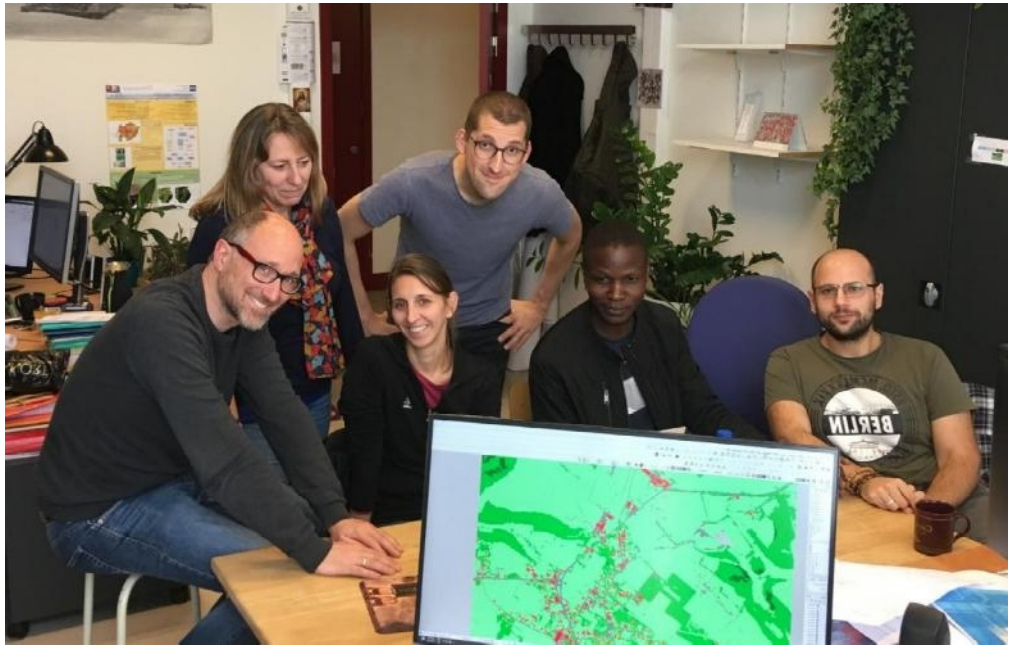
DEEP-LEARNING FOR VERY-HIGH RESOLUTION LAND-COVER MAPPING

A story about issues and difficulties of using deep learning

Who we are

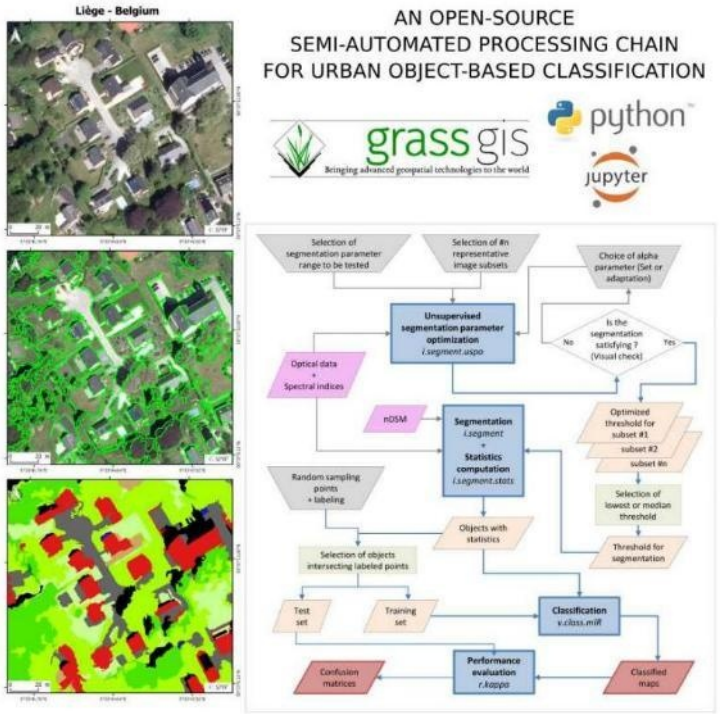
- ANAGEO research unit at ULB
- Specialized in remote sensing and GIS
- With one deep learning expert : Nicholus Mboga
- With guest star Vero Andreo from the Argentinian Council for Science and Technology (CONICET)



Moritz, Sabine, Vero, Taïs, Nick, Stef

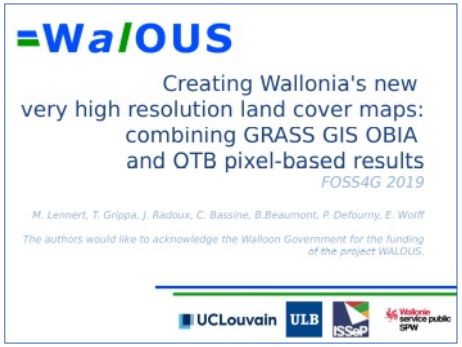
Who we are

- Used to OBIA techniques
- Development of toolchain in GRASS GIS
- Applied on both Belgian and African context
- Recent project : land cover and land use mapping for Wallonia



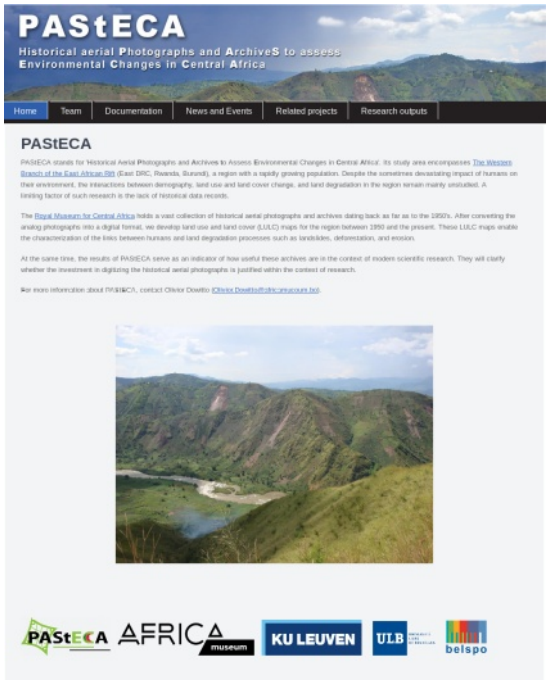
source : Grippa et al (2017), Remote Sensing, https://dx.doi.org/10.3390/rs9040358

⇨ See presentation in afternoon session :

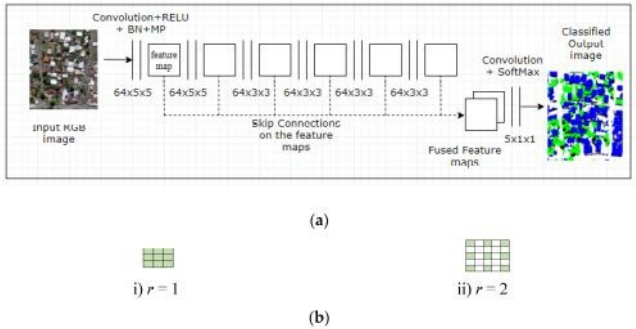


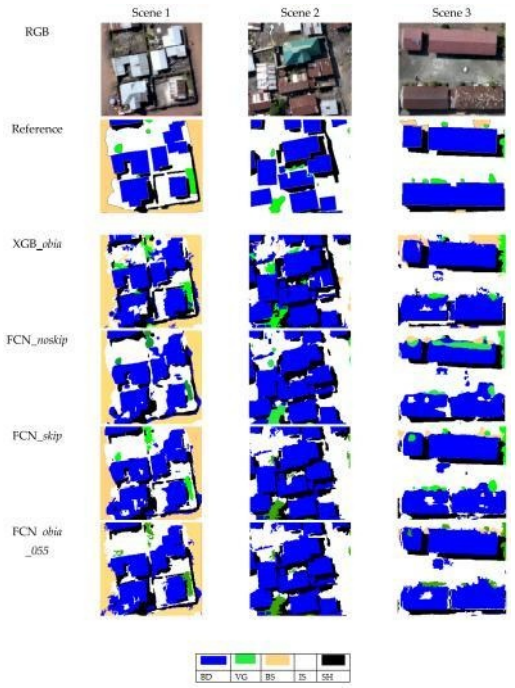
Who we are

- First contact with Deep Learning : land cover mapping using current and historical imagery
- Nick
  - developed own architecture
  - experimented with combinations OBIA + Deep Learning



http://pasteca.africamuseum.be/





Mboga, N. et al (2019), Fully Convolutional Networks and Geographic Object-Based Image Analysis for the Classification of VHR Imagery. Remote Sens. 2019, 11, 597.

