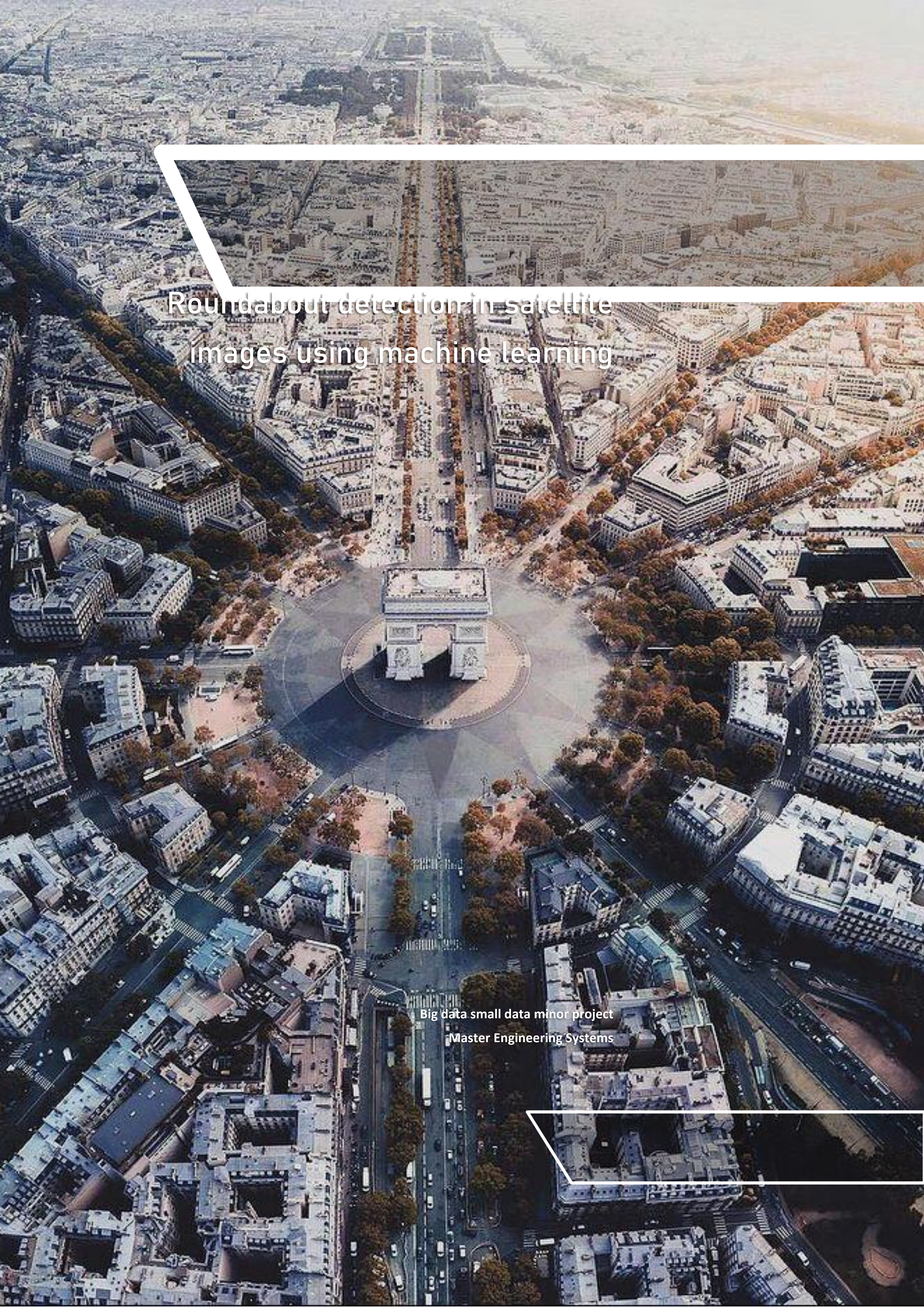
Roundabout detection in satellite images using machine learning



Big data small data minor project

Master Engineering Systems

FRONTMATTER

Name/number of the group: BDSD Minor project group 3

Name of students: Jerom Mestrum, Anurag Shinde, and Sjoerd Riemersma

Company: HAN University of applied science
Client: Dixon Devasia

HAN Supervisor: Aishwarya Aswal

Date: 12th of June, 2024

STATEMENT OF OWN WORK

Master: Engineering Systems
Date of submission: 17-06-2024
Manner of submission: handin app report: Big data small data – Minor project
Name, student number and phone number: • Jerom Mestrum student nr. 2129133 - 06203960177 • Anurag Shinde student nr. 2132290 - 0684249082 • Sjoerd Riemersma student nr. 1635862 - 0683141066
Signature(s): Jerom [illegible] [illegible]

By signing this form, we declare that the above-mentioned report that we have submitted (hereafter referred to as the document: 'Roundabout detection in satellite images using machine learning') was independently created by us without any external help and that we are aware of the rules concerning irregularities/fraud as set out in the degree statute.

Parts of the document that are literally or almost literally cited from external sources (such as internet, books, journals etc.) have been referenced by us according to the IEEE norm, for example, or preferably the Elsevier norm (e.g. footnote) in the cited part of the text (in italics).

SUMMARY

CONTENTS

1 INTRODUCTION

This project is part of the ZEFES project launched by Horizon. They created this project to proof that it is possible to deploy 9 different long-haul truck configurations in various use cases [1]. This project aims to contribute to the ZEFES project by measuring the radius and position of various roundabouts, making it possible to simulate the manoeuvrability around the roundabouts on the set out route. Since as Hough transform method was already tried by researchers from HAN Automotive Research, it is chosen to test the effectiveness of machine learning.

Decarbonizing freight transport is crucial for reducing greenhouse gas emissions and meeting international climate targets, such as those set by the Paris Agreement. Long-haul freight transport traditionally relies on diesel-powered vehicles, which contribute significantly to pollution. The adoption of battery electric vehicles (BEVs) and fuel cell electric vehicles (FCEVs) in combination with smarter truck-trailer combinations offers a promising solution to this problem.

1.1 Background

Currently, Horizon launched a project called the ZEFES (Zero Emissions flexible vehicle platforms with modular powertrains serving the long-haul Freight Eco System) project. This project addresses the decarbonization of long-distance freight transport by demonstrating real-world applications with battery electric vehicles (BEVs) and fuel cell electric vehicles (FCEVs) across Europe. The ZEFES project aims to deploy 9 different long-haul trucks configurations (BEV and FCEV) in various use cases [1].
HAN Automotive Research (AR) is involved in a project to develop a tool called- right vehicle right job. With long modular heavy-duty vehicles, it is important to see if they are fit for the current infrastructure. Difficulties that arise are for example: Sharp corners and roundabouts. These infrastructures pose challenges to manoeuvrability of the long modular vehicles. HAN AR has been asked to investigate a way to determine the size and position of these difficult infrastructures based on satellite pictures. An initial model has been created that can determine the position and radius of circular roundabouts (Figure 1). However, this model is based on a mathematical approach. The strongest circle then possibly represents the roundabout infrastructure. It also has its downsides, mainly being capable of detecting only circular roundabouts. The more 'complex' roundabouts cannot be detected, and it sometimes struggles with lower quality pictures.

