Imperial College London

Introduction into CNNs for medical imaging

Professor Daniel Rueckert, FREng, FIEEE
Biomedical Image Analysis Group
Department of Computing
Imperial College London, UK

Thanks to Kostas Kamnitsas for many of the slides

Artificial Intelligence and Machine Learning is everywhere!

Advertisement & social media

Robotics

Autonomous navigation

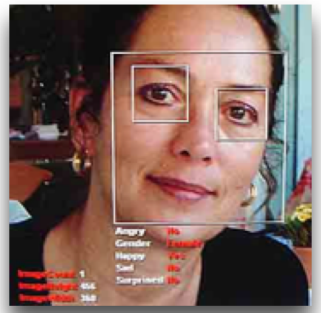
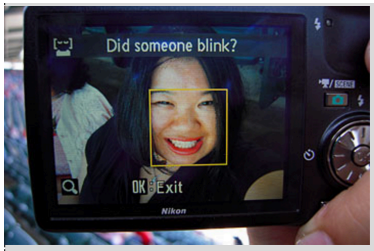


Image and speech understanding

Artificial Intelligence and Machine Learning is everywhere!

TIME

Are Face-Detection Cameras Racist?



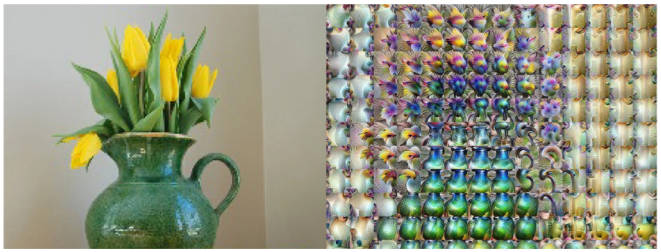
"You would think that Nikon, being a Japanese company, would have designed this with Asian eyes in mind."

- Random commenter(url)

The New York Times

30

Google Researchers Are Learning How Machines Learn



make a contribution | subscribe / find a job | theguardian

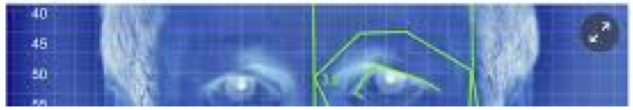
news / opinion / sport / arts / life

tech / US / world / environment / US politics / business / more

Artificial intelligence (AI)

New AI can guess whether you're gay or straight from a photograph

An algorithm deduced the sexuality of people on a dating site with up to 91% accuracy, raising tricky ethical questions



Artificial Intelligence and Machine Learning

THE NEW YORKER

APRIL 3, 2017 ISSUE

A.I. VERSUS M.D.

What happens when diagnosis is automated?

By Siddhartha Mukherjee

THE WALL STREET JOURNAL.

Subscribe Now | Sign In

SPECIAL OFFER: JOIN NOW

Europe Edition | July 28, 2018 | Today's Paper | Video

Home World U.S. Politics Economy Business Tech Markets Opinion Life & Arts Real Estate WSJ. Magazine

U.S. NEWS
U.S. Military Begins Arduous Job: Identifying Remains From Korean ...

OPINION
Devin Nunes, Washington's Public Enemy No. 1

OPINION
Papa John Defends Himself Large, Plain and Proud

WSJ WHAT'S NEWS

ECONOMY | CAPITAL ACCOUNT

How Robots May Make Radiologists' Jobs Easier, Not Redundant

Artificial intelligence programs that diagnose disease by analyzing images still require human judgment

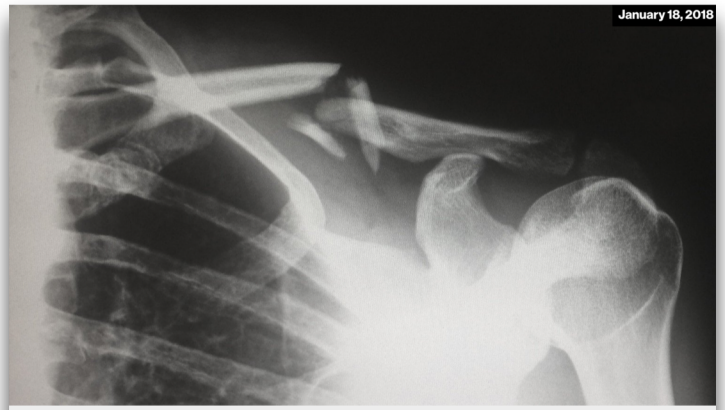
By Greg Ip
Updated Nov. 22, 2017 12:56 p.m. ET

Earlier this month a team of computer scientists at Stanford

Recommended Videos

1 Can Grownups Do

MIT Technology Review



AI Is Continuing Its Assault on Radiologists

A new model can detect abnormalities in x-rays better than radiologists—in some parts of the body, anyway.

www.news.cn 新华网 NEWS www.xinhuanet.com XINHUANET

Monday, July

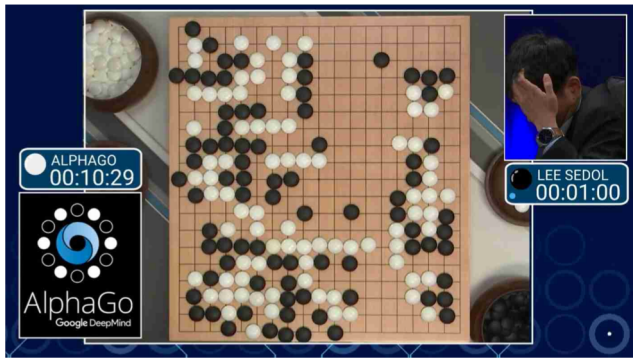
China Focus: AI beats human doctors in neuroimaging recognition contest

Artificial Intelligence and Machine Learning

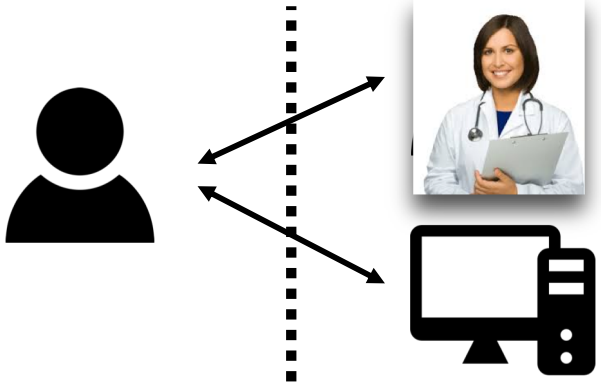
- What are the differences?

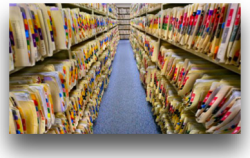
Artificial intelligence (AI) involves machines that can perform tasks that are characteristic of human intelligence

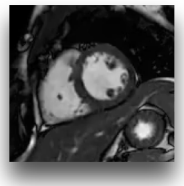
Play games ?



Turing test ?





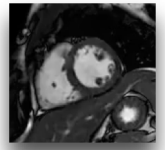


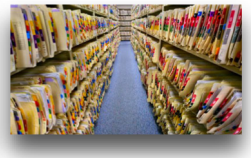
Artificial Intelligence and Machine Learning

- What are the differences?

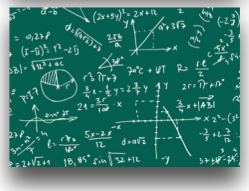
Machine Learning (ML) is a form of computational statistics:
Data + Model ⟶ Prediction

Data: observations



Model: assumptions; priors based on previous experience; inductive bias



Prediction: action to be taken; categorisation or a quality score

Machine learning in medical imaging

“They should stop training radiologists now.”
Geoffrey Hinton (godfather of deep learning) in 2017

"To the question, will AI replace radiologists, I say the answer is no…”

“… but radiologists who do AI will replace radiologists who don’t."
Curtis Langlotz in 2017


RSNA News

Machine Learning Plays Central Role at RSNA 2017

BY MIKE BASSETT

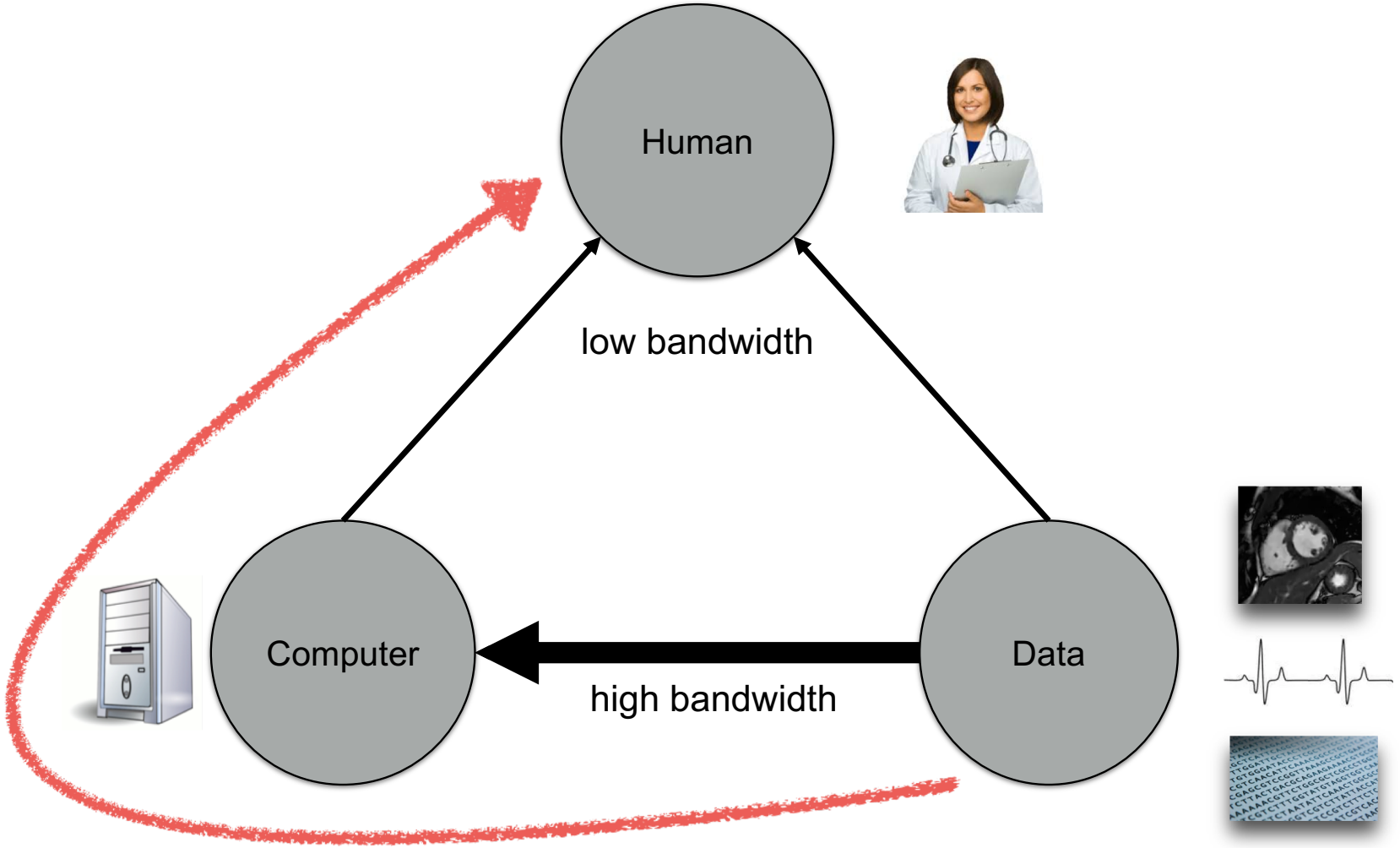
November 1, 2017

Share Tweet Share Email

Machine Learning (ML) and the role it will play in the future of radiology will be central to a broad scope of programming at RSNA 2017.

Langlotz

How can we leverage machine learning?



Types of Machine Learning

- Supervised learning
  – Learn to predict an output for a given input

(most successful to date)

focus of this talk

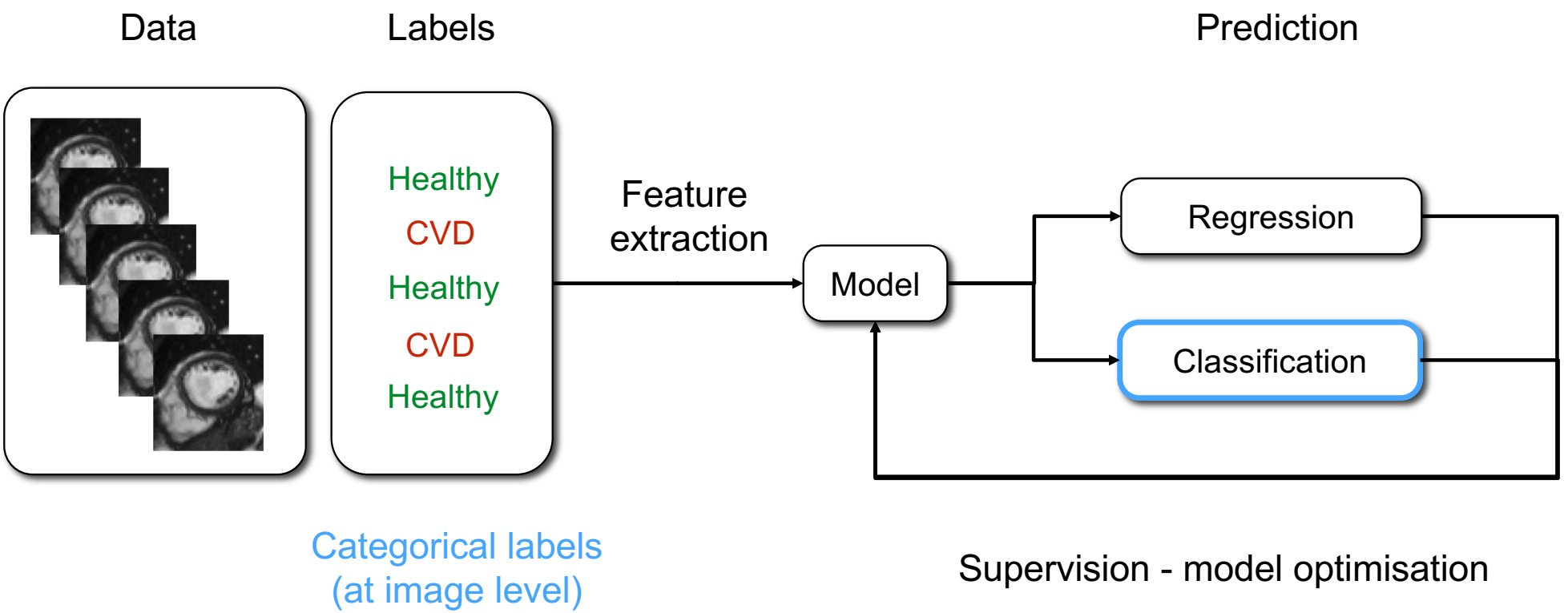
- Unsupervised learning
  – Discover patterns and reduce dimensionality

(most useful in the real-world)

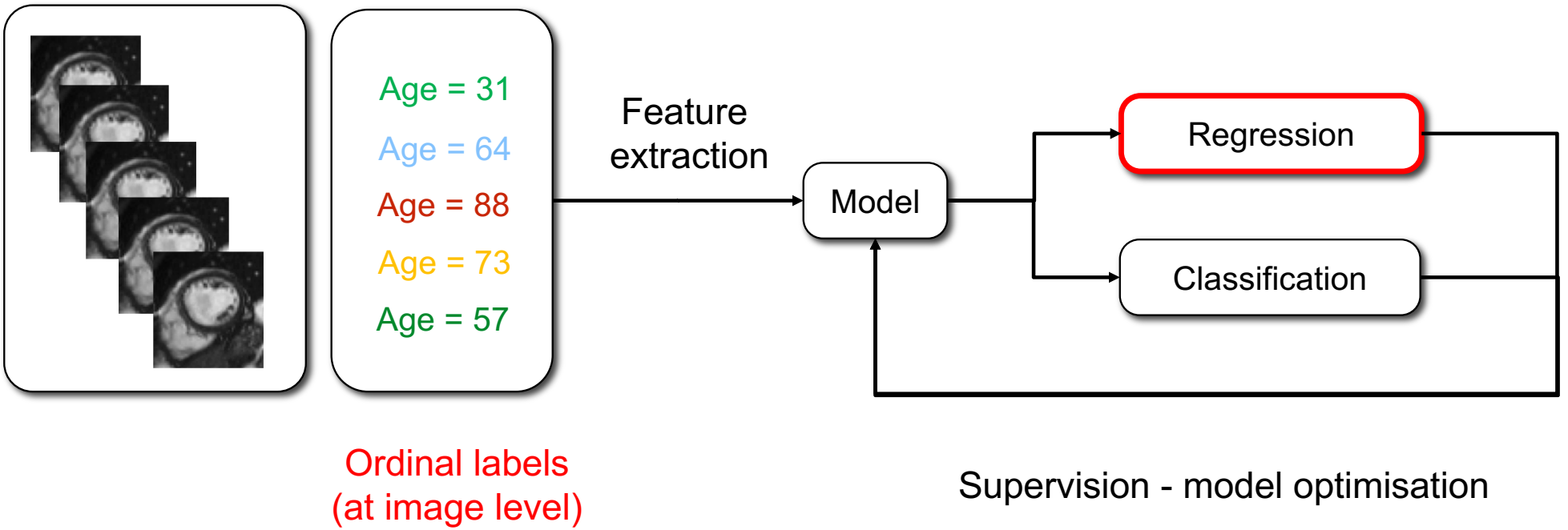
- Reinforcement learning
  – Learn what action to take to maximize a reward

(useful in scenarios such as games)

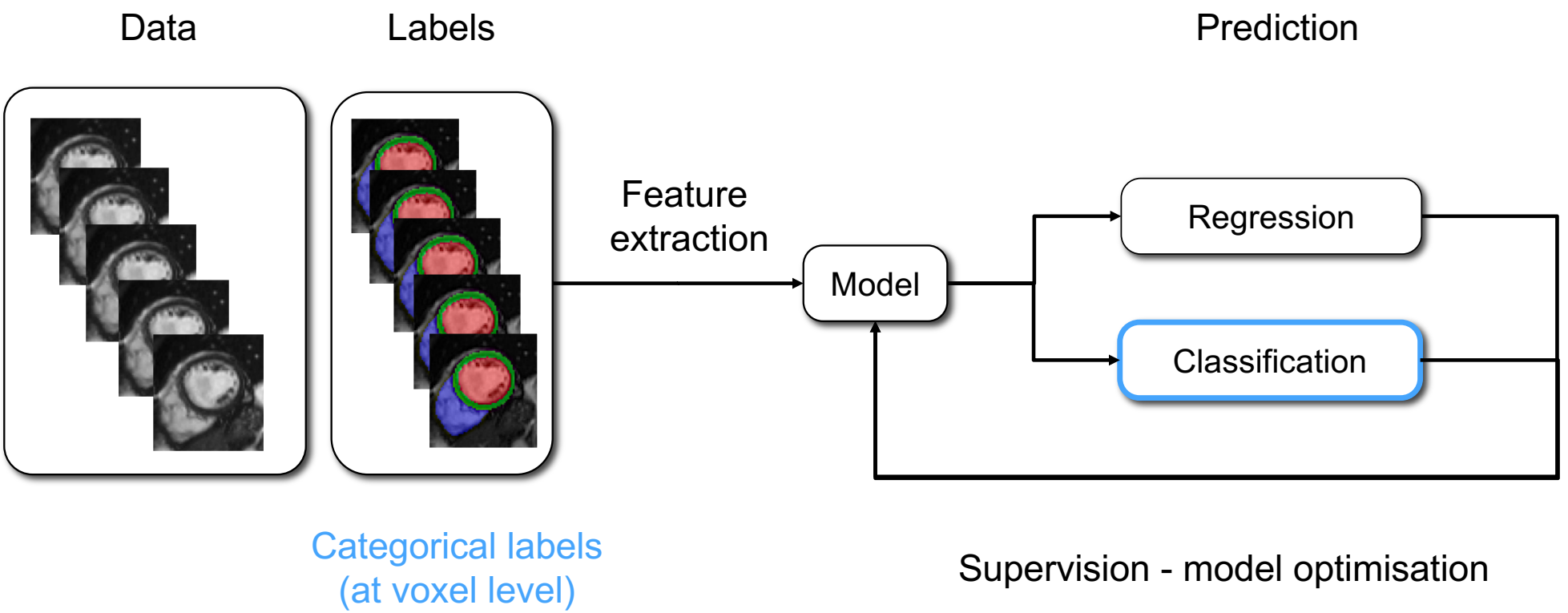
Supervised learning



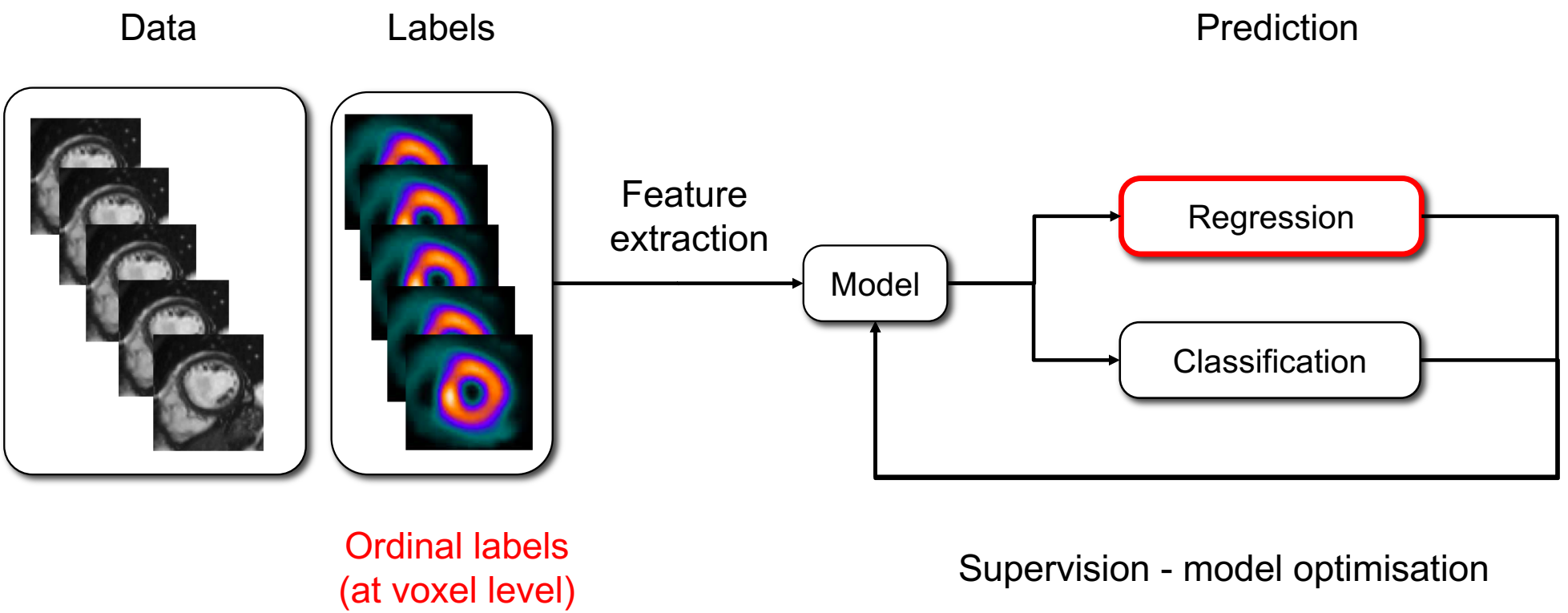
Supervised learning



Supervised learning



Supervised learning



Supervised learning

- Given a vector of input features $\mathbf{x}$, we want to learn a predictor $h_\Theta(\mathbf{x})$, that outputs an accurate prediction y

$$h_\Theta(\mathbf{x}) = y$$

hypothesis

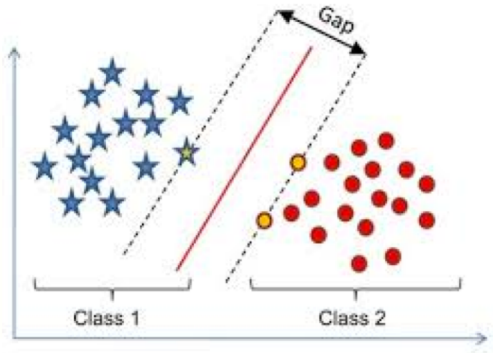
parameters

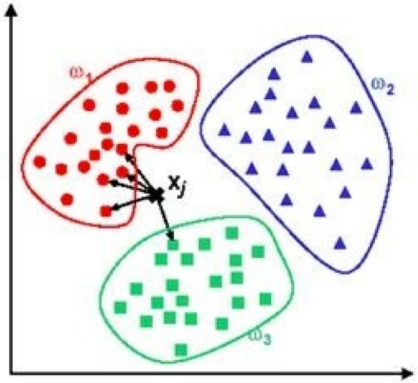
- Using a training set $T = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{m}$ with m examples, the predictor is trained (somehow) such that

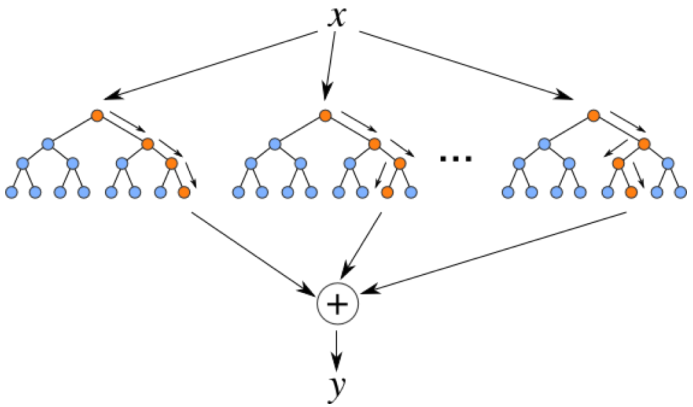
$$\forall i \left[h_\Theta(\mathbf{x}^{(i)}) \approx y^{(i)} \right]$$

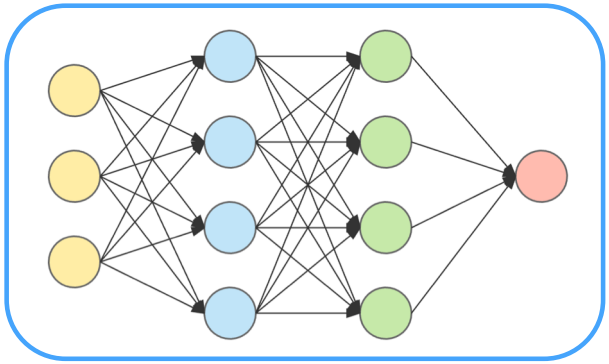
Supervised learning

- What form is $h_\Theta(\mathbf{x})$?
- Many options:
  - Support Vector Machines (SVM)
  - Logistic regression
  - Naive Bayes
  - Linear discriminant analysis
  - Decision trees or forests
  - K-nearest neighbors
  - Neural Networks



