Deep learning for medical imaging

Olivier Colliot, PhD
Research Director at CNRS
Co-Head of the ARAMIS Lab – www.aramislab.fr
PRAIRIE – Paris Artificial Intelligence Research Institute

Maria Vakalopoulou, PhD
Assistant Professor at CentraleSupelec
Mathematics and Informatics (MICS)
Office: Bouygues Building Sb.132

Master 2 - MVA

Course website: http://www.aramislab.fr/teaching/DLMI-2019-2020/
Piazza (for registered students): https://piazza.com/centralesupelec/spring2020/mvadlmi/

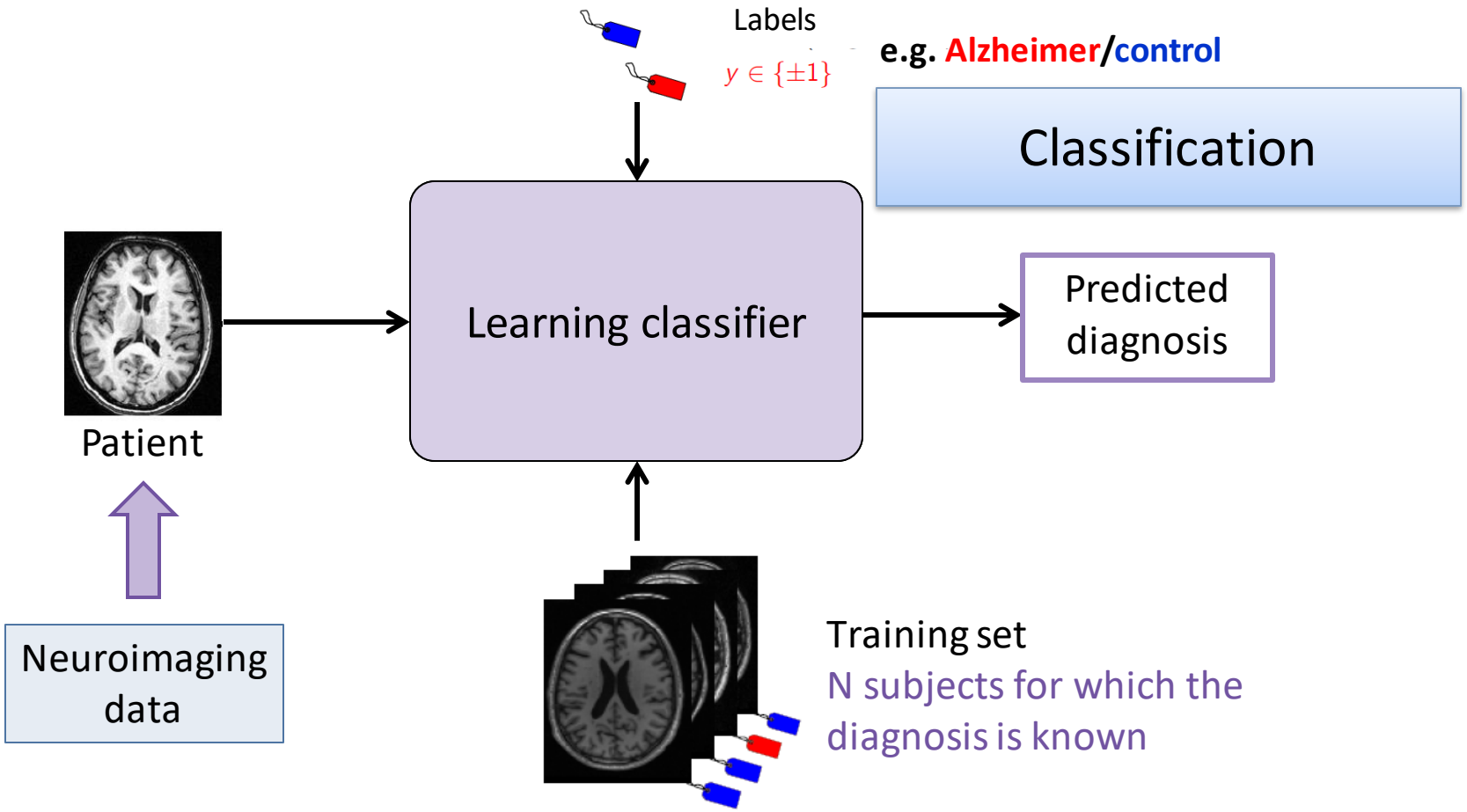
Acknowledgements

- The lecture is partially base on material by:
  - Andrej Karpathy
  - Fei-Fei
  - Daniel Rueckert
  - Ben Glocker
  - Shubhendu Trivedi & Risi Kondor

Thank you!!

Previous Lecture
Classification & Detection

Example in brain image classification

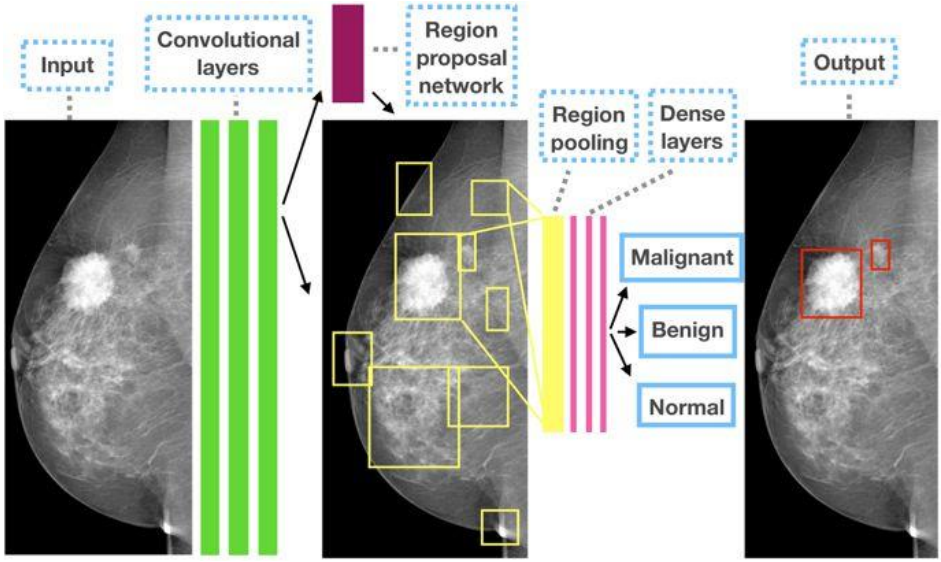


Detecting lesions in mammograms

- Classification & Localization
- Object Detection
- R-CNN
- Fast R-CNN
- Faster R-CNN
- ….

Medical Imaging

- 3D extensions
- 2 stage processes to remove false positives
- …



Ribli et al. 2018

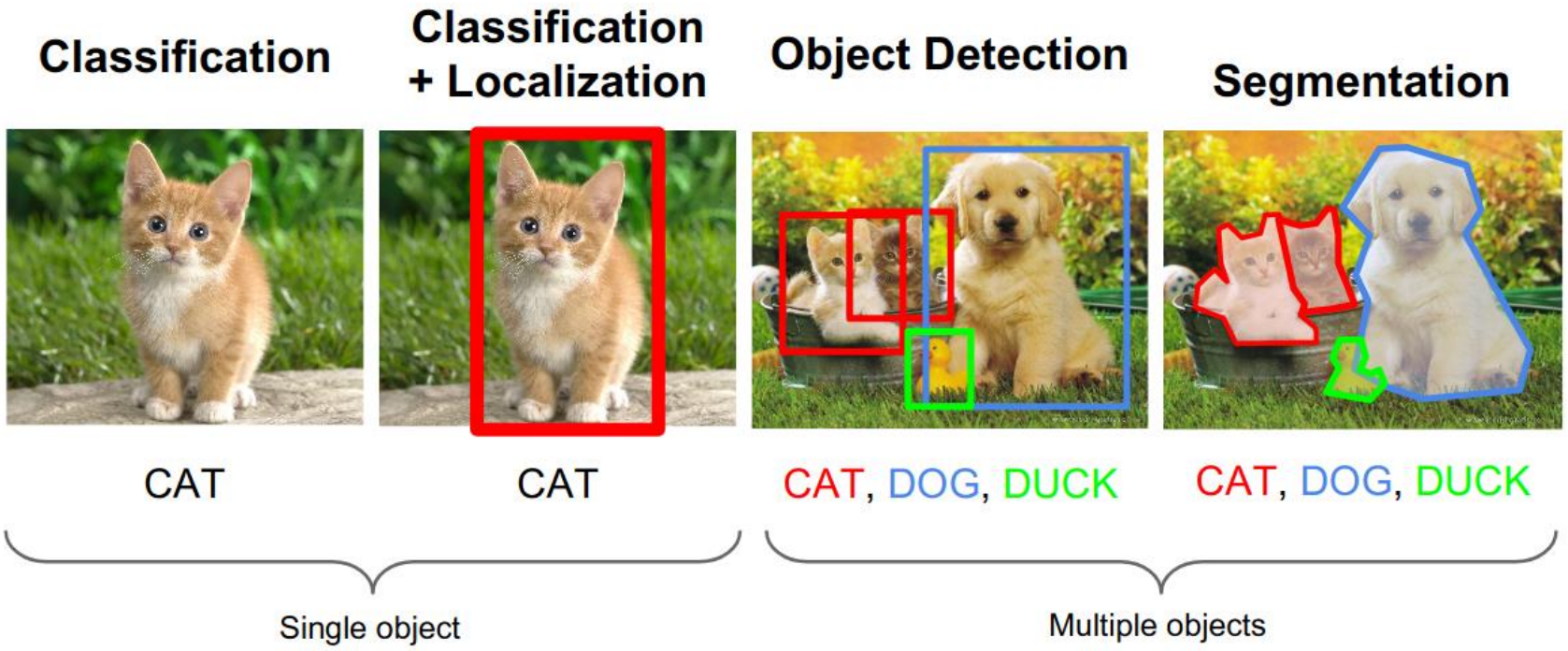
Part 4 – Segmentation

Outline

- Segmentation
  - Upsampling & Dilated Filters
  - Loss Functions
  - Evaluation Metrics
- Medical Imaging

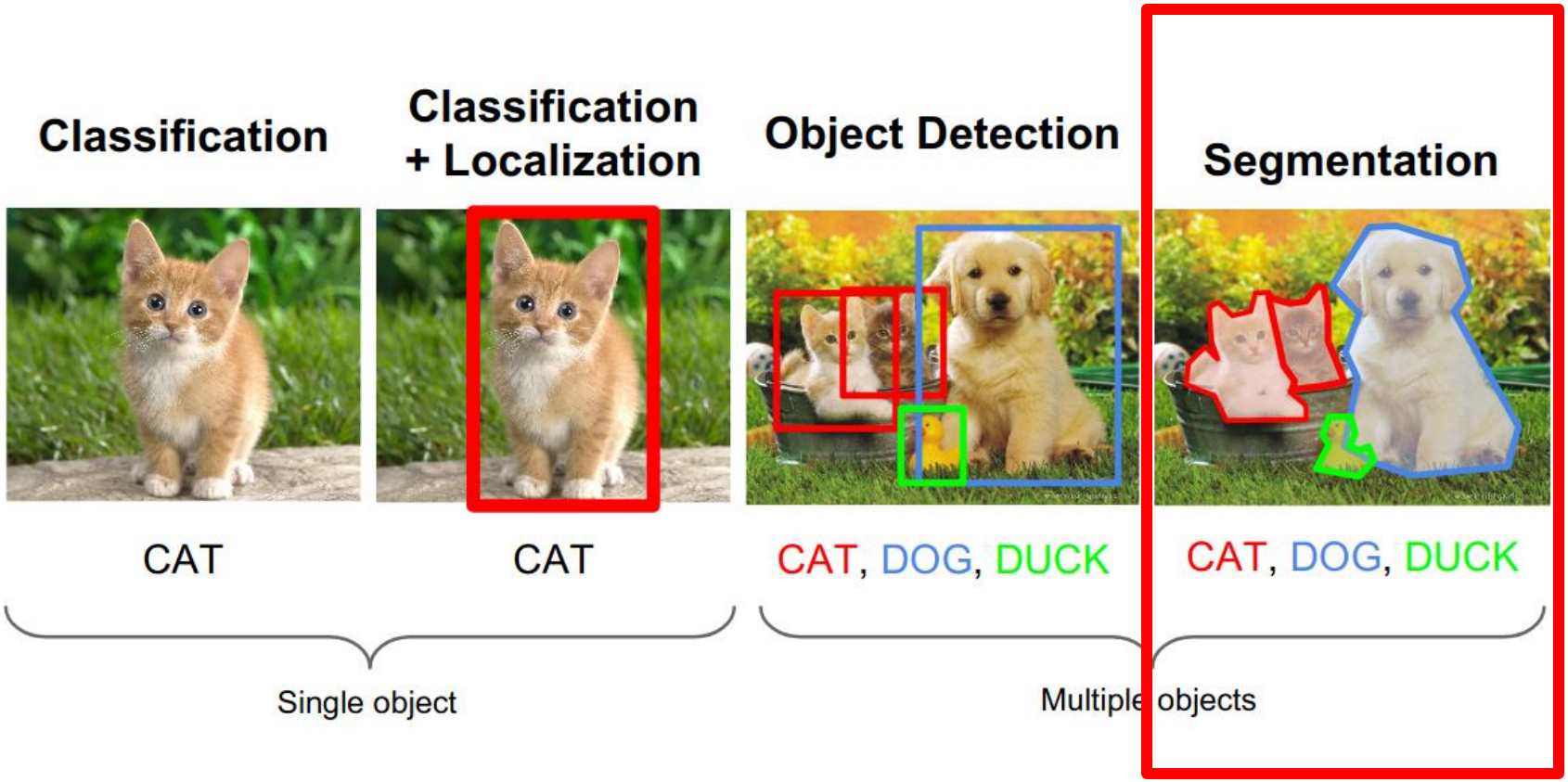
Introduction

- Different problems for vision



Introduction

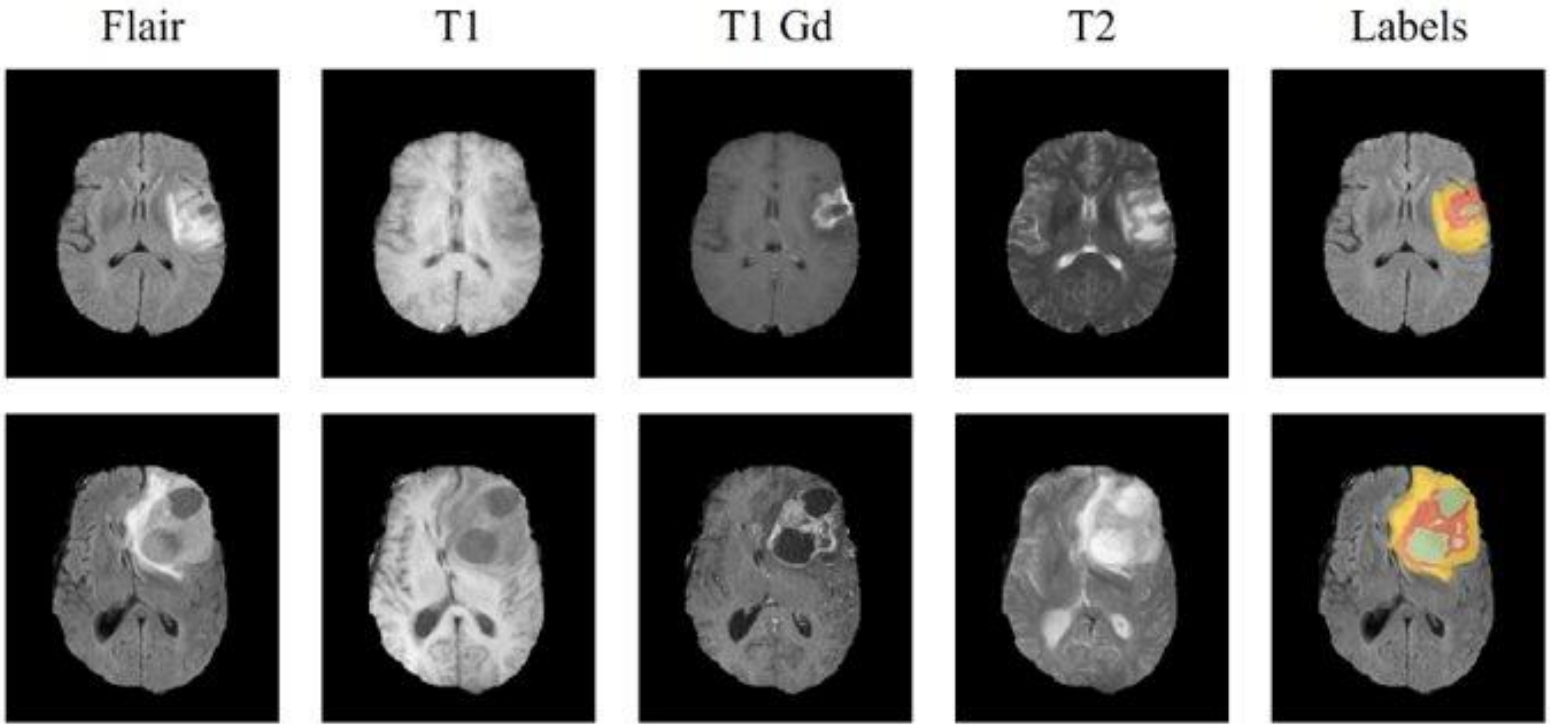
- Different problems for vision



Introduction

- Semantic Segmentation

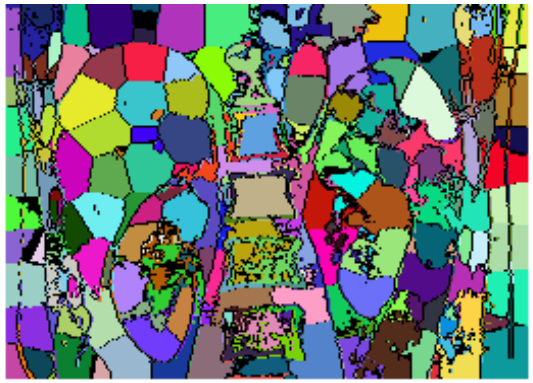
  - Label each pixel in the image with a category label
  - Do not differentiate instances only care about pixels, we want to find the mesh of pixels with the same label



