DEGREE PROJECT IN ELECTRICAL ENGINEERING,
SECOND CYCLE, 30 CREDITS
STOCKHOLM, SWEDEN 2021

Screw Hole Detection in Industrial Products using Neural Network based Object Detection and Image Segmentation

A Study Providing Ideas for Future Industrial Applications

JAKOB MELKI

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Screw Hole Detection in Industrial Products using Neural Network based Object Detection and Image Segmentation

A Study Providing Ideas for Future Industrial Applications

JAKOB MELKI

Master's Programme, Systems, Control and Robotics, 120 credits
Date: November 8, 2021

Supervisors: Jörg Conradt, Juan Romero
Examiner: Arvind Kumar
School of Electrical Engineering and Computer Science
Host company: Ericsson AB
Swedish title: Skruvhålsdetektering på Industriella Produkter med hjälp av Neurala Nätverksbaserade Objektdetektering och Bildsegmentering
Swedish subtitle: En Studie som Erbjuder Ideér för Framtida Industriella Applikationer

Abstract

This project is about screw hole detection using neural networks for automated assembly and disassembly. In a lot of industrial companies, such as Ericsson AB, there are products such as radio units or filters that have a lot of screw holes. Thus, the assembly and disassemble process is very time consuming and demanding for a human to assemble and disassemble the products. The problem statement in this project is to investigate the performance of neural networks within object detection and semantic segmentation to detect screw holes in industrial products. Different industrial models were created and synthetic data was generated in Blender. Two types of experiments were done, the first one compared an object detection algorithm (Faster R-CNN) with a semantic segmentation algorithm (SegNet) to see which area is most suitable for hole detection. The results showed that semantic segmentation outperforms object detection when it comes to detect multiple small holes. The second experiment was to further investigate about semantic segmentation algorithms by adding U-Net, PSPNet and LinkNet into the comparison. The networks U-Net and LinkNet were the most successful ones and achieved a Mean Intersection over Union (MIoU) of around 0.9, which shows that they have potential for further development. Thus, conclusions draw in this project are that segmentation algorithms are more suitable for hole detection than object detection algorithms. Furthermore, it shows that there is potential in neural networks within semantic segmentation to detect screw holes because of the results of U-Net and LinkNet. Future work that one can do is to create more advanced product models, investigate other segmentation networks and hyperparameter tuning.

Keywords

Artificial intelligence (AI), Automated assembly and disassembly, Computer vision, Machine learning, Neural networks, Object detection, Screw hole detection, Semantic segmentation

Sammanfattning

Det här projektet handlar om skruvhålsdetektering genom att använda neurala nätverk för automatiserad montering och demontering. I många industriföretag, såsom Ericsson AB, finns det många produkter som radioenheter eller filter som har många skruvhål. Därmed, är monterings - och demonteringsprocessen väldigt tidsfördröjande och krävande för en människa att montera och demontera produkterna. Problemformuleringen i detta projekt är att undersöka prestationen av olika neurala nätverk inom objekt detektering och semantisk segmentering för skurvhålsdetektering på indutriella produkter. Olika indutriella modeller var skapade och syntetisk data var genererat i Blender. Två typer av experiment gjordes, den första jämförde en objekt detekterings algoritm (Faster R-CNN) med en semantisk segmenterigs algoritm för att vilket område som är mest lämplig för hål detektering. Resultaten visade att semantisk segmentering utpresterar objekt detektering när det kommer till att detektera flera små hål. Det andra experimentet handlade om att vidare undersöka semantiska segmenterings algoritmer genom att addera U-Net, PSPNet och LinkNet till jämförelsen. Nätverken U-Net och PSPNet var de mest framgångsrika och uppnåde en Mean Intersection over Union (MIoU) på cirka 0.9, vilket visar på att de har potential för vidare utveckling. Slutsatserna inom detta projekt är att semantisk segmentering är mer lämplig för hål detektering än objekt detektering. Dessutom, visade sig att det finns potential i neurala nätverk inom semantisk segmentering för att detejtera skruvhål på grund av resultaten av U-Net och LinkNet. Framtida arbete som man kan göra är att skapa flera avancerade produkt modeller, undersöka andra segmenterisk nätverk och hyperparameter tuning.

Nyckelord

Artificiell intelligens (AI), Automatiserad montering och demontering, Datorseende, Maskininlärning, Neurala nätverk, Objekt detektering, Skruvhålsdetektering, Semantisk segmentering

Acknowledgments

First of all, I would like to thank the people at Ericsson that have been a part of this journey. Especially, I want to thank Joakim Sedvall and Sakari Salmi, who were my manager and supervisor at Ericsson. Furthermore, I would like to thank Juan Romero for his time and effort, who was my co-supervisor at KTH and who supported and guided me throughout this project.

Stockholm, November 2021
Jakob Melki

Contents

List of Figures

List of acronyms and abbreviations

ADAM	Adaptive Moment Estimation
AI	Artificial Intelligence
CNN	Convolutional Neural Network
Faster R-CNN	Faster Regional Convolutional Neural Network
FCN	Fully Convolutional Network
GPU	Graphics Processing Unit
HOG	Histogram of Oriented Gradients
IoU	Intersection over Union
MIoU	Mean Intersection over Union
MSE	Mean Squared Error
PSPNet	Pyramid Scene Parsing Network
R-CNN	Regional Convolutional Neural Network
ReLu	Rectified Linear Unit
ResNet	Residual Neural Network
RoI	Region of Interest
RPN	Regional Proposal Network
SGD	Stochastic Gradient Descent
SIFT	Scale-Invariant Feature Transform
SSD	Single Shot Mutlibox Detector
YOLO	You Only Look Once

Chapter 1

Introduction

The project is about screw hole detection on industrial products for assembly and disassembly. It provides an investigation and a base of using machine learning algorithms for screw hole detection. This could lead to further development of Artificial Intelligence (AI) and automation in industry.

The project is done at Ericsson that have a lot of products such as radio units or filters with a lot of screw holes. Thus, it is very demanding and time consuming for a human to assemble and disassemble the products. Thereby, Ericsson is interested to automate its assembly and disassembly process for such products. To do that, a system that can at least detect the screw holes is required, which is where this project comes in.

There are different areas within computer vision to detect certain objects or in this case screw holes. Two interesting areas for this project are object detection and semantic segmentation. Object detection algorithms detect objects in an image with a bounding box, whereas semantic segmentation algorithms classify each pixel to a certain class.

The most popular networks for object detection are You Only Look Once (YOLO), Single Shot Mutlibox Detector (SSD) and Faster Regional Convolutional Neural Network (Faster R-CNN). The pros of YOLO and SSD are that they have low computational time and are therefore suitable to detect objects fast in real time. While YOLO and SSD have low computational time, Faster R-CNN has longer computational time, but yields higher precision in general when it has sufficient resources and time [9], [10], [11].

When it comes to semantic segmentation for neural networks, SegNet is a popular network. It does not require much memory compared to other networks such as U-net. However, the accuracy is not always better. Other interesting networks for this project, are Pyramid Scene Parsing Network

(PSPNet) and LinkNet.

1.1 Problem

The problem that has been been investigated or solved within the framework of this project is screw hole detection in industrial objects. Many products in industries have a lot of (more than 100) small screw holes. Thus, to automate the assembly and disassembly process, one of the first steps is to detect the screw holes. In this project, the possibility of using neural networks for object detection or semantic segmentation for screw hole detection has been investigated.

Is it possible to use object detection or semantic segmentation for screw hole detection? Which of the methods are more appropriate?

1.1.1 Research questions

The main research question within this work, is stated below:

- How do neural networks for object detection (for example Faster R-CNN) and/or semantic segmentation (for example SegNet) perform in terms of accuracy when trained on synthetic data of industrial objects with around of 100 small screw holes with diameter of around 3.5 mm?

Furthermore, some additional research questions are also investigated, which are stated below:

- Which of object detection or semantic segmentation is more suitable for the scenario in the question above?
- For the area that is most appropriate, which of the investigated algorithms achieves the highest performance in terms of accuracy for detecting the investigated screw holes?

1.2 Purpose

The purpose of this master's thesis is to provide the customer company, Ericsson AB, research in terms of simulation results and documentation that is relevant for their needs. Furthermore, this project may also provide ideas and potential solutions for Ericsson in the future of automated screw hole detection. In addition, it also provides research and ideas for the academic and engineering community since it investigates the performance of the proposed algorithms on an unique dataset (industrial objects with a lot of screw holes).

