ISPRS 2D Semantic Labeling Contest

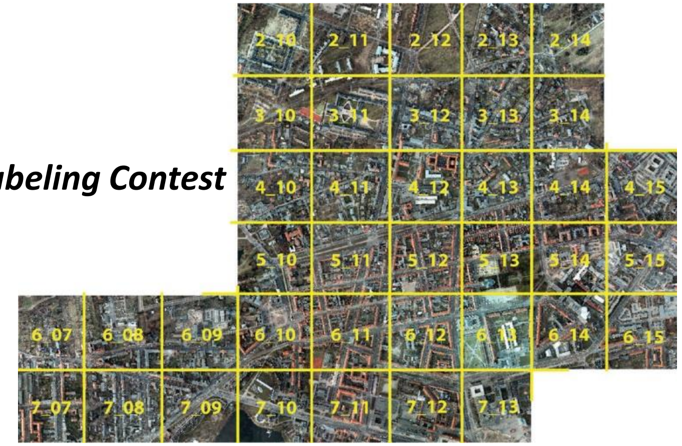
- 2D semantic segmentation that assigns labels to multiple object categories

- Acquired by airborne sensors
  - Very high resolution true ortho photo tiles
  - Digital surface models (DSMs) derived from dense image matching techniques

- Very heterogeneous appearance of objects
  - High intra-class variance and low inter-class variance

Vaihingen
Town with many detached buildings and small multi story buildings

[26] 2D Semantic Labeling Contest



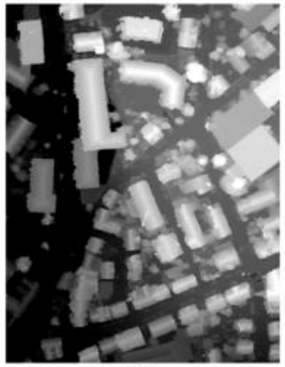
Potsdam
Historic city with large building blocks, narrow streets and dense settlement structure

Vaihingen Dataset (1)

- 33 patches of different sizes with 9 cm spatial resolution
- Manually classified into six land cover classes
  - *Impervious surfaces, Building, Low vegetation, Tree, Clutter/background*

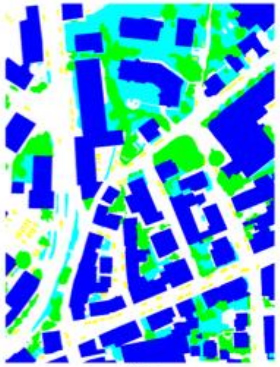
True orthophoto

Near infrared
Red
Green

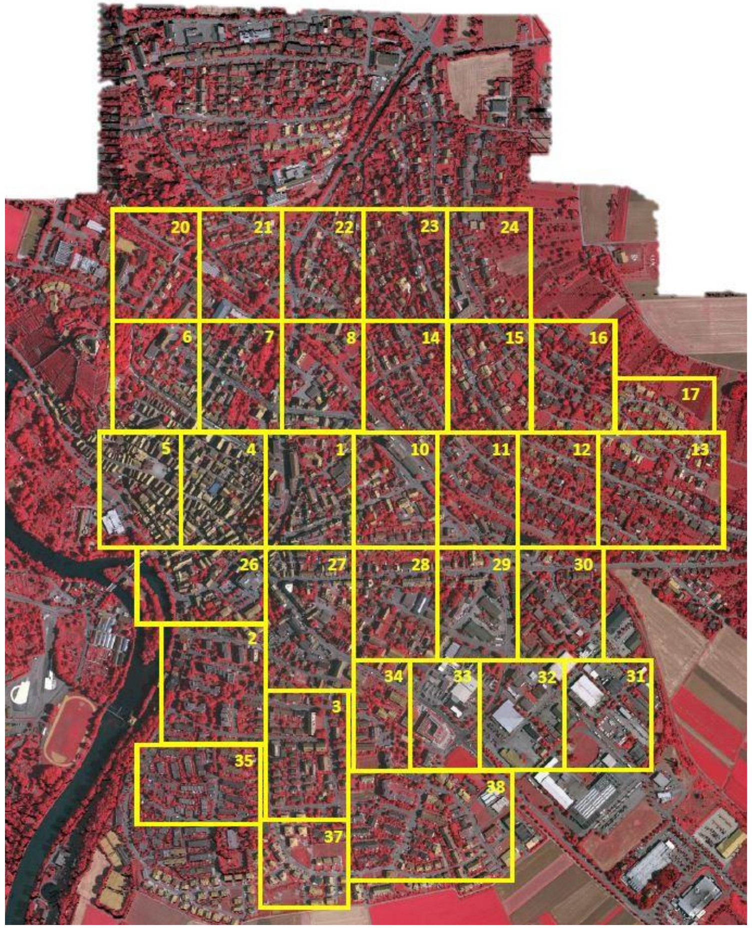


DSM

One band
Grey levels mapped to DSM heights



Groundtruth



Vaihingen Dataset (2)

- The groundtruth is provided for only 16 patches
- For the remaining scenes is it unreleased and used for evaluation of submitted results

