EXAMENSARBETE INOM DATALOGI OCH DATATEKNIK, AVANCERAD NIVÅ, 30 HP

STOCKHOLM, SVERIGE 2020

Telecom Equipment Segmentation and Detection Using Drone Images

MIKAEL KVIST

KTH
SKOLAN FÖR ELEKTROTEKNIK OCH DATAVETENSKAP

Telecom Equipment Segmentation and Detection Using Drone Images

MIKAEL KVIST

Master in Computer Science
Date: July 1, 2020
Supervisor: Mårten Björkman
Examiner: Danica Kragic
School of Electrical Engineering and Computer Science
Host company: Ericsson
Swedish title: Segmentering och objektdetektering av Telekomutrustning med hjälp av drönarbilder

Abstract

An automated AI solution for out-door Telecom equipment segmentation is beneficial to most of the workflow for site survey and engineering performed by human. AI solutions that perform segmentation tasks are today trained with supervised learning which requires manually labeled images. However, labeling images is both time consuming and expensive, which makes semi-supervised learning attractive where unlabeled data is used to further improve the performance of models.

To determine if semi-supervised learning can be used to improve the performance of instance segmentation, the effectiveness of a semi-supervised learning approach called FixMatch [1] was tested for instance segmentation using a custom dataset. The dataset contains 590 labeled and 1000 unlabeled drone-captured images of Telecom equipment. An extension was made to FixMatch [1] where the predicted bounding boxes and masks are augmented like the images, which makes it possible to use FixMatch [1] for instance segmentation. The extension was evaluated with mean Average Precision (mAP) but only achieved 1 point higher mAP than without using the extension. The small improvement in performance shows that this semi-supervised approach is not suitable for instance segmentation of Telecom equipment where the amount of unlabeled data is twice the labeled.

Sammanfattning

En automatiserad AI lösning för segmentering av Telekomutrustning utomhus är välgörande för inspektioner av Telecomutrustning som utför manuellt. AI lösningar för segmentering tränas idag med övervakad inlärning vilket kräver märkt data. Men att märka bilder är både tidskrävande och dyrt, vilket gör delvis övervakad inlärning attraktivt där omärkt data används för att förbättra modellen.

En method kallad FixMatch [1] är har testats på ett eget dataset för instans segmentering för att bestämma om delvis övervakad inlärning kan användas för att förbätre modellen inom instans segmentering. Datasettet innehåller 590 märka och 100 omärkta bilder av Telekomutrustning tagna med en drönare. En utveckling har gjorts till FixMatch [1] där boxarna och maskerna augmenteras likt bilderna, detta gör det möjligt att använda FixMatch [1] till instans segmentering. Prestandan hos utvecklingen har mätts med mean Average Precision (mAP) men uppnår endast 1 poäng högre mAP jämfört med att inte använda utvecklingen. Den lilla prestandaförbättringen visar att den här delvis övervakande inlärningsmetoden inte passar till instans segmentering av Telekomutrustning där den omärkta deten är dubbelt så stor som den märkta.

Contents

Chapter 1

Introduction

Radio tower and its equipment are used to emit and transmit radio signals, which is necessary for communication over 4G, older generations of mobile networks, and soon 5G. To maintain and improve the performance, the equipment needs to be inspected, for example, the equipment might be broken, or you might need to update the equipment. However, before you can make these decisions you need to know what kind of equipment there are in the towers. All of this is currently done manually, where workers climb up the towers and inspect the equipment. However, radio tower equipment is sometimes placed in places that are hard to reach or the manual inspection might even be dangerous, e.g. the tower equipment could be high up in the towers in combination with bad weather conditions. By using a drone and its images to inspect the equipment the manual labor can be reduced which reduces the cost of the inspection, the risk of accidents, but also makes the inspection of tower equipment faster. There are different ways of facilitating the inspection of the images, e.g. object detection where bounding boxes are created and instance segmentation which create colored masks for the equipment. Both of these makes it easier to know where the equipment is in the images, but instance segmentation does a better job of highlighting where the equipment is.

1.1 Objectives

In this project an AI solution which helps the inspection of Telecom equipment is sought after. As mentioned above this can be done with both object detect and instance segmentation. There are a few reasons why instance segmentation was chosen over object detection. First, since the equipment is tightly packed in a small area of the image, i.e. the tower. This will cause the bounding

boxes from object detection to form clusters and it will not be obvious what equipment is where with only a glance at the images. In instance segmentation pixel-wise masks are created which makes this easier. Secondly, the masks in instance segmentation are colored according to equipment types which makes it possible to recognize the type of equipment without much effort. Even if the bounding boxes were colored, they are still thin which makes is harder to recognize the equipment by only looking at the color.

1.1.1 Research Question

When creating a model for instance segmentation it is important that the masks created by the model is at the right location but also fit the Telecom equipment well. Since creating labels for images is expensive it would be useful if unlabeled images could be incorporated to improve the quality of the masks produced by the model. Hence, the research question is whether techniques for semi-supervising learning improve the performance of Telecom equipment detection and segmentation using drone images.

1.1.2 Limitations

This project do not involve collecting the images as they already exist. There are about 1600 images of Telecom towers in total and 590 will be labeled. There are many different kinds of Telecom equipment depending on who created them. This means that the model trained on these images will be able to detect the Telecom equipment in the images but will not be a general Telecom equipment detector that can detect Telecom equipment of all kinds.

1.2 Ethics, sustainability and societal impact

The ethical problem of using a dataset consisting of Telecom towers is that the background might contain persons or locations that can be recognised, which could be used to locate the towers and destroy them. To avoid those situations the images shown in this paper will be cropped. When working with sensitive images it is important not to upload the images online. This affect the way the images can be labeled as not all online tools could be used.

Ericsson is commited to sustainability which is reflected in their projects. It is important to inspect and repair the possible broken Telecom equipment for maintaining a good mobile network.

Without working Telecom equipment the mobile network will suffer and this will affect the society since we use it for everything. For example, a functioning mobile network is needed to search for directions when lost, find where to buy a product and calling friends.

Chapter 2

Background

There was a lot of information to go through before semi-supervised learning could be done for instance segmentation. We will start by briefly presenting what a convolutional neural network and a backbone network are, followed by the different kinds of architecture types of object detection. Then techniques for small object detection will be shown with an explanation of what instance segmentation is, and lastly present techniques of semi-supervised learning.

2.1 Fully-connected layer

A fully-connected layer, in a feedforward neural network that consists of layers of neurons, connects the output of all neurons in one layer to the input of the subsequent layer. Thus, if the input is a vector of size 1000 and the output size is 10 the number of weights needed to connect everything is $1000 \times 10 = 10000$. This type of layer is typically used last in a network, where the output is a single or a set of numbers. An example of a fully-connected layer is shown in Fig. 2.1.

2.2 Convolutional layers

A convolutional layer is a layer that performs a convolution between two matrices. In the context of this study a convolution, or more specifically a 3D convolution, is an operation between a filter and an image, where the filter can also be called a kernel. The image has a width and height and three channels for RGB, the filter has the same number of channels but its own width and height. In a convolution the output is produced by sliding the filter over the image. For each position of the filter in the image, a number is computed as the

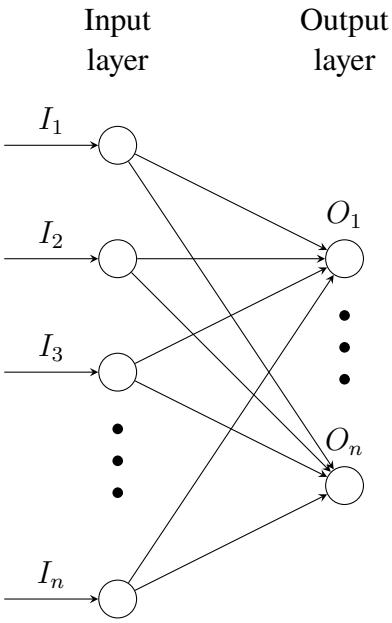


Figure 2.1: A fully-connected layer. The arrows represents the different weights.

sum of the element-wise multiplication between the filter and the local window of the image. These numbers are then placed in a matrix, this is shown in Fig. 2.2. Apart from the size of the filter there are two parameters that can be set for a convolution. The first is padding, as seen in Fig. 2.2 the outputs size is smaller than the image. To avoid this a border around the image, called padding, can be used. Usually the padding consists of zeros. The other parameter is the stride, which has the opposite effect. The stride is the size of the jump of the filter when sliding over the image. If the stride is set to one every position of the filter in the image is used, but if the stride is two, the filter skips every other position. The output's width and height depends on the padding, the stride, and the size of image and the filter, while the number of channels in the output is the number of filters used. For example, if a vertical and a horizontal edge detector are used the output will have two channels. In networks the filters consist of weights which are learned with back-propagation, [2].

2.3 Pooling

Another common type of layer is pooling. Pooling uses the same sliding action like the convolutional layer but without padding and instead of summing the

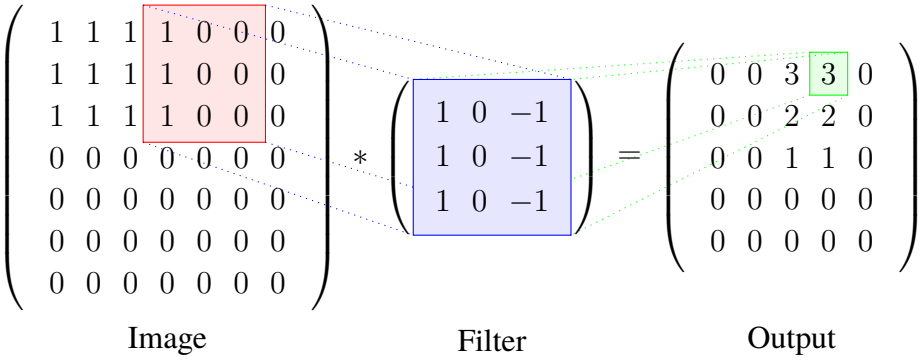


Figure 2.2: A convolution between an image and a filter that detects edges where the is no padding and the stride is 1.

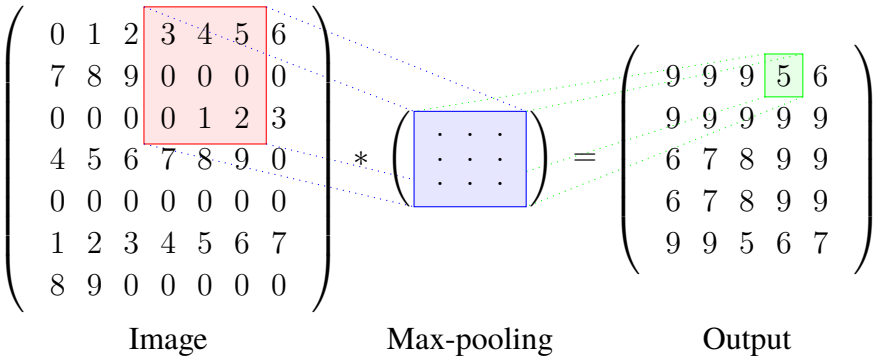


Figure 2.3: Max-pooling with the stride set to 1.

element-wise products an operation like maximum or average is performed on the local window of the input image. In a pooling layer the operation is performed on local windows per channel, meaning that the values from the different channels are not mixed. The output's width and height depends on the size of the filter and the stride, while the number of channels is the same as for the input. A pooling layer makes the network invariant to local translations and by reducing the dimensionality of the input the network becomes faster, but with a cost of sacrificing some information. Fig. 2.3 shows an example of max-pooling, [2].

2.4 Convolutional neural networks

Convolutional neural networks (CNNs) are networks that can take images as inputs. A CNN may have different kinds of layers, for example convolutional

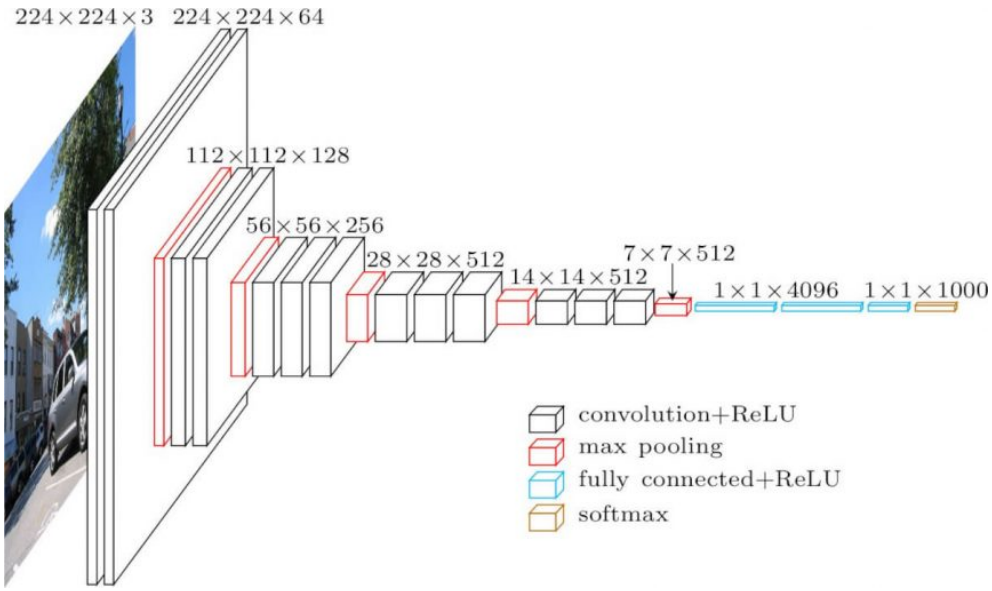


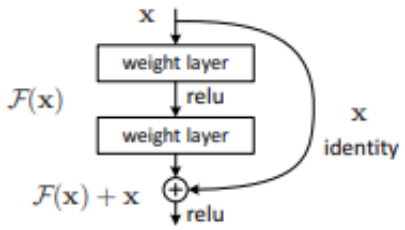
Figure 2.4: VGG16 [3] is a famous convolutional neural network, from [4].

layers, pooling layers and fully-connected layers. VGG16 [3], shown in Fig. 2.4, is a famous CNN, that starts with a few convolutional layers followed by a pooling layer that reduced the width and height of the layer. This grouping of convolutional layers followed by a pooling layer is repeated a few times and then ends with fully-connected layers, more information about VGG16 [3] is in section 2.5. Each layer of a CNN will be a 3-dimensional matrix which is called a feature map because it shows where a particular feature is present. For example, if a filter is an edge detector the output of the convolution with the image (or previous layer) will be a feature map that shows where the edges are.

