Hochschule Karlsruhe
University of
Applied Sciences

HKA

Master Thesis

Deep Learning for Photovoltaic Potential - Evaluation of an Alternative Approach for Roof Segmentation and Orientation Determination

Deepak Parajuli

2022

Deep Learning for Photovoltaic Potential - Evaluation of an Alternative Approach for Roof Segmentation and Orientation Determination

©2022 by Deepak Parajuli • bindaaspratiks@gmail.com

Submitted in partial fulfillment of the requirements for the M.Sc. degree in Geomatics

October 20, 2022

Author: Deepak Parajuli
Study Course: Geomatics M.Sc.
Supervisors: Prof. Dr. Christine Preisach (HKA), M.Sc. Sebastian Krapf (TUM)
Reviewer: M.Sc. Sebastian Krapf
Submission Date: 20.10.2022

Declaration

I hereby declare that the thesis entitled "Deep Learning for Photovoltaic Potential - Evaluation of an Alternative Approach for Roof Segmentation and Orientation Determination" submitted to the Faculty of Information Management and Media, Karlsruhe University of Applied Sciences, Karlsruhe, is my original piece of work done under the supervision of Prof. Dr. Christine Preisach and M.Sc. Sebastian Krapf. Works contributed by others were stated clearly with references in the thesis.

Karlsruhe, 20.10.2022 Deepak Parajuli

Acknowledgments

This Master's thesis project was a great learning experience for me. Working on the project provided me with a handful of experience with Deep Learning. I want to thank my supervisor, Sebastian Krapf, who gave valuable guidance and support throughout the process. His Deep Learning and Machine Learning expertise further supported and motivated me to improve my work. I could not have asked for a better mentor than him. I would also like to thank Professor Dr. Christine Preisach for her constant feedback, support, and ideas.

I would like to extend my gratitude to HKA and TUM for allowing me to access the required resources during the thesis work.

Last but not least, I cannot thank my parents and beloved wife enough for their immense support and motivation in every step of this journey.

Certificate of Approval

This is to certify that the project entitled "Deep Learning for Photovoltaic Potential - Evaluation of an Alternative Approach for Roof Segmentation and Orientation Determination" submitted by Mr. Deepak Parajuli, in partial fulfillment of the requirement for the Master's degree in Geomatics, is his original work. No part of this work has been submitted to any other Institute or University. This is to acknowledge that the accompanying thesis by Mr. Deepak Parajuli has been accepted for evaluation leading to the award of International Master's in Geomatics at Karlsruhe University of Applied Sciences.

Prof. Dr. Christine Preisach

Supervisor,

Professor of Data Science and Informatics,

Faculty of IMM, HKA

M.Sc. Sebastian Krapf

Supervisor,

Chair of Automotive Technology, TUM

ABSTRACT

The importance of renewable energy has increased over the years due to the scarcity of non-renewable resources. Instead of using non-renewable resources, we should seek other energy sources to cut carbon emissions, which can help mitigate global warming and combat climate change. Solar energy is more prevalent among other renewable resources as house owners can easily install solar panels on rooftops. However, one needs to hire a solar consultant, and everything must be done manually. Different new technologies are being used to figure out how to estimate solar potential reducing manual work. LiDAR technology is one of the most prominent among them. However, obtaining LiDAR data for every location is not practicable, and obtaining LiDAR data would be costly due to the need to use drones and aircraft and complex pre-processing stages. This study develops a new deep learning approach to find the roof segment and access their orientation for assessing the photovoltaic (PV) potential using widely available satellite images. The primary goal of this research is to develop a workflow for determining the roof segment and its orientation, and the long-term goal is to estimate PV potential globally. The learning technique determines the best places for solar panel installation in terms of the roof's orientation. The evaluation of the predicted orientation shows an orientation classification accuracy of 58.55% and a Mean Orientation Error (MOE) of ±74.82°. This research might assist the local government in determining the real Photovoltaic potential and planning accordingly.

Table of Contents

List of Abbreviations

3D	3-Dimensional
ANN	Artificial Neural Network
API	Application Programming Interface
AI	Artificial Intelligence
BCE	Binary Cross Entropy
CCE	Categorical Cross Entropy
CF	Categorical Focal
CNN	Convolutional Neural Network
CityGML	City Geography Markup Language
CO2	Carbon dioxide
DCNN	Deep Convolutional Neural Network
DL	Dice Loss
FL	Focal Loss
FN	False Negative
FP	False Positive
FPN	Feature Pyramid Network
GPU	Graphics Processing Unit
GIS	Geographic Information System
GW	Gigawatt
IoU	Intersection over Union
JL	Jaccard Loss
LiDAR	Light Detection and Ranging
MOE	Mean Orientation Error
NN	Neural Network
OSM	Open Street Map
PV	Photovoltaic
ReLU	Rectified Linear Unit
ResNet	Residual Net
RGB	Red Blue Green

RID	Roof Information Dataset
OpenCV	Open-Source Computer Vision
TN	True Negative
TP	True Positive
TUM	Technical University of Munich
TWh	Terawatt hour
VGG	Visual Geometry Group

1 Introduction

1.1 Motivation

The use of fossil fuels alone contributed to 36.4 billion tonnes of CO2 emissions into the atmosphere globally in 2021 [1], making it the sixth hottest year at 0.84°C warmer than average (1880-2021) [2]. The ecosystem has been deteriorated due to the excessive extraction and use of fossil fuels, and many environmental concerns, such as global warming, have arisen. Similarly, fossil fuel shortages will soon result in an energy catastrophe [3]. Solar panel installation is one approach to saving fossil fuels and the environment. The public is now self-encouraged to put solar panels on rooftops. The reason is the increase in the photovoltaic (PV) performance and the decrease in the installation cost. As a result, more than 848 gigawatts of solar capacity have already been installed worldwide in 2021 [4]. The installed PV panel produces a gross of 1032.5 TWh of energy. According to Germany's federal ministry of economic affairs and climate action, 59.8 gigawatts of solar capacity have been installed by the end of 2021, generating 48.45 TWh of energy which is 9.9% of the total energy used for electricity generation. Due to the ongoing energy crisis, the fact shows a sharp increase in PV installation in the year 2022 (end of September), which already counts 63.52 GW but was 59.98 and 58.98 GW in the year 2021, and 2020 respectively [3].

Most of the PV installations are residential, with rooftop deployments. However, not every rooftop is suitable for solar panel installation. The model would forecast the roof topology's viability. The solar potential is determined by the solar irradiance received by the roof's surface and is influenced by the roof's orientation and slope. It is well known that the south-facing roof receives more sunlight in the countries in the northern hemisphere [5]. Only predicting the roof and calculating the PV potential from the entire roof would give potentially incomplete results; we must also consider the rooftop's orientation and slope. The other factors, like trees and other structures' shading on the roof segment, are not considered in the thesis due to the lack of a labeled dataset.

Two alternative PV potential models were developed independently at the Technical University of Munich. The first uses CityGML-based simulations using 3D city models [6]. It calculates the geographic potential by considering direct, diffuse, and global solar irradiation and shadowing effects [7]. The other uses a convolutional neural network (CNN) to segment accessible roof surfaces from aerial images, arrange PV panels appropriately, and evaluate the resultant economic PV potential [5]. Similar to the second strategy outlined above, the master's thesis would focus on an alternate approach for roof segmentation and the roof's orientation rather than assessing the PV potential. The ultimate goal is to assess the PV potential using all the factors like shading, the roof's orientation, slope, and geographical viability, which is out of the scope of the thesis.

The authors [5], [8], [9] employ artificial neural networks to assess solar roof potential from aerial images. The key benefit of this method over LiDAR-based approaches is that the required data is less expensive to acquire and more generally available [10],

allowing for cost-effective evaluation even in distant and under-surveyed locations. Some related tasks include detecting roof areas and classifying roof types from the images [10][11]. A fully convolutional DCNN with convolutional and deconvolutional layers was developed to perform building roof segmentation and provide an end-to-end operational strategy for correctly mapping labeled roofs with feature extraction and image segmentation. Krapf et al. [9] designed and trained two convolutional neural networks (CNNs) that employ semantic segmentation to recognize roof segments and superstructures, respectively, and then used the results in their technique to calculate economic PV potential.

Previous similar work has designed a model in which the whole roof is used as a labeled dataset and trained along with the aerial images. The thesis would be more concerned with extracting information about the roof, mainly the orientations using the roof information like gables, gutters, and rooflines. After extracting the roof and other roof information using the semantic segmentation technique, the post-processing and some geometric operations help to identify the roof segment's orientation.

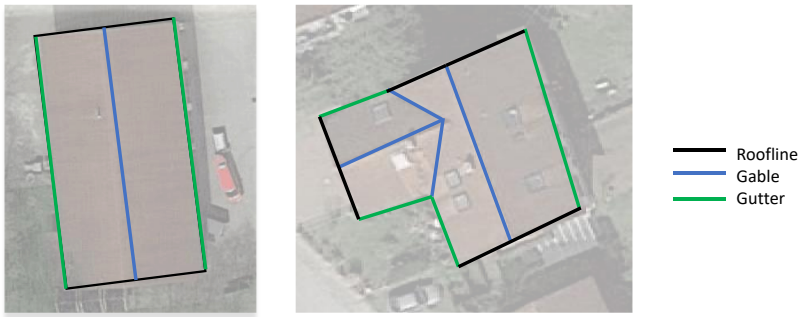


Figure 1: Diagram showing different lines: roofline, gable, and gutter line present in the roof structure

1.2 Problem and Objective

The estimation of rooftop PV potential has been the subject of many researchers. The primary criteria in assessing rooftop PV potential are the PV installation's usable roof area, the PV energy received by the rooftop, and the technical characteristics of the PV system [12]. Various research uses different methods to establish these factors. Statistical approaches, LiDAR, GIS-based, and machine learning methods are prevalent for estimating the rooftop PV potential [13]. Based on the scope of the research and the level of detail required, some of these methodologies offer benefits over others. For large-scale studies, statistical approaches that rely on statistical data like weather patterns and building features in an area are beneficial [5]. The statistical approach is used when the speed and complexity of the study are more essential than accuracy, and it usually incorporates the entire region for the result.

On the other hand, GIS-based approaches that depend on LiDAR data are more accurate than statistical methods. However, their application is restricted by the availability of 3D data for a specific location, and the process is computationally costly. In places where 3D data is unavailable, estimating PV potential analysis needs the use of publicly accessible data like satellite images and Open Street Maps (OSM) [14].

The thesis concentrates on segmenting rooflines, roof segments, and gutters while ignoring other superstructures, such as chimneys. Though detecting the superstructure would aid in calculating the useable roof area, the primary emphasis would be detecting the roofline, roof segments, and gutters to determine the roof's orientation. Roof orientation should not be ignored since it is a significant aspect in determining the real solar potential of the roof [5]. In the proposed thesis, it is not feasible to determine the slope of the roof or estimate the solar irradiation; nevertheless, additional work would be required to incorporate these aspects into the same application. Using costly GIS data and 3D city models is critical in this use case. Hence, this thesis does not consider slope determination as it might increase the project cost. Two roof surfaces of the same size and location but with different orientations and inclinations (azimuth and tilt) may have significantly varied solar potential [15], [16]. The thesis aims to develop a model for roof segmentation utilizing the human-labeled dataset and the Google Maps aerial image dataset using the knowledge provided above. This thesis also requires a significant amount of post-processing work. Post-processing comprises converting the resulting raster to vector format and establishing the orientation (North, East, West, and South) to the fixed (True North). The critical job of this project would be the post-work on selecting the proper orientation for the specific location and estimating the potential of the broad extent region.

PV potential analysis using aerial images requires extracting the roof segments and their orientations. The existing approaches use deep learning for this task and train a model on 18 orientation classes [5]. The existing approach is cumbersome and inefficient. Even though results seem promising, the approach of using 18 orientation classes has two significant downsides:

1) It defines 18 classes to derive one information, the roof orientation.

2) Roof orientation is inherently skewed due to architectural reasons (preferred north/south orientation)

Both aspects result in a more challenging situation for training the network. Reducing the number of classes would lead to less detailed roof orientation (bins of 45° instead of 22.5°). Deriving a balanced dataset is challenging and only possible with data augmentation.

Therefore, this thesis explores an alternative deep learning approach that applies a different class definition to derive the required roof information. The objective is to compare the results from the two approaches quantitatively to point out the strengths and weaknesses.

1.3 Structure of thesis

The thesis is separated into six different sections. The introduction is already mentioned in chapter 1.

Chapter 2 introduces the theoretical background, where the basic concepts of machine learning, Artificial Neural Networks (ANN), Convolution Neural Networks (CNN),

semantic segmentation, and U-Net are presented. This chapter is classified further into five sections with several sub-sections.

Chapter 3 presents the literature review, where different research papers are discussed and extract the vital information for the thesis. This chapter includes the study and summary of the essential related papers that have used machine learning and CNN techniques for rooftop segmentation and post-processing. The brief introduction of the related paper is presented and summarized with the research gap.

Chapter 4 demonstrates methodology, where the different model architectures and post-processing steps are included. The image processing steps are detailed in these sections with seven sub-sections, including georeferencing the predicted images, roofline removal, gutter extraction, conversion to vector, azimuth calculation techniques, and orientation determination.

Results and Discussions are discussed in chapter 5. The chapter includes the training results on different model architectures with training curves. The predicted images are also assessed with the ground truth dataset, and the result is presented in the confusion matrix in sub-section 5.3. This section also includes the selection of the best configuration, the orientation result, and the review of the Hypothesis.

The last chapter 6 concludes the thesis. This chapter explains the pros and cons of the experiments performed in the thesis. The summary and the suggestion for further development have been presented in this chapter.

2 Theoretical Background

When AI pioneer Arthur Samuel created the first self-learning checkers system in the 1960s, the phrase "machine learning" was first used [17]. After Geoffrey Hinton and colleagues [18] made a breakthrough in AI in 2006 by suggesting a strategy for developing deeper neural networks and a method to prevent gradient vanishing during training, AI gained prominence. Since then, the interest in data-centric AI applications has steadily increased. Machine Learning is a part of AI that a computer or a program can use to learn and acquire intelligence without human intervention. Big data technologies and the improvement of computing power have made deriving features and information from massive data samples more efficient. ML methods have been developed to analyze high throughput data to obtain valuable insights, categorize, predict, and make novel evidence-based decisions. Unlike traditional approaches – sets of explicitly written instructions- a sub-discipline of Artificial Intelligence- ML allows the computer to learn themselves and find the functions that map inputs to the desired output. The machine learning problem varies based on the data and desired output. The machine learning problem could be broadly classified as supervised machine learning and Unsupervised machine learning problems [19]. The data sample is a pair of input features and the corresponding label or ground truth in the first type of problem. According to the classes defined, the machine learning model divides the input into discrete variables and places the variable in the specific class determined by the probability distribution over the given classes. This kind of machine-learning problem is also called classification.

On the contrary, if there is only input data and no information on the ground truth data, the problem is called an unsupervised machine learning problem. It is often referred to as regression. The predicted output is a continuous variable, and the machine learning algorithm detects the pattern and clusters them with the help of distinguishing categories [20].

2.1 Artificial Neural Network

Deep learning is a subset of machine learning which involves letting the machine learn from the data itself. Neural networks are the core of all deep learning algorithms. The neural network algorithm is inspired by the performance of the biological brain and its processes. Like the brain, the neural network takes the input, processes the input, and generates the result. The primary processing component of a neural network is called a neuron, which could be imagined as logistic regression. ANN is a group of multiple neurons. ANN is called a feed-forward neural network, as inputs are processed only in the forward direction. It consists of 3 layers: an input layer that accepts the input, one or more hidden layers which process the input, and finally, the output layer, which generates the output. Images are fed into the input layer in the form of a number. During the forward propagation, an input to the neuron is transformed into the new form using the linear transformation followed by the non-linear activation function. The

weighted sum with a bias from the input is passed to the activation function to produce the output. This activation function helps the network learn any complex relationship between the inputs and the outputs. The use of deep learning has advantages over traditional machine learning because feature extraction happens automatically in deep learning. In contrast, it has to be done manually in the latter case.

While solving a complex dataset, for example, image segmentation, the input must be converted from 2- dimensions to 1- dimension before training. The all-to-all connection between the numerous adjacent layer in ANN increases the size of the trainable parameters, increasing the complexity and the computational cost. ANN cannot capture sequential information, which is required in capturing the sequence data. ANN loses the spatial features of an image while converting the 2-dim inputs into a single-dimension result. The reasons above showed that the classical neural network is unsuitable for performing image-based machine learning tasks.

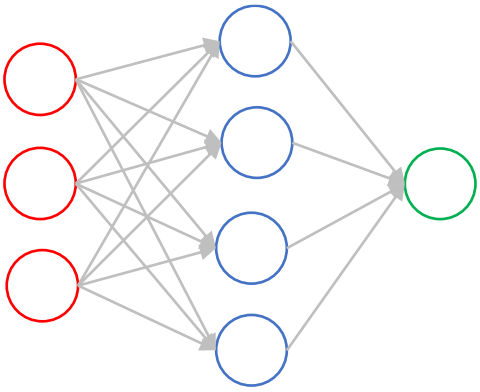


Figure 2: A feedforward ANN with input layers (in red), a hidden layer (Blue), and the output layer (Green), which are connected to the weighted connection lines

2.2 Convolution Neural Network

Convolutional neural networks are the subgroup of deep learning algorithm that takes the input, assign the weights and biases to the objects in the images, and differentiate different objects in the image. CNNs are more often used in image classification and computer vision tasks. Before the introduction of CNNs, the manual classification and feature extractions task were used to identify the objects in the images. With the introduction CNNs algorithms, the traditional methods are being phased out, as the CNNs are scalable, efficient, and cost-effective. CNN could perform image classification and object recognition, leveraging principles from linear algebra, specifically matrix multiplication, to identify patterns within an image [21]. However, it

is computationally demanding, requiring a high Graphical Processing Unit to train the model.

2.2.1 Architecture of CNN

The architecture of CNN was analogous to the connectivity pattern of the human brain's neurons and was inspired by the organization of the Visual Cortex. The visual cortex in mammals consists of a simple cell with a local receptive field similar to the filters or kernel of CNN and a complex cell similar to the pooling, which enables us to detect the rich features in the input image [22]. Convolution neural network consists of 3 primary layers: Convolution Layer, Pooling Layer, and Fully Connected Layer. The first layer, the convolution layer, could be followed by further convolutional layers or pooling layers; finally, in the fully connected layer [23]—the complexity of CNN increases in each subsequent layer which identifies the more significant portion of the image. The previous layer focuses on the image color and the edges. As the data are passed into the subsequent layers, it starts recognizing the elements and objects in an image and consequently identifies the element in an image. CNN can capture an image's spatial and temporal correlation by applying filters. The restricted region of the visual field, which is spatially correlated, is called the local receptive field. Each respective local receptive field's pixel of the subsequent convolution layer is connected to the previous convolution layer preserving the spatial correction [24].

2.2.2 Convolution Layer

The convolution Layer is a building block of a CNN where most of the computation occurs. The layer requires components like input data, a filter, and a feature map. If the input is an RGB image made up of a matrix of pixels in 3D, it would have three dimensions: height, width, and depth. A kernel or filter is a feature detector that moves across the image's receptive fields and checks whether the feature is present. A kernel is an operator that operates on a single input channel. When the collections of kernels operate on multiple inputs, it is generally called a filter. The kernel is a matrix of numerical values with a certain standard weight and will be updated during the learning process. The dot product between the input and kernel matrix is calculated to get the convolved element. The kernel (filter) shifts by stride over the input matrix. The process is continuous until it covers the entire image. The stride size determines how the kernel moves in the input matrix. This process is known as convolution.

The input channels were convolved using the kernel and summed up together. A bias is added to the dot product, and the feature map is created. Thus created feature map can also be termed an activation map or convolved feature. As seen in Figure 2, there is no need for every output feature (pixel) to be connected to every pixel in the input. However, every element of the output feature should be connected to the respective receptive field. Since every pixel of the output features is not connected to each pixel

of the input features, the convolved layer is also called the “partially connected” layer. It could also be termed local connectivity.

The number of feature maps is determined by the number of filters used, which could be determined and optimized during the model's training. The number of filters, stride, and zero padding are the three hyperparameters that control the output feature's size volume [23]. The CNN has sparse interaction and weight-sharing features, enabling it to rely on fewer parameters than the traditional neural network. The same kernel slides over the input channel sharing the same weight and helping reduce parameters, reducing the memory requirement, and increasing the model's statistical accuracy. The weights are adjusted gradually during backpropagation and gradient descent. The number of parameters involved is calculated by using the formula:

$$n_p = k_h * k_w * n_{in} * n_{out} + n_{out} \quad (1)$$

Where n_p is the number of parameters, k_h is the height of the kernel, k_w is the width of the kernel, n_{in} and n_{out} are the number of input channels and output channels, respectively.

Similarly, the output size is calculated using the formula:

$$S_{out} = \frac{S_{in} - k + 2p}{s} + 1 \quad (2)$$

Where S_{out}, S_{in}, k, p, and s are the size of the output volume, sin is the input size, k is the kernel size, and s is the stride along the width, respectively.

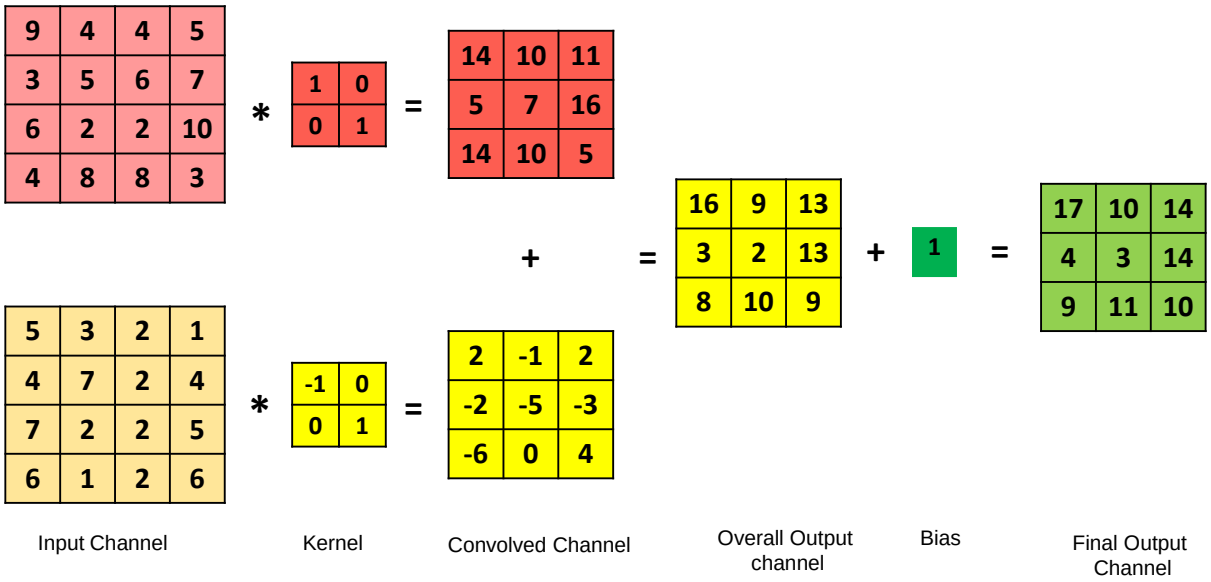


Figure 3: Convolution operation on 4*4*2 input to produce an output channel of 3*3 with a kernel of 2*2 stride size of 1*1

After each convolution layer, a Rectified Linear Unit (ReLU) transformation is carried out to the feature map to introduce non-linearity to the model.

The size of the layer is reduced during the convolution. Some information gets lost during the convolution process, as the corner part of the images is covered less frequently than the middle part when the kernel moves along the image. Due to this, zero padding is generally introduced, which applies 0 value to the outer frame of the image. The padding increases the input size but preserves the information contained in the corners.

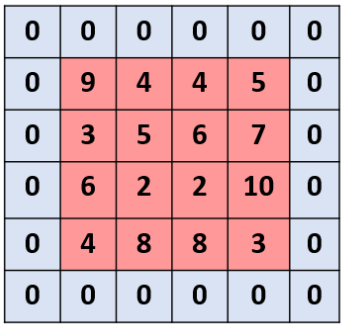


Figure 4: Input channel by adding zero padding to preserve the output channel size

The network's structure becomes hierarchical as additional convolution layers are added after the first. Those layers may be able to perceive the pixels in the last layer's receptive field. As a result, by using the lower-level pattern to the higher-level pattern, a feature hierarchy is established. For instance, if a human face is used as the input image, the lower hierarchy would be the human face's features, such as the eyes,

nose, mouth, ears, and hair. Moreover, the entire lower hierarchy is put together to create a face [25].

2.2.3 Pooling layer

Pooling is similar to the kernels that have been used in the convolution layer, but the difference is that the pooling layer does not have weight and trainable parameters. It sweeps the filter across the input channel, and it helps to reduce the dimension of the input layers. Additionally, the input layer's parameter is decreased. Because of this, it is also known as downsampling. The two types of pooling are max pooling and average pooling. The filter is passed into the receptive field. If the maximum value in the receptive field of the input channel is passed into the output array, it is termed max pooling. Similarly, if the average value of pixels of the receptive field is calculated and sent to the output array, it is called average pooling. During the pooling, much information is lost, but it decreases complexity, increases efficiency, and reduces the risk of overfitting.

