Visual scene real-time analysis for Intelligent Vehicles:

Deep-Learning for visual scene analysis

Pr. Fabien MOUTARDE
Center for Robotics,
MINES ParisTech
PSL Université Paris

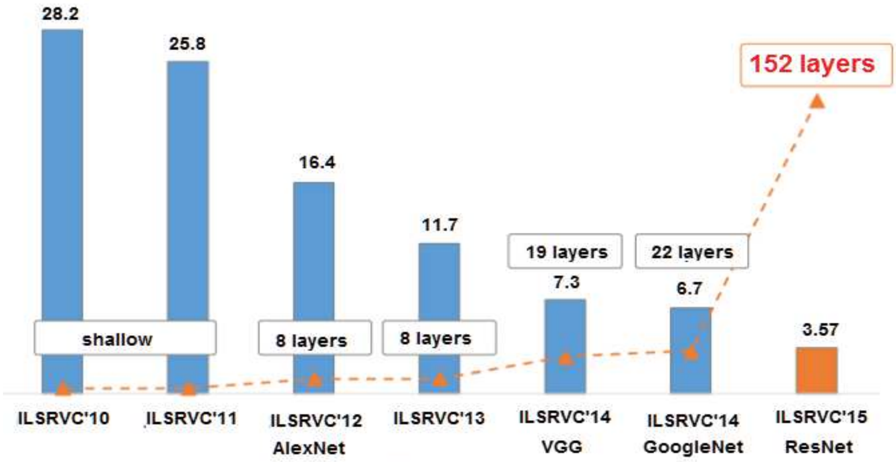
Fabien.Moutarde@mines-paristech.fr
http://people.mines-paristech.fr/fabien.moutarde

Outline

- Recalls on Convolutional Neural Networks (CNN or ConvNets) and Deep-Learning
- Transfer Learning
- Beyond Image Classification: DETECTION OF OBJECTS
- Instance segmentation with DeepLearning
- DL for Human pose inference and depth estimation
- Semantic segmentation with DeepLearning
- Interest and use of simulations / synthetic videos

DEEP-LEARNING APPROACHES for visual objects categorization

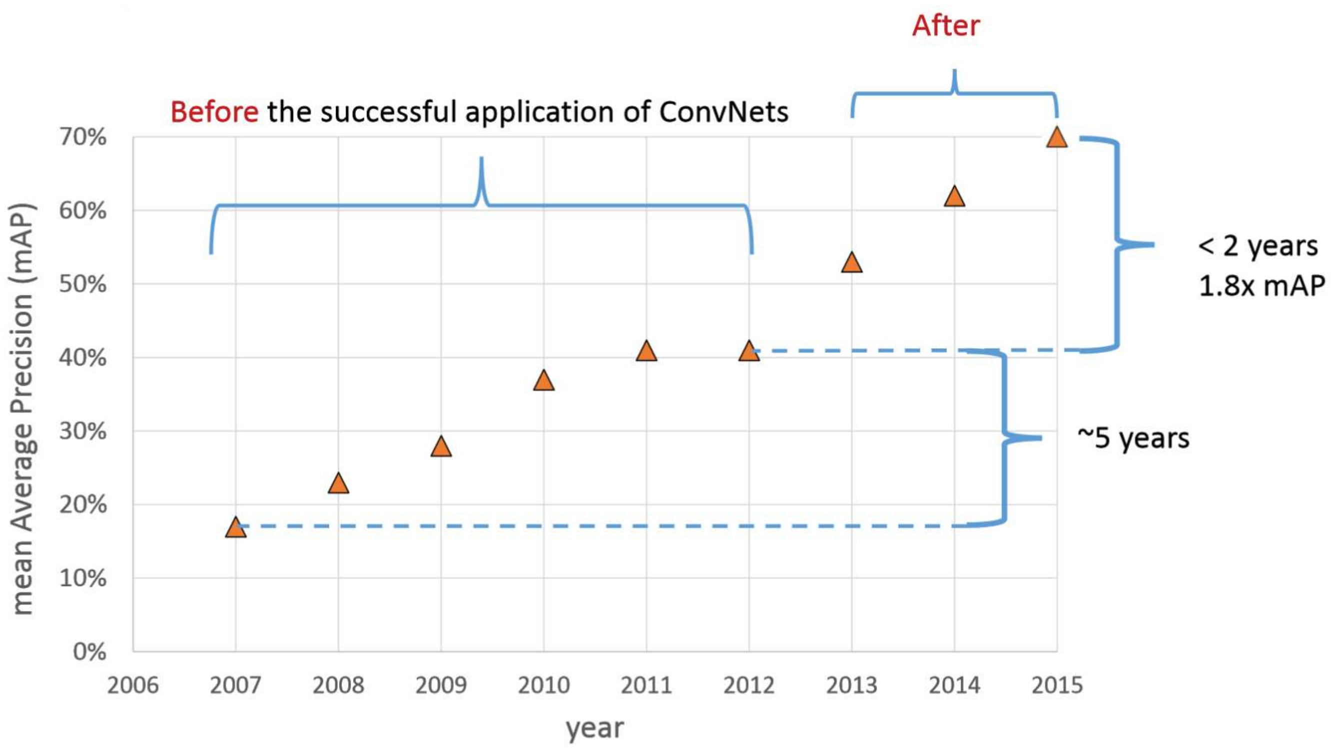
Since ~2012, Deep-Learning has brought very significant improvement over State-of-the-Art in Pattern Recognition and Image Semantic Analysis



ImageNet Classification top-5 error (%)

- won many vision pattern recognition competitions (OCR, TSR, object categorization, facial expression,…)
- deployed in photo-tagging by Facebook, Google,Baidu,…

Performance evolution of Pascal VOC object detection



ILSRVC challenges

ImageNet Large Scale Visual Recognition Challenge

- **Public dataset and benchmark**
- **Worldwide research competition on image classification and visual objects detection & recognition/categorization**

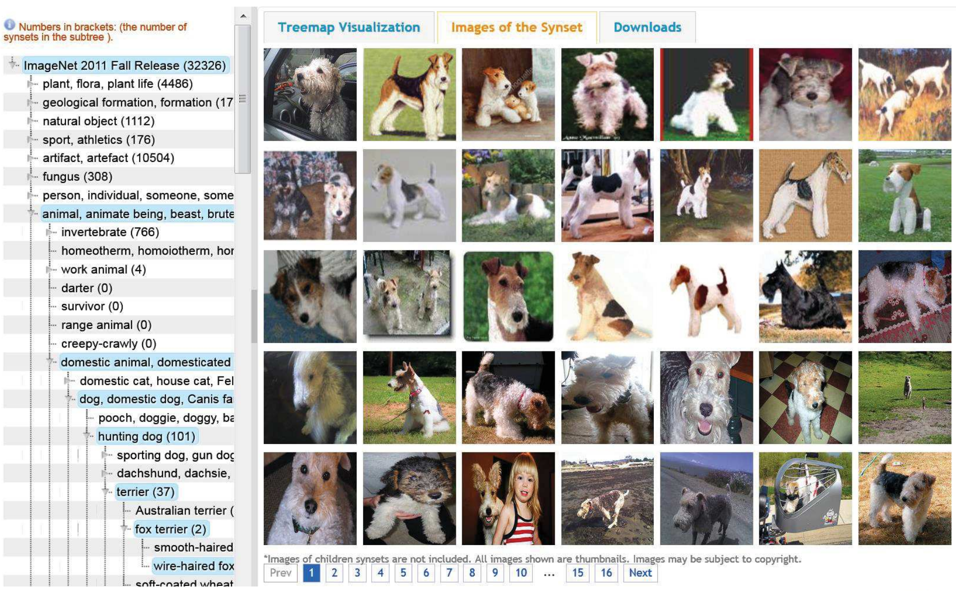
		PASCAL VOC 2012	ILSVRC 2013
Number of object classes		20	**200**
Training	Num images	5717	395909
	Num objects	13609	345854
Validation	Num images	5823	20121
	Num objects	13841	55502
Testing	Num images	10991	40152
	Num objects	---	---

Dramatic scale increase in image challenges in 2013
Challenge won by Deep-Learning methods every year since 2012

ImageNet dataset

Huge dataset of labelled images:

- **1000 classes of objects**
- **> 1 million labelled examples**



IMAGENET

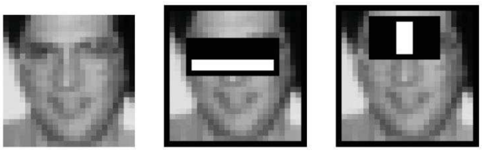
Importance of « features » in classical Machine-Learning

Input → Feature Extractor → Features → Traditional ML Algorithm → Output

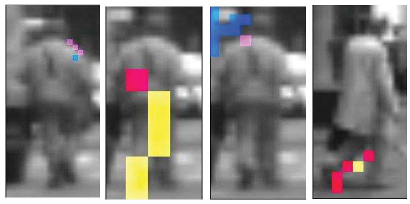
Traditional Machine Learning Flow

Examples of *hand-crafted* features

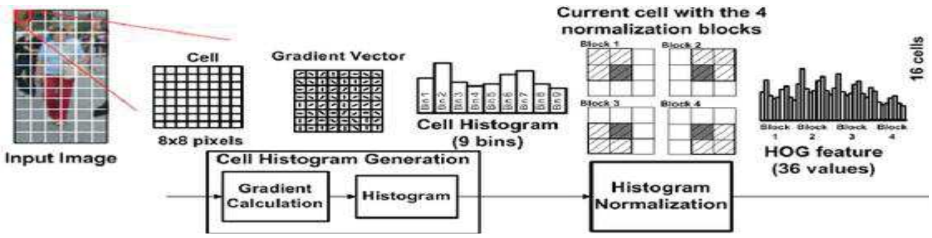
Haar features



Control-points features



HoG (Histogram of Gradients)



Deep-Learning (DL) general principle

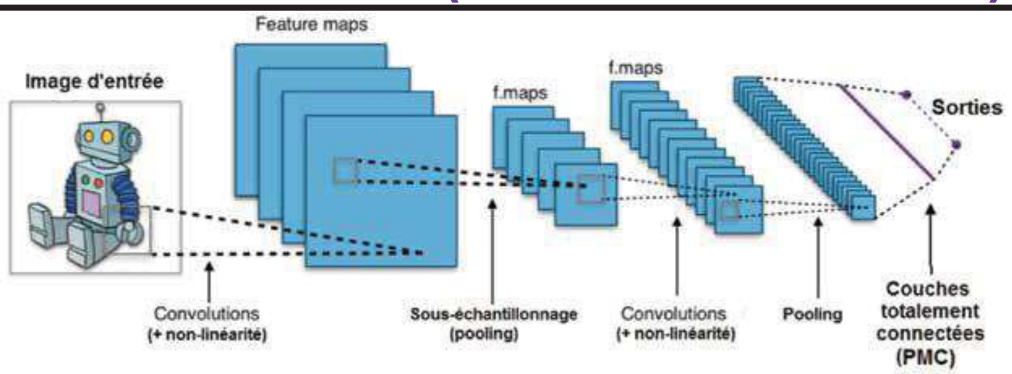
Input (raw)
↓
Handcrafted features
↓
Learnt transformation
↓
Output

Shallow ML using *handcrafted* features

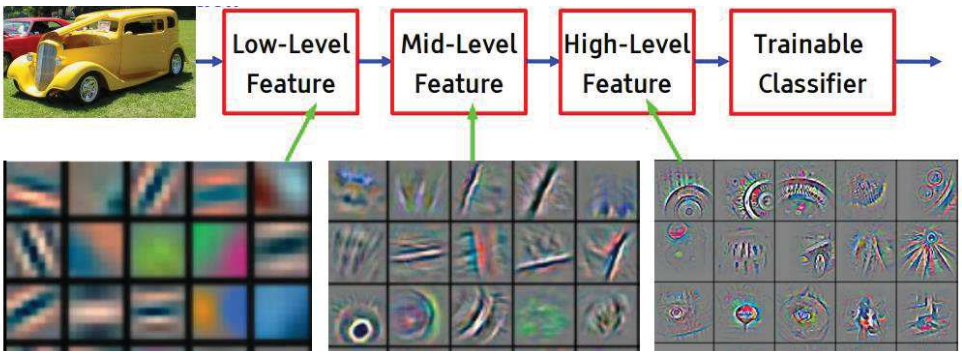
Input (raw)
↓
Learnt low-level features
↓
Learnt intermediate level features
↓
Learnt high-level (≈abstract) features
↓
Learnt transformation
↓
Output

DL: *jointly* learn classification *and features*

Convolutional Neural Networks (CNN, or ConvNet)

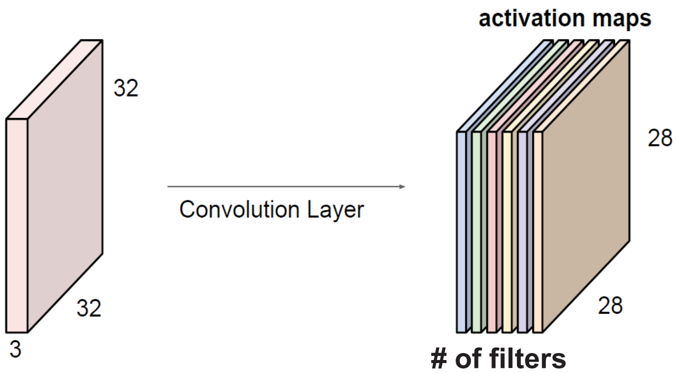


- For inputs with ***correlated dims*** (2D ***image***, 1D signal,…)
- Succession of Convolutions and « pooling » layers

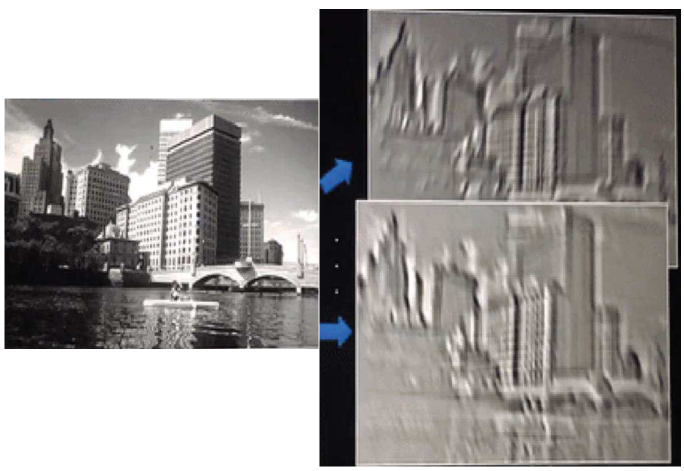


[Proposed in 1998 by Yann LeCun (French prof. @ NYU, now AI research director of Facebook)]

Convolution layers

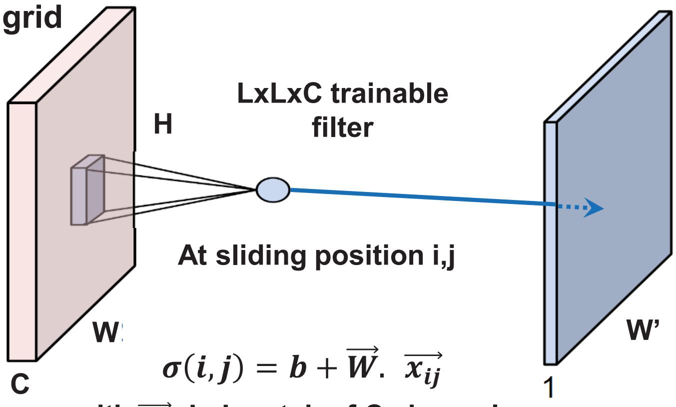


One "activation map" for each convolution filter



A Convolution layer applies several 3D filters to input image *(or to input set of activation maps from previous layer)*

Convolution: sliding a 3D filter over a multi-channel image



$\sigma(i,j) = b + \vec{W}.\ \vec{x_{ij}}$

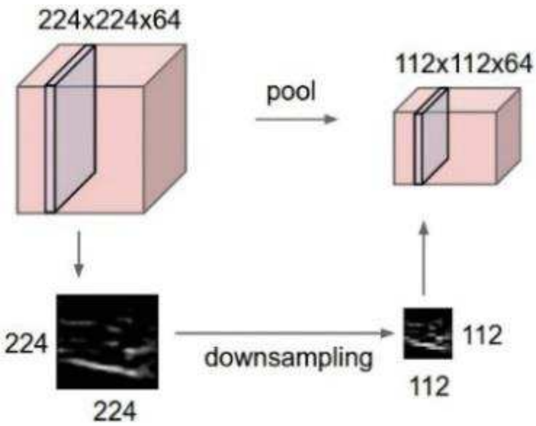
with $\vec{x_{ij}}$: LxL patch of C channels
→ vector of dimension= L^2xC,
➔ $\vec{W}$ of same dimension

Optional non-linear activation function
$o(i,j) = f(\sigma(i,j))$,
with f= tanh, ReLU, …

For each filter, a grid of W'xH' neurons with *shared* weigths $\vec{W}$
(each neuron applies same filter at a different sliding position in input)

See illustrative animation at: `http://cs231n.github.io/convolutional-networks/`

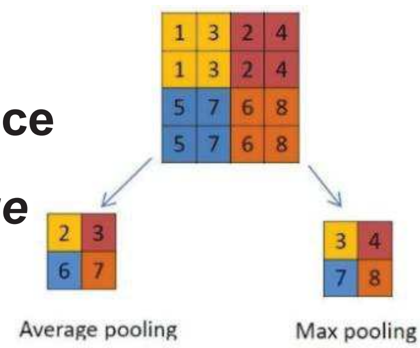
Pooling layers



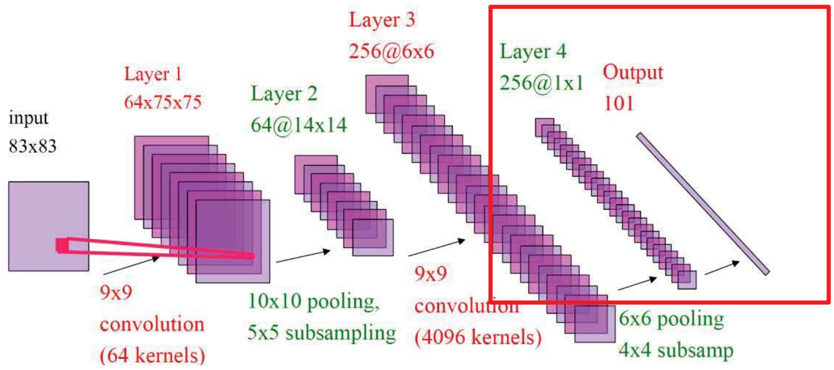
Pooling ≈ Downsampling

A pooling layer aggregates over space for:
- **Dimension reduction**
- **Noise reduction**
- **Small translation and scaling invariance**