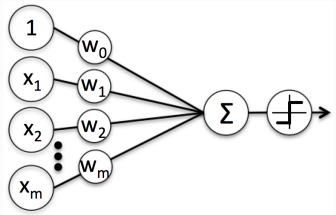


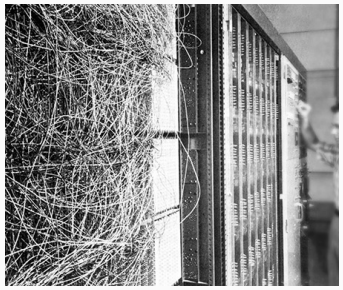




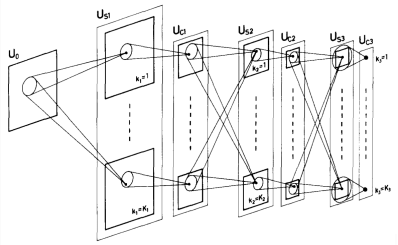
Some history on neural networks…

- **Perceptron** [Rosenblatt 1958]:





- **Neocognitron** [Fukushima 1980]:



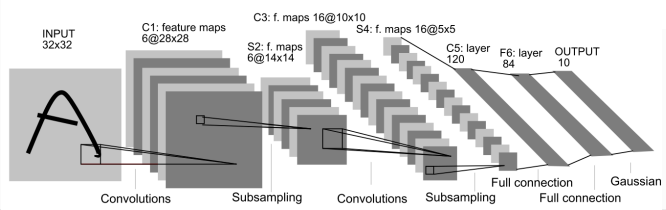
- **Back-propagation** [Rumelhart 1985]:

$$\Delta_p w_{ji} = \eta \delta_{pj} o_{pi}$$

$$\delta_{pj} = (t_{pj} - o_{pj}) f'_j(net_{pj})$$

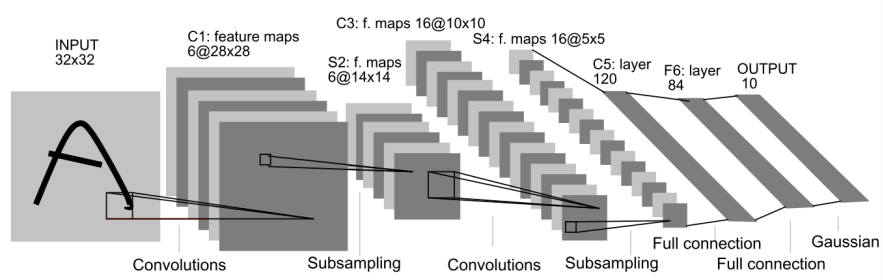
$$\delta_{pj} = f'_j(net_{pj}) \sum_k \delta_{pk} w_{kj}$$

- **CNN** [LeCunn1989, 1998]:



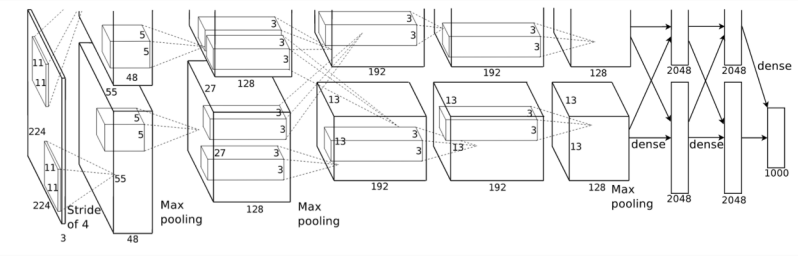
The deep learning revolution

LeNet [LeCun1989, 1998]:

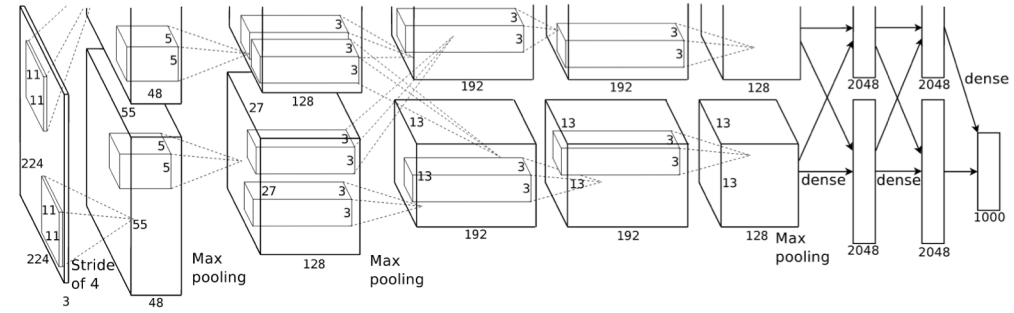


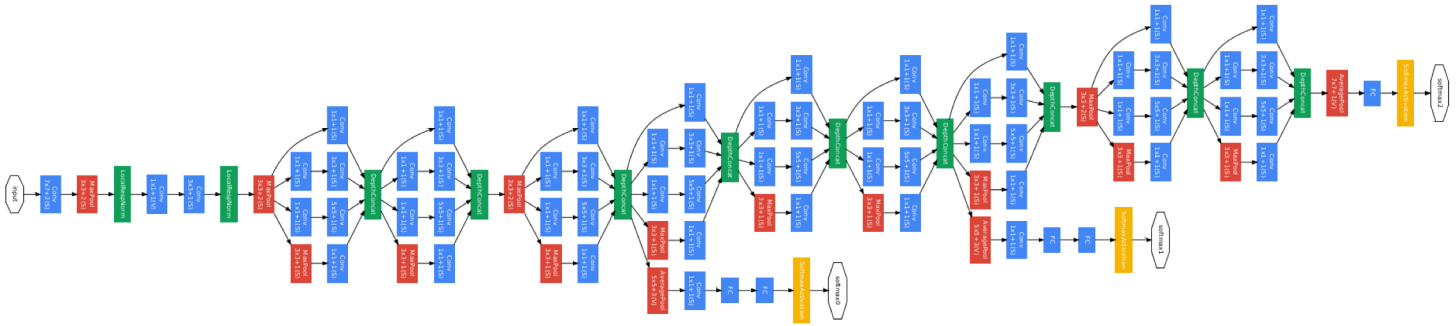
General Purpose GPU (GPCPU), e.g. CUDA 2007:

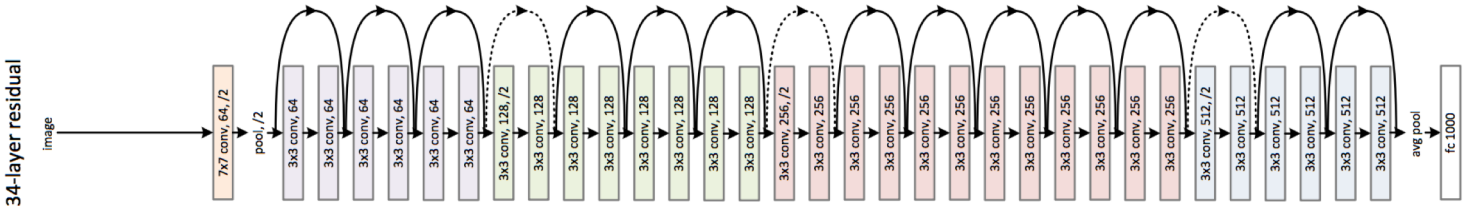
AlexNet [Krizhevsky 2012]:



The deep learning (r)evolution

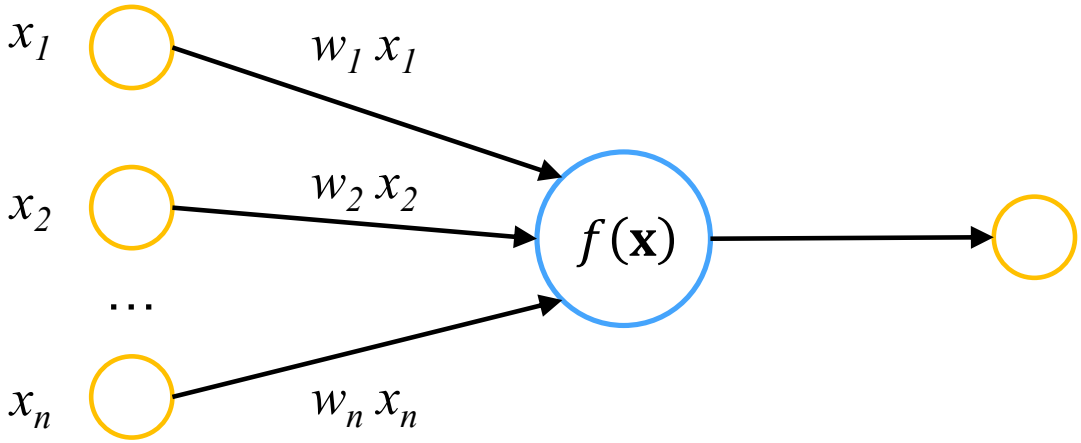






Perceptron

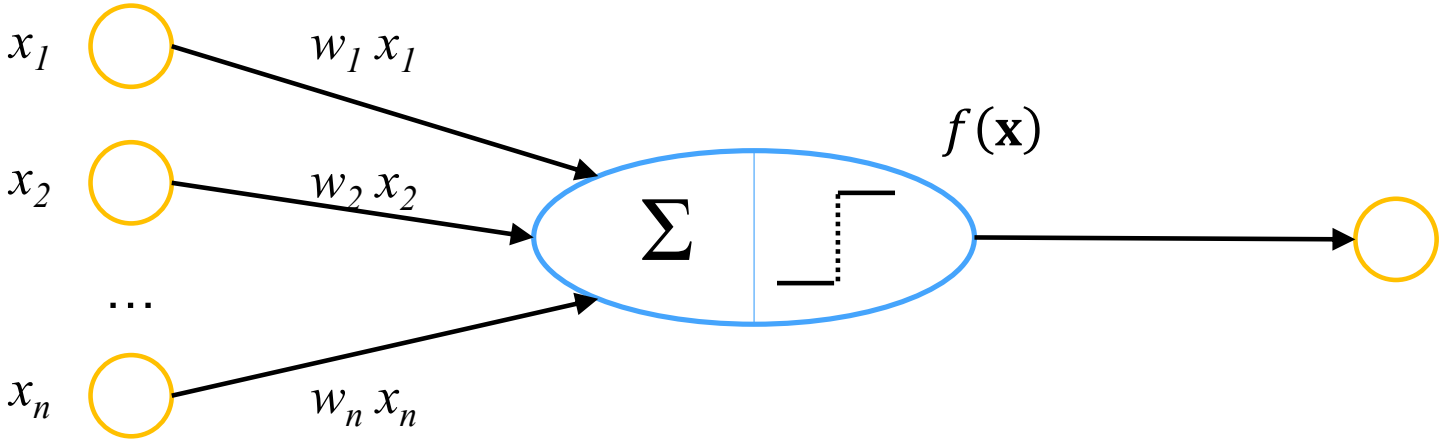
- A perceptron takes several inputs, $x_1, x_2, \ldots, x_n$ and produces a single binary output $f(\mathbf{x})$:



$$f(\mathbf{x}) = \begin{cases} 1 & \text{if } \sum_i w_i x_i > \text{threshold} \\ 0 & \text{if } \sum_i w_i x_i \leq \text{threshold} \end{cases}$$

Perceptron

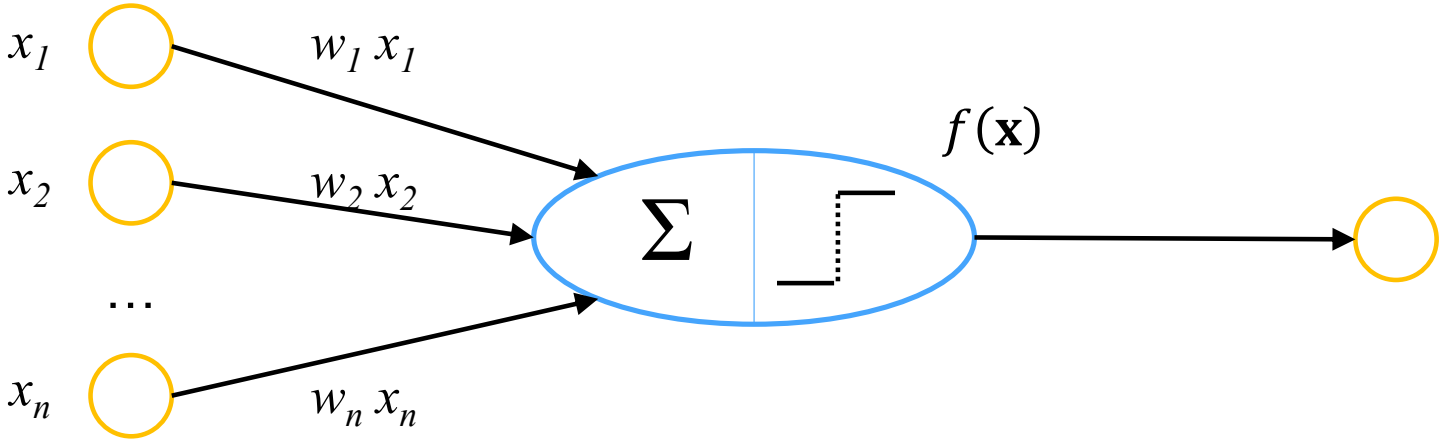
- A perceptron takes several inputs, $x_1, x_2, \ldots, x_n$ and produces a single binary output $f(\mathbf{x})$:



$$f(\mathbf{x}) = \begin{cases} 1 & \text{if } \sum_i w_i x_i > \text{threshold} \\ 0 & \text{if } \sum_i w_i x_i \leq \text{threshold} \end{cases}$$

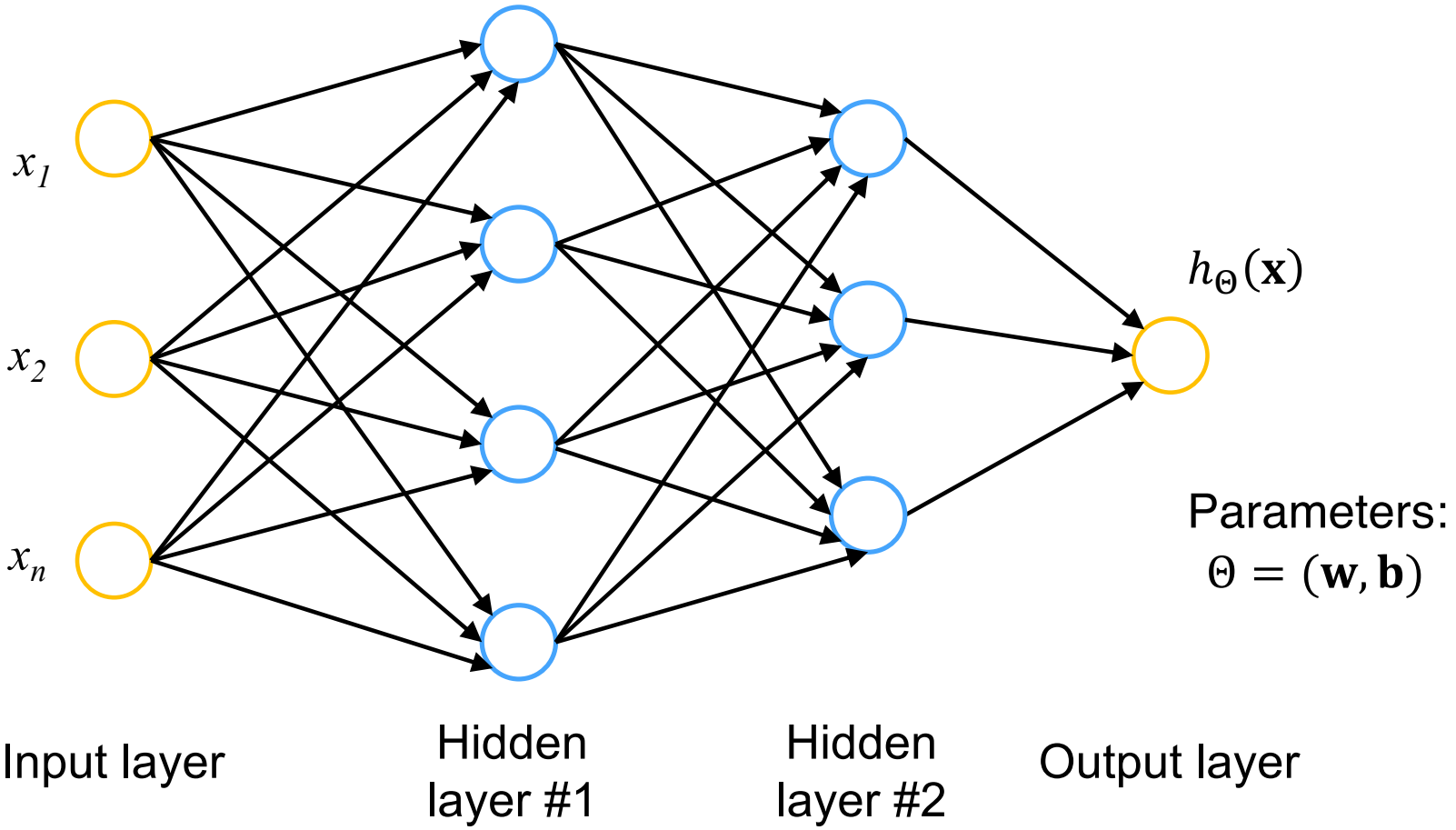
Perceptron

- A perceptron takes several inputs, $x_1, x_2, \ldots, x_n$ and produces a single binary output $f(\mathbf{x})$:



$$f(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{w} \cdot \mathbf{x} + b > 0 \\ 0 & \text{otherwise} \end{cases}$$

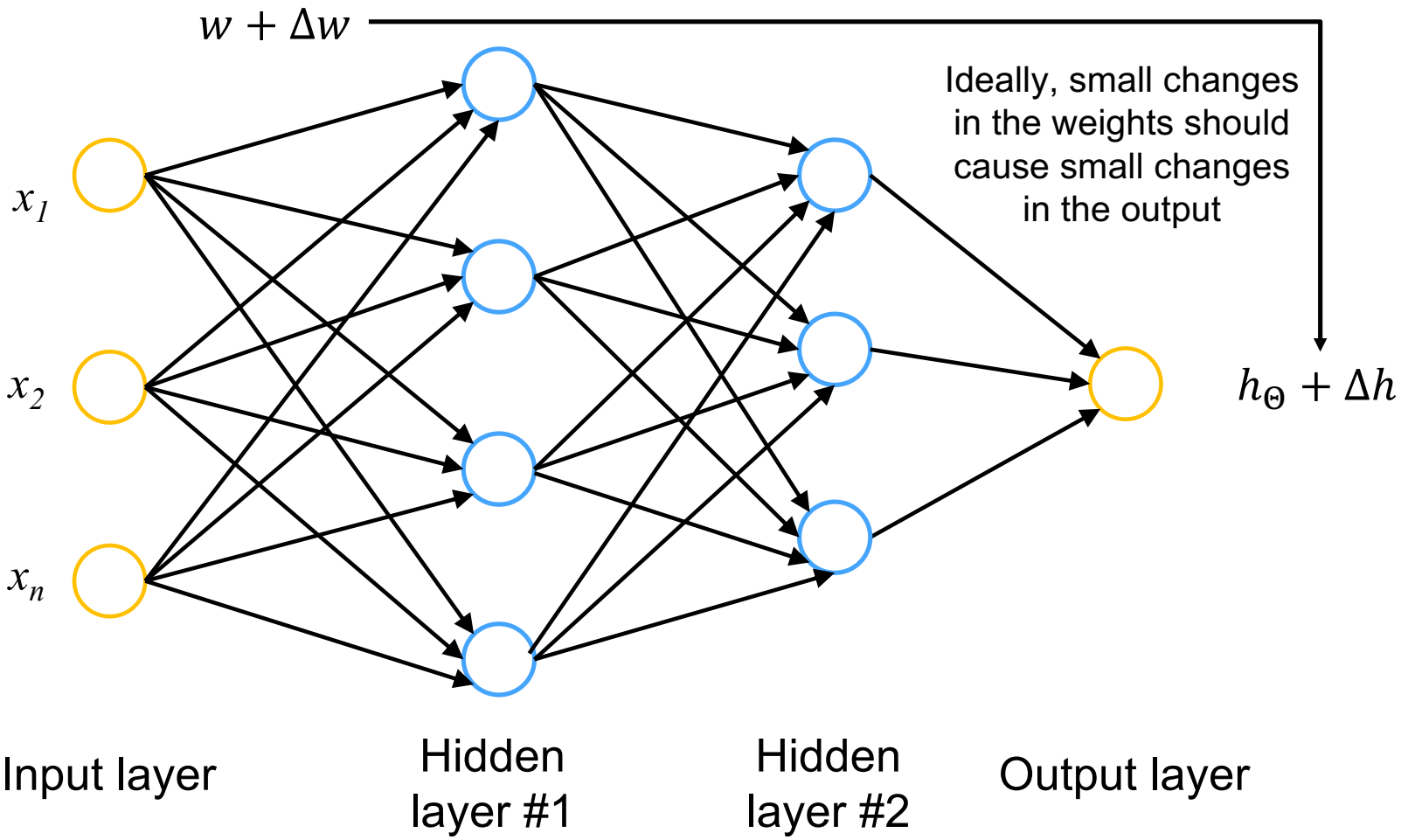
Multi-layer perceptron



Learning with neural networks

Learning means adjusting the parameters $\Theta = (\mathbf{w}, \mathbf{b})$ of the network

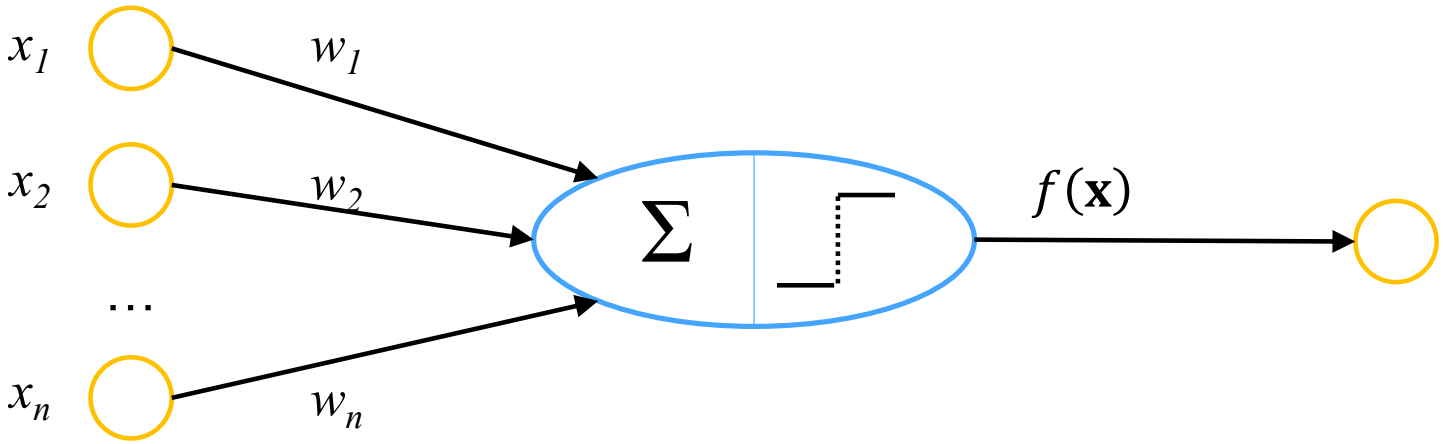
Learning with neural networks



Learning with neural networks

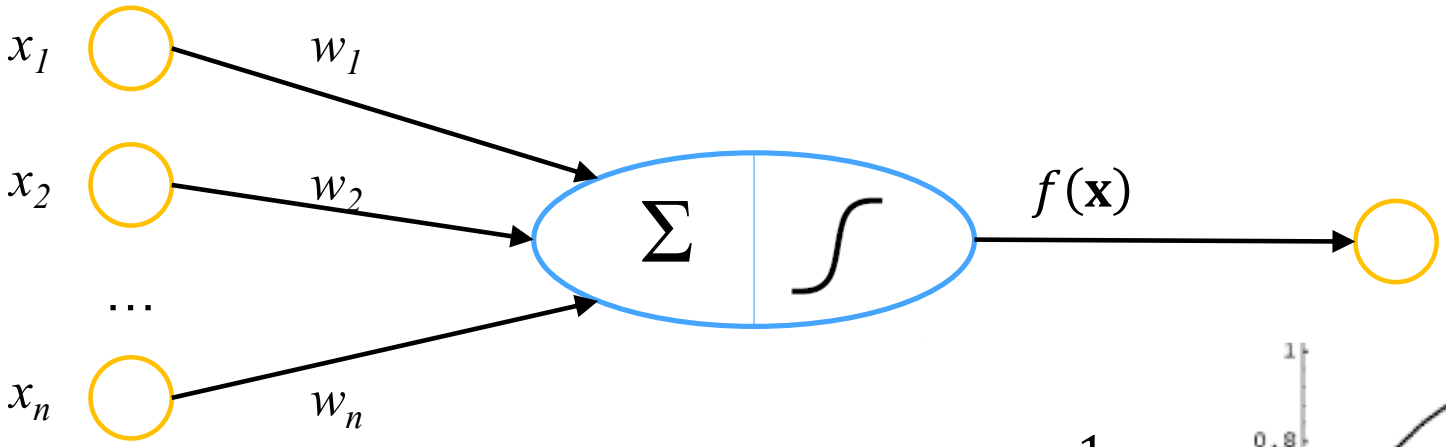
However, small changes of weights $\mathbf{w}$ can lead to large changes in the output

Perceptron



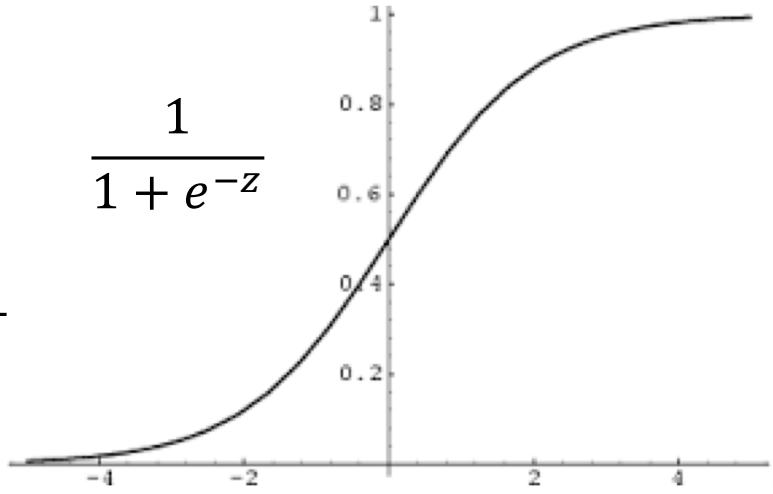
$$f(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{w} \cdot \mathbf{x} + b > 0 \\ 0 & \text{otherwise} \end{cases}$$

Neurons



$$f(\mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{w} \cdot \mathbf{x})}$$

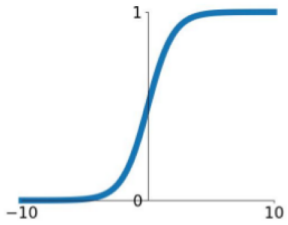
$$\frac{1}{1 + e^{-z}}$$



Neurons: Activation functions

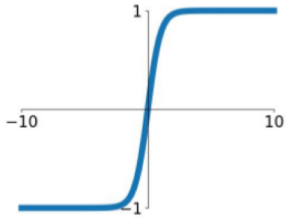
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



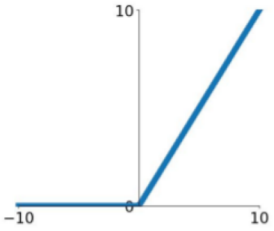
tanh

$$\tanh(x)$$



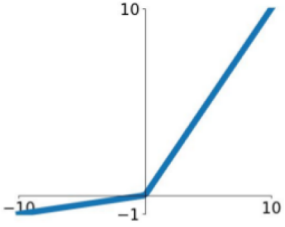
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

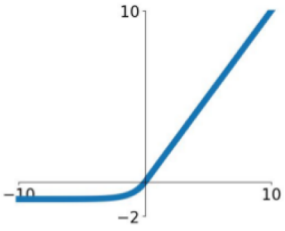


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Back to learning with neural networks

- Let us assume the following loss function

$$\mathcal{L} = \frac{1}{N}\sum_{j=1}^{N}(h_{\Theta}(\mathbf{x}_j) - y_j)^2$$

- In this case the parameters are $\boldsymbol{\Theta} = (\mathbf{w}, b)$, so

$$\mathcal{L} = \frac{1}{N}\sum_{j=1}^{N}(h_{(\mathbf{w},b)}(\mathbf{x}_j) - y_j)^2$$

- $\mathcal{L}$ can be minimized using gradient descent

Requires calculation of partial derivatives loss $\mathcal{L}$ with respect to Θ_j, i.e. all weights and biases

Learning NN parameters via gradient descent

- Given a output h_Θ and loss function $\mathcal{L}(\Theta)$

repeat until convergence

$$\Theta_j := \Theta_j - \alpha \frac{\partial}{\partial \Theta_j} \mathcal{L}(\Theta)$$

learning rate

partial derivative

parameter update

Learning NN parameters via gradient descent: Backpropagation

- How to calculate partial derivatives loss $\mathcal{L}$ with respect to Θ_j, i.e. all weights and biases?

