BABEŞ BOLYAI UNIVERSITY, CLUJ NAPOCA, ROMÂNIA
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

PEDESTRIAN DETECTION

– MIRPR report –

Team members
Name, specialisation, group, email
Jurj Flaviu Andrei, informatics in english, 934, jurj.andrei12@gmail.com
Maxim Tudor, informatics in english, 934
Moisi Teofana Ionela, informatics in english, 934

2019

Abstract

Statistics show that thousands of people are killed in traffic every year. Looking at the reason of these accidents, it is almost always the lack of attention on the side of the driver. There has been a great deal of interest in recent years in the development of pedestrian detection systems that could help reduce the number and impact of these accidents.

Pedestrian detection represents one of the well-researched domains of object detection alongside face recognition.Consequently, many perspective were checked during its life course and non-intelligent solutions were found unsufficient for the variety of factors which should be taken into consideration. We are going to use and compare two different intelligent methods which will hopefully improve previous research on the topic.

For our intelligent methods, we have used YoloV3. Yolo is an object detector that uses features learned by a deep convolutional neural network to detect an object. Its name stands for 'You only look once', referring to the idea that it evaluates all the necessary actions in one forward step.

Prior detection systems repurposed classifiers or localizers to perform detection. They apply the model to an image at multiple locations and scales. High scoring regions of the image are considered detections.

Yolo applies a single neural network to the full image. The network divides the image into regions and predicts bounding boxes and probabilities for each region. These bounding boxes are weighted by the predicted probabilities. More discussion is found in the following chapters.

Our dataset is taken from 'PASCAL VOC Project', which provides standardised image data sets for object class recognition. We have taken the VOC2012 dataset and adapted such that we consider only the annotations relevant for our problem.

We have trained and evaluated both YoloV3 and YoloV3 Tiny versions of the latest Yolo architecture. By using, the Pascal VOC Challenge metrics for our validation data, we have reached to the following results and the default thresholds:
- an AP (average precision) of 68% for the normal architecture
- an AP (average precision) of 48% for the tiny architecture
More details can be found in the Results Chapter.

Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

1.1 What? Why? How?

- What is the (scientific) problem?
  - The pedestrian detection task represents a special case of object detection being related to computer vision and image processing. From an intelligent algorithm perspective, we can say that it represents a classification problem. From a more detailed perspective we are talking about a subdivision of object detection.
- Why is it important?
  - Pedestrian detection represents one of the core functionalities necessary for an autonomous vehicle. Moreover, the detection must be both reliable and fast taking into consideration all the possible factors which can occur.
- What is your basic approach?
  - Our basic approach resumes to training our own network on a given dataset. We will start with small datasets and basic objects and build on the top of it so that we would be able to detect pedestrians from different kind of images(with or without pedestrians).
  - We will use YOLO object-detection system with its latest version and architectures, while improving our datasets throughout the project's lifecycle

1.2 Paper structure and original contribution(s)

The main contribution of this report is to present an intelligent algorithm for solving the problem of **pedestrian detection**.

The second contribution of this report consists of building an intuitive, easy-to-use and user friendly software application. Our aim is to build an algorithm that will help both car makers for further improvements to the automated driving system and casual drivers with live alerts in the traffic.

The present work contains 9 bibliographical references and is structured in five chapters as follows.

The first chapter is a short introduction in the problem of pedestrian detection.

The second chapter describes the scientific problem and the approaches we used to solve it.

The third chapter contain the main points from some related papers from the bibliography.

The fourth chapter describes our proposed approach as solution to the presented problem.

The fifth chapter, and maybe the most important one, contains the exact details of the developed application, including the method, data description, results and a short discussion.

The last chapter represents the conclusion and it also contains a short list of our future work.

Chapter 2

Scientific Problem

2.1 Problem definition

Object detection has been a trending topic in the recent years due to innovation in intelligent methods, but also in hardware performance. Pedestrian detection represents a canonical sub-problem with diverse applications, the main one being autonomous driving. Despite the extensive research on this particular problem, recent papers still show significant improvements over older ones while still being far from the accuracy of a human.To cite a recent piece of research will evaluated the best methods available: "The results indicate that there is still a ten fold improvement to be made before reaching human performance."
The presented problem is really complex due to its high variety of factors and challenges such as different height, weight, clothing, frequent occlusion betweend pedestrians etc. Consequently, conventional non-intelligent methods constructed in the past were found to be insufficient to solve the problem in a generalized manner, being only very accurate on specific cases and environments. As a result, the trend has shifted towards variations of intelligent algorithms which may generalize more easily the problem while maintaining ideas from the specific implementations.

Since the topic consists mostly in the analysis of visual images, the usage of convolutional neural network with their variations has been a major player in the recent time. The advantages of this method stems from the fact that these types of networks tend to learn by themselves which filters are relevant for the specific problem, eliminating the need to hand-engineer them like in traditional algorithms of image classification. One major disadvantage represents the need for a relatively large training data set which isn't always available.

The inputs for this problem might be of two types: images and short clips, the later one being converted into a set of relevant frames which will be interpreted as normal images. The resolution

among the input data should be kept consistent in order to ensure the correctitude of the evaluation. Otherwise, different methods of rescaling should be used, while maintaining the original details: as little distortion, blur and noise as possible.

The main outputs of the application will be classified images where each pedestrian will be bounded with a box in order to denote its position. Moreover, we may generate secondary textual outputs which will contain only the relevant information found by the intelligent algorithm: the corners coordinates for the boxes.

Table 2.1: The parameters for the standard Yolo training model

Parameter	**Value**
Learning Rate	0.001
Score Threshold	0.5
IOU Threshold	0.5
Input Image Size	416x416
Number of Classes	1

Algorithm 1 Detect Object Predictions

BEGIN
@ Read and Transform Image
@ Divide the image in a SxS grid
@ Go through the **darknet53** extractor to obtain features
@ Send the features to the detection convolutional layers at multiple scales
@ Detect bounding box
@ Calculate various scores:
boxConfidenceScore = Pr(object) * IoU
conditionalClassProbability = Pr(class|object)
classConfidenceScore = Pr(class) * IoU
@ DetectBoxes(threshold, IoU)
@ Apply NonMaxSupression
if detection is over IoU and treshold **then**
  @ Return(detection);
end if
END

Chapter 3

State of art/Related work

The theory of the methods utilised until now in order to solve the given problem.

Answer the following questions for each piece of related work that addresses the same or a similar problem.

- What is their problem and method?
- How is your problem and method different?
- Why is your problem and method better?

Detecting Pedestrians by Learning Shapelet Features

Payam Sabzmeydani and Greg Mori

School of Computing Science Simon Fraser University

- What is their problem and method?
  - They approach the problem of pedestrians detection in still images. They use AdaBoost training algorithm for face and pedestrians classification on the MIT and INRIA datasets. A major drawback is the fixed set of feature descriptors and the problem with defining them is that there could be some discriminative information that is missed by those features . For training, they use three steps: low-level features, shapelet features and the final classifier.
- How is your problem and method different?
  - Our aim is the same: to find a method to detect the pedestrians in the most accurate way. Besides detecting wether there are or not pedestrians, we would like to see how close

they are. We will use TensorFlow and compare the results afterwards and also use different datasets. Another option is to use the AdaBoost training algorithm as well and try to change the training features and final classifier, in order to get better results, bu comparison. They compare their results with HOG-SVM detector of Dalal and Triggs and maybe we could find another detector results to compare with.

- Why is your problem and method better?
  - Detecting how close we are to the pedestrians could be an improvement besides detecting wether there are pedestrians or not.

