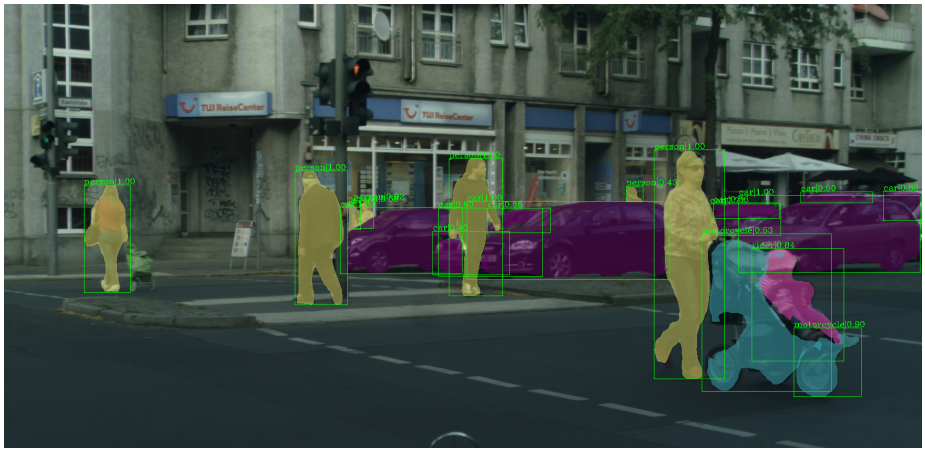

TDDE19 Group 3 - Instance segmentation

Martin Jirenius, August Johnson, Jacob Olausson, Jacob Larsson

2021-01-03

1 Introduction

1.1 Background

Instance segmentation is part of the science field known as computer vision. Computer vision dates back more than a half century, however the complexity of the subject wasn't as apparent back then as for today. A memo from 1966 at MIT, *The Summer Vision Project* [1], clearly shows this as they assign summer workers the task of constructing an entire visual system. However, even though they were far off achieving all their goals, the proposed method wasn't unlike what we usually use today. In the memo, it's stated that the plan for the project was to extract features like edges, using the results to extract higher level features like shapes and in a step-wise manner end up at object identification. This feature-based method were mostly disregarded due to computational inconvenience but resurrected in the late 1900 and early 2000 with the rise of machine learning [2].

In 1994 Shi and Tomasi published a paper which introduced an algorithm for finding good features to track, which ended the research on the object tracking area [3]. Until approximately 2006 the tracking problem was labelled as solved, but since then the research has resumed and after the deep network revolution in 2012 with introduction of the Alex Net, the research intensified. Previously, the object detection and object tracking algorithms was dominated by bounding boxes both for indicating a detected object and looking for the same object in the next image frame. However since the deep network revolution this has been questioned and improvements to this has been researched.

Instance segmentation is such an improvement. Instead of tracking and detecting a bounding boxes, the goal of the algorithm is supposed to find a mask which encapsulates the entire object. This has been popular and in the challenge for visual object tracking 2020 (VOT2020) all competing algorithms should use this methodology. [1] Instance segmentation is what could be described as a combination of classical object detection, in which the goal is to localize all objects of a certain class in an image, and semantic segmentation, where each pixel is labelled to a pre-defined set of classes. Instance segmentation has, unlike semantic segmentation, both the masks of each interesting object in an image and can separate between different instances of the same class.

1.2 Objective

The objective of this project is to implement three different state-of-the-art models for instance segmentation: Mask-RCNN, Cascade Mask-RCNN and GCNet. The purpose is to achieve an accurate model for detection of objects in images and videos. Furthermore, the report will provide insight about how these different models operate and possibly differ from each other.

[1] https://www.votchallenge.net/vot2020/

2 Method

2.1 Dataset and Evaluation

The primary dataset used for evaluating the different methods is the Cityscapes [4] dataset which consists of 5000 annotated images with 30 different classes. Many different settings and conditions are represented in the dataset in the form of different urban environments, weather, seasons as well as varying illuminations and occlusions. The dataset contains both annotations for bounding boxes and segmentation masks so the methods can be evaluated using both metrics.

The evaluation is done by training all methods for 8 epochs using different learning rates. After each epoch random flip augmentations are applied to the training data to artificially expand it. When the training is done the methods are evaluated by mean average precision of the bounding boxes and masks, where precision is the proportion of true positives. A box or mask is considered correct if the intersection over union (IoU) is over some threshold. The thresholds used in this report is the same as those used by the COCO dataset: mAP50 (IoU > 0.5), mAP75 (IoU > 0.75) and mAP (IoU $> [0.5 : 0.05 : 0.95)]$. Also, performance for differently sized objects will be presented as well.

