Object Detection with Single Shot MultiBox Detector

Ruprecht-Karls-Universität Heidelberg

Heidelberg Collaboratory for Image Processing

Object Recognition and Image Understanding
Prof. Dr. B. Ommer

Supervisor: Nikolai Ufer

Paul Warkentin

p.warkentin@stud.uni-heidelberg.de

July 26, 2018

Abstract

Object detection has been a major topic of image understanding and deep learning in general for many years now. Different applications of object detection are possible, like vehicle detection in autonomous driving or people counting at an event. Most applications require fast detection of the objects in a given image.

In the present work, we implement such an object detector. Single Shot MultiBox Detector (in short: SSD) delivers great precision while being faster than other neural networks. We discuss two different types of the SSD network, one for images of size 300×300 and the other one for images of size 512×512.

Contents

1 Introduction

Object detection has become more important over the last few years. The analysis of street scenes for autonomous driving or the counting of people at an event are just a few of many applications. By now, there are many different algorithms and concepts for object detection published. Just a few are lightweight enough to run them on live inference that results in an appropriate frame rate. Other object detection networks are for example the You Only Look Once (in short: YOLO) network or the Faster R-CNN [1] network. Both models are discussed in Section 3 as comparison to the implementation of this project.

We discuss the Single Shot MultiBox Detector (in short: SSD) [2] in this project as it is more efficient on live inference while improving the accuracy at the time compared to other object detection models like Faster R-CNN and YOLO. Let us take a look at the components in the name.

- **Single Shot:** this means that the tasks of object classification and localization are done in a single forward pass of the network.
- **MultiBox:** this is the name of a technique for bounding box regression developed by *Szegedy et al.* [3].
- **Detector:** the network is an object detector that also classifies those detected objects.

First, we will discuss the implementation of the model for image inputs of size 300×300 and 512×512. After that, the already mentioned algorithms YOLO and Faster R-CNN are shortly described before we will take a look at some results given by the trained network of this project.

2 Implementation

We will discuss two different sizes of SSD networks in this project. The SSD 300 and SSD 512 networks accept image inputs of size 300×300 and 512×512 respectively. The smaller SSD 300 network is not as precise as the bigger SSD 512 network but can therefore score with a better inference time.

Some of the tricks in this implementation will be based on the original Caffe implementation [4]. All presented methods are implemented in Python 3.6[1] making prevalent use of TensorFlow 1.9[2] and NumPy 1.14[3]. The source code is publicly available at `https://github.com/paulwarkentin/tf-ssd-vgg`. The `README.md` provides additional information on how to get started.

[1]`https://www.python.org/`
[2]`https://www.tensorflow.org/`
[3]`https://www.numpy.org/`

2.1 Model structure

The model of the SSD network consists of three parts.

First, the input image is fed into a classifier network, here called the base network. In this project we use the VGG 16 network for this purpose. The fully connected layers in the standard architecture are replaced by convolutional feature layers.
Secondly, the output of the modified base network is fed into a series of extra feature layers. These layers decrease in size and allow the prediction of detections at multiple scales.
Finally, the output of the extra feature layers plus a selected layer from the base network are used as detection predictors. Each of these layers can produce a fixed set of predictions using a new set of convolutional layers.

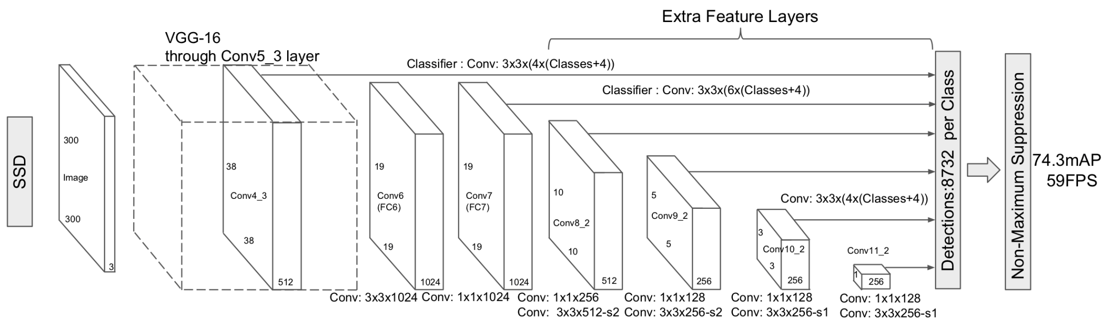


Figure 1: *Architecture of the SSD 300 network.*

The structure above is about the SSD 300 network. The slightly bigger SSD 512 network has two more extra feature layers.

