OTBTF

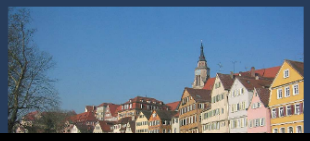
The open-source deep learning framework for remote sensing images processing

Rémi Cresson (IRSTEA/UMR TETIS)

Dino Ienco (IRSTEA/UMR TETIS)

DEEP LEARNING

APPLICATIONS



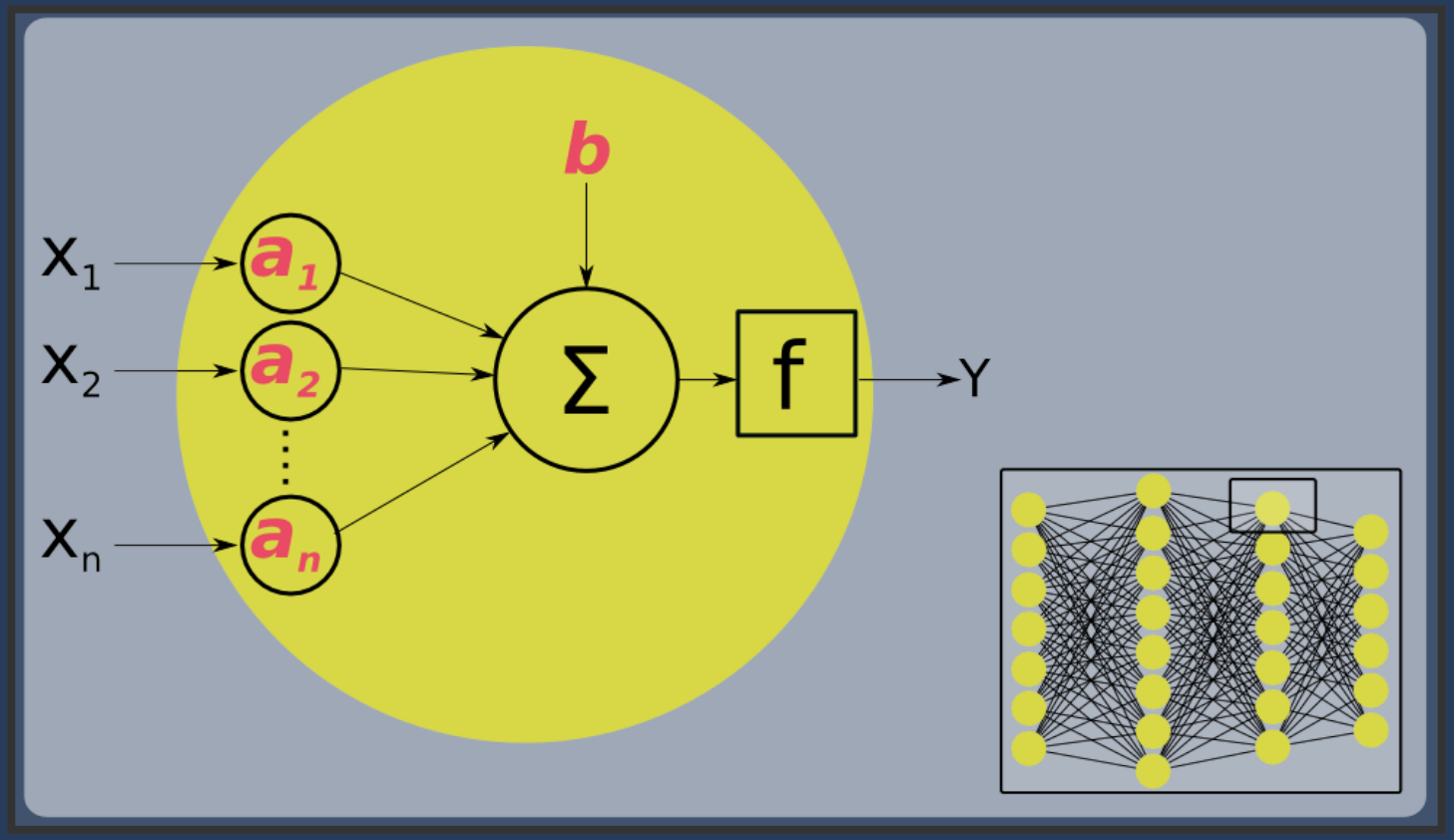
Self driving cars, arts, games, langage processing, images processing, bioinformatics, robotics, health, fraud detection, security, advisor systems, video processing, vocal recognition, pose estimation, astronomy, oceanography, communication, mobile devices, advertising, ...


Gatys et Al.

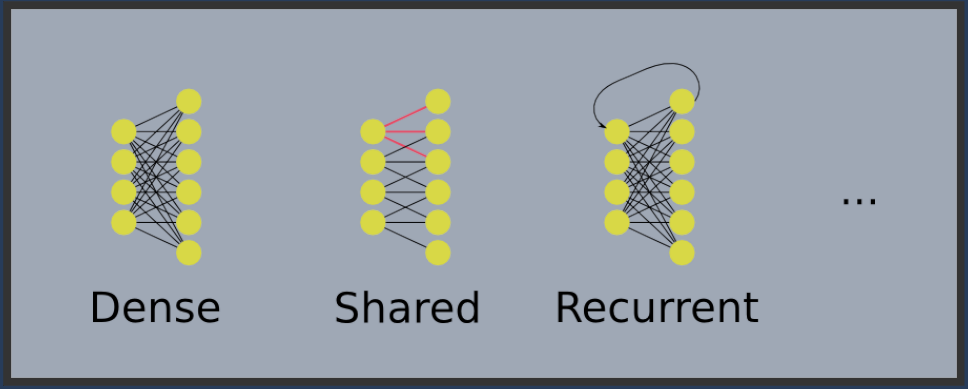

Self-Driving
Metamorworks/Adobe Stock Images


OpenPose

WHAT IS A DEEP NET?