In addition, due to limitations in computational resources, further analysis of Mask-RCNN will be conducted in order to evaluate model performance with more tuned parameters. In this case, the cigarette butt dataset is used [5]. This dataset contains 2000 annotated images with one class, and an additional 10 images for out-of-sample validation. 4 epochs, with 500 steps per epochs are used during training when evaluating different learning rates. The ResNet-50 and ResNet-100 backbones are also evaluated.

2.2 Previous work

Since mask R-CNN builds upon previous work on object detection, this section briefly describes the papers and methods that mask R-CNN is based on.

2.2.1 R-CNN

The region-based convolutional neural network (R-CNN)[6] is a classifier that most modern instance-segmentation algorithms are based on. It works by taking an input image and running it through a network to pick out region proposals. It produces about 2000 proposals where each of the proposals are passed to a class-specific linear support vector machine (SVM). If one region proposal has an overlap (IOU) larger than a threshold with a region with higher score, the region is deleted. This is a type of non-max suppression.

2.2.2 Fast R-CNN

The paper about fast R-CNN [7] highlights 3 flaws in the R-CNN implementation

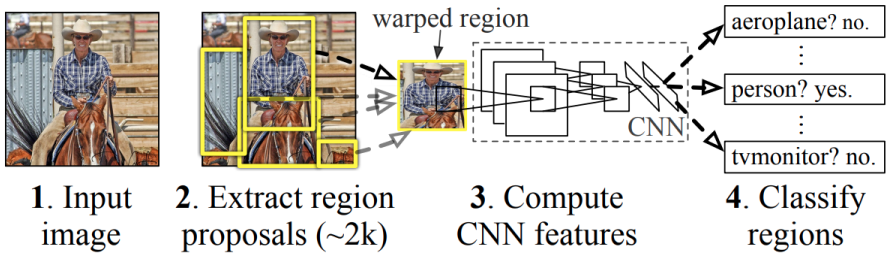

Figure 1: R-CNN network architecture[6]

1. Training is a multistage pipeline
2. Training is expensive in time and space
3. Object detection is slow

Their solution adds a fully convolutional layer which takes the image and the region proposals as input. The region of interests are first pooled using max-pooling to feature maps and then mapped to feature vectors by fully connected layers. This network has two outputs: probabilities that the region belongs to a specific class and the bounding box coordinates. The fast R-CNN network increases the detection quality, speed and reduces the amount of storage needed. The network overview can be seen in figure 2.

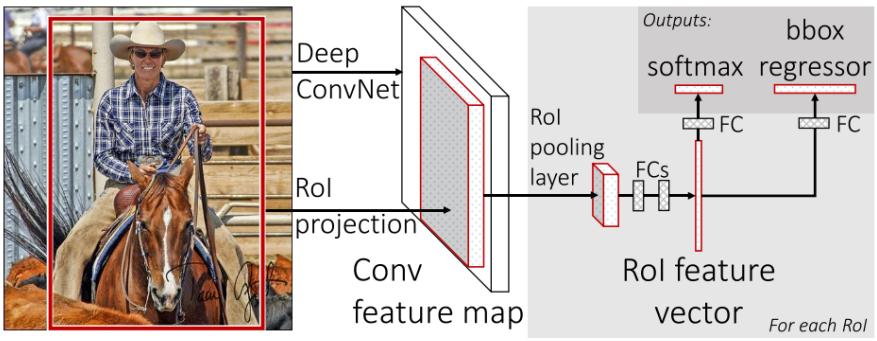

Figure 2: Fast R-CNN network architecture[7]

2.2.3 Faster R-CNN

The main idea behind the faster R-CNN network is to improve the speed in which the region proposals are extracted [8]. Fast R-CNN is fast enough when ignoring the time spent on region proposals, but faster R-CNN aims to provide a network where the region proposal networks (RPNs) share convolutional layers with the object detection network in order to improve the speed. The faster R-CNN paper also introduces a new RPN scheme which avoids enumerating images of different scale, thus improving the speed of the RPN.

