Machine learning introduction

Part II – Deep Neural Networks

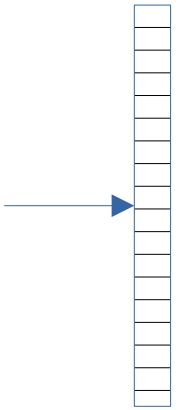
1 The deep networks

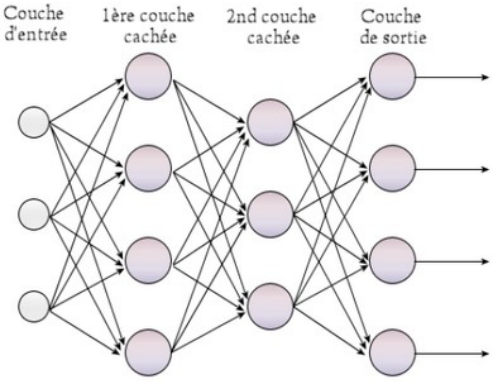
Simon Gay

Part II-1 : the deep networks

- **The beginning: image recognition**

  - ‘Classic’ approach:
    - Detection of features (circles, lines, rectangles…)





    - These features are used to fill a vector (number, position…)
    - A neural network is then trained on this vector
      - Not very reliable, error rate >25 % at ILSVRC 2011

Part II-1 : the deep networks

- **Deep learning**

  - Can a neural network generate such a feature vector from image ?
  - Problems:
    - More layers means more neurons and more connections to train
    - More neurons means larger training datasets

      → If the dataset is too small, the network would become expert for this data but would fail to generalize !

    - More neurons also means more computational resources

  - For many decades, neural networks were limited both by available resources and by available data sets.

Part II-1 : the deep networks

- **Deep learning**

  - Since middle of 2000s:
    - Computing power significantly increased
      - Parallelization on GPU (e.g. CUDA, 2007)
      - Dedicated architectures (Tensor Processing Unit, 2016)

    - But also collected data
      - GAFA and private life
      - Big data
      - Captcha
      - voluntarism

    - Open-source Dataset COCO (Common Objects in COntext) :
      >200k labeled images, 80+ classes

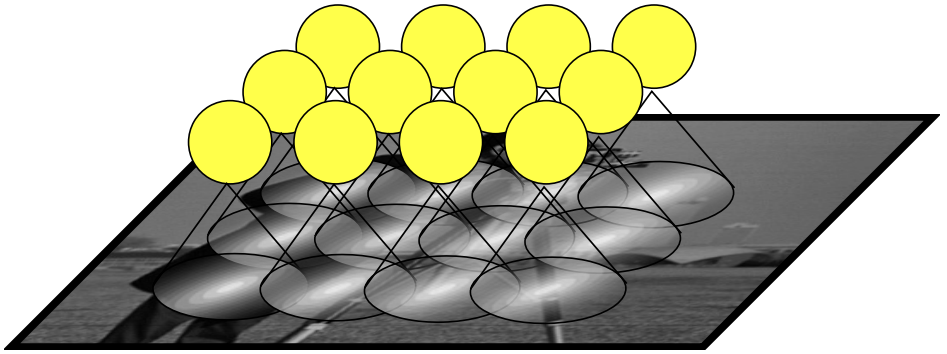
Part II-1 : the deep networks

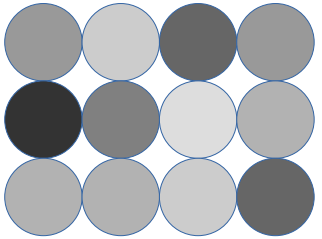
- **Deep learning**

  - But still not sufficient !
    - Example :

      an image of 100x100 pixels requires 10 000 connections per input neuron !

    - Biological inspiration :
      - 1968: analysis on animals show that neurons of first processing layers are connected to small area of the retina
      - Several neurons detects the same feature



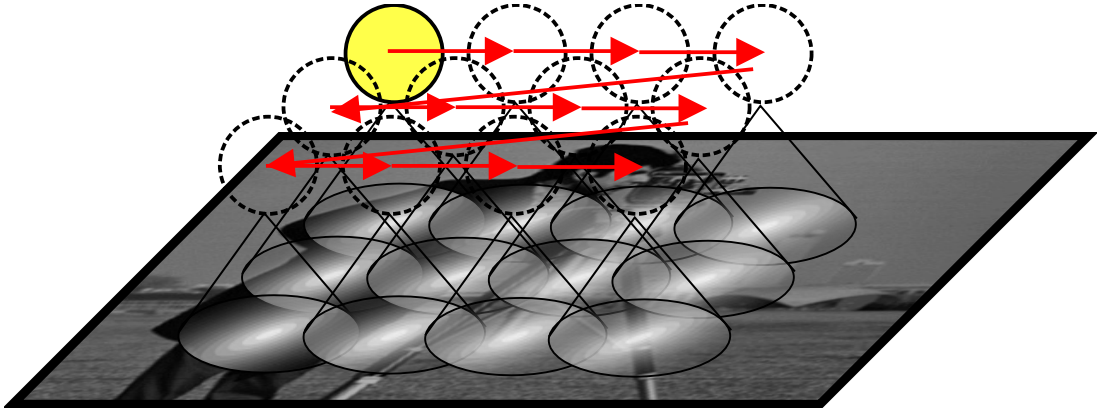


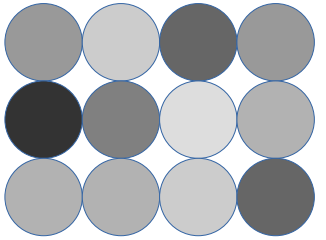
Detection of a feature
on the whole image

Part II-1 : the deep networks

- **Deep learning**

  - Yann Le Cunn (middle of 2000s) :
    - Idea: each neuron define a small feature and is applied on each position of the image



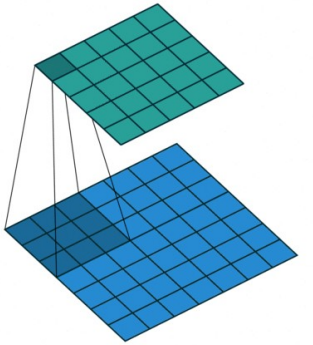


  - Feature detection with few weights
  - Works like a convolution → *convolutional neurons*

Part II-1 : the deep networks

- **Convolutions**
  - A kernel

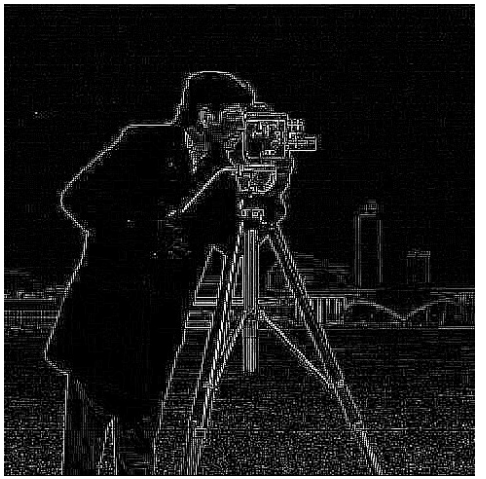
$$\begin{bmatrix} 0 & 0{,}25 & 0{,}5 & 0{,}25 & 0 \\ 0{,}25 & 0{,}5 & 0{,}75 & 0{,}5 & 0{,}25 \\ 0{,}5 & 0{,}75 & 1 & 0{,}75 & 0{,}5 \\ 0{,}25 & 0{,}5 & 0{,}75 & 0{,}5 & 0{,}25 \\ 0 & 0{,}25 & 0{,}5 & 0{,}25 & 0 \end{bmatrix}$$



https://github.com/vdumoulin

  - Scalar product is applied on each position of the image
  - The result is a matrix giving the presence of kernel's feature on each position of the image

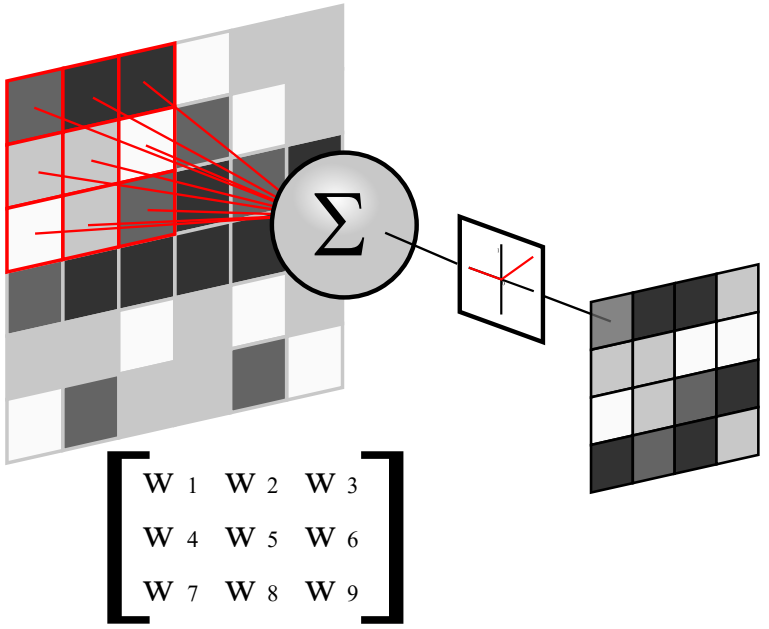
$$\otimes \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} =$$



