Technical University of Munich
School of Engineering and Design
Chair of Geoinformatics & Institute of Automotive Technology

Study project report in the program
Environmental Engineering M. Sc.

Improving CNN roof segment detection by dataset extension using 3D city models

Florian Faltermeier, B. Sc.

Supervisors:
Sebastian Krapf, M. Sc.
Bruno Willenborg, M. Sc.

Munich, April 2022

Abstract

Estimating the solar potential of roof areas suitable for photovoltaic panels can be aided by artificial neural networks identifying roof segments and their orientation in remote sensing imagery. A primary factor limiting the performance of such neural networks is the availability of training data, which is commonly generated in a time-consuming manual labeling process. In this project, a method was developed that uses roof positional information from existing semantic 3D city models to generate datasets of orthophoto crops of buildings with roof segment labels. The method was applied to create a large dataset of more than 120 000 samples from buildings in urban and rural regions in Bavaria. A convolutional neural network (CNN) was trained on this and four other, smaller datasets, including one consisting of more than 1800 manually labeled samples of a Bavarian town created from Google satellite imagery. This allowed a detailed performance comparison with respect to source, quality and amount of the data, and identification of benefits and drawbacks of training data derived from 3D city models. Overall, the results indicate a clear improvement in segmentation performance when the CNN is trained on the extended dataset. However, the CNNs also showed a considerable degree of specialization on the data source they were trained on: While the extended dataset did help the CNN's ability to generalize to images from a different data source (Google satellite imagery) to a limited extent, the strongest improvements were seen with aerial images from the same source. This and other findings discussed in this report offer potential for further investigations.

Contents

List of Figures

List of Tables

1 Introduction

Photovoltaic (PV) energy plays an important role in global efforts to move beyond the fossil era and towards a future of sustainable energy production. The requirement of large surface areas for PV module deployment is a key characteristic of sunlight as renewable energy source. Roof areas constitute one pool of potentially suitable surfaces that is not yet fully exploited: In the European Union, up to 24.4 % of the electricity demand could be provided by rooftop PV systems (Bódis et al., 2019). Development of inexpensive and widely applicable approaches to estimate roof solar potential facilitates further expansion of rooftop PV.

Work by Melius et al. (2013) and Freitas et al. (2015) gives a comprehensive overview of existing approaches that use solar radiation algorithms combined with a variety of geospatial analysis methods based for instance on Light Detection and Ranging (LiDAR) data, monocular or stereo remote sensing imagery, or cadastre data. Generally, solar potential is assessed in a hierarchical manner, where physical, geographic, technical, economic, and even a social potential can be distinguished (Izquierdo et al., 2008). Novel approaches make use of artificial neural networks to estimate roof solar potential from aerial imagery (Lee et al., 2019; Castello et al., 2021; Krapf et al., 2021). This has the main advantage over LiDAR-based approaches that the required data are less expensive in acquisition and more widely available (Xu et al., 2018), allowing cost-efficient assessment even in remote and less well surveyed areas. In these applications, the neural network is usually responsible for the step that requires identification of outline, orientation and potentially also slope of roof segments in two-dimensional remote sensing imagery. This task is an area of research in itself: Zhao et al. (2022) trained a graph neural network for this purpose, Muftah et al. (2021) use a sequence of models performing a classification and a semantic segmentation step, followed by further processing, with the aim to derive 3D building models. Related tasks include the detection of roof areas (e. g., Qin et al., 2019; Collier et al., 2021) and the classification of roof types from the images. Other projects use similar approaches to detect solar panels from remote sensing imagery and automatically create databases of existing solar deployments (Yu et al., 2018; Castello et al., 2019; Hoog et al., 2020; Mayer et al., 2020; Rausch et al., 2020; Costa et al., 2021).

Willenborg et al. (2018) have developed a tool that estimates roof solar potential based on semantic 3D city models following the CityGML specification (Kolbe, 2009). Kemmerzell (2020) and Krapf et al. (2021), on the other hand, designed and trained two convolutional neural networks (CNNs) that use semantic segmentation to identify roof segments and superstructures, respectively, and used the results in their method to determine economic PV potential. A factor limiting the improvement of a neural network's ability to identify roof segments is the availability of training data, which is usually generated manually in a time-consuming annotation process. Prior to this project, the CNN was trained using a dataset of 444 training samples that was also used by Lee et al. (2019). While the network delivered reasonably good results, Kemmerzell (2020) noted several shortcomings with respect to the dataset. It is quite

homogeneous in several aspects: It covers a limited variety of roof types, mainly single house roofs and larger flat roofs. The contained images all come from the same geographic region and are uniform in exposure, season, shadow length, and general quality. As a result, the trained network performs subpar in situations where roofs appearing in the imagery deviate in their characteristics from the roofs present in the training dataset, or if the light conditions and image quality are different. A possible solution to this problem would be an extension of the training dataset.

In an attempt to leverage synergies between the methods used by Willenborg et al. (2018) and Krapf et al. (2021), the present study project has two goals: First, to investigate whether and how roof positional information from semantic 3D city models can be used to derive training data for a neural network that identifies roof segments in aerial images. This would enable the generation of a vastly extended dataset based on available 3D city data, thereby avoiding the extensive manual labeling process. Special attention must be paid to evaluation of quality and consistency of training samples created this way. Second, the developed method shall be used to automatically generate datasets of training samples for several geographic scenarios, then train CNNs for each scenario and compare their performance both among each other and to a CNN that was trained only on a smaller, manually labeled dataset. A research questions is posed as follows: Does the ability of a given convolutional neural network to identify roof segments improve when the training dataset is extended with samples that were generated automatically from aerial imagery and roof positional information from semantic 3D city models? It can be subdivided into the following, falsifiable hypotheses:

1. Roof positional information from 3D city models can be extracted and used to label the corresponding roof areas in aerial or satellite images.
2. The automatically generated labels are accurate enough in outlining the roofs to allow effective training of the neural network.
3. Training the neural network with the extended and less homogeneous dataset delivers superior performance compared to the original hand-labeled dataset.

Following this introduction, at first a short overview of the theoretical background is given. Then, the data and methods that were used are introduced and the results are presented and discussed.

2 Theoretical background

2.1 CityGML

Kolbe (2009) introduces CityGML as a data modeling standard for semantic 3D city and landscape models. It represents spatial objects on five different levels of detail (LoD0 to LoD4) and comprises information on their geometry, semantics, topology, and appearance. The family of ISO 19100 standards (ISO, 2015) provides the data model for the semantic information, whereas Unified Modeling Language (UML) is used to define object oriented models of spatial features and thus enables hierarchical structures of generalization and aggregation. All city objects are derived from the parent class *_CityObject*, where the underscore indicates that it is an abstract class, which again is a subclass of the basic GML class *_Feature*. All instances of *_CityObject* then constitute the *CityModel* of class *FeatureCollection*. Further, the city objects are subdivided into several thematic categories such as *Building*, *CityFurniture*, *Transportation*, *Vegetation*, or *WaterBody*. The geometries of city objects are described by boundary representation (B-rep), where solids are defined by an aggregation of their bounding surfaces, as opposed to a representation by volumetric primitives as applied in constructive solid geometry (CSG).

The feature class *_AbstractBuilding* and its relations with other classes are of particular interest for this project because its constituent boundary surfaces of class *_BoundarySurface* include those of class *RoofSurface*, which were utilized to extract information on the geographic position and extent of roof segments. Figure 1 shows a UML diagram of *_AbstractBuilding* and some of its relations to other classes. A more comprehensive representation can be found in the OGC CityGML Encoding Standard (Open Geospatial Consortium, 2012).

2.2 3D City Database

Storage of and access to 3D building data in this project was handled using the CityGML standard in an implementation as 3D City Database (3DCityDB) in PostGIS. 3DCityDB is an open-source implementation of the CityGML schema for use with spatially enhanced relational database management systems (SRDBMS), in this case PostgreSQL with PostGIS extension, and ORACLE Spatial (Yao et al., 2018). It also comprises several software tools that enable data import, export, analysis and visualization in accordance with CityGML. Within the scope of this project, a PostGIS-based 3DCityDB was used in a Docker environment with the software DBeaver as administration tool.

2.3 Convolutional neural networks

Convolutional neural networks (CNNs) are a category of artificial neural networks for deep learning that uses convolutional layers (Lecun et al., 1998). Generally, an artificial neural

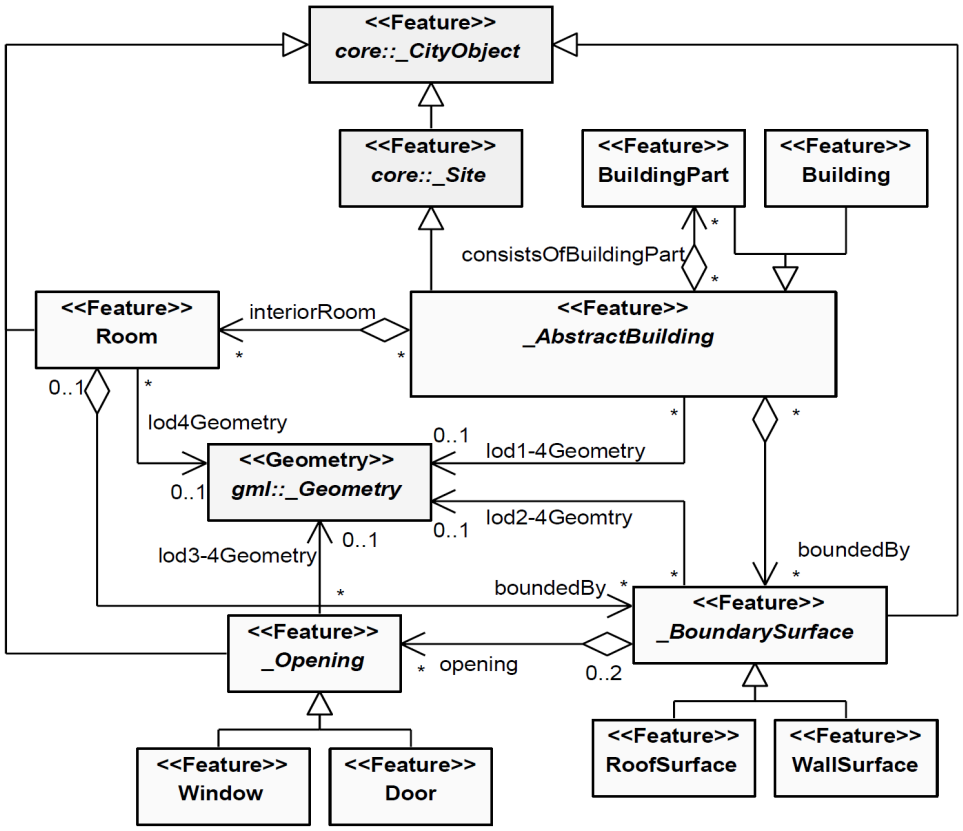


Figure 1: UML diagram of the CityGML feature class *_AbstractBuilding* and some of its dependencies. From Kolbe (2009).

network for deep learning is comprised of a series of layers where each linearly manipulates an input x with weights W and a biases b and then subjects it to a non-linear activation function before passing the output on to the next layer (Goodfellow et al., 2016). By choosing appropriate dimensions for the weight matrix, the output dimension of each layer can be controlled. Thereby, the output of the final layer can be mapped to a problem of interest, for instance, a classification problem with a certain number of classes. The introduction of a distance measure or loss function that quantifies the deviation of the delivered outputs from the desired outputs enables the optimization of the network's parameters (weights and biases) with respect to the loss and, thus, minimize it. This is called training or learning and will, given a sufficient number of iterations, appropriate network architecture, optimization technique, and input training data, allow the network to generalize from the training data to other, previously unseen data to solve the problem under consideration.

In a fully connected layer, for an input x of dimension $(m, 1)$ and a desired output y of dimension $(n, 1)$, a weight matrix W of dimension (n, m) and a bias b of dimension $(n, 1)$ are required. Further, a scalar, non-linear activation function $f(x)$ is used:

$$y = f(Wx + b) \tag{2.1}$$

Accordingly, every single input value is assigned n weights for the n output values it contributes to. This can be problematic in certain applications for several reasons: For large inputs (e. g., high resolution images) and deep networks with many layers, the number of trainable parameters and the associated memory requirement and computational effort can become exceedingly large. Moreover, such a network can be prone to over-fitting because its capacity allows it to memorize the input. One solution for these problems is the use of convolutional layers.

As opposed to the matrix multiplication performed in a fully connected layer, the convolution operation uses a smaller set of weights, referred to as the kernel, to process the entire input. Each output value is a weighted average of several neighboring input values. Following Goodfellow et al. (2016), the two-dimensional discrete convolution can be expressed as:

$$O(x, y) = (K * I)(x, y) = \sum_{dx} \sum_{dy} K(dx, dy) I(x - dx, y - dy) \tag{2.2}$$

Here, I is the input, K is the convolution kernel, and O is the output. Figure 2 demonstrates this graphically. In practice, this corresponds to the application of an image filter that has learnable weights to an image. For RGB images with three color channels, in fact, a three-dimensional convolution is computed. Compared to fully connected layers, the number of parameters is significantly reduced.

CNN architectures have proven to be particularly successful in computer vision tasks, such as image classification or semantic segmentation (Schmidhuber, 2015). Apart from convolution layers they also utilize other layer types such as pooling and upsampling layers.

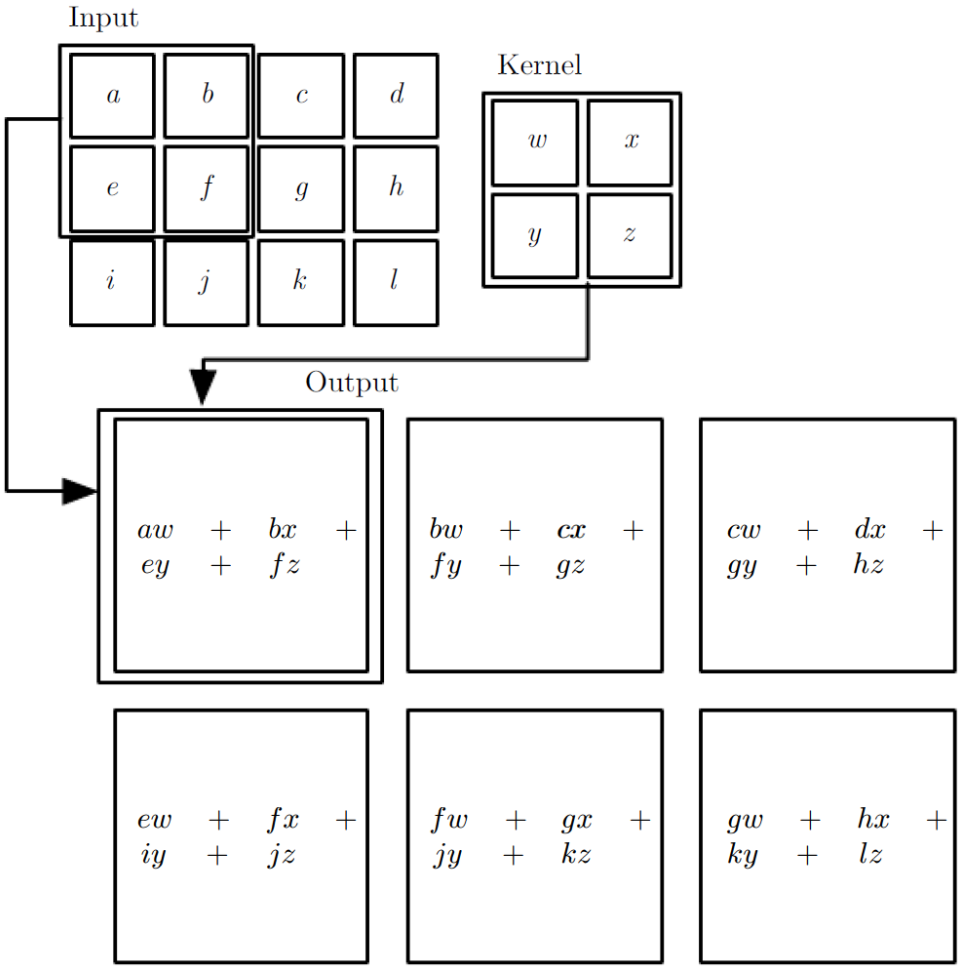


Figure 2: Basic principle of a two-dimensional convolution of an input with a kernel. Note that the output has smaller dimensions compared to the input. This can be avoided by padding the input with additional values. From Goodfellow et al. (2016).

3 Data and methods

3.1 Generation and structure of datasets

3.1.1 Data sources

The automatically labeled dataset of roof segments was derived from 3D city data available at level of detail 2 (LoD2). The manually labeled dataset was created using Google satellite imagery that was retrieved by means of the Google Maps Static API. Because these images are not orthorectified, the 3D city data that was to be used in the automated labeling process would not exactly match the roof segments in the Google data. For this reason, orthorectified aerial imagery from the Bavarian Surveying and Mapping Authority (BVV, Bayerische Vermessungsverwaltung) was used that aligns well with the 3D city data from the same source. These orthophoto crops were obtained from a Web Map Service provided by the BVV.

3.1.2 Selection of study areas and scenarios for comparison

In this section and in table 1, all training scenarios used for comparison of neural network performance are described. A dataset of manually labeled roof segments was available from previous investigations: It comprises roof-centered crops of Google satellite images and corresponding label files for 1878 buildings with 4473 roof segments in the town of Wartenberg in Upper Bavaria. This dataset served as the baseline for all comparisons.

For a first comparison scenario, an automatically labeled dataset of the same town was to be created. This would allow identification of differences in the underlying data sources and their quality, and their effects on the neural network's ability to determine roof segments. Since the 3D city data is structured in a spatial grid, four cells that cover the town of Wartenberg were selected. These contain 2835 buildings with 6071 roof segments. The higher number of buildings and roof segments compared to the manually labeled set mainly stems from their different definition in the 3D city model with a finer subdivision. The differences in these numbers compared to the manual dataset were explored in further detail and are explained in section 4.1.2. To control for the number of training samples and its effect on the neural network performance, in a second scenario, another automatically labeled dataset was generated, this time centering the training samples at the building centroids from the manual dataset, resulting in an equal number of buildings (and thus, training samples) as for the manual dataset.

For the third comparison scenario, areas from different parts of southern Bavaria were selected with the aim to maximize the diversity of building types present in the data. The Regional Statistical Spatial Typology for Mobility and Transport Research (RegioStaR) by the German Federal Ministry of Transport and Digital Infrastructure (BMVI, Bundesministerium für Verkehr und digitale Infrastruktur) was used for this purpose, and in particular, the combined regional statistical spatial type RegioStaR 17 (BMVI, 2018). Areas with a total of 123 050

Table 1: Training scenarios and corresponding numbers of buildings.

ID	Scenario name	Abbreviation	Number of buildings
0	Wartenberg manual	*wb_m*	1878
1	Wartenberg automatic	*wb_a*	2835
2	Wartenberg automatic (manual centroids)	*wb_a@m*	1878
3	Bavaria automatic	*bv_a*	125 885
4	Wartenberg manual + automatic	*wb_m+a*	4713

buildings were selected that are distributed as follows: 58 580 buildings from central Munich (types 111 metropolis and 112 large city from regional type 11 metropolitan urban region), 30 808 buildings from the regional towns of Erding and Freising (type 113 medium-sized city from regional type 11 metropolitan urban region), and 33 662 buildings from a large rural area southwest of Munich (types 225 small-town area, village area and 224 urban area from regional type 22 peripheral rural region). The training samples from the automatically labeled Wartenberg dataset were added, totaling 125 885 buildings in this scenario.

In a fourth and final comparison scenario, the two datasets of Wartenberg (manually and automatically labeled) were combined. Since these two were generated based on different data sources (see section 3.1.1), their imagery differs in qualitative aspects and their labels, too, may exhibit systematic differences. It is the aim of this scenario to examine to what extent a neural network trained only on one of these two types of datasets specializes on them or is able to generalize to the other type of dataset, compared to a network that was trained using data from both generation procedures. Figure 3 shows buildings and areas of the Bavaria dataset. Appendix B also contains similar plots for the other training scenarios.

3.1.3 Processing and extraction of 3D city data

All 3D city data were available in GML format following the CityGML specification and, at first, imported into a 3DCityDB instance of a PostgreSQL database running the PostGIS extension. Then, several processing steps were performed on the database: First, all roof segments contained in the respective dataset were obtained by joining the tables *building*, *thematic_surface*, and *surface_geometry*. For each building, the corresponding thematic surfaces are stored in the table *thematic_surface*, which again references the concrete surface geometries in the table *surface_geometry*. Within the table *thematic_surface*, roof surfaces are identified by the attribute *objectclass_id* assuming the value 33. Second, for each roof segment, its azimuth (orientation in the horizontal plane) and slope were computed in degrees based on its corresponding surface geometry's normal vector. Finally, the required attributes for all segments were exported as comma-separated-value (CSV) file for further processing in Python. Table 2 lists and describes these attributes. Listing 1 shows the SQL query that was used to obtain the described information.

