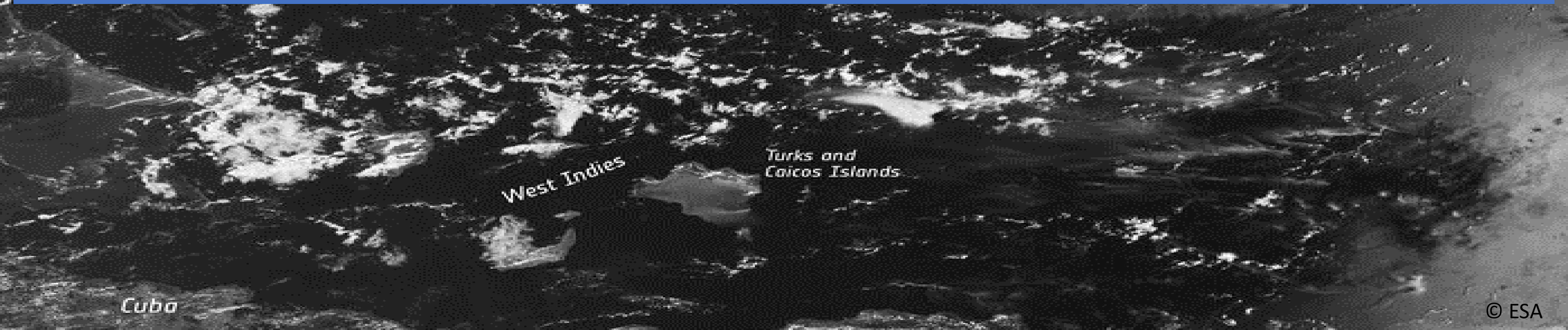
Synergetic use of Planet data and high-resolution aerial images for windthrow detection based on Deep Learning

Melanie Brandmeier, Zayd Hamdi, Wolfgang Deigele, Christoph Straub



Sustainably manage forests, combat desertification, halt and reverse land degradation, halt biodiversity loss

15 LIFE ON LAND

Forests cover 30.7 per cent of the Earth's surface and, in addition to providing food security and shelter, they are key to combating climate change, protecting biodiversity and the homes of the indigenous population. By protecting forests, we will also be able to strengthen natural resource management and increase land productivity.

At the current time, thirteen million hectares of forests are being lost every year while the persistent degradation of drylands has led to the desertification of 3.6 billion hectares. Even though up to 15% of land is currently under protection, biodiversity is still at risk. Deforestation and desertification – caused by human activities and climate change – pose major challenges to sustainable development and have affected the lives and livelihoods of millions of people in the fight against poverty.

Efforts are being made to manage forests and combat desertification. There are two international agreements being implemented currently that promote the use of resources in an equitable way. Financial investments in support of biodiversity are also being provided.

The Lion's Share Fund

On 21 June, 2018, the United Nations Development Programme (UNDP), FINCH and founding partner Mars, Incorporated, announced the **Lion's Share**, an initiative aimed at transforming the lives of animals across the world by asking advertisers to contribute a percentage of their media spend to conservation and animal welfare projects. The Lion's Share will see partners contribute 0.5 percent of their media spend to the fund for each advertisement they use featuring an animal. Those funds will be used to support animals and their habitats around the world. The Fund is seeking to raise US$100m a year within three years, with the money being invested in a range of wildlife conservation and animal welfare programs to be implemented by United Nations and civil society organizations.

THE 17 GOALS

1 NO POVERTY

2 ZERO HUNGER

3 GOOD HEALTH AND WELL-BEING

4 QUALITY EDUCATION

5 GENDER EQUALITY

6 CLEAN WATER AND SANITATION

7 AFFORDABLE AND CLEAN ENERGY

8 DECENT WORK AND ECONOMIC GROWTH

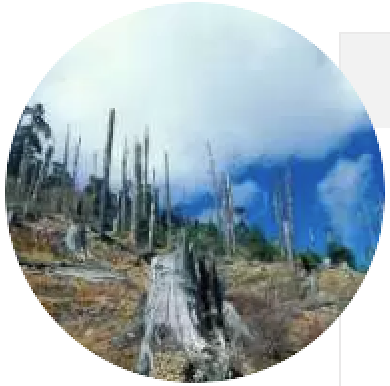
SUSTAINABLE DEVELOPMENT GOALS

Search SDG Site

A-Z Site Index

Sustainably manage forests, combat desertification, halt and reverse land degradation, halt biodiversity loss

15 LIFE ON LAND



Facts and figures | Goal 15 targets | Links

15.1 By 2020, ensure the conservation, restoration and sustainable use of terrestrial and inland freshwater ecosystems and their services, in particular forests, wetlands, mountains and drylands, in line with obligations under international agreements

The Lion's Share Fund

On 21 June, 2018, the United Nations Development Programme (UNDP), FINCH and founding partner Mars, Incorporated, announced the **Lion's Share**, an initiative aimed at transforming the lives of animals across the world by asking advertisers to contribute a percentage of their media spend to conservation and animal welfare projects. The Lion's Share will see partners contribute 0.5 percent of their media spend to the fund for each advertisement they use featuring an animal. Those funds will be used to support animals and their habitats around the world. The Fund is seeking to raise US$100m a year within three years, with the money being invested in a range of wildlife conservation and animal welfare programs to be implemented by United Nations and civil society organizations.

THE 17 GOALS

1 NO POVERTY

2 ZERO HUNGER

4 QUALITY EDUCATION

6 CLEAN WATER AND SANITATION

7 AFFORDABLE AND CLEAN ENERGY

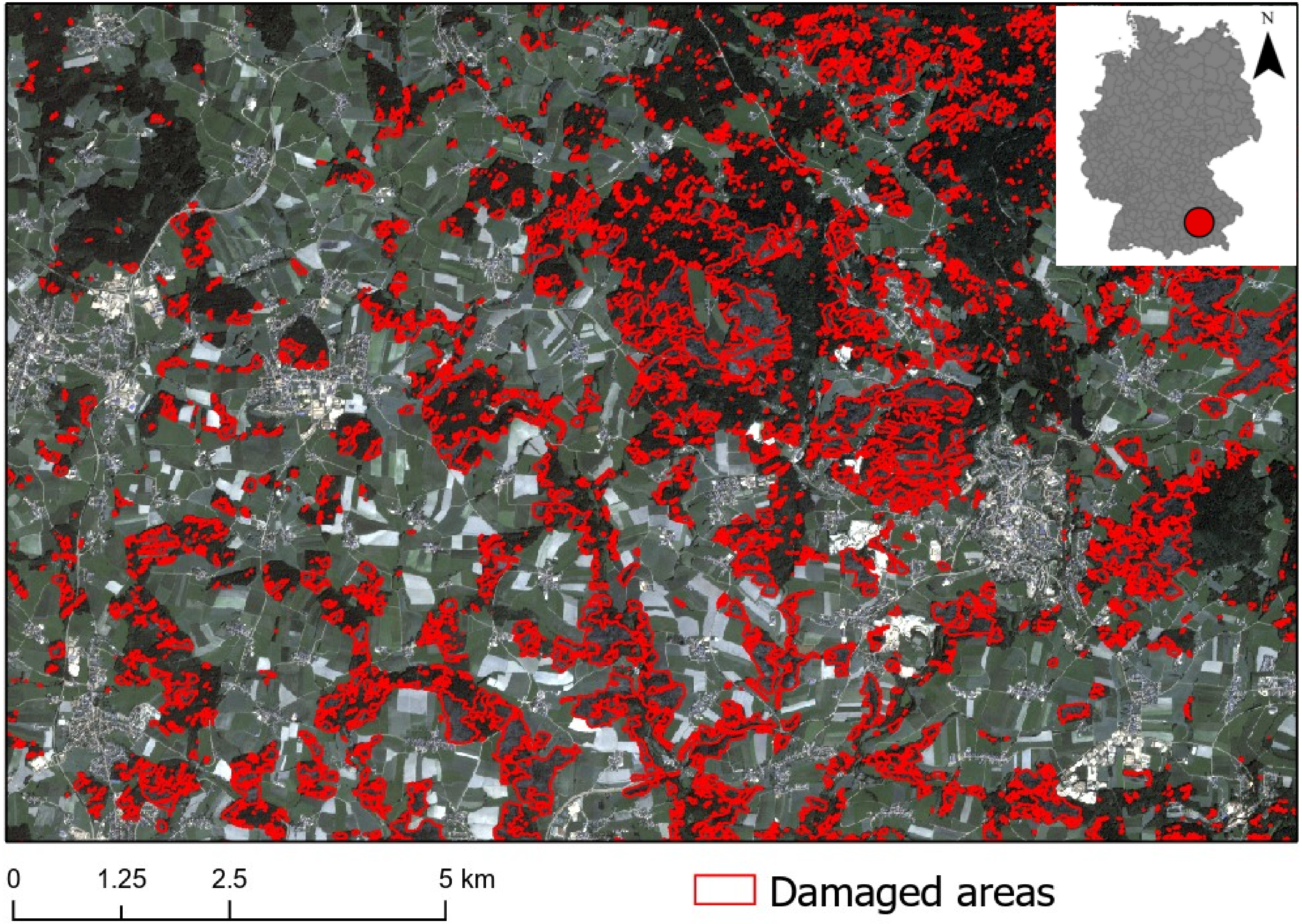
8 DECENT WORK AND ECONOMIC GROWTH

- Fast, automatic detection of fallen trees, blocked roads… needed for effective forest management
- Large affected areas
- State of the art: Manual digitalization
- Change detection using pre- and post storm imagery (active and passive systems)

→ CNNs only need one post-storm image

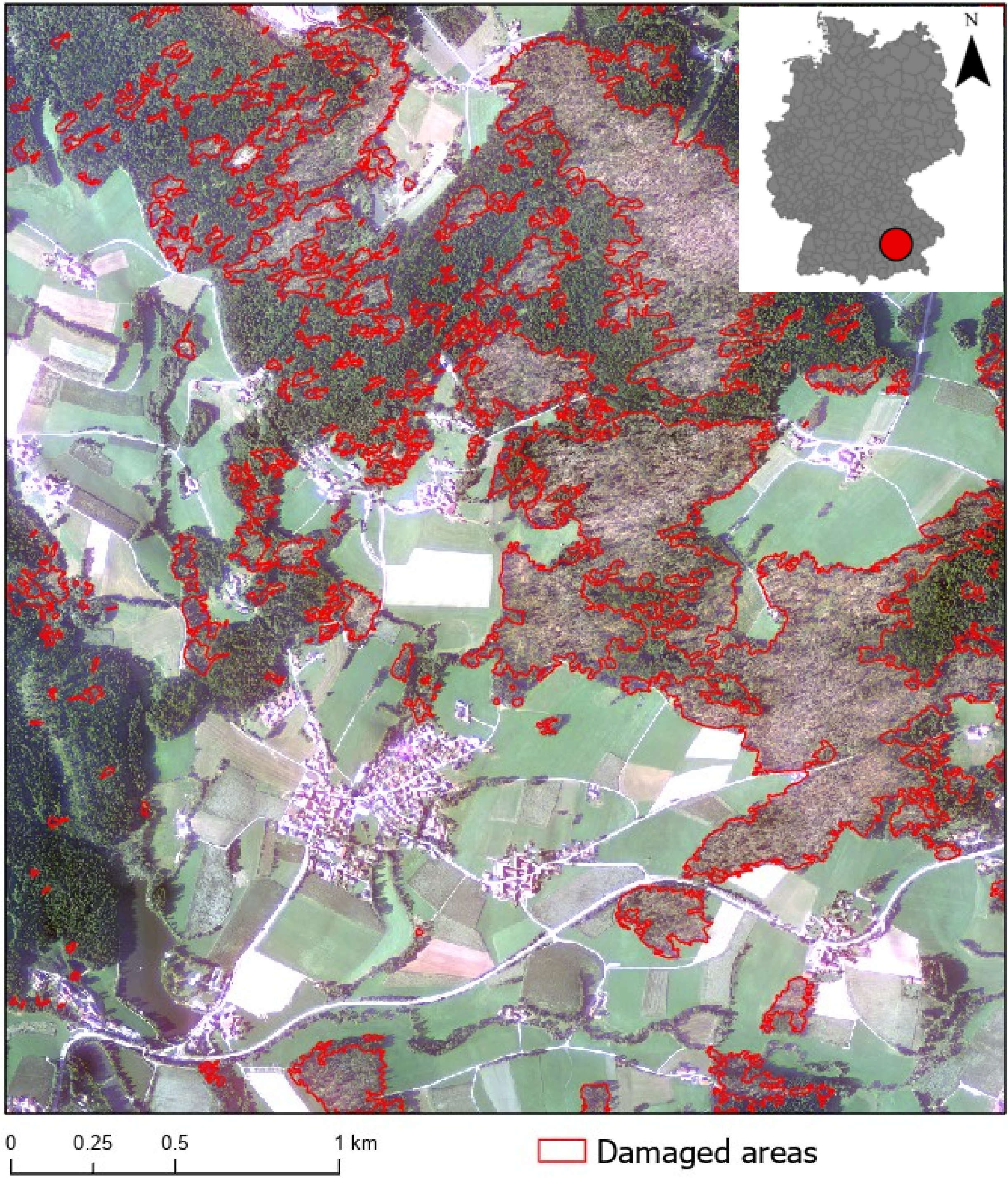
Study area

- Bavaria (confidential area)
- Over 2 million Festmeter of damaged trees during the last Thunderstorm in Bavaria (LWF aktuell 115)



