Semantic segmentation of roof superstructures

MSc Thesis | Geomatics for the Built Environment

Irène Apra

June, 2022

Cover image:
Vector roof superstructures overlaid on digital ortho-photos;
Wartenberg, Bavaria

Semantic Segmentation of Roof Superstructures

MSc Thesis | Geomatics for the Built Environment

by Irène Apra

A thesis submitted to the Delft University of Technology in partial fulfillment
of the requirements for the degree of Master of Science in Geomatics

TU Delft
June, 2022

Irène Apra: *Semantic Segmentation of Roof Superstructures* (2022)

🅭🅯 This work is licensed under a Creative Commons Attribution 4.0 International License. View a copy of this license at http://creativecommons.org/licenses/by/4.0/.

This work has been co-supervised by:

Delft University of Technology,
Faculty of Architecture & the Built Environment:
Dr. Ken Arroyo Ohori
Dr. Giorgio Agugiaro

Technical University of Munich,
Chair of Geoinformatics:
Bruno Willenborg, M.Sc.
Chair of Automative Engineering:
Sebastian Krapf, M.Sc.

Co-reader: Ir. Shenglan Du
Board of Examiners: Ir. Herman de Wolff

Abstract

Automated reconstruction of detailed semantic 3D city models is challenging due to the need for high-resolution (HR) and large-scale input datasets, the ambiguous definition of the ensuing model, the intricacy of the processing pipeline, and its costs. Furthermore, existing methods mainly focus on geometry rather than semantics. Detailed semantic models may include roof installations whose size and function vary: dormers, windows, chimneys, etc. All elements visible on the roof from an aerial view are called ‘‘superstructures”. Deep Learning techniques can facilitate their modelization. A convolutional neural network (CNN) applied to aerial images outputs a segmentation of superstructure categories. These results can then be vectorized, extruded in 3D with their semantic description, and added to a simple 3D model. This thesis demonstrates that building height data fused to a CNN on RGB aerial images improves the semantic segmentation of roof superstructures for classes with relief. Fusion of absolute and relative height data with different interpolation methods applied to LiDAR point cloud data is achieved through a fusion network from the state-of-the-art (FuseNet). First of all, experiments prove that prediction accuracies increase by 11% on average for dormers and 12% for chimneys compared to U-Net output on the same dataset. Best performance is reached with the fusion of absolute height (rather than normalized) and IDW or NN interpolation technique (rather than none). However, although superstructure types are better recognized, their boundaries are fuzzier due to data input mismatches, and more background pixels are classified. Secondly, the predictions and modelization of both a Bavarian and Dutch test set prove the technique scalability. However, a training set annotated for Bavaria and applied to a test set in the Netherlands yields inaccurate results due to local architectural typologies and different input data characteristics.

Acknowledgements

I am grateful for the qualitative guidance that supported me through the research process. First of all, Ken for his valuable insights and suggestions to improve the form and substance of this work. My thanks also go to Giorgio for providing me with knowledge about semantic 3D models, their database management, and a Dutch test case. I express my gratitude to Shenglan, for her pertinent comments and Herman for his interest in this work.

Additionally, my thanks go to Bruno and Sebastian for giving me the chance to participate in this project at the Technical University of Munich. Thank you for helping me settle into the project and providing recurrent and relevant feedback.

Moreover, I wish to acknowledge Florian for answering questions regarding U-Net tuning. And Matthias for his crucial help with the 3D modelization of network predictions and labels.

Thirdly, my thanks go to my family and Samuel for their unconditional support. Special thanks to Benjamin for his encouragement, programming, and mathematical advice over the last two years.

Finally, I am grateful to the Technical Universities of Delft and Munich and the European Union for allowing such cooperation on research projects.

Contents

List of Figures

List of Tables

Acronyms

Acronym	Definition
α	Learning rate (model training)
abs.	Absolute height maps (obtained from raw Point Cloud data)
AHN	Actueel Hoogtebestand Nederland (Current altitude data of the Netherlands)
ANN	Artificial Neural Network
AP	Aerial Photography
(3D-)BAG	Basisregistraties Adressen en Gebouwen (Dutch Addresses and Buildings key register)
CE	Cross-entropy (loss function)
CFL	Categorical focal loss (loss function)
CNN	Convolutional Neural Network
CSV	Comma-Separated Values
CV	Computer Vision
DL	Deep Learning
DNN	Deep Neural Network
DOP	Digital Ortho-Photo
DSM	Digital Surface Model
DTM	Digital Terrain Model
FP	False Positive (model prediction)
FN	False Negative (model prediction)
GDAL	Geospatial Data Abstraction Library
GML	Geography Markup Language
GT	Ground Truth
HR	High-resolution
IDW	Inverse-Distance-Weighting interpolation
IE	Information Entropy
IoU	Intersection over Union
LAS	LASer, file format for LiDAR point cloud
LDBV	Landesamt für Digitalisierung, Breitband und Vermessung (State Office for Digitization, Broadband and Surveying)
LiDAR	Light Detection And Ranging
LOD	Level of Detail
ML	Machine Learning
NN	Neural Network Nearest-Neighbor interpolation
nDSM	normalized Digital Surface Model
PDAL	Point Data Abstraction Library
PC	Point Cloud
PNG	Portable Network Graphics
PV	Photovoltaic
rel.	Relative height maps (obtained from PC normalization through semantic 3D model)
RGB	Red-Green-Blue image bands
SQL	Structured Query Language
TIF	Tagged Image File
TP	True Positive (model prediction)
uint8	unsigned integer, 8 bit
V-HR	Very High-Resolution
WKT	Well-Known Text
WMS	Web Map Service

1 Introduction

Digital representations of the built environment support diverse purposes such as city planning and analysis, or disaster management. Particularly relevant, semantic 3D city models virtually reproduce urban areas by incorporating descriptive attributes to building geometries. The open data model and exchange format CityGML defines four Levels of Detail (LOD) for object depiction (OGC 2012), further subdivided by Biljecki et al. 2014. Each of them is relevant for different applications and, thus, necessitates a clear definition. Some usages require more advanced geometries, whereas others benefit from concise representations (Peters et al. 2021). Detailed models have high LODs and include small architectural specificities (dormers, windows, etc.) as in figure 1.1. They can serve energy demand estimation, roof insulation assessment by detecting irregularities and thermal bridges, urban wind flow analysis, runoff, and noise diffusion modeling. Moreover, they can support urban planning and be used to assess the age and condition of buildings through their architectural characteristics (dormers, chimneys) or their potential for vertical extension. Finally, these models can support the energy transition by helping to determine the roof surface suitable for photovoltaic (PV) or solar-thermal installations, which convert sunlight into electricity and heat, respectively.

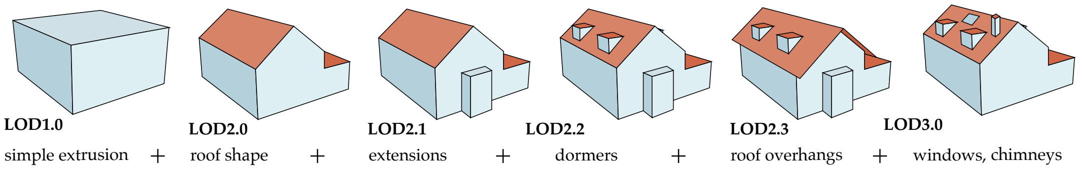


Figure 1.1: Level of Details according to Biljecki et al. 2014

1.1 Motivation

Automatically generating these models over city scales is a challenging practice, especially to reach high LODs. Although several reconstruction techniques exist for buildings up to LOD2.0., more detailed 3D representations capturing roof installations (LOD2.2 and further) are rare, and methods to generate them are lacking. Given the numerous applications of such models, it is urgent to develop appropriate techniques to produce them. Obtaining them for a single and recent building can be achieved by simplifying its Building Information Model (BIM), a tool used by architects and engineers to share and manage construction processes (Donkers et al. 2016, Arroyo Ohori et al. 2018). However, on a broad scale, a technique must be elaborated from elementary input data, available for all constructions uniformly.

The challenges arise from the need for high-resolution (HR) and large-scale input datasets, the ambiguous definition of the ensuing model, and its complex data structure. Moreover, the processing pipeline to efficiently extract advanced LODs on a broad scale is intricate, costly, and time-consuming. Finally, existing methods mainly focus on geometry rather than semantics. However, the output should be light-weighted and contain thematic information to broaden their potential analysis usages, unlike 3D mesh representations principally dedicated to visualization (Willenborg et al. 2018). Since Deep Learning (DL) allows extracting information from extensive datasets with little manual work, it offers high potential for automatizing the generation of 3D urban objects based on various data inputs. The studied approach detects “roof superstructures” through a DL algorithm, extrudes them in 3D, and adds them to a simple 3D model available. In the context of this work, superstructures have a broad definition, including all that is visible on the roof from aerial views. It varies from architectural specificities (chimneys, dormers, windows, etc.) to technical ones (PV installations, ventilation systems, etc.). Specifically, this research seeks to improve the DL performance for superstructure detection by exploring the combination of 2D and 3D data sources within a Deep Neural Network (DNN).

1.2 Research context

The work is carried out in the context of an existing project, developed since 2018 at the Chair of Automotive Engineering of the Technical University of Munich. The elaborated tool assesses the PV potential of buildings in the German state of Bavaria to support the development of sustainable solar charging stations for electric vehicles. Therefore, this research benefits from the current implementation framework and datasets while aiming to improve the results.

Among other aspects, the existing processes include an evaluation of the physical suitability of buildings for PV panel installation based on the geometrical characteristics of the roofs. Those are two folds: roof azimuths and available surface. Both are assessed independently through a Convolutional Neural Network (CNN) on aerial images, a particular type of DNN applied to raster data for 2D Computer Vision (CV) tasks. The algorithm estimating the roof surface available detects roof superstructures through CNN and subtracts their areas from the total roof surface. Relevant classes are determined during the research process and yield corresponding geometrical and semantics enhancement of the output model.

The current implementation uses simple DNN architectures applied to 2D data. This thesis strives to improve the geometrical assessment of roofs by fusing 3D information into the network to enhance the multi-class segmentation of superstructures. Final surfaces detected are extruded in 3D in a simplified form, and the semantic 3D model already available can be upgraded. The resulting representation includes characteristics of LOD2.2 through dormers' depiction and particularities of LOD3.0 with chimneys and windows (cf. figure 1.1). However, roof overhangs are not depicted since unidentifiable from aerial data, resulting in a model different from any described by Biljecki et al. 2014.

1.3 Research question

The main research question is the following: How can building height data fused to a Convolutional Neural Network (CNN) on RGB aerial images improve the semantic segmentation of roof superstructures?

Sub-questions regarding input, network architecture, and evaluation of its output are thereby posed:

- Which types of building height information are the most relevant for detecting roof superstructures through data fusion?
- How should the input data be processed? How to deal with different resolutions of available data (point cloud and images) and temporal mismatches?
- How should the existing pipeline be modified to fuse height data efficiently to the network?
- How to evaluate the results' accuracy for multi-class superstructure segmentation and the added value of different height input?

1.4 Hypotheses

Three types of hypotheses are drawn:

- **Improvement for some classes only.** Superstructures will benefit differently from height data according to their type. Detection of chimneys and other higher-profile obstacles will be improved, unlike flat superstructures such as solar panels and windows.
- **Absolute versus relative height.** Relative elevation, normalized using the simple building model, highlights roof installations more than absolute elevation data since it has a narrower range of values corresponding to each building independently.
- **Interpolation methods.** An interpolated dataset provides a higher understanding of the roof structure and is more likely to improve predictions than punctual data. The interpolation technique taking the most points into account is expected to provide richer information and thus, produce better results.

1.5 Approach

First, a network architecture for height data fusion is chosen from the state-of-the-art. Then, relevant 3D data is identified (LiDAR point cloud) and processed to provide height maps. A normalized dataset called "relative", is obtained by computing the difference between the points and a simple 3D city model. Absolute and relative height maps are derived, and diverse interpolation methods are investigated. Next, each dataset is fused to the CNN, and their respective results are assessed experimentally. The best results are eventually modeled in 3D, and PV potential assessment derived from the resulting model is demonstrated.

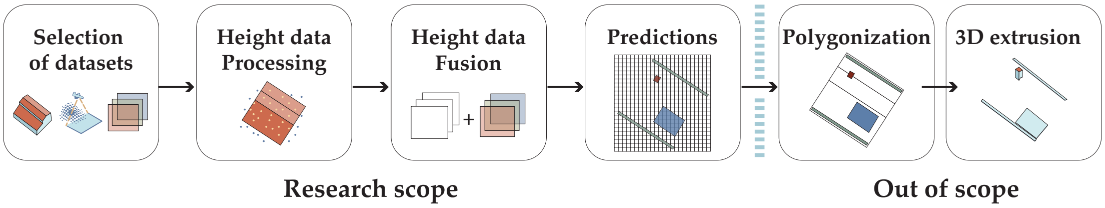


Figure 1.2: Research scope

1.6 Organization of the document

The following chapter defines the main terms and presents relevant work related to DL and its application to aerial imagery combined with other data types. It also describes the status of DL in the frame of the existing pipeline for PV potential estimation. The third chapter presents the methodology, processes to generate height maps and quantitative assessment tools for CNN predictions. Next, datasets, data splits, and DL experiments are detailed. The subsequent part exposes the results and analyzes them through quantitative and qualitative criteria. The final chapter answers the research question, discusses the outcomes, and describes contributions and potential future work.

2 Theoretical background and related work

The introduction chapter exposed the relevance of the research and the need for efficient roof superstructure segmentation. This chapter explores state-of-the-art semantic segmentation and 3D data fusion methods. Consequently, a methodology to answer the research problem is determined.

First, main DL terms are defined, and the mathematical background for understanding a neural network functioning is exposed. Then, performing network architectures are chronologically presented, as well as methods for incorporating multiple data types and their performance. Finally, an overview of the works achieved for the PV assessment project is provided, including a description of how it has applied DL so far. Lastly, a subsection concludes on a methodology built upon the works aforementioned.

2.1 Deep Learning, Context and Definitions

Artificial intelligence (AI) can be considered a branch of computer science, aiming at simulating, extending, and expanding human intelligence (Shi and Zheng 2006) and making machines able to reproduce human behavior based on human-intelligence understanding. Application domains are broad, including robotics, voice and image recognition, and natural language processing (Niu et al. 2016). The growth of research on AI exploded in the 1990s, mainly due to the increase in computational power (*ibid.*).

Artificial Neuron Networks (ANN), often just called Neural Networks (NN), were developed to build such “intelligent” machines. These networks are inspired by the biological neural network functioning (Gupta 2013). They can have various applications, including engineering, and can be employed for pattern recognition or image segmentation which is the case in this research. In supervised Machine Learning (ML), the model learns from its previous experience, obtained through labeled data, to output an answer (e.g. classification tasks, assigning a category to the data). Whereas in unsupervised ML, the model outputs a result merely based on the input, with no previous knowledge (e.g. clustering tasks, grouping data based on common characteristics). This work employs the first technique.

2.1.1 Neural Network

A NN is composed of neurons (or “units”) implementing functions mapping an input with output. More precisely, the input, comparable to a synapse, is multiplied by a weight computed with a mathematical function, whereas another one—the activation—determines the neuron’s output (Gupta 2013). A succession of neurons forms a “Neural Network”. In the case of supervised ML, these are trained to make the functions provide the desired output when fed with an input. The units located at a similar depth in the network form a hidden layer, and the first and last network layers are, respectively, called input and output layers (figure 2.1). A network composed of numerous layers is called a “Deep Neural Network”, from which the appellation “Deep Learning” derives. The training data, usually manually labeled, allows the network to figure out the functions of the nodes connecting best the input x to the correct output $\hat{y}$ (Ng 2021).

Different network sorts have been developed for distinct aims and adapted to various input types (sound, images, etc.). One of the purposes is image classification, also called “semantic segmentation”, which assigns an object-class probability value to each pixel on an image. Minaee et al. 2020 provides a review of the architectures developed for this task until 2019.

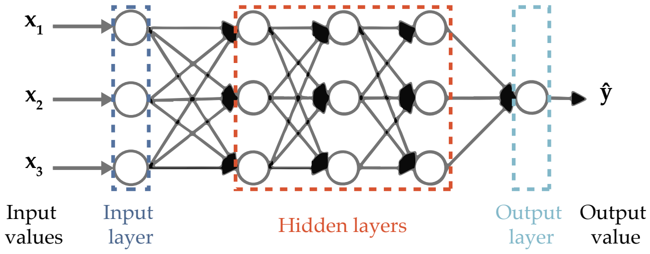


Figure 2.1: Standard Neural Network, adapted from Ng 2021

2.1.2 Training, validation and test datasets

In the case of supervised ML, a "training dataset" is necessary to train the network and fit the model. Additionally, verification and test datasets are commonly employed to improve and assess network performance. The validation dataset is a sample of examples, independent of the training dataset, used to evaluate the model fit on the training dataset. It is still used in the development stage and allows for adjusting the network's higher-level hyperparameters. The test dataset, also independent, allows for evaluating the final model and illustrates various real-world cases the model would face.

2.2 Mathematical background

DNNs are often described as "black boxes" since it is unlikely to apprehend what happens in each of their neurons. However, DNNs are able to figure out the best parameter values through calculus. Mathematical aspects allow the practitioner to define blocks composing a network architecture, assess its performance, and rectify its parameters. Therefore, this subsection based on Ng 2021 introduces fundamentals to understand the network architectures presented afterward.

2.2.1 Neuron weight and bias

An ANN layer is composed of several neurons, connected to all or only part of those of the next layer. For each neuron a weight and a bias are applied to the input, resulting in z, before computing the output through an activation function a (figure 2.2). The weight w and bias b are learnable parameters, where b is a constant.

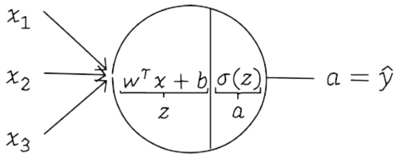


Figure 2.2: Operations occurring in a neuron; source: Ng 2021

2.2.2 Activation functions

The activation function acts like a synapse for the next layer by activating it with its output. These activations can either be linear or non-linear: the early age of DL usually employed the sigmoid function σ (function 1), whereas nowadays, depending on the application, it mainly uses rectifiers, such as the Rectified Linear Unit (ReLU). The latter does not modify the result if it is positive or outputs zero if it is negative.

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{1}$$

In the last layer, the function usually differs since it yields the network output, that is, the final classification result. For binary classification, logistic regression is generally used (based on a sigmoid function), whereas, for multi-class outputs, softmax regression is employed, generalizing the logistic function to several dimensions. The softmax function normalizes the output to a probability distribution.

2.2.3 Cost and loss functions

The loss error function allows for assessing the algorithm performance by defining the error between the prediction $\hat{y}$ and the ground truth y. The loss function L is defined for a single training example, whereas the cost function J does it for the entire training set, averaging the sum of the loss function applied to each training example.

The cost function can have different definitions. A regression model—estimating a variable value based on the other—can employ a regression cost function: for example, mean error (ME), mean squared error (MSE), or mean absolute error (MAE). A convex function can be used to enhance the optimization problem, such as the logistic loss function, illustrated below with its corresponding cost function J:

$$L(\hat{y},y) = -(ylog(\hat{y}) + (1-y)log(1-\hat{y})), \quad \text{and} \quad J(w,b) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}^{(i)},y^{(i)}) \tag{2}$$

where $\hat{y}$ is the predicted label, y is the ground truth, m the number of training examples, w and b the parameters considered.

Performance on multi-classes can be estimated through the cross-entropy (CE) function or binary cross-entropy in the case of two output categories. In all cases, the aim is to determine the parameters w and b that minimize the overall cost function J. For this, w and b are initialized, and the model is trained to learn these parameters by converging to a global optimum.

Combination of loss functions. Duque-Arias et al. 2021 highlights that loss functions can be categorized into two types: statistical and geometrical ones. Jaccard index (Jaccard 1912) and Dice loss (Sørensen 1948), applied to semantic segmentation evaluation, are considered as *geometrically*-based, since they perform a pixel-based evaluation of the predictions, as shown in equations 3 and 4. On the contrary, CE and categorical-focal-loss (CFL) are *statistically*-based since penalizing more or less pixels based on the difficulty to classify them, solving therefore class imbalance issues.

Complementary of both types is beneficial for segmentation tasks suffering from classes' imbalance and where classification performance at pixel-level is crucial (cf, IoU explanation in 3.4.1). Such combined loss functions have been implemented by Yakubovskiy 2019, openly available on "segmentation models" repository that is used for the experiments of part 4.3. Calculations are applied as follows:

$$\text{Dice loss} = (1+\beta^2)\frac{precision \cdot recall}{\beta^2 \cdot precision + recall} \tag{3}$$

where *precision* and *recall* are expressed in function of true positives (TP), false positives (FP) and false negatives (FN) as illustrated in figure 2.3, and β a precision coefficient;

$$\text{Jaccard loss} = L(A,B) = 1 - \frac{A \cap B}{A \cup B} \tag{4}$$

where A is the prediction and B the ground truth for a given class.

$$\text{CFL} = L(gt,pr) = -gt \cdot \alpha \cdot (1-pr)^{\gamma} \cdot \log(pr) \tag{5}$$

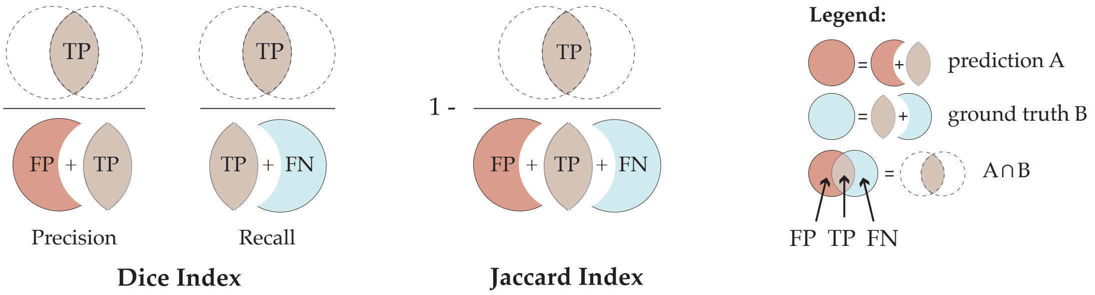


Figure 2.3: Geometrical loss functions, based on TPs, FP, and FNs

where gt is the ground truth, pr the prediction, α is a weighting factor (default value 0.25) and γ a module factor (default value 2.0).

2.2.4 Optimization algorithms

For converging to this optimum, a gradient descent algorithm can be applied. It is composed of a for- and backpropagation sequence. The latter goes back through the layers, computing the gradients of the loss function J with respect to the weight parameters on the whole training set. The derivatives allow computing the slope of the cost function J at the current state of the parameters. Then, those variables are incremented or decremented accordingly, permitting steps towards the steepest descent until convergence. The parameter α, called "learning rate", allows controlling the step amplitude between each elevation.

In logistic regression the two parameters are updated as follows:

$$w := w - \alpha \frac{dJ(w,b)}{dw}, \quad \text{and} \quad b := b - \alpha \frac{dJ(w,b)}{db}, \tag{6}$$

where α is the learning rate, $\frac{dJ(w,b)}{dw}$ is the derivative of J with respect to w and $\frac{dJ(w,b)}{db}$ is the derivative of J with respect to b.

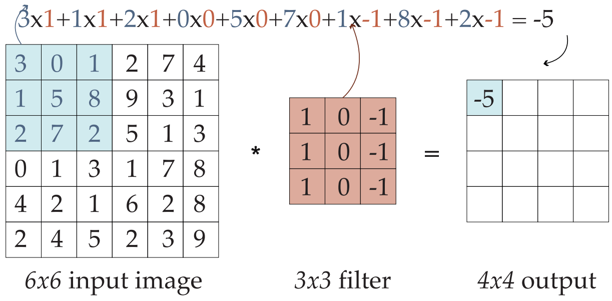
Other optimization algorithms include Stochastic Gradient Descent (SGD), gradient descent with momentum, Root Mean Square propagation (RMSprop), and the Adaptive Moment Estimation (Adam). The latter, widely used, combines the effects of gradient descent with momentum and RMSprop. Although several parameters are involved, the learning rate α is common to them and needs to be adjusted by the practitioner.

2.3 Convolutional Neural Networks

Computer Vision (CV) domain uses DL extensively for image recognition tasks such as image classification (predicting a label per image) and object detection (figuring out the location of an object and drawing a bounding box around it). However, images provide countless input features, and training such a network would require too much computational power. Therefore, convolutional networks (*ConvNets*) have been developed, based on three types of layers: the Convolutional layer (*Conv*), the Pooling layer (*Pool*) and the Fully Connected layer (*FC*).

2.3.1 Convolutional Layer

The first type of layer relies on the "convolution operation" which "convolves" a filter on the pixels of the image by sliding this window ("filter" or "kernel") from one-pixel position to another. The filter—usually of size *3x3*, *5x5* or *7x7*—contains values allowing to detect a specific kind of feature on the image. The operation applied between the input pixel and the filter, denoted by an asterisk in figure 2.4, consists of an element-wise multiplication followed by the summation of the values obtained.



3x1+1x1+2x1+0x0+5x0+7x0+1x-1+8x-1+2x-1 = -5

3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

*

1	0	-1
1	0	-1
1	0	-1

=

-5			

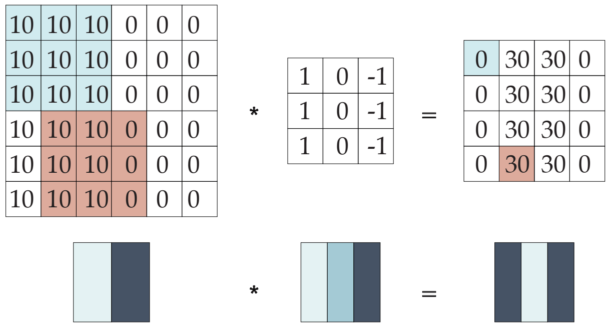
6x6 input image

3x3 filter

4x4 output

Figure 2.4: Convolution operation; adapted from Ng 2021

In figure 2.5, the *3x3* filter allows to detect vertical edges. The intuition is that the filter detects contrasts between the left (bright pixels) and right side (dark pixels) of the central pixel considered. This kernel rotated by 90 degrees would allow the detection of horizontal edges. Similarly, different filter types can be employed to detect various features. To enhance network performance kernel values can be learned instead of manually entered.



10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

*

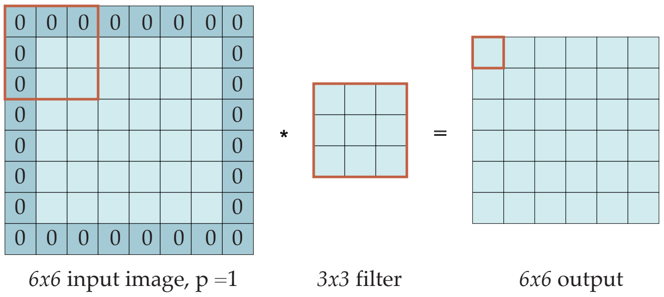
1	0	-1
1	0	-1
1	0	-1

=

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0

Figure 2.5: Convolution operation using a vertical edge detection filter; adapted from Ng 2021

Basic convolution operations can be altered through padding and strides. The first adjoins additional pixel rows (one or several) to the input image to prevent it from shrinking when implementing convolution. A padding size *p* of one would add one row of pixels all along (figure 2.6). Therefore, convolution can be "valid" if no padding is applied; or "similar" ("same" convolution) if the output image is of equal size to the input. On the other hand, the stride *s* impacts the sliding distance of the kernel between two operations. The higher the stride value is, the smaller the output size.



0	0	0	0	0	0	0	0
0							0
0							0
0							0
0							0
0							0
0							0
0	0	0	0	0	0	0	0

6x6 input image, p =1

3x3 filter

6x6 output

Figure 2.6: Padding operation, example of padding value *p* equal to one

Convolution operation can also be applied to a multi-dimensional input such as an

RGB image. In that case, the filter has the same number of channels (or dimensions) as the input. The number of filters used results in the channels' quantity in the output (figure 2.7). Consequently, an RGB image convolved with a single *3x3* filter would result in a one-channel image. To turn the operation into a *ConvNet*, a non-linearity function (like ReLU) and a bias should be applied to the convolved activations from the previous layer, yielding the final layer outcome. The outputs from different filters are stacked together to form a *Conv* layer.

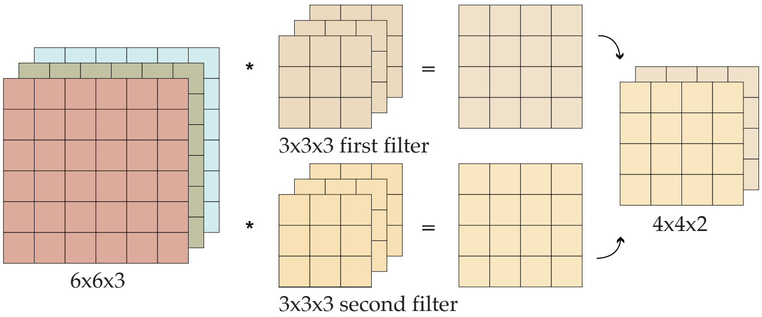
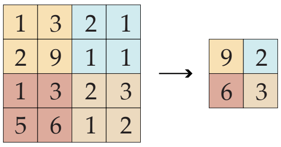


Figure 2.7: 3D Convolutional operation using two filters; adapted from Ng 2021

To summarize, a *Conv* layer is characterized by the filter size, its padding, its stride, and the number of filters used (with each kernel having the same number of channels as the input). The values of the filters stacked together correspond to the weights (used for linear operation), and the bias is unique per filter.

2.3.2 Pooling Layer

Although a CNN may only use *Conv* layers, *Pool*s and *FC*s are usually also implemented. *Pool*s allow to reduce the representation size, speeding up the computation and making feature detection more robust. Essentially, the input is divided into regular regions, based on a filter size and stride value, and the maximum value (*Max Pooling*, figure 2.8) or average value (*Average Pooling*) per region is kept. The intuition is that by keeping a statistically meaningful value per piece, the features detected are preserved. *Pool*s have some hyperparameters (filter size and stride) but no parameters to learn, and the operation preserves the number of dimensions. By convention, a *Conv* layer followed by a pooling operation forms only one layer since only a layer having weights is reckoned.



1	3	2	1
2	9	1	1
1	3	2	3
5	6	1	2

⟶

9	2
6	3

Figure 2.8: Example of MaxPooling; adapted from Ng 2021

2.3.3 Fully Connected Layer

An *FC* layer contains units similar to those previously described, densely connected to the activations' output from the previous layer. Therefore, *FC*s have a lot of parameters. On the contrary, *Conv* layers use shared parameters (that are the values in a filter), and their connections are sparse, making them advantageous. As illustrated in figure 2.9, *Conv* layers followed by *Pool* layers, with *FC* at the end, before a softmax activation function, is a common pattern for CNNs.

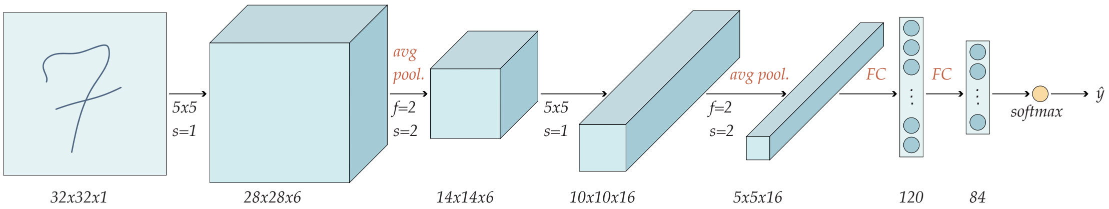


Figure 2.9: Typical ConvNet; adapted from LeNet-5 network (Le Cun et al. 1998)

2.4 CNN architectures

The three building blocks, *Conv, Pool, and FC*, are combined to form architectures, which the CV community is researching to reach better performances. This subsection introduces some relevant for this research.

First of all, LeNet-5, illustrated in figure 2.9, has been developed for document recognition purpose (Le Cun et al. 1998). In 2012, AlexNet extended LeNet-5 into a deeper CNN, achieving great performance on image classification tasks (Krizhevsky et al. 2012). A few years later, the very deep network VGG-16 was created for large-scale image recognition tasks, 16 referring to the number of weighted layers the network contains (Simonyan and Zisserman 2015). The uniformity of this architecture—with systematic rates of size decrease and channel increase between layers—makes it attractive. However, it contains a large number of parameters to train.

2.4.1 ResNets

Very deep networks are hard to train because of the problem of vanishing and/or exploding gradients, making the training error get worse in deep layers. To tackle this issue, skip connections have been researched, allowing to pass the activations functions from one layer to another deeper in the network. Residual blocks using this concept (figure 2.10) have been proposed by He et al. 2015. A *ResNet* is built by stacking together several of these blocks.

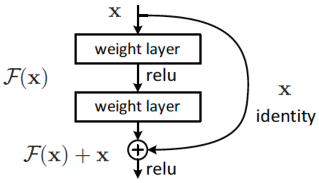

x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 2.10: Building block using residual learning; source: He et al. 2015

2.4.2 Network in Network

The concept of "Network in Network" was developed by Lin et al. 2014, inspiring many architectures later on, including the Inception architecture (Szegedy et al. 2014). The idea is to use a *1x1* convolutional layer (also called Network in Network), allowing to modify of the number of channels according to the number of filters, but not the size (height and width) of an image (figure 2.11). If the number of filters corresponds to the number of input channels, then *1x1 Conv* is useful to learn more complex functions.

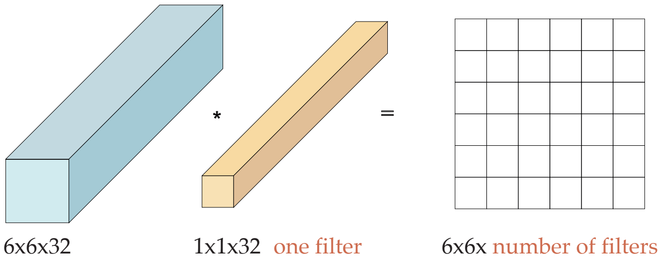


Figure 2.11: *1x1* Convolutional layer; adapted from Ng 2021

2.5 Encoder-decoder network architectures

This section has described classification networks so far, that go deeper through feature detection layers to output a label class for the whole image. However, for semantic segmentation, the final output has a size similar to the one of the input image, assigning a probability per class per pixel. For this purpose, a second "decoding" part is stacked to the "encoding" part of the network. The first part refers to image pixels' encoding into low-level feature maps detected through filters (e.g. lines, textures), resulting in the increase of channel number. Whereas decoding refers to the opposite process, decoding the features back to the image size, allowing high-level, global information detection (e.g. object classification as a whole). The decoder part projects the encoder information on the pixel space to output a dense segmentation.

The Fully Convolutional Network (FCN), extending *ConvNets* image classification purpose to segmentation tasks, has been defined to combine deep semantic information and shallow appearance information through the use of skip connections (Long et al. 2015). The authors introduce the concept of deconvolution to upsample the image back to its original size, allowing pixel-wise prediction (figure 2.12). The final prediction layer is linked to lower layers permitting local and dense predictions to be coherent with the global structure.

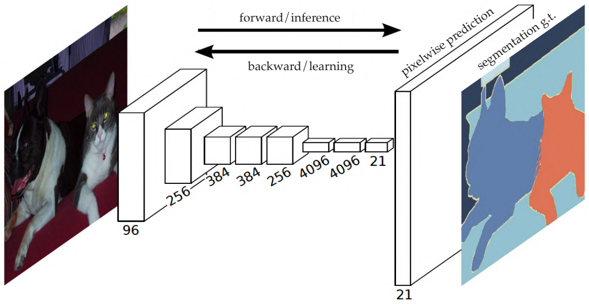


Figure 2.12: FCN architecture, allowing pixel-wise prediction; source: Long et al. 2015

2.5.1 U-Net

U-Net, built upon FCN, elaborates on the decoding part. It uses transpose convolutions for the upsampling process, making the network symmetric (Ronneberger et al. 2015). The shape of the figure 2.13, illustrating U-Net's functioning, has given its name to the network. For simplification purposes, each rectangle represents the height (vertical axis) and the number of channels of the image (horizontal axis), whereas its width is omitted. Therefore, the evolution of the image size and the number of channels can easily be understood. This concept can be applied to different backbone architectures, such as *vgg19* or *ResNet*.

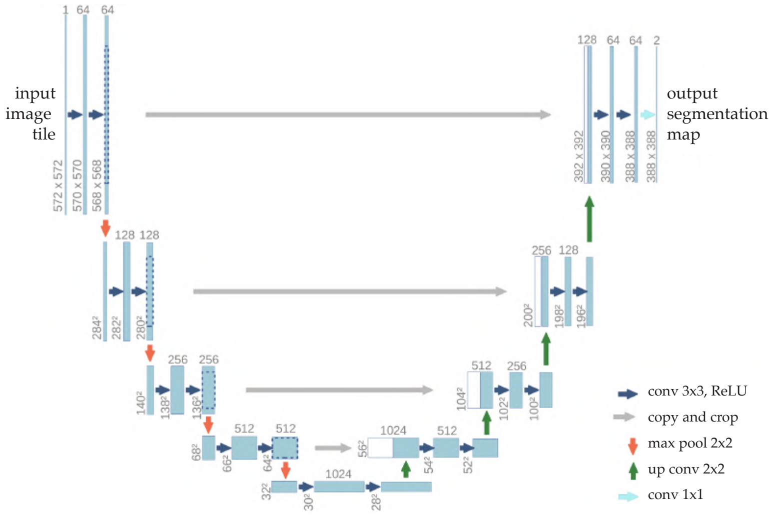


Figure 2.13: U-Net architecture; source: Ronneberger et al. 2015

2.6 Multimodal network architectures

Another aspect of the research on network performances involves multimodalities. Those refer to different data types: image, sound, video, etc. Their combination within NNs opens new and countless possibilities to improve networks' performance since these diverse inputs offer complementary information. Multimodal NNs offer various possibilities: e.g., modeling two modalities jointly, generating a modality from another one, or using a modality as a label for the second one (eg, the sound of a video, labeled by the corresponding images) (Dean 2017).

In the case of RS and 3D city modeling, modalities include aerial images, hyperspectral data (HIS, capturing light within wider bands of the electromagnetic spectrum than RGB images), Light Detection And Ranging (LiDAR) point clouds, and semantic 3D models. As reviewed by Yuan et al. 2021, several works have researched multimodalities to improve semantic segmentation of aerial images. Some of them use HIS data fusion to enhance land-cover classification (Roy et al. 2020) or roof material identification (Nimbalkar et al. 2018).

2.6.1 Height data fusion to CNNs

LiDAR point clouds (PC) provide discrete 3D data acquired by measuring the range of light pulses emitted by a laser-scanning device. Usage of those points in ML can either be achieved through 3D architectures like 3D CNN (an application of CNN theory to 3 dimensions (Guo et al. 2020)) or through converting it to 2D information by means of raster representation. For this, points can be projected to depict a "height map" in case of parallel data or "depth map" in case of perspective data. The first implementation case allows 3D shape classification, object detection, or point segmentation (Guo et al. 2020), whereas the second allows enhancement of 2D segmentation results by providing complementary data. This subsection explores some techniques of the second approach.

Combining RGB and height data can be achieved through data fusion, implemented for the first time by Hazirbas et al. 2017 for RGB-Depth camera images. Gu et al. 2021 distinguishes two types of data fusion to the network architecture. The most frequent one adopts a bottom-up approach (figure 2.14, left), where the additional data type is encoded through another branch of the network. The top-down approach (figure 2.14, right) feeds encoded information of the different branches to their corresponding decoding depth.

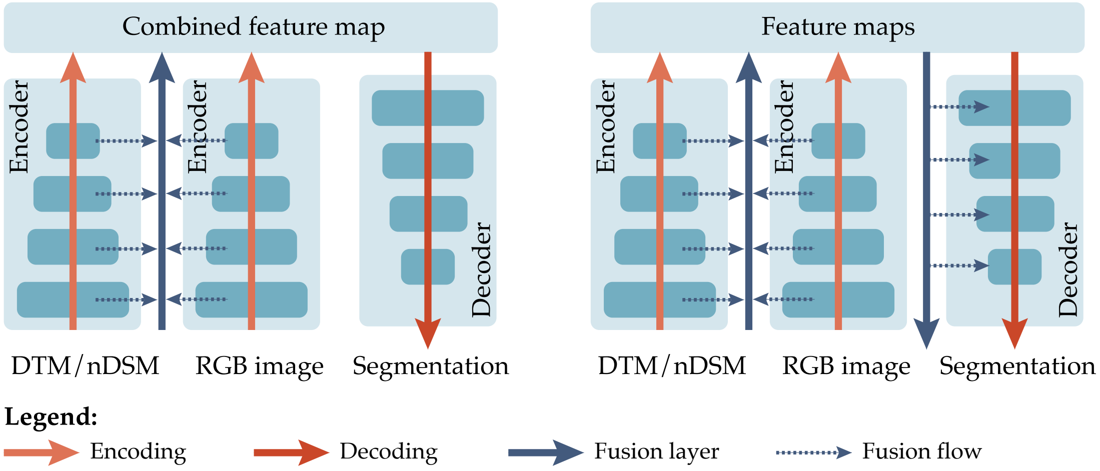


Figure 2.14: Two different fusion approaches: left, the bottom-up and right, the top-down fusion methods; adapted from Gu et al. 2021

2.6.2 Bottom-up Fusion Network architectures

Hazirbas et al. 2017 implemented depth data fusion for the purpose of perspective image segmentation. The author tested stacking depth information on an additional channel. And fusing it through a parallel encoder (or "auxiliary branch"), which activations are aggregated to those of the main RGB branch before decoding (figure 2.15). Data fusion yielded better results.

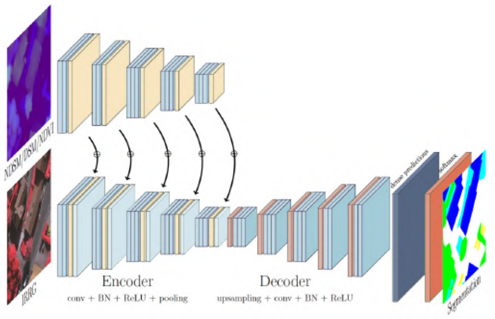


Figure 2.15: FuseNet architecture encoding auxiliary data to the main datasource; source: Audebert et al. 2018 adapted from Hazirbas et al. 2017

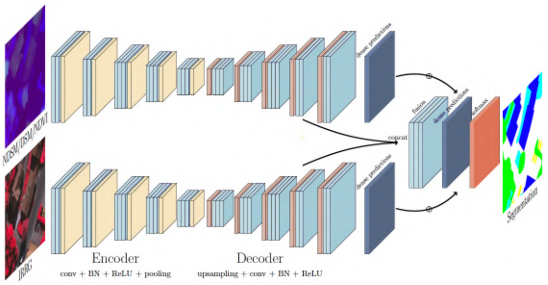


Figure 2.16: Virtual-FuseNet architecture, a symmetric version of data fusion; source: Audebert et al. 2018

Similarly, for land-cover classification, other authors tested height data fusion to RGB images: Digital Terrain Model (DTM), Digital Surface Model (DSM), and normalized DSM (nDSM), which depicts height data relative to ground level. For instance, Virtual-FuseNet was developed to fuse nDSM, DSM, or NDVI (difference between visible and near-infrared) data to aerial images, tackling the asymmetry issue of FuseNet (Audebert et al. 2018). Both inputs are not hierarchized anymore, thanks to the use of a third "virtual" branch, aggregating activations of both branches through fusion layers (figure 2.16).

