university of groningen / faculty of science and engineering

Human detection using different deep neural networks

Bachelor's Project Thesis

Steff Groefsema, s.groefsema@student.rug.nl,
Supervisors: Dr M.A. Wiering

Abstract: In this thesis we compare the performance of two different deep neural networks on the task of human detection. The first network is Faster regional based convolutional neural network (Faster R-CNN), the second Single Shot Multibox Detector. The research question is: Is Faster R-CNN able to achieve a higher mean average precision in comparison to a Single Shot Multibox Detector? The hypothesis is that Faster R-CNN is able to achieve a higher mean average precision in comparison to a Single Shot Multibox Detector. We compare both networks in pairs using ResNet-50 and Inception-V2 as a backbone. Both networks are trained on a subset of the Pascal VOC 2007+2012 dataset and the performance scores are measured using the COCO metrics. Each network has been trained 10 times and the averages of each network are compared with a t-test. After conducting the experiment, the results show that Faster R-CNN was able to achieve a significantly higher mean average precision in comparison with the Single Shot Multibox Detector with ResNet-50 and Inception-V2 as a backbone.

1 Introduction

In the last decade research in machine learning has delivered promising results for real world applications. The increasing availability of lots of data, the increase of computer power and the development of new training strategies all contribute to the increased activity in machine learning. For example in the field of medicine, machine learning has been applied to be able to detect cancer (Kourou, Exarchos, Exarchos, Karamouzis, and Fotiadis, 2015). Another subject of research is autonomous driving vehicles. To enable these vehicles to drive safely on the road, a lot of sensory data about the environment needs to be processed. Object detection is one specific field of research where machine learning and more specifically deep learning has been applied a lot over the last year. One specific task is to be able to detect humans using camera data as input. In this thesis we are going to investigate the performance of different kinds of deep neural networks on the task of human detection.

The first network we are going to test is the Faster regional based convolutional neural network (Faster R-CNN) which was developed by Ren, He, Girshick, and Sun (2015). The second network is the Single Shot Multibox Detector (SSD) which was developed by Liu, Anguelov, Erhan, Szegedy, Reed, Fu, and Berg (2015). A lot of research has already been conducted using these specific networks. One example is the research where the performance of Faster R-CNN is measured using the mean average precision on 20 different categories using the Pascal VOC 2007 dataset (Adam, Zaman, Yassin, Zainol Abidin, and I. Rizman, 2017). In the technical report of Yolo-V3 (Redmon and Farhadi, 2018) there is an overview available of the performance of Yolo-V3 in comparison with Faster R-CNN and SSD. On the specific task of human detection, the research of applying Histograms of Gradients (HoG) (Dalal and Triggs) is very important. Faster R-CNN has also been used as a benchmark for the performance of other networks (Liu, Zhang, Wang, and Metaxas, 2016).

In this thesis the following research question is at hand: Is Faster R-CNN able to achieve a higher mean average precision in comparison to a Single Shot Multibox Detector? The hypothesis is: Faster R-CNN will achieve a higher mean average precision when compared with the Single Shot Multibox

Detector.

In the next section we discuss the methods. In this section each deep neural network will be discussed in detail first. Secondly the train pipeline will be discussed along with the specific configurations tested. Next the sub setting of the PASCAL VOC dataset will be explained and lastly how the models will be evaluated is explained. In section 3 all results will be presented. After the result section the research question will be answered in the discussion section.

2 Methods

2.1 Deep Learning

The term deep learning refers to neural networks which have multiple hidden layers between the input and output layer. Each hidden layer consists of a number of nodes and each connection between nodes of different layers has its own weight. These weights are being initialized at random first and then a forward pass is conducted. In supervised learning there is an error defined based on the target output of the network and the actual output of the network. During training this error is minimized by adjusting the weights slightly using backward propagation. One architecture which is very suitable for explaining the basic concepts of the different layers within the architectures is a convolution neural network (CNN) called VGG-16 (Simonyan and Zisserman, 2014).

