THIS TALK IS NOT ABOUT DEEP LEARNING...

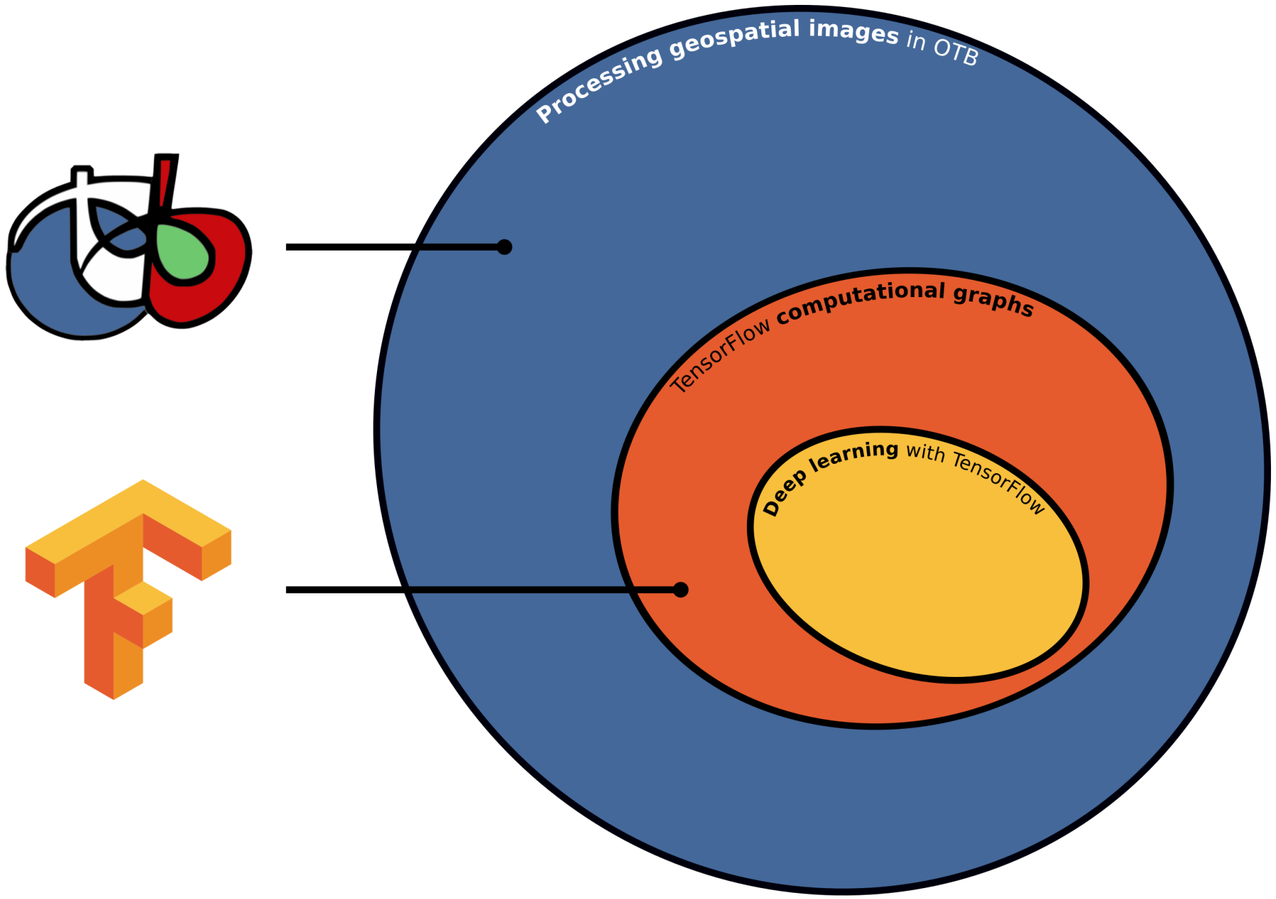
Rémi Cresson[1], Nicolas Narçon[1]

(1) French National Research Institute for Agriculture, Food and the Environment (INRAE)

OTB Users Days, 29 Nov. 2021, Toulouse, Fr.

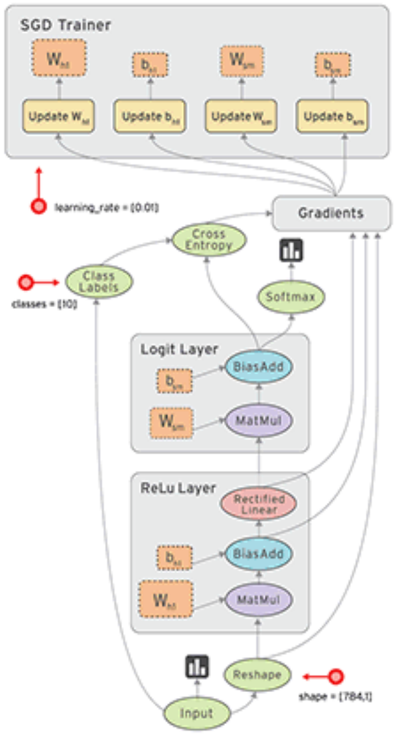
...IT IS ABOUT...

RUNNING TENSORFLOW COMPUTATIONAL GRAPHS INSIDE OTB!



TENSORFLOW GRAPHS

HOW IT WORKS



Source: the TensorFlow website

INSIDE A TENSORFLOW MODEL

EXAMPLE: SCALAR PRODUCT


```
import tensorflow as tf

# Input
x1 = tf.keras.Input(shape=[None, None, None], name="x1")
x2 = tf.keras.Input(shape=[None, None, None], name="x2")

# Compute scalar product
y = tf.reduce_sum(tf.multiply(x1, x2), axis=-1)
```

TENSORFLOW

PROS

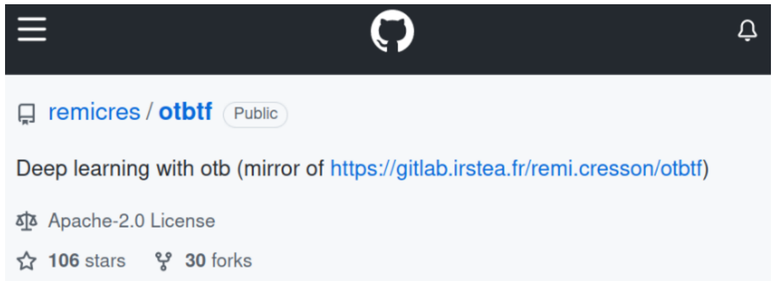
- Highly optimized
- Can be run on CPUs, GPUs, TPUs without any code change
- Since TensorFlow 2.0: easy to use (thanks, Keras!)

