Solar Panel Detection From Aerial Images

DUCOURAU Maxime (329544), PERBET Jean (341418), ZOGHLAMI Mehdi (326381)

1 Abstract

This project leverages Swiss national aerial imagery to address gaps in the Swiss photovoltaic installation database. High-resolution images are processed using Convolutional Neural Networks (CNNs) to detect and segment solar panels automatically. A U-Net architecture, enhanced with data augmentation techniques, and DeepLabV3+ demonstrated strong performance, delivering promising results for this task.

Keywords

Semantic Segmentation, Solar Panels, U-Net, DeepLabv3+

2 Problem statement

Given a 1000×1000-pixel aerial image of Switzerland, captured at a resolution of 10 cm per pixel, classify each pixel to determine whether it covers a solar panel or not. This problem falls under the category of semantic segmentation.

3 Data and Labeling

3.1 Data extraction

Since we are using **supervised methods**, the first step is to generate labeled training data that we can feed to the model. Our main data source is **Swisstopo database** made available by the Federal Office of Topography. This dataset is a composition of aerial photographs of Switzerland with a ground resolution of 10 cm per pixel in the plain areas.
We sampled 75 images of 10'000×10'000 pixels (equivalent to 1km×1km), from which we extracted 750 images of 1000×1000 pixels. In order to ensure that our images contained **roofs**, we used during our search another dataset called `solarenergie-eingung-daecher` which tells the roofs' positions. This way, we only kept images that had a number of roofs higher than a chosen threshold. We also ensured that we also got **rural areas** to have a well diversified dataset.

3.2 Labeling

Using Encord, a data annotation platform, we **manually annotated** the 750 images with 2 different labels: **solar_panel** and **unsure**. To be conservative, we used the **unsure** label for regions where the object type could not be determined confidently. The annotation process was simple; it consisted of covering the target regions with **labeled polygons**.

4 Data Pre-Processing

4.1 Reducing data complexity

Using `solarenergie-eingung-daecher` dataset, we preprocessed our images by **coloring** everything that is **not a roof** in magenta. This step reduces the complexity of the data. The assumption that motivated this step is that solar panels are only placed on roofs. This way we can ensure that the model quickly understands that magenta pixel (equivalent to non-roof pixels) cannot be part of a solar panel.

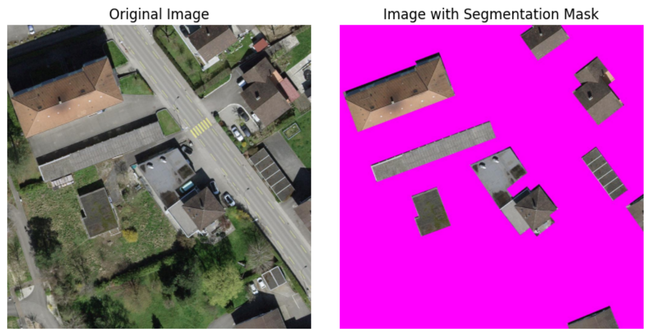


Figure 1: Original and preprocessed images

4.2 Splitting the data

The dataset was divided into three subsets: **training** set (80%), **validation** set (10%), and **test** set (10%). This division ensures that the model is trained, fine-tuned, and evaluated on distinct subsets of data, preventing overfitting and ensuring robust performance on unseen data.

5 Methods

5.1 Data Augmentation

We applied data augmentation techniques using the **Albumentations** library. Data augmentation adds **diversity** to the training data by applying random transformations to the images, helping the model **generalize better** on unseen data. The idea is that we simulate inputs the model might encounter in **real-world** scenarios by altering the original data without changing its fundamental meaning. The transformations were only applied on the **training set** (we keep our validation and test dataset as-is):

- **Rotation** `A.rotate`: Each image was rotated using one of the following fixed angles: **0°**, **90°**, **180°**, or **270°**. Each rotation angle was applied with a probability of **1/4** to ensure a balanced distribution of rotations. This helps the model become **rotation-invariant**.