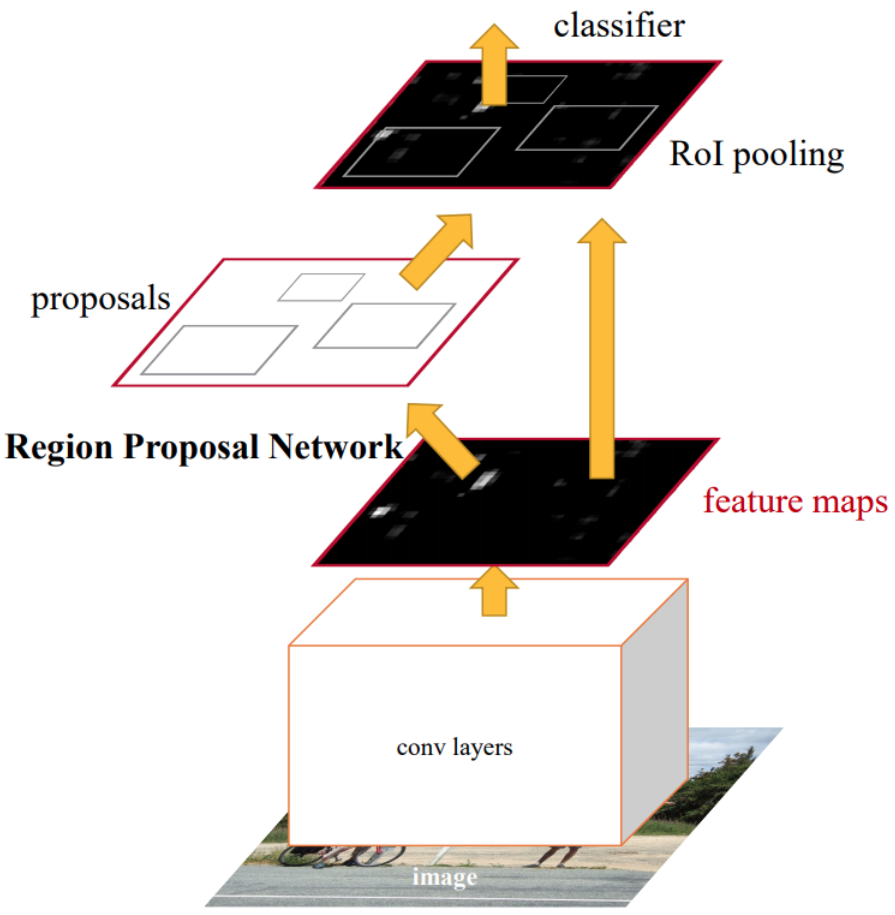

Figure 3: Faster R-CNN network architecture[8]

2.3 Mask R-CNN

The mask R-CNN network extends the work of Faster R-CNN with another branch which predicts the mask of the object. Thus the output of the mask R-CNN network is the class label, the bounding box offsets and the mask[9].

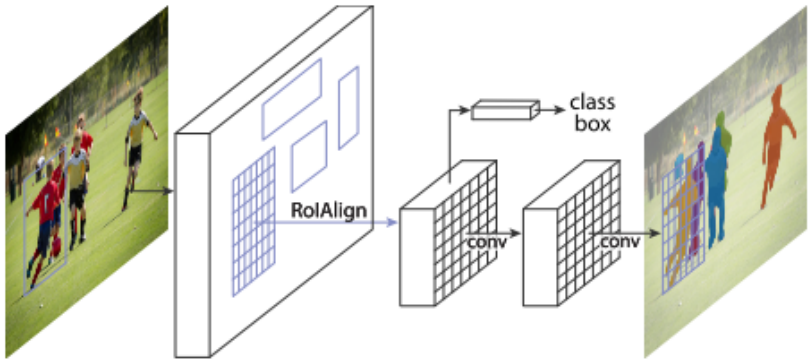

Figure 4: Mask R-CNN overview[9]

2.3.1 RoIAlign

The RoI pooling in the faster R-CNN network introduce a misalignment between the RoI and the extracted features which turns out to affect the pixel-accurate mask predictions introduced in this work. The solution to this is to remove the RoI pooling used in Faster R-CNN and add the RoIAlign layer which aligns the extracted features with the input image more efficiently. Each sampling point in the extracted RoIs is interpolated using bilinear interpolation from nearby feature map grid points. This is illustrated in figure 5. This methodology removes the quantification completely from the system and improves the mask accuracy with 10%-50%[9].

2.3.2 Loss function

The loss function during training used in this work was defined as

$$L = L_{cls} + L_{box} + L_{mask} \tag{1}$$

i.e basically adding the mask loss to the already existing loss function from faster R-CNN. L_{cls} and L_{box} are defined in the faster R-CNN paper [8] as

$$L_{cls} = -log(p_u) \tag{2}$$

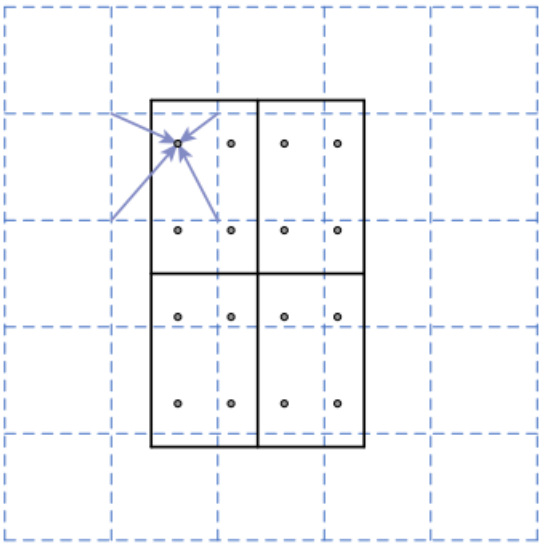

Figure 5: RoIAlign - the dashed grid is the feature map and the solid points the sampled points in the RoI[9]

where p_u is the probability p for the true class u.

