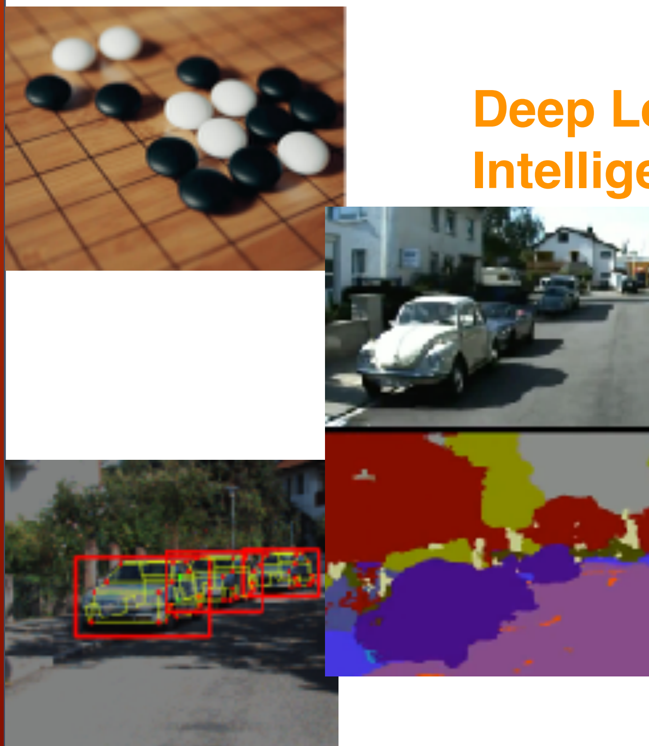
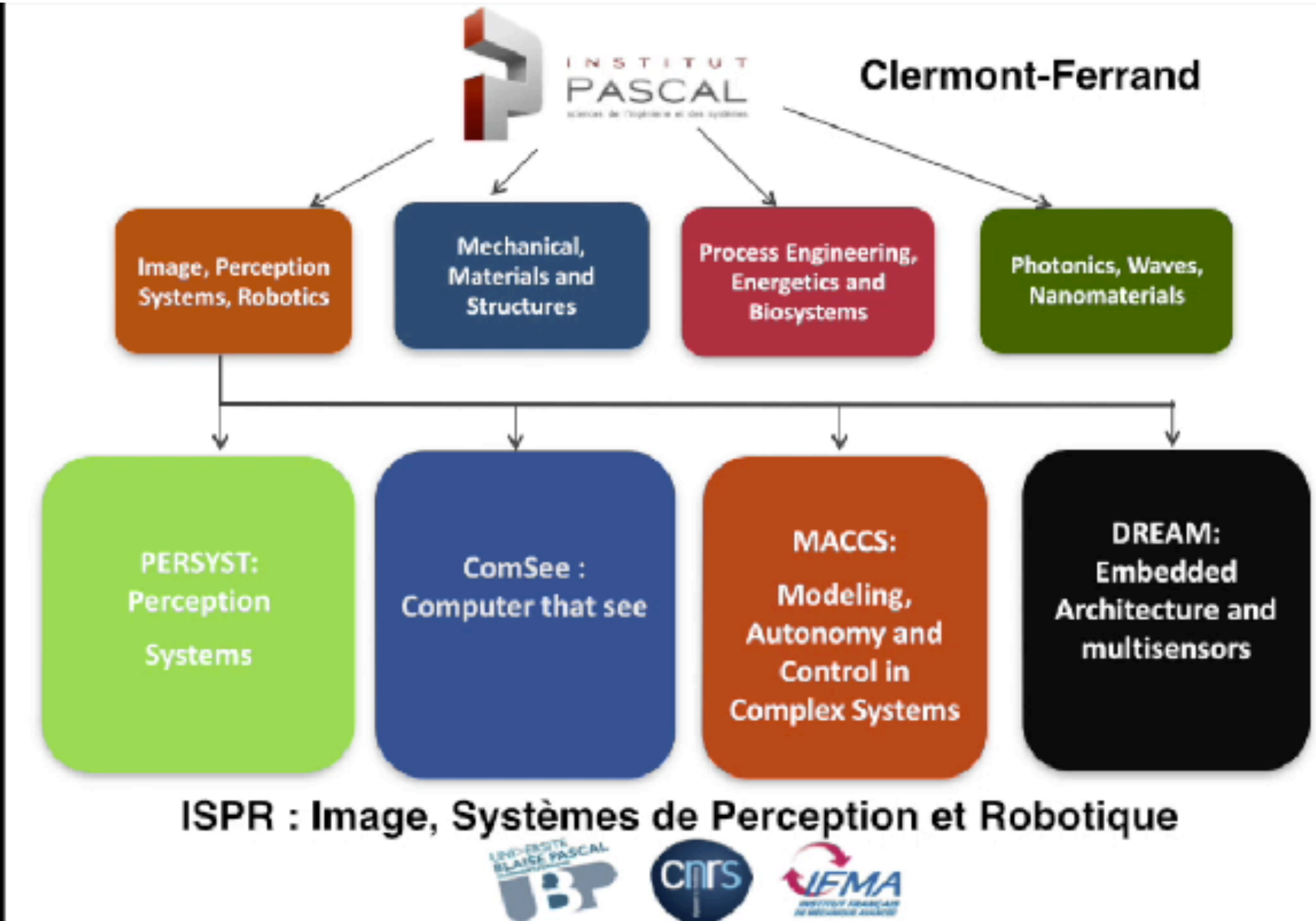


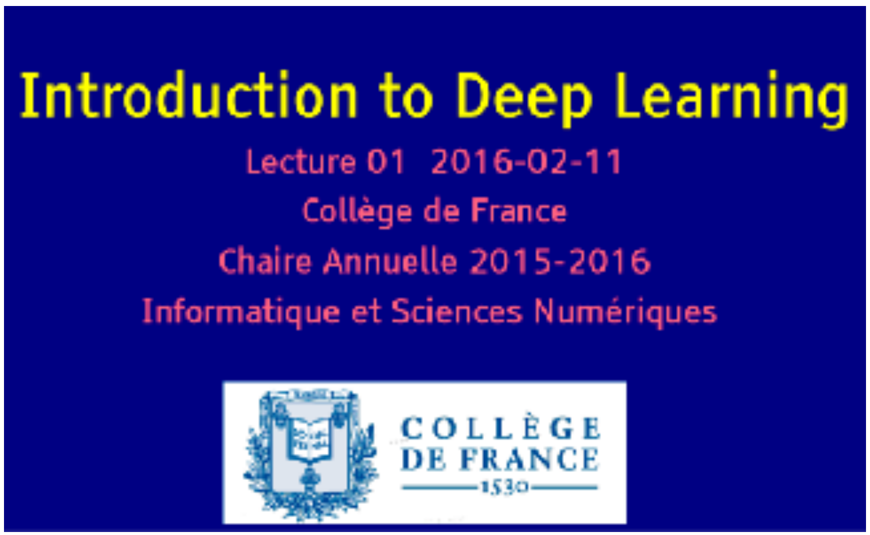
Deep Learning and Applications to Intelligent Transportation Systems

T. CHATEAU, ISPR/ Pascal Institute
UMR 6602, CNRS/UBP/IFMA,
Clermont Ferrand, France

ISPR : ImageS, Perception systems and Robotics

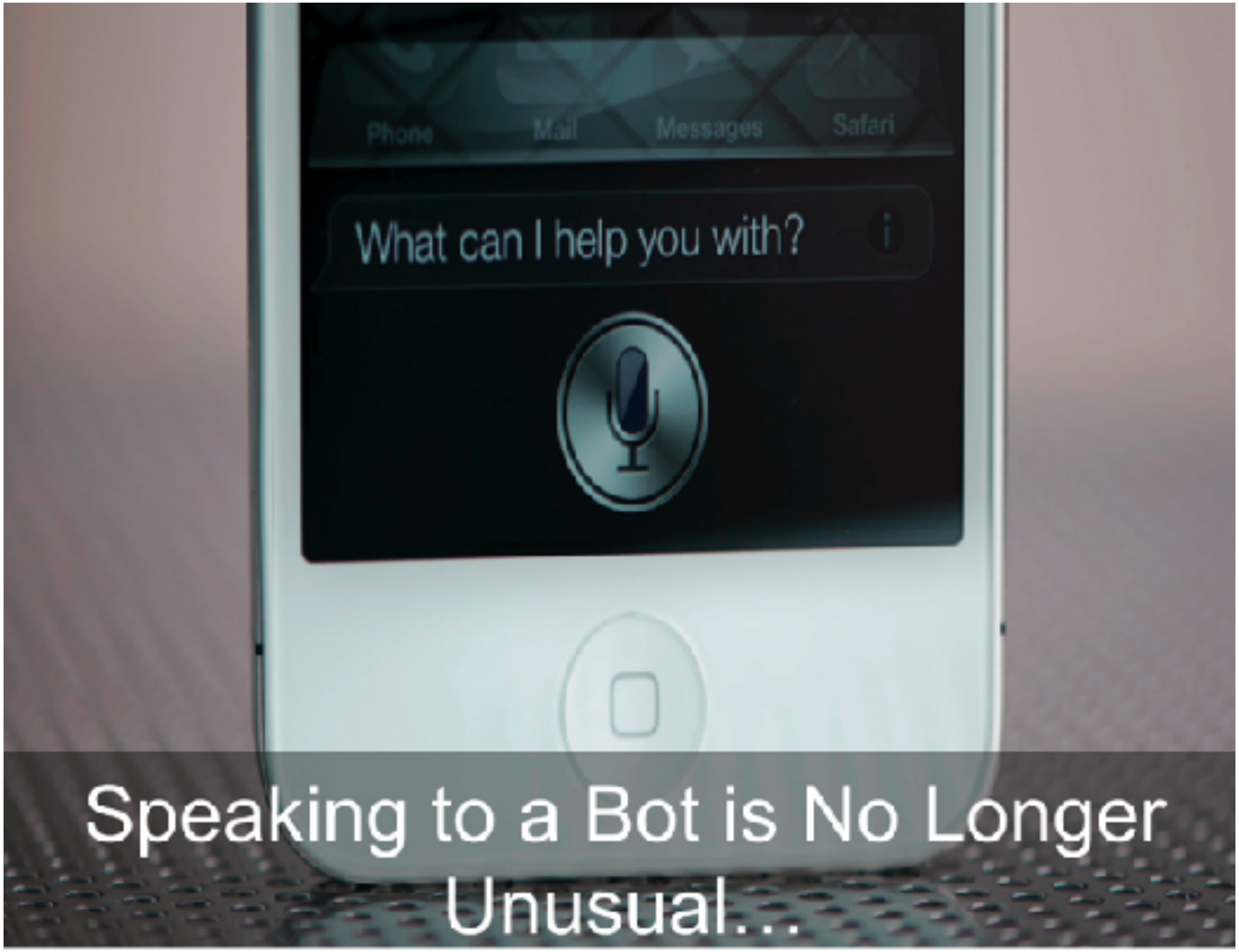


References



http://www.fhnw.ch/technik/bachelor/informatik/computer-science-seminar/archiv/Deep_Learning.pptx

Content


Phone
Mail
Messages
Safari
What can I help you with?
Speaking to a Bot is No Longer Unusual…


March 2016:
World Go Champion
Beaten by Machine

A new revolution seems to be in the work after the industrial revolution.

Devices are becoming intelligent.

And Deep Learning is at the epicenter of this revolution.

IT Companies are Racing into Deep Learning

AI Breakthrough

- **Deep Learning:** machine learning algorithms based on learning multiple levels of representation / abstraction.

Amazing improvements in error rate in object recognition, object detection, speech recognition, and more recently, in natural language processing / understanding

Breakthrough in deep learning

A Canadian-led trio at CIFAR initiated the deep learning AI revolution

- Fundamental breakthrough in 2006:

first successful recipe for training a deep supervised neural network

- Second major advance in 2011, with rectifiers
- Breakthroughs in applications since then, especially the AlexNet 2012.

Canadian Institute for Advanced Research

Machine Learning

The machine learning framework

- Apply a prediction function to a feature representation of the image to get the desired output:

f() = “apple”

f() = “tomato”

f() = “cow”

Traditional Machine Learning

Training

Training Images

Image Features → Training → Learned model

Training Labels → Training

Testing

Test Image

Image Features → Learned model → Prediction

Slide credit: D. Hoiem and L. Lazebnik

DEEP LEARNING = Learning Representations/Features

- **The traditional model of pattern recognition (since the late 50's)**
  - Fixed/engineered features (or fixed kernel) + trainable classifier

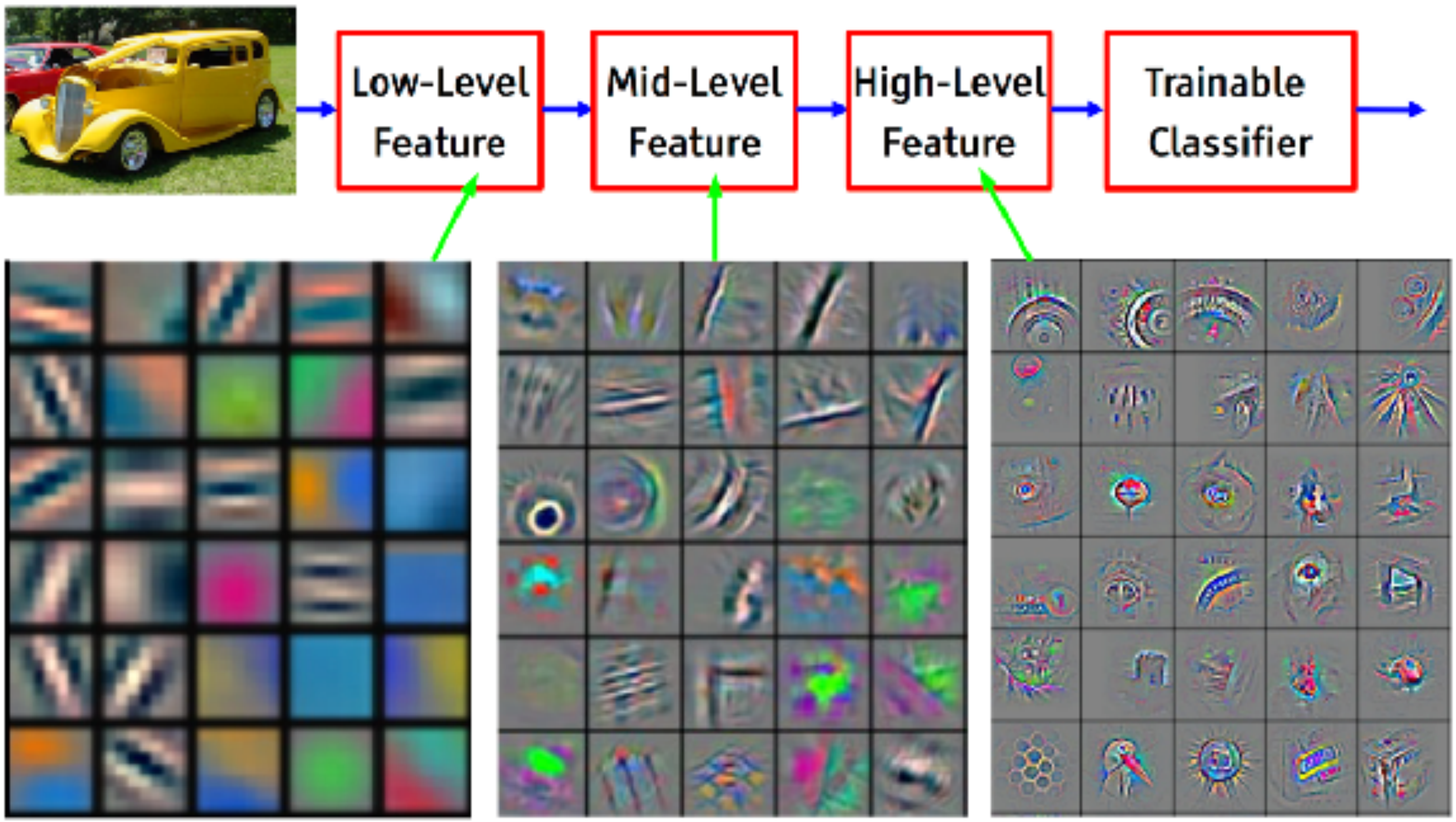
hand-crafted Feature Extractor → "Simple" Trainable Classifier →

- **End-to-end learning / Feature learning / Deep learning**
  - Trainable features (or kernel) + trainable classifier

Trainable Feature Extractor → Trainable Classifier →

Deep Learning = Learning Hierarchical Representations

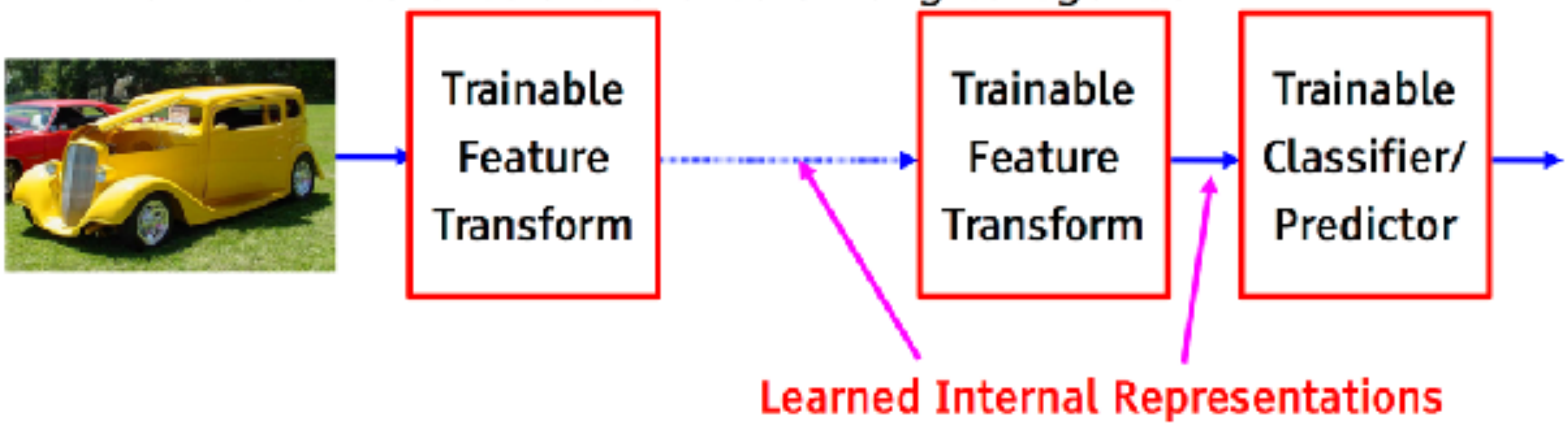
- It's deep if it has more than one stage of non-linear feature transformation



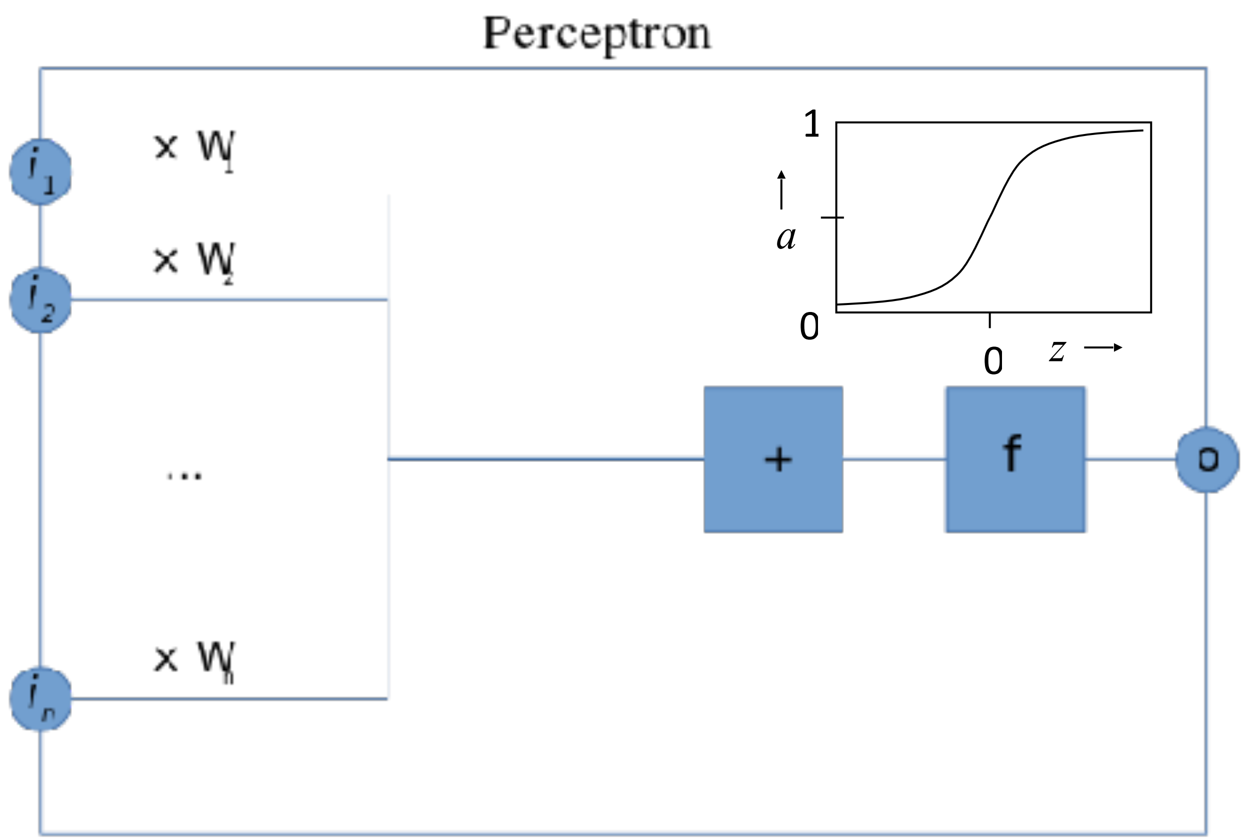
Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