- **Horizontal Flip** `A.HorizontalFlip`: Each image was flipped horizontally with a probability of **1/2**. This helps the model become **invariant** to **up-down** orientation changes.
- **Vertical Flip** `A.VerticalFlip`: Similarly, each image was flipped vertically with a probability of **1/2**. This helps the model become **invariant** to **left-right** orientation changes.

Augmentations should help the model learn to detect solar panels under **different orientations and perspectives**, mitigating the risk of over-fitting and hopefully improving performance on real-world aerial images.

5.2 Model: U-Net

We used **U-Net**[3] as a first architecture to tackle the problem. U-Net is a Fully Convolutional Network (FCN) designed specifically for image segmentation.

5.2.1 Motivation

Unlike generic Convolutional Neural Networks , U-Net focuses on predicting **pixel-level** outputs, which makes it an excellent choice for tasks where detailed segmentation masks are required. Moreover, U-Net performs well even with **limited training data**, which is exactly our case.

5.2.2 Architecture

The U-Net architecture has three components:

- **Encoder**: captures features through repeated convolution and max-pooling layers, progressively reducing spatial dimensions while increasing feature depth
- **Bottleneck**: connects the encoder and decoder, providing high-level feature representation
- **Decoder**: Upsamples the features using convolutions and transposed convolutions.

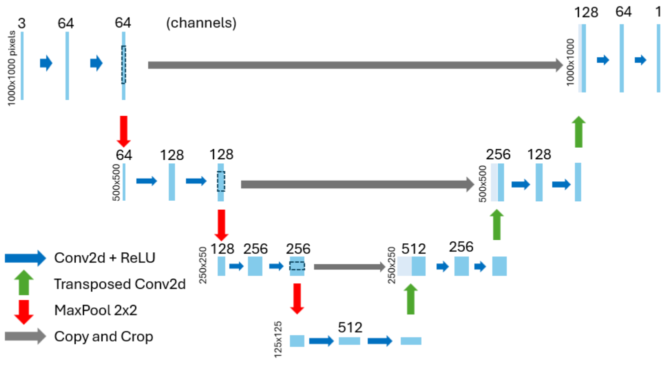


Figure 2: U-Net diagram with 3 layers

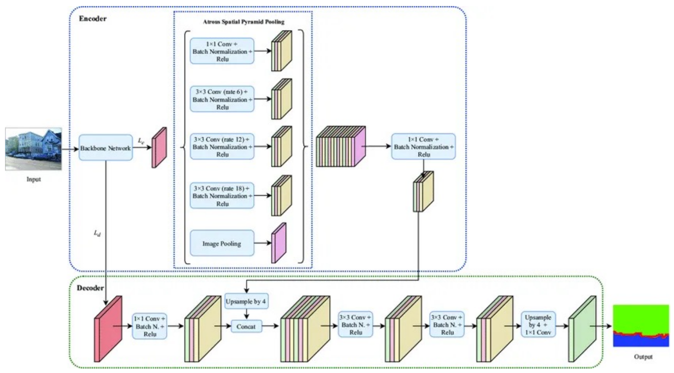
5.3 Model: DeepLabv3+

The second approach was through another model called **DeepLabv3+**[1] which is another architecture of Fully Convolutional Network (FCN). We wanted to test a more complex model than U-Net to check if it would capture best the patterns of the data due to its higher number of convolutions.

5.3.1 Motivation

DeepLabv3+ is a leading semantic segmentation model, utilizing **Atrous convolutions** and the **Atrous Spatial Pyramid Pooling (ASPP)** module to capture multi-scale contextual information and fine details. Its flexible architecture supports various backbones, balancing accuracy and efficiency. DeepLabv3+ is among the leading architectures for semantic segmentation, achieving high accuracy on multiple benchmarks.