Apart from screw hole detection, this project propose how one can use certain programs such as Blender to model industrial objects for data generation for object detection and semantic segmentation. It shows the potential and applications of neural networks in industry by creating its own synthetic data, which looks realistic. Thus, there is a possibility that one can solve a lot of problems that have not yet been investigated with neural networks and synthetic data, which makes this project relevant and it sets the bar for future studies of neural networks in industrial applications.

1.3 Goals

The goal of this project is to provide an investigation of neural networks for screw hole detection. This has been divided into the following three sub-goals:

1. To satisfy the engineering and academic community, it should have a research value, i.e. should answer a research question and provide a research within the area. In this case, different algorithms within object detection and image segmentation will be compared and evaluated on different datasets of product models. The possibility and performance in terms of accuracy of using neural networks within computer vision together with synthetic data will be investigated.

2. To satisfy the industry, it should be realistic and related to the company's desires. The investigated models should look similar to an real industrial product with many (around 100) screw holes and the used algorithms should be able to detect holes. The project, should also set the base or provide suggestions for further development within the area and potential implementation by using its simulation results and studies.

3. Finally, it should also satisfy the requirements as an master's thesis according to KTH's standards. In this case, the project should also be related to AI, robotics and/or control systems.

1.4 Research Methodology

The research methodology in this project is based on simulations only. The reason of only working with simulations and not actual hardware implementation is due to the time constraint of the project and the complexity of the problem in reality.

Different models of industrial products on different levels of complexity have been designed and synthetic datasets have been generated for neural networks in object detection and semantic segmentation.

Algorithms within object detection and semantic segmentation have been compared in performance in terms of accuracy of screw hole position.

1.5 Delimitations

Due to the time constraints, the algorithms were not tested on the real robotic manipulator. Additionally, the 3D models that were developed didn't represent the full complexity of the real objects.

Chapter 2

Background

Before continuing to the theory background within this project, it is worth mentioning the reason of investigating neural networks within computer vision in this project instead of the conventional methods within computer vision. As mentioned in chapter 1, this project was done at an industrial company, Ericsson AB, who wanted a system that could automatically detect screw holes in different product models, i.e. it should detect screw holes as an AI. Thus, neural networks could be a possible solution for detecting screw holes because of their ability to learn and generalize over different models. On the other hand, the conventional methods, do not have the ability to learn and generalize. Instead, they are often more programmed manually to detect certain things, which is why they are not investigated in this project.

2.1 Object detection

Object detection is a common computer vision task that classifies objects to a certain class and localize them with bounding boxes. There are two main fields within object detection. The first one consists of the conventional methods, which usually are a several step process, staring with edge detection and feature extraction with techniques like Scale-Invariant Feature Transform (SIFT) and Histogram of Oriented Gradients (HOG) [9]. However, with the evolution of deep learning and Convolutional Neural Networks (CNNs), a new field within object detection has arise. CNNs are capable to automatically extract more complex and appropriate features, they clearly outperform the conventional methods and have become the state-of-the-art methods in object detection [11].

The input data for the CNNs usually consist of one image of one or multiple objects and the corresponding output is the bounding boxes with labels of the surrounding objects. There are two approaches for CNNs in object detection, two-step object detection and one-step object detection [9].

One-step object detection involves algorithms that uses the idea of "regressing" the bounding box predictions and they do not work on every bounding box separately compared to the two step detectors. This makes one-step detectors run faster but at the cost of precision, especially in small objects and similar objects in close areas [9].

Two-step object detection includes algorithms that first identify bounding boxes that may contain objects and then classify each bounding box separately. In general, these are known to perform better in precision compared to the one-step object detection algorithms [9]. One example of two -step algorithms is the Faster R-CNN, which can be read in more detail in next section.

2.1.1 Faster R-CNN

The Faster R-CNN is an object detection system that consist of two main parts, an RPN and a Fast Regional Convolutional Neural Network (R-CNN) detector. The first part, RPN, which is a deep fully conventional network, generates proposed regions. It takes an image of any size as input and outputs a set of rectangular object proposals [1]. These proposals or regions are then feed into the second module, Fast R-CNN, which uses the regions for detection. In other words, the RPN, tells the Fast R-CNN where to look. Together, they create a single, unified network for object detection, see figure 2.1 [1].

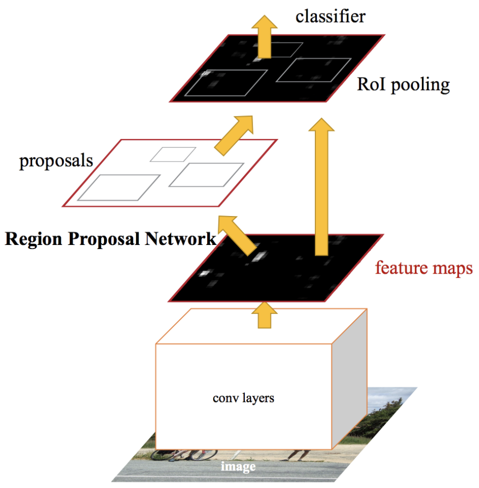


Figure 2.1 – Overall architecture of Faster R-CNN, consisting of a RPN and a classifier. Adapted from [1].

Fast R-CNN

The Fast R-CNN takes an entire image and a set of object proposals as input. It first processes the entire image with several convolutional and max pooling layers to create a convolutional feature map. Then, a Region of Interest (RoI) pooling layer extracts a feature vector from the feature map for each object proposal [2]. The feature vector are then sent into a sequence of fully connected layers that branches into two output layers. One of the outputs, is the output of a classifier, that produces softmax probability estimates over the object classes (classifies the object inside the bounding box). The other output, outputs 4 values for the coordinates of the predicted bounding boxes [2]. See figure 2.2 for the architecture of Fast R-CNN.

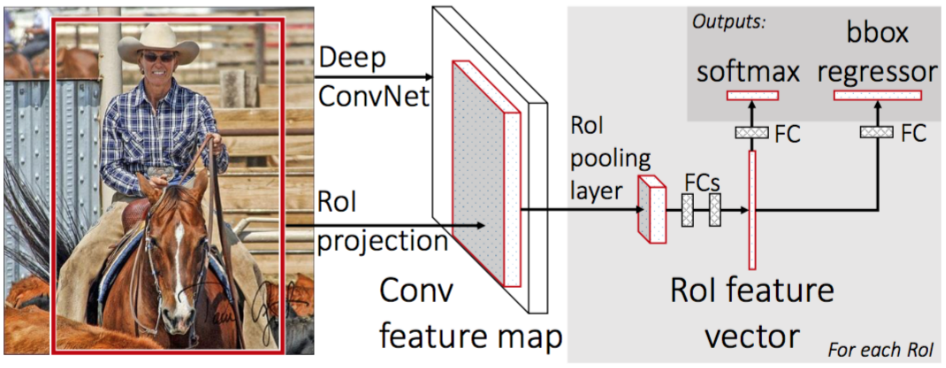


Figure 2.2 – Architecture of Fast R-CNN, adapted from [2].

2.2 Semantic segmentation

Semantic segmentation is the process to classify each pixel to a certain class by assigning it a certain value. It can be seen as image classification on pixel level. As with most areas in computer vision, before the breakthrough of deep learning, there were other methods that were used for semantic segmentation such as K-means clustering and random forest [12]. However, today the deep learning methods dominate and perform much better than the conventional techniques [13]. In the next sections below, the networks that will be used for segmentation in this project are presented.

2.2.1 SegNet

SegNet is a neural network for semantic segmentation and consists of a encoder and a decoder network, followed by a pixelwise classification layer. An encoder takes an input (for example an image) and downsamples it to a feature map, which holds the information, features, of the input. The decoder takes the feature map as input and upsamples it and tries to output the closest prediction compared to the ground truth data [3]. In SegNet, the encoder network consists of 13 convolutional layers, which corresponds to the 13 first convolutional layers in VGG16 (see section 2.3). The decoder network also consists of 13 convolutional layers, where its output is sent into multi-class softmax classifier layer that generates class probabilities for each pixel [3]. See figure 2.3 for the whole architecture of SegNet.