Pedestrian Detection: A Benchmark

Piotr Dollar, Christian Wojek, Bernt Schiele, Pietro Perona

- What is their problem and method?
  - Pedestrian detection represents a key problem in computer vision and most of the recent important evolution in this field has been realized thanks to the challenging public datasets. They come with the new dataset: Caltech Pedestrians Dataset, which is two times larger than the existing ones. They propose improved performance metrics and they benchmark 7 algorithms. Moreover, they classify the pedestrians by the distance from the camera: near, medium and far.
- How is your problem and method different?
  - Our problem resumes basically to the same thing and we will try to obtain similar results using other methods and algorithms. We are going to use implement a neuronal network using TensorFlow and see how our results will differ.
- Why is your problem and method better?
  - We are going to work on more datasets, so that the images will vary more. We could also try to process short videos, not only images.

Understanding Pedestrian Behavior in Complex Traffic Scenes

Amir Rasouli, Iuliia Kotseruba, and John K. Tsotsos

This piece of research aimed to identify, analyze and present the pedestrian behaviour in traffic from a moving car perspective. Its purpose was to evaluate the crossing patterns, how pedestrians communicate their intention and to find other factors which may influence the pedestrians decisions. Consequently, they create a new dataset composed of short clips presenting possible events of crossing in different conditions. The data was annotated in two manners: bounding boxes representative for pedestrian detection and textual associated with behavioural data about the pedestrians and drivers actions and contextual information such as demographics, ambient conditions and environmental factors. Multiple labelers were used in order to minimize the subjective bias while assessing the data and some actions were removed from the dataset since there wasn't a consensus between the labelers.
The study evaluated many perspectives and found out some interesting remarks:

- the context in which pedestrians are observed plays an important role along their position and head orientation, indicator of crossing intention;
- there are often used explicit means of communications to resolve conflicts in traffic scenes (e.g. handwaves, driver using the led lights)
- various elements in a scene such as width of the street, the presence of a zebra crossing or a traffic signal can greatly influence the pedestrian confidence of crossing, representing a valuable way to predict his further actions
- the driver's dynamic state with respect to the pedestrians influence their behaviour, Time-To-Collision being used as a representative factor

The study seems to suggest that there's also an interrelationship between various contextual elements which influence the pedestrian behaviour as a whole. There are more factors which can be furtherly studied such as environmental conditions (weather, light), social conditions and demographic conditions. Moreover, studies based on surveys should be conducted in order to fully eliminate subjectivity bias in determining and labeling pedestrians' intentions and actions.

How Far are We from Solving Pedestrian Detection?

Shanshan Zhang, Rodrigo Benenson, Mohamed Omran, Jan Hosang and Bernt Schiele

Max Planck Institute for Informatics
SaarbrÃ¼cken, Germany

This piece of research goal was to improve the current best performers intelligent algorithms over the Caltech Dataset on pedestrian detection. In the last years, two categories of methods have been defined and chosen for their performance towards this topic: integral channel feature detector (ICF) and training convolution neural networks (convnets) for these specific types of images.
The study compared previous implementations performance on the mentioned dataset to a proprietary variation of the original **Checkerboards** detector which furtherly is a generalisation of the Integral Channels Feature Detector (ICF). Their implementation called **Rotated Filters** decreased the number of filters used for training and being applied at three different scales.
When it comes to convolution neural networks they compared two of them: AlexNet and VGG16. Both used a subclass of convnets, called R-CNN (region convolution neural networks).In R-CNN the CNN is forced to focus on a single region at a time because that way interference is minimized because it is expected that only a single object of interest will dominate in a given region. The regions in the R-CNN are detected by selective search algorithm followed by resizing so that the regions are of equal size before they are fed to a CNN for classification and bounding box regression. These types of convnets are specifically designed for object detection from which pedestrian detection is part of. Based on their statistics, they found that the convnets have difficulties in evaluating windows surrounding true positivies especially when talking about small objects as dimension in the images. They hypothesise that the performance is a limitation on the tested convnets architecture which should be reconfigured to some which are closer to the ones used for semantic labelling or boundaries estimation tasks which require pixel-accurate output.
We are going to take these suggestions into consideration when designing our convolution neural networks in order to improve their performance on a bigger variety of scenarious.

In order to cite a given work you can use a bib file (see the example) and the *cite* command: [5], [7], [8], [3], [9], [1], [2], [4], [6].

Chapter 4

Proposed approach

Initial Approach

This was our initial brainstorming regarding the Pedestrians Detection problem:

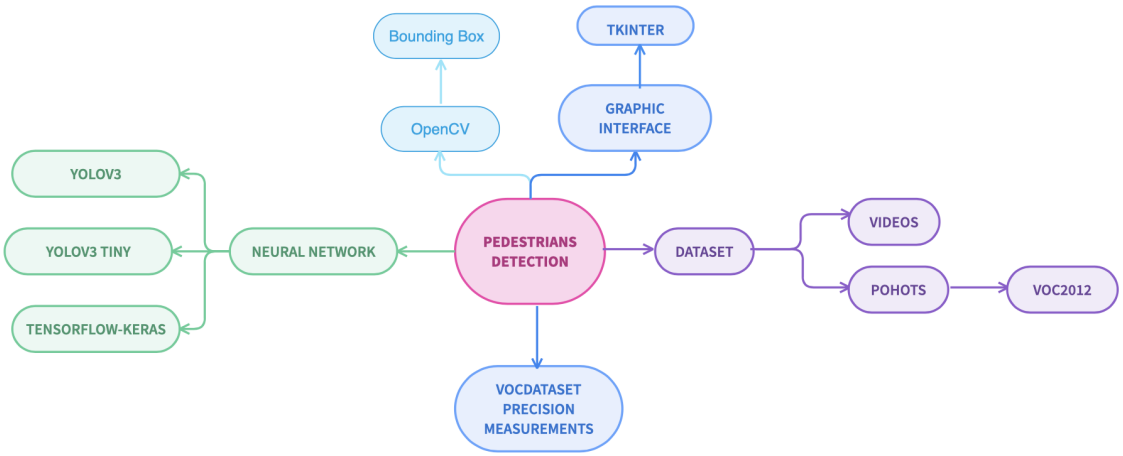

Figure 4.1: MindMap

Further we are going to discuss the idea behind Yolo.
"You Only Look Once" is an algorithm that uses convolutional neural networks for object detection. You only look once, or YOLO, is one of the faster object detection algorithms out there. Though it is not the most accurate object detection algorithm, but it is a very good choice when we need real-time detection, without loss of too much accuracy. In comparison to recognition algorithms, a detection algorithm does not only predict class labels but detects locations of objects as well. So, it not only classifies the image into a category, but it can also detect multiple Objects within an Image. This

Algorithm applies a single Neural network to the Full Image. It means that this network divides the image into regions and predicts bounding boxes and probabilities for each region. These bounding boxes are weighted by the predicted probabilities. But before diving into details, let us see the actual architecture of it:

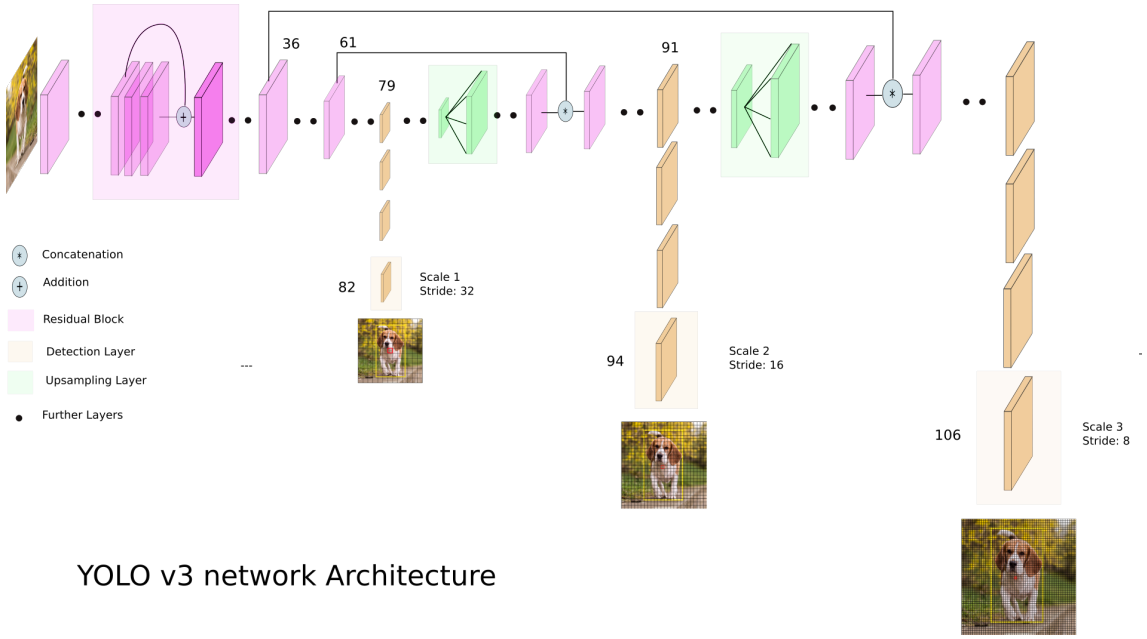

Figure 4.2: Yolo Layers Structure

YOLO makes use of only convolutional layers, making it a fully convolutional network (FCN). In YOLO v3 paper, the authors present new, deeper architecture of feature extractor called Darknet-53. As its name suggests, it contains of 53 convolutional layers, each followed by batch normalization layer and Leaky ReLU activation. No form of pooling is used, and a convolutional layer with stride 2 is used to downsample the feature maps. This helps in preventing loss of low-level features often attributed to pooling.

YOLO is invariant to the size of the input image. However, in practice, we might want to stick to a constant input size due to various problems that only show their heads when we are implementing the algorithm. The network downsamples the image by a factor called the stride of the network. For example, if the stride of the network is 32, then an input image of size 416 x 416 will yield an output of size 13 x 13. Generally, stride of any layer in the network is equal to the factor by which the output of the layer is smaller than the input image to the network.

YOLO can be seen as being composed of two big parts: the feature extractor discussed earlier and the rest of convolutional layers responsible for detecting the object. Moreover, from the schema we can also see that YOLO detects objects at three different scales in order to grasp the differences between

small, medium and large objects.

In order to understand better, we'll try to start from the output of the network. The output is a list of bounding boxes along with the recognized classes. Each bounding box is represented by 6 numbers (pc, bx, by, bh, bw, c).If we expand pc into an 80-dimensional vector (considering 80 classes like for COCO dataset), each bounding box is then represented by 85 numbers.

In YOLO, the prediction is done by using a convolutional layer which uses 1 x 1 convolutions. So, the first thing to notice is our output is a feature map. Since we have used 1 x 1 convolutions, the size of the prediction map is exactly the size of the feature map before it. In YOLO v3, the way you interpret this prediction map is that each cell can predict a fixed number of bounding boxes.

For example if we have (B x (5 + C)) entries in the feature map. B represents the number of bounding boxes each cell can predict. According to the paper, each of these B bounding boxes may specialize in detecting a certain kind of object. Each of the bounding boxes have 5 + C attributes, which describe the center coordinates, the dimensions, the objectness score and C class confidences for each bounding box. YOLO v3 predicts 3 bounding boxes for every cell - one for each anchor.

We expect each cell of the feature map to predict an object through one of it's bounding boxes if the center of the object falls in the receptive field of that cell. This has to do with how YOLO is trained, where only one bounding box is responsible for detecting any given object. First, we must ascertain which of the cells this bounding box belongs to. To do that, we divide the input image into a grid of dimensions equal to that of the final feature map. For example, if the input image is 416 x 416, and stride of the network is 32, the dimensions of the feature map will be 13 x 13. We then divide the input image into 13 x 13 cells. Will have a figure to visualize some things.

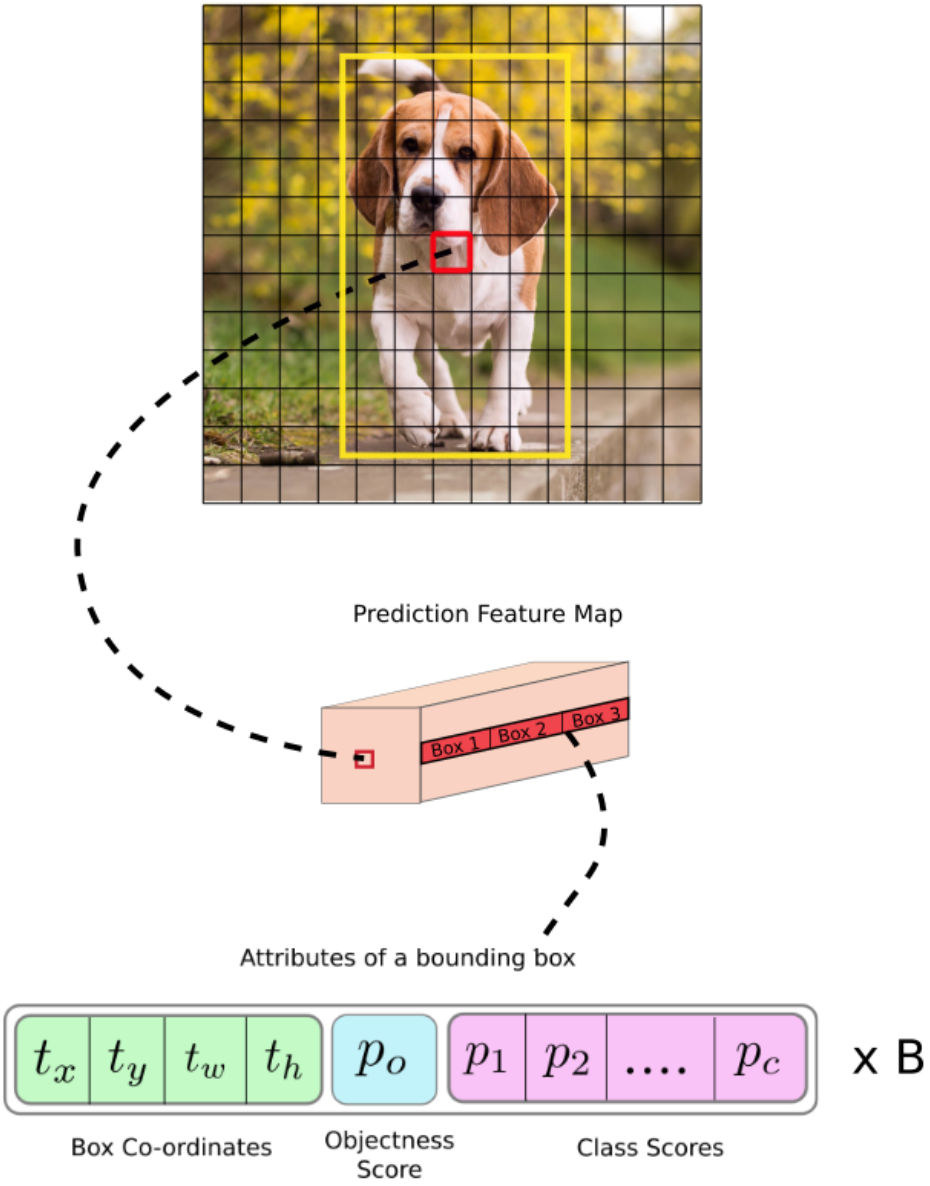

Figure 4.3: Output Strategy Yolo

Now that we are somewhat familiar with Yolo outputs, we still need to discuss one important step (if not the most important one, especially for Yolo) - filtering the boxes based on a threshold.

For an image of size 416 x 416, YOLO predicts ((52 x 52) + (26 x 26) + 13 x 13)) x 3 = 10647 bounding boxes. However, in case of our image above, there's only one object, a dog. So it is clear that we need to filter boxes out, while concatenating others.