CONS

- Heavyweight library (~10 hours to compile)

THE OTBTF REMOTE MODULE

RUNNING TENSORFLOW GRAPHS IN OTB SINCE 2018



https://github.com/remicres/otbtf

IN SHORT

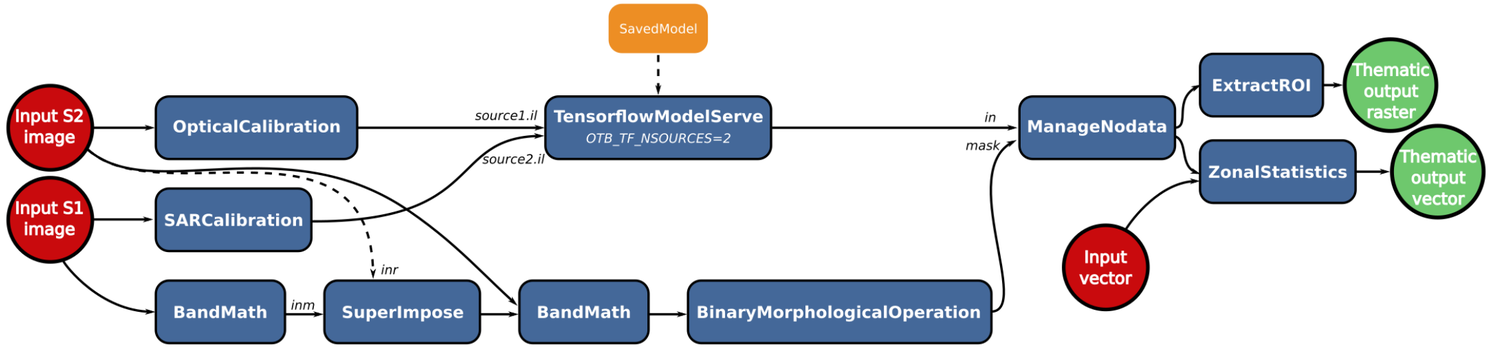
- Developped at **INRAE** to put in production research results
- Uses the **C++** TensorFlow API
- Enables to apply **TF models** on rasters in OTB **filters** and **applications**
- Preserves the **streaming** in OTB pipelines

NEW OTB APPLICATIONS

- **TensorflowModelServe**: Inference
- **TensorflowModelTrain**: Training/validation (great for educational purpose)
- **PatchesExtraction**: extract patchs in one or multiple rasters
- **TrainClassifierFromDeepFeatures**: connects **TensorflowModelServe** to **TrainImageClassifier**
- **ImageClassifierFromDeepFeatures**: connects **TensorflowModelServe** to **ImageClassifier**
- **PatchesSelection**: for patches selection from rasters (experimental)
- **LabelImageSampleSelection**: select patches from a label image (experimental)
- **DensePolygonClassStatistics**: same as **PolygonClassStatistics** but faster (experimental)

IT PROVIDES STREAMABLE PIPELINES

A KEY FEATURE TO GO LARGE SCALE!



The raster part is a typical pipeline used in convolutional neural networks

AND SOME PYTHON CLASSES FOR DATA SCIENTISTS

Say that you have generated patches images with the PatchesExtraction app:

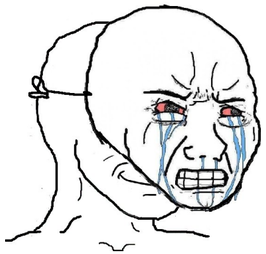
```
otbcli_PatchesExtraction -vec myvec.gpkg -field fid \
-source1.patchsizex 64 -source1.patchsizey 64       \
-source1.il raster_x.tif -source1.out x1.tif        \
-source2.patchsizex 64 -source2.patchsizey 64       \
-source2.il raster_y.tif -source2.out y1.tif
```

You can use otbtf.DatasetFromPatchesImages to feed your training process:

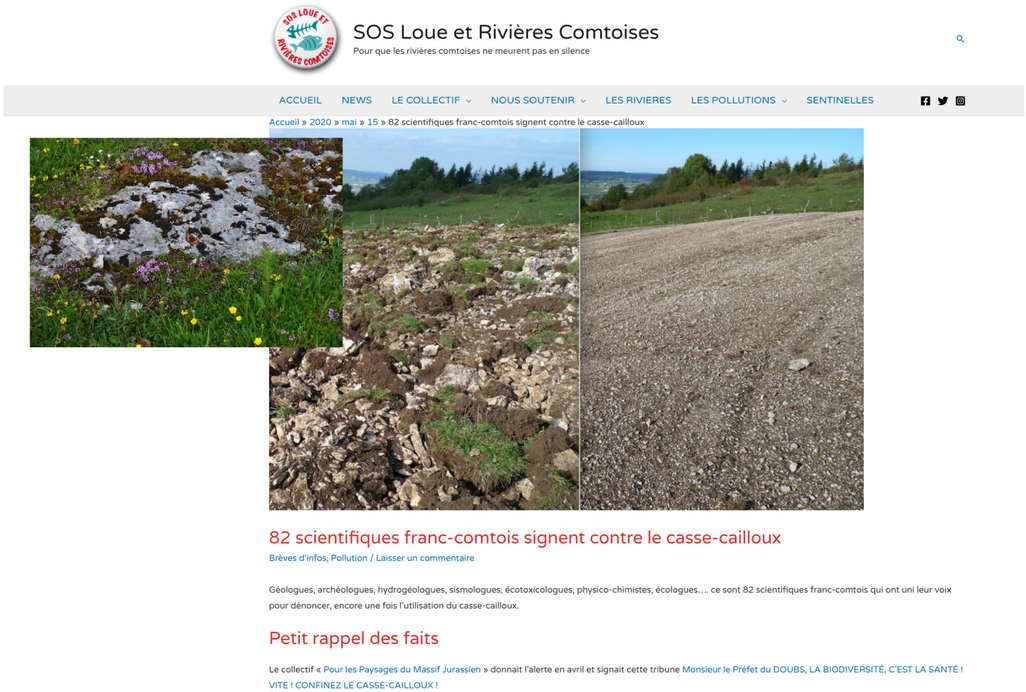
```
import otbtf
files_dict={"x": ["x1.tif", ..., "xN.tif"], "y": ["y1.tif", ..., "yN.tif"]}
ds = otbtf.DatasetFromPatchesImages(filenames_dict=files_dict, use_streaming=True)
tf_ds = ds.get_tf_dataset(batch_size=8)
```