TOP	DSM	N_{col}	N_{row}	GT
top_mosaic_09cm_area1	dsm_09cm_matching_area1	1919	2569	top_mosaic_09cm_area1
top_mosaic_09cm_area2	dsm_09cm_matching_area2	2428	2767	
top_mosaic_09cm_area3	dsm_09cm_matching_area3	2006	3007	top_mosaic_09cm_area3
top_mosaic_09cm_area4	dsm_09cm_matching_area4	1887	2557	
top_mosaic_09cm_area5	dsm_09cm_matching_area5	1887	2557	top_mosaic_09cm_area5
top_mosaic_09cm_area6	dsm_09cm_matching_area6	1887	2557	
top_mosaic_09cm_area7	dsm_09cm_matching_area7	1887	2557	top_mosaic_09cm_area7
top_mosaic_09cm_area8	dsm_09cm_matching_area8	1887	2557	
top_mosaic_09cm_area10	dsm_09cm_matching_area10	1887	2557	
top_mosaic_09cm_area11	dsm_09cm_matching_area11	1893	2566	top_mosaic_09cm_area11
top_mosaic_09cm_area12	dsm_09cm_matching_area12	1922	2575	
top_mosaic_09cm_area13	dsm_09cm_matching_area13	2818	2558	top_mosaic_09cm_area13
top_mosaic_09cm_area14	dsm_09cm_matching_area14	1919	2565	
top_mosaic_09cm_area15	dsm_09cm_matching_area15	1919	2565	top_mosaic_09cm_area15
top_mosaic_09cm_area16	dsm_09cm_matching_area16	1919	2565	
top_mosaic_09cm_area17	dsm_09cm_matching_area17	2336	1281	top_mosaic_09cm_area17
top_mosaic_09cm_area20	dsm_09cm_matching_area20	1866	2315	
top_mosaic_09cm_area21	dsm_09cm_matching_area21	1903	2546	top_mosaic_09cm_area21
top_mosaic_09cm_area22	dsm_09cm_matching_area22	1903	2546	
top_mosaic_09cm_area23	dsm_09cm_matching_area23	1903	2546	top_mosaic_09cm_area23
top_mosaic_09cm_area24	dsm_09cm_matching_area24	1903	2546	
top_mosaic_09cm_area26	dsm_09cm_matching_area26	2995	1783	top_mosaic_09cm_area26
top_mosaic_09cm_area27	dsm_09cm_matching_area27	1917	3313	
top_mosaic_09cm_area28	dsm_09cm_matching_area28	1917	2567	top_mosaic_09cm_area28
top_mosaic_09cm_area29	dsm_09cm_matching_area29	1917	2563	
top_mosaic_09cm_area30	dsm_09cm_matching_area30	1934	2563	top_mosaic_09cm_area30
top_mosaic_09cm_area31	dsm_09cm_matching_area31	1980	2555	
top_mosaic_09cm_area32	dsm_09cm_matching_area32	1980	2555	top_mosaic_09cm_area32
top_mosaic_09cm_area33	dsm_09cm_matching_area33	1581	2555	
top_mosaic_09cm_area34	dsm_09cm_matching_area34	1388	2555	top_mosaic_09cm_area34
top_mosaic_09cm_area35	dsm_09cm_matching_area35	2805	1884	
top_mosaic_09cm_area37	dsm_09cm_matching_area37	1996	1995	top_mosaic_09cm_area37
top_mosaic_09cm_area38	dsm_09cm_matching_area38	3816	2550	

ISPRS Semantic Labeling Contest (2D): Results

Click here for a description of evaluation measures

Vaihingen: 2D Labelling challenge

All quality measures except for *overall* are F1 scores in [%] using the reference with eroded boundaries.
Mouse over the column headings will display more information.

Abbrev.	imp surf	building	low_veg	tree	car	Overall	Strategy	Deta
SVL_1	86.3	90.8	78.2	84.2	56.8	84.7	s	D F
SVL_2	82.1	82.8	71.6	81.6	51.9	79.4	s	D F
SVL_3	86.6	91.0	77.0	85.0	55.6	84.8	s	D F
SVL_4	86.1	90.9	77.6	84.9	59.9	84.7	s	D F
SVL_5	86.1	90.3	75.6	84.7	45.9	84.0	s	D F
SVL_6	86.0	90.2	75.6	82.1	45.4	83.2	s	D F
ADL_1	88.1	92.0	79.0	86.5	59.0	86.1	s	D F
ADL_2	89.0	93.0	81.0	87.8	59.5	87.3	s	D F
ADL_3	89.5	93.2	82.3	88.2	63.3	88.0	s	D F
UT_Mev	84.3	88.7	74.5	82.0	9.9	81.8	u	D F
HUST	86.9	92.0	78.3	86.9	29.0	85.9	s	D F
ONE_1	83.0	84.4	75.0	84.9	44.7	81.0	s	D F

[26] 2D Semantic Labeling Contest

"Participants shall use all data with ground truth for training or internal evaluation of their method"

Vaihingen Dataset in JURECA

- Access JURECA: **$ ssh –X train???@jureca.fz-juelich.de**
- Data location: ***/homea/hpclab/train001/data/vaihingen/***

```
[train002@jrl05 ~]$ ls /homea/hpclab/train001/data/vaihingen/
vaihingen_11.hdf5  vaihingen_1.hdf5   vaihingen_28.hdf5  vaihingen_37.hdf5
vaihingen_13.hdf5  vaihingen_21.hdf5  vaihingen_30.hdf5  vaihingen_3.hdf5
vaihingen_15.hdf5  vaihingen_23.hdf5  vaihingen_32.hdf5  vaihingen_5.hdf5
vaihingen_17.hdf5  vaihingen_26.hdf5  vaihingen_34.hdf5  vaihingen_7.hdf5
[train002@jrl05 ~]$
```

- HDF5 files creation:

  str = 'Vaihingen_1.hdf5';
  h5write(str , '/x_1', cat(3, Near Infrared, Red, Green, Normalized DSM,NDVI*))*
  h5write(str , '/y_1', Groundtruth)
  h5write(str , '/m_1', Boundaries)

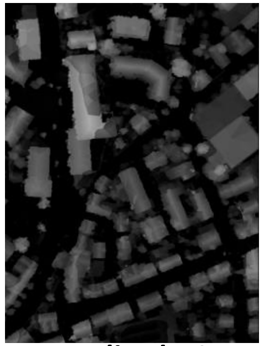
E.G., Vaihingen_1.hdf5

/x_1

Near Infrared

Red

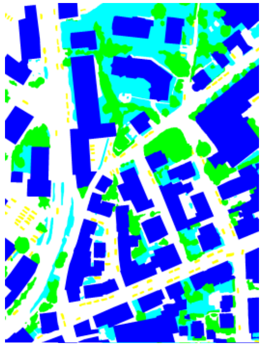
Green



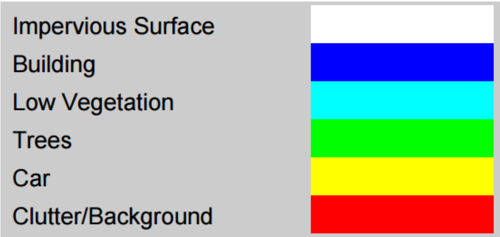
Normalized DSM

NDVI

/y_1



Groundtruth



/m_1

- Uncertainty associated with the boundary of objects in the groundtruth
- These boundaries are ignored for the evaluation
- Groundtruth images with boundary pixels (in a 3 pixel radius) provided

Approach for Training and Validation Set Generation

256

...

