Abstract

In the last few years detecting objects with cameras got more and more relevant in various applications. Also due to the coronavirus epidemic, wearing masks has become increasingly important. Checking if people wearing a mask is a time and personal costing task. We propose a face mask detector which detects if people wear a mask or not or if they wear a mask incorrectly. For this we compare the performance of a YOLO based network with a Faster R-CNN network. Since training such networks is very time consuming and there are already good performing object detectors which knowledge we can use as a starting point for a new model. This is called Transfer learning.

Contents

1 Introduction

In the last few years detecting objects with cameras got more and more relevant in various applications. Also due to the coronavirus epidemic, wearing masks has become increasingly important. In this paper, we introduce a face mask detection which uses state of the Art object detectors to detect if a person wears a face mask or not. We differentiate worn face masks in three classes:

1. **Correct worn**: mask is worn over the nose
2. **Incorrect worn**: mask is worn under the nose
3. **No face mask**: no face mask at all

Convolutional neural networks perform very well when the training and the testing data share the same domain. Since there exist already good performing Object-Detectors for various tasks we can transfer the knowledge of the Object-Detector to a new task, so we don't have to train the model from scratch. This is called transfer learning. For the transfer leraning we start with "You Only Look Once" (YOLO) and pretrained weights. This detector will then be finetuned on a new dataset consisting of images of people wearing a mask. With transfer learning we speed the learning process up by a big margin since we can keep the already good performing parts of the detector (e.g. Feature Extraction). YOLO is a state-of-the-art, real-time object detection system used in a broad band of applications. In the second part we will implement the same workflow with a region based convolutional neural network (R-CNN). For this we will use a Faster R-CNN because of the speed advantage against a normal R-CNN. The Faster R-CNN will be slower than the YOLO based network but more accurate.

2 Method

The goal of image detection algorithms is to locate the object of interest and draw a bounding box around it. A bounding box therefore describes the objects position and label. To train our detector we use transfer learning as stated before. For the implementation we will first finetune the YOLO and the Faster R-CNN network with the train set. After finetuning we use the test set to evaluate the detectors. Last we quantitatively and qualitatively evaluate the performance of the detectors. This workflow is also seen in Figure 1.

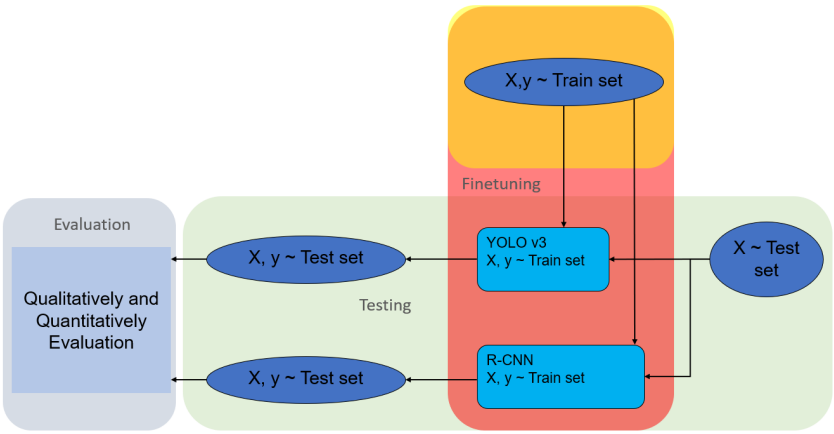


Figure 1: Our proposed workflow

2.1 Image Detection with YOLO and R-CNN

The goal of image detection algorithms is to locate and classify the object of interest. For our implementation of the mask detector we use a YOLO and Faster R-CNN based Network. In this abstract we briefly explain the difference between these two algorithms and how they work.

YOLO: This Network is a end-to-end deep learning model and was developed by Joseph Redmon. To split the input picture into a grid of cells and each cell classificates the object and predicts a bounding box is the approach of this method. Then this Algorithm has a large number of bounding boxes and they are simplified by a post-processing step. It only looks at the parts of the image which have high probabilities of containing the object of interest. In YOLO a single convolutional network predicts the bounding boxes and the class probabilities for these boxes. YOLO is orders of magnitude faster than other object detection algorithms but it struggles with small objects in the image [10]. Compared with R-CCNs this method is not so good in accuracy but its very fast in their detection of objects.

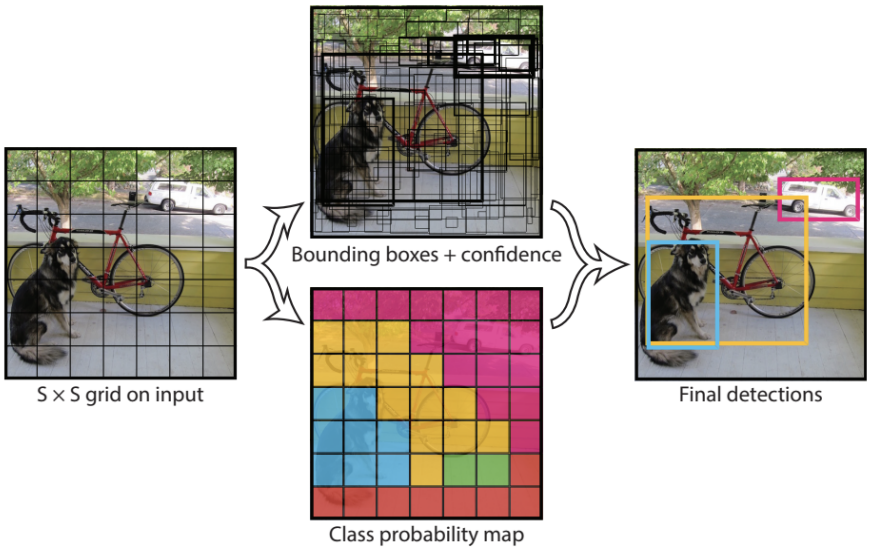


Figure 2: The workflow of YOLO to find important parts of the image and detect objects. Image taken from [10]

Faster R-CNN: R-CNN uses regions to localize the object within the image. For this the image is divided into approximately 2000 regions. These regions are called region proposals and are the bounding boxes that may contain an object. These region proposals are calculated by an algorithm called "selective search". The problem with R-CNN is that it takes a huge amount of time to train the network as you would have to classify 2000 region proposals per image [3]. In Difference to a normal R-CNN, Faster R-CNN uses a separate network to predict the region proposals. With a Region of Interest pooling (RoI pooling) Layer the proposed regions get reshaped and than used for classification. This is much faster than the selective search algorithm used for the region proposal [12]. So the detection speed is improved by two main steps. The first is to performing feature extraction over the image before proposing regions and the second is to replace the SVM with a softmax-layer. So the Fast R-CNN performed much better in speed. This method has a extremely high accuracy but its very slow in pace.

