Tampere University

Veera Kallio

SKY SEGMENTATION OF FISHEYE IMAGES FOR IDENTIFYING NON-LINE-OF-SIGHT SATELLITES

Master of Science Thesis
Faculty of Information Technology and Communication Sciences
Examiners: Assoc. Prof. Esa Rahtu
M.Sc. Manuel Vonlanthen
June 2023

ABSTRACT

Veera Kallio: Sky Segmentation of Fisheye Images for Identifying Non-Line-of-Sight Satellites
Master of Science Thesis
Tampere University
Master's Degree Programme in Electrical Engineering
June 2023

GNSS (global navigation satellite system) receivers are often deployed in environments where some satellite signals are blocked by buildings and other obstructions. This non-line-of-sight situation is challenging for GNSS positioning because the signals can still be received via indirect paths, which causes errors to the calculated position. Knowledge of the blocked satellites would help in mitigating these errors.

A sky-pointing fisheye camera can be used to gather information of the surroundings of the receiver in order to detect non-line-of-sight situations. Using semantic segmentation of the sky, the image can be segmented into line-of-sight and non-line-of-sight regions. By projecting the satellite locations onto the image, each satellite can be classified according to the segmentation.

The objective of this thesis is to study the use of neural networks in segmenting the sky from fisheye images and to classify the possibly visible satellites as line-of-sight and non-line-of-sight based on the segmentation. Several popular segmentation networks were trained and evaluated to compare their performance on the task. A manually labeled, small dataset was prepared, containing images with different weather conditions and environments, including tunnels. The results were validated on a larger test set using GNSS data.

The study shows that neural networks can segment the sky from fisheye images very precisely, reaching almost 99% intersection over union and over 99% F1-score. The best-performing model was a U-Net with EfficientNetB6 encoder, but there was little difference between the tested models. The satellite classification performed after the segmentation was also accurate and in line with the signal strengths. It can be concluded on the basis of the study that fisheye sky segmentation with neural networks is an effective and useful method for line-of-sight detection.

Keywords: semantic segmentation, convolutional neural network, GNSS, sky detection, line-of-sight classification, fisheye camera, deep learning

TIIVISTELMÄ

Veera Kallio: Taivaan segmentointi kalansilmäkuvista epäsuoran näköyhteyden satelliittien tunnistamiseen
Diplomityö
Tampereen yliopisto
Sähkötekniikan DI-ohjelma
Kesäkuu 2023

GNSS-vastaanottimia (engl. global navigation satellite system) käytetään usein ympäristöissä, joissa joidenkin satelliittisignaalien reitti estyy rakennusten tai muiden esteiden takia. Tällainen epäsuoran näköyhteyden tilanne on haastava GNSS-paikannuksessa, koska signaali saatetaan silti vastaanottaa epäsuoraa reittiä pitkin, mikä aiheuttaa virheitä laskettuun sijaintiin. Tieto estetyistä satelliiteista auttaisi vähentämään näitä virheitä.

Taivaalle suunnattua kalansilmäkameraa voidaan käyttää keräämään tietoa vastaanottimen ympäristöstä, jotta epäsuoran näköyhteyden tilanteet voitaisiin havaita. Taivaan semanttisen segmentoinnin avulla kuva voidaan jakaa suoran ja epäsuoran näköyhteyden alueisiin. Projisoimalla satelliittien sijainnit kuvaan kukin satelliitti voidaan luokitella segmentoinnin mukaisesti.

Tämän työn tarkoituksena on tutkia neuroverkkojen käyttöä taivaan segmentointiin kalansilmäkuvista ja luokitella mahdollisesti nähtävissä olevat satelliitit suoran ja epäsuoran näköyhteyden kategorioihin segmentoinnin perusteella. Useita suosittuja segmentointineuroverkkoja opetettiin ja testattiin, jotta niiden suoriutumista tehtävässä voitiin verrata. Pieni joukko kuvia luokiteltiin manuaalisesti niin, että mukaan otettiin erilaisia sääolosuhteita ja ympäristöjä, mukaan lukien tunneleita. Tulokset vahvistettiin suuremmalla määrällä testikuvia käyttäen GNSS-dataa.

Tutkimus osoittaa, että neuroverkot pystyvät segmentoimaan taivaan kalansilmäkuvista erittäin tarkasti, saavuttaen lähes 99% IoU- (engl. intersection over union) ja yli 99% F1-tuloksen. Parhaiten suoriutuva malli oli U-Net EfficientNetB6 enkooderilla, mutta tutkittujen mallien välillä oli vain vähän eroa. Segmentoinnin jälkeen suoritettu satelliittien luokittelu oli myös täsmällinen ja linjassa signaalien voimakkuuden kanssa. Työn perusteella voidaan päätellä, että taivaan segmentointi kalansilmäkuvista neuroverkoilla on tehokas ja hyödyllinen menetelmä suoran näköyhteyden tunnistamiseen.

Avainsanat: semanttinen segmentointi, konvoluutioneuroverkko, GNSS, taivaan tunnistaminen, suoran näköyhteyden luokittelu, kalansilmäkamera, syväoppiminen

PREFACE

This thesis was carried out for u-blox during spring 2023. The thesis project would not have been possible without help from others and I want to express my sincere gratitude to those who contributed to the thesis.

I want to give special thanks to my supervisor Manuel Vonlanthen for his guidance throughout the project and his valuable feedback and ideas, as well as overall support. Thank you also to professor Esa Rahtu for his advice and ideas.

I want to thank u-blox for this very interesting thesis opportunity, as well as for first introducing me to the fascinating world of GNSS. Big thanks to my team and other colleagues at u-blox for always supporting me, keeping me company, and often making me smile.

Thank you to my friends and family for always believing in me and supporting me.

Lastly and most importantly, I want to thank Him who has created this amazing world and our amazing brains to study it. For everything.

Tampere, 12th June 2023

Veera Kallio

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND SYMBOLS

C/N_0	carrier-to-noise density ratio
CNN	convolutional neural network
FPN	feature pyramid network
GNSS	global navigation satellite system
GPS	Global Positioning System
IoU	intersection over union
LOS	line-of-sight
NLOS	non-line-of-sight
PSPNet	pyramid scene parsing network
PVT	position, velocity, time
ReLU	rectified linear unit
ResNet	residual neural network

1. INTRODUCTION

GNSS (global navigation satellite system) receivers are currently a core technology, enabling many applications by providing accurate and reliable position, velocity and time information, all around the world. As an example, every smartphone and most cars have GNSS positioning service available. Modern GNSS receivers are able to provide even centimeter-level 3D positioning accuracy in real time [1]. With applications such as autonomous driving, the accuracy and reliability of the positioning has become more and more important.

The core of GNSS positioning is to calculate the receiver position, velocity, and time (PVT) based on the propagation time of the received GNSS signals. Knowing the satellite positions, the propagation time can be converted into distance, assuming a direct path. Using at least four such measurements enables determining the receiver position and time. However, GNSS receivers are increasingly deployed in urban environments and other challenging conditions, where many satellites are behind buildings or other objects. In that case, the direct line-of-sight (LOS) signal cannot always reach the receiver but the signal can still be received via reflections. This poses a problem to GNSS positioning, because the propagation path, and thus propagation time, is longer than for a direct signal. [2]

To avoid errors due to using non-line-of-sight (NLOS) signals when computing the position of the receiver, it would be beneficial to know when the satellites are blocked by objects. This would allow to either ignore, downweight, or even try to correct the NLOS measurements. Multipath related problems, such as NLOS signals, are one of the biggest remaining problems in GNSS-based positioning [2].

In this thesis, we study the use of a sky-pointing fisheye camera to continuously analyze the receiver's environment. With the setup, we can record the surroundings while driving a car, and map the visible satellites into the image, as illustrated in Figure 1.1. The research question that the thesis aims to answer is the following:

- Can NLOS satellites be reliably identified from fisheye videos with image processing techniques?

Thus, the objective of the work is to find an accurate method for classifying the satellites as LOS or NLOS using the visual information of the images. Semantic segmentation

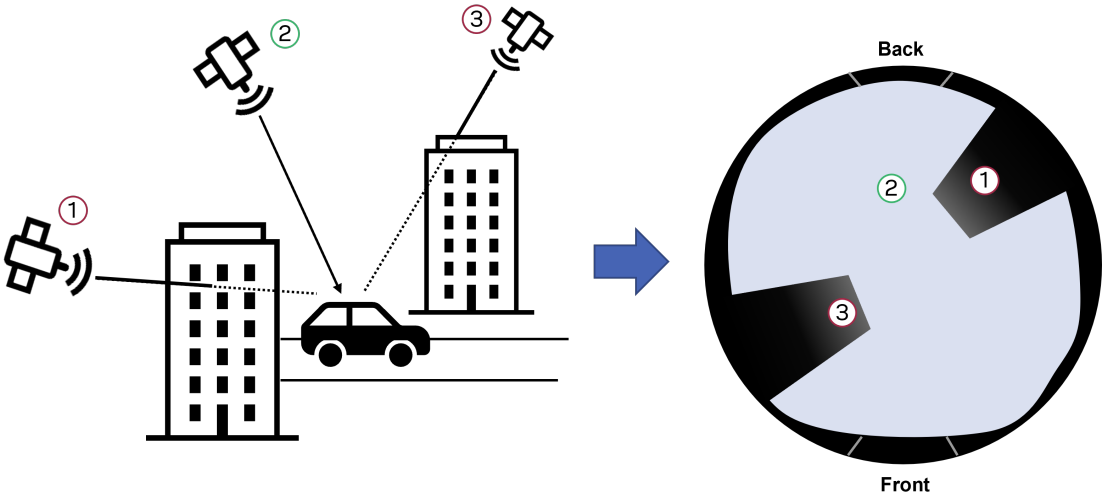

Figure 1.1. *Example 3D scene and the resulting fisheye image when the camera is on top of the car. The camera captures the whole field of view, and by projecting the satellites onto the image, the NLOS signals (1 and 3) can be identified.*

of the sky is a way to achieve this. While adding a camera to a GNSS setup to classify satellites at runtime might not be unfeasible, this algorithm is intended to serve testing and analyzing purposes. This is why we desire not to use additional data for the classification, and therefore we also do not have restrictions on the inference time or the capacity of the network.

We segment the fisheye images into LOS and NLOS areas using neural networks and make the classification of each satellite based on the segmentation in the location of the satellite. Neural networks are suitable for the task since they can utilize color, texture, and semantic information to find representative features for accurate classification. In the recent years, deep learning has become very powerful in segmentation, and convolutional neural networks can segment images highly accurately, often outperforming traditional image processing techniques, e.g. [3]. We train several popular semantic segmentation networks on a hand-crafted dataset and compare their performance in order to find the best one for this problem.

The thesis is structured as follows. Chapter 2 explains the background to better understand the problem of NLOS in GNSS and how semantic segmentation works. Related work on NLOS detection from fisheye sky images is presented as well. The segmentation models and performance metrics used in this study are introduced in Chapter 3 and the used dataset in Chapter 4. Chapter 5 covers the training and evaluation while Chapter 6 presents and discusses the obtained results. Finally, Chapter 7 summarizes the study and discusses the meaning of the results as well as possible future work.

2. BACKGROUND

NLOS and multipath propagation are fundamental problems in GNSS positioning, which is why techniques to mitigate their effects are of high interest [1]. The use of a fisheye camera for detecting the NLOS satellites is a know approach [1], and since fisheye cameras are affordable and easily available nowadays, it is a sensible method to consider for e.g., driving applications where the camera can be placed on top of the vehicle. While driving, the surroundings can change rapidly so the line-of-sight status of the different satellites constantly changes.

To detect the NLOS satellites from fisheye images, we need to know if the satellite position within the image belongs to the sky area (LOS) or not (NLOS). Semantic segmentation is a method to classify the image pixels into determined class labels such as sky. The segmentation can be done in various ways, but the most popular and effective method is to use convolutional neural networks, which can process challenging data with highly accurate results [4] [5].

2.1 Line-of-sight propagation and multipath

In conventional open sky conditions, the GNSS receiver is able to receive the signals of all satellites in its horizon on a direct path, which is called the LOS propagation path. As GNSS-based positioning and timing solutions have become more robust, reliable and precise, GNSS has been increasingly deployed in more challenging situations, such as urban environments, where the signals are often received via multiple paths due to reflection, diffraction, and scattering from objects between the transmitter and receiver. Multipath components always arrive later than the direct signal, which distorts the correlation function in the GNSS receiver, which causes errors to the PVT solution. While many error sources have been reduced by GNSS augmentations and modernization, multipath remains as one of the most significant sources of error in positioning. [2]

Even when the direct signal path is completely obstructed, a significant fraction of the signal power can still reach the receiver via multipaths, which is called the NLOS reception. Figure 2.1 illustrates a situation where some satellites are blocked by buildings but the signal is received through reflection or diffraction. While the LOS satellites can also produce reflections, if the delay in arrival compared to the direct path is large, receivers

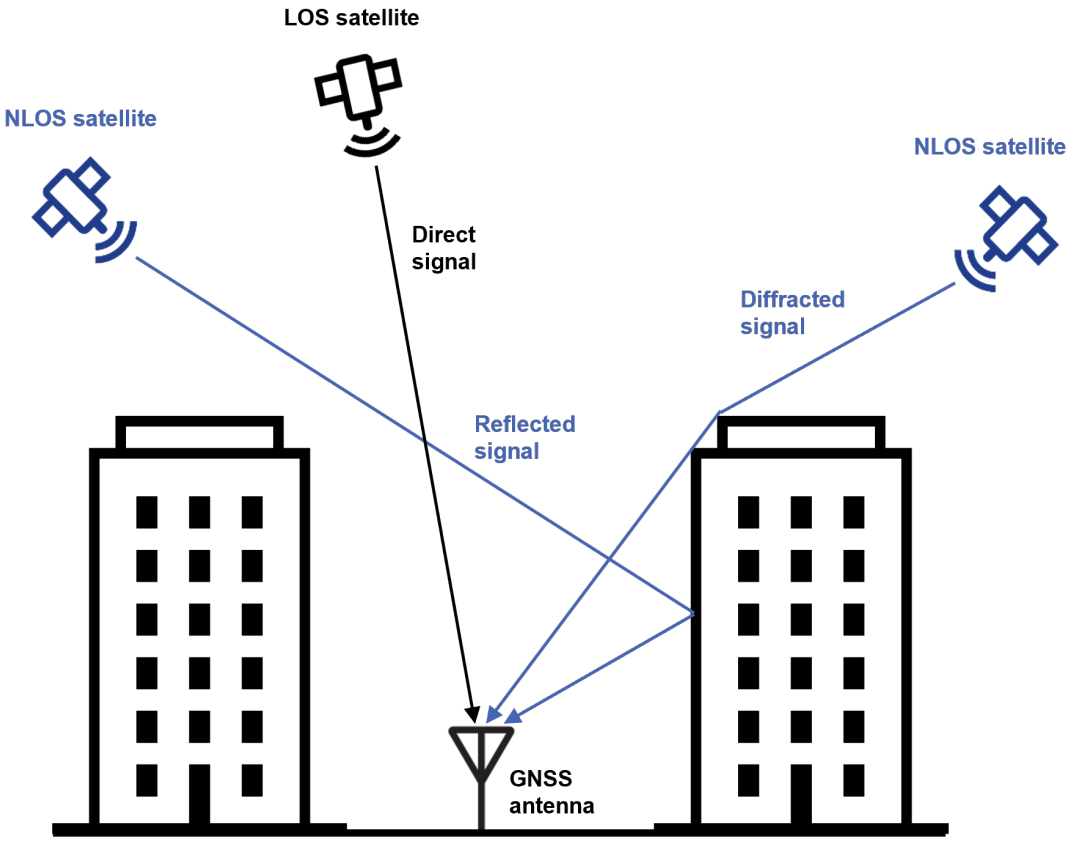

Figure 2.1. *LOS and NLOS propagation paths in urban environment*

can typically reject the multipath [2]. In NLOS reception, where the LOS signal is not received, the measurement errors in pseudorange, i.e., the approximated distance between the satellite and receiver, can be hundreds of meters [1]. The longer propagation path of a reflected or diffracted signal makes the satellite appear to be further away than it is in reality. Without the LOS signal it is difficult to know the true pseudorange, which is why signals from NLOS satellites should be downweighted or rejected if there are enough other measurements available [1].