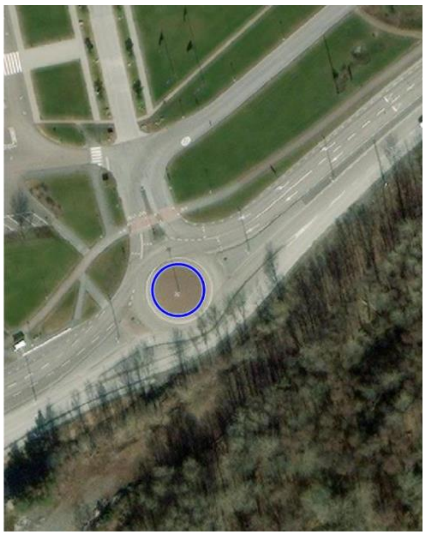


Figure 1: Roundabout detection with current mathematical model [15].

1.2 Problem definition

Next to the mathematical approach, there is another field of interest. Another interest is to determine the possibility of creating a machine learning (ML) algorithm that can predict the location of the centre island and radius of a circular roundabout in pixels. If it is deemed possible, the ML model performance (e.g. accuracy, speed) can be compared to the current mathematical model and assessed if further investigations are profitable.

1.3 Problem objective

"Create a machine learning algorithm that can determine the radius- and the centre island location of a circular roundabout in pixels based on satellite images."

1.4 Research question

“How accurate and -fast can machine learning detect- and estimate the radius- and centre location of a circular roundabout?”

1.5 Approach

To achieve the objective and answer the research question some things need to be organised. It starts with generating a dataset for the ML algorithm to train, which is the most important part of a ML project, since a ML model can only be as useful as the dataset used for training. Since there is currently no dataset available, the initial goal is to determine how a dataset can be created that is large enough and includes all relevant information to detect the wanted output. At the same time, possible architectures have been investigated that have potential to reach the objective. After that, the architecture can be trained with all the hyper parameters and possibly optimized. Finally, the model needs to be assessed on performance using a part of the dataset separated for final testing purposes.

1.6 Report outline

This report is organized using the minor project template and following the machine learning workflow from [2] presented in Figure 2. First, a literature survey is presented which discusses four papers that are related to the topic of interest within this project and their implemented ML models. Second, the methodology is presented where the question "how something can be implemented" is answered. After that, a reflection is made on the selected approach and is discussed. Third, the results of the selected approach are presented. After that, the results and outcomes are discussed. Finally, conclusions are formed, and recommendations are provided for possible steps that can be taken after the project finishes.

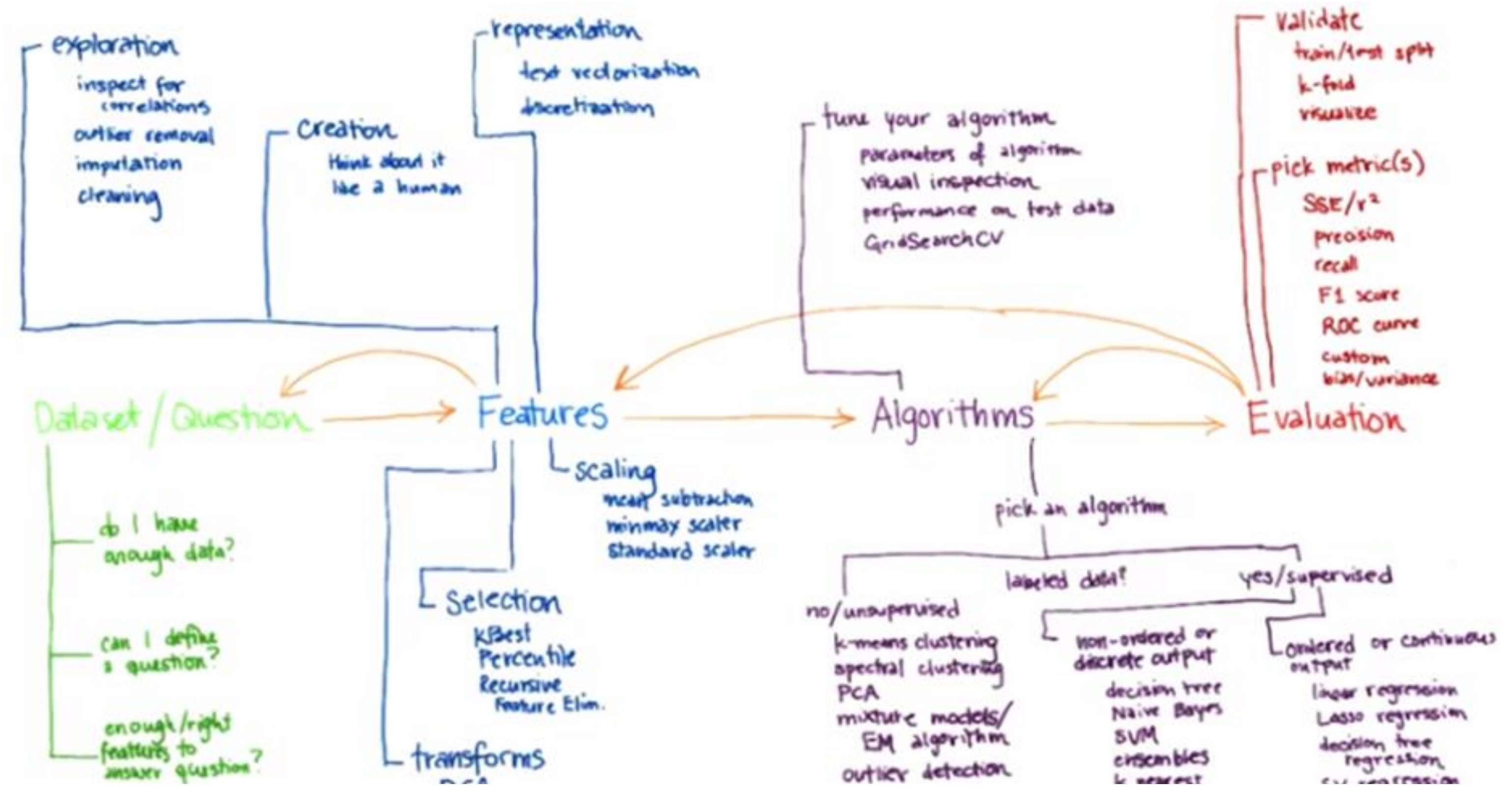


Figure 2 Machine learning workflow [2]

2 LITERATURE SURVEY

This section offers a foundation upon which the methodology, analysis, and discussions are built in this project plan. This project focuses on detecting circles, such as centre islands of roundabouts in images of highways using machine learning techniques. In this section, some of the papers reviewed in the literature study are briefly discussed. The papers go through several related domains such as digital image processing, pattern recognition, geometric shape detection, and the use of machine learning in computer vision. These papers form the foundation for the methodology of the project.

2.1 Survey on Hough Transform, Theory, Techniques and Application:

The original use of the Hough Transform was for detecting lines. However, it evolved over the years to detect shapes such as circles and ellipses. The Circular Hough Transform (CHT) depends on converting grey-scale images to binary images. It uses a preliminary edge detection technique such as Sobel and Canny and additionally depends on a voting scheme. It mainly relies on the equation of circle having three parameters and requires more computation time and memory storage.