Listing 1: SQL query used to obtain roof segment data from 3DCityDB.

```
select
  b.id as "b_id",
  sg.id as "sg_id",
  degrees(
    st_azimuth(
      st_makepoint(0, 0),
      st_rotate(
        st_force2d(citydb.normalvector_norm(sg.geometry)),
        pi()
      )
    )
  ) as "azimuth",
  90 + degrees(
    citydb.slope_from_normv(citydb.normalvector_norm(sg.geometry))
  ) as "slope",
  st_force2d(sg.geometry) as "geometry"
from building b, thematic_surface ts, surface_geometry sg
where
  b.id = ts.building_id and
  ts.objectclass_id = 33 and
  sg.root_id = ts.lod2_multi_surface_id and
  sg.geometry is not null;
```

Table 2: Attributes included in database CSV export of roof segments required for further processing.

Attribute name	Description	Source of information
b_id	Building ID (identical for all segments of the same building)	Given in source 3D city data
sg_id	Surface geometry / roof segment ID (unique)	Given in source 3D city data
azimuth	Horizontal orientation of the roof segment in degrees	Derived from surface geometry's normal vector
slope	Vertical orientation of the roof segment in degrees	Derived from surface geometry's normal vector
geometry	Geometry of the roof segment as well-known text (WKT)	Given in source 3D city data

3.1.4 Generation of training samples

Generation of training samples from orthophotos and 3D city data. The table of roof segments (structure see table 2) was processed further in Python. Depending on their azimuth and slope, roof segments were assigned to one of 17 classes: Sloped roofs were categorized into 16 orientation classes, subdividing the 360° range into 22.5° slices (N, NNE, NE, ENE, E, ESE, SE, SSE, S, SSW, SW, WSW, W, WNW, NW, and NNW), and flat roofs into another separate class. All other areas were labeled as background, amounting to a total of 18 different semantic classes (see table 3).

A single sample for training of the neural network in the present application was to consist of a building-centered 3-channel color image showing the roof and a single-channel image containing the labels, with pixel values between 0 and 17 corresponding to the 18 semantic classes. For each building in a scenario's dataset, a 256 · 256 px orthophoto-crop was retrieved from a Web Map Service (see section 3.1.1) centered to the building centroid. The geographical extent of this crop was then used to rasterize the corresponding roof segment polygons, assign the respective class values, and store them as label image file. The BVV orthophotos were available at a ground resolution of 0.2 m px^{-1}.

Table 3: Semantic classes of training samples: Background, 16 orientation classes with azimuth ranges, 1 flat roof class.

Class name	ID	Azimuth range [°]	
		Min	Max
Background	0	–	–
N	1	−11.25	11.25
NNE	2	11.25	33.75
NE	3	33.75	56.25
ENE	4	56.25	78.75
E	5	78.75	101.25
ESE	6	101.25	123.75
SE	7	123.75	146.25
SSE	8	146.25	168.75
S	9	168.75	191.25
SSW	10	191.25	213.75
SW	11	213.75	236.25
WSW	12	236.25	258.75
W	13	258.75	281.25
WNW	14	281.25	303.75
NW	15	303.75	326.25
NNW	16	326.25	348.75
Flat	17	–	–

Adaptation of training samples from manually labeled dataset. The manually annotated training samples of Wartenberg were originally structured slightly differently and therefore adapted to match the structure of the automatically generated data. First, the semantic labels were created using the same classification, but the pixel-level class values were different and had to be mapped to the class values used here. Second, the manual Wartenberg samples were originally generated at a resolution of $512 \cdot 512$ px. To match both pixel and ground resolution of the BVV orthophotos used with 3D-city-data-derived labels, each training sample in the hand-labeled dataset was scaled down to $256 \cdot 256$ px, which corresponds to a ground resolution of about 52.7m or $0.2\,\mathrm{m\,px^{-1}}$. Label images were resized using nearest neighbor interpolation to ensure conservation of pixel values.

3.1.5 Execution of data split

For each training scenario, the corresponding dataset was split into three subsets: One for neural network training, one for validation during training, and one for testing, i. e., performance evaluation of the final model. Working with geospatial data, several considerations had to be made with respect to the approach to split the data. It was central to these considerations that the training samples were generated in a building-centered manner, showing buildings in the image center, as opposed to a grid-based dataset.

This decision was made for various reasons: It was thought to best reflect the real-world application of the neural network model in identification of roof segments of concrete buildings where they would appear in the image center. Further, it would ensure that most buildings could be depicted fully within the geographic bounds of their training sample, whereas in a grid-based dataset, the likelihood of buildings cut off at the image edges is, on average, higher. However, it follows from this approach that each building, or parts of it, can appear in several training samples in the frequent case that a sample's extent includes neighboring buildings for which another sample is generated. This is relevant for the data split because the validation and test sets should not contain samples or even parts thereof from the training set.

Therefore, a geographic data split was performed as follows: Within the data region, several positions were chosen for both the test and validation set along with a specific number of buildings to be selected around each of these positions. First, around each test position, this pre-defined number of buildings was identified within a circular area whose radius was determined iteratively. These buildings were then subtracted from the main dataset. To ensure that the datasets are entirely disjoint, i. e., that no parts of buildings contained fully or partially in the test set are depicted in any of the other sets, all buildings intersecting the extent of the samples selected for the test set were also subtracted from the main dataset. Based on the remaining buildings, the identical procedure was then repeated for the chosen validation positions to find the buildings for the validation set. After that, all remaining build-

ings comprised the training set. Because of the approach that was used to ensure that the datasets are disjoint, the number of training samples is not equal to the difference between the total number of samples and the number of validation and test samples.

Table 4 shows numerical information on the data splits as they were applied for each of the different scenarios. For the Wartenberg scenarios (IDs 0, 1, 2, and 4), three positions were selected for both the validation and test sets. Sizes of validation and test set were chosen to be about 10 % of the total number of samples. As described in section 3.1.2, the Bavaria dataset (ID 3) consists of data from several regions. A total of 11 positions for each validation and test set were selected and distributed across these regions: 4 in the rural area southwest of Munich, 3 in the central Munich area, and 2 each in Erding and Freising. For each region, the number of validation and test set samples were chosen to be about 5 % of the respective region's total number of samples. Table 5 gives an overview of this. Figure 3 gives a graphical representation of the data split with training, validation, and test set areas for the Bavaria dataset. The same representation for the other training scenarios and close-ups of the Bavaria dataset subregions can be found in appendix B.

Table 4: Data splits as performed for each of the scenarios. Numbers of samples in training, validation and test sets, and number of geographical positions from which validation and test samples were selected.

Scenario		Number of samples				Number of positions	
ID	Abbreviation	Total	Training	Validation	Test	Validation	Test
0	*wb_m*	1878	1364	180	180	3	3
1	*wb_a*	2835	2085	270	270	3	3
2	*wb_a@m*	1878	1364	180	180	3	3
3	*bv_a*	125885	111462	6270	6270	11	11
4	*wb_m+a*	4713	3449	450	450	3	3

Table 5: Distribution of validation and test set samples across the regions in the Bavaria dataset. (Excluding the samples from *wb_a* that were added for training.)

Bavaria dataset region	Number of samples					Positions per set
	Total (relative)		Test	Validation	Per position	
Rural Bavaria	33662	(0.27)	1620	1620	540	3
Munich	58580	(0.48)	2880	2880	720	4
Erding, Freising	30808	(0.25)	1500	1500	375	4
All	123050	(1.00)	6000	6000		

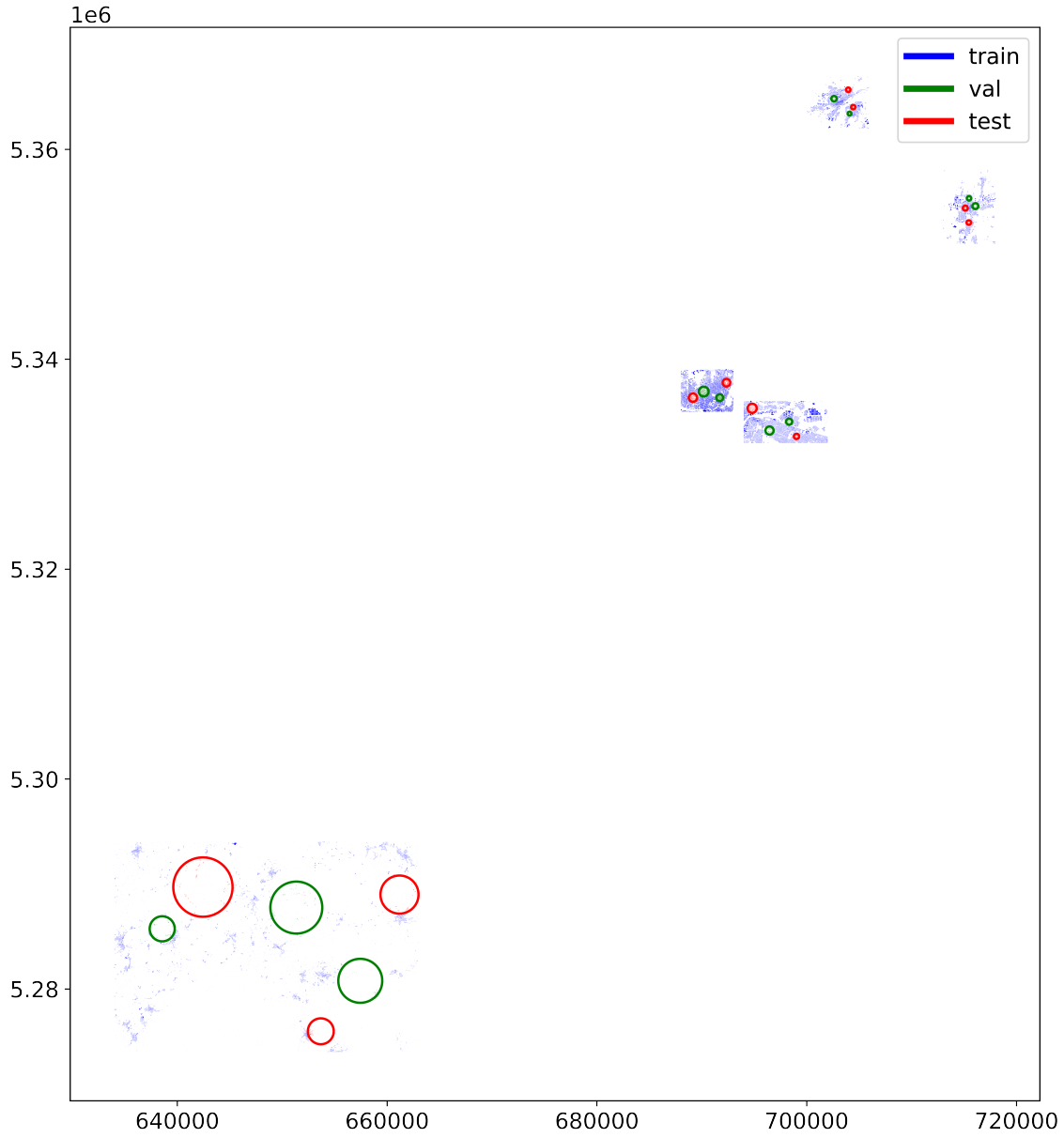


Figure 3: Data split of the Bavaria dataset with areas containg training (train), validation (val), and test data. Visible areas: Freising in the very north, Erding in the very east, two areas in Munich centrally, sparsely populated rural area in the south-west.

3.2 Semantic segmentation of roof segments

3.2.1 Neural network architecture and hyperparameters

For each scenario, a convolutional neural network (CNN) was trained for 40 epochs (20 epochs in the case of the Bavaria dataset). An extensive process of manual hyperparameter tuning was conducted to optimize the training results (see appendix A for details). It lead to the following settings for the final training runs: A U-Net architecture (Ronneberger et al., 2015) with a ResNet-152 backbone (He et al., 2016) consistently delivered the best results. Several loss functions designed to handle highly imbalanced class distributions (e. g., in the automatically labeled Wartenberg dataset, 79 % of all pixels belong to the background class) were explored (Jadon, 2020; Sugino et al., 2021), and the categorical focal loss was found to deliver the best results. Due to hardware limitations and the depth of the network, batch size could only be set as high as 8 samples. Learning rate was set to 10^{-4} and decreased by a factor of 10 whenever a plateau was reached during training.

3.2.2 Data augmentation

To decrease overfitting and improve the network's ability to generalize, the training data was augmented in several ways during training. Randomly applied augmentations methods were: Resizing and shifting, addition of Gaussian noise, change of brightness, contrast, saturation, and gamma value, sharpening, and blurring.

Further, with a probability of 0.5, training samples were subdivided into a regular 2-by-2 or 3-by-3 grid, and the grid cells then were shuffled. Although this can disrupt the topological structure of the depicted roofs, in first experiments it appeared to improve the model's performance in terms of IoU on the respective test set, indicating that the benefits of more complexly structured training data on model performance outweigh any potential drawbacks stemming from disordered roof topology. However, further exploration of this augmentation method (three training runs with identical settings each with and without the method) did not yield conclusive results as to whether it actually helps to improve performance. Because, at that point, all models had already been trained with this augmentation method enabled and at least no deterioration in performance could be detected due to it, it was decided to continue working with the obtained models. Also, even though the mentioned investigation did not indicate an improvement on average, it is noteworthy that each training run was performed twice, once with and once without this augmentation method, and in every case the model trained with this method performed better than the other (in terms of the employed performance measure as described in the following).

3.2.3 Neural network performance evaluation

For final evaluation of model performance, intersection over union (IoU) was used as a measure of comparison, which is defined as the ratio of true positives to the sum of true positives (TP), false positives (FP), and false negatives (FN). Several distinctions can be made as to how IoU is computed from raw neural network output, i. e., results of the final softmax function.

First, IoU can be derived either directly from the softmax values or after application of an argmax function that interprets the softmax outputs as probabilites and sets the value of the most likely class to 1 and all other classes to 0 for each pixel. A third possibility is to not simply assign the class with the highest probability to a pixel, but only to do so if a threshold probability (e. g., 0.5) is exceeded, assigning no class if this is not the case.

Second, there is a distinction between IoU on dataset level and on image level. On the image level, IoU can be computed either by taking the average of all class IoU values in the image (”macro”) or by summing up TP, FP, and FN numbers over all classes and therefrom deriving total image IoU (”micro”). A similar distinction applies for the dataset level IoU: It can be computed either by taking the average of all image IoU values (”macro”) or by summing up TP, FP, and FN numbers over all images and only then computing overall IoU (”micro”). In the latter case, the summing up of TP, FP, and FN may be done class-wise (image-level macro) or without distinguishing classes (image-level micro). This leads to a total of four ways to compute overall IoU, depending on which approach is selected for the image and dataset level. They are here referred to as, e. g., ”micro/macro” (image-level approach/dataset-level approach). Three further distinctions must be made:

1. In the macro case on the image level, when averaging over the single class IoU values, one may also choose to weight them by their support, i. e., the true class frequency (equaling the sum of TP and FN). This reduces the importance given to classes that are underrepresented in the image, which may or may not be desired in imbalanced class scenarios, where some classes may cover only small areas of samples. For instance, a class that appears in only a very small fraction of the image and that is falsely or not predicted by the model has an IoU of zero. Without weighting, this will disproportionately affect overall IoU, even if other, more frequent classes are predicted well. Conversely, if the most frequent class (e. g., background) is predicted very well and weighting is used, this will result in a good IoU even if the model performs poorly on the other classes. It is clear that the choice whether to use weighting here also affects the results of a dataset-level IoU computed using the macro/macro approach.

2. Again in the macro case on the image level, there are two possible ways to deal with classes that do not appear in a sample and whose absence is predicted correctly, i. e., both intersection and union are zero. One option is to set IoU for such classes to *NaN*, considering that the fraction is mathematically not defined, and to ignore them for the image IoU. Another option is to set IoU for such classes to 1 in recognition of the correct

prediction. However, if there are several classes that do not appear in an image and their absence is predicted correctly, the resulting image IoU as average of all class IoU values may become rather high but then contains little information about the segmentation performance on the classes that actually appear in the image. Of course, if a weighted average is taken with class true frequency as weights (as described in 3.), there is no difference between setting IoU for classes with correctly predicted absence to *NaN* or 1 because their weight will be zero.

3. The final distinction is similar to 1. but applies to the macro/micro case on dataset level. That is, if (1) the values of TP, FP, and FN are summed up class-wise over all images of the dataset, (2) dataset-wide per-class IoU values are computed, and (3) the average of these is taken. Here, one may choose to use the total, dataset-wide support of each class as weights in taking the average over classes. This again emphasizes performance on frequent classes and reduces the influence of less frequent classes with the same implications as discussed previously.

In appendix C, a guide to the different types of IoU computation is given in terms of equations. For the purpose of this project, the micro/macro approach based on argmax model output was used for evaluation. This allows comparison of IoU values between single images as well as across datasets. To provide additional information, an unweighted macro/micro IoU is also given on the dataset level.

All five trained models as described in section 3.1.2 were evaluated on four of the corresponding test datasets: *wb_m*, *wb_a@m*, *wb_a*, and *bv_a*. Evaluation results on the test dataset of the mixed *wb_m+a* scenario (containing both manually and automatically labeled data) are not given, since this test dataset simply combines those of the scenarios *wb_m* and *wb_a*. Information on the performance on these two data sources is better represented by looking at the two test datasets individually.

3.3 Software and hardware used

The Python library Segmentation Models, which is based on Keras and TensorFlow, was used for setup and training of the CNN (Yakubovskiy, 2019). Augmentations were implemented using the library Albumentations (Buslaev et al., 2018). All training was performed on a Nvidia GeForce GTX 1060 GPU with 6144 MB of VRAM.

4 Results and discussion

4.1 Assessment of automatically generated labels

4.1.1 Label quality and consistency

Wartenberg datasets. Since the training samples for the two corresponding scenarios *wb_m* and *wb_a@m* cover the same geographical extent around identical locations (the centroids of the manual dataset), it was possible to compare them with respect to their quality and consistency. To obtain a quantitative measure, their IoU was computed and found to be 0.79 (micro/macro). While this generally indicates a high degree of consistency, several sources for discrepancies between the two datasets and, thus, their labels can be identified. Figure 4 gives an exemplary comparison showing two training samples from both datasets in which several of the discrepancies occur that are described in the following.

- The manual labels were created using non-orthorectified Google satellite imagery. The automatic labels were derived from CityGML 3D city data, which, according to the data specifications, matches orthorectified aerial imagery. Hence, a certain degree of misalignment is to be expected.
- For both datasets there are cases where one provides more detailed labels than the other, e. g. with respect to the individual delineation of dormers or their omission. From qualitative assessment of random samples it appears that, generally, dormers are more often delineated individually in the manual dataset. Also, roof geometries in the LoD2 3D city model are in some cases simplified to an extent that leads to erroneous representation in the derived labels, particularly for cross-gabled buildings with one or several wings.
- Because the 3D city data at LoD2 does not model roof overhangs whereas they of course appear in the BVV orthophotos, the automatic labels do in many cases not cover the depicted roofs completely, i. e., to their edges. This is a systematic lack of accuracy stemming from the data source, and its effect on the performance of the models should be investigated. This could be done by labeling a certain amount of orthophoto samples manually, train a model on this data and compare its results to those of a model based on 3D city data labels.
- The manual dataset in general only has labels for visible roofs, while the automatically generated labels also cover roof areas that may be hidden underneath vegetation in the orthophotos.
- There is a considerable amount of cases in which the assignment of an orientation class differs between the manual and automatic datasets. This occurs when a roof segment's

orientation is at the boundary between classes. Then, the outcome of the manual labeling process may fall in one orientation class while the orientation computed from the segment's normal vector in 3D city data results in the other orientation class. Potential effects on segmentation performance are unclear and could be investigated separately in the future, but are considered likely to be negligible.