First there is an input layer with the size of the image. If the input size is 224 × 224 pixels, the size of the input layer is 224×224×3. Next there is a convolutional layer with size $width \times height \times number\ of\ filters$. A convolutional layer performs a convolution between the input and a filter of size 3×3. The output of this convolution layer depends on the number of filters, the spatial extent of the filters, the stride and the amount of zero padding. In the VGG 16 model the first two layers are convolutional layers with a ReLU activation function applied to the output. After the two convolutional layers there is a pooling step which performs max pooling with a 2×2 filter. Applying max pooling reduces the size of the input matrix because for every 2×2 grid only the highest value is put into the output matrix. This pooling step reduces the size from 224×224×64 to 112×112×128. Next there are two additional convolutional layers with at the fourth convolutional layer another max pooling step which reduces the size further from 112×112×128 to 56×56×256. After another set of three convolutional layers using 256 filters with max pooling, there are two block of three convolutional layers with 512 filters and max pooling at the end of every third convolutional layer. After the convolutional layers there are two fully connected layers with 4096 nodes. The difference between the fully connected layers and the convolutional layer is that a neuron within the convolutional layers is only connected to a few nodes of the previous layer and each node in the fully connected layer is connected with all nodes from the previous layer. This means that there are in the first fully connected layer more than 100,000,000 weights. Finally there is an output layer with 1000 nodes using the softmax activation function. Using this architecture the model could predict up to 1000 classes using a binary vector as output. The total number of parameters which can be trained is 138,000,000 when trained on the Imagenet dataset (Deng, Dong, Socher, Li, Li, and Fei-Fei, 2009).

2.2 Faster R-CNN

Faster R-CNN is an object detection network consisting of two stages. The first is a deep fully convolutional network which proposes regions, the second stage is a Fast R-CNN detector which performs detections on the proposed regions by the first stage.A schematic overview of the entire Faster R-CNN network is shown in Figure 2.1.

2.2.1 First stage

The first stage is a Region Proposal Network (RPN) called fast R-CNN (Girshick, 2015). It takes an input image and outputs a set of rectangular bounding boxes. The authors of Faster R-CNN use two different backbones for feature extraction; VGG-16 and the ZF-model (Zeiler and Fergus, 2013). To generate Region Proposals a small network is slid over the convolutional feature map output of the last shared convolutional layer. This small network takes as input an $n \times n$ spatial window of the convolutional feature map (in the original paper $n = 3$). Next each sliding window is mapped to

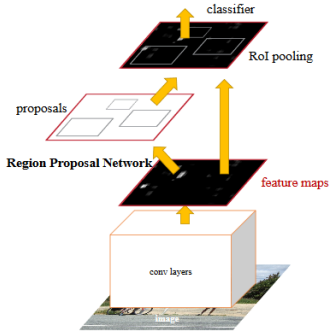


Figure 2.1: A schematic overview of the two stages of the Faster R-CNN network. Taken from Ren et al. (2015)

a lower dimensional feature and is activated using the ReLU activation function. This feature is fed to two fully connected layers; one bounding box regression layer, the other a box classification layer. Because the small network goes over the input in a sliding window fashion, the fully connected networks are shared across all spatial locations. This is implemented as an $n \times n$ convolutional network followed by two 1×1 convolutional layers for the box regression and the box classification. See Figure 2.2 for an example of the small network at one location sliding over the input image.

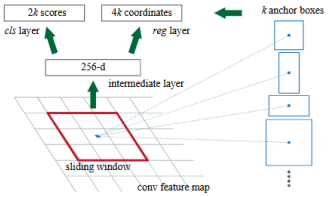


Figure 2.2: A schematic overview of the small network sliding over the input image. Taken from Ren et al. (2015).

2.2.2 Anchors

At each sliding window location multiple region proposal are predicted, the maximal number of possible proposals is denoted as k. The box regression layer outputs $4k$ output coordinates for k boxes, the box classifier layer outputs $2k$ scores that estimate whether or not the box represents an object of a certain class. The k proposals are relative to k reference boxes, commonly referred to as anchors. An anchor is centered in the sliding window and has a scale and an aspect ratio. By default three different scales and aspect ratios are used, yielding 9 anchors for each sliding window position. One important aspect of this method of anchor box generation is that it is translation invariant, making it less likely to overfit the data on small datasets.

2.2.3 Loss function

For training RPNs a binary class label is assigned for each anchor. It is either part of a class or not. In Faster R-CNN labels are assigned to two kinds of anchors; one label to the set of anchors with the highest Intersection-over-Union (IoU) overlap with a ground-truth box or the anchor that has an IoU overlap of 0.7 or higher. A negative label is assigned if its IoU ratio is lower than 0.3 for all ground-truth boxes. Using this definition the loss function is defined (Ren et al., 2015) as:

$$L(p_i, t_i) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*)$$

$$+\lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*)$$

i is here the anchor index in a mini-batch, p_i is the predicted probability of anchor i being an object. The ground-truth label p_i^* is 1 if the anchor is positive, 0 if the anchor is negative. t_i is a vector representation of the parameterised bounding box coordinates, while t_i^* is that of the ground-truth box associated with a positive anchor. The classification loss L_{cls} refers to a log loss over two classes (object vs no object). For the regression loss L_{reg}, the robust loss function L1 is used.