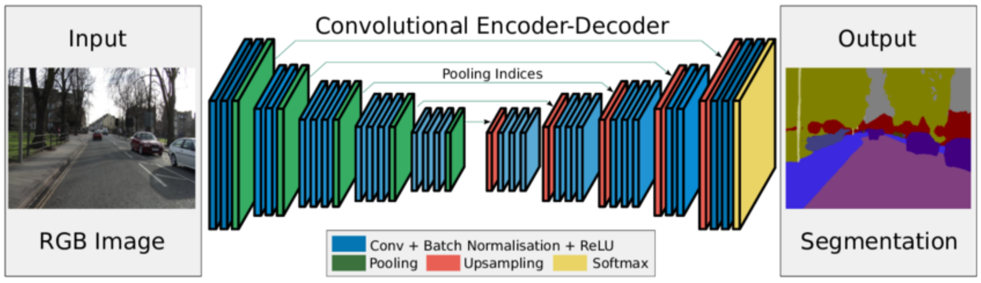


Figure 2.3 – Architecture of SegNet adapted from [3].

2.2.2 U-Net

As the SegNet, U-Net also consists of a encoder and decoder network. The encoder network, has the structure of a typical convolutional network. It consists of the repeated process of 3x3 convolutions followed by a Rectified Linear Unit (ReLu) and a 2x2 max pooling operation with stride 2 for downsampling [4]. On the other hand, the decoder network consists of the repeated application of an upsampling of the feature map followed by 2x2 convolution, a concatenation with the cropped feature map from the encoder and 3x3 convolutions with ReLu. In the end, there is a 1x1 convolution layer that maps the feature vector in to the desired number of classes [4]. See figure 2.4 for the whole architecture of U-Net.

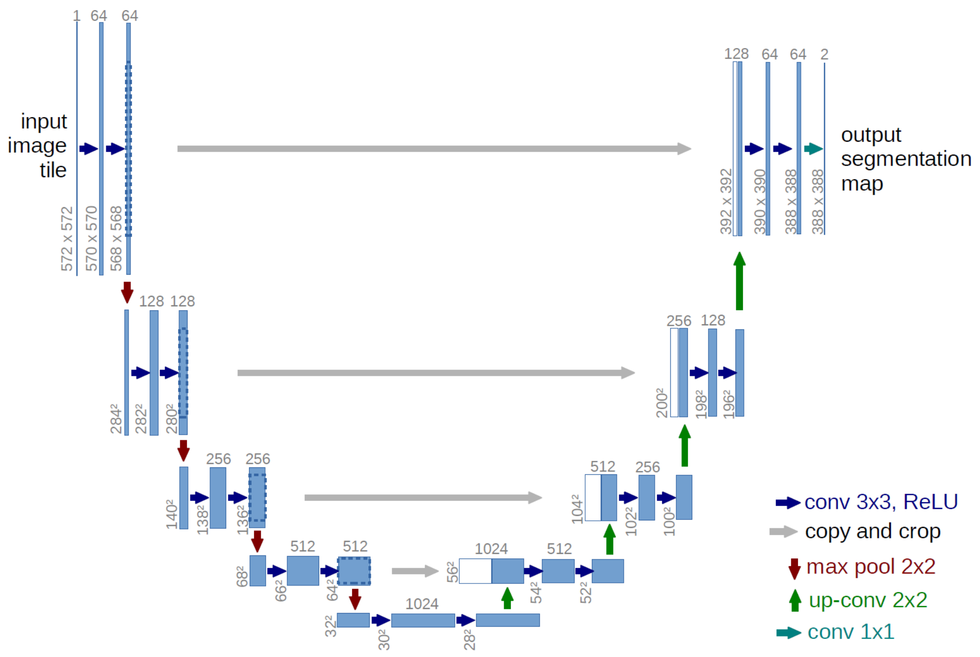


Figure 2.4 – Illustration of U-Net architecture adapted from [4]. A blue box corresponds to a multi-channel feature map, where the number of channels are denoted on top. The x-y size are denoted at the lower left edge of the box. The white boxes corresponds to copied feature maps.

2.2.3 PSPNet

PSPNet is a network that takes the global context of an image into account to predict the local level predictions. For example, in the paper about PSPNet [6], a comparison between PSPNet and Fully Convolutional Network (FCN), which is also an segmentation network, is done, see figure 2.5. In the figure one can see that FCN predicts the boat as a car, while PSPNet predicts the boat correctly [6]. The reason that FCN predicts the boat as a car, is that the boat look like a car. However, PSPNet takes the global context of the image into account, and therefore notice that there is water under the object and those can understand that the object is a boat and not a car [6].

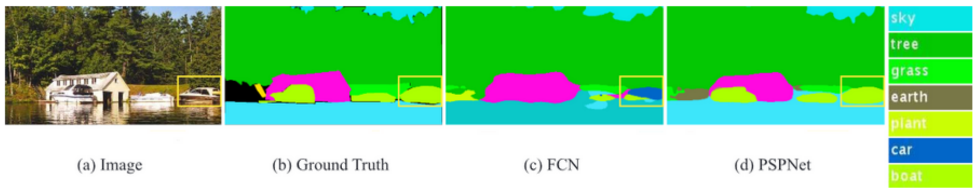


Figure 2.5 – Comparison between PSPNet and FCN. The images shows the significance of PSPNet's global scene parsing, which make it classify the boat correctly instead of a car as FCN. Adapted from [5].

PSPNet has an architecture shown in figure 2.6. Given an input image, it first downsamples it with a convolutional network such as Residual Neural Network (ResNet) and extracts the feature map. The feature map is then sent into the so called pyramid pooling module, which is the special part in the PSPNet architecture, since it is here the global context of the image is captured [6]. In the pyramid pooling module, the feature map is pooled at different scales and then passed to a convolutional layer. Thereafter, the resized feature map are then sent into an upsampler to rescale them as the size of the original feature map. The upsampled maps are then concatenated with the original feature map and sent to a convolutional layer where the final predictions are done. This method fuses features at different scales, thus aggregating the overall context [6]. However, there is a downside with the PSPNet model. In most cases, it uses a 8 x upsampling decoder where most parameters are not learnable, thus it results in blurry images and fails to capture high resolution information [5].

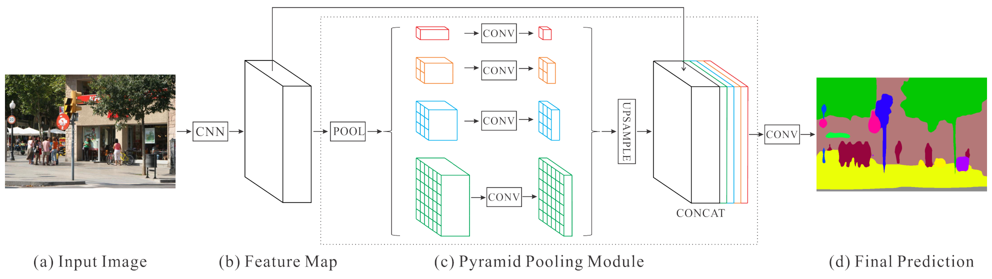


Figure 2.6 – Overview of PSPNet adapted from [6].

2.2.4 LinkNet

The architecture of LinkNet consists of 4 encoder and decoder blocks. The input layer consists of a convolutional layer followed by a maxpooling layer, while the output layer consists of two full convolutional layers and one convolutional layer. It uses batch normalization after each convolutional layer and ReLu non-linearity [7]. See figure 2.7 for the architecture of the whole network, the encoder and decoder blocks.

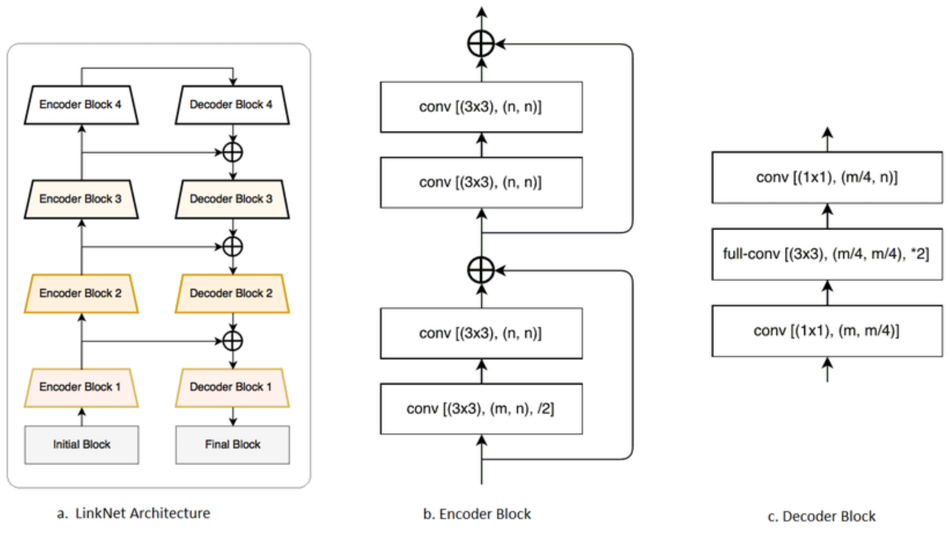


Figure 2.7 – Overview of LinkNet and it's encoder and decoder blocks. In the figure, conv means convolution and full-conv means full convolution. Further, /2 means a downsampling by a factor of 2, while *2 means an upsampling of factor of 2. Adapted from [7].

