Solar Potential Analysis using Semi-Supervised Image Segmentation based on Differentiable Feature Clustering

Magnus Müller*
Institute of Cognitive Science
University of Osnabrück
Osnabrück, Germany
magmueller@uni-osnabrueck.de

Michael Ramich*
Institute of Cognitive Science
University of Osnabrück
Osnabrück, Germany
mramich@uni-osnabrueck.de

*Michael Ramich and Magnus Müller contributed equally to this paper.

***Abstract*—Currently, the need for renewable energy rises. For that, solar panels come in handy. Roofs will get more crowded but also solar panel potential needs to be analysed easily on roofs. For this, a whole lot of labeled data is needed to archive high quality industry grade results. Thus, labelers need to be ideally confronted with prelabeled data, so that this labeling process is easier. We analysed a method for unsupervised image segmentation, converted it to a semi-supervised one, and found that this method archives good enough results for our goal.**

***Index Terms*—semi-supervised, image segmentation, roof solar potential analysis, solar energy, solar potential**

I. Introduction

Since the need for energy will rise in the future, there is a big need for renewable sources of energy. Also, fossile energy sources make a country dependend on certain regions and their stability as well as the international politics of a country. Especially in our current time period, this dependence can be seen. [Ritchie and Roser, 2020]

Renewable energy can be provided with mainly solar, water and wind leading the energy cart in Germany. Studies show, that solar energy might provide the best solution for home-owners and industries growing energy demand. We also believe, that homes might want to be self-sufficient and more or less independend from rising energy prices. [Kannan and Vakeesan, 2016]

A. How to fullfil growing energy demand

This can be archived via solar panels on the roof, as they are providing direct energy to the house or industry complex without the need for a big storage and without buying the energy from a provider. The need for solar will rise in the future. [Campbell, 2022]. This is, why we want to analyse the solar-potential for solar roofs, that is, what percentage of the roof is suitable for solar panels and what it's potential is. For that, we need to predict the free space on roofs from satellite images. These can but don't have to be high resolution.

B. What we want to do

As we aim to have a network trained supervised to get the highest quality results out of it, we need a lot of labeled satelite images for that. Thus, there will be a labeling process of images, where each region will be assigned a certain category.

For that training to happen, the data has, as already said, be labled. For this, we want to have a unsupervised network sort our data, so that the labeler is getting precategorized data. With this we aim to limit the spending on labeling and also ease the workflow of the labeler. As seen in Fig. 1 the labeler can easily get only segments, that are suspected to have a roof and then segment the roof and - or - answer if there is a roof in this segment of the picture, see Figure 1

Fig. 1: Unsupervised Image Segmentation on satellite images

C. Our Goal

In this paper we want to explore how and if sufficient prelabeling can be archived using the differentiable feature clustering method of the reference paper. We aim to mimic

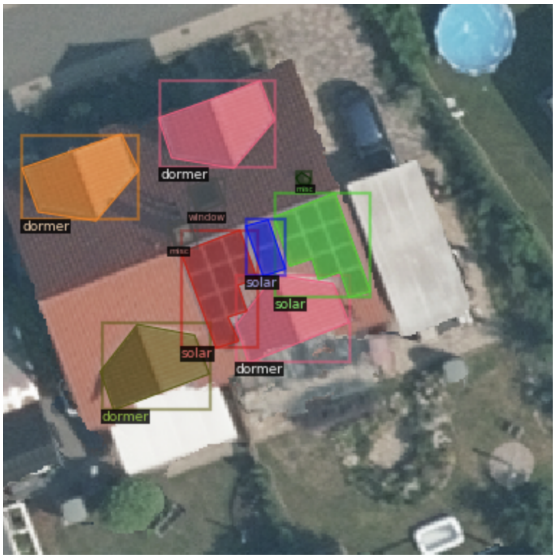


Fig. 2: Supervised approach, segments of each class are needed.

the model structure of this paper and try to improve the architecture using several ideas to archive even better results. We then also explore if pretraining with limited labled data archives better results. In the end we want to compare different architectures, including the vanilla paper architecture and our improved versions and find the best method to archive a good prelabeled dataset. [Kim et al., 2020]

II. Related Work

Here we give a brief overview of related approaches and the field to give some context.