$$L_{loc} = \sum_{i \subset \{x,y,w,h\}} smooth_{L1}(t_i^u - v_i) \tag{3}$$

in which

$$smooth_{L1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases} \tag{4}$$

t^u is the scale-invariant translation relative to an object proposal and v is the ground truth position.

The addition to the already existing loss function L_{mask} introduced in [9] is defined as the average cross-entropy loss when applying a per-pixel sigmoid to the mask branch output. The perk of defining the loss in this way is that for a ground truth RoI class k, the loss is only defined on the k-th mask. This means that other masks does not contribute to the loss.

2.3.3 Generated masks

The mask predicting branch which is the main part of the mask R-CNN network generate binary m x m regions for each RoI. The mask for each RoI is predicted using a FCN which requires appropriate RoI-features, handled by the RoIAlign as described previously.

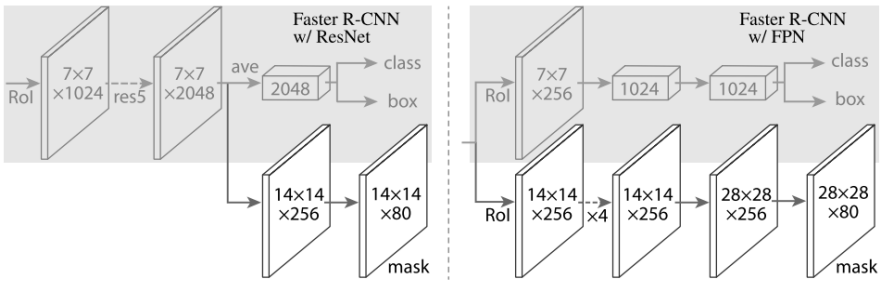

Figure 6: Mask R-CNN for two different architectures[9]

One advantage of mask R-CNN and a reason to why it has been used widely after its release is its generality. The mask branch can be added to different architectures for bounding box recognition, such as ResNet[10] and Feature Pyramid Network (FPN)[11]. The mask R-CNN customized for two different architectures can be seen in figure 6.

2.4 Cascade Mask-RCNN

2.4.1 Cascade RCNN

Cascade RCNN [12] is a multi-stage extension to RCNN which adds multiple detectors in sequence, where each stage has an increasingly higher intersection-over-union threshold and each uses the positive outputs of the previous stage. Due to the nature of having multiple stages the bounding-box regression becomes a set of cascades as well, one regressor for each stage, and they are optimized for the inputs arriving at each stage. In other words, the initial regressor is optimized for the input and the second regressor is optimized for the output of the initial regressor and so on.

$$r(x, b) = r_T(x, b_T) \circ r_{T-1}(x, b_{T-1}) \circ ... \circ r_1(x, b)$$

The detection part of the architecture becomes cascaded in a similar way where a separate classifier and regressor are used at different stages.

2.4.2 Adding Instance Segmentation

Cascade *Mask*-RCNN adds instance segmentation to this architecture. Compared to the Mask-RCNN there are multiple places where the instance segmentation part can be added since there are multiple cascades of detection's. Cai et al. [12] suggests adding a segmentation part to each cascade for the best results, which is what is used in this report. The general architecture of Cascade Mask-RCNN can be seen in figure 7

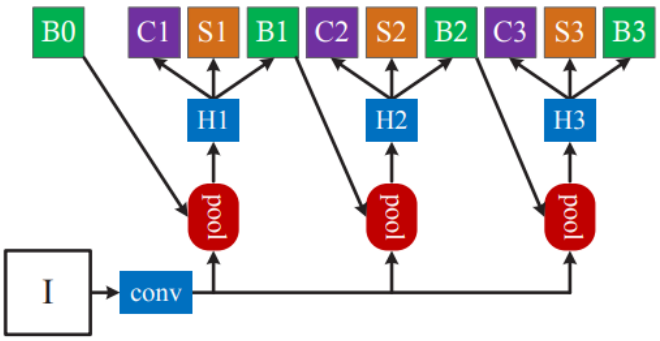

Figure 7: Cascade Mask-RCNN architecture [12]

2.5 GCNet

Cao *et al.* [13] presents a potential extension for instance segmentation solutions called a global context block, which simplifies and combines the features of Non-Local Networks (NLNet) [14]. and Squeeze-Excitation Networks (SENet) [15]. NLNet aims to find how pixels depend on each other on a global scale, while SENet tries to find the relation between different channels from a given feature map. The GCNet is a combination of these two networks and forms the global context block.