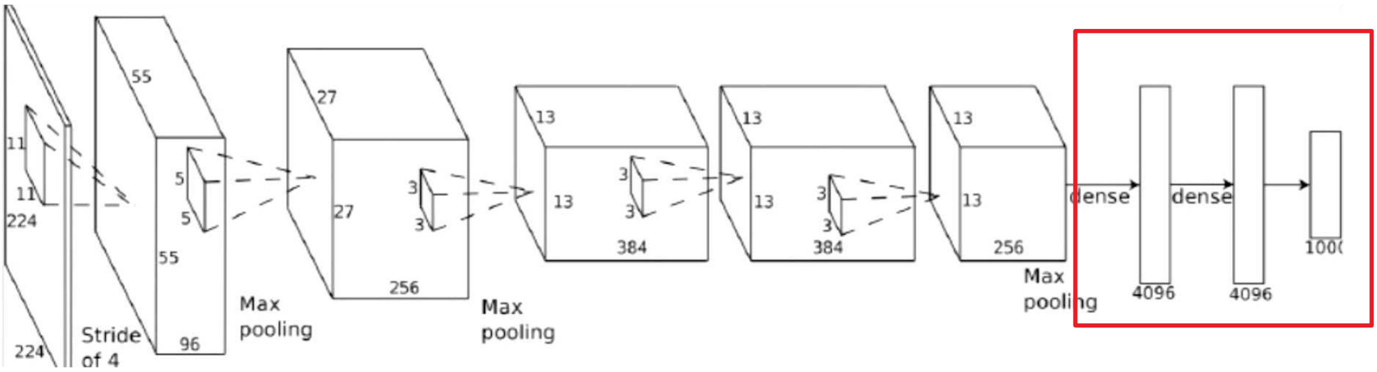
The pooling operation typically uses *average or max* on sets of 2x2 (or pxp) pixels



Final classification layer: often classical fully-connected MLP

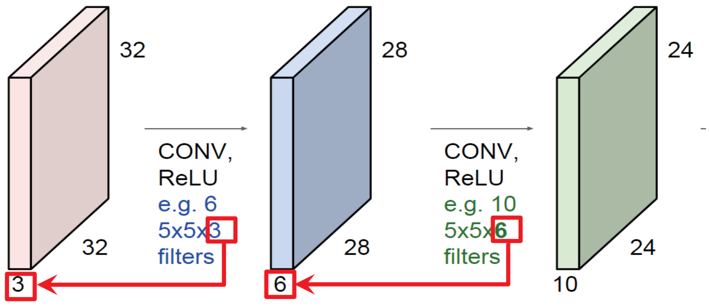


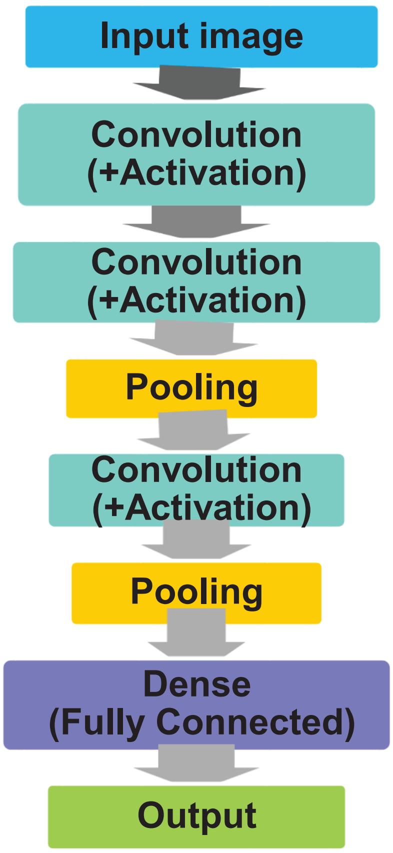
AlexNet



Global architecture of a convNet

Succession of Convolution (+ optional activation) layers and Pooling layers, which extract the hierarchy of features, followed by dense (fully connected) layer(s) for final classification



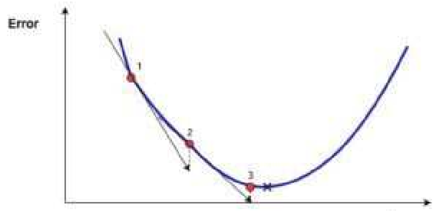


NB: each convolution layer <u>processes</u> <u>*FULL DEPTH* of previous set of activation maps</u>

ConvNet training

All successive layers of a convNet forms a Deep neural network (with weigh-sharing inside each conv. Layer, and specific pooling layers).

Training = optimizing values of weights&biases
Method used = gradient descent



➔ <u>Stochastic Gradient Descent</u> (SGD), using <u>*back-propagation*</u>:

- Input 1 (or a few) random training sample(s)
- Propagate
- Calculate error (loss)
- Back-propagate through all layers from end to input, to compute gradient
- Update convolution filter weights

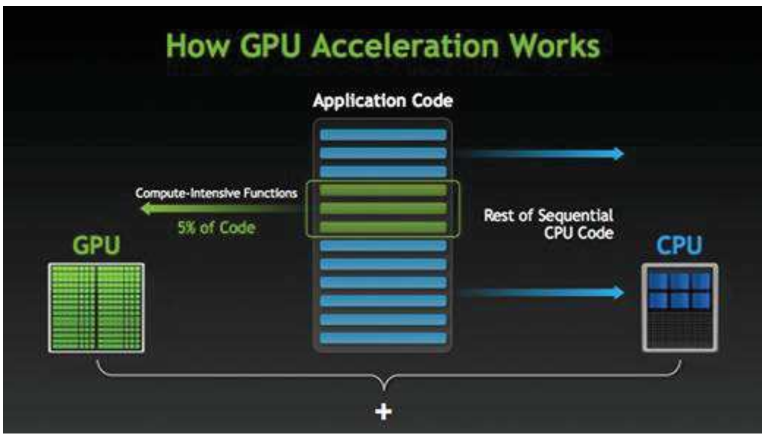
convNets and GPU

Good convNets are very big (millions of parameters!)

Training generally performed on BIG datasets

➔ Training can be extremely computer-intensive

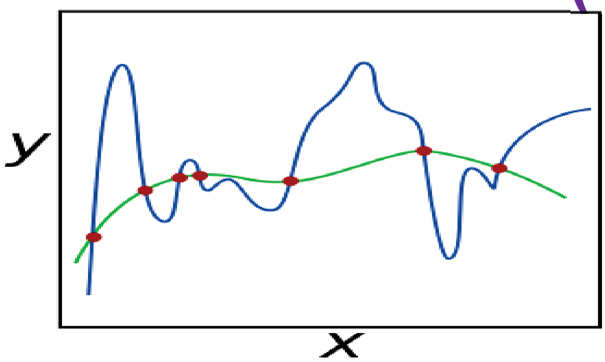
➔ More manageable using <u>GPU</u> (Graphical Processing Unit) acceleration for ultra-parallel processing



Some ConvNet training « tricks »

- Importance of input normalization
  (zero mean, unit variance)
- Importance of weights initialization
  random but SMALL and prop. to 1/sqrt(nbInputs)
- Decreasing (or adaptive) learning rate
- Importance of training set size
  ConvNets often have a LARGE number of free parameters
  ➔ train them with a sufficiently large training-set !
- Avoid overfitting by:
  - Use of L1 or L2 regularization (after some epochs)
  - Use « *Dropout* » regularization (esp. on large FC layers)

Avoid overfitting using L1/L2 *regularization* (« weight decay »)



Trying to fit too many free parameters with not enough information can lead to overfitting

Regularization = *penalizing too complex models*

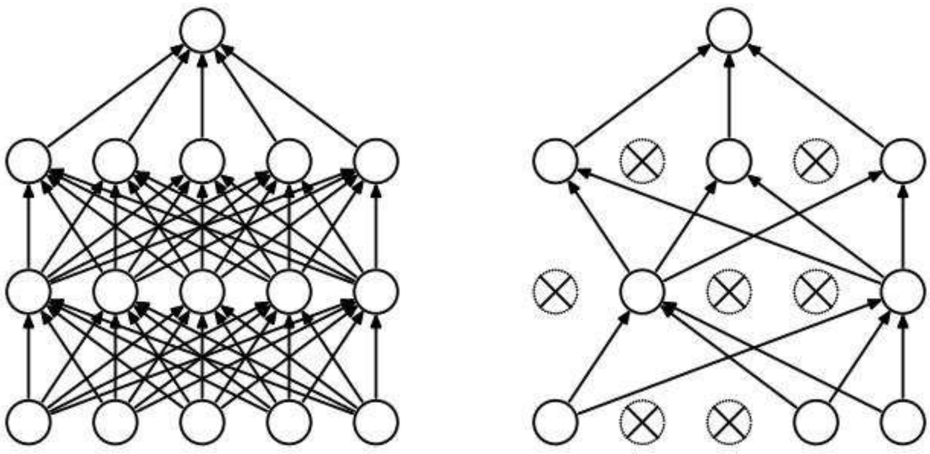
Often done by adding a special term to cost function

For neural network, the regularization term is just the L2- or L1- norm L2 of the vector of all weights:

$K = \Sigma_m(\text{loss}(Y_m,D_m)) + \beta\ \Sigma_{ij} |W_{ij}|^p$ with p=2 (L2) or p=1 (L1)

→ name *"Weight decay"*

DropOut Regularization for convNet training

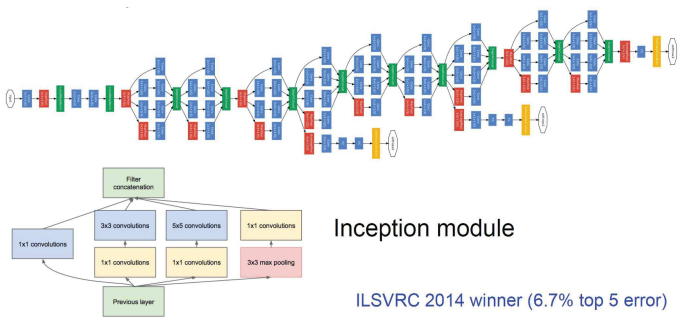


(a) Standard Neural Net (b) After applying dropout.

At each training stage, individual nodes can be temporarily "dropped out" of the net with probability p (usually ~0.5), or re-installed with last values of weights

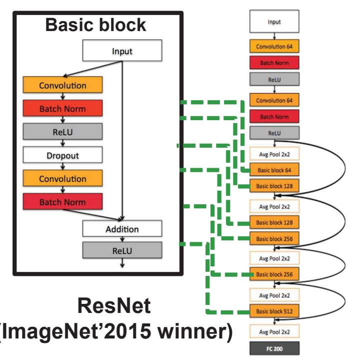
ImageNet dataset and state-of-the-art convNets

- The most performant ConvNets have millions of trainable weights, so they must be trained on very large datasets
- Training on ImageNet, was an essential factor of their recognition performances



Inception module

ILSVRC 2014 winner (6.7% top 5 error)



ResNet
(ImageNet'2015 winner)

- Every year, a new ImageNet challenge winner, with better acuracy using new ConvNet architecture & algo
- Those general-purpose pre-trained image classifiers (AlexNet, GoogleNet_Inception, ResNet, etc…) are publicly available

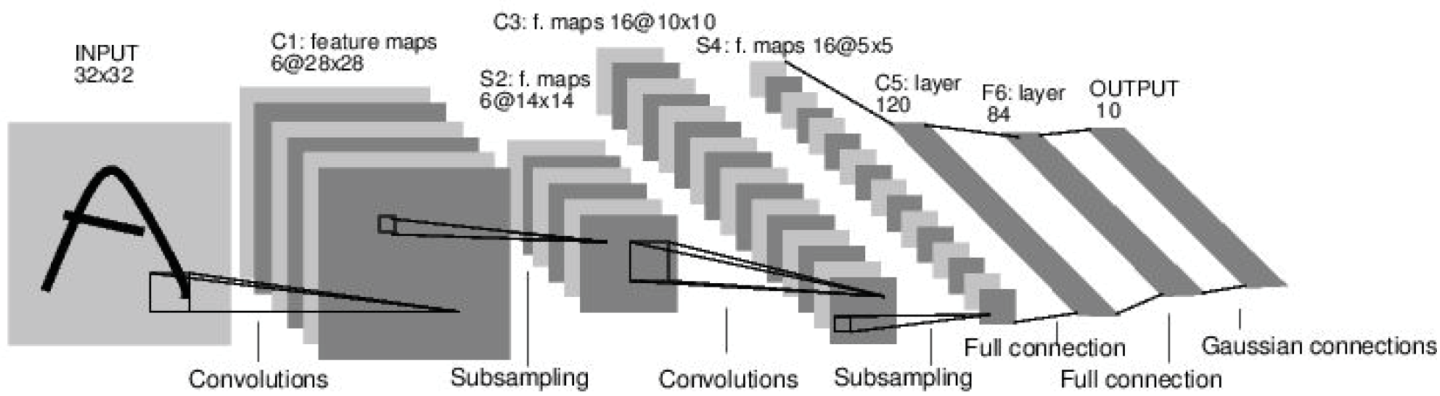
Examples of successful ConvNets

- **LeNet:** 1st successful applications of ConvNets, by Yann LeCun in 1990's. Used to read zip codes, digits, etc.
- **AlexNet:** Beginning of ConvNet "buzz": largely outperformed competitors in ImageNet_ILSVRC2012 challenge. Developped by Alex Krizhevsky et al., architecture similar to LeNet (but deeper+larger, and some chained ConvLayers before Pooling). 60 M parameters !
- **GoogLeNet:** ILSVRC 2014 winner, developed by Google. Introduced an *Inception Module*, + AveragePooling instead of FullyConnected layer at output. Dramatic reduction of number of parameters (4M, compared to AlexNet with 60M).
- **VGGNet:** Runner-up in ILSVRC 2014. Very deep (16 CONV/FC layers) → 140M parameters !!
- **ResNet:** ILSVRC 2015, "Residual Network" introducing "skip" connections. Currently ~ SoA in convNet. Very long training but fast execution.

LeNet, for digits/letters recognition

[LeCun et al., 1998]

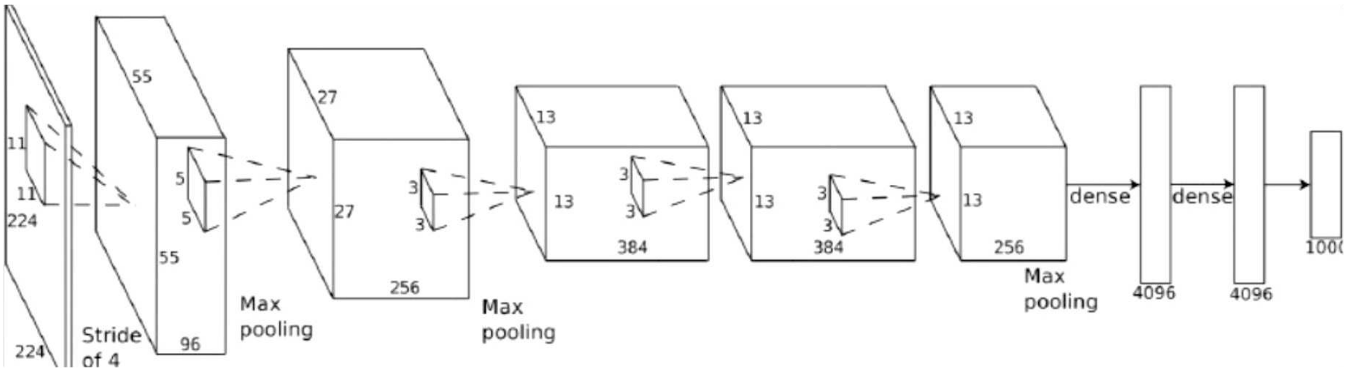
Input: 32x32 image



Conv filters were 5x5, applied at stride 1
Subsampling (Pooling) layers were 2x2 applied at stride 2
i.e. architecture is [CONV-POOL-CONV-POOL-CONV-FC]

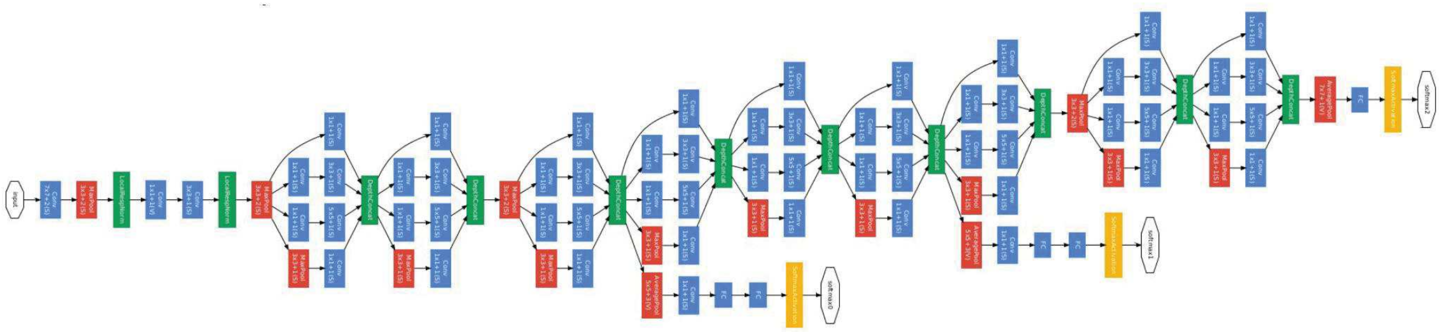
AlexNet, for image categorisation
[Krizhevsky et al. 2012]

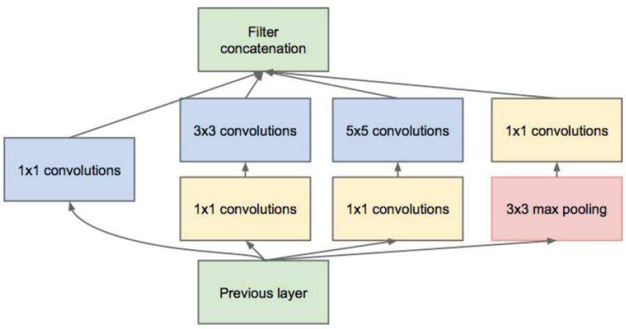
Input: 224x224x3 image



60 million parameters !...

