UPPSALA
UNIVERSITET

UPTEC STS 22025
Examensarbete 30 hp
Juni 2022

Locating power lines in satellite images with semantic segmentation

Erik Lundman

Civilingenjörsprogrammet i system i teknik och samhälle

UPPSALA
UNIVERSITET

Locating power lines in satellite images with semantic segmentation

Erik Lundman

Abstract

The inspection of power lines is an important process to maintain a stable electrical infrastructure. Simultaneously it is very time consuming task considering there are 164 000 km of power lines in Sweden alone. A cheaper and more sustainable approach is an automatic inspection with drones. But for a successful inspection with drones, exact power line coordinates is needed, which is not always available.

In order to identify power lines in satellite images a machine learning approach was implemented. In machine learning, semantic segmentation is the process of pixel-wise classification of an image. Where you not only label the entire image, but every pixel individually. This way not only the existence of a power line will be identified, but their position inside the image. This thesis aims to investigate if semantic segmentation is an effective approach to locate power lines in satellite images. And what methods can be used on the segmented output data to extract linestring coordinates representing the power line. Linear regression and a polygon centerline extraction method was implemented on the segmented output data in order to define a line that represents the true location of the power line.

The semantic segmentation model could find power lines where they were clearly visible, but struggled where they were not very visible. From good output data from the segmentation model, the linear regression and the polygon centerline extraction methods could successfully extract linestring coordinates that represented the true location of the power line. In the best case around 67\% of power lines was correctly identified. But still, with good output data from the model, complex shapes such as intersections might still get bad results. Even if the approach need further work, and can not reliably identify all power lines in the current state, it has proven that this could be a promising method to identify power lines in satellite images.

Teknisk-naturvetenskapliga fakulteten

Uppsala universitet, Utgivningsort Uppsala/Visby

Handledare: Tobias Fridén Ämnesgranskare: Anders Hast

Examinator: Elísabet Andrésdóttir

Populärvetenskaplig Sammanfattning

Stabil och kontinuerlig elförsörjning är en absolut nödvändighet i dagens samhälle. Regelbunden inspektion av elledningar är en del av arbetet för att göra detta möjligt. Bara i Sverige finns det 160 000 km luftburna elledningar, så inspektionen är både en tidskrävande och dyr process. Ett alternativ för mer effektiva, säkra och autmatiserade inspektioner är användadet av drönare. Men med automatisering dyker nya utmaningar upp. För att inspektera med drönare blir exakta koordinater på elledningar extra viktiga. Detta för att göra det möjligt att faktiskt flyga på rätt ställe längst med elledningen. Exakta koordinater innebär också en möjlighet att flyga närmare för att samla bättre data för inspektionen. Men eftersom stora delar av Sveriges elnät byggdes upp under 1900-talet finns det inte alltid tillgängligt. En metod för att skilja mellan bra och dåliga koordinater innan den första flygningen är ett utmanande problem. Men ett problem som måste lösas för att optimera processen.

Satellitbilder har både ökat i kvalité och tillgänglighet genom åren. Via Google Maps kan vem som helst få tillgång till satellitbilder där elledningar kan vara synliga för blotta ögat. Därmed uppstår frågan om satellitbilder har tillräcklig kvalité och kan bistå med tillräcklig information för att bestämma elledningarnas sanna position. Kan vi använda oss av AI och bildanalysmetoder för att automatiskt identifiera och extrahera koordinater på elledningar.

I detta projekt implementeras en maskininlärningsmetod för att identifiera elledningar i satellitbilder. "Semantic segmentation" är en process inom maskininlärning med målet att klassifiera varje individuell pixel i en bild. Till skillnad från standard klassifiering där en hela bilden får en klass, kan semantic segmentation också ge information om form och position av objekten som klassifieras. Vilket kommer vara absolut nödvändigt då vi inte vill veta **om** det finns en elledning utan **var** den finns i en satellitbild. DeepLab är en maskininlärningsmodell speciellt utvecklad för semantic segmentation. Tillsammans med satellitbilder tagna från Google Maps implementeras DeepLab-modellen för att indentifiera elledningar. Ungefär 5800 satellitbilder, som innehåller elledningar, samlades för att träna maskininlärningsmodellen att hitta elledningar.

Då modellen endast kommer att ge färgkodade bilder som utdata behövs ytterligare en metod för att konvertera detta till koordinater (en linje) som representerar elledningen. Två olika metoder undersöktes för att hitta den linjen som bäst representerade elledningen utifrån de segmenterade bilderna skapade av DeepLab-modellen.

Resultaten från DeepLab modellen visade goda resultat där elledningen var tydligt synlig. Dock där elledningar var svåra att se var resultaten relativt dåliga. I flera fall identifierades även väggrenar, spår i åkrar eller andra linjära strukturer inkorrekt som en elledning. I bästa fall var ungefär 67% av de framställda linjerna is stort sett korrekt placerade längst med elledningen. Resultaten indikerar därmed att mer än hälften av maskerna från DeepLab modellen ger tillräcklig information för att identifiera elledningens sanna position. Många förbättringar krävs för att denna metod ska minska antalet felklassifieringar och ge tillräckligt bra resultat för att identifiera ett helt elnätverk. Men med mer arbete och förbättringar kan detta tillvägagångssätt vara lovande för att få en bättre uppfattning om elledningarnas sanna position.

Acknowledgements

This master thesis has been conducted for Uppsala University in collaboration with Airpelago, a company providing efficient power line inspections with drones.

I would like to give special thanks to my supervisor Tobias Fridén and the co-workers at Airpelago for all the help and guidance during the project, while giving me the freedom and trust to follow my own ideas.

I also want to thank my subject reader Anders Hast for all the encouragement and valuable feedback during the project.

Contents

1 Introduction

The process of inspecting power lines is important to maintain a stable electrical infrastructure. But at the same time it can be a daunting task, since Sweden alone have 164 000 km of power lines above ground [1]. One option for more efficient, safe and automated inspections can be provided by drones. A cheaper and more sustainable option than inspection by helicopter. But in the process of automation new challenges arises. With automatic inspection with drones, the exact location and coordinates of the power lines becomes more important. Better coordinates can make sure the drones actually fly along the power lines and will give the opportunity to fly closer to get better data for the inspection. Currently exact coordinates is not always available, since most power lines in Sweden were built during the 1900s. If the power line is not correctly captured by the drones a proper inspection can not be made. A method to differentiate between the good and bad coordinates before the first flight is a challenging problem. But a necessary issue to resolve in order to optimize the process.

Satellite images have improved in quality during the years, and even thin objects such as power lines can be visible to the human eye. Therefore the question arise if satellite images can give us the information we need about the true location of the power lines. And if the use of machine learning and image analysis can give us the tools needed to automatically detect and extract the coordinates of power lines found in satellite images.

In machine learning, semantic segmentation is the process of assigning a label to every pixel in an image [2]. In other words, semantic segmentation is a classification method on individual pixels. In contrast to standard classification that assigns a label to the entire image, semantic segmentation will also indicate where in the image the classified object resides. When only two classes are available, a power line or not a power line, semantic segmentation becomes a binary classification problem on individual pixels. DeepLab [3] is a deep Convolutional Neural Network designed for the task of image segmentation. Together with selected satellite imagery from Google Maps the DeepLab model will be implemented in order to identify power lines.

Since output data from the segmentation model is only color labeled images. Two different methods will be tested to extract power line coordinates from the DeepLab output data. Linear regression and polygon-centerline extraction are the two methods investigated. Where a centerline is defined as the line that "flows" through the center of a polygon. Therefore a pipeline has been established in order to investigate if power line coordinates can be extracted from Google Maps satellite imagery with semantic segmentation.

1.1 Purpose

The purpose of this study is to investigate if semantic segmentation can be used to identify and extract power line coordinates from satellite images. In order to identify if the existing coordinates are correct or not. This will be accomplished by investigating following questions:

- How consistently can semantic segmentation identify power lines in satellite images?
- Is semantic output data sufficient to extract exact coordinates?