The basic algorithm of the CHT is as below:

```
1. Extract edges of the image using Canny detector.
2. Create a 3D Accumulator array of zeros, A[x,y,r] for center locations and radius values according
   to the image dimensions.
3. For each edge point $(x_i, y_i)$ in the image:
   3.1 For each candidate radius r:
       3.1.1 For each candidate x for center horizontal location:
             3.1.1.1 Find each possible y verifying the equation:
                     $(x_i - x)^2 + (y_i - y)^2 = r^2$
             3.1.1.2 Increment the cell:
                     $A[x, y, r] = A[x, y, r] + 1$
4. Search for local Maxima in A[x,y,r], to obtain the center locations and the radii of the circles in
   the image
```

Basically, it maps any three points to a three-dimensional parameter by traversing all the edge points to determine the true circle, which leads to serious computational time consumption.

The below image from this paper is the result applying CHT to three objects:

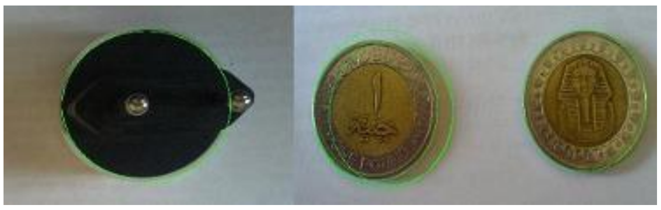


Figure 3 Application of CHT to three objects

Here it can be observed and was also mentioned in the paper that the shades affect the detection of the coin in the middle. Therefore, an approach of concentric circle detection was also proposed, where the image is first pre-processed by performing edge detection and then the circles are allocated using the gradient Hough Transform, and then finally the radii are found using the (1D)-Hough Transform.
The detection efficiency is improved by the image discretization, and by reducing the resolution ratio in the process of circle center detection. The proposed combination of the gradient HT and the 1D-HT is reliable to noise and can deal with distortion, and edge discontinuity.

A fast randomized HT (FRHT) for circle detection was also mentioned, which requires less computation time. In this approach, one point is picked at random as a seed point, then a checking rule is used to confirm if it is truly on the circle or not. This method is more suitable for dense images compared to the previous Hought Transform versions.

Overall, this paper provides a comprehensive overview of the Hough Transform, its variations, and applications over the years. It highlights the crucial milestones, including the circular Hough Transform, and discusses solutions to challenges like high storage and computation time. This survey is valuable for understanding the evolution and application of the Hough Transform in detecting circles and other shapes. [3]

2.2 A Fast Circle Detection Algorithm Based on Circular Arc Feature Screening:

This research introduces a novel circle detection algorithm that emphasizes circular arc feature screening to improve the efficiency and accuracy of circle detection in complex images. The method proposes enhancements to the fuzzy inference edge detection algorithm, utilizes two feature matrices for stepwise sampling, and employs a square verification support region for identifying true circles. It demonstrates superior speed, accuracy, and robustness compared to traditional methods like the Random Hough Transform (RHT) and the Random Circle Detection (RCD). [4]

2.3 Circle detection in images: A deep learning approach

This paper discusses an experiment of detecting circular objects in underwater images using the latest deep learning algorithms instead of conventional algorithms. In the experiment a real-time detection rate is required, as it was used for an underwater submarine. Therefore, meta-architectures that are lightweight and faster such as SSD (Single Shot Multi-Box Detector) and Faster R-CNN were considered for the experiment.

The 'Faster R-CNN' carries out detection in two stages. The first stage extracts features, and the second stage feeds it to the remaining feature extractors to predict a class and class-specific box.

The below image shows the architecture of a Faster R-CNN network:

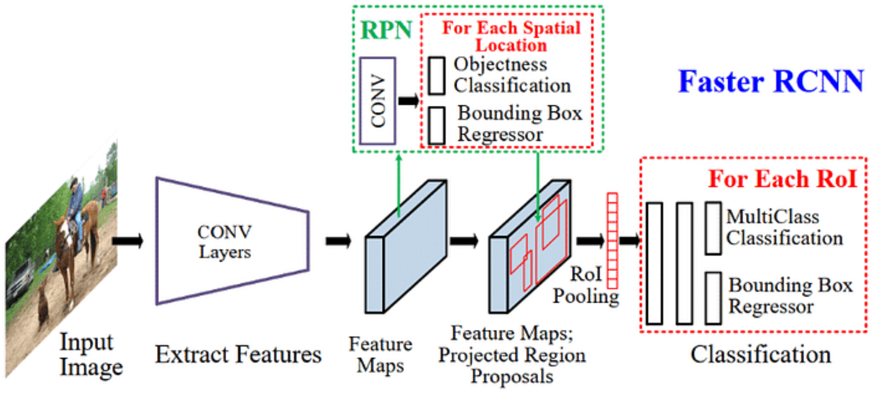
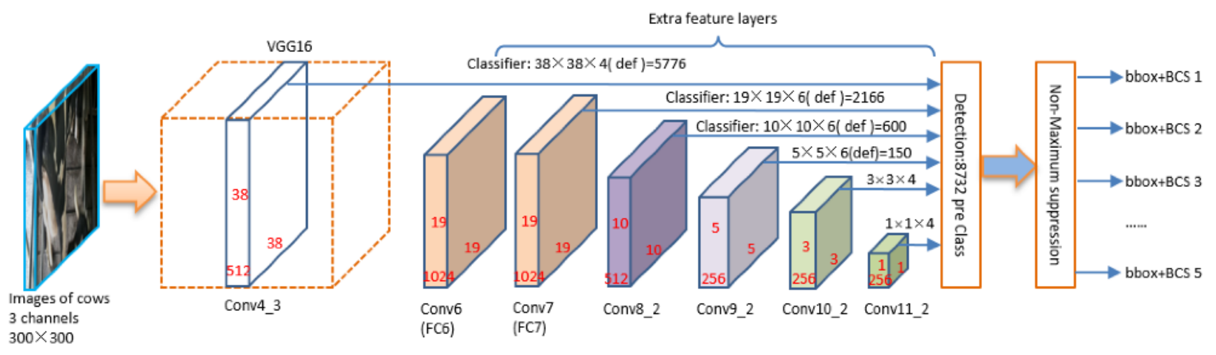


Figure 4 Faster R-CNN Architecture

The 'Single Shot Multi-Box Detector' architecture uses a single feed-forward convolutional network to predict classes and anchor offsets.
The below image shows the architecture of an SSD network:



Both Faster R-CNN and SSD were tested on a set of artificially generated datasets and a dataset of pictures of circular objects underwater.