2.2 MultiBox priors

MultiBox starts with a fixed set of default anchor boxes as priors for the predictions and attempt to approach the groundtruth bounding boxes using the priors.

For each feature layer, we compute default anchor boxes for a given set of aspect ratios and scales. The number of anchor boxes for a feature layer of size (h, w) of b different scales and ratios can be calculated by $b \cdot h \cdot w$. In the implementation of this project we use the following presets for the SSD 300 network.

Layer	**Height**	**Width**	**# Ratios**
`conv4_3`	38	38	4
`conv7`	19	19	6
`conv8_2`	10	10	6
`conv9_2`	5	5	6
`conv10_2`	3	3	4
`conv11_2`	1	1	4

Table 1: *Presets for the SSD 300 network.*

This sums up to a total of 8732 default anchor boxes. The SSD 512 network is defined by the following presets.

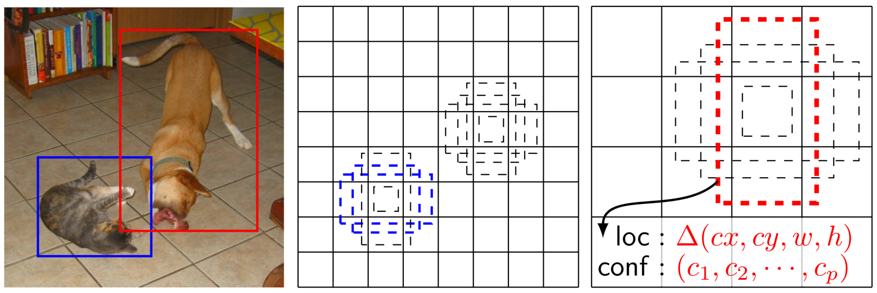
Layer	**Height**	**Width**	**# Ratios**
`conv4_3`	64	64	4
`conv7`	32	32	6
`conv8_2`	16	16	6
`conv9_2`	8	8	6
`conv10_2`	4	4	6
`conv11_2`	2	2	4
`conv12_2`	1	1	4

Table 2: *Presets for the SSD 300 network.*

This sums up to a total of 24564 default anchor boxes.

2.3 Matching strategy

Before the training step, we need to match the groundtruth boxes to the previous calculated default anchor boxes. The image below describes the logic of this step well. The bounding box of the dog is matched to the red dotted anchor box in the 4×4 feature layer but not to any shown box in the 8×8 feature layer. The criteria of a match is given by the Intersection of Union (in short: IoU) and a given threshold, here $\frac{1}{2}$. The IoU is dependent on the location, size and aspect ratios of the boxes. The bounding box of the cat on the other hand is matched to the blue dotted anchor box on the 8×8 feature layer.


loc : $\Delta(cx, cy, w, h)$
conf : $(c_1, c_2, \cdots, c_p)$

(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map

Figure 2: *Matching default anchor boxes.*

In this project, we follow a certain strategy for finding those matches. First, we match each groundtruth box to the default anchor box with the highest IoU. Then, each other default anchor box is matched to any groundtruth box with an IoU over a given threshold, here $\frac{1}{2}$.

2.4 Training objective

The loss function is the sum of two components.

- **Confidence loss:** this loss measures the confidence of the predicted class of the computed bounding box. To compute this loss, categorical softmax cross-entropy is used.
- **Localization loss:** this loss measures the distance of the computed bounding box from the groundtruth box. To compute this loss, the L2-Norm is used.