GoogleNet
[Szegedy et al., 2014]



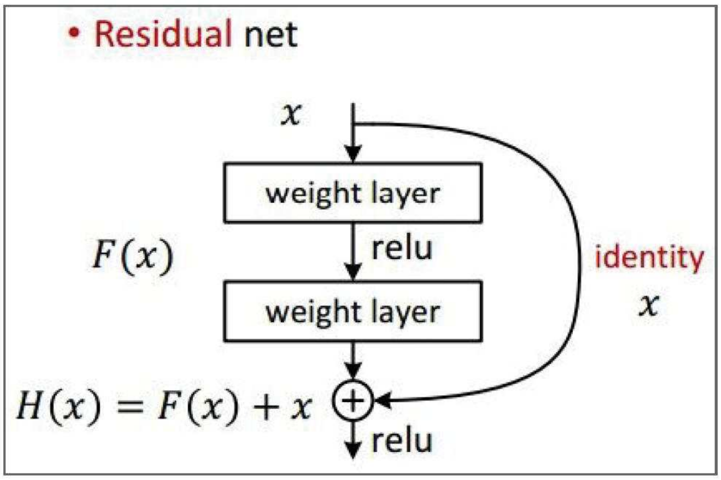


Inception module

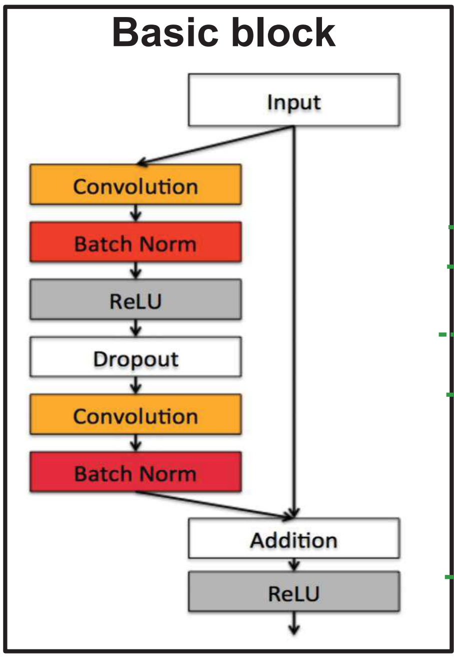
ILSVRC 2014 winner (6.7% top 5 error)

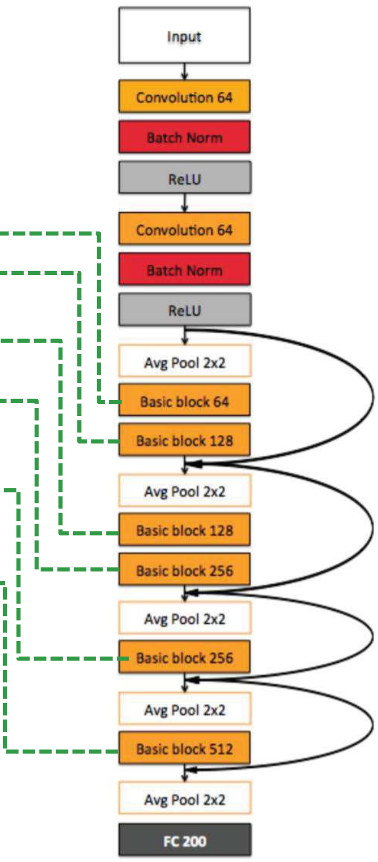
ResNet (Residual Net), by Microsoft *[He et al., 2015]*

- **ILSVRC 2015 *large* winner in 5 main tracks (3.6% top 5 error)**
- **152 layers!!!**
- **But novelty = *"skip" connections***



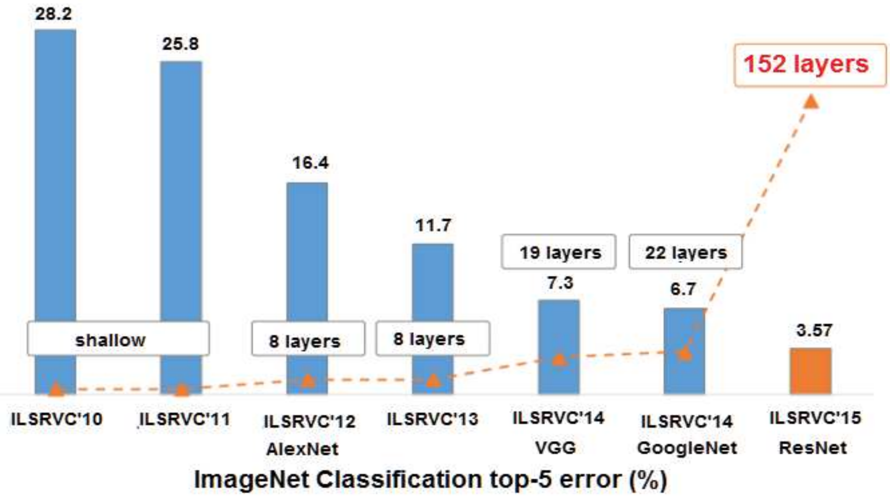
ResNet global architecture





- **2-3 weeks of training on 8 GPU machine !!**
- **However, at runtime faster than a VGGNet! (even though it has 8x more layers)**

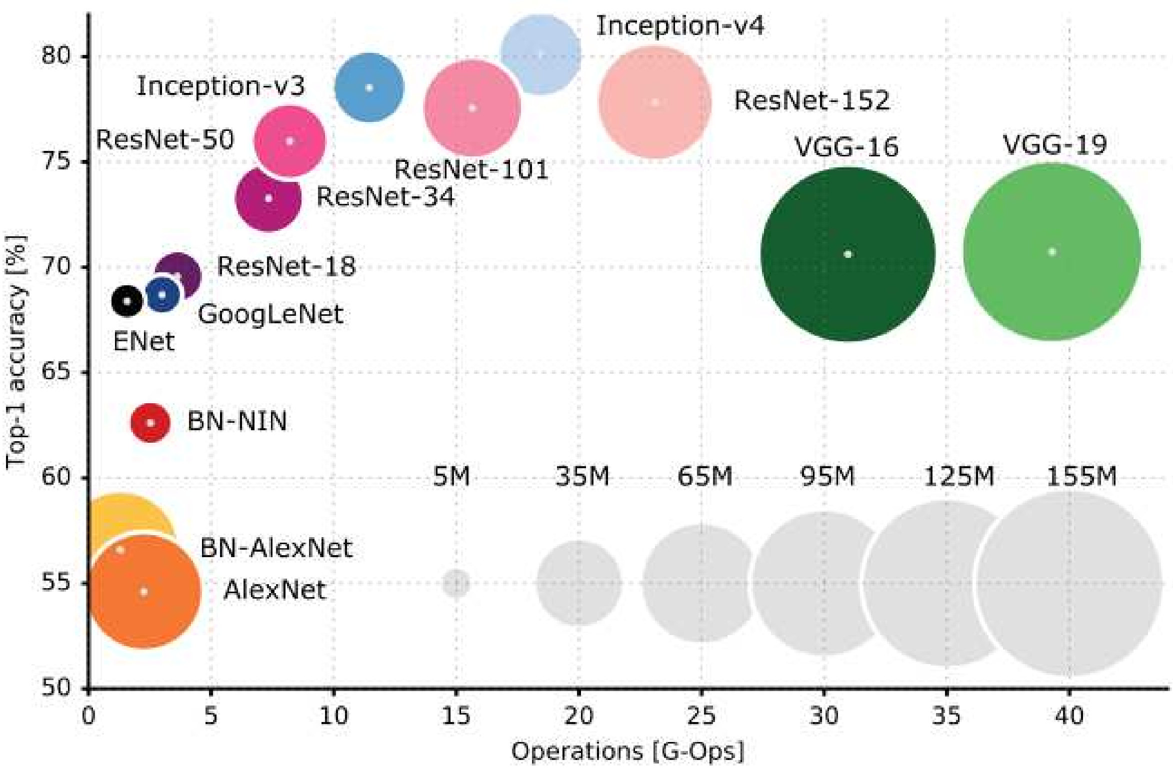
Summary of recent ConvNet history



But most important is the choice of
ARCHITECTURAL STRUCTURE

+ *More and more modified architectures for tasks other than just image classification*

Performance comparisons of common pre-trained convNets



Programming environments for Deep-Learning

- **TensorFlow** https://www.tensorflow.org
- **KERAS** https://keras.io

  Python front-end APIs mapped either on Tensor-Flow or Theano back-end
- **PyTorch** https://pytorch.org/
- ***Caffe*** http://caffe.berkeleyvision.org/

  C++ library, hooks from Python → notebooks
- ***Theano*** http://www.deeplearning.net/software/theano/
- ***Lasagne*** http://lasagne.readthedocs.io

  lightweight library to build+train neural nets in Theano

All of them handle transparent use of GPU, and most of them are used in Python code/notebook

Example of convNet code in Keras

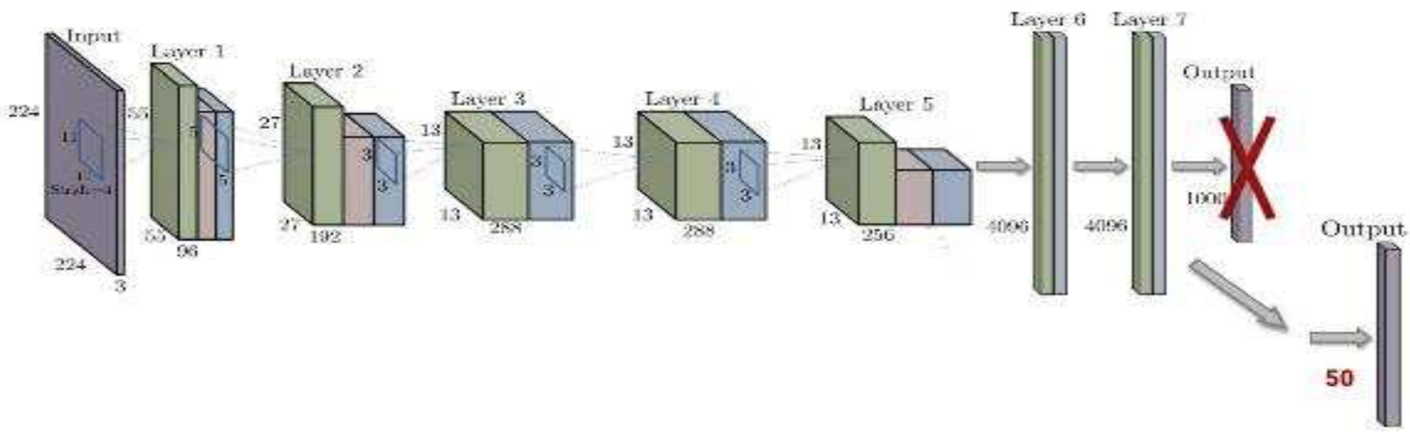
```
model = Sequential()
# 1 set of (Convolution+Pooling) layers, with Dropout
model.add(Convolution2D(conv_depth_1, kernel_size, kernel_size,
             border_mode='valid', input_shape=(depth, height, width)))
model.add( MaxPooling2D(pool_size=(pooling_size, pooling_size)) )
model.add(Activation('relu'))
model.add(Dropout(drop_prob))
# Now flatten to 1D, and apply 1 Fully_Connected layer
model.add(Flatten())
model.add(Dense(hidden_size1, init='lecun_uniform'))
model.add(Activation('sigmoid'))
# Finally add a Softmax output layer, with 1 neuron per class
model.add(Dense(num_classes, init='lecun_uniform'))
model.add(Activation('softmax'))
# Training "session
sgd = SGD(lr=learning_rate, momentum=0.8) # Optimizer
model.compile(loss='categorical_crossentropy', optimizer=sgd)
model.fit(X_train, Y_train, batch_size=32, nb_epoch=2, verbose=1,
                                   validation_split=valid_proportion)

# Evaluate the trained model on the test set
model.evaluate(X_test, Y_test, verbose=1)
```

Outline

- Recalls on Convolutional Neural Networks (CNN or ConvNets) and Deep-Learning
- Transfer Learning
- Beyond Image Classification: DETECTION OF OBJECTS
- Instance segmentation with DeepLearning
- DL for Human pose inference and depth estimation
- Semantic segmentation with DeepLearning
- Interest and use of simulations / synthetic videos

Generality of learnt representation + Transfer learning



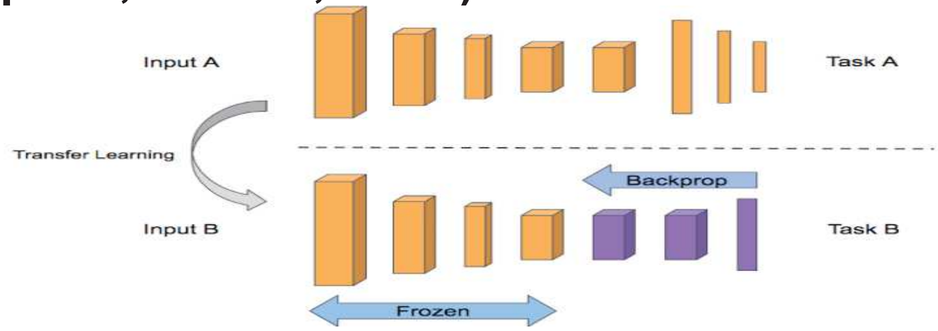
By <u>removing *last layer(s)*</u> (those for classification) of a convNet trained on ImageNet, one obtains a <u>*transformation of any input image into a semi-abstract representation,*</u> which can be used for learning SOMETHING ELSE (« <u>transfer learning</u> »):

- either by just <u>using learnt representation as features</u>
- or by creating new convNet output and perform <u>learning of new output layers + fine-tuning of re-used layers</u>

Transfer learning

- SoA convNets winning ImageNet are image CLASSIFIERS for one object per image
- Many object categories can be irrelevant (e.g. boat when onboard a car)

➔ For each particular application, models are usually obtained from state-of-the-art ConvNets pre-trained on ImageNet (winners of yearly challenge, eg: AlexNet, VGG, Inception, ResNet, etc…)

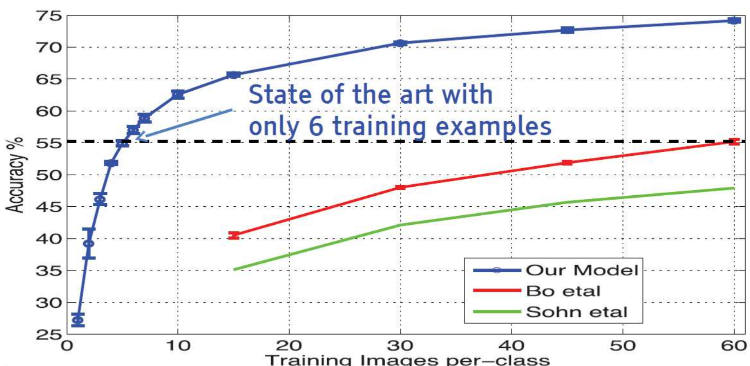


➔ Adaptation is performed by ***Transfer Learning***, ie modification+training of last layers and/or fine-tuning of pre-trained weights of lower layers

Transfer Learning and fine-tuning

- Using a CNN pre-trained on a large dataset, possible to adapt it to another task, ***using only a SMALL training set***!

- Network first trained on ImageNet.
- Last layer chopped off
- Last layer trained on Caltech 256,
- first layers N-1 kept fixed.
- State of the art accuracy with only 6 training samples/class



# Train	Acc % 15/class	Acc % 30/class	Acc % 45/class	Acc % 60/class
Sohn *et al.* [16]	35.1	42.1	45.7	47.9
Bo *et al.* [3]	40.5 ± 0.4	48.0 ± 0.2	51.9 ± 0.2	55.2 ± 0.3
Non-pretr.	9.0 ± 1.4	22.5 ± 0.7	31.2 ± 0.5	38.8 ± 1.4
ImageNet-pretr.	**65.7 ± 0.2**	**70.6 ± 0.2**	**72.7 ± 0.4**	**74.2 ± 0.3**

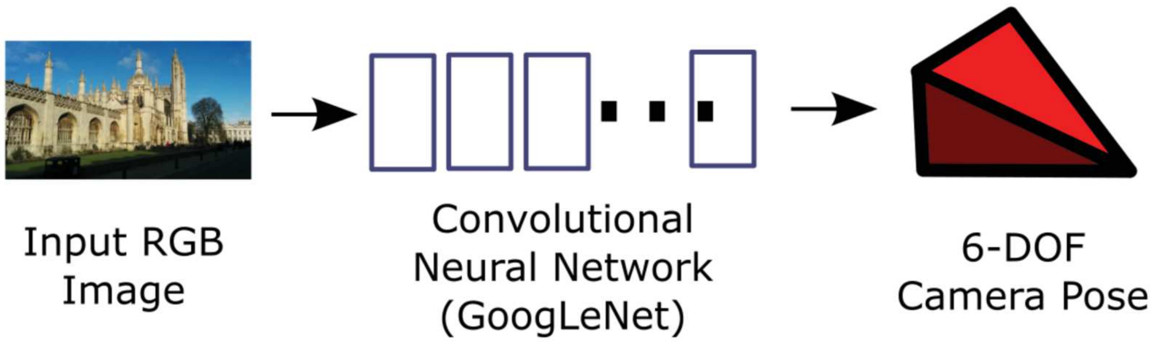
3: [Bo, Ren, Fox. CVPR, 2013] 16: [Sohn, Jung, Lee, Hero ICCV 2011]

Examples of transfer-learning applications

- Learning on simulated synthetic images
  + fine-tuning on real-world images
- Recognition/classification for OTHER categories or classes
- Training an objects detector (or a semantic segmenter)

- Precise localization (position+bearing) = PoseNet
- End-to-end driving (imitation Learning)
- 3D informations (depth map) from monovision!

