Developing an ML Model for Orthophoto Segmentation Using Automatically Generated Labels

Kevin Kocon[1,2][a] and Michel Krämer[1,2][b]
[1]*Fraunhofer Institute for Computer Graphics Research IGD, Fraunhoferstr. 5, 64283 Darmstadt, Germany*
[2]*Technical University of Darmstadt, 64289 Darmstadt, Germany*

Keywords: Machine Learning, Remote Sensing, Image Segmentation, Scarce Labels, Geospatial Data.

Abstract: In this paper, we develop a machine learning (ML) model for the segmentation of orthophotos in urban areas throughout Germany. We present our ML pipeline as well as the results of evaluating the model. The main challenge is that labels required for training do not exist for the classes we want to identify in the available aerial imagery. Manual labelling is too time-consuming and the commonly used approach of generating synthetic training data has the disadvantage that artificial data cannot fully represent real-world characteristics. Instead, we generate our labels automatically based on official cadastral information, as well as results from an existing convolutional neural network (CNN) segmenting 3D point clouds from car-based terrestrial mobile mapping. This is a novel approach that, to the best of our knowledge, has not been explored before. Our evaluation shows that the model reliably identifies all classes throughout Germany, regardless of region, season, or camera. Notably, the model correctly classifies surfaces even in areas where the training labels contained no information. The predictions are thus visually closer to reality than the generated labels themselves. Our approach therefore represents a viable alternative in cases where there are no labels available for training. The resulting model is deployed in production by Deutsche Telekom for fiber Internet rollout planning.

1 INTRODUCTION

Artificial intelligence (AI) is becoming increasingly popular and nowadays provides the basis for a wide range of computer vision applications. Some important areas are medical image analysis (Altman, 2017), environmental perception in autonomous driving (Gupta et al., 2021), as well as the classification of aerial images of urban and forest areas (Li et al., 2024; Kocon et al., 2022). Thanks to their superior performance, convolutional neural networks (CNNs) have become one of the leading modern approaches for automatic and reliable recognition of features learned during training.

Training data consisting of accurate and complete labels is key to the development of not only CNNs but machine learning (ML) models in general. Manual labeling is time-consuming, and is therefore often not done at all. This is underlined by the work of Mäkinen who identify data-related issues—specifically data messiness, scarcity, and limited accessibility—as leading obstacles (Mäkinen et al., 2021). One common approach to mitigate this is to automatically generate synthetic training data. However, such artificial data cannot fully represent the complex variability and patterns of the real world, and using it for training potentially reduces model generalizability.

The use case we focus on in this paper is the pixel-based classification of aerial imagery to identify various surfaces, such as asphalt and paving, as well as to detect buildings (see Figure 1). For this, we aim to train an ML model, but the problem is that there are no semantic labels for the aerial images.

To address this, we could generate synthetic labels, but this would lead to the issues described above. In this work, we want to show a different approach instead: We combine the segmentation results from an existing CNN with official cadastral data. The existing CNN works on 3D point clouds recorded with a car-based terrestrial mobile mapping system. Its main purpose is to identify various surface materials. So, in general, it covers most of our use case, although there is no building information, and it works on entirely different input data.

[a] https://orcid.org/0000-0003-4799-1967
[b] https://orcid.org/0000-0003-2775-5844

Kocon, K. and Krämer, M.
Developing an ML Model for Orthophoto Segmentation Using Automatically Generated Labels.
DOI: 10.5220/0015131500004091

In *Proceedings of the 15th International Conference on Data Science, Technology and Applications (DATA 2026) - Volume 1*, pages 555-562
ISBN: 978-989-758-854-9; ISSN: 2184-285X

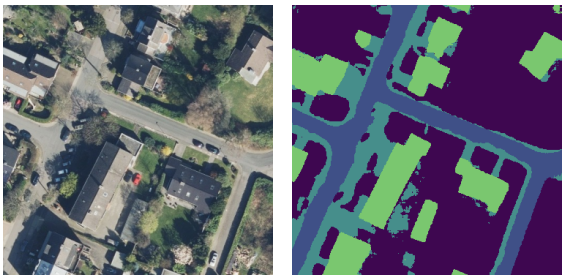


Figure 1: Example of an aerial image (left) and the results of pixel-based classification (right).

