BRNO UNIVERSITY OF TECHNOLOGY

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF CONTROL AND INSTRUMENTATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

BASICS OF PEDESTRIANS DETECTION IN IMAGE BY MACHINE LEARNING

ZÁKLADY DETEKCE OSOB V OBRAZU POMOCÍ METOD STROJOVÉHO UČENÍ

BACHELOR'S THESIS
BAKALÁŘSKÁ PRÁCE

AUTHOR — **Peter Lučanský**
AUTOR PRÁCE

SUPERVISOR — **Ing. Karel Horák, Ph.D.**
VEDOUCÍ PRÁCE

BRNO 2019

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní obor **Automatizační a měřicí technika**
Ústav automatizace a měřicí techniky

Student: Peter Lučanský ***ID:*** 197720
Ročník: 3 ***Akademický rok:*** 2018/19

NÁZEV TÉMATU:

Základy detekce osob v obrazu pomocí metod strojového učení

POKYNY PRO VYPRACOVÁNÍ:

Lokalizace i klasifikace chodců na snímku dopravní scény je obvykle prováděna modelováním konvolučními neuronovými sítěmi. Tento přístup je i cílem této práce. Provedte:
1. rešerši literatury detekce chodců z pohledu autonomního vozidla
2. seznam dostupných datasetů (trénovací a testovací část)
3. implementace vybraného nebo návrh vlastního algoritmu ML
4. ověření kodu na datasetu typu CityPersons

DOPORUČENÁ LITERATURA:

1. Goodfellow I., Bengio Y., Courville A.: Deep Learning. MIT Press, 2016. ISBN 9780262035613. (deeplearningbook.org)

2. Buduma, N., Locascio, N. Fundamentals of Deep Learning. O'Reilly Media, Inc. 2017. ISBN 9781491925614.

Termín zadání: 4.2.2019 ***Termín odevzdání:*** 20.5.2019

Vedoucí práce: Ing. Karel Horák, Ph.D.
Konzultant:

doc. Ing. Václav Jirsík, CSc.
předseda oborové rady

ABSTRACT

This thesis deals with cutting-edge computer vision task the detection of persons/pedestrians in images by using machine learning methods with its possible utilization, history of progress and explanations of functionalities. It also includes testing the today's best method available on various circumstances and comparing aspects that has impact on its performance.

At the beginning the matter is fundamentally explained and then are in details described up to date achievements in the subject of matter. In the following part are described available datasets that may be used for training with pointed out their pros and cons. In the last section is in details explained how to use the chosen method. Lastly is executed its training on various situations and comparison of the results is made.

KEYWORDS

convolutional neural network, object detection, dataset, average loss, mean average precision, intersection over union, batch size, iterations, feature extractor

ABSTRAKT

Táto Bakalárska práce sa zaoberá významnou problematikou v oblasti počítačového videnia, ktorou je detekcia osôb/chodcov v obraze, za pomoci metod strojového učenia, spolu s jej možným využitím, vývojom a vysvetlením princípov. Taktiež sa zaoberá testovaním dnes najlepšieho dostupného algoritmu, pričom sa porovnávajú faktory ktoré vplívajú na kvalitu jeho činnosti.

Na začiatku je problematika stručne popísaná, potom sa prejde k podrobným popisom dosiahnutých pokrokov. V nasledujúcej časti sú popísané dostupné datasety, ktoré by sa dali použiť pri tréningu detekčného algoritmu. V poslednom rade sú vykonané trénovacie procesy za rozličných podmienok, pričom sú jednotlivé výsledky porovnávané.

KLÍČOVÁ SLOVA

konvolučná neuronová sieť, detekcia objektov, testovaci dataset, trenovaci dataset, average loss, mean average precision, extraktor príznakov, iterácie

LUČANSKÝ, Peter. *Basics of pedestrians detection in Image by machine learning*. Brno, Rok, 57 p. Bachelor's Thesis. Brno University of Technology, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizační a měříci techniky. Advised by Ing. Karel Horák, Ph.D.

DECLARATION

I declare that I have written the Bachelor's Thesis titled "Basics of pedestrians detection in Image by machine learning" independently, under the guidance of the advisor and using exclusively the technical references and other sources of information cited in the thesis and listed in the comprehensive bibliography at the end of the thesis.

As the author I furthermore declare that, with respect to the creation of this Bachelor's Thesis, I have not infringed any copyright or violated anyone's personal and/or ownership rights. In this context, I am fully aware of the consequences of breaking Regulation § 11 of the Copyright Act No. 121/2000 Coll. of the Czech Republic, as amended, and of any breach of rights related to intellectual property or introduced within amendments to relevant Acts such as the Intellectual Property Act or the Criminal Code, Act No. 40/2009 Coll., Section 2, Head VI, Part 4.

Brno ....

author's signature

Contents

List of Figures

List of Tables

Introduction

Person or pedestrian detection in images is an essential and significant computer vision task that can be used in various modern applications. For example in any intelligent video surveillance system, as it provides the fundamental information for semantic understanding of the video footages. It has also an obvious extension to automotive applications due to the potential for improving safety systems and autonomous driving systems. In recent years it presents subject of intense development and significant progress.

The theoretical part mentiones and explaines some of the today most promising detection methods such as R-CNN, its modifications, SPP-net, YOLO and many others. Also with some of the older ones.

In the chapter of datasets there are mentioned datasets that may be potentially useful for training and testing chosen detection system. Five of them are described in more details with their pros and cons.

In the practical part is discussed chosen implementation of detection system, used hardware and composition of sets that will be used in training process. Then all the configurations including configuring and running project are described in details in manual form. In the end of practical part is choosen system trained on sets that were composed in various kinds of ways. While the development of parameters such as loss and mAP are plotted in graphs. Obtained results are illustrated in attached graphs and illustrative images. Then are the performances of those trained instances criticaly discussed. In the conclusion is present the whole summary of results with achievements of this thesis.

1 Theoretical part

Methods that solve padestrians detection problem without CNN are for instance, Histograms of oriented gradients for human detection[1] Fig.1.1, Crosstalk cascades for frame-rate pedestrian detection[2]. The most advanced algorithms of them are probably modifications of the Deformable Part Model for Object Detection (DPM)[4][5], which uses histograms of oriented gradients. These methods are today almost completely replaced with methods using Convolutional neural networks(CNN), that are more accurate. The only advantage of DPM may be the real time speed but that was also outperformed by the latest CNN-based detectors. Therefore in the next sections will be mentioned only methods that uses CNN.

As we know CNN are designed for image classification, but for detecting people in scene we don't need just classification, because in pedestrians images may have various sizes and localizations within image, therefore we need to solve the localization problem also. In the following sections will be described methods that was successfully used in the past for classification well as localization. They are ordered from the oldest and the most inferior of them to the newest and best performing.

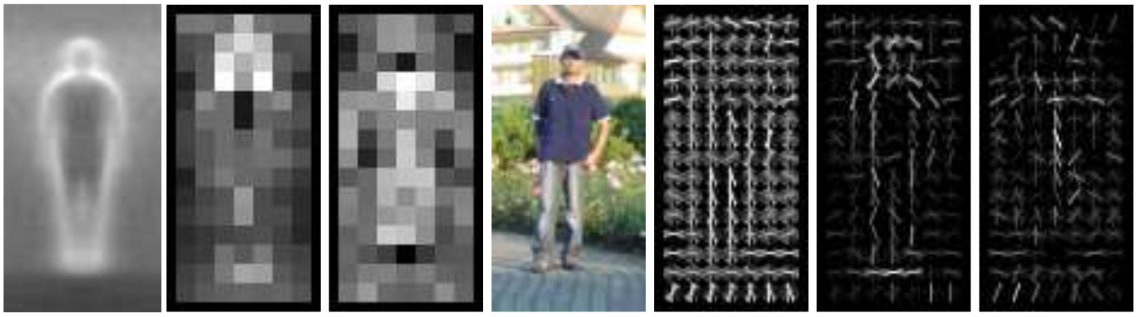


Fig. 1.1: Histogram of Oriented Gradient(HOG) detectors cue mainly on silhouette contours (especially the head, shoulders and feet). The most active blocks are centred on the image background just outside the contour. (a)The average gradient image over the training examples. (b)Each "pixel" shows the maximum positive SVM weight in the block centred on the pixel. (c)Likewise for the negative SVM weights. (d)A test image. (e)It's computed R-HOG descriptor. (f,g)The R-HOG descriptor weighted by respectively the positive and the negative SVM weights.

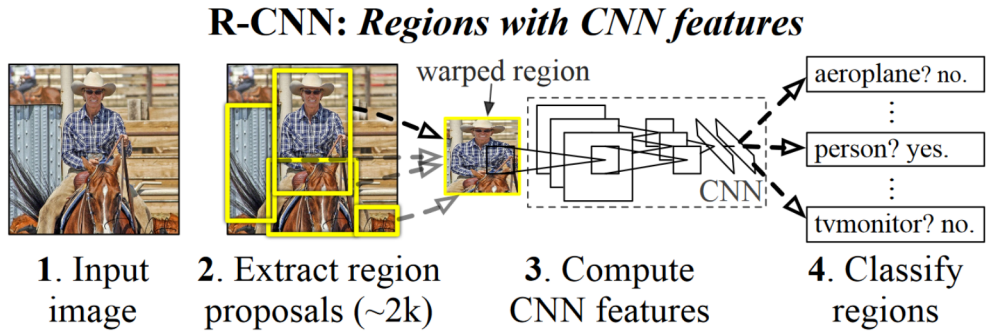


Fig. 1.2: R-CNN System overview

1.1 Sliding-Window Detector

This method uses CNN to classify objects in section of the picture (window) that is sliding through the picture with fixed step (sliding-window approach). There are used various-sized windows for detecting big objects as well as small ones. It is typically useful for constrained objests such as faces and padestrians[3]. This approach was used for localization using CNN at least two decades. Downside is that wasteful number of image proposals have to be processed.

1.2 R-CNN

[7] R-CNN was first time introduced by Ross Girshick, Jeff Donahue, Trevor Darrell and Jitendra Malik in October 2014. This object detection system consist of three modules. The first one generates category-independent region proposals (about 2000). The second is the CNN that extracts features and third the binary SVN(support vector machine). The first module consists of selective search algotithm, which generates category-independent region proposals, by running segmentation algorithm that generates segmentation map, and then the bounding boxes are placed arround every blob that was generated (about 2000)Fig.1.3.

After region was proposed image must be converted into the form that is compatible with CNN (originally fixed 227x227 pixel size). So regardless of the size or aspect ratio of the box candidate region, all pixels are warped into bounding box of required size as you can see in fig.1.4. For feature extraction was originally used CNN described by Krizhevsky[8].with five convolutional layers that extracts 4096-dimensional feature vector from each region proposal. Results on PASCAL VOC 2011-12 was 53.3%, which overcame every method in that time.

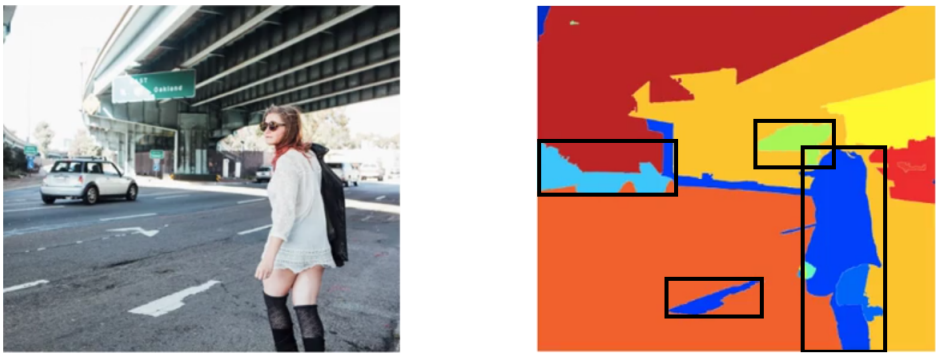


Fig. 1.3: Selecive search explanation, only 4 possible boxes are shown

Fig. 1.4: Warped training samples from VOC 2007 train.

However R-CNN has notable drawbacks:

1.Training is multi-stage pipeline firstly is tuned ConvNet on object proposals using log loss. Then SVM is fit to ConvNet features. These SVMs act as object detectors, replacing the softmax classifier. And in the third stage bounding-box regressors are learned.

2. Training is expensive in space and time. During training features are extracted from each object proposal in each image and written to disc, these features hundreds of gigabytes of storage.

3. Object detection is slow Using CNN VGG16 detection takes about 47s/image using GPU, because features are extracted from each object proposal.

1.3 SPP-net-based system

[6] This method solves problem where CNNs require inputs of fixed sizes, which limits the aspect ratio and the scale of the input image. And was introduced by Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun in April 2015. Without SPP-net we have to edit pictures that do not fit either via cropping or warping as in fig.1.5. But cropping may eliminate usefull parts of the object and warping may

results in unwanted geometric distortions. So authors asked themselves, why do CNNs require a fixed input size. As we know CNNs consists of convolutional layers and fully-connected layers. The convolutional layers outputs the feature maps via convolution, which represent the spatial arrangement of the activations fig.1.6, but convolutional layers don't need input of fixed size, in fact they can generate feature maps of any size from inputs of any sizes, but on the other hand, the fully connected layers strictly require fixed-size inputs. Therefore the fixed-size constraint of CNNs comes from fully connected layers that comes after the convolutional layers. What is special about SPP-net is that there was added the SPP(Spatial Pyramid Pooling) layer right after the last convolutional layer to eliminate the fixed-size constraint of the network. So we don't have to crop or warp image in the beginning see in fig.1.5.

The idea of Spatial Pyramid Pooling is derived from traditional Bag of Visual Words algorithm[9]. In detection system the spatial pyramid is constructed on the top of the region of interest. Where the first level of the pyramid is a region of interest itself. On the second level of pyramid, the region is divided into four cells (2x2 grid) and on the third level, region is divided for example into 16 cells (4x4 grid). Average or max pooling is applied to each cell. If the last convolutional layer has for instance 256 maps, then pooling in each cell produce one vector with length 256. Feature vectors for all cells that were generated are then concatenated, and then passed as input to the fully connected layer. fig.1.7

SPP-net not only hepls making representations from arbitrarily sizes, but also hepls during training, because it allows us to feed in the variable-size images during training. Which increase scale-invariance and reduce over-fitting.