The below image shows a table from the paper which shows the results of testing on the artificial datasets:

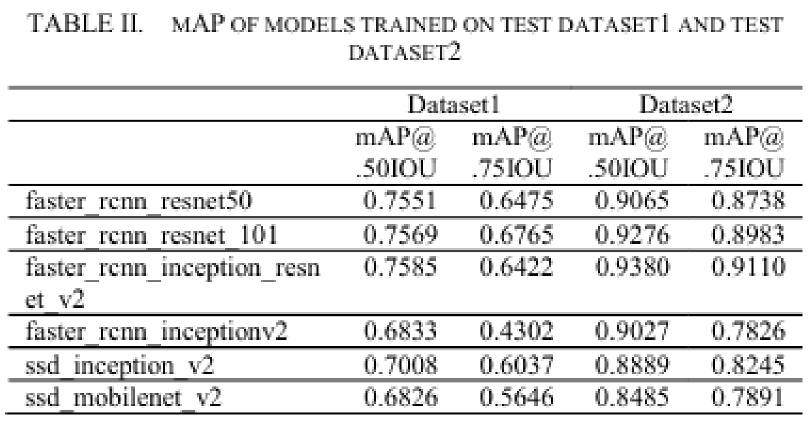


TABLE II. MAP OF MODELS TRAINED ON TEST DATASET1 AND TEST DATASET2

	Dataset1		Dataset2	
	mAP@ .50IOU	mAP@ .75IOU	mAP@ .50IOU	mAP@ .75IOU
faster_rcnn_resnet50	0.7551	0.6475	0.9065	0.8738
faster_rcnn_resnet_101	0.7569	0.6765	0.9276	0.8983
faster_rcnn_inception_resnet_v2	0.7585	0.6422	0.9380	0.9110
faster_rcnn_inceptionv2	0.6833	0.4302	0.9027	0.7826
ssd_inception_v2	0.7008	0.6037	0.8889	0.8245
ssd_mobilenet_v2	0.6826	0.5646	0.8485	0.7891

Figure 5 Mean Average Precision of testing on artificial datasets

It was observed that deep learning was faster and more accurate than circular Hough Transform in detecting both real and artificial images. Among the considered networks, Faster R-CNN provided higher accuracy than SSD although SSD was a lot faster. [5]

This paper proved that the method of Convolution can be used as a reliable and efficient feature extractor, hence more literature research was conducted about this method and an interesting research paper was found to be greatly helpful and has been discussed next.

2.4 A new approach for the regression of the centre coordinates and radius of the solar disk using a deep convolutional neural network

In this paper a case similar as this project has been investigated. It focussed on the regression of the centre location and radius of a hydrogen alpha solar disk. The architecture used is a convolutional neural network (CNN). It has been trained on a dataset containing 93686 solar observation images. The research investigated an initial approach training a CNN without complicated images. For further implementations, things like clouds and different atmosphere need to be considered as they have a significant impact on model prediction. The final performance of the model is a mean squared error (MSE) of 1.6 pixels on the separate test set. This performance is promising, since the unsharp edge can already cause this kind of deviation. Next to the results, the authors also provided the created Python code to train and evaluate the model. [6]

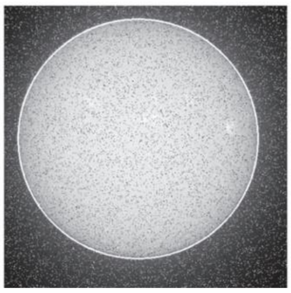


Figure 6 Circle detected on solar disk with white noise density of 0.09 [6]

The approach used by the authors is promising. Therefore, a deeper dive in what a CNN does has been performed. This starts with the question: "What is a convolutional neural network and how does it work?". A CNN is a neural network architecture that consists of three different types of layers (Table 1).

Table 1 Convolutional neural network layers' explanation

Type of layer	Purpose
Convolutional	Extraction of features through feature maps
Pooling	Reduce size of feature map and computational complexity of the model
Fully connected	Regular neural network layers. With activation functions, the outputs are regressed.

A convolutional layer works with as the names suggests convolution. Convolution in image processing is a mathematical operation of a matrix (kernel or filter) applied to an image. The result is image with certain features highlighted which is called a feature map. This feature map is then the input for another layer, which can be a convolutional layer or a pooling layer. A pooling layer also applies a kernel to an image/feature map and can perform different mathematical operations. Two most used operations are: Max pooling and average pooling. Max pooling takes the maximum value of the filter at the current position on the image and plots it in a single cell on a new feature map. Average pooling takes the average of the cell values in the kernel and plots it in a single cell on a new feature map. Pooling reduces the image size and creates a more abstract version of the feature map created by the convolutional layer. While for humans the feature map can seem off, it contains enough information for a computer to use it for prediction. Therefore, pooling is valuable for decreasing computational cost.

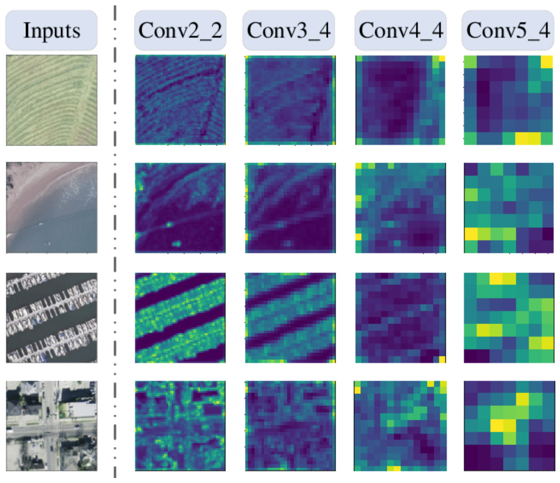


Figure 7 Visual representation of an image going through multiple convolutional- and pooling layers [16]

Figure 7 presents a visual representation of an image going through multiple convolutional layers that extract image features, and pooling layers decreasing image size while detaining feature information necessary for a machine. After the desired number of convolutional- and pooling layers, the final feature map is provided to fully connected layers, which are ‘regular’ neural network layers. These layers asses each pixel intensity with a certain weight and finally regress or classify the desired output.

2.5 Summary of literature survey

These papers provide a deep dive into the techniques for circle detection in images, offering insights into both traditional methods such as the Hough Transform and advanced machine learning approaches. The paper discussed in chapter 2.4 presents a promising approach that has similarities to this project and has a Python code to provide insights in their approach. Therefore, it has formed a base for the project execution.

3 METHOD

In this chapter the research design is presented and discussed. The method is ordered as: Dataset creation, architecture- and algorithm selection, and model assessment. The sub-chapters are chronologically ordered to create a logical step-by-step understanding of the project. Initially, a CNN had been investigated, prepared for training, and attempted to train. However, after implementation it became evident that training the architecture only once on the 100,000 images would be already impossible time wise, since 10,000 images takes 7 hours of training to complete (70 hours for the total dataset). Therefore, after consulting with the project owner a different approach has been determined. Which is to use another way of extracting features from images and feed them to a fully connected neural network. The method for each approach will be explained to present knowledge about the subject and possibly head start future research.

3.1 Dataset generation

Each ML project starts with generating or gathering a dataset as it determines the outcome of the project. As mentioned in chapter 1.5, there was no dataset at the start. The size and complexity of the dataset determines what kind of architecture can be used. For example, if only a database of 100 images can be created, a complex neural network cannot perform as intended. Since the regression of the outputs in images is not a simple task, a considerable dataset needs to be created. The extraction of roundabout satellite images alone takes too much time to create the necessary size dataset. Therefore, artificial image creation is used to increase the size.

For dataset creation, two methods have been used: Procedural artificial image creation and satellite image extraction. Chapter 3.1.1 to 0 discuss the procedural artificial image creation, and

The definition of procedural data generation is that there is no human intervention for creating new data. If possible, this method can create an unlimited source of training data for a machine learning algorithm to learn from. In this project the generated data is not perfect but used to create a starting point. In this chapter a brief description of how a dataset of +/- 30,000 procedurally generated images is created. This data set is afterwards randomly split into: 80% training, 10% validation, and 10% test.

3.1.1 Background

The background colour can have a big effect in the overall colour intensities. The biggest challenge is to create a background that is unique in every new created picture. This is achieved by using a shader (A script that controls how colour, reflection etc. are perceived), which imports two different materials and blends these using gaussian noise as shown in Figure 8. Since this gaussian noise is completely random generated the colours in the background are to. It is possible to use different materials to be blended.

