Satellite Computer Vision mit Keras und TensorFlow

Best Practices und Beispiele aus der Forschung

Dr.-Ing. Jens Leitloff
Felix M. Riese
Mannheim, May 2019

Minds Mastering Machines

Karlsruhe Institute of Technology

IPF Institute of Photogrammetry and Remote Sensing

Outline

Why

is satellite computer vision relevant?

How

does computer vision help remote sensing?

What

do you need to start?

Why

is satellite computer vision relevant?

→ Theory

~ 10 minutes

How

does computer vision help remote sensing?

→ Theory

→ Code examples in slides

→ Research results

~ 25 minutes

What

do you need to start?

→ Best practices

→ Links to resources

~ 5 minutes

GitHub

jensleitloff/CNN-Sentinel

Why

is satellite computer vision relevant?

How

does computer vision help remote sensing?

What

do you need to start?

1. Why is this talk relevant (for me)?
2. What is satellite CV?
3. How is it applied in today's life?

Why is this talk relevant (for me)?

Engineers & scientists

State-of-the-art methods

Smart data interpretation

Publishing methods & code

Similar challenges

Multi-modal data fusion

Weak & sparse labels

Curse of dimensionality

Relevant data

2D, 3D, time series, GIS

A lot of free datasets

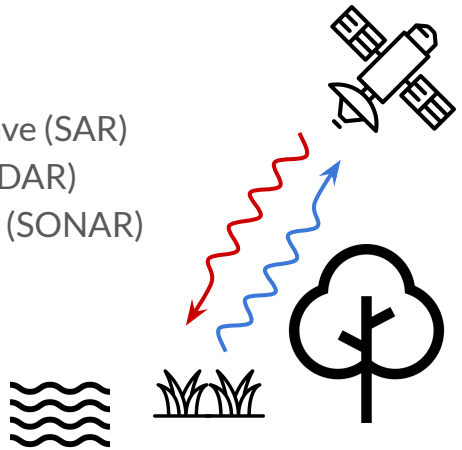
Big data & small data

Icons made by Freepik from www.flaticon.com is licensed by CC 3.0 BY

Remote sensing & satellites

Active

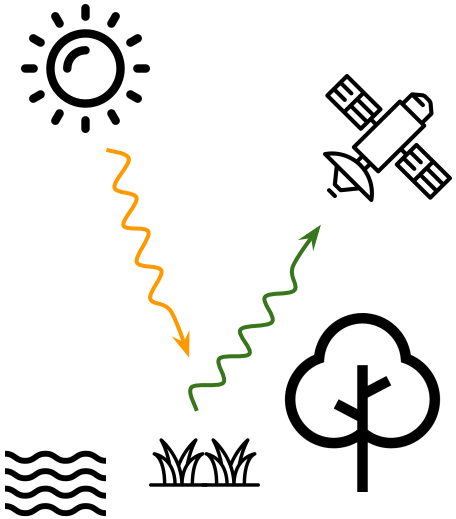
- Microwave (SAR)
- Laser (LIDAR)
- Acoustic (SONAR)



Passive

- RGB
- Multispectral

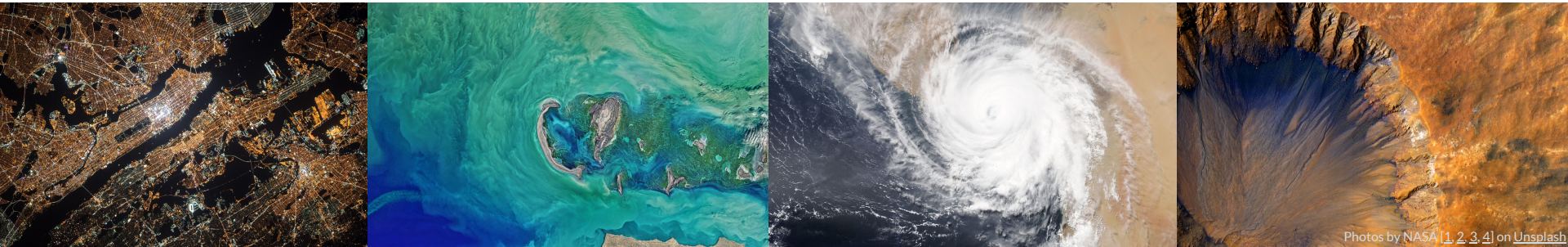
→ Focus of this talk



Icons made by Freepik from www.flaticon.com is licensed by CC 3.0 BY

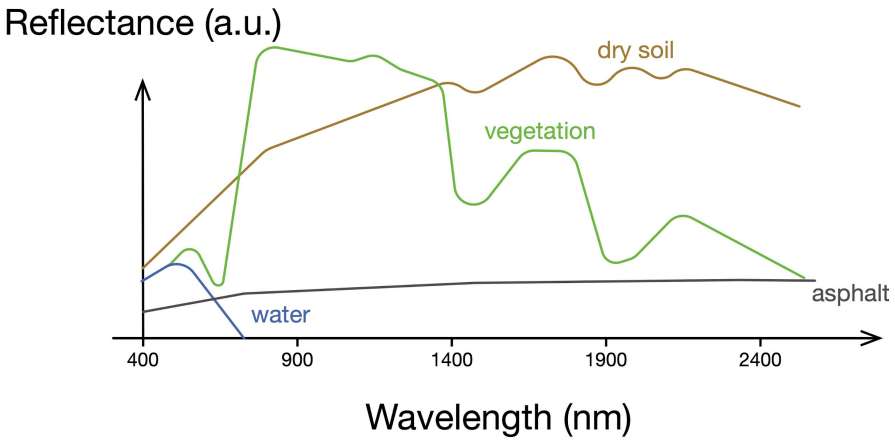
Applications of satellite remote sensing

- **Meteorology** → weather forecasts, climate research, monitoring of glaciers and sea ice
- **Natural resources** → agriculture, forestry, environmental monitoring, coastal monitoring
- **Geosciences** → mapping, road networks, hydrology
- Land **administration** → land use / cover, building detection
- **Disaster** response → monitoring of natural hazards like wildfires, floods, earthquakes, volcanoes



