BR AI and Automation Lab: Satellite Imagery

Innovation Lab Big Data Science, Institut für Statistik, LMU München

Project Partner: Bayerischer Rundfunk

Supervisers: M.Sc. Christoph Molnar and M.Sc. Florian Pfisterer

Presenters: Alex, Ke, Sanchit and Simon

OVERVIEW

Table of Contents

Getting a Story out of Satellite Data

There were lots of potential topics, but it was hard to find one that fits all the criteria:

- **regional interest**
- **Scalable**
- **political impact**

After much research we agreed on the detection of Urban Heat Islands as basis for a potential story.

What are Urban Heat Islands?

Small areas in cities that have significantly higher temperature spikes than surrounding neighbourhood

Potentially dangerous to population and the environment

Generally it‘s hard to find a broadly accepted definiton, so we had to define it by ourselves! (More on this later.)

Project Scope given by BR

1. Evaluation of Data Scources

Landsat? Sentinel? Both? Paid Data?

Which Thermalbands are included?

Timeframe?

2. Evaluation of Surface Temperature, Vegetation and Soilmoisture for a limited area

Practical use of thermal infrafred? Suitable indices?

Visualization? True/false color images? Time Series?

What Summary Statistics?

3. Scalabilty

How can we make the analysis flexible to scale to an arbitrary area?

4. Develop algorithm(s) for occurence and change detection

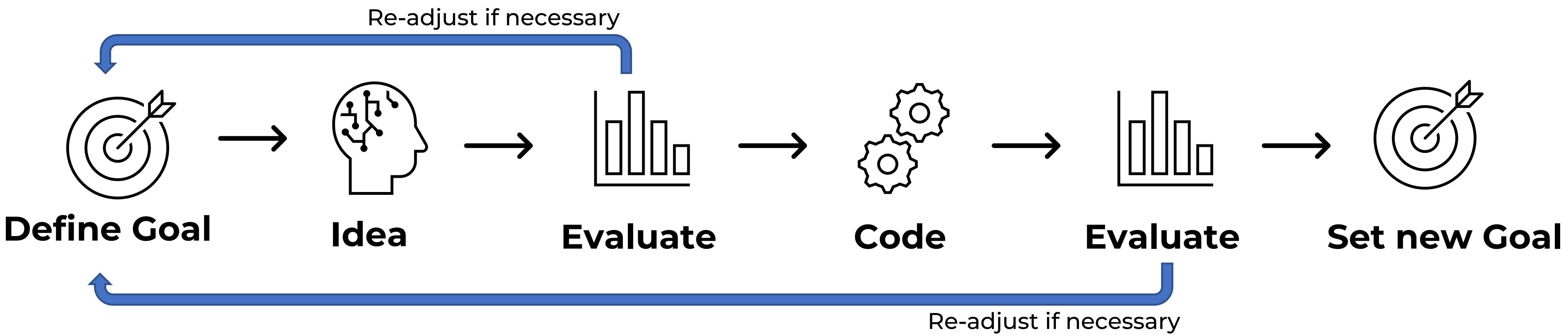
Classify Urban Structure and Vegetation?

How can we detect Heat Islands?
Can we detect changes in urban structure, vegetation etc?

5. Output Format

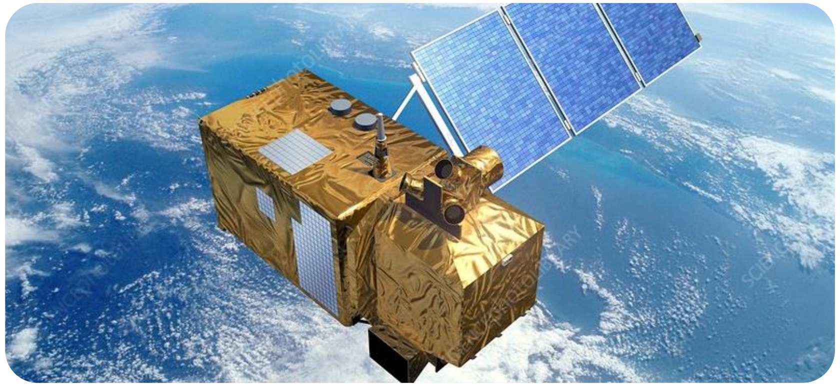
What will the final product look like?

Development Process



The Project Scope was defined loosley and contained many complex points. As it was hard to determine what is achievable in time, and how big the workload of each step would be, we agreed on taking an iterative Go-with-the-Flow process.