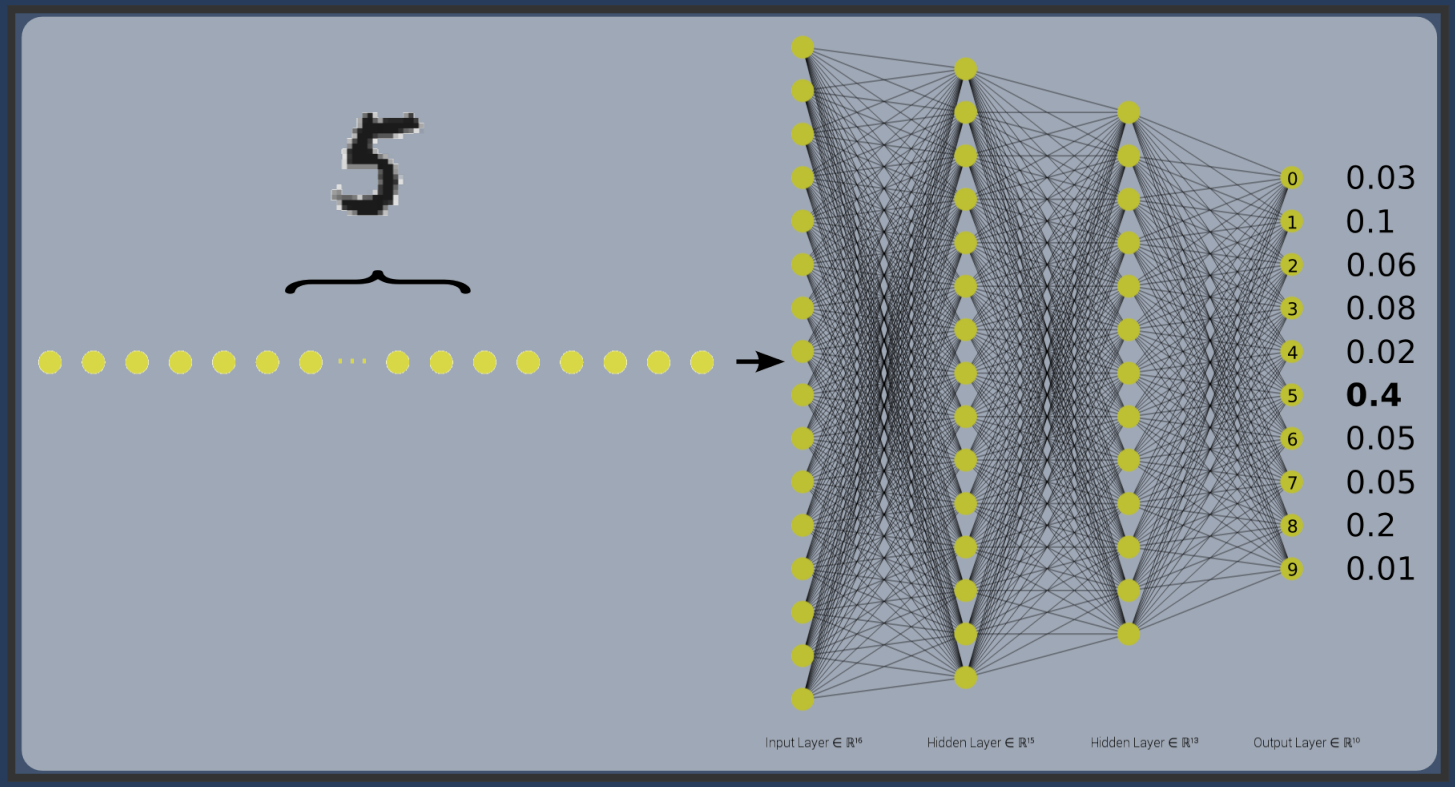
TYPE OF CONNECTIONS



e.g. convolution can be modeled connections with shared weights:

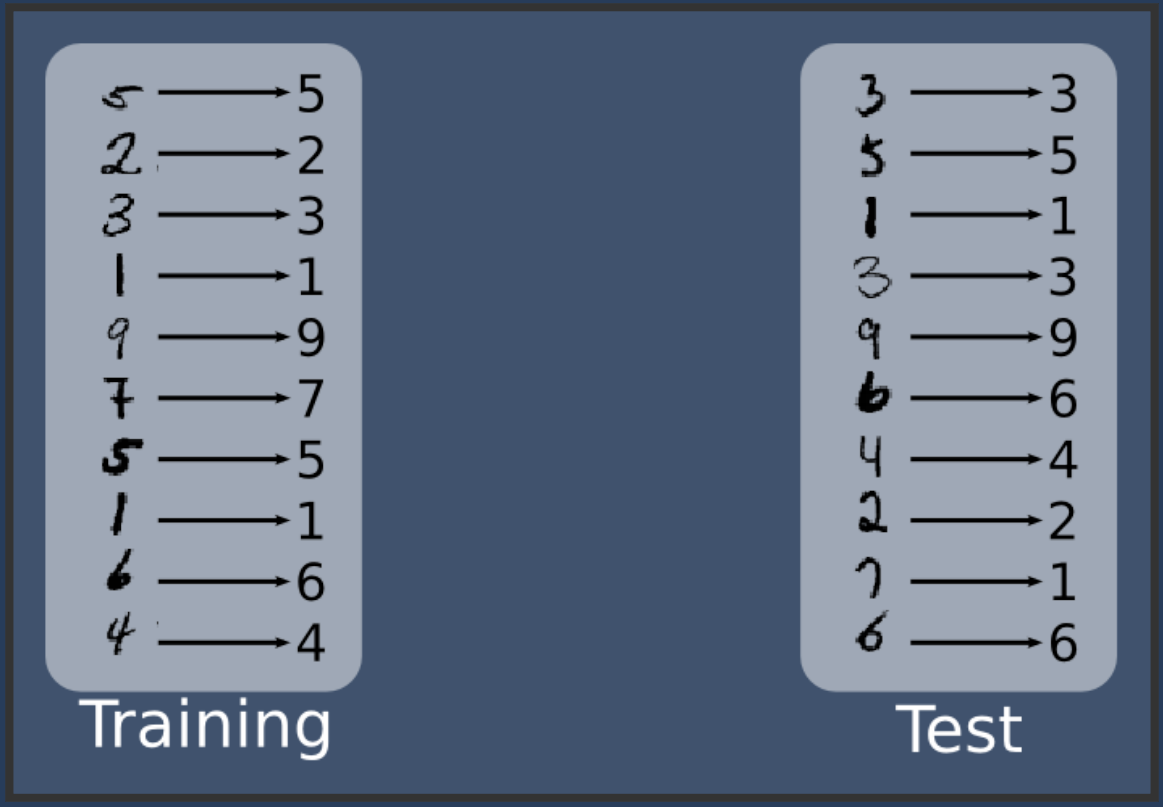
$$Y(x,y) = (X \circledast a)(x,y) = \sum_{i=1} \sum_{j=1} X(x-i, y-j) \times a(i,k)$$

TASK EXAMPLE

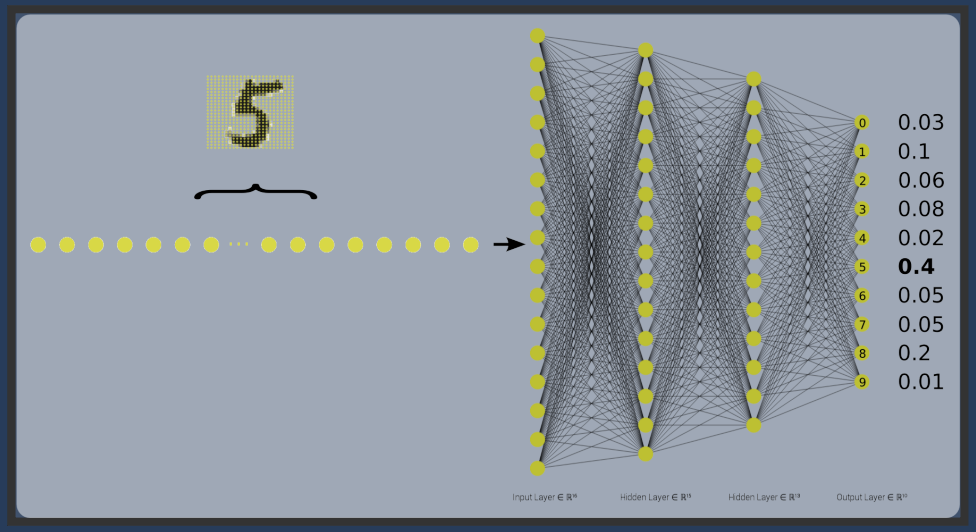


HOW DOES NETWORK TRAINING WORK?