SPP-net for object detection works similarly to R-CNN but it eliminates some of its drawbacks the most significant is the speed, which is of magnitude faster.

In the first stage of Detection Algorithm of SPP-net for Object Detection is used algorithm for selective seach that is similar to that in R-CNN. Then the image is resized and feature map is extracted from entire image at once. Which is a big advantage against R-CNN and is the cause that SPP-net is way much faster than R-CNN, where feature map was generated for each candidate. Then in each candidate is used Spatial Pyramid Pooling and these representations are provided to fully-connected layers.

SPP-net-based system for object detection is 24-102x faster than R-CNN, while having better or same accuracy. It takes about 0.5s to process image what makes it practical for real-world applications.

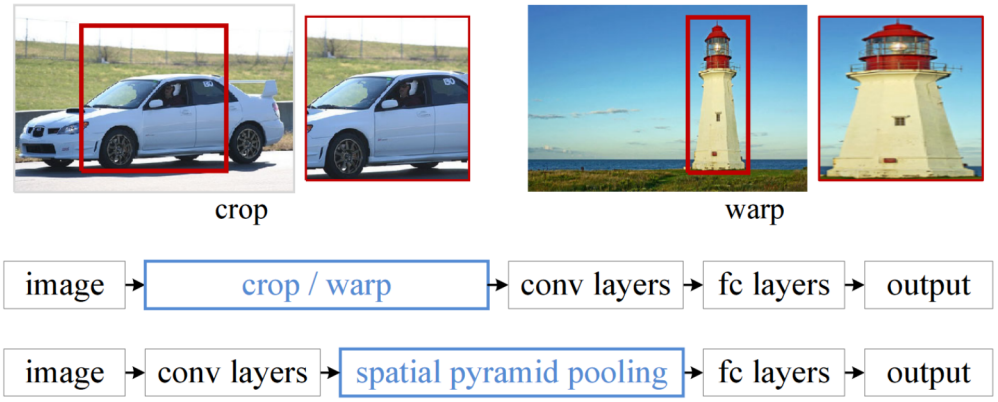


Fig. 1.5: Top: cropping or warping to fit a fixed size. Middle: a conventional CNN. Bottom: spatial pyramid pooling network structure.

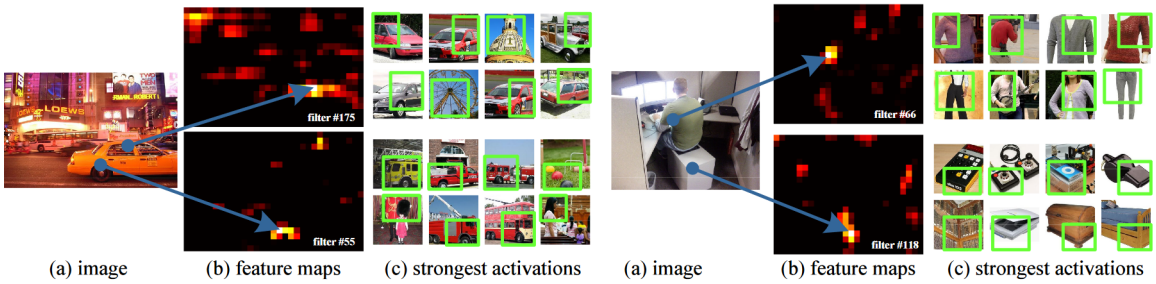


Fig. 1.6: Visualization of the feature maps. (a) Two images in Pascal VOC 2007. (b) The feature maps of some conv5 filters. The arrows indicate the strongest responses and their corresponding positions in the images. (c) The ImageNet images that have the strongest responses of the corresponding filters. The green rectangles mark the receptive fields of the strongest responses.

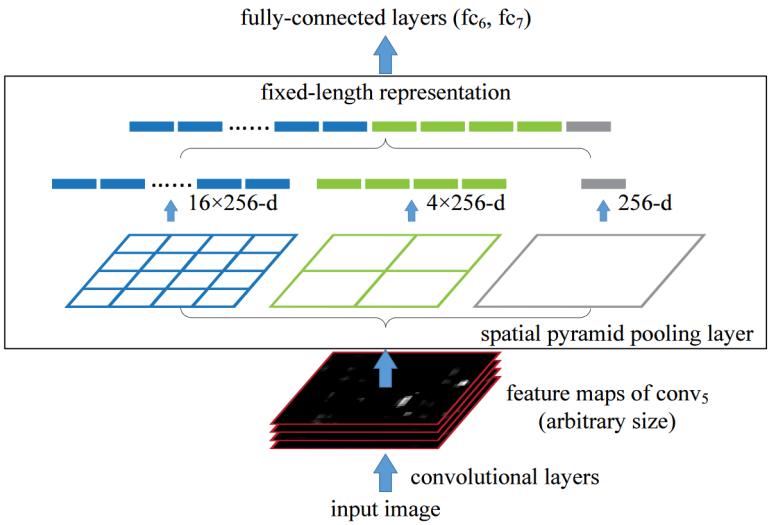


Fig. 1.7: A network structure with a spatial pyramid pooling layer. Here 256 is the filter number of the last convolutional layer.

1.4 Fast R-CNN

[10] This network method was introduced by Ross Girshick in Microsoft Research in September 2015. It is built on previous R-CNN and SPP-based network. Besides better accuracy Fast-RCNN is much more faster than R-CNN in detecting as well as in training.

Fast R-CNN gains from advantages that SPP-based-network posses, for instance the feature map is made only once per image and in the beginning we don't warp the candidates. But fast R-CNN also solves some drawbacks that SPP-based Network still possess:

- 1. Training is multi-stage pipeline that involves extracting features, fine-tuning a network with log loss, training SVMs, and finally fitting bounding-box regressors.
- 2. All features are also written to disc as in R-CNN.
- 3. Unlike R-CNN, the fine-tuning algorithm in SPP cannot update the convolutional layers that precede the spatial pyramid pooling.

Fast R-CNN comes with a new features that fixes the disadvantages of both R-CNN and SPP-net, while improving on speed and accuracy.

Fast R-CNN takes as input whole image and region proposals. The network firstly processes whole image through convolutional and max-pooling layers to create the feature map. Then for each region proposal a region of interest pooling layer extracts constant-size feature vector for fully connected layers that branch into two sibling output layers. Where one ouputs the softmax probability of K object classes.

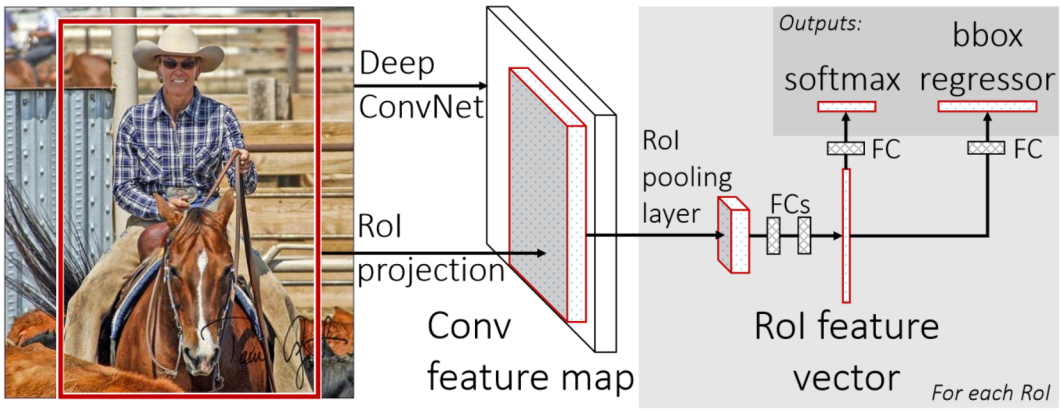


Fig. 1.8: Fast R-CNN architecture. An input image and multiple regions of interest (RoIs) are input into a fully convolutional network. Each RoI is pooled into a fixed-size feature map and then mapped to a feature vector by fully connected layers (FCs). The network has two output vectors per RoI: softmax probabilities and per-class bounding-box regression offsets. The architecture is trained end-to-end with a multi-task loss.

And second outputs 4 real-valued numbers refining bounding-box posistion for each of the object classes. fig.1.8

Region of interest pooling layer is modified spatial pyramid pooling layer where we use only one pyramid level. So we use max-pooling in each cell of constant 7x7 grid.

To eliminate our third problem where the back propagation through spatial pyramid pooling layer is highly inefficient, we sample minibatches hierarchically, first by sampling N-images(typically 2) and then by sampling R/N(ordinary 64) regions of interests from each image. This method unlike the SPP-based network takes advantage of sharing features of one whole image during training. This also saves required memory for training.

For streamlined training process(where in one stage softmax classifier is jointly optimized with bounding box regresor) we use for training a multi-task loss. This multi-task loss is the rate of sum of classification loss (log. loss) and bounding box regression box loss(L1 loss). Because there is not separate SVM training, therefore end-to-end training is allowed, what is much faster and doesn't require intense write and read to the hard drive. Precision was also improved.

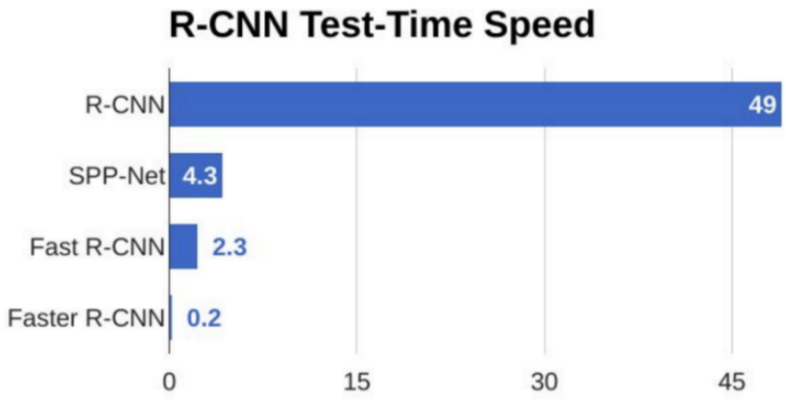


Fig. 1.9: Comparison of test-time speed of object detection algorithms.

1.5 Faster R-CNN

[11] Computaional demands drastically decreased from R-CNN to fast R-CNN, thanks to sharing of feature map. Now the only handicap appears to be the region proposals. Region proposal takes as much time as the detection network. Faster R-CNN comes with solution where proposals are computed by Region Proposal Network (RPNs) that shares convolutional layers with object detection network, that was described in previous sections.

1.5.1 Region Proposal Network(RPN)

RPN is a kind of fully convolutional network[12] and serves as a proposal generator. And can be made by adding a few additional convolutional layers on the top of shared convolution layers. In the original paper of faster R-CNN[11] was used Zeiler and Fergus model[14](ZF), which has 5 shareable convolutional layers and the Simonyan and Zisserman model[15](VGG-16), which has 13-shareable layers.

Experiments in neural network visualizations have shown that by decoding convolutional responses that comes from one place, we can still roughly guess the object outline. And that is what RPN does.

Region Proposal Network is a relatively small network that is slided over convolutional feature map. Its input size is usually 3. RPN simultaneously classify the object as unknown object and regress the bounding box locaion. Position of the sliding window provides localization information, with reference to the image and box regresion provides finer localization with reference to this sliding window. At each sliding window position is defined a set of object proposals. Each of these proposal has different size and aspect ratio. Such proposals are called anchors.

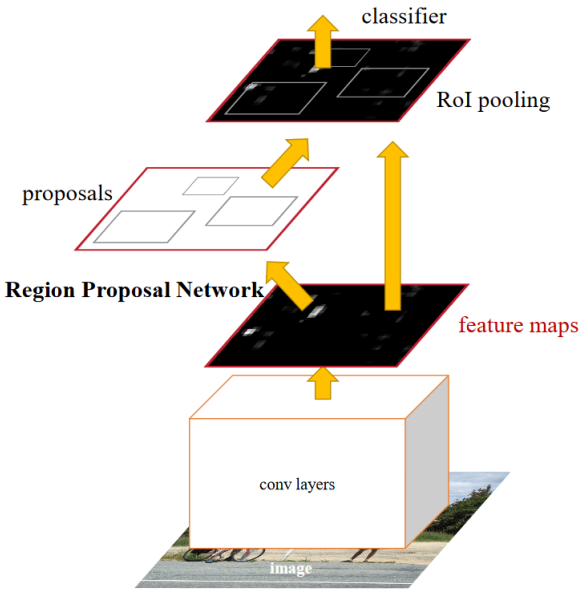


Fig. 1.10: Faster R-CNN is a single, unified network for object detection. The RPN module serves as the 'attention' of this unified network.

In training process of RPN the proposed anchors are marked as positive example if intersection over union with ground true particular example is larger than 0.7, or if none of them reached this value we take anchor that reached the maximum value of intersection. And as negative examples are anchors marked if intersection over union is lower than 0.3. The box regresion is trained to regress the boxes of positive examples to ground truth boxes.

However, it can be very difficult for RPN to handle objects of very different scales, because the receptive field is fixed. There appears problems when receptive field is larger that receptive field or opposite, as you can see in the fig.1.11. This problem is sloved by using RPNs of various scales. These various scales are achieved by placing each RPN after different convolutional layers, so the receptive fields will be of various sizes fig1.12. This method will significantly improve network ability to detect objects of various sizes.