First, we filter boxes based on their objectness score. Generally, boxes having scores below a threshold (default 0.5) are ignored. Next, Non-maximum Suppression (NMS) intends to solve the problem of multiple detections of the same image. For example, all the 3 bounding boxes of the red grid cell may detect a box or the adjacent cells may detect the same object, so NMS is used to remove multiple detections. Basically, the NMS algorithm takes care of two things:

- eliminating boxes with a low score (the confidence class is low for that box)
- when several boxes overlap and detect the same object, take one in such a manner that you'll try to keep the best one

A simple idea for the implementation is something along the way
Repeating:

- Select the box with the highest score
- Computes IOU with every other box and remove those which overlap with more than yolo iou threshold (this way we keep the best box, while eliminating good ones which still give kinda the same result)
- iterate back in the loop until no more boxes can be removed

Now you should have some insights with respect to YOLOV3. For more information, search in the Bibliography for the official paper.

Chapter 5

Application (numerical validation)

The initial approach was made on the tiny version of the network, with a learning transfer and a relatively small dataset of approximately 1000 images.

We made two different kind of fine-tune trainings, with the frozen darknet (feature extractor). For each of them, 3 tests were executed, on a 743 images dataset with 235 validation images:

- Only the darknet was copied and the other layers have been randomly initialised. For the first test, the overfitting appeared after 10 epochs and the training was stopped after 75 epochs, while for the other two tests, we used the same dataset and the overfitting appeared after 75 epochs and the training was stopped after 165 epochs. We have tried several ways of modifying the learning rate:
  - static learning rate of 0.001
  - dynamic learning rate started at 0.1 and reducing by a factor of 0.5 after 5 epochs without improvement on validation loss (minimum value allowed: 0.001)
  - dynamic learning rate started at 0.01 and reducing by a facor of 0.75 after 5 epochs without improvement on validation loss (minimum value allowed: 0.001)

  The results were pretty similar for the last two the one starting with the lower learning rate becoming a bit better, while the first one obtained the best validation loss.
- Besides the darknet, we have also copied the basic convolution layers. For the first test, the overfitting appeared after 20 epochs and the training was stopped after 40 epochs. By the time the overfitting appeared, the learning rate was reduced only once. On the other hand, for the other 2 tests, the overfitting appeared quite later, the training stopped after approximately 70

epochs and the learning rate has been reduced 4 times. We have tried several ways of modifying the learning rate:

- dynamic learning rate started at 0.01 and reducing by a factor of 0.5 after 5 epochs without improvement on validation loss
- dynamic learning rate started at 0.001 and reducing by a factor of 0.8 after 5 epochs without improvement on validation loss
- static learning rate of 0.001

The results were pretty similar for the last two which started from the recommended value of 0.001, while the first one obtained the worst validation loss - the starting learning rate was too high initially, causing too many changes in the network.

From this initial approach we managed to obtain some conclusions. Firstly, the learning rate shouldn't have large values in this network causing to oscillate too much without actually decreasing to a good value. We have also seen that it's better to start with the convolutional layers from pre-trained states than from random ones. However, the final difference is not really significant. We have also concluded that the dataset is definitely insufficient for specializing our network to pedestrian detection. While checking empirically random images for detections, we have observed that the network managed to learn only clear examples in which the pedestrian was being a large part of the image and also multiple-object detection in a scene was pretty bad.

For our next approach, we have decided to concentrate in improving our dataset and managing to train also the normal YoloV3 architecture which is a lot more precise (and significantly slower) than the tiny one. We are not going to play anymore with the learning rate and we will keep it at the static recommended value of 0.001.

Improved Approach

In this approach, we have used both YoloV3 and YoloV3 Tiny architectures and we will go with a compare approach between them for different measurements.

We have also changes our dataset to the one offered by 'PASCAL VOC Project', more specifically the latest one available: VOC2012. The standard dataset was used for an object detection contest consisting of 20 different classes. However, we only want to train our network for pedestrian detection, so we adapted the dataset by considering only the relevant annotations. The VOC2012 dataset consists of a set of 10.000 images which isn't really large by modern standards, but it represents a huge improvement from the initial approach.

We will begin by comparing the loss graphs between our architectures. The loss function is a conglomerate taking into account for each set of anchors: - the distance between the object center point and the detected center point - the difference between the groundtruth box dimensions (width and height) and the detected box dimensions - the confidence of the detection and also the class loss (which for our specific problem is not relevant since we have only one class)

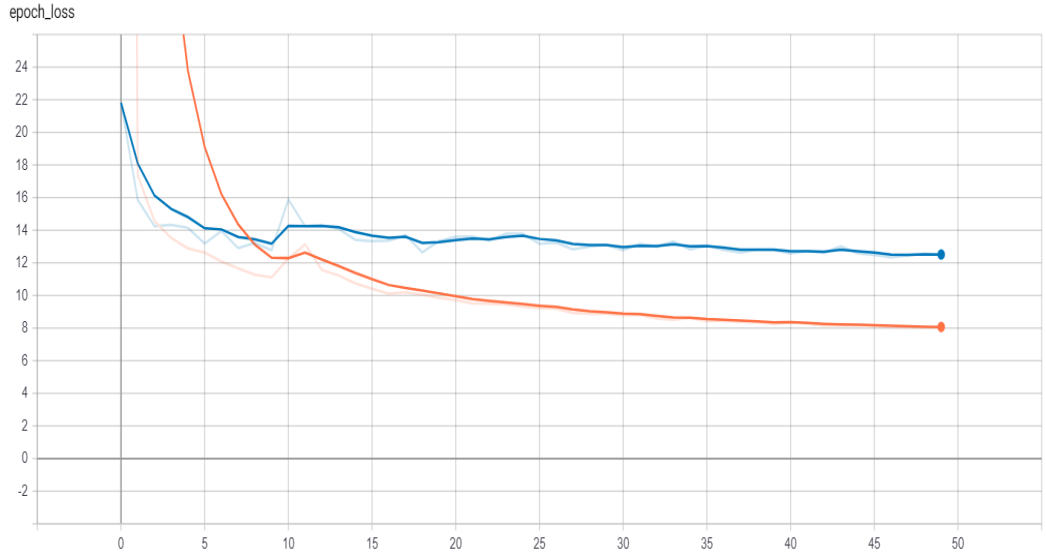

Figure 5.1: Main Loss for Normal Network with Smoothing

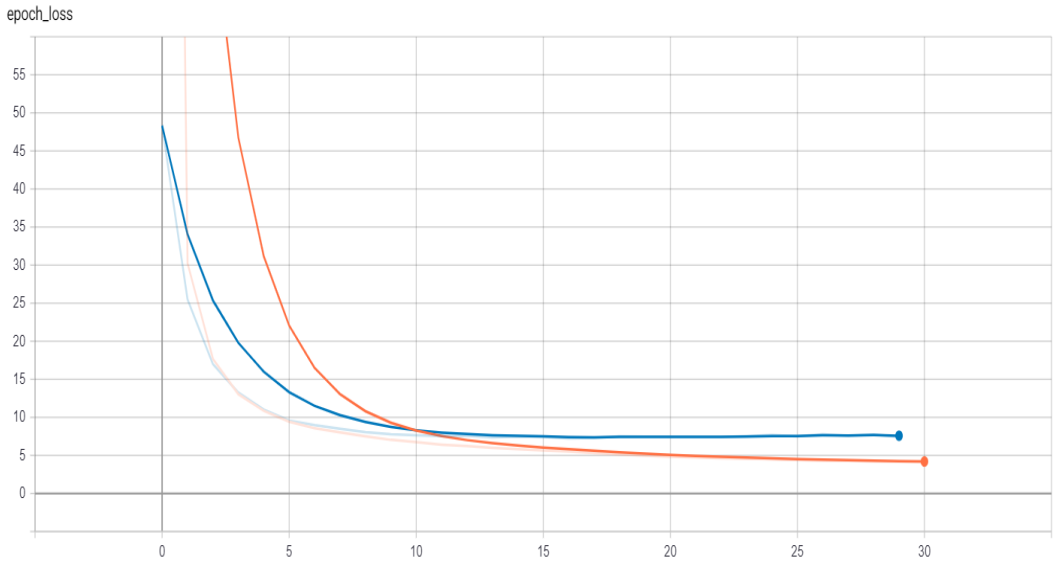

Figure 5.2: Main Loss for Tiny Network with Smoothing

In these examples, we can clearly observe the training loss (orange color) and validation loss (blue color). The graphics have a smoothing effect applied to make them more curvier.