- Which is the best method to extract coordinates from segmented output data?

1.2 Deliminations

Since an existing machine learning model specifically developed for semantic segmentation is implemented in this project. Model improvements, that might increase performance on this specific data set, is not a priority. Evaluation of different machine learning models, fine tuning of model parameters or other experiments that might improve the model performance are therefore not investigated further. Focus is concentrated on data collection and the processing of output data, in order to evaluate the entire pipeline in the limited time available.

2 Background

This section will explain the theory, concepts and tools used during this project. Section 2.1 cover the usage and handling of coordinates. Sections 2.2 - 2.7 cover different machine learning concepts and performance measures. Sections 2.8 - 2.10 presents the tools used for coordinate extraction from the model output data.

2.1 Coordinate systems and zoom level

Latitude and longitude is a coordinate system used to reference different points in the world. Decimal degrees are a way of expressing latitude and longitude as decimal fractions of a degree. Like latitude and longitude decimal degrees is bound to +-90 and +-180 degrees each. As opposed to another popular alternative, degrees, minutes and seconds, decimal degrees are simply fractions of a number which make them easier to use in mathematical instances.

Google Maps have different zoom levels available. From zoom level 0 (lowest zoom with entire world map visible) to 21+ (which can depict single buildings or streets). 21+ simply depicts that some areas might have higher zoom levels available than others [4].

In order to translate a location in the world to a location on a map, latitude and longitude is first translated to world coordinates. In Google Maps world coordinates is a Mercator projection. The Mercator projection can be described as the world, projected onto a cylinder and therefore creating a flat map from a sphere. This results in a map where areas far away from the equator becomes disproportionately large [5].

World coordinates in Google Maps are defined in the coordinate space $[0, 256]$, $[0, 256]$ since a basic map tile in zoom level 0 is 256 x 256 pixels. The coordinates has the origin in the northwest corner of a standard western world map, x increasing to the east and y increasing to the south [6].

The world coordinates [7] for x and y is calculated with

$$x = 256 * (0.5 + \frac{long}{360}), \tag{1}$$

$$y = 256 * (0.5 - \frac{\log \frac{1+\sin(\frac{lat*\pi}{180})}{1-\sin(\frac{lat*\pi}{180})}}{4 * \pi}). \quad (2)$$

The world coordinates is then translated to pixel coordinates. Pixel coordinates is a reference to a specific pixel at a specific zoom level on a map.

In order to calculate pixel coordinates the equation used is

$$pixelCoordinate = worldCoordinate * 2^{zoomLevel}. \quad (3)$$

Therefore every pixel of a map, in any zoom level, can be translated to decimal degrees and vice versa [6].

2.2 Convolutional Neural Networks

Deep Learning is a specific kind of machine learning which refers to the structure of a network with multiple layers [8]. Artificial Neural networks(ANN) and other Deep-learning architectures have shown impressive results when solving complex problems and discovering complicated patterns in various kinds of data. Convolutional Neural Networks(CNN) is a special kind of ANN that uses convolution in at least one of the layers and have been very successful when processing for example image data [8].

The success of CNN:s in image analysis comes from the convolutional layers use of the grid-like structure of image data. The convolutional layer will better preserve information concerning pixels and the relation to their neighbours [9]. Traditional layers i neural networks uses matrix multiplication where every input unit interacts with every output unit. Which becomes very computational heavy with large input data. CNN:s on the other hand make use of what is called sparse interactions. Sparse interactions is accomplished by making the kernel smaller than the input data and therefore each output will only depend on a portion of the input [8]. Figure 1 demonstrates sparse connectivity between input and output.

To better represent how the convolution and sparse connectivity operates on grid-like data, such as images, Figure 2 demonstrates the convolution operation but in two dimensions with a kernel size 3 x 3 [9].

2.3 Semantic segmentation

Semantic segmentation is a method in computer vision that aims to classify each pixel in an image to a certain class. Which is useful when you want to classify objects in an image but also want to know their shape and position [2]. The data set used for semantic segmentation consist of the original images and the ground truth. An example of semantic segmentation with two labels is shown in Figure 3. The ground truth is a mask labeled with different colors representing the objects to be classified. Each image have their corresponding mask. In a perfect world expected output from the model should represent the ground truth.

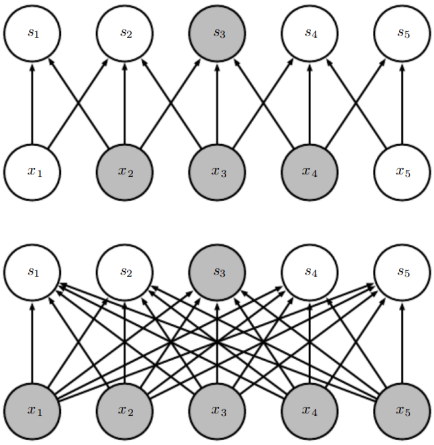


Figure 1: Sparse connectivity. One output unit, s_3 is highlighted, together with all the input units x that affect s_3. Top image represents the convolutional operation with a kernel size 3, then only three inputs directly affect s_3. Bottom image demonstrates traditional matrix multiplication where all inputs affect s_3.

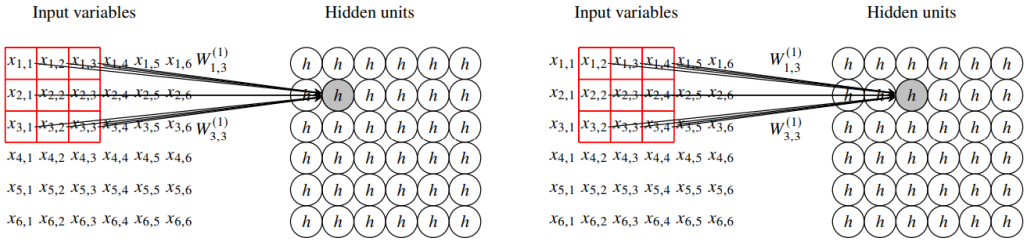


Figure 2: Demonstrated in the left image, input variables (pixels in image) is mapped to a layer of hidden units (output) using a 3 x 3 kernel, here the red square. Therefore each hidden unit is only dependant on a small portion of the input data. Demonstrated in the right image, the kernel moves one step to the right and it will then map to the hidden unit to the right.

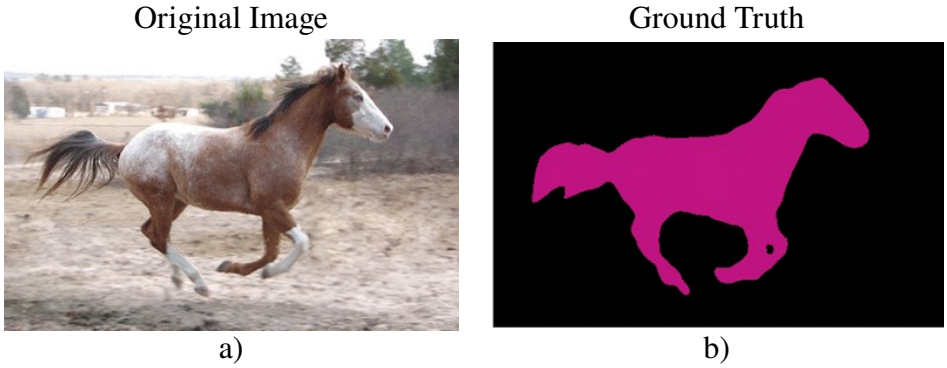


Figure 3: Example of semantic segmentation with two labels. Black for background and purple for horse. a) The original image. b) The corresponding ground truth.