1.5.2 Training Faster R-CNN

Faster R-CNN is trained as one network using the following four losses:

1. RPN classification loss (good/bad anchor)
2. RPN regression loss (anchor->proposal)
3. Fast R-CNN classification loss (over classes)
4. Fast R-CNN regression (proposal->box)

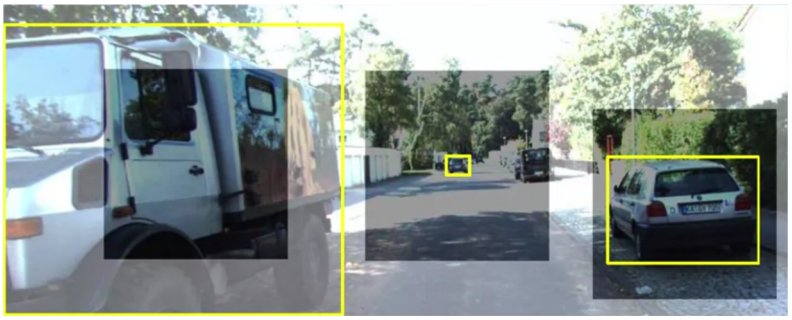


Fig. 1.11: RPN problem with objects of various sizes. I the left the receptive field is much smaller than object itself and in the right the object is much smaller than receptive field.

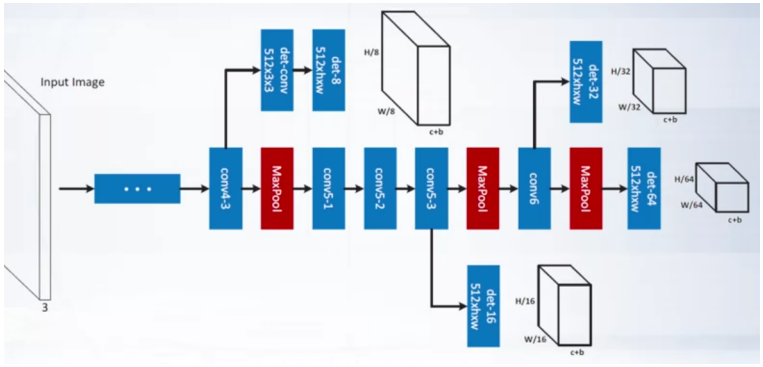


Fig. 1.12: Structure of using multiple RPNs for various sizes of receptive fields.

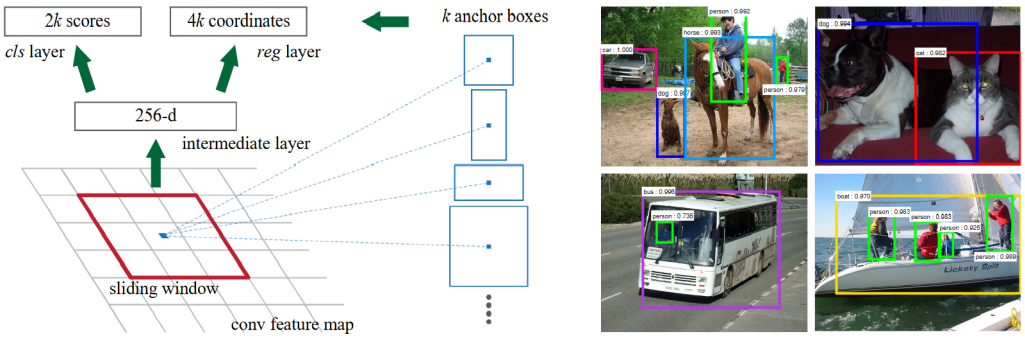


Fig. 1.13: Left: Region Proposal Network (RPN). Right: Example detections using RPN proposals on PASCAL VOC 2007 test. Method detects objects in a wide range of scales and aspect ratios.

1.6 YOLO

[16] YOLO (you only look once) comes with revolutionary approach of detection that is more simple than complex faster R-CNN and it is remarkably faster. Base model of YOLO process images in real-time at 45 frames per second. Smaller version, fast YOLO process 155 frames per second, while still achieving double the mAP of other real-time detectors. Compared to state-of-art detection systems, YOLO makes more localization errors but is less likely to predict false positives on background. YOLO is also better in learning general representations of objects, therefore is better performing in other domains of images like artwork Fig.1.16.

YOLO was inspired by human glance at an image, where humans instantly know what objects are in the image, where they are and they interact.

1.6.1 Unified detection

YOLO uses single convolutional network that simultaneously predicts multiple bounding boxes and class probabilities for those boxes. This also enables end-to-end learning.

Here is how YOLO works: Firstly is the image divided into SxS grid. Then each grid cell predicts B bounding boxes, that are assigned to those predefined anchors that has similar shapes.

Each bounding box consist of 5 predictions: x, y, w, h and confidence. X, y represents coordinates of box center, they are relative to the bounds of grid cell. W and h represents width and height of box relative to the whole image. Finally confidence represents the intersection over union of predicted box with ground true box(IOU), in other words this number reflects how certain is the network that the box is accurate and how certain it is that the box is containing an object.

Each grid cell also predicts C conditional probabilities, Pr(Class|Object). These probabilities are conditioned on the grid cell containing an object. Only one set of probabilities are predicted per grid cell, regardless the number of boxes.

Then those two numbers are multiplied, which give us class-specific confidence score for each box. These scores encode both the probability of that class appearing in that box and how well the predicted box fits an object. Fig.1.15 In the original paper was used S=7, B=2, and number of classes C=20.

1.6.2 Selection of proposed boxes

We obtained yet SxSxB, bounding boxes in total, with their class specific confidence. As first step those class-specific confidences of each box are compared with some threshold (usually about 0.6), boxes with lower class-specific confidence are then

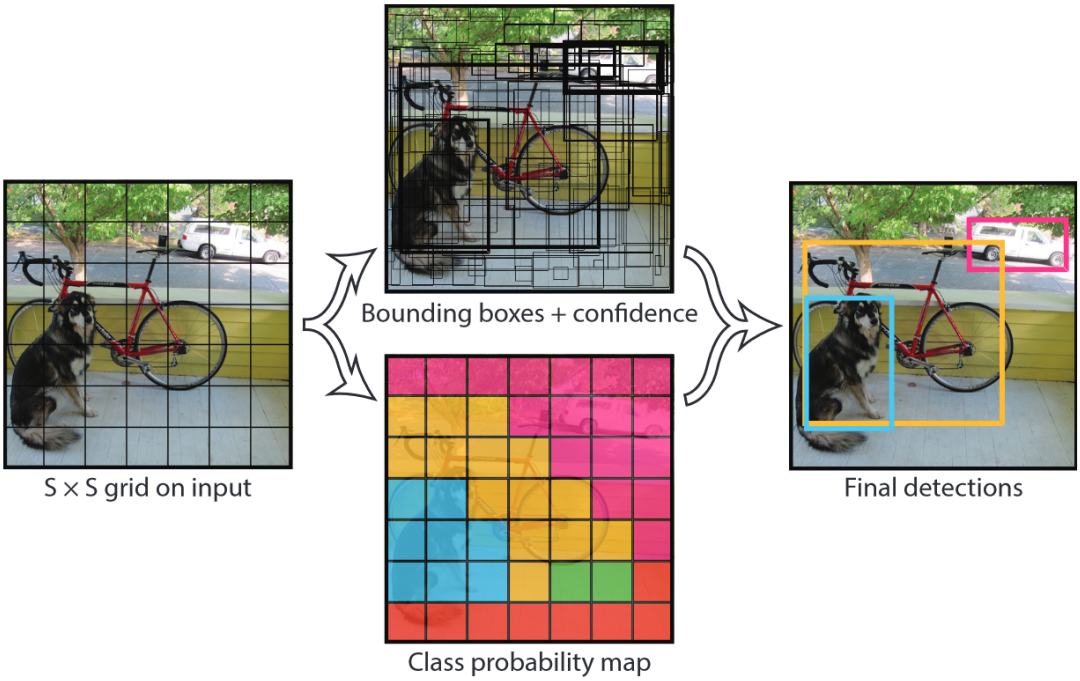
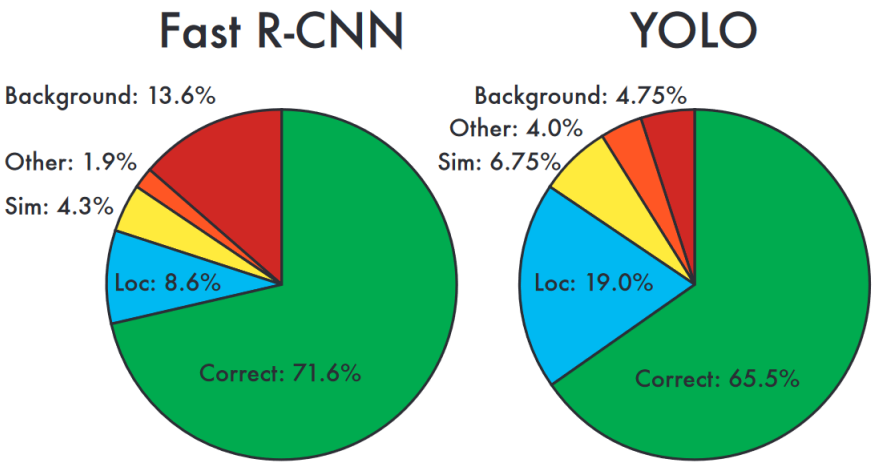


Fig. 1.14: The Model. Our system models detection as regression problem. It divides the image into an S×S grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an SxSx(B*5+C) tensor.



comparison.png

Fig. 1.15: Error Analysis: Fast R-CNN vs. YOLO These charts show the percentage of localization and background errors in the top N detections for various categories (N = objects in that category).

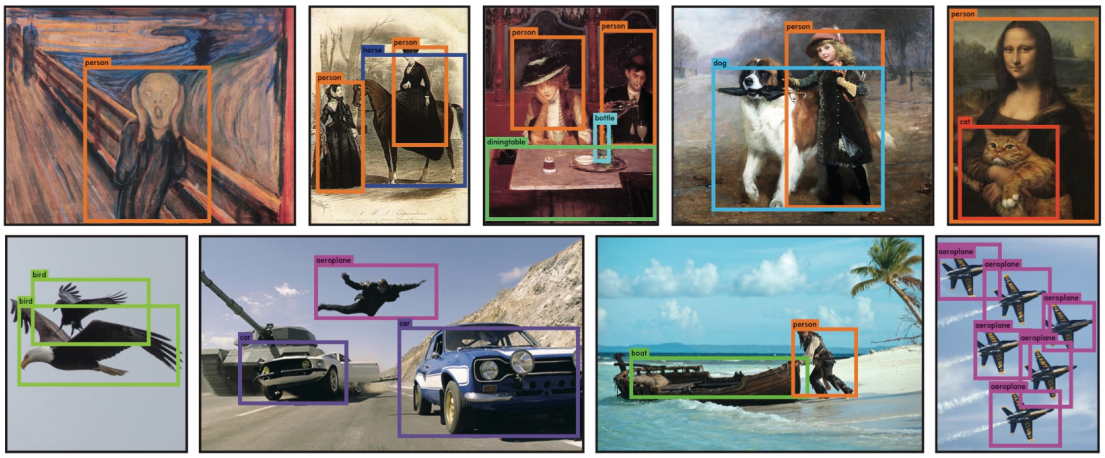


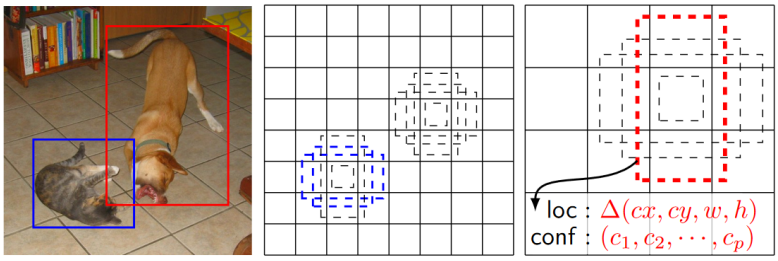
Fig. 1.16: YOLO running on sample artwork and natural images from the internet. It is mostly accurate although it does think one person is an airplane.

eliminated. Then the Non-Max Suppression algorithm is applied, which eliminates multiple boxes that corresponds to the same object. Here is how it works: Firstly we get box with the highest confidence and then delete all boxes that has higher intersection over union with that box than some threshold (ordinary 0.5). Then we repeat this process with the next remaining box with lower confidence until there is none box left.

1.7 SSD

[17] SSD (Single Shot MultiBox Detector) was introduced in December of 2016. Although the YOLO is much faster than faster R-CNN it posses worse detection accuracy (for PASCAL VOC 63.4% mAP), than faster R-CNN (73.2%). Most errors made by YOLO cames from images of different scales. This problem solves the SSD algorithm. Its principle of working is very similar to YOLO but has some other beneficial features.

The core of SSD is as in YOLO in the predicting fixed-size collection of category scores with box offsets for a fixed set of default-shaped bounding boxes(in YOLO called anchors), using small convolutional filters(in YOLO the spliting of picture into SxS grid) applied to the feature map. But in SSD we apply it to several feature maps of different resolutions, as resolution decreases from early layers to end. Then are used tresholding and non-max suppression to extract only relevant proposals.



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

Fig. 1.17: (a) SSD only needs an input image and ground truth boxes foreach object during training. In a convolutional fashion, we evaluate a small set (e.g. 4)of default boxes of different aspect ratios at each location in several feature maps withdifferent scales (e.g.8×8and4×4in (b) and (c)). For each default box, we predictboth the shape offsets and the confidences for all object categories ((c1,c2, · · · ,cp)).At training time, we first match these default boxes to the ground truth boxes. Forexample, we have matched two default boxes with the cat and one with the dog, whichare treated as positives and the rest as negatives. The model loss is a weighted sumbetween localization loss (e.g. Smooth L1 [6]) and confidence loss (e.g. Softmax).