Photos by NASA [1, 2, 3, 4] on Unsplash

Example:
Land cover



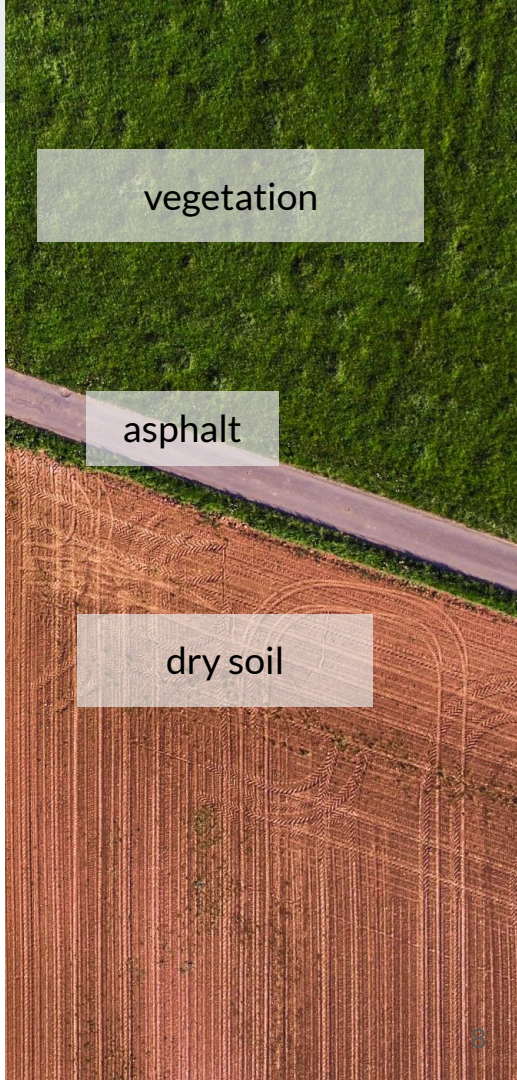
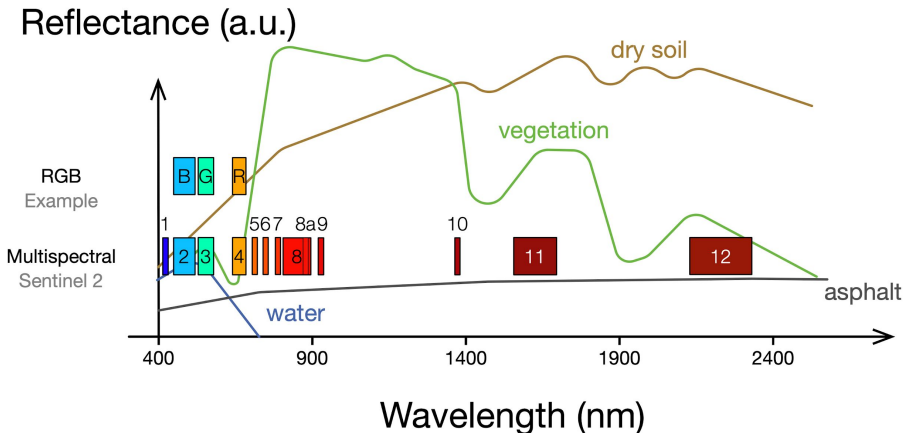


Photo by jean wimmerlin on Unsplash

Example:
Land cover & passive satellite



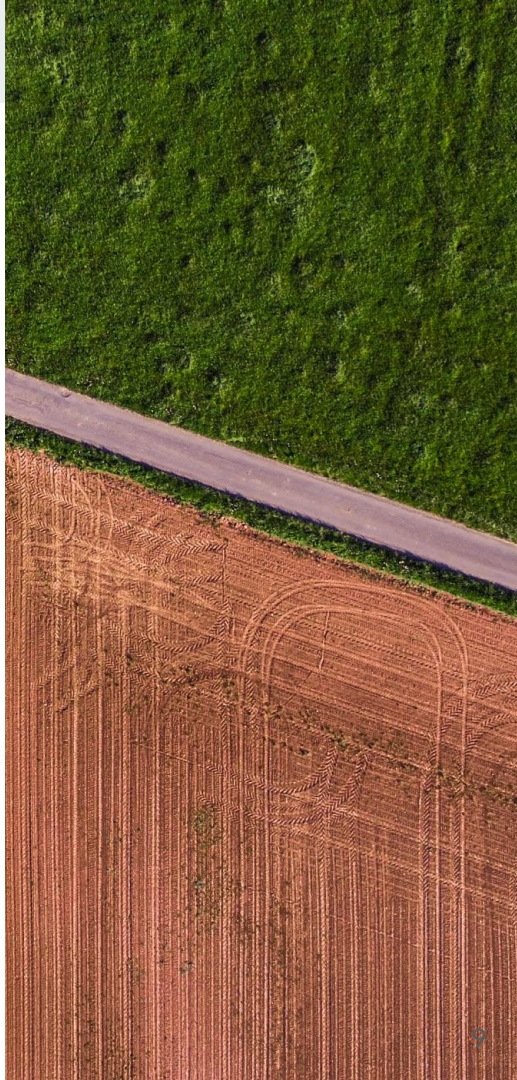


Photo by jean wimmerlin on Unsplash

The Copernicus programme (by ESA)

Launch date	Mission	Objective
2014	Sentinel-1	Radar for land and ocean
2015	Sentinel-2	Vegetation and soil monitoring
2016	Sentinel-3	Marine observation
→ 2023	Sentinel-4	Air quality monitoring
→ 2021	Sentinel-5	Air quality monitoring
→ 2020	Sentinel-6	Atmospheric measurements

5 days / world

It takes the Sentinel-2 satellites five days to cover the earth's surface.

3.2 TB / day

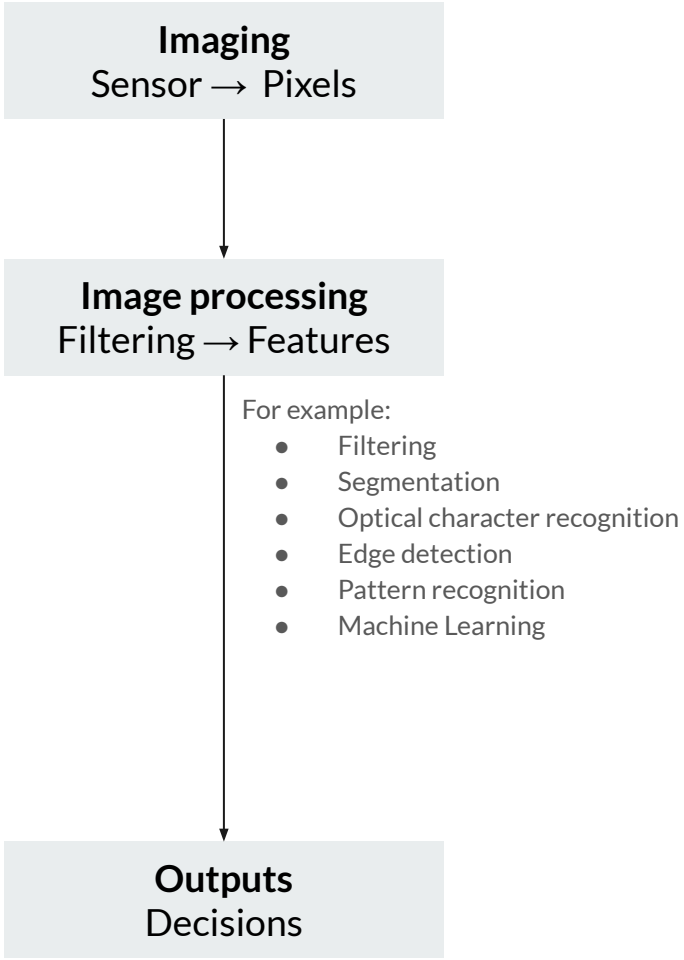
The Sentinel-2 satellites alone produce images in the size of 3.2 Terabytes per day, 365 days a year.