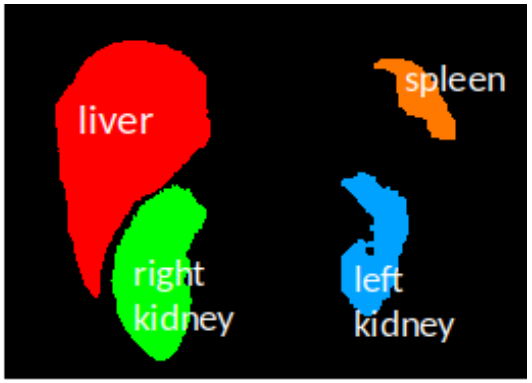
Introduction

- In Semantic Segmentation, a segmentated region is also assigned a **semantic meaning**
- This is in contrast to segmentation based on 'pure' clustering of image into coherent regions
- For this lecture, we are mostly interested in semantic segmentation

raw image



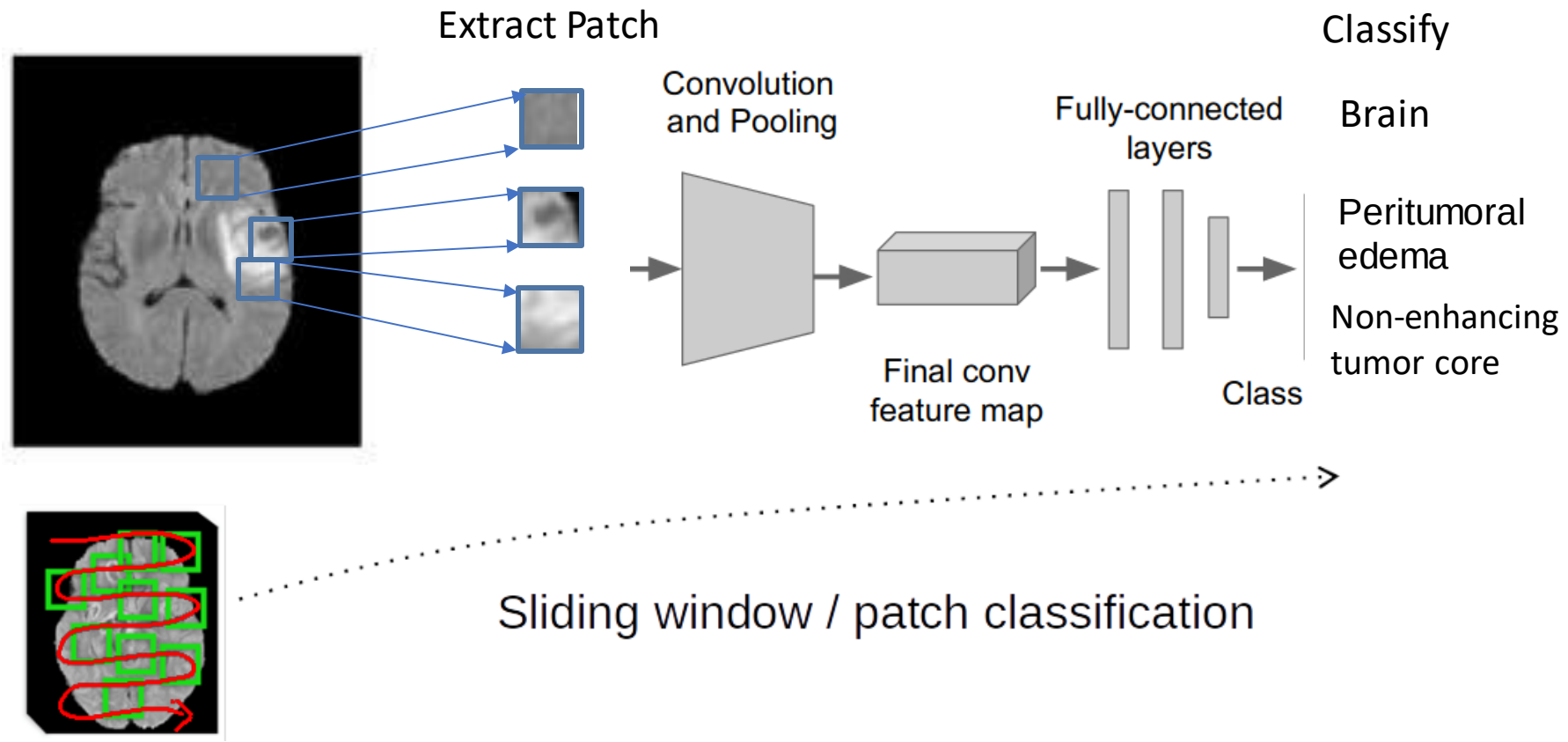
segmentation/clustering



semantic segmentation

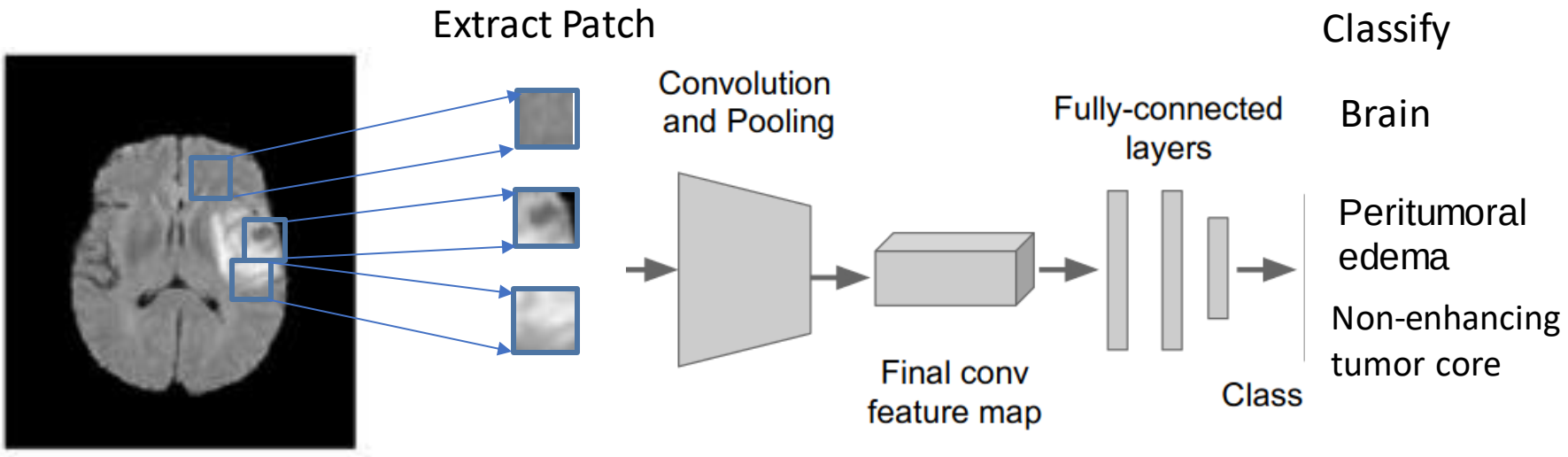
Introduction

- Semantic Segmentation
- Idea #1: Image Segmentation as Classification using sliding window



Introduction

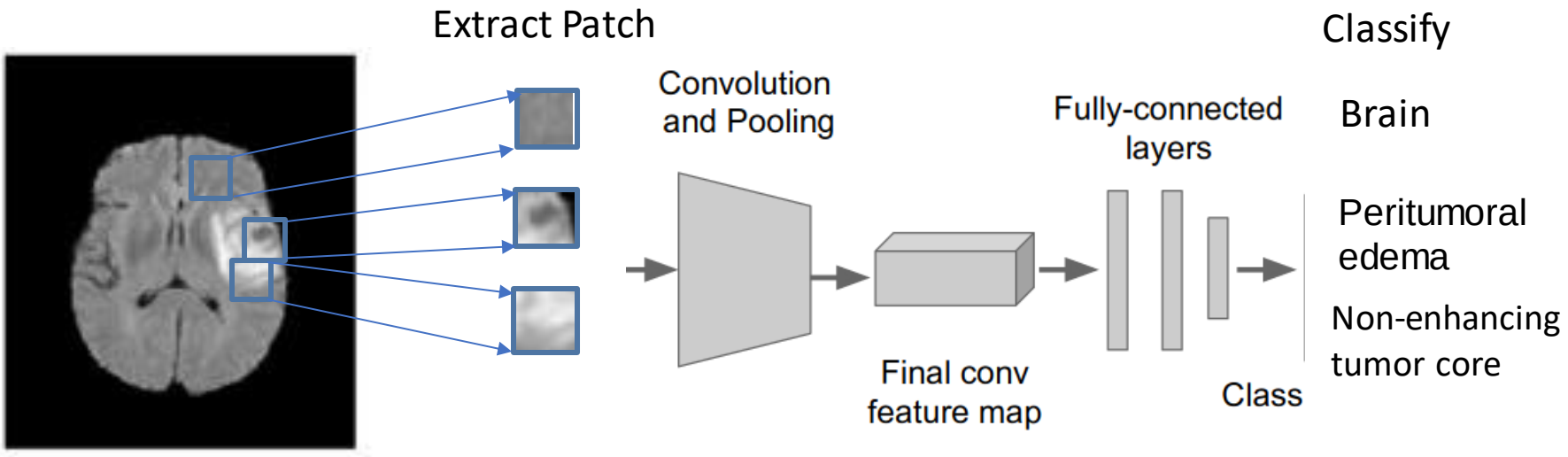
- Semantic Segmentation
- Idea #1: Image Segmentation as Classification using sliding window



- Problems??

Introduction

- Semantic Segmentation
- Idea #1: Image Segmentation as Classification using sliding window

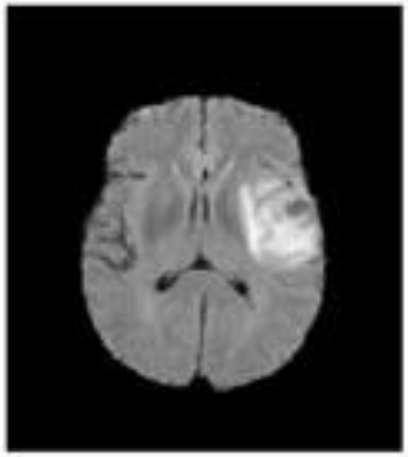


- Problems??
  - Very inefficient (both time and complexity)
  - Using information only in a small part of the image
  - Do not reuse shared features between overlapping patches

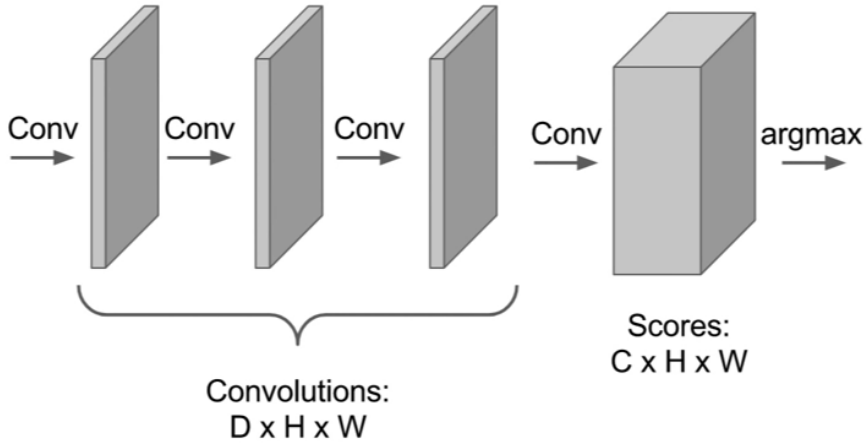
Introduction

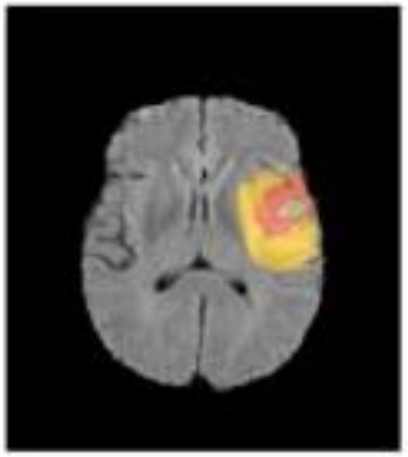
- Semantic Segmentation

- Idea #2: Image Segmentation using Fully Convolutional Layers
  - Design a network as a bunch of convolutional layers to make predictions for pixels all at once!



Input HxW



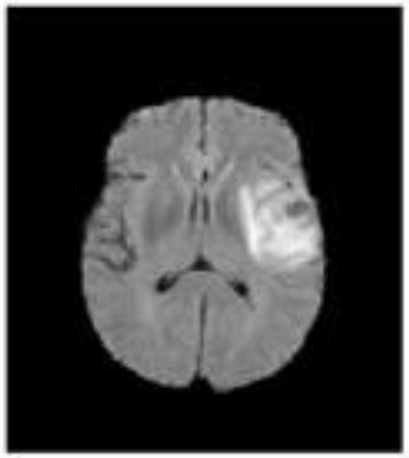


Output HxW

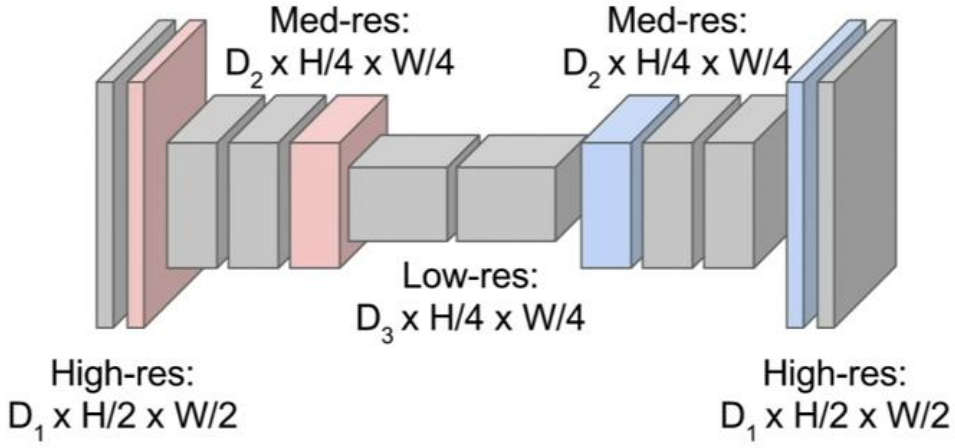
Problems?

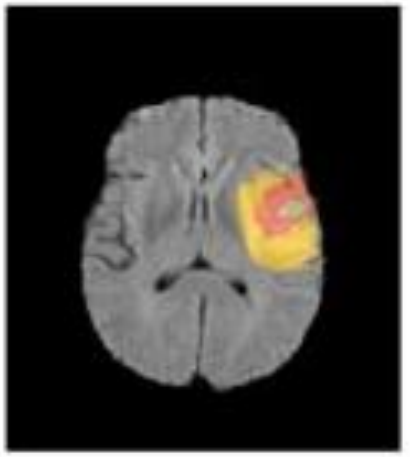
Introduction

- Semantic Segmentation
- Idea #2: Image Segmentation using Fully Convolutional Layers
  - Design a network as a bunch of convolutional layers to make predictions for pixels all at once!



Input HxW





Output HxW

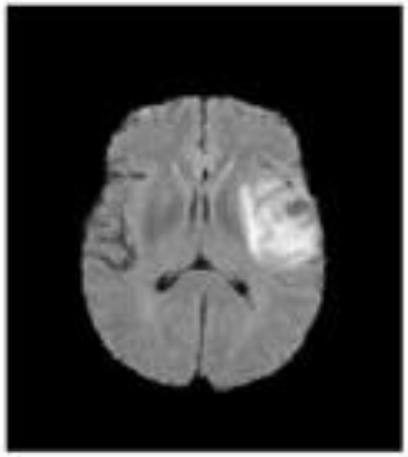
- Use Downsampling and upsampling inside the network!

Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015
Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

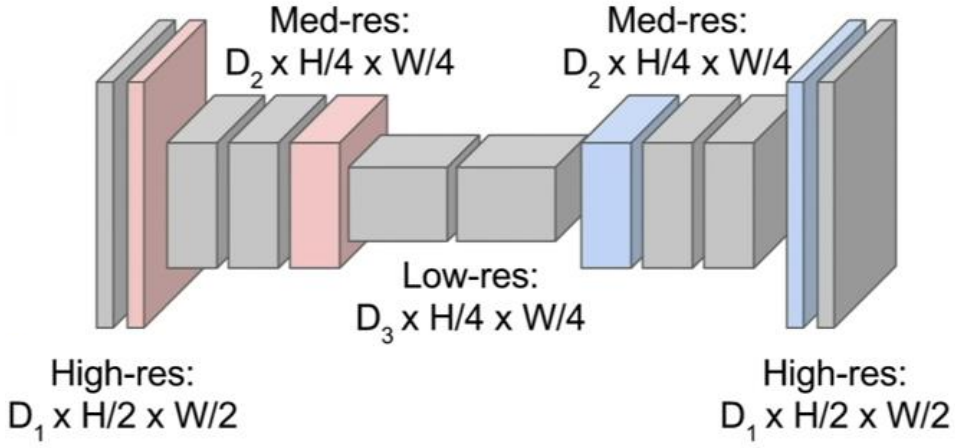
Introduction

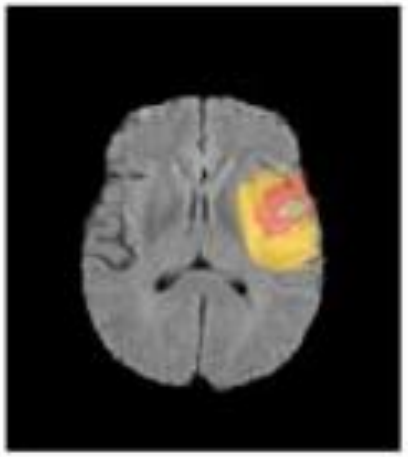
- Semantic Segmentation

- Downsampling: Pooling, strided convolution
- Upsampling ??



Input HxW





Output HxW

Upsampling

- In-Network upsampling: "Unpooling"

Nearest Neighbor

1	2
3	4

→

1	1	2	2
1	1	2	2
3	3	4	4
3	3	4	4

Input: 2 x 2 Output: 4 x 4

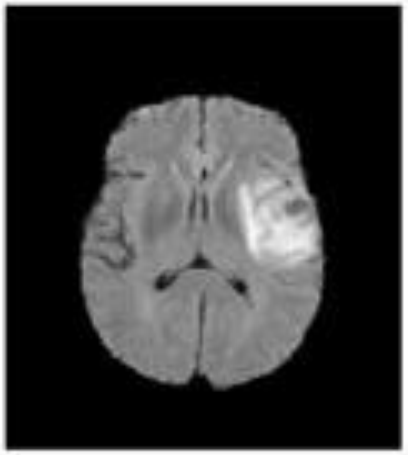
"Bed of Nails"

1	2
3	4

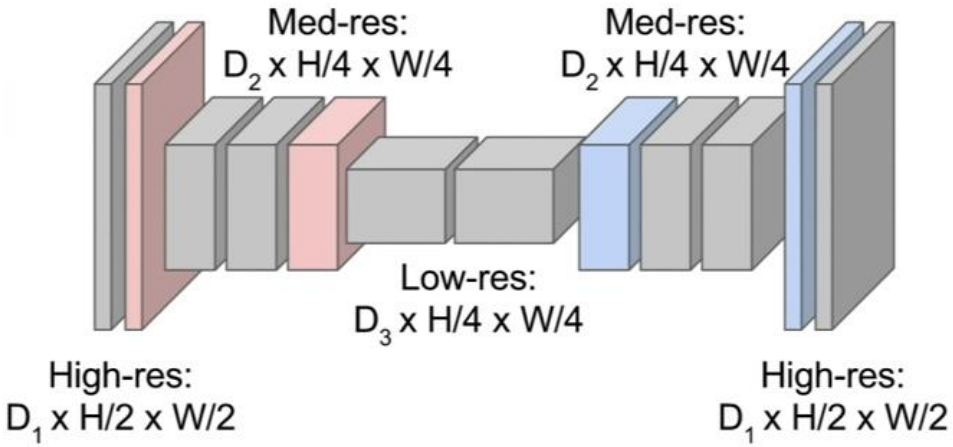
→

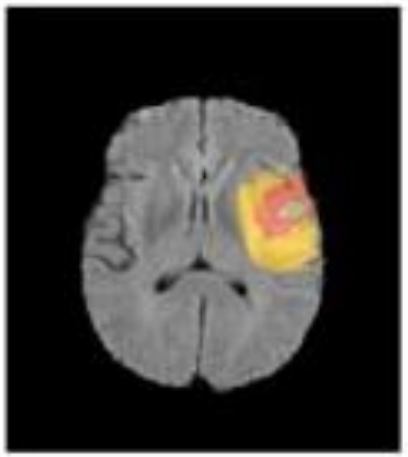
1	0	2	0
0	0	0	0
3	0	4	0
0	0	0	0

Input: 2 x 2 Output: 4 x 4



Input HxW





Output HxW

Upsampling

- Artefacts

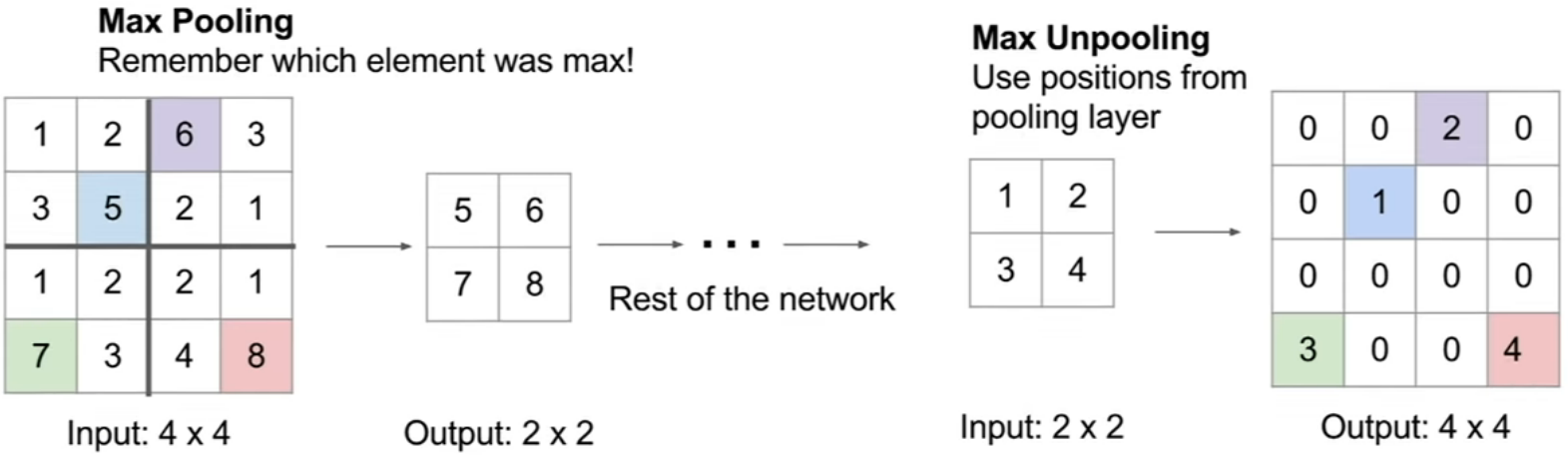
Transpose convolution with **zero-fill / "bed of nails"**