Another author compared land-cover classification performance of a network using respectively additional Digital Terrain Model (DTM), Digital Surface Model (DSM), normalized DSM (nDSM) and standardized nDSM (StdnDSM) fused to RGB images (figure 2.17, Zhou et al. 2019). In StdnDSM, height values are rescaled within the bounds of RGB values present on the image (0 to 255), so both RGB and height information have an equal impact on the network output. Fusion of StdnDSM yielded better results.

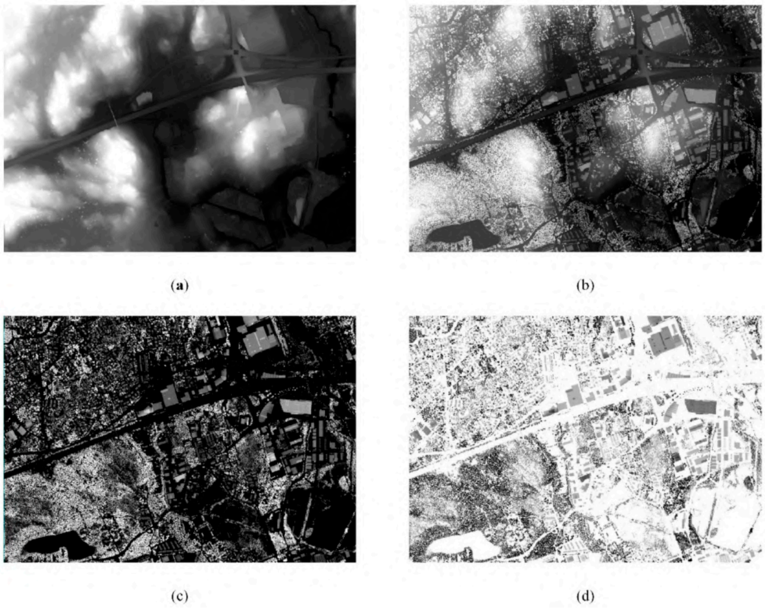


Figure 2.17: Comparison of height data sources: a) DTM, b) DSM, c) nDSM and d) standardized nDSM; source: Zhou et al. 2019

Accordingly, Mulder 2020 concluded that height information fused to the network, rather than stacked, improves land-cover semantic segmentation results on RGB images for some classes (buildings but not water surfaces). The best results are reached using relative height, obtained through DTM pixel-level subtraction from DSM.

2.6.3 Top-down Fusion Network architectures

Gu et al. 2021 developed a light-weighted Top-down Pyramid Fusion Network (TdPFNet), tackling the issue of information loss during the decoding phase, happening in usual multi-branches fusion approaches, as well as the complexity of such networks. Their solution allows guiding the fusion of low-level texture information (in shallow layers) with the high-level semantics of deeper layers. It is possible through fusion layers at each network depth, gathering information from the different encoder branches and feeding them to the corresponding decoding depth (figure 2.14, right). Finally, cumulated high and low feature information from multi-levels are arranged and fused to obtain dense segmentation results. Tests yield the best segmentation results for Potsdam land-use classification when fusing the Open Street Map (OSM) dataset. It also showed that TdPFNet is less complex than other state-of-the-art models (FuseNet, V-FuseNet, etc.), being, therefore, easier to train.

More recently, Yan et al. 2022 developed an “Efficient Depth Fusion Transformer for Aerial Image Semantic Segmentation” (called *EDFT*) that decreases the number of parameters, computational and memory costs through downsampling techniques applied to depth information for feature extraction.

2.7 Roof superstructure detection and modelization

Although (multimodal) CNNs have been extensively researched for land-cover, land-use and perspective view semantic segmentation and prove satisfying results, they are seldom employed for roof segmentation tasks and even less for roof superstructure detection. It can be explained by the necessity of high-resolution (HR) input data and the need for large training datasets requiring extensive preparation work.

2.7.1 Input data resolution and availability

Roof superstructure detection through the DL method requires (Very) High-Resolution (V-HR) images and/or LiDAR data.

In remote sensing, HR to VHR satellite images involves a representation from 30cm to 5m per pixel according to Kraetzig 2021. In the first case, a chimney would approximately be represented by 4 pixels, illustrating the need for VHR images for roof superstructure detection. Since the launch of the first HR satellite sensor in 1999 (Ikonos), finer multispectral and panchromatic images are acquired each year (Marcello and Eugenio 2019). It yields corresponding research and development of various use-cases. Marcello and Eugenio 2019 gathers such works and mentions, among other processes, image segmentation and classification, that facilitates change detection, land monitoring, and urban mapping. Although VHR images exist for multiple areas, they are not always publicly available nor provided as orthophotos, making them complex to use as combined with other datasets.

Additionally, LiDAR data acquisition requires extensive work since airborne laser scanners require their plane to fly over broad areas. Then, point processing and classification are necessary to make them usable for different purposes. The availability and quality of such datasets are therefore limited or very recent. Despite these challenges, European countries are increasingly investing in their acquisition. For example, in the Netherlands, the fourth national LiDAR dataset (AHN4) will be fully available by the end of 2022 with a resolution of 10 to 12 points/m^2 (AHN 2022). In France (13 times larger than the Netherlands), a LiDAR national dataset is being acquired since 2021, fully available by 2026 with an average resolution of 10 points/m^2 (IGN 2022). In Germany, to our knowledge, no national LiDAR dataset is available, but Bavaria proposes one of at least four points/m^2 (LDBV 2021b) that will be used for this research.

2.7.2 Roof superstructures' detection through CNNs

Semantic segmentation applied to RGB images at the scale of buildings' depiction has been practiced for several purposes but, to our knowledge, not for superstructure detection in general.

First of all, it has been used for geometrical purposes: to detect building footprints (Wei et al. 2020), extract roof segments (surfaces with their orientation) (Lee et al. 2019) or roof edges (Ahmed and Byun 2019). The latter can be used to model buildings in 3D: Alidoost et al. 2020 use DL to extract nDSM and rooflines from mere RGB input to reconstruct LOD2 buildings.

Secondly, DL methods applied on roof images have been used for specific application cases. For instance, PV-mapping by detecting solar panels already installed on roofs (Ioannou and Myronidis 2021), or roof material analysis using various spectral data input (Nimbalkar et al. 2018). However, to our knowledge, no DL method has been developed to detect simultaneously all types of roof installations. Therefore, no training data have been found for such a purpose, and manual labeling was required to carry out experiments.

2.7.3 Roof superstructures' modelization

Once detected, roof superstructures can be modeled in 3D. The four LODs described by CityGML standards have been refined by Biljecki et al. 2014 to describe nuances of such models. Each subdivided LOD depicts a model suitable for different application purposes. 3D city models are available in Europe at different LODs, but mostly LOD1 or LOD2 (e.g. Bavaria). Availability of LOD2.2 models that include dormers is rare in large areas since complex to generate automatically. To our knowledge, the Netherlands is the only country that has such a model nationally available (3D BAG) (tudelft3d 2021). A 3D cadastral dataset has been researched, downloadable since 2021 in 3 different LODs—1.2, 1.3 and 2.2 (tudelft3d 2021). Peters et al. 2021 describes the method used for the automatic generation of 3D buildings based on cadastral and AHN LiDAR PC datasets: although the process requires building edges detection, no ML technique is used.

The method of photogrammetry is commonly used to reconstruct buildings in 3D from photographs taken at different angles (e.g. Google Earth). However, Kudinov 2021a proposes a ML-based method developed upon R-CNN (Gkioxari et al. 2020) to reconstruct 3D-mesh buildings including roof superstructures from mere orthophotos (Kudinov 2021a, Kudinov 2021b). Input data include RGB images and nDSM, whose efficiency is tested separately and combined. The latter provides the best results (figure 2.18), allowing even to model overhangs, therefore reaching LOD2.3 representation according to Biljecki et al. 2014. However, it tends to smooth out the edges, is sensitive to image noises, to shadows, and does not include semantics. As a result, the use-cases of such a mesh model are limited. Nonetheless, it illustrates the potential of ML for accurate and detailed 3D modelization.

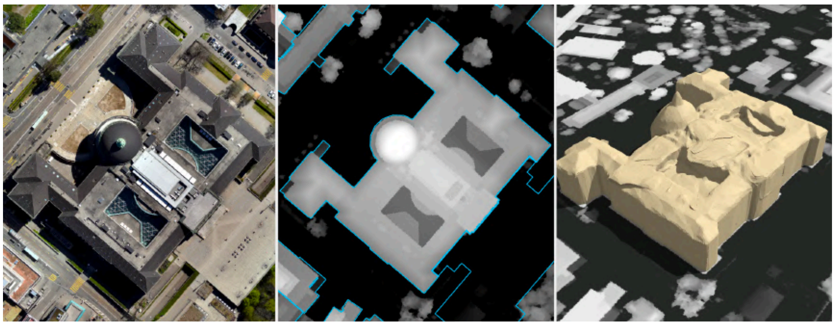


Figure 2.18: 3D mesh-reconstruction (right) obtained by fusing RGB aerial image (left) and nDSM (middle); source: Kudinov 2021b

2.8 Current status of the pipeline

Finally, the way ML theory is implemented in the existing pipeline for buildings' PV potential assessment is described in parallel with the different works carried out in the frame of the project. Implementation details and results until 2021 are provided by Krapf et al. 2021.

2.8.1 Pipeline overview

Figure 2.19 illustrates the whole pipeline for PV potential assessment. It includes the estimation of the physical suitability of roofs for PV panel installation based on radiation data and roof geometrical potential. Furthermore, the pipeline also considers technical aspects of PV installations and electricity prices on the market.

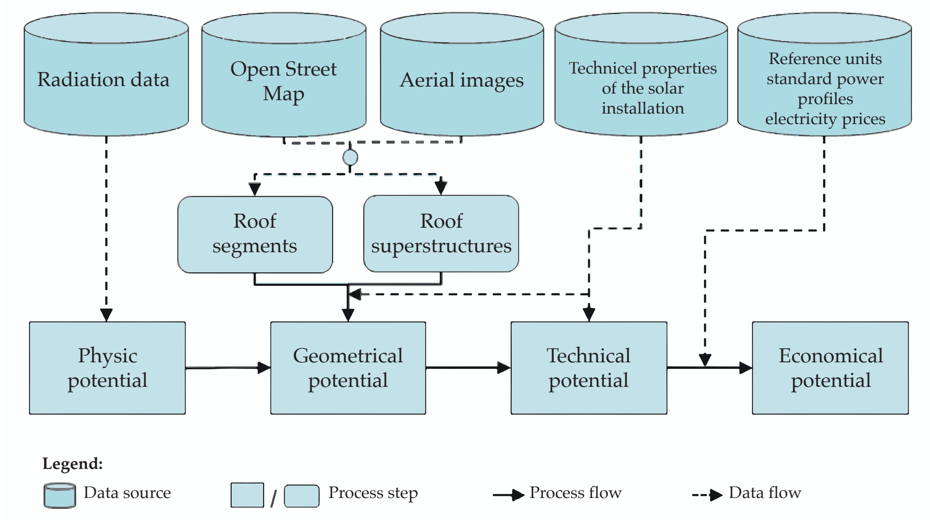


Figure 2.19: General pipeline for PV potential assessment; adapted from Prummer 2021

The geometrical assessment consists of two aspects: roof azimuth and roof surface available. Each is estimated by means of a DNN applied to RGB aerial images. The azimuth categorizes the roof orientation into 16 classes: North, East, South, West, and the variations. Secondly, to estimate the roof surface available, the processing steps involve the detection of roof superstructures through DNN and their subtraction from the total roof surface. Roof superstructures include all visible elements from aerial views, varying from architectural specificities (chimneys, dormers, windows, etc.) to technical ones (PV installations, ventilation, etc.).

2.8.2 Geometrical assessment implementation

First of all, segmentation of the roofs into azimuth (figure 2.20) has been implemented by Kemmerzell 2020, based on Lee et al. 2019, using a U-Net architecture with a *ResNet-152* backbone. The same author has also implemented PV technical and economic assessment.

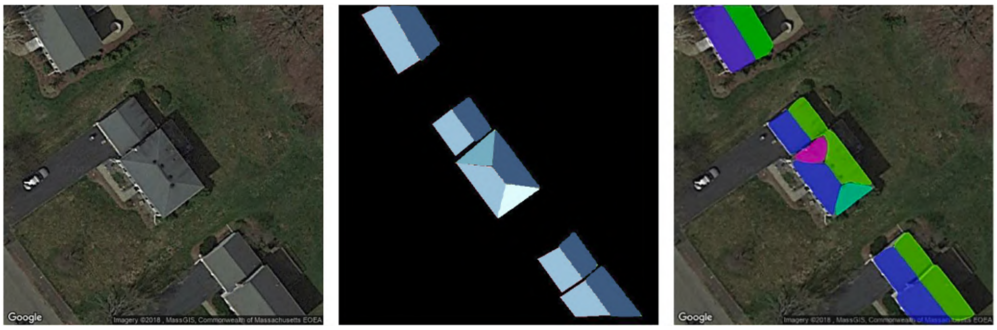

Google
Imagery ©2018 , MassGIS, Commonwealth of Massachusetts EOEA
Google
Imagery ©2018 , MassGIS, Commonwealth of Massachusetts EOEA

Figure 2.20: From left to right: Google aerial photographs, segments ground truth and segments detected by the network; source: Syed 2020

Then, roof superstructures detection is implemented by Syed 2020, using a U-Net architecture. It results in a single and a multi-class segmentation as illustrated in figure 2.21. Both CNNs use images obtained through Google API as input and focus on the city of Wartenberg (Bavaria), from which training, validation, and test datasets are generated. Since no training dataset is publicly available for roof superstructures, the latter have been manually labeled by the team, and its extension and improvement is an ongoing process since 2018 (Krapf et al. 2021). Next, Prummer 2021 carries out a data-centric approach to improve the network performance.

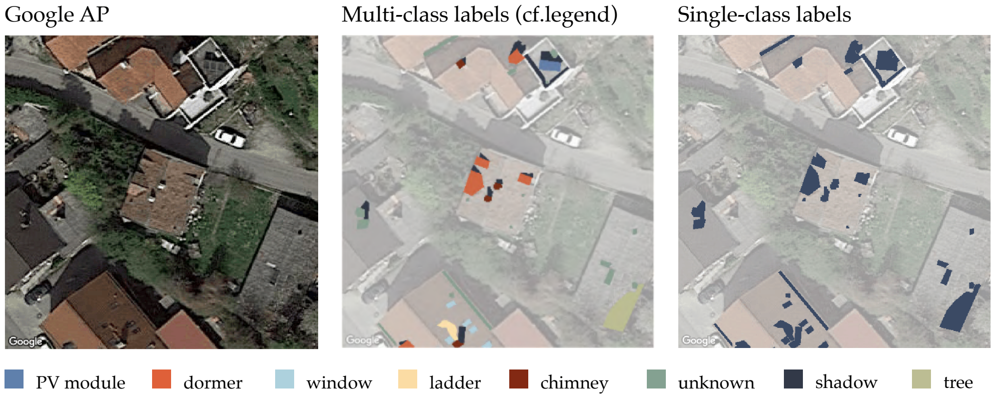


Figure 2.21: Roof superstructure detection into multi-classes (center) and single-class (right)

Finally, in parallel with this work, an automatic generation of training data for roof azimuth segmentation based on the 3D model available is researched (Faltermeier 2022). For this, several encoder backbones are evaluated within a U-Net architecture implementation. Additionally, a thesis is carried out to model in 3D the chimneys and dormers detected by the network for superstructure segmentation (Bruhse 2022).

2.9 Conclusion on the State-of-the-Art

To conclude, this research aims to improve the current U-Net implementation for segmentation of roof superstructures by migrating to an architecture allowing height data fusion.

As demonstrated, numerous architectures have been developed, seeking the best data fusion process, both through results' performance and computational power efficiency. Each month new publications are released, proposing various methods for the fusion of multimodalities. For height or depth map fusion, *FuseNet* stays a common reference and new approaches are often developed upon it. This thesis, therefore, employs this architecture, focusing on evaluating the performance of different height data fused to the network, following a *data-centric* approach. Several height maps are generated at a building scale, and their performance is assessed to determine which is more efficient for roof superstructure segmentation. This purpose, which requires finer geometrical information and image understanding than land-cover or land-use classification, has been hardly explored until now.

3 Methodology and development

The previous chapter provided a scientific background for building up a methodology. This section elaborates on the approach determined to answer the research question.

First, an overview defines the research' general sequence, highlighting the scope of the work. Then, the procedure is detailed through several steps, justifying their necessity and implementation. It includes training data and height data preparation, height data incorporation, and application of the model and pipeline to another test area. Next, the processing pipeline to generate height maps is specified, and their resulting content is analyzed. The final subsection explains the quantitative evaluation of CNN results.

3.1 Research process overview

Figure 3.1 describes the research' sequence.

First, datasets are selected, and the annotations are reviewed. Then, height data maps are prepared to be compatible with RGB images: having similar sizes, value extents [0–255], and coinciding patterns. The existing U-Net implementation is run with different settings, and the model resulting in the best predictions is kept for further comparison. Next, a fusion architecture is tested with various height data inputs: absolute and normalized height, each with different interpolation methods. Results are assessed by comparing them to those obtained through CNN merely based on aerial images. Predictions from each data type are also compared to one another to identify their specificities. Finally, the most performant model is applied to another geographic area for which predictions are modeled in 3D using the algorithm designed by Bruhse 2022.

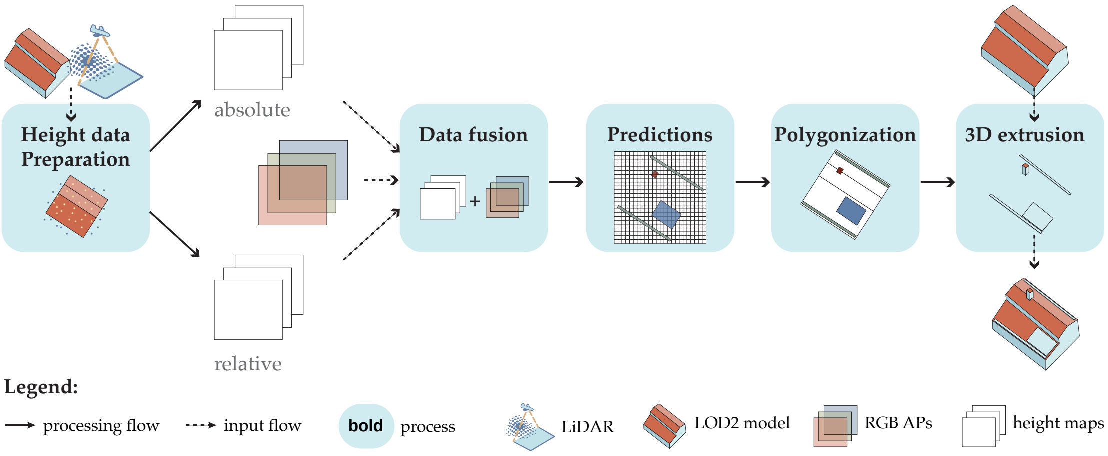


Figure 3.1: Overview of the research process

3.2 Research steps

3.2.1 Training data preparation

First, available datasets are explored to determine which suit best the research needs. Height data derived from LiDAR appears to relate to the Digital orthophotos (DOPs) but not to Google aerial photographs (APs) (cf. figure 3.2). Therefore, the non-ortho-rectified labels are reviewed to coincide with DOPs and height data. Secondly, only relevant superstructure classes are kept. Shadow and tree classes are discarded since the first ones should be learned by the network through height data input, whereas the seconds correspond to trees hanging over the roofs but are no superstructures. The final six classes, from which is computed the model loss, comprise PV modules, dormers, windows, ladders, chimneys and unknown.

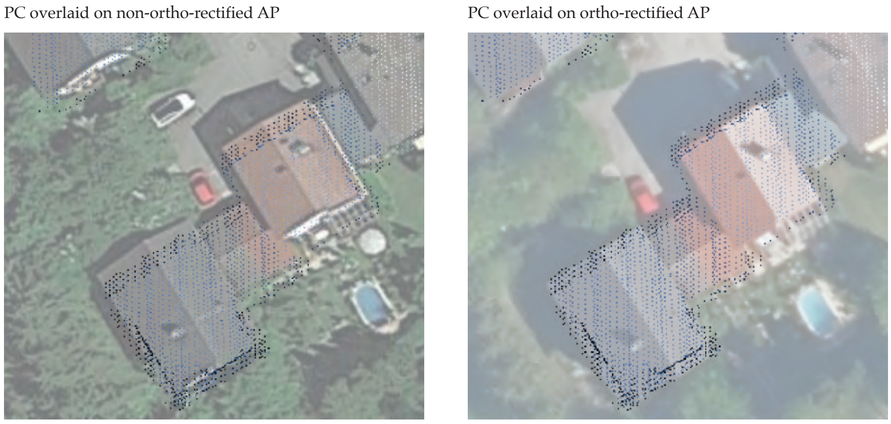

PC overlaid on non-ortho-rectified AP
PC overlaid on ortho-rectified AP

Figure 3.2: Rasterized PC overlaid on non-ortho-rectified and ortho-rectified aerial images

Overall, one can justify DOP usage by several factors. First, to correlate height and image data; Second, when new and higher-resolution APs are available, labels can be reused; finally, the project aims at scalability, which is possible with DOPs since it has a universal definition. When new DOP datasets are published, APs of diverse years could be incorporated and associated with similar labels to create more training data.

All datasets used for the research include DOPs, classified LiDAR PC, and the LOD2.0 model of Bavaria used for relative height computation and the final model's enhancement.

3.2.2 Height data preparation and assessment

Datasets available are processed to generate rasterized height data. First, absolute building heights are extracted from the classified LiDAR PC (LAS classification 6); second, height values are normalized. Subtracting the terrain to the buildings' height would result in accuracy loss since the ground first needs to be interpolated at each building pixel before being subtracted to the absolute building heights. Therefore, normalization is achieved per building using the 3D model available for Bavaria: a new height value is assigned to each point following the concept illustrated in figure 3.3. Ortho-projected LiDAR points within the semantic model footprint get a new elevation value, whereas others have "no data". If both 3D representations correlate in height, the new value is zero; if the point is higher than the 3D model, it gets a positive value; whereas below, it obtains a negative one.

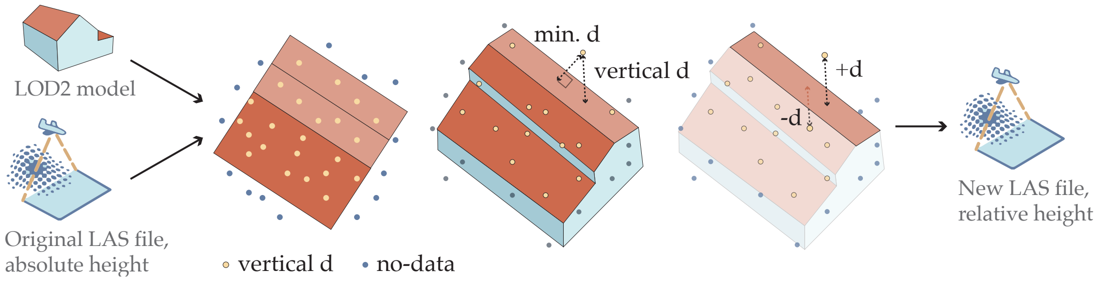

LOD2 model
Original LAS file, absolute height
min. d
vertical d
+d
-d
New LAS file, relative height
∘ vertical d
• no-data

Figure 3.3: Concept for relative height calculation. d = distance

Then, interpolation methods are applied to obtain a similar resolution for height maps than featured by the aerial images. For this, the 3D points are projected on a raster grid, and three different interpolation methods are applied and named as follows in the whole paper: *no* interpolation means that only cells containing a point are given a height value, whereas other cells have no data; *nn* interpolation refers to the nearest-neighbor (NN) interpolation, where each cell is assigned the height value of the nearest point—if this distance is not too high; *idw* interpolation describes the inverse-distance-weighting (IDW) interpolation, that assigns to each cell a height value inverse-squared proportional to the distance between the cell and the points contained within a chosen radius. *idw* interpolation can be differently tuned, but standard parameters are implemented: an inverse proportion to the distance power to two and a radius of 1.5 meters. The algorithm to generate this data is set publicly available at Apra 2022 and described in 3.3.

Finally, the height maps' information content can be evaluated and compared thanks to an Information Entropy (IE) criteria as described by Zhou et al. 2019 who compares this way nDSM with standardized DSM.

3.2.3 Height data integration

Height data fusion experiments employs FuseNet, implemented in PyTorch, which functioning is illustrated in figure 3.4 (Hazirbas et al. 2017). It uses a *vgg16* backbone, that is initialized prior RGB and depth encoding. Only RGB is decoded, whereas depth activations are concatenated to RGB ones during the encoding phase. Concatenation of the activations from the two branches composes the fusion module. A sparse approach is used, meaning that fusion is inserted only once per layer, before pooling (Hazirbas and Aygun 2018).

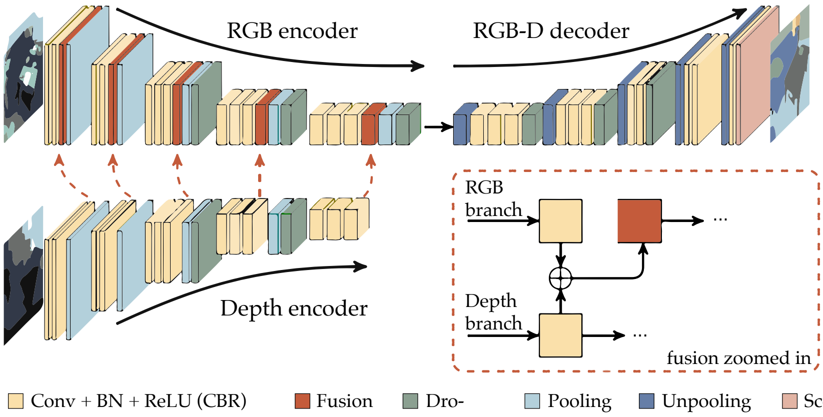


Figure 3.4: FuseNet functioning; source: adapted from Hazirbas and Aygun 2018

The model is fitted with the newly labeled data and run for the Wartenberg test dataset using different height inputs and training parameters. The experiments allow determining the height data source yielding the best predictions. The results are evaluated through confusion matrices, accuracy calculations, and observation.

3.2.4 Dutch test location

The most performing model and input data type are then applied to a test dataset from another geographical location. It is chosen in the Netherlands because all necessary input datasets are openly available. A qualitative analysis of the predictions is performed to refine the research outcomes. Moreover, this step proves the scalability of the pipeline, although the training data should be reviewed to offer the expected results.

3.2.5 3D model generation and application

The detected superstructures for the Netherlands are extruded in 3D, using the implementation of Bruhse 2022. And added to the existing GML model available in LOD2. The extrusions, including semantics describing the superstructure types, allow the user to determine the roof area still available for PV installations. A demonstration employs the enhanced Wartenberg 3D model obtained from modeling the labels. The enhanced model can be employed for other applications.

3.3 Height-data processing

The algorithm is composed of two main parts: computation of points' height relative to the LOD model (further called "relative height"), and generation of raster grids with different interpolation methods, corresponding to the RGB images. The latter is applied to both absolute and relative height (cf. figure 3.5). Finally, the TIF images obtained are converted to PNG of datatype unsigned integer 8 (uint8), so that their values' extent coincide with those of the RGB images [0–255].

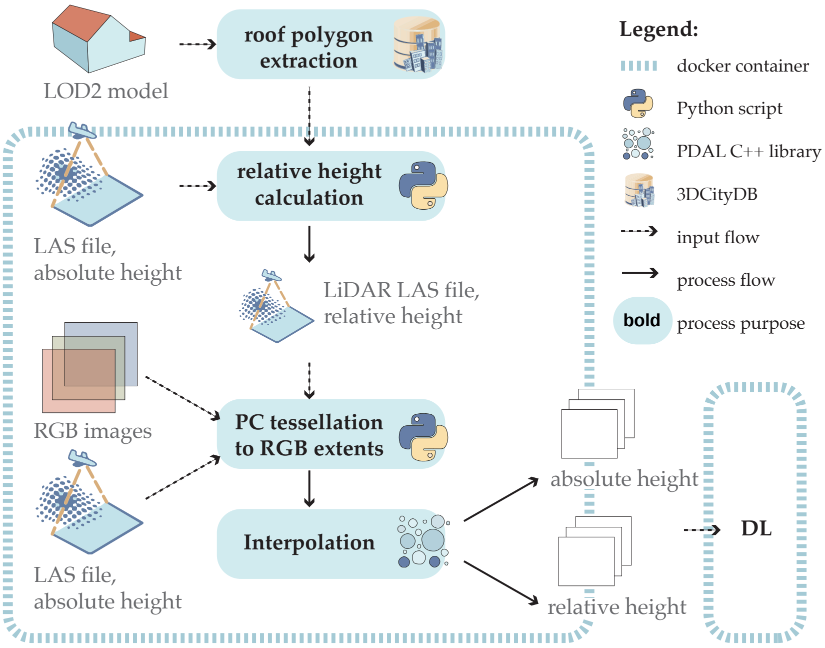


Figure 3.5: Height data preparation: overview of the pipeline

3.3.1 Relative height

Roof polygons' extraction. Since the semantic 3D model is encoded in GML format, roof polygons can be extracted using the geo-database management tool 3DCityDB (Yao et al. 2018). The whole model for Wartenberg is loaded in the database using the corresponding "importer/exporter" tool. SQL queries indicated in the snippet 1 are carried out to retrieve the roof surfaces and their corresponding geometries projected in 2D with PostGIS function "st-force2d". The resulting table, containing roof polygons' unique id, 3D and 2D WKT geometries, is exported as CSV file.

```
CREATE TABLE wartenberg.query1 AS
    (SELECT  sg.id, sg.gmlid, sg.parent_id, sg.geometry AS geom
    FROM     wartenberg.surface_geometry sg,
             wartenberg.thematic_surface ts
    WHERE    ts.lod2_multi_surface_id = sg.parent_id
             AND ts.objectclass_id = 33);

CREATE TABLE wartenberg.query2 (geom_3d_id, geom_3d, geom_2d) AS
    (SELECT id, geom, st_force2d(geom)
    FROM wartenberg.query1);

SELECT  geom_3d_id AS poly_id,
        ST_AsText(geom_3d) AS geom_3d,
        ST_AsText(geom_2d) AS geom_2d
FROM wartenberg.query2;

-----> export resulting table

DROP TABLE wartenberg.query1, wartenberg.query2;
```

Listing 1: Extraction of roof polygons from Wartenberg GML 3D City model using SQL

A semantic model encoded in CityJSON format (like the 3D model of the Netherlands, tudelft3d 2021) requires another parsing solution. However, since *json*-based encoding aims to facilitate information extraction, more practical solutions, directly through Python, could probably be established.

Relative height calculation. The polygon file is further processed through a Python script that also parses LiDAR points with laspy module. The underlying polygon of each point is retrieved in 2D through a "polygon contains point" test. If the point has no underlying polygon, it is assigned a new height value no data. Otherwise, the distance between the 3D point and its corresponding 3D roof polygon is computed.

The distance can be either "vertical" or "minimum" (*min*). The first one corresponds to the altitude difference between the point (z_{actual}) and its expected elevation given its x and y coordinates and the equation of the roof plane. It differs from the minimum distance, which is the distance of this point to the roof surface along the plane's normal axis passing through it.

$$\text{vertical distance} = z_{actual} - z_{expected} \tag{7}$$

Distances are negative if the points lie under the polygon. Moreover, the steeper the roof plane is, the higher the vertical distance is than the minimum one. In the case of horizontal planes, both lengths are similar.

The first height type is preferred since most installations, like chimneys, are installed along a vertical axis. The algorithm generates a new LAS file with these distances replacing height values z. Interpolations are applied to both original and new LAS files to obtain absolute and relative height grids, respectively.

3.3.2 Interpolation

The input LAS files are merged, if applicable. The resulting file is cropped into mini-PC corresponding to the boundaries of the TIF images. Each of them is then parsed using laspy library and rasterized to obtain height maps. Point Data Abstraction Library (PDAL) pipelines are used to perform *no* and *idw* interpolations, whereas *nn* is computed "manually" with Python. *No* interpolation is achieved by applying a radius search equal to the cell half-diagonal, resulting in more cells with height information than there actually should be. However, a smaller radius search results in information loss. Hence, the first option, although not perfect, is kept.

Image conversion. Conversion from float to uint8 is achieved through normalization of all values per image. First, the algorithm subtracts the lowest image value from each pixel; then, it divides all of them by the maximal value; finally, it multiplies all results by 255.

$$data = \frac{data - min(data)}{max(data)} \times 255 \tag{8}$$

3.3.3 Height data content evaluation

Figure 3.6 presents some result's visualization, whereas the split maps in figure 4.6 depicts images missing height information.

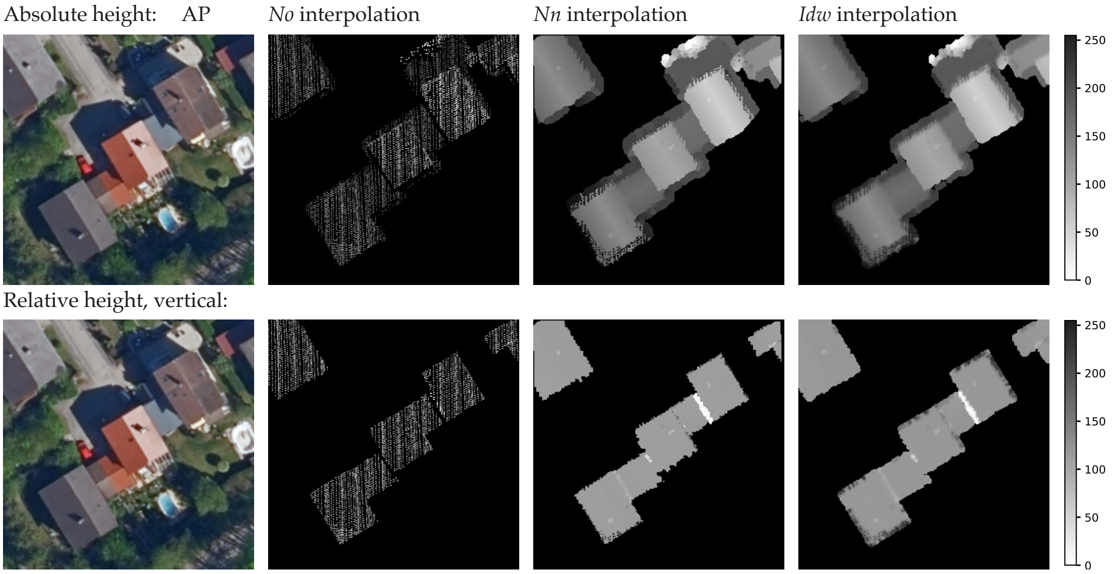


Figure 3.6: Height maps' visualization for an example building

Qualitative assessment. Absolute height data grids provide more context for building understanding than relative heights. The latter is highly dependent on how the LOD2 model was generated and its simplification principles. As a result, more reliefs appear than there are in reality. Typical examples are due to roof overhangs: the LOD2 model does not represent overhangs, but LiDAR points capture them, resulting in height differences located at the border between two buildings (cf. figure 3.7, bottom). Other examples are at buildings' junctions: gabled-roof intersections are sometimes simplified, and the crossing between two building roofs is captured in relief in the relative height dataset (cf. figure 3.7, top). As a result, the network could wrongly interpret it as a dormer.

In relative height datasets, dormers and chimneys obtain positive values, whereas some windows are located beneath the roof surface. Glass surfaces, due to their shiny appearance, hardly reflect laser pulses back to the sensor, yielding data gaps. It might give *no* interpolation dataset a chance for the network to detect windows more accurately than interpolated datasets since lacking local height information at the place of windows. However, as observed, PV modules' surfaces reflect laser pulses more than windows and do not lack height data. Height differences are rare for this

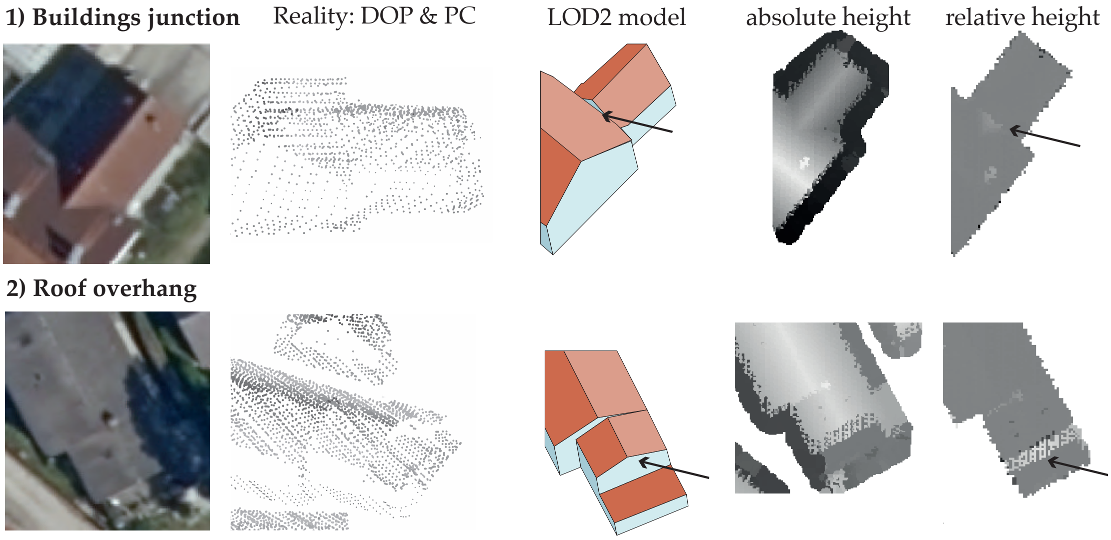


Figure 3.7: Examples of wrong height interpretation in relative height dataset (black arrow)

class, only when panels are forming an angle with the roof plane.

Temporal mismatches and height information lack. The LiDAR data being older than the APs, some tiles have inconsistent height data. Figure 3.8 illustrates an image missing height information for a building (left). The right images illustrate the opposite case: if a building was destroyed between the two datasets' acquisition or ground points were wrongly classified. The first case is recurrent, and the split maps in figure 4.6 depict 63 tiles fully missing height information.

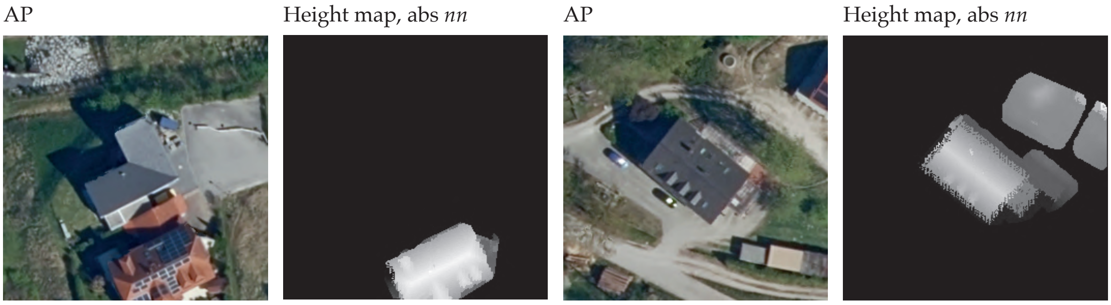


Figure 3.8: Examples of height data mismatch with the RGB image

Additionally, small reliefs like chimneys are sometimes missing since their surfaces are reduced and were not hit by any laser pulse. In any case, chimneys' surfaces observed on the height maps are smaller than those on the RGB images. On the other hand, figure 3.9 shows a rare example of missing dormers, probably due to PC acquisition issues.

Information Entropy. Information entropy (IE) allows for objective assessment of the information content of an image by means of a kernel. The Python implementation described in Maucher 2013 is used and applied to all height datasets, with a kernel size of 3. The filter is applied to each input pixel, and cell values within the considered region of 3×3 are assessed by computing their entropy E. For this, the probability p of each value that occurs within the kernel is calculated, forming the set of probabilities P; then, E is computed as:

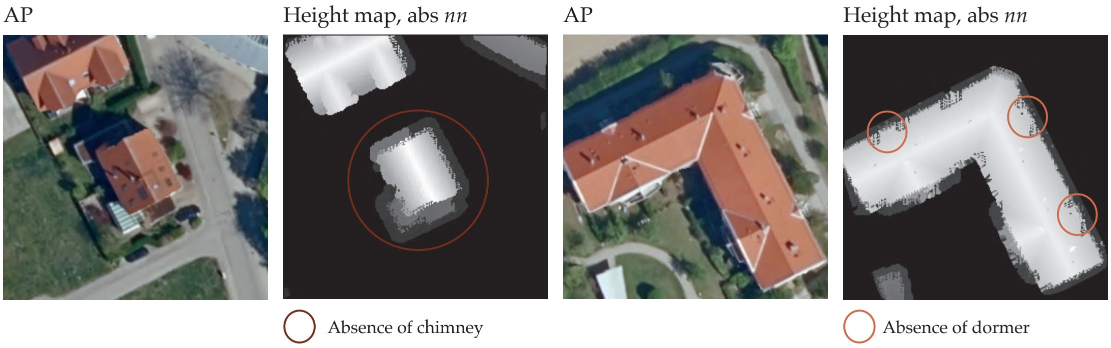


Figure 3.9: Examples of height data lack

$$\mathrm{E} = \sum_{p \in P} log_2(\frac{1}{p}) \tag{9}$$

All height datasets, including those of minimum height, are assessed, and results are gathered in table 3.1: the mean IE is first calculated per height image encoded in uint8. Values of zero are ignored for mean computation since numerous cells have no data. Then all means are averaged per dataset. The aim is to observe if there are correlations between a dataset IE and the results obtained later through the corresponding dataset fusion to the network.

height type	**dataset**	**kernel-size**	**mean IE**
absolute	*no* interpolation	3×3	**1.36**
absolute	*nn*	3×3	**1.32**
absolute	*idw*	3×3	**1.89**
relative, min	*no* interpolation	3×3	**1.09**
relative, min	*nn*	3×3	**1.16**
relative, min	*idw*	3×3	**1.78**
relative, vertical	*no* interpolation	3×3	**1.09**
relative, vertical	*nn*	3×3	**1.17**
relative, vertical	*idw*	3×3	**1.79**

Table 3.1: Mean IE per height dataset, based on 100 corresponding building images

To conclude, absolute height generally contains more information than relative height. Regarding the different interpolation methods, *idw* contains richer features, whereas *nn* contains the least information for absolute height and *no* interpolation the least for relative height.

Additionally, IE per pixel can be visualized for qualitative comparison: figure 3.10 depicts corresponding images from different datasets. Figure 3.11 and mean IE values, indicate that datasets with minimum and vertical distances comprise comparable information. It can be explained by the conversion process to uint8, smoothing out their differences. Since vertical height shows a slightly broader value range and makes more sense regarding the installations' axis, it is kept for the experiments. Therefore, "relative height" refers further to the vertical one.