The signal power of received NLOS signals is typically weak, but with receivers becoming more sensitive, NLOS signals are also received more often. However, the amount of attenuation depends on the reflecting surface. Water, glass, and metal are particularly reflective, and the signal can be attenuated by as little as 2-3 dB. In diffraction, the attenuation increases in proportion to the diffraction angle, and typically diffracted signals can be received at up to 5° deflection angle. [1] Signals from low-elevation satellites are more likely to produce multipath components, and they are also more often obstructed than higher-elevation satellites. However, using lower-elevation signals improves positioning precision, which is why it is desired to use them even with the increased risk of multipath and NLOS. [2]

Because of the large measurement errors caused by NLOS reception, many techniques

for mitigating these errors have been developed. With careful antenna design, some of the reflected signals can be sufficiently attenuated. There are also a number of ways to minimize the impact of NLOS to the navigation solution, most of which try to detect the signals affected by multipath or NLOS so that they can be downweighted or excluded. Some indicators are e.g., reduced or fluctuating signal power or low elevation angle, or inconsistencies in comparison with signals from different satellites. Additional data of the environment can also be used, such as images of a sky-pointing camera from which the NLOS satellites can be identified, as studied in this thesis. [1]

2.2 Fresnel zones

When a satellite is behind a building, it is clear that the direct path is blocked and the satellite can be considered non-line-of-sight. However, depending on the size and material of the obstacle, the LOS signal may also be received through obstructions, although attenuated. Streetlights and thin treetops are examples of such situations. On the other hand, even if the LOS path would be free of obstacles, the signal might be significantly weakened by nearby obstacles. The signal path is not a simple ray between the satellite and receiver, but instead defined by Fresnel zones.

Fresnel zone is an ellipsoid-shaped volume between a radio transmitter and receiver, which defines the propagation path of the LOS signal. There are multiple nested Fresnel zones, of which the first, or primary, zone indicates the region where the signal is the strongest, and it should be kept mostly clear of obstacles. The radius of the Fresnel zone n at a certain distance is defined as [6]

$$R_n = \sqrt{\frac{nd_1d_2\lambda}{d_1 + d_2}}, \tag{2.1}$$

where d_1 is the distance from the receiver, d_2 is the distance from the transmitter, and $\lambda = \frac{c}{f}$ is the wavelength defined by the speed of light c and the frequency of the signal, f. Figure 2.2 depicts the ellipsoidal Fresnel zone regions. In GNSS, the distance of the satellites from the Earth is about 20 000 km, and the region of interest is usually only a few hundred meters from the receiver, since further away the Fresnel zone is always mostly free of obstacles. Thus, we can say that $d_2 \approx d_1 + d_2$, and the equation for the primary Fresnel zone ($n = 1$) simplifies to [1]

$$R_1 = \sqrt{d_1\lambda}. \tag{2.2}$$

In order for the signal to not be significantly impacted by obstacles, the primary Fresnel zone should ideally be about 80% clear of obstruction, and it must be at least 60% clear [7]. In a typical city scenario, the rooftops of nearby apartment buildings might be

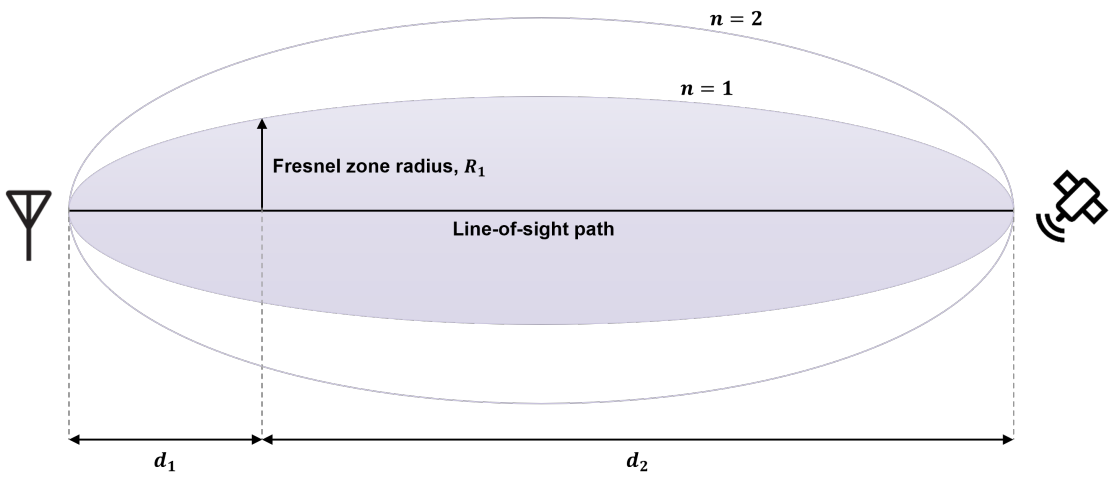

Figure 2.2. *Fresnel zones 1 and 2 between the receiver and transmitter*

at about 30 meter distance from a street. GNSS signals operate at 1-2 GHz frequencies, and the most used signal, GPS L1 C/A, operates at frequency 1575.42 MHz. If we insert these numbers to equation 2.2, the Fresnel zone radius is 2.4 meters. Thus, if the direct signal path is in the sky 2.4 m away from the edge of the roof and there are no other obstacles, the whole Fresnel zone is clear and we can be sure that the satellite is LOS. However, if we are closer to the edge, it is not certain if the LOS signal can be acquired, since part of the Fresnel zone is blocked. In this transition region the status of the satellite changes from LOS to NLOS when enough of the Fresnel zone becomes obstructed, already before reaching the building edge.

2.3 Semantic segmentation

Semantic segmentation is a computer vision task where each pixel of an image is assigned a class label. The label might be an object, such as a car, human or tree, or a background instance, such as sky or road. The number of classes can be chosen freely. Whereas in image classification, the whole image is given one class label, segmentation extends the classification to pixel-level. The purpose is to recognize distinct items in the image. An example can be seen in Figure 2.3, where an image has been segmented into several class categories. A similar method is object detection, but it requires the object to fit inside a bounding box. Semantic segmentation can be used for more irregularly shaped objects, such as the sky. [8]

There are many different applications where semantic segmentation is used. Some of the main industries are autonomous driving and robotics, which highly benefit from the precise image maps semantic segmentation can produce. For instance, for self-driving vehicles it is crucial to identify the road and obstacles. For these purposes the speed of the segmentation model is important as well. [9] Semantic segmentation is also used in healthcare to detect abnormalities in medical images. It helps radiologists and doctors to

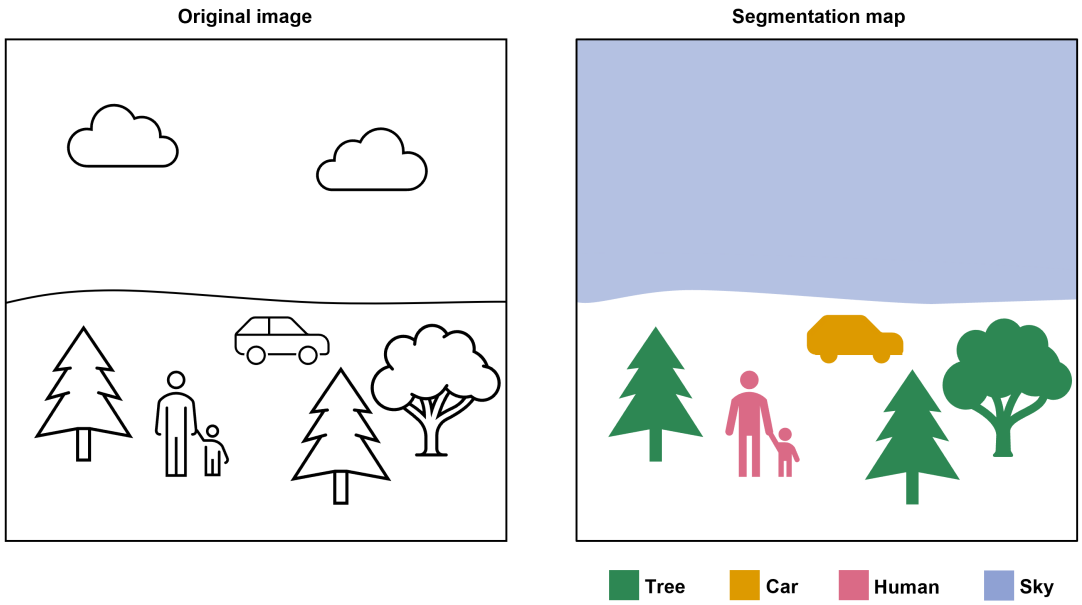

Figure 2.3. *Semantic segmentation of an image into different class labels*

spot and diagnose diseases, such as cancer, thus possibly saving lives of the patients. [5] Similarly, in industrial inspection semantic segmentation can be used to detect defects in materials [10]. Other applications are aerial and satellite imagery, where the terrain must be segmented. That can help, for instance, to track deforestation [11] or to examine areas of natural disasters like a flood [12].

The current state-of-the-art technique for semantic segmentation is deep learning [4] [5]. Convolutional neural networks (CNN) are able to combine color, texture, and semantic information, which enables highly accurate segmentation results. However, there are also more traditional image segmentation techniques developed before the rise of deep learning, and they utilize colors and low-level texture information. The advantage of those is that they are often much more lightweight than a CNN, and do not require a training set.

One of the simplest methods of semantic segmentation is thresholding, which divides the image pixels into two classes based on if the pixel value is larger or smaller than a threshold value. This is called image binarization. It is useful if the pixel values of two classes differ a lot. Thresholding can also be used as a part of other segmentation algorithms [13]. Adaptive and automatic thresholding methods exist as well [14]. Another approach is to utilize edge detection, which can be done using image gradients [15]. The edges help to find the different segments of the image. In region-based methods, such as watershed algorithms, the segmentation starts with seed pixels and looks for similarities between neighboring pixels, growing the regions until the whole image is segmented [16]. A similar method, but without the need of seed points, is clustering algorithms, which is nowadays the most popular image processing based segmentation approach [17] [18].

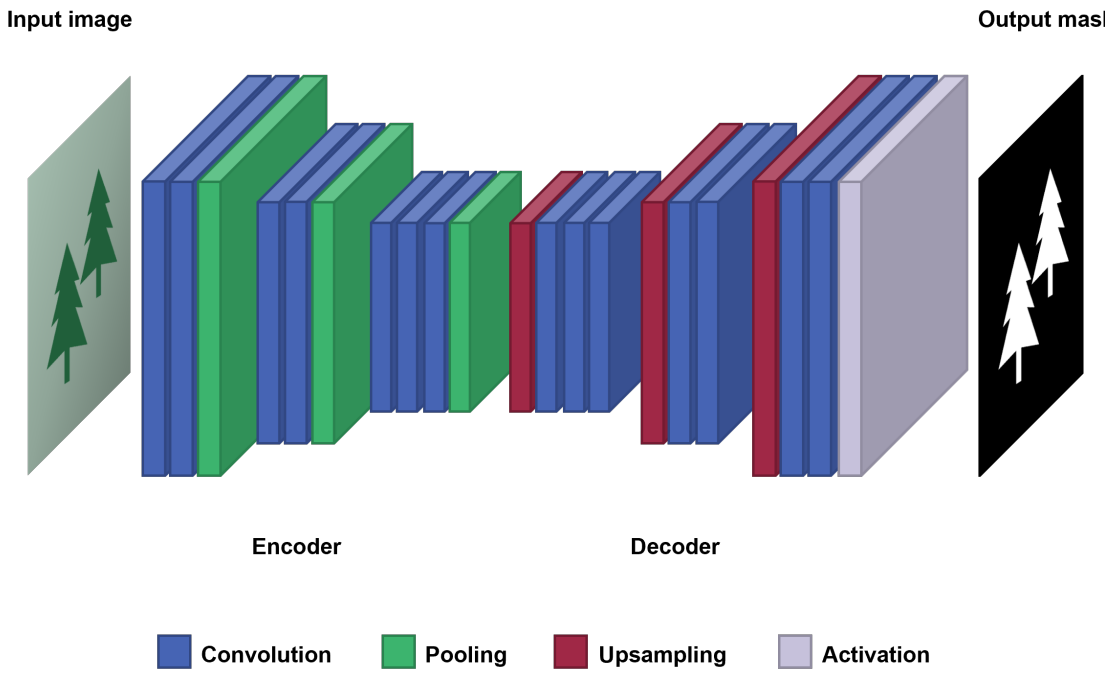

Figure 2.4. *An encoder-decoder architecture. The encoder downsamples the image while extracting features and the decoder upsamples the image back to the original resolution.*

These algorithms cluster together pixels with common attributes, forming the different segments.

2.4 Segmentation with neural networks

Convolutional neural networks are known for their capability to extract representative features from images, which is why they are very suitable for semantic segmentation. In the traditional CNNs that are used for classification, the input is downsampled more and more as we go deeper in the network and concentrate on increasingly more high-level features, and finally, a fully connected layer at the end yields the class label prediction [19]. In segmentation, however, fully connected layers are not needed, making the network fully convolutional, but instead upsampling is required in order to produce a segmentation mask with the same resolution as the input image [3].

The most common architecture for segmentation networks is an encoder-decoder structure, which is illustrated in Figure 2.4. The encoder part is used for the feature extraction, while also downsampling the image with pooling operations. The upsampling part is called a decoder, and it mirrors the encoder, upsampling the image back to the original size. Since information is lost when reducing the resolution, the upsampling layers typically use information from the previous pooling layers in order to produce finer segmentation results [3] [9]. The final activation layer produces the output segmentation map where each pixel is assigned a class label.

The convolutional layers are the core of the CNN and they perform the convolution operation to the input by sliding a small kernel across the image and computing the dot product between the kernel and a local image region. The kernel, or filter in multidimensional case, operates as the feature detector and contains the weights that are tuned in the training process. After each convolutional layer, there is a non-linearity layer, such as a rectified linear unit (ReLU) [20], which introduces non-linearity to the network. The pooling layers reduce the spatial size so that computational load is decreased, and there are different types of pooling operations. For example, max pooling takes the maximum value from a small neighborhood. As the resolution decreases with the pooling layers, the convolutional layers also look for more high-level features. The upsampling layers can perform either simple interpolation or upsampling with an unpooling operation or transpose convolution. [21] Finally, the output activation layer can use e.g., the softmax activation function to classify each pixel [4].

Segmentation models often use well-performing classification networks as a backbone. The hypothesis is that if a model works well for a classification task, it will extend well for a similar segmentation task also. The backbone CNN can be used as the encoder of the segmentation network by just removing the fully connected layers, and the decoder can then be build based on the encoder. For efficient training of the network, the encoder can even be initialized with weights pretrained on a classification task. [22] [9]

2.5 Related work on non-line-of-sight detection from fisheye sky images

Sky detection from fisheye images is a known approach for NLOS GNSS signal identification and it has been studied previously using various methods. The earlier studies are often based on region extraction or edge detection, whereas the more current approaches utilize neural networks. Each of those methods can produce highly accurate results even though it is difficult to compare the studies since they use different datasets and performance metrics.

