Object Detection Using Convolutional Neural Network Trained on Synthetic Images

Margareta Vi
Abstract

Training data is the bottleneck for training Convolutional Neural Networks. A larger dataset gives better accuracy though also needs longer training time. It is shown by finetuning neural networks on synthetic rendered images, that the mean average precision increases. This method was applied to two different datasets with five distinctive objects in each. The first dataset consisted of random objects with different geometric shapes. The second dataset contained objects used to assemble IKEA furniture. The neural network with the best performance, trained on 5400 images, achieved a mean average precision of 0.81 on a test which was a sample of a video sequence. Analysis of the impact of the factors dataset size, batch size, and numbers of epochs used in training and different network architectures were done. Using synthetic images to train CNN’s is a promising path to take for object detection where access to large amount of annotated image data is hard to come by.
Acknowledgments

I would like to thank my supervisor at my company Alexander Poole, for always being helpful and coming with interesting ideas. I would also like to thank my supervisor at the university, Mikael Persson for helping me with the report and my examiner Michael Felsberg.

Additionally, I would like to give my thanks to IKEA for providing the CAD models. Lastly, I would like to thank my family and boyfriend for supporting me through all the hard times.

Linköping, November 2018
Margareta Vi
### Contents

**Notation**

<table>
<thead>
<tr>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Introduction</td>
</tr>
<tr>
<td>1.1 Neural network/convolutional neural network in brief</td>
</tr>
<tr>
<td>1.2 Problem Formulation</td>
</tr>
<tr>
<td>1.2.1 Limitation</td>
</tr>
<tr>
<td>1.3 Thesis Outline</td>
</tr>
<tr>
<td>2 Related work</td>
</tr>
<tr>
<td>2.1 Using synthetic data</td>
</tr>
<tr>
<td>2.2 Finetuning</td>
</tr>
<tr>
<td>2.3 Object classification</td>
</tr>
<tr>
<td>2.4 Object detection</td>
</tr>
<tr>
<td>2.5 Summary: Related Work</td>
</tr>
<tr>
<td>3 Method and Experiments</td>
</tr>
<tr>
<td>3.1 Generating the Datasets</td>
</tr>
<tr>
<td>3.1.1 Rendering Images</td>
</tr>
<tr>
<td>3.1.2 Video Recording</td>
</tr>
<tr>
<td>3.1.3 Creation of Ground Truth Data</td>
</tr>
<tr>
<td>3.2 Dataset Distribution and Network Pairings</td>
</tr>
<tr>
<td>3.3 Evaluation Metrics</td>
</tr>
<tr>
<td>3.3.1 Losses</td>
</tr>
<tr>
<td>3.3.2 PASCAL Mean Average Precision</td>
</tr>
<tr>
<td>3.4 Parameters to tune</td>
</tr>
<tr>
<td>3.5 Experiments</td>
</tr>
<tr>
<td>4 Results</td>
</tr>
<tr>
<td>4.1 Testing Different Network Configuration</td>
</tr>
<tr>
<td>4.1.1 Faster R-CNN and Inception</td>
</tr>
<tr>
<td>4.1.2 SSD and Inception</td>
</tr>
<tr>
<td>4.1.3 SSD and MobileNet</td>
</tr>
<tr>
<td>4.1.4 Summary: Single-Shot Multibox Detector</td>
</tr>
</tbody>
</table>
## Contents

4.1.5 Summary: Different Network Architecture And Batch Sizes 43  
4.2 Epochs Versus Dataset Size ................................................. 46  
4.3 Testing on real images ..................................................... 49  
4.4 Automatic and Manual Annotations ....................................... 50  

5 Discussion 57  
5.1 Networks ........................................................................... 57  
5.1.1 Single-Shot Multibox Detector ......................................... 57  
5.1.2 Faster R-CNN and Inception ............................................. 57  
5.2 Epochs versus Batches .......................................................... 58  
5.3 Testing On Real Images, Video Sequence ................................. 58  
5.4 Annotation: Manual vs Automatic ......................................... 58  

6 Conclusions And Future Work 61  
6.1 Conclusions ......................................................................... 61  
6.2 Future Work ......................................................................... 61  

A Datasets 65  

Bibliography 67
## Notation

### Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAD</td>
<td>Computer Aided Design</td>
</tr>
<tr>
<td>ILSVRC</td>
<td>ImageNet Large Scale Visual Recognition Challenge</td>
</tr>
<tr>
<td>CNN</td>
<td>Convolutional Neural Network</td>
</tr>
<tr>
<td>SVM</td>
<td>Support Vector Machine</td>
</tr>
<tr>
<td>MAP</td>
<td>Mean Average Precision</td>
</tr>
<tr>
<td>RCNN</td>
<td>Regional Convolutional Neural Network</td>
</tr>
<tr>
<td>SSD</td>
<td>Single Shot Multibox Detector</td>
</tr>
<tr>
<td>IOU</td>
<td>Intersection over Union</td>
</tr>
</tbody>
</table>
Everyone has at least once compiled furniture from IKEA. They are relatively cheap and come in flat packages, the key thing is you need to build them yourself with the help of a booklet. The compilation starts by laying out all the pieces in front of you. The large pieces are easy to recognize, though the screws and plugs might cause problems. The items are small and have the same look, which increases the difficulty to distinguish them from each other. Therefore when building IKEA furniture, one could say the hardest part is to find the correct piece to use.