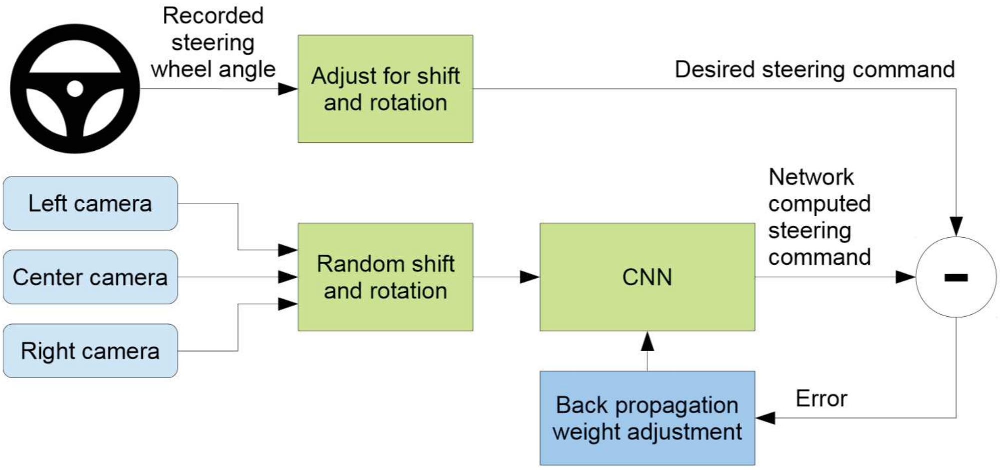
Transfer-Learning for 6-DOF Camera Relocalization



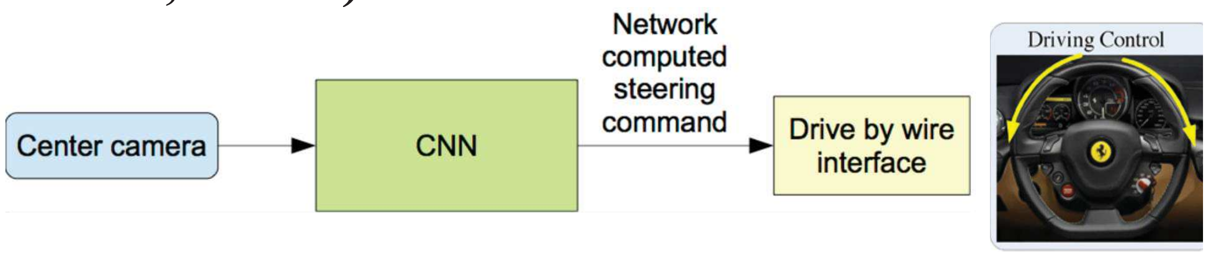
[A. Kendall, M. Grimes & R. Cipolla, "PoseNet: A Convolutional Network for Real-Time 6-DOF Camera Relocalization« , ICCV'2015, pp. 2938-2946]

King's College

End-to-end driving!



- **<u>End-to-end driving</u> by « <u>*imitation Learning*</u> » (nVidia, Valeo)**



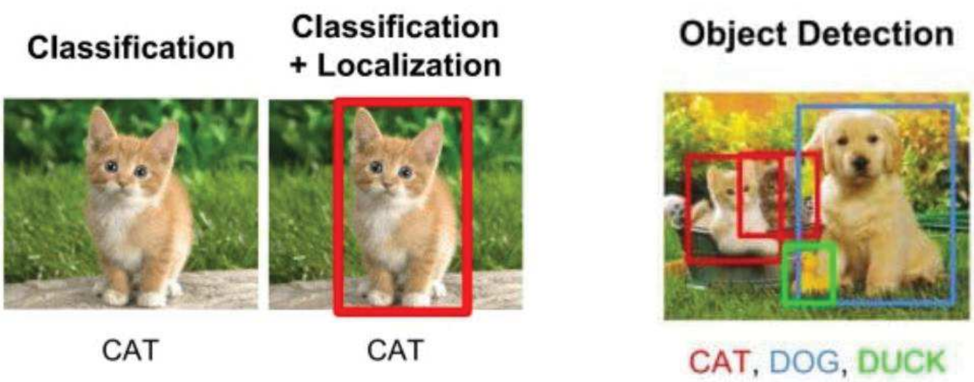
Transfer Learning code example in Keras

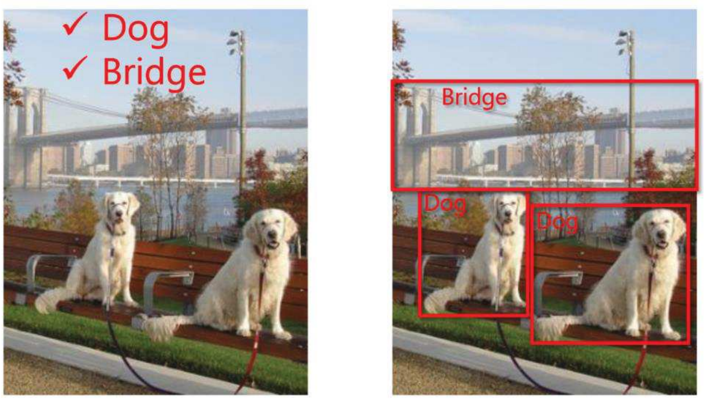
```
from keras.applications.inception_v3 import InceptionV3
from keras.preprocessing import image
from keras.models import Model
from keras.layers import Dense, GlobalAveragePooling2D
from keras import backend as K
# create the base pre-trained model base_model = InceptionV3(weights='imagenet',
                                                            include_top=False)
# add a global spatial average pooling layer
x = base_model.output x = GlobalAveragePooling2D()(x)
# let's add a fully-connected layer
x = Dense(1024, activation='relu')(x)
# and a logistic layer -- let's say we have 200 classes
predictions = Dense(200, activation='softmax')(x)
# this is the model we will train
model = Model(input=base_model.input, output=predictions)
# first: train only the top layers (which were randomly initialized)
# i.e. freeze all convolutional InceptionV3 layers
for layer in base_model.layers:
  layer.trainable = False
# compile the model (should be done *after* setting layers to non-trainable)
model.compile(optimizer='rmsprop', loss='categorical_crossentropy')
# train the model on the new data for a few epochs
model.fit_generator(...)
```

Outline

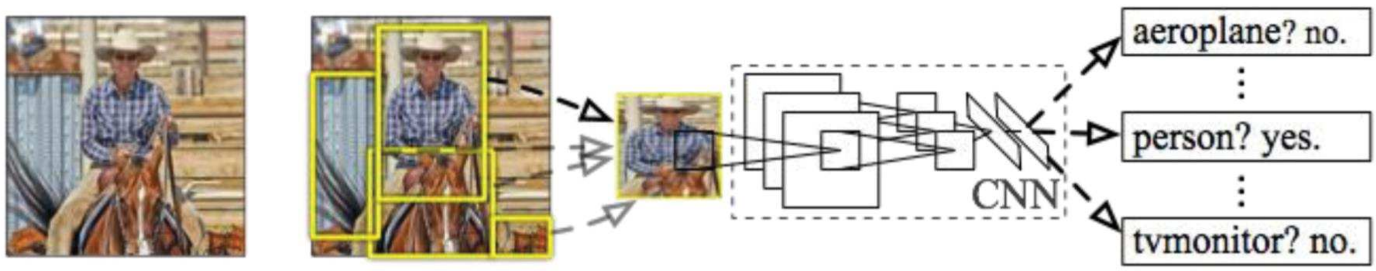
- Recalls on Convolutional Neural Networks (CNN or ConvNets) and Deep-Learning
- Transfer Learning
- <u>Beyond Image Classification: DETECTION OF OBJECTS</u>
- Instance segmentation with DeepLearning
- DL for Human pose inference and depth estimation
- Semantic segmentation with DeepLearning
- Interest and use of simulations / synthetic videos

Classification vs Detection





First simple idea: R-CNN



Input image

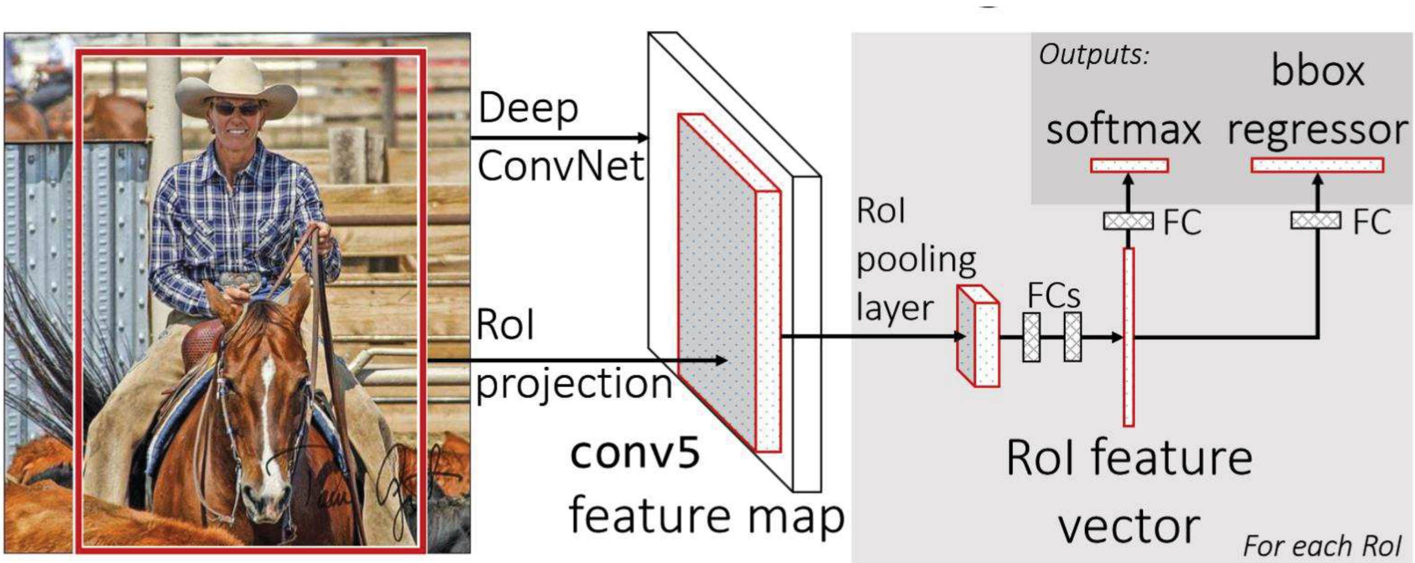
Extract region proposals (~2k / image)
e.g., selective search
[van de Sande, Uijlings et al.]

Compute CNN features on regions

Classify and refine regions

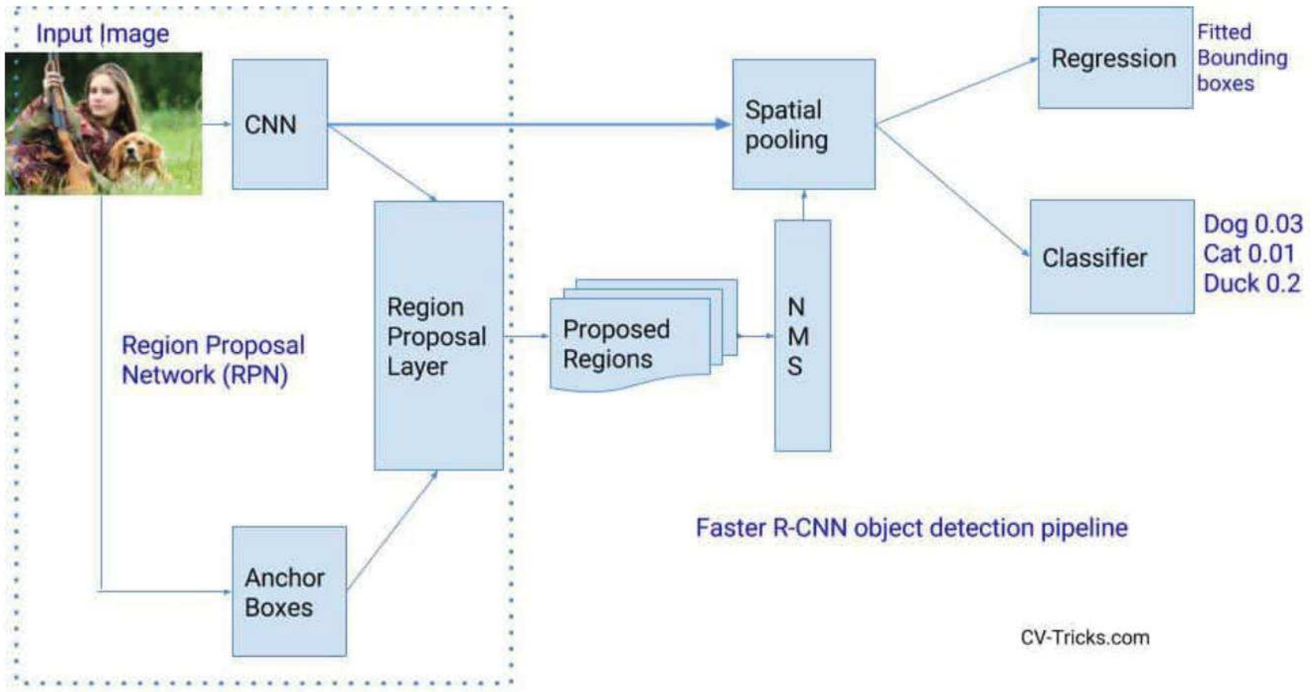
Very slow + rather approximate bounding-boxes

Better: Fast R-CNN



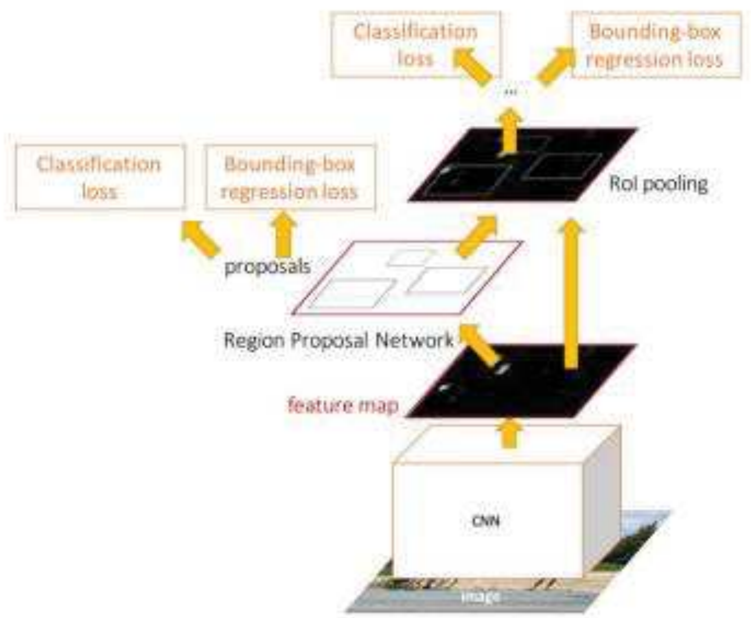
Learn a bounding-box regressor together with the class estimation (combination of 2 losses)

Faster_R-CNN



Learn also a « *Region Proposal Network* »
→ objects' bounding-boxes.

Training of Faster_RCNN



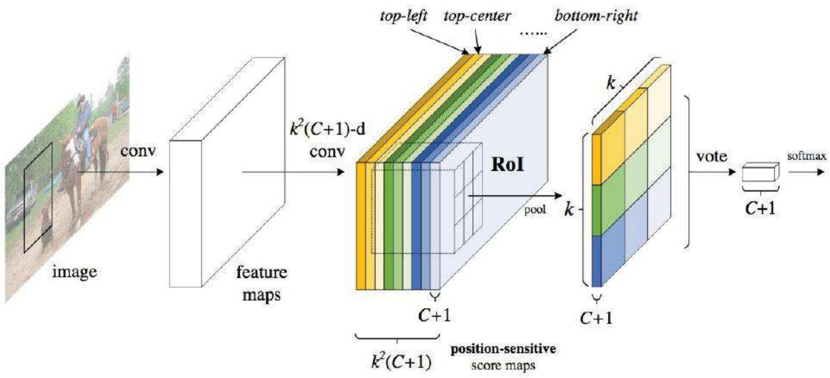
Combining 4 losses!