Data Scources



Landsat 8

- Spacial Resolution: 30 – 100 Meters
- Temporal Resolution: approx. 16 days
- 11 different Spectral Bands
- Thermal Infrared Bands!

Sentinel 2

- Spacial Resolution: 10 – 60 Meters
- Temporal Resolution: 5 days
- 13 different Spectral Bands
- NO Thermal Infrared Bands!

Acessing Landsat 8 Data via Sentinelhub

We created a function that lets us fetch Landsat 8 Data through the Sentinelhub API. This function allows us to:

- Select an arbitrary Bbox as Area of Interest
- Select an arbitrary Time Intervall
- Select the maximal cloud coverage allowed

The function also makes sure corrupted data (e.g. pictures with many missing values) is not downloaded.

```python
def get_landsat8_range(aoi=None,config=None,year_range=None,
                        month = None,date_range=(1,30),maxcc=.1):
    '''
    Download uncorrupted landsat8 image for a given time range,cloud coverage.
    This function makes sure that you don't get any image which requires Mask data or have certain pixels
    with missing data.

    Args:
        param aoi: Area of Interest.
        type aoi: shapely.geometry.multipolygon.MultiPolygon.

        param year_range: list of range of year for which we want to download image.
        type time_interval: list
```

1 → 2 → 3 → 4 → 5

Fetch Data | Calculation | Detection | Data Pipeline | Dash App

Determining and Calculating Metrics

NDVI as Vegetation Index

$$NDVI = \frac{NIR - RED}{NIR + RED}$$

- Ranges between 1 and -1
- Can be used to determine how much (healthy) vegetation lies in a given area

NDWI as Moisture Index

$$NDWI = \frac{NIR - SWIR}{NIR + SWIR}$$

- Ranges between 1 and -1
- Can be used to determine the water content (or moisture) in a given area

Landsurface Temperature

$$\mathrm{T} = \frac{K_2}{\ln(\frac{K_1}{L_\lambda} + 1)}$$

- Air temperature can’t be calculated using satellite data
- Landsurface Temperature is a good indicator and can be calculated using Landsat

Finding the right Detection Algorithm

Blob Detection via OpenCV?

Self Code it?

Laplacian of Gaussian?

Difference of Gaussian?

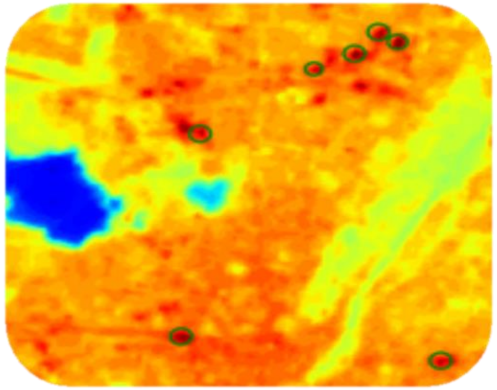
Determinant of Hessian?

1 Fetch Data → 2 Calculation → 3 Detection → 4 Data Pipeline → 5 Dash App

Determinant of Hessian Algorithm

- Detects blobs by finding maximas in the matrix of the Determinant of Hessian of the image
- Detection speed is independent of the size of blobs
- Threshhold parameter controls significance level
- Max.sigma parameter controls size of detected area
- Detection internally performed on b/w images

$$H(f,g) = \begin{bmatrix} 0 & \frac{\partial g}{\partial x_1} & \frac{\partial g}{\partial x_2} & \cdots & \frac{\partial g}{\partial x_n} \\ \frac{\partial g}{\partial x_1} & \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \, \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \, \partial x_n} \\ \frac{\partial g}{\partial x_2} & \frac{\partial^2 f}{\partial x_2 \, \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \, \partial x_n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial g}{\partial x_n} & \frac{\partial^2 f}{\partial x_n \, \partial x_1} & \frac{\partial^2 f}{\partial x_n \, \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$



BUT: As Detection is based on curvature, the algorithm detects areas of high temperature as well as low temperature, so we had to adjust some more...

Determinant of Hessian Algorithm

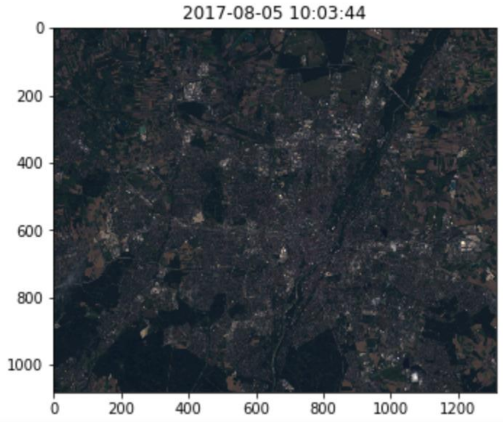
We classify a point as heat island only if its mean temperature is higher than the 0.98 percentile of the temperature of its surrounding area.

