

Deep Learning

State of the Art Convolutional Architectures

Michaël Sdika ¹

¹ CNRS, CREATIS UMR 5220

Machine learning



Unstructured data

...



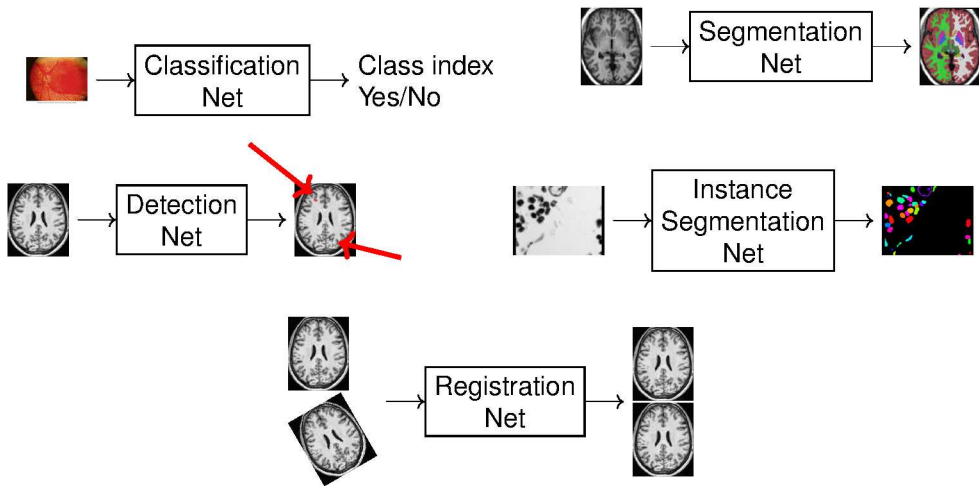
- physiological parameters
- Yes/No
- Category
- ...

Supervised Deep Learning

- ▶ How to represent the mapping ?
 - Deep learning : Neural network
 - Which architecture for the network ? 

- ▶ How to estimate the network coefficient ?
 - Loss functions ?
 - Optimization ?
 - Generalization ?

5 classes of architectures addressed in this course



Outline

Short reminder on MLP and CNN

Architectures for some important tasks

- Classifiers

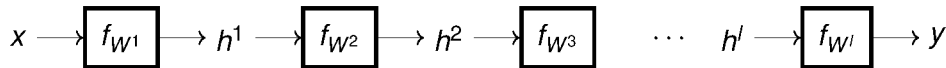
- Encoder / Decoder architectures

- Detection

- Instance Segmentation

- Image Registration

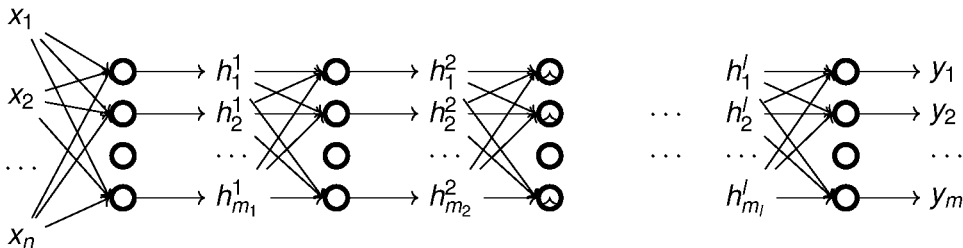
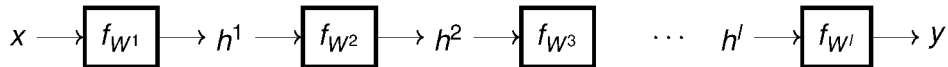
Deep Neural Network



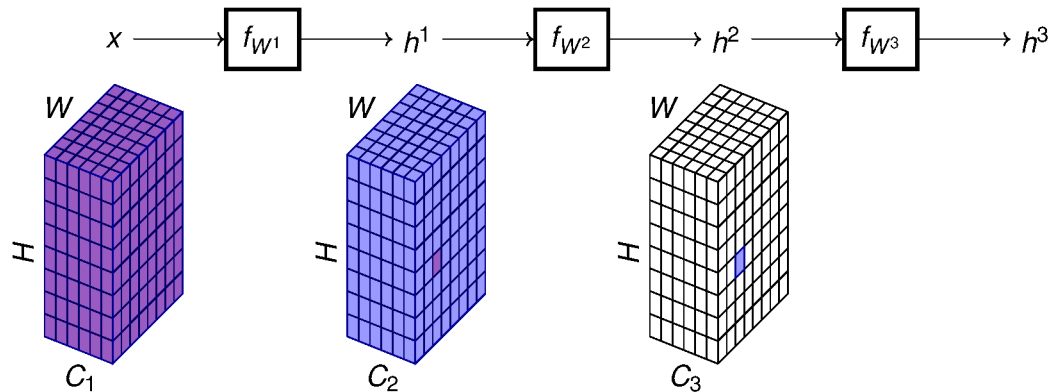
Basic Layers :

- ▶ Linear Layers : Fully Connected / Convolution : mixing features
- ▶ Activation layers : introducing nonlinearity
- ▶ Pooling layers : spatial aggregation, subsampling
- ▶ Normalization layers : stabilizing the training

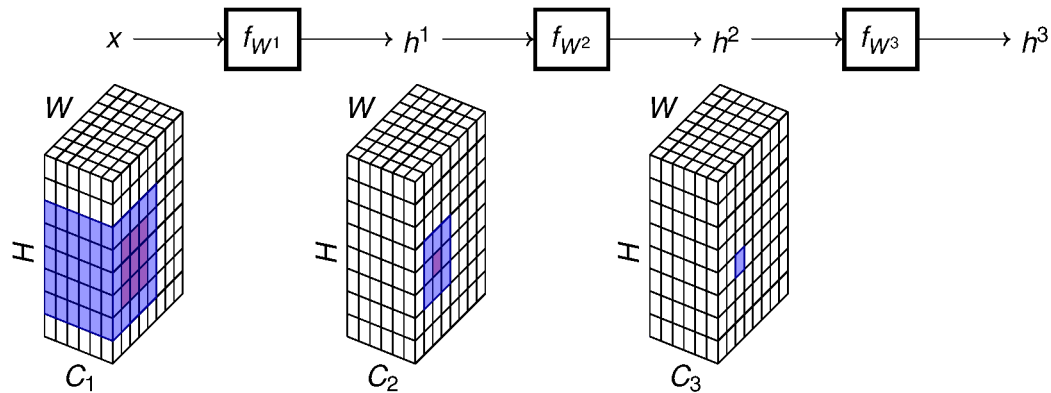
Multi Layer Perceptron



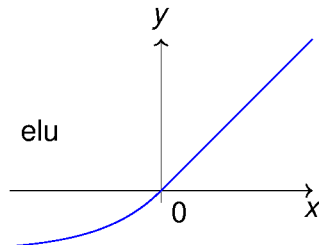
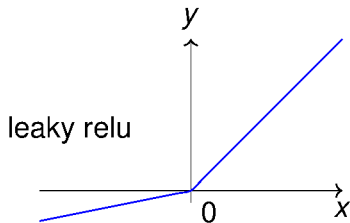
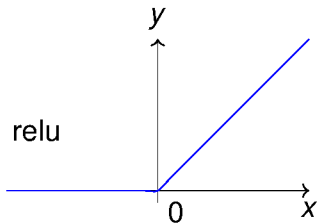
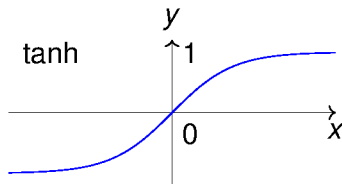
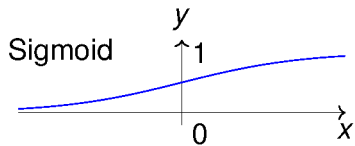
Multi Layer Perceptron



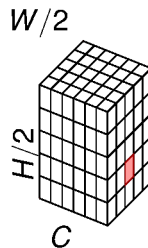
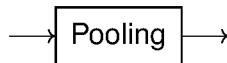
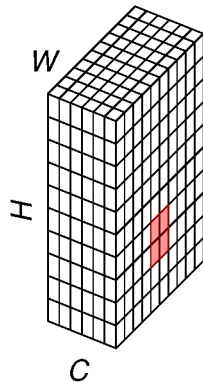
CNN



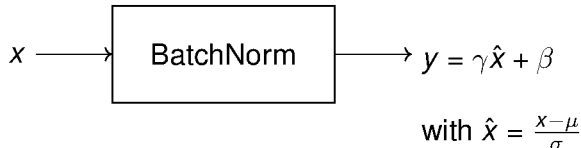
Activation functions



Pooling

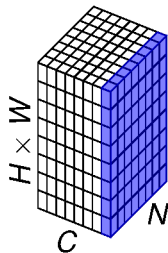


Batch Normalization



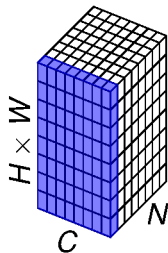
- ▶ μ, σ : mean, std of x over a minibatch
- ▶ γ, β : trainable parameters
- ▶ Inference : use average μ, σ from training

Related Normalization



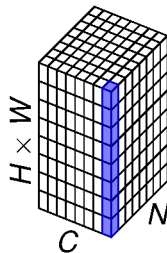
Batch Norm

☹️ small
minibatch



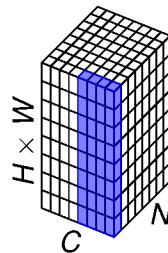
Layer Norm

😊 small
minibatch



Instance Norm

remove contrast
style transfer



Group Norm

Ulyanov et al arxiv 2016, Instance normalization : The missing ingredient for fast stylization
Ba et al, 2016, Layer Normalization
Wu & He 2018, Group Normalization

Outline

Short reminder on MLP and CNN

Architectures for some important tasks

- Classifiers**

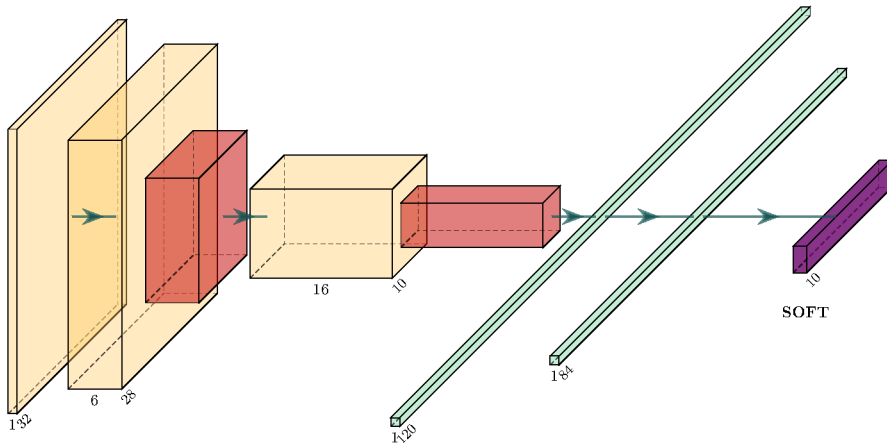
- Encoder / Decoder architectures

- Detection

- Instance Segmentation

- Image Registration

Le Net



LeCun et al., Neural Computation 1989, "Backpropagation Applied to Handwritten Zip Code Recognition"
LeCun et al., 1998, Proceedings of the IEEE, Gradient-based learning applied to document recognition.