Already in 2009 and 2010, Cohen et al. [16][23] studied sky segmentation with fisheye images. They utilized the color and texture information of the image by calculating the morphological texture and color gradient and combining them with a fixed, adaptive or supremum methods. After that they segmented the image into regions with a color watershed algorithm and finally classified the regions to sky or not-sky classes with the K-means clustering algorithm. Their methods achieved around 95% classification rates.

In their article from 2015, Kato et al. [24] exploited the movement of obstacles in the fisheye video to segment the sky. They first divided the image into regions with K-means and then performed feature point matching with the SIFT (scale-invariant feature transform)

algorithm as well as template matching. Based on that, they were able to calculate the traveled distance of the regions and classify them as obstacles if the distance is large and sky if it is small. They manually evaluated that in 84% of about 3000 images the whole sky area was correctly detected. Also the GNSS positioning accuracy was improved when rejecting the satellites classified as NLOS. Of course, their method only works if the camera is moving.

El Merabet et al. [17][25][26] have also studied the use of color information in sky detection in 2016 and 2017. They again first segmented the image into smaller regions, with a statistical region merging (SRM) algorithm. For them they calculated several different local color histogram descriptors, and compared the similarity between the characterized regions and learning databases of sky and non-sky regions to classify them. They were able to segment the sky almost perfectly, with around 99% classification accuracies.

Using edge detection for sky segmentation was studied by Sánchez et al. [27] in 2016 as well as Gakne and O'Keefe [15] in 2017. They both used a Canny edge detector, which finds edges in high-gradient regions of the image. The sky area was then determined by using the floodfill operation. Gakne and O'Keefe assumed that the centre pixel of the image belongs to sky and then filled the following pixels as sky until an edge was reached. Sánchez et al. had a similar approach, but they utilized the information of the satellite C/N_0s, or carrier-to-noise density ratios, in addition to just the image. They used the projected locations of the satellites with C/N_0 higher than 45 dBHz as the starting points of the floodfill operation to define the sky. Both studies showed good results by following these approaches, although their methods have some limitations because of the assumptions that they made about the floodfill start.

In 2015, Gakne and Petovello [14] compared several image segmentation algorithms for identifying the sky from upwards-facing regular camera. The algorithms were Otsu, mean shift, HMRF-EM (hidden Markov random field expectation maximization), and graph cut. The Otsu's method, which is the simplest of them, performed clearly the best with 95% segmentation accuracy for cloudy images and 81% for sunny. The method is an automatic image thresholding algorithm which finds the separating threshold value by minimizing intra-class intensity variance.

Horide et al. [28] used a simple fully convolutional neural network for fisheye sky segmentation in their article from 2019. Their architecture was based on the classification network VGG16. Even though they had a very limited amount of training images, they were able to significantly reduce the GNSS positioning errors after excluding the satellites classified as NLOS. A similar approach was chosen by Lee et al. [29] in 2020. They used the Resnet50 backbone, and in most of their experiments were able to improve the positioning accuracy compared to other methods, when they compared their generated sky mask to a 3D city model to find the position.

Neural networks were also used for regular and fisheye, although not sky-pointing, image segmentation by Romera et al. [30][31] in 2017 and 2019. With their encoder-decoder architecture called the ERFNet (efficient residual factorized network), their segmentation performance for the sky class reached an IoU (intersection over union) score of 94% for regular and 90% for fisheye images.

The use of infrared [32][33] or ultraviolet [34] cameras has also been shown to be successful for the sky detection task. The benefit over a regular RGB camera is that there is no need for complex segmentation algorithms, but simple thresholding is enough to separate the sky. Furthermore, IR cameras work well even at night where RGB cameras are unusable, whereas UV cameras are robust to different weather conditions. IR cameras, however, are not, and IR sky images have even been used for cloud segmentation [35].

3. RESEARCH METHODOLOGY

There exists various popular neural network architectures for semantic segmentation. One of the earliest very successful models that is still used today is the U-Net [36], while DeepLab is one of the current state-of-the-art networks [4]. In this thesis, we studied both of those, as well as a few other models. They are all introduced in the following sections. For evaluating the performance of the models, we use several metrics which are also explained in this chapter.

3.1 Segmentation models

The segmentation models introduced here are U-Net, LinkNet, FPN (feature pyramid network), PSPNet (pyramid scene parsing network) and DeepLab. Each of them can use as a backbone different classification networks, and the backbones presented here are ResNet (residual neural network), MobileNet, Inception and EfficientNet.

3.1.1 U-Net

The U-Net model was developed by Ronneberger et al. [3] in 2015 at the University of Freiburg. It has been popular especially in biomedical image segmentation, but can be used for any semantic segmentation problems. The U-Net architecture follows the encoder-decoder design shown in Figure 2.4, but it has skip connections from each pooling layer to the corresponding upsampling layer, as illustrated in Figure 3.1.

Each of the encoder blocks in the original U-Net, which was not based on any classification backbone, consists of two stacked convolutions, each followed by a rectified linear unit (ReLU). Then the image is downsampled with a max pooling operation to reduce the computational cost when the number of feature channels increases. However, in the pooling process information is lost, which is why the skip connections are used. In the decoder blocks, after upsampling with up-convolution, the features are concatenated with the output of the ReLU from the corresponding encoder block. Then there are again two convolutions followed by ReLU activation so that the decoder is symmetric with the encoder. The concatenation of high-level and low-level features helps to get finer information, producing more accurate results also with little training data. The original U-Net

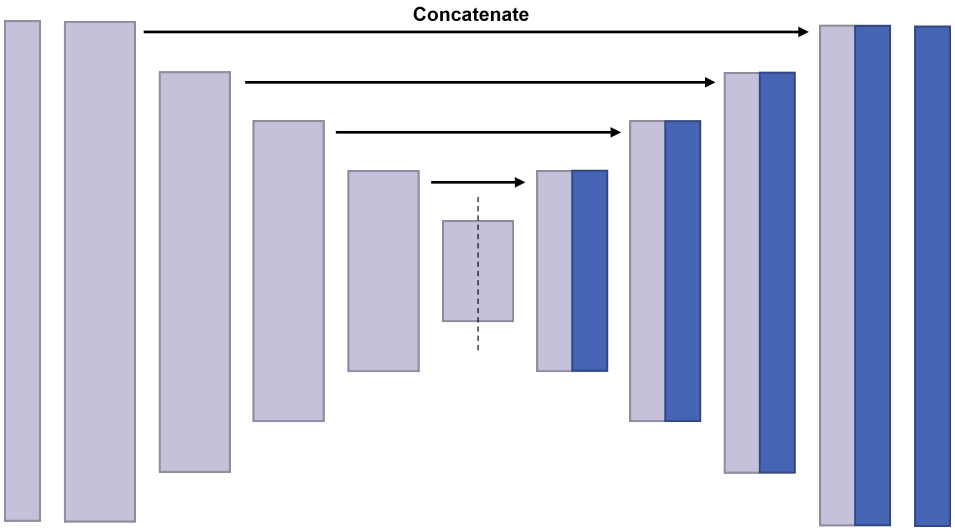

Figure 3.1. *U-Net architecture. To avoid information loss coming from downsampling, the encoder (lavender) features are concatenated with the corresponding decoder blocks (blue) using skip connections.*

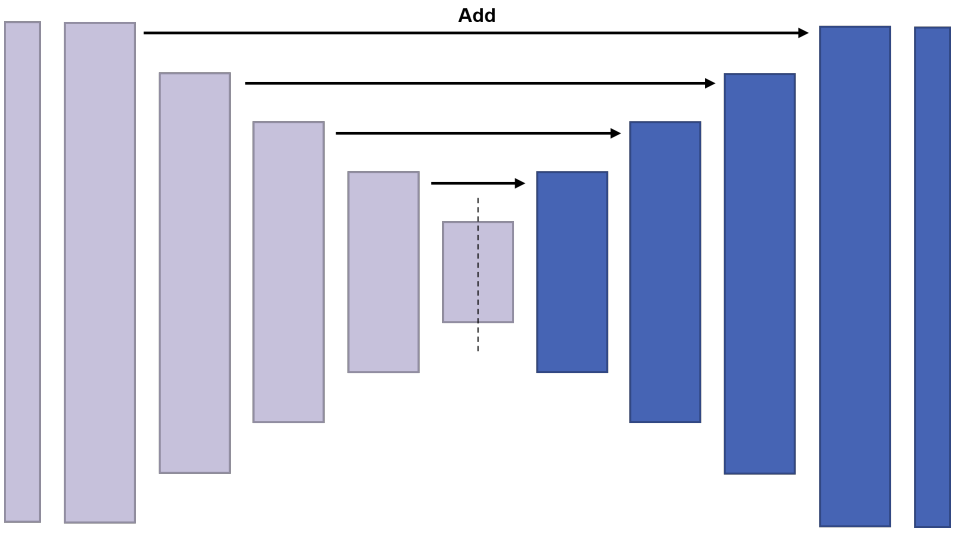

Figure 3.2. *LinkNet architecture. The architecture is similar to U-Net, but the skip connections add the encoder (lavender) and decoder (blue) features together instead of concatenating them.*

architecture has 23 convolutional layers in total. [3]

3.1.2 LinkNet

LinkNet is a very similar network to U-Net and it was designed in 2017 by Chaurasia and Culurciello [9] at Purdue University. The only difference between them is that LinkNet uses addition instead of concatenation in the skip connections, which is visualized in Figure 3.2. The LinkNet model is light and can perform real-time, making it useful for tasks such as autonomous driving.

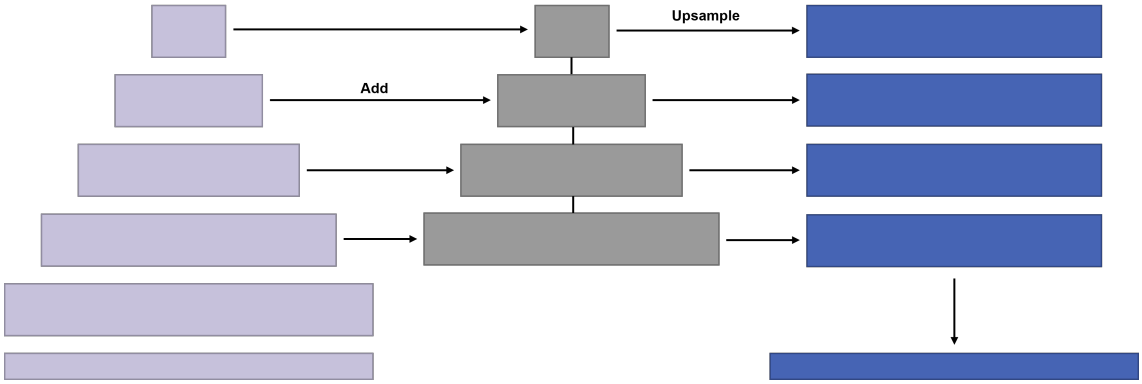

Figure 3.3. *FPN architecture. The bottom-up pathway (lavender) is the encoder, while the top-down pathway (gray) upsamples the image. The feature pyramid (blue) consists of upsampled features of each top-down block, which are then concatenated to produce the final segmentation map.*

Just like U-Net, LinkNet has an encoder-decoder architecture that addresses the problem of information loss in the downsampling operations by adding skip connections from each encoder block to the corresponding decoder block. However, when the features are added instead of concatenated, the number of parameters is reduced while still preserving some information from the previous layers. LinkNet usually performs similarly as U-Net or sometimes even better. Because LinkNet was designed for real-time applications, the original architecture uses as a backbone the light ResNet18 network, but any other model can be used as well. [9]

3.1.3 Feature pyramid network (FPN)

Feature pyramid network (FPN) is similar to LinkNet but it is slightly different from the simple encoder-decoder architecture. The model was developed in 2017 by Lin et al. [37] from Facebook and Cornell University. FPN is used especially for object detection and segmentation, since the feature pyramid structure helps in recognizing objects at different scales. Figure 3.3 shows the FPN architecture for segmentation as proposed in [38].

The FPN model has a bottom-up and top-down pathway, and it produces a feature pyramid. The bottom-up pathway is the encoder and it can be ResNet or any other network. In each block of the top-down pathway, there is a convolution with the same number of channels in each block, and also a convolution for the output of the corresponding bottom-up block to reduce the number of channels to that same number. Then they are combined using addition-based skip connections like in LinkNet. The image is upsampled until the second block of the encoder. A separate feature map is created from each of the top-down blocks with one more convolution, and for the segmentation they are then upsampled to the same size and concatenated together, to do the final prediction [38]. [37]

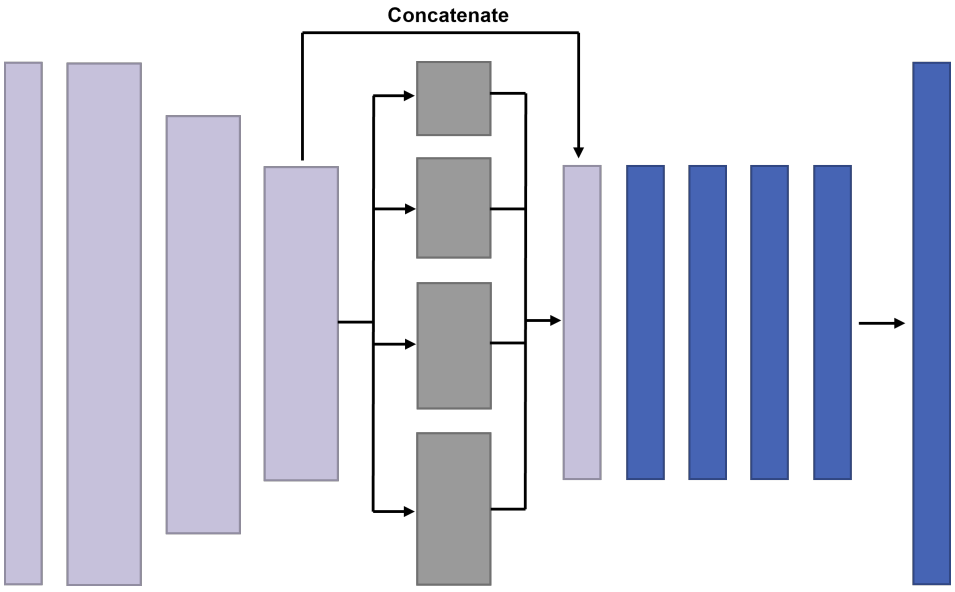

Figure 3.4. *PSPNet architecture. After the encoder (lavender), there is a pyramid pooling module (gray), which produces multiple feature maps (blue) that are concatenated to produce the segmentation map.*

3.1.4 Pyramid scene parsing network (PSPNet)

Pyramid scene parsing network (PSPNet) also exploits the idea of different sized feature maps. The model was proposed by Zhao et al. [22] from Chinese University of Hong Kong, again in 2017. PSPNet is designed especially for segmenting all the objects in a scene and it has a pyramid pooling module which helps to use global context information. The architecture is illustrated in Figure 3.4

PSPNet again starts with the backbone encoder, which was originally ResNet. The feature map of the last convolutional layer then goes to the pyramid pooling module, which has different sized pooling blocks. They separate the feature map to subregions and with pooling each block yields a different sized feature map. Each of them is then upsampled to the same size as the original feature map, so that they can be concatenated together, and with the original feature map. With a convolution the final prediction is done directly from that. [22]

3.1.5 DeepLab

DeepLab is one of the most successful semantic segmentation networks and it uses atrous convolution to achieve denser prediction without increasing computational cost. It was originally developed in 2016 by Chen et al. [39] from Google, while the latest version, DeepLabV3+, which has an encoder-decoder structure was published in 2018 [4]. The DeepLabV3+ architecture, which is fairly similar to PSPNet, is shown in Figure 3.5.