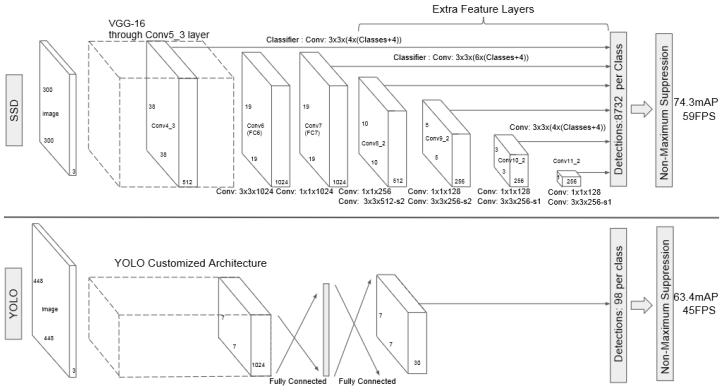


Fig. 1.18: A comparison between two single shot detection models: SSD and YOLO [5].Our SSD model adds several feature layers to the end of a base network, which predictthe offsets to default boxes of different scales and aspect ratios and their associatedconfidences. SSD with a300×300input size significantly outperforms its448×448YOLO counterpart in accuracy on VOC2007testwhile also improving the speed.

2 Datasets

2.0.1 Caltech Padestrian Detection Dataset

[20] The first version of this dataset was introduced by Piotr Dollár on his site in 2009. It was made for training and evaluating the padestrians detection algorithms. There are also charts of the most advanced detection algotihms at that time. This

Fig. 2.1: Caltech dataset samples: images with labels.

dataset consists of approximately 10 hours of 640x480 30Hz video, taken from vehicle driving through regular traffic in urban environment. I is about 250,000 frames in 137 approximately minute long segments, with a total of 350,000 bounding boxes and 2300 unique padestrians were anotated.

The whole dataset is divided into two parts:

- training data consists of six training sets, approximately 1GB each and each contains 6-13 one-minute long seq files, along with all annotation information.
- testing data consists of 5 similar sets.

2.0.2 Seq Video Format

A seq file is a series of concatenated image frames with a fixed size header. It is essentially the same as merging a directory of images into a single file. Seq files are convenient for storing videos because:

- no video codec is required
- seek is instant and exact
- seq files can be read on any operating system

This datasets websitealso contains authors Matlab toolbox for handlig seq files (reading/writing/manipulating). It can also be used to to extract an seq fie to a directory of images.

2.0.3 Annotations

The annotations use a custom "video bounding box" (vbb) file format. The site also contains utilities to view seq files with anotations overlaid (the labeled boxes). It also contains evaluation routines for evaluating algorithms and labeling tool that was used to create the dataset and can be used for creating a new dataset.

2.0.4 Conclusion

Since this dataset consists of videos it contains a huge number of images what is only good for machine learning in computer vision. But there may also appears a problem that the network will overfit those images, because they were all taken in the same enviroment. And sequence of images from video are verry simmilar others in that video.

2.1 INRIA Person Dataset

[21] This dataset was collected as part of research work on detection upright people in images and video. The research is described in paper Histograms of Oriented Gradients for Human detection[1] by Navneet Dalal and Bill Trigg in 2005.

The dataset contains PNG images from these sources:

- Images from older dataset Graz01, but annotations were made newly.
- Authors personal digital image collections.
- Images taken from the web using google images.

Author also mentioned that only upright persons (with height over 100cm) are marked in each image. And not all annotations may be 100% right.

Dataset contains 2 types of "subdatasets":

- Original Images
- Normalized Images

2.1.1 Original Images

Contains original, non resizes images and Consists of two folders 'Train' and 'Test'. Both folders have three sub folders: (a) 'pos' (positive training or test images), (b) 'neg' (negative training or test images), and (c) 'annotations'. Annotation files for positive images are in Pascal Challange format.

Fig. 2.2: INRIA dataset samples.

2.1.2 Normalized Images

Contains normalized images of sizes 64x128, 96x160, 70x134 pixels with margins around each side for avoiding boundary conditions. It also contains 64x128 images without margins. Each folder of Normalized images is divided same sa folders with original images.

2.1.3 Conclusion

This dataset contains approximately 2500 images what is much less than in Caltech dataset, but INRIA images are much more varied thus could make detection system more general. Other problem may be the low resolution of images. Also small and not-upright people are not labeled, which could be in some cases usefull.

2.2 Penn-Fudan Database for Pedestrian Detection and Segmentation

[22] The images are taken from scenes around campus of University of Pennsylvania and Fudan and urban street. Each image has at least one pedestrian in it. The heights of labeled pedestrians in this database fall into [180,390] pixels. All labeled pedestrians are straight up.

There are 170 images with 345 labeled pedestrians, among which 96 images are taken from around University of Pennsylvania, and other 74 are taken from around Fudan University.

The annotation format is compatible with PASCAL Annotation Version 1.00.

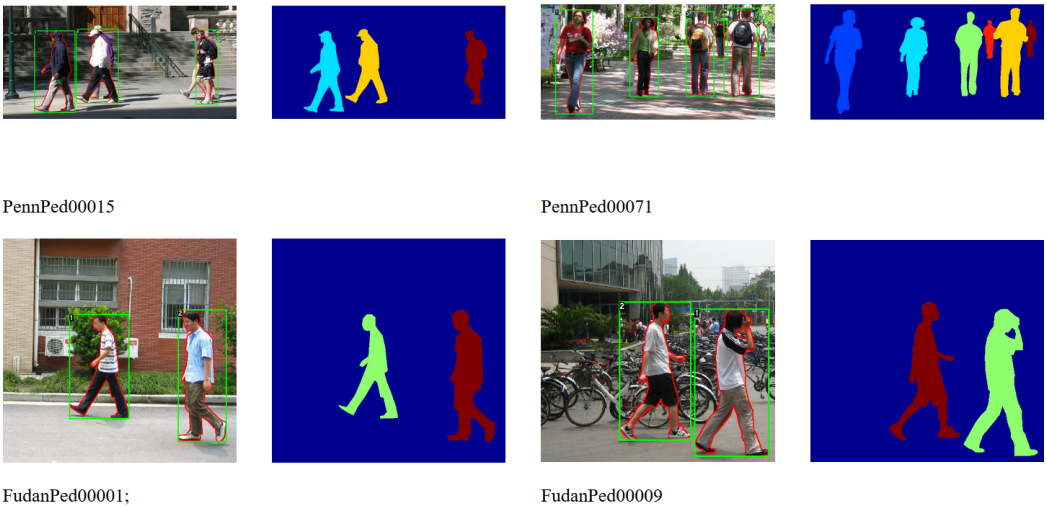


Fig. 2.3: Penn-Fudan example images with labeled masks and detection boxes.

2.2.1 Conclusion

This dataset is the smallest of all mentioned datasets. Advantage may be the precise bounding boxes.

2.3 GM-ATCI Rear-view pedestrians dataset

[23] This dataset was collected using a vehicle-mounted standard automotive rear-view display camera for evaluating rear-view padestrian detection. The production 180 degrees fisheye-lens camera was mounted in front of the vehicle in the typical rear-view installation pose: 107cm height and 25 degrees downward tilt angle. The dataset contains 15 filming sessions, each taken in a different day with different scenarios. Each session contains multiple clips with duration ranging from several seconds to several minutes. In total, the dataset contains 250 clips with a total duration of 76 minutes and over 200K annotated pedestrian bounding boxes. But for free there is only available the test portion comprising 70 of the clips. There are two types of sessions, containing either staged or "in-the-wild" pedestrians. The staged scenarios include mainly pedestrians walking in front of the camera at different positions and directions in a controlled manner, spanning the different use cases of a rear alert or braking automotive feature. In the remaining sessions the vehicle drove either in public roads or in parking lots and captured incidental pedestrians. The different locations include: indoor parking lots, outdoor paved/sand parking lots, city roads and private driveways. Authors filmed both day and night scenarios,

Fig. 2.4: GM-ATCI Rear-view pedestrians dataset examples of output of some detection system.

with different weather and lighting conditions. More information can be found in the rear-view detection system papers[18][19].

The videos are same as in Caltech dataset in Seq format and annotations in vbb.

2.3.1 Conclusion

This dataset contains relatively big number of annotated padestrian boxes also of non-upright people. There was used camera that wasn't used in any other dataset, what makes this dataset only beneficial, also in general the detection systems has a high potential to be used with car cameras like was used in making this dataset.

2.4 COCO (Common objects in context)

[24][25] This relatively new dataset firstly appeared in 2014 with support from Microsoft. Its goal is advancing the state-of-art in object recognition by placing the question of object recognition in the context of the broader question of scene understanding. This is achieved by gathering images of complex everyday scenes containing common objects in their natural context. Dataset contains photos of 91 object types, with total of 2.5 million labeled instances in 328k images. It supports novel interfaces for category detection, instance spotting and instance segmentation.

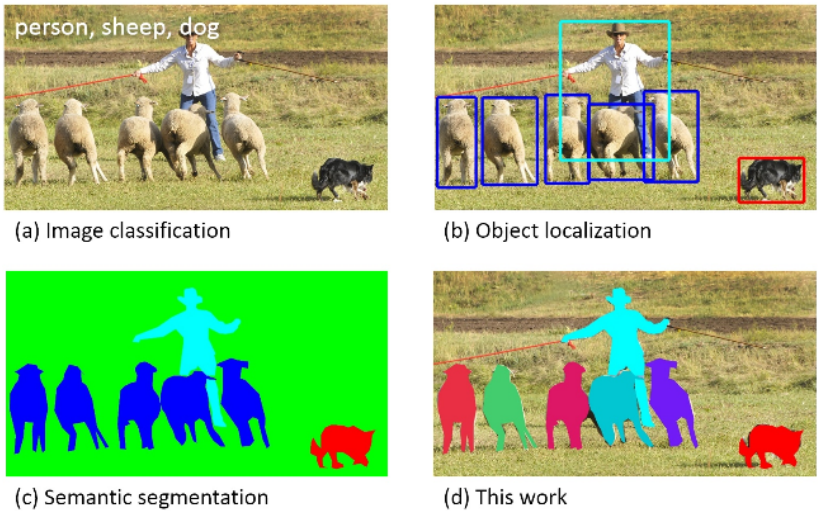


Fig. 2.5: COCO example - (a)image classification, (b) object detection, (c) semantic pixel-level segmentation, (d) segmenting individual object instances.

2.4.1 Conclusion

As this dataset is composed of 328k images, where approximately half of them contain person, it makes this dataset very useful. Its annotations are also very precise and images comes from various contexts and scenarios.

2.5 Other useful padestrians datasets

- Daimler Mono Pedestrian Detection Benchmark Dataset[26] Fig.2.6
- Tsinghua-Daimler Cyclist Detection Benchmark Dataset[27] Fig.2.7
- Data61 Pedestrian Dataset from NICTA[28]
- CVC-ADAS: datasets including pedestrian videos acquired on-board, virtual-world pedestrians (with part annotations), and occluded pedestrians[29] Fig.2.8

Fig. 2.6: Daimler Mono Pedestrian Detection Benchmark Datase examples.

Fig. 2.7: Tsinghua-Daimler Cyclist Detection Benchmark Dataset examples.

Fig. 2.8: Virtual-world padestrian dataset

3 Practical part

Up to now, we were dealing only with possibly useful algorithms and datasets rather in the theoretical level. In this chapter practical things will be tried, one algorithm will be chosen, one set (representing dev and test dataset at once) will be composed from mentioned datasets. And then chosen Algorithm will be trained on various types of set and tested on composed "test" set.

3.1 Choice of algorithm

If one look for simplicity of implementation, then Sliding-Window Detector would be the best option by far, but there are already many of precisely implemented models available in open-sourse. The most accurate appears to be faster R-CNN or possibly YOLO which makes less positive errors. SSD is relatively new and not well developed yet with eventual potential, but that task would be far beyond scope of this thesis. Moreover if we take into account the speed of detection, then YOLO is the best option, therefore in the ongoing sections we will be handling with YOLO algorithm in its most advanced version the YOLOv3 (see YOLO version 3 improvements [30]).

3.2 Choice of software

Author of YOLO algorithm offers his implementation on his own website [31] for free. YOLO was there implemented in neural network framework named Darknet which was written in C and CUDA and supports CPU as well as GPU computation. Manual for his implementation is also described on mentioned website.

On some github repositories such as [32] are available the Tensorflow implementations of YOLO. This could appears more "friendly" to those who uses Tensoflow and wants to try tweaking the implementation possibly with Tensorboard environment. What is also favourable is that functions for the detection that are available on this github repository can be simply used in other python applications, but the chosen implementation comes from this github repository [33]. It offers Windows and linux version of Darknet YOLOs v3 and v2, where also modern tensor cores can be used.

Here are the main improvements in chosen repository because of which it was picked out:

- improved neural network performance Detection 3x times, Training 2 x times on GPU Volta (Tesla V100, Titan V, ...) using Tensor Cores
- added correct calculation of mAP, F1, IoU, Precision-Recall

- added drawing of chart of average-Loss and accuracy-mAP during training

3.3 Choice of hardware

As my laptops graphical card (Intel HD) does not support CUDA parallel computing platform, it makes detection using my laptops CPU not very nimble. It takes approximately 30s to complete detection on the single frame after the compilation was completed. After taking into consideration that training is even more tedious process, I decided to use cloud service.

One option is Floydhub that is deep learning platform which offers cloud GPU units. But it is not so well designed for chosen type of project (not python project), therefore the classical cloud machine from Google Cloud Platform was chosen. This instance posses:

- 4 vCPUs, 15 GB memory
- GPU - NVIDIA Tesla T4 (posses 2,560 CUDA cores with 320 tensor cores)
- OS - Microsoft Server 2016

Great advantage is also that we can change number of GPUs in our VM instance according to our demands anytime.

3.4 Running program and how to use it

3.4.1 Program compilation

For project compilation the following dependencies had to be installed:

- CMake
- MS Visual Studio 2015/2017/2019
- CUDA $>$ 10.0, cuDNN $>$ 7.0
- OpenCV $>$ 2.4

After installing all the mentioned dependencies, setting all the required system variables and cloning chosen repository, generation of project may be started using CMake (generator to x64). Configuration should look as in Fig.3.1. Then the project is just build in release mode using Visual Studio.