0€

The data of the Sentinel program is free for everyone.
No matter if commercial or not.
No matter if European or not.

Computer Vision

To bridge the gap between pixels and “meaning”.



Computer Vision in Satellite Remote Sensing

To bridge the gap between pixels and “meaning”.

Imaging
Sensor → Pixels

- Active: SAR, LIDAR, SONAR
- Passive: RGB, multispectral

Image processing
Filtering → Features

For example:

- Segmentation of high-resolution satellite images
- Edge detection of crop fields
- Machine Learning for the classification and regression of environmental variables
- Multimodal data fusion
- Interpretation hyperspectral image analysis (classification, regression)
- Interpretation of SAR images

Outputs
Decisions

Why

is satellite computer vision relevant?

How

does computer vision help remote sensing?

1. Learning 2D patterns with CNN
2. Learning 1D & 3D patterns
3. Learning sequences

What

do you need to start?

Learning 2D patterns

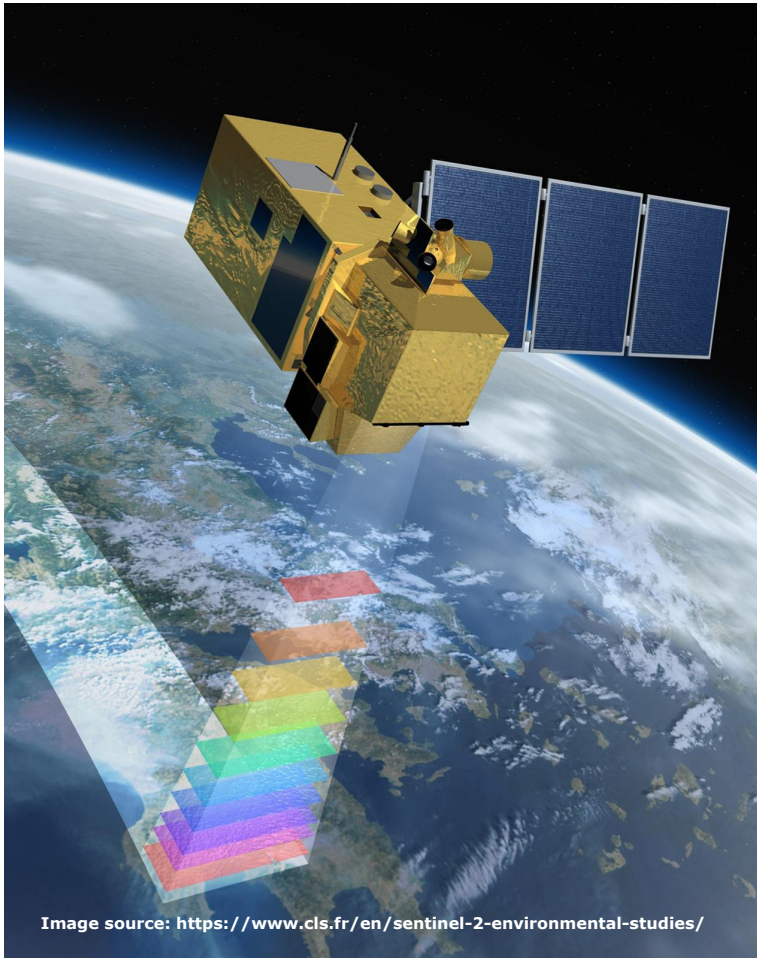
The EuroSAT dataset

Links:

- Introduced in arXiv:1709.00029
- Download here: http://madm.dfki.de/downloads

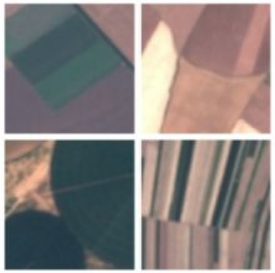
Properties:

- Sentinel-2 satellite images
  - **Raw dataset:** 13 different spectral bands (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 8a)
  - **RGB dataset:** RGB colors
- 27 000 labeled images
- 10 classes (→ next slide)