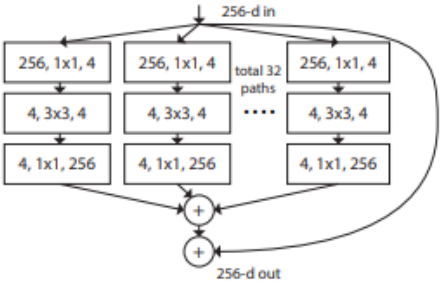
2.5 Backbone network

The first step in object detection involves sending the images through a network, called a backbone network. This network, typically a CNN produces an encoding for the image, which is called a feature map. This feature map is then used to interpret the images and detect objects. A summary of common backbone networks will be given below.

VGG is a network proposed by Visual Geometry Group [3], which increases the depth, i.e. adds more layers of previous networks by using a very small 3×3 convolutional filter. A VGG network [3] is shown in Fig. 2.4. VGG [3] also increased the accuracy of the network by going deeper [3]. Since



(a) Residual block which ResNet [5] consists of, image taken from [5].



(b) ResNeXt [6] have many parallel network blocks with the same topology, image taken from [6].

Figure 2.5: ResNet [5] and ResNext [6] both uses short cut connections.

then newer more complex structures have been proposed like residual network (ResNet) [5]. A ResNet [5] facilitates deeper networks by using shortcut connections and in [5], it is proven that networks can gain accuracy by going deeper. ResNet is constructed by repeated use of residual blocks, see Fig. 2.5a, blocks that make it possible to form deeper networks.

Another newer option is ResNeXt [6], this network was proposed for image classification and uses parallel network blocks with the same topology, i.e. instead of having a single path, the path branches out to several paths with the similar layers, the different path-outputs are then aggregated. They call this new dimension *cardinality* and it has been shown that increasing the cardinality improves the performance and is more effective in doing so than increasing the width, i.e. the width and height of the layers, or depth of the network.

Inception [7] is another popular network that can be used for feature extraction, i.e. creating the feature map, it increases the depth and width while maintaining the computational cost constant. The inception network [7] consists of stacked inception modules that apply different convolutional filters, e.g. 1×1, 3×3, and 5×5 and max-pooling and then concatenate these as output. This reduces the hyperparameters since you use filters of all sizes and do not have to choose the size of the convolutional filter. To keep the computational cost down a 1×1 convolution is done before the 3×3 and 5×5 convolutions, this reduces the channels which reduces the size of the representation, but the new representation should still contain a lot of information from the previous layer. It is thanks to the 1×1 convolution that they manage to increase the depth and width of the network while keeping the computational cost the same. Inception [7] is similar to ResNet [5] in the way that it also

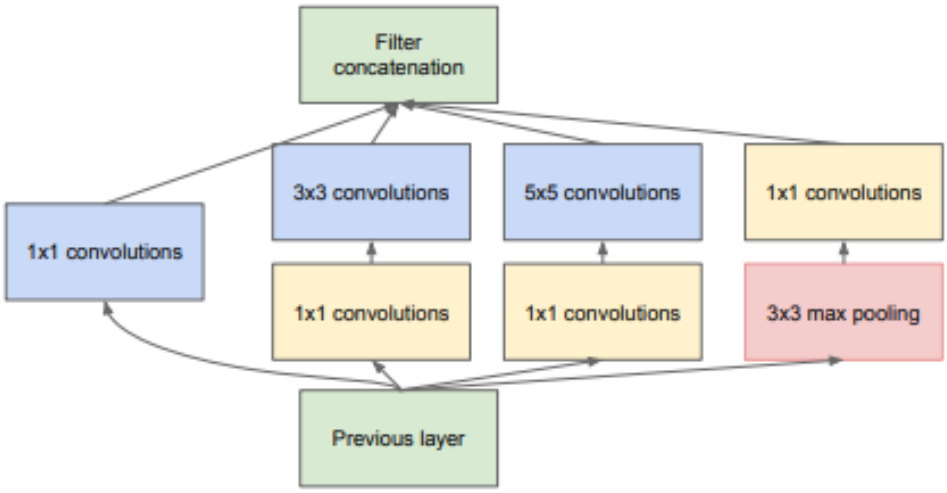


Figure 2.6: Inception module [7] with dimentionality reduction, Note that the direction if the flow is opposite from the previous images. Image taken from [7].

repeatedly uses the same modules to build the network; the Inception module is shown in Fig. 2.6. Since the introduction of Inception [7], improvements have been made and now there exists a network called Inception-ResNet [8] that combines the ideas of the Inception module and residual learning.

2.6 Object Detection

Traditional detection algorithms were in the early 2000 constructed with hand-crafted features but lead to a plateau in 2010. Later in 2012, convolutional neural networks were reintroduced [9] and in 2014 deep convolutional networks were used in object detection with huge success [10], which lead to the continuously evolving area of object detection we have today. Apart from the faster pace, object detection has also, in recent years, split into two branches; one-stage detector and two-stage detectors. For a more detailed description of the history of object detection, we recommend reading [11].

2.6.1 One-stage detectors

One-stage detectors do everything in one step. This leads to low computational times which makes these detectors especially suitable for real-time detection. A well known one-stage object detector is You Only Look Once (YOLO) [12],

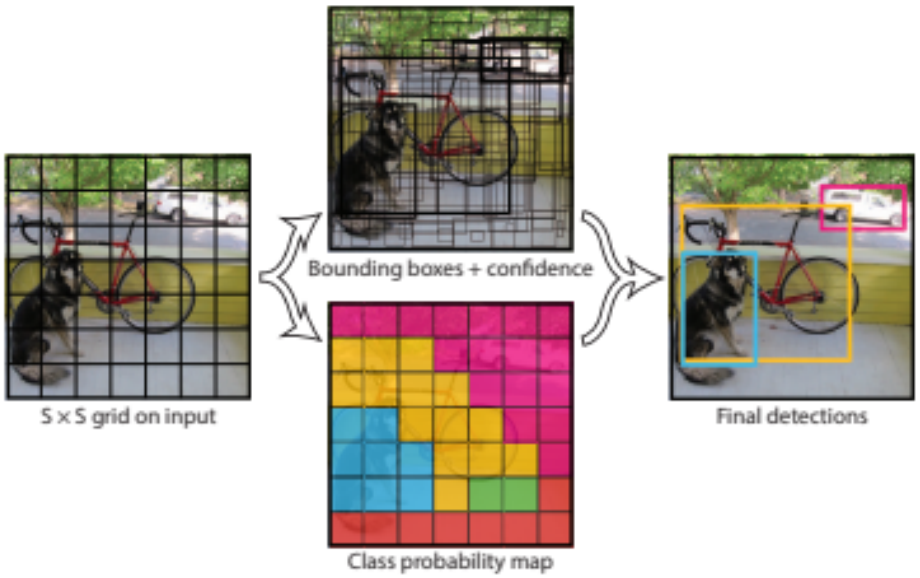


Figure 2.7: YOLO [12]. Image taken from [12].

see Fig. 2.7. YOLO [12] divides an image into a $S \times S$ grid and let each cell in the grid detect B number of objects, where S and B are parameters that need to be set. If the center of an object lies within a cell that cell is expected to detect that object. Thanks to the formulation of the detection problem YOLO [12] will only use one network that can be trained end-to-end. Since the introduction of YOLO [12], improvements have been made and the latest version is called YOLOv3 [13].

Single Shot Detector (SSD) [14] is another one-stage object detector. This detector adds convolutional feature layers at the end with decreasing size, and at each grid cell evaluate a few selected boxes with different aspect ratios. This makes SSD [14] better at detecting objects of different scales than YOLO [12] which only has one size of the feature layer. Despite using feature layers of different resolution SSD [14], like YOLO [12], do not detect small objects well [15]. Fig. 2.8 shows the idea behind SSD [14].

2.6.2 Two-stage detectors

Two-stage detectors refer to the detectors that in a first step detects the objects and in the second step refines the detections. These frameworks are generally slower than a one-stage detector but has better accuracy. Regions with CNN features (R-CNN) [10] is a successful two-stage framework introduced

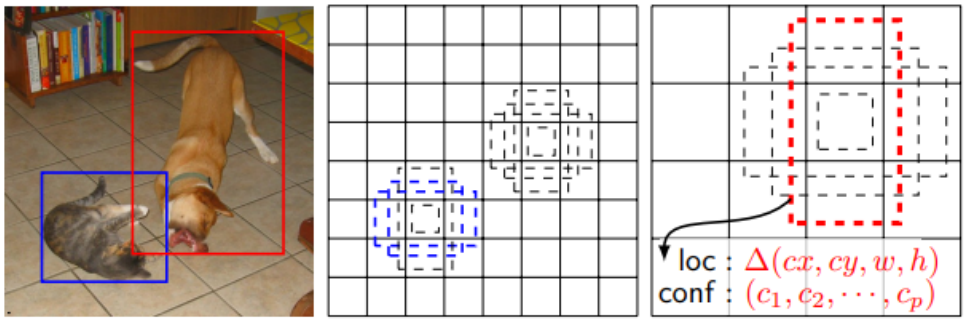


Figure 2.8: SSD [14] has several feature maps of different resolution and applies, for each of the positions in the feature map, a set of boxes of different aspect ratios. A prediction is done for each box applied. Image taken from [14].

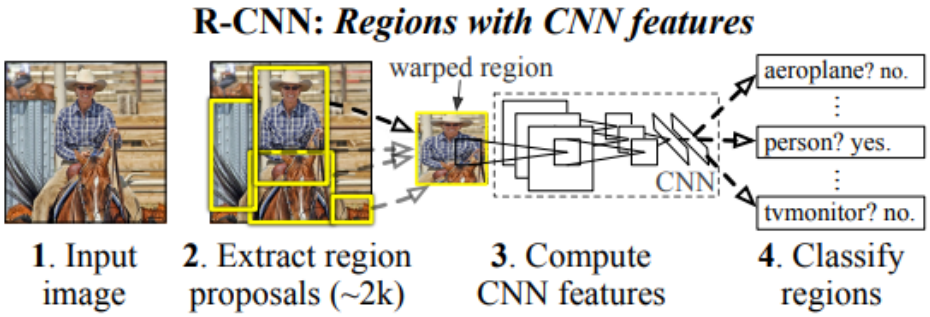


Figure 2.9: The steps in R-CNN [10], image taken from [10].

in 2014. In the first stage, R-CNN [10] generates region proposals that may contain objects and the second stage is a CNN that extracts feature vectors for each region, see Fig. 2.9.

An improvement to R-CNN [10] is Fast R-CNN [16] which speeds up R-CNN [10] by taking the whole image as input instead of each region. By inputting the whole image, Fast R-CNN [16] creates a feature map for the image and can then, for each proposal, use a part of the feature map instead of propagating a part of the image like in R-CNN [10], see Fig 2.10. Since propagation of images is costly, this leads to Fast R-CNN [16] being 9 times faster at training and 213 times faster at testing than R-CNN [10].

Faster R-CNN [17] is a further improvement and improves Fast R-CNN

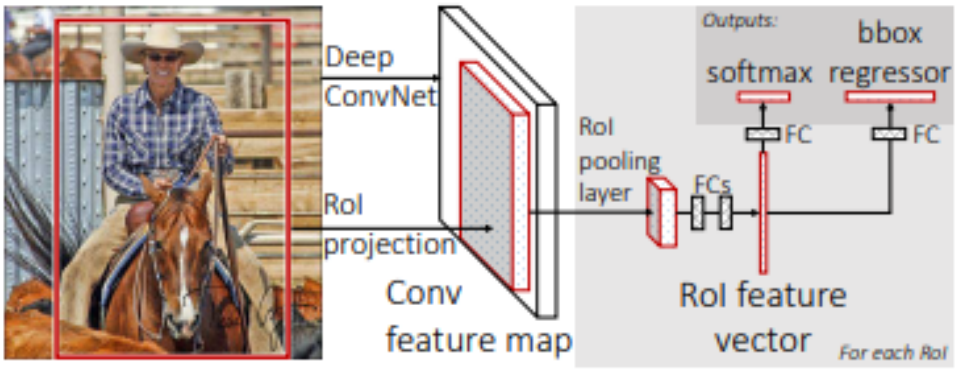


Figure 2.10: Fast R-CNN [16] creates a feature map for the image and then projects the region of interest (RoI) from the image to the feature map, image taken from [16].

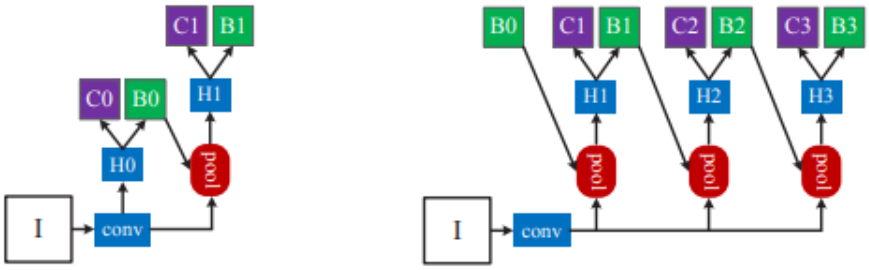
[16] by adding a Region Proposal Network (RPN) that shares full-image convolutional features with Fast R-CNN [16]. This makes the region proposals almost cost-free. Faster R-CNN [17] is one of the object detector with the best performance but is slow, the speed can be improved by limiting the number of generated proposals [15].

An alternative to Faster R-CNN [17] is Region-based Fully Convolutional Networks (R-FCN) [18], since this network is fully convolutional it is faster than Faster R-CNN [17] but generally not as accurate [15].

A quite different and recent object detector named Grid R-CNN [19] uses a grid guided mechanism to localize the bounding box instead of box offset regression as in Faster R-CNN [17]. In their paper, they show that this method outperforms Faster R-CNN [17] for a strict Intersection over Union (IoU) threshold above 0.8, for more information about IoU see section 3.1.4.