While for the tiny network, we can probably take the conclusion that overfitting has occured after

15-20 epochs, the normal network may have been able to increase further since its apparent validation loss slope is slowly, but surely decreases. Here we can also observe some major differences between the networks. While the tiny network has a more simplistic arhitecture overfitting easily, the normal network has a more sophisticated architecture with many dropout layers allowing it to escape from a local minimum like the one seen in epoch 9. In order to highlight this phenomenon, we will also present the unsmoothed figure the normal architecture. (The tiny one is not necessary, being very similar):

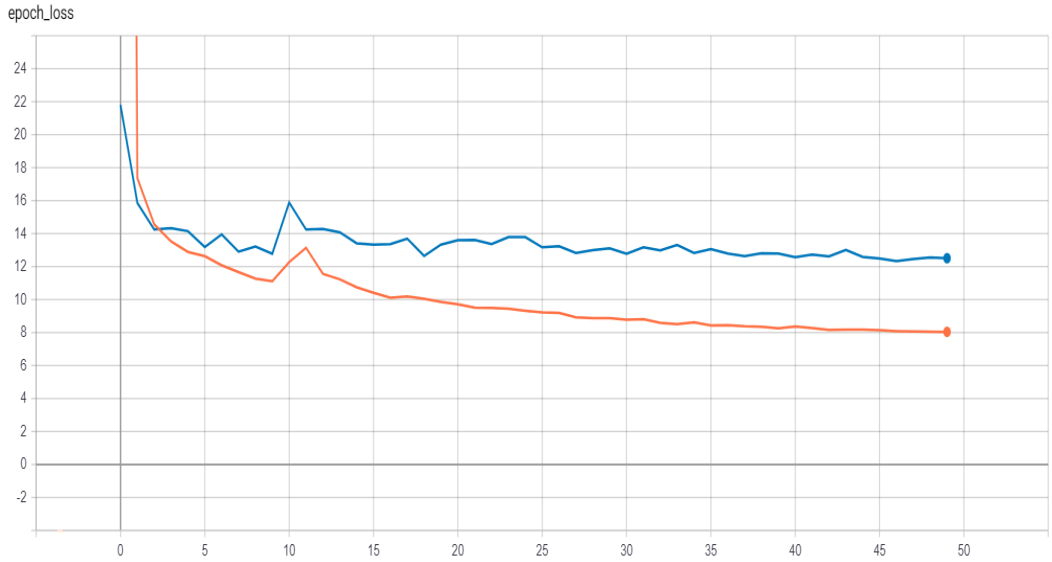

Figure 5.3: Main Loss for Normal Network without Smoothing

In order to fully evaluate, the validation losses, we'll also take into account the losses based on the specific anchors of the architecture. We'll begin first with the Tiny network which offers two sets of anchors: - (81, 82), (135, 169), (344, 319) - (10, 14), (23, 27), (37, 58) We'll choose the smoothed graphs since there isn't relevant information in the other ones

We may observe that the first set of anchors overfits a lot faster, but neither of them has a behaviour which might be able to influence the network to not overfit after the first clear local minimum.

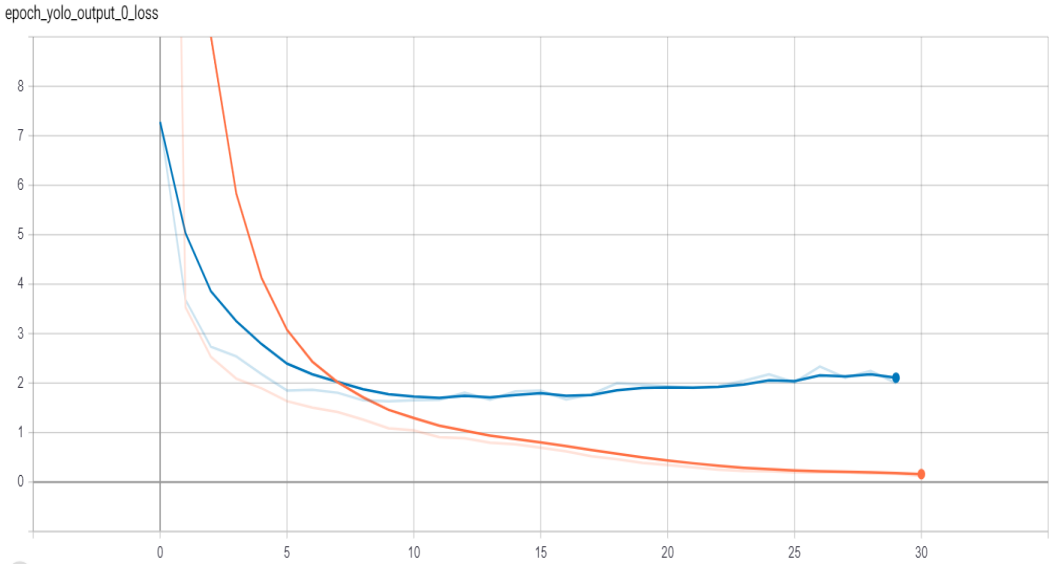

Figure 5.4: Anchor1 Loss for Tiny Network with Smoothing

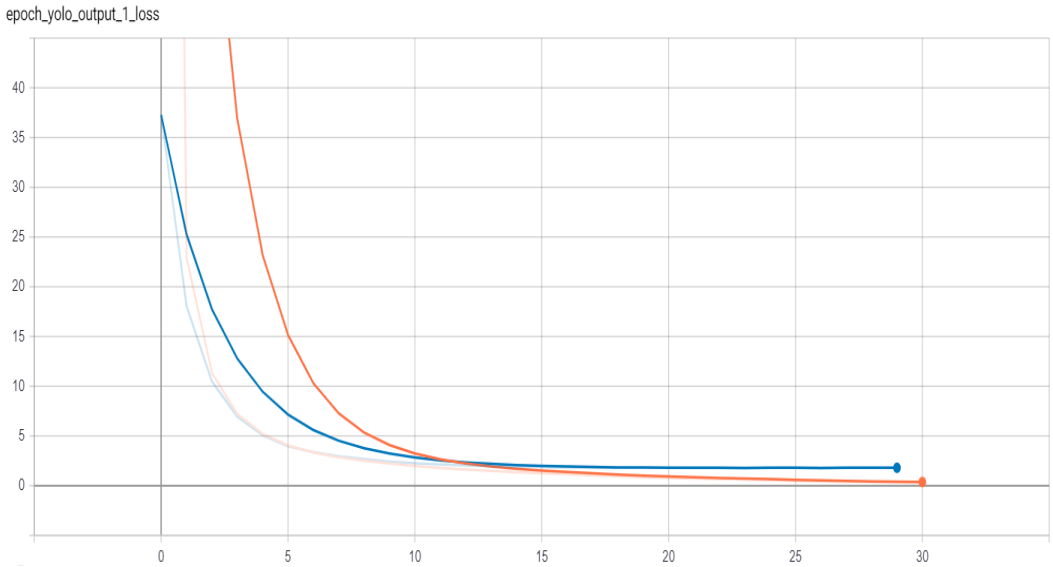

Figure 5.5: Anchor2 Loss for Tiny Network with Smoothing

Now we'll also introduce the anchors for our Normal network which are in three sets: - (116, 90), (156, 198), (373, 326) - (30, 61), (62, 45), (59, 119) - (10, 13), (16, 30), (33, 23)

We'll choose the unsmoothed version in order to compare one anchor (which is more dynamic) with the other two.