DATASET

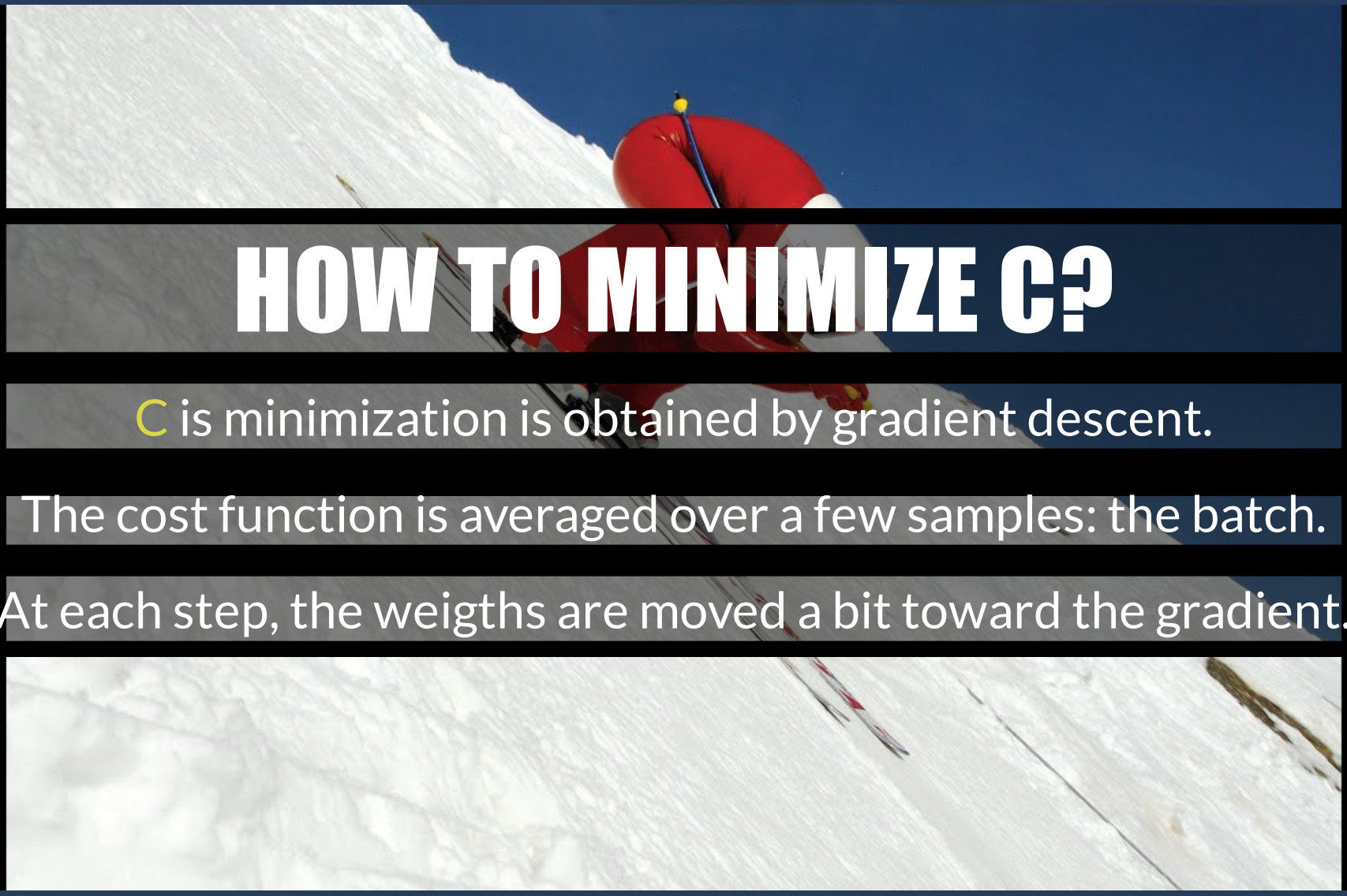

5 → 5
2 → 2
3 → 3
1 → 1
9 → 9
7 → 7
5 → 5
1 → 1
6 → 6
4 → 4
Training
3 → 3
5 → 5
1 → 1
3 → 3
9 → 9
6 → 6
4 → 4
2 → 2
7 → 1
6 → 6
Test

OBJECTIVE



We minimize a cost function C expressing how good is the net output (the estimated 10 components vector) w.r.t. the actual value

$$C = (y_{estim} - y_{truth})^2$$



HOW TO MINIMIZE C?

C is minimization is obtained by gradient descent.

The cost function is averaged over a few samples: the batch.

At each step, the weigths are moved a bit toward the gradient.

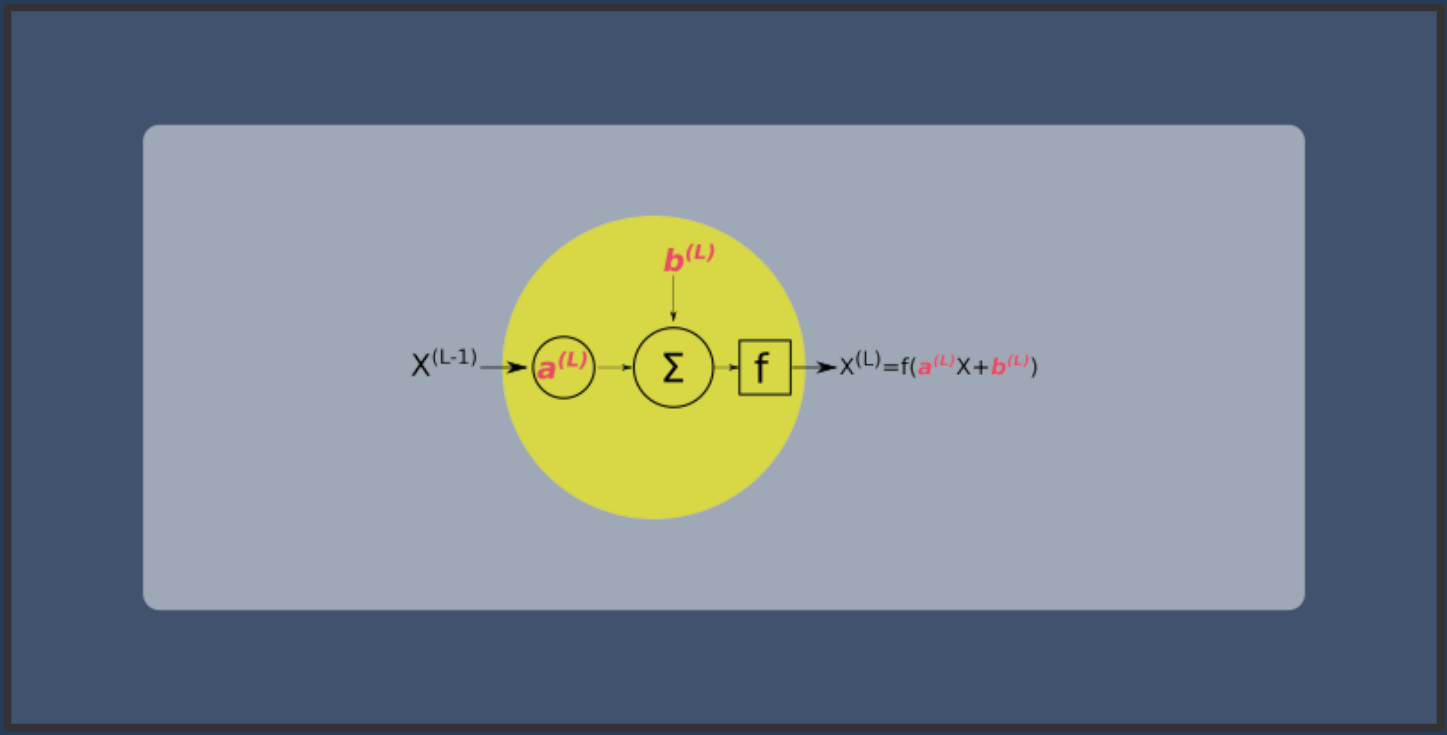
THE LOSS FUNCTION GRADIENT

We want to find weights that minimize C

The gradient of C is the following:

$$\nabla C = \begin{vmatrix} \frac{\partial C}{\partial a^{(1)}} \\ \frac{\partial C}{\partial b^{(1)}} \\ \dots \\ \frac{\partial C}{\partial a^{(L)}} \\ \frac{\partial C}{\partial b^{(L)}} \end{vmatrix}$$

ARTIFICIAL NEURON PARAMETERS?



Derivatives are computed starting from the last layer L of the graph

$$\frac{\partial C}{\partial a^{(L)}} = \frac{\partial}{\partial a^{(L)}}[(x^{(L)} - Y_{ref})^2]$$
$$= \frac{\partial}{\partial a^{(L)}}[(f(ax^{(L-1)} + b) - Y_{ref})^2]$$
$$= 2x^{(L-1)} f'(ax^{(L-1)} + b)(f(ax^{(L-1)} + b) - Y_{ref})$$
$$= 2x^{(L-1)} f'(ax^{(L-1)} + b)(x^{(L)} - Y_{ref})$$

$$\text{and} \frac{\partial C}{\partial b^{(L)}} = \frac{\partial}{\partial b^{(L)}}[(x^{(L)} - Y_{ref})^2]$$
$$= 2f'(ax^{(L-1)} + b)(x^{(L)} - Y_{ref})$$

IN SHORT

TRANING CONSIST IN

1. Initialize the weights
2. For each batches of the training dataset, do:
    - Compute the output
    - Compute the averaged loss gradient (over the batch)
    - Move a bit the weights

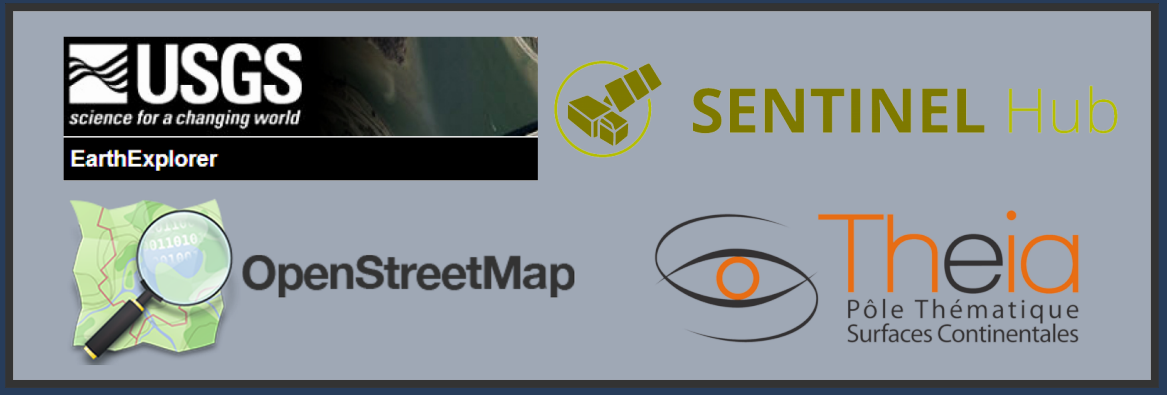
Usually, step 2 is repeated multiple times over the training dataset.
This is called one epoch.

