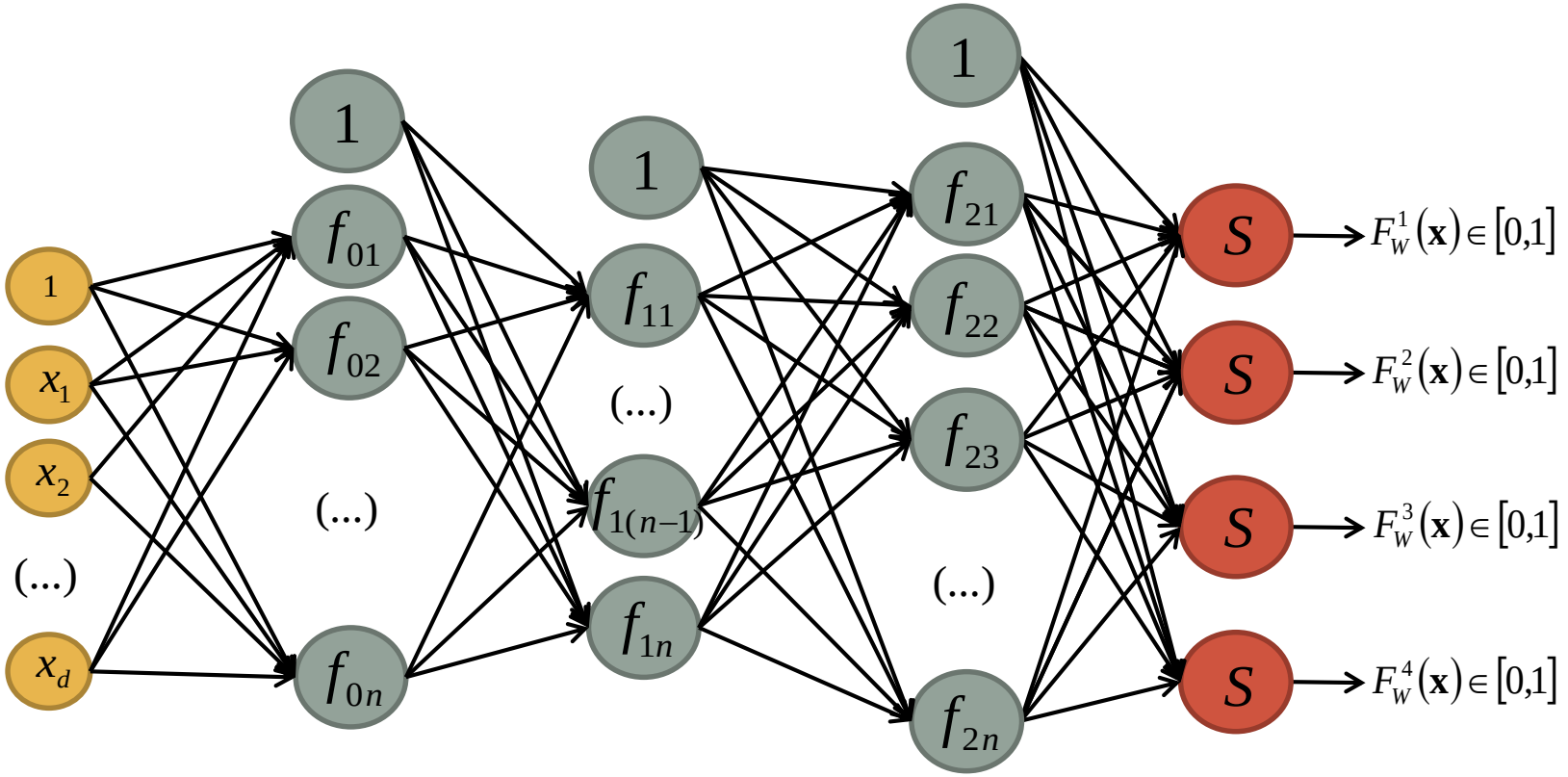
All fully-connected layers

N-Class segmentation



Softmax $F_W^i(\mathbf{x}) = \dfrac{\exp(a_i)}{\sum_k \exp(a_k)}$

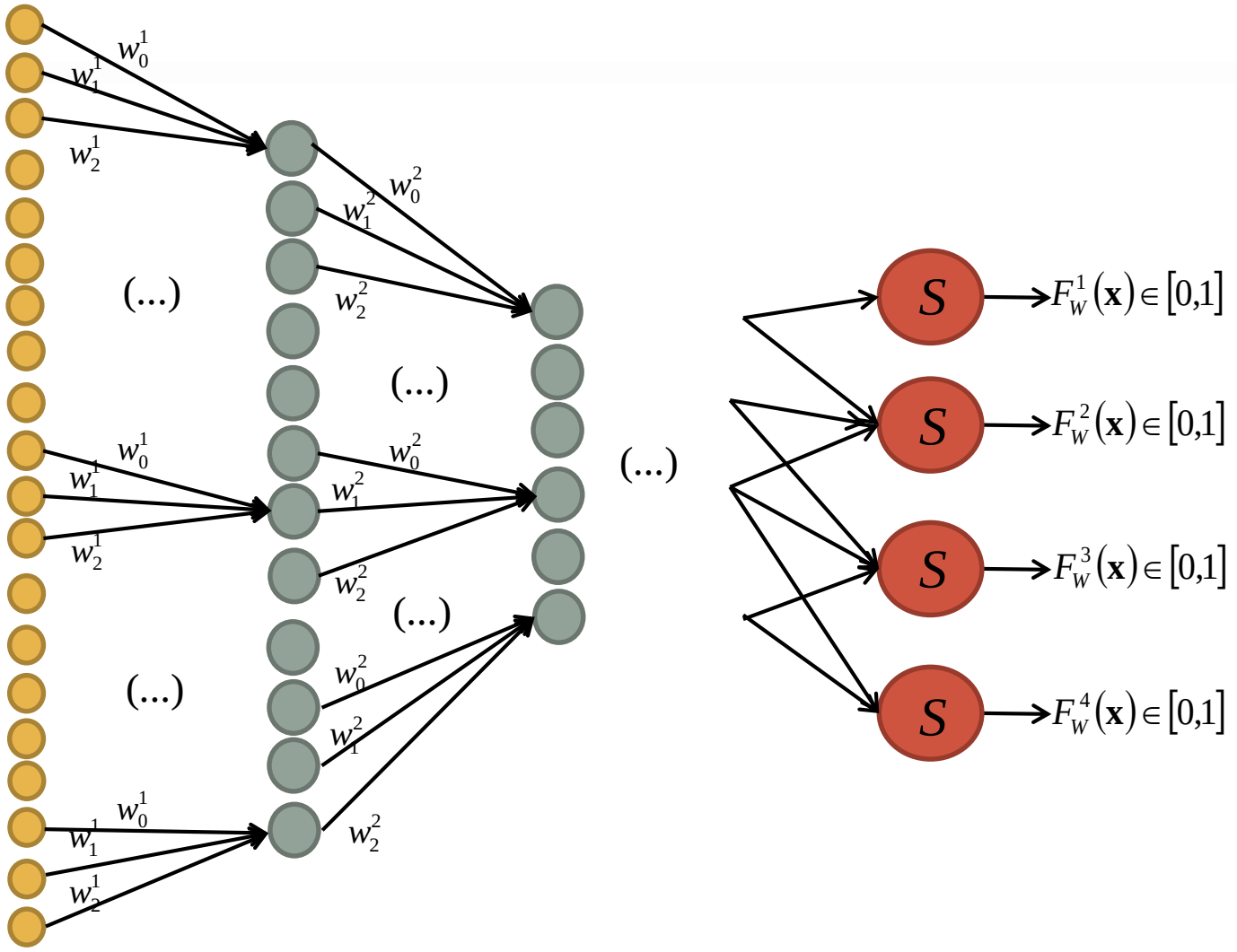
Too many parameters!