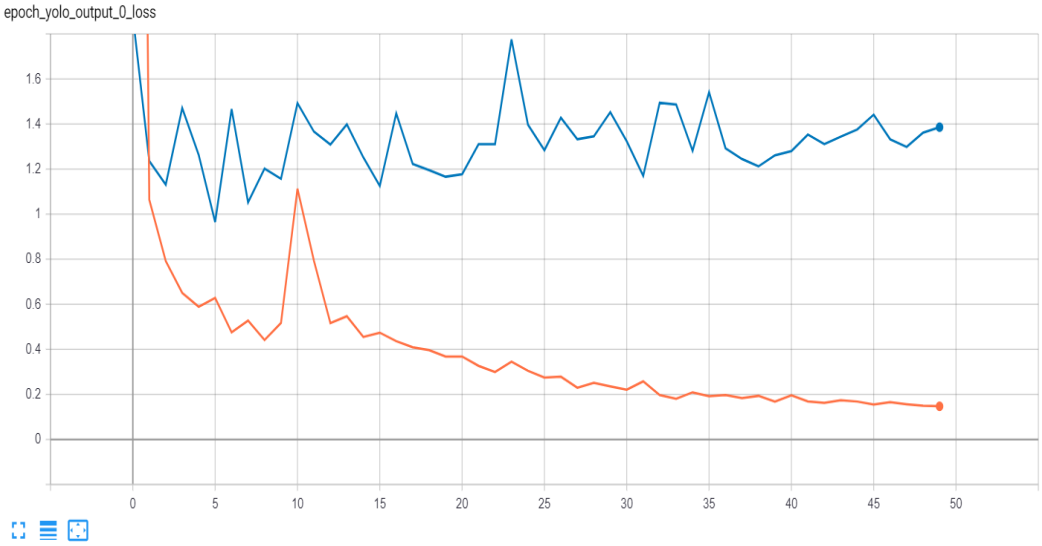

Figure 5.6: Anchor1 Loss for Normal Network without Smoothing

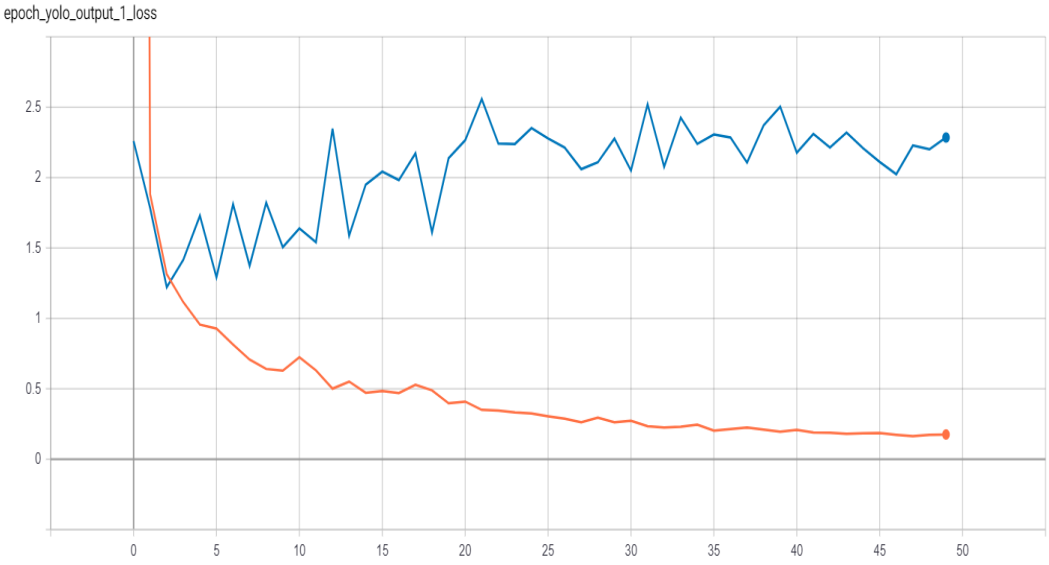

Figure 5.7: Anchor2 Loss for Normal Network without Smoothing

As we may observe, the dynamics of these anchors is a lot higher than for the Tiny network. These are made possible with the improvements brought to YoloV3 such that it can reach out from a local minimum. Observing that in all anchors, we do have a validation loss larger than the minimum attained during the 50 epochs, but taking into consideration the initial graph with the entire validation

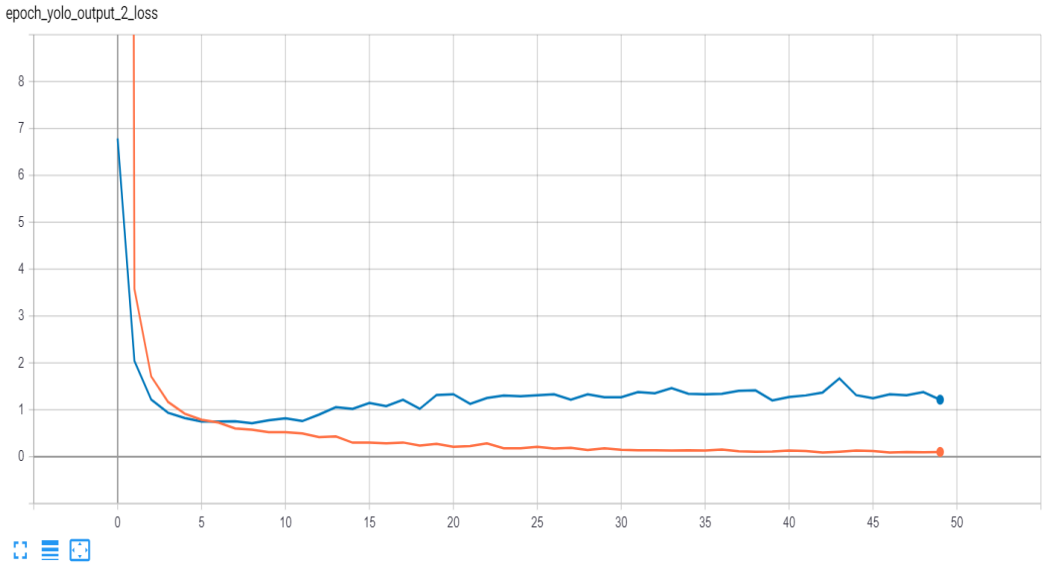

Figure 5.8: Anchor3 Loss for Normal Network without Smoothing

loss where the value was close to the actual minimum during training, we can safely assume that the network was still in the learning process and with more time and processing power it might have reached a different minimum for the validation loss. However, since the training loss is very close to 0 in all examples, the improvement would still have been minimum.

Results

In the following section, we will discuss about the metrics used to quantify our results different from the previous validation loss. For evaluating the differences between our detection boxes and the groundtruths, we have used a variant of the 'PASCAL VOC Challenge' metric which computes a **Precision** x **Recall** curve for each class in the dataset and Average Precision (**AP**).

Some basic concepts used by this metric are:

- True Positive (**TP**): A correct detection. Detection with IOU >= threshold
- False Positive (**FP**): A wrong detection. Detection with IOU < threshold
- False Negative (**FN**): A ground truth not detected

IOU represents 'Intersection Over Union' and is denoted as the division between the overlapping area and the union area.

We have also have the two measurements:

- Precision - the ability of a model to identify only the relevant objects.

$$Precision = \frac{TP}{TP + FP} = \frac{TP}{alldetections}$$

- Recall - the ability of a model to find all the relevant cases (all ground truth bounding boxes)

$$Precision = \frac{TP}{TP + FN} = \frac{TP}{allgroundtruths}$$

For the interpolation, we have used the standard one taking into account each point to compute the Average Precision.

