Zurich University of Applied Sciences

Department Life Sciences and Facility Management

Institute of Natural Resource Sciences

TERM PAPER 2

Deep Learning for Pixel-Level Classification of Impervious and Pervious Surfaces in Remote Sensing

JANUARY 20, 2025

Author:
Julian Kraft[1]

Tutor:
Dr. Johann Junghardt[1]

Affiliations:
[1] Institute of Natural Resource Sciences

Imprint

Project type:	Term Paper 2
Title:	Deep Learning for Pixel-Level Classification of Impervious and Pervious Surfaces in Remote Sensing
Date:	January 20, 2025
Keywords:	classification, deep learning, machine learning, remote sensing, land cover classification, impervious surfaces, pervious surfaces, surface sealing, urbanization, environmental monitoring, pixel-based analysis
Copyright:	Zurich University of Applied Sciences

Author:	Julian Kraft[1] (kraftjul@students.zhaw.ch)
Tutor:	Dr. Johann Junghardt[1] (johann.junghardt@zhaw.ch)
Affiliations:	[1]Institute of Natural Resource Sciences

Abstract

Understanding the distribution of impervious and pervious surfaces is critical for effective urban planning, environmental management, and rainfall impact analysis. This study explores the use of convolutional neural networks (CNNs) for pixel-based classification of aerial remote sensing data to assess surface sealing. Using high-resolution SwissImage RS data, the analysis employs a simplified ResNet-18 architecture adapted for four-channel inputs, including RGB and near-infrared bands. A detailed workflow was developed, describing the preprocessing of the data, the training, and the evaluation of the model. A data augmentation strategy was implemented to improve the model's performance, and a hyperparameter tuning process was conducted to optimize the model. The best-performing model achieved a classification accuracy of 0.927, which is in a similar range to the results of previous studies utilizing a geoprocessing approach. While challenges such as mixed-pixel problems or limited data availability remain, this study demonstrates the potential of deep learning for detailed surface sealing analysis.

Contents

Code, and LaTeX source are to be found on GitHub:
https://github.com/juliankraft/TermPaper2_RasterClassification

List of Figures

List of Tables

1 Introduction

Understanding surface sealing is a critical aspect of urban planning, environmental monitoring, and sustainable development. Sealed surfaces, such as roads, parking lots, and buildings, are impervious to water and can change the hydrological properties of an area. Water cannot infiltrate the ground, reducing groundwater recharge and increasing the risk of flooding. In addition, the collected water flows together with wastewater into water treatment plants, which can lead to overloading. This can result in untreated water being released into the environment. Therefore, knowledge about the perviousness of surfaces is essential for informed decision-making and policy development.

Traditionally, remote sensing techniques have been used to address this challenge. These methods rely on spectral data, including near-infrared (NIR) bands, which are particularly effective for classifying sealed and unsealed surfaces. However, conventional approaches often involve rule-based geoprocessing or manual interpretation, which can be time-consuming and limited in scalability (Kadhim et al., 2016).

In recent years, advancements in deep learning have revolutionized remote sensing by enabling more automated, accurate, and scalable analysis. This is heavily driven by rapid improvements in computational power and the availability of large datasets. Neural networks have demonstrated great potential for tasks like image classification, segmentation, and object detection. While many studies in this domain focus on object-based or scene-based approaches (Thapa et al., 2023), pixel-level classification is an area with significant potential for granular analysis of surface characteristics. By factoring in the information of neighboring pixels, pixel-based methods can potentially provide very detailed analyses of an area, especially if high spatial resolution spectral data is available (Zheng & Chen, 2023).

Convolutional Neural Networks (CNNs) have been widely used for remote sensing applications, including surface sealing detection and land cover classification. Prominent examples include the use of models like ResNet-18 and ResNet-50, which have demonstrated strong performance in tasks requiring high-resolution image classification due to their ability to effectively learn hierarchical feature representations (Natya et al., 2022). VGG19, known for its simplicity and depth, has also been employed in remote sensing studies to classify urban landscapes and detect sealed surfaces, benefiting from its consistent feature extraction capabilities (Alem & Kumar, 2022). These architectures highlight the effectiveness of deep CNNs in capturing both spatial and contextual information for detailed surface analysis.

1.1 Background and Methodology

This term paper builds on previous research conducted by researchers at the Zurich University of Applied Sciences (ZHAW) for the canton of Basel-Landschaft, Switzerland. The

original study examines a geoprocessing approach to determine the perviousness of surfaces on a pixel level. The study combined high-resolution SwissImage remote sensing (RS) data with survey data identifying building footprints, roads, and other sealed surfaces. This paper explores an alternative methodology by feeding the RS data into a CNN to classify surface sealing. CNNs have demonstrated great potential for image classification by learning hierarchical feature representations directly from raw data (Zhao et al., 2024). Successfully training a CNN to detect sealed surfaces would provide a cost-effective and scalable alternative to the geoprocessing approach.

Unlike modern deep learning approaches that often focus on object-based or scene-based classifications (Thapa et al., 2023), this research examines a pixel-based classification approach. Pixel-based classification allows for a very fine-grained analysis. This is especially useful when high-resolution data is available, as with the SwissImage data with a resolution of 10 cm. By inputting small patches of pixels, the information of neighboring pixels can be used to improve the classification.