2.3 Single Shot Multibox Detector

The Single Shot Multibox Detector (SSD) is a feed-forward convolutional network that produces a fixed sized collection of bounding boxes and scores for the classes present in these bounding boxes. The base of the SSD network is the standard VGG-16 network with multiple convolutional layers added behind it as shown in (Liu et al., 2015). These extra convolutional layers decrease in size and thus enables the network to perform predictions of different scales and sizes. Besides the multiscale feature maps for detection, there are also convolutional predictors for detection. Each added feature layer can

produce a fixed set of predictions using a set of convolutional filters, as shown in Figure 2.3.

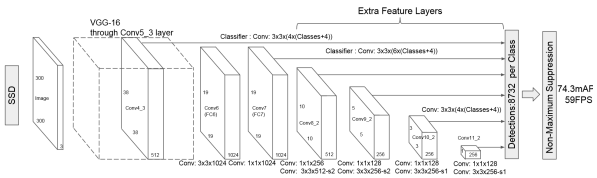


Figure 2.3: A schematic overview of the SSD architecture. Taken from Liu et al. (2015).

A feature layer is of the size $m \times n$ with p channels. A small kernel which predicts parameters of a potential detection has a size of $3 \times 3 \times p$. This kernel can produce two types of output; the first is a score of a certain category, the second is the offset relative to the default box coordinates. At each of the $m \times n$ locations where the kernel is applied it produces an output value. The offset of a bounding box is measured relative to the default bounding box position relative to each feature map position.

2.3.1 Bounding boxes

In SSD a set of default bounding boxes is associated with each feature map cell, for multiple feature maps at the top of the network. The position of the default bounding boxes relative to its feature map cell is fixed in a convolutional manner. Next the offsets of all boxes relative to the default bounding box shapes is computed along with the class scores of possible detections within each bounding box for each feature map cell. For each box there are k relative locations and each class scores c plus an offset of 4, the total amount of filters is set to be $(c+4)k$. These filters yield together for each location of the feature map $(c+4) \times k \times n \times m$ outputs. The bounding boxes of SSD are similar to the anchor boxes of Faster R-CNN, the key difference is however that SSD applies this process to multiple feature maps of different resolutions. The authors claim that this results in a more efficient discretization of the space of possible output box shapes.

2.3.2 Loss function

The training objective of SSD is a weighted sum over the localization loss and the confidence loss defined by (Liu et al., 2015) as:

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

where N is the number of default boxes. When $N = 0$, the loss function is set to be 0. α is a parameter to set the weight of the localization loss, l is the predicted box and g stands for the ground-truth box. The localization loss will be explained in more detail first.

Similar to Faster R-CNN the localization loss is a smooth L1 Loss between the predicted box(l) and the ground-truth box(g):

$$L_{loc}(x, l, g) = \sum_{i \in Pos}^{N} \sum_{m \in \{cx, xy, w, h\}} smooth_{L1}(l_i^m - \hat{g}_m^j)$$

Where (cx, cy) is the center of default bounding box d, w and h denote its width and height. The smooth L1 function is defined as:

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

The confidence loss is a softmax loss over multiple class confidences, denoted as c. Let $x_{ij}^p = \{1, 0\}$ be an indicator for matching the i-th bounding box to the j-th ground truth box of class p. The confidence loss is defined as:

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

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

The first summation in the function states that for every positive prediction matching with class c the loss will be reduced with the confidence score of the current class c in the formula denoted as $\hat{c}_i^p$. This ensures the overall confidence loss will be lowered with every correct positive prediction. The second summation states that in the case of a negative match, the loss will be reduced with the confidence score of the background class 0, which is denoted as $\hat{c}_i^0$.

2.4 Backbones used in this Experiment

2.4.1 ResNet 50

ResNet 50 is a deep neural network which applies residual blocks (He, Zhang, Ren, and Sun, 2015). The main goal of residual learning is to explicitly fit a stack of layers to a residual mapping, instead of hoping a few sets of layers might fit this mapping. With $H(x)$ defined as the underlying mapping for input x, we define the residual function which the set of layers needs to approximate as:

$$F(x) = H(x) + x$$

Sets of layers which are applying residual learning are refered to as building blocks. Formally a building block can be described as:

$$y = (F(x, \{W_i\}) + x$$

Here x represent the input, y the output and $(F(x, \{W_i\})$ represents the residual mapping to be learned. In Figure 2.4 there is a two layered example of a building block with as function for F:

$$F(x) = W_2\sigma(W_1x)$$

with σ being the ReLU function. $F(x) + x$ is performed as an element-wise addition after the set of layers.

