Detecting and Analyzing Solar Panels in Switzerland using Aerial Imagery (SolAI)

Adrian Meyer
Institute Geomatics
University of Applied Sciences
Northwestern Switzerland
Team

Institute Geomatics @ FHNW

Adrian Meyer
Data Scientist

Prof. Denis Jordan
Statistics & Mathematics

Prof. Martin Christen
Geoinformation & Computer Graphics

Project Partners

Martin Hertach
Federal Office for Energy (BFE)

Peter Barmet
Energy Department of Canton Aargau (AG)
SolAI – Detection of Solar Systems
IGEO/FHNW and Federal Office for Energy (BFE)
### Object Information

**Suitability of roofs for the use of solar energy** (Swiss Federal Office of Energy)

<table>
<thead>
<tr>
<th>Suitability</th>
<th>Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>Roof area [m²]</td>
<td>282</td>
</tr>
<tr>
<td>Orientation [*]</td>
<td>25</td>
</tr>
<tr>
<td>Inclination [*]</td>
<td>32</td>
</tr>
<tr>
<td>Earning [CHF]</td>
<td>3100.0</td>
</tr>
<tr>
<td>More info</td>
<td>sonnendach.ch</td>
</tr>
</tbody>
</table>
Swiss Buildings 3D Dataset
Swissimage Dataset

10cm GSD
(limited access)

Dataset:
300 MB / sqkm
Uncompressed TIF

PNG Tiles:
1000x1000 Px
(2-3 Mbyte)

25cm GSD
Summary of Input Data

• **We have:**
  • Areal imagery (partially in 10cm², partially 25cm² per Pixel)
  • Vector data / 3D Data of all roofs in Switzerland
  • Roof size and solar potential
  → There are around 2 million buildings in Switzerland in total

• **We don’t have:**
  • Location, Size, and Type of solar panels (PV, Thermal)


Images: DeepSolar/Stanford
Common Object Detection Tasks

Classification

Classification + Localization

Object Detection

Instance Segmentation

CAT

CAT

CAT, DOG, DUCK

CAT, DOG, DUCK

Single object

Multiple objects
Faster RCNN & TF to Identify Tiles with Panels

Region-based CNN, implemented with Tensorflow
Train your own Solar Detector!


• Available under: tinyurl.com/solar-detect
Pre-Detection

Photovoltaic Systems: 92% mAP
Thermal Systems: 62% (ca. 30% are detected as PV)
Multilayered Workflow

• Split *Swissimage* Dataset into 1000x1000 px tiles
• Using Faster RCNN to identify tiles with Solar Panels
• Letting trained professional experts specify the geometry of a few thousand solar systems using Cloud Contribution Client
• Train Mask RCNN to find geometry in single class paradigm
• Run Inferencing on multi GPU HPC over the total area of Switzerland
• Read/Write on NoSQL Databases
• Train Xception + Random Forest to decide on class type of panel
• GDAL Geoconversion to vector with joined attributes
Next Step: Mask RCNN
Instanciation
Cloud Contribution Client for Labelling

Code-Sprint: Europython 2019
Labelling Workshop

• 7’839 Image Tiles
• 31’401 Polygons (22K PV)
• 5 Days by 10 Experts
Images 1 & 2 of 7’839
Generating Masks as PNGs
Polygon Size Distribution of Solar Installations

Area in Square Meters (m²)

Number of Samples
## Framework Selection

<table>
<thead>
<tr>
<th></th>
<th>PyTorch</th>
<th>TensorFlow</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Time</td>
<td>160 ms</td>
<td>197 ms</td>
</tr>
<tr>
<td>Preprocessing</td>
<td>48 ms</td>
<td>22 ms</td>
</tr>
<tr>
<td>Forward prop</td>
<td>31 ms</td>
<td>55 ms</td>
</tr>
<tr>
<td>Backward prop</td>
<td>33 ms</td>
<td>120 ms (parallel reduce)</td>
</tr>
<tr>
<td>All reduce (reduce+broadcast)</td>
<td>48 ms</td>
<td></td>
</tr>
<tr>
<td>Images</td>
<td>Raw image</td>
<td>TFrecord</td>
</tr>
<tr>
<td>kernel</td>
<td>bn-fw_tr_1C11_singleread</td>
<td>bn_fw_tr_1C11_kernel_new</td>
</tr>
</tbody>
</table>
PyTorch

- Open Source Library for Machine Learning (BSD-License)
- Based on Torch Framework (Lua, C++, CUDA), published in 2002
- PyTorch was published in October 2016
- Main Developers are the AI R&D Teams of Facebook
- Development with Python

Advantages:
- Pythonic Interface
- GPU Support with nice Interface
  - `a.from_numpy`/`a.numpy` torch tensor bridge
  - Pretrained Models available (Torchvision)
  - Multiple Optimizers (SGD/Adam/etc.)
Project: Mask RCNN for Early Warning Detection of Avalanches (Stamm, 2019)
Hardware: HPE Apollo 6500

- 48 cores
- 192 GB RAM
- Attached to 120 TB HD (~1 GB/s)

Nvidia Tesla V100 SXM2
- 21 Billion transistors
- 5120 CUDA-cores
- 900 GB/s Mem-Bandwidth
- 12nm
- 300W

JupyterHub using:
- Python 3.7 – Kernel
- Python 3.6 – Kernel
- R-Kernel
- Custom Kernels

13.3.2020
Running Pytorch in Jupyterlab
Extending Models & Multi GPU Support

• In PyTorch we can add new custom datasets for object detection and instance segmentation by inheriting the class `torch.utils.data.Dataset`.