2.6.3 Multi-stage detectors

Cascade R-CNN [20] is an extension to existing two-stage object detectors and reduces the overfitting at training. This is done by using similar architecture for the bounding box regression and classification several times, and each time the output bounding box from the previous stage is used as the input to the next stage. This allows Cascade R-CNN [20] to progressively improve the bounding box. The sub-networks, called heads, that do the bounding box regression and classification of Cascade R-CNN [20] are trained with increasing IoU threshold in the order of the sequence, i.e. if there are three heads the



(a) Faster R-CNN [17] (b) Cascade R-CNN [20]

Figure 2.11: Showing the achitecture of Faster R-CNN [17] and Cascade R-CNN [20] where "I" is input image, "conv" backbone convolutions, "pool" region-wise feature extraction, "H" network head, "B" bounding box, and "C" classification. "B0" is proposals in all architectures. Image taken from [20].

first one might be trained with IoU=0.5, the second IoU=0.6 and the third with IoU=0.7. This helps to progressively refine the predictions. As seen in Fig. 2.11b this architecture does no longer consist of two stages, hence it is called a multi-stage detector and is, as expected, slower than detectors with fewer stages. The architecture of Faster R-CNN [17] and Cascade R-CNN [20] are compared in Fig. 2.11.

2.7 Small Object Detection

Small object detection, as the name suggests, is the problem of detecting small objects. This is harder than detecting larger objects due to several reasons, e.g. few pixels in the image belongs to the object and therefore not much information is available. Architectures usually uses a fixed input size that is smaller than the image and then even less information is left, and the convolutional networks used, usually use pooling which negatively affects small object because even more information is removed. To improve the performance of small object detection one or a combination of methods can be used. Below some techniques will be described that can be used to improve the detection of small objects.

One technique that is suitable when only a few training images contain small objects or the images with small objects contain too few small objects, is oversampling and augmentation of the data to improve the performance. This is done in [21] where they used a copy-paste strategy for augmentation, involving picking small objects in the image and pasting them at random locations.

They achieved the best results when combining copy-pasting with oversampling, but since more focus was put on the small object the performance for medium and larger objects suffered. Another approach of changing the inputs is to increase the resolution of the image. In [22], a deep learning method for increasing the resolution of images is proposed. As mentioned above, architectures usually have a fixed input size for the images. This means that this method probably does not work when the images need to be down-scaled, because the information gain from this method will be lost when down-scaling the images.

A different way of dealing with small objects is to change the loss function to capture the small objects better. One approach of changing the loss function, for one-stage object detectors, is dynamically scaled cross-entropy loss, which was introduced in [23] where they created a new one-stage detector called RetinaNet [23] with the new loss function. The more focus is set on hard negative examples by weighing down the easy ones. They discovered that extreme background-foreground imbalance in one-stage detectors is a big part of why a one-stage detector is not as good as a two-stage detector. Even if this method is not designed to capture small objects it can be assumed that it would work for small objects, because small objects should be the hard negatives.

Another way to deal with this problem is to include a featurized image pyramid. A featurized image pyramid is obtained by first scaling the image to a different resolution, i.e. creating an image pyramid, and then, for each scale, convolving with a filter to get several feature maps, see Fig. 2.12a. This help detecting both small objects in images with high resolution and large objects in images with low resolution. However, constructing a feature pyramid by scaling the image is costly. Feature pyramid network (FPN) [24], uses the natural structure of a convolutional network to build a pyramid, which reduces the computational time while preserving the benefits of a featurized image pyramid, see Fig. 2.12b. Fig. 2.12 shows the difference between an featurized image pyramid and an FPN [24].

Yet another approach is NAS-FPN [25], where neural architecture and reinforcement learning is used to search for the best way to fuse the different levels of the FPN [24] such that the accuracy is maximized.

In object detectors, anchor boxes are used to generate object proposals. The anchor boxes have different shapes and sizes and will be detecting objects with similar size the best, e.g. a standing person will be best detected by a long rectangle while a ball with a square. By adjusting the anchor boxes you can change what object the model will detect, by reducing the size of the anchor boxes the model will be better at detecting smaller objects, but will, of course,

(a) Featurized Image Pyramid (b) Feature pyramid network [24]

Figure 2.12: In a feature pyramid network [24], the pyramid is constructed as the output of layers of different scales in the network. The outputs are, in the top-down path, merged with up-scaled higher feature pyramid levels. This leads to higher accuracy than just using the feature pyramid levels without merging. Image taken from [24].

reduce the performance for larger objects. Another way of adjusting the anchor boxes is to dynamically learn the anchor shapes, which was done in [26] where they saw an significant improvement compared to baseline methods on many benchmark datasets.

Path Aggregation Network (PANet), proposed by [27], boosts the information flow by augmentation and aggregates the paths. The framework consists of an FPN [24], an additional bottom-up path, and then adaptive feature pooling. Since small objects are detected at the lower levels of the FPN [24], propagating these features should help the detection of small objects, whereas other methods that improve the accuracy of small object detection, worsen the performance for large object detection, PANet [27] improves the accuracy of detecting objects of all sizes. Since lower levels of the FPN [24] contain more accurate location information, due to their smaller receptive fields, propagating low-level features thus help the localization of objects, which is beneficial for all kinds of object detection. The architecture of PANet [27] is shown in Fig. 2.13.

Perceptual GAN [28] is another interesting method that improves small object detection, the method generates enhanced representations for small objects. The idea is that the representation of large and small objects differ. While the representations of large objects are easier to detect and distinguish, due to the higher resolution, the representation of small objects have lower resolution which makes the detection and classification of objects harder. By using a GAN they enhance the low-resolution representations of small objects to be more similar to the representations of larger objects.

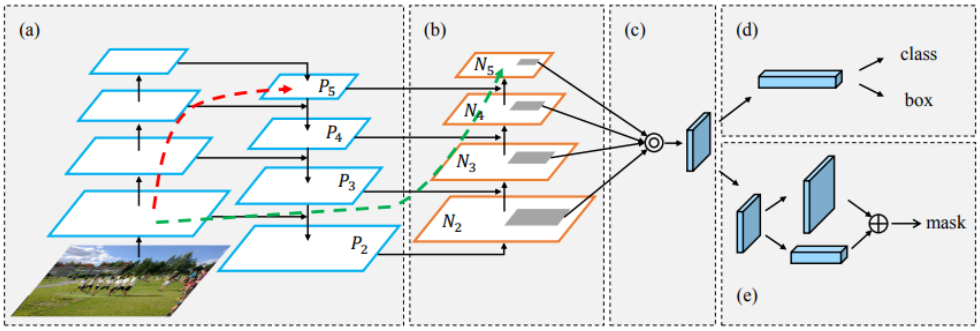


Figure 2.13: PANet [27] contains (a) an FPN [24], (b) an additional bottom-up path and (c) adaptive feature pooling. (d) is used for object detection and (e) for instance segmentation. The red and green arrow illustrate the propagation of low-level features. Image taken from [27].

2.8 Instance Segmentation

Instance segmentation is the process of determining what pixels belong to what object. This includes detecting the object (classify and localize) and then create a mask for the object. This is an extension to object detection and is harder and computationally heavier. As with object detection, there are mainly two types of methods; one-stage and two-stage methods.

Mask R-CNN [29] is a two-stage detector that extends Faster R-CNN [17] by adding a new branch for creating masks. This architecture has been very successful and has been used for many other extensions, Mask R-CNN [29] is shown in Fig. 2.14. One extension is called Mask Scoring R-CNN (MS R-CNN) [30], where an additional branch for predicting a score for the quality of the mask is added. MS R-CNN [30] then includes this score in the loss function. This leads to improved performance and is the first framework to consider the correctness of the mask and not just the classification score, which can be misleading. Mask R-CNN [29] is even used in PANet [27] described above.

Since instance segmentation is such a hard problem there is barely any framework that operates in real-time. However, You Only Look At CoefficienTs (YOLACT) [31] is one framework that do instance segmentation in real-time. To be real-time YOLACT [31] splits the instance segmentation onto two parallel tasks, where one uses a fully convolutional network (FCN) to calculate a set of prototype masks and the other predicts mask coefficients. The prototypes are then combined according to the coefficients from which the result becomes the detection. YOLACT [31] achieves 40 frames-per-second (FPS),

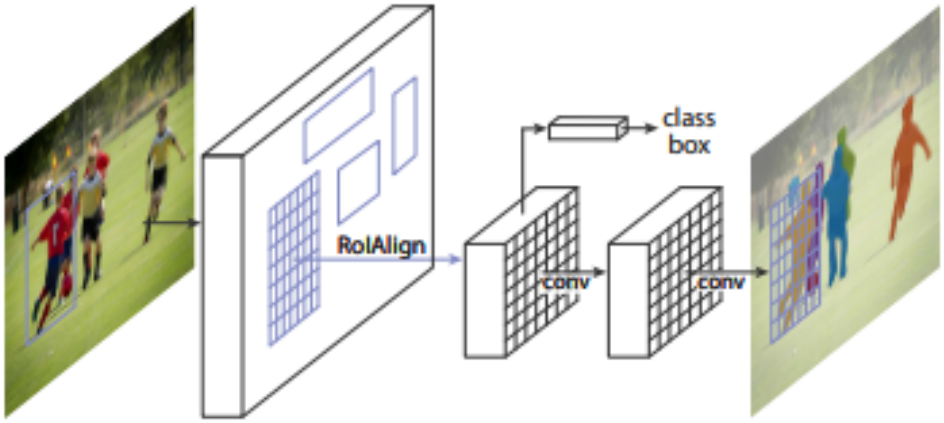


Figure 2.14: Mask R-CNN [29] is similar to Faster R-CNN [17] but with a new branch for predicting masks. Image taken from [29]

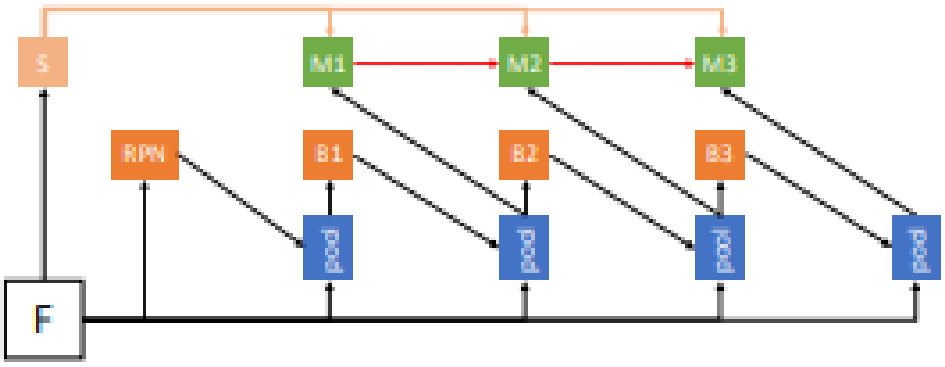


Figure 2.15: Hybrid Task Cascade [33]. Image taken from [33]

but the performance is worse compared to a two-stage method like Mask R-CNN [29].

The leader in the Common Object in Context (COCO) instance segmentation test-dev dataset [32] was, in the beginning of this project, Hybrid Task Cascade (HTC) [33]. HTC [33] combines the bounding box regression and mask prediction instead of performing them in parallel, adds a semantic branch and fusing it with box and mask branches to explore more contextual information, and incorporates a direct path to reinforce the information flow between mask branches. The structure of HTC [33] is kind of similar to Cascade R-CNN [20] in a way that uses an existing framework but adds new paths and more stages to get better results, see Fig 2.15.

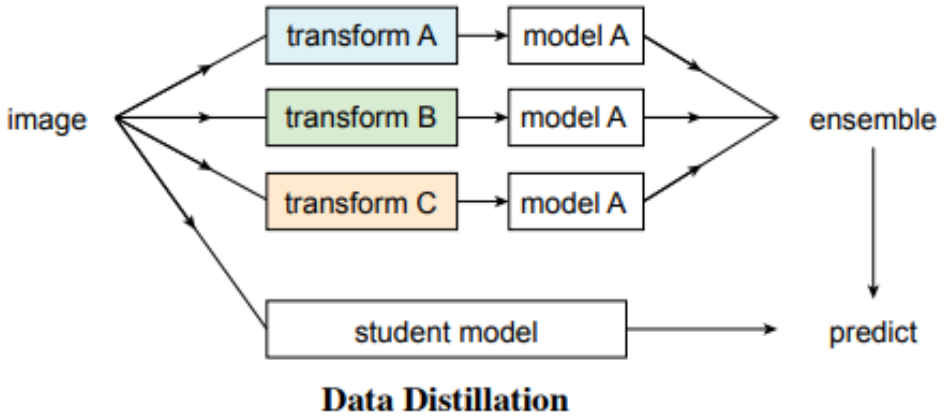


Figure 2.16: Data distillation [34] combines the output of differently transformed unlabeled images and then uses it to train the model. Image taken from [34].

2.9 Semi-supervised learning

Semi-supervised learning is the situation when you have labels for part of the data but not all and you train the model with both the labeled and unlabeled data, usually the unlabeled part is much smaller than the labeled. In general, the performance of the unlabeled and labeled data will be lower bounded by the performance you would get if you only used the labeled data and upper bounded if you labeled the unlabeled data, as is the case in [34]. This is reasonable because the labeled data contains more information. The case described in [34] is called Omni-supervised learning, which is when you use all possible labeled data and then use additional unlabeled data to push the performance further. This is different from semi-supervised learning because in semi-supervised learning you could use part of your labeled data as the unlabeled data. In [34] they propose a method called data distillation. In data distillation [34], the model is first trained on labeled data, then performs inference on unlabeled data with different transforms and then combine the outputs with an ensemble method, the combined output is then used as labels to retrain the network. The process is shown in Fig. 2.16.