Let $x_{ij}^p \in \{0, 1\}$ be the status of matching the i-th default anchor box to the j-th groundtruth box of class p. The set X_P contains all default anchor boxes that were matched to a groundtruth box, the set X_N on the other hand contains all unmatched anchor boxes.

The confidence loss is calculated for positive and negative samples by

$$L_{conf}(x, c) = -\sum_{i \in X_P} x_{ij}^p \log(\hat{c}_i^p) - \sum_{i \in X_N} \log(\hat{c}_i^0),$$

where $\hat{c}_i^p$ is the softmax function over confidences c_i^p of class p,

$$\hat{c}_i^p = \frac{\exp(c_i^p)}{\sum_p \exp(c_i^p)}.$$

After the matching step, most of the default anchor boxes are not matched with any object, i.e. they are negatives. To compensate this imbalance of positive and negative samples, we perform hard negative mining. We sort the confidence losses of the negative samples in descending order and take only the samples with the highest loss so that the ratio of positive to negative samples is 1:3.

The localization loss is calculated for positive samples by using the predicted box b and the groundtruth box g,

$$L_{loc}(x, b, g) = \sum_{i \in X_P} \sum_{m} x_{ij}^p \text{smooth}_{L1}(b_i^m - \hat{g}_j^m),$$

where $m \in \{cx, cy, w, h\}$ and $\hat{g}_j$ is the SSD encoding of the j-th groundtruth box using the i-th default anchor box d_i defined by

$$\hat{g}_j^{cx} = \frac{1}{d_i^w}(g_j^{cx} - d_i^{cx}), \quad \hat{g}_j^{cy} = \frac{1}{d_i^h}(g_j^{cy} - d_i^{cy}),$$
$$\hat{g}_j^w = \log\left(\frac{g_j^w}{d_i^w}\right), \qquad \hat{g}_j^h = \log\left(\frac{g_j^h}{d_i^h}\right).$$

The smooth L1-Norm is defined as

$$\mathrm{smooth}_{L1}(y) = \begin{cases} \frac{1}{2}y^2 & \text{if } |y| \leq 1 \\ |y| - \frac{1}{2} & \text{otherwise.} \end{cases}$$

The overall training objective is then calculated by adding and normalizing the latter two losses,

$$L(x, c, b, g) = \frac{1}{N}(L_{conf}(x, c) + L_{loc}(x, b, g)).$$

If the number of positive samples N is equal to zero, we set the loss to zero.

2.5 Data augmentation

Before training, various data augmentation tricks are randomly applied to the input images to increase the number of samples and to reduce overfitting of the network.

First, the colors of the image are randomly distorted choosing one of four distortion packages. Each package adjusts the brightness, saturation, hue and contrast of the image with random parameters. Each package applies the distortions in a different order. After the color distortions, the RGB channels of the image are randomly reordered.

Secondly, the image is expanded, i.e. the image is randomly placed on a canvas up to four times bigger than the image itself. The background color of the canvas is set to the mean value of the dataset that was used to train the base network. All bounding boxes are transformed accordingly.

Thirdly, the image is randomly cropped so that the cropped image contains at least $\frac{1}{2}$ of any bounding box. All bounding boxes are transformed accordingly.

Lastly, the image is randomly flipped horizontally.

2.6 Evaluation metric

To evaluate the training results the mean average precision (in short: mAP) is used. mAP is often used in object detection. The metric is calculated using the precision and the recall of objects of one class.

- **Precision:** measures how accurate the prediction is, i.e. the percentage of the positive predictions that are correct,

$$\text{Precision} = \frac{TP}{TP + FP}.$$

- **Recall:** measures how good all the positives were found,

$$\text{Recall} = \frac{TP}{TP + FN}.$$

The average precision is then calculated by computing the area under thre precision-recall curve. This value is computed for every class. The mAP is then the mean of the average precision values of all classes.

3 Comparison

Next to the SSD network are many other object detection algorithms published. In this section, we will take a look at two other networks.