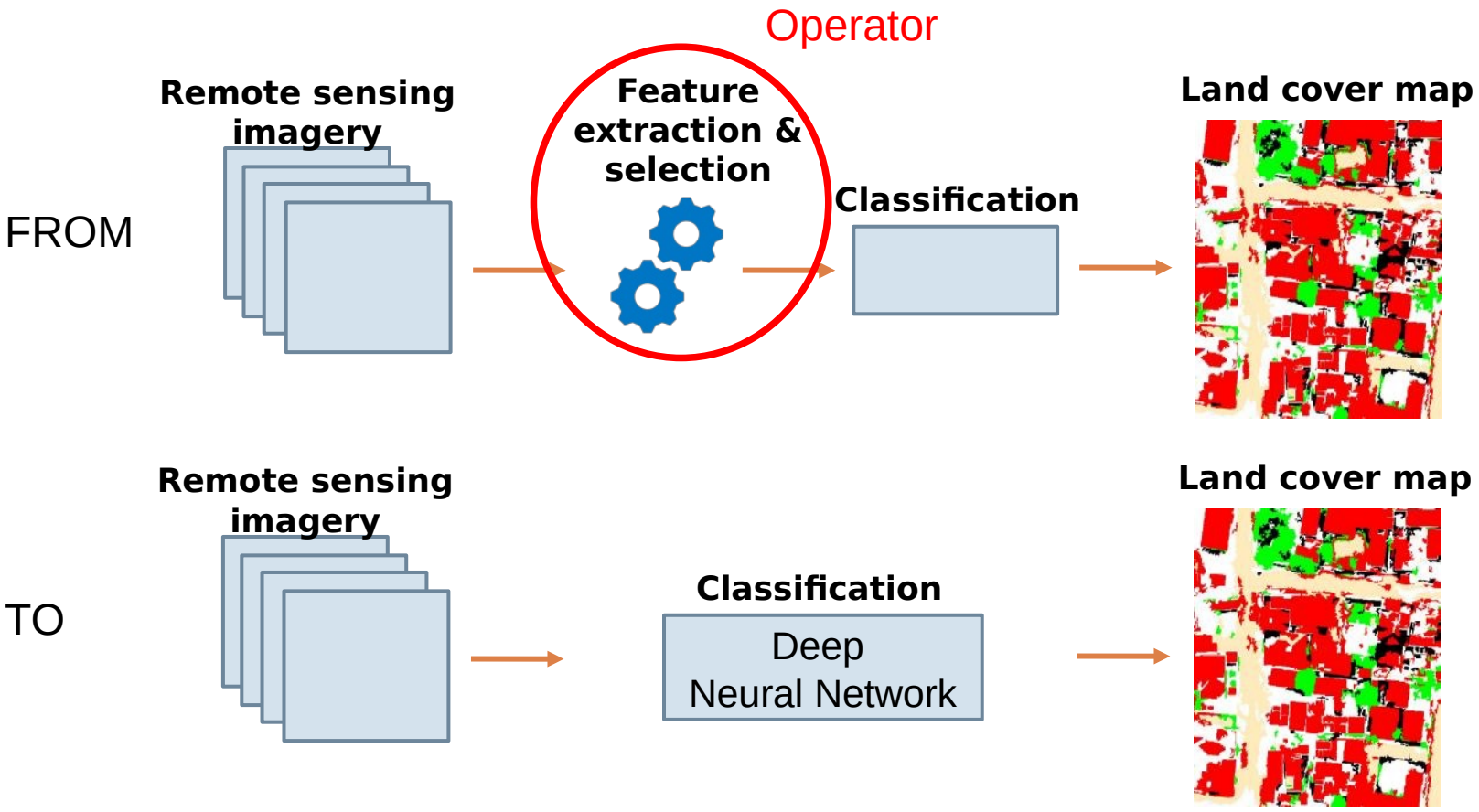
Why Deep Learning

ULB ANAGEO

- It's the current cool thing !
- Studies indicate significantly better performance
- It reduces the intervention of the operator
- Hope for better transferability

Why Deep Learning

- It reduces the intervention of the operator



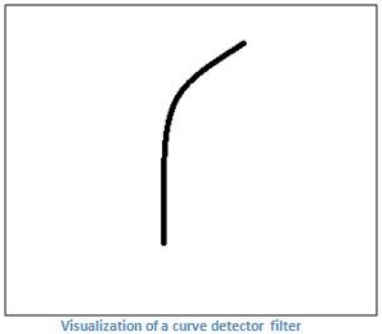
Why Deep Learning

- It reduces the intervention of the operator : <u>but how</u> ?
  - Deep Neural Network = layers of « neurons »
  - In image treatment « neurons » = convolutional filters
  - Convolutional filters defined by weights in each pixel
  - Network finds best values for these weights going back and forth between original imagery and training examples => network defines by itself the filters it needs

Example : curve detecting filtre

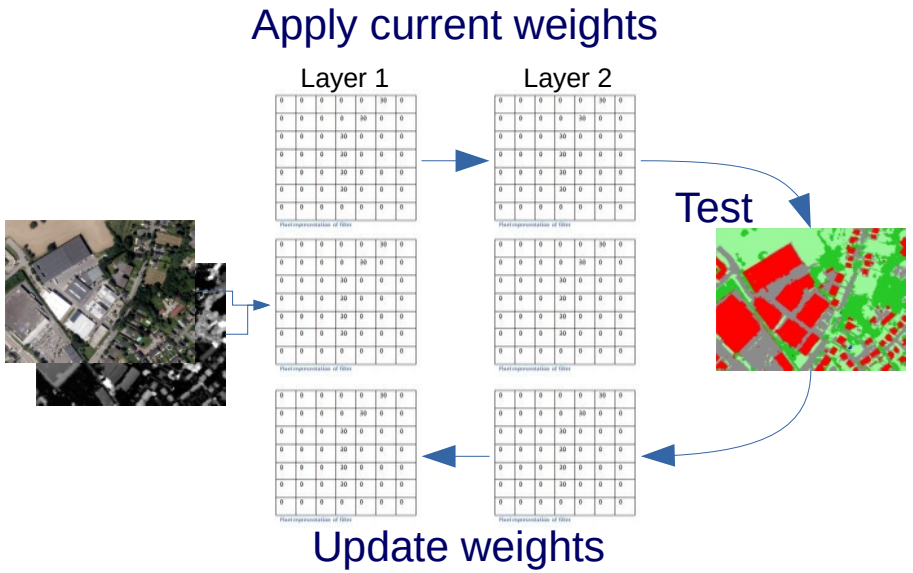
0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter



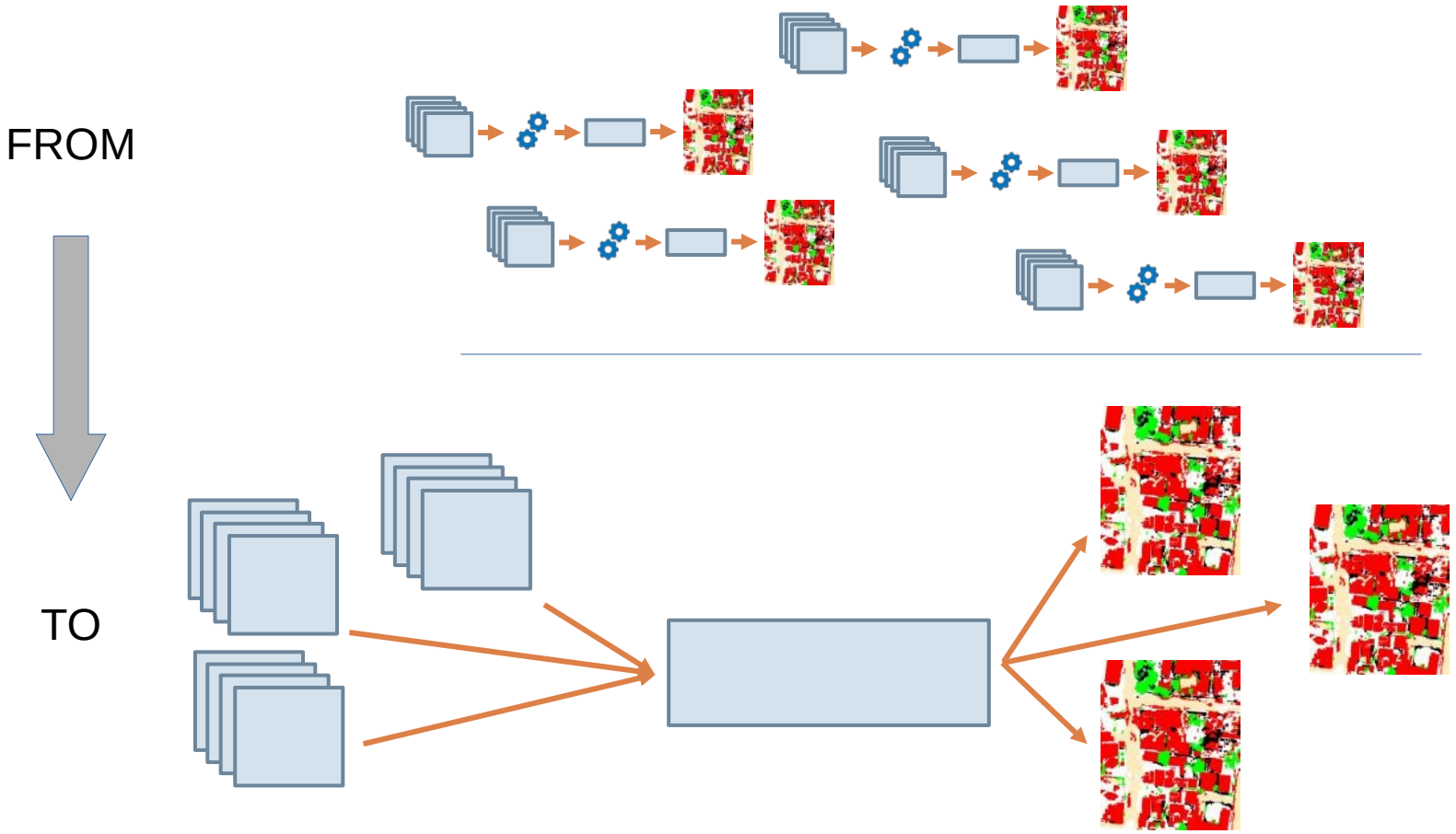
Visualization of a curve detector filter