As stated in [35], there are two main categories in semi-supervised learning for deep learning image classification, one that changes the loss function to better incorporate unlabeled data and the other that creates pseudo-labels, i.e. labels that are treated as true, for the unlabeled data. Since the methods from the two categories have different approaches they should not interfere and it

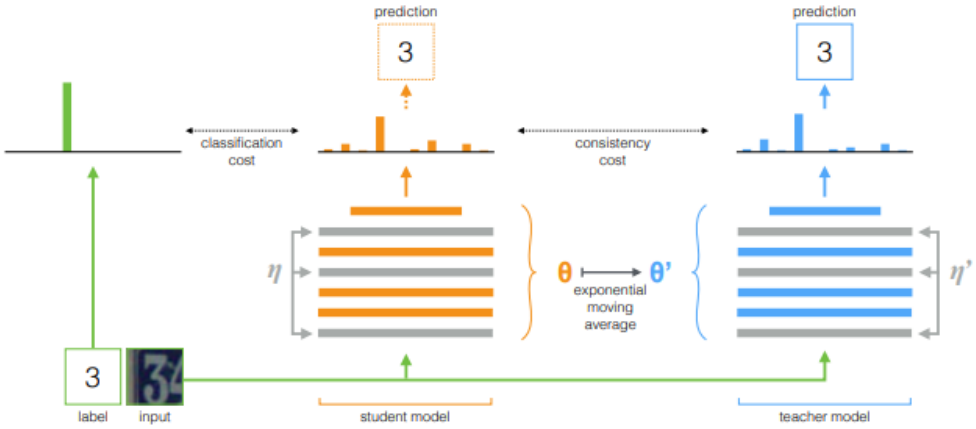


Figure 2.17: In Mean Teacher [36] η and η' are noise applied within the computations of the student and the teacher. Image taken from [36].

is possible to combine techniques from the two categories. The method used in [35] is called label propagation. It uses a neighbor graph to propagate the labels from labeled to unlabeled data. The labeled unlabeled data are used to train the network and new labels are generated before each iteration. They also mentioned that the predictions of diffusion, i.e. label propagation [35], are consistently better than network predictions, e.g. the method used in [34].

Mean Teacher (MT) was proposed in [36] which is based on the teacher-student framework where the teacher is a good network and the task is to learn the student network from the teacher. As the name suggests they use the mean of teachers to teach the student. The mean teacher is an exponential moving average of the student weights. The output of the student is compared with the output of the teacher and a consistency loss between the two is used as regularization, Fig. 2.17 shows the idea.

Even if networks are good at detecting and classify images, a small perturbation of the images that do not make a difference for humans can have a big impact on the output of the network. The perturbations that maximize the prediction error were given the name *adversarial examples* by [37]. One regularization method that takes advantage of this in semi-supervised learning is Virtual Adversarial Training (VAT) introduced in [38]. This method uses adversarial training which adds a regularization term that measures the divergence between the ground-truth and the estimated label distributions for the worst (adversarial) perturbation. Doing so increases the robustness of the model by reducing the sudden label changes. The difference between virtual and original adversarial training is that virtual uses the predicted labels as ground-truth

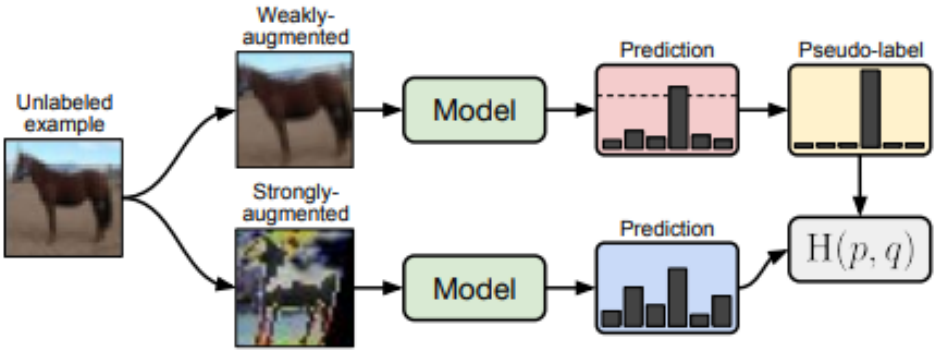


Figure 2.18: FixMatch [1] is a framework for semi-supervised learning. Image taken from [1].

in cases when the label is missing. The benefit of this is that this method can now be used with unlabeled data, and in semi-supervised learning, they call this new loss term *virtual adversarial loss*.

A recent semi-supervised approach for image classification is FixMatch [1], which augment the unlabeled images with weak and strong augmentation. The weak augmented images are propagated through the network and if the classification score is high enough they use this predicted label as a pseudo label for the strong augmented version. A cross-entropy loss between the predicted label and the pseudo label for the strong augmented images is added to the loss function, Fig. 2.18 shows the process of FixMatch [1].

2.10 Related work

Since this project involves many areas; instance segmentation, semi-supervised learning and small object detection there are many methods and models that are related. For example, almost everything in sections 2.5, 2.6, 2.7, 2.8 and 2.9 are related. However, there are a few things that are directly related, since they are being used in this project, and those are explained in more detail below.

2.10.1 Mask R-CNN

As mentioned in section 2.8, Mask R-CNN [29] extends Faster R-CNN [17] by adding a new branch for instance segmentation.

Backbone

The first part of Mask R-CNN [29] is the backbone network which produces the feature map. This network can be any network as long as it can produce a feature map. In [29] where Mask R-CNN [29] was introduced they used different versions of ResNet [5] and ResNeXt [6] as backbones.

Region proposal network

The next step, which is the first stage in a two-stage detector, is the RPN which generates proposal bounding boxes for the objects. The proposals are created by sliding a small network over the feature map produced by the backbone. For each position in the feature map the small network will predict k region proposals corresponding to the k anchor boxes used, which are centered at the current position. The small network uses a fixed number of filters of size $n \times n$ to get a one dimensional vector. This vector is then convolved with $2k$ and $4k$ 1×1 filters to get vectors with classification scores and bounding boxes for each of the k anchors.

Detection and instance segmentation

The second stage branches off to a detection and a instance segmentation branch. The proposals, also called regions of interest (RoIs), from the first stage are passed to the two branches, where the classes and bounding boxes are predicted in the detection branch and the binary masks are predicted in the instance segmentation branch.

RoIAlign

The size of the bounding boxes from the RPN can not always be evenly divided by the size of the feature map for the image. This lead, prior to Mask R-CNN [29], to quantization errors which may not be harmful to classification but has an impact on the masks. To solve this misalignment Mask R-CNN [29] has a layer called RoIAlign which creates the RoI by using bilinear interpolation.

2.10.2 ResNet

It is stated in [5], adding more layers a network leads to saturated accuracy and eventually the accuracy become worse, but that it is not caused by overfitting because the training error is also increased. A deeper networks should not have a higher training error because the function of the shallower network can

be expressed in the deeper, e.g. by setting the added layers to identity the deeper network will have the same mapping as the shallower. This indicates that deeper network are harder to train and the solver did not find a very good solution.

A residual block, see Fig. 2.5a, consists of a few network layers and a shortcut connection which takes input before the layers and connects it to the output. This changes the trivial solution from zero to identity and the idea is that this will help the solver find the better solutions for deeper networks. A ResNet [5] is a network that uses residual blocks and it is shown in [5] that using residual blocks help the solver find better solutions for deeper networks. For example, it was shown in [5] that a 34-layer ResNet [5] had lower validation error rates than a 18-layer ResNet [5] while the plain version, without residual blocks, had higher validation error rates for the 34-layer network compared to the 18-layer network.

2.10.3 Feature Pyramid Network

The layers of a network with different size will encode different things. Early layers of network, that have a larger size, have features corresponding to something local in the image, but early features can not be as complex are features in later layer of smaller size. Hence, these early features have high-resolution but are semantically weak. The opposite applies for features in later layers of smaller size. They have low-resolution but are semantically strong. To be able to detect something more complex like an object, semantically strong features are needed but to localize something accurate, the features with high resolution are needed. As mentioned in section 2.7, FPN [24] is a technique that uses an existing network and create a feature pyramid, an FPN [24] is shown in Fig. 2.12b. By using the layers of the existing network, a set of feature maps with decreasing size is obtained. This step is called the bottom-up pathway. In the top-down pathway, i.e. right part of Fig. 2.12b, the feature map of smallest size which is the semantically strongest, is upsampled to a larger size and merged with the feature map of the same size from the bottom-up pathway. This creates a feature map that combines semantically strong features with features of higher resolution. This procedure, of upsampling and merging, is continued a few times. The output of an FPN [24] is multiple feature enhanced maps, i.e. all feature maps from the top-down pathway.

In object detectors, the technique of FPN [24] is applied to the backbone network and the new feature maps is used for the RPN to detect RoIs. Object detection is improved by using an FPN [24] since the feature maps are

enhanced, but also because the RPN has more feature maps to use. However, using more feature maps will slow the model down. Therefore, there is a trade-off between using more feature maps which increases the performance, and less which is faster.

2.10.4 FixMatch

FixMatch [1], shown in Fig. 2.18, is a semi-supervised approach for classification which looks for consistency between the outputs of differently augmented images, as mentioned in section 2.9. The approach uses labeled data and unlabeled data, as semi-supervised approaches do, and obtains two losses; a supervised and a unsupervised. These losses are then combined as

$$\text{total loss} = \text{loss}_x + \lambda \cdot \text{loss}_u \tag{2.1}$$

where loss_x is the supervised loss, loss_u the unsupervised loss and λ is a weight that can be adjusted. The supervised loss is obtained by usual supervised learning. To get the unsupervised loss, the input images are first weakly augmented by, for example, flipping and shifting the image. The model then predicts the label of the weakly augmented image and if the classification score is above a predefined threshold, the prediction is used as a pseudo-label. Since the weakly augmented images are unlabeled, no information about the correctness of the prediction exists, which means that a pseudo-label might still be incorrect. If a pseudo-label is produced the model predicts the label for the strongly augmented image and the unsupervised loss is obtained as the cross-entropy between the predicted label for the strongly augmented image and its pseudo-label, [1].

Usually the labeled images are also weakly augmented which means that the weakly augmented unlabeled images can be treated in the same way as the labeled images.

Strong augmentation

The strong augmentation in FixMatch [1] is done with CutOut [39] and either RandAugment [40] or CTAugment [41].

CutOut [39] is a simple augmentation method that colors random areas in the image in a solid color.

RandAugment [40] is an augmentation method where images might, for example, be translated, sheared, and rotated. It has two parameters; the maximum number (n) of augmentations and the maximal severity to be used (m) of

the augmentations. RandAugment [40] starts by randomly selecting n transformation types which will be used with a 50% chance, i.e. if you set $n = 2$, 25% of the images will not be transformed at all. For each of the transformations to be done a random integer from 1 to m is chosen. If the random integer is 10 the maximum severity will be used, but if it is 1 only 10% of the maximum severity will be used. The maximum severities for the transformations have to be set. In FixMatch [1], the maximal rotation is 30 degrees. Where the negative transformation applies, there is a 50% chance that the transformation is the negative. For example, if the image is going to be rotated there is an equal chance to be rotated clockwise as counterclockwise.

CTAugment [41] is an augmentation method which also samples the augmentation type as in RandAugment [40]. However, CTAugment [41] learns the magnitudes of each augmentation type during training, which in RandAugment [40] have to be set manually.

In [1] they found that the augmentation types RandAugment [40] and CTAugment [41] have similar performance except in an extreme case where only four labels per class existed and CTAugment [41] performed better.

Chapter 3

Methods

To determine if semi-supervised learning can be used to improve the performance for instance segmentation of Telecom equipment, a baseline model was used to which the semi-supervised approach was compared. Mask R-CNN [29] was used as the baseline and an extension which makes it possible to use FixMatch [1] for instance segmentation for the semi-supervised approach. The models were trained and evaluated on a custom dataset containing Telecom equipment.

3.1 Dataset

There are no publicly available datasets containing Telecom equipment that looks like the Telecom equipment the final model should be able to detect. Therefore, a custom dataset was used. In this section, the format of the dataset and how the performance is evaluated is presented.

3.1.1 Common Objects in Context

Common Object in Context (COCO) [32] is a famous dataset for object detection, segmentation, and captioning. The dataset contains 330 000 images where over 200 000 are labeled and it has 80 object categories. The models we will be using are pre-trained on this dataset and we will be using their evaluation metrics, which are presented in section 3.1.4, to get results you more easily can compare with other models.

3.1.2 Custom dataset

The dataset used in this project contains images showing towers with tower equipment. The towers are usually located in the center of the image and span the full height of the image. The dataset contains 590 manually pixel-wise labeled images and additionally 1000 unlabeled. Some examples of images are shown in Fig. 3.2a, 3.2b, 3.2c and 3.2d. Due to privacy reasons the images have been cropped, the original images all have the resolution 3648×5472. Most of the images shown in Fig. 3.2a-3.2d are looking up at the tower but there are as many looking down and straight on, but those may not be shown.

Classes

The equipment included in the labeling are antennas, Radio Remote Units (RRUs), and occluded RRUs. The antennas look like a tall rectangle but different antenna-models have different shapes, with varying width. There are two main models of RRUs, one is larger with a more square side and grey color while the other is less square and white but both of them are smaller than the antennas. Fig. 3.2c has the larger RRUs at the top of the tower and the smaller RRUs further down. The antennas and RRUs are mainly white but have been painted on some towers. The reason for using a class for occluded RRUs is that RRUs are often partly hidden but it would still be useful to detect them. They were not included in the RRU class because the occluded RRUs can look quite different and it could be useful to distinguish between the classes. An occluded class of antennas was not included because antennas are easier to detect and usually when they are occluded a larger surface of the antenna can still be seen, i.e. they can still be recognized as an antenna. Due to labeling being so time-consuming other classes of equipment were not included in the annotations.

Training and test split

The performance the model achieves on the test set should be the performance the model has on unseen images, which means that the images in the test set should not be too similar to the images in the train set. In the custom dataset, different images of the same tower site could be similar, Fig. 3.1 shows two images of the same tower. Since images of the same tower site might be similar the training and test split were split according to tower sites, i.e. all images of the same tower are either in train or test.

The test set consists of 30% of the sites which corresponds to about 25%

(a) (b)

Figure 3.1: Two images from our custom dataset that were captured after each other.

of the images, this is only for the labeled images, the unlabeled images are only used for training. No validation set was used because the toolbox, see section 3.2.1, used does not support validation for single GPU training. Cross-validation is useful for getting a more accurate view of the performance, as the train and test split might not always be split in the best way, but due to training taking a long time cross-validation could not be used.