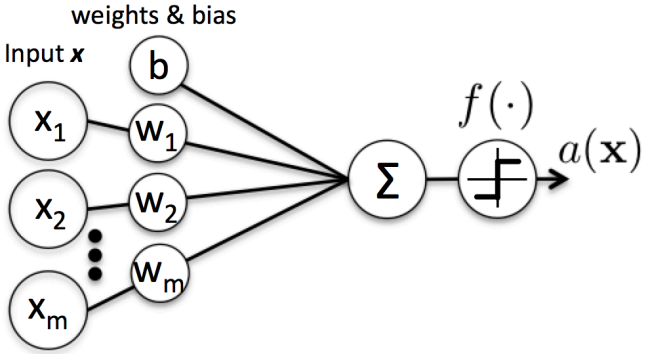
$$\frac{\partial}{\partial \Theta_j} \mathcal{L}(\Theta) = ?$$

- Solution: Use backpropagation (Rumelhart 1986)

- For more details check:
  - http://neuralnetworksanddeeplearning.com/
  - http://cs231n.stanford.edu/syllabus.html

Perceptron: Summary

Neuron activation: $a(\mathbf{x}) = f\left(\sum_i w_i x_i + b\right)$

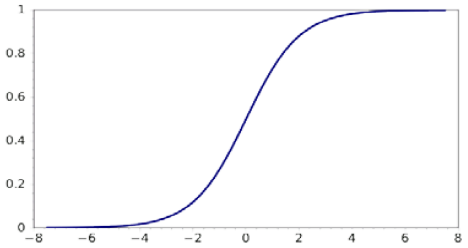


Loss: $\mathcal{L} = -\frac{1}{N}\sum_i^N (a(x_i) - y_i)^2$

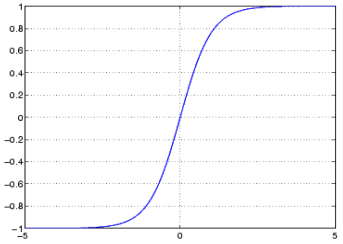
Learning: $(\mathbf{w}, b) = \min_{(\mathbf{w},b)} \mathcal{L}(\mathbf{w}, b, \mathbf{X}, \mathbf{Y})$ via gradient descent: $w_i^{(t+1)} = w_i^{(t)} + \frac{\partial \mathcal{L}}{\partial w_i^{(t)}}$

Activation functions:

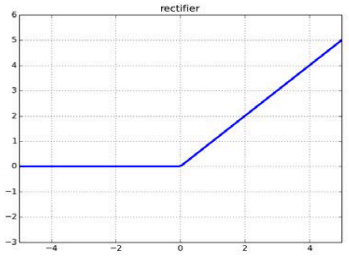
$S(x) = \frac{1}{1+e^{-x}}$



$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$



$f(x) = \max(0, x)$



Deep Learning

Automatically learn data-driven, hierarchical representations, to accomplish a task.

From Neocognitron [Fukushima 1980]:

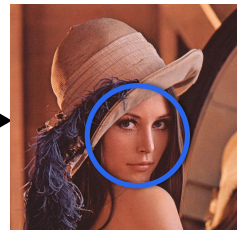
- **Learns data-driven features itself**: “If a set of stimulus patterns (input data) are repeatedly presented to it, it gradually acquires the ability to recognize these patterns.”

- **Do not engineer for one task**, adapt by learning: “It is not necessary to give any instructions about the categories to which the stimulus patterns should belong.”

Traditional machine learning pipeline

Features

Classifier

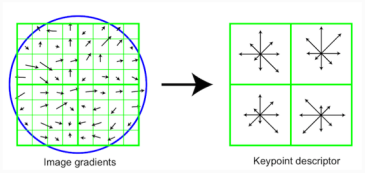


Calculate features $\mathbf{z}$ from image $\mathbf{x}$

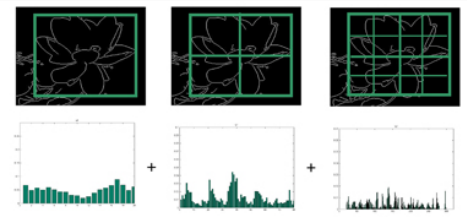
$h_\Theta(\mathbf{z})$

R	G	B	…	R	G	B
R	G	B	…	R	G	B
R	G	B	…	R	G	B
R	G	B	…	R	G	B
R	G	B	…	R	G	B

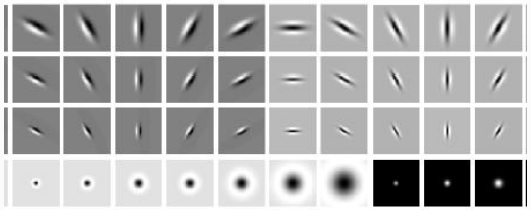
Pixel values



Sift

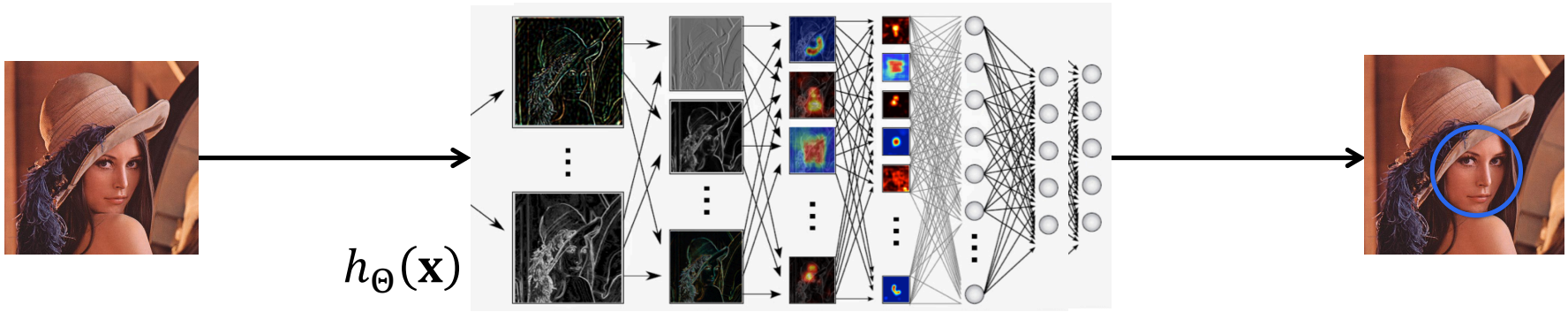


HoG

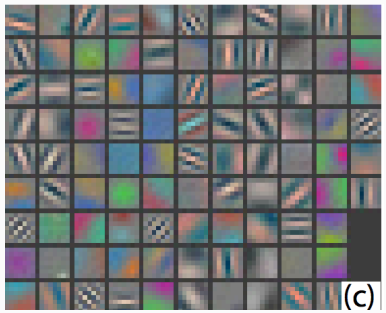


Texton

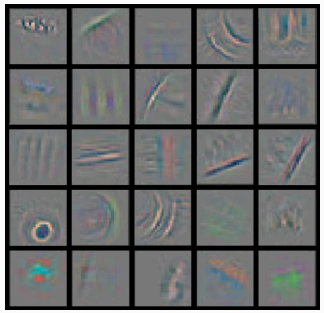
Deep learning pipeline



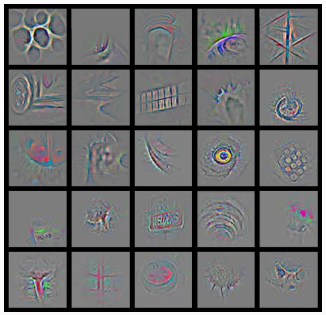
Hierarchical Representation



Low-level

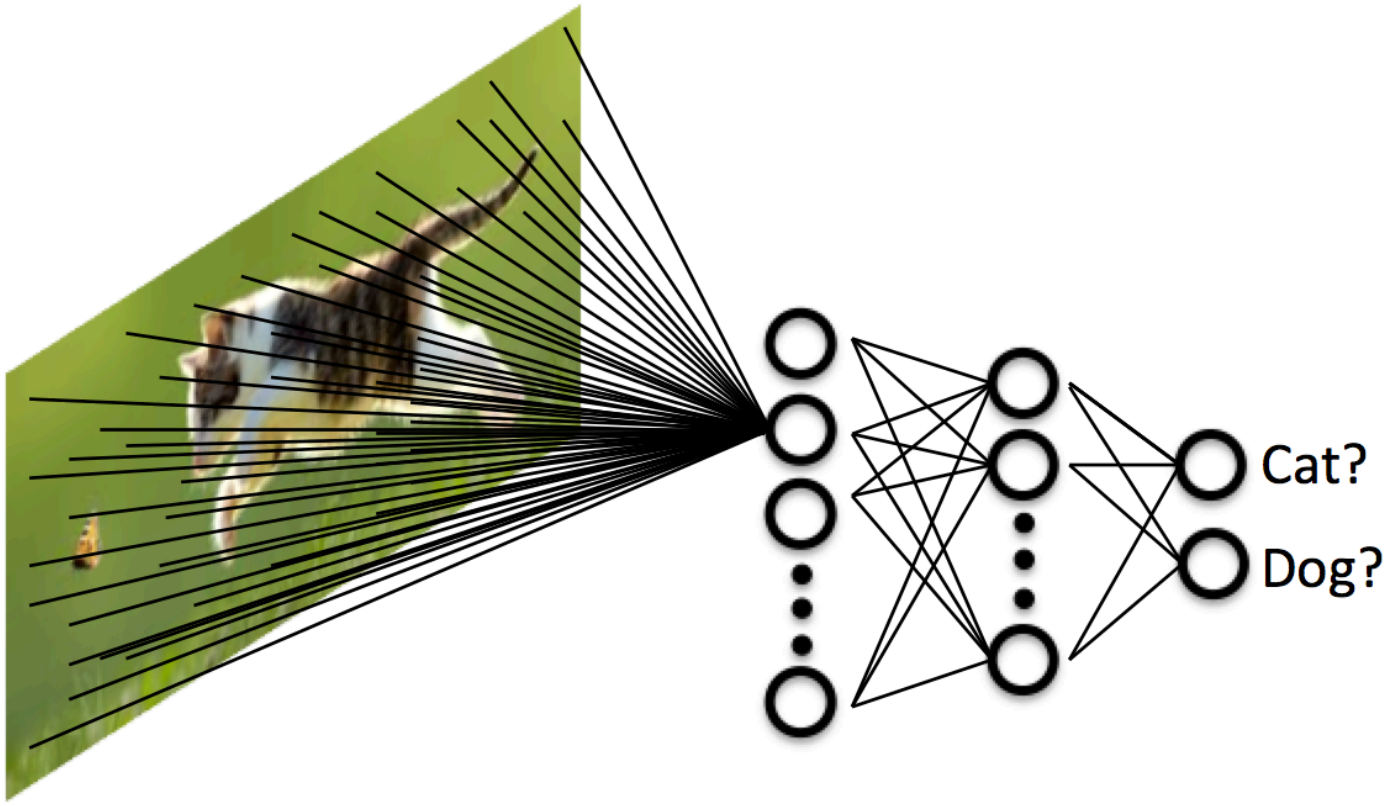


Mid-level

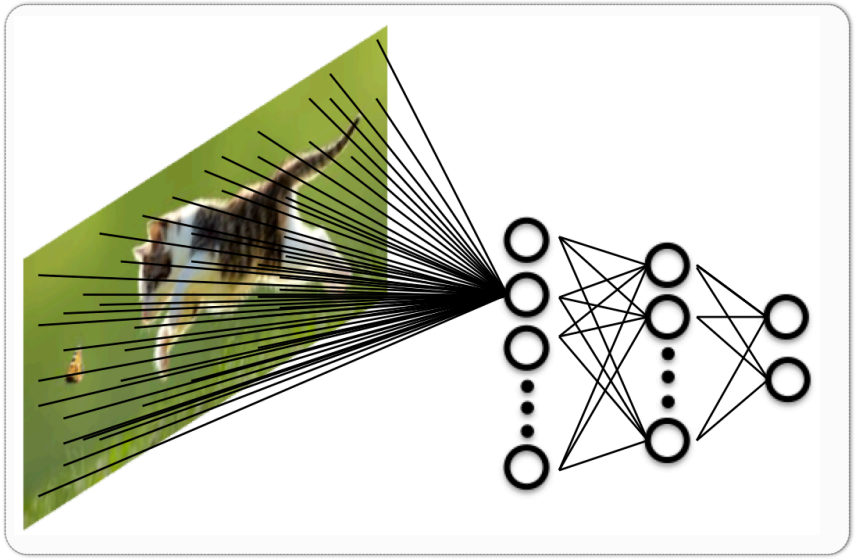


High-level

Multi-layer Perceptron



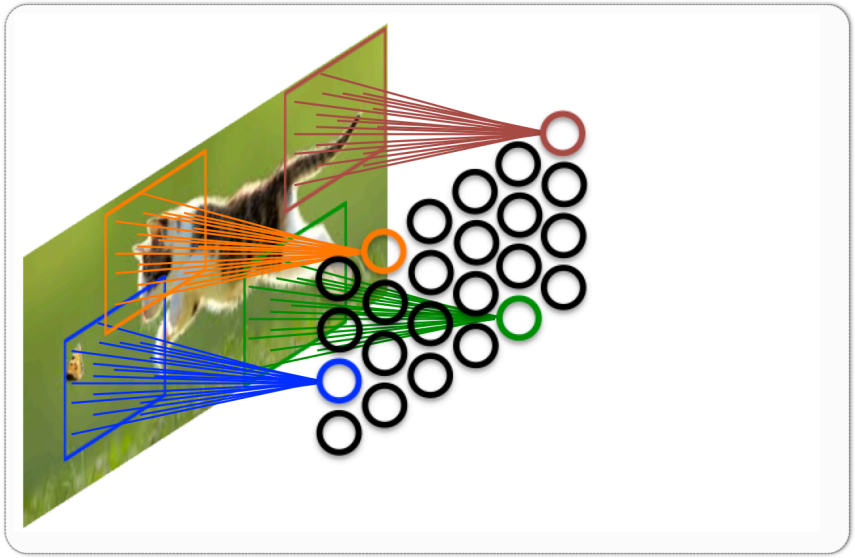
Fully Connected Networks



Features are local: Only neighboring pixels are correlated!

- Each neuron detects one feature/pattern.
- Too many weights for each neuron/pattern.
  - Example: 100x100 image = 10000 weights/pattern.

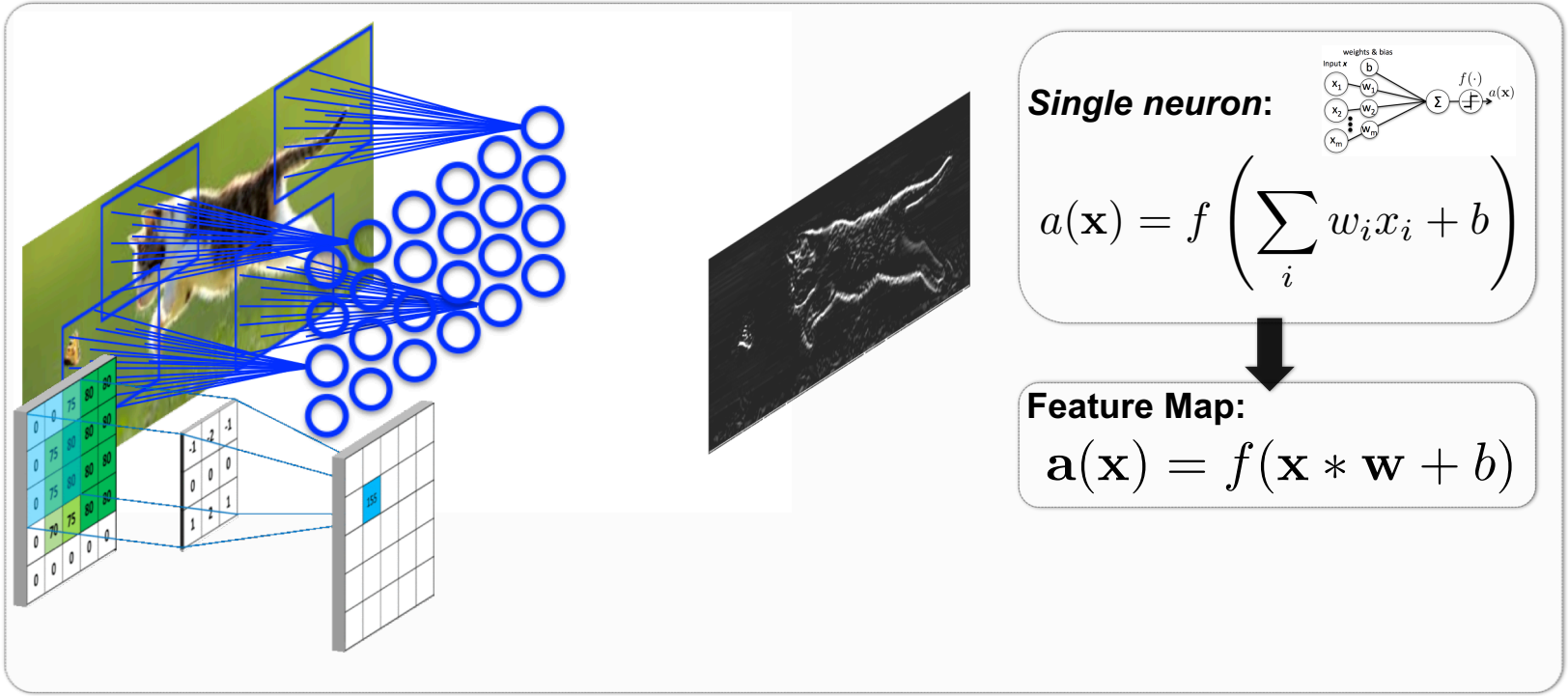
Locally Connected Networks



Stationarity in vision:
The same feature may appear anywhere!

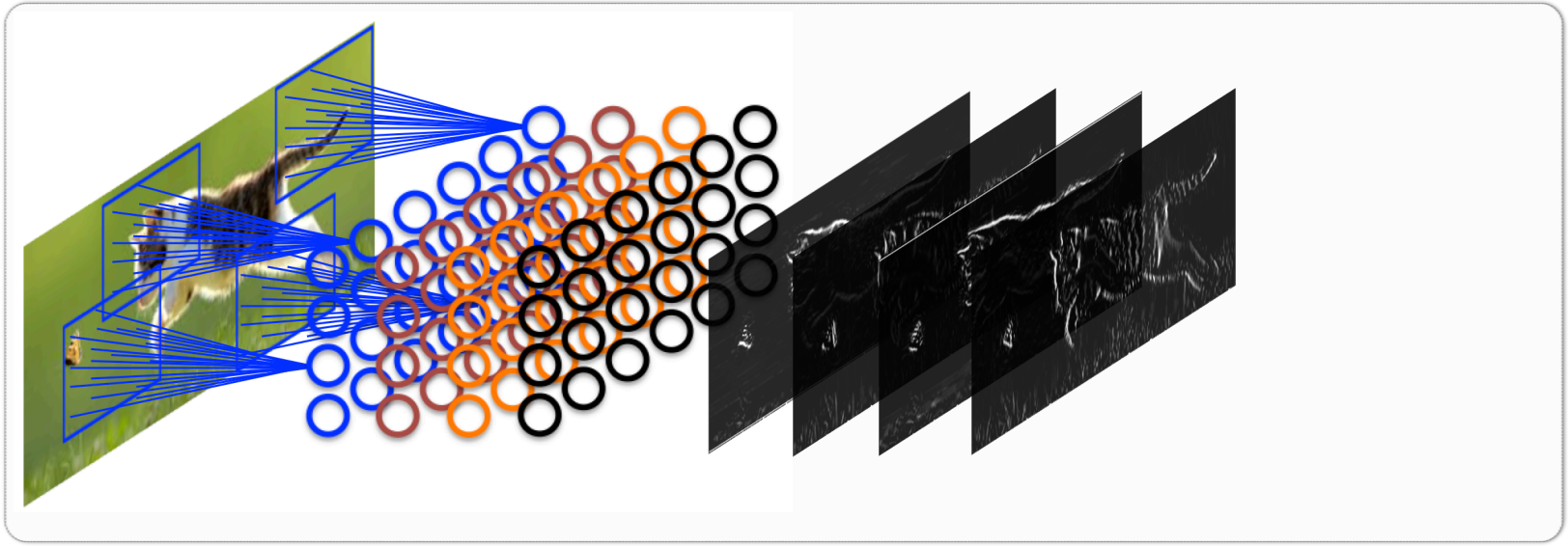
- Each neuron detects a different pattern.
- Kernel of a neuron: Spatially limited connections.
  - Example: If kernel = 10x10, 100 weights/pattern.
  - Each neuron only detects a pattern at a certain location.

Weight Sharing – Feature Maps



- Feature Map (FM), aka Channel:
  - Multiple neurons that share the same kernel (weights).
  - Activations of a FM's neurons computed with convolution with FM's kernel!

Convolutional Layer

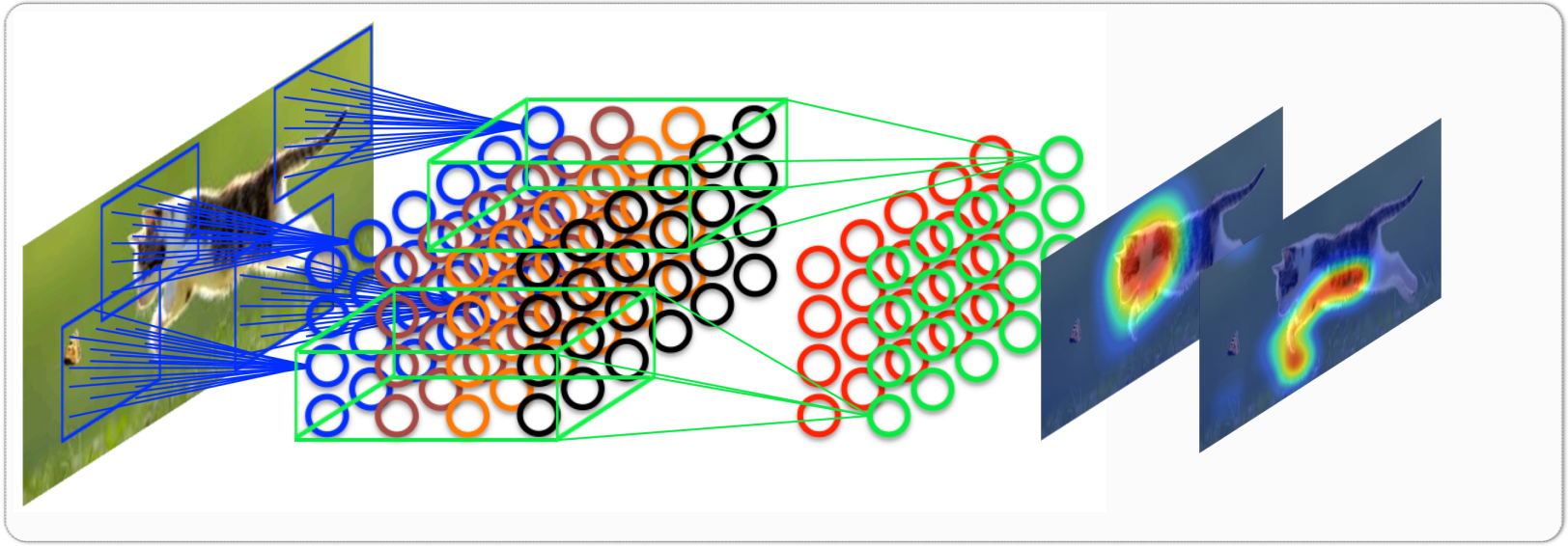


Convolutional Layer:

– Multiple Feature Maps – Detect multiple features per layer.

– A layer can be perceived as a multi-channel image (similar to RGB channels).