256x256 color image with 150 neurons in Layer 1 => **10 M parameters!!**

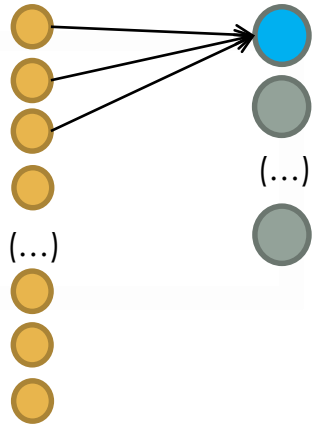
CONVOLUTIONAL NEURAL NETWORKS

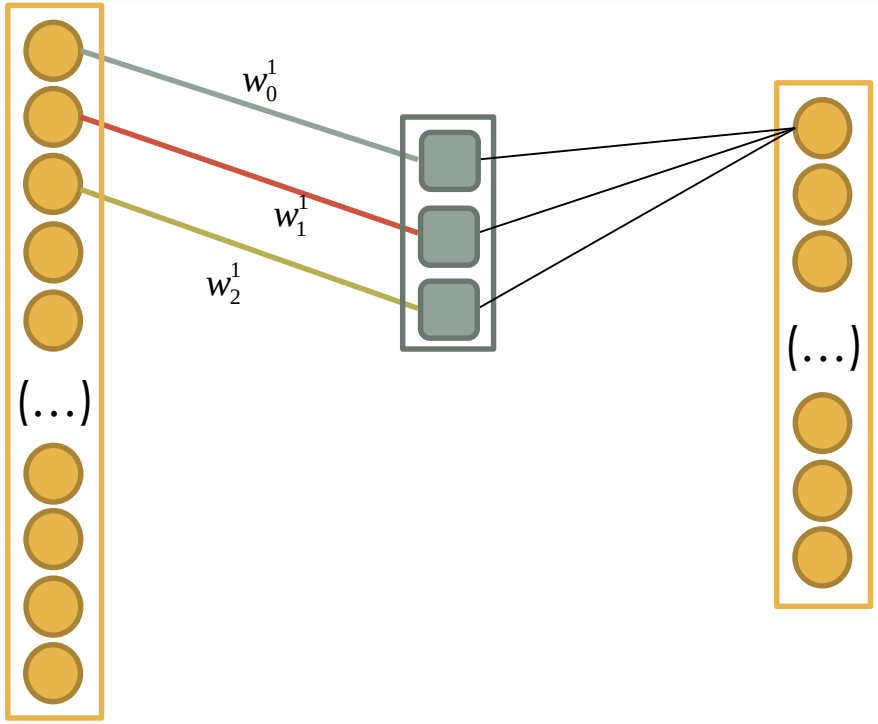
Share weights



256x256 color image with 150 neurons in Layer 1 => **3 parameters!!**

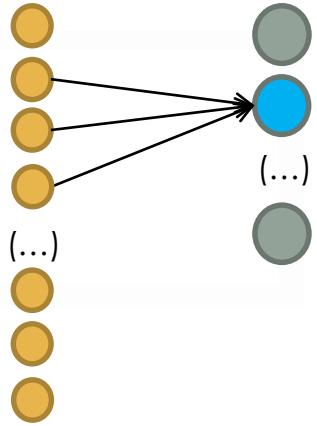
Convolution 1D

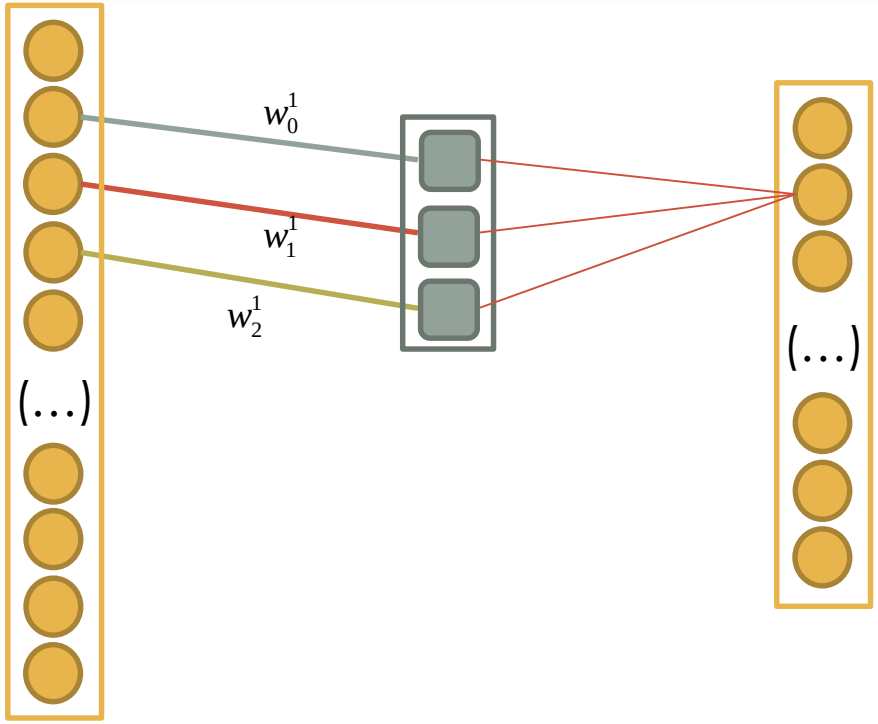




1D input (N) Filter (3) Output

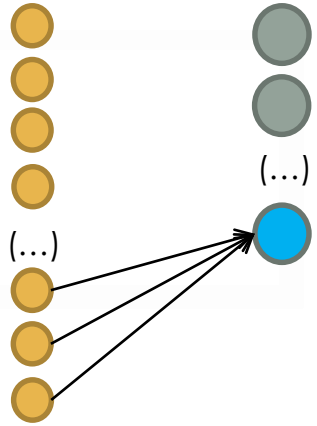
Convolution 1D

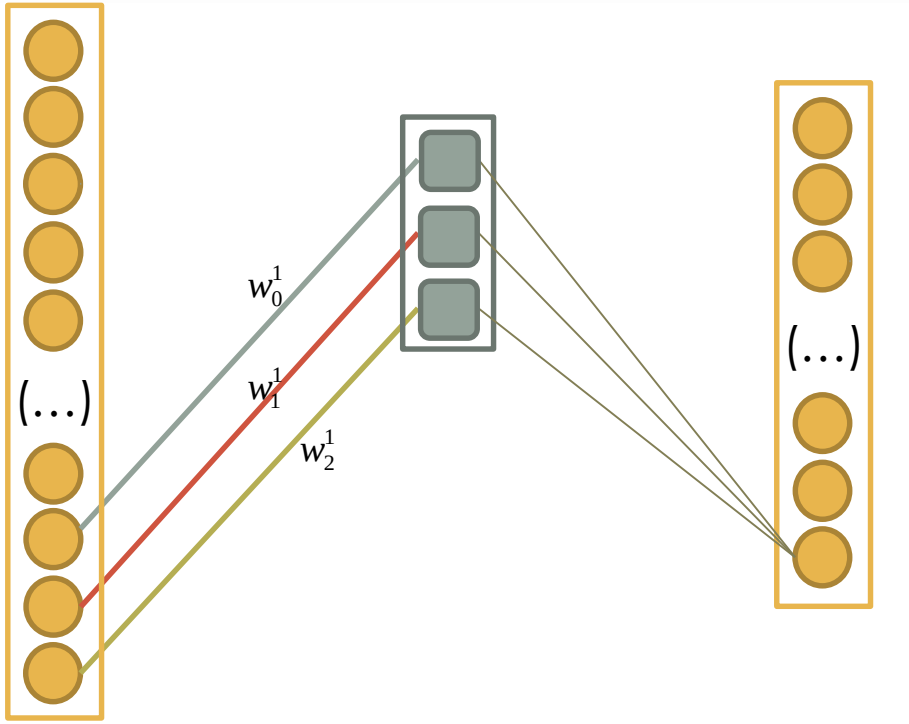




1D input (N) Filter (3) Output

Convolution 1D

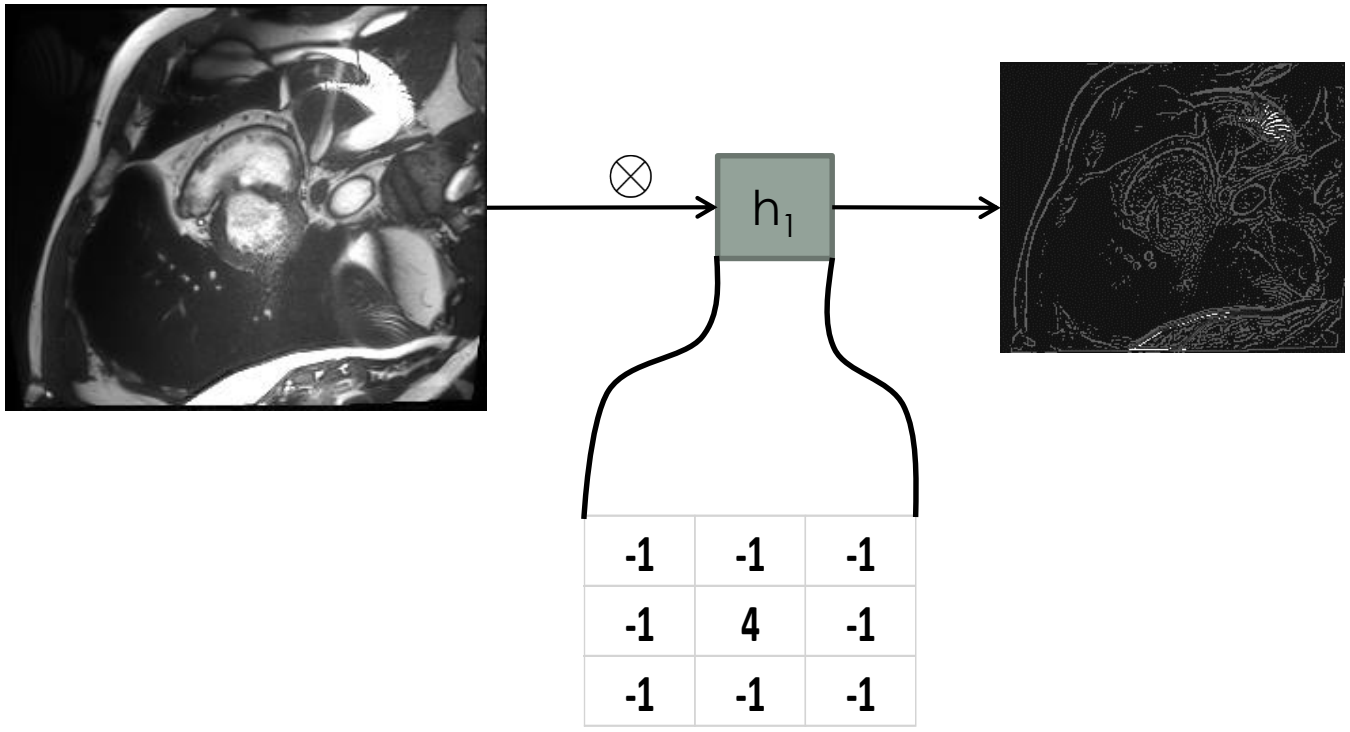




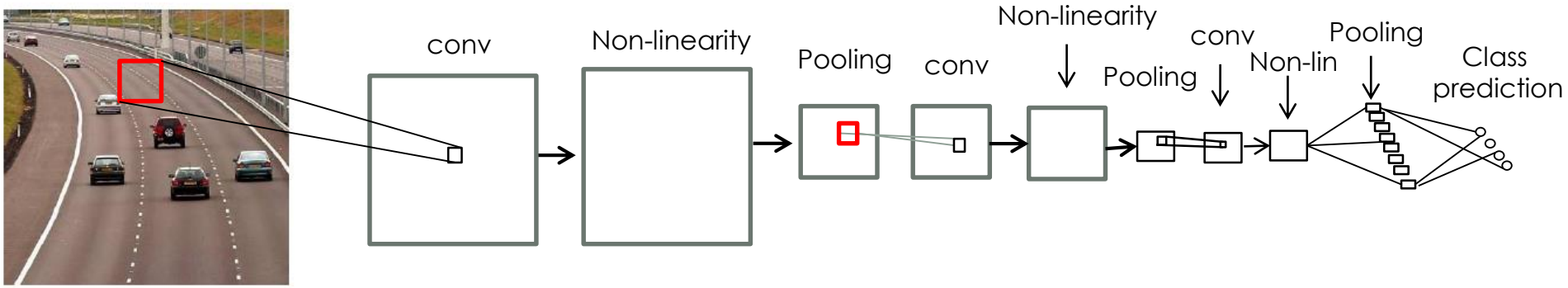
1D input (N) Filter (3) Output

Convolutional 2D

Input: 1, filter size: (3x3), output: 1

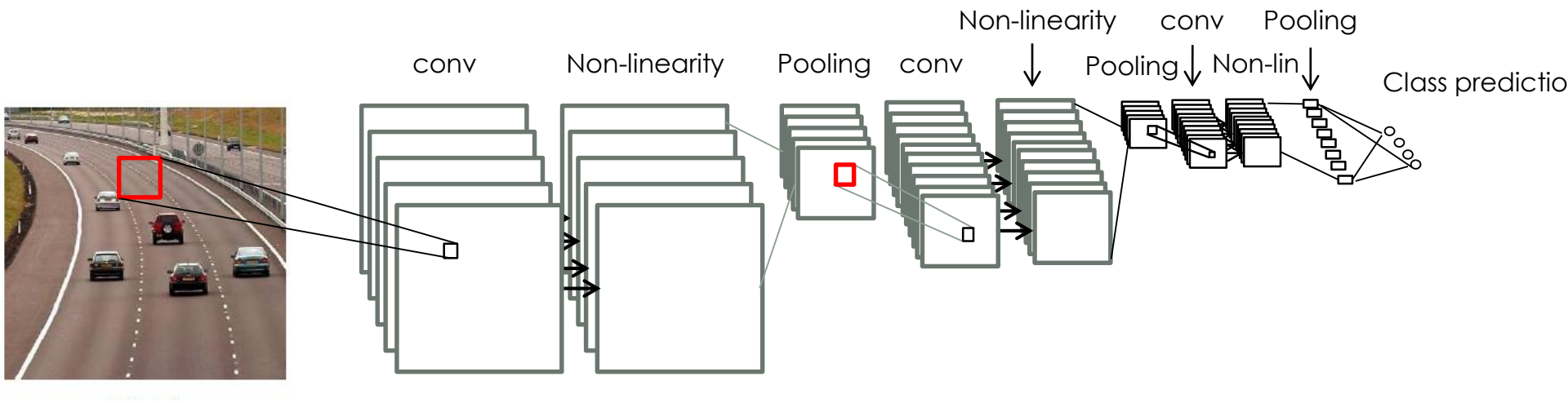


Conv Nets

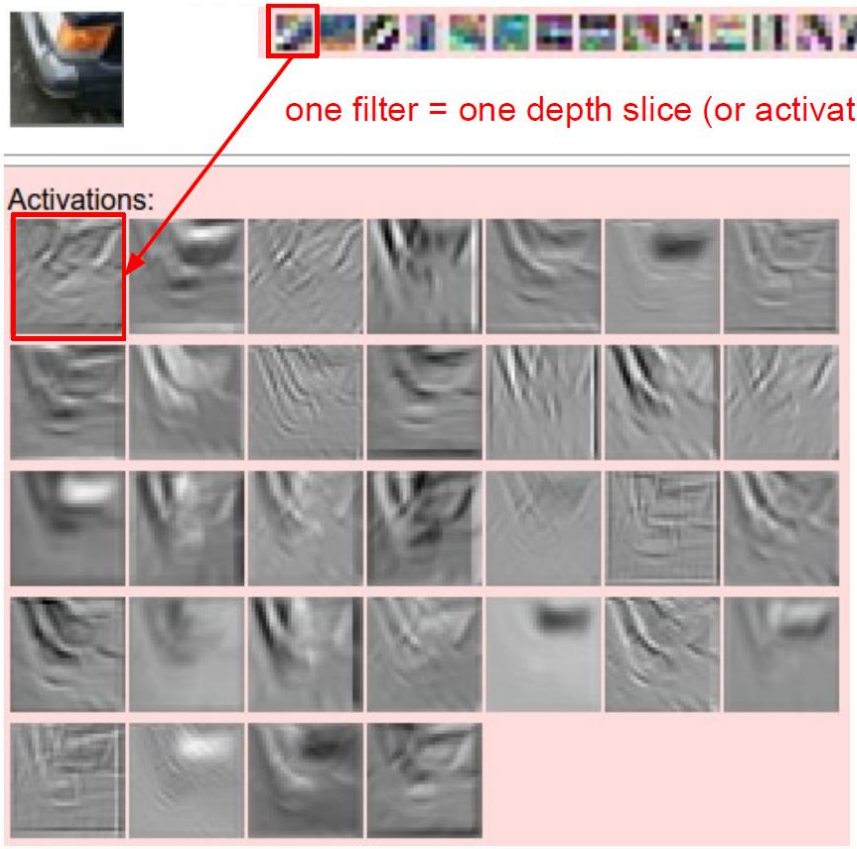


Too few parameters

Conv Nets


conv
Non-linearity
Pooling
conv
Non-linearity
conv
Pooling
Pooling
Non-lin
Class predictio

Layer 1 - 32 5x5 filters



5x5 filters

Can call the neurons “filters”

We call the layer convolutional because it is related to convolution of two signals (kind of):

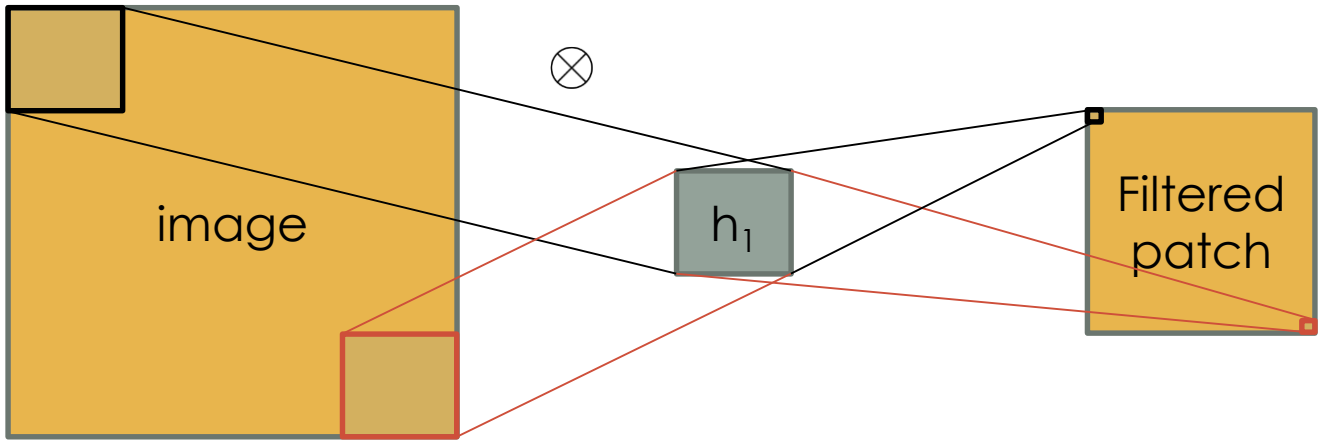
$$f[x,y] * g[x,y] = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} f[n_1,n_2] \cdot g[x-n_1, y-n_2]$$

elementwise multiplication and sum
a filter and the signal (image)
= np.dot(w,x) + b

[Credit : Fei-Fei Li & Andrej Karpathy]

Why using conv net ?

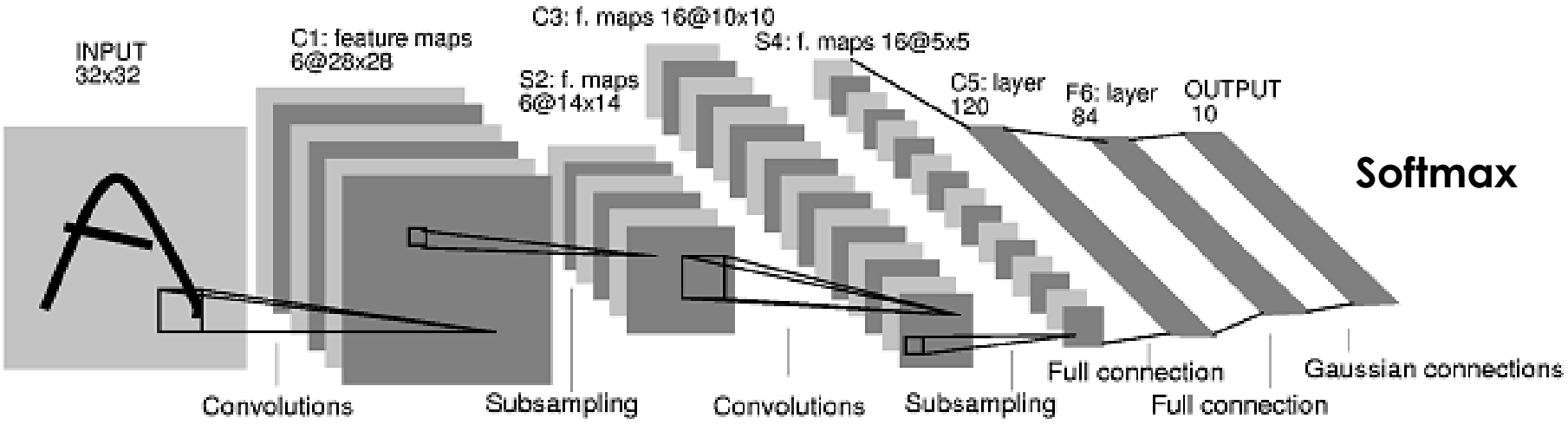
- Keep local information
  - Accross 2D, 3D input
- Sharing weights (reduce number of parameters)



- Scale with bigger image
- Invariant to local translation

RESULTS

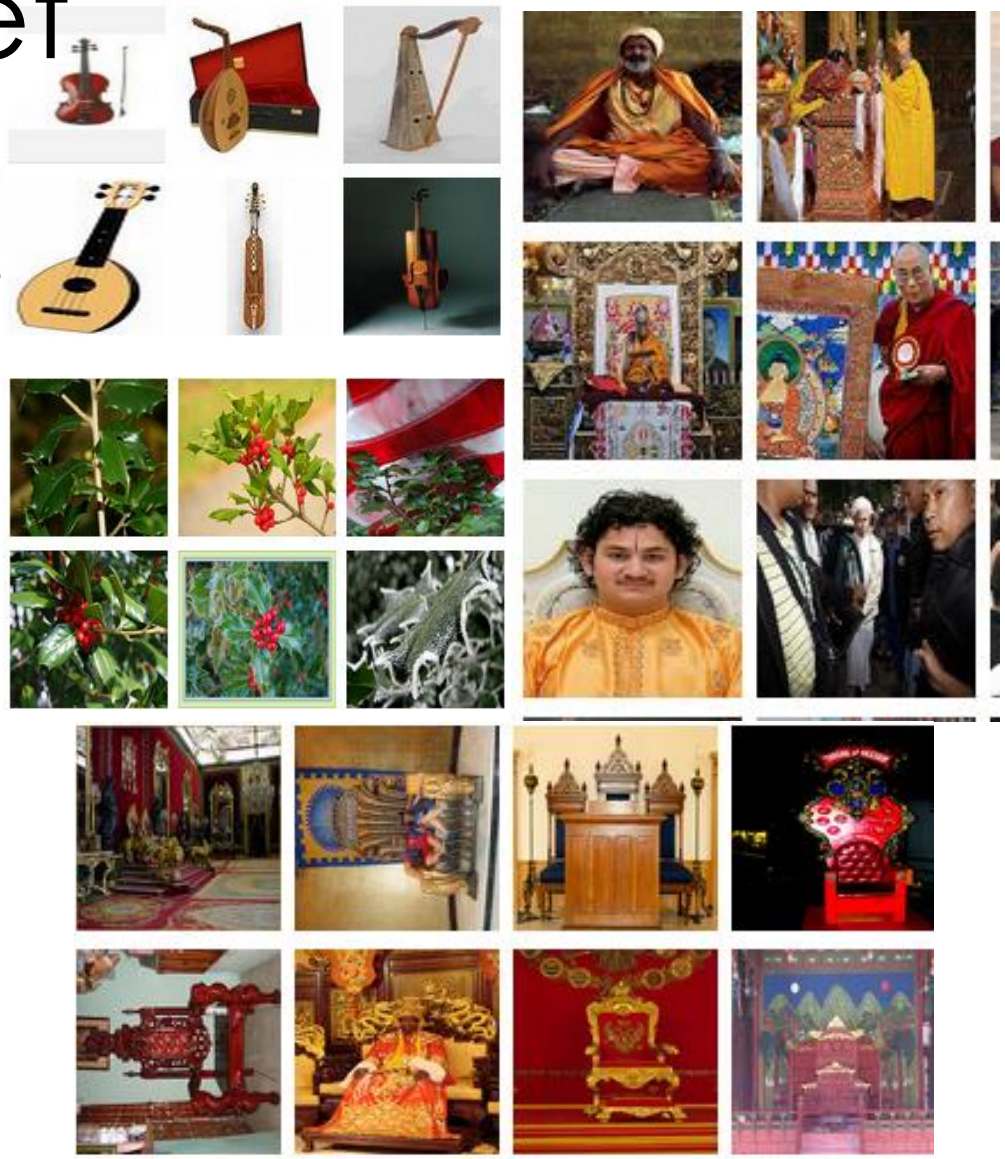
Conv Nets : "LeNet" (1989)



$$W^{[k+1]} = W^{[k]} - \eta \nabla E\left(W^{[k]}\right)$$

Image Net

- 1.5 Million image
- 1000 categories



http://karpathy.github.io/2014/09/02/what-i-learned-from-competing-against-a-convnet-on-imagenet/

The 2012 revolution

A. Krizhevsky I. Sutskever G. Hinton "**ImageNet Classification with Deep Convolutional Neural Networks**", NIPS 2012., 2482 citations!

Image Net Challenge

- Error rate: 15% (previous best: 25% error)

Method:

- Convolutional neural network
- Deep : 5 layers with 60M parameters
- Used the right tricks
  - Contrast normalization
  - GPU optimization
  - ReLU (REctified Linear Unit)
  - Drop out
  - Etc.

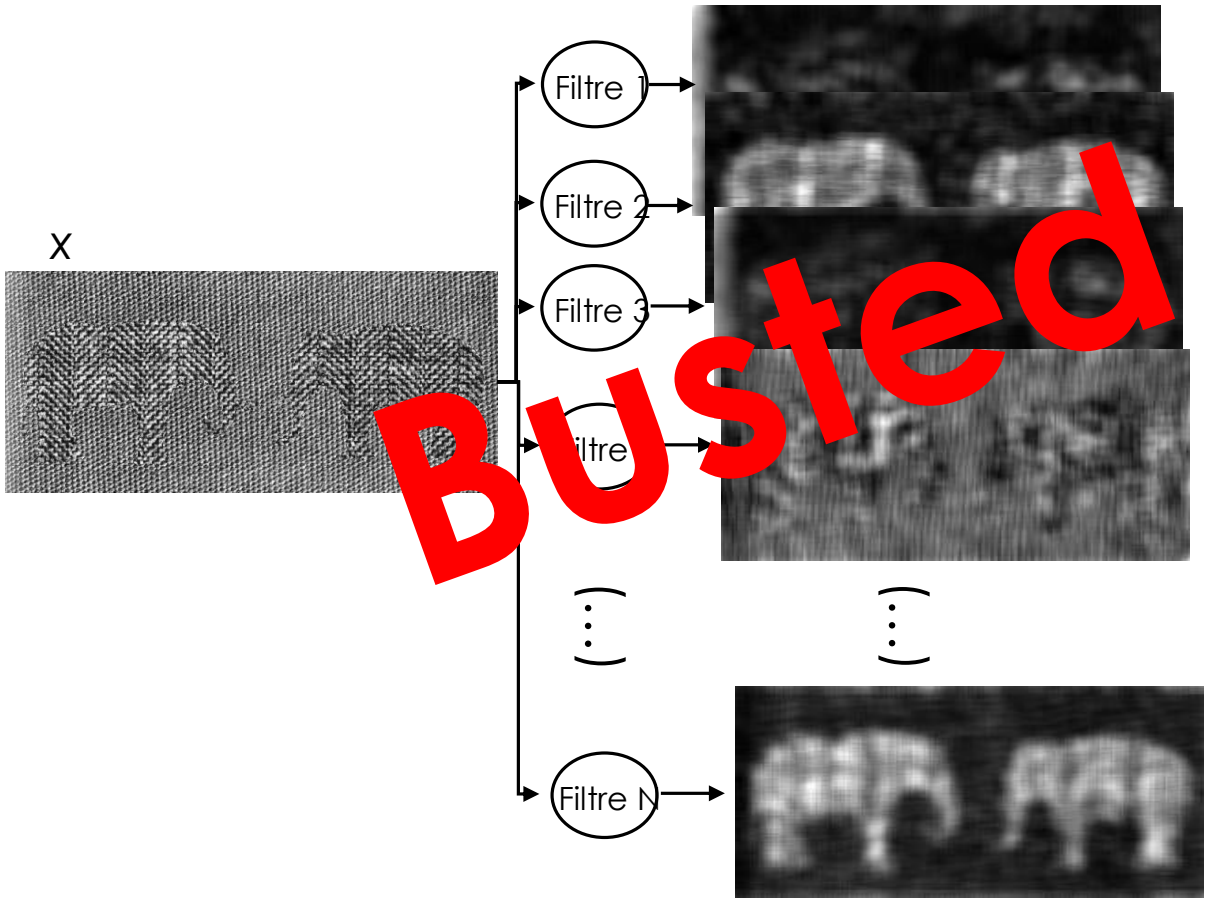
2015 results : MSRA

Classification error rate of only 3.5%

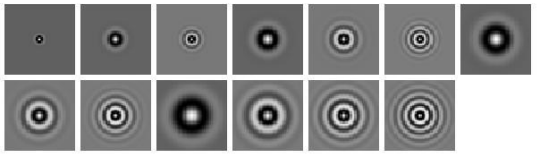
Human error 5.1%

Remember that?