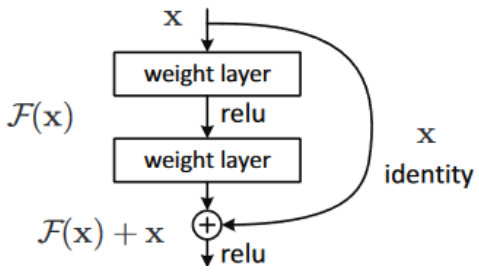


Figure 2.4: A schematic overview of a building block. Taken from He et al. (2015).

ResNet 50 uses this principle of residuals learned using the following set of layers: First there is a convolutional layer with a size of 7 × 7 using 64 filters and a stride of 2. Next there is a 3 × 3 max pooling layer with a stride of 2. Next there is a set of 3 building block with convolutional layers of size 1 × 1 using 64 filters, 3 × 3 using 64 filters, 1 × 1 using 256 filters. Next there are four blocks with the sizes 1 × 1 using 128 filters, 3×3 using 128 filters and 1 × 1 using 512 filters. After that 6 blocks with the sizes 1 × 1 using 256 filters, 3 × 3 using 256 filters and 1 × 1 using 1024 filters. The last set of three blocks have convolutional layers with the sizes 1 × 1 using 512 filters, 3 × 3 using 256 and 1 × 1 using 2048 filters. After the last set of building blocks average pooling is applied. The last layer is a fully-connected layer with 1000 nodes using softmax when trained on ImageNet.

2.4.2 Inception-V2

Inception-V2 was built by Szegedy, Vanhoucke, Ioffe, Shlens, and Wojna (2015) to reduce the loss of information due to too much reduction of the input dimension, better known as the representational bottleneck. For example instead of using a patch of size 5 × 5 with n filters at the convolution an inception module uses two 3 × 3 modules with the same number of filters. This is done because a 3 ×3 convolution is 2.78 times less computational intensive as a 5 × 5 convolution (Szegedy et al., 2015). In the Inception-V2 network the first two 7 × 7 convolutional layers are factorised to a set of three 3 × 3 convolutional layers. Next there is 3x Inception layer which factorizes each 5 × 5 convolution to a set of 3 × 3 convolutions as shown in Figure 2.5.

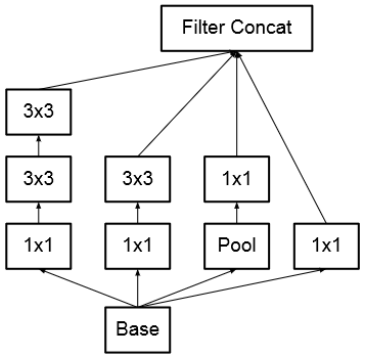


Figure 2.5: A schematic overview of the 3x inception layer where a 5 × 5 convolution is factorised into smaller 3 × 3 modules. Taken from Szegedy et al. (2015).

A similar structure is used in the next 5x Inception layer where $n = 7$ for a 17 × 17 filter grid as

shown in Figure 2.6.

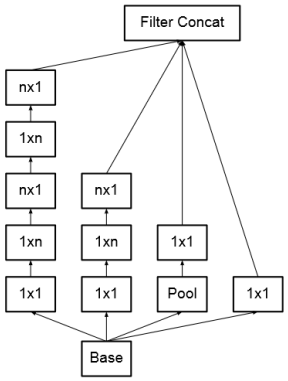


Figure 2.6: A schematic overview of the 5x Inception layer where $n = 7$ for a 17×17 convolution. Taken from Szegedy et al. (2015).

The structure of the last Inception 2 layer is shown in Figure 2.7.

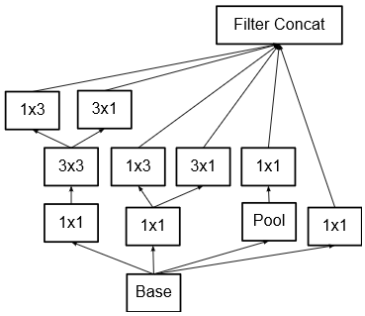


Figure 2.7: A schematic overview of the Inception 2 layer which expands on the output of the filter banks. Taken from Szegedy et al. (2015).

After the inception layers there is an 8×8 max pooling layer with a ReLU layer behind it. Lastly the output layer is again a fully-connected layer with 1000 nodes which uses the softmax function.