• PyTorch provides an example for that: [https://pytorch.org/tutorials/intermediate/torchvision_tutorial.html](https://pytorch.org/tutorials/intermediate/torchvision_tutorial.html)

• The tricky part is to support our 4 GPUs. Data Parallelism is implemented using `torch.nn.DataParallel`.

• PyTorch doesn’t really provide many examples for multi-GPU, so it was a little bit try and error.
import os
import numpy as np
import torch
import torch.utils.data
from torchvision import transforms, utils, datasets
from PIL import Image, ImageDraw, ImageFont, ImageOps

class SolarDataset(torch.utils.data.Dataset):
    def __init__(self, root, transforms=None):
        self.root = root
        self.transforms = transforms

        self.imgs = list(sorted(os.listdir(os.path.join(root, "PNGImages"))))
        self.masks = list(sorted(os.listdir(os.path.join(root, "ObjMasks"))))
        self.labs = list(sorted(os.listdir(os.path.join(root, "LabMasks"))))

    def __getitem__(self, idx):
        img_path = os.path.join(self.root, "PNGImages", self.imgs[idx])
        mask_path = os.path.join(self.root, "ObjMasks", self.masks[idx])
        labels_path = os.path.join(self.root, "LabMasks", self.labs[idx])

        img = Image.open(img_path).convert("RGB")
Loss Graph Needs ±6 Epochs

Model Run with Separate Classes (PV // THM // Other)
Preliminary Results: Metrics for Class Combination

**Precision**

- Bbox 0.75
- Bbox 0.5
- Segm 0.75
- Segm 0.5

**Recall**

- Bbox 0.75
- Bbox 0.5
- Segm 0.75
- Segm 0.5

Legend:
- 1: PV + Thm + Other
- 2: PV / Thm + Other
- 3: PV / Thm / Other
Statistical Linearity

PV+Thm+Other Bounding box

\[ y = 1,3286x - 0,3368 \]
\[ R^2 = 0,9968 \]

PV+Thm+Other Segmentation

\[ y = 1,6126x - 0,5041 \]
\[ R^2 = 0,9879 \]
Challenges
Challenge: Small Panels?

• Single Class Paradigm (PV+Thm+Other) including no modules smaller than 3qm did not increase Precision or Recall

⇒ Cleaning up the Labels is more important
Challenge Labels: Class «Others»

- These elements probably are photovoltaic panels but display somewhat difficult characteristics
Challenge Labels: Class «Other»

• These are most likely NOT solar panels
Challenge Labels: Class «Others»

• Difficult
Challenge: Labelling Mistakes
Computational Load for a Single Run over Complete Switzerland

• 4 Million Images with 2-3 Mbyte

• Inferencing & IO Operation Duration per Image:
  CPU (44 Cores): 2.1 Seconds (100 Days)
  1 GPU: 1.6 Seconds
  4 GPUs: 1.0 Seconds (46 Days)
Optimization

• Currently a Country Level Inferencing Run Takes ±10 Days
• Still Potential with Model Optimizations
• Inferencing Times for Tensorflow Possibly Faster but Requires TF Records
• Increase the Load on GPUs
• Hybrid CPUs/GPUs on Multiple HPCs
More Potential: Optimization for higher GPU Load

<table>
<thead>
<tr>
<th>NVIDIA-SMI 418.87.00</th>
<th>Driver Version: 418.87.00</th>
<th>CUDA Version: 10.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPU Name</td>
<td>Persistence-M</td>
<td>Bus-Id</td>
</tr>
<tr>
<td>Fan</td>
<td>Temp</td>
<td>Perf</td>
</tr>
<tr>
<td>0</td>
<td>Tesla V100-SXM2...</td>
<td>Off</td>
</tr>
<tr>
<td>N/A</td>
<td>35C</td>
<td>P0</td>
</tr>
<tr>
<td>1</td>
<td>Tesla V100-SXM2...</td>
<td>Off</td>
</tr>
<tr>
<td>N/A</td>
<td>35C</td>
<td>P0</td>
</tr>
<tr>
<td>2</td>
<td>Tesla V100-SXM2...</td>
<td>Off</td>
</tr>
<tr>
<td>N/A</td>
<td>34C</td>
<td>P0</td>
</tr>
<tr>
<td>3</td>
<td>Tesla V100-SXM2...</td>
<td>Off</td>
</tr>
<tr>
<td>N/A</td>
<td>38C</td>
<td>P0</td>
</tr>
</tbody>
</table>
Outlook

• Pytorch currently just supports ResNet50
  • We want to try out ResNet101 or ResNet+Inception v2 but at the moment we would need Tensorflow for it
  • Trying different sets of optimizers
    • Adapt Learning Rate dynamically

• Some more manual labelling

• Post-Classification Strategies
Big Data Inferencing & Classification Workflow

Run multiple models for inferencing, using heuristic measures for edges and segment probability

→ Rasterio / fiona / GDAL
Include Near-Infrared Data
Random Forest & Xception for Post-Classification

- Cadastre Data
- GIS Attributes
- Xception RGB & NIR

Random Forest Classifier
Thank you!