Nevertheless, combing its results with the cadastral data allows us to generate training data that reflects reality. With this, we can train a new ML model that works on aerial images.

Our evaluation shows that this new model produces results with a very good quality and high generalizability, even covering areas that the existing CNN is not able to classify. Our model is not limited to local regions or seasons. It works throughout Germany and is used in production by Deutsche Telekom, the largest telecommunications provider in Europe, to classify surfaces during the planning process for the rollout of high-speed fiber Internet (Krämer et al., 2024). There, it complements the existing CNN, and is applied in cases where the mobile mapping car has not collected 3D point cloud data (either because the car was not deployed to the area that should be classified yet or because a certain part of the area is unreachable).

2 RELATED WORK

As mentioned before, training data is key to the development of ML models, but the lack thereof is a critical bottleneck and one of the biggest challenges in AI (Mäkinen et al., 2021). There are various approaches to counteract this problem. Well-known methods for dealing with a limited amount of labeled data are, for example, Pseudo Labeling (Ran et al., 2024), Active Learning (Tharwat and Schenck, 2023), or Transfer Learning (Hutchinson et al., 2017). In the cases of Pseudo Labeling and Active Learning, the model is first pre-trained with a limited amount of data. With Pseudo Labeling, this model is then used to generate predictions, which are subsequently added to the training dataset as labels. In Active Learning, data is selected based on a sampling strategy (e.g., uncertainties), which is then usually labeled by a human annotator and then used for further training. Transfer Learning includes several strategies. A widely used approach is fine-tuning, which is typically performed in two phases: first, pre-training a base model using labeled data from a related source, and second, adapting the model using the limited labeled data available for the target domain. However, all of these methods require a certain amount of labeled data, which, as mentioned earlier, does not exist in our use case.

If no labeled data is available, there are still various options. If the data exhibits sufficiently obvious patterns and structures, clustering methods (Bock, 2007) can be used. This allows the data to be classified directly. The use of self-supervised methods (Chen et al., 2020) can also be considered. Modern self-supervised methods are already capable of recognizing more complex patterns.

However, for the training of CNNs, labeled data is essential. One option is to generate synthetic data, i.e. artificial datasets. Generative adversarial networks (GANs) are commonly used for this purpose (Lekavičius and Gružauskas, 2024). Simulations and 3D scene rendering also enable the generation of synthetic training data, as demonstrated by Reyes et al. (Reyes et al., 2023). Because such datasets are artificial, they do not fully capture the complex variations and patterns of real-world data, which can reduce model generalizability. Consequently, there is no guarantee that models trained on synthetic data will perform well on real data.

Another common approach for generating training data is to derive labels from other data sources. Our literature review showed that using OpenStreetMap as a basis for generating labels is very popular in land use classification. Examples include the work of Radhakrishnan et al. (Motlagh et al., 2021), Parekh et al. (Parekh et al., 2021), and Zhao et al. (Zhao et al., 2025). Radhakrishnan et al. present a framework that semi-automatically labels satellite images based on OpenStreetMap. By training a classifier to distinguish between construction and non-construction sites, they achieve good results with this approach. Parekh et al. predict impervious surfaces from Landsat 8 imagery. Using OpenStreetMap, they programmatically generate large-scale training and evaluation datasets, thus eliminating the need for manual labeling. Zhao et al. detect buildings based on labels extracted from OpenStreetMap. They also present convincing results.

In addition to OpenStreetMap, other data sources can also be used. For example, public information, as in the work of Voelsen et al. (Voelsen et al., 2020). Using the German landscape model ATKIS, they generate labels and train their model. Albrecht et al. (Albrecht et al., 2021), instead, use 3D point clouds as a basis for generating labels. They implement a rule-based approach to extract labels for land use classification from the 3D point clouds.