Data: airborne

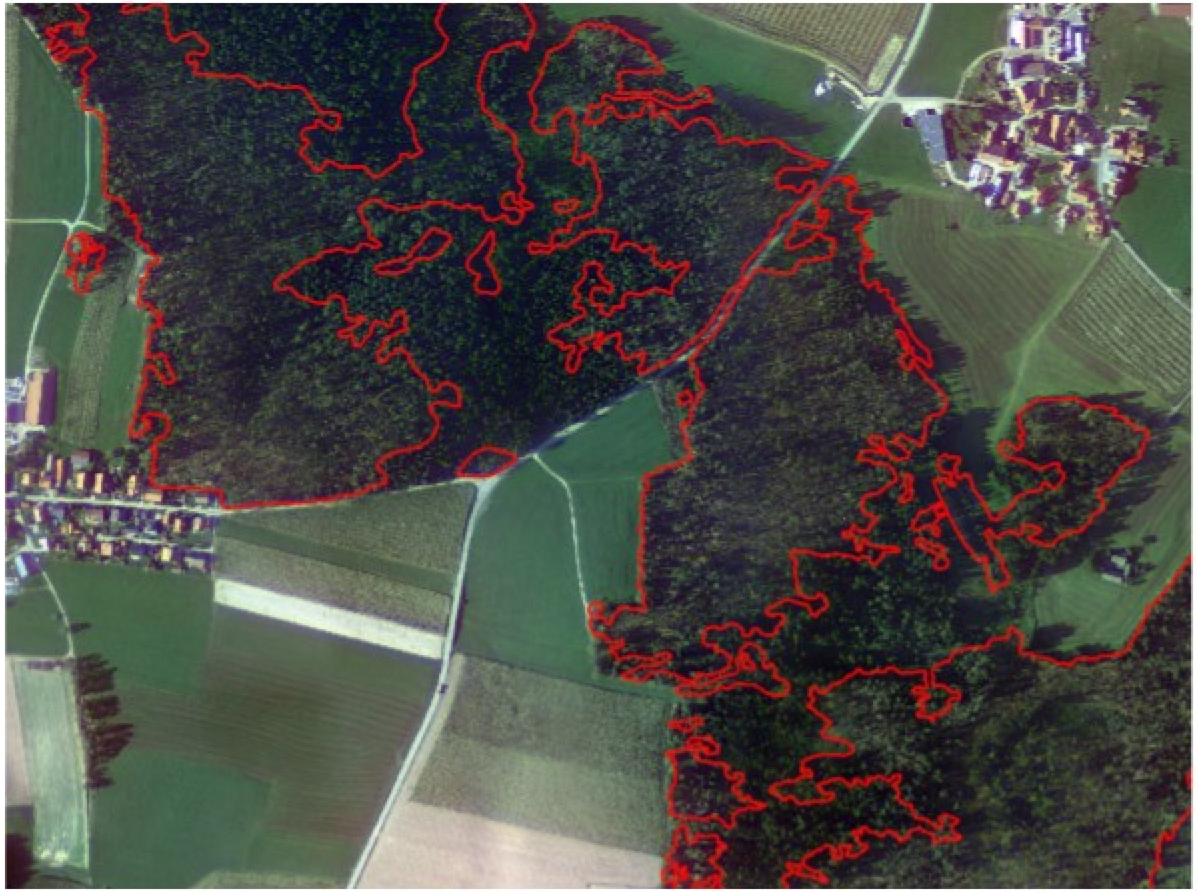
- 45 10,000x10,000px Orthophotos (RGB + NIR)
- 20 cm spatial resolution
- 10 for training (and validation)
- 2 for testing
- Shapefile containing polygons around each damaged area, manually digitized by LWF (17,3 km^2 damaged area)



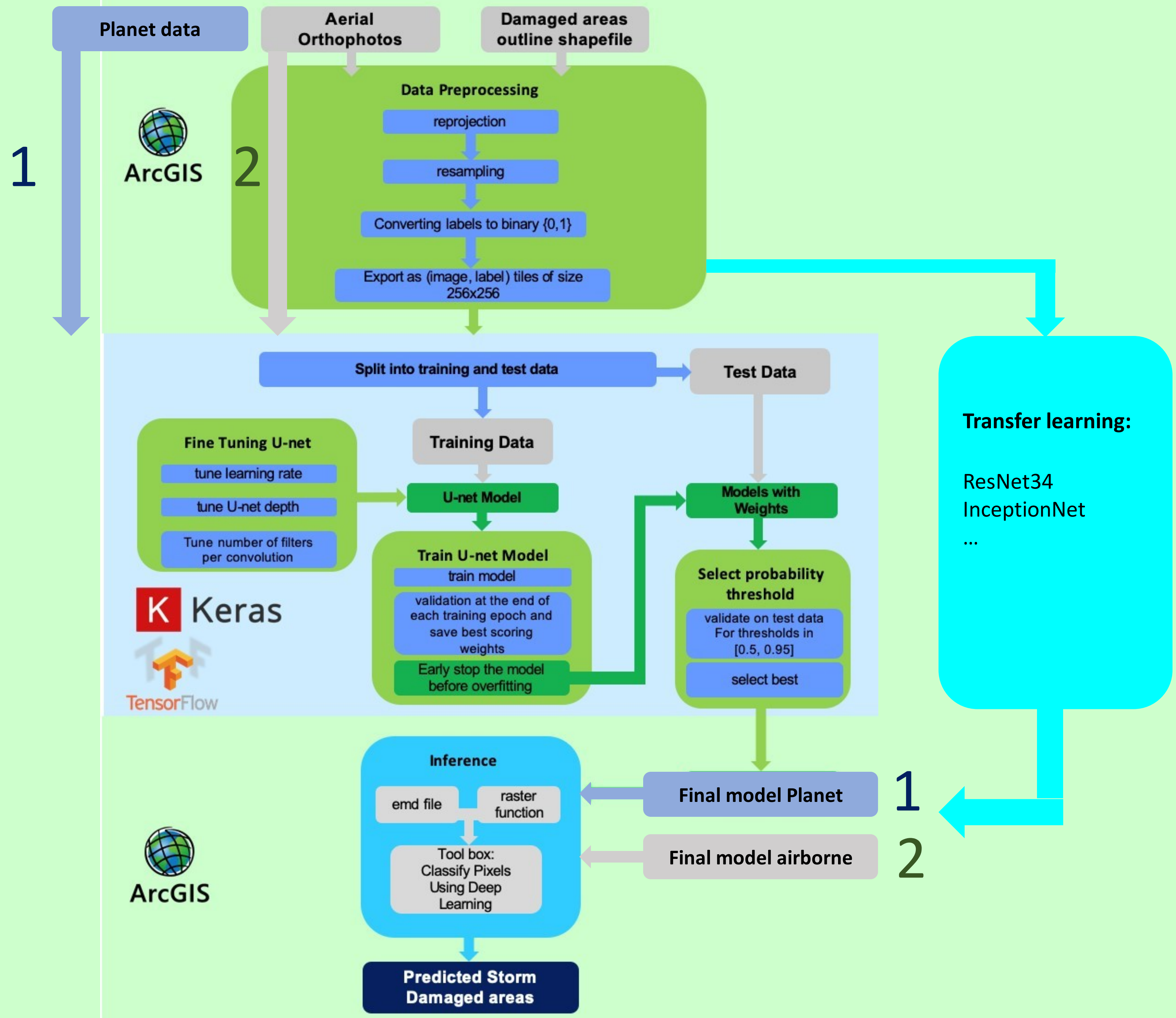
Data: Planet dove

- Three scenes of Planet Dove data (RGB + NIR), multitemporal after storm (August 2017)
- 3m spatial resolution
- Variable signal to noise ratio
- Separate labels derived from Planet data (7,9 km^2 damaged area)

Data: Comparison of the label datasets

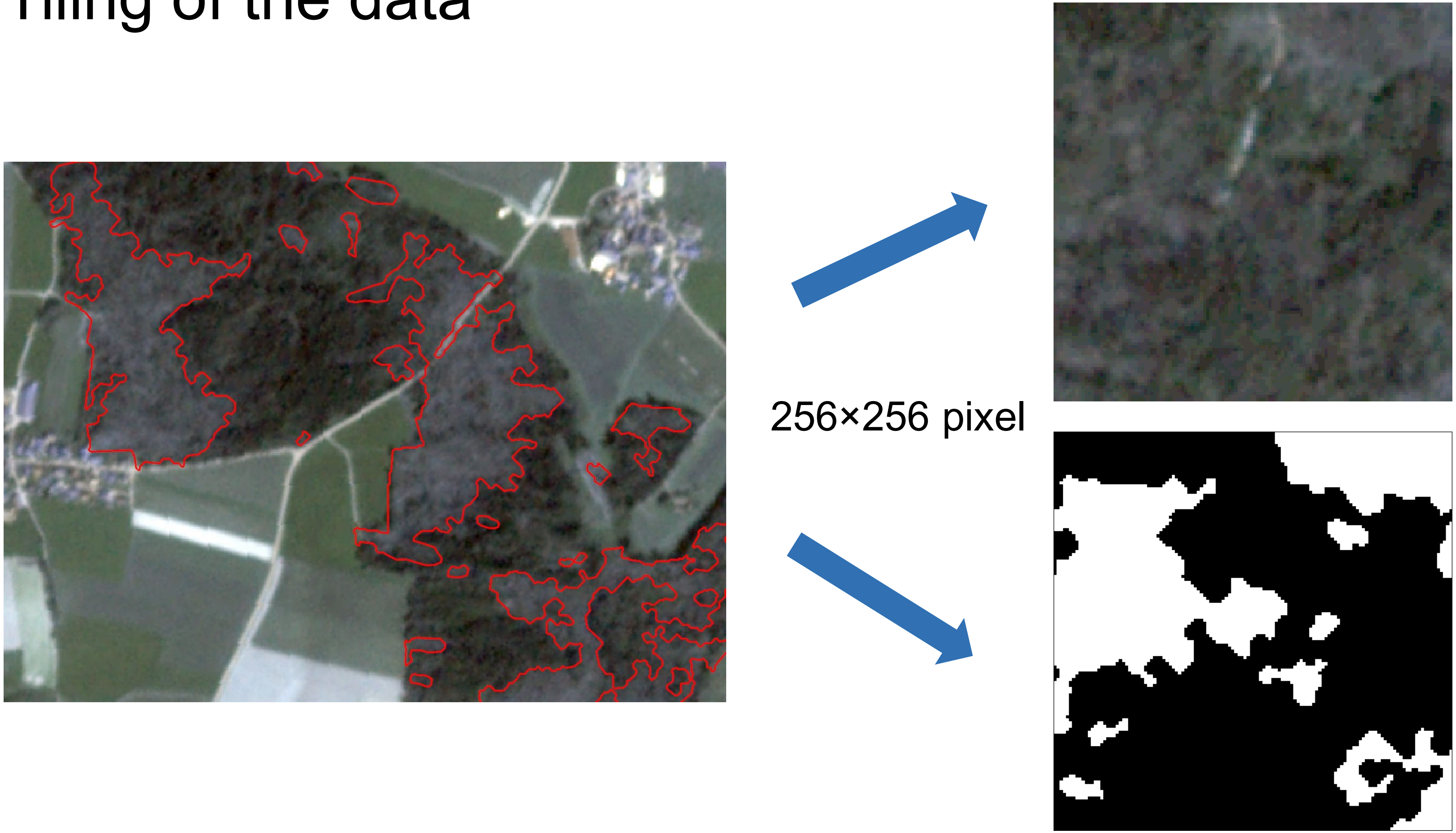


Time after a storm

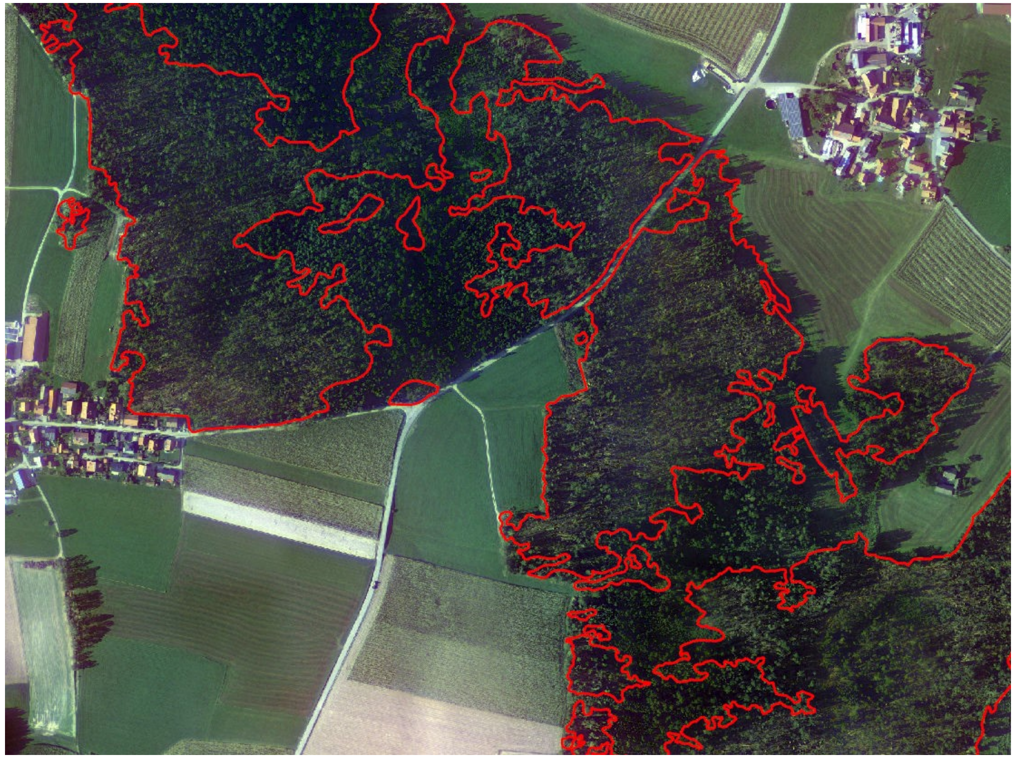


Planet data
Aerial Orthophotos
Damaged areas outline shapefile
1
ArcGIS
2
Data Preprocessing
reprojection
resampling
Converting labels to binary {0,1}
Export as (image, label) tiles of size 256x256
Split into training and test data
Test Data
Fine Tuning U-net
tune learning rate
tune U-net depth
Tune number of filters per convolution
Training Data
U-net Model
Train U-net Model
train model
validation at the end of each training epoch and save best scoring weights
Early stop the model before overfitting
Models with Weights
Select probability threshold
validate on test data For thresholds in [0.5, 0.95]
select best
Keras
TensorFlow
Transfer learning:
ResNet34
InceptionNet
…
Inference
emd file
raster function
Tool box: Classify Pixels Using Deep Learning
Final model Planet
1
Final model airborne
2
ArcGIS
Predicted Storm Damaged areas