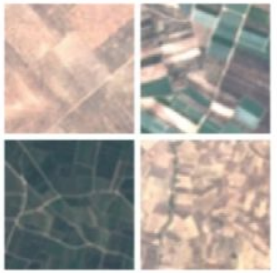


(a) Industrial

(b) Residential

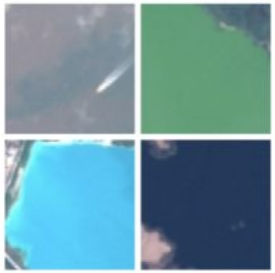


(c) Annual crop



(d) Permanent crop

(e) River



(f) Sea & lake

(g) Herbaceous vegetation

(h) Highway

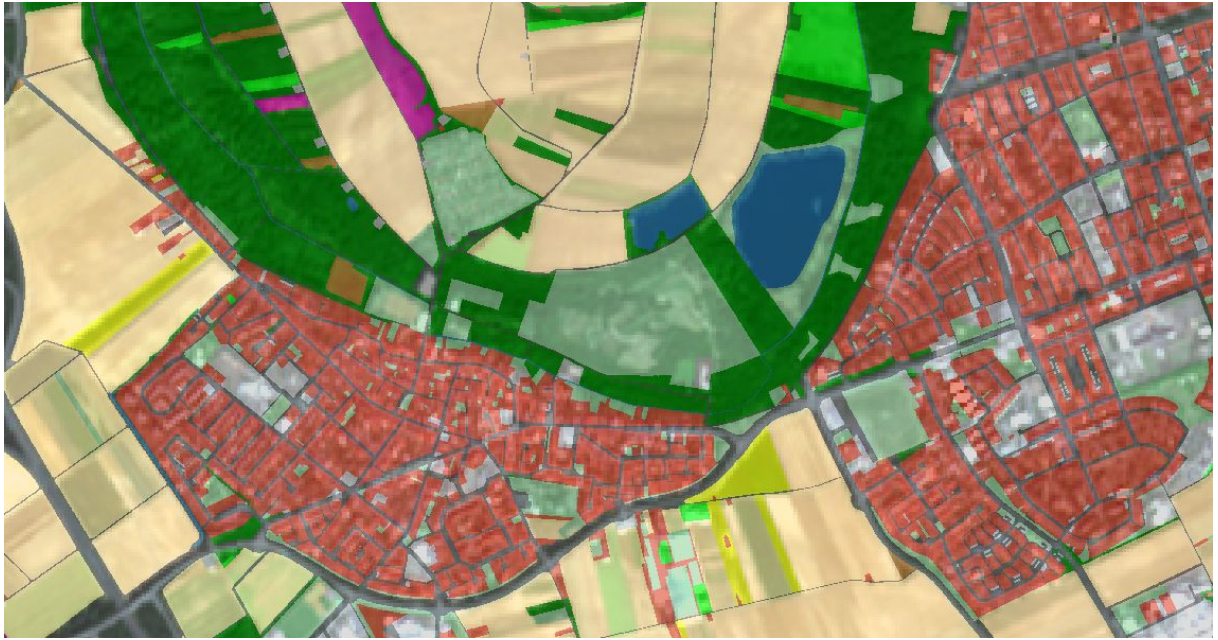
(i) Pasture

(j) Forest

one class per **image**

Image source: arXiv:1709.00029

Segmentation training



one class per **pixel**

Learning from scratch

1. Network structure with **randomly initialized** weights
2. Training of **complete** network on training dataset

- **slow** convergence
- **lots of** training data necessary
- easily implemented for **every kind** of input data

Transfer learning

Learning from scratch

1. Network structure with **randomly initialized** weights
2. Training of **complete** network on training dataset

- **slow** convergence
- **lots of** training data necessary
- easily implemented for **every kind** of input data

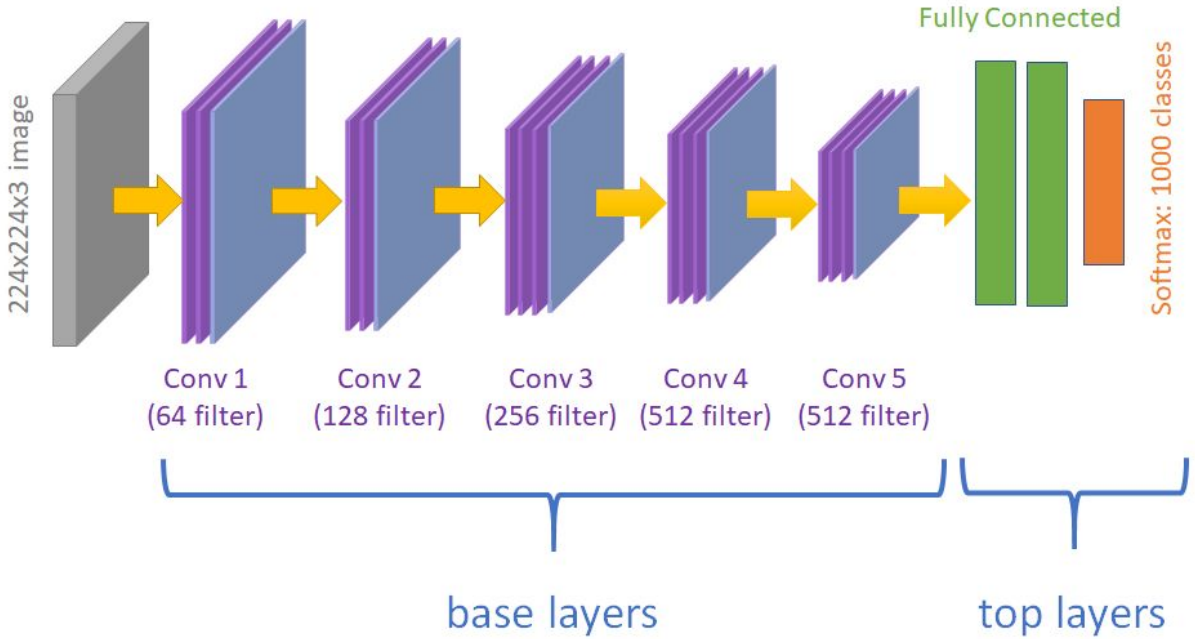
Transfer learning

1. Network structure with **pretrained** weights
2. Training of **selected** layers on the training dataset (= fine tuning)

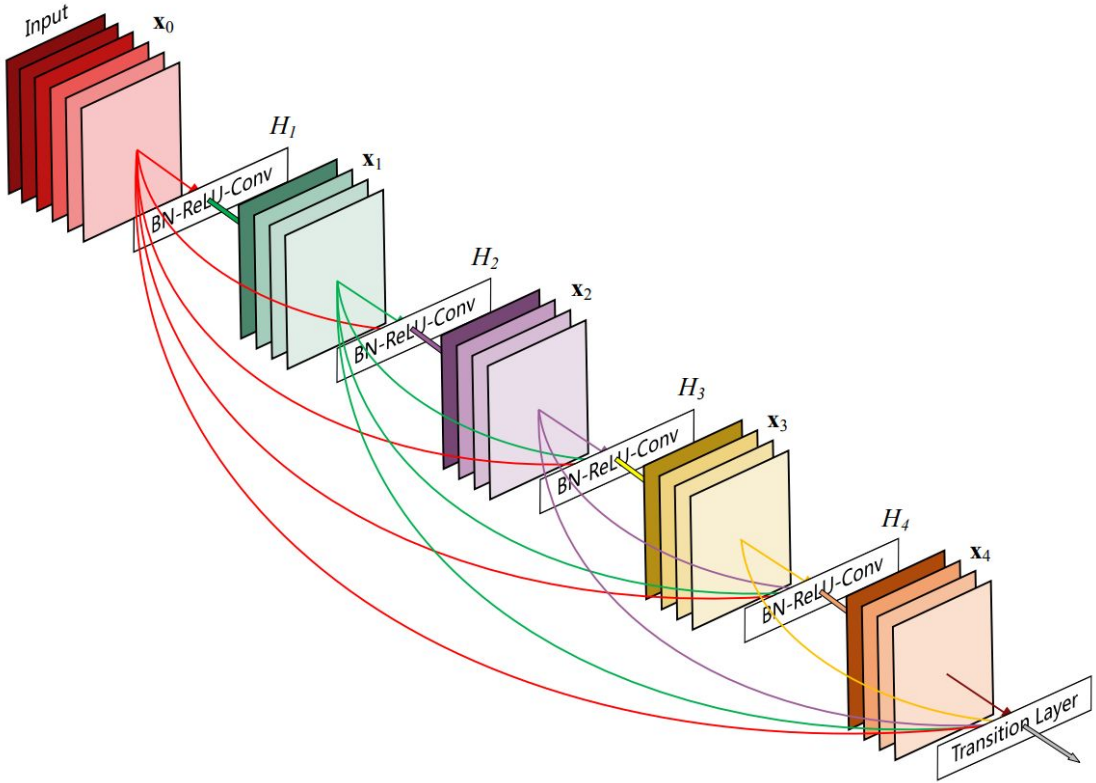
- much **faster** convergence
- **less** training data necessary
- mostly implemented **for RGB**
- modifications for MS necessary