This study employs a simplified ResNet-18 architecture, adapted to accept four-channel inputs, including RGB and NIR bands. The labeled data from the original research is used both to train and test the model. The results will provide an indication of the potential of deep learning for surface sealing classification, and in the best case, a viable alternative to the geoprocessing approach.

2 Methods

2.1 Data

The data used for this project is the SwissImage RS (SwissTopo, 2024) data from the Swiss Federal Office of Topography (SwissTopo). It is a raster dataset with a resolution of 0.1 m containing four bands: RGB and NIR. To cover the area of interest (AOI), six tiles of the dataset are needed. Over this large AOI, there are three areas labeled with the corresponding labels Figure 1. These labels were provided by a team of researchers from the ZHAW. The three areas are distributed over a residential area (83,487 m^2), an industrial area (132,642 m^2), and a rural area (82,740 m^2). Two kinds of labels are available: the land cover category Figure 2 and an assessment of the degree of perviousness Figure 3. In Figure 4, the distribution of the data available for each label is shown for both the land cover category (`category`) and the sealing assessment (`sealed`). A third kind of label was generated by simplifying the degree of perviousness into two classes: pervious and impervious (`sealed_simple`). This was done by reclassifying the unknown areas to sealed ones since they only consist of BuildingDistortions and ConstructionSites.

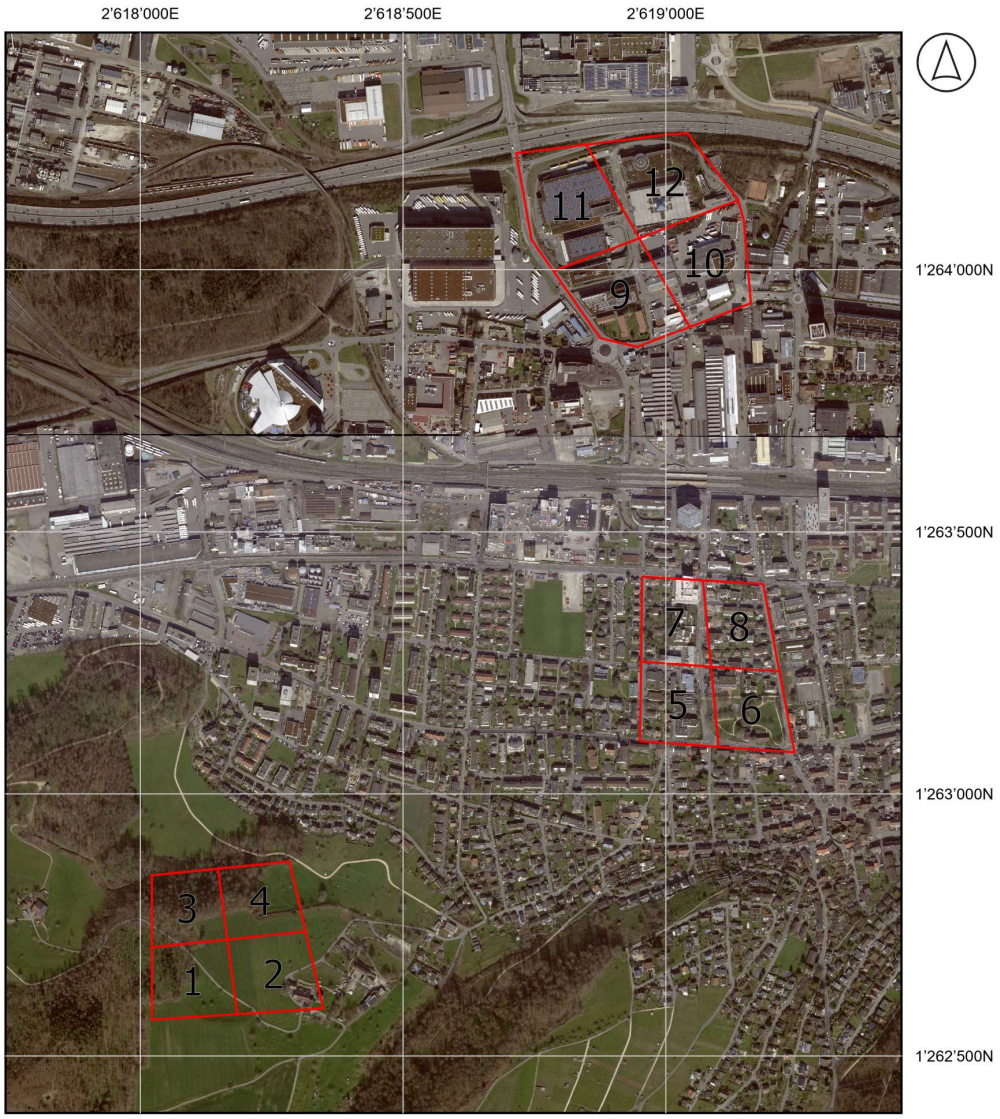

Figure 1: Map of the AOI, Pratteln, a municipality in the canton of Basel-Landschaft, Switzerland. The three marked areas, each split into four tiles, are the areas with corresponding labels. 1 - 4 are in the rural area, 5 - 8 in the residential area, and 9 - 12 in the industrial area.