https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/



Why Deep Learning

- Hope for better transferability



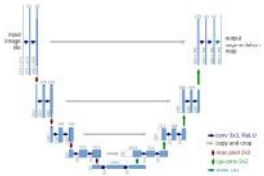
ANAGEO Hackathon

- One-week internal hackathon
- Our aims / constraints
  - learn deep learning
  - experiment existing architecture(s) on other imagery
  - play around with architecture
  - test transfer learning of trained network on different images
- What we did
  - use existing code from Nick and tried three networks : Nick's original net, a modified version, own implementation of Unet

    https://lmb.informatik.uni-freiburg.de/people/ronneber/u-net/

    Keras implentation :

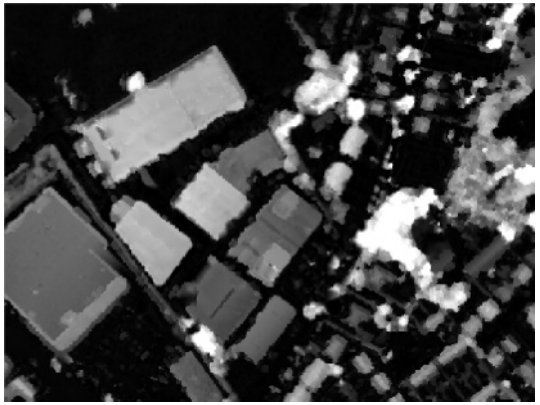
    https://gist.github.com/hlamba28/6073e8ef011a1d47080f67522018d7e4



  - trained and applied on Walloon imagery using WALOUS results as training data (see presentation this afternoon on WALOUS)
  - applied on Flanders and Cordoba imagery to test transferability

Data

- WALOUS : open data orthophotos (25cm, 16bit) + height (0,5m), labels from preliminary results of project
- Flanders : open data orthophotos (40cm, 8bit) + height (1m)
- Cordoba, Argentina : orthophotos (20cm, 8bit), no height information



Software

ULB ANAGEO

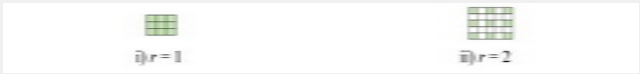
- Keras library using Tensorflow as backend
- Most code is about data preparation and general logistics (file i/o, bands to arrays, etc)
- Actual network definitions quite simple
- Architect of a network has to define :
  - Size of convolutions to apply (but not the weights!)
  - Number of filters per layer
  - Number of layers
  - Patch size & batch size at training
  - + a series of other possible parameters

Software

- Actual network definitions quite simple

Definition of a layer using Keras:

```
def block_1(inpt):
    x = inpt
    x_0 = ZeroPadding2D((4, 4))(x)
    x_0 = Conv2D(64, (5, 5), padding = 'valid', dilation_rate = (2,2))(x_0)
    x_0 = Activation ('relu')(x_0)
    x_0 = BatchNormalization()(x_0)
    x_0 = ZeroPadding2D((1, 1))(x_0)
    x_0 = MaxPooling2D(pool_size=(3, 3),strides=1)(x_0)
    return x_0
```



```
def block_2(inpt):
    x = inpt
    x_2 = ZeroPadding2D((2, 2))(x)
    x_2 = Conv2D(64, (3, 3), padding = 'valid', dilation_rate = (2,2))(x_2)
    x_2 = Activation ('relu')(x_2)
    x_2 = BatchNormalization()(x_2)
    x_2 = ZeroPadding2D((1, 1))(x_2)
    x_2 = MaxPooling2D(pool_size=(3, 3),strides=1)(x_2)
    return x_2
```

Software

- Actual network definitions quite simple

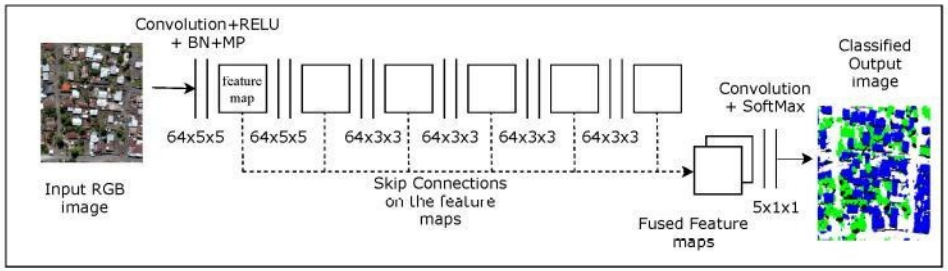
Combinations of layers into network (here : original network by Nick) :

```
x0=block_1(inp)
x1=block_1(x0)
x2=block_2(x1)
x3=block_2(x2)
x4=block_2(x3)
x5=block_2(x4)
x6=block_2(x5)

xc= Concatenate(axis = 3)([x0,x1,x2,x3,x4,x5,x6])

x7=Conv2D(nc, (1, 1))(xc)

out_p = Activation("softmax")(x7)
```



Results

- Different architectures

Original image | Nick's net without height | Nick's net with height | Unet

Results

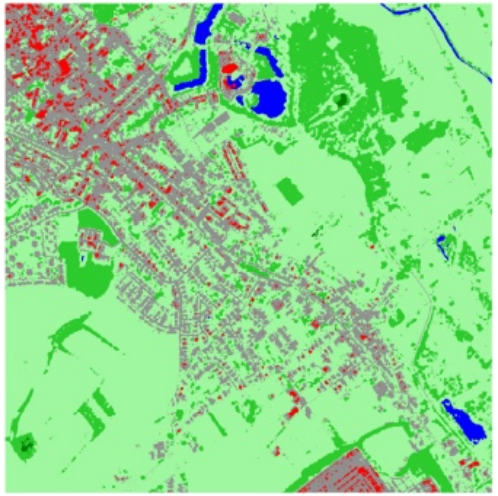
ULB ANAGEO

- Time spent during the hackathon (estimation) :
  - 30% => Data preprocessing
  - 10% => Setting up network architecture
  - 40 % => Training networks
  - 5 % => Predictions
  - 15% => Checking outputs and back-propagation of our ideas to run new models, and again…
- Computing times
  - Preparing training data : ~ 30 minutes (24 000 training + 6 000 validation patches of 64x64 pixels)
  - Training : ~ 200 minutes
  - Prediction : ~ 1 minute per 8000x8000 tile
- Accuracies : over 0,8 overall accuracy