Bavaria dataset. Like the automatically labeled Wartenberg dataset, the Bavaria dataset for the scenario *bv_a* was created from LoD2 3D city data. Therefore, the same observations that were made for the former and described in the previous paragraph also apply to the latter. An additional issue seems to be that in some cases the LoD2 data does not represent roof geometries, leading to entire roofs being labeled as flat even though they clearly are not. The reason for this was found to be the method of roof geometry generation applied in creation of the 3D city data: An algorithm attempts to identify the roof geometry from a point cloud, and, upon failure of this method for instance in cases of complex geometries, it assigns a flat roof at a height derived from other parameters. A generic attribute indicates the roof generation method and allows the quantification of erroneous geometries (Bayerische Vermessungsverwaltung, 2018).

Table 6 shows the distribution of roofs and segments in the Bavaria dataset by their generation method. For 6678 (5.4 %) of all 123050 buildings in the Bavaria dataset (excluding Wartenberg data) no geometry could be automatically deduced from the point cloud, resulting in the assignment of a default flat roof shape that does not match the real structure (values 3100, 3210, 3220). Another 231 roofs apparently were edited manually, but the correctness of their geometries cannot be verified since the corresponding method value 9999 is not specified in the documentation. Figure 5 shows two exemplary training samples in which this problem is visible.

This issue provides a clear starting point for further investigation of the impact these flawed data have on the network's performance by generating training data that only feature labels created from correct roof geometries.

4.1.2 Comparison of roof and segment subdivision between datasets

To account for differences between the Wartenberg datasets that were labeled manually (*wb_m*) and automatically (*wb_a*), their data structures were compared with respect to the number of buildings and roof segments. In the manually labeled dataset *wb_m*, there are 1878 buildings with 4473 roof segments, averaging 2.38 segments per building. In case of the automatically labeled Wartenberg dataset *wb_a*, there are 2835 buildings with 6071 segments, averaging 2.14 segments per building. The dataset *wb_a* identifies 1.35 times as many roof segments and 1.51 times as many separate buildings as the dataset *wb_m*. This indicates that, in comparison to *wb_m*, *wb_a* is characterized by a more fine-grained distinction of separate

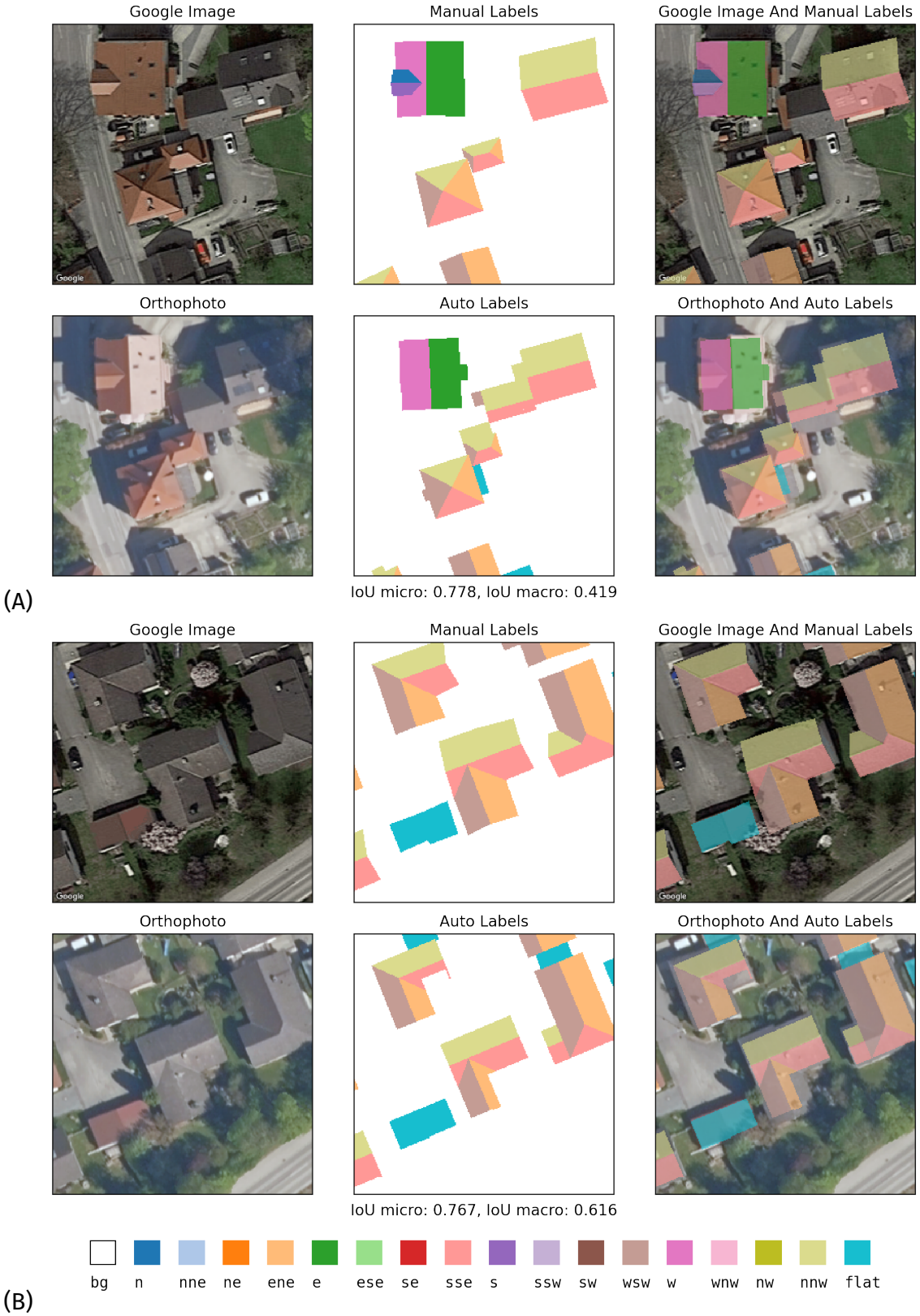


Figure 4: Exemplary comparison of manual and automatic Wartenberg training samples (IoU is given). (A) Manual labels identify dormer, auto labels do not; on the other hand, auto labels identify segments missing in manual dataset. (B) Auto labels do not cover roof overhangs and wrongly represent roof geometry of cross-gabled building to the right.

Table 6: Distribution of Bavaria dataset by roof generation method of the 3D city data: Numbers of buildings and single roof segments for each method. Classification according to Bayerische Vermessungsverwaltung (2018).

Roof generation method		Object count	
Value	Description	Buildings	Segments
1000	Identification algorithm (automatic)	61581	98776
2000	Identification algorithm (semi-automatic, edited)	54560	196308
3100	Unidentified: Flat roof with minimum height	1398	1398
3210	Unidentified: Flat roof with derived height (automatic)	3425	3425
3220	Unidentified: Flat roof with derived height (edited)	1855	1855
4000	Manual input of roof geometry	0	0
9999	Unknown	231	542
	Sum	123050	302304

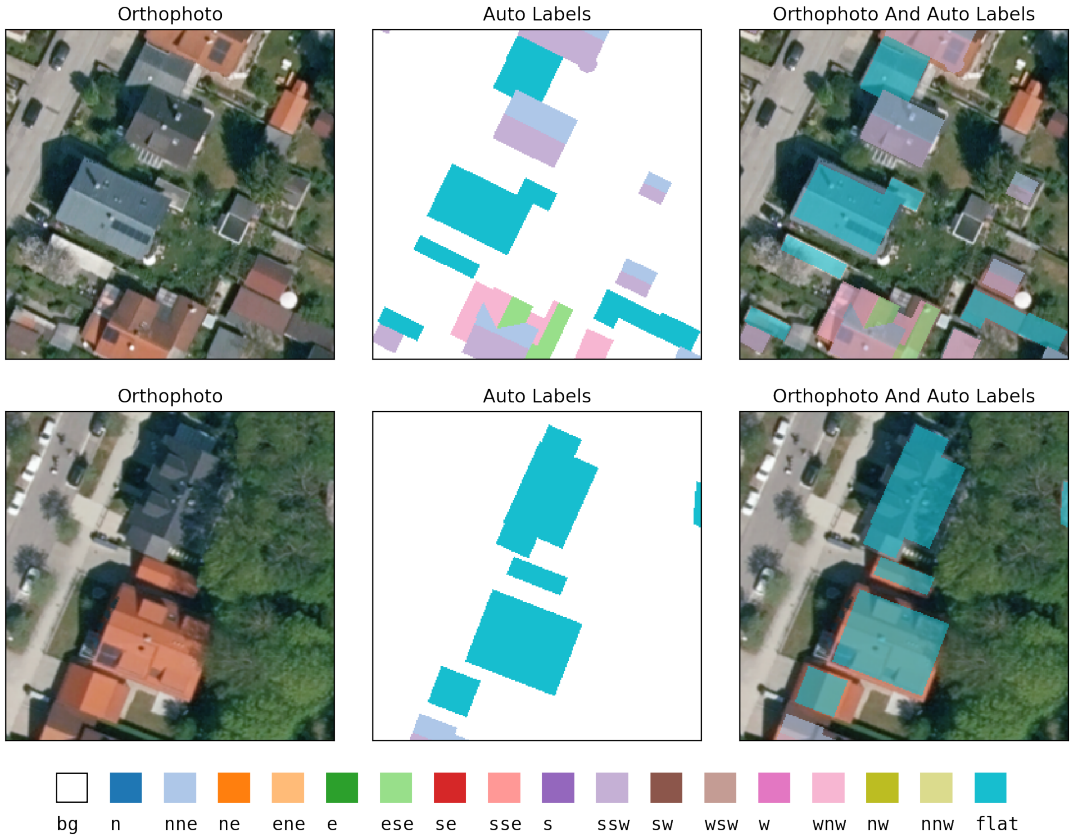


Figure 5: Two exemplary training samples from the Bavaria dataset in which roofs are falsely labeled as flat due to the roof geometry missing in the LoD2 3D city data.

buildings, leading to a higher number of both total buildings and roof segments, but on average fewer segments per building, because there are more smaller, subdivided buildings that therefore comprise fewer segments.

Further, for each building of *wb_m* its number of segments was compared to the number of *wb_a* segments belonging to the same building. The aim was to investigate whether and how the subdivision of identical roof areas into segments is different between the datasets. Due to the different data sources (manually annotated roofs aligned to Google satellite imagery vs. orthorectified 3D city data roofs), the segment and roof outlines of both datasets do not match perfectly. For this reason, the same *wb_a* segment can intersect more than one *wb_m* building, which would distort the results of the analysis by falsely increasing the average number of *wb_a* segments per *wb_m* building. To avoid this, an intersection threshold was implemented as a condition to count a *wb_a* segment as being part of a *wb_m* building: The intersection of both must amount to a minimum ratio of both the *wb_m* building and the *wb_a* segment. The results differ depending on what this ratio is selected to be.

Table 7 shows the results of this analysis. It becomes apparent that in particular for *wb_m* buildings with few roof segments there are on average more roof segments in the *wb_a* dataset, indicating a smaller-scale subdivision of roofs into segments in the 3D city data. For buildings from *wb_m* with higher numbers of roof segments, presumably reflecting larger buildings, this relation does not apply anymore and the number of *wb_a* segments is relatively closer to the number of *wb_m* segments. From 8 *wb_m* segments upwards the numbers appear to become somewhat arbitrary, likely due to a very small number of buildings with that many roof segments, and are therefore hard to interpret with respect to the aim of this analysis.

4.2 Neural network performance

4.2.1 Intersection over union cross-evaluation

Table 8 lists the performance of the five CNN models in terms of intersection over union (micro/macro) evaluated on four of the test datasets. One finds quite a clear separation between the model trained with manually labeled samples and those trained with automatically labeled samples, but also between the Wartenberg models and the model trained on the large Bavaria dataset. The same applies for the corresponding test datasets and the model results on these.

With an IoU of 0.863, the model *wb_m* is the best performer on the manually labeled dataset *wb_m*, which it was trained on and therefore seems to be specialized on. Notably, the model *wb_m+a* trained on mixed (both manually and automatically labeled) data of Wartenberg comes close to this performance at an IoU of 0.847. Also, the Bavaria model scores higher than those only trained on automatic Wartenberg data, indicating a benefit from training on an extended BVV orthophoto datasest when required to generalize to Google satellite data

Table 7: Number of *wb_a* segments that on average intersect *wb_m* buildings with a certain number of segments (first column), depending on an intersection threshold: Minimum ratio of the intersection area to both the *wb_m* building and the *wb_a* segment.

wb_m	Average number of *wb_a* segments by intersection threshold									
segments	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%
1	2.03	1.69	1.58	1.54	1.52	1.51	1.49	1.47	1.45	1.47
2	3.26	2.86	2.68	2.59	2.55	2.52	2.49	2.44	2.37	2.25
3	3.77	3.47	3.42	3.38	3.34	3.33	3.31	3.31	3.24	2.95
4	5.20	4.76	4.59	4.45	4.38	4.33	4.27	4.23	4.16	3.93
5	6.58	6.21	6.13	5.96	5.88	5.75	5.58	5.46	5.13	4.83
6	7.39	7.04	6.79	6.65	6.56	6.49	6.46	6.35	6.25	5.98
7	8.09	7.91	7.55	7.55	7.45	7.45	7.36	7.18	7.18	7.09
8	6.60	6.40	6.40	6.30	6.30	6.00	5.90	5.80	5.80	5.40
9	10.00	10.00	10.00	10.00	10.00	10.00	9.33	9.00	9.00	7.33
10	13.33	13.22	13.22	13.22	13.22	13.22	13.11	12.78	12.56	11.22
11	13.00	13.00	13.00	13.00	13.00	13.00	13.00	13.00	13.00	13.00
12	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00

with manual labels. The lowest IoU values on this dataset are achieved by the two models trained on the automatically labeled Wartenberg datasets, *wb_a@m* and *wb_a*, at 0.744 and 0.757, respectively. The latter delivers a slightly better performance, presumably because it was trained with a higher number of samples.

The results on the test datasets of *wb_a@m* and *wb_a* are very comparable, which can easily be attributed to their strong similarity: Their main difference is in number of training samples and their centroids. Here, the Bavaria model scores highest at 0.877, again indicating the benefit arising from dataset extension. It is, however, closely followed by the models *wb_a@m*, *wb_a*, and *wb_m+a*, which score between 0.85 and 0.86. The worst performing model on the automatically labeled Wartenberg dataset unsurprisingly is the one that was not exposed to the corresponding training data, but only to the manually labeled Wartenberg data: *wb_m* with an IoU of 0.747 and 0.741, respectively.

On average, the evaluation results on the Bavaria test dataset show the lowest IoU scores. This is expectable considering that it is the most challenging dataset with very heterogeneous data from rural and urban areas across Bavaria, where especially the latter differ significantly from the Wartenberg area. The model *bv_a* trained on the corresponding training data achieves the best score at 0.839. All models trained on similarly structured, i. e., automatically labeled data, follow at between 0.744 and 0.758. Interestingly, among these three, the model that was additionally exposed to manually labeled data in training (*wb_m+a*) performs best, albeit only by a slight margin, indicating a benefit from increased training data heterogeneity even if this heterogeneity does not stem from the same parent population. It is unclear, however, whether this result would hold if a statistical analysis were to be conducted

on the outcomes of several training runs. Finally, the model *wb_m* which was trained only on manually labeled data of Wartenberg shows the poorest performance with an IoU of 0.686.

With respect to the main diagonal in table 8, which represents the results of the first four models evaluated on their corresponding test datasets, it is striking that the Bavaria model *bv_a* falls behind the others. Nevertheless, the model manages to surpass the others on the automatically labeled Wartenberg datasets, providing evidence for the fact that dataset extension in the investigated scenario improves roof segment identification. And in relative terms, when looking only at the *bv_a* test set, the *bv_a* model stands out significantly from the others. This, on the one hand, puts into perspective its somewhat sub-par performance on its own test dataset but, on the other hand, simply points to this particular test dataset being the most challenging of all, especially for those models that were limited to Wartenberg training data.

Another noteworthy observation can be made with respect to the model *wb_m+a*, which was trained on the combination of manually and automatically labeled Wartenberg data. While it performs slightly worse than the best models on the test datasets *wb_m* and *wb_a@m* (those which were trained purely on the corresponding training data), on average over both test sets it has a considerably higher IoU (0.849) than either of the more specialized models (0.805 for *wb_m* and 0.799 for *wb_a@m*) and even a slightly higher IoU than the model trained on the extended Bavaria dataset (0.840). This clearly demonstrates that the models are subject to quite a strong degree of specialization on the data they were trained on, and that to enhance a model's capability to identify roof segments in heterogeneously sourced data, ideally samples from all data sources should be represented in the training data. Merely extending a homogeneous datasat with (even much) more data from the same source does not help improving the performance on data from another dataset to the same extent as training on data from both sources does, clearly pointing out a limited potential for generalization.

Table 9 provides results structured identically as in table 8 but in terms of an unweighted macro/micro IoU, which, owing to the computation approach, delivers results differing in absolute values but similar in their relation to each other. As opposed to the micro/macro results, which include an implicit weighting by class frequency, the unweighted macro/micro IoU does not over-emphasize the predominant background class, instead giving all classes identical weight independent of their frequency of occurrence. It is noticeable that the variance of the results is larger, indicating stronger differences between the models according to this measure. Here, the relative improvement of the *bv_a* model over the other models turns out to be larger, indicating that it is in particular better at correctly identifying roof segments, also independent of the background class. Appendix D lists the complete model evaluation results, including further IoU performance metrics.

Table 8: Intersection over union (micro/macro based on argmax model output) of the five models on four test datasets.

	Dataset						
Model	*wb_m*	*wb_a@m*	*wb_a*	*bv_a*	Mean	Std	Mean(*wb_m*, *wb_a@m*)
wb_m	0.863	0.747	0.741	0.686	0.759	0.064	0.805
wb_a@m	0.744	0.854	0.856	0.744	0.799	0.056	0.799
wb_a	0.757	0.860	0.861	0.750	0.807	0.053	0.808
bv_a	0.802	0.877	0.877	0.839	0.849	0.031	0.840
wb_m+a	0.847	0.852	0.856	0.758	0.828	0.041	0.849
Mean	0.802	0.838	0.838	0.756			
Std	0.047	0.046	0.049	0.049			

Table 9: Intersection over union (unweighted macro/micro based on argmax model output) of the five models on four test datasets.

	Dataset						
Model	*wb_m*	*wb_a@m*	*wb_a*	*bv_a*	Mean	Std	Mean(*wb_m*, *wb_a@m*)
wb_m	0.596	0.359	0.342	0.300	0.399	0.116	0.477
wb_a@m	0.338	0.597	0.593	0.403	0.483	0.115	0.467
wb_a	0.385	0.619	0.619	0.422	0.511	0.109	0.502
bv_a	0.511	0.691	0.695	0.626	0.631	0.075	0.601
wb_m+a	0.579	0.606	0.619	0.455	0.565	0.065	0.593
Mean	0.482	0.574	0.574	0.441			
Std	0.104	0.113	0.121	0.106			

4.2.2 Confusion matrices

Appendix E contains confusion matrices of all five models executed on the test sets of the four relevant datasets. Here, only the ones that give the most valuable insights are discussed. Figure 6 shows confusion matrices and dataset-wide IoU values for each class of the four models *wb_a@m*, *wb_a*, *wb_m*, and *bv_a* as found when run on their respective test sets. Some general observations can be made with all models: Across all roof classes, the most frequent false negative prediction is background. Flat roofs appear to be hardest class to distinguish from background, since on all Wartenberg models they have more false negatives due to classification as background than true positives. Only the Bavaria model manages to classify more flat roofs correctly than falsely as background. On the other hand, the Bavaria model more frequently falsely predicts sloped roofs to be flat roofs than the Wartenberg models. This could potentially be attributed to the 5.4 % of roofs in the Bavaria dataset that are labeled as flat by default due to failure of the roof geometry identification algorithm (see section 4.1.1). Regarding sloped roofs in general, there seems to be a weak pattern showing that their orientation is, if at all, prevalently confused with the opposite or orthogonal orientations (differing by 90°, 180°, or 270°). An explanation could be that roof segments in opposing and orthogonal orientations would have the same outline and potentially also similarly oriented textures, so there is presumably less evidence for the model to distinguish between them.

While the confusion matrices of *wb_a@m* and *wb_a* look very much alike as is to be expected considering that they were trained on similar datasets, true positive rates and class-wise IoU are slightly higher for the model *wb_a*, which was exposed to a larger number of training samples (2085 vs. 1364). Another interesting comparison is that between the models *wb_a@m* and *wb_m*, which were trained on identically positioned samples from the two different sources. In fact, their performance on their own test datasets is very comparable, confirming the observation made earlier in terms of IoU (see section 4.2.1). There are some differences in details: For instance, the model *wb_m* performs slightly better at identifying flat roofs. Finally, the model *bv_a* achieves the most consistent performance with respect to class-wise IoU and true positive rates, the lowest values for sloped roofs being 0.58 and 0.7, respectively. While its results on flat roofs fall somewhat behind in comparison to that, it still outperforms all three Wartenberg models in this regard.