What makes LinkNet special compared to the prior networks within segmentation, is the link between the encoder and decoder blocks. The prior networks such a U-Net and SegNet, uses only the output of the last layer in the encoder as input to the decoder. This can be an issue when performing a lot of downsampling operations were spatial information can get lost, which is hard to recover when only using the output of the last encoder layer [7]. LinkNet takes this issue into account, by passing the output from each encoder block to the input of its corresponding decoder block. With this approach, the aim is to recover the lost information that can be used for the decoders. Furthermore, since the decoders share knowledge from the corresponding encoders, this results in fewer decoder parameters. This makes the network more efficient and suitable for real-time operations [7].

2.3 Transfer learning

Transfer learning is a technique within deep learning that involves using knowledge from earlier problems and apply to a new related problem. This usually means that one loads the weights from a pretrained network on a specific dataset. This makes learning much easier and the convergence is much faster [14].

Neural networks within object detection and segmentation often has a networks that extracts features from the input, which have similar structures to the convolutional networks that are used for image classification. Thus, the image classification networks can work as encoder for the object detection and semantic segmentation networks, they are then called backbones. Some example of backbones are VGG16, ResNet50 and EfficientNet B3. The pretrained weights that are loaded for those types of backbones usually comes from the weights trained on ImageNet, which is a enormous dataset consisted of 1.2 million images with 1000 classes [14].

The networks used in this project had the VGG16 as backbone and used the pretrained weights from ImageNet.

2.4 Related work

There are no direct related work to screw hole detection using neural networks that can be found online. The reason could be that the problem is very specific and there is not a lot of data on screw holes. Thus, using neural networks could be a tricky solution. However, there are some works that could still be related to the topic within this project, which are presented in the text below.

In [15], the authors investigate the problem of screw detection on various electronic devices for automated disassembly. A universal, generalizable and extendable screw detector is implemented and can be used in automated disassembly lines. The best state-of-the-art classifiers were selected and compared with the authors' architecture, which combines Hough transform for edge detection and two deep convolutional neural networks for classification. It is shown that the proposed method outperforms the existing methods, while still maintaining a high computational speed.

Another article that can be related to this project, is presented in [16]. Here the authors investigate the detection of rail defects. They propose a method based on recurrent neural networks. As an example, screw hole cracks are studied and tested for verification. According to the results, the authors conclude that their method can automatically complete the identification and classification of B-scan data and rail defect location.

Finally, in [17], coronal hole detection is investigated by using convolutional neural networks. The authors develop an automatic and reliable method that provides consistent full-disk segmentation maps over the full solar cycle and can perform in real time. The results show that the proposed model achieves an Intersection over Union (IoU) of 0.63 and from total of 261 coronal holes, 98.1 % were correctly detected.

Chapter 3

Methods

3.1 Object models

Since the topic and the dataset that is investigated during this project are very specific, there are no predefined models or datasets for screw hole detection when using neural networks for object detection and semantic segmentation. Thus, the models of the investigated objects and the generation of the datasets need to be created from scratch. In this project the 3D content-creation program Blender 2.92 (see Appendix A.1) is used to model the objects and to generate the datasets for the object detection and semantic segmentation algorithms.

3.1.1 Simple model

Two different object models are investigated during this project, a simple model and an advanced model (described in next subsection). The simple model of the object consists of a plain metallic cuboid with multiple holes on the top surface. The amount of holes varies during the experiments, but the cases that are investigated are with 5, 10, 50 and 100 holes. The dimensions of the cuboid are 1x2x0.25 m and the radius of the holes are 5 cm with 10 cm depth. See figure 3.1 for a case with 100 holes.

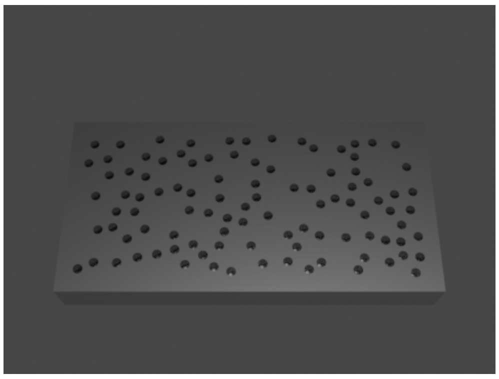
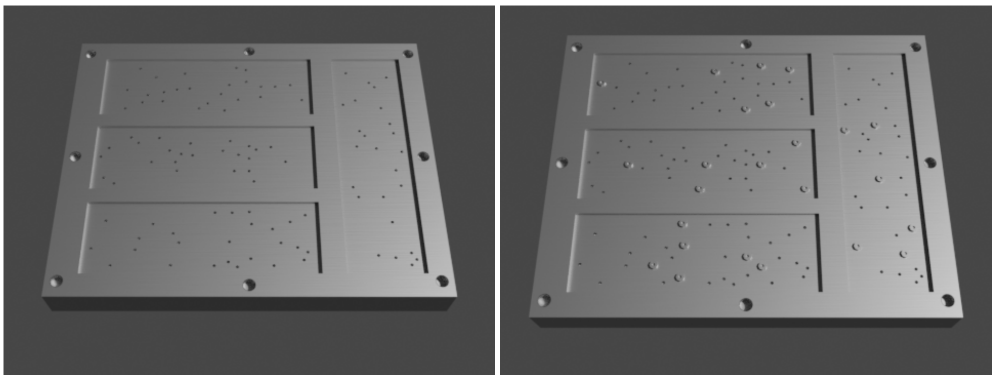


Figure 3.1 – Simple model with 100 holes.

One important notification when it comes to modelling the holes in this project is to model each hole as a separate object and not an actual hole. The reason behind this, is that the model will be used for data generation for object detection and semantic segmentation where the holes needs to be detected. Thus, every hole needs to be treated as an individual object so that it will be labelled as "hole" and not "cube" or any other class. This is done by first creating an actual hole and then putting an almost infinite thin hollow cylinder with a bottom that fits into the hole. In the simple model, the cylinder's radius is 4.5 cm and the height is 9 cm. By placing this cylinder into the holes, the holes will now be treated as objects when generating the data. To put it into more perspective, if not using the cylinder and just using plain holes in the cuboid, the holes will not be treated as individual objects since they are a part of the cuboid. Instead, the cuboid with its holes will be treated as a single object.

3.1.2 Advanced model

The advanced model tries to mimic or resemble the real radio and filter products at Ericsson. Thus, its design is inspired from a filter product. It is a metallic cuboid with dimensions 40x50x5 cm with various surface depths. The model contains 100 small holes (radius 2 mm and depth 4 mm) that represent the screw holes in the real product. In addition it also contains 8 bigger holes (radius 7 mm and depth 8 mm) at the edge of the product. Furthermore, the same model but with 24 screws is also designed. See figures 3.2 a) and b).



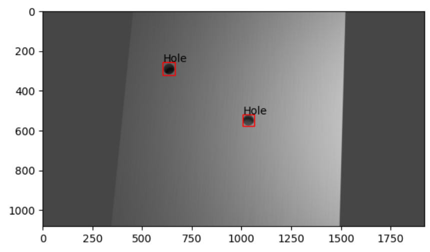
(a) Advanced model. **(b)** Advanced model with screws.

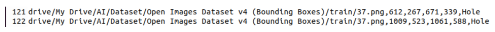
Figure 3.2 – Advanced models.

3.2 Dataset generation

3.2.1 Object detection

The dataset for the object detection algorithms generally consists of two parts, a set of images and an annotation file with the information of the coordinates of the bounding boxes and the class they are highlighting in the images. The object detection algorithm used in this project uses an annotation file in form of a txt file that consists of amount of rows corresponding to the amount of bounding boxes, one row for each bounding box. Further, the rows consist of a string divided into 6 parts, separated by a comma sign. The first part of the string is the path for the corresponding image. The second to the fifth part of the string corresponds to the coordinates of a bounding box in the form (x_1, y_1, x_2, y_2), where (x_1, y_1) is the coordinate of the top left corner of the box and (x_2, y_2) is the coordinate of the bottom right corner of the box. Finally, the last part is the class of the highlighted object in the image, which in this case is only "Hole". See figure 3.3 for an example image with bounding boxes and the corresponding rows in the annotation file.





```
drive/My Drive/AI/Dataset/Open Images Dataset v4 (Bounding Boxes)/train/37.png,612,267,671,339,Hole
drive/My Drive/AI/Dataset/Open Images Dataset v4 (Bounding Boxes)/train/37.png,1009,523,1061,588,Hole
```

(b) Corresponding rows in annotation file.