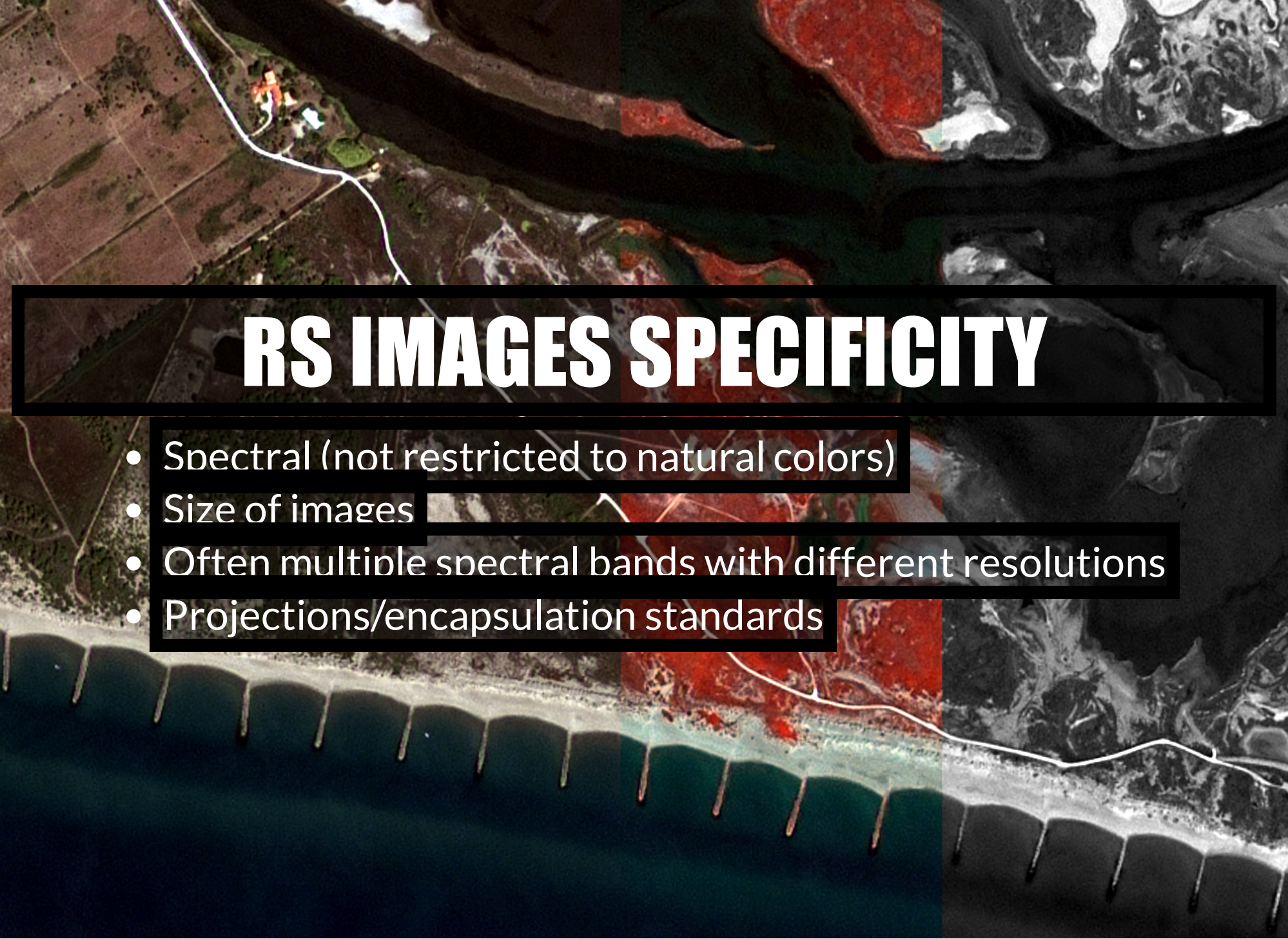
REMOTE SENSING

EARTH OBSERVATION CONTEXT

MORE AND MORE...

- Free available images (USGS, ESA/Copernicus, In France: THEIA/GEOSUD)
- Sensors (optical, SAR)
- High resolution (spatial, spectral, temporal)
- Crowdsourcing (OSM)





RS IMAGES SPECIFICITY

- Spectral (not restricted to natural colors)
- Size of images
- Often multiple spectral bands with different resolutions
- Projections/encapsulation standards

NEEDS

- Generic, multi-purpose, deep learning oriented tools for Remote Sensing
- Enable the use of deep nets on images without limitation on (1) number of sources, (2) dimension of sources
- User-oriented, that integrates naturaly in existing frameworks
- Enforcing geospatial standards

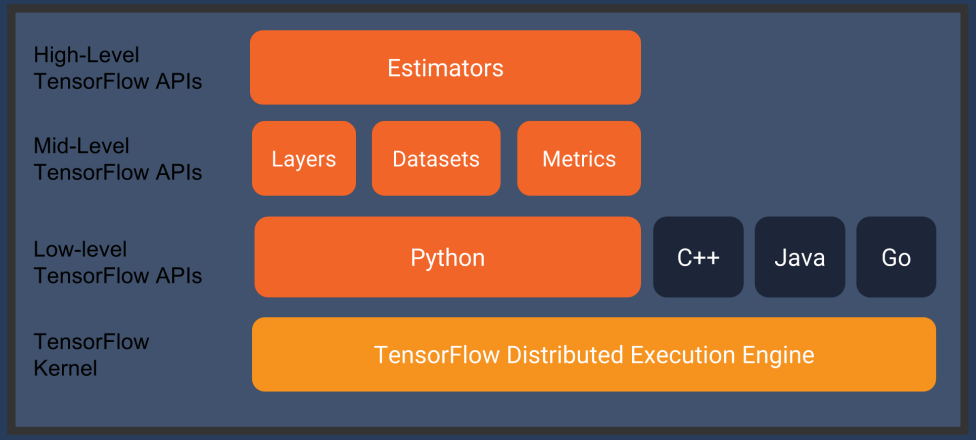
GUTS

ORFEO TOOLBOX

- Open-source, created by CNES
- Can process very large images
- Enforces geospatial standards
- ... Ok, we already know what is is! ;)