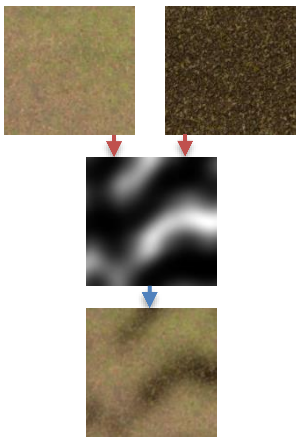


Figure 8: Backgrouna shader, own work

3.1.2 Roads

Creating a randomized road network is the most complex feature of this data generating algorithm, this chapter will only scratch the surface. First a 2D Boolean array is created to form a grid. It starts with a 5x5 grid sized roundabout that can vary its radius. The 25 squares the roundabout is placed on will be set to true to keep track of which squares are filled. The coordinates of the exits of this roundabout are added to a list. A function of the program will be triggered when any coordinates are present and will place a road tile (shown in Figure 9) using the first in first out method. If the placed road tile is not occupying a filled square following the Boolean array it will set its own coordinate to true and adds the coordinates of newly created exists to the list. If a new tile is placed on a filled square or it goes outside the range of the grid it will eliminate the instantiated. When the list of exits is empty a random road network is created.

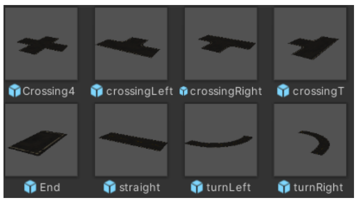


Figure 9: Road tiles, own work

3.1.3 Vehicles

Each tile has different points where it is possible for a vehicle to be instantiated. Determined by the master script the likeliness of a vehicle to be instantiated at one of those points can be between 0 and 30 percent. If a vehicle is instantiated it is possible to have different type and colours of vehicles.

3.1.4 Buildings

A function goes through the 2D Boolean array mentioned in chapter 0

Roads which keeps track of which grid squares are filled. Every time it encounters a square that is empty but is next to a square that is filled by a road it will instantiate a building with the orientation that connects the front door to this road.

3.1.5 Vegetation

The script will randomly place between 0 and 1000 plants on in the scene. All plants have a component which in unity is called a "collider" (shown in Figure 10). It is possible in unity to check if this collider is colliding with other colliders. The building, roads and vehicles have colliders to making it possible for a script to eliminate the plant object again if the collider collides with other objects.

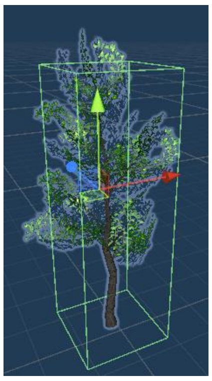


Figure 10: Collider (green box), own work

3.1.6 Light and camera randomization

The light used in this scene is a directional light which simulates the sun. The direction and intensity of this is randomized to create shadows in different directions and give the impression of different times of day.

Since satellites are very far from the area, they are photographing the light hitting the lenses are near laminar. An orthographic camera has only picked up light in a certain direction. Placing the simulated camera above the scene and making the size of the camera covering the scene will create the same effect as a real camera zooming in from a very big distance. The orientation tangent to the background is randomized in 360 degrees and the actual size of the camera is randomized as well to get scenes with different amounts of zoom.

3.1.7 Possible improvements

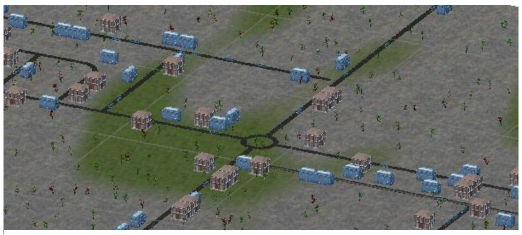


Figure 11: 3D scene, own work

The algorithm for creating random automotive environments can have a lot more implements since the scene is created in 3D as well as shown in Figure 11. However, in this project the algorithm is not perfected yet This is because the data had to be created when the convolution neural network was ready, not when this algorithm was fully completed.

From an architecture perspective the algorithm is nearly as advanced as it can be. However, from a content perspective there can be a lot of improvements. A table is made to show the possible content improvements and how easy they are to implement based on availability of the assets and needed adjustments to the software.

Table 2: Improvements for procedurally data generating program

	Used assets	Ideally used assets	Difficulty
Vehicles	1	20	Very easy
Buildings	2	15	Easy
Ground textures	2	10	Medium
Road textures	1	5	Hard
Vegetation	8	8	N/A

3.1.8 Satellite image extraction

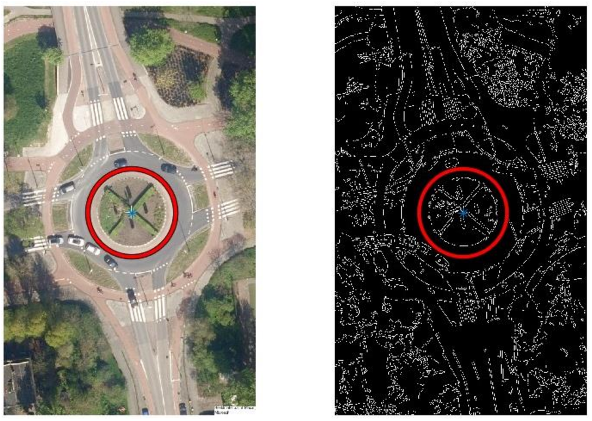


Figure 12 (Left) Hough-transform to collect labels & (Right) detected edges during preprocessing

Artificially created images can only represent real-life up to a certain degree. Therefore, satellite images need to be extracted to train and test the ML model for real-life performance. A Matlab code that can extract roundabout images on a route based on start- and end coordinates is provided. However, this way image extraction would consume too much time. Therefore, roundabout locations have been gathered and their satellite images extracted using part of the Matlab code. After image extraction, the centre location and radius of each roundabout need to be determined to form the image labels. The previous approach of the Hough-transform can provide a solution. The images are passed through the same pre-processing steps and are after the Hough-transform visually assessed on accuracy. Unfortunately, due to the performance of the previous method, only 400 images out of 770 images have been processed correctly. Since this process takes an immense time-effort, only 400 images can be used partially for training and partially for testing the final model.

3.2 Architecture- and algorithm selection

Next to the dataset, an appropriate architecture- and algorithm need to be determined. The literature survey presented that the main reason of different architectures for image recognition is time efficiency and accuracy. The goal of this project is to determine the outputs with an accuracy that is visually acceptable. Time efficiency is a plus, but not a criterion. Therefore, initially the more sophisticated CNN had been selected. Another reason is that a Python code to train and evaluate that specific architecture is provided by the authors [6]. Also, feature extraction from images with a non-constant background and clutter is extremely difficult. Since a CNN extracts- and interprets features based on importance for the output, it would be the ideal candidate.

As explained in the introduction of chapter 3, two different approaches have been investigated. First, the initial approach will be explained which is the CNN (Chapter 3.2.1). Finally, the less computational expensive approach will be explained which uses feature extraction through histogram of oriented gradients (Chapter 3.2.2).

