HIDA Hackathon on Computer Vision

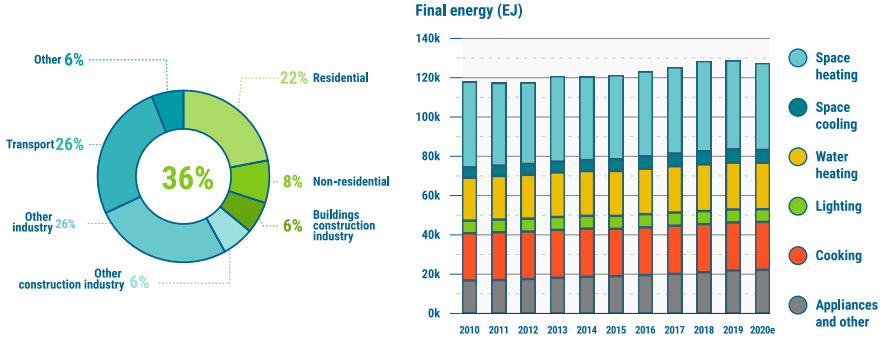
Day 1

Markus Götz, Sebastian Ziegler, Oskar Taubert, Fabian Isensee, Charlotte Debus

Helmholtz AI@KIT, Helmholtz Imaging@DKFZ / 2024-02-27

What Can You See?

Thermal Bridges



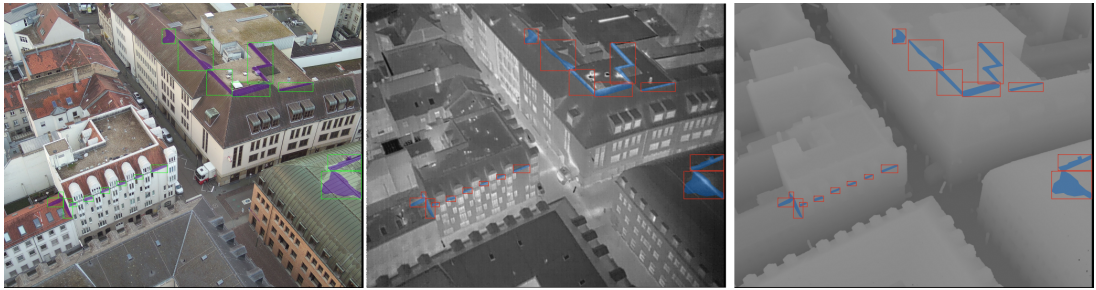
Source: United Nations Environment Programme (2021), Global Status Report for Buildings and Construction: Towards Zero-emission, Efficient and Resilient Buildings and Construction Sector, https://globalabc.org/resources/publications/2021-global-status-report-buildings-and-construction

- **Thermal bridges** are **weak points** of building envelopes leading to **energy loss**

→ mold growth, decrease in building longevity, uncomfortable living spaces

Thermal Bridge Detection

Data

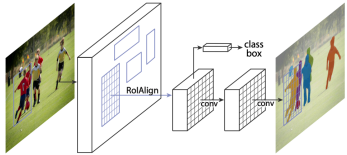


- 997 drone images, each 2680×3370 pixels
- 5 channels $\rightarrow$ RGB, thermal information and depth map
- 42 GB total data volume

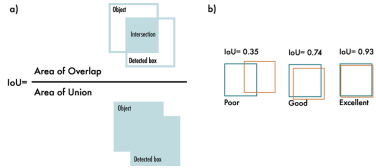
$\rightarrow$ `/hkfs/work/workspace_haic/scratch/qx6387-hida-hackathon-data/train`

Thermal Bridge Detection
Model

- Challenge: binary **semantic segmentation** challenge
- Baseline: **Mask R-CNN** with ResNet 50 backbone
- Metric: **intersection over union (IoU)** of semantic masks
  - Train IoU: 0.518
  - Validation IoU: 0.265
  - Test IoU: 0.216



Source: He, Kaiming, et al. "Mask r-cnn." Proceedings of the IEEE international conference on computer vision. 2017.



Source: Terven, J., Cordova-Esparza, D. (2023). A comprehensive review of YOLO: From YOLOv1 to YOLOv8 and beyond. arXiv preprint arXiv:2304.00501.