In order to have multiple comparisons, we have varied three parameters across our two architectures:

- **yolo core threshold**
- **yolo iou threshold**
- **metric iou threshold**

Basically the first two, modify the behaviour in which the architecture apply its detection and non-max-suppression algorithm, while the last one evaluates the detections based on a threshold in order to choose whether we accept that detection or not.

We have varied the first two with **0.3** and **0.5** (default) with each combination => four possible detections files. For the metric threshold, we have used three values: **0.3**, **0.5** (default) and **0.75**

In order to not enumerate the results, we'll first discuss how the first two affect the results on the same threshold and then how the metric threshold affects the final values.

Firstly, only decreasing the **yolo iou threshold** from the default 0.5 helped us reduce considerable the number of **FP** detections through our validation dataset, especially on the Tiny architecture. However, for the normal architecture the performance actually decreased since, the number of **FN** increased with almost the same rate.

Secondly, only decreasing the **yolo score threshold** from the default 0.5 helped use reduce the number of **FN**, while increasing a lot more the number **FP** for the Tiny architecture. However, the normal one was objectively better.

We have found out that there is the correct correlation between varying one or the other parameter for our network, but in general if we want to have a balanced version, we should modify them at the same time. We will further compare each network based on two sets of parameters at all thresholds:

Tiny network

yolo iou threshold = 0.5 **yolo score threshold** = 0.5			**yolo iou threshold** = 0.3 **yolo score threshold** = 0.3		
metric iou threshold = 0.3	**metric iou threshold** = 0.5	**metric iou threshold** = 0.75	**metric iou threshold** = 0.3	**metric iou threshold** = 0.5	**metric iou threshold** = 0.75
AP = 50.83%	**AP** = 48.24%	**AP** = 20.07%	**AP** = 60.05%	**AP** = 56.48%	**AP** = 22.12%
TP = 2888	**TP** = 2801	**TP** = 1777	**TP** = 3179	**TP** = 3065	**TP** = 1857
FP = 1220	**FP** = 1307	**FP** = 2331	**FP** = 591	**FP** = 705	**FP** = 1913
FN = 2222	**FN** = 2309	**FN** = 3333	**FN** = 1931	**FN** = 2045	**FN** = 3253

As we may observe, there is constant improvement between the default threshold and the lowered ones. While decreasing the metric threshold down from the default value doesn't influence the results too much, increasing definitely tells us that the network is not even close to detect boxes very close to the groundtruth ones.

Normal network

yolo iou threshold = 0.5 **yolo score threshold** = 0.5			**yolo iou threshold** = 0.3 **yolo score threshold** = 0.3		
metric iou threshold = 0.3	**metric iou threshold** = 0.5	**metric iou threshold** = 0.75	**metric iou threshold** = 0.3	**metric iou threshold** = 0.5	**metric iou threshold** = 0.75
AP = 69.59%	**AP** = 67.35%	**AP** = 47.42%	**AP** = 70.90%	**AP** = 68.29%	**AP** = 47.86%
TP = 3636	**TP** = 3558	**TP** = 2898	**TP** = 3695	**TP** = 3600	**TP** = 2914
FP = 534	**FP** = 612	**FP** = 1272	**FP** = 494	**FP** = 589	**FP** = 1275
FN = 1474	**FN** = 1552	**FN** = 2212	**FN** = 1415	**FN** = 1510	**FN** = 2196

The main idea from the tiny architecture remains true in the normal architecture also. However, there is a clear difference in precision between those two, especially for high values of the metric threshold.

Based on the data obtained, our general expectations are reflected in the results. Modifying only one of the network parameters usually causes a opposite relation between what happens with **TP** and **FP**, while decreasing them both proportionally generally improves the results a bit, but not significantly enough to justify the decrease in accepted confidence.

We'll end the chapter graphs generated across our metrics results which demonstrate the Precision x Recall curve relation. We'll begin with two for the Tiny architecture, choosing the variant with lower threshold for the network.

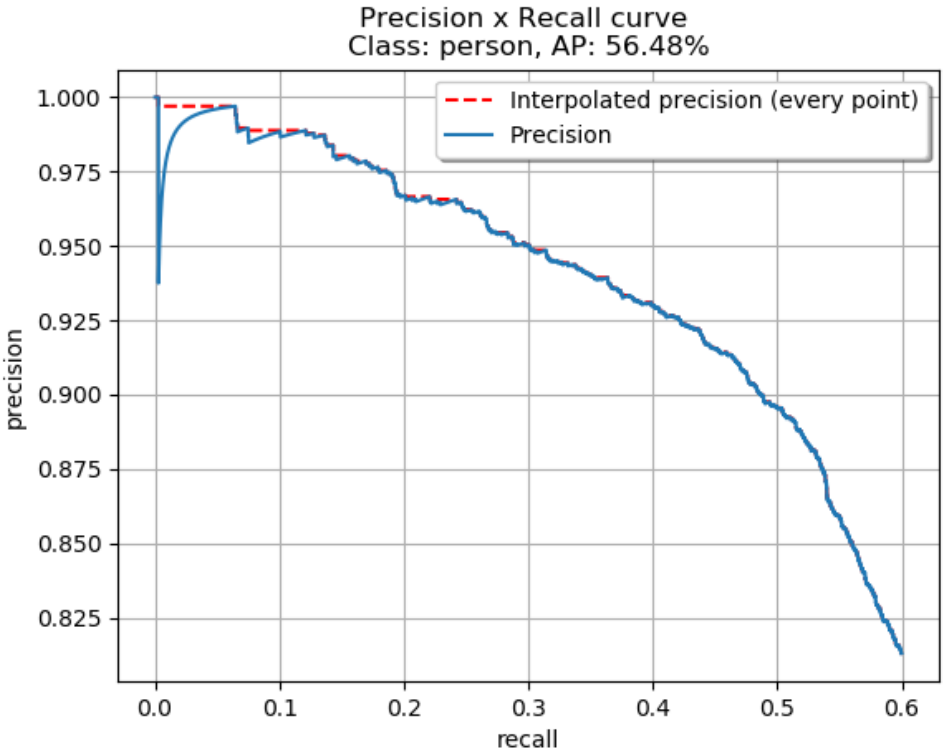

Figure 5.9: Tiny network, yolo iou=0.3, yolo score=0.3, metric iou=0.5

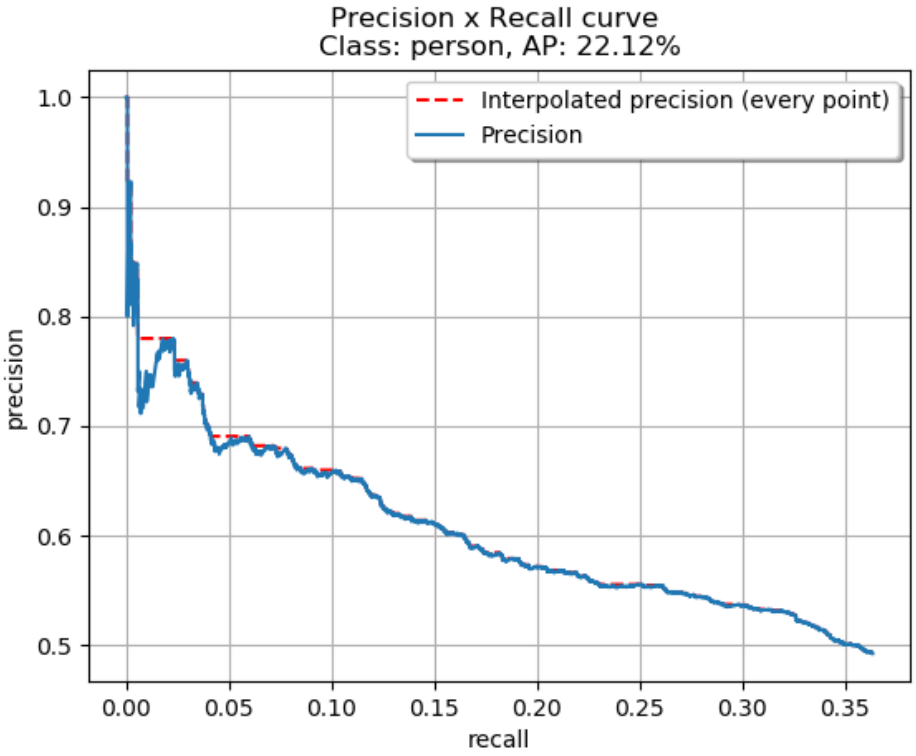

Figure 5.10: Tiny network, yolo iou=0.3, yolo score=0.3, metric iou=0.75

Now we'll present couple of graphs for the normal architecture.