TENSORFLOW

- Open source, created by Google
- Scalable on heterogeneous architectures

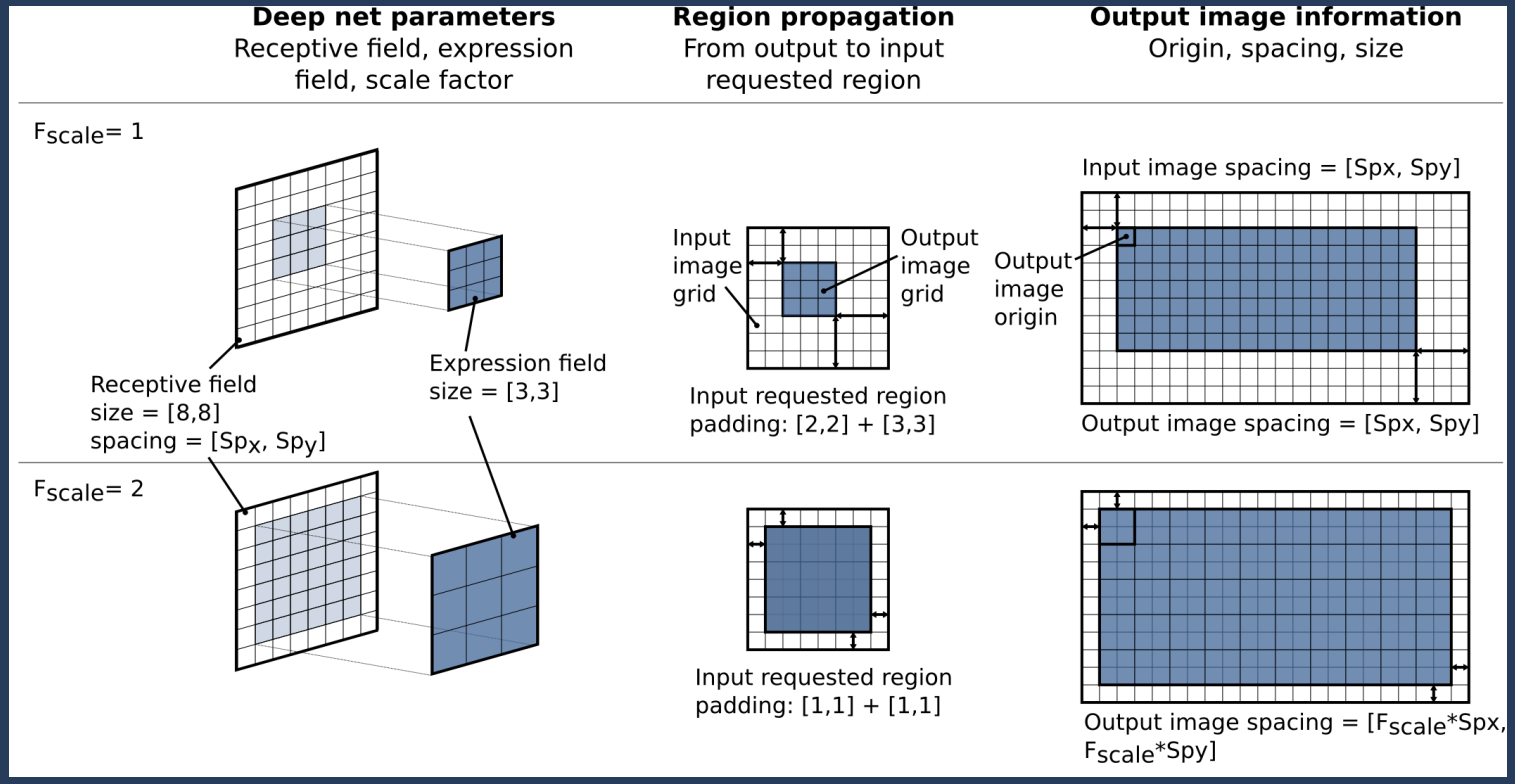


KEY CONCEPTS IN OTBTF

- Receptive fields
- Expression fields
- Spacing ratio
- *exact* model serving

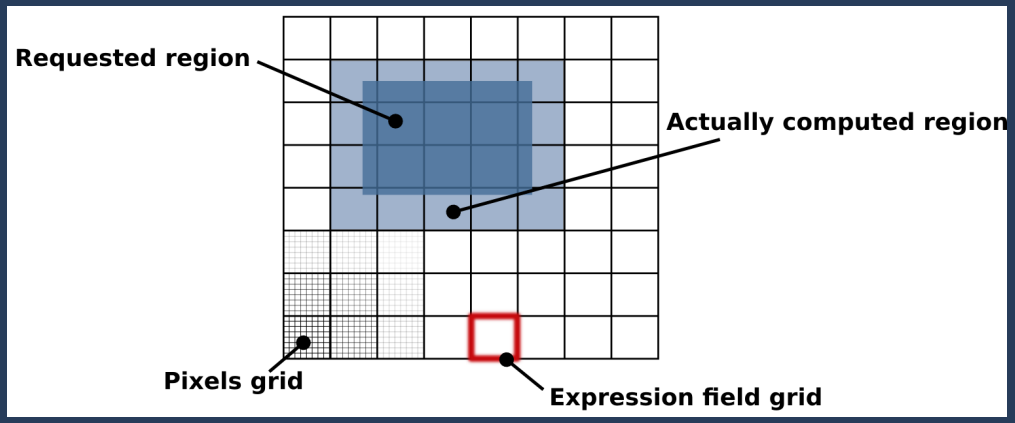
KEY CONCEPTS IN OTBTF

- Receptive fields
- Expression fields
- Spacing ratio
- *exact* model serving

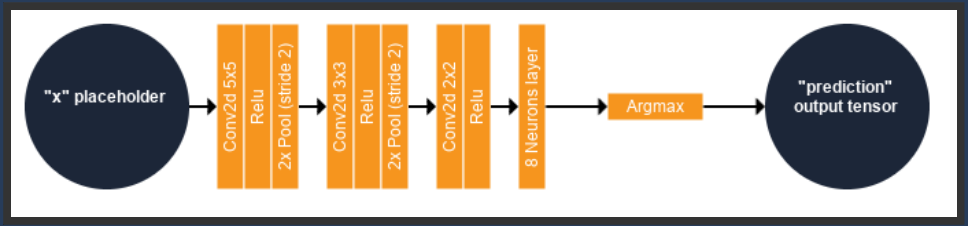

Deep net parameters
Receptive field, expression field, scale factor
Region propagation
From output to input requested region
Output image information
Origin, spacing, size
Fscale= 1
Receptive field size = [8,8] spacing = [SpX, SpY]
Expression field size = [3,3]
Input image grid
Output image grid
Input requested region padding: [2,2] + [3,3]
Input image spacing = [Spx, Spy]
Output image origin
Output image spacing = [Spx, Spy]
Fscale= 2
Input requested region padding: [1,1] + [1,1]
Output image spacing = [Fscale*Spx, Fscale*Spy]

KEY CONCEPTS IN OTBTF

- Receptive fields
- Expression fields
- Spacing scale
- *exact* model serving


Requested region
Actually computed region
Pixels grid
Expression field grid

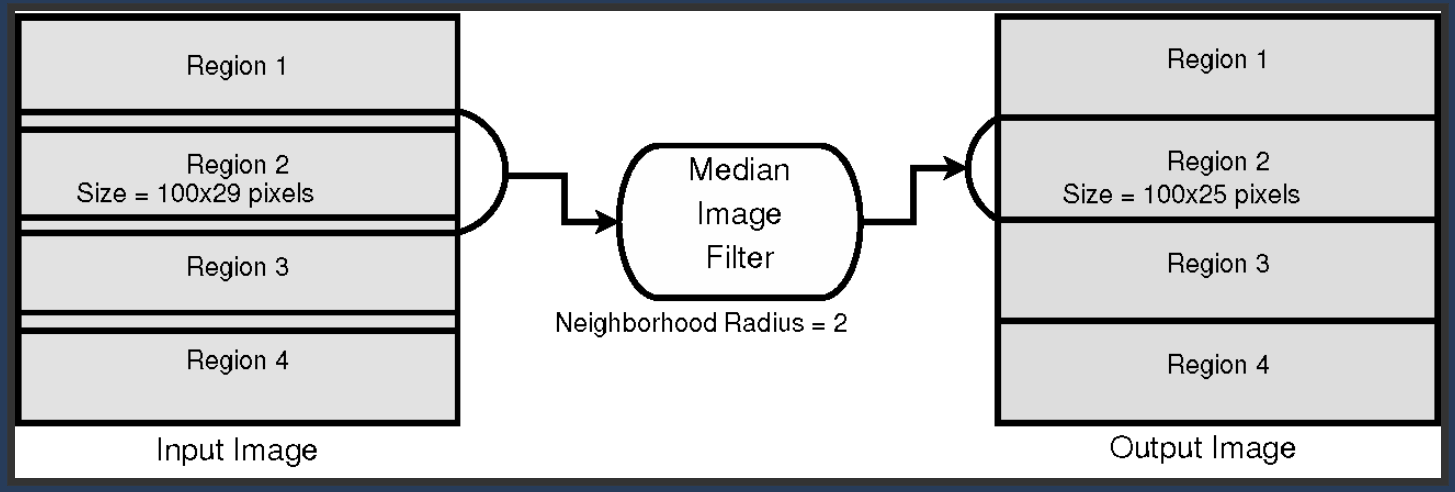
EXAMPLE



1. input : spcscale=1, size=n
2. conv 5x5 stride 2: spcscale=1*2=2, size=[(n-4)/2]
3. conv 3x3 stride 2: spcscale=1*2*2=4, size=[([(n-4)/2]-2)/2]
4. conv 2x2 stride 1: spcscale=1*2*2*1=4, size=[([(n-4)/2]-2)/2]-1