2.4 DeepLabv3+

DeepLabv3+ by Chen et al. is a deep CNN specifically designed to perform segmentation tasks [3]. It is built on the previous versions of DeepLab by Chen et al, and have all shown

promising result [10, 11, 12]. The DeepLabv3+ model has proposed several techniques to improve the task of segmentation and reduce computational cost. Two of the concepts used are called "atrous convolution" and "atrous spatial pyramidal pooling".

Atrous convolution is a modification on normal convolution where the size of the kernel is increased by inserting gaps, as illustrated in Figure 4. This effectively broadens the field of view to get a larger context without increasing the amount of parameters. By increasing what is called dilation rate, the size of the atrous convolution can be set.

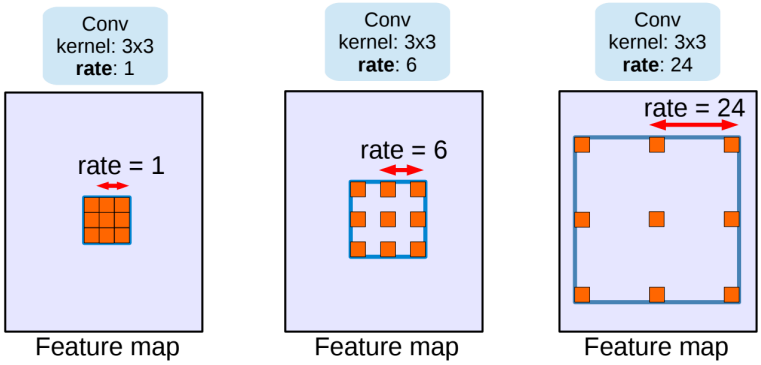


Figure 4: Atrous convolution with kernel size 3 x 3 and different rates. Dilation rate = 1 represents a normal convolution. With higher rate the field of view gets bigger.

Atrous spatial pyramid pooling is basically the concept of merging information from different scales. Atrous convolution of different rates is used together to get multi-scale information, visualised in Figure 5. Which can help to classify objects of different sizes.

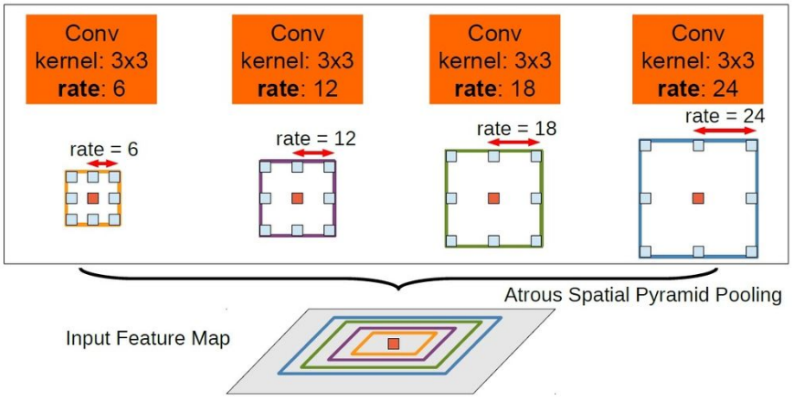


Figure 5: Atrous spatial pyramid pooling. When classifying the orange pixel in the middle, information from multiple parallel filters of different rates are used together.

The loss function used in this implementation of the DeepLab-model was the categorical cross-entropy loss function. It is defined as

$$-\sum y_i * \log \hat{y}_i, \tag{4}$$

where $\hat{y}_i$ is the predicted value of y_i.

More extensive and detailed information about the DeepLab model and its features are available for further read [3, 10, 11, 12].

2.5 Overfitting

For a successful implementation of a machine learning model, like a CNN, the model must be able to handle new unseen data. Not only the data that was used for training. Overfitting occurs when the model get very low errors during training but high error during testing [8]. The model recognizes features tied to the training data and struggles to generalize features in order to perform well on new data. Generally if a model is to complex or the data set is to small, the model can tend to overfit on that data.

2.6 Data augmentation

Generally the more samples in a data set, the better a model will perform. More images will make it easier for the model to generalize features and reduces the chance of overfitting. The task of generating thousands of unique images with corresponding labels is very time consuming and sometimes several thousand images might not even be available. Data augmentation is a method to generate more samples from existing data. Such as rotating, flipping, color alterations etc. This is a way to artificially increase the size of the data set. The new images will still be correlated to the original images, and more unique original images is still superior. But it is a method proved to improve performance and decrease the chance of overfitting [13].

2.7 Performance metrics

In a binary classification problem with either a positive label (a power line) or a negative label (not a power line). The results of the model can be represented in a 2 x 2 confusion matrix, as seen in Figure 6. The confusion matrix has four different values. True positives (TP) are all the positive labels correctly classified as positive. False positives (FP) are the negative labels incorrectly classified as positive. True negatives (TN) are all negative labels correctly classified as negative and false negatives (FN) are positive labels incorrectly classified as negative [14].

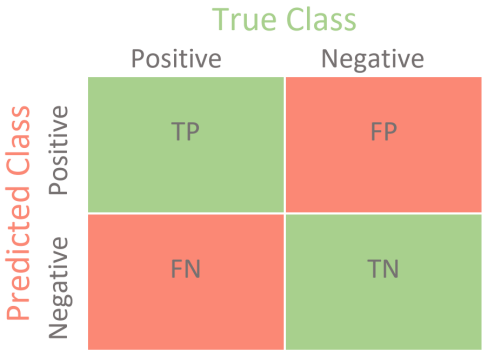


Figure 6: Confusion matrix from binary classification problem.

From the confusion matrix some performance metrics can be calculated. The accuracy

$$A = \frac{TP + TN}{TP + TN + FP + FN}, \tag{5}$$

is the fraction of both positive and negative labels correctly classified. Further we have precision

$$P = \frac{TP}{TP + FP}, \tag{6}$$

and recall

$$R = \frac{TP}{TP + FN}, \tag{7}$$

where precision is the fraction of positive labels that are classified correctly. And recall is the fraction of positive labels that is found. In binary classification recall is also referred to as sensitivity.

An inbalanced data set means that the distribution of positive and negative labels are not evenly distributed. And in those cases appropriate performance measurements will be important. For example, in an extreme case, where the positive labels only accounts for 1% of the total data points. A bad prediction with only negative labels will get 99% accuracy. While precision and recall will give 0%. Therefore accuracy might be a misleading measurement on inbalanced data sets.

2.8 Linear regression

Linear regression is a machine learning algorithm that aims to best fit all input data to a linear function [8]. An example of linear regression is demonstrated in Figure 7. If $\hat{\mathbf{y}}$ is the predicted value of $\mathbf{y}$ and $\mathbf{x}$ is a vector of input data. Together with a vector of weights $\mathbf{w}$ the linear output function can be defined as

$$\hat{y} = w^T x + b, \tag{8}$$

where $\mathbf{b}$ is a bias term that allows the predicted lite to not pass through the origin.

In order to predict outputs $\mathbf{y}$ from inputs $\mathbf{x}$ the weights are improved. The weights are improved by minimizing the mean squared error (MSE) given by

$$MSE = \frac{1}{m} \sum_i (\hat{y} - y)_i^2, \tag{9}$$

where $\mathbf{m}$ is the number of inputs. MSE aims to minimize the total distance from all data points to the line. The weights are improved in a way that reduces the MSE in order to find the optimal line.

2.9 Centerline

The centerline of a polygon can be described as the line that flows through the center of a polygon, as illustrated in Figure 8. The process of extracting the centerline can be rather complex, but one method used, extracts the centerline with the help of Voronoi diagrams [15], [16]. An example of a Voronoi diagram can be seen in Figure 9c. The Voronoi Diagram is the polygon structure created when radially expanding circles from several points as demonstrated in Figure 9a-c. When the points chosen are on the edges of a polygon the intersections from the Voronoi diagram inside the polygon will somewhat represent the centerline.