Figure 7 shows confusion matrices and dataset-wide IoU values for each class of the models *bv_a* and *wb_m+a* when evaluated on the test sets of the scenarios *wb_m* and *wb_a@m*. Its purpose is to further explore how well the Bavaria model and the model trained on mixed (manually and automatically labeled) Wartenberg data perform on purely manually or automatically labeled Wartenberg data.

Again, it becomes clear that the extended Bavaria dataset based on 3D city data provides only limited improvement in terms of the model's ability to generalize to manually labeled Google satellite images of Wartenberg, as was already discussed in the previous section. Re-

sults of the Bavaria model on the automatically labeled Wartenberg dataset *wb_a@m*, on the other hand, show a definite improvement over the model trained purely on those data: Class-wise IoU values and true positive rates are more consistent, identification of flat roofs more reliable than for the model *wb_a@m* as seen in fig. 6.

The bottom two matrices in fig. 7 again illustrate the benefit of a model trained on data from both sources when applied to this use case. The results indicate that the CNN has the capacity to identify roof segments in data from either source, provided that it was exposed to corresponding data during training.

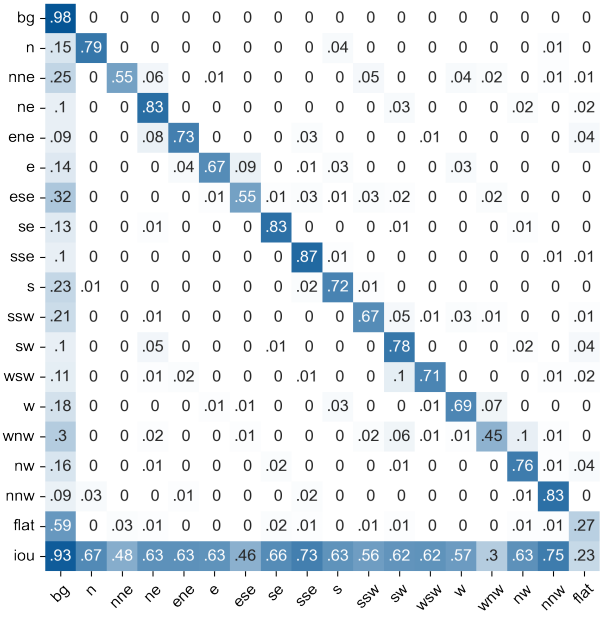
4.2.3 Exemplary model predictions

Figure 8 presents predictions from all five models on six exemplary test samples from the datasets *wb_m* and *wb_a@m*: One sample from each of the two datasets at three different locations. In all cases, the depicted roof structures are rather complex.

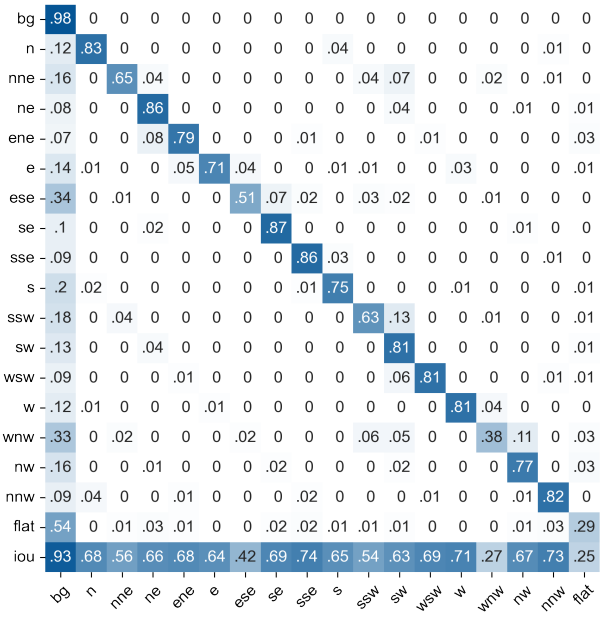
The samples from location (A) contain residential buildings mainly with roofs sloped towards north and south. Manual and 3D-city-data-derived labels show good consistency. The models *wb_m* and *wb_m+a* perform best on the Google satellite image, the first scoring a higher micro IoU and the second a higher macro IoU. This could be interpreted as a better prediction of background pixels in the first case and better prediction of roof pixels in the second case. Qualitatively, however, *wb_m* seems to deliver more accurate outlines of the roof segments. Among the models trained purely on automatically labeled data, the Bavaria model *bv_a* scores highest, coming quite close to the other two. It is able to detect almost all constituent roof segments except one at the right image edge, which has a low contrast compared to the background. The models *wb_a@m* and *wb_a* fall behind in comparison.

Roof segments in the corresponding BVV orthophoto at the same location are predicted best by the Bavaria model *bv_a*. It is the only one capable of outlining the small roof connecting the two buildings in the south-west corner of the sample. The models *wb_m+a*, *wb_a*, and *wb_a@m* follow in this order sorted by performance, corresponding to the number of samples they were trained on. The model *wb_m*, trained exclusively on manually labeled data, clearly has difficulties interpreting the BVV orthophoto, delivering the lowest micro IoU and second-lowest macro IoU among all examples from this location.

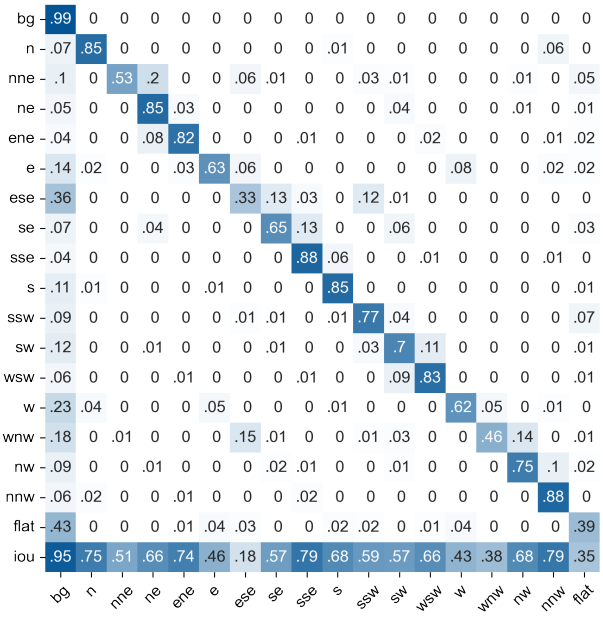
Location (B) contains two pyramid roofs, which generally pose a greater challenge to the networks due to their lower (although unquantified) frequency in the training data. Comparison of the labels for the location reveals a roof that was falsely classified as flat in the manual data: The two-dimensional information does not allow identification of its gabled geometry. The automatically generated labels, on the other hand, contain a gable roof where in the BVV orthophoto only a parking lot is visible, indicating outdated data. In view of the predictions on the Google satellite image, it is interesting to observe that the models *wb_m* and *bv_a* make the same mistake as the human labeler, classifying the roof in the south-east corner



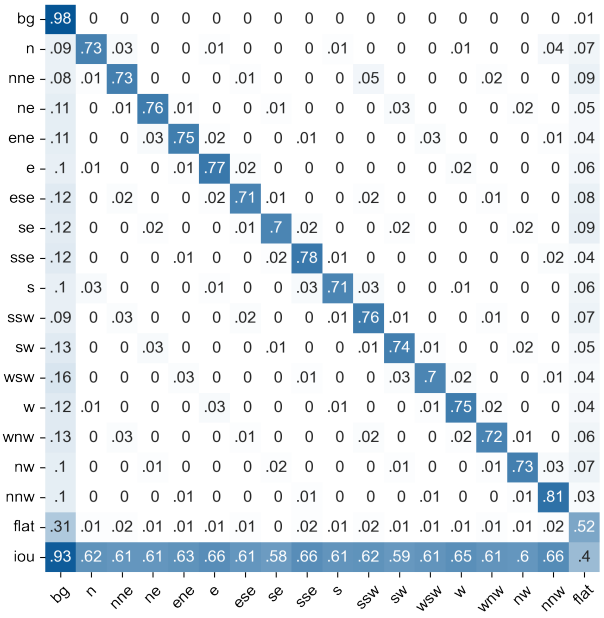
wb_a@m (.854, .597)



wb_a (.861, .619)

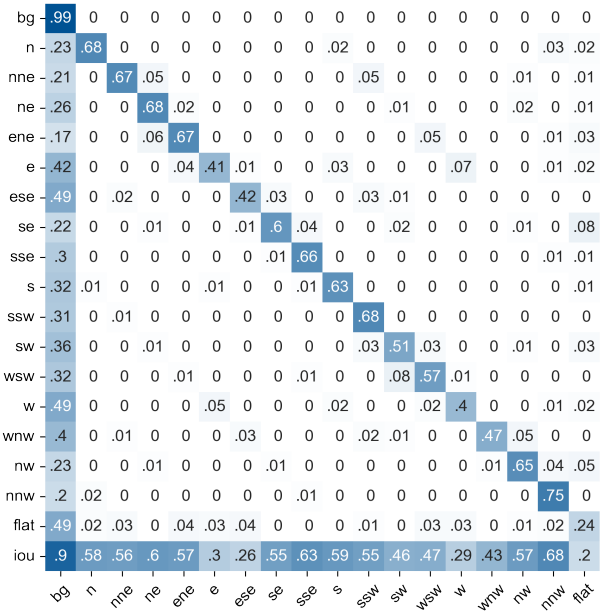


wb_m (.863, .596)

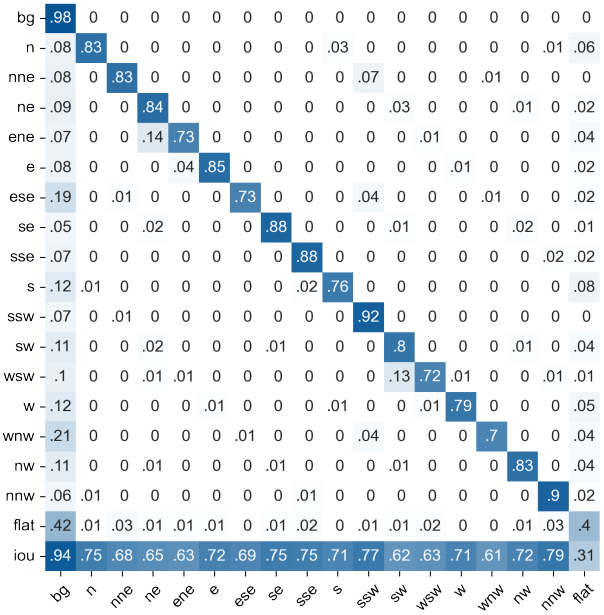


bv_a (.839, .626)

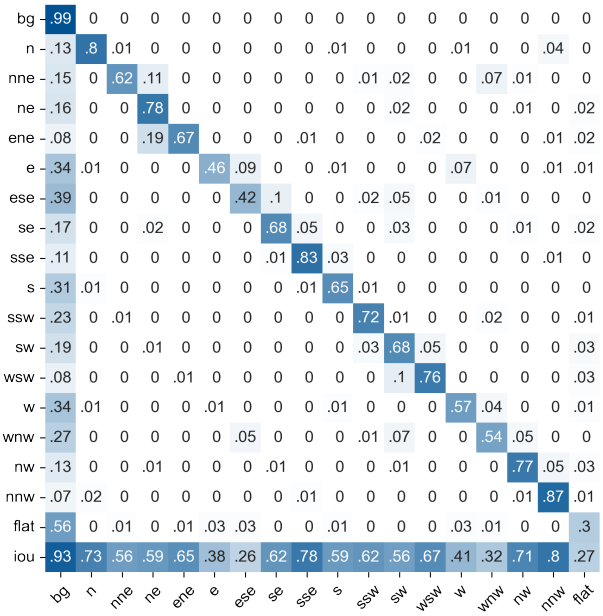
Figure 6: Confusion matrices of the models *wb_a@m*, *wb_a*, *wb_m*, and *bv_a* on their respective test sets. Rows are ground truth labels, columns are predictions. Rows are normalized to total number of predictions, thus, sum up to 1. Last row shows IoU of the corresponding class on the dataset level (macro/micro). Numbers in brackets next to identifiers are micro/macro and unweighted macro/micro IoU values.



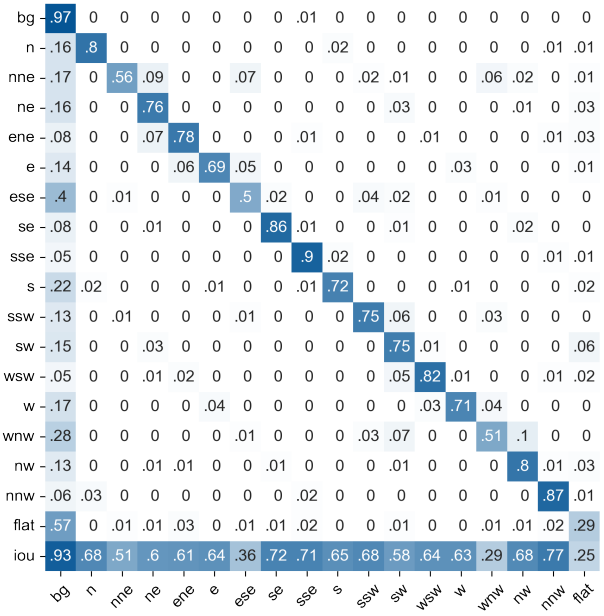
bv_a → *wb_m* (.802, .511)



bv_a → *wb_a@m* (.877, .691)



wb_m+a → *wb_m* (.847, .579)



wb_m+a → *wb_a@m* (.852, .606)

Figure 7: Confusion matrices of the models *bv_a* and *wb_m+a* on the test sets of *wb_m* and *wb_a@m*. Rows are ground truth labels, columns are predictions. Rows are normalized to total number of predictions, thus, sum up to 1. Last row shows IoU of the corresponding class on the dataset level (macro/micro). Numbers in brackets next to identifiers are micro/macro and unweighted macro/micro IoU values.

of the sample as flat. Quantitatively, the model *wb_m* scores highest, and it also delivers the best qualitative performance. Its predictions of the pyramid roofs are mostly accurate but potentially not sufficient for a practical application. The model *wb_m+a* follows but, purely visually, does not give satisfactory results. The Bavaria model *bv_a* does a significantly better job at classifying the pixels than the two Wartenberg models *wb_a* and *wb_a@m*, but all three have difficulties.

In the BVV orthophoto at location (B), the pyramid roofs are identified quite well by the Bavaria model. It has difficulties, however, to correctly classify two of the gabled roofs in the sample. The south-east one is again determined to be flat, which could be due to the 5.4 % of roofs in its training data that were falsely labeled as flat, or simply because the roof's geometry is, in fact, hard to discern in the image. The models *wb_m+a* and *wb_a* take the second places with respect to macro and micro IoU, respectively, followed by *wb_a@m*. The model *wb_m*, not exposed to automatically labeled orthophotos in training, manages to identify some segments but fails with most, reflected in the lowest scores.

Location (C) shows a group of buildings with roofs oriented in various directions. Again, some inconsistencies between the labels of Google satellite image and BVV orthophoto are visible. Looking at the Google satellite image, the best performance is in this case delivered by *wb_m+a*, which apparently benefits from the additional training data with automatically generated labels. The model *wb_m* has difficulties identifying correct outlines and unambiguous orientations. Among the models only trained on 3D-city-data-based samples, only *bv_a* performs decently. The others return mostly chaotically structured patches. Roof segments in the BVV orthophoto at location (C) are well outlined by most models, in particular by *bv_a*, and at least by *wb_m*.

In summary, these examples reflect the results that were found in terms if overall performance: The Bavaria model clearly benefits from the vastly increased amount of training data, both when applied to Google satellite images and BVV orthophotos. It is, in general, closely followed by *wb_m+a* in the case of BVV orthophotos, and surpassed by this model for Google satellite images. The model *wb_m* in most instances has difficulties identifying outlines and orientations of roof segments in BVV orthophotos. The examples illustrate the point made in the previous section that specialization on a data source is more important that sheer amount of training data in the present case.

Figure 9 shows predictions from all models on three exemplary test samples from the Bavaria dataset *bv_a*. They are located in urban areas in central Munich and therefore contain buildings of a type that only the Bavaria model was trained on. This is clearly reflected in predictions and performance measures: In all three cases it is apparent that only *bv_a* is capable of identifying the roof segments with good quality, as is visible and indicated by IoU. The other models are generally able to detect portions of the roof segments, even their correct orientation, but fail to identify their full area and outline. Among them, the model *wb_m+a* scores best in all cases, likely benefiting from its increased amount of training data

and potentially also their higher diversity compared to the other Wartenberg models. The poorest performance is, unsurprisingly, delivered by *wb_m*, which was not exposed to any BVV orthophotos during training.

4.3 Discussion of research hypotheses

The results shall now be discussed with respect to the research hypotheses that were established in the beginning:

1. Roof positional information from 3D city models can be extracted and used to label the corresponding roof areas in aerial or satellite images.
2. The automatically generated labels are accurate enough in outlining the roofs to allow effective training of the neural network.
3. Training the neural network with the extended and less homogeneous dataset delivers superior performance compared to the original hand-labeled dataset.

The first hypothesis was the least debatable and can be confirmed with certainty: Following the steps described in section 3.1 it was possible to generate datasets of training samples from 3D city data with labels outlining roof segments in corresponding BVV orthophoto crops. Contrary to what was assumed before the project, the roof segments could be transferred to two-dimensional format by simply projecting them onto the two-dimensional plane and no further geometrical transformations were required.

Depending on the definition of effectivity, one may also confirm the second hypothesis. Even though it was found that the labels created from 3D city data lack accuracy in some respects (see section 4.1.1), their overall quality proved to be sufficient to enable training of CNN models that achieve a performance at least comparable to and in some cases surpassing that of the model trained on manually labeled data.

The third hypothesis can only be confirmed partially: Indeed it is the case that the model trained on the extended Bavaria dataset performs better than the models trained only on Wartenberg data, but only when tested on datasets that were created from the same data source, i. e., 3D city data and BVV orthophotos. When tested on manually labeled Google satellite data, the Bavaria model's performance is inferior to that of the model that is specialized on this data source, even though the latter only knows Wartenberg and was not exposed to even closely the same quantity or variety of buildings as the Bavaria model. In other words: Specialization on a data source beats quantity and variety of training data in this case.

4.4 Further insights and suggestions for follow-up investigations

There is room for improvement concerning the approach to splitting the datasets into subsets for training, validation, and testing. Under ideal conditions, all datasets would follow

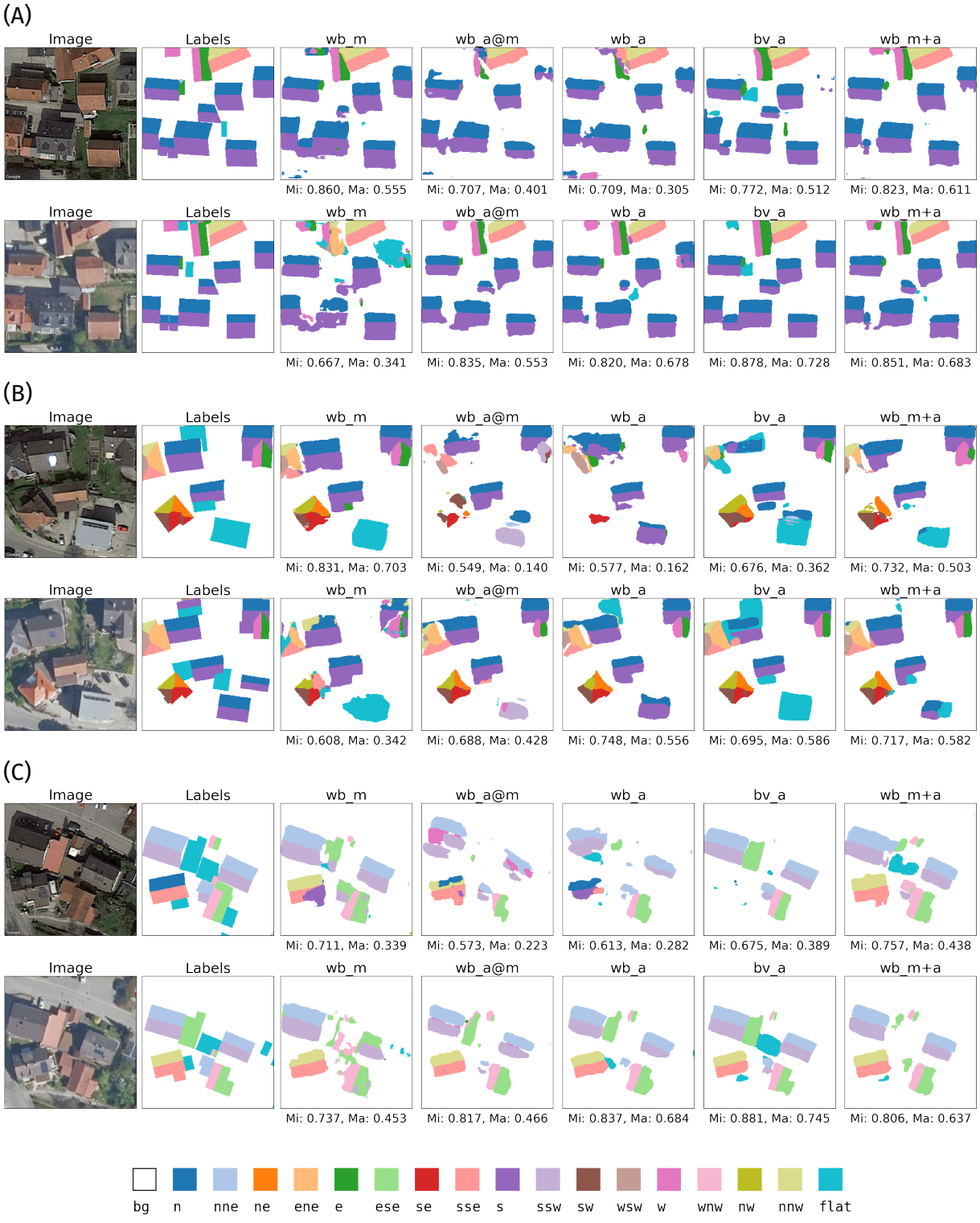


Figure 8: Six test samples from the Wartenberg datasets *wb_m* and *wb_a@m* at three locations (A), (B), and (C), and corresponding predictions from all models. For each location, the upper sample is from *wb_m* (Google satellite image, manual labels) and the lower sample is from *wb_a@m* (BVV orthophoto, auto labels). Micro (Mi) and macro (Ma) IoU with respect to labels are given for each prediction.