2.5 Experimental Setup

For this experiment, we use the TensorFlow Object detection API as a base for training and testing different deep learning networks on the task of human detection. Four different networks were selected from the TensorFlow Model Zoo to start the training with:

- Faster R-CNN with ResNet-50 as backbone
- SSD with ResNet-50 as backbone
- Faster R-CNN with Inception-v2 as backbone
- SSD with Inception-v2 as backbone

2.5.1 The PASCAL VOC dataset

For this experiment we use a subset of the PASCAL VOC dataset which is a dataset containing 20 different classes for an object detection challenge organized by Everingham, Gool, Williams, Winn, and Zisserman (2009). We extracted all images of the class person from the PASCAL VOC 2007 and 2012 dataset in such a way that all models can be trained within TensorFlow. This results in the data in Table 2.1.

Name	**Number of images**	**Number of anchor boxes**
Pascal VOC 07+12 training set	13775	28075
Pascal VOC 2007 test for validation set	2097	5227
Pascal VOC 2012 test set	5138	7326

Table 2.1: The number of images and anchor boxes in each data set used for training and testing all networks.

2.5.2 Training and testing the networks

All networks will be trained and tested using the config files provided with the models at the TensorFlow model zoo as a baseline, but the number of iterations of SSD with ResNet 50 was doubled to achieve a better performance. For each backbone both Faster R-CNN and SSD will be trained on the Pascal VOC 07+12 subset and tested on the Pascal VOC 2012 test set as described in Table 2.1. Each network will be trained and tested ten times. The performance of each network will be summarized using the COCO evaluation metrics from the TensorFlow Object Detection API. Because we compare the network performances on just one class, we prefer the COCO evaluation metric over the standard Pascal VOC metric. The COCO metrics provide more detailed information about mean Average Precision (mAP) for different sizes of objects, along with information about recall, rather than only an mAP score. The average of all 10 runs will be compared with a t-test in order to test which

network performs (significantly) better in terms of mAP and recall.

3 Results

In this section the average scores over 10 runs will be presented for both networks using Resnet-50 and Inception V-2. After the scores there will be an overview of the t-test statistics. Some output images of the networks are also provided in this section to visualize some performance differences.

ResNet-50	**average precision @IoU= 0.50:0.95**	**average precision @IoU=0.50**	**average precision @IoU=0.75**
Faster R-CNN	**0.833 ± 0.003**	**0.593 ± 0.007**	**0.681 ± 0.008**
SSD	0.796 ± 0.011	0.537 ± 0.021	0.600 ± 0.028

Table 3.1: The average precision and standard deviations calculated for Faster R-CNN vs SSD using ResNet-50 on the VOC 2012 test set. In the first column the average precision measured from 0.5 till 0.95 at their correspondent IoU threshold is given, in the second the average precision measured at an IoU threshold of 0.5 and in the last column the average precision is measured at an IoU threshold of 0.75.

Resnet-50	**average precision @IoU= 0.50:0.95 small**	**average precision @IoU= 0.50:0.95 medium**	**average precision @IoU= 0.50:0.95 large**
Faster R-CNN	**0.061 ± 0.006**	**0.404 ± 0.014**	**0.647 ± 0.007**
SSD	0.038 ± 0.011	0.310 ± 0.012	0.600 ± 0.022

Table 3.2: The average precision and standard deviations calculated for Faster R-CNN vs SSD using ResNet-50 on the VOC 2012 test set. In the first column the average precision measured from 0.5 till 0.95 at their correspondent IoU threshold is given for the small objects, in the second for medium objects and in the last column the average recall is given for large objects.

ResNet-50	**average recall @IoU= 0.50:0.95**	**average recall @IoU=0.50**	**average recall @IoU=0.75**
Faster R-CNN	**0.484 ± 0.004**	**0.745 ± 0.008**	**0.761 ± 0.008**
SSD	0.444 ± 0.013	0.694 ± 0.017	0.732 ± 0.014

Table 3.3: The average recall and standard deviations calculated for Faster R-CNN vs SSD using ResNet-50 on the VOC 2012 test set. In the first column the average recall measured from 0.5 till 0.95 at their correspondent IoU threshold is given, in the second the average recall measured at an IoU threshold of 0.5 and in the last column the average recall is measured at an IoU threshold of 0.75.