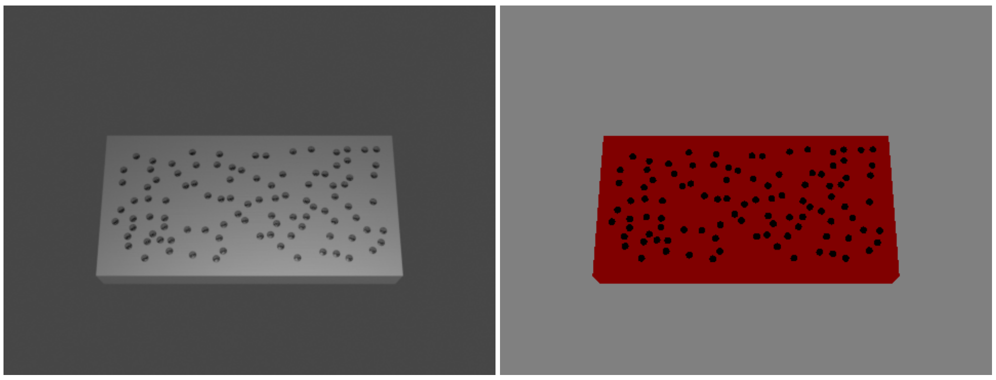
(a) Image with bounding boxes.

Figure 3.3 – Illustration of image and annotations for object detection.

3.2.2 Semantic segmentation

For semantic segmentation, the dataset also consists of two parts, a set of the actual images and a set of the corresponding masks, i.e. the segmented images. The pixels in the segmented images have a certain value that corresponds to a certain class. Generally, the segmented images are 8-bit grayscale images, while the actual images are RGB images.

In this project, the images are RGB images and the segmented images are 8-bit grayscale images with pixel values between 0, 1, 2, which correspond to the classes "background", "cube" and "hole", where "cube" means the actual object and "hole" means screw hole. See figure 3.4 of an example of image with its corresponding segmented image.



(a) Original image. **(b)** Segmented image.

Figure 3.4 – Example of image and it's segmented image.The segmented image is divided into three segments: grey pixels for background, red pixels for cube and black pixels for holes.

Note the segmented image 3.4b, is not the actual masked image used for the algorithms since the pixel values in the actual masked image are so low, it is hard to see something in the image.

3.3 Experimental design

The experiments done during this project consist of two parts. The first part compares the accuracy of hole detection between an object detection algorithm, in this case Faster R-CNN, and a semantic segmentation algorithm, which in this case is SegNet. Depending on the outcome of the first part, the second part of the experiments investigates a comparison in depth of the method that performed best in the first part of the experiments. The framework of the experiments is Google Colaboratory, which allows free Graphics Processing Unit (GPU) usage, see Appendix A.2. See the subsections below for further details.

3.3.1 Object detection vs semantic segmentation

In this part of the experiments, the accuracy of hole detection of an object detection algorithm and a semantic segmentation algorithm were compared. The algorithms that were investigated are Faster R-CNN and SegNet. The purpose of this part is to see which method, object detection or semantic segmentation, is more suitable for detecting holes in objects. Thus, the simple model was used during the experiments in this part. Depending of the outcome of this part, a comparison in depth between different algorithms in the same area was done in the next section.

As mentioned, the algorithms were trained and evaluated on a dataset that consisted of images of the simple model and the corresponding annotation depending on the algorithm. The dataset consisted of 1000 training samples and 250 test samples. The samples were rendered in Blender from a camera rotating 50 steps (7.2 degrees per step) around the object from heights 5-9 m above and distances 1-3 m from the center of the object.

The images taken for the object detection algorithm had the size 1920 x 1080 pixel and saved as png files. The reason for this large size was that it gave more precision when labeling the bounding boxes. Due to the large images, the render engine in blender was Eevee, which has a faster running time compared to Cycles.

On the other hand, when it comes to the dataset for the semantic segmentation, the images and masks had the size of 480 x 360 pixels and saved as png files as well. The reason for this size, is that the render engine now needs to be Cycles because segmented images was generated. Thus, smaller images were needed to speed up the running time.

Different cases of amount of holes were investigated for the two algorithms. For the object detection algorithm the cases consisted of smaller amount of holes compared to the segmentation algorithm due to the hypothesis that the segmentation performs better in this scenario. The basis for this hypothesis lies in the fact that segmentation methods classify each pixel in an image, while object detectors need to localize and then classify the object in the bounding box. This makes segmentation methods in general more accurate, which explains the reasoning behind the mentioned hypothesis.

3.3.2 Semantic segmentation algorithms

In this experiment, a comparison between different semantic segmentation algorithms was done. The algorithms were SegNet, U-Net, PSPNet and LinkNet. The purpose of this part was to see which algorithm performed best for this problem and how well segmentation methods can perform for screw hole detection. The networks were trained and evaluated on a dataset of the advanced model consisting of 1000 training samples and 250 test samples. The training samples consisted of 500 images taken of the product and 500 images generated trough data augmentation by using the 500 images taken of the product as base. The data augmentation that was made were random brightness, horizontal flip, sharpness, contrast, blur and Gaussian noise. The samples were rendered in Blender from a camera rotating 50 steps around the object from heights 0.5 - 0.8 m above and distances 0.1 - 0.4 m from the center of the object. The render engine was Cycles and the images had the size of 480 x 360 pixels and saved as png files.

Prior to the comparison of the networks, a hyper parameter tuning was done by using k-fold cross validation. When the hyper parameters for each model were tuned, then the models were compared based on their performance on the test data.

In addition, the models were also trained and evaluated on a dataset (with the same settings as the one above) of the advanced model with screws with the parameter settings obtained from the k-fold cross validation.

Chapter 4

Results

4.1 Object detection vs semantic segmentation

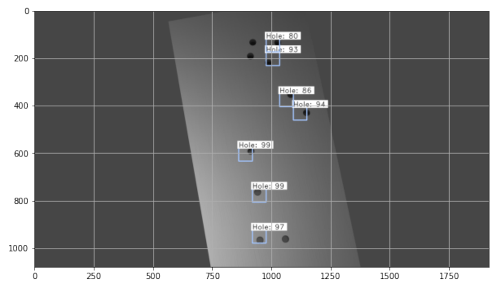
The algorithms were trained and evaluated on different cases of amount of holes. Both used VGG16 as backbone. SegNet and the classifiers in Faster R-CNN used cross entropy loss for classifying the objects or pixels in the image, while the regression outputs in Faster R-CNN used smooth L1 loss. See table 4.1 for the other parameter settings during training for each algorithm for this experiment.

Table 4.1 – Parameter settings for the algorithms in the following experiment.

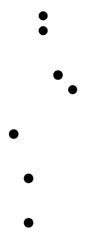
Algorithm	Epochs	Batch size	Optimizers	Learning rates
Faster R-CNN	9	1	ADAM	1e-5
SegNet	16	8	ADAM	1e-4

The metrics used for measuring the accuracy of the algorithms under this experiment was Mean Squared Error (MSE) of screw hole positions, average percentage of missed holes in an image and the Mean Intersection over Union (MIoU) of segmented images. The error in MSE in Faster R-CNN's case was calculated by first taking the difference between the centers of the predicted bounding boxes and the centers in the corresponding ground truth bounding boxes. The mean was then taken of all the differences of centers in an image and then the mean of all images. For SegNet, the center of hole was calculated by thresholding the segmented image such that the holes had pixel values 255 and the background and cube had 0 as pixel value. Then a simple blob detector from OpenCV was applied from which the centers could be extracted from the output. The same procedure as the Faster R-CNN's case was then applied by taking the mean of all differences in an image and then taking the mean for all images.

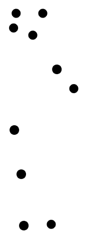
The metric "Missed holes" is a measurement of the mean percentage of missed holes for every image. The MIoU is a typical metric for semantic segmentation, where the IoU can be seen as the ratio between the set of intersection and set of union between the predicted and ground truth image. One could then ask, how the IoU was calculated from the output of Faster R-CNN since it is not a segmentation algorithm? The solution was to create an artificial segmented image from the bounding boxes. The centers of the circles in the segmented image was calculated from the centers of the bounding boxes and the radius was set equal to the min(h, w) of the bounding box, where h is the height and w is the width of the bounding box. See figure 4.1 for an example of artificial segmented image.