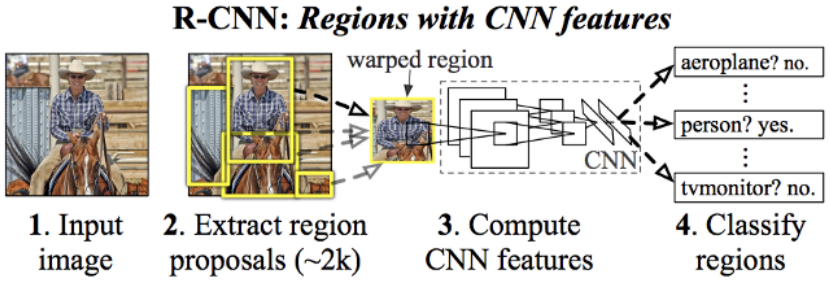


Figure 3: Object detection in R-CNN. Image taken from [3]

Resnet: Resnet is a popular Image Classification Model. Its a learning framework to improve the training of networks. During the training of deep networks the depth increases and the accuracy becomes saturate. This is the so called "degradation problem". With help of the so called "residual mapping" resnet solve this problem. Resnet is easy to optimize and the accuracy increases if the depth of the network increases.

Resnet vs VGG: VGG is mostly replaced by Resnet for extracting features. The biggest advantage of resnet is that it is a big framework which has a huge capacity to learn what is really needed. A huge capacity is very important by object detection. VGG do not have "batch normalization" and "residual connections". Resnet have this two things, so its easier to train deep models.

Resnet vs inception feature extractor: Inception feature extractors allows more efficient computations. The approach is to take multiple kernel filter sizes within the CNN and ordering them to operate on the same level. Inception Models are a way to reduce computational expense. The main difference between these two methods is, that inception focuses on computational costs and resnet focuses on computational accuracy.

2.2 Transfer Learning

The detection performance of already existing detectors trained on public datasets is often insufficient or there are no detectors trained on a specific dataset. But often the detectors are trained on a similar dataset or work good on similar classes. So we can use the knowledge of the detector and don't have to train the detector from scratch which saves a lot of training time. This is done by loading the already trained weights of a detector and finetune and retrain it with a new dataset of the target domain. This transfer of knowledge is also called transfer learning. The difference of standard Supervised Learning and Transfer Learning is seen in Figure 4.

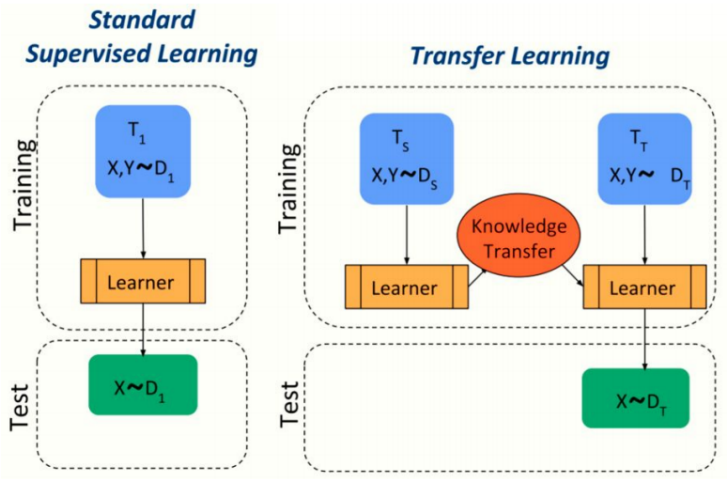


Figure 4: The difference between standard supervised learning and transfer learning. The blue field shows the task T and the dataset D where X is the input (images) and Y is the output (labels for the images). Image taken from [6]

2.3 Evaluation with mAP

A popular metric for measuring the accuracy of object detection is the mean average precision (mAP). mAP bases on the intersection over union (IoU) score between two bounding boxes. Figure 5 shows a graphical illustration of calculating the IoU.

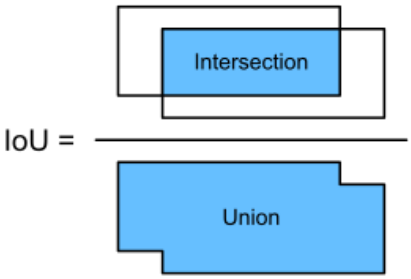


Figure 5: Illustration of the IoU measure. Image taken from [2]

mAP uses the calculation of IoU between a ground truth bounding box and a predicted bounding box. If the IoU is over a certain threshold we count the prediction as correct if not as wrong. With the help of the IoU we can calculate the true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN). We can now calculate the precision and the recall with Equation 1 and 2.

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

For the average precision (AP) we calculate the area under the precision-recall curve. For the mAP we calculate the AP for every class and average them.

3 Experiments

3.1 Setup

For the implementation of the transfer learning part we use Google Colab [4]. Colab is a product from Google Research and allows anybody to write and execute python code trough the browser. The GPU which we use in Colab is the Nvidia Tesla T4 with 16GB of memory.

3.1.1 YOLO Setup

For training the YOLO Network we used darknet, which is a library for python and contains an implementation of YOLO[1]. We trained the Network for 6000 iterations which is equivalent to about 10 hours.