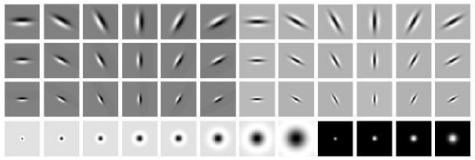
Features : which hand-designed features?
Method : hand-designed approach (no learning)



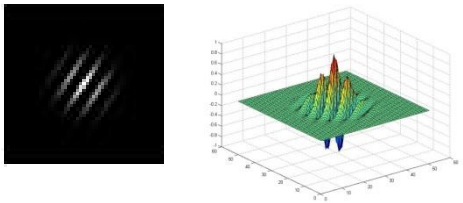
« texton filters »



« *steerable filters* »



Gabor filters



Not just fashion

Since 1960

hand-crafted Feature Extractor → "Simple" Trainable Classifier →

Low-level

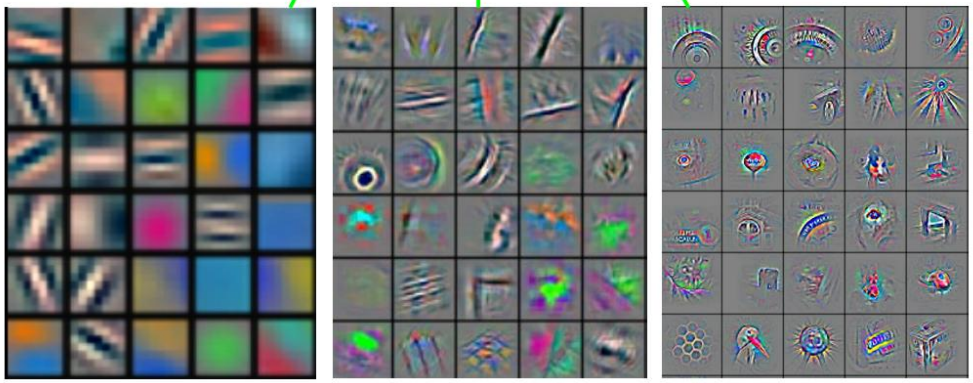
2003-2012

SIFT HoG → K-means Sparse Coding → Pooling → Classifier →

Low-level mid-level

2012-now

Low-Level Feature → Mid-Level Feature → High-Level Feature → Trainable Classifier →



Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

Credit : Yann Lecun and Marc'Aurelio Ranzato
http://www.cs.nyu.edu/~yann/talks/lecun-ranzato-icml2013.pdf

Conv Nets for segmentation

- Extract patch from the image
- Train on pixel prediction

or

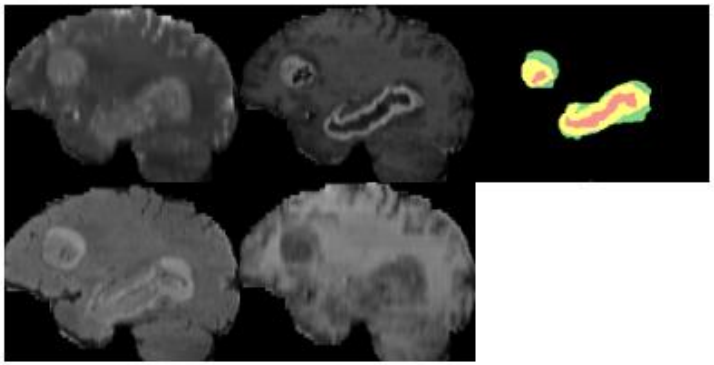
- Extract patch from the image
- Train on patch prediction

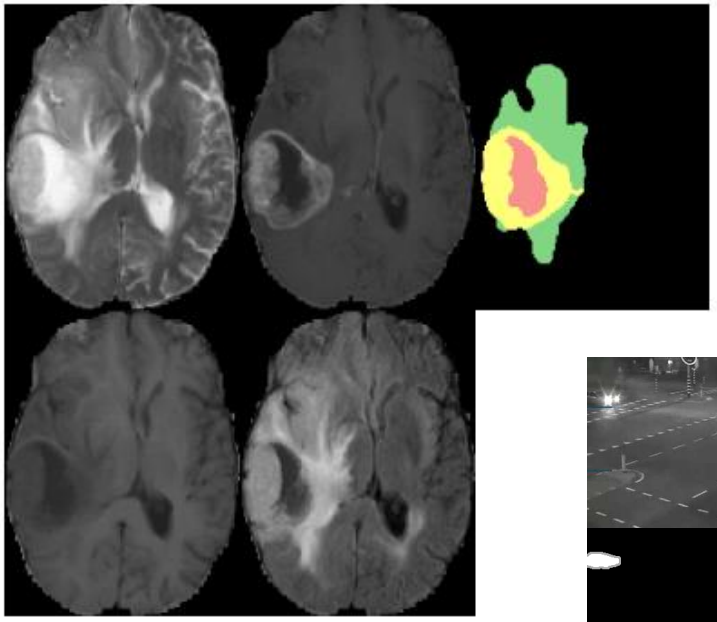
or

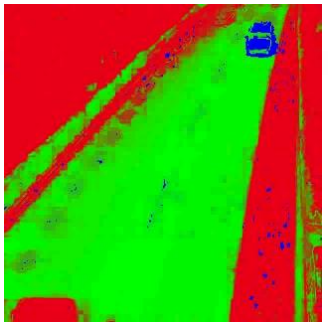
- Train on image prediction

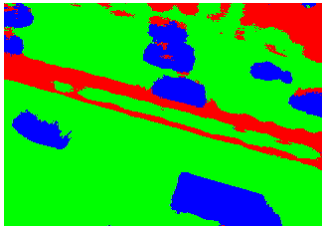
etc ...