5.3.2 Architecture



Encoder
Atrous Spatial Pyramid Pooling
1×1 Conv + Batch Normalization + Relu
3×3 Conv (rate 6) + Batch Normalization + Relu
3×3 Conv (rate 12) + Batch Normalization + Relu
3×3 Conv (rate 18) + Batch Normalization + Relu
Image Pooling
Input
Backbone Network
L_e
L_d
1×1 Conv + Batch Normalization + Relu
Decoder
1×1 Conv + Batch N. + Relu
Upsample by 4
Concat
3×3 Conv + Batch N. + Relu
3×3 Conv + Batch N. + Relu
Upsample by 4 + 1×1 Conv
Output

Figure 3: DeepLabv3+ diagram

5.4 Training

In order to accelerate our training we used **early stopping**; a regularization technique that **halts** training when the validation performance **stops improving**, preventing overfitting to the training data. It monitors the loss after each epoch and stops training if no improvement is observed for a specified number of epochs (patience). The best-performing model parameters are saved before performance starts to degrade. This approach is particularly useful as it **balances training duration with model performance**.

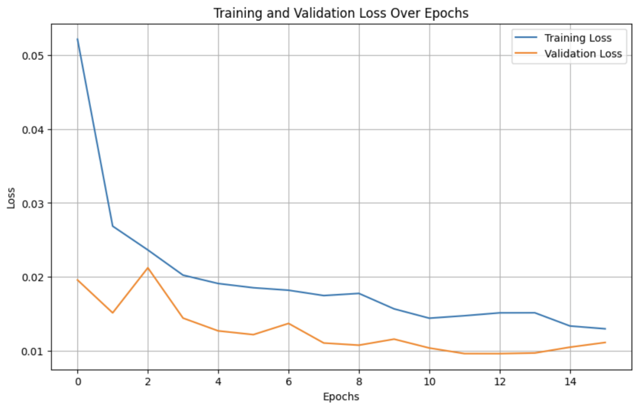


Figure 4: Training and validation loss

5.5 Post-Processing

Our model's outputs exhibited some small characteristic issues, such as **misclassified regions** that do not lie on roofs, **uneven edges** around solar panels, and overall **noisiness**. To address these problems, we implemented post-processing steps to refine the predictions. These included **smoothing the edges** of the predicted areas, **eliminating** predictions that were **too small** to qualify as solar panels, **filling holes** predicted regions, and **removing** all predictions that **don't lie on a roof**.

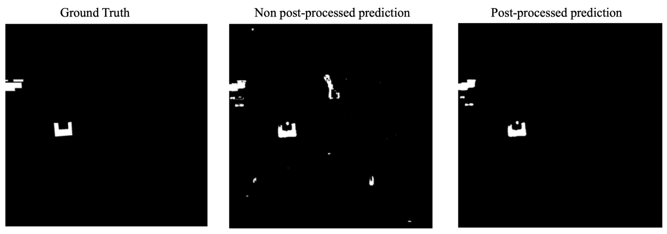


Figure 5: Impact of post-processing

5.6 Fine-tuning

We used `Optuna`, a **hyperparameter optimization** framework, to fine-tune our U-Net's parameters. **Learning rate**, number of **encoder layers**, number of **epochs** and **kernel size** were the parameters we sought to optimize. The optimization process aimed to minimize the validation loss. **Binary Cross-Entropy (BCE)** was chosen as the **loss criterion**, as it is well-suited for our binary classification task. Optuna explored different combinations of hyperparameters systematically and efficiently to identify the configuration that yielded the lowest validation loss. Each trial involved training the model with a specific set of hyperparameters and evaluating its performance on the validation set. Given the high computational and timing resources demanded by this step we only ran it once on U-Net with unprocessed data. The results are displayed in figure 6

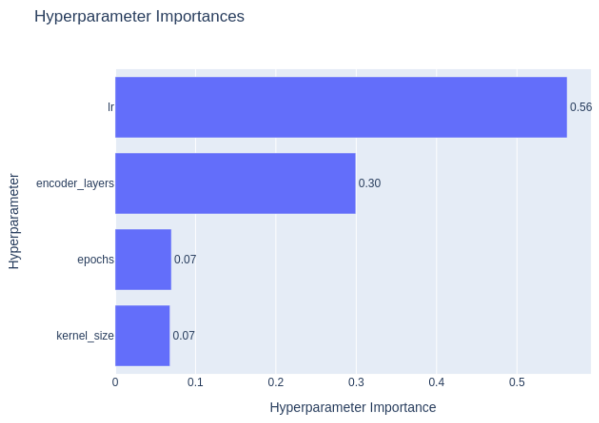


Figure 6: Hyperparameters importance for U-Net

The most impactful hyperparameter is the learning rate. This makes sense since it is very **tightly correlated** with the convergence of the loss. On the other hand kernel size and the number of epochs had **very low impact**. For DeepLabv3+, given the limited amount of time we have, we chose not to fine-tune since it already performed very well as it was initialized.