EXAMPLES

WARNING: DEEP-LEARNING INVOLVED!



ROCKS MAPPING



SOS Loue et Rivières Comtoises

Pour que les rivières comtoises ne meurent pas en silence

ACCUEIL NEWS LE COLLECTIF NOUS SOUTENIR LES RIVIERES LES POLLUTIONS SENTINELLES

Accueil » 2020 » mai » 15 » 82 scientifiques franc-comtois signent contre le casse-cailloux

82 scientifiques franc-comtois signent contre le casse-cailloux

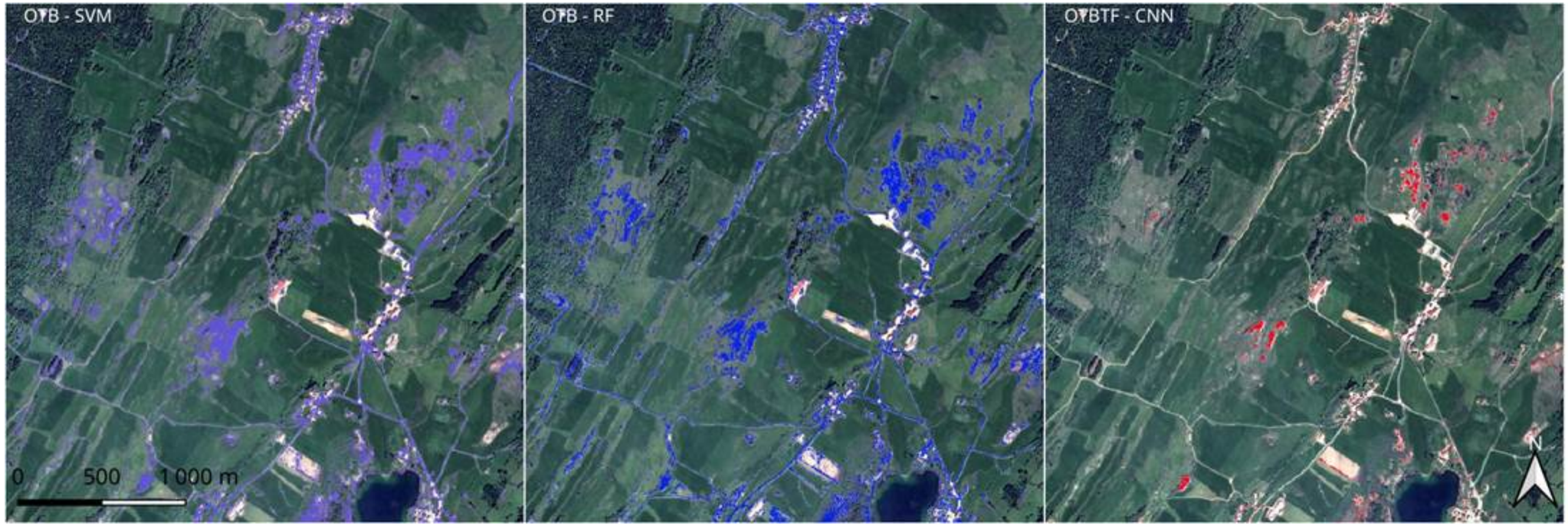
Brèves d'infos, Pollution / Laisser un commentaire

Géologues, archéologues, hydrogéologues, sismologues, écotoxicologues, physico-chimistes, écologues…. ce sont 82 scientifiques franc-comtois qui ont uni leur voix pour dénoncer, encore une fois l'utilisation du casse-cailloux.

Petit rappel des faits

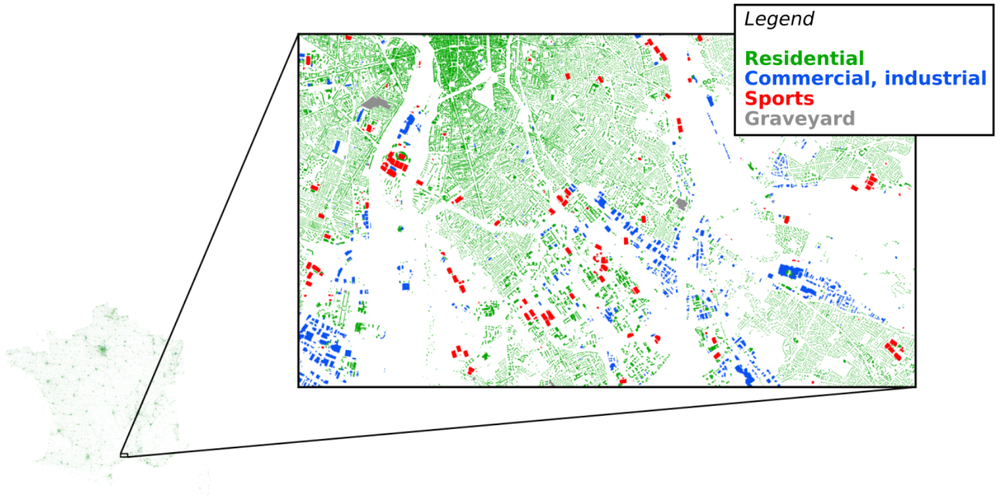
Le collectif « Pour les Paysages du Massif Jurassien » donnait l'alerte en avril et signait cette tribune Monsieur le Préfet du DOUBS, LA BIODIVERSITÉ, C'EST LA SANTÉ ! VITE ! CONFINEZ LE CASSE-CAILLOUX !

