Pedestrian Instance Segmentation with Mask R-CNN

Ignacio Perez Mahiques
M.Sc. Student, Group 7
Technical University of Munich
Professorship of Multiscale Modeling of Fluid Materials
Boltzmannstraße 15, 85747 Garching

Lab course "Hands-on Deep Learning"

Instance segmentation is a complex yet useful task in computer vision. In this paper, the task of segmenting pedestrians in an image is accomplished with a Mask R-CNN model on the Penn-Fudan Dataset. Moreover, object detection with a simpler backbone is executed. The performance of both models is compared and it is discussed why the precision of this application is higher than the original model on the COCO dataset.

I. Introduction

In the last few years the performance of deep convolutional neural networks (CNNs) has improved rapidly in the tasks of object detection and image segmentation. One technique that has driven this progress is the region-based convolutional neural networks (R-CNNs). The R-CNN architecture has been used in multiple models of object detection such as R-CNN [1] and Fast/Faster R-CNN [2], [3]. In this report the framework used is Mask R-CNN [4]. This architecture is an extension of Faster R-CNN in order to solve the task of instance segmentation. While the objective of semantic segmentation is to classify pixels of images into determined labels, instance segmentation also detects the object instance. Instance segmentation is challenging because it requires the detection of all objects in an image and the segmentation of each instance. DeepMask [5] and "fully convolutional instance-aware semantic segmentation" (FCISS) [6] are some earlier methods with worse performance than Mask R-CNN in this task [4].

II. Methods

A. Deep Learning Architecture

The outputs of the Faster R-CNN architecture are a class label and a bounding box offset for each object instance. Mask R-CNN extends Faster R-CNN with a new branch that outputs the segmentation mask of the object. This mask requires extraction of much finer spatial layout of the object. Therefore, some changes are needed in the Faster R-CNN architecture in order to achieve a precise instance segmentation.

Faster R-CNN is composed of two stages. The first stage, called Region Proposal Network (RPN), is a deep CNN that proposes a set of object bounding boxes with an

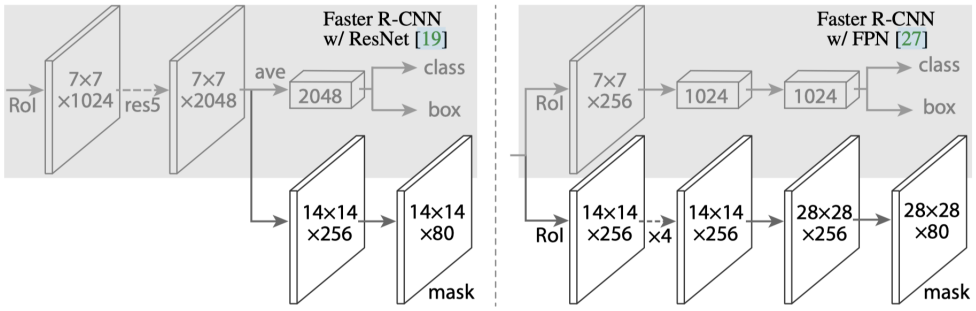

Figure 1: Mask R-CNN architecture [4].

objectness score. The second stage is the Fast R-CNN detector, which extracts features from the proposed regions of interest (RoI) given by the RPN with a pooling layer called RoIPool and then predicts the label and the bounding-box offset of each RoI.

Mask R-CNN maintains the first stage with a RPN but in the second stage the RoIPool layer is changed for a RoIAlign layer, which has better results in predicting segmentation masks. In order to achieve mask predictions Mask R-CNN adds a branch on the second stage in parallel to the classification and bounding-box regression tasks. This branch outputs a binary mask, which determines which pixels of the RoI belong to the object. This branch is a fully convolutional network (FCN) in contrast to [5] which uses fully connected layers to solve this task. The reason behind the usage of FCN is that fully connected layers lose the spatial structure of masks by compressing the information into vectors that lack spatial dimensions.

The architecture of the second stage of Mask R-CNN with a ResNet C4 backbone and with a ResNet backbone with Feature Pyramid Network (FPN) can be seen in figure 1. In the practical tasks of this report the ResNet backbone with FPN is used because it has better performance in [4].

The loss function used is a multi-task loss on each RoI for the three outputs of the network and is defined as $L = L_{cls} + L_{box} + L_{mask}$. The classification loss L_{cls} is the log loss for the true class of the object, the bounding box loss L_{box} uses a smooth L1 loss function to calculate the bounding-box offset [2] and the semantic mask loss L_{mask} is defined as an average over the binary cross-entropy loss over the pixels [4].