(a) Image with predicted bounding boxes.



(b) Corresponding artificial predicted segmented image.



(c) Corresponding artificial ground truth segmented image.

Figure 4.1 – Example of artificial segmented image and corresponding output image from object detection algorithm.

See table 4.2 for the results of the algorithms on the different cases of amount of holes in the simple model. Note that the amount of holes differ between the algorithms. The reason for this, is that the hypothesis was that it would be difficult for the object detection method to perform well to detect a lot of small holes in an object. Thus, when starting easy for Faster R-CNN, was enough to show that it did not have the same potential as SegNet, which saved a lot of time and justified the hypothesis mentioned in the previous sentence.

Table 4.2 – Results from the experiments of FRCNN and SegNet.

Algorithm	Amount of holes	MSE	Missed holes	MIoU
FRCNN	5	21.5	9.6 %	0.59
FRCNN	10	20.2	38.6 %	0.57
SegNet	50	0.82	3.1 %	0.87
SegNet	100	0.87	5.1 %	0.86

4.2 Semantic Segmentation Algorithms

In this part of the experiments, SegNet, U-Net, PSPNet and LinkNet were compared. First, a hyper parameter tuning of batch size, number of epochs, optimizer and learning rate was done. The k-fold cross validation technique was used during the hyperparameter tuning with k = 3. The metric that was used during this part was the MIoU. To speed up the process, hyper parameter tuning for batch size and epochs was first applied. Then the optimal combination of batch size and number of epochs for each network was extracted and used in the hyper parameter tuning for optimizer and learning rate. For each combination, the standard deviation and mean of the performance (MIoU) was calculated. See the figures below for the performances during the hyperparameter tuning for the different networks.

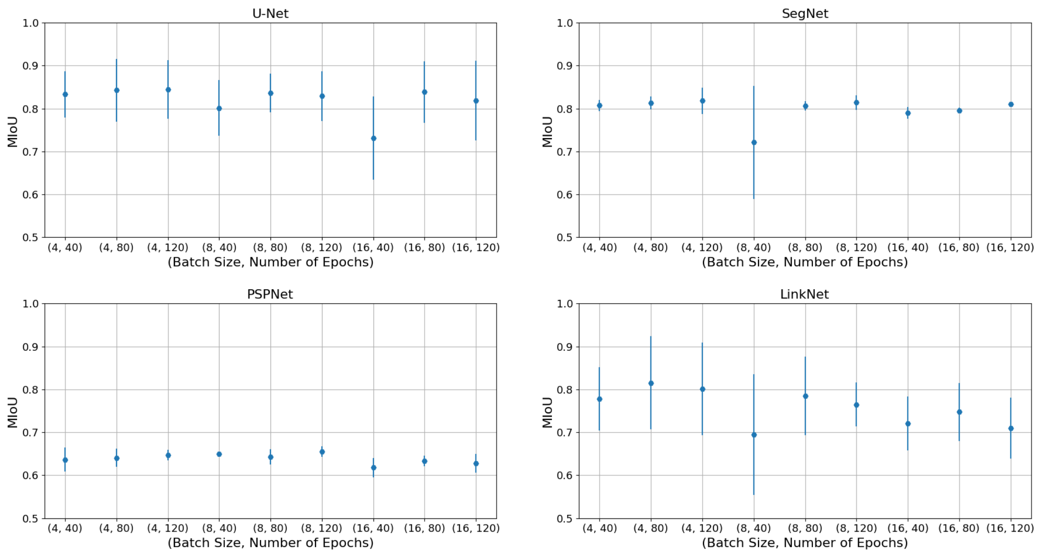


Figure 4.2 – Mean scores (denoted by the blue dots) and the standard deviation (denoted by the ranges or whiskers) for different combinations of batch sizes and total number of epochs for different networks.

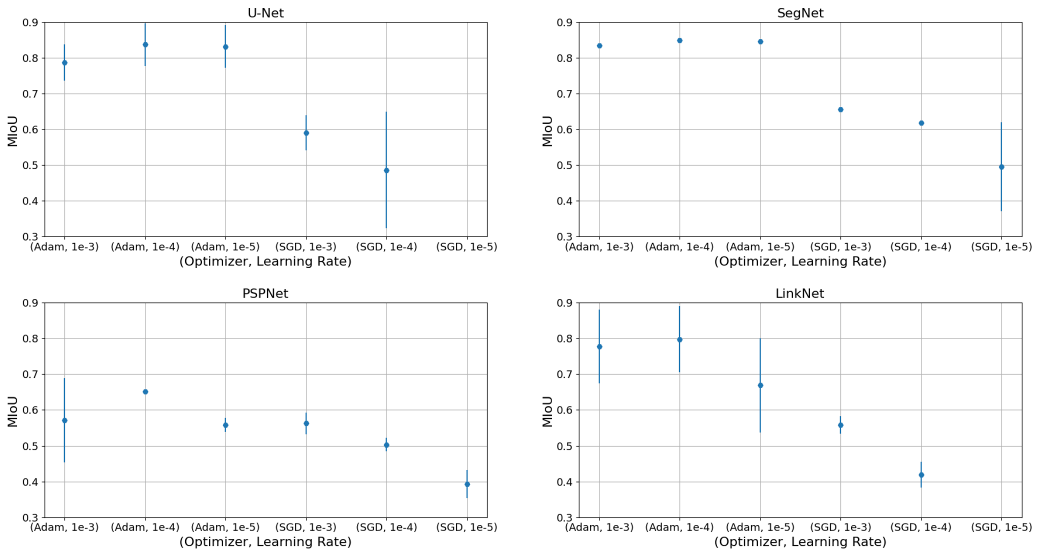


Figure 4.3 – Mean scores (denoted by the blue dots) and the standard deviation (denoted by the ranges or whiskers) for different combinations of optimizers and learning rates for different networks.

When the optimal hyperparameter settings was decided for each network, then the algorithms were trained with these hyperparameter settings and tested on the test data. They were also trained and tested on the advanced data with screws with the hyperparameters obtained previously. See the table below for the algorithms' performances under the two cases.

Table 4.3 – Final results of the semantic segmentation networks with their optimal parameter settings obtained from the hyperparameter tuning. The column "Without screws" stands for the advanced model without screws, while the column "With screws" stands for the advanced model with screws. The learning rate is denoted by η.

Models	MIoU		Batch size	Epochs	Optimizer	η
	Without screws	With screws				
U-Net	0.924	0.890	4	120	ADAM	1e-4
PSPNet	0.688	0.686	8	120	ADAM	1e-4
SegNet	0.782	0.777	4	120	ADAM	1e-4
LinkNet	0.923	0.918	4	80	ADAM	1e-4

Chapter 5

Discussion

5.1 Object detection vs Semantic Segmentation

The purpose of this experiment was to investigate which type of method, object detection or semantic segmentation, is most appropriate to detect holes in an object. From the results in table 4.2, one can see that SegNet clearly outperforms Faster R-CNN in the investigated cases. Something to take into account besides SegNet's performance, is that the cases that it is tested on are much more difficult and advanced compared to the Faster R-CNN's case and it still manages to outperform Faster R-CNN. One can also see in table 4.2 that the results of MSE, percentage of missed holes and MIoU goes hand in hand, which make the metrics more valid and logical. When the MSE is high, the percentage of missed holes is also high, while the MIoU is low. For example for Faster R-CNN with 5 amount of holes, the MSE is 21.5, 9.6 % of missed holes and the MIoU is 0.59, while for SegNet with 50 holes, the MSE is 0.82, 3.1 % of missed holes and the MIoU is 0.87.

Furthermore, the results in this comparison are also logical and they support the hypothesis that semantic segmentation algorithms are more suitable for detecting a lot of holes in an object compared to object detection algorithms. The reason could be that segmentation algorithms classify each pixel to a certain class, while the object detection algorithm only tries to find an object in the image and classify it. This naturally gives segmentation algorithms higher accuracy, which is very important in those kinds of tasks when one want to find the exact positions of certain objects in an image.