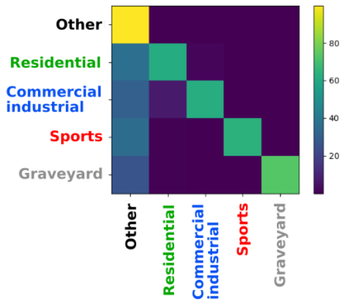
Source: https://www.soslrc.com/2020/05/15/82-scientifiques-franc-comtois-signent-contre-le-casse-cailloux



LARGE SCALE LAND COVER MAPPING

SEMANTIC SEGMENTATION OF BUILDINGS FOOTPRINT OVER FRANCE MAINLAND AT 1.5M SPACING





	Pre.	Rec.	F-Sc.
Other	99.7	99.7	99.7
Residential	66.2	61.9	64.0
Com.& ind.	62.5	62.0	62.3
Sports	51.3	62.7	56.4
Graveyard	77.8	72.8	75.2

https://www.theia-land.fr/en/product/buildings-footprint/

SUPER-RESOLUTION

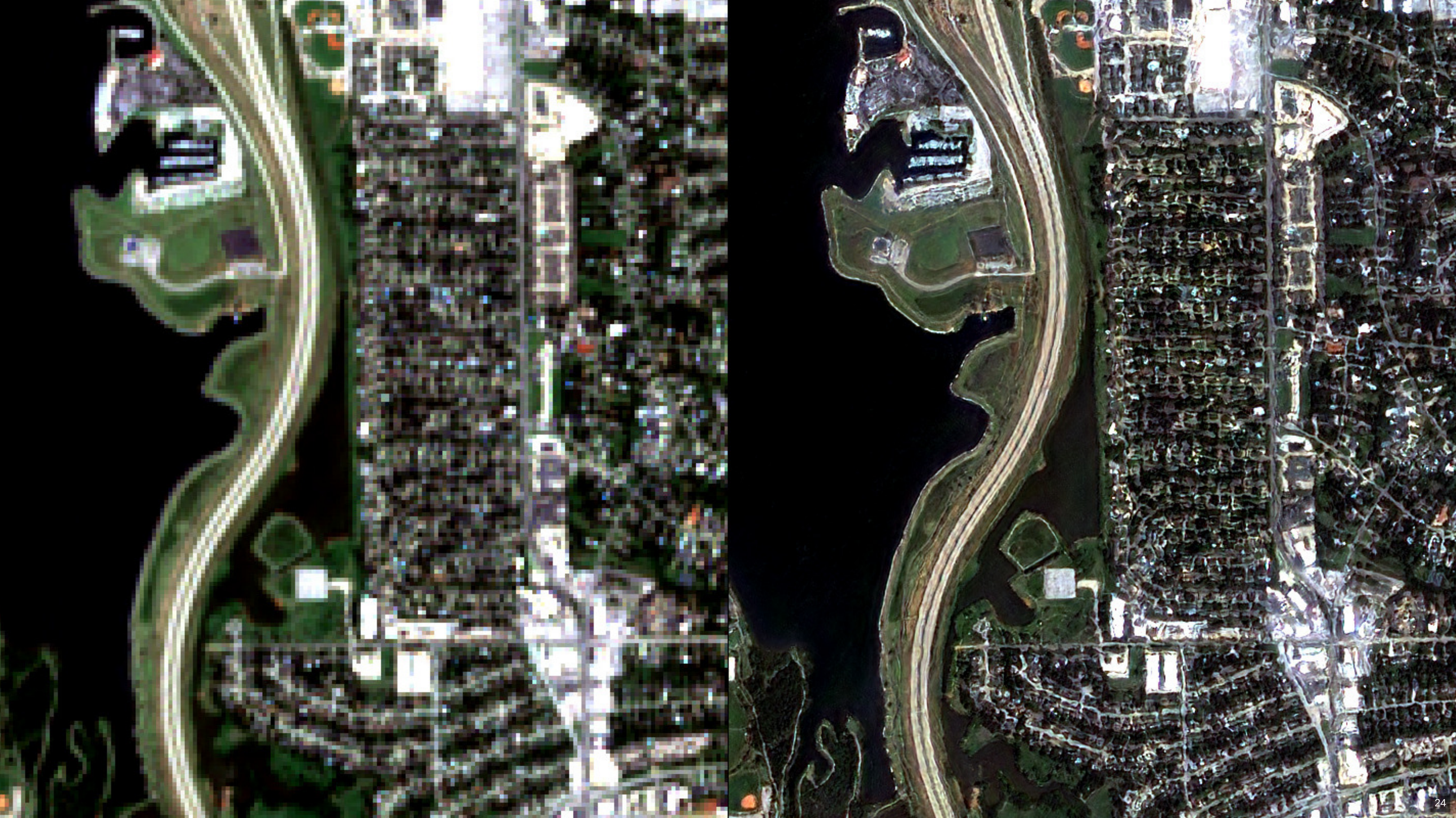
https://github.com/remicres/sr4rs

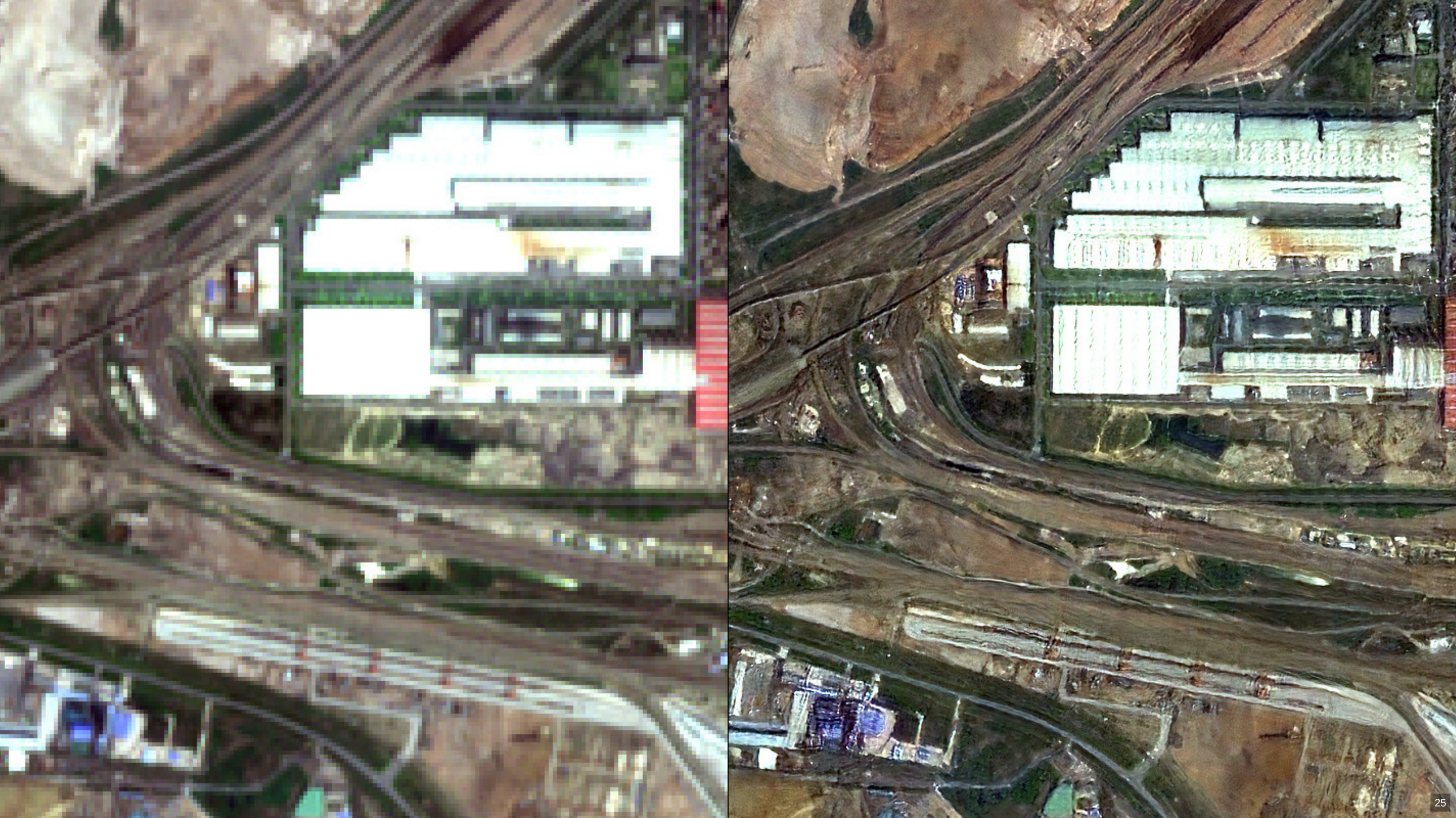
EASY TO RUN

```
# Download the pre-trained SR4RS model
wget https://tinyurl.com/sr4rsmodelv2
unzip sr4rsmodelv2

# Run SR4RS over a Sentinel-2 image (Bands 4, 3, 2, 8)
python sr4rs/code/sr.py \
--savedmodel sr4rs_sentinel2_bands4328_france2020_savedmodel \
--input S2_image_channels_4328_10m.tif \
--output S2_SR.tif
```

INFERENCE USING OTB, FULLY STREAMABLE

```
if __name__ == "__main__":
    # Some secondary stuff to retrieve parameters values, etc
    ...

    # The important stuff