Receptive field: 16, Expression field: 1, Spacing scale: 4

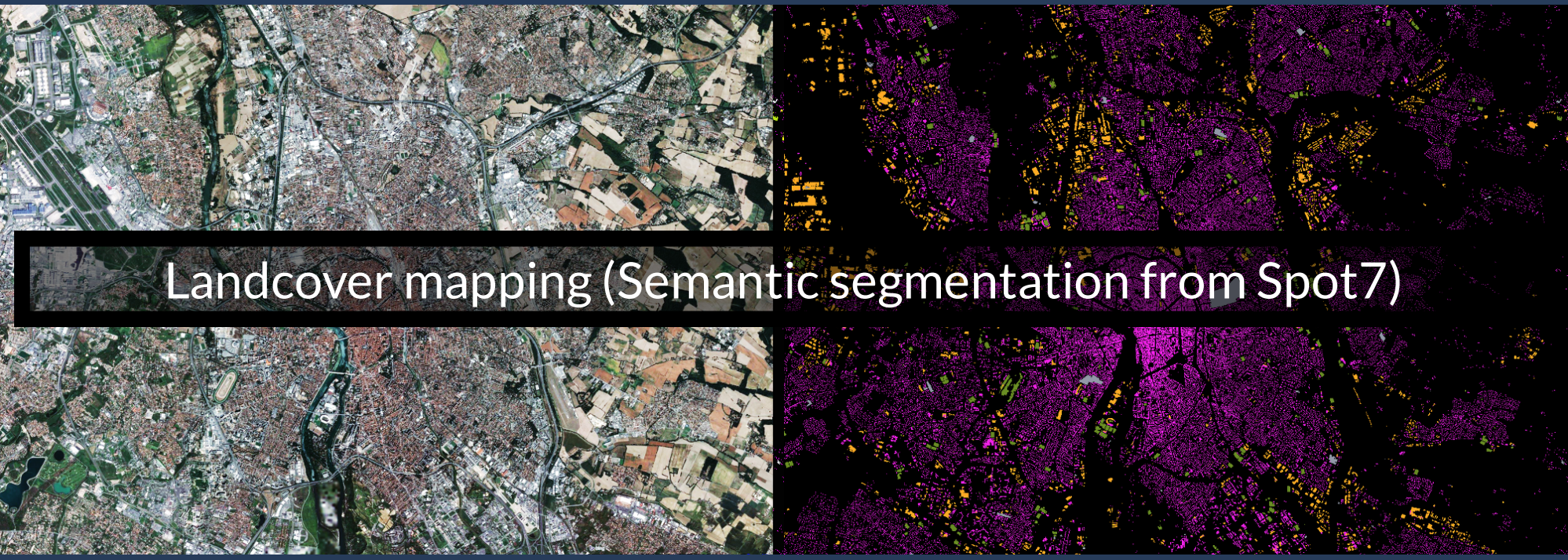
STREAMING


Region 1
Region 2
Size = 100x29 pixels
Region 3
Region 4
Input Image
Median
Image
Filter
Neighborhood Radius = 2
Region 1
Region 2
Size = 100x25 pixels
Region 3
Region 4
Output Image

A FEW EXAMPLES

Directly transposed from computer vision, applied to real world remote sensing images and geospatial data


Image translation (Spot7 to OSM)


Landcover mapping (Semantic segmentation from Spot7)

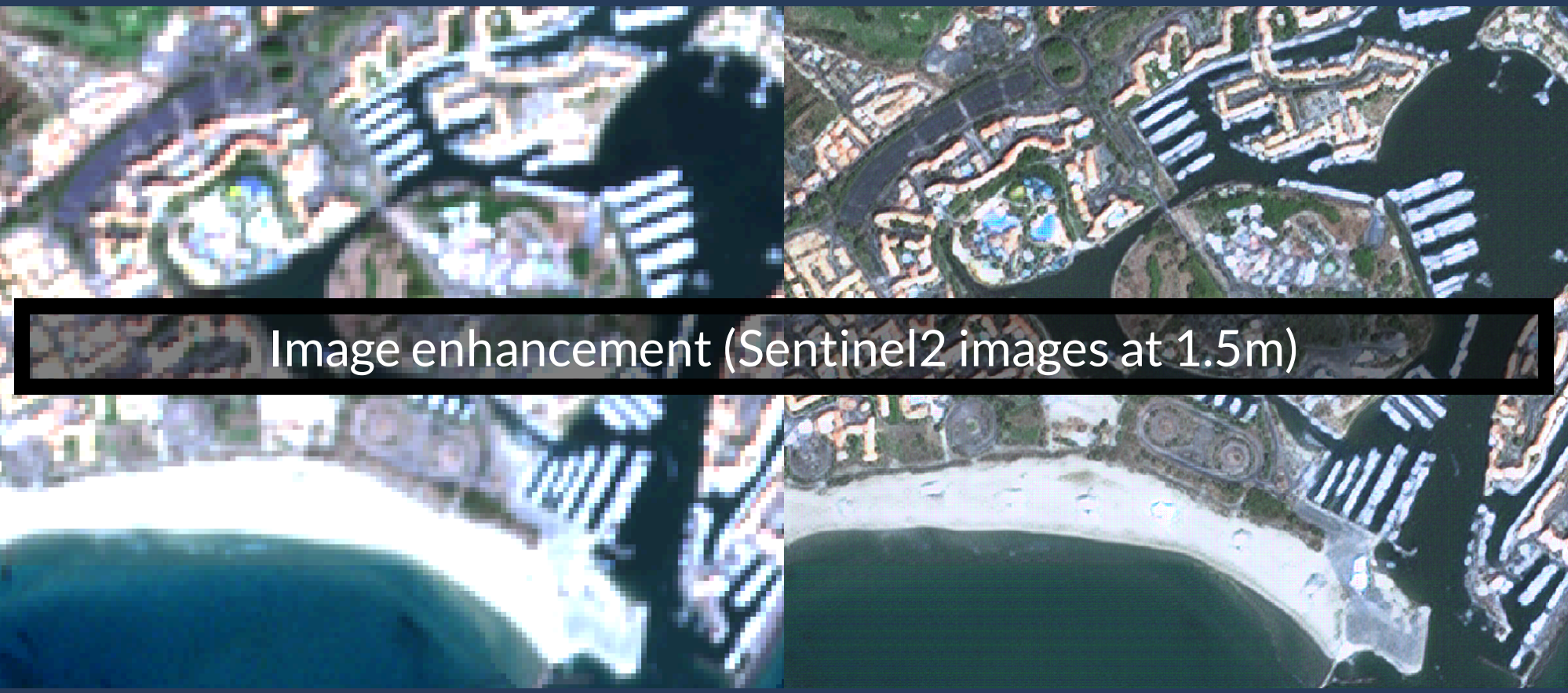
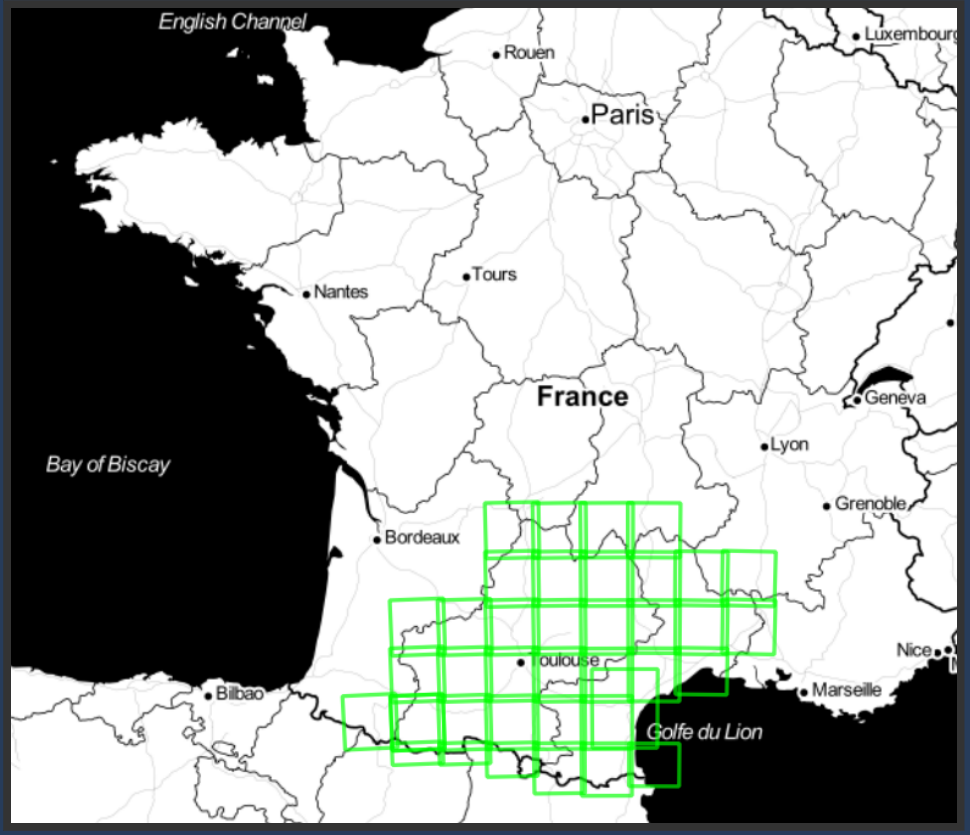