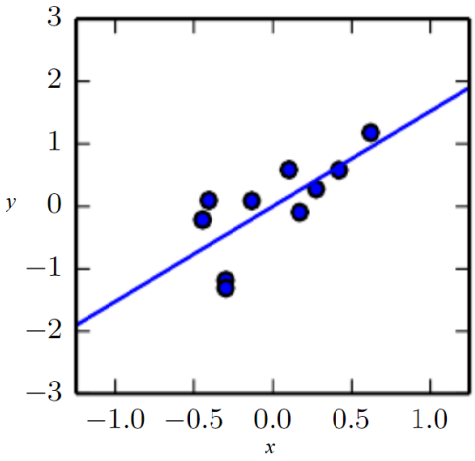


Figure 7: Linear regression with ten data points. The line is drawn to minimize the total distance from all points to the line.

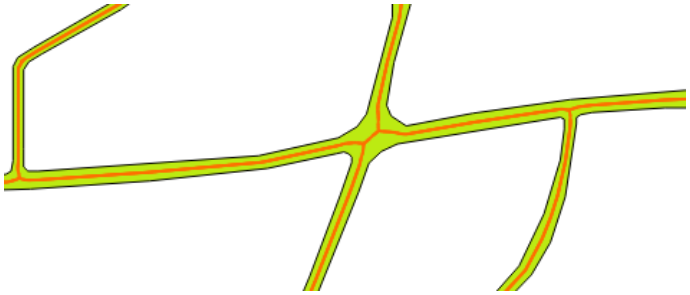


Figure 8: Red line represents the centerline of the polygon.

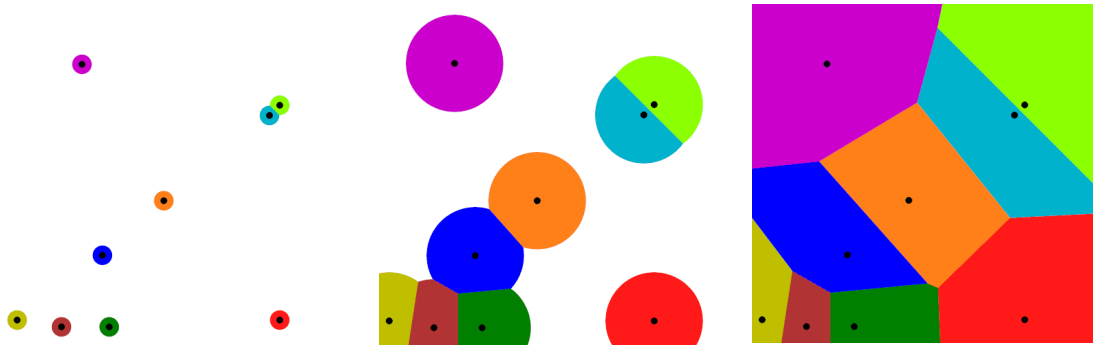


Figure 9: Creation of Voronoi diagram. a) - b) Radial expansion of circles from selected points. c) A Voronoi diagram

2.10 Visvalingam-Whyatt algorithm

In order to reduce the complexity of a line but keep the general shape M. Visvalingam and J. D. Whyatt proposed an algorithm in an article "Line generalisation by repeated elimination of points" [17]. Where the effective area of points of a line is calculated, which is the area of the triangle of its neighbouring points, as shown in Figure 10. The point with the smallest effective area is removed and the process is repeated until a certain threshold is reached.

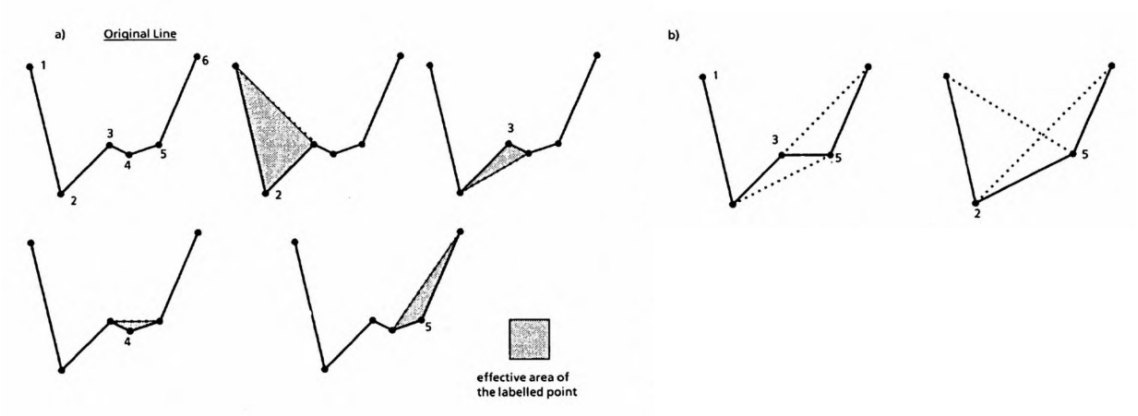


Figure 10: The Visvalingam-Whyatt algorithm. a) The effective areas of different points on a line, calculated by the triangle of their neighbouring points. b) The repeated removal of points with the smallest effective area.

3 Method

To understand how to approach this problem the process can be divided into three parts. First and foremost the process of collecting and preparing images and data for the machine learning model. Secondly the structure and handling of the model itself. And finally the post process of the data i.e. how the output of the machine learning model can be used to predict and estimate the coordinates of power lines. Python with the library Tensorflow have been the platform used for implementing the DeepLab model [18]. And Python together with the library OpenCV have been the base framework when handling images [19].

3.1 Data selection

To build a data set large enough to properly train a neural network the most important aspect was to generate images as easy and effective as possible. In order to get easy and free access to a large amount of satellite images, Google Maps was chosen as the source of all the images. Maps Static API is an API in Google maps where satellite images can be downloaded with an HTTP request. Images with desired center coordinates, resolution, zoom level (0-21+) etc. This approach enabled an easily generated data set of images. The quality of the images was high enough to detect power lines with the naked eye and was deemed sufficient for the task. Image quality on high zoom levels can differ depending on location. Zoom level was decided simply by comparing images from different zoom levels and see where power lines where most visible. Zoom level 18 was chosen since power lines was often visible and image quality was the best.

With given coordinates of power line structures, images was downloaded evenly distributed along the power line. The coordinates used to create the data set was very precise line string coordinates of existing power line networks. Precise coordinates was preferable in order to get more control in the data selection process. The best resolution available, with Maps Static API, was images with 1280 x 1280 pixels. Therefore images was downloaded along the power line divided in such a way that the best resolution would give a small overlap to cover every part of the power line, demonstrated in Figure 11.

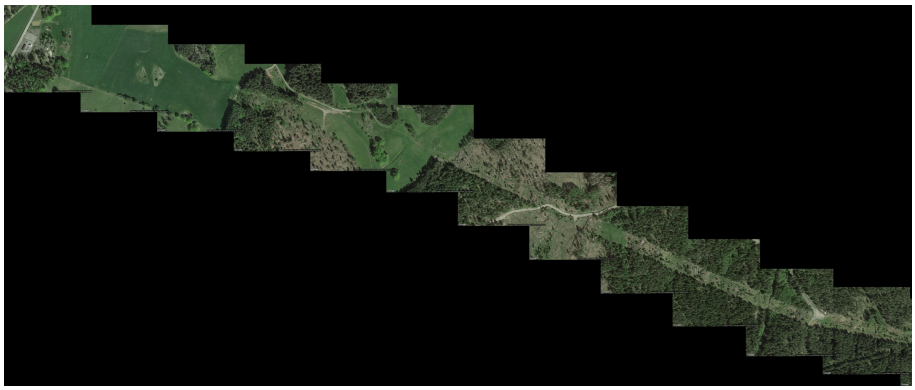


Figure 11: How images was collected with a small overlap to cover all parts of the power line. Every rectangle is a image with resolution 1280 x 1280.