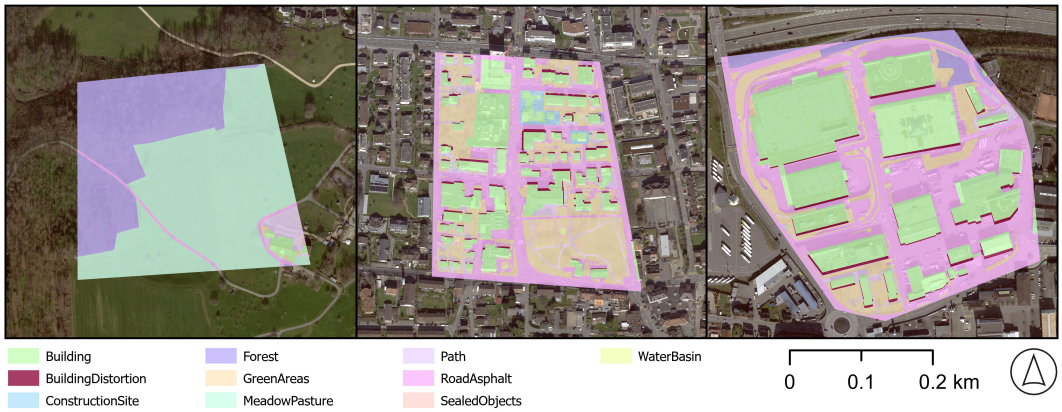

Figure 2: The three areas with the corresponding land cover categories.

Figure 3: The three areas with the corresponding degree of perviousness.

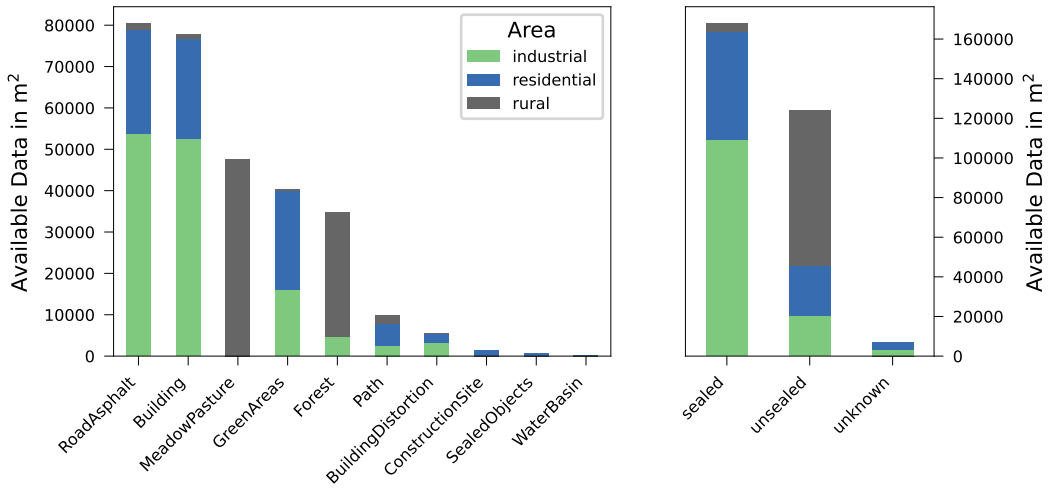

Figure 4: Available data by label, colored by the area.

2.2 Programming Language and Frameworks

To build and train the deep learning model, the programming language Python was used. The frameworks PyTorch and Lightning are widely used and powerful tools for building deep learning models.

2.3 Model Architecture

For this study, an adapted version of the ResNet-18 architecture was used to classify aerial imagery with high spatial resolution. ResNet-18 is a CNN model using residual connections to solve the issue of vanishing gradients during training. The residual connections allow the model to learn deep representations, reducing the risk of overfitting.

To make the model suitable for the given problem with the available data, some modifications to the standard ResNet-18 architecture were necessary:

- **Input Channels**: The original ResNet-18 was modified to work with four input channels (e.g., red, green, blue, and near-infrared bands) instead of the standard three channels (RGB). Adding this fourth channel provides essential spectral information. The NIR is especially suited to detect vegetation (Rouse et al., 1974), presumably aiding significantly in the classification of pervious surfaces.
- **Layer Configuration**: To adjust for the limited data available, one of the usually four fully connected layers was removed from the model, and the number of planes per block was reduced significantly from 64, 128, 256, 512 to 4, 8, 16.

- **Output Design**: Originally, the ResNet-18 model was designed for image classification tasks, outputting a single label for the whole image. Due to efficiency reasons, the model was adapted to output a patch of pixels instead. For this, the last fully connected layer was replaced with a convolutional layer mapping the output to a patch of pixels. The size of the output patch is adjustable via the `output_patch_size` parameter.

Some batch normalization layers help to stabilize the training and improve generalization. For the training, the AdamW optimizer was used, which combines adaptive learning rate methods with weight decay for regularization. The model was trained using a weighted cross-entropy loss function to handle class imbalances effectively.

The modified ResNet-18 model was implemented using PyTorch and was trained using the Lightning framework, which provides a high-level interface for PyTorch and is quite accessible even for beginners.

2.4 Data Processing and Augmentation

To feed data into the neural network, it is necessary to process it into a format that can be used for training. A standardized sample size is needed to ensure that the model can be trained on the data.

2.4.1 Preprocessing

In the first step, the data was processed using ArcGIS Pro. The process included several steps implemented as a model with the Model Builder Figure 5. The six tiles were mosaicked together, and the area of interest was clipped. The area labels 1 - 12 and both versions of available data labels were transformed to raster datasets and added as additional bands to the dataset. The data was then exported as a GeoTIFF file. To use the data in a neural network, an additional step was necessary. Using Python, the data was transformed into Zarr format. This format is a chunked, compressed, N-dimensional array storage format with multi-scale support. This allows for lazy loading and therefore for more memory- efficient data access during training.