3.1 Faster R-CNN

Faster R-CNN [1] is a faster version of the Region-based Convolutional Neural Network (in short R-CNN). R-CNN uses an algorithm called Selective Search to propose objects and reduces the number of region proposals to around 2000. For each regional proposal, CNN features are calculated and finally classified. Faster R-CNN replaces that Selective Search with a convolutional network as the Selective Search is very slow. The new convolutional network is called Region Proposal Network (in short RPN) that generates regions of interests.

With Faster R-CNN, the idea of anchor boxes comes to life. For each location in a feature map extracted by convolutional layers from the image, anchor boxes with 3 different scales and 3 different aspect ratios are computed, resulting in 9 anchor boxes per location. A sliding window is then run spatially on these feature maps which are then fed to another network for classification and regression. The regression network predicts a bounding box, the classification network outputs a probability indicating whether the predicted box contains an object or background.

3.2 YOLO

You Ony Look Once (in short YOLO) [5] is a Regression-based object detector. We will look into YOLOv2 here.

YOLO takes an image of size 448×448 as input and feeds it to a convolutional neural network with a tensor of $(7, 7, 1024)$ as output. This tensor is then fed into two fully connected layers that performs linear regression. The final output is then a tensor of size $(7, 7, 30)$. Each cell in the 7×7 grid predicts two bounding boxes and its confidence scores. The score measures how accurate the bounding box is and how likely the box contains an object or background.

4 Results

In this section, the results of the training from the projects' implementation is discussed. The VGG 16 model was initialized with the pre-trained weights on the

ILSVRC-2012-CLS[4] image classification dataset. The network was trained for research by TensorFlow-Slim [6].

4.1 Training

The SSD 300 network was trained for a total of 100k steps with batch size 32. One step is defined as one forward pass and one backward pass of the network. The loss was minimized using the Momentum optimizer with momentum value 0.9. For the learning rate decay, we used the following rule: the first 70k steps were trained with learning rate 1e-3, the next 15k steps with 1e-4 and the last 15k steps with 1e-5. This training process took about 24 hours.

The SSD 512 network was trained with batch size 16, the same optimizer settings and with learning rate 1e-3 for 100k steps, 1e-4 for 20k steps and 1e-5 for another 20k steps.

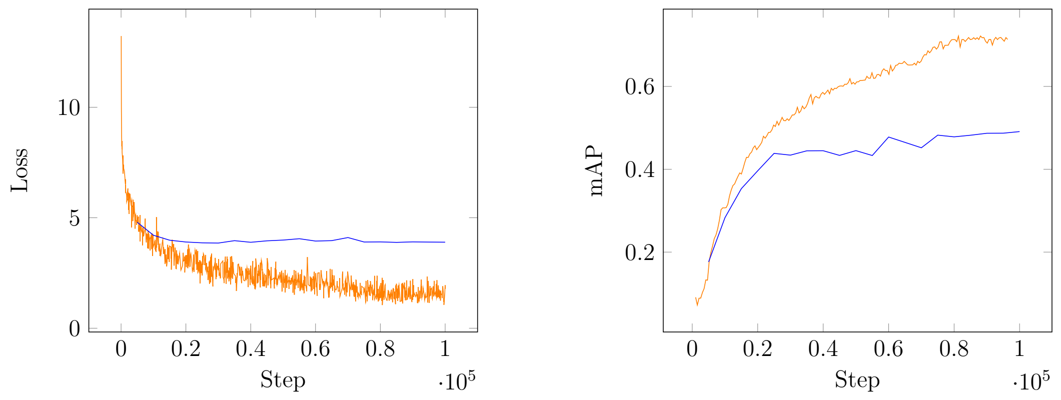


Figure 3: *SSD 300: Train (orange) and evaluation (blue) loss and mAP. The network starts to overfit although the dataset was heavily expanded using many data augmentation tricks.*

Both networks were trained on the Pascal VOC 2007 + 2012 trainval dataset which is the union of the Pascal VOC 2007 [7] and Pascal VOC 2012 [8] trainval sets. The trained network was finally evaluated on the Pascal VOC 2007 test set.