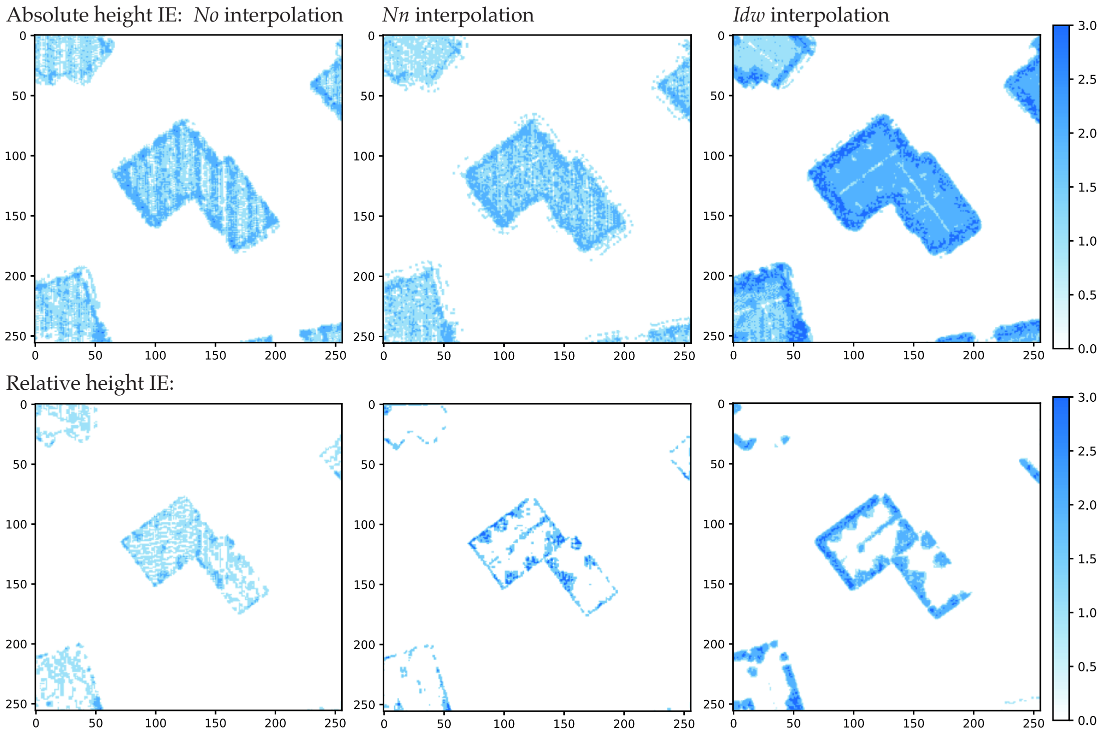


Figure 3.10: IE visualization for an example building, absolute and relative height with different interpolation methods

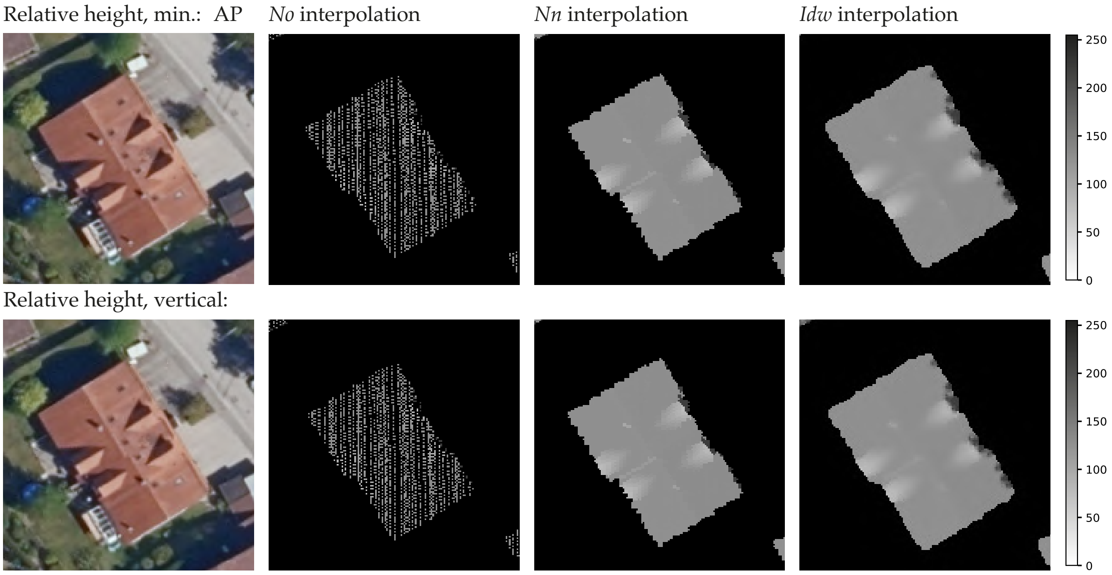


Figure 3.11: Example zoomed in to compare minimum and vertical distances converted to uint8

3.4 Evaluation and comparison of the network's results

Segmentation results are analyzed pixel-wise through confusion matrices, considering model predictions for each class separately. Since classes are imbalanced, with a large majority of the background (no superstructure), such a matrix is a relevant assessment tool. It is two-dimensional, as illustrated in figure 3.12. Each cell contains a number of pixels corresponding to the ground truth (GT) class (read on the y axis) and to the predicted class (read on the x axis). Following the x axis, a column corresponds to the number of pixels the model predicted as belonging to this class. However, only those pixels whose y axis class is similar are correctly classified. They are called True Positives (TPs). The other values of that column are False Positives (FPs) since the GT shows they belong to another category. Finally, reading from the y axis, all the values of a row that do not correspond to the same class on the x axis are called False Negatives (FNs) since the model did not predict them as belonging to the proper category.

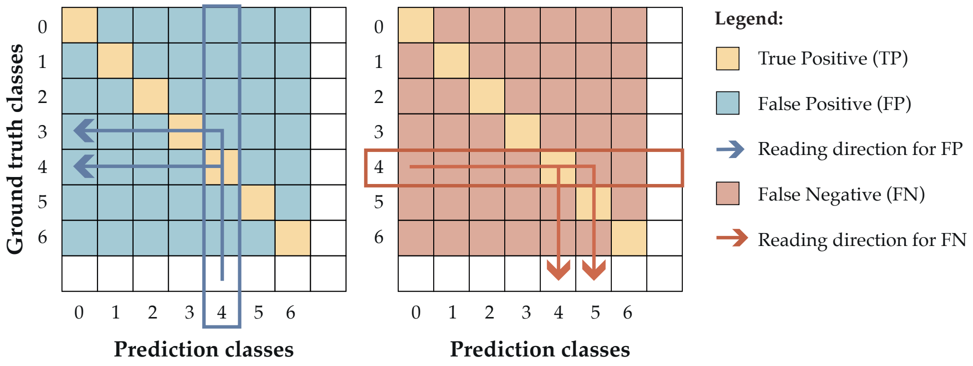


Figure 3.12: True Positives, False Positives and False Negatives in a confusion matrix

3.4.1 Image level and dataset level evaluation

The results of each image tested can be evaluated. As a result, the approach to prediction evaluation on the whole dataset can be two folds. The *micro* approach to the dataset consists in summing up all the image confusion matrices before computing the accuracies from it. In contrast, the *macro* one consists in calculating the accuracies for each image individually before deducing global accuracy values as the mean of that of all images. The first approach considers each pixel in the final accuracy results, whereas the second synthesizes information at the image level. The first approach providing a more accurate global evaluation with no previous generalization is chosen.

Similarly, the evaluation on the image level can be two folds: the *micro* approach to the image consists in summing up all occurrences of TPs, FPs, and FNs to compute accuracies; whereas the *macro* one calculates the accuracies per class before averaging them. The second approach is adopted to calculate the accuracies per class and the IoU score. The latter is based on a *micro* dataset approach and a *macro* image approach.

For each class, accuracies can be calculated, either based on the TPs and FPs (prediction accuracy) or the TPs and FNs (accuracy on GT). Each combination independently corresponds to the total number of pixels on the image.

- **Prediction accuracy.** By combining TPs and FPs to evaluate the prediction, the percentage of pixels detected in a class that actually belong to it can be calculated. For each column, the accuracy is calculated as:

$$\text{prediction accuracy} = \frac{TPs}{TPs + FPs} \tag{10}$$

  The result is called "prediction accuracy" in the rest of the paper and corresponds to the last row of the confusion matrices, scaled from 0 (0 % accuracy) to 1 (100 % accuracy).

- **Accuracy on ground truth.** Combining TPs and FNs allows to evaluate, given the GT, what percentage of the actual pixels per class are detected as belonging to this class by the model. For each row, the accuracy is calculated as:

$$\text{accuracy on GT} = \frac{TPs}{TPs + FNs} \tag{11}$$

The result is called "prediction on GT" in the rest of the paper and corresponds to the last column of the confusion matrices, scaled from 0 (0 % accuracy) to 1 (100 % accuracy).

- **Overall accuracy.** Overall Accuracy (OA) requires a micro approach and corresponds to the total number of TPs over the total number of pixels. It is not a determining criterion in the present case since it does not consider class imbalance, making the background class too impactful. Confusion matrices of the results chapter indicate that this value is always between 0.98 and 1.

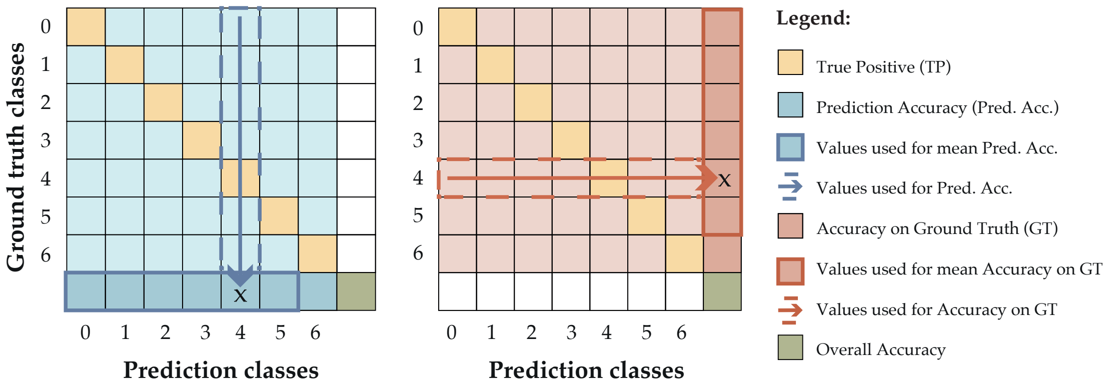

Ground truth classes
0
1
2
3
4
5
6
x
0 1 2 3 4 5 6
Prediction classes
0
1
2
3
4
5
6
x
0 1 2 3 4 5 6
Prediction classes
Legend:
True Positive (TP)
Prediction Accuracy (Pred. Acc.)
Values used for mean Pred. Acc.
Values used for Pred. Acc.
Accuracy on Ground Truth (GT)
Values used for mean Accuracy on GT
Values used for Accuracy on GT
Overall Accuracy

Figure 3.13: Accuracy computation from confusion matrices

3.4.2 Intersection over Union

Intersections over Unions (IoU) refers to the number of pixels correctly classified according to the GT, over the union of all the classification results, as follows:

$$\text{IoU} = \frac{TPs}{TPs + FPs + FNs} \tag{12}$$

In this calculation, information is synthesized, but the denominator is higher than the total number of pixels, resulting in accuracy decreases. This operation is used in experiments to compute IoU evolution. However, for more accurate analyses, the two accuracy criteria are also considered independently through confusion matrices.

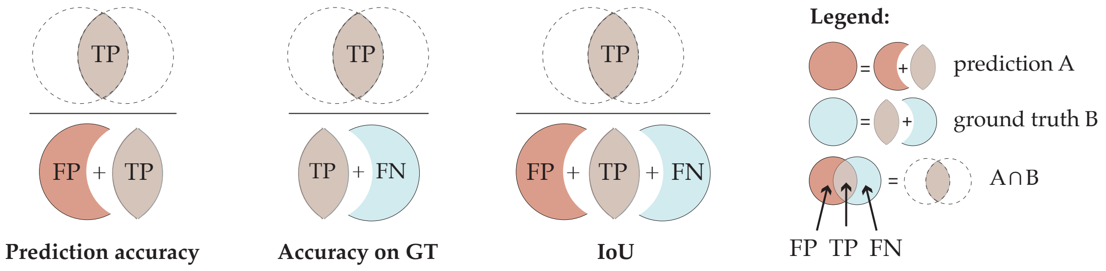


Figure 3.14: Illustration of accuracies and IoU score based on TPs, FP, and FNs

4 Implementation and experiments

The preceding chapter defined the methodology, its scope, and processing steps. This section presents implementation details, the datasets used for DL experiments, and investigations using DL frameworks. It aims at defining the most performing settings for height data incorporation through FuseNet.

First, an overview of the pipeline is provided together with the required tools. Secondly, input datasets are listed, including their resolution and geographical location. The content of training data is described and statistically analyzed. Then, an algorithm is implemented to distribute the annotated data into sets used in the following experiments. The second part introduces U-Net experiments, their scope, and results, followed by the FuseNet experiments allowing to determine the best parameters for height data fusion.

4.1 Implementation overview

Height data processing is achieved through Python scripts and using the C++ library PDAL. Extraction of roof polygons from the GML 3D model and geometry projection in 2D is done with SQL, using 3DCityDB (Yao et al. 2018) and PostGIS functions (PostGIS 2022). The final tables for polygons and LAS files are inputs for the following processing scripts.

ML is achieved in Python language. First, as base for later fusion comparison, a U-Net implementation is carried out, using Keras API built upon TensorFlow (Chollet et al. 2015). The existing project repository is used for training and prediction. For data fusion, FuseNet implementation within PyTorch framework is employed (Hazirbas and Aygun 2018).

Furthermore, model training is done on a remote server, using Graphics Processing Units (GPU), speeding up the process up to 20 times in comparison to a Central Processing Unit (CPU). All requirements are set up by means of Docker containers: one for height data processing requiring GDAL (Geospatial Data Abstraction Library, Rouault et al. 2022) and PDAL (PDAL Contributors 2020), one for ML and GPU leverage. The whole process is illustrated in figure 4.1.

4.2 Datasets

The project aims at using datasets openly available, so the processes can be applied to other geographic areas. This subsection introduces Wartenberg datasets used for the experiments, labeling, and Dutch datasets used to apply the final model to a different geographic area.

4.2.1 Wartenberg datasets

Open datasets for Bavaria are available on LDBV 2021c for Bavaria. However, the LOD2 3D model is not freely available, but was specially provided for research purpose by the State Office for Digitization, Broadband and Surveying (LDBV). Nonetheless, in other countries like the Netherlands, this model is openly available and the method developed in this work can be applied (tudelft3d 2021).

Among available datasets, this thesis utilizes the three following:

- **APs.** RGB photographs can be downloaded through Web Map Service (WMS) from the Bavarian geoservices portal (LDBV 2021c). Although they are provided as true orthophotos, their resolution is only of 20cm/pixel, which is less than

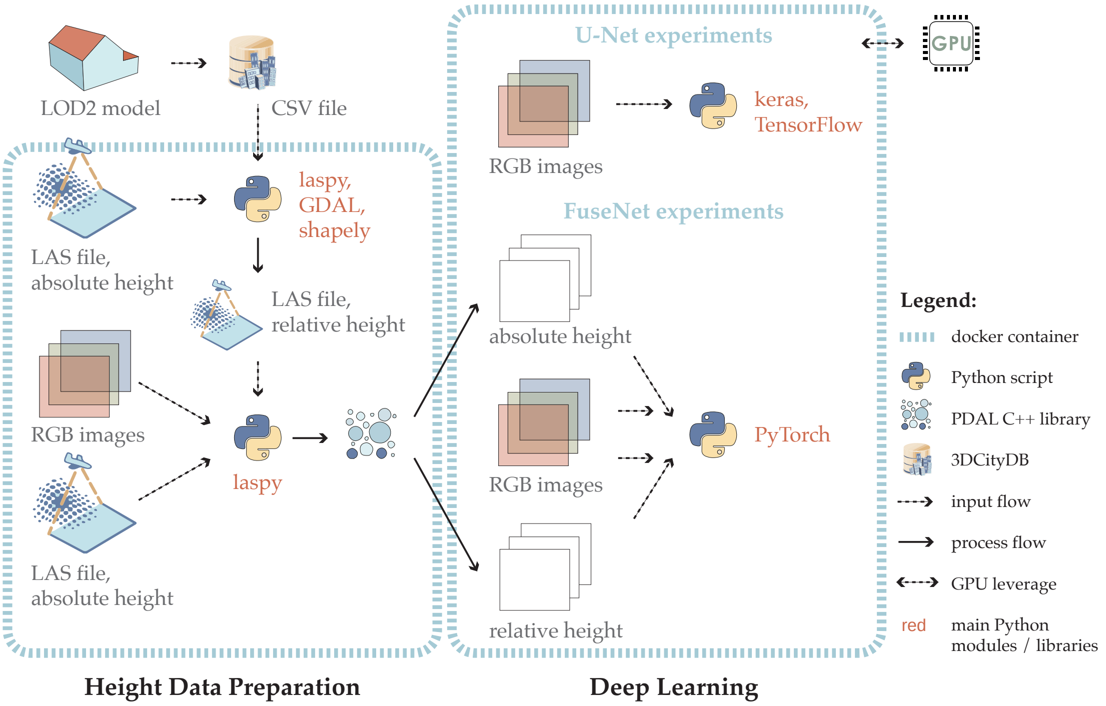


Figure 4.1: Implementation: overview of the pipeline and tools

Google Application Programming Interface (API) images (10cm/pixel), that are not ortho-rectified.

- **LiDAR PC.** According to LDBV 2021b, it has been acquired for Bavaria between 2011 and 2021. The training data area is part of the Taufkirchen zone, which has been acquired in 2012. The resolution is of at least 4 points/m^2. The points are divided into seven classes (last pulse) and eight classes (first pulse) including ground, building, water bodies, objects and bridges (LDBV 2016).

  The first pulse dataset contains points which laser emission has been first reflected, whereas last pulse consists of the last reflections. Therefore, first pulse is mostly suitable to study the canopy of vegetation, whereas last pulse contains more points lying under the canopy, i.e. soil and low buildings' roof. Last pulse is therefore chosen for the current study since containing more building points.

- **Semantic LOD2 model (CityGML).** Existing since 2012 for Bavaria, it has been generated from the building footprints of the cadastral map combined with the PC. The latest model was generated between 2016 and 2021 depending on the location in Bavaria. Buildings are fitted to one of the 13 typical roofshapes described in the ALKIS information system (LDBV 2021a). If height information is missing due to temporal data acquisition mismatches, the roofs are modeled flat with a height dependent on the roof size (more or less than $25\,\mathrm{m}^2$).

4.2.2 Training data preparation

Existing data has been annotated on the city of Wartenberg, Bavaria (figure 4.2), since it offers standard data conditions and would ensure the project is scalable for the whole state. 1880 buildings have been annotated during the last three years, based on Google API image dataset of 2018. In the scope of this thesis, new labels are manually drawn for the same area but based on DOPs.

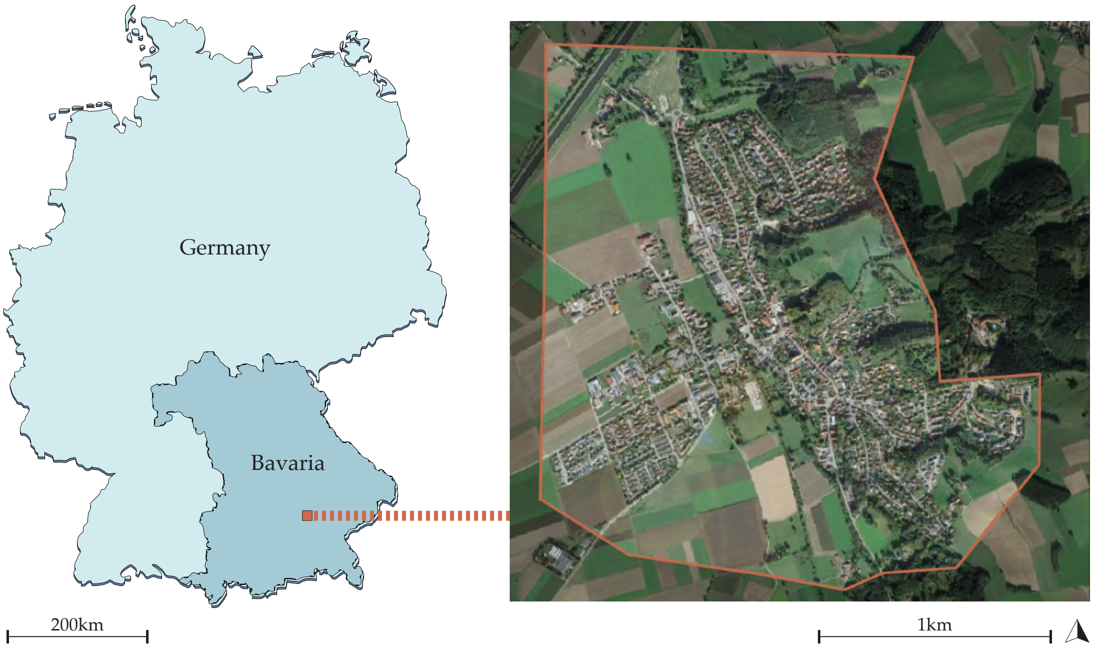


Figure 4.2: Wartenberg location in Germany, state of Bavaria

APs. Image extents are chosen based on a roof-centered approach. Since buildings are studied, it is preferred to generate an image per building, using its center point extracted from Open Street Map (OSM), rather than a regular grid of images. In the latter case, most images would not contain buildings. Roof-centered approach has also the advantage of not cutting buildings on several images but displaying the would roof on the same image. However, it has the disadvantage to provoke overlaps between images. This has to be taken into account when splitting the datasets to avoid data leakage which impact has been studied by Prummer 2021. If similar images are encountered in several of the three datasets—training, validation, test—final accuracies wrongly increase.

Georeferenced images in TIF format are obtained through WMS request. Georeferencement is crucial to correlate height data extracted from the LiDAR datasets. However PNG format is then used for training, since it is a more common encoding. Previous image size per image for the project was of 512×512 pixels. Due to resolution loss provoked by changing from Google APs to DOPs, the image size decreases accordingly, reaching 256×256 pixels. They have three channels: Red, Green, and Blue.

Labels. QGIS software is used to draw vector polygons that are further rasterized with Python using GDAL. The resulting masks have the same size as the APs, and are in PNG format with 7 channels, corresponding to the 6 classes and the background.

Proportions of the new labels are illustrated in figure 4.4. As observed the occurrence of a class (polygon count) is proportionally contrasting with the pixel count for that class. For instance, the number of PV modules represents only 8% of the labeled polygons, however, it represents about half of the labeled pixels. In contrast, chimneys are numerous but represent only few pixels.

The total number of annotated pixels is equal to $256 \times 256 \times 1880$ and the total number of pixels belonging to one of the 6 classes is 29849. Therefore, background class represents 99.98% of the total amount of pixels:

$$\text{Background class} = (256 \times 256 \times 1880 - 29849)/256 \times 256 \times 1880 = 99.98\% \qquad (13)$$

This imbalance represents a major challenge for semantic segmentation and is taken into account both by weighting classes to compute the loss function, and by measuring accuracies while omitting the background class.

Since labeled pixels only add up to 0.02%, the importance of accurate annotation is highlighted. However, the identification of superstructures and their boundary delimitation is not straightforward given the low image resolution and the contrasts making objects not recognisable in shadows. Moreover, the rasterization process into masks increases inaccuracies. Roofs are not aligned with the raster grid, creating aliased superstructure masks, as depicted in figure 4.3.

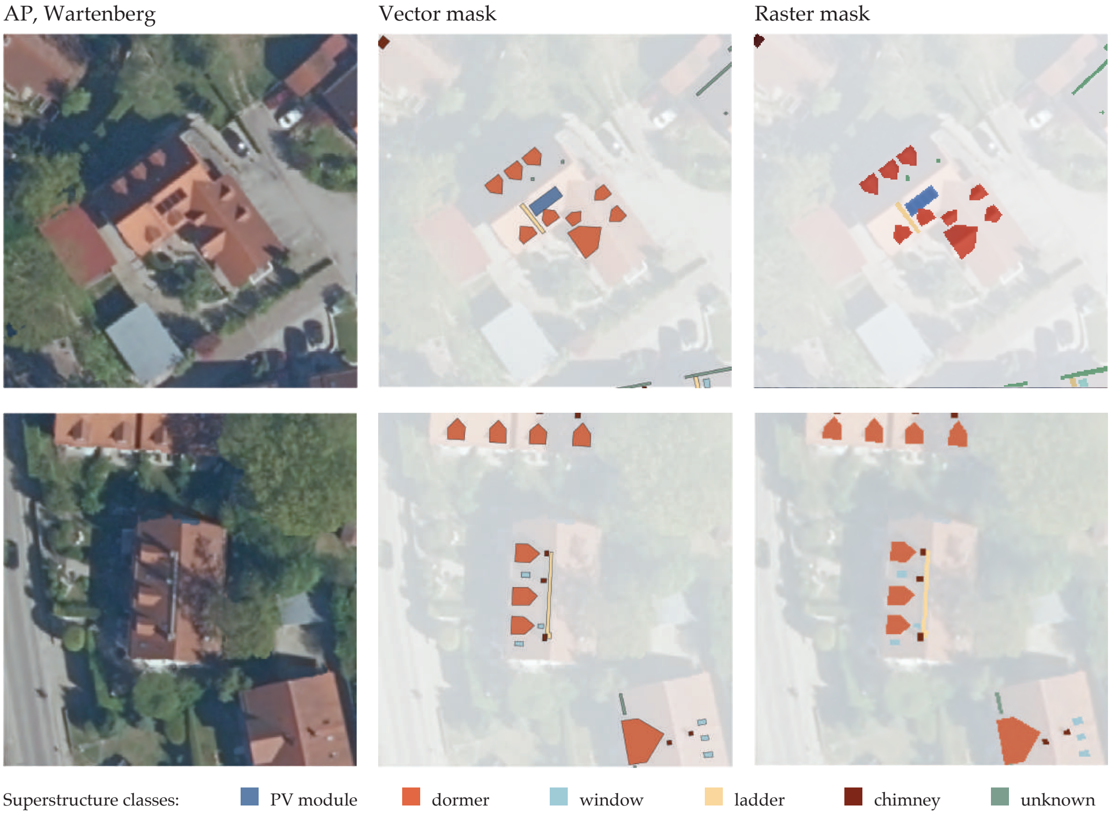


Figure 4.3: Examples of vector labels and raster masks

A way to improve this issue is to implement label uncertainty, as described by Bressan et al. 2021. This factor is applied to the labeled pixel based on their distance to the label border: the smaller is that distance, the less certain is the label. Due to time restriction, this concept has not been carried out.

Content of the classes. Given the wide variety of superstructures' types, labeling is not straightforward. Dormers are the most easily recognisable, and have three main architectural typologies in the dataset: gabled, shed and eyebrow. Some buildings' extension might look similar to dormers because of their gabled-shape. However they are identifiable since not located within the buildings' outline but extending the footprint. Such expansions are not classified.

PV modules and windows are not always differentiable and might be interchanged.

The ladder class includes stairs yielding from roof edges (or from windows) to chimneys, but also horizontal path linking chimneys together (figure 4.5). Ladders are

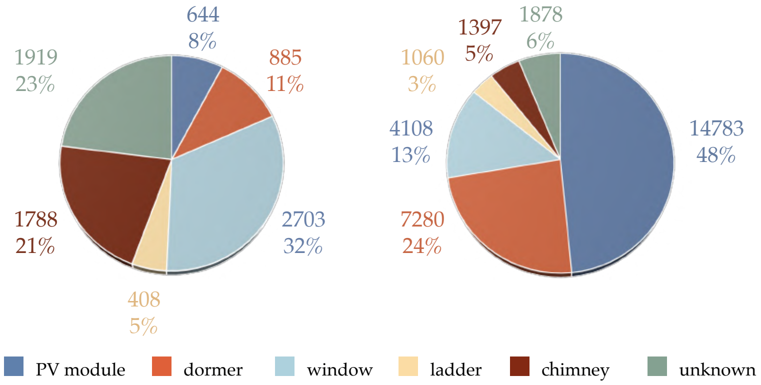


Figure 4.4: Roof superstructures: class occurrence (left), pixel count per class (right)

very common in Bavaria due to a law stipulating their necessity for secure access to chimneys.

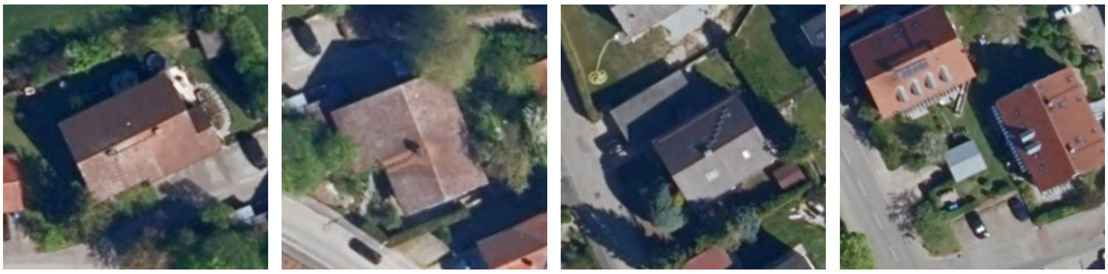


Figure 4.5: Examples of ladders

The unknown class is mainly composed of snow fences, that are discernible because of their linear aspect and distance to the roof edge—unlike gutter which are directly on the edge, and therefore not taken into account. Another element of the unknown are the satellite dishes. Overall, this class is very heterogeneous.

Finally, the chimney class contains also ventilation installations.

4.2.3 Data split for training, validation, test

The training set includes 60% of these annotations, whereas 20% is used for validation and the remaining 20% for model testing. An algorithm avoiding overlaps between datasets is designed, available on github (Apra 2022). It uses a queue data-structure, adding recursively the overlapping images to the same dataset as the image currently considered. This image is removed from the queue whereas overlapping images are added to it, to be examined for overlaps as well. Whenever the queue is empty and the desired amount of images is not yet reached, a random image is added to it.

Given that the labeled dataset is located on a reduced and relatively densely urbanised geographic area, most of the images are overlapping and only few gaps occur. Moreover, since the dataset is already very limited, discarding any content is avoided. Therefore, running the algorithm several times to obtain the desired percentage for the data split always yields similar training dataset, and only few possible configurations for validation and test.

Two distributions, used for the following experiments are illustrated and referred to as “split-01”, and “split-02” in the following chapters.

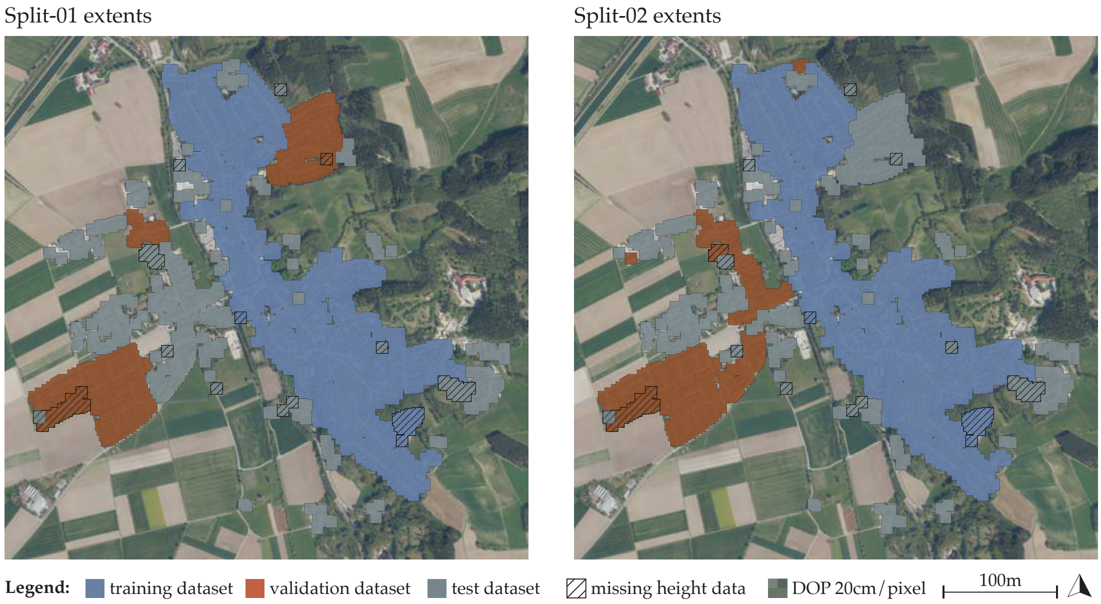


Figure 4.6: Datasets' splits

4.2.4 Dutch datasets

To test the final model on an different geographical area, a location in the Netherlands is elected. The district of Holten within the commune of Rijssen-Holten is chosen (figure 4.7) since it offers comparable rural characteristics to Wartenberg. However, this test set comprises more than 4000 buildings which is ten times more than the Bavarian village previously studied.

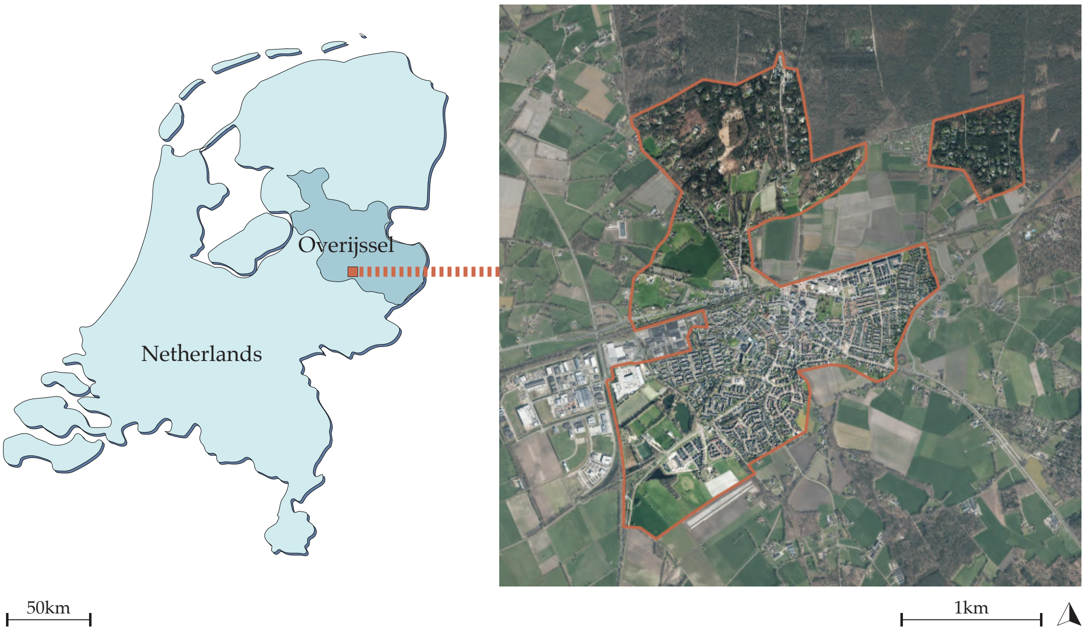


Figure 4.7: Holten location in the Netherlands, province of Overijssel

All necessary datasets are openly available in the Netherlands, including a 3D city model, APs and a LiDAR PC. They are obtained as follows:

- **Semantic LOD2 model.** First, 3D model tiles of the corresponding area are downloaded from 3D BAG registry (tudelft3d 2021). It is converted to GML format and imported to a database from where the coordinates of buildings' centers are extracted.

- **APs.** Then, aerial images centered on those coordinates are obtained through WMS requests from the PDOK server, which offers HR ortho-images (PDOK 2021, layer "Actueel-orthoHR"). They are queried with the necessary image size of 256×256 pixels, with 20cm resolution.
- **LiDAR PC.** Besides, an HR LiDAR dataset is downloaded from AHN 2022. The third AHN version (2014-2019), captured in 2017 for the studied area, is utilized since AHN4 is not yet available for the eastern part of the Netherlands. Once building points are extracted, the absolute height *idw* interpolation dataset is obtained by applying the pipeline aforementioned in 3.3.

Additionally, the final model is trained on all annotation data available, involving a new split of only training and validation data from Wartenberg.

4.3 U-Net experiments

First of all, experiments using the existing U-Net implementation are carried out on DOPs in order to evaluate later the added value of height data fusion. For this, ultimate parameters are determined, based on previous works achieved on non-ortho-rectified images. The best results are chosen as reference for ulterior height-data fusion evaluation. Results are conveyed through matrices, and summed up in tables comprising mean prediction accuracies (mean Pr.Acc), mean accuracies on GT (mean Acc.GT) and IoU scores. Both accuracies take only the 6 classes into account, thus excluding the background to lower the impact of imbalance classes in the evaluation phase.

4.3.1 Scope of U-Net experiments

Based on previous works by Krapf et al. 2021, Prummer 2021 and Bruhse 2022, fundamental settings are applied: Adam, as optimization algorithm, softmax as activation function, a learning rate of 0.001 and batch-size of 8. Other parameters are tested in a reduced range as indicated in table 4.1.

Parameter	Tested values
epoch	**[40:80]**
batch-size	**8**
α	**0.001**
optimizer	**Adam**
activation	**softmax**
loss function	**CFL + Jaccard, CFL + Dice**
backbone	**vgg19, resnet152**

Table 4.1: U-Net parameters to test

4.3.2 Model tuning

Loss function. Statistically-based and geometrically-based functions can be employed to evaluate the model loss (2.2.3). For semantic segmentation tasks, their association is recommended. The implementation of Yakubovskiy 2019 is used for the following experiments in which Dice loss and Jaccard index, respectively combined with CFL, are compared. Additionally, Krapf et al. 2021 suggests that best performance on non-orthorectified data is achieved by combining both Dice and Jaccard index to CFL. Such a combination is also evaluated (table 4.2, id-3).

id	**split**	**epoch**	**loss**	**base**	**mean**	**mean**	**IoU**
					Acc. GT	**Pr.Acc.**	
1	01	40	CFL+Dice	vgg19	**0.47**	**0.53**	**0.45**
2	01	40	CFL+Jaccard	vgg19	**0.43**	**0.58**	**0.45**

3	01	40	0.5(CFL+Dice)
			+0.5(CFL+Jacc.)
				vgg19	**0.43**	**0.61**	**0.44**

Table 4.2: U-Net experiments, parameters and accuracy results: loss function experiments

With rounding to 2 digits, performances are comparable between the diverse loss functions. Since it is computationally more efficient to keep only one geometrical index, *CFL+Jaccard index* is kept for the next experiments.

Backbone. Network trainings are carried out to determine the best backbone between *vgg19* and *resnet152*. For this, standard values for different parameters are used, and a different backbone is tested on both split-01 and split-02. Table 4.3 presents the results. It is determined that *vgg19* produces the best results on both splits and is therefore kept as backbone for the following steps.

id	**split**	**epoch**	**loss**	**base**	**mean Acc.**	**mean**	**IoU**
					GT	**Pr.Acc.**	
2	01	40	CFL+Jaccard	vgg19	**0.43**	**0.58**	**0.45**
4	01	40	CFL+Jaccard	resnet152	**0.40**	**0.53**	**0.43**
5	02	40	CFL+Jaccard	vgg19	**0.44**	**0.63**	**0.46**
6	02	40	CFL+Jaccard	resnet152	**0.42**	**0.62**	**0.45**

Table 4.3: U-Net experiments, parameters and accuracy results: backbone experiments

Epoch. The number of epochs working best is determined by running 80 of them and plotting the evolution of IoU score and loss function (figures 4.9 and 4.8). These curves allow to determine the finest epoch value by identifying the loss minimum and IoU maximum. Both loss and IoU values stabilize around 40 epochs, and no more improvements is observed afterwards. Accordingly, table 4.4 indicates that mean accuracies are not better but similar after 80 epochs than after 40. Therefore, this number is determined as best compromise and kept in the following experiments.

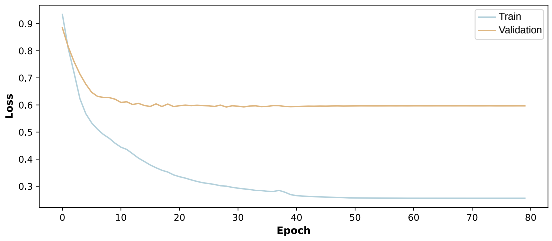


Figure 4.8: Loss evolution through 80 epochs, split-01

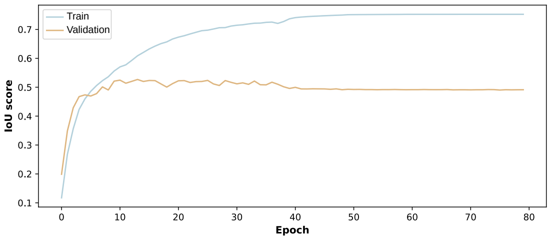


Figure 4.9: IoU score evolution through 80 epochs, split-01

id	split	epoch	loss	base	mean Acc. GT	mean Pr.Acc.	IoU
7	01	80	CFL+Jaccard	vgg19	**0.42**	**0.62**	**0.45**
8	02	80	CFL+Jaccard	vgg19	**0.46**	**0.64**	**0.47**

Table 4.4: U-Net experiments, parameters and accuracy results: experiments on epoch number

Data augmentation. Finally, in order to increase training data and the model's robustness, augmentation data is implemented, in a way that images stay realistic. Transformations involve slight saturation, contrast and luminosity modifications as well as image rotations varying from -30 to +30 degrees. Such transformations are illustrated in figure 4.10. Input dataset is shuffled after each epoch, meaning that augmentation is applied randomly again.

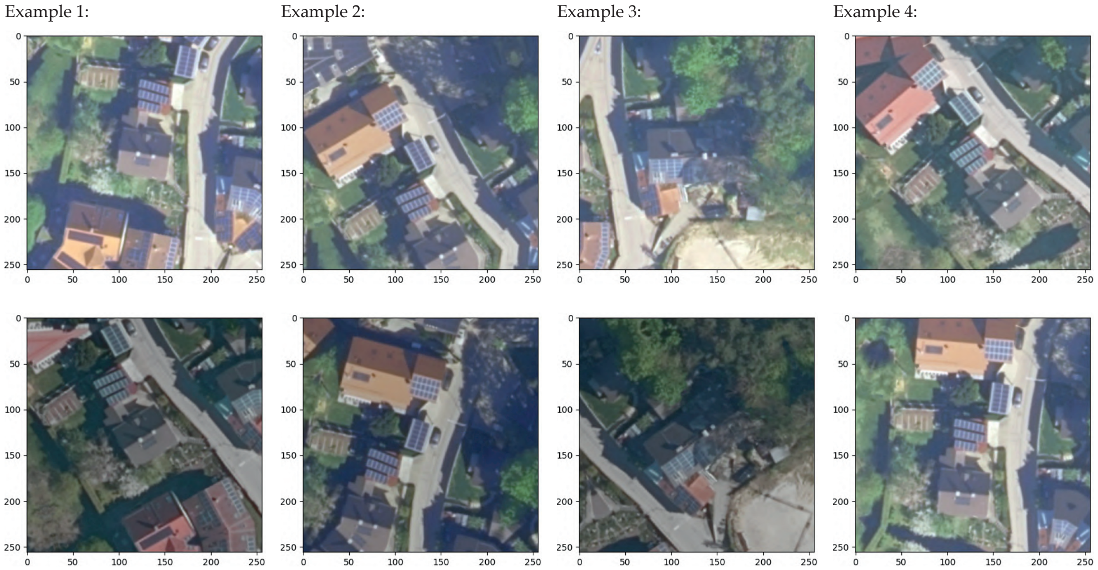


Figure 4.10: Examples of data augmentation

Data augmentation is applied by keeping all previously determined parameters, to determine the impact of increased dataset. First, only the training and validation datasets are transformed, whereas the test dataset is left as original. Then, the tested data is also transformed. As illustrates table 4.5, results are slightly improved regarding the prediction accuracy but not regarding accuracy on GT, meaning that the model identifies less superstructure pixels but also makes less mistakes on the ones that it detects. The dormer class is the one benefiting the most from data augmentation.