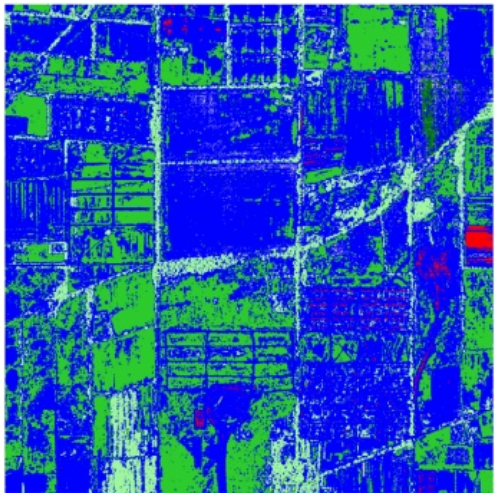
Results

- Transfer learning

Flanders



Argentina



Take-away messages

- Feature identification & selection replaced by architecture building / selection
- Tuning absolutely necessary
- Input data selection also an issue (include height info or not)
- Difficulty how to integrate data at different resolutions (image at 0,25m vs height at 0,5m)
- Not easy to implement an existing architecture in our code : takes care and time (one week = too short)
- Transferability limited (but was not easiest case since different imagery)
- Hardware makes a big difference (GPU) : NVDIA GeForce GTX 1080, 8 GB, Intel® Xeon® CPU E5-2690 0 @ 2.900 GHz 2.90GHz (2 processors), RAM 96.0 GB

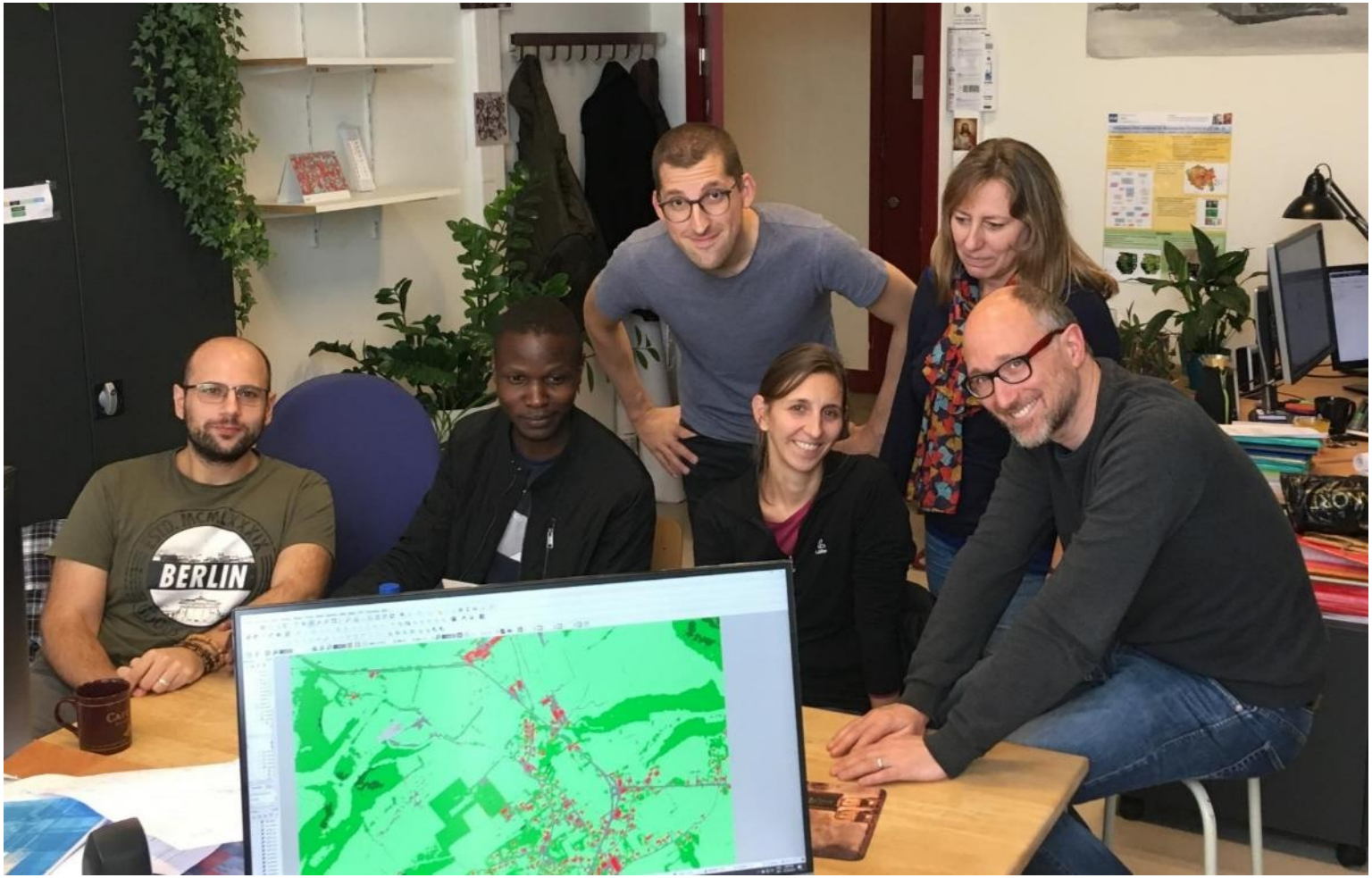
Take-away messages

ULB ANAGEO

- Currently : deep learning at point of passage from research to industrialisation
- Reflection necessary on how FOSS4G community can best integrate deep learning
  - Most of the work & code = data preparation => existing GIS software ideally prepared for that
  - Actual network code very short => easily integratable into wrapper modules
- Community effort for building (& providing !) training data sets would be useful

Thanks for your attention.
Any Questions ?