B. Dataset

The dataset used to train the model is the Penn-Fudan Database for Pedestrian Detection and Segmentation, which is used in [7]. The dataset contains 170 images with 345 instances of pedestrians. A label, a bounding box and a mask is assigned to each instance. The dataset has two classes, background and person. In figure 2, sample images and labeled masks are shown.

The dataset was divided into 3 parts, training, validation and test. A classical machine learning splitting was used due to the small size of the dataset. The training subset had 136 images and the validation and test subset had 17 images each.

C. Training Procedure

Two tasks were carried out in this report. The first task was finetuning the pretrained Mask R-CNN model with a ResNet-50 [8] backbone. The second task was changing the backbone of a Faster R-CNN model for object detection. In this subsection the training procedures of both tasks are presented.

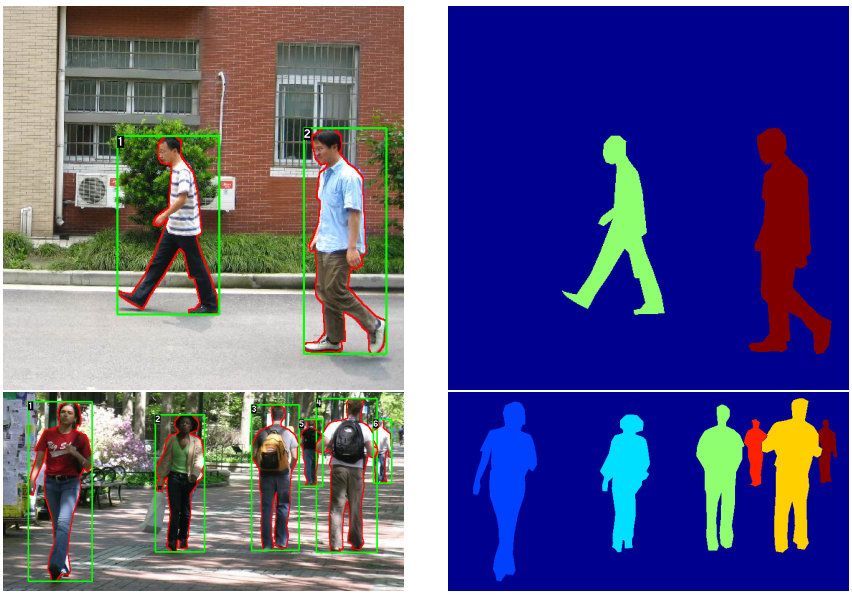

Figure 2: Sample images and labeled masks from Penn-Fudan Database.

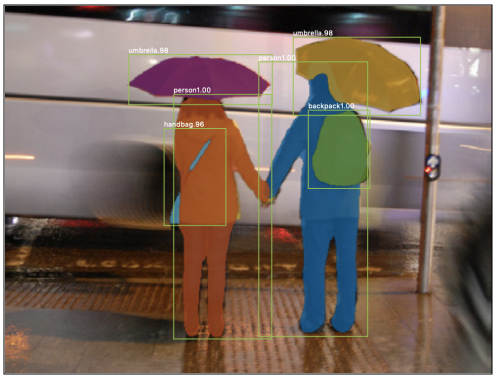

Figure 3: Mask R-CNN results on the COCO Dataset [4].

C.1. Finetuning

The model used for the finetuning is a Mask R-CNN model with a ResNet-50 backbone pretrained in the COCO dataset [9]. This dataset for object detection and instance segmentation contains photos of 91 object types. It contains 328k images with 2.5 millions labeled and segmented instances. Figure 3 is an example of a Mask R-CNN prediction on this dataset.

In order to achieve a finetune of the model, the last layer has to be adapted to the characteristics of the new dataset and the model is trained again on this dataset. In this application, the output layer was changed from the number of classes in the COCO dataset, 91 to the number in the Penn-Fudan dataset, 2. From the three outputs of the model, only the classification output depends on the number of classes. Therefore, only its last layer was modified.

Two different settings of hyper-parameters were used for the training. The first setting has a similar hyper-parameter selection to the one used in the pretraining [4]. A stochastic Gradient Descent (SGD) optimizer with momentum, a decaying learning rate starting at 0.005 and a weight decay of 0.0005 was used. The training was done during 10 epochs. The second setting achieved a slightly better performance with a longer training of 20 epochs. The hyper-parameters were an ADAM optimizer with a decaying learning rate starting at 0.01 and a weight decay of 0.005.