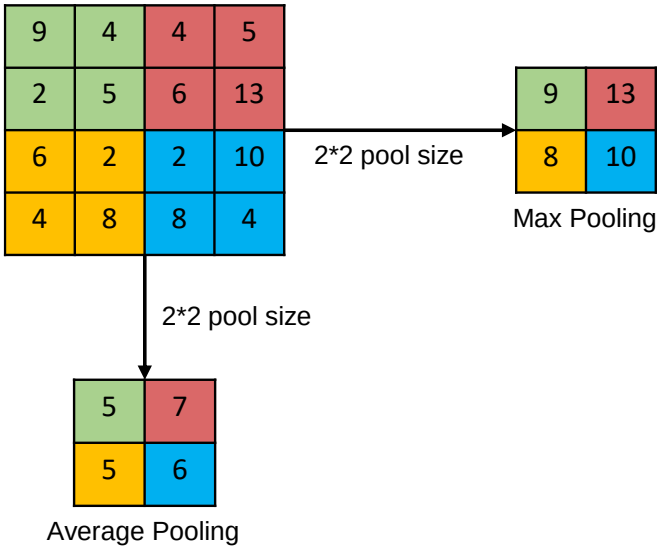


Figure 5: Max Pooling and Average Pooling operation with 2*2-pixel filter size from 4*4 pixel input

2.2.4 Fully Connected Layer

The feed-forward network, the final layer of the CNN, is the fully connected layer. The output of the pooling layer or the convolution layer after flattening serves as the fully connected layer's input. The input image is transformed into a format that the Multi-Level Perceptron can use. The image is flattened into a tensor (column vector). The flattened tensors are then connected to a fully connected layer like the previously mentioned network called artificial neural network, and backpropagation is applied to every iteration of training. Hence, each node in the output layer is connected to every node (neuron) in the previous layer. This layer performs the classification task,

enabling the model to distinguish between dominating and low-level features in an image.

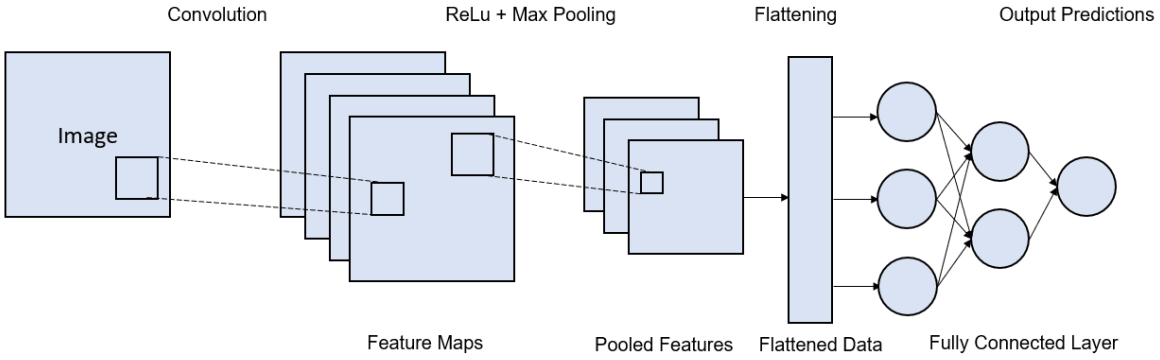


Figure 6: Fully Connected Layer

CNN architectures have various variants like AlexNet, VGGNet, GoogLeNet, and ResNet.

2.2.5 Activation Function

The activation function determines if the neuron should be activated or not before sending it to the output layer. It will introduce a simple mathematic function that determines if the neuron plays a role in the prediction and classification tasks. The primary purpose of the activation function is to introduce the non-linearity to the neural network before generating an output from the input images. The qualities of activation functions include non-linearity, continuous differentiation, monotonicity, and a particular output range. Activation functions can be categorized into 3 types: Linear activation function, binary step function, and non-linear activation function. Since our main concern is introducing non-linearity to the network, the non-linear activation functions are only discussed. A neural network without an activation function would be like a simple linear regression model, and it would not be able to learn the complex functional mapping required for the complex data. A linear activation function could not perform backpropagation as it is non-differentiable and would lack efficiency and accuracy. Some of the common activation functions used in Machine Learning are described below.

2.2.5.1 Sigmoid

The sigmoid function is mainly used in classification problems whose output value ranges between 0 and 1. It has all the qualities of the activation functions and gives the output in probability. The drawbacks of sigmoid function include two significant problems. One of them is vanishing gradients. Since the output value ranges from 0 to 1, the loss gradient would be saturated, and the weights could not be updated, resulting in saturation in the output. Another problem is that if it is not zero-centered, the gradient diverges too far in a different direction, making the optimization difficult and unstable. Mathematically, the sigmoid function is given by $\sigma(x)$

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{3}$$

Figure 7 shows the sigmoid graph in which the x-axis represents the input x, and the y-axis represents the output y.

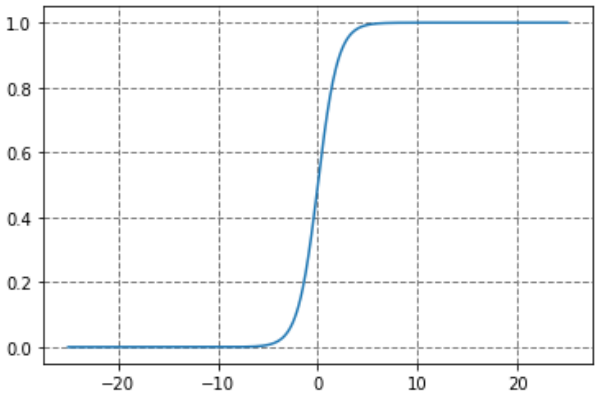


Figure 7: Sigmoid Function

2.2.5.2 TanH

TanH is similar to a sigmoid function but solves some disadvantages of sigmoid. The output value of the function ranges from -1 to 1 and is zero-centered. However, there is still the vanishing gradient problem, but the gradient is more robust than sigmoid.

Mathematically, TanH function could be represented by $\tanh(x)$, where the x is the input value.

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{4}$$

Figure 8 shows the sigmoid graph in which the x-axis represents the input x, and the y-axis represents the output y. Here the value of the output ranges from -1 to 1.

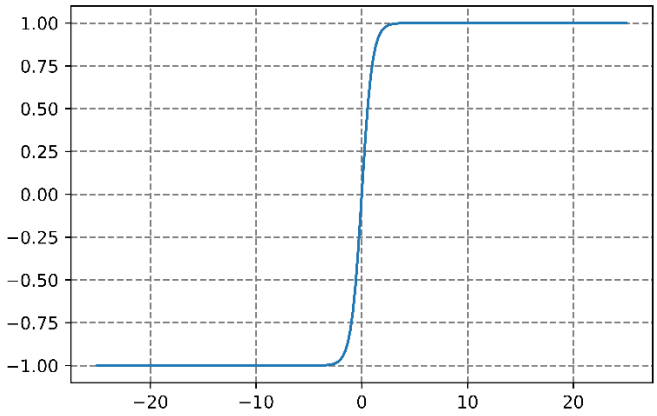


Figure 8: TanH Function

2.2.5.3 Rectified Linear Unit (ReLU)

ReLU has the problem of being computationally efficient as it has solved the problem of vanishing gradient or exploding gradient. Its output value ranges from 0 to infinity. The ReLU unit blocks the neuron by thresholding the negative values to 0 and passing the positive values unchanged.

$$f(x) = max(0, x) \quad (5)$$

Since only a certain number of neurons are activated in ReLU, this function is computationally efficient. During the training, some gradients could be fragile, i.e., for ReLU activation, if the value is negative, the output would be 0, and the weight of this neuron (deactivated neuron) would not get adjusted during descent which leads to the dying ReLU problem. Another problem with ReLU is the exploding gradient since the activated neuron could range to an infinite value.

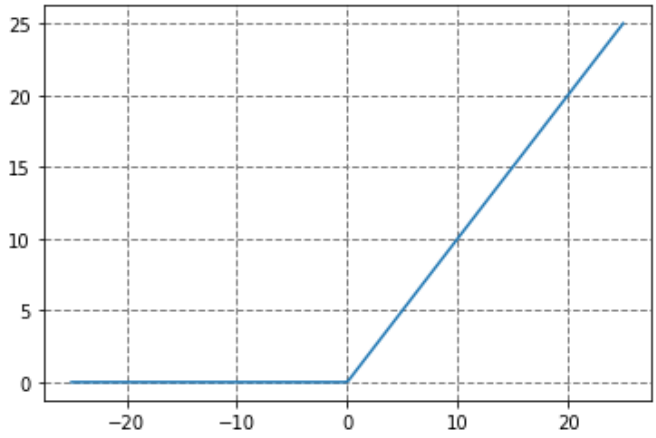


Figure 9: Rectified Linear Unit (ReLU) activation function

2.2.5.4 Leaky ReLU

Leaky ReLU is the version of the ReLU activation function in which a small value α is introduced in the negative value to prevent the dying ReLU problem. The small value

prevents the negative output from zero and helps introduce a slight positive slope in the negative area. This help prevents the neuron from being deactivated. It could also be called parametric ReLU.

$$y_x = \begin{cases} \alpha x & if\ x < 0 \\ x & if\ x \geq 0 \end{cases} \quad (6)$$

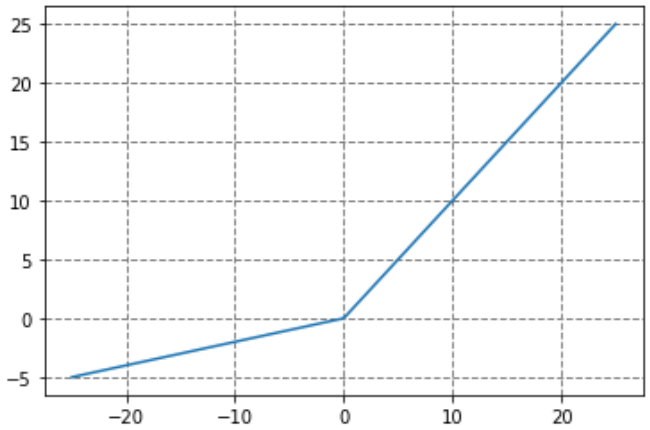


Figure 10: Leaky ReLU activation function

2.2.5.5 Exponential Linear Units (ELU)

ELU is one of the variations of the ReLU function. ELU also considers the negative values as in Leaky ReLU and prevents them from being deactivated by introducing the exponential curve to the leaky ReLU. The introduction of the exponential operation in the ELU could increase the computational time.

$$y_x = \begin{cases} \alpha(e^x - 1) & if\ x < 0 \\ x & if\ x \geq 0 \end{cases} \quad (7)$$

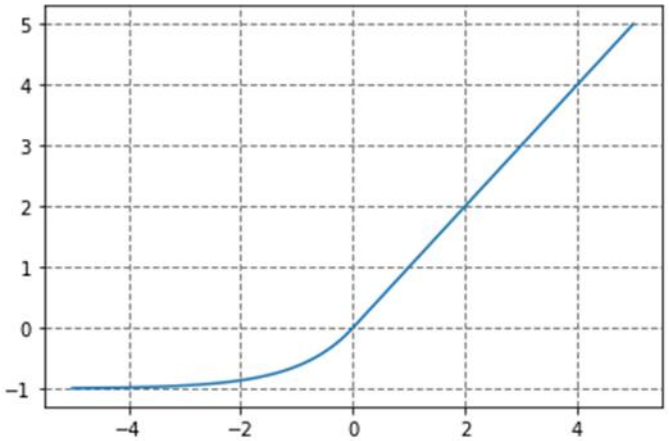


Figure 11: ELU activation function

2.2.5.6 SoftMax function

A combination of various sigmoid functions is referred to SoftMax function, which is used to predict the multinomial probability distribution. It is for multiclass classification problems, which is used to solve the problem of assigning an instance to one class when there are numerous classes. It is generally used in the last layer of the network. It gives the probability of the current class concerning the others.

2.2.6 Training of CNN

The training of CNN includes the optimization tasks, which minimize the loss to a minimum value by optimizing the parameters (weights and bias) on each epoch. The learning process includes 2 phases:

A Forward phase includes the input process, non-linearity of the network, and error calculation. The network, computing cross-correlation, entirely processes the input. Then non-linearity is applied using the activation function. As in equation (8), the error is calculated between the last output layer and the ground truth value. The objective is to update the parameter by minimizing the distance between y_i and $\widehat{y_i}$.

$$E = \frac{1}{n}\sum_{i=1}^{n}|y_i - \widehat{y_i}|^2 \tag{8}$$

The backward phase and parameter update is the fundamental concept in the learning process of neural networks. The objective is to update the parameters regularly and feed the updated parameter into the network [26].

2.3 Semantic Segmentation

Image Segmentation divides the images into separate segments. It is not essential to process the whole image at a time as some regions might not contain any information we wanted to extract. Image segmentation aims to change the image's representation into something meaningful and unique. Image is the collection of pixels. Hence the main task of semantic segmentation is to assign a label to every pixel in the image, classify the particular class of the image and separate it from the rest of the classes by overlaying it with a mask [27]. Semantic segmentation [28] could be considered the classification of the images at a pixel level. The goal of the semantic segmentation is to take an image as an input containing pixel values from 0 to 255 and convert them to a class label according to the classes defined by the user (0,1,2,…,n). Extracting features that could derive some meaningful correlation in the input and removing the noise is an essential aspect of semantic segmentation in deep learning. CNN performs the task of semantic segmentation. There are other types of segmentation like Instance segmentation and panoptic segmentation. However, the aim is to segment some features like Roof segment, roofline, and gables from the background, which could be possible w of semantic segmentation.

Classification [29] involves classifying particular objects in the images, which is termed classification. It would only classify the particular object in the input images but does not give any spatial information or the object's shape. By constructing a bounding box,

localization [27] locates the object in the input image [30]. Though it gives the spatial location of the object class, it does not give any information about the object's shape. Hence it would be too vague for our purpose. Hence segmentation came to play a role that could do both the task mentioned above by grouping the pixels with similar attributes into a class and overlaying these classes into a suitable class while giving an output.

CNN has fully connected layers at the end to predict the label class. However, our task does not need to predict the label class but needs a segmentation with the spatial information. The aim is to extract the feature from the image. Hence the fully connected layer is not needed at the end. Due to max-pooling layers, the images' size is reduced; instead, we used the upsampling layer to maintain the size of the output. We upsample it using the interpolation techniques like the nearest neighbor, max unpooling, and Bed of Nails [31]. The convolution network extracted features from the image is called an encoder. It downsamples the images. The network which upsamples the images is known as the decoder. Since the output of the decoder is rough because the important message gets lost in the last convolution layer, an upsampling technique is used. They are FCN-16 [32] which takes the information from the previous pooling to the final feature map, and FCN-8, which even takes the information from one more previous pooling layer, which helps to give a better result.

Some of the standard deep networks that play a role in the field of computer vision and semantic segmentation are AlexNet[33], VGG-16[29], GoogLeNet[34], and ResNet[30]. We have used ResNet in our model as it is well known due to its depth (multiple layers) and has introduced residual block. It introduces identity skip connections to address the problem of training in very deep architecture, which enables copying the layers from their input to the next layer.

2.4 Regularization techniques

The main aim of the regularization technique in deep learning is to lower the complexity of the neural network and prevent overfitting. The standard regularization techniques are Data augmentation[35], L1 regularization, L2 regularization[36], and drop out. If the training data is limited, it will lead to less generalization and may lead to overfitting the model. Hence it is considered one of the standard techniques to improve CNN's performance. It means increasing the number of trainable data without adding any external data. This could be done by translating, rotating, scaling, flipping the existing data, and adding to the trainable dataset. Another method of regularization is L1 regularization and L2 regularization, which add the sum of total weights and squared weights to the loss function. During the training, some neuron with some probability P gets turned off. These neurons get ignored and are not considered part of the neural network. This technique, called a dropout, makes the model less complex and reduces overfitting.

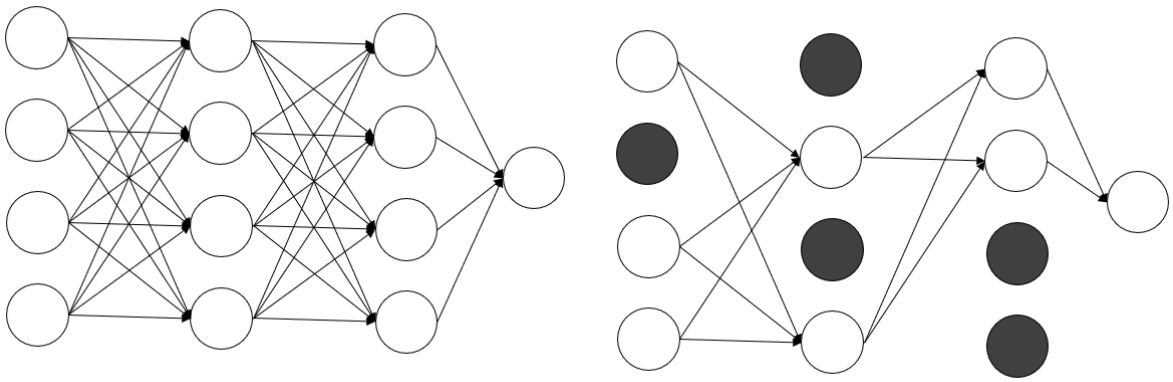


Figure 12: Illustration of a dropout in a network. The standard network is on the left, and the dropout network is on the right

2.5 U-Net

U-Net [37] is the most widely utilized CNN architecture for image segmentation that is both quick and exact. It may be thought of as a two-phase encoder and decoder. The first is a downsampling phase that combines convolution with max-pooling layers to collect features at all scales, from the smallest to the largest. The second step involves upsampling the first's result to restore the features to the original image's format. The output of each level of the down-sampling phase is supplied directly to the corresponding up-sampling phase to avoid the need for separate encoder and decoder training. A graphical illustration of the network and a typical max-pool layer operation is shown in Figure 13.

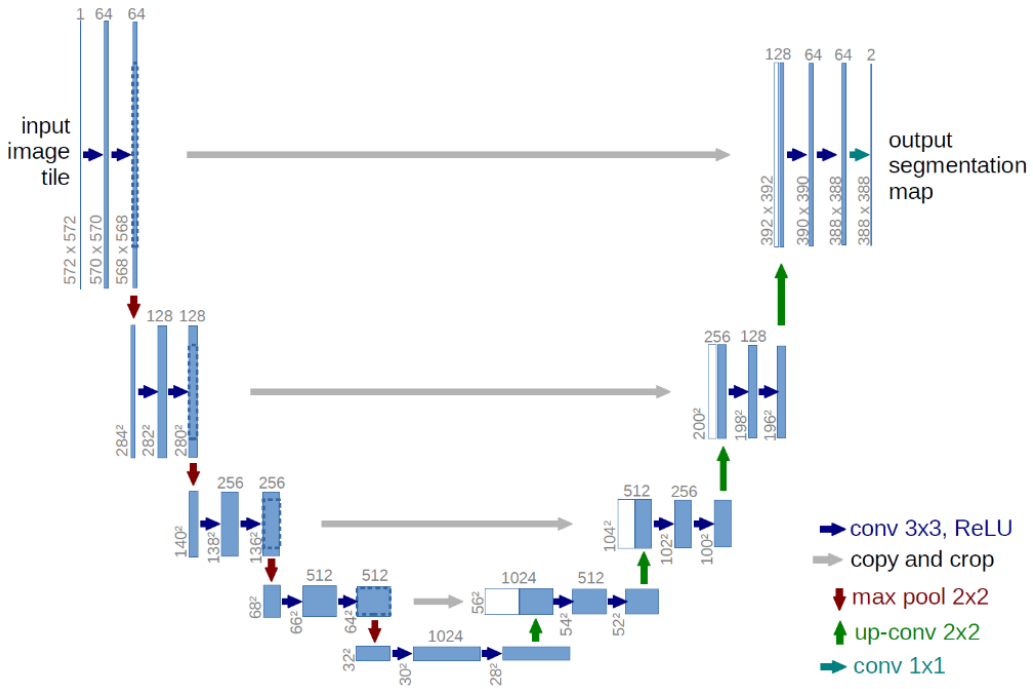


Figure 13: Figure of Original U-Net, which shows the encoder and decoder [38]

3 State of Research

3.1 Literature Review

Identifying PV potentials has previously been covered in various papers. The evaluation of accessible area for PV modules, the modeling of solar irradiance on slanted module surfaces, and the computation of generated electrical power from the irradiance on these modules are the primary phases in PV potential estimating methodologies [10]. Some significant works have been thoroughly analyzed in order to identify research gaps. The approaches, methodology, and assessment scale employed for estimating various parameters and outcomes have been investigated. This part will serve as the foundation for this thesis.

Using Chinese Very High Resolution (VHR) satellite (i.e., GF2) data, Qin et al. [11] provide a unique technique for semantic segmentation of building roofs in congested urban areas using a Deep Convolution Neural Network (DCNN). A predictive model with convolution layers was created for building roof segmentation using building roof samples gathered from several cities. The validation indicates that DCNN-based semantic segmentation results have an overall accuracy of 94.67 percent and a mean Intersection Over Union (mIOU) of 0.85, respectively. In contrast, Conditional Random Fields (CRF) refined segmentation results have an overall accuracy of 94.69 % and an mIOU of 0.83. The findings suggest that this method could generate roof mapping with VHR imagery in a crowded area with various buildings.

Lee et al. [5] present a semantic segmentation Convolutional Neural Network (CNN) approach for determining roof segments and their orientation. The author has developed a methodology that assesses a roof's solar potential using freely available satellite imagery and a convolution neural network to offer a pixel-level assessment of each planar roof segment's solar potential. The model is assessed in Framingham, Massachusetts, and the dataset is based on annotated images from six cities in the United States. Planar roof segments are identified using a Feature Pyramid Network (FPN). The validation indicates that the model can extract the roof geometry, such as planar roof segments and orientation, with a valid positive rate of 91.1% of the roof and a mean orientation error (MOE) of 9.3%. The ground truth labeled data and model output was compared for validation. They double-checked their findings with two independent solar specialists and received an 8 out of 10 rating. They compared their deep learning results to the Google Sunroof, a LiDAR-based technology, and found a -11 % to +11 % difference in the available solar installation area.

Castello et al. [8] used pixel-wise image segmentation to demarcate rooftop solar panels and determine their sizes. High-quality aerial images given by the Swiss Federal Office of Topography were utilized as input. In order to improve model performance, several data augmentation techniques were applied, and network parameters were changed. According to the findings, the identification of test images had an accuracy of around 94% and an Intersection over the Union index of up to 0.64. The previous images were augmented by producing two lighting settings and three

90° rotation sets for each image, and the number of images was increased from 780 to 4680 (780*2*3=4680). The author used a weighted pixel-wise categorical cross-entropy function to give the false-negative loss more weight. The article adopts a U-net architecture with a total of 471586 trainable parameters. The validation loss and convergence of Adam and the Stochastic Gradient Descent optimizer were compared, and it was discovered that Adam provided decent convergence until 75 epochs.

Krapf et al. [39] have presented the Roof Information Dataset for semantic segmentation of roof segments and superstructures. The RID dataset contains 1880 labeled roofs, with an area covering 1.5 km^2, without considering overlap, and 4.9 km^2. The roof section annotations contain 4520 polygons, and their azimuths were used to determine their classes. With 12359 superstructure labeled polygons, the author has developed eight roof superstructure classes that may be assigned to structural items (PV module, dormer, window, ladder, and chimney), natural objects (tree and shadow), or other objects with an unknown class. The author evaluated the problem of categorizing superstructures in high-resolution aerial images and explored obstacles by analyzing the annotation agreement of five annotators on 26 images. The IOU of the dormers and PV modules is 0.7 and 0.68, respectively, while annotation agreement was low for more ambiguous classes like shadows (0.29) and unknowns (0.15). They have given early and approved annotations for each dataset to encourage data-centric exploration. They discovered that the technical PV potential was severely reduced when studying superstructures.

Mainzer et al. [12] tried to access the remaining PV potential by combining publicly available geographical building data and aerial images. They have used image recognition techniques and machine learning approaches to determine the roof's azimuth. For example, for the roof ridge line detection on the aerial image, they used methods like bilateral filtering, color filtering, histogram equalization, canny edge detection, hough line transformation, and logical filtering. When multiple lines exist, they use the weighted sum of criteria length and brightness difference to filter the correct ridge line out of multiple lines. The authors report an accuracy of 90.97% for detecting possible PV installations for Freiburg. Using CNN, they applied the method to the city of Freiburg, Germany and compared the result of the azimuth with a 3D model of Freiburg containing 26,412 matched partial roof areas. The 3D model was assumed to be 100% correct, and a density plot between the model's simulated azimuth and the azimuth of the 3D model was compared. They concluded that most errors occur due to a deviation of ±90° or ±270° when the roof gutters could not be identified. They considered that their algorithm was fooled by multiple ridges, e.g., on hip roofs. They concluded that the errors are pretty symmetric, which means the algorithm does not favor a deviation in a specific direction. Hence, the result does not produce any systematic error, as an existing error could compromise the result in terms of power generation.

Soares et al. [40] applied a segmentation algorithm to obtain each roof segment and the superstructures contained in a roof that may obstruct the installation of the PV module. The roof's 3D geometry (pitch and azimuth) was created by combining geometrical techniques and a Random Forest algorithm. A geometric packing algorithm calculated the maximum number of modules that could be fitted in the

remaining roof segments. Finally, a shading mask was calculated, and photo voltaic power was calculated using meteorological data, module orientation, and the shading effect. They divided the task into two semantic segmentation tasks. One is employed to segment the image into the background, roof sections, and roof ridges. In contrast, another is employed to subdivide the images into the background and a set of chosen roof objects. Both the segmentation models use U-Net architecture using a ResNet-34 as a backbone. The dataset includes the aerial images obtained from IGN (French National Institute of Geographic and Forest Information). They used 3D city data to obtain the roof label, but manually tagged images are used for the superstructures data. A different algorithm for the azimuth calculation was employed.