ULB
ANAGEO

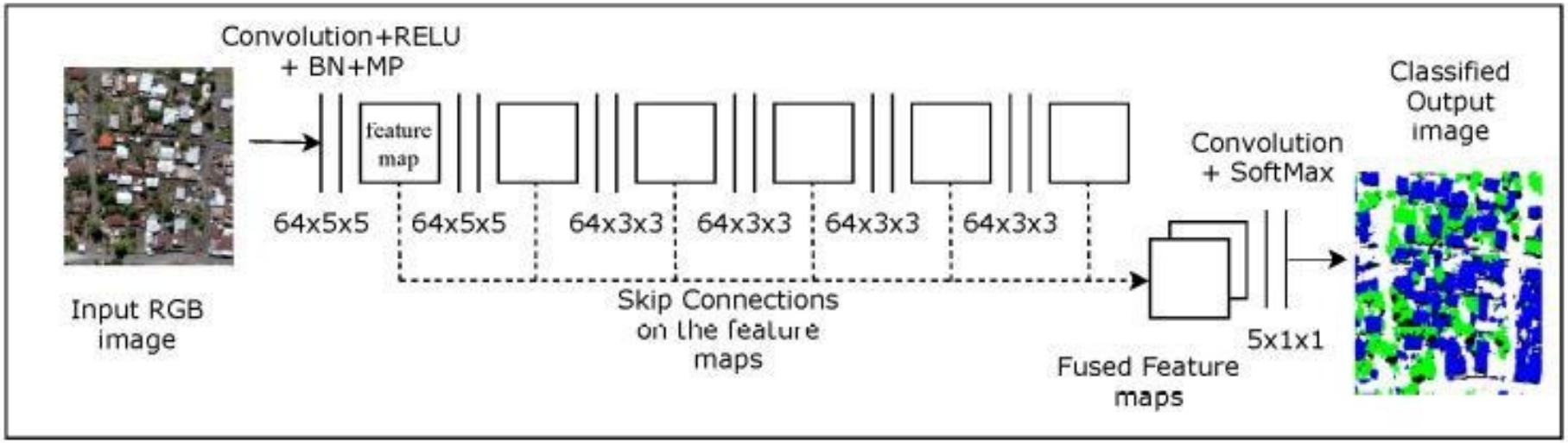


BACKUP SLIDES

Network architectures

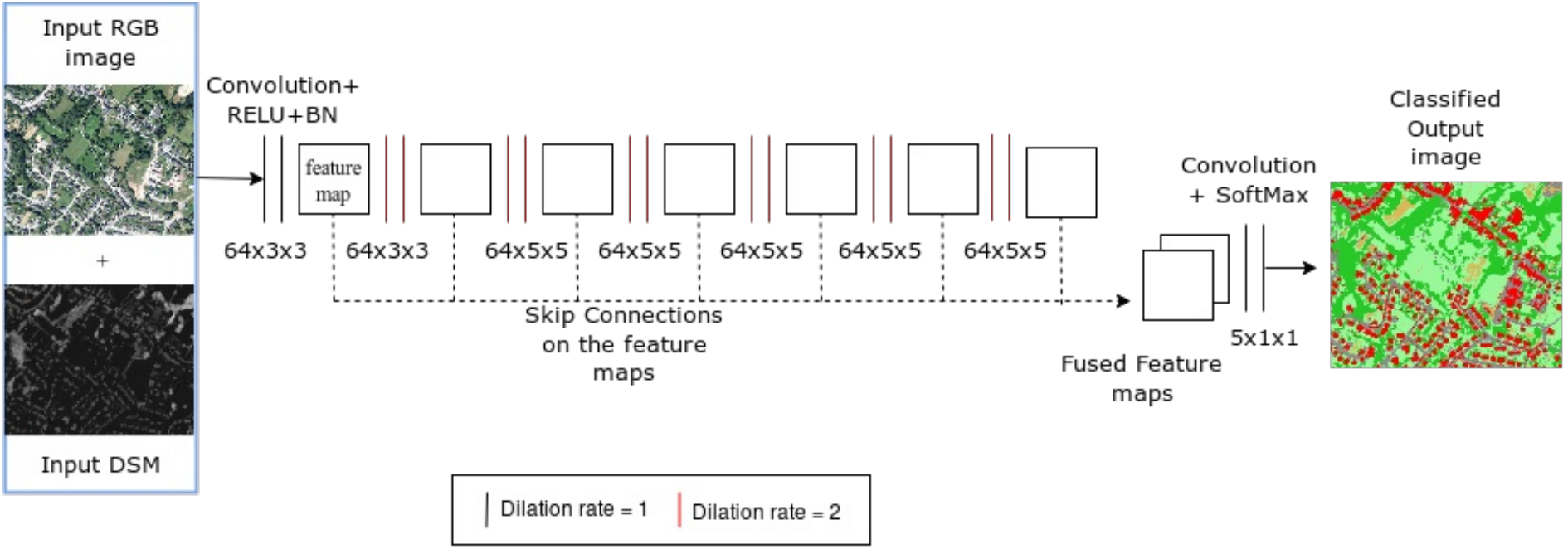
ULB ANAGEO

- NickNet

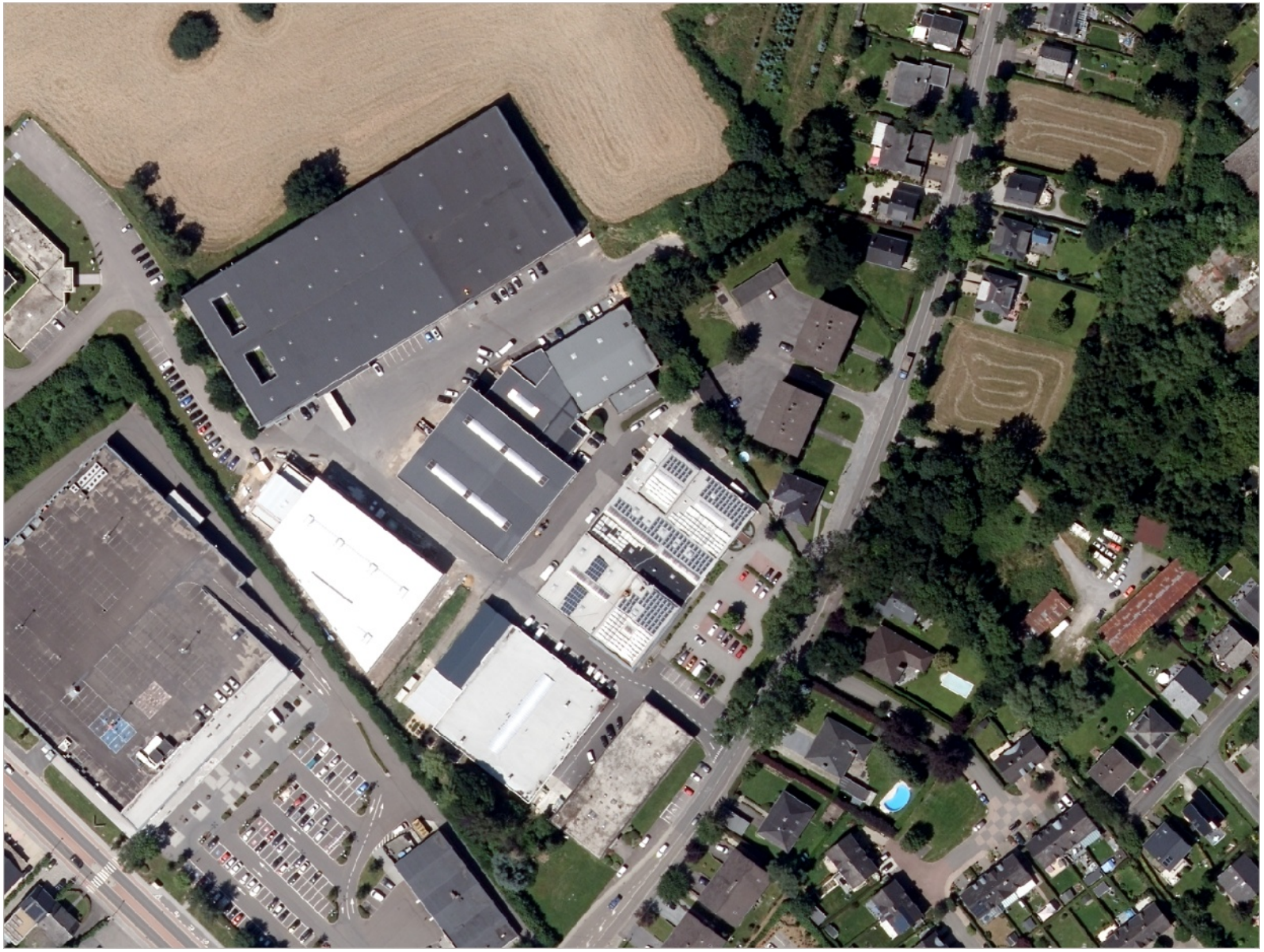


Network architectures

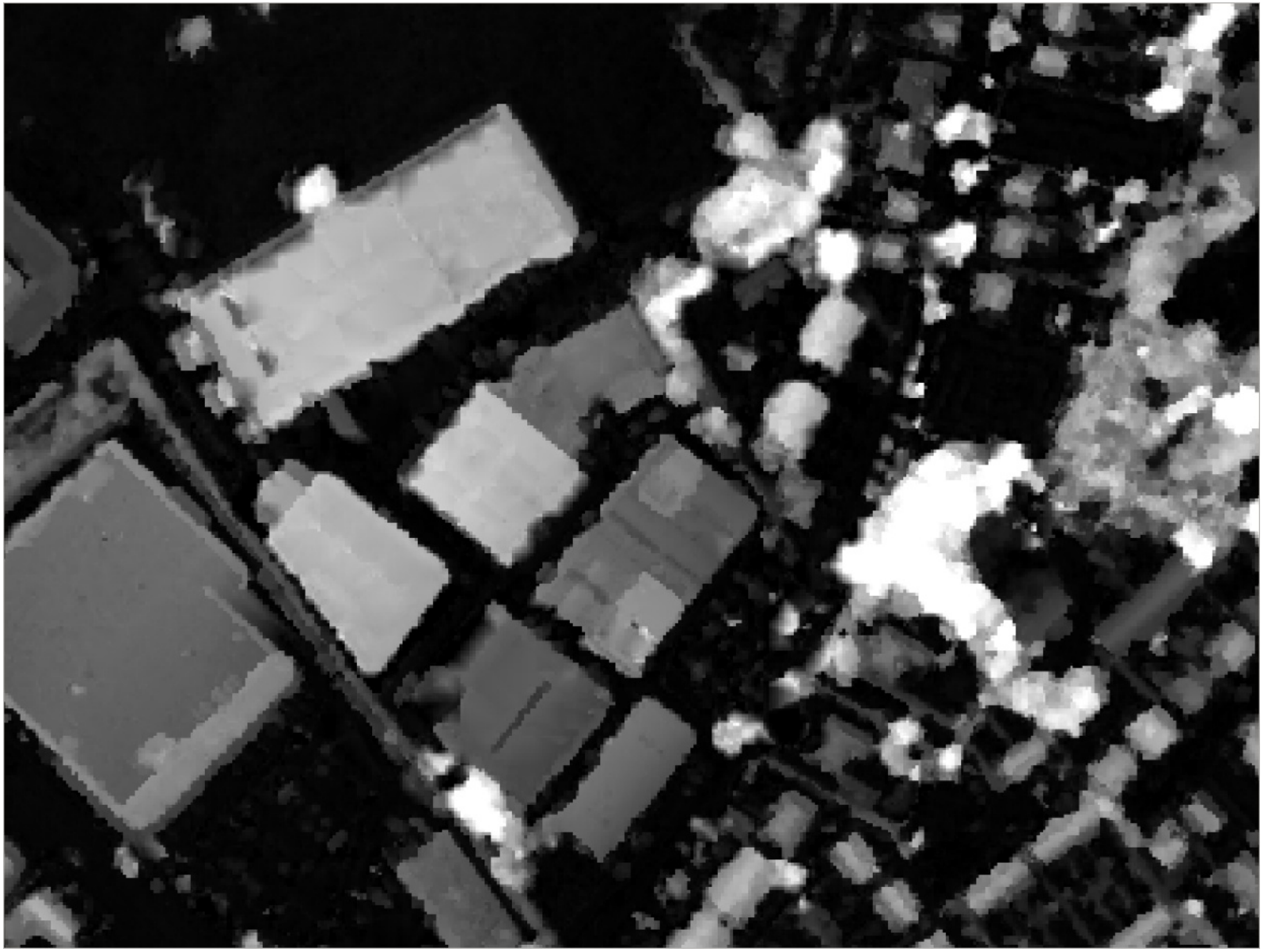
- "AnageoNet"



