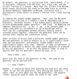
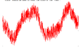
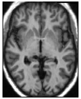
Deep Learning
State of the Art Convolutional Architectures

Michaël Sdika [1]

[1] CNRS, CREATIS UMR 5220

Michaël Sdika
CREATIS - CNRS

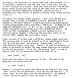
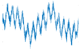
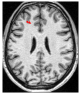
Machine learning

Unstructured data

...

Input $\longrightarrow$ $\boxed{\text{Mapping}}$ $\longrightarrow$ Answer

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

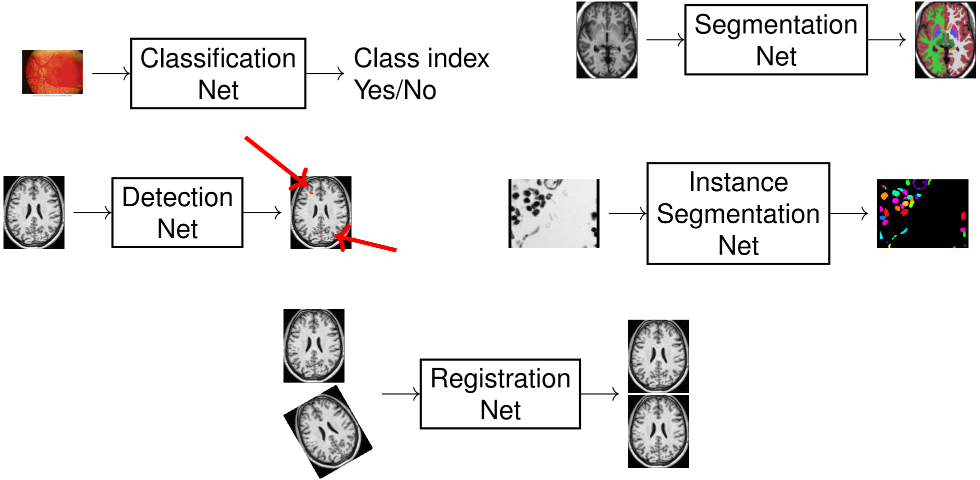
Michaël Sdika
CREATIS - CNRS

Supervized 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 ?

Michaël Sdika
CREATIS - CNRS

5 classes of architectures adressed in this course



Michaël Sdika
CREATIS - CNRS

Outline

Short reminder on MLP and CNN

Architecture for some important applications

- Classifiers
- Encoder / Decoder architectures
- Detection
- Instance Segmentation
- Image Registration

Extra

- What about memory ?

Michaël Sdika
CREATIS - CNRS

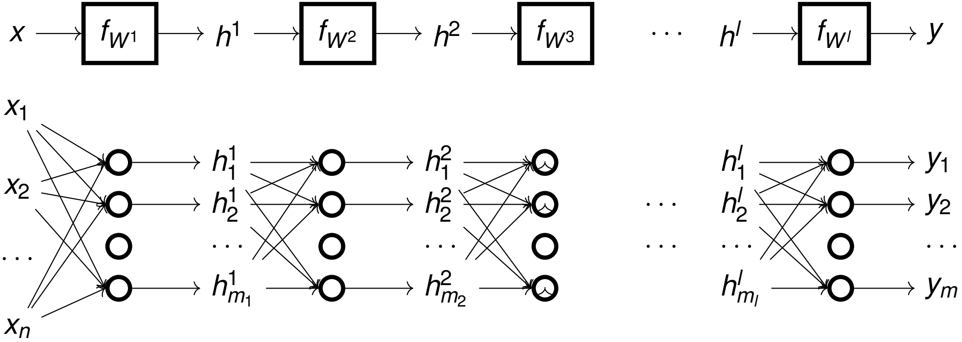
Deep Neural Network

$$x \longrightarrow \boxed{f_{W^1}} \longrightarrow h^1 \longrightarrow \boxed{f_{W^2}} \longrightarrow h^2 \longrightarrow \boxed{f_{W^3}} \quad \cdots \quad h^l \longrightarrow \boxed{f_{W^l}} \longrightarrow y$$

Basic Layers :

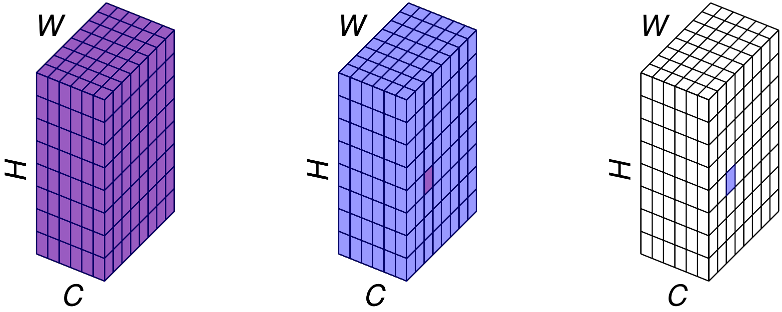
- Linear Layers : Fully Connected / Convolution : mixing features
- Activation layers : introducing nonlinearity
- Pooling layers : spatial aggregation, subsampling
- Normalization layers : stabilizing the training

Multi Layer Perceptron



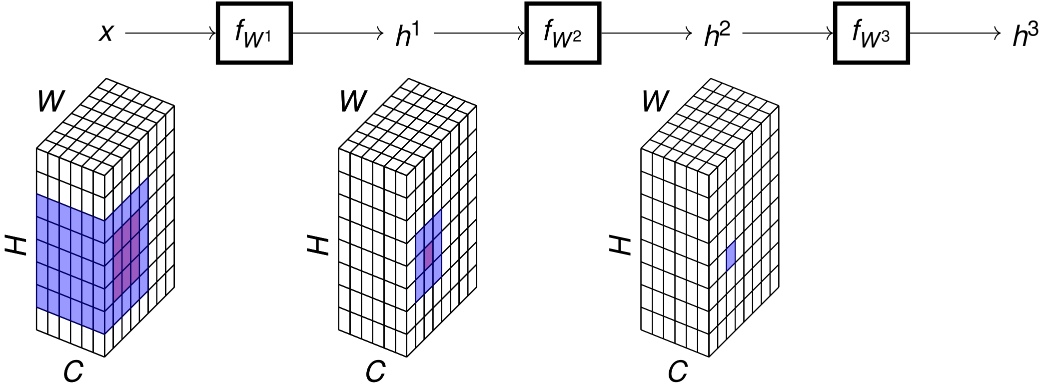
Multi Layer Perceptron

$$x \longrightarrow \boxed{f_{W^1}} \longrightarrow h^1 \longrightarrow \boxed{f_{W^2}} \longrightarrow h^2 \longrightarrow \boxed{f_{W^3}} \longrightarrow h^3$$



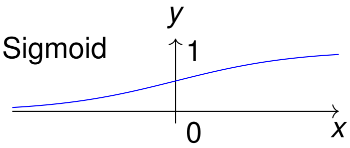
Michaël Sdika
CREATIS - CNRS

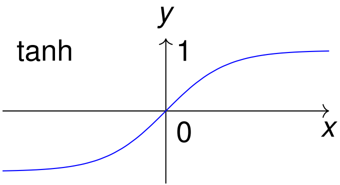
CNN

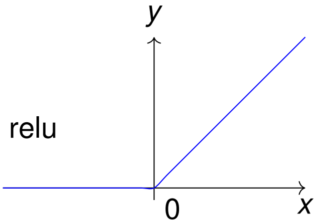


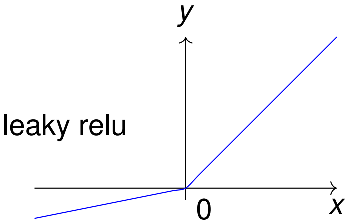
Michaël Sdika
CREATIS - CNRS

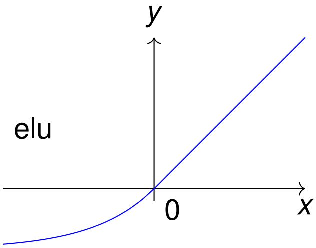
Activation functions



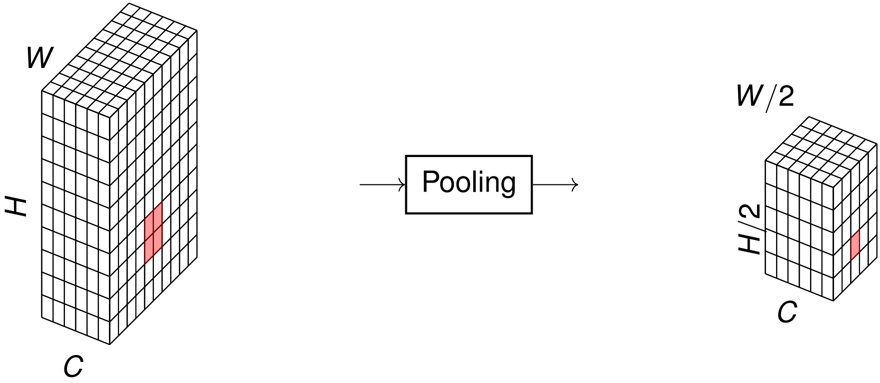








Pooling



Michaël Sdika
CREATIS - CNRS

Normalization

- deep network : need to normalize input x such that x $N(0, 1)$
- Z-normalization
- what about features within the network ?

Michaël Sdika
CREATIS - CNRS

Batch Normalization

$$x \longrightarrow \boxed{\text{BatchNorm}} \longrightarrow y = \gamma \hat{x} + \beta$$

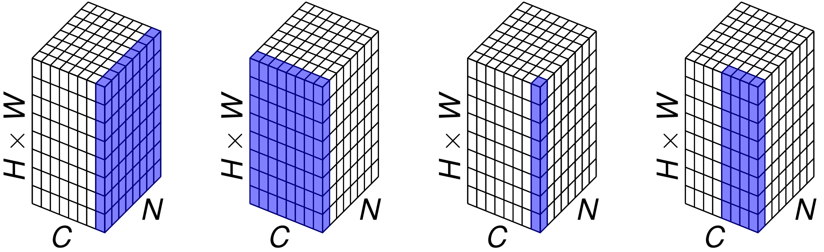
with $\hat{x} = \frac{x-\mu}{\sigma}$

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

Ioffe & Szegedy, ICML 2015, Batch normalization : Accelerating deep network training by reducing internal covariate shift

Michaël Sdika
CREATIS - CNRS

Related Normalization

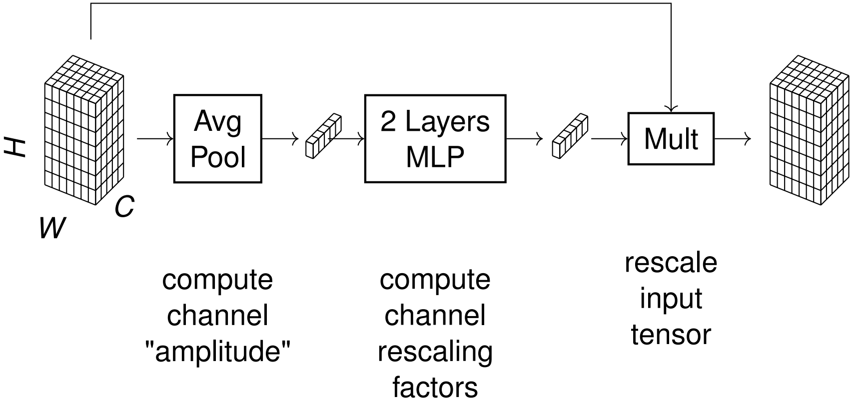


Batch Norm	**Layer Norm**	**Instance Norm**	**Group Norm**
☹ small minibatch	☺ small minibatch	remove contrast style transfer	

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

Michaël Sdika
CREATIS - CNRS

Squeeze and Excitation



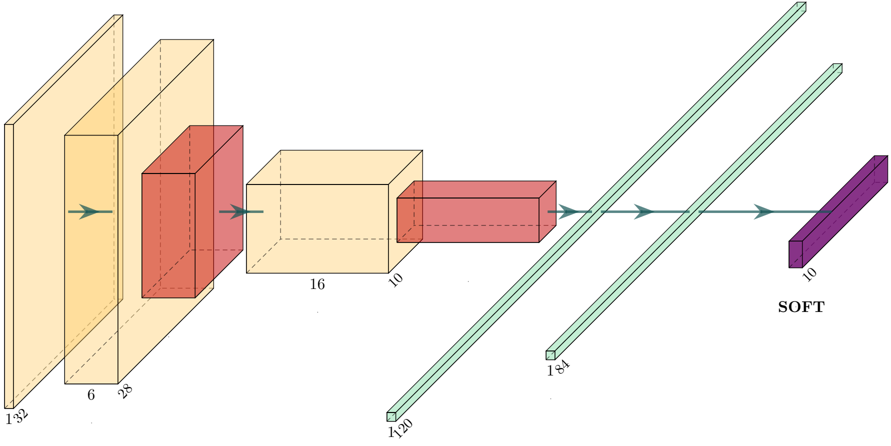
Hu et al CVPR 2018, Squeeze-and-excitation networks
Roy et al, MICCAI 2018, Concurrent Spatial and Channel Squeeze & Excitation in Fully Convolutional Networks

Michaël Sdika
CREATIS - CNRS

Outline

Michaël Sdika
CREATIS - CNRS

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.