The azimuth is calculated using a strictly geometric algorithm and conforms to the roof's orientation. In order to determine the direction of the bounding box of the roof, θ_{bb} was computed. Then, they proposed that the roof azimuth takes the following form: $\theta_{bb}+\Delta\theta$, which may have one of the following four values: 0°, 90°, 180°, or 270°. This results in a classification issue with four classes for prediction. Finally, based on the theory that roof parts often face away from surrounding roofs, they have utilized nearby roof sections as a signal of the optimal orientation. When excluding the background, the roof segmentation achieved pixel-wise accuracy of 77% for roof sections and 30% for roof objects. They presented that the azimuth model reached good results with an accuracy of 79%. They observed that the East-facing rooftop reduces the solar potential by 25%. They concluded that the roof object (superstructure) segmentation and pitch prediction presented relatively poor results and found that roof section segmentation and azimuth prediction gave excellent results.

Deep Roof Refiner [41] studied delineating roof structure lines using satellite imagery. The author employed a deep learning model to find the roof structure lines (RSL) rather than the building footprints, which could provide much more sophisticated data reference. Qualitative and quantitative experiments were performed, resulting in an F1 score of 60.89% for the optimal dataset scale and 63.48% for the optimal image scale. The paper concludes that the RSL allows for the segmentation of roofs into various components, each of which may be mapped with additional attributes to provide data via GIS. The azimuth angle is determined using the longest line, clockwise, starting north.

3.2 Research Gap

PV energy plays a vital role in generating renewable energy. No concrete method has been developed to estimate the exact PV potential of the rooftop on a large scale. The precise assessment of PV potential on a broad scale is limited by the high cost of data sources (such as digital surface models and data from LiDAR) and often by the inadequacy of conventional image processing techniques. The project aims to develop an alternative approach for estimating the rooftop PV potential using remotely sensed data, GIS, and deep learning. The output of the intended project may be helpful in the renewable energy planner to estimate a rooftop’s PV potential capacity more accurately and affordably. Finding the most scalable approach that could accurately and inexpensively assess the PV potential has been the subject of extensive research.

Mainzer et al. [12] used the technique of image processing rather than CNN for roofline/ ridgeline detection. Due to shadows, window roofs, and building walls, the algorithm established in the research was unable to choose the right ridge line and instead selected a false ridge. Multiple ridgelines deceive the algorithm and could only be used for simple roof types like gabled roofs. The paper by Soares et al. [40] considered the nearby roof sections as an indicator of the correct orientation. The algorithm considers the hypothesis that roof sections tend to be oriented away from the nearby roof. However, the paper could not solve the issue of multiple roof segments and edge roofs. Lee et al. [5] provided a semantic segmentation CNN technique in their research for finding roof segments and their orientation. Krapf et al. [39] carried out the critical investigation that Lee et al. did not carry out, which involves the semantic segmentation of roof superstructures and the roof. The segmentation of the roof superstructure improved the PV potential assessment but lacked the roof orientation information, which was fulfilled by [42]. To determine orientation, the algorithm by [5] and [42] utilized the 16 different orientation classes for training a model. The labeling of the 16 distinct classes is time-consuming and expensive. The aimed project will replicate the previously described study to identify the optimal method for azimuth calculation and orientation determination. Instead of employing image processing techniques for computer vision, the project employs state-of-the-art image segmentation algorithms to improve the roof segment estimation. Later, the output produced by the segmentation model is subjected to post-processing, which uses image processing techniques to retrieve the relevant data. This project could be used broadly to help energy planners, local administration, and decision-makers working on renewable energy. The details of the implementation for the orientation determination is presented in the chapter 4.

Hypothesis Formulation

The thesis proposes two significant hypotheses for the experiment. The thesis aims to provide an answer to the following hypothesis:

1. CNN can segment the rooflines, roof, and background.
2. The predicted image from the model, which only has three classes: rooflines, roof, and background, can be used to estimate the orientation of a roof correctly.

4 Methodology

This section discusses an approach for determining the orientation of the roof. The procedures for data preparation, image processing, gutter and roof segment semantic segmentation, and azimuth calculation techniques are described in this section.

4.1 Overview

The workflow for the orientation determination is depicted in Figure 14. The thesis work has been completed in 3 parts. The first part includes preparing the data and training the model to output a prediction segmenting the roof segments and gutters. The roof and gutter segmentation tasks are done separately, as the data in segmenting these elements are performed differently. The second part consists of post-processing and vectorization, which includes extracting the vital information from the predicted image and vectorizing them. The last part consists of calculating the azimuth and determining its corresponding orientation. The training process, the model's architecture, and the post-processing of the roof segment and gutter are discussed separately in the later section.

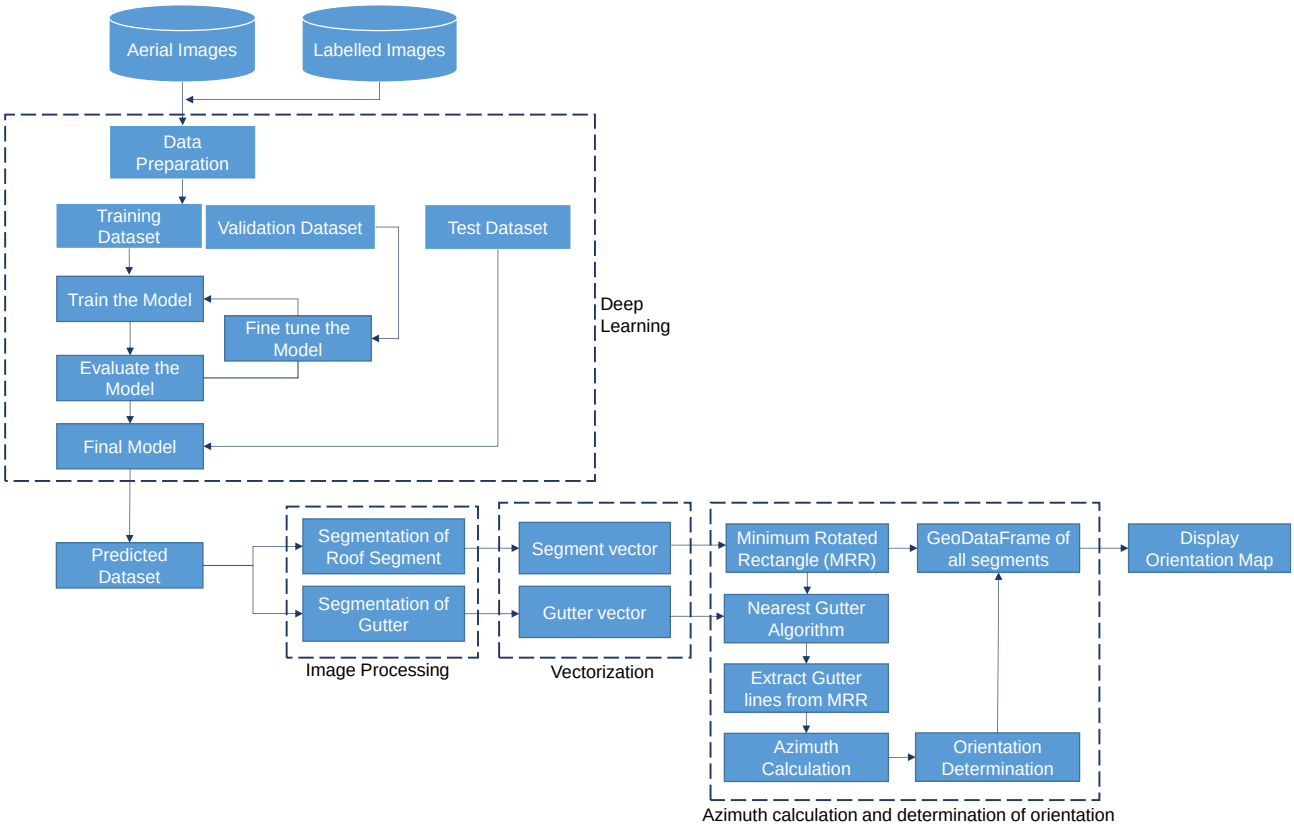


Figure 14: Overview of the presented approach showing the different methods involved

This thesis uses a deep learning approach for the semantic segmentation task, using a network architecture U-Net. U-Net seems to be the perfect architecture as it has proved a better result and accuracy where there is less training data. As sunlight is essential for solar potential, one must determine which roof segment would have more sunlight during the day. The south-facing roof gets more sunlight than the north-facing roof, which is a determining factor for the PV potential. Hence, this thesis tends to acquire the proper orientation of each roof and checks whether the roof can get enough sunlight.

4.2 Data and Software used

4.2.1 Data

A part of Wartenberg's municipality in Bavaria served as the basis for the thesis study area. The roof-centered aerial images from Google is taken as the primary dataset. The main goal of this thesis is to use deep learning to segment rooftops and determine their orientation using objects like gutters, rooflines, and the actual roof. In this thesis, the ridgelines present on the roof are considered rooflines. These ridgelines are represented as the center or outer lines for the gabled roofs, whereas they are depicted as the center, diagonal, or outer lines for the hip, complex, and pyramid roofs. The portion of the roof that does not include the rooflines and gutters is referred to as the roof segment. The gutter is an outer roof line that touches the roofline, mainly the edge line consisting of steel/iron structures used to discharge rainwater from the roof. The gutter might play an essential role in determining the roof's orientation and would be the most crucial element in calculating the azimuth of the roof segment.

The roof types are divided as gabled with a value of 1277. It is followed by a flat roof, complex roof, hip roof, and pyramid roof with a value of 397, 169, 32, and 22, respectively.

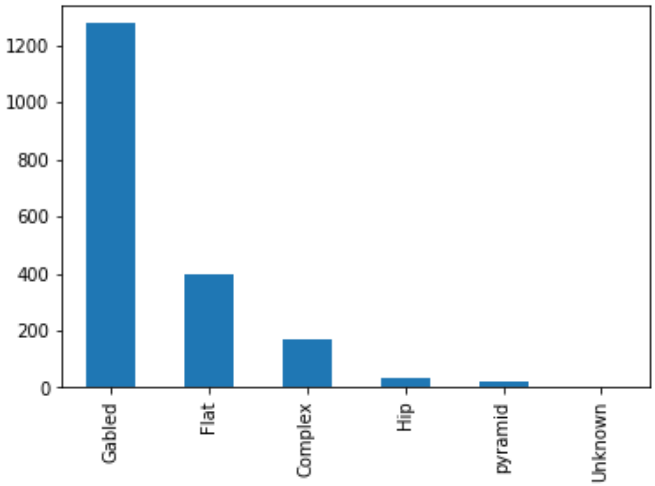


Figure 15: Illustration showing different types of roofs with their numbers that are used in the experiment.

4.2.2 Software and Hardware used:

This thesis used an open-source, cross-platform IDE called Spyder for the programming purpose. The labeling tool developed by TUM [39] was used to prepare the labeled data. The labeled image could not be rendered by the standard image viewer software of windows; hence, a Java-based software ImageJ [43], is used for viewing the labeled images, primarily grayscale. A licensed version of ArcGIS Pro and open-source GIS software are used to display geospatial data in the form of a map. A list of essential python packages employed in the programming part is keras, segmentation model, albumentation, h5py, numpy, pandas, OpenCV, matplotlib, seaborn, shapely, GeoPandas, and gdal. Different software from Microsoft Office is used for the preparation of this paper. All training was performed on an NVIDIA GeForce GTX 1080 Ti GPU with 11 GB memory.

4.3 Preparation of Data

The data need to be fed to the neural network during the model's training so that we can feed the test data to see the predicted output in the model. The aerial images from the Google API served as a primary data source in this work. The study area of the thesis is based on the part of the municipality Wartenberg, located in Bavaria. The 1880 number of roof-centered images were prepared with their corresponding annotations. The annotation includes the rooflines, roof, gutter, and superstructures. Though computationally expensive, images of 512*512 pixels are used. In this thesis, the superstructures were ignored, as our main aim was to find the roof segment’s orientation, and the superstructures would play little role in it. The datasets are divided into training, validation, and testing datasets, and the data splitting is described in the section below. Here, the network learns from the training dataset, the validation dataset is used to validate the model to prevent overfitting. Finally, the test dataset is used to predict the model used as our input in the further post-processing steps.

Data splitting

For each training scenario, the accompanying datasets are split into three subsets: one for neural network training, one for validation during training, and one for testing, which is also utilized for model evaluation. Instead of creating the grid-based dataset, the roof-centered images were created. A supposition is made that the roof’s center image would best fit the neural network model rather than the grid-based dataset, which could be considered a padded dataset. Another reason for taking the roof-centered image is that there would be a higher chance of the building being cut off at the corner in the grid-based dataset [42]. However, it is necessary to keep in mind the roof-centered dataset's drawback: the same roof would appear in different image datasets prepared for the training dataset. Hence, it is necessary to split the data so that the training dataset and the other datasets, the validation dataset and training dataset, do not include the samples in the training dataset. Out of 1880 images and masked datasets, the data are divided unconventionally into training, validation, and testing datasets with a percentage of 73%, 18.7%, and 8.1%, respectively. Hence the number of trained data is 1374, validation data is 352, and test data is 154. Random split of the roof-centered dataset leads to overlapping training and test images resulting in overfitting of the model [39]. Hence, these datasets are split in a way that the part of one dataset would not belong to another using a spatial operation.

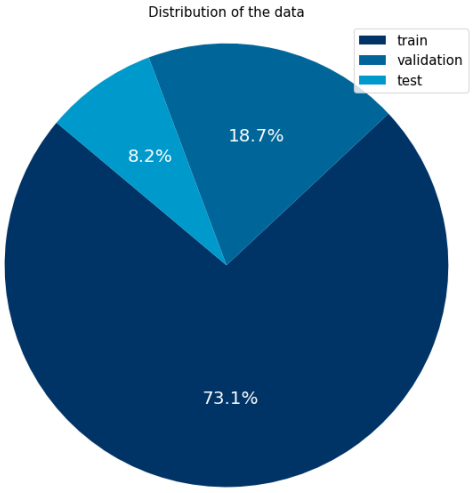


Figure 16: Pie Chart showing the data split into train, validation, and test dataset

In order to solve the above problem, the data were split geographically. Several portions of the area were selected for the validation and test dataset. The buildings of the portion were selected using the buffer operation by creating a circular area. The buildings included in the buffer zone were subtracted from the primary dataset. All buildings that intersected the extent of the samples chosen for the test set were also subtracted from the primary dataset to ensure that the datasets are entirely disjoint, meaning that no parts of buildings contained fully or partially in the test set are depicted in any of the other sets. A similar operation is conducted for the validation dataset in the remaining primary dataset. The remained dataset in the main dataset acts as a training dataset. The training could also be performed in another splitted dataset.

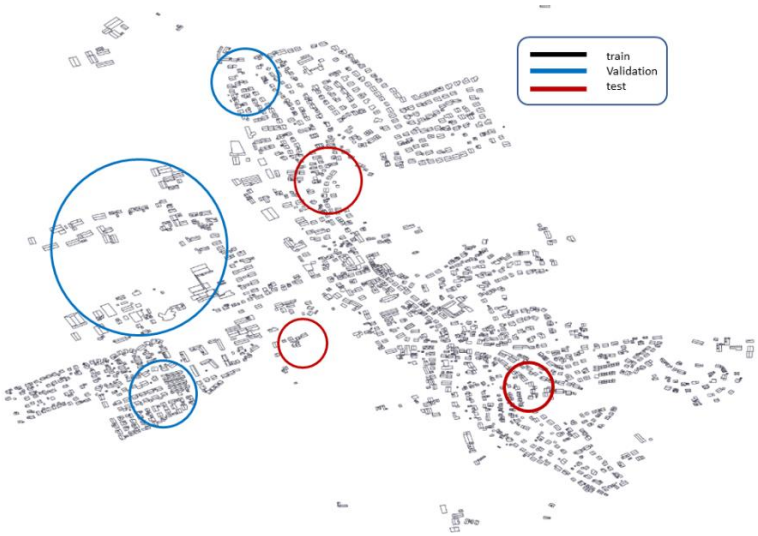


Figure 17: Map showing the data split of the Bavaria dataset into training (train), validation (val), and testing (test) dataset

However, since we are using a small area dataset and the roof structure of the whole area is similar, the network could learn from the present dataset.

4.4 Deep Learning model for the image segmentation

The deep learning model is constructed for the segmentation of the input image. The training image dataset and testing image dataset were fed into the CNN network to build the model. The images were preprocessed to ensure that the input images fit the neural network input. The preprocessing includes the preparation of labeled data according to the requirement. For the first experiment, labeled images of the roof, the background, and the roofline were created.

In contrast, labeled images of the roof, the background, and the gutter were prepared for the second experiment. The images of 512*512 pixels were used. The images were then passed into the neural network, built for segmentation purposes. It performs inference in the neural network model and outputs the segmentation image accordingly. Three classes were used in the input images by annotating them with three different masks: roof, rooflines, and background. Hence the prediction passing to the built model would also be of 3 similar classes. The left images below show the predicted image, an output of the model trained for 40 epochs using a categorical focal Jaccard loss and backbone resnet34. Similarly, a different dataset was prepared for the next experiment to extract the gutter segment, and the training was carried out with the same network and the same loss function.

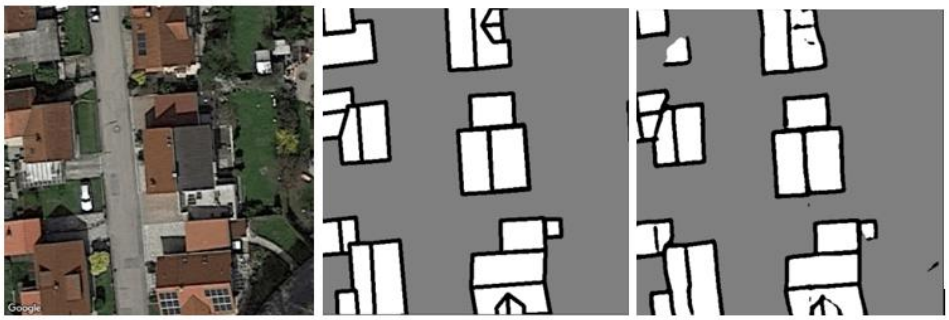


Figure 18: Example showing the Aerial Image, ground truth image, and the predicted image

4.5 Preprocessing

The images should be able to fit the neural network. Though the 512*512 pixel size of the input images would be computationally costly, the high-resolution input would preserve the spatial relationship between the image pixels. The rooflines are the most crucial class that needs to be segmented as an output, and the low-resolution input images could not fulfill our demand as they cannot segment rooflines well. Using keras, image data generators were created that produce batches of 8 images from the training dataset and train the network. We have 1374 training datasets. Hence eight images would be passed to the network in one forward/backward pass. Hence for one epoch, the training images would be sent about 172 times (1374/8), and the process

is called iteration. The default batch size in the algorithm would be 32. The model's quality will be degraded when large batch sizes are used.

4.6 Model Architectures

The various model architectures were employed to find the best for our segmentation task. The evaluation is performed if these models could effectively learn from the input datasets. The models were tested and compared to the previous studies' other models. Previous studies at Technical University Munich found that U-Net architecture played an essential role in semantic segmentation when there was less training dataset [39]. Hence U-Net was chosen with a pretrained backbone resnet34. However, the loss function was tuned to see the different predictions, which helped us evaluate the models and choose the best model for our post-processing steps.

4.6.1 U-Net

Modified versions of the original U-Net utilized in biomedical image segmentation are employed in the proposed models in this work. Four contracting stages in the encoder, four expansive steps in the decoder, four skip connections between the encoder and decoder layers, and a classification task-performing output layer made up the original model. The main difference between the ResNet-U-Net and the U-Net is that ResNet-U-Net uses the ResNet 34 model in the down-sampling encoder. The ResNet 34 model consists of 7*7 convolutional layers, a max pooling layer, and 16 total residual blocks [44]. Each residual block contains a 3*3 convolutional layer with ReLU activation and a shortcut connection. Hence, there are 34 layers in the ResNet down-sampling section. The four feature maps from the downsampling structure that correspond to the upsampling portion's image size changes [32, 64, 128, 256] are as follows: (1) output from the 7*7 convolution layer (feature map size 256*256); (2) output from the first three residual blocks (size 128*128); (3) output from the second six residual blocks (64*64); and (4) output from the third four residual blocks (32*32). Starting with the 512-channel 16*16 feature map, the upsampling phase begins. A 2*2 transposed convolution with an upsampling factor of 2 (stride = 2) is used. The upsampled output with feature map size 32*32 and 128 channels is concatenated by a convolution of 1*1 from its corresponding downsampling counterpart. The similar upsampling and corresponding concatenation takes place until it reaches the size of

256*256. Finally, the output image of 1 channel with a feature map size of 512*512 is formed from the transposed convolution of a 256-channel 256*256-sized feature map.

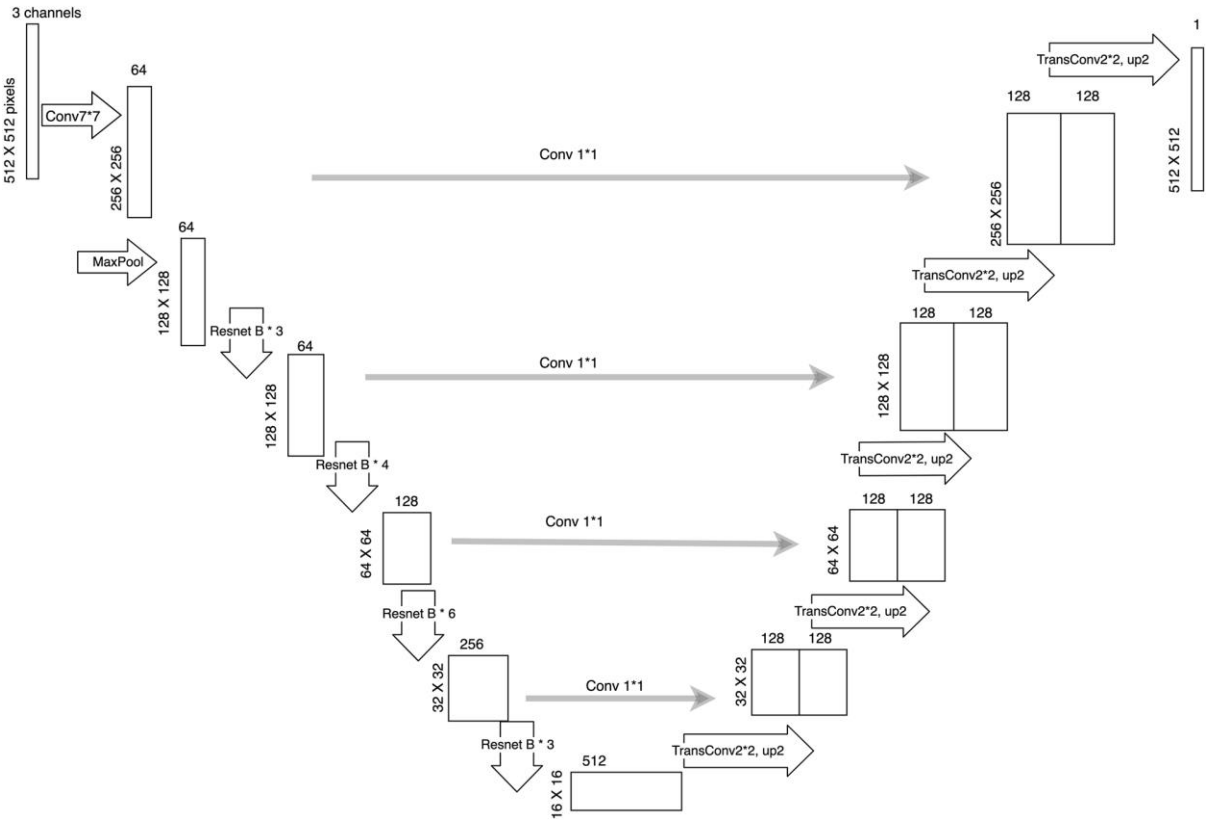


Figure 19: ResNet-U-Net structure. The ResNet-34 model takes the role of the down-sampling encoder in the U-Net where TransConv2*2, up2 is 2*2 transposed convolution with 2*2 stride [44]

4.6.2 Pretrained model

There are several pretrained segmentation models within Keras. ResNet, efficientnet, inception, mobilenet, and VGGNet are some of the pre-trained models that are commonly used. These models have all been pre-trained using the ImageNet dataset, which contains various images, to recognize over 1000 classes. These models' efficacy in learning the semantic segmentation of aerial images will be evaluated alongside the U-Net [45].

4.6.3 Resnet34

Among the pre-trained models mentioned above, the training was performed on two different models: efficientnetb2 and resnet34, keeping other parameters the same. The evaluation of the output from these models was compared with each other. Resnet34 seems to give a better prediction result than the efficientnetb2, which was used as our pretrained backbone for further work. The Resnet-34 architecture utilized inserting shortcut connections to transform a plain network into an equivalent residual network was the first ResNet architecture (see the study article here). In this instance, the convolutional networks included 3*3 filters, while the plain network was influenced by VGG neural networks (VGG-16, VGG-19). ResNets, however, are more straightforward and contain fewer filters than VGGNets. Furthermore, it adhered to two straightforward design principles: the layers had the same number of filters for the

same output feature map size, and the number of filters doubled if the feature map size was cut in half to maintain the time complexity per layer. It consisted of 34 weighted layers [46].