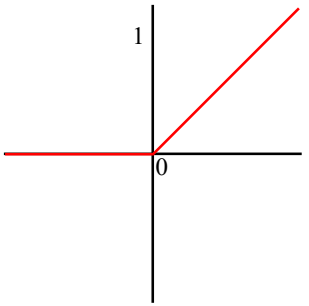
Part II-1 : the deep networks

- **Convolutional neurons**

  - Similar to a formal neuron
    - Input vector X
    - A set of weights W
      → kernel
    - An activation function
    - Reinforcement based on gradient descent



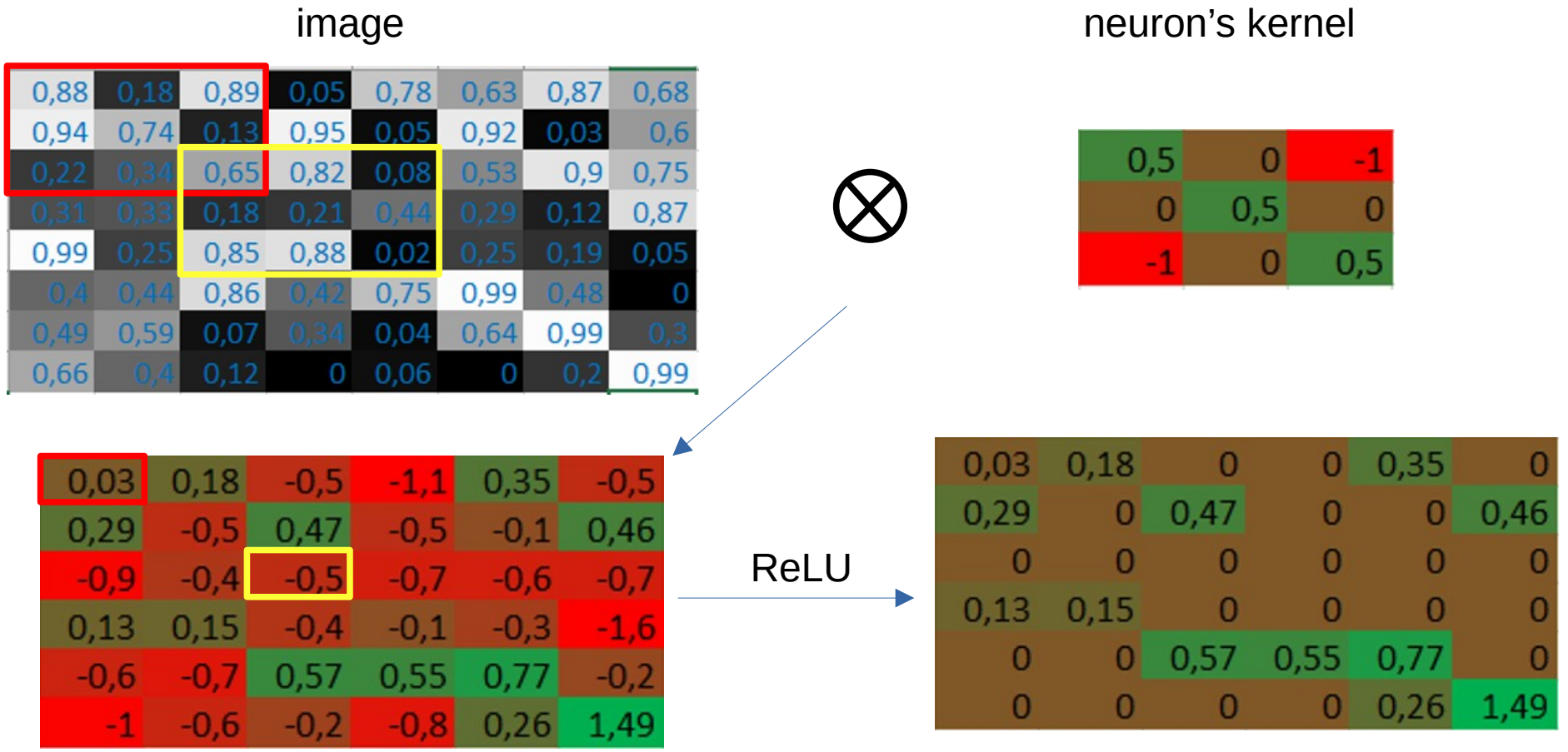
  - Output is an “image”
    - Each pixel p_{xy} is the neuron output at position (x,y)



  - Function RELU is often used : detect the presence of features

Part II-1 : the deep networks

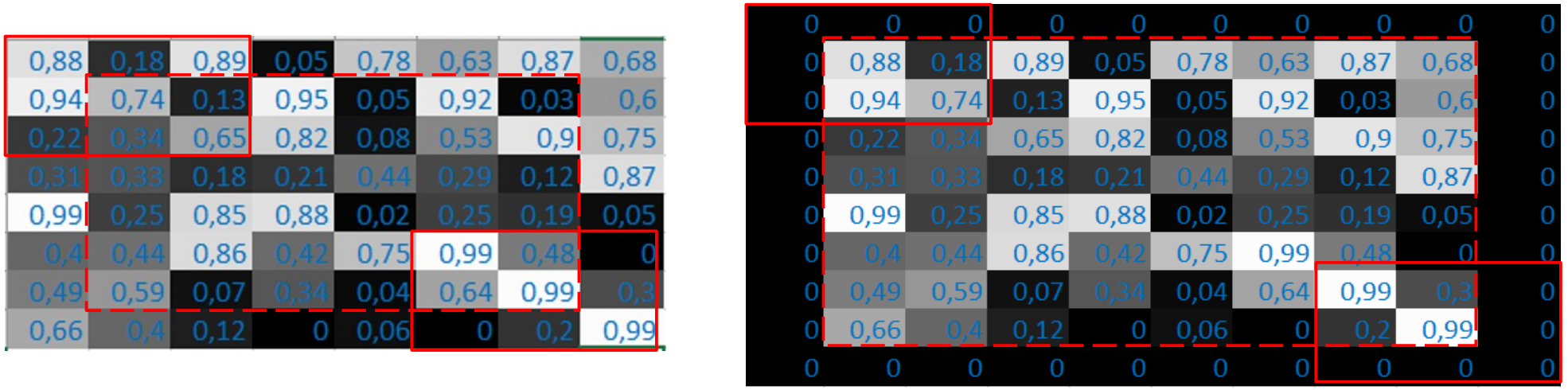
- **Convolutional neurons**



  - We can notice that the output is smaller than the image

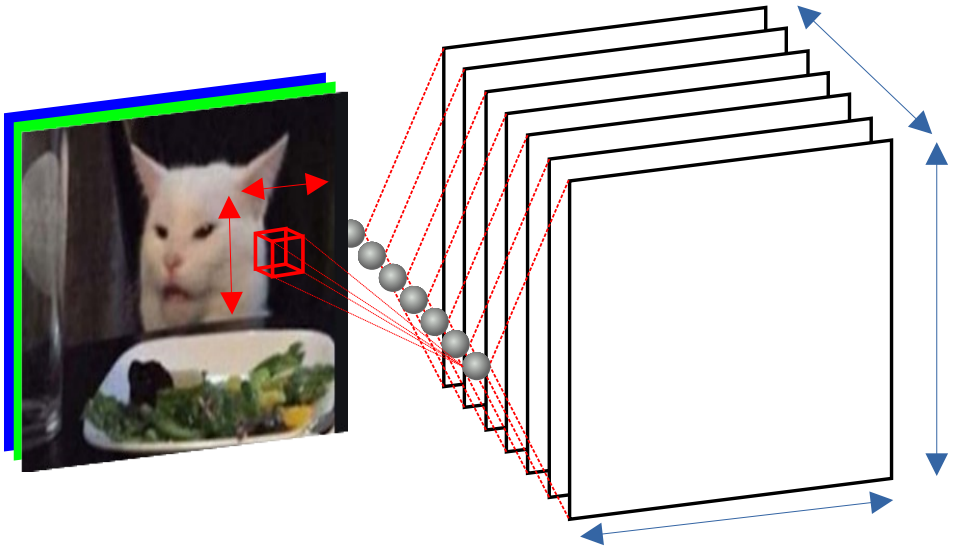
Part II-1 : the deep networks

- **Deep Learning**
  - Padding :
    - Pixels with values of 0 are added around the image