256 → **Not a random number, will be later explained …**

- Generate dataset of 256x256 sized image patches

```
def main(arguments):

    data_path = arguments[1]
    output_path = arguments[2]

    # files used for training:
    training_nums = [1, 3, 5, 7, 11, 13, 17, 21, 26, 28, 34, 37]

    # files used for validation:
    validation_nums = [30, 32]

    # generate and save the training and validation set:
    overlap = 0.6
```

Get the Code and Test the Python Environment

1. Get a copy of the folder **/homea/hpclab/train001/tools/resnet50-fcn**
   - Create a new folder in your local path **$ ~/semseg**
   - Copy **$ cp -R /homea/hpclab/train001/tools/resnet50-fcn ~/semseg/**

2. All modules and python packages have been already prepared
   - Just run -> **$ module restore dl_tutorial**
   - How was it setup?
     - $ module use /usr/local/software/jureca/OtherStages
     - $ ml Stages/Devel-2017a
     - $ ml GCC/5.4.0 MVAPICH2
     - $ ml TensorFlow/1.4.0-Python-2.7.13
     - $ pip install --user virtualenv
     - $ pip install --user h5py
     - $ pip install --user keras
     - $ pip install --user sklearn
     - $ module store dl_tutorial

3. Check if Keras is available
   - **$ python**
   - **>>> import keras**

```
[train002@jrl12 code]$ python
Python 2.7.13 (default, Feb 14 2018, 10:29:12)
[GCC 5.4.0] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import keras
/homea/hpclab/train002/.local/lib/python2.7/site-packages/h5py/__init__.py:36: FutureWarnin
g: Conversion of the second argument of issubdtype from `float` to `np.floating` is depreca
ted. In future, it will be treated as `np.float64 == np.dtype(float).type`.
  from ._conv import register_converters as _register_converters
Using TensorFlow backend.
```

Generate the Training and Validation Set

4. Use the function ***~/semseg*/resnet50-fcn/data_io.py**

   - *If you run **$ python data_io.py***

```
[train002@jrl12 resnet50-fcn]$ python data_io.py
/homea/hpclab/train002/.local/lib/python2.7/site-packages/h5py/__init__.py:36: FutureWarning: Conve
 future, it will be treated as `np.float64 == np.dtype(float).type`.
  from ._conv import register_converters as _register_converters

***********************************
Two paremeters need to be specified:
1. Location of the input image patches (e.g., /homea/hpclab/train001/data/vaihingen/ )
2. Location to write training and validation sets (e.g., /homea/hpclab/train002/semseg/vaihingen/ )
***********************************
```

- Create a new folder where to save the sets **$ makdir ~/semseg/vaihingen**
- Run the function:
  - **$ python data_io.py /homea/hpclab/train001/data/vaihingen/ ~/semseg/vaihingen/**

The Outcome

- The patches are assigned to the training and validation sets

```
[train002@jrl03 resnet50-fcn]$ python data_io.py /homea/hpclab/train001/data/vaihingen/ ~/semseg/vaihingen/
/homea/hpclab/train002/.local/lib/python2.7/site-packages/h5py/__init__.py:36: FutureWarning: Conversion of t
econd argument of issubdtype from `float` to `np.floating` is deprecated. In future, it will be treated as `n
oat64 == np.dtype(float).type`.
  from ._conv import register_converters as _register_converters
/homea/hpclab/train001/data/vaihingen/vaihingen_1.hdf5
/homea/hpclab/train001/data/vaihingen/vaihingen_3.hdf5
/homea/hpclab/train001/data/vaihingen/vaihingen_5.hdf5
/homea/hpclab/train001/data/vaihingen/vaihingen_7.hdf5
/homea/hpclab/train001/data/vaihingen/vaihingen_11.hdf5
/homea/hpclab/train001/data/vaihingen/vaihingen_13.hdf5
/homea/hpclab/train001/data/vaihingen/vaihingen_17.hdf5
/homea/hpclab/train001/data/vaihingen/vaihingen_21.hdf5
/homea/hpclab/train001/data/vaihingen/vaihingen_26.hdf5
/homea/hpclab/train001/data/vaihingen/vaihingen_28.hdf5
/homea/hpclab/train001/data/vaihingen/vaihingen_34.hdf5
/homea/hpclab/train001/data/vaihingen/vaihingen_37.hdf5
Generated 2083 samples!
/homea/hpclab/train001/data/vaihingen/vaihingen_30.hdf5
/homea/hpclab/train001/data/vaihingen/vaihingen_32.hdf5
Generated 368 samples!
```

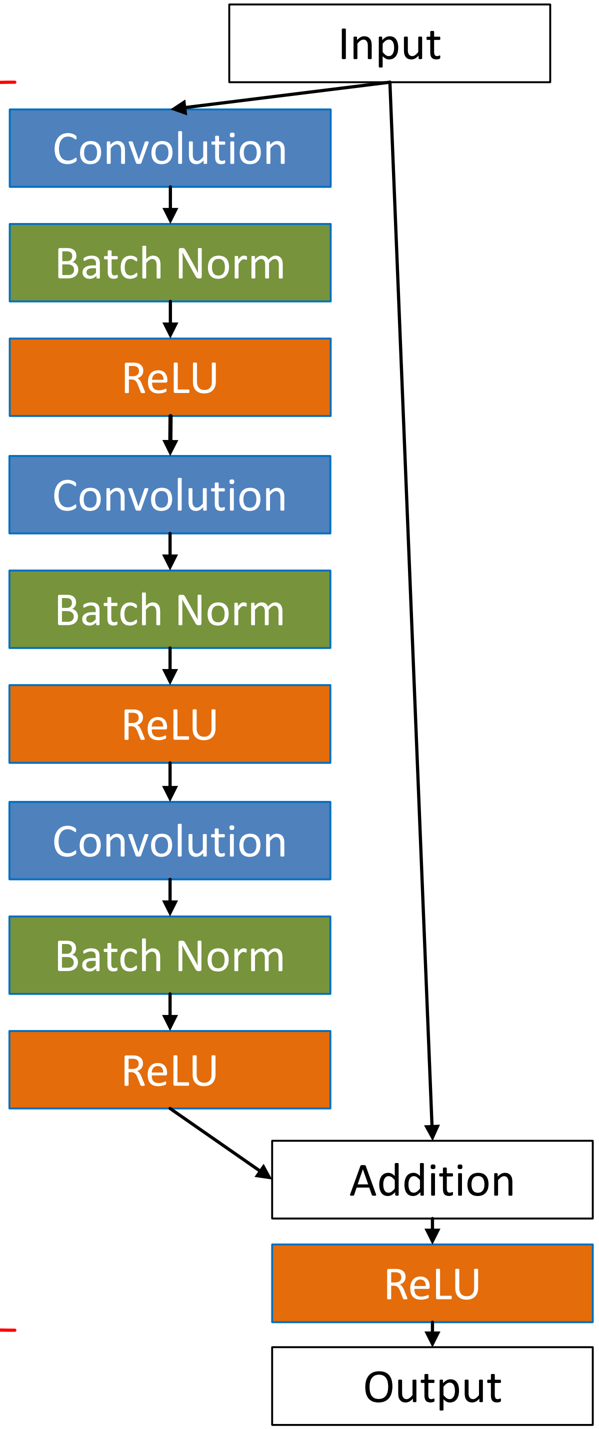
  - These sets will be used for training the network

```
[train002@jrl10 vaihingen]$ pwd
/homea/hpclab/train002/semseg/vaihingen
[train002@jrl10 vaihingen]$ ls
vaihingen_train.hdf5  vaihingen_val.hdf5
[train002@jrl10 vaihingen]$
```