As mentioned before it is hard to find a clear definition of what constitutes a heat island, so we came up with the above definition by ourself.

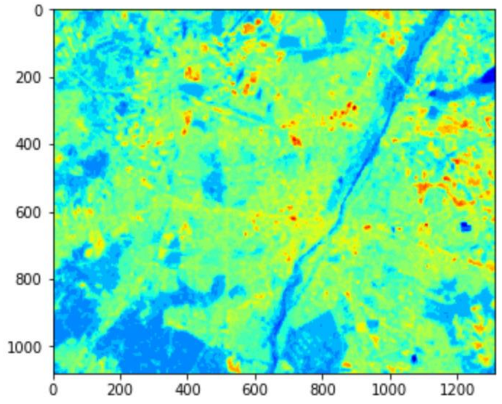
With variation of the q and threshold parameters, different results can be explored.

```
def temperature_threshold(vdesired, blobs):
    vfinal = []
    vrange = 10
    vper = np.percentile(vdesired, q=98)
    for blob in blobs:
        y, x, r = blob
        y, x = y.astype(np.int64), x.astype(np.int64)
        #     vmean = vdesired[x-vrange:x+vrange,y-vrange:y+vrange].mean()
        vmean = vdesired[y - vrange:y + vrange, x - vrange:x + vrange].mean()
        if vmean > vper:
            vfinal.append(np.array([y, x, r]))
    vfinal = np.array(vfinal)
    return vfinal
```

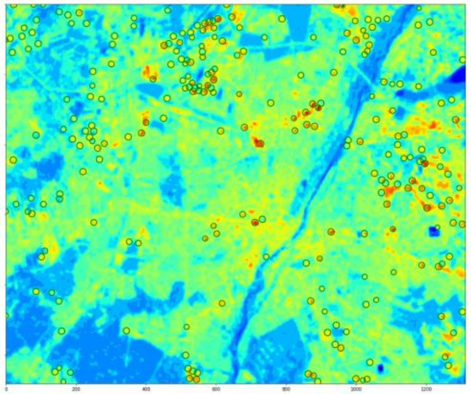
Determinant of Hessian Algorithm



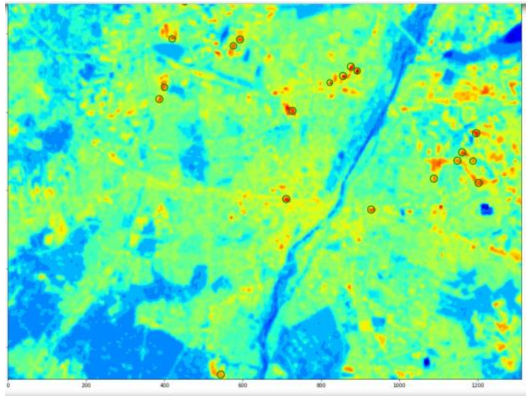
True Colour Image



Heat Map of LST



Circles are Maxima in the DOH



Only Areas with high LST remain

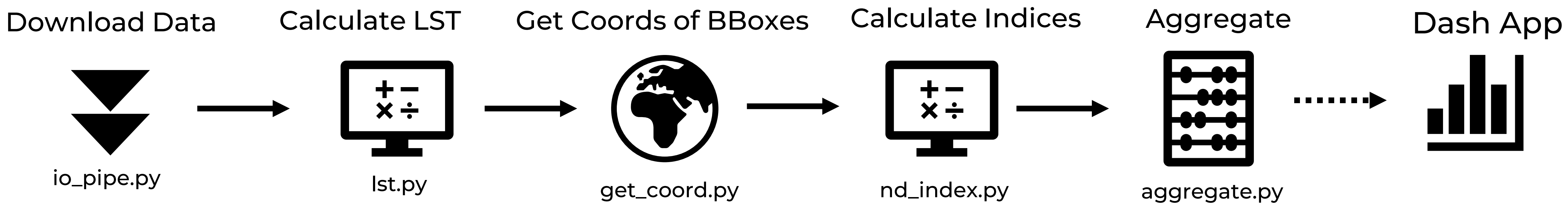
1. Computing LST

2. Determinant of Hessian

3. Filter out unplausible Points

Build a Data Pipeline

Code Workflow



Each function utilizes multiple subfunctions.