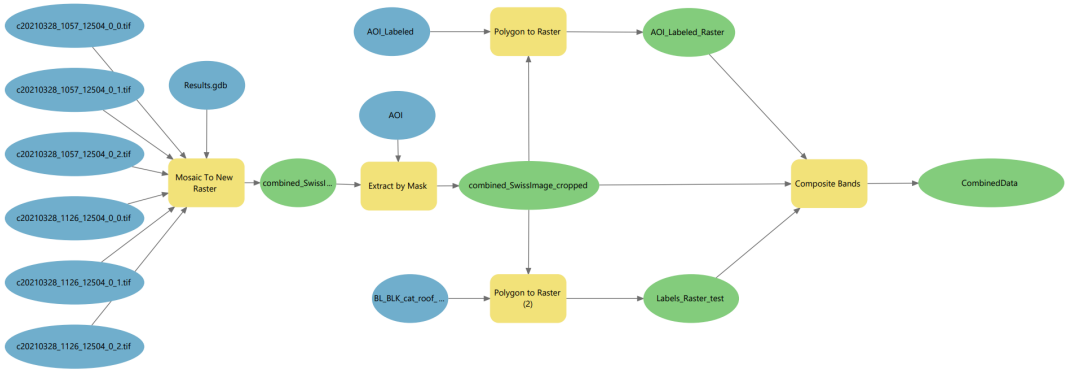

Figure 5: ArcGIS model used for preprocessing the data.

2.4.2 Processing and Augmentation

To feed the data into a neural network, a PyTorch DataLoader was implemented. The data was sampled as cutouts of size 51x51 pixels, predicting the 5x5 center pixels as output. This cutout size and output size were implemented as a parameter, but no other values were tested. The DataLoader indexes the available data depending on the cutout size and the corresponding area labels depending on the purpose:

- Areas for Training: 1, 2, 5, 6, 9, 10
- Areas for Validation: 3, 7, 11
- Areas for Testing: 4, 8, 12

It handles the data augmentation on the fly, during training steps. For the data augmentation, a parent class BaseAugmentor was implemented from which the different augmentors inherit. The implemented augmentors - refer to Table 1 - are then chained into an object of the class AugmentorChain which can be used in the DataLoader. Only the Flip- and RotateAugmentor were used for the training of the models.

Table 1: Implemented augmentors

Augmentor	Description
FlipAugmentor	Randomly flips the image vertically and horizontally
RotateAugmentor	Randomly rotates the image 0, 90, 180 or 270°
PixelNoiseAugmentor	Adds random noise to the image (parameter: scale)
ChannelNoiseAugmentor	Adds random noise to the channels (parameter: scale)

2.5 Fitting the Model

Fitting the model was done on the IUNR HPC cluster using node 301, an HPE Apollo 6500 Gen10+ node running Rocky Linux 8. The node is equipped with 8 NVIDIA L40S GPUs (48 GB each), dual AMD EPYC 7742 processors, 512 cores, and 5800 GB of storage, providing the computational power needed for high-performance tasks. The training was done using a potential limit of 300 epochs and an early stopping callback set to stop after 10 epochs without improvement. Batch size was set to 128 for training and 256 for validation and prediction. During training, only the best model and the last model were saved. Upon completion of the training, the best model was loaded and used for prediction on the whole dataset. These predictions were saved as a Zarr file for later use in the evaluation. For both the category classification and the perviousness classification, two models were trained with and without data augmentation. The data augmentation was done using the Flip- and RotateAugmentor. The learning rate was set to 0.001 and weight decay to 0.01.

For the simplified perviousness classification, a limited hyperparameter search was conducted. All combinations of the following parameters were tested:

- Learning rate: 0.001, 0.0001
- Weight decay: 0, 0.1, 0.01
- Data augmentation: no augmentation, Flip- and RotateAugmentor

2.6 Evaluation

For the performance evaluation of the models, the Python library Scikit-Learn was used. The predictions were loaded from the Zarr file and compared to the ground truth labels. Accuracy, F1-Score (weighted and per class) were calculated for every model and stored in a CSV file for later comparison. For the best model, the accuracy was also calculated, grouped by corresponding land cover category to provide insight into which category was classified best.

3 Results

The best-performing model achieved an accuracy of 0.9274 and a weighted F1 Score of 0.9275. The results for all trained models are summarized in Table 2. All trained models for the simplified perviousness lead the ranking, followed by the models for the perviousness classification and the category classification with accuracies of around 0.88 and 0.59, respectively. Only the label type `sealed_simple` will be considered for further analysis, as it is the most relevant for the study area.

Table 2: Accuracy and F1 Score for all trained models, arranged by accuracy.