A. Supervised image segmentation

In semantic image segmentation, pixels that meet similar criteria are assigned to the same label. This can be done using different approaches. If many labeled images are available, a supervised approach is possible. Labeled in this case means not only an image with true and false or cat and dog, but an image with polygons around each label. Often there are many different and equivalent labels in one image, all of which need to be segmented and their coordinates stored in an external json file, for example in COCO format. This data can be used later for your training process. For example with detectron2 from Facebook Wu et al. [2019]. Detectron2 provides state-of-the-art detection and segmentation algorithms. It includes many pre-trained models, which lead after a fast training process to excellent results like shown in Fig. 3.

Another supervised method is shown by Yang et al. . Here, user input is used directly as a label to optimize the model. [Yang et al.]

Fig. 3: Results from detectron2

B. Classical unsupervised image segmentation

When segments are very small or few and the resources to create target segments are not available, unsupervised approaches can be beneficial. Traditionally, k-means can be used. In k-means clustering, pixels are divided into k clusters, where each pixel belongs to the cluster that contains the closest mean value. The mean values are optimized iteratively.

C. Semi supervised image segmentation

The approaches can also be combined, e.g., in semi-supervised approaches where some segmented images are used. Semi-supervised learning is a machine learning approach that combines a small set of labeled data with a large set of unlabeled data during training. In our dataset roof masks where given. This means we have some information about the image, but not the complete target image. Using this information to help the model to go in the right direction can also be count to semi supervised learning.

D. Encoder - decoder unsupervised image segmentation

Widespread unsupervised methods also include encoder - decoder models: The encoder maps the images to a low-dimensional feature representation and the decoder attempts to reconstruct the original image from this compact feature space. Sometimes this dual network is cut in half for the final segmentation. In this case, the compact feature representation represents the clusters. [Xia and Kulis]

In our method, we also encode the images, but not in a lower dimension. Constraints can still be defined by the loss function, similar to what previously was done by other authors. [Kanezaki], [Kanezaki and Tanaka]. This will be shown in the following Methods section.

III. Methods

A. Dataset collection

First we needed to collect the dataset. For that we used the Geodatabase NRW and extracted thousand satellite images of

houses including the mask of the houses. We then uploaded them to Google Drive to use them with Google Colab in order to have them in the cloud, ready for training on any device. The images are provided in the PNG format, while the masks are encoded in JSON. [Bezirksregierung-Köln, 2022]

B. Data check

As we encountered errors during first training with a mockup model, we saw that the dataset is inconsistent. Thus, and as it is good practice, we needed to check the data for quality and consistence. For this, we import all files in variables and check weather the number of JSON files matches the number of image files. Furthermore, we checked if every image file has a corresponding JSON file with a mask inside.

C. Mask extraction

To extract the mask from the JSON file we looked at the structure of the JSON file. In each of the files, we found the polygon marking one roof of the corresponded to a object called geometry with type Polygon. Thus, we could extract the polygons using a Regex search for first the string and then the coordinates. After cleanup of the coordinates we checked for valid data as in an equal number of brackets and then mapped coordinates to tuples (x,y), as they were provided in a string format.

D. Loading data into a generator

As there are quite a lot of images, being thousand images, to train with in this exploratory approach, there was a need for a generator to train efficiently and not load all images into memory, which exhausted 24GB of memory easily when trained on simultaneously. Also when using our approach in production, many more images will have to be loaded and trained on. Thus we created a function which yields pairs of (image, mask) from our file dataset, checking weather they exist and are not empty in case any mask is empty and cannot be pretrained on. As the last step, we converted our generator to a Tensorflow dataset using the from generator function from Tensorflow. [1]

E. Improving data quality

Many people spend a lot of time tuning hyperparameters and improving models, but forget the importance of their data. Inspired by Motamedi et al., we want to improve our data quality. [Motamedi et al., 2021] Therefore, we use the masks to calculate what percentage of an image is covered by a roof. If the value is below or above a certain threshold, we simply drop the image. This leads to a much higher quality of our roof dataset on which the learning process is based.

[1] tf.data.Dataset.from_generator()

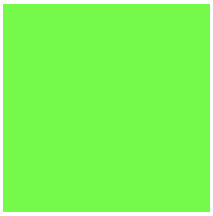


Fig. 4: Results with Batch Normalization

F. Testing data to dataset pipeline

To test the data pipeline until this step, we visually checked the generator by printing the mask and image for several images and validated that the mask indeed matches the roof.