4.7 Loss function

The choice of the loss function is crucial in the segmentation task. The most common loss function used in the segmentation task is pixel-wise cross-entropy loss which checks each pixel individually to the ground truth. Since the cross entropy loss assigns equal weight to each pixel class by accessing the class prediction for each pixel and averaging them across all pixels. This would create a problem in the imbalanced class, which would be biased toward the class with high pixels. Hence, to solve the class imbalance problem, the other loss functions were preferred to the cross entropy loss function that would penalize the bias to the prevalent pixel. For example, in our first experiment, the classes used in the training process were roof, rooflines, and background. Hence, the background class is too high compared to the roof lines. Hence the model would be biased toward the background, and the segmentation of rooflines would be complex in this case. Different loss functions were used for the experiment while keeping other parameters constant. The impact of the loss function on the model's accuracy was studied. As we performed two major experiments: one with the roof, roofline, and background and another with the gutter added in the previous experiment. There was an extreme difference in the loss value in these experiments, which is discussed in section 5.1. The loss function experimented on were: binary cross-entropy, binary focal dice loss, binary focal loss, categorical cross-entropy, and categorical focal Jaccard loss. While looking thoroughly at the predicted images and comparing them with our eyes, categorical focal Jaccard loss gives the best prediction output. Also, the evaluation of the prediction images of different loss functions and the target images was done, discussed in chapter 5.1. The above loss is the combination of the different loss functions. For example, categorical focal Jaccard loss combines categorical focal loss and Jaccard loss [47].

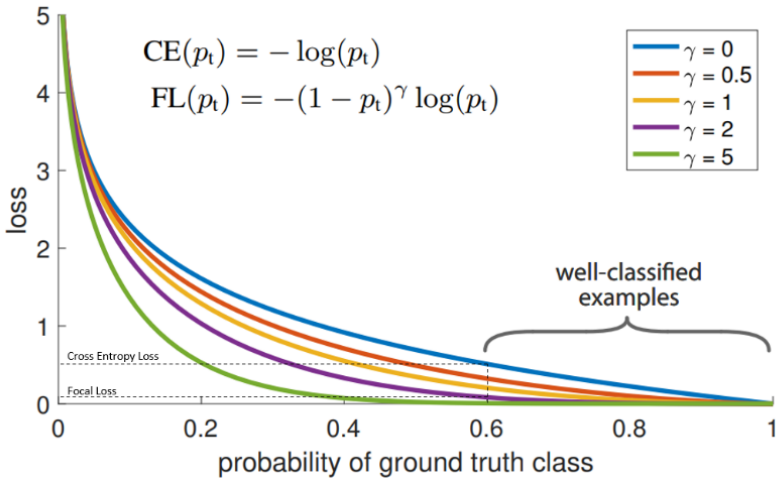


Figure 20: Illustration to show the difference between Cross-Entropy and Focal Loss and showing the fact that the performance of focal loss is better than the cross entropy [48]

4.8 Evaluation metrics

Both the IoU score and F-score with a threshold value of 0.5 have been used as evaluation metrics, and the comparison of the accuracy and their values are presented in the result section. The main reason to use IoU is to prioritize the intersection between the target image and the predicted image (True positive) and minimize the union(sum of False Positive, True Positive, and False Negative). The value of IoU ranges from 0 to 1, where 0 indicates that there is no intersection between the target image and the predicted image. At the same time, 1 represents the total overlapping of these images with each other. The IoU score is given by:

$$\mathrm{IoU} = \frac{\mathrm{TP}}{\mathrm{FP} + \mathrm{TP} + \mathrm{FN}} \tag{9}$$

where TP, FP, and FN represent True Positive, False Positive, and False Negative, respectively.

Numerically it could be written as:

$$\mathrm{IoU} = \frac{\sum \hat{y}\, y}{\sum \hat{y} + y - \hat{y}y} \tag{10}$$

$\hat{y}$ and y are the value of the prediction and target images, respectively.

Fscore, which is identical to the dice coefficient, is another statistic we used. The IoU measure generally penalizes single instances of poor classification more quantitatively than the F score, even though they agree that this occurrence is awful. Similarly to how L2 penalizes the most severe errors more than L1, the IoU metric has a "squaring" effect on errors relative to the F score. As a result, the F score tends to be closer to average performance, but the IoU value is closer to worst-case performance. Dice weights the intersection part twice as much as IoU [49].

$$\mathrm{Fscore} = \frac{2 * \mathrm{TP}}{2 * \mathrm{TP} + \mathrm{FN} + \mathrm{FP}} \tag{11}$$

The above formula is a discrete form, while we could also write a continuous form as follows:

$$\mathrm{Fscore} = \frac{2 \sum \hat{y}\, y}{\sum \hat{y} + y} \tag{12}$$

4.9 Output

A model is generated that takes a test image as an input and gives a prediction as an output. The model was trained with the training dataset and its corresponding mask and validated through the validation dataset. The model's output was stored as a hdf file which could be further used for evaluation purposes and comparison with other models. The trained model that uses different loss functions was stored individually. Later, these models were used to predict and evaluate the test images.

4.10 Model Experiment

With a description of the architecture and the hyperparameters utilized for training various networks, Table 1 specified the U-Net implementation using different architectures. By experimenting with various loss functions and backbone architecture to maximize the accuracy of the network prediction on the test dataset using the metrics provided, the final network utilized for implementation into the software pipeline was developed. The best model that gives more accuracy is further implemented for the image processing and orientation calculation mentioned in section 4.11.7. The experiments include various architectures and datasets. However, we will focus on the loss function, as previous studies conducted at TUM have already found

the best hyperparameters [39]. The architectures that were used for the experiment purpose are presented.

Table 1: Different ten network configurations on different classes with varied loss functions

Network	Dataset	No of classes	Loss function	Backbone	Encoder Weights
Network 1	Roof-centered	3 (rooflines, roof, background)	Binary Cross Entropy + Dice Loss	Resnet34	-
Network 2	Roof-centered	3 (rooflines, roof, background)	Binary Focal + Dice Loss	Resnet34	-
Network 3	Roof-centered	3 (rooflines, roof, background)	Binary Focal Loss	Resnet34	-
Network 4	Roof-centered	3 (rooflines, roof, background)	Categorical Cross Entropy	Resnet34	-
Network 5	Roof-centered	3 (rooflines, roof, background)	Categorical Focal + Jaccard Loss	Resnet34	-
Network 6	Roof-centered	4 (rooflines, roof, gutter, background)	Binary Focal + Dice Loss	Resnet34	-
Network 7	Roof-centered	4 (rooflines, roof, gutter, background)	Binary Focal + Jaccard Loss	Resnet34	-
Network 8	Roof-centered	4 (rooflines, roof, gutter, background)	Categorical Focal + Jaccard Loss	Resnet34	
Network 9	Roof-centered	2 (gutter, background)	Categorical Focal + Jaccard Loss	Resnet34	-
Network 10	Roof-centered	2 (gutter, Background)	Binary Focal + Jaccard Loss	Resnet34	

4.11 Image processing

For the 1st experiment, the labeled data consisting of roof, roofline, and background is created. The image and the labeled masks were used for training purposes. The model is created with the help of these training and validation datasets. The prediction is conducted for all the models that had been created, where 154 test images are passed to the model and output the predicted images. The predicted image has the same size as in input, i.e., 512*512 pixels. The predicted output has the same class as in the input labeled data which are classified as numbers 0, 1, and 2, rooflines, background, and roof, respectively. The orientation determination pipeline is described in the subsection below with minute details.

4.11.1 Georeference predicted images

The thesis mainly works in georeference images, which would make valid sense in comparing the output in terms of orientations and also helps in finding the rooftop area. However, the predicted image would lose this information about the georeferencing. In order to georeference these predicted images, original images with specified mask id were matched with the same mask id of the predicted images. Hence the bounding box of the predicted images would get the same geometry as in the original images.

4.11.2 Rooflines removal

The roof lines were removed and equalized with the background pixels. The reason for doing this task is to remove the gable lines, which are the roof's centerlines for each roof segment. As the output from the predicted images consists of the whole roof, our primary purpose is to extract the roof segment from the whole predicted image. The open computer vision library OpenCV [50] has been used to equalize the rooflines' pixels to the background. Now there would be only two classes, namely: roof and background, which can be seen in the figure below.

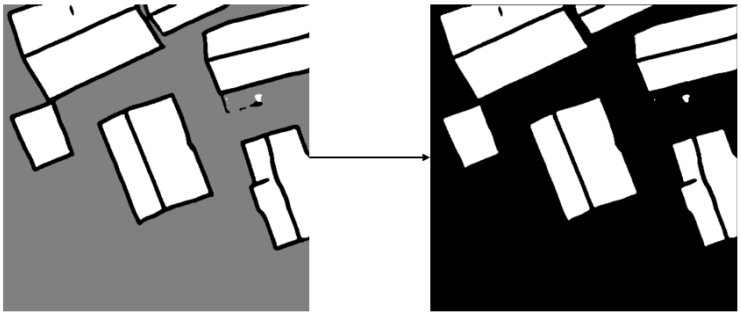


Figure 21: Example of removing the roof line and background from the predicted image to separate the roof segments

Since the segments touching the border are classed differently after processing the image, a border of 1 pixel has been placed at each corner of the output image with the same pixel as the background. This way, the roof segment touching the roof would also be classified as the centered roof. The image's small roof segment on the right depicts the original roof segment, but the model could not predict the roof segments

in that area. The small segments were removed whose areas are under some threshold value.

4.11.3 Gutter Extraction

A similar operation was performed to extract the gutter from the predicted image. All the roofline and roof pixels are equalized to the background pixel. The output in the right image shows that the gutters were extracted with the help of image processing. The classes are too imbalanced. There are large pixels of roofs and backgrounds, while the pixels of the roofline and the gutter are rare. The training could be performed on the labeled dataset, which contains only gutters, but the other masks, like the roof and rooflines, were added to improve the learning process. In this way, the neural network would learn the spatial relationship between the different classes and segment them accordingly.

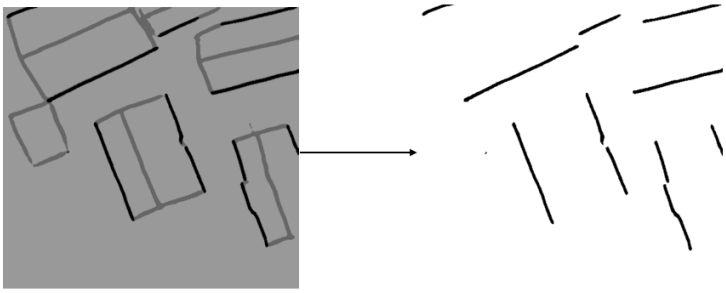


Figure 22: Removal of rooflines, roof segments and background to extract gutter from the predicted image

4.11.4 Conversion to Vector

The obtained images consist of the roof segments and the background. That means they contain only 2-pixel values in the image. Similarly, another training provides the images with gutter and background pixels. It would be, therefore, easy to perform contour detection using OpenCV [51]. A curve connecting all continuous points (along the border) with the same hue or intensity is a straightforward way to describe a contour. For the examination of shapes and the detection and identification of objects, contours are a valuable tool [51]. This way, vector data of the roof segments and gutters is obtained. The vector has been stored as a geodataframe using a python library GeoPandas[52]. It stores the vector's mask id, geometry, and label type.

4.11.5 Algorithm for Azimuth calculation

Azimuth calculation is a crucial part of this thesis. Different algorithms have been developed to assess the roof segment's correct orientation. The roof’s orientation is a vital factor in assessing the PV potential of the rooftop. Three algorithms are developed, and compared their results with the ground truth. The problem with the roof segment’s azimuth calculation is the inability to find the outer roofline. The roofs are generally tilted and oriented outward from the top. There are too few structures whose roofs are oriented inward. The outer-oriented roofline could be easily recognizable by

the human eye and perceived clearly by the human brain. However, an algorithm had to be developed to make the computer recognize the outer rooflines (gutters). Section 5.4 compares among the three algorithms to get the best algorithm.

Longest line algorithm

The vector data calculated from the processed image are irregular, meaning they have many coordinate points. For one roof segment, we expect that it consists of only four boundary lines. Since the acquired vector data has many coordinates and the centerline, sideline, and outer line could not be identified with such a massive number of geometry in a single segment vector, the vector's shape is minimized to a rectangle. We used the shapely library's function minimum rotated rectangle [53]. The rectangle we acquired from the above function would be an oriented rectangle, hence helping to preserve the shape and orientation of the geometry.

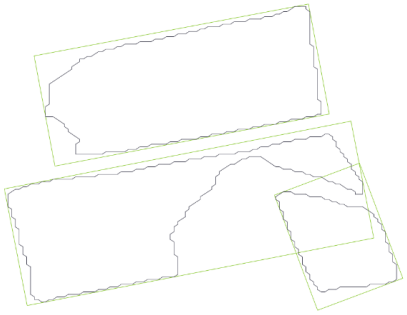


Figure 23: Conversion of the irregular roof segment to minimum rotated rectangle

The longest line method considers that the longest line is considered the outer roofline, i.e., the gutter. Considering this hypothesis, an algorithm has been developed. The algorithm takes one roof segment contained in one mask as a candidate roof whose orientation has to be determined. Then it creates a minimum rotated rectangle out of the irregular segment. The distance between them is calculated using the corner coordinates of the rectangle. The opposite sides of a rectangle are equal; therefore, the two longest lines of a rectangle are calculated. The function randomly chooses one line out of 2 lines as the longest line. The selected line serves as a gutter in this experiment. Then the selected line and the segment are sent to the function, which determines the azimuth. The hypothesis can occasionally be proven incorrect in this way since the longest line is not always the gutter line. This hypothesis fails to identify the right gutter line because it chooses the longest line randomly and cannot distinguish between the outer and center lines. This algorithm's output is shown in the section 5.4.

Farthest roof segment algorithm

Similarly to the above case, the vector data is converted to the minimum rotated rectangle, as the geometric operation is difficult to perform when too many coordinates are present in a shapefile. In this algorithm, all the roof lines were given equal priority for determining the gutter. This hypothesis would overcome the first problem

encountered above. This hypothesis considered that the short line could also be a gutter. The spatial operations are performed using the shapely function and the geodataframe from the GeoPandas library. First, the candidate shape would be selected. The centroid of four line segments of the roof segments was calculated and stored. Similarly, the centroid of all the other segments was calculated. One line segment would be taken at first, and calculate the distance between the center of the line segment and all the other segments. In this way, we could get the numerous list of distances. The minimum value of the list of the calculated distances is taken and further used to determine the gutter. Similarly, a similar operation is performed for all the line segments of the candidate roof segment. The maximum value of all the minimum distances is then calculated. The line segment which gives this maximum value of the minimum is considered a gutter in this experiment.

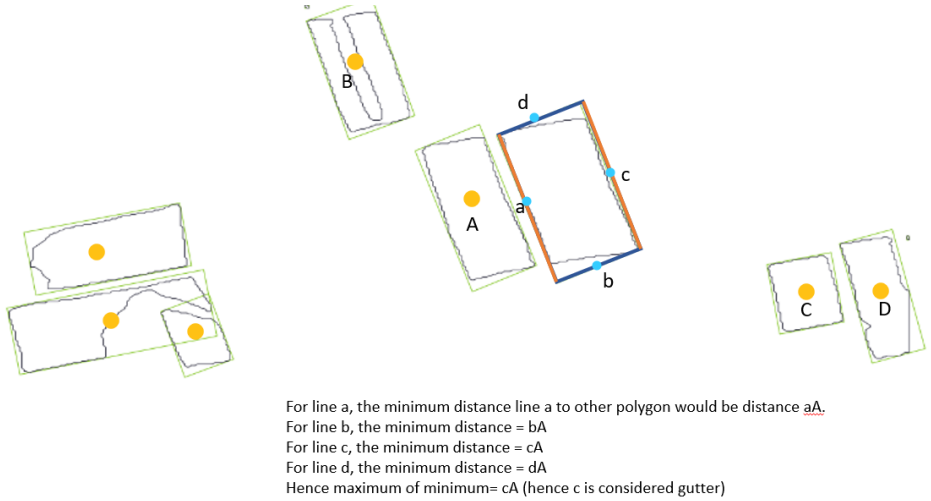


Figure 24: Figure showing the working method of the Farthest roof segment algorithm

An algorithm also experimented with taking a line instead of the centroid of a line. In the above figure, cA seems to be the maximum of the minimum distances. The algorithm fails in some cases, like the three roof segment case, where three roof segments are in a single roof. Another case of failure of this algorithm is the edge roof case.

Nearest gutter algorithm

The Nearest gutter algorithm is computationally expensive as new training has to be repeated for gutter segmentation. Similar to the roof segmented from the model, the gutter lines were also segmented similarly. The image processing steps have already been mentioned above. The image is vectorized similarly to the above steps and then converted to a geodataframe. Thus obtained geodataframe would be used to extract the correct gutter for the azimuth calculation. Both roof segment geodataframe and gutter geodataframe would be used in this algorithm. This algorithm uses a similar approach as in the farthest roof segment algorithm. The difference between them is that it uses minimum of minimum to determine the gutter. Instead of using the roof segment polygon, it uses the gutter and calculates the distance between the line

segments and the gutters in the mask id. For the first line segment, it calculates the distance between the line segment's centroid and the gutters present around the segment. The smallest distance for the first line is stored. The looping process is repeated for the remaining four lines. Now the minimum of the minimum distances determines the actual gutter in this algorithm. Suppose there is an absence of the predicted gutter segment of that particular roof segment. In that case, the algorithm fails as it tends to take the minimum distances and considers the false gutter as its nearest gutter. Also, the algorithm fails when there is a gutter on all sides. Confusion is created when there is more than one predicted gutter for a particular roof segment. Here the minimum rotated rectangle and the vectorized predicted gutter have been used for the depiction. The analysis of its correctness has been discussed in the result and evaluation section.

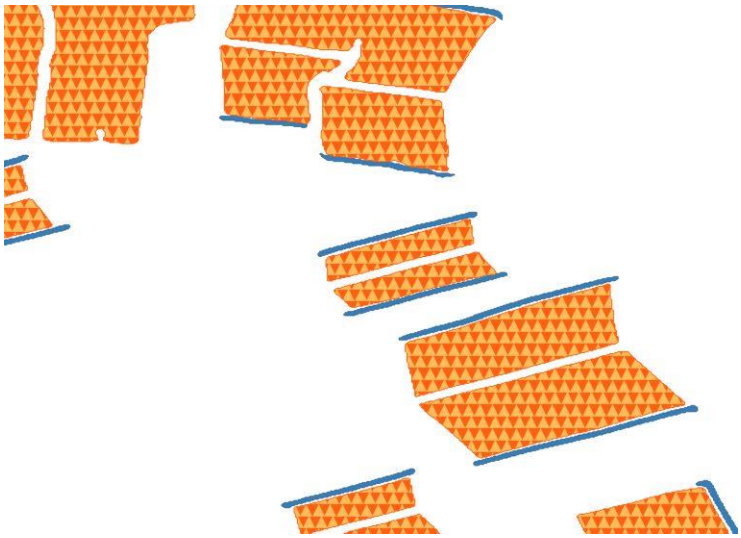


Figure 25: Illustration of predicted roof segments and gutter

4.11.6 Azimuth calculation

The azimuth calculation task that RID has adapted requires the gutter (outer line) and the corresponding roof segment. Most of the roofs in the world are oriented outward so that the roof orientation would be calculated according to the roof's outer line. The above task gives the gutter and roof segments required for the azimuth calculation algorithm. The angle is calculated using a formula:

$$angle\ (\theta) = (\tan^{-1}\frac{y}{x}) * \frac{180}{\pi} \quad (13)$$

The distribution of the ground truth azimuth of the roof segment is shown in Figure 26. It shows that the roofs with azimuth around 150° and 320° have the highest number.

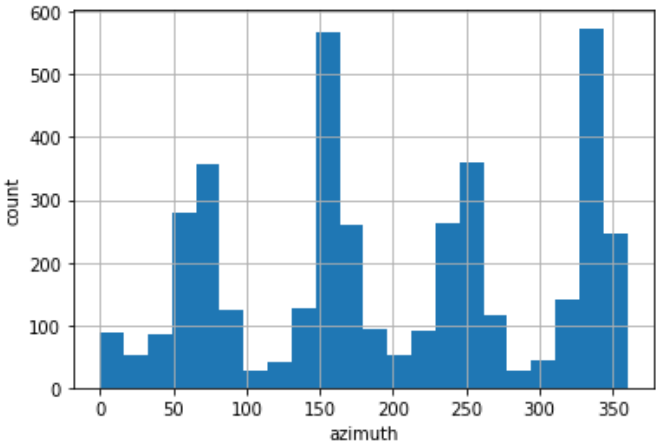


Figure 26: Distribution of the whole dataset roof's azimuth (Training, Validation, and Testing dataset)

The nearest point from the centroid of the segment to the gutter is calculated. The nearest point would be formed when the centroid touches the line perpendicularly. The difference between the points of the perpendicular lines Δx and Δy of the perpendicular to adjust azimuth according to the direction of the perpendicular vector. The calculated azimuth is now used for the orientation determination. The result of this method of azimuth calculation is sent to the function prepared for the orientation determination because it might be impossible to get the predicted gutter for every roof segment, above technique of taking the gutter line from the segment itself was adapted. The technique used by the RID paper [42] is different and requires the extension of the gutters to meet the requirement. Hence a new algorithm is developed in order to fulfill the needs. The following steps take place during the azimuth calculation:

The perpendicular line to the gutter is created. Hence the angle of the perpendicular line will be calculated instead. The following reason illustrates why perpendicular lines are created for the azimuth calculation.

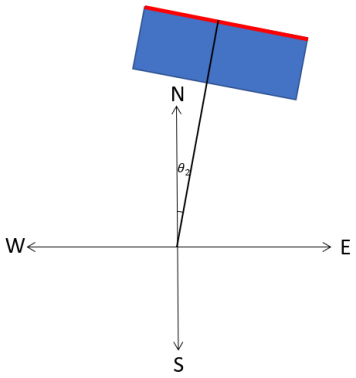


Figure 27: Figure showing the concept of perpendicular to the gutter to calculate the azimuth

The perpendicular line was generated to make a calculation easier and more understandable. In the above figure, the blue line is the roof segment, and the red line depicts the gutter of the roof segment. Hence the orientation of the roof is determined using the red line of the roof segment. When finding the angle of the gutter line (red line), the coordinate of the extreme points was taken, and the angle was calculated using equation (14). Somehow the angle line is directed towards NW and SE. However, the actual direction the roof is facing is North (or NE). We are trying to calculate the gutter's facing direction rather than the gutter line's direction, which ultimately calculates the face direction of the roof segment, which could be easily visualized in the form of a map. It needs another mathematics again for the conversion. The concept of perpendicular solves this problem where the algorithm takes the gutter and creates a copy of the gutter at some distance in the same direction as 'right'. It calculates the centroid of the original gutter and the copied gutter, and then a LineString is generated using two centroid points. This LineString acts as a perpendicular line and is used to calculate azimuth.

There are two cases of how the angle is defined: 1) Reference with the x-axis (East-West), and 2) Reference with Y-axis (North-South), as shown in the figure below. The formulas give the angle of the line:

In Figure 28 (a),

$$angle\ (\theta_1) = (\tan^{-1}\frac{e}{n}) * \frac{180}{\pi} \qquad (14)$$

Similarly, for Figure 28(b), if the angle with reference to the N-S direction is considered, the angle of the line is given by

$$angle\ (\theta_2) = (\tan^{-1}\frac{n}{e}) * \frac{180}{\pi} \qquad (15)$$

Hence we could conclude that they are supplementary to each other. That means the sum of these angles is equal to 90°. We could use this trick to calculate the quadrantal bearing mentioned in this section.

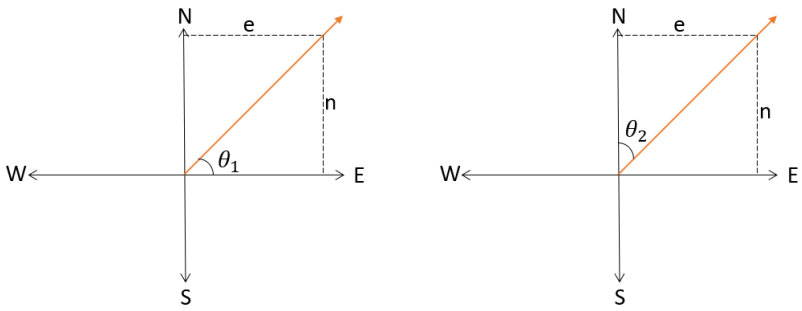


Figure 28: Figure (a) shows the angle θ_1 with respect to x-axis, and figure (b) shows θ_2 with respect the y-axis

The newly developed algorithm calculates the angle with respect to the N-S axis. The angle calculated would be θ_2. Thus calculated angle would be the azimuth for that line only if the angle is positive. If the calculated angle is negative, 360° has to be added to the angle to get the azimuth. If the calculated angle is negative, the angle will be anti-clockwise. Hence we have to add 360° to the negative angle to find the correct azimuth. That means if the calculated angle is -93.45°, then the azimuth of the line will be 360°+ (-93.45°), which is equal to 266.54°, which is a value less than 360°. It is also known as Whole Circle Bearing (W.C.B), measured with the north in a clockwise direction. The value of W.C.B ranges from 0° to 360°.