	Method	Dataset	mAP
	Fast R-CNN	VOC 2007+2012	70.00%
	Faster R-CNN	VOC 2007+2012	76.80%
	SSD 300	VOC 2007+2012	74.30%
	SSD 512	VOC 2007+2012	76.80%
×	SSD 300	VOC 2007+2012	49.09%
×	SSD 512	VOC 2007+2012	54.37%

Table 3: *mAP results compared to the original paper. The rows marked with × are the results from this project.*

[4] `http://www.image-net.org/challenges/LSVRC/2012/`

When we look at the average precision values of each class, we can see that the network of this project can badly detect objects of the classes bottle, chair and potted plants. The following table compares the precision values of the implementation of this project to the values from the paper.

	Method	Data	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow
	SSD300	07+12	74.3	75.5	80.2	72.3	66.3	47.6	83.0	**84.2**	**86.1**	54.7	78.3
	SSD512	07+12	76.8	82.4	84.7	78.4	73.8	53.2	**86.2**	**87.5**	**86.0**	57.8	83.1
×	SSD300	07+12	49.1	62.5	53.6	43.5	40.5	15.9	60.8	53.5	**71.2**	22.7	51.1
×	SSD512	07+12	54.4	63.0	64.2	53.6	41.0	27.8	63.8	**71.2**	**80.7**	33.6	48.2

	Method	Data	mAP	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
	SSD300	07+12	74.3	73.9	**84.5**	85.3	82.6	76.2	48.6	73.9	76.0	83.4	74.0
	SSD512	07+12	76.8	70.2	84.9	85.2	83.9	79.7	50.3	77.9	73.9	82.5	75.3
×	SSD300	07+12	49.1	41.8	60.9	**62.6**	60.3	51.0	22.9	43.3	41.8	**70.2**	51.8
×	SSD512	07+12	54.4	50.8	62.1	54.6	62.1	61.4	27.2	53.5	43.3	**71.0**	54.4

Table 4: *Average precision results compared to the original paper. The 3 top values per method are highlighted. The rows marked with × are the results from this project.*

Some reasons for the differences in the implementations may be the implementation of the pre-processing like data augmentation, computation of default anchor boxes and connected with that the matching strategy as well as the post-processing like selecting the final bounding boxes.

4.2 Inference

The original paper used a Titan X and cuDNN 4 with Intel Xeon E5-2667v3 @ 3.20GHz for inferece. The results of this project were tested using a GTX 1080 Ti and cuDNN 9.0 with Intel Core i7-6850K @ 3.60GHz.

In theory, both SSD networks have a higher mAP than other object detection models. With about 50 frames per second, SSD 300 delivers a good precision-speed ratio.

	Method	mAP	FPS	BS	# Boxes	Resolution
	Faster R-CNN	73.2%	7	1	∼ 6000	∼ 1000 × 600
	Fast YOLO	52.7%	155	1	98	448 × 448
	YOLO	66.4%	21	1	98	448 × 448
	SSD 300	74.3%	46	1	8732	300 × 300
	SSD 512	76.8%	19	1	24564	512 × 512
	SSD 300	74.3%	59	8	8732	300 × 300
	SSD 512	76.8%	22	8	24564	512 × 512
×	SSD 300	49.09%	49	1	8732	300 × 300
×	SSD 512	54.37%	35	1	24564	512 × 512

Table 5: *Inference results compared to the original paper. The rows marked with × are the results from this project.*

4.3 Sample images

The following sample images were annotated using the SSD 300 network. The images are taken from the Pascal VOC 2007 test set which was also taken for the network evaluation.

Objects that stand alone seem to be detected better than objects that overlap with other objects. The combination of horses and people were detected very good in most of the times.