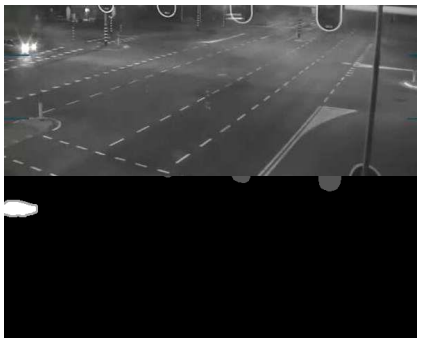
Conv Nets segmentation





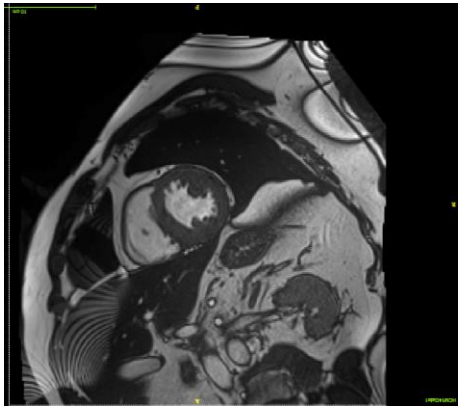




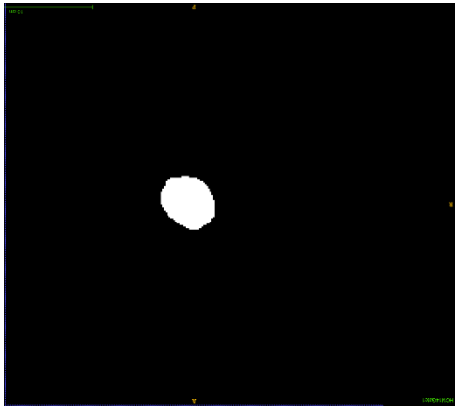


Medical Imaging

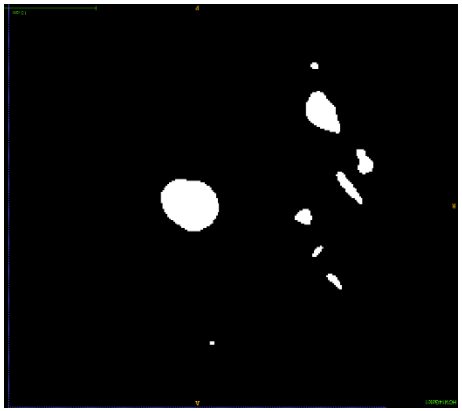
Hypertrophic cardiomyopathy



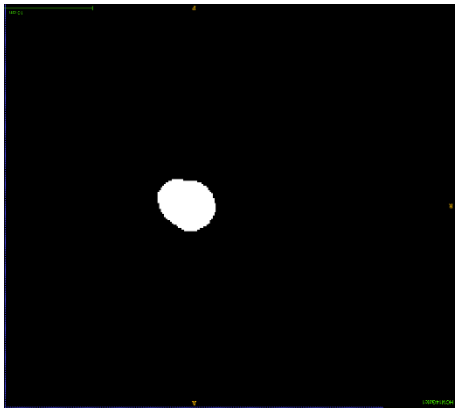
image



ground truth



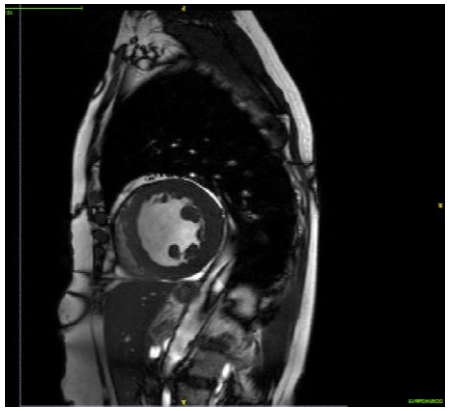
prediction



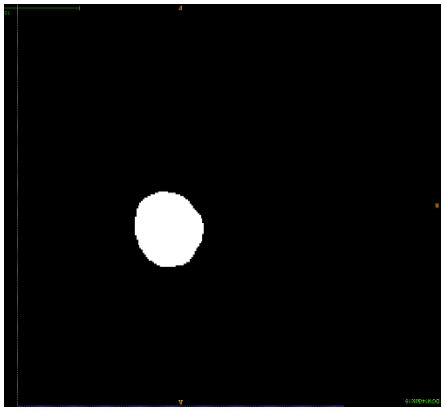
post-processing

Medical Imaging

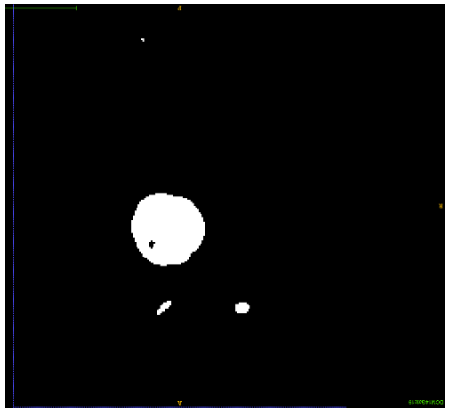
Dilated cardiomyopathy



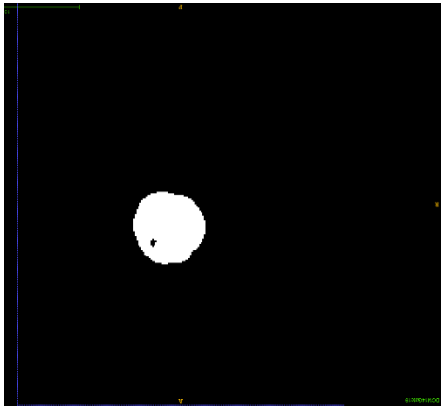
image



ground truth



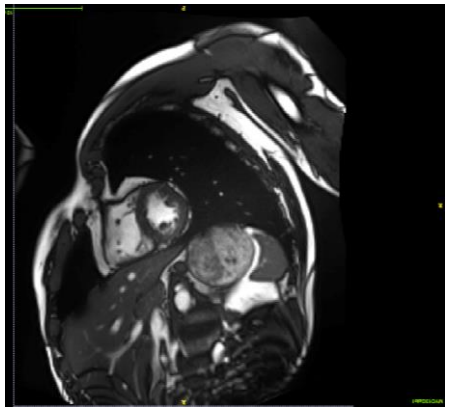
prediction



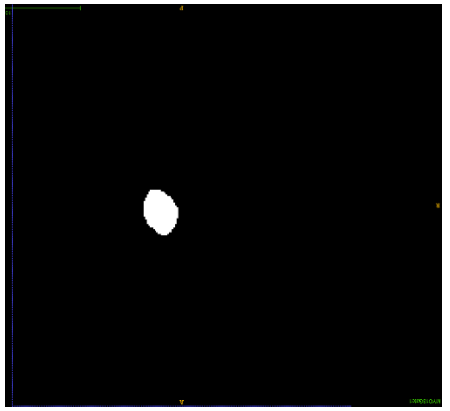
post-processing

Medical Imaging

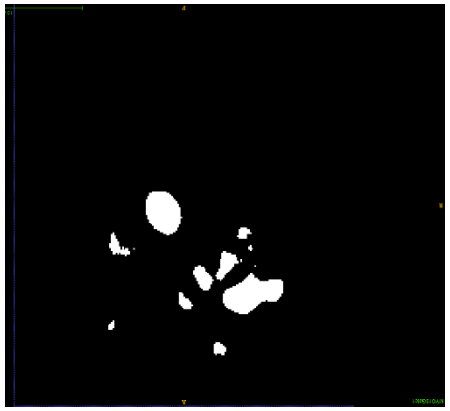
Myocarditis



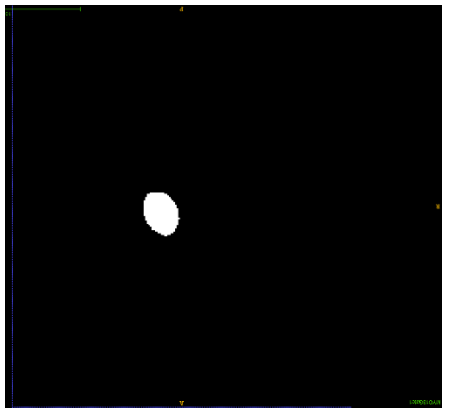
image



ground truth



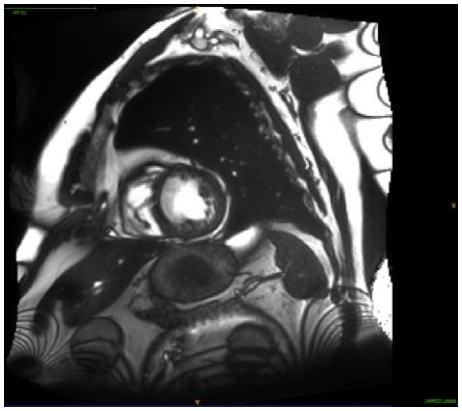
prediction



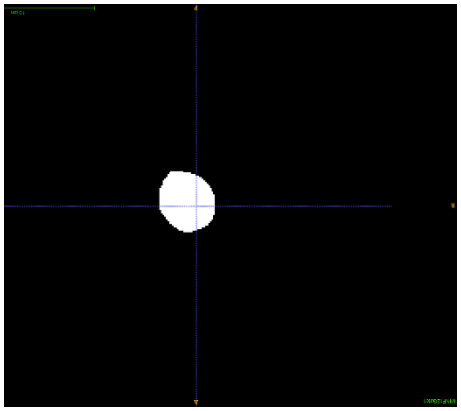
post-processing

Medical Imaging

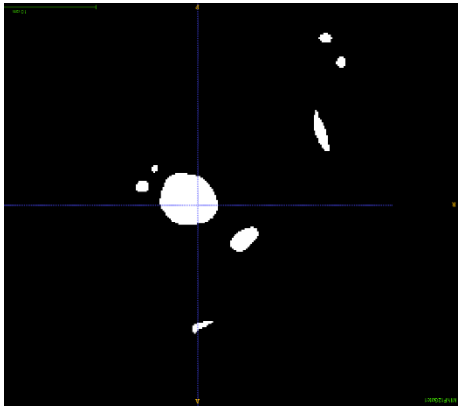
Myocardial infarction



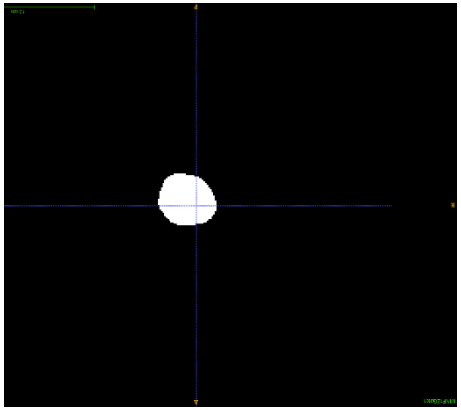
image



ground truth



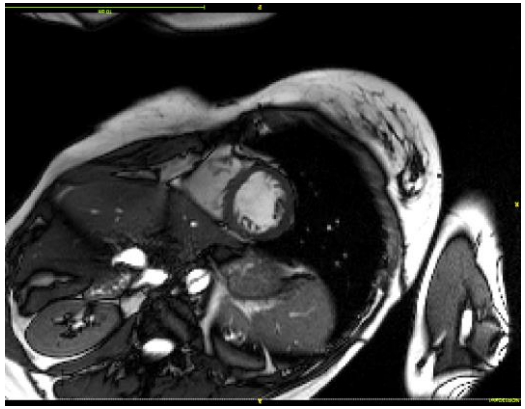
prediction



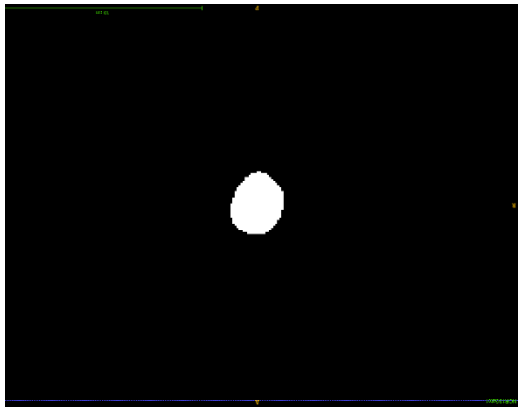
post-processing

Medical Imaging

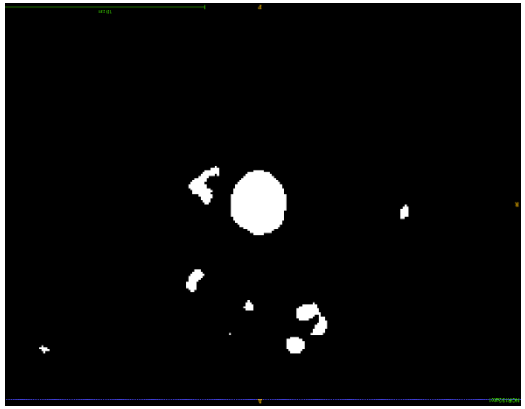
Healthy



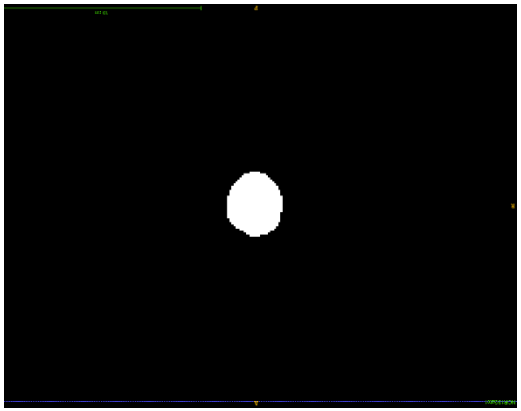
image



ground truth



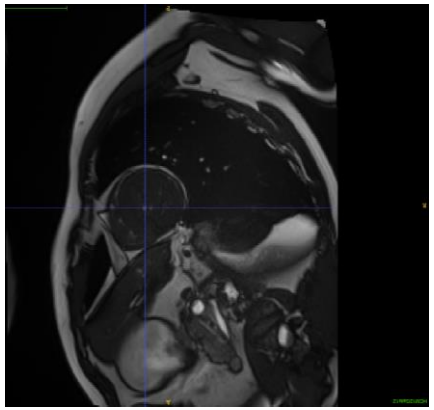
prediction



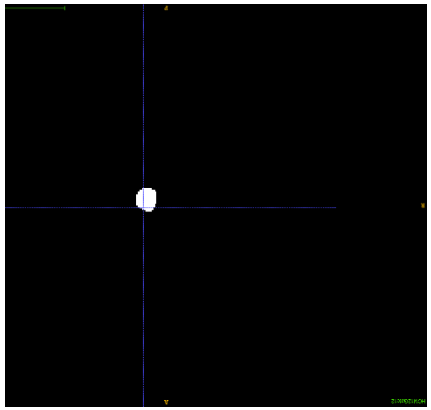
post-processing

Medical Imaging

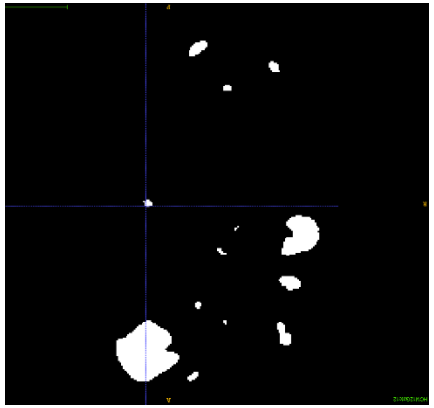
Degenerated result



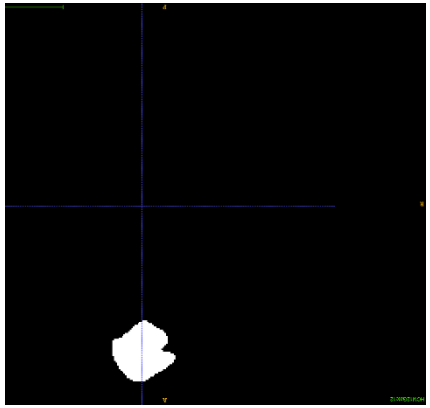
image



ground truth



prediction



post-processing

Medical Imaging

Quantitative results

Name	Hausdorff	Dice
DCM15Gate1.nii.gz	4.12	0.86
DCM15Gate13.nii.gz	5	0.78
HCM13Gate1.nii.gz	6.16	0.89
HCM13Gate10.nii.gz	9.11	0.73
MINF12Gate1.nii.gz	8.72	0.89
MINF12Gate11.nii.gz	9.95	0.83
MYO12Gate1.nii.gz	3	0.90
MYO12Gate10.nii.gz	2.24	0.90
NOR12Gate1.nii.gz	2	0.95
NOR12Gate9.nii.gz	2.83	0.87

Resources

- Stanford cs231n :
http://cs231n.stanford.edu/

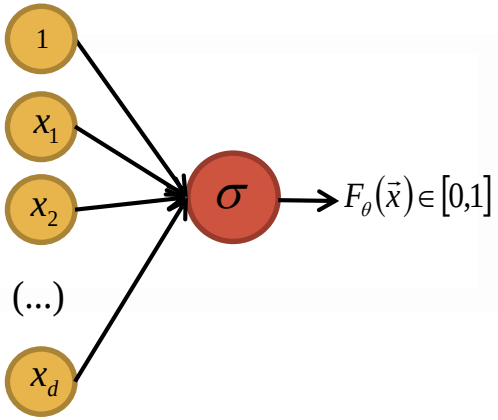
- Sherbrooke IFT725 :
http://www.dmi.usherb.ca/~larocheh/cours/ift725_A2014/contenu.html

Question ?

HIDDEN SLIDES

- Not used during the presentation

Error function



$$\left.\begin{aligned} P(y_i = 1 \mid \vec{x}_i) &= F_\theta(\vec{x}_i) \\ P(y_i = 0 \mid \vec{x}_i) &= 1 - F_\theta(\vec{x}_i) \end{aligned}\right\} \quad P(y_i \mid \vec{x}_i) = F_\theta(\vec{x}_i)^{y_i}\left(1 - F_\theta(\vec{x}_i)\right)^{1-y_i}$$

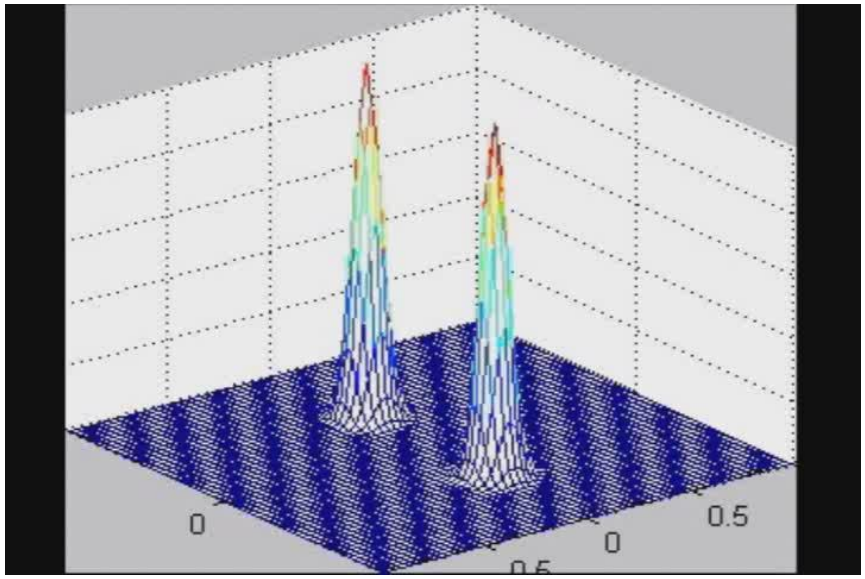
$$P(Y \mid X) = \prod_{i=1}^{n} P(y_i \mid \vec{x}_i)$$

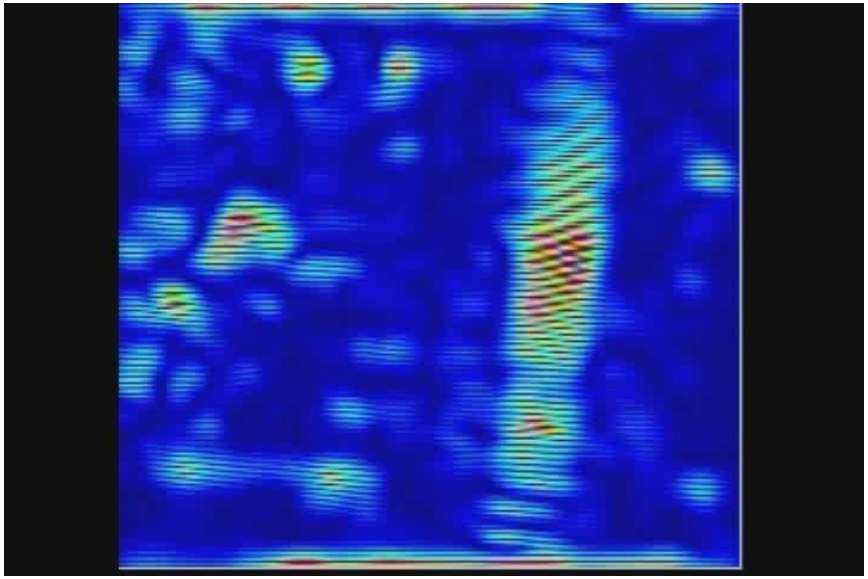
$$= \prod_{i=1}^{n} F_\theta(\vec{x}_i)^{y_i}\left(1 - F_\theta(\vec{x}_i)\right)^{1-y_i}$$

$$E(\theta) = -\ln P(Y \mid X)$$

$$= \sum_{i=1}^{n} y_i \ln F_\theta(\vec{x}_i) + (1 - y_i)\ln\left(1 - F_\theta(\vec{x}_i)\right)$$

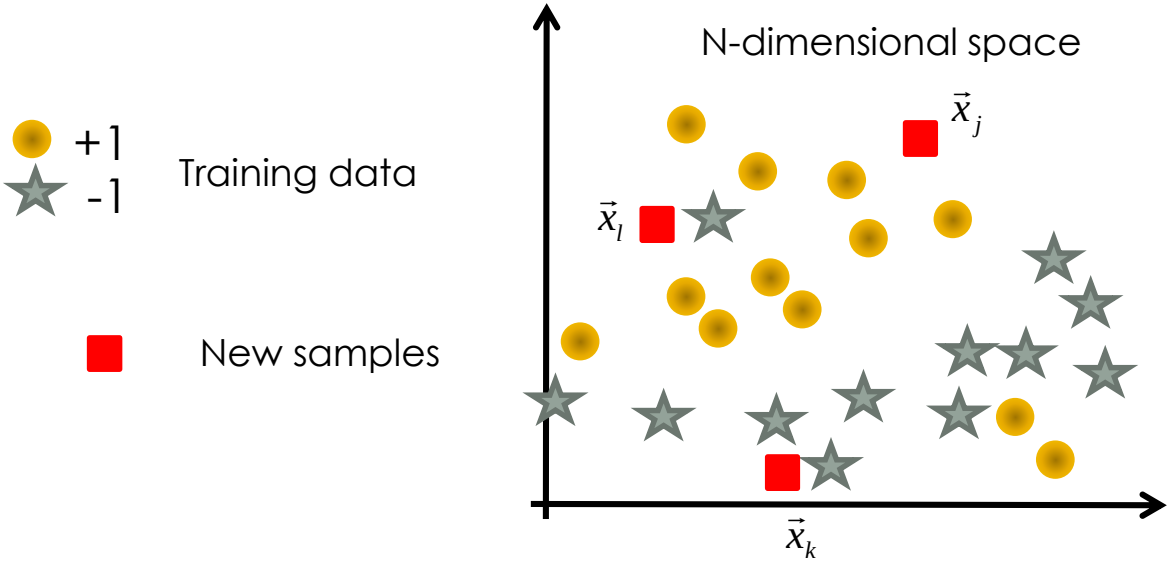
Texture features





(Vidéo)

K-NN



By looking at the nearest training samples, we can say that

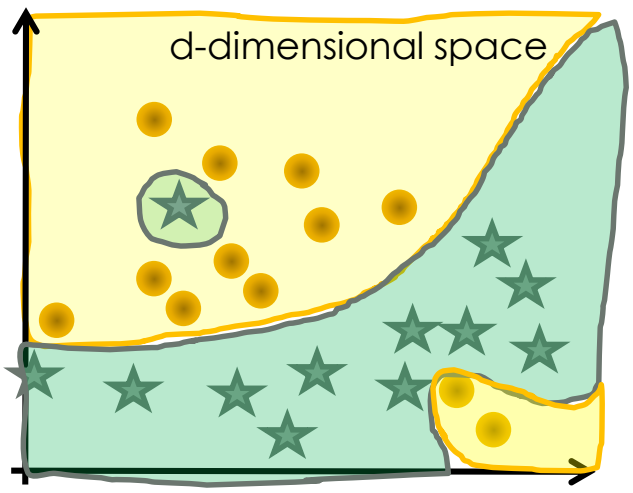
j is most likely in **class +1**,
k is most likely in **class -1**
l could be in both.

K-NN

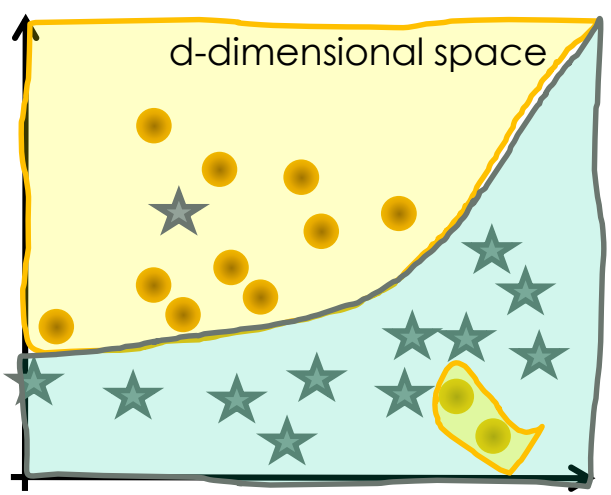
The classification function of the K-Nearest neighbor method is

$$F(\vec{x}_j, Z) = \begin{cases} +1 & \text{if at least K/2 NN among the K NN are in class } +1 \\ -1 & \textit{otherwise} \end{cases}$$

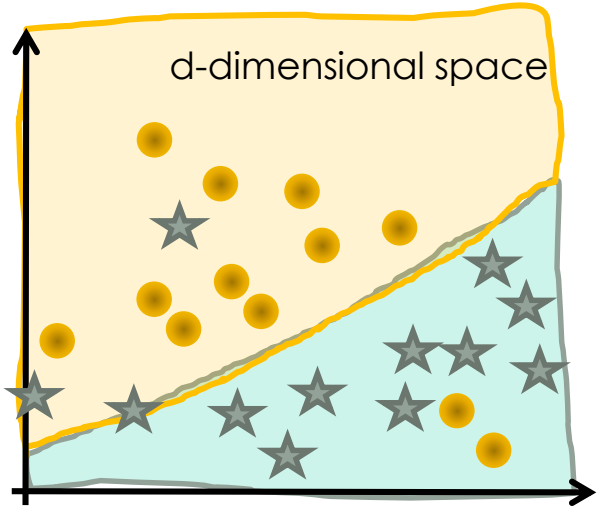
K=1



K=3

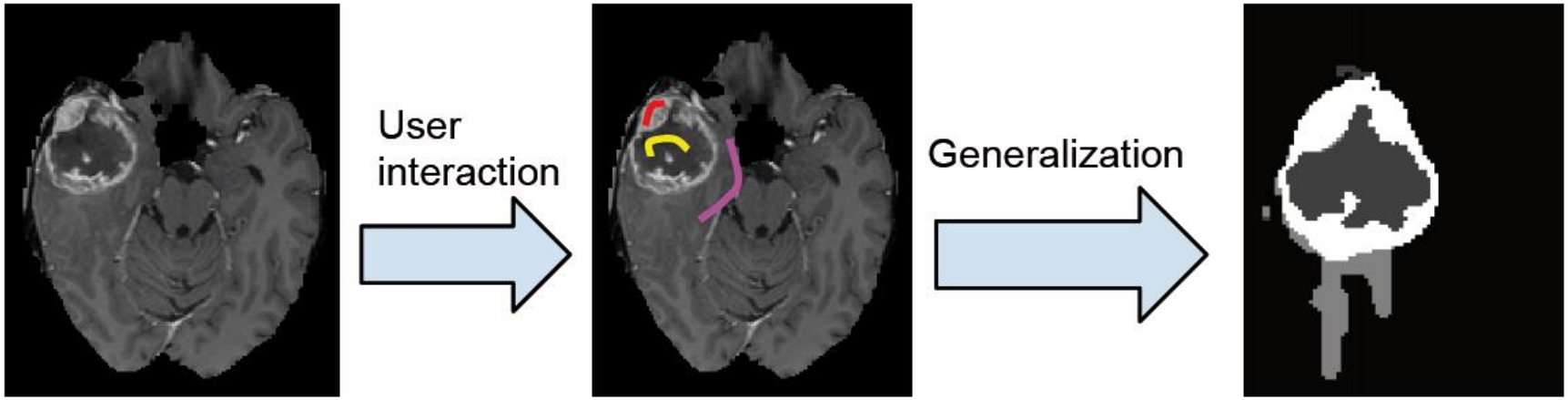


K=5



Within-brain learning

Interactive approach



Havaei M., Larochelle H., Poulin P., Jodoin P-M. (2015)
Within-Brain Classification for Brain Tumor Segmentation
(in press) International Journal of Computer Assisted Radiology and Surgery.

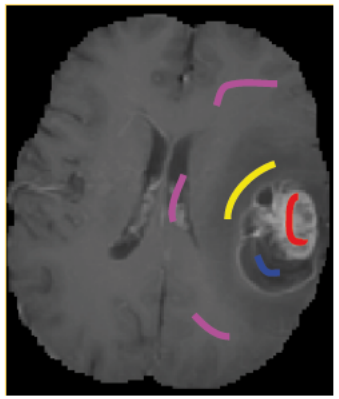
Within-brain learning

Interactive approach

Feature space:

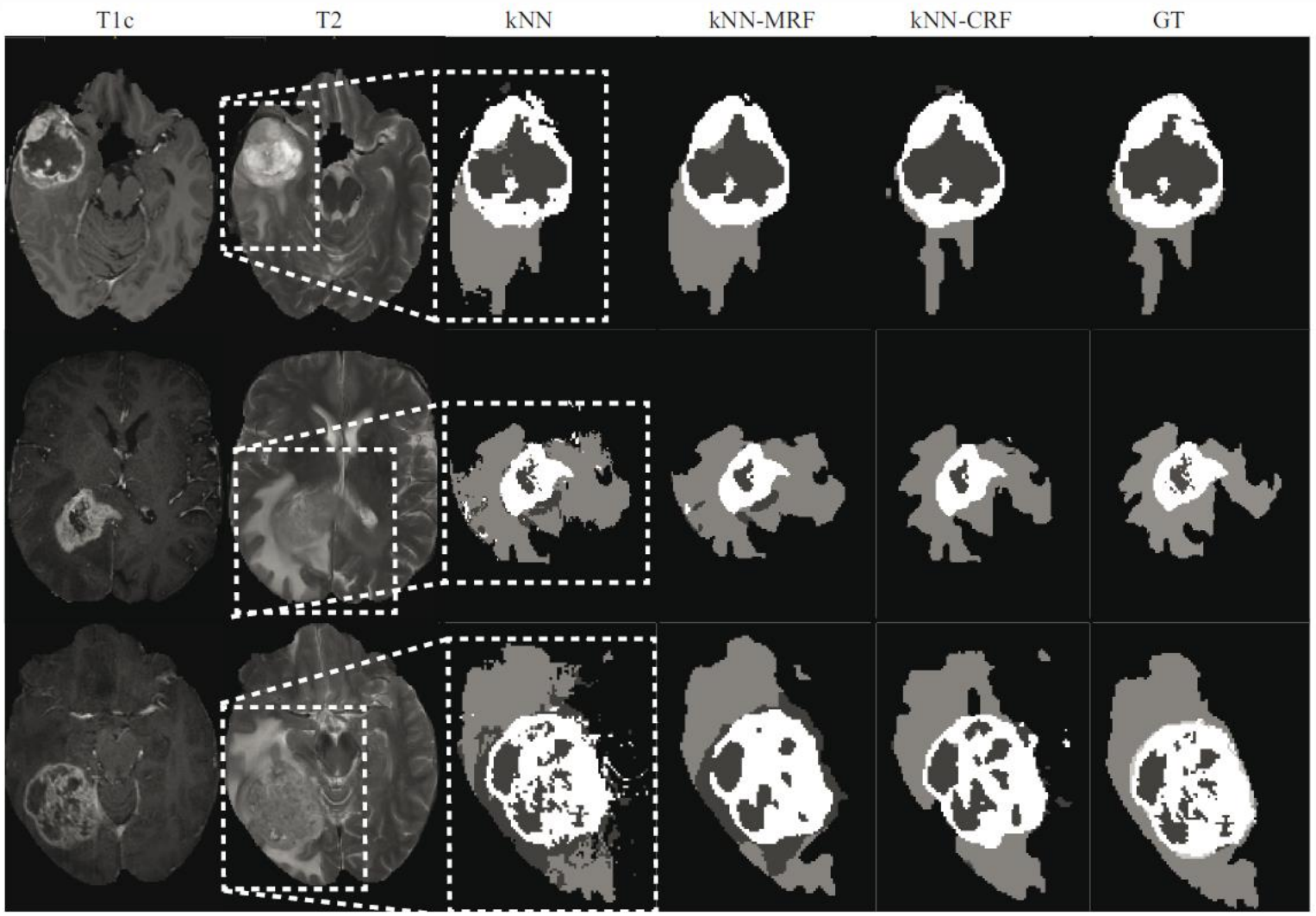
- For each voxel v , $\mathbf{x}_v = \{I_v^1, I_v^2, I_v^3, i, j, k\}$
- I^1, I^2, I^3 are intensity values of the T1C, T2 and Flair. modalities
- i, j, k are the 3D position of voxel v.