Introduction to Deep Learning

- **A hierarchy of trainable feature transforms**
  - Each module transforms its input representation into a higher-level one.
  - High-level features are more global and more invariant
  - Low-level features are shared among categories



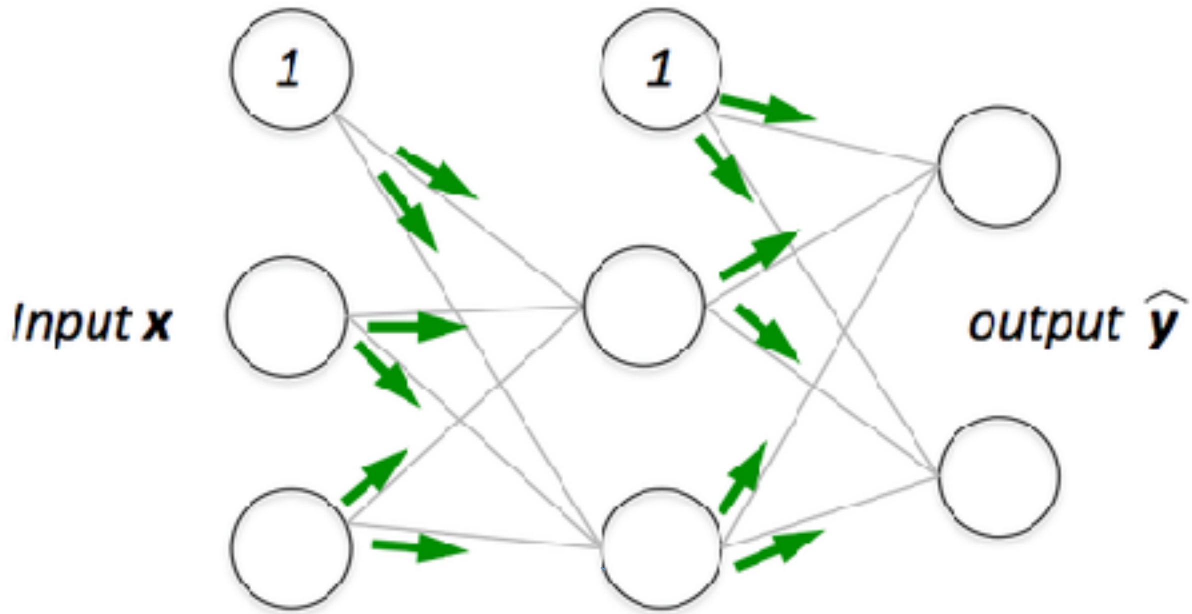
One Neuron



$$o = f\left(\sum_{k=1}^{n} i_k \cdot W_k\right)$$

https://fr.wikipedia.org/wiki/Perceptron

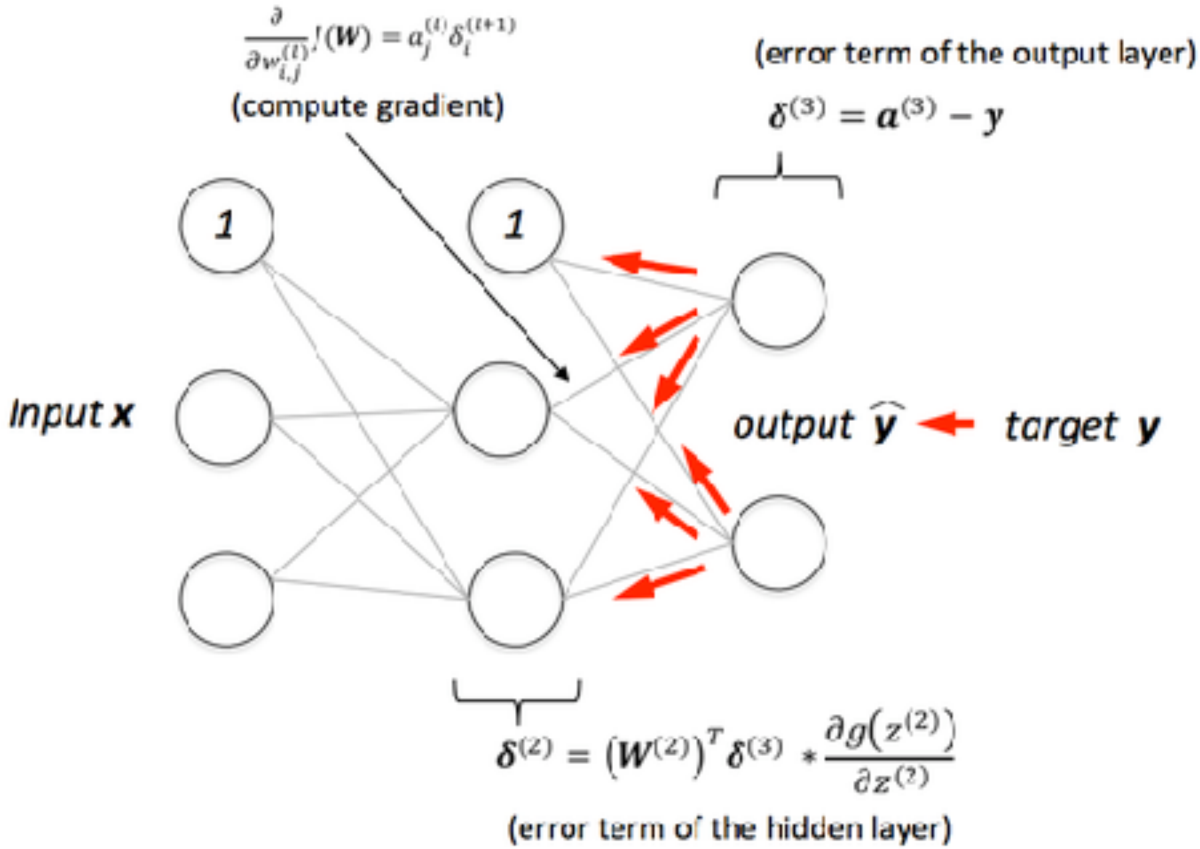
Multi Layer Perceptron



- Multiple Layers
- Feed Forward
- Connected Weights
- 1-of-N Output

i

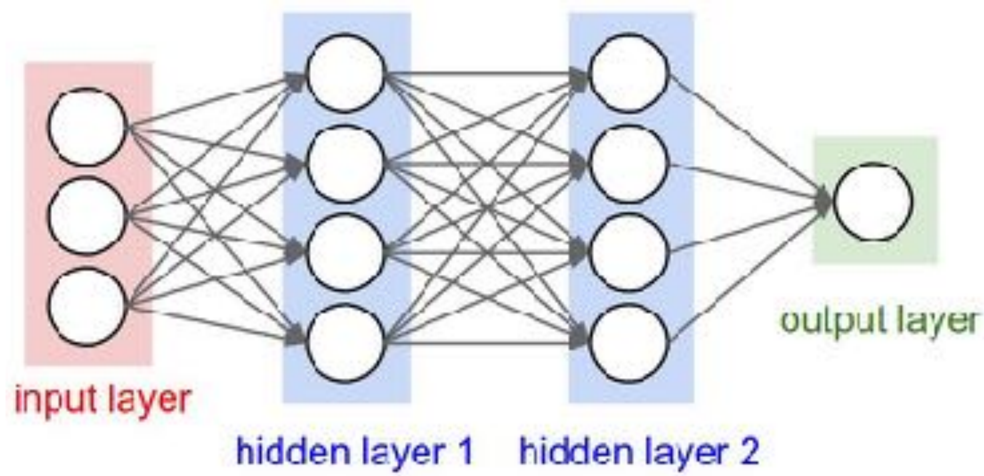
Backpropagation

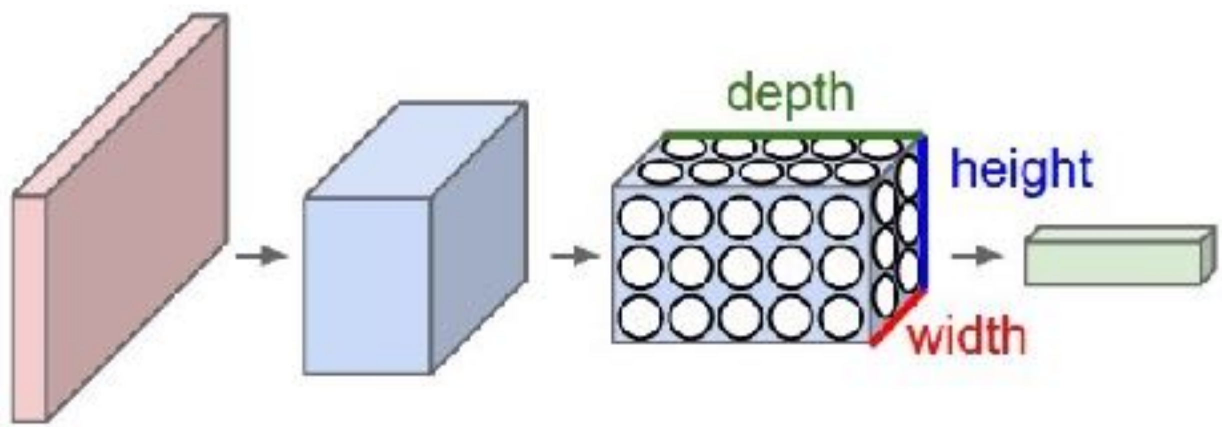


http://sebastianraschka.com/faq/docs/visual-backpropagation.html

Deep Convolutional Neural Network (DCNN)

Arranges neurons in 3D

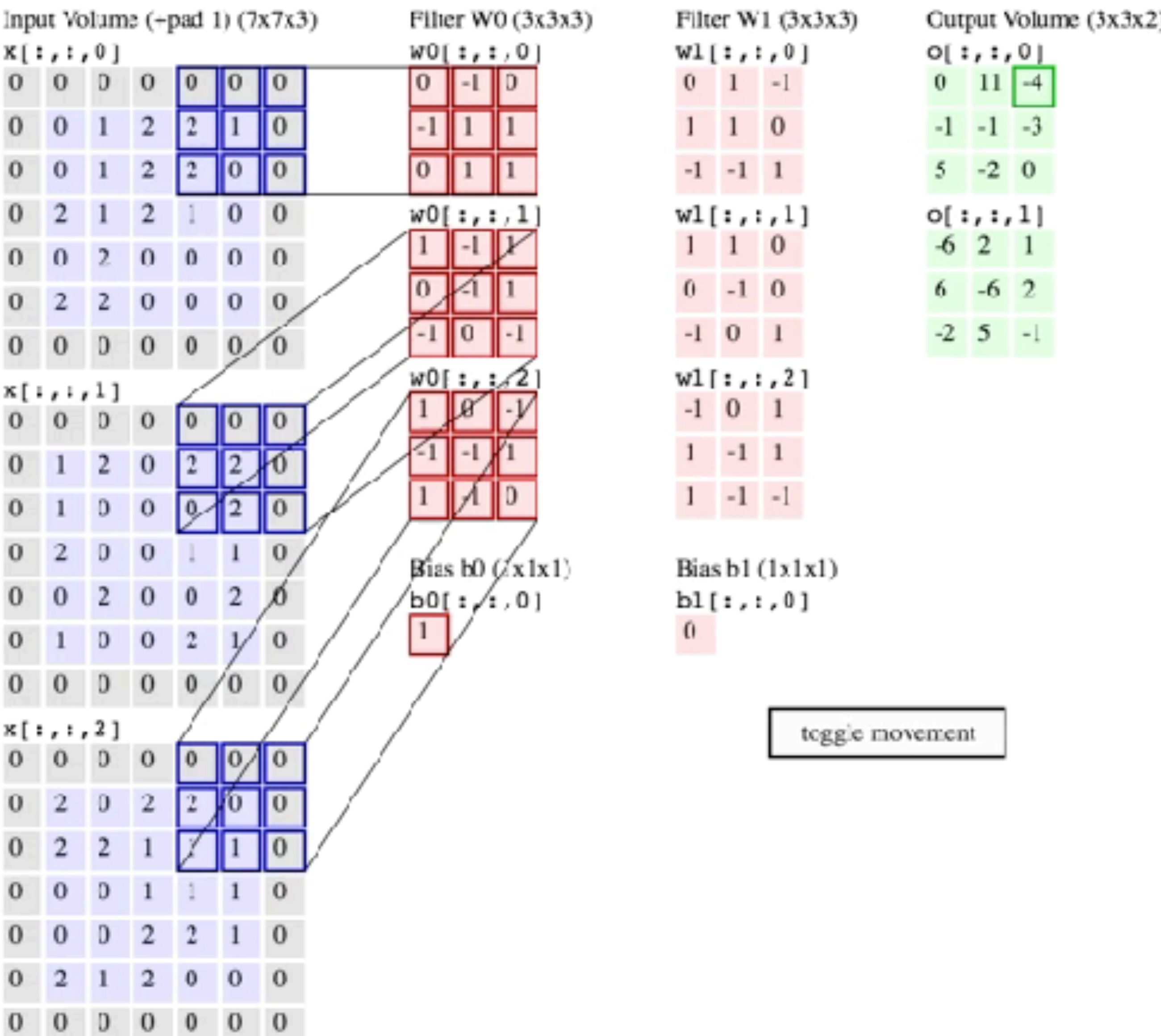




http://cs231n.github.io/convolutional-networks/

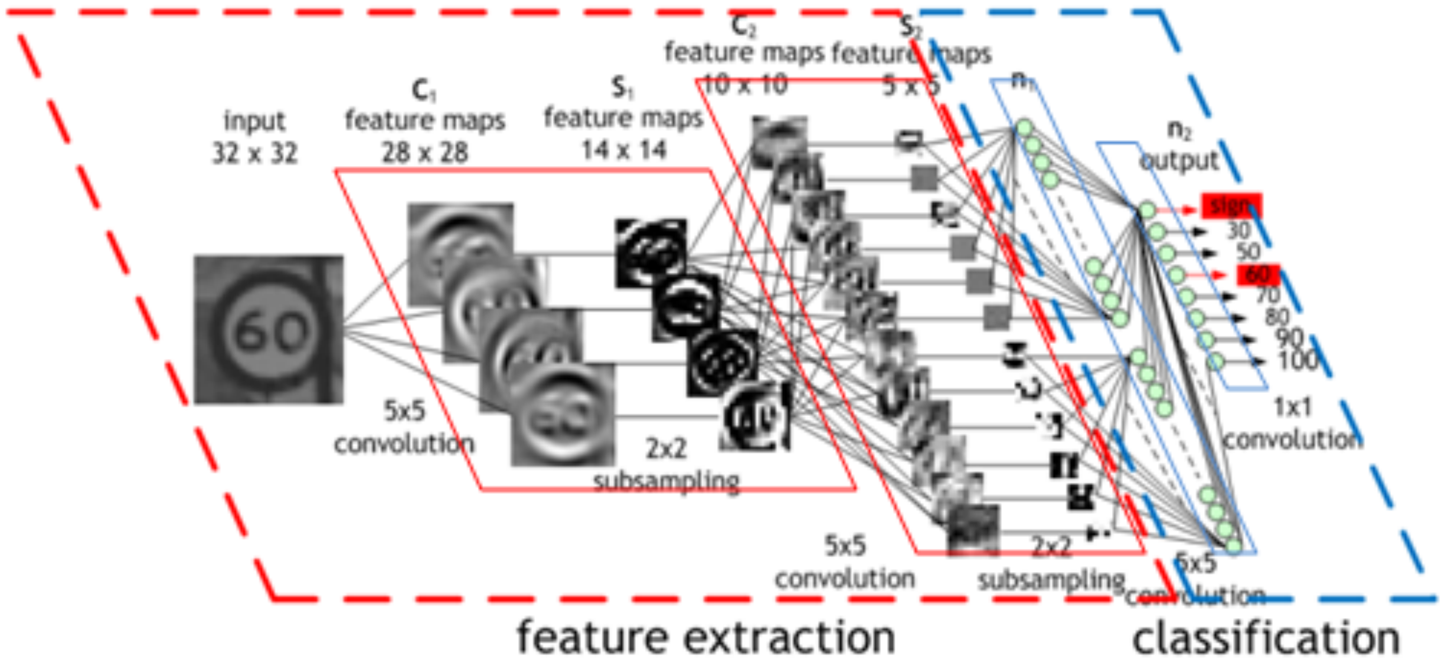
Deep Convolutional Neural Network (DCNN)

Convolution



Deep Convolutional Neural Network (DCNN)

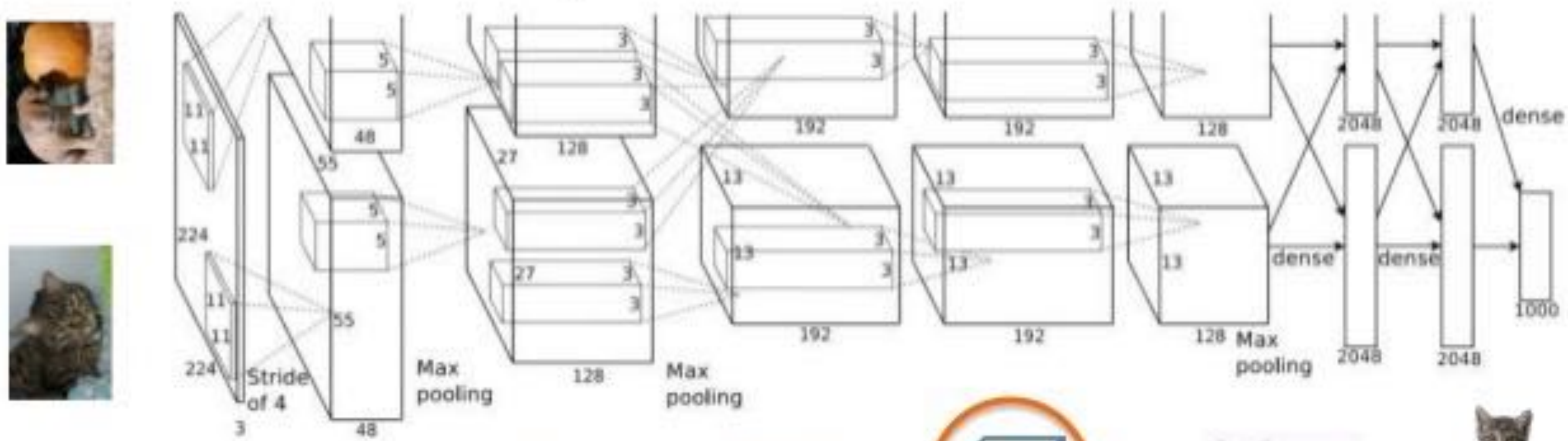
DCNN for trafic sign recognition



DCNN networks are more and more deeper

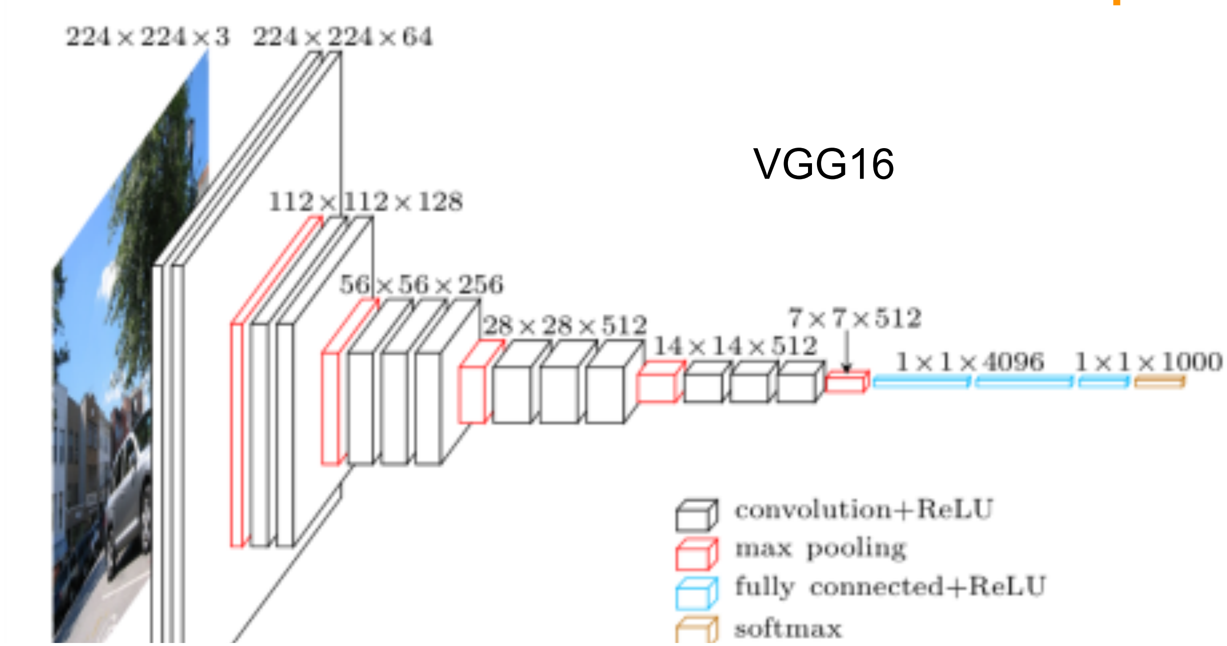
AlexNet (Krizhevsky et al. 2012)