Image enhancement (Sentinel2 images at 1.5m)

APPLICATION: OCCITANIE BUILDING MAP

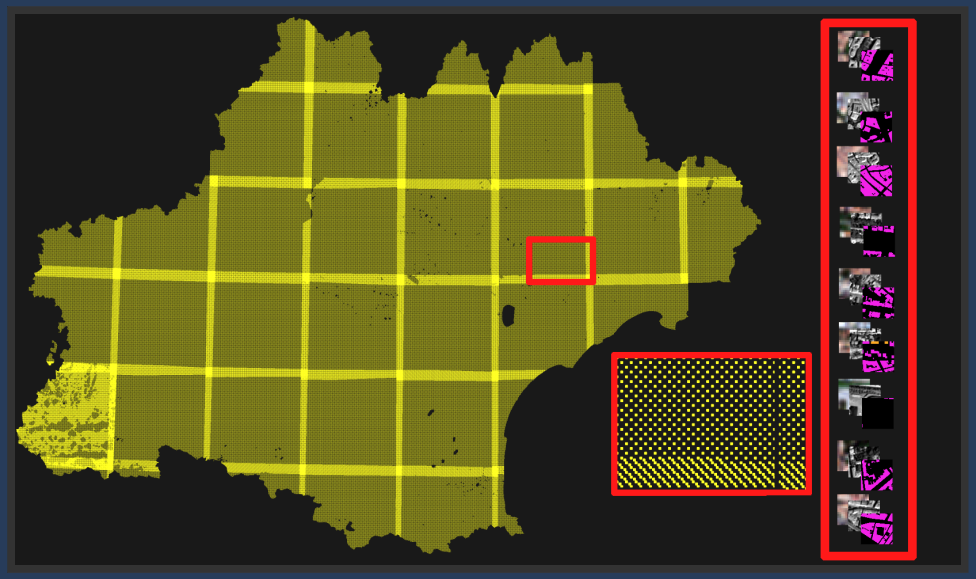
GOAL

Urban mapping from Spot-6/7 images over a $72.10^3 km^2$ area



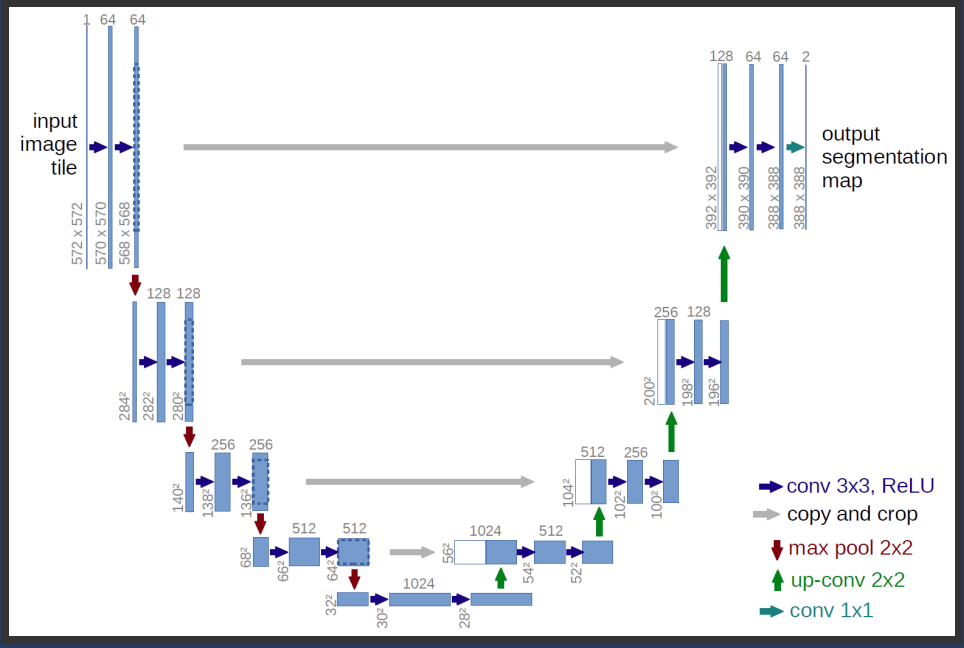
DATA

- 40 Spot-6/7 images: 1 Panchromatic channel@1.5m, 1 multispectral channel (Red, Green, Blue, Near-Infrared) @6m.
- Terrain truth: Geographic National Institute *BDTOPO* database



NET ARCHITECTURE

U-Net like



LOSS FUNCTION

CROSS ENTROPY

$$H(y, p) = -\sum_{i} y_i log(p_i)$$

Used when the output is a probability distribution

OUTPUT NORMALIZATION

SOFTMAX FUNCTION

$$p_i = \frac{e_{X_i}}{\sum_{k=1}^{N} e^{X_k}}$$

PROCESSING

- Sampling
- Training
- Mapping

SCORES

Validation

1. Residentiel
2. Commercial, Industriel et Agricole
3. Terrains de sports
4. Cimetières
5. Autre

Classe	Précision	Rappel	F-score
1	73.5	69.6	71.4
2	71.7	57.6	63.9
3	79.5	43.4	56.1
4	84.0	68.8	75.6
5	99.9	99.8	99.8
Overall	**77.3**	**69.1**	**72.9**