Image Net

SUN, 131K

[Xiao et al. '10]

LabelMe, 37K

[Russell et al. '07]

PASCAL VOC, 30K

[Everingham et al. '06-'12]

Caltech101, 9K

[Fei-Fei, Fergus, Perona, '03]

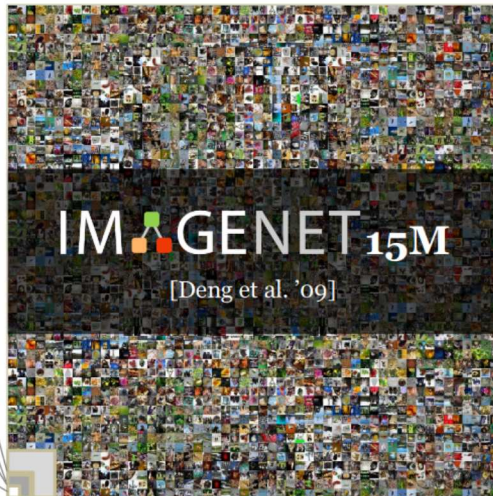
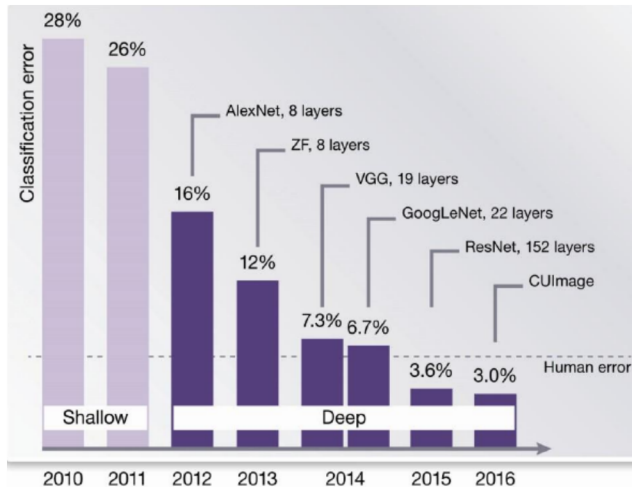


Image Net



Alex Net

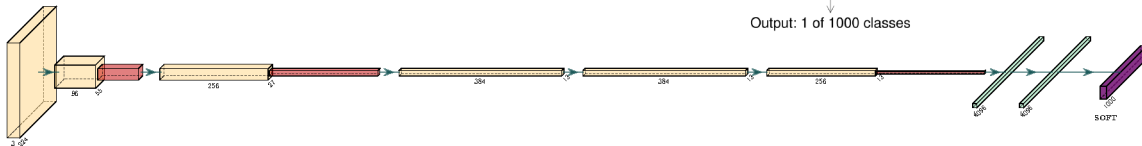
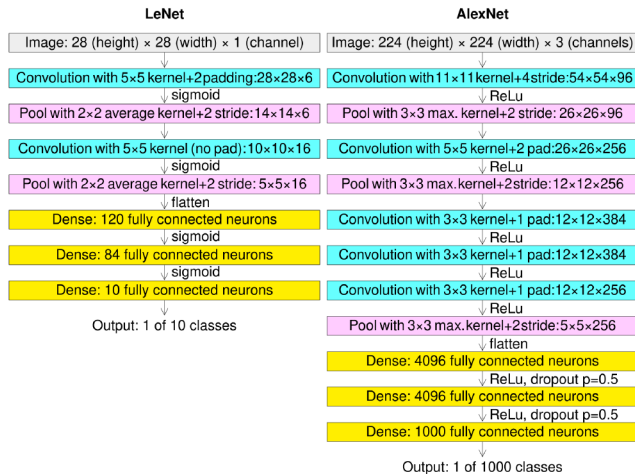
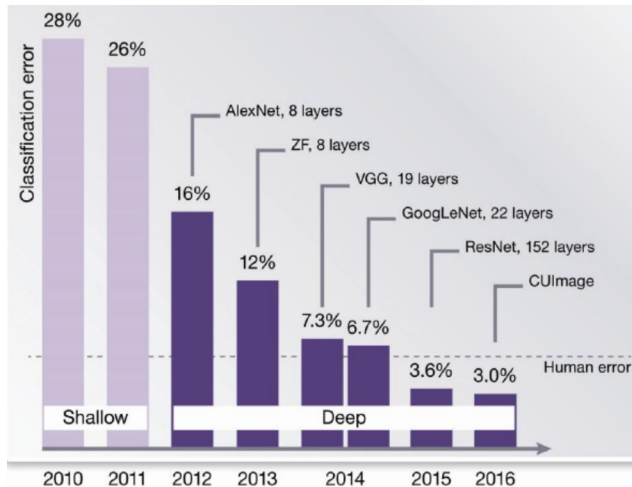


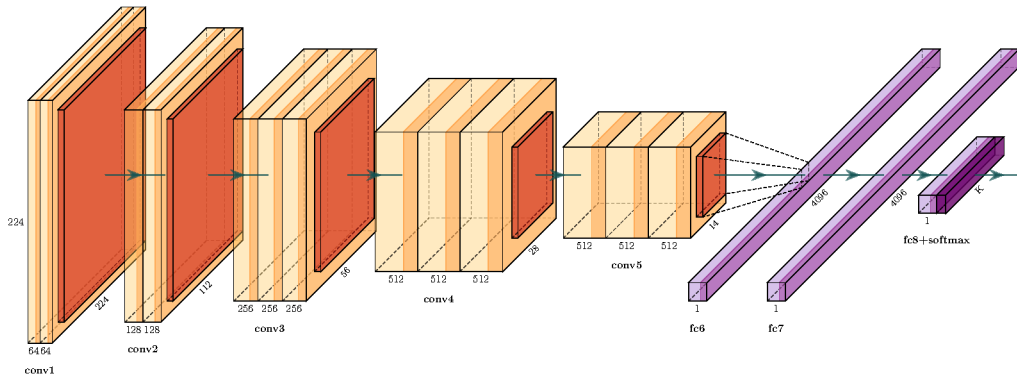
Image Net



VGG

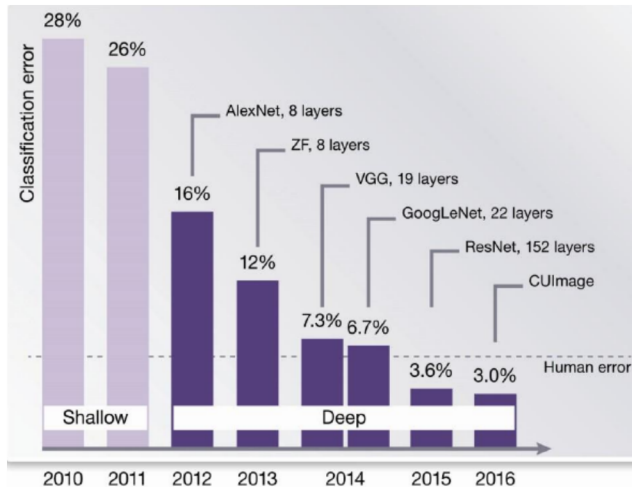
Deeper network

3x3 convolutions

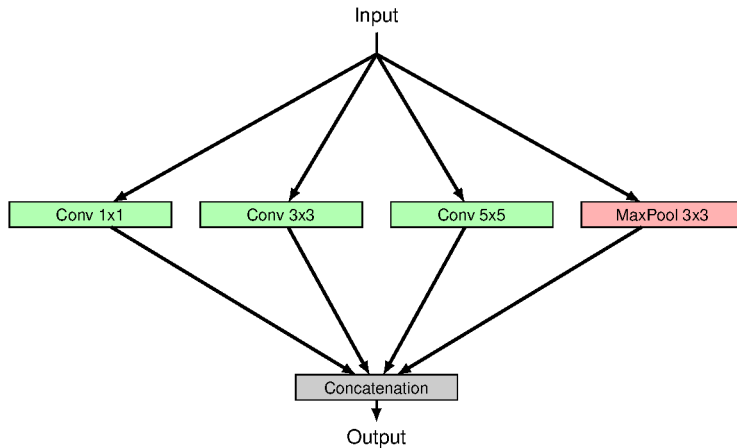


Simonyan & Zisserman, ICLR 2015, Very Deep Convolutional Networks for Large-Scale Image Recognition