An important part of the GCNet is that it does not directly use the NLNet, but a simplified versions of it. Cao *et al.* found that the attention maps that are computed and used inside the block, one for each dependency, are usually very similar. Therefore a global attention map is introduced to reduce computations and make it faster. The global context block architecture can be seen in figure 8

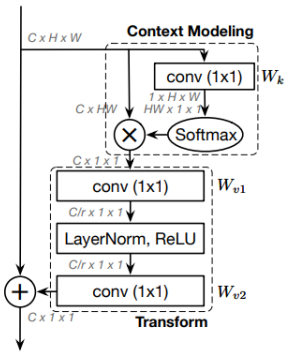

Figure 8: GCNet block architecture [13]

3 Results

3.1 Model Comparison

The results of the models listed in section 1.2 will be presented in this section in order of listed appearance. The performance of the different evaluations for each model can be seen in table 1, where the results for each model in the table is taken from the best result from testing using different learning rates. How each models mean average precision change for the set of learning rates can be seen in figure 9.

Table 1: Mean average precision for all models

Model	Mask-RCNN	Cascade Mask-RCNN	GCNet
mAP	0.368	0.102	0.369
mAP50	0.638	0.220	0.639
mAP75	-1	-1	-1
mAPs	0.126	0.039	0.117
mAPm	0.341	0.110	0.34
mAPl	0.581	0.164	0.584

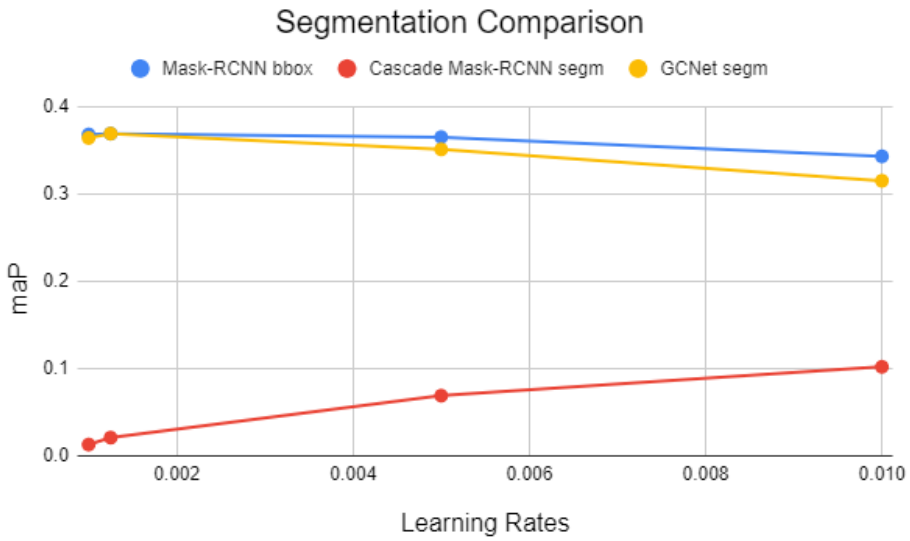

Figure 9: mAP over learning rate

Considering figure 9, the trend of all three methods can easily be seen. Firstly, Mask-RCNN shows its best performance for a learning rate of 0.001,

GCNet peaks just around the corner at 0.00125 while Cascade Mask-RCNN performed best for a learning rate of 0.01. It is worth noting that 0.001 is the lower bound and 0.01 is the upper bound of this parameter evaluation.

3.2 Fine-tuning Mask-RCNN

In this section, the results from evaluating Mask-RCNN are presented, with the configurations and cigarette butts dataset, which is described in section 2.1.

3.2.1 Learning rate and backbone evaluation

Below are the results from training mask-RCNN with four different learning rates. Figure 10 illustrates the results from applying ResNet-50 as a backbone, and Figure 11 shows the results from applying ResNet-101. By comparing the two figures, one can observe that ResNet-101 achieves a higher precision for each learning rate. The highest achieved mAP is 0.9425 with a learning rate of 0.0005, as seen in Figure 11.