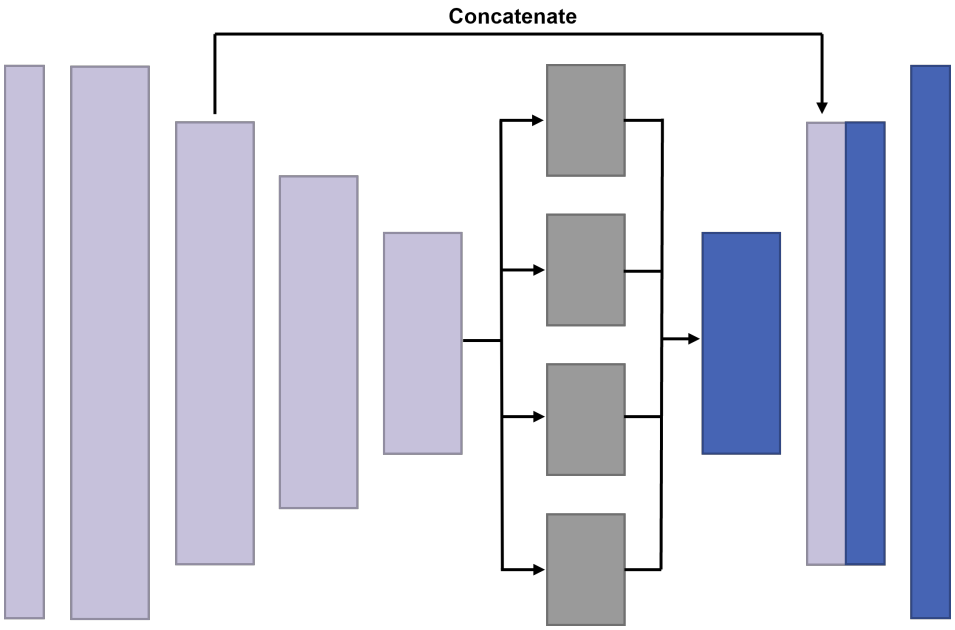

Figure 3.5. *DeepLabV3+ architecture. After the encoder (lavender), there is an atrous spatial pyramid pooling module (gray), which produces multiple feature maps. In the decoder (blue), they are concatenated and upsampled, and there is also one skip connection which concatenates low-level encoder features.*

As usual, the model first has the encoder to extract features. However, in the last few layers the downsampling with pooling is replaced with atrous, or dilated, convolution. It means that the convolution filter has a dilation rate, which determines gaps of one or more pixels in the convolution kernel, so that the field of view increases while keeping the number of parameters. In addition, there is an atrous spatial pyramid pooling module, where the atrous convolution is applied to the last feature map with different dilation rates to again study the features at different scales. The outputs are then concatenated together, and in the decoder, upsampled once and concatenated with the corresponding sized encoder features with reduced number of channels. Finally, a few more convolutions are applied before upsampling to the original resolution and doing the final prediction. [4]

3.1.6 Backbone networks

Many of the introduced segmentation networks were originally build based on the ResNet backbone, which is a classification CNN. As explained in Section 2.4, segmentation models can use as an encoder any classification network. This can be done because the purpose of the encoder is the same as of a classification model, to extract features. If the segmentation model backbone is changed, the network architecture naturally changes, but the key characteristics like the shortcuts and pyramid modules are kept. The same segmentation model with a different backbone can have a slightly different performance, similarly as the same backbone on a different segmentation model can as well.

The highly popular residual networks (ResNets) were developed in 2015 by He et al. [19] from Microsoft. The idea of ResNet are the residual connections that enable deeper networks without performance degradation or added complexity. The residual connections are shortcuts that skip one or more convolutional layers, and the input is added to the output of the skipped layers before activation. The ResNet architecture simply has convolutional layers stacked and every other layer there is a shortcut connection skipping two layers. The layers consist of a convolution, usually with a 3x3 filter, followed by batch normalization and ReLU activation. In the end there is a global average pooling and fully connected layer. The different ResNet versions have e.g., 18, 34, 50, 101, or 152 layers in total.

MobileNet is a light CNN designed for mobile vision applications, and it was published in 2017 by Howard et al. [40] from Google. The model utilizes depthwise separable convolution to reduce computational cost. It means that regular N KxKxM convolution filters, where N is the number of output channels, M is the number of input channels, and K is the kernel size, are replaced with a single KxKx1 convolution for each input channel, called depthwise convolution, followed by N 1x1xM convolutions, called pointwise convolution. After both convolutions, there is batch normalization and ReLU. The improved MobileNetV2 has these convolutions in reversed order, and after them another 1x1 convolution without nonlinearity, as well as residual connections skipping these three layers [41].

The Inception model was introduced already in 2014 by Szegedy et al. [42] from Google, and later on improved versions of it have been published. The network is quite complicated, including many tricks to improve the accuracy and speed. The main idea is to use convolutions with different filter sizes in the same layer, as well as max pooling. The outputs of them are then concatenated to form the input for the next layer. This helps to better extract features in different scales. The later versions of the Inception perform also factorization of the convolutions into a combination of smaller convolutions, reducing the computational cost [43]. Another idea of Inception was calculating the loss at different stages of the network and combining them with the final loss.

One of the most efficient CNN models in terms of accuracy and speed, the EfficientNet, was developed in 2019 by Tan and Le [44] from Google. The architecture is quite simple and similar to MobileNetV2, but instead the key is their approach to the scaling of the model. While e.g., ResNet is scaled larger by just adding more layers, the compound scaling method of EfficientNet performs balanced scaling in width, depth, and resolution. Width scaling means adding more filters in the convolutional layers, depth scaling adding more layers, and resolution scaling increasing the image spatial resolution. Following the compound scaling, EfficientNet has many different versions, B0 being the smallest and B7 the largest.

3.2 Performance metrics

Intersection over union (IoU) and F-score are popular metrics to evaluate the performance of semantic segmentation models [45]. To train the models, cross-entropy loss or Dice loss can be used [5]. The metrics are explained in the following sections.

The evaluation metrics are defined by true and false positives and negatives. A true positive means a sample that is correctly classified as belonging to a certain class, whereas a true negative is correctly classified as not belonging to it. A false positive is a sample that is classified as a given class when it should not, and a false negative is when it is classified as not belonging to a given class, when it actually should. [45]

3.2.1 Intersection over union (IoU)

The intersection over union score, or Jaccard index [46], measures the similarity of sample sets, and in semantic segmentation, it indicates how much the prediction and ground truth overlap with each other. The Jaccard index is noted as [45]

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{tp}{tp + fp + fn}, \tag{3.1}$$

where A and B are the sample sets to compare and tp is true positives, fp is false positives and fn is false negatives. The IoU score value is always between 0 and 1, so that 0 means no overlap and 1 is a perfect overlap.

IoU is a standard metric in segmentation because it gives a more fitting result than pixel accuracy, especially when the segmented object covers only a small part of the image. In that case, even if the object was not detected at all, pixel accuracy would be high, but IoU would give a lower score. In object detection IoU score is also useful to measure the overlap of the bounding boxes [47].

3.2.2 F-score and Dice loss

F-score, or Dice coefficient [48], is another common segmentation metric and it is similar to the IoU. The general F-score is the weighted average of precision and recall and it is defined as

$$F_\beta = \frac{(1 + \beta^2) \cdot tp}{(1 + \beta^2) \cdot tp + \beta^2 \cdot fn + fp}, \tag{3.2}$$

where $\beta = 1, 2, 3...$ and tp is true positives, fn is false negatives, and fp is false positives. The F1-score, which is the most common, is then [45]

$$F_1 = \frac{2tp}{2tp + fp + fn}. \tag{3.3}$$

The IoU and F1-score are positively correlated and F1 is always higher or equal to IoU. The difference is that IoU penalizes single mistakes more than F1, making IoU closer to the worst case performance and F1 to the average performance. If the F-score is subtracted from 1, we get the Dice loss.

3.2.3 Cross-entropy loss

Cross-entropy loss, or log loss, measures the accuracy of the predicted probabilities. In segmentation, if a pixel should be classified as 1, the cross-entropy loss is higher the smaller the predicted probability is, but especially when the probability is very close to 0, the loss increases rapidly. That means that especially confident wrong predictions are penalized. [49]

The cross-entropy loss is calculated as [49]

$$L = -\sum_{c=1}^{N} y \log(p), \tag{3.4}$$

where N is the number of classes, y is the ground truth, i.e., 0 or 1 depending on if class c is the correct label for that sample, and p is the predicted probability for the sample to be of class c. The total loss is the sum of the loss for each class label. Unlike IoU and F-score, cross-entropy is not confined between 0 and 1.

4. DATASET

The used fisheye video dataset was collected by u-blox in 2019. It contains 36 videos that are up to 3 hours long, totalling 55 hours of video. The videos were recorded in Zurich and the surrounding areas in Switzerland between May and October, in varying weather conditions and environments. The settings range from rural to urban with high buildings, and from sunny and clear to cloudy or even rainy. There are also tunnels, where the sky is not visible. All the videos were recorded during the day, so there are no very low light images. In addition to the videos, the dataset contains log files of the GNSS data at the time of each recording, providing information about the visible satellites. The recording setup is explained in detail in the next section.

4.1 Recording setup

The videos were recorded while driving a u-blox test vehicle and simultaneously collecting GNSS data. To achieve full coverage of the signal path from satellites to the receiver, the camera has a fisheye lens that can cover 360° horizontally and 90° vertically. The camera that was used to record the videos is an ELP camera [50] that has been available at Amazon for an affordable price. The setup records videos at resolution 1944x1944 pixels and frame rate of 10 frames per second.

The camera was mounted on the roof of the test vehicle, facing towards the zenith (directly upwards) so that it covers the whole horizon. Close to the camera there is a GNSS patch antenna, which is connected to a u-blox M8U evaluation kit [51] that contains the GNSS receiver and an inertial measurement unit (IMU). The camera and evaluation kit in their protective housing can be seen in Figure 4.1, as well as the installation on top of the van.

While recording the video, a log file of the GNSS receiver is also recorded. It contains e.g., the PVT-solution and the C/N_0 of each signal. This information updates every second. The recording software synchronizes with the video all the necessary data, such as the satellite locations and attitude information, which is needed for projecting the satellite positions onto the video frames. This operation is explained in the next section.

Figure 4.1. *Recording setup. The fisheye camera and GNSS evaluation kit are installed on top of a van in a protective housing.*

4.2 Satellite mapping

A GNSS receiver continually calculates the satellite positions with respect to the receiver position. The position is expressed in azimuth and elevation angles. The azimuth tells the horizontal direction of the satellite with respect to north, clockwise. A satellite directly in the east would have an azimuth angle of 90°. The elevation angle tells how high up in the sky the satellite is, so that a satellite on the horizon has a 0° elevation, and a satellite at the zenith has a 90° elevation. [1]

Using a calibrated camera and the GNSS evaluation kit, the azimuth and elevation of the satellites can be converted to image plane coordinates. First, the mapping from the spherical coordinates to north, east, and down coordinates of a local navigation frame can be done with the following equations: [1]

$$\begin{aligned} x &= \cos(a)\cos(e) \\ y &= \sin(a)\cos(e) \\ z &= -\sin(e), \end{aligned} \tag{4.1}$$

where a is the azimuth and e is the elevation. The projection onto the image plane is then defined as

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{K}_{3x3} \left(\mathbf{R}_{3x3} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \mathbf{T}_{3x1} \right), \tag{4.2}$$

where (u, v) is a point on the image plane, $\mathbf{K}$ is a camera matrix, $\mathbf{R}$ is a rotation matrix and $\mathbf{T}$ is a translation vector. In addition, the fisheye lens distortion has to be cor-

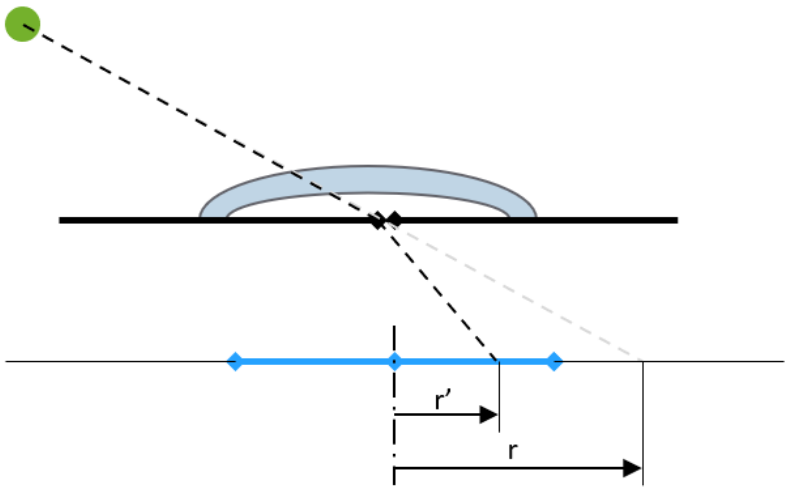

Figure 4.2. *Fisheye lens distortion. The green dot is projected onto the image plane at distance r', while the image plane point without the lens distortion would be r.*

rected. [52]

The rotation $\mathbf{R}_{world \to camera}$ consists of four steps. First, the rotation from world frame, meaning the north, east, down frame, to vehicle frame is defined by the attitude information provided by the receiver. Roll, pitch, and heading define the rotation in x, y, and z directions, respectively. Next is the rotation from vehicle frame to installation frame, which is constant. Then follows the rotation from installation frame to IMU frame, where we can use the attitude information from the IMU. Again, roll, pitch, and yaw define the rotation in x, y, and z directions. [1] Finally, the rotation from IMU frame to camera frame is constant that can be found with an extrinsic camera calibration [53]. The setup that was used to record the videos has been calibrated so the IMU to camera rotation matrix is known. The translation $\mathbf{T}$ is assumed to be zero.

The camera matrix $\mathbf{K}$ and the fisheye lens distortion coefficients can be found with an intrinsic camera calibration, so they are also known constants [53]. The fisheye lens distortion can be corrected with [52]

$$\begin{aligned} a &= \frac{x}{z}, b = \frac{y}{z} \\ r &= \sqrt{a^2 + b^2} \\ \theta &= \arctan(r) \\ r' &= \theta(1 + k_1\theta^2 + k_2\theta^4 + k_3\theta^6 + k_4\theta^8), \end{aligned} \tag{4.3}$$

where k_1, k_2, k_3, k_4 are the distortion coefficients, r is the radius of a point (a, b) on the image plane without the distortion and r' is the radius of a fisheye lens distortion corrected image plane point. Figure 4.2 illustrates the situation in one dimension. Finally, the distortion corrected image plane point will be

$$x' = \frac{r'}{r}a$$
$$y' = \frac{r'}{r}b. \tag{4.4}$$

The fisheye lens distortion correction is done after the rotation but before the camera matrix correction. [52]

Since the fisheye camera has 180° field of view, every possible visible satellite position can be mapped into the image pixels. In the fisheye image, while the close objects are distorted, the sky that is curved around the camera is projected so that each image pixel corresponds to approximately the same amount of sky. We can assume that the satellites are uniformly distributed on the image, although to be exact, on the northern hemisphere there are less satellites in the north and less with high elevation [1].