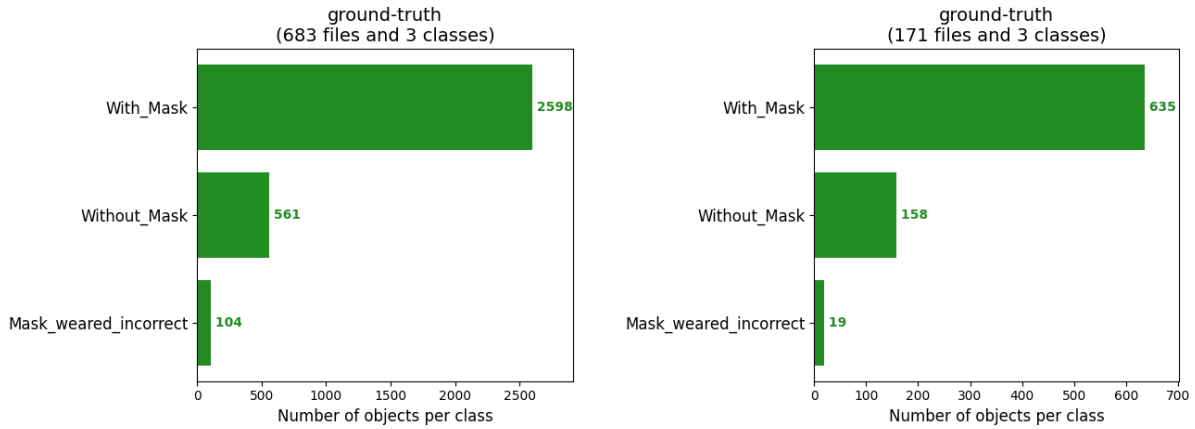
3.1.2 Faster-RCNN Setup

We use Tensorflow and PyTorch for our implementation. Tensorflow is a free and open-source software library for machine learning developed by Google [5]. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks. PyTorch is an open source machine learning library based on the Torch library [9]. It is developed by the Facebook's AI Research lab and is one of the preferred deep learning research platforms built to provide maximum flexibility and speed.
In PyTorch we used a pretrained faster-RCNN Modle with a resnet50 as feature extractor [11]. We also tried to train a faster-RCNN with the help of Tensorflow. There we used a faster-RCNN pretrained on the COCO Dataset [14][8]. We did not include results of the Tensorflow trained model since it yielded not good results. This is due to the small dataset. The PyTorch Model on the normal dataset was trained for 20 Epochs with 127 iterations. For the training we used a Github project as a base starting point and adopted it to detect the three classes and plot the results [16]. This is also equivalent to 10 hours of training. With the oversampled dataset the Model also was trained for 20 Epochs but with 2226 iterations per epoch. The iterations increased in comparison to before since we have more annotations so we have to decrease the batchsize for memory reasons. The iterations are calculated by the amount of annotated images divided by the batchsize we used a batchsize of two (in comparison before we used a batchsize of 5).

3.2 Dataset

The dataset which we use is called "Kaggle Face Dataset" [7]. This dataset contains real images of people wearing masks belonging to 3 classes namely with mask, without mask and mask worn incorrect. Kaggle is an open Source Dataset and allows us to use a test- and trainset combined with 853 files with different images to learn our algorithm. The distribution of the train and test set is seen in Figure 3.2. The images get randomly selected from the dataset. 80% of the images are used for training and 20% for testing.
An other dataset is "Moxa3K" which contains 3000 images, with 2800 images in the training set and 200 images in the testing set. The dataset contains the Kaggle data set of medical masksHsun 2020, which contains 678 images. These images are mainly from China, Russia and Italy region taken during the ongoing pandemic. Most of these pictures depict a crowded area with a large number of people [13].
We decided to used the "Kaggle Face Dataset" because it works without problems and it looks qualitatively quite good [13][15]. Also this is a smaller dataset and we want to see if its enough to use a small dataset. The third reason we choose the Kaggle Dataset was that it also contains objects of incorrectly worn faces. This dataset is used for the Training of the YOLO network

and the faster-RCNN network implemented in PyTorch. For the training of the faster-RCNN implemented in Tensorflow we also tried to train it wit the Moxa3K dataset, since it features a much higher amount of labels. On the contrary it only contains objects of correctly and incorrectly worn masks.

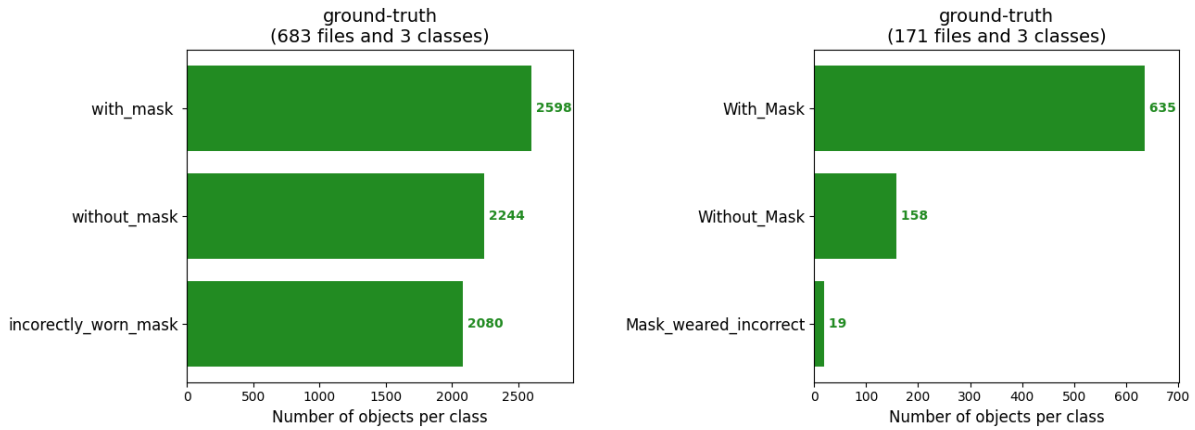


(a) (b)

Figure 6: Kaggle trainset (a) and testset (b) class distribution and ammount of annotations per class

3.2.1 Oversampling

Since the distribution of the classes in the dataset are not equal we created an oversampled version of the Dataset. When oversampling the annotations, we increased every annotation with "incorrectly worn masks" by 20 and every annotation with "without mask" by 4. This increases the amount of "incorrectly worn masks" and "without mask" annotations, and therefore reduced the imbalance between the three classes. We only had time to train the faster-RCNN with this dataset. The annotation distribution is seen in Figure **??**.