→ Implement for RGB & multispectral

VGG16 architecture



DenseNet architecture



Densely Connected Convolutional Networks, arXiv:1608.06993

Steps to do

Training from scratch

1. Initialize network model without top layers
2. define new top layers
3. define data augmentation
4. define callbacks
5. -
6. -
7. -
8. fit model

Transfer-learning

1. Pretrained network model without top layers
2. define new top layers
3. define data augmentation
4. define callbacks
5. set base layers non trainable
6. fit model (train new top layers)
7. set (some) base layers trainable
8. fit model (fine-tune base and top layers)

Learning from scratch

```
base_model = VGG(include_top=False,
                 weights=None,
                 input_shape=(64, 64, 13))
```

Transfer learning

```
base_model = VGG(include_top=False,
                 weights='imagenet',
                 input_shape=(64, 64, 3))
```

```
# add a global spatial average pooling layer
top_model = base_model.output
top_model = GlobalAveragePooling2D()(top_model)
# let's add a fully-connected layer
top_model = Dense(2048, activation='relu')(top_model)
top_model = Dense(2048, activation='relu')(top_model)
# and a logistic layer
predictions = Dense(num_classes, activation='softmax')(top_model)

# this is the model we will train
model = Model(inputs=base_model.input, outputs=predictions)
```

```
# first: train only the top layers
# i.e. freeze all convolutional layers
for layer in base_model.layers:
    layer.trainable = False
```

Nice tools in Keras I

- Image augmentation

```
# defining ImageDataGenerators
# ... initialization for training
train_datagen = ImageDataGenerator(fill_mode="reflect",
                                   rotation_range=45,
                                   horizontal_flip=True,
                                   vertical_flip=True)
```

- Use image patches in directories

```
train_datagen.flow_from_directory(path_to_train,
                                  target_size=(64, 64),
                                  batch_size=batch_size,
                                  class_mode='categorical')
```

Nice tools in Keras II

- Define some callbacks for training

```
checkpointer = ModelCheckpoint(...)

earlystopper = EarlyStopping(...)

tensorboard = TensorBoard(...)
```

- and train ...

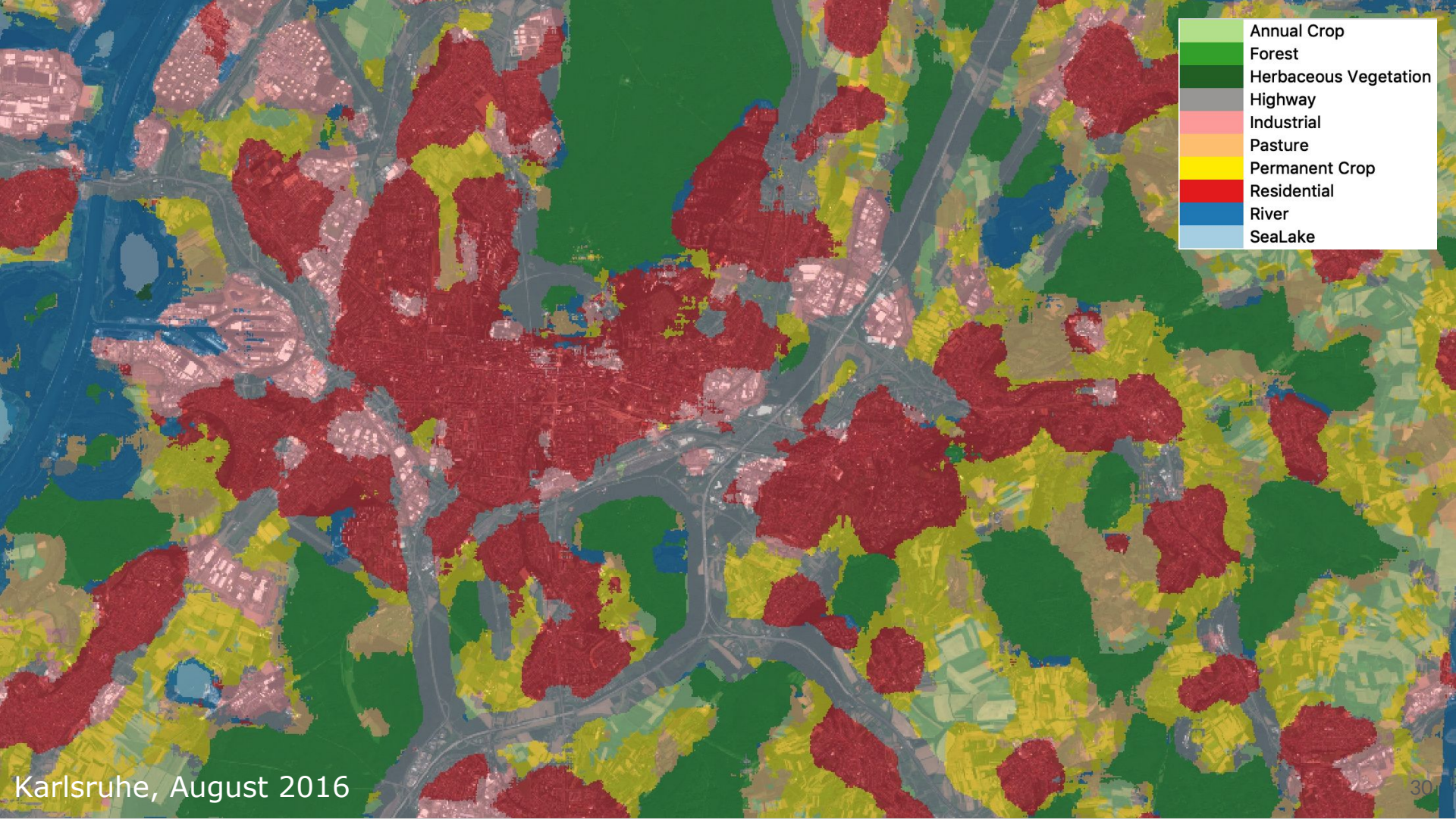
```
model.fit_generator(train_generator,...
                    callbacks=[checkpointer, earlystopper,
                               tensorboard],
                    validation_data=validation_generator)
```