    infer = otbApplication.Registry.CreateApplication("TensorflowModelServe")
    infer.SetParameterStringList("source1.il", [params.input])
    infer.SetParameterInt("source1.rfieldx", rfield)
    infer.SetParameterInt("source1.rfieldy", rfield)
    infer.SetParameterString("source1.placeholder", constants.lr_input_name)
    infer.SetParameterString("model.dir", params.savedmodel)
    infer.SetParameterString("model.fullyconv", "on")
    infer.SetParameterStringList("output.names", [ph])
    infer.SetParameterInt("output.efieldx", efield)
    infer.SetParameterInt("output.efieldy", efield)
    infer.SetParameterFloat("output.spcscale", ratio)
    infer.SetParameterInt("optim.tilesizex", efield)
    infer.SetParameterInt("optim.tilesizey", efield)
    infer.SetParameterInt("optim.disabletiling", 1)
    infer.SetParameterString("out", out_fn)
    infer.SetParameterOutputImagePixelType("out", encoding)
    infer.ExecuteAndWriteOutput()
```

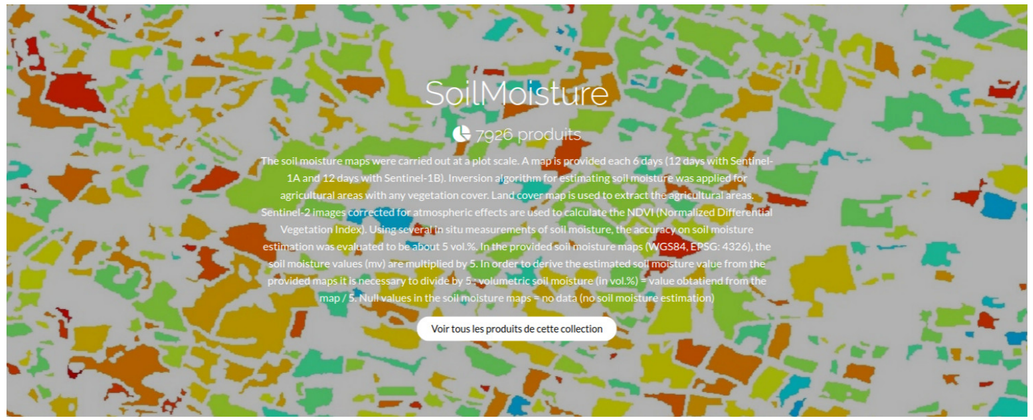




SOIL MOISTURE MAPPING (THEIA PRODUCT)

APPROX. 20 THEIA PRODUCTS/MONTH. MORE THAN 6K AVAILABLE PRODUCTS AROUND THE GLOBE.