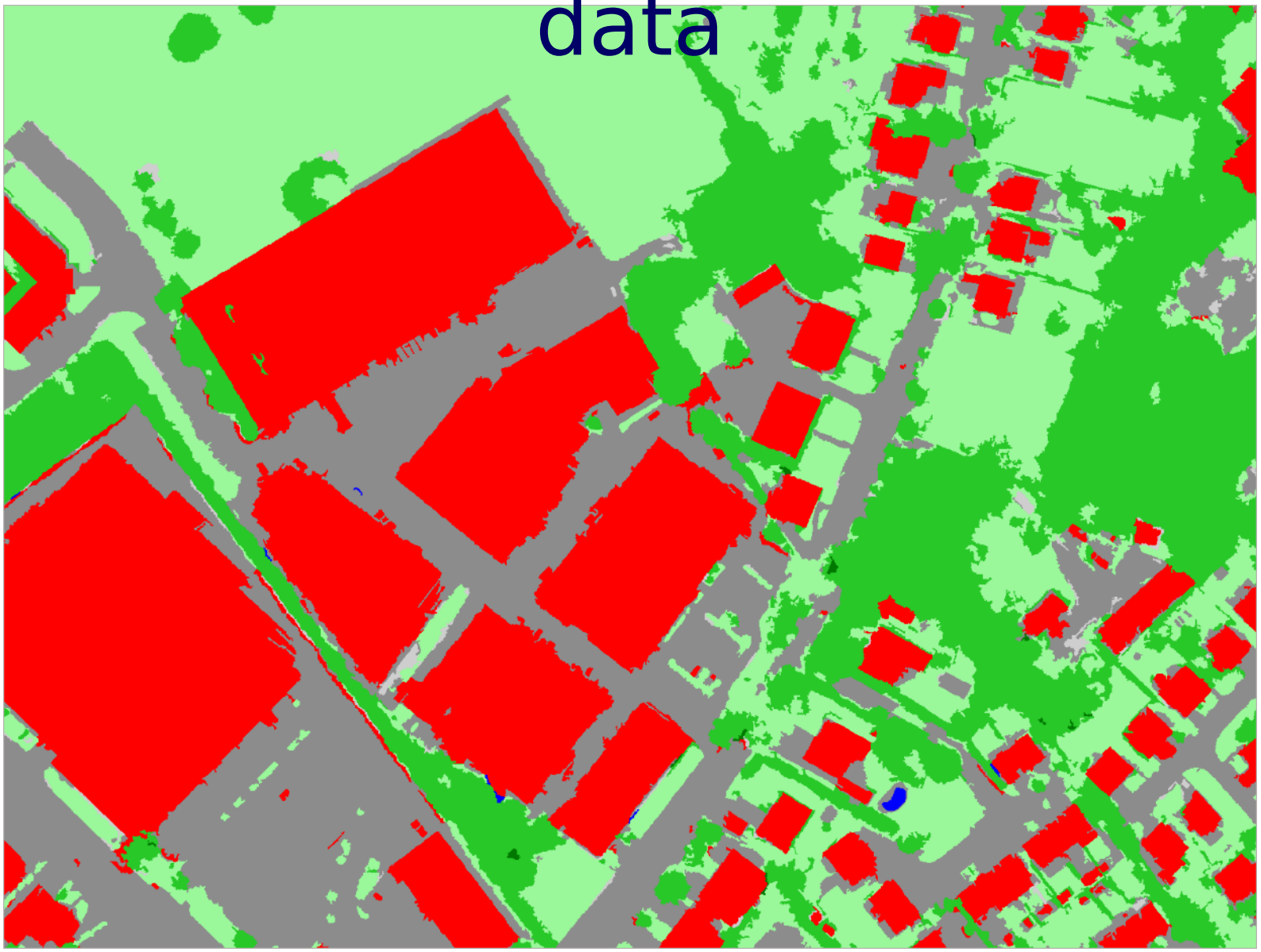
Optical bands



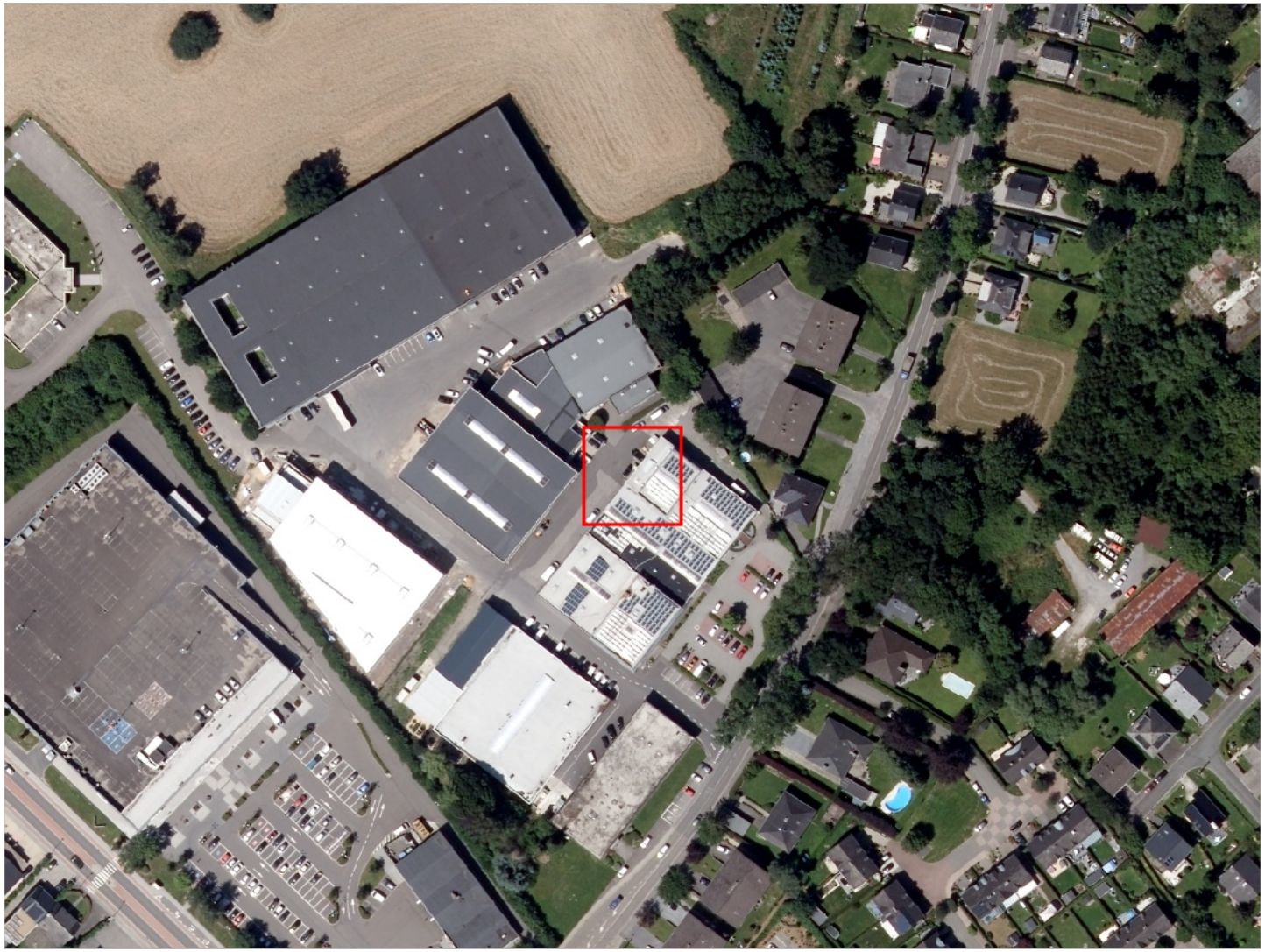
nDSM



Fully labelled reference data



Patch size for training



Patch size for training 128x128

Patch size for training 64x64

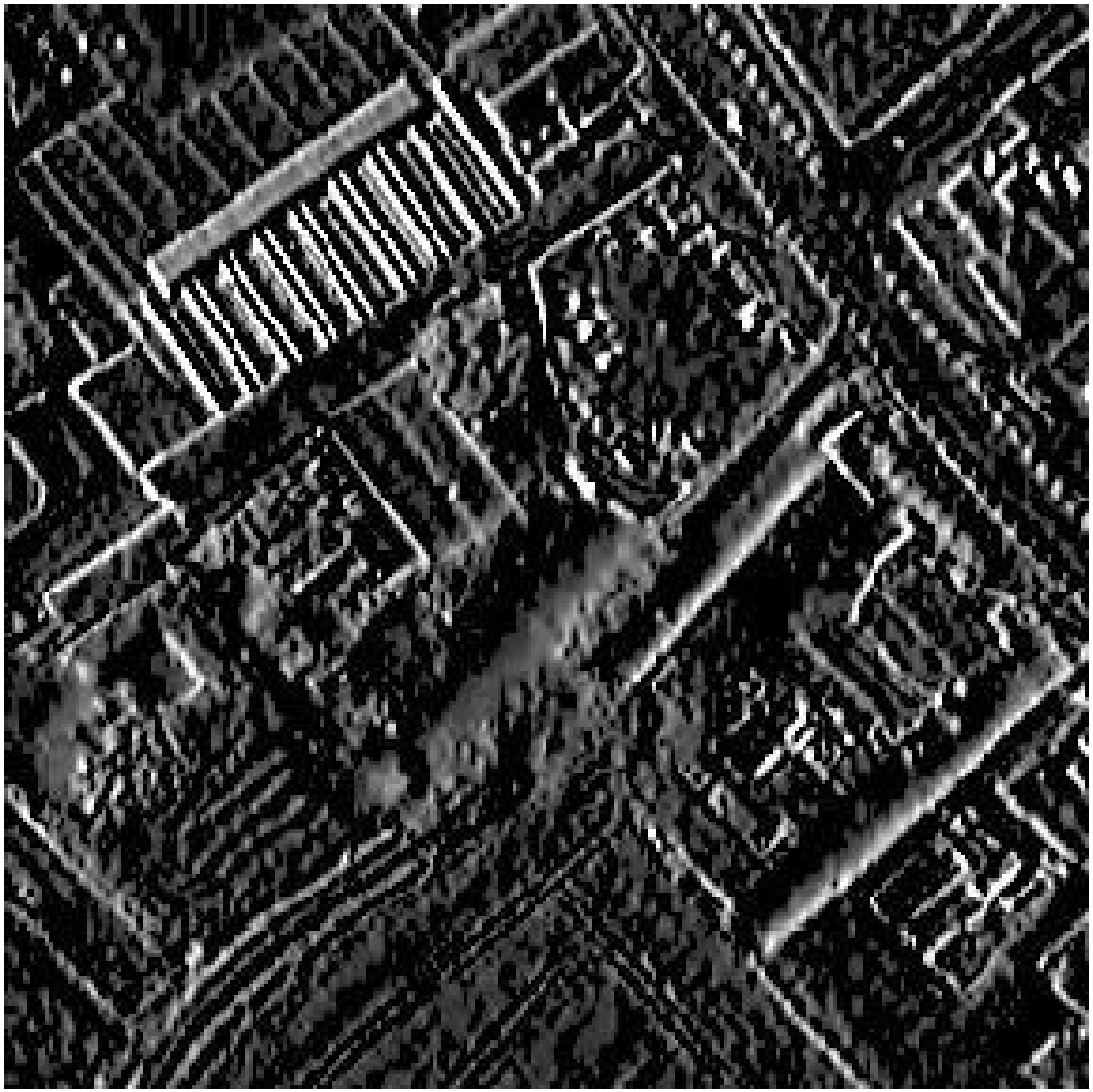
ATROUS

ULB ANAGEO

- NIR

ATROUS

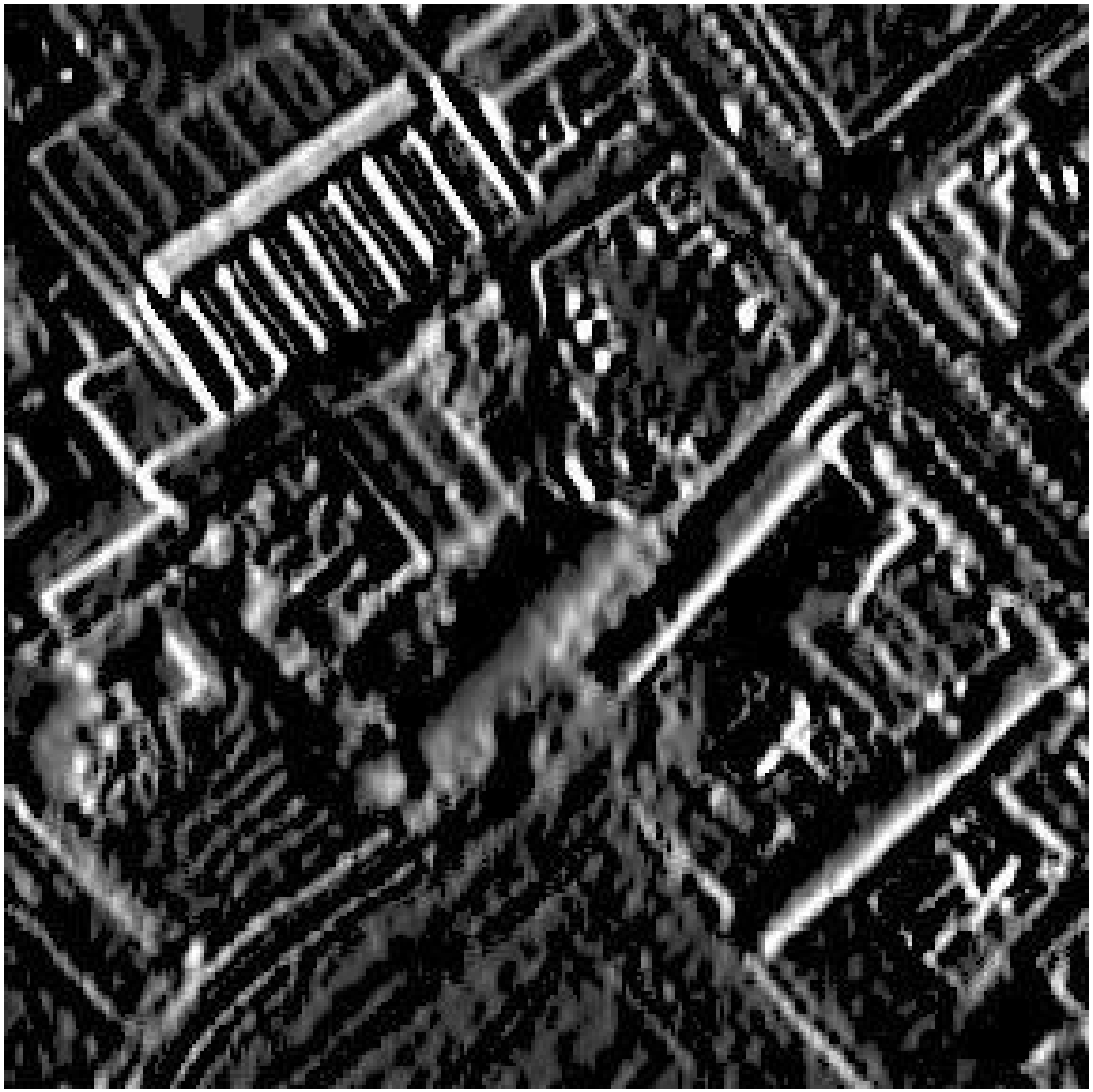
- NIR
- High-pass
- 3x3



ATROUS

ULB ANAGEO

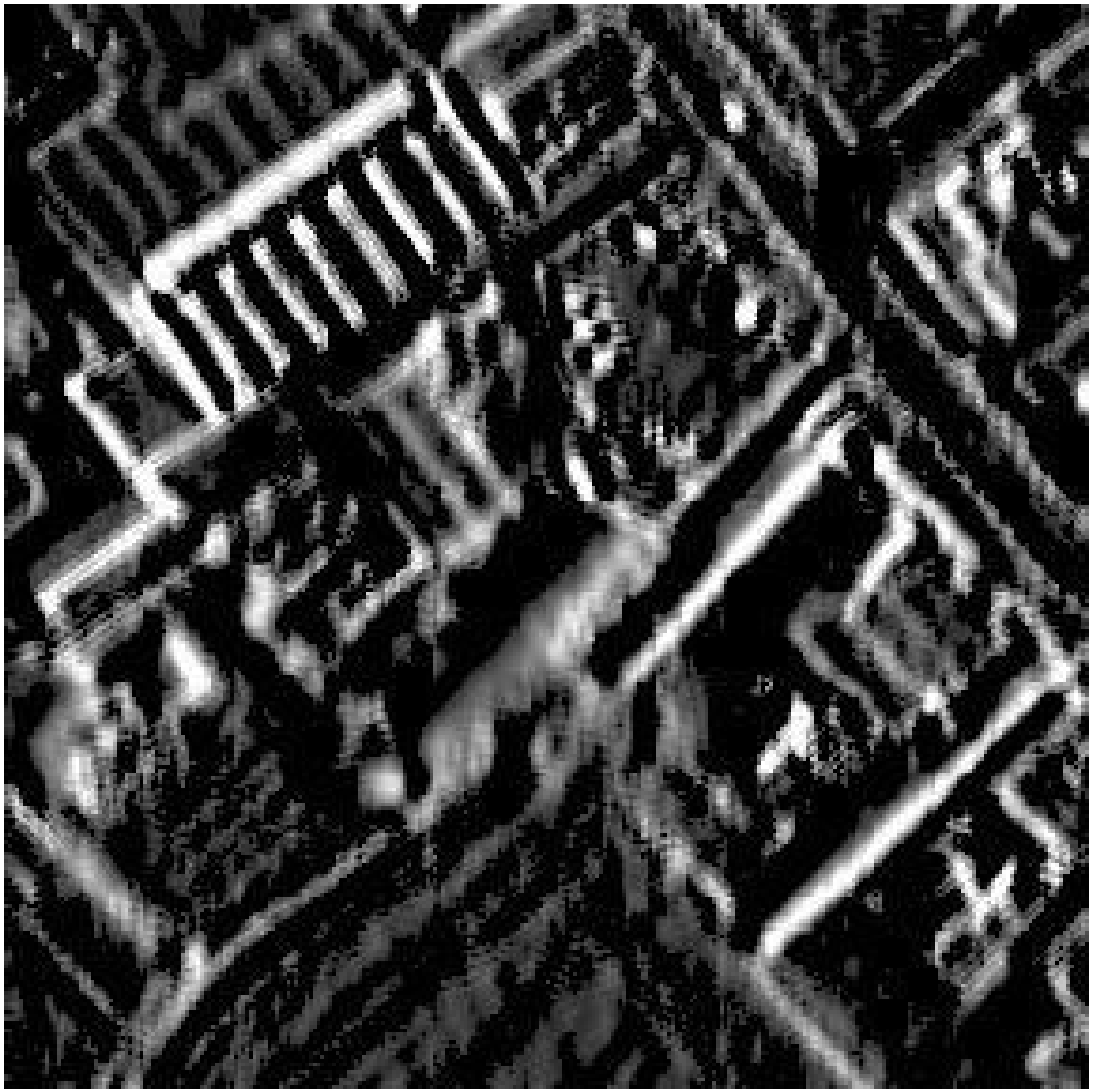
- NIR
- High-pass
- 3x3
- Dilation 5x5



ATROUS

ULB ANAGEO

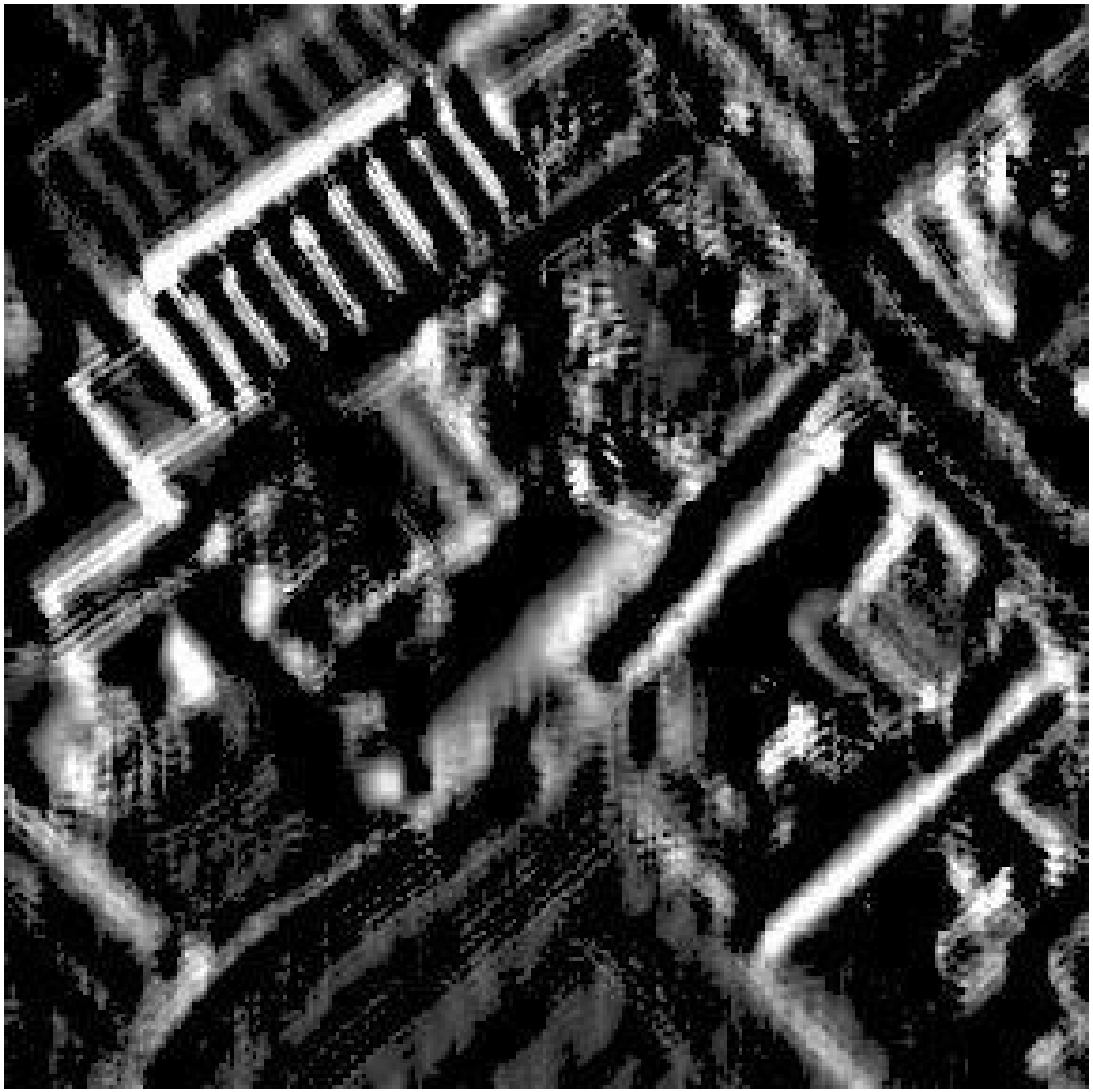
- NIR
- High-pass
- 3x3
- Dilation 7x7



ATROUS

ULB ANAGEO

- NIR