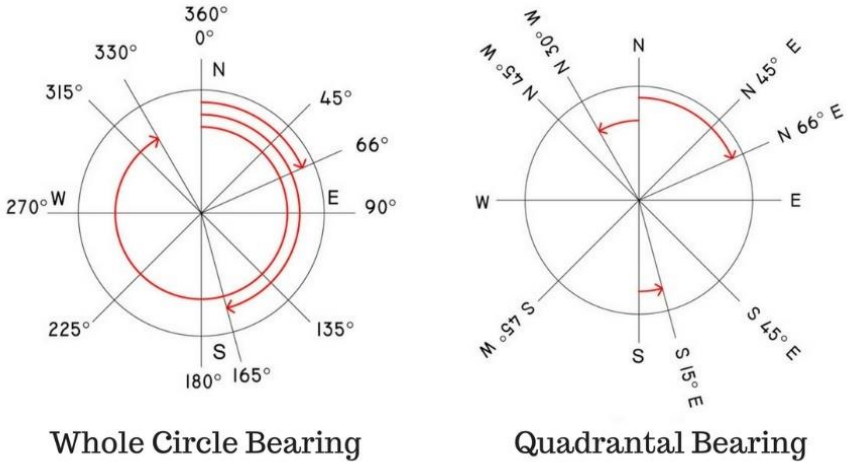


Figure 29: Illustration of the difference between Whole Circle Bearing and Quadrantal Bearing[54]

The whole circle bearing is converted to the Reduced Bearing or Quadrant, described in the next section of orientation determination.

4.11.7 Orientation determination

Determination of the orientation of the segment is the most critical aspect of this thesis. The thesis is focused on determining the roof's orientation using the neural network's output. A different approach was used in the previous study conducted at TUM. The different orientations were labeled with 16 different orientation classes and one flat class with 17 total classes [42]. The model is trained on the labeled data of 17 classes, and the predictions are determined. It would be a lengthy and costly process as it requires a manual label and 17 different orientation has to be calculated and labeled properly. However, this approach does not need 17 different labeled datasets. The only data that we require for the orientation determination is the gutter. According to the gutter of the roof, orientation is determined. The approach developed in this thesis is very cost-effective and simple to execute. First, the approach from the RID paper is tried out. The azimuth is calculated using the gutter and the segment and then passed to calculate the orientation. The different azimuth values are classified with different orientation classes as follows:

Table 2: Table containing different orientation classes with their azimuth range [42]

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

After the calculation of the azimuth of each gutter, the azimuth is classified into the different orientation categories as in the figure. A total of 18 classes is prepared, out of which Background and Flat are excluded for orientation determination. The division is carried out into different orientations by dividing the 360° range into 22.5° slices and categorizing them as N, NNE, NE, ENE, E, ESE, SE, SSE, S, SSW, SW, WSW, W, WNW, NW, and NNW [42]. The orientation calculated from this method is stored in the corresponding roof segment as a geodataframe supported by a python library GeoPandas.

The algorithm developed in this thesis is very similar to the RID paper except for some concepts and mathematical improvements. The azimuth (Whole Circle Bearing) of the perpendicular line thus calculated is used for the calculation of quadrantal bearing, which is also called reduced bearing. The orientation will be classified into 16 classes: N, NNE, NE, ENE, E, ESE, SE, SSE, S, SSW, SW, WSW, W, WNW, NW, and NNW.

The classification of the azimuth and the developed code's directions parameters are the same as the above classification in the RID paper, except for the second column, which represents North. The minimum and maximum values of azimuth for the second column were inputted as 348.75 and 11.25, respectively. The boolean ‘OR’ is used to adapt the code accordingly only in the case of North. The code is written using the

'AND' operator between the minimum and maximum values for other calculations. For example, a sample code is shown below.

```
if azimuth >= 348.75 or azimuth < 11.25:
        direction = 'N'
elif azimuth >= 11.25 and azimuth < 33.75:
        direction = 'NNE'
```

The whole circle bearing is already known; hence there is no need to find the quadrantal bearing. The aim is not to find the exact angle like S45°N but to find the quadrant SN. So, there is no need for further conversion of W.C.B to Q.B., but the quadrant is divided into several directions to find the precise orientation. Table 2 can be used and expanded for 16 different directions with their associate angle for their axis:

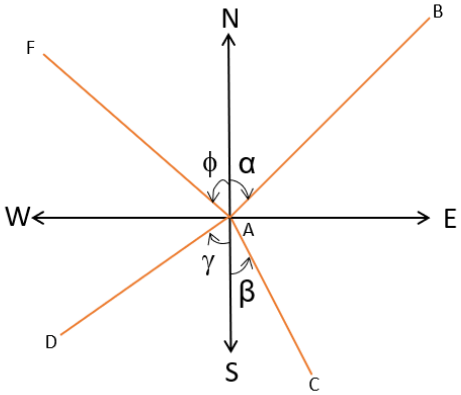


Figure 30: Illustration of reduced bearing and quadrantal bearing, which are always calculated from the NS axis

Table 3: Table showing the conversion between WCB and reduced bearing to get the quadrantal bearing [55]

Line	W.C.B. between	Reduced Bearing	Quadrant
AB	0° and 90°	R.B.= W.C.B	NE
AC	90° and 180°	R.B.= 180°- W.C.B.	SE
AD	180° and 270°	R.B.= W.C.B.- 180°	SW
AF	270° and 360°	R.B.= 360°- W.C.B.	NW

Other quadrants and reduced bearing can be calculated using the above approach. However, the thesis aims to find an easy way to compare the orientation between the ground truth and predicted images in quadrant form, which is generated from the deep neural network.

5 Results and Discussions

The networks with different architectures were trained, which resulted in a range of accuracy in the model. This section discusses different network architecture, their accuracy, and the result in the final orientation. The main aim of the experiment is to extract the roof segment and the line segment (gutter) from the roof image. Most of the experiments were done on U-Net with different loss functions keeping the other parameter the same. In the first experiment, we discuss different loss functions and their effect on the trained model for the roof segment extraction. The second experiment consists of gutter extraction, in which two different pieces of training were performed. The first training in the second experiment was done with four classes: rooflines, roof, gutter, and background, while the second training included only two classes: gutter and background. Each step of the learning process for both the training and validation datasets involved monitoring the model's performance using the loss function. The validation dataset calibrates the effectiveness and reduces overfitting during the model [56]. The results of these experiments are analyzed with the training curves and the evaluation metrics.

5.1 Training Result

The result of the neural network training performed in the network mentioned in Table 1 is presented in this section. The training was performed on nine different networks, and their evaluations were carried out. The IoU of the roof, roofline, and the mean IOU are presented in Table 4, calculated on each network, and an extra class ‘gutter’ is added in networks 6, 7, and 8 for the gutter extraction. Similarly, networks 9 and 10 were trained using two classes: gutter and background. The training result on different loss functions on all these nine networks is presented in the following table, along with the mean IoU validation score and F1-Score.

Table 4: Experiment table showing mean IoU and F1-Score on the validation dataset on n different networks with different parameters

Network	Epochs	Number of classes	Loss Function	Mean IoU on Validation Dataset	F1-Score
Network 1	40	3	BCE + DL	80.76%	88.46%
Network 2	40	3	BF + DL	78.99%	88.46%
Network 3	40	3	BF	69.58%	79.49%
Network 4	40	3	CCE	77.04%	85.76%
Network 5	40	3	CF + JL	82.86%	89.87%
Network 6	40	4	BF + DL	69%	80.11%
Network 7	40	4	BF + JL	67%	77.46%
Network 8	40	4	CF + JL	69.91%	80%
Network 9	60	2	CF + JL	25%	33.23%
Network 10	60	2	BF + DL	2%	3%

5.1.1 Summary of Training Result

The first five network consists of 3 classes: roofline, roof, and background. The training was performed on 40 epochs for every experiment. A combination of Binary Cross Entropy and Dice Loss (BCE + DL), Binary Focal and Dice Loss (BF + DL), Binary Focal (BF), Categorical Cross Entropy (CCE), and a combination of Categorical Focal and Jaccard Loss (CF + JL) were respectively used as a loss function for five different networks mentioned in Table 4. The padding is not added to the ground truth to preserve the spatial relationship between these three classes. The main aim of these networks is to extract the roof and roofline information. Background pixels would also help the network learn the spatial relationship and help distinguish between the roofline and background pixels. Mean IoU of 80.76% and F1-Score of 88.46% were acquired during the training network 1.

Compared to network 2, the mean IoU and F1- Score was 78.99% and 88.46%. The validation scores further decreased in network 3 with a mean IoU of 69.58% and F1-Score of 79.49%, which is the least validation score in the first five experiments. Network 4 shows the mean IoU and F1-Score of 77.04% and 85.76%, respectively. From Table 4, it is assured that network 5 gives the best result among the first five experiments performed on three classes with the highest mean IoU and F1-Score value. The mean IoU and F1-Score for network 5 were 82.86% and 89.87%, respectively. Hence, the predicted images from network five would be used for the roof separation task. The three networks, 6, 7, and 8, were trained with four classes: roofline, roof, gutter, and background. The hyperparameters in these networks were not changed except for the loss function and number of classes. When networks using three and four classes are examined, it is shown that the network using four classes has a much lower validation score. The highest validation score, corresponding to a network 8, is determined to be 69.91% and 80% as mean IoU and F1-Score, respectively. Network 8 is utilized for gutter extraction after being used for output prediction since it outperforms networks 6 and 7 in terms of validation scores. Another network, 9 and 10, that used only two classes: gutter and background was used for experiment purposes. The primary purpose of this network is to extract the gutter segment without further image processing steps. However, when the networks were trained, the validation score was too low to perform the prediction. Therefore, the last two networks were not even employed to produce prediction images.

5.1.2 Prediction result

A prediction result is presented in this section. The first eight networks show the prediction result along with its corresponding aerial image (left) and the ground truth labeled image (middle). The last two networks: network nine and network ten, were not included in the prediction output as the validation score is already too low for prediction image generation. The images on the left show the image with the highest mean IoU on the network, while the right with the least mean IoU. For the first five networks, it is observed that network one and five show the best prediction result when visually inspected. The lowest IoU in all the networks is due to the different obstructions. The trees in the aerial image hide some essential features like roofs and rooflines. It is evident that more trees are blocking the roof structure in the result on the right with the lowest mean IoU score than in the result on the left with the highest mean IoU. Additionally, it is discovered that the network finds it challenging to learn about more complicated roof structures. When compared between the ground truth and predicted images, it appears that some ground truth data were not adequately annotated, leading to poor IoU.

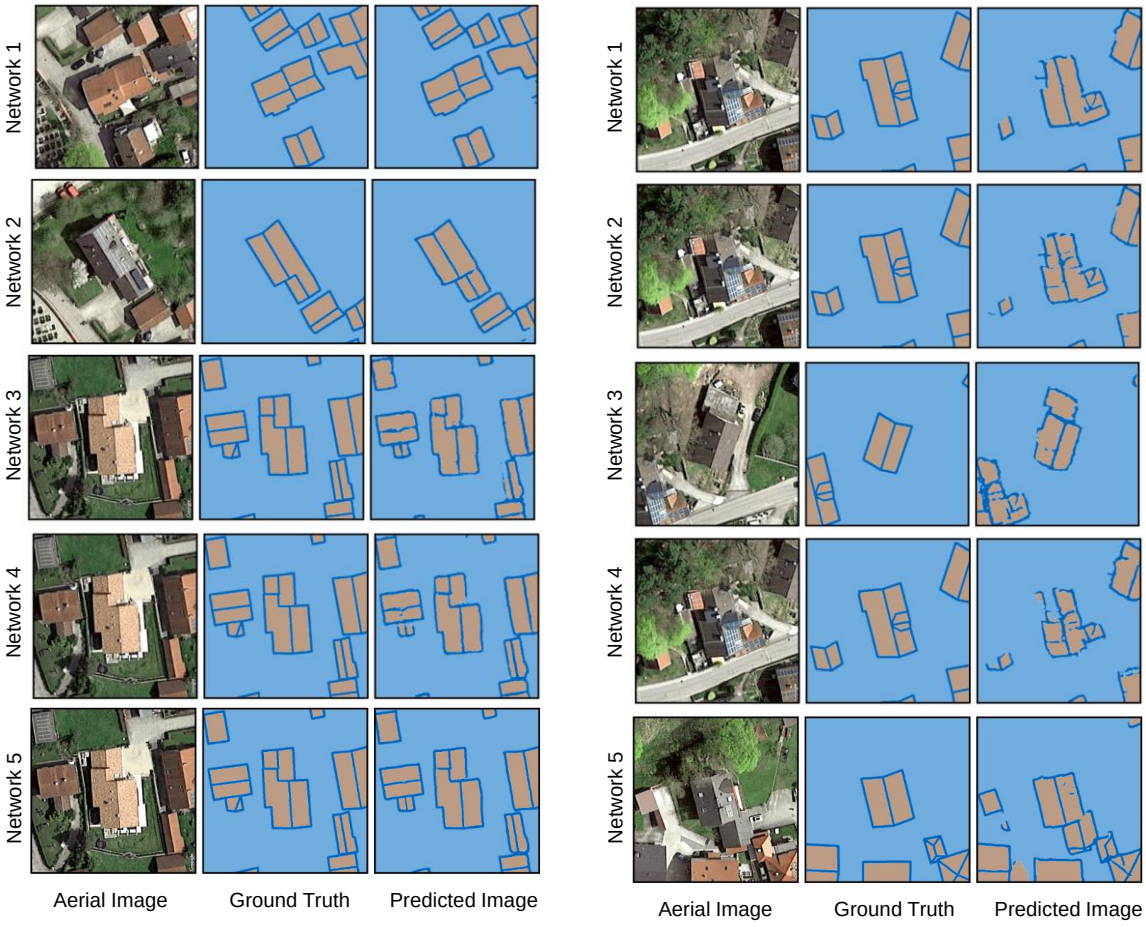


Figure 31: Aerial image, ground truth image, and predicted image on first five with highest mean IoU (right) and lowest mean IoU (right). Roofline (dark blue), roof (light orange), and background (sky blue) are the classes in the first five network

When reviewing the predictions of the following three networks, it appears that the images with the highest mean IoU are the same in all three networks. Similarly, the

lowest mean IoU was acquired in the same image for all three networks: networks 6, 7, and 8.

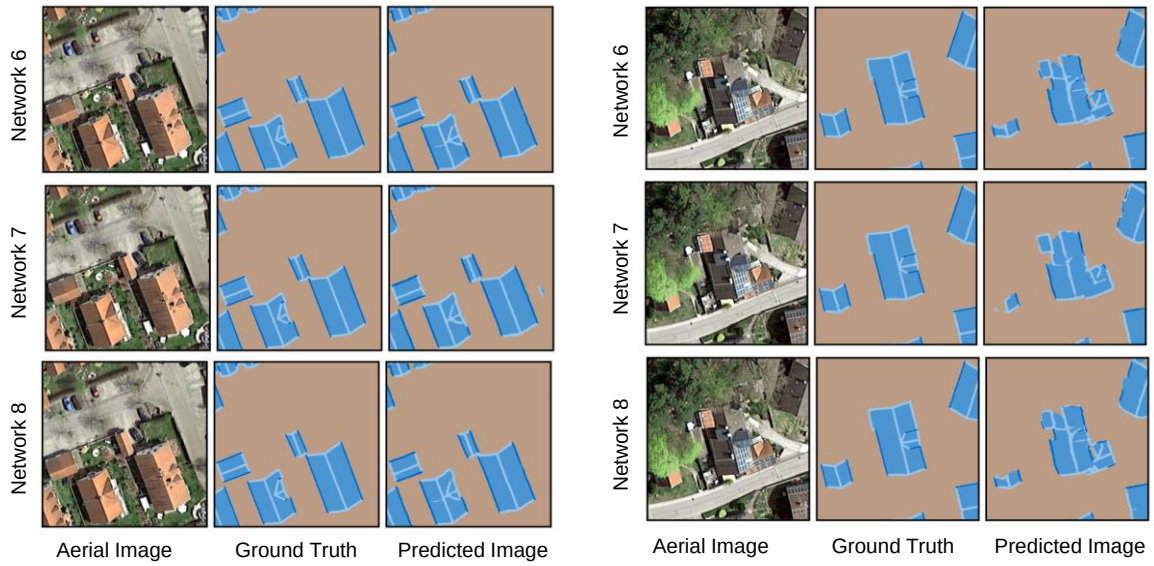


Figure 32: Aerial image, ground truth image, and predicted image on networks 6, 7, and 8 with highest mean IoU (right) and lowest mean IoU (right). Roofline (sky blue), roof (light blue), gutter(dark blue) background (orange) are the classes in the network 6,7, and 8

Table 5 shows the result of the testing dataset. The predicted image is compared with the test image, and its IoU is calculated. When the IoU for a particular class in network one is calculated, for instance, the IoU value for Roofline is discovered to be 35%, which is the lowest of all the images in the test dataset. The highest IoU in the roofline class is found to be 0.78. Similarly, the minimum and maximum IoU for the roof class is 85% and 99%, respectively. The table shows that the roofline is not well segmented concerning other classes, as their accuracy is too low when compared. It is also clearly seen from Figure 31 that the roofline is not segmented well.

Similarly, when discussing the best network for roof segment extraction, network 5 gives the highest IoU with 0.8, 0.99, and 0.91 on the roofline, roof classes, and mean, respectively. In order to extract the separate roof segment, the most important class is considered rooflines. The value for the highest IoU for the roofline was found to be 0.8 in network five among the first five networks. Additionally, it is discovered that even in the lowest IoU value, it provides the greatest value of 0.39 among the first five networks. When the mean IoU is compared, network 5 has the highest mean IoU. Network 5's lowest and maximum roof IoU are 0.89 and 0.99, respectively. It seems that the roof class has been predicted well in the network.

Since networks 6, 7, and 8 were trained specifically for the gutter extraction, the comparison for gutter IoU is performed. For the gutter class, network 8 has the lowest IoU of 0.30, and the highest IoU of 0.69 compared to the last three networks in the table and is considered the best IoU. Networks 6 and 7 give the highest IoU of 0.63 and 0.64, respectively, but network 8 has a better gutter IoU than networks 6 and 7, with a value of 0.69. Network 6 performs well in terms of roof class IoU, with a value

of 0.94; the gutter is prioritized higher in this experiment; hence, network eight is selected for further post-processing.

Table 5: Table showing the lowest IoU and highest IoU of the predicted image on a different network

Network	Class	Lowest IoU value	Highest IoU value
Network 1	Roofline	0.35	0.78
	Roof	0.85	0.99
	Mean	0.64	0.90
Network 2	Roofline	0.32	0.74
	Roof	0.85	0.99
	Mean	0.60	0.88
Network 3	Roofline	0.26	0.65
	Roof	0.83	0.99
	Mean	0.53	0.84
Network 4	Roofline	0.3	0.70
	Roof	0.91	0.97
	Mean	0.6	0.86
Network 5	Roofline	0.39	0.80
	Roof	0.89	0.99
	Mean	0.67	0.91
Network 6	Roofline	0.15	0.68
	Roof	0.63	0.94
	Gutter	0.30	0.64
	Mean	0.53	0.80
Network 7	Roofline	0.06	0.65
	Roof	0.51	0.93
	Gutter	0.30	0.63
	Mean	0.51	0.78
Network 8	Roofline	0.17	0.67
	Roof	0.67	0.93
	Gutter	0.30	0.69
	Mean	0.52	0.80

5.1.3 Training curve

The learning curve of the highest and lowest validation score on the first five networks is shown in Figure 35. The lack of a sufficient training dataset compared to the validation dataset might be the reason for the gap between the training curve and the validation curve. As seen in the left figure's training loss and validation curve, the network training and validation losses were declining, and the model could still learn from the dataset. However, it appears that the model overfits when compared with the outcome (on the right) since the validation loss is not decreasing but instead increasing after a particular epoch. A better prediction could not be expected in the overfitted network. The training curve for all the other networks is presented in chapter Appendices.

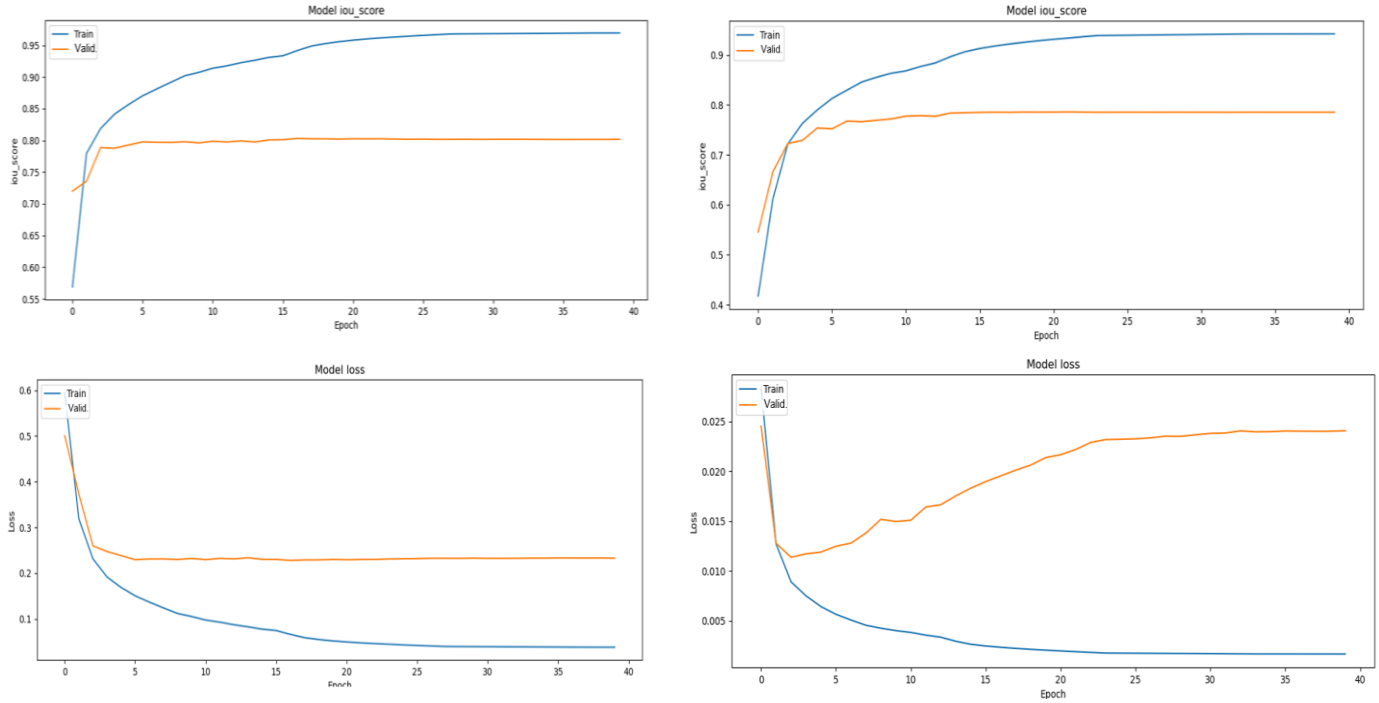


Figure 33: Training curve on network 5 and network 3 showing the comparison between two networks (3 classes)

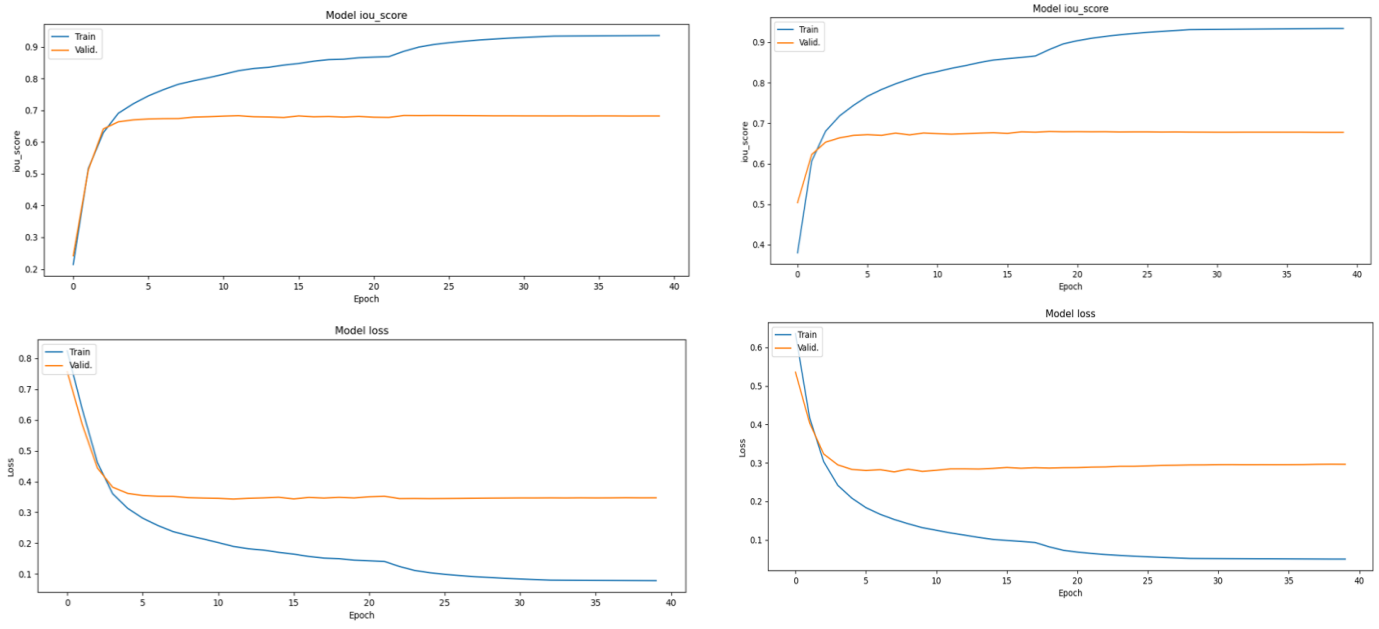


Figure 34: Training curve on network 8 and 6 showing the comparison between the two networks (4 classes)

5.2 Best Configuration Selection

The model with the best configuration is selected and presented based on the evaluation of the metrics. Table 6 presents the summary of the network configuration with the evaluation metrics obtained from the model.