ResNet50 FCN

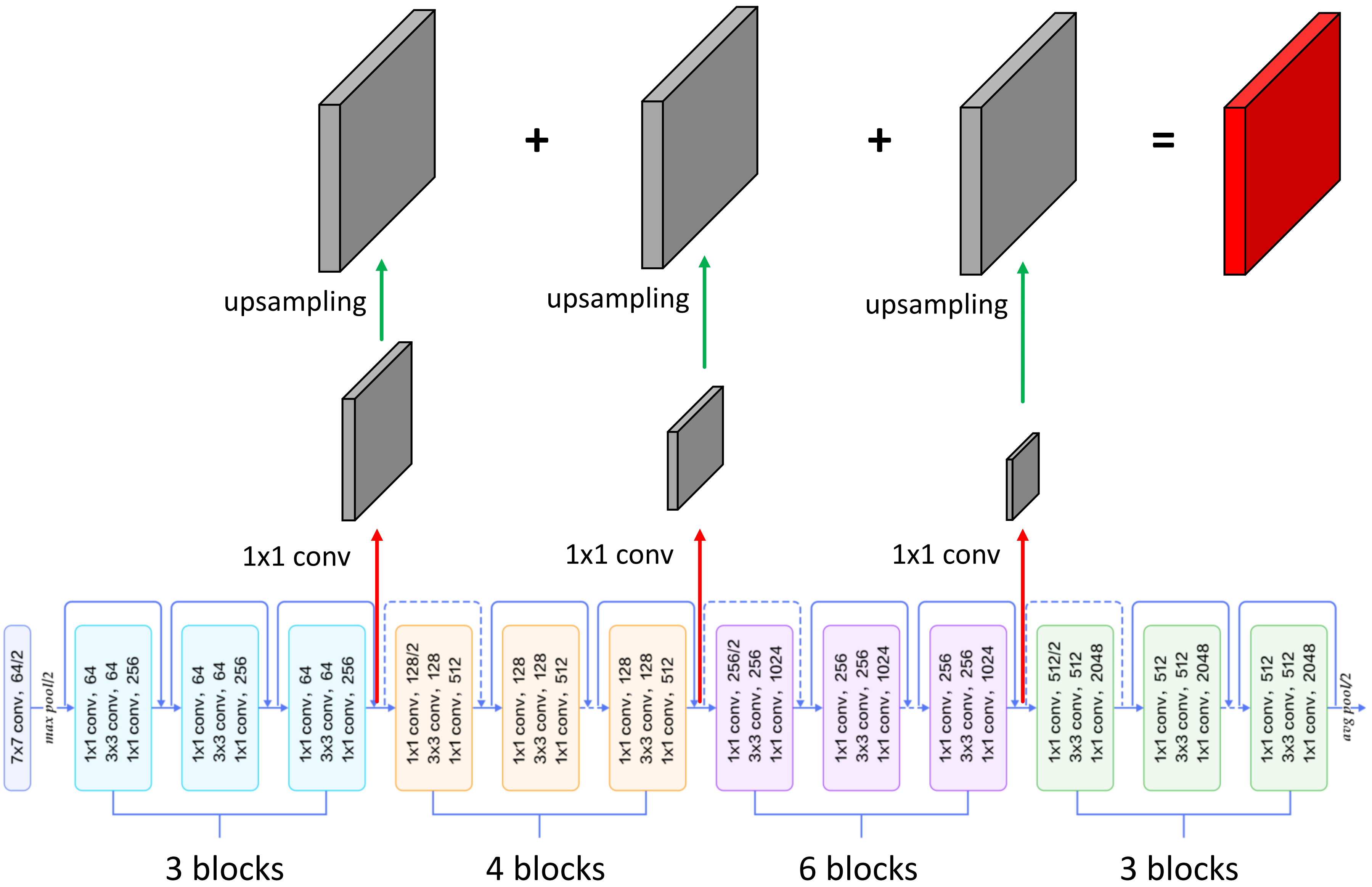
- *~/semseg/resnet50-fcn/resnet50_edit.py*

```
x = ZeroPadding2D((3, 3))(img_input)
x = Conv2D(64, (7, 7), strides=(2, 2), name='conv1')(x)
x = BatchNormalization(axis=bn_axis, name='bn_conv1')(x)
x = Activation('relu')(x)
x = MaxPooling2D((3, 3), strides=(2, 2))(x)

x = conv_block(x, 3, [64, 64, 256], stage=2, block='a', strides=(1, 1))
x = identity_block(x, 3, [64, 64, 256], stage=2, block='b')
x = identity_block(x, 3, [64, 64, 256], stage=2, block='c')

x = conv_block(x, 3, [128, 128, 512], stage=3, block='a')
x = identity_block(x, 3, [128, 128, 512], stage=3, block='b')
x = identity_block(x, 3, [128, 128, 512], stage=3, block='c')
x = identity_block(x, 3, [128, 128, 512], stage=3, block='d')

x = conv_block(x, 3, [256, 256, 1024], stage=4, block='a')
x = identity_block(x, 3, [256, 256, 1024], stage=4, block='b')
x = identity_block(x, 3, [256, 256, 1024], stage=4, block='c')
x = identity_block(x, 3, [256, 256, 1024], stage=4, block='d')
x = identity_block(x, 3, [256, 256, 1024], stage=4, block='e')
x = identity_block(x, 3, [256, 256, 1024], stage=4, block='f')

x = conv_block(x, 3, [512, 512, 2048], stage=5, block='a')
x = identity_block(x, 3, [512, 512, 2048], stage=5, block='b')
x = identity_block(x, 3, [512, 512, 2048], stage=5, block='c')

x = AveragePooling2D((7, 7), name='avg_pool')(x)
```