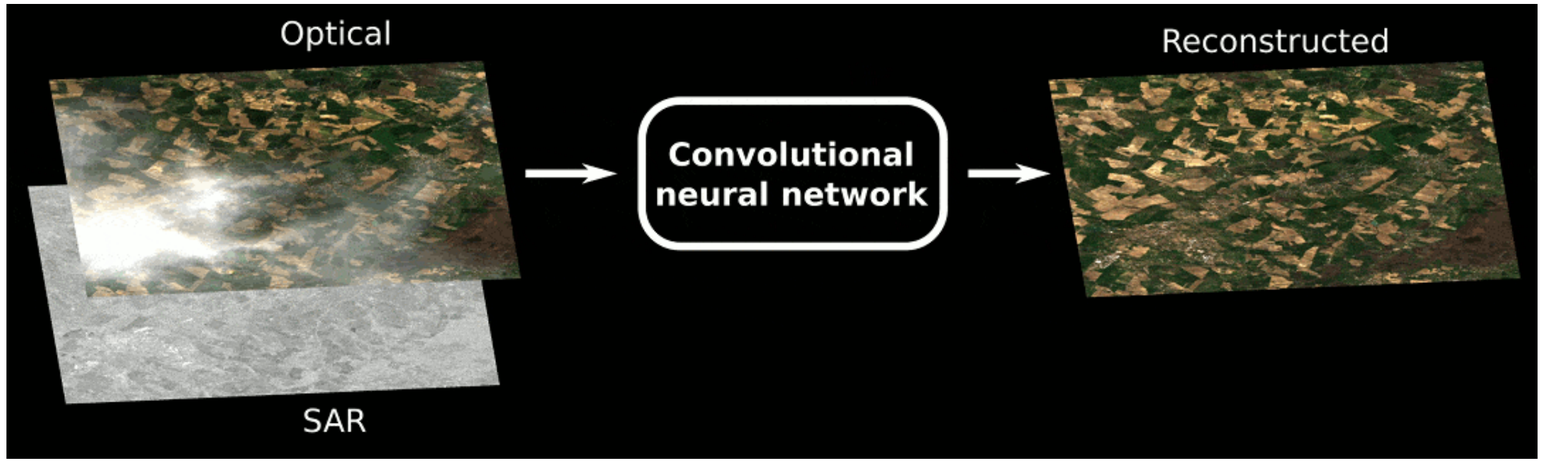
SAR backscattering model inversion using ANN from S1 and S2



https://www.theia-land.fr/product/humidite-du-sol-a-tres-haute-resolution-spatiale/

CLOUD REMOVAL IN OPTICAL IMAGES

FROM JOINT SAR/OPTICAL IMAGES



decloud toolbox (available soon on github.com/cnes and gitlab.inrae.fr)

Left: original Sentinel-2 image. Right: reconstructed

Left: original Sentinel-2 image. Right: reconstructed

LIVE DEMO!

```
OTB_TF_NSOURCES=3 otbcli_TensorflowModelServe                                  \
-source1.il input/s1a_31TCJ_vvvh_DES_110_20171021txxxxxx_from-10to3dB.tif      \
-source1.placeholder "tower_0/s1_t_cloud" -source1.rfieldx 256 -source1.rfieldy 256 \
-source2.il input/SENTINEL2A_20171023-105154-753_L2A_T31TCJ_C_V2-2_FRE_10m.tif \
-source2.placeholder "tower_0/s2_t_cloud" -source2.rfieldx 256 -source2.rfieldy 256 \
-source3.il input/T31TCJ.tif                                                   \
-source3.placeholder "tower_0/dem" -source3.rfieldx 128 -source3.rfieldy 128  \
-model.dir meraner_unet_savedmodel/ -model.fullyconv on                       \
-output.efieldx 128 -output.efieldy 128 -output.names tower_0/s2_estim_pad64  \
-out "S2A_RESTORED_20171023_toulouse.tif?&box=6500:7300:1024:1024" int16
```

4 years of reconstructed optical time series (Occitanie area)

EXAMPLES

TENSORFLOW GRAPHS WITHOUT VARIABLES

GENERATING SAVEDMODEL

USE CASE: SCALAR PRODUCT

The above graph computes the scalar product between pixels of input images x1 and x2

```
import tensorflow as tf

x1 = tf.keras.Input(shape=[None, None, None], name="x1")  # Input image 1
x2 = tf.keras.Input(shape=[None, None, None], name="x2")  # Input image 2

# Compute scalar product
y = tf.reduce_sum(tf.multiply(x1, x2), axis=-1)

# Create model
model = tf.keras.Model(inputs={"x1": x1, "x2": x2}, outputs={"y": y})
model.save("scalprod_x1x2_savedmodel")
```

RUN THIS MODEL

```
OTB_TF_NSOURCES=2 otbcli_TensorflowModelServe            \
-source1.il S2A_RESTORED_20171023_toulouse.tif           \
-source2.il S2A_RESTORED_20171023_toulouse.tif           \
-model.dir scalprod_x1x2_savedmodel/ -model.fullyconv on \
-out scalprod_x1x2_output.tif -optim.disabletiling on
```

Since OTBTF 3.0, input names are automatically resolved (in **alphabetical order**).