Transpose convolution with **nearest-neighbour-fill**

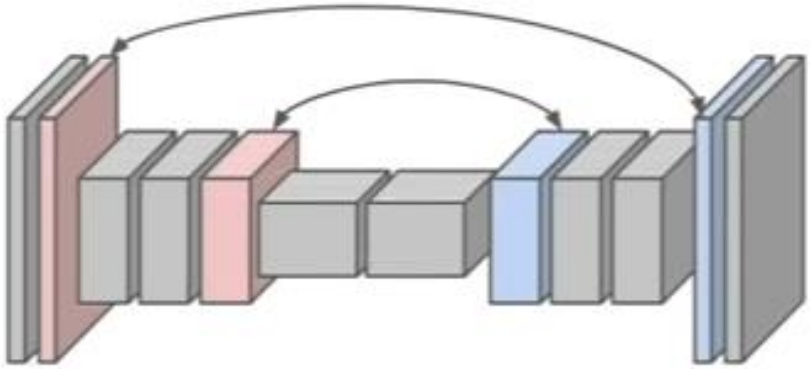
From: http://distill.pub/2016/deconv-checkerboard/

Upsampling

- In-Network upsampling: "Max Unpooling"



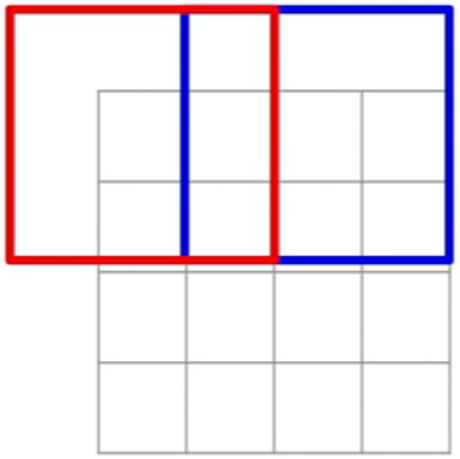
Corresponding pairs of downsampling and upsampling layers



Upsampling

- Learnable Upsampling: Transpose Convolution *(Deconvolution)*

Recall: Normal 3 x 3 convolution, stride 2 pad 1



Input: 4 x 4

Dot product between filter and input

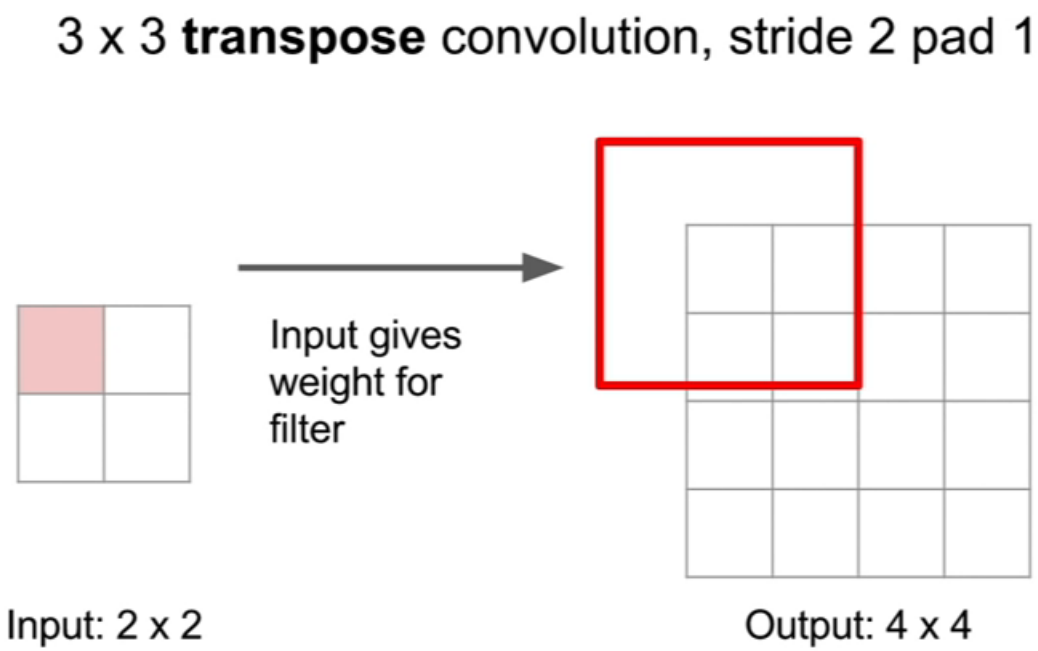
Output: 2 x 2

Filter moves 2 pixels in the input for every one pixel in the output

Stride gives ratio between movement in input and output

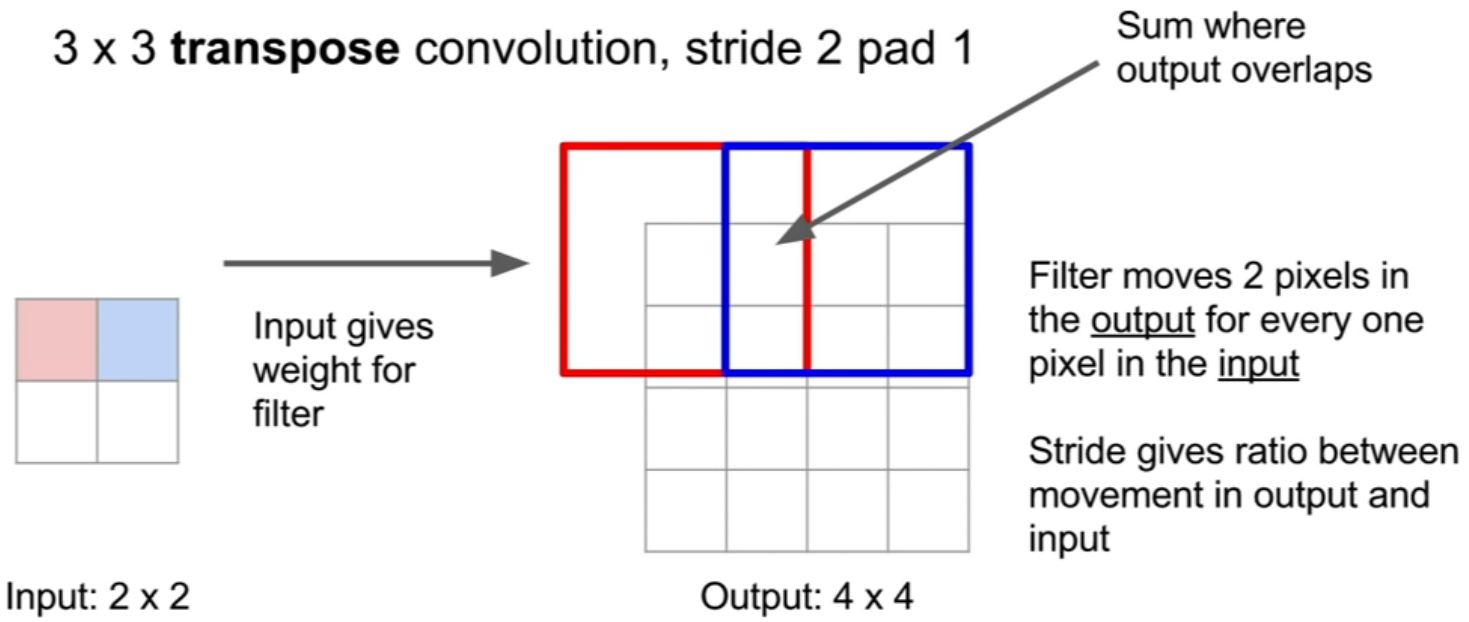
Upsampling

- Learnable Upsampling: Transpose Convolution *(Deconvolution)*



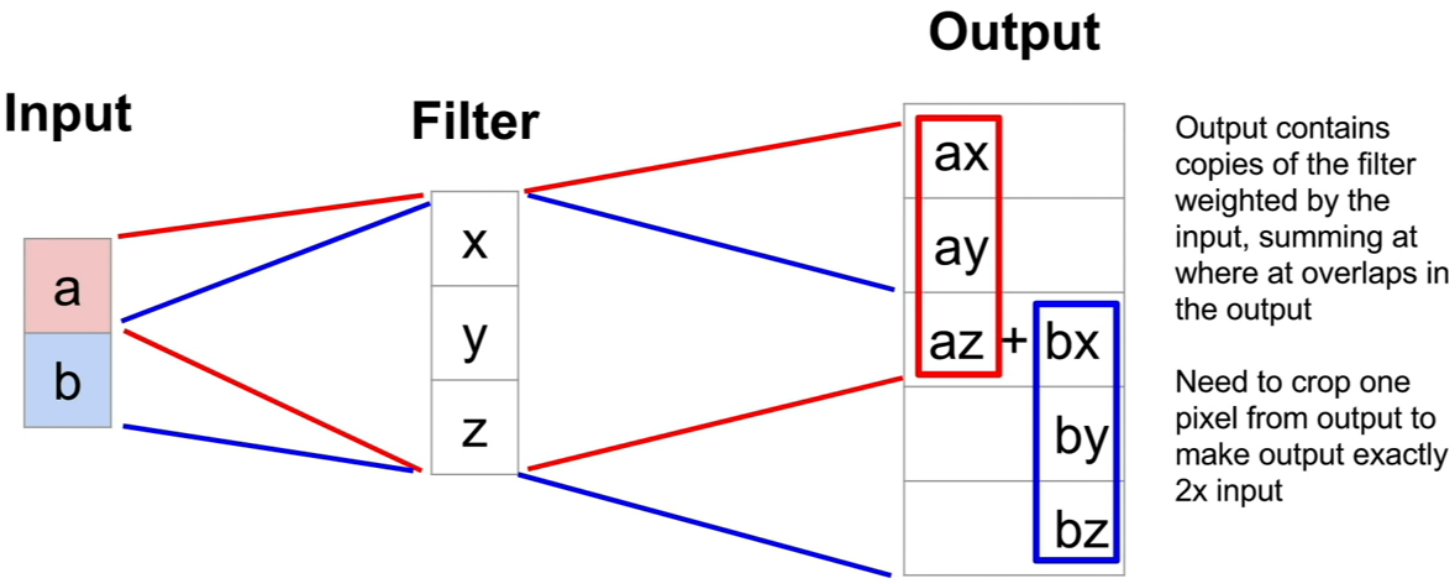
Upsampling

- Learnable Upsampling: Transpose Convolution *(Deconvolution)*



Upsampling

- Transpose Convolution: 1D Example



Upsampling

- Convolution as Matrix Multiplication: 1D Example

We can express convolution in terms of a matrix multiplication

$$\vec{x} * \vec{a} = X\vec{a}$$

$$\begin{bmatrix} x & y & x & 0 & 0 & 0 \\ 0 & x & y & x & 0 & 0 \\ 0 & 0 & x & y & x & 0 \\ 0 & 0 & 0 & x & y & x \end{bmatrix} \begin{bmatrix} 0 \\ a \\ b \\ c \\ d \\ 0 \end{bmatrix} = \begin{bmatrix} ay + bz \\ ax + by + cz \\ bx + cy + dz \\ cx + dy \end{bmatrix}$$

Example: 1D conv, kernel size=3, stride=1, padding=1

Upsampling

- Convolution as Matrix Multiplication: 1D Example

We can express convolution in terms of a matrix multiplication

$$\vec{x} * \vec{a} = X\vec{a}$$

$$\begin{bmatrix} x & y & x & 0 & 0 & 0 \\ 0 & x & y & x & 0 & 0 \\ 0 & 0 & x & y & x & 0 \\ 0 & 0 & 0 & x & y & x \end{bmatrix} \begin{bmatrix} 0 \\ a \\ b \\ c \\ d \\ 0 \end{bmatrix} = \begin{bmatrix} ay + bz \\ ax + by + cz \\ bx + cy + dz \\ cx + dy \end{bmatrix}$$

Example: 1D conv, kernel size=3, stride=1, padding=1

Convolution transpose multiplies by the transpose of the same matrix:

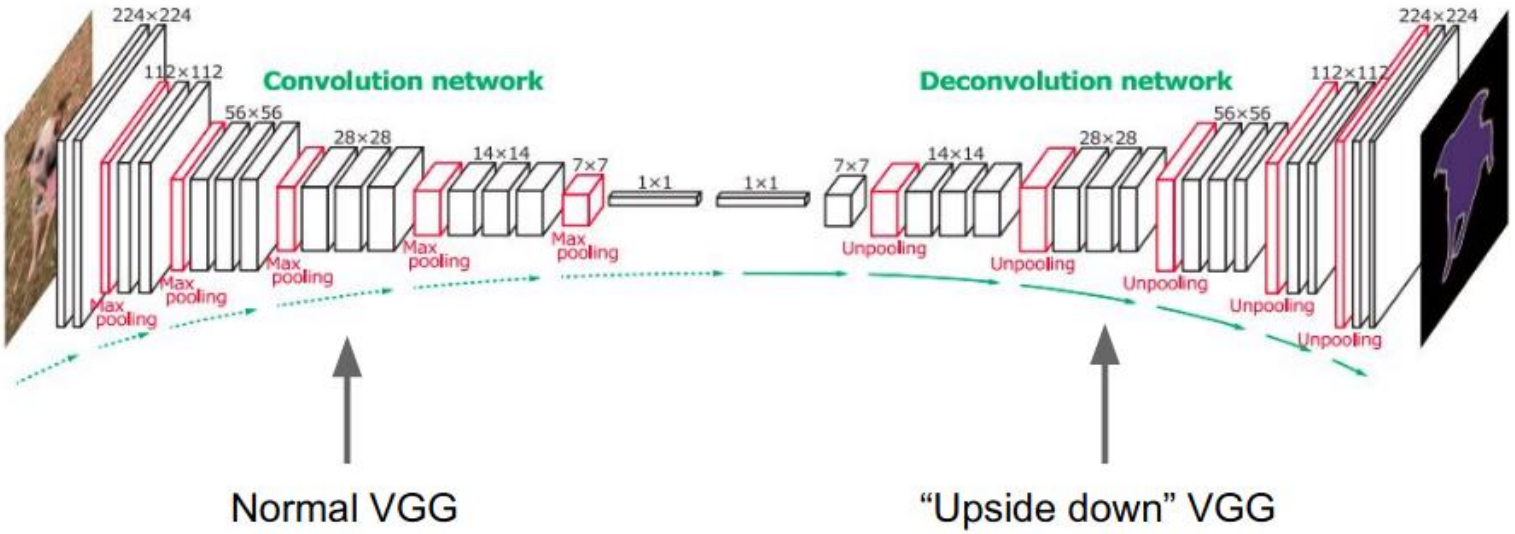
$$\vec{x} *^T \vec{a} = X^T \vec{a}$$

$$\begin{bmatrix} x & 0 & 0 & 0 \\ y & x & 0 & 0 \\ z & y & x & 0 \\ 0 & z & y & x \\ 0 & 0 & z & y \\ 0 & 0 & 0 & z \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} ax \\ ay + bx \\ az + by + cx \\ bz + cy + dx \\ dz + dy \\ dy \end{bmatrix}$$

When stride=1, convolution transpose is just a regular convolution (with different padding rules)

Upsampling

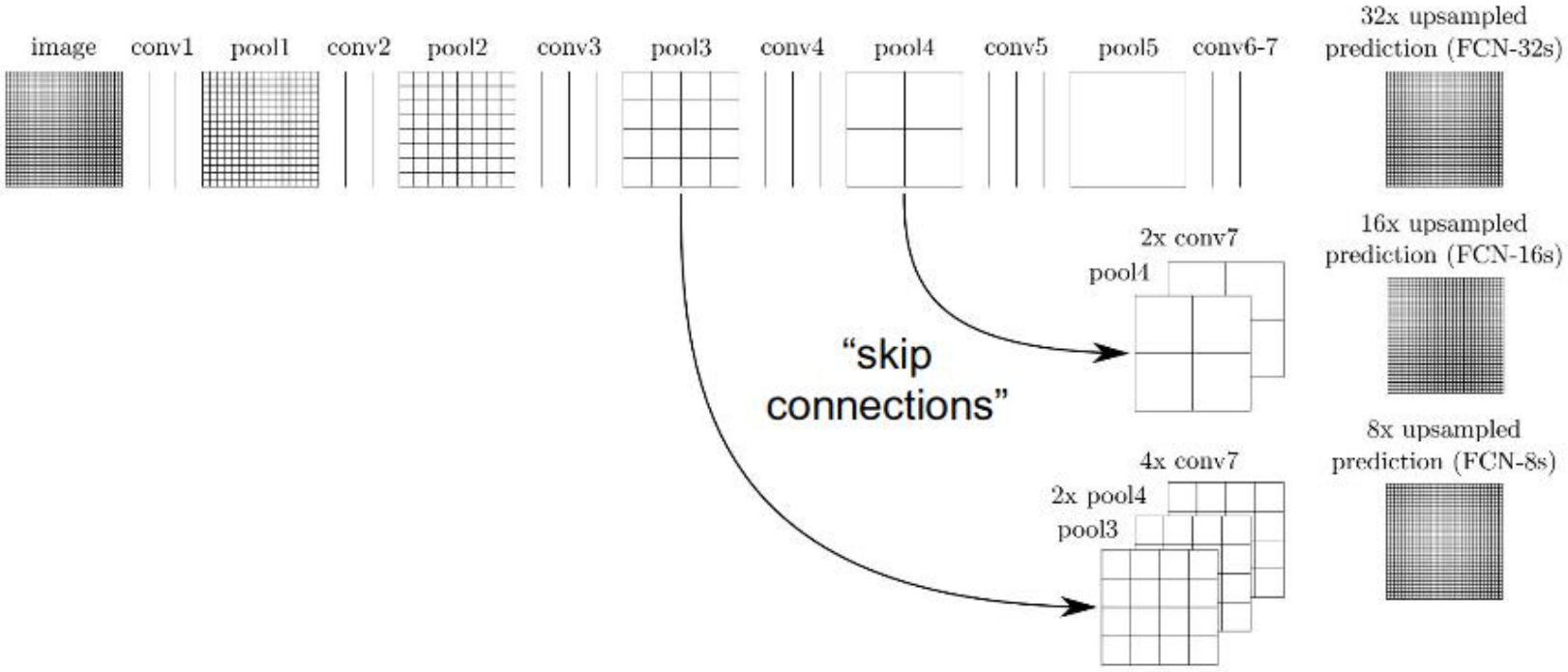
- Example:



Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

Upsampling

- For better performance preserving more details on the boundaries usually it is combined with skip connections



Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015

Causal Convolutions

- Connections in CNNs are sparse but units in deeper layers are connected to more of the input. At what rate does the effective receptive field size increase with depth?