Additional things in the repository

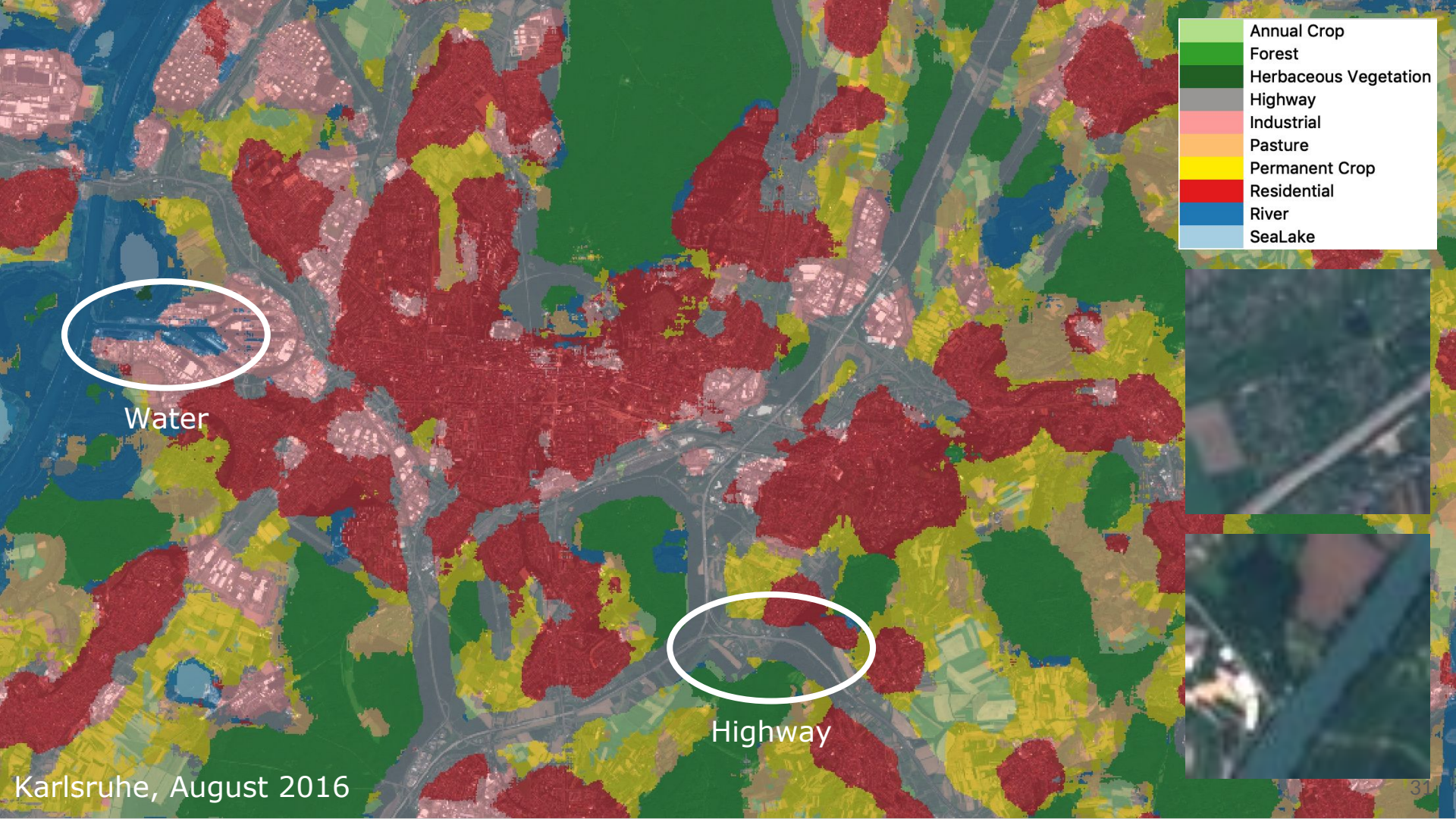
- in detail notebooks for
  - learning from scratch
  - transfer learning
  - customized data generators
- python scripts for
  - panchromatic data
  - multispectral data
- many additional information about
  - setup an environment
  - data
  - ...