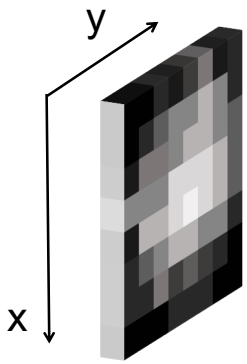
Multiple Convolutional Layers



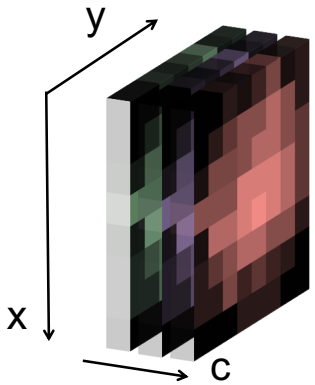
Convolutional Layer:

- Multiple Feature Maps – Detect multiple features per layer.
- A layer can be perceived as a multi-channel image (similar to RGB channels).
- Deeper layers process FMs (channels) of previous layer.
- Combine previous features to extract more complex representations.

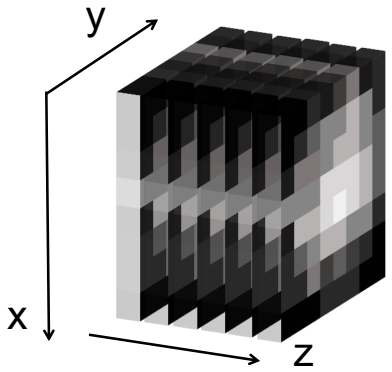
Input images



2D image, 1 channel
2D tensor (x,y)
Eg: black & white image

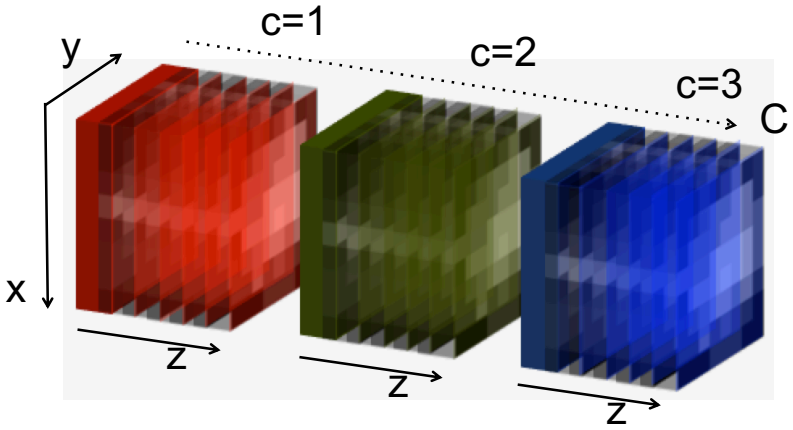


2D image, 3 channels
3D tensor (x,y,c)
Eg: RGB image

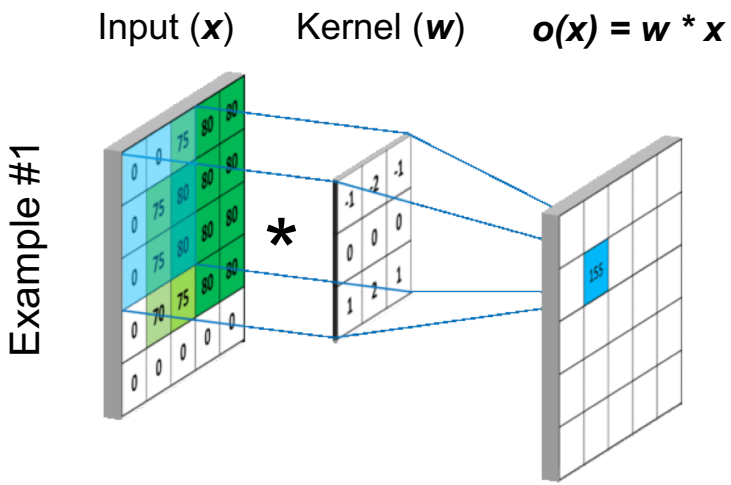


3D image, 1 channel
3D tensor (x,y,z)
Eg: CT volume

3D image, 3 channels
<u>4D</u> tensor (x,y,z,c)
Eg: T1, T2 and PD MRI volumes



Convolution



Feature Map:

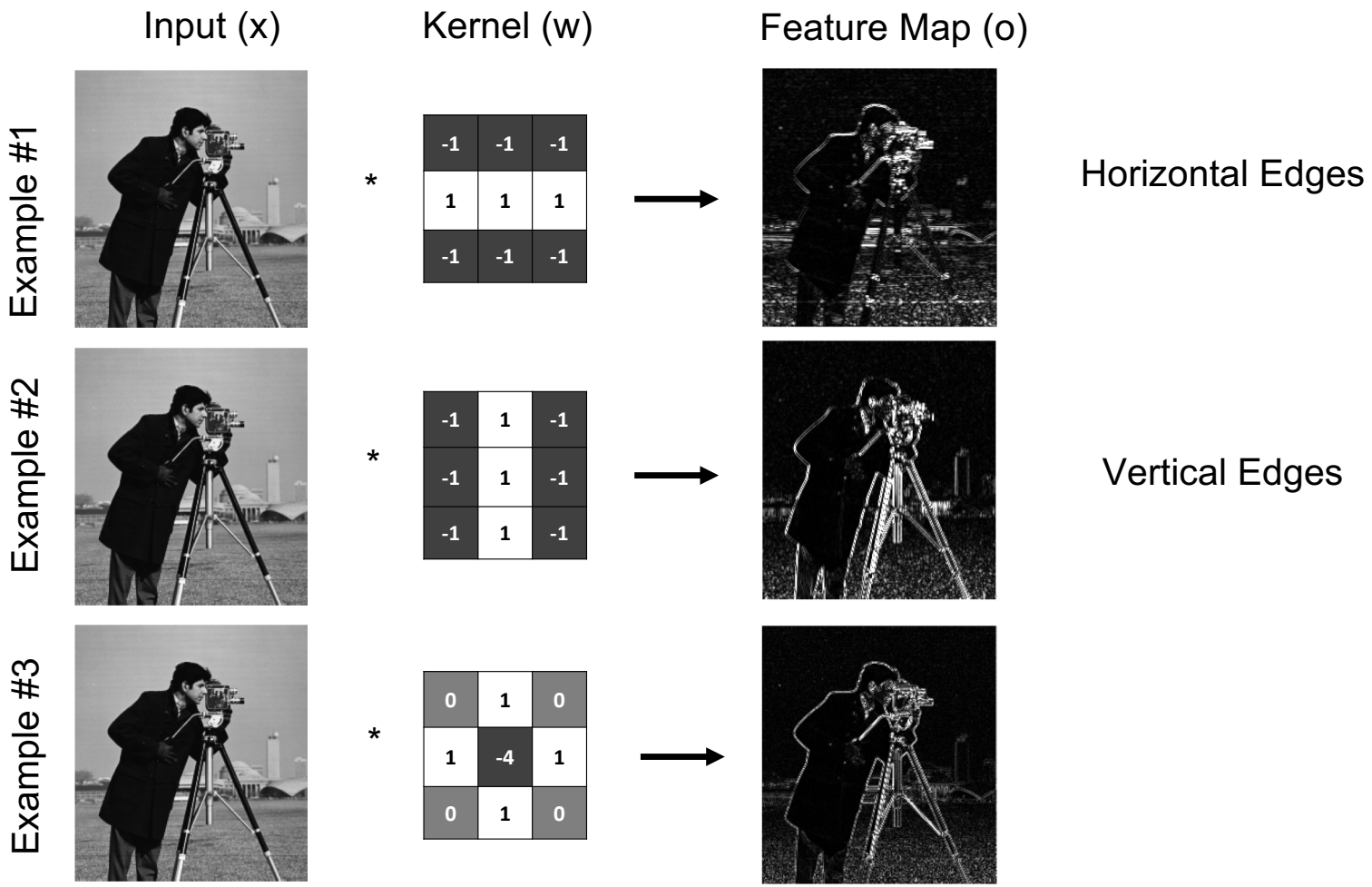
$$\mathbf{a}(\mathbf{x}) = f(\mathbf{x} * \mathbf{w} + b) = f(\mathbf{o}(\mathbf{x}) + b)$$

$$\mathbf{o}(\mathbf{x}) = \mathbf{x} * \mathbf{w}$$

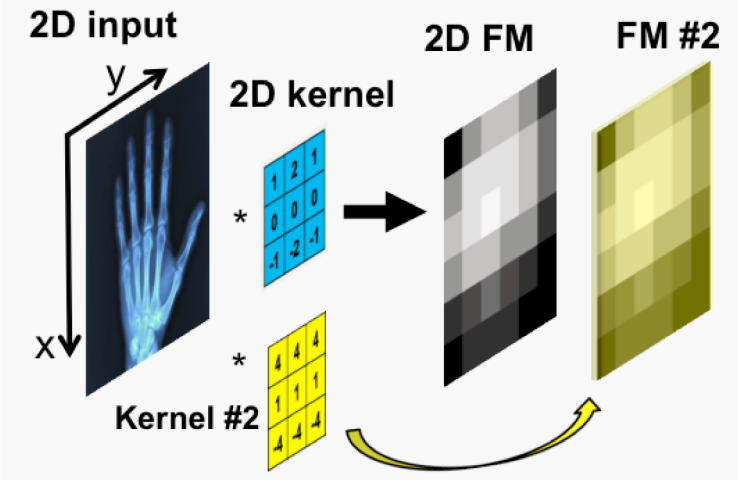
If 2D kernel ***w*** of dimensions $K^{(x)}$ x $K^{(y)}$:

$$\mathbf{o}[i,j] = \sum_{u=1}^{K^{(x)}} \sum_{v=1}^{K^{(y)}} \mathbf{x}[i+u, j+v] \cdot \mathbf{w}[u,v]$$

Convolution



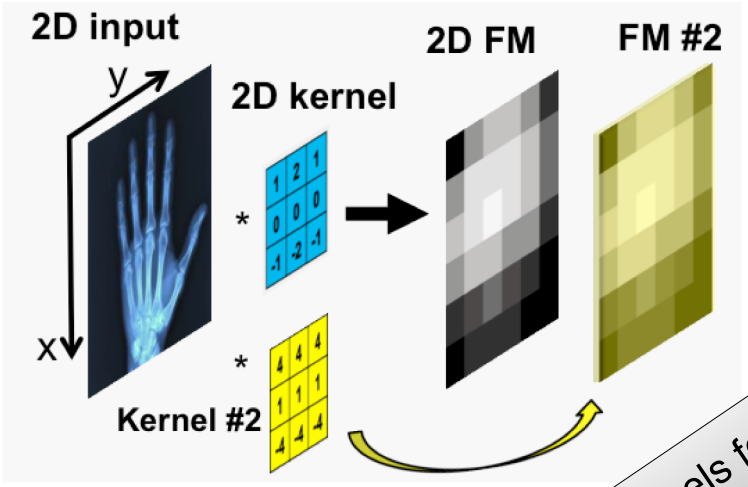
2D images



Each feature map:

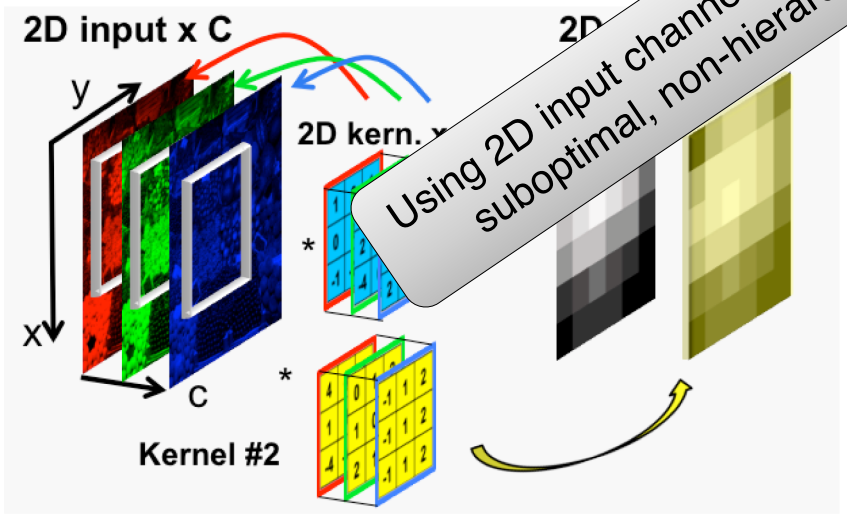
$$\mathbf{o}[i,j] = \sum_{u=1}^{K^{(x)}} \sum_{v=1}^{K^{(y)}} \mathbf{x}[i+u, j+v] \cdot \mathbf{w}[u,v]$$

2D images



Each feature map:

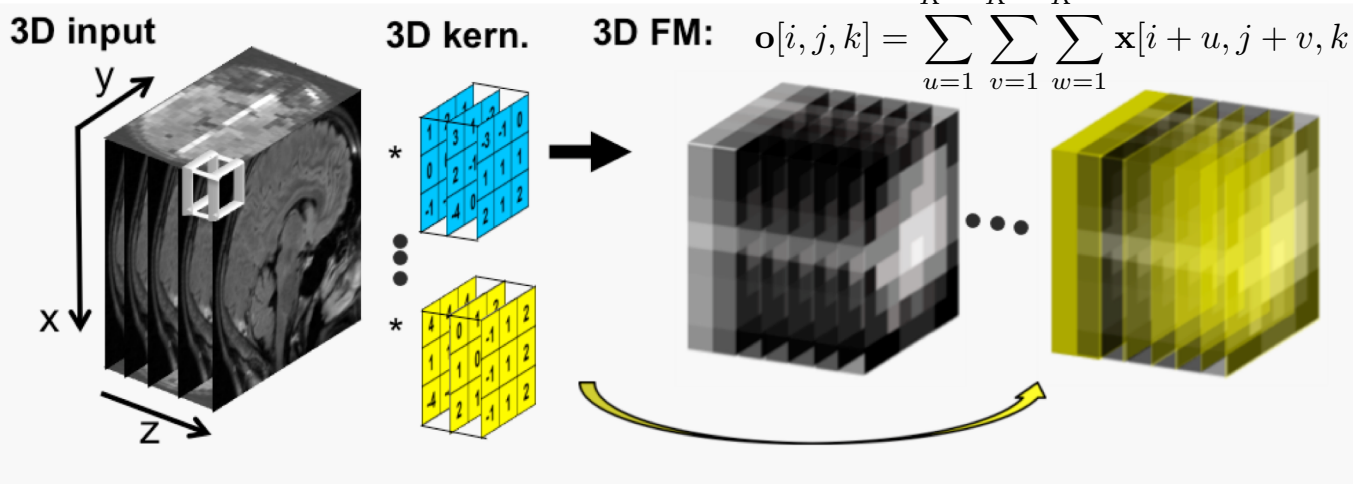
$$\mathbf{o}[i,j] = \sum_{u=1}^{K^{(x)}} \sum_{v=1}^{K^{(y)}} \mathbf{x}[i+u, j+v] \cdot \mathbf{w}[u,v]$$



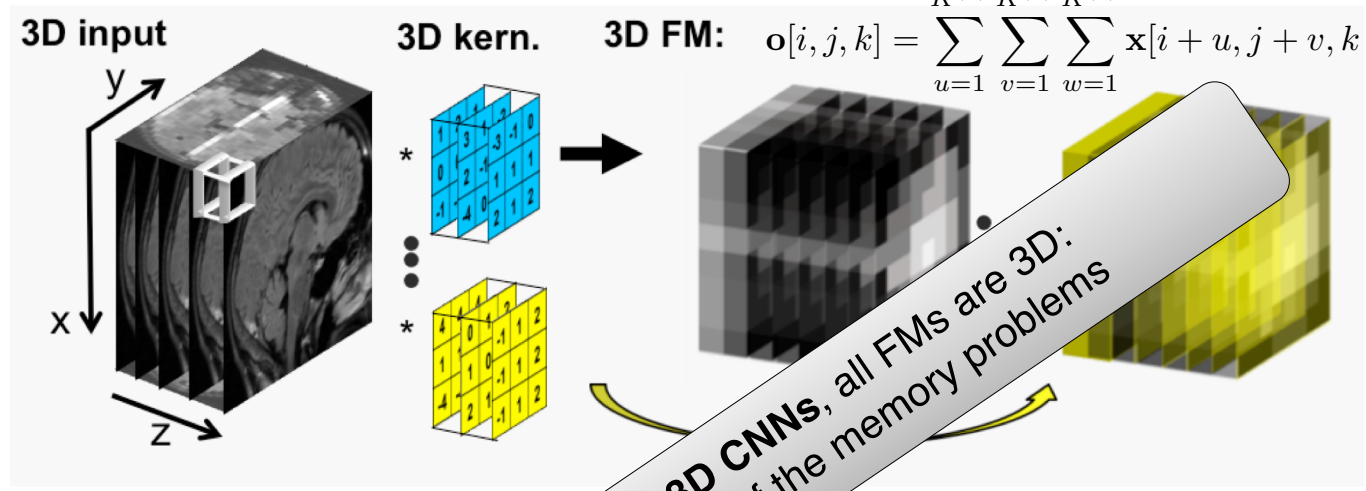
Using 2D input channels for different z-slices (2.5D), is suboptimal, non-hierarchical spatial processing

$$\mathbf{o}[i,j] = \sum_{c=1}^{C} \sum_{u=1}^{K^{(x)}} \sum_{v=1}^{K^{(y)}} \mathbf{x}[i+u, j+v, c] \cdot \mathbf{w}[u,v,c]$$

3D images

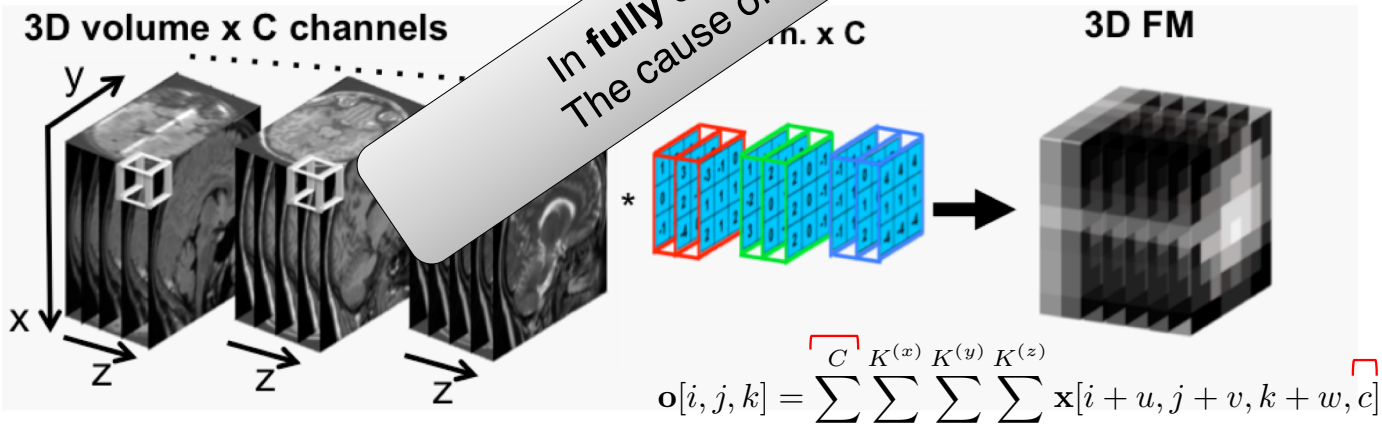

3D input
y
x
z
3D kern.
*
*
3D FM:
$\mathbf{o}[i,j,k] = \sum_{u=1}^{K^{(x)}} \sum_{v=1}^{K^{(y)}} \sum_{w=1}^{K^{(z)}} \mathbf{x}[i+u, j+v, k+w] \cdot \mathbf{w}[u,v,w]$

3D images



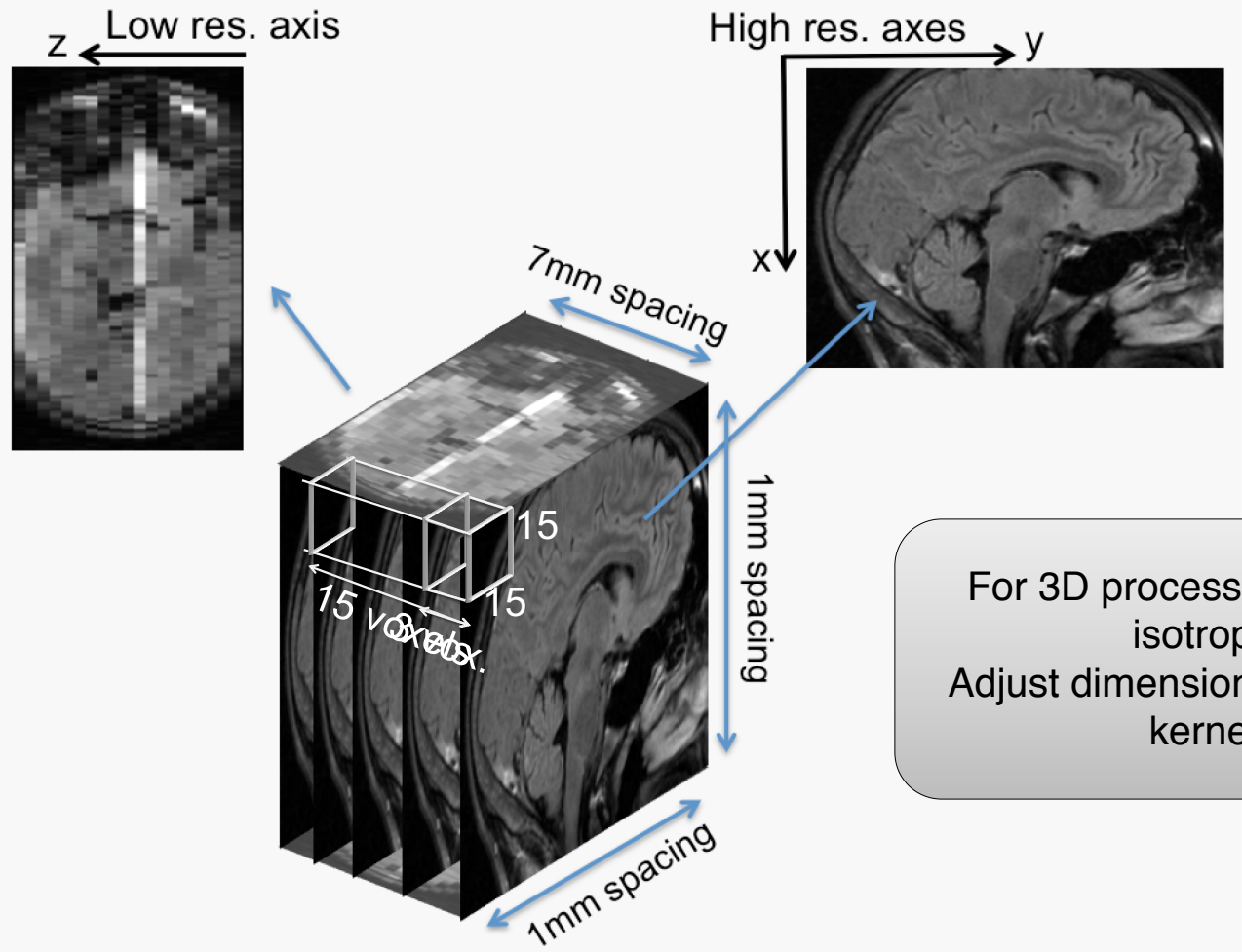
$$\mathbf{o}[i,j,k] = \sum_{u=1}^{K^{(x)}} \sum_{v=1}^{K^{(y)}} \sum_{w=1}^{K^{(z)}} \mathbf{x}[i+u, j+v, k+w] \cdot \mathbf{w}[u,v,w]$$

In **fully 3D CNNs**, all FMs are 3D:
The cause of the memory problems



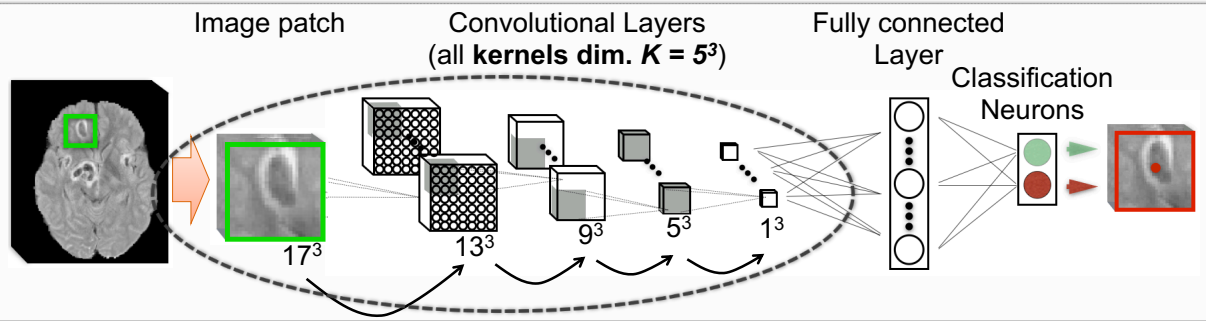
$$\mathbf{o}[i,j,k] = \sum_{c=1}^{C} \sum_{u=1}^{K^{(x)}} \sum_{v=1}^{K^{(y)}} \sum_{w=1}^{K^{(z)}} \mathbf{x}[i+u, j+v, k+w, c] \cdot \mathbf{w}[u,v,w,c]$$

Anisotropic image resolution

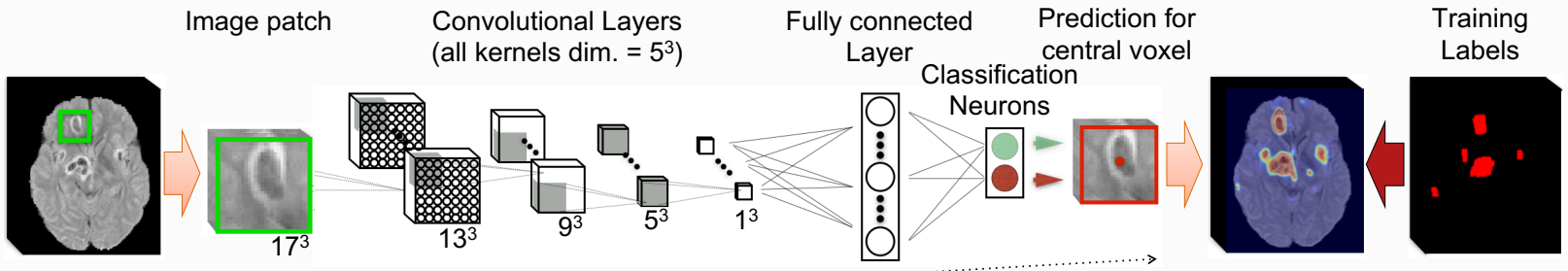


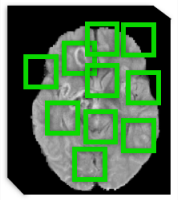
For 3D processing: Resample volumes to isotropic resolution OR
Adjust dimensions of kernels. Square/cubic kernels not required!