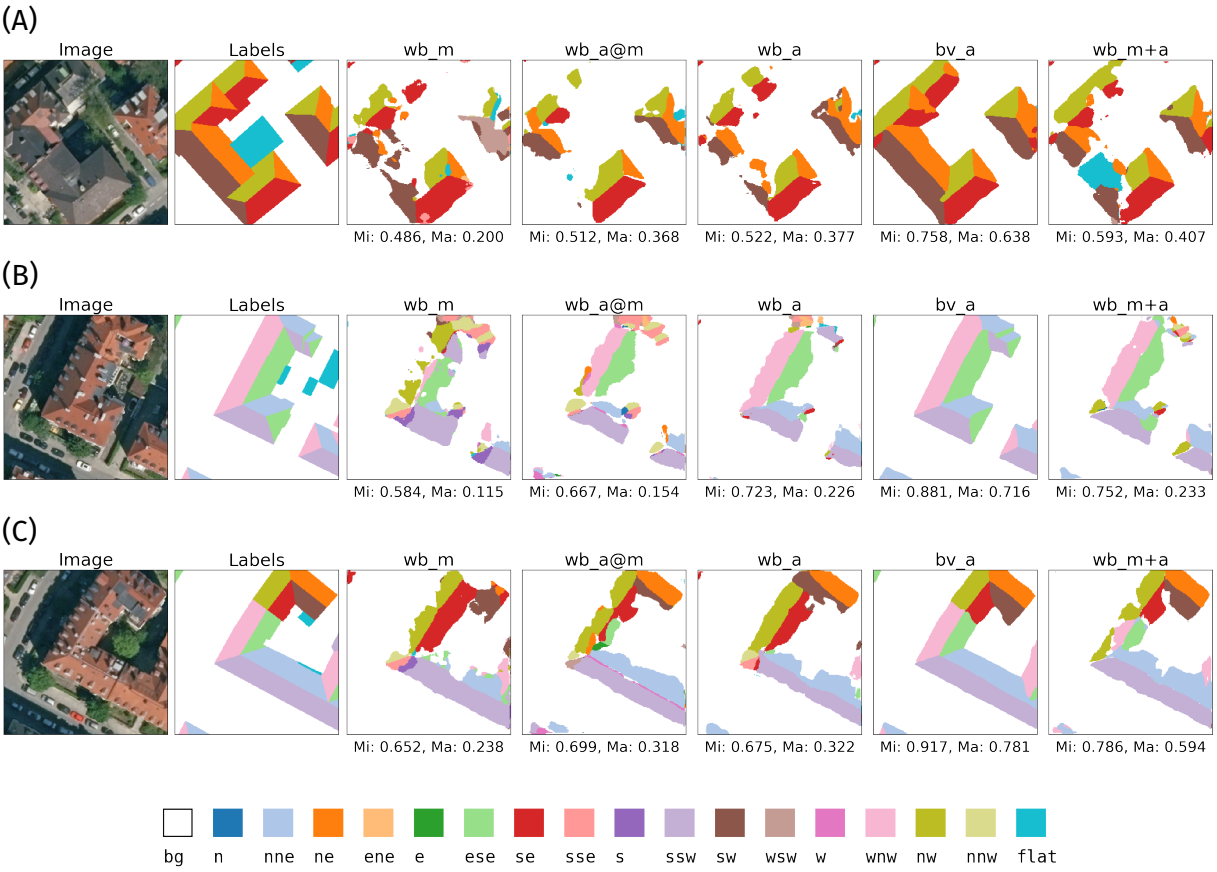


Figure 9: Three test samples (A), (B), and (C) from the Bavaria dataset *bv_a*, and corresponding predictions from all models. Micro (Mi) and macro (Ma) IoU with respect to labels are given for each prediction.

the same distribution, but this is difficult to achieve in spatially heterogeneous data. The approach used in this study aims at reducing spatial bias by allowing the selection of several locations for each of the subsets around which their samples are selected instead of only one. While this may reduce the bias somewhat, it will certainly not eliminate it. It would be better to select the subset samples completely at random. This would, however, lead to a greater reduction of usable samples than the lumped approach taken here, because all samples intersecting buildings that themselves intersect validation and test set samples cannot be used in any of the other sets. Another promising approach could be to generate grid-based instead of roof-centered samples. By means of shifting the grid by a fraction of the cell size one could further increase the number of samples, equaling a pre-training augmentation strategy. A potential downside would be a lower number of roofs that are depicted fully within samples.

The quality of the automatically generated labels could be improved in two ways. First, the problem of roofs that are erroneously labeled as flat (discussed in section 4.1.1) should be addressed by excluding all samples containing such roofs from the data. Considering that this concerns 5.4 % of all samples in the Bavaria dataset, this should lead to an improvement in segmentation performance. Second, a threshold value could be implemented for the distinction between flat and sloped roofs. So far, any roof with non-zero slope as computed from its normal vector was classified as sloped. Among all 2835 (6071) buildings (segments) in the *wb_a* dataset, 776 (851) have zero slope, 37 (52) have a slope smaller than 5°, and 127 (198) have a slope smaller than 10°. Among all 123 050 (302 304) buildings (segments) in the *bv_a* dataset, 45 333 (59 532) have zero slope (including the falsely labeled ones), 2733 (4199) have a slope smaller 5°, and 11 118 (17 637) have a slope smaller than 10°. It is conceivable that many of these barely sloped roofs are hard to distinguish from flat roofs in the images, which would suggest a potential improvement in training effectiveness and model performance if they were classified as flat in the label generation process.

Any interpretation of the results depends entirely on the measure used for their quantification and comparison. As discussed in section 3.2.3 and further elaborated in appendix C, there are manifold ways to compute the seemingly unambiguous intersection over union, each of which emphasizes different qualities in the predictions. They therefore can yield quite different results, which must be interpreted with respect to the method of computation. Depending on the scenario at hand, one metric's advantage can become a drawback and vice-versa. For instance, the micro IoU at image level has the advantage that a weighting of classes by their frequency is implied, which helps avoid the influence of very small classes that are predicted badly if the larger classes show good results. On the other hand, if there is one class (like the background class) that is exceedingly more frequent than all other classes, a model can obtain good results (in terms of this measure) by mostly predicting background areas, turning the advantage into a drawback. For this reason, all image-level micro IoU values reported here should be taken with a pinch of salt and compared to the corresponding macro IoU values. The importance of an appropriate application and interpretation

of object detection and segmentation performance metrics in consideration of their inherent limitations is also being discussed in recent literature (Kofler et al., 2021; Reinke et al., 2021). Another drawback of IoU as a performance measure is that it cannot be optimized directly in training because it has non-zero gradients in some cases. Several new, IoU-derived metrics were proposed in recent work, such as a generalized IoU (GIoU) and a generalized IoU for pixelwise prediction (PixIoU), that aim to circumvent this problem (Rezatofighi et al., 2019; Yu et al., 2021). For future investigations it might also be worth trying to use a micro IoU metric that excludes the background class from its computation entirely. This avoids the problem of one disproportionately frequent class distorting the results but retains the advantage of giving more weight to more frequent classes in each image.

Considering the finding that the CNNs show quite strong specialization on the data source they were trained on, and that data augmentation alone does not suffice to overcome this limitation, one could try to find approaches that allow better generalization of the CNNs to other sources of remote sensing imagery. This was already tried in a first attempt by combining the datasets *wb_m* and *wb_a* into a single dataset *wb_m+a* and, indeed, this helped the corresponding CNN to achieve good performance on data from both Google satellite images and BVV orthophotos. A next step could be for example to apply the same approach to the Bavaria model.

5 Conclusions

In this project, roof positional information from semantic 3D city models was successfully used to generate datasets of images and labels from BVV orthophotos for the purpose of training CNNs to identify roof segments and their orientation. The involved methods included spatial data analyses in a 3DCityDB instance in a PostGIS-extended PostgreSQL database, automated generation of training samples by retrieving orthophoto crops from a BVV Web Map Service and rasterizing the corresponding roof surface geometries from 3D city data, choosing suitable areas for a vastly extended training dataset, the execution of a spatial data split for several training scenarios, the setup, tuning, and training of CNNs on the generated datasets, followed by numerical as well as visual evaluation of their results.

As can be seen from the discussion, it is difficult to come to a definitive conclusion as to whether a dataset extension using 3D city models as conducted in this project leads to better performance of an identically structured neural network compared to the original, manually labeled dataset. This is mainly due to the specialization of the models on the respective type of training data (Google satellite images and BVV orthophotos, respectively), which inhibits ideal comparability between them. In order to unambiguously identify any improvements achieved by using an extended, 3D-city-data-based dataset with its potential drawbacks instead of a smaller but manually labeled and, therefore, presumably less error-prone dataset, it would be better to have the latter be based on the same image material, i. e., BVV orthophotos. This would enable identification of the impact of label characteristics that are introduced by properties of the 3D city data, such as their incomplete coverage of visible roof areas due to roof overhangs not being represented in the 3D model, and to separate this from the additional effect of network specialization on data source.

Another problem arising from roof overhangs frequently not being included by the automatic labels is that IoU as a performance measure loses meaning: A perfect IoU would not imply perfect model performance, but rather that many roof segments are predicted incompletely. Conversely, a perfect model performance in identifying roof segments and their orientation would not be reflected in a perfect IoU. One could argue that the 3D-city-data-based labels do not teach the model to identify roof segments but to predict the extent of the underlying building footprint. Availability of manual labels for a part of the same base imagery could again help overcome this problem by enabling quantification with respect to the actual predicition target: complete roof segments.

Nevertheless, the present data allow a first evaluation of the potential that arises from this type of automated labeling. The results are promising and point to the fact that roof segment information from 3D city models can, in fact, be used to quickly and cost-efficiently generate large datasets that enable effective training of a CNN for the purpose of roof segment identification, although further work must be done to improve both the generated datasets and performance evaluation.

References

Bayerische Vermessungsverwaltung (2018). *Kundeninformation LoD2 Gebäudemodelle. Stand 3/2018*.

BMVI (2018). *Regionalstatistische Raumtypologie (RegioStaR) des BMVI für die Mobilitäts- und Verkehrsforschung. Arbeitspapier Version V1.1 (06.06.2018)*. Bundesministerium für Verkehr und digitale Infrastruktur.

Bódis, K., I. Kougias, A. Jäger-Waldau, N. Taylor, and S. Szabó (2019). "A high-resolution geospatial assessment of the rooftop solar photovoltaic potential in the European Union". In: *Renewable and Sustainable Energy Reviews* 114. PII: S1364032119305179, p. 109309. DOI: 10.1016/j.rser.2019.109309.

Buslaev, A., A. Parinov, E. Khvedchenya, V. I. Iglovikov, and A. A. Kalinin (2018). "Albumentations: fast and flexible image augmentations". In: *ArXiv e-prints*.

Castello, R., S. Roquette, M. Esguerra, A. Guerra, and J.-L. Scartezzini (2019). "Deep learning in the built environment: automatic detection of rooftop solar panels using Convolutional Neural Networks". In: *Journal of Physics: Conference Series* 1343. nov, p. 012034. DOI: 10.1088/1742-6596/1343/1/012034.

Castello, R., A. Walch, R. Attias, R. Cadei, S. Jiang, and J.-L. Scartezzini (2021). "Quantification of the suitable rooftop area for solar panel installation from overhead imagery using Convolutional Neural Networks". In: *Journal of Physics: Conference Series* 2042.1, p. 012002. DOI: 10.1088/1742-6596/2042/1/012002.

Collier, E., S. Mukhopadhyay, K. Duffy, S. Ganguly, G. Madanguit, S. Kalia, G. Shreekant, R. Nemani, A. Michaelis, S. Li, and A. Ganguly (2021). "Semantic Segmentation of High Resolution Satellite Imagery using Generative Adversarial Networks with Progressive Growing". In: *Remote Sensing Letters* 12.5, pp. 439–448. DOI: 10.1080/2150704X.2021.1895444.

Costa, M. V. C. V. d., O. L. F. de Carvalho, A. G. Orlandi, I. Hirata, A. O. de Albuquerque, F. V. e. Silva, R. F. Guimarães, R. A. T. Gomes, and O. A. d. C. Júnior (2021). "Remote Sensing for Monitoring Photovoltaic Solar Plants in Brazil Using Deep Semantic Segmentation". In: *Energies* 14.10. PII: en14102960, p. 2960. DOI: 10.3390/en14102960.

Freitas, S., C. Catita, P. Redweik, and M. C. Brito (2015). "Modelling solar potential in the urban environment: State-of-the-art review". In: *Renewable and Sustainable Energy Reviews* 41. PII: S1364032114007461, pp. 915–931. DOI: 10.1016/j.rser.2014.08.060.

Goodfellow, I., Y. Bengio, and A. Courville (2016). *Deep learning*. Adaptive computation and machine learning. Cambridge, Massachusetts: The MIT Press.

He, K., X. Zhang, S. Ren, and J. Sun (2016). "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (Las Vegas, NV, USA, 2016). IEEE, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

Hoog, J. de, S. Maetschke, P. Ilfrich, and R. R. Kolluri (2020). "Using Satellite and Aerial Imagery for Identification of Solar PV". In: *Proceedings of the Eleventh ACM International Conference on Future Energy Systems*. e-Energy '20: The Eleventh ACM International Conference on Future Energy Systems (Virtual Event Australia, 2020). New York, NY, USA: ACM, pp. 308–313. DOI: 10.1145/3396851.3397681.

ISO, ed. (2015). *ISO 19109:2015. Geographic information — Rules for application schema*.

Izquierdo, S., M. Rodrigues, and N. Fueyo (2008). "A method for estimating the geographical distribution of the available roof surface area for large-scale photovoltaic energy-potential evaluations". In: *Solar Energy* 82.10. PII: S0038092X08000625, pp. 929–939. DOI: 10.1016/j.solener.2008.03.007.

Jadon, S. (2020). "A survey of loss functions for semantic segmentation". In: *2020 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*. 2020 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB) (Via del Mar, Chile, 2020). IEEE, pp. 1–7. DOI: 10.1109/CIBCB48159.2020.9277638.

Kemmerzell, N. (2020). "Automatic analysis of economic pv-potential via aerial images and GIS". Chair for Automotive Technology. Master's thesis. Technical University of Munich.

Kofler, F., I. Ezhov, F. Isensee, F. Balsiger, C. Berger, M. Koerner, J. Paetzold, H. Li, S. Shit, R. McKinley, S. Bakas, C. Zimmer, D. Ankerst, J. Kirschke, B. Wiestler, and B. H. Menze (2021). *Are we using appropriate segmentation metrics? Identifying correlates of human expert perception for CNN training beyond rolling the DICE coefficient*. DOI: 10.48550/arXiv.2103.06205.

Kolbe, T. H. (2009). "Representing and Exchanging 3D City Models with CityGML". In: *3D Geo-Information Sciences*. Ed. by J. Lee and S. Zlatanova. Lecture Notes in Geoinformation and Cartography. Berlin and London: Springer, pp. 15–31. DOI: 10.1007/978-3-540-87395-2_2.

Krapf, S., N. Kemmerzell, S. Khawaja Haseeb Uddin, M. Hack Vázquez, F. Netzler, and M. Lienkamp (2021). "Towards Scalable Economic Photovoltaic Potential Analysis Using Aerial Images and Deep Learning". In: *Energies* 14.13. PII: en14133800, p. 3800. DOI: 10.3390/en14133800.

Lecun, Y., L. Bottou, Y. Bengio, and P. Haffner (1998). "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11, pp. 2278–2324. DOI: 10.1109/5.726791.

Lee, S., S. Iyengar, M. Feng, P. Shenoy, and S. Maji (2019). "DeepRoof: A Data-Driven Approach For Solar Potential Estimation Using Rooftop Imagery". In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. KDD '19. New York, NY, USA: Association for Computing Machinery, pp. 2105–2113. DOI: 10.1145/3292500.3330741.

Mayer, K., Z. Wang, M.-L. Arlt, D. Neumann, and R. Rajagopal (2020). "DeepSolar for Germany: A deep learning framework for PV system mapping from aerial imagery". In: *2020 International Conference on Smart Energy Systems and Technologies (SEST)*. Sep., pp. 1–6. DOI: 10.1109/SEST48500.2020.9203258.

Melius, J., R. Margolis, and S. Ong (2013). *Estimating Rooftop Suitability for PV: A Review of Methods, Patents, and Validation Techniques*. DOI: 10.2172/1117057.

Muftah, H., T. S. L. Rowan, and A. P. Butler (2021). "Towards open-source LOD2 modelling using convolutional neural networks". In: *Modeling Earth Systems and Environment*. PII: 1159. DOI: 10.1007/s40808-021-01159-8.

Open Geospatial Consortium, ed. (2012). *OGC City Geography Markup Language (CityGML) Encoding Standard*. OpenGIS Encoding Standard.

Qin, Y., Y. Wu, B. Li, S. Gao, M. Liu, and Y. Zhan (2019). "Semantic Segmentation of Building Roof in Dense Urban Environment with Deep Convolutional Neural Network: A Case Study Using GF2 VHR Imagery in China". eng. In: *Sensors (Basel, Switzerland)* 19.5. PII: s19051164 Journal Article, p. 1164. DOI: 10.3390/s19051164.

Rausch, B., K. Mayer, M.-L. Arlt, G. Gust, P. Staudt, C. Weinhardt, D. Neumann, and R. Rajagopal (2020). "An Enriched Automated PV Registry: Combining Image Recognition and 3D Building Data". In: *CoRR* abs/2012.03690.

Reinke, A., M. D. Tizabi, C. H. Sudre, M. Eisenmann, T. Rädsch, M. Baumgartner, L. Acion, M. Antonelli, T. Arbel, S. Bakas, P. Bankhead, A. Benis, M. J. Cardoso, V. Cheplygina, B. Cimini, G. S. Collins, K. Farahani, B. Glocker, P. Godau, F. Hamprecht, D. A. Hashimoto, D. Heckmann-Nötzel, M. M. Hoffmann, M. Huisman, F. Isensee, P. Jannin, C. E. Kahn, A. Karargyris, A. Karthikesalingam, B. Kainz, E. Kavur, H. Kenngott, J. Kleesiek, T. Kooi, M. Kozubek, A. Kreshuk, T. Kurc, B. A. Landman, G. Litjens, A. Madani, K. Maier-Hein, A. L. Martel, P. Mattson, E. Meijering, B. Menze, D. Moher, K. G. M. Moons, H. Müller, F. Nickel, J. Petersen, G. Polat, N. Rajpoot, M. Reyes, N. Rieke, M. Riegler, H. Rivaz, J. Saez-Rodriguez, C. S. Gutierrez, J. Schroeter, A. Saha, S. Shetty, B. Stieltjes, R. M. Summers, A. A. Taha, S. A. Tsaftaris, B. van Ginneken, G. Varoquaux, M. Wiesenfarth, Z. R. Yaniv, A. Kopp-Schneider, P. Jäger, and L. Maier-Hein (2021). *Common Limitations of Image Processing Metrics: A Picture Story*. This is a dynamic paper on limitations of commonly used metrics. The current version discusses metrics for image-level classification, semantic segmentation, object detection and instance segmentation. For missing use cases, comments or questions, please contact a.reinke@dkfz.de or l.maier-hein@dkfz.de. Substantial contributions to this document will be acknowledged with a co-authorship. DOI: 10.48550/arXiv.2104.05642.