Training data is collected by user interaction. $\{(\mathbf{x}_t, y_t)\}_{t=1}^n$



Within-brain learning

Interactive approach



K-NN

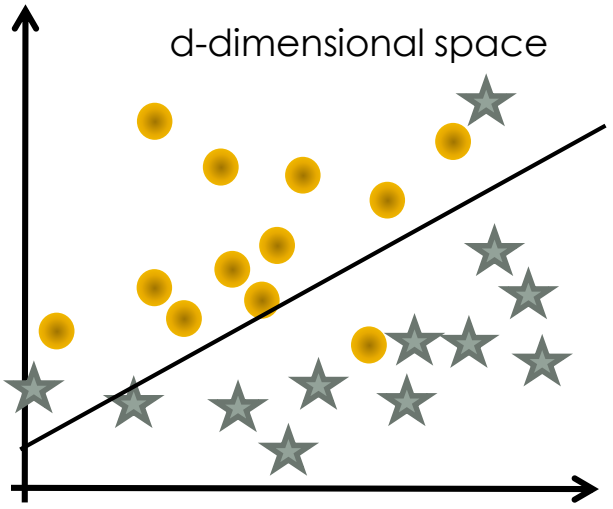
Advantages:

- it is the simplest supervised classification method
- it allows for a simple (when K is large) or complex (when K is small) decision boundary. All depends on the size of K
- No training required

Inconvenients :

- slow when dealing with many high dimensional data
- expensive memory wise $F(\vec{x}_j, Z)$

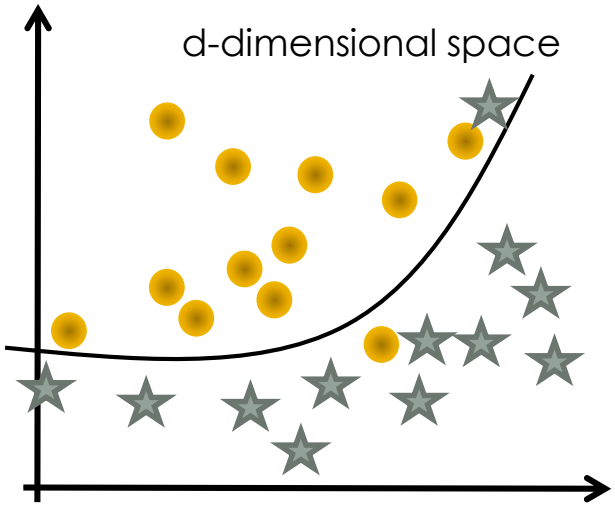
Training = estimate a decision boundary

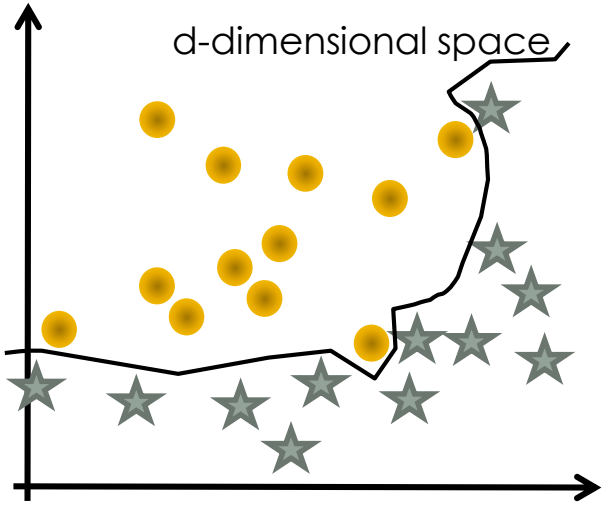


Under fitting
Simple model

Large bias

High training error
Low processing



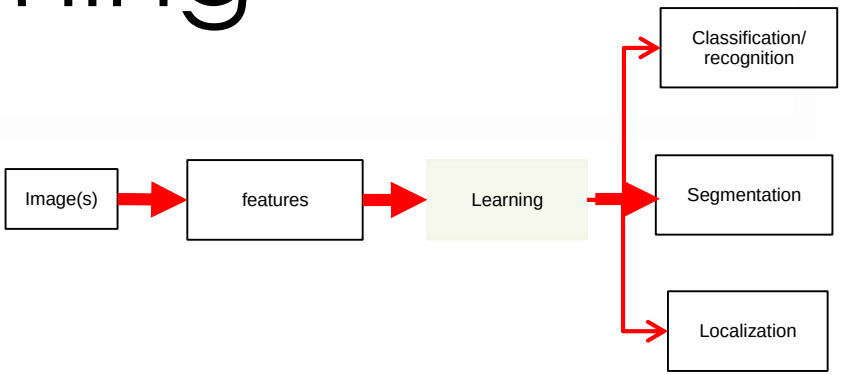


Over fitting
Complex model

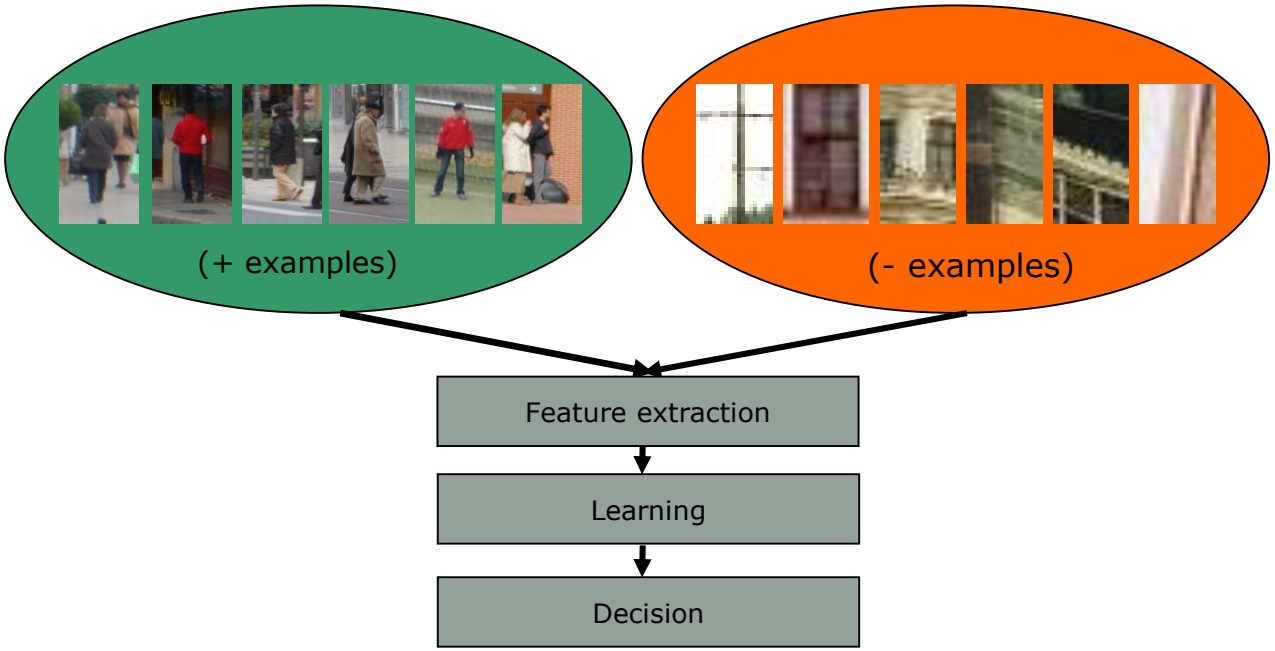
Large variance

Low training error
High generalization error
More processing

Supervised Learning

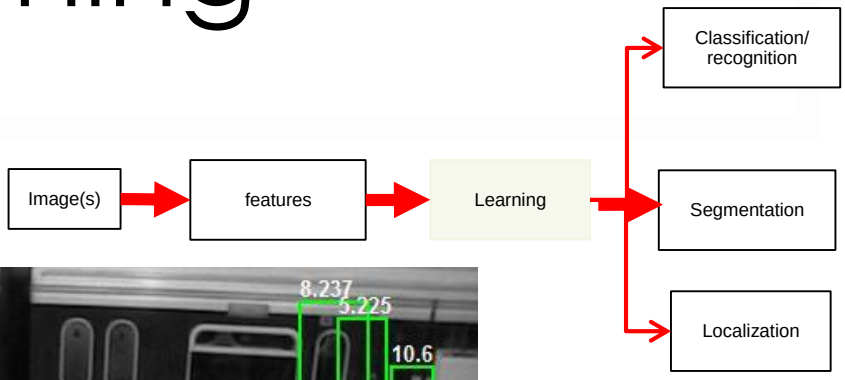


Localization

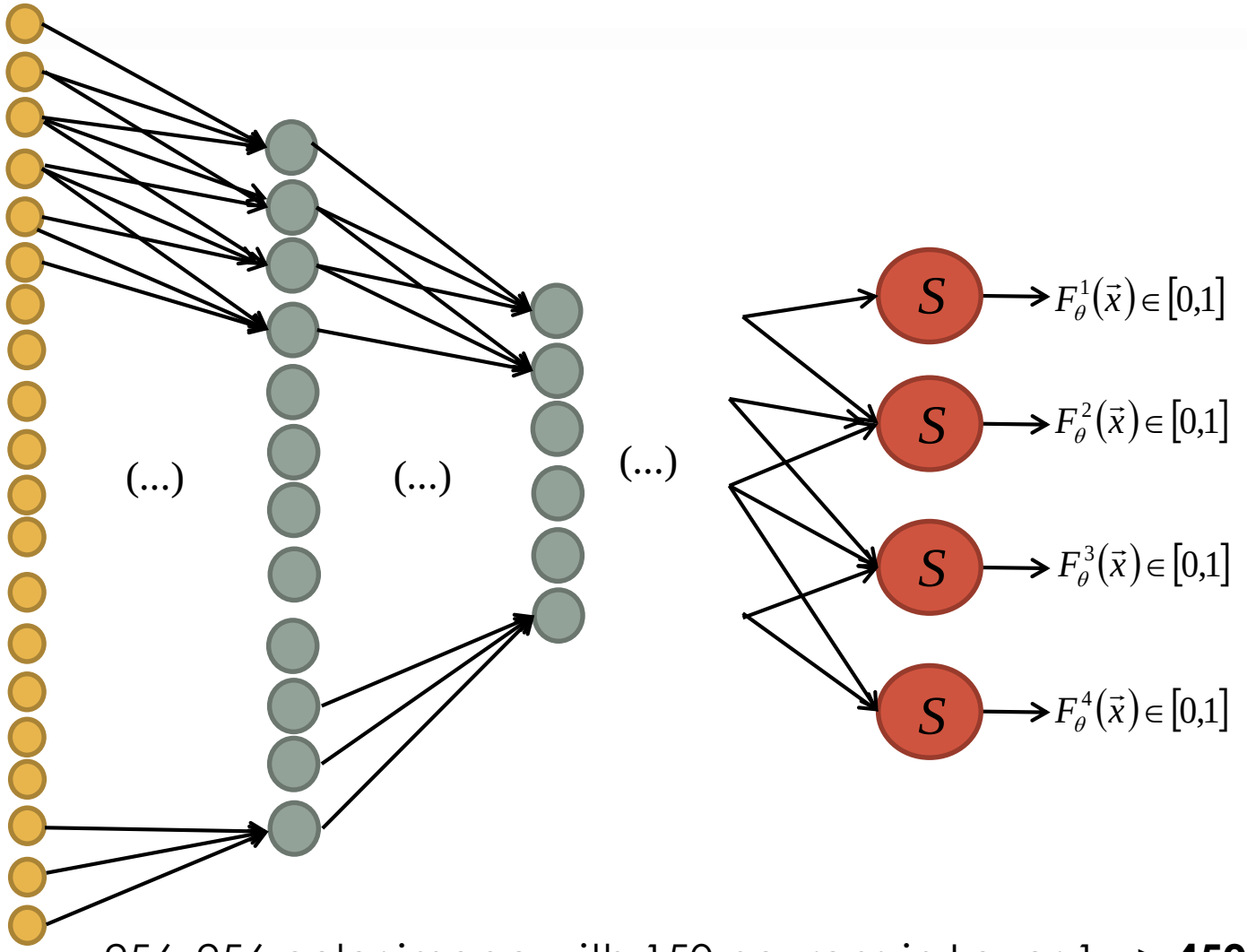


Supervised Learning

Localization

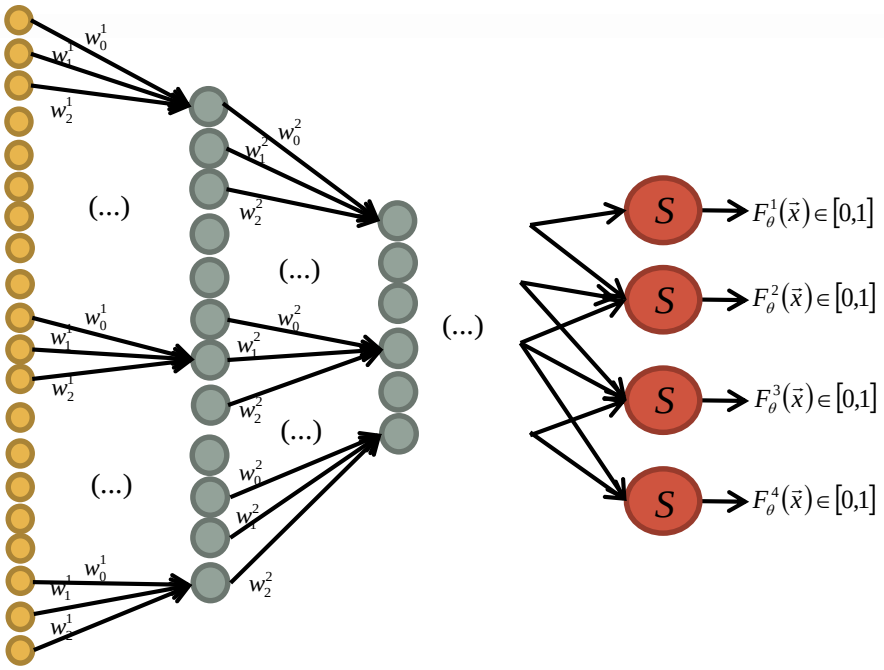


No full connection



256x256 color image with 150 neurons in Layer 1 => **450 parameters!!**

Share weights

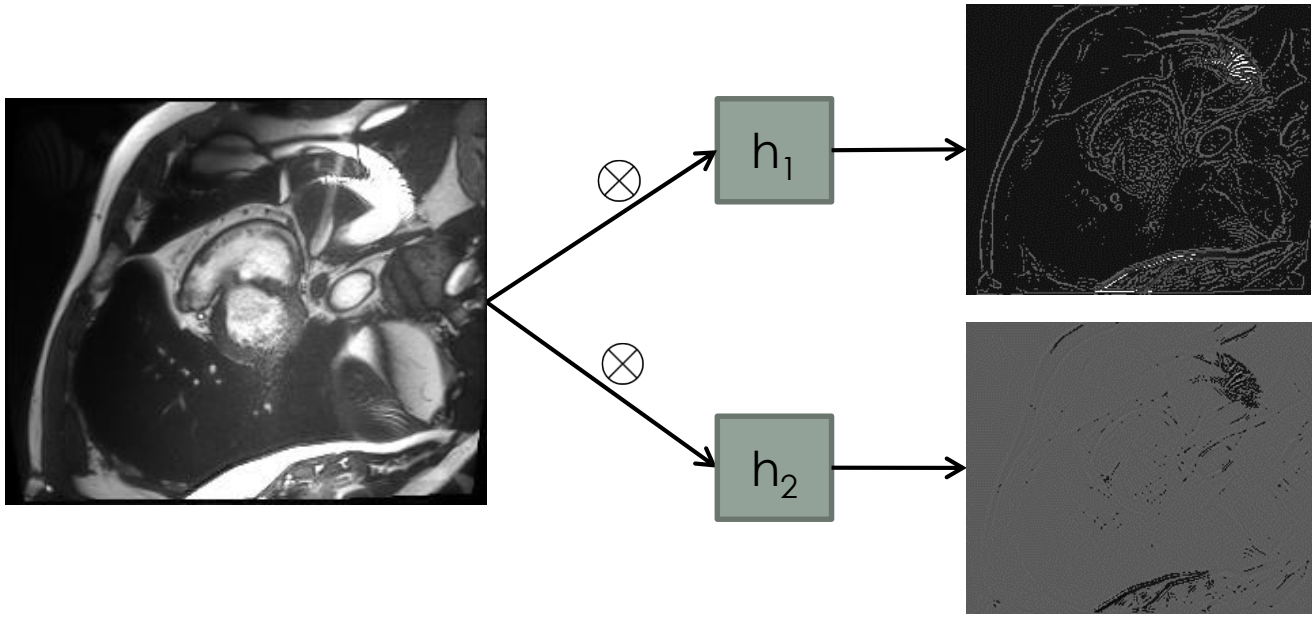


1- Learning convolution filters!
2- Small number of parameters = can make it deep!