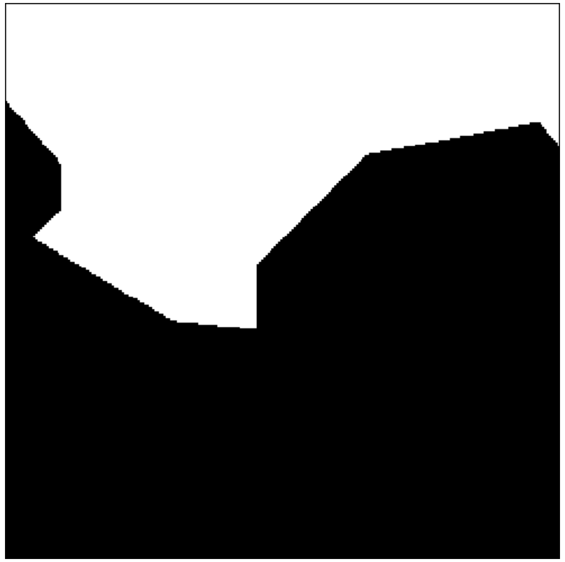
Tiling of the data



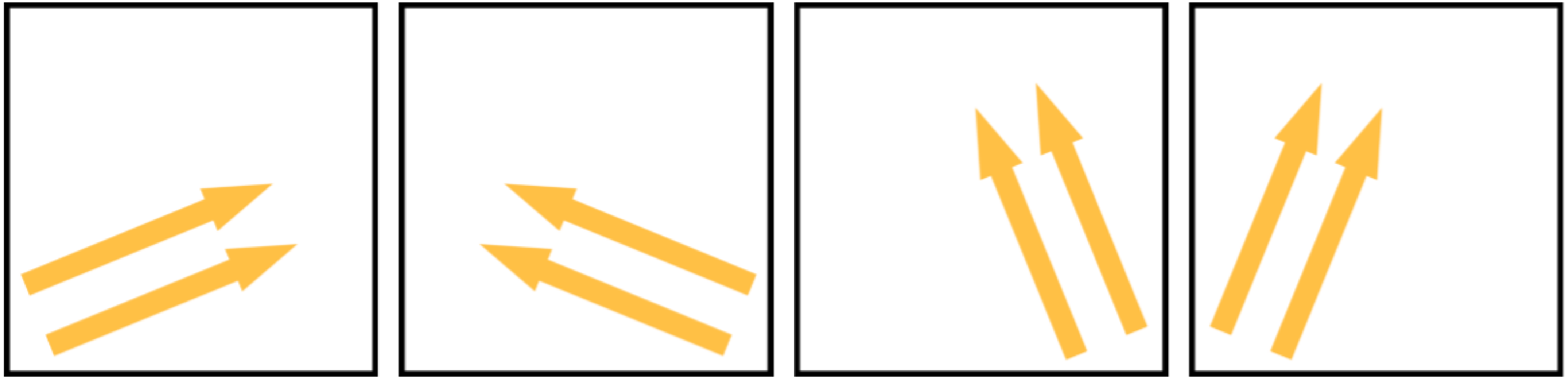
Tiling of the data



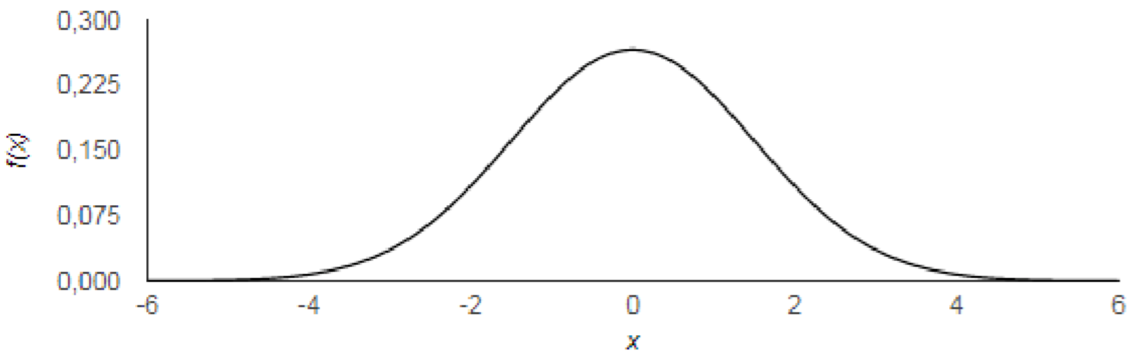
256×256 pixel



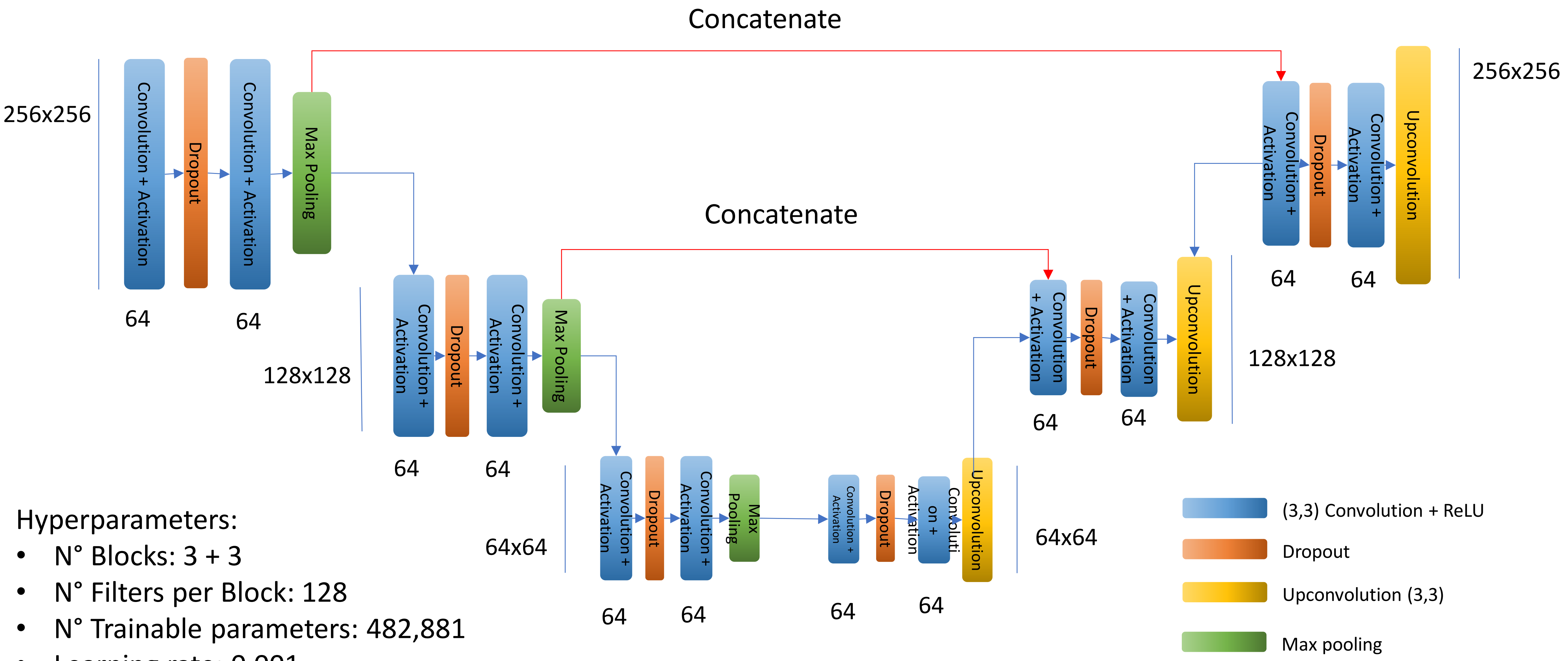
Data augmentation: geometric and radiometric



Example: Normal distribution ofrandom noise; sigma of 1.5; center at 0



Model Setup: U-Net architecture (airborne)

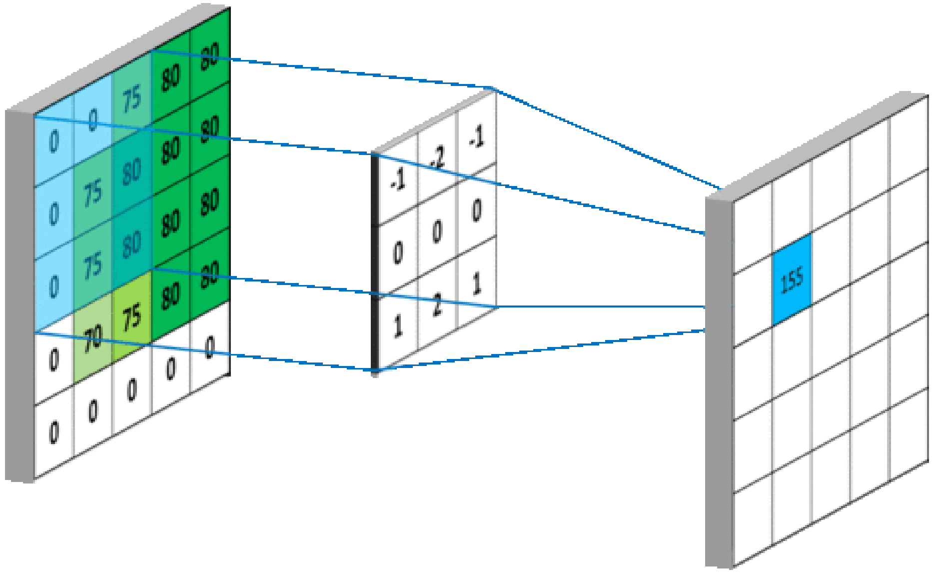


Hyperparameters:

- N° Blocks: 3 + 3
- N° Filters per Block: 128
- N° Trainable parameters: 482,881
- Learning rate: 0.001

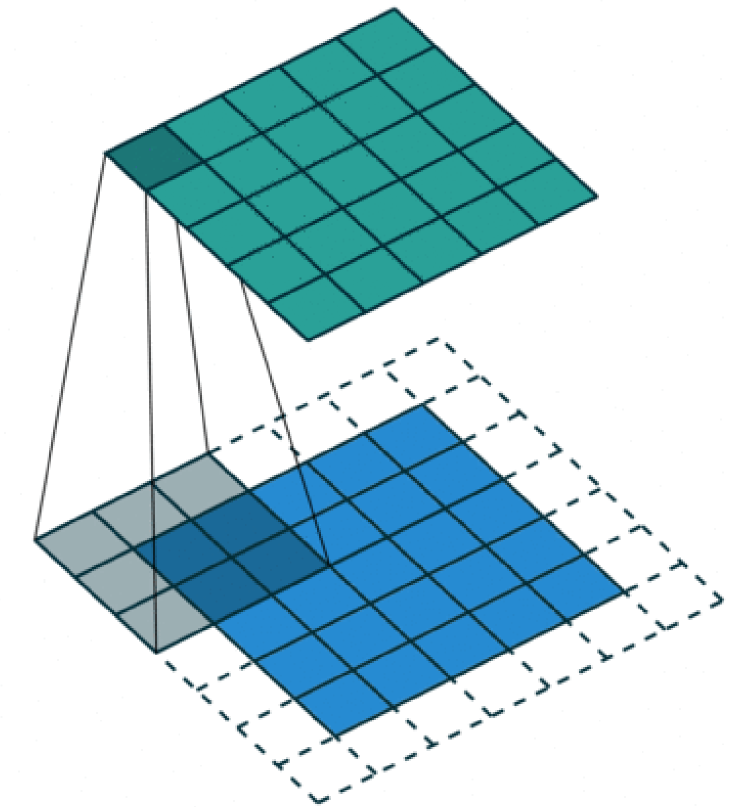
Convolutional Neural Network

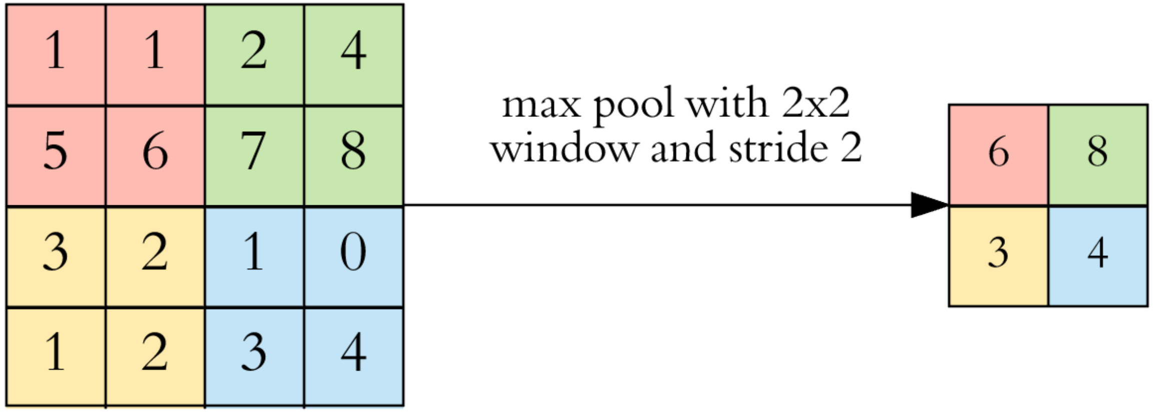
Image Convolution



Convolutional Neural Network

Padding and Pooling





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

Loss Function and Optimizer

- Loss function:
  - Minimizing the cross entropy (and weighted cross entropy, damaged pixels only 0.5%) between the distribution of the prediction $\hat{y}_i$ = f(Y|X) and the ground truth Y:

$$L = -\frac{1}{N}\sum_{i=0}^{N} y_i.\log(\hat{y}_i) + (1 - y_i).\log(1 - \hat{y}_i)$$

- Filter initialization using the Normal distribution centered on 0 (Lecun_Normal)
- Optimizer for weights updating: Adam (Adaptive Moment estimation)