One question that one could have in mind is why other object detection algorithms were not also investigated in this experiment. The answer to this question is simply that Faster R-CNN is in general the most accurate object detection algorithm of the popular object detection algorithms such as YOLO and SSD, see Appendix B.1. Thus, with the time constraint in this project and that Faster R-CNN is in general most accurate, it makes it sufficient to test only Faster R-CNN. In addition, the difference in accuracy between Faster R-CNN and SegNet were so large, that even if there could be a more accurate object detection algorithm than Faster R-CNN, it would not probably be enough to conquer with the segmentation methods.

5.2 Semantic Segmentation Algorithms

The purpose of this part of the experiments, was to compare the neural networks for segmentation on an advanced model of an industrial object. First, a hyperparameter tuning was done, where the k-fold cross validation was also used. In figure 4.2, one can see the networks' performances for different batch sizes and number of epochs. To decide the optimal combination, the mean value is prioritized, but the standard deviation is also taken into account. For the combination that has the highest mean score, it is also important to check that the standard deviation is not too large. For U-Net one can see that batch size equal to 4 and 120 epochs yields the optimal results since it has the highest mean and its upper limit is one of the higher ones, while its lower limit is the lowest one. By following the same principle, for PSPNet, the optimal value was (8, 120), for LinkNet it was (4, 80) and for SegNet it was (4, 120). The same process was done for finding the optimal combination for optimizer and learning rate, which turned to be Adaptive Moment Estimation (ADAM) optimizer with learning rate 1e-4 for all networks. One clear conclusion that one can draw from figure 4.3, is that the networks perform worse with the Stochastic Gradient Descent (SGD) and that the ADAM optimizer is the one to go with.

If one looks at the results on table 4.3, one can see that U-Net achieves the highest score on the advanced model, while LinkNet has the highest score on the model with screws. This tells also about the robustness of the algorithms. For example, U-Net has a larger drop when going from the advanced model to the advanced model with screws compared to the other networks, which means that it lacks robustness. However, both U-Net and LinkNet have MIoU around 0.9, which indicates that these algorithms have potential to be used and developed for real applications on more advanced product models. One can also see that PSPNet has a much lower MIoU than the other networks. The reason could be its decoder structure that causes blurry images and fails to capture high resolution information, which is crucial in this dataset.

Furthermore, one can see in table 4.3, that the MIoU drops a bit from the case without screws to the case with screws. This seems to be logical since the case with screws is of course more difficult to detect, because there are more parameters or disturbances in the image. In addition, the results in table 4.3 are valid from a statistical point of view. The networks are tested with the parameter settings obtained from the k-fold cross validation, in which the performances were almost similar (see figure 4.2 and 4.3). Thus, it makes the results in table 4.3 statistical valid.

Chapter 6

Conclusions and Future work

6.1 Conclusions

The research question in this project was to investigate how object detection and/or semantic segmentation algorithms perform on a dataset consisting of an industrial object with multiple small screw holes.

The conclusion that one can draw from the analysis in the first part of the discussion, is that object detection algorithms perform poorly to detect multiple small holes, while the semantic segmentation method perform very good. Thus, semantic segmentation methods are more appropriate to detect screw holes in an industrial object.

From the second part of the discussion, one can draw the conclusion that U-Net and LinkNet are the algorithms that perform best to detect screw holes. In addition, their performance is good enough to show that there is potential to use them in reality for more advanced models. However, the algorithms are still not ready to be implemented in reality, since they do not have 99 % accuracy, but they are close with around 90 % accuracy.

6.2 Limitations

The limitations within this project were the designs of the objects and the amount of investigated algorithms. The main reason for these limitations was the time constraint of the project. If one had more time, one could test other networks such as DeepLab, do further hyperparameter tuning, more object designs, or do some more advanced experiments. In addition, the generation of the dataset was very time consuming (500 images took about 4 -5 hours), especially when doing it multiple times for different cases and experiments. A GPU that could connect to Blender, would be useful in this situation.

6.3 Future work

Due to the breadth of the problem, only some of the initial goals have been met. In these section, the focus will lie on some of the remaining issues that should be addressed in future work.

Next things to be done are to continue to research and develop the algorithms' performances within this subject. This can be done by training the algorithms with data of more advanced models, do some more hyperparameter tuning of for example optimizers or cost functions and investigate other possible algorithms such as DeepLab. By doing this, one can push the algorithms' performances toward 99 % accuracy and possibly implement them on real applications.

6.3.1 What has been left undone?

One of the things that have been left undone is the transfer of the networks to real-world scenarios, which was one of the company's desires. Due to the time constraint of this master's thesis and the fact that the algorithms' accuracies were not enough to be tested on images of real products, which have more complicated designs than the models used in this project, this could not be accomplished. Still, the algorithms (U-Net and LinkNet) showed a good performance of around 90 % accuracy on an advanced model with screws, which shows that there is potential of further development for implementation on real products.

6.4 Reflections

In this section, reflections of the project will be done. Questions like positive effects and drawbacks, what insights I gained and what would I done differently will be discussed. In addition, the economic, social, environmental and ethical aspects of the project will also be mentioned.

The positive effects of this project was that I learned and gained a lot of technical, practical and social aspects. I learned and deepened my knowledge within AI and neural networks. In addition, I also got a lot of practical experience as an engineer and got an insight of how it is to work within a project in the working life. This also improved my social and communication skills and how to manage such challenging projects. The final positive effect from this project, was the results that showed that there is potential of neural networks for detecting screw holes in industrial objects. The main drawback that I can see with this project, was that it was very time consuming and challenging (which can also be interpreted as a positive effect). Furthermore, it was also a bit stressful since I had to satisfy both KTH and Ericsson at the same time.

There are two kinds of insights that I have gained through this project, technical insights and project managing insights. The technical insights that I have gained through this project, is further knowledge and experience within neural networks and computer vision. I also gained insights in 3D-object modelling and data generation for object detection and semantic segmentation.

Furthermore, I also gained insights how one can manage and work within such challenging and advanced project. I had to design my own research questions, solutions and ideas to complete this project, while still satisfying both company's (Ericsson) and university's (KTH) requirements. It required a lot of communication and presentation skills since I had to present and sell my ideas for Ericsson every third week. It also gave me practical experience as an AI engineer and how one can solve problems that occurred during the project. For example, in the beginning I had some problem of finding a GPU, but I eventually solved it by using Google Colaboratory.

If I had to do this project again, there are two things I would have done differently or think about. The first thing, is to have a prepared research question when starting the master's thesis. This time, I didn't have a prepared valid research question when starting the project, which cost me around 2 months to come up with one. The other thing that I would think about, is to do a more proper research about the needed resources such as GPU. During this project, I was stuck for 3 weeks, trying to find a GPU that I could use. If I had knew in the start of the project that I needed a GPU, this would go much faster.

6.4.1 Economic, social, environmental and ethical aspects of the work

The main application of this project was to detect screw holes in industrial objects for automated assembly and disassembly. Thus it is a key in the system of automated assembly and disassembly. This benefits the companies' economics, since they do not need to pay humans to assemble or disassemble their products. Furthermore, it can also provide time efficiency, since robots can work non-stop. In addition, it also provides safety and health in the long term for humans that use to assemble or disassemble multiple of hundreds of screws. However, one might ask what happens with the humans, do they lose their job? My answer, is that I don’t think they will lose their job, but have other tasks that are more comfortable and less strenuous such as machine operator.

References

[1] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017.

[2] R. Girshick, "Fast r-cnn," in *2015 IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2015. ISBN 1467383910. ISSN 2380-7504 pp. 1440–1448.

[3] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

[4] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," 2015.

[5] How pspnet works? [Online]. Available: https://developers.arcgis.com/python/guide/how-pspnet-works/

[6] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," 2016.

[7] A. Chaurasia and E. Culurciello, "Linknet: Exploiting encoder representations for efficient semantic segmentation," 2017.

[8] A. Sachan. Zero to hero: Guide to object detection using deep learning: Faster r-cnn,yolo,ssd. [Online]. Available: https://cv-tricks.com/object-detection/faster-r-cnn-yolo-ssd/

[9] P. Ganesh. (2019, Aug.) Object detection : Simplified. [Online]. Available: https://towardsdatascience.com/object-detection-simplified-e07aa3830954

[10] J. Brownlee. (2019, May) A gentle introduction to object recognition with deep learning. 151 Calle de San Francisco, Suite 200 – PMB 5072, San Juan, PR 00901, USA. [Online]. Available: https://machinelearningmastery.com/object-recognition-with-deep-learning/

[11] Object detection guide. [Online]. Available: https://www.fritz.ai/object-detection/

[12] A. Matcha. (2021, May) A 2021 guide to semantic segmentation. 2261 Market Street, San Francisco, CA 94114, USA. [Online]. Available: https://nanonets.com/blog/semantic-image-segmentation-2020/

[13] D. Mwiti. (2019, Jul.) A 2019 guide to semantic segmentation. [Online]. Available: https://heartbeat.fritz.ai/a-2019-guide-to-semantic-segmentation-ca8242f5a7fc

[14] M. Wosinski. (2020, Oct.) Transfer learning in a nutshell. [Online]. Available: https://medium.com/yosh-ai/transfer-learning-in-a-nutshell-71e1dd0b59da

[15] E. Yildiz and F. Wörgötter, "Dcnn-based screw detection for automated disassembly processes," in *2019 15th International Conference on Signal-Image Technology Internet-Based Systems (SITIS)*, 2019. doi: 10.1109/SITIS.2019.00040 pp. 187–192.