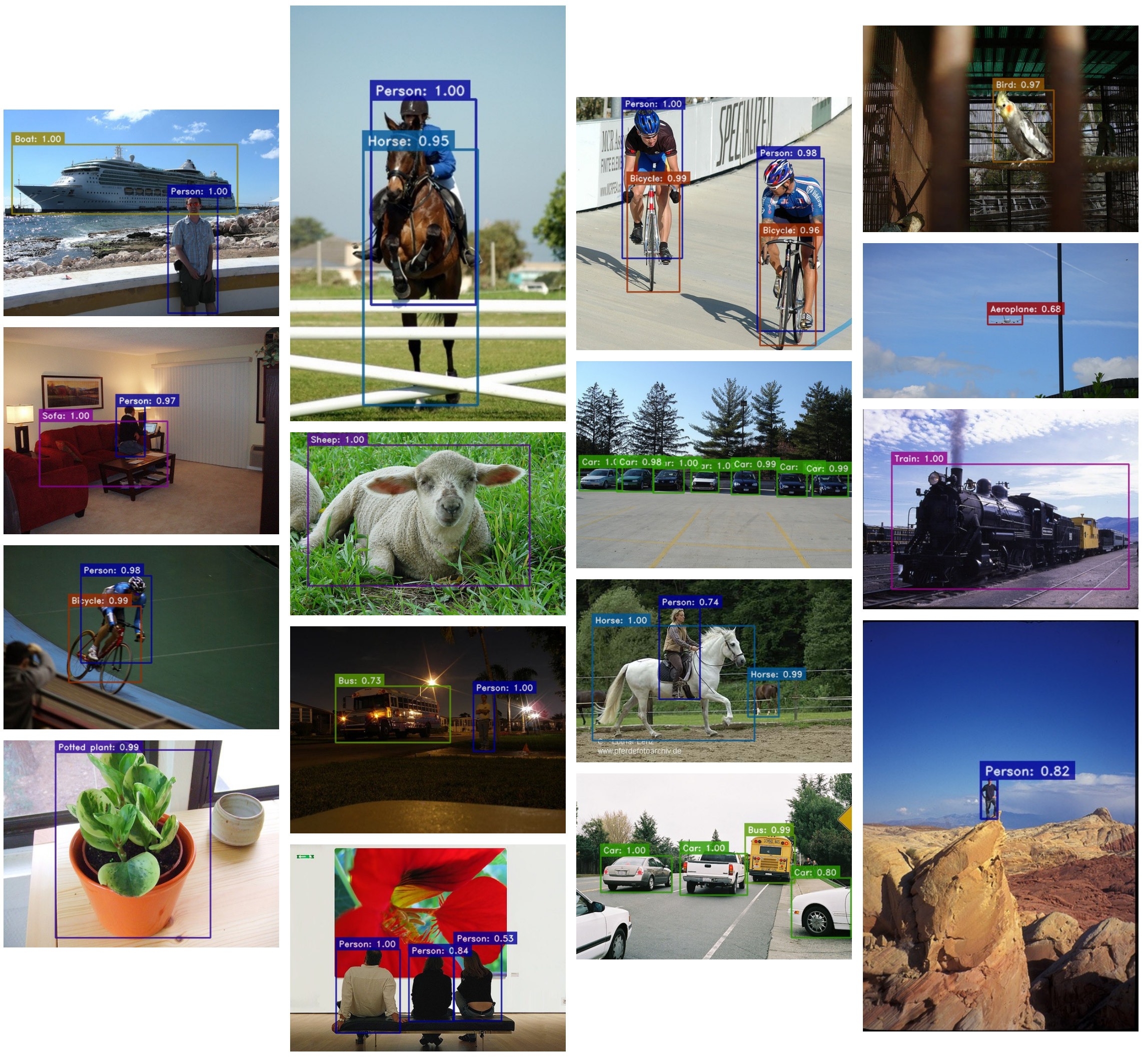


Table 6: *Selected good sample images from Pascal VOC 2007 test set detected by the SSD 300 network.*

Objects that were often false classified are animals like cats, cows and dogs. Bottles were also often classified as people. The network has some difficulties to detect groups of objects like people. The predicted bounding boxes often overlap or describe multiple objects.