5.7 Evaluation metrics

We will evaluate our models use two main metrics: **F1-score** and **Intersection over Union (IoU)** defined as:

$$\text{F1-Score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

$$\text{IoU} = \frac{|\text{ground truth} \cap \text{prediction}|}{|\text{ground truth} \cup \text{prediction}|} = \frac{\text{TP}}{\text{TP} + \text{FP} + \text{FN}}$$

The F1 score is useful for understanding classification performance across classes, while IoU assesses the quality of the predicted spatial regions. These metrics are also particularly useful in our case since our classes are **highly unbalanced**, (i.e accuracy is not very insightful).

6 Experiments and Results

During this step we trained all of the models and experimented with them on different inputs.

6.1 U-Net on original data

This model is a U-Net trained on unprocessed data where we feed the images to the model as they are.

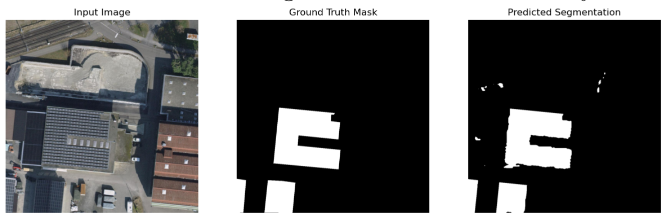


Figure 7: U-Net on unprocessed data prediction

The model distinguishes well solar panel pixels from non solar panel ones.

6.2 U-Net on roofs only data

This model is a U-Net where we filter out non-roof pixels from input images and color them in magenta before feeding the images to the model during training

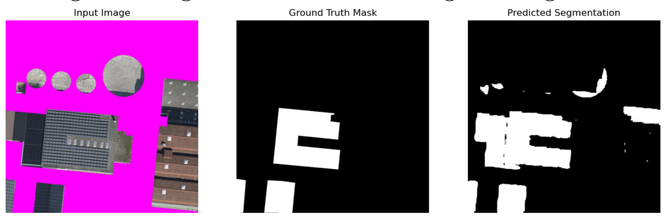


Figure 8: U-Net prediction on roofs only data

The model has a decent prediction but still mistakes a roof section as a solar panel on this example.

6.3 U-Net on augmented data

This model is a U-Net where we augment the input data using random rotations and flips.

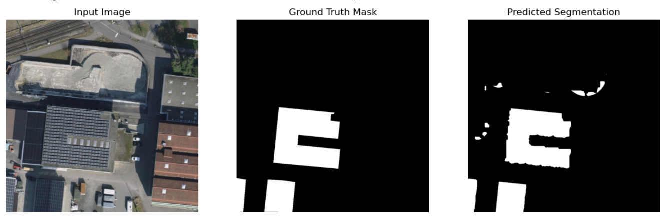


Figure 9: U-Net prediction on augmented data

The prediction is good. The model identifies the solar panels and is able to represent their shapes pretty accurately.

6.4 DeepLabv3+ on unprocessed data

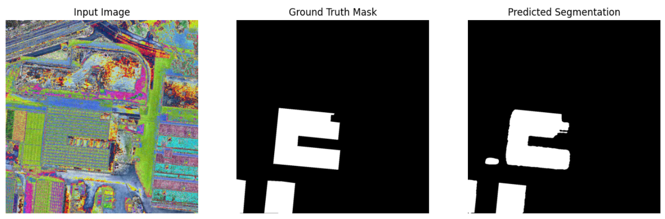


Figure 10: DeepLabv3+ prediction on original data

DeepLabv3+ seems to be offering the best prediction so far. It's worth noting that the input image appears altered when using DeepLabv3+ because the model applies normalization to the image before using it.