Furthermore, improvements are higher when the test dataset is not augmented, because not enough transformed data has been encountered when training. Although 50% of images encountered has been transformed, several modifications happen with a probability of 50% as well. Therefore, only few tenth of images with each type of change has been encountered (e.g. positive saturation change versus negative saturation change, etc.). For further exploration, more epochs and higher probabilities per change should be applied.

Overall, this illustrates the potential of data augmentation, which is especially useful if test datasets have different sources than the ones for training and validation. This is the case when we apply the final model to a test area outside Bavaria. For this, data augmentation is implemented for RGB training data to match the characteristics of Dutch APs.

id	split	epoch	loss	base	augmentation	mean Acc. GT	mean Pr.Acc.	IoU
9	01	40	0.5(CFL+Dice) +0.5(CFL+Jacc.)	resnet152	Train Val.	**0.48**	**0.58**	**0.47**
10	01	40	0.5(CFL+Dice) +0.5(CFL+Jacc.)	resnet152	Train Val. Test	**0.46**	**0.58**	**0.46**

Table 4.5: U-Net experiments, parameters and accuracy results: data augmentation experiments

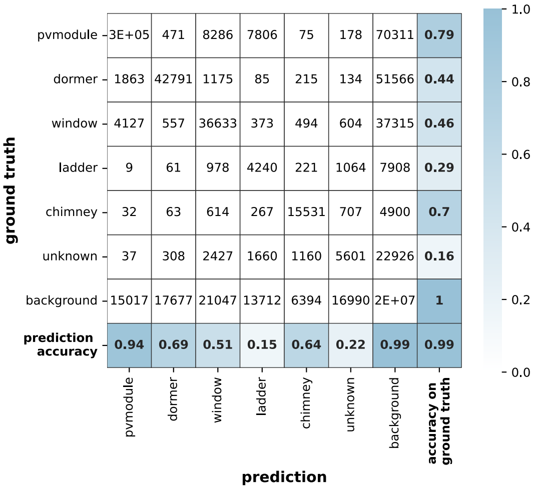
4.3.3 U-Net conclusion

Settings' performance. Prediction accuracies are usually higher than accuracies on GT, meaning that most classified pixels are correctly segmented. However, a large number of pixels expected to be identified as superstructures have been allocated to the background, resulting in low accuracies on GT.

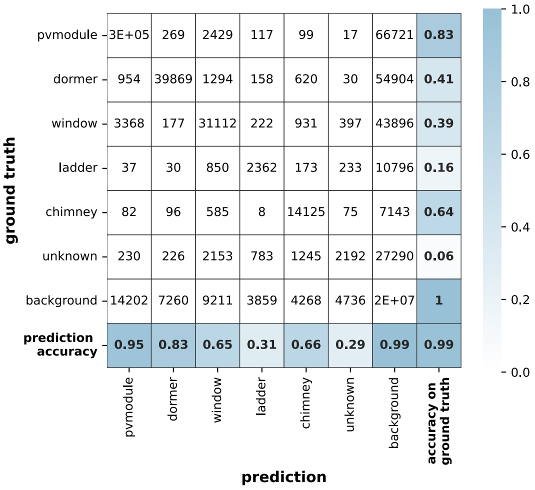
Settings yielding best accuracies on split-01 and split-02 are gathered in table 4.6. The best confusion matrices on the first split, illustrated in 4.11 and 4.12, are kept as reference for further comparison with data fusion results.

id	split	epoch	loss	base	augm.	mean Acc. GT	mean Pr.Acc.	IoU
1	01	40	CFL+Dice	vgg19	None	**0.47**	**0.53**	**0.45**
7	01	80	CFL+Jaccard	vgg19	**0.42**	**0.62**	**0.45**	
5	02	40	CFL+Jaccard	vgg19	None	**0.44**	**0.63**	**0.46**

Table 4.6: U-Net experiments, parameters and accuracy results. *augm.* = data augmentation



	pvmodule	dormer	window	ladder	chimney	unknown	background	accuracy on ground truth
pvmodule	3E+05	471	8286	7806	75	178	70311	**0.79**
dormer	1863	42791	1175	85	215	134	51566	**0.44**
window	4127	557	36633	373	494	604	37315	**0.46**
ladder	9	61	978	4240	221	1064	7908	**0.29**
chimney	32	63	614	267	15531	707	4900	**0.7**
unknown	37	308	2427	1660	1160	5601	22926	**0.16**
background	15017	17677	21047	13712	6394	16990	2E+07	**1**
prediction accuracy	**0.94**	**0.69**	**0.51**	**0.15**	**0.64**	**0.22**	**0.99**	**0.99**



	pvmodule	dormer	window	ladder	chimney	unknown	background	accuracy on ground truth
pvmodule	3E+05	269	2429	117	99	17	66721	**0.83**
dormer	954	39869	1294	158	620	30	54904	**0.41**
window	3368	177	31112	222	931	397	43896	**0.39**
ladder	37	30	850	2362	173	233	10796	**0.16**
chimney	82	96	585	8	14125	75	7143	**0.64**
unknown	230	226	2153	783	1245	2192	27290	**0.06**
background	14202	7260	9211	3859	4268	4736	2E+07	**1**
prediction accuracy	**0.95**	**0.83**	**0.65**	**0.31**	**0.66**	**0.29**	**0.99**	**0.99**

Figure 4.11: Confusion matrix split-01, id-1

Figure 4.12: Confusion matrix split-01, id-7

U-Net performance per split. Results' quality varies between splits. This can be explained because of the different regions covered by validation and test datasets and therefore the different types of buildings encountered in each case. Split-02 obtains better performances than split-01 in most classes, except from PV modules and dormers.

U-Net performance per class. Boxplots obtained from concatenated IoU per class per image (figures 4.13 and 4.14) illustrate the imbalanced results obtained per class as well as the wide range of predictions' quality. In average, best predictions are achieved for PV modules, chimneys and windows, whereas the two worst classes are the unknown and ladder ones. However, for the best detected class, PV modules, half of the results IoU scores are spanning from 0.22 to 0.9 (split-01), resulting in a median value much lower than the mean. On the opposite, mean value for dormer detection is much higher than the median, probably due to a large amount of IoU scores of zeros and few very high scores.

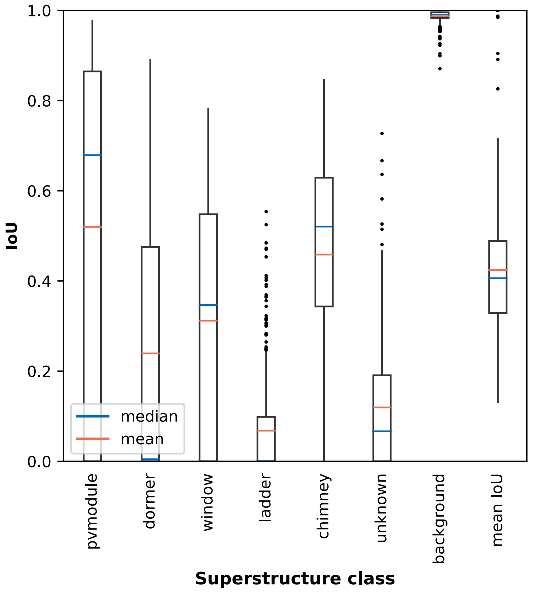


Figure 4.13: Boxplot IoU per class, split-01, id-1

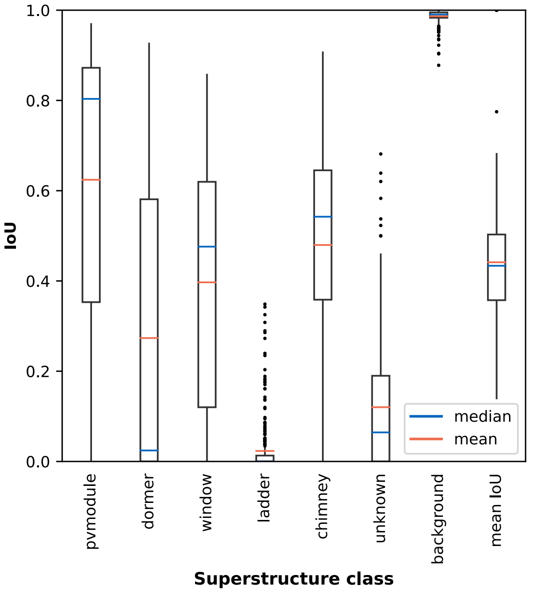


Figure 4.14: Boxplot IoU per class, split-02, id-5

To conclude, height data has potential to improve chimney, dormer and window detection, whereas PV modules are already well identified. Unknown and ladder classes are the most challenging ones to improve.

4.4 FuseNet experiments

4.4.1 Scope of FuseNet experiments

The main experiments are carried out with FuseNet architecture implemented in PyTorch (Hazirbas and Aygun 2018). First, through the choice of one height dataset, ultimate parameters—listed in table 4.7—are narrowed down. Due to time limitation, only one dataset is used to figure out best parameters. *Nn* interpolation dataset is chosen from the absolute height datasets, since it has the lowest IE mean. Hence, it might be the most challenging one to use.

Parameter	Tested values
classes' weights	**td**
epoch	**[0:1000]**
batch-size	**[4, 8, 16]**
α	**[0.001, 0.05, 0.01]**
optimizer	**SGD, Adam**
loss function	**Weighted CE, Dice+CE**
normalization	**Batch normalization**

Table 4.7: FuseNet parameters to test or to determine (*td*)

Only quantitative assessment is used to determine the best settings. Then, different height datasets are consecutively fused using the resulting parameters. Their results are displayed and analyzed (quantitatively and qualitatively) in the next chapter.

4.4.2 Model setup

A deterministic approach is used, so the model's results stay comparable when fed with similar input data. A manual seed of zero is implemented and calculation is carried out with GPU support.

Input data. All input datasets are in uint8 type. Aerial images have three bands for RGB, whereas the height branch has only one channel. Masks are on one band, which values vary from 0 to 6.

Pretraining. For computation efficiency, a pre-trained model is used, initializing weights. Those are taken from PyTorch vision for *vgg16* backbone.

4.4.3 Weight per class determination

Since the classification task is highly imbalanced, weights are applied per class for loss computation. Experiments carried out by Bressan et al. 2021 show good results with a weight calculated as follows:

$$w^c = \frac{n}{C \times n^c} \tag{14}$$

where w^c is the weight for the considered class, n is the total number of pixels in the dataset, C is the number of classes, and n^c is the number of pixels in that class.

Values are then normalized on the sum of all dataset pixels, so the sum of all 7 weights is equal to one. Resulting weights are shown in table 4.8.

Class	Weight
PV module	**0.025**
dormer	**0.053**
window	**0.094**
ladder	**0.38**
chimney	**0.28**
unknown	**0.18**
background	[**0.000003, 0.01, 0.02, 0.03**]

Table 4.8: Weights per class applied to FuseNet loss calsulation. Several background weights are tested.

Applying no weights results in all pixels predicted as background. Therefore background class should have almost no impact but rather all other classes, in inverted order of their pixel occurrence. However, a weight of 0.000003 (resulting from equation 14) applied to the background class yields low prediction accuracies. Since loss calculation almost does not take background into account, prediction on this class does not improve. Superstructures are detected on much wider areas than they actually are, explaining the low accuracy value.

id	split	bgrd weight	epoch	α	batch- size	optimizer	loss	mean Acc.GT	mean Pr.Acc.	IoU
1a	01	.000003	100	0.01 dc25	8	SGD	CE	**0.65**	**0.09**	**0.17**
4b	01	0.02	100	0.01 dc25	4	SGD	CE	**0.47**	**0.51**	**0.44**
5a	01	0.03	100	0.01 dc25	4	SGD	CE	**0.43**	**0.53**	**0.42**
6	01	0.01	100	0.01 dc25	4	SGD	CE	**0.44**	**0.52**	**0.43**

Table 4.9: Experiments on FuseNet using absolute *nn* interpolation dataset: background weight experiments. *bgrd* = background, *dc* = decay

Therefore, it is decided to higher this value so it has still the lowest impact on loss calculation but enough weight to be improved through epochs. Results are compared in table 4.9 after 100 epochs, when assigning a weight of 0.01, 0.02 and 0.03 for the background class. The middle value, yielding the best results, is kept for following experiments.

4.4.4 Model tuning

Epoch. First, 1000 epochs are run with standard parameters to determine the lowest loss value and highest IoU. This time consumption is tolerable because of the low image resolution (256 × 256) and the reduced training dataset size.

Figure 4.15 illustrates that the loss function decreases until 400 epochs on the training dataset before reaching a plateau. On the other hand, IoU continuously increases until 600 epochs on the training dataset but reaches the best scores on the test dataset at around 200 epochs, which illustrates a data overfit. Learning rate is explored to fix this issue, whereas epoch number is identified as optimal around 150 to 200 epochs.

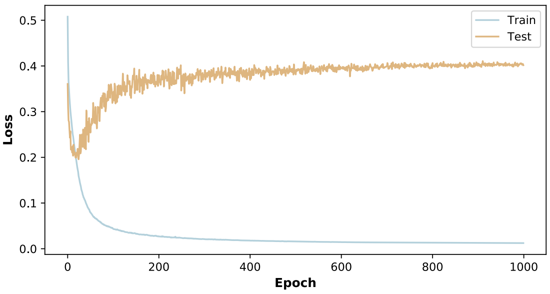


Figure 4.15: Loss evolution through 1000 epochs, split-01

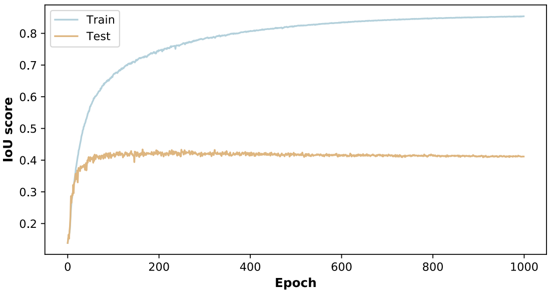


Figure 4.16: IoU score evolution through 1000 epochs, split-01

Batch size. Batch sizes 4, 8, and 16 are tested. Table 4.10 indicates that smaller size yields the best results. A batch of 4 is therefore kept for the following experiments. Illustration 4.17 shows the evolution of IoU score until epoch 50, using similar parameters but 3 different batch sizes.

id	split	bgrd weight	epoch	α	batch- size	optimizer	loss	mean Acc.GT	mean Pr.Acc.	IoU
2	01	0.02	50	0.01 dc25	8	SGD	CE	**0.44**	**0.45**	**0.40**
3	01	0.02	50	0.01 dc25	16	SGD	CE	**0.42**	**0.37**	**0.36**
4a	01	0.02	50	0.01 dc25	4	SGD	CE	**0.45**	**0.52**	**0.42**

Table 4.10: Experiments on FuseNet using absolute *nn* interpolation dataset: batch-size experiments

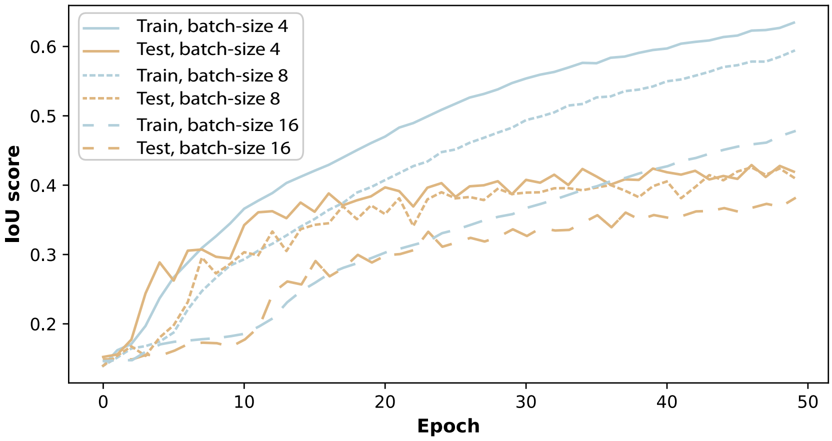


Figure 4.17: IoU score evolution through 50 epochs with different batch-sizes, split-01

Learning rate. Values of 0.01, 0.005 and 0.001 are tested with decays each 10, 20, 25 or 50 epochs. The smallest learning rate value requires too many epochs to start giving meaningful results as demonstrated by experiment 9; it is therefore discarded. On the other hand, after many epochs with a higher rate, data overfitting occurs. Hence, table 4.11 indicates that a value of 0.01 with a decay every 50 epochs (and without decay) constitutes the best compromise.

id	**split**	**bgrd**	**epoch**	α	**batch-**	**optimizer**	**loss**	**mean**	**mean**	**IoU**
		weight			**size**			**Acc.GT**	**Pr.Acc.**	
4c	01	0.02	144	0.01 dc25	4	SGD	CE	**0.48**	**0.54**	**0.45**
7b	01	0.02	111	0.01 dc50	4	SGD	CE	**0.47**	**0.56**	**0.46**
7c	01	0.02	140	0.01 dc50	4	SGD	CE	**0.49**	**0.55**	**0.46**
7d	01	0.02	144	0.01 dc50	4	SGD	CE	**0.50**	**0.53**	**0.46**
13b	01	0.02	135	.01 no-dc	4	SGD	CE	**0.48**	**0.57**	**0.46**
13c	01	0.02	143	.01 no-dc	4	SGD	CE	**0.51**	**0.56**	**0.46**

Table 4.11: Experiments on FuseNet using absolute *nn* interpolation dataset: learning rate experiments

Optimizer. Both Stochatic Gradient Descent (SGD) with momemtum of 0.9 and Adam with a momentum term of 0.5 are tested. The first one produces the best results.

Loss. Combined with an appropriate learning rate, the loss function has the most impact. The statistically-based loss function Cross-Entropy (CE) is applied since it allows to assign weights to classes. A loss computation combining Dice and weighted CE is implemented to include a geometric assessment to the statistical one. However, table 4.12 shows that results are not improved. Therefore, the next experiments keep a mere weighted CE function.

id	**split**	**bgrd**	**epoch**	α	**batch-**	**optimizer**	**loss**	**mean**	**mean**	**IoU**
		weight			**size**			**Acc.GT**	**Pr.Acc.**	
11	01	0.02	300	.005 dc20	4	SGD	Dice+CE	**0.42**	**0.52**	**0.42**
12b	01	0.02	100	0.01 dc20	4	SGD	Dice+CE	**0.49**	**0.32**	**0.35**
12c	01	0.02	150	0.01 dc20	4	SGD	Dice+CE	**0.45**	**0.32**	**0.35**

Table 4.12: Experiments on FuseNet using absolute *nn* interpolation dataset: loss function experiments

Normalization. Batch-normalization is used with a momentum of 0.9.

4.4.5 FuseNet setting conclusion

Setting's performance. As a compromise between the two accuracy types, best performances are reached with experiment 7 nearby 150 epochs, with a learning rate of 0.01. Similar settings are applied to the second split. Experiments 7 and 13, both yielding the best results on split-01, are displayed in table 4.13 with the best performance on split-02.

id	split	bgrd weight	epoch	α	batch-size	optimizer	loss	mean Acc.GT	mean Pr.Acc.	IoU
7c	01	0.02	150	0.01 dc50	4	SGD	CE	**0.47**	**0.60**	**0.45**
13c	01	0.02	143	.01 no-dc	4	SGD	CE	**0.51**	**0.56**	**0.46**
14	02	0.02	130	0.01 dc50	4	SGD	CE	**0.54**	**0.51**	**0.46**

Table 4.13: Experiments on FuseNet: parameters and accuracy results using absolute *nn* interpolation height dataset. *bgrd* = background, *dc* = decay

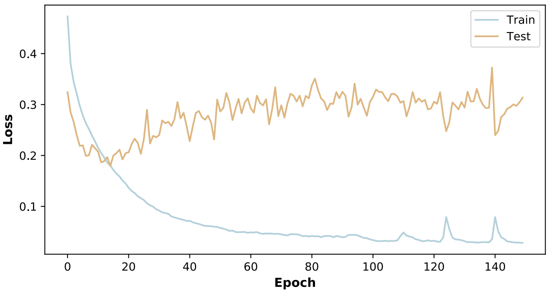


Figure 4.18: Loss evolution through 150 epochs, split-01, id-7c

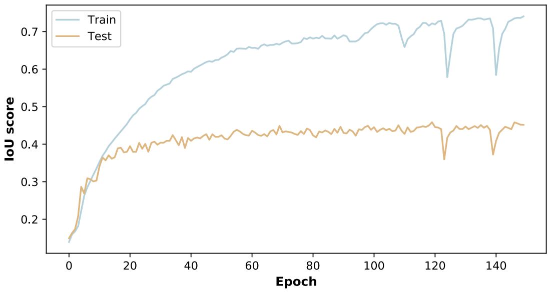


Figure 4.19: IoU score evolution through 150 epochs, split-01, id 7-c

Absolute *nn*, performance per split. Split-02 has better accuracies on GT than split-01, but significantly lower prediction accuracies. Overall, the IoU value is higher for the second split than for the first one. As a result, split-01, more challenging, is kept for the final experiments.

Absolute *nn*, performance per class. PV module, window, and chimney detection improves through time, whereas ladders and unknowns stay confusing for the model. This is highlighted by the next chapter presenting confusion matrices and visualization of predictions. For this purpose, the six different height data inputs are consecutively fused, using previously determined parameters, and applied to the first data split.

5 Results and analysis

The previous chapter determined the best settings for FuseNet implementation. First, it presented the datasets employed. Next, it provided us with results from U-Net implementation on new labels. Then, FuseNet was experimented with the *nn* interpolation dataset, narrowing down the best parameters for performing height data fusion.

This section aims to determine the best height data input for optimal results. First, it comprises predictions for the other height datasets: remaining interpolation types from absolute and relative elevation grids. Next, qualitative and quantitative analyses are carried out. Finally, the best model is applied to another test area from the Netherlands, and predictions are modeled in 3D.

In form, the analysis of FuseNet results consists of a quantitative assessment through confusion matrices and IoU scores, followed by a qualitative one via observation of sample predictions. In substance, the section evaluates FuseNet related to U-Net's performance and the impacts of the different height data incorporation. Consequently, the best height data type for fusion is determined. Then, the network is trained with the corresponding dataset, using optimal settings on all the labeled data. The subsequent model is applied to a test area in the Netherlands. Predictions obtained are modeled in 3D, and a use-case of the resulting semantic model is demonstrated.

5.1 FuseNet Results

5.1.1 Absolute height data fusion

Performance scores are obtained for each absolute height dataset input, using similar settings but assessed at the most appropriate epoch number. Table 5.1 gathers the results.

id	height	epoch	α	batch-	optimizer	mean Acc.	mean	IoU
	data			size		GT	Pr.Acc.	
i	*no*	160	0.01 dc50	4	SGD	**0.40**	**0.51**	**0.40**
ii	*nn*	143	.01 no-dc	4	SGD	**0.51**	**0.56**	**0.46**
iii	*idw*	164	0.01 dc50	4	SGD	**0.49**	**0.56**	**0.46**

Table 5.1: FuseNet parameters and accuracy results using absolute height datasets.

Based on all accuracy criteria and IoU score, *nn* interpolation produces the best fusion results with absolute height datasets.

5.1.2 Relative height data fusion

Similarly, relative height maps are fused to the network. Again, mean prediction accuracies are higher than those of accuracies on GT. The best results are obtained with the *idw* dataset, yielding an IoU score of 0.45.

id	height	epoch	α	batch-	optimizer	mean Acc.	mean	IoU
	data			size		GT	Pr.Acc.	
iv	*no*	225	0.01 dc50	4	SGD	**0.27**	**0.54**	**0.34**
v	*nn*	168	0.01 dc50	4	SGD	**0.51**	**0.51**	**0.45**
vi	*idw*	155	0.01 dc50	4	SGD	**0.46**	**0.57**	**0.45**

Table 5.2: FuseNet parameters and accuracy results using relative height datasets.

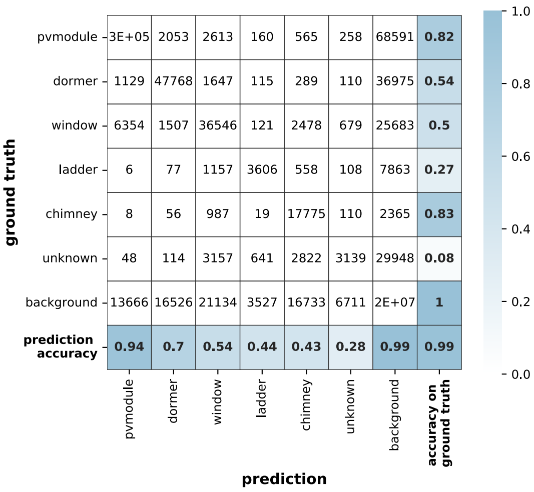


Figure 5.1: *Nn* fusion's best results, absolute height

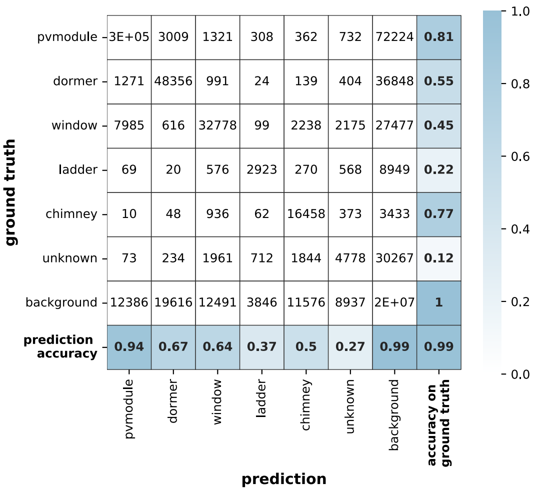


Figure 5.2: *Idw* fusion's best results, absolute height

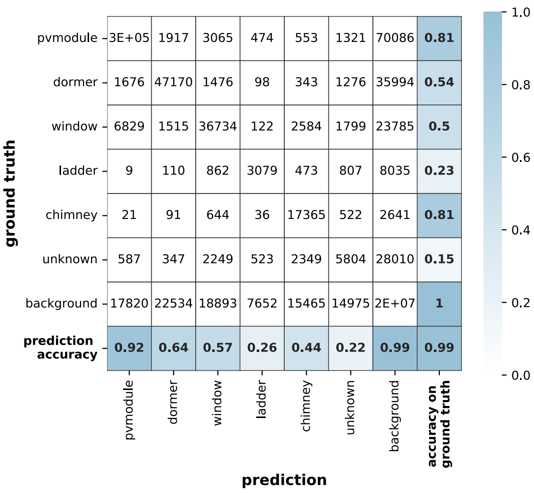


Figure 5.3: *Nn* fusion's best results, relative height

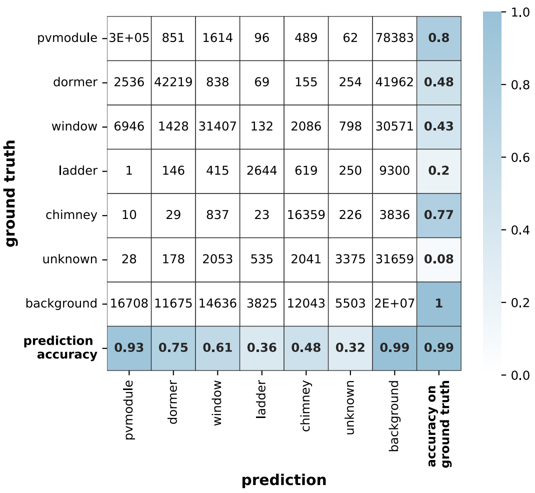


Figure 5.4: *Idw* fusion's best results, relative height

5.2 FuseNet Analysis

First, this section analyzes the input dataset's influence, including data resolution and manual labeling. Then, the network's results are evaluated quantitatively, based on confusion matrices and IoU scores. And qualitatively through observation of predicted masks versus U-Net's output.

First of all, results from FuseNet experiments (absolute and relative) are compared to the U-Net reference confusion matrices by subtracting their corresponding values, allowing an assessment of the fusion performance per class (cf. figure 5.5).

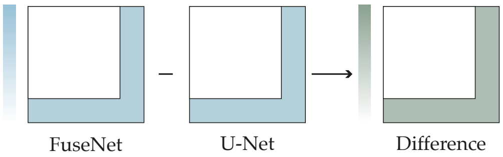


Figure 5.5: Subtraction concept to compare U-Net and FuseNet confusion matrices

Secondly, visualization of prediction masks allows identifying the source of scores' improvements or decreases. For both evaluation processes, the fusion network and U-Net are compared to one another, followed by a comparison of results obtained through the different height datasets' incorporation.

5.2.1 Impact of resolution and labels

Resolution of input datasets. The low resolution of the images impacts the quantitative assessment of the predictions. Some superstructure areas are small, and their boundary shifted from a few pixels decreases their accuracy by tenths percent. It principally affects compact superstructure types such as chimneys. In these cases, the network seems to predict correctly, whereas quantitative assessments based on a pixel approach are uncompromising. Although both ortho-rectified, the alignment of height data derived from LiDAR and images might vary from a few pixels, making boundary shifts more probable.

Labeling impact. Visualizing images with low IoU scores shows labeling mistakes, thereby highlighting the subjectivity of such a process. First of all, mislabeling might occur because of the building-centered approach: the annotation was accomplished building per building and not image per image. Therefore some constructions located on the borders of images occasionally lack labels, as illustrated in figure 5.6 left. Additionally, the definition of a class might not be straightforward. For instance, "horizontal ladders" could rather be interpreted as unknown.

In both cases, the network accomplished its task better than what was achieved through annotation since it predicted some windows and chimneys from a building on the image corner and classified the horizontal path as ladder (cf. figure 5.6).

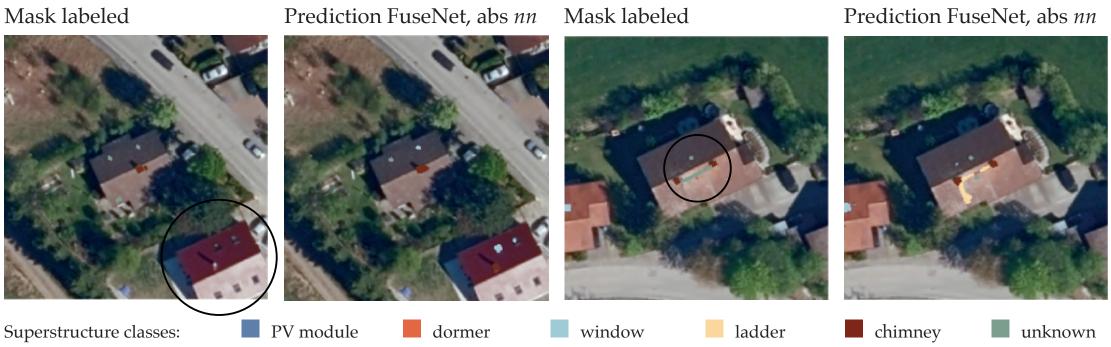


Figure 5.6: Examples of labeling mistakes (black circles), better predicted by the network than annotated

The dormer class offers a similar example since its definition is not straightforward based on APs' observation. In figure 5.7 bottom, the network predicts some dormers that are not annotated on the GT. However, it is hardly discernible to an observer how they should be classified. Since this superstructure class is wide, numerous pixels are penalized.

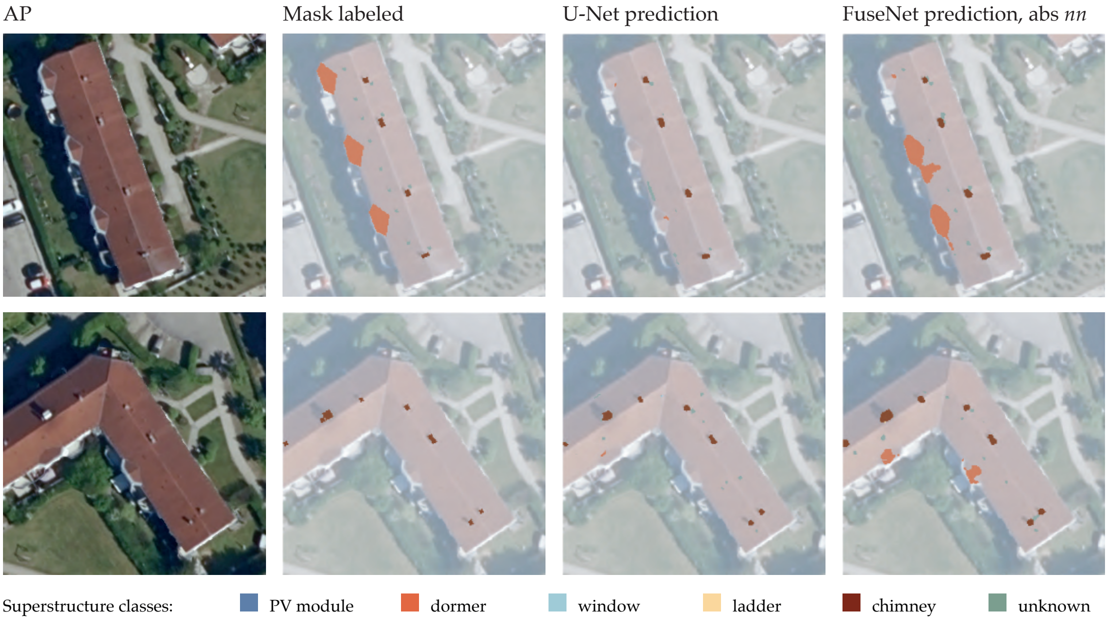
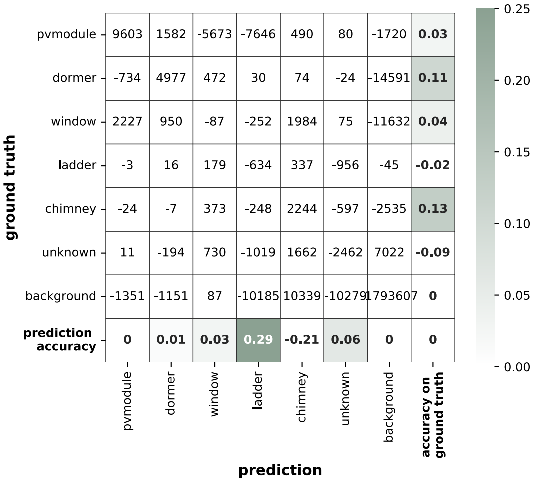


Figure 5.7: U-Net versus FuseNet predictions: dormer and chimney classes

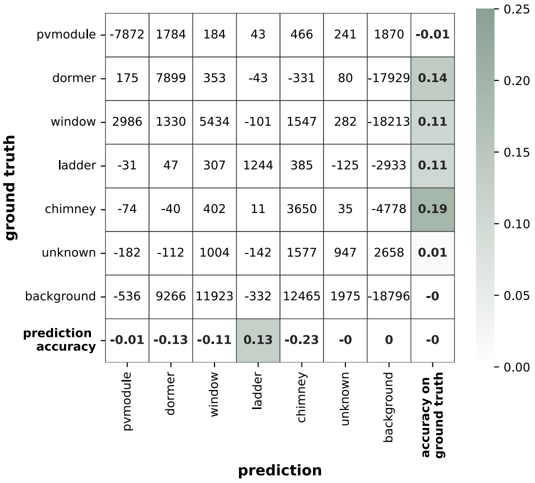
5.2.2 Quantitative analysis

The IoU score, considering all classes with the background, provides a synthetic tool to assess the network's performance. However, to conduct an in-depth analysis, the whole confusion matrices provide more nuances thanks to their accuracy criteria focusing on superstructure evaluation class per class.

Height data fusion versus U-Net. The *nn* dataset achieves the finest scores and is compared to the best U-Net scores by subtracting U-Net's confusion matrices from it. The resulting matrices highlight the classes with score improvement through the green color scale (figures 5.8 and 5.9).



	pvmodule	dormer	window	ladder	chimney	unknown	background	**accuracy on ground truth**
pvmodule	9603	1582	-5673	-7646	490	80	-1720	**0.03**
dormer	-734	4977	472	30	74	-24	-14591	**0.11**
window	2227	950	-87	-252	1984	75	-11632	**0.04**
ladder	-3	16	179	-634	337	-956	-45	**-0.02**
chimney	-24	-7	373	-248	2244	-597	-2535	**0.13**
unknown	11	-194	730	-1019	1662	-2462	7022	**-0.09**
background	-1351	-1151	87	-10185	10339	-10279	1793607	**0**
prediction accuracy	**0**	**0.01**	**0.03**	**0.29**	**-0.21**	**0.06**	**0**	**0**



	pvmodule	dormer	window	ladder	chimney	unknown	background	**accuracy on ground truth**
pvmodule	-7872	1784	184	43	466	241	1870	**-0.01**
dormer	175	7899	353	-43	-331	80	-17929	**0.14**
window	2986	1330	5434	-101	1547	282	-18213	**0.11**
ladder	-31	47	307	1244	385	-125	-2933	**0.11**
chimney	-74	-40	402	11	3650	35	-4778	**0.19**
unknown	-182	-112	1004	-142	1577	947	2658	**0.01**
background	-536	9266	11923	-332	12465	1975	-18796	**-0**
prediction accuracy	**-0.01**	**-0.13**	**-0.11**	**0.13**	**-0.23**	**-0**	**0**	**-0**

Figure 5.8: Absolute *nn* fusion's best results compared to U-Net's results, id-1

Figure 5.9: Absolute *nn* fusion's best results compared to U-Net's results, id-7

Six FuseNet models scoring 0.45 and higher are gathered in table 5.4 and compared class by class to the two most performing U-Net models with no data augmentation (table 5.3). These U-Net references offer complementary characteristics since one has high prediction accuracies, the other high accuracies on GT.

id	split	epoch	loss	base	augmentation	mean Acc. GT	mean Pr.Acc.	IoU
1	01	40	CFL+Dice	vgg19	None	**0.47**	**0.53**	**0.45**
7	01	80	CFL+Jaccard	vgg19	None	**0.42**	**0.62**	**0.45**

Table 5.3: Most performing U-Net models: parameters and accuracy results

id	split	bgrd weight	epoch	α	batch-size	optimizer	loss	mean Acc.GT	mean Pr.Acc.	IoU
4c	01	0.02	144	0.01 dc25	4	SGD	CE	**0.48**	**0.54**	**0.45**
7b	01	0.02	111	0.01 dc50	4	SGD	CE	**0.47**	**0.56**	**0.46**
7c	01	0.02	140	0.01 dc50	4	SGD	CE	**0.49**	**0.55**	**0.46**
7d	01	0.02	144	0.01 dc50	4	SGD	CE	**0.50**	**0.53**	**0.46**
13b	01	0.02	135	.01 no-dc	4	SGD	CE	**0.48**	**0.57**	**0.46**
13b	01	0.02	143	.01 no-dc	4	SGD	CE	**0.51**	**0.56**	**0.46**

Table 5.4: Best experiments on FuseNet using absolute height, *nn* interpolation dataset

Both U-Net's confusion matrices are subtracted from those of FuseNet. The differences obtained are plotted together in one figure 5.10, enabling the definition of an overall trend. (All the values used for this graph are indicated in appendix 6.4.)

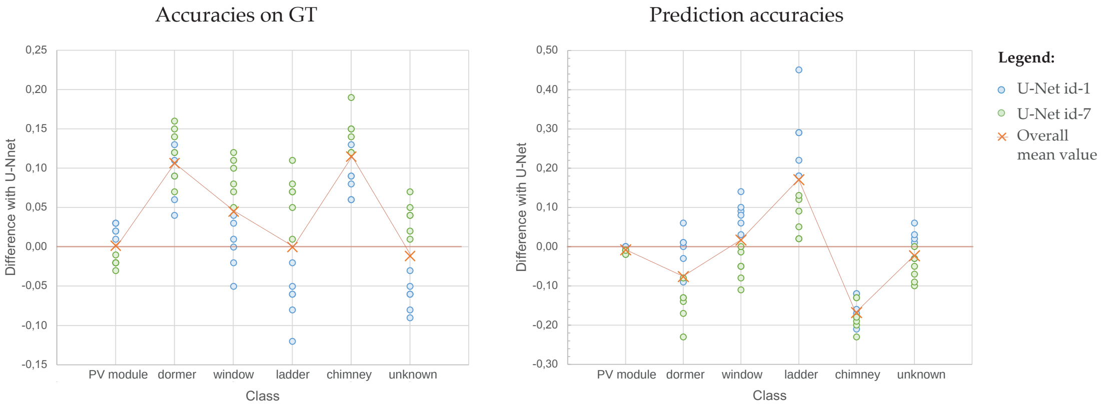


Figure 5.10: Difference between best absolute *nn* scores and U-Net models (id-1 and 7)

The combination of figures 5.8, 5.9 and 5.10 show that GT accuracies are significantly enhanced for dormers and chimneys, with 11% and 12% gain in average. The window class is slightly ameliorated, whereas remaining categories are not improved.

On the other hand, prediction accuracies show no improvement except from ladders, increasing by 17% in average. Dormers and chimneys' prediction accuracy decreases because their surfaces are predicted too large. It is noticed in the confusion matrices 5.8, 5.9, where a lot of background pixels are erroneously classified. It can be related to the mismatches between RGB and height datasets, making the boundaries of small superstructures less clear.

Overall, mean accuracies on GT are significantly improved, whereas prediction accuracies present an ambivalent evolution. In conclusion, superstructure types are better recognized through height data fusion, but not their outlines. The most significant increases concern chimneys and dormers. Additionally, given the broad data span obtained for the ladder class, this category offers less reliable outcomes.

Height type. The plots in figure 5.11 show absolute and relative height data results compared to the same two U-Net results, allowing us to compare both elevation types with one another. (All the values used for this graph are indicated in appendix 6.4.)

According to the mean values observed, both datasets yield comparable results related to U-Net. Only the ladder class presents different outcomes since the network detects less of them through normalized height data. Other categories offer similar results. However, the IoU scores obtained through absolute height data fusion are slightly higher than those obtained from relative height.