Rezatofighi, H., N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese (2019). "Generalized Intersection Over Union: A Metric and a Loss for Bounding Box Regression". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June.

Ronneberger, O., P. Fischer, and T. Brox (2015). "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *Medical image computing and computer-assisted intervention - MICCAI 2015. 18th International Conference, Munich, Germany, October 5-9, 2015, proceedings / Nassir Navab, Joachim Hornegger, William M. Wells, Alejandro F. Frangi (eds.)* Ed. by N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi. Vol. 9351. LNCS sublibrary: SL6 - Image

processing, computer vision, pattern recognition, and graphics 9349-9351. Cham: Springer, pp. 234–241. DOI: 10.1007/978-3-319-24574-4_28.

Schmidhuber, J. (2015). "Deep learning in neural networks: An overview". In: *Neural Networks* 61, pp. 85–117. DOI: 10.1016/j.neunet.2014.09.003.

Sugino, T., T. Kawase, S. Onogi, T. Kin, N. Saito, and Y. Nakajima (2021). "Loss Weightings for Improving Imbalanced Brain Structure Segmentation Using Fully Convolutional Networks". eng. In: *Healthcare (Basel, Switzerland)* 9.8. Journal Article. DOI: 10.3390/healthcare9080938.

Willenborg, B., M. Pültz, and T. H. Kolbe (2018). "Integration of semantic 3D city models and 3D mesh models for accuracy improvements of solar potential analyses". In: *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XLII-4/W10, pp. 223–230. DOI: 10.5194/isprs-archives-XLII-4-W10-223-2018.

Xu, S., X. Pan, E. Li, B. Wu, S. Bu, W. Dong, S. Xiang, and X. Zhang (2018). "Automatic Building Rooftop Extraction From Aerial Images via Hierarchical RGB-D Priors". In: *IEEE Transactions on Geoscience and Remote Sensing* 56.12, pp. 7369–7387. DOI: 10.1109/TGRS.2018.2850972.

Yakubovskiy, P. (2019). *Segmentation Models*. https://github.com/qubvel/segmentation_models.

Yao, Z., C. Nagel, F. Kunde, G. Hudra, P. Willkomm, A. Donaubauer, T. Adolphi, and T. H. Kolbe (2018). "3DCityDB - a 3D geodatabase solution for the management, analysis, and visualization of semantic 3D city models based on CityGML". In: *Open Geospatial Data, Software and Standards* 3.1. PII: 46. DOI: 10.1186/s40965-018-0046-7.

Yu, J., Z. Wang, A. Majumdar, and R. Rajagopal (2018). "DeepSolar: A Machine Learning Framework to Efficiently Construct a Solar Deployment Database in the United States". In: *Joule* 2.12. PII: S2542435118305701, pp. 2605–2617. DOI: 10.1016/j.joule.2018.11.021.

Yu, J., J. Xu, Y. Chen, W. Li, Q. Wang, B. Yoo, and J.-J. Han (2021). "Learning Generalized Intersection Over Union for Dense Pixelwise Prediction". In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by M. Meila and T. Zhang. Vol. 139. Proceedings of Machine Learning Research. 18–24 Jul. PMLR, pp. 12198–12207.

Zhao, W., C. Persello, and A. Stein (2022). "Extracting planar roof structures from very high resolution images using graph neural networks". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 187. PII: S092427162200065X, pp. 34–45. DOI: 10.1016/j.isprsjprs.2022.02.022.

Appendices

A Records of hyperparameter tuning

Table 10: This table gives an overview of most model runs that were performed during hyperparameter tuning, along with the most important settings. The column TEST IOU contains the IoU as reported by the Python library Segmentation Models, corresponding to a macro/macro IoU with a class assignment threshold of 0.5, setting IoU for classes whose absence was predicted correctly to 1, and without weighting by class frequency. RGS refers to RandomGridShuffle augmentation.

ID	NAME	DATASET	LOSS FUNCTION	LR	BACKBONE	BATCH SIZE	EPOCHS	TEST IOU	NOTE	RGS SETTING
1	21-12-20_pvbackend	wb_m	categorical focal jaccard loss	1.00E-04	resnet152		40	0.63179		
2	21-12-20_wbg_v4	wb_a	categorical focal jaccard loss	1.00E-04	resnet152		40	0.7185		
3	22-01-02_pvbackend	wb_m	categorical focal loss	1.00E-04	resnet152	8	40	0.76425		
4	22-01-02_wbg_v4	wb_a	categorical focal loss	1.00E-04	resnet152	8	40	0.79099		
5	22-01-03_pvbackend	wb_m	dice loss (no class weights)	1.00E-04	resnet152	8	40	0.72858		
6	22-01-04_pvbackend	wb_m	categorical focal + dice loss	1.00E-04	resnet152	8	40	0.72261		
7	22-01-04_wbg_v4	wb_a	dice loss (no class weights)	1.00E-04	resnet152	8	40	0.7512		
8	22-01-05_pvbackend	wb_m	categorical focal loss	1.00E-04	resnet152	8	40	0.74652	No shift/scale/crop	
9	22-01-05_pvbackend_2	wb_m	categorical focal loss	1.00E-04	vgg19	16	40	0.68843		
10	22-01-07_pvbackend	wb_m	categorical focal loss	1.00E-03	vgg19	16	40	0.69548		
11	22-01-07_pvbackend_2	wb_m	categorical focal loss	1.00E-05	vgg19	16	80	0.75385	Continuation of 9	
12	22-01-09_pvbackend	wb_m	categorical focal loss	1.00E-04	inceptionresnetv2	14	40	0.74918		
13	22-01-09_pvbackend_2	wb_m	categorical focal loss	1.00E-05	inceptionresnetv2	14	20	0.75021	Continuation of 12	
14	22-01-09_pvbackend_3	wb_m	categorical focal loss	1.00E-04	efficientnetb4	6	40	0.75239		
15	22-01-09_pvbackend_4	wb_m	categorical focal loss	1.00E-04	efficientnetb6	4	40	0.74293		
16	22-01-12_pvbackend	wb_m	categorical focal loss	1.00E-04	inceptionv3	22	40	0.74347		
17	22-01-12_pvbackend_2	wb_m	categorical focal loss	1.00E-04	inceptionv3	20	20	0.74356	Continuation of 16	
18	22-01-13_pvbackend	wb_m	dice loss (with class weights)	1.00E-04	resnet152	10	40	0.70354		
19	22-01-13_pvbackend_2	wb_m	categorical cross entropy loss	1.00E-04	resnet152	8	40	0.77013		
20	22-01-15_wbg_v4	wb_a	categorical cross entropy loss	1.00E-04	resnet152	8	40	0.7841		
21	22-01-15_pvbackend	wb_m	categorical focal loss	1.00E-04	resnet152	8	40	0.78026		
22	22-01-15_wbg_v4_2	wb_a	categorical focal loss	1.00E-04	resnet152	8	40	0.78544		
23	22-01-16_pvbackend	wb_m	categorical focal loss	1.00E-04	resnet152	8	40	0.7701	With RGS	2x2, p = 0.5
24	22-01-16_pvbackend_2	wb_m	categorical focal loss	1.00E-04	resnet152	8	40	0.78286	With RGS	2x2, 3x3, p = 0.5
25	22-01-16_pvbackend_3	wb_m	categorical focal loss	1.00E-04	resnet152	8	40	0.7804	With RGS	2x2, 3x3, 4x4, p = 0.75
26	22-01-17_pvbackend	wb_m	categorical focal loss	1.00E-04	resnet152	8	40	0.77528	With RGS	2x2, 3x3, p = 0.67
27	22-01-17_wbg_v4	wb_a	categorical focal loss	1.00E-04	resnet152	8	40	0.80199	With RGS	2x2, 3x3, p = 0.5
28	22-01-17_pvbackend_2	wb_m	categorical focal loss	1.00E-04	resnet152	8	40	0.77451	With RGS	2x2, 3x3, p = 0.5
29	22-01-17_wbg_v4_subset	wb_as	categorical focal loss	1.00E-04	resnet152	8	40	0.79023	With RGS	2x2, 3x3, p = 0.5
30	22-01-20_pvbackend	wb_m	categorical focal loss	1.00E-04	resnet152	8	40	0.74316	No augmentations	
31	22-01-22_pvbackend	wb_m	categorical focal loss	1.00E-04	resnet152	8	40	0.78172	RandomResizedCrop	
32	22-01-22_pvbackend_2	wb_m	categorical focal loss	1.00E-04	resnet152	8	25	0.77484	Continuation of 32	
33	22-01-22_pvbackend_3	wb_m	categorical focal loss	1.00E-04	resnet152	8	25	0.77103	Continuation of 33	
34	22-01-23_bavaria	bv_a	categorical focal loss	1.00E-04	resnet152	8	20	0.81551	With RGS; > 3 hours / epoch	2x2, 3x3, p = 0.5
35	22-02-08_pvbackend	wb_m	categorical focal loss	1.00E-04	resnet152	8	40	0.76881	With RGS	2x2, 3x3, p = 0.5
36	22-02-08_pvbackend_2	wb_m	categorical focal loss	1.00E-04	resnet152	8	40	0.77895		
37	22-02-13_pvb_wbg	wb_m+a	categorical focal loss	1.00E-04	resnet152	8	40	0.78959	With RGS	2x2, 3x3, p = 0.5
38	22-02-13_pvb_wbg_2	wb_m+a	categorical focal loss	1.00E-04	resnet152	8	40	0.76348	RandomResizedCrop	
39	22-03-12_bavaria	bv_a	categorical focal loss	1.00E-04	resnet152	8	20	0.81037	> 3 hours / epoch	
40	22-03-16_pv_areas_wbg_a	wb_a@m	categorical focal loss	1.00E-04	resnet152	8	40	0.77712		
41	22-03-16_pv_areas_wbg_a_2	wb_a@m	categorical focal loss	1.00E-04	resnet152	8	40	0.7863	With RGS	2x2, 3x3, p = 0.5
42	22-03-25_pvbackend	wb_m	jaccard loss	1.00E-04	resnet152	8	40	0.69076	With RGS	2x2, 3x3, p = 0.5

B Plots of buildings and data splits

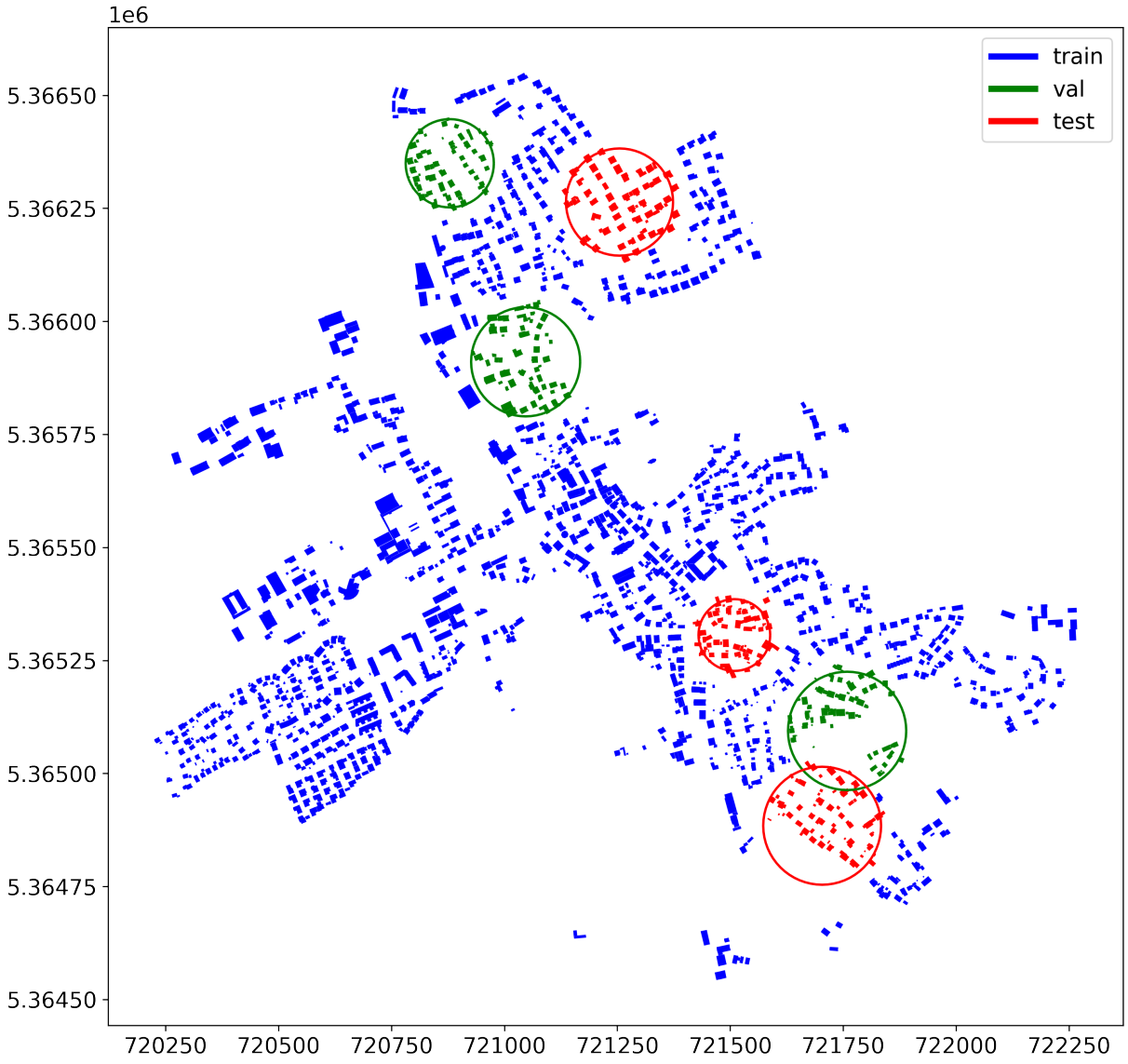


Figure 10: Data split of the datasets *wb_m* and *wb_a@m* (identical number, location, and size of training samples) with areas containg training (train), validation (val), and test data.

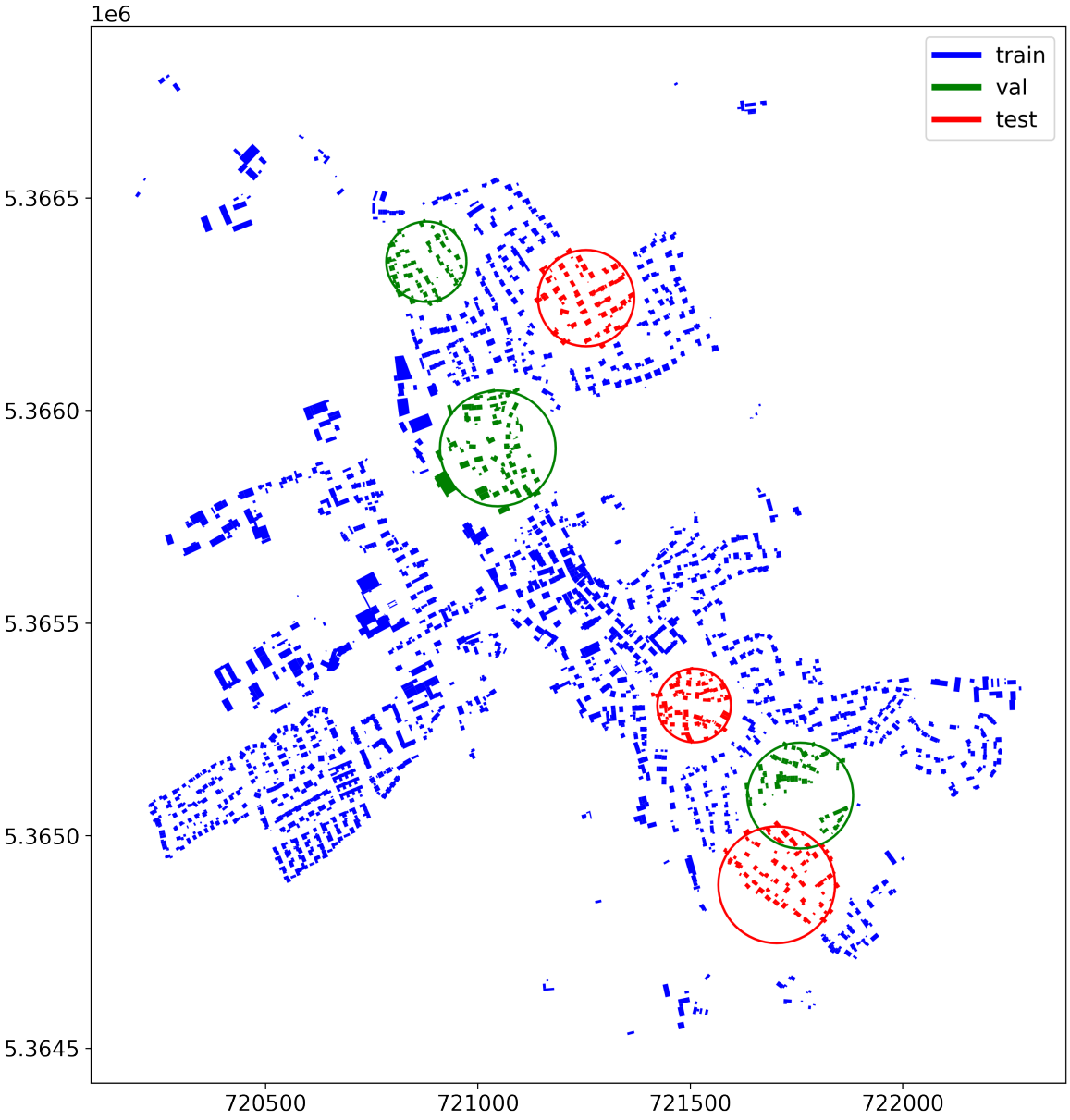


Figure 11: Data split of the dataset *wb_a* with areas containg training (train), validation (val), and test data.

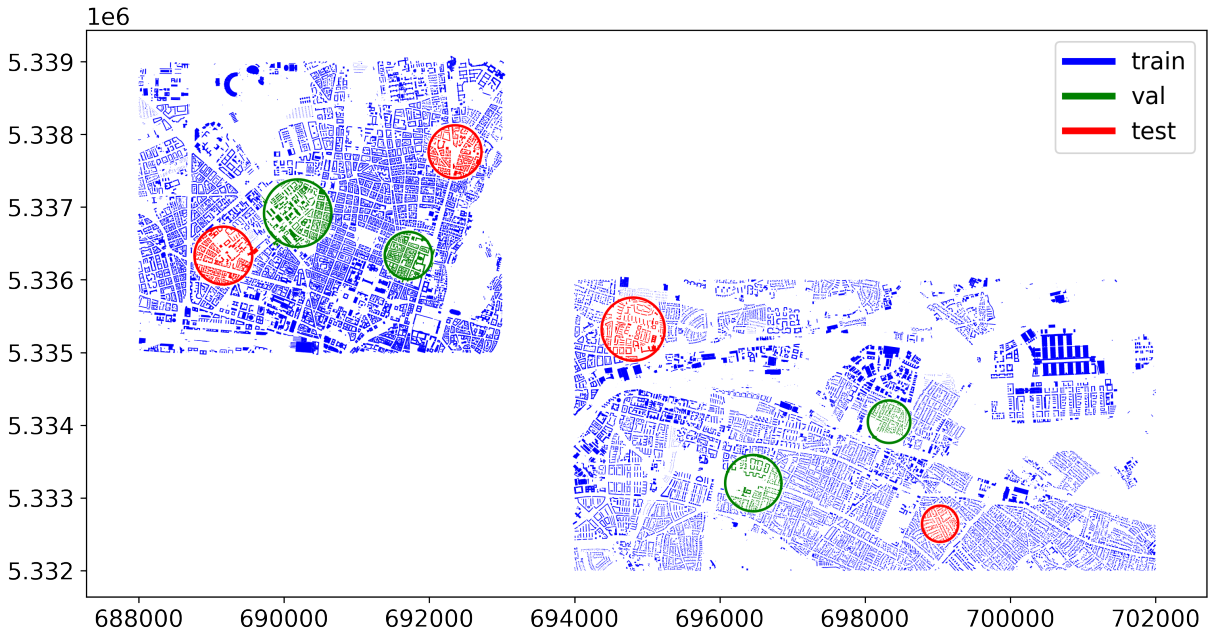


Figure 12: Data split of the Munich subregion of the dataset *bv_a* with areas containg training (train), validation (val), and test data.