Segmentation as a classification problem



Segmentation as a classification problem

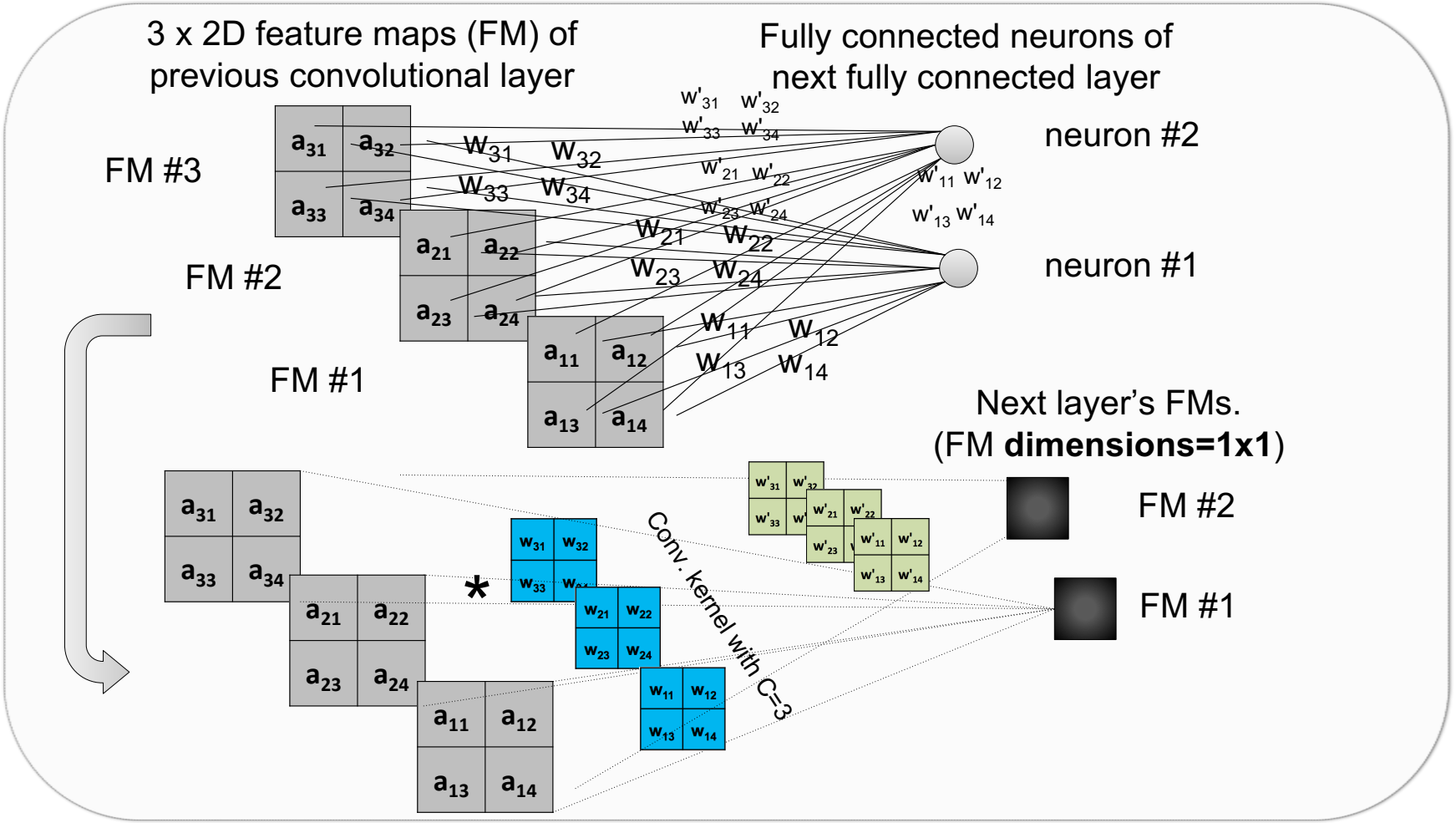




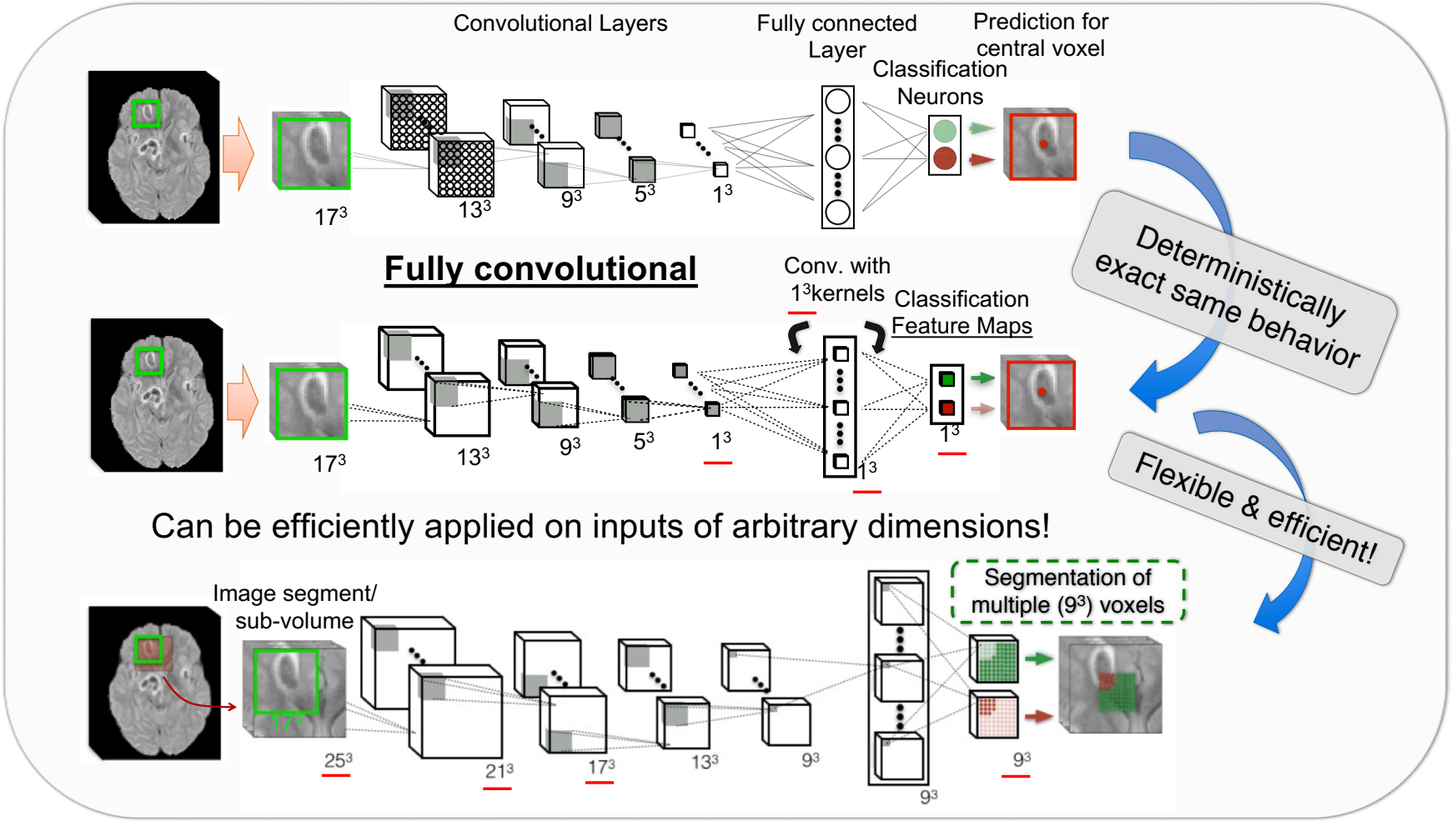
Overlapping patches:

- Redundant
- (re-convolve)
- Inefficient memory usage
  (context reloaded, e.g. in same batch)

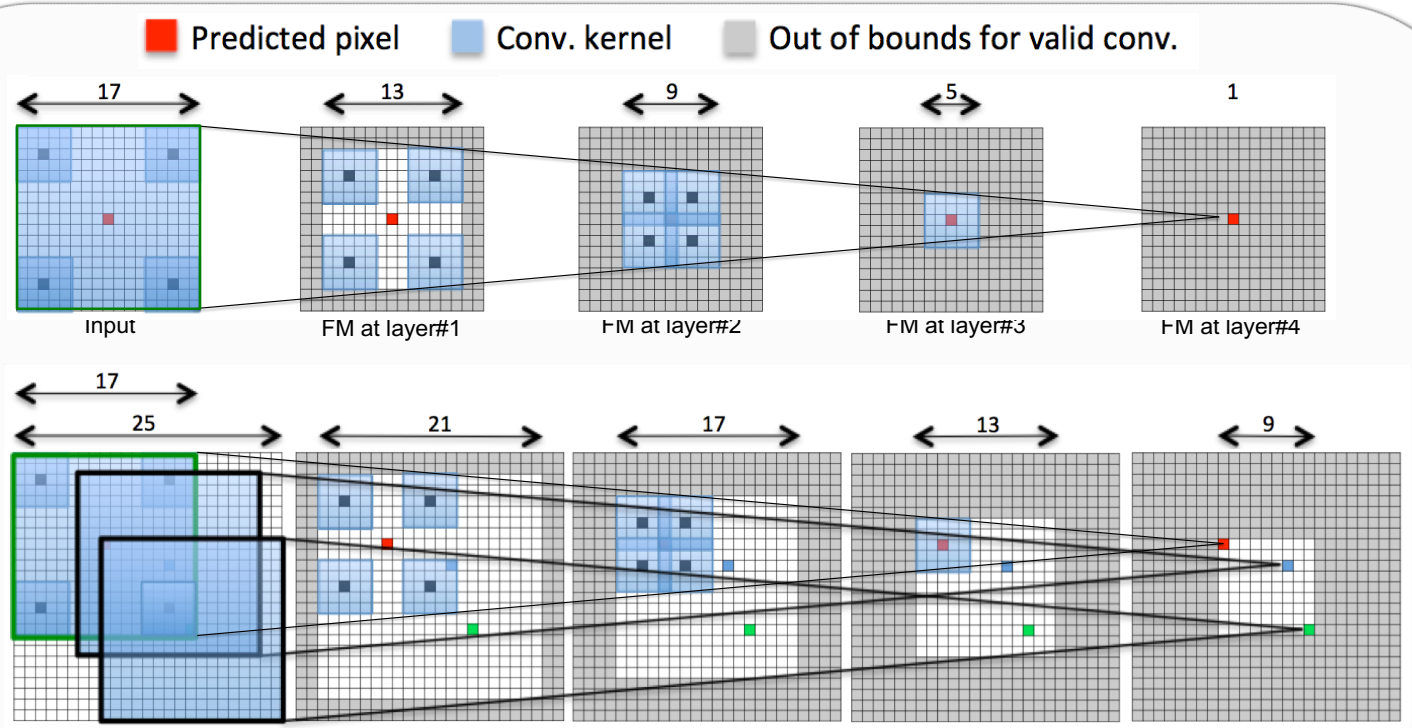
Changing fully connected layers to convolutional layers



CNNs: Fully convolutional networks



CNNs: Fully convolutional networks: Size of feature maps expands with input



- **Size of receptive field** remains the same!
- Each prediction is only influenced by its own receptive field. NOT the whole input!

CNNs: Fully convolutional networks

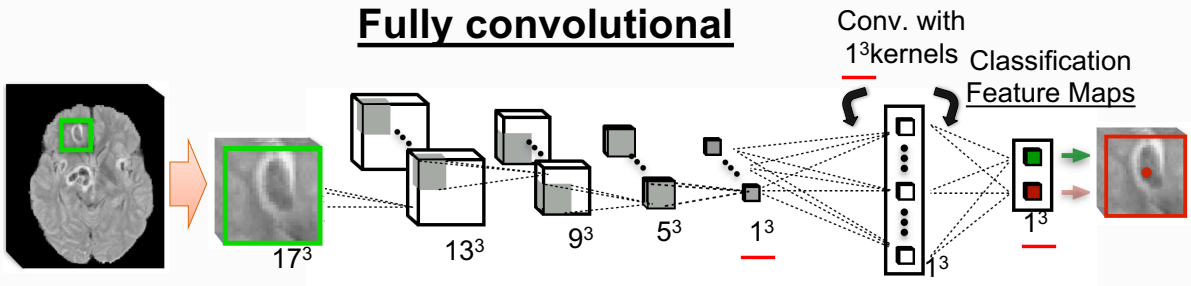
Fully convolutional implementation:

i. Dimensions of all FMs scale with input dim.

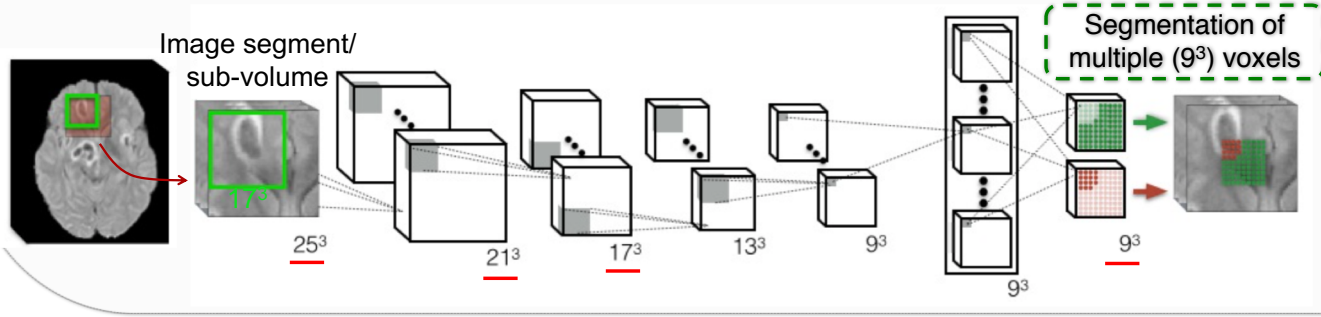
ii. Including Classification FMs!

iii. Efficiency: Densely-predict multiple voxels in one pass, avoiding computational redundancy on overlapping context!

Fully convolutional

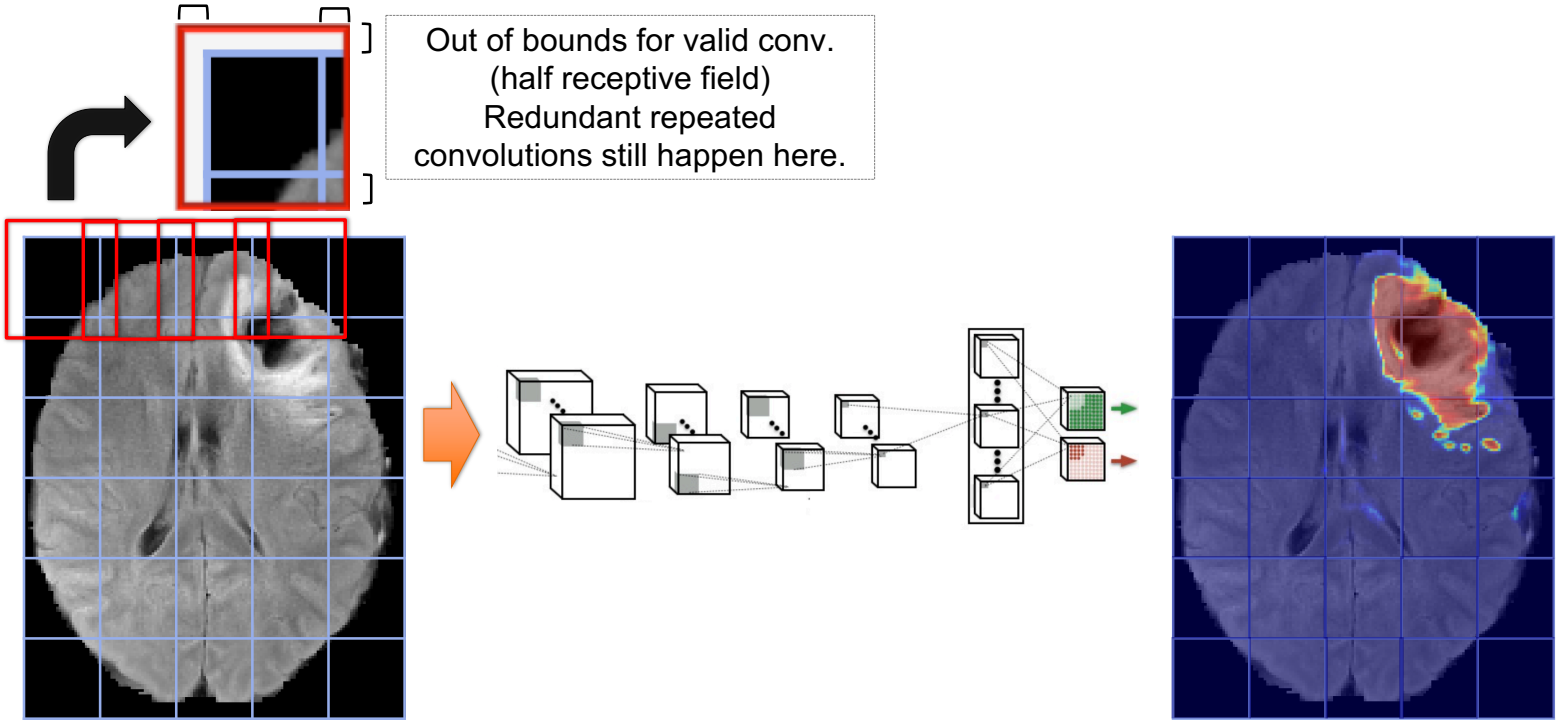


Can be efficiently applied on inputs of arbitrary dimensions!



Flexible & efficient!

Tiling an image for segmentation



- Bigger input tiles (segments):
  - Require more memory (because the size of all feature maps increases).
  - Less computational redundancy because of less overlap in between.
  - Tiles at testing time do not need to be the same size as at training.

Training a fully convolutional net

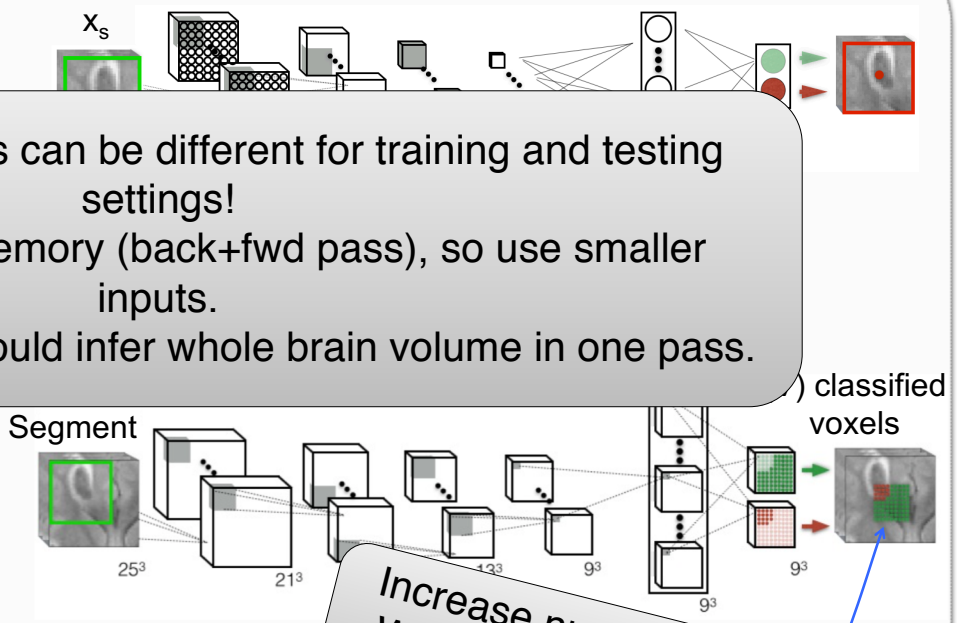
Training with single predictions on each sample:

Learn parameters $\theta = \min_{\theta} \mathcal{L}$

$\mathcal{L} = -$

Cross entr
x_s a trainin
$f_c(x)$ the C
[.] the indica

Size of input segments can be different for training and testing settings!
Train: Takes more memory (back+fwd pass), so use smaller inputs.
Test: Only fwd pass. Could infer whole brain volume in one pass.



Training on dense predictions:

Over the (B) segments (s) in a batch.

$$\mathcal{L} = -\frac{1}{B \cdot V} \sum_{s=1}^{B} \sum_{v=1}^{V} \sum_{c} [c = y_s^v] log(f_c(x_s^v; \theta))$$

Over the V classified voxels in a segment.

Increase number of samples per batch.
Without linear increase in computation
(e.g. if loading V separate patches).

Building CNNs for image segmentation

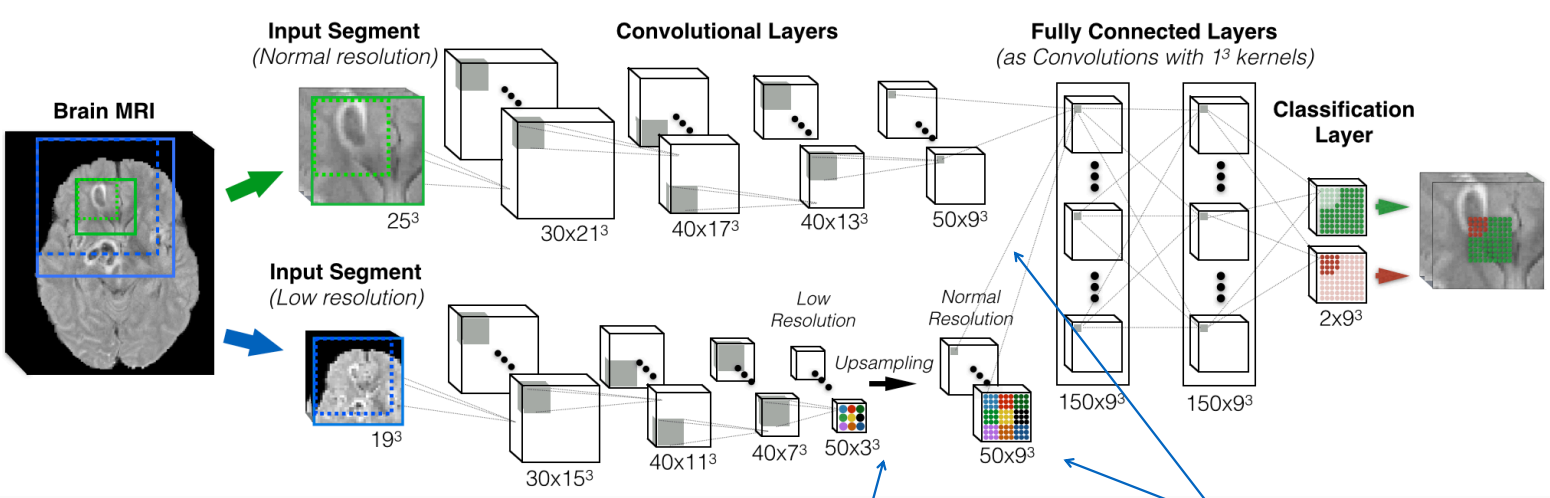
- How to process more context?

- Add more layers?
  - Receptive field increases linearly with each layer (by k-1).
  - Required memory increases linearly (almost) with number of conv layers.

- Process input at multiple scales.

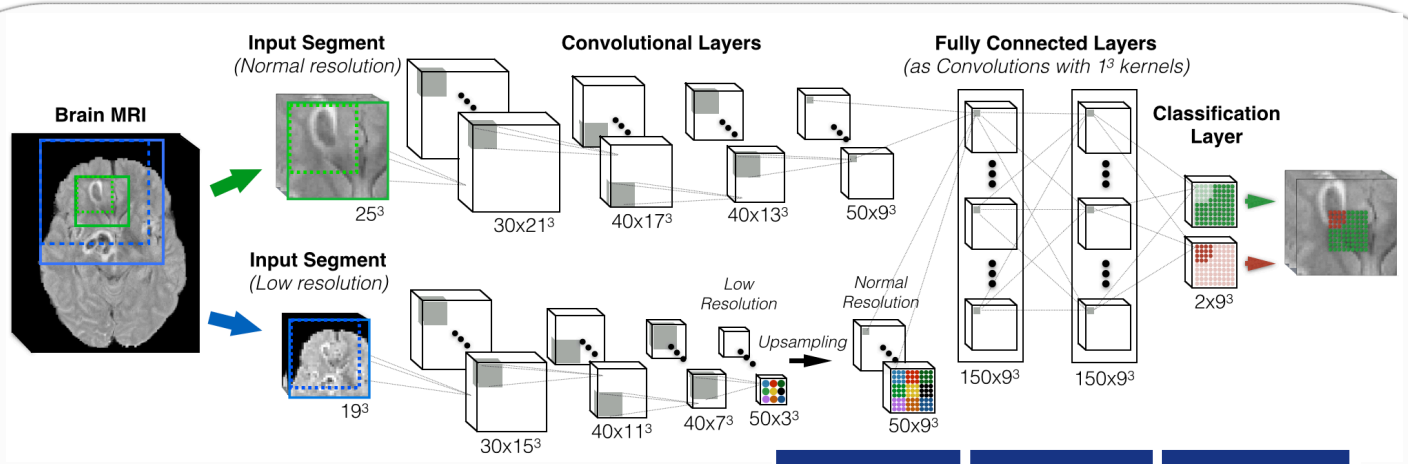
Building CNNs for image segmentation

- Centered at same location, extract context at normal and at lower resolution.
- Separate pathways process image at different resolutions (can be more than 2).
- Low resolution pathway processes **greater physical context** with the **same number of kernels**.
- The down-sampling factor (here x3) controls amount of context (*context de-coupled* from memory/computational requirements).
- Can adjust pathways separately. Commonly symmetrical.