R-CNN family

- **History**
  - **R-CNN**: Selective search → Cropped Image → CNN
  - **Fast R-CNN**: Selective search → Crop feature map of CNN
  - ***Faster R-CNN***: CNN → Region-Proposal Network → Crop feature map of CNN
- **Best performances, but longest run-time**
- **End-to-end, multi-task loss**

[https://github.com/endernewton/tf-faster-rcnn]

R-FCN



- **Addresses translation-variance in detection**
  - **Position-sensitive ROI-pooling**
- **Good balance between speed & performance**
  - **2.5 - 20x faster than Faster R-CNN**

https://github.com/daijifeng001/R-FCN

FRCNN vs RFCN

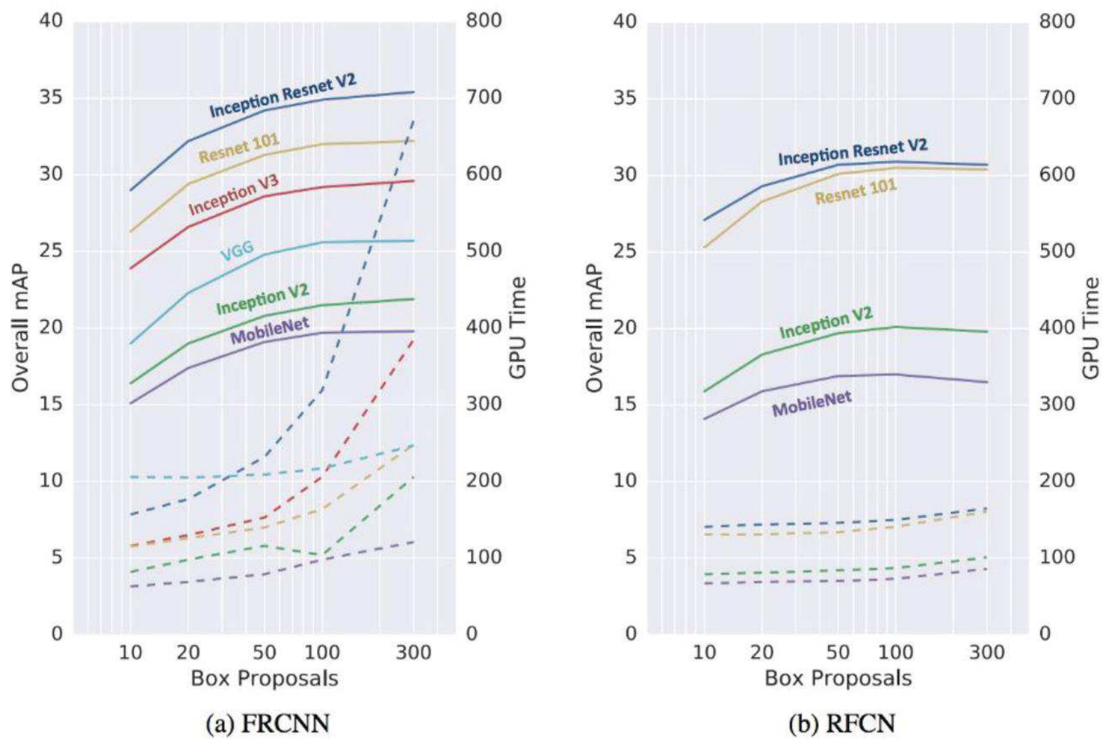


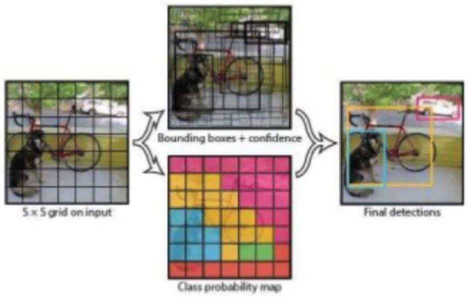
Image from: https://arxiv.org/pdf/1611.10012.pdf

Example video of objects visual simultaneous detection and categorization with R-CNN

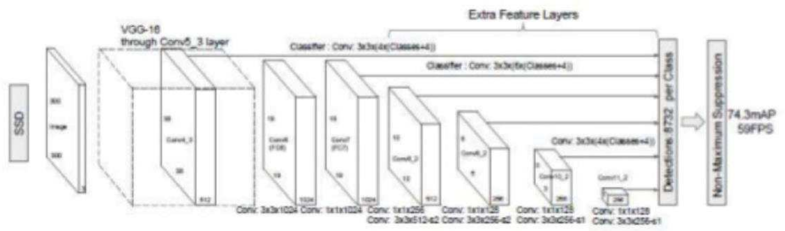
Detection *without proposal*

Solve detection as a regression problem (“single-shot” detection)

YOU ONLY LOOK ONCE(YOLO)



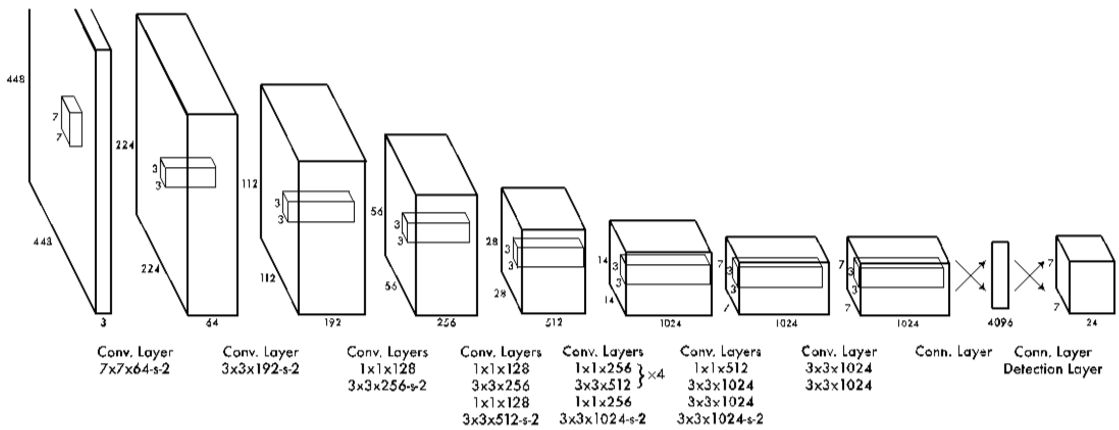
SINGLE SHOT MULTIBOX DETECTOR(SSD)



Images from: https://www.slideshare.net/TaegyunJeon1/pr12-you-only-look-once-yolo-unified-realtime-object-detection

YOLO

You Only Look Once



- **Modified GoogleNet/Inception**
- **Super fast (21~155 fps)**
- **Finds objects in image grids *in parallel***
- **Only slightly worse performance than Faster R-CNN**

YOLO principle

Unified Detection

- All BBox, All classes

1) Image → **S x S** grids

2) grid cell
→ **B**: BBoxes and Confidence score

$\boxed{x,\ y,\ w,\ h} \ \boxed{confidence} \rightarrow Pr(Object) * IOU_{pred}^{truth}$

→ **C**: class probabilities w.r.t #classes

$Pr(Class_i \mid Object)$

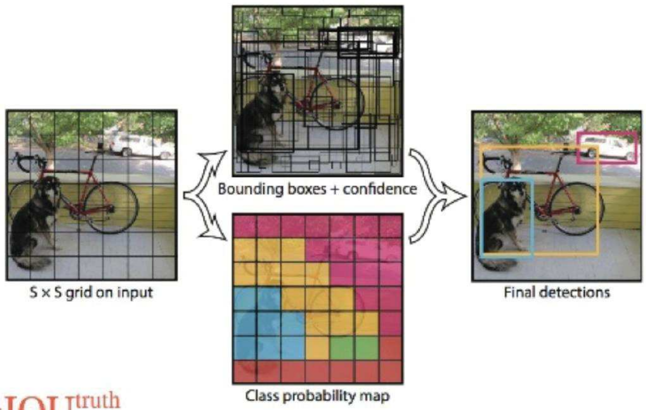


Figure 2: The Model. Our system models detection as a regression problem. It divides the image into an $S \times S$ grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor.

Slide from: https://www.slideshare.net/TaegyunJeon1/pr12-you-only-look-once-yolo-unified-realtime-object-detection

YOLO limitations and v2 improvements

- **Groups of small objects**
- **Unusual aspect ratios**
- **Localization error of bounding boxes**

➔ YOLOv2: Many improvements
+ Custom architecture – Darknet
(instead InceptionNet for YOLO)

YOLO versions

YOLO (darknet) - https://pjreddie.com/darknet/yolov1/ (C++)

YOLO v2 (darknet) - https://pjreddie.com/darknet/yolov2/ (C++)

- Better and faster - 91 fps for 288 x 288

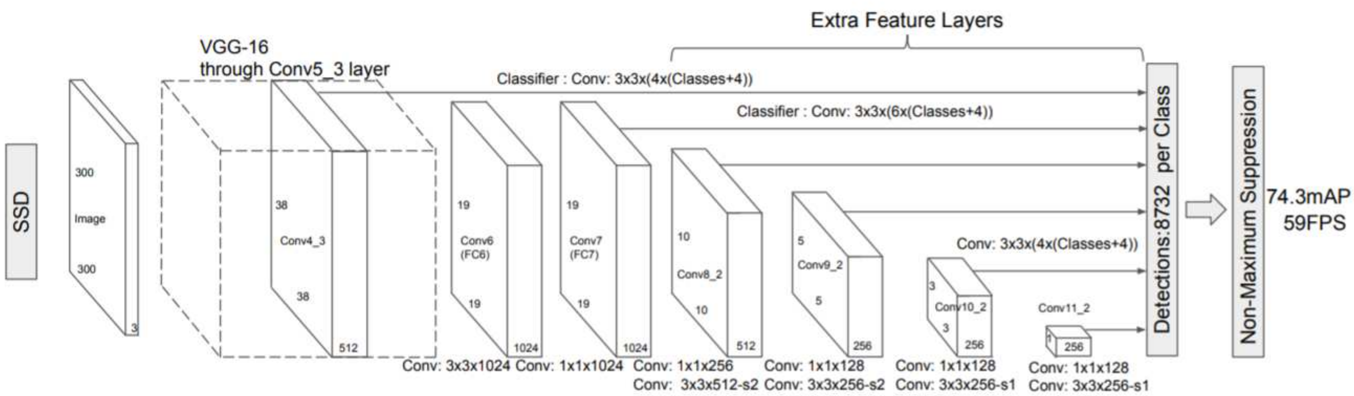
YOLO v3 (darknet) - https://pjreddie.com/darknet/yolo/ (C++)

YOLO (caffe) - https://github.com/xingwangsfu/caffe-yolo

YOLO (tensorflow) - https://github.com/thtrieu/darkflow

SSD (Single Shot Detector)

Architecture



Slower but more accurate than YOLO
Faster but less accurate than Faster_R-CNN

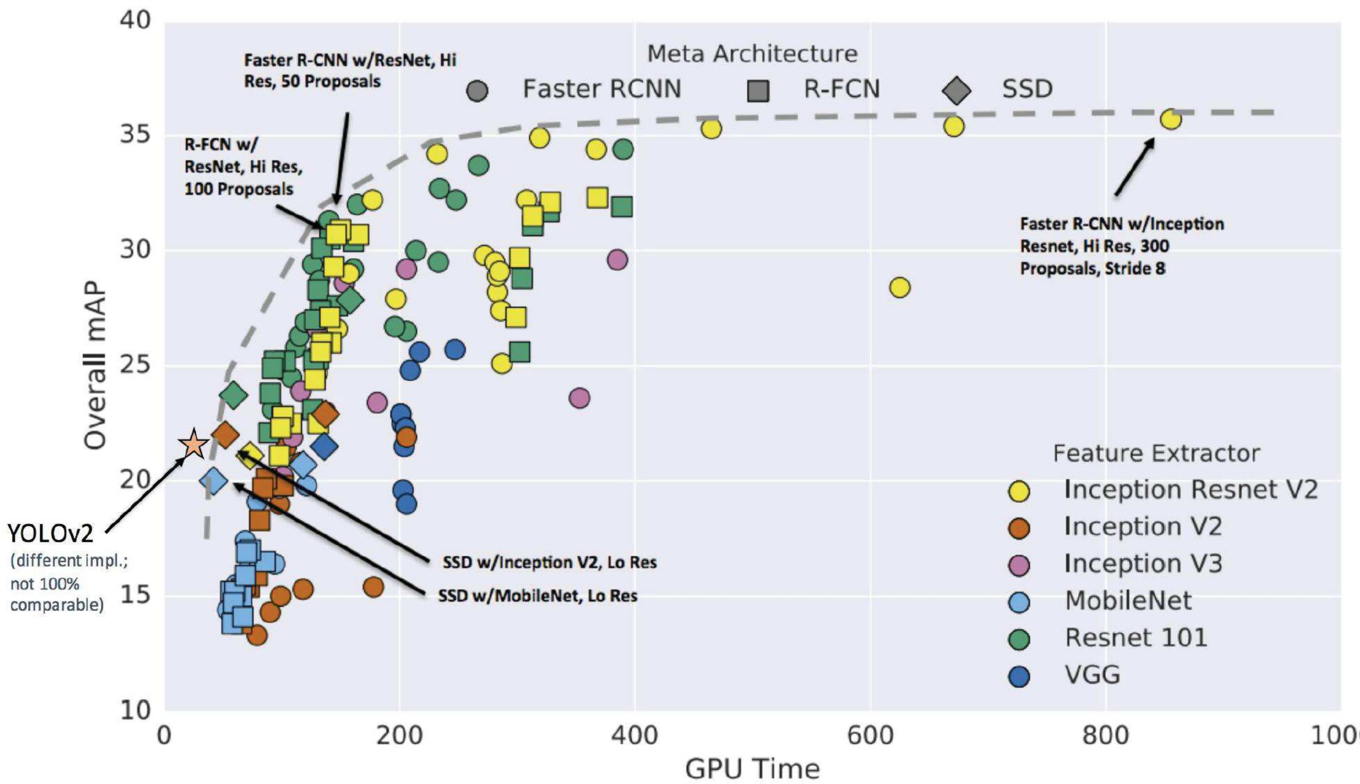
SSD versions

SSD (caffe) - https://github.com/weiliu89/caffe/tree/ssd

SSD (tensorflow) - https://github.com/balancap/SSD-Tensorflow

SSD (pytorch) - https://github.com/amdegroot/ssd.pytorch

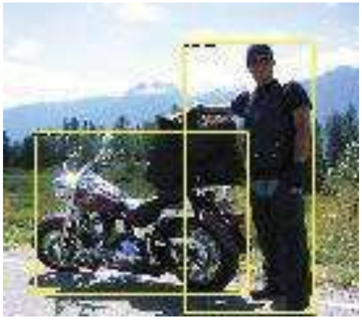
Recent comparison of convNets for object detection



Slide from Ross Girshick's CVPR 2017 Tutorial, Original Figure from Huang et al

Training sets for Visual objects detection

- **Training a visual objects detector requires a training set containg images WITH BOUNDING-BOXES (or even mask) ANNOTATION**
- **Two main « reference » training sets of this type:**
  - **Pascal VOC (Visual Object Class)**
    `http://host.robots.ox.ac.uk/pascal/VOC/`
  - **Coco (Common Objects in Context)**
    [more classes + MASK annotations]
    `http://cocodataset.org/`




VOC and COCO categories

VOC categories

aeroplane
bicycle
bird
boat
bottle
bus
car
cat
chair
cow
diningtable
dog
horse
motorbike
person
pottedplant
sheep
sofa
train
tvmonitor

COCO categories

person
bicycle
car
motorbike
aeroplane
bus
train
truck
boat
traffic light
fire hydrant
stop sign
parking
meter
bench
bird
cat
Dog
horse
sheep
Cow
elephant
bear
zebra
giraffe

backpack
umbrella
handbag
tie
suitcase
frisbee
skis
snowboard
sports ball
Kite
baseball bat
baseball glove
skateboard
surfboard
tennis racket
Bottle
wine glass
cup
fork
knife
spoon
bowl
banana

Apple
Sandwich
Orange
broccoli
carrot
hot dog
pizza
donut
cake
chair
Sofa
pottedplant
bed
diningtable
toilet
Tvmonitor
laptop
mouse
remote
keyboard
cell phone

microwave
oven
toaster
sink
refrigerator
book
clock
vase
scissors
teddy bear
hair drier
toothbrush

Summary on Visual object DETECTION with Deep-Learning

- If very fast inference is essential, better choose latest version of YOLO (or even MobileNet)
- If quality of detections (precision and recall) is more important, better choose Faster_RCNN
- For a compromise, SSD can be considered
- Pre-trained ConvNet detectors are available for many pre-defined categories (those of VOC or COCO)

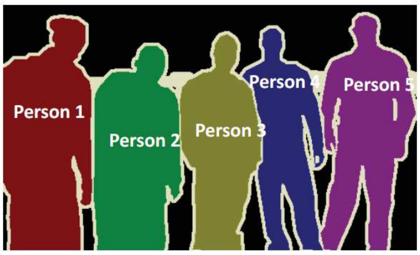
Outline

- Recalls on Convolutional Neural Networks (CNN or ConvNets) and Deep-Learning
- Transfer Learning
- Beyond Image Classification: DETECTION OF OBJECTS
- <u>Instance segmentation with DeepLearning</u>
- DL for Human pose inference and depth estimation
- Semantic segmentation with DeepLearning
- Interest and use of simulations / synthetic videos

Beyond bounding-boxes: getting detailed *contours* of objects of a given category