4.3 Annotation

We prepared an annotated dataset for the sky segmentation task. We first extracted frames from each of the 36 videos, taking a frame every 10 minutes to avoid having the same information in multiple frames. From those over 300 images, 224 images were manually annotated using the Image Labeler app in MATLAB [54]. Rainy images were excluded but several tunnels included. Since the segmentation will be applied to testing purposes, we can choose to not do a test drive in a rainy weather, but we always want to be able to detect tunnels in the test routes. Figure 4.3 shows an example of an annotated binary LOS/NLOS segmentation mask.

As can be seen in the example mask, the images were annotated so that small objects such as streetlamps and poles are marked as sky. As explained in Section 2.2, they do not significantly disturb the signal propagation since most of the Fresnel zone is clear, so a satellite behind them can still be considered line-of-sight. The same applies to mostly transparent objects such as very thin trees, transmission towers, and other lattices. Most trees were annotated as NLOS, however, since usually we cannot know how thick the tree layer is, and it might well be enough to block the signal.

From the annotated images, 40 were chosen to form the test set, and the remaining 184 were left as the training and validation set. The images in the test set are from three different videos that are not present in the training and validation set. We chose the test images to be diverse and include different environments and weather conditions:

- Video 1: sunny, few clouds, rural and urban, tunnels, reflections
- Video 2: cloudy, rural and urban, tunnels
- Video 3: overcast, rural and urban

Figure 4.3. *Original image and the annotated sky mask, where the line-of-sight area is colored yellow. Small lamps are considered part of the sky.*

The images in the training set have similar conditions but were recorded at a different time.

5. IMPLEMENTATION

We trained the U-Net, LinkNet, FPN, PSPNet and DeepLab with the annotated fisheye dataset. The training process and chosen hyperparameters are explained in the next section. We evaluated both the segmentation results and the satellite classification, and the evaluation methods are discussed in this chapter as well.

5.1 Training

The annotated dataset explained in Section 4.3 was used for the training. The images were downsampled from 1944x1944 to 512x512 using bilinear interpolation. This makes the training faster but also acts as a preprocessing step to filter the image, since not so much detail is needed for this task. We tried several image sizes and found that 512 works well and very little information is lost. In fact, training with this size seemed to result in better performance than higher resolution images. The 184 training images were further split into 148 training and 36 validation images, so that the split is random each time we train. After that, each training and validation image was augmented 6 times resulting in 1036 training and 252 validation images.

We deployed several different augmentation methods included in the Albumentations Python library [55]. We stayed with realistic augmentations that resulted in images that could be produced by the fisheye camera. For every augmentation, the image was flipped horizontally with 50% probability and always rotated either 0, 90, 180, or 270 degrees. This results in 8 different possible orientations of the image. With 20% probability, we added rain spatter to simulate the occluded lens during or after rain. Finally, we always applied random color jitter which changes the brightness, contrast, saturation, and hue. An example augmentation is shown in Figure 5.1.

The models were trained for 50 epoch using the Adam optimizer [56] and batch size 8. We also tried the stochastic gradient descent (SGD) optimizer and other batch sizes, but these yielded the best results. Depending on what works best for the specific model, the learning rate was 0.0001 or 0.00005 at first and decayed to 0.00001 if there was no improvement in the validation loss for 10 epochs. As the loss function we used a sum of the cross entropy loss and Dice loss. We considered both the sky and background classes to calculate the loss. Both the Jaccard loss and the focal loss [37] were considered as

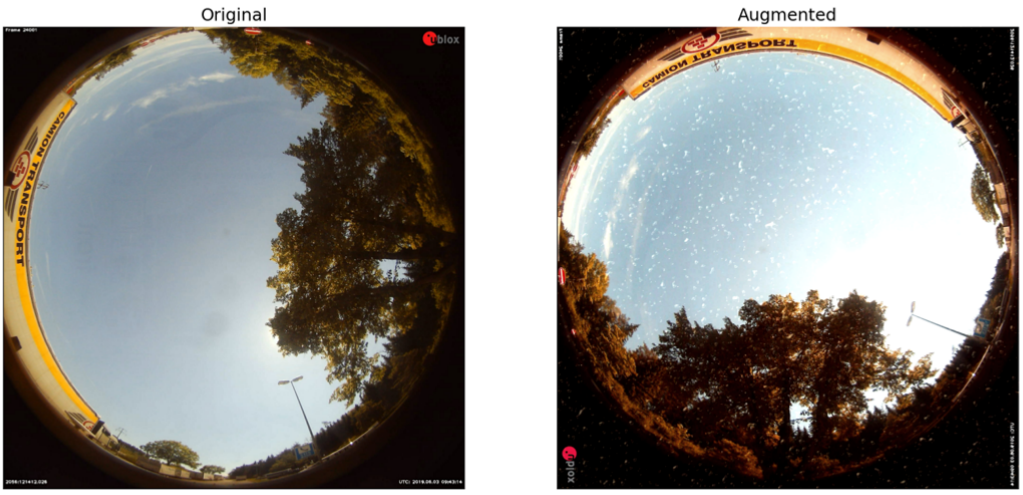

Figure 5.1. *Original and augmented image. The image has been rotated 90° and flipped, color jitter has been applied affecting especially brightness, and raindrops have been added.*

well, but led to worse results.

The training was done using the Tensorflow [57] and Keras [58] frameworks and the CNN and metrics implementations from the Segmentation Models library [59], except the DeepLab. All the backbone networks were initialized with weights pretrained on the ImageNet dataset [60] for faster convergence. After the training was completed, we saved the weights of the epoch with the best validation loss. Each model was trained 3-5 times to perform cross-validation since the dataset is so small.

5.2 Evaluation

For each training epoch, we calculated the mean IoU and F1-score for the training and validation set to see how the model is performing. To compare the different models, we calculated the mean of the best validation scores of each run. For our test set of 40 images that was described in Section 4.3, we calculated a few more performance scores as well. We computed the IoU score separately for the sky and background to see if there are differences. Since we downsampled the images for training, the predicted masks are also the same size as the training images. Therefore, we upsampled the predictions back to the original size with bilinear interpolation and calculated the IoU score also for the full resolution mask. Naturally, we also looked at all the test set prediction masks and many other, not annotated, images to see if the segmentation looks accurate also visually.

As a reference method we used the Otsu's thresholding method [61]. As explained in Section 2.5, the algorithm finds a threshold value which separates the image into two classes by minimizing intra-class intensity variance. Although the method is not as accu-

rate as some of the other previously used methods for sky segmentation, it is probably the simplest of them, with still rather high accuracy and beating some more complex segmentation algorithms [14]. We compared its performance on the test set against the neural networks to see if there is much benefit in using these highly complex models over the very simple one. We found that using the blue color channel works better than the grayscale image, so we applied the Otsu's thresholding only for the blue channel.

After the model has been trained, we can classify the in-horizon satellites as LOS or NLOS from a segmented image. We mapped the satellites into the image pixels as explained in Section 4.2, and classified each satellite based on the segmentation around the satellite location. We took a 100x100 pixels window around the satellite and if more than 60% of the pixels inside that window were segmented as sky, we classified the satellite as LOS, otherwise NLOS. This window approximates, with not perfect but sufficient accuracy, the Fresnel zone that was explained in Section 2.2. At the same time, it increases the robustness to small segmentation errors, when we do not classify the satellite only based on the pixel classification value at the exact location. As an example, small objects such as lamps can be segmented as NLOS even though we annotated them as LOS, because they differ from the sky region. With this approach we can avoid false classifications when satellites are behind these objects.

The classification process is illustrated in Figure 5.2. The segmentation mask is drawn on top of the image and each satellite is plotted and colored based on the classification. Both the window-based classification and the pixel classification of the direct signal path are visualized. We can see that in this example, the only satellite where these two differ is the one named S136 on the center right, and since the satellite is right on the edge of the building, according to the Fresnel zone theory the correct classification would be NLOS like it is when we use the classification window.

Since our test set is very small, we also used an additional method to evaluate the satellite classification, and thus also the segmentation accuracy. Signal-to-noise density ratio (C/N_0) is a measure of the strength of the signal. Therefore, the C/N_0 of LOS signals should be higher than NLOS, although there is not any exact threshold value. We extracted from the recorded log file the C/N_0 measurements of each signal and compared them with the classification of the corresponding satellite at the corresponding time. We plotted a C/N_0 histogram for the LOS and NLOS satellites, as well as the C/N_0s of individual satellites in time.

We looked at the C/N_0s of all satellites in the three videos from which the test images are, to see if there is a difference between the satellites classified as LOS and NLOS, and to detect possible errors not present in the test set of 40 frames. Although the signal strength can be affected by many other factors than just the LOS status, detecting an unusually low C/N_0 for a LOS classification or vice versa can help to discover problems

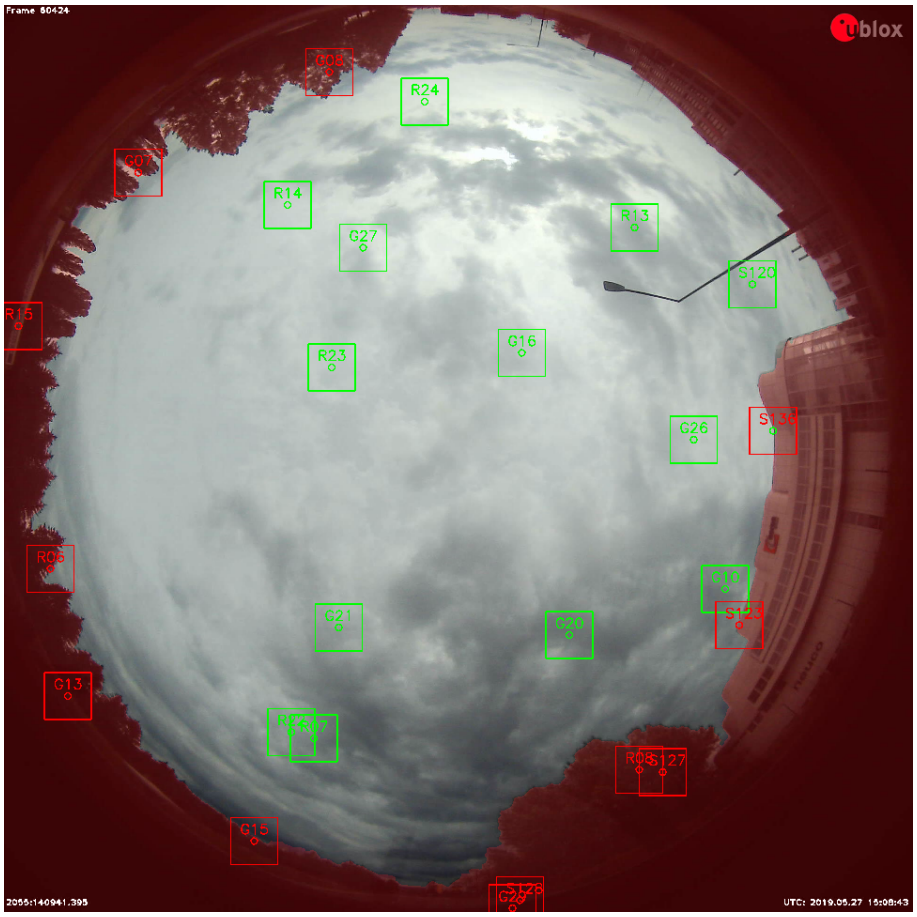

Figure 5.2. *Satellite classification based on the segmentation. The small circle marks the exact satellite location and it is colored according to the pixel classification value there (green=LOS, red=NLOS). The rectangle around it is the classification window colored according to the window-based classification. The NLOS segmentation mask is drawn on top of the image in red.*

with the segmentation. This evaluation method can be applied to all the video frames that have the C/N_0 measurement available, which in our dataset is every second.

6. EXPERIMENTS

The segmentation models were trained and evaluated as explained in Chapter 5. We were able to reach state-of-the-art segmentation results and also classify the satellites accurately. We found that the choice of the network does not make a big difference, and our relatively small training dataset is sufficient for the task. In this chapter, we present the obtained results in detail and analyze the reasons behind them.

6.1 Comparison of the models

The performance of each of the tested models are presented in Table 6.1. We can see that there is not much difference neither with the IoU scores of the different CNN models, nor the validation and test sets. The IoU scores range from 97-99%, which is an excellent score. The reference Otsu's method, however, performs significantly worse with an 83.7% test set score. Thus, we can say that using neural networks is indeed a useful approach, providing much higher accuracy than simpler methods.

The validation and test scores do not differ much from each other and sometimes the test score is even higher. This is not surprising because even though the test set images are from different videos than the training and validation, they are very similar. Since the test set is only 40 images, it is also possible that some model happens to work very well on that set, which might be the case with the EfficientNet backbone, which usually gives a higher testing than validation score. It is also important to note that the validation scores are calculated with different validation sets for each model, so they are not fully comparable. However, the validation scores in the table are an average of 5 runs with different splits for most of the models and 3 runs for some of the worse ones, so they are quite reliable. As we can see, both the validation scores with more images and the test scores with consistent images give similar ranking to the different models. However, the dataset is not large and diverse enough to have full confidence in the ranking especially for performance scores very close to each other.

While the differences are small, the U-Net with EfficientNetB6 encoder has the highest validation and testing IoU score. It is not surprising that the EfficientNet is the best backbone, since it has been shown to outperform the other classification networks [44]. For most models only the lightest B0 version was tested, but the deeper versions should give

Table 6.1. *Comparison of the models performance on the validation and test set*

Model	Validation mean IoU (%)	Testing mean IoU (%)
Otsu's method	-	83.69
U-Net - ResNet34	98.14	97.73
U-Net - ResNet50	98.18	97.95
U-Net - MobileNetV2	98.34	97.73
U-Net - InceptionV3	98.43	98.06
U-Net - EfficientNetB0	98.54	98.61
U-Net - EfficientNetB3	98.58	98.63
U-Net - EfficientNetB6	**98.60**	**98.71**
LinkNet - InceptionV3	**98.54**	**98.41**
LinkNet - EfficientNetB0	98.28	98.52
FPN - InceptionV3	97.92	98.25
FPN - EfficientNetB0	98.09	98.58
PSPNet - InceptionV3	97.86	97.95
PSPNet - EfficientNetB0	97.85	97.84
PSPNet - EfficientNetB6	98.09	98.42
DeepLabV3+ - ResNet50	**98.44**	**98.37**
DeepLabV3+ - EfficientNetB0	98.20	98.41

a little higher scores since they can produce more detailed prediction masks. That is why the U-Net, which performed the best on the EfficientNetB0, was also trained with the B6 version, and the fastest segmentation model PSPNet as well to verify that it cannot beat the U-Net with the bigger encoder either.

Why the U-Net is the best of the tested segmentation models can be due to several reasons. One is that the FPN, PSPNet, and DeepLab all have some kind of pyramid structure, which is designed to help in recognizing objects at different scales. Since our images are all in the same scale, this does not bring much improvement to our task. The LinkNet, which has a very similar architecture as the U-Net also has quite similar performance. Since U-Net concatenates the features of all the encoder blocks with the decoder blocks, it might help to produce a more detailed output than the other networks that only add the features or concatenate only from some encoder blocks. One possible reason is also that U-Net, being such a simple architecture, can also generalize better than some of the other models.