3.4.2 Running program

In this section are described only basics of launching program that are sufficient for our purposes. Program is run by command line from its folder by typing:

"darknet.exe detector [mode] [data file] [cfg file] [weights] [flaq] [ext. output file if -ext_output flaq was added]"

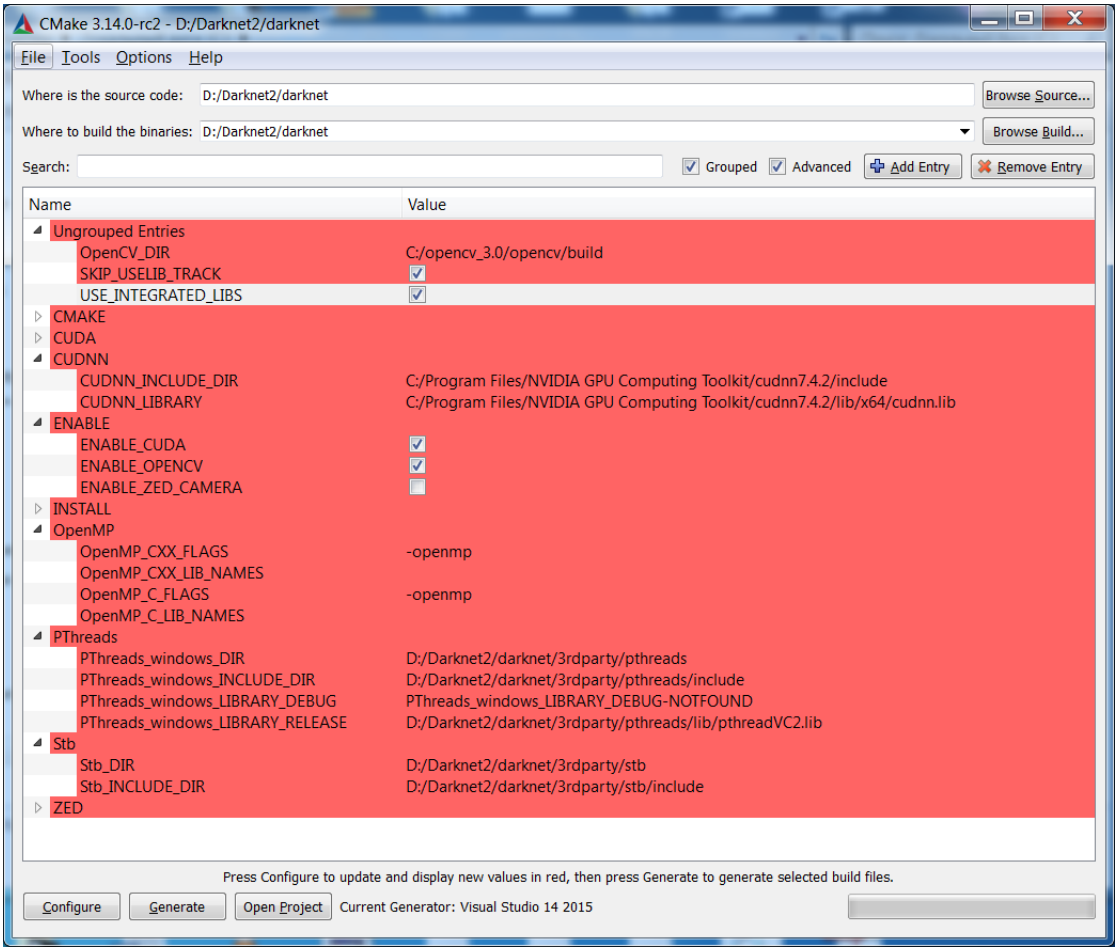


Fig. 3.1: CMake configuration

In the folowing subsections are all the arguments described:

Mode

If one choses "test"-mode it makes detection on given photo, where the output is the given picture with bounding boxes and labels.

Using "demo" mode we can make detection on video, webcam and external device.

We can also train out network using "train" argument.

Data file

This argument refers to the small 5-lines long file which defines number of classes, relative paths to text files containing relative paths to training and all the validation images. Also defines relative path to "names" file where are just listed names of classes that corresponds to the first number in YOLO annotation format, which represents class type. Backup variable defines the relative path to location of saving the trained weights after every 1000 iterations and last trained weights before program stopped. (Relative paths are meant to be relative to darknet.exe file.)

Cfg file

These prearranged "cfg" files are located in folder named cfg. Purpose of cfg-file is definition of layout and configuration of all the convolutional and "YOLO" layers. As far it is not in our intention to tweak configuration of network, we let the convolutional layers unchanged. What we have to change are parameters "classes" an "filters" ((classes+5)*3) in YOLO layers according to number of classes that are being detected. (In yolov3 there are 3 YOLO layers in different "depths" of the network to detect objects of various sizes.)

There are also defined other parameters that configures network and training, such as batch size, learning_rate, max_batches, anchors,...

For our purpose of training the network, we will change the batch and subdivision size to 64 which is recommended for our size of sets. max_batches defines for how much iterations we want our training to last.

3.4.3 Flaq and ext. output file

We can determine on which picture or video we want our detection to be done by the last parameter, but before we have to type flaq -ext_output. There is are also available -map flaq to show mAP chart during training.

3.5 Putting together test set

I want to mention that this set will represent the test and validation set at once. In this work I will reffer to it just as test set.

After consulting the test set problem with the head of my thesis. I decided to compose test set from 2 datasets:

- Inria person dataset
- COCO dataset

The main reason is that other datasets are not very precisely labeled and leads to wrong evaluation in testing. Furthermore other datasets has almost the same context of images. Therefore it does not matter whether we use more datasets. Test dataset consists of 600 images, where 400 comes from Inria and the ramining 200 from Coco. Inria dataset has more simmilar content to CityPerson dataset. Therefore I have chosen bigger number of images from this dataset, despite the coco is much bigger.

To remain the distribution of content of datasets I have copied random images to test set, using command line command:

"gci sourceFolder | random -c numOfImages | mi -dest destinationFolder"

3.6 Preparing the project for training

YOLO dataset format requires images and text files with defined labels to be in the same folder. One text file corresponds to one image where text file has to have the same name as corresponding image. Each line in text file is for each bounding box in the image. The labels definitions are in format:

<object-class> <x> <y> <width> <height>

Where x, y, width, and height are relative to the image's width and height. X and y defines position of center of image.

There are available programs and scripts to convert other formats of datasets into YOLO format. For example in project folder named "scripts" is present python scipt named "voc_label.py" for converting Pascals Voc labest to our text files.

In this link [34] are present scripts for converting labels from "vbb" format that was used in Caltech or Daimler Mono Pedestrian Dataset. And scipt to convert "seq" video format to separate images.

For converting COCO "json" format there is available pre-build Java program in this link [35]. The maual how to use it is present.

For training we need to have:

- 2 folders with pictures and "txt" files one folder for training and second for evaluation.

- 2 "txt" files with relative paths to every image, one "txt" file for each folder with images. For making that file I made simple python script "sprav_txt_s_cestami" that generates those paths. format of command is following:
  "python sprav_txt_s_cestam [name of "txt" output fie] [folder containing images]"
- correctly configured "data" file as we described in subsection "Data file" from section 3.5.2.
- "cfg" file with defined network structure (we use Darknet-53 see Fig.3.2) and variables configuring training process (we had to change number of classes to 1 and filters to 18 for every YOLO layer, baches to 64, max_batches and steps according to concrete training set. Anchors we let in default as we work with various-sized images.).
- Pre-trained weights, I used pre-trained weights on Imagenet from network configuration Darknet-53, that are available on YOLO authors website in this link[31].

3.7 Script that draws YOLO predictions, ground true boxes and IOU in test images

This script outputs given images with predicted boxes by YOLO (green color), gound true boxes (blue color) and intersection over union (orange color), see in fig.3.4. This script takes as input path to directory with images with its predicted labels and the second argument is path to directory with ground true boxes. It outputs the edited images to other predefined folder, but before we can run that script we need to execute the "pseudo" labeling by our trained model by command:

"darknet.exe detector test data-file cfg-file weights -dont_show -save_labels < text-file"

where text-file contains relative paths to images we want to make labels.

3.8 Training network on training sets of various sizes

In this section will be examined the dependency of mAP on number of images in testing set. These images in training set will come only from Coco dataset. It is also ensured that there are not same image twice in every training and test set.

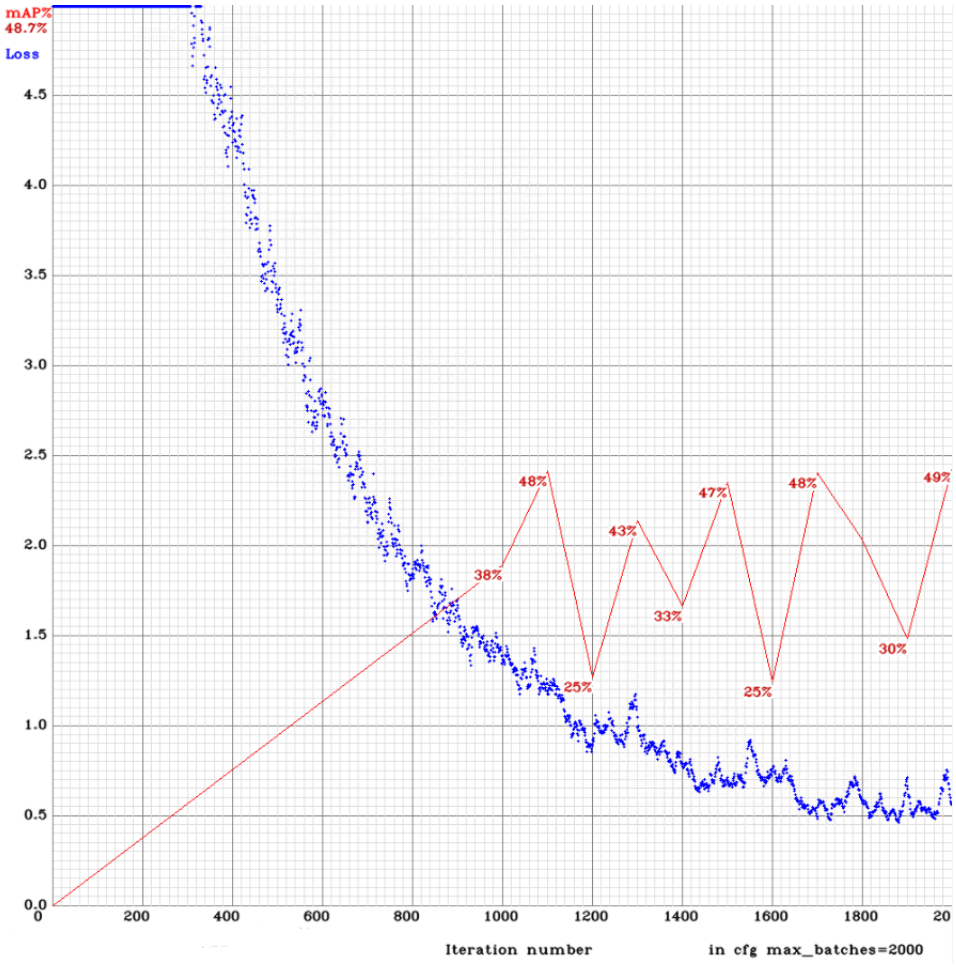
	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Fig. 3.2: Darknet-53, the network specialized for image classification which is used as feature extractor

3.8.1 Training on set of 100 images from Coco

This number of images was the lowest for which I succeeded in training the network to "meaningful" state. As you can see in Fig.3.3 model reaches its settled mAP value, 38% on average after just 1000 iterations with relatively sharp fluctuation. Output makes a lot of true positive as well as false positive errors and labels has poor intersection over union with ground true boxes, see fig.3.4.

Fig. 3.3: Grafh of average loss and mAP for 100 images from Coco dataset



3.8.2 Training on set of 200 images from Coco

See in Fig.3.5 model reaches its settled mAP value, 49% on average after approximately 1200 iterations.

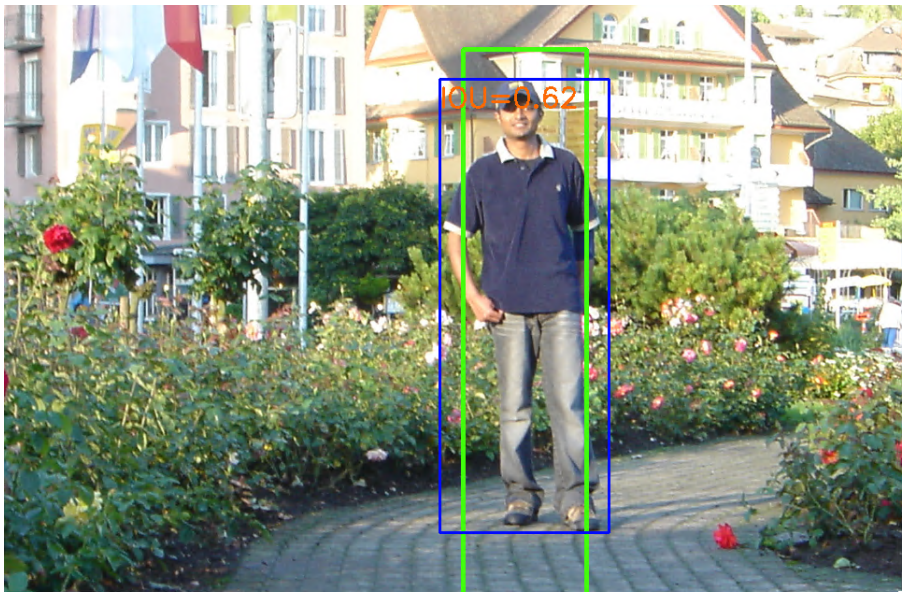


Fig. 3.4: Test Image with ground true box and YOLO prediction (net trained on 100 images)

3.8.3 Training on set of 500 images from Coco

See in Fig.3.6 model reaches its settled mAP value, 56% on average after approximately 1500 iterations.