We humans solve this problem by first doing a coarse filtering by localizing objects with the same form as the one we seek. The next step is a fine search through the remaining items, with a specific image of the component in mind. The problem boils down to having a scene full of items and we want to localize a specific object i.e. object detection. Object detection is finding where an object is and what type of object it is.

Manual feature matching is costly, therefore it is desirable to computerize the task of finding such features. Two common techniques within object detection are using handcrafted features combined with machine learning approaches such as Support Vector Machine, and using artificial neural networks. Handcrafted features are image properties derived using different algorithms. These features include among other SIFT (Scale Invariant Feature Transform), SURF (Speeded up robust feature), and BRIEF (Binary Robust Independent Elementary Features). An artificial neural network on the other hand is a complicated model inspired by how the human brain is structured. The difference between handcrafted features and artificial neural networks is that the neural network try to learn patterns.
Neural networks have great potential to solve problems which involve detection of patterns or trends. Scientists have created neural networks which can solve tasks such as digit or word recognition, image classifications, face recognition, and object detection to name a few. Examples of neural networks which solve such tasks are Watson and AlphaGo. Watson played and won against Jeopardy champions [12] and AlphaGo was the first computer program which won against a Go world champion player [11].

1.1 Neural network/convolutional neural network in brief

In a human brain neurons are connected to each other via synapses, while in artificial neural network neurons are functions and synapses are weights. The model is shown in Figure 1.0.

*Figure 1.0: A biological neuron and its mathematical representation. Image acquired from [22].*
From here on artificial neural networks will be referred as neural networks (NNs). A neural network consists of many different layers: the input layer, the hidden layer, and the output layer. The input layer contains of images, and the output layer is the result of the task that the NN is trying to solve i.e. object detection. The hidden layer consist of many different layers. In each layer different mathematical operations occur, such as pooling, normalization, and convolution.

![Neural Network Diagram](image)

**Figure 1.1: Neural network**

A specific type of a neural network which focuses on object detection is the convolutional neural network (CNN). CNN is a neural network which uses the convolution operation in at least one of its layers [15].

One of the big disadvantages of NNs is that they need a large amount of training data to have adequate performance. Therefore, getting access to data is the bottleneck for neural networks. On the internet many 3D models of different objects are available for free in various formats such as Computer Aided Design (CAD). From a CAD model, it is possible to generate thousands of different synthetic images by alternating the background and adding texture to the objects.

It is possible to decrease the amount of training data needed by using a method called finetuning. This method is described further in chapter 2.

Many different types of neural networks exist, where the difference lies in the combination of hidden layers. In this thesis, the networks used are Faster R-CNN, Inception, SingleShot Multibox Detector, and MobileNet, all described in chapter 2.
1.2 Problem Formulation

This thesis will investigate if neural networks can be fine-tuned with synthetic images for the task of object detection on a video sequence. To optimize the development, the network will first be tested on images before being tested on a video sequence.

1.2.1 Limitation

To reduce training time, fine-tuning will be used. There also will be limitations on what type of objects the network will be able to detect.

There will be two different datasets: dataset A and dataset B. Dataset A consists of objects such as screw and plugs provided by IKEA. Background and texture combinations in dataset A were realistic since the purpose was to test if the network could differentiate objects in the real world. Dataset B is a video sequence taken of the real objects in dataset A. Dataset A and B are shown in Appendix A.

A computer with Intel Core i7-7700, NVIDIA GTX 1080 Ti was used. A HoverCam web camera was used to capture the video sequence. No new neural network architecture will be created. Instead an API called Tensorflow Object Detection API (version 1.7) [18] will be used, together with OpenCV (version 3.4.1) [8], Python (3.5) and Blender [7].

1.3 Thesis Outline

In chapter 2 the related work is presented. The method used is described in chapter 3. The experiment is presented in section 3.5. The results are shown in chapter 4 and discussed in chapter 5. The conclusions and future work of this thesis are presented in chapter 6.
Related work

Four topics are addressed in this chapter: neural networks trained on synthetic data, the concept of finetuning, object classification, and object classification using convolutional neural network. This chapter ends with a conclusion, describing a solution to the problems.