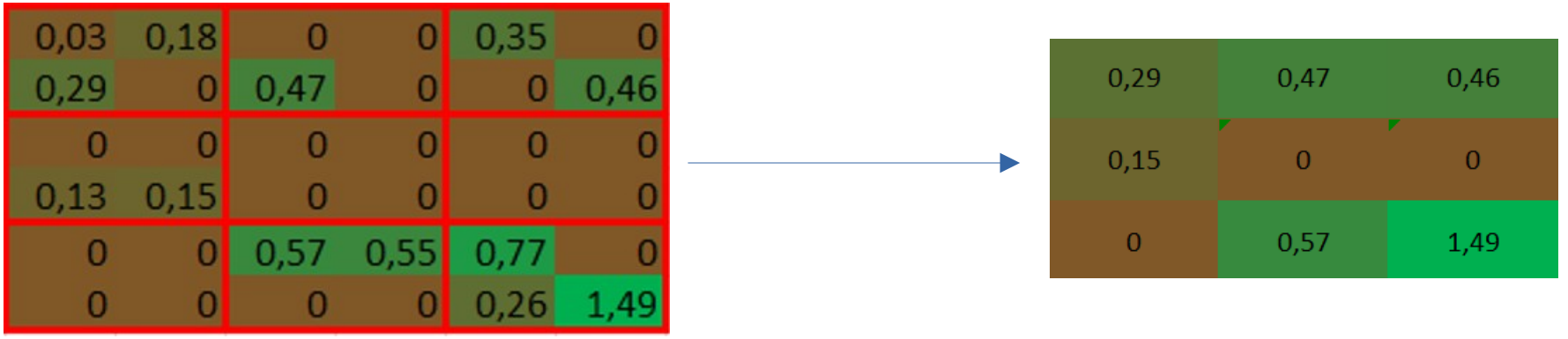
Part II-1 : the deep networks

- **Deep learning**
  - Each neuron of the first layer generate its own convolution image
    - These images are gathered as a unique image with a depth
    - Depth of an image is the number of “layers”
      - A RGB image has a depth of 3
    - The kernel of a neuron has 3 dimensions (width, height and depth) and moves on 2 dimensions on the image



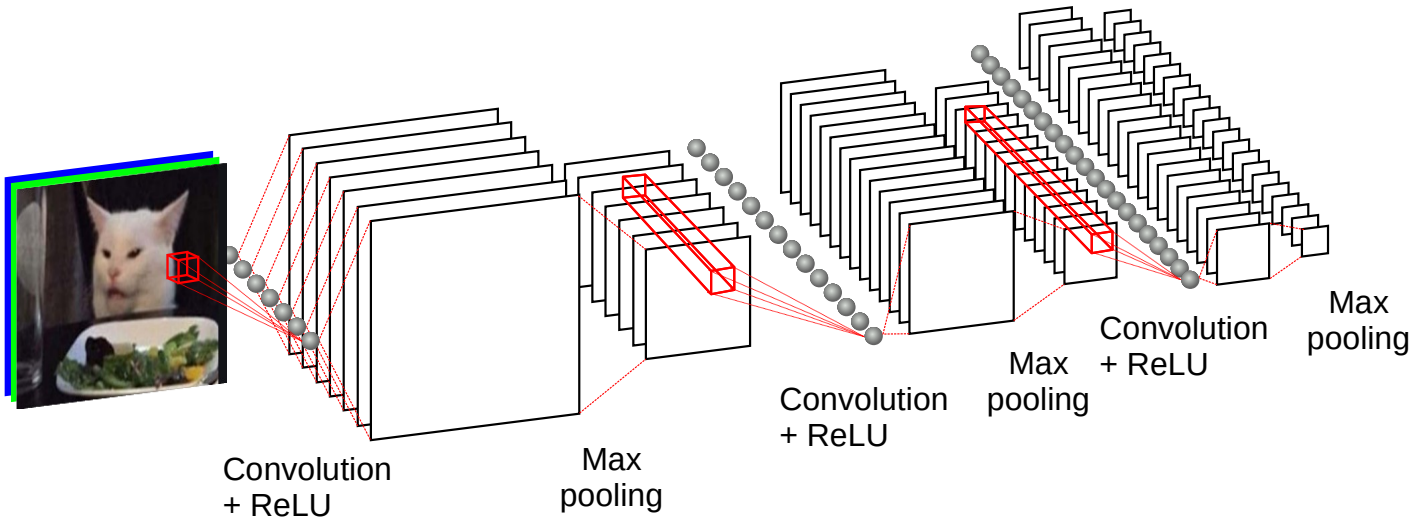
Part II-1 : the deep networks

- **Pooling**
  - As we looks for the presence of features, it is not necessary to keep adjacent pixels with high value
  - Pooling layer :
    - Adjacent pixels are gathered
      - Average pooling → average of the group
      - Max pooling → keep maximum value (most used method)
    - Reduces the number of pixels for next layers



Part II-1 : the deep networks

- **Deep learning**
  - Convolution and pooling layers can be chained
  - We lose spatial information but get information about complex features

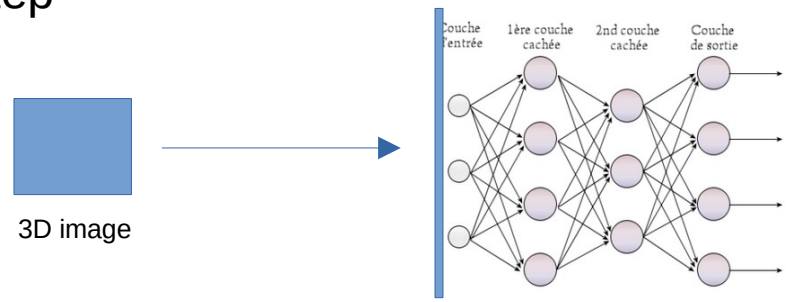


Part II-1 : the deep networks

- **Fully-connectec layers**

  - After several layers, the image looks more like a vector !
    → We thus can use it as input for a ‘normal’ neuronal network

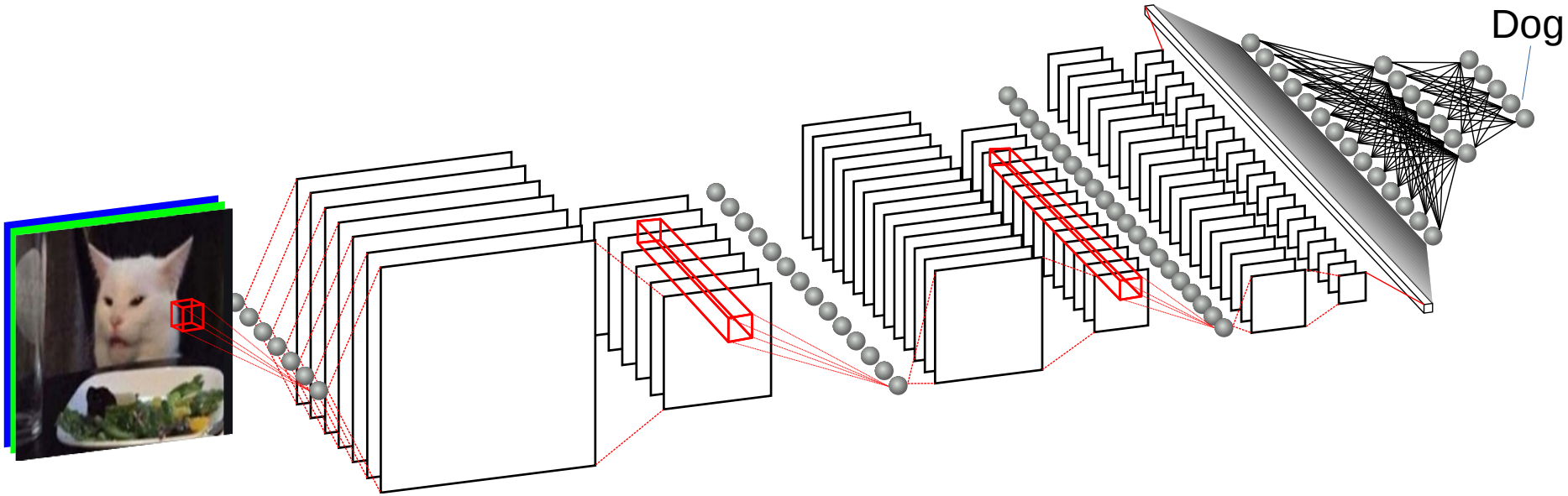
    - Last image is converted to a 1-D vector
      → *flattening* step