LabelType	Augmented	LearningRate	WeightDecay	Accuracy	F1 Score
sealed_simple	True	0.0001	0.1	0.9274	0.9275
sealed_simple	False	0.0001	0.01	0.9109	0.9112
sealed_simple	True	0.001	0.1	0.9069	0.9072
sealed_simple	True	0.0001	0.01	0.9034	0.9038
sealed_simple	True	0.001	0	0.9028	0.9032
sealed_simple	False	0.001	0.1	0.9021	0.9024
sealed_simple	True	0.0001	0	0.9007	0.9011
sealed_simple	False	0.0001	0.1	0.9000	0.9004
sealed_simple	False	0.001	0.01	0.8999	0.9001
sealed_simple	False	0.001	0	0.8996	0.9000
sealed_simple	True	0.001	0.01	0.8980	0.8984
sealed_simple	False	0.0001	0	0.8977	0.8980
sealed	False	0.001	0.01	0.8817	0.8716
sealed	True	0.001	0.01	0.8763	0.8619
category	True	0.001	0.01	0.5902	0.5722
category	False	0.001	0.01	0.5806	0.5665

3.1 Hyperparameter Tuning

The hyperparameter tuning results included in Table 2 and visualized in Figure 6 show that a `learning_rate` of 0.0001 performs better and that a `weight_decay` of 0.1 leads to the best results when combined with data augmentation. Overall, data augmentation seems to have a positive effect on accuracy, but there is an exception for a `weight_decay` of 0.01, where the accuracy is higher without data augmentation.

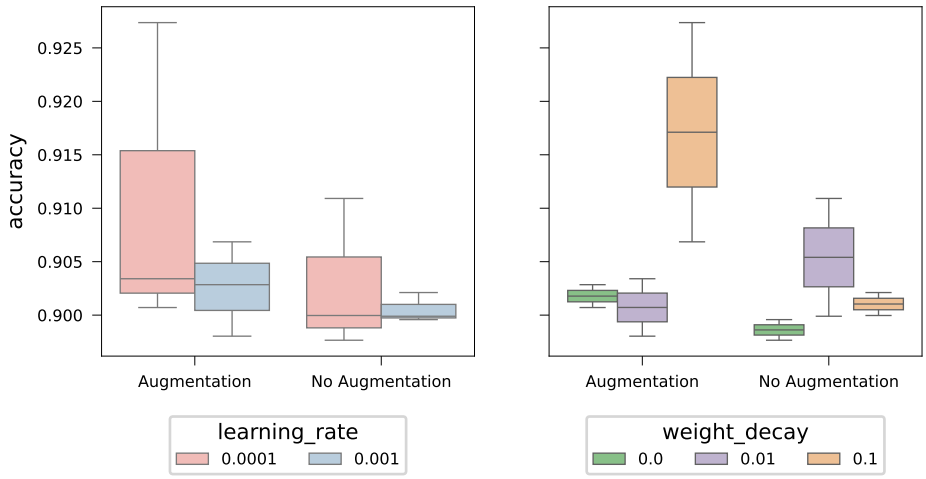

Figure 6: Accuracy of the models for the different hyperparameters grouped by use of data augmentation.

3.2 Best Model

The best-performing model was the one for the simplified perviousness classification with the following hyperparameters:

- Learning rate: 0.001
- Weight decay: 0.01
- Data augmentation applied

It achieved an accuracy of 0.927 and a weighted F1 Score of 0.927. The F1 Score per class was 0.92 for the class `'unsealed'` and 0.93 for the class `'sealed'`. For a visual inspection of the predictions, refer to Figure 7. The accuracy per category, shown in Figure 8, shows categories like `Building`, `BuildingDistortion` and `GreenArea` predicted with an accuracy between 0.8 and 0.9. The categories `RoadAsphalt`, `Forrest` and `MeadowPasture` where predicted with an accuracy above 0.95. The categories `ConstructionSite`, `WaterBasin`, `Path` and `SealedObjects` where predicted with an accuracy below 0.8. To investigate the incorrect predictions further, a plot was created that shows the incorrectly predicted pixels colored by land cover category in Figure 9. Since in the rural test area it was not showing much, the plot is limited to the urban and industrial test areas.