Table 6: Best network Configuration Selection with its IoU-Score and F-Score

Network no	Number of classes	Loss Function	IoU-Score	F1-Score
Network 1	3	BCE + DL	80.76%	88.46%
Network 2	3	BF + DL	78.99%	88.46%
Network 5	3	CF + JL	82.86%	89.87%
Network 8	4	CF + JL	69.91%	80%

5.3 Confusion Matrix

For the performance evaluation of the classification of the pixels used, a confusion matrix is prepared. It is also known as an error matrix. It helps to determine if a prediction model predicts the test dataset accurately or not by comparing it with the ground truth data.

Confusion matrices for each model tested on the test datasets have been calculated below. From the confusion matrix, we could see the actual and predicted label, and the misclassification of the data could be realized. From the confusion matrix created between the ground truth and the predicted label, it is clear that the roofline gets misclassified the most. The reason behind the misclassification might be the smaller pixel value in the dataset. Due to the image containing the background and the image having a large pixel portion, the model gets biased to the higher pixel. In all the confusion matrices calculated for all the networks mentioned above, we could see that network 5 gives a promising result.

The confusion matrix's each row contains the instance of the ground truth, which is considered the true value. The instances contained in the confusion matrix column, known as a predicted value, are compared against the ground truth. The comparison could be made in the confusion matrix to get the best network out of the mentioned nine networks. When discussed on network 1, which has used Binary Cross Entropy and Dice loss as a loss function, the confusion matrix shows that the roofline that the model predicts is true compared to the ground truth with an accuracy of 77%. The first diagonal element of the matrix shows the true classification known as True Positive. The second column of the first row, with a value of 0.11, shows that the roofline has been falsely classified as a roof.

Similarly, the roofline has been misclassified as a background with an accuracy of 13%. A normalized confusion matrix is prepared to make it more understandable. The total sum of all the rows and columns should be equal to one. We could also find the total accuracy of the classification by performing the accuracy test. The accuracy is calculated using a formula:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \quad (16)$$

TP, TN, FP, and FF are True Positive, True Negative, False Positive, and False Negative, respectively. We got a mean IoU of prediction value against a ground truth equal to 88.7% from the calculation. Similarly, if we perform a similar calculation on network 5, we get an accuracy value of 90.03%. By calculating the IoU of all the networks, we found that the accuracy of network 5 is higher than other networks. Hence, the predicted image dataset we acquired using the model with network architecture mentioned in network five is used to calculate azimuth and orientation. The first experiment was carried out with three classes to extract rooflines and the roof segment. To get an excellent result for the roof segment, the roofline is considered a superior class to the roof and background. If the rooflines were correctly segmented, the roofs could be separated by using that roofline in the image processing steps. However, if the rooflines were not well classified, the roof segments could not be separated, and our goal could not be achieved. Thus, focusing on the roofline, the True positive value of all the networks in increasing order: 0.68 (Binary Focal Loss),0.72 (Binary Focal + Dice Loss), 0.72 (Categorical Cross Entropy), 0.77 (Binary Cross Entropy + Dice Loss), 0.79 (Categorical Focal + Jaccard Loss) were observed.

Similarly, the true positive value for the roof is high in all the networks exceeding 98%. The background true positive rate is highest in network 5's prediction with a value of 93%. Since the true positive value for all the classes: rooflines, roof, and background, are the highest compared to the other network, the predicted image from network 5 is taken for further post-processing. All of the work mentioned above is done for the roof segmentation. However, to extract the gutter essential for the orientation determination, a new network was developed with 4 classes (roofline, roof, gutter, and gutter) and trained in 3 different configurations. The confusion matrix of the predicted mask with respect to the ground truth is shown in Figure 37. The three loss functions were experimented with, and the result for all three networks gave the expected result. The predicted output from network eight was sent for further image processing to extract the gutter and, ultimately, the azimuth calculation.

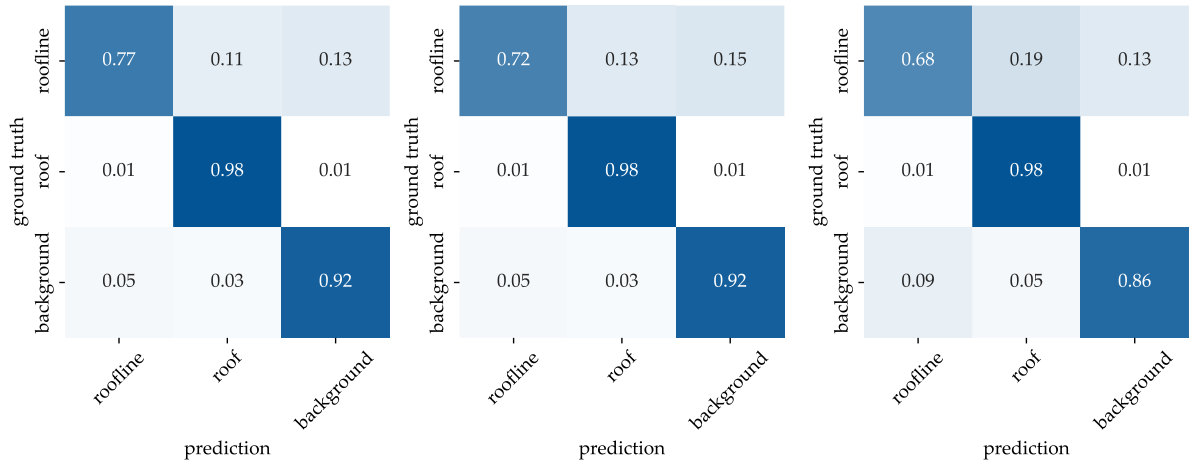


Figure 35: Confusion matrix between the ground truth and the prediction (3 classes) on network 1 (left), network 2 (middle), and network 3 (right)

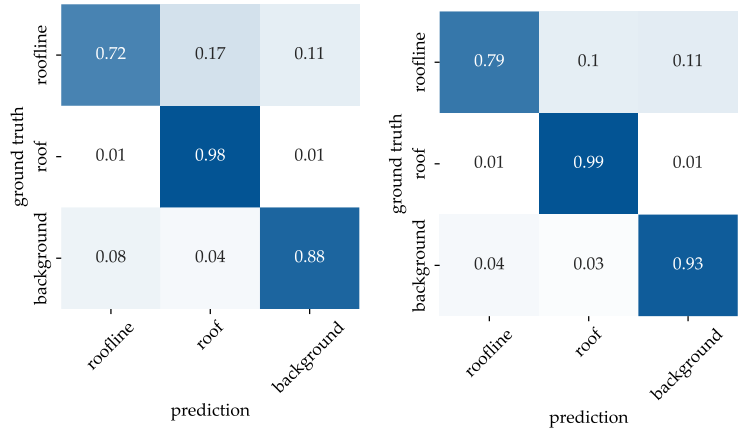


Figure 36: Confusion matrix between the ground truth and the prediction (3 classes) on network 4 (left), network 5 (right)

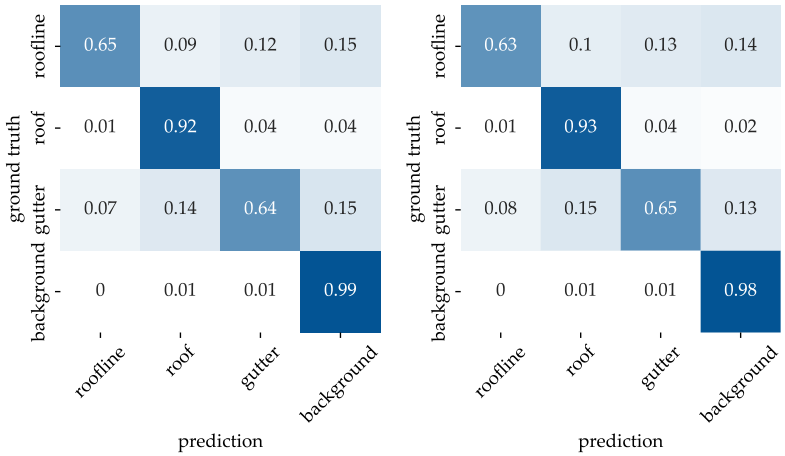


Figure 37: Confusion matrix between the ground truth and the prediction (4 classes) on network 6 (left) and network 8 (right)

5.4 Orientation Result and Evaluation

This section includes the result of the roof segments' orientation using the algorithm mentioned above in section 4.11.5. The result using the longest line algorithm is presented in Figure 38: Confusion matrix between the roof's orientation of the ground truth (nearest gutter algorithm) and the prediction (longest line algorithm) for 16 different orientations. It uses the longest line algorithm for the predicted dataset and nearest gutter algorithm for the ground truth dataset to determine their orientations and prepare a comparison matrix. The ground truth dataset consists of the annotated roof segments performed by removing the rooflines from the whole roof. The predicted dataset contains the predicted roof segments, which contain the output of the test dataset predicted from network 5. With the help of 2 datasets, the longest line algorithm and nearest gutter algorithm, an orientation result and the comparison are presented. A map depicting the orientation result of the predicted image is presented in Figure 45. A ground truth dataset is used for comparison with the predicted orientation, which is presented as a confusion matrix. A normalized confusion matrix is shown in the figure below, where each value depicts the probabilistic value. The longest line is taken as the gutter, but finding the longest line is still tricky after converting the segment to the minimum rotated rectangle. The two lines are opposite, equal in length, and could not be figured out as the correct longest length. Hence there is a high chance of the centreline being supposed to be a gutter. The 16 different orientation classes were classified according to the gutter's orientation, as in section 4.11.7. The results show the chance of failure on the opposite axis. Among the predicted data, most roofs are oriented southern side, with 449 roofs oriented on the southern side and the least 18 roofs oriented to the NW. The confusion matrix shows the normalized value of the probability of classification.

The overall classification accuracy is 38.52%, with the highest accuracy in the classification of NW (67%) and the least in SE (19%). The confusion matrix's diagonal element shows the classification accuracy of each class. The anti-diagonal matrix values are also high compared to the non-diagonal elements. This shows that there is a relation between these diagonal elements and anti-diagonal elements. For example, it could be seen that the East has been chiefly misclassified as the West and vice versa. It means there is confusion in classification between the East and West and North and South, which are the vertically opposite axis. Different algorithm for the azimuth calculation and roof orientation has been carried out to solve this problem that is persisting in classifying the orientation.

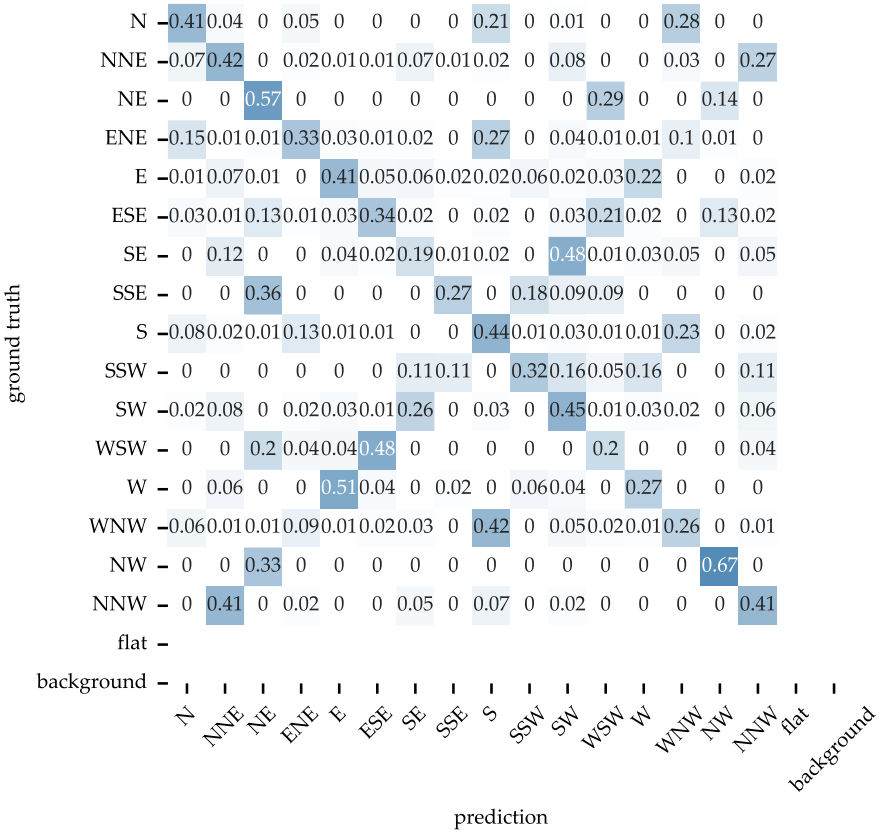


Figure 38: Confusion matrix between the roof's orientation of the ground truth (nearest gutter algorithm) and the prediction (longest line algorithm) for 16 different orientations

The farthest roof segment algorithm has been applied to compare the result with the longest line method. The longest line algorithm has one false supposition that the gutter is always the longest line. Hence, to overcome this problem, a new approach has been described in the Farthest roof segment algorithm section, and the result is presented in this section. The result shows the confusion matrix between the classification of orientation between the predicted dataset and the test dataset (ground truth). The confusion matrix is created between each roof's orientation for predicted and test datasets. The farthest roof segment algorithm is used for the predicted dataset, while the nearest gutter algorithm is used for the ground truth so that the result can be comparable. The ground truth data by the nearest gutter algorithm are taken as true data for comparison purposes, though they might not depict the correct information. The predicted dataset has more roof segments than the annotated dataset; hence, a new geometric function from the GeoPandas library was implemented. The overlay of the geometries with intersection was carried out so that the data frame would contain only one geometry and the orientation determined by two algorithms. For example, the data frame stores the orientation using the farthest roof segment algorithm in a new column direction1 and the nearest gutter algorithm (ground truth) in a new column direction2.

Moreover, a confusion matrix between direction1 and direction2 is created. The maximum number of roofs is discovered to be oriented toward the East, with a total number of 343. The lowest counts for SE with a value of 26. Compared to the previous

algorithm, the maximum number of orientations was toward the South in the previous experiment. The accuracy of the confusion matrix is only 14.9% which is very low. Since the confusion matrix is created between predicted and ground truth, with the ground truth orientation calculated by the same algorithm, ‘nearest gutter algorithm’ it can be concluded that the longest line algorithm is better than the farthest roof segment algorithm.

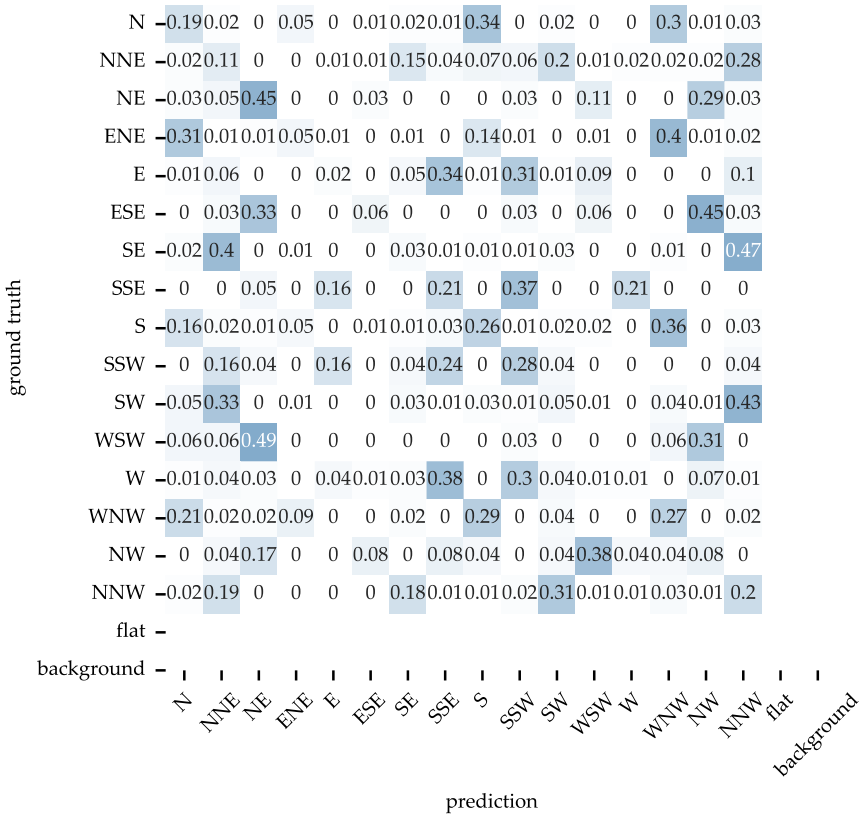


Figure 39: Confusion matrix between the roof’s orientation of the ground truth (nearest gutter algorithm) and the prediction (farthest roof segment algorithm) for 16 different orientations

The above figure shows no correlation between the ground truth and the prediction. No diagonal and anti-diagonal elements are matched in the result. Hence, the farthest roof segment algorithm could not predict the roof's orientation well.

The two results, which have used the nearest gutter algorithm, are presented below. The orientation acquired from this method is already used as ground truth in the above two results. Since the above algorithm could not work well, another method sought to find a better solution. First, the training is performed in four classes: rooflines, roof, gutter, and background. The test images are predicted using the trained network (network 8), and the image is processed only to get the gutter's information. Hence, the dataset used in this algorithm is the roof segments processed image after the prediction on network 5 and the gutter processed in network 8. The ground truth data are the processed test image. The code in the RID paper [39] was implemented at first to calculate the confusion matrix. The image shown below is the confusion matrix between the orientation of the predicted roof and the ground truth by using the nearest

gutter algorithm using RID code for azimuth calculation. The normalized confusion matrix shows that there is less misclassification between the predicted and ground truth as compared to the previous algorithm. However, the problem still exists in finding the correct orientation, although the correct gutter has been found. The matrix's diagonal elements show a reasonable classification between the predicted and ground truth classes. The SW orientation has the highest classification accuracy of 60%.

Similarly, the lowest classification accuracy counts for SSW, with only 20%. A matrix's anti-diagonal elements clearly show that the roof's orientation has been misclassified in the vertically opposite axis. For example, the SE orientation in the ground truth has been misclassified as SW. Similarly, W is misclassified as E, and the percentage of misclassification is also high. A new algorithm has been created to solve this problem with a small amount of modification in the code prepared in the RID code [39], which has been mentioned in section 4.11.6

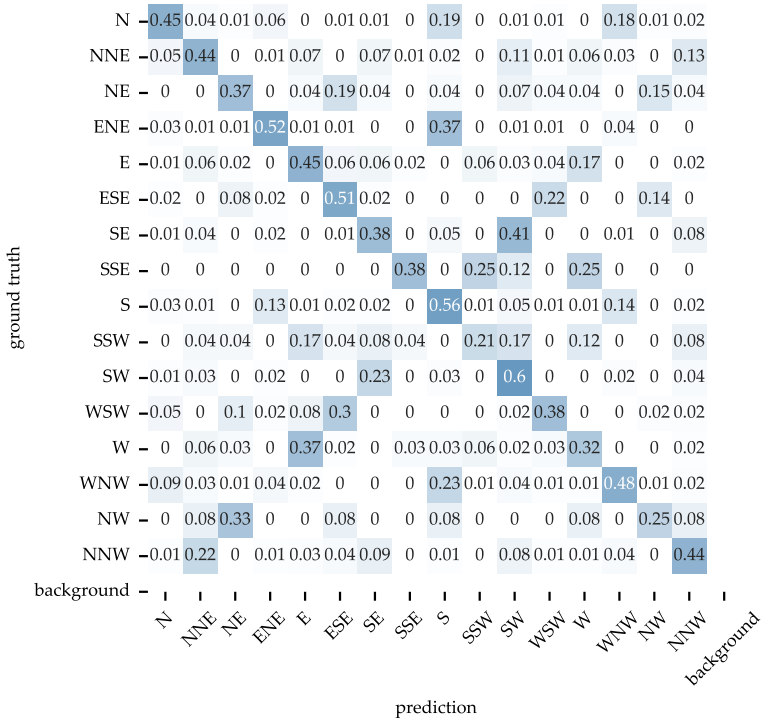


Figure 40: Confusion matrix between the roof's orientation of the ground truth (nearest gutter algorithm) and the prediction (nearest gutter algorithm) for 16 different orientations (RID code used for azimuth calculation)

Another result presented below has used the same algorithm for finding the gutter but has used different techniques for the azimuth calculation, which results in different orientation results. The orientation classes were classified according to the azimuth value, which is the same as in the RID paper, except for the northing classification, which used the azimuthal range from 348.75° to 11.25° instead of -11.25° to 11.25°. The algorithm has already been mentioned in section 4.11.5. The predicted roof's orientation map is presented in Figure 45. The same algorithm, the 'nearest gutter algorithm' is used for both the predicted and the ground truth. According to the classification, the highest number of roof orientations counts for the south with a total

roof of 351 and the least for NW with 19 roofs, shown in Figure 42. The diagonal elements of the confusion matrix show that the classification was well carried out. The value in the confusion box has shown that this algorithm has better results than the previous ones. The highest count of true classification is 72% for ENE and SW both. The lowest accuracy is found in the SSW-oriented roof, with 25%. The overall accuracy was found to be 58.55%. Even when visually inspected, this algorithm gives better results than the previous one. The roof's orientation of each roof is diagrammatically presented as this algorithm presented a promising result. The geographical map that shows the roof orientation in different colors is presented for visual comparison in Figure 44 and Figure 45.

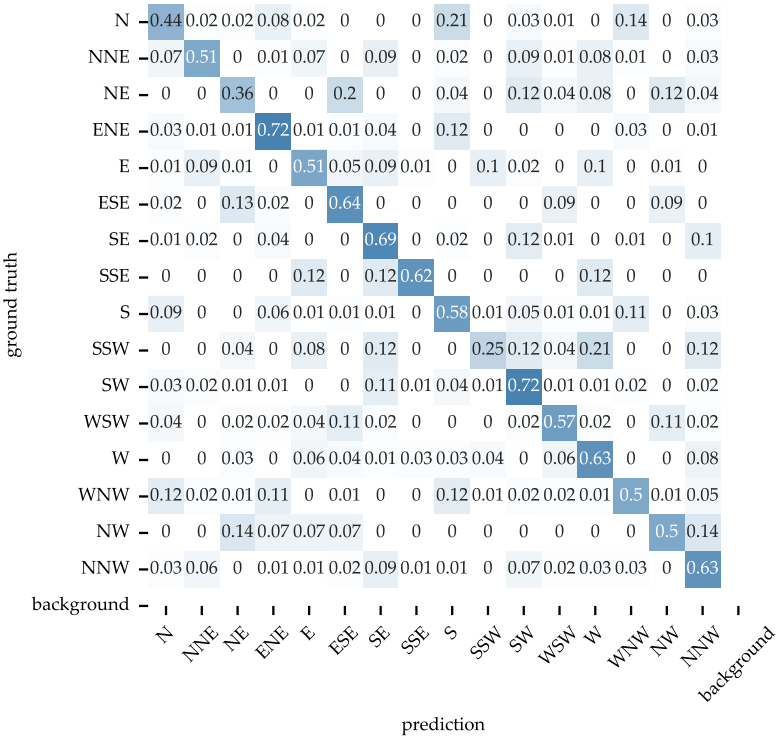


Figure 41: Confusion matrix between the roof's orientation of the ground truth (nearest gutter algorithm) and the prediction (farthest roof segment algorithm) for 16 different orientations (Modification on RID code for azimuth calculation)

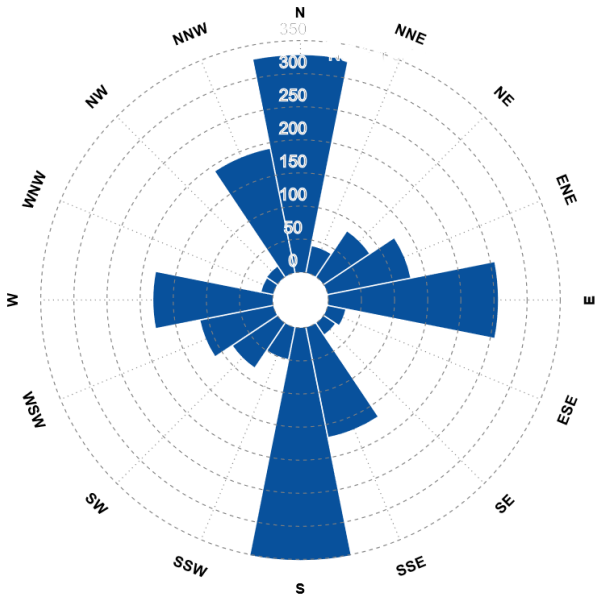


Figure 42: Graph showing the distribution of the predicted roof's orientation where the maximum counts for the south and minimum for the NW

Since the ground truth data used for the comparison is perfectly not valid as it has used the algorithm developed during the thesis work, this algorithm must be checked against some actual value. To perform that, a labeled dataset that already contains the orientation information is now considered the true label. The comparison is made between the true label and the predicted label. Figure 43 shows the confusion matrix between the ground truth and the predicted dataset. The highest accuracy is found in the ENE direction, with 83% accuracy, followed by S and WSW with 76%. The highest outlier in the confusion matrix is NW, predicted as SSE with 71%. The problems with 0% could not be figured out. There might be an annotation problem. For example, if the annotator inputs the orientation of SE and SSE as a SE, the confusion matrix shows that the True Positive is 0. From the confusion matrix, it is clear that the accuracy could be improved if the orientation is carried out for only four or eight directions instead of 16 different directions. The figure depicts that the direction present in one quadrant is spread out into different directions within that quadrant, decreasing accuracy. As there is no information on the flat roof in the prediction dataset, the flat roof classified in the ground truth is falsely classified as another orientation in the prediction.