The class with the highest likelihood is the one the DNN selects

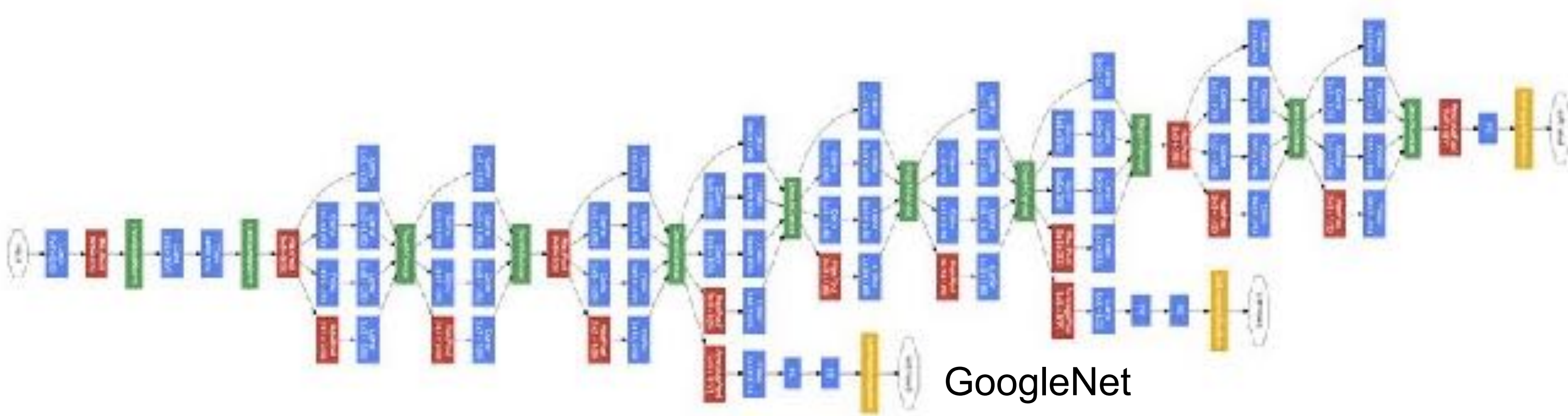


Deep Convolutional Neural Network (DCNN)

DCNN networks are more and more deeper



VGG16



GoogleNet

Deep Convolutional Neural Network (DCNN)

DCNN networks are more and more deeper

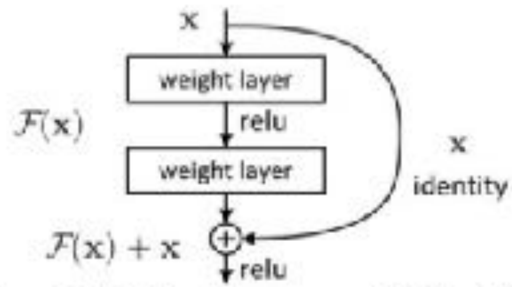
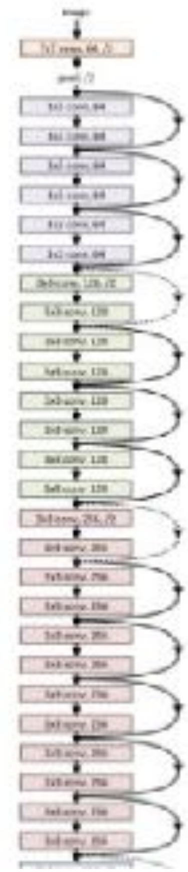


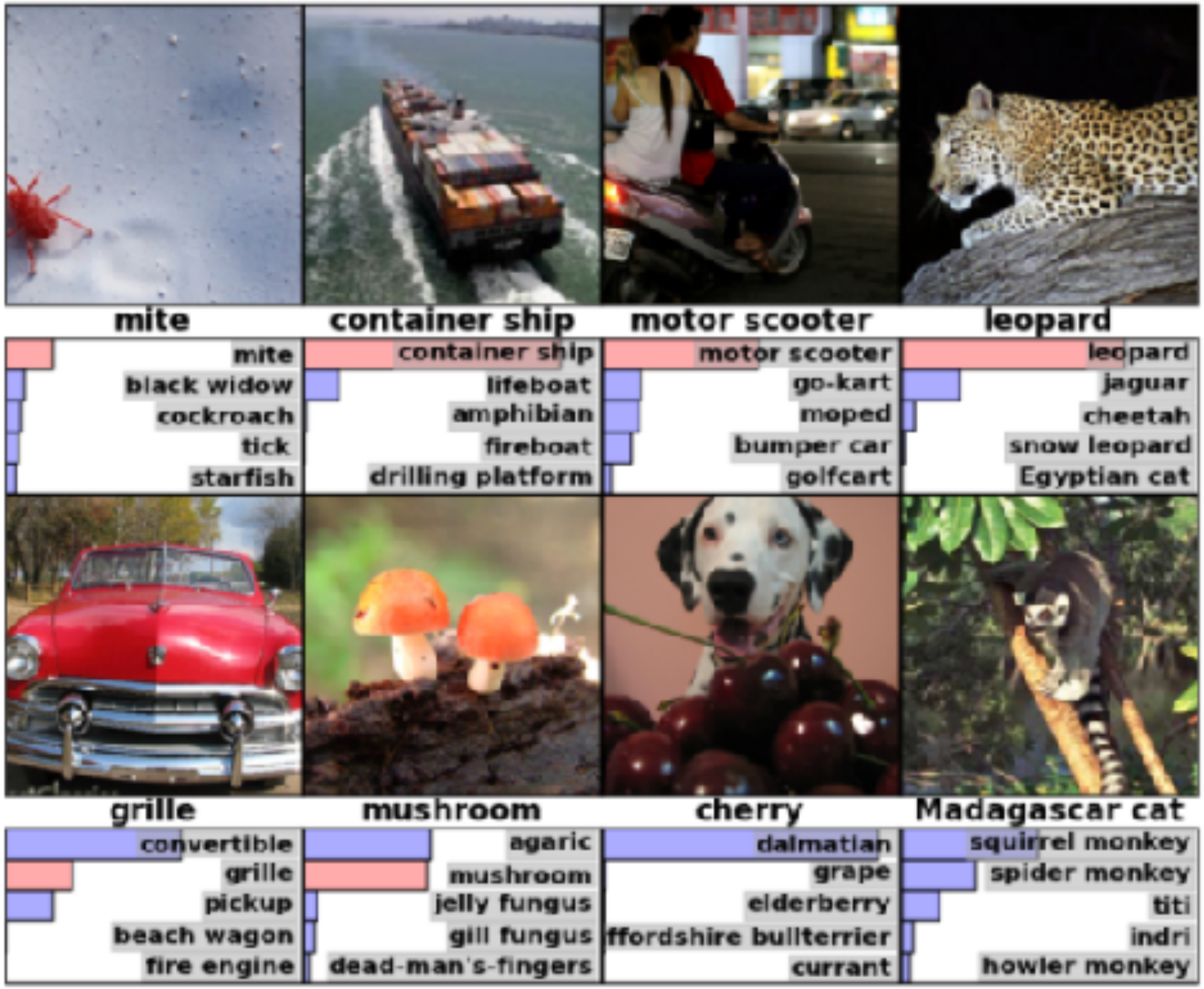
Figure 2. Residual learning: a building block.



Microsoft

Deep Convolutional Neural Network (DCNN)

DCNN for image classification



But it sometime fails …

correctly classified

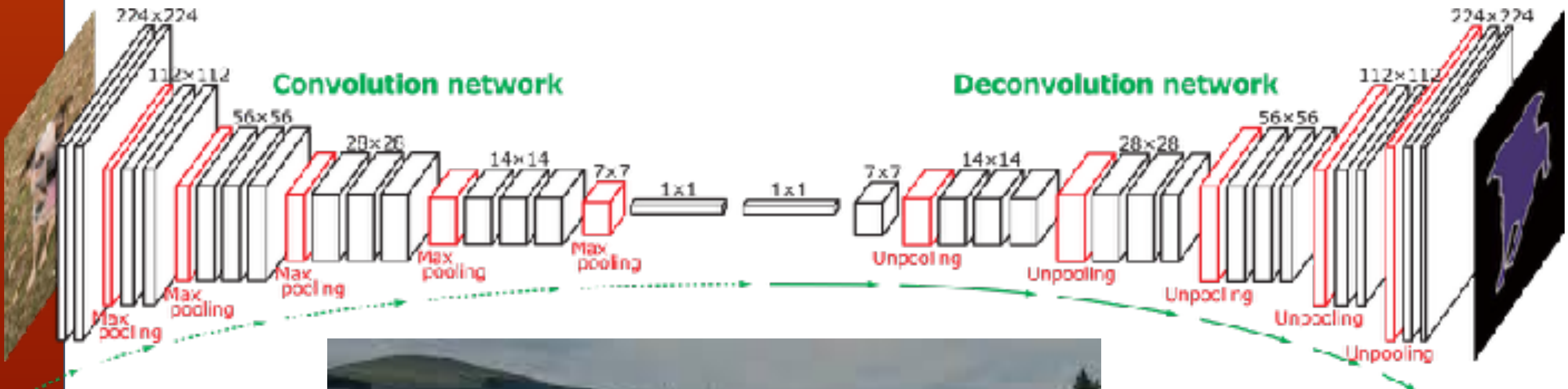
classified as ostrich

Trial and error testing can not guarantee reliability

Szegedy, Zaremba, Sutskever, Bruna, Erhan, Goodfellow, Fergus

Deep Convolutional Neural Network (DCNN)

DCNN for image segmentation



Deep Convolutional Neural Network (DCNN)

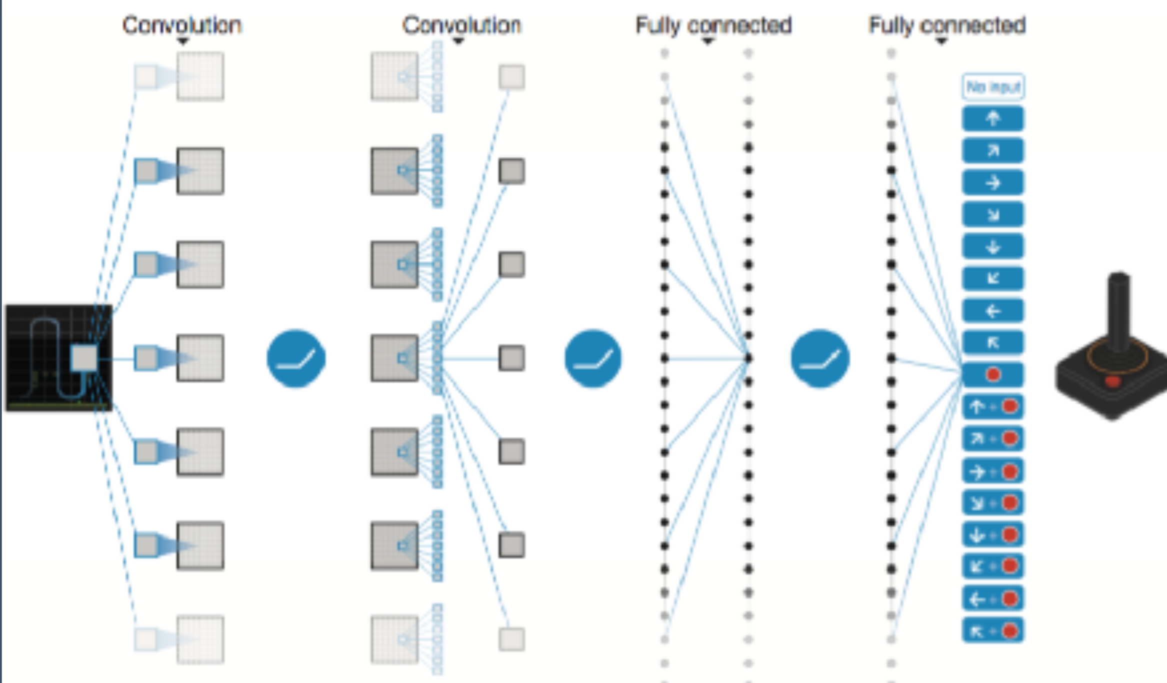
DCNN for image segmentation

http://youtube.com/watch?v=kMMbW96nMW8

Deep Q-Network (Google: published in nature)

DQN: Q-Learning + Deep Learning

- Add action to machine learning

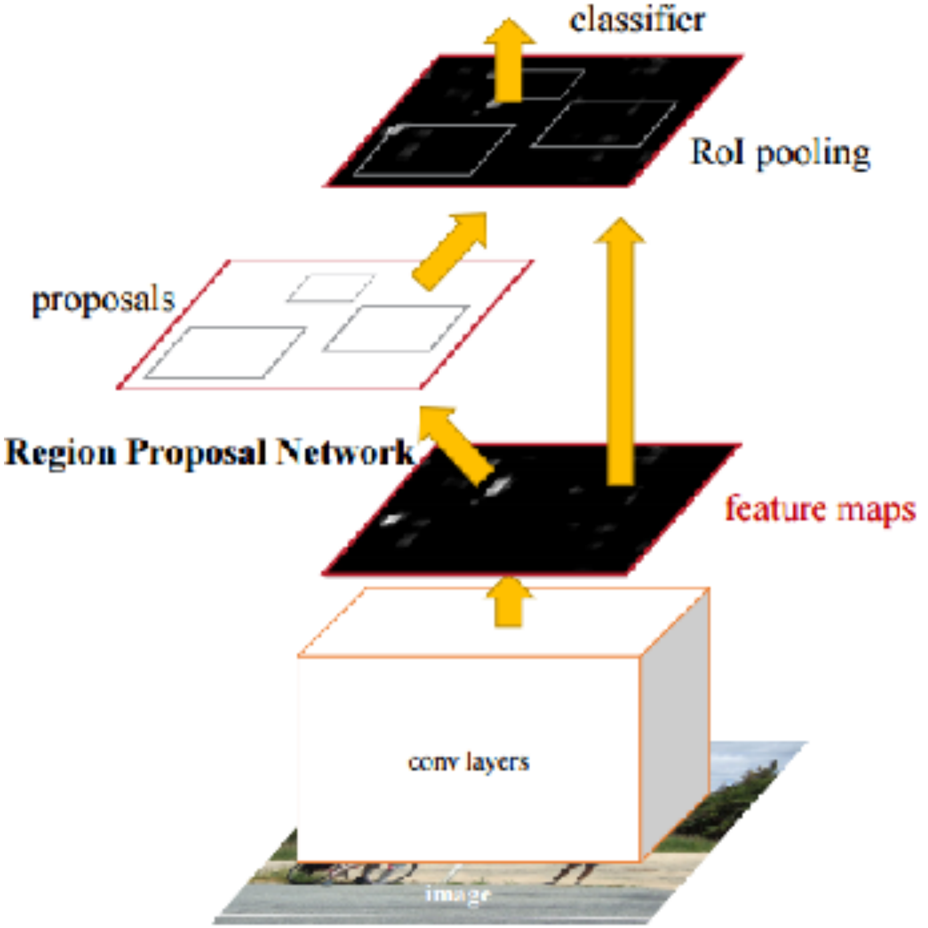


DQN playing enduro

http://youtube.com/watch?v=Ci8uvfVg_24

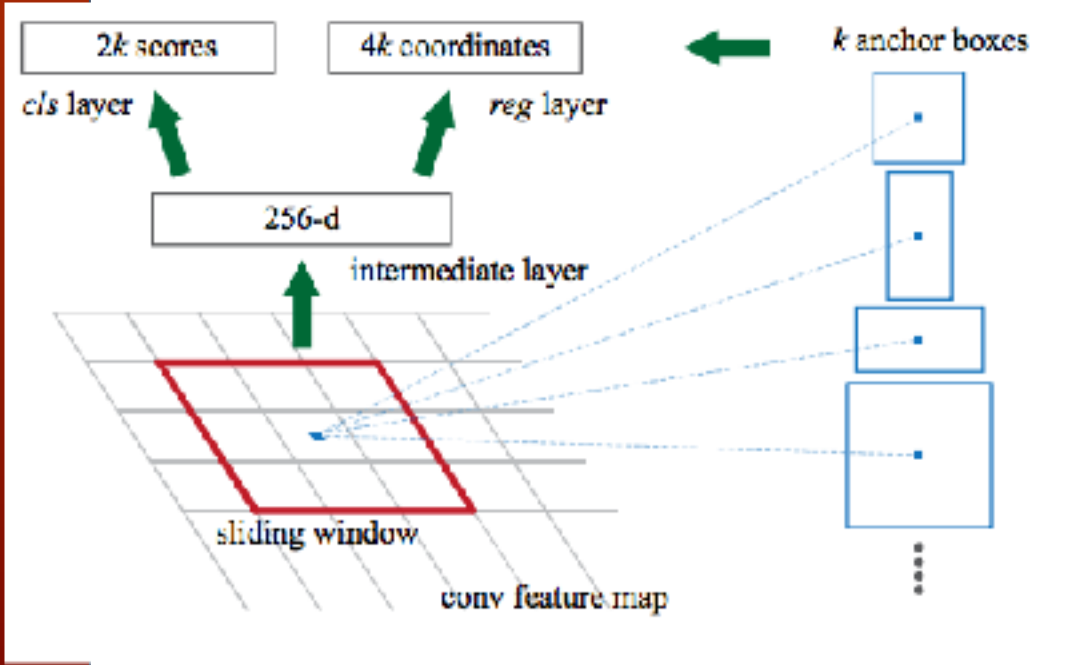
Object Localisation and Categorization (FasterRcnn)

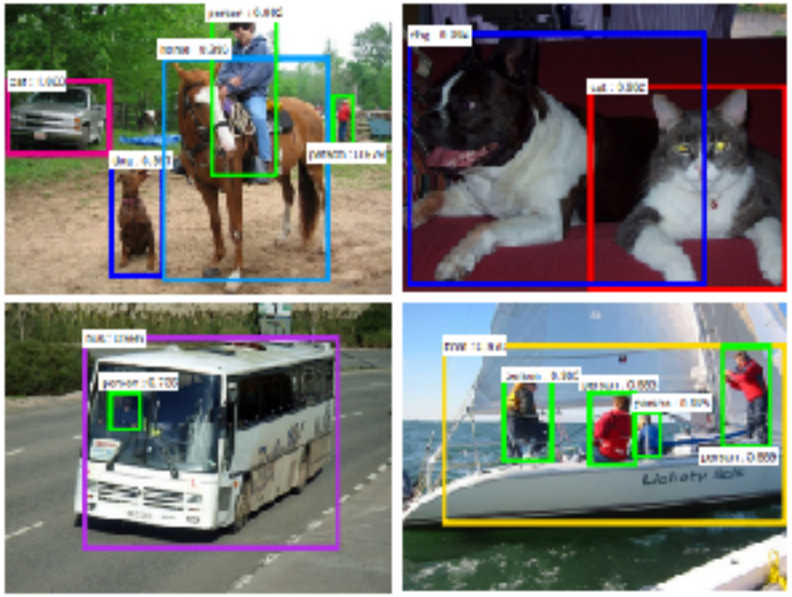
FasterRcnn:
Region Proposal Network + Classification Network



Object Localisation and Categorization (FasterRcnn)