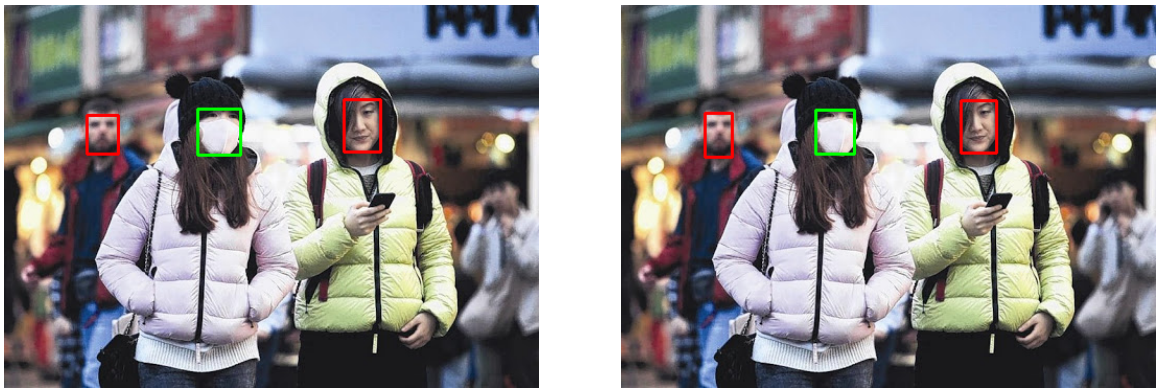
(a) (b)

Figure 7: Kaggle trainset oversampled (a) and testset (b) class distribution and ammount of annotations per class

3.3 Qualitatively Results YOLO

To demonstrate the performance of our trained detector we show some qualitative results of the trained YOLO detector. When we compare our detections to the ground truth the detection,

positions and classes are very accurate (Fig. 8). The detector performs good on images with a lot of faces. In Figure 9 we see the result on an image with a lot of small faces. Our trained detector, detects faces from different viewpoints with or without masks very good (Fig. 10). Also different type of masks get detected correctly (Fig. 10c).

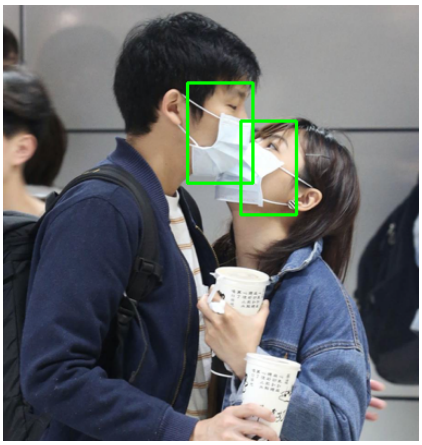


(a) Ground truth bounding boxes

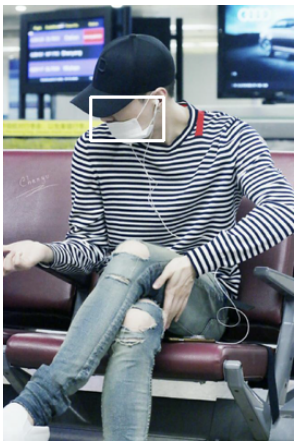
(b) Detection results

Figure 8: Ground truth (a) and detection result (b) compared. All faces got detected correctly. Red Boxes are faces without a mask, green boxes are faces with a mask

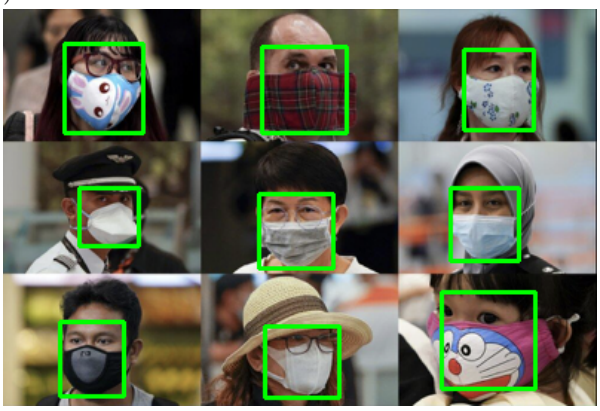
Figure 9: Detection of a lot of small faces with and without masks



(a)



(b)

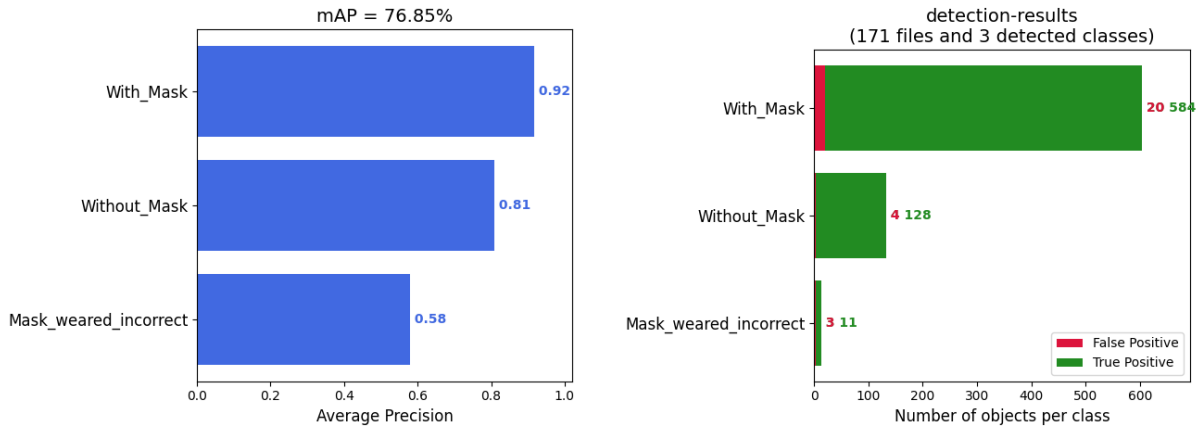


(c)

Figure 10: Different detection results: (a) Detection of faces with mask from the side. (b) Detection of partly covered face with incorrectly worn mask. (c) Detection of different mask types

3.4 Quantitatively Results YOLO

Previous we showed some qualitatively results. In this section we show quantitatively results. To get a measure how good our detector detects masks we use the mAP with an IoU of 0.5. Figure 11a shows the AP of the three different classes (with mask, without mask and mask worn incorrect). We see that almost all faces with a correctly worn mask got detected (AP of 92%). The faces without a mask got detected not as good but still very good (AP of 81%). The faces with incorrectly worn masks got not detected as good as the other two classes (AP of 58%). This again comes from the unbalanced dataset. Overall the mAP of our detector after training for 6000 iterations is 76.85%. In Figure 11b we see the true and false positive detections of the network. We see that there are only a few false positive detections.