COMPETITORS

Urbain regroupé

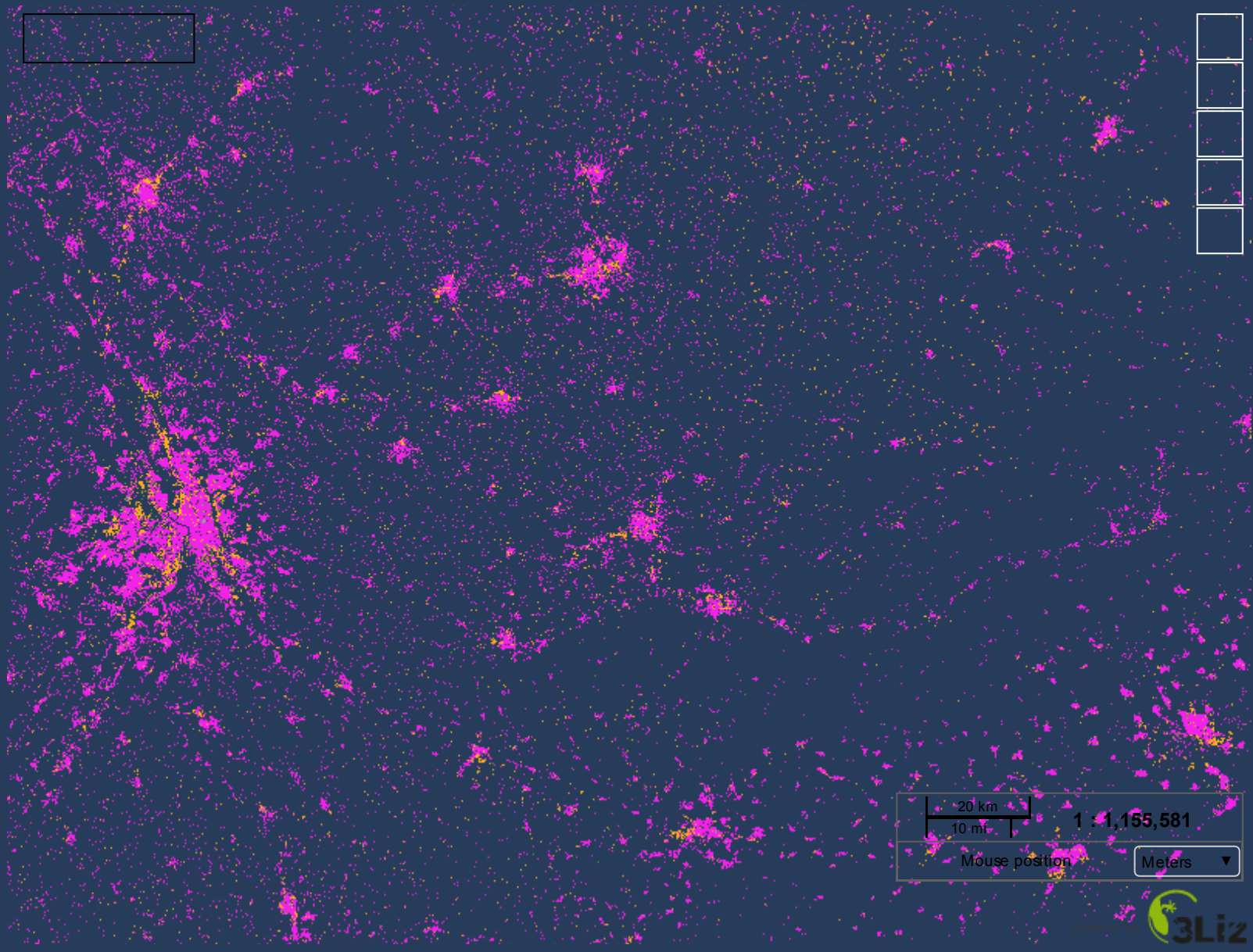
produit	GHSL	OSO	TETIS
precision	31.7	17.6	**80.0**
recall	88.8	**95.6**	73.2
f-score	46.7	29.7	**76.4**

Residentiel

produit	GHSL	OSO	TETIS
precision	-	15.8	**76.0**
recall	-	**76.0**	70.3
f-score	-	23.2	**73.1**

Industriel et commercial

produit	GHSL	OSO	TETIS
precision	-	10.1	**73.8**
recall	-	**82.7**	64.5
f-score	-	18.0	**68.8**


20 km
10 mi
1 : 1,155,581
Mouse position
Meters
3Liz

APPLICATION: SENTINEL-2 IMAGE ENHANCEMENT

GOAL

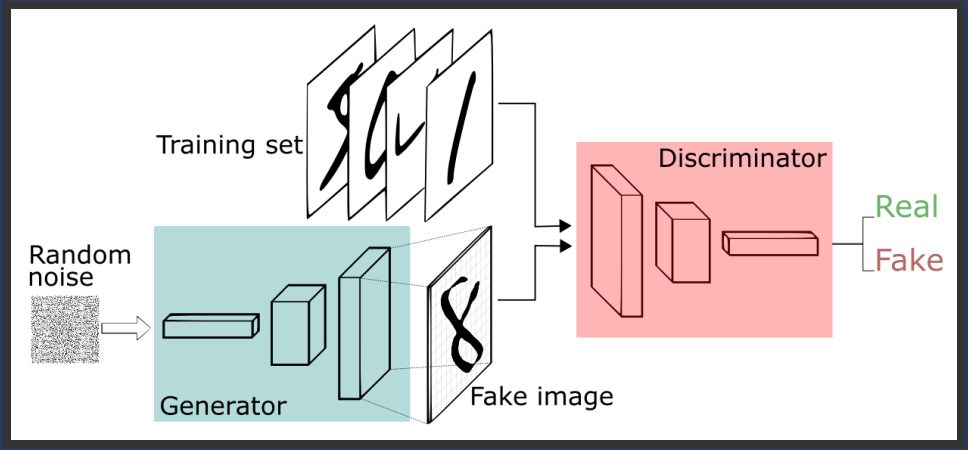
Sentinel-2 images are 10m spacing.

Spot-7 images are 1.5m spacing.

Let's train a deep net to transform Sentinel images into Spot!

Credit: original SRGAN paper (Ledig et Al.)

GANS EXPLAINED



Credit: Thalles Silva

- Generative model, G
- Discriminative model, D

GAN OBJECTIVE

- Training is accomplished in 2 steps: first, descend gradient of Obj1 loss function, then Obj2 loss function
- Obj1: G must mimics the data distribution of learning samples to fool D. Let *m* samples of noise *z* and data *x*:

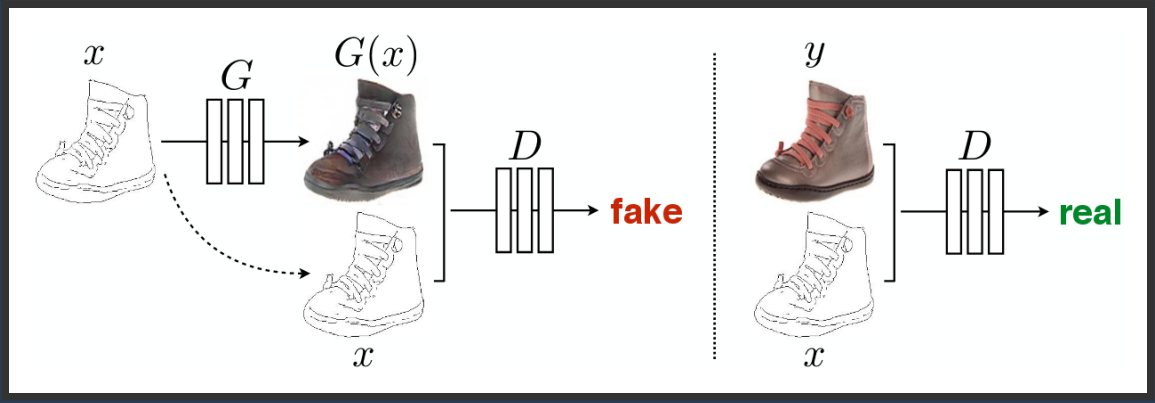
$$C_G = \frac{1}{m}\sum_{i=1}^{m}[logD(x_i) + log(1 - D(G(z_i)))]$$

- Obj2: D must find whether its input if a fake (from G) or geniune

$$C_D = \frac{1}{m}\sum_{i=1}^{m} log(1 - D(G(z_i)))$$

CGANS

CGANs are GANs but instead of random noise *z*, a conditional information *y* is used.



Credit: Isola et Al. (Pix2pix)

SUPER-RESOLUTION USING CGANS

- In SRGAN (Ledig et Al.) *y* is the low resolution image, and *x* is the high resolution image
- Obj1. is enriched with a *L2* loss (i.e. the squared mean error between) and a *perceptual loss* that is computed from layers of VGG19 trained on ImageNet. Those two losses compares generated and genuine HD images.

PROCESSING

- Sampling
- Training
- HR image generation

Computations performed on 1 NVIDIA GTX1080Ti GPU

SAMPLING

TRAINING

- 6000 samples
- Low res patches: 64x64
- High res patches: 256x256
- 50 epochs, training duration ~11h

HR IMAGE GENERATION

- Output HR Sentinel-2 image size: 72000x72000
- About 2h30 to produce one image

Let's see some results...


Original (left) Enhanced (right)

Thank you

Read more at

https://mdl4eo.irstea.fr/