[7] Semantic Segmentation of Aerial Imagery

The ResNet is Adapted Into an FCN

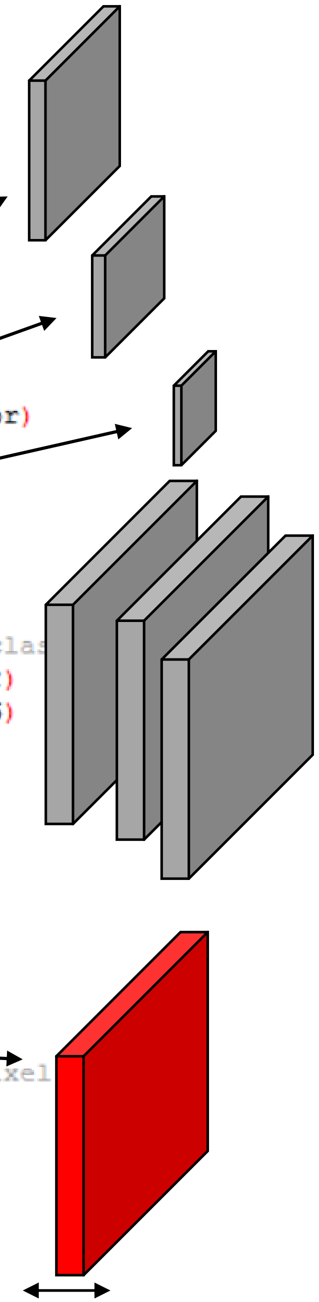


[7] Semantic Segmentation of Aerial Imagery

Residual Network 50 FCN

- Python fu*~/semseg/resnet50-fcn/ model_generator.py*

```python
# the function to generate a FCN version of the ResNet50 model:
def generate_resnet50_fcn(use_pretraining):
    num_labels = 6
    input_dim_row = 256
    input_dim_col = 256
    input_shape = (input_dim_row, input_dim_col, 3)
    input_tensor = Input(shape=input_shape)
    weights = 'imagenet' if use_pretraining else None
    standard_model = ResNet50(include_top=False, weights=weights, input_tensor=input_tensor)

    # get the activations after different network parts by name:
    x32 = standard_model.get_layer('act3d').output
    x16 = standard_model.get_layer('act4f').output
    x8 = standard_model.get_layer('act5c').output

    # apply 1x1 convolution to compress the depth of the output tensors to the number of clas
    c32 = Convolution2D(filters=num_labels, kernel_size=(1, 1), name='conv_labels_32')(x32)
    c16 = Convolution2D(filters=num_labels, kernel_size=(1, 1), name='conv_labels_16')(x16)
    c8 = Convolution2D(filters=num_labels, kernel_size=(1, 1), name='conv_labels_8')(x8)

    # resize the spatial dimensions to fit the spatial input size:
    r32 = Lambda(resize_bilinear, name='resize_labels_32')(c32)
    r16 = Lambda(resize_bilinear, name='resize_labels_16')(c16)
    r8 = Lambda(resize_bilinear, name='resize_labels_8')(c8)

    # sum up the activations of different stages to get information of different solution
    m = Add(name='merge_labels')([r32, r16, r8])

    # apply a softmax activation function to get the probability of each class for each pixel
    x = Reshape((input_dim_row * input_dim_col, num_labels))(m)
    x = Activation('softmax')(x)
    x = Reshape((input_dim_row, input_dim_col, num_labels))(x)

    # return the FCN version of the ResNet50 model:
    return Model(inputs=input_tensor, outputs=x)
```



D= number of classes

[7] Semantic Segmentation of Aerial Imagery

Python Function for ResNet50 FCN with Augmentation

- Location **~/semseg/resnet50-fcn/train_resnet50_fcn.py**

```python
def train_with_augmentation(data_path,output_model,transfer_learning_flag):
    num_labels = 6

    print('Load data ... ')
    x_train, y_train = dio.load_data(data_path + 'vaihingen_train.hdf5')
    print('Training samples: {}'.format(x_train.shape))
    x_val, y_val = dio.load_data(data_path + 'vaihingen_val.hdf5')
    print('Validation samples: {}'.format(x_val.shape))
```

Load:
vaihingen_train.hdf5
vaihingen_validation.hdf5

```python
    print('generate augmented images ...')
    x_train_aug, y_train_aug = augmentation.every_element_randomly_once(x_train, y_train)
    x_val_aug, y_val_aug = augmentation.every_element_randomly_once(x_val, y_val)
```

augmentation.py

Apply one random augmentation to each 256x256 patch
Rotate(90,180,270) or
Flip (up,down)

```python
    print(x_train.dtype)
    print(x_train_aug.dtype)

    # put each array together with its augmented version:
    print('a')
    x_train = np.concatenate([x_train, x_train_aug])
    y_train = np.concatenate([y_train, y_train_aug])
    x_val = np.concatenate([x_val, x_val_aug])
    y_val = np.concatenate([y_val, y_val_aug])
```

Concatenate original data with the augmented data

```python
    # shuffle the samples:
    print('b')
    x_train, y_train = augmentation.shuffle_4d_sample_wise(x_train, y_train)
    x_val, y_val = augmentation.shuffle_4d_sample_wise(x_val, y_val)

    print('preprocess the input data (normalization, centering) ... ')
    x_train = preprocess_input(x_train, mode='tf')
    x_val = preprocess_input(x_val, mode='tf')

    print('preprocess the labels ... ')
    y_train -= 1
    y_val -= 1
    y_train = to_categorical(y_train, num_labels)
    y_val = to_categorical(y_val, num_labels)
```