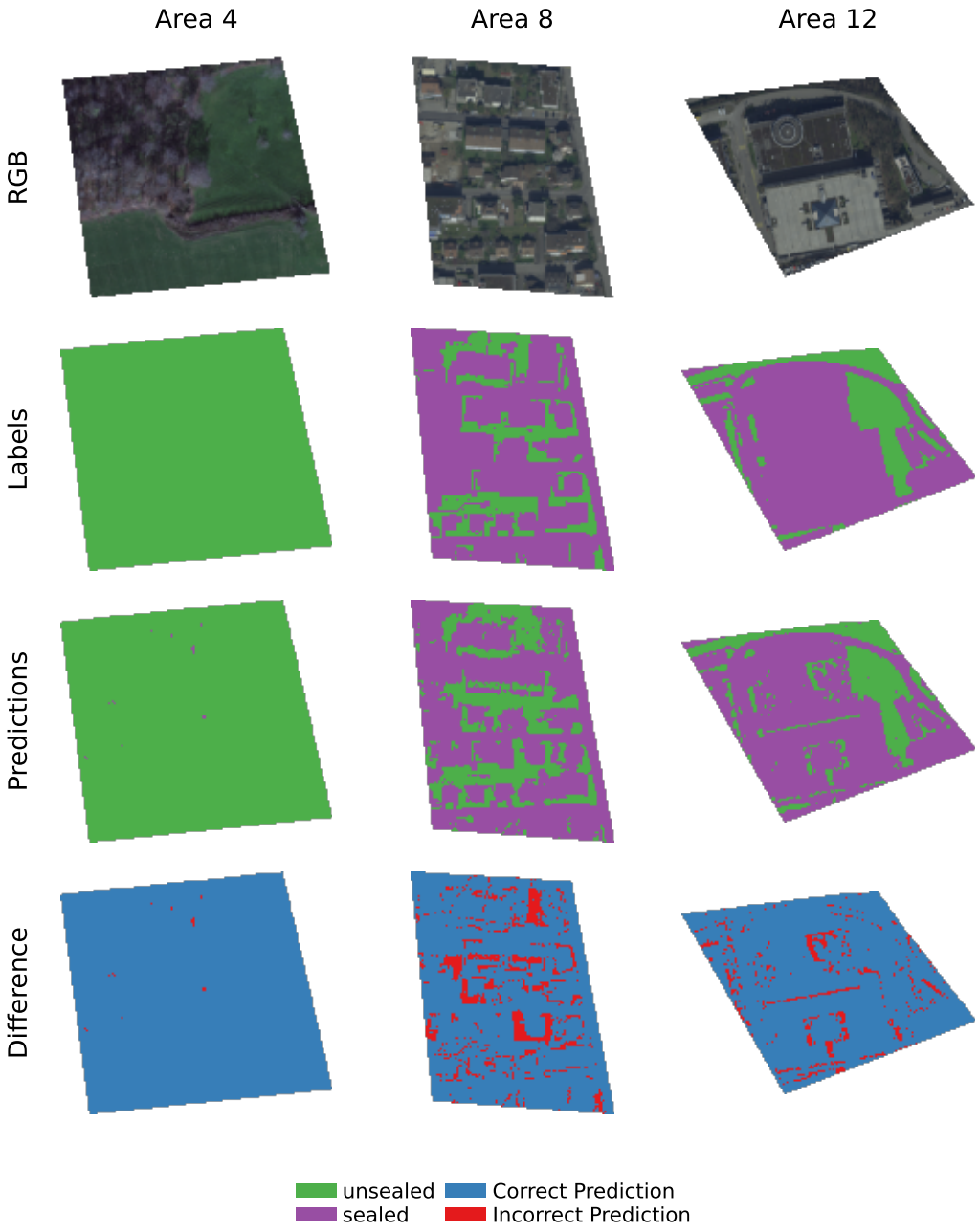

Figure 7: Visual inspection of the predictions for the test areas of the best model.

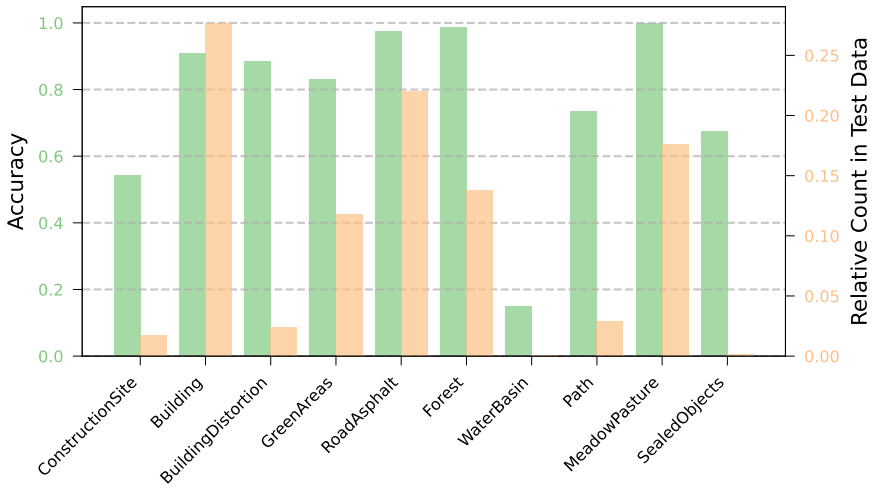

Figure 8: Accuracy per category for the best-performing model, including the relative count of available data for each category.