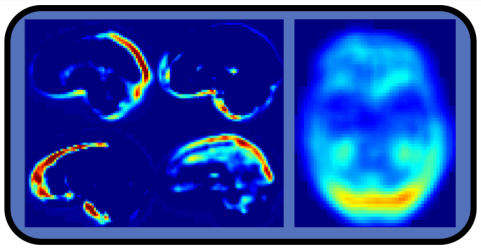


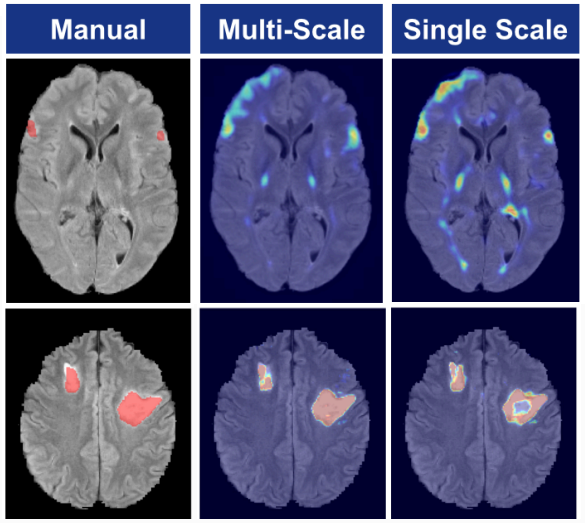
Kamnitsas et al, “Efficient multi-scale 3d CNN with fully connected CRF for accurate brain lesion segmentation”, MedIA, 2017.

Building CNNs for image segmentation



Additional spatial information enhances network's localization capabilities





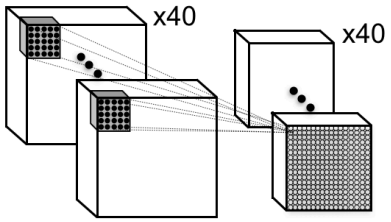
Building CNNs for image segmentation: Going deeper with smaller kernels

Deeper architectures can represent more complex functions.

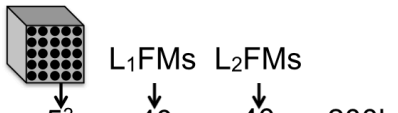
- Too many free parameters if we simply start adding more layers.

➢ Use only layers with small kernels [Simonyan et al, 2014].

Convolution with 5^3 kernel



Number of weights in layer l , with $C^{(l)}$ FMs,
that uses kernels of dimensions k^3: $k^3 \times C^{(l-1)} \times C^{(l)}$



➢ Total receptive field remains the same.

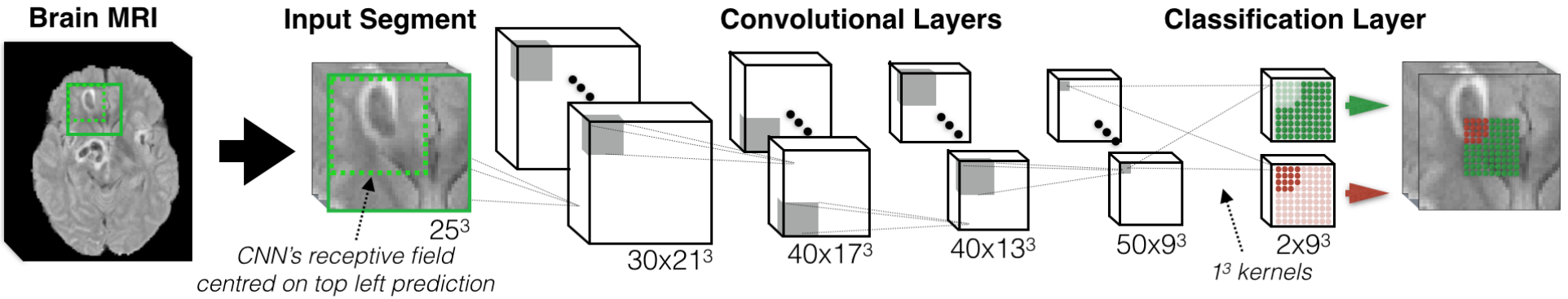
➢ Less total parameters, especially for 3D [Kamnitsas 2017]!

➢ They tend to work (much) better. Common nowadays.

DeepMedic: Overview

- Baseline CNN architecture:
  - Four layers with 5^3 kernels for feature extraction, leading to a receptive field of size 17^3.
  - The classification layer is implemented as convolutional with 1^3 kernels, which enables efficient dense inference.
  - Cross-entropy as loss function

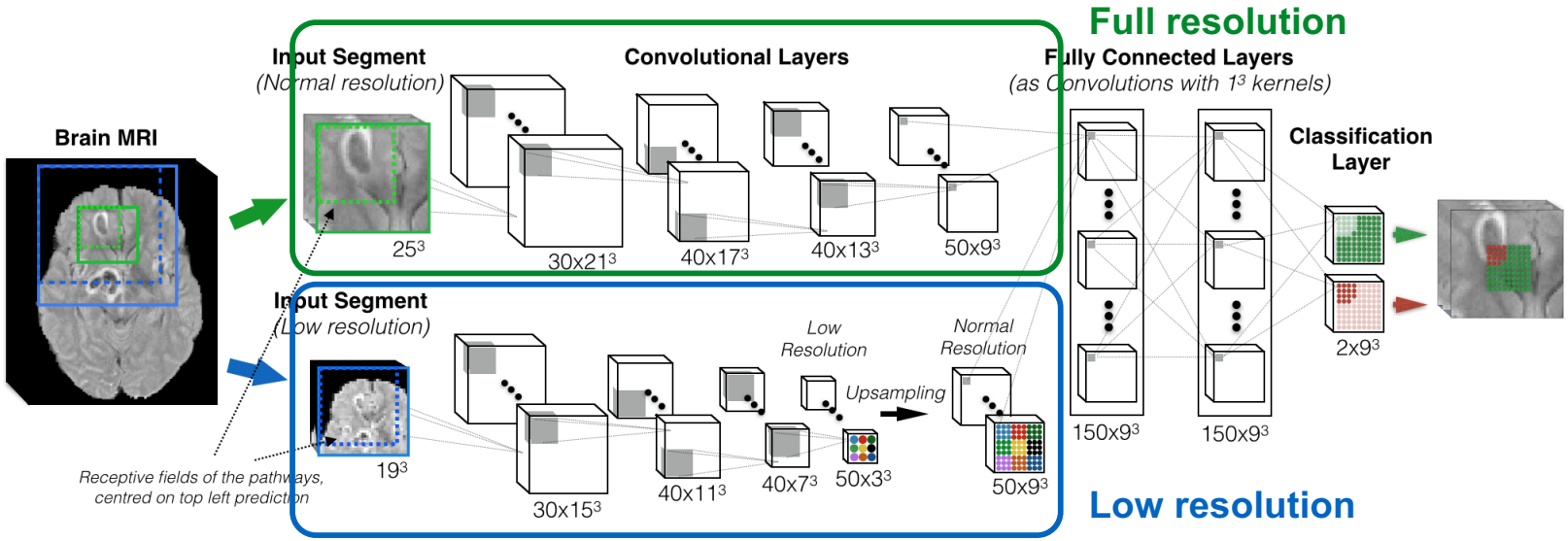
$$\mathcal{L}_{seg} = -\frac{1}{m}\sum_{i=1}^{m}[f(x_i) = y_i]log(f(x_i))$$



K. Kamnitsas et al. Medical Image Analysis, 2017

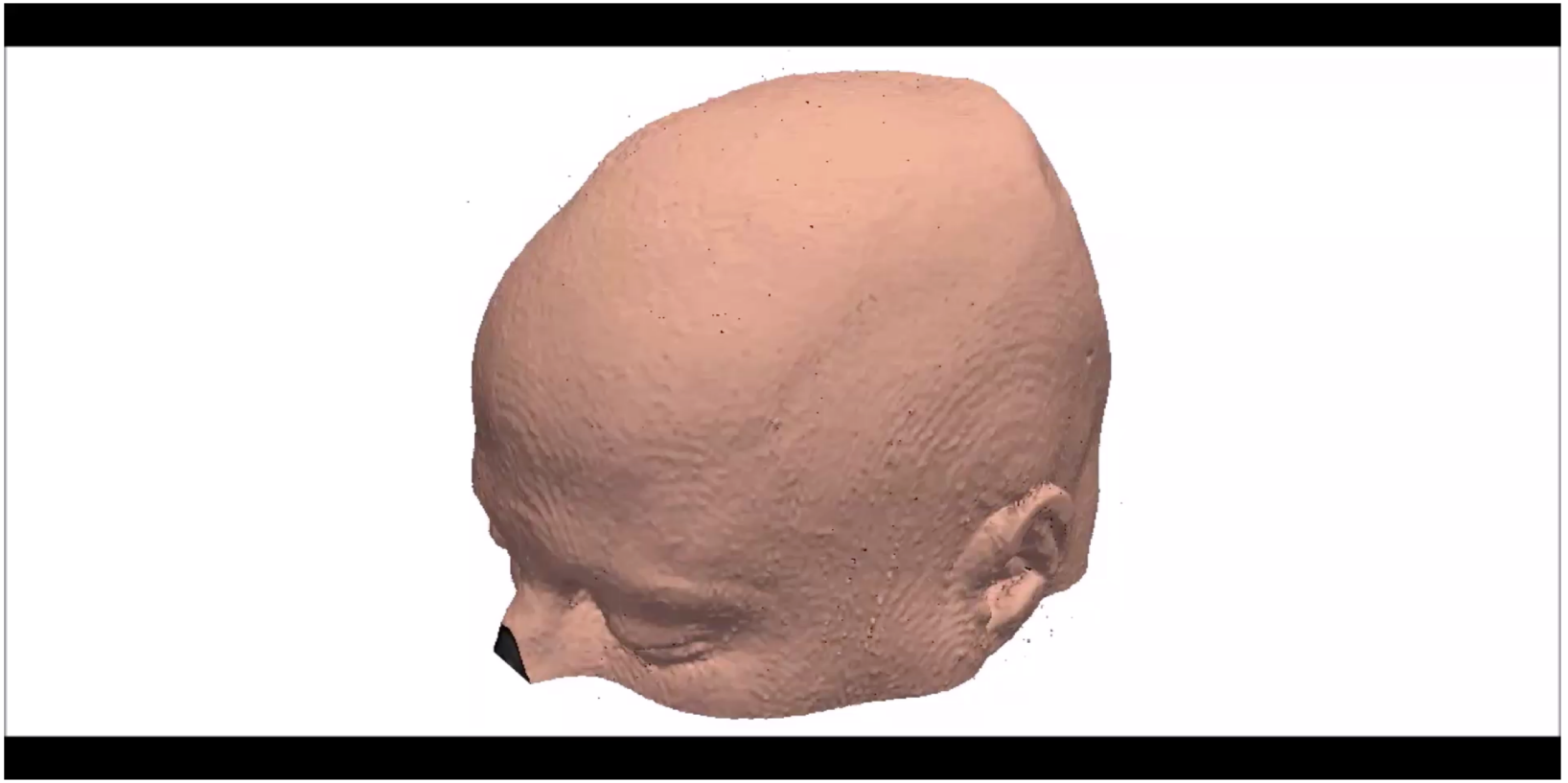
DeepMedic: Overview

- The size of receptive field of CNNs is important:
  - Large receptive increases computation and memory requirements
  - Pooling leads to loss of the spatial information
- Solution: Use multi-scale approach



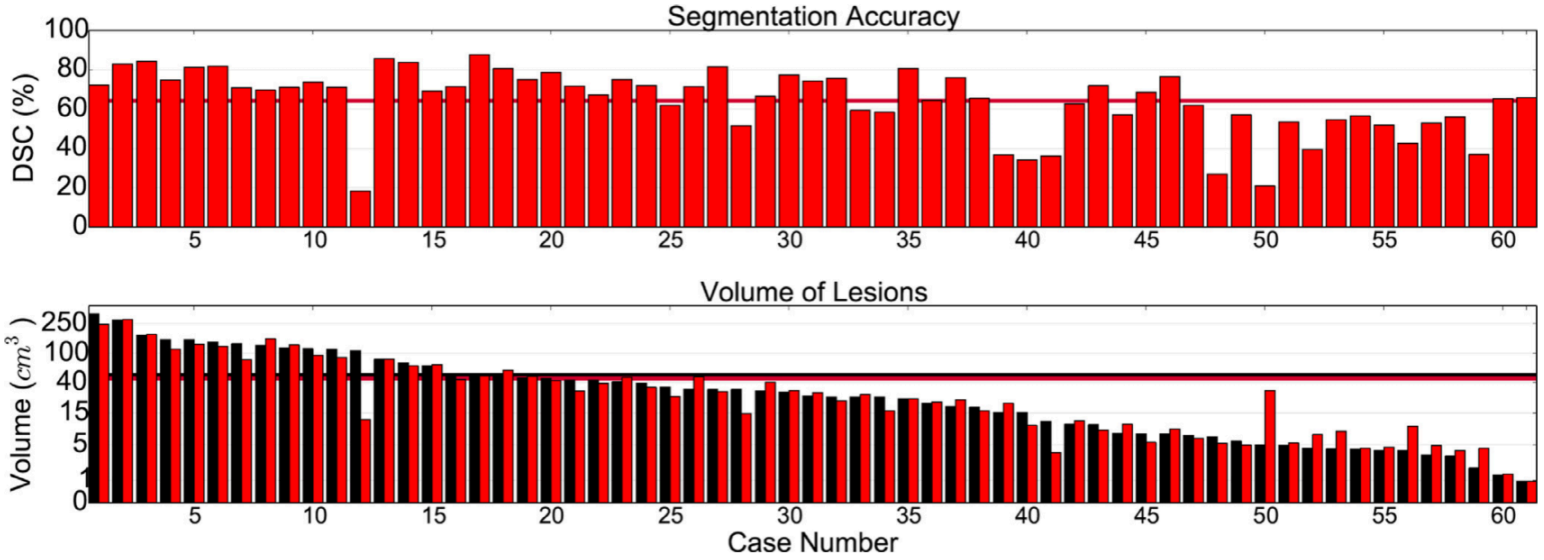
K. Kamnitsas et al. Medical Image Analysis, 2017

DeepMedic: Application to brain lesions



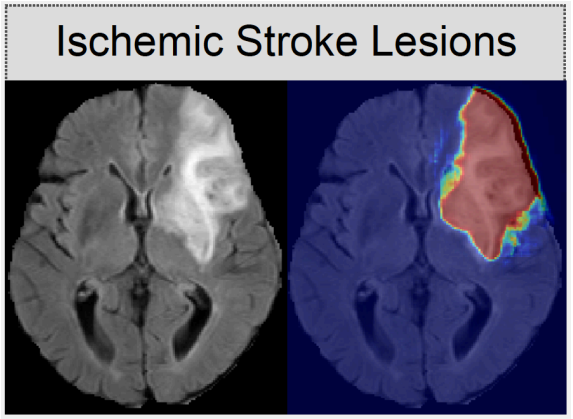
K. Kamnitsas et al. Medical Image Analysis, 2017

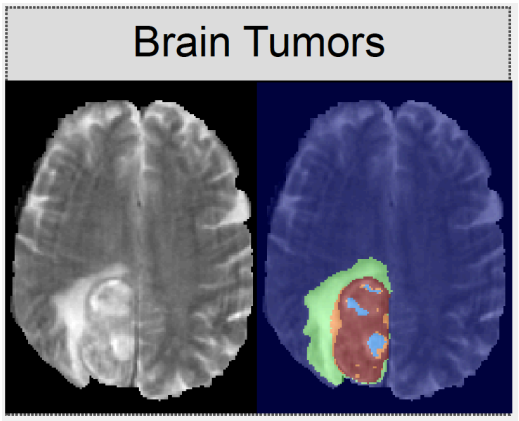
DeepMedic: Application to TBI

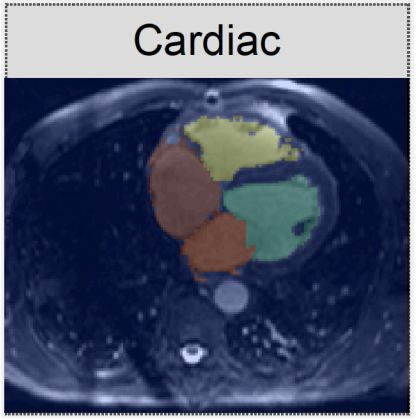


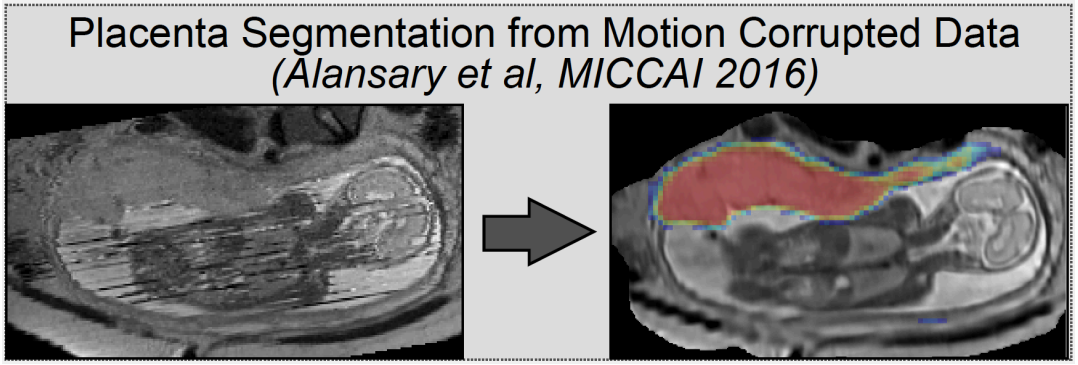
K. Kamnitsas et al. Medical Image Analysis, 2017

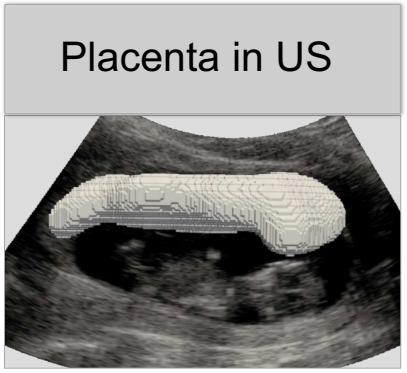
DeepMedic: Applications



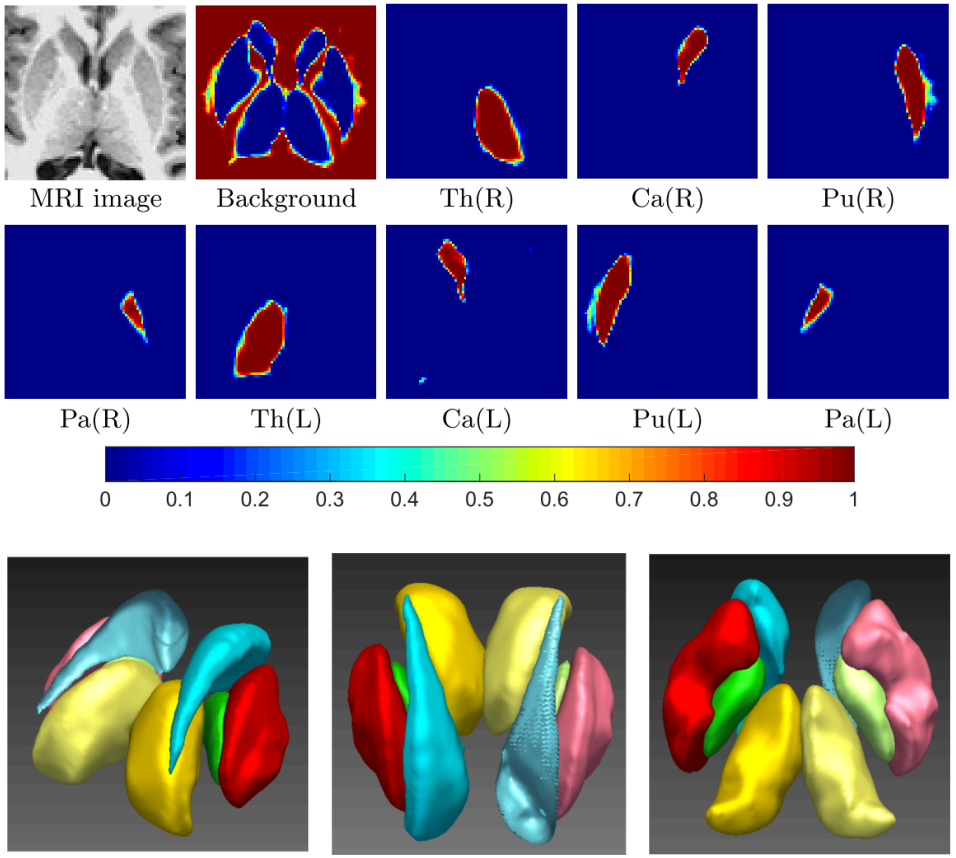




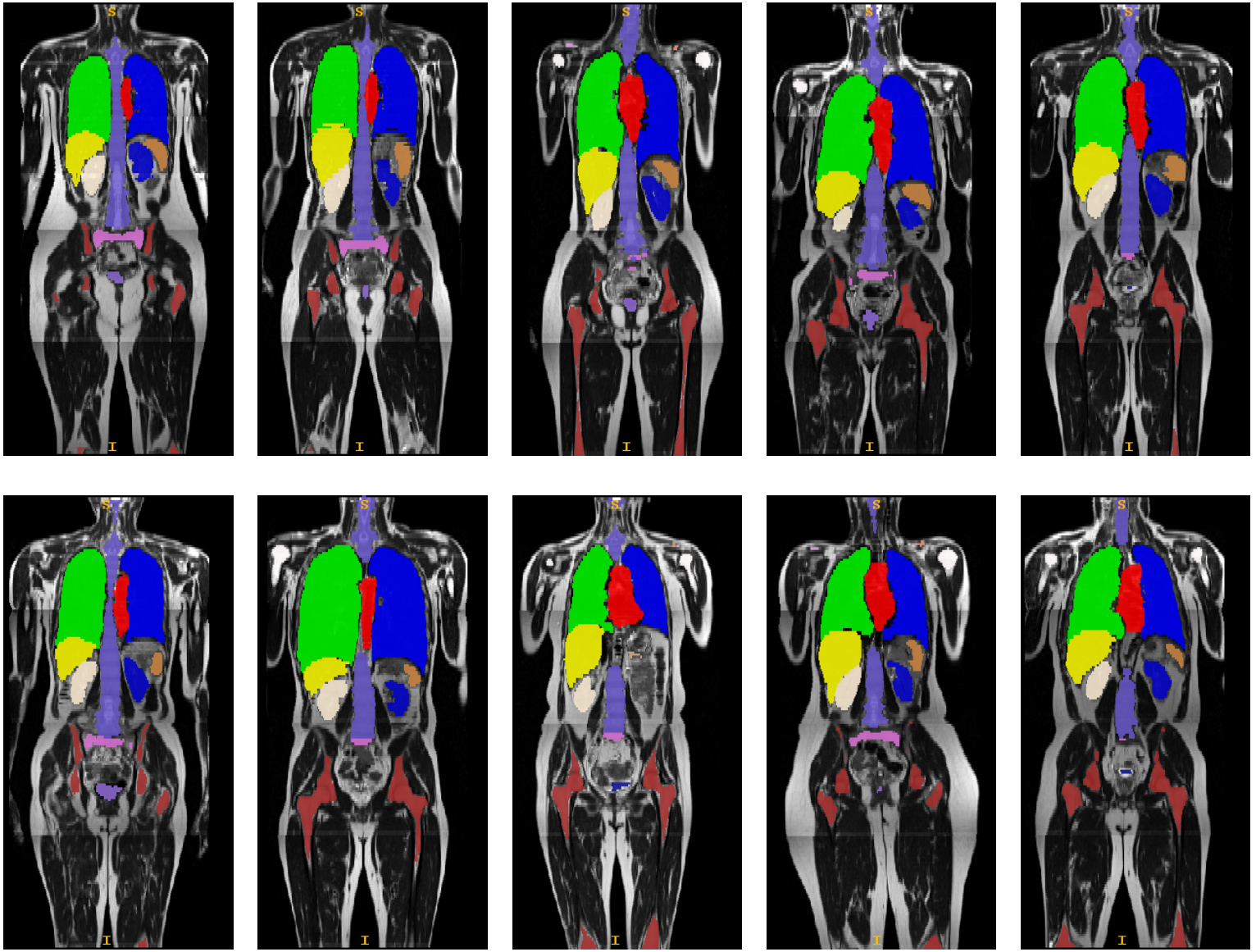




DeepMedic: Applications



Dolz et al, “3D fully convolutional networks for subcortical segmentation in MRI A large-scale study”, NeuroImage 2017


S
I

CNNs: Downsampling via Pooling

Pooling operator:

- Kernel of dimension ***k***
- Stride ***s*** of convolution
- Kernel function: max (mostly), mean…

➢ **No learnt parameters**.

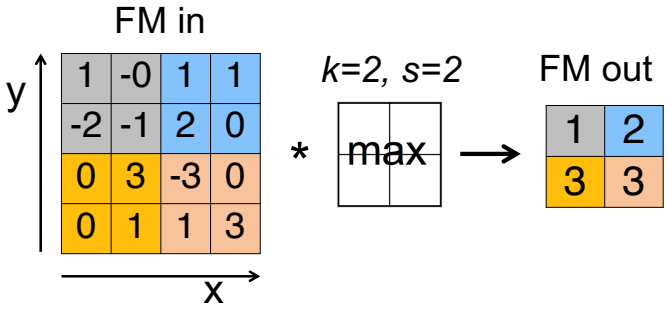
➢ Applied **individually on each FM**.

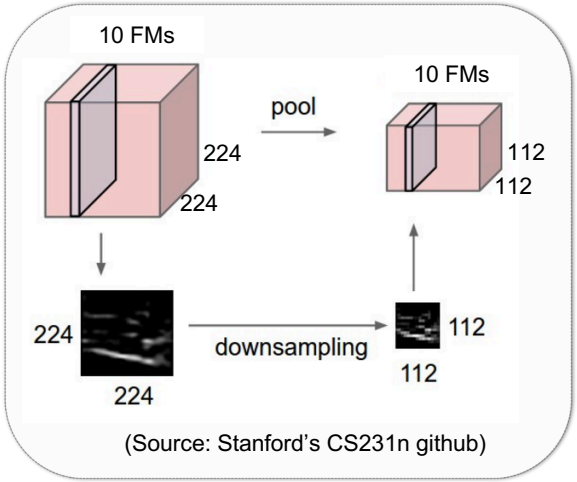
Motivation:

I. Reduce size (memory) of deeper layers

II. Invariance to small translations

III. Contraction forces network to learn high-level features



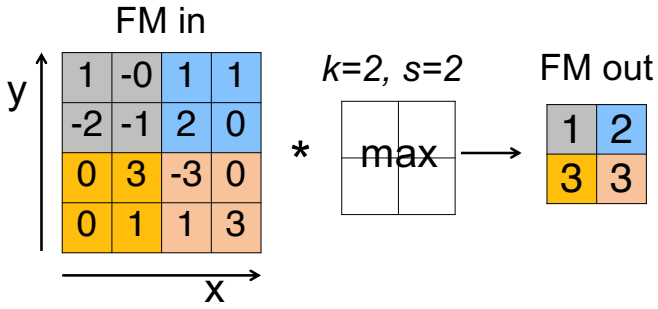


CNNs: Downsampling via strided convolution

Pooling operator:

- Kernel of dimension ***k***
- Stride ***s*** of convolution.
- Kernel function: max (mostly), mean…

How to down-sample

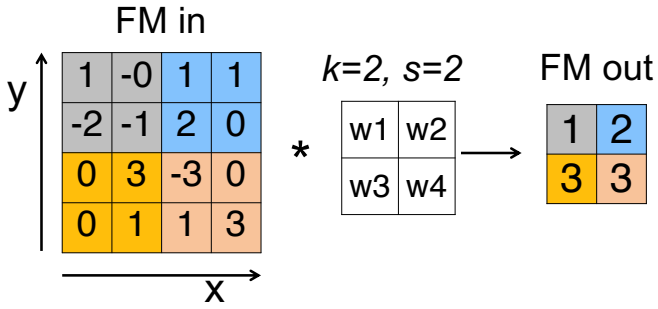


CNNs: Downsampling via strided convolution

Pooling operator:

- Kernel of dimension ***k***
- Stride ***s*** of convolution.
- Kernel function: max (mostly), mean…

How to down-sample → Just learn it!