Image Net

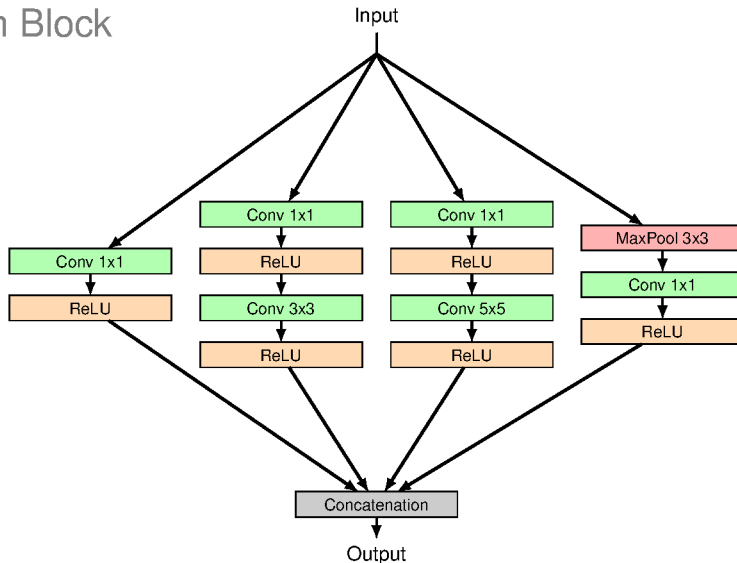


Inception Block

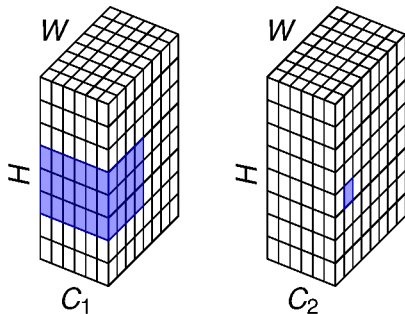


Szegedy et al, CVPR 2015, Going Deeper With Convolutions

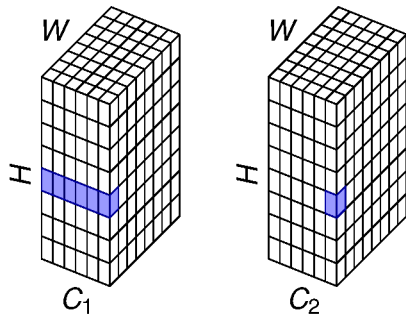
Inception Block



1x1 Convolution



3x3 convolution
receptive field



1x1 convolution
receptive field

GoogLe Net

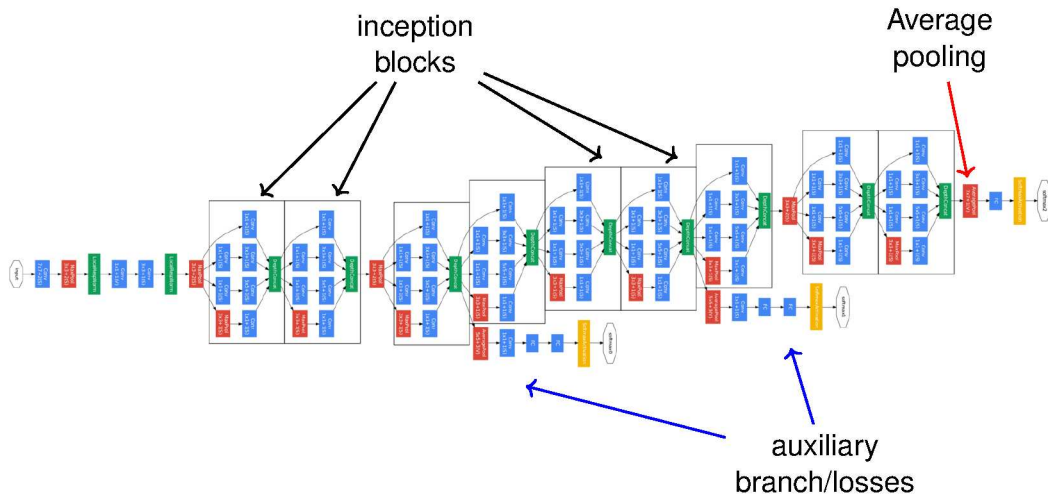
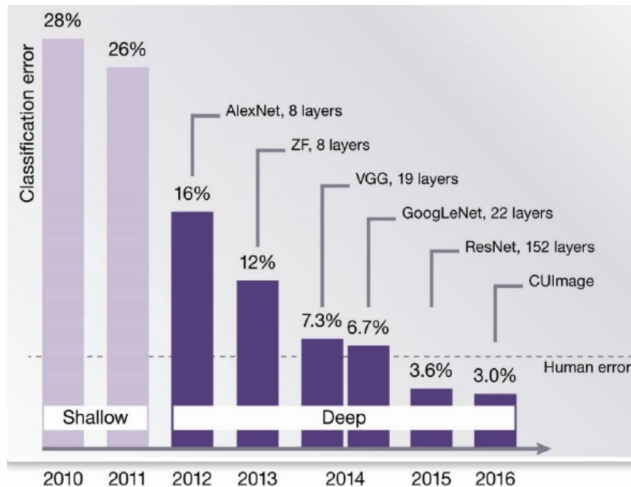
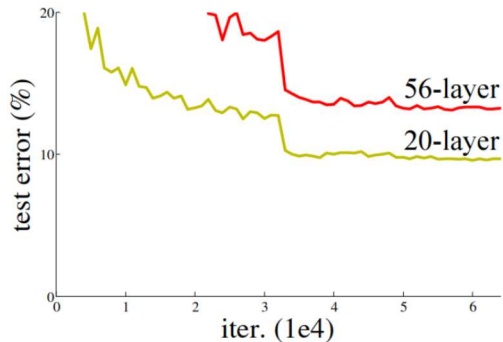
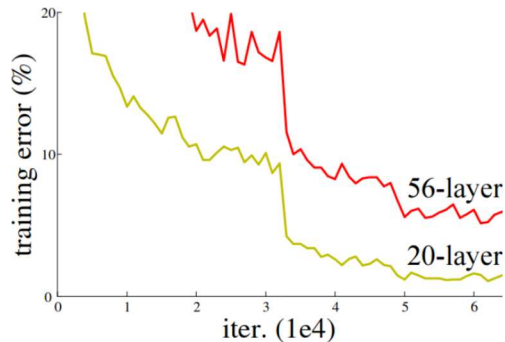


Image Net



Going Deeper ??



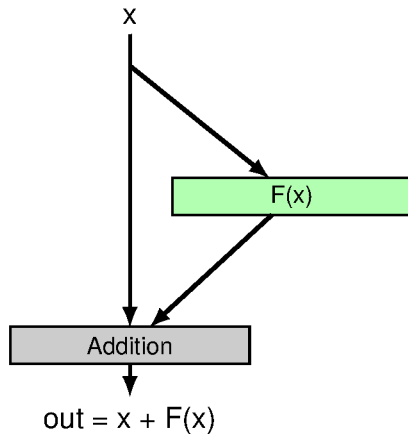
Residual Block

observation :

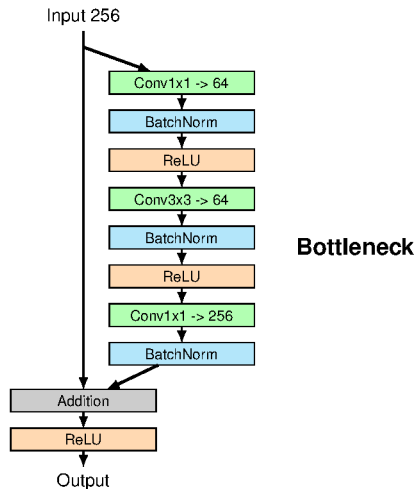
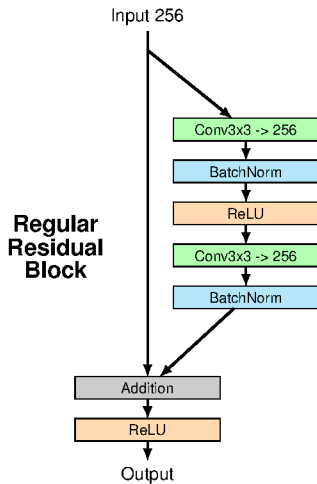
- ▶ more layers $\Rightarrow$ higher train errors
- ▶ Problem is training

Architecture easier to train

Vanishing gradient



Residual Block



Res Net

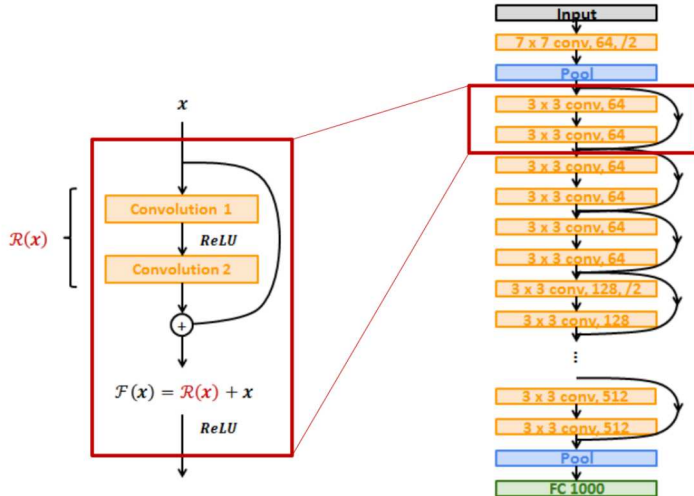
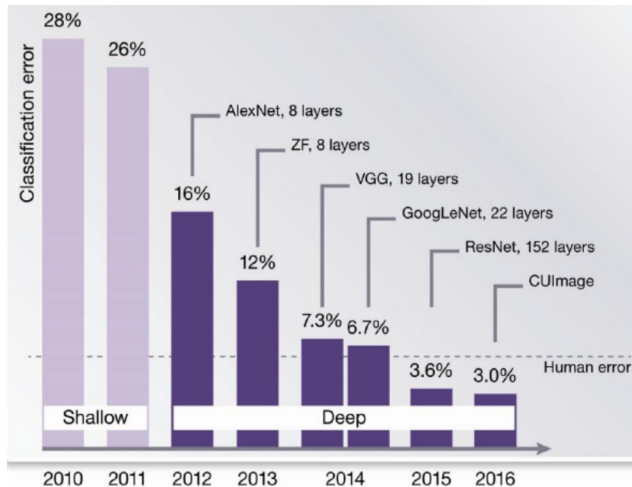
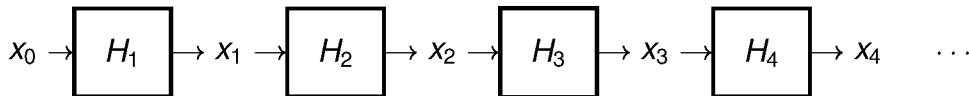


Image Net



Dense Block



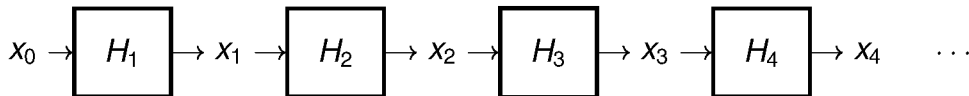
Res block :

$$x_l = x_{l-1} + H_l(x_{l-1})$$

Dense block :

$$x_l = H_l([x_0, x_1, \dots, x_{l-1}])$$

Dense Block



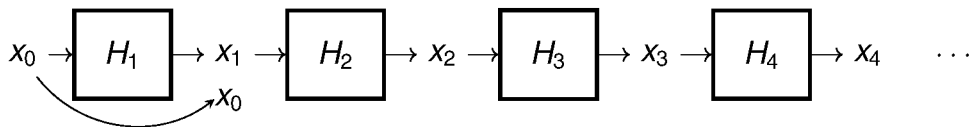
Res block :

$$x_l = x_{l-1} + H_l(x_{l-1})$$

Dense block :

$$x_l = H_l([x_0, x_1, \dots, x_{l-1}])$$

Dense Block



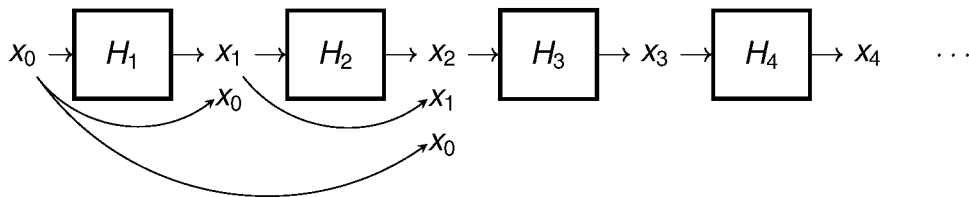
Res block :