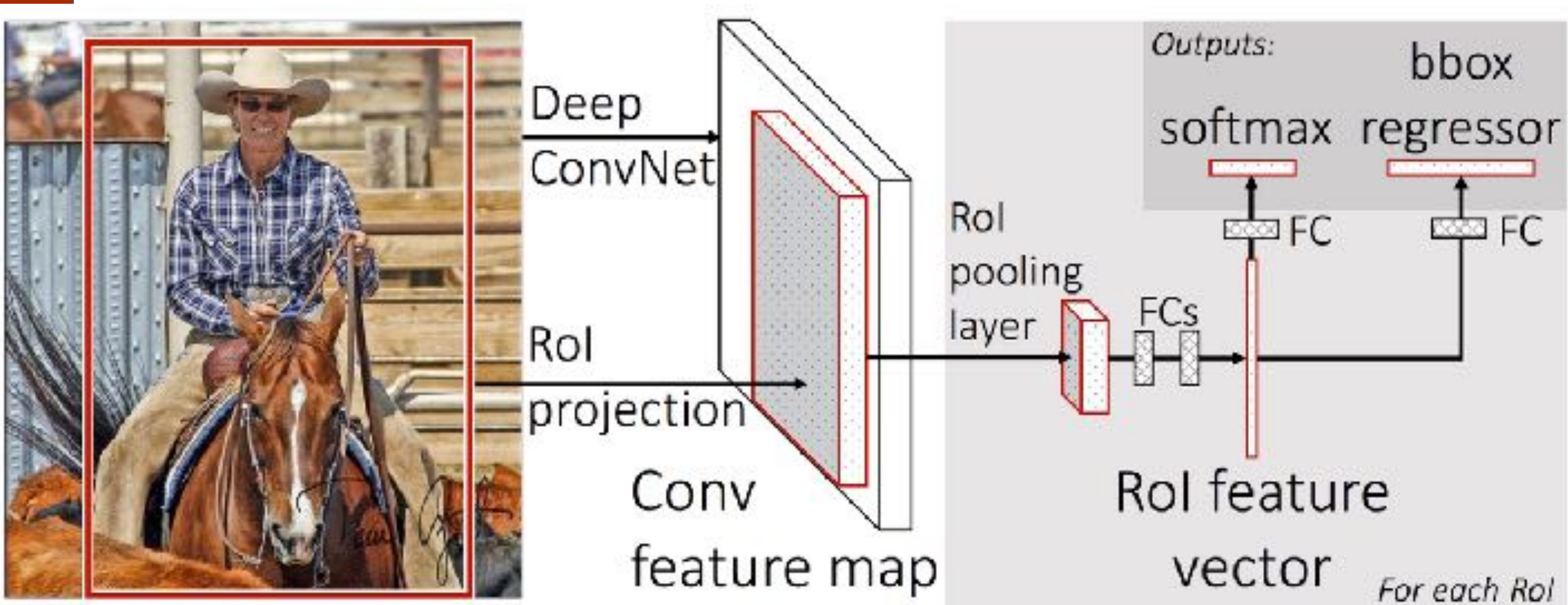
Region Proposal Network





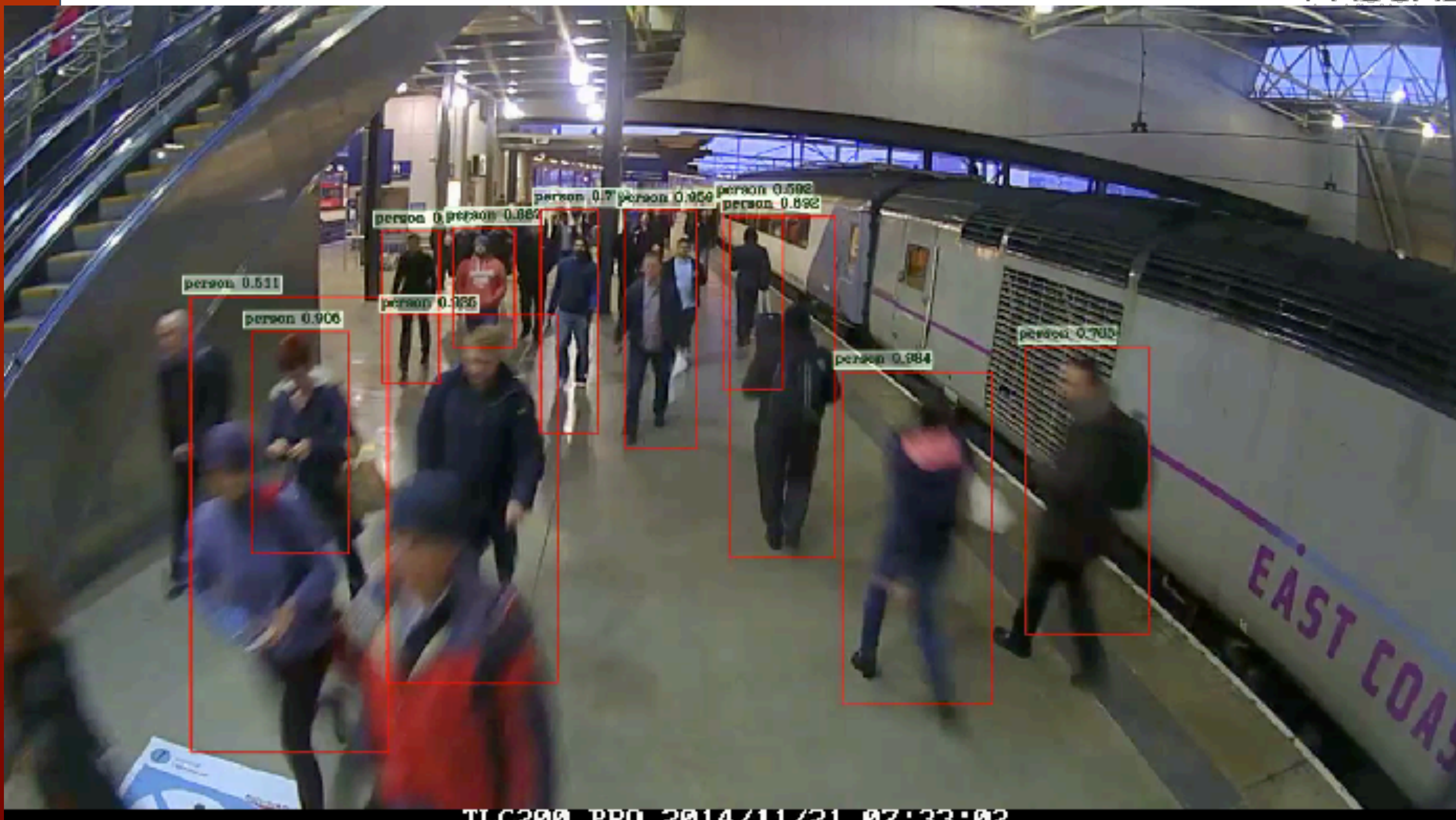
Object Localisation and Categorization (FasterRcnn)

RCNN



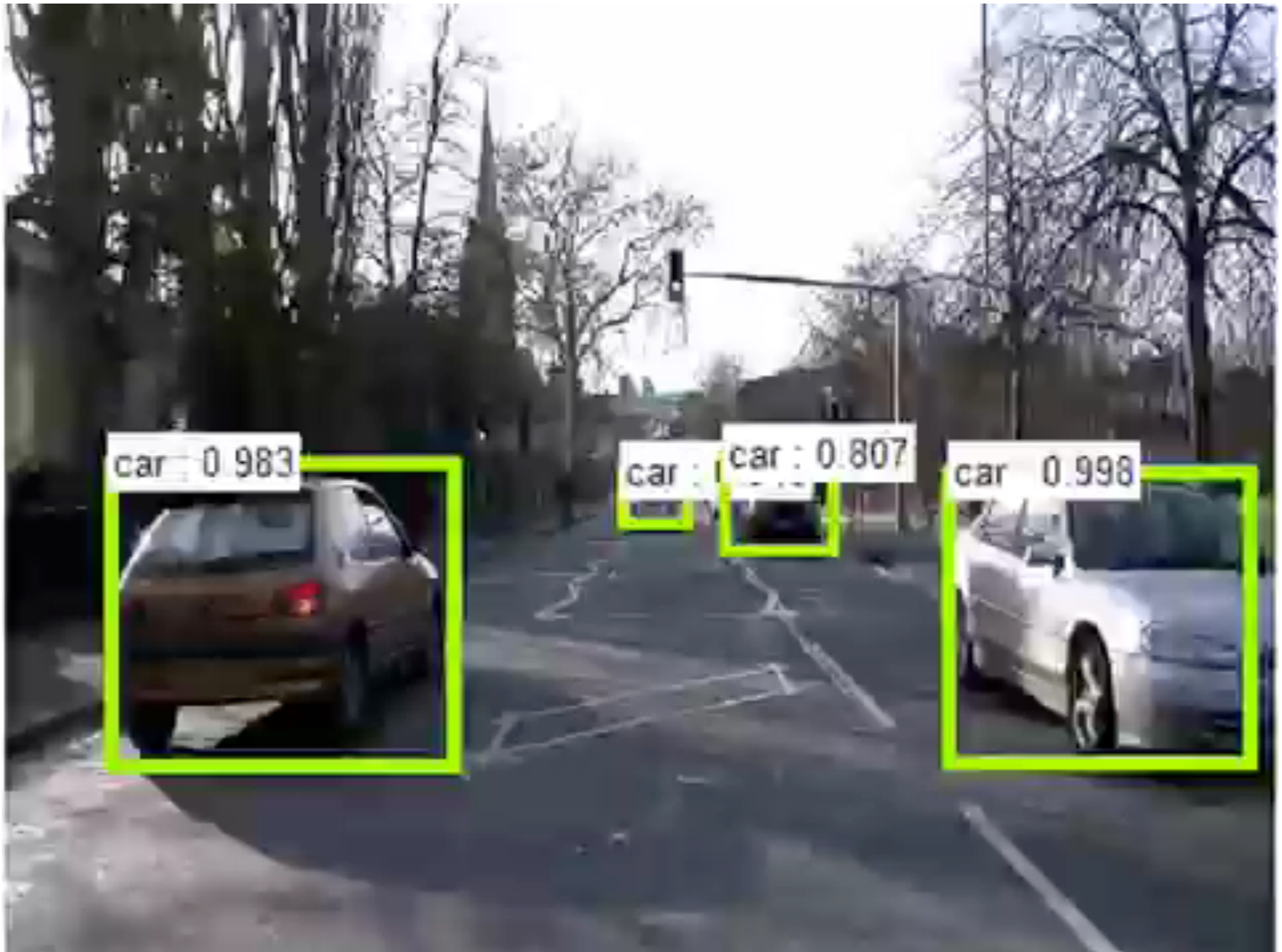
Object Localisation and Categorization (FasterRcnn)

Faster-Rcnn (realtime)



Object Localisation and Categorization (FasterRcnn)

Faster-Rcnn (realtime)



https://www.youtube.com/watch?v=WZmSMkK9VuA

Scene specialization

The intra-class variability issue (huge databases)

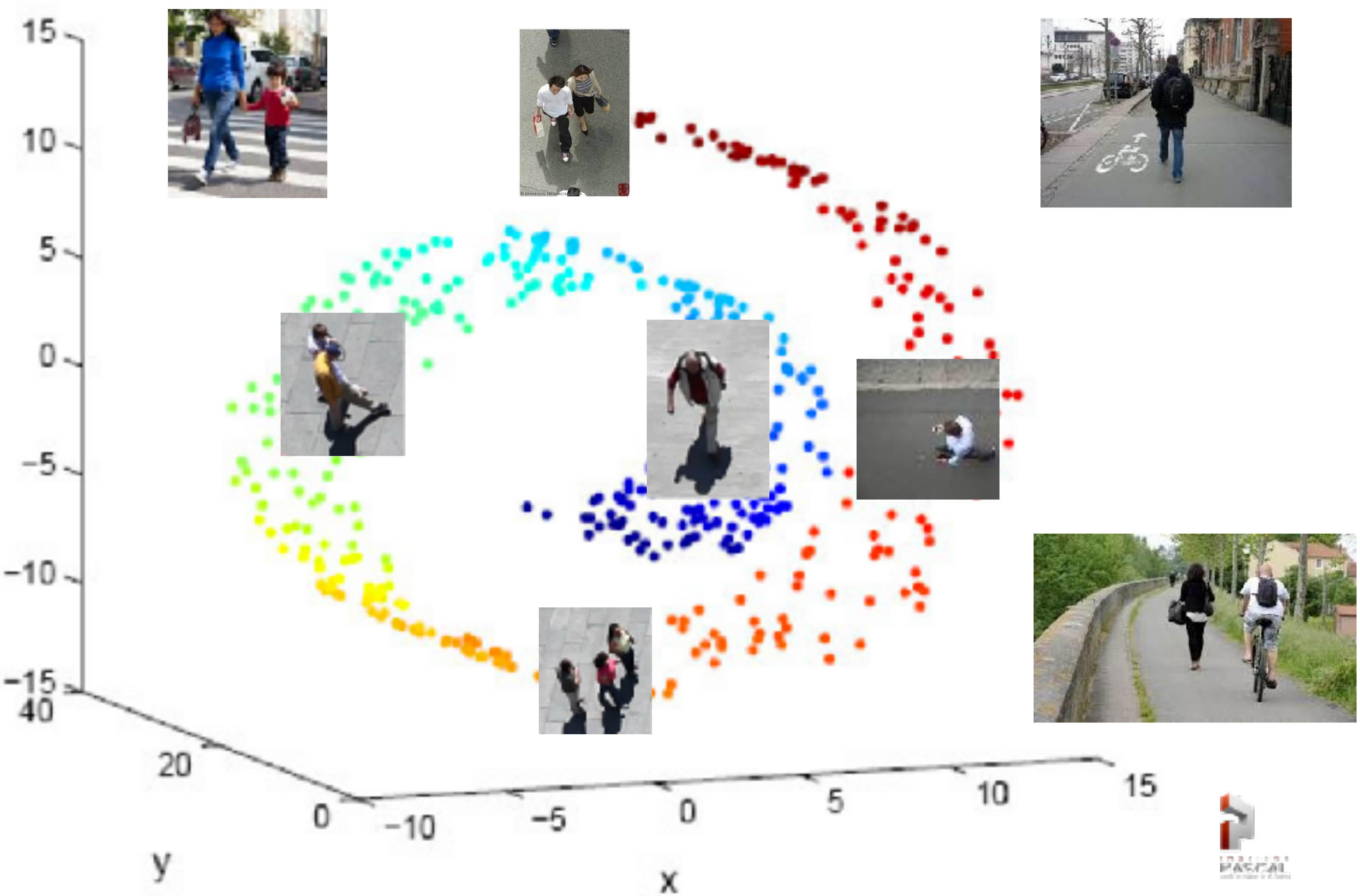
Scene specialization

The intra-class variability issue (static camera)

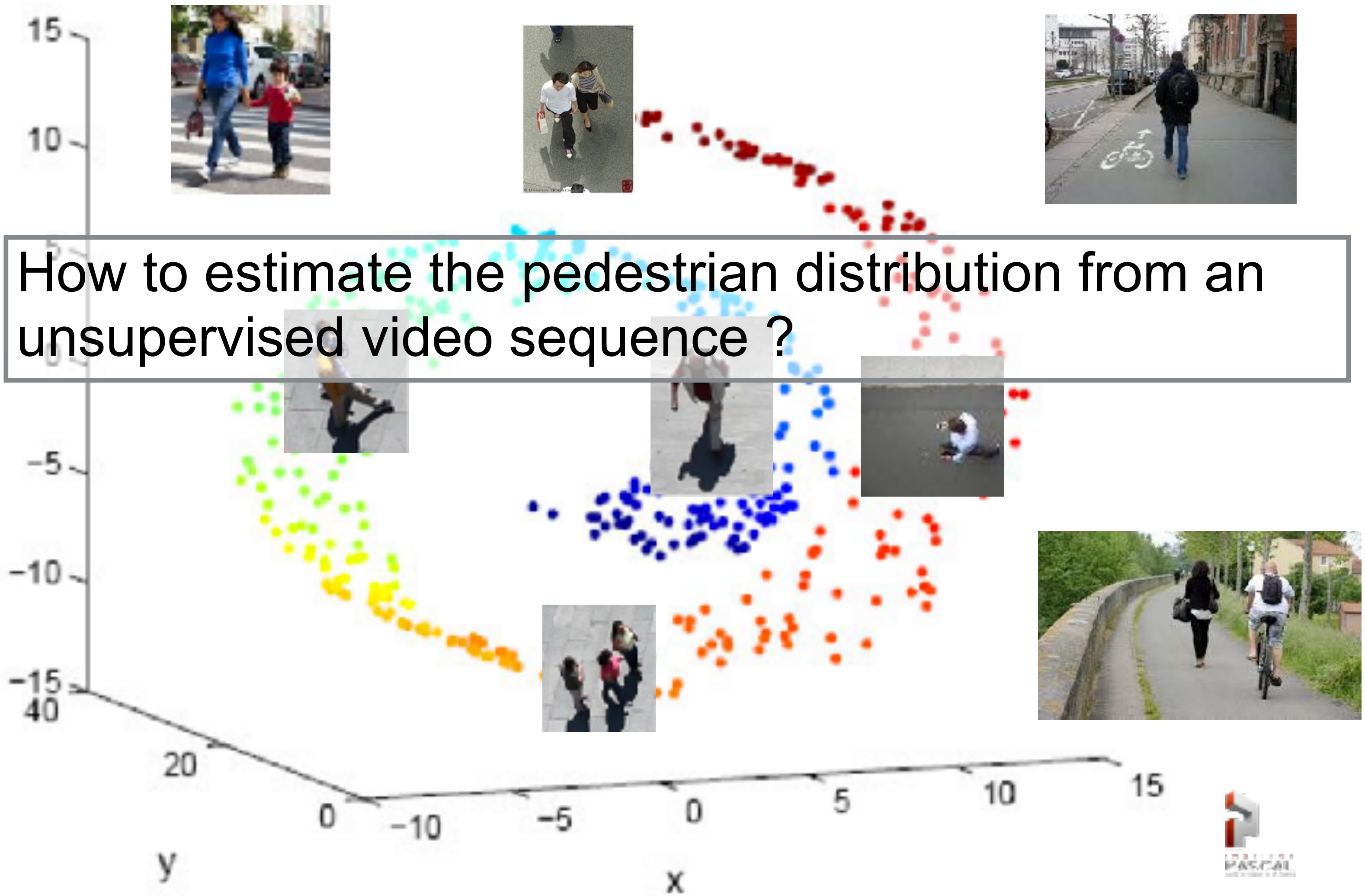
but several parameters are scene dependent (camera pose and view angle, trajectory)

Scene specialization

All the objects of a specific scene belong to a manifold of a large feature space



Scene specialization



How to estimate the pedestrian distribution from an unsupervised video sequence ?