Skeleton Code

- GitHub repository
- PyTorch implementation
  - `dataset.py` contains index-able data access
  - `model.py` has Mask R-CNN implementation
  - `train.py/predict.py` training loop and inference code
  - `.sh` template job scripts
- **Note:** maintain the API, we will use the `predict.py` for evaluation

```
|-- LICENSE
|-- README.md
|-- dataset.py
|-- images
    |-- example.png
|-- model.py
|-- predict.py
|-- predict.sh
|-- requirements.txt
|-- team-access.sh
|-- team.csv
|-- train.py
|-- train.sh

2 directories, 12 files
```

Your Mission

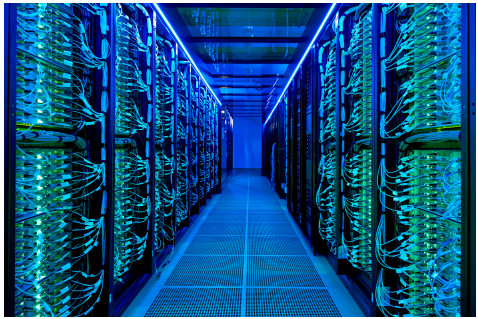
... should you choose to accept it

- Aim: **achieve the highest Intersection-over-Union on held-out test data**
- Everything goes...
  - **traditional computer vision** approaches or classical machine learning
  - (Pretrained) **deep learning** models
  - **API requests** to online models, e.g. Segment Anything by Meta (SAM)
- Things to watch out for
  - **Cheating** in any form will lead to **disqualification**, e.g. printing better IoU values
  - Prizes

HAICORE Crash Course

HAICORE

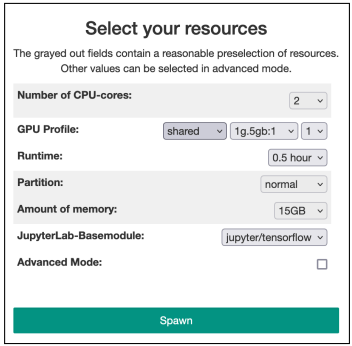
- **H**elmholtz **AI** **CO**mputing **RE**sources
- Two installation sides
  - Forschungszentrum Jülich (72 GPUs)
  - **Karlsruhe Institute of Technology (72 GPUs)**
- Registration process
- HAICORE@KIT Docs
- Reservation: `hida`
- Intended for your general AI research even after the hackathon



- Browser access via **Jupyter**
  - KIT: haicore-jupyter.scc.kit.edu
  - Drop-down resource selection
- Bare metal access via `ssh`

```
ssh <USER>@haicore.scc.kit.edu
```

  - Unix shell
- **SLURM** resource scheduling system
- Documentation: Jupyter@KIT



- Commonly used **software preinstalled**
  - **Why:** no administrator rights
  - Usually multiple versions available
- Troubleshooting: self-compilation
- **NOTE:** modules always first

Searching

```
module spider cuda
```

```
Versions:
    devel/cuda/11.8
    devel/cuda/12.0
```

Loading

```
module load devel/cuda/11.2
```

Collections

```
module save <NAME>
module purge
module restore <NAME>
```

- Various **Python** versions as **system packages** available
- **Virtual Environments (venv)** for package management

```
python3.9 -m venv <PATH>
source <PATH>/bin/activate
pip install ...
```

- Installing an active virtual environment in Jupyter

```
python -m ipykernel install --user (--display-name <NAME>)
```

```
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --gres=gpu:1
#SBATCH --time=40:00
#SBATCH --mail-type=BEGIN
#SBATCH --mail-user=markus.goetz@kit.edu

module restore <COLLECTION>
source ~/.venv/<VENV>/bin/activate
srun python -u <SCRIPT>
```

- Generally **two parts** for a **SLURM job** script
  - Declarative **header** for **resource** allocation/request
  - Main **body** with actual processing commands
  - **REMEMBER:** restore modules and venv first

- **Submission** to the scheduler

```
sbatch <JOB_SCRIPT>
```

- **Interactive** jobs on the shell, block and spawns remote shell

```
salloc <PARAMS>
salloc --nodes=1 --gres=gpu:1
```

- **Monitoring** job queue

```
squeue
```

- **Cancelling** jobs

```
scancel <JOB-ID>
```

Teams

The Groups

Group	Member 1	Member 2	Member 3
Testy Tortoise	dz4120	qx6387	hgf_pdv3669
Loquacious Lemurs			
Fastidious Falcons			
Cantankerous Crabs			
Paradoxical Parrots			
Quixotic Quails			
Ubiquitous Unicorns			