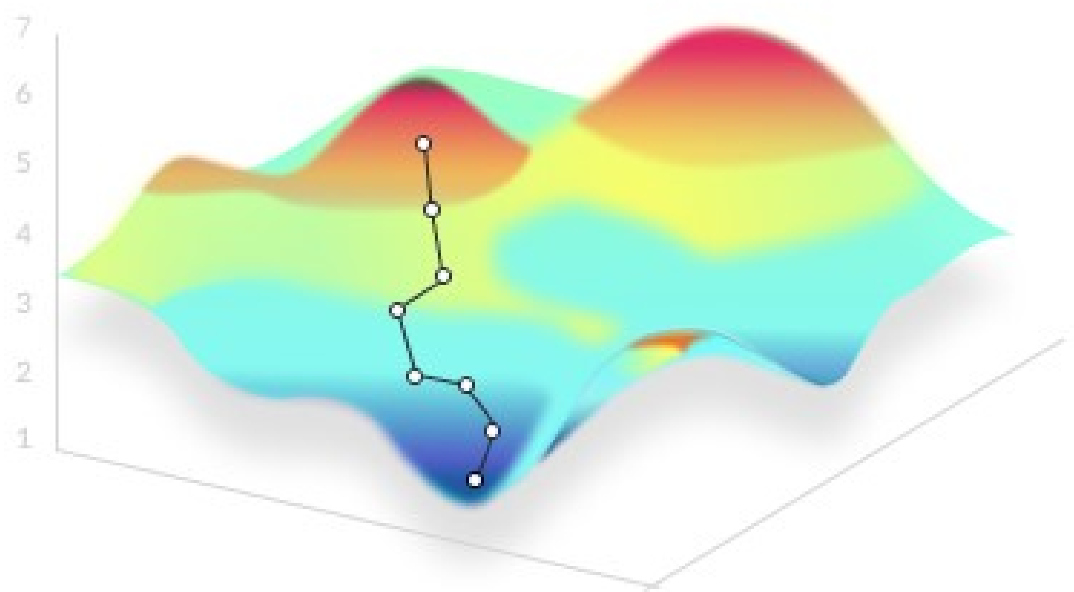
$$w_{t+1} = w_t - \eta \frac{\widehat{m_w}}{\sqrt{\widehat{v_w}} + \epsilon} + 1$$

$$\widehat{m_w} = \frac{m_w^{t+1}}{1-(\beta_1)^{t+1}} \quad , \quad \widehat{v_w} = \frac{v_w^{t+1}}{1-(\beta_2)^{t+1}}$$

$$m_w^{t+1} = \beta_1 m_w^t + (1 - \beta_1)\nabla_{\mathrm{w}}\mathrm{L}^{\mathrm{t}} \quad , \quad m_w^{t+1} = \beta_2 m_w^t + (1 - \beta_2)(\nabla_{\mathrm{w}}\mathrm{L}^{\mathrm{t}})^2$$

β forgetting factor

Gradient descent



Source: https://missinglink.ai/guides/neural-network-concepts/backpropagation-neural-networks-process-examples-code-minus-math/

Hyperparameters

Tile size:

			128 × 128		256 × 256	
Number	Learning rate	Blocks	IoU	Seconds	IoU	Seconds
1	0.001	[32, 32, 32, 32]	0.4573	290		475
2	0.0015	[8, 16, 32, 64]	0.4566	260		445
3	0.001	[16, 32, 64]	0.4461	370		530
4	0.002	[16, 16, 32, 32]	0.4481	310		420
5	0.001	[8, 16, 32, 64, 128]	0.4632	410		610

Learning rate:

Number	Learning rate	IoU
1	0.0001	0.4329
2	0.0003	0.4521
3	0.0005	0.4581
4	0.0007	0.4564
5	0.0009	0.4600
6	0.0011	0.4598
7	0.0013	0.4610
8	0.0015	0.4594
12	0.0023	0.4581
13	0.0025	0.4561
14	0.0027	0.4587
15	0.0029	0.4547
16	0.0031	0.4567

Architecture: (Lr = 0.001, 256 x 256)

Number	Blocks	IoU	Seconds/Epoch
1	[64,64,64,64]	0.4666	880
3	[16,32,64,128]	0.4640	760
4	[32,64,128]	0.4629	830
5	[16,16,64,64]	0.4627	660
6	[32,32,32,32]	0.4576	480
7	[32,32,16,16]	0.4554	430
8	[64,32,16,8]	0.4538	560
9	[16,16,32,32]	0.4537	410
10	[4,8,16,32,64,128]	0.4527	600
11	[16,32,64]	0.4513	530
12	[8,16,16,32]	0.4489	360
13	[16,16,16,16]	0.4468	400
14	[16,16,16,16,16]	0.4457	430
15	[64,32,16,8,4]	0.4382	600
16	[4,8,16,32,64]	0.4365	410

U-net based Model

Example for airborne data

```
____________________________________________________________________________________________________
Layer (type)                     Output Shape          Param #     Connected to
====================================================================================================
input_1 (InputLayer)             (None, 256, 256, 4)   0
____________________________________________________________________________________________________
conv2d_1 (Conv2D)                (None, 256, 256, 64)  2368        input_1[0][0]
____________________________________________________________________________________________________
alpha_dropout_1 (AlphaDropout)   (None, 256, 256, 64)  0           conv2d_1[0][0]
____________________________________________________________________________________________________
conv2d_2 (Conv2D)                (None, 256, 256, 64)  36928       alpha_dropout_1[0][0]
____________________________________________________________________________________________________
max_pooling2d_1 (MaxPooling2D)   (None, 128, 128, 64)  0           conv2d_2[0][0]
____________________________________________________________________________________________________
conv2d_3 (Conv2D)                (None, 128, 128, 64)  36928       max_pooling2d_1[0][0]
____________________________________________________________________________________________________
alpha_dropout_2 (AlphaDropout)   (None, 128, 128, 64)  0           conv2d_3[0][0]
____________________________________________________________________________________________________
conv2d_4 (Conv2D)                (None, 128, 128, 64)  36928       alpha_dropout_2[0][0]
____________________________________________________________________________________________________
max_pooling2d_2 (MaxPooling2D)   (None, 64, 64, 64)    0           conv2d_4[0][0]
____________________________________________________________________________________________________
conv2d_5 (Conv2D)                (None, 64, 64, 64)    36928       max_pooling2d_2[0][0]
____________________________________________________________________________________________________
alpha_dropout_3 (AlphaDropout)   (None, 64, 64, 64)    0           conv2d_5[0][0]
____________________________________________________________________________________________________
conv2d_6 (Conv2D)                (None, 64, 64, 64)    36928       alpha_dropout_3[0][0]
____________________________________________________________________________________________________
concatenate_1 (Concatenate)      (None, 64, 64, 128)   0           conv2d_6[0][0]
                                                                   conv2d_6[0][0]
____________________________________________________________________________________________________
conv2d_7 (Conv2D)                (None, 64, 64, 64)    73792       concatenate_1[0][0]
____________________________________________________________________________________________________
alpha_dropout_4 (AlphaDropout)   (None, 64, 64, 64)    0           conv2d_7[0][0]
____________________________________________________________________________________________________
conv2d_8 (Conv2D)                (None, 64, 64, 64)    36928       alpha_dropout_4[0][0]
____________________________________________________________________________________________________
conv2d_transpose_1 (Conv2DTrans (None, 128, 128, 64)  36928       conv2d_8[0][0]
____________________________________________________________________________________________________
concatenate_2 (Concatenate)      (None, 128, 128, 128 0           conv2d_transpose_1[0][0]
                                                                   conv2d_4[0][0]
____________________________________________________________________________________________________
conv2d_9 (Conv2D)                (None, 128, 128, 64)  73792       concatenate_2[0][0]
____________________________________________________________________________________________________
alpha_dropout_5 (AlphaDropout)   (None, 128, 128, 64)  0           conv2d_9[0][0]
____________________________________________________________________________________________________
conv2d_10 (Conv2D)               (None, 128, 128, 64)  36928       alpha_dropout_5[0][0]
____________________________________________________________________________________________________
conv2d_transpose_2 (Conv2DTrans (None, 256, 256, 64)  36928       conv2d_10[0][0]
____________________________________________________________________________________________________
conv2d_transpose_3 (Conv2DTrans (None, 256, 256, 1)   577         conv2d_transpose_2[0][0]
====================================================================================================
Total params: 482,881
Trainable params: 482,881
Non-trainable params: 0
```

U-net based Model

conv2d_10 (Conv2D)	(None, 16, 16, 128)	147584	alpha_dropout_5[0][0]
batch_normalization_10 (BatchNo	(None, 16, 16, 128)	512	conv2d_10[0][0]
activation_10 (Activation)	(None, 16, 16, 128)	0	batch_normalization_10[0][0]
concatenate_1 (Concatenate)	(None, 16, 16, 256)	0	activation_10[0][0] activation_10[0][0]
alpha_dropout_6 (AlphaDropout)	(None, 16, 16, 256)	0	concatenate_1[0][0]
conv2d_11 (Conv2D)	(None, 16, 16, 16)	36880	alpha_dropout_6[0][0]
batch_normalization_11 (BatchNo	(None, 16, 16, 16)	64	conv2d_11[0][0]
activation_11 (Activation)	(None, 16, 16, 16)	0	batch_normalization_11[0][0]
conv2d_transpose_1 (Conv2DTrans	(None, 32, 32, 16)	2320	activation_11[0][0]
concatenate_2 (Concatenate)	(None, 32, 32, 80)	0	conv2d_transpose_1[0][0] activation_8[0][0]
alpha_dropout_7 (AlphaDropout)	(None, 32, 32, 80)	0	concatenate_2[0][0]
conv2d_12 (Conv2D)	(None, 32, 32, 32)	23072	alpha_dropout_7[0][0]
batch_normalization_12 (BatchNo	(None, 32, 32, 32)	128	conv2d_12[0][0]
activation_12 (Activation)	(None, 32, 32, 32)	0	batch_normalization_12[0][0]
conv2d_transpose_2 (Conv2DTrans	(None, 64, 64, 32)	9248	activation_12[0][0]
concatenate_3 (Concatenate)	(None, 64, 64, 64)	0	conv2d_transpose_2[0][0] activation_6[0][0]
alpha_dropout_8 (AlphaDropout)	(None, 64, 64, 64)	0	concatenate_3[0][0]
conv2d_13 (Conv2D)	(None, 64, 64, 64)	36928	alpha_dropout_8[0][0]
batch_normalization_13 (BatchNo	(None, 64, 64, 64)	256	conv2d_13[0][0]
activation_13 (Activation)	(None, 64, 64, 64)	0	batch_normalization_13[0][0]
conv2d_transpose_3 (Conv2DTrans	(None, 128, 128, 64)	36928	activation_13[0][0]
concatenate_4 (Concatenate)	(None, 128, 128, 80)	0	conv2d_transpose_3[0][0] activation_4[0][0]
alpha_dropout_9 (AlphaDropout)	(None, 128, 128, 80)	0	concatenate_4[0][0]
conv2d_14 (Conv2D)	(None, 128, 128, 128	92288	alpha_dropout_9[0][0]
batch_normalization_14 (BatchNo	(None, 128, 128, 128	512	conv2d_14[0][0]
activation_14 (Activation)	(None, 128, 128, 128	0	batch_normalization_14[0][0]
conv2d_transpose_4 (Conv2DTrans	(None, 256, 256, 128	147584	activation_14[0][0]
conv2d_transpose_5 (Conv2DTrans	(None, 256, 256, 1)	1153	conv2d_transpose_4[0][0]

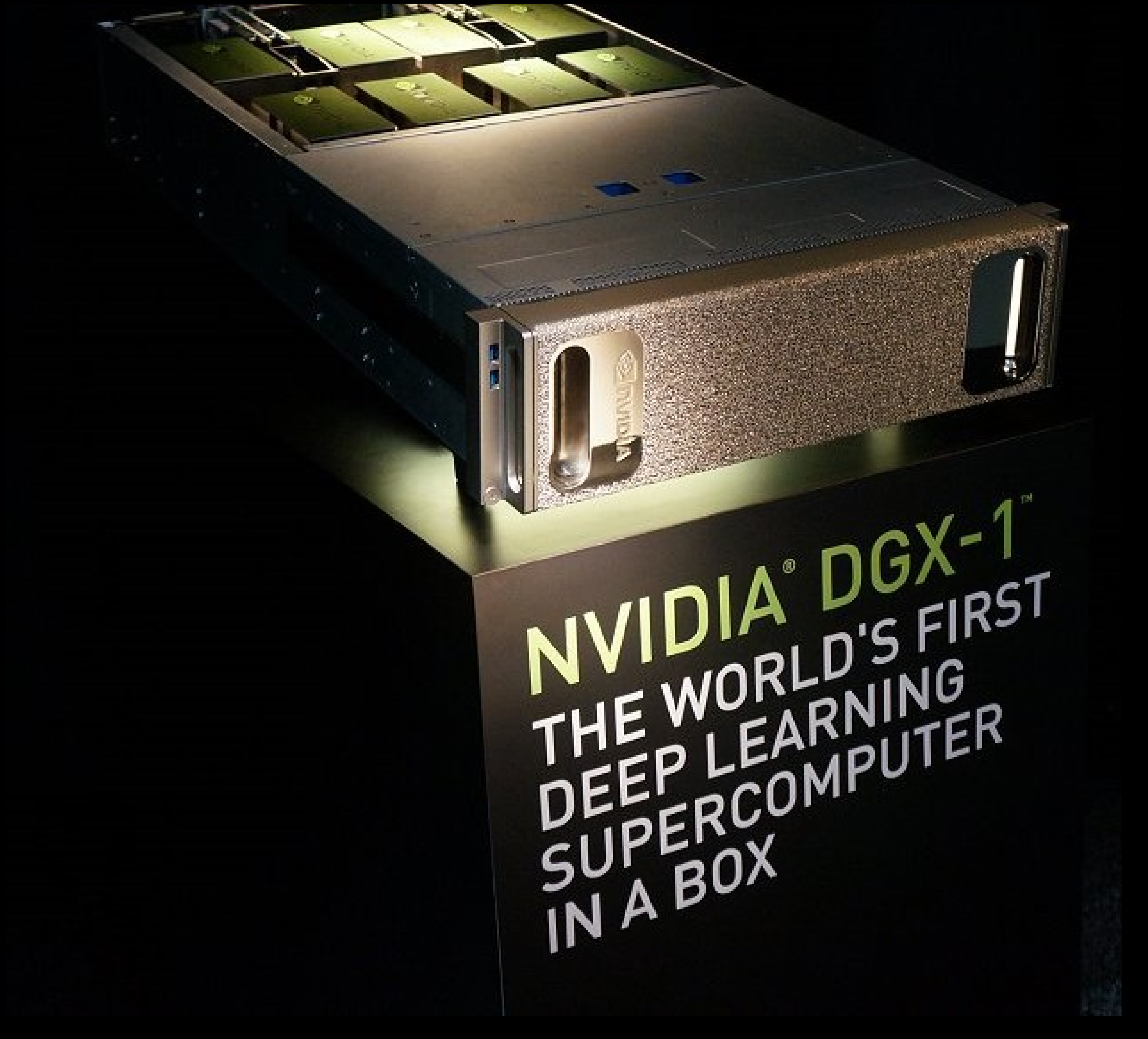
Total params: 684,465
Trainable params: 682,993
Non-trainable params: 1,472

Excerpt of the Planet model.