- High-pass
- 3x3
- Dilation 9x9



Accuracy assessment

ULB ANAGEO

- VNIR + nDSM

```
Performance evalutation:
Overall Accuracy: 0.826656893299126 5
Cohen's Kappa: 0.7547518076016747
F1-score: 0.828978052995499
```

Classification report:

	precision	recall	f1-score	support
Bare soil	0.71	0.62	0.66	156769
Herbaceous	0.52	0.97	0.68	84663
Asphalt	0.47	0.16	0.24	15423
Building	0.86	0.88	0.87	309664
Coniferous	0.74	0.91	0.82	472683
Deciduous	0.96	0.80	0.87	955463
Water	0.84	0.80	0.82	5328
avg / total	0.85	0.83	0.83	1999993

Accuracy assessment

ULB ANAGEO

- VNIR only

```
Performance evalutation:
Overall Accuracy: 0.843761529647740 8
Cohen's Kappa: 0.776260062250056 3
F1-score: 0.842706805758319 5


Classification report:
              precision    recall  f1-score   support

  Bare soil       0.76      0.80      0.78    153472
 Herbaceous       0.74      0.73      0.73     83335
    Asphalt       0.64      0.10      0.17     14862
   Building       0.83      0.86      0.84    297859
 Coniferous       0.78      0.86      0.82    463838
  Deciduous       0.92      0.86      0.89    881639
      Water       0.91      0.76      0.83      4988

avg / total       0.85      0.84      0.84   1899993
```