Michaël Sdika
CREATIS - CNRS

Image Net

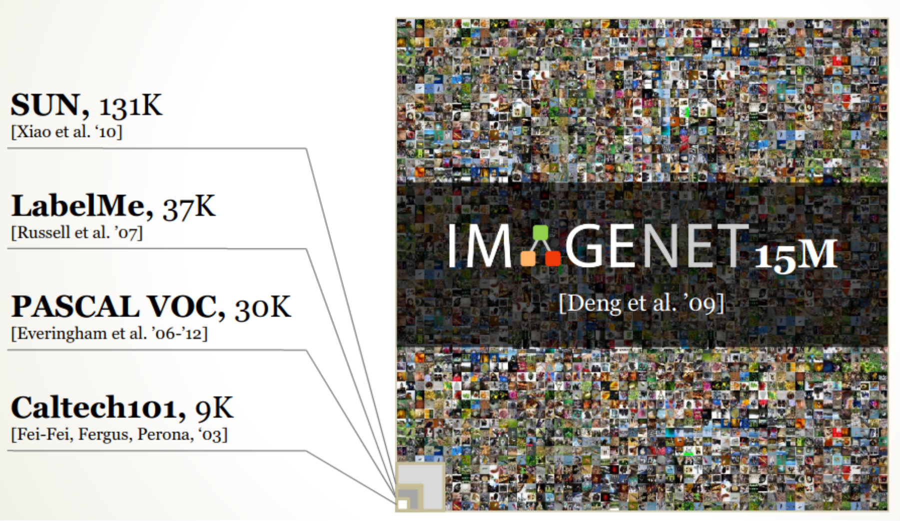
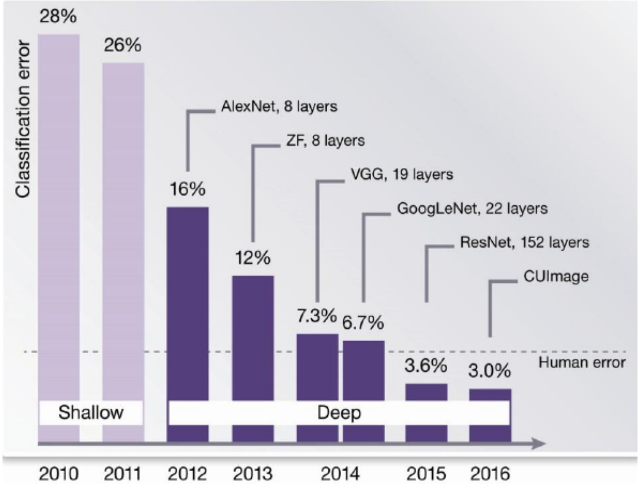


Image Net


28%
26%
16%
12%
7.3%
6.7%
3.6%
3.0%
Classification error
AlexNet, 8 layers
ZF, 8 layers
VGG, 19 layers
GoogLeNet, 22 layers
ResNet, 152 layers
CUImage
Human error
Shallow
Deep
2010
2011
2012
2013
2014
2015
2016

Michaël Sdika
CREATIS - CNRS

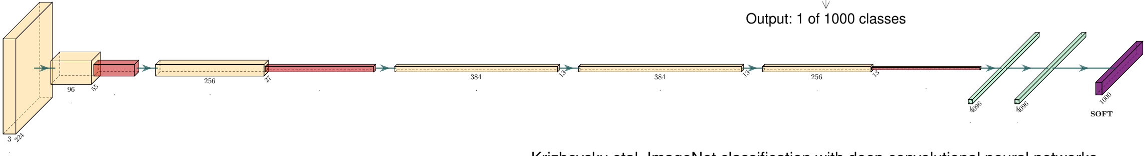
Alex Net

LeNet

Image: 28 (height) × 28 (width) × 1 (channel)
↓
Convolution with 5×5 kernel+2padding:28×28×6
↓ sigmoid
Pool with 2×2 average kernel+2 stride:14×14×6
↓
Convolution with 5×5 kernel (no pad):10×10×16
↓ sigmoid
Pool with 2×2 average kernel+2 stride: 5×5×16
↓ flatten
Dense: 120 fully connected neurons
↓ sigmoid
Dense: 84 fully connected neurons
↓ sigmoid
Dense: 10 fully connected neurons
↓
Output: 1 of 10 classes

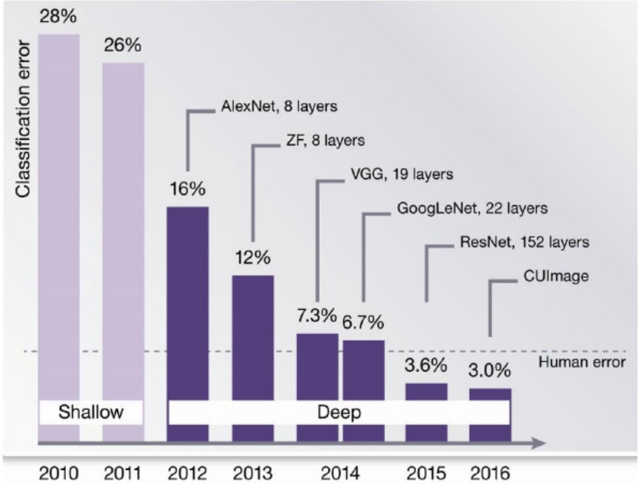
AlexNet

Image: 224 (height) × 224 (width) × 3 (channels)
↓
Convolution with 11×11 kernel+4 stride:54×54×96
↓ ReLu
Pool with 3×3 max. kernel+2 stride: 26×26×96
↓
Convolution with 5×5 kernel+2 pad:26×26×256
↓ ReLu
Pool with 3×3 max. kernel+2 stride:12×12×256
↓
Convolution with 3×3 kernel+1 pad:12×12×384
↓ ReLu
Convolution with 3×3 kernel+1 pad:12×12×384
↓ ReLu
Convolution with 3×3 kernel+1 pad:12×12×256
↓ ReLu
Pool with 3×3 max. kernel+2 stride:5×5×256
↓ flatten
Dense: 4096 fully connected neurons
↓ ReLu, dropout p=0.5
Dense: 4096 fully connected neurons
↓ ReLu, dropout p=0.5
Dense: 1000 fully connected neurons
↓
Output: 1 of 1000 classes



Krizhevsky etal. ImageNet classification with deep convolutional neural networks

Image Net

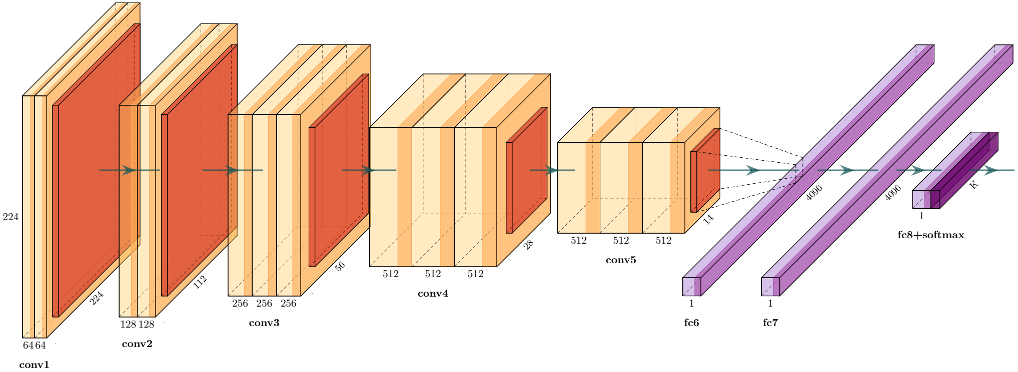

Classification error
28%
26%
16%
12%
7.3%
6.7%
3.6%
3.0%
AlexNet, 8 layers
ZF, 8 layers
VGG, 19 layers
GoogLeNet, 22 layers
ResNet, 152 layers
CUImage
Human error
Shallow
Deep
2010
2011
2012
2013
2014
2015
2016

Michaël Sdika
CREATIS - CNRS

VGG

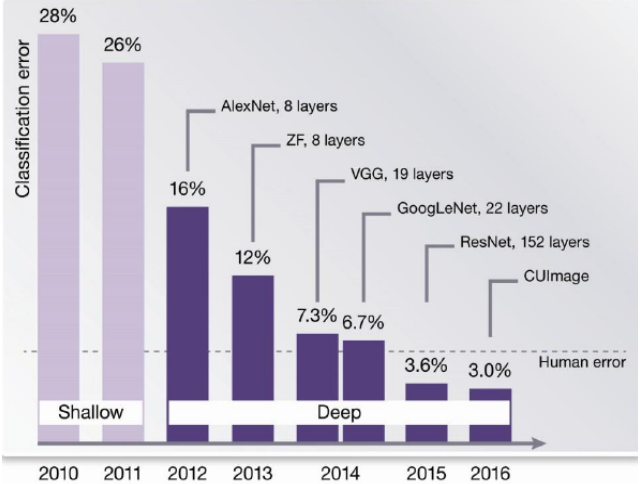
Deeper network

3x3 convolutions



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

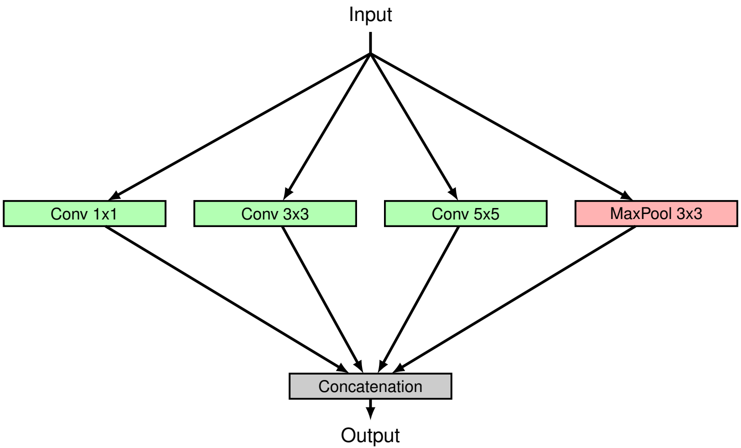
Image Net



Classification error
28%
26%
16%
12%
7.3%
6.7%
3.6%
3.0%
AlexNet, 8 layers
ZF, 8 layers
VGG, 19 layers
GoogLeNet, 22 layers
ResNet, 152 layers
CUImage
Human error
Shallow
Deep
2010
2011
2012
2013
2014
2015
2016