Looking at these works, it becomes clear that this approach is well suited. The authors have shown that

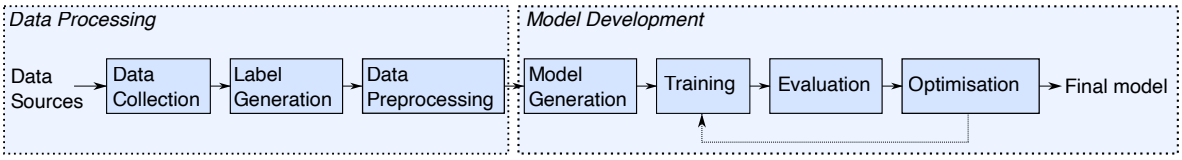


Figure 2: Abstract representation of the ML pipeline.

deriving labels from other data sources can be used to generate good training data, but specific cases, such as distinguishing between asphalt and pavement, as in our case, has not been implemented yet. Furthermore, the existing works have not yet tried to combine data sources such as the real estate cadastre with the segmentation results of an existing ML model.

3 MACHINE LEARNING PIPELINE

The model we aim to train should be able to classify aerial images. The classes we want to identify are: *asphalt/concrete*, *pavement*, *unbound surface*, and *building*. The segmentation should deliver reliable results throughout Germany, regardless of external influences such as lighting conditions or seasons, as well as the various cameras used to take the images. In addition, the entire process must be time-efficient, as very large areas are to be segmented.

To meet these requirements, we have decided to train a CNN, more precisely a U-Net (Ronneberger et al., 2015). CNNs are capable of learning complex patterns. This is key in our case, as aerial images often have a relatively low resolution (e.g. 20 cm per pixel) and fine differences must still be recognizable to differentiate between, for example, asphalt and pavement. CNNs are also known to generalize external influences very well. Another aspect is that they can be highly parallelized on the GPU, making them very efficient.

Figure 2 shows an abstract representation of our ML pipeline. It consists of two overarching parts: data processing and model development. For the sake of completeness, it should be mentioned that a full pipeline would also include model deployment as a final step. This step represents the part in which the developed model is transferred to the production environment. Since our focus is on the development of the model itself, model deployment would go beyond the scope of this paper and is therefore omitted.

The first part of the pipeline covers all steps of data processing, resulting in finished training data that will be transferred to the model development. This means that the data is first downloaded and collected from the various sources. In the next step, labels are generated. After that, a data preprocessing step prepares the data for the training. This includes, for example, converting it to the correct resolution. The entire data processing part of our pipeline is discussed in detail in Section 4.

The second part of the ML pipeline is model development. It begins with model generation, i.e. the selection and implementation of the model, including tuning of hyperparameters. This is followed by model training. After training, the model is evaluated and optimized. This is often an iterative process, which is why further training may be necessary. The result is the final model. The model development part of our ML pipeline is described in Section 5.

4 DATA PROCESSING

Our pipeline starts with the processing of input data. The goal is to collect all necessary data and to transform it into a format that can be used for the training. As shown in Figure 2, the data processing part consists of Data Collection, Label Generation, and Data Preprocessing. In the following, we go through each step and explain what we had to do in preparation for the training of our model.

4.1 Data Collection

Our main input data source was aerial imagery covering the whole of Germany. It was provided by the Federal Agency for Cartography and Geodesy (BKG) through a Web Map Service (WMS), a geospatial data exchange interface standardized by the Open Geospatial Consortium (OGC).

The imagery had a resolution of 20 cm per pixel and was provided in full color (RGB) and infrared (IR). Using both types was beneficial for training, although in some parts of Germany, the images were taken on different dates. This can be seen in Figure 3 where cars appear in different places and shadows are cast into different directions. This was, however, not a problem. Our results presented in Section 5.2 show that the ML model was capable of abstracting differences like these.

The aerial images were not labelled, which means, used alone, they were unsuitable for training. We there-

(a) (b)

Figure 3: Example RGB (left) and IR (right) pair of aerial images used as input for our training. The imagery was provided by the German Federal Agency for Cartography and Geodesy (BKG).

fore combined two other data sources to obtain labels: (a) Results of an already available convolutional neural network (CNN) for the classification of surfaces in 3D point clouds acquired by car-based terrestrial mobile mapping (Reiterer et al., 2020), and (b) information from the German Real Estate Cadastre Information System (ALKIS) as a fallback for areas where no results from the existing surface classification were available.