[16] Q. Xu, Q. Zhao, G. Yu, L. Wang, and T. Shen, "Rail defect detection method based on recurrent neural network," in *2020 39th Chinese Control Conference (CCC)*, 2020. doi: 10.23919/CCC50068.2020.9188823 pp. 6486–6490.

[17] R. Jarolim, A. M. Veronig, S. Hofmeister, S. G. Heinemann, M. Temmer, T. Podladchikova, and K. Dissauer, "Multi-channel coronal hole detection with convolutional neural networks," *Astronomy Astrophysics*, vol. 652, p. A13, Aug 2021. doi: 10.1051/0004-6361/202140640. [Online]. Available: http://dx.doi.org/10.1051/0004-6361/202140640

[18] S. Srivastava, A. V. Divekar, C. Anilkumar, I. Naik, V. Kulkarni, and V. Pattabiraman, "Comparative analysis of deep learning image detection algorithms," *Journal of big data*, vol. 8, no. 1, pp. 1–27, 2021.

[19] U. Alganci, M. Soydas, and E. Sertel, "Comparative research on deep learning approaches for airplane detection from very high-resolution

satellite images," *Remote sensing (Basel, Switzerland)*, vol. 12, no. 3, p. 458, 2020.

[20] G. Hyams and D. Malowany. (2020, Mar.) The battle of speed vs. accuracy: Single-shot vs two-shot detection meta-architecture. 1200 McGill College Ave 7th Floor, Montréal QC H3B 4G7, Canada. [Online]. Available: https://clear.ml/blog/the-battle-of-speed-accuracy-single-shot-vs-two-shot-detection/

[21] X. Xu. (2018, Aug.) Yolo vs faster rcnn. [Online]. Available: https://everitt257.github.io/post/2018/08/10/object_detection.html

Appendix A

Software Environments

A.1 Blender

Blender is a free open source 3D creation program that supports the complete 3D-pipeline. These includes modelling, rigging, animation, simulation, rendering, compositing, motion tracking, video editing and game creation. Blender's API for python scripting allow users to customize their own application and tools. It also, gives opportunity for more flexible modelling and rendering. In addition, with Blender's high quality of rendering images and python scripting, it is possible to generate synthetic datasets for object detection and image segmentation.

A.2 Google Colaboratory

Google Colaboratory (or Google Colab for short), allows for python scripting and execution in your browser with zero configuration required, free access to GPUs and easy sharing. Although, Google Colab allows for free access to GPUs, there is some limitations. One can only use a GPU for maximum 12 hours per day and the GPU that is used is a 12GB NVIDIA Tesla K80 GPU. However, if one want to increase the resources, one can buy Google Colab Pro, which costs 9.99 dollar per month. It allows for up to 24 hours of GPU usage per day and offer faster GPUs like NVIDIA Tesla T4 and P100.

Appendix B

Neural Networks

B.1 Comparison between object detection algorithms

In this section, a comparison between the most popular object detection algorithms, Faster R-CNN, YOLO and SSD, will be done.

In article [18], a comparative analysis of Faster R-CNN, YOLO and SSD is done on the Microsoft COCO dataset. The comparison was mainly based on the algorithms' performances in terms of accuracy and computational time. Their conclusion was that YOLO is the fastest of the three, with SSD following close and Faster R-CNN finishing in third place. However, when it comes to accuracy, Faster R-CNN is the one that wins the race. In addition, they also conclude that YOLO and SSD have difficulties with detecting small objects, something that Faster R-CNN has no problem with. They also suggest to go with Faster R-CNN if one has a relatively small dataset where no real time computation is important. On the other hand, they suggest to use YOLO when one wants to detect objects in a live video.

In [19], the authors do a comparative research between Faster R-CNN, YOLO and SSD for airplane detection from very high resolution satellite images. In their conclusions, they state that the best results in terms of accuracy was achieved with Faster R-CNN, while YOLO achieved the fastest convergence of them all.

The online websites [20], [21] and [8] compares Faster R-CNN with YOLO and SSD in general and states as the articles above, that Faster R-CNN has in general higher accuracy, while the other two have faster computational time, see the figure below.

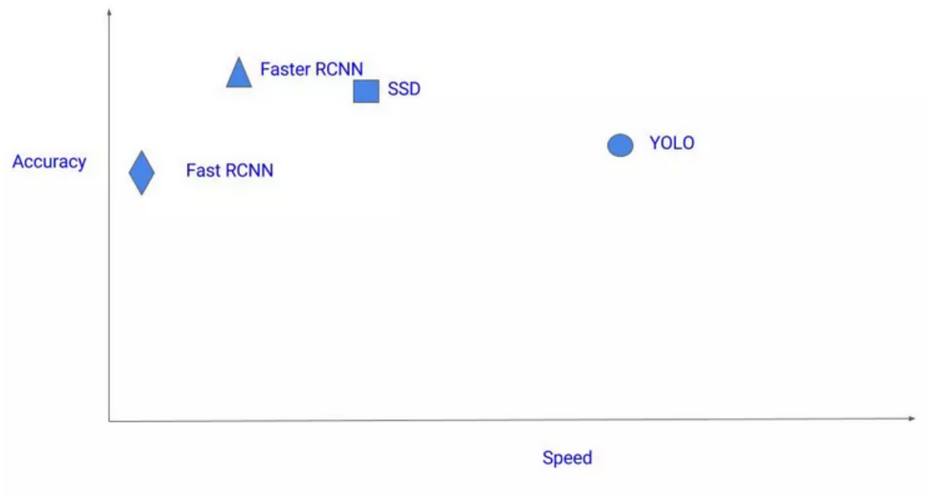


Figure B.1 – Relation between different object detection algorithms in terms of accuracy and speed. Adapted from [8].

In [8], they also look at the algorithms' performances in different object sizes and Faster R-CNN is the one with the highest accuracy in all cases. See the figure below.

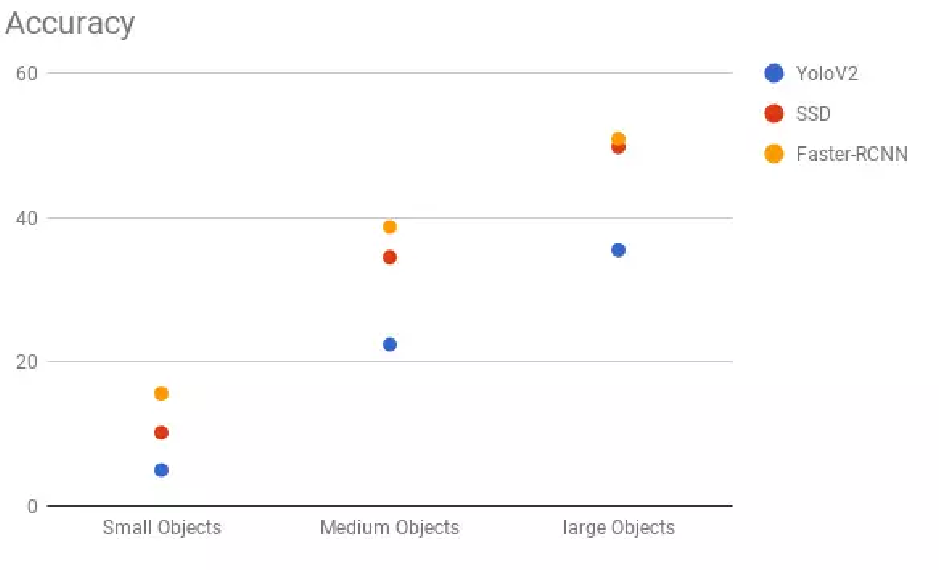


Figure B.2 – Comparison between YOLO, SSD and Faster R-CNN in accuracy of different sizes of objects. Adapted from [8].

TRITA -EECS-EX-2022:6