$$x_i = x_{i-1} + H_i(x_{i-1})$$

Dense block :

$$x_i = H_i([x_0, x_1, \dots, x_{i-1}])$$

Dense Block



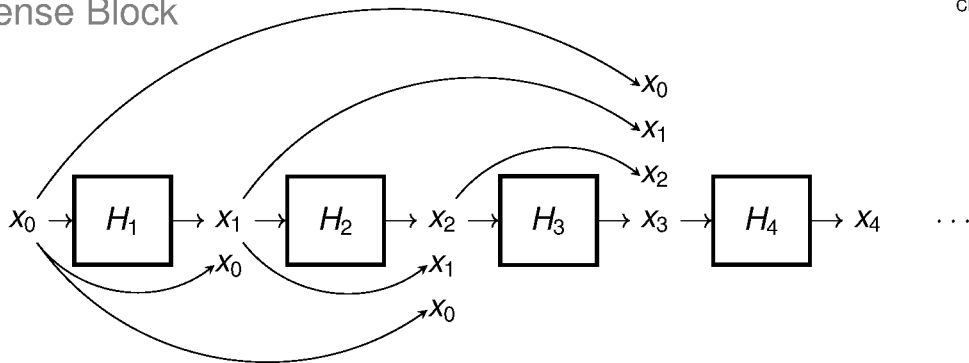
Res block :

$$x_i = x_{i-1} + H_i(x_{i-1})$$

Dense block :

$$x_i = H_i([x_0, x_1, \dots, x_{i-1}])$$

Dense Block

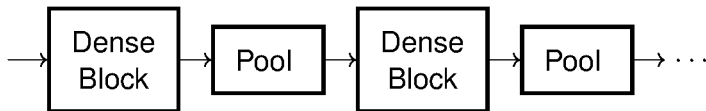
**Res block :**

$$x_i = x_{i-1} + H_i(x_{i-1})$$

Dense block :

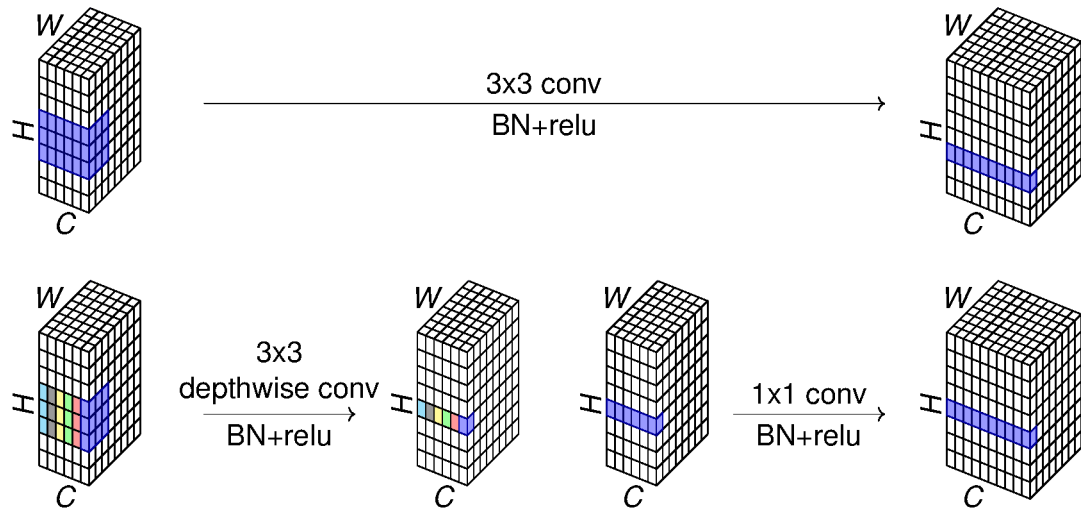
$$x_i = H_i([x_0, x_1, \dots, x_{i-1}])$$

Dense Net

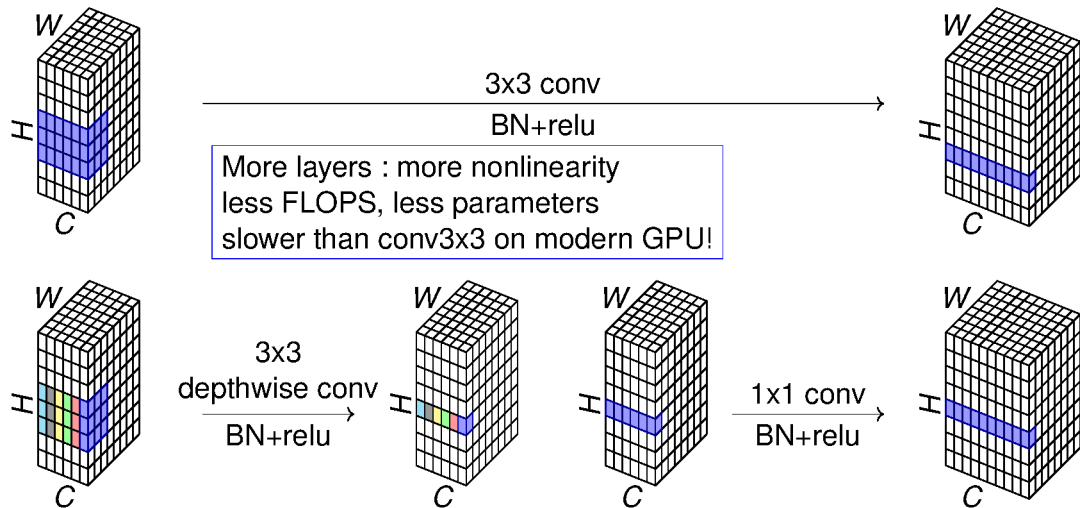


Gao, et al. CVPR 2017, Densenet : densely connected convolutional networks

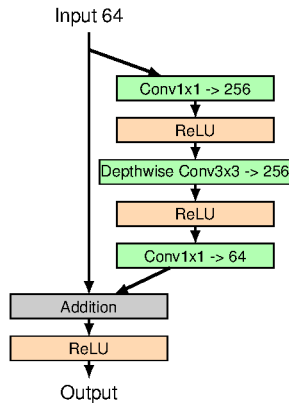
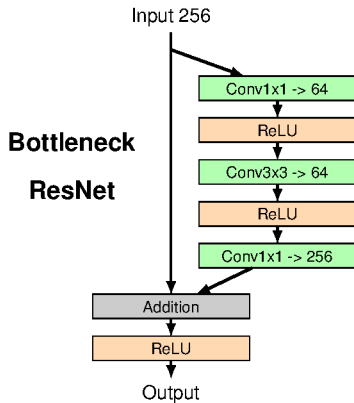
Mobile Net V1 : depthwise conv



Mobile Net V1 : depthwise conv

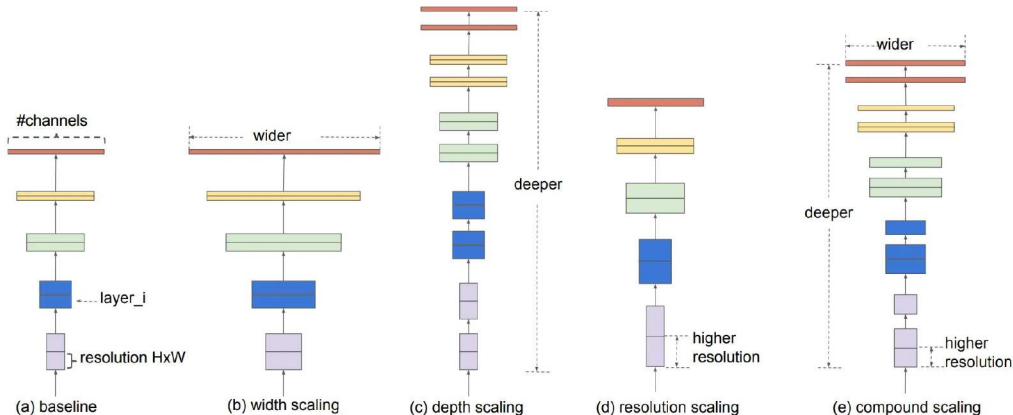


Mobile Net V2 : inverted bottleneck

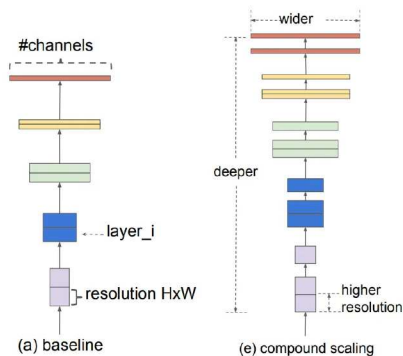


Sandler et al, CVPR 2018, MobileNetV2 : Inverted Residuals and Linear Bottlenecks

Efficient Net : compound scaling of networks



Efficient Net : compound scaling of networks



► depth, width, resolution for B1

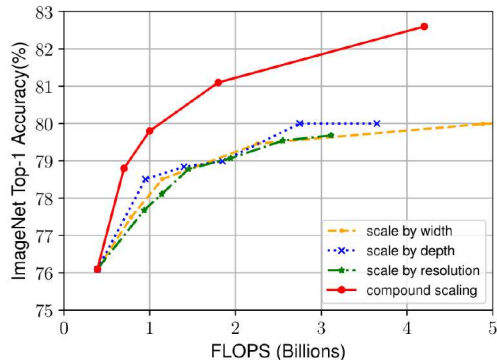
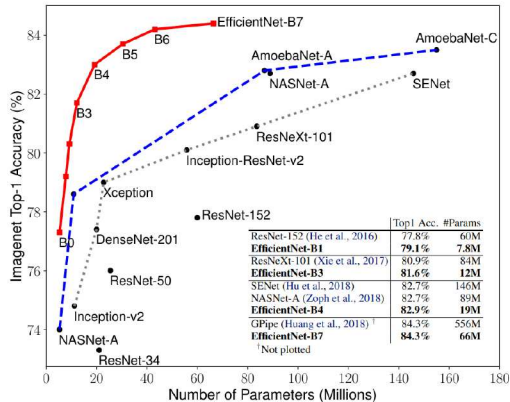
- $d_1 = \alpha d_0$
- $w_1 = \beta w_0$
- $r_1 = \gamma r_0$