    - A neuronal network is then used to “interpret” the feature vector
      → these neuron layers are called *‘fully connected’*

Part II-1 : the deep networks

- **“Classical” architecture of a convolutional deep network**

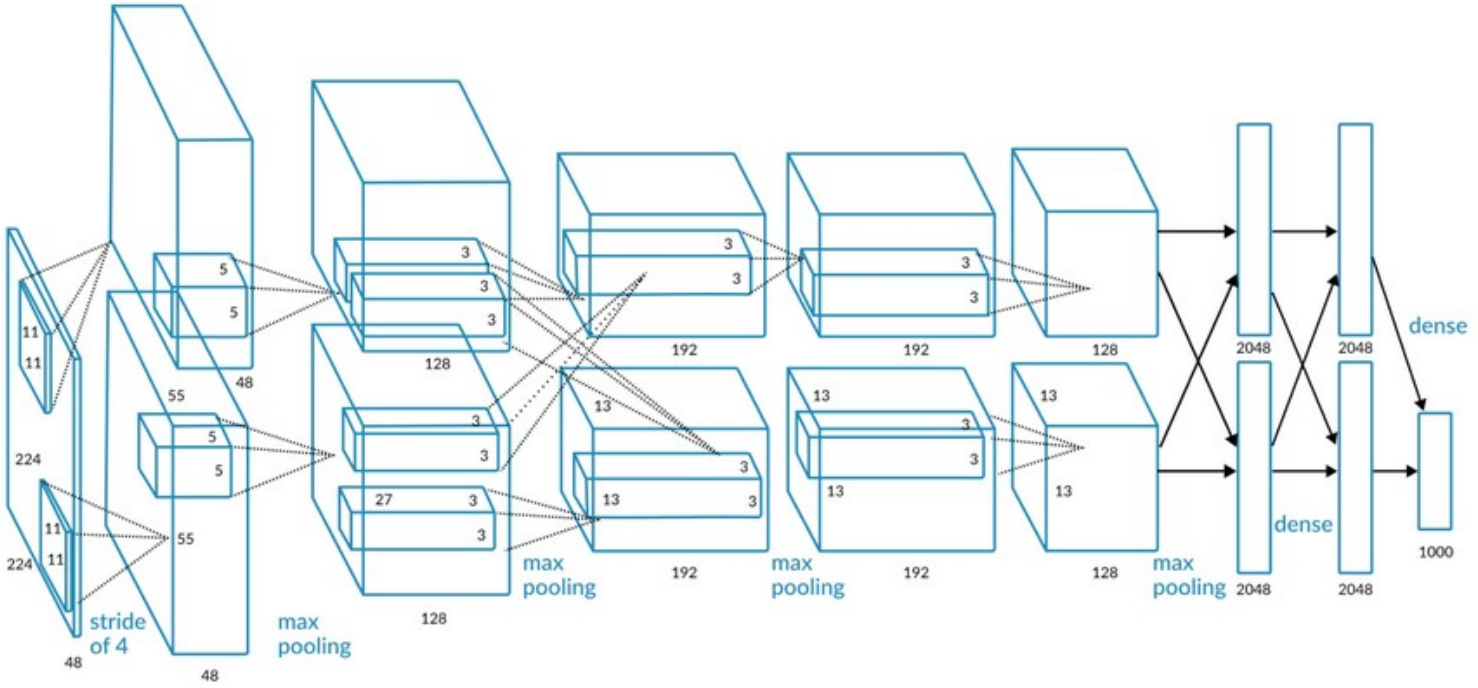


  - convolution + ReLU
  - Pooling
  - convolution + ReLU
  - Pooling
  - …
  - Flattening
  - fully connected layer
  - Output layer

Part II-1 : the deep networks

- **The architecture can be more complex:**

  example : AlphaNet (2012)

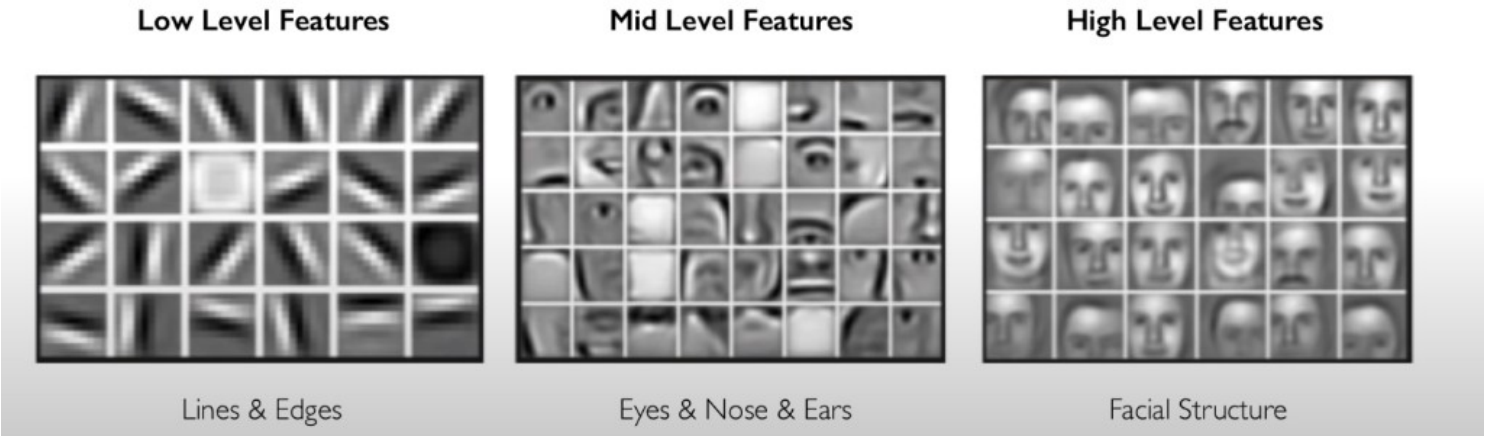


Training a deep network: the question of the dataset

Part II-1 : the deep networks

- **Training a deep network**

  - The more layers of neurons there are, the larger the learning dataset must be
    - A deep network may require a dataset of tens, even hundreds thousand images (or other kind of data)
  - Do we need a huge dataset for each new problem ?
  - For two similar problems, features defined by lowest layers are similar !

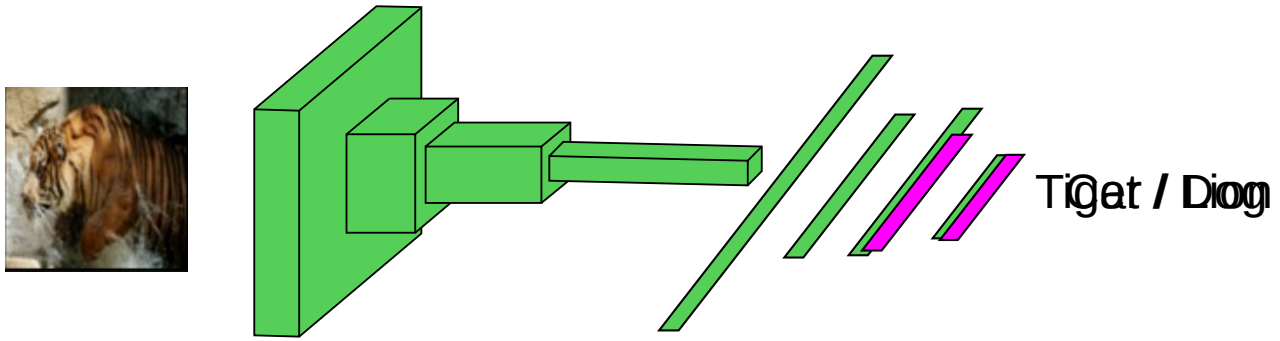


Part II-1 : the deep networks

- **Training a deep network**
  - For similar problem, it is possible to re-use low layers of an existing trained network