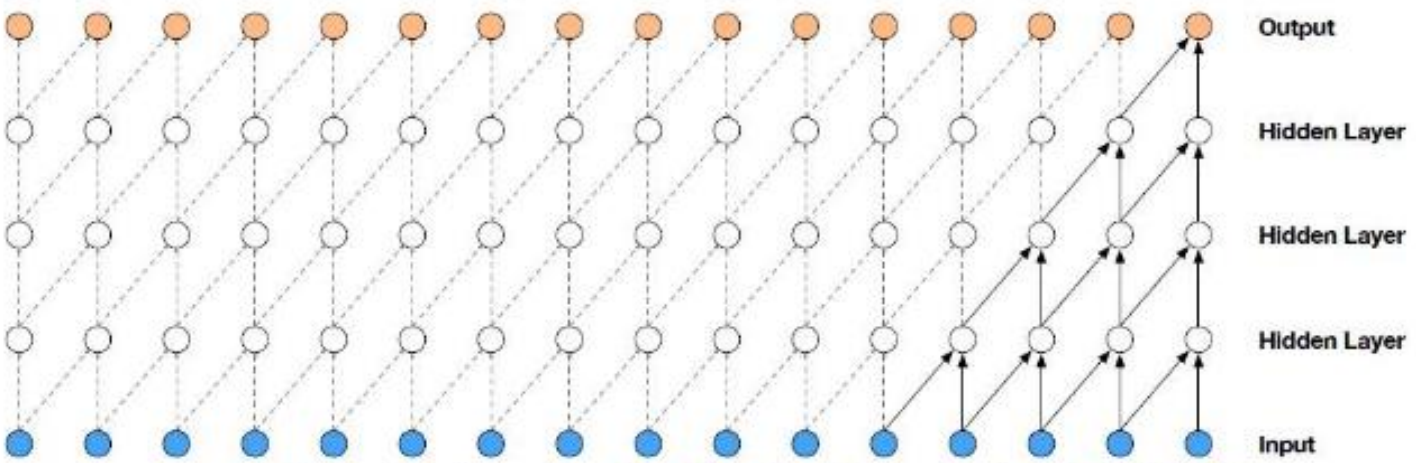


Image: *WaveNet: A Generative Model for Raw Audio*, Oord *et al.*, ICLR 2016

- The global context of convolutional neural networks grows too slow for some applications

UNET

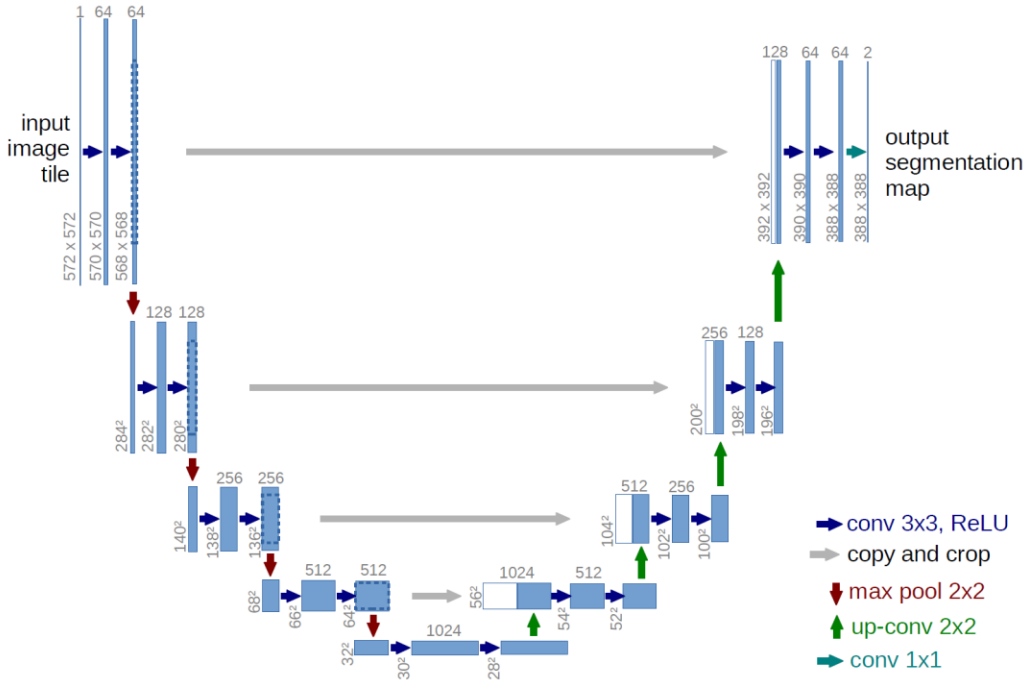
- Ronneberger et al. "U-Net: Convolutional Networks for Biomedical Image Segmentation"

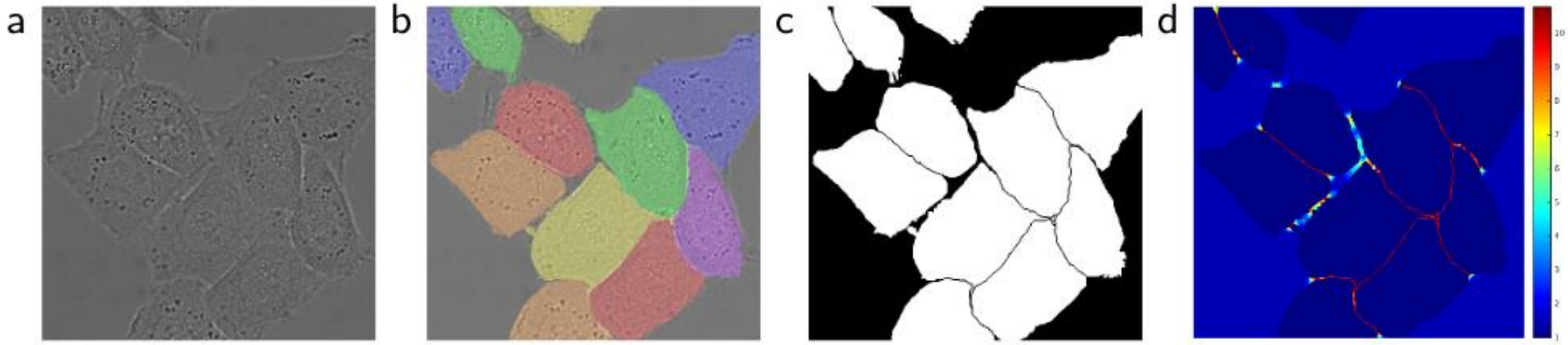
Loss function

$$E = \sum_{\mathbf{x}\in\Omega} w(\mathbf{x}) \log(p_{\ell(\mathbf{x})}(\mathbf{x}))$$

Weighted strategy

$$w(\mathbf{x}) = w_c(\mathbf{x}) + w_0 \cdot \exp\left(-\frac{(d_1(\mathbf{x}) + d_2(\mathbf{x}))^2}{2\sigma^2}\right)$$



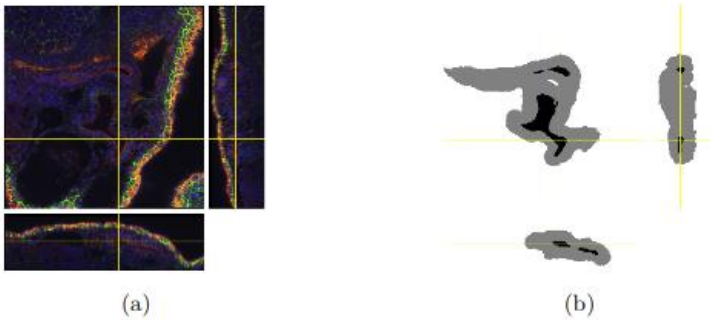


3D-UNET

- Cicek et al. "3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation"

Loss Function

$$E = \sum_{\mathbf{x} \in \Omega} w(\mathbf{x}) \log(p_{\ell(\mathbf{x})}(\mathbf{x}))$$



(a) (b)

Fig. 3: **(a)** The confocal recording of our 3rd *Xenopus* kidney. **(b)** Resulting dense segmentation from the proposed 3D u-net with batch normalization.

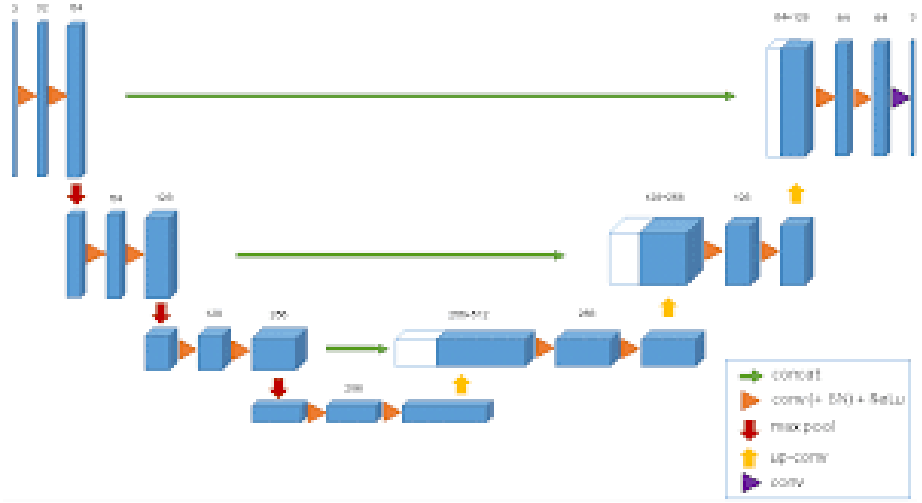


Table 1: Cross validation results for semi-automated segmentation (IoU)

test slices	3D w/o BN	3D with BN	2D with BN
subset 1	0.822	0.855	0.785
subset 2	0.857	0.871	0.820
subset 3	0.846	0.863	0.782
average	0.842	0.863	0.796

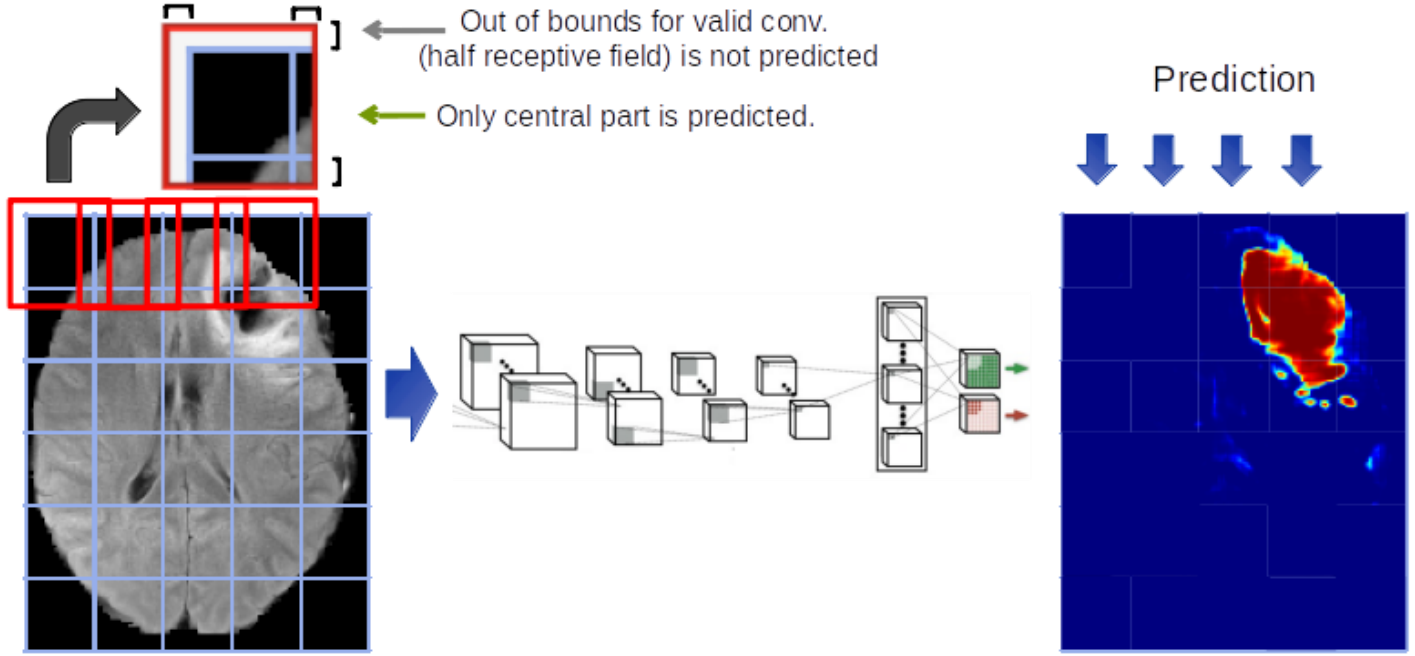
Table 2: Effect of # of slices for semi-automated segmentation (IoU)

GT slices	GT voxels	IoU S1	IoU S2	IoU S3
1,1,1	2.5%	0.331	0.483	0.475
2,2,1	3.3%	0.676	0.579	0.738
3,3,2	5.7%	0.761	0.808	0.835
5,5,3	8.9%	0.856	0.849	0.872

Table 3: Cross validation results for fully-automated segmentation (IoU)

test volume	3D w/o BN	3D with BN	2D with BN
1	0.655	0.761	0.619
2	0.734	0.798	0.698
3	0.779	0.554	0.325
average	0.723	0.704	0.547

Tiling for 3D networks



Bigger input tiles (or segments):

- Require more memory (FMs expand)
- But fewer redundant computations (less overlap)

Tile size during testing can be different from training

Dilated Convolutions

- Recall discrete convolution

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n)$$

- Dilated Convolution

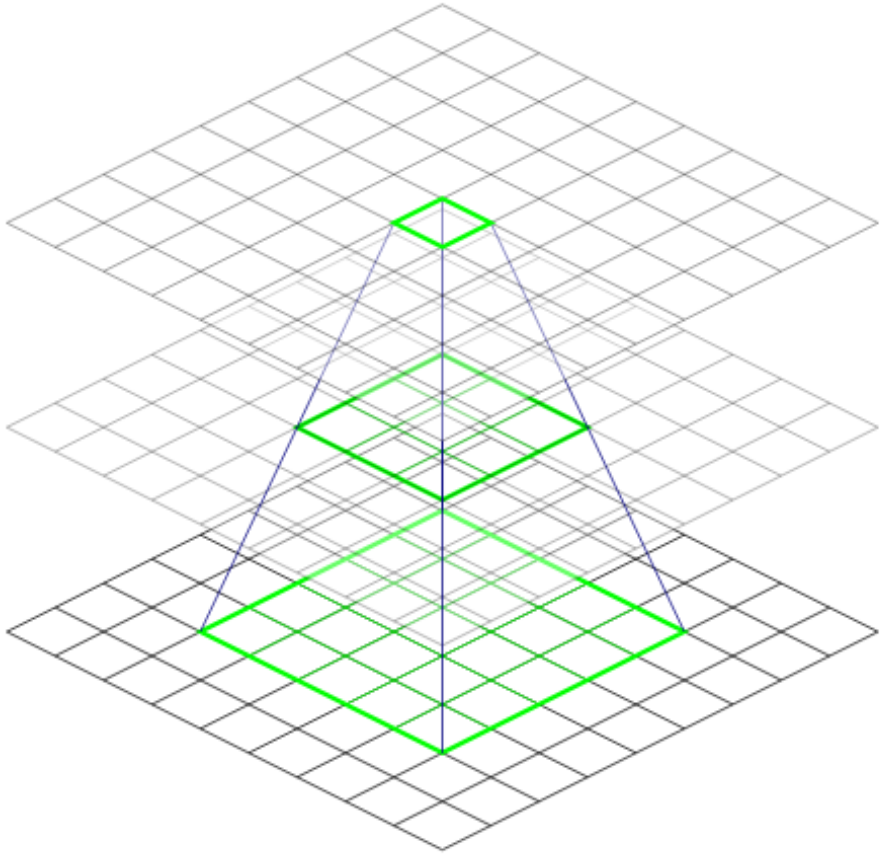
$$S(i,j) = (I *_l K)(i,j) = \sum_m \sum_n I(m,n)K(i-lm, j-ln)$$

L is a dilation factor

- Very old idea going to the 80s wavelet theory literature

Dilated Convolutions

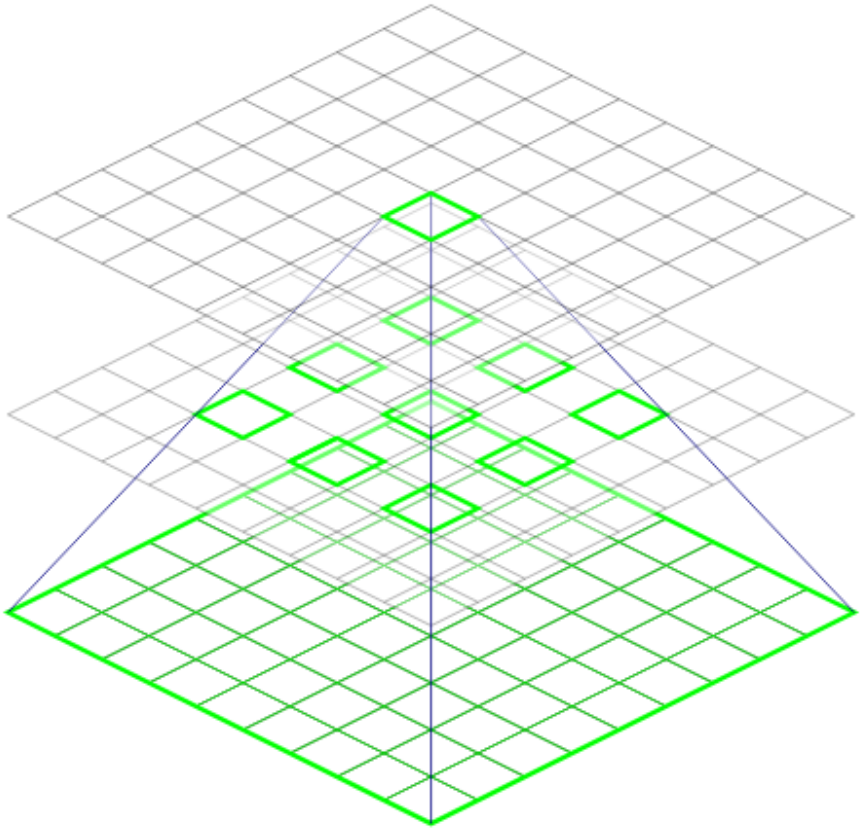
- Regular Convolution



The unit on the third layer has an effective receptive field of size 5x5

Dilated Convolutions

- Dilated Convolution



The unit on the third layer has an effective receptive field of size 9x9

Dilated Convolutions

- Dilated Convolution

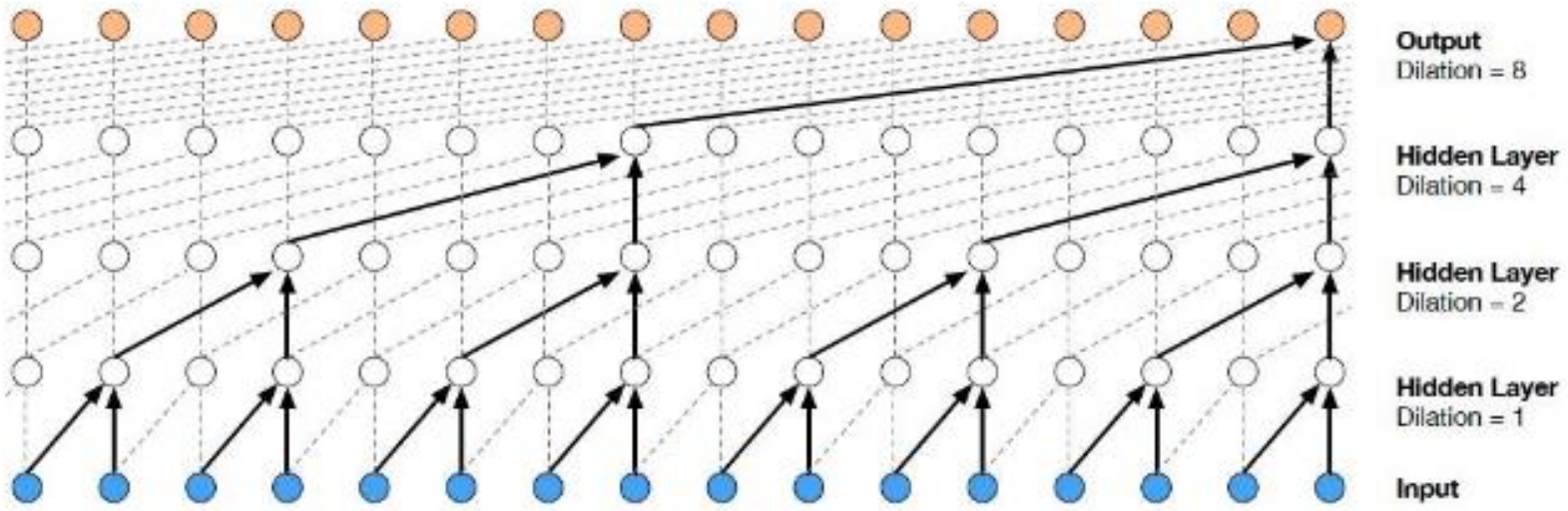
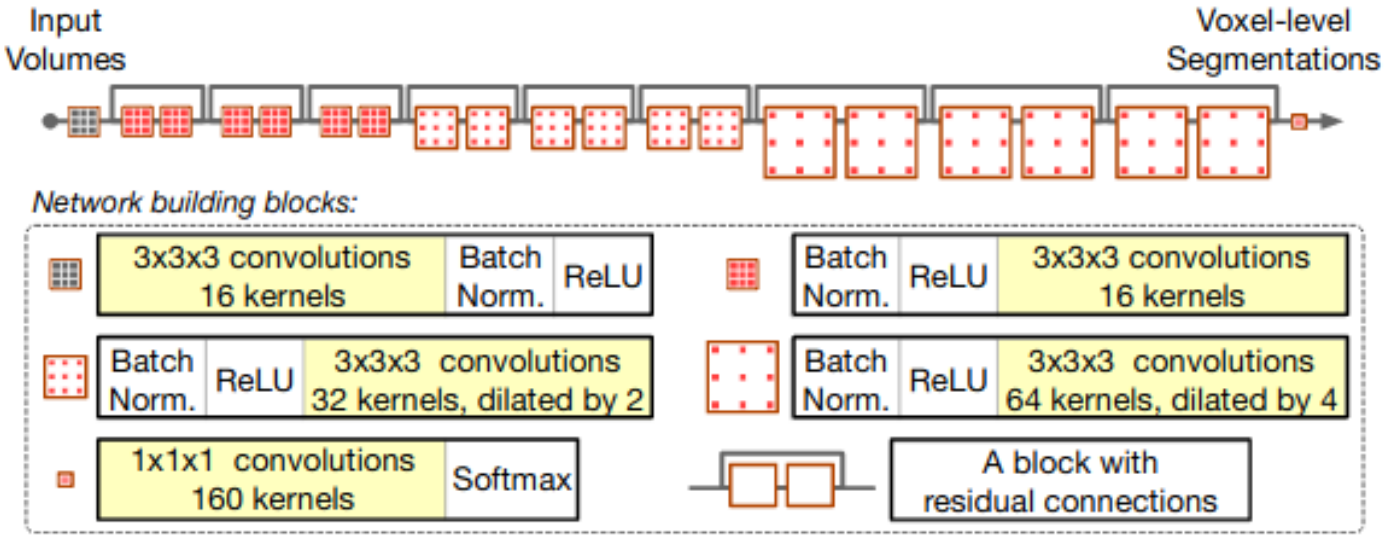
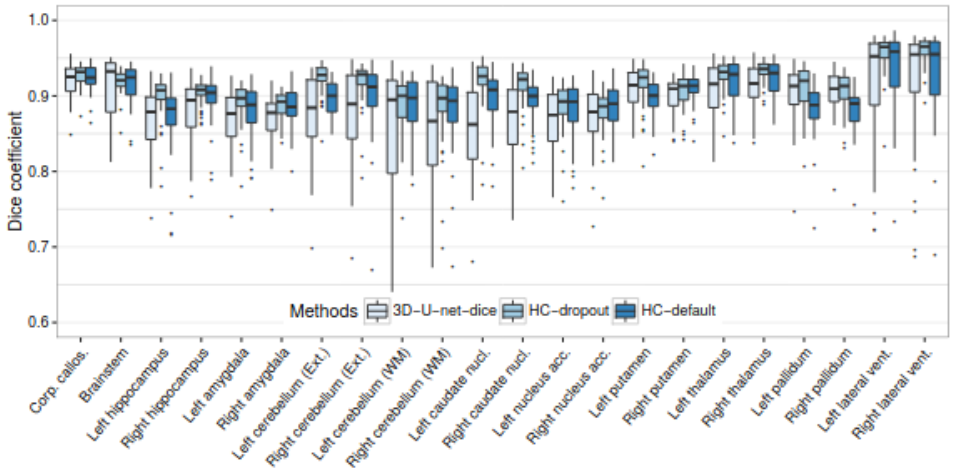