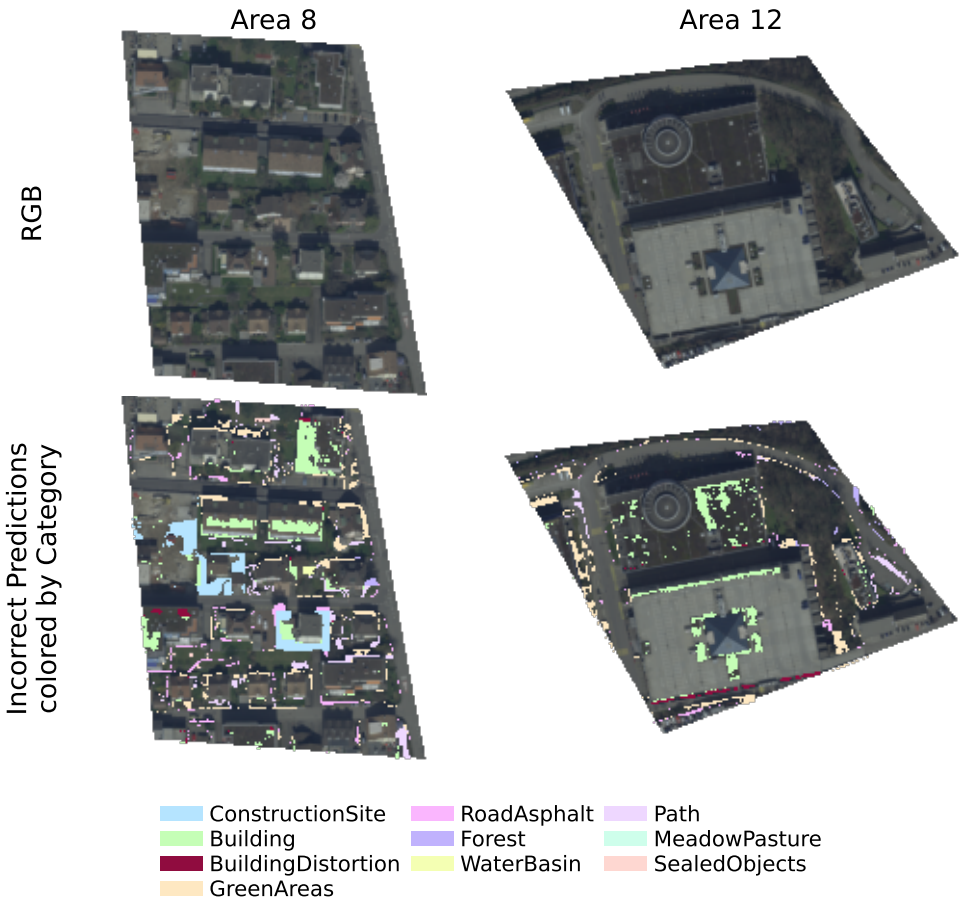

Figure 9: Incorrectly predicted pixels colored by land cover category on the bottom row. The top row shows the RGB image for reference.

4 Discussion

4.1 Best Model Performance

The best-performing model achieved a reasonably high accuracy of 0.92. The validation metrics, loss, and accuracy, shown in Figure 10, suggest reasonable progress during the fitting process. Since they both are not completely flattening out, it might even be possible to further improve the model using the same architecture and hyperparameters by training it for more epochs. This could be achieved by using a higher value for the `patience` parameter – the next value to be tested could be 20 instead of 10.

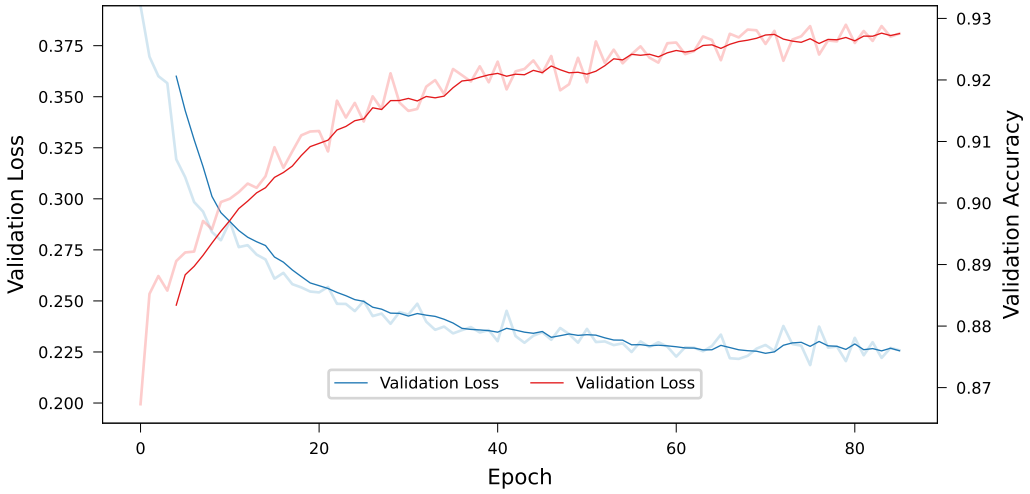

Figure 10: Validation loss and accuracy for the best model.

The accuracy per class, shown in Figure 8, aligns with expectations. Classes like `Building`, `GreenAreas`, `RoadAsphalt`, `Forest`, and `MeadowPasture` are predicted with high accuracy. Conversely, classes where there is very little data available, like `WaterBasin`, or classes where the data is more vague, like `ConstructionSites`, are predicted with lower accuracy.

From the visual inspection of the predictions in Figure 7, it looks like the borders between the classes are difficult to predict. This can be explained by the fact that the classes are not always very precise, which specifically affects these areas. Another explanation could be the resolution of 10 cm, which leads to pixels in reality consisting of multiple classes. This issue, referred to as mixed pixels, could be part of the reason for the lower accuracy in the border areas.

4.2 Hyperparameter Tuning

The hyperparameter tuning process was successful in finding the best hyperparameters within the grid search. From Figure 6, it can be seen that lower values for the `learning_rate` seem to work better and that a higher regularization during training – a higher value of `weight_decay` – leads to better results. Very interesting is the fact that the benefit of the data augmentation seems to correlate with more regularization during training. The grid of evaluated hyperparameters was rather limited due to limited computational resources and the remaining time. A more extensive search could potentially lead to even better results. What might really be worth trying is testing even smaller values of `learning_rate`, higher values for `weight_decay`, and using other options for data augmentation – pixel- and channel-noise in different combinations.

4.3 Comparing the Results to the GIS Approach

Since the validation for the original studies was done for a different area in Rünenberg, using a different dataset, it is hard to compare the results directly. Adding to that, different land cover categories were used, as well as different resolutions. Overall, the original studies achieved an accuracy of 0.9 to 1 for the more obvious classes like `Buildings`, `SealedRoads`, `GreenAreas`, and `MeadowPastures`. For the more difficult classes, like `PathUnsealed` and `SealedObjects`, the accuracy was lower as well. The results of this study – as differently as they were created – reach a similar range of accuracy.