    → Principle of **transfer learning**

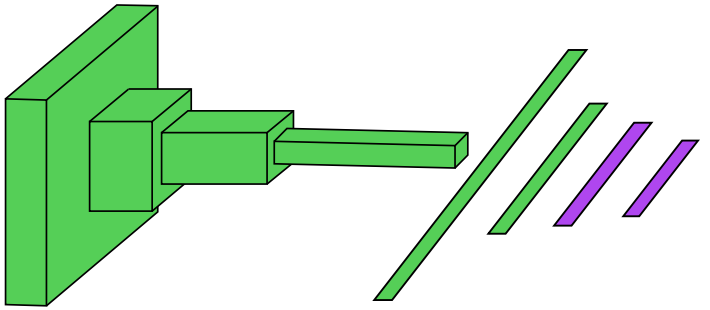
  - We can use a network trained on a similar problem
  - We then replace last layers of the network with new layers



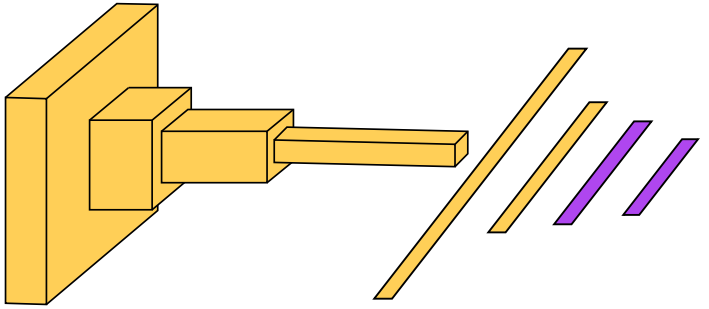
  - The number of layers to replace depends on problems similarities and available dataset's size

Part II-1 : the deep networks

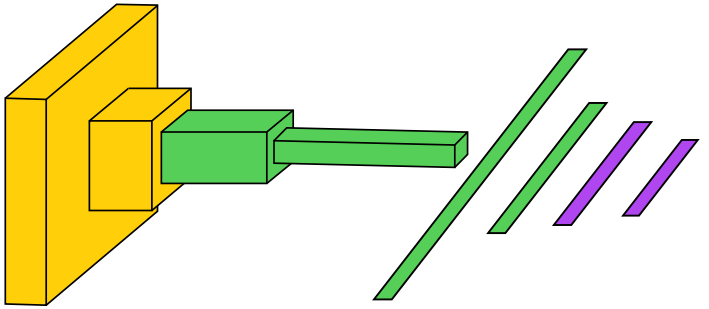
- **Transfer Learning**
  - Fine tuning: last layers (often fully connected part) are replaced with new initialized layers
    - Faster learning with less examples
    - Low level features can adapt to the new problem
    - Still requires a large dataset



  - Feature extraction: last layers are replaced and remaining layers are frozen
    - Fast learning with few examples
    - Resulting network is not always efficient



  - Partial Fine tuning: olast layers are replaces, but only lowest layers are frozen
    - Good compromise between above methods
    - Low level features are more ‘universal’

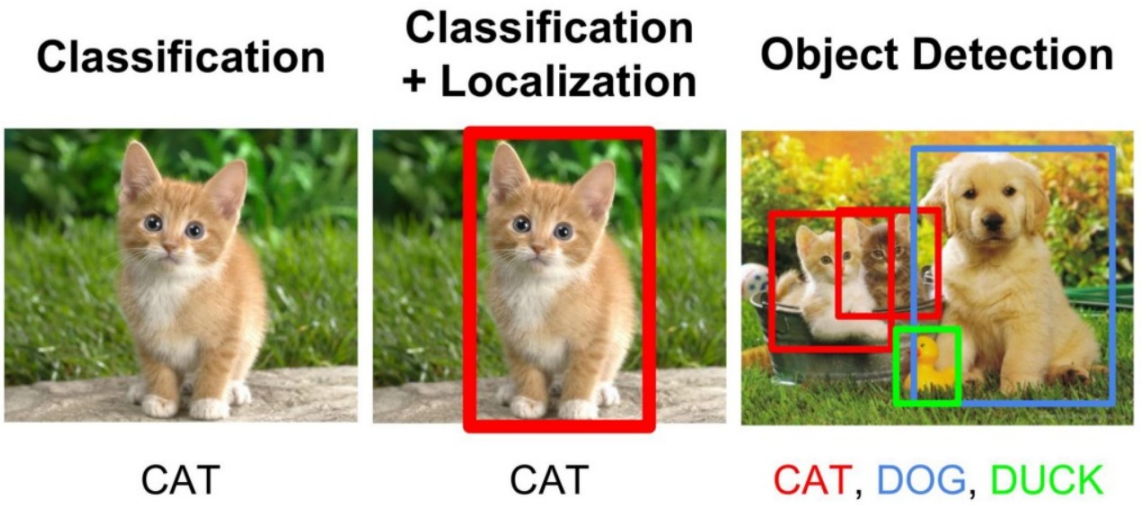


Part II-1 : the deep networks

- **Transfer Learning**
  - **Several well-trained open-source networks are proposed for transfer learning :**
    - AlexNet : architecture used for ILSVRC 2012
    - VGGnet : one of most versatile. Uses 16 convolution layers
    - GoogleNet : winner of ILSVRC 2014, uses 22 convolution layers
    - ResNet : winner of ILSVRC 2015 (with an error ratio of 3,57 %, it is more efficiant than human participants!), uses 152 convolution layers

Part II-1 : the deep networks

- **Applications**
  - Recognition of images, but also sound, texts...



  - Classification, text translation

Part II-1 : the deep networks

- **Applications**
  - Question : is it possible to recreate the image from the feature vector ?
    - Successive convolution generate an information loss
    - The feature vector only contains a fraction of image information
    - New layers to reverse convolutional layer’s processing
      - De-pooling
      - transposed convolution

Part II-1 : the deep networks

- **Image reconstruction**
  - De-Pooling :
    - Nearest neighbor

0,29	0,47	0,46
0,15	0	0
0	0,57	1,49

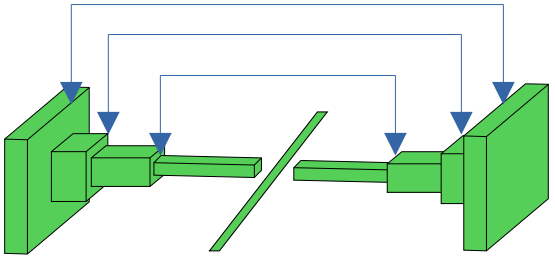
0,29	0,29	0,47	0,47	0,46	0,46
0,29	0,29	0,47	0,47	0,46	0,46
0,15	0,15	0	0	0	0
0,15	0,15	0	0	0	0
0	0	0,57	0,57	1,49	1,49
0	0	0,57	0,57	1,49	1,49

    - Bed of nails

0,29	0	0,47	0	0,46	0
0	0	0	0	0	0
0,15	0	0	0	0	0
0	0	0	0	0	0
0	0	0,57	0	1,49	0
0	0	0	0	0	0