A control image was made in order to inspect that the power line was evenly divided and properly covered. Figure 12 shows part of a control image, a zoomed out satellite image over the power line network. Red line indicates the power line and blue markers indicate the center coordinates of each image. The image was used to inspect that the power line was evenly divided and properly covered before downloading images.

Figure 12: Visualisation of the center coordinates of each downloaded image distributed along the power line. Red line marks the power line and blue dots marks the center coordinates of each image.

3.2 Data set structure

To all satellite images a corresponding binary mask was created, the ground truth. The ground truth was created as black and white masks. Basically semantic segmentation with only two different classes. Black for background and white for power lines.

In order to create masks for all the images, an effective and automated approach was important. When creating the ground truth data set we could again take advantage of the precise power line coordinates. Each image was named after the coordinates of the center

pixel. The center coordinates together with the width and zoom of the image gave full information to determine the coordinates for all pixels in the image. Every image and the power line coordinates was converted to pixel coordinates. Each image was defined as a square in the pixel coordinate plane, every power line that intersected the image square could then be automatically drawn.

For each image, one mask and one control image was made. The mask was a black image (same size as original image) with a white line drawn along the power line coordinates. The control image was a copy of the original image but with a red line drawn, identical to the mask, along the power line. This was used as a control to see if the coordinates was good enough. The width of the line drawn was set experimentally to where it properly covered the power line. In order to asses the quality of the masks and remove the bad ones, all the control images was manually inspected to see if the power line was successfully covered. An example of the data set structure can be seen in Figure 13.

a) b) c)

Figure 13: Structure of data set. a) Original image downloaded. b) Control image that marks the power line from given coordinates. c) The corresponding mask that is the ground truth.

Since the coordinates was converted from decimal degrees to world coordinates using the Mercator projection, in equation 1 and 2. And then world coordinates to pixel coordinates according to equation 3. There were some some rounding errors and small inaccuracies present. Additionally Google Maps different map sections are not always merged correctly, which could lead to disappearing and misaligned power lines, as demonstrated in Figure 14. Therefore even with exact coordinates, some power lines was not covered properly with this method. But with the approach explained above approximately 67% percent of the masks were produced automatically. The rest of the images where annotated manually to further increase the size of the data set.

3.3 Preprocessing

The first step was to crop all the images and masks from 1280 x 1280 to 1200 x 1200 pixels to remove unwanted text along the edges. Then each original image, and corresponding mask, was divided equally into several smaller images following the power line, as illustrated in Figure 15. Since the power line coordinates in the image was known this could be done automatically. This was done for several reasons. With smaller images some of the irrelevant background could be removed to focus more on the region of interest. Therefore remove areas where no power lines was present. Simultaneously large images will in

Figure 14: Example of an inaccurate representation of a power line in Google Maps.

general require more computer memory and be more time consuming when training the model. And at the same time it would help to increase the number of images in the data set. The smaller images produced had the resolution 400 x 400 pixels. It is hard to know what image size would give best results. According to [20] increasing image size from 128 x 128 to 256 x 256 gave better performance for a segmentation task. But the exact same model was not used and there might be an upper limit as well. At the same time it would be unnecessary to classify areas we know are just background, since it will only increase model training time. Trial and error is the only way to really know, but the time available is limited.

a) b)

Figure 15: Visualisation of how a satellite image and corresponding mask was divided into smaller images. a) Original image. b) Corresponding mask.

Directly downloading images with 400 x 400 resolution with Google Maps API would give approximately the same data set but without the extra dividing step as described in Figure 15. There are majorly two reasons why this approach was chosen. The number of HTTP request done with Maps Static API decreases by a factor of three in this case. And only a certain amount of HTTP requests are available for free. Secondly all the manual inspecting and manual labeling could be done on the larger images before dividing them into smaller ones. Therefore all the manual work could be divided by three. Additionally this approach gives the flexibility to easily change the image size of the data set if needed. In this project 136 masks was done manually. Without this approach approximately 408 masks had to be done manually.

In total 427 satellite images was downloaded with Maps Static API from different areas. 20 was removed because of bad quality. 271 images had automatically created masks. After dividing into smaller images the total data set was 1674 images with resolution 400 x 400. Images was divided approximately into 88% training and 12% validation. Test data was also collected separately from a completely different area.

3.4 Augmentation

In order to increase the size of the data set, some augmentation was done on the training data. The augmentations used was a random brightness variation, horizontal flip and random 90 degree rotation. With all three augmentations, 4413 additional images was added to a total of 5884 images in the training data set.

3.5 DeepLabv3+

A reduced version of DeepLabv3+ was implemented on the created data set. The model was reduced in order to not exceed memory capacity on the laptop used. The computer memory was an issue when processing thousands of images with a complex model. With reduced dimensionality and reduced batch size the Deeplabv3+ model could properly run on a laptop with this data set. The final model trained with 50 epochs and took approximately 72 hours. Accuracy and loss for training and validation was plotted during the training of the model.

The imbalance of the data is relatively large. Which means the distribution of white and black pixels are not even, since the majority of the data is actually background (black pixels). Therefore it is important with appropriate performance measurements. True positivies, true negatives, false positives and false negatives was extracted from the output data. Then precision and recall could be calculated to get a performance measurement better representing an imbalanced data set.

3.6 Coordinate extraction

In the post processing the output data of the deep learning model was analyzed in order to extract linestring coordinates. The expected output data from the model is black and white masks, where the white pixels are identified as power lines. In order to process this information it is convenient to refer the white pixels to clusters or polygons. The goal in this process is to find one or multiple lines that represent the shape of the cluster/polygon to extract linestring coordinates. In order to convert output images back into coordinates some alternatives was investigated.

Some pre-processing of the output data was done. Small clusters was removed in order to reduce noise and get cleaner images. Since the output could significantly vary in shape and appearance multiple methods might be needed. Since power lines in general represent lines, linear regression was applied to find a general direction of white clusters in the data. Since the linear regression gives us a line, this would be an easy and attractive method if the results was good.

On the other hand if there was an intersection or curve present another approach was needed. If the contours of the white clusters was identified, they can be seen as polygons.

With polygons we could extract the centerline and find the general direction of the polygon regardless of the shape. But the centerline extracted with the use of Voronoi diagrams will be very complex with possibly hundreds of coordinates defining the line. Therefore a line simplification algorithm, the Visvalingam-Whyatt algorithm, was introduced to simplify the centerline while the general shape was preserved. With the goal of finding the general "flow" of the polygon with the most simple line possible. The regression and centerline method was both tested on the validation data set. For extracting and simplifying the centerline the libraries Centerline [21] and Simplification [22] was used.

More measurements was done in order to determine if the predicted centerline, and regression line, properly covered the actual power line. After a line was predicted it was analyzed to see if the centerline was contained inside the white cluster of the ground truth. All predicted lines produced was compared to the ground truth data in the validation data set. The total length of the line was measured, together with the length of the line contained inside the white mask. This way we could calculate what percentage of the line that actually was on the power line.

4 Results

Section 4.1 will present samples of segmented output images from the DeepLab model. Section 4.2 covers model performance and section 4.3 show results from the two proposed coordinate extraction methods.

4.1 Model output data

Following images shows input images to the left, output images to the right and an overlap image in the middle.

Figure 16 shows some good examples of output data from the model from the test data set. And Figure 17 is representing some good results from the validation data set. The images from the test data set in Figure 16 was taken from a completely different location than the training and validation data to ensure minimal bias. The white pixels are classified as power line, and black pixels as background.

Figure 16: A subset of good results from the test data set. Images in column a is input images to the model. Images in column c is the output images. Column b is an overlap between input and output images.