Note batch_norm layers and deeper structure (12 conv blocks in the encoder and decoder)

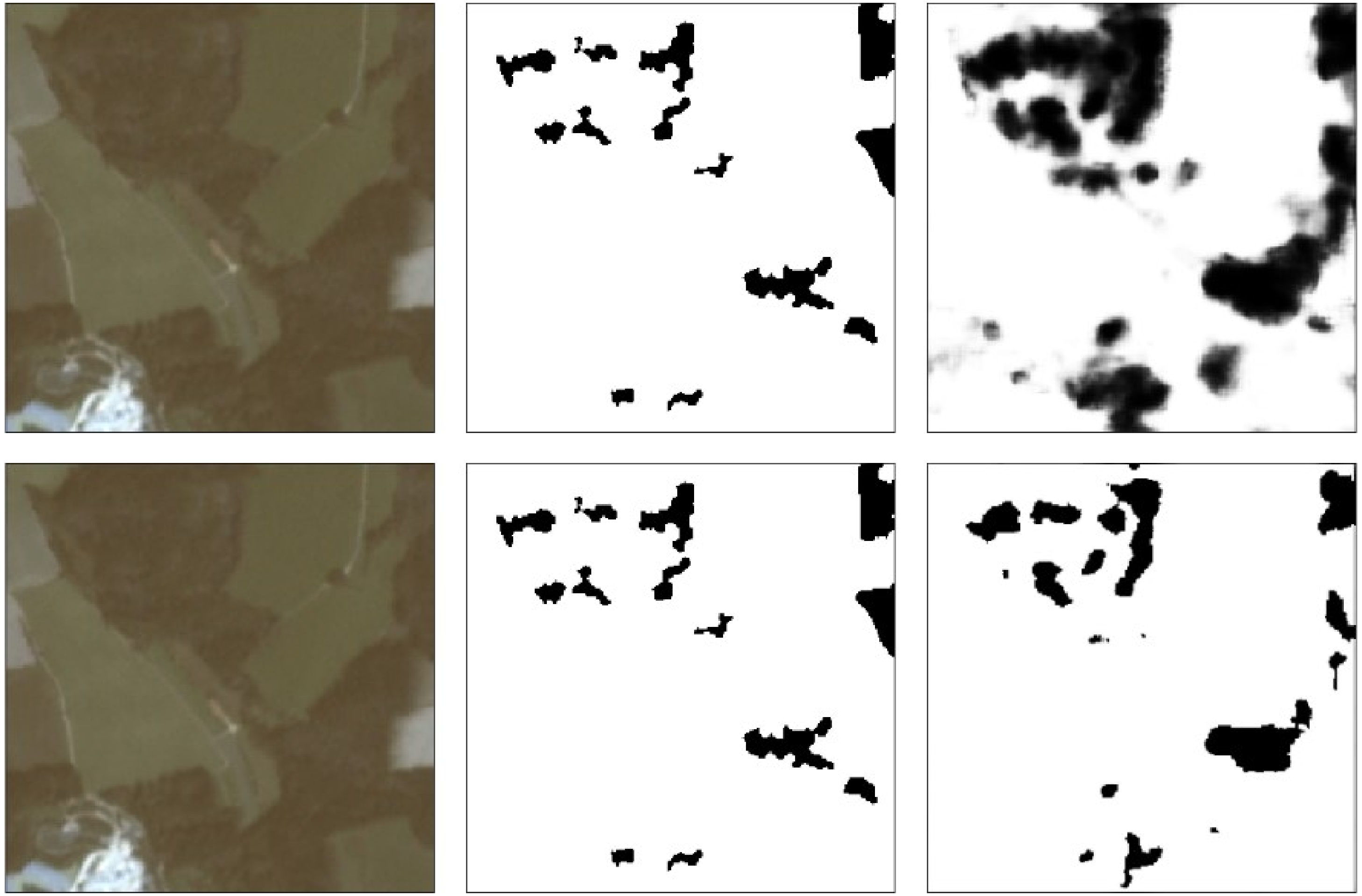
Trained with…

- LRZ’s supercomputer P100 with 8 GPUs
- NVIDIA DGX-1



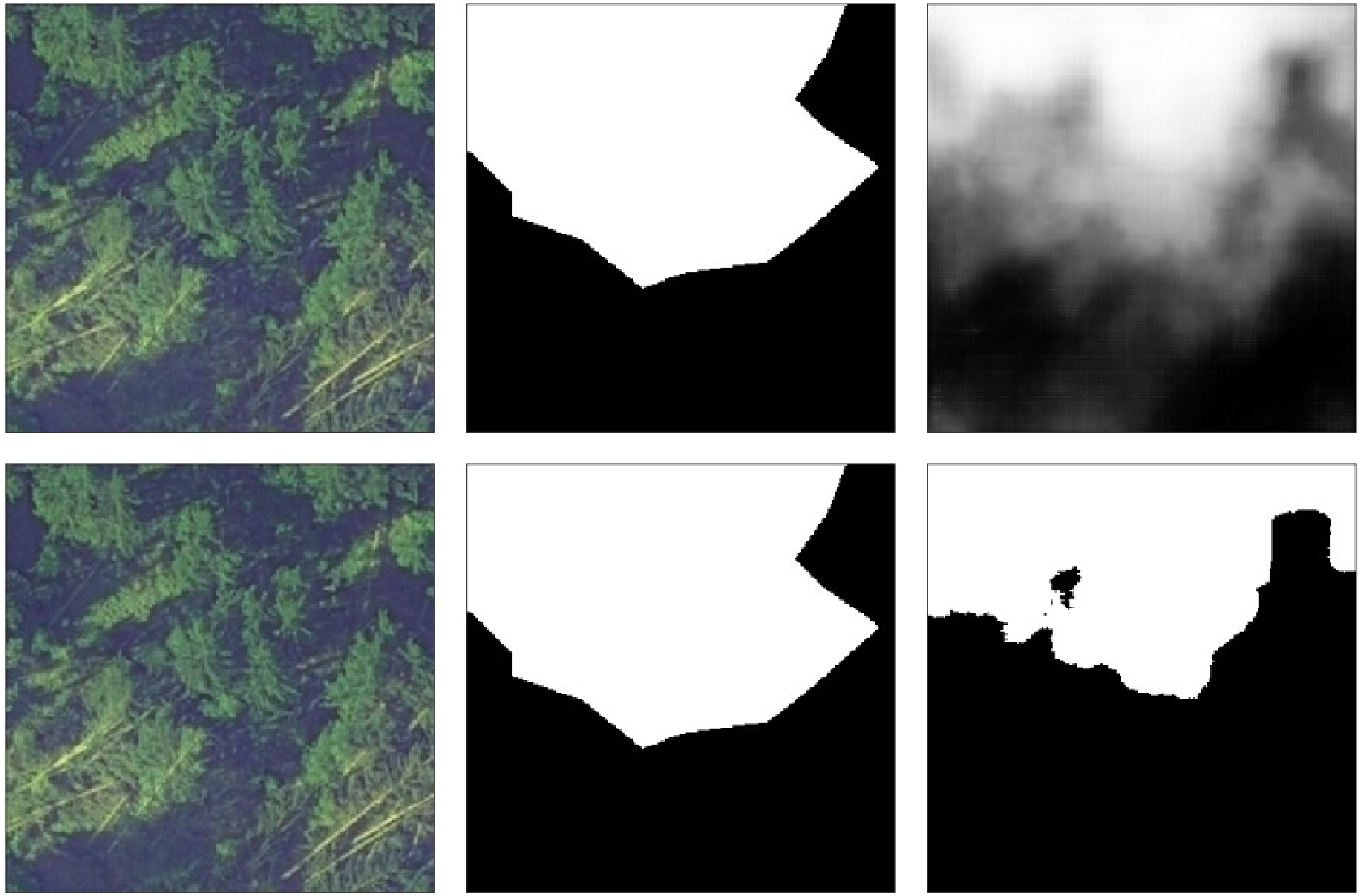
Results

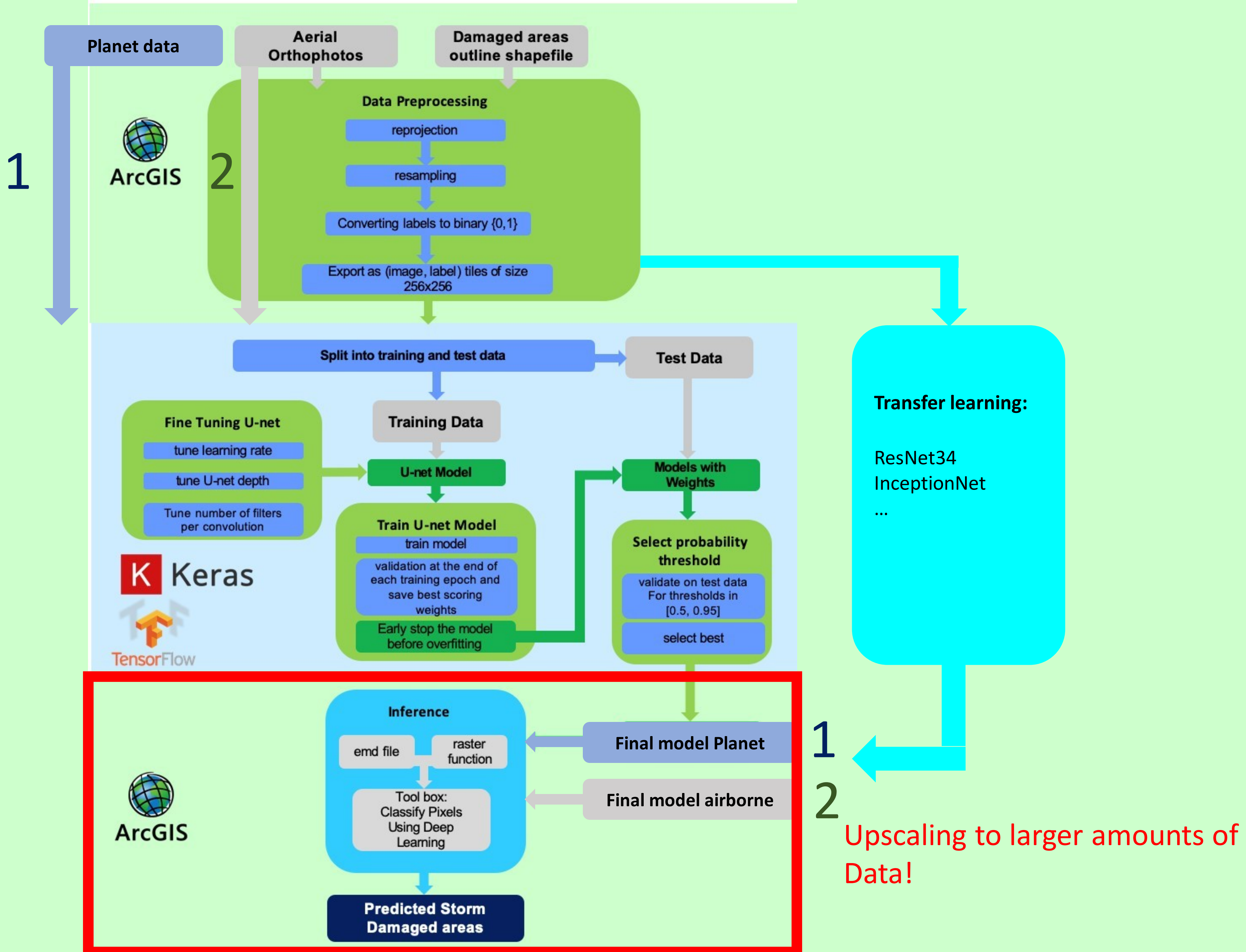
- Prediction of the damage using satellite images & with threshold of 0.05