Image: *WaveNet: A Generative Model for Raw Audio*, Oord *et al.*, ICLR 2016

HighRes3DNet

- Li et. al, "On the Compactness, Efficiency, and Representation of 3D Convolutional Networks: Brain Parcellation as a Pretext Task", IPMI 2017





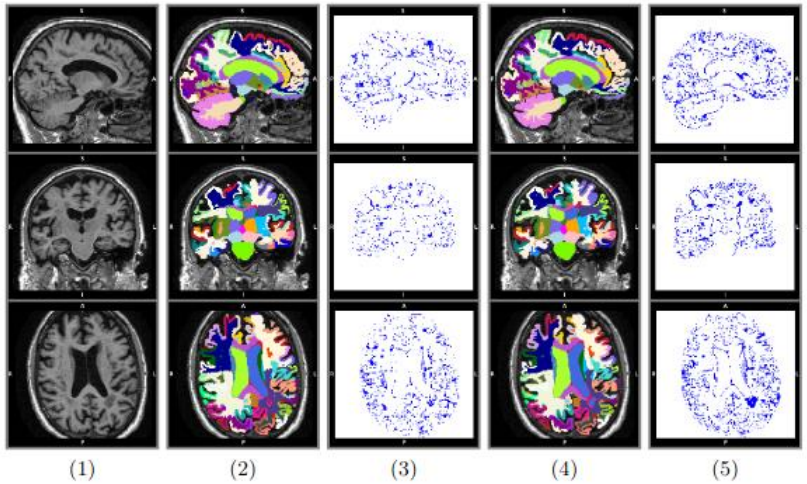
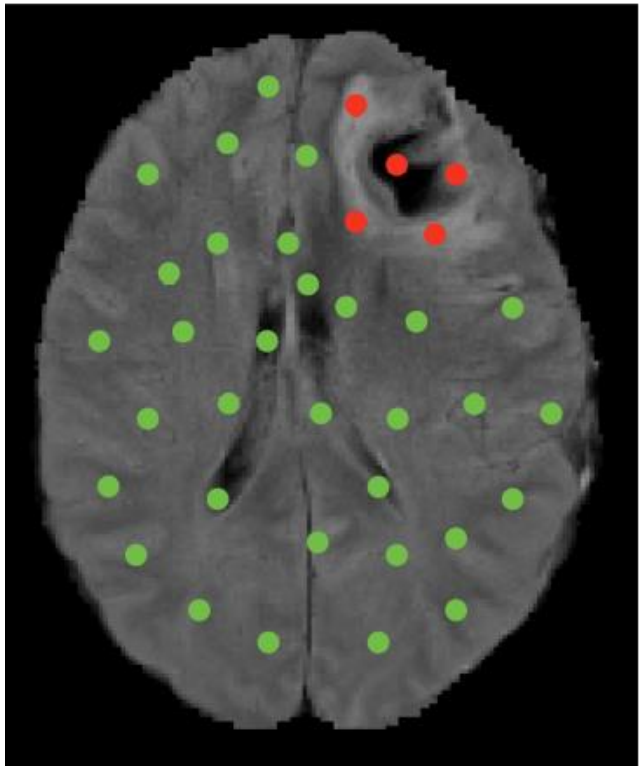


Fig. 4. Visualisations of segmentation results. (1) slices from a test image volume, segmentation maps and false prediction maps generated by HC-dropout (2, 3), and 3D U-net-dice (4, 5).

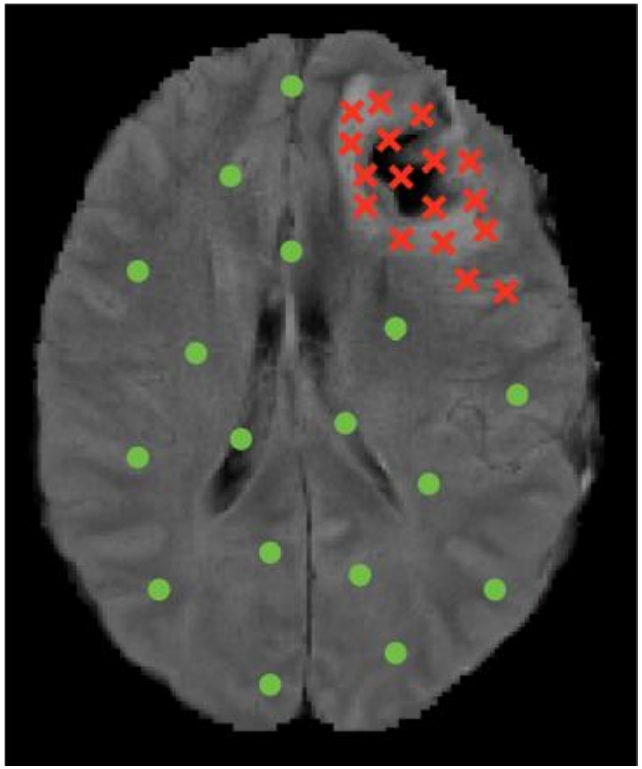
Loss Functions

Class Imbalance and Sampling

Uniform sampling



Weighted sampling: 50/50

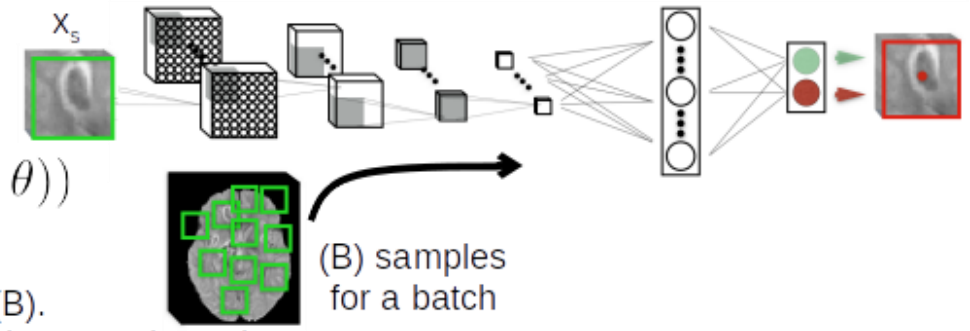


Training

Training with single predictions on each sample:

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

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



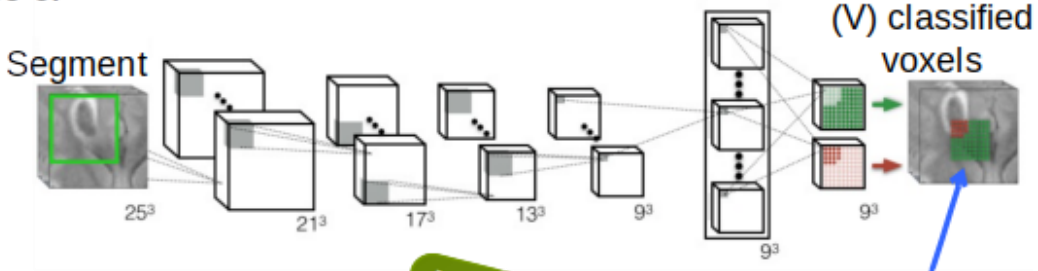
Cross Entropy over all the samples of a batch (B).
x_s a training sample (patch), y_s the true label of its central voxel.
$f_c(x)$ the CNN's confidence for predicting class **c**.
[.] the indicator function.

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 (eg if loading V separate patches).

Additional Losses

- **Dice Loss** [Milletari et al., 2016]

$$D = \frac{2\sum_i^N p_i g_i}{\sum_i^N p_i^2 + \sum_i^N g_i^2} \qquad \frac{\partial D}{\partial p_j} = 2\left[\frac{g_j\left(\sum_i^N p_i^2 + \sum_i^N g_i^2\right) - 2p_j\left(\sum_i^N p_i g_i\right)}{\left(\sum_i^N p_i^2 + \sum_i^N g_i^2\right)^2}\right]$$

g_n the value of segmentation at n and p_n the predicted probabilistic map

Additional Losses

- **Dice Loss** [Milletari et al., 2016]

$$D = \frac{2\sum_i^N p_i g_i}{\sum_i^N p_i^2 + \sum_i^N g_i^2} \qquad \frac{\partial D}{\partial p_j} = 2\left[\frac{g_j\left(\sum_i^N p_i^2 + \sum_i^N g_i^2\right) - 2p_j\left(\sum_i^N p_i g_i\right)}{\left(\sum_i^N p_i^2 + \sum_i^N g_i^2\right)^2}\right]$$

g_n the value of segmentation at n and p_n the predicted probabilistic map

- **Generalized Dice Loss** [Sudre et al., 2017]

$$\mathrm{GDL} = 1 - 2\frac{\sum_{l=1}^2 w_l \sum_n r_{ln} p_{ln}}{\sum_{l=1}^2 w_l \sum_n r_{ln} + p_{ln}}, \qquad \frac{\partial \mathrm{GDL}}{\partial p_i} = -2\frac{(w_1^2 - w_2^2)\left[\sum_{n=1}^N p_n r_n - r_i \sum_{n=1}^N (p_n + r_n)\right] + N w_2 (w_1 + w_2)(1 - 2r_i)}{\left[(w_1 - w_2)\sum_{n=1}^N (p_n + r_n) + 2N w_2\right]^2}$$

r_n the value of segmentation at n, p_n the predicted probabilistic map, $w_l = 1/\Sigma(r_{ln})^2$

Used to deal with the correlation that exists between the size of the segment and the result of the dice!

Additional Losses

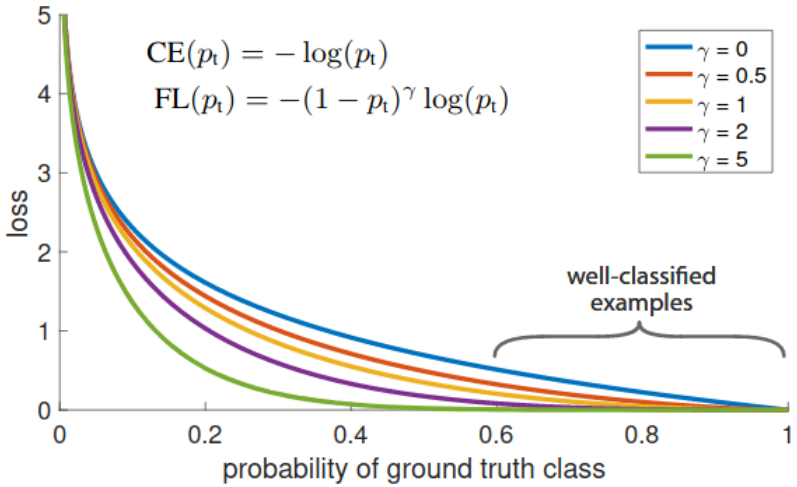
- Most of the losses balance the importance of positive/negative examples without integrating the importance between easy/ hard examples.
- **Focal Loss** [Lin et al. 2018]
  - Propose to reshape the loss function to down-weight easy examples and thus focus training on hard negatives.

Binary Cross entropy

$$\mathrm{CE}(p, y) = \begin{cases} -\log(p) & \text{if } y = 1 \\ -\log(1-p) & \text{otherwise.} \end{cases}$$

Focal Loss

$$\mathrm{FL}(p_t) = -(1-p_t)^{\gamma} \log(p_t).$$



Where $\gamma \geq 0$.

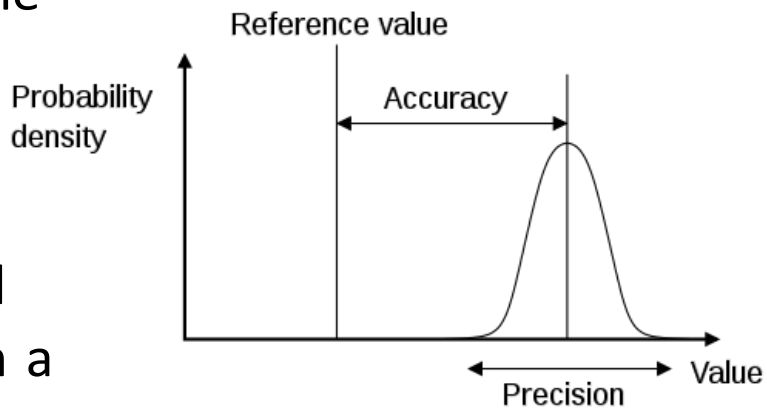
when an example is misclassified and p is small

the modulating factor is near 1 and the loss is unaffected. As p -> 1, the factor goes to 0 and the loss of the well-classified examples is down-weighted

Evaluation Metrics

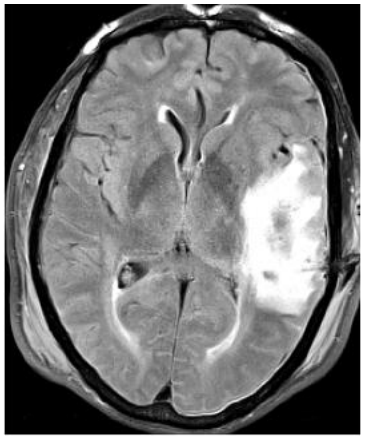
How to Assess Performance?

- **Precision**
  - is a description of random errors, a measure of statistical variability.
  - the repeatability, or reproducibility of the measurement
- **Accuracy**
  - More commonly, it is a description of systematic errors, a measure of statistical bias; as these cause a difference between a result and a "true" value, ISO calls this trueness.
- **Robustness**
  - Refers to the degradation in performance with respect to varying noise levels or other imaging artefacts

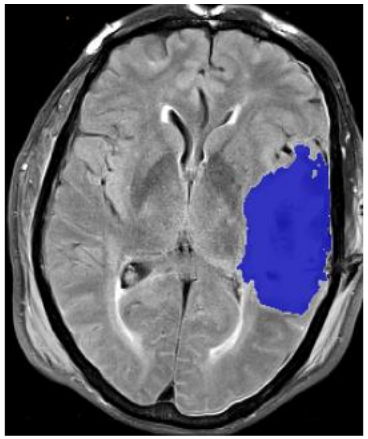


How to Assess Performance?

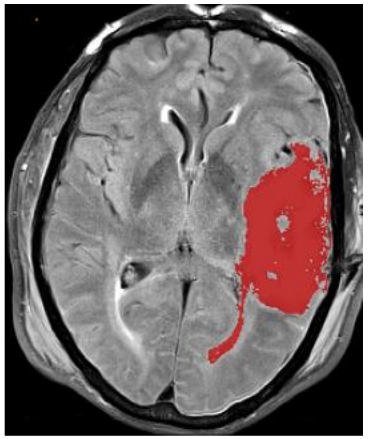
- Assume we have an automated segmentation algorithm, how do we assess "how good it is"?
- There are several ways of quantifying segmentation performance



MR image

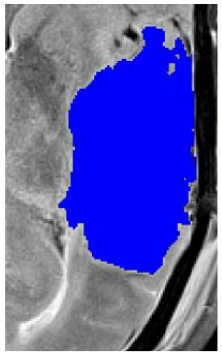


Gold standard

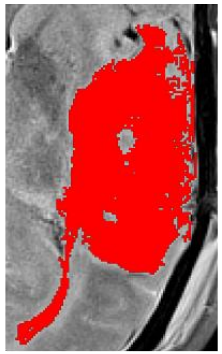


Auto Segmentation

MR image



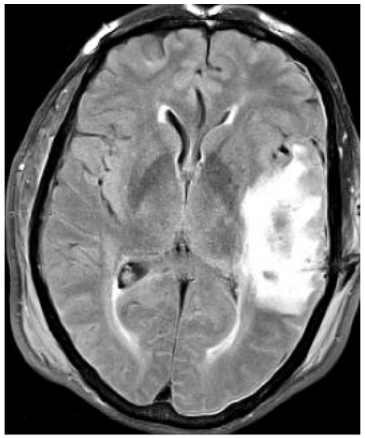
Gold standard



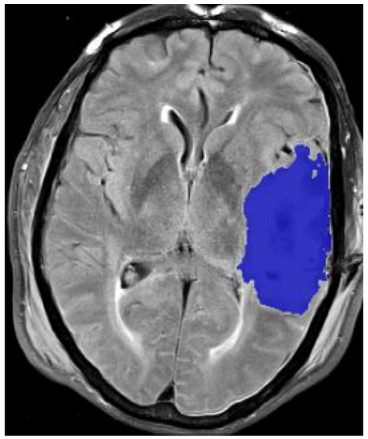
Auto Segmentation

How to Assess Performance?