Scene specialization

Some notations

X: a state vector associated to the target object distribution
Z: the measure vector (target video sequence)

We have to estimate:

$$p(\mathbf{X}|\mathbf{Z})$$

Scene specialization

The solution:

Approximate the probability distribution by a set of samples

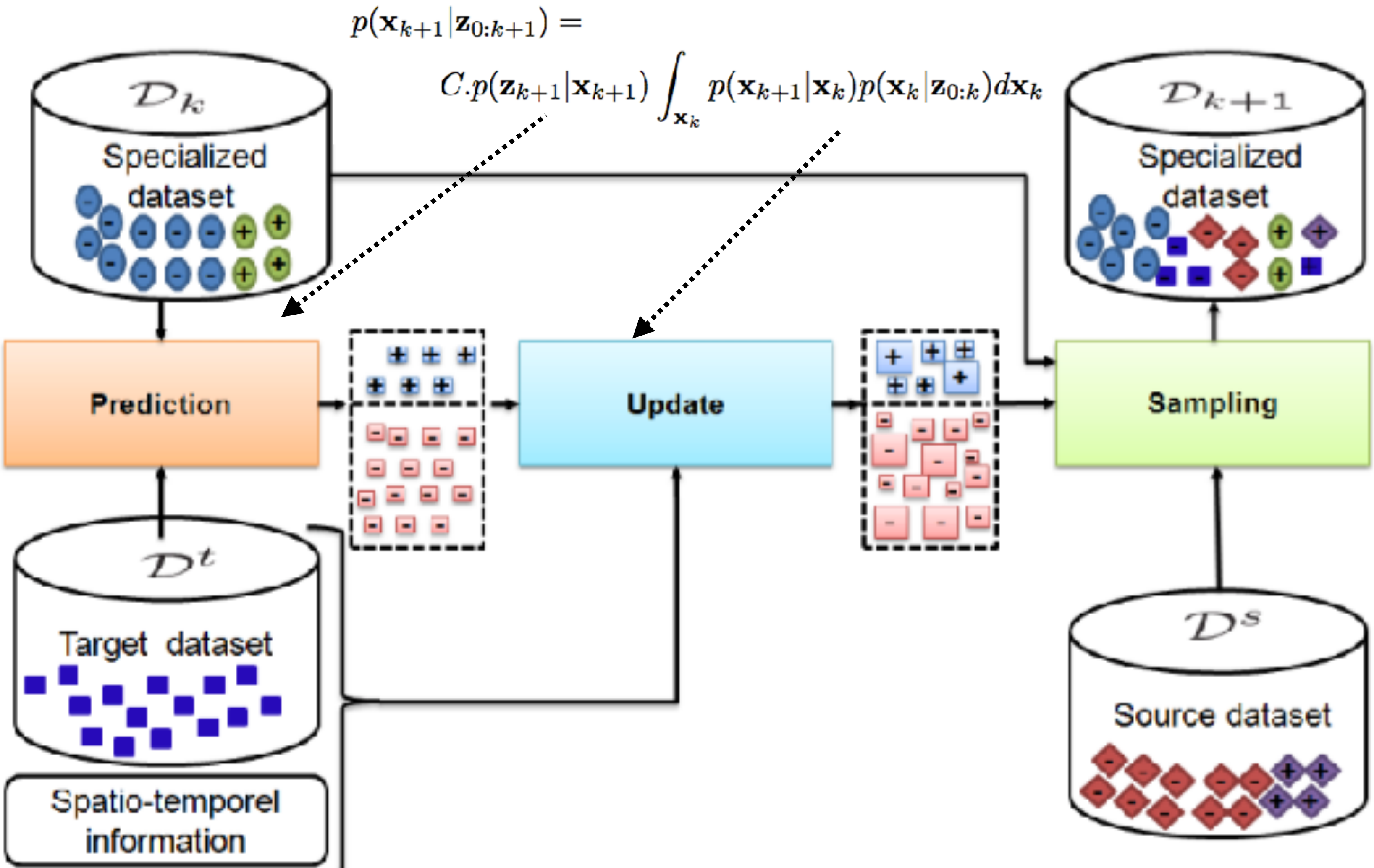
$$p(\mathbf{x}_k|\mathbf{z}_k) \approx \{\mathbf{x}_k^{(n)}\}_{n=1}^{N_k}$$

with a sequential Bayesian filter:

$$p(\mathbf{x}_{k+1}|\mathbf{z}_{0:k+1}) = C.p(\mathbf{z}_{k+1}|\mathbf{x}_{k+1}) \int_{\mathbf{x}_k} p(\mathbf{x}_{k+1}|\mathbf{x}_k) p(\mathbf{x}_k|\mathbf{z}_{0:k}) d\mathbf{x}_k$$

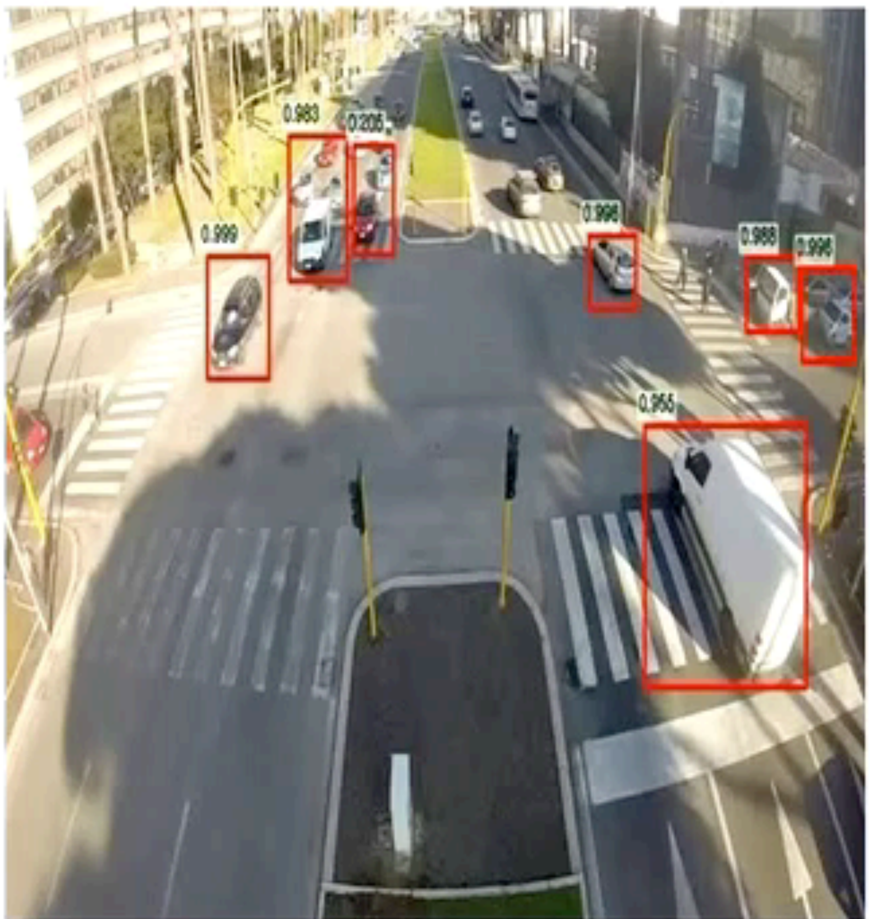
Scene specialization

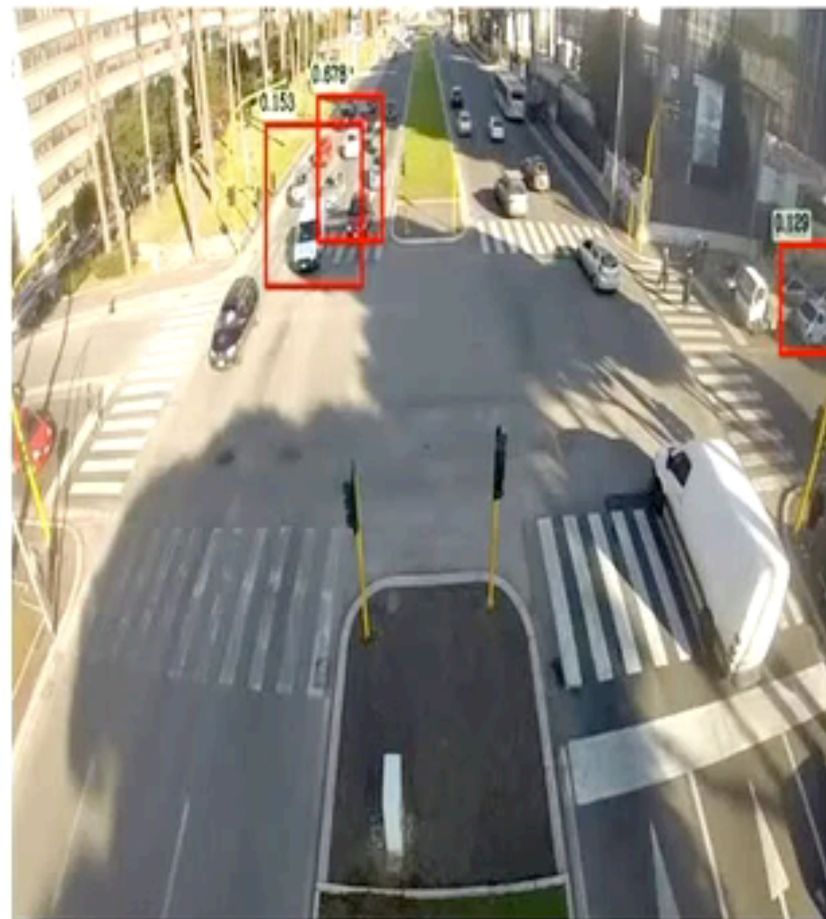
Approximate the probability distribution by a set of samples with a sequential Bayesian filter: (PhD H. Maamatou)



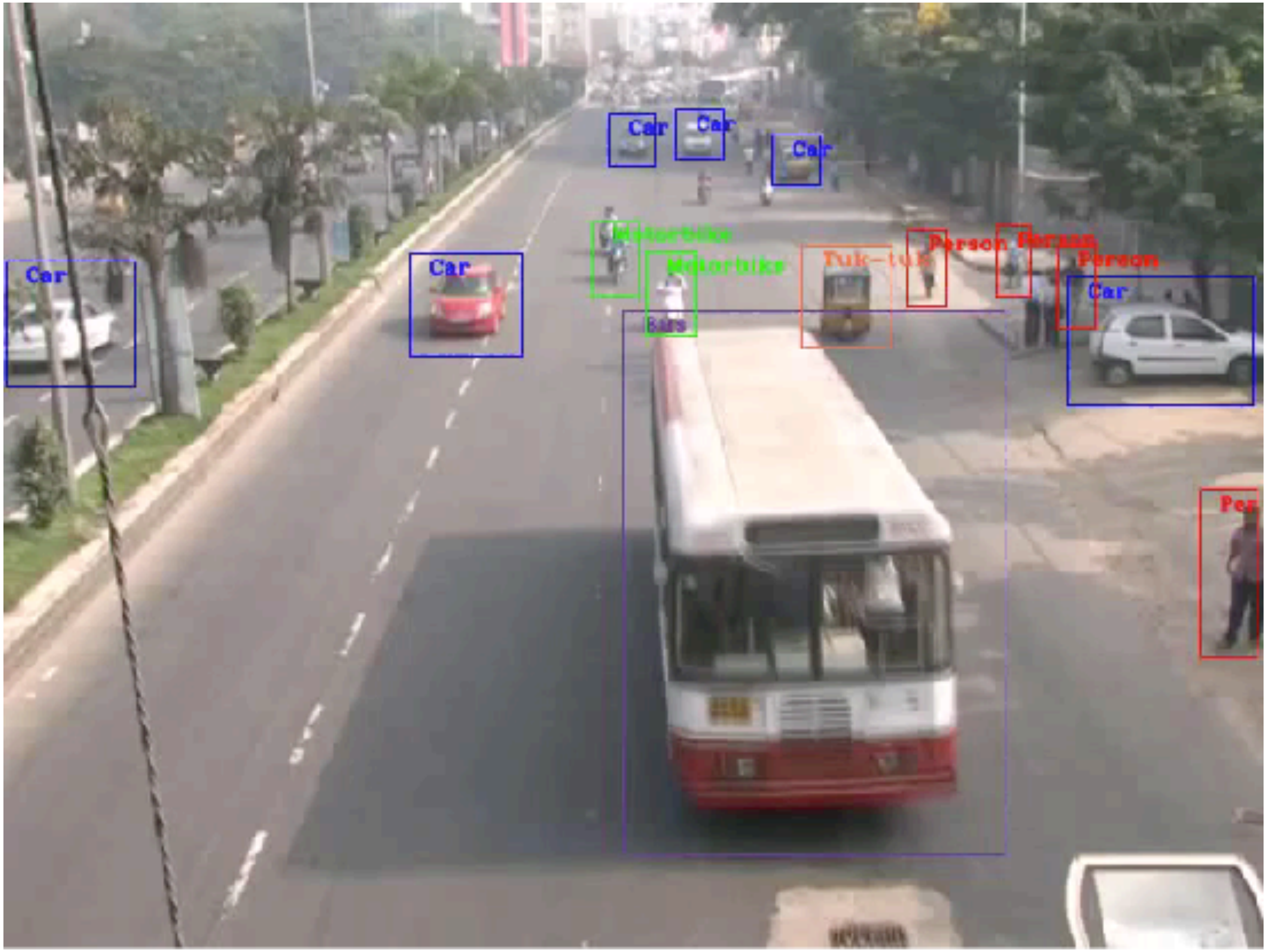
Scene specialization

Specialized **Generic**

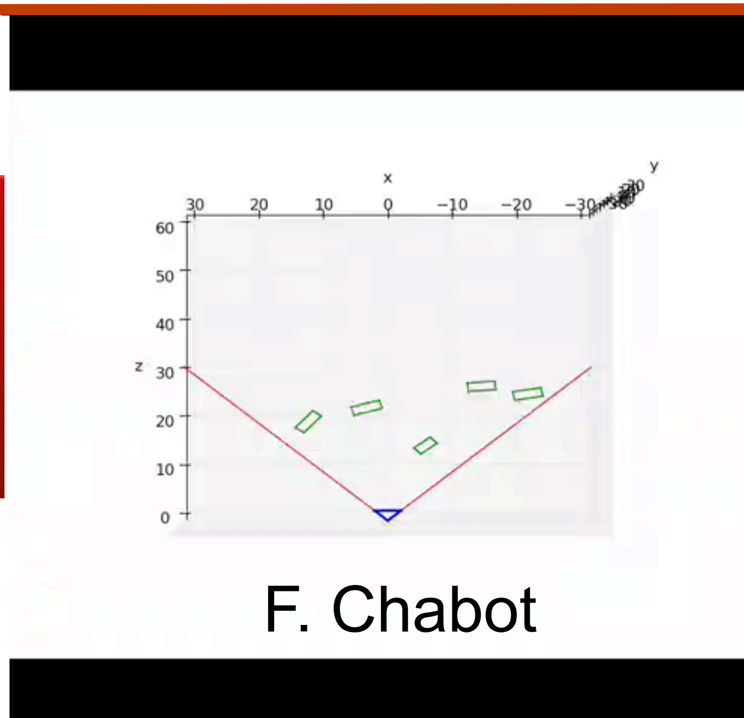




Scene Specialization

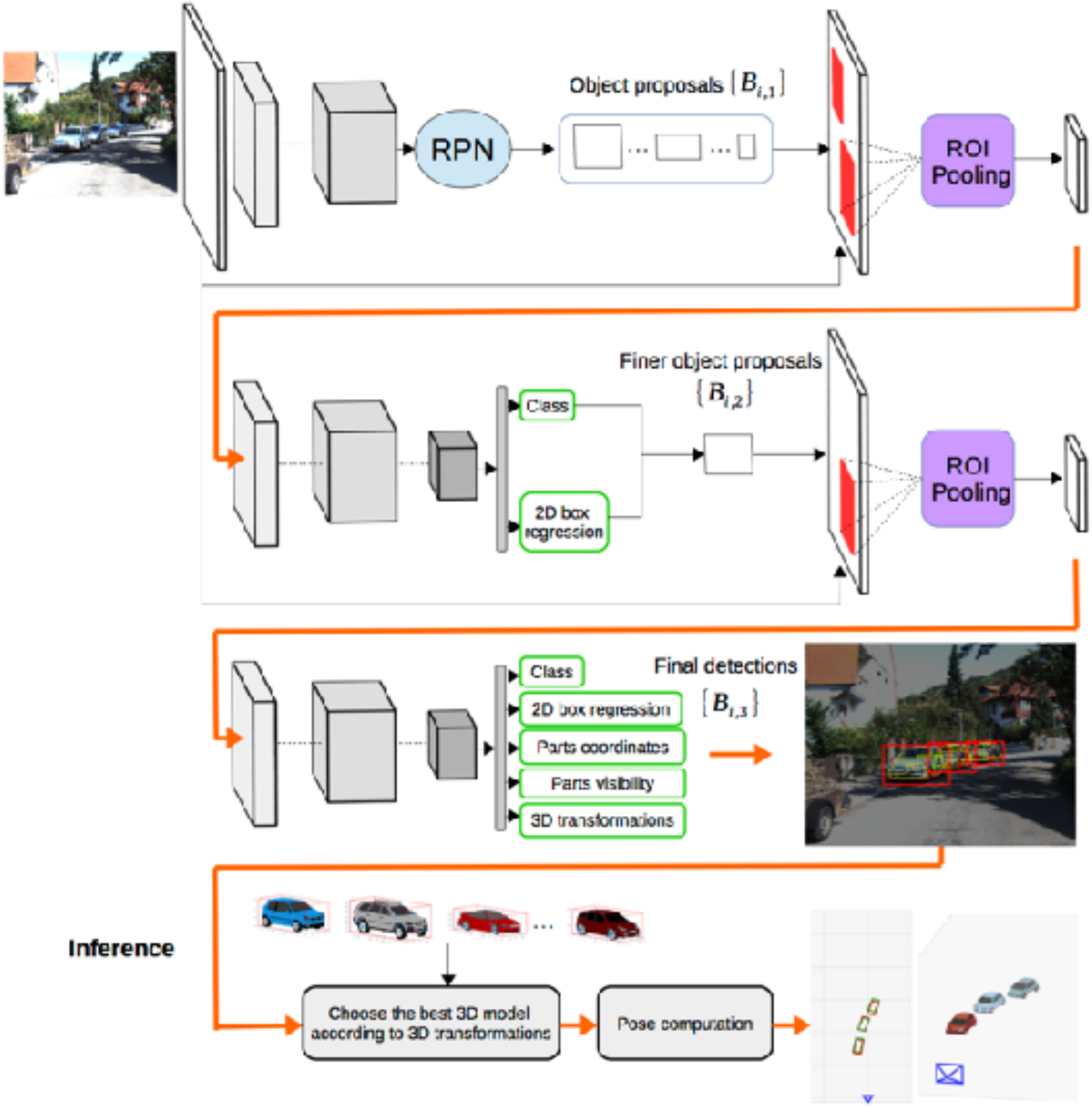


Deep Learning for 3D vehicle understanding from monocular images: toward many-task networks



Deep Learning for 3D vehicle understanding from monocular images: toward many-task networks

System Outputs



Object proposals $\{B_{i,1}\}$
RPN
ROI Pooling
Finer object proposals $\{B_{i,2}\}$
Class
2D box regression
ROI Pooling
Final detections $\{B_{i,3}\}$
Class
2D box regression
Parts coordinates
Parts visibility
3D transformations
Inference
Choose the best 3D model according to 3D transformations
Pose computation

INSTITUT PASCAL

Loss functions

Detection loss

ROI localisation loss

Parts Loss

Visibility Loss

3D transformation loss

Experiments (Kitti Dataset)

Detection and orientation

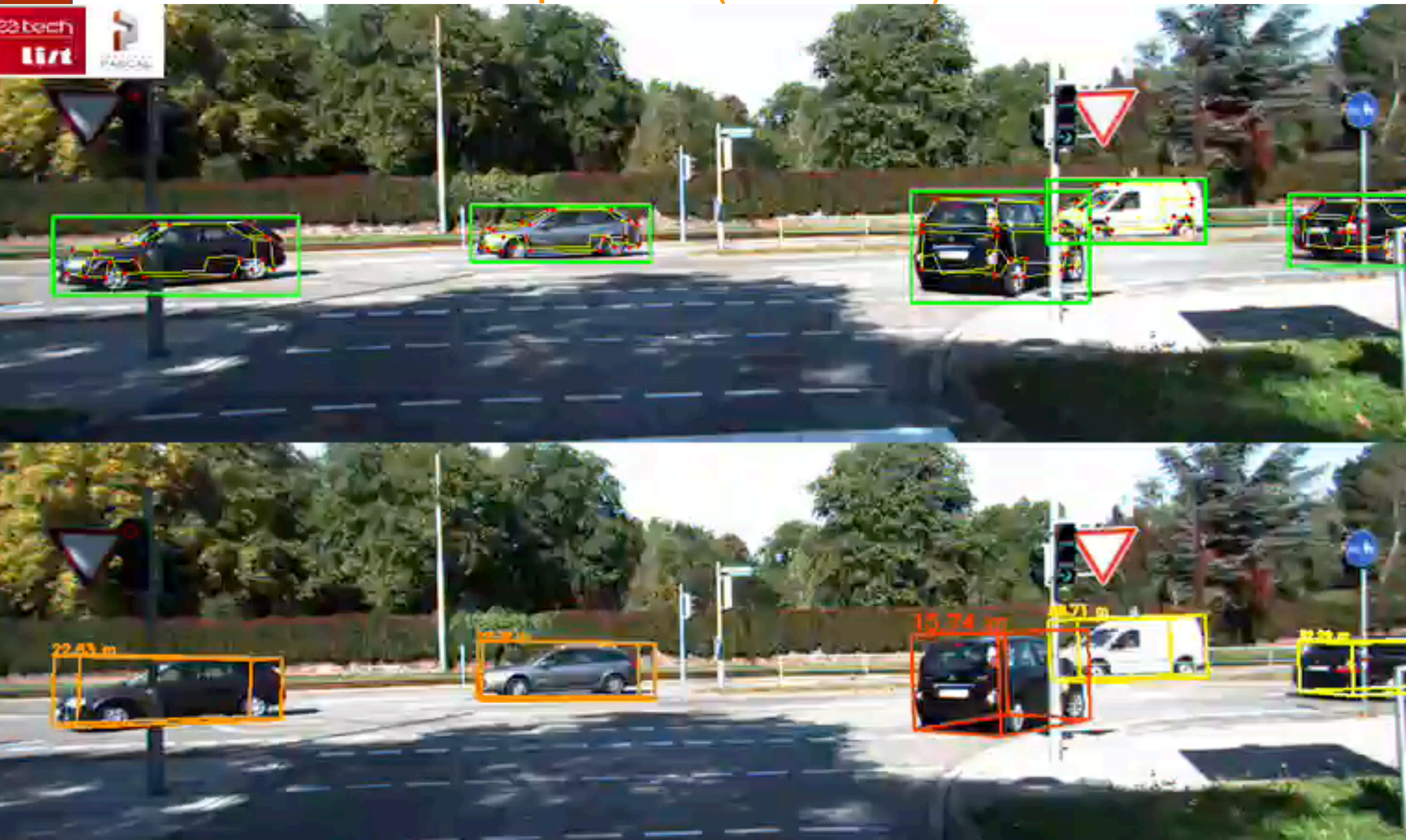
			val1					
			AP			AOS		
Method	Type	Time	Easy	Moderate	Hard	Easy	Moderate	Hard
3DVP [31]	Mono	40 s	80.48	68.05	57.20	78.99	65.73	54.67
Faster-RCNN [27]	Mono	2 s	82.91	77.83	66.25	-	-	-
SubCNN [32]	Mono	2 s	95.77	86.64	74.07	94.55	85.03	72.2
Ours nms = 0.4	Mono	0.7 s	97.05	88.94	78.25	96.90	88.68	77.83
Ours nms = 0.5	Mono	0.7 s	96.98	89.58	79.77	96.83	89.31	79.31
Ours w vis	Mono	0.7 s	**97.90**	**91.01**	**83.14**	**97.60**	**90.66**	**82.66**

The KITTI Vision Benchmark Suite
A project of Karlsruhe Institute of Technology and Toyota Technological Institute at Chicago

AP: mean average precision
AOS: average orientation similarity

Deep Learning for 3D vehicle understanding from monocular images

Experiments (Kitti Dataset)



Technical aspects

open source libraries for deep learning (all with GPU implementation)

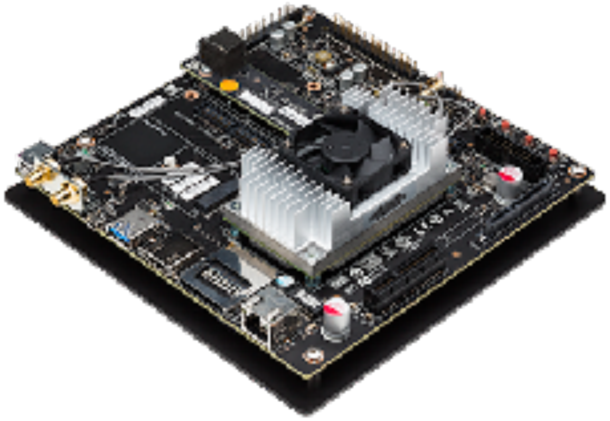
- Tensor Flow (Google, C++, Python)
- Caffe (Berkeley, C++, Python)
- Torch (« facebook », Lua)
- Theano (. , python)
- ...