► grid search for α, β, γ

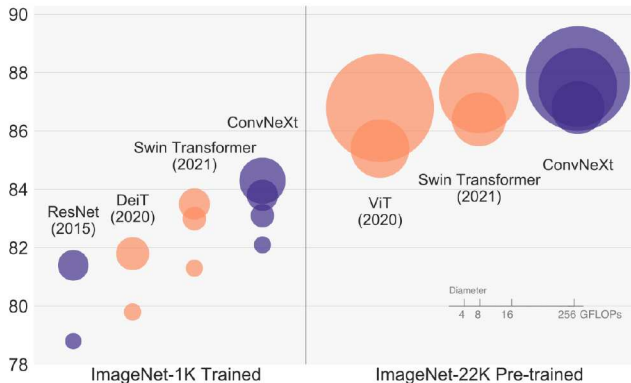
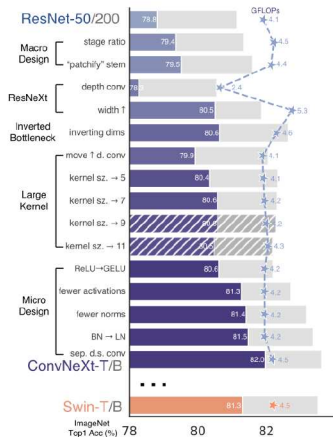
► Bk : $\alpha^k, \beta^k, \gamma^k$

Efficient Net v2 :
architecture grid search fo B0

Efficient Net



ConvNext



Outline

Short reminder on MLP and CNN

Architectures for some important tasks

- Classifiers

- Encoder / Decoder architectures**

- Detection

- Instance Segmentation

- Image Registration

Encoder/Decoder architecture

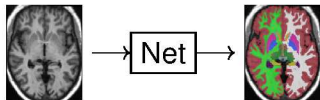


Image Segmentation

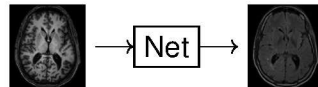
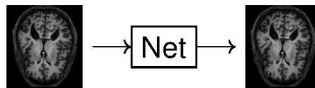
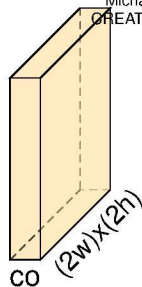
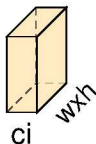


Image Synthesis, Domain adaptation

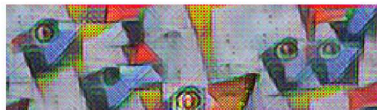


Denoising

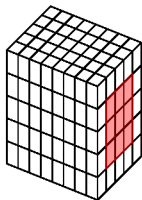
Upsampling Layer



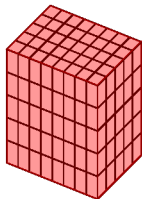
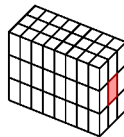
- ▶ deconv
 - transpose of strided conv matrix
 - learn the upsampling coefficient
- ▶ unpool :
 - upsample on maxpool indices
- ▶ interpolation
 - bi/tri linear
 - no chessboard artifact



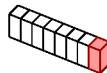
Fully Convolutional Network : FC as convolution



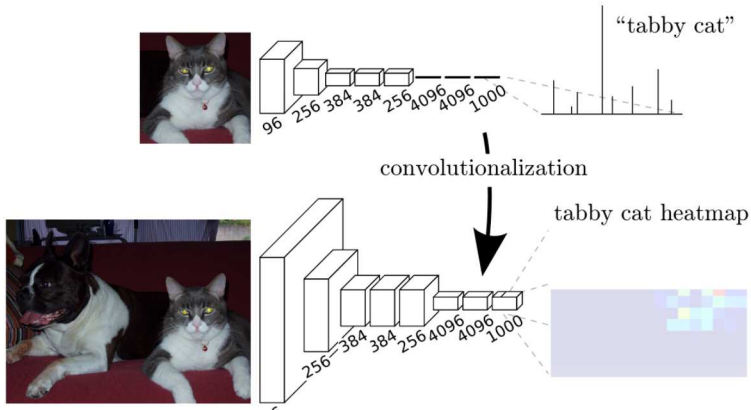
3x3 conv



5x5 conv

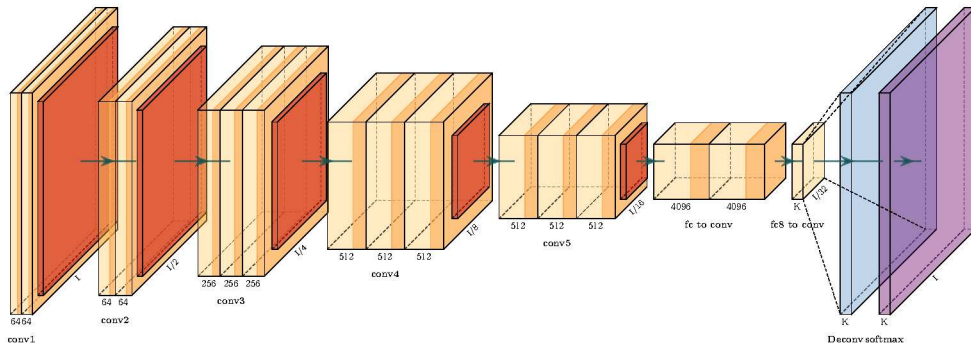


Fully Convolutional Network : FC as convolution



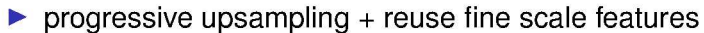
- use kernels that cover their entire input regions

Fully Convolutional Network



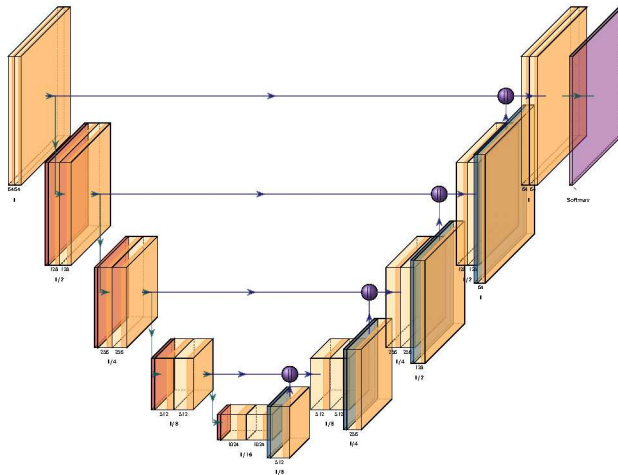
- deconv layer + pixelwise cross entropy

Long et al., CVPR 2015, Fully convolutional networks for semantic segmentation

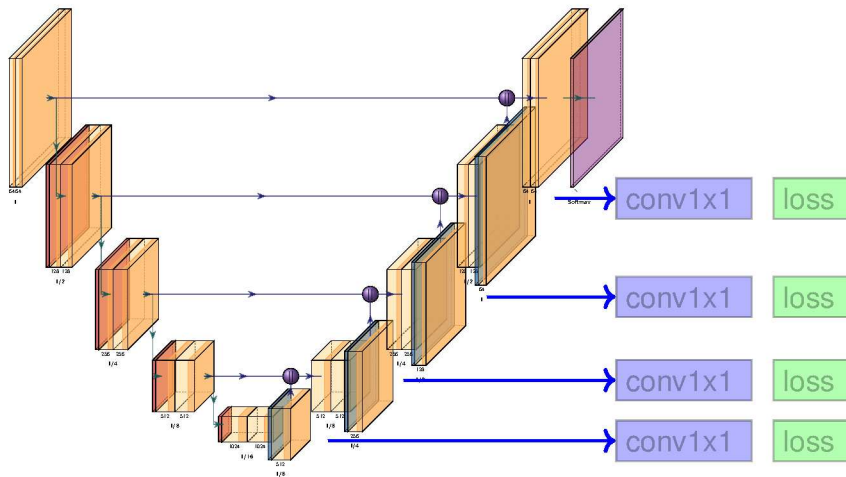


43/73

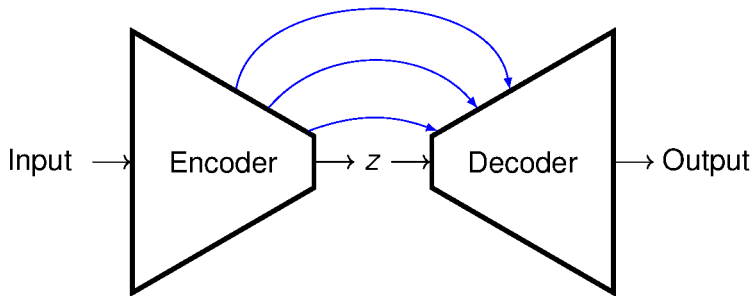
Unet



Unet



Encoder / Decoder



Tiramisu Net :

- * conv → dense block

Eff-UNet :

- * Encoder is efficient net
- * standard unet Decoder

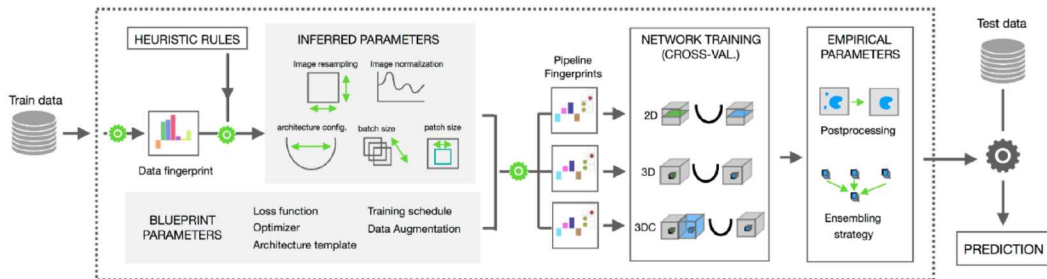
Unet+ / Unet++ :

- * add skip connection across scale

Jegou et al, CVPR 2017, The One Hundred Layers Tiramisu : Fully Convolutional DenseNets for Semantic Segmentation