2.1 Using synthetic data

The time consuming parts of NNs are the training time and the gathering of training data. For the detection and classification of the object, the training data consist of two parts: the images and the corresponding annotations for each image. For this thesis, annotation means the creation of ground truth data, i.e. the bounding box for each object in the images. To get access to lots of training data, one can use synthetic data since it is possible to generate them automatically. Synthetic images, in this thesis, are images generated by sampling CAD models unless otherwise stated. By generating data automatically the ground truth is always accessible.

Annotating training data is a problem for scientists since it takes a long time and good accuracy is needed. Richter et al. were creative with annotating their data. Using the video game engine from Grand Theft Auto they could get access to both scenes with realistic appearances and labels at pixel level [32].

By using these realistic images, they showed that the work needed for annotation could be notably reduced. By combining the semantic segmentation dataset with real-world images, the accuracy increase even more.

Successful attempts have been made to train NNs using synthetic data to solve classification problems. In this case, successes mean having the best result for a
specific type of benchmark. The neural network created by Jaderberg et al. was trained for scene text classification [20], to classify whole words. The training images were computer generated with different fonts, shadows, and color. Distortion and noise were added to the rendered images to simulate the real world. It outperformed previous state-of-the-art methods for scene word classifications in the benchmarks ICDAR 2003, Street View Text, and IIIT5k-dataset. ICDAR 2003 is a competition in robust reading [25]. The amount of training data used was between 4 millions and 9 million, depending on the benchmark used.

Jaderberg et al. also created another neural network for text spotting, meaning detection and recognition of words. They created an end-to-end system for text spotting [21]. For the word detection part, they used a region proposal based mechanism and a CNN for the word recognition task. Their dataset was created in the same way as in the work [20]. The dataset contained 9 million images, 32x100 pixels. They used 900000 for testing, the same amount for validation and the rest for training. For the task of text recognition, their method had the best accuracy compared to the previous state of the art methods. Jaderberg et al. had good performance in the text spotting task, outperforming the previous state-of-the-art method [21].

Georgakis et al. trained their network with a combination of both real images and synthetic images [13]. The synthetic images were real images augmented. Objects with different scales and positions had been superimposed onto these images. The task for the network was to do object detection in a cluttered indoor environment.

Another work which trains a convolutional neural network with synthetic images is [30]. The network’s task was to predict a bounding box and the object class category for each object of interest on RGB images captured inside a refrigerator. Training the neural network with 4000 synthetic images, the network scored a mean Average Precision (mAP) of 24% on a test set. By adding 400 real images the mAP increased with 12%. In this paper, they used IoU (intersection over Union) for evaluating the bounding box predictions, see section 3.3.1 for a description of IoU.

### 2.2 Finetuning

The concept of finetuning refers to the approach of reusing training weights. These training weights comes from another neural network that as been created for another task. The weights are used to initialize the training [28]. For example, a neural network trained to classify cats can be fine-tuned to classify dogs. This method has resulted in state-of-the-art performances for several tasks. Examples of such tasks are object detection [33], [26], [27], tracking [36], segmentation [4], and human pose estimation[9]. With finetuning, the training time can also be
Object classification is identification of the object class in an image. Automatic classification, where no human is involved in the classification step, can be done using machine learning. Support Vector Machines (SVM) are methods used for classification. The Support Vector Machine, was first invented for binary classification problem [10]. An SVM tries to find a function which can separate the input data into categories, by mapping the input data non-linearly to a high dimensional vector space. In, for example, [14], [17], and [29], SVMs were used for the task of classifying land cover images.

More recent progress in object classification has been achieved by neural networks. Two state-of-the-art object classification networks are ResNet [5] and Inception net[34].

ResNet is a deep residual network, hence the name ResNet, and consists of 152 layers. Due to its large depth, it managed to achieve a 3.6% error rate (top-5 error) in the 2015’s edition of ImageNet Large Scale Visual Recognition Competition (ILSVRC) and thus won the classification task in the 2015 edition of ILSVRC [23]. A human has an error rate between 5 – 10%, meaning ResNet outperforms humans on this task [5].

The other neural network, Inception net, is a network that consists of inception modules. An inception module is a block of multiple parallel convolutional and max-pooling layers with different kernel sizes. The inception module makes the Inception net different from the traditional networks, which stack up convolutional and max-pooling layers [34]. It won the classification and detection task of ILSVRC in 2014 [23].

Neural networks are computationally heavy, requiring capable hardware to do the calculations. However, there is a network MobileNet which is a neural network developed for Mobile Vision Applications. Instead of both filtering and combining the output signal in one go, Mobilenet divides this step into two layers, one for filtering and one for combining. The two-layer separation greatly reduced the computation and model size [16].