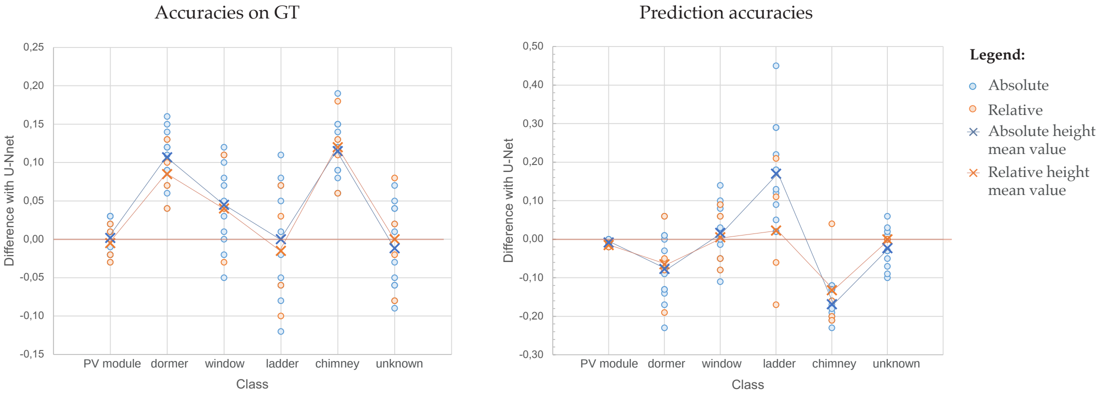


Figure 5.11: Difference between best scores of *absolute* and *relative* height data fusion compared to the U-Net models, id-1 and 7

Interpolation type. Among the three types tested, *no* interpolation dataset yields the worst results. It can be explained by the heterogeneity of values, especially the large amount of no data (zeros), making the structure hardly understandable to the network. This outcome is coherent with IE criteria since this dataset had the lowest score.

On the other hand, *nn* and *idw* yield comparable results. As a result, a high IE score does not seem to be correlated to more efficient information for the network since *idw* has greater IE values than *nn*.

5.2.3 Qualitative analysis

For observation-based analysis, prediction masks are visually compared to U-Net's output and to GT data (RGB and height map). This part compares U-Net with FuseNet's predictions and different height data fusion results.

Height data fusion versus U-Net.

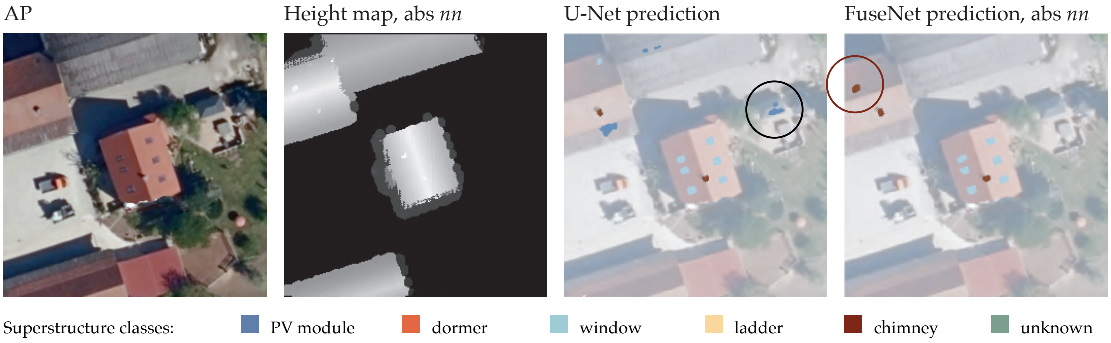


Figure 5.12: U-Net versus FuseNet predictions: example of inaccuracies specific to each network

FuseNet presents advantages over U-Net: no matter the data type, height input helps the network identify the buildings' boundaries and localize superstructures only on them. It acts thereby as construction footprint information. On the opposite, U-Net sometimes identifies superstructures on the ground (cf. figure 5.12, black circle).

Moreover, volumetric classes (chimneys and dormers) are generally better recognized by FuseNet. Figure 5.13 illustrates that U-Net does not identify some dormers at all.

FuseNet also presents disadvantages: figure 5.12 (red circles) shows that it sometimes detects roof installations that are not visible on the images, probably due to outlier points from the LiDAR dataset.

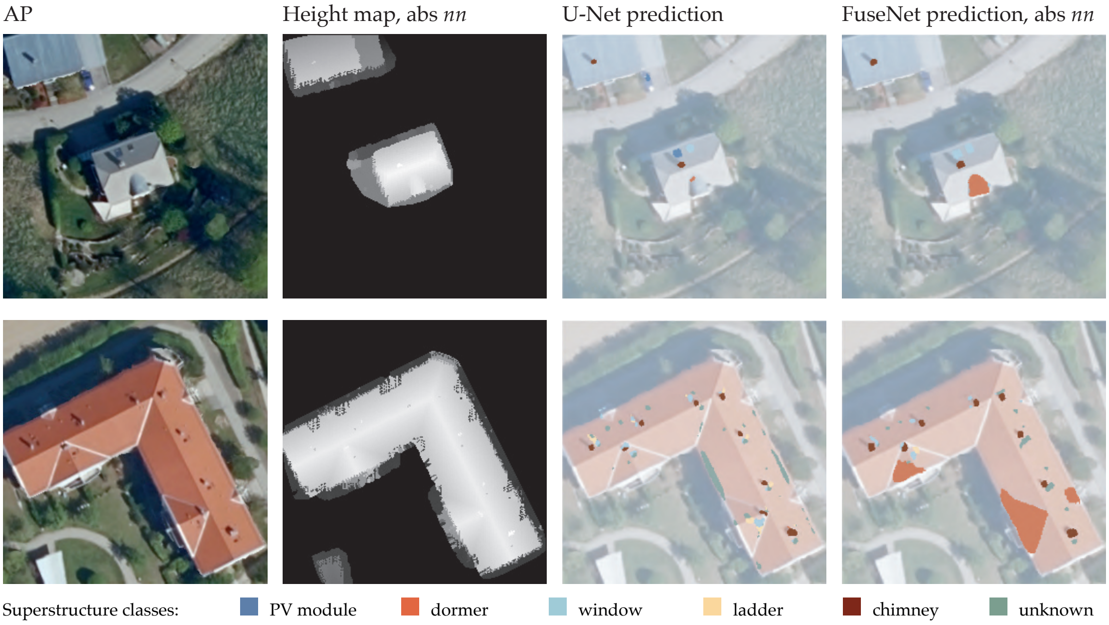


Figure 5.13: U-Net versus FuseNet predictions: example of volumetric classes better detected by FuseNet

Another weakness of FuseNet relates to height information lack.

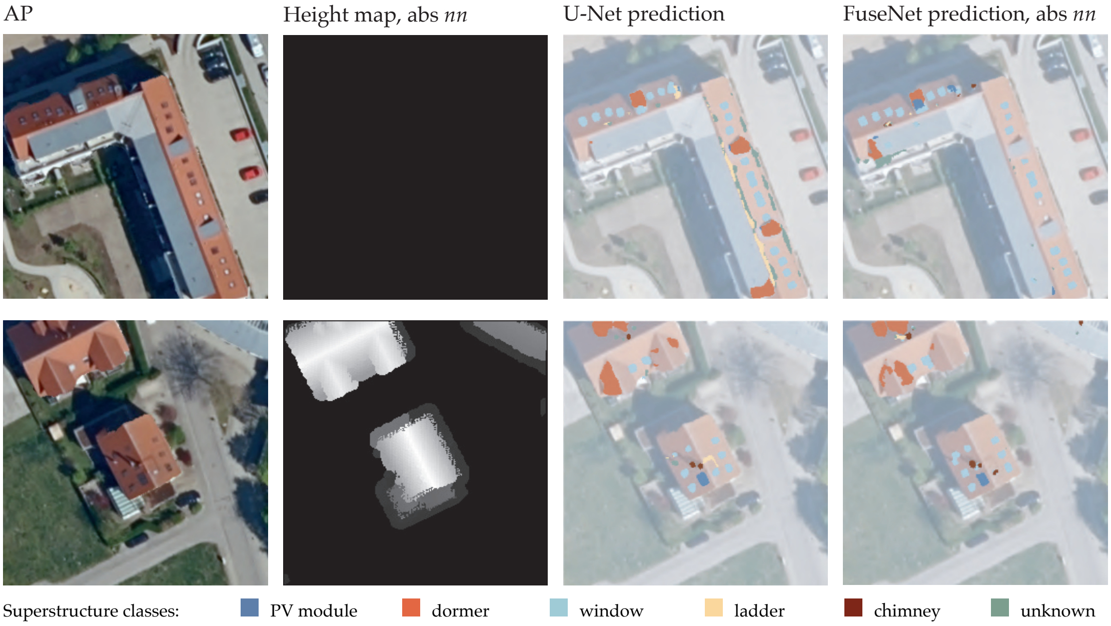


Figure 5.14: U-Net versus FuseNet predictions: examples of height data missing

Figure 5.14 shows first an image where height data is absent. In this case, U-Net detects superstructures better, especially dormers. On the second example, the central building misses height data its chimneys. In this situation, although they are detected by both networks, U-Net performs better regarding their boundaries and the ladder class.

Finally, FuseNet tends to cluster more background pixels to the predictions. It explains the decrease in prediction accuracies since the matrices indicate high values in the background entries. Qualitatively, it is observable through the superstructures' boundaries. Especially the chimneys, which areas are superior through FuseNet segmentation than U-Net's one. This is observable in figure 5.13.

Height type. Absolute data indirectly instruct slope knowledge. It is not the case of relative elevations. Therefore in the first case, the network can learn from more structural information. As a result, figure 5.15 illustrates that detection from absolute height is more accurate for ladders and dormers. The yellow circle shows an incorrect ladder identified using relative height. Absolute height is favorable for ladders recognition because this class is usually perpendicular or parallel to the slope. And missing dormers detection can be caused by the 3D model used for normalization that includes some of them already.

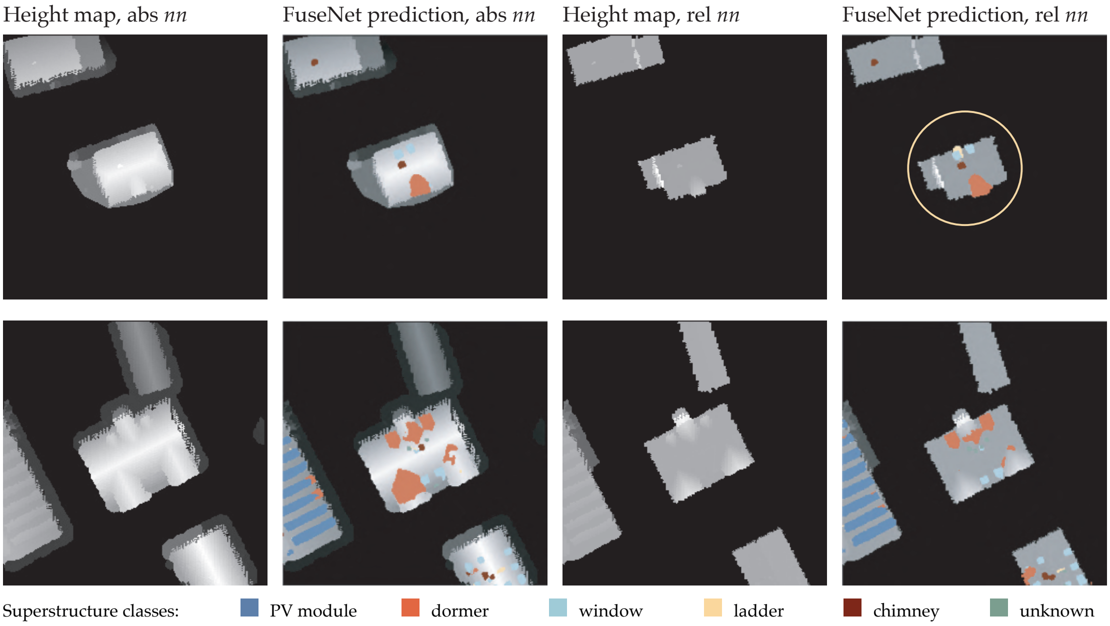


Figure 5.15: Absolute versus relative height predictions: better detection of ladders (yellow circle) and dormers based on absolute height data input

Interpolation type. *No* interpolation makes the network understand local reliefs. However, it is not able to understand the scale of the superstructure and therefore misclassifies these reliefs. Figure 5.16 shows small areas wrongly segmented as dormers and chimneys (black circles).

Nn depicts more clearly superstructure boundaries than *idw* interpolation. However, their surfaces are more limited, making this interpolation technique more rigid. However, since FuseNet has an asymmetric architecture prioritizing the RGB branch over the height one, the difference between the two interpolation techniques cannot be generalized. Another architecture conferring more weight to the elevation data should be used to determine which one is the best.

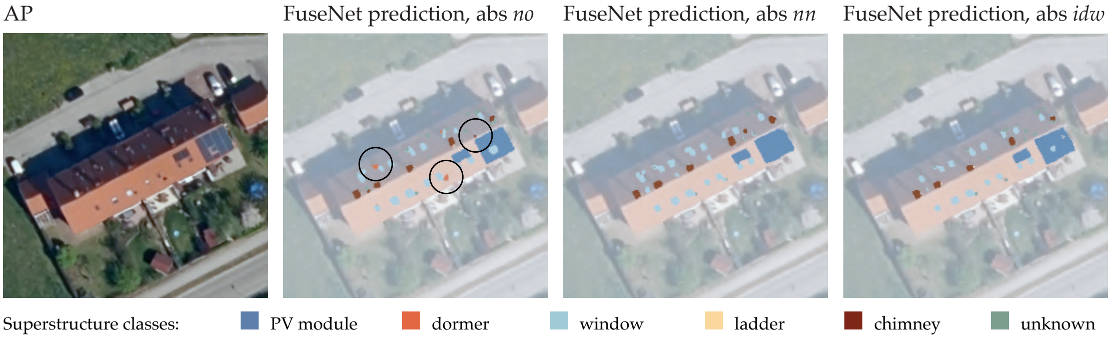

AP
FuseNet prediction, abs *no*
FuseNet prediction, abs *nn*
FuseNet prediction, abs *idw*
Superstructure classes:
PV module
dormer
window
ladder
chimney
unknown

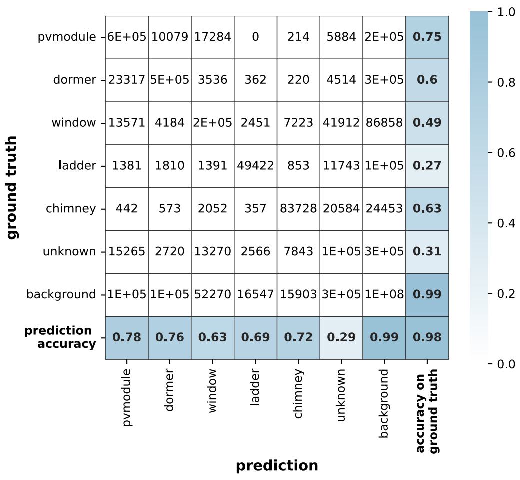
Figure 5.16: Comparison of results from the different interpolation techniques

Conclusion. To conclude, chimneys and dormers are better recognized by FuseNet than U-Net. Nevertheless, more surrounding background pixels are clustered with the predictions. Moreover, absolute height is preferred over relative one since it provides more structural context (like slope). Furthermore, interpolated height data is more favorable than *no* interpolation since it gives a better understanding of superstructures' scale and shape.

5.2.4 Comparison to previous results on non-ortho-rectified dataset

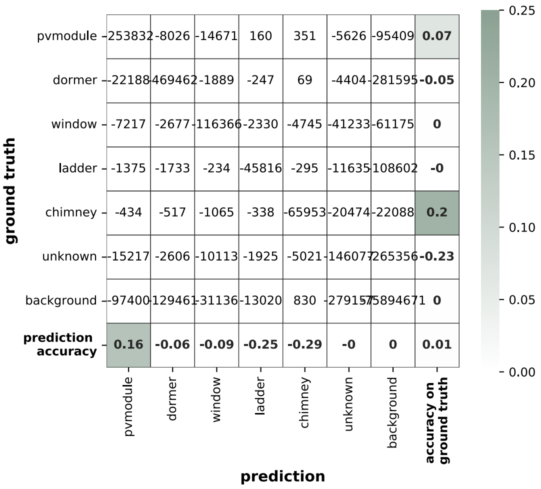
Previous work on the chair were carried out on non-ortho-rectified photos of resolution 512 × 512, resulting in 4 times more pixels classified. Moreover, classes included shadows and overhanging trees, increasing the number of labeled pixels in comparison to the background. The annotations used are different and drawn by other people, thereby adding subjective factors. Additionally, the data split is different from the one employed in the current research. As demonstrated in part 4.3 when comparing split-01 and split-02, this distribution has a considerable impact on the results. Therefore, drawing a comparison is not fully reliable.

Nevertheless, at the time of this publication, the best results are achieved by Bruhse 2022 who obtains an IoU score of 0.45, using a Feature Pyramid Network (FPN) with a loss function combining weighted Jaccard and focal loss. Per class, comparable accuracies are obtained between the two projects, except on the chimneys, as indicated by the matrix 5.18, confirming the added value of height data fusion.



	pvmodule	dormer	window	ladder	chimney	unknown	background	accuracy on ground truth
pvmodule	6E+05	10079	17284	0	214	5884	2E+05	**0.75**
dormer	23317	5E+05	3536	362	220	4514	3E+05	**0.6**
window	13571	4184	2E+05	2451	7223	41912	86858	**0.49**
ladder	1381	1810	1391	49422	853	11743	1E+05	**0.27**
chimney	442	573	2052	357	83728	20584	24453	**0.63**
unknown	15265	2720	13270	2566	7843	1E+05	3E+05	**0.31**
background	1E+05	1E+05	52270	16547	15903	3E+05	1E+08	**0.99**
prediction accuracy	**0.78**	**0.76**	**0.63**	**0.69**	**0.72**	**0.29**	**0.99**	**0.98**

Figure 5.17: Confusion matrix from Bruhse 2022 showing only classes of interest



	pvmodule	dormer	window	ladder	chimney	unknown	background	accuracy on ground truth
pvmodule	-253832	-8026	-14671	160	351	-5626	-95409	**0.07**
dormer	-22188	469462	-1889	-247	69	-4404	-281595	**-0.05**
window	-7217	-2677	-116366	-2330	-4745	-41233	-61175	**0**
ladder	-1375	-1733	-234	-45816	-295	-11635	-108602	**-0**
chimney	-434	-517	-1065	-338	-65953	-20474	-22088	**0.2**
unknown	-15217	-2606	-10113	-1925	-5021	-146077	-265356	**-0.23**
background	-97400	-129461	-31136	-13020	830	-279157	-5894671	**0**
prediction accuracy	**0.16**	**-0.06**	**-0.09**	**-0.25**	**-0.29**	**-0**	**0**	**0.01**

Figure 5.18: FPN from Bruhse 2022 - FuseNet (id-13)

5.3 Dutch test area

The fusion models yielding best performance employ absolute height data with *nn* or *idw* interpolation. The latter is chosen with parameters of experiment iii. It is trained on the whole labeled data to be applied to buildings from another geographic area chosen from the Netherlands: Holten. Sample images of the city are visible in figure 5.19.

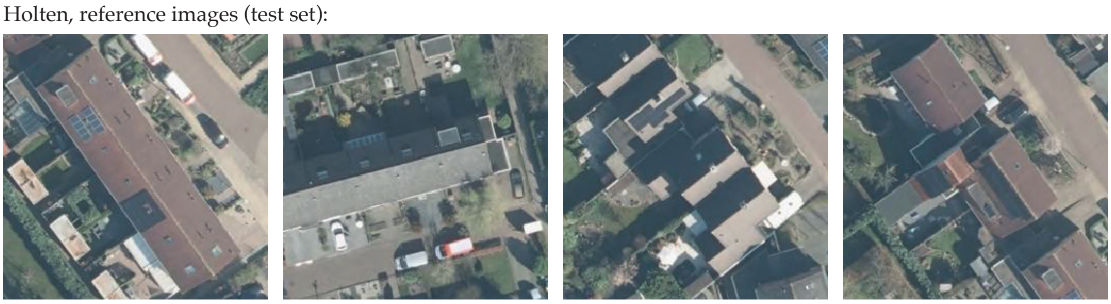


Figure 5.19: Sample images of Holten test dataset

5.3.1 Data augmentation

RGB images. Wartenberg images appear more contrasted and saturated but darker than Dutch images. To counterbalance this phenomenon, transformation factors are applied to the whole dataset: saturation (0.52), contrast (0.6), brightness (1.4), and sharpness (2). Figure 5.20 illustrates such modifications. Although the resulting tones look comparable to the new test set, Holten's images remain sharper. Favorably, the shadows' orientation is almost similar between the two datasets.

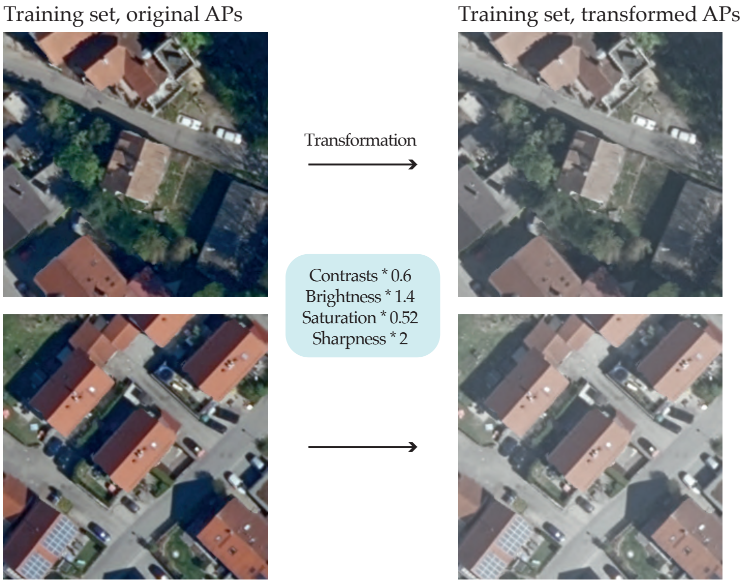


Figure 5.20: Transformation of training data to match the appearance of the test dataset

Height maps. Figure 5.21 considers the correlation between elevation data and RGB images, highlighting mismatches on building boundaries (black circles). These examples illustrate that the correspondence between height and images is superior for the Bavarian training dataset than for Holten. Therefore, augmentation with shifts could have improved the model performance on Holten. But due to time constraints, it has not been implemented.

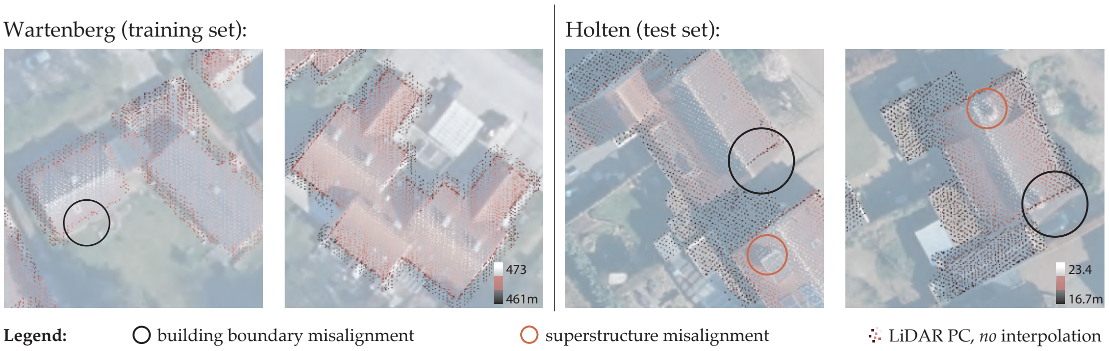

Wartenberg (training set):
Holten (test set):
473
461m
23.4
16.7m
Legend:
○ building boundary misalignment
○ superstructure misalignment
LiDAR PC, *no* interpolation

Figure 5.21: Comparison of Wartenberg and Holten input images related to their corresponding PC

5.3.2 Qualitative analysis

Since no labels are available for this region, only a qualitative analysis is carried out.

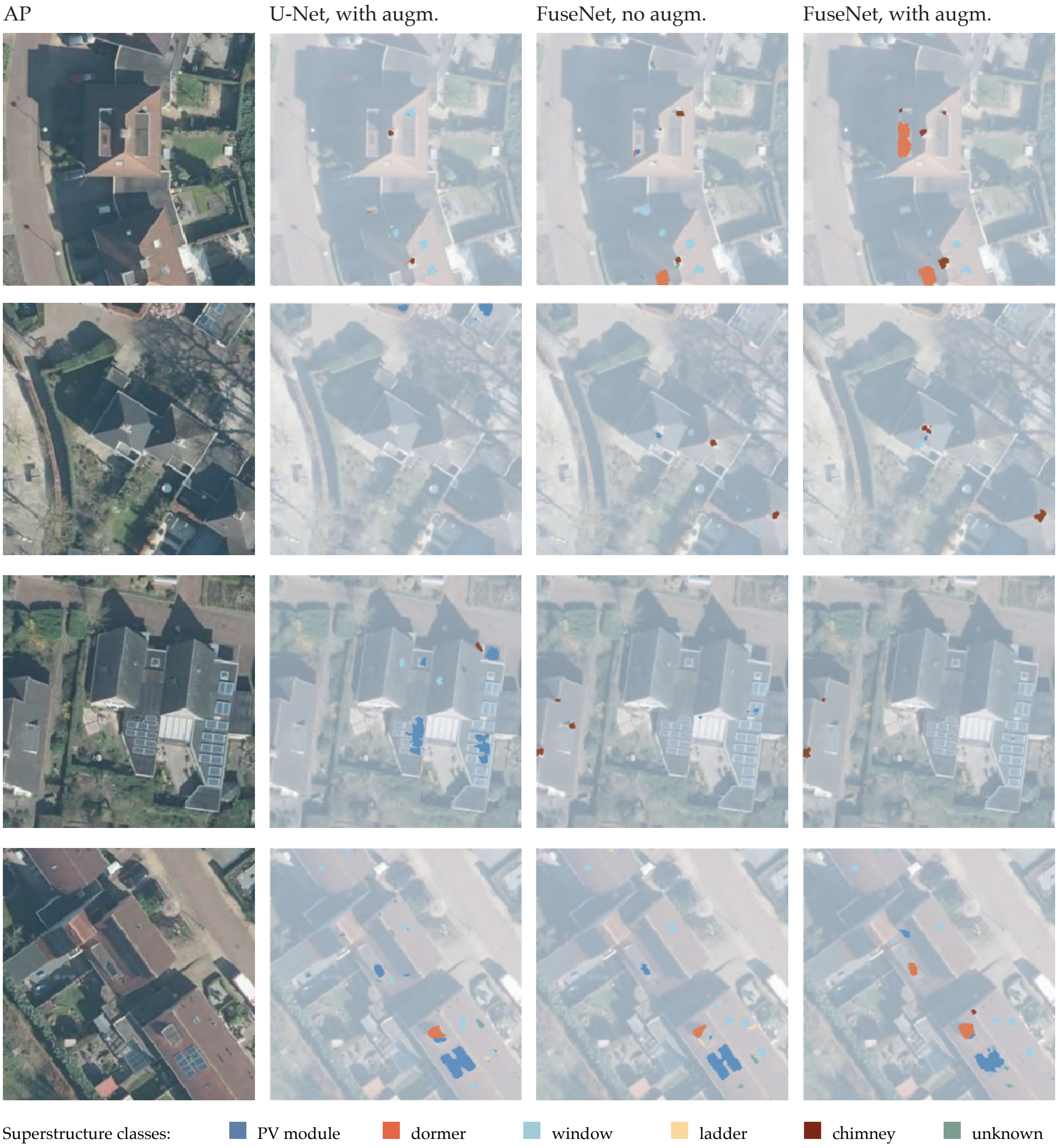


Figure 5.22: Comparison between U-Net and FuseNet predictions for Holten. *augm.* = augmentation

Segmentation results are compared to U-Net predictions on the same test set. Then, an analysis per class regarding FuseNet outcomes is carried out. No ladders or snow fences (included in the unknown class for Wartenberg) are visible in the area considered. Finally, the 3D model is generated and related to the 3D-BAG available at LOD2.2 for this location.

U-Net versus FuseNet. chimneys are better recognized by FuseNet, whereas dormers present only slight improvements (figure 5.22, two first examples). On the other hand, windows amd PV modules seem better recognized by U-Net (figure 5.22, third and fourth example). Ladders are absent from the samples observed, and unknown is very limited for both models.

FuseNet performance per class. The height data information appears qualitative, resulting in a fine detection of chimneys (figure 5.23).

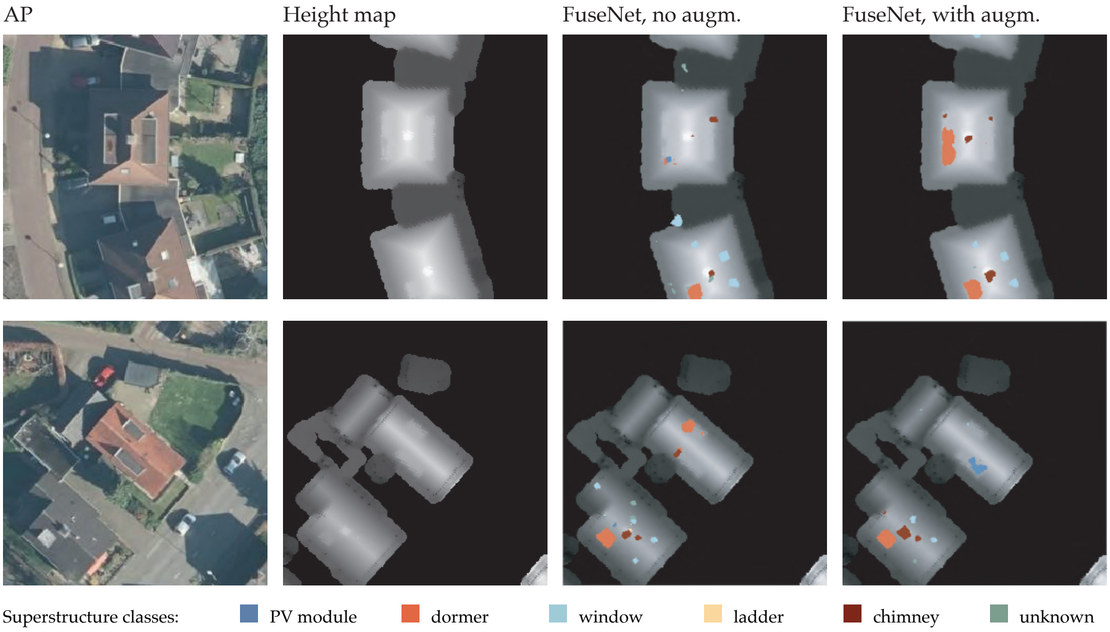


Figure 5.23: Examples of FuseNet dormer predictions for Holten, with and without data augmentation

However, dormers are hardly recognized. It is understandable through the different architectural typologies between the Bavarian training data and the Dutch test set. Most of Wartenberg dormers are gabled, whereas Holten's ones are flat boxes (figure 5.24). Moreover, their scale and orientation according to the roof segment are different.

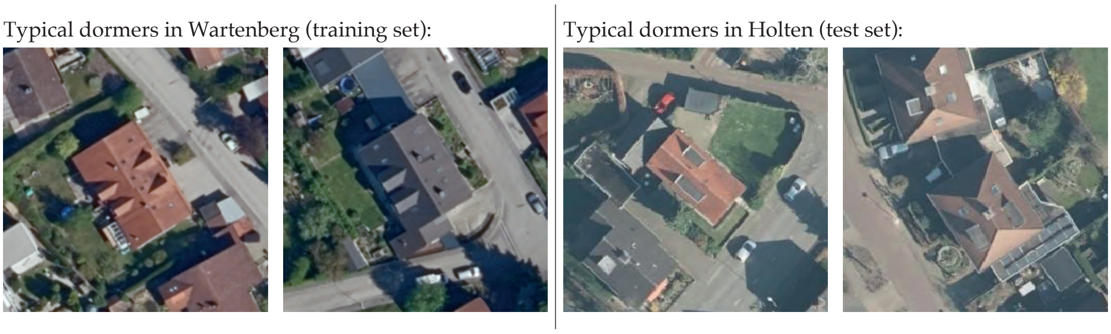

Typical dormers in Wartenberg (training set):
Typical dormers in Holten (test set):

Figure 5.24: Comparison of typical dormers from each dataset

Wartenberg's dormers are narrower and perpendicular to the main roof axis, whereas those of Holten are elongated and parallel to it. As a result, the network clearly identifies the narrow dormer of the second row in figure 5.24, but not the others. Thirdly, their colors and contrasts are different. Wartenberg presents mostly dormers whose material and shades are similar to the roof. On the other hand, Holten's architecture often employs a grey coating contrasting with the roof material. As a result, they are sometimes confused with PV modules (figure 5.23, second example).

Windows are overall well detected. However, they also include unknown elements due to shadowing effects. The training data was less sharp, involving such confusion.

Differently, the model hardly detects PV modules. This class shows the worst evolution in comparison to the Wartenberg test set. The different photographic conditions can explain this deterioration since Holten's PV modules appear mainly black or shiny. Also, the panels' scale seems slightly different between the datasets (figure 5.25). However, the cars on the images confirm that both datasets have comparable scales. Besides, figure 5.22 (last two examples) show that data augmentation does not improve predictions for that class.

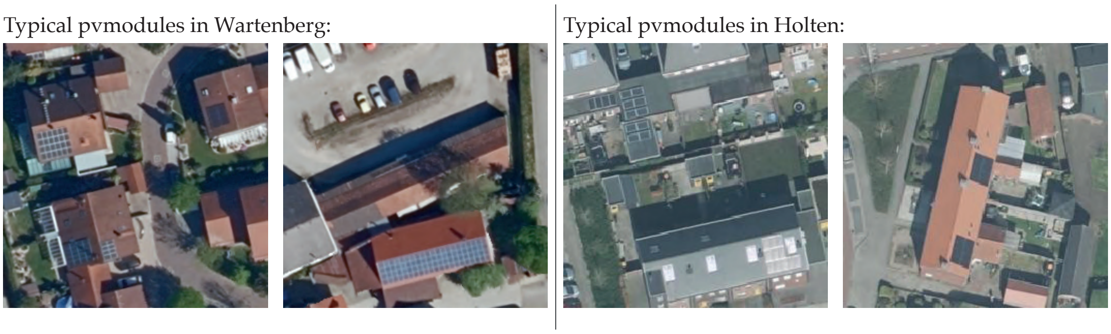

Typical pvmodules in Wartenberg:
Typical pvmodules in Holten:

Figure 5.25: Comparison of typical PV modules from each dataset

Conclusion. Even on a test area presenting different architectural typologies and datasets' quality, FuseNet improves chimney and dormer detection over U-Net. Nevertheless, detected chimneys and other classes rather correspond to RGB input pixels than those of height maps (figure 5.26). It is due to the asymmetry of FuseNet, prioritizing RGB over height data.

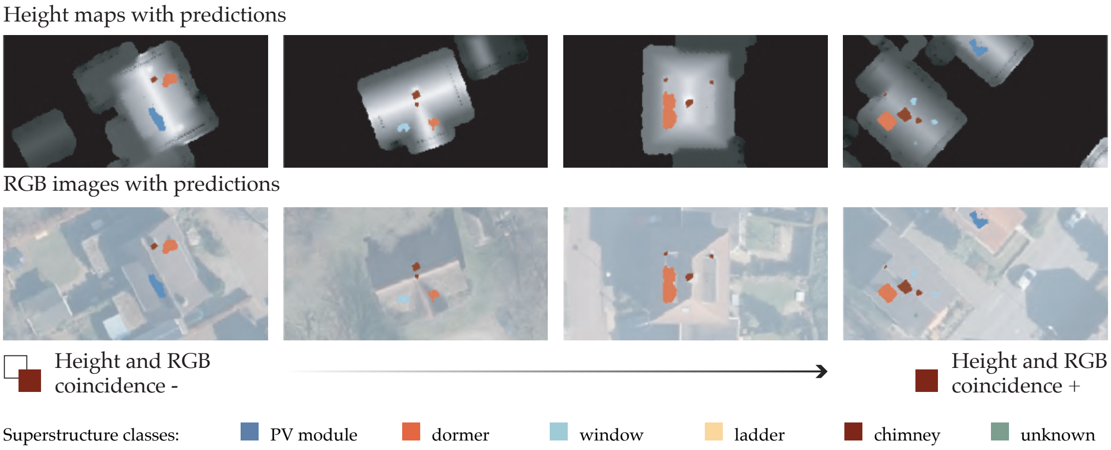


Figure 5.26: Coincidence of input data with the chimney predictions of FuseNet with augmentation

The activations from the height data branch are concatenated to the RGB ones during the encoding phase, whereas only the RGB branch is decoded. As a result, in the case of data mismatch, U-Net and FuseNet seem to perform similarly, whereas, if data coincide, FuseNet has a superior performance.

Given the quality of Holten's height maps (dormers are easily recognizable), a symmetric network such as the one proposed by Audebert et al. 2018 might be more suitable. However, it might be less tolerant with input data mismatches, which represents a disadvantage. Due to time constraints, the latter was not implemented.

5.4 3D model generation and application

Finally, the best predictions are vectorized and modeled in 3D. Superstructures' geometry and semantics are added to the LOD2 model available for Bavaria. This step is enhanced through the usage of ortho-rectified labels since superstructures correspond better to the buildings' footprint than those obtained from Google APs.

5.4.1 3D model generation

The model refinement through superstructure modelization was carried out by Bruhse 2022 using 3DCityDB. Chimneys and dormers are modeled volumetrically, whereas other classes are polygons parallel to the roof surfaces, as illustrated in figure 5.27. All include a semantic description. On top of geometrical modelization, the algorithm solves the issue of duplicate structures' detection, which occurs due to overlapping images.

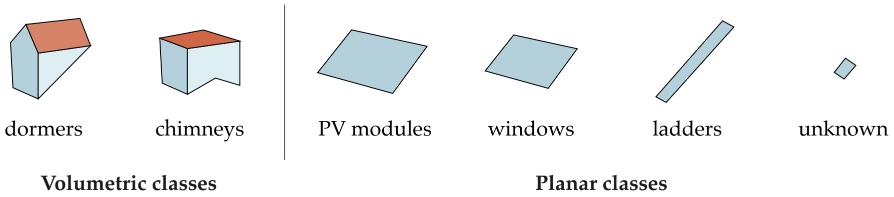


Figure 5.27: Superstructure representations in 3D

Both 2D polygonization and 3D modelization processes involve simplification mechanisms, affecting the reality of the final representation. At the current development stage, dormers can depict several typologies, whereas chimneys are extruded to a one-meter height above the roof plane. Other planar superstructures are modeled parallel to the corresponding roof segment. All are modeled as “BuildingInstallations” geometries. The included semantics enhance the model, but geometrical nuances are missing.

First, Dutch superstructures from Holten are modeled. The result can be compared to the existing detailed LOD available from 3D BAG (tudelft3d 2021). Then, ortho-labels from Wartenberg are employed to illustrate optimal results and their use case for PV-potential assessment.

Holten 3D model from predictions. Figure 5.28 and 5.29 compare the GT AP with the 3D model refined from predictions and the 3D model depicted by 3D-BAG as LOD2.2. The first figure illustrates a global example, where roofs and dormers appear coherent with the GT AP.

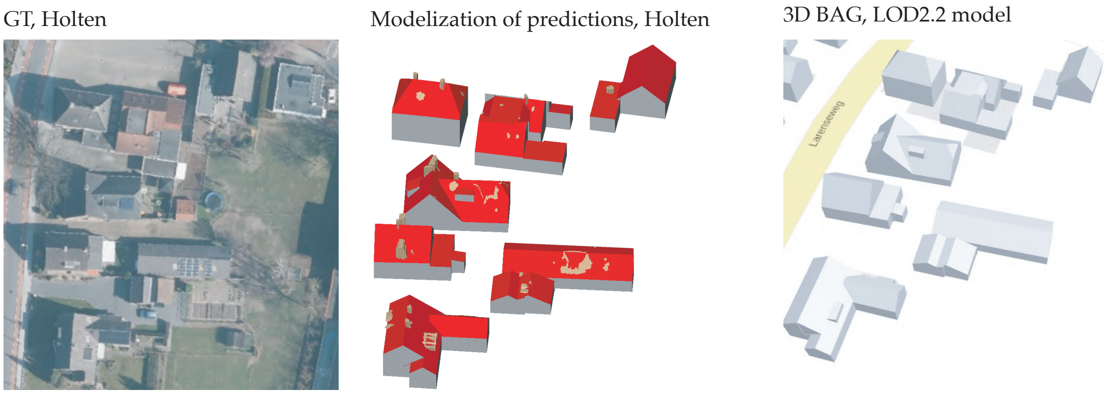


Figure 5.28: Visualization of Holten 3D model derived from predictions

Figure 5.29 shows two building examples zoomed-in. Dormers and chimneys are quite accurately detected by FuseNet. The second example shows that 3D-BAG represents a dormer exactly at the same location as detected by FuseNet since they are overlapping on the model.

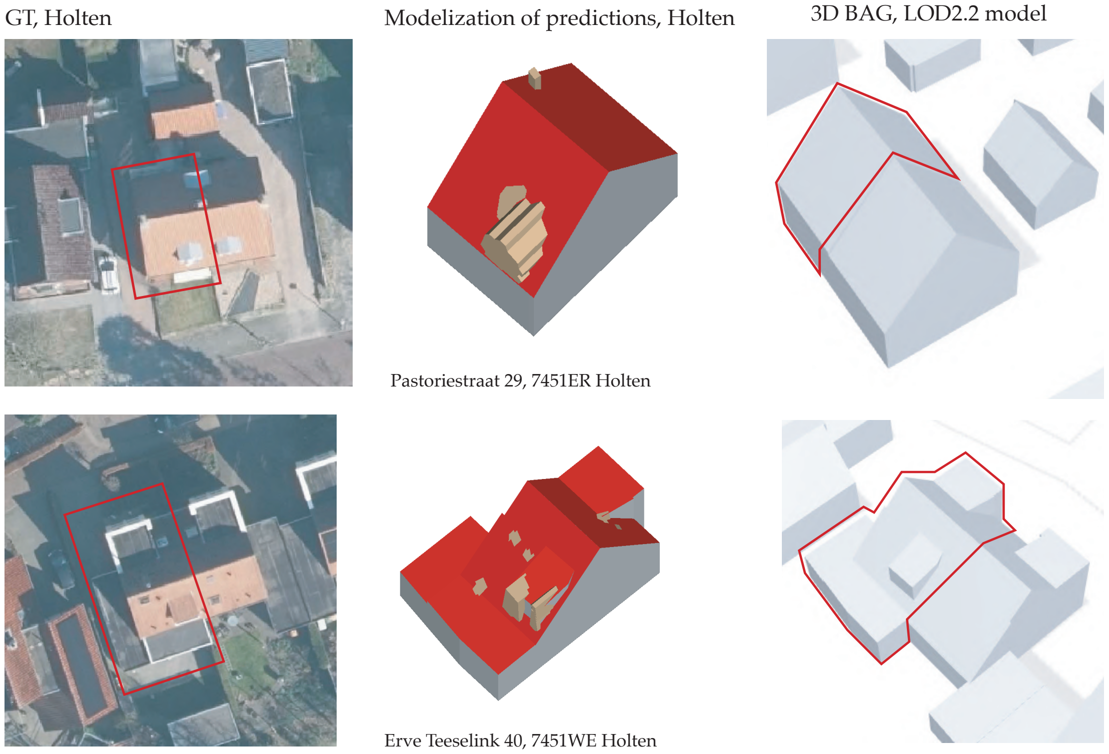

GT, Holten
Modelization of predictions, Holten
3D BAG, LOD2.2 model
Pastoriestraat 29, 7451ER Holten
Erve Teeselink 40, 7451WE Holten

Figure 5.29: Visualization of Holten 3D model derived from predictions: successful examples

Wartenberg 3D model from predictions and labels. Wartenberg test set presents more qualitative predictions. They are constructed in 3D as well. Figure 5.30 displays an extract of the village.