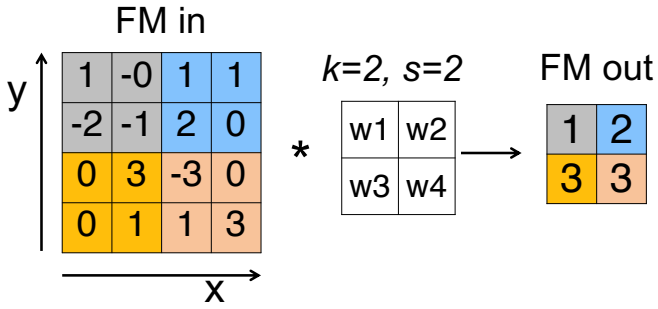
CNNs: Downsampling via strided convolution

Pooling operator:

- Kernel of dimension ***k***
- Stride ***s*** of convolution.
- Kernel function: max (mostly), mean…

How much to down-sample

How to down-sample → Just learn it!

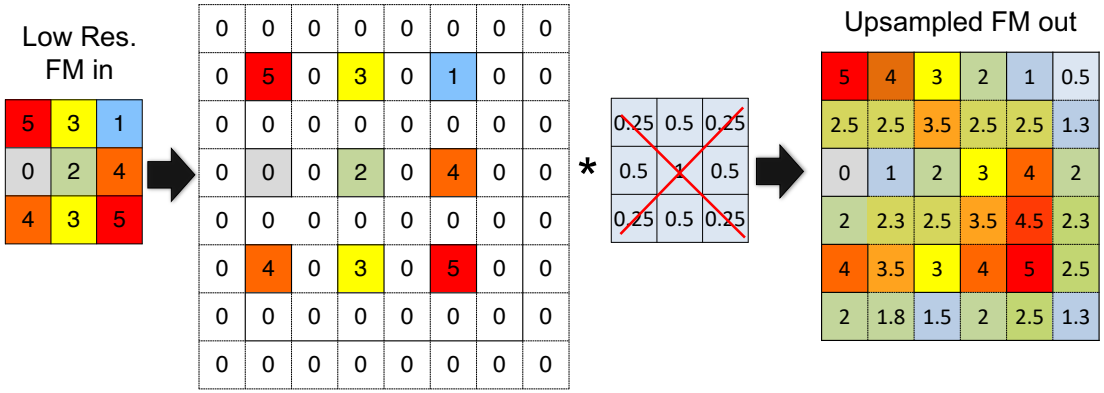


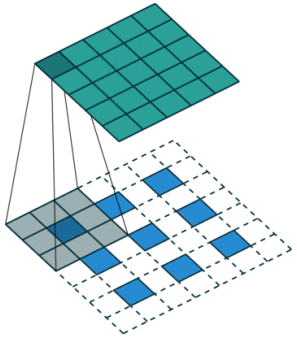
Strided convolution:

- Like normal convolutional (as before), but with a stride ***s*** != 1.
- Down-samples input by a factor of ***s***.
- Learn the kernel weights, so it “learns how to down-sample”.
- Often works better, commonly now replacing max-pooling.

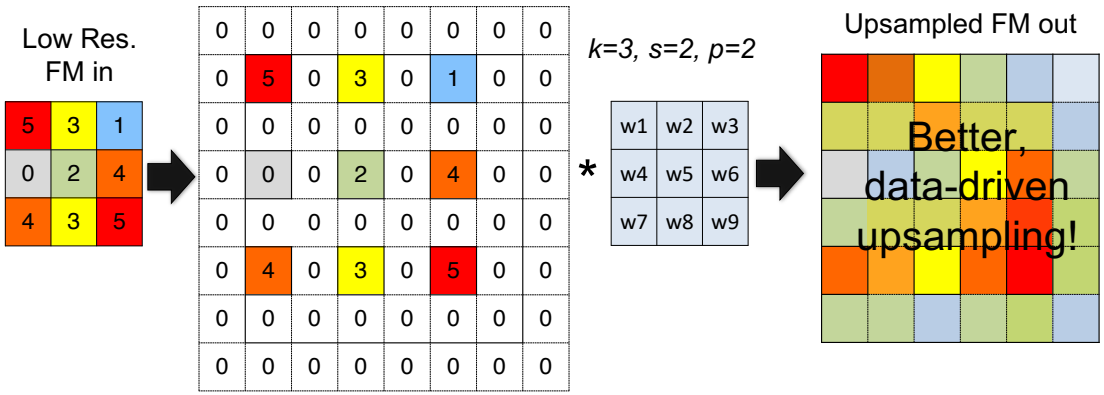
Springenberg et al, “Striving for simplicity: the all convolutional net”, ICLR, 2015

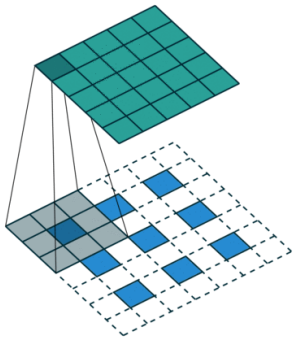
CNNs: Upsampling





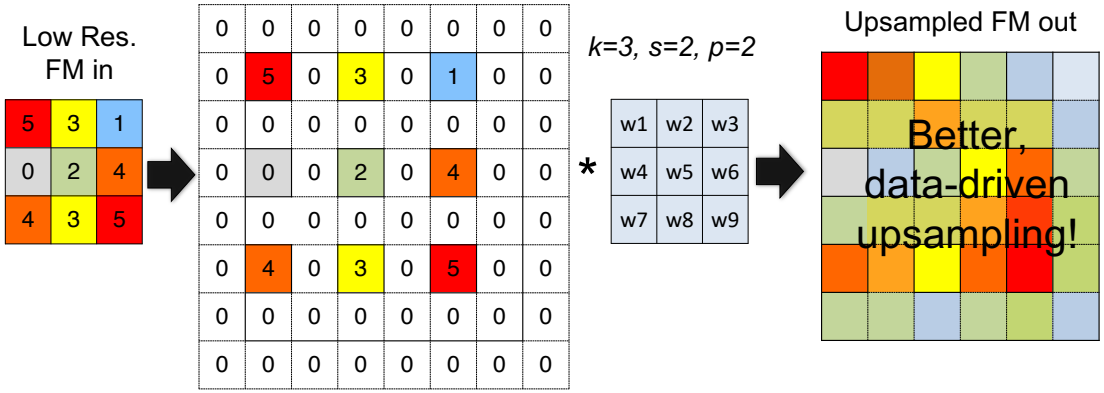
CNNs: Upsampling (Deconvolution or transposed convolution)

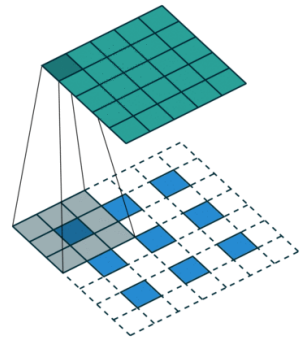




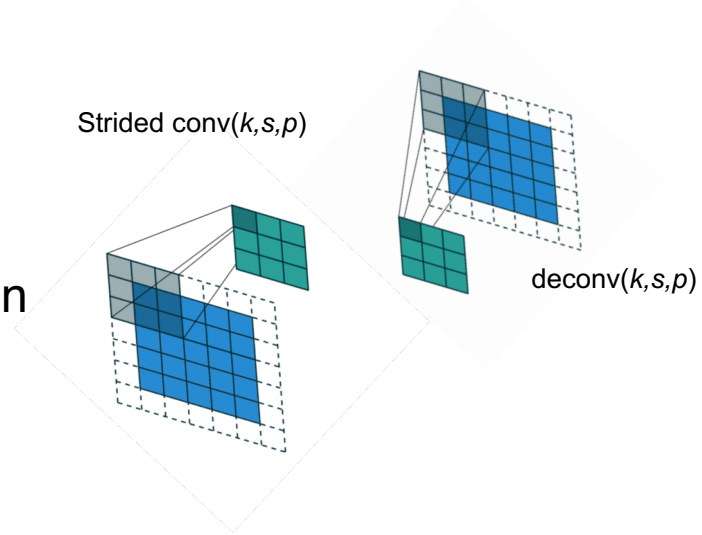
- Deconvolution / transposed conv: Learn to upsample
  - Kernel dimension **k**: how much context influences upsampling
  - Stride **s** of convolution: The upsampling factor
  - Padding **p**: careful, to get exact output shape wanted

CNNs: Upsampling (Deconvolution or transposed convolution)

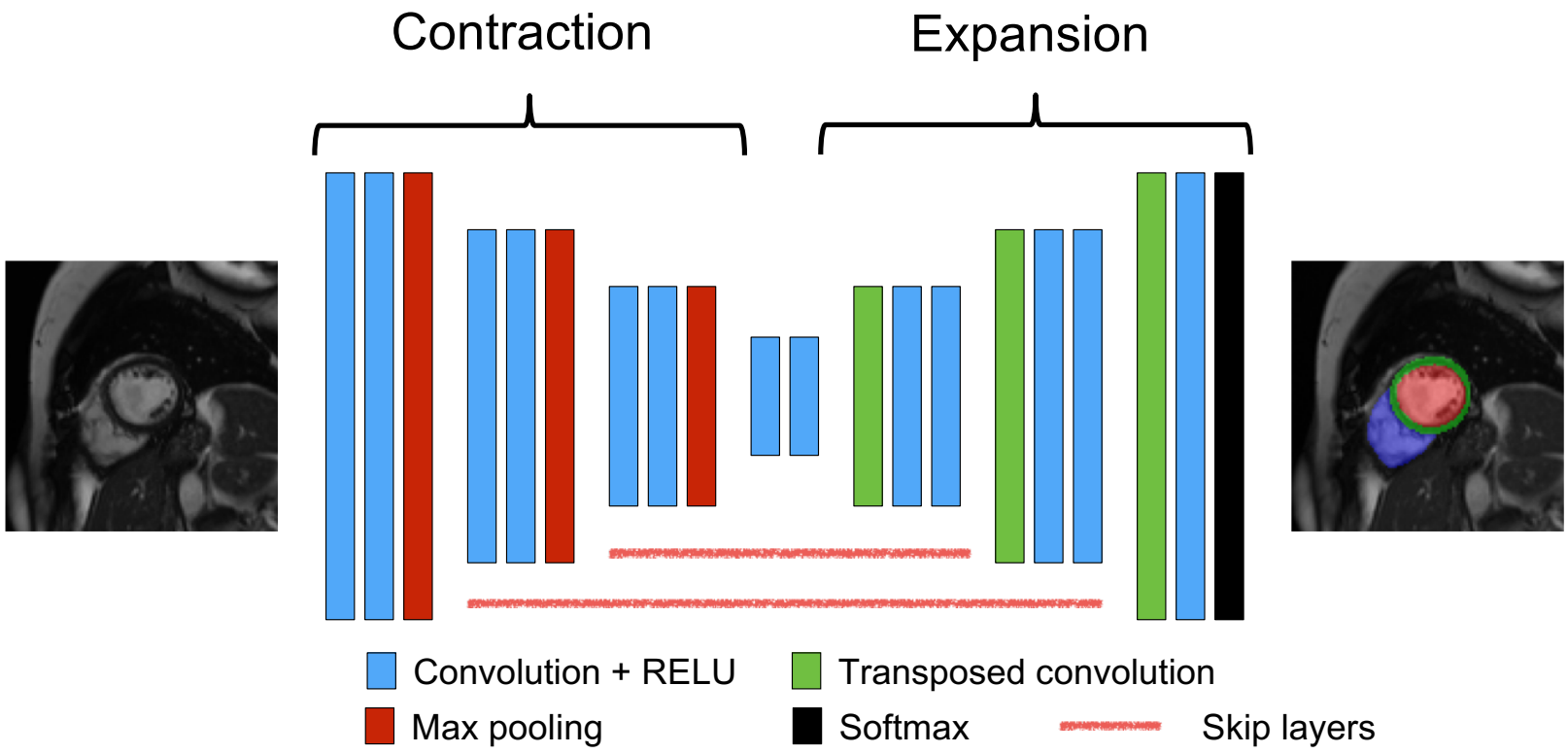




- Alternative interpretation:
  - Deconvolution with (*k,s,p*) reverses the strided convolution with (*k,s,p*) which created the downsampled feature map.

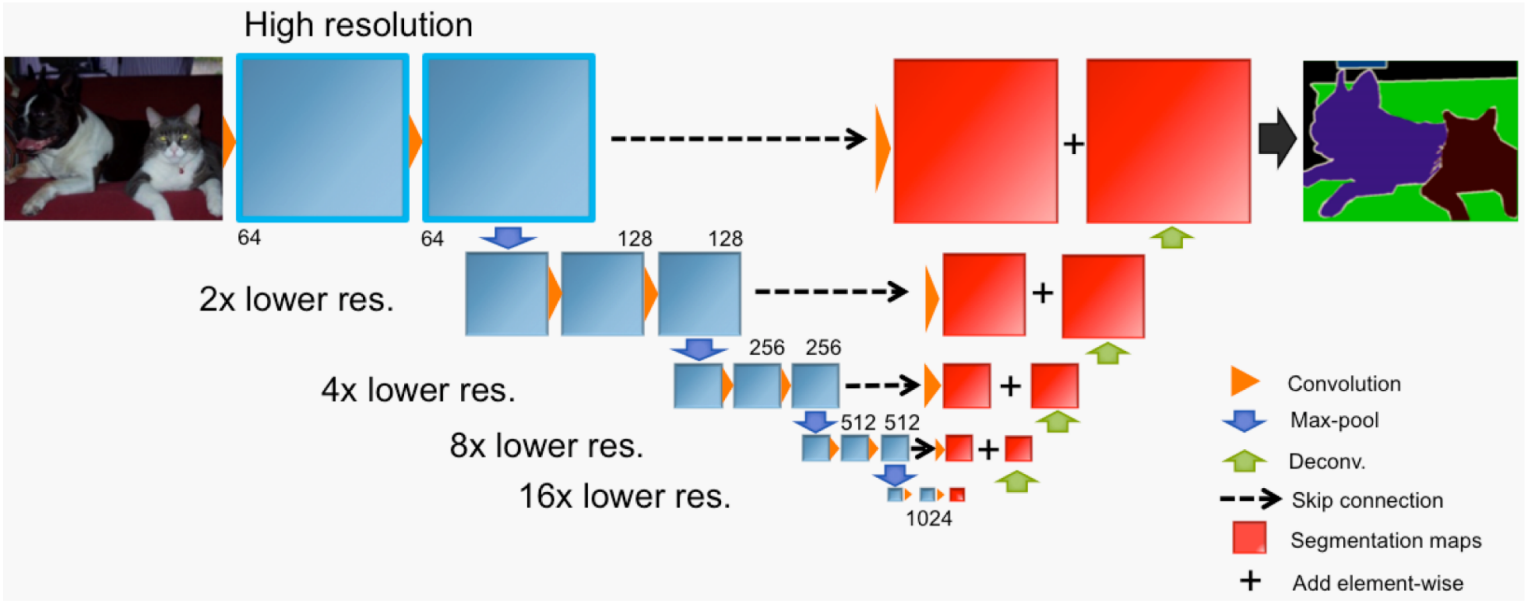


CNNs: Encoder – Decoder networks



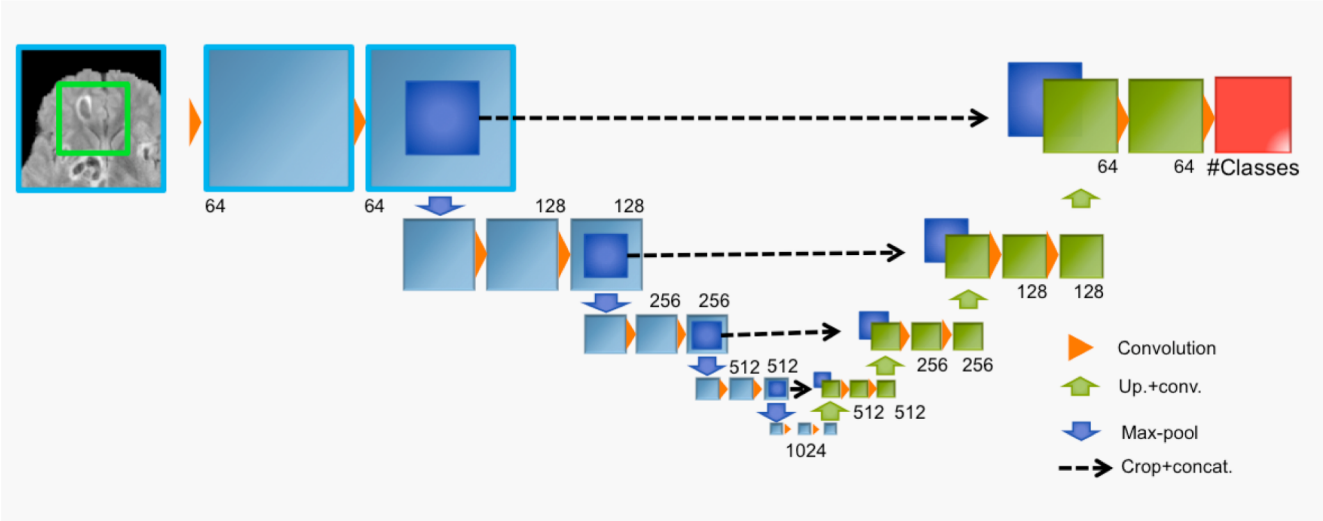
CNNs: Encoder – Decoder networks
Fully convolutional network (FCN)

- Long et al., *Fully convolutional networks for semantic segmentation*, CVPR 2015
  - – Deconvolution to upsample segmentation.
  - – Skip connections to combine multi-scale info for better accuracy.
  - – Popularized fully conv net (previously [LeCun ‘98, Farabet ’13])

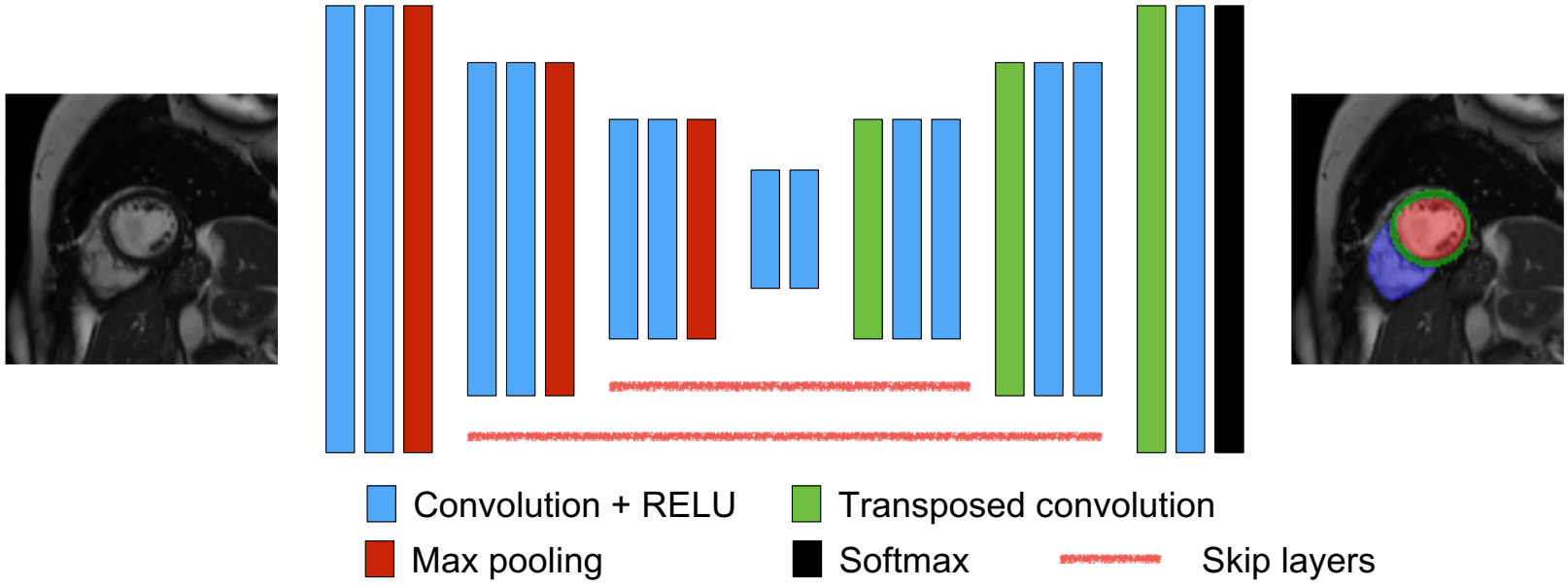


CNNs: Encoder – Decoder networks
U-Net

- Ronneberger et al., *U-net: Convolutional networks for biomedical image segmentation*, MICCAI'15.
  - Extended FCN decoder with more filters.
  - Learns to up-sample and combine multi-scale features, not segmentations (unlike FCN).
  - Original version uses only "valid" convolution (no padding), processing only real content.
  - Often works better than FCN, with the trade-off of more expensive decoder.



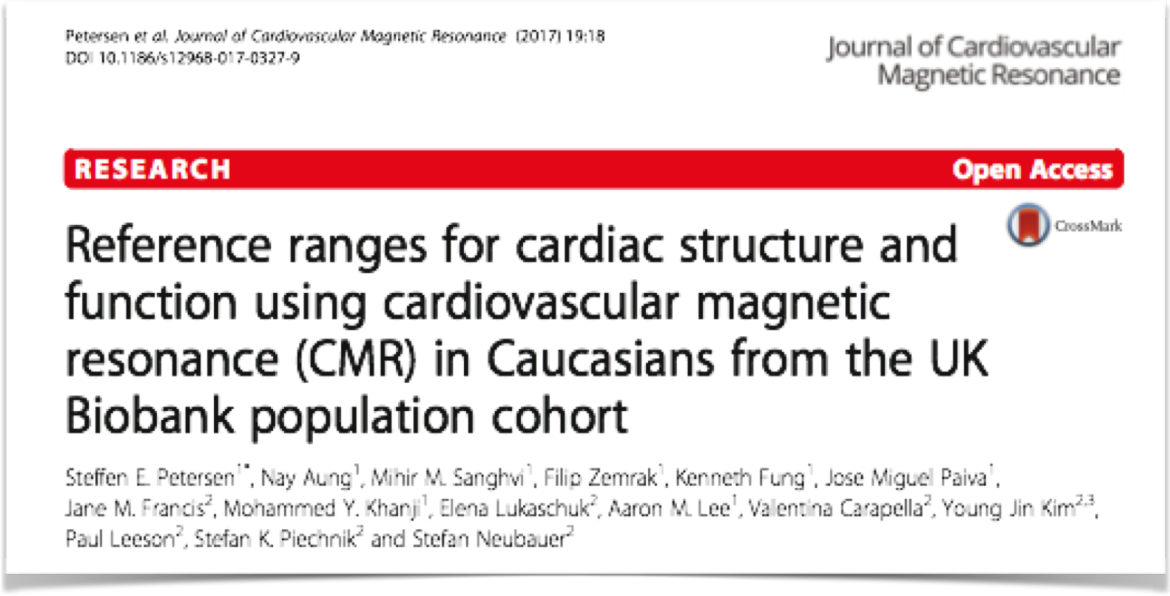
CNNs: Encoder – Decoder networks: Application to cardiac image segmentation



Bai et al., JCMR, *in press*, 2018

CNNs: Encoder – Decoder networks: Application to cardiac image segmentation

- Fully connected networks (Long et al., 2015)
- Manual annotations of **4,872 subjects** (QMUL/Oxford) with **93,128 pixelwise annotated 2D images** slices
- Divided into training/validation/test: 3,972/300/600



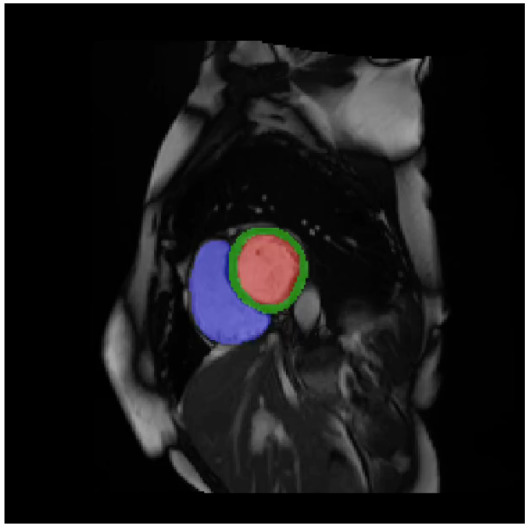
Petersen et al. Journal of Cardiovascular Magnetic Resonance (2017) 19:18
DOI 10.1186/s12968-017-0327-9

Journal of Cardiovascular Magnetic Resonance

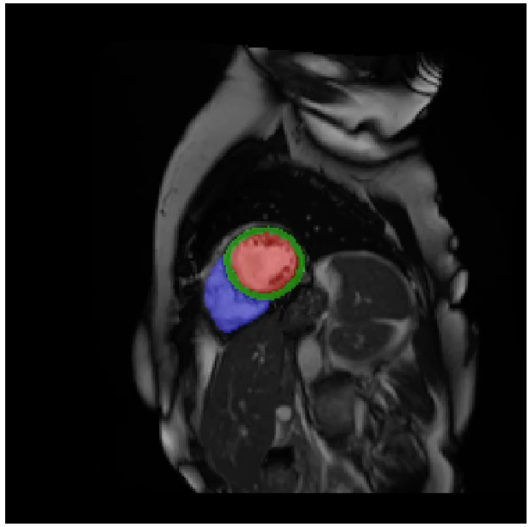
RESEARCH Open Access

Reference ranges for cardiac structure and function using cardiovascular magnetic resonance (CMR) in Caucasians from the UK Biobank population cohort