2.4 Object detection

Object detection includes object classification, since object detection is about finding the object’s location and its category. The object’s location is mostly represented as a bounding box, shown in Figure 2.1 and Figure 2.2.

![Figure 2.1: Input image](image1.png)  ![Figure 2.2: Result](image2.png)

Two of the recent state-of-the-art methods for object detection are Faster R-CNN and Single-Shot Multibox Detector (SSD).

A Faster Region based convolutional network (F-CNN) consists of two modules. One module is a deep, fully convolutional network, a Region Proposal Network (RPN). A RPN takes an image as the input and outputs a set of rectangular regions. Each rectangle has a score indicating if the region is an object or background. The second module is a Fast R-CNN detector, which applies object detection on the regions proposed from the RPN. Faster R-CNN achieved a state-of-the-art accuracy on the dataset PASCAL VOC 2007 [31].

Single-Shot Multibox Detector is a feed-forward CNN. It produces a collection of bounding boxes with fixed size, and the probability for the presence of the object class in each box. To get the final detection, it has a non-maximum suppression step. SSD achieved an increase in accuracy and speed, compared to Faster R-CNN, when tested on the PASCAL VOC 2007 dataset [24].

The networks in section 2.4 used PASCAL VOC 2007 as a benchmark. Their datasets were divided as follows: 50% for training/validation and 50% for testing, i.e. images the network had not seen before, with a total of 9963 images [3].

Since they had the double amount of images, 9936 versus 4830, this thesis used 10% of the dataset for testing and the rest to train to compensate.

The remaining 90% was divided between the training and validation, 70% for training and 30% for validation.
Getting access to a large dataset is a limiting parameter for neural networks; it takes time and it is costly. This thesis will investigate how well NN performs after finetuning with synthetic images.

Hyper-parameters connected to the images are the batch sizes (and the image size), and number of batches/epochs to run the training. The effect of these parameters was the focus of this thesis and therefore pre-trained networks were used. The Support vector machine is an old technique and newer methods have surfaced with better accuracy. Thus, only deep learning will be used. The main interest was compare the Single-Shot multi-box Detector against the Faster R-CNN. Also to combine the object detection networks with the object classification networks, since all networks are state-of-the-art methods, is an interesting aspect. section 3.2 states all combinations this thesis will use. The Residual network was not used due to computer limitations.

Being inspired by [32], a comparison of the time needed to do manual and automatic annotation on a dataset was done. It is also investigated how the network performs on automatically annotated datasets vs manually annotated. The procedure is described later in subsection 3.1.3.
First presented in this chapter is the rendering of synthetic data, followed by combinations of neural networks and the evaluation method. In Figure 3.1 a flow chart over the work flow is shown.

**Figure 3.1: Flow chart of work flow.**
3.1 Generating the Datasets

This section will describe the creation of the two datasets; A and B. Dataset A is synthetic images of five different objects; attachment, shelf plug, dowel, expandable plug, and screw. This dataset was used to train the neural network. Dataset B are images sampled from a video sequence containing the physical objects and was used to evaluate the networks. Examples of the two dataset are seen in Appendix A.

3.1.1 Rendering Images

All computer generated data were created from CAD models. The models were either provided by the furniture company IKEA or found on the website GrabCad [2]. The images were rendered by the open source 3D creation suite program Blender [7]. To generate a large variety of data, different backgrounds, object texture and object rotation and camera locations were used. The background images were taken from the website Pexels [1].

3.1.2 Video Recording

To create the dataset B, physical objects of the dataset A were acquired. By recording with a web camera, the objects were introduced into the scene one by one.

3.1.3 Creation of Ground Truth Data

Ground truth data was created using either an open-source program, LabelImg [35] or Blender. LabelImg allows the user to create a bounding box around each object and save the data as an .xml (extensible markup language, file). This file can then be converted to other types. The same information was created when rendering the synthetic images using Blender. In this thesis, annotation implies creating ground truth data.

3.2 Dataset Distribution and Network Pairings

Different network architectures were compared and evaluated against each other. The method is described in section 3.3.

The network pairings used are stated below:

- Faster R-CNN + Inception
- SSD + Inception
- SSD + MobileNet

The networks were chosen based the literature study that was done in chapter 2 and the availability of pre-trained models. Pre-trained models means the network has already been trained on another dataset. Since no pre-trained models exists for Faster R-CNN + Mobilenet this pairing was not used.
3.3 Evaluation Metrics

To evaluate the networks, two different losses were used: classification loss and localization loss. They were calculated for the three different stages: training, validation, and testing. These losses were provided by the Tensorflow Object Detection API. PASCAL mAP was also used on the validation and testing dataset.