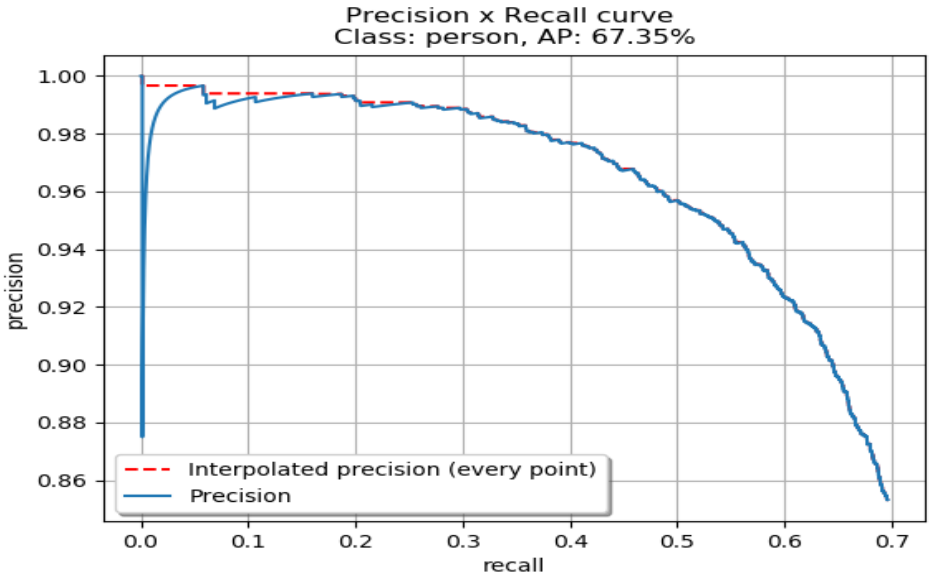

Figure 5.11: Normal network, yolo iou=0.5, yolo score=0.5, metric iou=0.5

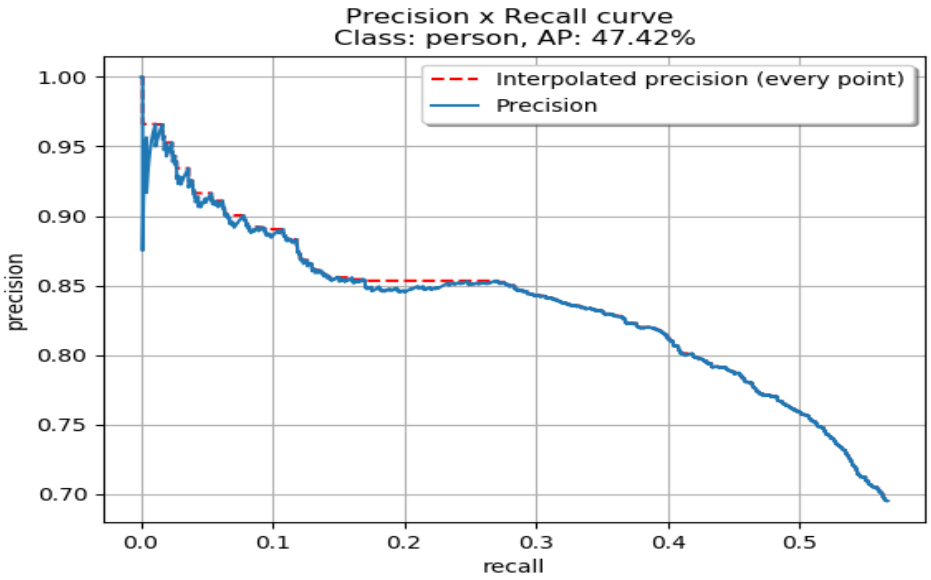

Figure 5.12: Normal network, yolo iou=0.5, yolo score=0.5, metric iou=0.75

If we were to compare them, it can be easily seen that the normal architecture has a more smoother graph, a lot closer to a parabola, being definitely the better architecture. However, for the high threshold, we are still far from an ideal graph with both architectures.

Chapter 6

Conclusion and future work

Our paper attempted to offer a clear and easy point of view from the perspective of pedestrian detection. Our results strength lie in the use of YOLO which is an acknowledged powerful detector for object detection. We have used its last version in order to have the latest improvements when talking about object detection. We have illustrated that it is possible to train a detector on a relatively small dataset through transfer learning and fine tuning.

However, the final results are far from what the state of the art would require and I would further enumerate some of the reasons.

- Firstly, we haven't used any sophisticated techniques for our image resizing such that it will become the standard dimensions required by YOLO. Consequently, this step may include many things to our input image such as: noise, blur, invalid aspect-ratio, artifacts etc. We strongly believe that this is the first step in improving the current algorithm flow.

- Secondly, we haven't used a specialized dataset for our problem. Our dataset was relatively small compared with the number of modifiable parameters in the network (especially for the normal architecture) and the dataset didn't target pedestrian as the main subject. Consequently, there are surely a lot of edge cases which weren't found in the dataset.

- Thirdly, as an addition to improving the actual dataset, some sort of data augmentation should be used in order to extend even more our dataset spectrum.

- Lastly, we may discuss more complicated ideas such as modifying Yolo input size to accept larger images (which usually means more details) and possibly to change the standard aspect ratio of the image which currently is 1:1, while all cameras are 4:3 or 16:9.

Bibliography

[1] Piotr Dollár, Christian Wojek, Bernt Schiele, and Pietro Perona. Pedestrian detection: A benchmark. 2009.

[2] Piotr Dollar, Christian Wojek, Bernt Schiele, and Pietro Perona. Pedestrian detection: An evaluation of the state of the art. *IEEE transactions on pattern analysis and machine intelligence*, 34(4):743–761, 2011.

[3] Uwe Franke, Dariu Gavrila, Axel Gern, Steffen Görzig, Reinhard Janssen, Frank Paetzold, and Christian Wöhler. From door to doorâprinciples and applications of computer vision for driver assistant systems. In *Intelligent Vehicle Technologies*, pages 131–188. Elsevier, 2001.

[4] David Geronimo, Antonio M Lopez, Angel D Sappa, and Thorsten Graf. Survey of pedestrian detection for advanced driver assistance systems. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (7):1239–1258, 2009.

[5] Brian Paden, Michal Čáp, Sze Zheng Yong, Dmitry Yershov, and Emilio Frazzoli. A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Transactions on intelligent vehicles*, 1(1):33–55, 2016.

[6] Amir Rasouli, Iuliia Kotseruba, and John K Tsotsos. Understanding pedestrian behavior in complex traffic scenes. *IEEE Transactions on Intelligent Vehicles*, 3(1):61–70, 2017.

[7] Amir Rasouli and John K Tsotsos. Autonomous vehicles that interact with pedestrians: A survey of theory and practice. *IEEE Transactions on Intelligent Transportation Systems*, 2019.

[8] Payam Sabzmeydani and Greg Mori. Detecting pedestrians by learning shapelet features. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007.

[9] Shanshan Zhang, Rodrigo Benenson, Mohamed Omran, Jan Hosang, and Bernt Schiele. How far are we from solving pedestrian detection? In *Proceedings of the iEEE conference on computer vision and pattern recognition*, pages 1259–1267, 2016.