G. Data preprocessing

For data preprocessing we first casted [2] the image arrays to tf.float32 and normalized them by dividing through 255, as this is the maximum value for each RGB color represented in the image array. We then cache the progress in memory, as it is deterministic, thus should not be redone every time. As a last step, we shuffle the dataset and then define the batch size using the shuffle and batch method of Tensorflow.Dataset. The preprocessing is applied through the apply method. [3]

H. Model and training

1) Starting point: Our main idea is inspired by Kanezaki and Tanaka. They use a really simple model, three Conv2d layers with Relu activation function, 100 channels and BatchNormalization. [Kanezaki and Tanaka] For the unsupervised image segmentation process we used the same three constraints for the training process:

1) pixel with similar features should have the same label
2) Spatially close pixel should have the same label
3) there should be many different clusters

The constraints contradict each other and will never be fulfilled together completely. 1 and 2 are achieved by two different loss functions, one cross entropy for similarity and one that includes spatial continuity. In their work, the third is achieved by a batch normalization layer. [Kanezaki and Tanaka] This was our starting point from which we were trying to optimize this approach for our satellite database to find obstacles on rooftops and eventually have a model that helps predict the available space on rooftops to build solar systems to combat global warming.

2) Change Normalization: First, we implemented the entire approach in Tensorflow, but since we plotted some outputs during the training, we realized pretty quickly that something was going wrong. The model returned an image with only one color Fig. 4. This means that the model

[2] https://www.tensorflow.org/api$_d ocs/python/tf/cast?hl = de$

[3] https://www.tensorflow.org/api$_d ocs/python/tf/data/Dataset$

optimized the first and second constraints very well in terms of the loss functions, but completely ignored the third constraint, which would lead to many unique labels. On closer inspection, we find that something is going wrong with the normalization, which is why we have moved from batch normalization to instance normalization, which leads to more unique classes. Not as many as 100 as in the study, but between 10 and 20. The reason for this could be that our images really look the same and only have a few different colors.

3) Skip connection: To make it easier for the model to generilize and to use local and global features we used similar like in ResNet a skip connection, which skips some layers to concatenate an early layer to a layer near the output layer. Therefore global features from an early layer can be passed directly to the end of the net, where they come together with very detailed information from the hidden layers.

4) Pretraining with roof masks: Because the model often labeled roofs same as the street, we decided to use roof masks, which were included in the metadata of the dataset. We created a function to extract the masks, to use them as targets in our Tensorflow dataset. We used the masks in two different ways. In a first scenario we decided to do pre-training with the masks, so that the model learns just to distinguish roofs from everything else. After some epochs of pre-training we switch to our previous loss function, which was derived from the previous work by Kanezaki and Tanaka. [Kanezaki and Tanaka]

5) Additional loss criteria with roof masks: In a second scenario we used the mask as an extension to the loss function given in [Kanezaki and Tanaka]. In this scenario our loss contained similarity loss, continuous loss and the new loss between the output and the mask of the roof. One problem of this approach is, that the masks are not perfect, therefore they often just segment the central roof and ignore surrounding roofs. This leads to wrong targets and not optimal learning. Therefore we lowered the stepsize of this third loss component, which leads to lower weight of the mask loss.

6) Max and average pooling: With average pooling, an approach not widely used anymore, the textures are more or less blurred after the convolution, collecting features relevant to any part of the image, while not extracting the most salient features like edges. In contract to that, we, because of that, used max pooling that does extract especially the edges. The downside of 'forgetting' some information was considered and tested previously with test runs of the network and deemed acceptable. We then pretrained the outer skipping connections - containing the max pooling layers - on images, so that these layers will be predispositioned to watch out for edges of the houses and objects. [Eli Stevens]

IV. Results

In the following section we describe the results we archived through different model ideas, starting with the nearest model to the origional paper [Kanezaki and Tanaka]. Then we go ahead and compare that baseline to different models, which we gradually adjusted according to our methods section, getting results after each major adjustment.

A big problem in unsupervised learning compared to supervised learning is, that you can't just compare your accuracy to compare different approaches. There is no accuracy, because you don't have a perfect target. So the only possibility is to look at the results and compare them.

A. Near origional

The following results in Figure 5 is based on the original methodology, while translated to Tensorflow. [Kanezaki and Tanaka] This archives good results with 50 pictures, while it does not improve much with each iteration. After 500 pictures is reduced the number of classes visible even more and does not detect the roof as well as in the beginning.