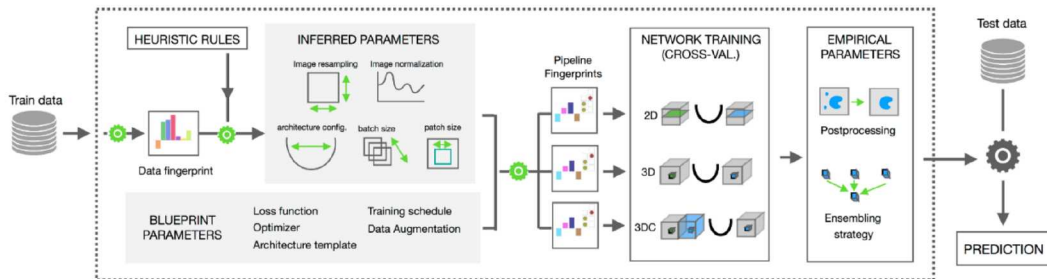
Baheti et al, CVPR 2020, Eff-UNet : A Novel Architecture for Semantic Segmentation in Unstructured Environment

nn-Unet : self configuration



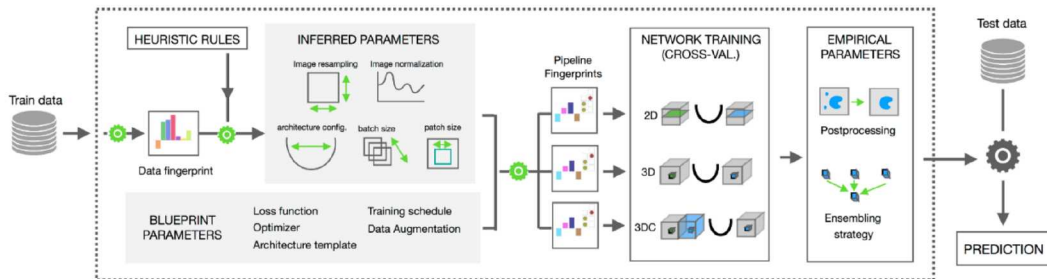
Isensee, et al. Nature 2021, nnU-Net : a self-configuring method for deep learning-based biomedical image segmentation

nn-Unet : self configuration



Isensee, et al. Nature 2021, nnU-Net : a self-configuring method for deep learning-based biomedical image segmentation

nn-Unet : self configuration



Isensee, et al. Nature 2021, nnU-Net : a self-configuring method for deep learning-based biomedical image segmentation

Outline

Short reminder on MLP and CNN

Architectures for some important tasks

- Classifiers

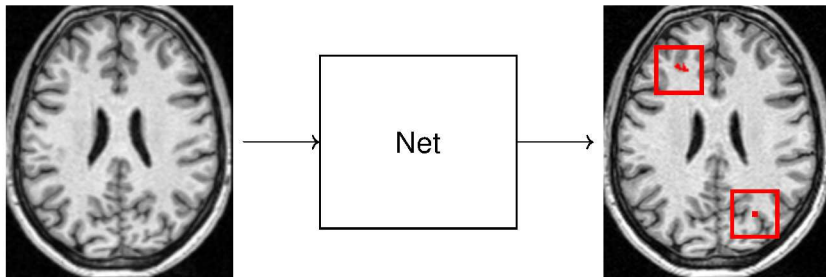
- Encoder / Decoder architectures

- Detection**

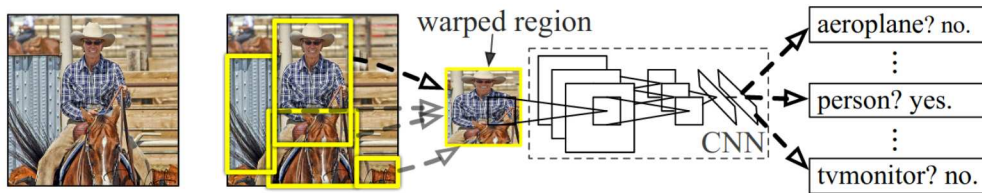
- Instance Segmentation

- Image Registration

Object Detection



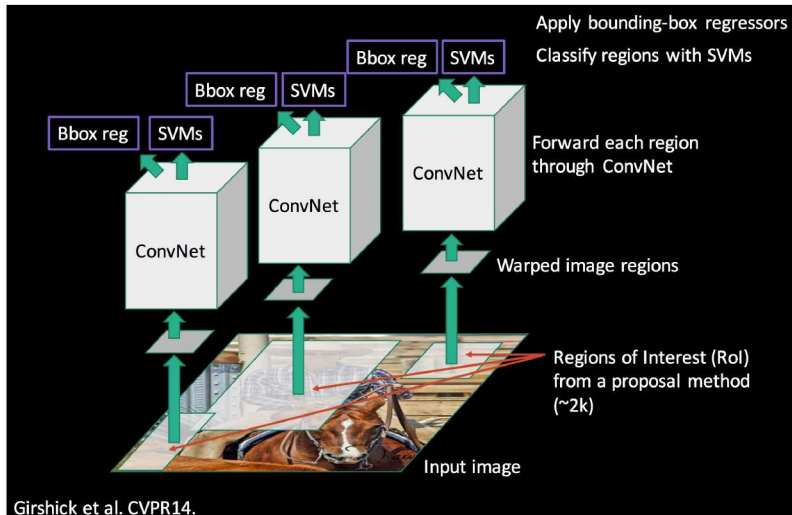
R-CNN



- ▶ regions extractor (non deep)
- ▶ for each region
 - deep feature
 - classif + box regression

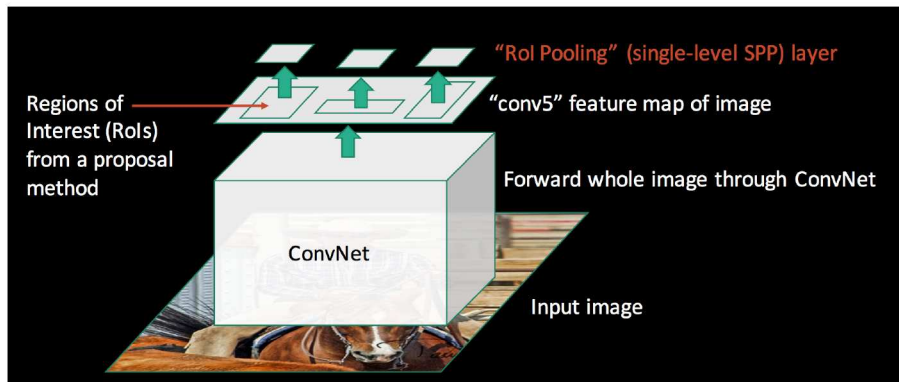
→ **very slow**

R-CNN



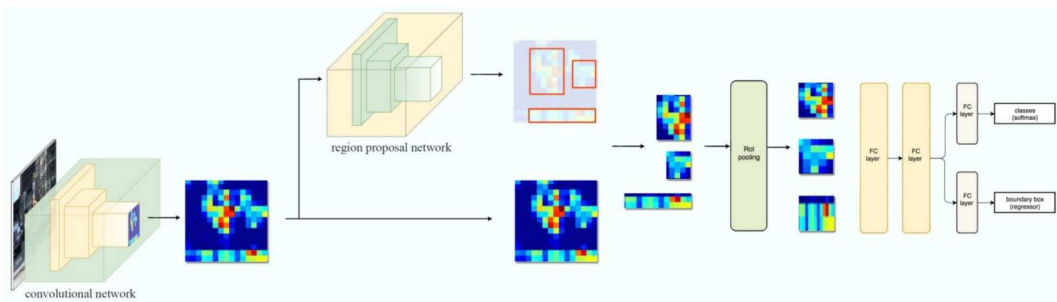
Girshick, et al. CVPR 2015, Rich feature hierarchies for accurate object detection and semantic segmentation
(credit : [jhui.github.io/2017/03/15/Fast-R-CNN-and-Faster-R-CNN](https://github.com/jhui))

Fast RCNN



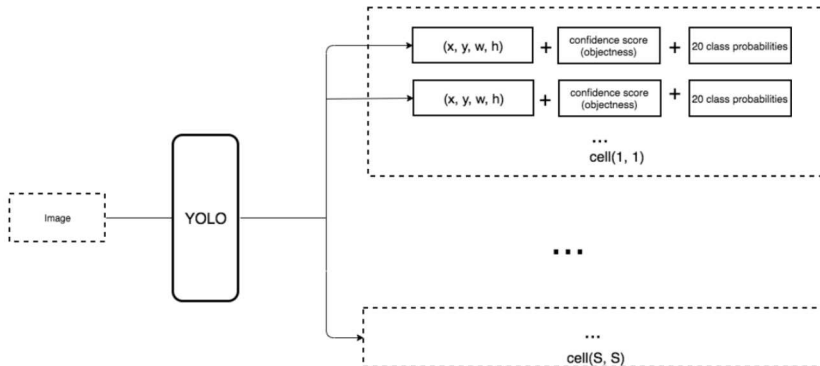
- all the feature computed at once

Faster RCNN



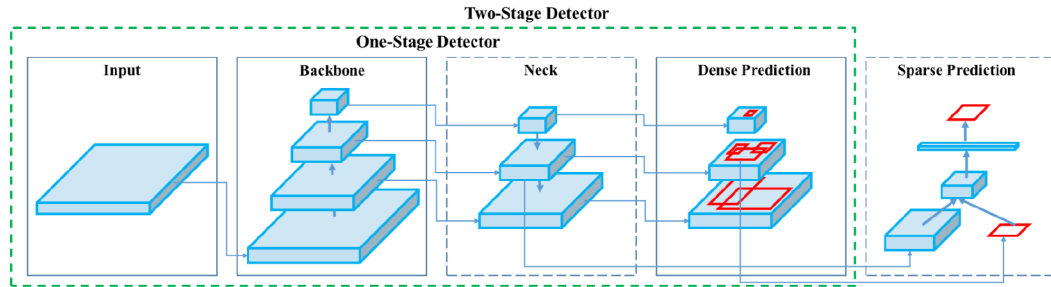
- DEEP region proposal network :
for each position in the feature map, output
 - k proba : object vs non object
 - k offset for bounding box proba

YOLO (You Only Look Once)



- ▶ yolo V1 : CNN $\rightarrow$ pb with small object
- ▶ yolo V2, V3 : Unet

YOLO (You Only Look Once)



Bochkovskiy et al, arxiv 2020, Yolov4 : Optimal speed and accuracy of object detection