Michaël Sdika
CREATIS - CNRS

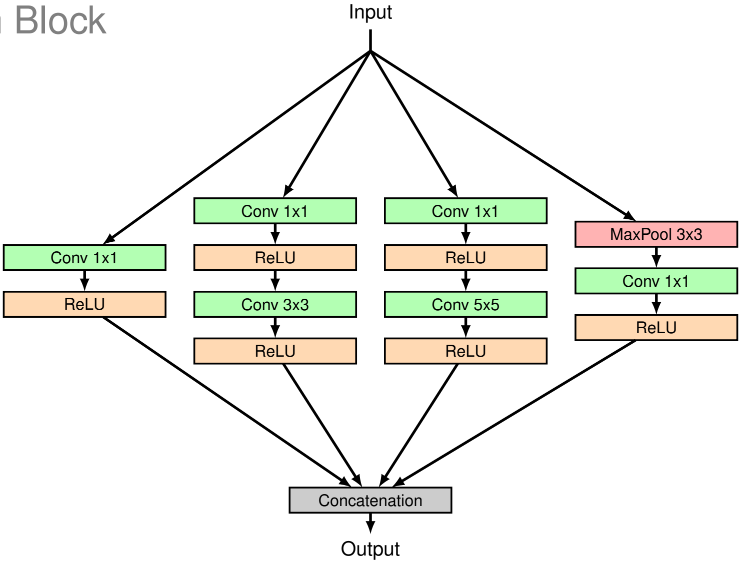
Inception Block



Szegedy et al, CVPR 2015, Going Deeper With Convolutions

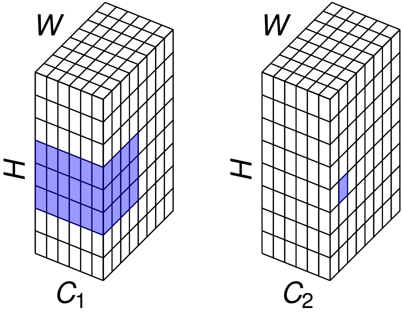
Michaël Sdika
CREATIS - CNRS

Inception Block

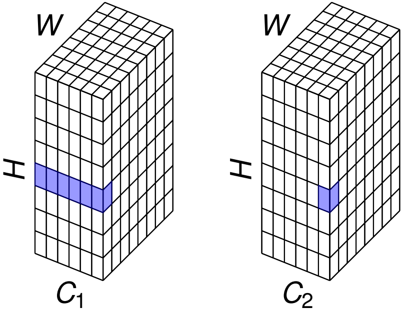


Szegedy et al, CVPR 2015, Going Deeper With Convolutions

1x1 Convolution

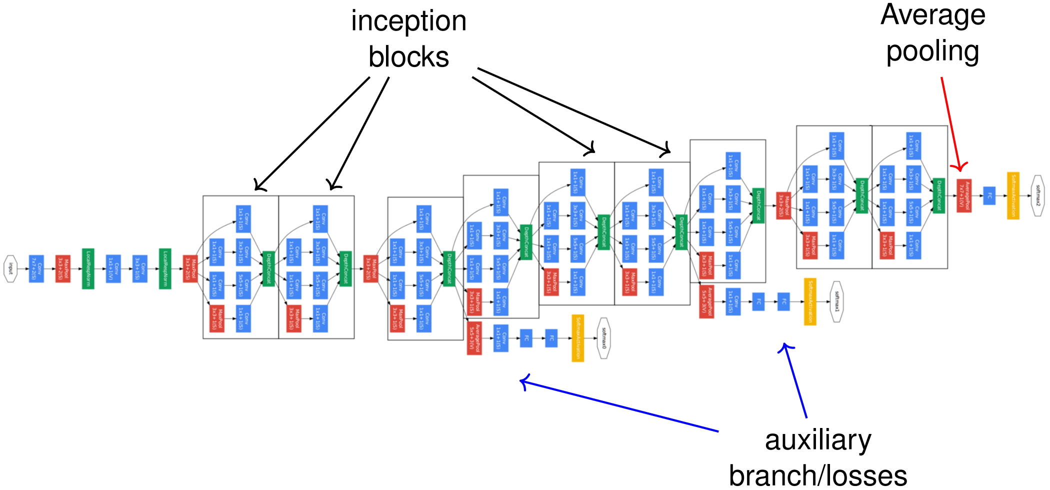


3x3 convolution
receptive field



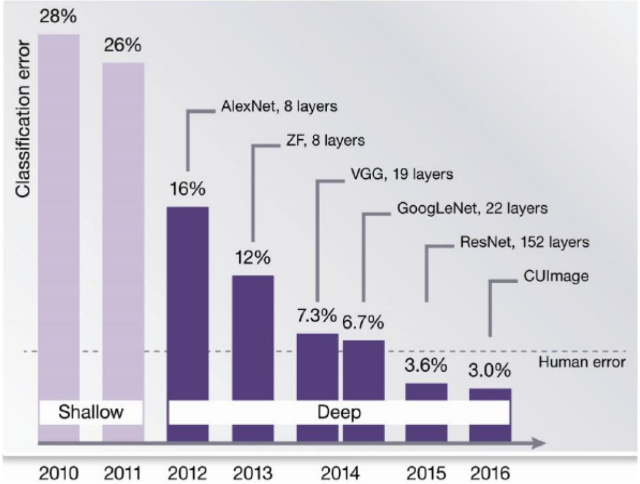
1x1 convolution
receptive field

GoogLe Net



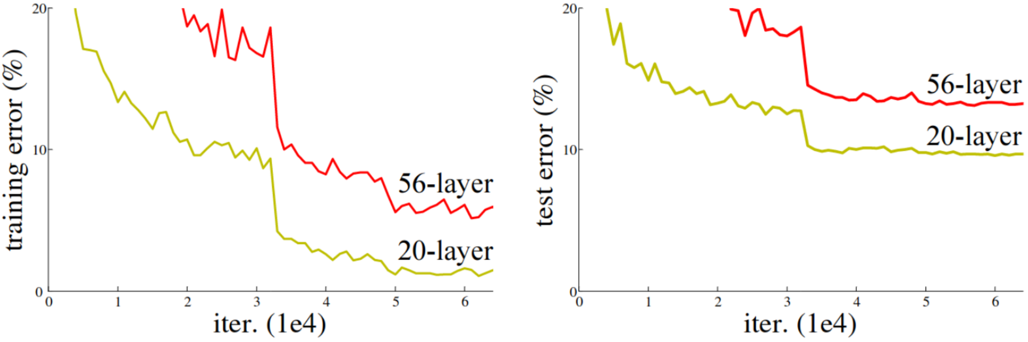
Szegedy et al, CVPR 2015, Going Deeper With Convolutions

Image Net


Classification error
28%
26%
AlexNet, 8 layers
ZF, 8 layers
VGG, 19 layers
GoogLeNet, 22 layers
ResNet, 152 layers
CUImage
16%
12%
7.3%
6.7%
3.6%
3.0%
Human error
Shallow
Deep
2010
2011
2012
2013
2014
2015
2016

Michaël Sdika
CREATIS - CNRS

Going Deeper ? ?



He et al, CVPR 2016, Deep Residual Learning for Image Recognition

Michaël Sdika
CREATIS - CNRS

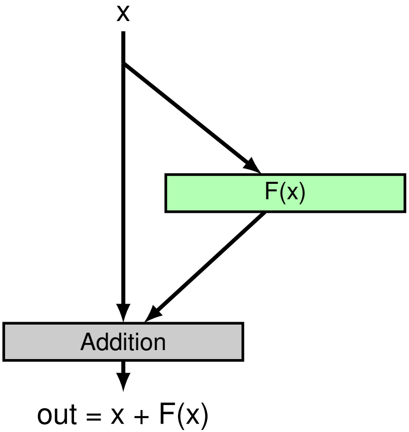
Residual Block

observation :

- more layers ⇒ higher train errors
- Problem is training

Architecture easier to train

Vanishing gradient

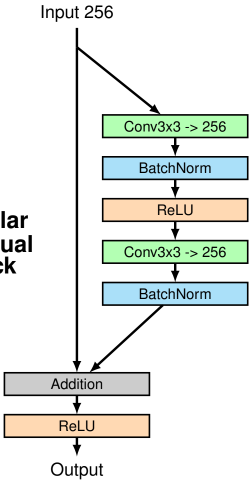


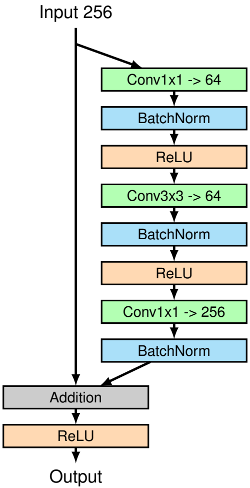
He etal, CVPR 2016, Deep Residual Learning for Image Recognition.

Michaël Sdika
CREATIS - CNRS

Residual Block

Regular Residual Block



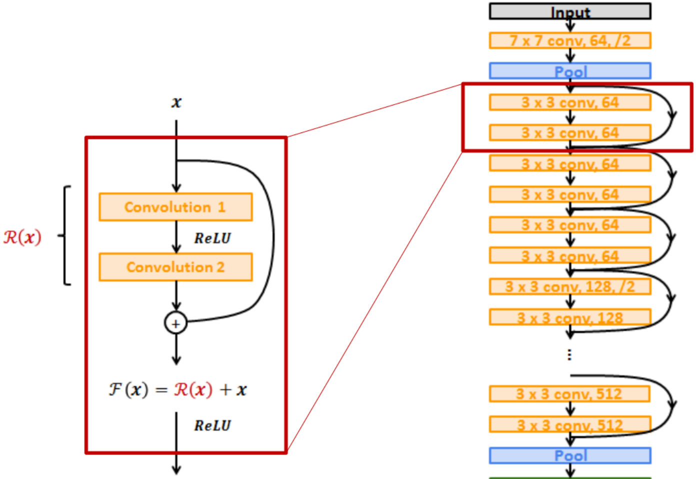


Bottleneck

He et al, CVPR 2016, Deep Residual Learning for Image Recognition

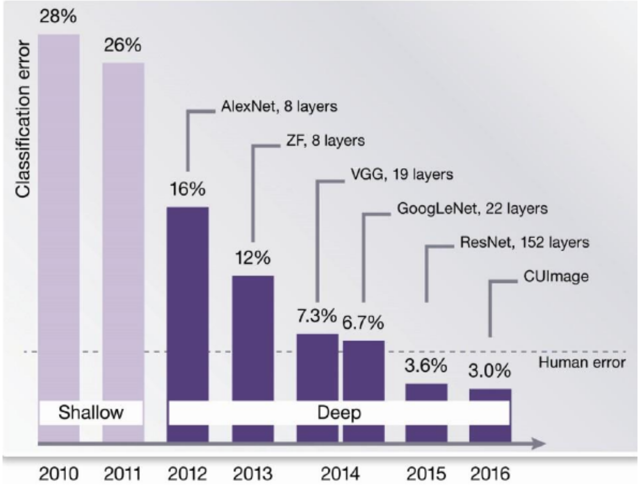
Michaël Sdika
CREATIS - CNRS

Res Net



He et al, CVPR 2016, Deep Residual Learning for Image Recognition

Image Net


28%
26%
16%
12%
7.3%
6.7%
3.6%
3.0%
Classification error
AlexNet, 8 layers
ZF, 8 layers
VGG, 19 layers
GoogLeNet, 22 layers
ResNet, 152 layers
CUImage
Human error
Shallow
Deep
2010
2011
2012
2013
2014
2015
2016