ResNet-50	**average recall @IoU= 0.50:0.95 small**	**average recall @IoU= 0.50:0.95 medium**	**average recall @IoU= 0.50:0.95 large**
Faster R-CNN	0.461 ± 0.016	**0.670 ± 0.009**	**0.792 ± 0.008**
SSD	0.476 ± 0.160	0.636 ± 0.014	0.763 ± 0.014

Table 3.4: The average recall and standard deviations calculated for Faster R-CNN vs SSD using ResNet-50 on the VOC 2012 test set. In the first column the average recall measured from 0.5 till 0.95 at their correspondent IoU threshold is given for the small objects, in the second for medium objects and in the last column the average recall is given for large objects.

As you can see in Table 3.1 Faster R-CNN has higher scores on all 3 metrics shown in comparison to SSD. The difference in mAP measured at IoU thresholds ranging from 0.50 till 0.95 is significant ($t(13.55) = -8.94$, $p < 0.05$). The difference in mAP measured at the IoU threshold of 0.75 is significant ($t(11.18) = -7.82$, $p < 0.05$). The difference in mAP measured at the IoU threshold of 0.50 is significant ($t(10.74) = -8.58$, $p < 0.05$). According to the other t-tests the difference between the mAP scores of Faster R-CNN and SSD measured for the classes small, medium and large defections shown in Table 3.2 are significant ($p < 0.05$). Comparing the differences between the recall scores of both Faster R-CNN and SSD in Table 3.3 with a t-test show the differences are significant ($p < 0.05$). The only other metric where the difference is not significant ($t(15.56) = 1.65$, $p > 0.05$)), is the average recall of the group small. There is SSD able to achieve a higher score than Faster R-CNN. Faster R-CNN scores higher on 11 metrics in comparison with SSD. There is a small difference visible for the

recall score of the class small. However this difference is not visible for the medium and large detection classes.

Inception-V2	**average precision @IoU= 0.50:0.95**	**average precision @IoU=0.50**	**average precision @IoU=0.75**
Faster R-CNN	**0.843 ± 0.003**	**0.599 ± 0.007**	**0.692 ± 0.008**
SSD	0.792 ± 0.007	0.525 ± 0.008	0.577 ± 0.012

Table 3.5: The average precision and standard deviations calculated for Faster R-CNN vs SSD using Inception-V2 on the VOC 2012 test set. In the first column the average precision measured from 0.5 till 0.95 at their correspondent IoU threshold is given, in the second the average precision measured at an IoU threshold of 0.5 and in the last column the average precision is measured at an IoU threshold of 0.75.

Inception-V2	**average precision @IoU= 0.50:0.95 small**	**average precision @IoU= 0.50:0.95 medium**	**average precision @IoU= 0.50:0.95 large**
Faster R-CNN	**0.063 ± 0.06**	**0.405 ± 0.009**	**0.654 ± 0.007**
SSD	0.020 ± 0.005	0.250 ± 0.011	0.604 ± 0.005

Table 3.6: The average precision and standard deviations calculated for Faster R-CNN vs SSD using Inception-V2 on the VOC 2012 test set. In the first column the average precision measured from 0.5 till 0.95 at their correspondent IoU threshold is given for the small objects, in the second for medium objects and in the last column the average recall is given for large objects.

Inception-V2	**average recall @IoU= 0.50:0.95**	**average recall @IoU=0.50**	**average recall @IoU=0.75**
Faster R-CNN	**0.485 ± 0.004**	**0.746 ± 0.008**	**0.762 ± 0.007**
SSD	0.445 ± 0.005	0.669 ± 0.007	0.697 ± 0.005

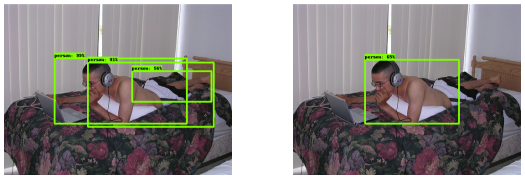
Table 3.7: The average recall and standard deviations calculated for Faster R-CNN vs SSD using Inception-V2 on the VOC 2012 test set. In the first column the average recall measured from 0.5 till 0.95 at their correspondent IoU threshold is given, in the second the average recall measured at an IoU threshold of 0.5 and in the last column the average recall is measured at an IoU threshold of 0.75.

Inception-V2	**average recall @IoU= 0.50:0.95 small**	**average recall @IoU= 0.50:0.95 medium**	**average recall @IoU= 0.50:0.95 large**
Faster R-CNN	**0.473 ± 0.016**	**0.668 ± 0.009**	**0.793 ± 0.008**
SSD	0.207 ± 0.023	0.527 ± 0.007	0.754 ± 0.006

Table 3.8: The average recall and standard deviations calculated for Faster R-CNN vs SSD using Inception-V2 on the VOC 2012 test set. In the first column the average recall measured from 0.5 till 0.95 at their correspondent IoU threshold is given for the small objects, in the second for medium objects and in the last column the average recall is given for large objects.