(a) 5 Epochs , each 10 pictures

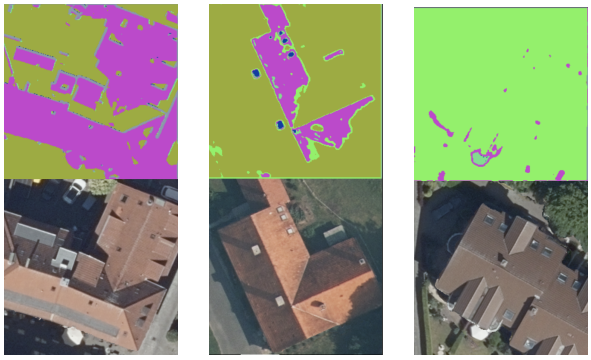
(b) 25 Epochs , each 10 pictures

(c) 50 Epochs , each 10 pictures

Fig. 5: Orginal loss, no skip connection and no pooling

B. Original with Pretraining and Skip Connection

While pretraining for our implementation of the original method improves the results after 50 pictures, it also makes the model not distinguish the edges as well as before. With continuing epochs, this model continues to get worse and tries to merge more and more classes into one object. The results decline much faster than in the original model. We suspect this is due to the skip connection making changes more visible in the last layers, as fewer layers are entangled in a cascade. Smaller obstacles are not detected as obstacles, rather also as part of the roof sometimes. Visible in Figure 6.

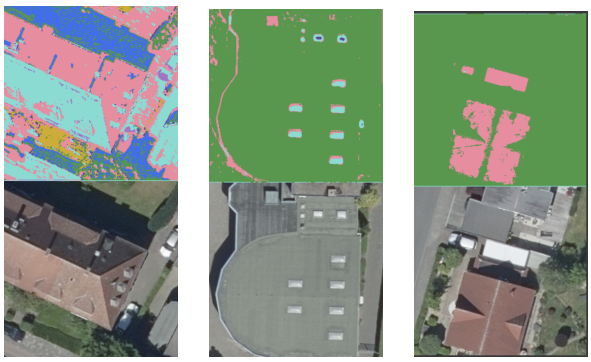


(a) 5 Epochs , each 10 pictures (b) 25 Epochs , each 10 pictures (c) 50 Epochs , each 10 pictures

Fig. 6: new loss with masks, skip connection and no pooling

C. Original with Pretraining

This model starts similar to the original with a fairly good segmentation, detecting objects on the roof as obstacles better than the original at first for the 50 image checkpoint. This could be due to the pretraining of the model and it having a better 'understanding' of the roof prior to the unsupervised segmentation part. As for 250 and 500 pictures, it is similar to the previous model, merging the roof and surrounding area, as well as obstacles into one class. Thus, it does not perform well at 250 and greater anymore. (Fig. 7)

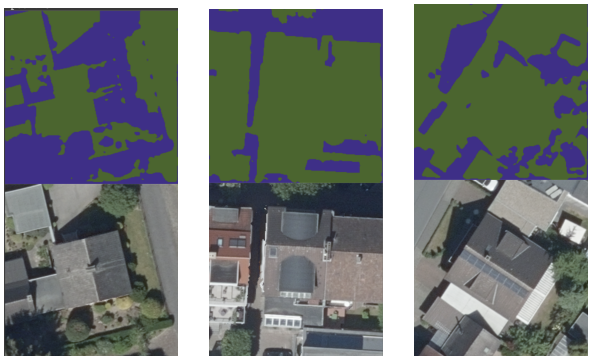


(a) 5 Epochs , each 10 pictures (b) 25 Epochs , each 10 pictures (c) 50 Epochs , each 10 pictures

Fig. 7: Pretraining on masks, no skip connection and no pooling

D. Original with Pretraining and Max Pooling

This model performs the worst for obstacle detection whilst performing the best at even 50 images for a general segmentation of the area into two categories. This is due to the max pooling reducing the detail level after each convolution layer and the absence of skip connection not allowing global features and local features to go 'different' routes. It does cluster shadowed areas of the roof together with green grass, as also seen in the third image at 500 (Fig. 8) images into the training loop. Although generally speaking, this model does not identify obstacles as well as the other ones and as already said confuses the roof often with the ground, not differentiating between ground, roof and obstacles. This is visible in Figure 8.