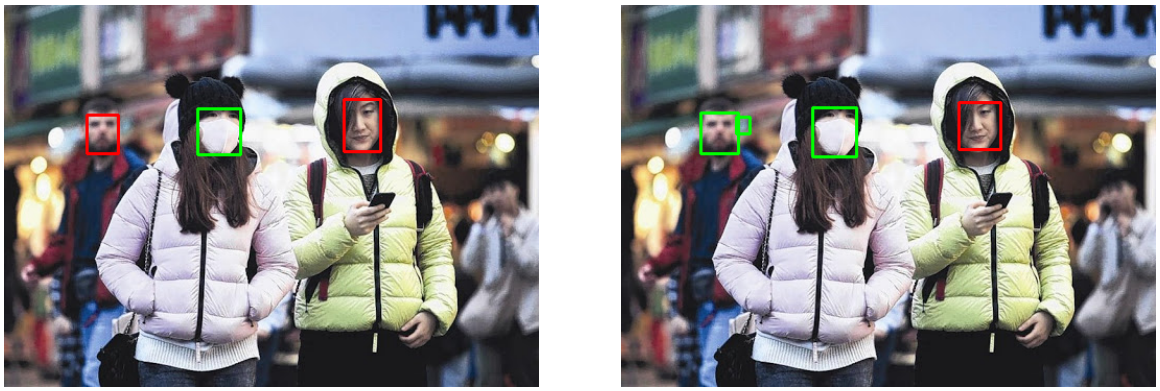


(a) (b)

Figure 11: Network evaluation (a) AP of the three classes. Overall mAP of 76.85% (b) True and False positives

3.5 Qualitatively Results Faster-RCNN

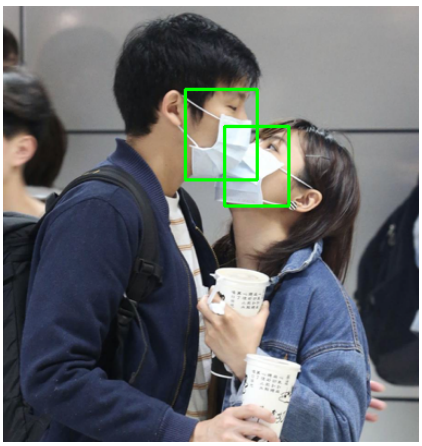
To demonstrate the performance of our trained detector we show some qualitative results of the trained faster-RCNN detector. When we compare our detections to the ground truth the detection, positions are very accurate but the classes are not always accurate (Fig. 12). This is also seen in the results of the quantitative evaluation. We also see some false detections (Fig. 12). The detector performs good on images with a lot of faces. In Figure 13 we see the result on an image with a lot of small faces. Our trained detector, detects faces from different viewpoints with or without masks very good (Fig. 14). Different type of masks get not detected as good (Fig. 14c). The model detected non of the incorrectly worn masks in the testset (Fig. 14b.



(a) Ground truth bounding boxes (b) Detection results

Figure 12: Ground truth (a) and detection result (b) compared. All faces got detected correctly. Red Boxes are faces without a mask, green boxes are faces with a mask

Figure 13: Detection of a lot of small faces with and without masks



(a)

(b)

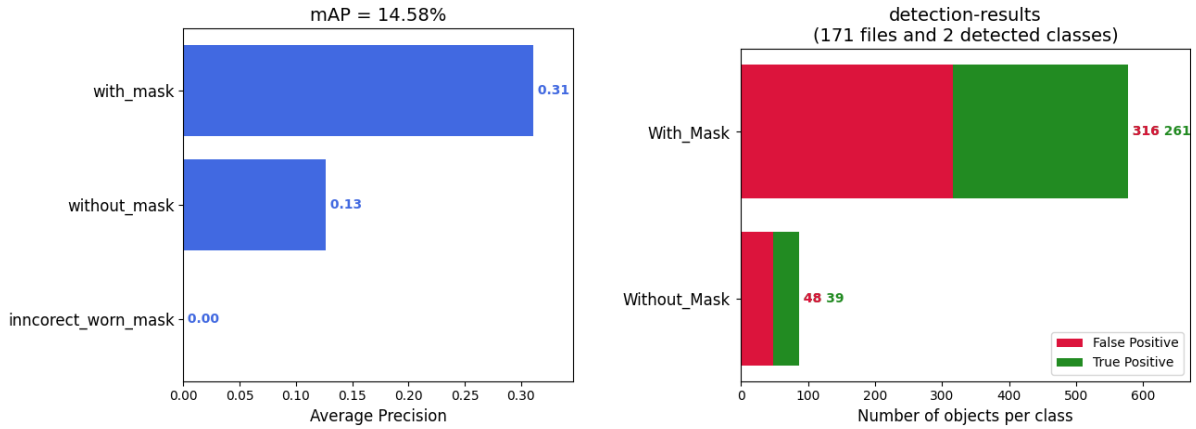
(c)

Figure 14: Different detection results: (a) Detection of faces with mask from the side. (b) Detection of partly covered face with incorrectly worn mask. (c) Detection of different mask types

3.6 Quantitatively Results Faster-RCNN

Previous we showed some qualitatively results. In this section we show quantitatively results. To get a measure how good our detector detects masks we use the mAP with an IoU of 0.5. Figure 15a shows the AP of the three different classes (with mask, without mask and mask worn

incorrect). We see that not a lot of mask got detected. Especially incorrect worn mask never got detected. Also faces with out masks got only detected 13% of the time while faces with mask got detected 31% of the time. This is due to the small amount of labels with this two classes in the dataset. In Figure 15b we also see that there is a high amount of false positives. This is also seen in the qualitative results where a lot of faces without a mask got detected as a face with a mask.