3.8.4 Training on set of 1000 images from Coco

See in Fig.3.7 model reaches its settled mAP value, 58% on average after approximately 1300 iterations. And appears to be still growing, therefore I set max-batches to 2000 set to 8000 in the next training.

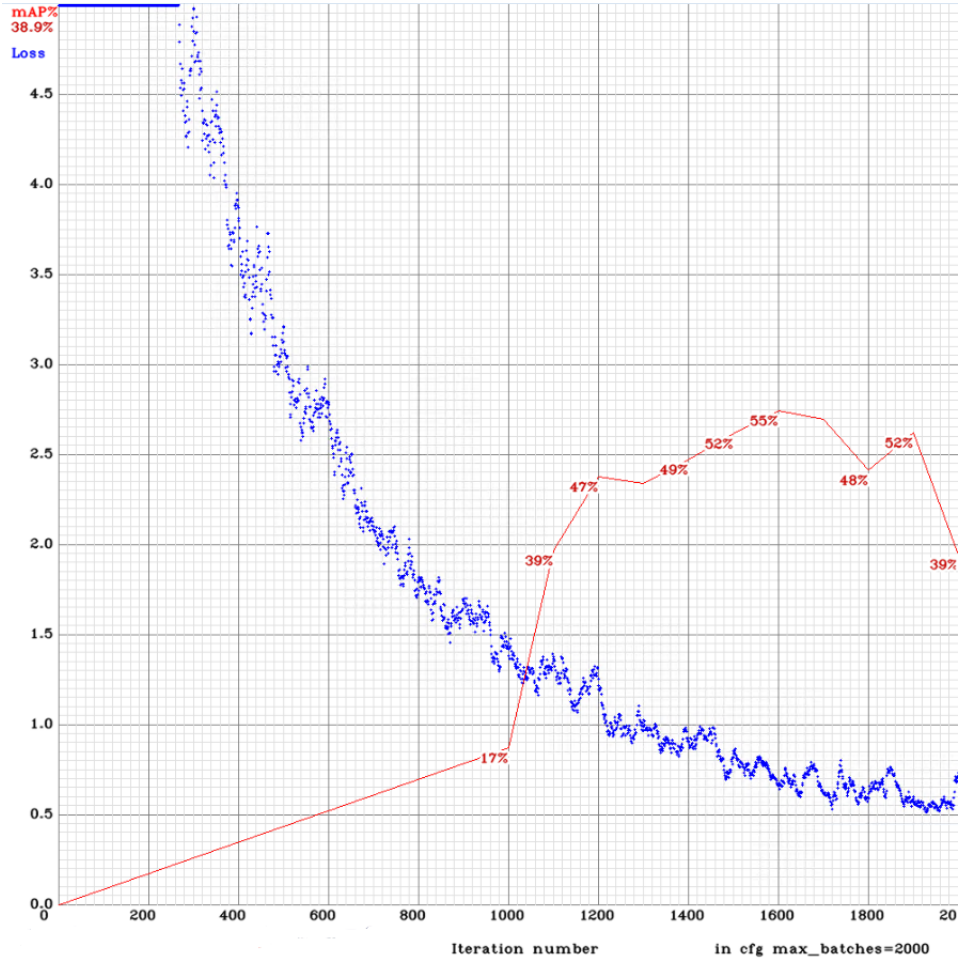
3.8.5 Training on set of 2000 images from Coco

See in Fig.3.8 model reaches its mAP settled value, 61% on average after approximately 1100 iterations.

3.8.6 Training on set of 4000 images from Coco

See in Fig.3.9 model reaches its mAP settled value, 65% on average after approximately 1500 iterations. And appears to have the lowest fluctuation. As we expected the achieved mAP was the highest from previous examples. As you can see in Fig.3.10 the intersection over union can reach very high values in contrast to model trained on 100 mages Fig.3.4. But still we can find some images with false positive

Fig. 3.5: Grafh of average loss and mAP for 200 images from Coco dataset

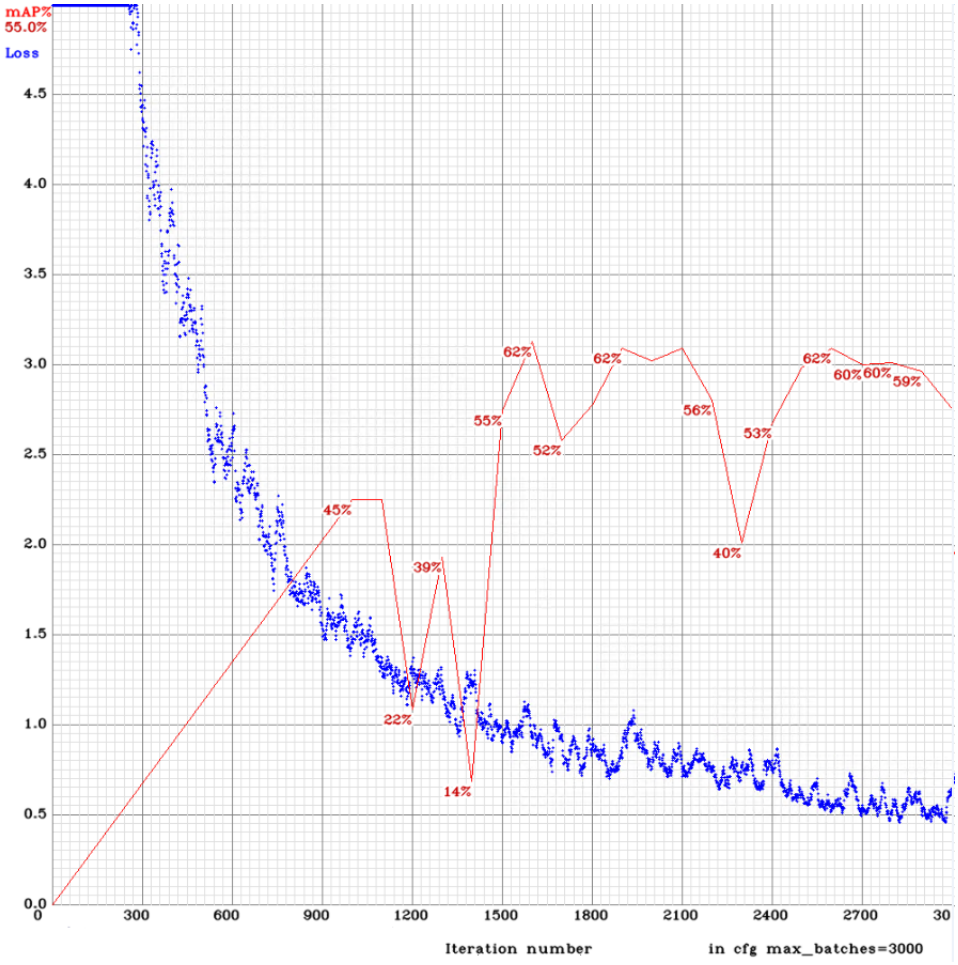


and true negative errors mainly made in photos from unusual environment, with low illumination Fig.3.10.

3.8.7 Conclusion on dependency of model precision on number of training images

In the table Tab.3.1 and chart Chart.3.8.7, one can see that the more training images the better results we can expect. But for bigger numbers of images (about 2000) the improvement of mAP is almost insensible. In previous graphs it is also visible that with smaller sets the average loss decreases much quicker and to lower values. The number of iterations after which the models mAP precision settles does not depend on sets sizes. It means that model is trained after constant number of iterations (for one class approximately 1300 using batch size 64), no matter the size of set. In the summary we can say that ideal set size could be 2000 and number

Fig. 3.6: Grafh of average loss and mAP for 500 images from Coco dataset



of iterations approximately 1500 for detecting one class. What is interesting is that network does not seem to everfit training set even after big amount of iterations, that may be caused by batch normalization implemented in Darknet-53 (network used as feature extractor).

num. of images	100	200	500	1000	2000	4000
achieved mAP	38%	49%	56%	58%	61%	65%

Tab. 3.1: Dependency of achieved settled mAP on number of images from Coco dataset in train set.

Fig. 3.7: Grafh of average loss and mAP for 1000 images from Coco dataset

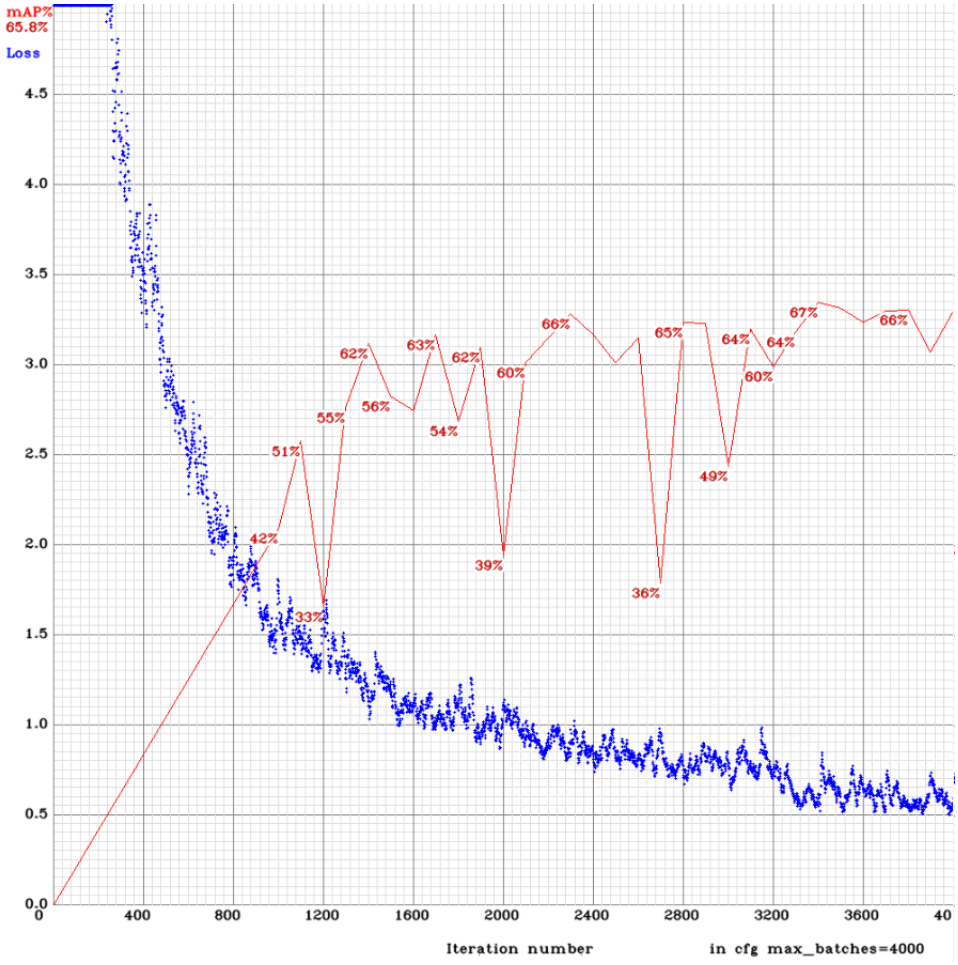


Chart 3.8.7: Achieved value of mAP on number of images.

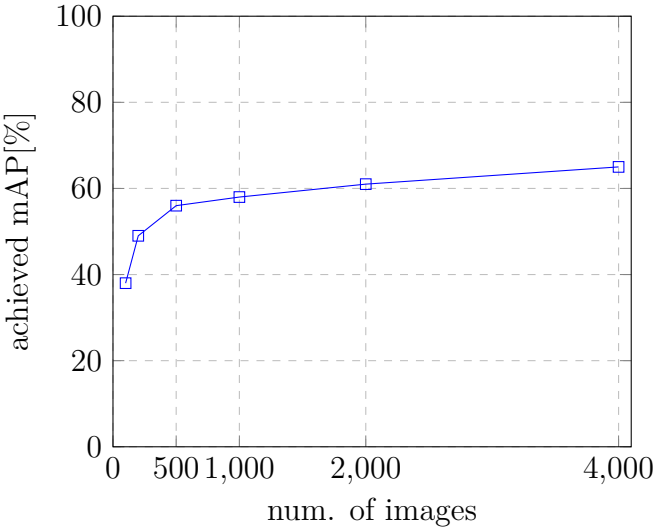
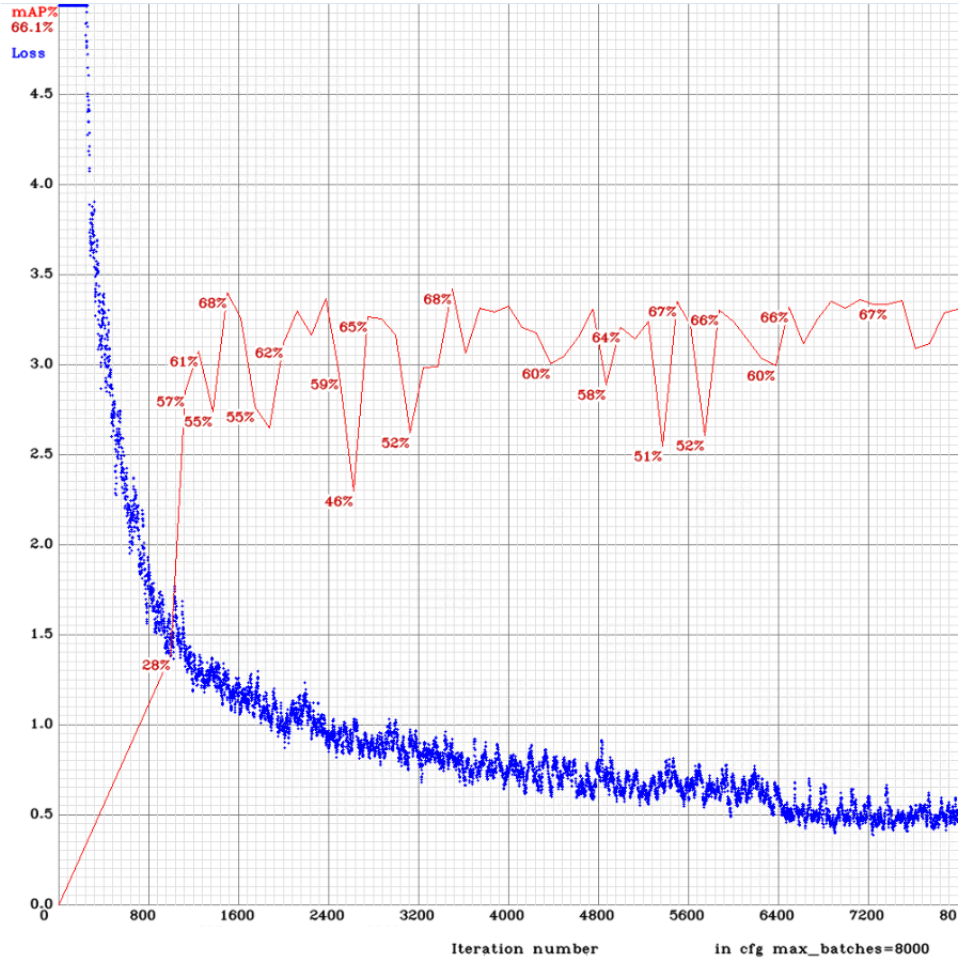