(a) 5 Epochs , each 10 pictures (b) 25 Epochs , each 10 pictures (c) 50 Epochs , each 10 pictures

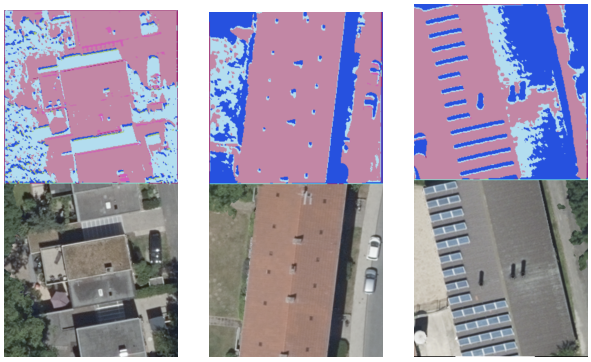
Fig. 8: new loss with masks, no skip connection and max pooling

E. Final model, data centric approach, skip connections and average pooling

To combine all these little improvements into one network, we decided to implement the skip connections with a average pooling to extract the more obvious features as edges. Then we used average pooling with multiple convolutions layers on the other path (left path in Figure 17, at the end of this paper) to extract local and fine details. Average pooling was smoothing textures, so that little imperfections in the texture of the roof was not weighted so heavily. Also, we pretrained the model on the roofs. While looking through the data, we found that masks of images not following the mentioned constraints are mostly not of high quality. Thus this time we disqualified masks and images of these masks for pretraining, if the masks covered less than 15% or more than 90% of the area of the image. The results are quite impressive (Fig. 9. The segmentation after 50 images yields good results, whilst not very different from the original model. On the other hand, looking on respective 250 and 500 images into training, the model can differentiate well between obstacles on the roof, the roof itself and the street.

F. Loss and best stopping point

As seen in the loss graph in Figure 10 the loss is going down slowly until Epoch 60, after which it drops significantly. In comparison, the number of Labels first drops until Epoch 50 as seen in Figure 11 , after which it recovers to approximately 9 again, and then drops again. This shows, that the best stopping point is not only defined by the loss as usually but rather especially also by the number of the labels combined with the loss. This is due to the nature of the loss being 'perfect' for the network at number of Labels



(a) 5 Epochs , each 10 pictures (b) 25 Epochs , each 10 pictures (c) 50 Epochs , each 10 pictures

Fig. 9: Final model - data centric approach - just images with sufficient roof size, new loss with masks, skip connection and average pooling

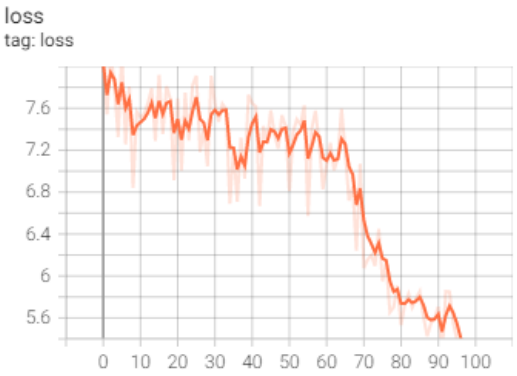


Fig. 10: Loss during training on final model

equal to one. Thus the ideal stopping point is where the loss is the lowest whilst the number of Labels at the same Epoch number is above a defined threshhold.

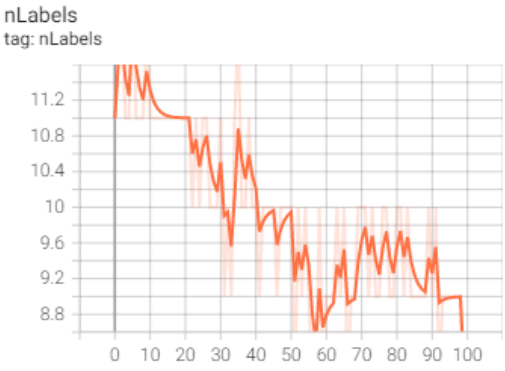
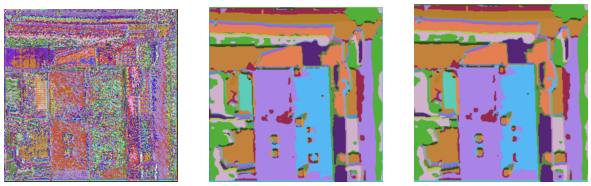


Fig. 11: Number of unique labels during training on final model

G. Compare results to pytorch model from the initial paper

Fig. 12 shows the model of [Kanezaki and Tanaka] on our satellite image dataset. A clear difference is the high number of unique labels. This constraint is due to the normalization layers. Batch normalization is used in [Kanezaki and Tanaka], which did not work for us, as shown in Fig. 4. With more time this could be investigated further.