Dense Block

$$x_0 \rightarrow \boxed{H_1} \rightarrow x_1 \rightarrow \boxed{H_2} \rightarrow x_2 \rightarrow \boxed{H_3} \rightarrow x_3 \rightarrow \boxed{H_4} \rightarrow x_4 \qquad \cdots$$

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

Dense block :
$x_l = H_l\left([x_0, x_1, \ldots, x_{l-1}]\right)$

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

Dense Block

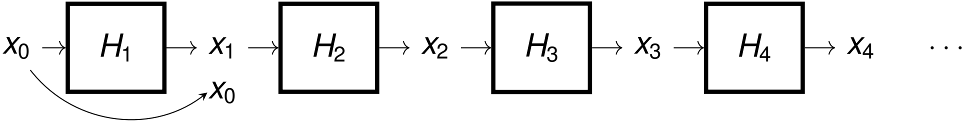
$$x_0 \rightarrow \boxed{H_1} \rightarrow x_1 \rightarrow \boxed{H_2} \rightarrow x_2 \rightarrow \boxed{H_3} \rightarrow x_3 \rightarrow \boxed{H_4} \rightarrow x_4 \qquad \cdots$$

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

Dense block :
$x_l = H_l\left([x_0, x_1, \ldots, x_{l-1}]\right)$

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

Dense Block

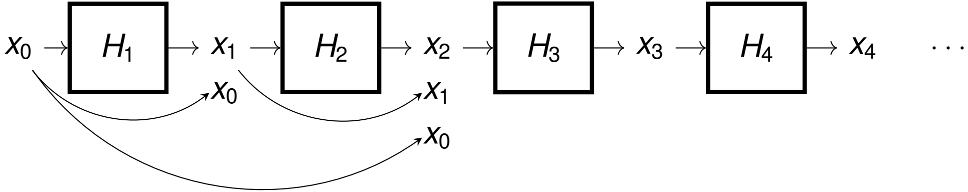


Res block :
$x_l = x_{l-1} + H_l(x_{l-1})$

Dense block :
$x_l = H_l\left([x_0, x_1, \ldots, x_{l-1}]\right)$

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

Dense Block



Res block :

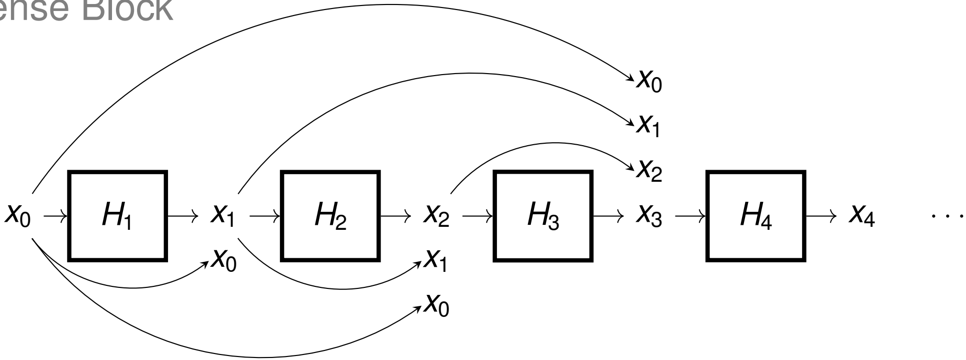
$x_l = x_{l-1} + H_l(x_{l-1})$

Dense block :

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

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

Dense Block

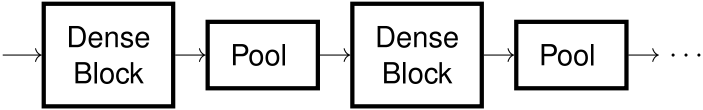


Res block :
$x_l = x_{l-1} + H_l(x_{l-1})$

Dense block :
$x_l = H_l\left([x_0, x_1, \ldots, x_{l-1}]\right)$

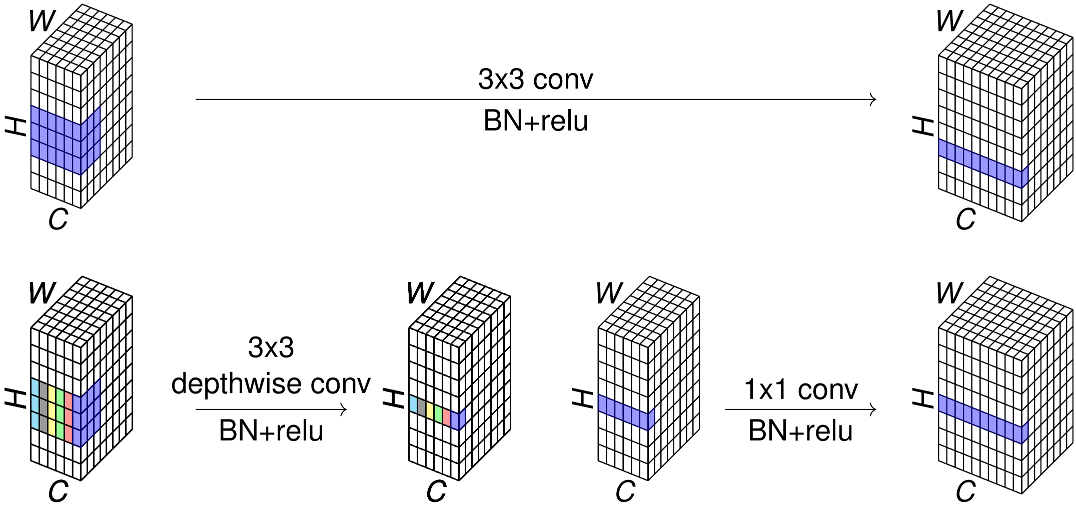
Gao, et al. CVPR 2017, Densenet : densely connected convolutional networks

Dense Net



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

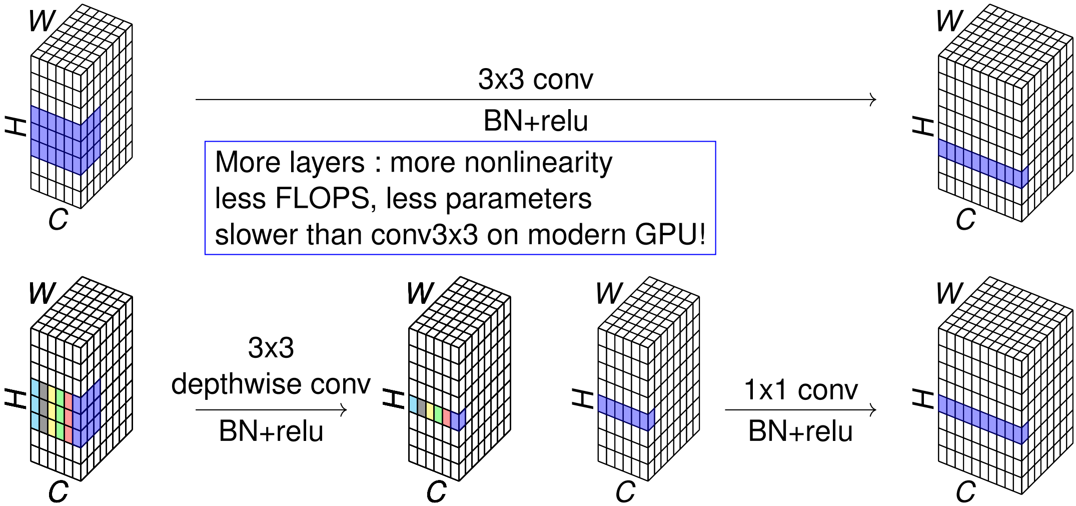
Mobile Net V1 : depthwize conv



L. Sifre. Rigid-motion scattering for image classification. PhD thesis, Ph. D. thesis, 2014. 1, 3
Howard et al, arxiv 2017, MobileNets : Efficient Convolutional Neural Networks for Mobile Vision Applications

Michaël Sdika
CREATIS - CNRS

Mobile Net V1 : depthwize conv

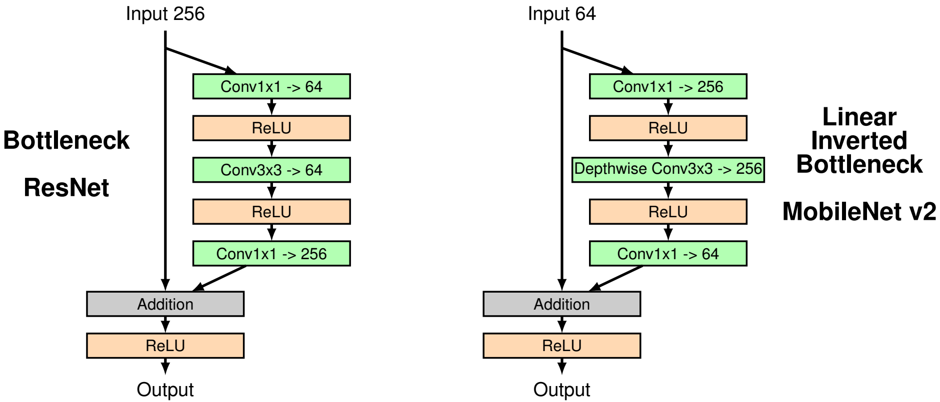


More layers : more nonlinearity
less FLOPS, less parameters
slower than conv3x3 on modern GPU!

L. Sifre. Rigid-motion scattering for image classification. PhD thesis, Ph. D. thesis, 2014. 1, 3
Howard et al, arxiv 2017, MobileNets : Efficient Convolutional Neural Networks for Mobile Vision Applications

Michaël Sdika
CREATIS - CNRS

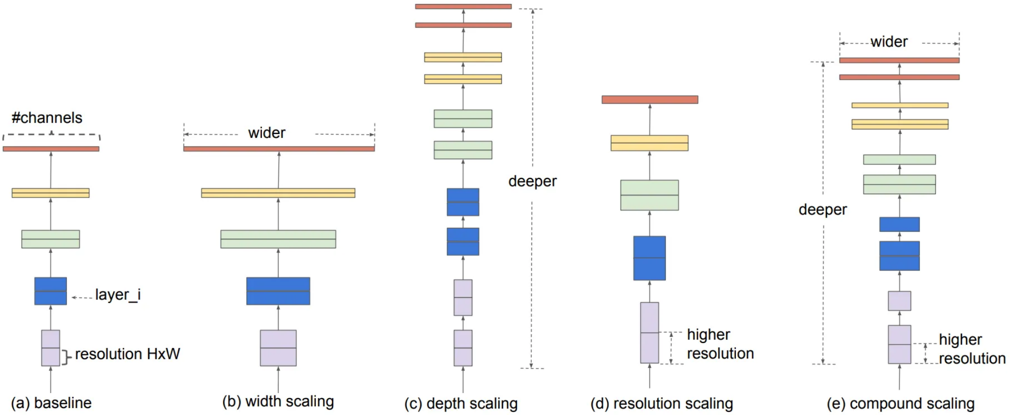
Mobile Net V2 : inverted bottleneck



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

Michaël Sdika
CREATIS - CNRS

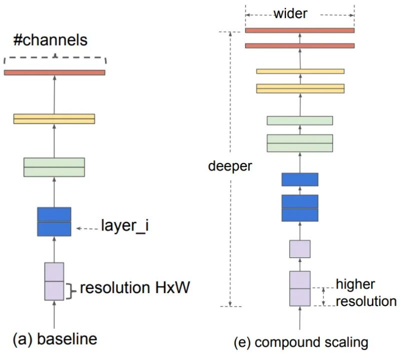
Efficient Net : compound scaling of networks