3.3.1 Losses

This thesis used the same loss as the networks stated in section 3.2 due to usage of fine-tuning. The loss for categorizing a detected object into categories, object vs background, is the binary classification loss and is described as a sigmoid function, shown in (3.1). The localization is the loss of the bounding box regression and is represented as a smooth L1 loss, the Auber loss, see (3.2).

\[ L_c(x) = \frac{1}{1 + e^{-x}} \]  
(3.1)

\[ L_R = \begin{cases} 
0.5x^2 & \text{if } |x| < 1 \\
|x| - 0.5 & \text{otherwise}
\end{cases} \]  
(3.2)

The lower the losses, the better the network performs.

Intersection Over Union

Intersection over Union (IoU) is a measurement for the overlap of two bounding boxes, A and B. In this case it is the overlap between the ground truth and the network’s output. The IoU is the quotient of the intersection and the area of union [19]. In both [30] and [21], IoU was used as an evaluation metric. Due to the simplicity of interpreting IoU, this metric will be used for evaluation within this thesis.

Intersection area

Union area

Figure 3.2: Area of intersection

Figure 3.3: Area of union
3.3.2 PASCAL Mean Average Precision

To describe PASCAL mAP we need five terms:

- True Positive (TP)
- False Positive (FP)
- False Negative (FN)
- Precision
- Recall

The true positive rate is the number of correct detections. False negatives are missed detections. False positive occurs when multiple detections of the same object are detected, all detections other than the first correct one are false.

Recall is defined as the proportion of all positive detections with IoU equal or greater than a certain value, in this case 0.5 [3].

Precision is the proportion of all recalls that are true positive [3].

PASCAL mAP is defined as the mean precision at a set of eleven equally spaced recall levels \([0, 0.1, ..., 1]\) [3], see (3.3).

\[
AP = \frac{1}{11} \sum_{\text{Recall}_i} \text{Precision}(\text{Recall}, i) \quad (3.3)
\]

The higher the mAP value is, the better the network performs.

3.4 Parameters to tune

When training a neural network several parameters can be tuned to give better performance. The ones evaluated in this thesis are:

- Batch size: number of images in one batch.
- Number of epochs: number of times all of the training data has gone through the network.
- Total numbers of images used in training
3.5 Experiments

As stated in chapter 3, the parameters tuned were batch size, number of epochs and the total number of images used in training. Three main experiments were executed; experiment 1, experiment 2, and experiment 3. 100000 batches were used for all runs: training, validation and testing.

Experiment 1 only used synthetic data and consists of the following sub-experiments:

1. Testing different network configurations
2. Batch size vs epochs
3. Largest image size manageable

Sub-experiment 1 was done using sub-experiment 2. Table 3.1 specifies how the batch size and image size was varied in experiment 2.

In sub-experiment 3, due to large images, the batch size needs to be small. The image size of 600x1040 with a batch size of 1 was used due to hardware limitations. Also this experiment was done with the network architecture that had the best performance when testing on dataset B, which would later be shown to be Faster R-CNN + Inception net. Sub-experiment 2 and 3 used the network that had the best performance in experiment 1.

<table>
<thead>
<tr>
<th>Test</th>
<th>Baseline</th>
<th>#1</th>
<th>#2</th>
<th>#3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Batch Size</td>
<td>1</td>
<td>24</td>
<td>35</td>
<td>1</td>
</tr>
<tr>
<td>Image Size</td>
<td>300x300</td>
<td>300x300</td>
<td>240x240</td>
<td>600x1040</td>
</tr>
<tr>
<td>Epochs</td>
<td>100000</td>
<td>4166</td>
<td>2857</td>
<td>100000</td>
</tr>
</tbody>
</table>

In experiment 2 five different networks were tested, which are listed below:

- Faster R-CNN + Inception: 10 percent
- Faster R-CNN + Inception: 50 percent
- Faster R-CNN + Inception: 100 percent
- SSD + Inception
- SSD + MobileNet.

These networks were trained on the dataset A and then validated on dataset B. Table 3.2 shows how many images that were used to train the different Faster R-CNN + Inception networks. The reason why there are three different versions of Faster R-CNN + Inception is due to it had the best mAP when testing on dataset B, see Figure 4.44.
Table 3.2: Experiment 2: Testing different dataset size. The percentage is in terms of total amount of images in the dataset.

<table>
<thead>
<tr>
<th>Test</th>
<th>#1</th>
<th>#2</th>
<th>#3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of real images</td>
<td>540 (10%)</td>
<td>2686 (50%)</td>
<td>5392 (100%)</td>
</tr>
<tr>
<td>Batch size</td>
<td>24</td>
<td>24</td>
<td>24</td>
</tr>
</tbody>
</table>

For experiment 2 the only interesting evaluation metric is the mean average precision, since no parameters are tuned, thus only the mAP will be plotted.