To improve the robustness of our model, we selected a total of about 60,000 image tiles with a size of 100×100 m (512×512 pixels at a resolution of 20 cm per pixel) randomly throughout the whole of Germany, primarily focussing on inhabited areas, although no distinction between large cities and small settlements was made. This way, we were able to collect images with a strong variety in terms of lighting conditions and seasons. We set up an automated download script to fetch the images in RGB and IR format from the BKG WMS.

Our second data source contained the segmentation results of the existing CNN. They were vector-based (i.e. polygons) and stored in a proprietary database, which could be accessed through an OGC Web Feature Service. The same applied to our third data source, the cadastral data. Again, we set up an automated downloading script to fetch the polygons for the exact same 100×100 m tiles. The data was stored in the GeoJSON format. Based on these two data sources, we were then able to generate labels.

4.2 Label Generation

Figure 4 shows an example of the used data. As you can see in image (a), the segmentation results of the existing CNN were sometimes incomplete. In this particular case, the mobile mapping car was coming from the north and took a turn to the west. It did not record 3D point cloud data for the south-east part of the image and, therefore, no segmentation results were available. The cadastral data in image (b), on the other hand, was more complete. It also included buildings, which the existing CNN did not identify.

Note that the existing CNN was able to segment a range of surface materials. Due to its precision, we prioritized its segmentation results over the cadastral data. The combined labels are shown in image (c).

Due to the fact that the images we wanted to classify only had a resolution of 20 cm per pixel, it was hard to differentiate between certain kinds of surfaces, even with the human eye. For example, at such a resolution, both concrete and asphalt more or less look the same (i.e. like a gray area). We therefore grouped them into one label class. The same applied to large and small paving, as well as loose surface and gutter curb stone.

The individual labels from the existing CNN and those from the cadastral data were further mapped into the final classes we wanted our model to identify. Concrete/asphalt and streets became the class *asphalt/concrete*. Large and small paving as well as the paths from the cadastre became *paving*. Buildings stayed *buildings*, and loose surface and gutter curb stone became the more generic class *unbound surface*. For unlabeled pixels, we made up a class named *unknown*. An example of the final generated labels is shown in Figure 4 (d).

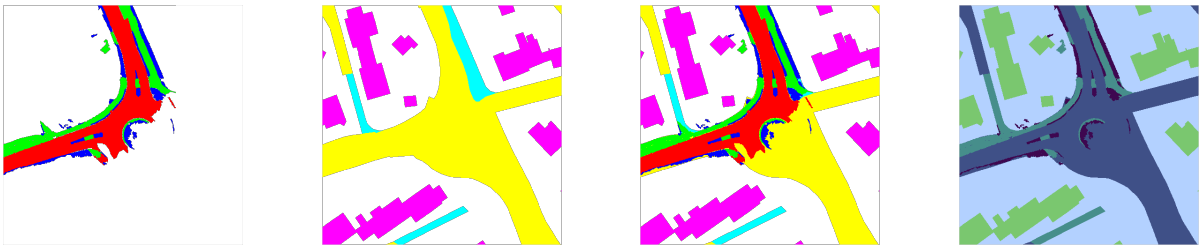
4.3 Data Preprocessing

We rasterized the vector labels into image tiles with a size of 512×512 pixels and a resolution of 20 cm per pixel. We cropped them so that they matched the downloaded aerial images exactly.

To simplify the training step, we stored the generated labels one-hot encoded in binary files.

Note that data preprocessing in an ML pipeline normally contains data augmentation, too. Here, additional image/label pairs are generated by transforming the existing ones. However, we decided to perform data augmentation during training in the batch generator (see Section 5.1). This approach has several advantages. First, it further increases the variance of the training data, as each image is newly augmented in each epoch. Second, since the augmentation is performed by the CPU in parallel with the training taking place on the GPU, no additional time is required. Furthermore, the augmented images do not have to be stored on the hard disk but can be transferred directly from the CPU to the GPU for training.

In summary, as mentioned earlier, we collected about 60,000 image pairs, consisting of aerial image tiles (RGB and IR) and rasterized labels. All in all, our training dataset had a total size of 142 GB.



(a) Results of existing CNN. (b) Cadastral data. (c) Combined labels. (d) Final label.