Outline

Short reminder on MLP and CNN

Architectures for some important tasks

Classifiers

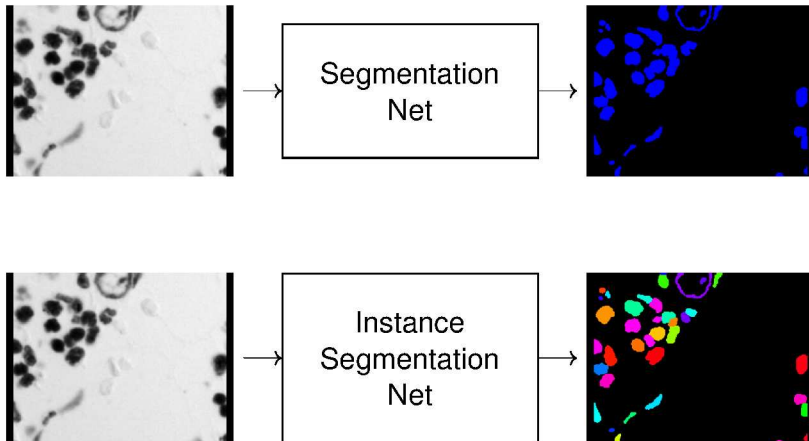
Encoder / Decoder architectures

Detection

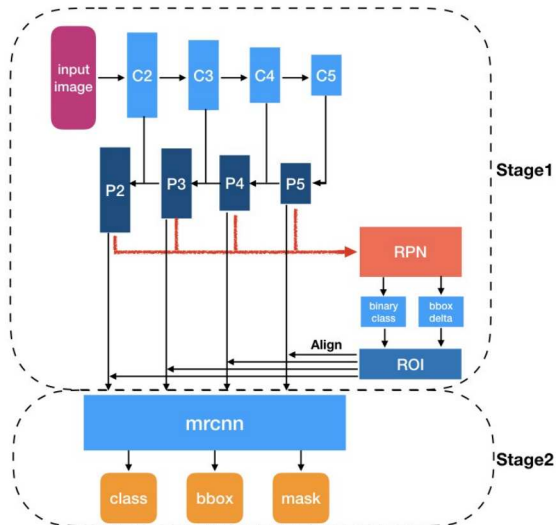
Instance Segmentation

Image Registration

Instance Segmentation



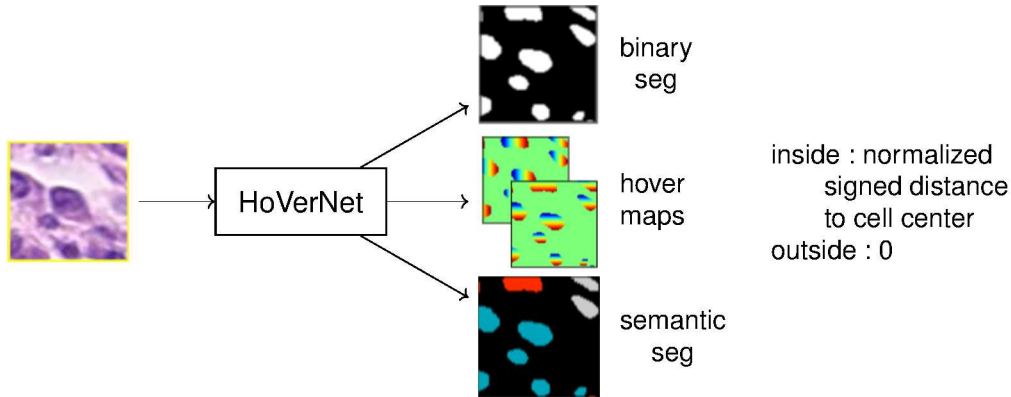
Mask R-CNN



He, Kaiming, et al, ICCV 2017, Mask r-cnn

<https://alittlepain833.medium.com/simple-understanding-of-mask-rcnn-134b5b330e95>

HoVerNet

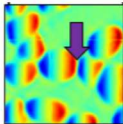


Graham, et al. "Hover-net : Simultaneous segmentation and classification of nuclei in multi-tissue histology images." Medical image analysis, 2019

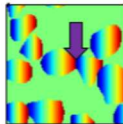
HoVerNet



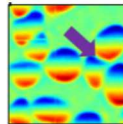
Image
Crop



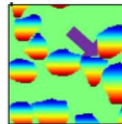
Horizontal Map
Prediction



Horizontal Map
Ground Truth



Vertical Map
Prediction



Vertical Map
Ground Truth

Graham, et al. "Hover-net : Simultaneous segmentation and classification of nuclei in multi-tissue histology images." Medical image analysis, 2019

Outline

Short reminder on MLP and CNN

Architectures for some important tasks

- Classifiers

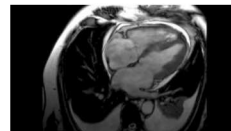
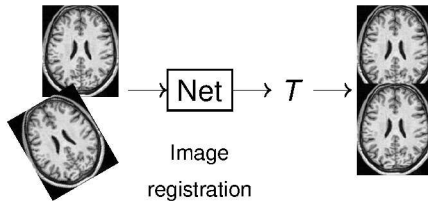
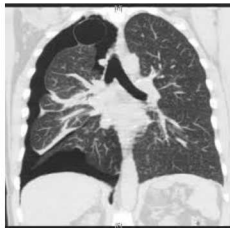
- Encoder / Decoder architectures

- Detection

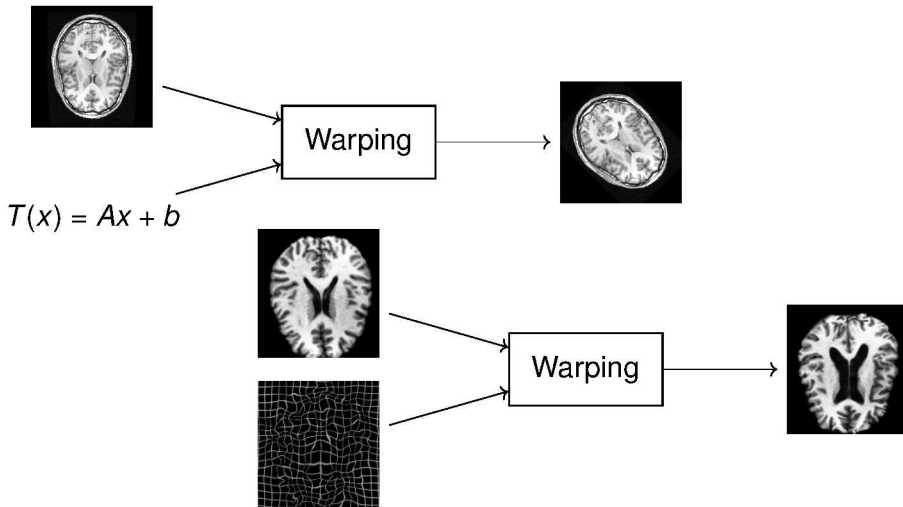
- Instance Segmentation

- Image Registration**

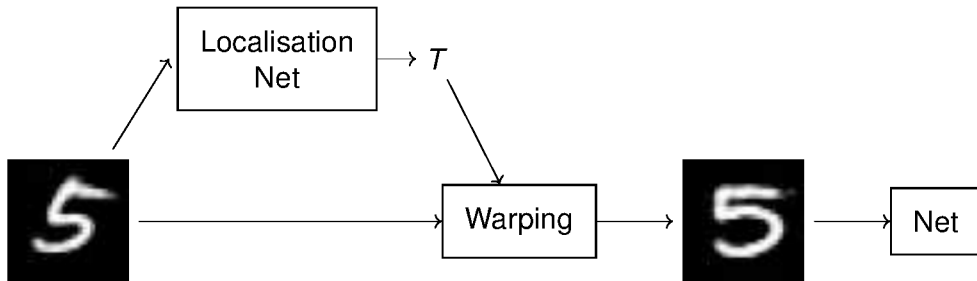
Motion/Registration



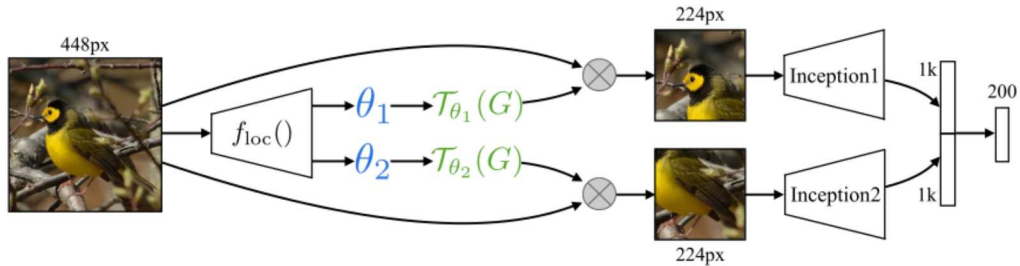
Warping Layer



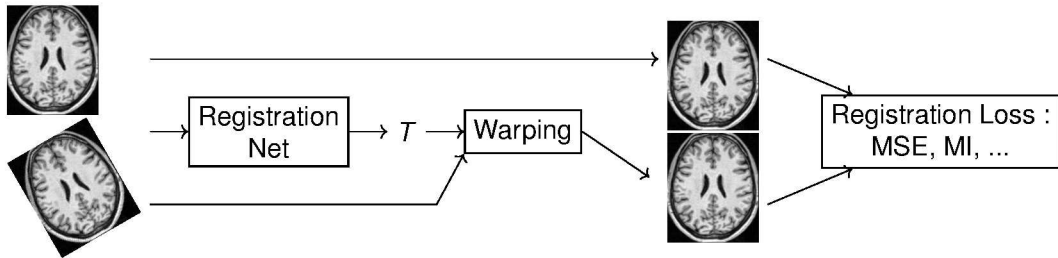
Spatial Transformer Networks



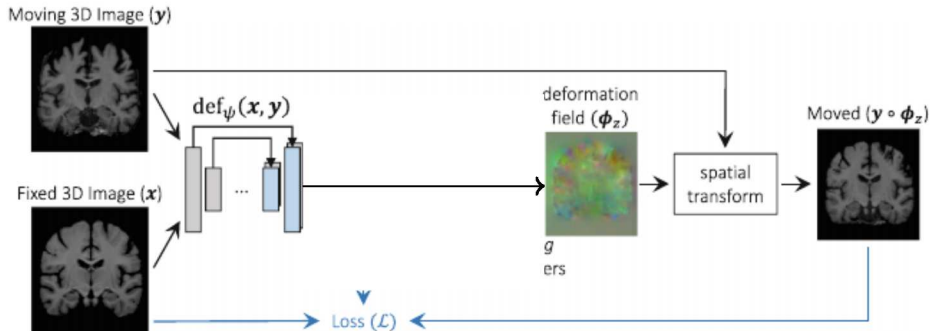
Spatial Transformer Networks



Unsupervised registration with deep learning

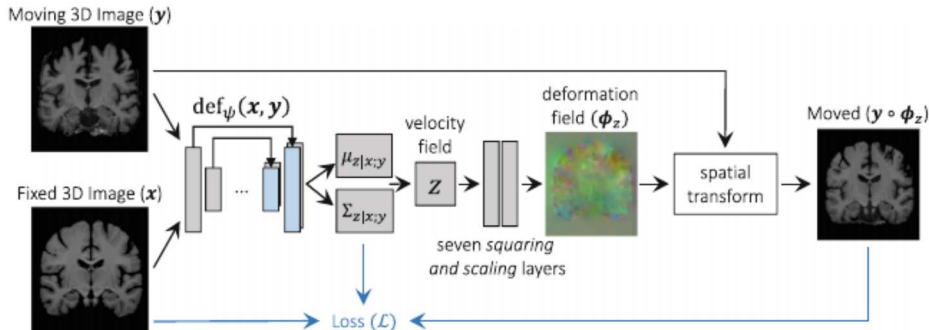


Unsupervised learning, VoxelMorph



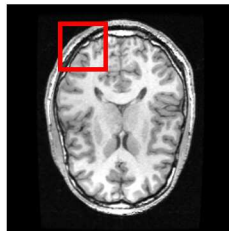
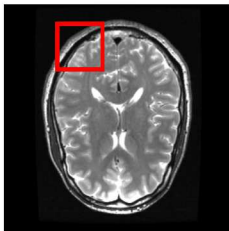
- registration loss : no reference warp needed

Unsupervised learning, VoxelMorph

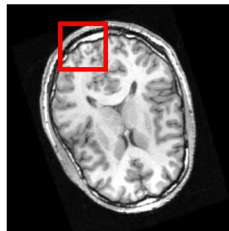
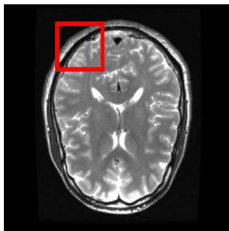


- registration loss : no reference warp needed
- $T(x) = \text{Exp}(v)$: diffeomorphic $\leftarrow$ scaling and squaring layers

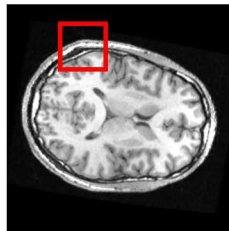
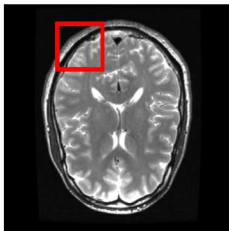
Concat input images for registration ??