(a) (b)

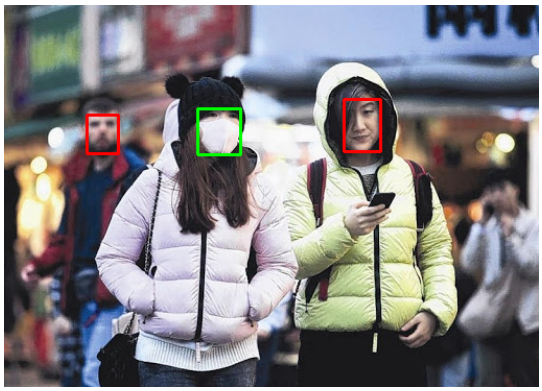
Figure 15: faster-RCNN Network evaluation (a) AP of the three classes. Overall mAP of 14.58% (b) True and False positives

3.7 Results with Oversampled Dataset

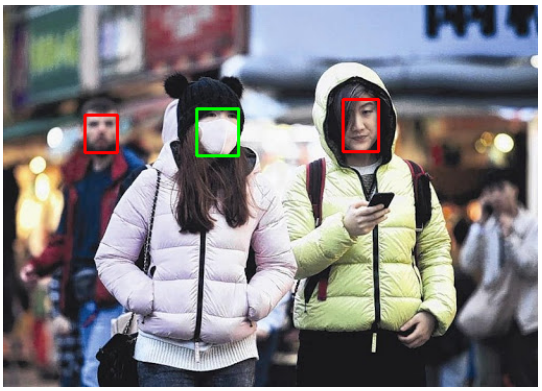
Since we only had time to retrain one network we choose the faster-RCNN network because it performs way worse than the YOLO network which performs already very good.

3.7.1 Qualitatively Results Faster-RCNN trained on Oversampled Dataset

To demonstrate the performance of our trained detector we show some qualitative results of the trained faster-RCNN detector. When we compare our detections to the ground truth the detection, positions and classes are always very accurate (Fig. 16). The detector performs good on images with a lot of faces. In Figure 17 we see the result on an image with a lot of small faces. Our trained detector, detects faces from different viewpoints with or without masks very good (Fig. 18). Different type of masks get not detected as good as the YOLO trained network (Fig. 18c). Also incorrect worn masks get detected very good (Fig. 18b.



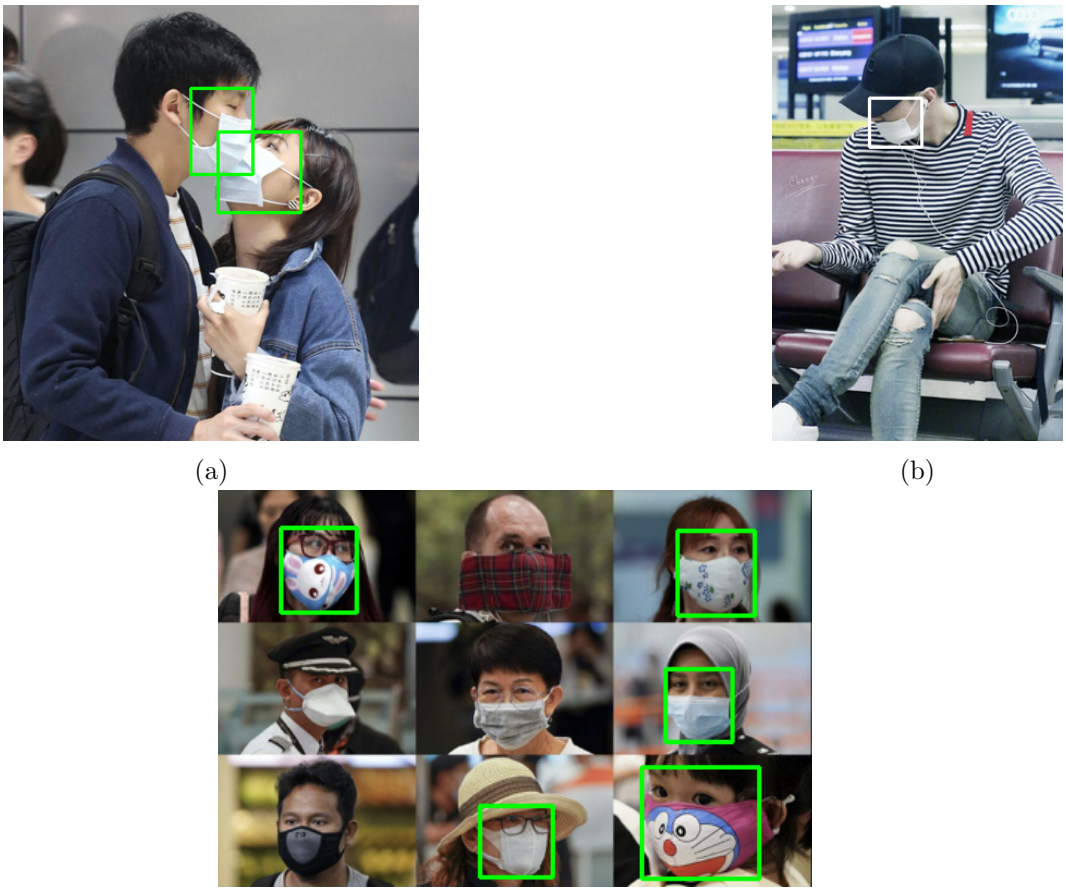
(a) Ground truth bounding boxes



(b) Detection results

Figure 16: Ground truth (a) and detection result (b) compared. All faces got detected correctly. Red Boxes are faces without a mask, green boxes are faces with a mask