3.2.1 Convolutional neural network approach

The initial approach was to implement a CNN. Since a CNN has been investigated in the literature survey, only the question: "How can a CNN be implemented for this project?" needs to be answered. As mentioned in the literature survey, circle center location and radius detection of solar disk images with a CNN has been performed before [6]. The authors discussed their implementation of a CNN to detect the same outputs as required for this project but on different kinds of images. Their model is based on the VGGNet-13 architecture that was created by other engineers for the ImageNet Challenge in 2014 [7]. This architecture is expanded with 4 extra convolutional layers and instead of classification, regression has been selected in the last layer (Chapter 6.1). Their model was trained on +/- 75,000 images and randomly relocated to increase model robustness. The final accuracy of the model reached a mean square error (MSE) of 1.75, which means that the average deviation of the predicted values of the true values was 1.75 pixels. This is an incredible result considering the size of the circles (Radius of 210 to 230). Another reason to investigate their implementation is the provided Python code created by the authors to train and evaluate their model, with the final weights included as well.

The model has certain algorithm choices as well. These are: Which optimizer to use, how to initialize kernel values, which activation unit to use, which padding to use, and if necessary which type of regularization and what value to use. In Table 3 the algorithm choices are presented and briefly explained. Since the goal is not immediately to optimize the model but to explore the possibility, the current selected methods are evaluated and if deemed applicable are used.

Table 3 Most important algorithm option choices with brief explanation

Algorithm option	**Choice**	**Brief explanation**
Optimizer	Adam [8]	Gradient descent method with adapting learning rate.
Kernel and weight initializer	He Normal [9]	Initializer with normal distribution centered on 0 with standard deviation = $\sqrt{\frac{2}{number\ of\ input\ units}}$
Activation unit	Rectified linear unit (ReLU) [9]	Every negative value becomes zero and the positive values are not altered (Figure 13).
Regularization	L2 [10]	Regularization with same lambda for each weight.
Padding	Same [11]	Determines size difference of input and output of convolutional layer.
Stride	1 [12]	Step size of the convolution along height and width.

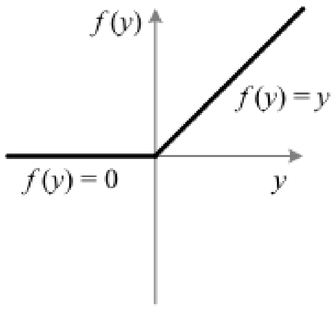


Figure 13 ReLU activation function visualization [9]

All the algorithm choices are applicable for the application. This is determined by looking at each choice.

- The Adam optimizer has shown a large potential in convolutional neural networks and is becoming a standard for deep learning [8].
- The He Normal initializer is created as a robust initializer to increase the learning ability of a highly non-linear system with ReLU [9].
- The ReLU activation unit is effective at creating non-linearity in models and is a standard for complex deep learning problems [9].
- L2 regularization is the normal regularization which influences all weights the same and does not need initial information about which weights to prioritize [10].
- Setting padding to 'same' excludes the risk of input- and output size changing during convolution. This makes it easier to understand and less prone to errors [11].
- Stride is the distance the convolutional kernel takes after each computation. The default is 1, which is suitable for most implementations. If stride is to be increased, the input- output size during convolution could change if padding is not set to 'same' [12].

Looking at the selected decisions, no changes will be made since every choice is suitable for the application. After analyzing the solar disk implementation, architecture- and algorithm selections, the provided Python code had been checked, and the errors had been resolved. After the first attempt to train the model with the 100,000-image data set, it became evident that training the model only once already would take too much time (7 hours for 10,000 images for 50 epochs out of 100,000). Therefore, another approach needed to be determined. The final idea has been to exclude the convolutional and pooling layers, and extract image features a different way. The selected method is called Histogram of Oriented Gradients (HoG) and will be discussed in chapter 3.2.2.

3.2.2 Histogram of oriented gradients with neural network approach

The final approach that has been determined is using a feature extractor: Histogram of oriented gradients (HoG) and feed these features to a neural network. The change of excluding feature extraction using convolutional layers which take considerable computational effort, should result in faster training times. First, the question: "What is histogram of oriented gradients and how does it work?" needs to be answered.

HoG is a way of extracting image features. It does so by plotting each pixel orientation and gradient in a histogram. This way, it simplifies the representation of an image by only keeping crucial information and reducing influence of background noise (
Figure 14).

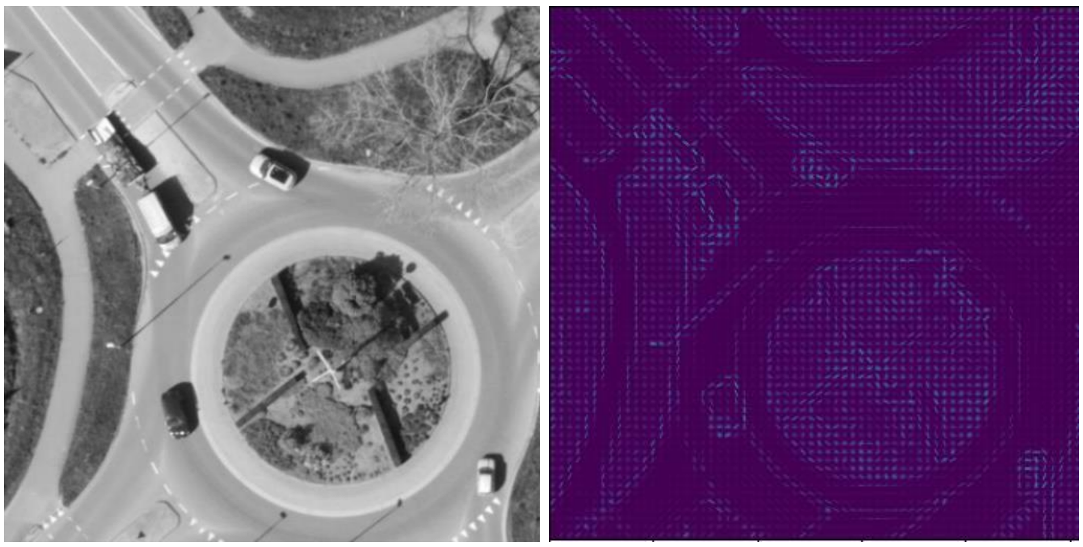


Figure 14 Grayscale image (Left) converted to HoG image (Right) [13]

To perform HoG feature extraction an image needs to be converted to a 62x128 shape where as much pixels as possible need to be kept. Fortunately, the gathered images have already been converted to 512x512, which can be resized to 128*4x64*8 in Python. Next, the HoG command can be used to extract the image (Like
Figure 14) and the features (Gradient direction and magnitude). The specifications provided to the HoG command are the same as the original creator ones [14]. If the specifications are altered to represent higher resolution features, the process slows down tremendously and does not suit the application.

After the HoG features have been determined, they are provided to a neural network. The architecture of the neural network consists of five layers. The number of layers and neurons has been determined through optimization. The architecture started as a simple one-layer NN but has been extended to a more complex multi-layer architecture based on performance results. The input layers have as many units as the features created by HoG extraction, the first hidden layer 128 units, the second 64, the third 32 and the output layer 3 units due to the required outputs. The other specifications like activation function, optimization algorithm etc. has been taken from the CNN approach. Regularization can only be determined by training multiple times with the same architecture and varying lambda. The dataset is trained initially with lambda equal to 0 and if increased, the validation loss increases. Therefore, regularization is not used in this final application.