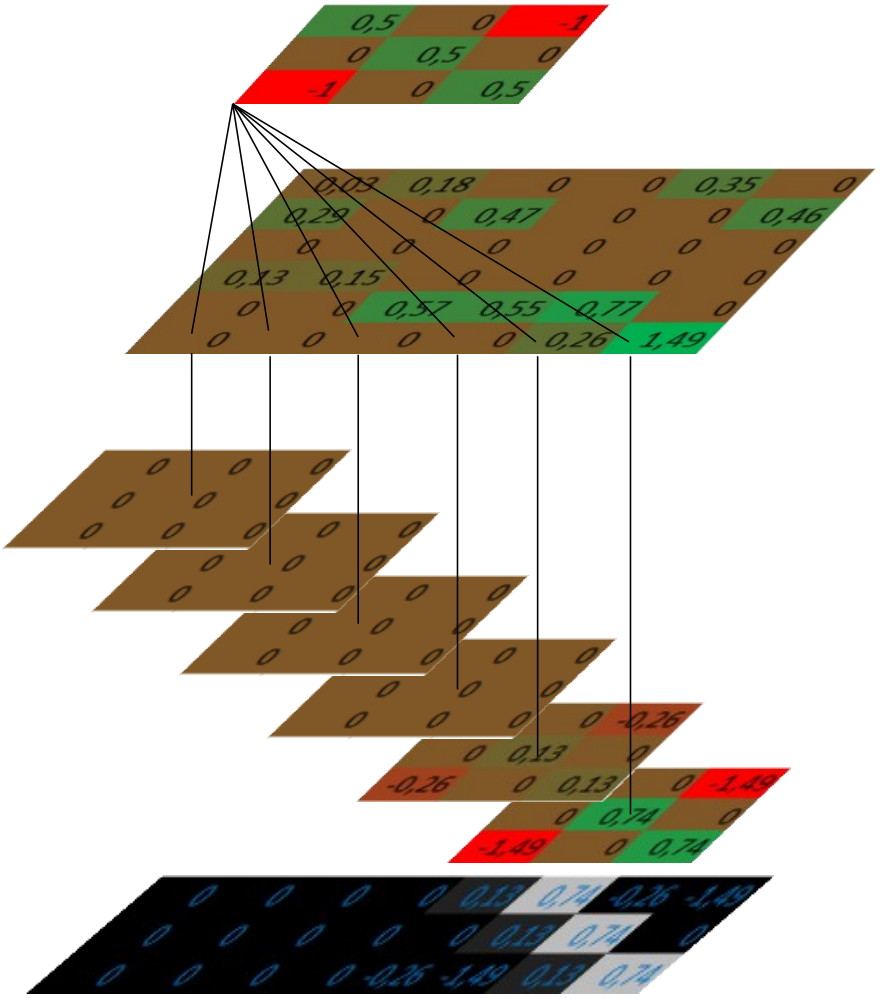
    - Max UnPooling

      → use the mirror pooling layer to define which pixel get the value



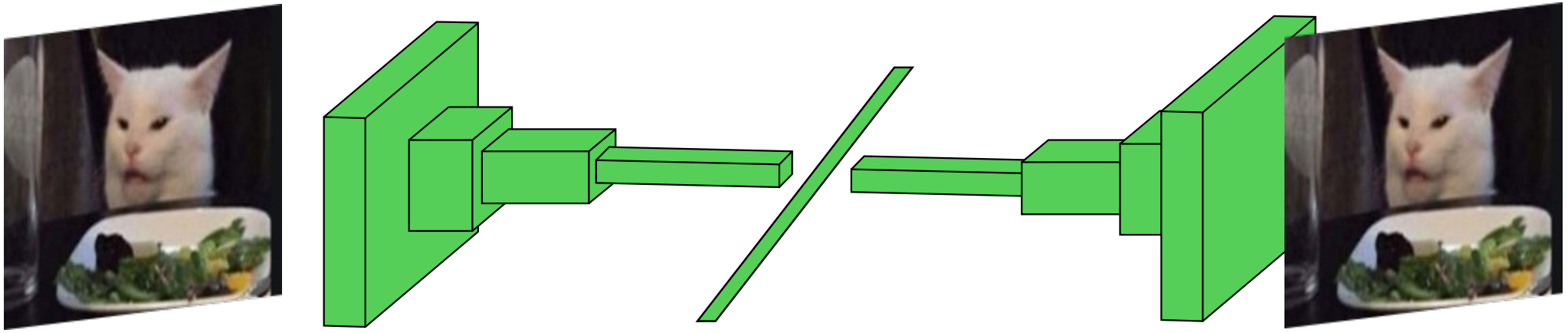
Part II-1 : the deep networks

- **Image Reconstruction**
  - Transposed Convolution:
    - Principle of convolution is ‘reversed’
    - Output image is the sum of kernel weighted by convolution image values



Part II-1 : the deep networks

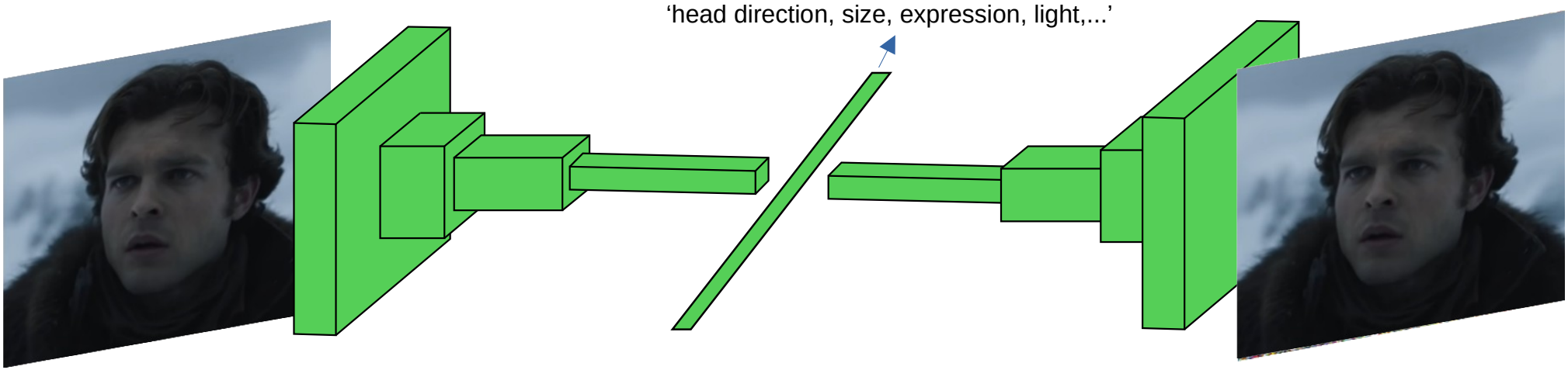
- **Image reconstruction**
  - The fully-connected part is replaced by a ‘mirror’ version of the network



  - Input image is used as expected result ($E = | Img_{ref} – Img_{gen} |^2$)
  - First part encodes the information: ‘encoder’ network
  - Second part ‘decode’ the feature vector to reconstruct the image: ‘decoder’ network
    - Decoder learns to generate textures, edges… from feature description
  - Very efficient way to de-noise an image

Part II-1 : the deep networks

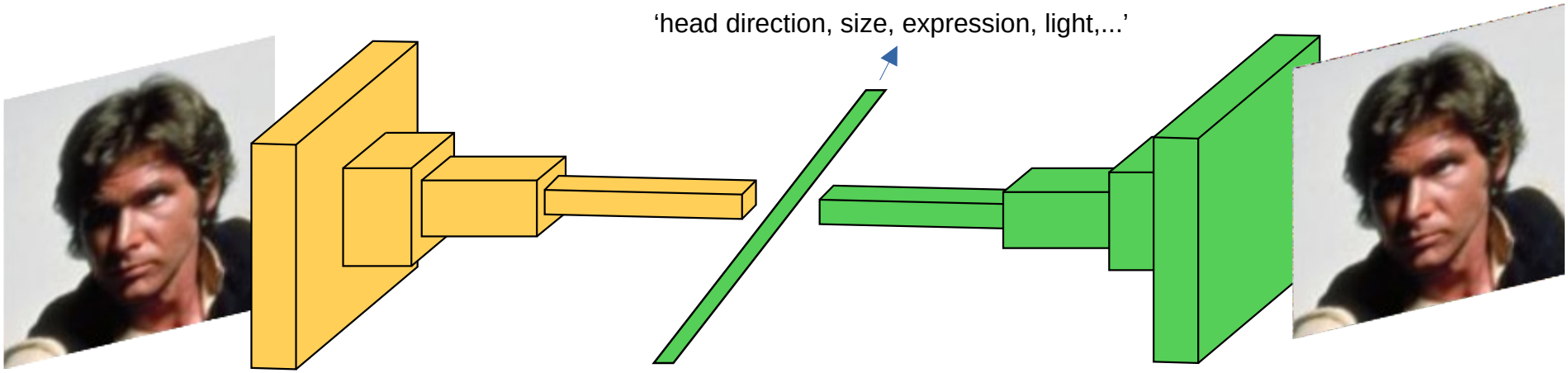
- **Image reconstruction**
  - The network ‘learns’ to define minimal information
  - Let’s train the network on images of a specific face



  - The network will learn to encode minimum data to reconstruct the face
    - Head orientation and size, facial expression… (implicitly)
    - The decoder learns to reconstruct the image with these features