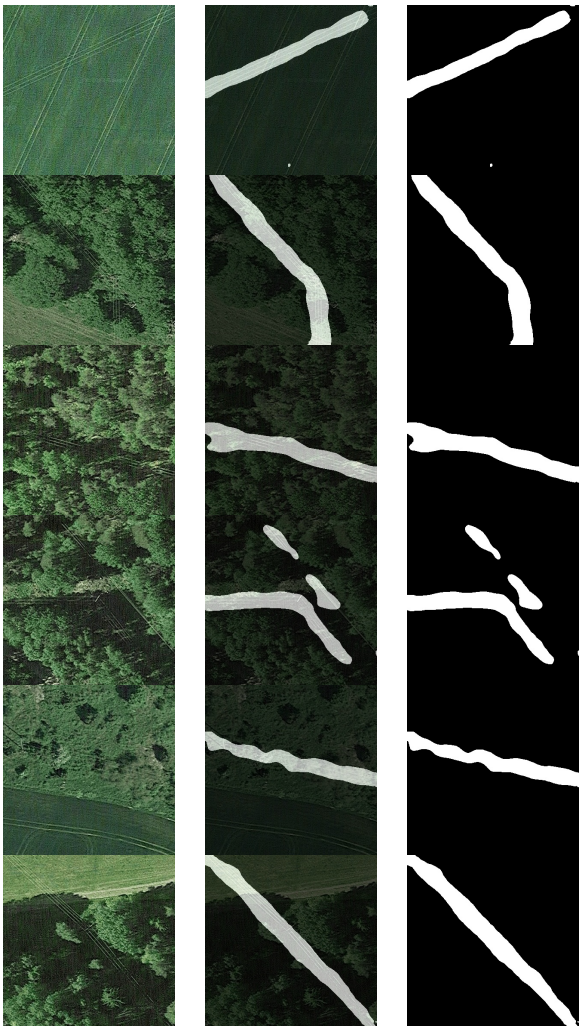


Figure 17: A subset of good results from the validation data set. Images in column a is input images to the model. Images in column c is the output images. Column b is an overlap between input and output images.

Figure 18 shows some bad results from the model, from both test and validation data set.

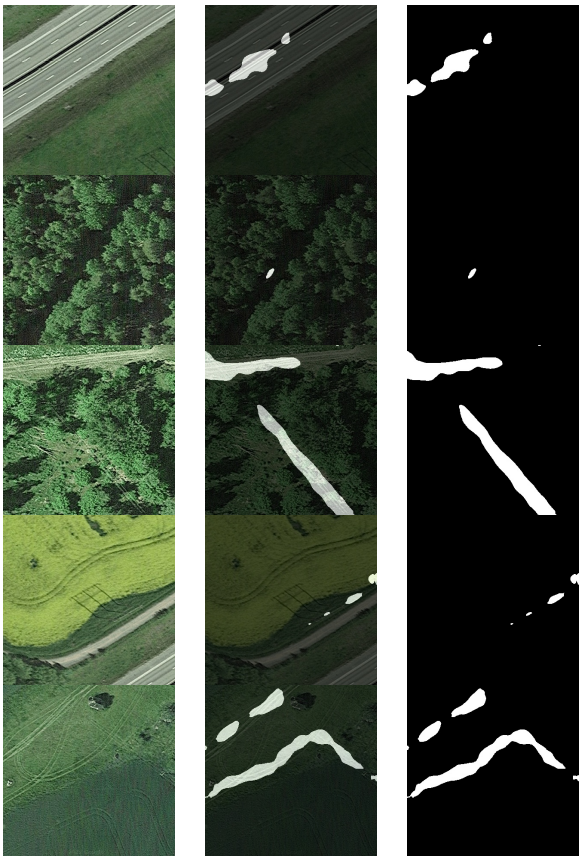


Figure 18: A subset of bad results from both validation and test data set. Images in column a is input images to the model. Images in column c is the output images. Column b is an overlap between input and output images.

4.2 Model performance

Figure 19 shows the training accuracy of the simplified DeepLab model developing over 50 epochs. Figure 20 shows the validation accuracy of the same model over 50 epochs.

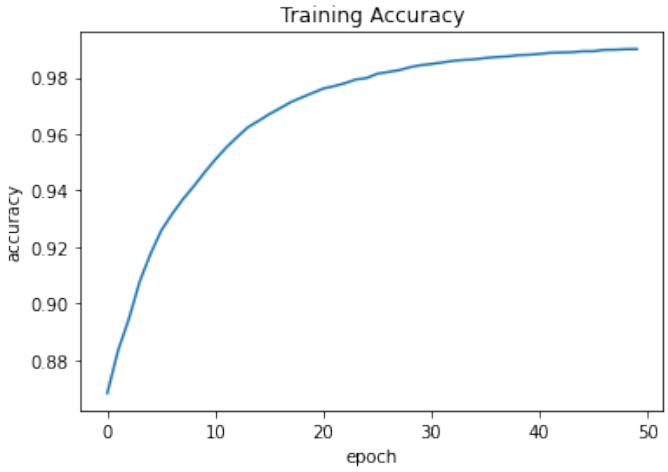


Figure 19: Training accuracy of the DeepLab model during 50 epochs.

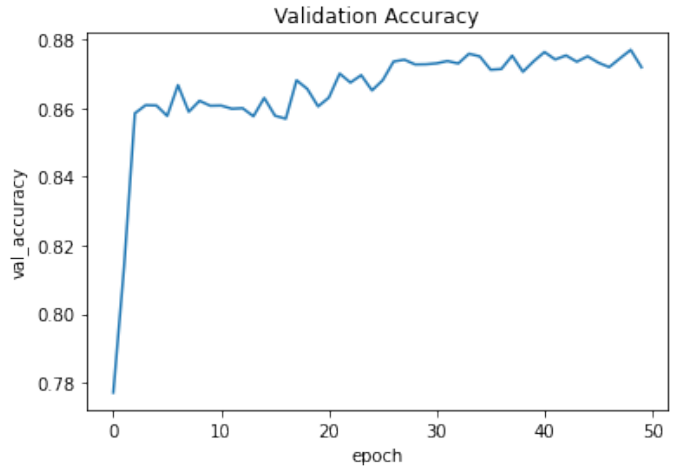


Figure 20: Validation accuracy of the DeepLab model during 50 epochs.

Figure 21 shows the training loss from the model over 50 epochs of training. Figure 22 shows the validation loss during training of the same model. The loss was calculated using the categorical cross entropy loss function, from equation 4.

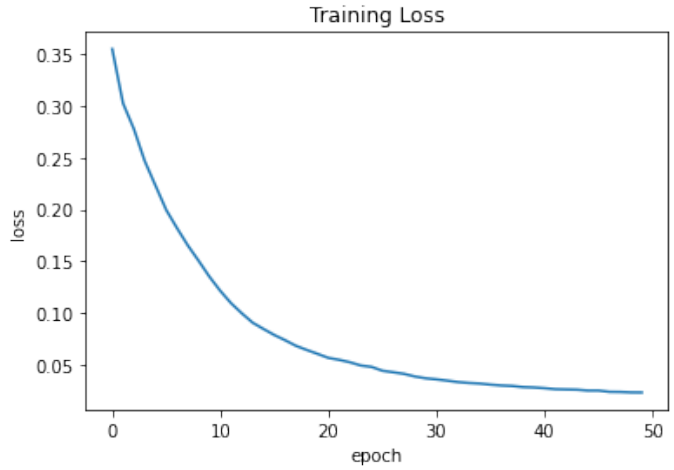


Figure 21: Training loss of the DeepLab model during 50 epochs.

Figure 22: Validation loss of the DeepLab model during 50 epochs.

Table 1 represents a confusion matrix on the validation data. Each value is divided by the total amount of data points in their corresponding true class. Table 2 shows the calculated precision and recall on the validation data set.

		Predicted class	
		Positive	Negative
True class	Positive	0.31	0.69
	Negative	0.03	0.97

Table 1: Confusion matrix on validation data. 31% of white pixels where correctly classified as white and 69% was classified as black. 97% of black pixels where classified correctly as black and 3% was classified as white.