Problems

Probably the biggest problem with the custom dataset is that it contains other unlabeled equipment, similar to the labeled RRUs. Tower Mounted Amplifiers (TMA) are also white quite square-shaped boxes which can easily be confused with RRUs, Fig. 3.3 shows the difference between an RRU and TMA. Other problems are that there are background objects that can easily be confused for a Telecom equipment, e.g. containers and tall poles, and that there are more antennas than RRUs, i.e. not a balanced dataset.

3.1.3 Labeling

The labeling of the equipment was done in VGG Image Annotator (VIA) [42]. As mentioned in the previous section, only antennas, RRUs and occluded RRUs are included in the annotations, because labeling takes time and these types of equipment were judged to be the most important.

Masks

Usually, the masks for instance segmentation may have holes or separated parts belonging to the same object, but in the annotations for this dataset, the equip-

(a)

(b)

(c)

(d)

Figure 3.2: Images from our custom dataset. All images have the same resolution but due to privacy reasons the images have been cropped

Figure 3.3: The green circles mark the RRUs while the red the TMAs.

ment is labeled as one completely filled area, i.e. no separated parts nor holes. This is due to not knowing how an instance segmentation mask could be and the not so obvious instructions of VIA [42]. Since the equipment is labeled as one filled area there may be other objects inside the masks created for the equipment, e.g. cables crossing the equipment. Because some pixels of other objects may be included in the masks, a threshold of the amount of other object in the masks had to be decided. This is not a numeric threshold, i.e. other objects were included in the segmentation masks if the pixels on the other side of the objects contain valuable information for the equipment. Fig 3.4a shows examples where the labeled equipment has other objects in their masks. If too much of the equipment is covered with other objects no mask was created, Fig. 3.4b shows some examples.

Quality of the masks

When labeling the equipment the mounts holding the equipment was not included while extruding parts of the equipment was, see Fig. 3.5a and 3.5b. This was done to help the model learn masks that are more accurate on a pixel-level. After all, the model will not produce better masks than the mask it has learned from during training. By comparing the mask of our custom dataset with the mask in COCO [32], it is clear that our masks are more detailed, see Fig. 3.6. This means that the model trained on our dataset has the possibility to produce more detailed masks than if it was trained on the COCO dataset [32].

(a) The equipment is labeled as one filled area, there may be other objects in the masks but cannot cover the equipment too much. The three leftmost images are RRUs, the middle three occluded RRUs and the right is an antenna.

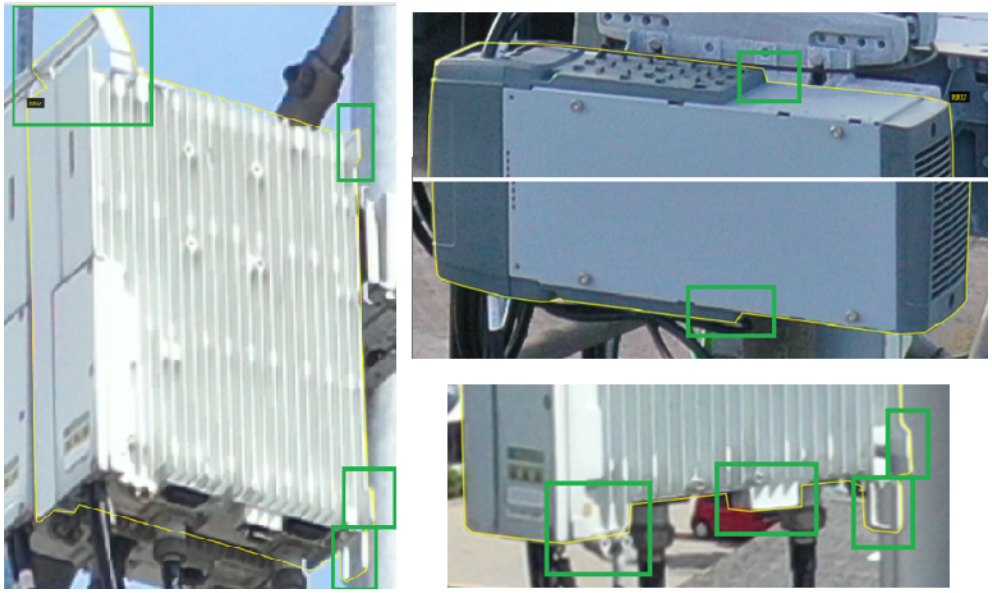
(b) This equipment was not labeled due to too much occlusion.

Figure 3.4: The equipment in the left figure was labeled while the equipment in the right was not.

Detection

The detection problem this dataset provides is not exactly the detection and segmentation of objects in general but rather detection and segmentation of objects fulfilling certain conditions. For example, the RRUs have very different sides. The RRUs that only show the largest side are included in the annotations but not when they are only showing the thinner side. This leads to a threshold orientation where masks for the equipment were stopped produced, Fig. 3.7a shows both sides of the threshold orientation for the RRU class and Fig. 3.7b shows the threshold orientation for antennas. The argument for not including all the orientation of the equipment in the annotations is that the views are so different and in the final product when this is used in combination with a drone, the drone can move around the tower and view the equipment from different angles, so the model will still have its chance to detect the object.

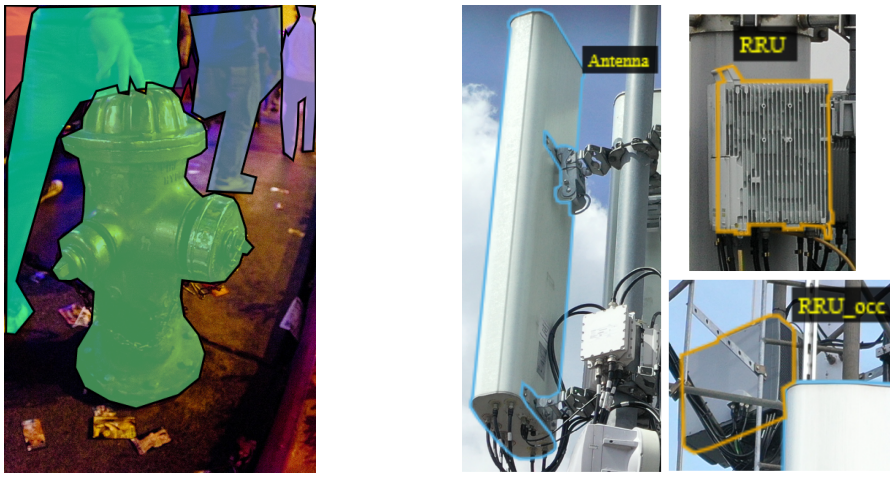
The size of the equipment is another threshold that needs to be decided. In this dataset everything was labeled no matter the size. Fig. 3.8a contains some small labeled equipment. In that resolution the RRUs are barely visible while in the original resolution the equipment is easy to recognize. Fig. 3.8b shows an example of larger equipment.



(a) When labeling the RRUs, the most characteristic extruding parts was included in the mask, see the green boxes.

(b) The mounts holding up the antennas are not included in the masks.

Figure 3.5: Extruding parts of the equipment are included in the mask while the mounts holding up the equipment are not.



(a) Mask quality of the COCO dataset [32].

(b) The mask quality of our custom dataset.

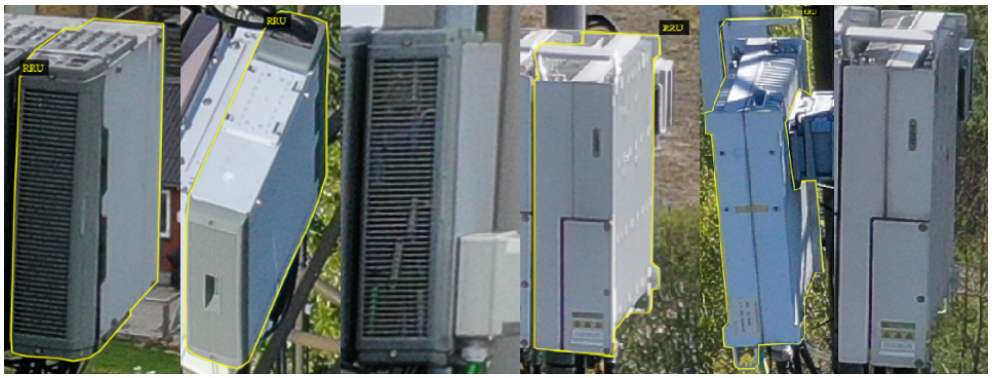
Figure 3.6: Comparison of the mask quality.

3.1.4 Evaluation

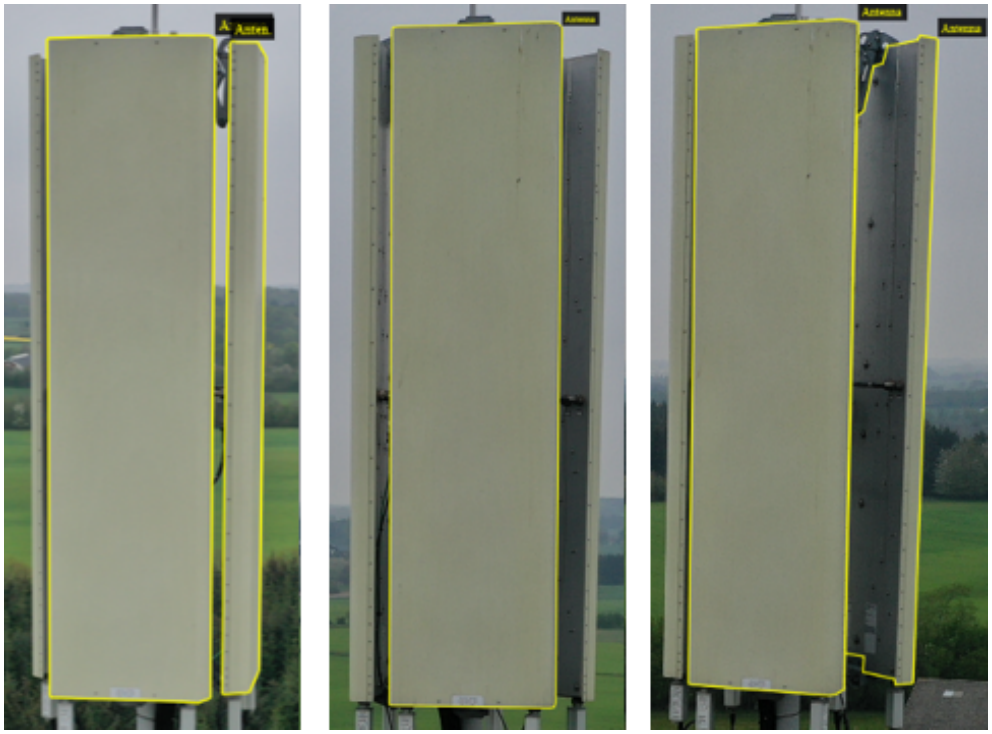
Detecting objects involves both classification and localization, where the model will output a class label and a bounding box describing each detection. In case of instance segmentation even a mask of the object will be output. However, for something to be considered detected we have to evaluate the detection.

Intersection over Union

Intersection over Union (IoU) is a measure to determine if the localization of the object is correct. IoU is what the name suggests, the intersection of two areas divided by the union of the areas. The value of IoU ranges from 0, where the areas are not overlapping, to 1, where the areas are identical. When evaluating a detection a hyperparameter of the IoU threshold has to be set. This parameter determines how much overlapping the areas have to be, to be considered a correct localization. Typical values of IoU ranges from 0.5 to 0.9. The IoU can either be calculated with the bounding boxes or the masks but since this project involves instance segmentation, the IoU will be calculated with the masks.



(a) The third and last RRUs (going from the left) will not be labeled because too little of the larger side is showing.



(b) The antennas with yellow borders have been labeled, the other antennas were judged to be too much hidden and are not included in the annotations.

Figure 3.7: The images show for what angle the equipment is not labeled.

(a)

(b)

Figure 3.8: Two images from our dataset with annotations.

True Positive, False Positive, True Negative and False Negative

When the model is evaluated, it is evaluated on each of the classes separately. In the following examples let the IoU threshold be 0.5, the IoU will be calculated with the masks and the model is evaluating its performance for RRUs. If the model predicts an area as an RRU it will be a True Positive (TP) if the IoU with an RRU from the annotations is above 0.5. It will be a False Positive (FP) if the IoU is below 0.5 for all RRUs in the annotation. A False Negative (FN) is a RRU in the annotation that was not detected. When evaluating the performance the True Negative (TN) predictions are not considered because every area that is not an RRU and the model do not predict it as an RRU is a TN. This means that there are a lot of TNs for any model.

Precision and Recall

Precision is kind of a measure of the quality of the detections and is defined as the number of correct detections divided by the number of detections the model did; the formula is shown in Eq. 3.1. Recall is the other usual measure of performance and recall determines how well the model detects what we want. Since TP + FN is the total number of ground truth objects we want to detect, recall is the number of correctly detected objects divided by the number of objects we want to detect, the formula is shown in Eq. 3.2.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{3.1}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{3.2}$$

When using a model for detection it will output several predictions, which in our case will be class labels, classification scores, bounding boxes and masks. By sorting the prediction with decreasing classification score you get a list with the most confident prediction at the top. If you then plot the precision versus recall as you include more and more of the sorted predictions, you get the prediction-recall curve. The recall will monotonically increase as more predictions are included, since there is no penalty for bad predictions, see Eq. 3.2. The precision will both increase and decrease but tends to zero as more predictions are included. In an ideal case both the recall and precision would be one.

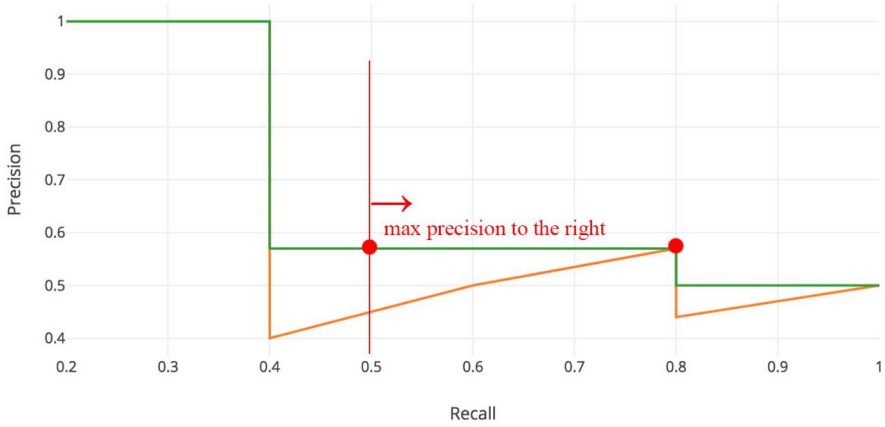


Figure 3.9: The area under the orange curve is the average-precision curve while the green is the smoothed version, from [43].