Experiment 3 was to compare the automatic annotation with manual annotation. In this experiment SSD + MobileNet was used due to the short training time, shown in Figure 4.40. The validation was done by comparing the IoU of the ground truth, the automatic generated and the manual one, and to compare the classification and localization loss.
In this chapter, the results of the different network configurations stated in section 3.5 are presented. The chapter also includes the comparison of manual and automatic annotation is evaluated.

In all figures where mean average precision is plotted for the whole dataset, only results from the validation and the testing are shown. This is due the Tensorflow API only calculating the mAP for the validation and testing set.

### 4.1 Testing Different Network Configuration

In this section, the results of the different network architectures are presented. The classification and localization losses, and mean average precision are plotted for each subset: training, validation, and testing.

#### 4.1.1 Faster R-CNN and Inception

Here the results of the different configurations with Faster R-CNN + Inception net are presented. The dataset used is dataset B.

**Baseline**

- Time needed for training: 3.3 hours
- Image size: 300x300
- Batch size: 1
- Results are shown in Figure 4.1 - Figure 4.3.
**Figure 4.1:** Classification loss, Faster R-CNN + Inception, batch size 1

**Figure 4.2:** Localization loss, Faster R-CNN + Inception, batch size 1
4.1 Testing Different Network Configuration

Figure 4.3: Mean average precision, Faster R-CNN + Inception, batch size 1
**Configuration 1, #1**

Time needed for training: 22 hours  
Image size: 300x300  
Batch size: 24  
Results are shown in Figure 4.4 - Figure 4.6.

*Figure 4.4: Classification loss, Faster R-CNN + Inception, batch size 24*
4.1 Testing Different Network Configuration

Figure 4.5: Localization loss, Faster R-CNN + Inception, batch size 24

Figure 4.6: Mean average precision, Faster R-CNN + Inception, batch size 24
**Configuration 1, #2**

Time needed for training: 31.5 hours  
Image size: 300x300  
Batch size: 35  
Results are shown in Figure 4.7 - Figure 4.9.

*Figure 4.7: Classification loss, Faster R-CNN + Inception, batch size 35*
4.1 Testing Different Network Configuration

![Localization Loss](image)

**Figure 4.8:** Localization loss, Faster R-CNN + Inception, batch size 35

![Mean Average Precision](image)

**Figure 4.9:** Mean average precision, Faster R-CNN + Inception, batch size 35
**Batch of size 1, Image size 600x1040**

Time needed for training: 5.3 hours
Image size: 600x1040
Batch size: 1
Results are shown in Figure 4.10 - Figure 4.11.

*Figure 4.10: Classification loss, Faster R-CNN + Inception, batch size of 1*
4.1 Testing Different Network Configuration

**Figure 4.11:** Localization loss, Faster R-CNN + Inception, batch size of 1

**Figure 4.12:** Mean average precision, 0.5 IOU, Faster R-CNN + Inception, batch size of 1
Summary: Faster RCNN and Inception

The results of the testing dataset with all three different batch sizes, image size 300x300, are plotted together in Figure 4.13 to Figure 4.15.

\[\text{Classification Loss}\]

 Förmer 4.13: Classification loss, Faster R-CNN + Inception
4.1 Testing Different Network Configuration

**Figure 4.14:** Localization loss, Faster R-CNN + Inception

**Figure 4.15:** Mean average precision, Faster R-CNN + Inception
4.1.2 SSD and Inception

In the following sections, results from different SSD + Inception runs are presented.

**Baseline**

Time needed for training: 2.5 hours  
Image size: 300x300  
Batch size: 1  
Results are shown in Figure 4.16 - Figure 4.18.

![Classification Loss](image)

*Figure 4.16: Classification loss, SSD + Inception, batch size of 1*
4.1 Testing Different Network Configuration

**Figure 4.17:** Localization loss, SSD + Inception, batch size of 1

![Localization Loss Graph](image)

**Figure 4.18:** Mean average precision, SSD + Inception, batch size of 1

![Mean Average Precision Graph](image)
Figure 4.17 has some incomplete values for the validation run, the values were NaN and therefore not plotted.
Configuration 1, #1

Time needed for training: 14 hours
Image size: 300x300
Batch size: 24
Results are shown in Figure 4.19 - Figure 4.21.

Figure 4.19: Classification loss, SSD + Inception, batch size of 24
Figure 4.20: Localization loss, SSD + Inception, batch size of 24

Figure 4.21: Mean Average Precision, SSD + Inception, batch size of 24
Configuration 1, #2

Time needed for training: 8.4 hours
Image size: 300x300
Batch size: 35
Results are shown in Figure 4.22 - Figure 4.24

![Classification Loss](image)

**Figure 4.22:** Classification loss, SSD + Inception, batch size of 35
\textbf{Figure 4.23:} Localization loss, SSD + Inception, batch size of 35

\textbf{Figure 4.24:} Mean Average Precision, SSD + Inception, batch size of 35
4.1.3 SSD and MobileNet

In this section, the results when using SSD together with mobilenet are presented.