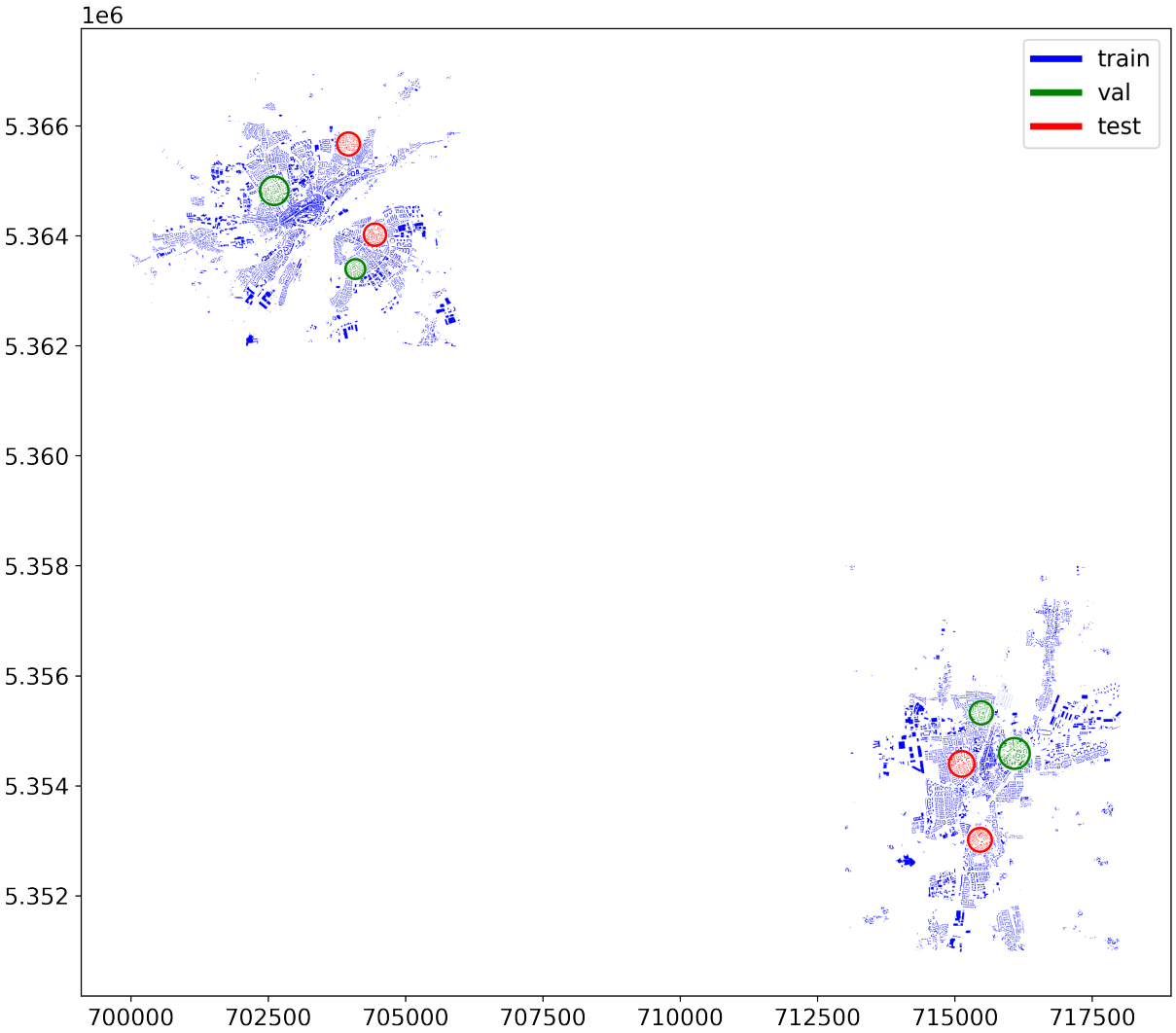


Figure 13: Data split of the subregion Erding/Freising of the dataset *bv_a* with areas containg training (train), validation (val), and test data.

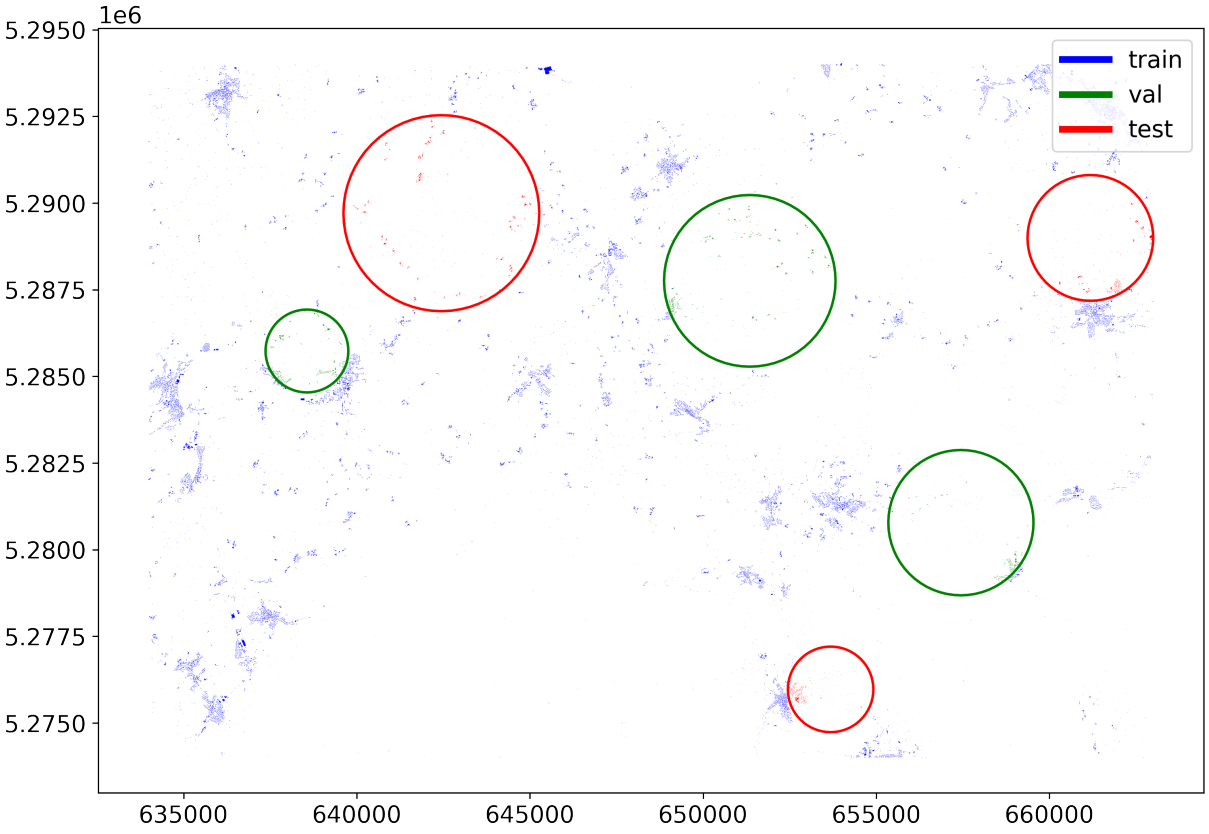


Figure 14: Data split of the rural subregion of the dataset *bv_a* with areas containg training (train), validation (val), and test data.

C A guide to computation of intersection over union

C.1 Image-level micro

Does not distinguish between classes. Instead, treats all classes equally by summing up true positives (TP), false positives (FP), and false negatives (FN) of all classes in the image and deriving IoU from these. More frequent or predominant classes will therefore have a stronger influence on the result.

$$TP_{image} = \sum_{classes} TP_{class,image} \tag{C.1}$$

$$FP_{image} = \sum_{classes} FP_{class,image} \tag{C.2}$$

$$FN_{image} = \sum_{classes} FN_{class,image} \tag{C.3}$$

$$IoU_{image} = \frac{TP_{image}}{TP_{image} + FP_{image} + FN_{image}} \tag{C.4}$$

C.2 Image-level macro

Distinguishes between classes: Computes IoU for each class in the image, then takes the average of these. Some distinctions described below.

$$IoU_{class,image} = \frac{TP_{class,image}}{TP_{class,image} + FP_{class,image} + FN_{class,image}} \tag{C.5}$$

Two distinctions:

1. If intersection and union (numerator and denominator) are zero, $IoU_{class,image}$ can either be set to 1 (rewarding the correct prediction of the class's absence) or to NaN (then being ignored in computing the image IoU by averaging all class IoU values). The former has the disadvantage that it reduces the meaningfulness of the IoU with regard to the model's performance in outlining the classes that are actually present in the image; the latter has the disadvantage that correct prediction of a class's absence is not reflected in the IoU.
2. Unweighted vs. weighted (by support) average.

Unweighted average of image-level class-wise IoU values:

$$IoU_{image} = \frac{\sum_{classes} IoU_{class,image}}{n_{classes}} \tag{C.6}$$

Weighted average of image-level class-wise IoU values, using image-level class support (true frequency) as weights:

$$support_{class,image} = TP_{class,image} + FN_{class,image} \tag{C.7}$$

$$IoU_{image} = \frac{\sum_{classes} IoU_{class,image} \cdot support_{class,image}}{n_{classes}} \tag{C.8}$$

C.3 Dataset-level micro

Does not distinguish between images. Either uses dataset-level class-wise IoU values (macro/micro) or uses dataset-level total TP, FP, and FN (micro/micro).

Based on image-level macro: Dataset-level macro/micro. Distinguishes between classes: Computes IoU for each class in the dataset, then takes the average of these. Some distinctions described below.

$$TP_{class,dataset} = \sum_{images} TP_{class,image} \tag{C.9}$$

$$FP_{class,dataset} = \sum_{images} FP_{class,image} \tag{C.10}$$

$$FN_{class,dataset} = \sum_{images} FN_{class,image} \tag{C.11}$$

$$IoU_{class,dataset} = \frac{TP_{class,dataset}}{TP_{class,dataset} + FP_{class,dataset} + FN_{class,dataset}} \tag{C.12}$$

Distinction: Unweighted vs. weighted (by support) average. Unweighted average of dataset-wide class-wise IoU values:

$$IoU_{dataset} = \frac{\sum_{classes} IoU_{class,dataset}}{n_{classes}} \tag{C.13}$$

Weighted average of dataset-wide class-wise IoU values, using dataset-wide class support (true frequency) as weights:

$$support_{class,dataset} = TP_{class,dataset} + FN_{class,dataset} \tag{C.14}$$

$$IoU_{dataset} = \frac{\sum_{classes} IoU_{class,dataset} \cdot support_{class,dataset}}{n_{classes}} \tag{C.15}$$

Based on image-level micro: Dataset-level micro/micro. Does not distinguish between classes nor images. Instead, treats all classes and images equally by summing up true positives (TP), false positives (FP), and false negatives (FN) of all images in the dataset and deriving IoU from these.

$$TP_{dataset} = \sum_{images} TP_{image} \tag{C.16}$$

$$FP_{dataset} = \sum_{images} FP_{image} \tag{C.17}$$

$$FN_{dataset} = \sum_{images} FN_{image} \tag{C.18}$$

$$IoU_{dataset} = \frac{TP_{dataset}}{TP_{dataset} + FP_{dataset} + FN_{dataset}} \tag{C.19}$$

C.4 Dataset-level macro

Distinguishes between images: Takes average of all image IoU values. Therefore, dataset-level micro/macro and macro/macro IoU are computed identically. Independently of whether the image-level IoU was computed following the micro or macro approach, the corresponding dataset-level macro IoU is computed as:

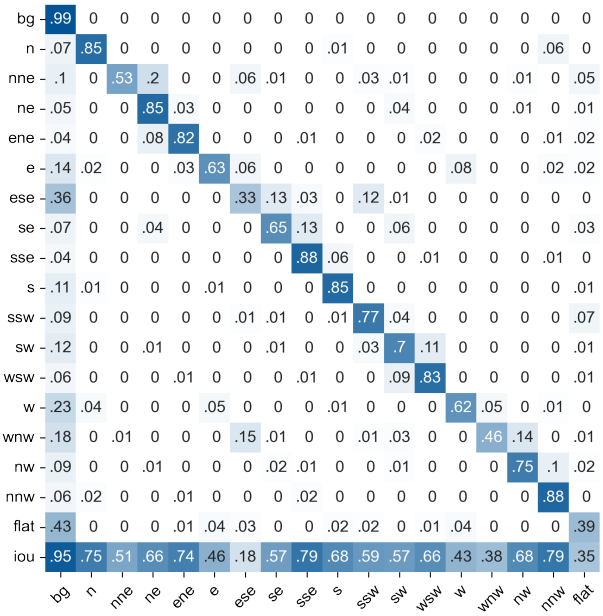
$$IoU_{dataset} = \frac{\sum_{images} IoU_{image}}{n_{images}} \tag{C.20}$$

D Complete results of model evaluation

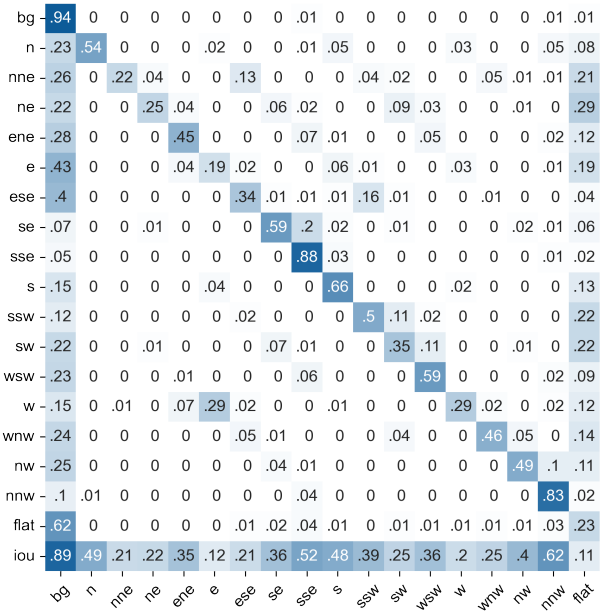
Table 11: Evaluation results of each scenario's model on each test dataset. Intersection over union and F1-score as reported by Python library Segmentation Models, and IoU values computed from argmax model output following the micro/macro, macro/macro, macro(weighted)/macro, micro/micro, macro/micro, and macro/micro(weighted) computation approaches.

Model	Dataset	sm_iou	sm_f1	mi_ma	ma_ma	ma_w_ma	mi_mi	ma_mi	ma_mi_w
wb_m	wb_m	0.783	0.811	0.863	0.492	0.878	0.859	0.596	0.868
wb_m	wb_a	0.694	0.723	0.741	0.312	0.781	0.737	0.342	0.773
wb_m	bv_a	0.690	0.713	0.686	0.260	0.711	0.670	0.300	0.693
wb_m	wb_a@m	0.699	0.728	0.747	0.313	0.784	0.743	0.359	0.777
wb_a	wb_m	0.710	0.739	0.757	0.344	0.761	0.752	0.385	0.755
wb_a	wb_a	0.802	0.833	0.861	0.535	0.872	0.859	0.619	0.865
wb_a	bv_a	0.742	0.769	0.750	0.372	0.758	0.732	0.422	0.738
wb_a	wb_a@m	0.796	0.828	0.860	0.521	0.870	0.857	0.619	0.864
bv_a	wb_m	0.760	0.791	0.802	0.464	0.811	0.798	0.511	0.803
bv_a	wb_a	0.828	0.858	0.877	0.599	0.889	0.875	0.695	0.883
bv_a	bv_a	0.816	0.843	0.839	0.543	0.854	0.832	0.626	0.843
bv_a	wb_a@m	0.828	0.858	0.877	0.592	0.889	0.875	0.691	0.883
wb_ma	wb_m	0.773	0.803	0.847	0.511	0.857	0.843	0.579	0.848
wb_ma	wb_a	0.800	0.833	0.856	0.531	0.867	0.854	0.619	0.862
wb_ma	bv_a	0.753	0.781	0.758	0.394	0.769	0.742	0.455	0.751
wb_ma	wb_a@m	0.790	0.823	0.852	0.513	0.864	0.850	0.606	0.858
wb_a@m	wb_m	0.698	0.727	0.744	0.327	0.744	0.738	0.338	0.738
wb_a@m	wb_a	0.792	0.823	0.856	0.504	0.867	0.854	0.593	0.861
wb_a@m	bv_a	0.723	0.749	0.744	0.342	0.752	0.726	0.403	0.731
wb_a@m	wb_a@m	0.786	0.818	0.854	0.485	0.866	0.852	0.597	0.859

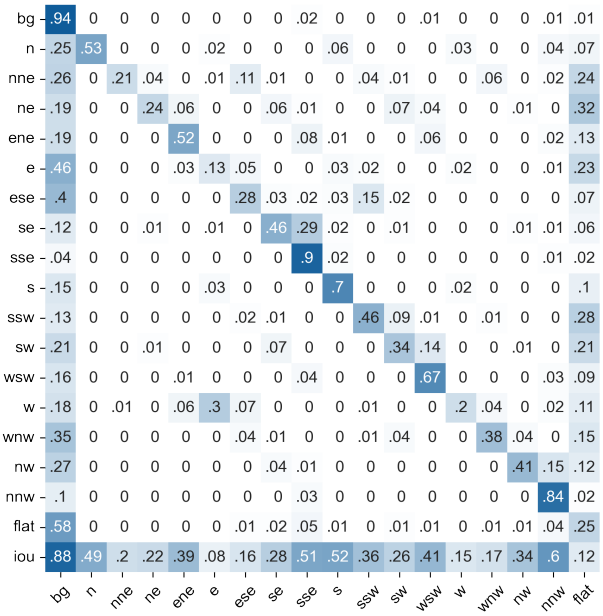
E Confusion matrices



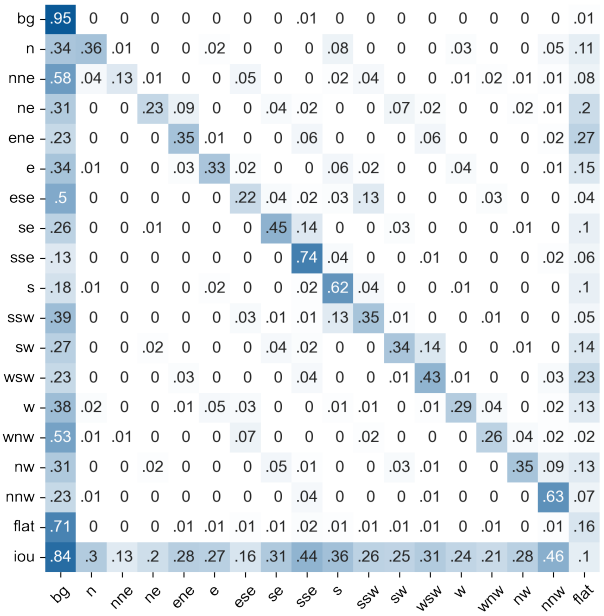
wb_m → *wb_m* (.863, .596)



wb_m → *wb_a@m* (.747, .359)

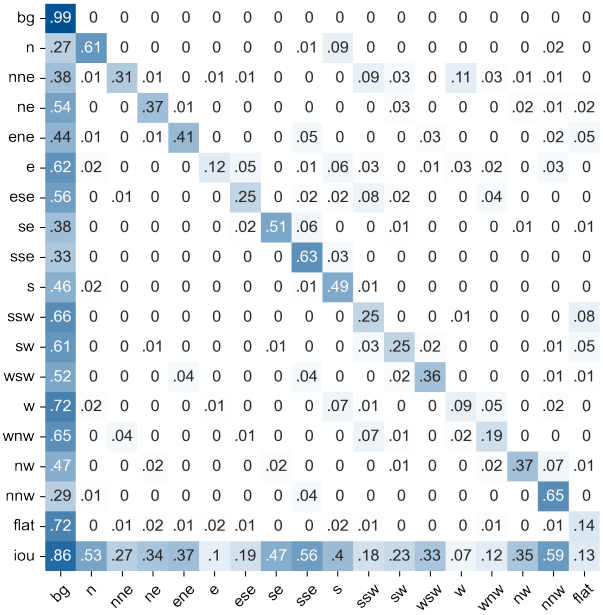


wb_m → *wb_a* (.741, .342)