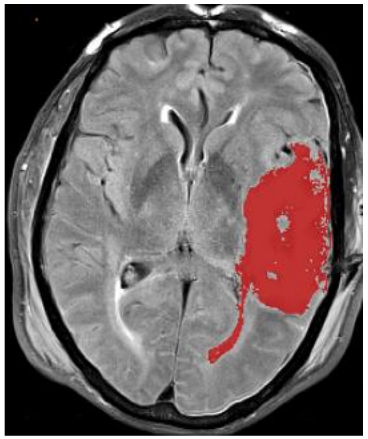
- Assume we have an automated segmentation algorithm, how do we assess "how good it is"?
- There are several ways of quantifying segmentation performance



MR image

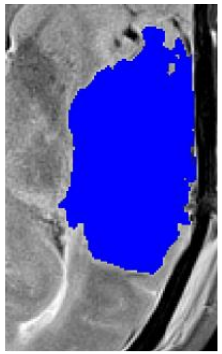


Gold standard

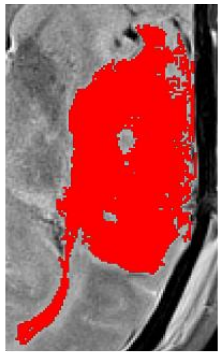


Auto Segmentation

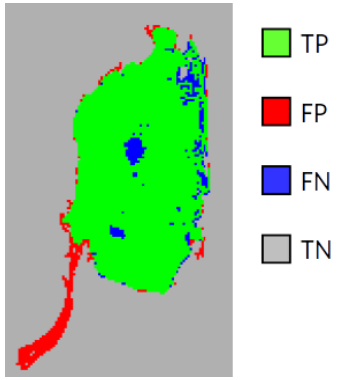
MR image

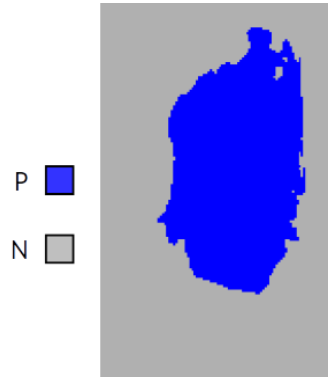


Gold standard



Auto Segmentation





Confusion Matrix

condition positive ()
the number of real positive cases in the data
condition negatives ()
the number of real negative cases in the data

		True condition	
	Total population	Condition positive	Condition negative
Predicted condition	Predicted condition positive	**True positive**	**False positive**, Type I error
	Predicted condition negative	**False negative**, Type II error	**True negative**

true positive ()
eqv. with hit
true negative ()
eqv. with correct rejection
false positive ()
eqv. with false alarm, Type I error
false negative ()
eqv. with miss, Type II error

Accuracy, Precision, Recall, ...

accuracy

$$ACC = \frac{TP+TN}{P+N},\ P = TP + FN,\ N = TN + FP$$

precision or **positive predictive value**

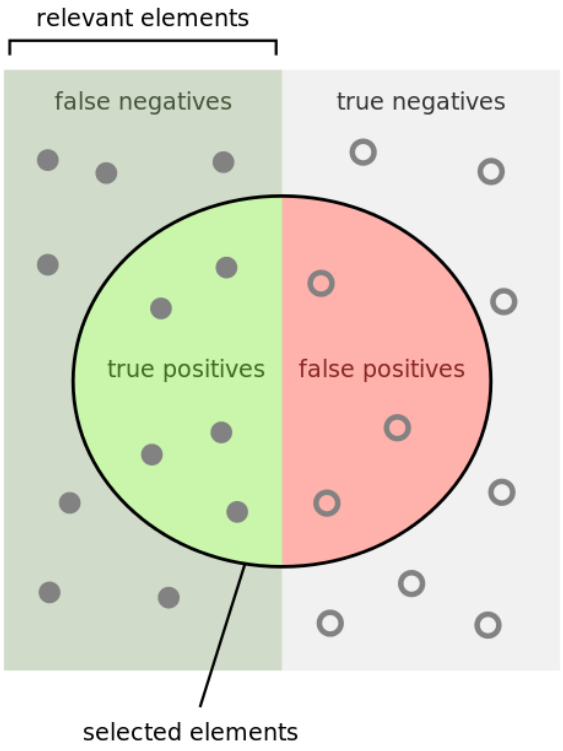
$$PPV = \frac{TP}{TP + FP}$$

recall, sensitivity, hit rate or **true positive rate**

$$TPR = \frac{TP}{P} = \frac{TP}{TP + FN}$$

specificity or **true negative rate**

$$TNR = \frac{TN}{N} = \frac{TN}{TN + FP}$$



How many selected items are relevant?

Precision =

How many relevant items are selected?

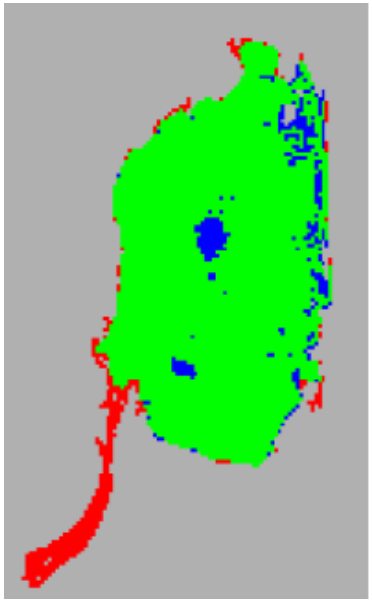
Recall =

F1 score

F1 score
is the harmonic mean of precision and recall

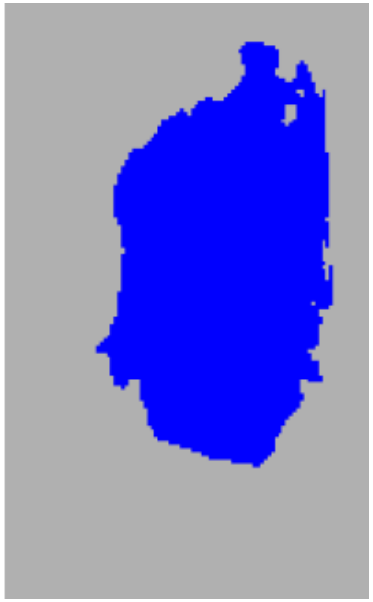
$$F_1 = 2\frac{PPV \cdot TPR}{PPV + TPR} = \frac{2TP}{2TP + FP + FN}$$

precision $PPV = \frac{TP}{TP+FP}$ **recall** $TPR = \frac{TP}{TP+FN}$



TP
FP
FN
TN

P
N



Dice Similarity Coefficient

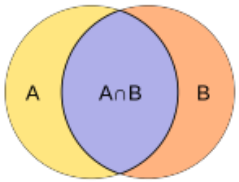
- Most widely used measure for evaluating segmentation
- Assume is the reference, and the prediction

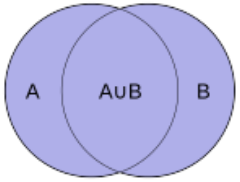
$$DSC = \frac{2|A \cap B|}{|A| + |B|}$$

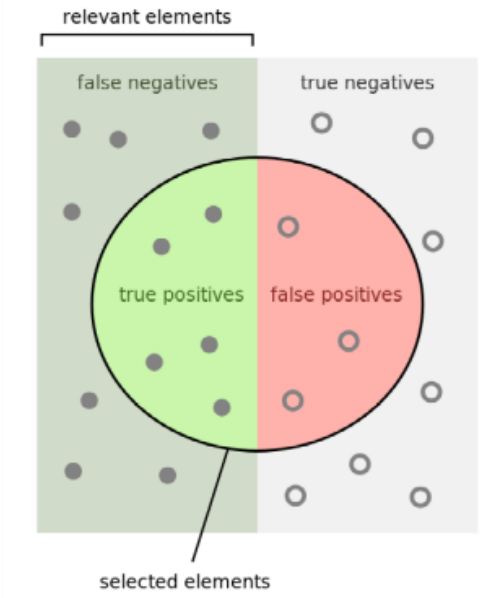
- $|A| = TP + FN$
- $|B| = TP + FP$
- $|A \cap B| = TP$

$$DSC = \frac{2TP}{2TP + FP + FN} = F_1$$

DSC is equivalent to F1 score!







Other Measures

Tana & Hanbury: "Metrics for evaluating 3D medical image segmentation: analysis, selection, and tool"

volume similarity

$$VS = 1 - \frac{\big||A| - |B|\big|}{|A| + |B|} = 1 - \frac{|FN - FP|}{2TP + FP + FN}$$

surface distance measures

- **Hausdorff distance**

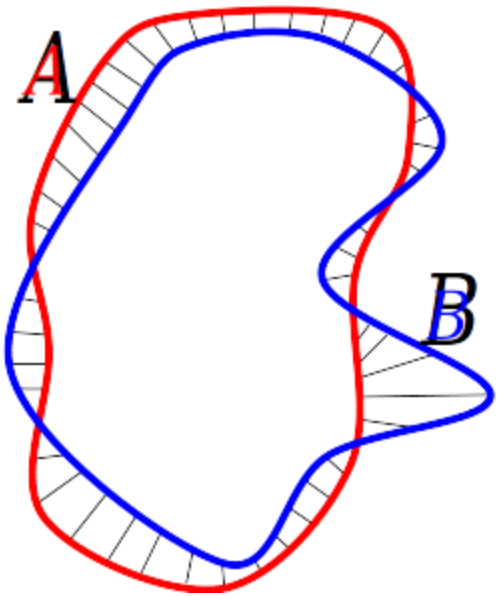
$$HD = \max\big(h(A,B), h(B,A)\big)$$

$$h(A,B) = \max_{a \in A} \min_{b \in B} \|a - b\|$$

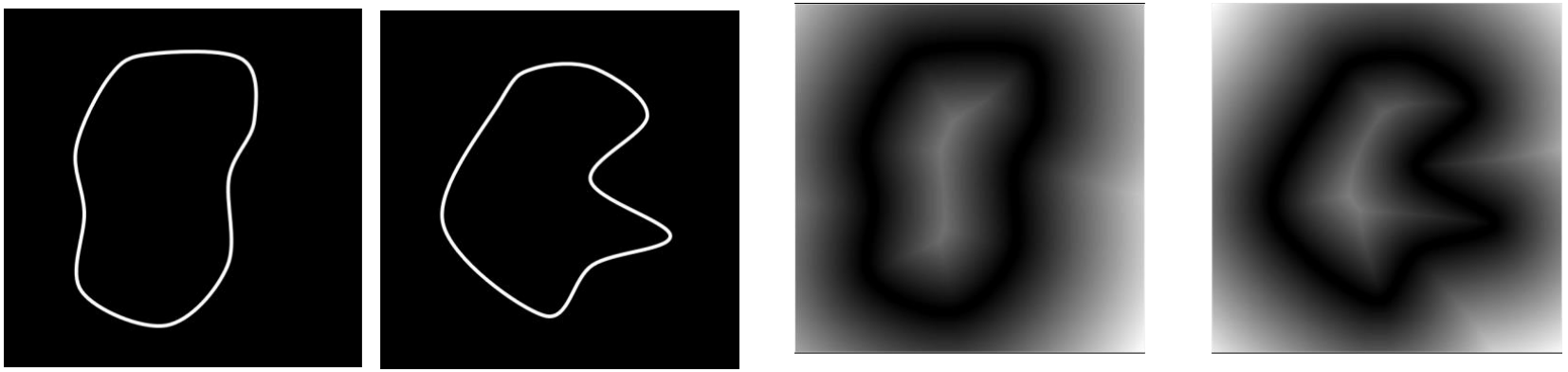
- **(symmetric) average surface distance**