Technical aspects

Hardwares for Deep Learning

- Learning needs GPU (NVIDIA)
-
- ...

Google Asic

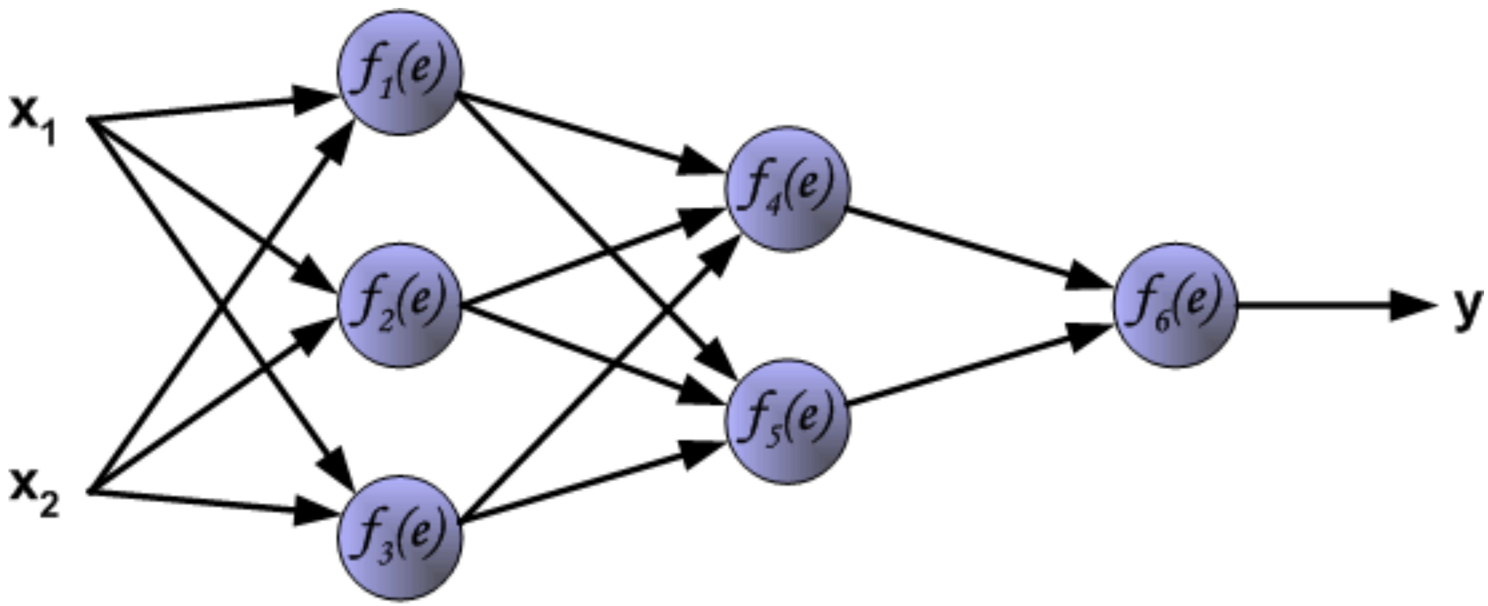


NVIDIA Jetson TX1

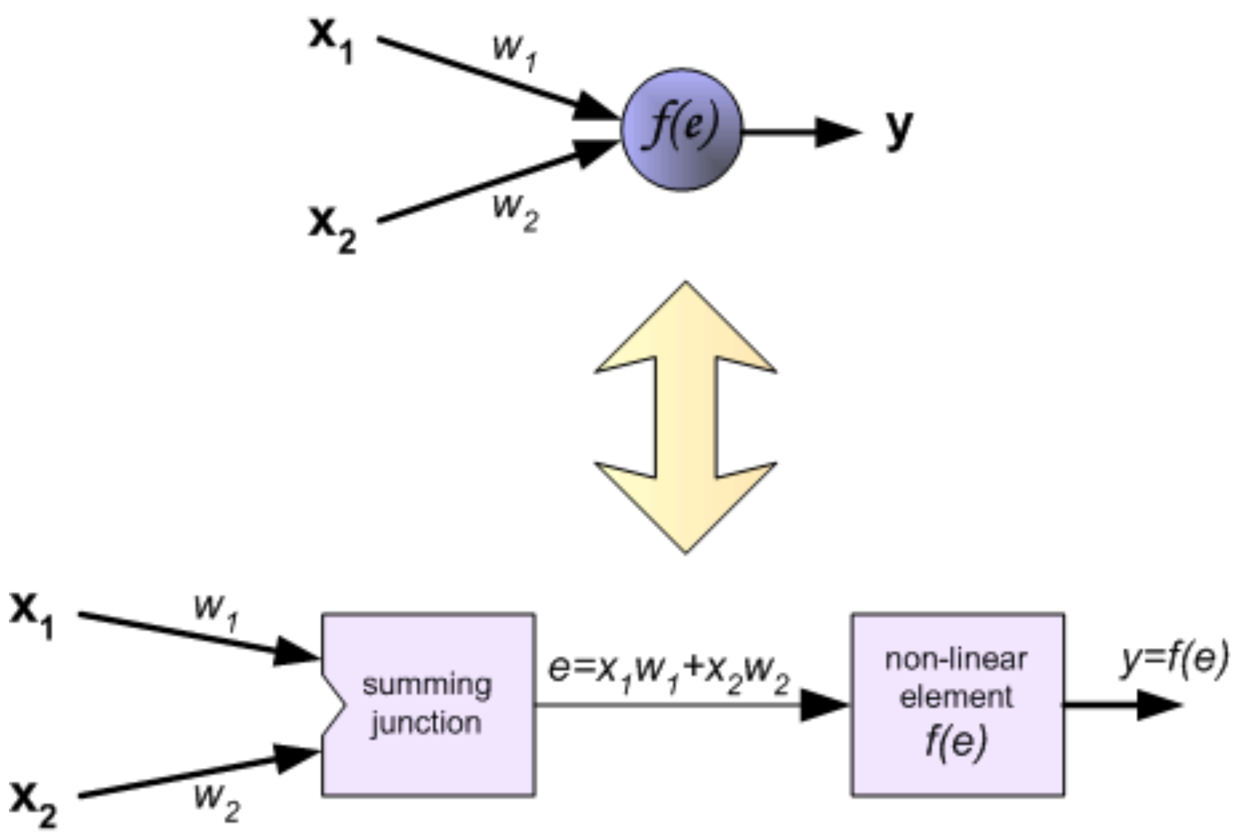
Conclusion

- Deep learning outperforms other approaches for detection and classification
- Hardware systems are specifically designed for DCNN (Nvidia, Google, Altera)
- How to prove the robustness of such method (Trial and error testing can not guarantee reliability)(real problem for Autonomous Driving Systems)?
- Databases are needed to learn DCNN (What about new sensors or multi sensors systems)