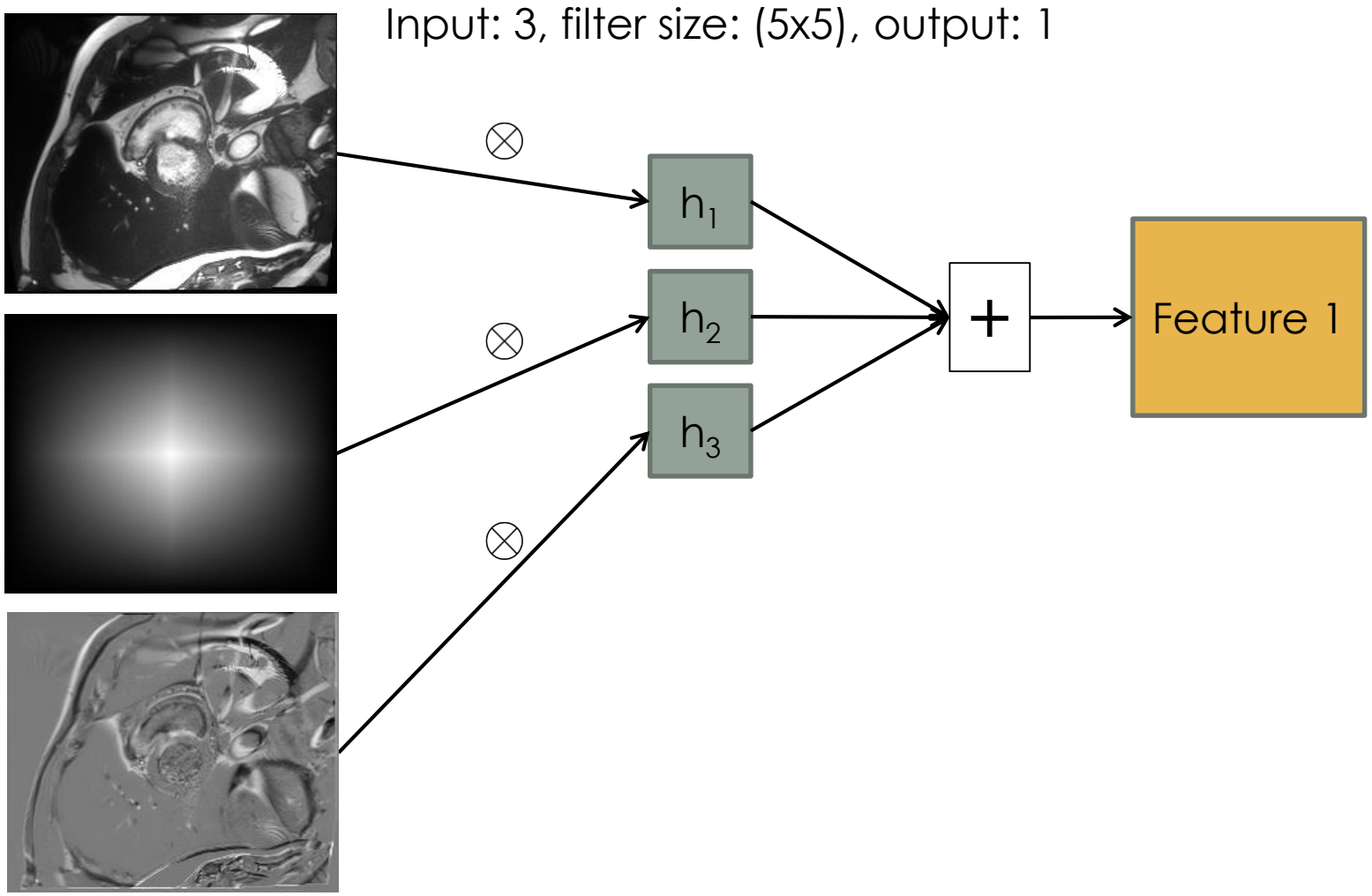
256x256 color image with 150 neurons in Layer 1 => **3 parameters!!**

Convolutional Network

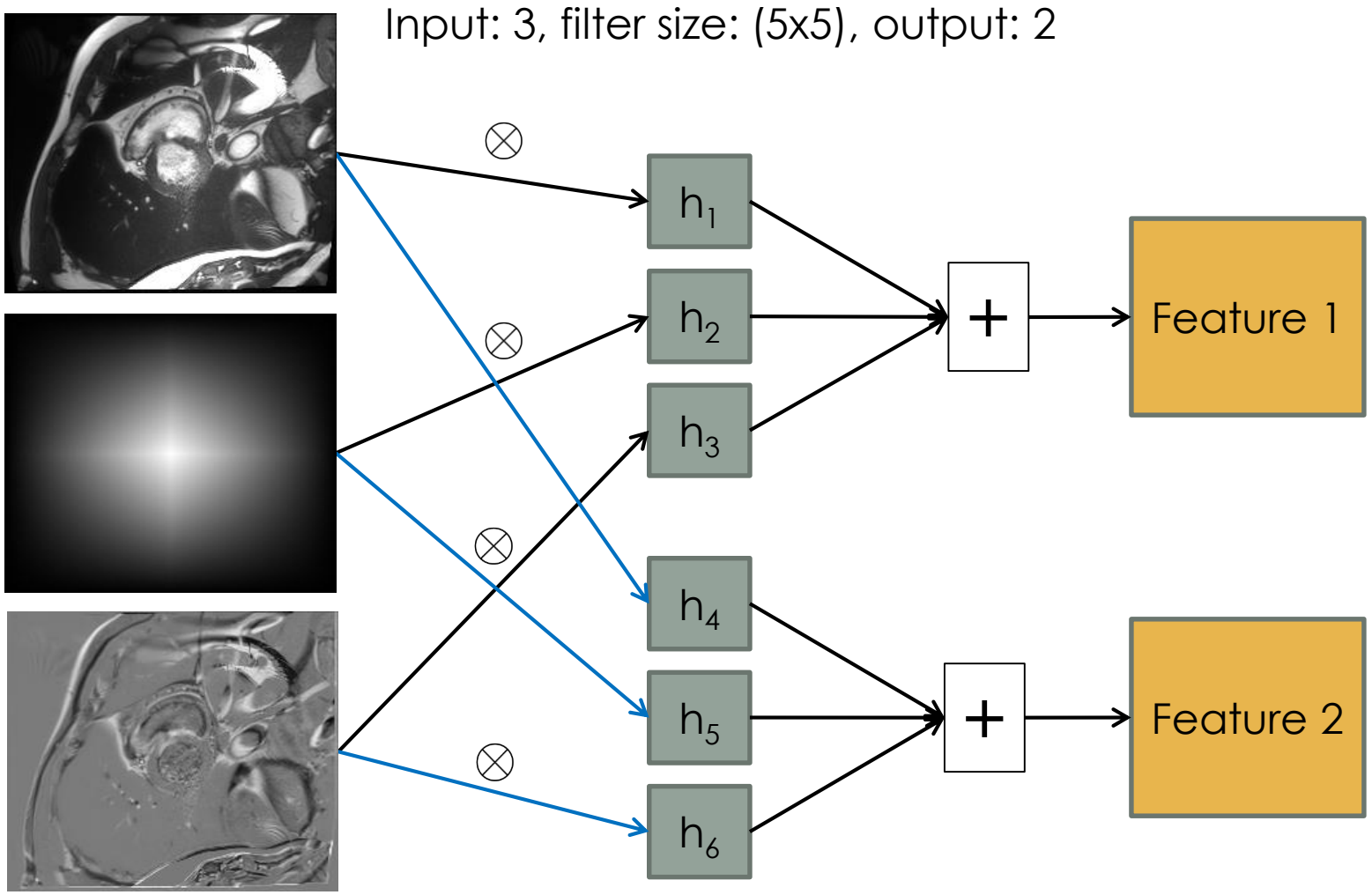
Input: 1, filter size: (5x5), output: 2



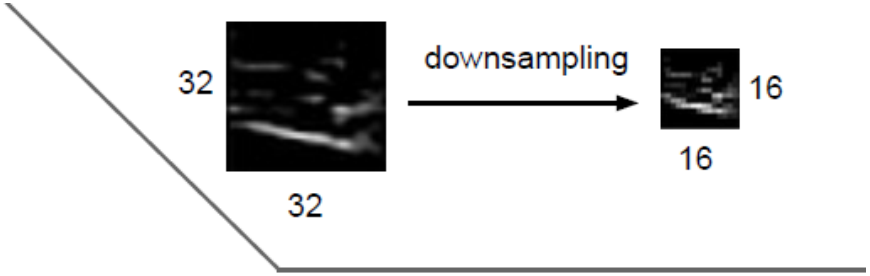
Convolutional Network



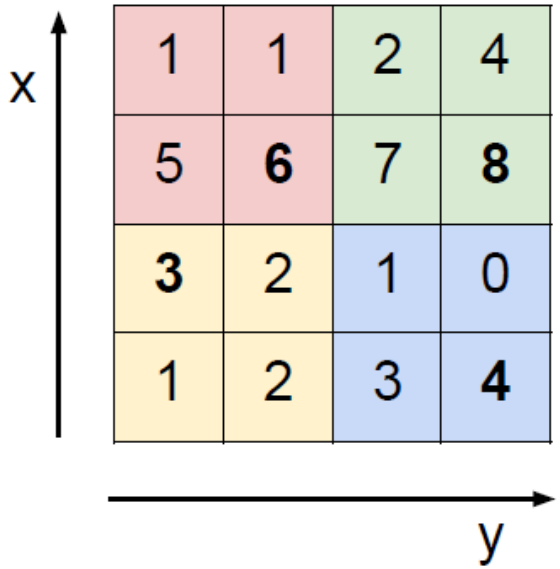
Convolutional Network



Max Pooling Layer



Single depth slice



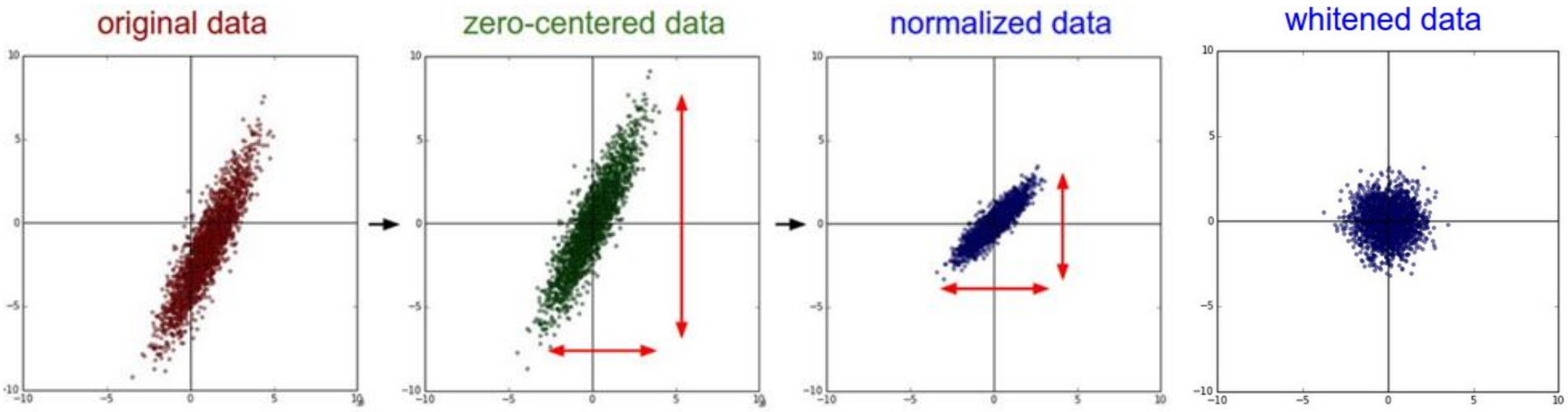
max pool with 2x2 filters and stride 2

6	8
3	4

Pooling layer downsamples every activation map in the input independently with max.

Some useful tricks

Preprocessing



(covariance matrix is the identity matrix)

For an image : **contrast normalization**

Some useful tricks

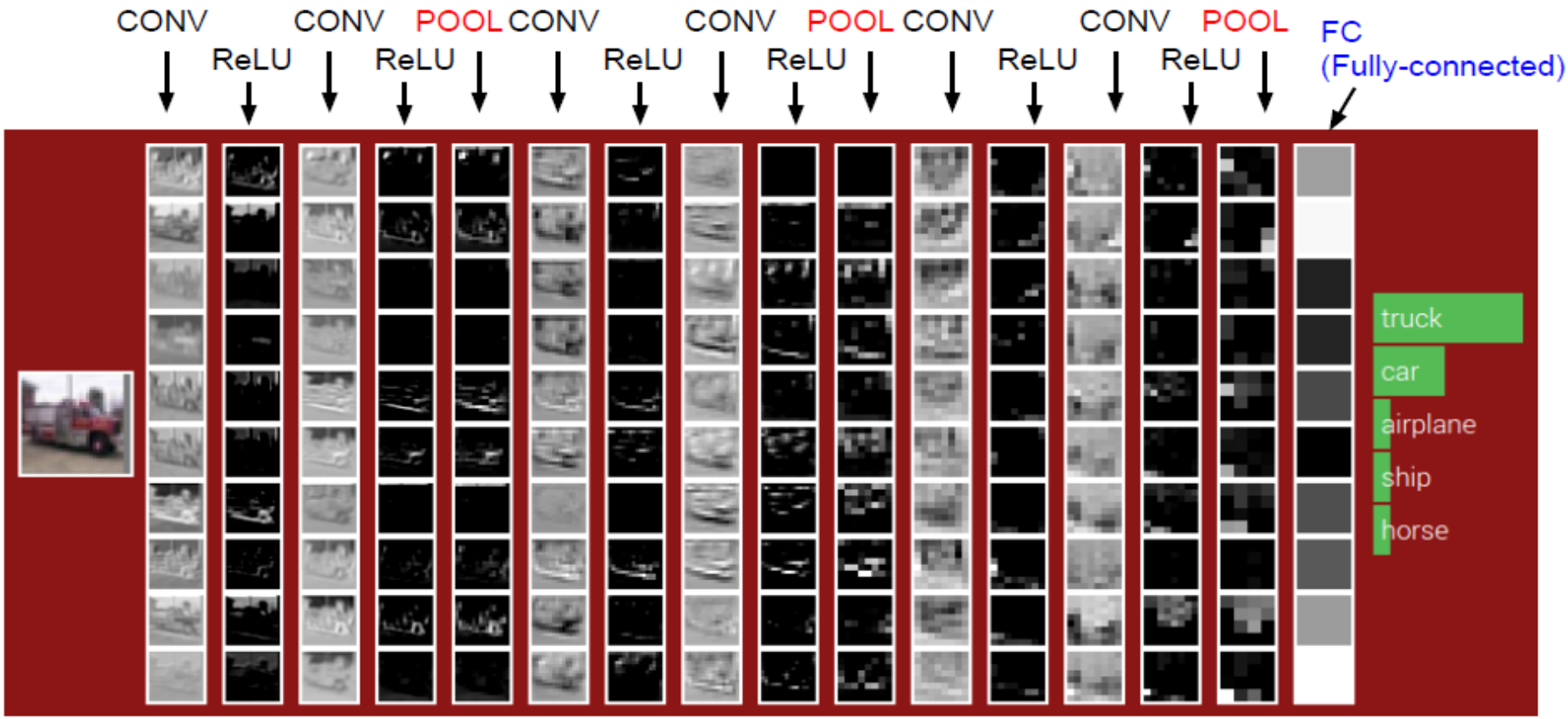
- Why is the deep learning revolution arriving just now?
- It used to be hard and cumbersome to train deep models due to **sigmoid** nonlinearities.

$$\sigma(a) = \frac{1}{1+\exp(-a)}$$

$$\tanh(a)$$

$$\mathrm{ReLU}(a) = \max(0, a)$$

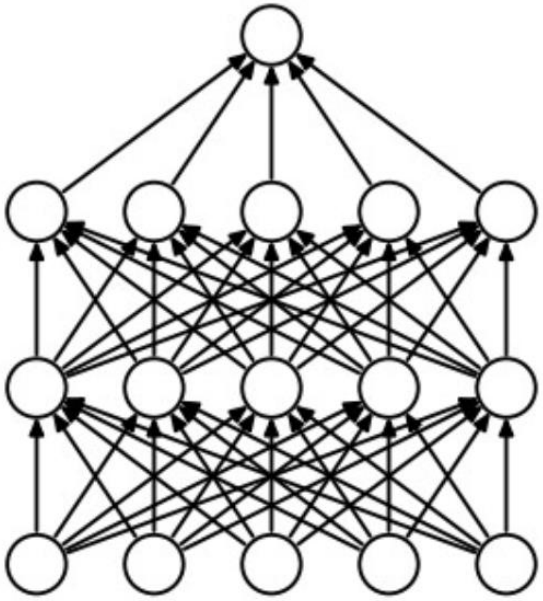
Conv Nets



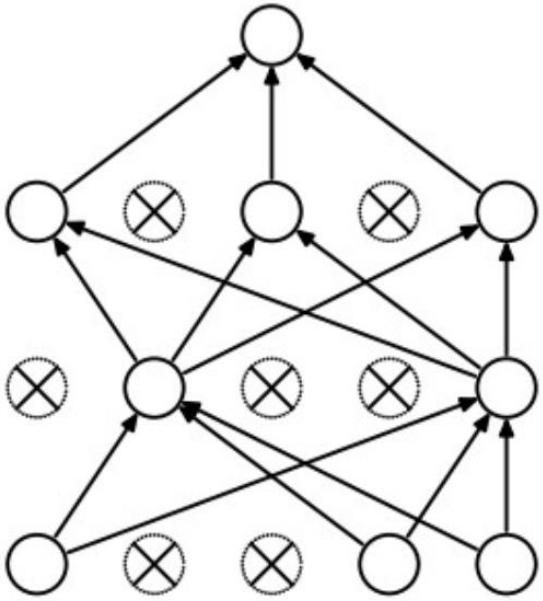
[Credit : Fei-Fei Li & Andrej Karpathy]

Conv Nets

Regularization: **Dropout**
"randomly set some neurons to zero"



(a) Standard Neural Net



(b) After applying dropout.

[Srivastava et al.]