6.5 Evaluation

To have a better understanding of our models' performances we test them on a 76 image test dataset which gives the following results:

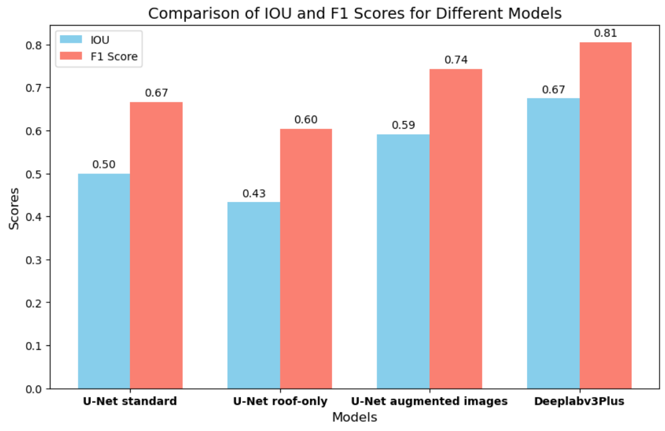


Figure 11: IoU and F1 score for different configurations

For U-Net, we see that **pre-processing** the data to only keep the roofs before inputting **diminishes** both **F1** and **IOU** scores. On the other hand using data **augmentation** makes a real difference and **substantially improves** the metrics. DeepLabv3+ is the best performing model with an **F1-score 0.81** and an **IoU of 0.67**.

7 Conclusion

By leveraging U-Net and DeepLabv3+ architectures, along with rigorous **data processing**, **data augmentation**, and **hyperparameter optimization**, we achieved promising results. Post-processing refined the predictions, addressing issues like noise and edge inaccuracies. Evaluation using F1-score and IoU confirmed the robustness of our approach in handling such unbalanced classes. This study demonstrates the potential of deep learning in aiding renewable energy mapping, paving the way for more accurate and scalable solar panel detection systems.

8 Discussion

Our model was trained on aerial images of Switzerland and could potentially be **applied** to aerial images from **other countries** with similar landscapes. However, a key difference is that roof location data, which we currently use to process images for some versions of the model, may not be available in other regions. Also, `solarenergie-eingung-daecher` is not always accurate which could bias our results for some outputs.

9 Potential improvements

Given the limited amount of time we have, we only labeled 750 images. To scale up the model and make it perform better we need more data. The next step to improve our results is to label more images.
We could also fine-tune DeepLabv3+, which has already very promising results.

10 Environmental concern

Training a segmentation model, especially on large datasets, is associated with significant energy consumption and substantial carbon emissions. [4]. Fine tuning and training every iteration of our models (including failed attempts and deadends) on just 750 images required a total of **120 GPU hours**. It is estimated that one GPU hour generates on 34 grams of CO_2 per GPU hour as a lower bound [2]. We can safely assume that our models generated at least **4.2 Kg of carbon emissions**. In order to track the emissions of training each one of our models we used the library `CodeCarbon` that estimates the amount the carbon emissions of running a piece of code. Here are the results for the training of the final version of each model

Model	**CO_2 emissions (Kg)**
U-Net on original images	0.06
U-Net on roofs images	0.026
U-Net + augmentation	0.05
DeepLabv3+	0.073

Unfortunately, there isn't much we could do to limit the power consumption of the model. From a hardware point of view, we run our models on third-party GPUs which leaves us with little room for optimization too.
Our project had a minimal environmental impact; however, the real challenge arises when scaling this work to real-world applications. Training the model on thousands more images would significantly increase computational time, leading to a much larger environmental footprint.

Acknowledgment

This work was supported and made possible through the valuable assistance of the Swiss Data Science Center (SDSC.)

References

[1] Liang-Chieh Chen et al. "Rethinking Atrous Convolution for Semantic Image Segmentation". In: *arXiv preprint arXiv:1706.05587* (2017).

[2] Jesse Dodge et al. "Measuring the Carbon Intensity of AI in Cloud Instances". In: *arXiv preprint arXiv:2206.05229v1* (2024). [cs.LG].

[3] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *arXiv preprint arXiv:1505.04597* (2015).

[4] Yinlena Xu et al. "Energy Efficiency of Training Neural Network Architectures: An Empirical Study". In: *arXiv preprint arXiv:2302.00967* (2023). [cs.LG].