Part II-1 : the deep networks

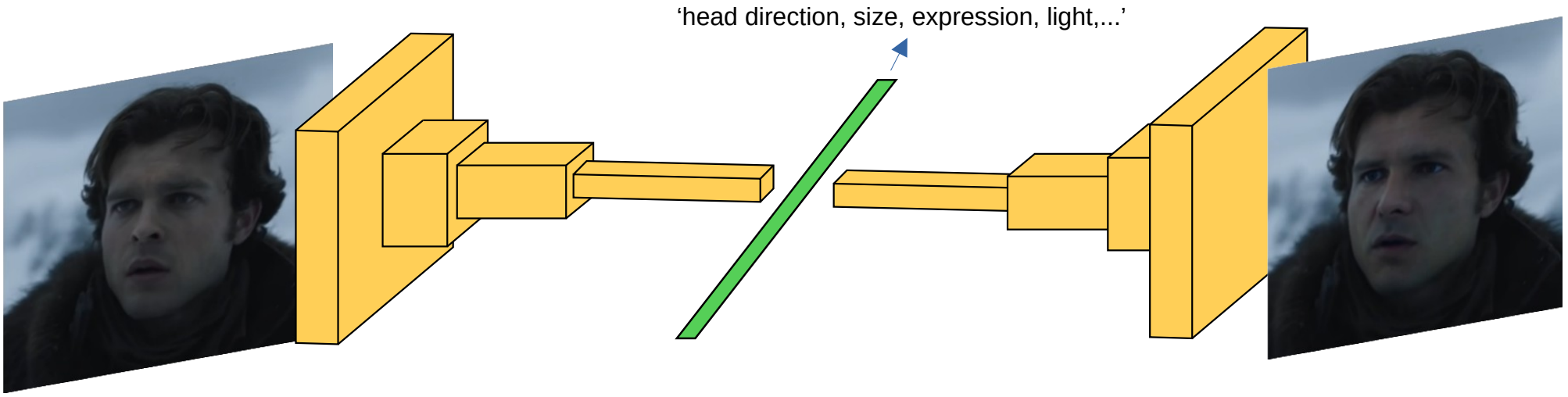
- **Image reconstruction**
  - The encoder’s weights are frozen
  - Decoder is reinitialized and trained with another face



  - The encoder will provide feature vectors in a similar way
  - But the decoder must learn to generate this face from features

Part II-1 : the deep networks

- **Image reconstruction**
  - We now froze the whole network
  - An image of first face is presented to the network
    - The encoder extracts features from this face
    - The decoder will generate an image of the second face from these features



- Principle of deep fake

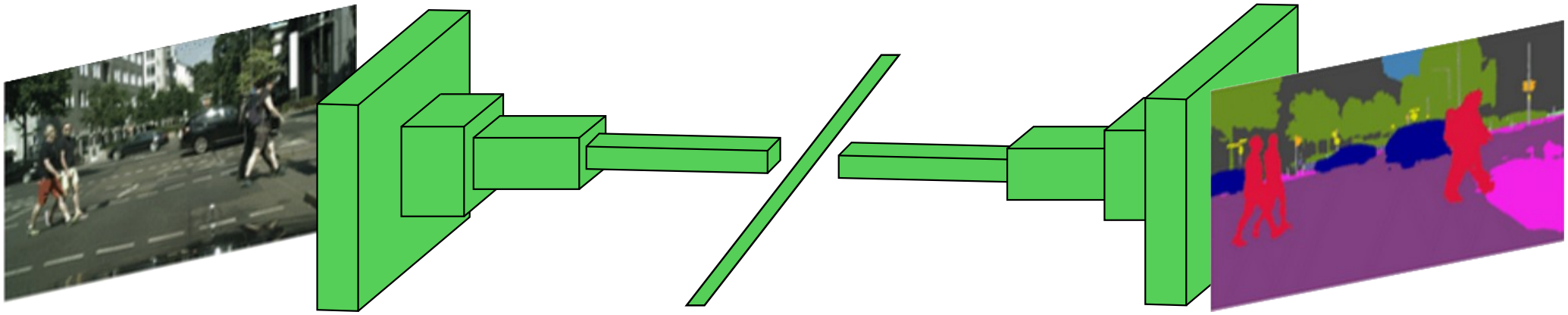
Part II-1 : the deep networks

- **Image reconstruction**

Harrison Ford in Solo: A Star Wars Story (Shamook)

Part II-1 : the deep networks

- **Fully-convolutional networks**
  - We can also use other type of images as output
  - Example : Mask-RCNN
    - The network is trained on images, with hand-labeled mask images

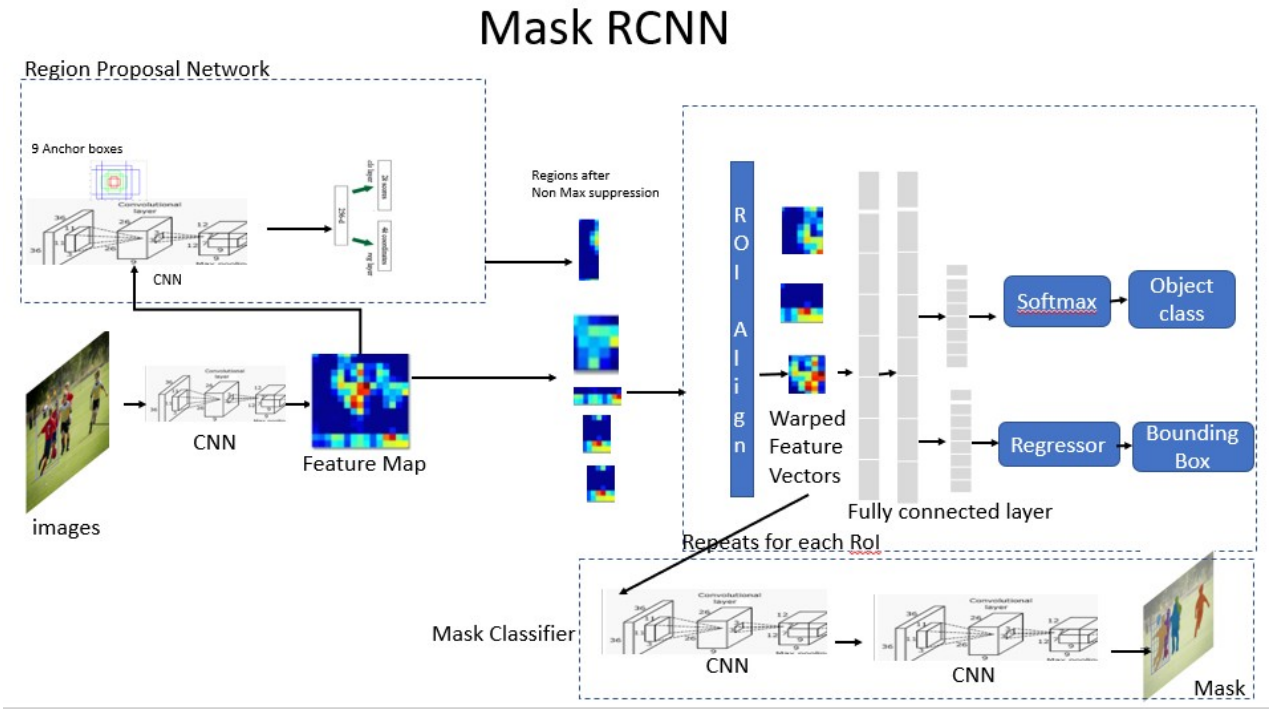


  -

Part II-1 : the deep networks

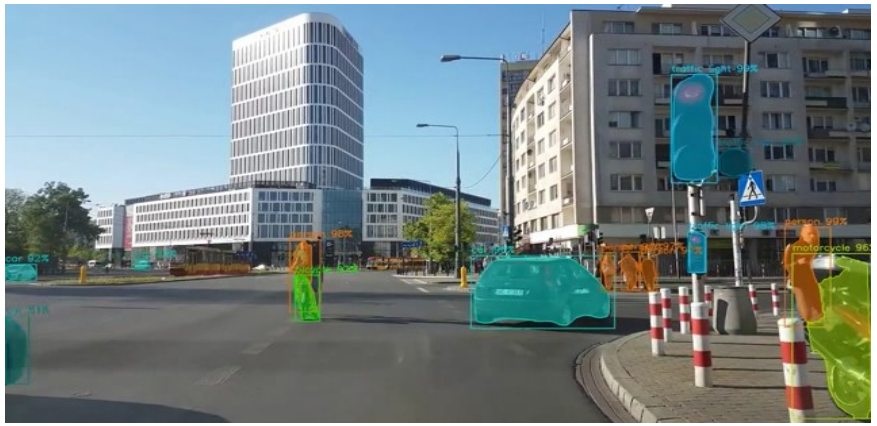
- **Fully-convolutional network**
  - Mask-RCNN : in practice, a little more complex
    - Detection of objects in the image

      → extraction of region of interest containing objects
    - Detection pixel by pixel to generate a mask



Part II-1 : the deep networks

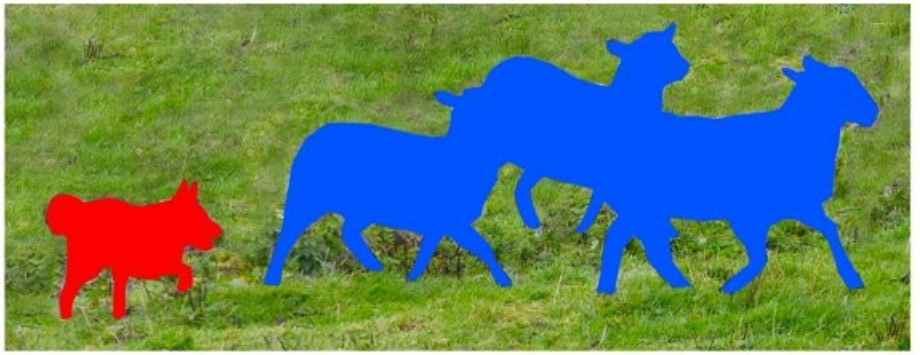
- **Fully-convolutional network**
  - Mask-RCNN, trained on dataset COCO (Common Objects in COntext)