Figure 6.1 shows the performance of the different segmentation networks with the EfficientNetB0 encoder on one test set image. We can see that as the numbers suggest, also visually the performance is excellent and there is very little difference between the models. Only with the Otsu's method there is a clear mistake in the bottom left corner

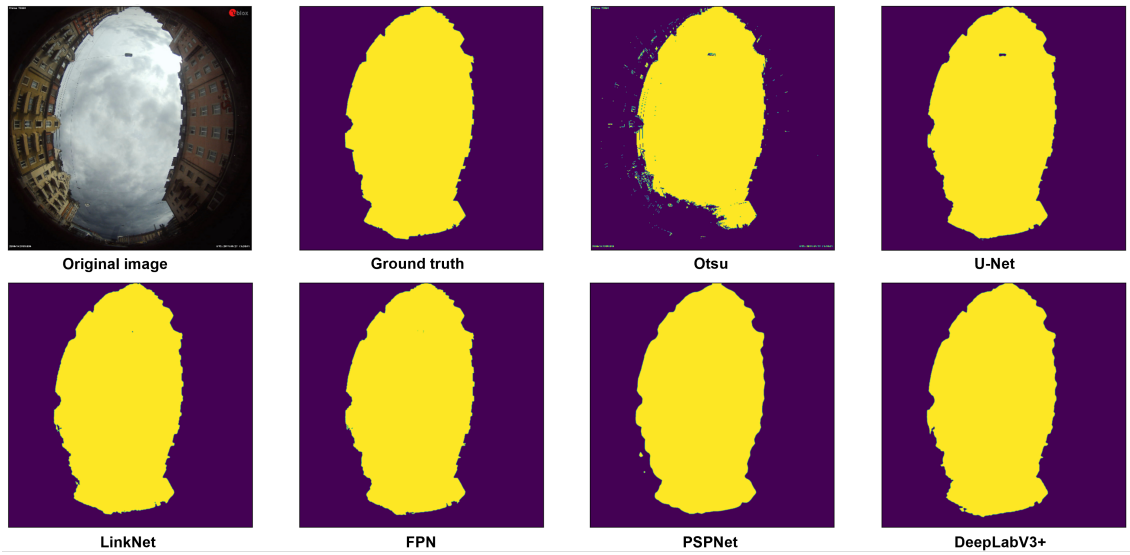

Figure 6.1. *Visual comparison of the models with EfficientNetB0 backbone on a test image*

where the darkest part of the sky is incorrectly segmented. The overall performance is still reasonably good but it is inevitable that with just thresholding, there will be errors when the color difference between the sky and background is not so clear.

Between the CNNs, the worst performing seems to be the PSPNet that has a little smoother result than the others, which is probably because of shallower depth and simpler decoder architecture. This explains the PSPNet having the lowest IoU scores. Otherwise, it is difficult to say which of the other 4 models is performing the best on this example. Recognizing the lamp in the middle is not considered a mistake, and all of the models sometimes detect them although usually not. The results look similar on other images as well, and with different backbones the difference also comes from the amount of detail. For the satellite classification, the precision of any of the CNNs would be good enough because not so much detail is necessary there. Of course, all the models sometimes have significant segmentation errors as well but visually it is not obvious which of them have them more often.

High performance is often a trade-off with the speed of the model. This we can also see from Figure 6.2, where we have plotted the IoU of each model as a function of the training time of one epoch in our training environment. The inference time will correlate with the training time. We can see that while the models with EfficientNet backbones clearly have the best performance, they are also the slowest. On the other hand, PSPNet is clearly the lightest but always has lower performance. U-Net seems to generally be the second fastest, though, while also having high accuracy. However, if the speed of the model would be important for our application, we might want to consider e.g., LinkNet with InceptionV3 or U-Net with EfficientNetB0 instead of the U-Net with EfficientNetB6.

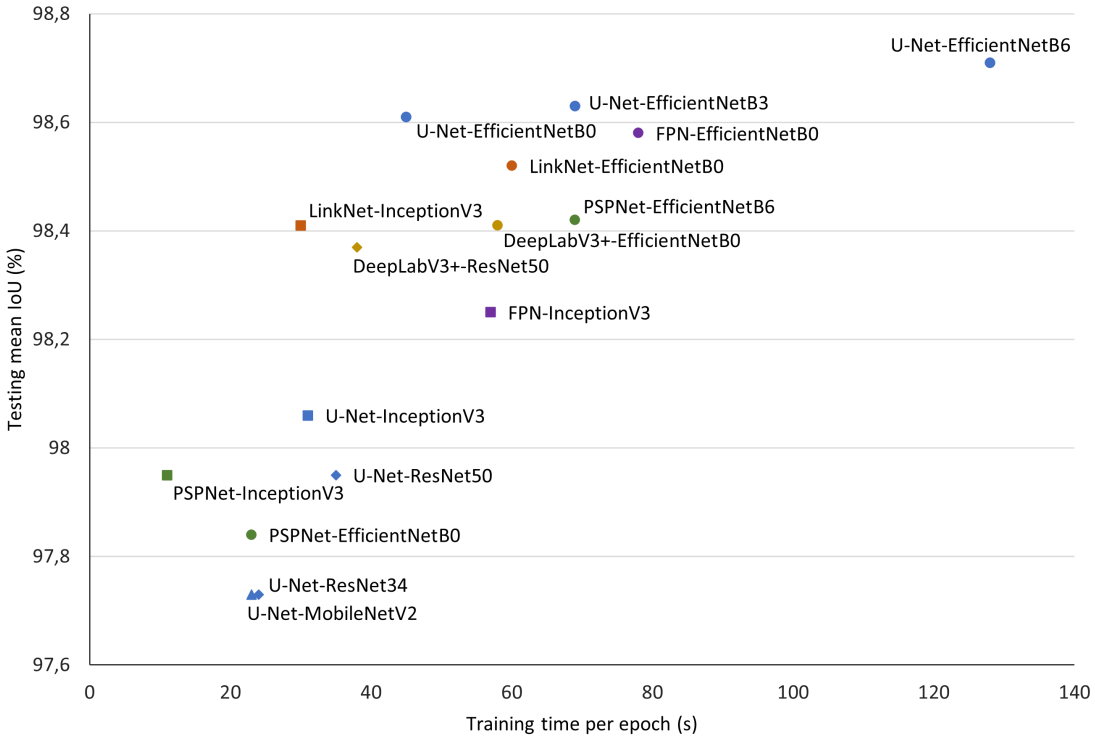

Figure 6.2. *Models performance compared to the training time*

6.2 Chosen model performance

Based on the model comparison, U-Net with EfficientNetB6 encoder was our model of choice. Some performance metrics for it are presented in Table 6.2. The IoU scores differ from Table 6.1 because those were an average of 5 runs but here we only show the best run. We can see that the test set mean IoU is 98.8% and the F1-score even higher, 99.4%. The scores are calculated for the downsampled mask which was used for training, but if we upsample the predicted mask and compare to the original one, we can see that the IoU score stays basically the same. That means that we do not lose much information in the downsampling so it is reasonable to use the smaller training size. Furthermore, we can see that the IoU for the sky is a little higher than for the background, but the difference is very small. That means that both are segmented accurately, and the difference is only because there is more sky than background area in the images.

The training curves of the chosen model are shown in Figure 6.3. We can see that the validation loss reaches the minimum really quickly and then, except for some fluctuations, it converges and starts to slightly increase. The training loss keeps still decreasing meaning the model starts overfitting, which is why we save the weights on the best validation loss and not in the end. The best epoch was number 19. The low loss already on the first epoch is because of the pretrained weights. The IoU curves look similar to the loss and the validation score is staying just below 99%.

Table 6.2. *Chosen U-Net-EfficientNetB6 model performance*

Metric	Value (%)
Validation mIoU	98.89
Testing mIoU	98.79
Testing mIoU, full resolution	98.77
Testing IoU, sky	98.87
Testing IoU, background	98.71
Validation F1-score	99.44
Testing F1-score	99.39

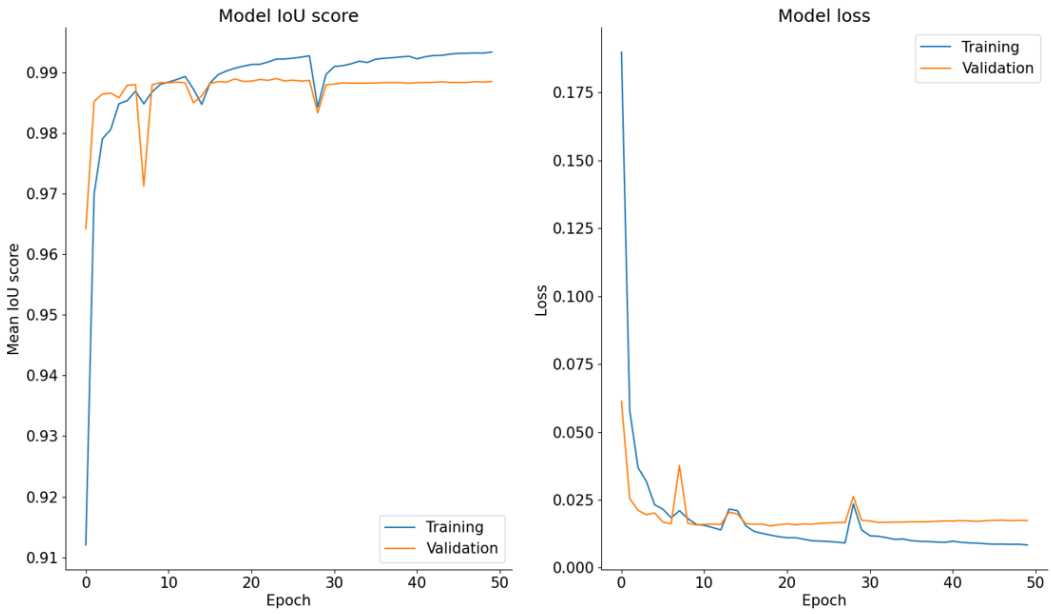

Figure 6.3. *Chosen U-Net-EfficientNetB6 model training curves for IoU and loss*

Figure 6.4 shows three example predictions on the chosen model. We can see that the results are very accurate and detailed. The first image is very bright and has quite heavy reflections, but our model handles it very well. In addition, there is a truck having white and blue colors that could be mistaken for sky, but it is not the case with this model. Also the small poles are correctly segmented as LOS, although segmenting them as NLOS would not be a big mistake.

On the second example, the sky is very different with dark overcast, which is not a problem for the model either. We can also see how the trees are handled. As mentioned before, the images were annotated so that the very thin trees that clearly do not bother the satellite signal were segmented as sky. We can see that the model has learned this distinction as well, but since it is not obvious where to draw the line, the prediction and

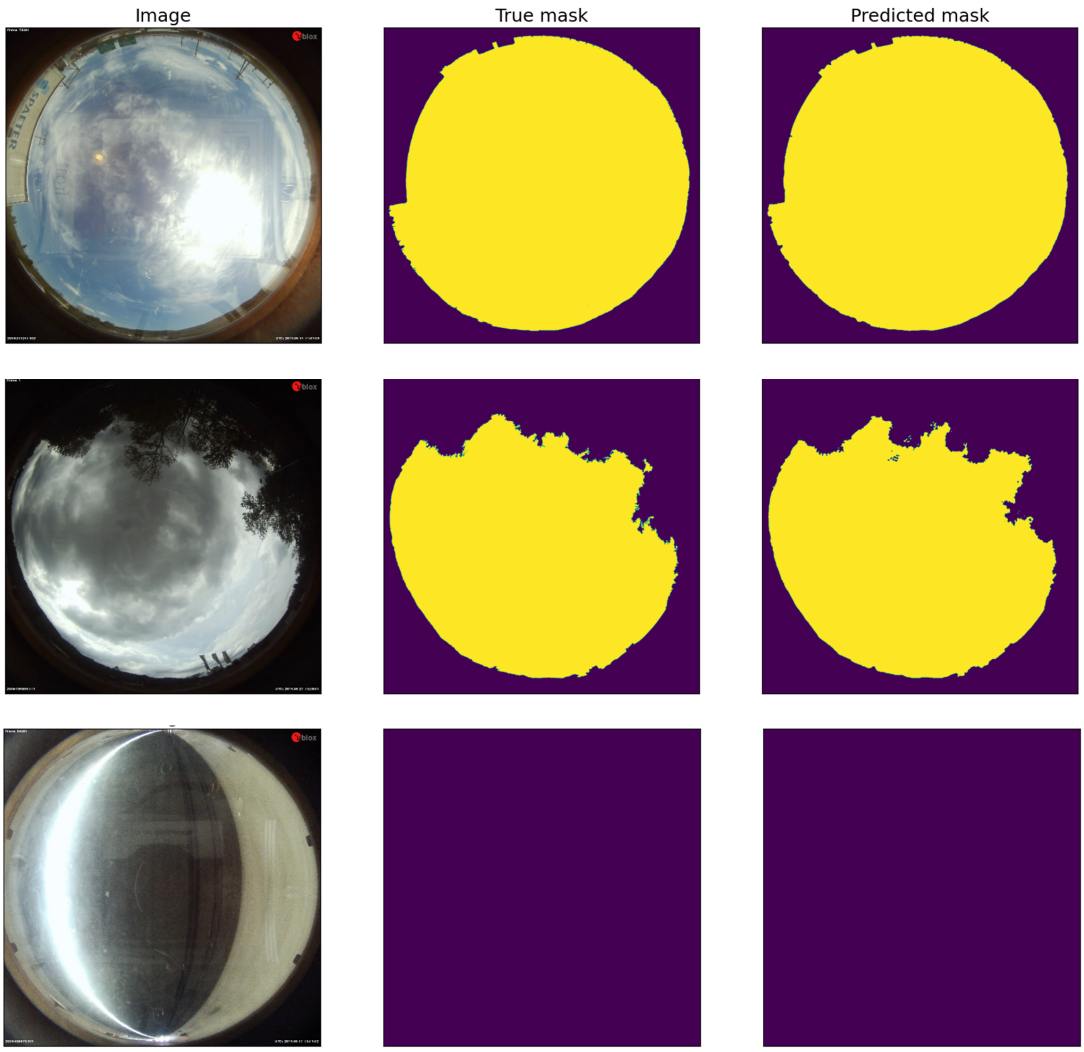

Figure 6.4. *Example predictions on a sunny, cloudy, and tunnel image*

truth differ slightly. It is possible that the CNN model does a better job in this than a human annotator, though. Because of manual annotation, it is impossible that the performance scores would be at 100%.

The third image is taken inside a tunnel, which is not an easy case either as we can see from the lamp on the ceiling and white walls. However, the model is able to perfectly segment the image as all NLOS. The Otsu's method would not segment tunnel images correctly because it always divides the image into two classes, and the same applies to many other possible methods. In fact, it is likely that no traditional image processing based methods could handle tunnels without separate tunnel detection. Apparently none of the previous studies on the sky segmentation topic considered situations where there is no sky visible. With our approach, adding just a few tunnel images, which do not even need annotation, to the training set seems to produce very accurate tunnel detection.