Average Precision

Average Precision (AP), or rather mean Average Precision (mAP) is the primary evaluation metric for COCO [32] and the metric used in this project. AP can be calculated in slightly different ways, e.g. interpolation or the exact area under the precision-recall curve. However, before interpolating or calculating the area under the curve, the precision-recall curve is usually smoothed. When smoothing the precision becomes the maximum precision for the current and higher recall values. Smoothing is reasonable because, in real applications, if the precision is lower for fewer predictions, then, of course, more predictions will be used to increase both the precision and recall. The number of predictions that cause the precision to have sub-optimal values would not be used in applications, hence the smoothed precision-recall curve only contains the values that would be used in applications. Fig. 3.9 shows the difference between AP and smoothed AP.

In COCO [32], AP is calculated with interpolation, as the mean precision of 101 recall values from the precision-recall curve, this is for one IoU threshold. The AP is then averaged over different IoU values, 10 IoU values are used starting at 0.5 and, with step size 0.05, goes to 0.95. As mentioned above, each class is evaluated alone, the final mAP is obtained by taking the mean of the averaged AP for each class. The final mAP is the precision averaged over 101 recall values, averaged over 10 IoU values and averaged over all classes.

3.2 Model

In this section, the toolbox used, the baseline model, and the extension of FixMatch [1] is discussed.

3.2.1 MMDetection

MMdetection [44] is an open-source toolbox for object detection in PyTorch, the toolbox contains many state-of-the-art models and evaluates the models according to mAP as in COCO [32]. The reason for choosing this toolbox is its large library of models which makes it easier to compare models since they can be evaluated and trained by the same toolbox.

3.2.2 Baseline

The baseline was chosen to be Mask R-CNN [29] with ResNet-50 [5] as the backbone. The Mask R-CNN [29] uses an FPN [24] which outputs 5 feature maps of different scales. There are more complex models with better performance than Mask R-CNN [29] that could have been used as the baseline, but since those models are more complex their training times are longer. The speed is not the primary objective but to be able to perform a lot of tests a faster model is appreciated, which means that the lower speed of more complex model outweighs its boost in performance. Similar argument can be made for choosing ResNet-50 [5].

3.2.3 Losses

The are many losses that can be used in a model for instance segmentation. Below the losses used for class labels, bounding boxes and masks are presented, i.e. these computes the supervised loss. All these losses are the default losses in MMDetection [44] for Mask R-CNN [29].

Class labels

For classification, cross-entropy was used as the loss, see Eq. 3.3.

$$loss(x, class) = -\log\left(\frac{\exp(x[class])}{\sum_j \exp(x[j])}\right) \tag{3.3}$$

where x is the vector of classification scores and $class$ is the index corresponding to the true class label index. The loss is infinite if the classification score

of the correct class is zero and is zero if the classification score of the right class is one.

Bounding boxes

Smooth L1 loss, defined in Eq. 3.4 was used for calculating the localization loss of the bounding boxes.

$$\begin{aligned} loss(b, gt) &= \sum_i L_1^{smooth}(b_i - gt_i) \\ L_1^{smooth}(\Delta) &= \begin{cases} \frac{\Delta^2}{2\beta} & \text{if } \Delta < \beta \\ |\Delta| - \frac{\beta}{2} & \text{else} \end{cases} \end{aligned} \tag{3.4}$$

where $b = \{x, y, h, w\}$ is the predicted bounding box, gt the ground truth bounding box and β a parameter that needs to be set. In the RPN of Mask R-CNN [29] $\beta = 0.11$ and in the second stage of Mask R-CNN [29] $\beta = 1$.

Masks

Binary cross-entropy loss with logits was used to compute the loss for the masks. The loss is shown in Eq. 3.5.

$$\begin{aligned} loss(m, gt) &= mean(L) \\ L &= \{l_1, \ldots, l_N\} \\ l_n &= \begin{cases} -\log(\sigma(m[n]) & \text{if } gt[n] == 1 \\ -\log(1 - \sigma(m[n])) & \text{else (i.e.} (1 - gt[n]) == 1) \end{cases} \end{aligned} \tag{3.5}$$

where σ is the sigmoid function, m is the predicted mask and gt the ground truth mask. A loss is computed for each pixel and the mean of the losses creates the total loss. Mean in not part of binary cross-entropy loss with logits but is used by default in the toolbox for Mask R-CNN [29].

3.2.4 Semi-supervised learning

The semi-supervised approach implemented is an extension of FixMatch [1]. The reason for choosing FixMatch [1] is that it consistently achieved better performance than other semi-supervised approaches like Mean Teacher [36], which was shown in the paper [1]. However, no comparison between FixMatch [1] and Virtual Adversarial Training (VAT) [38] was done in the paper.

By comparing the results from the respective papers it can bee seen that FixMatch [1] achieves lower error rates than VAT [38], therefore FixMatch [1] was chosen as the method used for semi-supervised learning.

The datasets used for evaluating FixMatch [1] and Mean Teacher [36] are the datasets from Canadian Institute for Advanced Research (CIFAR-10 and CIFAR-100 introduced in [45]) and the Street View House Numbers (SVHN) dataset introduced in [46]. These datasets are for classification only and at least CIFAR [45] is a balanced dataset, our custom dataset is not.

Data augmentation

When strongly augmenting the images FixMatch [1] uses a method called RandAugment [40] where images might be translated, sheared, and rotated. This is not a problem for classification where a classification label for the whole image is produced. This becomes a problem in object detection and instance segmentation where bounding boxes and masks are created, which depend on the location of the objects in the image. The bounding boxes and masks predicted for the weakly augmented image can not directly be used as ground truth for the strongly augmented image. The bounding boxes and masks need to undergo the same translation, shearing, and rotation as the strongly augmented image before they can be treated as ground truth. This is the extension we have done, i.e. to transform the bounding boxes and masks to enable FixMatch [1] for object detection and instance segmentation. In Fig. 3.10 the weak and strong augmentation are shown where the bounding boxes and masks have been transformed like the image.

Another difference from FixMatch [1] is that our extension is not using CutOut [39]. CutOut [39] can easily be used for classification where the whole image displays a single object, but to do this in object detection and instance segmentation you would have to use your bounding boxes and masks predictions and then randomly cut out a piece from the inside of them. We choose not to include this because the dataset already provides objects that are not fully visible, e.g. hidden behind other equipment or the tower. Fig. 3.11 shows how an image could look after CutOut [39], of course, the size of the area can be adjusted.

Hyperparameters

In total the original FixMatch [1] has 3 hyperparameters; the weight λ in Eq. 2.1, the amount of unlabeled images per labeled image, i.e. how much unlabeled data should be used and the threshold for how high the classification

(a) Weak augmentation

(b) Strong augmentation with transformed bounding boxes and masks. The TMA at the bottom right of the tower is not a detection, the color is due to the augmentation.

Figure 3.10: The upper figure shows the weakly augmented image and the bottom shows the strongly augmented image.

Figure 3.11: In CutOut [39] a piece of the image is removed as part of the augmentation.

score has to be for the weakly augmented image to be used as ground truth. If RandAugment [40] is used then more hyperparameters exist; n, m and all the maximum severities for each of the transformations, as explained in section 2.10.4.

Extension

The consistency loss in FixMatch [1] is computed in the same way as the supervised loss for the baseline, i.e. cross-entropy for the labels and masks and smooth L1 for the bounding boxes, see section 3.2.3.

Our extension comes with one additional hyperparameter. Sometimes during translation, shearing and rotation parts of the objects get moved outside the image and are lost when cropping the image to its original shape. The new hyperparameter determines how much of the object must remain inside the image before removing it as a ground truth label.

In the extension, we have set the model in evaluation mode and do not compute the gradients when the consistency loss between the weakly and strongly augmented images are computed. This means that the value of the loss is the only output and that backpropagation can not be done for this loss. However, since the loss obtained for supervised learning and the loss for unsupervised learning are combined as in Eq. 2.1, gradients still exist but only for the supervised part. Our reasoning for this is that we are interested in producing the best

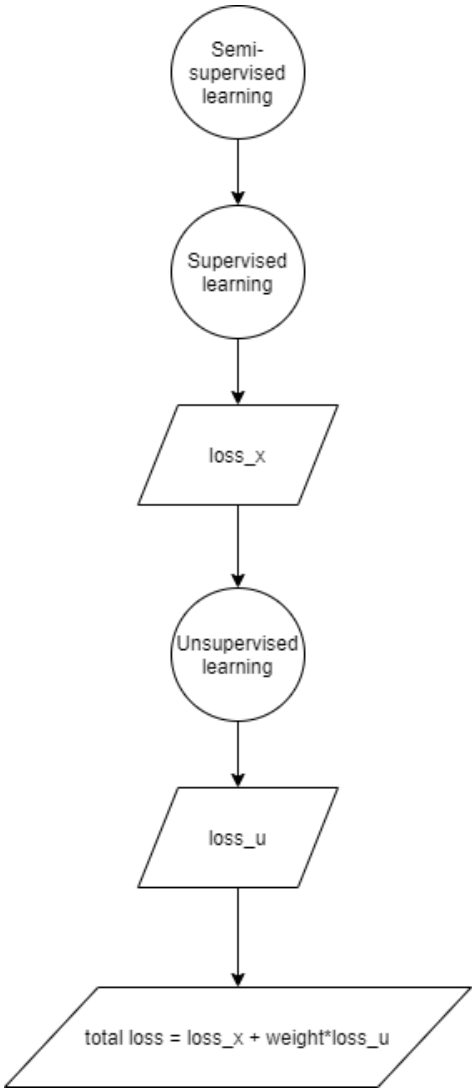
labels for the unlabeled data. Techniques like dropout and batch-norm that are used in the training-mode are useful when you already know the labels and help to create a more robust model. However, when the labels are unknown those techniques prohibit the model from predicting at its best. The unlabeled loss can be considered a consistency loss for the model's evaluations. As long as the model is consistent with its predictions for the weakly and strongly augmented image, the loss is zero, even if it is consistently wrong. The idea is that the gradients from the supervised part will lead the model in the right direction while the loss from the unsupervised part will provide a better value of the loss function which should lead to a better solution. The workflow of the extension can be seen in Fig. 3.12a, 3.12b and 3.13.

3.2.5 Training

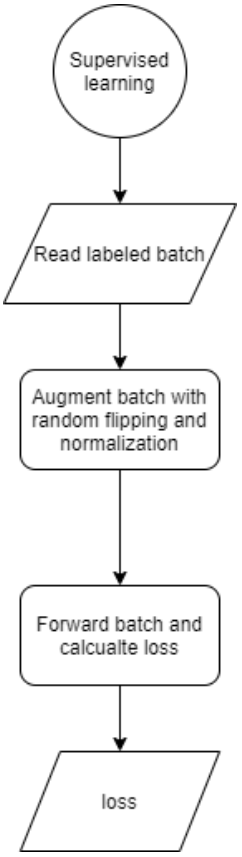
Since there is not validation set, validation during training could not be done. Instead all model were trained for twelve epochs and then evaluated on the test set for every epoch. If the performance of the model already converged after twelve epochs the training was stopped. Otherwise, the model was trained further until convergence. After convergence the epoch with the best performance was used to represent the model.

3.2.6 Hardware

The hardware used in this project are; Nvidia Tesla K80, Intel Xeon E5-2690 v3 @ 2.6 GHz and 55 GB RAM.



(a) The semi-supervised workflow used in FixMatch [1] is to first calculate the supervised loss like in Fig. 3.12b and the unsupervised loss as in Fig. 3.13. The losses are then combined with a weight for the unlabeled loss.



(b) The process of supervised learning. First a batch of images is read, then possibly a light augmentation is done and finally, the loss is calculated.

Figure 3.12: The left figure shows the workflow of the extension and the right figure shows the workflow of ordinary supervised learning.

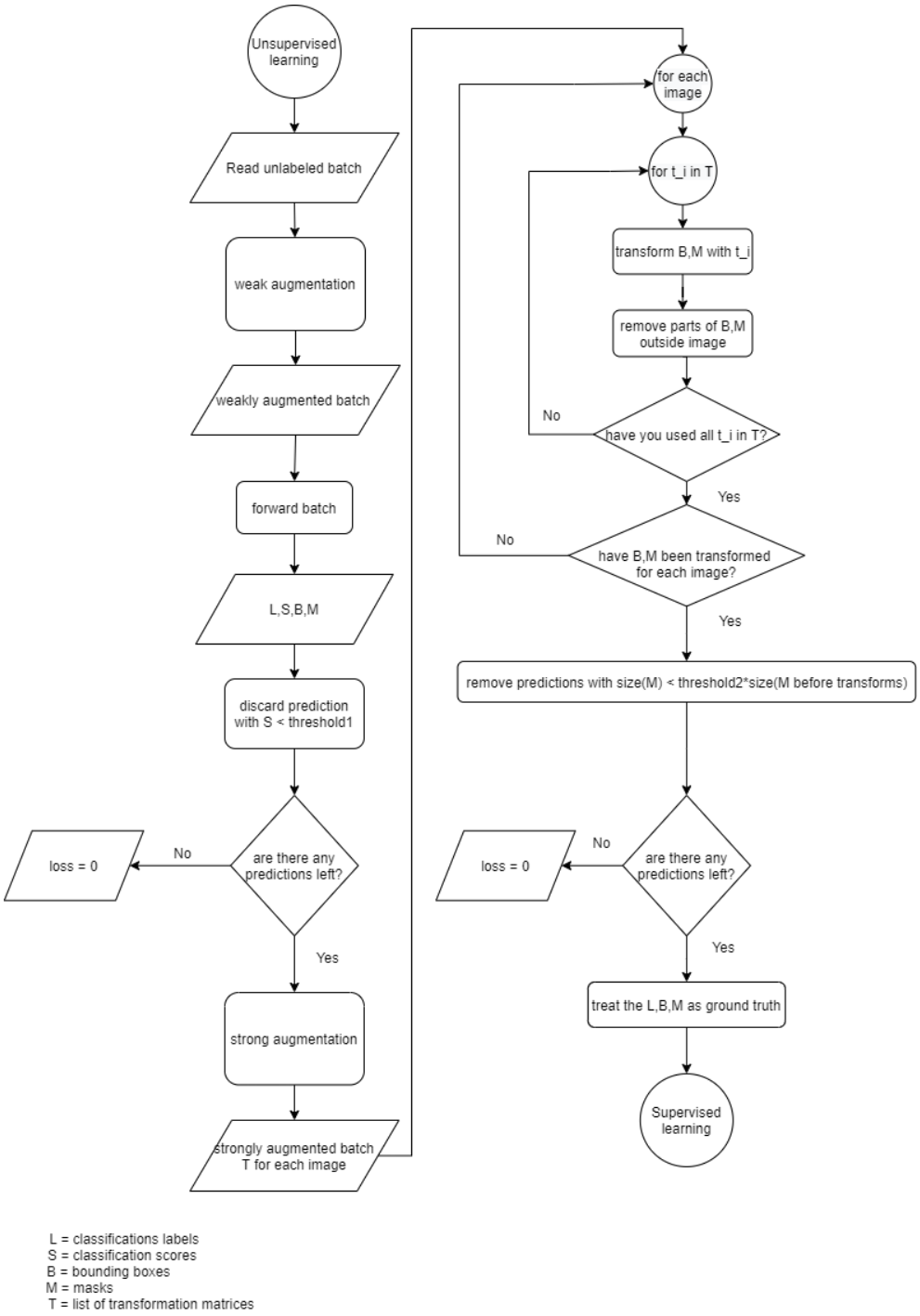


Figure 3.13: The workflow of our extension of FixMatch [1]. The idea is to transform the bounding boxes and segmentation masks and remove bad transforms. In the end, the supervised process in Fig. 3.12b is used where the transformed bounding boxes and segmentation masks are treated as ground truth. "threshold2" is the *inside* threshold.