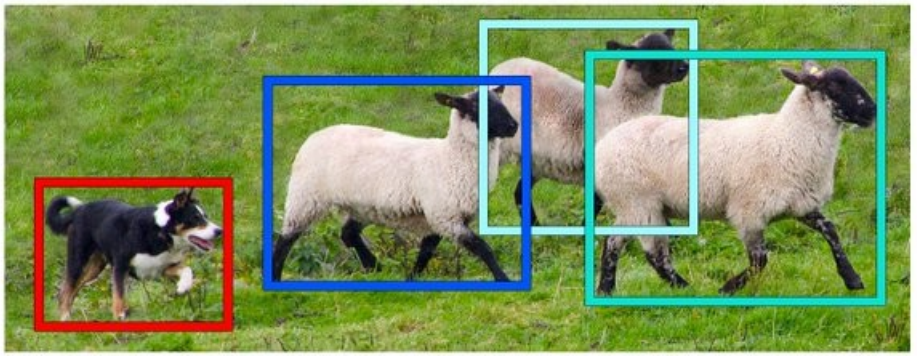
Part II-1 : the deep networks

- **Fully-convolutional network**

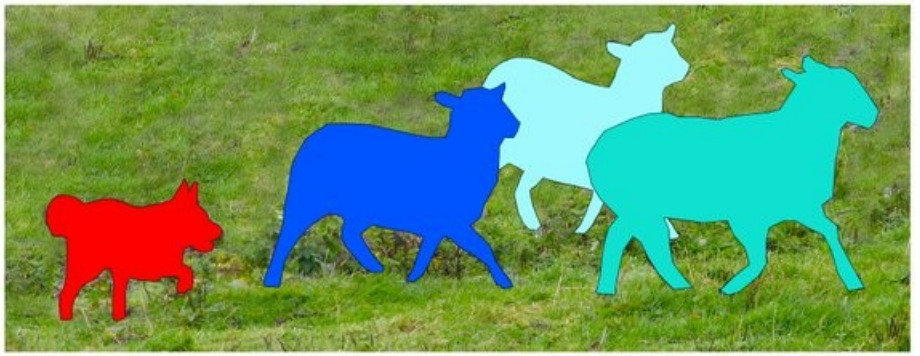
Image Recognition



Semantic Segmentation



Object Detection



Instance Segmentation

Training a deep network: bias and security

Part II-1 : the deep networks

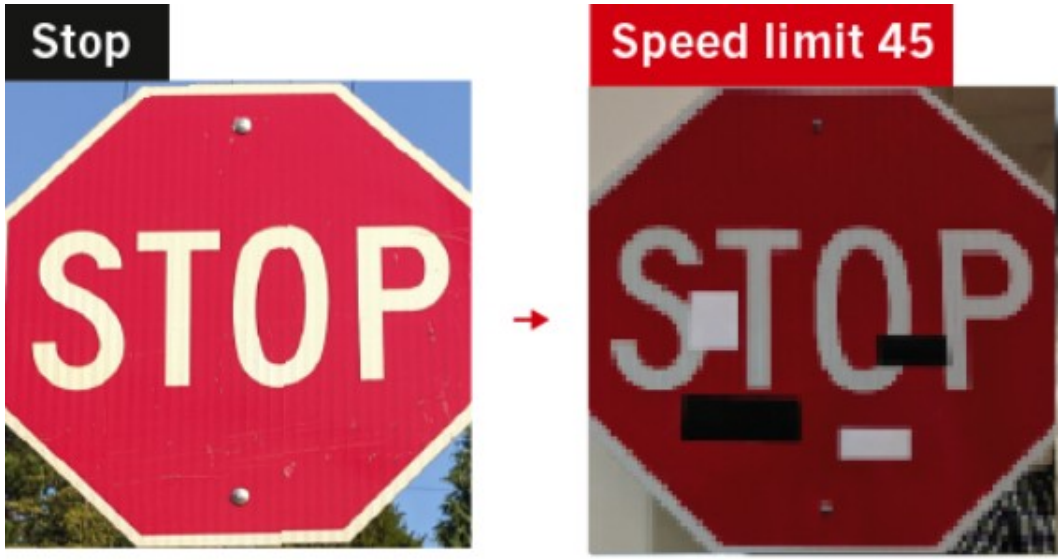
- **The dataset bias**
  - Training a deep network requires a huge number of examples
    - Gathering a large number of example can be difficult, leading to statistical bias in the collected samples
    - Neural networks are very sensitive to statistical bias in datasets and will use ANY regularity they can found to minimize errors
      - Example : a network defining the age of a person uses the size of ears
    - These bias may generate serious issues
      - In 2014, Amazon used a deep network to analyze applicants' resume
      - This network massively rejected women's resume
      - The network was trained on resumes collected during 10 past years, most of them were from men...

Part II-1 : the deep networks

- **The dataset bias**
  - Example de biais :
    - Network trained to recognize wolves from husky
    - High success ratio
    - But some (obvious) errors !
  - After analyzing features, it appeared that the network only detect snow on the background !
    - In the dataset, a large majority of wolf pictures have snow in the background

Part II-1 : the deep networks

- **Security**
  - Training a network with a biased dataset not only reduce its reliability, but also makes it more vulnerable to attacks



Part II-1 : the deep networks

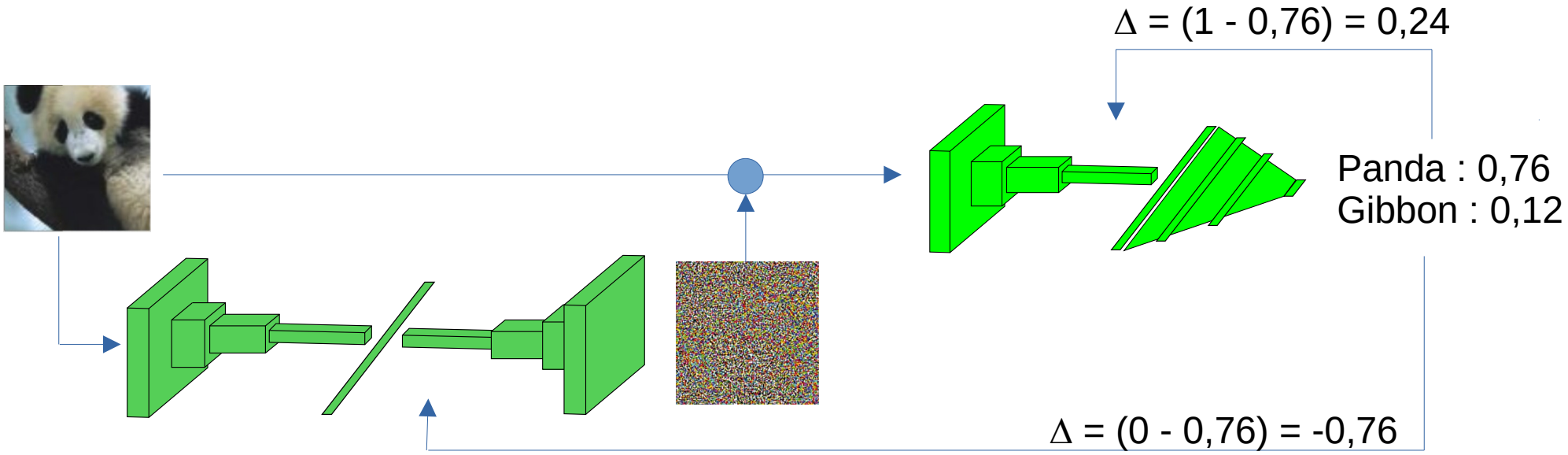
- **Security**
  - It is also possible to ‘manipulate’ the network to modify its output

  - The added noise disrupted low level features, which also interfere with higher level features, and thus, changed the result.
  - How this variation was created ?

Part II-1 : the deep networks

- **Manipulating a network**

  - A ‘normal’ deep network recognize animals on images
  - An encoder-decoder network is added to generate a ‘noise image’ that is added to the initial image
  - The success of second network is related to the failure of first network



  - If the two networks learn simultaneously, they have antagonist objectives !