3.3 Model assessment

Model assessment considers which performance metrics need to be implemented. The performance metric is determined based on the kind of prediction used. Since the goal is to regress the circular roundabout centre coordinates and the radius, a solid option is the root mean squared error (RMSE). RMSE will give enough information of model performance to assess the usability for the intended implementation since it represents the average absolute error. Next to RMSE, a visual check will be performed since the performance criteria is to visually match up. However, since this is not a measurable performance metric, RMSE needs to be used as well.

3.4 Approach discussion

First, the final machine learning approach is discussed. After that, the method used during this project will be evaluated and possible improvements for future machine learning projects will be discussed.

The final approach consists of three components. The data set, pre-processing which includes feature extraction, and the neural network architecture and algorithm. Unfortunately, the data set created with artificial image creation did not result in a desired performance. After training with all 30,000 images, the RMSE for training and validation were: 8.73 and 44.19 pixels. Since the artificial images have been created to act as 'training wheels' for the ML model, the next step has been to continue training with real images. However, after continuing training with 304 real images, the validation RMSE improved drastically to: 25.04 pixels. This raised the question to start training with only the 304 real images. After initiating multiple times and training with only the 304 real images, the training- and validation RMSE decreased to 0.4049 and 22.69 pixels. Unfortunately, this means that the effort spent on creating the large artificial data set could have been better spent elsewhere. A further discussion about the results is provided in chapter 5 Discussion.

The pre-processing consists of: Gray scaling the resized 512x512 images and extracting HoG features. These pre-processing steps are not time consuming, except for the HoG features. However, this is only an issue for large training data sets, since one image only takes +/- 0.8 seconds to convert. The selected neural network architecture is a rather simple version of a neural network. Therefore, the training time is mainly dependent on the training data set size. Overall, the pre-processing and model architecture can quickly predict the three outputs [s]: 0.019 for grey-scaling, 0.5 for HoG feature extraction, and 0.9 to predict (Total of 1.419 [s]).

After the knowledge about the training time of a complex CNN, looking back it would have been wiser to start simple and expand in complexity if necessary. Next to the large training time there is another learning outcome, which is to start writing a code from scratch. Using the assumed 'working' Python code provided by [6], it is never completely clear what the authors had in mind while writing it. Next to that, almost no code is commented through extensively so that anyone can implement and replicate it. Therefore, the final HoG approach has been started from scratch and excluded most errors that were present in the originally provided code. However, to get acquainted with Python and how to utilize it for ML, the CNN code did provide a head start.

4 RESULTS AND DISCUSSION

In this chapter, the results are presented and discussed. The results are presented in chapter 4.1 and they are discussed in chapter 4.2.

4.1 Results

As mentioned in chapter 3.4, the artificial image data set has not been used for final model training. Since only training on the real images increased model performance, no artificial images have been used. However, since there are little real- compared to artificial images, a different data set split needed to be used. The 400 images have finally been split into: 304 for training, 76 for validation, and 20 for testing. Initially, less training images have been used. However, this resulted in a higher validation loss. Every time after taking more images for training the validation loss decreased. Therefore, the final split has been determined. With this split, the final training performance (RMSE) has been measured: 0.4049 for training and 22.69 pixels for validation. After training the test set can be utilized to determine the final performance of the model, which resulted in an RMSE of 21.39 pixels. In the figure below four images are presented out of the testing set.

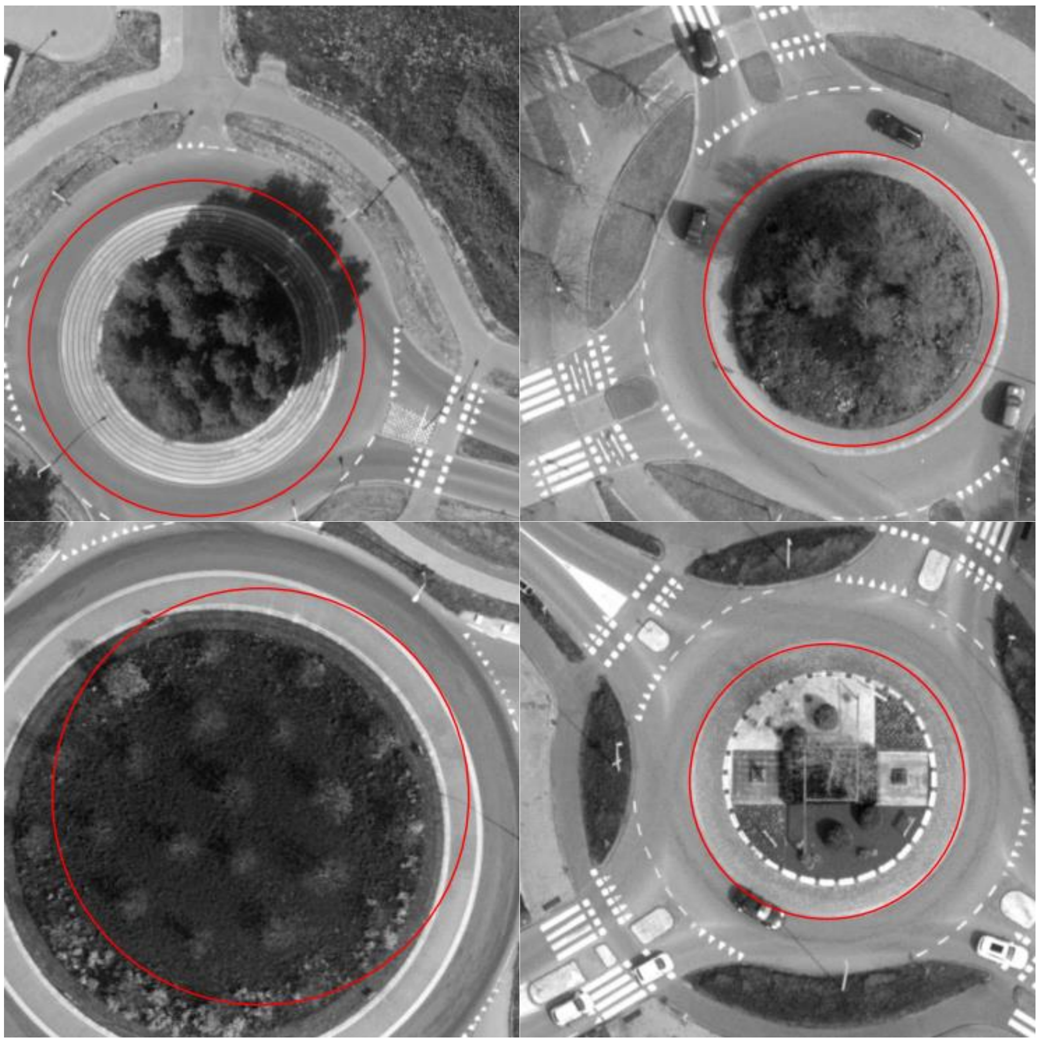


Figure 15 Four test set images to explain performance. One good image (Bottom right), one decent image (Top right), and two bad images.

The 20 testing results are a mix of the presented images in Figure 15. Currently, the model is not reliable to predict the outputs accurately each prediction. However, the model is on the right track since validation loss decreases if more images are used for training. Evaluating the test set results, it appears that the model most of the time touches an edge and expands the circle outwards in the direction of the roundabout. This could mean that the model takes a strong HoG feature that represents a circular edge. Next to this observation there is no reasoning to how it could perform better without increasing the training data set size.