Chapter 4

Results

A grid search was done to find the best combinations of parameters for the extension. The grid search was performed on the following parameters:

λ: the weight for the unlabeled part, see Eq. 2.1
score: the classification score threshold
inside: the new parameter that determine how much of an object that can be lost during the transformations
n: the maximum number of transformations
m: the maximum severity to be used for the transformations

with the main focus on the parameters directly related to FixMatch [1], i.e. λ $score$ and $inside$. The parameter setting with the best performance is $\lambda = 0.2$, $score = 0.9$, $inside = 0.5$, $n = 2$ and $m = 10$. The amount on unlabeled data was held constant to two times the amount of labeled data and the same values as in FixMatch [1] was used for the maximum severity of the transformations in RandAugment [40]. Unless stated otherwise the values above have been used.

The results for the extension with the best parameter settings are shown in Table 4.1, where the same parameter settings has been used five times. The baseline was also evaluated five times, this is shown in Table 4.2.

extension	run nr. 1	run nr. 2	run nr. 3	run nr. 4	run nr. 5	mean
mAP	0.542	0.533	0.532	0.548	0.544	0.540

Table 4.1: The best mAP for the best parameter settings of the extension when training from the same starting point five times.

The grid searches of the different hyperparameters are presented in Table 4.3. In Fig. 4.1a, 4.1b and 4.1c the mAP from Table 4.3a, 4.3b and 4.3c

baseline	run nr. 1	run nr. 2	run nr. 3	run nr. 4	run nr. 5	mean
mAP	0.529	0.532	0.541	0.532	0.518	0.530

Table 4.2: The best mAP for the baseline when training from the same starting point five times.

λ	0.1	(0.2)	0.3	0.4	0.5	0.6	0.7
mAP	0.534	0.540	0.531	0.529	0.544	0.527	0.524

(a) mAP for different values of λ.

$score$	0.3	0.4	0.5	0.6	0.75	0.8	(0.9)	0.95
mAP	0.531	0.537	0.535	0.527	0.516	0.534	0.540	0.53

(b) mAP for different thresholds of the classification score.

$inside$	0.2	0.3	0.4	(0.5)	0.6	0.75	0.8	0.9
mAP	0.511	0.545	0.542	0.540	0.533	0.539	0.531	0.523

(c) mAP for different values of the *inside* threshold.

n	(2)	3	4	5
mAP	0.540	0.523	0.527	0.533

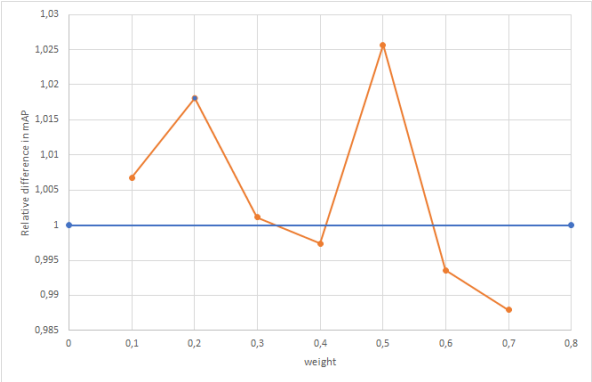
(d) mAP for different maximum number of transformations (n).

m	5	8	(10)
mAP	0.534	0.524	0.540

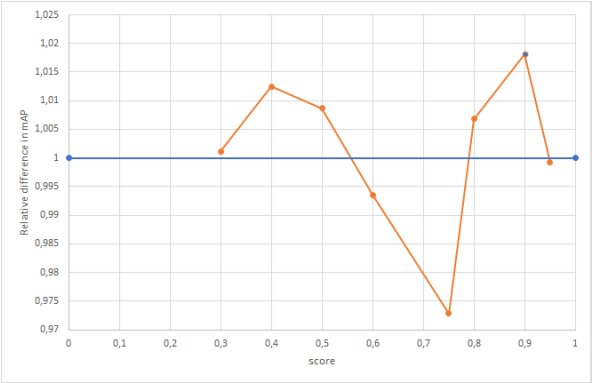
(e) mAP for different maximum severity of the transformations (m).

Table 4.3: Grid searches showing how the parameter choices affects the performance. The parameter settings in parentheses is the mean from Table 4.1, i.e. the mean performance of five runs, while the other values are obtained by training the model once.

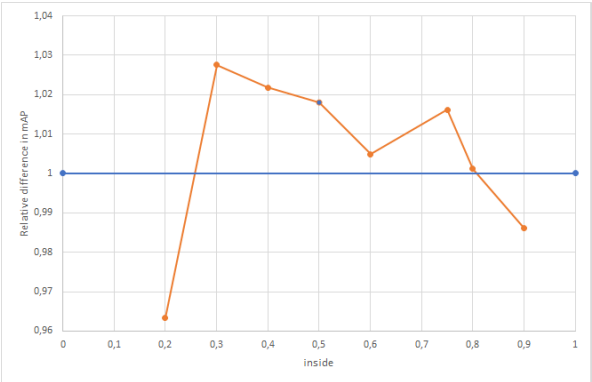
have been divided by the mean of the baseline in Table 4.2. This was done to see how the parameter settings affect the relative improvement compared to the baseline. As can be seen from the figures, the results are not particularly sensitive to any of the parameters.



(a) The mAP for different λ values was divided by the mean mAP of the baseline.

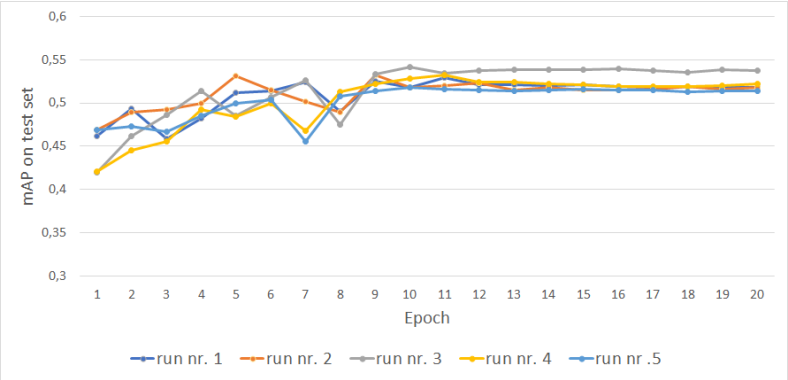


(b) The mAP for different classification score thresholds was divided by the mean mAP of the baseline.

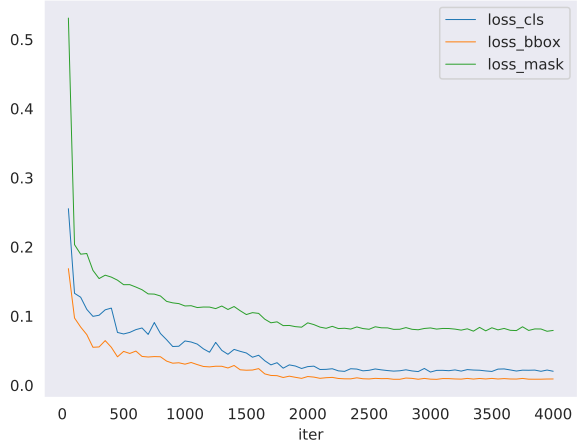


(c) The mAP for different values of the *inside* parameter was divided by the mean mAP of the baseline.

Figure 4.1: The figures show the relative improvement compared to the mean baseline for the grid searches in Table 4.3a, 4.3b and 4.3c. The blue filled datapoints are the mean performance of the extension in Table 4.1. The blue horizontal line indicates where the performance is as good as the mean of the baseline in Table 4.2.



(a) mAP on the test set for all five runs of the baseline model.

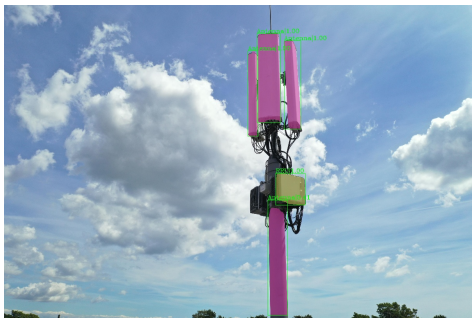


(b) Losses during training for baseline run nr. 2. There are 200 iterations in each epoch, i.e. $200 \cdot 20 = 4000$ iterations in total.

Figure 4.2: Performance and losses during training for the baseline.

The test performance for the baselines in Table 4.2 during training is shown in Fig. 4.2a and loss during training of the second baseline (run nr.2) in Table 4.2 is shown in Fig. 4.2b.

The quality of the predictions can be seen in Fig. 4.3 where images have been selected for which the model performed well or not so well. The predictions shown in the images are the predictions with a classification score above 0.3. The model used, has an mAP of 0.528 on the test set and the images in Fig 4.3a-4.3b are from the test set, meaning that the segmentation masks seen in the images were used to calculate the mAP.



(a) All the equipment was detected but the model also thought the tower was an antenna, but the classification score was only 0.41 which is good.

(b) All labeled equipment was found and no false detections.

(c) The model detects most of the equipment, the antenna at the bottom right of the tower was in the annotations while the detected RRU above was not.

(d) The model detects all the equipment but also detects some TMAs as RRUs (the two yellow masks at the bottom). The antenna at the top left of the tower was not in the annotations.

Figure 4.3: Images showing the segmentations for a model with mAP of 0.528. The objects detected as antenna are pink, the RRUs are yellow and the occluded RRUs are purple.

Chapter 5

Discussion

In this section the results, our extension of FixMatch [1], the custom dataset, our labeling decisions and semi-supervised learning are discussed.

5.1 Interpretation of evaluation scores

Since only training and test sets exist, the scores on the test set can be interpreted in different ways. One way is to view the test set as the validation set, because we evaluate each epoch of the model on the test set and then choose the version of the model with the highest score and this is what the validation set is used for. This would lead to actually having a validation set but no test set and the performance of the model would be overfitted on the validation set and the model can not be truly evaluated on a test set.

The other option is to view the test set as a test set. If only train and test sets exist, it would mean that a version of the model can not be picked, since this is done with the validation set. When evaluating the model on the test set there will not be one version to evaluate but all possible version of the model. By evaluating all versions of the model many scores are obtained but since no validation set was used it can not be determined what score is the right one, i.e. the score which would have been obtained if a validation set existed and a version of the model would have been picked. The maximum of those scores can be calculated to receive an upper-bound of the performance the model could get on unseen data.

The standard is to use train, validation and test sets but in our situation where the dataset is small we argue that the standard way is not perfect either. When dividing the images of the dataset into sets, the images of the same tower site should be in the same set as described in section 3.1.2. With the current

split there are 150 images in the test set and some tower sites contain around 50 images. This would mean that if the dataset would be divided into more sets, small sets like validation and test sets would only contain a few sites, e.g two sites. Hence, the performance on the test set would not mean much because a good performance on those sites do not directly relate to a good performance on general unseen data.

The options are to view the scores as a bit overfitted or an upper-bound for unseen data when no validation set exists, or as an accurate score for only few sites if validation would be used. Neither is perfect, due to the quite insufficient amount of data. In the end only a train and test set was used because it was not possible to smoothly incorporate the validation in the training procedure, as mentioned in section 3.1.2.

5.2 Results

Comparing Table 4.1 with Table 4.2 it is clear that our extension of FixMatch [1] is not a great improvement. The variation in performance of the baseline in Table 4.2 makes it harder to draw conclusions whether our extension is an improvement or not. However, our extension of FixMatch [1] can still be considered an improvement since five runs was made for both the baseline and our extension, and our extension quite constantly achieved better performance.

Improvements for object detection and instance segmentation do not usually have a much higher mAP score. For example, FPN [24] described in section 2.10.3 is a technique that is considered useful and achieved about 1.5 higher mAP score than without it, which was tested on Faster R-CNN [17]. This shows that even small gains are useful.

It is hard to determine the best parameter settings since the performance in the grid searches in Table 4.3 and Fig. 4.1 varies a lot. Since no datapoints in the grid searches have much better performance, we suspect that the parameter settings do not matter much. There are of course some poor parameter settings but most of them have about the same performance. To determine exactly how the performance is affected by the parameter settings, a larger dataset have to be used or many runs have to be averaged. We believe that the small dataset is the main reason for the varying performance while there are many things that could be the cause to why the extension did not perform even better.