$$ASD = \frac{d(A,B) + d(B,A)}{2}$$

$$d(A,B) = \frac{1}{N} \sum_{a \in A} \min_{b \in B} \|a - b\|$$

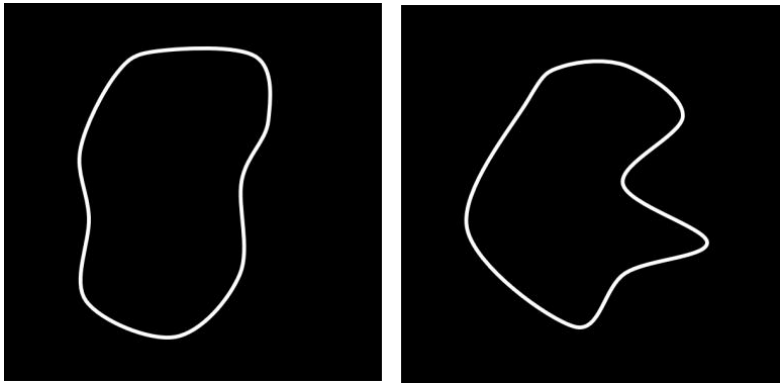


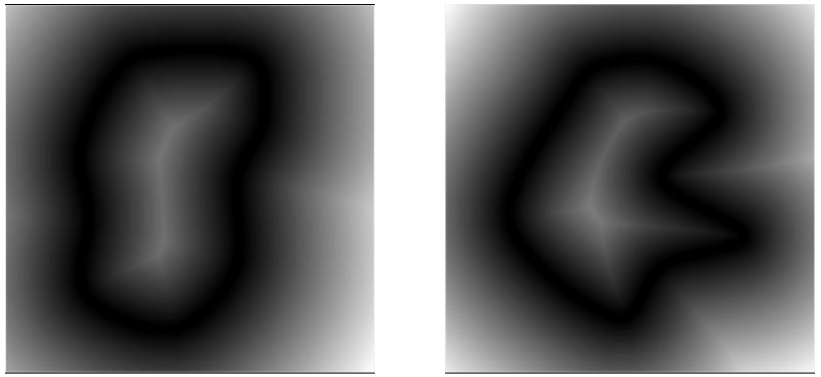
Surface Distance



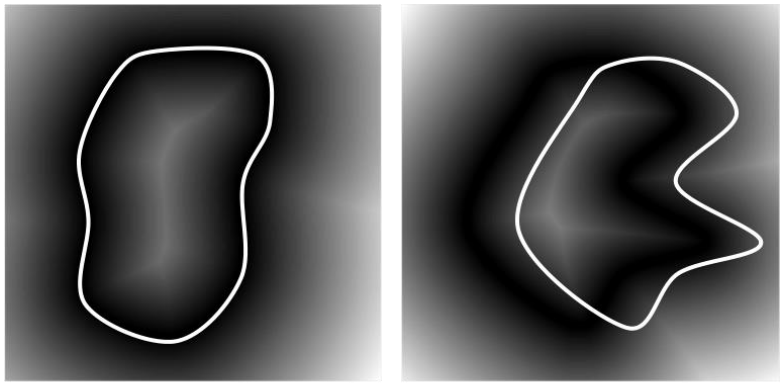
Euclidean Distance Maps

Surface Distance

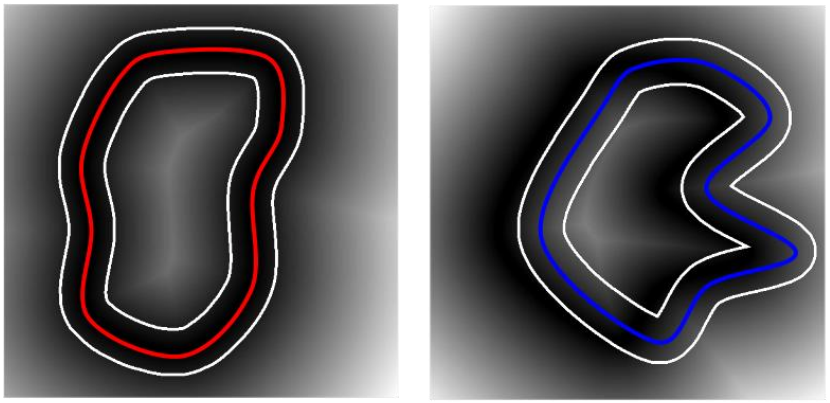




Euclidean Distance Maps

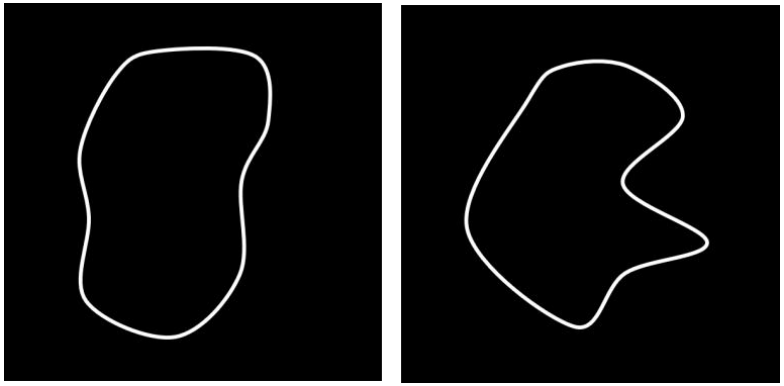


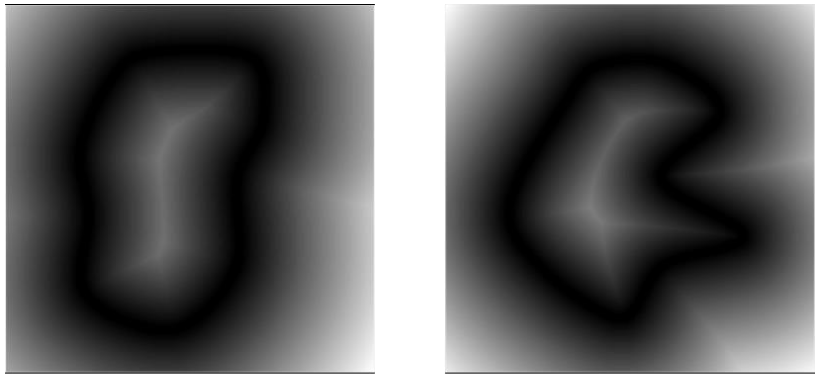
Zero-Level Surface



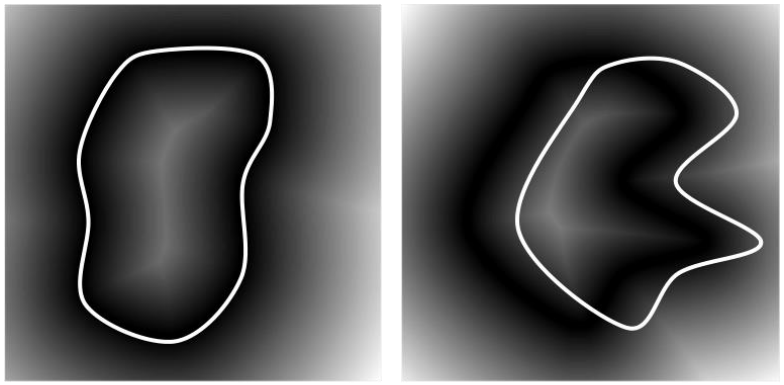
30-Pixels Distance Surface

Surface Distance

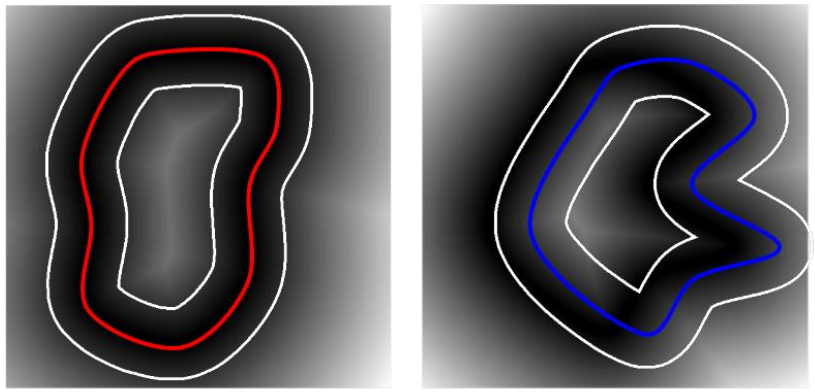




Euclidean Distance Maps

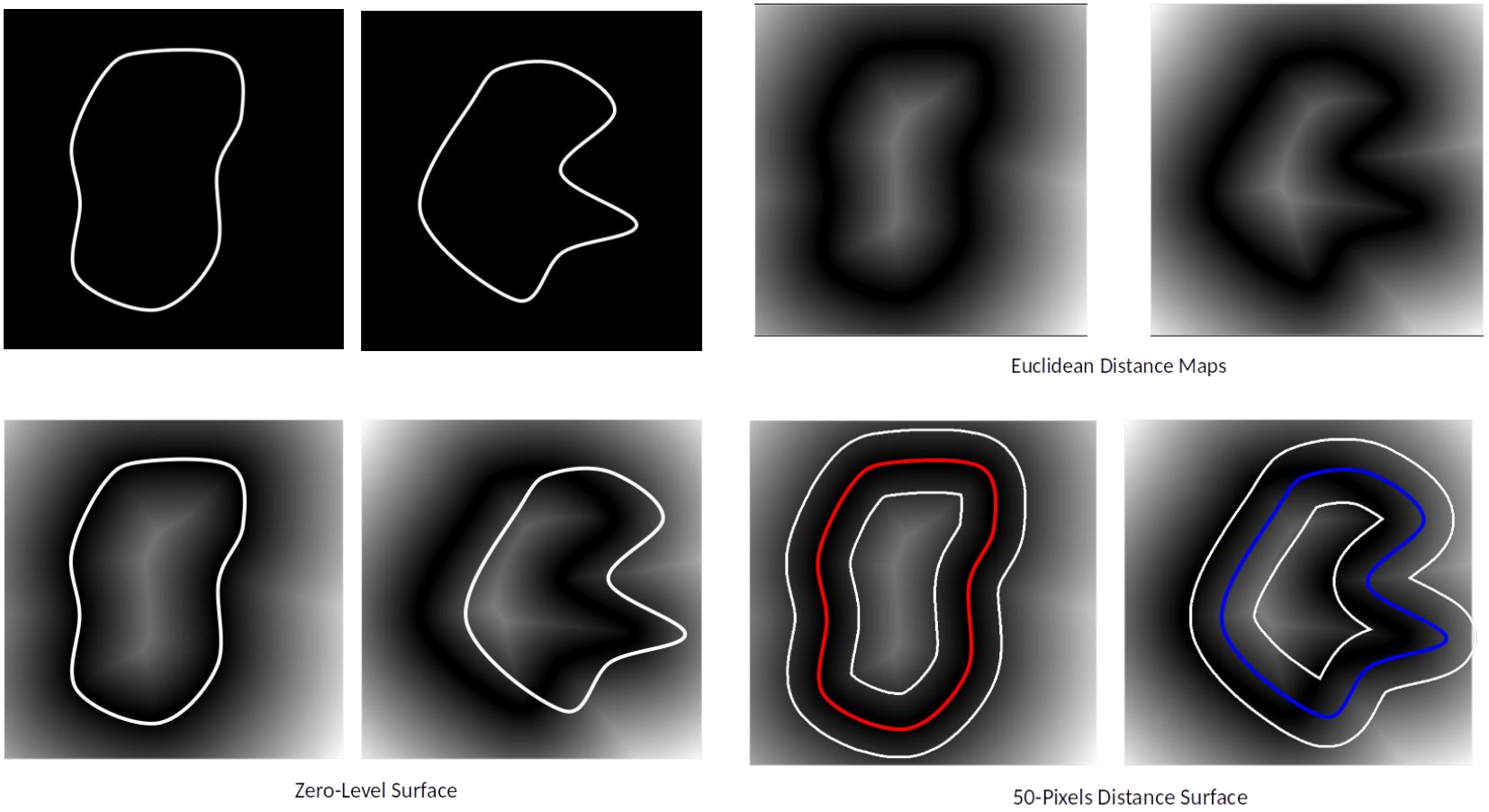


Zero-Level Surface



50-Pixels Distance Surface

Surface Distance

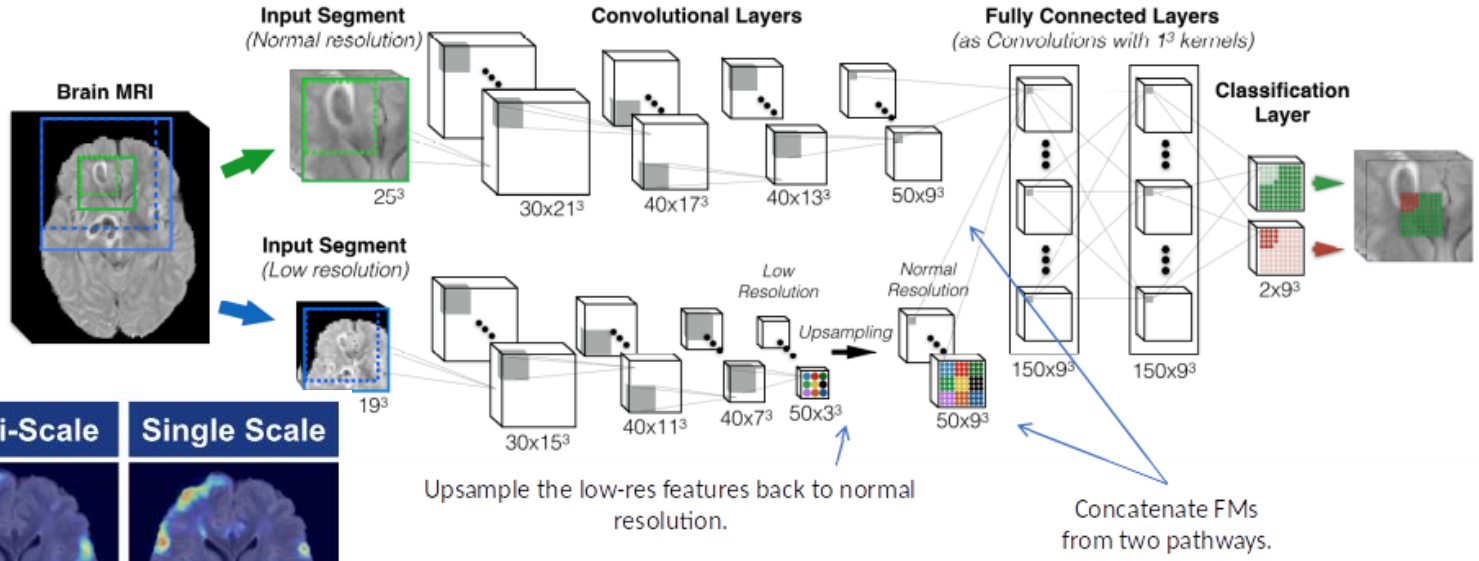


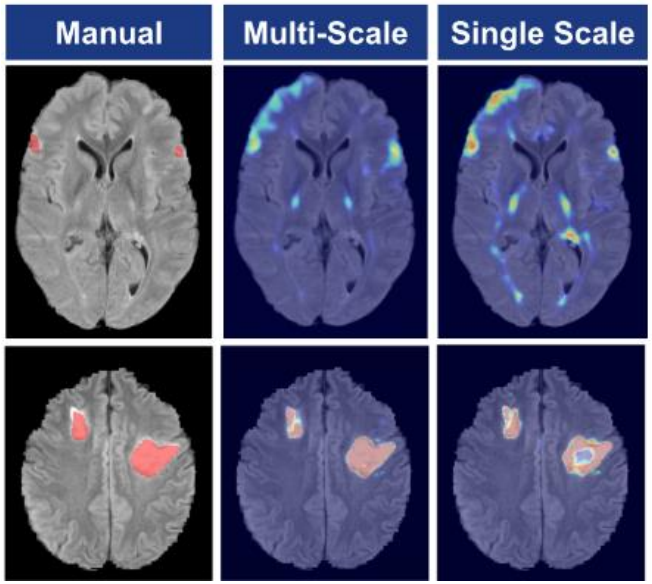
Sum up distances along pixels on the boundaries of one contour overlaid on the distance map of the other.

More Papers

Multi-Scale Processing

- How can we make the network to "see" more context?
- **Idea:** Add more pathways which process downsampled images

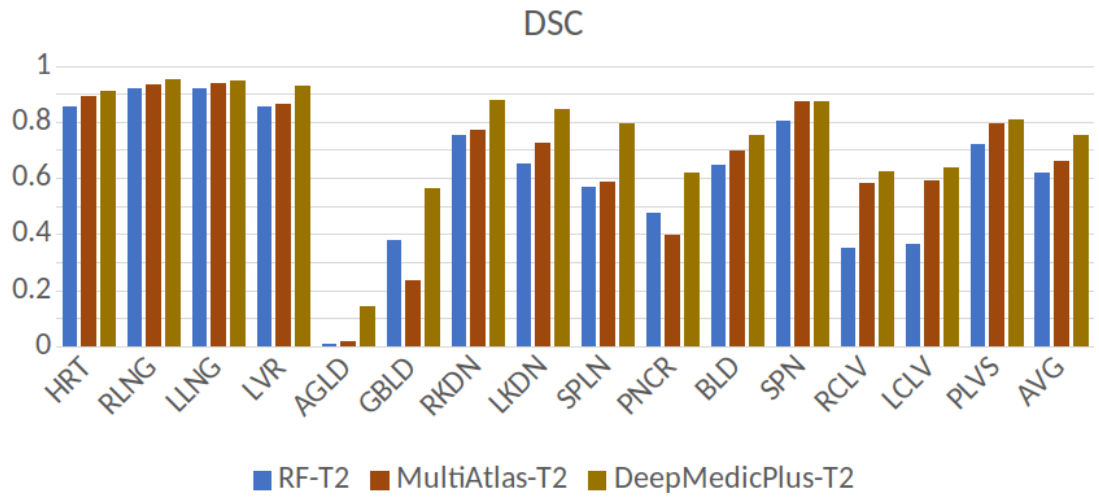


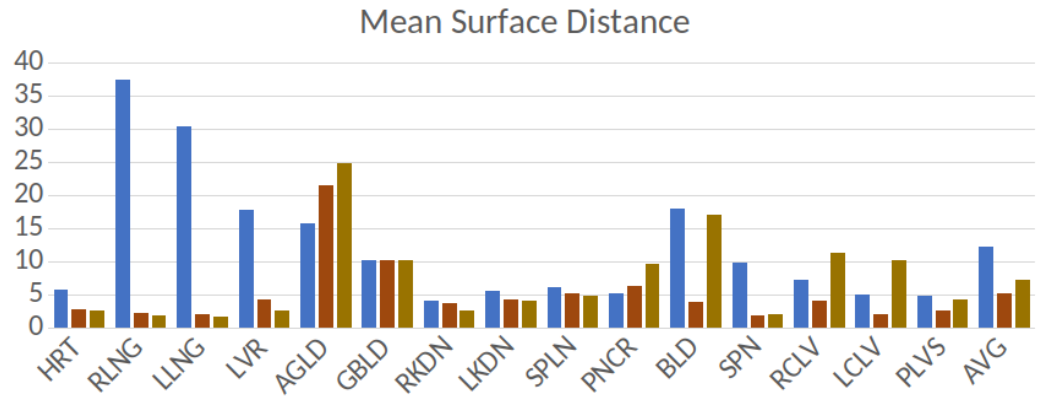


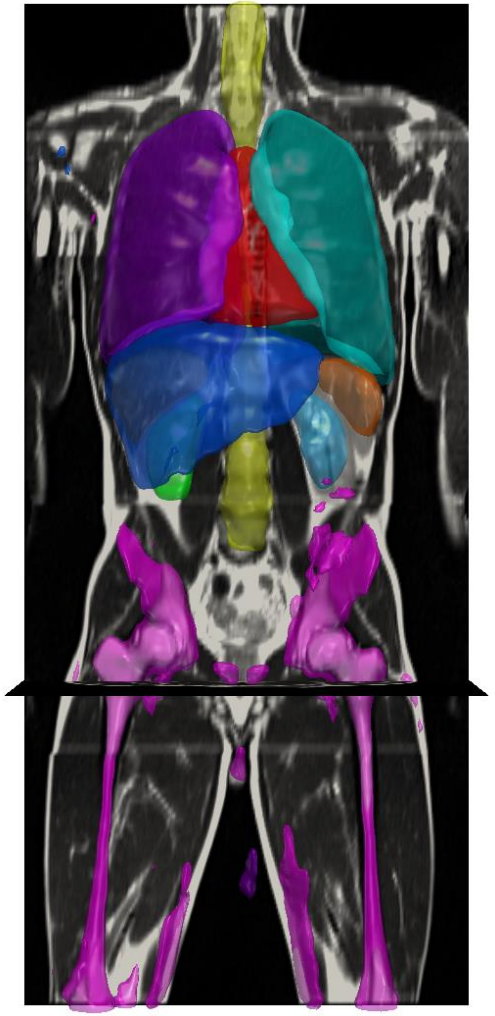
Kamnitsas et al, "Efficient multi-scale 3D CNN with fully connected CRF for accurate brain lesion segmentation", MedIA, 2017
Source code: https://github.com/Kamnitsask/deepmedic

Multi-Organ Segmentation

Lavdas et al. "Fully automatic, multiorgan segmentation in normal whole body magnetic resonance imaging (MRI), using classification forests (CFs), convolutional neural networks (CNNs), and a multi-atlas (MA) approach"

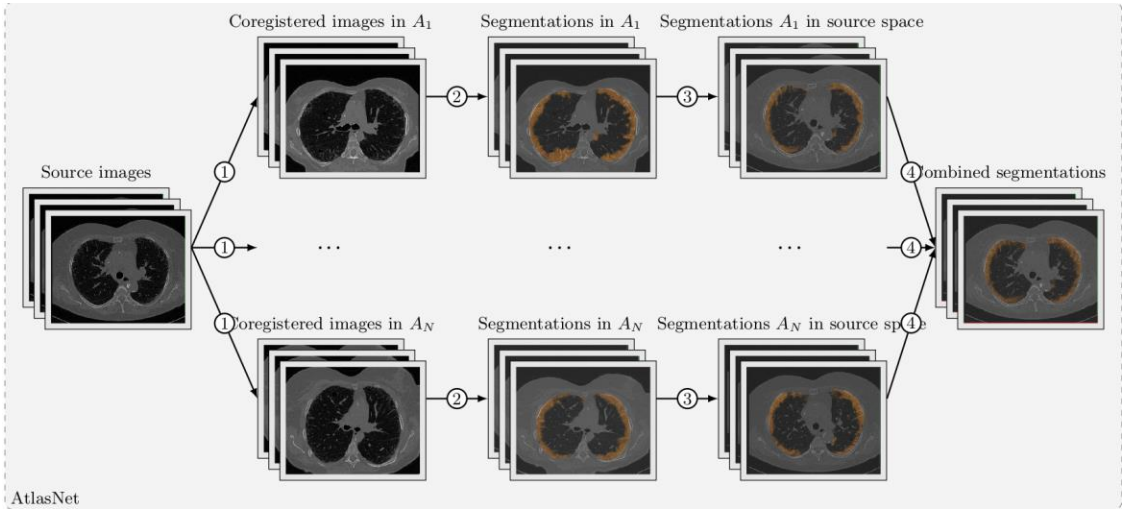


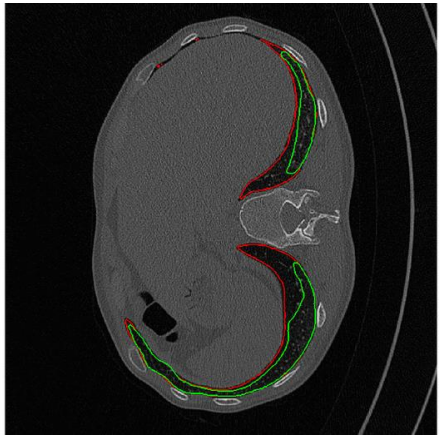


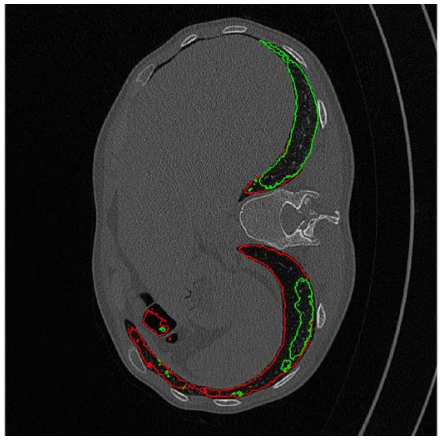


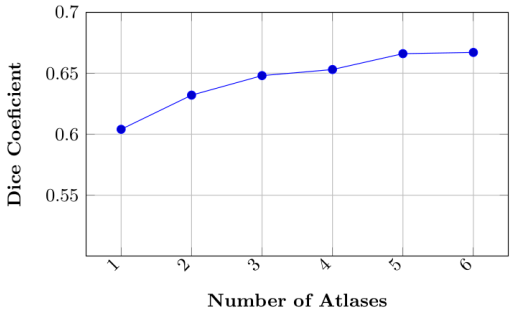
AtlasNet

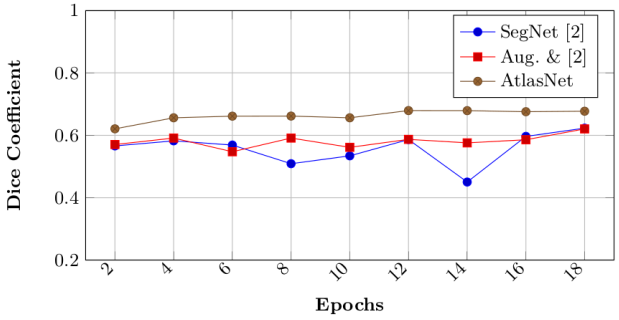
Vakalopoulou et al. "AtlasNet: Multi-atlas Non-linear Deep Networks for Medical Image Segmentation"











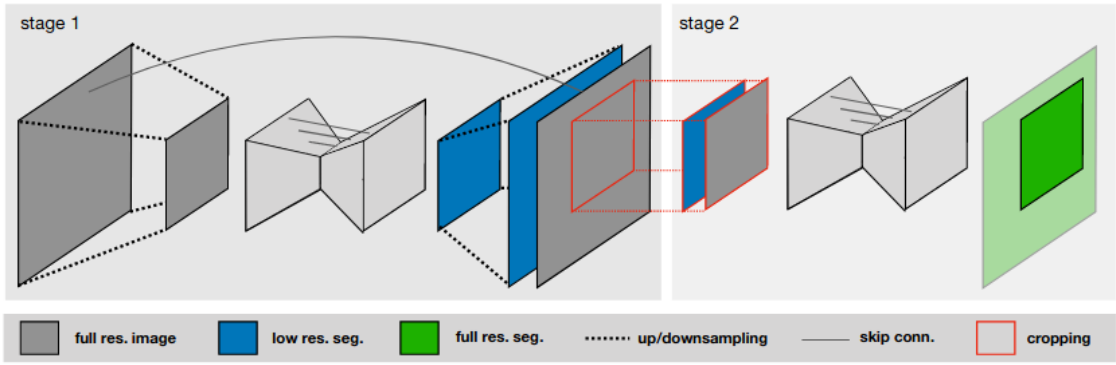
nnU-Net

Isensee et al. "nnU-Net: Self-adapting Framework for U-Net-Based Medical Segmentation"

$$\mathcal{L}_{total} = \mathcal{L}_{dice} + \mathcal{L}_{CE}$$

		2D U-Net	3D U-Net	3D U-Net lowres
BrainTumour	median patient shape	169x138	138x169x138	-
	input patch size	192x160	128x128x128	-
	batch size	89	2	-
	num pool per axis	5, 5	5, 5, 5	-
Heart	median patient shape	320x232	115x320x232	58x160x116
	input patch size	320x256	80x192x128	64x160x128
	batch size	33	2	2
	num pool per axis	6, 6	4, 5, 5	4, 5, 5
Liver	median patient shape	512x512	482x512x512	121x128x128
	input patch size	512x512	128x128x128	128x128x128
	batch size	10	2	2
	num pool per axis	6, 6	5, 5, 5	5, 5, 5
Hippocampus	median patient shape	50x35	36x50x35	-
	input patch size	56x40	40x56x40	-
	batch size	366	9	-
	num pool per axis	3, 3	3, 3, 3	-
Prostate	median patient shape	320x319	20x320x319	-
	input patch size	320x320	20x192x192	-
	batch size	26	4	-
	num pool per axis	6, 6	2, 5, 5	-
Lung	median patient shape	512x512	252x512x512	126x256x256
	input patch size	512x512	112x128x128	112x128x128
	batch size	10	2	2
	num pool per axis	6, 6	4, 5, 5	4, 5, 5
Pancreas	median patient shape	512x512	96x512x512	96x256x256
	input patch size	512x512	96x160x128	96x160x128
	batch size	10	2	2
	num pool per axis	6, 6	4, 5, 5	4, 5, 5

label	BrainTumour 1	BrainTumour 2	BrainTumour 3	Heart 1	Liver 1	Liver 2	Hippoc. 1	Hippoc. 2	Prostate 1	Prostate 2	Lung 1	Pancreas 1	Pancreas 2
2D U-Net	78.60	58.65	77.42	91.36	94.37	53.94	88.52	86.70	61.98	84.31	52.68	74.70	35.41
3D U-Net	**80.71**	**62.22**	**79.07**	92.45	94.11	61.74	**89.87**	**88.20**	60.77	83.73	55.87	77.69	42.69
3D U-Net stage1 only (U-Net Cascade)	-	-	-	90.63	94.69	47.01	-	-	-	-	65.33	79.45	49.65
3D U-Net (U-Net Cascade)	-	-	-	92.40	95.38	58.49	-	-	-	-	**66.85**	**79.30**	**52.12**
ensemble 2D U-Net+ 3D U-Net	80.79	61.72	79.16	**92.70**	94.30	60.24	89.78	88.09	**63.78**	**85.31**	55.96	78.26	40.46
ensemble 2D U-Net+ 3D U-Net (U-Net Cascade)	-	-	-	92.64	95.31	60.09	-	-	-	-	61.18	78.79	45.46
ensemble 3D U-Net+ 3D U-Net (U-Net Cascade)	-	-	-	92.63	**95.43**	**61.82**	-	-	-	-	65.16	79.70	49.14
test set	67.71	**47.73**	**68.16**	**92.77**	**95.24**	**73.71**	**90.37**	**88.95**	**75.81**	**89.59**	**69.20**	**79.53**	**52.27**