Tan and Le. PMLR 2019, Efficientnet : Rethinking model scaling for convolutional neural networks

Michaël Sdika
CREATIS - CNRS

Efficient Net : compound scaling of networks



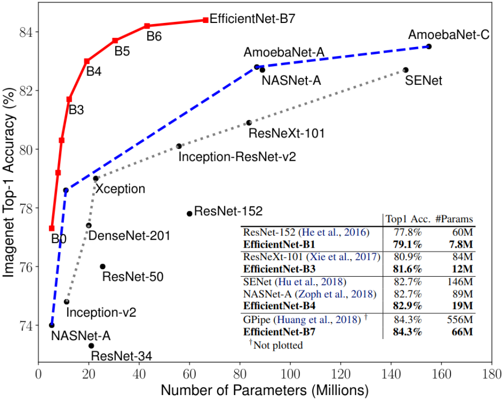
- depth, width, resolution for B1
  - $d_1 = \alpha d_0$
  - $w_1 = \beta w_0$
  - $r_1 = \gamma r_0$
- grid search for α, β, γ
- Bk : α^k, β^k, γ^k

Efficient Net v2 :
architecture grid search fo B0

Tan and Le. PMLR 2019, Efficientnet : Rethinking model scaling for convolutional neural networks
Tan and Le. ICML 2021, EfficientNetV2 : Smaller Models and Faster Training

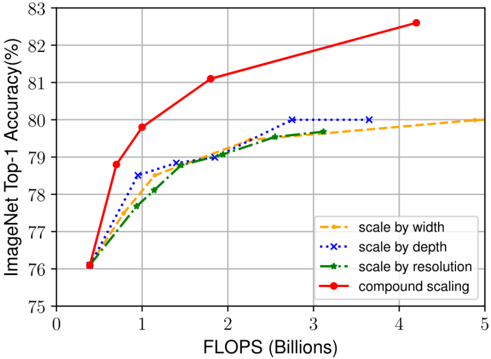
Michaël Sdika
CREATIS - CNRS

Efficient Net



	Top1 Acc.	#Params
ResNet-152 (He et al., 2016)	77.8%	60M
EfficientNet-B1	**79.1%**	**7.8M**
ResNeXt-101 (Xie et al., 2017)	80.9%	84M
EfficientNet-B3	**81.6%**	**12M**
SENet (Hu et al., 2018)	82.7%	146M
NASNet-A (Zoph et al., 2018)	82.7%	89M
EfficientNet-B4	**82.9%**	**19M**
GPipe (Huang et al., 2018) †	84.3%	556M
EfficientNet-B7	**84.3%**	**66M**

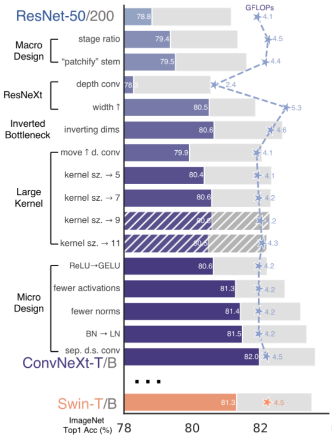
†Not plotted

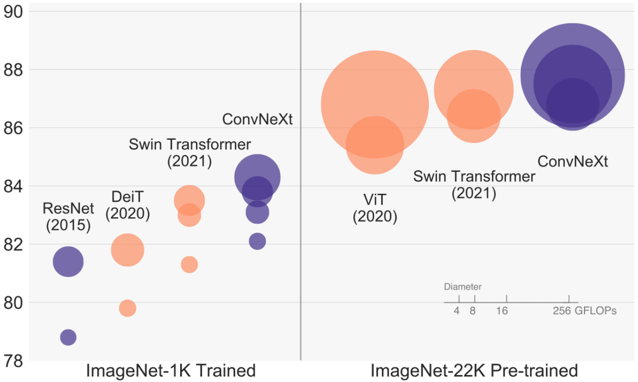


Tan and Le. PMLR 2019, Efficientnet : Rethinking model scaling for convolutional neural networks

Michaël Sdika
CREATIS - CNRS

ConvNext





Liu etal, CVPR 2022, A ConvNet for the 2020s

Michaël Sdika
CREATIS - CNRS

Outline

Michaël Sdika
CREATIS - CNRS

Encoder/Decoder architecture

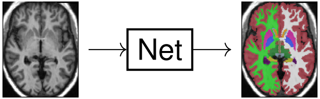


Image Segmentation

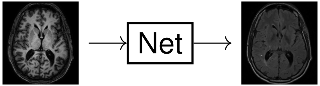
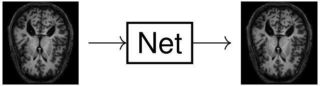


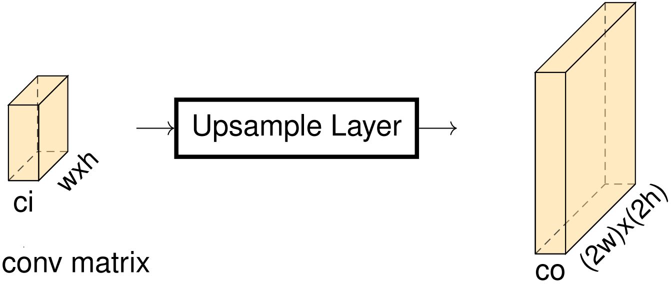
Image Synthesis, Domain adaptation



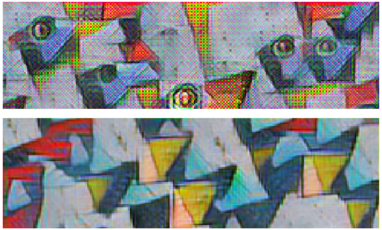
Denoising

Michaël Sdika
CREATIS - CNRS

Upsampling Layer

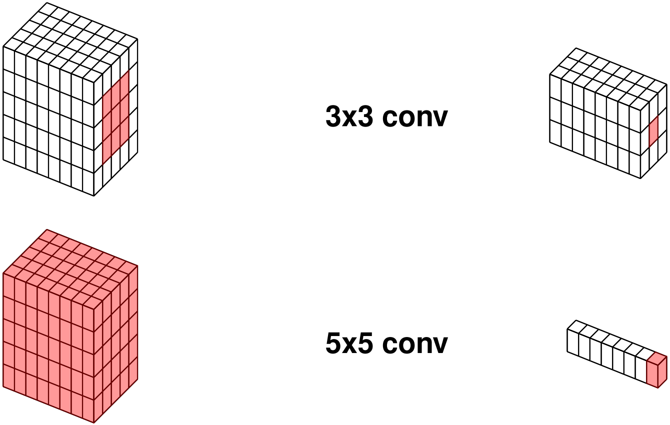


- deconv
  - transpose of strided conv matrix
  - learn the upsampling coefficient
- unpool :
  - upsample on maxpool indices
- interpolation
  - bi/tri linear
  - no chessboard artifact