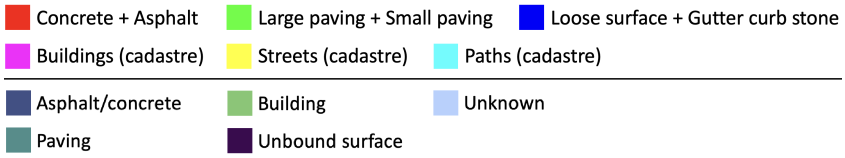
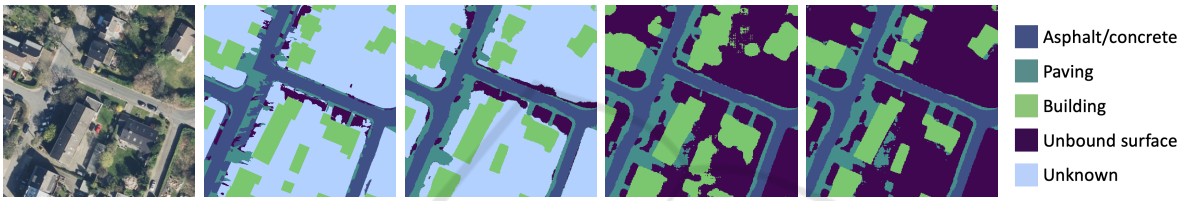


Figure 4: Rasterized source data (a)-(c) and resulting training label (d).



(a) (b) (c) (d) (e)

Figure 5: Example images of the optimization process. (a) RGB orthophoto. (b) labels. (c) prediction without post-processing. (d) prediction with the second most probable class instead of unknown. (e) final result with complete post-processing.

5 MODEL DEVELOPMENT

In this section, we present the model development steps of our pipeline. This part begins with the selection and implementation of a model. We have chosen a classic U-Net (Ronneberger et al., 2015) for our use case. Originally proposed for biomedical image segmentation, U-Net is a convolutional neural network whose encoder—decoder design with skip connections preserves spatial detail and supports accurate, efficient inference, making it also widely applicable beyond medicine. Building on our prior experience with U-Net for satellite-based forest-type classification (Kocon et al., 2022), we adopted the same architecture for this work.

For the implementation, we used the Python programming language and the Keras library (Chollet et al., 2015) with the Tensorflow backend (Google, 2015). As input, the U-Net received the previously generated images with a resolution of 512×512 pixels and 4 channels. The output was an array of probability distributions with a resolution of $512 \times 512 \times 5$.

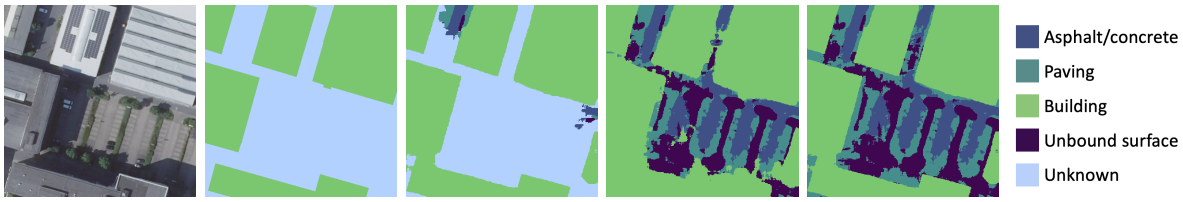
Before starting training, hyperparameter tuning was performed. For this purpose, we used the Hyperband algorithm (Li et al., 2018) from the Keras-Tuner library. The following hyperparameters were optimized in the search spaces shown. The values shown in bold correspond to the results of the tuning, which were then used for training.

- Learning rate: 0.1, 0.01, 0.001, **0.0001**, 0.00001, 0.000001
- Loss: Categorical CrossEntropy, **Sigmoid Focal CrossEntropy**, Jaccard Distance
- Optimizer: **Adam**, Gradient Descent
- Batch size: 4,8,**16**

5.1 Training

The U-Net was trained in 50 epochs. A batch generator was implemented for the training, which inherited from the class *tf.keras.utils.Sequence* and provided the data for training one batch. First, the generated training data—RGB images, IR images, and labels—were read in. Then, the data of one batch was augmented.