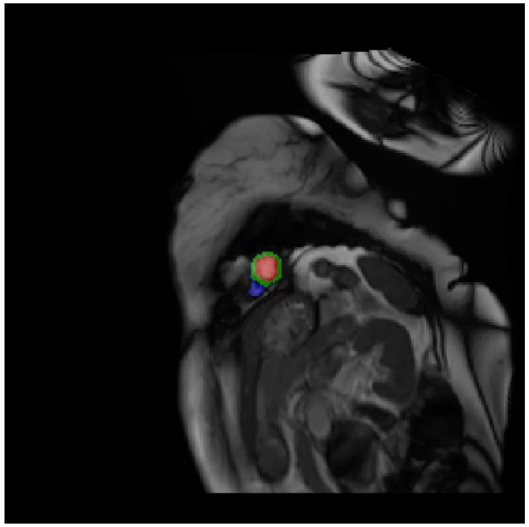
Steffen E. Petersen[1*], Nay Aung[1], Mihir M. Sanghvi[1], Filip Zemrak[1], Kenneth Fung[1], Jose Miguel Paiva[1], Jane M. Francis[2], Mohammed Y. Khanji[1], Elena Lukaschuk[2], Aaron M. Lee[1], Valentina Carapella[2], Young Jin Kim[2,3], Paul Leeson[2], Stefan K. Piechnik[2] and Stefan Neubauer[2]



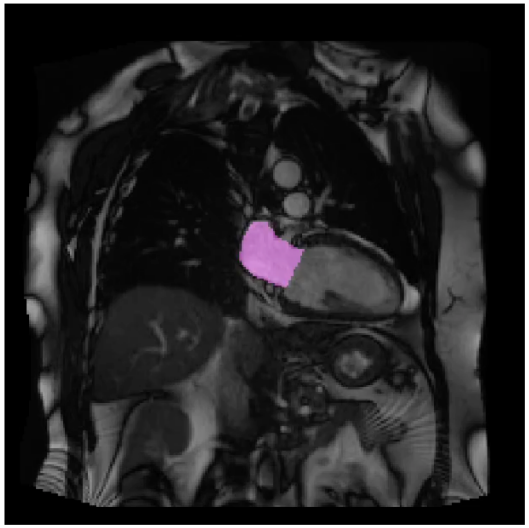
SA, basal



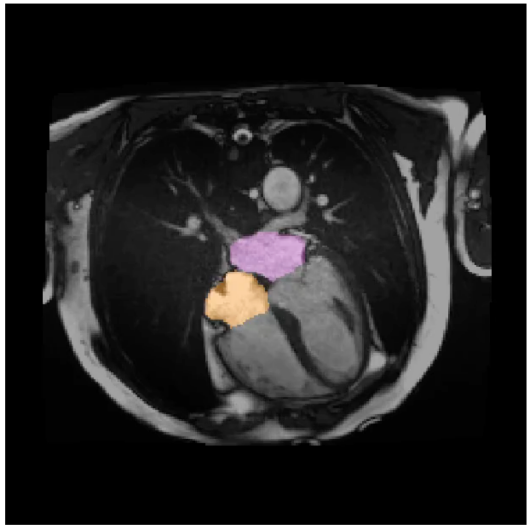
SA, mid-ventricular



SA, apical



LA, 2 chamber



LA, 4 chamber

Bai et al., JCMR, *in press*, 2018

Evaluation of segmentation accuracy
Comparison to expert observers

(a) Absolute difference

	Auto vs Man (n = 600)	O1 vs O2 (n = 50)	O2 vs O3 (n = 50)	O3 vs O1 (n = 50)
LVEDV (mL)	$6.1_{\pm 5.3}$	$6.1_{\pm 4.4}$	$8.8_{\pm 4.8}$	
LVESV (mL)	$5.3_{\pm 4.9}$	$4.1_{\pm 4.2}$		
LVM (gram)	$6.9_{\pm 5.5}$	$4.2_{\pm 3.2}$		
RVEDV (mL)	$8.5_{\pm 7.1}$			$8.7_{\pm 5.8}$
RVESV (mL)	$7.2_{\pm 6.8}$			$11.7_{\pm 6.9}$

	vs O2 (n = 50)	O2 vs O3 (n = 50)	O3 vs O1 (n = 50)
LV		$4.2_{\pm 3.1}$	$6.3_{\pm 3.3}$	$3.4_{\pm 2.2}$
		$6.8_{\pm 7.5}$	$12.5_{\pm 8.5}$	$11.7_{\pm 5.1}$
		$4.4_{\pm 3.3}$	$6.0_{\pm 3.7}$	$6.7_{\pm 4.6}$
V (%)		$8.0_{\pm 5.0}$	$4.2_{\pm 3.1}$	$5.7_{\pm 3.6}$
RVESV (%)	$11.8_{\pm 12.2}$	$30.6_{\pm 15.5}$	$10.9_{\pm 8.3}$	$16.9_{\pm 9.2}$

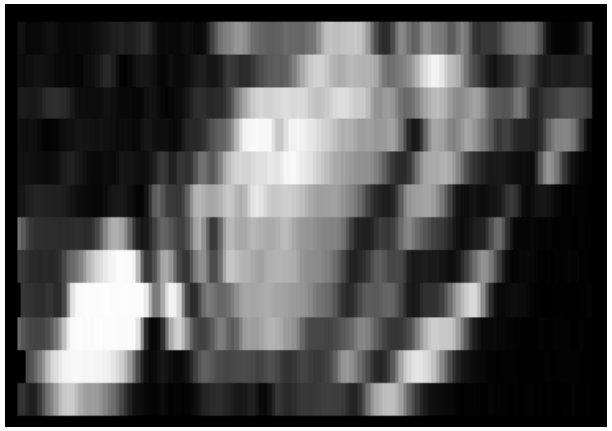
Computer performs as well as different expert observers

Automated

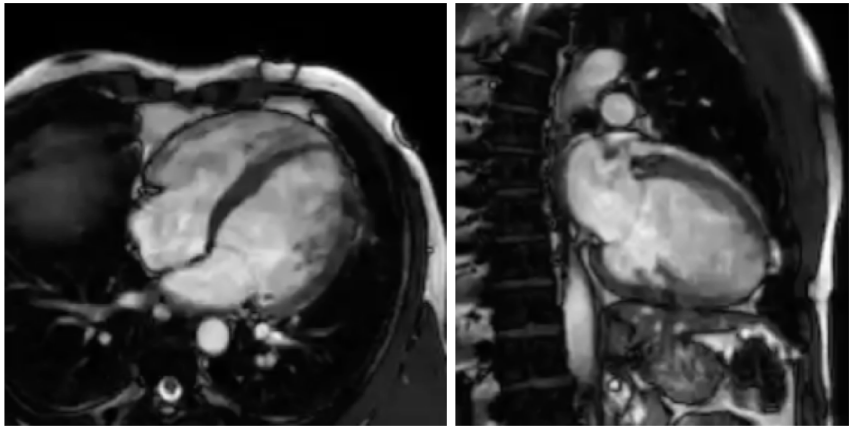
Manual

Bai et al., JCMR, *in press*, 2018

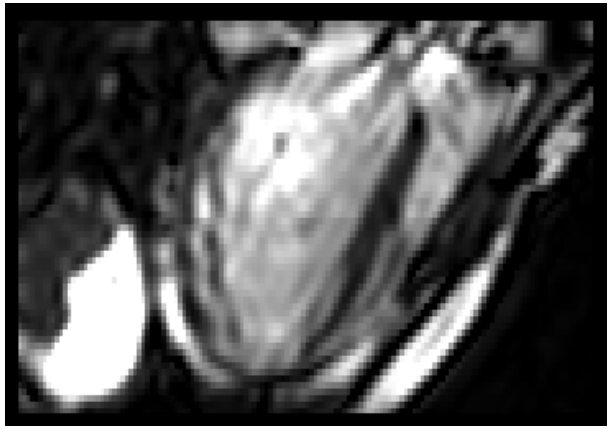
CNNs: Encoder – Decoder networks
Application to super-resolution



plus

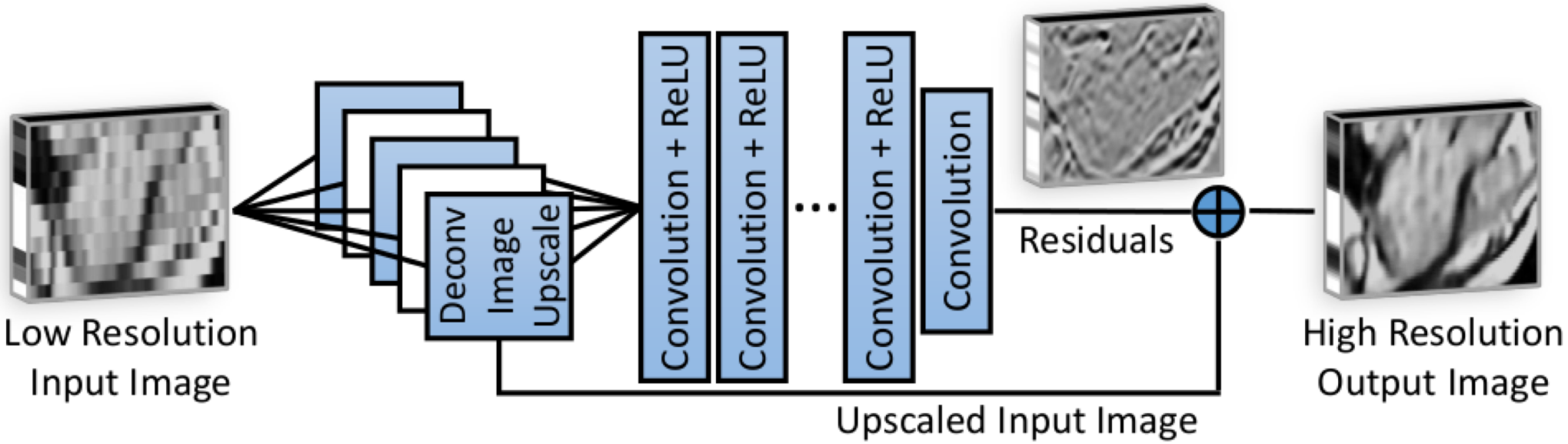


?



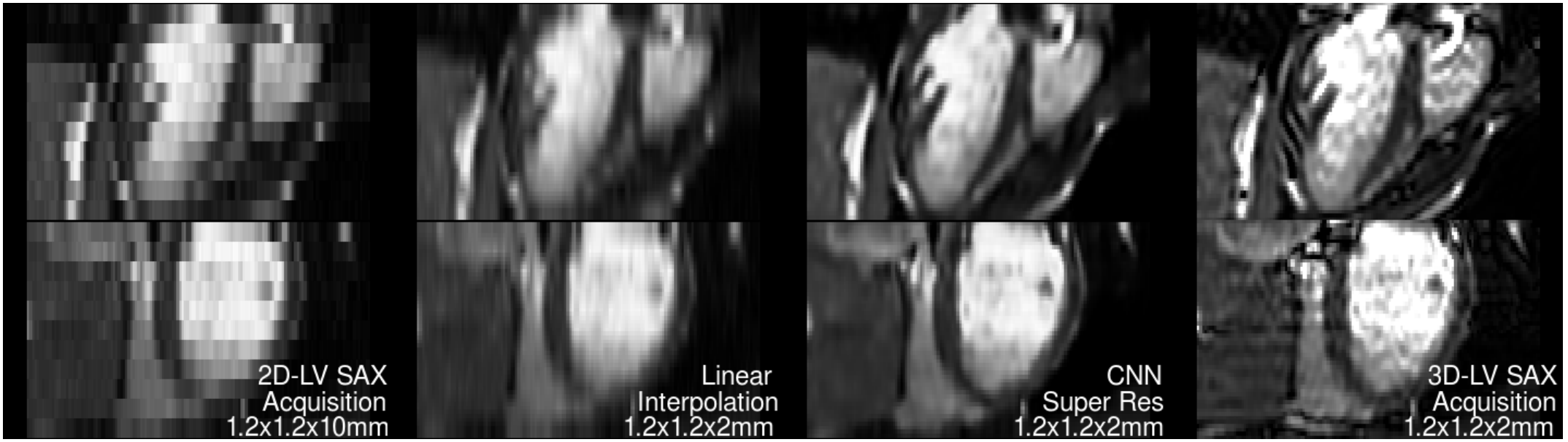
O. Oktay et al. MICCAI 2016

CNNs: Encoder – Decoder networks
Application to super-resolution



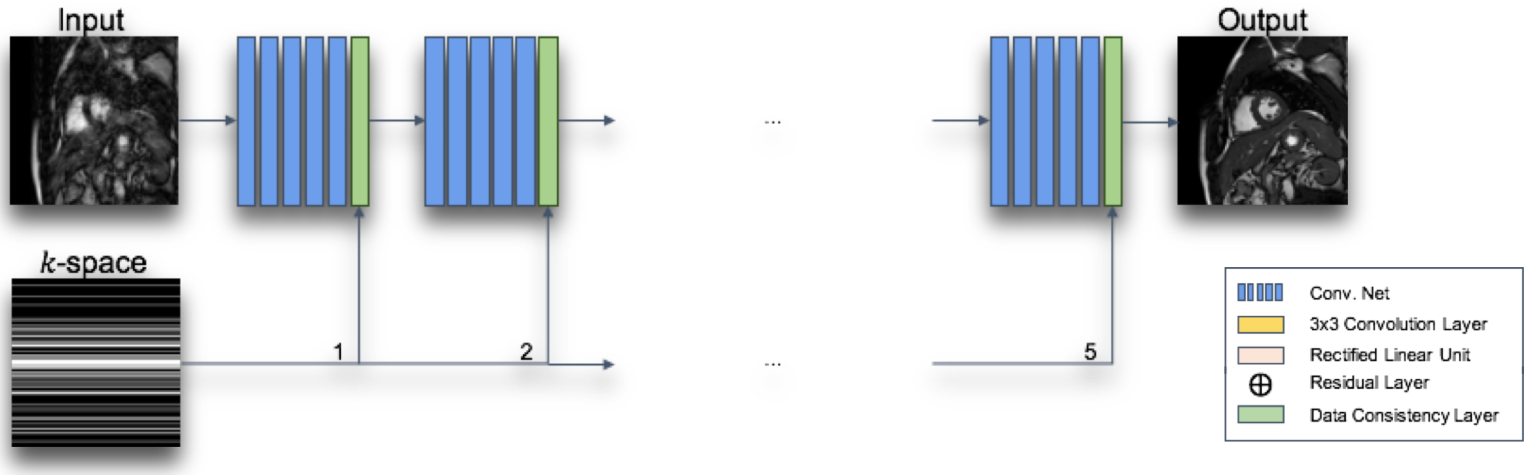
O. Oktay et al. MICCAI 2016

CNNs: Encoder – Decoder networks
Application to super-resolution



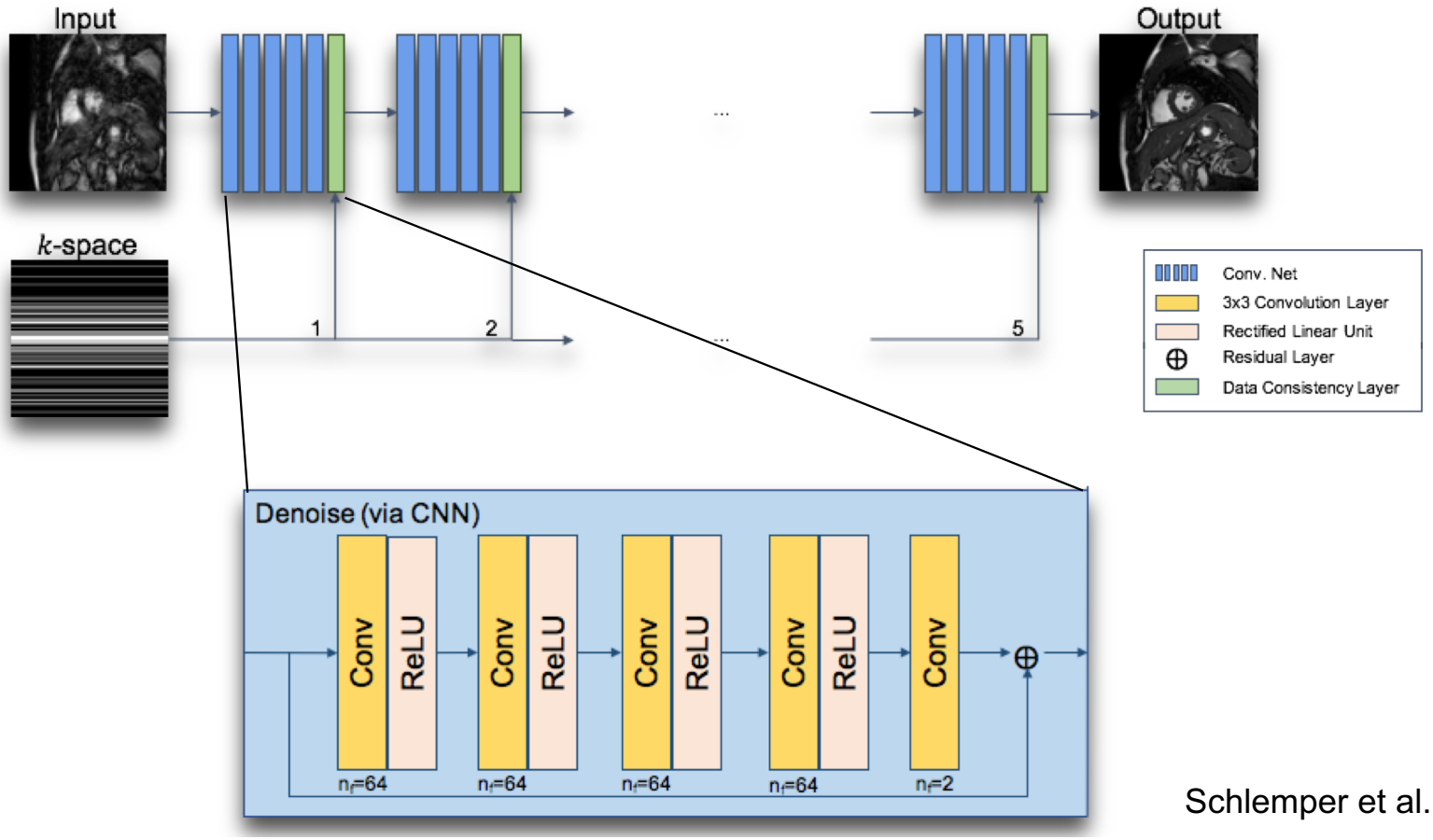
O. Oktay et al. MICCAI 2016

CNNs: Encoder – Decoder networks
Application to image reconstruction



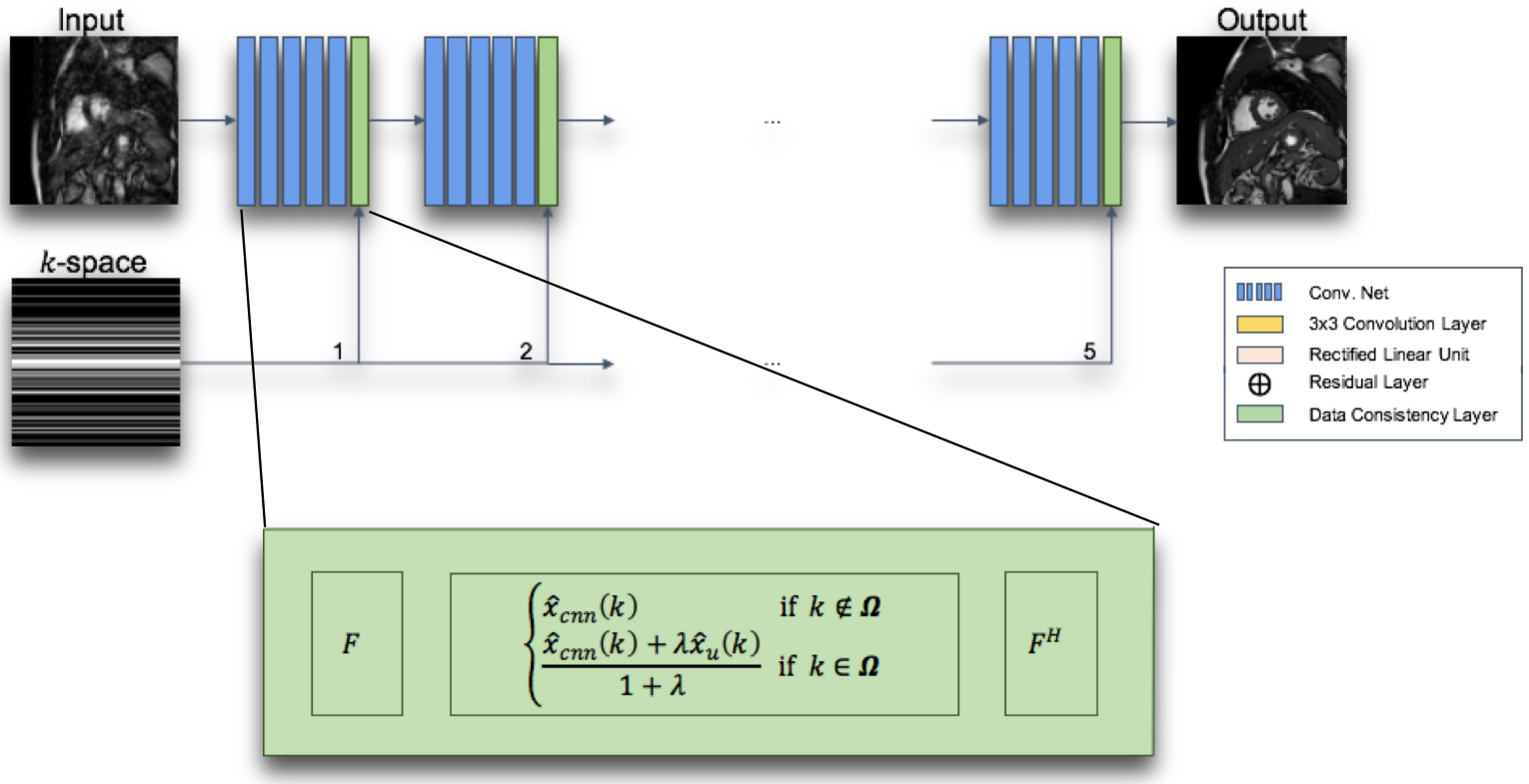
Schlemper et al. IEEE TMI 2017

CNNs: Encoder – Decoder networks
Application to image reconstruction



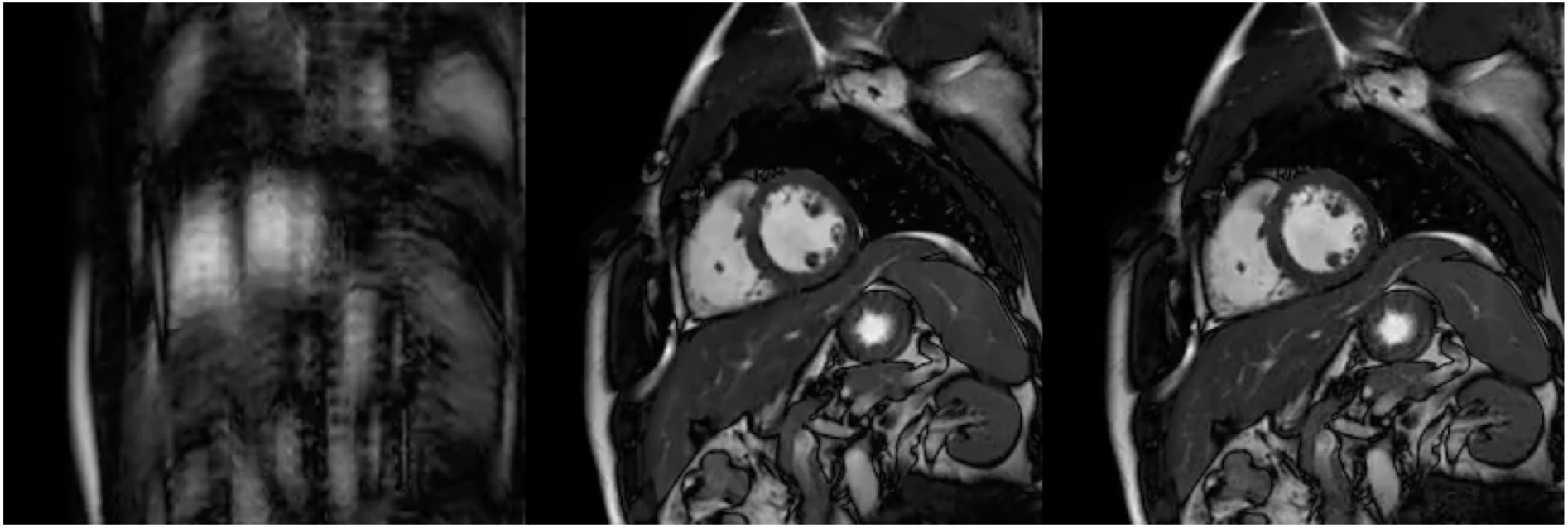
Schlemper et al. IEEE TMI 2017

CNNs: Encoder – Decoder networks
Application to image reconstruction



Schlemper et al. IEEE TMI 2017

CNNs: Encoder – Decoder networks
Application to image reconstruction



(a) 6x Undersampled **(b) CNN reconstruction** **(c) Ground Truth**

Schlemper et al. IEEE TMI 2017

More to come on Thursday!

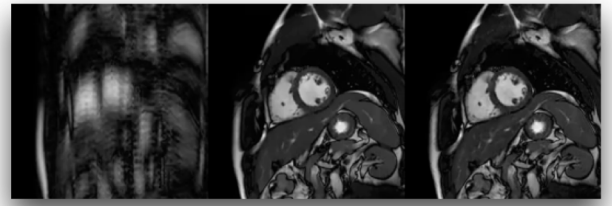


Image reconstruction

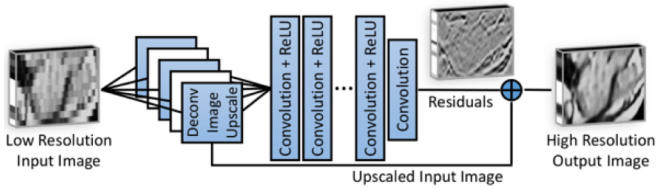


Image super-resolution

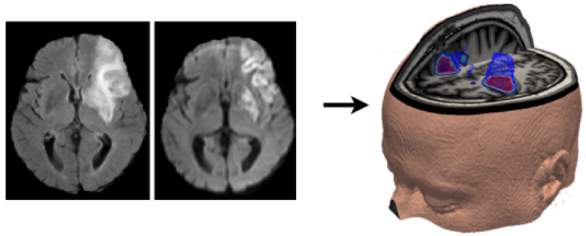


Image segmentation

Abdominal View
Confidence: 98%

Image classification

Thank you