As you can see in Table 3.5 till Table 3.8, Faster R-CNN achieves a higher score on all 12 metrics in comparison with SSD when using Inception-V2 as a backbone. When comparing the mAP scores measured at the IoU threshold ranging from 0.5 till 0.95, the difference is significant ($t(12.71) = -20.58$, $p < 0.05$). Also the difference in mAP scores measured at the IoU threshold of 0.75 and 0.5 is significant ($t(17.90) = -22.28$, $p < 0.05$ and $t(15.73) = -26.14$, $p < 0.05$). According to the other t-tests the difference between the mAP scores of Faster R-CNN and SSD measured for the classes small, medium and large defections are significant ($p < 0.05$). Comparing the differences between the recall scores of both Faster R-CNN and SSD with a t-test shows the differences are very significant ($p < 0.05$). When comparing the scores of both networks with ResNet-50 and Inception-V2, there is no backbone which scores always higher than the other. Faster R-CNN scores higher with Inception-V2 on mAP measured at IoU from 0.50 til 0.95 in comparison with ResNet-50. SSD on the other hand, scores on the same metric higher with ResNet-50 as a backbone.

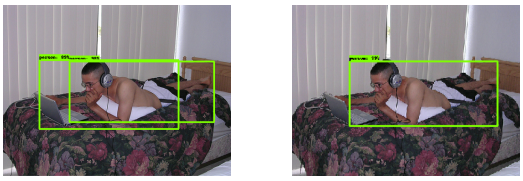
3.1 Images



(a) Faster R-CNN **(b) SSD**

Figure 3.1: An example output of both networks using ResNet-50.

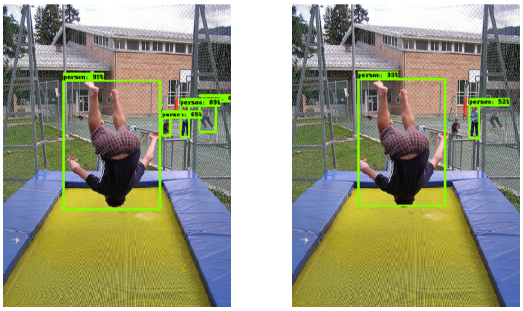
As you can see in Figure 3.1 both networks are able to detect the person in this figure. Faster R-CNN produces a lot more outputs with lower confidence intervals in comparison with SSD, which only produces one bounding box with a high confidence of 99%.



(a) Faster R-CNN **(b) SSD**

Figure 3.2: An example output of both networks using Inception-V2.

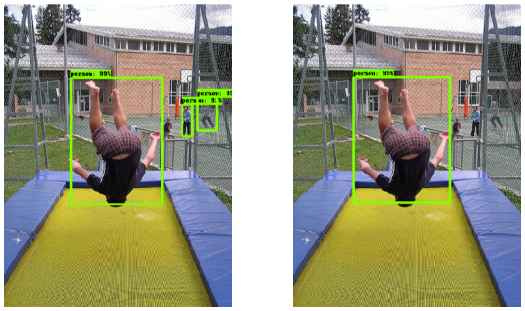
As you can see in Figure 3.2 both networks are able to detect the person in this figure using Inception-V2 as a backbone. Faster R-CNN produces two outputs for the same person, SSD produces one bounding box and it fits the person.



(a) Faster R-CNN **(b) SSD**

Figure 3.3: Another example output of both networks using ResNet-50.

In Figure 3.3 an interesting difference between Faster R-CNN and SSD using ResNet-50 as a backbone can be observed. Faster R-CNN is able to detect all four persons in the picture, while SSD fails to detect some of the people in the background.



(a) Faster R-CNN **(b) SSD**

Figure 3.4: Another example output of both networks using Inception-V2.

When both Faster R-CNN and SSD use Inception-V2 as a backbone, you can see in Figure 3.4 that both Faster R-CNN and SSD detect the person in front, while two persons in the background are just detected by Faster R-CNN and completely ignored by SSD.