(a) After 1 image (b) After 50 images (c) After 100 images

Fig. 12: Our dataset on the pytorch model from the initial paper

H. Compare results to grayscale segmentation

In Figure 13 we just build a simple grayscale filter to segment images. This is one of the most simplest approaches to cluster an images into different clusters. You can already recognize the shape of the roof and some obstacles. At the same time really small clusters can be build. Of course a hard threshold is no comparison to the other models, as it can only distinguish between two 'classes'. At least it could be used to detect roof-like objects and clear fields in the image.

I. Compare results to K-Means

Fig. 14 shows the results of K-Means with different k on one image. The advantage here is a very simple and fast training process. The result for k=10 looks very good as it classifies similar colors into the same clusters. The

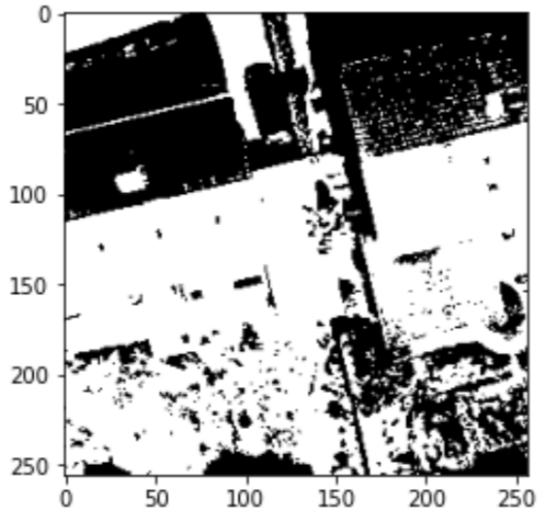


Fig. 13: Compare to simple grayscaling

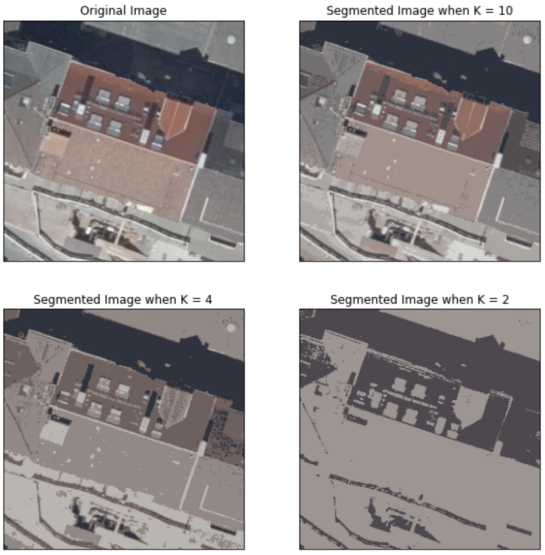


Fig. 14: Compare to K-Means for k=10,4 and 2

advantage of our model over K-Means is that we include spatially continuous pixels and local and global features and our model itself decides how many unique labels an image should have. K-Means is more limited here, since it has to find as many clusters as specified by k, even though this might be nonsense. However, sometimes this can also be an benefit, because you can simply manual try different ks for the same image.

J. Compare results to supervised approach

The main motivation behind our unsupervised approach was to have a model, which does not need thousands of labels. We investigated in this notebook [4] what results are possible with just a few (14) labeled images combined with a pre-trained state of the art segmentation model. With less than four minutes of training (and just 14 segmented images!) it lead to outstanding results like shown in 15. Of course not all test images were as outstanding as this one, but this one shows its potential.
We published the dataset and the annotations.json [5], so that other researcher can save time.