Fig. 3.8: Grafh of average loss and mAP for 2000 images from Coco dataset



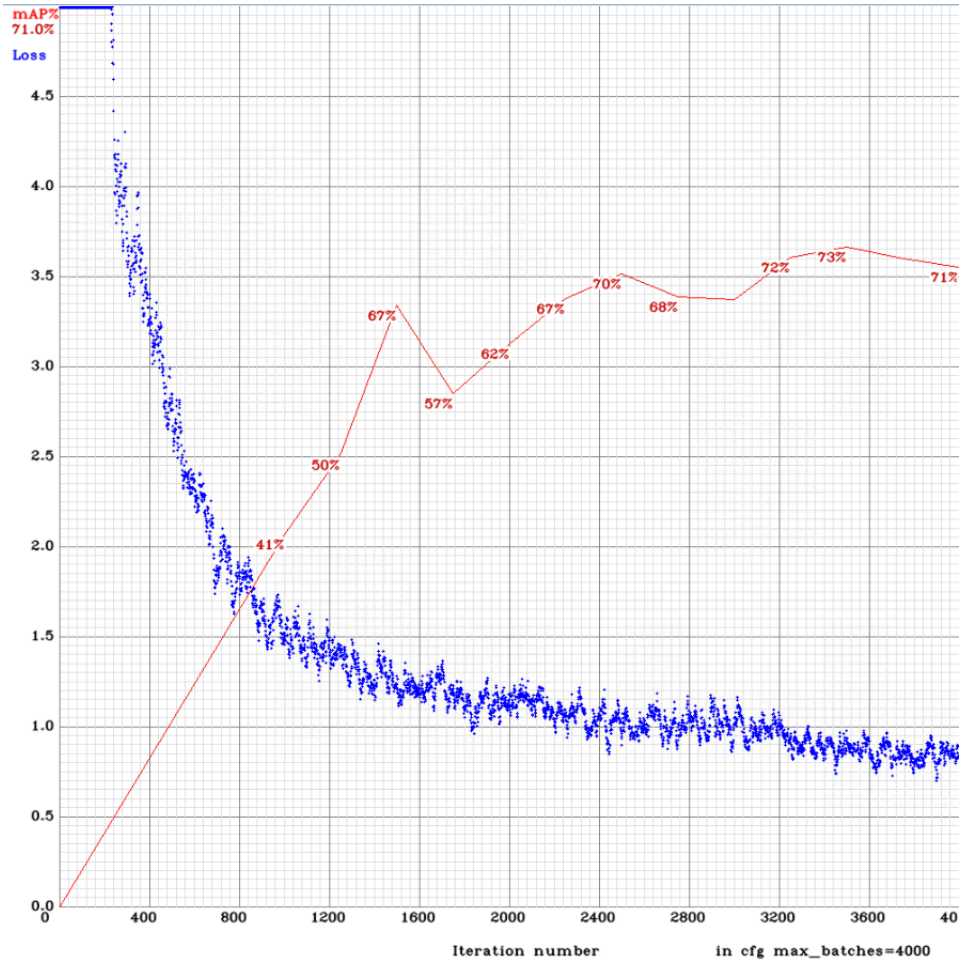
3.9 Training network on sets of various portions of images without any person

Now we will examine the dependency of trained instance performance on proportion of images that does not contain any person ("neg" images opposite is "pos" images). The sets are composet only from Inria dataset.

3.9.1 Training on set of 500 images where every contains some person

See in Fig.3.11, we reached the mAP settled value 57%. what is little better than we saw on training on 500 images from coco. Examples of outputs with ground true boxes are present in Fig.3.14, Fig.3.13.

Fig. 3.9: Grafh of average loss and mAP for 4000 images from Coco dataset

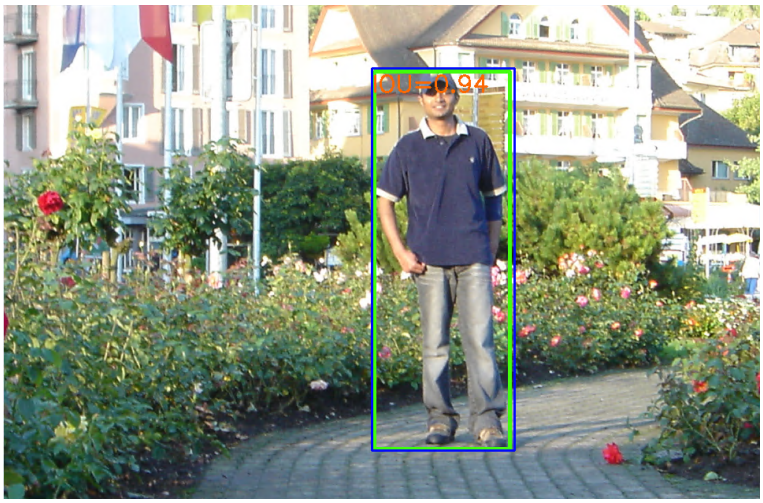


3.9.2 Training on set of 500 images where 50% of them contains some person

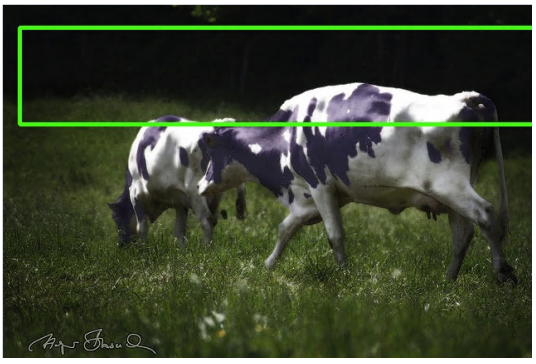
As you can see in Fig.3.12 the loss decreases more rapidly to lowest values. The mAP reaches its average value on 54%, what is 3% less than we saw in previous set. Result with ground true box can be seen in Fig.3.14.

3.10 Conclusion on proportion of images which does not contain any person

After comparing results from 2 previous sets we can see that proportion of "neg" images in set does not seem to have negative impact on performance to some point. It has however bigger influence on descent of loss parameter, so with bigger proportion



(a) True positive prediction

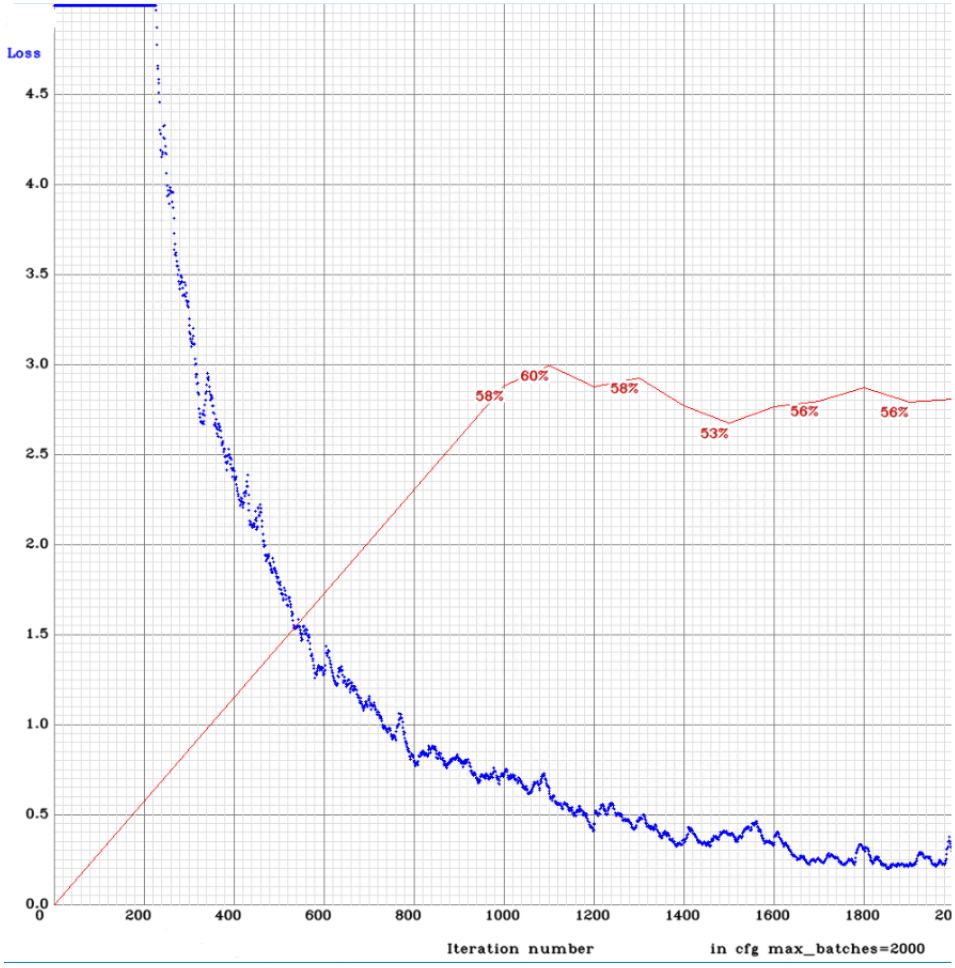


(b) False positive prediction

(c) True negative prediction

Fig. 3.10: Images with grout true boxes and YOLO predictions and IOU (net trained on 4000 images).

Fig. 3.11: Graph for 500 images from Inria where every contains some person.



of "neg" images we can expect diminishing the performance caused by low number of "pos" images and consequent overfitting.

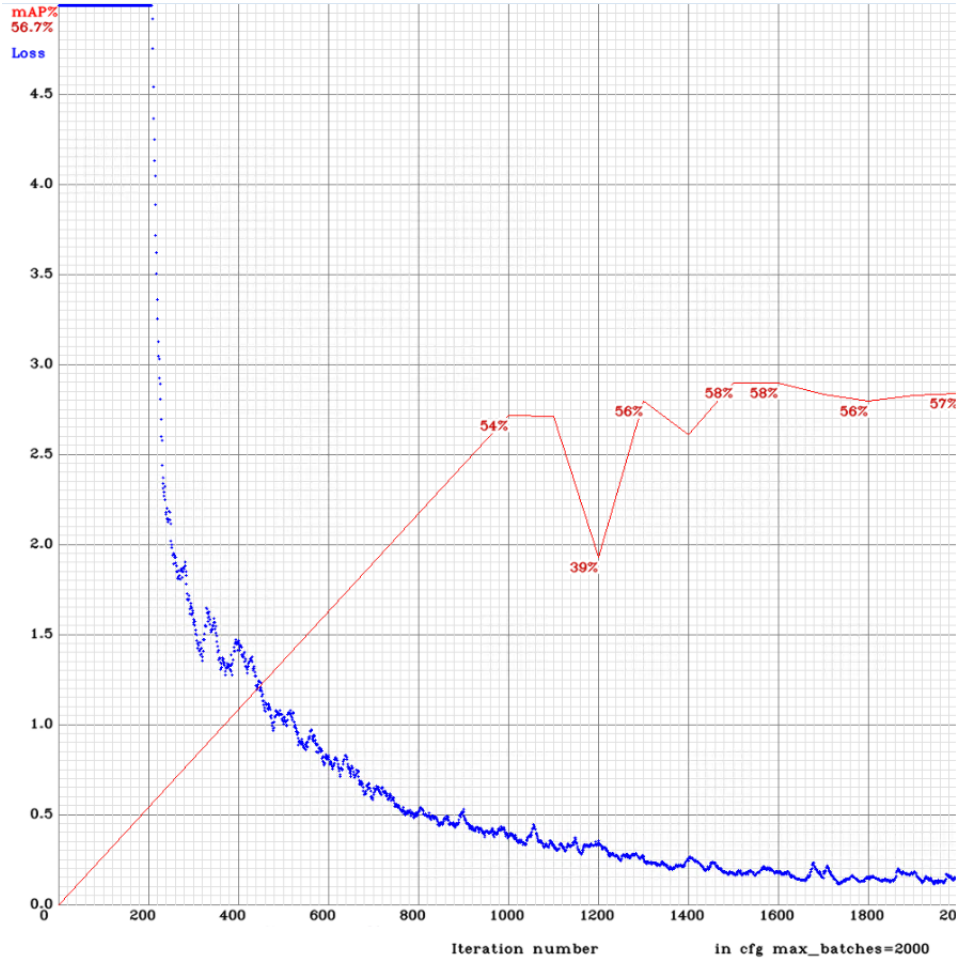
We can see by comparing pictures that the instance trained only on "pos" images predicts more precise bounding boxes, see fig.3.14, but on the other side it makes more false positive and true negative errors for example fig.3.13.

As summary we can say that images that do not contain any person may be favorable, but only then if we have sufficient number of images containing some person.