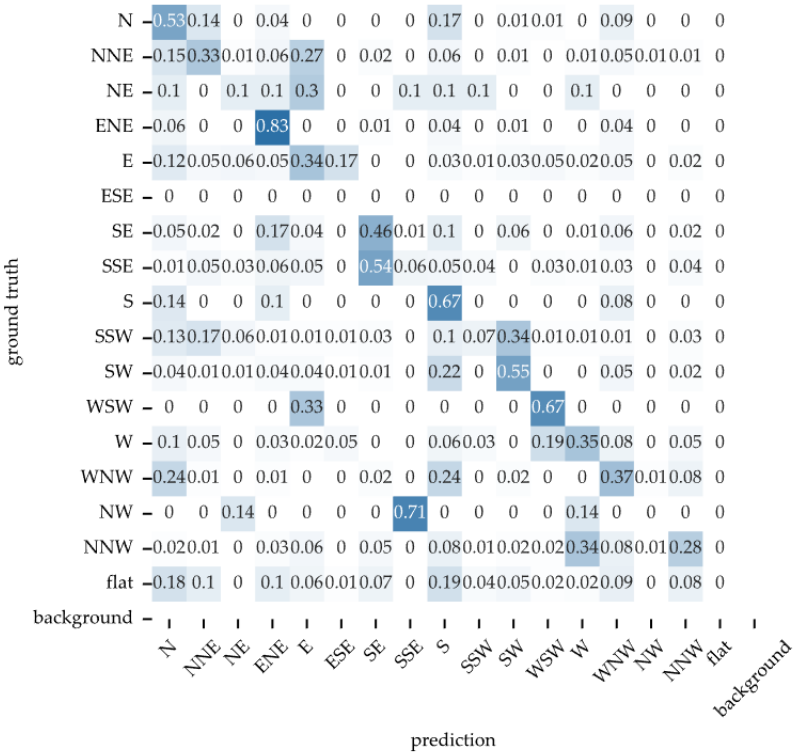


Figure 43: Confusion matrix between the ground truth (annotated dataset with 16 orientation classes) and the prediction (nearest gutter algorithm) to show the 16 different orientations

The two results are presented in the form of a geographical map. Figure 44 shows the orientation map of the ground truth (with 16 different orientations) depicted in 16 different colors. Figure 45 represents the orientation map of the predicted image in 16 different colors. The legend shows the directions with different colors for better map understanding. The difference between these two figures could be quickly figured out. The same color palette has been used for depicting the orientation. Some roofs in the top left are not annotated in the ground truth map. This could be the problem of low accuracy in the above confusion matrix. There are overlapping vectors in the predicted images as the same roof segment is present in the multiple annotated masks. It causes the predicted map to be multi-colored in the same roof segment, representing the same roof's different orientations. The problem persists mainly in the bordered roof segments.

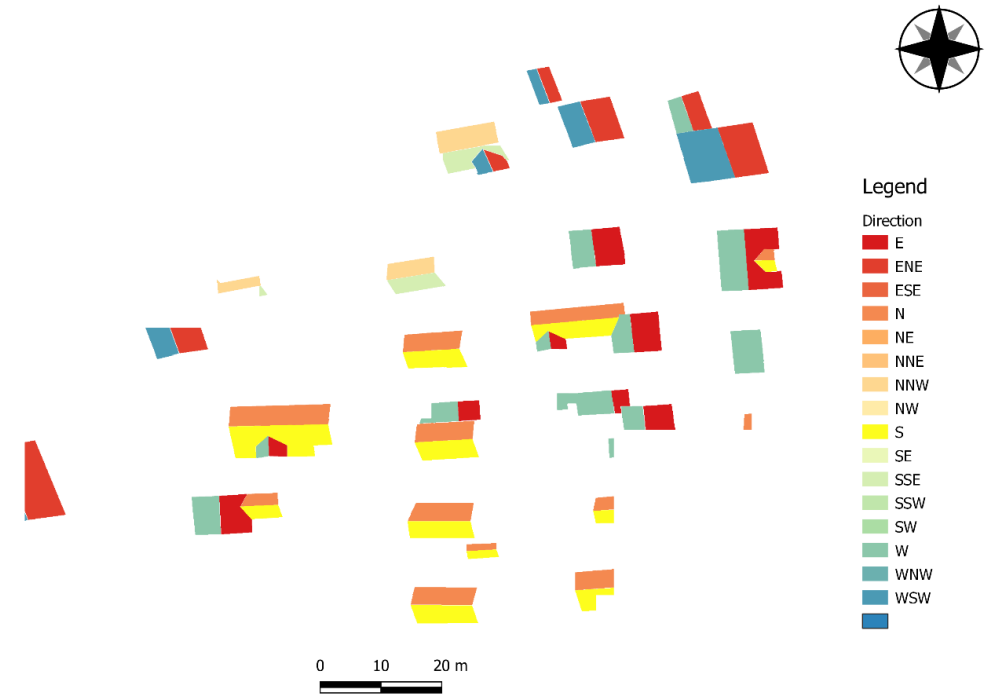


Figure 44: Geographical Map of the annotated classes (16 orientations)

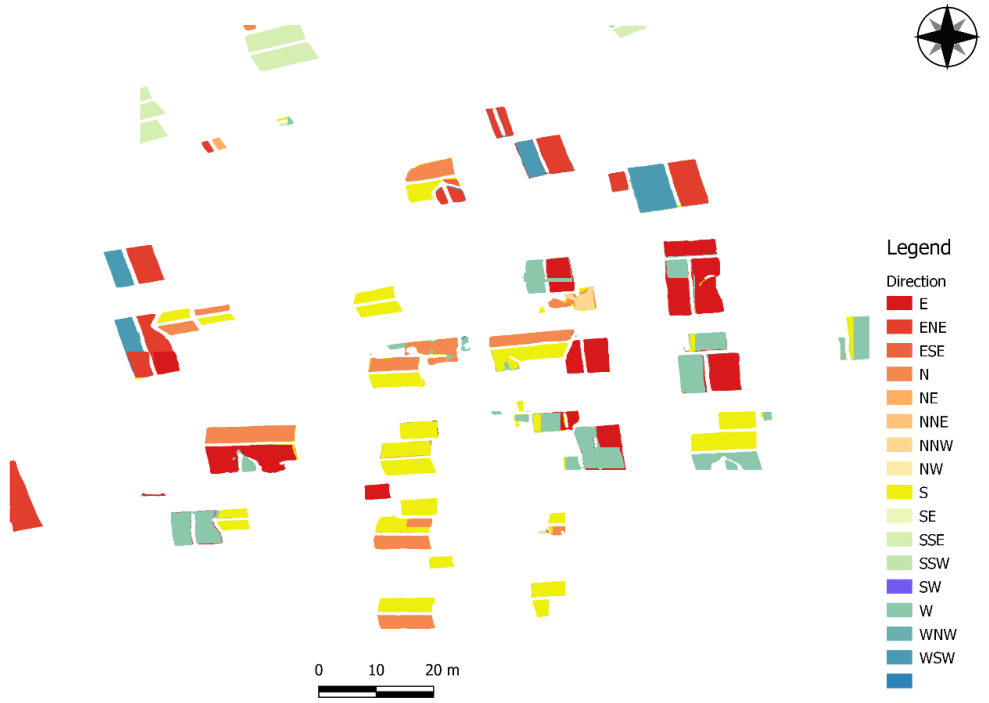


Figure 45: Geographical Map of the predicted orientation (16 orientations)

Orientation Error

Root Mean Squared Error (RMSE) estimates the error between the ground truth and the predicted roof's azimuth. It is a standard way to measure the error of a model in predicting quantitative data.

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}} \qquad 17$$

where $\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_n$ are predicted azimuthal values, $y_1, y_2, \ldots, y_n$ are ground truth azimuth values, and n denotes the total number of samples.

The azimuth of each predicted roof segment is calculated as mentioned in section 4.11.6. Similarly, the ground truth roof segment's azimuth is calculated using the same algorithm. The nearest gutter algorithm will be used for the azimuth calculation of both the ground truth and the predicted image. Hence, the azimuth calculated by this algorithm is used for the orientation error calculation. The data frames were checked first to avoid getting an empty data frame, ultimately giving an output of equal azimuth samples in both data frames. Then, azimuth calculation was carried out individually for two datasets: ground truth and predicted images. The differences between the azimuth of the two datasets ground truth and predicted, were studied. A significant difference of more than 180° was observed since the value of the azimuth ranges from 0° to 360°. It is noted that standard error metrics such as absolute error are not an ideal evaluation metric for the orientation performance calculation. For example, if the azimuth in the ground truth is 10° (N), but the predicted data has a corresponding azimuth of 340° (NNW), a considerable error of 330° will be accumulated in the result of Root Mean Squared Error. The error in predictions should be 30° instead. This large value might have a high influence in the performance calculation. The following solution tried to solve the problem of the difference in the azimuthal value. The value of MOE is between 0°(perfect prediction) and 180°(opposite prediction) [5]. Considering this, a program has been developed that checks if the difference between them is greater than 180°. If the absolute difference between the azimuth of the corresponding roof are greater than 180°, an operation is performed to get the minimum angle (actual error = 360°-absolute difference of azimuth). While taking the above example in the calculation, we get an absolute difference of azimuth as |340°-10°|, which is equal to 330°. Since it is greater than 180 °, when this absolute difference is substituted in the equation above, we get an actual error of 30° (i.e., 360°-330°).

The root mean square of the difference between the azimuth values of the prediction and the ground truth was carried out, which is generally known as Root Mean Squared Error (RMSE). The squared of the difference between the corresponding corrected azimuths was calculated. The mean of the squared difference is carried out, followed by the squared root of the mean giving the final orientation result. When using the technique above, a mean orientation error of ±74.82° was obtained. When the Roof Mean Square Root (RMSE) calculation was carried out without manipulation, an

orientation error between the ground truth and the predicted azimuth was found to be ± 98.13°.

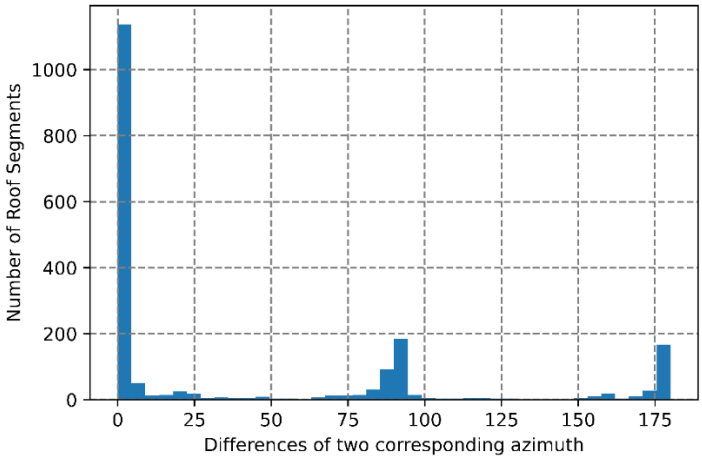


Figure 46: Figure showing the differencces between the ground truth and prediction's azimuth and the total number of roofs

It is evident from the above figure that deviations of 90° and 180° cause a significant number of errors. Over 1000 roof segments have the largest number of error differences that are close to zero. The error difference is significant between 80° and 90°, with roof segments close to 200 in number. Roofs are often predicted to be the opposite, and their number is also near 200, although fewer than in the preceding scenario. These two deviations have a share of 36%, leading to a maximum error in the Mean Orientation Error. Rather than deviation at 90°, the deviation at 180° contributed more error in the result as the difference between the ground truth and predicted dataset would be near 180°, ultimately accumulating in the final error. If these two deviations of 90° and 180° are ignored, and a threshold of differences is maintained while calculating the MOE, a deviation of ± 21.65° could be observed.

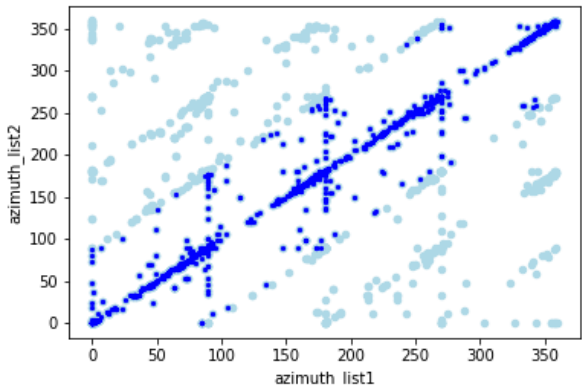


Figure 47: Scatter plot between ground truth and predicted azimuth. The blue color in the plot is the scatter plot after outlier removal, and light blue (also the region hidden by Blue) is the plot without any outlier removal

5.5 Review of Hypothesis

The results are now discussed with the research hypothesis developed in section 3.2.

The training and predicted result show that hypothesis 1 is accepted, showing that the rooflines, roof, and background can be individually segmented and that the pixel of rooflines is enough to separate the roof into different segments. Figure 21 clearly shows that the rooflines can separate the roofs.

It is evident while discussing the second hypothesis that the roof segment might be separated using the three classes (roofline, roof, and background). However, an extra class (gutter) is needed to determine the roof segment's proper azimuth. The algorithm which does not use a gutter (longest line algorithm, farthest roof segment algorithm) has some disadvantages of not correctly finding the correct outer line for orientation determination.

6 Closing

6.1 Summary

The feasibility and viability of a data-driven algorithm that could determine the orientation of the roof segments present in the aerial image were explored. This thesis sought an alternative approach to determine the roof's orientation. The thesis aims to reduce the time and labor used to label different roof orientations for training purposes. This work includes the label of the roof, rooflines, and gutter only for the orientation determination, which has significantly increased the efficiency and effectiveness. Previous studies used 16 different labels for different orientations, which was cumbersome and impractical. A CNN network is used for the training purpose, and the model generated from the training is used to predict the image from the test dataset. Since the network with three classes gives more IoU accuracy and F1-Score value, this network is used for the roof segment separation from the gable line. A separate four classes network was prepared to segment the gutter from the image. Network 7 could also be used for the roof and roofline extractions, but due to lower accuracy than the above network, it was just used for the gutter extraction. In order to minimize the work, the training was only performed for three classes to separate roof segments using a gable line. However, the reference roofline could not be determined, which could help find the correct orientation. Hence a gutter line was considered one of the significant parts of the orientation calculation. The algorithm was developed for the azimuth calculation using the gutter line mentioned in chapter 4. The training and orientation results have been presented in chapter 5. The best network configuration was considered network 5, which used Categorical Focal Loss and Jaccard loss as their loss function, ultimately used for generating predicted images. The predicted images were post-processed to get the roof segment and the gutter, then vectorized. The azimuth was calculated using a gutter line, and the corresponding azimuth and orientation information was stored in the vectorized data frame of the roof segment. Finally, a geographical map was prepared using the orientation information stored in the roof segment's data frame.

6.2 Conclusions

The complete pipeline proposed above can find the azimuth of a roof and determine its orientation using aerial imagery and labeled data. The CNN could generate a nice predicted image that could be used to segment rooflines, roofs, and gutters from the background. The azimuth calculation and orientation determination were only possible due to higher accuracy in the training network. The IoU of the predicted images against the ground truth gives an accuracy of 0.79, 0.99, and 0.93 for the roofline, roof, and background, respectively, on best network 5. Similarly, the training acquired a mean IoU of 82.86% and an F1-Score of 89.87%. Another network used for the gutter extraction: network 8, gives a mean IoU of 69.91% and an F1-Score of 80%. The predicted images on network 8 have an IoU score of 0.65, 0.92, 0.64, and 0.99 for rooflines, roof, gutter, and background, respectively. The training network should give higher accuracy to segment the rooflines accurately to segment the multiple roof

segments from a one roof structure. This thesis claims that without using sophisticated methods like LiDAR or any 3D dataset, the roof's orientation could be determined using cheap and freely available datasets. The result found that among the predicted images, most of the roofs are oriented toward the south direction, with a total number of 351, and the least towards NW, with a number of 19. As mentioned in chapter 1, the south roof receives more sunlight, and the orientation of the roofs of the test dataset shows that these roofs are suitable for solar power installation. The confusion matrix between the predicted dataset and the ground truth shown in Figure 41 shows the highest overall accuracy of 58.55%. The highest count of true classification is 72% for ENE and SW, and the lowest accuracy is found in the SSW-oriented roof, with 25%. These results were also evaluated using an RMSE, where the orientation error of ± 98.13° could be observed.

Different validations were carried out assuming different ground truths, as there was a lack of validation datasets. The ground truth datasets had a missing label and created a problem in validation. All the comparison of the orientation between the predicted and assumed ground truth were carried out and is presented in section 5.3. The orientation of the ground truth was also determined by the same algorithm adapted to determine the predicted roof's orientation. Hence the results might be biased. But the comparison of the ground truth (annotated dataset with 16 orientation classes) and predicted dataset shows that the resulting prediction orientation is rightly classified. If the datasets were to be classified only in 8 directions instead of 16, the accuracy of the classification would be significantly higher. The majority of inaccuracies are caused by deviations around 90° and 180°. The aim of finding orientation using the roofline, roof, and gutter is achieved despite minor errors. The annotation of the roofline, roof, and gutter is enough to achieve the required orientation.

6.3 Further Works

Due to time and data limitations, various adaptations, tests, and experiments have been left. Several possibilities could be improved in the future. They are:

- To achieve more accuracy, the missing labels could be labeled again. Comparing the roof's orientation between the test dataset and the projected dataset proved challenging due to the test dataset's missing labels, necessitating a compromise, and removing several roof segments.
- The training dataset size might be increased to achieve a better-trained model. Additionally, an alternative dataset with a different kind of roof than the current one might be tested.
- Finding a better network that could extract the gutter using just the background and gutter classes would be possible.
- The additional test dataset could use in the trained model to get a more predicted dataset to cover a large area and predict the feasibility of PV installation. The robustness of the approach can be checked by utilizing the model in other study areas.

List of Figures

List of Figures

List of Tables

Bibliography

[1] International Energy Agency, “Global CO2 emissions rebounded to their highest level in history in 2021 - News - IEA,” *Press Release*. 2022. [Online]. Available: https://www.iea.org/news/global-co2-emissions-rebounded-to-their-highest-level-in-history-in-2021

[2] “Annual 2021 Global Climate Report | National Centers for Environmental Information (NCEI),” *NOAA National Centers for Environmental Information*. 2022. [Online]. Available: https://www.ncei.noaa.gov/access/monitoring/monthly-report/global/202113#ref

[3] FEDERAL MINISTRY FOR ECONOMIC AFFAIRS AND CLIMATE ACTION, “Renewable Energy,” 2022.

[4] I. Renewable Energy Agency, “Renewable Capacity Statistics 2022.” pp. 20–22, 2022. [Online]. Available: www.irena.org

[5] S. Lee, S. Iyengar, M. Feng, P. Shenoy, and S. Maji, “Deeproof: A data-driven approach for solar potential estimation using rooop imagery,” *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, pp. 2105–2113, 2019, doi: 10.1145/3292500.3330741.

[6] Open Geospatial Consortium, “Open Geospatial Consortium OGC City Geography Markup Language (CityGML) En- coding Standard,” 2012.

[7] K. Chaturvedi, B. Willenborg, M. Sindram, and T. H. Kolbe, “SOLAR POTENTIAL ANALYSIS AND INTEGRATION OF THE TIME-DEPENDENT SIMULATION RESULTS FOR SEMANTIC 3D CITY MODELS USING DYNAMIZERS,” vol. IV, no. October, pp. 26–27, 2017.

[8] R. Castello, S. Roquette, M. Esguerra, A. Guerra, and J. L. Scartezzini, “Deep learning in the built environment: Automatic detection of rooftop solar panels using Convolutional Neural Networks,” *J. Phys. Conf. Ser.*, vol. 1343, no. 1, 2019, doi: 10.1088/1742-6596/1343/1/012034.

[9] S. Krapf, N. Kemmerzell, S. K. H. Uddin, M. H. Vázquez, F. Netzler, and M. Lienkamp, “Towards Scalable Economic Photovoltaic Potential Analysis Using Aerial Images and Deep Learning,” *Energies*, vol. 14, no. 13, 2021, doi: 10.3390/en14133800.

[10] S. Xu, X. Pan, E. Li, B. Wu, and S. Bu, “Images via Hierarchical RGB-D Priors,” pp. 1–19, 2018, doi: 10.1109/TGRS.2018.2850972.

[11] Y. Qin, Y. Wu, B. Li, S. Gao, M. Liu, and Y. Zhan, “Semantic Segmentation of Building Roof in Dense Urban Environment with Deep Convolutional Neural Network: A Case Study Using GF2 VHR Imagery in China,” *Sensors*, vol. 19, no. 5, p. 1164, Mar. 2019, doi: 10.3390/s19051164.

[12] K. Mainzer, S. Killinger, R. McKenna, and W. Fichtner, “Assessment of rooftop photovoltaic potentials at the urban level using publicly available geodata and image recognition techniques,” *Sol. Energy*, vol. 155, pp. 561–573, 2017, doi:

10.1016/j.solener.2017.06.065.

[13] A. Verso, A. Martin, J. Amador, and J. Dominguez, “GIS-based method to evaluate the photovoltaic potential in the urban environments: The particular case of Miraflores de la Sierra,” *Sol. Energy*, vol. 117, pp. 236–245, 2015, doi: 10.1016/j.solener.2015.04.018.

[14] K. H. U. Syed, “A Deep Learning Approach to Roof Superstructure detection for PV potential estimation,” 2020.

[15] H. Yang and L. Lu, “The optimum tilt angles and orientations of PV claddings for Building-Integrated Photovoltaic (BIPV) applications,” *J. Sol. Energy Eng. Trans. ASME*, vol. 129, no. 2, pp. 253–255, 2007, doi: 10.1115/1.2212439.

[16] T. Santos, N. Gomes, S. Freire, M. C. Brito, L. Santos, and J. A. Tenedório, “Applications of solar mapping in the urban environment,” *Appl. Geogr.*, vol. 51, pp. 48–57, 2014, doi: 10.1016/j.apgeog.2014.03.008.

[17] J. A. N. Lee, “Computer Pioneers - Arthur Lee Samuel,” *IEEE computer society*. 2013. [Online]. Available: https://history.computer.org/pioneers/samuel.html

[18] G. E. Hinton, S. Osindero, and Y.-W. Teh, “A Fast Learning Algorithm for Deep Belief Nets,” *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, Jul. 2006, doi: 10.1162/neco.2006.18.7.1527.

[19] M. Alloghani, D. Al-Jumeily, J. Mustafina, A. Hussain, and A. J. Aljaaf, “A Systematic Review on Supervised and Unsupervised Machine Learning Algorithms for Data Science BT - Supervised and Unsupervised Learning for Data Science,” M. W. Berry, A. Mohamed, and B. W. Yap, Eds. Cham: Springer International Publishing, 2020, pp. 3–21. doi: 10.1007/978-3-030-22475-2_1.

[20] Onlim, “Supervised vs.” 2020. [Online]. Available: https://i.imgur.com/x6gQdYh.png

[21] I. C. Education, “What are convolutional neural networks (CNN)?,” *Bdtechtalks.Com*. p. What is?, 2020. [Online]. Available: https://www.ibm.com/cloud/learn/convolutional-neural-networks

[22] G. W. Lindsay, “Convolutional neural networks as a model of the visual system: Past, present, and future,” *J. Cogn. Neurosci.*, vol. 33, no. 10, pp. 2017–2031, 2021, doi: 10.1162/jocn_a_01544.

[23] J. J. and Y. S. F. Li, “CS231n: Convolutional Neural Networks for Visual Recognition,” *Stanford University*. pp. 1–9, 2020. [Online]. Available: https://cs231n.github.io/convolutional-networks/

[24] Q. Zhang, M. Zhang, T. Chen, Z. Sun, Y. Ma, and B. Yu, “Recent advances in convolutional neural network acceleration,” *Neurocomputing*, vol. 323, pp. 37–51, 2019, doi: 10.1016/j.neucom.2018.09.038.

[25] A. Dertat, “Applied Deep Learning - Part 4: Convolutional Neural Networks | by Arden Dertat | Towards Data Science.” pp. 1–29, 2017. [Online]. Available: https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

[26] J. Kafunah, “Backpropagation In Convolutional Neural Networks,” *Deep Grid*, vol. 1, no. September. pp. 1–14, 2016. [Online]. Available: http://www.jefkine.com/general/2016/09/05/backpropagation-in-convolutional-neural-networks/

[27] S. PULKIT, “Image Segmentation | Types Of Image Segmentation,” *Analyticsvidhya*. 2019. [Online]. Available: https://www.analyticsvidhya.com/blog/2019/04/introduction-image-segmentation-techniques-python/

[28] A. Arnab and P. H. S. Torr, “Pixelwise instance segmentation with a dynamically instantiated network,” *Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 879–888, 2017, doi: 10.1109/CVPR.2017.100.

[29] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc.*, pp. 1–14, 2015.

[30] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-Decem, pp. 770–778, 2016, doi: 10.1109/CVPR.2016.90.

[31] “Aman’s AI Journal • CS231n • Detection and Segmentation.” [Online]. Available: https://aman.ai/cs231n/detection/

[32] M. Saifi, J. Singla, and Nikita, *Deep Learning based Framework for Semantic Segmentation of Satellite Images*. 2020. doi: 10.1109/ICCMC48092.2020.ICCMC-00069.

[33] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” in *Advances in Neural Information Processing Systems*, 2012, vol. 25. [Online]. Available: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

[34] C. Szegedy *et al.*, “Going deeper with convolutions,” in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1–9. doi: 10.1109/CVPR.2015.7298594.

[35] C. Shorten and T. M. Khoshgoftaar, “A survey on Image Data Augmentation for Deep Learning,” *J. Big Data*, vol. 6, no. 1, p. 60, 2019, doi: 10.1186/s40537-019-0197-0.