Data augmentation is a fundamental part of our model development. It increases the variance of the training data by transforming the original data. Augmenting increases the robustness of the model, which counteracts overfitting. This is especially important



(a) (b) (c) (d) (e)

Figure 6: Example images of the optimization process. (a) RGB orthophoto. (b) labels. (c) prediction without post-processing. (d) prediction with the second most probable class instead of unknown. (e) final result with complete post-processing.

for use cases like ours, where images from different sensors, locations, and seasons are used. We achieved the best results by generating seven additional augmented images for each image. To do this, we first performed randomly one or two of the following color space transformations for each image: multiply brightness, multiply saturation, gamma contrast, gaussian noise, gaussian blur, and enhance sharpness. Finally, one of the following geometric transformations was randomly selected and applied: horizontal and/or vertical flip as well as rotation.

5.2 Evaluation and Optimization

After the training, the results were evaluated. Different challenges appeared in the various iterations. In this section, we summarize the most important findings and proposed solutions.

The first challenge was to achieve scalability of the model across different regions and seasons throughout Germany. This has already been described in detail in the previous chapters. In summary, the decisive factors were, firstly, the large number of images from a wide variety of regions in Germany and, secondly, data augmentation, in particular color space transformation. The results of the first iterations without following these aspects showed poor scalability. This is also consistent with the findings of the works by Drory et al. (Drory et al., 2018) and Voelsen et al. (Voelsen et al., 2020). In particular, Drory et al. already describe that a large number of labels of inadequate quality can be considered random noise under certain conditions, which consequently does not have a negative effect on the results of the networks. Voelsen et al., meanwhile, discuss the relevance of including a wide variety of lighting conditions, seasons, and regions in the training data for the scalability of the model.

Figures 5 and 6 illustrate the next challenge. In both figures, images (a) show an RGB orthophoto and images (b) show the corresponding generated labels. Images (c) in each figure illustrate the prediction of the network after training without post-processing. Looking at the prediction in image 5 (c), it can be seen that this result is very similar to the generated label in image 5 (b). The buildings are recognized almost perfectly, and the asphalt/concrete and pavement classes do not even show any false interruptions. The *unbound surface* and the *unknown* classes are also recognized correctly in relation to the labels in image 5 (b). The same can be seen in image 6 (c). In relation to the labels we generated, our model has an overall accuracy of 83% and a one-hot mean intersection-over-union of 59%. However, this is where the next problem arises.

Since we generate our labels from predictions of point clouds generated from road surveys and ALKIS data, we only have information about the immediate surroundings of the road and the locations of the buildings. All other areas were labeled with the class *unknown*. This behavior has now also been trained into the U-Net. As a result, the predictions reflect the characteristics of the training data, but not reality. Looking at images 5 (a) and 5 (c), it can be seen that the paving and the unbound surface on the access paths and the properties themselves are not classified as such. This becomes even clearer in image 6 (c). The entire parking lot, including the unbound surface, is classified correctly in terms of the labels but does not reflect reality.

To counteract this behavior, we further process our predictions. Since the output of our network is the probability distribution of the respective classes for a pixel, we select the second most probable class in the case of the class *unknown*. These results can be seen in images 5 (d) and 6 (d). This creates further artifacts at first and results in a high proportion of false positives for the class *building*. This is understandable given that the probabilities for the class *building* are high at the edges of the buildings. To further counteract this behavior, we select the third most probable class in areas where *unknown* is the most probable class and *building* is the second most probable. This is feasible because our investigations have shown that buildings are recognized with a high degree of certainty and a low false negative rate. We can therefore assume that if *building* is not the most probable class, there is usually no building. The results of this approach are shown

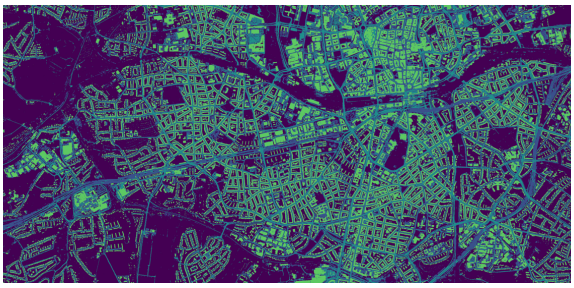


Figure 7: Final result: multiple predictions merged into one large area.

in images 5 (e) and 6 (e). Image 5 (e) clearly shows that buildings continue to be recognized very well, as well as the *asphalt/concrete* classes. The *paving* and *unbound surfaces* classes are now also recognized away from roads. In image 5 (e), the driveways to the buildings are now clearly visible, and in image 6 (e), the paved and unpaved areas of the parking lot, where previously everything was predicted as *unknown*, are now visible. Thus, our predictions are closer to reality than our generated labels.