V. Further research

One downside of our approach is: the longer the training process is going on and the more the loss goes down, the fewer unique labels are generated. This means that in the beginning the training can be really good, but with time just two classes are found, where the loss is minimized. Therefore one can investigate a way to weight a higher number of labels more. Furthermore a semi supervised approach can be investigated, where just a few labels or scribbles like in

[4]https://github.com/michael-ra/SSIS$_{R}oof_{D}iffFeatClust/blob/main/Supervised_{C}omparison_{D}etectron2.ipynb$

[5]https://github.com/michael-ra/SSIS$_{R}oof_{D}iffFeatClust$

Fig. 15: Test results from detectron2 with 14 training images after 4 minuetes of training

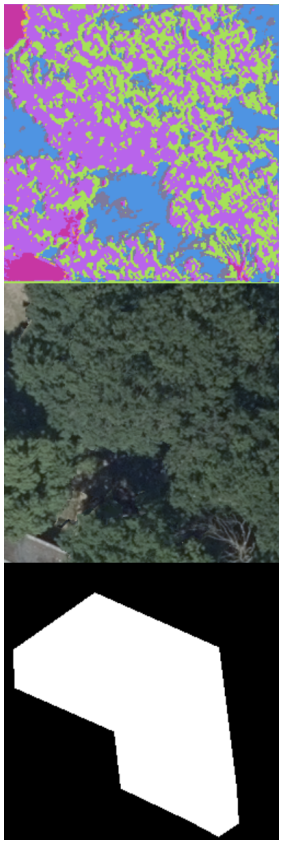


Fig. 16: Outlier problems we faced: Masks didn't match image

[Kanezaki and Tanaka] are needed. Additionally one can investigate ways to train a model, which can even more separate the streets from the roofs, because here they are often labeled in the same class. Also last but not least, the data quality or cleanup process for the supervised part has to be improved, as there were still some pictures remaining without a roof visible, although the mask made the model believe something else as seen in Figure 16

VI. Conclusion

In the study project, the aim was to construct a good unsupervised or semi-supervised model for image segmentation to segment roofs from obstacles on roofs and roofs from their surroundings, enabling better and faster image segmentation by humans. This promise was fulfilled by our final model. It distinguished fairly well between these classes whilst keeping obstacles on the roof separate from the roof itself. These clusters can now be shown to human workers, to fine-tune the segmentation already done by the network. Also, we can detect, if there is a roof on a picture or not, and if so, where it is. This is beneficial for satellite images in cities, where there is no clear designated data available for distinguishing house roofs from grass, street and other obstacles. Further, with the segmented images, the obstacles on the roof can be analyzed for solar panel like objects and free space, detecting, weather there is place for solar panels on a roof or if there are already solar panels on the roof.

Acknowledgment

We would like to thank Leon Schmidt for their support and feedback during the whole project. We are grateful they offered us the opportunity and guidance to implement our project during their engaging seminar "Implementing Artificial Neural Networks in Tensorflow" in the winter semester 2021. Finally, thanks to the county Northrhine Westfalia for providing us the needed satellite images.

Appendix

Our Code can be found under the following github repository: https://github.com/michael-ra/SSIS_Roof_DiffFeatClust

Final Presentation can be viewed under this link: https://ramich.me/ianntfpresentation

References

Bezirksregierung-Köln. Geodatendienste. *Geobasis NRW*, 2022. https://www.bezreg-koeln.nrw.de/brk$_i nternet/geobasis/webdienste/geodatendienste/$.

M. Campbell. More solar panels need to be made in europe to cut dependency on russian gas, says eu. *Euronews*, 2022. https://www.euronews.com/green/2022/03/31/more-solar-panels-need-to-be-made-in-europe-to-cut-dependency-on-russian-gas-says-eu.

T. V. Eli Stevens, Luca Antiga. Build, train, and tune neural networks using python tools. *Deep Learning with PyTorch*, 1 (1).

A. Kanezaki. Unsupervised image segmentation by backpropagation. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP*. IEEE.

W. K. A. Kanezaki and M. Tanaka. Unsupervised learning of image segmentation based on differentiable feature clustering. *IEEE Transactions on Image Processing*, 29:8055–8068.

N. Kannan and D. Vakeesan. Solar energy for future world: - a review. *Renewable and Sustainable Energy Reviews*, 62: 1092–1105, 2016. ISSN 1364-0321. doi: https://doi.org/10.1016/j.rser.2016.05.022. URL https://www.sciencedirect.com/science/article/pii/S1364032116301320.

W. Kim, A. Kanezaki, and M. Tanaka. Unsupervised learning of image segmentation based on differentiable feature clustering. *CoRR*, abs/2007.09990, 2020. URL https://arxiv.org/abs/2007.09990.

M. Motamedi, N. Sakharnykh, and T. Kaldewey. A data-centric approach for training deep neural networks with less data. *CoRR*, abs/2110.03613, 2021. URL https://arxiv.org/abs/2110.03613.