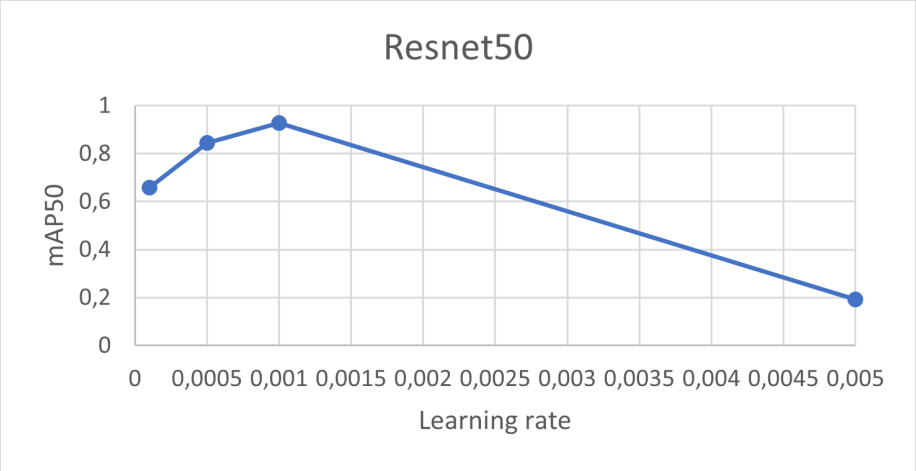

Figure 10: mAP over learning rate, using resnet50 as backbone

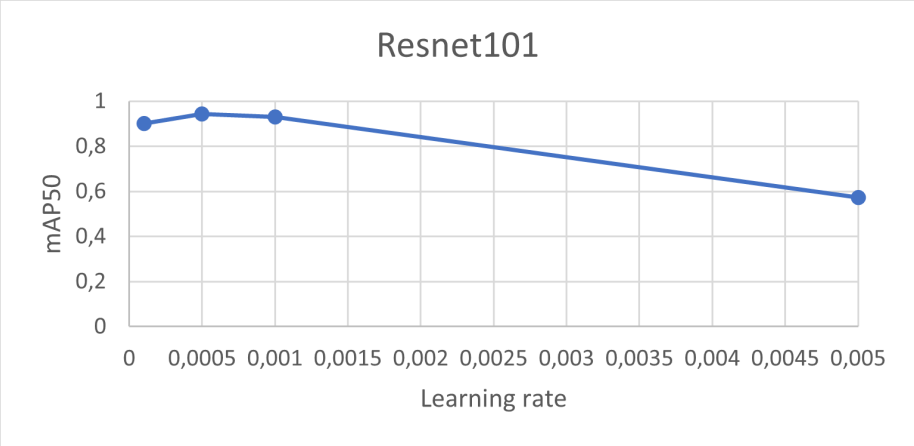

Figure 11: mAP over learning rate, using resnet101 as backbone

3.2.2 Out-of-sample Validation

The results in Figure 12 presents the confidence scores from classifying cigarettes in the six images below. The results indicate that the model, for the most part, correctly identifies cigarettes in the images. Although, it miss-classifies twice in the top-right image in Figure 12.

Figure 12: Resulting confidence score for each identified cigarette in the validation set

4 Discussion

4.1 Mask-RCNN

As can be seen in table 1, Mask-RCNN gets a mean average precision of about 37%. Its mAP50 is about 64% meaning that it usually finds, at least, a rough mask estimate even though it is not precise. -1 on mAP75 shows that it gets none correct, which is disappointing. Looking at figure 9, the model shows best performance at the lower bound, meaning that a larger evaluation needs to be conducted in order to find an optimal learning rate. However, the curve is farily flat, suggesting that a significant better learning rate is unlikely to be found below the lower bound value. With that in mind, one may find a more representative learning rate if more data were to be added during training.

When observing the results from training with the cigarette butts dataset, one can also see that backbone plays an important role in model precision. In addition, one could have also performed more tests with different batch sizes, since it has a vital role in training and prediction performance. When considering how close the results was when evaluating Mask-RCNN and GCNet in Figure 9, we also see that further testing would be needed in order to gain more reliable results.

4.2 Cascade Mask-RCNN

The results of Cascade Mask-RCNN in table 1 shows that the model is having trouble finding correct masks. We do see an aggressive increase in performance with increased learning rate, meaning that an optimal value is likely to be found beyond the upper bound. However, even with that in mind, it is still notably worse than both regular Mask-RCNN and the results of the original paper [12], suggesting something is not quite right. Cascade Mask-RCNN essentially consists of multiple Mask-RCNNs, making the convergence time in training longer and more sensitive to smaller datasets. This could be one explanation for the performance issue. Most of these networks requires datasets large enough to thoroughly train the model which takes a significant amount of time. The framework that was used, mmdetection [16], expects that multiple workstation-GPUs are available but since only a single GTX 1070 GPU was available, the training had to be reduced.