Figure 17: Detection of a lot of small faces with and without masks



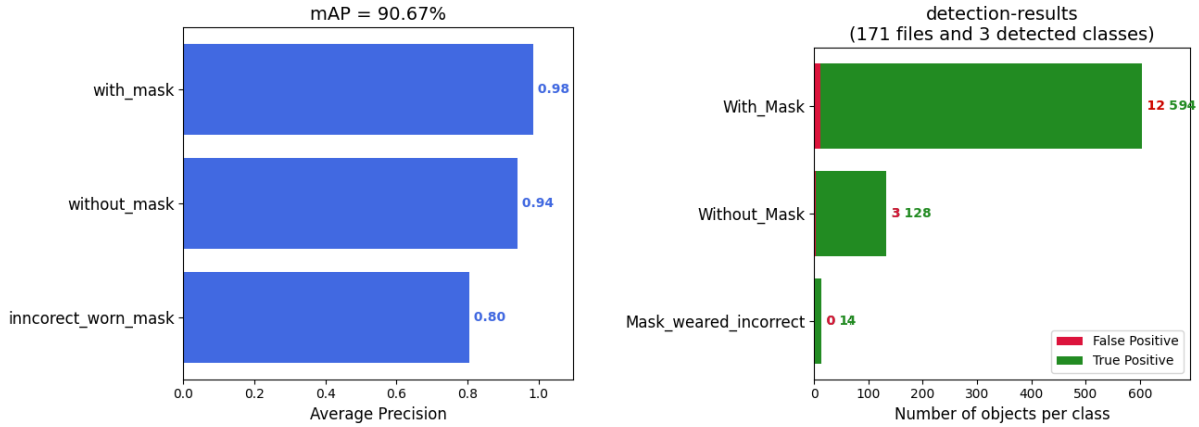
(a) (b)

(c)

Figure 18: Different detection results: (a) Detection of faces with mask from the side. (b) Detection of partly covered face with incorrectly worn mask. (c) Detection of different mask types

3.7.2 Quantitative Results Faster-RCNN trained on Oversampled Dataset

Previous we showed some qualitatively results. In this section we show quantitatively results. To get a measure how good our detector detects masks we use the mAP with an IoU of 0.5. Figure 19a shows the AP of the three different classes (with mask, without mask and mask worn incorrect). We see that almost all classes got detected very good. In comparison with the non oversampled dataset we see that the AP of all the classes increased significantly especially the AP of incorrect worn masks. This is also seen in the qualitative results. As seen in Figure 19b the false positive rate is very low.



(a) (b)

Figure 19: faster-RCNN Network with oversampling evaluation (a) AP of the three classes. Overall mAP of 90.67% (b) True and False positives

3.8 Detection Problems with YOLO

As we saw in Section 3.2 the dataset is unbalanced regarding the different classes. There are only a few objects of the incorrect worn masks class. This is also seen in the result that some incorrect worn masks got detected as correctly worn or some covered faces without a mask got predicted as faces with incorrectly worn masks (Fig. 20). As we see in Figure 21 sometimes very small objects in the background get not detected.

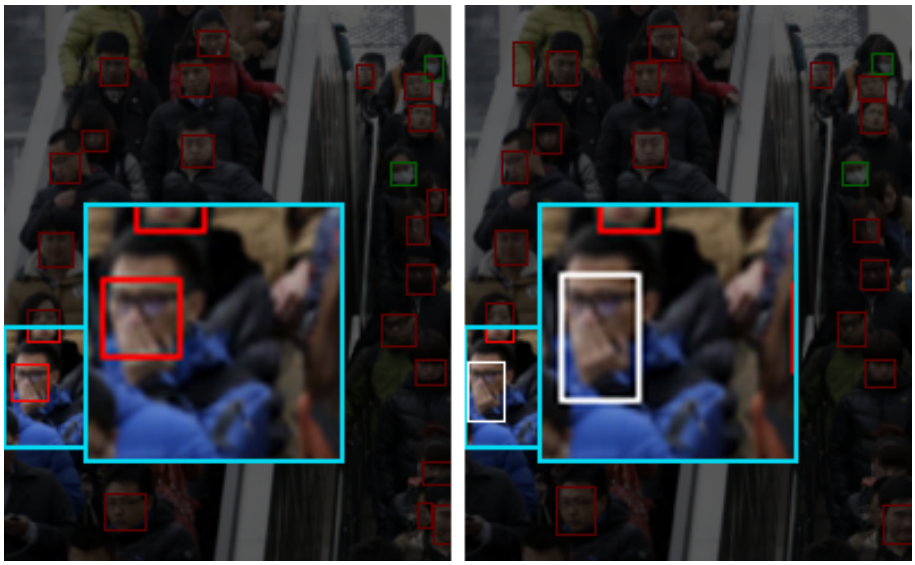


Figure 20: A man which covers his face with his hands. Left the ground truth with the correct label no mask. Right the detection of our detector, which predicts a wrong worn mask instead of no mask.

Figure 21: Small faces in the background got not detected. Left ground truth and right detection of our detector

3.9 Detection Problems with Faster-RCNN

With the faster-RCNN model trained on the normal dataset, the same problems as in the YOLO model arise. Further the faster-RCNN model have problems with detecting darker face masks (Fig. 22. This is probably due to the smaller amount of dark than light face masks. Further the model does not detect incorrectly worn face masks very good. Most of the time incorrectly worn face masks get detected as correctly worn (Fig. 23). This problem is also because of the unbalanced dataset. With the oversampled Dataset the last problem got solved.