Preventing overfitting

$$-\log p(\mathbf{Y}|\mathbf{X}) + \lambda_1 \|\mathbf{W}\|_1 + \lambda_2 \|\mathbf{W}\|^2$$

Encourages sparsity

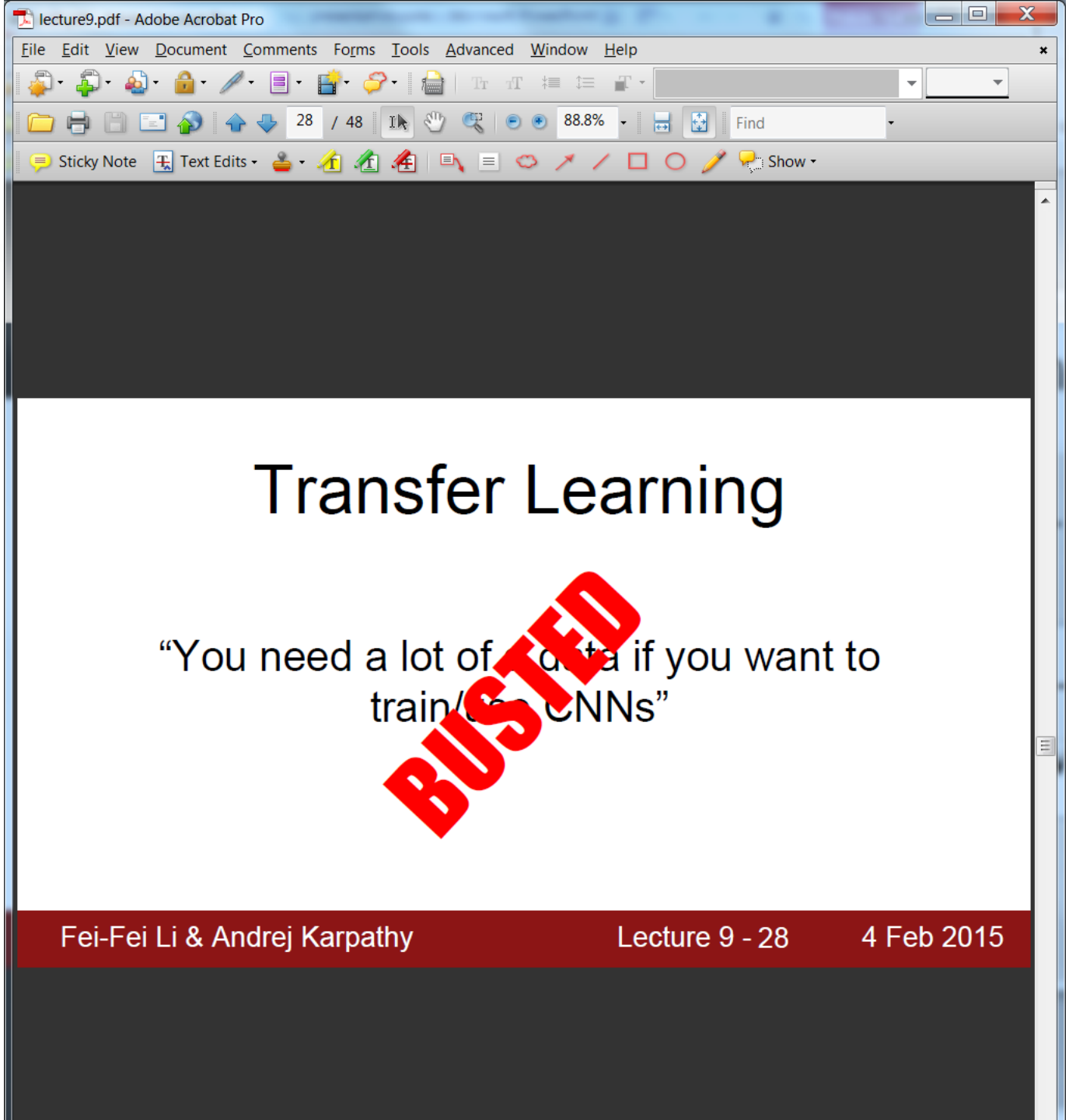
Encourages smaller weights
And large margins

Early stopping : stop training when the validation performace stop improving

Training - Momentum

$$\theta^{[k+1]} = \theta^{[k]} - \eta \nabla E\left(\theta^{[k]}\right)$$

$$\begin{aligned} \mathbf{V}_{i+1} &= \mu * \mathbf{V}_i - \alpha * \nabla \mathbf{W}_i \\ \mathbf{W}_{i+1} &= \mathbf{W}_i + \mathbf{V}_{i+1} \end{aligned}$$


lecture9.pdf - Adobe Acrobat Pro
File Edit View Document Comments Forms Tools Advanced Window Help
28 / 48
88.8%
Find
Sticky Note
Text Edits
Show
Transfer Learning
"You need a lot of data if you want to train/use CNNs"
BUSTED
Fei-Fei Li & Andrej Karpathy
Lecture 9 - 28
4 Feb 2015

Transfer Learning with CNNs

1. Train on Imagenet

2. If small dataset: fix all weights (treat CNN as fixed feature extractor), retrain only the classifier

i.e. swap the Softmax layer at the end

3. If you have medium sized dataset, **"finetune"** instead: use the old weights as initialization, train the full network or only some of the higher layers

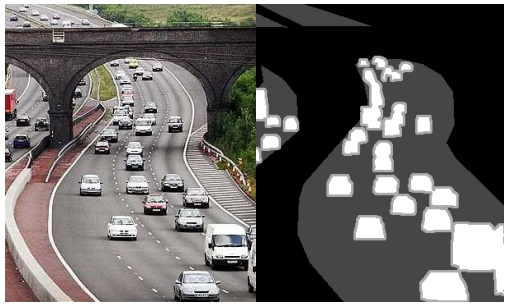
retrain bigger portion of the network, or even all of it.

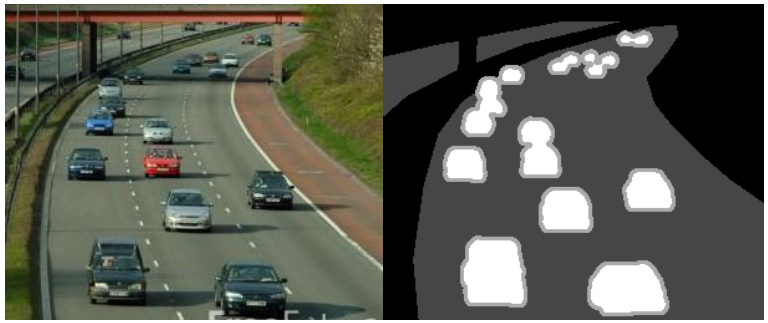
Results : traffic analysis

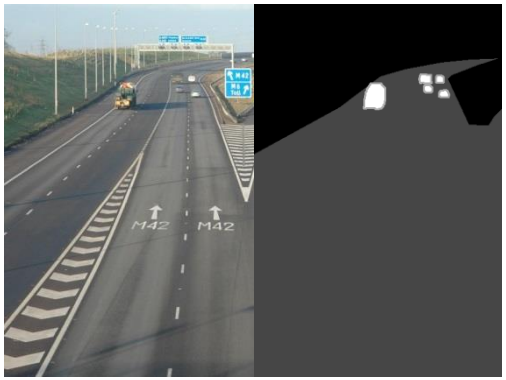
Table 3. Results of various methods on the UCSD dataset

Method	Accuracy	Method	Accuracy
Chan[5]	94.5%	Asmaa[12]	95.3%
Sobral[11]	94.5%	Caffe	**96.5%**
Derpanis[15]	95.3%	VGG (ROI)	**96.5%**
Riaz[9]	95.3%		

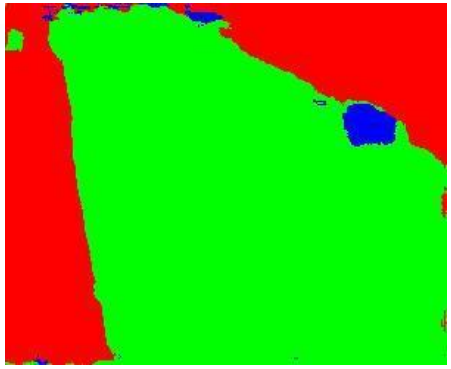
Results : traffic analysis



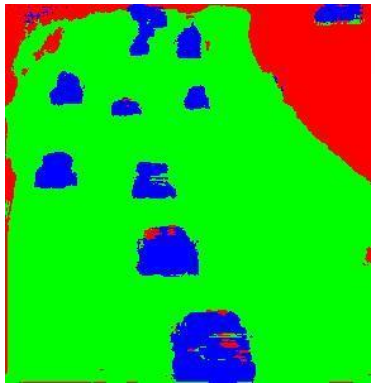




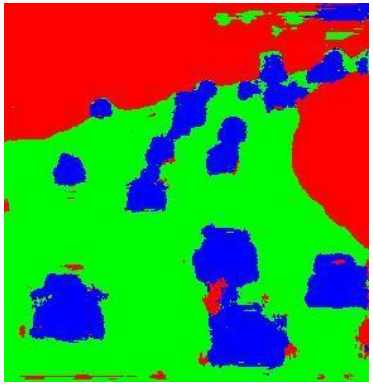
Results : traffic analysis

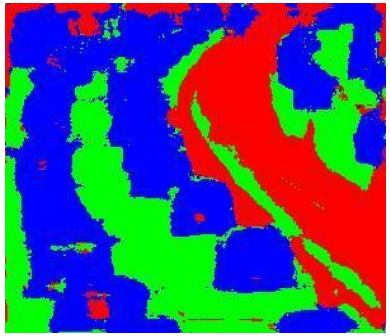
Empty

Fluid

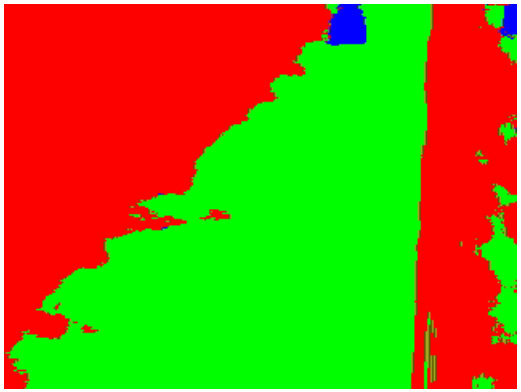
Heavy

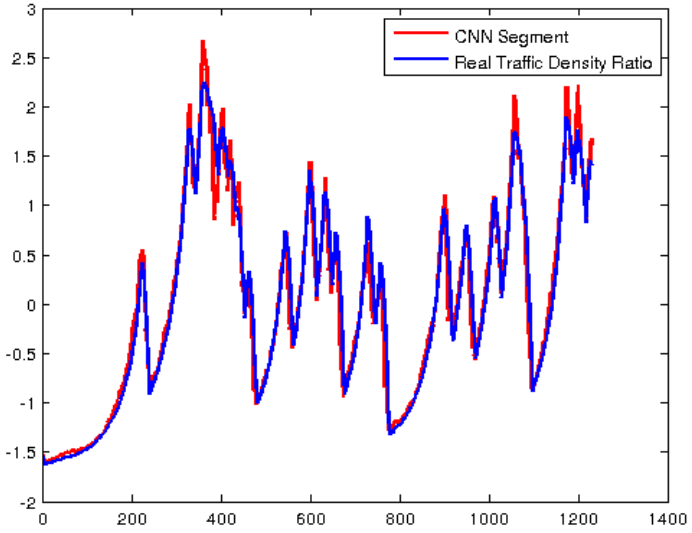
Jam

Results : traffic analysis

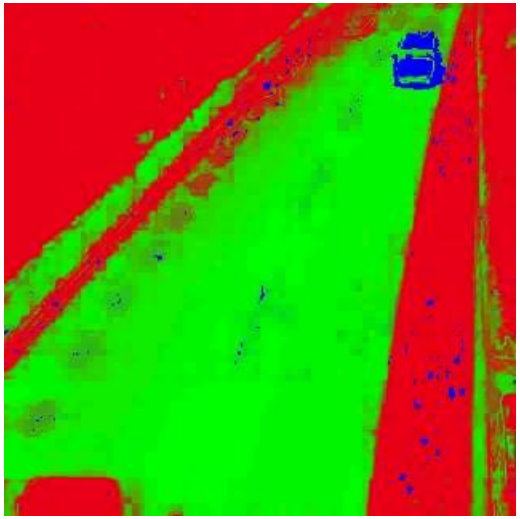
Frame-by-frame processing
No motion analysis
No post-processing