5.3 Unmeasured robustness

A possible reason for not performing better may be that RandAugment [40] used in FixMatch [1] will shear and rotate the images, but when evaluating the model the test set is used, which has no sheared, rotated, or even leaning towers. The possible robustness for leaning towers obtained by using FixMatch [1] will not be evaluated due to this missing feature of the dataset. This means that there might be other improvements that can not be seen by the AP obtained for this dataset. The purpose of the models trained on this dataset is to be able to perform instance segmentation on a stream of images from a drone. Since drones are stable the towers in the images captured will mostly be vertical, but if the drone is moving, the towers might lean a bit more and then robustness for learning towers will be useful. It would be a good idea to create a new test set with leaning towers and study if the performance is increased there.

5.4 Unlabeled images

Another reason why our extension did not do better might be the amount of unlabeled data. To get better performance a lot of unlabeled data is needed. The reason for needing much more unlabeled data is because the information semi-supervised approaches can extract from the unlabeled images is considerably lower than the information in manually labeled images. When the size of the unlabeled part is small, labeling everything should be considered. A more suitable situation for semi-supervised learning is when the unlabeled size is ten times the labeled.

5.5 Labels

The label of the dataset could also be a reason for not performing better, but this might cause both the baseline and the extension to do worse. If the labels of the dataset are too inconsistent the model might quickly reach the peak performance of what is possible for the dataset, any further training will not increase the performance because when improving some areas, others get worse. As can be seen in Fig. 4.2, the performance on the test set and the loss on the training set converge as the training progresses. This means that the model does not overfit the training data as more training is done and could be indicating that the model have stopped learning because the labels are too inconsistent.

The inconsistencies in the labels are caused by not having exact guidelines

to follow when labeling the equipment. Without exact guidelines the transitions between the classes will be blurred. For example, when deciding the class of an RRU to be; RRU, occluded RRU or not included in the labels, similar RRUs might be assigned to different classes. We believe this is a common occurrence then labeling objects as some objects will be too occluded to label and it is not easy to be consistent about where to stop labeling the objects. In our case where RRUs can belong to different classes, we think a good idea would be to change the loss function to capture this. An idea would be to split the calculation of the loss into two steps; first whether an RRU is detected as RRU or occluded RRU and then if the detection is the right class. This would punish detecting an occluded RRU as an antenna more than detecting it as an RRU. Another option that could have improved the performance is to include both occluded RRUs and RRUs in a common class or to include all RRUs no matter size, orientation or occlusion in the same class. In Fig. 4.3c the model did detect an RRU that was not in the annotations, this means that the model was penalized for detecting an RRU.

5.6 Hyperparameters

There are a lot of hyperparameters that can be further tuned in our extension of FixMatch [1]. We have not changed the maximum severity of the transformations used in RandAugment [40] and we have not changed the amount of unlabeled data. The new $inside$ hyperparameter determines how much of the equipment needs to be inside the image after the transformations. A similar decision has already been done in when labeling the images. For example, if objects are labeled when at least 50% of their area are inside the image, the labeled objects will be showing 50-100%. This then means that the model will detect equipment with at least 50% showing. If the *inside* parameter is also set to 50%, 50% of the detected equipment (which are showing 50-100%) may get lost during the transformations. In the new strongly augmented image you will end up with pseudo-labels for objects showing 25-100% but some labels for objects with 49% inside the image have been discarded. This is a problem because the pseudo-labels for the strongly augmented images will not have the same qualities as the manually labeled images. If the $inside$ parameter instead forces the objects to not lose any area during the transformations, the pseudo-labels for the strongly augmented images will contain objects showing 50-100% of their area, but labels for objects showing 99% will be discarded. Perhaps there is a better way to implement this $inside$ parameter which makes it easier to control and do not cause this imprecise threshold.

5.7 Semi-supervised learning

Semi-supervised learning usually looks for consistency. The idea is often to use the existing model and assume what it predicts is correct and then change the image slightly while treating the old predictions as true. This helps the model gain a little new knowledge. The main assumption is that the label, bounding box, or mask should not change if the image is just slightly changed. This makes the model more robust to slight changes. This robustness is often wanted, but there could be situations where it is not wanted. As explained in section 3.1.3, the classes do not consist of equipment types but rather equipment types in certain orientations. A slight change in the orientation of an RRU will determine if it should or should not be detected. Even if the augmentation done to the images do not change the orientation of the RRU, it might still change its label if we would have labeled it. For example, how clearly an RRU is visible will decide if it belongs to the RRU class, occluded RRU class, or background. Therefore, semi-supervised approaches, and hence our extension, might not be optimal on our custom dataset.

5.8 Final thoughts

Even if our semi-supervised approach is a slight improvement, we can not draw general conclusions whether semi-supervised learning do improve the performance of instance segmentation of Telecom equipment. The more important question is whether the improved performance from semi-supervised learning is worth the time making it to work. We believe that unless the semi-supervised learning approach is already implemented and the amount of unlabeled data is huge, semi-supervised learning is not the best way to improve the performance. It is harder to implement a semi-supervised approach than supervised approach because supervised learning is the usual way to train, hence there will be more work to get a semi-supervised approach working.

Another disadvantage with semi-supervised learning is that it uses a lot of unlabeled data, which means the training takes much longer. In our implementation, which is not optimized, it takes almost six hours to train for twelve epochs, while the baseline model takes about 80 minutes. The longer training time is due to the fact that more unlabeled images are used than labeled ones and that each unlabeled image needs to be forwarded twice; one for the weak augmentation and one for the strong augmentation. With the amount of unlabeled data we had, it would have been faster to label everything than

implementing the semi-supervised approach.

Apart from the negative aspects of semi-supervised learning there is one great benefit and it is the inference time. Since semi-supervised learning is only present during training, the inference time will be the same as for the base model. This means that if a semi-supervised learning approach achieved as good performance as a more complex model (which will be slower) the semi-supervised learning approach is preferred.

As mentioned in section 3.1.4 mAP is an average over many things which means it is hard to get a high score. The leader in the COCO test-dev [32] has an AP score of about 0.47, but a direct comparison between COCO [32] and our dataset can not be made since COCO [32] contains more classes and has more images.

Despite our extension not being a huge improvement the performance of the baseline is quite good. The images in Fig. 4.3 further confirms this statement, since the masks look good and almost all equipment were found. The false detections are understandable as they are part of the difficulties in our dataset.

Chapter 6

Conclusions

Semi-supervised learning is mostly done for classification problems but can also be extended to instance segmentation. The drawbacks of semi-supervised learning are that it can be harder to implement, it takes considerably longer to train with, and a lot of unlabeled data is needed to notice an improvement.

A custom dataset was used in this project which we labeled ourselves. We learned that labeling a dataset is difficult and it is very important to be consistent, the prediction quality of the model is after all learned from the labels.

In this project an extension of FixMatch [1] was proposed to answer the question whether semi-supervised learning can improve the performance of instance segmentation of Telecom equipment. The extension, applied to Mask R-CNN [29], achieves 1.0 point higher mAP compared to Mask R-CNN [29] trained with supervised learning. The dataset used is custom and consists of Telecom equipment from various tower sites. Due to the small increase in mAP, the extension is not considered an improvement in the current situation, where the amount of unlabeled data is twice the labeled. If more unlabeled data existed the extension should perform even better, but in our case the data was limited. However, there are still many hyperparameters that can be adjusted to achieve even better performance.

Chapter 7

Future work

There are many things that can be further tested or improved. For example, a more detailed grid search can be done where all the data points are the mean of many runs.

There are also more parameters that can tested. In RandAugment [40] the maxmimum severities for the transformations can be changed and the amount of unlabeled data can be changed. It would be interesting to see how the improvement in performance behaves as more unlabeled data is added.

The labeling technique can also be changed, maybe the RRUs should be included in a common class instead of two separate for RRUs and occluded RRUs. We also believe that labeling the TMA is a good idea, the model might then be able the distinguish between RRUs and TMAs better.

The loss function can be changed to give higher loss for an RRU detected as an antenna than an RRU detected as an occluded RRU.

To really determine if semi-supervised learning can be used to improve the performance of instance segmentation of Telecom equipment, other semi-supervised approaches should also be tested, it might be that FixMatch [1] was not suitable for this dataset.

Bibliography

[1] Kihyuk Sohn et al. “Fixmatch: Simplifying semi-supervised learning with consistency and confidence”. In: *arXiv preprint arXiv:2001.07685* (2020).

[2] Sumit Saha. *A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way*. [Online; accessed June 14, 2020]. 2018. URL: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

[3] Karen Simonyan and Andrew Zisserman. “Very deep convolutional networks for large-scale image recognition”. In: *arXiv preprint arXiv:1409.1556* (2014).

[4] Muneeb ul Hassan. *VGG16 – Convolutional Network for Classification and Detection*. [Online; accessed June 1, 2020]. 2018. URL: https://neurohive.io/en/popular-networks/vgg16/.

[5] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[6] Saining Xie et al. “Aggregated residual transformations for deep neural networks”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1492–1500.

[7] Christian Szegedy et al. “Going deeper with convolutions”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1–9.

[8] Christian Szegedy et al. “Inception-v4, inception-resnet and the impact of residual connections on learning”. In: *Thirty-first AAAI conference on artificial intelligence*. 2017.

[9] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[10] Ross Girshick et al. "Rich feature hierarchies for accurate object detection and semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 580–587.

[11] Zhengxia Zou et al. "Object detection in 20 years: A survey". In: *arXiv preprint arXiv:1905.05055* (2019).

[12] Joseph Redmon et al. "You only look once: Unified, real-time object detection". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 779–788.

[13] Joseph Redmon and Ali Farhadi. "Yolov3: An incremental improvement". In: *arXiv preprint arXiv:1804.02767* (2018).

[14] Wei Liu et al. "Ssd: Single shot multibox detector". In: *European conference on computer vision*. Springer. 2016, pp. 21–37.

[15] Jonathan Huang et al. "Speed/accuracy trade-offs for modern convolutional object detectors". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 7310–7311.

[16] Ross Girshick. "Fast r-cnn". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1440–1448.

[17] Shaoqing Ren et al. "Faster r-cnn: Towards real-time object detection with region proposal networks". In: *Advances in neural information processing systems*. 2015, pp. 91–99.

[18] Jifeng Dai et al. "R-fcn: Object detection via region-based fully convolutional networks". In: *Advances in neural information processing systems*. 2016, pp. 379–387.

[19] Xin Lu et al. "Grid r-cnn". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 7363–7372.

[20] Zhaowei Cai and Nuno Vasconcelos. "Cascade r-cnn: Delving into high quality object detection". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 6154–6162.

[21] Mate Kisantal et al. "Augmentation for small object detection". In: *arXiv preprint arXiv:1902.07296* (2019).

[22] Chao Dong et al. "Learning a deep convolutional network for image super-resolution". In: *European conference on computer vision*. Springer. 2014, pp. 184–199.

[23] Tsung-Yi Lin et al. “Focal loss for dense object detection”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2980–2988.

[24] Tsung-Yi Lin et al. “Feature pyramid networks for object detection”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 2117–2125.

[25] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. “Nas-fpn: Learning scalable feature pyramid architecture for object detection”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 7036–7045.

[26] Yuanyi Zhong et al. “Anchor box optimization for object detection”. In: *The IEEE Winter Conference on Applications of Computer Vision*. 2020, pp. 1286–1294.

[27] Shu Liu et al. “Path aggregation network for instance segmentation”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 8759–8768.

[28] Jianan Li et al. “Perceptual generative adversarial networks for small object detection”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1222–1230.

[29] Kaiming He et al. “Mask r-cnn”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2961–2969.

[30] Zhaojin Huang et al. “Mask scoring r-cnn”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 6409–6418.

[31] Daniel Bolya et al. “YOLACT: real-time instance segmentation”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 9157–9166.

[32] Tsung-Yi Lin et al. “Microsoft coco: Common objects in context”. In: *European conference on computer vision*. Springer. 2014, pp. 740–755.

[33] Kai Chen et al. “Hybrid task cascade for instance segmentation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2019, pp. 4974–4983.

[34] Ilija Radosavovic et al. “Data distillation: Towards omni-supervised learning”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 4119–4128.

[35] Ahmet Iscen et al. “Label propagation for deep semi-supervised learning”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 5070–5079.

[36] Antti Tarvainen and Harri Valpola. “Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results”. In: *Advances in neural information processing systems*. 2017, pp. 1195–1204.

[37] Christian Szegedy et al. “Intriguing properties of neural networks”. In: *arXiv preprint arXiv:1312.6199* (2013).

[38] Takeru Miyato et al. “Virtual adversarial training: a regularization method for supervised and semi-supervised learning”. In: *IEEE transactions on pattern analysis and machine intelligence* 41.8 (2018), pp. 1979–1993.

[39] Terrance DeVries and Graham W Taylor. “Improved regularization of convolutional neural networks with cutout”. In: *arXiv preprint arXiv:1708.04552* (2017).

[40] Ekin D Cubuk et al. “RandAugment: Practical data augmentation with no separate search”. In: *arXiv preprint arXiv:1909.13719* (2019).

[41] David Berthelot et al. “ReMixMatch: Semi-Supervised Learning with Distribution Alignment and Augmentation Anchoring”. In: *arXiv preprint arXiv:1911.09785* (2019).

[42] Abhishek Dutta and Andrew Zisserman. “The VIA Annotation Software for Images, Audio and Video”. In: *Proceedings of the 27th ACM International Conference on Multimedia*. MM ’19. Nice, France: ACM, 2019. ISBN: 978-1-4503-6889-6/19/10. DOI: 10.1145/3343031.3350535. URL: https://doi.org/10.1145/3343031.3350535.

[43] Jonathan Hui. *mAP (mean Average Precision) for Object Detection*. [Online; accessed May 31, 2020]. 2018. URL: https://medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121a31173.

[44] Kai Chen et al. “MMDetection: Open MMLab Detection Toolbox and Benchmark”. In: *arXiv preprint arXiv:1906.07155* (2019).

[45] Alex Krizhevsky, Geoffrey Hinton, et al. “Learning multiple layers of features from tiny images”. In: (2009).

[46] Yuval Netzer et al. “Reading digits in natural images with unsupervised feature learning”. In: (2011).

TRITA TRITA-EECS-EX-2020:629