Of course, sometimes our model also makes mistakes. Figure 6.5 shows three examples of small segmentation errors. On the first image, the problem is with the white truck on the

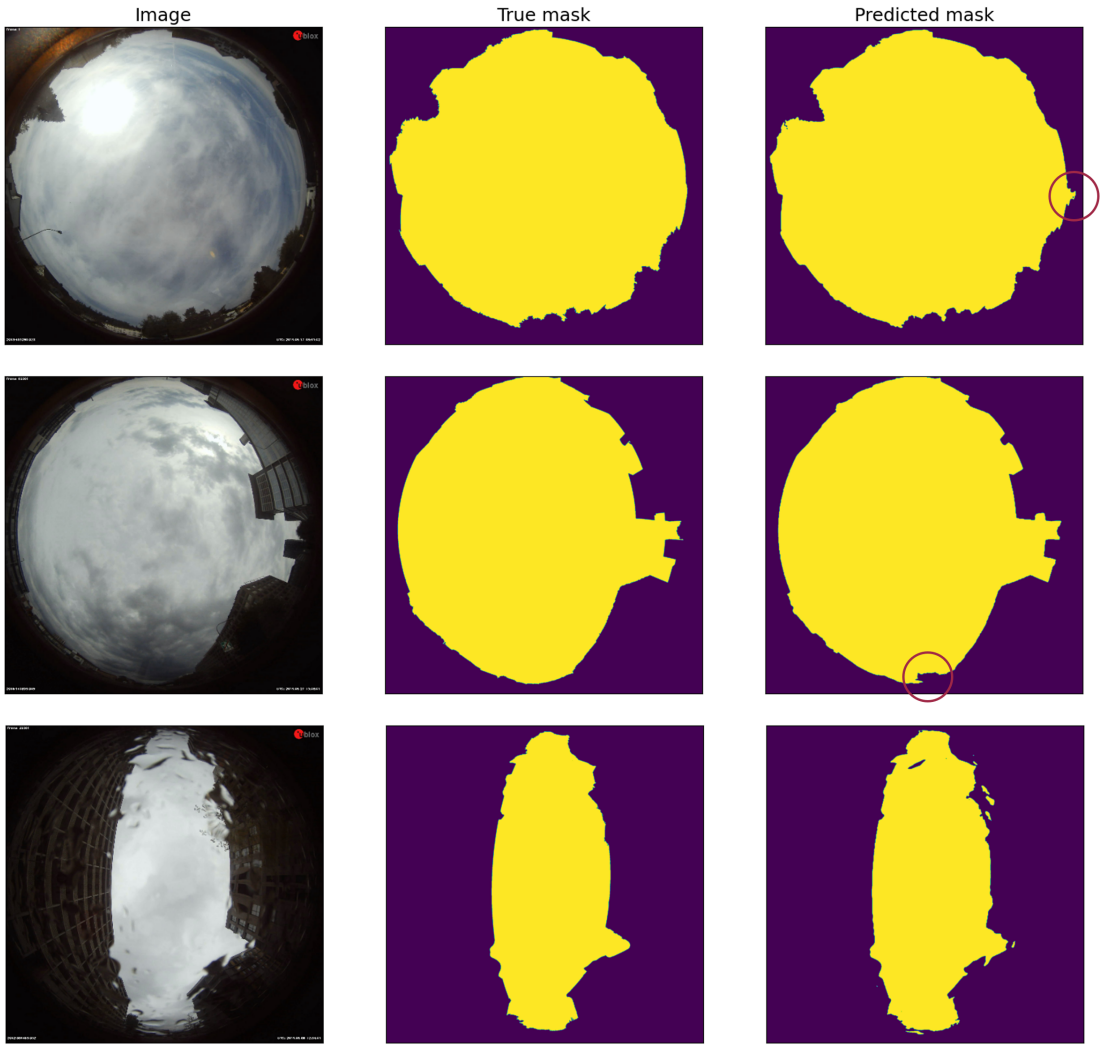

Figure 6.5. *Examples of segmentation errors. All the errors are caused by similar-colored sky and background.*

right, which is partly segmented as sky. It is easy to see why that is a difficult case since it does look a lot like sky, and even though the model usually handles well sky-like colors, sometimes they do cause errors. On the second example the issue is the opposite, the darkest part of the sky being incorrectly segmented. Again, the mistake makes a lot of sense, and while the model overall works well with very dark skies, sometimes it causes small errors.

The last example is a rainy image, which were not included in the training set, so it is natural that they are not perfectly segmented. Even though in the augmentation we added raindrops, the real rainy images look a little different with bigger and more irregular drops. The result is still reasonably good, though, and again it is very easy to see the cause of the errors. In fact, also the truth mask seems to have errors due to the raindrops, and that is why we decided not to annotate the rainy images, since it is so difficult or sometimes impossible to see where the skyline goes. If we would have rainy images with accurate

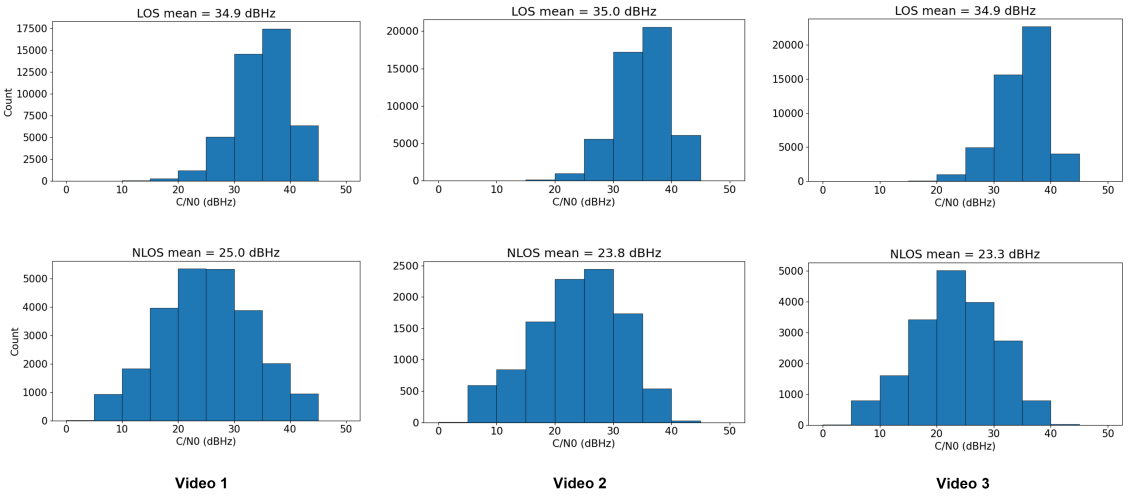

Figure 6.6. *C/N_0 histograms for the LOS and NLOS satellites in each test video during an hour*

masks in the training set, it is possible that the model would learn to handle them better. Unlike the previous examples, this image was not part of our test set but we still looked at the performance on the rainy images visually, and it usually looks similar to this one.

6.3 Satellite classification results

For each of the three test set videos (see Section 4.3), we segmented the frames every second for a clip of an hour, and classified the satellites as explained in Section 5.2. Thus, we processed 3600 images in each video. The C/N_0 histograms of those satellites can be seen in Figure 6.6. We can see that there is little difference between the 3 videos, meaning that the classification is not affected by different weather conditions, and as expected, the LOS C/N_0s are clearly higher than NLOS. Roughly speaking, the LOS signals are usually above 30 dBHz while the NLOS signals are below that. It should be noted that the 0 dBHz satellites are not plotted, because it means that the signal is not tracked (the receiver was not able to extract a message from the given satellite) which can also be due to other reasons than NLOS. Because of that, e.g., tunnels where none of the satellites are tracked do not show in this figure.

We could say that, with such setup as the one used in the test vehicle, signals below 20 dBHz are very unlikely to be LOS and signals above 40 dBHz very unlikely to be NLOS, and such cases are likely misclassifications. From the histograms we see that that happens very rarely, except for the video 1 where quite many above 40 dBHz signals are classified as NLOS. The explanation for this is a long static scene where a few satellites are partly behind leaves and classified as NLOS. Based on the signal strength we can consider them misclassifications, but the segmentation is still correct because we wanted

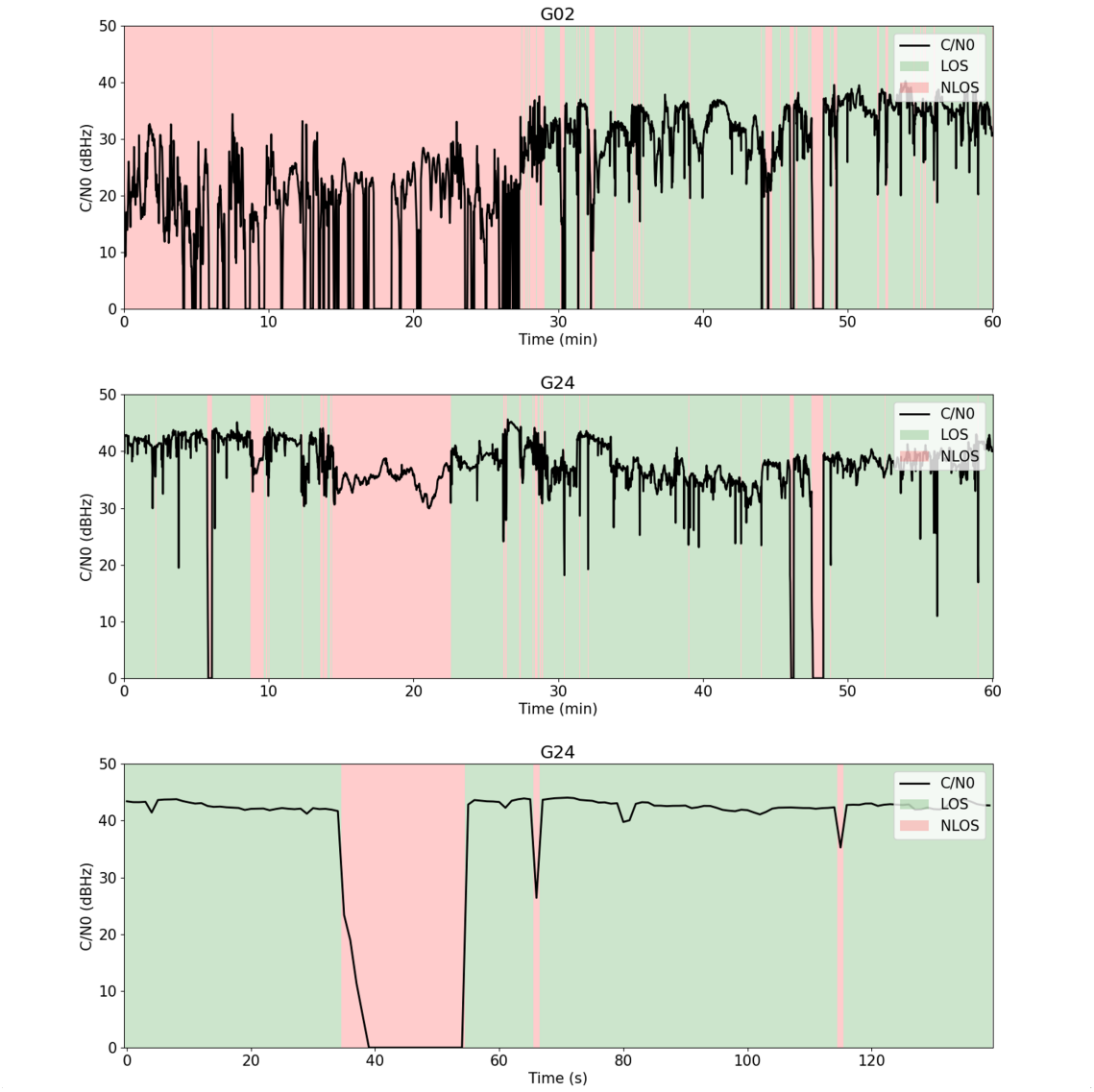

Figure 6.7. *Satellite C/N_0s and classification in time. In the first plot, the satellite is just rising above the horizon, the second one has a higher elevation, and the third one is a closeup of the second one, where the big drop is a tunnel.*

to segment trees as NLOS mostly. Looking at the videos, it looks like also the majority of other high C/N_0 (>35 dBHz) satellites are behind trees, or sometimes right on the edge of an obstacle, instead of wrong segmentation. However, usually in these cases the C/N_0 has still clearly dropped compared to a full LOS situation, so our choice to treat trees as NLOS seems reasonable.

If we plot the C/N_0s of individual satellites in time, we get more insights about the model performance in practice. This has been done in Figure 6.7 for two GPS satellites. The first two plots are an hour clip from the test video 1, so they correspond to the first histograms in Figure 6.6. The background color shows the classification at each point in time and we can see that there is a clear correlation with the C/N_0. The C/N_0 fluctuates a little all the time, though, which is why it is alone not the best measure of NLOS.

The plotted G02 is a low elevation satellite that is just rising above the horizon. At first, it is blocked so the C/N_0 is low and often dropping to zero, and our model correctly classifies it as NLOS. A little before 30 minutes, the satellite has risen above the trees and we see an increase in the signal strength and the classification changes to LOS. Because the elevation is still low, the signal frequently gets blocked behind obstacles which we see as drops in the C/N_0 that the classifier also detects. The model seems to work very reliably, although if we look closely we can see at least one misclassification at about 7 minutes. This is due to a limitation in our method: when coming out of a tunnel, the camera is blind for a few seconds because of the sudden change in brightness, meaning that the image is completely white and all satellites are classified as LOS. The classification will naturally be impossible there, but it would be easy to detect these cases and not do any classification.

The second satellite, G24, is higher up in the sky so it is mostly line-of-sight although it still sometimes gets blocked. The three times when the C/N_0 drops to zero are tunnels and they are detected well. The longer NLOS section around 20 minutes is the static scene where the satellite is behind a tree. The C/N_0 there is quite high but since it does clearly drop when the scene starts, the classification seems correct in this case also. The last plot is a closeup of the second one, and we see there the first tunnel. We can confirm that the classification is indeed correct for the whole length of the tunnel, and also the smaller drops are detected well.

7. CONCLUSION

Classifying the satellite signals as LOS and NLOS is beneficial in GNSS, and it can be done in various ways. This thesis studied the use of a fisheye camera with the satellites projected onto the image to perform the classification with the help of semantic segmentation of the image. The segmentation can also be done with different methods, and in this thesis we studied segmentation with deep learning and convolutional neural networks. We compared several popular CNN models that were expected to be suitable for the task.

Based on the study, we could confirm that NLOS satellites can be reliably identified from fisheye videos with image processing techniques. Using deep learning, the sky area can be segmented from the images accurately, which enables accurate satellite classification. Our best performing model, U-Net with EfficientNetB6 backbone, achieved state-of-the art sky segmentation performance, reaching 98.8% IoU and 99.4% F1-score on our test set. Our model works equally well in very sunny and cloudy conditions, and in particular it is also reliable in tunnels where no sky is visible. The most difficult cases are when the sky and obstacles have very similar colors. Our study of signal strengths further confirmed that our LOS/NLOS classification based on the segmentation is correct most of the time.

We found that there was little difference between any of the tested CNN models, and for the satellite classification task any of them would be sufficient. We also saw that not so much training data is needed, but performing light augmentations to increase the dataset size is beneficial. Furthermore, not so much detail is needed for the task so the images can be downsampled quite a lot, and it can be even advantageous. However, well representative and carefully annotated training data is important since the inferred segmentation results reflect the training data.

It is likely that with different CNN models that we did not try here, different versions of the tested models, or different architectural or training hyperparameters, we could reach slightly better segmentation performance than what we reported here. However, it would probably increase the computational cost as well as not improve the satellite classification much or at all. The model performance on rainy and low light conditions also remains to be studied, but it is possible that the accuracy there would be very good as well, if training data of such situations would be provided. Even without, the performance with raindrops

seems reasonably good. In addition, using the information of the previous video frames could be beneficial.