4.4 Prospects

The results of this study show that the approach of using deep learning for perviousness classification is promising. The results are in a similar range as the results of the original study using a GIS approach. Therefore, it might be worth further investigating the potential of deep learning for this task. The next steps, besides the mentioned hyperparameter expansion, more data augmentation, and more thorough training by going for more epochs, include other promising approaches. One could be to try different model concepts like U-Net – specifically designed for image segmentation tasks. Another approach could be to use a pre-trained model to build upon – the issue there is finding a model that is trained on similar data and accepts four channels as input. Another idea could be to implement some self-supervised learning techniques utilizing unlabeled data to improve the model. As with most deep learning tasks, the more data, the better – so it would be worth trying to get more data of possibly even higher quality to improve the existing models.

5 Acknowledgment and Declaration

5.1 Acknowledgment

I thank Dr. Johann Junghardt for his support. I would like to express my gratitude to my brother, Dr. Basil Kraft, for his continuous support with his expertise in the field of machine learning.

5.2 Declaration of AI Usage

GitHub Copilot was used to assist in writing the code and text for this project.

ChatGPT was used to assist in researching as well as writing the code and text for this project.

While paragraphs like the introduction and the abstract were originally generated by ChatGPT, they were checked word by word and adjusted to fit the context of this project and the overall language style of the author.

N-FO-Formular Statement of Authorship for

Student Work

Zürcher Hochschule
für Angewandte Wissenschaften

Life Sciences und Facility Management

Education Unit

Statement of Authorship for Student Work at the School of Life Sciences and Facility Management

By submitting the enclosed

☐ Project
☐ Literature review
☐ Course work
☒ Minor paper
☐ Bachelor's thesis
☐ Master's thesis (tick as appropriate)

the student affirms independent completion of the(ir) work without outside help.

The undersigned student declares that all printed and electronic sources used in the text and the bibliography are correctly indicated, i.e., that the work does not contain any plagiarism (no parts that have been taken in whole or in part from another's or his/her own text or from another's or his/her own work without clear identification and without stating the source).

In the event of misconduct of any kind, Paragraph 39 and Paragraph 40 of the General Academic Regulations for Bachelor's and Master's degree programmes at the Zurich University of Applied Sciences (dated 29 January 2008) and the provisions of the Disciplinary Measures of the University Regulations shall apply.

Location, date: Student signature:

Neuhausen, 2025-01-20 Julian Krutt

Note on submitting the Statement of Authorship:

Direct submission of the work: This Statement of Authorship is to be inserted in the appendix of the ZHAW version of all work with original signatures and date (copies will not be accepted).

Submission of the work via Complesis: The Statement of Authorship should be made directly in Complesis by clicking as directed and should not be inserted in the appendix of the work.

Erlassverantwortliche/-r	LeiterIn Stabsbereich Bildung	Ablageort	2.05.00 Lehre Studium
Beschlussinstanz	LeiterIn Stab	Publikationsort	Public
Genehmigungsinstanz			

Version	Beschluss	Beschlussinstanz	Inkrafttreten	Beschreibung Änderung
1.0.0	15.03.2022	LeiterIn Stab	15.03.2022	Originalversion
1.1.0	15.04.2024	Leiter:in Stab	01.5.2024	Adding clarification on self-plagiarism

References

Alem, A., & Kumar, S. (2022). Transfer Learning Models for Land Cover and Land Use Classification in Remote Sensing Image. *Applied Artificial Intelligence*, *36*(1), 2014192. https://doi.org/10.1080/08839514.2021.2014192

Kadhim, N., Mourshed, M., & Bray, M. (2016). Advances in remote sensing applications for urban sustainability. *Euro-Mediterranean Journal for Environmental Integration*, *1*(1), 7. https://doi.org/10.1007/s41207-016-0007-4

Natya, S., Manu, C., & Anand, A. (2022). Deep Transfer Learning with RESNET for Remote Sensing Scene Classification. *2022 IEEE International Conference on Data Science and Information System (ICDSIS)*, 1–6. https://doi.org/10.1109/ICDSIS55133.2022.9915967

Rouse, J. W., Haas, R. H., Schell, J. A., & Deering, D. W. (1974). Monitoring vegetation systems in the Great Plains with ERTS. Retrieved January 18, 2025, from https://ntrs.nasa.gov/citations/19740022614

SwissTopo. (2024). *SWISSIMAGE RS*. Retrieved November 15, 2024, from https://www.swisstopo.admin.ch/de/orthobilder-swissimage-rs

Thapa, A., Horanont, T., Neupane, B., & Aryal, J. (2023). Deep Learning for Remote Sensing Image Scene Classification: A Review and Meta-Analysis. *Remote Sensing*, *15*(19), 4804. https://doi.org/10.3390/rs15194804

Zhao, X., Wang, L., Zhang, Y., Han, X., Deveci, M., & Parmar, M. (2024). A review of convolutional neural networks in computer vision. *Artificial Intelligence Review*, *57*(4), 99. https://doi.org/10.1007/s10462-024-10721-6

Zheng, X., & Chen, T. (2023). High spatial resolution remote sensing image segmentation based on the multiclassification model and the binary classification model. *Neural Computing and Applications*, *35*(5), 3597–3604. https://doi.org/10.1007/s00521-020-05561-8