4.3 GCNet

GCNet had the best overall performance and an example of segmentations and classifications generated by the best GCNet model can be seen in figure 13. The best result when using the GCNet was for a learning rate of 0.00125 which had a segmentation mAP of 36.9%. It gets a similar result compared to Mask-RCNN for mAP50 and just like both of the other methods -1 for mAP75. Considering the different object sizes, it seems to be struggling for smaller and medium sized objects. GCNet should, according to the original article [13], increase the accuracy with 2.4% compared to Mask-RCNN. However, as shown in table 1, our results shows an increase of 0.27%, approximately 11% of the presented performance in the GCNet article. There could be several reasons to why our results did not match the presented performance by the authors. Insufficient size of the training dataset and poor parameter tuning could be two sources of the issue.

4.4 Conclusions

After testing the different types of models at different learning rates, it was concluded that GCNet was the best model for overall performance. However, it is worth noting that regular Mask-RCNN performed just slightly worse, and even performed better for smaller sized objects which are generally harder to detect and segment correctly. The results from Mask-RCNN are close to the expected results with the authors own experiments [9] while the other two models performed significantly worse than the proposed performance. In addition, Mask-RCNN gave very promising results when classifying one class from the smaller dataset. These results suggest that there is more room for improvement, in terms of hyperparameter optimization.

We expected better performance from Cascade Mask-RCNN and GCNet, especially better performance relative to the Mask-RCNN model. Our first

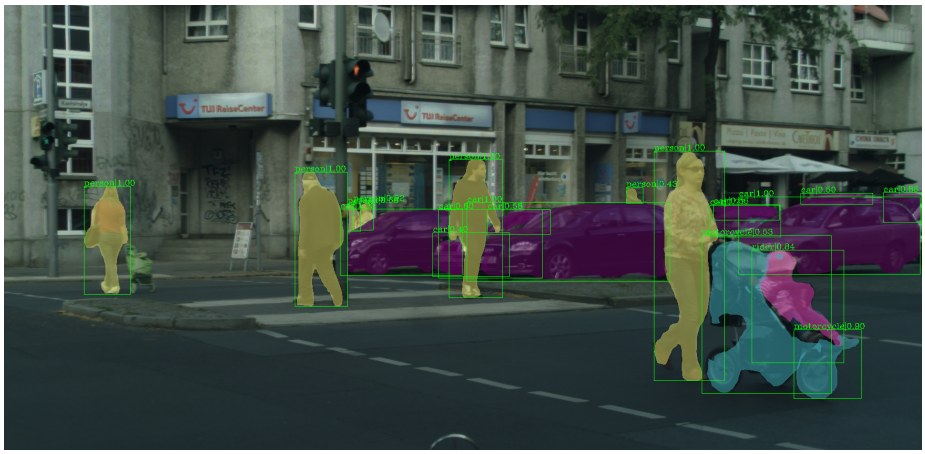

Figure 13: GCNet segmentations and classifications

suspicion is that we have not used enough data during training. We had to truncate the training set due to insufficient computing power which could play a role in this performance issue since the models probably have converged to a poor value or not at all. On the other hand, we gain some valuable insight due to this since is shows robustness of Mask-RCNN and GCNet over Cascade Mask-RCNN because they performed fairly well despite lack of training data. Secondly, tuning could also be part of the issue but due to insufficient time, a thorough cross-validation evaluation could not be conducted. Lastly, two different datasets were used to train and test the networks. This would definitely have an impact on the results, and may have affected the performance. If more time were available, it would be interesting to test our implemenation on the entire COCO dataset which would provide insight about what improvements that could be implemented.

4.5 Future work

As shown in the report, there is room for improvements for future research. One direction is to get better results with smaller sets. As demonstrated, the models can achieve near perfect results on specific tasks however when increasing the data sets and the amount of classes to predict, it struggles. This problem limits the specific hardware that can be used and complicates the task of further research.

Another direction is to run the models in real time, as this would have tremendous applications in autonomous vehicles. The problem to identify pedestrians and other obstacles is difficult but vital for the system to be reliable enough to be used in practice. With instance segmentation the localization accuracy for obstacles will improve, since the mask of each object will provide the

exact boundaries. However, if this was to be used in real world applications the speed, of which the system calculates the masks, has to be several frames per second (fps). This provides an area of future work where the instance segmentation algorithm needs to run at least at 30 fps. There is a lot of research in this subject already. For instance, the model *CenterMask* [17] can be run slightly above 36 fps and achieve a mAP of almost 36 percent on the *COCO* set, almost as good as *Mask-RCNN* and *GCNEt* in our experiments. In the future, if the autonomous vehicles industry takes the next step toward fully self driving cars, it is likely that even higher speeds would be necessary.