[36] C. Cortes, M. Mohri, and A. Rostamizadeh, “L2 Regularization for Learning Kernels.” arXiv, 2012. doi: 10.48550/ARXIV.1205.2653.

[37] W. Weng and X. Zhu, “INet: Convolutional Networks for Biomedical Image Segmentation,” *IEEE Access*, vol. 9, pp. 16591–16603, 2021, doi: 10.1109/ACCESS.2021.3053408.

[38] O. Ronneberger, P.Fischer, and T. Brox, “U-Net: Convolutional Networks for Biomedical Image Segmentation,” in *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2015, vol. 9351, pp. 234–241. [Online]. Available: http://lmb.informatik.uni-freiburg.de/Publications/2015/RFB15a

[39] S. Krapf, L. Bogenrieder, F. Netzler, G. Balke, and M. Lienkamp, “RID—Roof Information Dataset for Computer Vision-Based Photovoltaic Potential Assessment,” *Remote Sens.*, vol. 14, no. 10, 2022, doi: 10.3390/rs14102299.

[40] D. de B. Soares *et al.*, “Predicting the Solar Potential of Rooftops using Image Segmentation and Structured Data,” no. April, 2021, [Online]. Available: http://arxiv.org/abs/2106.15268

[41] Z. Qian *et al.*, “Deep Roof Refiner: A detail-oriented deep learning network for refined delineation of roof structure lines using satellite imagery,” *Int. J. Appl. Earth Obs. Geoinf.*, vol. 107, p. 102680, 2022, doi: 10.1016/j.jag.2022.102680.

[42] F. Faltermeier, B. Sc, S. Krapf, M. Sc, B. Willenborg, and M. Sc, “Improving CNN roof segment detection by dataset extension using 3D city models Supervisors :,” no. April, 2022.

[43] C. A. Schneider, W. S. Rasband, and K. W. Eliceiri, “NIH Image to ImageJ: 25 years of image analysis,” *Nat. Methods*, vol. 9, no. 7, pp. 671–675, 2012, doi: 10.1038/nmeth.2089.

[44] J. Charng *et al.*, “Deep learning segmentation of hyperautofluorescent fleck lesions in Stargardt disease,” *Sci. Rep.*, vol. 10, p. 16491, 2020, doi: 10.1038/s41598-020-73339-y.

[45] T. J. Yi, “Guidance, Control and Dynamics Commons Recommended Citation Recommended Citation Yi,” vol. 3593, 2020, [Online]. Available: https://scholar.afit.edu/etd/3593

[46] G. Boesch, “Deep Residual Networks (ResNet, ResNet50) ,” *Deep Learning*. 2021. [Online]. Available: https://viso.ai/deep-learning/resnet-residual-neural-network/

[47] A. Arora, “What is Focal Loss and when should you use it? | Committed towards better future.” 2020. [Online]. Available: https://amaarora.github.io/2020/06/29/FocalLoss.html

[48] B. Baloch, S. Kumar, A. Rehman, and T. Syed, *Focused Anchors Loss: cost-sensitive learning of discriminative features for imbalanced classification*. 2019.

[49] willem (https://stats.stackexchange.com/users/159052/willem), “F1/Dice-Score vs IoU.” [Online]. Available: https://stats.stackexchange.com/q/276144

[50] G. Bradski, “The OpenCV Library,” *Dr. Dobb’s J. Softw. Tools*, 2000.

[51] Doxygen, “OpenCV: Contours : Getting Started.” 2015. [Online]. Available: https://docs.opencv.org/3.1.0/d4/d73/tutorial_py_contours_begin.html

[52] K. Jordahl *et al.*, “geopandas/geopandas: v0.8.1.” Zenodo, Jul. 2020. doi: 10.5281/zenodo.3946761.

[53] S. Gillies and others, “Shapely: manipulation and analysis of geometric objects.” [Online]. Available: https://github.com/shapely/shapely

[54] K. Rajput, “Difference Between Whole Circle Bearing and Quadrantal Bearing | What Is WCB | What Is QB,” *CivilJungle*. 2021. [Online]. Available: https://civiljungle.com/wcb-vs-qb/

[55] Dr. B.C. PUNMIA, “Paras- Taneja.Blospo T.in,” vol. I, no. Volume I.

[56] D. S. Kim, M. Arsalan, and K. R. Park, “Convolutional Neural Network-Based Shadow Detection in Images Using Visible Light Camera Sensor,” *Sensors*, vol. 18, no. 4, 2018, doi: 10.3390/s18040960.

A Appendices

A.1 Attributes contained in a vector file (GeoDataFrame)

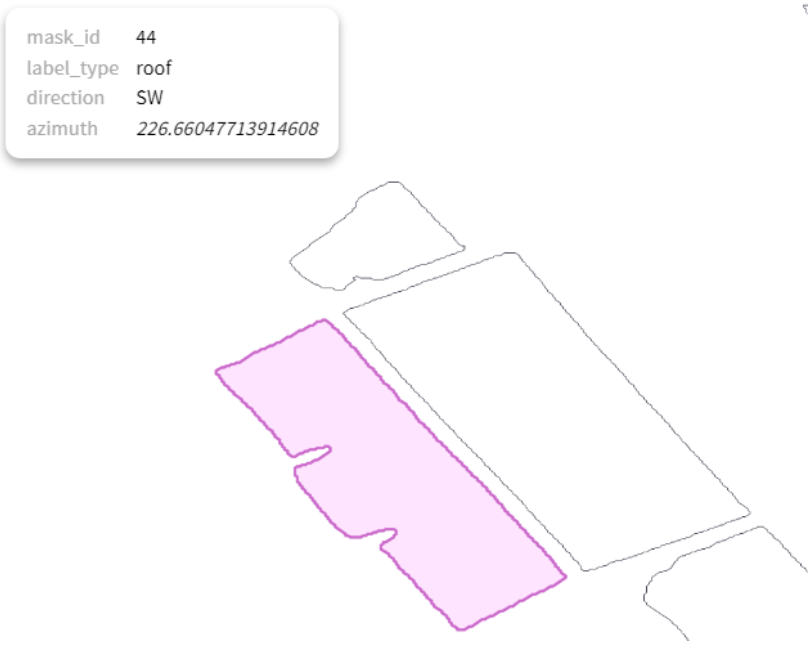
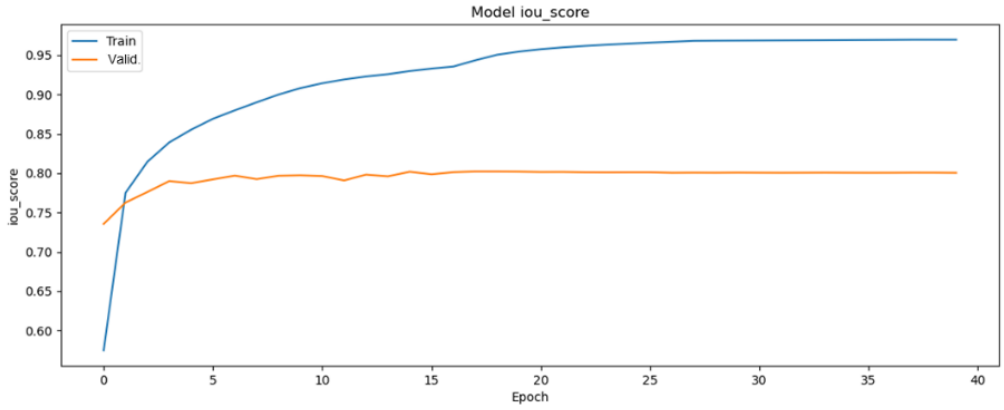


Figure A. 1: A vector file containing a dataframe which stores the calculated direction as well as azimuth of the roof with mask_id 44

A.2 Training curve on different network configurations



A Appendices

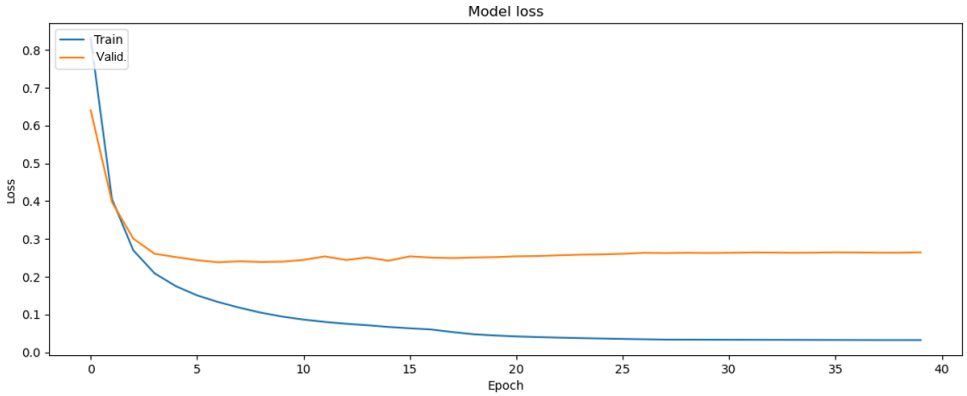


Figure A. 2: Training curve on network 1

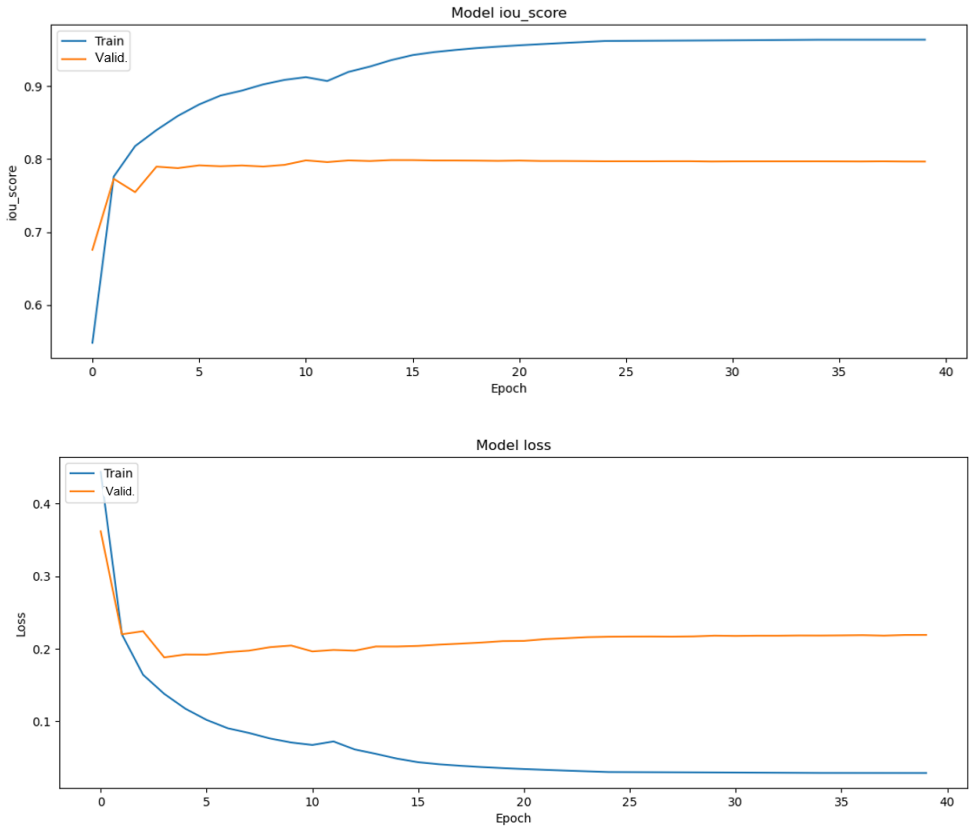


Figure A. 3: Training curve on network 2

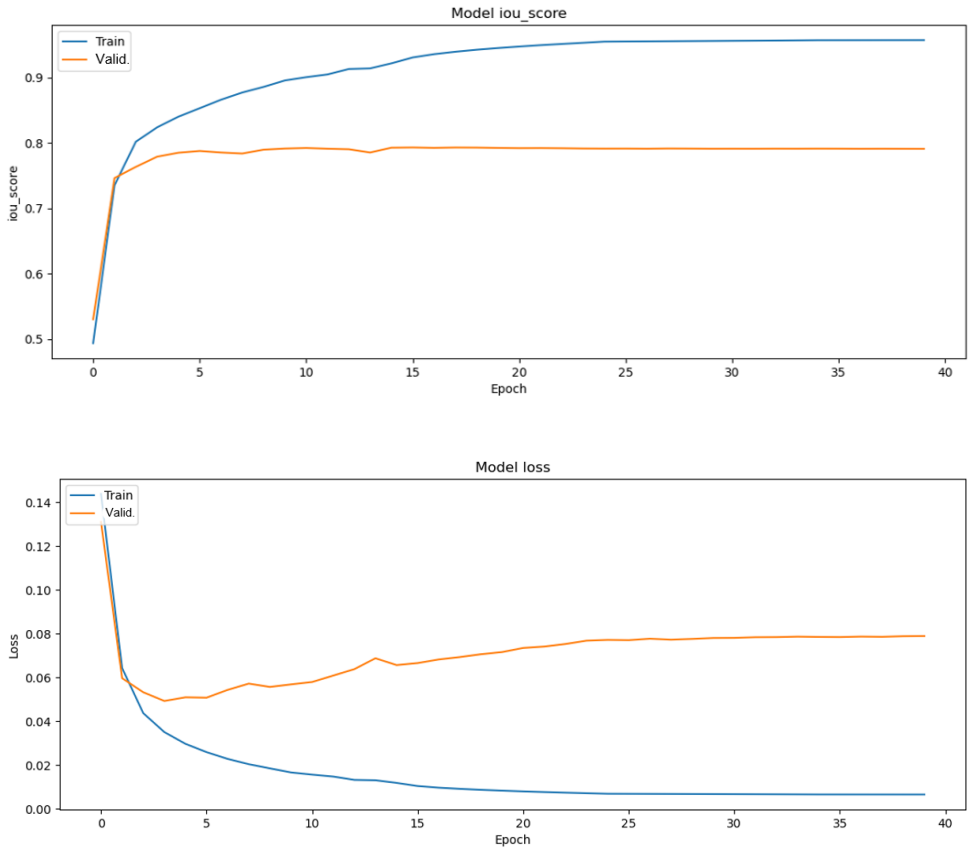


Figure A. 4: Training curve on network 4

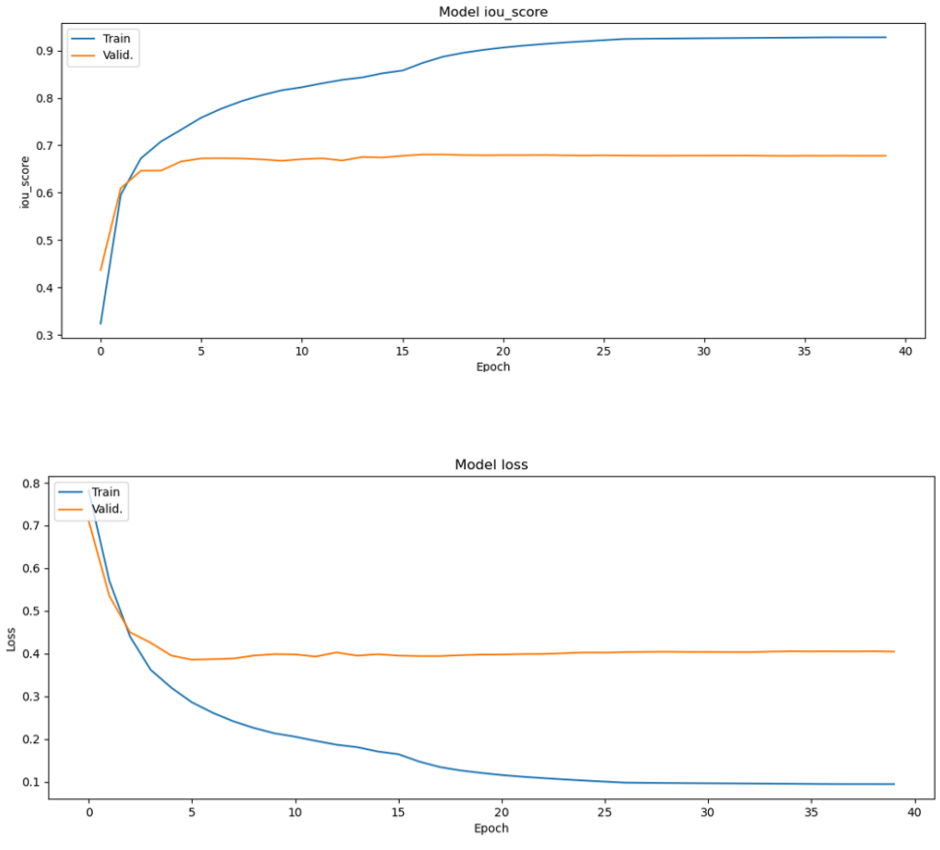


Figure A. 5: Training curve on network 7

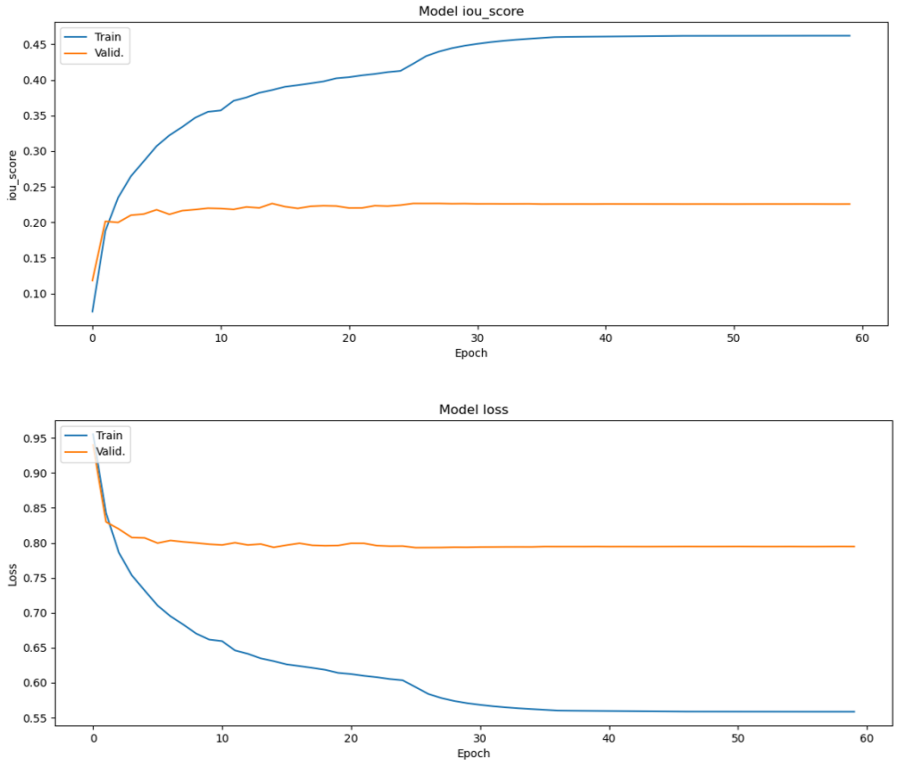


Figure A. 6: Training curve on network 9

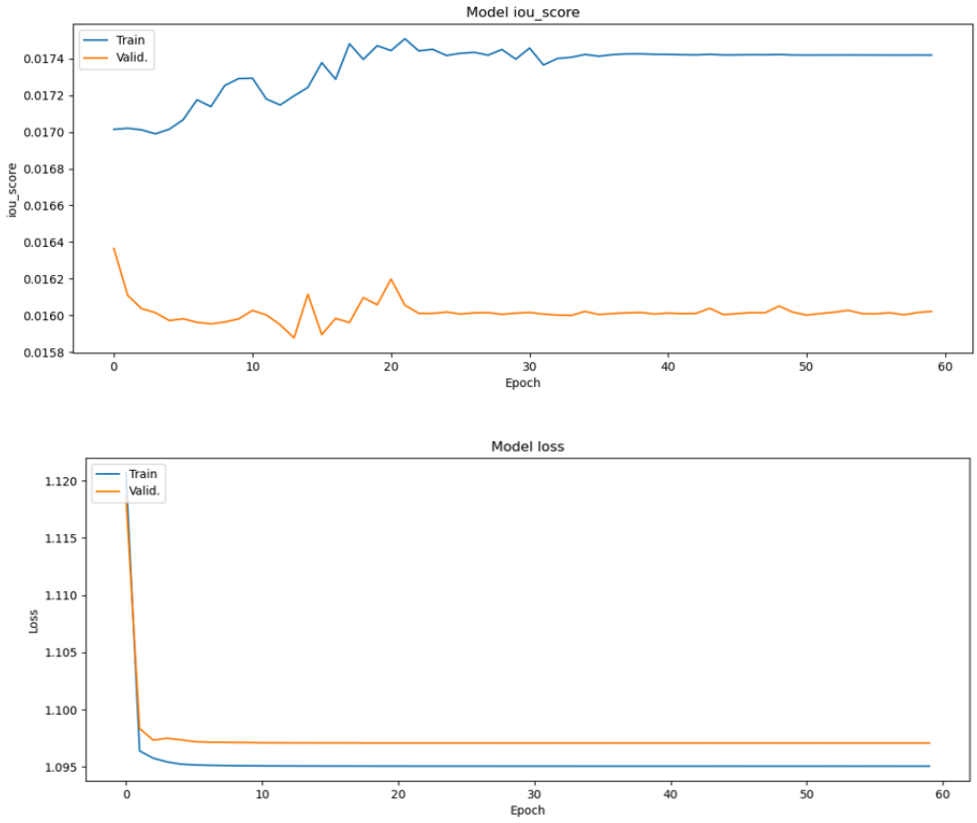


Figure A. 7: Training curve on network 10

A.3 Confusion matrix

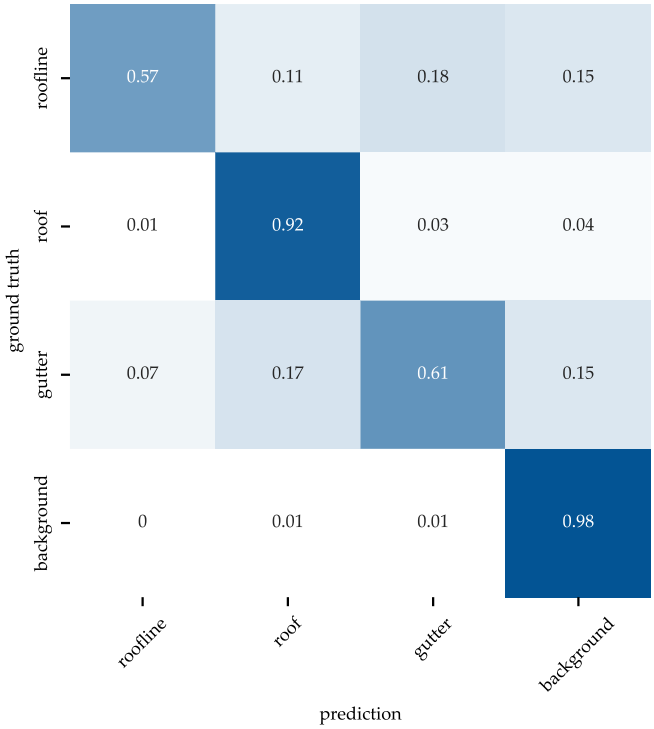


Figure A. 8: Confusion matrix between the ground truth and the prediction (4 classes) on network 7

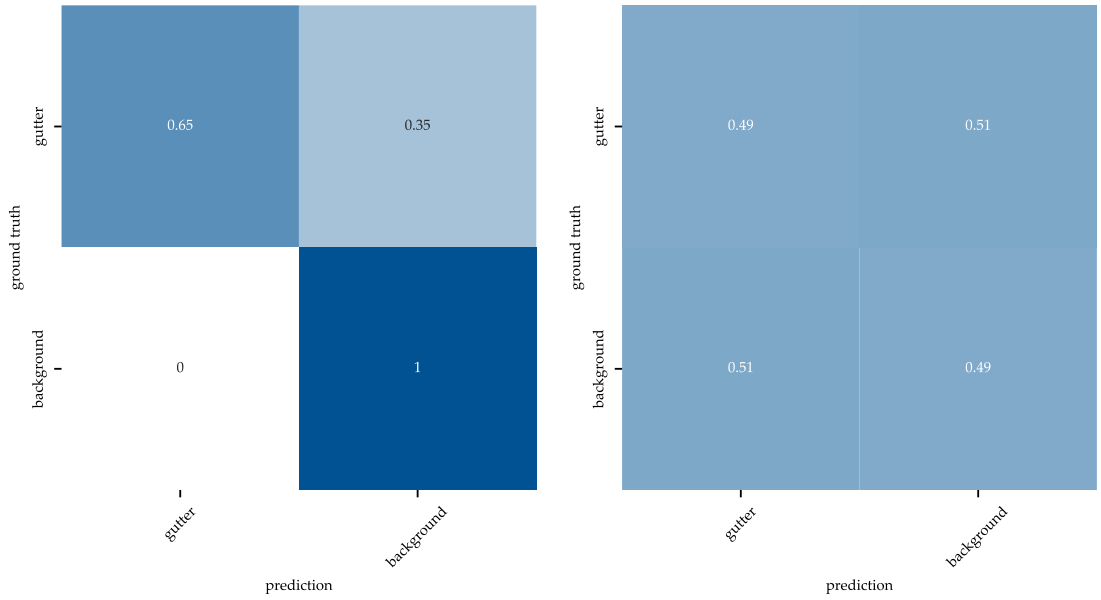


Figure A. 9: A confusion matrix between the ground truth and the prediction (2 classes) on network 9 (left) and network 10 (right)