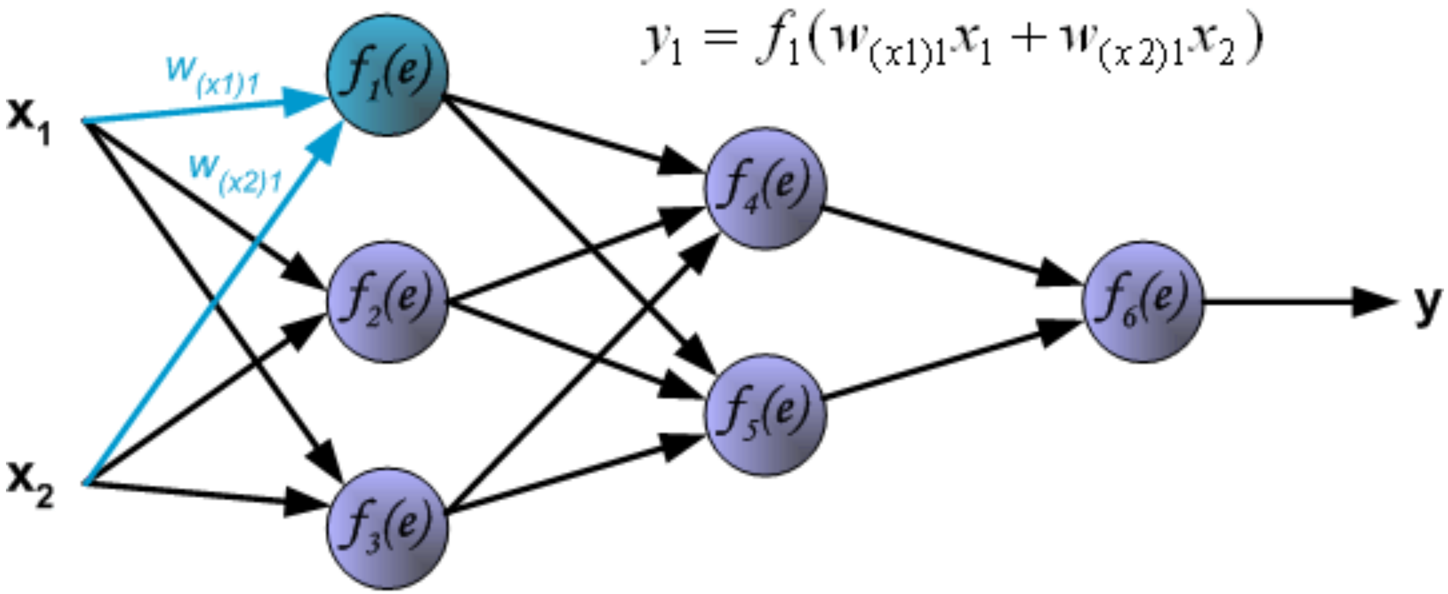
Backpropagation principle



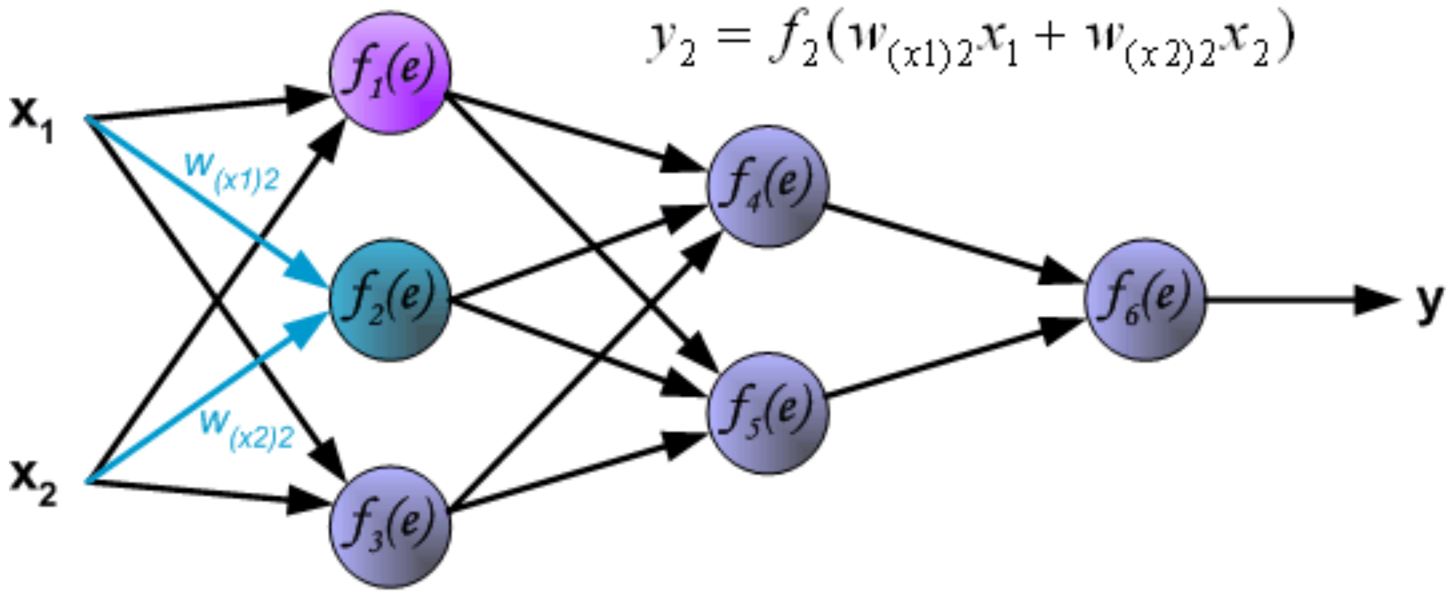
Backpropagation principle



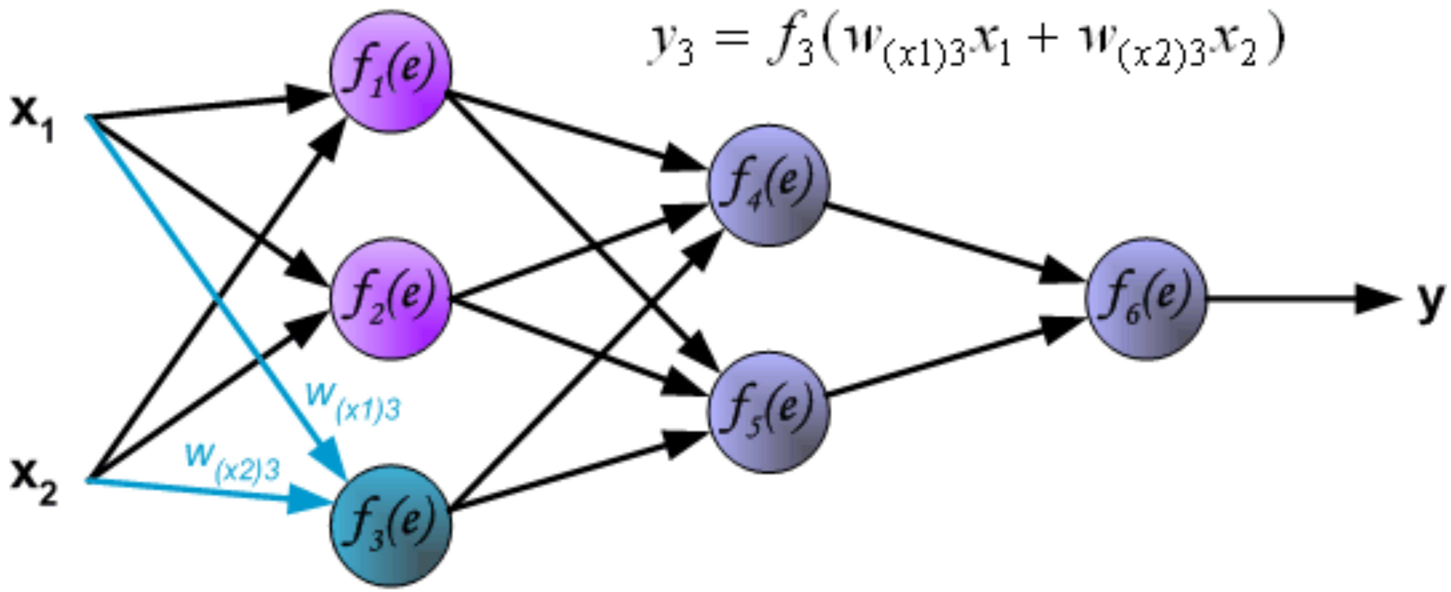
Backpropagation principle



Backpropagation principle

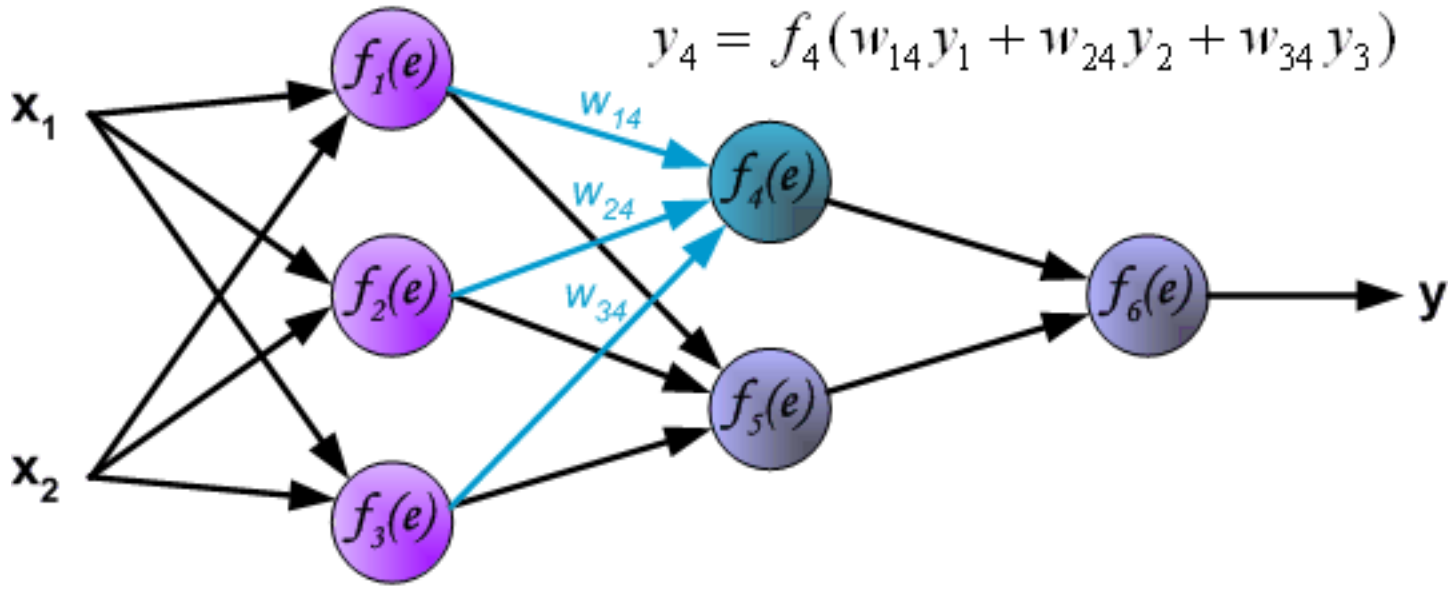


Backpropagation principle

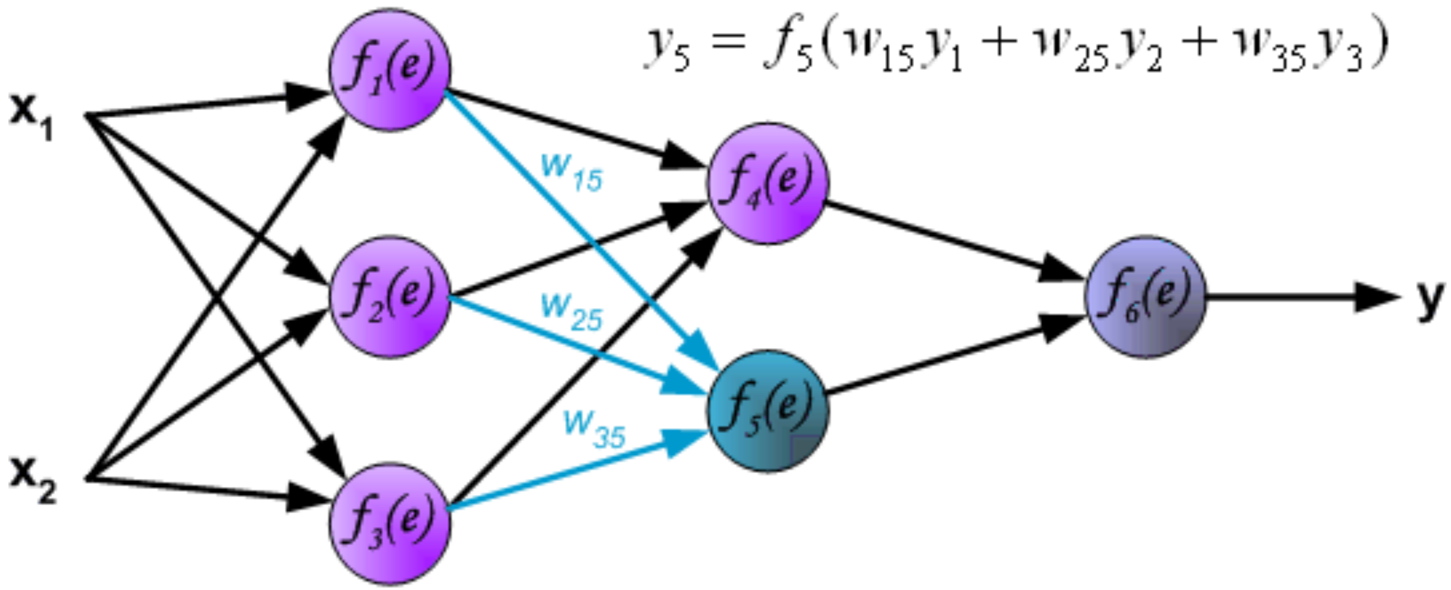


Backpropagation principle

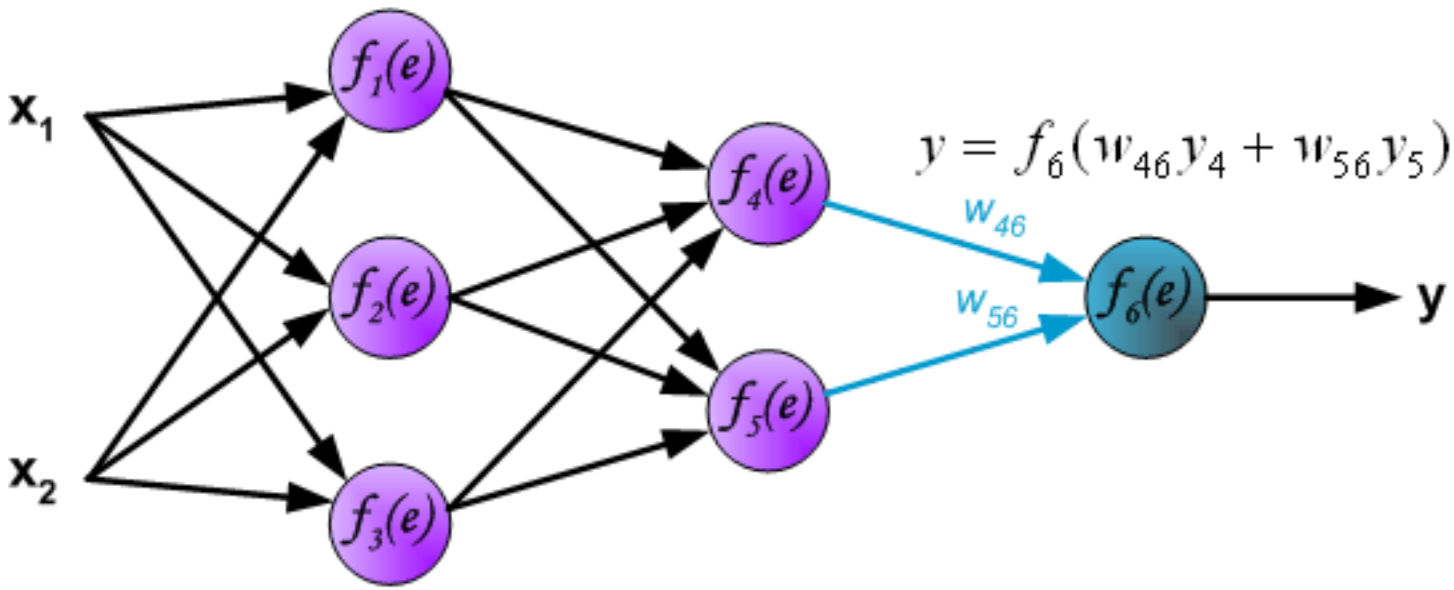
$$y_4 = f_4(w_{14}y_1 + w_{24}y_2 + w_{34}y_3)$$