Preprocessing

```python
    print('load model ... ')
    resnet50_fcn_model = model_generator.generate_resnet50_fcn(use_pretraining=transfer_learning_flag)
    resnet50_fcn_model.summary()
```

Generate the model

```python
    print('compile model ... ')
    resnet50_fcn_model.compile(optimizer=keras.optimizers.Adam(),
                               loss=keras.losses.categorical_crossentropy,
                               metrics=['accuracy'])

    resnet50_fcn_model.fit(x_train, y_train, batch_size=8, epochs=20, validation_data=(x_val, y_val))

    resnet50_fcn_model.save_weights(output_model)
```

Train the model

Python Function for ResNet50 FCN with Augmentation

- *~/semseg/*resnet50-fcn/train_resnet50_fcn.py
  - ***train_resnet50_fcn.py*** requires 4 parameters

```python
def main(arguments):

    data_path = arguments[1]
    output_model = arguments[2]
    augmentation_flag = arguments[3]
    transfer_learning_flag = arguments[4]

    if augmentation_flag=='True':
        train_with_augmentation(data_path,output_model,transfer_learning_flag)
    elif augmentation_flag=='False':
        train(data_path,output_model,transfer_learning_flag)
    else:
        sys.exit()
    end

if __name__ == '__main__':

    if len(sys.argv)<4:
      print ''
      print '***********************************'
      print 'Four paremeters need to be specified:'
      print '1. Location of vaihingen_train.hdf5 and vaihingen_val.hdf5 (e.g., /homea/hpclab/train002/semseg/data/ )'
      print '2. Location + name of the output model (e.g., /homea/hpclab/train002/semseg/models/resnet50_fcn_weights.hdf5)'
      print '3. Augmentation: True or False'
      print '4. Transfer learning (load weights trained on ImageNet): True or False'
      print '***********************************'

      sys.exit()

    main(argv)
```

Edit the Script and Submit the First Training Job

1. **Get a copy of the script /homea/hpclab/train001/scripts/submit_train_resnet50_fcn.sh**

2. **Modify the highlighted parts**

```
#!/bin/bash -x
#SBATCH--nodes=1
#SBATCH--ntasks=1
#SBATCH--output=train_resnet50_fcn_out.%j
#SBATCH--error=train_resnet50_fcn_err.%j
#SBATCH--time=01:00:00
#SBATCH--mail-user=g.cavallaro@fz-juelich.de
#SBATCH--mail-type=ALL
#SBATCH--job-name=train_resnet50_fcn

#SBATCH--partition=gpus
#SBATCH --gres=gpu:1

#SBATCH--reservation=deep_learning

### location executable
RESNET50_FCN=/homea/hpclab/train002/semseg/resnet50-fcn/train_resnet50_fcn.py

module restore dl_tutorial

### submit
python $RESNET50_FCN /homea/hpclab/train002/semseg/data/
/homea/hpclab/train002/semseg/models/resnet50_fcn_weights.hdf5 True False
```

Location of training and validation sets

Create a new folder where the trained model will be saved

3. **Submit: $ sbatch submit_train_resnet50_fcn.sh**

Edit the Script and Submit the Second Training Job

1. **Get a copy of the script**
   /homea/hpclab/train001/script/submit_train_resnet50_fcn_pretrained.sh
2. **Modify the highlighted parts**

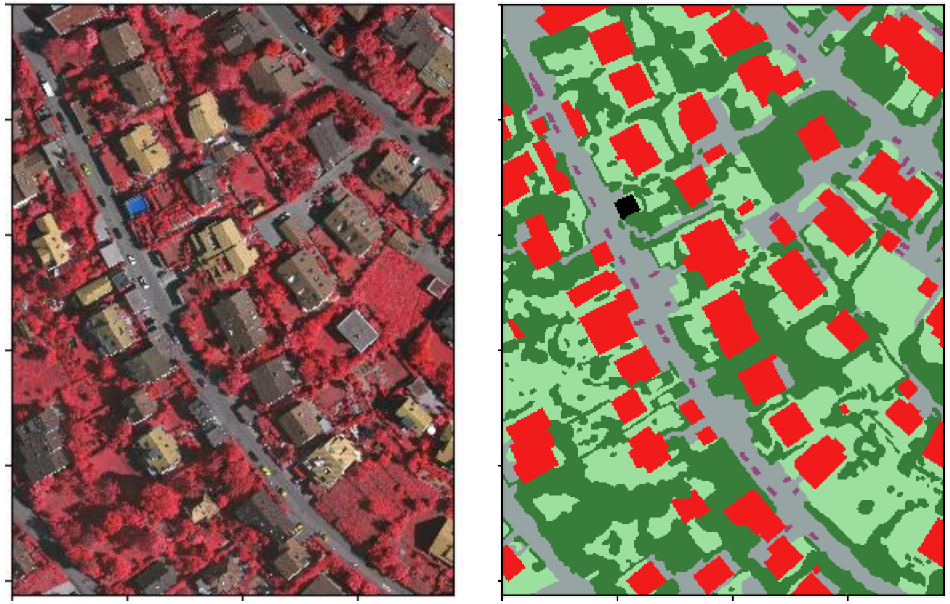
```
#!/bin/bash -x
#SBATCH--nodes=1
#SBATCH--ntasks=1
#SBATCH--output=train_resnet50_fcn_pretrained_out.%j
#SBATCH--error=train_resnet50_fcn_pretrained_err.%j
#SBATCH--time=02:00:00
#SBATCH--mail-user=g.cavallaro@fz-juelich.de
#SBATCH--mail-type=ALL
#SBATCH--job-name=train_resnet50_fcn_pretrained

#SBATCH--partition=gpus
#SBATCH--gres=gpu:1

#SBATCH--reservation=deep_learning

### location executable
RESNET50_FCN=/homea/hpclab/train002/semseg/resnet50-fcn/train_resnet50_fcn.py

module restore dl_tutorial

### submit
python $RESNET50_FCN /homea/hpclab/train002/semseg/data/
/homea/hpclab/train002/semseg/models/resnet50_fcn_weights_pretrained.hdf5 True True
```