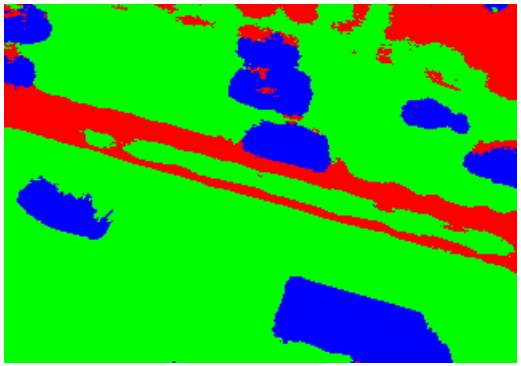


$$Density = \frac{Car}{Car + Road}$$



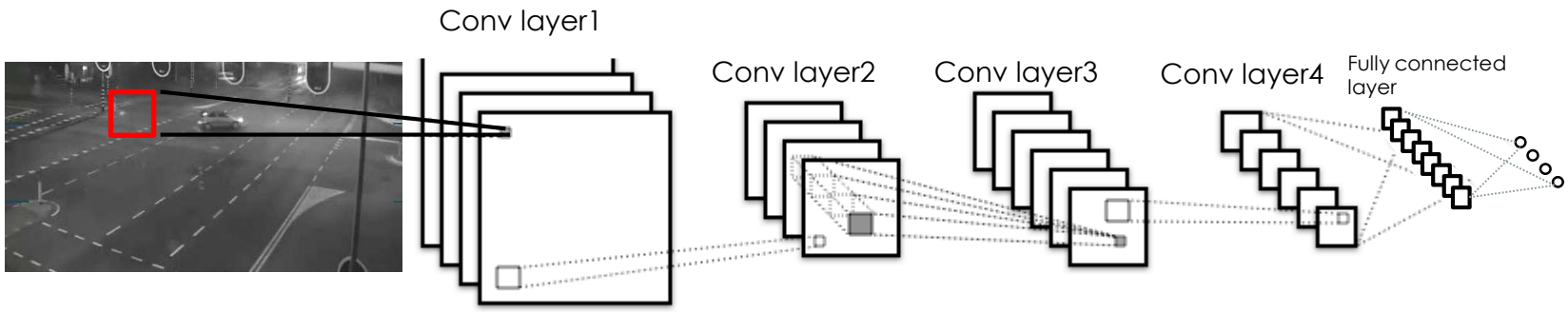
Results : traffic analysis





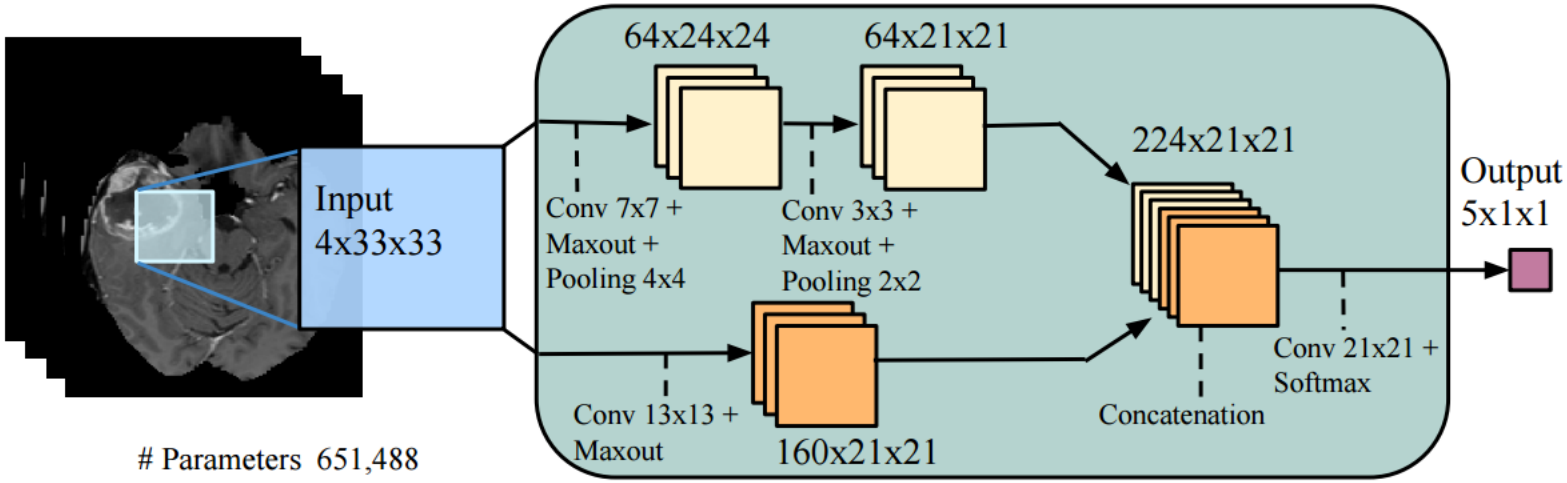
Results : traffic analysis

100 training frames

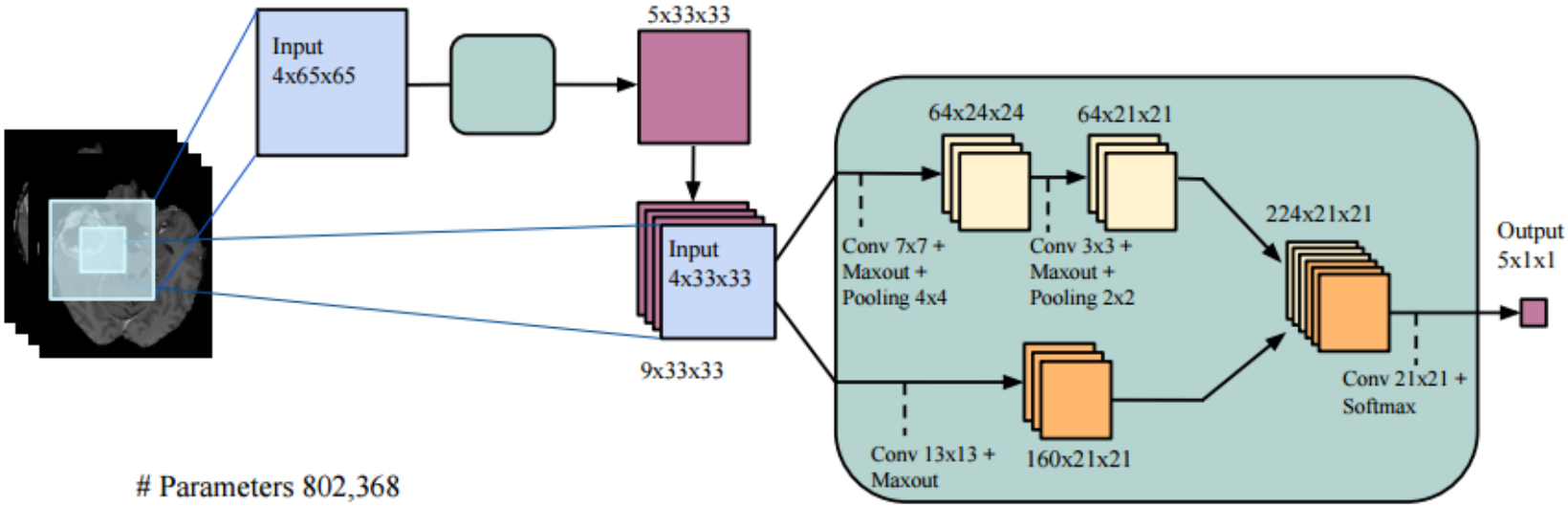


Method	Average ranking across categories	Average ranking	Average Re	Average Sp	Average FPR	Average FNR	Average PWC	Average F-Measure	Average Precision
Cascade [57]	2.36	1.00	0.9019	0.9949	0.0051	0.0981	0.6724	0.8601	0.8349
IUTIS-3 [25]	3.91	3.86	0.7779	0.9940	0.0060	0.2221	1.2985	0.7551	0.7875
SuBSENSE [14]	6.27	6.29	0.8124	0.9904	0.0096	0.1876	1.6780	0.7408	0.7509
FTSG (Flux Tensor with Split Gaussian mdoels)) [9]	6.45	7.29	0.7657	0.9922	0.0078	0.2343	1.3763	0.7283	0.7696
SaliencySubsense [17]	7.82	8.29	0.7714	0.9914	0.0086	0.2286	1.8969	0.7176	0.7628
Superpixel Strengthen Background Subtraction [26]	8.00	8.57	0.7416	0.9923	0.0077	0.2584	1.8902	0.7129	0.7754
M4CD Version 1.5 [29]	9.27	11.00	0.7895	0.9833	0.0167	0.2105	2.4032	0.6994	0.7275
M4CD Version 1.0 [27]	9.82	11.43	0.7750	0.9849	0.0151	0.2250	2.3609	0.6916	0.7320
Multimode Background Subtraction [20]	10.09	8.29	0.7389	0.9927	0.0073	0.2611	1.2614	0.7288	0.7382
EFIC v2.0 [28]	10.55	9.86	0.7976	0.9782	0.0218	0.2024	2.6316	0.7307	0.7543
Multimode Background Subtraction Version 0 (MBS V0) [19]	11.64	9.71	0.7192	0.9929	0.0071	0.2808	1.3922	0.7139	0.7435
EFIC [22]	12.45	13.00	0.7855	0.9779	0.0221	0.2145	2.7941	0.7088	0.7224
CwisarDH [7]	12.45	11.00	0.6608	0.9948	0.0052	0.3392	1.5273	0.6812	0.7725
Spectral-360 [8]	14.09	14.29	0.7345	0.9861	0.0139	0.2655	2.2722	0.6732	0.7054
IUTIS-2 [24]	15.55	18.14	0.6621	0.9838	0.0162	0.3379	3.1547	0.6026	0.7120
Bin Wang Apr 2014 [11]	16.45	17.14	0.7035	0.9794	0.0206	0.2965	2.9009	0.6577	0.7163
AAPSA [18]	17.09	17.00	0.6498	0.9905	0.0095	0.3502	2.0734	0.6179	0.6916
IUTIS-1 [23]	17.18	21.86	0.7654	0.9499	0.0501	0.2346	5.7503	0.5789	0.5928
KNN [3]	19.64	19.43	0.6650	0.9802	0.0198	0.3350	3.3200	0.5937	0.6788
GraphCutDiff [21]	19.82	23.57	0.6297	0.9780	0.0220	0.3703	3.6774	0.5684	0.6666
SC_SOBS [10]	19.91	19.86	0.7621	0.9547	0.0453	0.2379	5.1498	0.5961	0.6091
RMoG (Region-based Mixture of Gaussians) [16]	20.27	20.00	0.5940	0.9865	0.0135	0.4060	2.9638	0.5735	0.6965
Mahalanobis distance [6]	20.55	18.43	0.1644	0.9931	0.0069	0.8356	3.4750	0.2267	0.7403
SOBS_CF [15]	20.73	21.00	0.7805	0.9442	0.0558	0.2195	6.0709	0.5883	0.5831
KDE - ElGammal [2]	21.91	23.29	0.7375	0.9519	0.0481	0.2625	5.6262	0.5688	0.5811
CP3-online [12]	23.73	21.57	0.7225	0.9705	0.0295	0.2775	3.4318	0.5805	0.5559
GMM	Stauffer & Grimson [4]	23.82	22.00	0.6846	0.9750	0.0250	0.3154	3.7667	0.5707	0.6025
GMM	Zivkovic [5]	24.55	24.71	0.6604	0.9725	0.0275	0.3396	3.9953	0.5566	0.5973
Multiscale Spatio-Temporal BG Model [13]	26.82	26.29	0.6621	0.9542	0.0458	0.3379	5.5456	0.5141	0.5[illegible]

Medical Imaging



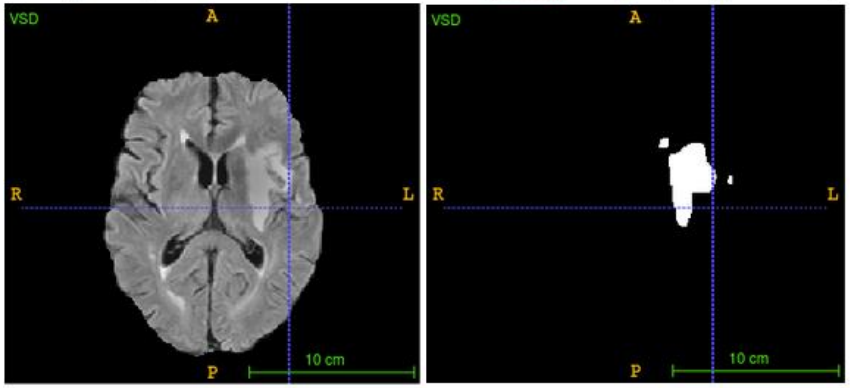
Parameters 651,488



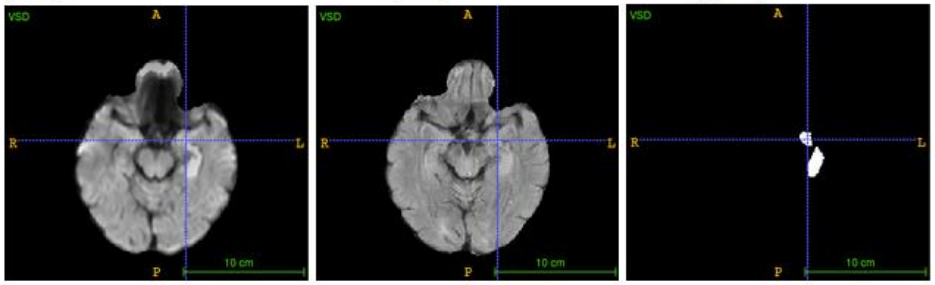
Parameters 802,368

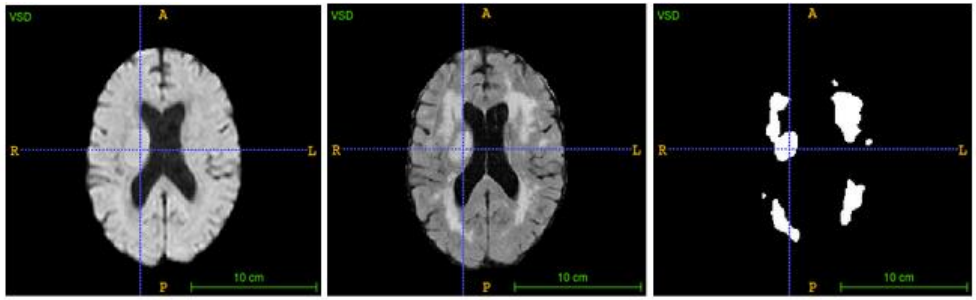
Results

ischemic stroke lesion segmentation (SISS) – 2015 Challenge



(DWI) (Flair) (pred)





Results

ischemic stroke lesion segmentation (SISS) – 2015 Challenge

Evaluation Results: SISS Training

Search:

		ASSD		Dice		Hausdorff Distance		Precision		Recall	
User	Covered Cases	average	std	average	std	average	std	average	std	average	std
doyls2	1 / 28	2.58	0.00	0.76	0.00	23.73	0.00	0.94	0.00	0.63	0.00
dutif1	28 / 28	8.92	19.23	0.69	0.30	31.75	28.52	0.72	0.31	0.67	0.31
kamnk1	28 / 28	5.00	10.33	0.66	0.24	55.93	28.55	0.77	0.24	0.63	0.25
fengc1	27 / 28	8.17	16.43	0.63	0.28	30.42	22.80	0.67	0.31	0.65	0.27
halmh1	28 / 28	6.44	12.93	0.61	0.24	30.96	29.40	0.68	0.26	0.58	0.25
rezas1	28 / 28	5.99	4.28	0.59	0.23	83.28	22.62	0.51	0.25	0.79	0.15
maieo1	28 / 28	7.91	13.09	0.58	0.29	34.05	29.01	0.68	0.32	0.58	0.30
robbd1	28 / 28	9.36	13.85	0.57	0.28	53.88	34.58	0.58	0.33	0.68	0.21
chenl2	28 / 28	12.42	14.29	0.55	0.29	77.17	28.60	0.52	0.32	0.69	0.27
mahmq2	28 / 28	10.30	11.11	0.54	0.26	82.78	23.95	0.67	0.33	0.50	0.25
haect1	27 / 28	14.43	25.88	0.53	0.26	69.67	30.77	0.62	0.31	0.56	0.29
pinta1	28 / 28	12.18	22.59	0.50	0.31	43.21	30.50	0.61	0.34	0.55	0.33
muscj1	28 / 28	56.77	79.90	0.48	0.38	76.88	81.77	0.57	0.43	0.44	0.37
jessa1	27 / 28	11.59	18.34	0.45	0.24	39.23	30.70	0.52	0.26	0.51	0.31

Results

ischemic stroke lesion segmentation (SISS) – 2015 Challenge

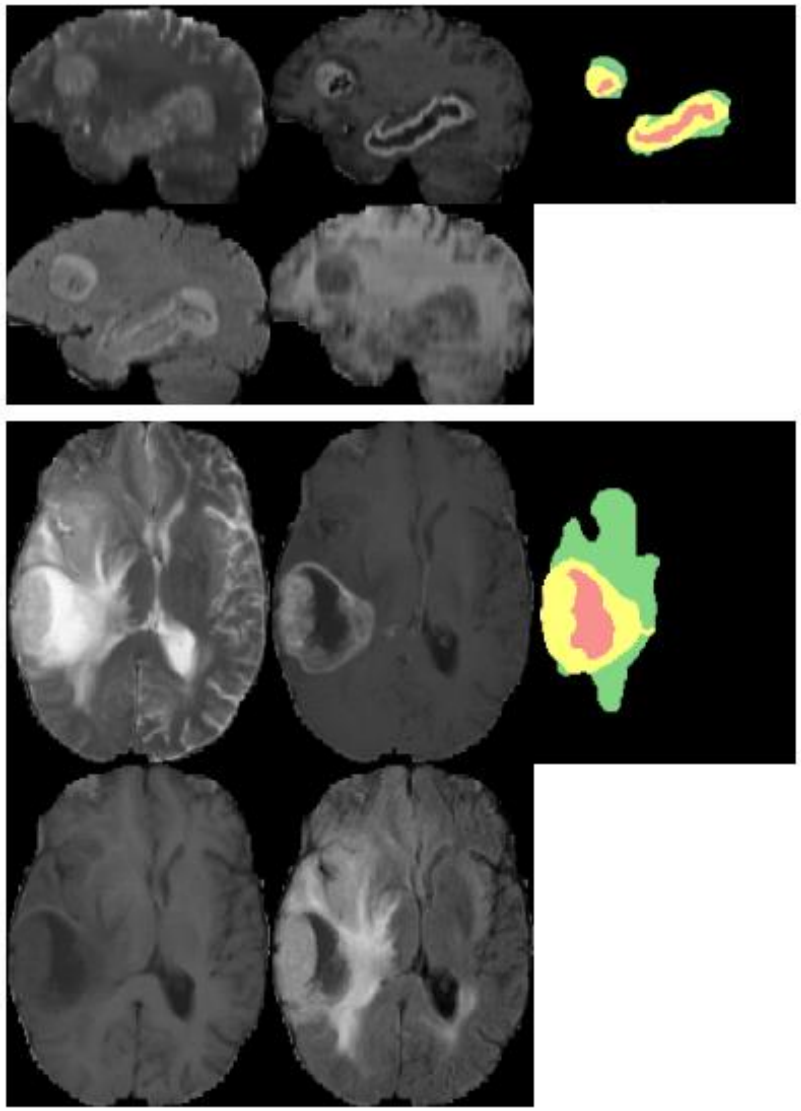
Evaluation Results: SPES Training

Search:

		ASSD		Dice		Hausdorff Distance		Precision		Recall	
User	Covered Cases	average	std	average	std	average	std	average	std	average	std
dutif1	30 / 30	1.76	0.94	0.85	0.08	23.28	14.13	0.83	0.11	0.88	0.08
mckir1	30 / 30	1.42	1.01	0.85	0.06	30.71	18.91	0.84	0.10	0.87	0.07
maieo1	30 / 30	1.38	0.66	0.83	0.06	23.33	13.02	0.83	0.10	0.85	0.09
fengc1	30 / 30	1.64	0.59	0.82	0.05	23.43	13.14	0.83	0.08	0.81	0.07
robbd1	30 / 30	2.03	1.35	0.82	0.07	44.29	27.59	0.81	0.14	0.85	0.07
kelle1	30 / 30	1.82	0.60	0.79	0.06	23.80	12.21	0.80	0.07	0.81	0.12
haect1	30 / 30	3.52	2.18	0.78	0.08	45.89	24.89	0.78	0.11	0.80	0.12

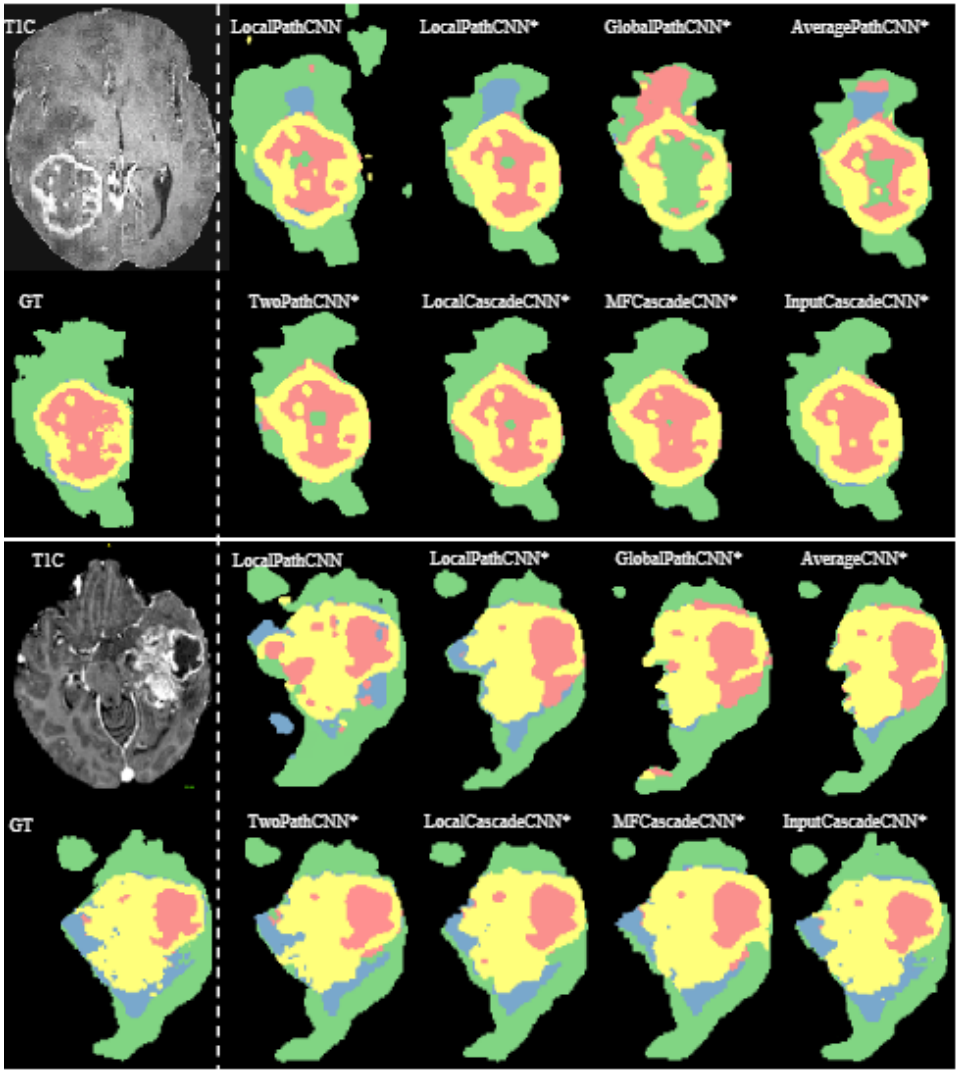
Results

Brain tumor segmentation – 2015 Challenge



Results

Brain tumor segmentation – 2015 Challenge



Results

Brain tumor segmentation – 2015 Challenge

Method	Dice			Specificity			Sensitivity		
	Complete	Core	Enhancing	Complete	Core	Enhancing	Complete	Core	Enhancing
InputCascadeCNN*	0.88	0.79	0.73	0.89	0.79	0.68	0.87	0.79	0.80
Tustison	0.87	0.78	0.74	0.85	0.74	0.69	0.89	0.88	0.83
MFCascadeCNN*	0.86	0.77	0.73	0.92	0.80	0.71	0.81	0.76	0.76
TwoPathCNN*	0.85	0.78	0.73	0.93	0.80	0.72	0.80	0.76	0.75
LocalCascadeCNN*	0.88	0.76	0.72	0.91	0.76	0.70	0.84	0.80	0.75
LocalPathCNN*	0.85	0.74	0.71	0.91	0.75	0.71	0.80	0.77	0.73
Meier	0.82	0.73	0.69	0.76	0.78	0.71	0.92	0.72	0.73
Reza	0.83	0.72	0.72	0.82	0.81	0.70	0.86	0.69	0.76
Zhao	0.84	0.70	0.65	0.80	0.67	0.65	0.89	0.79	0.70
Cordier	0.84	0.68	0.65	0.88	0.63	0.68	0.81	0.82	0.66
TwoPathCNN	0.78	0.63	0.68	0.67	0.50	0.59	0.96	0.89	0.82
LocalPathCNN	0.77	0.64	0.68	0.65	0.52	0.60	0.96	0.87	0.80
Festa	0.72	0.66	0.67	0.77	0.77	0.70	0.72	0.60	0.70
Doyle	0.71	0.46	0.52	0.66	0.38	0.58	0.87	0.70	0.55