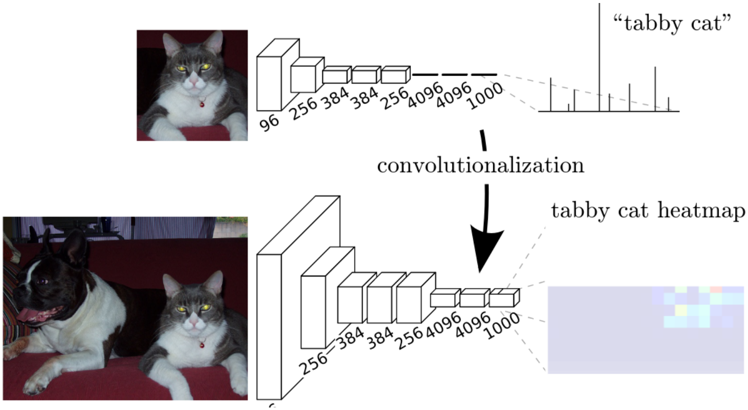
distill.pub/2016/deconv-checkerboard

Fully Convolutional Network : FC as convolution



Michaël Sdika
CREATIS - CNRS

Fully Convolutional Network : FC as convolution

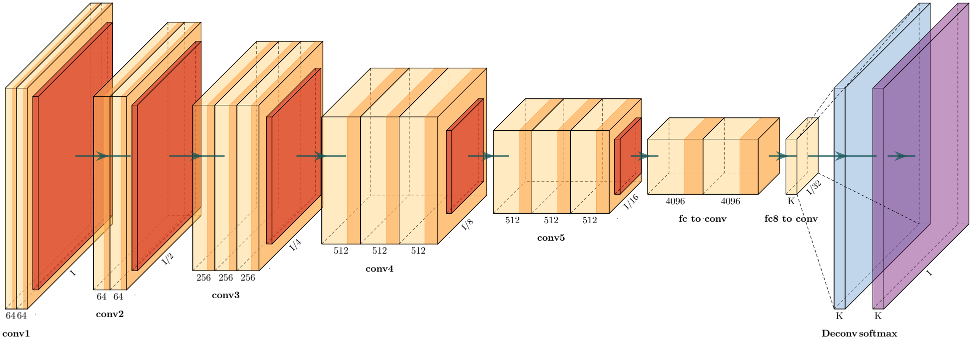


- use kernels that cover their entire input regions

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

Michaël Sdika
CREATIS - CNRS

Fully Convolutional Network

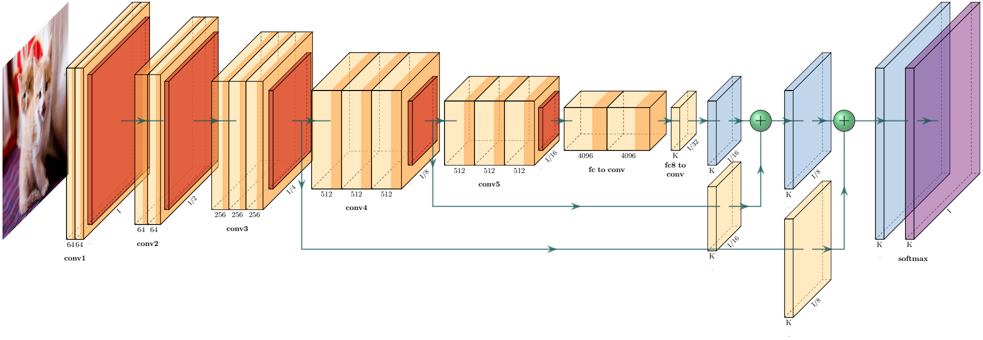


- deconv layer + pixelwize cross entropy

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

Michaël Sdika
CREATIS - CNRS

Fully Convolutional Network

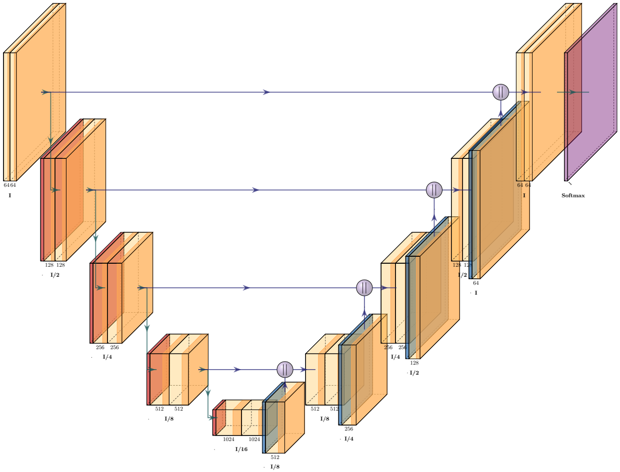


- progressive upsampling + reuse fine scale features

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

Michaël Sdika
CREATIS - CNRS

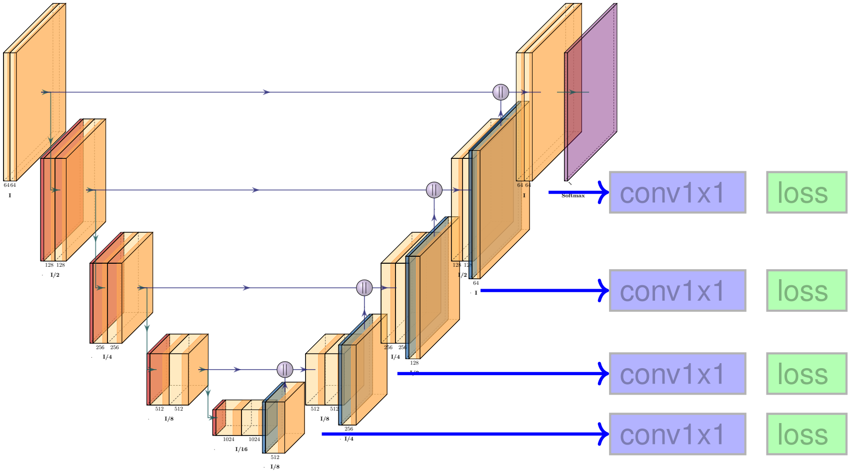
Unet



Ronneberger et al, MICCAI 2015. U-net : Convolutional networks for biomedical image segmentation

Michaël Sdika
CREATIS - CNRS

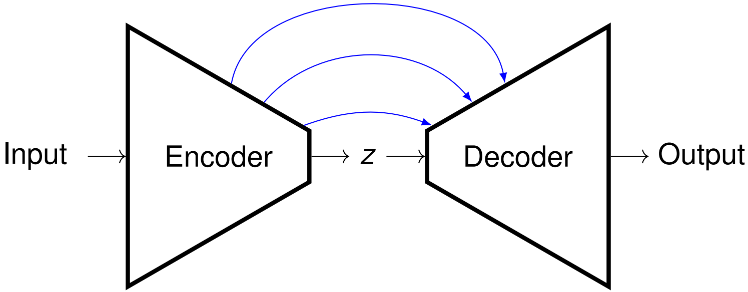
Unet



Ronneberger et al, MICCAI 2015. U-net : Convolutional networks for biomedical image segmentation

Michaël Sdika
CREATIS - CNRS

Encoder / Decoder



Tiramisu Net :
* conv $\rightarrow$ dense block

Unet+ / Unet++ :
* add skip connection across scale

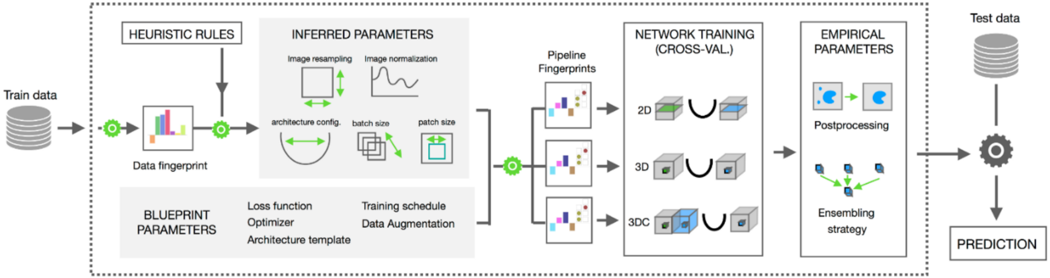
Eff-UNet :
* Encoder is efficient net
* standard unet Decoder

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

Michaël Sdika
CREATIS - CNRS

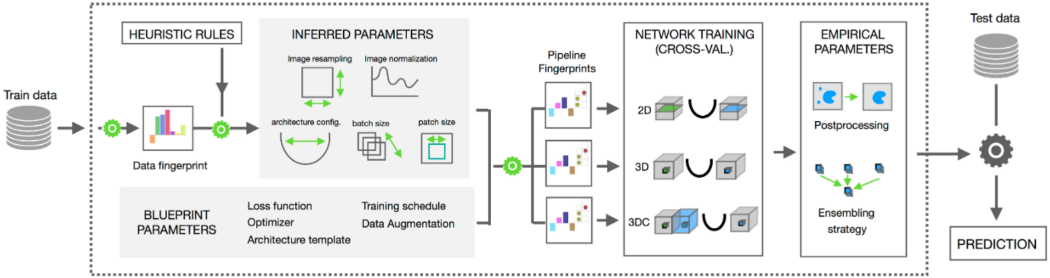
nn-Unet : self configuration



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

Michaël Sdika
CREATIS - CNRS

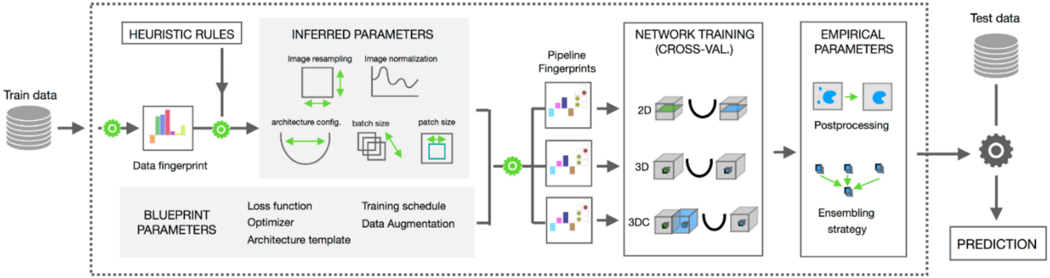
nn-Unet : self configuration



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

Michaël Sdika
CREATIS - CNRS

nn-Unet : self configuration

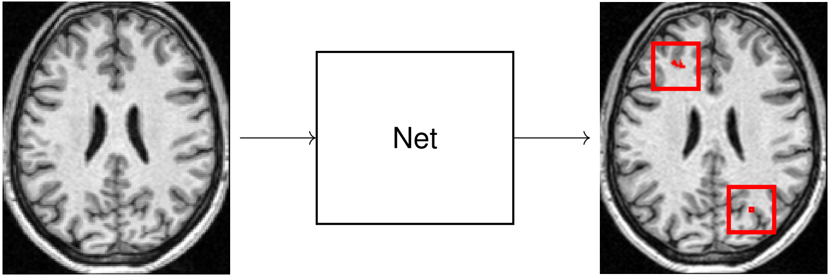


Isensee, et al. Nature 2021, nnU-Net : a self-configuring method for deep learning-based biomedical image segmentation

Michaël Sdika
CREATIS - CNRS

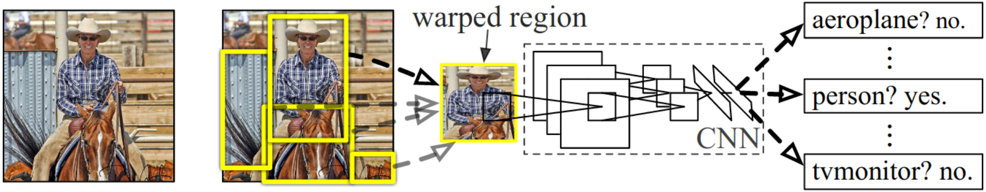
Outline

Object Detection



Michaël Sdika
CREATIS - CNRS

R-CNN



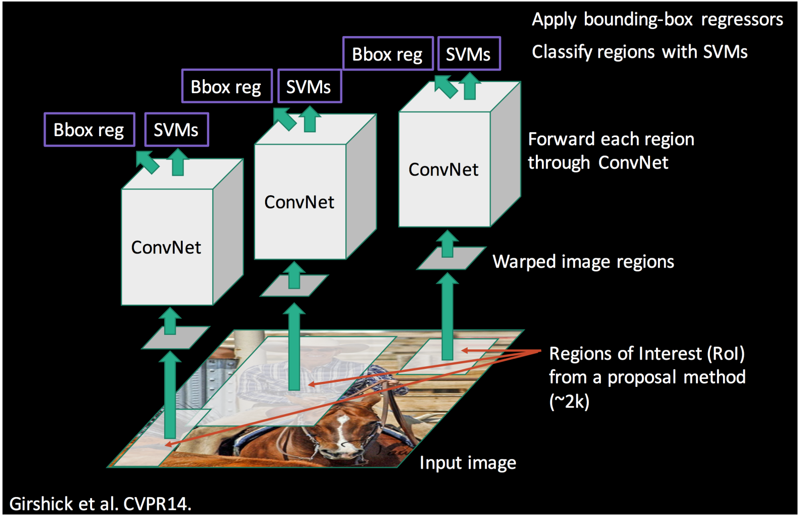
- regions extractor (non deep)
- for each region
  - deep feature
  - classif + box regression

$\rightarrow$ **very slow**

Girshick, et al. CVPR 2015, Rich feature hierarchies for accurate object detection and semantic segmentation

Michaël Sdika
CREATIS - CNRS

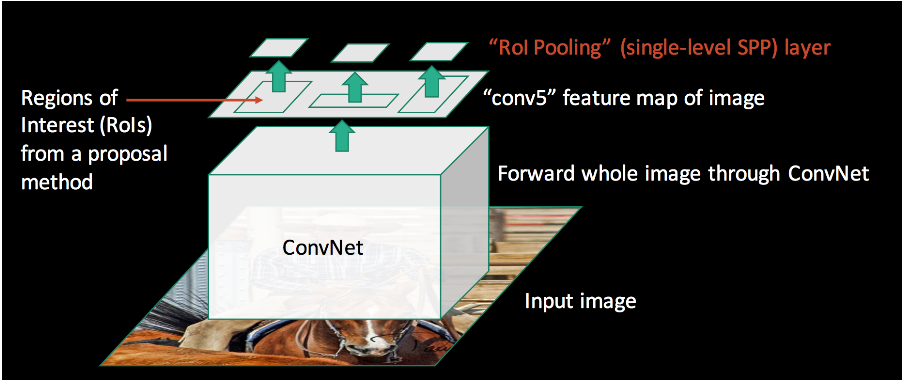
R-CNN



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)

Michaël Sdika
CREATIS - CNRS

Fast RCNN

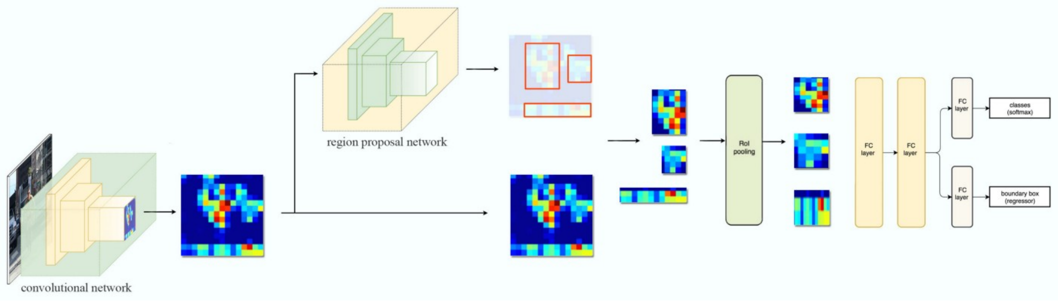


- all the feature computed at once

Girshick, ICCV 2015, Fast r-cnn
(credit : jhui.github.io/2017/03/15/Fast-R-CNN-and-Faster-R-CNN)