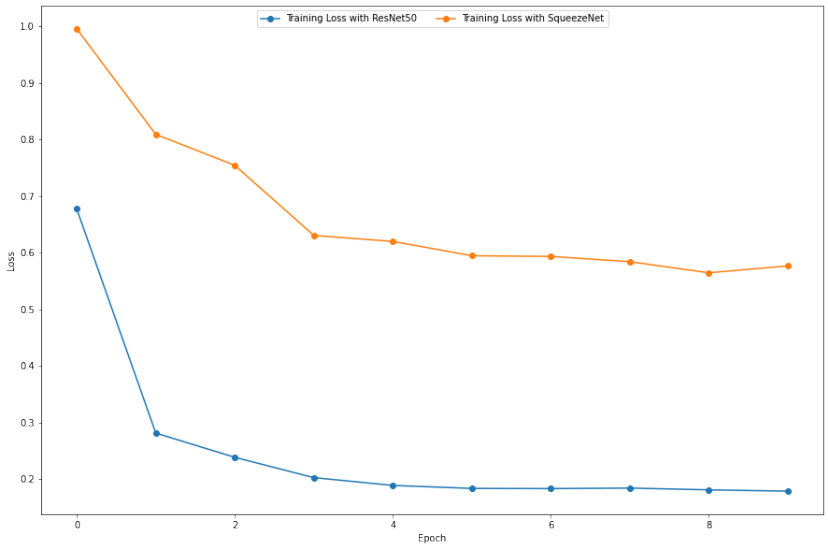

Figure 4: Training loss in both tasks during 10 epochs.

C.2. Faster R-CNN with a SqueezeNet backbone

The backbone used on the Faster R-CNN and Mask R-CNN model is ResNet-50, a residual CNN with 50 layers and more than 138 million parameters. For some applications this model is too big since some devices have limited memory. For this reason, the pretrained model SqueezeNet [10] was used as an alternative as a backbone. Moreover, another benefit of this model is an improvement on training time, which can be seen in subsection III.B.

The Faster R-CNN model was modified to add SqueezeNet as a backbone and it was trained during 10 epochs with this hyper-parameters setting. A SGD optimizer with momentum, a decaying learning rate starting at 0.005 and a weight decay of 0.0005.

III. Results

A. Training Convergence

The training loss is a good indicator that the performance of the algorithm is improving during training. The training duration is 10 epochs in order to avoid overfitting in this small dataset. The training convergence is shown in figure 4.

B. Training time

The objective of the second task was to reduce the memory usage of the model and improve the training time of the model by using a smaller backbone. Since Mask R-CNN adds only a small overhead in time compared to Faster R-CNN [4], the difference in time is mostly due to the different backbone. The following results are calculated in Google Colaboratory on a Tesla 4 GPU.

The training time per epoch of the Mask R-CNN model with ResNet-50 was 01:45 minutes while the time of the Faster R-CNN model with SqueezeNet was 00:21 minutes. The test time per image was 0.3455 seconds (2.89 FPS) for the Mask R-CNN with ResNet-50 and 0.0714 seconds (14 FPS) for the Faster R-CNN model with SqueezeNet.

Dataset	AP	AP_{50}	AP_{75}
COCO Dataset	33.6	55.2	35.3
Penn-Fudan (Standard H-P)	71.3	98.7	77.9
Penn-Fudan (Alternative H-P)	72.3	98.9	86.1

Table 1: Mask R-CNN results from our application compared to [4].

Figure 5: Results of the Mask R-CNN on the Penn-Fudan Dataset.

C. Performance

The performance was calculated on the average precision, which is the most used indicator in object detection and instance segmentation. The results of the instance segmentation of our application with both hyper-parameters setting is compared to the original model on the COCO Dataset in table 1. All the models use a backbone ResNet-50 with Feature Pyramid Network (FPN).

In figure 5 test images from the Penn-Fudan dataset are shown with their predicted masks.

Table 2 compares the performance of the object detection between the ResNet-50 backbone and the SqueezeNet backbone. The indicator used is the average precision of the bounding box since the Faster R-CNN model with SqueezeNet backbone does not output the segmentation of the instances.