Location of training and validation sets

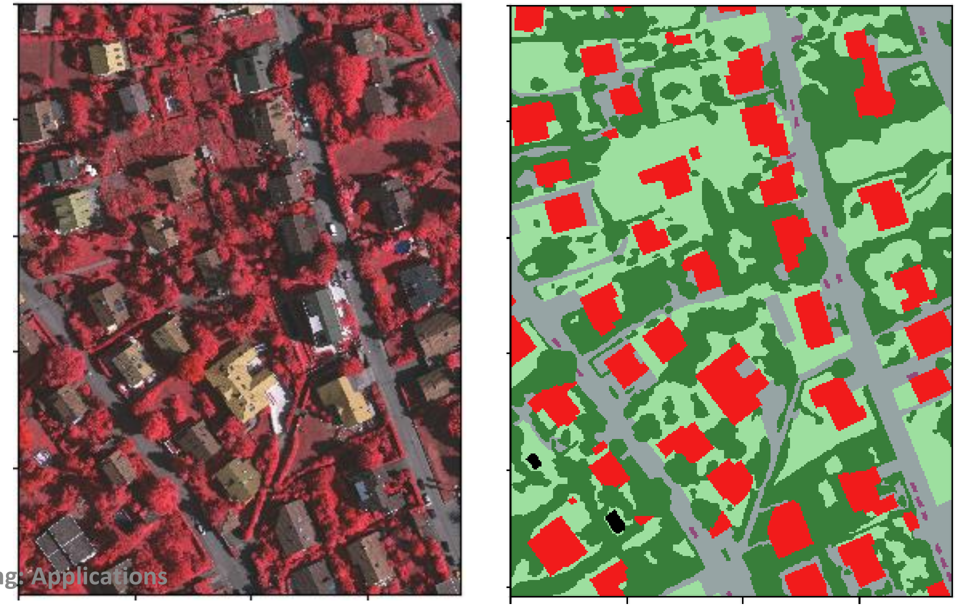
Create a new folder where the trained model will be saved

Test Set

Vaihingen_15



Vaihingen_23



Thematic classes:

- ***Impervious surfaces***
- ***Building***
- ***Low vegetation***
- ***Tree***
- ***Car***
- ***Clutter/background***

Test ResNet50 FCN

- Use the function ***~/semseg/*resnet50-fcn/evaluate_network.py**

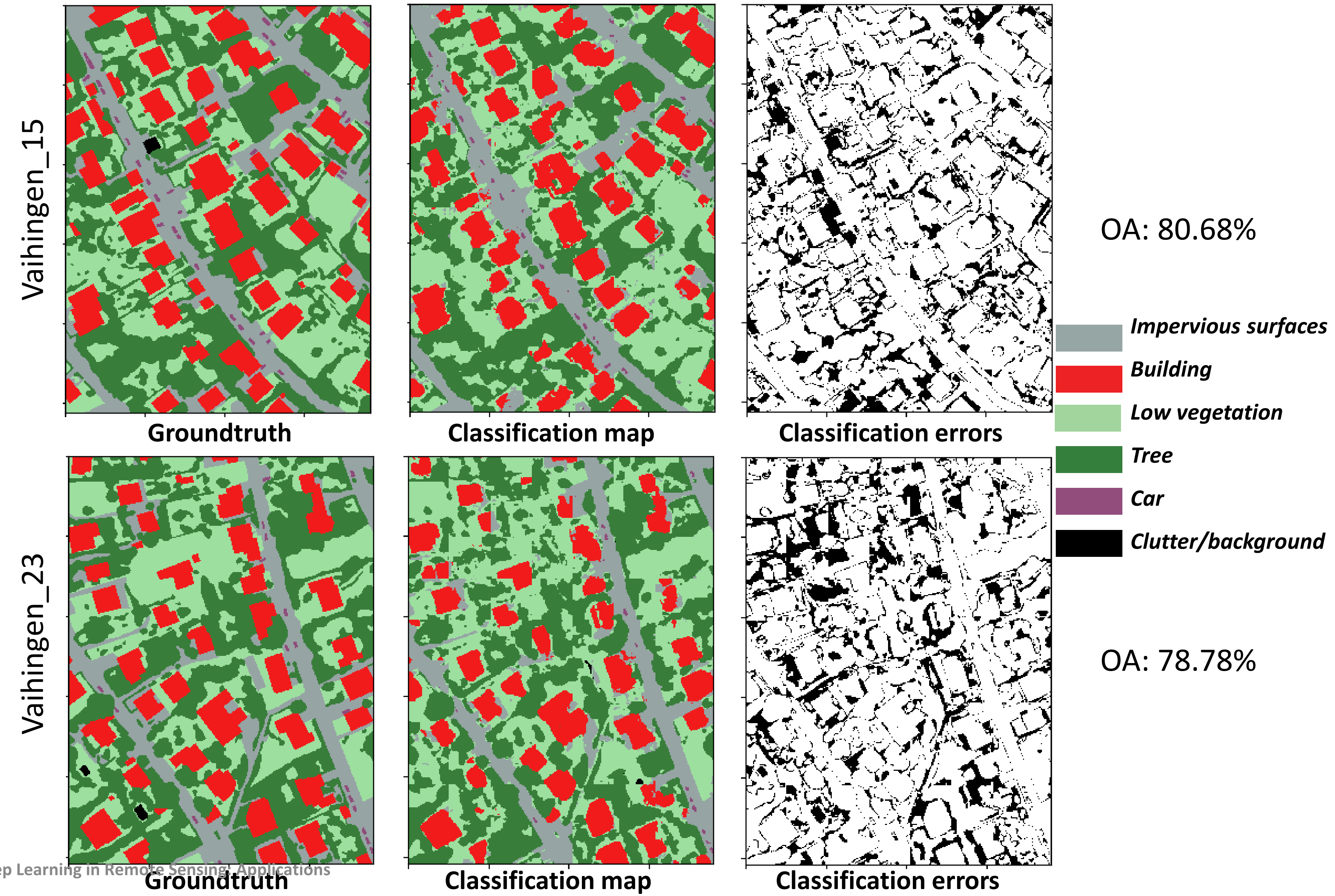
- **Run the test on the login node (i.e., no batch script submission)**

- **Run the test on the Vaihingen 15:**
  $ python evaluate_network.py 15 ~/semseg/models/resnet50_fcn_weights.hdf5

Or

- **Run the test on the Vaihingen 23:**
  $ python evaluate_network.py 23 ~/semseg/models/resnet50_fcn_weights.hdf5