Michaël Sdika
CREATIS - CNRS

Faster RCNN

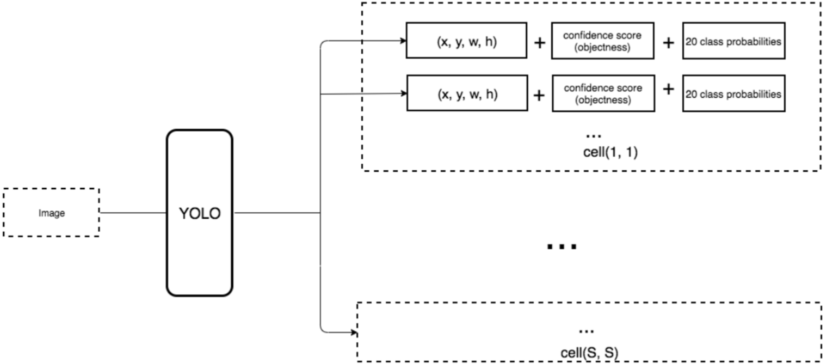


- DEEP region proposal network :
  for each position in the feature map, output
  - k proba : object vs non object
  - k offset for bounding box proba

, Shaoqing, et al. INIPS 2015. Faster r-cnn : Towards real-time object detection with region proposal networks
(credit : jhui.github.io/2017/03/15/Fast-R-CNN-and-Faster-R-CNN)

Michaël Sdika
CREATIS - CNRS

YOLO (You Only Look Once)

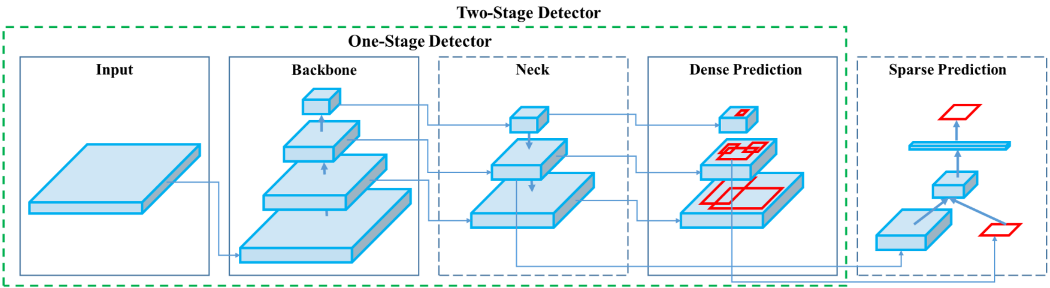


- yolo V1 : CNN $\rightarrow$ pb with small object
- yolo V2, V3 : Unet

Redmon et al, CVPR 2016, You Only Look Once : Unified, Real-Time Object Detection
Redmon, YOLOv3 : An Incremental Improvement

Michaël Sdika
CREATIS - CNRS

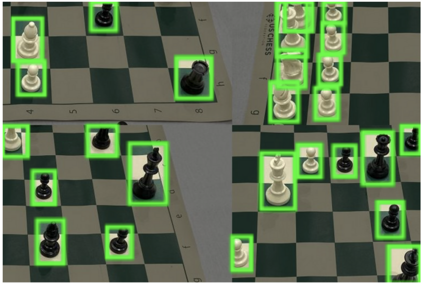
YOLO (You Only Look Once)

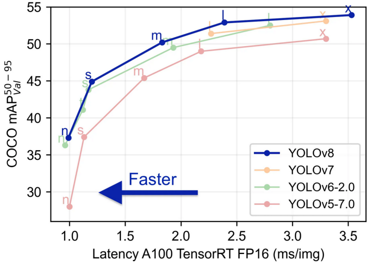


Bochkovskiy et al, arxiv 2020, Yolov4 : Optimal speed and accuracy of object detection

Michaël Sdika
CREATIS - CNRS

YOLO (You Only Look Once)





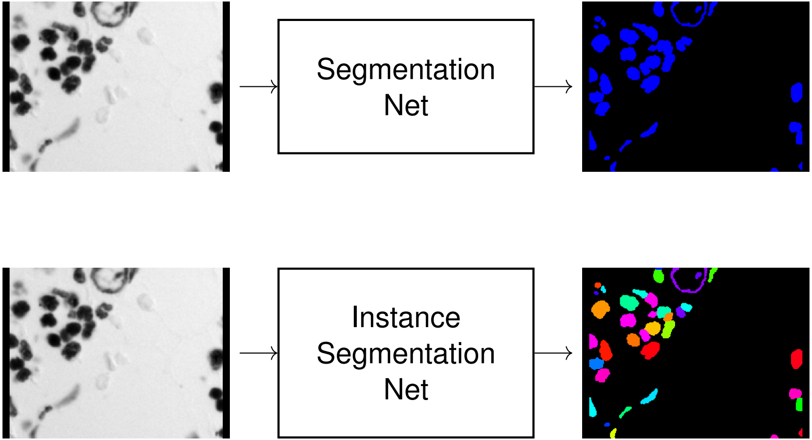
Architecture	mAP@50	GPU Latency
YOLOv8	0.62	1.3ms
EfficientDet	0.47	-
Faster R-CNN	0.41	54ms
YOLOv5	0.58	2.8ms

https ://ultralytics.com/yolov8
https ://medium.com/@rustemgal/yolov8-efficientdet-faster-r-cnn-or-yolov5-for-remote-sensing-12487c40ef68
https ://blog.roboflow.com/whats-new-in-yolov8/#yolov8-architecture-a-deep-dive

Outline

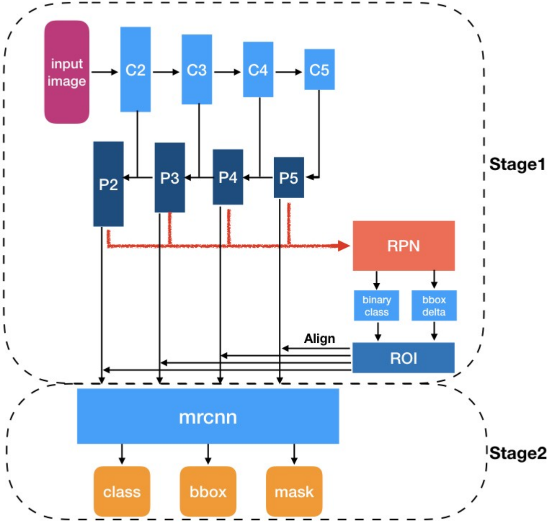
Michaël Sdika
CREATIS - CNRS

Instance Segmentation



Michaël Sdika
CREATIS - CNRS

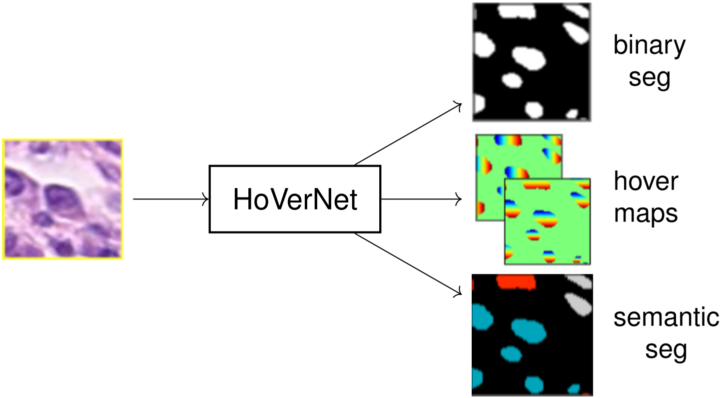
Mask R-CNN



He, Kaiming, et al, ICCV 2017, Mask r-cnn
https ://alittlepain833.medium.com/simple-understanding-of-mask-rcnn-134b5b330e95

Michaël Sdika
CREATIS - CNRS

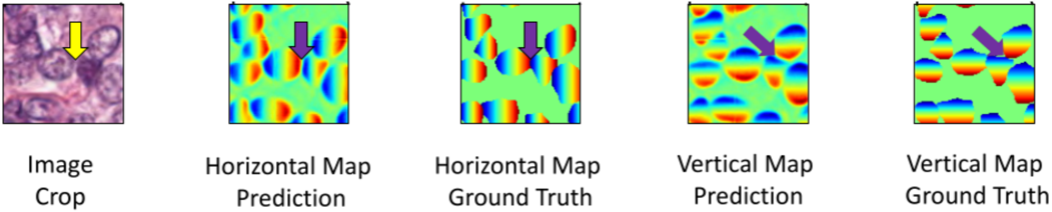
HoVerNet



inside : normalized signed distance to cell center
outside : 0

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

HoVerNet



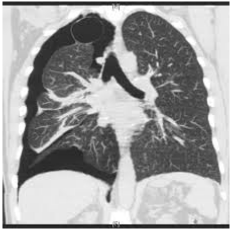
Graham, et al. "Hover-net : Simultaneous segmentation and classification of nuclei in multi-tissue histology images." Medical image analysis, 2019

Michaël Sdika
CREATIS - CNRS

Outline

Michaël Sdika
CREATIS - CNRS

Motion/Registration



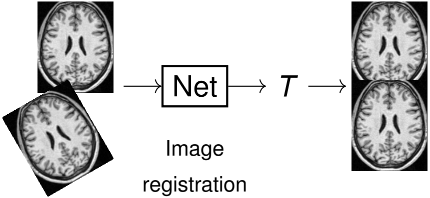
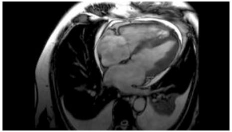
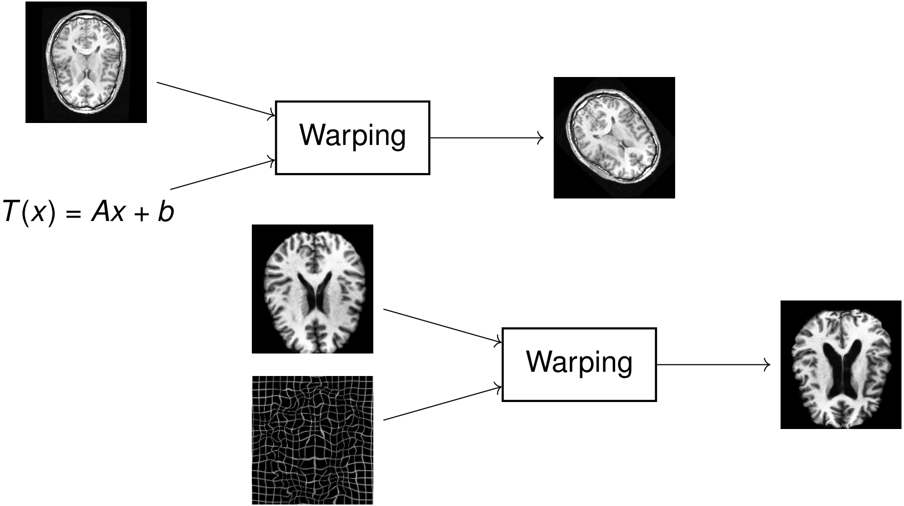