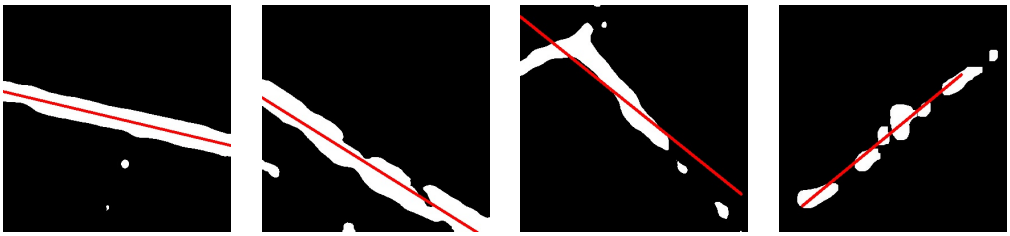
Precision	Recall
67%	31%

Table 2: Precision and recall on validation data set. The recall also corresponds to top left value of the confusion matrix.

4.3 Coordinate extraction

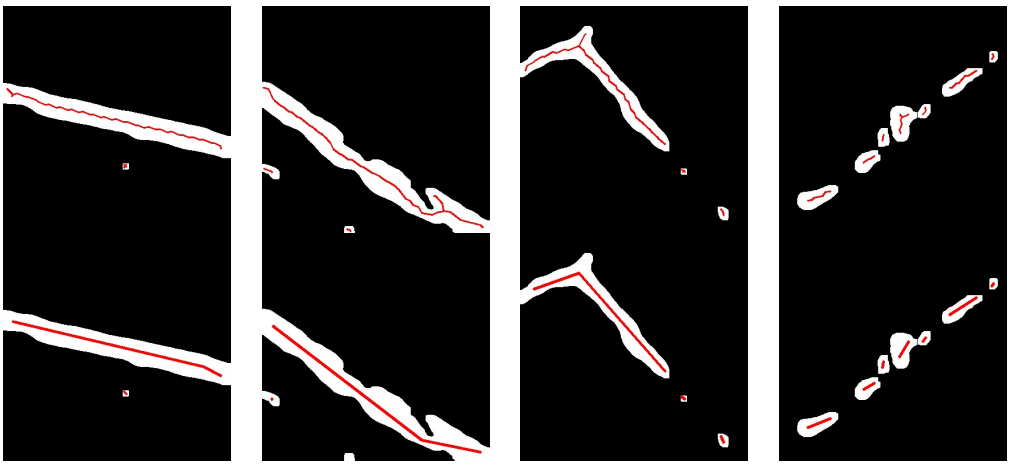
Figure 23 shows some of the results using linear regression to extract linestring coordinates from the output data. And Figure 24 is the same images but the results using centerline together with the line simplification algorithm. White clusters is the output from the validation data. And the red line represents the predicted line.

Results of power line coverage of both coordinate extraction methods are represented in Figure 25. Power line coverage is how many percent of the predicted line that is on top of the actual power line. Each column represents a 10% interval, starting from 1-10% and



a) b) c) d)

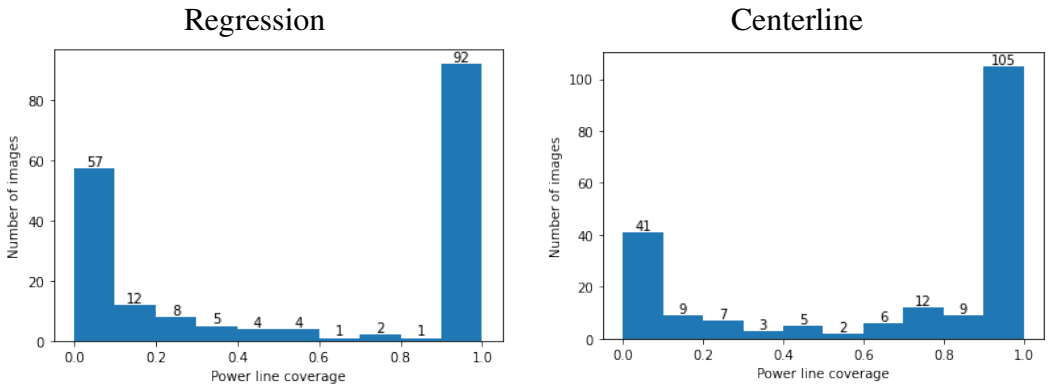
Figure 23: Examples of linear regression on output images to identify coordinates of power line.



a) b) c) d)

Figure 24: Results from the centerline and line simplification method on output images. Top row shows the calculated centerline. Bottow row shows the simplified centerline to capture the general shape of the centerline.

ending at 90-100%. The number of predicted lines in each interval is shown at the top of each column. The mean coverage using linear regression was 0.56. The mean coverage using centerline was 0.67.



a) b)

Figure 25: The number of predicted lines with different power line coverage. a) Line coverage of each image using linear regression. b) Line coverage of each image using centerline.

5 Discussion

5.1 Model output data

When there are a power line clearly visible the model tend to classify that pretty accurately. On the other hand, there is generally a clear trend, what images get bad results. When the power line is either hidden (by trees or shadows) or not visible in contrast to the ground, the model performs very bad. Even if it is distinguishable by the shadows or the power line poles. Other linear structures such as road boarders might also be falsely classified as a power line. Therefore clear linear features seems to be important for the model when identifying power lines. This should maybe not be a surprise, the non visible power lines is obviously harder to identify. But at the same time there are a lot of them covered by trees. Which clearly will be a problem.

There might be a problem with low representation of those images in the data set. And an increased data set with more non visible power lines could likely improve those results. Or another approach could be to sample images with bad results too increase the training on those images. Bad image quality might also be a reason. Since image quality in Google Maps may differ depending on location and might directly affect the visibility of power lines. If it is possible to find information about image quality, that information could be used to weight predictions from high quality images higher. Or simply not use bad quality pictures at all if possible, but that might result in not enough data instead.

The appearance of power lines from above can differ a lot. There might even be conflicting characteristics that make it hard for the model to put them all under the same label. One suggested solution would be to add one more label to the data set. That is the non-visible power lines. So the model could label visible and non-visible power lines differently. Then it might be easier for the model to find features specifically for non-visible power lines and classify them correctly. Getting satellite images from a different season to increase the visibility in the forest and in contrasts to the ground might also be worth looking into.

5.2 Model performance

The training and validation accuracy was both relatively high when training the model. Important to understand what accuracy in this instant actually means. Since accuracy measure the classification accuracy for both black and white labels the results can be misleading. Since the data is imbalanced and the majority of labeled pixels are the background pixels, the accuracy favors the prediction for black labels. That is probably the reason for the high validation accuracy, already in the first epoch, in Figure 20. Since the prediction of all black pixels will still yield a relatively high accuracy. The precision of the model, calculated to 67%, seems to be a much better representation of the models actual performance. Since the precision measures how many of the pixels labeled white that was labeled correct. And the white labels actually carries the information that is needed.

While training and validation accuracy is relatively high and increases during the training process, seen in Figure 19 and 20. The validation loss function from Figure 22 is also increasing during the training process, while preferably it should decrease similar to the training loss in Figure 21. This can indicate some issues with the model. It can be a

sign of overfitting to the training data. Where the model have a hard time generalizing features and only performs well on the training data. And from the output images this is at least partly true. As discussed above, the model struggles to generalize the features of non-visible power lines. But again the measure of accuracy can be misleading and might exaggerate the indication of overfitting. The DeeplabV3+ model was used with 21 different classes, so even if the model was slightly simplified for this project, it might still be to complex for its original purpose. More experiments on model parameters and especially an addition of data samples should be implemented to get further insight in this issue.