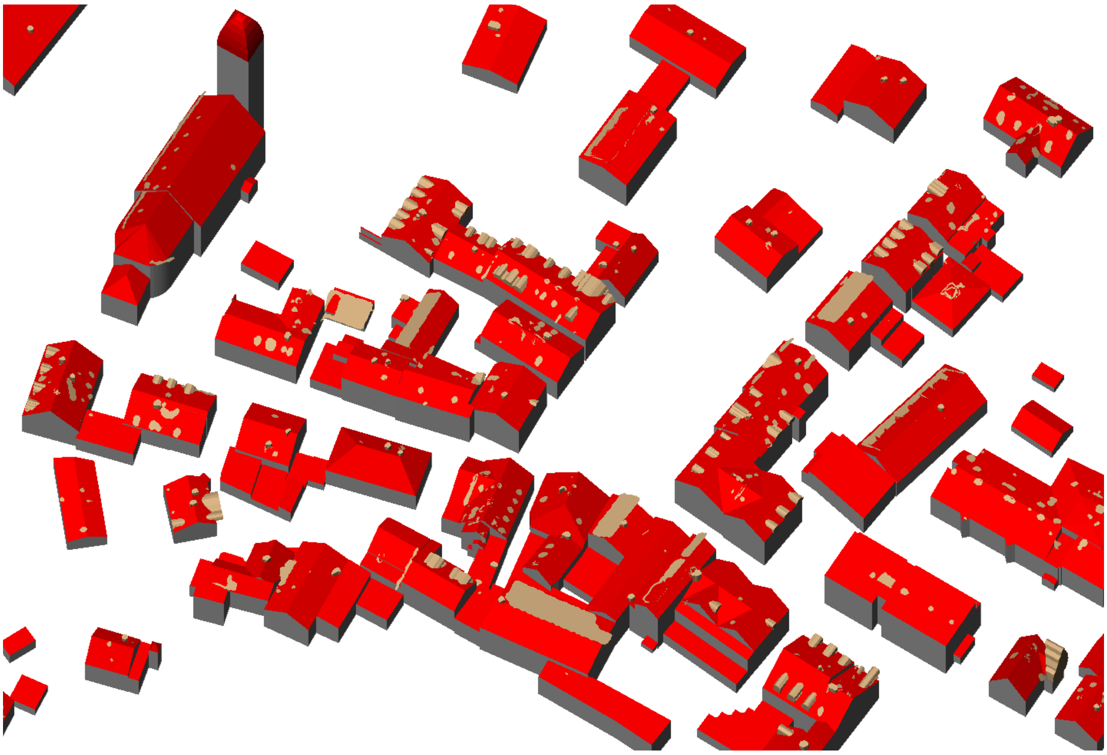


Figure 5.30: Visualization of Wartenberg 3D model derived from predictions of Bruhse 2022

Next, 3D modelization of the labels shows the case of perfect predictions (figure 5.31). Network and input refinements aim for such a 3D model. It is further used for the buildings' PV-assessment (Willenborg 2022).

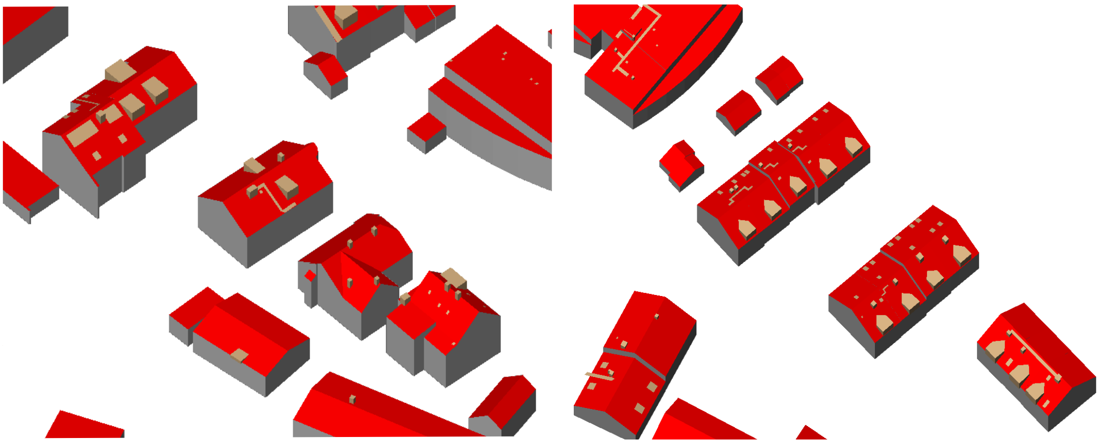


Figure 5.31: Visualization of Wartenberg 3D model derived from ortho-labels

5.4.2 3D model application

A use case of the resulting model, not depicting the superstructure geometries precisely, consists of buildings' PV-potential assessment. Once ortho-predictions (or labels) are modeled in 3D, the potential solar panels' layout based on the roof azimuth combined with its available surface can be carried out. The distribution and size of superstructures impact the panels' organization, whereas the azimuth affects the power generated per unit surface. Figure 5.32 illustrates the PV-potential of Wartenberg based on ortho-labels' depiction.

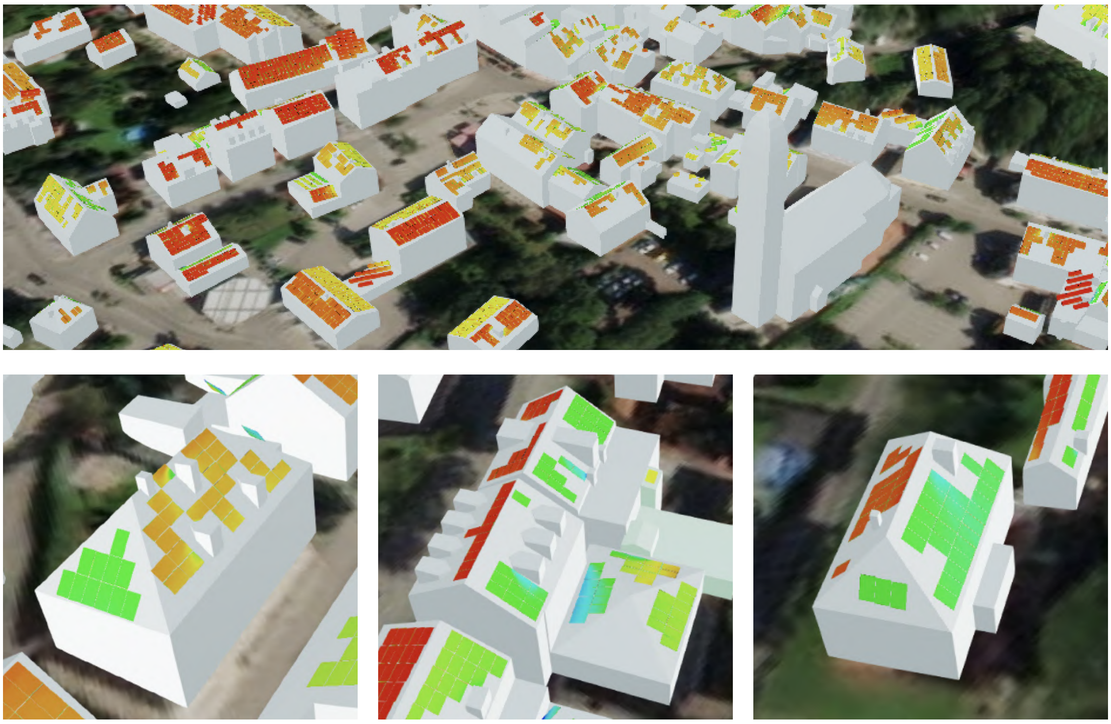


Figure 5.32: Example of buildings' PV potential assessment applied to Wartenberg model, visualization Cesium. Source: Willenborg 2022

Figure 5.33 depicts a sample building for which the superstructures largely impact the panels' distribution. Moreover, the predictions were qualitative compared to the original labels, confirming the potential of this research.

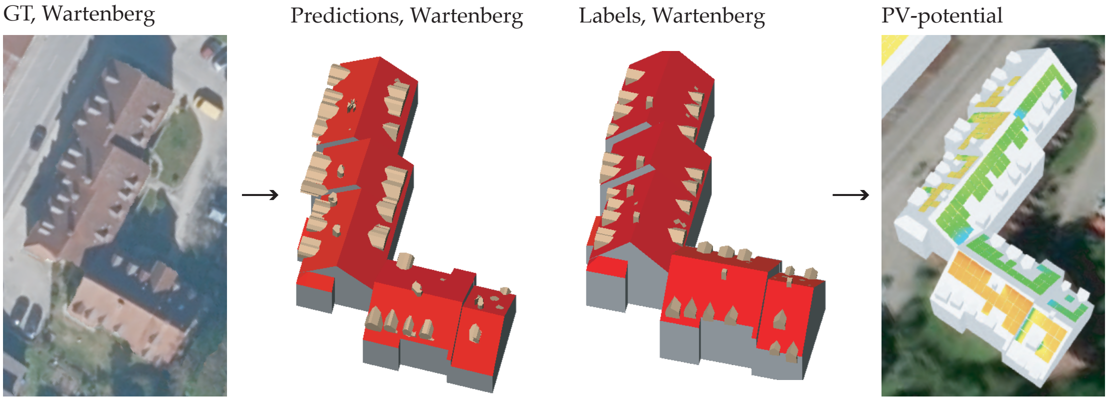


Figure 5.33: PV potential assessment for an example building

6 Conclusion, discussion and future work

The research demonstrates that building height data fused to RGB aerial images by means of a CNN improves the prediction accuracies and IoU scores for semantic segmentation of roof superstructures. Chimneys and dormers show the most significant enhancement.

Accuracies are improved for specific classes through height data fusion, whereas other classes do not benefit from this additional input. On the other hand, superstructure boundaries are harder to detect when based on several input datasets. Therefore, fusion network results include more background pixels classified within superstructures than predictions only based on aerial images.

Thereby, the hypotheses can be partly validated:

- **Improvement for some classes only.** This assumption is verified since installations with a relief (chimneys and dormers) show accuracy improvements, unlike other segmented classes. Additionally, the window class shows a slight improvement. However, the enhancement of ladder detection occurring through absolute height data fusion was not foreseen but can be explained by the indirect slope information integrated.

- **Absolute versus relative height.** Absolute height incorporation produces slightly higher scores than height data normalized through an available LOD model. The first dataset includes slope information indirectly, which benefits some classes. Additionally, absolute height requires a simpler processing pipeline, and relative elevation data include erroneous elevation due to the 3D model simplification mechanisms.

- **Interpolation methods.** The difference in performance between *nn* and *idw* interpolation techniques could not be determined using an asymmetric network prioritizing RGB input. However, *no* interpolation generated the worst results. This confirms the supposition that interpolated data is more efficient, which is explained by the enhanced understanding of superstructure scale and shape.

6.1 Discussion

Process efficiency can be assessed in addition to the result's quality. For this purpose, three main aspects are discussed: impact of labeling, resolution and timeframe of the input datasets, and practical implementation of the process.

First of all, labels were manually drawn on low-resolution images. Therefore, superstructure types and boundaries are hard to identify and annotation involves personal interpretations and decision-making.

Moreover, the classes were derived and determined based on the specific geographic area studied: a Bavarian village. Defining global superstructure classes is not straightforward due to local architectural characteristics. Hence, labels can hardly be used to train a network applied later to a different region. This was experimentally proven by using labels of Bavaria to test a Dutch dataset. Furthermore, defining classes within a single location is ambiguous: some entities could have been distinguished from the unknown class (e.g. snow fences).

Additionally, the robustness of the model is limited since training and validation datasets are extracted from the same area, and the number of labeled images is reduced to 1880. Furthermore, since the data is acquired for a village, it is mainly residential and entails only a few tertiary or industrial buildings. Extending the training data would involve determining more classes or expanding the definitions of current ones. In both cases, labeling and predictions become more challenging.

Finally, the labeled classes are imbalanced due to the dominant background. Accordingly, the superstructure surfaces are reduced to few pixels, especially the chimney class compared to dormers or PV modules. A way to deal with this issue is to implement label uncertainty that confers more importance to pixels located at the center of labeled areas than the ones positioned at their boundaries (Bressan et al. 2021).

Another aspect involves the resolution and acquisition timeframe of the datasets employed.

Firstly, the resolution of aerial images used, 20cm/pixel, turns out to be critically low for the task of roof superstructure segmentation.

Secondly, LiDAR and DOPs suffer from temporal mismatches. In the case studied, this resulted in an empty height raster for 63 buildings over 1880. Collecting and processing LiDAR data is time-consuming and costly. Therefore, time-lapses between different records are higher than those of aerial images. Building points are one of the LAS classifications the most affected by these time intervals since constructions change at a fast pace, and probably their superstructures even more. For the location used, Wartenberg, the PC is dating from 2012, reaching almost 10 years difference from the APs.

Also to consider, LiDAR datasets contain both first and last pulse records. The latter was chosen in order to access information underlying the canopy. However, both datasets present advantages, and combining them seems promising since first pulses tend to better detect volume edges and outlines, and last pulses contain information on elements covered by overhanging trees. Other preprocessing ideas include processing the LiDAR PC before generating height maps (e.g. filtering points, exaggerating their elevation values based on some criteria, etc.), and exploring various parameters for interpolation methods.

Lastly, the 3D city model is derived from the LiDAR record combined with cadastral data. Its generation involves simplification algorithms that alter real building shapes. This has detrimental consequences on the relative height, which is obtained by comparing LiDAR PC to the 3D model.

The last aspect relates to ML implementation and process.

First of all, the data split is crucial, and in this work was impacted by the use of roof-centered images that overlap each other. This building-centered approach was motivated by the focus on building scale and was possible since datasets are extracted from a village where most houses are independent. However, in more urbanized areas, grid images should be used since constructions are contiguous. It would prevent pictures from overlapping, although they might lack buildings when captured at the borders of metropolitan areas. An additional disadvantage of using a regular photography grid is that buildings might be cut over several images and their corresponding height data interpolation might be affected.

To solve the issue of overlapping images, an algorithm was implemented to obtain independent training, validation, and test datasets of 60%, 20%, and 20% respectively, with a 3% margin. As a result of density distribution in the village, the training dataset does not include buildings from urban borders and therefore almost no industrial ones. With more dispatched labeled data, the split would yield more randomness. However, this also implies that one cannot control which buildings are used to train, whereas some of them might be particularly relevant to learn from. As a result of this data split, the test dataset was reduced to 344 images on Wartenberg. Drawing conclusions from such a small set might not be fully reliable.

Furthermore, an asymmetric height data fusion implementation was used, which prioritizes RGB input over height maps. Such an architecture is a mid-way between stacking height data to the RGB input and maintaining distinct branches during the decoding phase as well. Different network architectures might come up with a refined conclusion, especially regarding the efficiency of *nn* and *idw*, respectively. However, more symmetric networks might be less tolerant with data input mismatches.

Finally, data fusion implies a more complex pipeline than CNN applied to simple aerial images. The datasets' preparation is time-consuming, computationally expensive, and requires the area of interest to dispose of both LiDAR data and qualitative APs. Relative height, besides involving more processing steps, necessitates an additional input (a 3D model) that is seldom openly available. Finally, model fitting involves more parameters and is, therefore, more computationally expensive. With twice the resolution used—which would be more suitable for superstructure detection—four times more pixels need to be classified, involving training parameters and computational costs to be accordingly extended.

6.2 Contributions

This research contributes to improving knowledge regarding semantic segmentation tasks performed at a building scale:

- **Added value of height data.** The investigations carried out with an ML fusion architecture confirm that interpolated height data benefit semantic segmentation for specific superstructure classes.

- **Labeling.** The work proved that labels on ortho-rectified data should be used for 3D data incorporation. The network learns better from inputs with coinciding patterns. Moreover, the classes used for this segmentation task are crucial. Two categories of annotations can be distinguished: the volumetric superstructures, improved through height data fusion, and the planar ones. Relevant classes should be defined accordingly.

- **Height data type.** The experiments conclude that interpolated height yields better performance than *no* interpolation dataset. Moreover, values extracted from a raw PC are preferred over normalized ones. They present more structural information, including insights on slope gradients. Additionally, their processing steps are smoother, and relative elevations are biased by the 3D model employed for their normalization.

6.3 Future work

Research can be pursued in multiple ways:

- Roof superstructure definition can be refined and categorized into two types, differentiating volumetric and planar installations. Fusion networks could be applied to the first category only. Pixel uncertainty of labels should be implemented to obtain a more accurate evaluation of the results and to better deal with low-resolution datasets and imbalanced classes.

- The pipeline to generate height data grids can be improved, both in terms of workflow and speed.
- Training and test data should be enlarged from other geographic areas. Augmentation, including height grids' transformations, can also be further explored to increase the robustness of the model.
- Other network architectures derived from the state-of-the-art should be tested to improve segmentation results. For example, the lightweight model *EDFT* achieves promising results on the ISPRS 2D dataset by developing another fusion module (Yan et al. 2022). Symmetric fusion architectures, such as Virtual-FuseNet, allow to equally balance RGB and height-data input instead of prioritizing aerial images over height maps (Audebert et al. 2018).
- The benefit of other dataset types can be evaluated. Regarding height data, PCs obtained from photogrammetry could be considered, although seldom available in large areas. Their color information is of interest. Other datasets different from height can be explored, such as infra-red to detect vegetation or HSRI for roof material identification.
- The pipeline and quality of the 3D city model refinement can be further improved. Superstructures' polygonization process could be enhanced to obtain more realistic shapes, e.g. by adjusting outlines that should be parallel to roof segments' delineation. An overall pipeline, from detection to modelization, could be elaborated.

6.4 Reflection

This thesis concludes a Master of Science in Geomatics. Consequently, this part reflects on two aspects: first, the work integration within the geomatics framework (data and methods); second, the research process efficiency and challenges encountered (product, process, and planning).

The work combines multiple aspects of geo-informatics: various input data sources and processing techniques.

First, data georeferencement is crucial to combine several sources: aerial images with their corresponding PC. Experiments based on different geographical locations highlights the system interoperability. Moreover, all types of geo-data are employed: vector (drawing of labels), raster (APs and height data interpolation), and PC (elevation source). Therefore, the research demonstrates the advantages and complementarity of each sort.

Secondly, the process is built upon tools learned during the past two years: programming languages (Python, SQL, C++), database management system (3DCityDB), city and terrain modeling (GML format, LODs, interpolation techniques), etc. However, the courses do not commonly include DL, although its principles are increasingly applied to georeferenced 2D and 3D data, thereby illustrating one of the application domains of geomatics.

The work process is analyzed chronologically: goal definition, planning, reason for its adjustments, and retrospective.

The aim was defined as improving the CNN detection of roof superstructures through 3D data integration to RGB. This enhanced output benefits different purposes while inscribing itself in the program specialized in the Built Environment.

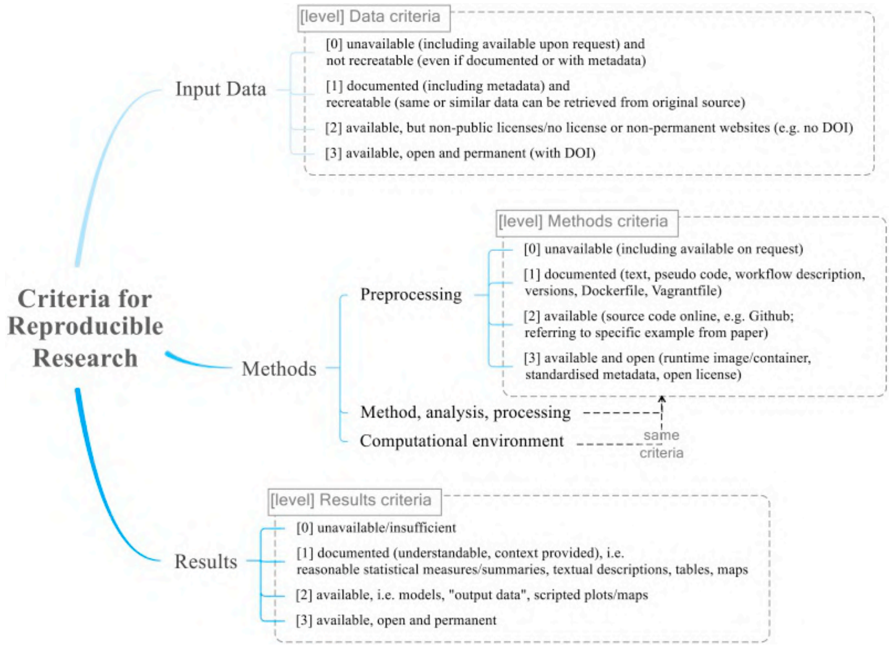
Planning evolved through the process due to the intricacy and time consumption to generate appropriate input data for DL. As a result, a learning output is that for DL, data preparation is a fundamental part whose time consumption should be correctly estimated. However, adapting the existing frameworks and running tests and predictions were fast enough to catch up on the lost time. A benefit of data low-resolution and small training set is that one FuseNet training with GPU lasted around 8 hours, enabling several of them per week. Nevertheless, experiments tested only one fusion architecture, unlike scheduled at the beginning.

More specifically, a challenge encountered was the heaviness of geo-data that are therefore complicated to process on large scales. As a result, the speed of data preprocessing could have been improved. Since time efficiency was not a priority, some parts remain ineffective, involving algorithms to run for hours. Accordingly, characteristics specific to geo-data should be further explored (e.g. spatial indices, database management combined with files), and a compiled language (like C++) could have been more extensively employed. Additionally, the labeling process could gain efficiency. The adaptation of existing labels (for non-ortho-rectified images) to DOPs was not straightforward. Different distortions were involved throughout the dataset, preventing from automatizing the process.

Looking back, since most labels were drawn again, another location with more qualitative datasets could have been chosen instead of Wartenberg.

Finally, the cooperation between the two universities grants both sides a vision of different perceptions and geo-data usages. The project in Munich is a collaboration between two faculties (Geo-informatics and Automotive Engineering). On the other hand, Delft specializes in the Built Environment. These overlaps between fields highlight the broad societal need for geo-data analyses.

A.1. Reproducibility criteria



Criteria for Reproducible Research
Input Data
[level] Data criteria
[0] unavailable (including available upon request) and not recreatable (even if documented or with metadata)
[1] documented (including metadata) and recreatable (same or similar data can be retrieved from original source)
[2] available, but non-public licenses/no license or non-permanent websites (e.g. no DOI)
[3] available, open and permanent (with DOI)
Methods
Preprocessing
[level] Methods criteria
[0] unavailable (including available on request)
[1] documented (text, pseudo code, workflow description, versions, Dockerfile, Vagrantfile)
[2] available (source code online, e.g. Github; referring to specific example from paper)
[3] available and open (runtime image/container, standardised metadata, open license)
Method, analysis, processing
Computational environment
same criteria
Results
[level] Results criteria
[0] unavailable/insufficient
[1] documented (understandable, context provided), i.e. reasonable statistical measures/summaries, textual descriptions, tables, maps
[2] available, i.e. models, "output data", scripted plots/maps
[3] available, open and permanent

Figure A.1: Reproducibility criteria; source: Nüst et al. 2018

A.2. Reproducibility assessment

The research is motivated by its reproducibility potential, which is partly possible since based on open data and implementation frameworks. Reproducibility scores based on Nüst et al. 2018 (figure A.1) are indicated in table A.1.

Part	**Component**	**Score (out of 3)**
1a. Input data (Wartenberg)	Aerial images	0 — unavailable
	LOD2 model	0
	LiDAR PC	0
	Training data	0
1b. Input data (the Netherlands)	Aerial images	3 — available, open, permanent
	LOD2 model	3
	LiDAR PC	3
2. Methods	Preprocessing	2 — available
	Method	1 — documented
	Analysis	1 — documented
	Processing (DL implementations)	3 — available, open
	DL frameworks	3
	Computational environment	1 — documented
3. Results	Quantitative results	1 — documented
	Qualitative results	1

Table A.1: Reproducibility scores

First, DL frameworks and implementation employed are public. Second, the algorithm for height data preparation is available on GitHub, whereas the method and analysis are documented.

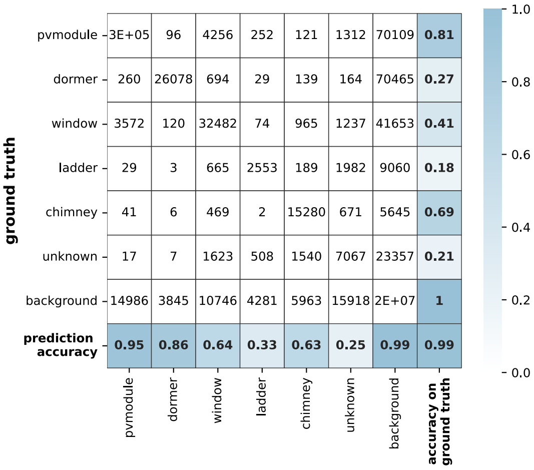
Regarding input data, Bavaria does not propose its HR datasets open-source, contrarily to the Netherlands. They are set available to the University of Munich for research purposes. Therefore, the two locations present opposite scores. The detrimental consequence of unavailable datasets for Bavaria is that labels, combined with their corresponding APs, cannot be shared.

Favorably, the European directive INSPIRE (INfrastructure for SPatial InfoRmation in Europe) provides a legal framework for a “Europe spatial data infrastructure” (INSPIRE 2022). Therefore, these APs and other datasets should be open in the coming years. Increased datasets’ availability and quality would benefit environmental policies, research communities, and economic growth. Accordingly, works developed on open datasets become increasingly appropriate and integrated into societal needs.

B. Quantitative results U-Net

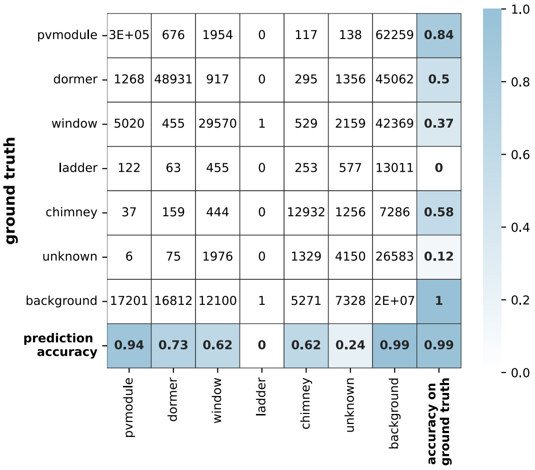
id	split	epoch	loss	base	augmentation	mean Acc. GT	mean Pr.Acc.	IoU
1	01	40	CFL+Dice	vgg19	None	**0.47**	**0.53**	**0.45**
2	01	40	CFL+Jaccard	vgg19	None	**0.43**	**0.58**	**0.45**
3	01	40	0.5(CFL+Dice) +0.5(CFL+Jacc.)	vgg19	None	**0.43**	**0.61**	**0.44**
4	01	40	CFL+Jaccard	resnet152	None	**0.40**	**0.53**	**0.43**
5	02	40	CFL+Jaccard	vgg19	None	**0.44**	**0.63**	**0.46**
6	02	40	CFL+Jaccard	resnet152	None	**0.42**	**0.62**	**0.45**
7	01	80	CFL+Jaccard	vgg19	None	**0.42**	**0.62**	**0.45**
8	02	80	CFL+Jaccard	vgg19	None	**0.46**	**0.64**	**0.47**
9	01	40	0.5(CFL+Dice) +0.5(CFL+Jacc.)	resnet152	Train Val.	**0.48**	**0.58**	**0.47**
10	01	40	0.5(CFL+Dice) +0.5(CFL+Jacc.)	resnet152	Train Val. Test	**0.46**	**0.58**	**0.46**

Table B.1: U-Net experiments, parameters and accuracy results. "Id" is the experience id used for easier referencing. In blue, best results, reference for later comparison.



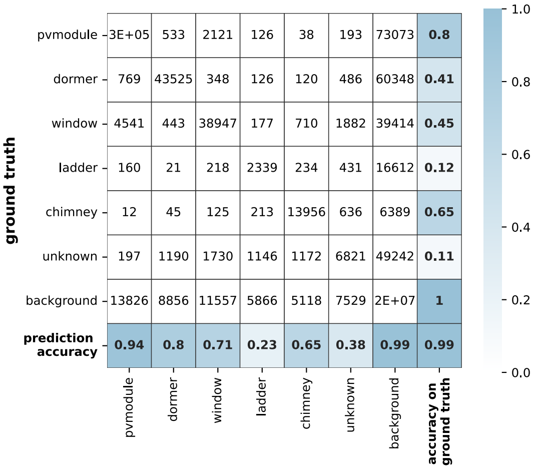
	pvmodule	dormer	window	ladder	chimney	unknown	background	accuracy on ground truth
pvmodule	3E+05	96	4256	252	121	1312	70109	**0.81**
dormer	260	26078	694	29	139	164	70465	**0.27**
window	3572	120	32482	74	965	1237	41653	**0.41**
ladder	29	3	665	2553	189	1982	9060	**0.18**
chimney	41	6	469	2	15280	671	5645	**0.69**
unknown	17	7	1623	508	1540	7067	23357	**0.21**
background	14986	3845	10746	4281	5963	15918	2E+07	**1**
prediction accuracy	**0.95**	**0.86**	**0.64**	**0.33**	**0.63**	**0.25**	**0.99**	**0.99**

Figure B.1: Confusion matrix split-01, id-3



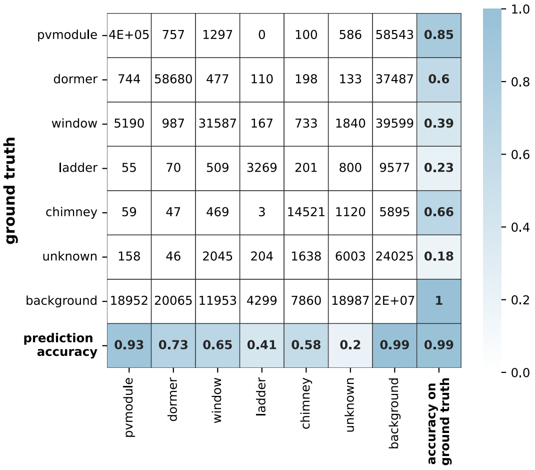
	pvmodule	dormer	window	ladder	chimney	unknown	background	accuracy on ground truth
pvmodule	3E+05	676	1954	0	117	138	62259	**0.84**
dormer	1268	48931	917	0	295	1356	45062	**0.5**
window	5020	455	29570	1	529	2159	42369	**0.37**
ladder	122	63	455	0	253	577	13011	**0**
chimney	37	159	444	0	12932	1256	7286	**0.58**
unknown	6	75	1976	0	1329	4150	26583	**0.12**
background	17201	16812	12100	1	5271	7328	2E+07	**1**
prediction accuracy	**0.94**	**0.73**	**0.62**	**0**	**0.62**	**0.24**	**0.99**	**0.99**

Figure B.2: Confusion matrix split-01, id-4



	pvmodule	dormer	window	ladder	chimney	unknown	background	accuracy on ground truth
pvmodule	3E+05	533	2121	126	38	193	73073	**0.8**
dormer	769	43525	348	126	120	486	60348	**0.41**
window	4541	443	38947	177	710	1882	39414	**0.45**
ladder	160	21	218	2339	234	431	16612	**0.12**
chimney	12	45	125	213	13956	636	6389	**0.65**
unknown	197	1190	1730	1146	1172	6821	49242	**0.11**
background	13826	8856	11557	5866	5118	7529	2E+07	**1**
prediction accuracy	**0.94**	**0.8**	**0.71**	**0.23**	**0.65**	**0.38**	**0.99**	**0.99**

Figure B.3: Confusion matrix split-02, id-6



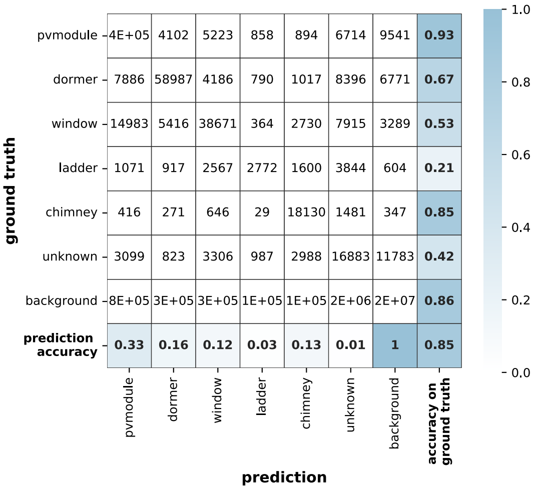
	pvmodule	dormer	window	ladder	chimney	unknown	background	accuracy on ground truth
pvmodule	4E+05	757	1297	0	100	586	58543	**0.85**
dormer	744	58680	477	110	198	133	37487	**0.6**
window	5190	987	31587	167	733	1840	39599	**0.39**
ladder	55	70	509	3269	201	800	9577	**0.23**
chimney	59	47	469	3	14521	1120	5895	**0.66**
unknown	158	46	2045	204	1638	6003	24025	**0.18**
background	18952	20065	11953	4299	7860	18987	2E+07	**1**
prediction accuracy	**0.93**	**0.73**	**0.65**	**0.41**	**0.58**	**0.2**	**0.99**	**0.99**

Figure B.4: Confusion matrix split-02, id-9

C. Quantitative results FuseNet

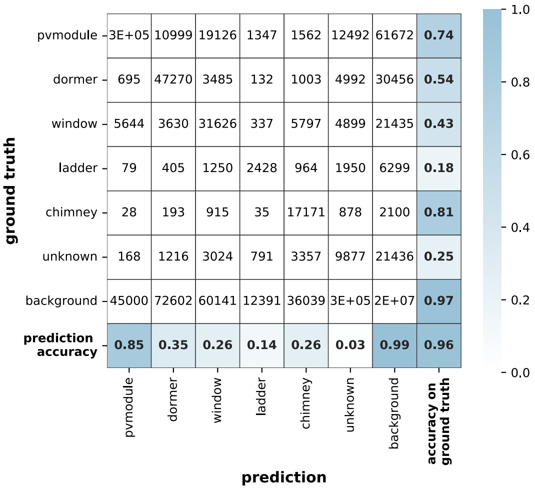
id	split	bgrd weight	epoch	α	batch-size	optimizer	loss	mean Acc.GT	mean Pr.Acc.	IoU
1a	01	0.000003	100	0.01 dc25	8	SGD	CE	**0.65**	**0.09**	**0.17**
1b	01	0.000003	400	0.01 dc25	8	SGD	CE	**0.60**	**0.13**	**0.23**
2	01	0.02	50	0.01 dc25	8	SGD	CE	**0.44**	**0.45**	**0.40**
3	01	0.02	50	0.01 dc25	16	SGD	CE	**0.42**	**0.37**	**0.36**
4a	01	0.02	50	0.01 dc25	4	SGD	CE	**0.45**	**0.52**	**0.42**
4b	01	0.02	100	0.01 dc25	4	SGD	CE	**0.47**	**0.51**	**0.44**
4c	01	0.02	144	0.01 dc25	4	SGD	CE	**0.48**	**0.54**	**0.45**
5a	01	0.03	100	0.01 dc25	4	SGD	CE	**0.43**	**0.53**	**0.42**
5b	01	0.03	200	0.01 dc25	4	SGD	CE	**0.45**	**0.54**	**0.44**
6	01	0.01	100	0.01 dc25	4	SGD	CE	**0.44**	**0.52**	**0.43**
7a	01	0.02	50	0.01 dc50	4	SGD	CE	**0.43**	**0.56**	**0.43**
7b	01	0.02	111	0.01 dc50	4	SGD	CE	**0.47**	**0.56**	**0.46**
7c	01	0.02	140	0.01 dc50	4	SGD	CE	**0.49**	**0.55**	**0.46**
7d	01	0.02	144	0.01 dc50	4	SGD	CE	**0.50**	**0.53**	**0.46**
8	01	0.02	50	0.05 dc10	4	SGD	CE	**0.42**	**0.35**	**0.34**
9	01	0.02	750	.001 dc50	4	SGD	CE	**0.17**	**0.26**	**0.21**
10	01	0.02	50	0.01 dc25	4	SGD	Dice+CE	**0.43**	**0.52**	**0.42**
11	01	0.02	300	.005 dc20	4	SGD	Dice+CE	**0.42**	**0.52**	**0.42**
12a	01	0.02	50	0.01 dc20	4	SGD	Dice+CE	**0.43**	**0.25**	**0.29**
12b	01	0.02	100	0.01 dc20	4	SGD	Dice+CE	**0.49**	**0.32**	**0.35**
12c	01	0.02	150	0.01 dc20	4	SGD	Dice+CE	**0.45**	**0.32**	**0.35**
13a	01	0.02	100	.01 no-dc	4	SGD	CE	**0.48**	**0.41**	**0.39**
13b	01	0.02	135	.01 no-dc	4	SGD	CE	**0.48**	**0.57**	**0.46**
13c	01	0.02	143	.01 no-dc	4	SGD	CE	**0.51**	**0.56**	**0.46**

Table C.1: Experiments on FuseNet: parameters and accuracy results using absolute nn-interpolation height dataset. *bgrd* = background, *dc* = decay, "id" refers to the experiment id with a letter meaning similar parameters but accuracies measured for different epochs



	pvmodule	dormer	window	ladder	chimney	unknown	background	accuracy on ground truth
pvmodule	4E+05	4102	5223	858	894	6714	9541	**0.93**
dormer	7886	58987	4186	790	1017	8396	6771	**0.67**
window	14983	5416	38671	364	2730	7915	3289	**0.53**
ladder	1071	917	2567	2772	1600	3844	604	**0.21**
chimney	416	271	646	29	18130	1481	347	**0.85**
unknown	3099	823	3306	987	2988	16883	11783	**0.42**
background	8E+05	3E+05	3E+05	1E+05	1E+05	2E+06	2E+07	**0.86**
prediction accuracy	**0.33**	**0.16**	**0.12**	**0.03**	**0.13**	**0.01**	**1**	**0.85**

Figure C.1: Confusion matrix split-01, id-1b



	pvmodule	dormer	window	ladder	chimney	unknown	background	accuracy on ground truth
pvmodule	3E+05	10999	19126	1347	1562	12492	61672	**0.74**
dormer	695	47270	3485	132	1003	4992	30456	**0.54**
window	5644	3630	31626	337	5797	4899	21435	**0.43**
ladder	79	405	1250	2428	964	1950	6299	**0.18**
chimney	28	193	915	35	17171	878	2100	**0.81**
unknown	168	1216	3024	791	3357	9877	21436	**0.25**
background	45000	72602	60141	12391	36039	3E+05	2E+07	**0.97**
prediction accuracy	**0.85**	**0.35**	**0.26**	**0.14**	**0.26**	**0.03**	**0.99**	**0.96**

Figure C.2: Confusion matrix split-01, id-12b

D. Comparison U-Net versus FuseNet

Difference between best FuseNet *absolute* height confusion matrices and U-Net id-1

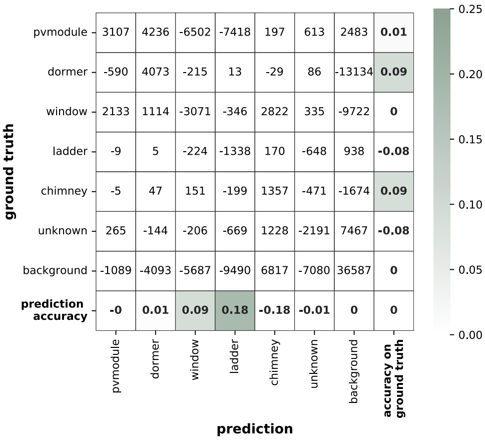


Figure D.1: FuseNet (id-4c) - U-Net (id-1)

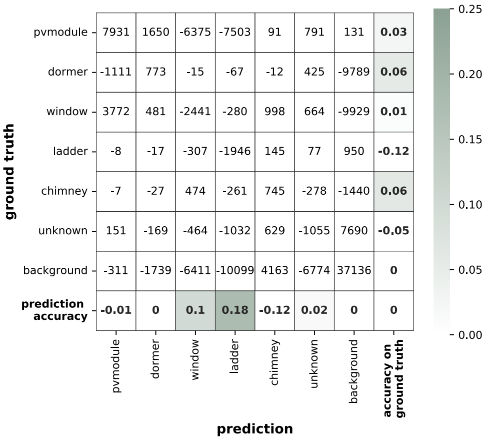


Figure D.2: FuseNet (id-7b) - U-Net (id-1)

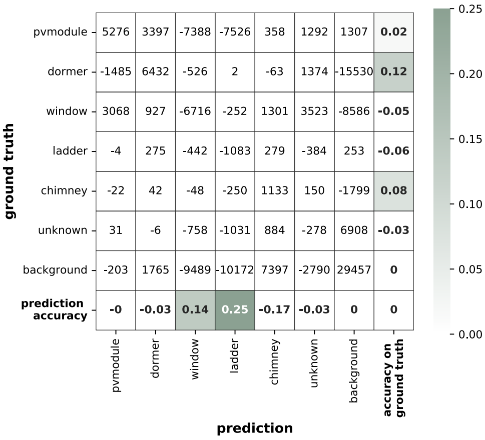


Figure D.3: FuseNet (id-7c) - U-Net (id-1)

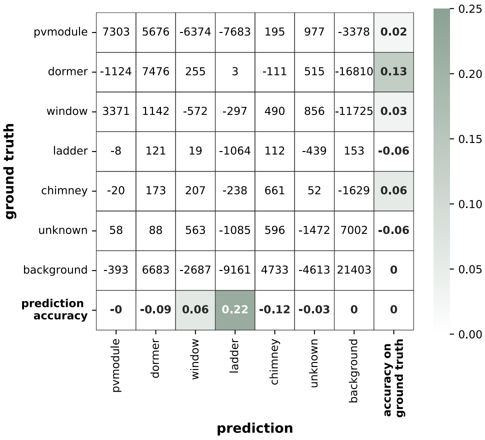


Figure D.4: FuseNet (id-7d) - U-Net (id-1)

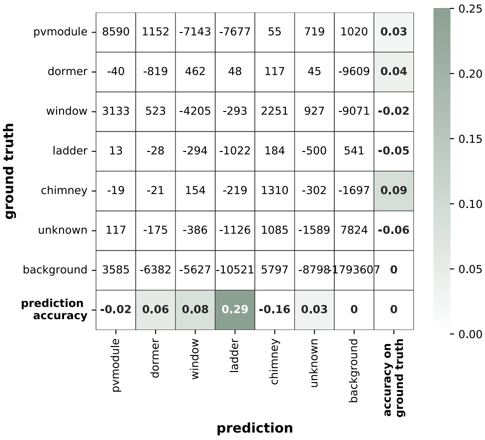


Figure D.5: FuseNet (id-13b) - U-Net (id-1)