Finally, since the labels are inaccurate and do not represent a ground truth, the predictions were evaluated primarily qualitatively after training and post-processing. Although metrics such as overall accuracy and mean intersection over union were used as orientation during training, the final results showed that the best quantitative results did not correspond to the best qualitative results.

After prediction and post-processing, the individual partial images were recomposed. We performed this without overlap, as no artifacts were detected using this procedure. The final result for a larger area can be seen in Figure 7.

6 CONCLUSION

We have presented an approach to develop an ML model for the German-wide semantic segmentation of orthophotos in urban areas. The key challenge we addressed was the lack of labeled data. The scarcity of training data is generally one of the biggest challenges in ML development (Mäkinen et al., 2021). Our literature review showed that while there are various ways to address the scarcity of training data, these were not directly applicable to our use case in practice. Methods such as Pseudo Labeling, Active Learning, or Transfer Learning achieve good results with a limited amount of labeled data, but they do require at least some labels, which were not available in our case. Generating synthetic data is a viable alternative, but it has the disadvantage that the generated data is artificial and does not fully capture the complex variations and patterns of the real world. There is no guarantee that models trained on synthetic data will perform well on real data.

Our work is therefore based on the approach of generating labels from other information sources. Existing works have deriving labels purely from OpenStreetMap, point clouds, or official cadastral data, but none of them have tried to incorporate segmentation results from an existing CNN. In our work, we combined official cadastral data with the results of a CNN that can classify 3D point clouds acquired by a car-mounted terrestrial mobile mapping system.

We presented the steps of our ML pipeline and discussed the individual steps of data processing and model development. In summary, in particular, the use of a large amount of training data from a wide variety of areas and seasons is essential for a scalable model. Furthermore, data augmentation, especially color-space transformations, is important. We implemented a UNet model, tuned the hyperparameters, and trained it using our generated labels.

The predictions from the developed model show very good results. Nevertheless, a critical view on the generated labels shows that these are not perfect. Due to the fact that the data was collected with a car-based terrestrial mobile mapping system, it has artifacts. There are areas that were hidden for the sensors while driving. As a consequence, our labels are incomplete and partly incorrect. However, we were able to show that by using a large amount of training data, these artifacts can be considered random noise and were not learned during training. Thus, it can be concluded that the predictions of our model are more realistic in this regard than the training data used.

The bigger issue are the areas that are not covered and therefore missing (essentially everything beyond the range of the mobile mapping laser scanner). Although we used official cadastral data for this, which works well for streets and buildings, it still leaves all land parcels unlabeled. This characteristic was trained as the *unknown* class. We address this issue by using a post-processing method, which assigns the second-most likely class to the unknown class. Additionally, in the case of the class *building* as the second-most likely class, we use the third-most likely one to prevent further artifacts. This approach works well, but it is important to note that it comes at the cost of high uncertainties and misclassifications.

Nevertheless, our results show that it is a good possibility to generate labels for training CNNs by using predictions from other CNNs and official cadastral data. Our model is not limited to specific regions or seasons, and it works throughout Germany. Due to the

high quality and generalizability, it is used in production by Deutsche Telekom to classify surfaces during the planning process for the rollout of high-speed fiber internet.

ACKNOWLEDGEMENTS

The aearial imagery used in this work was provided by the German Federal Agency for Cartography and Geodesy (BKG). © GeoBasis-DE / BKG (2026) Terms of use: https://sg.geodatenzentrum.de/web_public/nutzungsbedingungen.pdf

REFERENCES

Albrecht, C. M., Marianno, F., and Klein, L. J. (2021). Autogeolabel: Automated label generation for geospatial machine learning. In *2021 IEEE International Conference on Big Data (Big Data)*, pages 1779–1786. IEEE.