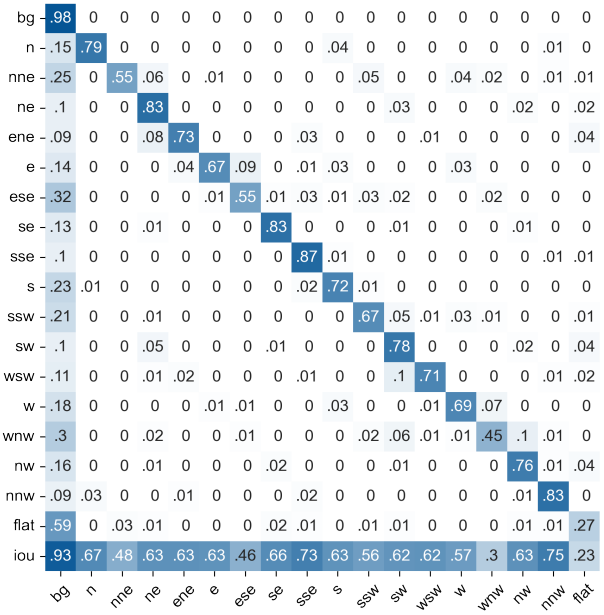


wb_m → *bv_a* (.686, .300)

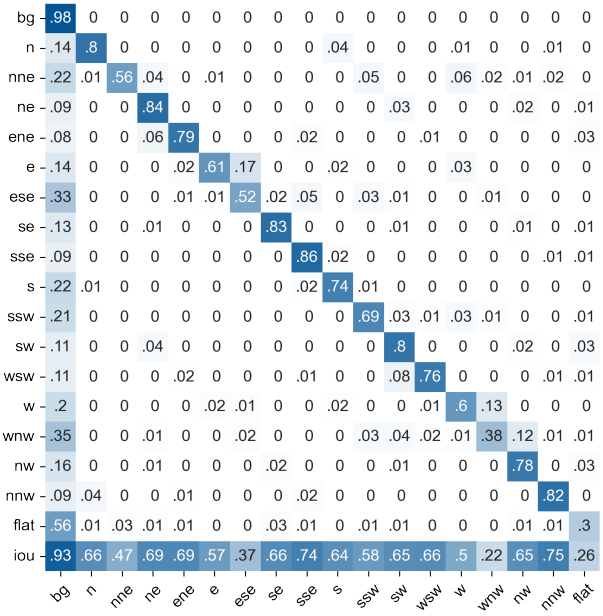
Figure 15: Confusion matrices of the model *wb_m* on four test sets. Rows are ground truth labels, columns are predictions. Rows are normalized to total number of predictions, thus, sum up to 1. Last row shows IoU of the corresponding class on the dataset level (macro/micro). Numbers in brackets next to identifiers are micro/macro and unweighted macro/micro IoU values.



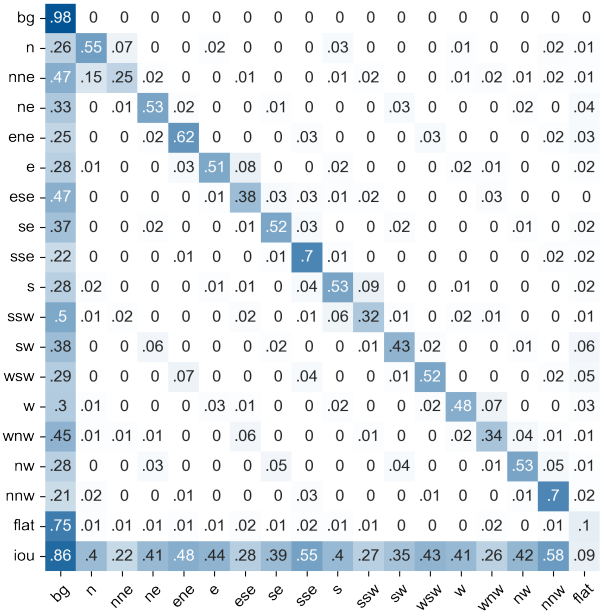
wb_a@m → *wb_m* (.744, .338)



wb_a@m → *wb_a@m* (.854, .597)

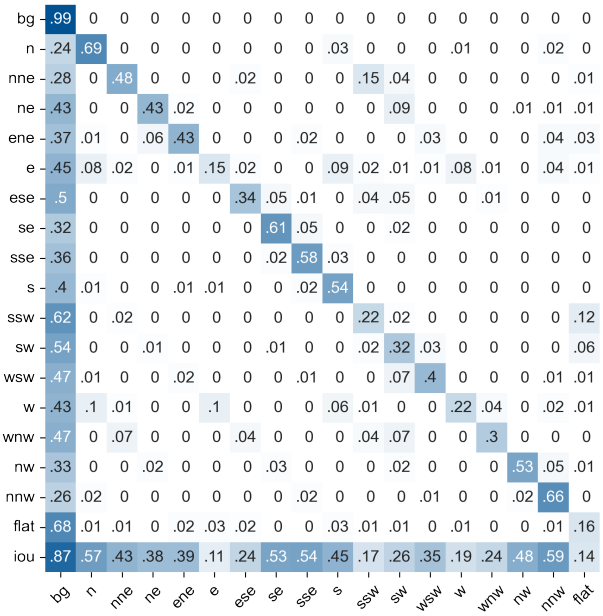


wb_a@m → *wb_a* (.856, .593)

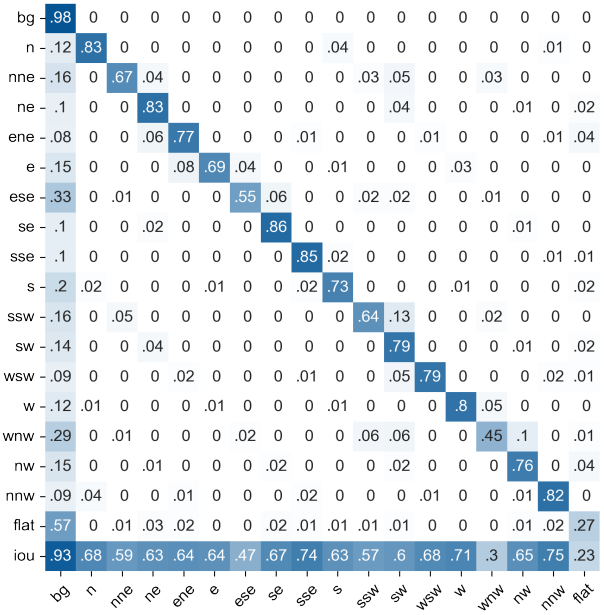


wb_a@m → *bv_a* (.744, .403)

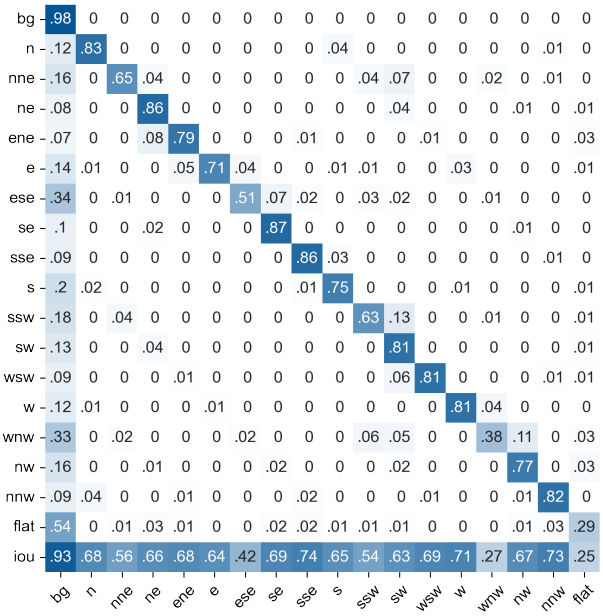
Figure 16: Confusion matrices of the model *wb_a@m* on four test sets. Rows are ground truth labels, columns are predictions. Rows are normalized to total number of predictions, thus, sum up to 1. Last row shows IoU of the corresponding class on the dataset level (macro/micro). Numbers in brackets next to identifiers are micro/macro and unweighted macro/micro IoU values.



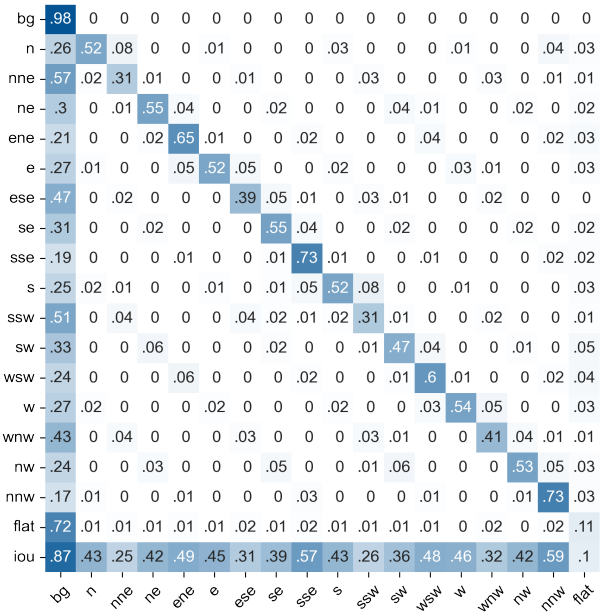
wb_a → *wb_m* (.757, .385)



wb_a → *wb_a@m* (.860, .619)

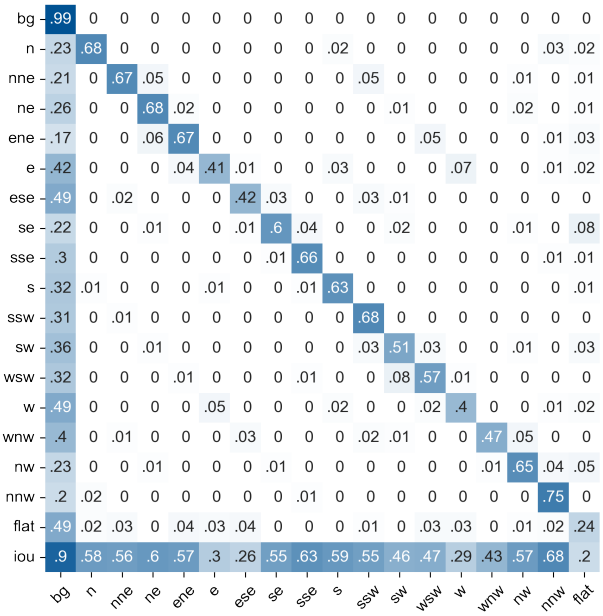


wb_a → *wb_a* (.861, .619)

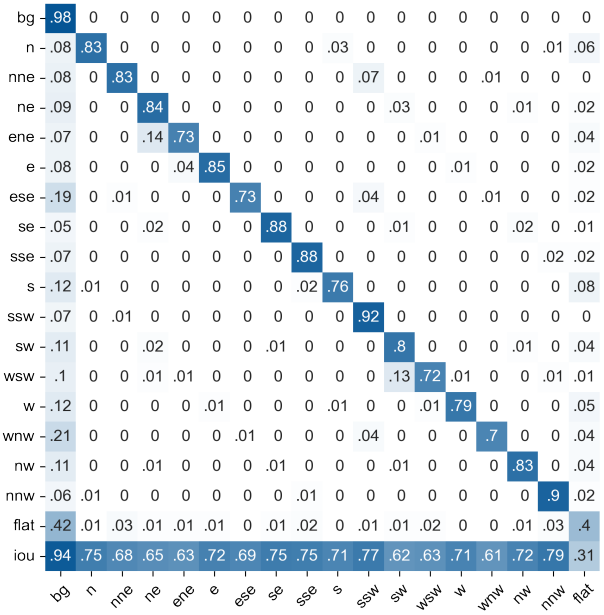


wb_a → *bv_a* (.750, .422)

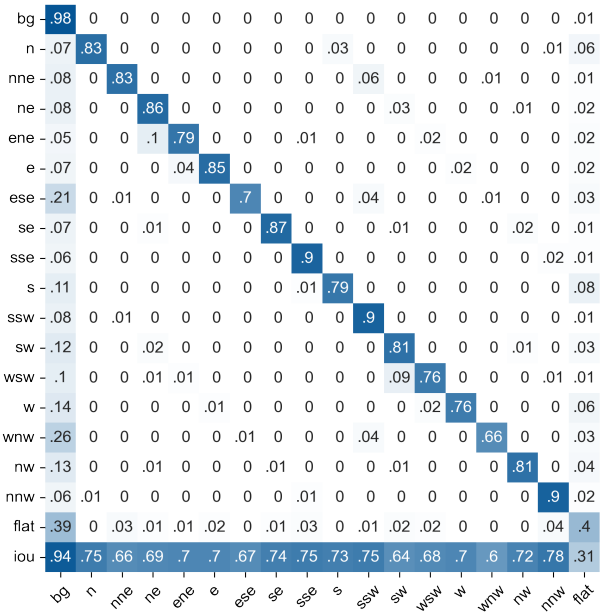
Figure 17: Confusion matrices of the model *wb_a* on four test sets. Rows are ground truth labels, columns are predictions. Rows are normalized to total number of predictions, thus, sum up to 1. Last row shows IoU of the corresponding class on the dataset level (macro/micro). Numbers in brackets next to identifiers are micro/macro and unweighted macro/micro IoU values.



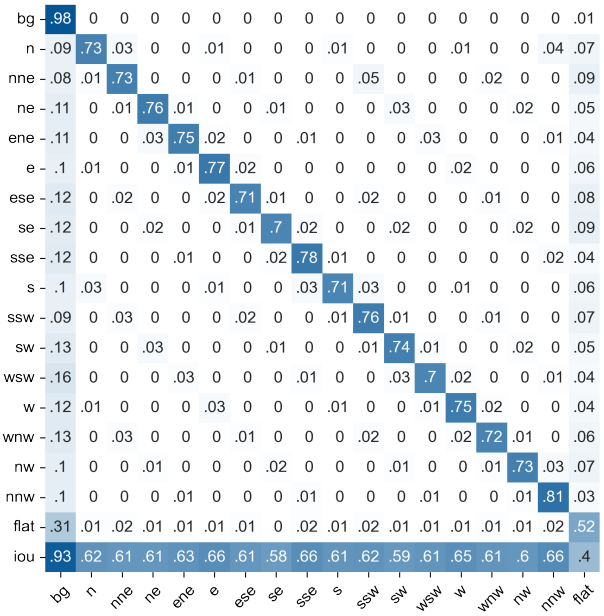
bv_a → *wb_m* (.802, .511)



bv_a → *wb_a@m* (.877, .691)

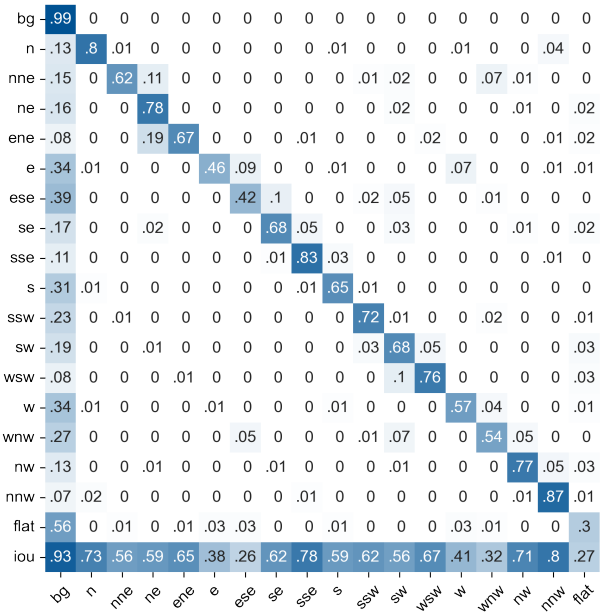


bv_a → *wb_a* (.877, .695)

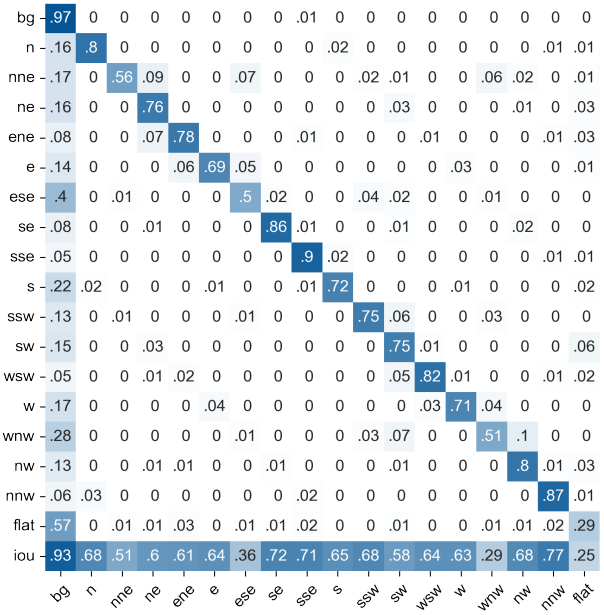


bv_a → *bv_a* (.839, .626)

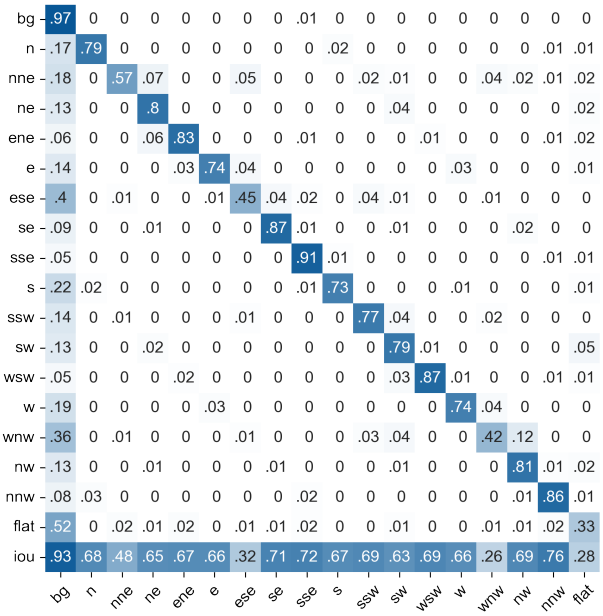
Figure 18: Confusion matrices of the model *bv_a* on four test sets. Rows are ground truth labels, columns are predictions. Rows are normalized to total number of predictions, thus, sum up to 1. Last row shows IoU of the corresponding class on the dataset level (macro/micro). Numbers in brackets next to identifiers are micro/macro and unweighted macro/micro IoU values.



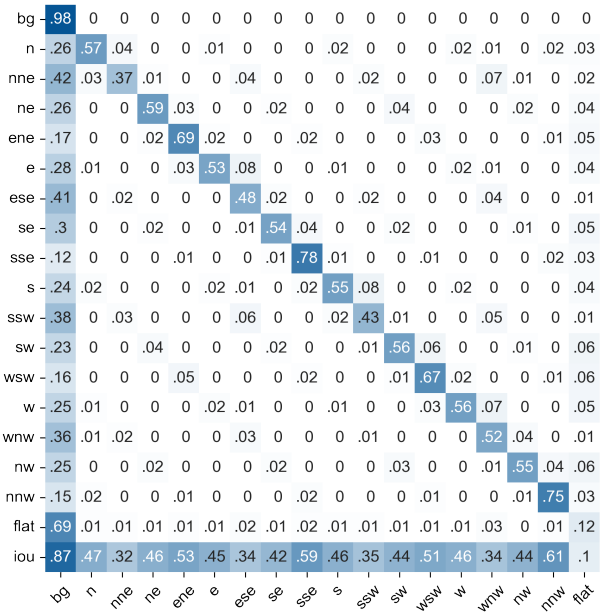
wb_m+a → *wb_m* (.847, .579)



wb_m+a → *wb_a@m* (.852, .606)



wb_m+a → *wb_a* (.856, .619)



wb_m+a → *bv_a* (.758, .455)

Figure 19: Confusion matrices of the model *wb_m+a* on four test sets. Rows are ground truth labels, columns are predictions. Rows are normalized to total number of predictions, thus, sum up to 1. Last row shows IoU of the corresponding class on the dataset level (macro/micro). Numbers in brackets next to identifiers are micro/macro and unweighted macro/micro IoU values.