**Baseline**

Time needed for training: 1.7 hours  
Image size: 300x300  
Batch size: 1  
Results are shown in Figure 4.25 - Figure 4.27

![Classification Loss Graph](chart)

**Figure 4.25:** Classification loss, SSD + MobileNet batch size of 1
Figure 4.26: Localization loss, SSD + MobileNet batch size of 1

Figure 4.27: Mean Average Precision, SSD + Inception, batch size of 1
4.1 Testing Different Network Configuration

**Configuration 1, #1**

Time needed for training: 14 hours
Image size: 300x300
Batch size: 24
Results are shown in Figure 4.28 - Figure 4.29.

![Classification Loss](image)

*Figure 4.28: Classification loss, SSD + MobileNet, batch size of 24*
Figure 4.29: Localization loss, SSD + MobileNet, batch size of 24

Figure 4.30: Mean Average Precision, SSD + Inception, batch size of 24
Configuration 1, #2

Time needed for training: 19.5 hours
Image size: 300x300
Batch size: 35
Results are shown in Figure 4.31 - Figure 4.32.

Figure 4.31: Classification loss, SSD + MobileNet, batch size of 35
Figure 4.32: Localization loss, SSD + MobileNet, batch size of 35

Figure 4.33: Mean Average Precision, SSD + MobileNet, batch size of 35
4.1.4 Summary: Single-Shot Multibox Detector

In this section, all results for Single-shot Multibox Detector are combined and plotted in Figure 4.34 to Figure 4.36. The network with the best performances is the SSD with a batch size of 24, irrespective of using Mobilenet or Inception net. SSD + Inception with a batch size of 35 performed almost as well as the batch size of 24 in the loss categories. Its mean average precision converged after the same number of epochs as batch 24. Using a batch size of 1 gave a high loss and the mAP was low.

![Classification Loss](image)

*Figure 4.34: Classification loss, SSD*
Figure 4.35: Localization loss, SSD

Figure 4.36: Mean Average Precision, SSD
4.1.5 Summary: Different Network Architecture And Batch Sizes

In Figure 4.37 and Figure 4.38, the classification loss and the localization loss are plotted for all the different network architectures. The results are from using the testing dataset. Training time is summarized in Figure 4.40.

![Classification Loss Graph](image)

*Figure 4.37: Classification loss of testing dataset*
Figure 4.38: Localization loss of testing dataset
4.1 Testing Different Network Configuration

**Figure 4.39:** Mean average precision of testing dataset

![Mean Average Precision](image)

**Figure 4.40:** Time to train

![Time to train](image)
4.2 Epochs Versus Dataset Size

In this section the results of varying the training dataset size are shown in Figure 4.41 to Figure 4.43. It is shown in subsection 4.1.5 that the Faster R-CNN + Inception had the best performance in all three categories. Therefore this network configuration with a batch size of 24 was chosen. The training ran for 400 epochs and the dataset used was the testing data from dataset B.

![Classification Loss](image)

*Figure 4.41: Classification loss, varying dataset size*
Figure 4.42: Localization loss, varying dataset size
Figure 4.43: Mean average Precision, varying dataset size
4.3 Testing on real images

The interesting metric to look at in this case is the mean average precision, which is shown in Figure 4.44.

Figure 4.44: Mean average precision on dataset C
4.4 Automatic and Manual Annotations

The time needed to generate the dataset via manual annotation and automatic annotation is shown in Table 4.1.

**Table 4.1: Time needed to create the dataset via manual and automatic annotation**

<table>
<thead>
<tr>
<th></th>
<th>Manual</th>
<th>Automatic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of images</td>
<td>3561</td>
<td>7806</td>
</tr>
<tr>
<td>Time</td>
<td>6h</td>
<td>38 min</td>
</tr>
</tbody>
</table>

Figure 4.45 shows the histogram over the IoU for the ground truth data between the manual and automatic method.

**Figure 4.45: Intersection over union, groundtruth data**

A comparison of manual and automatic annotated data with the SSD + MobileNet was done and the result is shown in figure Figure 4.46 to Figure 4.51. A batch size of 24 with 3561 images was used.
Figure 4.46: Classification loss comparison, training dataset
Figure 4.47: Classification loss comparison, validation dataset
Figure 4.48: Classification loss comparison, testing dataset
Figure 4.49: Localization loss comparison, training dataset
Figure 4.50: Localization loss comparison, validation dataset
Figure 4.51: Localization loss comparison, testing dataset
In this chapter, the results from chapter 4 are analyzed.