Altman, R. B. (2017). Artificial intelligence (ai) systems for interpreting complex medical datasets. *Clinical Pharmacology & Therapeutics*, 101(5):585–586.

Bock, H.-H. (2007). Clustering methods: a history of k-means algorithms. *Selected contributions in data analysis and classification*, pages 161–172.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. (2020). A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PmLR.

Chollet, F. et al. (2015). Keras. https://keras.io (last accessed: 2026-02-27).

Drory, A., Avidan, S., and Giryes, R. (2018). On the resistance of neural nets to label noise. *arXiv preprint arXiv:1803.11410*, 2.

Google (2015). TensorFlow. https://www.tensorflow.org/ (last accessed: 2026-02-27).

Gupta, A., Anpalagan, A., Guan, L., and Khwaja, A. S. (2021). Deep learning for object detection and scene perception in self-driving cars: Survey, challenges, and open issues. *Array*, 10:100057.

Hutchinson, M. L., Antono, E., Gibbons, B. M., Paradiso, S., Ling, J., and Meredig, B. (2017). Overcoming data scarcity with transfer learning. *arXiv preprint arXiv:1711.05099*.

Kocon, K., Krämer, M., and Würz, H. M. (2022). Comparison of CNN-based segmentation models for forest type classification. *AGILE: GIScience Series*, 3:42.

Krämer, M., Bormann, P., Würz, H. M., Kocon, K., Frechen, T., and Schmid, J. (2024). A cloud-based data processing and visualization pipeline for the fibre rollout in germany. *Journal of Systems and Software*, 211:112008.

Lekavičius, J. and Gružauskas, V. (2024). Data augmentation with generative adversarial network for solar panel segmentation from remote sensing images. *Energies*, 17(13):3204.

Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., and Talwalkar, A. (2018). Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of machine learning research*, 18(185):1–52.

Li, Z., Chen, B., Wu, S., Su, M., Chen, J. M., and Xu, B. (2024). Deep learning for urban land use category classification: A review and experimental assessment. *Remote Sensing of Environment*, 311:114290.

Mäkinen, S., Skogström, H., Laaksonen, E., and Mikkonen, T. (2021). Who needs MLOps: What data scientists seek to accomplish and how can MLOps help? In *2021 IEEE/ACM 1st workshop on AI engineering-software engineering for AI (WAIN)*, pages 109–112. IEEE.

Motlagh, N. K., Radhakrishnan, A., Davis, J., and Ilin, R. (2021). A framework for semi-automatic collection of temporal satellite imagery for analysis of dynamic regions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 704–712.

Parekh, J. R., Poortinga, A., Bhandari, B., Mayer, T., Saah, D., and Chishtie, F. (2021). Automatic detection of impervious surfaces from remotely sensed data using deep learning. *Remote Sensing*, 13(16):3166.

Ran, L., Li, Y., Liang, G., and Zhang, Y. (2024). Pseudo labeling methods for semi-supervised semantic segmentation: A review and future perspectives. *IEEE Transactions on Circuits and Systems for Video Technology*, 35(4):3054–3080.

Reiterer, A., Wäschle, K., Störk, D., Leydecker, A., and Gitzen, N. (2020). Fully automated segmentation of 2d and 3d mobile mapping data for reliable modeling of surface structures using deep learning. *Remote sensing*, 12(16):2530.

Reyes, M. F., Xie, Y., Yuan, X., d'Angelo, P., Kurz, F., Cerra, D., and Tian, J. (2023). A 2d/3d multimodal data simulation approach with applications on urban semantic segmentation, building extraction and change detection. *ISPRS Journal of Photogrammetry and Remote Sensing*, 205:74–97.

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer.

Tharwat, A. and Schenck, W. (2023). A survey on active learning: State-of-the-art, practical challenges and research directions. *Mathematics*, 11(4):820.

Voelsen, M., Bostelmann, J., Maas, A., Rottensteiner, F., and Heipke, C. (2020). Automatically generated training data for land cover classification with CNNs using sentinel-2 images.

Zhao, M., Zhang, C., Gu, Z., Cao, Z., Cai, C., Wu, Z., and He, L. (2025). Automatic generation of high-quality building samples using OpenStreetMap and deep learning. *International Journal of Applied Earth Observation and Geoinformation*, 139:104564.