3.11 Training on mixed dataset

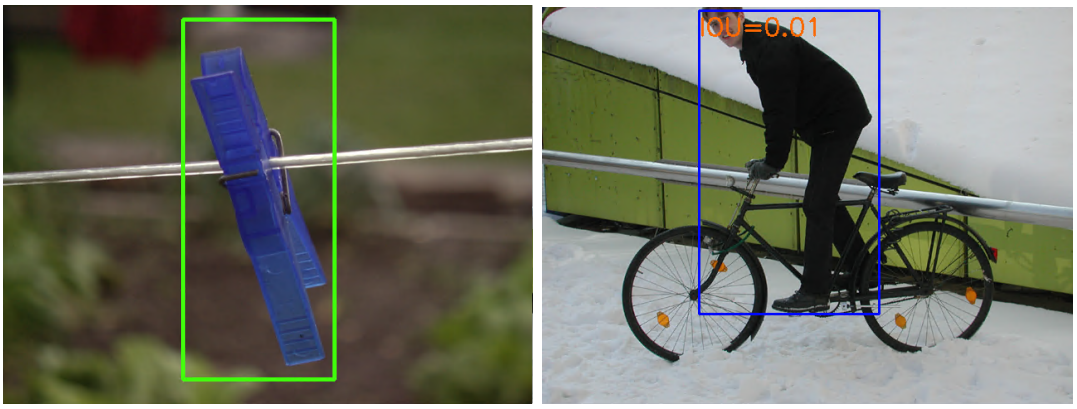
The instance will be now trained on the set that is supposed to be the ideal according to previous findings. It is composed of Inria and Coco datasets in exactly the same ratio as in test set (2:1). Set contains 2000 images where 1330 comes from Inria

Fig. 3.12: Graph for 500 images from Inria where 50% contains any person.



(half of images does not contain any person) and 670 images from coco.

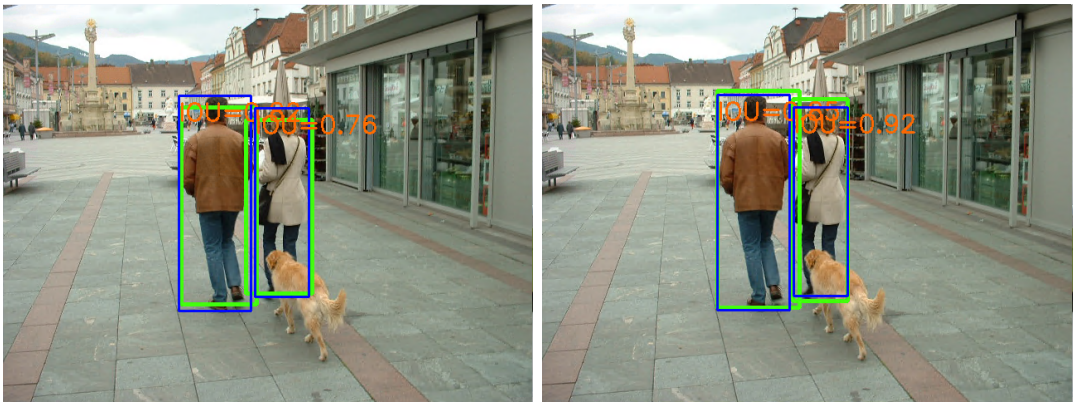
As you can see in 3.15, the results are the best so far. With achieved mAP 69%. All 600 test images with output and groud true boxes can be seen in attached CD.



(a) False positive prediction by net trained on only "pos" images

(b) True negative prediction by net trained on only "pos" images

Fig. 3.13: Errors made by net trained on only "pos" images

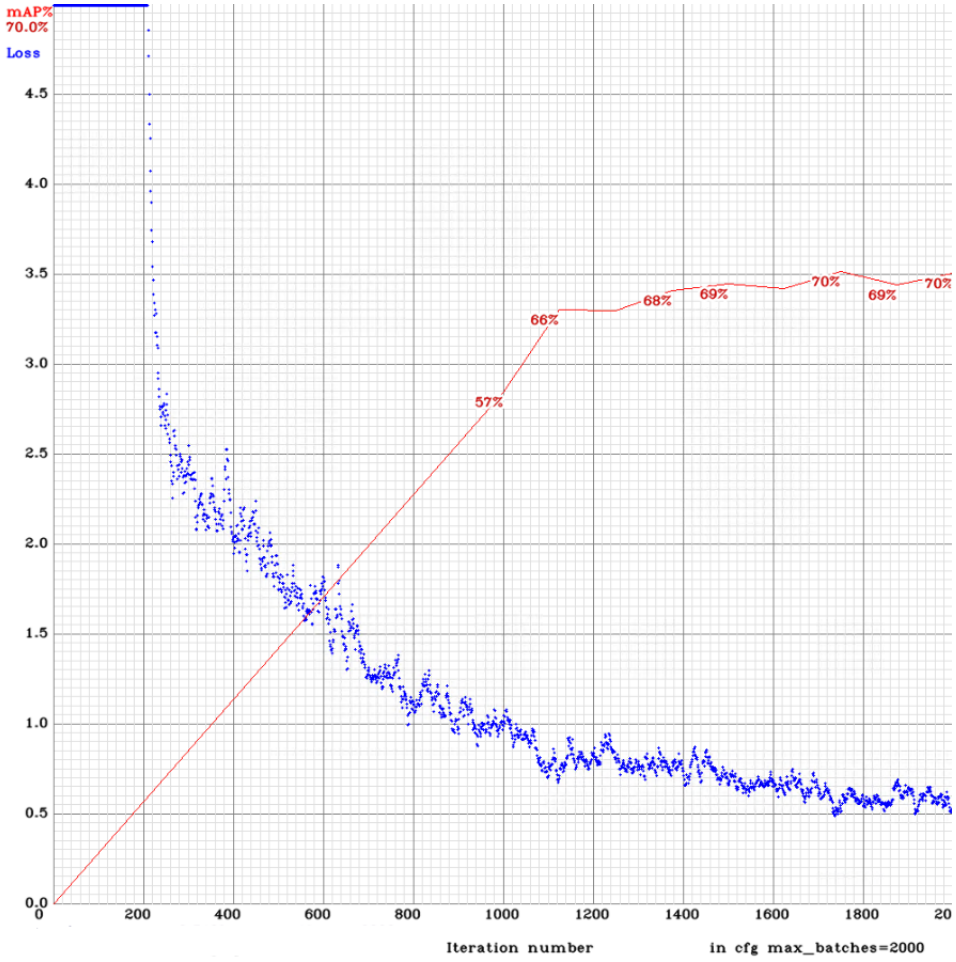


(a) Trained on set of 50% of "pos" images

(b) Trained on set of 100% of "pos" images

Fig. 3.14: Comparison of predictions of nets trained on different proportions of "neg" images

Fig. 3.15: Graph of set with 2000 images with same distribution as test set.



4 Conclusion

The analysis of theoretical part chapter leads to findings how essential the convolutional neural networks are in object detection. One may see that those state-of-art methods are changing almost every year in principles, while improving in performance.

After analyzing the available pedestrians/people datasets, it was realized that many of them has either poor annotation precision such as M-ATCI Rear-view pedestrians datase and Caltech or are too small as Penn-Fudan dataset, but fortunately there are present some datasets that meet our demands in precision, size and also content, for example Coco, Pascal or Inria.

Practical part shows how was proceeded the choice of implementation with training and testing processes, which resources were chosen with their preceding cogitations. As result the YOLO method was chosen as detection method. Then as methods implementation was chosen the original authors implementation with some added improving features, that comes handy in using. These features also speeds up training and testing, due to its ability to use graphical card possessing tensor cores inside. As far as my computer does not posses any useable graphical card neither the tensor-cores, I had to choose some cloud services where I have chosen Google cloud platform. After hardware demands were satisfied, in the following text is in details described the whole process how to launch the implementation from downloading to running the program. 2 python scripts were made to make this process more friendly (scripts generates required files).

Great attention was paid on composing the testing dataset that reflects performance of trained instances. As result Coco and Inria datasets were put together into 600 images where 400 images comes from Inria. By analyzing generated graphs during training process, it leads to conclusion that the number of images used for training only improves performance but just up to some point when it starts to stagnate (approximately 2000). Number of iterations does not appear to be depend on number of training images.

Results of the next experiment shows that adding images without any object (that is being detected) to our training set may only leads to improvement of performance especially in reducing false positive and true negative errors. But one should not forget to have sufficient number of images that contains some object (that is being detected). These results can be seen in pictures that were generated by my python script which put the predicted and ground true bounding boxes with their intersection over union into pictures Fig.3.14.

In the last training process is examined the performance of instance trained on set, that is expected to be ideal according to previous findings. As we can see in

Fig.3.15, the outstanding 69% of mAP(50%treshold) was reached what is 12% more than official results of YOLOv.3 trained and tested on COCO dataset, see results in [31]. The main reason of that perfect result is that only one type of object is being detected, what makes it for detection algorithm easier. Also pedestrian is a type of object that cannot be so easily mistaken with other types of objects.

The main downsides of this thesis may be that the desired number of training processes were not executed. The main reason is lack of computational power with time I possessed for training process.

This bachelors thesis presents a great asset for my knowledge as I gained considerable outlook not only in object detection but also in other machine learning disciplines.

Bibliography

[1] *N. Dalal and B. Triggs. Histograms of oriented gradients for human detectio.* In CVPR, 2005.

[2] *P. Dollar, R. Appel, and W. Kienzle. Crosstalk cascades for frame-rate pedestrian detection.*

[3] *P. Sermanet, K. Kavukcuoglu, S. Chintala, and Y. LeCun. Pedestrian detection with unsupervised multi-stage feature learning.* In CVPR, 2013.

[4] *J. Yan, Z. Lei, L. Wen, and S. Z. Li. The fastest deformable part model for object detection.* In Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on, pages 2497–2504. IEEE, 2014

[5] *M. A. Sadeghi and D. Forsyth. 30hz object detection with dpm v5.* In Computer Vision–ECCV 2014, pages 65–79. Springer, 2014

[6] *K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognitio.* In ECCV, 2014.

[7] *R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation.* In CVPR, 2014.

[8] *A. Krizhevsky, I. Sutskever, and G. Hinton. ImageNet clas-sification with deep convolutional neural network.* In NIPS, 2012.

[9] *J. Sivic and A. Zisserman, "Video google: a text retrieval approach to object matching in videos,"* in ICCV, 2003.

[10] *R. Girshick, "Fast R-CNN,"* in IEEE International Conference on Computer Vision (ICCV), 2015.

[11] *S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks.* arXiv preprint arXiv:1506.01497, 2015.

[12] *J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in IEEE Conference on* Computer Vision and Pattern Recognition (CVPR), 2015.

[13] *C. Szegedy, A. Toshev, and D. Erhan, "Deep neural networks for object detection".* In International Conference on Learning Representations (ICLR), 2015.

[14] *M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional neural networks,* Computer Vision and Pattern Recognition (CVPR), 2015

[15] *K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition,"* Computer Vision and Pattern Recognition (CVPR), 2015

[16] *Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi, "You Only Look Once: Unified, Real-Time Object Detection"* In University of Washington, Allen Institute for AI, Facebook AI Research, 2016.

[17] *Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, Alexander C. Berg, "SSD: Single Shot MultiBox Detector"* In ECCV 2016.

[18] *Tracking and motion cues for rear-view pedestrian detection. Dan Levi and Shai Silberstein.* In IEEE Intelligent Transporation Systems Conference (ITSC) 2015.

[19] *Vision-Based Pedestrian Detection for Rear-View Cameras. Shai Silberstein, Dan Levi, Victoria Kogan and Ran Gazit.* In IEEE Intelligent Vehicles Symposium 2014.

[20] Caltech dataset website - `http://www.vision.caltech.edu/Image_Datasets/CaltechPedestrians`

[21] Inra dataset website - `http://pascal.inrialpes.fr/data/human/`

[22] Penn-Fudan dataset website - `https://www.cis.upenn.edu/~jshi/ped_html/`

[23] GM-ATCI Rear-view pedestrian dataset website - `https://sites.google.com/site/rearviewpeds1/`

[24] Coco dataset website - `http://cocodataset.org/#home`

[25] *Microsoft COCO: Common Objects in Context. Tsung-Yi Lin Michael Maire Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, Piotr Dollár.* In Microsoft 2015.

[26] Daimler Mono Pedestrian Detection Benchmark Dataset - `http://www.gavrila.net/Datasets/Daimler_Pedestrian_Benchmark_D/Daimler_Mono_Ped__Detection_Be/daimler_mono_ped__detection_be.html`

[27] Tsinghua-Daimler CYclist Detection Benchmark Dataset - `http://www.gavrila.net/Datasets/Daimler_Pedestrian_Benchmark_D/Tsinghua-Daimler_Cyclist_Detec/tsinghua-daimler_cyclist_detec.html`

[28] Pedestrian Dataset from Nicta - https://research.csiro.au/data61/automap-datasets-and-code/#pedestrian-dataset

[29] CVC-ADAS dataset - http://adas.cvc.uab.es/elektra/datasets/pedestrian-detection/

[30] *Joseph Redmon, Ali Farhadi: YOLOv3: An Incremental Improvement.* In University of Washington 2018

[31] YOLO website - pjreddie.com/darknet/yolo/

[32] https://github.com/thtrieu/darkflow

[33] Github repository of used project - https://github.com/AlexeyAB/darknet

[34] Caltech to YOLO annotation format converter - github.com/mitmul/caltech-pedestrian-dataset-converter

[35] Coco(.json) to YOLO annotation format converter - http://commecica.com/wp-content/uploads/2018/07/cocotoyolo.jar

List of appendices

A Contend of attached CD

At the end of Thesis is attached the CD, with the following contend.

```
/.....................................................root folder of CD
├── test_set_images_with_gt_and_predicted_boxes_by_weights_trained_on_mixed_set
├── pedestrian_with_predicted_boxes.mp4
├── weights_trained_on_mixed_set.weights
├── weights_trained_on_100_images.weights
├── bachelor_thesis.pdf
└── README.txt
```