Figure 22: Darker face masks get much worse detected than lighter

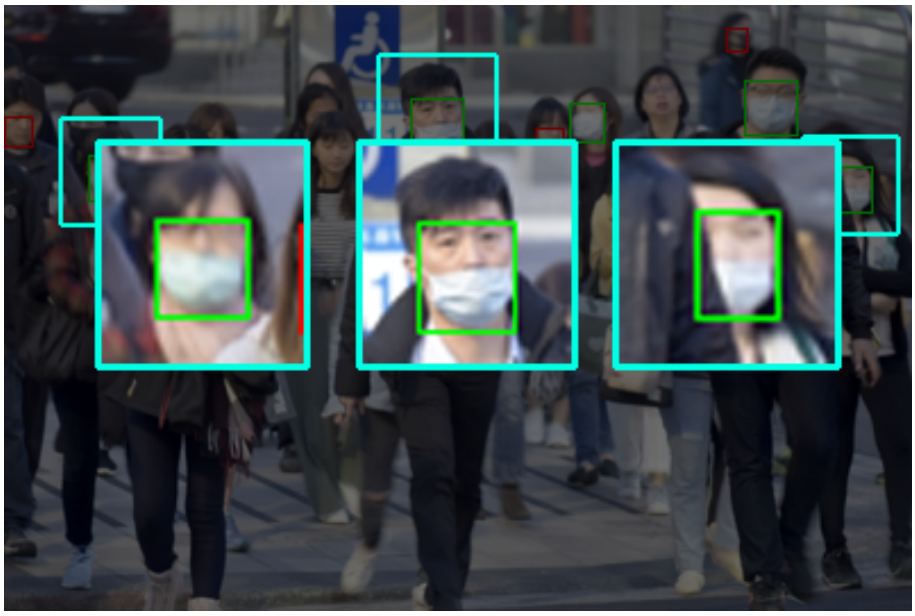


Figure 23: Problem with detecting incorrectly worn face masks

3.10 YOLO vs. Faster-RCNN

In previous sections we saw the quantitative and qualitative results of the trained networks. In comparison the YOLO network performs better than the faster-RCNN network trained on the same dataset. The faster-RCNN network performs much worse when it comes to detect incorrectly worn face masks. This is seen in Figure 10b and 14b. We expected that the faster-RCNN outperforms the YOLO network when it comes to detect masks far in the background. But also there the YOLO network is better seen in Figure 9 and 13. When we compare the results of the faster-RCNN trained on the oversampled dataset we see that the performance increased significantly and is with a mAP of 90.67%, 13.82% better than the YOLO trained Network and 76.09% better than the faster-RCNN on the not oversampled dataset.
As expected the faster-RCNN performs better on smaller images in the background (Fig. 24)

Figure 24: Example of small object detected in the backgroung. Left detection results of the faster-RCNN trained network on the oversampled dataset. Right detection results of the YOLO trained network

When it comes to detecting different types of masks the YOLO trained Network is still better (Fig. 25). This is since the YOLO network in comparison to the faster-RCNN need a lot more data to train the network. Since there are only a few annotation with special masks the faster-RCNN network does not perform very good on them.

Figure 25: Examples of different Face mask types. Below detection results of the YOLO trained network. Above detection results of the faster-RCNN network trained on the oversampled dataset

In Table 3.10 we see the AP scores of the different classes and Networks. There we see that the faster-RCNN network trained on the oversampled dataset performs the best.

	AP with mask in %	AP without maks in %	AP incorrect worn mask in %
YOLO	92	81	58
f-RCNN	31	13	0
f-RCNN oversampled	**98**	**94**	**80**

4 Conclusion

The conclusion of our project is that we found out that the Yolo-Network performs better than the RCNN-Network which depends also on the used dataset(oversampled or not oversampled dataset). The most occurred problem of the Yolo Network is that incorrectly worn masks are detected as correctly worn face masks. The Faster-RCNN Network has the same problem as described above and it has also problems with detecting darker face masks.

The problem with incorrectly worn masks, which occur in the Yolo and Faster-RCNN Network has been solved by Faster-RCNN trained on oversampled Dataset. Faster-RCNN performs better on smaller images in background but Yolo was better in detecting different types of masks. So as conclusion we learned that the Faster-RCNN trained on oversampled Dataset performs better than the YOLO-Network and this performs better than Faster-RCNN trained on not oversampled Dataset.

5 Future Work

5.1 Improve Results

As we seen before the network trained on the oversampled dataset performs better than the model on the non oversampled dataset. Using the oversampled data also for the YOLO network

would yield even better results than it already did.
To improve the results of our proposed detector even further without using a bigger new dataset, some data augmentation procedures like changing hue, saturation, cropping, and transposing the images to increase the total image amount would also increase the detector performance.

References

[1] AlexeyAB. Yolo v4, v3 and v2 for windows and linux. https://github.com/AlexeyAB/darknet. Accessed: 27.08.2020. 6

[2] Aston Zhang and Zachary C. Lipton and Mu Li and Alexander J. Smola. Dive into deep learning. https://d2l.ai/chapter_computer-vision/anchor.html#intersection-over-union, 2020. 5

[3] Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524, 2013. 3, 4

[4] Google. Colab. https://colab.research.google.com/notebooks/intro.ipynb. Accessed: 27.08.2020. 6

[5] Google. Tensorflow. https://www.tensorflow.org/. Accessed: 14.01.2021. 6

[6] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. Continual Learning: A Comparative Study on How to Defy Forgetting in Classification Tasks. *arXiv CoRR*, abs/1909.08383, 2019. 5

[7] Larxel. Face mask detection. https://www.kaggle.com/andrewmvd/face-mask-detection?select=annotations. Accessed: 01.12.2020. 6

[8] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common Objects in Context. In *Proceedings of the IEEE European Conference on Computer Vision (ECCV)*, Zürich, 2014. 6

[9] PyTorch. Pytorch. https://pytorch.org/. Accessed: 14.01.2021. 6

[10] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015. 3

[11] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks, 2016. 6

[12] Rohith Gandhi. R-CNN, Fast R-CNN, Faster R-CNN, YOLO Object Detection Algorithms. https://bit.ly/38es4Jb, 2018. Accessed: 01.12.2020. 3

[13] Biparnak Roy, Subhadip Nandy, Debojit Ghosh, Debarghya Dutta, Pritam Biswas, and Tamodip Das. Moxa: A deep learning based unmanned approach for real-time monitoring of people wearing medical masks. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7382322/. Accessed: 14.01.2021. 6

[14] Tensorflow. Tensorflow Detection Model Zoo. https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md. Accessed: 12.01.2021. 6

[15] Zhongyuan Wang, Guangcheng Wang, Baojin Huang, Zhangyang Xiong, Qi Hong, Hao Wu, Peng Yi, Kui Jiang, Nanxi Wang, Yingjiao Pei, Heling Chen, Yu Miao, Zhibing Huang, and Jinbi Liang. Masked face recognition dataset and application, 2020. 6

[16] Mengliu Zhao. Face Mask Detection using Faster RCNN. https://github.com/adoskk/KaggleFaceMaskDetection. Accessed: 14.01.2021. 6