4 Conclusion

In this thesis we looked at the performance of deep neural networks on the task of human detection. In this experiment we used Faster R-CNN and SSD both with ResNet-50 and Inception-V2 as a backbone. The performance scores of both networks using Resnet-50 and Inception-V2 can be found in the previous section. We started this experiment with the following research question: Is Faster R-CNN able to achieve a higher mean average precision in comparison to a Single Shot Multibox Detector? The hypothesis is that Faster R-CNN is able to achieve a higher mean average precision in comparison to a Single Shot Multibox Detector. Using the results presented in section 3 we can conclude the following: In both experiments, with Resnet-50 and Inception V2, Faster R-CNN is able to achieve a higher mean average precision in comparison to a Single Shot Multibox Detector on the Pascal VOC 2012 data set. According to the t-test data the difference in mean average precision between Faster R-CNN and SSD is significant. Based on this result, we can't reject our hypothesis that Faster R-CNN is able to achieve a higher mean average precision in comparison to a Single Shot Multibox Detector.

4.1 Discussion

We found in our experiments that Faster R-CNN is able to achieve a higher mean average precision score in comparison with the Single Shot Multibox Dectector. We think the main reason for Faster R-CNN scoring higher is the fact that Faster R-CNN generates 9 different anchor boxes which are translation invariant. Although SSD applies the same sort of anchor boxes to different sizes of feature maps, the performances scores are not at the same level as Faster R-CNN.

It is very hard to make a good comparison with previous research scores. The reason for this is that authors of machine learning algorithms publish most of the time only the performance scores on common data set such as Pascal VOC and COCO, for 20 or 80 classes, instead of just one for the person class. We know from the original paper of SSD (Liu et al., 2015), that the Single Shot Multibox Detector is able to achieve a higher mAP score in comparison to Faster R-CNN when using all 20 classes from the Pascal VOC Challenge. In the research (Redmon and Farhadi, 2018), a comparison is made between different versions of Faster R-CNN, SSD and Yolo-V3 with the mean average precision at the Intersection of Union at 0.50. This is not a very good comparison, because not only the networks differ, but also the backbones they use differ from this experiment.

One other thing that makes these comparisons difficult is the fact that all networks are programmed using a different network architecture. Yolo V3 is built on DarkNet by Redmon and Farhadi and can't be compared with Faster R-CNN and SSD using a different backbone. That's why in this experiment only networks within the TensorFlow object detection API are compared. This has been done to ensure no other side effects will influence the performance score.

For further research one possible suggestion could be that all deep learning networks use the same network architecture as a base. If the field uses one general format, it will become more transparent which network is 'best'. It will also allow other researchers to conduct experiments with new, larger data sets more easily.

References

B. Adam, Fadhlan Zaman, Ihsan Yassin, Husna Zainol Abidin, and Z. I. Rizman. Performance evaluation of faster R-CNN on GPU for object detection. *Journal of Fundamental and Applied Sciences*, 2017:909–923, 09 2017. doi: 10.4314/jfas.v9i3s.64.

N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR05)*. IEEE. doi: 10.1109/cvpr.2005.177. URL `https://doi.org/10.1109/cvpr.2005.177`.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (VOC) challenge. *International Journal of Computer Vision*, 88(2):303–338, sep 2009. doi: 10.1007/s11263-009-0275-4. URL `https://doi.org/10.1007/s11263-009-0275-4`.

Ross B. Girshick. Fast R-CNN. *CoRR*, abs/1504.08083, 2015. URL `http://arxiv.org/abs/1504.08083`.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL `http://arxiv.org/abs/1512.03385`.

Konstantina Kourou, Themis P. Exarchos, Konstantinos P. Exarchos, Michalis V. Karamouzis, and Dimitrios I. Fotiadis. Machine learning applications in cancer prognosis and prediction. *Computational and Structural Biotechnology Journal*, 13:8–17, 2015. doi: 10.1016/j.csbj.2014.11.005. URL `https://doi.org/10.1016/j.csbj.2014.11.005`.

Jingjing Liu, Shaoting Zhang, Shu Wang, and Dimitris N. Metaxas. Multispectral deep neural networks for pedestrian detection. *CoRR*, abs/1611.02644, 2016. URL `http://arxiv.org/abs/1611.02644`.

Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: single shot multibox detector. *CoRR*, abs/1512.02325, 2015. URL `http://arxiv.org/abs/1512.02325`.

Joseph Redmon and Ali Farhadi. Yolo v3: An incremental improvement. *CoRR*, abs/1804.02767, 2018. URL `http://arxiv.org/abs/1804.02767`.

Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015. URL `http://arxiv.org/abs/1506.01497`.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. URL `http://arxiv.org/abs/1409.1556`.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567, 2015. URL `http://arxiv.org/abs/1512.00567`.

TensorFlow. Tensorflow object detection. URL `https://github.com/tensorflow/models`.

Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. *CoRR*, abs/1311.2901, 2013. URL `http://arxiv.org/abs/1311.2901`.

TensorFlow Model Zoo. Model zoo. URL `https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md`.