USER PLACEHOLDERS

USE YOUR OWN CONSTANTS IN YOUR GRAPH

In the above graph, x is an input for the raster, and a in an input constant

```
import tensorflow as tf

x = tf.keras.Input(shape=[None, None, None], name="x")  # Input image
a = tf.keras.Input(shape=[], name="a")                  # Input constant

# Compute scalar product
y = tf.reduce_sum(tf.multiply(x, a), axis=-1)

# Create model
model = tf.keras.Model(inputs={"x": x, "a": a}, outputs={"y": y})
model.save("scalprod_ax_savedmodel")
```

RUN THIS MODEL

```
otbcli_TensorflowModelServe                                  \
-source1.il S2A_RESTORED_20171023_toulouse.tif               \
-model.userplaceholders "a=(1.0, 2.0, 2.0, 1.0)"             \
-model.dir scalprod_ax_savedmodel/ -model.fullyconv on \
-out scalprod_ax_output.tif -optim.disabletiling on
```

Note the expression for constants (aka **model.userplaceholders**). Since OTBTF 3.0 they can also be vectors!

BANDMATHX VS TENSORFLOW: A FEW BENCHMARKS (CPU)

ROUND #1 "ADD {1, 1, 1, 1}"

BandMathX: **75s** (w/o. write: **43s**)

```
otbcli_BandMathX                                                    \
-il input/SENTINEL2A_20171020-104323-598_L2A_T31TCJ_C_V2-2_FRE_10m.tif \
-exp "im1+{1; 1; 1; 1}"                                             \
-out tmp_out/plus1_bandmathx.tif
```

TensorflowModelServe: **51s** (**19s**)

```
otbcli_TensorflowModelServe                                         \
-source1.il input/SENTINEL2A_20171020-104323-598_L2A_T31TCJ_C_V2-2_FRE_10m.tif \
-model.fullyconv on -model.dir my_model/ -optim.disabletiling on    \
-out tmp_out/plus1_otbtf.tif
```

TF model:

```
y = x + tf.constant([1, 1, 1, 1], shape=[1, 1, 1, 4], dtype=tf.float32)
```

ROUND #2 "EUCLIDEAN NORM"

BandMathX: 53s (45s)

```
otbcli_BandMathX                                                    \
-il input/SENTINEL2A_20171020-104323-598_L2A_T31TCJ_C_V2-2_FRE_10m.tif \
-exp "sqrt(im1*im1')"                                               \
-out tmp_out/l2_bandmathx.tif
```

TensorflowModelServe: 21s (13s)

```
otbcli_TensorflowModelServe                                         \
-source1.il input/SENTINEL2A_20171020-104323-598_L2A_T31TCJ_C_V2-2_FRE_10m.tif \
-model.fullyconv on -model.dir my_model/ -optim.disabletiling on    \
-out tmp_out/l2_otbtf.tif
```

TF model:

```
y = tf.norm(x, axis=-1)
```

ROUND #3 "SCALAR PRODUCT"

BandMathX: 54s (46s)

```
otbcli_BandMathX                                                        \
-il input/SENTINEL2A_20171020-104323-598_L2A_T31TCJ_C_V2-2_FRE_10m.tif \
    input/SENTINEL2A_20171023-105154-753_L2A_T31TCJ_C_V2-2_FRE_10m.tif \
-exp "im1*im2'"                                                         \
-out tmp_out/ps_bandmathx.tif
```

TensorflowModelServe: 36s (28s)

```
OTB_TF_NSOURCES=2 otbcli_TensorflowModelServe                           \
-source1.il input/SENTINEL2A_20171020-104323-598_L2A_T31TCJ_C_V2-2_FRE_10m.tif \
-source2.il input/SENTINEL2A_20171023-105154-753_L2A_T31TCJ_C_V2-2_FRE_10m.tif \
-model.fullyconv on -model.dir my_model/ -optim.disabletiling on        \
-out tmp_out/ps_otbtf.tif
```

TF model:

```
y = tf.reduce_sum(tf.multiply(x1, x2), axis=-1)
```

ROUND #4 "MATRIX COMPUTATION"

BandMathX: 189s (157s)

```
otbcli_BandMathX                                                        \
-il input/SENTINEL2A_20171020-104323-598_L2A_T31TCJ_C_V2-2_FRE_10m.tif \
    input/SENTINEL2A_20171023-105154-753_L2A_T31TCJ_C_V2-2_FRE_10m.tif \
-exp "(im1*im2) mlt {1; 1; 1; 1}"                                       \
-out tmp_out/ps_bandmathx.tif
```

TensorflowModelServe: 83s (51s)

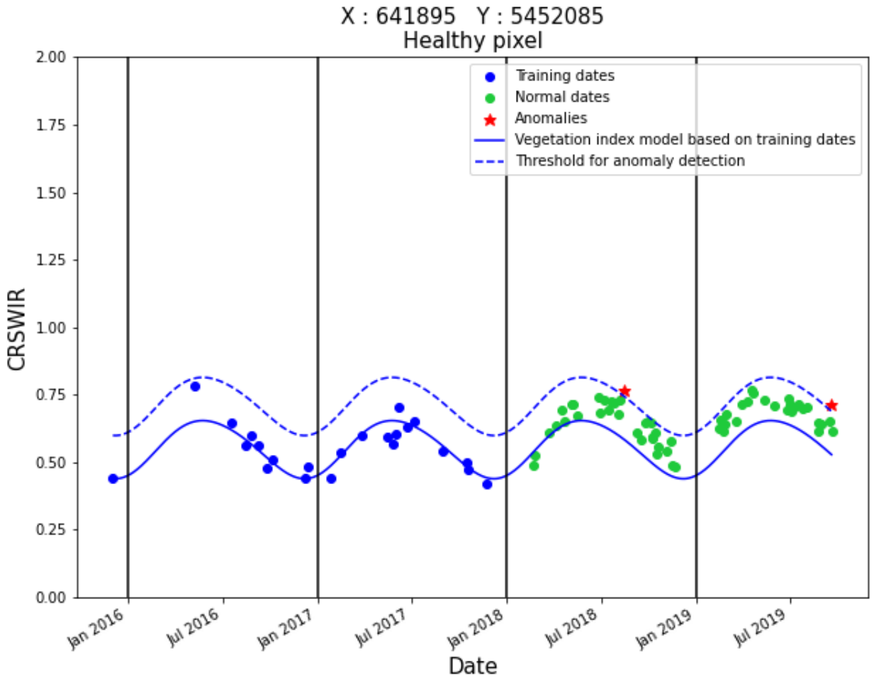
```
OTB_TF_NSOURCES=2 otbcli_TensorflowModelServe                                \
-source1.il input/SENTINEL2A_20171020-104323-598_L2A_T31TCJ_C_V2-2_FRE_10m.tif \
-source2.il input/SENTINEL2A_20171023-105154-753_L2A_T31TCJ_C_V2-2_FRE_10m.tif \
-model.fullyconv on -model.dir my_model/ -optim.disabletiling on             \
-out tmp_out/ps_otbtf.tif
```