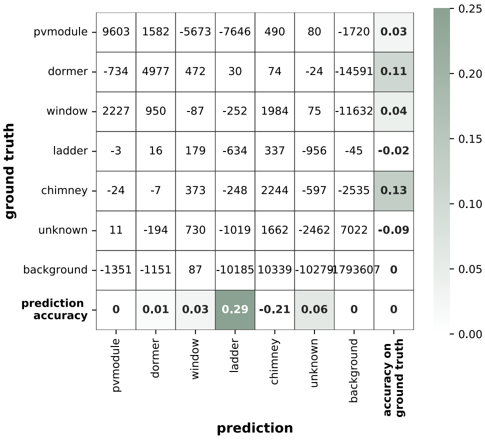
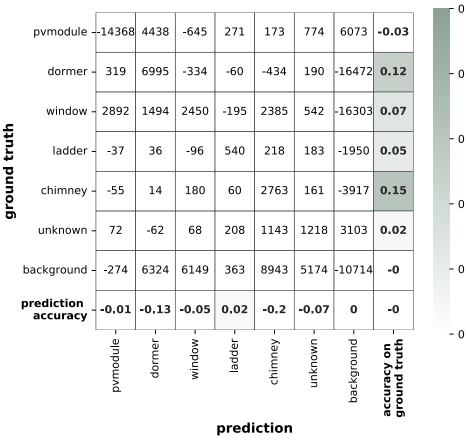


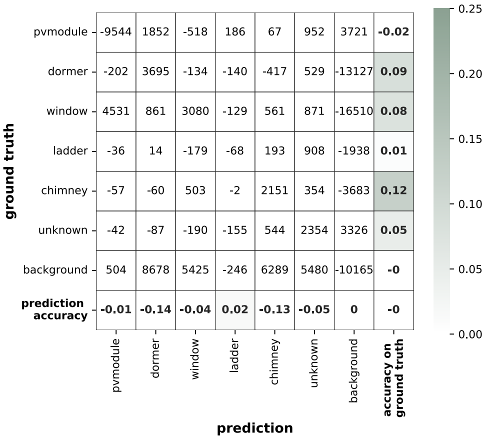
Figure D.6: FuseNet (id-13c) - U-Net (id-1)

Difference between best FuseNet confusion matrices *absolute* height and U-Net id-7



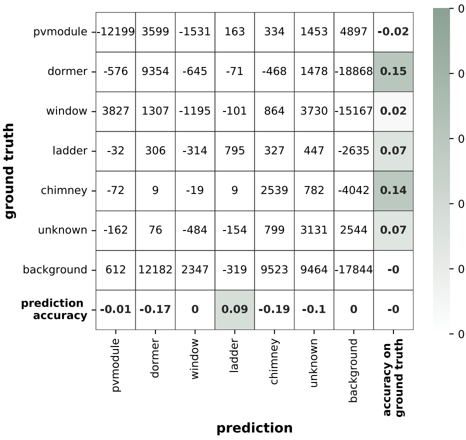
	pvmodule	dormer	window	ladder	chimney	unknown	background	**accuracy on ground truth**
pvmodule	-14368	4438	-645	271	173	774	6073	**-0.03**
dormer	319	6995	-334	-60	-434	190	-16472	**0.12**
window	2892	1494	2450	-195	2385	542	-16303	**0.07**
ladder	-37	36	-96	540	218	183	-1950	**0.05**
chimney	-55	14	180	60	2763	161	-3917	**0.15**
unknown	72	-62	68	208	1143	1218	3103	**0.02**
background	-274	6324	6149	363	8943	5174	-10714	**-0**
prediction accuracy	**-0.01**	**-0.13**	**-0.05**	**0.02**	**-0.2**	**-0.07**	**0**	**-0**

Figure D.7: FuseNet (id-4c) - U-Net (id-7)



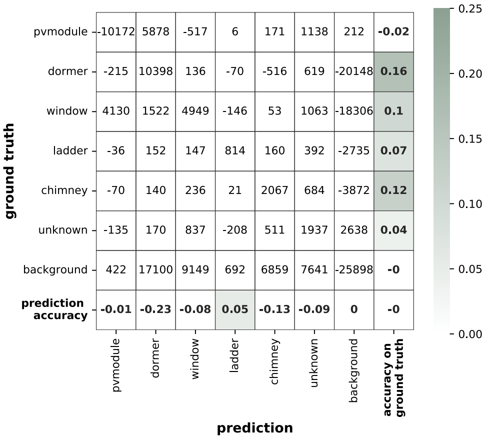
	pvmodule	dormer	window	ladder	chimney	unknown	background	**accuracy on ground truth**
pvmodule	-9544	1852	-518	186	67	952	3721	**-0.02**
dormer	-202	3695	-134	-140	-417	529	-13127	**0.09**
window	4531	861	3080	-129	561	871	-16510	**0.08**
ladder	-36	14	-179	-68	193	908	-1938	**0.01**
chimney	-57	-60	503	-2	2151	354	-3683	**0.12**
unknown	-42	-87	-190	-155	544	2354	3326	**0.05**
background	504	8678	5425	-246	6289	5480	-10165	**-0**
prediction accuracy	**-0.01**	**-0.14**	**-0.04**	**0.02**	**-0.13**	**-0.05**	**0**	**-0**

Figure D.8: FuseNet (id-7b) - U-Net (id-7)



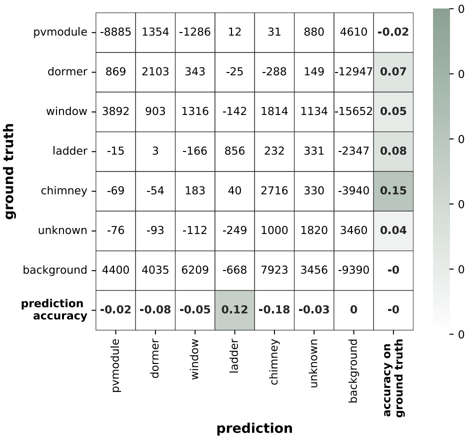
	pvmodule	dormer	window	ladder	chimney	unknown	background	**accuracy on ground truth**
pvmodule	-12199	3599	-1531	163	334	1453	4897	**-0.02**
dormer	-576	9354	-645	-71	-468	1478	-18868	**0.15**
window	3827	1307	-1195	-101	864	3730	-15167	**0.02**
ladder	-32	306	-314	795	327	447	-2635	**0.07**
chimney	-72	9	-19	9	2539	782	-4042	**0.14**
unknown	-162	76	-484	-154	799	3131	2544	**0.07**
background	612	12182	2347	-319	9523	9464	-17844	**-0**
prediction accuracy	**-0.01**	**-0.17**	**0**	**0.09**	**-0.19**	**-0.1**	**0**	**-0**

Figure D.9: FuseNet (id-7c) - U-Net (id-7)



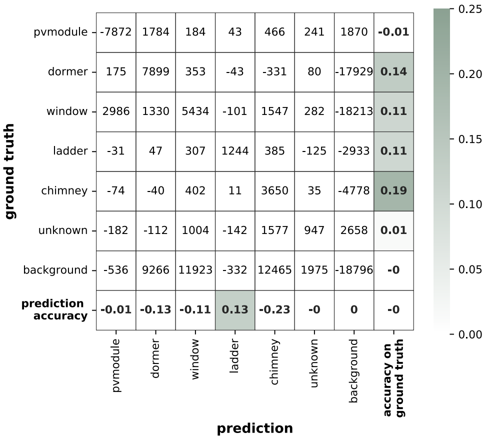
	pvmodule	dormer	window	ladder	chimney	unknown	background	**accuracy on ground truth**
pvmodule	-10172	5878	-517	6	171	1138	212	**-0.02**
dormer	-215	10398	136	-70	-516	619	-20148	**0.16**
window	4130	1522	4949	-146	53	1063	-18306	**0.1**
ladder	-36	152	147	814	160	392	-2735	**0.07**
chimney	-70	140	236	21	2067	684	-3872	**0.12**
unknown	-135	170	837	-208	511	1937	2638	**0.04**
background	422	17100	9149	692	6859	7641	-25898	**-0**
prediction accuracy	**-0.01**	**-0.23**	**-0.08**	**0.05**	**-0.13**	**-0.09**	**0**	**-0**

Figure D.10: FuseNet (id-7d) - U-Net (id-7)



	pvmodule	dormer	window	ladder	chimney	unknown	background	**accuracy on ground truth**
pvmodule	-8885	1354	-1286	12	31	880	4610	**-0.02**
dormer	869	2103	343	-25	-288	149	-12947	**0.07**
window	3892	903	1316	-142	1814	1134	-15652	**0.05**
ladder	-15	3	-166	856	232	331	-2347	**0.08**
chimney	-69	-54	183	40	2716	330	-3940	**0.15**
unknown	-76	-93	-112	-249	1000	1820	3460	**0.04**
background	4400	4035	6209	-668	7923	3456	-9390	**-0**
prediction accuracy	**-0.02**	**-0.08**	**-0.05**	**0.12**	**-0.18**	**-0.03**	**0**	**-0**

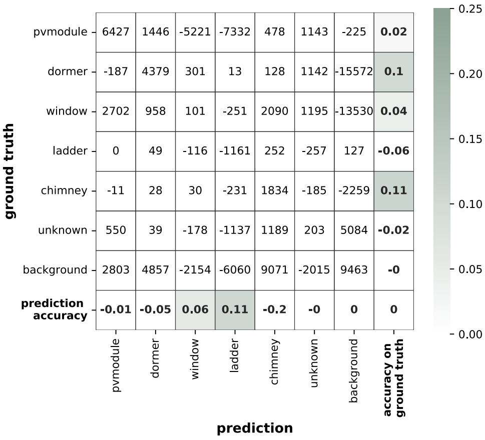
Figure D.11: FuseNet (id-13b) - U-Net (id-7)



	pvmodule	dormer	window	ladder	chimney	unknown	background	**accuracy on ground truth**
pvmodule	-7872	1784	184	43	466	241	1870	**-0.01**
dormer	175	7899	353	-43	-331	80	-17929	**0.14**
window	2986	1330	5434	-101	1547	282	-18213	**0.11**
ladder	-31	47	307	1244	385	-125	-2933	**0.11**
chimney	-74	-40	402	11	3650	35	-4778	**0.19**
unknown	-182	-112	1004	-142	1577	947	2658	**0.01**
background	-536	9266	11923	-332	12465	1975	-18796	**-0**
prediction accuracy	**-0.01**	**-0.13**	**-0.11**	**0.13**	**-0.23**	**-0**	**0**	**-0**

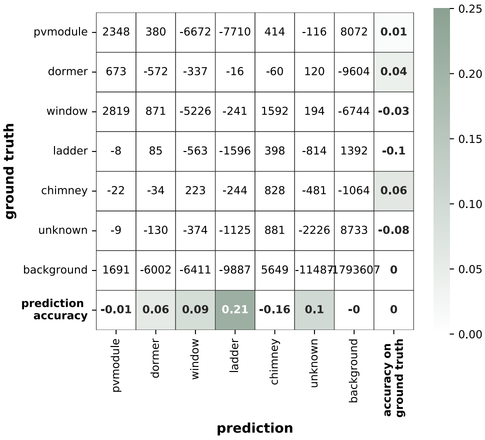
Figure D.12: FuseNet (id-13c) - U-Net (id-7)

Difference between best FuseNet confusion matrices *relative* height and U-Net id-1



ground truth \ prediction	pvmodule	dormer	window	ladder	chimney	unknown	background	accuracy on ground truth
pvmodule	6427	1446	-5221	-7332	478	1143	-225	**0.02**
dormer	-187	4379	301	13	128	1142	-15572	**0.1**
window	2702	958	101	-251	2090	1195	-13530	**0.04**
ladder	0	49	-116	-1161	252	-257	127	**-0.06**
chimney	-11	28	30	-231	1834	-185	-2259	**0.11**
unknown	550	39	-178	-1137	1189	203	5084	**-0.02**
background	2803	4857	-2154	-6060	9071	-2015	9463	**-0**
prediction accuracy	**-0.01**	**-0.05**	**0.06**	**0.11**	**-0.2**	**-0**	**0**	**0**

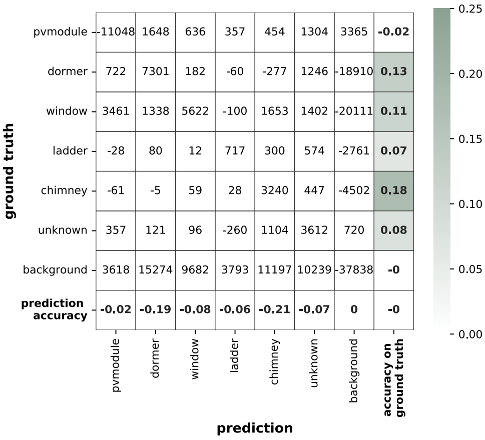
Figure D.13: FuseNet (id-v) - U-Net (id-1)



ground truth \ prediction	pvmodule	dormer	window	ladder	chimney	unknown	background	accuracy on ground truth
pvmodule	2348	380	-6672	-7710	414	-116	8072	**0.01**
dormer	673	-572	-337	-16	-60	120	-9604	**0.04**
window	2819	871	-5226	-241	1592	194	-6744	**-0.03**
ladder	-8	85	-563	-1596	398	-814	1392	**-0.1**
chimney	-22	-34	223	-244	828	-481	-1064	**0.06**
unknown	-9	-130	-374	-1125	881	-2226	8733	**-0.08**
background	1691	-6002	-6411	-9887	5649	-11487	-1793607	**0**
prediction accuracy	**-0.01**	**0.06**	**0.09**	**0.21**	**-0.16**	**0.1**	**-0**	**0**

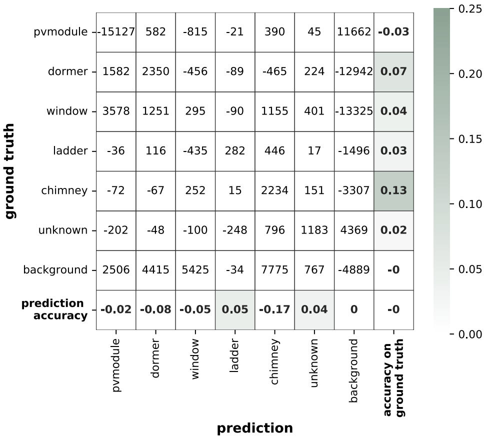
Figure D.14: FuseNet (id-vi) - U-Net (id-1)

Difference between best FuseNet confusion matrices *relative* height and U-Net id-7



ground truth \ prediction	pvmodule	dormer	window	ladder	chimney	unknown	background	accuracy on ground truth
pvmodule	-11048	1648	636	357	454	1304	3365	**-0.02**
dormer	722	7301	182	-60	-277	1246	-18910	**0.13**
window	3461	1338	5622	-100	1653	1402	-20111	**0.11**
ladder	-28	80	12	717	300	574	-2761	**0.07**
chimney	-61	-5	59	28	3240	447	-4502	**0.18**
unknown	357	121	96	-260	1104	3612	720	**0.08**
background	3618	15274	9682	3793	11197	10239	-37838	**-0**
prediction accuracy	**-0.02**	**-0.19**	**-0.08**	**-0.06**	**-0.21**	**-0.07**	**0**	**-0**

Figure D.15: FuseNet (id-v) - U-Net (id-7)



ground truth \ prediction	pvmodule	dormer	window	ladder	chimney	unknown	background	accuracy on ground truth
pvmodule	-15127	582	-815	-21	390	45	11662	**-0.03**
dormer	1582	2350	-456	-89	-465	224	-12942	**0.07**
window	3578	1251	295	-90	1155	401	-13325	**0.04**
ladder	-36	116	-435	282	446	17	-1496	**0.03**
chimney	-72	-67	252	15	2234	151	-3307	**0.13**
unknown	-202	-48	-100	-248	796	1183	4369	**0.02**
background	2506	4415	5425	-34	7775	767	-4889	**-0**
prediction accuracy	**-0.02**	**-0.08**	**-0.05**	**0.05**	**-0.17**	**0.04**	**0**	**-0**

Figure D.16: FuseNet (id-vi) - U-Net (id-7)

E. Qualitative results, Wartenberg

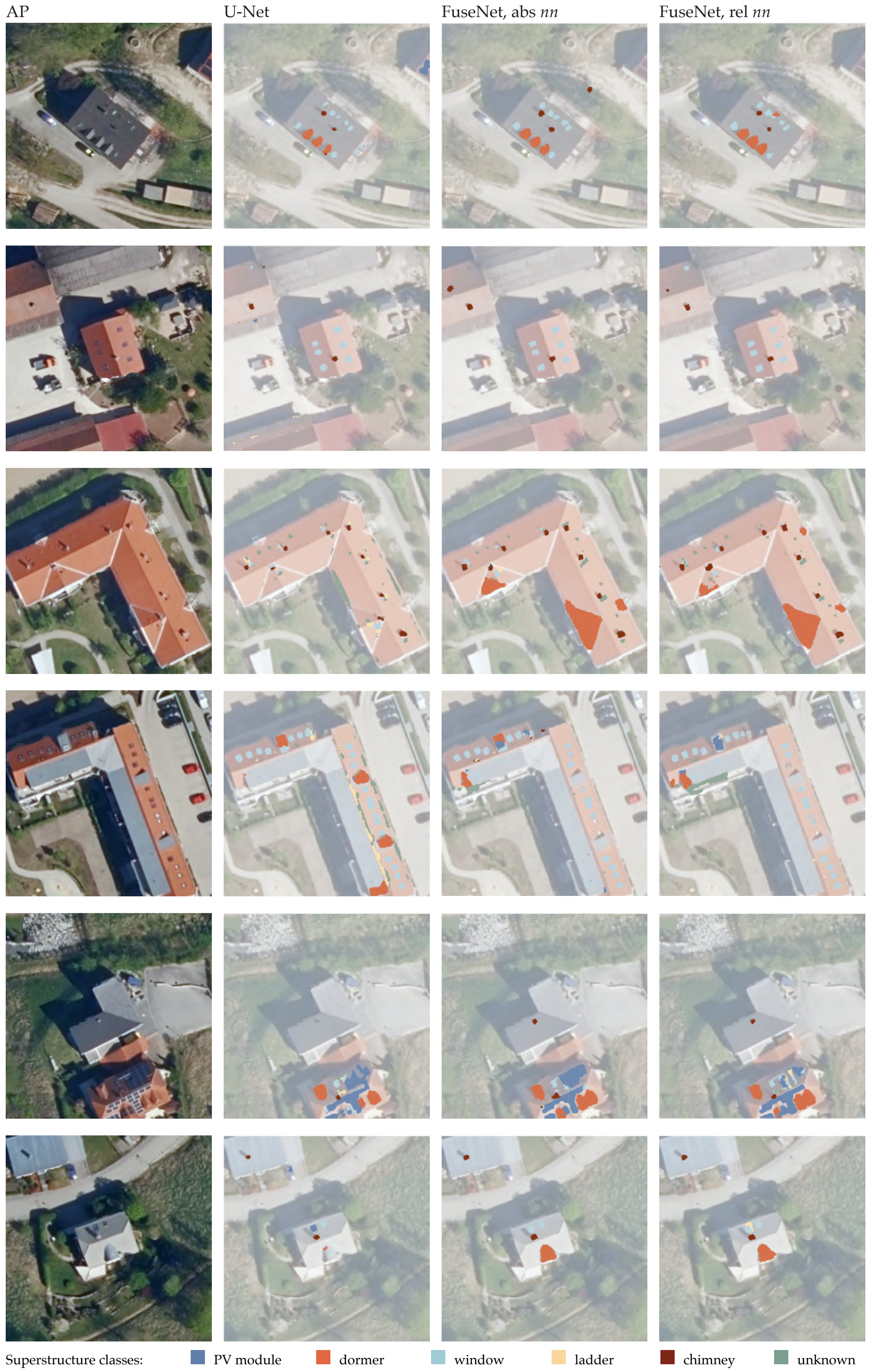


Figure E.1: U-Net (id-1) versus FuseNet (id-13c), samples 1/2

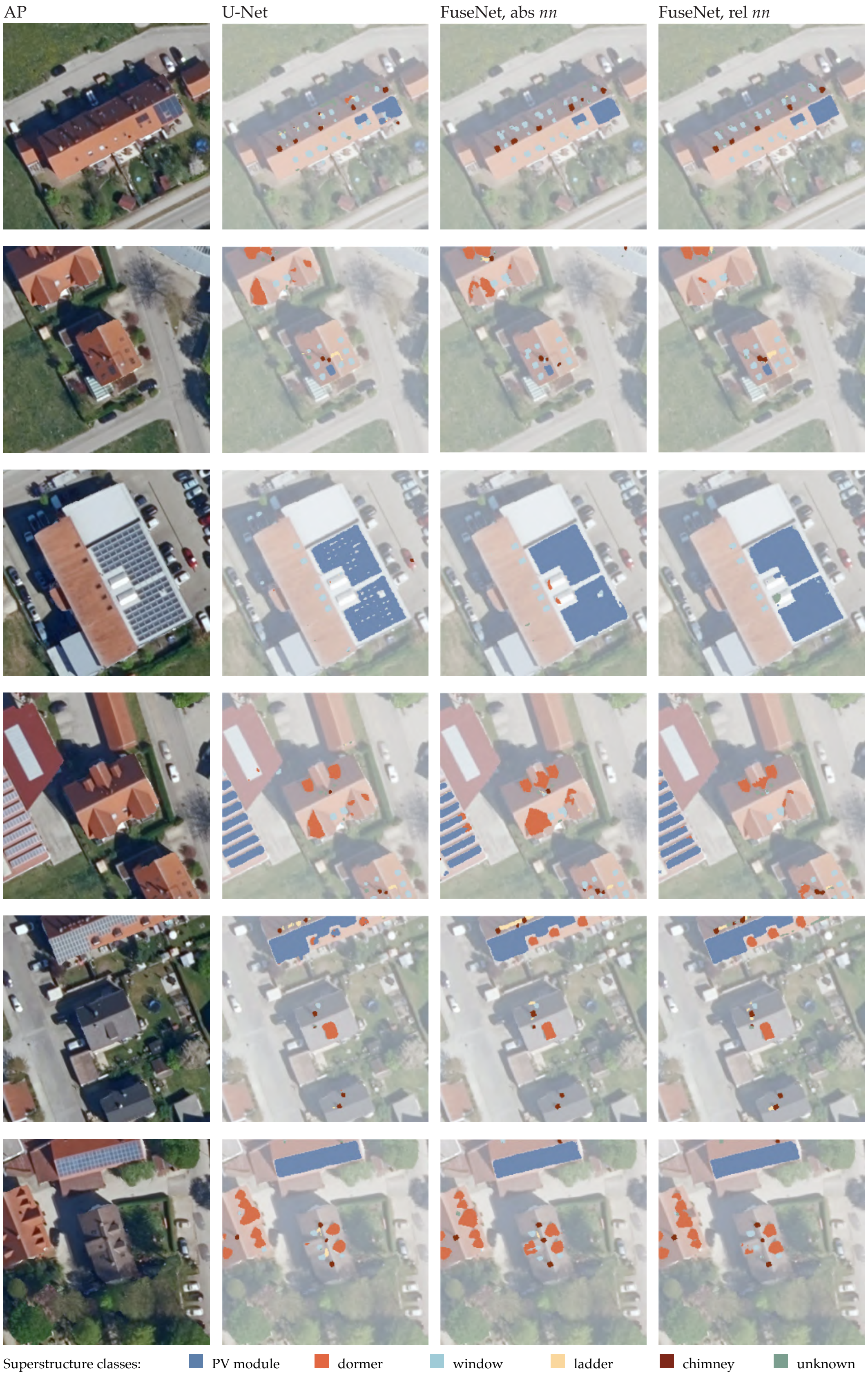


Figure E.2: U-Net (id-1) versus FuseNet (id-13c), samples 2/2

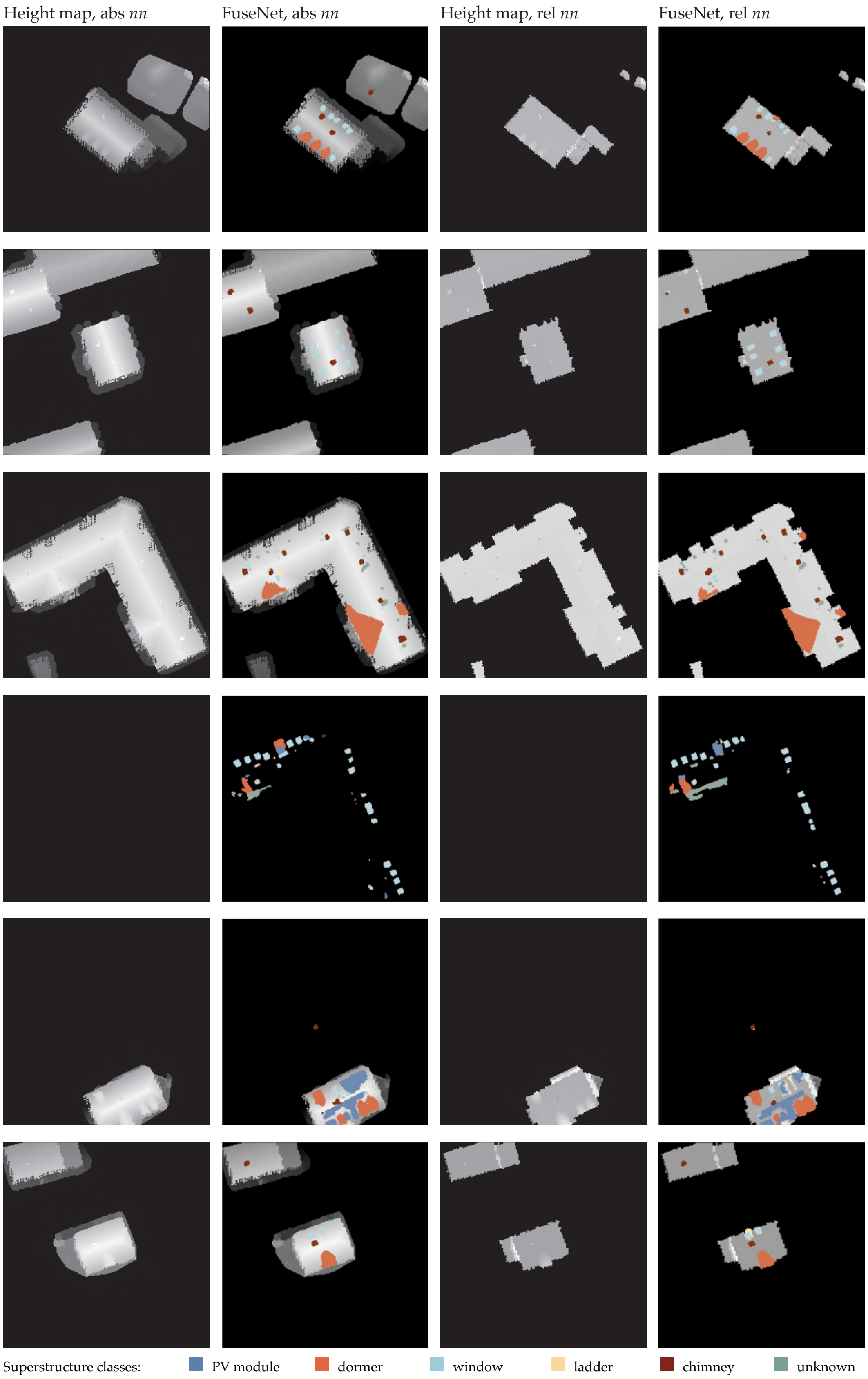


Figure E.3: Absolute versus relative height (FuseNet id-13c and v), samples 1/2

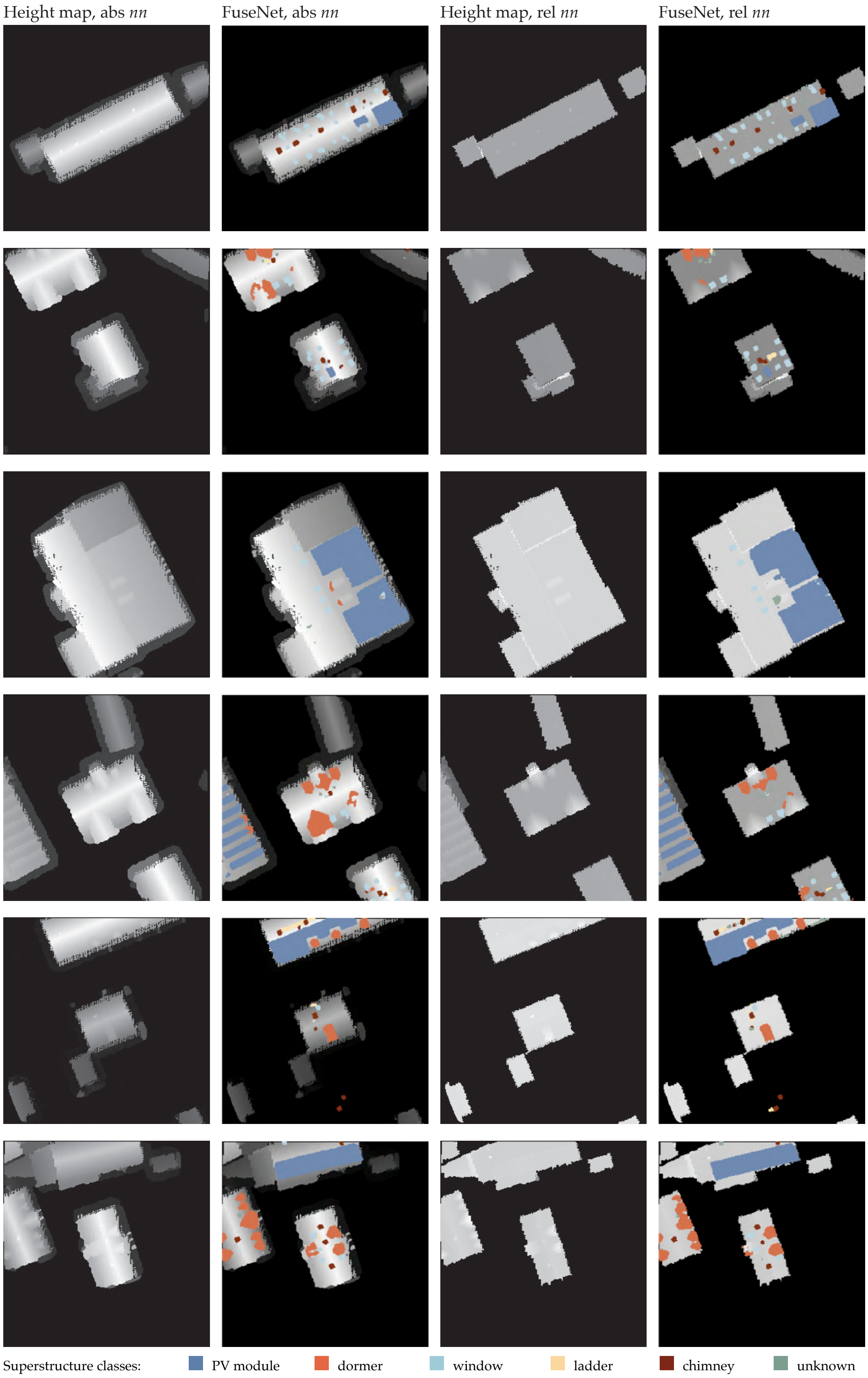


Figure E.4: Absolute versus relative height (FuseNet id-13c and v), samples 2/2

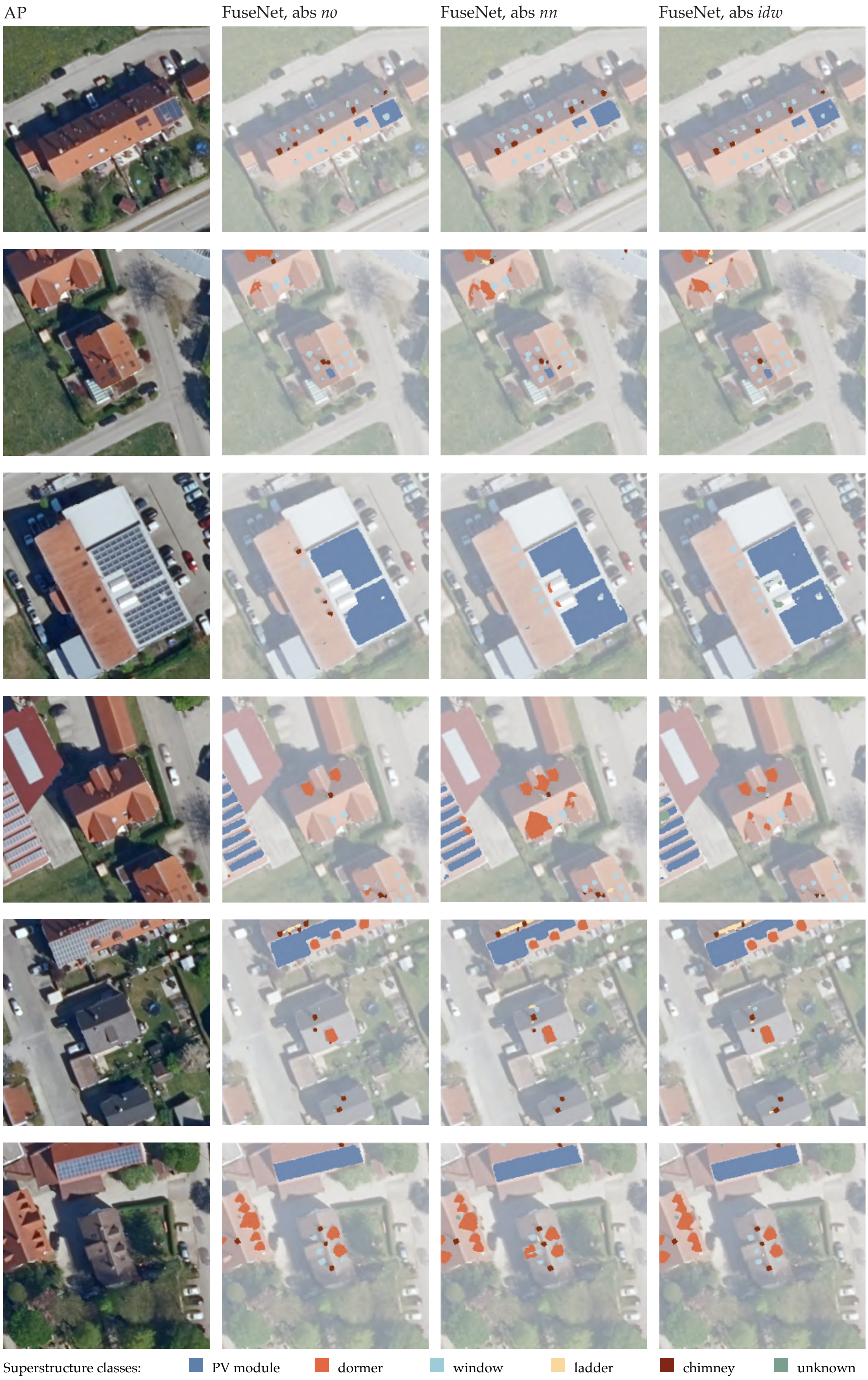


Figure E.5: *No* interpolation versus *nn* interpolation versus *idw* interpolation (FuseNet id-i, ii, and iii)

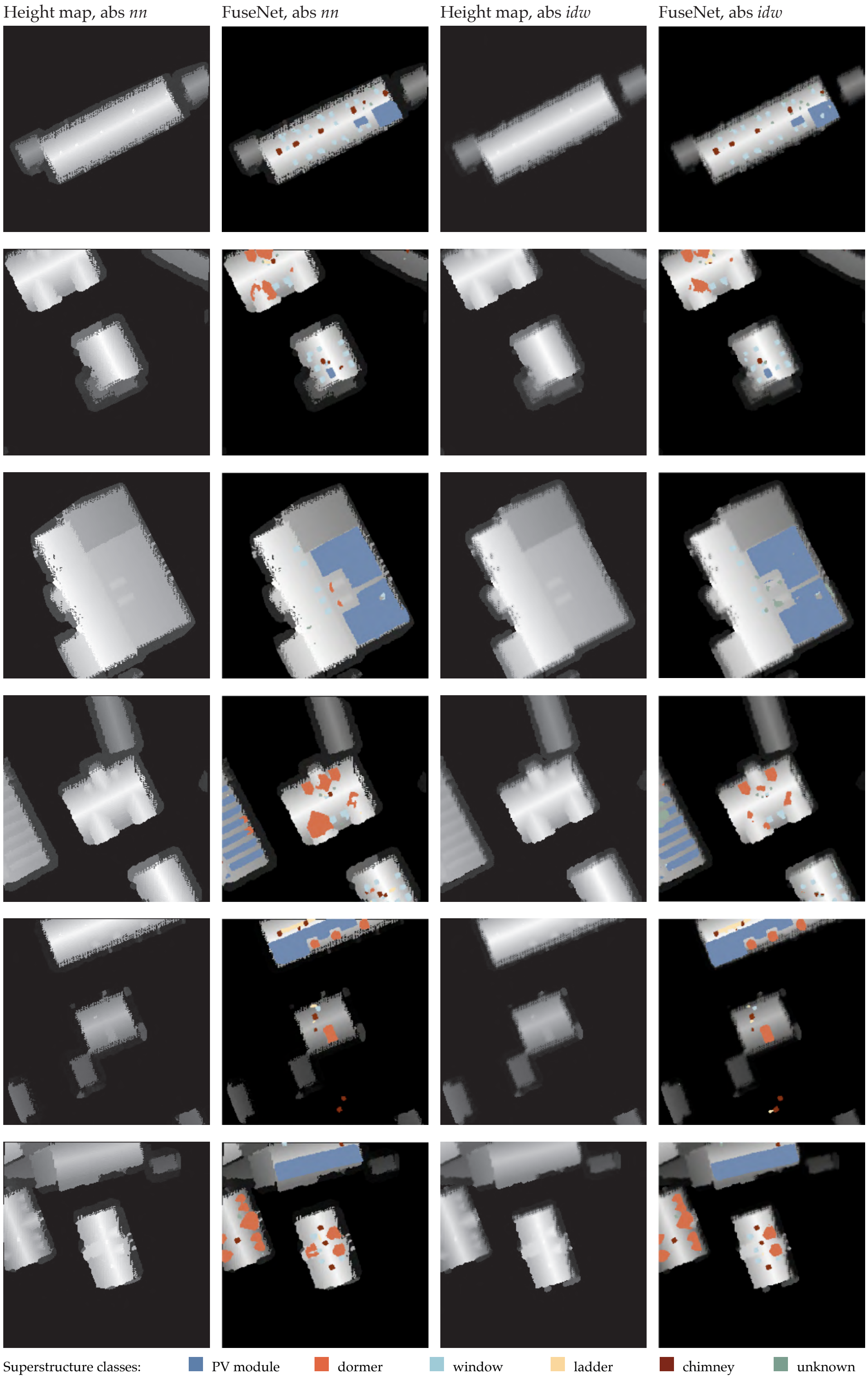


Figure E.6: *Nn* interpolation versus *idw* interpolation (FuseNet id-i and iii), samples 1/2

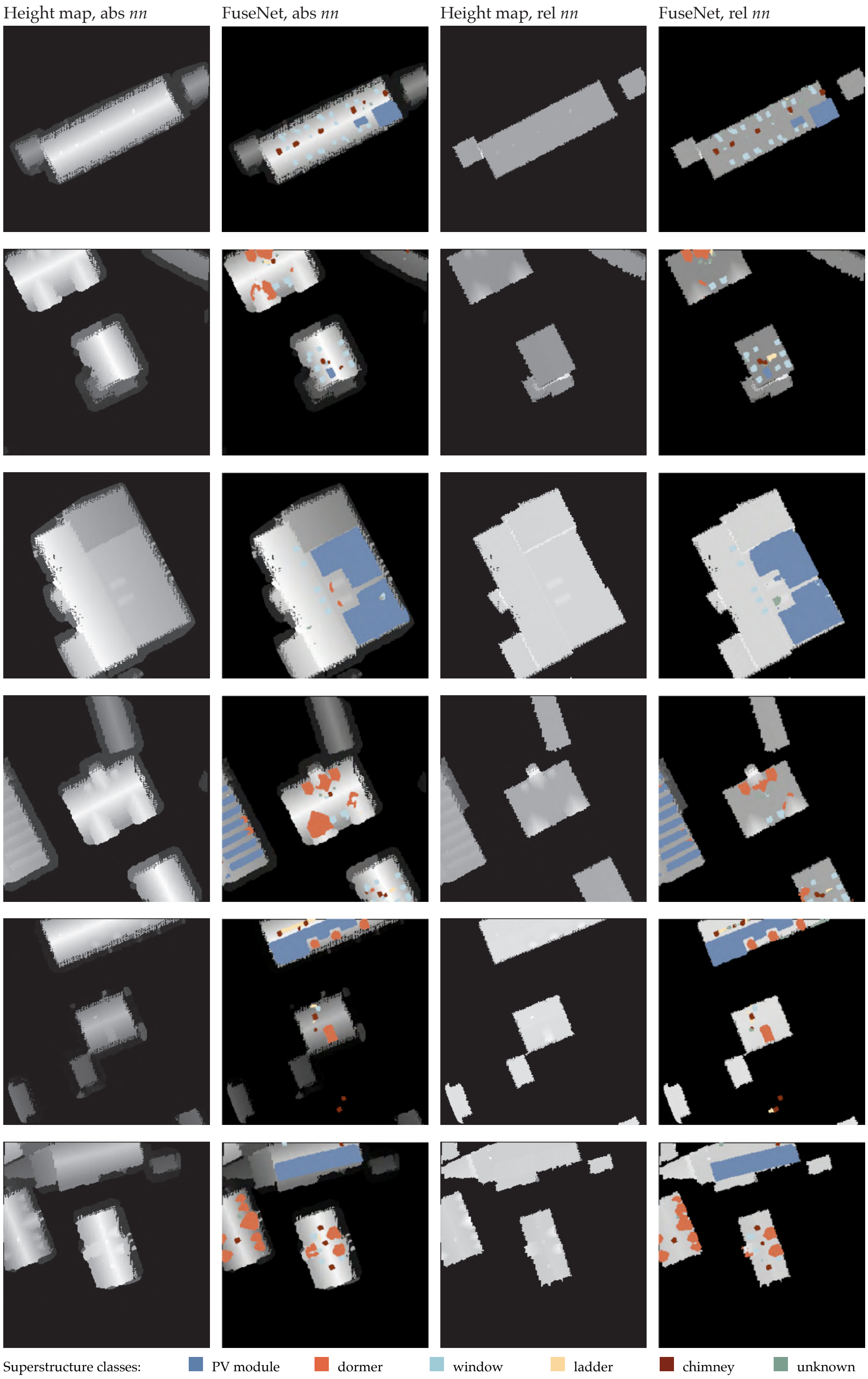


Figure E.7: *Nn* interpolation versus *idw* interpolation (FuseNet id-i and iii), samples 2/2