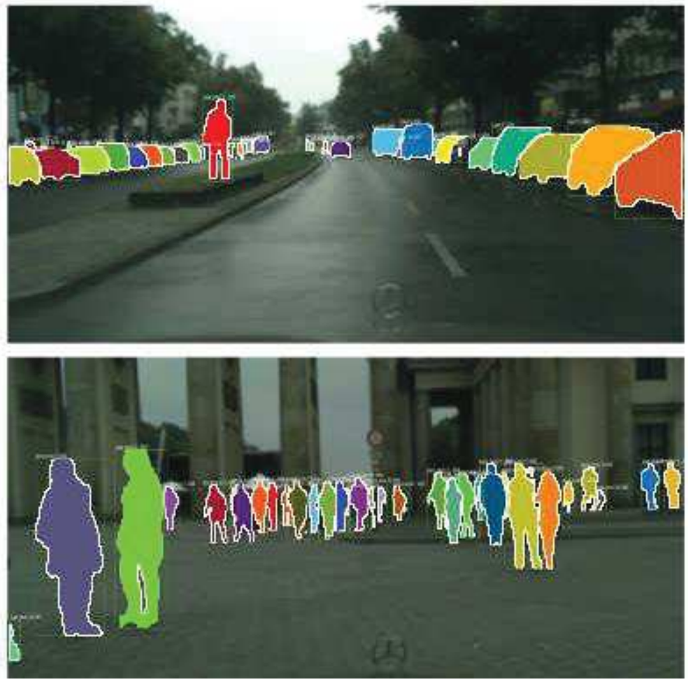
Object Detection

✓



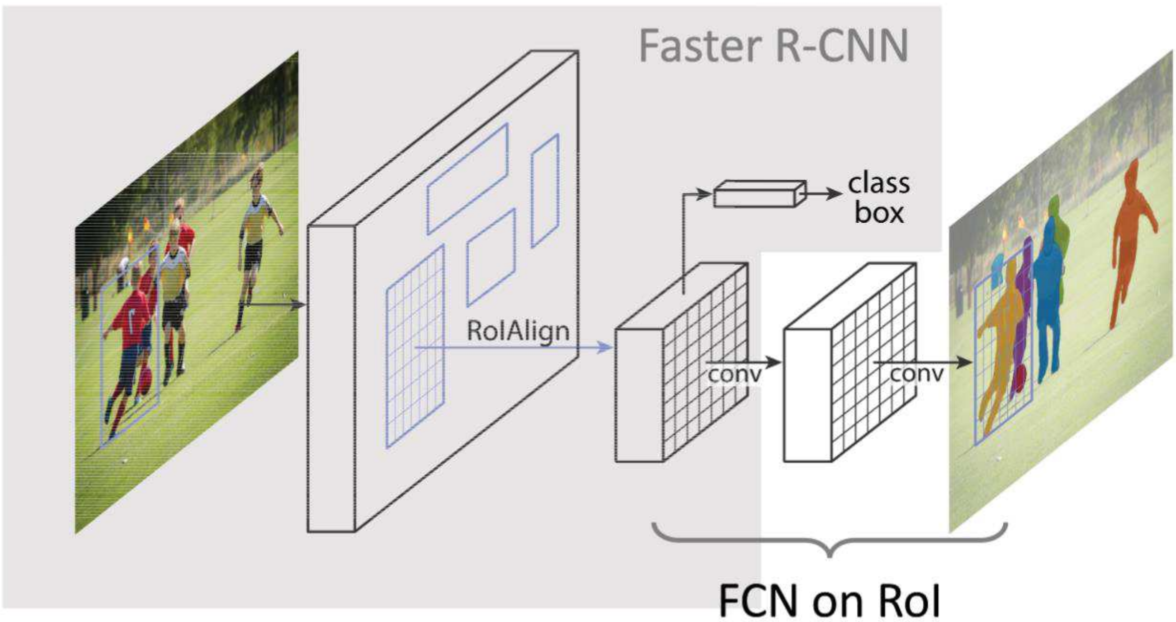
Instance Segmentation

?



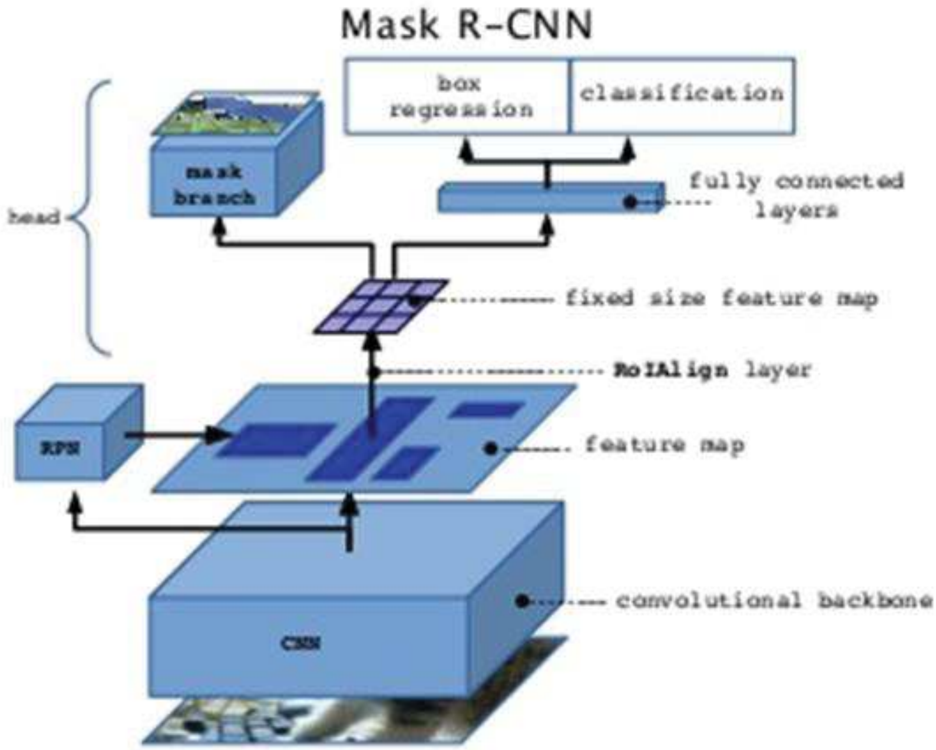
Mask R-CNN principle

Mask R-CNN = **Faster R-CNN** with **FCN** on RoIs



Mask R-CNN architecture extract detailed contours and shape of objects instead of just bounding-boxes

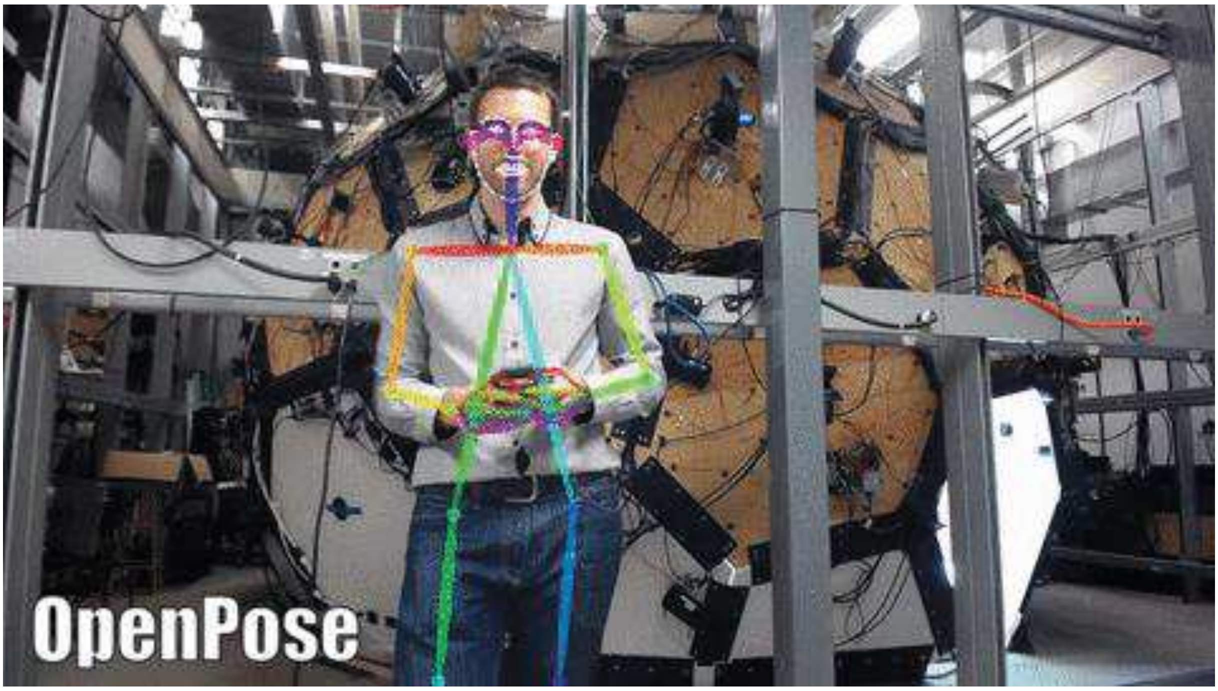
Mask R-CNN architecture



Outline

- **Recalls on Convolutional Neural Networks (CNN or ConvNets) and Deep-Learning**
- **Transfer Learning**
- **Beyond Image Classification: DETECTION OF OBJECTS**
- **Instance segmentation with DeepLearning**
- **<u>DL for Human pose inference and depth estimation</u>**
- **Semantic segmentation with DeepLearning**
- **Interest and use of simulations / synthetic videos**

Now possible to estimate Human poses from RGB images!



Real-time estimation of Human poses on <u>*RGB*</u> video

[Realtime Multi-Person 2D Pose Estimation using Part Affinity Field, Cao et al., CVPR'2017 [CMU]

OpenPose on streets

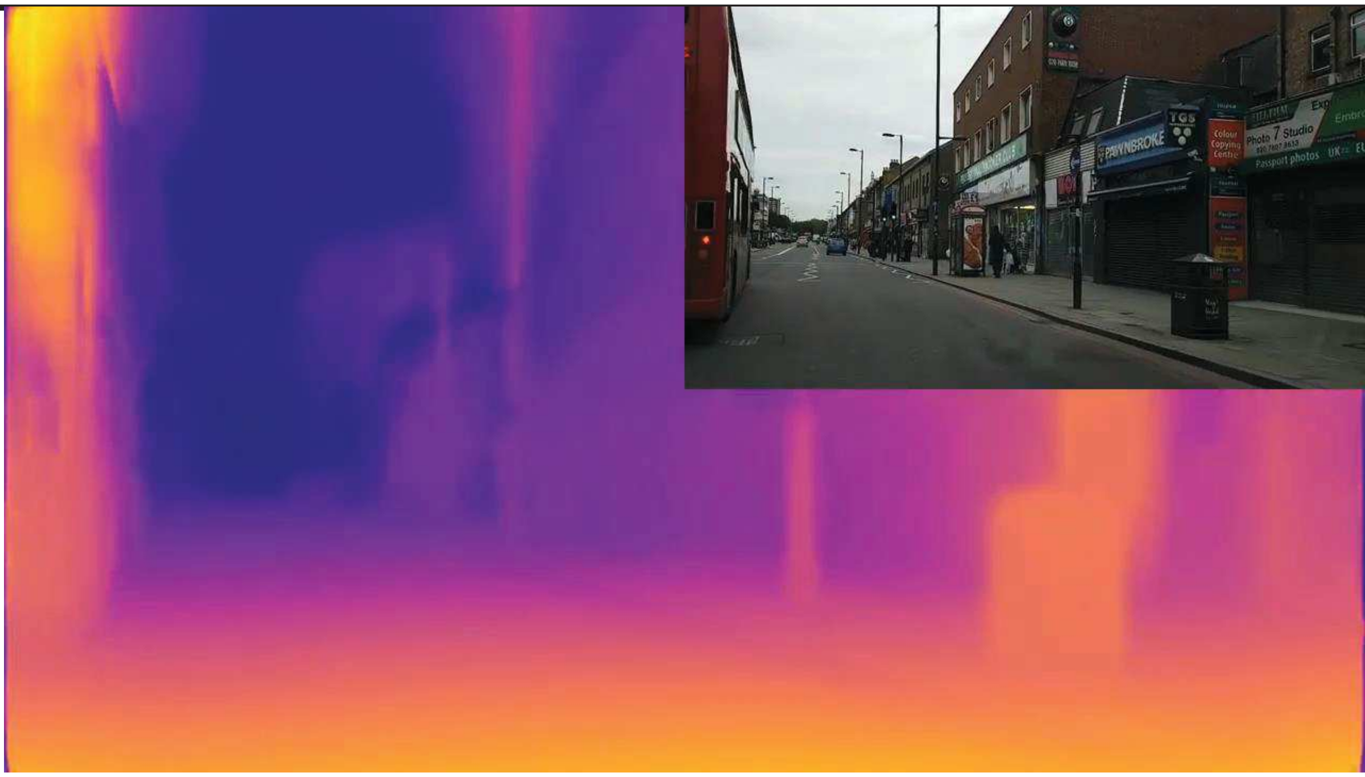
Human Pose estimation by DL methods

- **OpenPose = 2D pose, bottom-up**
  (localize joints, then assemble them into skeletons)

- **AlphaPose = 2D pose, top-down, slower and less robuts**

- **HMR (Human Mesh Recovery) = 3D pose + estimate body SURFACE as a mesh**

Inference of 3D (depth) from monocular vision

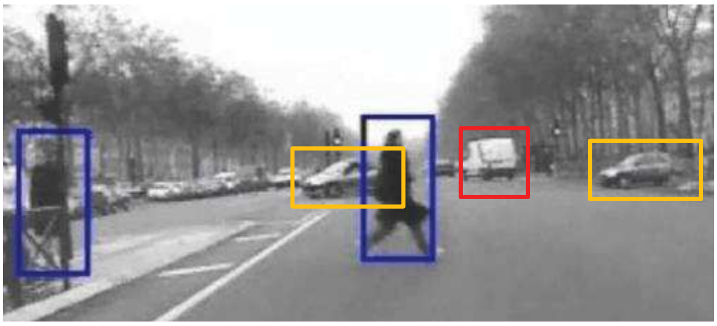


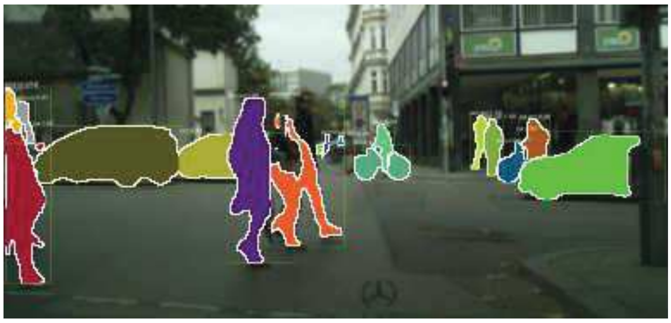
Unsupervised monocular depth estimation with left-right consistency
C Godard, O Mac Aodha, GJ Brostow - CVPR'2017 [UCL]

Outline

- Recalls on Convolutional Neural Networks (CNN or ConvNets) and Deep-Learning
- Transfer Learning
- Beyond Image Classification: DETECTION OF OBJECTS
- Instance segmentation with DeepLearning
- DL for Human pose inference and depth estimation
- <u>Semantic segmentation with DeepLearning</u>
- Interest and use of simulations / synthetic videos

Drawbacks of object detections approach

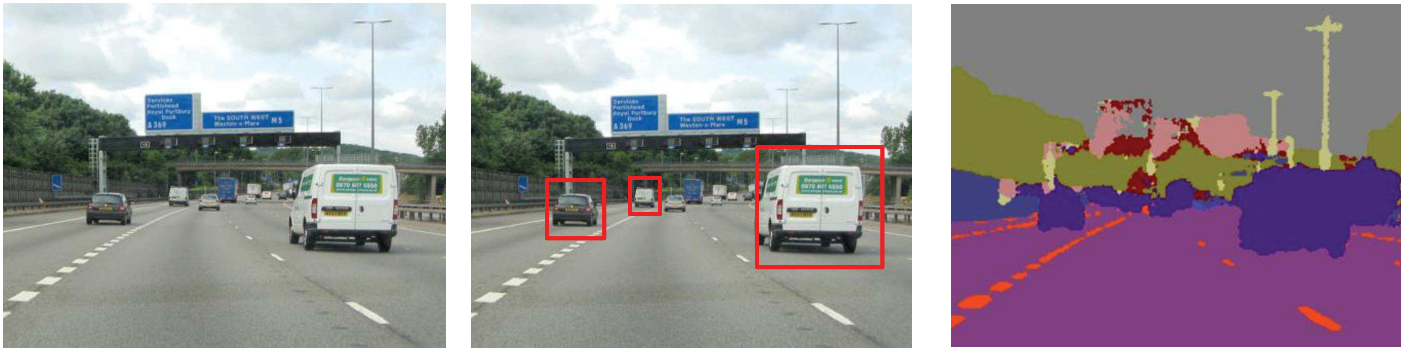




- Problem for objects without sharp boundaries (trees, …) or very dense group of objects (crowd of pedestrians, …)

- Only « compact » objects are categorized
  (what about « road », « sidewalk », « building », …?)

Advantage of Semantic (full) segmentation



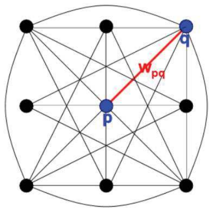
- One single semantic segmenter → all interesting object categories (cars, pedestrians, signs, etc…) and categorization of whole image
- Can also categorize non-compact areas (road, sky, buildings, trees, traffic lanes…)

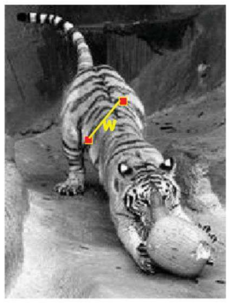
What is image SEGMENTATION?

Identify groups of *contiguous* pixels (connex sets) that « go together »

Many ≠ approaches for image segmentation

- **Clustering (K-means, GMM, MeanShift, …)**
- **Graph-based (graph-cuts)**

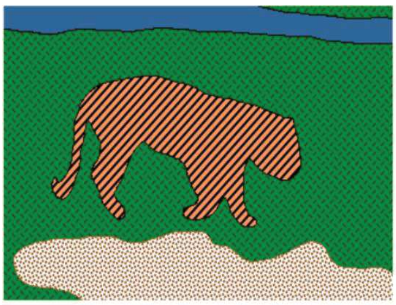


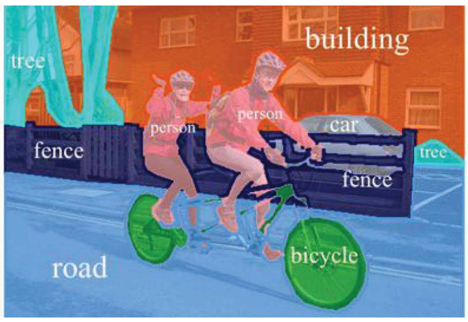


  - Node (vertex) for every pixel
  - Edge between pairs of pixels, (p,q)
  - Affinity weight w_{pq} for each edge
    - w_{pq} measures similarity
    - Similarity is inversely proportional to difference (in color and position…)

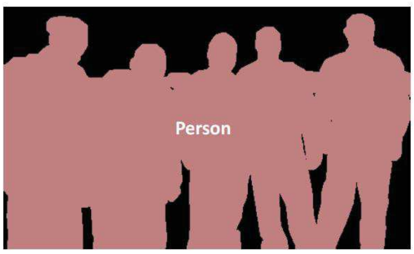
- **Mathematical Morphology (watershed, etc…)**
- **Energy minimization (Conditional Random Fields)**
- **Deep-Learning**

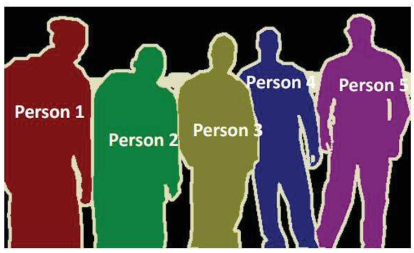
What is SEMANTIC Image Segmentation?