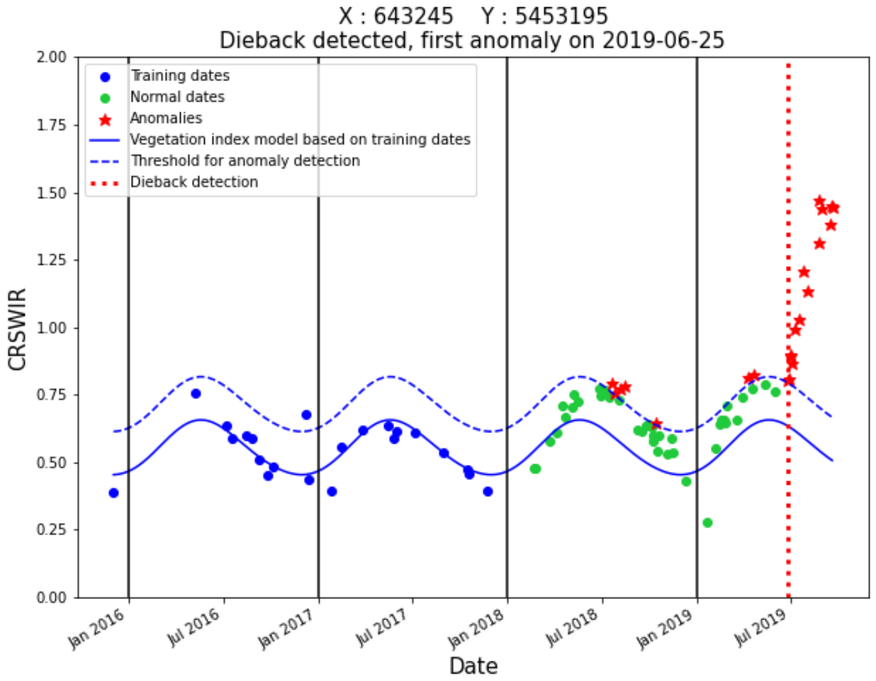
TF model:

```
dims = tf.shape(x1)
x1r = tf.reshape(x1, shape=[dims[0]*dims[1]*dims[2], dims[3], 1])
x2r = tf.reshape(x2, shape=[dims[0]*dims[1]*dims[2], 1, dims[3]])
y = tf.matmul(tf.matmul(x1r, x2r), tf.ones_like(x1r))
```

USE CASE

TIME SERIES HARMONIC MODELING





THE TENSORFLOW MODEL

```
import tensorflow as tf
import math

# Inputs
series = tf.keras.Input(shape=[None, None, None], name="ts")
dates = tf.keras.Input(shape=[], name="dates")

# Harmonic terms
pi = tf.constant(math.pi)
fac = tf.multiply(dates, 2.0 * pi / 365.25)
harmonic_terms = [tf.ones_like(fac), tf.math.sin(fac), tf.math.cos(fac), tf.math.sin(
A = tf.stack(harmonic_terms, axis=-1)

# No-data masks
masks = tf.cast(tf.math.greater(series, tf.zeros_like(series)), tf.float32)

# Reshape pixel blocks as matrices
N = tf.shape(dates)[0]
m = tf.shape(masks)[1] * tf.shape(masks)[2]
At = tf.transpose(A)
B = tf.reshape(series, shape=[m, N])
M = tf.reshape(masks, shape=[m, N])

# Avoid pixels with too few observations
```

RUN THIS MODEL

```
# Generate the SavedModel
python generate_fordead5params.py
```

```
# List the available rasters
files=$(find $DATADIR/VegetationIndex/ -type f | sort)
```

```
# Extract the dates
dates=$(echo $(echo "$files" | rev | cut -f1 -d_ | rev \
| cut -f1 -d. | xargs -i date -d {} +%j) | sed "s/ /.0,/g")
```

```
# Run the model
otbcli_TensorflowModelServe                          \
-source1.il $files                                   \
-model.userplaceholders "dates=($dates)"             \
-model.dir fordeadparams_model/ -model.fullyconv on  \
-out coefs.tif -optim.tilesizex 10000
```

CONCLUSION

TENSORFLOW CAN ALSO BE USED AS A BIG CALCULATOR

1. Use python to **build TensorFlow models**
2. Run TensorFlow model in a **streamable OTB application...**
3. ...that can be used in any **OTB pipeline** built with C++ or Python APIs

HACK WITH OTBTF 3.0

RELEASE CANDIDATE 2 PUSHED ON 25/11/2021

```
docker pull mdl4eo/otbtf3.0:cpu-basic     # no optimization
docker pull mdl4eo/otbtf3.0:cpu-basic-dev # no optimization + development files
docker pull mdl4eo/otbtf3.0:gpu           # NVIDIA GPUs enabled
docker pull mdl4eo/otbtf3.0:gpu-dev       # NVIDIA GPUs enabled + development files
```

Enjoy the hackaton on wednesday. Submit your issues on Github!

Happy OTB Users Days !