Experiment 1: Test ResNet50 FCN



Experiment 1: Test ResNet50 FCN

Vaihingen_15

Class	**Pixels**	**Accuracy**
Impervious surfaces	855112	79.3%
Building	1075170	84.8%
Low vegetation	1309897	80.1%
Tree	1643189	80.3%
Car	31684	39.9%
Clutter/background	7183	0.0%

Confusion matrix

PREDICTED

ACTUAL

678470	27787	111927	35596	1332	0
77693	911674	73864	11848	91	0
50176	35116	1049665	174888	44	8
15034	4887	304333	1318935	0	0
18528	13	181	305	12657	0
226	581	6103	13	260	0

Vaihingen_23

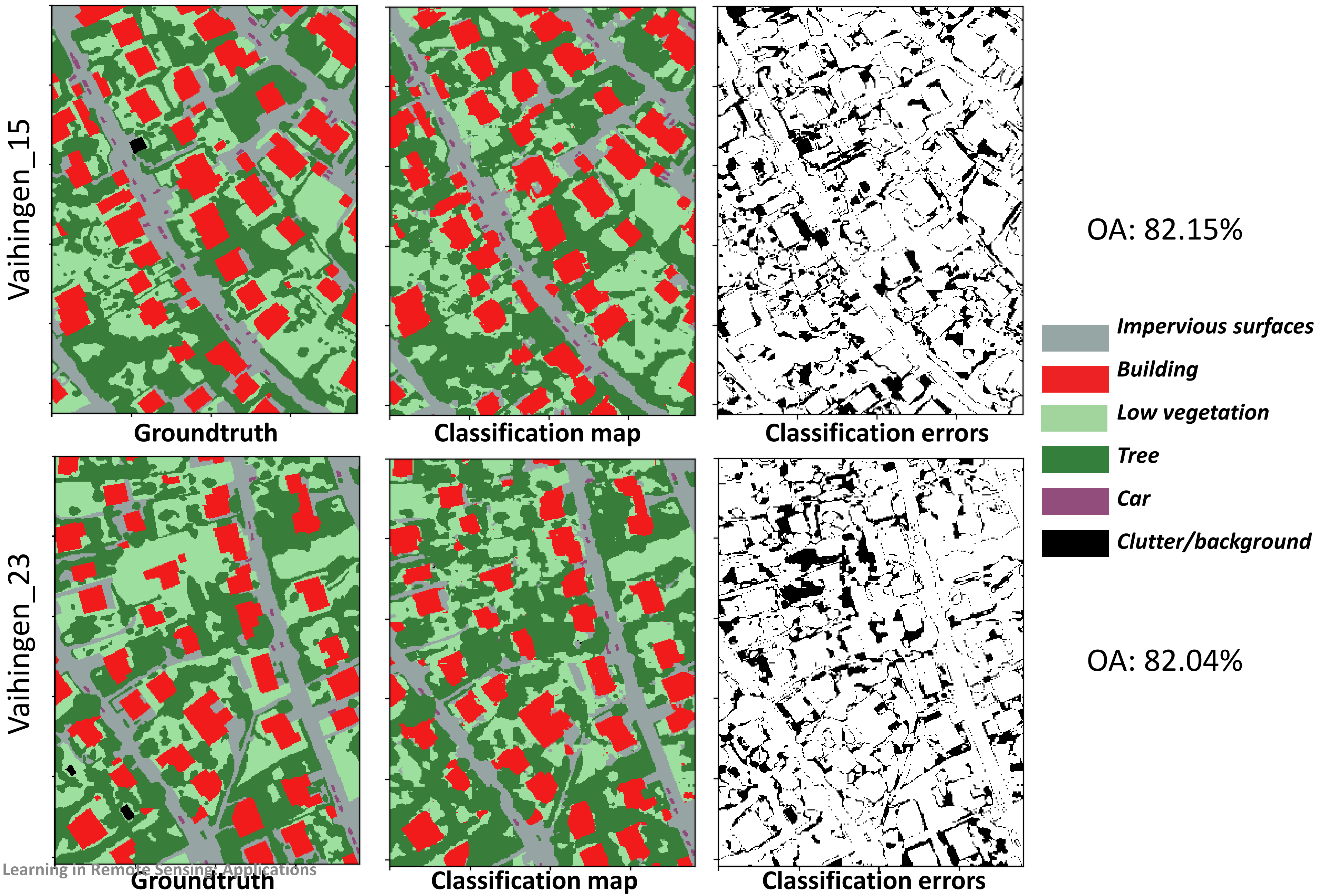
Class	**Pixels**	**Accuracy**
Impervious surfaces	801652	72.3%
Building	885284	82.8%
Low vegetation	1345728	79.3%
Tree	1789171	79.9%
Car	15447	42.3%
Clutter/background	7756	0.0%

PREDICTED

ACTUAL

579374	27187	153102	40069	581	1339
62403	733366	78575	9724	22	1194
43527	23396	1067768	210626	411	0
14370	5981	338902	1429851	0	67
8464	92	189	164	6538	0
0	1614	5929	213	0	0

Experiment 2: Test Pre-Trained ResNet50 with ImageNet



Experiment 2: Test Pre-Trained ResNet50 with ImageNet

Vaihingen_15

Class	Pixels	Accuracy
Impervious surfaces	855112	75.6%
Building	1075170	93.6%
Low vegetation	1309897	72.2%
Tree	1643189	86.4%
Car	31684	81.9%
Clutter/background	7183	0.0%

Confusion matrix

PREDICTED

ACTUAL

646590	40574	102219	60719	5003	7
12621	1006203	39785	16494	67	0
50571	49736	945549	263760	281	0
17049	7515	199078	1419534	8	5
5685	24	6	6	25963	0
634	1	6337	0	211	0

Vaihingen_23

Class	Pixels	Accuracy
Impervious surfaces	801652	76.2%
Building	885284	94.6%
Low vegetation	1345728	66.6%
Tree	1789171	90.4%
Car	15447	84.9%
Clutter/background	7756	0.0%

PREDICTED

ACTUAL

611119	34778	93020	60029	2482	224
8924	837288	21510	17213	279	70
54358	39947	896551	354659	197	16
17677	8520	145840	1616651	361	122
2270	7	0	55	13115	0
112	239	7290	115	0	0