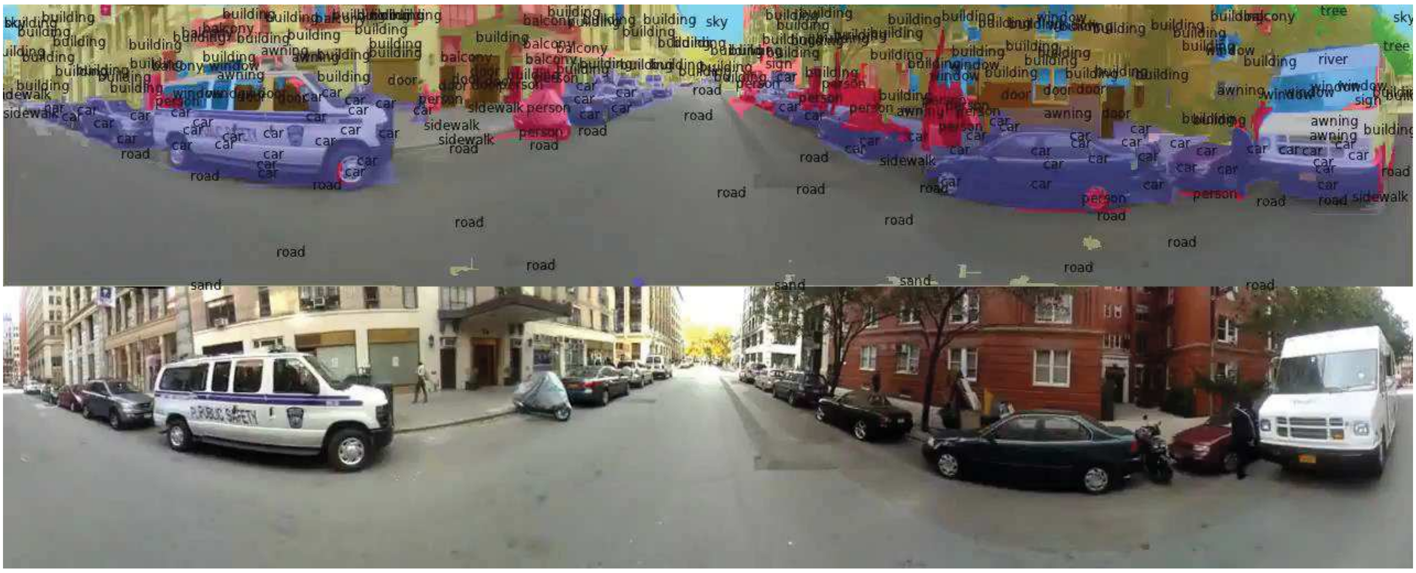
SEMANTIC segmentation:
« go together » = same « type of object »
≠ from just grouping pixels with similar colors or texture





Objects detection **Semantic Segmentation** Instance Segmentation

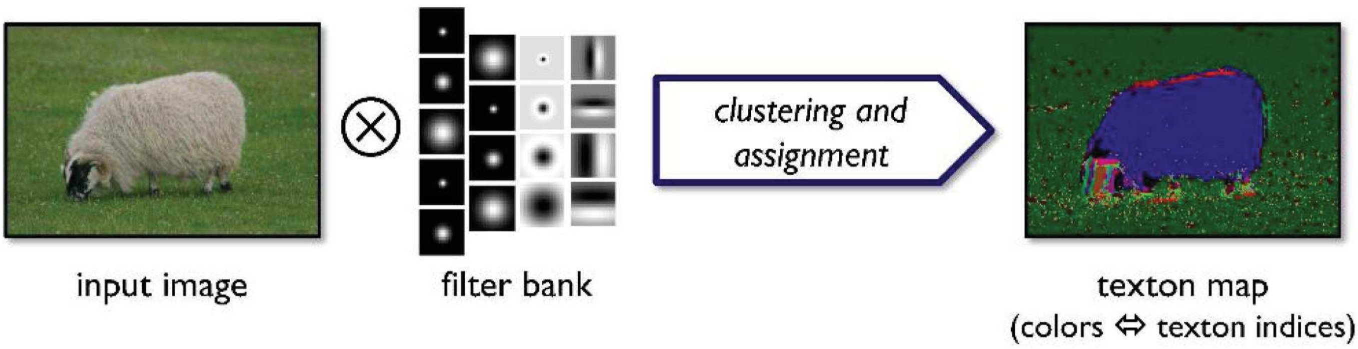
Video example of semantic segmentation *with category labels*



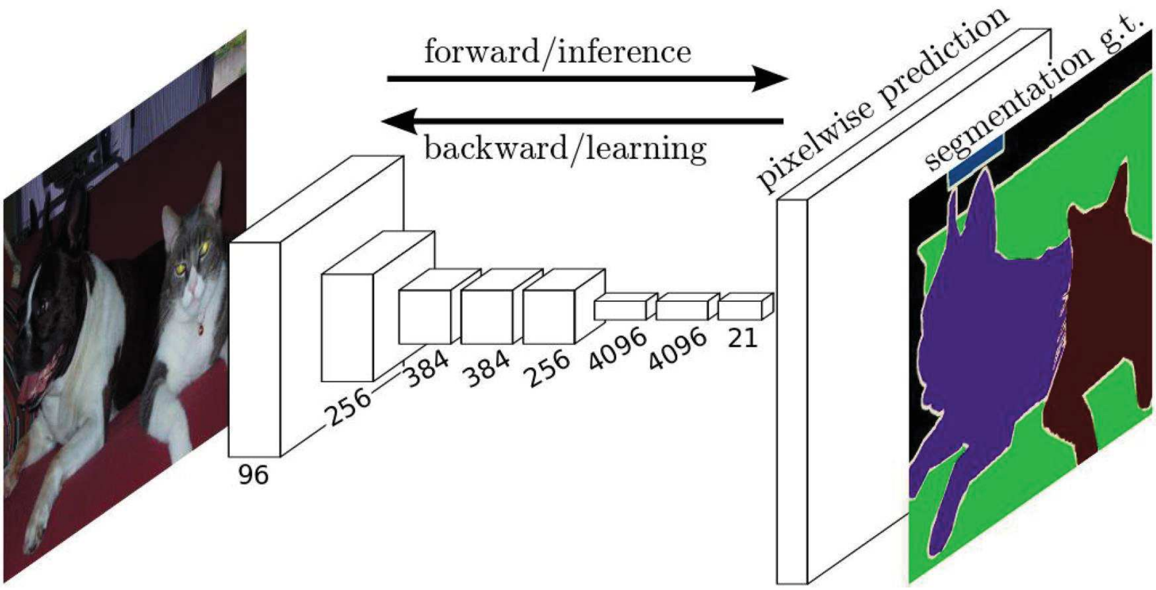
[C. Farabet, C. Couprie, L. Najman & Yann LeCun: Learning Hierarchical Features for Scene Labeling, IEEE Trans. PAMI, Aug.2013.

Semantic segmentation BEFORE Deep-Learning

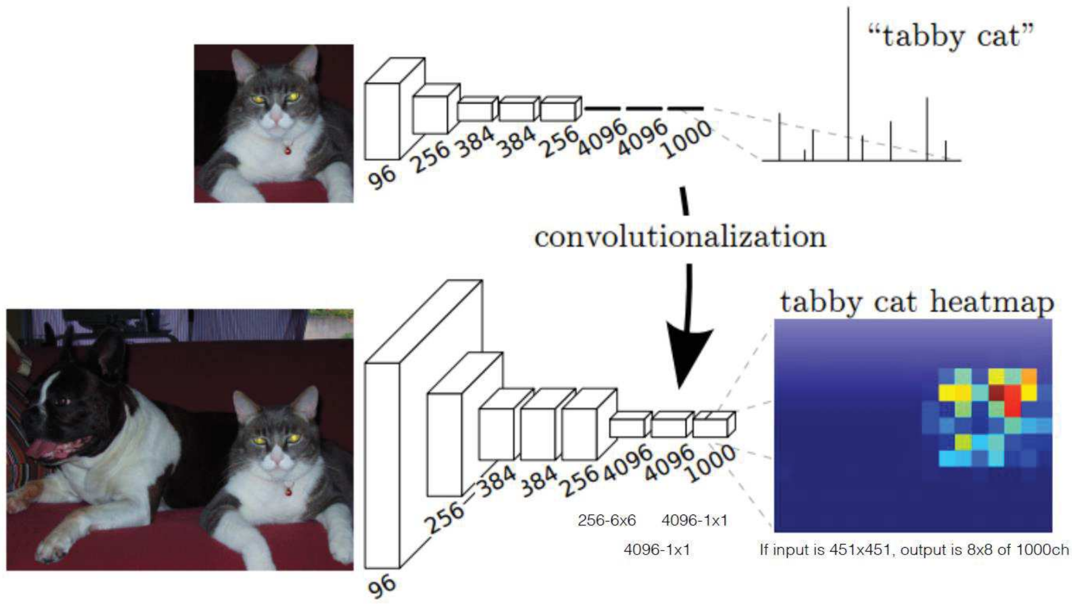
- Relying on Conditional Random Field (CRF)
- Operating on pixels or superpixels
- Interactions between label assignments



Deep-Learning approach for semantic segmentation

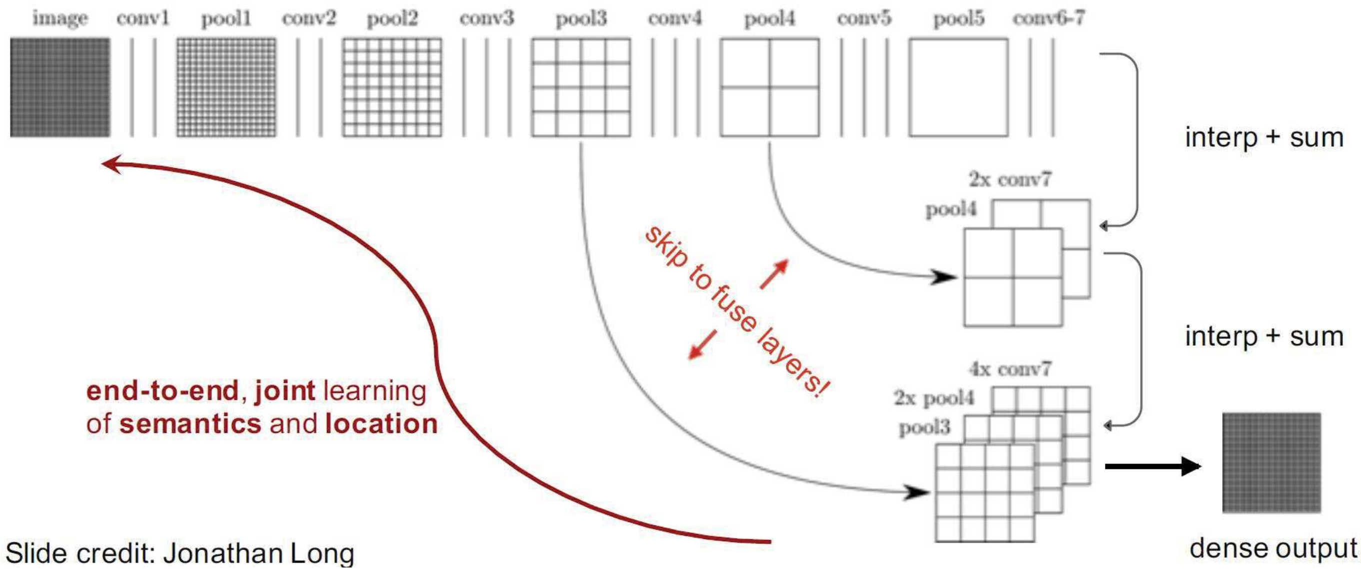


Fully Convolutional Network (FCN)



« Fully Convolutional Networks for Semantic Segmentation », Evan Shelhamer, Jonathan Long, and Trevor Darrell, *[Berkeley, 2015]*

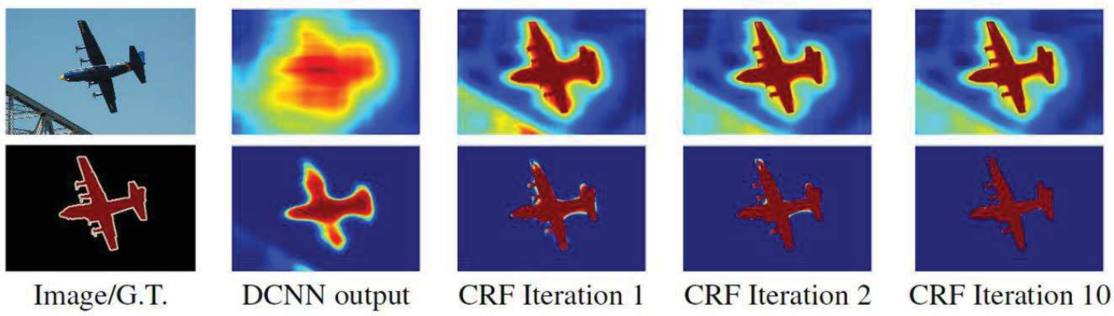
FCN principle



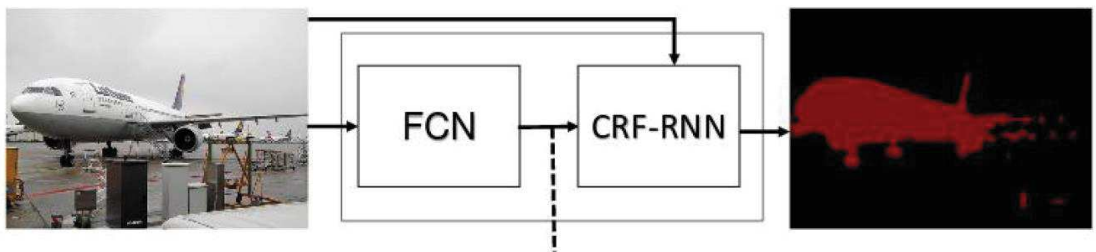
Slide credit: Jonathan Long

Trick = some connections skipping directly to « fuse layers »

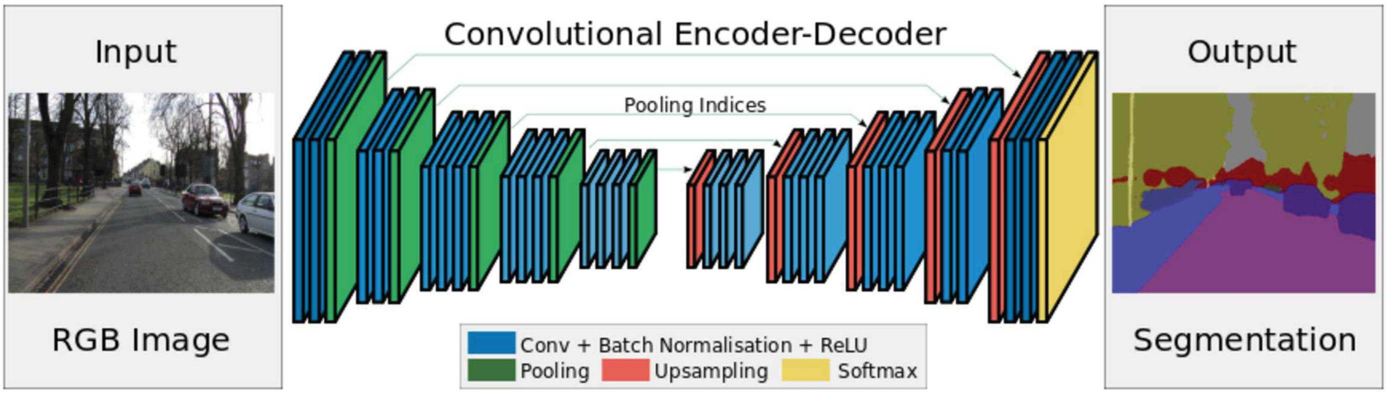
FCN + CRF

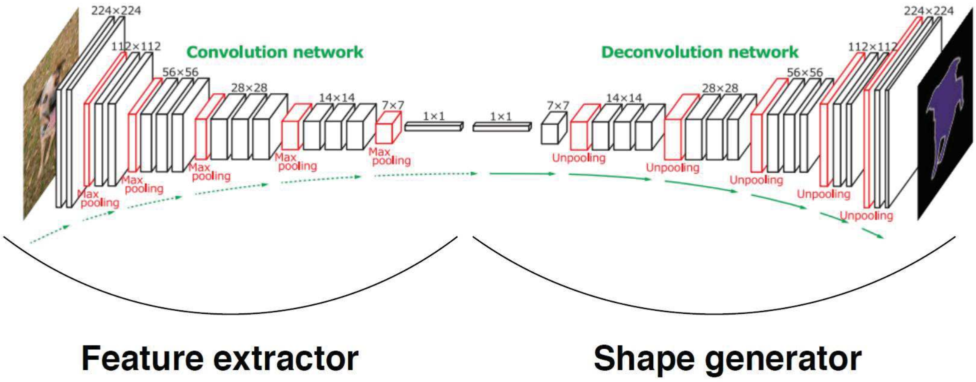


Output from FCN rather blurry and inaccurate, but can be improved by CRF post-processing