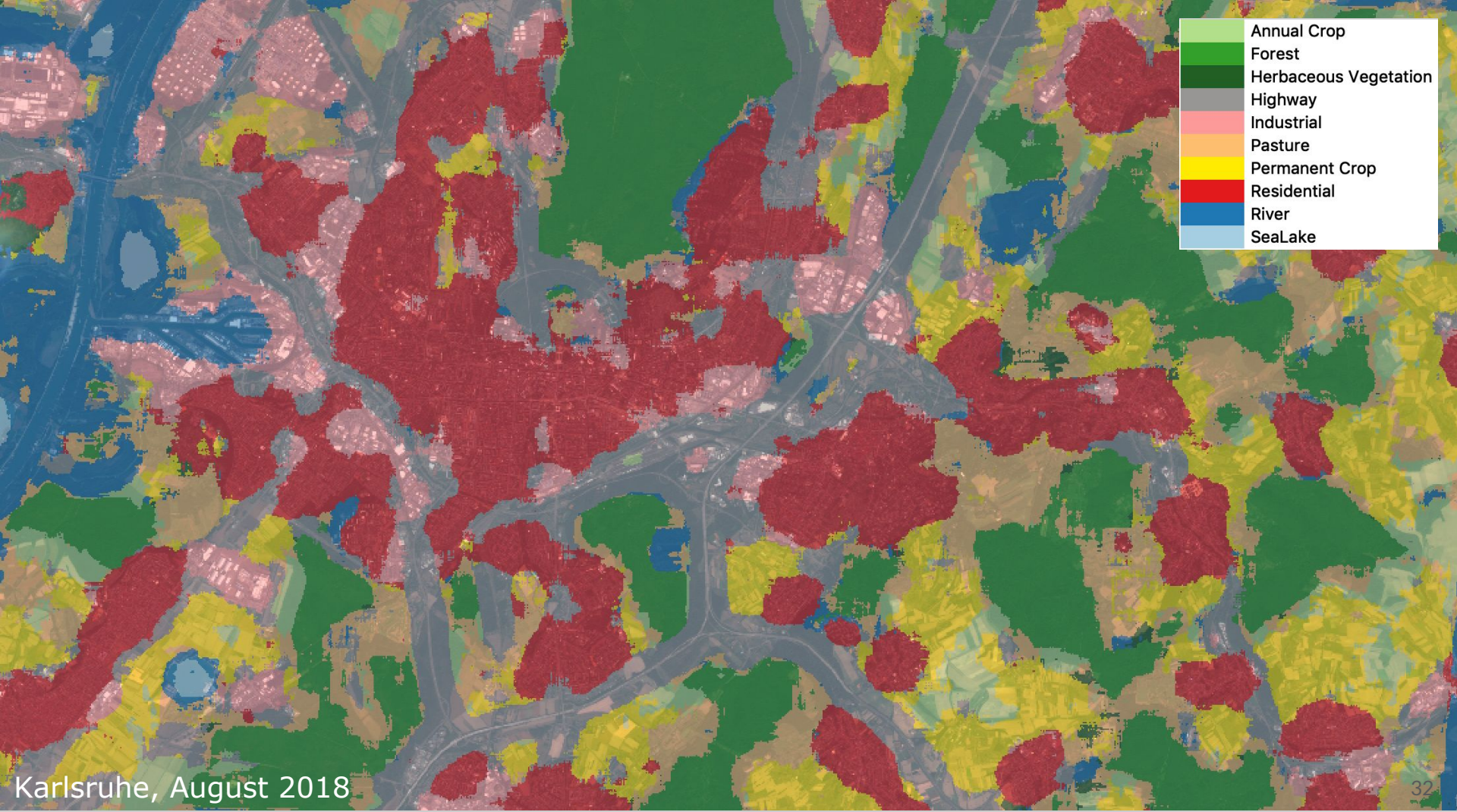
GitHub

jensleitloff/CNN-Sentinel



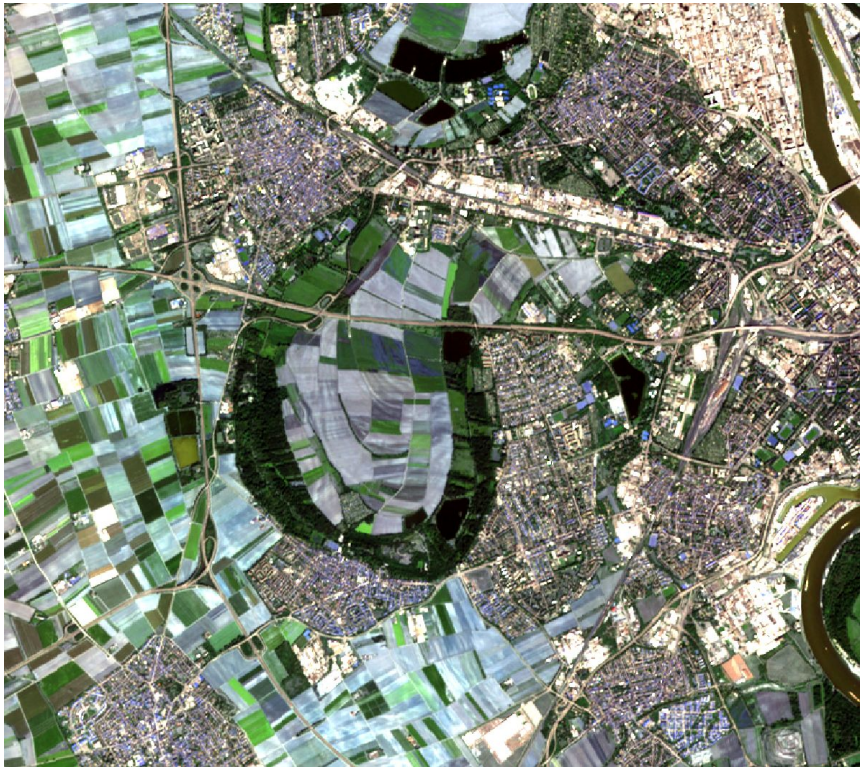
Karlsruhe, August 2016


Annual Crop
Forest
Herbaceous Vegetation
Highway
Industrial
Pasture
Permanent Crop
Residential
River
SeaLake
Water
Highway
Karlsruhe, August 2016