With the current results, the model is not sufficient to be implemented in the future application. The performance criteria is for the circles to visually match up, which is not occuring each prediction. Therefore, further training of the model is necessary and possible if more real images are gathered.

4.2 Discussion

Based on the results and the discussed methods, two things need to be discussed. First, the RMSE values of all data set parts need to be discussed. Second, the reliability of the final model performance.

After analysing the train and validation loss, an assessment of bias and variance needs to be performed. Usually, high performance on the training set and low on the validation set indicates a high variance problem (Overfitting). However, since the training performance is high and the validation is not horrible, it looks like another case that can occur with neural networks. Which is a combination of a high variance and a high bias problem (Underfitting). The current guess is that this phenomenon is occurring and can be fixed by increased the number of real images used for training.

The final model performance is measured as 21.39 RMSE pixels. However, this number is less accurate than usually. Since more training data was required, some testing images have been transferred to the training set. This results in only 20 images available for testing and to determine final model performance. Since the performance measure is an average (RMSE), a larger testing set would most likely result in a more reliable final model performance measure.

5 CONCLUSION AND RECOMMENDATION

In this chapter, the conclusion regarding the research question and objective is provided. Any answers for remaining questions are discussed as recommendations.

5.1 Conclusions

First the completion of the project objective is assessed. The objective has been: "Create a machine learning algorithm that can determine the radius- and the centre island location of a circular roundabout in pixels based on satellite images.". This objective has been reached, since a ML algorithm has been created that can determine the radius- and the centre island location based on a satellite image. However, it has been reached up until a certain level.

This level of achievement is based on the answer to the research question: "How accurate and -fast can machine learning detect- and estimate the radius- and centre location of a circular roundabout?". As mentioned in chapter 3.4 and -4.1, the model ML can predict with an assumed RMSE of 21.39 pixels in 1.419 seconds.

Based on these statements, the conclusion is that the model is not yet accurate enough to be utilized for the intended purpose.

5.2 Recommendation

Based on the conclusion and the discussed results, one recommendation has been determined. Which is to gather more satellite images of circular roundabouts to train the model on. It is believed that if enough images are gathered, it could result in a reliable ML model that can predict the desired outputs.

REFERENCES

[1] E. U. „ZEFES Introduction,” [Online]. Available: https://zefes.eu/.

[2] Udacity, „Introduction in machine learning,” [Online]. Available: https://classroom.udacity.com/courses/ud120/lessons/3033138574/concepts/30962787190923.

[3] „A Survey on Hough Transform, Theory, Techniques and Applications,” [Online]. Available: https://arxiv.org/abs/1502.02160.

[4] „A Fast Circle Detection Algorithm Based on Circular Arc Feature Screening,” [Online]. Available: https://www.mdpi.com/2073-8994/15/3/734.

[5] „Circle detection in images: A deep learning approach,” [Online]. Available: https://www.researchgate.net/publication/344801439_Circle_detection_in_images_A_deep_learning_approach.

[6] G. Zhu, G. Lin, D. Wang en X. Yang, „A new approach for the regression of the centre coordinates and radius of the soldar disk using a deep convolutional neural network,” *The Astrophysical Journal,* 2020.

[7] K. Simonyan en A. Zisserman, „Very deep convolutional networks for large-scale image recognition,” in *ICLR*, Oxford, 2015.

[8] Keras, „Adam,” Keras, [Online]. Available: https://keras.io/api/optimizers/adam/.

[9] K. He, X. Zhang, S. Ren en J. Sun, „Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification,” Microsoft research, 2015.

[10] Google, „Regularization for simplicity: L2 regularization,” 18 July 2022. [Online]. Available: https://developers.google.com/machine-learning/crash-course/regularization-for-simplicity/l2-regularization.

[11] K. S. Mehta, „Difference between 'SAME' and 'VALID' padding in Tensorflow,” 25 2 2022. [Online]. Available: https://wandb.ai/krishamehta/seo/reports/Difference-Between-SAME-and-VALID-Padding-in-TensorFlow--VmlldzoxODkwMzE.

[12] „Keras.Conv2D-class,” 11 1 2023. [Online]. Available: https://www.geeksforgeeks.org/keras-conv2d-class/.

[13] „Histogram of oriented gradients,” [Online]. Available: https://de.wikipedia.org/wiki/Histogram_of_oriented_gradients.

[14] N. Dalal en B. Triggs, „Histogram of oriented gradients for human detection”.

[15] D. Devasia, „Project_Introduction_Spring_2024,” Arnhem, 2024.

[16] S. Mei, K. Yan, M. Mingyang en X. Chen, „Remote Sensing Scene Classification Using Sparse Representation-Based Framework With Deep Feature Fusion,” *IEEE Journal of selected topics in applied earth observations and remote sensing PP,* 2021.

6 APPENDIX

In each appendix, a brief explanation of the content is provided at the start.

6.1 Appendix 1 Adapted VGGNet-13 CNN architecture

In this appendix, the adapted architecture created by [6] is presented. The input is a grayscale image, and each colour block represents a different kind of layer. A blue block is a convolutional layer with the kernel size in brackets and the number of filters after that. An orange block is the kind of pooling layer, and the size of the kernel in brackets. The green block is a fully connected layer with the number of neurons in between brackets.

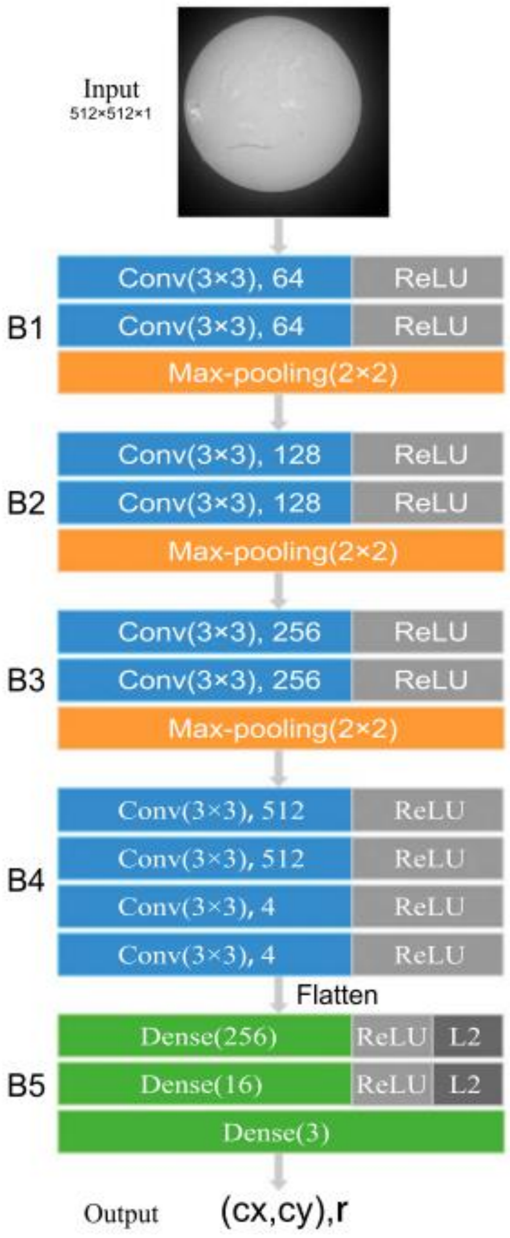


Figure 16 Adapted VGGNet-13 CNN architecture [6]