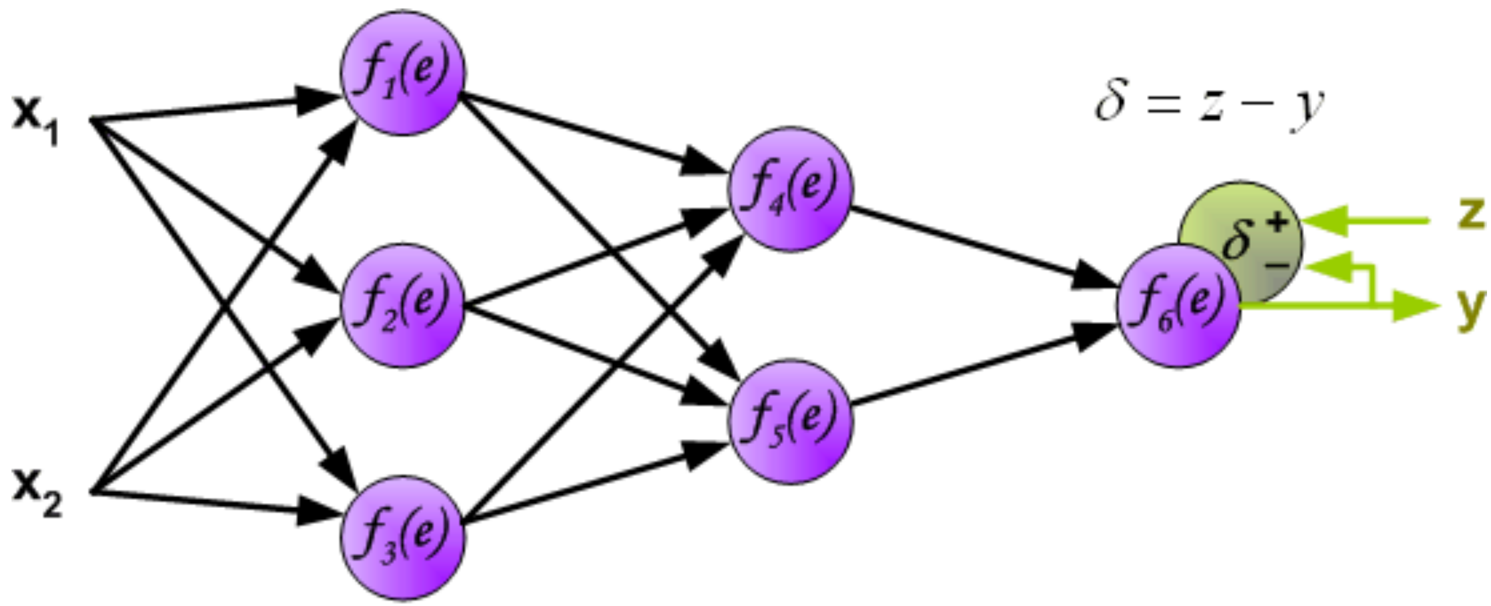
Backpropagation principle



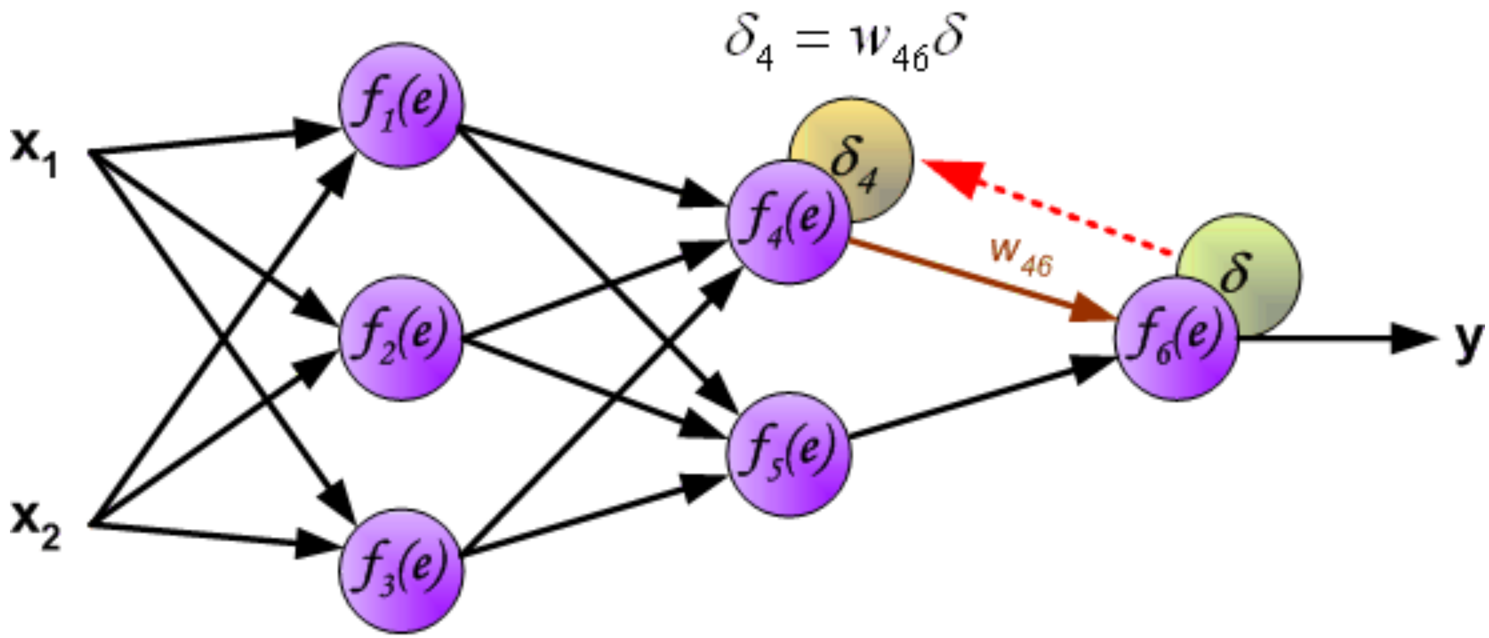
Backpropagation principle



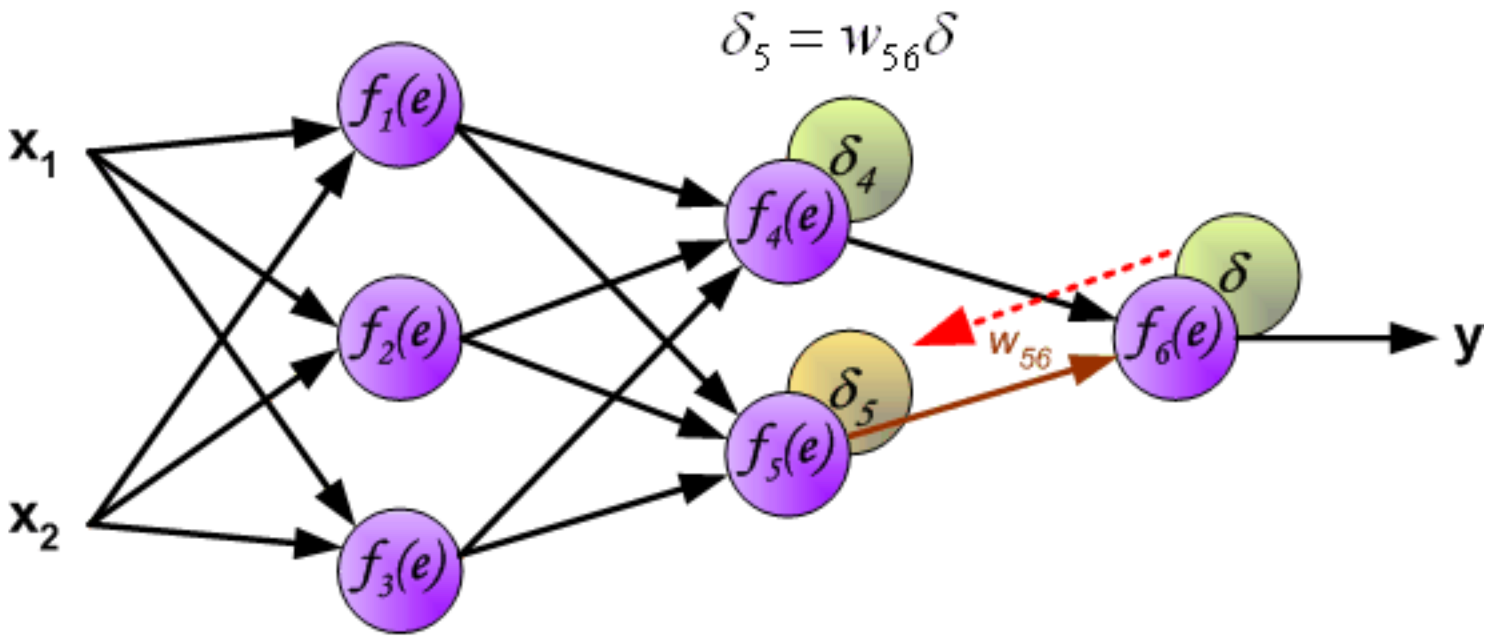
Backpropagation principle



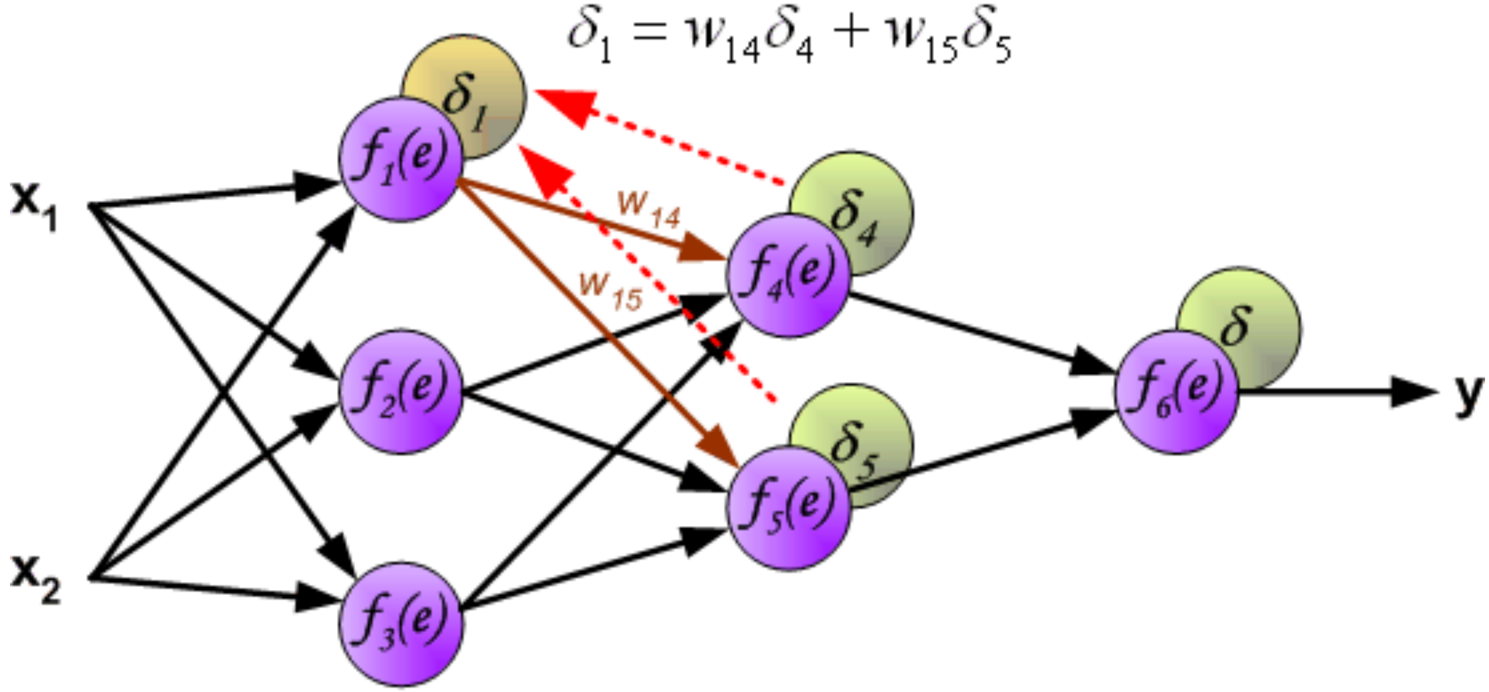
Backpropagation principle



Backpropagation principle

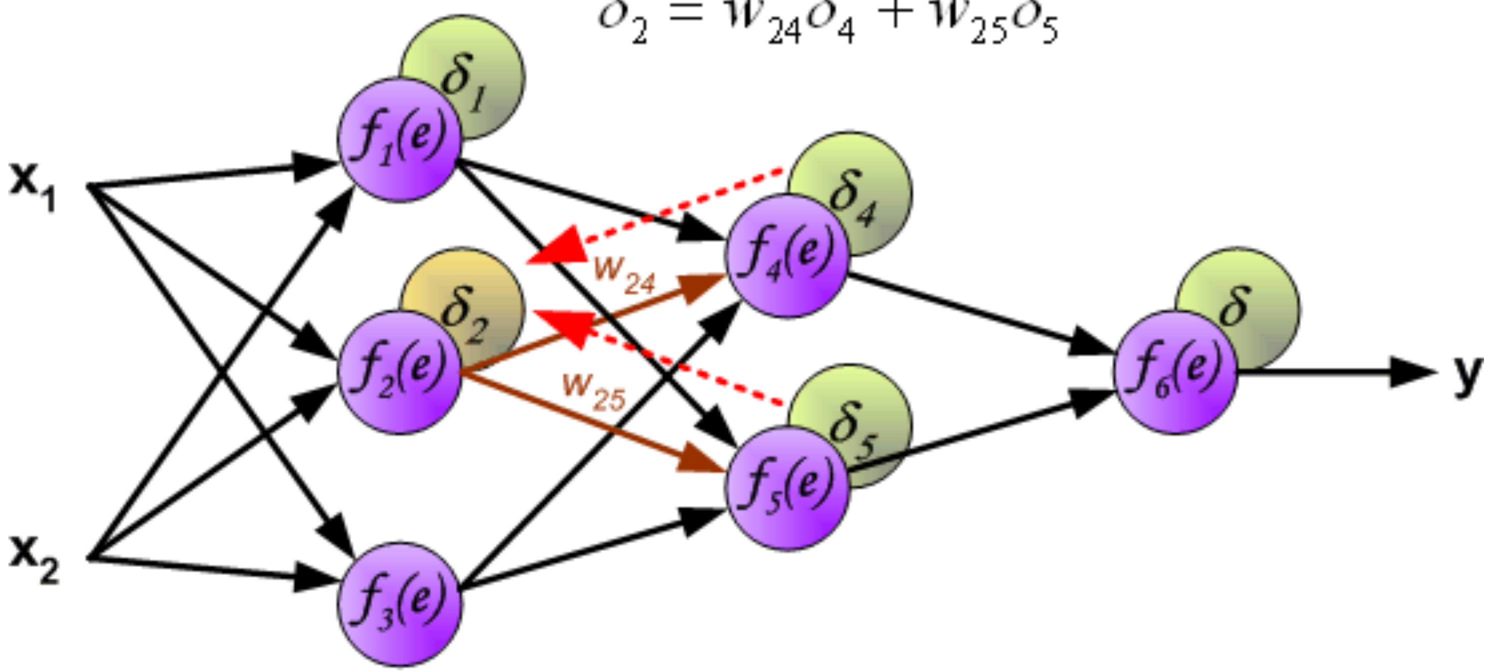


Backpropagation principle

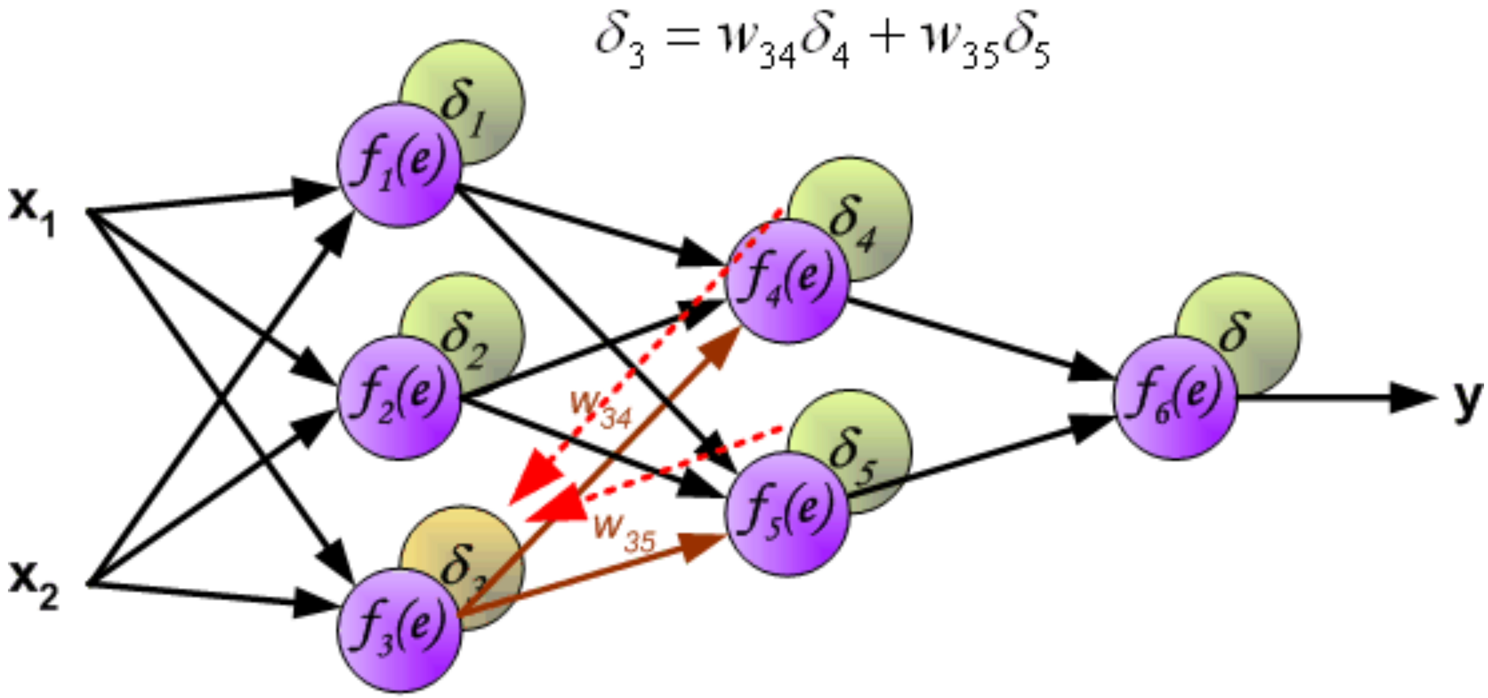


Backpropagation principle

$$\delta_2 = w_{24}\delta_4 + w_{25}\delta_5$$



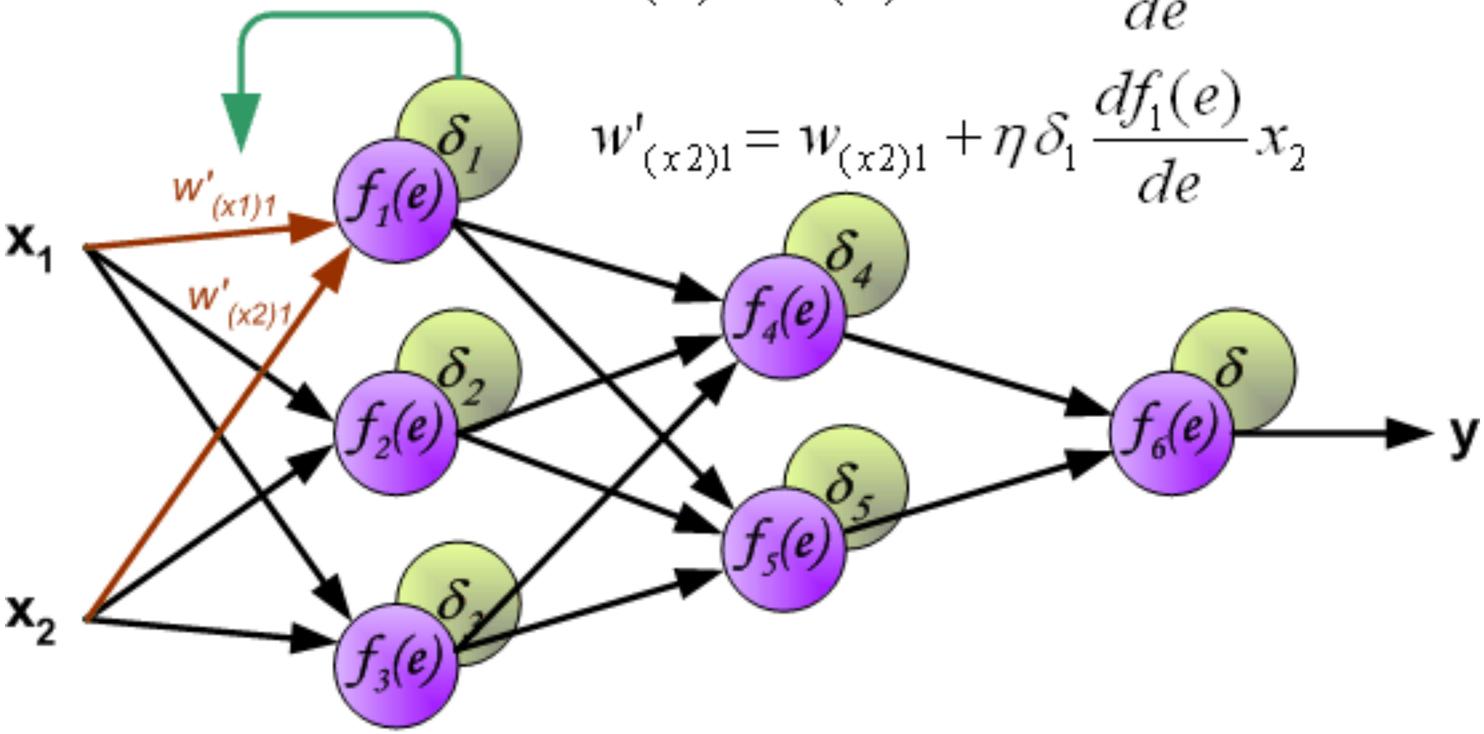
Backpropagation principle



Backpropagation principle

$$w'_{(x1)1} = w_{(x1)1} + \eta \delta_1 \frac{df_1(e)}{de} x_1$$

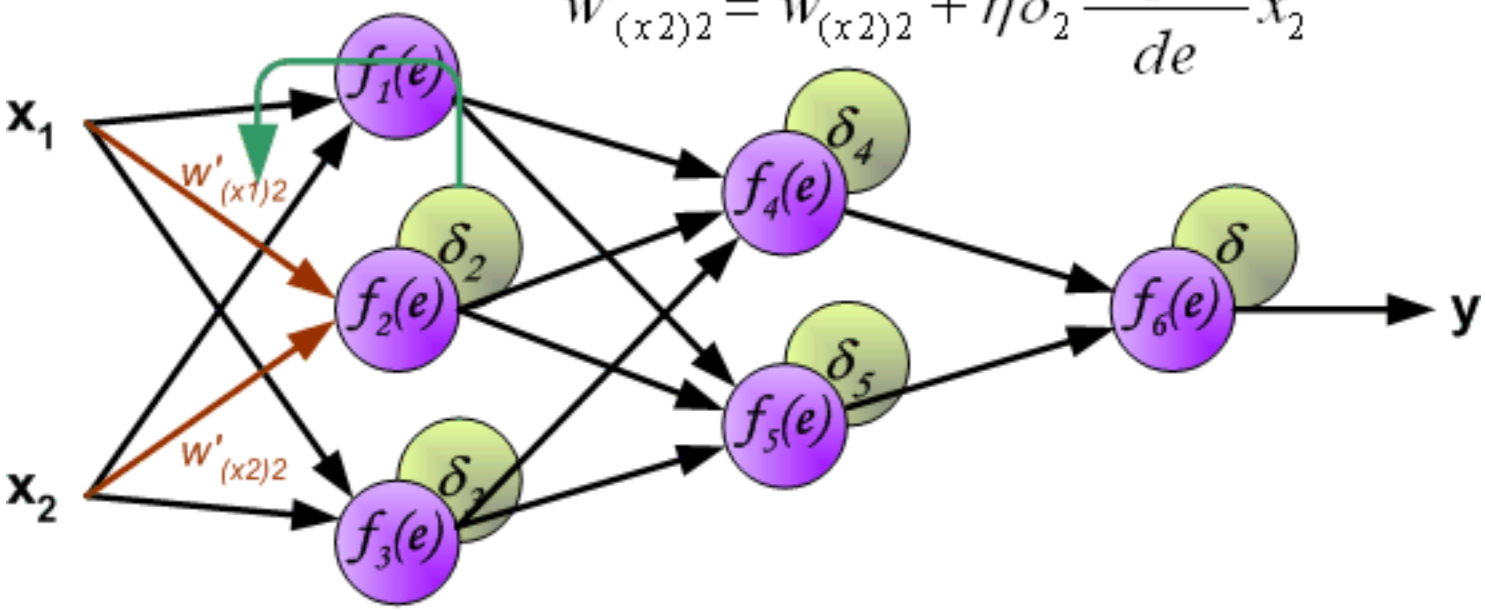
$$w'_{(x2)1} = w_{(x2)1} + \eta \delta_1 \frac{df_1(e)}{de} x_2$$



Backpropagation principle

$$w'_{(x1)2} = w_{(x1)2} + \eta\,\delta_2 \frac{df_2(e)}{de} x_1$$

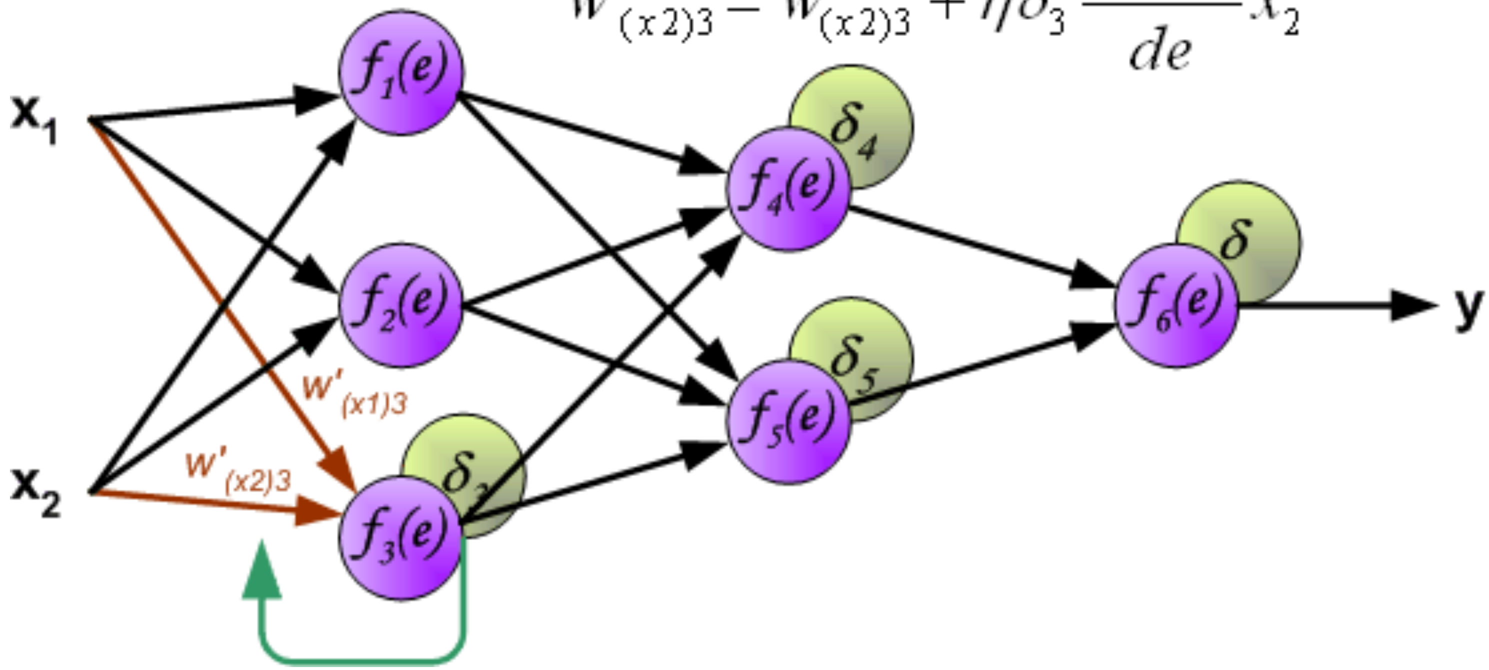
$$w'_{(x2)2} = w_{(x2)2} + \eta\,\delta_2 \frac{df_2(e)}{de} x_2$$



Backpropagation principle

$$w'_{(x1)3} = w_{(x1)3} + \eta \delta_3 \frac{df_3(e)}{de} x_1$$

$$w'_{(x2)3} = w_{(x2)3} + \eta \delta_3 \frac{df_3(e)}{de} x_2$$

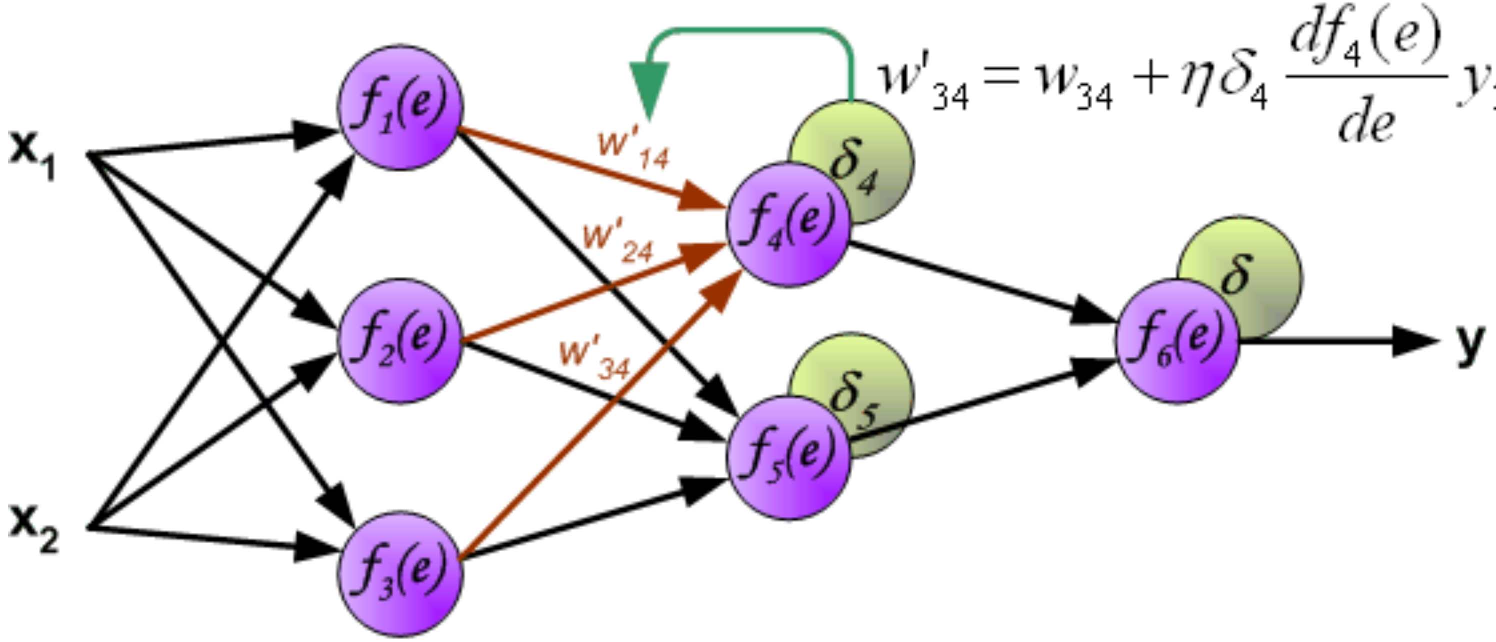


Backpropagation principle

$$w'_{14} = w_{14} + \eta\delta_4 \frac{df_4(e)}{de} y_1$$

$$w'_{24} = w_{24} + \eta\delta_4 \frac{df_4(e)}{de} y_2$$

$$w'_{34} = w_{34} + \eta\delta_4 \frac{df_4(e)}{de} y_3$$

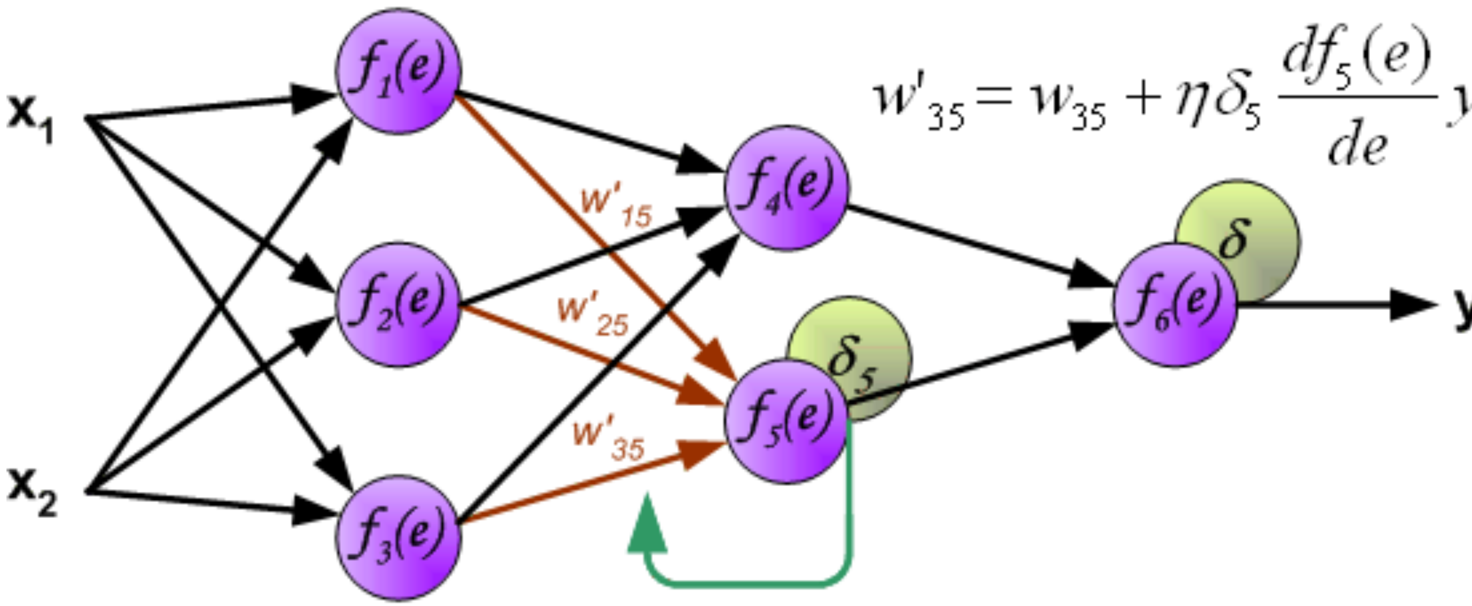


Backpropagation principle

$$w'_{15} = w_{15} + \eta \delta_5 \frac{df_5(e)}{de} y_1$$

$$w'_{25} = w_{25} + \eta \delta_5 \frac{df_5(e)}{de} y_2$$

$$w'_{35} = w_{35} + \eta \delta_5 \frac{df_5(e)}{de} y_3$$



Backpropagation principle

$$w'_{46} = w_{46} + \eta\,\delta\,\frac{df_6(e)}{de}\,y_4$$

$$w'_{56} = w_{56} + \eta\,\delta\,\frac{df_6(e)}{de}\,y_5$$