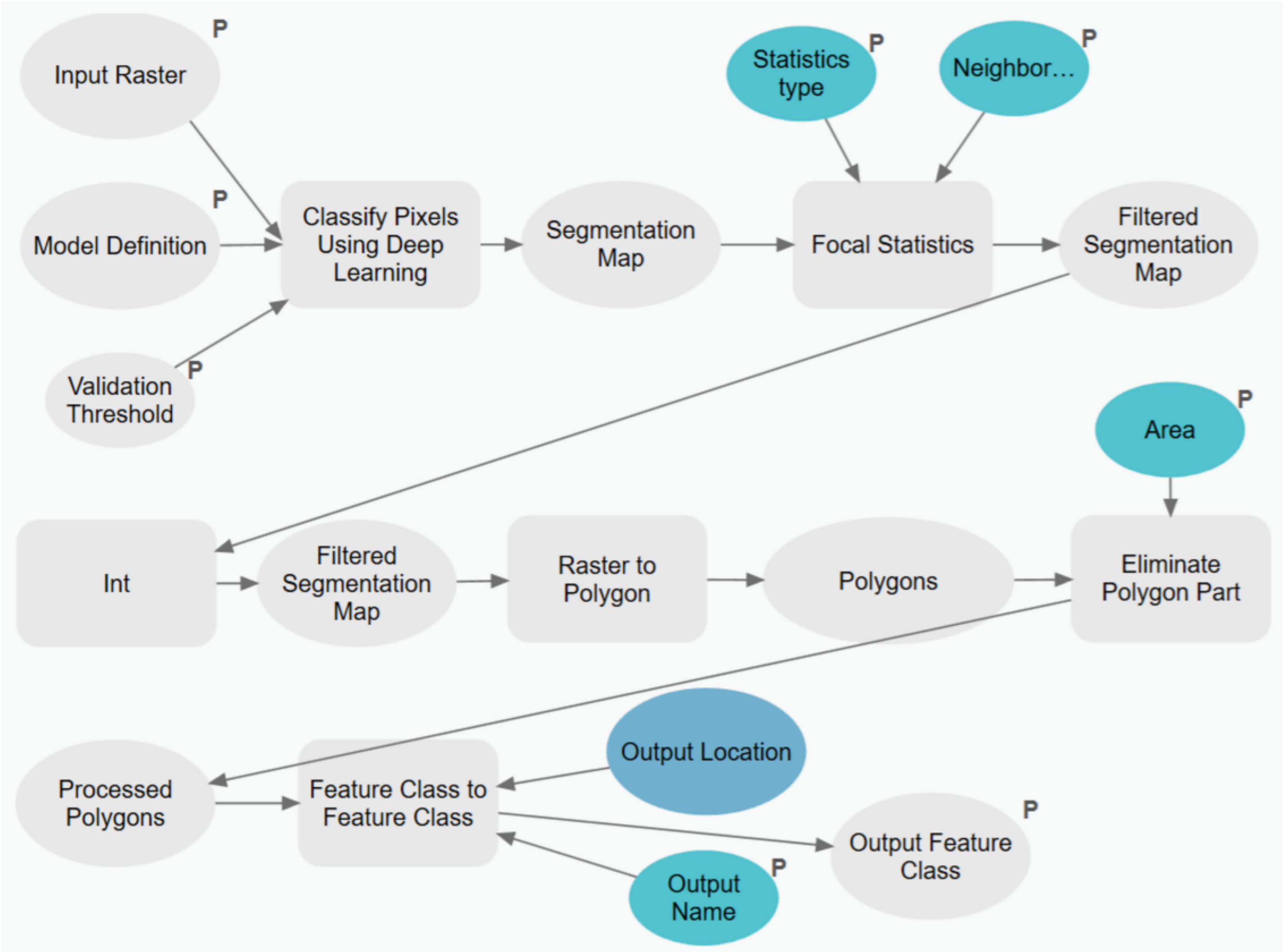
Results

- Prediction of the damage using ortho images & with threshold of 0.67



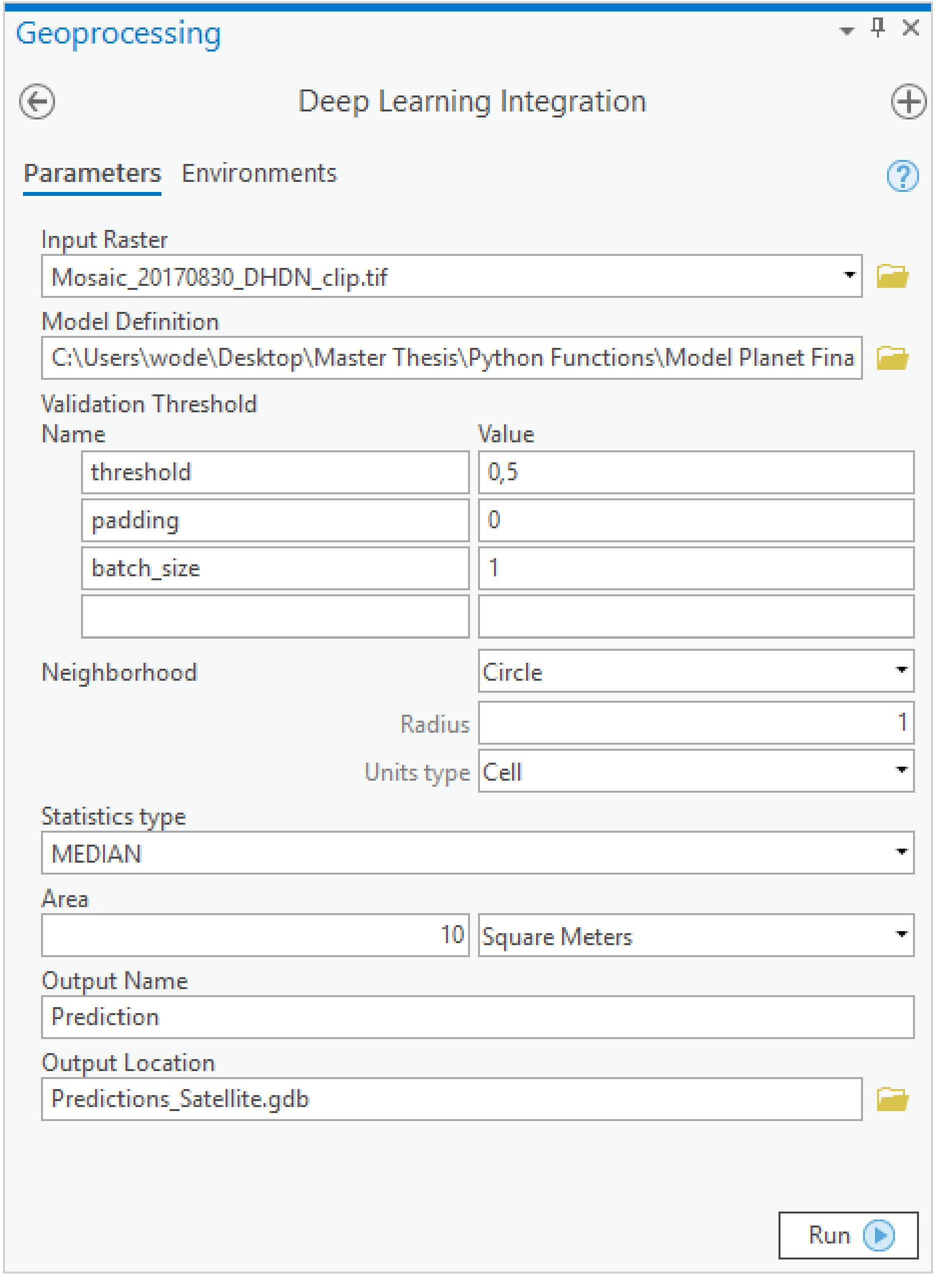

Planet data
Aerial Orthophotos
Damaged areas outline shapefile
1
ArcGIS
2
Data Preprocessing
reprojection
resampling
Converting labels to binary {0,1}
Export as (image, label) tiles of size 256x256
Split into training and test data
Test Data
Training Data
Fine Tuning U-net
tune learning rate
tune U-net depth
Tune number of filters per convolution
U-net Model
Models with Weights
Train U-net Model
train model
validation at the end of each training epoch and save best scoring weights
Early stop the model before overfitting
Select probability threshold
validate on test data For thresholds in [0.5, 0.95]
select best
Keras
TensorFlow
Transfer learning:
ResNet34
InceptionNet
…
Inference
emd file
raster function
Tool box: Classify Pixels Using Deep Learning
Final model Planet
Final model airborne
ArcGIS
Predicted Storm Damaged areas
1
2
Upscaling to larger amounts of Data!

Post-processing in model builder



Integration into ArcGIS Pro

- Prediction of the damage
- Smoothing of the prediction
- Conversion to polygons
- Elimination of small features
- Saving of the prediction

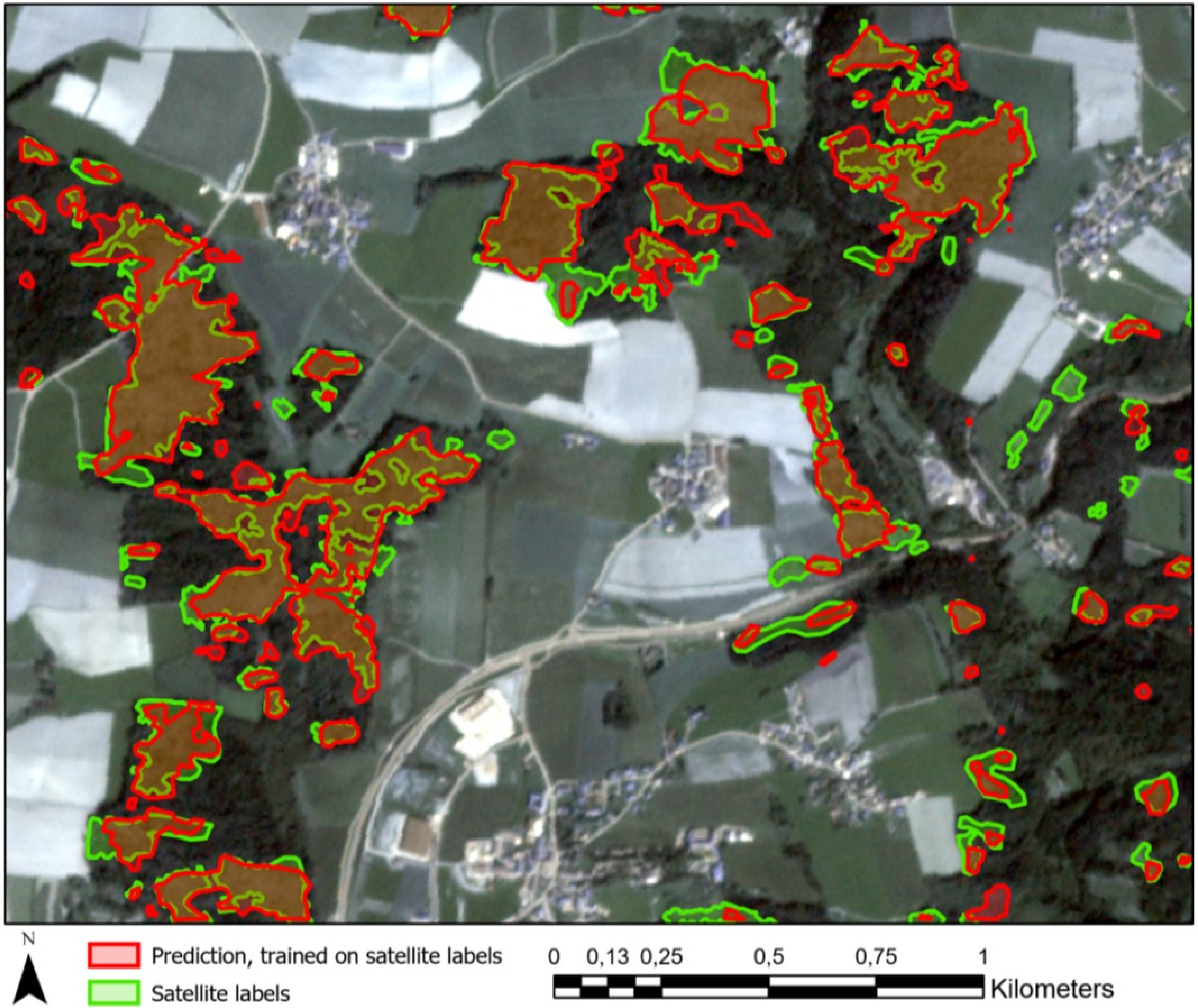


Results Planet data

Results Planet dove depending on label type

Note difference in thresholds → class imbalnces, false positives

	Test on satellite labels	Test on ortho labels
Training on satellite labels	0.5114, Threshold 0.05, Epoch 9	0.4951, Threshold 0.80, Epoch 4
Training on ortho labels	0.4576, Threshold 0.05, Epoch 8	0.5526, Threshold 0.26, Epoch 20

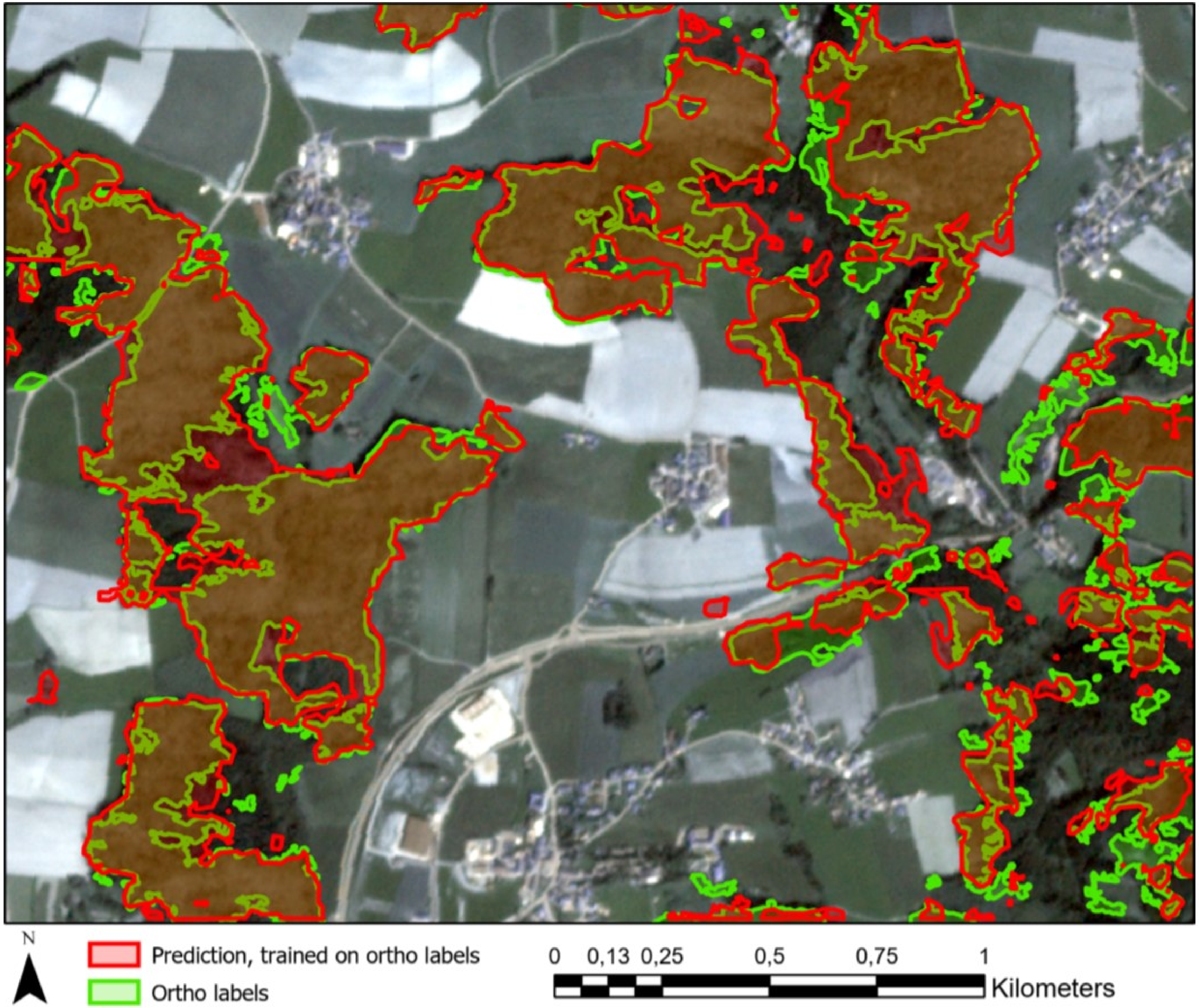

N
Prediction, trained on satellite labels
Satellite labels
0
0,13
0,25
0,5
0,75
1
Kilometers