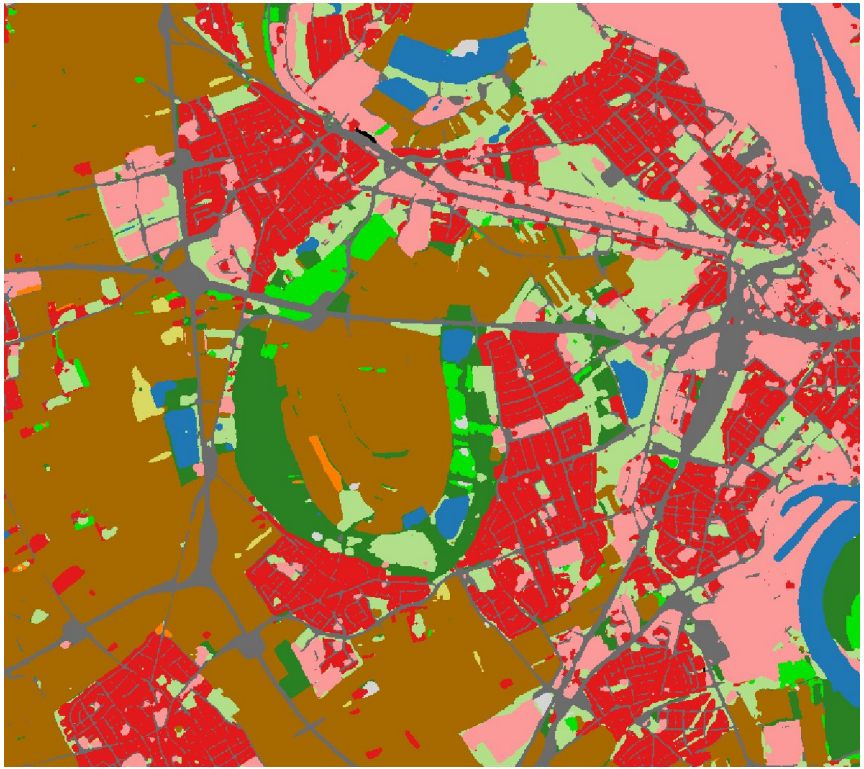


Karlsruhe, August 2018

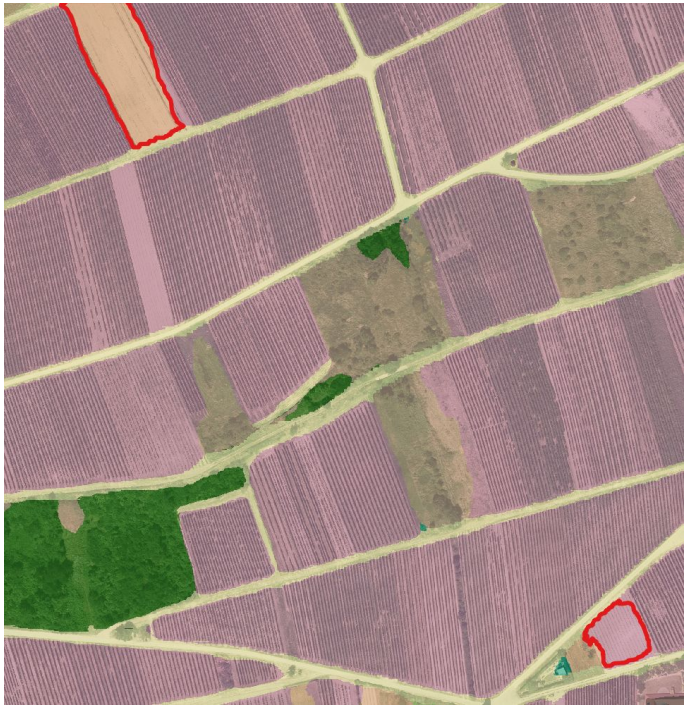
Ludwigshafen, August 2015



Results from project with LVermGeo Rheinland-Pfalz



Results with image segmentation



Klassifikation 20cm

Learning 1D & 3D patterns
Learning sequences

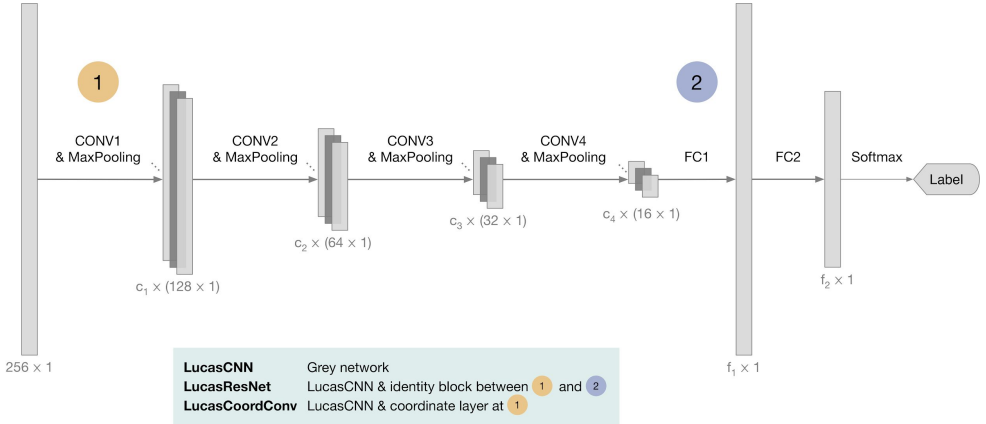
Other convolution approaches

Convolution approaches:

- No convolution
- 1D convolution over spectral dimension
- 2D convolution over spatial dimension → this talk
- 3D convolution over spectral & spatial dimension
- 3D convolution over temporal & spatial dimension

Review paper: arXiv:1710.03959

Learning 1D patterns with CNNs



- Paper will be presented in June 2019
- Code available on GitHub
- Free soil texture dataset
- Hyperspectral → more features than multispectral
- 1D convolution over the spectral dimension

Felix M. Riese and Sina Keller, "Soil Texture Classification with 1D Convolutional Neural Networks based on Hyperspectral Data," 2019, Accepted at Geospatial Week 2019 in Enschede (NL). Available: arXiv:1901.04846

Learning 3D patterns with CNNs

- 3D convolution over **spatial** & **spectral** dimension
  - spatial dimension → earth surface (cf. Learning 2D patterns)
  - spectral dimension → making use of multi-/hyperspectral data (cf. Learning 1D patterns with CNNs)
- 3D convolution over **spatial** & **temporal** dimension
  - temporal dimension → learn from temporal changes, e.g. crop

Learning sequences

- Temporal changes → learning sequences
- Recurrent neural networks, LSTM → Rußwurm, Körner (arXiv:1802.02080)

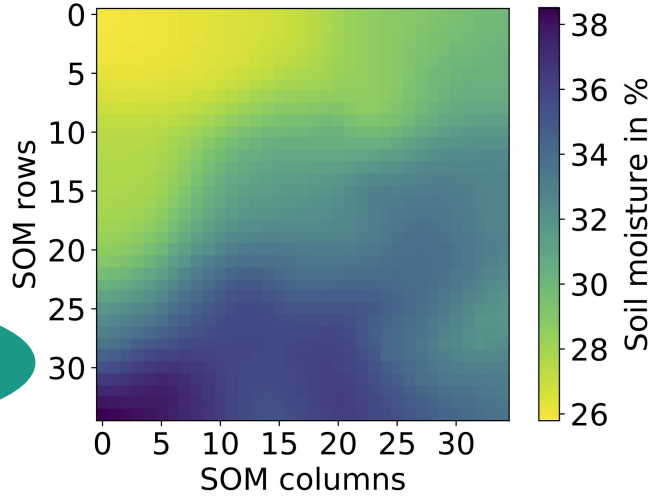
Icons made by Good Ware from www.flaticon.com is licensed by CC 3.0 BY

One more thing: SUSI

SUpervised Self-organIzing maps in Python

- Self-organizing maps for unsupervised **and** supervised learning
- Code and usage in **scikit-learn style**
- Example applications
  - 2D visualization of high-dimensional data
  - Classification and regression on small datasets
- Code and paper freely available

see talk by Oliver Zeigermann today



Source: felixriese/susi

Why

is satellite computer vision relevant?

How

does computer vision help remote sensing?

What

do you need to start?

1. Best practices from research
2. Starting point with satellite CV

Best practices

Basics

- Use existing frameworks like **Keras**
- Make use of **build-in functions** like
  - data augmentation generators
  - callbacks
- Try **established architectures** and try **pre-trained** networks to reduce training time
- **Big data** beats data cleaning
- Measure error on training **and** test subset
- Training **strategy:**
  1. Training of estimator on small dataset until training error is 0
  2. Implementation of regularization, e.g. dropout, batch-normalization, …

Best practices

Further reading

Recommended resources:

- CPVR 2014 Talk by Marc'Aurelio Ranzato
- "Visualizing and Understanding Convolutional Networks" by M. D. Zeiler and R. Fergus, 2013, arXiv:1311.2901
- Coursera Deep Learning Specialization part 3: Structuring Machine Learning Projects by deeplearning.ai

Starting point with satellite CV

GitHub

jensleitloff/CNN-Sentinel

- **Requirements** (Versions …)
- **Environment** setup (Conda …)
- **Code** for
  - setting up the data
  - CNN transfer learning RGB & MS
  - CNN learning-from-scratch RGB & MS
- How to download Sentinel-2 **data**
- **Links** to more datasets

Jens Leitloff
✉ jens.leitloff@kit.edu
jensleitloff/CNN-Sentinel

Felix M. Riese
✉ felix.riese@kit.edu

Thank you for your attention!