F. Qualitative results, Holten

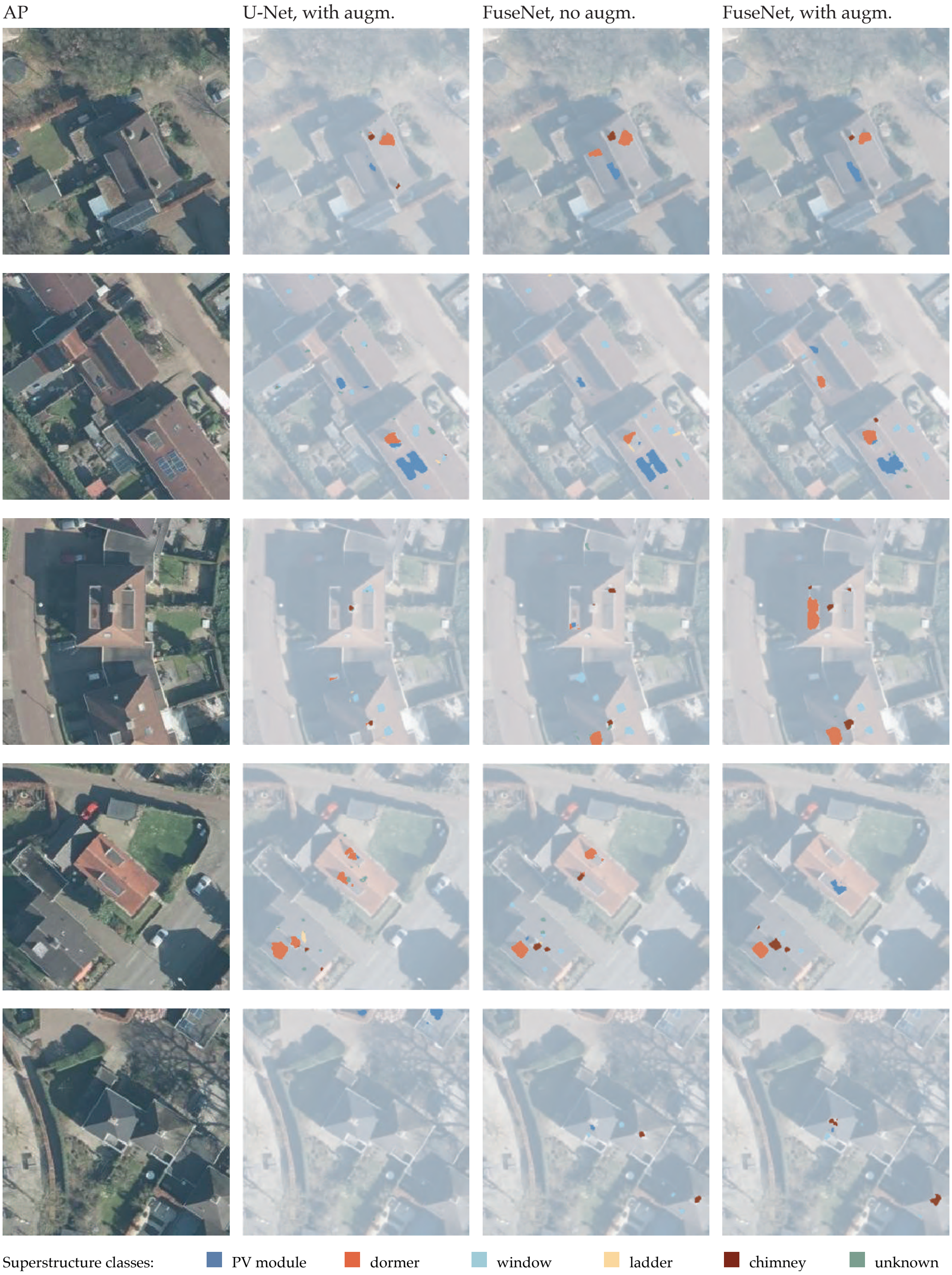


Figure F.1: Predictions on Holten dataset, predictions related to RGB images, samples 1/2

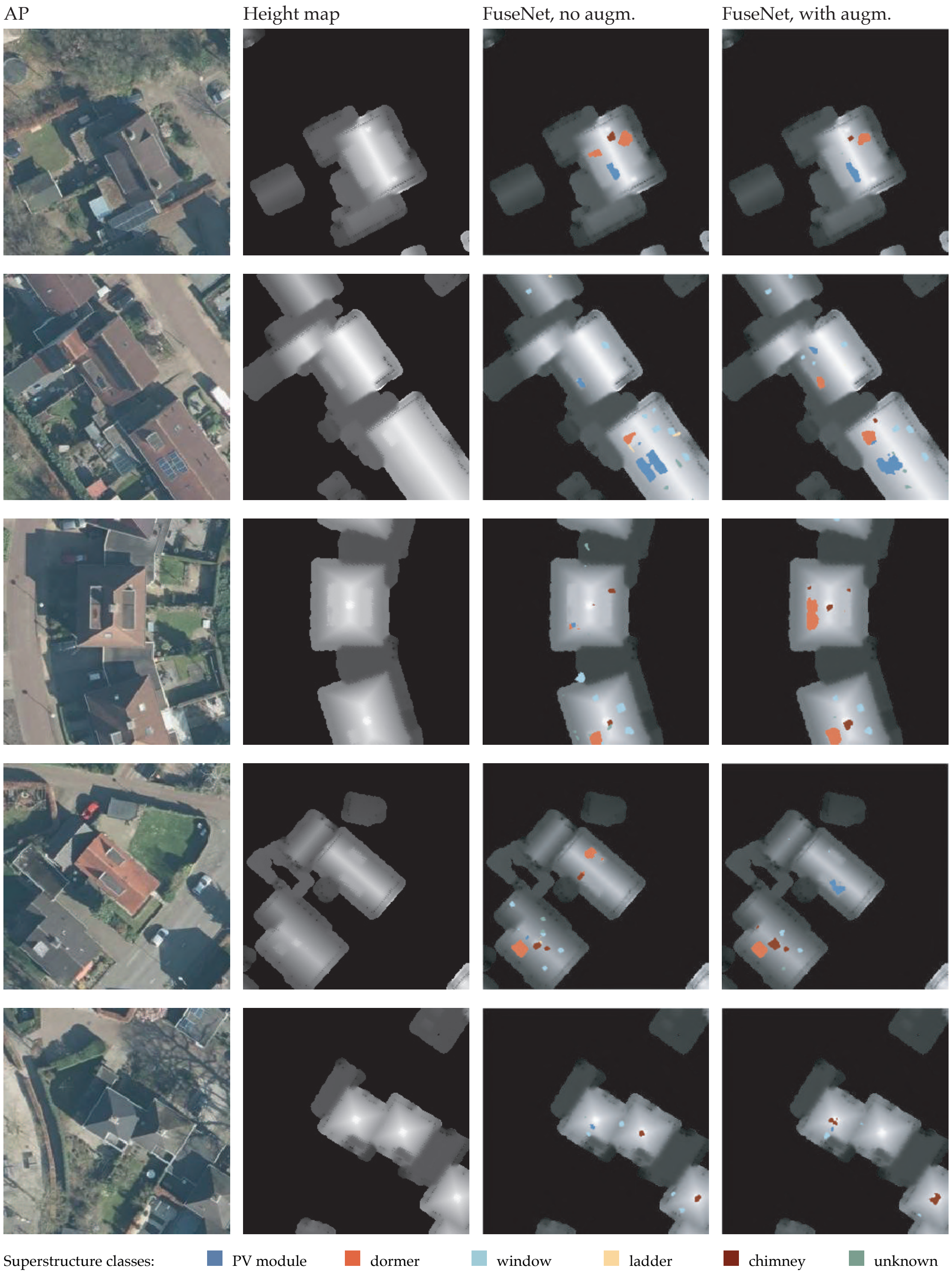


Figure F.2: Predictions on Holten dataset, predictions related to height images, samples 1/2

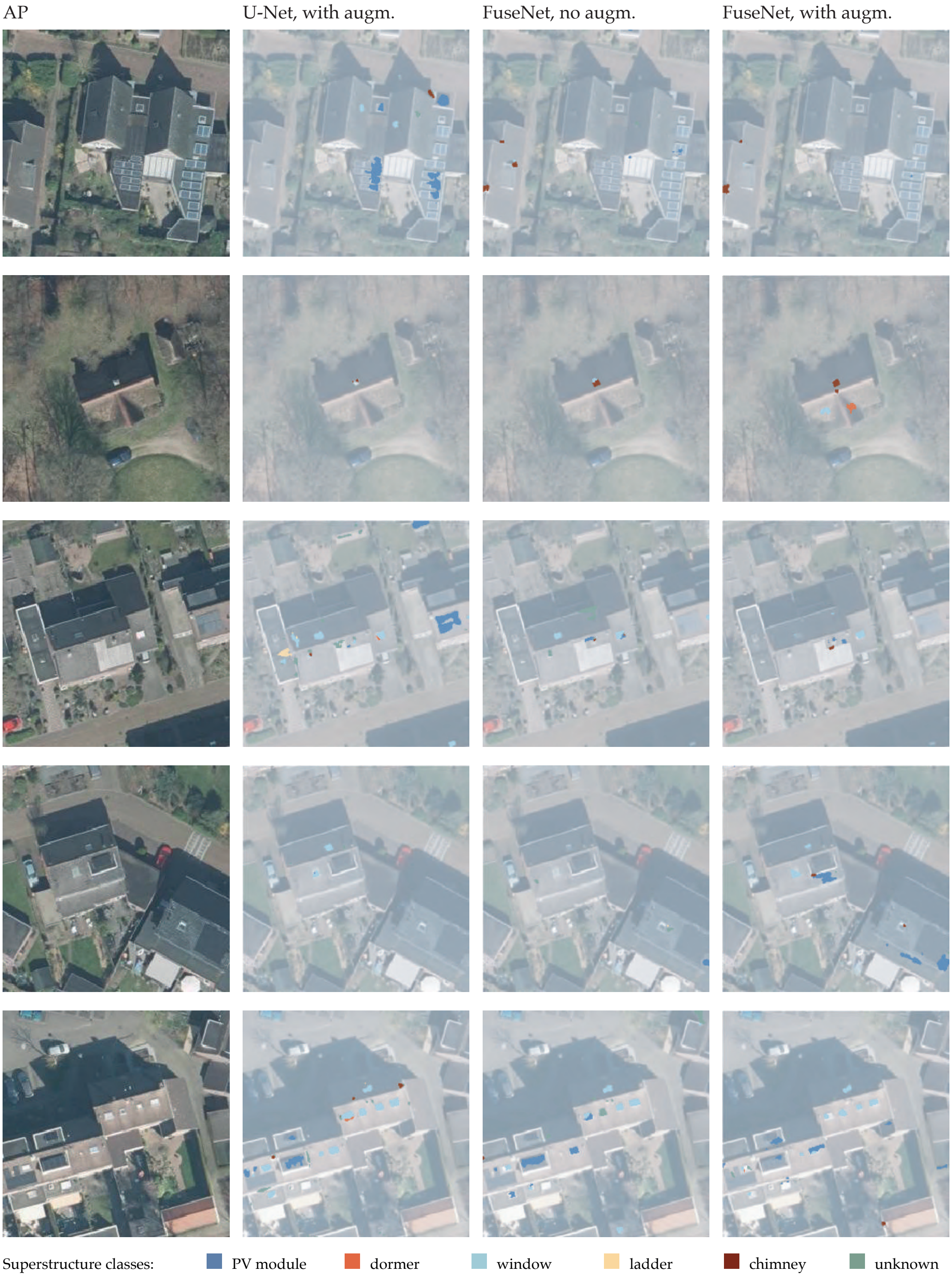


Figure F.3: Predictions on Holten dataset, predictions related to RGB images, samples 2/2

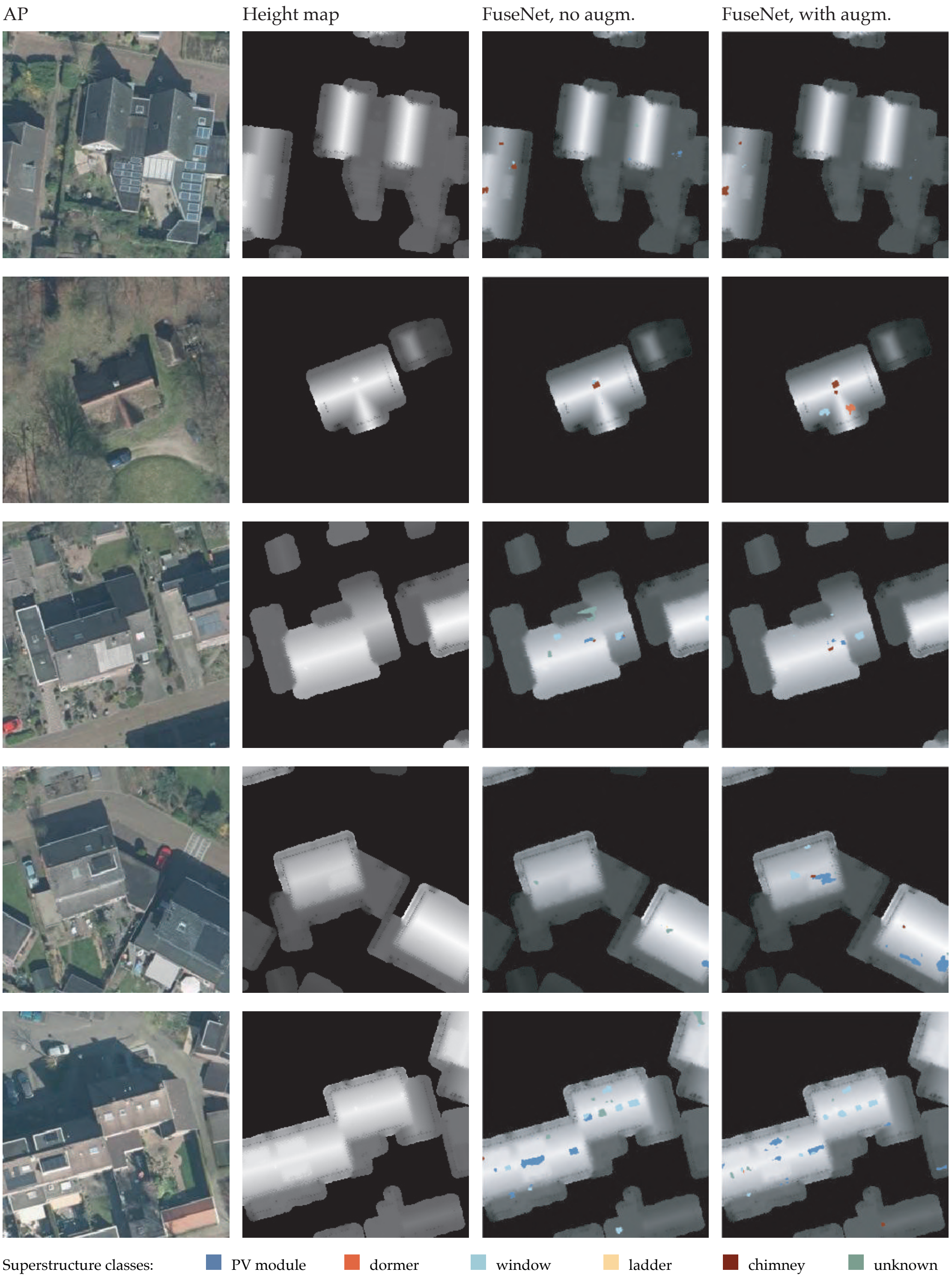


Figure F.4: Predictions on Holten dataset, predictions related to height images, samples 2/2

G. Qualitative results 3D, Holten

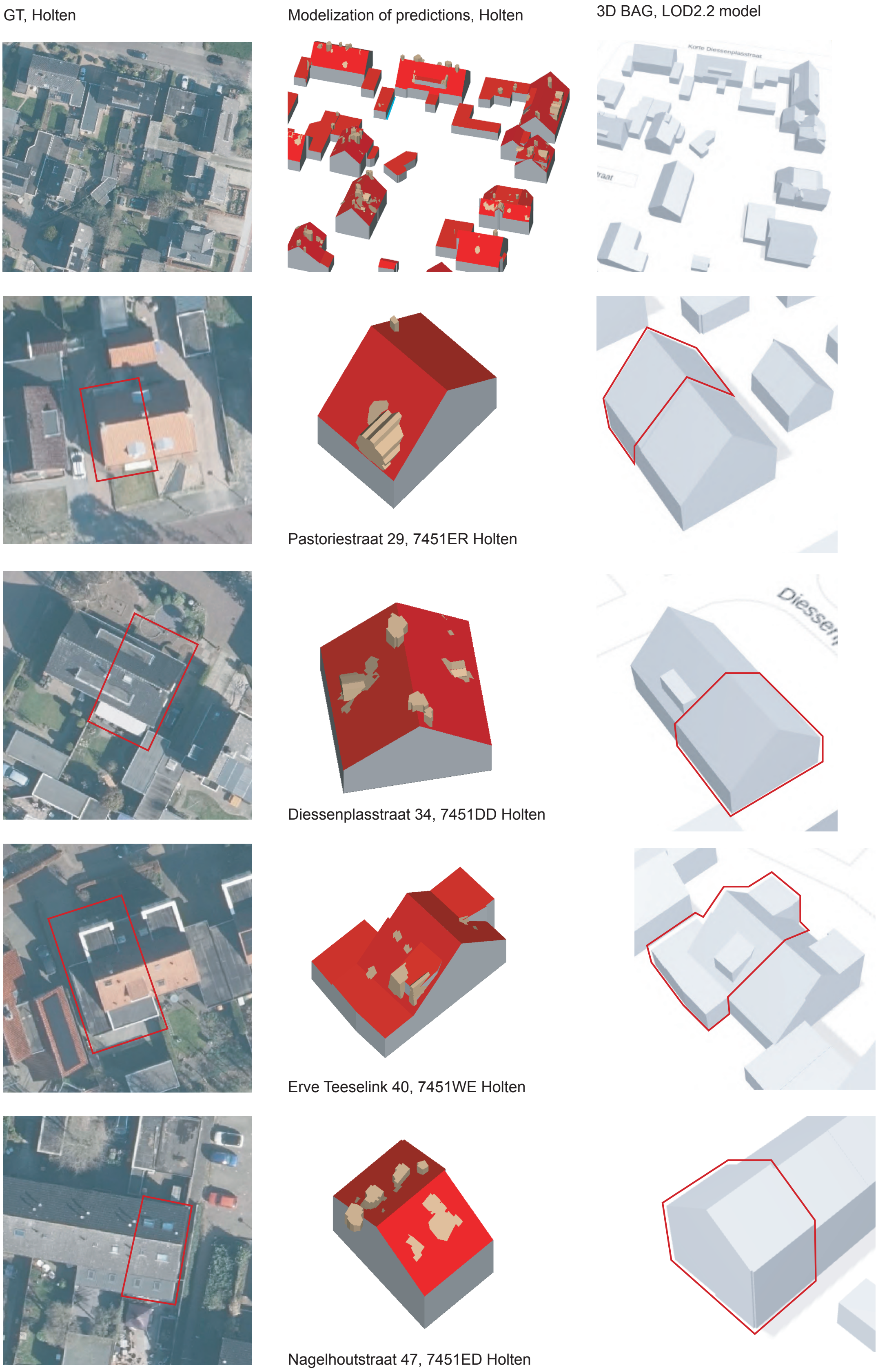

GT, Holten
Modelization of predictions, Holten
3D BAG, LOD2.2 model
Korte Diessenplasstraat
traat
Pastoriestraat 29, 7451ER Holten
Diessen
Diessenplasstraat 34, 7451DD Holten
Erve Teeselink 40, 7451WE Holten
Nagelhoutstraat 47, 7451ED Holten

Figure G.1: Visualization of Holten 3D model from predictions, samples 1/2

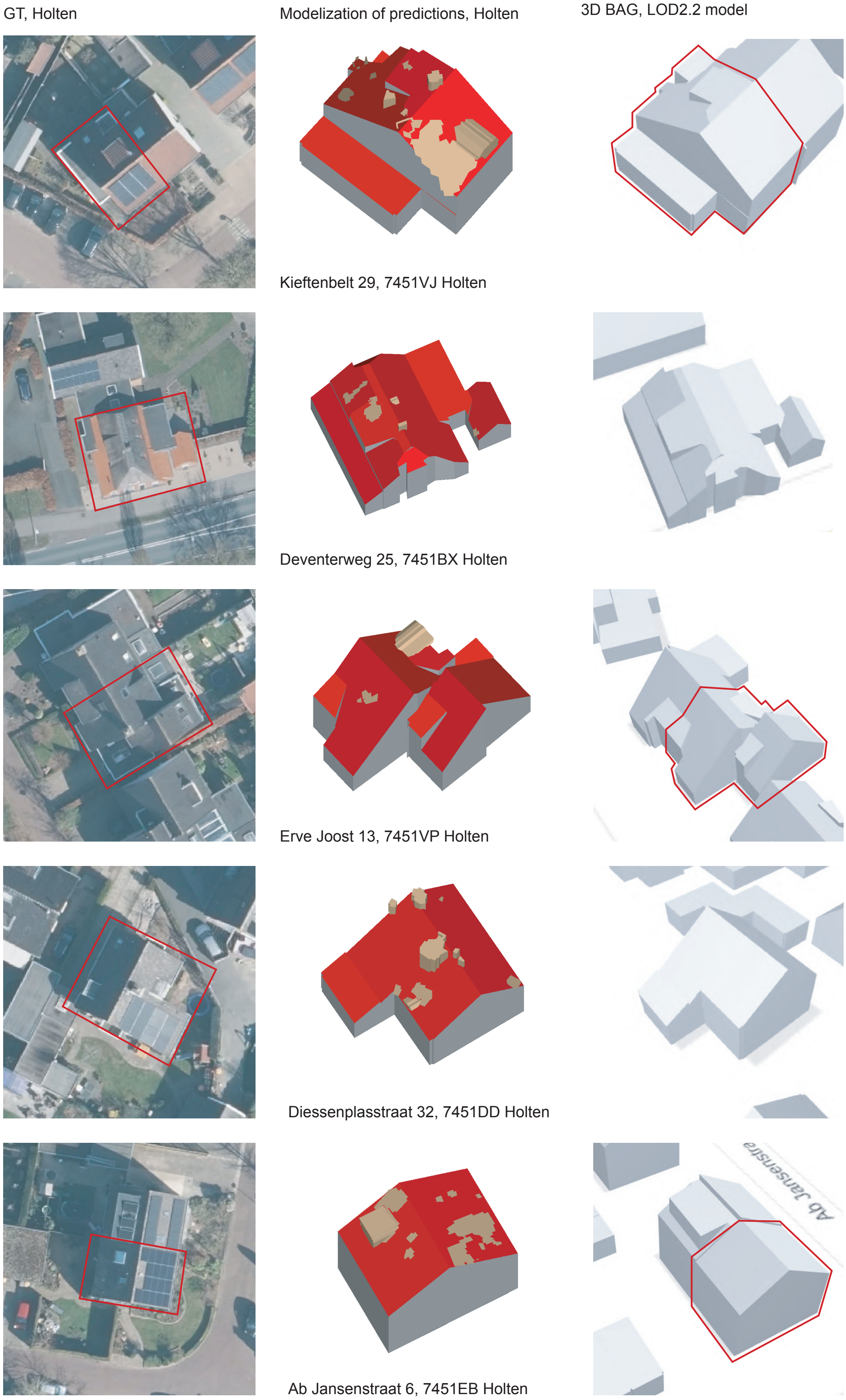

GT, Holten
Modelization of predictions, Holten
3D BAG, LOD2.2 model
Kieftenbelt 29, 7451VJ Holten
Deventerweg 25, 7451BX Holten
Erve Joost 13, 7451VP Holten
Diessenplasstraat 32, 7451DD Holten
Ab Jansenstraat 6, 7451EB Holten
Ab Jansenstra

Figure G.2: Visualization of Holten 3D model from predictions, samples 2/2

References

Ahmed, Aneeqa, and Yung-Cheol Byun. 2019. “Edge Detection using CNN for Roof Images.” In *Proceedings of the 2019 Asia Pacific Information Technology Conference,* 75–78. APIT 2019. New York, NY, USA: Association for Computing Machinery, January 25, 2019. ISBN: 9781450366212, accessed January 16, 2022. https://doi.org/10.1145/3314527.3314544. https://doi.org/10.1145/3314527.3314544.

AHN. 2022. “Actueel Hoogtebestand Nederland (AHN).” Accessed January 14, 2022. https://www.ahn.nl/.

Alidoost, F., et al. 2020. “Y-Shaped Convolutional Neural Network for 3d Roof Elements Extraction to Reconstruct Building Models from a Single Aerial Image.” *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* V-2-2020 (August 3, 2020): 321–328. ISSN: 2194-9050, accessed December 15, 2021. https://doi.org/10.5194/isprs-annals-V-2-2020-321-2020. https://www.isprs-ann-photogramm-remote-sens-spatial-inf-sci.net/V-2-2020/321/2020/.

Apra, Irène. 2022. “height-data-preparation.” Accessed June 28, 2022. https://github.com/iapra/height-data-preparation.git.

Arroyo Ohori, Ken, et al. 2018. “Processing BIM and GIS Models in Practice: Experiences and Recommendations from a GeoBIM Project in The Netherlands.” *ISPRS International Journal of Geo-Information* 7, no. 8 (August 2, 2018): 311. ISSN: 2220-9964, accessed June 10, 2022. https://doi.org/10.3390/ijgi7080311. http://www.mdpi.com/2220-9964/7/8/311.

Audebert, Nicolas, et al. 2018. “Beyond RGB: Very high resolution urban remote sensing with multimodal deep networks.” *ISPRS Journal of Photogrammetry and Remote Sensing,* Geospatial Computer Vision, 140 (June 1, 2018): 20–32. ISSN: 0924-2716, accessed December 10, 2021. https://doi.org/10.1016/j.isprsjprs.2017.11.011. https://www.sciencedirect.com/science/article/pii/S0924271617301818.

Biljecki, Filip, et al. 2014. “Formalisation of the level of detail in 3D city modelling.” *Computers, Environment and Urban Systems* 48 (November 1, 2014): 1–15. ISSN: 0198-9715, accessed December 20, 2021. https://doi.org/10.1016/j.compenvurbsys.2014.05.004. https://www.sciencedirect.com/science/article/pii/S0198971514000519.

Bressan, Patrik Olã, et al. 2021. “Semantic Segmentation with Labeling Uncertainty and Class Imbalance.” *arXiv:2102.04566 [cs]* (February 8, 2021). Accessed April 22, 2022. arXiv: 2102.04566. http://arxiv.org/abs/2102.04566.

Bruhse, Matthias. 2022. “Automatic generation of 3D Roof Superstructures for Semantic 3D City Models using Deep Learning.” Master Thesis, Technical University of Munich.

Chollet, François, et al. 2015. “Keras (github repository).” https://github.com/fchollet/keras.

Dean, Victoria. 2017. *Multimodal Deep Learning.* Accessed December 13, 2021. https://www.youtube.com/watch?v=6QewMQT4iMM.

Donkers, Sjors, et al. 2016. “Automatic conversion of IFC datasets to geometrically and semantically correct CityGML LOD3 buildings: Automatic conversion of IFC datasets to CityGML LOD3 buildings.” *Transactions in GIS* 20, no. 4 (August): 547–569. ISSN: 13611682, accessed June 10, 2022. https://doi.org/10.1111/tgis.12162. https://onlinelibrary.wiley.com/doi/10.1111/tgis.12162.

Duque-Arias, David, et al. 2021. “On Power Jaccard Losses for Semantic Segmentation:” in *Proceedings of the 16th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications,* 561–568. 16th International Conference on Computer Vision Theory and Applications. Online Streaming, — Select a Country —: SCITEPRESS - Science / Technology Publications. ISBN: 9789897584886, accessed April 19, 2022. https://doi.org/10.5220/0010304005610568. https://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0010304005610568.

Faltermeier, Florian. 2022. “Improving CNN roof segment detection by dataset extrusion using 3D city models.” Study project report in the program Environmental Engineering M.Sc., Technical University of Munich.

Gkioxari, Georgia, et al. 2020. “Mesh R-CNN.” *arXiv:1906.02739 [cs]* (January 25, 2020). Accessed December 12, 2021. arXiv: 1906.02739. http://arxiv.org/abs/1906.02739.

Gu, Yuhang, et al. 2021. “Top-Down Pyramid Fusion Network for High-Resolution Remote Sensing Semantic Segmentation.” *Remote Sensing* 13, no. 20 (January): 4159. Accessed December 14, 2021. https://doi.org/10.3390/rs13204159. https://www.mdpi.com/2072-4292/13/20/4159.

Guo, Yulan, et al. 2020. “Deep Learning for 3D Point Clouds: A Survey.” *arXiv:1912.12033 [cs, eess]* (June 23, 2020). Accessed January 17, 2022. arXiv: 1912.12033. http://arxiv.org/abs/1912.12033.

Gupta, Neha. 2013. “Artificial Neural Network.” *IISTE* 3 (1). Accessed December 4, 2021. https://core.ac.uk/reader/234686479.

Hazirbas, Caner, and Mehmet Aygun. 2018. “fusenet-pytorch (github repository).” Accessed April 15, 2022. https://github.com/tum-vision/fusenet.

Hazirbas, Caner, et al. 2017. “FuseNet: Incorporating Depth into Semantic Segmentation via Fusion-Based CNN Architecture.” In *Computer Vision – ACCV 2016,* edited by Shang-Hong Lai et al., 10111:213–228. Cham: Springer International Publishing. ISBN: 9783319541808 9783319541815, accessed December 10, 2021. https://doi.org/10.1007/978-3-319-54181-5_14. http://link.springer.com/10.1007/978-3-319-54181-5_14.

He, Kaiming, et al. 2015. “Deep Residual Learning for Image Recognition.” *arXiv:1512.03385 [cs]* (December 10, 2015). Accessed December 12, 2021. arXiv: 1512 . 03385. http://arxiv.org/abs/1512.03385.

IGN. 2022. “LiDAR HD: vers une nouvelle cartographie 3D du territoire.” Accessed January 14, 2022. https://www.ign.fr/institut/lidar-hd-vers-une-nouvelle-cartographie-3d-du-territoire.

INSPIRE. 2022. “INSPIRE Directive.” Accessed June 18, 2022. https://inspire.ec.europa.eu/inspire-directive/2.

Ioannou, Konstantinos, and Dimitrios Myronidis. 2021. “Automatic Detection of Photovoltaic Farms Using Satellite Imagery and Convolutional Neural Networks.” *Sustainability* 13, no. 9 (May 10, 2021): 5323. ISSN: 2071-1050, accessed January 16, 2022. https://doi.org/10.3390/su13095323. https://www.mdpi.com/2071-1050/13/9/5323.

Jaccard, Paul. 1912. “The distribution of the flora in the alpine zone 1.” *New Phytologist* 11, no. 2 (February): 37–50. ISSN: 0028-646X, 1469-8137, accessed May 7, 2022. https://doi.org/10.1111/j.1469-8137.1912.tb05611.x. https://onlinelibrary.wiley.com/doi/10.1111/j.1469-8137.1912.tb05611.x.

Kemmerzell, Nils. 2020. “Automatische Analyse des ökonomischen PV-Potenzials mittels GIS und Luftbildern - Automatic analysis of economic pv-potential via aerial images and GIS.” Master thesis.

Kraetzig, Nikita Marwaha. 2021. “A Definitive Guide to Buying and Using Satellite Imagery.” UP42 Official Website. Accessed January 16, 2022. https://up42.com/blog/tech/a-definitive-guide-to-buying-and-using-satellite-imagery.

Krapf, Sebastian, et al. 2021. “Towards Scalable Economic Photovoltaic Potential Analysis Using Aerial Images and Deep Learning.” *Energies* 14, no. 13 (June 24, 2021): 3800. ISSN: 1996-1073, accessed January 14, 2022. https://doi.org/10.3390/en14133800. https://www.mdpi.com/1996-1073/14/13/3800.

Krizhevsky, Alex, et al. 2012. “ImageNet Classification with Deep Convolutional Neural Networks.” In *Advances in Neural Information Processing Systems,* vol. 25. Curran Associates, Inc. Accessed December 13, 2021. https://papers.nips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html.

Kudinov, Dmitry. 2021a. “3D Buildings from Imagery with AI: Part 1.” GeoAI, October 28, 2021. Accessed December 12, 2021. https://medium.com/geoai/3d-buildings-from-imagery-with-ai-fbbc1852e4dd.

———. 2021b. “3D Buildings from Imagery with AI: Part 2.” GeoAI, October 28, 2021. Accessed December 12, 2021. https://medium.com/geoai/3d-buildings-from-imagery-with-ai-part-2-ef129dca6dc.

LDBV. 2016. *Laserdaten Beschreibung der Punktklassen (Laserdata Description of the point classes).* Accessed December 22, 2021. https://www.ldbv.bayern.de/file/pdf/10574/Laserdaten%20-%20Beschreibung%20der%20Punktklassen.pdf.

LDBV. 2021a. “ALKIS - Geodaten online.” Accessed December 22, 2021. https://geodatenonline.bayern.de/geodatenonline/seiten/dfkalkis_info.

———. 2021b. “BayernAtlas.” https://geoportal.bayern.de/bayernatlas/?topic=ba&lang=de&bgLayer=atkis&w=100%25&h=500px&catalogNodes=11&layers=luftbild,KML%7C%7Chttps:%2F%2Fwww.geodaten.bayern.de%2Fdownload%2Fuebersicht_DGM%2FLaserscanningbefliegungen.kml%7C%7Ctrue,luftbild_parz,tk_by&E=723070.17&N=5417849.58&zoom=3.5166666666666435&layers_visibility=false,true,true,false.

———. 2021c. “Landesamt für Digitalisierung, Breitband und Vermessung.” Accessed December 21, 2021. https://www.ldbv.bayern.de/.

Le Cun, Yann, et al. 1998. “GradientBased Learning Applied to Document Recognition.” Accessed December 27, 2021. http://vision.stanford.edu/cs598_spring07/papers/Lecun98.pdf.

Lee, Stephen, et al. 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,* 2105–2113. KDD ’19: The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. Anchorage AK USA: ACM, July 25, 2019. ISBN: 9781450362016, accessed December 20, 2021. https://doi.org/10.1145/3292500.3330741. https://dl.acm.org/doi/10.1145/3292500.3330741.

Lin, Min, et al. 2014. “Network In Network.” *arXiv:1312.4400 [cs]* (March 4, 2014). Accessed December 27, 2021. arXiv: 1312.4400. http://arxiv.org/abs/1312.4400.

Long, Jonathan, et al. 2015. “Fully Convolutional Networks for Semantic Segmentation.” *arXiv:1411.4038 [cs]* (March 8, 2015). Accessed December 12, 2021. arXiv: 1411.4038. http://arxiv.org/abs/1411.4038.

Marcello, Javier, and Francisco Eugenio. 2019. *Very High Resolution (VHR) Satellite Imagery: Processing and Applications.* OCLC: 1163815192. ISBN: 9783039217571, accessed January 16, 2022. https://openresearchlibrary.org/content/db813f58-5522-443e-a178-36eb0d48671c.

Maucher, Johannes. 2013. “Information Theory — Multimedia Codec Excercises 1.0 documentation.” Accessed April 20, 2022. https://www.hdm-stuttgart.de/~maucher/Python/MMCodecs/html/basicFunctions.html.

Minaee, Shervin, et al. 2020. “Image Segmentation Using Deep Learning: A Survey.” *arXiv:2001.05566 [cs]* (November 14, 2020). Accessed December 8, 2021. arXiv: 2001.05566. http://arxiv.org/abs/2001.05566.

Mulder, Amber. 2020. “Semantic Segmentation of RGB-Z Aerial Imagery Using Convolutional Neural Networks.” Master thesis. Accessed December 3, 2021. https://repository.tudelft.nl/islandora/object/uuid%3Ab936953b-4c73-4ce1-a897-7da4287ff79a.

Ng, Andrew. 2021. “Neural Networks & Deep Learning (Coursera).” Course. Accessed December 4, 2021. https://www.coursera.org/specializations/deep-learning.

Nimbalkar, Prakash, et al. 2018. “Optimal Band Configuration for the Roof Surface Characterization Using Hyperspectral and LiDAR Imaging.” *Journal of Spectroscopy* 2018 (April 18, 2018): e6460518. ISSN: 2314-4920, accessed December 14, 2021. https://doi.org/10.1155/2018/6460518. https://www.hindawi.com/journals/jspec/2018/6460518/.

Niu, Jiqiang, et al. 2016. “Global Research on Artificial Intelligence from 1990–2014: Spatially-Explicit Bibliometric Analysis.” *ISPRS International Journal of Geo-Information* 5, no. 5 (May): 66. Accessed December 4, 2021. https://doi.org/10.3390/ijgi5050066. https://www.mdpi.com/2220-9964/5/5/66.

Nüst, Daniel, et al. 2018. “Reproducible research and GIScience: an evaluation using AGILE conference papers.” *PeerJ* 6 (July 13, 2018): e5072. ISSN: 2167-8359, accessed June 18, 2022. https://doi.org/10.7717/peerj.5072. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6047504/.

OGC. 2012. “OGC City Geography Markup Language (CityGML) En-coding Standard.” Accessed January 17, 2022. https://portal.ogc.org/files/?artifact_id=47842.

PDAL Contributors. 2020. “PDAL Point Data Abstraction Library,” August 28, 2020. Accessed April 16, 2022. https://doi.org/10.5281/ZENODO.2556737. https://zenodo.org/record/2556737.

PDOK. 2021. “PDOK datasets: Luchtfotos.” Accessed December 4, 2021. https://www.pdok.nl/introductie/-/article/luchtfoto-pdok.

Peters, Ravi, et al. 2021. “Automated 3D reconstruction of LoD2 and LoD1 models for all 10 million buildings of the Netherlands.” *arXiv:2201.01191 [cs, eess]* (December 30, 2021). Accessed January 14, 2022. arXiv: 2201.01191. http://arxiv.org/abs/2201.01191.

PostGIS. 2022. “About PostGIS.” PostGIS. Accessed April 16, 2022. https://postgis.net/.

Prummer, Johanna. 2021. “Untersuchung eines datenzentrierten Ansatzes zur Dachaufbauerkennung mittels Deep Learning - Assessment of a Data-Centric Approach for Roof Superstructure Detection using Deep Learning.” Master Thesis, Technical University of Munich.

Ronneberger, Olaf, et al. 2015. “U-Net: Convolutional Networks for Biomedical Image Segmentation.” Version: 1, *arXiv:1505.04597 [cs]* (May 18, 2015). Accessed December 3, 2021. arXiv: 1505.04597. http://arxiv.org/abs/1505.04597.

Rouault, Even, et al. 2022. “GDAL,” March 14, 2022. Accessed April 16, 2022. https://doi.org/10.5281/ZENODO.6352176. https://zenodo.org/record/6352176.

Roy, Swalpa Kumar, et al. 2020. “FuSENet: fused squeeze-and-excitation network for spectral-spatial hyperspectral image classification.” *IET Image Processing* 14 (8): 1653–1661. ISSN: 1751-9667, accessed December 12, 2021. https://doi.org/10.1049/iet-ipr.2019.1462. https://onlinelibrary.wiley.com/doi/abs/10.1049/iet-ipr.2019.1462.

Shi, Zhong-Zhi, and Nan-Ning Zheng. 2006. “Progress and Challenge of Artificial Intelligence.” *Journal of Computer Science and Technology* 21, no. 5 (September 1, 2006): 810. ISSN: 1860-4749, accessed December 4, 2021. https://doi.org/10.1007/s11390-006-0810-5. https://doi.org/10.1007/s11390-006-0810-5.

Simonyan, Karen, and Andrew Zisserman. 2015. “Very Deep Convolutional Networks for Large-Scale Image Recognition.” *arXiv:1409.1556 [cs]* (April 10, 2015). Accessed December 12, 2021. arXiv: 1409.1556. http://arxiv.org/abs/1409.1556.

Sørensen, Thorvald. 1948. “A method of establishing groups of equal amplitude in plant sociology based on similarity of species and its application to analyses of the vegetation on Danish commons.” 5 (4): 1–34.

Syed, Khawaja Haseeb Uddin. 2020. “A Deep Learning Approach to Roof Superstructure detection for PV potential estimation.” Master thesis, Technical University of Munich, November 11, 2020.

Szegedy, Christian, et al. 2014. “Going Deeper with Convolutions.” *arXiv:1409.4842 [cs]* (September 16, 2014). Accessed December 15, 2021. arXiv: 1409.4842. http://arxiv.org/abs/1409.4842.

tudelft3d. 2021. “3D BAG.” Accessed December 8, 2021. https://3dbag.nl/en/viewer.

Wei, Shiqing, et al. 2020. “Toward Automatic Building Footprint Delineation From Aerial Images Using CNN and Regularization.” *IEEE Transactions on Geoscience and Remote Sensing* 58, no. 3 (March): 2178–2189. ISSN: 0196-2892, 1558-0644, accessed January 16, 2022. https://doi.org/10.1109/TGRS.2019.2954461. https://ieeexplore.ieee.org/document/8933116/.

Willenborg, B., et al. 2018. “Integration of semantic 3D City models and 3D mesh models for accuracy improvements of solar potential analyses.” *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XLII-4/W10 (September 12, 2018): 223–230. ISSN: 2194-9034, accessed June 10, 2022. https://doi.org/10.5194/isprs-archives-XLII-4-W10-223-2018. https://www.int-arch-photogramm-remote-sens-spatial-inf-sci.net/XLII-4-W10/223/2018/.

Willenborg, Bruno. 2022. “3DCityDB Web Map client Demo Wartenberg.” Accessed June 13, 2022. https://rb.gy/q6qkhl.

Yakubovskiy, Pavel. 2019. “Segmentation Models.” Accessed April 19, 2022. https://github.com/qubvel/segmentation_models.

Yan, Li, et al. 2022. “Efficient Depth Fusion Transformer for Aerial Image Semantic Segmentation.” *Remote Sensing* 14, no. 5 (March 7, 2022): 1294. ISSN: 2072-4292, accessed March 27, 2022. https://doi.org/10.3390/rs14051294. https://www.mdpi.com/2072-4292/14/5/1294.

Yao, Zhihang, et al. 2018. “3DCityDB - a 3D geodatabase solution for the management, analysis, and visualization of semantic 3D city models based on CityGML.” *Open Geospatial Data, Software and Standards* 3, no. 1 (May 14, 2018): 5. ISSN: 2363-7501, accessed April 16, 2022. https://doi.org/10.1186/s40965-018-0046-7. https://doi.org/10.1186/s40965-018-0046-7.

Yuan, Xiaohui, et al. 2021. "A review of deep learning methods for semantic segmentation of remote sensing imagery." *Expert Systems with Applications* 169 (May 1, 2021): 114417. ISSN: 0957-4174, accessed December 10, 2021. https://doi.org/10.1016/j.eswa.2020.114417. https://www.sciencedirect.com/science/article/pii/S0957417420310836.

Zhou, Keqi, et al. 2019. "CNN-Based Land Cover Classification Combining Stratified Segmentation and Fusion of Point Cloud and Very High-Spatial Resolution Remote Sensing Image Data." *Remote Sensing* 11, no. 17 (January): 2065. Accessed December 12, 2021. https://doi.org/10.3390/rs11172065. https://www.mdpi.com/2072-4292/11/17/2065.

Colophon

This document was typeset using LATEX. The main font is Palatino. The concept figures were drawn using Adobe® Illustrator.