5.1 Networks

In the following sections, the different results from each network configuration are discussed. First the Single-Shot multibox Detector, followed by Faster R-CNN.

5.1.1 Single-Shot Multibox Detector

It is shown in Figure 4.34 and Figure 4.35 that increasing the batch size to a moderate size for SSD networks gives better results. However, a huge batch size yields a poorer outcome. The batch size of 35 had worse results for both SSD + Inception and SSD + Mobilenet in the loss category. While for the mean average precision, see Figure 4.36, a batch size of 35 gave the same result as a batch size of 24 for the SSD + Inception network, it converged to 1. Training loss for all network architectures and batch sizes was unstable at each run with which could be a sign for overfitting. But also high learning rate or regularization could cause this pattern.

5.1.2 Faster R-CNN and Inception

A batch size of 1 had an mAP converging to roughly 0.58, a batch size of 35 converged to 0.6 and a batch size of 24 converged to approximately 0.95. One reason why a batch size of 1 performed worst could be that the network has yet not learned enough features. The network suffered from overfitting when using
a batch size of 35. It is also shown in Figure 4.39 that the outcome of using an image size of 660x1040 with batch size 1 is as good as using a batch size of 24 with image size of 300x300.

5.2 Epochs versus Batches

It is shown in section 4.2 that using the smallest size of the dataset, 10% of total amount of test images, gave the worst result in the classification/localization loss. In addition, it required longer training time for the mean average precision to converge. Using 50% or 100% of the dataset gave approximately the same results when comparing the classification/localization loss. The mean average precision converged the fastest when using 50% of the dataset.

5.3 Testing On Real Images, Video Sequence

The mAP decreased for all the networks when comparing the metrics between synthetic data and real images. The network’s mAP will converge to 1 (see Figure 4.39), and it is shown in Figure 4.44 that when testing the network after 100000 batches the best result we get is 0.81. The decrease in performance might be explained by the scale of an object in each image. An example of a frame can be seen in Figure 5.1, where the distance to the camera is further away compared to the training images can be seen in Appendix A.

![Figure 5.1: Frame from video sequence](image)

Another reason for worse performance could be the sharpness of the images. Even though some of the training images had blur added to them as a pre-processing step, the network had trouble with the object being out of focus.

It is shown in Figure 4.44 that all the networks performed approximately the same except for SSD + Mobilnet.

5.4 Annotation: Manual vs Automatic

As seen in Table 4.1, the time necessary to manually annotate the images was much longer. The manual annotation had a total of 3561 images due to a time
limitation, doing 3561 images took 6 h.

It is shown in Figure 4.46 - Figure 4.51 that the two losses are higher in every case. Implying that the automatic generated ground truth yields higher accuracy. The reason for this can be found in Figure 4.45, which is human error.
In this chapter, the conclusions and future works are presented.

6.1 Conclusions

There are several conclusions that can be drawn from this thesis. A neural network with the purpose of object detection can be fine-tuned using synthetic data to detect other objects. Faster R-CNN + Inception network had the best accuracy. Out of the three different network architectures used, while also taking the longest time to train.

The result shows further that longer training time does not necessarily give the best result, what mattered was the size of the datasets and the batch sizes. The larger the dataset, the higher the accuracy. Yet having too large batch size results in overfitting.

Large dataset requires a lot of labeling, if automatically generating ground truth data can both increase the accuracy but also reduces the amount of manual labor then large dataset would no longer be a problem. Less manual labor also decreases the chance of human errors.

6.2 Future Work

Easy access to ground truth data is achievable by generating synthetic data automatically. Instead of saving the bounding box of an object, also an object mask could be used. An object mask means only the pixels of the object are marked. The reason one would want to save an object mask instead of the bounding box
is due to the bounding box having background noise while an object mask would only contain the interesting pixels i.e. the object. It would be interesting to verify whether this improves the accuracy further.

Also, in this thesis the networks were finetuned. An interesting aspect would be to train a neural network from scratch using only computer-generated images in order to verify that synthetic data is suitable for learning from scratch, too.
Appendix
Datasets

Two different datasets were used throughout, dataset B was only used for testing purposes.

**Dataset A**

Dataset A consists of five different objects and consists of 5392 images. These can be seen in Figure A.6 and Figure A.10.

**Dataset B**

Dataset B is a video recorded by a web-camera of the real objects in real life.

*Figure A.1: handle  Figure A.2: car  Figure A.3: eStop*
Figure A.4: cabel-Protetor
Figure A.5: Turn knob
Figure A.6: Attachment
Figure A.7: Shelf plug
Figure A.8: Dowel
Figure A.9: Expandable plug
Figure A.10: Screw
Bibliography