H. Ritchie and M. Roser. Energy. *Our World in Data*, 2020. https://ourworldindata.org/energy.

Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019.

X. Xia and B. Kulis. W-net: A deep model for fully unsupervised image segmentation. arXiv preprint arXiv:1711.08506,.

W. Yang, J. Cai, J. Zheng, and J. Luo. User-friendly interactive image segmentation through unified combinatorial user inputs. *IEEE Transactions on Image Pro- cessing*, 19(9).

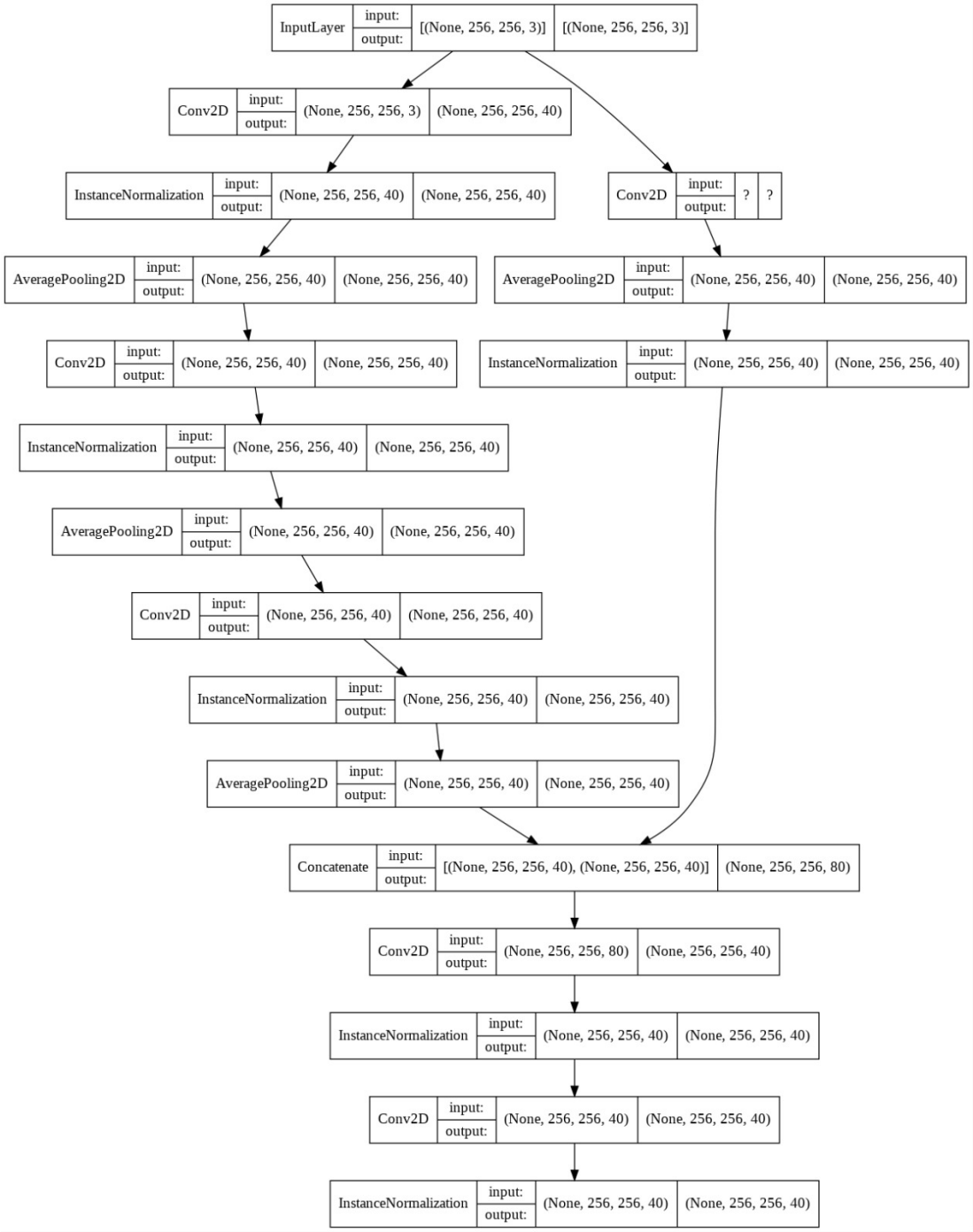


Fig. 17: Our final model