The confusion matrix give interesting insight of the model. We can see that only 31% of white pixels was classified correctly. And only 3% of black pixels where falsely labeled white. This can be seen as that the model is very careful, it conservatively classifies power lines, but when it does the majority is correct. But importantly to remember that this is a measure over all images, so one single image can still have all classified white labels wrong. But worth noting that all white pixels does not need to be found to get a very good prediction. For example if only 1% if white pixels are found in a specific image. If that 1% represents a one pixel wide line through the middle of the power line. We still have a very good prediction of the power line. But in general the confusion matrix shows that the model tends to preferably label pixels as background.

5.3 Coordinate extraction

With a straight line cluster present in the result, linear regression is an easy and effective solution to extract linestring coordinates from the output data. But as shown in the result, it is not consistent in all situations. The centerline method can handle more complex shapes and define the general shape of individual clusters, but not the relation between them. Centerline might therefore ignore useful information available, but handle complex shapes. While linear regression can determine the relations between all available clusters but generalize to much.

With the measure of "line coverage" we can get further insight on both the quality of the segmented output data, and the coordinate extraction method. In both methods we can see, from Figure 25, that most predictions are either completely on top of the power line, or completely outside. Output data containing only black pixels, or too few white labels to find a line at all, is included as 0% coverage. And the centerline method seem to generally better cover the power line, but not much of a difference. The results here are closely related to the output images from the model. With very good results from the model you get 100% line coverage, and with very bad results you get 0%.

The coordinate extraction process heavily depends on the quality of output data from the model. Noisy and inaccurate data will heavily punish the extraction method. So for now the model seems to be the biggest bottle neck for the process. To increase the performance of the pipeline more focus is needed on the segmentation process. At the same time, complex shapes such as intersections and T-sections might get weird results using this extraction method, even with good input data.

Worth noting is that there might be a small bias towards 100% coverage in the line coverage measure from the extraction methods. The regression and centerline methods only predicts

a line that is proportional to the size of the clusters or polygons. With very small clusters a very short line can be produced that does not give good information of the direction of the power line, but the coverage might still be 100%. It is an edge case worth noting but should not affect the results in a meaningful way.

A combination of centerline and regression might be an idea to look further into. But for it to work somewhat consistently there need to be a way to determine when to use what method. Something that is not too trivial. There might also be an idea to evaluate and handle different sizes of clusters differently. Another idea is to use the available power line coordinates, that was used to retrieve the images. Even though they might not be exact coordinates, they may at least help to narrow down the search area or give an idea of the direction of the power line. This will help to remove false positives (wrong white pixels) and get a higher prediction accuracy.

6 Conclusions

The use of semantic segmentation to identify power lines in satellite images seems to be a promising approach to identify power lines. The model can, with visible power lines, give a good estimation of their coordinates. Some work need to be done to further develop the model based on DeeplabV3+.The data set used is much smaller than the general standard. Increasing the data set should result in better performance and help the model to better generalize the characteristics of a power line. And also help to increase the accuracy on images where power lines are not clearly visible. Even if current method is not proficient enough to accurately identify an entire power line network. It shows that parts of a power line can be accurately identified.

The two methods proposed to extract linestring coordinates can get good results given good output data from the model. And even if the recall is relatively low, meaning not the entire power line is found, the prediction can still be very good. The centerline method shows slightly better result but might not use all available information in the best way. A combination of both methods could be a reliable method looking further into. More work in this area is needed in order to handle complex shapes such as intersections and multiple power lines in the same image. But the results show that this workflow is possible. Given a satellite image, with power lines present, you can find and extract coordinates to identify the true location of the power line.

7 Future work

In future work it would be very interesting to further investigate how to combine results from multiple images to properly identify an entire power line network. And see if missing data can be properly guessed with the use of neighbouring images. It would be very interesting to find out what percentage of images that need a good result to get a good estimation of the entire power line structure investigated.

The assistance of known power line coordinates, in the coordinate extraction methods, could be used to further improve results even though the existing coordinates are not exact. This could help in the process of removing noise and false positives to get a better

prediction. For example predicted lines that are in the completely wrong direction can be ruled out.

The expansion to a larger data set is one of the most important aspects in order to further improve on this method. Both more unique images and more augmentations could be added. With a better computer, the non simplified version of DeepLab could run properly. The full capacity of DeepLab together with an expanded data set would be the first obvious step to seek further improvement.

References

[1] Energiföretagen, "Elnätets längd," 2022. https://www.energiforetagen.se/energifakta/elsystemet/elnatet--distribution-av-el/elnatets-langd/ (2022-05-12).

[2] L. Shapiro and G. Stockman, *Computer Vision*. Prentice Hall, 2001.

[3] L. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," *CoRR*, vol. abs/1802.02611, 2018. http://arxiv.org/abs/1802.02611 (2022-03-11).

[4] Google, "Maps static api," 2022. https://developers.google.com/maps/documentation/maps-static/start (2022-06-05).

[5] Brittanica, "Mercator projection," 2018. https://www.britannica.com/science/Mercator-projection (2022-05-29).

[6] Google, "Map and tile coordinates," 2022. https://developers.google.com/maps/documentation/javascript/coordinates (2022-06-05).

[7] Google, "Showing pixel and tile coordinates," 2022. https://developers.google.com/maps/documentation/javascript/examples/map-coordinates (2022-06-05).

[8] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org (2022-05-07).

[9] A. Lindholm, N. Wahlström, F. Lindsten, and T. B. Schön, *Machine Learning - A first course for engineers and scientists*. Cambride University Press, 2021. http://smlbook.org/book/sml-book-draft-latest.pdf (2022-05-22).

[10] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Semantic image segmentation with deep convolutional nets and fully connected crfs," 2014. https://doi.org/10.48550/arxiv.1412.7062 (2022-03-11).

[11] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *CoRR*, vol. abs/1606.00915, 2016. http://arxiv.org/abs/1606.00915 (2022-03-11).

[12] L. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *CoRR*, vol. abs/1706.05587, 2017. http://arxiv.org/abs/1706.05587 (2022-03-11).

[13] F. Chollet, *Deep Learning with Python*. Manning Publications Co., 2018.

[14] L. A. Jeni, J. F. Cohn, and F. De La Torre, "Facing imbalanced data–recommendations for the use of performance metrics," in *2013 Humaine Association Conference on Affective Computing and Intelligent Interaction*, pp. 245–251, 2013. 10.1109/ACII.2013.47 (2022-05-01).

[15] X. Bai, Z. Zhu, P. Zou, J. Chen, J. Yu, and Y.-W. Chang, "Voronoi diagram based heterogeneous circuit layout centerline extraction for mask verification," in *2022 27th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 172–177, 2022. 10.1109/ASP-DAC52403.2022.9712516 (2022-05-16).

[16] C. Chen, X. Mei, D. Hou, Z. Fan, and W. Huang, "A voronoi-diagram-based method for centerline extraction in 3d industrial line-laser reconstruction using a graph-centrality-based pruning algorithm," *Optik*, vol. 261, p. 169179, 2022. https://doi.org/10.1016/j.ijleo.2022.169179 (2022-05-16).

[17] M. Visvalingam and J. D. Whyatt, "Line generalisation by repeated elimination of points," *The cartographic journal*, vol. 30, pp. 46–51, 1993. https://hull-repository.worktribe.com/output/376330 (2022-05-10).

[18] M. A. et al., "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. tensorflow.org (2022-03-27).

[19] G. Bradski, "The OpenCV Library," 2000. https://opencv.org/ (2022-02-01).

[20] O. Rukundo, "Effects of image size on deep learning," *CoRR*, vol. abs/2101.11508, 2021. https://arxiv.org/abs/2101.11508 (2022-05-15).

[21] F. Todić, "Centerline," 2014. https://centerline.readthedocs.io/en/latest/index.html (2022-04-11).

[22] S. Hügel, "Simplification," 2021. https://github.com/urschrei/simplification (2022-04-20).