Our C/N_0 analysis indicated that trees are likely the most significant source of classification errors. The C/N_0 of satellites behind trees is often so high that it is possible that the correct classification would be LOS, which is often not the case. As the correct classification of the satellites behind trees is not clear, it might be a good idea to treat trees as a separate class instead of doing binary LOS/NLOS classification. The segmentation networks would probably learn to detect the trees very well. However, behind trees there can be other obstacles that are not visible but block the signals completely, which is why our approach might still be better.

The study showed that neural networks can segment the sky from fisheye images very accurately, and they bring a clear improvement compared to simpler methods. The approach is attractive not only because of the excellent performance, but also because of the easiness. Letting the CNN do the work of finding representative features and patterns reduces the need of manual work of developing highly complex image processing algorithms including e.g., preprocessing and considering different kinds of situations. With modern tools, training and deploying CNN models is easy and fast.

REFERENCES

[1] P. Groves. *Principles of GNSS, Inertial, and Multisensor Integrated Navigation Systems, Second Edition*. Artech House, 2013, pp. 13, 33–34, 301, 344, 401–407, 458–462. ISBN: 9781608070053.

[2] E. Kaplan and C. Hegarty. *Understanding GPS/GNSS: Principles and Applications, Third Edition*. Artech House, 2017, pp. 2, 591–614. ISBN: 9781630810580.

[3] O. Ronneberger, P. Fischer, and T. Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. Springer International Publishing, 2015, pp. 234–241. DOI: 10.1007/978-3-319-24574-4_28.

[4] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam. "Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation". *Computer Vision – ECCV 2018*. Springer International Publishing, 2018, pp. 833–851. DOI: 10.1007/978-3-030-01234-2_49.

[5] Z. Li, J. Zhang, T. Tan, X. Teng, X. Sun, H. Zhao, L. Liu, Y. Xiao, B. Lee, Y. Li, Q. Zhang, S. Sun, Y. Zheng, J. Yan, N. Li, Y. Hong, J. Ko, H. Jung, Y. Liu, Y.-c. Chen, C.-w. Wang, V. Yurovskiy, P. Maevskikh, V. Khanagha, Y. Jiang, L. Yu, Z. Liu, D. Li, P. J. Schuffler, Q. Yu, H. Chen, Y. Tang, and G. Litjens. "Deep Learning Methods for Lung Cancer Segmentation in Whole-Slide Histopathology Images-The ACDC@LungHP Challenge 2019". *IEEE Journal of Biomedical and Health Informatics* 25.2 (2021), pp. 429–440. DOI: 10.1109/JBHI.2020.3039741.

[6] *Fresnel Zone Clearance*. SoftWright. URL: https://www.softwright.com/faq/engineering/Fresnel%20Zone%20Clearance.html (visited on 06/07/2023).

[7] D. Coleman and D. Westcott. *CWNA: Certified Wireless Network Administrator, Official Study Guide*. Sybex, 2012, p. 126. ISBN: 9781118127797.

[8] *Semantic Segmentation*. MathWorks. URL: https://se.mathworks.com/solutions/image-video-processing/semantic-segmentation (visited on 02/17/2023).

[9] A. Chaurasia and E. Culurciello. "LinkNet: Exploiting Encoder Representations for Efficient Semantic Segmentation". *2017 IEEE Visual Communications and Image Processing (VCIP)*. 2017, pp. 1–4. DOI: 10.1109/VCIP.2017.8305148.

[10] X. Shi, S. Zhang, M. Cheng, L. He, X. Tang, and Z. Cui. "Few-shot Semantic Segmentation for Industrial Defect Recognition". *Computers in Industry* 148 (2023), p. 103901. DOI: 10.1016/j.compind.2023.103901.

[11] R. de Andrade, G. Mota, and G. da Costa. “Deforestation Detection in the Amazon Using DeepLabv3+ Semantic Segmentation Model Variants”. *Remote Sensing* 14.19 (2022), p. 4694. DOI: 10.3390/rs14194694.

[12] E. Nemni, J. Bullock, S. Belabbes, and L. Bromley. “Fully Convolutional Neural Network for Rapid Flood Segmentation in Synthetic Aperture Radar Imagery”. *Remote Sensing* 12.16 (2020), p. 2532. DOI: 10.3390/rs12162532.

[13] K. Siang Tan and N. Mat Isa. “Color Image Segmentation using Histogram Thresholding – Fuzzy C-means Hybrid Approach”. *Pattern Recognition* 44.1 (2011), pp. 1–15. DOI: 10.1016/j.patcog.2010.07.013.

[14] P. Gakne and M. Petovello. “Assessing Image Segmentation Algorithms for Sky Identification in GNSS”. *2015 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. 2015, pp. 1–7. DOI: 10.1109/IPIN.2015.7346956.

[15] P. Gakne and K. O’Keefe. “Skyline-based Positioning in Urban Canyons Using a Narrow FOV Upward-Facing Camera”. *Proceedings of the 30th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2017)*. 2017, pp. 2574–2586. DOI: 10.33012/2017.15223.

[16] A. Cohen, C. Meurie, Y. Ruichek, J. Marais, and A. Flancquart. “Quantification of GNSS Signals Accuracy: An Image Segmentation Method for Estimating the Percentage of Sky”. *2009 IEEE International Conference on Vehicular Electronics and Safety (ICVES)*. 2009, pp. 35–40. DOI: 10.1109/ICVES.2009.5400321.

[17] Y. El Merabet, Y. Ruichek, S. Ghaffarian, Z. Samir, T. Boujiha, R. Touahni, and R. Messoussi. “Horizon Line Detection from Fisheye Images Using Color Local Image Region Descriptors and Bhattacharyya Coefficient-Based Distance”. *Advanced Concepts for Intelligent Vision Systems*. Springer International Publishing, 2016, pp. 58–70. DOI: 10.1007/978-3-319-48680-2_6.

[18] Q. He, J. Shao, J. Pu, M. Zhou, S. Xiang, and W. Su. “FRFCM Clustering Segmentation Method for Medical MR Image Feature Diagnosis”. *Journal of Intelligent and Fuzzy Systems* 40.2 (2021), pp. 2871–2879. DOI: 10.3233/JIFS-189327.

[19] K. He, X. Zhang, S. Ren, and J. Sun. “Deep Residual Learning for Image Recognition”. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

[20] K. Fukushima. “Visual Feature Extraction by a Multilayered Network of Analog Threshold Elements”. *IEEE Transactions on Systems Science and Cybernetics* 5.4 (1969), pp. 322–333. DOI: 10.1109/TSSC.1969.300225.

[21] E. David and N. Netanyahu. “DeepPainter: Painter Classification Using Deep Convolutional Autoencoders”. 2016, pp. 20–28. DOI: 10.1007/978-3-319-44781-0_3.

[22] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. “Pyramid Scene Parsing Network”. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 6230–6239. DOI: 10.1109/CVPR.2017.660.

[23] A. Cohen, C. Meurie, Y. Ruichek, and J. Marais. “Characterization of the Reception Environment of GNSS Signals Using a Texture and Color Based Adaptive Segmentation Technique”. *2010 IEEE Intelligent Vehicles Symposium*. 2010, pp. 275–280. DOI: 10.1109/IVS.2010.5548018.

[24] S. Kato, M. Kitamura, T. Suzuki, and Y. Amano. “NLOS Satellite Detection Using a Fish-Eye Camera for Improving GNSS Positioning Accuracy in Urban Area”. *Journal of Robotics and Mechatronics* 28 (2016), pp. 31–39. DOI: 10.20965/jrm.2016.p0031.

[25] Y. El Merabet, Y. Ruichek, S. Ghaffarian, Z. Samir, T. Boujiha, R. Touahni, R. Messoussi, and A. Sbihi. “Hellinger Kernel-Based Distance and Local Image Region Descriptors for Sky Region Detection from Fisheye Images”. *Proceedings of the 12th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 4: VISAPP*. SciTePress, 2017, pp. 419–427. DOI: 10.5220/0006092404190427.

[26] H. Bour, Y. El Merabet, Y. Ruichek, R. Messoussi, and I. Benmiloud. “An Efficient Sky Detection Algorithm From Fisheye Image Based on Region Classification and Segment Analysis”. *Transactions on Machine Learning and Artificial Intelligence* 5 (2017). DOI: 10.14738/tmlai.54.3335.

[27] J. Sánchez, A. Gerhmann, P. Thevenon, P. Brocard, A. Ben Afia, and O. Julien. “Use of a FishEye Camera for GNSS NLOS Exclusion and Characterization in Urban Environments”. *Proceedings of the 2016 International Technical Meeting of The Institute of Navigation*. 2016, pp. 283–292. DOI: 10.33012/2016.13404.

[28] K. Horide, A. Yoshida, R. Hirata, Y. Kubo, and Y. Koya. “NLOS Satellite Detection Using Fish-Eye Camera and Semantic Segmentation for Improving GNSS Positioning Accuracy in Urban Area”. *Proceedings of the ISCIE International Symposium on Stochastic Systems Theory and its Applications* 2019 (2019), pp. 212–217. DOI: 10.5687/sss.2019.212.

[29] M. Lee, S. Lee, H.-F. Ng, and L.-T. Hsu. “Skymask Matching Aided Positioning Using Sky-Pointing Fisheye Camera and 3D City Models in Urban Canyons”. *Sensors* 20.17 (2020). DOI: 10.3390/s20174728.

[30] E. Romera, J. Álvarez, L. Bergasa, and R. Arroyo. “Efficient ConvNet for Real-Time Semantic Segmentation”. *2017 IEEE Intelligent Vehicles Symposium (IV)*. 2017, pp. 1789–1794. DOI: 10.1109/IVS.2017.7995966.

[31] Á. Sáez, L. Bergasa, E. López-Guillén, E. Romera, M. Tradacete, C. Gómez-Huélamo, and J. del Egido. “Real-Time Semantic Segmentation for Fisheye Urban Driving Images Based on ERFNet”. *Sensors* 19.3 (2019). DOI: 10.3390/s19030503.

[32] J.-i. Meguro, T. Murata, J.-i. Takiguchi, Y. Amano, and T. Hashizume. “GPS Multipath Mitigation for Urban Area Using Omnidirectional Infrared Camera”. *IEEE Transactions on Intelligent Transportation Systems* 10.1 (2009), pp. 22–30. DOI: 10.1109/TITS.2008.2011688.

[33] M. Petovello and Z. He. “Skyline Positioning in Urban Areas using a Low-cost Infrared Camera”. *2016 European Navigation Conference (ENC)*. 2016, pp. 1–8. DOI: 10.1109/EURONAV.2016.7530554.

[34] T. Stone, M. Mangan, P. Ardin, and B. Webb. “Sky Segmentation with Ultraviolet Images Can Be Used for Navigation”. *Robotics: Science and Systems 2014*. 2014. DOI: 10.15607/RSS.2014.X.047.

[35] G. Terrén-Serrano and M. Martínez-Ramón. “Comparative Analysis of Methods for Cloud Segmentation in Ground-based Infrared Images”. *Renewable Energy* 175 (2021), pp. 1025–1040. DOI: 10.1016/j.renene.2021.04.141.

[36] N. Siddique, S. Paheding, C. Elkin, and V. Devabhaktuni. “U-Net and Its Variants for Medical Image Segmentation: A Review of Theory and Applications”. *IEEE Access* 9 (2021), pp. 82031–82057. DOI: 0.1109/ACCESS.2021.3086020.

[37] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. “Feature Pyramid Networks for Object Detection”. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 936–944. DOI: 10.1109/CVPR.2017.106.

[38] A. Kirillov, K. He, R. Girshick, and P. Dollár. *A Unified Architecture for Instance and Semantic Segmentation*. URL: http://presentations.cocodataset.org/COCO17-Stuff-FAIR.pdf (visited on 04/19/2023).

[39] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. Yuille. “DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs”. *IEEE Transactions on Pattern Analysis and Machine Intelligence* PP (2016). DOI: 10.1109/TPAMI.2017.2699184.

[40] A. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications”. (2017).

[41] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. “MobileNetV2: Inverted Residuals and Linear Bottlenecks”. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 4510–4520. DOI: 10.1109/CVPR.2018.00474.

[42] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. “Going Deeper with Convolutions”. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 1–9. DOI: 10.1109/CVPR.2015.7298594.

[43] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. “Rethinking the Inception Architecture for Computer Vision”. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 2818–2826. DOI: 10.1109/CVPR.2016.308.

[44] M. Tan and Q. Le. “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks”. *Proceedings of the 36th International Conference on Machine Learning*.

Vol. 97. PMLR, 2019, pp. 6105–6114. URL: https://proceedings.mlr.press/v97/tan19a.html.

[45] Y. Cai, Y. Yang, Q. Zheng, Z. Shen, Y. Shang, J. Yin, and Z. Shi. “BiFDANet: Unsupervised Bidirectional Domain Adaptation for Semantic Segmentation of Remote Sensing Images”. *Remote Sensing* 14.1 (2022), p. 190. DOI: 10.3390/rs14010190.

[46] P. Jaccard. “The Distribution of the Flora in the Alpine Zone”. *The New Phytologist* 11.2 (1912), pp. 37–50. URL: http://www.jstor.org/stable/2427226.

[47] U. Seidaliyeva, D. Akhmetov, L. Ilipbayeva, and E. Matson. “Real-Time and Accurate Drone Detection in a Video with a Static Background”. *Sensors* 20.14 (2020), p. 3856. DOI: 10.3390/s20143856.

[48] L. Dice. “Measures of the Amount of Ecologic Association Between Species”. *Ecology* 26.3 (1945), pp. 297–302. URL: http://www.jstor.org/stable/1932409.

[49] *ML Glossary: Loss Functions*. Read The Docs. URL: https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html (visited on 06/01/2023).

[50] *ELP 5MP USB HD 180 Degree Panoramic Full Circle Lens Windows Linux Web Camera Wide Angle + USB Webcam Camera for Video Meeting, Machine Vision*. Amazon. URL: https://www.amazon.de/dp/B01NAEMG8N (visited on 02/27/2023).

[51] *EVK-8/EVK-M8*. u-blox. URL: https://www.u-blox.com/en/product/evk-8evk-m8 (visited on 02/27/2023).

[52] *Fisheye Camera Model*. OpenCV. URL: https://docs.opencv.org/4.x/db/d58/group__calib3d__fisheye.html (visited on 06/02/2023).

[53] *What Is Camera Calibration?* MathWorks. URL: https://se.mathworks.com/help/vision/ug/camera-calibration.html (visited on 06/12/2023).

[54] *Image Labeler*. MathWorks. URL: https://www.mathworks.com/help/vision/ref/imagelabeler-app.html (visited on 04/11/2023).

[55] A. Buslaev, V. Iglovikov, E. Khvedchenya, A. Parinov, M. Druzhinin, and A. Kalinin. “Albumentations: Fast and Flexible Image Augmentations”. *Information* 11.2 (2020). DOI: 10.3390/info11020125.

[56] D. Kingma and J. Ba. “Adam: A Method for Stochastic Optimization”. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*. 2015. DOI: 10.48550/arxiv.1412.6980.

[57] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. 2015. URL: https://www.tensorflow.org.

[58] F. Chollet et al. *Keras*. 2015. URL: https://keras.io.

[59] P. Iakubovskii. *Segmentation Models*. 2019. URL: https://github.com/qubvel/segmentation_models.

[60] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. “ImageNet Large Scale Visual Recognition Challenge”. *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252. DOI: 10.1007/s11263-015-0816-y.

[61] N. Otsu. “A Threshold Selection Method from Gray-Level Histograms”. *IEEE Transactions on Systems, Man, and Cybernetics* 9.1 (1979), pp. 62–66. DOI: 10.1109/TSMC.1979.4310076.