1 Fetch Data → 2 Calculation → 3 Detection → 4 Data Pipeline → 5 Dash App

Building an App

After we were able to detect heat islands, we needed to put the detected points in context with NDVI and NDWI, as well as their temporal development. As our goal was to build an exploratory tool, it also needed to include a great amount of interactiveness. We decided that building an app would be the beast way to achieve all of these requirements. The workflow looks as following:

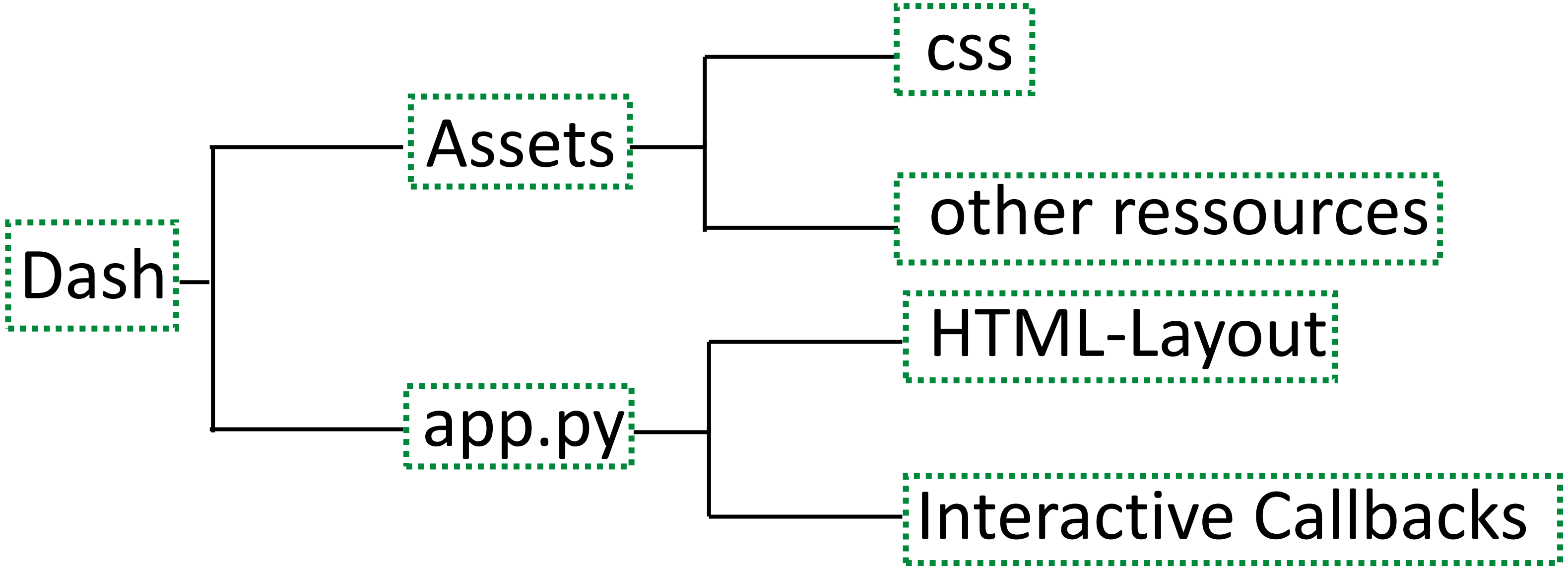
1. Select Area of Interest
2. Possible Heat Islands are detected automatically
3. Examine Heat Island Candidate in Close up View
4. Switch Filter to check LST/NDVI/NDWI
5. Examine Time Series Development

1 → 2 → 3 → 4 → 5

1. Fetch Data
2. Calculation
3. Detection
4. Data Pipeline
5. Dash App

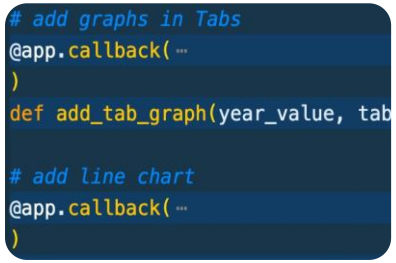
Building an App using Dash

Dash is a Framework for building Web Analytic Applications. Based on Flask, plotly.js and React.js, dash Apps are completely coded in Python and rendered in the Web Browser. Afterwards they can simply be shard via URLs.



css

HTML-Layout



Callbacks

Lack of sufficient Data

Landsat

- Temporal Resolution limits amount of data points heavily
- Spacial Resolution makes it hard to detect small areas

Weather

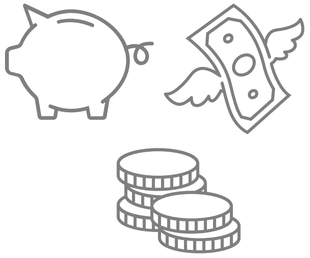
- Naturally heat islands only occur during the summer months
- Cloud Coverage Filter needs to be low
- High Variance of Temperature

Our Analysis is very limited by the amount of Data Points we get. A plausible time Development Analysis is basically impossible with the Data we have.