References

[1] Seymour A. Papert. *The Summer Vision Project*. 1966. URL: http://hdl.handle.net/1721.1/6125.

[2] Nicu Sebe et al. *Machine Learning in Computer Vision*. Springer, 2005. ISBN: 978-1-4020-3274-5.

[3] Jianbo Shi and Carlo Tomasi. "Good Features". In: *Image (Rochester, N.Y.)* (1994), pp. 593–600. ISSN: 1063-6919.

[4] Marius Cordts et al. "The Cityscapes Dataset for Semantic Urban Scene Understanding". In: *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[5] *Cigarette Butt Dataset*. URL: https://www.immersivelimit.com/datasets/cigarette-buttsl. (accessed: 10.11.2020).

[6] Ross Girshick et al. "Rich feature hierarchies for accurate object detection and semantic segmentation". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2014), pp. 580–587. ISSN: 10636919. DOI: 10.1109/CVPR.2014.81. arXiv: 1311.2524.

[7] Ross Girshick. "Fast R-CNN". In: *Proceedings of the IEEE International Conference on Computer Vision* 2015 International Conference on Computer Vision, ICCV 2015 (2015), pp. 1440–1448. ISSN: 15505499. DOI: 10.1109/ICCV.2015.169. arXiv: 1504.08083.

[8] Shaoqing Ren et al. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.6 (2017), pp. 1137–1149. ISSN: 01628828. DOI: 10.1109/TPAMI.2016.2577031. arXiv: 1506.01497.

[9] Kaiming He et al. "Mask R-CNN". In: *2017 IEEE International Conference on Computer Vision (ICCV)* (Oct. 2017).

[10] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* 2016-December (2016), pp. 770–778. ISSN: 10636919. DOI: 10.1109/CVPR.2016.90. arXiv: 1512.03385.

[11] Xiaohan Li et al. "Weighted feature pyramid networks for object detection". In: *Proceedings - 2019 IEEE Intl Conf on Parallel and Distributed Processing with Applications, Big Data and Cloud Computing, Sustainable Computing and Communications, Social Computing and Networking, ISPA/BDCloud/SustainCom/SocialCom 2019* (2019), pp. 1500–1504. DOI: 10.1109/ISPA-BDCloud-SustainCom-SocialCom48970.2019.00217. arXiv: arXiv:1612.03144v2.

[12] Zhaowei Cai and Nuno Vasconcelos. *Cascade R-CNN: High Quality Object Detection and Instance Segmentation*. 2019. arXiv: 1906.09756 [cs.CV].

[13] Yue Cao et al. *GCNet: Non-local Networks Meet Squeeze-Excitation Networks and Beyond*. 2019. arXiv: 1904.11492 [cs.CV].

[14] Xiaolong Wang et al. *Non-local Neural Networks*. 2018. arXiv: 1711.07971 [cs.CV].

[15] Jie Hu et al. *Squeeze-and-Excitation Networks*. 2019. arXiv: 1709.01507 [cs.CV].

[16] Kai Chen et al. *MMDetection: Open MMLab Detection Toolbox and Benchmark*. 2019. arXiv: 1906.07155 [cs.CV].

[17] Youngwan Lee and Jongyoul Park. "CenterMask: Real-Time Anchor-Free Instance Segmentation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020.

A List of contributions

Here it is specified what each member contributed with during the project.

A.1 Jacob Olausson

Implemented and tested Mask RCNN with the cig butts dataset early on to get a greater understanding of the problem it was supposed to solve and how Mask RCNN solves it. Further on in the project the theoretical background and details of Mask RCNN were studied and written in the report. Primarily wrote in the introduction and method parts of the report.

A.2 Martin Jirenius

Implemented an early version where the main goal was to learn the different libraries related to Mask RCNN and Keras in particular. Have been studying related work to gain an extended knowledge of the subject and where our work lies in development. Have focused heavily on backing up claims in the report with relevant references as well as writing the introduction and discussion.

A.3 Jacob Larsson

Training and testing of the models on the Cityscapes dataset, mostly worked with getting the framework running properly. Wrote about the theoretical background of the Cascade Mask-RCNN and GCNet models and put together the results for all the models on the Cityscapes dataset. Also wrote parts of the discussion around the results.

A.4 August Johnson

Implementation, tuning, and testing of the Mask-RCNN model, worked primarily with the cigarette butt dataset. Produced the results seen in Figure 11 and Figure 10. Primarily wrote in the Dataset and Evaluation segment, Discussion, and results.