IV. Discussion

The model trained on the Penn-Fudan dataset achieves impressive results compared to the original model. It outperforms the original model in all the indicators by a significant difference. However, the comparison is not realistic since the complexity of the Penn-Fudan dataset is lower to the COCO dataset. The COCO dataset contains 91 classes in contrast to the 2 classes of the Penn-Fudan Dataset. If two images from the datasets are compared, for example figures 2 and 3, the size of the objects varies more on the COCO dataset which makes the task harder. Nevertheless, the model achieves outstanding results given the small size of the Penn-Fudan dataset.

Backbone	AP^{BB}	AP^{BB}_{50}	AP^{BB}_{75}
ResNet-50	81.2	98.7	89.9
SqueezeNet	49.3	88.2	47.4

Table 2: Results of the different backbones on object detection

The backbone comparison shows a direct correlation between time of execution and performance of the model. The SqueezeNet backbone achieves shorter processing time but the performance on object detection is notably worse compared to the ResNet-50 backbone. The model with SqueezeNet would have a better performance if the dataset was larger since a change of backbone requires more training data than a finetuning in order to update all the weights of the neural network instead that the last layers of finetuning. The SqueezeNet backbone is appropriate to be used for applications on devices with limited memory or on real-time object detection with a faster GPU.

Lastly, other models will be presented as an alternative to Mask R-CNN in instance segmentation. Mask Scoring R-CNN improves the performance of Mask R-CNN by adding a network block that learns the quality of the mask [11]. Mesh R-CNN adds a layer that recreates the 3D shape of the object [12]. In this application this model would output the body shape of the persons in the images. For real-time applications the model YOLACT achieves instance segmentation above 30 FPS [13].

V. Conclusion

In this report the architecture of the state-of-the-art Mask R-CNN model and how the model was adapted to the Penn-Fudan dataset was discussed. The performance of the model on this dataset is outstanding but it should be taken into account that the dataset is simpler as the dataset used to pretrain the model. The SqueezeNet backbone was used to reduce the training and test time but this improvement in time had a performance penalty.

A larger and more complex dataset could be used in forthcoming research to compare realistically the performance between applications. Furthermore, it would allow to improve the performance of the model with the SqueezeNet backbone.

References

[1] R. Girshick, J. Donahue, T. Darrell, J. Malik, and U. C. Berkeley, "Rich feature hierarchies for accurate object detection and semantic segmentation," 2012. arXiv: arXiv:1311.2524v5.

[2] R. Girshick, "Fast R-CNN," 2015. DOI: 10.1109/ICCV.2015.169.

[3] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN : Towards Real-Time Object Detection with Region Proposal Networks," pp. 1–14, arXiv: arXiv:1506.01497v3.

[4] K. He, G. Gkioxari, P. Dollár, and R. Girshick, *Mask r-cnn*, 2018. arXiv: 1703.06870 [cs.CV].

[5] P. O. O. Pinheiro, R. Collobert, and P. Dollar, "Learning to Segment Object Candidates," in *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, Eds., vol. 28, Curran Associates, Inc., 2015. [Online]. Available: https://proceedings.neurips.cc/paper/2015/file/4e4e53aa080247bc31d0eb4e7aeb07a0-Paper.pdf.

[6] Y. Li, H. Qi, J. Dai, X. Ji, and Y. Wei, *Fully convolutional instance-aware semantic segmentation*, 2017. arXiv: 1611.07709 [cs.CV].

[7] L. Wang, J. Shi, G. Song, and I.-f. Shen, "Object Detection Combining Recognition and Segmentation," in *Computer Vision – ACCV 2007*, Y. Yagi, S. B. Kang, I. S. Kweon, and H. Zha, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 189–199, ISBN: 978-3-540-76386-4.

[8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *CoRR*, vol. abs/1512.03385, 2015. arXiv: 1512.03385. [Online]. Available: http://arxiv.org/abs/1512.03385.

[9] T.-y. Lin, C. L. Zitnick, and P. Doll, "Microsoft COCO : Common Objects in Context," pp. 1–15, arXiv: arXiv:1405.0312v3.

[10] F. N. Iandola, M. W. Moskewicz, K. Ashraf, S. Han, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and ¡1MB model size," *CoRR*, vol. abs/1602.07360, 2016. arXiv: 1602.07360. [Online]. Available: http://arxiv.org/abs/1602.07360.

[11] Z. Huang, L. Huang, Y. Gong, C. Huang, and X. Wang, "Mask Scoring R-CNN," arXiv: arXiv:1903.00241v1.

[12] G. Gkioxari, "Mesh R-CNN," arXiv: arXiv:1906.02739v2.

[13] D. Bolya, "Real-time Instance Segmentation," arXiv: arXiv:1904.02689v2.