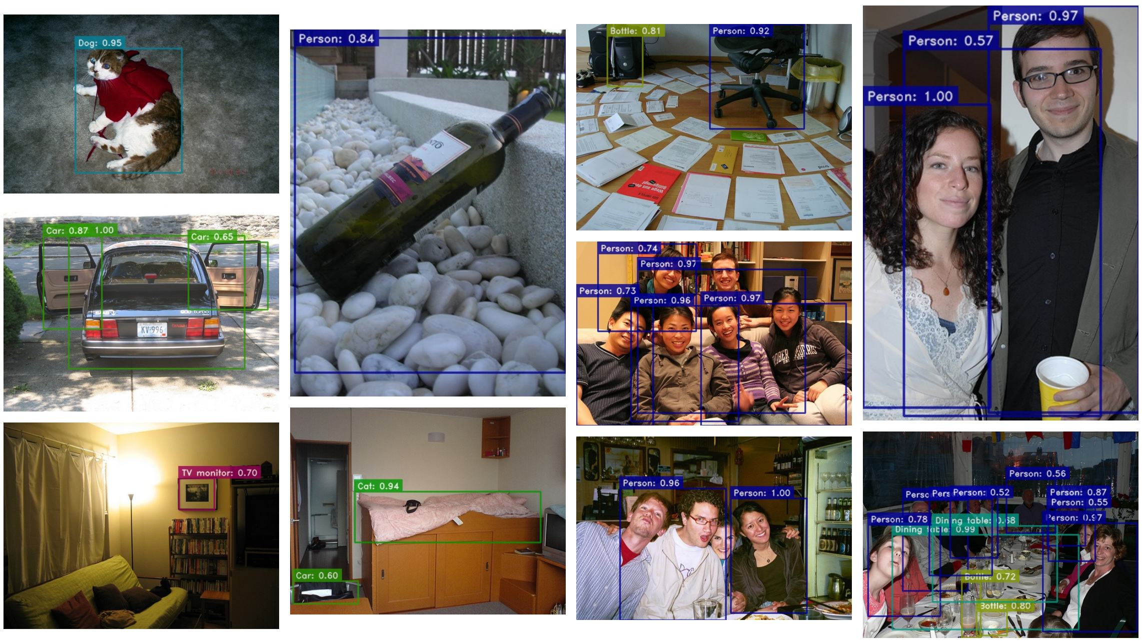


Table 7: *Selected bad sample images from Pascal VOC 2007 test set detected by the SSD 300 network.*

5 Conclusion

We can see that the theory behind the Single Shot MultiBox Detector has proven to be a valid and successful fundament for an object detection network. Next to a good precision the network provides a lightweight model structure that allows fast inference.

During the implementation of this project, we have seen that there are some possibilities to improve the training results of this work. One of the points would be to use a different optimizer and a more elegant way to decay the learning rate. Another way is to improve the data augmentation to expand the dataset.

Overall, the project has given us a great introduction into object detection and the different ways how to tackle the problem.

References

[1] S. Ren et al. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposed Networks”. In: (Jan. 4, 2015). arXiv: `1506.01497 [cs.CV]`.

[2] W. Liu et al. “SSD: Single Shot MultiBox Detector”. Version 5. In: (Dec. 29, 2016). arXiv: `1512.02325 [cs.CV]`.

[3] D. Erhan, A. Szegedy C. Toshev, and D. Anguelov. “Scalable Object Detection using Deep Neural Networks”. In: (Dec. 8, 2013). arXiv: `1312.2249 [cs.CV]`.

[4] W. Liu et al. *SSD-Caffe*. URL: `https://github.com/weiliu89/caffe/tree/ssd`.

[5] J. Redmon et al. “You Only Look Once: Unified, Real-Time Object Detection”. Version 5. In: (May 9, 2016). arXiv: `1506.02640 [cs.CV]`.

[6] N. Silberman and S. Guadarrama. *TensorFlow-Slim image classification model library*. URL: `https://github.com/tensorflow/models/tree/master/research/slim`.

[7] M. Everingham et al. *Pascal Visual Object Classes Challenge 2007*. URL: `http://host.robots.ox.ac.uk/pascal/VOC/voc2007/`.

[8] M. Everingham et al. *Pascal Visual Object Classes Challenge 2012*. URL: `http://host.robots.ox.ac.uk/pascal/VOC/voc2012/`.