Other Data Sources

Possible Solutions

Non Public (Paid) Satellite Data with higher spacial and temporal Resoltion



Auxiliary Data: If a heat Island is located near a temperature station, full time series data can be obtained

Munich-Maxvorstadt Highest Maximum Temperatures, 1982-2021

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	Ann
2021	14.4	21.2											
2020	13.8	16.9	19.9	25.1	25.1	28.1	32.9	34.1	29.7	22.3	21.2	14.1	34.1
2019	8.4	19.2	20.6	26.9	23.0	35.0	34.8	32.4	27.8	24.7	17.4	15.9	35.0
2018	17.0	6.0	18.2	30.5	30.5	29.9	34.6	36.1	32.3	25.9	20.7	14.8	36.1
2017	12.9	21.3	24.3	24.6	32.7	36.1	34.7	36.5	24.3	28.3	18.5	14.3	36.5
2016	15.8	18.8	24.4	24.4	29.2	32.7	35.4	32.7	30.5	23.5	19.5	14.2	35.4
2015	16.9	15.0	18.7	24.9	29.3	31.4	37.6	36.8	32.2	23.9	20.6	15.3	37.6
2014	17.0	20.1	23.1	23.0	30.8	34.6	33.8	30.9	27.7	27.8	23.6	14.7	34.6
2013	15.3	7.4	18.2	26.9	25.9	36.2	37.7	36.4	29.8	25.6	20.6	16.5	37.7
2012	11.6	14.7	22.6	32.1	31.3	33.0	33.4	35.6	28.5	24.7	19.3	20.7	35.6
2011	15.2	17.9	19.8	26.4	28.8	29.9	29.3	35.6	30.9	25.0	18.8	16.7	35.6
2010	7.4	14.7	24.1	27.0	27.1	33.6	34.3	31.3	25.4	23.5	19.9	14.5	34.3

Munich Climate Tables, 1982-2011 (uni-muenchen.de)

Tools for Coding and Communication

GitHub Actions

Dash
by plotly

python

sentinelhub

GeoPandas

scikit-image
image processing in python

slack

Trello

zoom

Splitting of Tasks

Alex
- IO Pipeline
- Dash App
- NDVI/NDWI Calculation

Sanchit
- LST Calculation
- Heat Island Detection

Ke
- LST Calculation
- Time Series Analyis

Simon
- Heat Island Detection
- NDVI/NDWI Calculation

What did we achieve?

By using an iterative Workflow, we managed to deconstruct a wide and complex topic into smaller steps that we could find Solutions for. If we recall the scope of our projects our main achievements are:

- Building a Data Pipeline on Landsat ✓
- Evaluate Metrics and calculate LST and Descriptive Indices ✓
- Scalability on arbitrary Cities ✓
- Implementing Detection Algorithm for possible Heat Islands ✓
- Building an interactive Dash App that allows exploration ✓

What did we not achieve?

As Time, Resources and Data were limited, there were also things that we could not achieve:

- Algorithmic Classification of City Surface Types
- Algorithmic Classification of Vegetation
- Plausible Time Series Analysis

What we learned..

- Satellite Data: Spectral Bands, Wave Lengths, Resolutions etc. in itself is a huge and interesting field
- Coding: Working with Spatial Data e.g. GeoPandas, EOlearn, Sentinelhub or ScikitImage as well as building an interactive App
- Deconstructing a highly complex task into small steps while working with a project partner
- Scrum methods accompanied by GitHub

Looking back...

We all learned a lot, and it had it's very own challenges as well as opportunities to work on a loosely defined project. A closer defined Guideline regarding the expectations from the beginning on would probably have been easier to work with, since many critical points only emerged while we were exploring or working towards them. If those could have been avoided preemptively, we would have been able tackle other issues such as vegetation or urban structure classification, that we could not include in the given timeframe.

We want to thank our Tutors and our Project Partner!