Prediction:

23.41 km²

Labeling:

17.35 km²



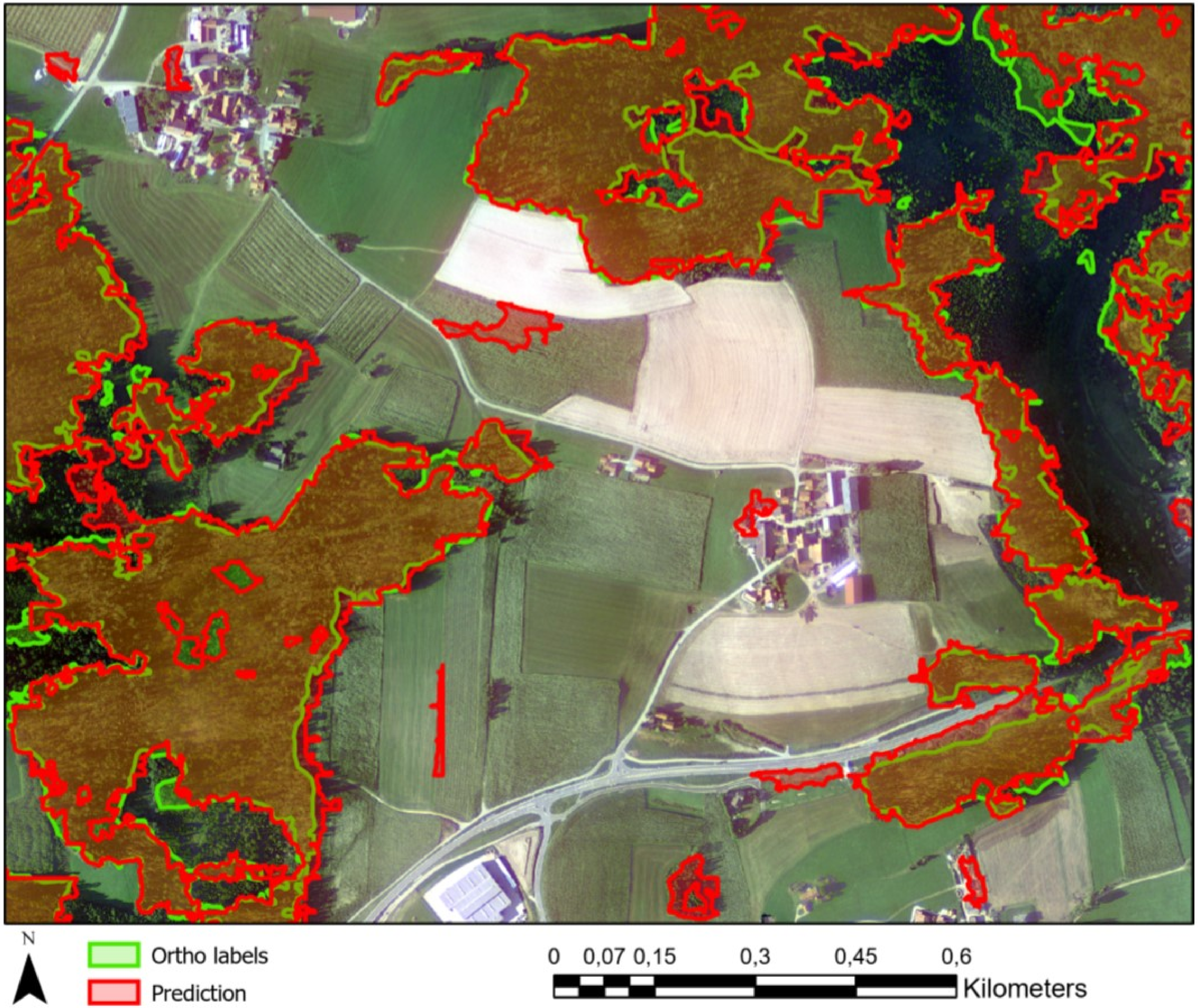
Results airborne data

Prediction:

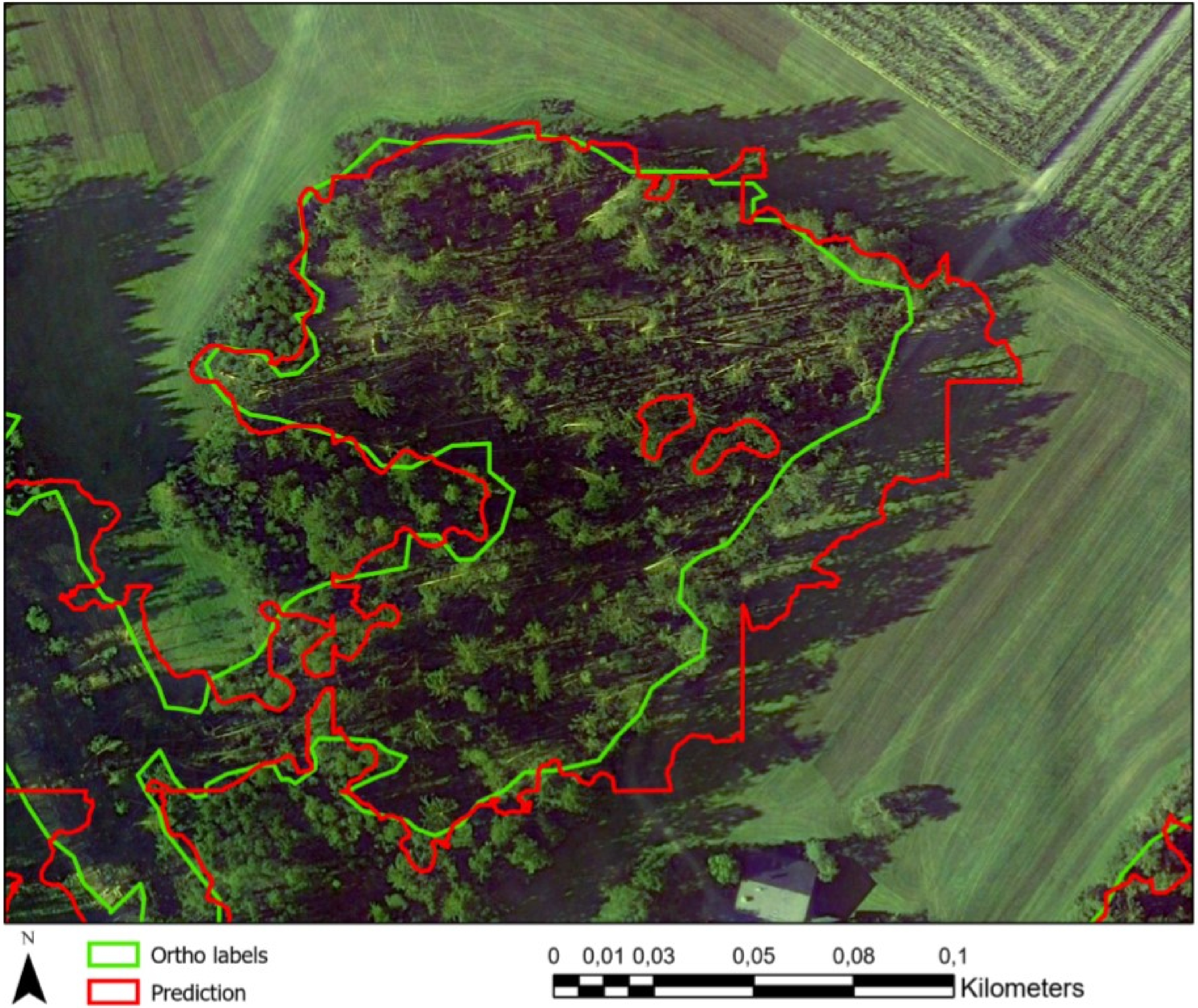
1.04 km²

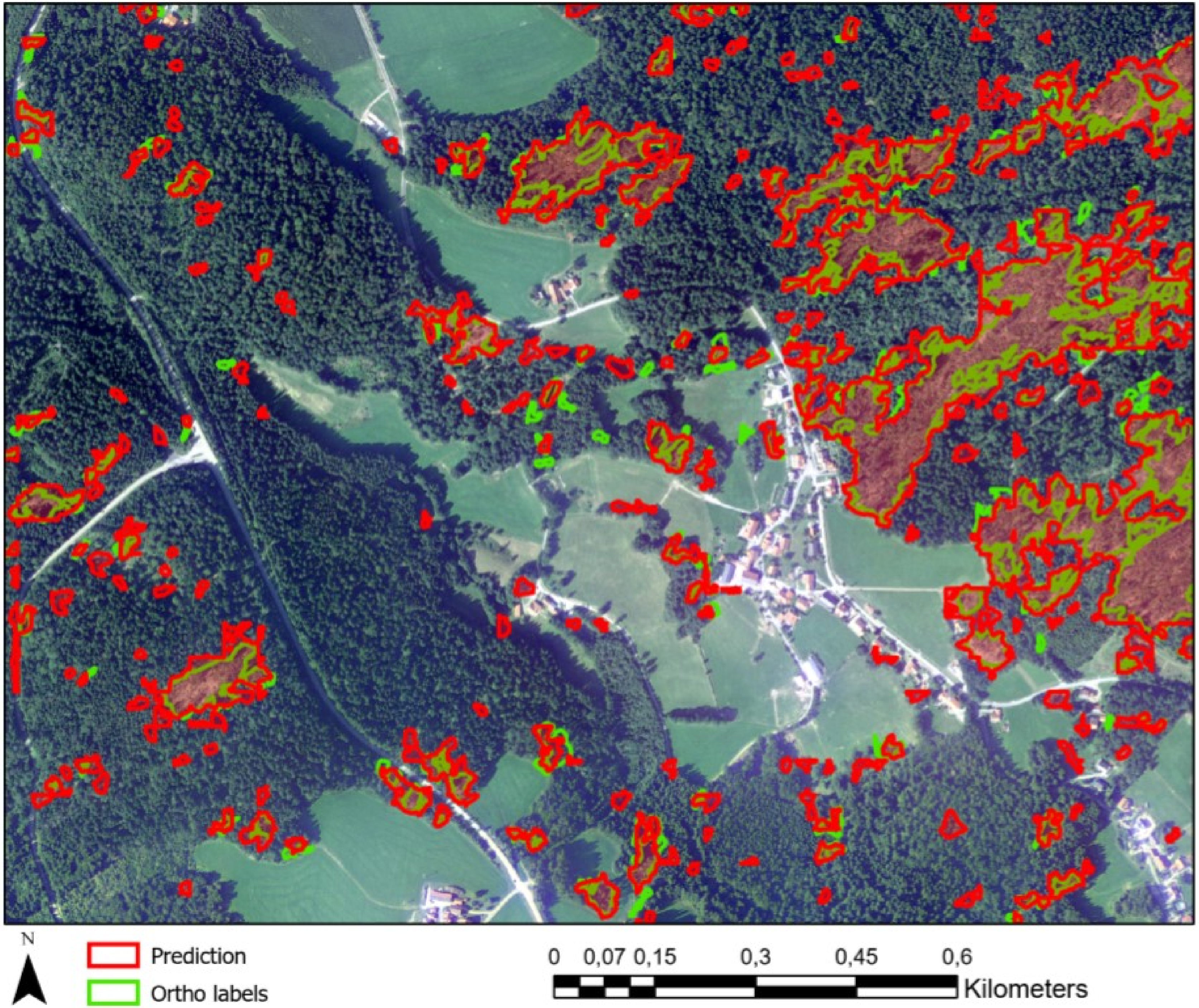
Labeling:

0.97 km²

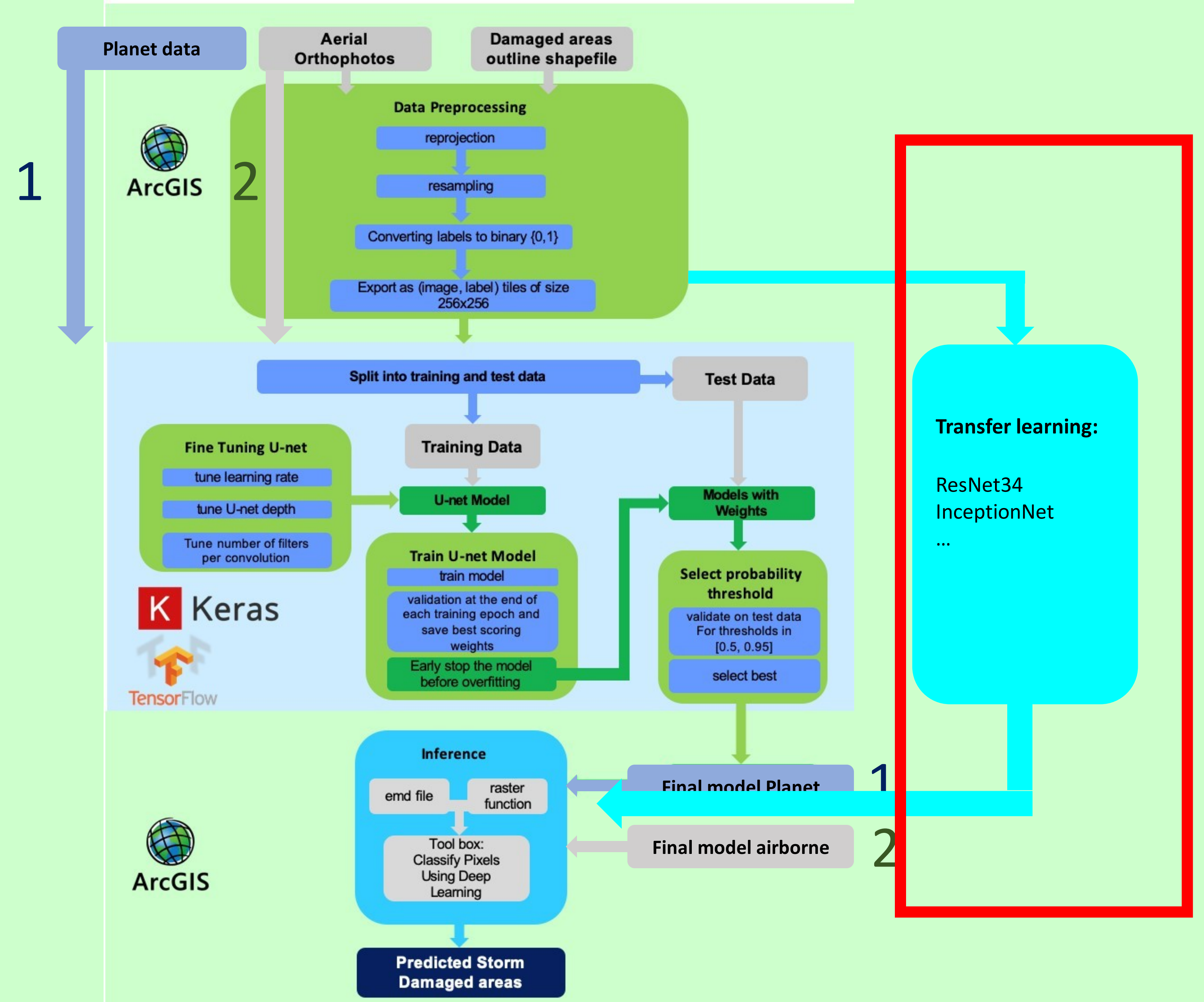


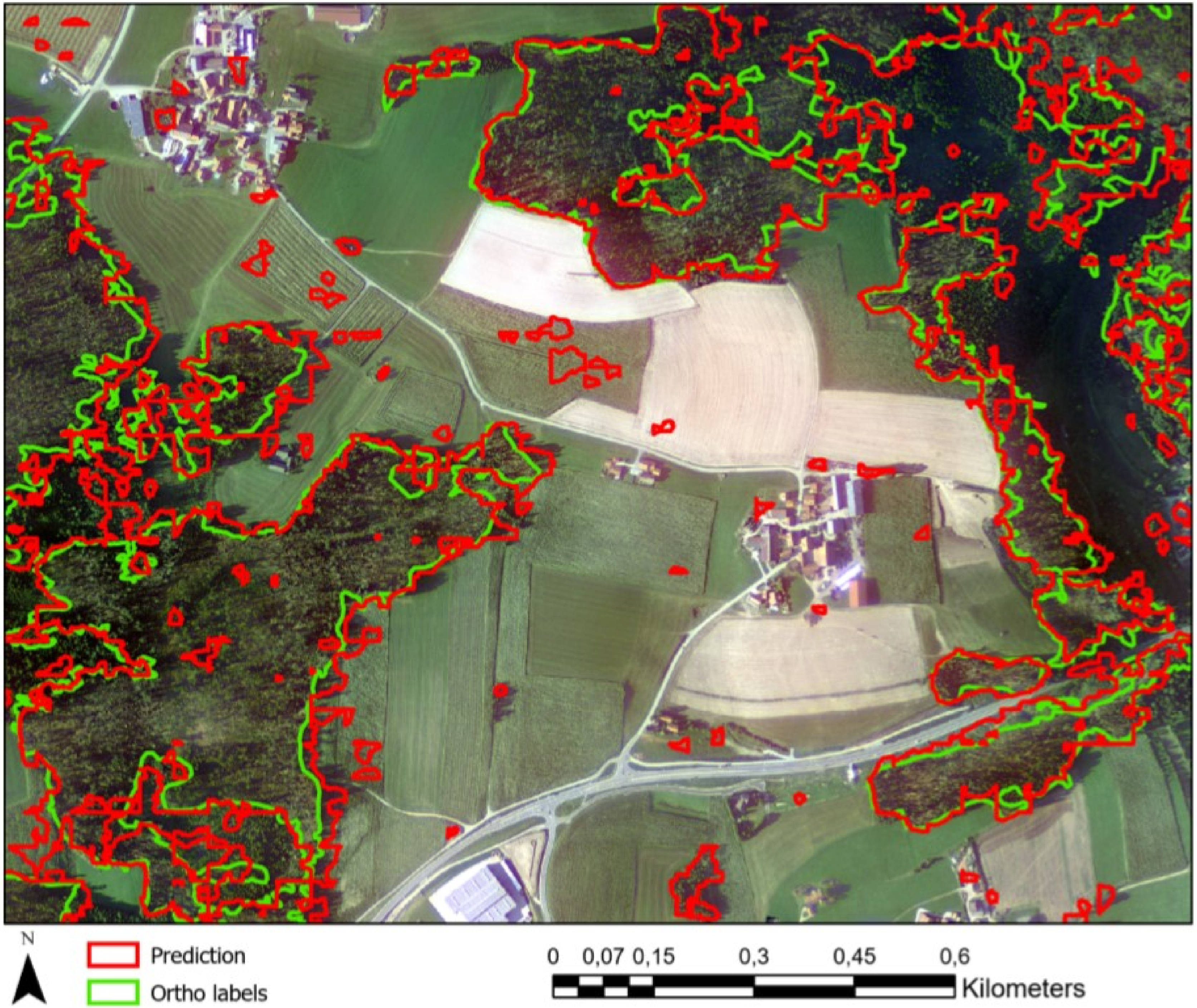
Accuracy (T=0.58)	86%
IoU	0.71


N
Ortho labels
Prediction
0 0,01 0,03 0,05 0,08 0,1
Kilometers


N
Prediction
Ortho labels
0
0,07
0,15
0,3
0,45
0,6
Kilometers

Transfer learning: VGG19


Planet data
Aerial Orthophotos
Damaged areas outline shapefile
1
ArcGIS
2
Data Preprocessing
reprojection
resampling
Converting labels to binary {0,1}
Export as (image, label) tiles of size 256x256
Split into training and test data
Test Data
Fine Tuning U-net
tune learning rate
tune U-net depth
Tune number of filters per convolution
Training Data
U-net Model
Models with Weights
Train U-net Model
train model
validation at the end of each training epoch and save best scoring weights
Early stop the model before overfitting
Select probability threshold
validate on test data For thresholds in [0.5, 0.95]
select best
Keras
TensorFlow
Transfer learning:
ResNet34
InceptionNet
...
Inference
emd file
raster function
Tool box: Classify Pixels Using Deep Learning
Final model Planet
1
Final model airborne
2
ArcGIS
Predicted Storm Damaged areas



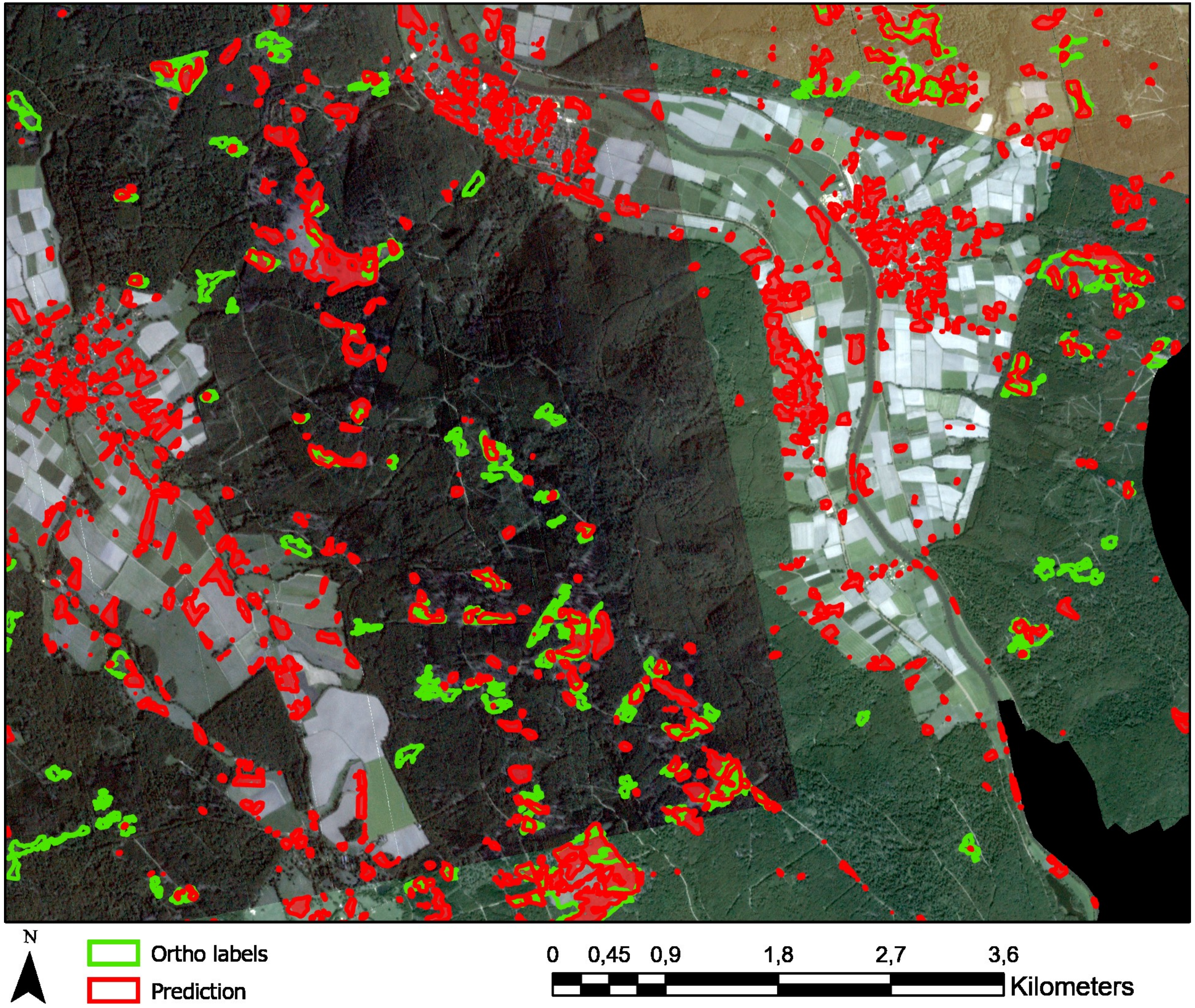
IoU (T=0.51)	0.75
Accuracy	84%

Application on a differnt area… (first tests…)

- Located in the state of Hesse
- Satellite images with 4.77 m resolution
- Aerial ortho images with 0.2 m resolution

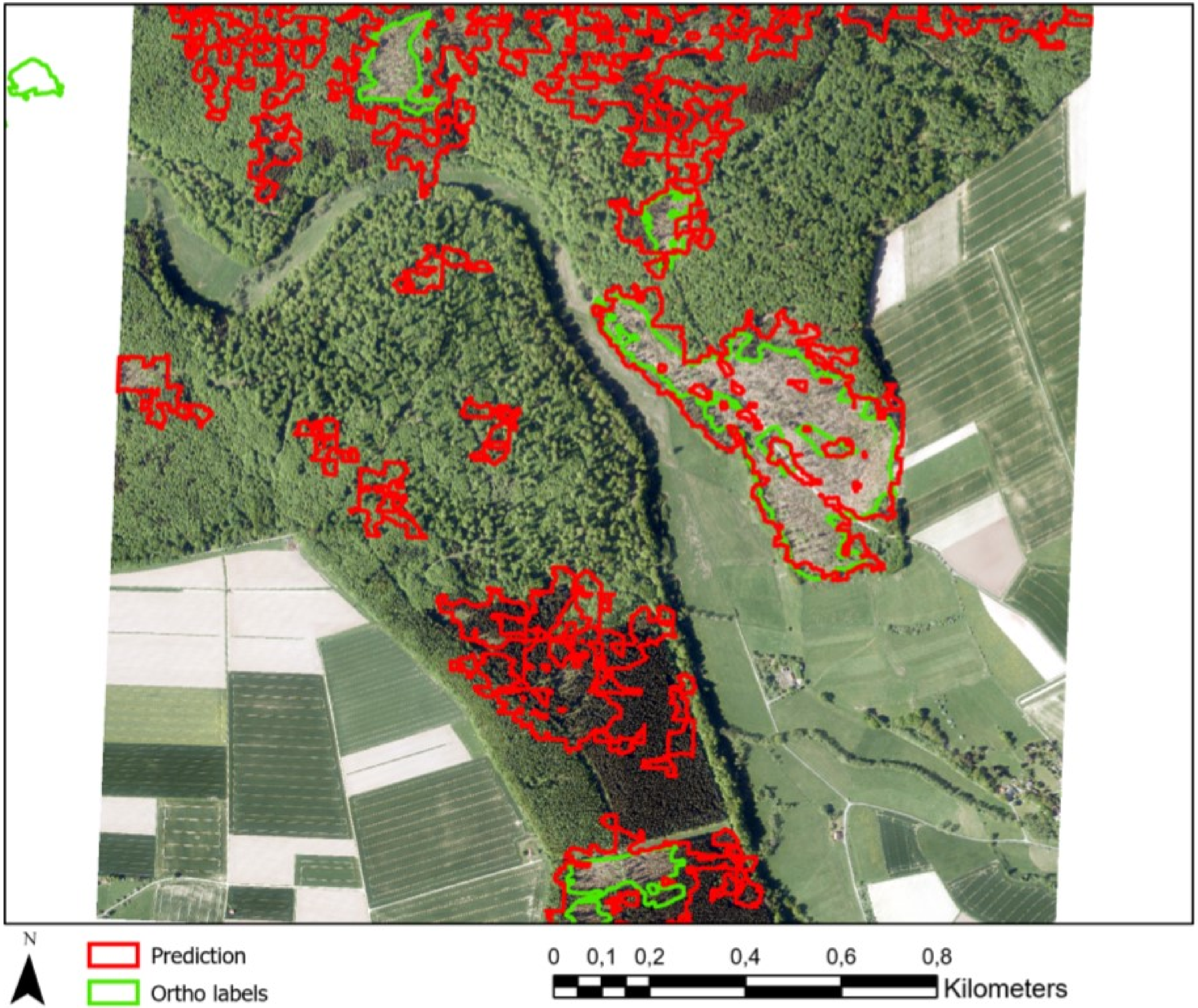
Results

- U-Net trained on satellite labels
- Ortho labels for comparison



VGG19

U-Net failed so far…



Technische Universität München

Conclusions

- U-Net is a powerful architecture for high-resolution remote sensing data
- Transfer learning great for high-resolution imagery
- Labelling errors might reduce accuracies
- The Integration of Deep Learning and ArcGIS provides a complete workflow for forest departments, including mobile mapping applications.
- Fast detection using Planet data and more accurate delineation using airborne data for disaster management
- Limitation: GPU availability, Data availability

THE SCIENCE OF WHERE

esri Deutschland