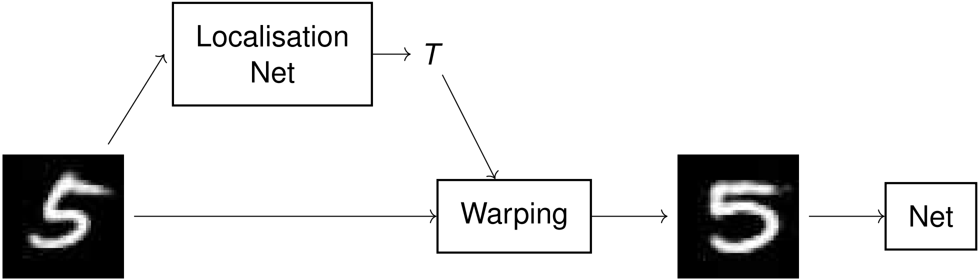
Image registration



Warping Layer



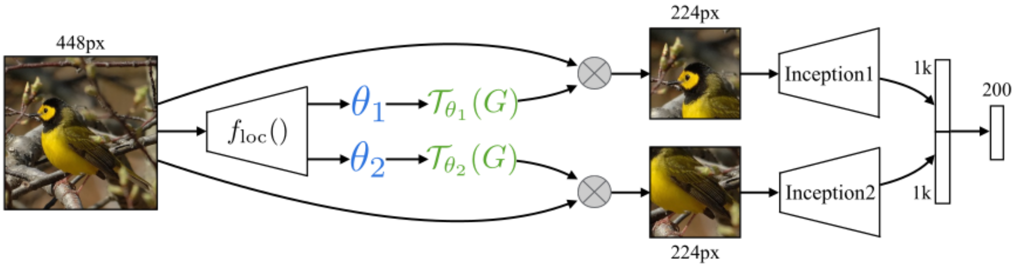
Spatial Transformer Networks



Jaderberg et al, NIPS 2015, Spatial Transformer Networks

Michaël Sdika
CREATIS - CNRS

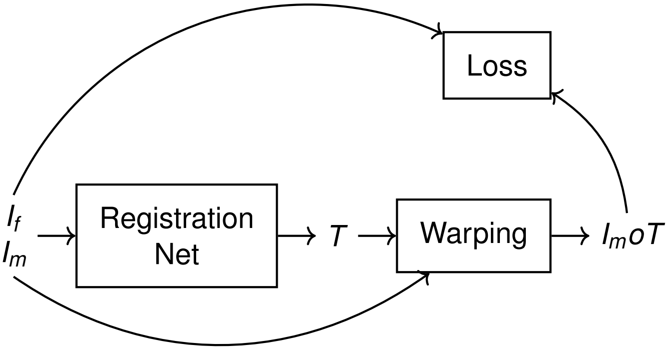
Spatial Transformer Networks



Jaderberg et al, NIPS 2015, Spatial Transformer Networks

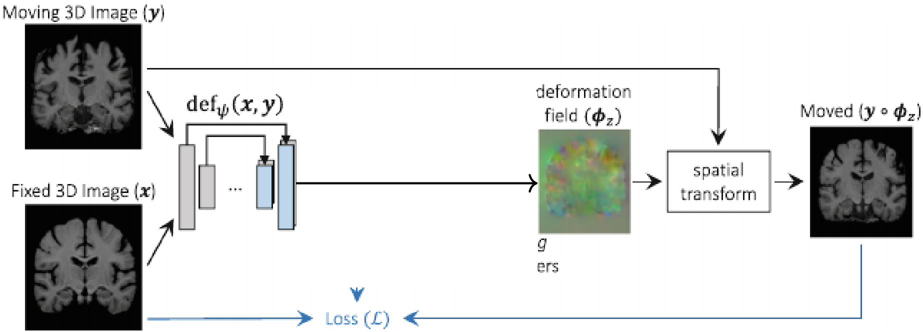
Michaël Sdika
CREATIS - CNRS

Image registration with deep learning



Michaël Sdika
CREATIS - CNRS

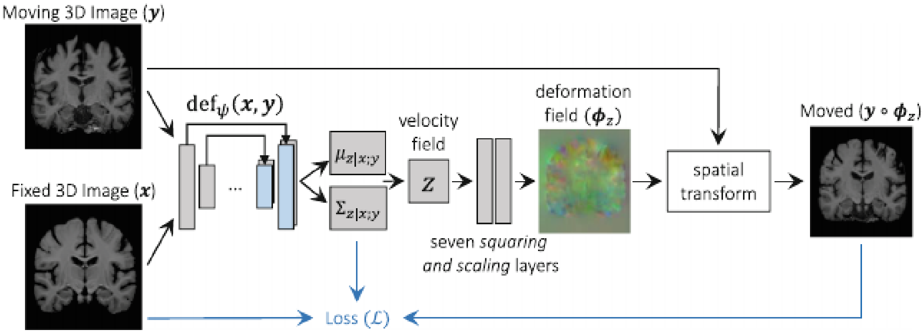
Unsupervized learning, VoxelMorph



- registration loss : no reference warp needed

Balakrishnan et al. IEEE TMI 2019, VoxelMorph : a learning framework for deformable medical image registration
Dalca et al, MICCAI 2018, Unsupervised learning for fast probabilistic diffeomorphic registration

Unsupervized learning, VoxelMorph

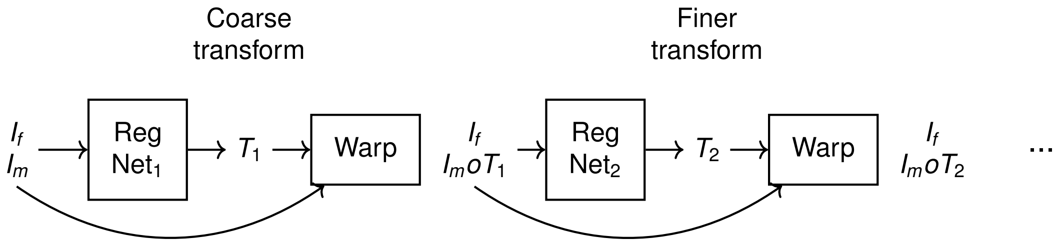


- registration loss : no reference warp needed
- $T(x) = Exp(v)$: diffeomorphic $\longleftarrow$ scaling and squaring layers

Balakrishnan et al. IEEE TMI 2019, VoxelMorph : a learning framework for deformable medical image registration
Dalca et al, MICCAI 2018, Unsupervised learning for fast probabilistic diffeomorphic registration

Michaël Sdika
CREATIS - CNRS

Coarse to fine registration



Bob D. de Vos et al, MEDIA 2019, A Deep Learning Framework for Unsupervised Affine and Deformable Image Registration

Michaël Sdika
CREATIS - CNRS

Outline

Short reminder on MLP and CNN

Architecture for some important applications

Extra

- What about memory ?

Michaël Sdika
CREATIS - CNRS

What about memory ?

```
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 ?

$$x \longrightarrow \boxed{f^1_{w_1}} \longrightarrow h^1 \longrightarrow \boxed{f^2_{w_2}} \longrightarrow h^2 \longrightarrow \boxed{f^3_{w_3}} \longrightarrow h^3 \longrightarrow \boxed{f^4_{w_4}} \longrightarrow y$$

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

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

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

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

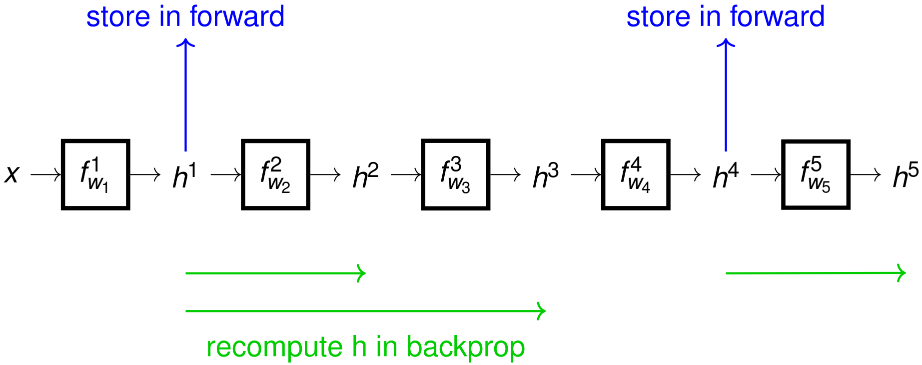
$$\frac{\partial f^1}{\partial x}(x, w_1)$$

First Trick

Reduce the batch size !!!

Michaël Sdika
CREATIS - CNRS

Second Trick : Checkpointing



Michaël Sdika
CREATIS - CNRS

Third Trick : Revertible Networks

do no store h_l in forward

$$x \longrightarrow \boxed{f^1_{w_1}} \longrightarrow h^1 \longrightarrow \boxed{f^2_{w_2}} \longrightarrow h^2 \longrightarrow \boxed{f^3_{w_3}} \longrightarrow h^3 \longrightarrow \boxed{f^4_{w_4}} \longrightarrow h^4 \longrightarrow \boxed{f^5_{w_5}} \longrightarrow h^5$$

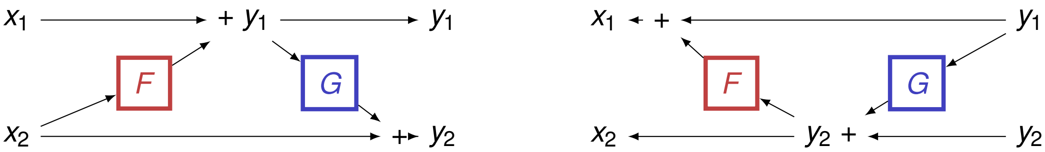
recompute h_l in backprop

Michaël Sdika
CREATIS - CNRS

Third options : Revertible 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

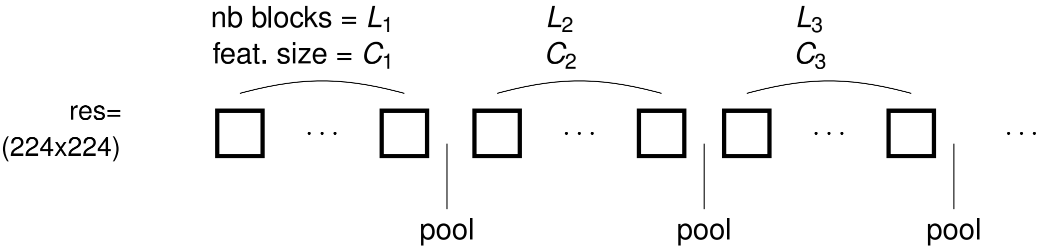
Conclusion

Take home message

Do not start your new network from scratch !

Thank you !!

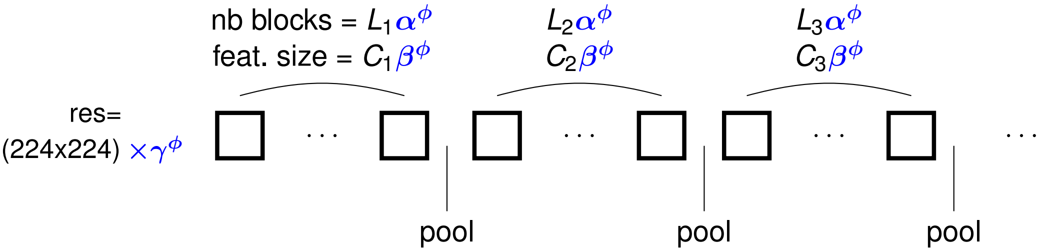
Efficient Net : compound scaling of networks



Tan and Le. PMLR 2019, Efficientnet : Rethinking model scaling for convolutional neural networks

Michaël Sdika
CREATIS - CNRS

Efficient Net : compound scaling of networks



- base network EffNet$_1$, (ϕ = 1)
- find α, β, γ :
  - ϕ = 1
  - optimize accuracy/flops s.t. $\alpha\beta^2\gamma^2 \approx 2$
- More Capacity : change ϕ : EffNet$_\phi$
- flops = flops$_1$ $\times$ $(\alpha\beta^2\gamma^2)^\phi$

Tan and Le. PMLR 2019, Efficientnet : Rethinking model scaling for convolutional neural networks