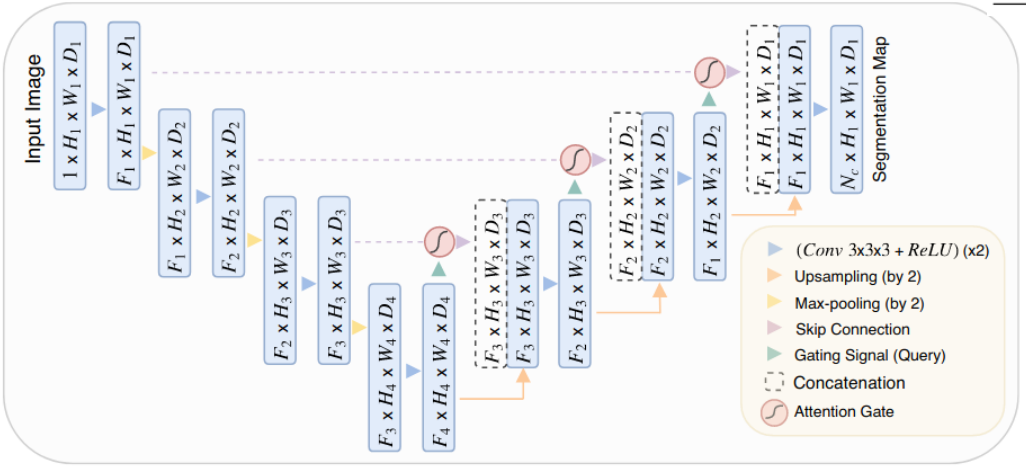
Attention U-Net

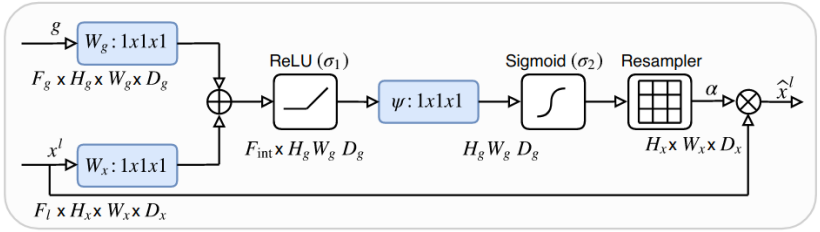
Oktay et al. "Attention U-Net: Learning Where to Look for the Pancreas"

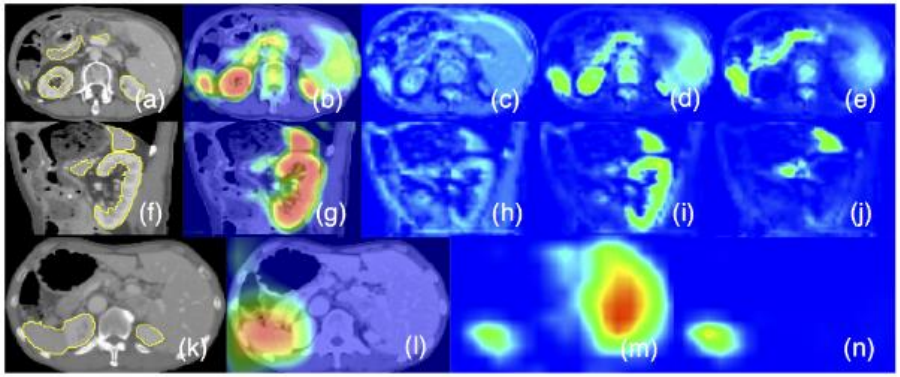
$$q_{att}^{l} = \psi^{T}\left(\sigma_1\left(W_x^{T} x_i^{l} + W_g^{T} g_i + b_g\right)\right) + b_\psi$$

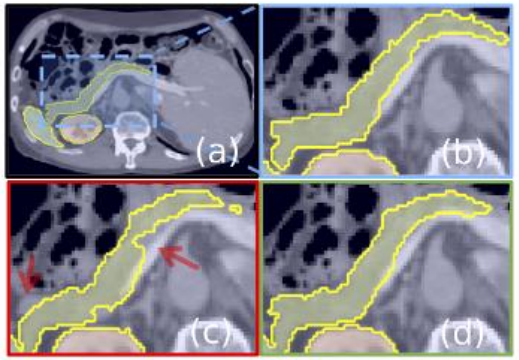
$$\alpha_i^{l} = \sigma_2\left(q_{att}^{l}(x_i^{l}, g_i; \Theta_{att})\right),$$

	Method	Dice Score	Precision	Recall	S2S Dist (mm)
BFT	U-Net [24]	0.690±0.132	0.680±0.109	0.733±0.190	6.389±3.900
	Attention U-Net	**0.712±0.110**	0.693±0.115	**0.751±0.149**	**5.251±2.551**
AFT	U-Net [24]	0.820±0.043	0.824±0.070	0.828±0.064	2.464±0.529
	Attention U-Net	**0.831±0.038**	0.825±0.073	**0.840±0.053**	**2.305±0.568**
SCR	U-Net [24]	0.815±0.068	0.815±0.105	0.826±0.062	2.576±1.180
	Attention U-Net	0.821±0.057	0.815±0.093	**0.835±0.057**	**2.333±0.856**



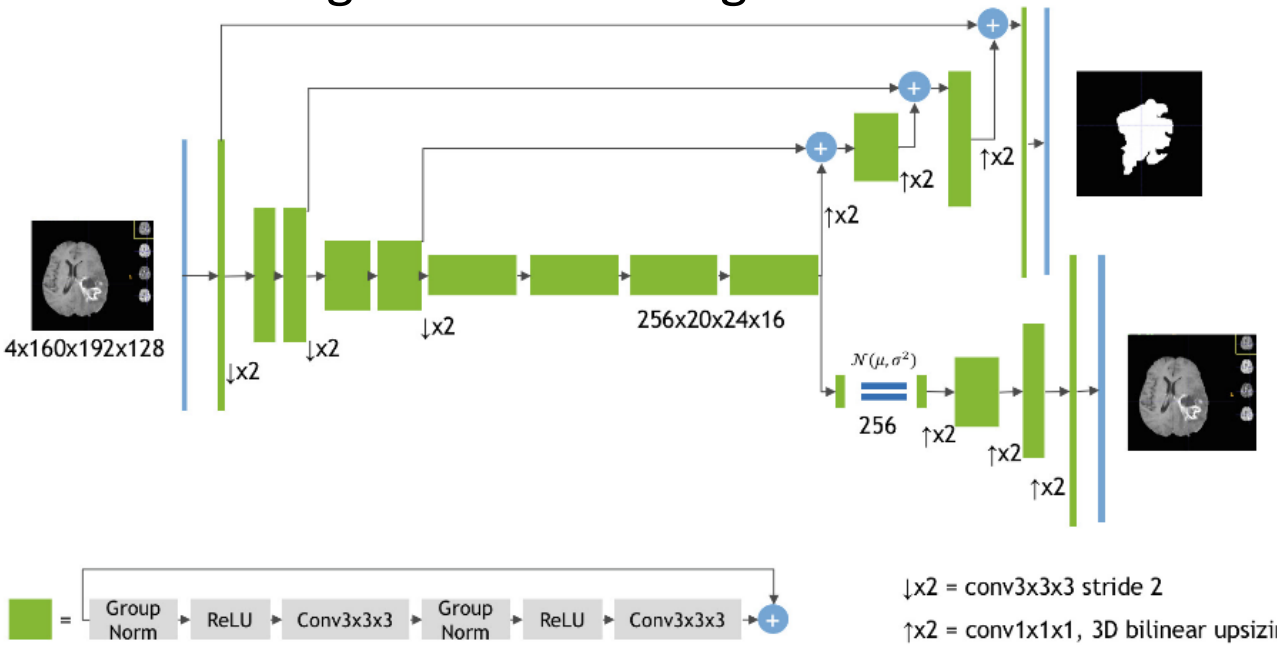




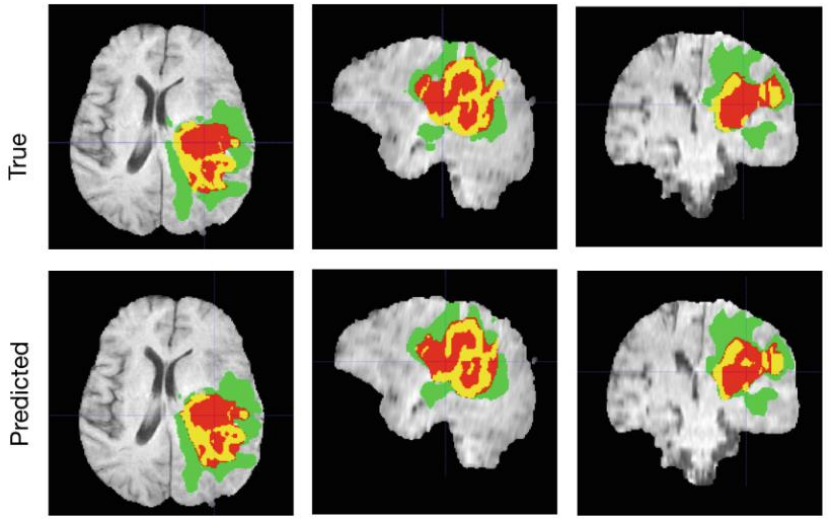


Multi-Task

Myronenko "3D MRI Brain Tumor Segmentation Using Autoencoder Regularization"

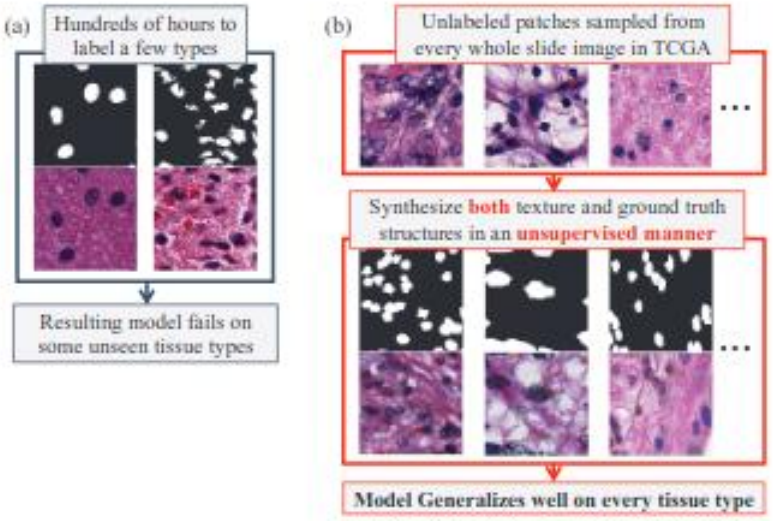


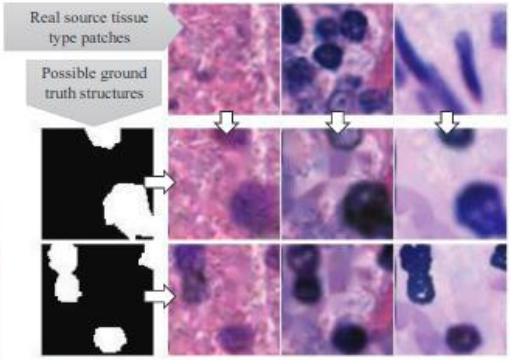
$$\mathbf{L} = \mathbf{L}_{dice} + 0.1 * \mathbf{L}_{\mathrm{L2}} + 0.1 * \mathbf{L}_{\mathrm{KL}}$$



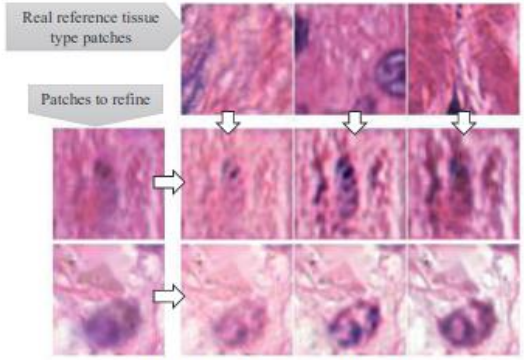
Synthesize

Hou et al. "Robust Histopathology Image Analysis: to Label or to Synthesize?"

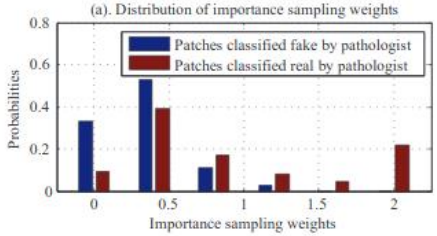


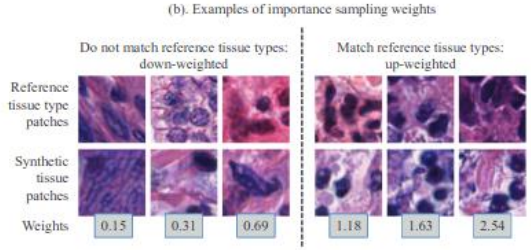


(a). Utilizing textures in various tissue types during the initial synthesis step.



(b). Refining the initial synthetic patches to be more like the reference type.



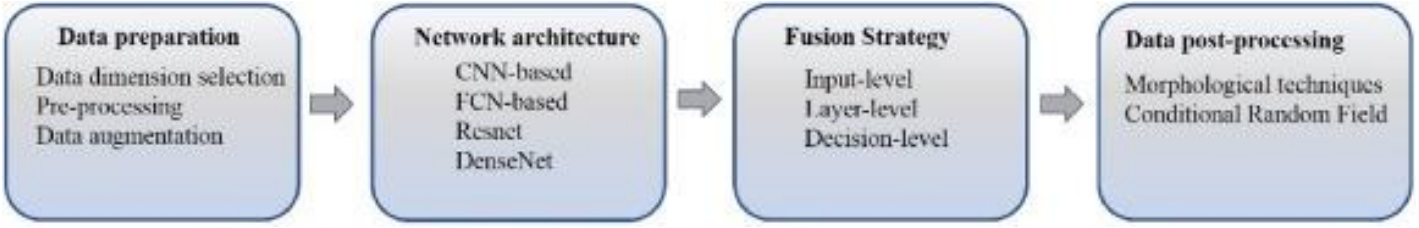


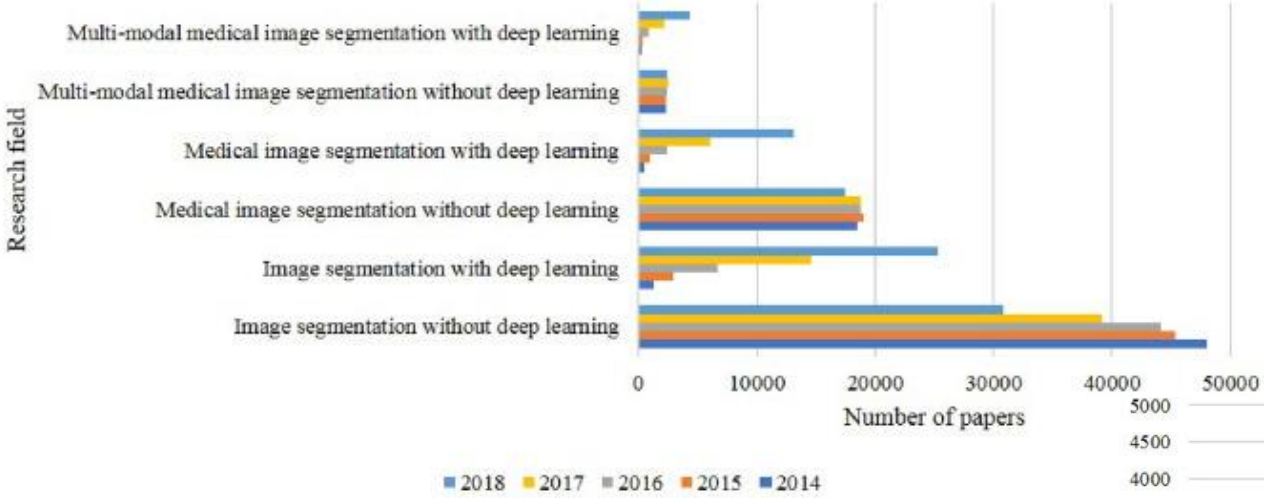
Nucleus segmentation methods	DICE Avg.
No hard examples	0.7476
No reference patch during refinement	0.7410
No importance weights	0.7533
Universal CNN (proposed)	**0.7612**

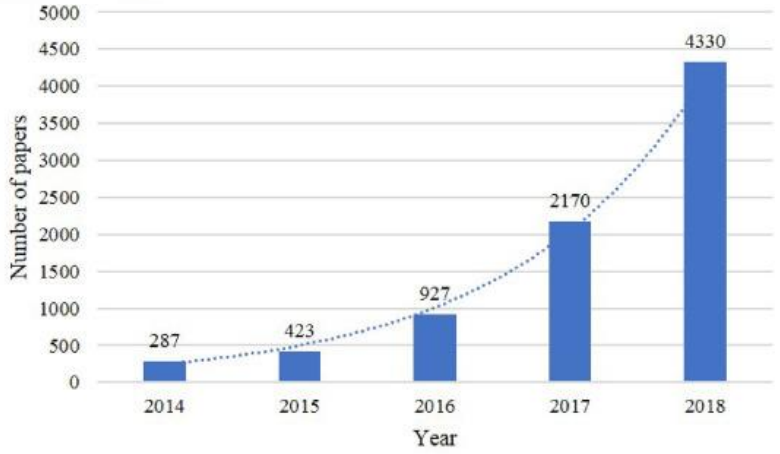
Lymphocyte detection methods	AUROC
Level Set features + supervised net [67]	0.7132
Fine-tuning VGG16 (supervised) [52]	0.6925
Universal CNN (proposed)	**0.7149**

Interesting Points

Zhou et al. "A review: Deep Learning for medical image segmentation using multi-modality fusion"







Agony of Choice

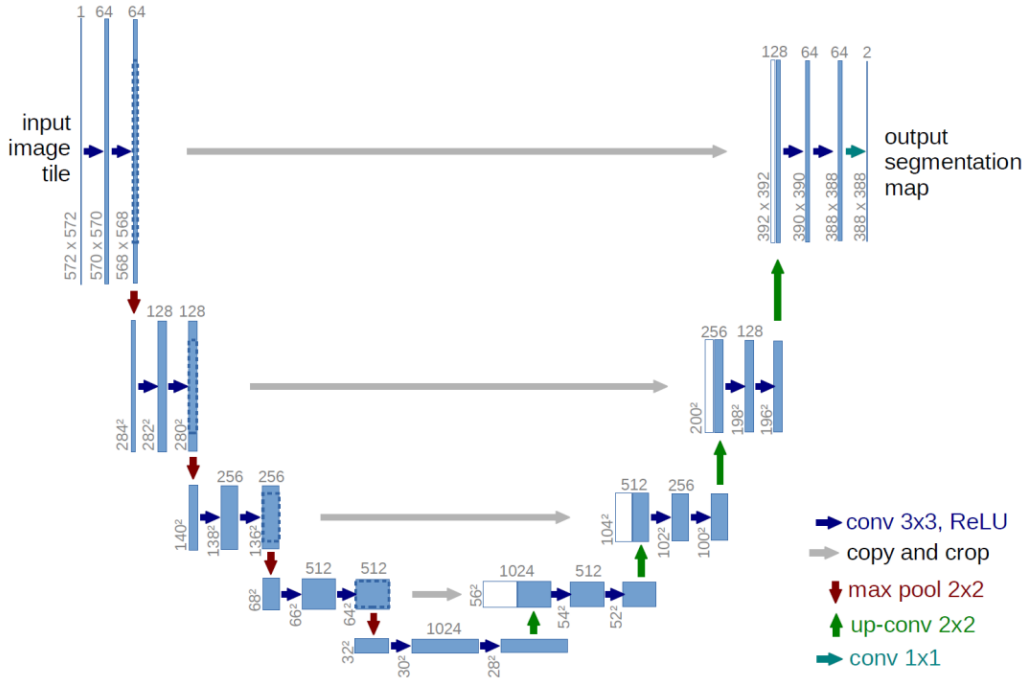
- Architecture: depth, width, scales, residuals, ….
- Loss function: (weighted) cross entropy, IoU, Dice, …
- Sampling strategy: equally per class, fore/background, uniform, …
- Optimization: optimizer, learning rate, momentum, regularization, ...
- Data normalization: z-score, bias field correction, histogram matching, …
- Post-processing: CRFs, ensemble of networks, ….

Conclusions

- Semantic Segmentation is one of the most studied problems in medical imaging.
  - Conducting **quantitative analyses**, e.g. measuring the volume of the ventricular cavity.
  - Determining the precise **location and extent** of an organ or a certain type of tissue, e.g. a tumour for treatment such as radiation therapy.
  - Creating **3D models** used for **simulation**, e.g. generating a model of an abdominal aortic aneurysm for simulating stress

Lab Session!

- Brain tumor segmentation
  - U-Net



TorchVision

Facebook Open Source

Deep learning for medical imaging

Olivier Colliot, PhD
Research Director at CNRS
Co-Head of the ARAMIS Lab – www.aramislab.fr
PRAIRIE – Paris Artificial Intelligence Research Institute

Maria Vakalopoulou, PhD
Assistant Professor at CentraleSupelec
Mathematics and Informatics (MICS)
Office: Bouygues Building Sb.132

Master 2 - MVA

Course website: http://www.aramislab.fr/teaching/DLMI-2019-2020/
Piazza (for registered students): https://piazza.com/centralesupelec/spring2020/mvadlmi/