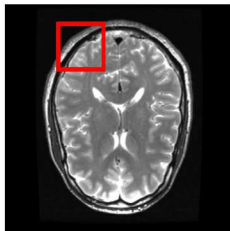
Concat input images for registration ??



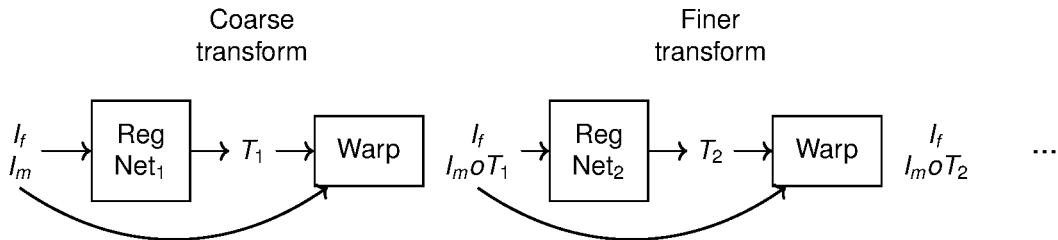
Concat input images for registration ??



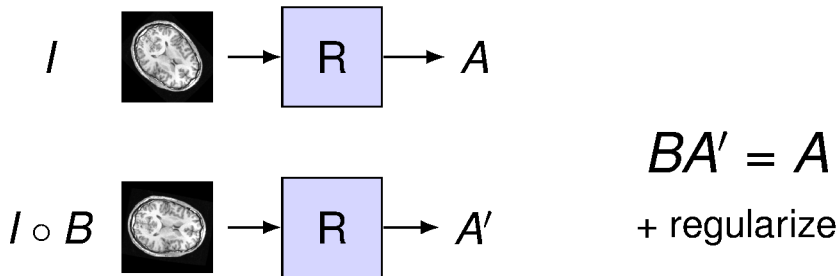
Concat input images for registration ??



Coarse to fine registration

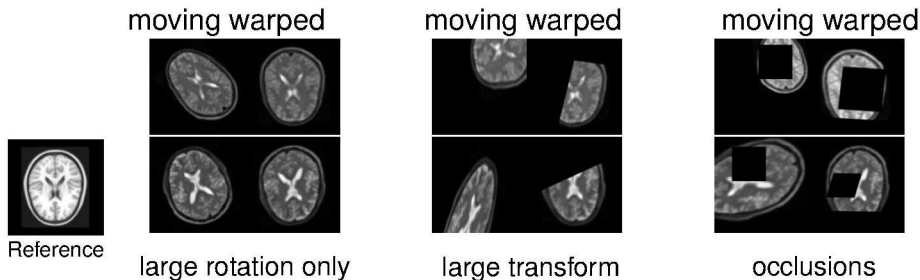


Equivariance as an unsupervised registration loss



Results

Loss	Mat_{err}	θ_{90}
ours wo MI	0.45 ± 0.27	0.00
ours w MI	0.39 ± 0.26	0.00
FSL flirt - MI	2.58 ± 1.21	49.22
Unsupervized - MI	3.63 ± 1.08	50.55
Supervized - L_{ref}	0.32 ± 0.22	0.00



Take home message

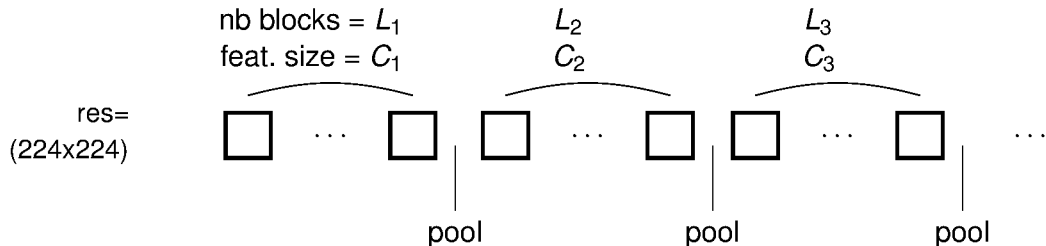
Take home message

- ▶ Do not start your new network from scratch !
 - reuse (pretrained) existing models for your task/organ/modality
- ▶ Architecture is not everything
 - optimization / learning rate
 - data augmentation is very important!!!

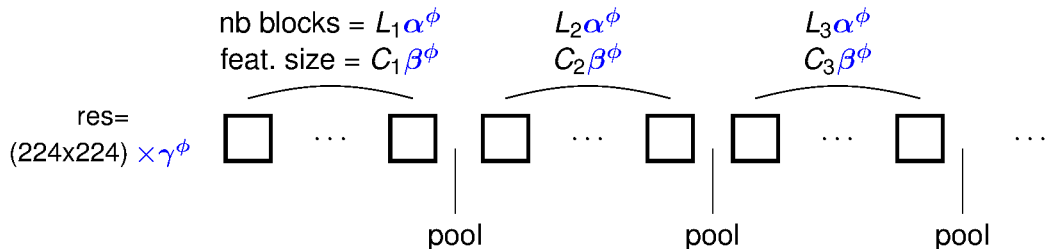
Thank you !!



Efficient Net : compound scaling of networks



Efficient Net : compound scaling of networks



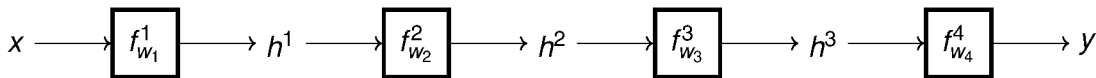
- base network EffNet₁, ($\phi = 1$)
- find α, β, γ :
 - $\phi = 1$
 - optimize accuracy/flops s.t.
 $\alpha \beta^2 \gamma^2 \approx 2$

- More Capacity : change ϕ : EffNet _{ϕ}
- $\text{flops} = \text{flops}_1 \times (\alpha \beta^2 \gamma^2)^\phi$

What about memory ?

```
File "/home/conda/.conda/envs/cuda11.0/lib/python3.8/site-packages/torch/nn/modules/conv.py", line 587, in forward
    return self._conv_forward(input, self.weight, self.bias)
File "/home/conda/.conda/envs/cuda11.0/lib/python3.8/site-packages/torch/nn/modules/conv.py", line 582, in _conv_forward
    return F.conv3d(
RuntimeError: CUDA out of memory. Tried to allocate 9.79 GiB (GPU 0;
11.91 GiB total capacity; 730.73 MiB already allocated; 8.67 GiB free
; 1.21 GiB reserved in total by PyTorch)
```

Where is the memory ?



$$\frac{\partial f^4}{\partial w_4}(h^3, w_4)$$

$$\frac{\partial f^3}{\partial w_3}(h^2, w_3) \longleftarrow \frac{\partial f^4}{\partial h_3}(h^3, w_4)$$

$$\frac{\partial f^2}{\partial w_2}(h^1, w_2) \longleftarrow \frac{\partial f^3}{\partial h_2}(h^2, w_3)$$

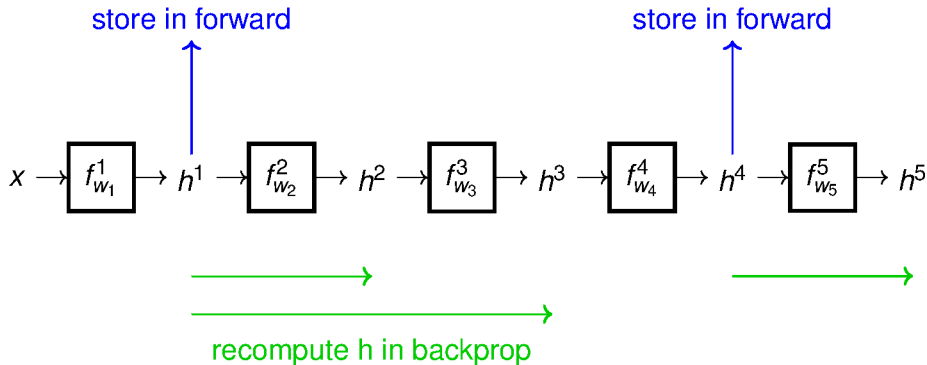
$$\frac{\partial f^1}{\partial w_1}(x, w_1) \longleftarrow \frac{\partial f^2}{\partial h_1}(h^1, w_2)$$

$$\frac{\partial f^1}{\partial x}(x, w_1)$$

First Trick

Reduce the batch size !!!

Second Trick : Checkpointing



Third Trick : Reversible Networks

do no store h_i in forwardrecompute h_i in backprop

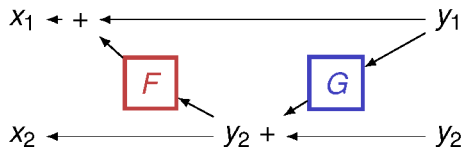
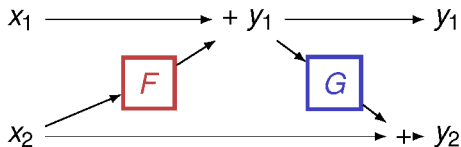
Third options : Reversible Networks

$$y_1 = x_1 + F(x_2)$$

$$y_2 = x_2 + G(y_1)$$

$$x_2 = y_2 - G(y_1)$$

$$x_1 = y_1 - F(x_2)$$



Gomez et al, Neurips 2017, The reversible residual network : Backpropagation without storing activations