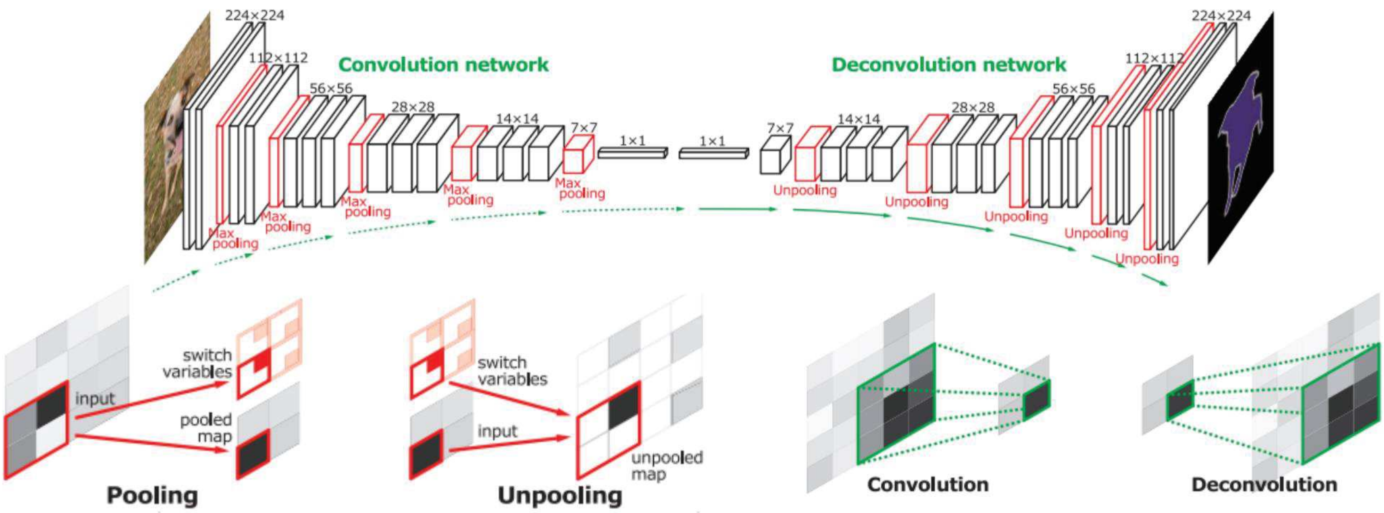


Convolutional Encoder-Decoder

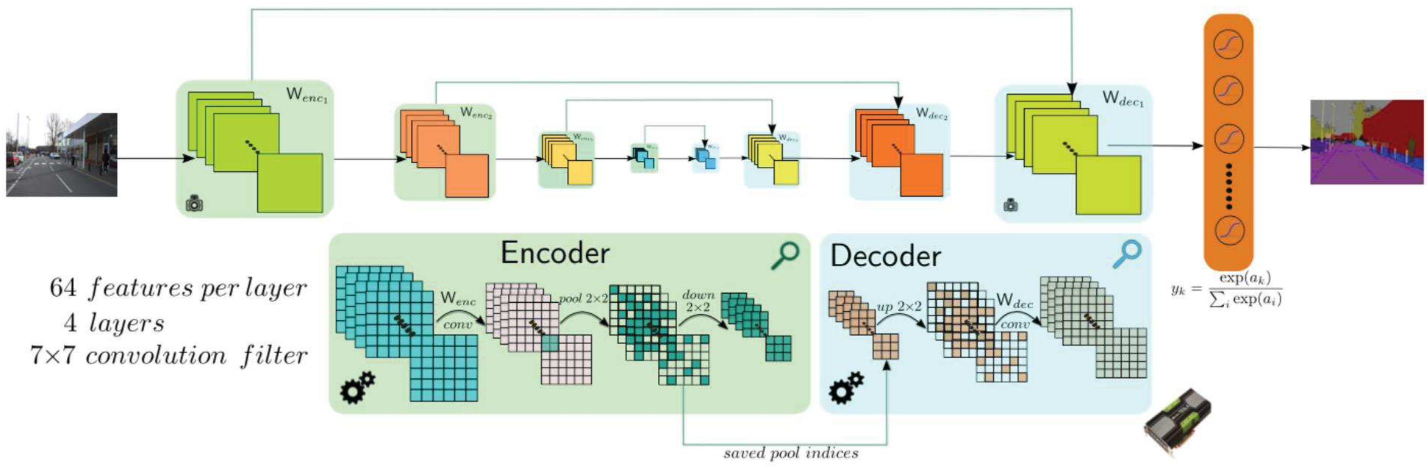




Deconvolution??

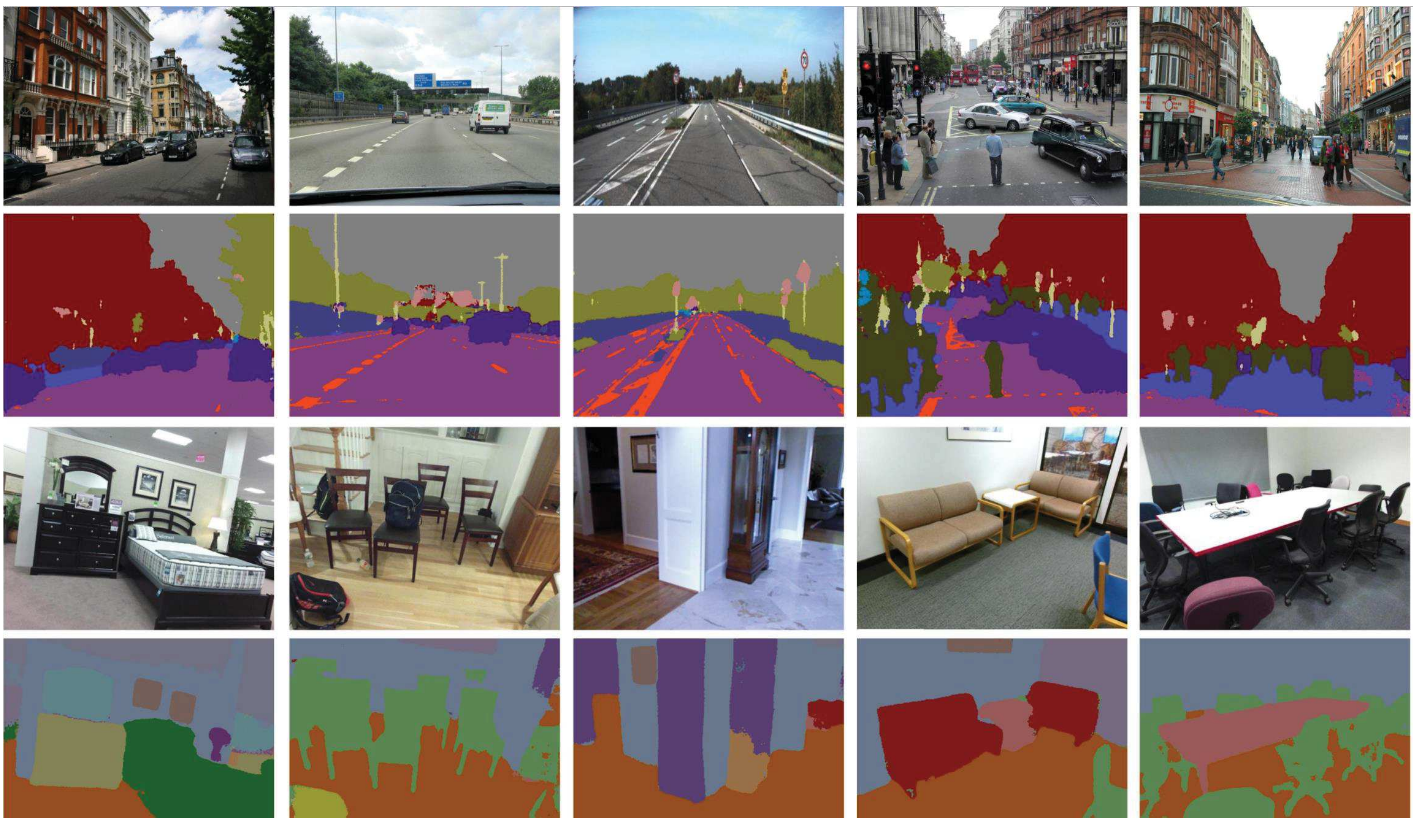


SegNet

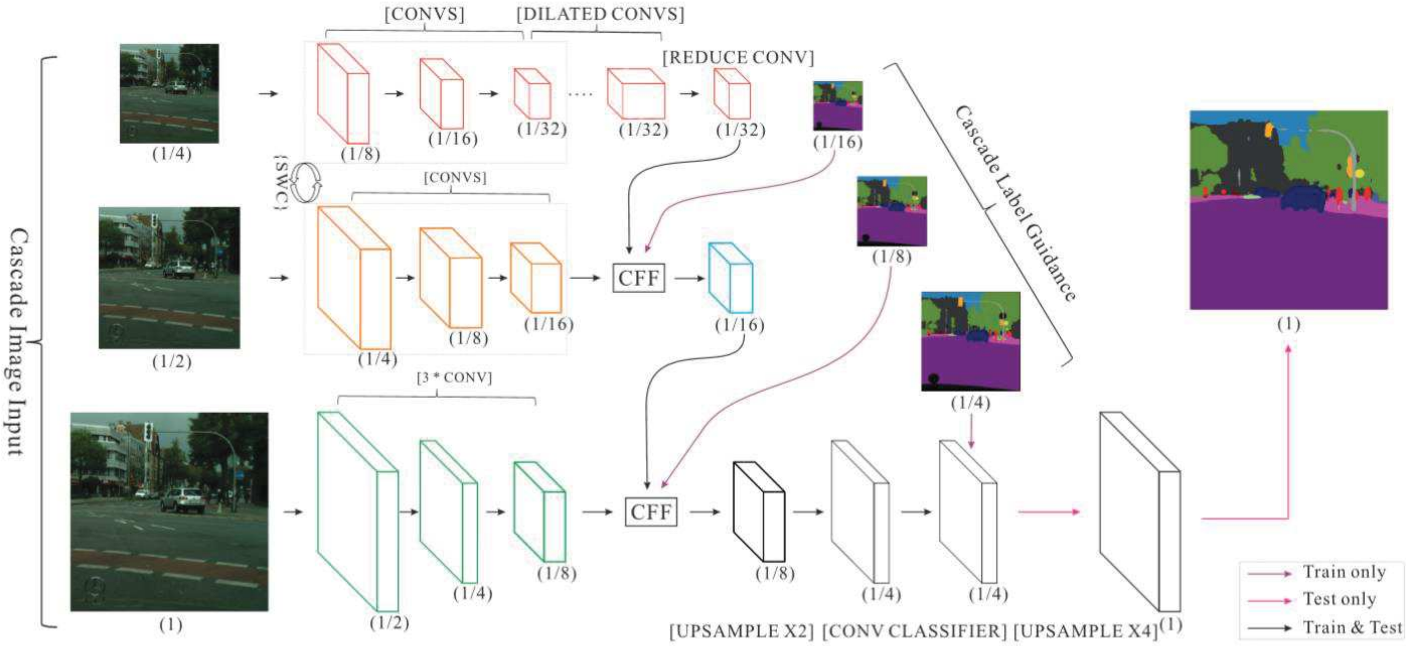


"SegNet: A Deep Convolutional Encoder-Decoder Architecture for ImageSegmentation", Vijay Badrinarayanan, Alex Kendall, Roberto Cipolla *[Cambridge (UK), 2015]*

SegNet example results



ICnet



« ICNet for Real-Time Semantic Segmentation on High-Resolution Images »,
Zhao, Hengshuang & Qi, Xiaojuan & Shen, Xiaoyong & Shi, Jianping & Jia, Jiaya.
Chinese University of Hong-Kong (2017).

And many other competitors!

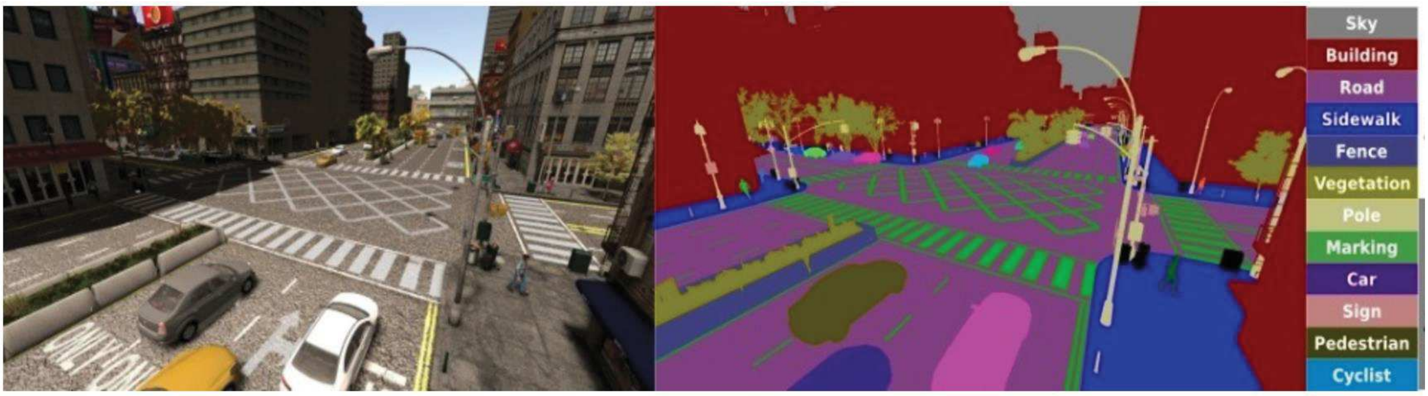
- **2015: U-Net (Keras) - https://github.com/zhixuhao/unet**
- **RefineNet (2016)**
- **DeepLab (Caffe) - https://github.com/Robotertechnik/Deep-Lab**
- **DeepLabv3 (Tensorflow) - https://github.com/NanqingD/DeepLabV3-Tensorflow**

Outline

- Recalls on Convolutional Neural Networks (CNN or ConvNets) and Deep-Learning
- Transfer Learning
- Beyond Image Classification: DETECTION OF OBJECTS
- Instance segmentation with DeepLearning
- DL for Human pose inference and depth estimation
- Semantic segmentation with DeepLearning
- <u>Interest and use of simulations / synthetic videos</u>

Synthetic images

More and more realistic



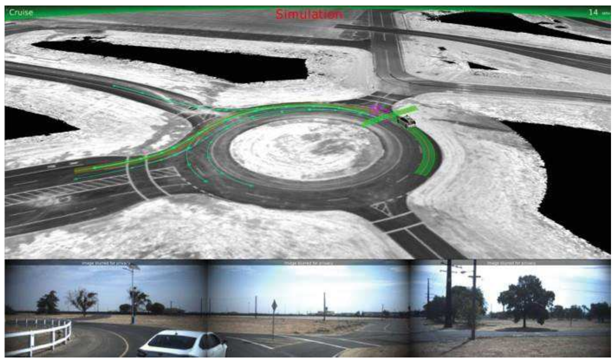
Example from SYNTHIA (http://synthia-dataset.net)

Interest of synthetic images for Machine-Learning in IV applications

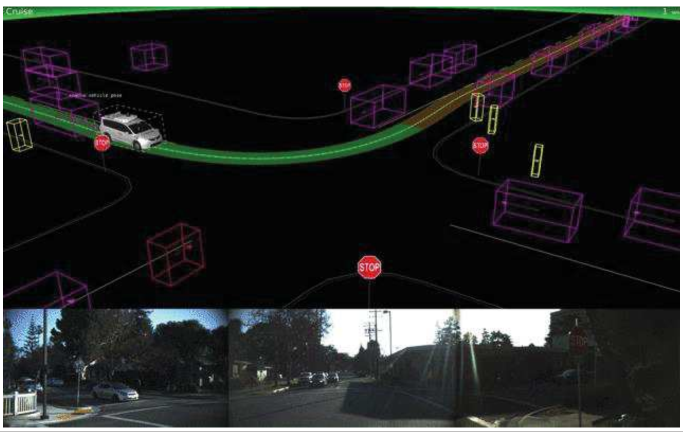
- **Possible to generate as many as needed at nearly no cost (in particular compared to recording while driving)**
- **Easy to generate controlled variability in environment, luminosity conditions, scenarii, etc + also images « dangerous situations »**
- **NO NEED FOR MANUAL LABELLING: ground truth (ie target value) for classifiers, localizers, and semantic segmentation provided automatically**

Simulators dedicated to Autonomous Vehicles

Scenario-buiding with CarCraft by Google/Waymo



Simulation of a virtual scenario in XView by Google/Waymo



CARLA open-source urban driving simulator

- **Still few driving simulators adapted for DL and RL, and best ones not totally mature**

Simulateur	GTA	DeepDrive.io	AirSim	CARLA[1]
Flexibilité	−−	++	++	++
Variété	++	−−	−	+
Complexité/Réalisme	++	−−	−	−
Objets mobiles	++	−−	−−	+
Vitesse exécution	−−	+	+	+
Multi-agent	−−	−	−	++

→ Choice of CARLA

[1] A. Dosovitskiy: CARLA: An Open Urban Driving Simulator (2017)

CARLA simulator

Synthetic images use in ML/DL for IV

- Initial training of a classifier / segmenter / controller only on simulated images / videos / scenarios
- Possible to then adaptation to real-world by fine-tuning on REAL images/video datasets
- Cheaper / more extensive testing than on real-world videos

- REINFORCEMENT LEARNING in simulation !

Examples of autonomous driving obtained by DRL in CARLA

Work by my PhD student Marin Toromanoff (Valeo/MINES). Ranked 1st (vision-only track) on CARLA "*Autonomous Driving challenge*" !!